FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Noise Map

Eine kartografische Visualisierung für Daten erhoben durch die Noise-O-Meter Applikation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Stefan Paula
Matrikelnummer 0553494

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Mag. Dr. Horst Eidenberger

Wien, 30. Dezember 2014

Stefan Paula	Horst Eidenberger

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Noise Map

A cartographic visualization for data captured by the Noise-O-Meter application

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics and Visual Computing

by

Stefan Paula
Registration Number 0553494

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Mag. Dr. Horst Eidenberger

Vienna, 30th December, 2014

Stefan Paula	Horst Eidenberger

Erklärung zur Verfassung der Arbeit

Stefan Paula

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. Dezember 2014

Stefan Paula

Danksagung

Mein Dank gilt insbesondere meinen Eltern Christine und Michael Paula, die mir durch ihren Rückhalt und ihre finanzielle Unterstützung mein Studium ermöglicht haben. Besondere Motivation und Antrieb habe ich durch das Engagement meiner Lebensgefährtin Andrea erhalten, die mich zu jeder Zeit gestärkt und mit ihren Ideen und Korrekturen einen wertvollen Beitrag zu dieser Arbeit geleistet hat. Allen Teilnehmern an der Evaluation der Noise Map danke ich für ihre tolle Mitarbeit, ihre Lärmmessungen, ihre Ideen und nicht zuletzt die Zeit die sie investiert haben. Meinen Großeltern Elfriede und Robert Paula gebührt großer Dank für ihre wertvollen Korrekturen und das Interesse an meiner Arbeit. Meinem Betreuer, Ao.Univ.Prof. Mag. Dr. Horst Eidenberger möchte ich für die tolle Unterstützung sowie die Möglichkeit an diesem Projekt arbeiten zu dürfen danken.

Acknowledgements

I wish to express my sincere thanks to my parents, Christine and Michael Paula, who made my studies possible through their continuous support as well as their financial backing. A lot of motivation and drive came from my partner in life, Andrea, who has strengthened me at any time and made very valuable contributions to this work with her ideas and corrections. Furthermore I'd like to thank all evaluation participants for their active cooperation, their useful input and especially the time they have invested. I am also grateful to my grandparents Elfriede and Robert Paula for their vital corrections and their interest in my work. Last but not least I'd like to thank my supervisor, Ao.Univ.Prof. Mag. Dr. Horst Eidenberger, for the opportunity to work on this project as well as the great support throughout the whole work.

Kurzfassung

Die vorliegende Diplomarbeit nutzt die Möglichkeiten der Informatik um Lärmbelastung im Alltag sichtbar zu machen. Im Mittelpunkt der Arbeit steht ein Prototyp, welcher es ermöglicht, die individuelle Exposition aufzuzeichnen und auf einer Karte nachzuvollziehen. Dieser umfasst eine mobile Sensorkomponente (Noise-O-Meter), einen zentralen Server sowie eine kartografische Visualisierung (Noise Map) zur Präsentation. Die Noise Map ist eine Web Map, welche Lärmmessungen anhand grafischer Repräsentationen darstellt und Interaktionsmöglichkeiten zu deren Untersuchung bietet.

Durch die Umsetzung dieses Prototyps wird die Realisierbarkeit einer durch Individuen befüllbaren Lärmkarte (Participatory Sensing) bewiesen. Es wird der Frage nachgegangen, ob mit Hilfe dieses Konzepts die Limitationen herkömmlicher Lärmvisualisierungen überwunden werden können.

In einem iterativen Prozess wurde mittels umfangreicher Literaturrecherche sowie einer Analyse inhaltlich verwandter Projekte ein Softwareentwurf erstellt auf dessen Basis die Implementierung des Prototyps erfolgte. Diese umfasst Erweiterungen an der Noise-O-Meter Android Applikation, Entwurf und Entwicklung einer Java Web Applikation mit standardisierter REST Schnittstelle sowie Entwurf und Entwicklung der Noise Map. Wichtige Verbesserungen bezüglich Bedienbarkeit und Lesbarkeit konnten durch eine ausführliche Evaluation erzielt werden.

Der realisierte Prototyp zeigt, dass Lärmvisualisierung aus der Fußgängerperspektive über Participatory Sensing möglich ist und das Potenzial birgt, individuelle Betroffenheit sichtbar zu machen.

Abstract

This thesis utilizes the possibilities of informatics in order to shed light on noise pollution in everyday life. The main focus is on a prototype which enables people to record and relate to their individual noise exposure. It consists of a mobile sensor component (Noise-O-Meter), a central server component as well as a cartographic visualization (Noise Map). The Noise Map is a web map which displays noise measurements and offers interaction for data exploration.

The realization of this prototype proves the feasibility of a noise map filled with data collected by individuals (participatory sensing). Thereby the project examines the question if the limitations of conventional noise maps can be overcome by this concept.

An iterative process of extensive literature research as well as analysis of related projects led to a software concept. Based on this concept the prototype consisting of three components was realized. Those components are a sensor component for data collection, a server component for data handling with a standardized API and a cartographic visualization, the noise map, in order to present the data. System tests and an evaluation in which participants collected noise measurements with their smartphones and finally explored them on the map led to improvements in usability as well as readability.

The realized prototype proves that noise visualization from the perspective of pedestrians by participatory sensing is feasible and has the potential to visualize individual noise exposure.

Inhaltsverzeichnis

Kurzfassung **xi**

Abstract **xiii**

Inhaltsverzeichnis **xv**

Abbildungsverzeichnis **xvii**

Tabellenverzeichnis **xix**

1 Einleitung **1**
1.1 Motivation . . . 2
1.2 Aufgabenstellung . . . 3
1.3 Methoden . . . 5
1.4 Ausblick . . . 6

2 Theoretische Grundlagen **7**
2.1 Noise-O-Meter . . . 7
2.1.1 Lärmmessungen . . . 7
2.1.2 Participatory Sensing . . . 9
2.1.3 Android Standortbestimmung . . . 11
2.2 Java Web Applikation . . . 14
2.2.1 Representational State Transfer (REST) . . . 14
2.3 Noise Map . . . 15
2.3.1 Informationsvisualisierung . . . 15
2.3.2 Geovisualisierung . . . 16
2.3.3 Web Maps . . . 18
2.3.4 Lärmkarten . . . 25
2.4 Stand der Technik . . . 26
2.4.1 Straßenlärmkartierung 2007 . . . 26
2.4.2 NoiseSpy . . . 27
2.4.3 NoiseTube . . . 28
2.4.4 Ear-Phone . . . 29
2.4.5 Da_sense . . . 29

3 Implementierung **31**
3.1 Entwurf . . . 31
3.1.1 Stakeholder . . . 31
3.1.2 Anforderungen . . . 32
3.1.3 Anwendungsfälle . . . 32
3.1.4 Architektur und Entwurfsmuster . . . 37
3.2 Noise-O-Meter . . . 38
3.2.1 Identifikation . . . 39
3.2.2 Standortbestimmung . . . 39
3.2.3 Datenübertragung . . . 40
3.3 Java Web Applikation . . . 41
3.3.1 REST Schnittstelle . . . 42
3.3.2 Datenzugriffsschicht . . . 45
3.3.3 Security-Layer . . . 47
3.3.4 Technologie . . . 48
3.4 Noise Map . . . 48
3.4.1 AngularJS Web Applikation . . . 49
3.4.2 Kartografische Visualisierung . . . 50
3.4.3 Technologie . . . 58

4 Evaluation **61**
4.1 Zielsetzung . . . 61
4.2 Ausgangslage . . . 62
4.3 Durchführung . . . 64
4.4 Resultate . . . 65
4.5 Schlussfolgerungen . . . 70

5 Fazit und Ausblick **73**

Literaturverzeichnis **77**

Abkürzungen **81**

Abbildungsverzeichnis

1.1 Farbskala und Diagramme in der Noise-O-Meter Applikation 2
1.2 Die drei Säulen der Noise Map 4
1.3 Eingesetztes Vorgehensmodell in Entwurf und Entwicklung 5

2.1 Noise-O-Meter an unterschiedlichen Punkten der Stadt 9
2.2 Typische Architektur in einem Participatory-Sensing-Projekt 10
2.3 Standortbestimmung über GPS, WiFi und das Telefon-Netz 11
2.4 Vergleich der beiden Android-Location-Frameworks 12
2.5 Latitude phi ϕ und Longitude lambda λ 13
2.6 Elemente der Informationsvisualisierung nach Yi et al. 15
2.7 Zwei Beispiele für Geovisualisierungen 17
2.8 Interaktionstypen in Geovisualisierungen nach Crampton 18
2.9 Historische Entwicklung der Web Maps und wichtige Technologien nach Neumann 19
2.10 Klassifizierung von Web Apps 20
2.11 Schichtenmodell moderner Web Maps 21
2.12 Map Kacheln und Marker 23
2.13 WC Anlagen im Wiener Rathauspark in GeoJSON und als Marker 24
2.14 Fußgängerzone auf der Mariahilfer Straße in GeoJSON und als Polygon . . . 24
2.15 Screenshot der Straßenlärmkartierung 2007 des BMLFUW 26
2.16 Screenshot der NoiseSpy Map und der NoiseSpy Google Earth Darstellung . . 27
2.17 Screenshot der NoiseTube Web Map und der Google Earth Darstellung . . . 28
2.18 Screenshot der EarPhone Map mit Daten von vier (li.) und sechs Personen (re.) 29
2.19 Screenshot der Lärmkarte da_sense 30

3.1 Anwendungsfälle im Prototyp 34
3.2 Client Server Architektur 37
3.3 Schichtenarchitektur 38
3.4 Lifecycle-Diagramm der Standortbestimmung 40
3.5 Lifecycle-Diagramm der Datenübertragung 41
3.6 Objektmodell der identifizierten Ressourcen 43
3.7 HTTP Kommunikation im Prototyp 44
3.8 Datenbank ER-Diagramm 45

3.9 Klassendiagramm der Data Access Objects 47
3.10 MVC in AngularJS 49
3.11 Noise Map Wireframe 50
3.12 Die sieben Schritte der Datenvisualisierung 51
3.13 Statistische Auswertung an Hand des Liniendiagramms im Prototyp 53
3.14 Statistische Auswertung an Hand des Kuchendiagramms im Prototyp 54
3.15 Icons der Noise Map 55
3.16 Noise Map mit Clustern 56
3.17 Pusher in der Noise Map 57
3.18 Noise Map Prototyp 60

4.1 Darstellung der unterschiedlichen Assoziationen der Probanden 66
4.2 Zeigt Cluster (li.) und eine Einzelmessung (re.) 66
4.3 Bewertung der grafischen Animation durch die Probanden 67
4.4 Eingesetzte Navigationsmöglichkeiten nach Häufigkeit geordnet 67
4.5 Prototyp der Noise Map, welcher für die Evaluation eingesetzt wurde 68
4.6 Mängel an Lesbarkeit und Bedienbarkeit bei der Anzeige für Detailinformationen sowie den Statistik Werkzeugen 68
4.7 Zeigt die bevorzugte Darstellungsvariante der Probanden 69
4.8 Beurteilung des Energieverbrauchs durch die Tester 70

Tabellenverzeichnis

2.1 Schalldruckpegel in Relation zu Schallereignissen 8
2.2 Energieverbrauch und Präzision in der Standortbestimmung [Goo15] 13

3.1 Funktionale Anforderungen im Prototyp 33
3.2 Nichtfunktionale Anforderungen im Prototyp 33
3.3 Anwendungsfall - Noise Map erforschen 34
3.4 Anwendungsfall - Vollbild aktivieren 35
3.5 Anwendungsfall - Hilfe aufrufen 35
3.6 Anwendungsfall - Details anzeigen 35
3.7 Anwendungsfall - Messungen filtern 36
3.8 Anwendungsfall - Statistik anzeigen 36
3.9 Anwendungsfall - Daten erfassen 36
3.10 HTTP Methoden im Prototyp 43
3.11 Java Web Applikation Abhängigkeiten 48
3.12 Noise Map Abhängigkeiten 59

4.1 Geschlecht und Alter der Probanden sowie die verwendeten Android Geräte . 64
4.2 Anzahl der durchgeführten Messungen mit niedrigstem, höchstem und durchschnittlichem Schalldruckpegel pro Proband 65

KAPITEL 1

Einleitung

Die Lärmbelastung im urbanen Raum mit ihren negativen gesundheitlichen Auswirkungen auf den Menschen, wird zunehmend als Problem wahrgenommen. Im Rahmen dieser Diplomarbeit soll mit Hilfe der Informatik eine Möglichkeit geschaffen werden, Lärmquellen zu erkennen und auf diese zu reagieren.

Den Ausgangspunkt für dieses Vorhaben stellt die Kooperation der TU Wien mit dem Unternehmen SOPHISYSTEMS dar. Gemeinsam wurde eine Smartphone Applikation entwickelt, welche es möglich macht, den Schalldruckpegel in Dezibel zu messen. Der sogenannte Noise-O-Meter, derzeit verfügbar für das Betriebssystem *Android*, soll eine Einschätzung der eigenen Lärmbelastung ermöglichen und dadurch ein Bewusstsein für oft unbemerkte Lärmquellen im Alltag schaffen. Ziel der Applikation ist vor allem die Belastung über längere Zeiträume sichtbar zu machen. Einmal aktiviert, misst der Noise-O-Meter in vordefinierten Intervallen so lange, wie es der Benutzer erlaubt. Ein konkretes Einsatzszenario wäre beispielsweise die Belastung auf dem eigenen Arbeitsweg oder Arbeitsplatz zu analysieren. Die Applikation kann an Hand einer Farbskala Aufschluss über die unmittelbare Belastung (letzte Messung) geben. Durch Diagramme kann zusätzlich die Entwicklung über die letzten Wochen und Monate verfolgt werden (siehe Abbildung 1.1). Bei dieser Form der Darstellung gehen jedoch sowohl räumliche als auch zeitliche Informationen verloren. Es kann im Nachhinein nicht mehr nachvollzogen werden, wo gemessen wurde und wann genau die Messung erfolgt ist.

Damit der Anwender den vollen Nutzen aus seinen eigenen Messungen ziehen kann, ist eine verständliche Repräsentation der abstrakten Daten in deren raum-zeitlichen Kontext notwendig. Zu diesem Zweck wurde im Rahmen dieser Arbeit eine kartographische Visualisierung prototypisch realisiert, die es ermöglicht die gemessenen Werte in Verbindung zu den Orten, an denen sie erhoben wurden, zu setzen. Die „Noise Map“ stellt mittels laufender Aktualisierungen durch die Noise-O-Meter Benutzer eine hoch aktuelle Lärmkarte aus der Fußgängerperspektive dar. Grafische Repräsentationen der gemessenen Lärmquellen machen es möglich, besonders laute Orte zu meiden als auch Orte der Ruhe und Erholung zu entdecken. Verschiedene Interaktionsmöglichkeiten laden

Abbildung 1.1: Farbskala und Diagramme in der Noise-O-Meter Applikation

den Benutzer ein, die Daten zu erforschen und sich damit auseinanderzusetzen. Durch den Einsatz von Filtern können darüber hinaus zeitliche Veränderungen entdeckt und nachvollzogen werden. Schnelle Updatezyklen und die Möglichkeit neue Daten live auf der Karte mitzuverfolgen unterscheiden die Noise Map von vielen bestehenden Infrastrukturen, welche sich häufig auf Visualisierungen verkehrsintensiver Knotenpunkte beschränken [RCK10].

1.1 Motivation

Den Anstoß für die Entwicklungen an der Noise Map geben die Erkenntnisse der Forschung über die Auswirkungen von Langzeitbelastung durch Lärm auf den Menschen. Besonders im urbanen Raum wird Lärm als ein wesentlicher Störfaktor empfunden [dKS03] und wirkt sich als solcher auf die Lebensqualität und Entscheidungen der Menschen aus. So wird zum Beispiel die Wahl des Wohnsitzes maßgeblich durch die empfundene Lärmbelastung beeinflusst. Tsai et al. schreiben dazu:

> „Excessive noise is a major environmental complaint in residential areas.“ [TLC09]

Lärmbelastung kann jedoch nicht nur störend wirken, sondern sich bei langfristiger oder besonders intensiver Exposition auch schädlich auf die Gesundheit eines Individuums auswirken. Niessen et al. beschreiben in ihrer Arbeit unter anderem den potenziell negativen Einfluss von Lärm auf das Verhalten, das Wohlbefinden, die Produktivität und die Gesundheit von Mensch und Tier [Nie09]. Diese Auswirkungen betreffen eine hohe Anzahl von Individuen auf der ganzen Welt. Im Jahre 1990 wurden daher durch die World Health Organization (WHO) und 2002 durch die Europäische Union Forschungsprogramme

initiiert und Maßnahmen gegen die zunehmende Lärmverschmutzung in die Wege geleitet [ARR11]. Rana et al. über das Ausmaß der Belastung:

> „At present, a large number of people around the world are exposed to high levels of noise pollution, which can cause serious illnesses ranging from hearing impairment to negatively influencing productivity and social behavior.“ [RCK10]

Durch die Noise-O-Meter Applikation wurde bereits ein erster Schritt gesetzt, um Menschen die Möglichkeit zu geben, ihre Lärmbelastung im Alltag differenziert wahrzunehmen. Aus dieser Entwicklung heraus ist der Wunsch entstanden, die Erkenntnisse und Informationen, die einzelne Personen durch den Noise-O-Meter gewinnen können, der Allgemeinheit zur Verfügung zu stellen. Eine Karte, frei zugänglich im Internet soll einerseits das Bewusstsein für Lärmverschmutzung steigern und andererseits helfen, die Verbindung zwischen den Lärmmessungen des Noise-O-Meters und den Orten ihrer Entstehung herzustellen. Die Verknüpfung der Messung mit ihrem Ursprungsort resultiert nicht nur in einer kartografischen Visualisierung von Lärm - Noise Map - sondern bringt den Noise-O-Metern Benutzern, auch durch die räumliche Dimension, eine zusätzliche Perspektive auf die erhobenen Daten.

Ein weiterer Motivationsfaktor ist die Verwendung eines alternativen Ansatzes in der prototypischen Realisierung, wodurch dessen Machbarkeit bewiesen werden soll. Statt über statische Sensornetzwerke, deren Errichtung und Betrieb mit hohen Kosten verbunden sind [RCK10], sollen die notwendigen Daten mit Hilfe der Noise-O-Meter Benutzer gesammelt werden. Durch diese Methode, das sogenannte *Participatory Sensing*, beschrieben in Kapitel 2.1.2, sollen eine hohe Aktualität der Daten gewährleistet und zusätzlich neue Perspektiven auf die eigenen Umgebung eröffnet werden. Die Vision der Noise Map lautet: Fußgänger zeichnen eine Lärmkarte ihrer eigenen Stadt.

1.2 Aufgabenstellung

Das Ziel dieser Diplomarbeit ist der Entwurf und die prototypische Realisierung einer kartografischen Visualisierung, welche die erhobenen Lärmmessungen durch die Noise-O-Meter Applikation in ihrem raum-zeitlichen Kontext darstellen soll. Die Umsetzung der sogenannten „Noise Map“ umfasst die Arbeit an drei großen Bereichen:

- *Erweiterung der Noise-O-Meter Applikation* - Zusätzlich zur Lärmmessung in Dezibel soll die aktuelle Position des Benutzers erfasst und gespeichert werden. Diese Standortbestimmung funktioniert über die jeweils zur Verfügung stehenden Standort-Provider. Dies kann entweder über das Global Positioning System (GPS) oder den jeweiligen Telefonnetz-Anbieter geschehen und bedarf eines Kompromisses zwischen der Qualität der Position und dem Energieverbrauch. Um Verwaltung und Analyse der Daten zu ermöglichen, muss jede Messung gemeinsam mit den raum-zeitlichen Metadaten eindeutig einem entsprechenden Benutzer zuordenbar sein. Daher ist es notwendig, eine Strategie zur Identifikation einzelner Geräte/Benutzer

zu implementieren. Damit die Daten visualisiert und verarbeitet werden können, müssen sie in regelmäßigen Intervallen an die entsprechende Schnittstelle übertragen werden. Hier sind einerseits Vorkehrungen zur Absicherung der Datenintegrität zu treffen und andererseits wieder ein Kompromiss zwischen einer möglichst hohen Aktualität der Daten und dem Energieverbrauch zu schließen.

- *Entwurf und Entwicklung einer Java Web Applikation* - Eine Java Web Applikation soll als Knotenpunkt für die gesammelten Daten dienen. Für die schnelle Ein- und Ausgabe von Daten wird eine geeignete Datenbank für Geodaten benötigt. Die Kommunikation mit den mobilen Geräten sowie der Visualisierung soll über entsprechende Schnittstellen nach den Representational State Transfer (REST) Prinzipien erfolgen. Da in der Theorie eine hohe Anzahl an Noise-O-Meter Benutzern gleichzeitig Daten übertragen könnten, birgt dies auch spezielle Anforderungen an die Skalierbarkeit der Applikation.

- *Entwurf und Entwicklung einer kartografischen Visualisierung* - Die kartografische Visualisierung soll ähnlich dem bekannten Vorbild Google Maps die gemessenen Daten im Internet verfügbar machen. Die Basis für die „Noise Map“ besteht aus modernen JavaScript Frameworks und Open Data Kartenmaterial, welches mittels grundlegender Interaktionsmöglichkeiten erforscht werden kann. Eine ansprechende grafische Repräsentation soll einerseits eine aggregierte Ansicht aller Lärmdaten und andererseits eine Analyse einzelner Lärmquellen ermöglichen. Eine erweiterte Analyse mit Hilfe statistischer Werkzeuge sowie die Darstellung durch Diagramme ist ebenfalls vorgesehen. Mit Hilfe von Filtern soll der Benutzer den angezeigten Datensatz in seinem Umfang manipulieren können. Dynamische Repräsentationen sowie Interaktionsmöglichkeiten sollen zum Erforschen der Daten motivieren und durch unterschiedliche Perspektiven Einsichten bringen. Die Visualisierung soll schnelle Updatezyklen unterstützen und dadurch sehr aktuelle Daten präsentieren.

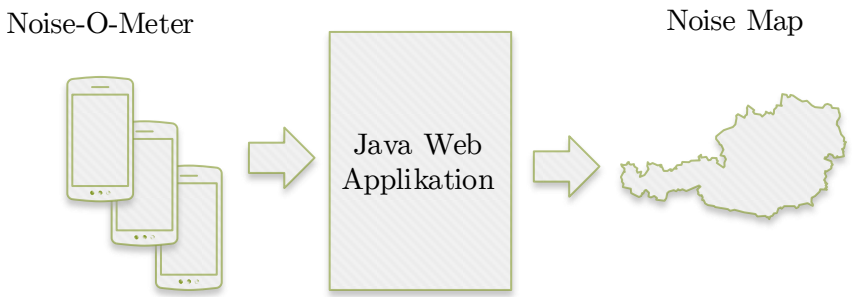

Abbildung 1.2: Die drei Säulen der Noise Map

Abschließend soll durch die Evaluation des Prototyps der Frage nachgegangen werden ob die Noise Map Einsichten in die Daten geben kann. Mit verschiedenen Aufgaben werden die Eignung der Repräsentation, die Lesbarkeit sowie die Bedienbarkeit überprüft. Eine Stichprobe von zehn Personen wird mit der Noise-O-Meter Applikation ausgerüstet. Mittels einer Einverständniserklärung werden die Rahmenbedingungen geklärt und ein abschließendes Interview vereinbart.

1.3 Methoden

Abbildung 1.3 zeigt die Verfahrensweise, welche im Laufe der Arbeit angewandt wird. Die einzelnen Phasen betreffen die drei in Kapitel 1.2 beschriebenen Bereiche. Im Rahmen der Analyse werden mittels Recherche Literatur zur Thematik identifiziert sowie aktuelle Projekte mit der gleichen Zielsetzung (siehe Kapitel 2.4) untersucht, um eine Wissensbasis für die Konzeption zu schaffen. Die Planungs- und Entwurfsphase beinhaltet sowohl die Identifikation der Stakeholder als auch die Erhebung von funktionalen sowie nichtfunktionalen Anforderungen an den Prototyp. Durch die Erstellung konkreter Anwendungsfälle werden die Interaktionsmöglichkeiten der Benutzer mit dem System definiert. Die Entwurfsphase schließt mit der Festlegung auf die Architektur sowie der Auswahl der zu verwendenden Technologien und Schnittstellen. Auf Basis dieses Entwurfs wird in der Implementierung ein Prototyp realisiert. Vorteile des *Prototyping* in der Softwareentwicklung liegen vor allem darin, dass fehlerhafte Anforderungsdefinitionen rasch erkannt und frühzeitig Qualitätssicherung durchgeführt werden können [LS92]. Dieser iterative Prozess spiegelt sich im vorliegenden Projekt in ausführlichen Systemtests sowie der anschließenden Evaluation wider, welche unter Einbeziehung der späteren Anwender die Basis für eine erneute Entwicklungsphase am Prototyp darstellt. Die gesammelte Erfahrung am Prototyp kann wertvolle Erkenntnisse für das Endprodukt liefern.

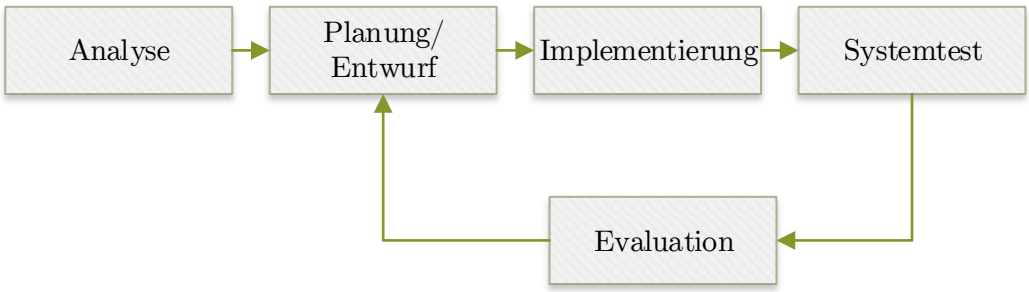

Abbildung 1.3: Eingesetztes Vorgehensmodell in Entwurf und Entwicklung

1.4 Ausblick

Das folgende Kapitel dient einer theoretischen Einführung in wichtige Themengebiete und deren Bedeutung in der vorliegenden Arbeit. Kapitel 2.4 beschreibt Projekte mit ähnlicher Zielsetzung und deren Herangehensweise sowie deren technische Umsetzung. Das Kapitel Implementierung 3 beschreibt die Realisierung des Prototyps vom Entwurf bis zur Entwicklung. Kapitel 4 gibt Einsichten in die Evaluation und beschreibt Planung, Zielsetzung, Ablauf und Resultate. Die Arbeit schließt mit Kapitel 5, worin ein Fazit gezogen sowie ein Ausblick auf mögliche zukünftige Entwicklungen gegeben wird.

KAPITEL 2

Theoretische Grundlagen

Dieses Kapitel behandelt die theoretischen Hintergründe der drei in Kapitel 1.2 beschriebenen Säulen der Noise Map. Diese sind die Noise-O-Meter Applikation (Messung), die Java Web Applikation (Verarbeitung) und die Noise Map (Präsentation). Im Folgenden wird zu jeder der genannten Säulen auf relevante Themen eingegangen. Im Kapitel 2.4 werden abschließend Konzept und Umsetzung von fünf inhaltlich verbundenen Projekten beschrieben.

2.1 Noise-O-Meter

Die Noise-O-Meter Applikation stellt eine Art Sensor-Komponente dar, über die der Schalldruckpegel gemessen werden kann. Daher widmet sich Abschnitt 2.1.1 der Messbarkeit von Lärm sowie der Bedeutung der gemessenen Werte. Ein wichtiges Prinzip, das die Noise Map und andere in Kapitel 2.4 aufgeführte Projekte eint, ist die kollektive Datenerfassung über mobile Endgeräte. Das Prinzip des sogenannten *Participatory Sensing* und seine Anwendung in der vorliegenden Arbeit wird in Abschnitt 2.1.2 beschrieben. Die durch diese „Schwarm-Intelligenz“ gewonnenen Daten bilden die Basis für die Visualisierung der Lärmquellen auf der Noise Map. Um diese kartografisch darstellen zu können, muss möglichst zeitnah zur Messung der Standort bestimmt werden. Diesem Vorgang widmet sich Abschnitt 2.1.3.

2.1.1 Lärmmessungen

Maue et al. definieren Lärm als unerwünschtes Schallereignis:

> „Lärm ist ein unerwünschtes Geräusch, das zu einer Belästigung, Störwirkung, Beeinträchtigung der Leistungsfähigkeit, besonderen Unfallgefahren oder Gesundheitsschäden führt.“ [MHL09]

Ob ein Schallereignis von einem Menschen als Lärm empfunden wird, ist einerseits von der Lautstärke und andererseits von der persönlichen Einstellung des Hörers abhängig [MHL09]. So kann beispielsweise laute Musik abhängig vom eigenen Geschmack auch als angenehm empfunden werden, was jedoch nichts an ihrer potenziell gesundheitsschädlichen Wirkung ändert. Das Lärmempfinden des Menschen ist also subjektiv und Lärm daher keine exakte Größe. Die Einwirkung eines Geräusches auf den Menschen wird darum über den Schalldruck bestimmt. Mit diesem Wert, gemessen in Pascal, kann der sogenannte Schalldruckpegel berechnet werden:

$$L_p = 10 \cdot lg(\frac{p^2}{p_0^2}) \text{ dB} \tag{2.1}$$

L_p ... Schalldruckpegel
p ... gemessener Schalldruck
$p_0 = 2 \cdot 10^{-5} Pa$ ... Hörschwelle

Führt man eine Messung mit dem Noise-O-Meter durch so wird der Schalldruckpegel L_p in Dezibel sowie Datum und Uhrzeit gespeichert. Ziel dieser Messung ist, ein Geräusch möglichst objektiv zu erfassen und zu beschreiben.

Da diese Werte für den Einzelnen eher schwierig in Relation zu den tatsächlichen Geräuschen zu setzen sind (wie dies in Tabelle 2.1 geschieht), werden sie im Noise-O-Meter auf einer Farbskala von leise (grün) bis laut (rot) dargestellt. Umweltlärm im Bereich von 65 - 70 dB, wie beispielsweise durch Verkehr verursacht, kann im Körper Stressymptome auslösen und erhöht dadurch das Risiko für Erkrankungen [MHL09]. Befindet sich die zuletzt durchgeführte Messung des Noise-O-Meters im roten Bereich und ist daher höher als 65 dB wird der Benutzer mit der Nachricht „Es war deutlich zu laut!“ gewarnt.

Schallereignisse	
Schallereignis	Schalldruckpegel
Schmerzgrenze	120 dB
Diskothek	100 dB
Hauptverkehrsstraße	80 - 90 dB
Pkw	60 - 80 dB
Unterhaltung	40 - 60 dB
Ruhiges Zimmer	20 - 30 dB

Tabelle 2.1: Schalldruckpegel in Relation zu Schallereignissen

Mit der Noise-O-Meter Applikation wurde ein Werkzeug geschaffen, welches es ermöglicht, sowohl auf längerfristige als auch auf unmittelbare Geräuschimmissionen im Alltag zu reagieren. Die resultierenden Daten schaffen ein Bewusstsein für die potenzielle Gefahr und könnten sich dadurch positiv auf die Lebensqualität auswirken. Darüber hinaus stellen sie die Voraussetzung für die Noise Map dar. Um aktuelle Daten in hoher Anzahl

zu erhalten wird das moderne Konzept des *Participatory Sensing* eingesetzt, welches im folgenden Abschnitt beschrieben wird.

2.1.2 Participatory Sensing

Die massive Verbreitung mobiler Endgeräte führt zu einer hohen Verfügbarkeit technologisch hochentwickelter Sensoren. Hier setzt das Konzept des *Participatory Sensing* an, indem alltägliche mobile Geräte ein Sensor-Netzwerk bilden, mit dem Ziel lokales Wissen zu sammeln, zu analysieren und zu verteilen [BEH06]. Die folgende Aufzählung zeigt die vielfältigen Möglichkeiten, Daten und Informationen über die Sensoren moderner Smartphones zu erhalten.

- Bild- und Videomaterial (Kamera)
- Akustische Informationen (Mikrofon)
- Standort-Informationen (GPS, WiFi)
- Kontext-Informationen (Gyroskop, Beschleunigungs- und Lage-Sensor) wie z.B. die Art der Fortbewegung (zu Fuß, Rad, Auto)
- Umwelt- und Biometrische Daten über zusätzlich verbundene Sensoren (via Bluetooth) [CRKH11]

Im vorliegenden Projekt sind sowohl akustische als auch Standort-Informationen notwendig, um den Schalldruckpegel kartografisch visualisieren zu können. Diese erhält man über das Mikrofon und je nach Verfügbarkeit durch einen Standortprovider. Gemeinsam mit einem Zeitstempel werden die gesammelten Sensordaten zur Verarbeitung und Aufbewahrung an einen zentralen Server übertragen. Zum Einsatz kommt die in Abbildung 2.2 beschriebene, für *Participatory Sensing* Projekte typische Architektur, bestehend aus fünf Komponenten [CRKH11]. Die Sensor- und die Report-Komponente

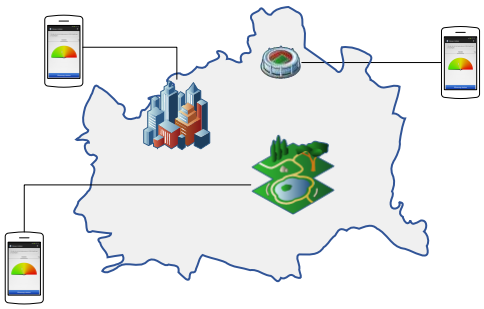

Abbildung 2.1: Noise-O-Meter an unterschiedlichen Punkten der Stadt

sind für die Datenerfassung sowie die Übertragung zuständig. Diese Funktionen sind im vorliegenden Projekt in der Noise-O-Meter Applikation gebündelt. Die zweite Säule, die Java Web Applikation, umfasst die Funktion der Aufbewahrung und Verarbeitung der Daten. Die Präsentation der Daten erfolgt durch die Noise Map.

Abbildung 2.2: Typische Architektur in einem Participatory-Sensing-Projekt [CRKH11]

Participatory Sensing ermöglicht den Einsatz einer großen Anzahl an Sensor-Devices zu geringen bis gar keinen Kosten. Die Anzahl sowie die Beweglichkeit führt zu einer nie dagewesenen raum-zeitlichen Abdeckung [CRKH11]. Der Benutzer des mobilen Endgerätes nimmt im *Participatory Sensing* die Rolle eines Operators ein, der zu einem bestimmten Zweck einen mobilen Sensor bedient. Durch diese starke Inklusion ist es besonders bei benutzerzentrierten Projekten möglich, den Alltag der Menschen massiv zu verbessern [CRKH11].

Die Qualität der Daten ist dabei nicht nur von einem möglichst störungsfreien Messergebnis, sondern insbesondere auch von den raum-zeitlichen Metadaten abhängig. Das bedeutet, je präziser die kontextuellen Daten, desto höher ist die Qualität und desto eher wird das wichtige Kriterium der Glaubwürdigkeit erreicht [BEH06]. Diese Forderung nach genauen Metadaten steht jedoch im Gegensatz zum Schutz der Privatsphäre, da diese Bewegungsmuster der Teilnehmer erkennen lassen [HKH10]. Huang et al. zur Notwendigkeit des Datenschutzes in *Participatory Sensing*:

> „Note that, participatory sensing relies on the altruistic participation of users for widespread penetration and successful operation. It is thus imperative that users are assured that their temporal and spatial privacy will not be violated to encourage sufficient participation.“ [HKH10]

Um dieser Sorge Rechnung zu tragen, überträgt der Noise-O-Meter keine benutzer- oder gerätespezifischen Daten, wie beispielsweise die Geräte Identifikationsnummer (ID) oder International Mobile Equipment Identity (IMEI). Der Übertragung wird lediglich eine generierte Benutzer ID hinzugefügt, wodurch Anonymität gewährleistet werden kann. Huang et al. argumentieren, dass dies unter Umständen nicht ausreicht, wenn bereits Vorwissen über einen Teilnehmer vorhanden ist [HKH10]. Aus diesem Grund wurden zusätzlich im Java Backend Vorbereitungen für sogenannte *Microaggregation* getroffen, bei der mittels Clustering Daten so aggregiert werden können, dass kein Rückschluss mehr auf einzelne Messungen möglich ist. Den Empfehlungen von Christin et al. folgend, erhält der Teilnehmer die Möglichkeit die Übertragung zu deaktivieren, wodurch er die Kontrolle über die Datensammlung erlangt [CRKH11].

Wie bereits erwähnt, ist die Versorgung mit Daten und damit der Erfolg des Projekts zumindest teilweise vom uneigennützigen Verhalten der Teilnehmer abhängig. Diese stellen über Energie und Leistung ihrer mobilen Geräte ihre privaten Ressourcen zur Verfügung. Es ist daher sinnvoll Anreize zu liefern, welche über den Schutz der Privatssphäre hinausgehen. Neben verschiedenen Vergütungsmodellen, die oft auf der Bewertung der Datenqualität und des Benutzers basieren, setzen besonders Projekte aus dem Sozial- und Gesundheitsbereich auf Anreize durch Information. Im vorliegenden Projekt werden die Daten für den Benutzer so aufbereitet, dass dieser einen persönlichen Vorteil daraus ziehen kann. Dies geschieht einerseits unmittelbar über den Noise-O-Meter und dessen Benutzeroberfläche und andererseits über die räumliche Darstellung auf der Noise Map. Wie der Standort im mobilen Einsatz möglichst präzise erhoben werden kann ist Thema des nächsten Abschnitts.

2.1.3 Android Standortbestimmung

Der räumliche Kontext stellt neben der Zeit eine der wichtigsten Komponenten im *Participatory Sensing* dar [CRKH11]. Für das mobile Betriebssystem *Android* gibt es zwei etablierte Standort-Frameworks. Das *Android-Location-Framework* und die neuere und empfohlene *Google Location API*. Eine Nutzung der *Google Location API* setzt den Einsatz

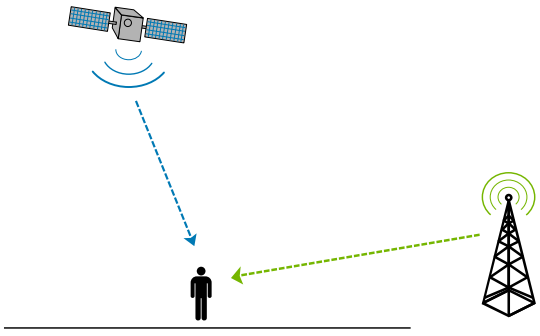

Abbildung 2.3: Standortbestimmung über GPS, WiFi und das Telefon-Netz

eines Google Kontos auf dem persönlichen *Android*-Gerät voraus. Dieser Umstand schließt jene Nutzer von der Standortbestimmung - und damit von der Möglichkeit zur Noise Map beizutragen - aus, die ihr Gerät ohne ein solches Konto betreiben. Die Vorteile überwiegen jedoch, da durch den sogenannten *Fused Location Provider* die Verwaltung der einzelnen für die Standortbestimmung notwendigen Technologien wegfällt. Dieser Unterschied stellt auch einen entscheidenden Faktor beim Kompromiss zwischen der Präzision des Standorts und dem Energieverbrauch dar. Durch das automatisierte Management der eingesetzten Technologien, kann der Energieverbrauch maßgeblich reduziert und dadurch ein besseres Erlebnis für den Benutzer geschaffen werden [Goo15]. Ein weiterer Vorteil der *Google Location API* ist die Geschwindigkeit der Standortbestimmung. Wird der Noise-O-Meter gestartet, hat der Benutzer sofort die Möglichkeit, die erste Messung durchzuführen. Zu diesem Zeitpunkt müssen also bereits möglichst präzise Koordinaten verfügbar sein. Abbildung 2.4 zeigt den Unterschied zwischen der Plattform-Implementierung und der *Google Location API*.

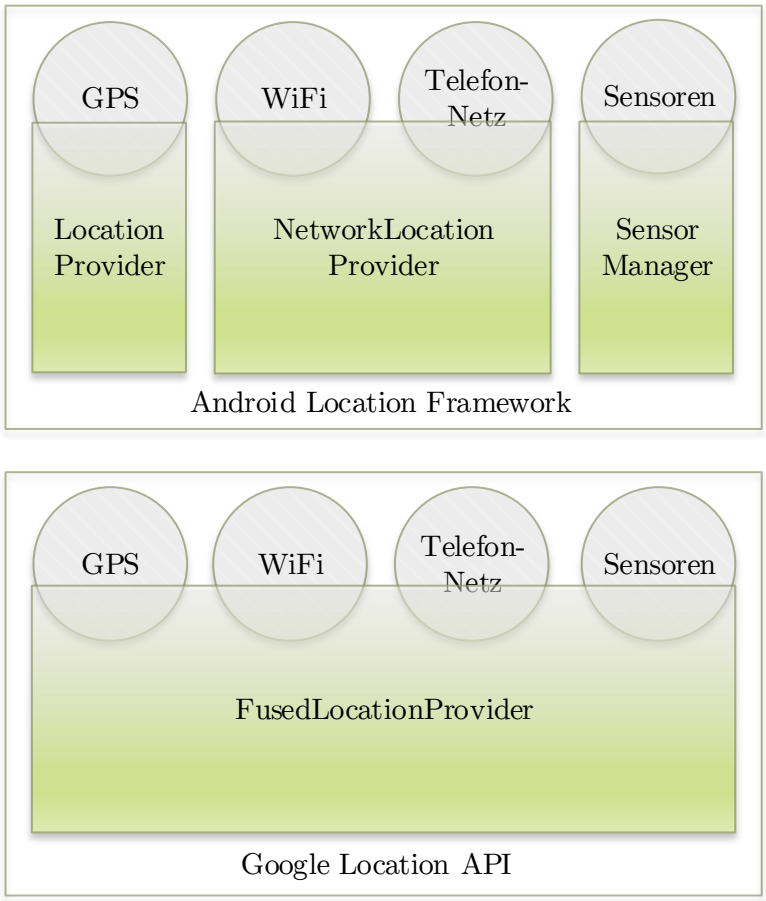

Abbildung 2.4: Vergleich der beiden Android-Location-Frameworks

Auch für Business-Stakeholder bietet die *Google Location API* mit *Geofencing* und *Activity recognition* interessante Optionen. Ersteres bietet die Möglichkeit ein geographisches Areal zu definieren, bei dessen Betreten oder Verlassen eine Funktion ausgeführt werden soll. *Activity recognition* kann über die Sensoren des Smartphones die Art der Fortbewegung erkennen. Dies könnte sich auch für *Participatory Sensing* als hilfreich herausstellen, um bei der Fortbewegung zu Fuß automatisch Messungen zu starten beziehungsweise bei anderen Fortbewegungsarten zu beenden. Bewegung ist auch in der Standortbestimmung ein relevanter Aspekt. Die Noise-O-Meter Applikation kann stationär aber auch in Bewegung eingesetzt werden. Daher ist es notwendig, den Standort regelmäßig zu aktualisieren, um die Messungen an ihrer tatsächlichen Position abbilden zu können. Dabei ist zu beachten, dass die häufige Abfrage des aktuellen Standorts besonders hohen Energieverbrauch verursacht. Die *Google Location API* bietet folgende Einstellungsmöglichkeiten die sich in Energieverbrauch und Präzision unterscheiden:

Google Location API Prioritäten		
Einstellung	Energieverbrauch	Genauigkeit
HIGH ACCURACY	moderat	hoch (~10m)
BALANCED POWER ACCURACY	niedrig	mittel (~100m)
LOW POWER	sehr niedrig	niedrig (~10km)
NO POWER	-	-

Tabelle 2.2: Energieverbrauch und Präzision in der Standortbestimmung [Goo15]

Wie bereits im vorherigen Kapitel beschrieben, sind die Metadaten entscheidend für die Datenqualität. Daher ist die Einstellung HIGH ACCURACY zu bevorzugen, welche am häufigsten auf GPS zugreift und dadurch den höchsten Energieverbrauch verursacht. Das Resultat der Standortbestimmung ist ein *android.location.Location* Objekt, welches durch die Winkel „latitude“ (phi ϕ) und „longitude“ (lambda λ) den gesuchten Ort auf der Geosphäre der Erde (siehe Abbildung 2.5) beschreibt.

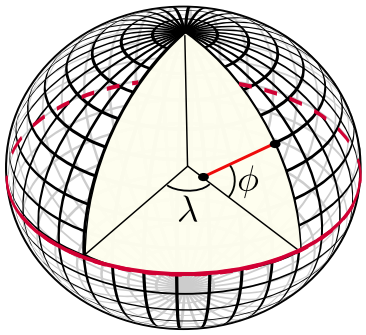

Abbildung 2.5: Latitude phi ϕ und Longitude lambda λ

Latitude ist der Winkel zwischen der Äquatorlinie und einer Geraden durch den gesuchten Punkt und das Zentrum der Sphäre. Longitude ist der Winkel zwischen dem Greenwich-Meridian und dem Meridian, welcher durch den gesuchten Punkt verläuft.

Die gemessenen Lärmdaten sowie der ermittelte Standort werden an einen zentralen Server mit standardisierter Schnittstelle übertragen. Diese wird im folgenden Kapitel beschrieben.

2.2 Java Web Applikation

Das Java Backend bildet die Schaltzentrale zwischen Noise-O-Meter und der Noise Map. Die Unterscheidung zwischen Backend und Frontend wird vorgenommen, um eine klare Trennung der Zuständigkeiten - Datenzugriff und Präsentation - zu erzielen. Die Daten aller Noise-O-Meter Benutzer werden in regelmäßigen Intervallen an das Backend übertragen und anschließend in diesem verarbeitet und gespeichert. Erfolgt ein Aufruf der Noise Map werden die Daten über das Backend bezogen. Zu diesem Zweck wird eine geeignete Schnittstelle nach den REST Architekturprinzipien definiert. Architektur und verwendete Entwurfsmuster finden sich in Abschnitt 3.1.

2.2.1 Representational State Transfer (REST)

Eine Web Anwendung, welche die REST Prinzipien umsetzt, ist in ihrer Funktionsweise dem World Wide Web (WWW) ähnlich. Über einen Uniform Resource Identifier (URI) können Ressourcen angefordert und über ein Anwendungsschicht-Protokoll übertragen werden. Dieses Protokoll ist häufig das Hypertext Transfer Protocol (HTTP). Richardson et al. vergleichen die Funktionsweise von HTTP mit dem Postversand eines Briefs. Ein Client (Mensch oder Maschine) steckt ein Dokument in einen Umschlag (Request) und sendet diesen ab. Bei Erhalt tut es der Server (Maschine) dem Client gleich und versendet eine entsprechende Rückmeldung (Response) [RR07]. Diese Request-Response Kommunikation zwischen Client und Server sowie die Beschränkung auf ein Minimum an notwendigen Methoden ist mit ein Grund für die Einfachheit, die REST auszeichnet.

Im Folgenden werden die Prinzipien auf der Basis der Arbeit von Burke et al. [Bur09] erläutert:

1. *Adressierbarkeit* - Jede Ressource muss über einen eindeutigen URI erreichbar sein. Alles was als Stream von Bits repräsentiert und auf einer Festplatte gespeichert werden kann, kann eine Ressource darstellen [RR07].

2. *Uniformes Interface* - Die Methoden zur Manipulation von Ressourcen sollen sich auf jene Operationen beschränken, die vom Protokoll angeboten werden. Im Fall von HTTP sind die wichtigsten Operationen GET, POST, PUT und DELETE.

3. *Zustandslosigkeit* - Jede Operation soll aus vollständiger Isolation heraus erfolgen. Das bedeutet, dass keine Daten über den Client auf dem Server gespeichert werden, sondern ausschließlich Daten über die verwalteten Ressourcen.

4. *Repräsentations-orientiert* - Services können unterschiedliche Repräsentationen annehmen, um unterschiedliche Clienten zu befriedigen. So könnte beispielsweise eine angeforderte Ressource für eine JavaScript Applikation in JavaScript Object Notation (JSON) und für eine Java Applikation in Extensible Markup Language (XML) übertragen werden.

2.3 Noise Map

Die Präsentation der Daten erfolgt durch die Noise Map, welche ein bedeutendes Element dieser Arbeit darstellt. Diesem Vorgang der Visualisierung von abstrakten Daten, mit dem Ziel sie visuell erfassbar zu machen, widmet sich Abschnitt 2.3.1. Es werden die beiden wichtigen Begriffe Repräsentation und Interaktion eingeführt. Während sich dieses Kapitel der Thematik eher allgemein nähert, geht Kapitel 2.3.2 speziell auf die Darstellung räumlicher Daten ein. Im Kapitel 2.3.3 wird der Prozess des Web Mappings beleuchtet und erläutert, wie Repräsentation und Interaktion in einer modernen Web Map funktionieren. Abschließend widmet sich Kapitel 2.3.4 zwei unterschiedlichen Konzepten zur Visualisierung von Lärm.

2.3.1 Informationsvisualisierung

Informationsvisualisierung ist ein Forschungsgebiet, welches die computerunterstützte Darstellung von Daten behandelt. Folgt man den Ausführungen von Mazza so stellt sie die Antwort auf die Datenexplosion dar, der wir gegenwärtig gegenüber stehen [Maz09]. Yi et al. nennen Repräsentation und Interaktion als die zwei Hauptbestandteile dieses Gebiets.

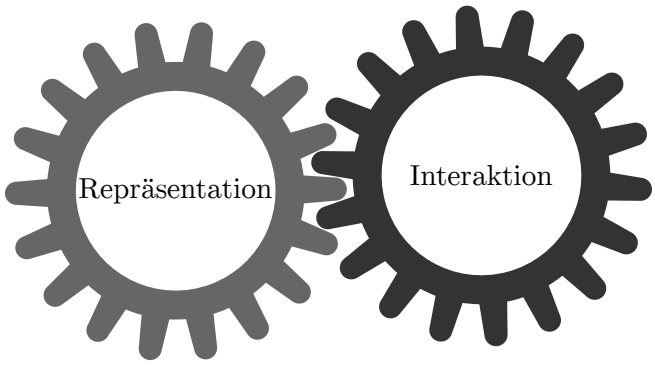

Abbildung 2.6: Elemente der Informationsvisualisierung nach Yi et al. [YKSJ07]

Der Einsatz visueller Repräsentationen kann helfen, komplexe Informationen so darzustellen, dass diese schnell und effizient erfasst werden können. Es kann zum Anstoß eines Kommunikations- und Verständnisprozesses kommen, welcher aus Daten Informa-

tionen und letzten Endes Wissen werden lässt [Maz09]. Als Kriterien für gute visuelle Repräsentationen nennt Mazza:

- Grafische Exzellenz:
  Repräsentationen sollen klar, präzise und effizient gestaltet sein.
- Grafische Integrität:
  Repräsentationen dürfen nicht zu falschen Interpretationen führen.
- Maximieren:
  Überladene Darstellungen sollen vermieden werden.
- Ästhetik:
  Es ist ein elegantes Design mit Liebe zum Detail anzustreben.

Diese Kriterien finden auch in der Noise Map Anwendung. Einerseits durch überlegte und evaluierte grafische Repräsentationen und andererseits durch Mechanismen, welche einen Verlust von Informationen durch überladene Ansichten verhindern.

Der zweite Hauptbestandteil, die Interaktion, spielt zunehmend eine wichtige Rolle. Sie bietet dem Benutzer zusätzliche Möglichkeiten, sich mit den Daten auseinanderzusetzen sowie Einsichten zu gewinnen und überwindet die Limitationen einer statischen Repräsentation [Cra02].Yi et al. über die Rolle von Interaktion in der Informationsvisualisierung:

> „Without interaction, an Infovis technique or system becomes a static image or autonomously animated images ... .“ [YKSJ07]

Interaktion kann beispielsweise als eine Kommunikation zwischen Benutzer und System definiert werden oder auch als eine Manipulation des Systems durch den Benutzer, welche unmittelbare Änderungen hervorruft. Zu beachten ist, dass Interaktion sich auch auf die Repräsentation auswirken kann [YKSJ07].

2.3.2 Geovisualisierung

Wenn es, wie im Fall der Noise Map, um die Darstellung räumlicher Daten geht, spricht man von Geovisualisierung, einem Spezialgebiet der Informationsvisualisierung. In dieser Disziplin sind zusätzlich Aspekte aus den Bereichen Geoinformation, Kartografie, Bildanalyse sowie Datenanalyse von Relevanz. Im Mittelpunkt steht die Darstellung räumlicher Daten zum Zweck der Erforschung und Wissensvermittlung. Dieser Zweig hat in den letzten Jahren einen starken Aufschwung erfahren und wird durch die hohe Verfügbarkeit von frei zugänglichem Datenmaterial und Werkzeugen (siehe 2.3.3) begünstigt [Kra03]. Wo früher Papierkarten und Statistik eingesetzt wurden, werden heute Tabellenkalkulation, Datenbanken und Grafiktools verwendet. Diese Entwicklung zu Gunsten digitaler Repräsentation ist auch auf die steigende Komplexität des räumlichen Datenmaterials zurückzuführen [Kra03].

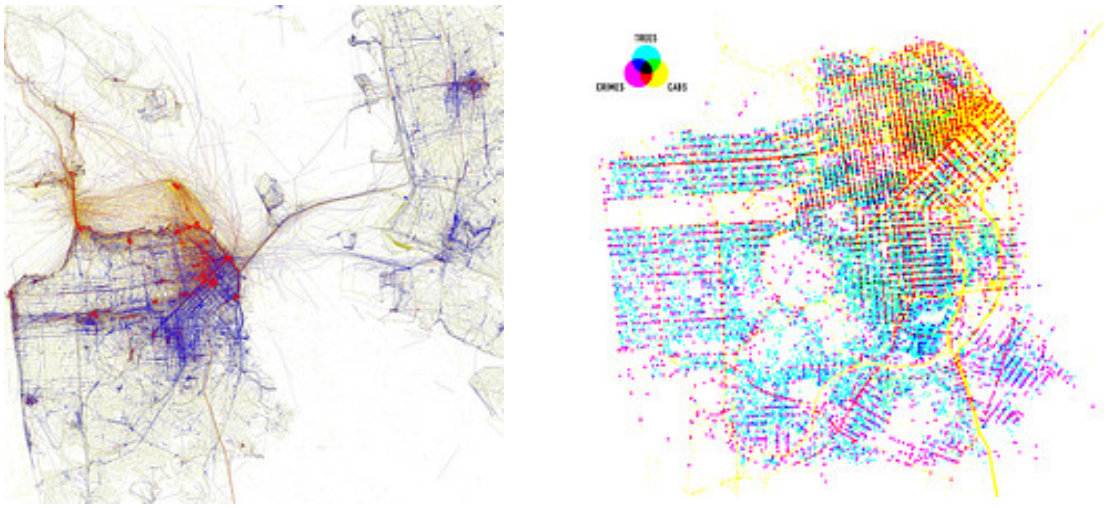

Abbildung 2.7: Zwei Beispiele für Geovisualisierungen (Quelle: `www.flickr.com` abgerufen April-2015; Lizenz: CC-BY-SA)

Abbildung 2.7 zeigt zwei Beispiele für Geovisualisierungen. Die linke Visualisierung von Eric Fisher bildet Orte ab, an denen Menschen in San Francisco fotografieren. Jene Orte, an welchen Touristen fotografieren, sind rot dargestellt, jene an welchen Einheimische fotografieren blau und Überschneidungen gelb. Die rechte Visualisierung von Shawn Allen zeigt in drei Farben das Vorkommen von Bäumen (cyan), Verbrechen (magenta) und Taxis (gelb) in San Francisco.

Interaktion

Nach Kraak unterstützt insbesondere die Visualisierung mittels interaktiver Karten den Informationszugang. Diese erleichtern und fördern durch verschiedene Perspektiven auf das Datenmaterial das aktive Erforschen der Daten [Kra03]. Crampton unternimmt in seiner Arbeit [Cra02] den Versuch Interaktionsmöglichkeiten in Geovisualisierungen in vier Gruppe einzuteilen. Diese sind in Abbildung 2.8 ersichtlich und lauten:

1. Interaktionen mit Datenrepräsentationen
2. Interaktionen mit der zeitlichen Dimension
3. Interaktionen mit den Daten
4. Interaktionen mit dem Kontext

Interaktionen der ersten Gruppe ermöglichen den bereits durch Kraak erwähnten Perspektivenwechsel, welcher auch zusätzliche Wertschätzung für die Daten ermöglichen soll [Kra03]. Interaktion mit der zeitlichen Dimension ermöglicht dem Benutzer, sich in der Visualisierung zu bewegen und dadurch vergangene Eindrücke mit gegenwärtigen

zu vergleichen. Diese Interaktionen stellen eine Herausforderung dar, durch welche der Benutzer vor die Wahl gestellt wird, in welche Richtung er weiterforschen möchte [Cra02]. In Kategorie drei tritt der Benutzer in Interaktion mit den Daten und hat durch die Anwendung von Statistik und Filtern die Möglichkeit Unbekanntes zu erforschen. Den Kontext, in dem die Daten eingebettet sind, sieht Crampton als besonders wichtigen Faktor für die Exploration. Er fordert daher unterschiedliches Kartenmaterial:

„... the era of the „single map solution“ to a problem is past.“ [Cra02]

Interaktionsmöglichkeiten aus allen genannten Gruppen finden sich auch in der Noise Map wieder. Diese werden ausführlich im Kapitel 3.4 beschrieben.

Abbildung 2.8: Interaktionstypen in Geovisualisierungen nach Crampton [Cra02]

Zusammenfassend charakterisiert Crampton Geovisualisierungen als in hohem Maße interaktiv, hochgradig explorativ und üblicherweise in privatem Setting [Cra02].

Das folgenden Kapitel bezieht sich speziell auf den Teilbereich der Kartographie und soll Repräsentation und Interaktion am Beispiel von Web Maps erläutern.

2.3.3 Web Maps

Unter Web Mapping versteht man den Prozess von Design, Implementierung und Bereitstellung von Maps im World Wide Web [Neu08]. Ziel ist die Präsentation oder Analyse

räumliche Informationen. Der starken Trend zu Web Maps ist der fortschreitenden Vernetzung und dem dadurch einfachen Transfer von Daten und Technologien zu verdanken [Neu08] [Zhu01]. 2005 leitete Google durch die *Google Maps API*, basierend auf Hypertext Markup Language (HTML), JavaScript und Ajax, das Zeitalter moderner Web Maps für die breite Masse ein [Neu08]. Die Anwendungsfelder für web-basiertes Mapping

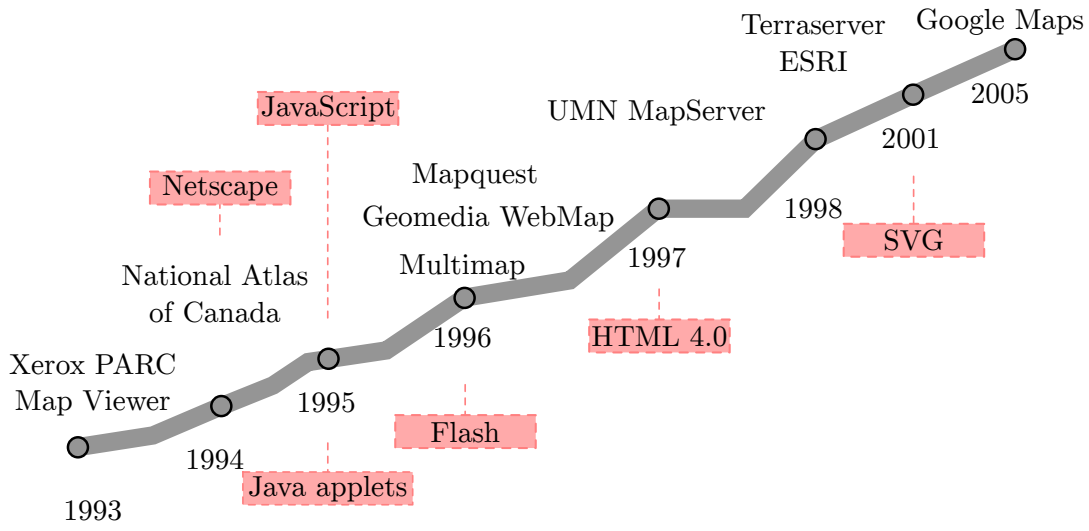

Abbildung 2.9: Historische Entwicklung der Web Maps und wichtige Technologien nach Neumann [Neu08]

sind vielfältig. Eingesetzt werden sie unter anderem im Ressourcen-Management, im Umwelt-Monitoring, in der Stadtplanung, in der Bildung, zur Navigation und in Planungsprojekten unter Beteiligung der Community [Neu08] [Zhu01]. Grundsätzlich sind Karten mit statischen Inhalten von Karten mit dynamischen Inhalten zu unterscheiden.

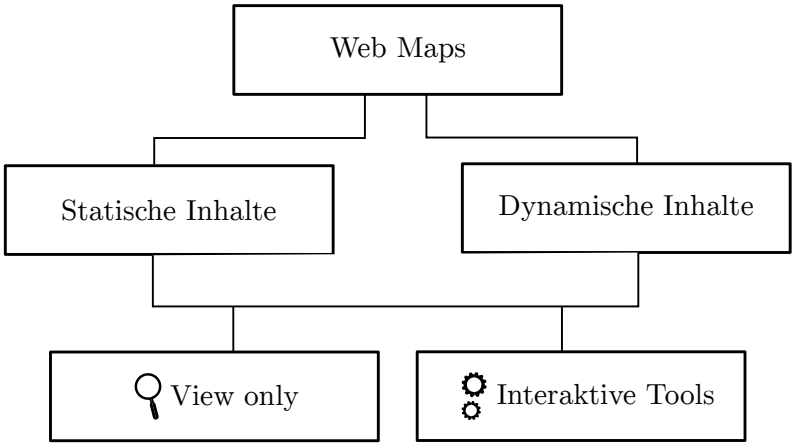

Abbildung 2.10: Klassifizierung von Web Apps [BC05]

Klassifizierung nach Neumann

Diese Einteilung berücksichtigt jedoch nicht die Diversität der heute verfügbaren Maps im Internet. Neumann unternimmt in seiner Arbeit [Neu08] den Versuch einer zeitgemäßeren Klassifizierung:

- Statisch: Dieser Typ wird genau einmal erstellt und verändert sich danach nicht mehr. Er verfügt weder über Animation noch Interaktivität.
- Dynamisch generiert: Bei jeder Anfrage wird die Karte neu erstellt. Dies kann beispielsweise auf Basis einer Geodatenbank erfolgen.
- Dezentral: Dieser Typ wird auf Basis verteilter Daten erstellt. So könnten zum Beispiel Straßenkarten über einen Server und Luftkarten vom einen anderen Server bezogen werden.
- Animiert: Die animierte Map macht Veränderungen über einen bestimmten Zeitraum sichtbar (z.B. das Wetter).
- Echtzeit: Dieser Typ stellt Phänomene nahezu in Echtzeit dar. Daten können über Sensoren an den Server gesendet und bei der nächsten Abfrage oder in regelmäßigen Intervallen aktualisiert werden.
- Personalisierbar: Stil und Symbole können durch den Benutzer angepasst werden. Die Daten können über Filter nach den Wünschen des Benutzers manipuliert werden.
- Interaktiv: Eine interaktive Map fördert Exploration durch Navigation oder die Möglichkeit zusätzliche Informationen anzuzeigen.

- Offen und wiederverwendbar: Diese bieten eine Schnittstelle zur Wiederverwendung wie zum Beispiel die *Google Maps API*.
- Analytisch: Bietet dem Benutzer eine Form von Analyse.
- Kollaborativ: Viele Menschen arbeiten gemeinsam an einer Karte, wie beispielsweise im Fall der *OpenStreetMap (OSM)*.

Eine Web Map ist nicht zwangsläufig einer dieser Klassen zuzuweisen, sondern kann Eigenschaften aus mehreren Klassen aufweisen. Die Differenziertheit einer Web Map ergibt sich aus der Summe ihrer Eigenschaften [Neu08].

Präsentation

Zum Aufrufen einer Web Map ist nicht mehr als ein Browser notwendig. Alle modernen Browser unterstützen JavaScript, welches für die Anforderungen interaktiver Web Maps die notwendigen Eigenschaften mitbringt. Event-Handling, Netzwerk-Requests und Manipulation des Document Object Model (DOM) sind notwendig um einige von den zuvor aufgeführten Eigenschaften interaktiver Karten anzubieten [Neu08]. Zur Implementierung solcher Web Maps benötigt man die entsprechenden Bibliotheken. Einige wichtige Vertreter sind Google Maps, Yahoo Maps und Bing Maps (kommerziell) sowie OpenLayers und Leaflet (open-source). Der Einsatz von Repräsentationen innerhalb solcher Bibliotheken basiert häufig auf einem Schichtenmodell (siehe Abbildung 2.11). In diesem werden grafische Repräsentationen auf hierarchisch angeordneten Ebenen dargestellt. Man unterscheidet zwischen Basisebenen (engl. baselayer) und überlagernden Ebenen (engl. overlay). Es können sowohl Karten als auch Geodaten dargestellt werden [Kra04]. Zu einem Zeitpunkt kann jeweils nur eine Basisebene angezeigt werden, jedoch mehrere Überlagerungen.

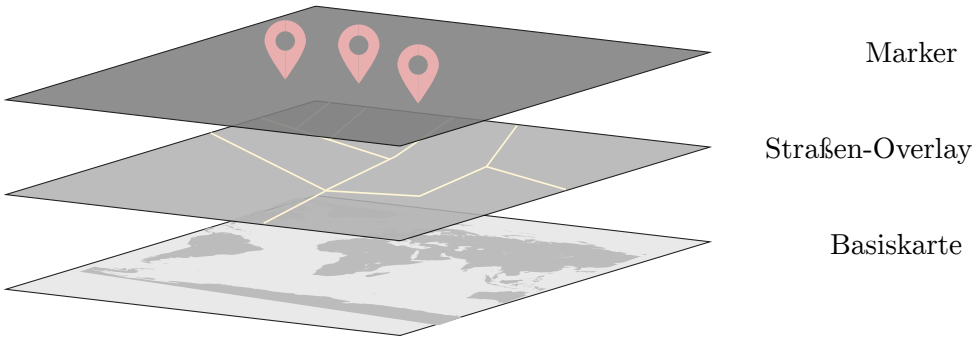

Abbildung 2.11: Schichtenmodell moderner Web Maps

Karten

Die Basis einer Web Map bildet das eingebundene Kartenmaterial. Es stehen zahlreiche zum Teil kostenfreie, zum Teil gegen Bezahlung erhältliche Varianten im World Wide Web zur Verfügung. Die im Folgenden beschriebenen Kartenvarianten wurden auf Grund ihrer Popularität sowie ihrer Relevanz für die Noise Map ausgewählt.

- *Google Maps* - Google Maps stellt als Vorreiter der „wiederverwendbaren“ Maps auch heute noch eine der meist verbreiteten Kartenvarianten dar. Die kostenlose Nutzung ist möglich, wird jedoch seit 2013 durch ein festgesetztes Trafficlimit und diverse Lizenzauflagen beschränkt. So ist die Nutzung des Google Kartenmaterials ausschließlich über die Google API erlaubt.

- *OpenStreetMap (OSM)* - Einer der bekanntesten Anbieter für frei zugängliches Kartenmaterial ist die OSM. Diese ist eine von wenigen Karten, die nicht nur wiederverwendbar, sondern auch zur Veränderung freigegeben ist [Ben10]. Die Geodatenbank hinter der OSM entstand und wächst durch Beiträge der Community. Benutzer zeichnen über GPS ihre Bewegungen auf und können dadurch Flächen und Straßen erschließen [Ben10].

- *Basemap* - Aus der Open Government Data Initiative heraus wurde speziell für den österreichischen Raum die sogenannte Basemap geschaffen. Diese bietet seit dem Jahr 2014 frei verfügbare Verwaltungskarten, welche sowohl privat als auch kommerziell genutzt werden dürfen. Die Basis bilden Verwaltungs-Geodaten der neun Bundesländer, die regelmäßig aktualisiert werden.

Wichtige Faktoren bei der Wahl der Karte sind Qualität und Aktualität. Besonders der Updatezyklus und die Geschwindigkeit, mit der auf Fehler reagiert wird, unterscheiden sich stark. Web-basierte Maps können mehrere Karten unterschiedlicher Anbieter enthalten. Eine könnte beispielsweise die Topographie und eine andere das Straßennetz eines Landes zeigen.

„Slippy Maps“

Um große Downloads zu vermeiden, wird das Kartenmaterial üblicherweise in Form rechteckiger Raster angeboten, die nach Bedarf/Ansicht geladen werden. Diese haben zumeist eine Größe von 256 x 256 Pixeln und liegen oft im JPEG oder PNG Datenformat vor. Wird die Kartenansicht durch den Benutzer verändert, übernimmt die JavaScript Bibliothek das Laden der benötigten Kacheln. Karten, die diese Form der dynamischen Darstellung anbieten, werden „Slippy Maps“ genannt.

Die richtigen Kacheln werden über Parameter in der URL angefordert. Für die OSM sieht diese URL wie folgt aus:

```
http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png
```

s... Karten Stil
z... Zoomlevel
x... X-Koordinate
y... Y-Koordinate

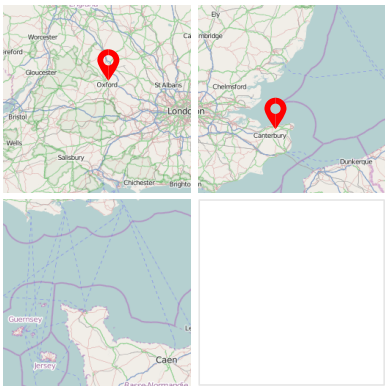

Abbildung 2.12: Map Kacheln und Marker

Tritt der Benutzer in Interaktion mit der Map und verändert den Sichtbereich oder den Zoomlevel, werden alle sichtbaren Raster geladen. Diese Methode verhindert unnötigen Datenverkehr, was besonders im mobilen Einsatz von Vorteil ist.

Geodaten

Im Web Mapping liegen Daten häufig im GeoJSON Format vor. Dieses besteht per Definition aus einem JSON Objekt ({ ... }), welches eine Geometrie, ein geografisches Feature oder eine Sammlung von Features - FeatureCollection - präsentiert [BDD+08]. Verfügbare Geometrie-Typen nach Butler sind:

- *Point* - Ein Punkt auf der Karte, beschrieben durch Koordinaten.
- *LineString* - Eine Linie auf der Karte, beschrieben durch ein Array aus Koordinaten.
- *Polygon* - Ein Polygon, beschrieben durch einen geschlossenen *LineString* mit vier oder mehr Koordinaten.

Neben den geometrischen Eigenschaften können für GeoJSON Objekte auch zusätzliche Eigenschaften (properties) festgelegt werden. In modernen Mapping Bibliotheken können GeoJSON Daten einfach und effizient verarbeitet werden.

Steht die Präsentation von einzelnen Punkten an verschiedenen Stellen der Map im Vordergrund (Geometrie - Point) werden häufig Marker eingesetzt. An den Koordinaten des jeweiligen Features findet sich eine grafische Repräsentation der darzustellenden

```
{
    type":"Feature",
    "geometry":
    {
        "type":"Point",
        "coordinates":[16.3603,48.2109]
    },
    "properties":
    {
    "BEZIRK":1,
    "STRASSE":"Rathauspark/Ring"
    }
}
```


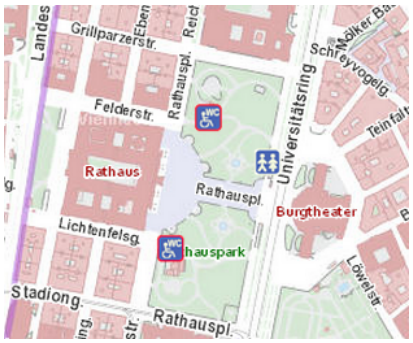

Abbildung 2.13: WC Anlagen im Wiener Rathauspark in GeoJSON und als Marker (Quelle: Stadt Wien - ViennaGIS)

Information wieder. Abbildung 2.13 zeigt Geodaten, welche den Standort einer öffentlichen Toilette beschreiben sowie den Marker, welcher diese darstellt.

Sollen Flächen wie Parks, Gebäudegrundrisse oder Landesgrenzen präsentiert werden, wird die Geometrie mit Polygonen definiert. Abbildung 2.14 zeigt die Fußgängerzone auf der Mariahilfer Straße (orange) sowie die entsprechende GeoJSON Definition. Die Koordinaten, welche die Fläche aufspannen sind aus Platzgründen nicht vollständig angegeben.


```
{
    type":"Feature",
    "geometry":
    {
        "type":"Polygon",
        "coordinates":[16.3528,...,48.1993]
    },
    "properties":
    {
    "ADRESSE":"6., Mariahilfer Straße",
    "ZEITRAUM":"Mo. - Sa. (werkt.) ..."
    }
}
```


Abbildung 2.14: Fußgängerzone auf der Mariahilfer Straße in GeoJSON und als Polygon (Quelle: Stadt Wien - ViennaGIS)

Interaktivität

Die meisten Mapping Bibliotheken inkludieren Funktionen, welche dem Benutzer ermöglichen, mit der Map in Interaktion zu treten. Dazu zählen nach [Kra04]:

- Zoom: Verändern des Zoomlevels.
- Verschieben: Verändern des Kartenausschnitts mittels Drag and Drop.
- Ebenen-Kontrolle: Austausch der angezeigten Karten oder der Geodaten.
- Abfragen: Manipulation des Datensatzes.

Eine wichtige Rolle spielt auch das Abrufen von zusätzlichen Informationen über das Datenmaterial. Kraak fordert in seiner Arbeit [Kra04] eine möglichst reduzierte Darstellung der Daten, um den Benutzer nicht zu überfordern. Zusätzliche Details sollen sich jedoch mittels Klick und Mouseover anzeigen lassen. Diese Forderung entspricht auch dem Kriterium des Maximierens, welches Mazza für ein gutes Visualisierungsdesign aufstellt [Maz09].

2.3.4 Lärmkarten

Lärmkarten sind unter anderem Teil eines 2002 beschlossenen Maßnahmenpakets der Europäischen Union zur Reduktion von Lärmverschmutzung und deren Auswirkungen auf den Menschen [ARR11]. Die Richtlinie 2002/49/EG besagt, dass für Ballungsräume sowie Hauptverkehrsverbindungen strategische Lärmkarten zu erstellen sind [MEGK11]. Diese Karten sollen das Beobachten des Umgebungslärms in städtischen Lebensräumen ermöglichen und dadurch einerseits Aufmerksamkeit schaffen und andererseits als Grundlage für die Entwicklung von Strategien dienen [RCK10]. Mit Hilfe grafischer Repräsentation von Lärmpegeln soll eine Einschätzung der vorliegenden Belastung vorgenommen werden. Diese Information kann nicht nur der Öffentlichkeit dienen sondern auch für Stadt- und Raumplanung eingesetzt werden [ARR11]. Neben Daten aus Beobachtungsstationen innerhalb eines Sensor-Netzwerks können Lärmkarten auch auf der Basis von Kalkulationen des Verkehrsflusses oder der durchschnittlichen Geschwindigkeit von Fahrzeugen erstellt werden [MEGK11]. Ein Beispiel für so eine Karte stellt die Lärmkarte des Bundesministeriums für Land und Forstwirtschaft, Umwelt und Wasserwirtschaft (BMLFUW) dar, die in Kapitel 2.4.1 beschrieben wird.

Visualisierungen auf der Basis statischer Daten sind häufig das Resultat großer Sensornetzwerke, die mittels fest installierter Messstationen den Schalldruckpegel an Verkehrsknotenpunkten aufzeichnen [RCK10]. Bedingt durch langsame Updatezyklen und den kostenintensiven Betrieb, zeichnet diese Methode jedoch ein eher begrenztes Bild der tatsächlichen individuellen Lärmbelastung [KSFN08].

Um dieser Kritik am bestehenden System zu begegnen, nutzen Projekte zunehmend die in Kapitel 2.1.2 beschriebene Technik des *Participatory Sensing*. Der Einsatz von Smartphones als mobile Sensoren führt zu einer enormen räumlichen Abdeckung bei geringen Kosten [CRKH11]. Einige dieser Projekte sowie deren Umsetzung werden im folgenden Kapitel im Detail beschrieben.

2.4 Stand der Technik

Im Folgenden werden fünf Projekte beschrieben, die sich mit der Visualisierung von Lärm befassen. Diese werden nach folgenden Kriterien beleuchtet:

- Sensor Komponente
- Server Komponente
- Übertragene Daten
- Präsentation
- Interaktion

Die Lärmkarte des BMLFUW stellt das einzige Projekt dar, in welchem kein *Participatory Sensing* zur Anwendung kommt und soll daher als Beispiel für eine „klassische“ Lärmkarte dienen.

2.4.1 Straßenlärmkartierung 2007

Das BMLFUW stellt neben zahlreichen anderen Geovisualisierungen auch mehrere Lärmkarten über die Plattform *Lärminfo.at* zur Verfügung. Die Straßenlärmkarte 2007 basiert auf an Hand von Verkehrsaufkommen, Geschwindigkeit und der Beschaffenheit des Geländes berechneten Daten [BfLuF13].

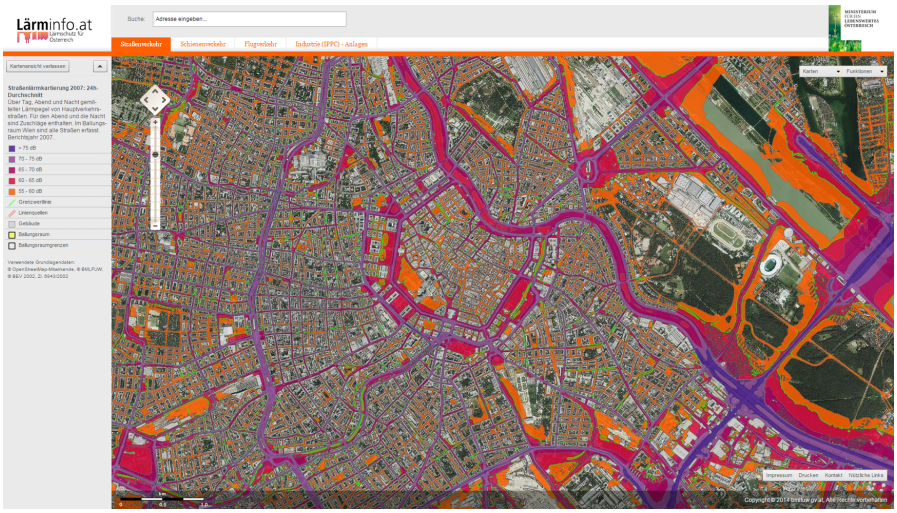

Abbildung 2.15: Screenshot der Straßenlärmkartierung 2007 des BMLFUW (Quelle: `www.laerminfo.at` abgerufen April-2015)

Sie zeigt die Lärmbelastung entlang verkehrsintensiver Straßen in Österreich. Die Präsentation der Daten erfolgt unter Einsatz der *OpenLayers API* mit Karten des

BMLFUW, des Bundesamts für Eich- und Vermessungswesen (BEV) und der OSM. Der Schalldruckpegel wird in fünf Intervalle unterteilt wobei lila (> 75dB) das lauteste und orange (55 - 60dB) das leiseste Intervall darstellen. Möglichkeiten der Interaktion sind durch Zoom, Verschieben, Ebenen-Kontrolle sowie Funktionen für Analyse und Vermessung gegeben. Zusätzliche Informationen können per Mouseover - Latitude und Longitude - und per Klick das zuständige Magistrat abgerufen werden. Auch auf Grund langsamer Updatezyklen und statischer Daten eignen sich die Lärmkarten des BMLFUW nicht zur Einschätzung der individuellen Exposition [BfLuF13].

2.4.2 NoiseSpy

Eines der ersten Projekte, welches den Ansatz über *Participatory Sensing* verfolgte, ist NoiseSpy [Kan09]. Ziel war es, Nokia Telefone als mobile Sensoren für Lärmmessungen einzusetzen. Daher wurden 2007, im Rahmen einer Evaluation, mehrere Fahrradkuriere mit einer Applikation für das mobile Betriebssystem *Symbian* ausgerüstet [Kan09]. Diese erfasst sekündlich den Schalldruckpegel in Dezibel sowie den Standort über einen externen GPS-Receiver. Die Daten werden zusammen mit Identifikationsmerkmalen (ID, IMEI und Username) via HTTP an einen Server übertragen. Dieser bietet Programmlogik (PHP) zur Verarbeitung der Daten sowie eine Datenbank (*PostGreSQL*) zur Aufbewahrung. Angebotene Services sind unter anderem die Umwandlung von Daten zu XML und Keyhole Markup Language (KML) sowie der Export zur Weiterverarbeitung. Im Rahmen des Projekts wurden sowohl Visualisierungen mit der *Google Maps API* als auch mit der *Google Earth* Software durchgeführt (siehe Abbildung 2.16) mit dem Ziel, die Aufmerksamkeit gegenüber der eigenen Lärmbelastung zu erhöhen. Mit dem Usernamen und einem Passwort konnten die Probanden im Rahmen der Evaluation ihre Routen nachvollziehen, einzelne Routen hervorheben und die angezeigten Routen nach Datum und Uhrzeit filtern. Messungen mit niedrigem Schalldruckpegel wurden in hellem blau und jene mit hohem Dezibelwert in rot dargestellt.

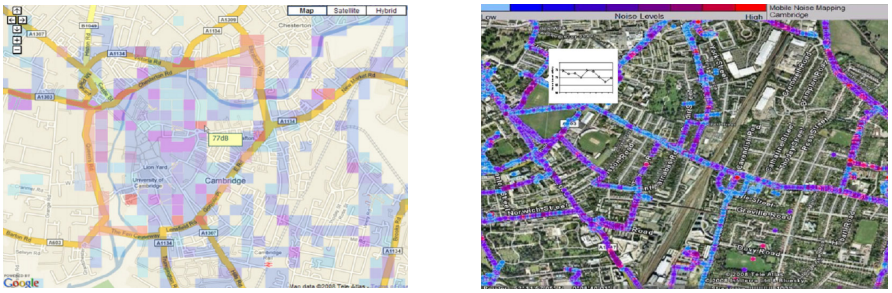

Abbildung 2.16: Screenshot der NoiseSpy Map und der NoiseSpy Google Earth Darstellung (Quelle: [Kan09])

Die Arbeit von Kanjo et al. [Kan09] sieht auch eine zukünftige Veröffentlichung der Visualisierung vor, welche zum Zeitpunkt dieser Arbeit jedoch nicht im Web gefunden werden konnte.

2.4.3 NoiseTube

Bereits im Jahr 2008 wurde im Rahmen des Projekts NoiseTube, auf der Basis von durch mobile Benutzer gesammelten Daten, individuelle Lärmbelastung visualisiert. Eine mobile Anwendung für *Android* und *iOS* überträgt Daten an einen Webserver. Neben dem Schalldruckpegel, dem Standort und der Zeit werden auch Tags erfasst, über die der Benutzer seine Situation näher beschreiben kann. Eine Web Applikation basierend auf *Ruby on Rails* sowie eine *MySQL* Datenbank fungieren als Backend. Die Visualisierung

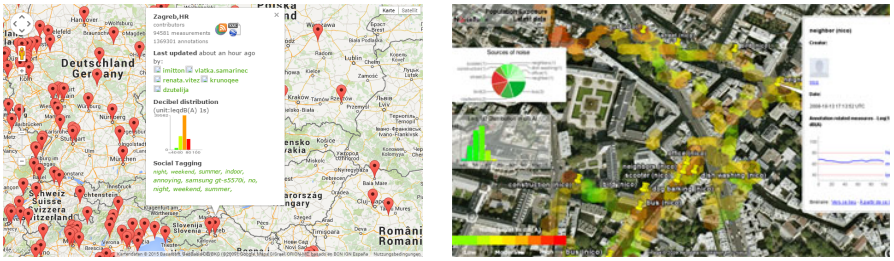

Abbildung 2.17: Screenshot der NoiseTube Web Map und der Google Earth Darstellung (Quelle: `www.noisetube.net` abgerufen April-2015)

basiert auf der Software *Google Earth* und ermöglicht Benutzern lokal an den eigenen Daten zu arbeiten. Messungen werden in den Farben grün - niedrige Intensität - bis rot - hohe Intensität - dargestellt. Ein Überblick über alle Messungen wird durch eine Web Map basierend auf der *Google Maps API* geboten. Per Klick auf einen der Standard-Marker können zusätzliche Informationen wie die Anzahl der Messungen, die Häufigkeiten der vorkommenden Schalldruckpegel in Form eines Balkendiagramms sowie die letzten Tags abgerufen werden. Die Abhängigkeit von *Google Earth* und die Notwendigkeit die Daten lokal zu speichern, hemmen den Benutzer möglicherweise bei der spontanen Erforschung der Daten. Während bei der Web Map wenig Wert auf Design gelegt wurde bietet die Darstellung in *Google Earth* eine dynamische Legende, verschiedene Ebenen zur Analyse der Daten sowie die Möglichkeit, die zeitliche Entwicklung der Daten nachzuvollziehen. NoiseTube ist ein open-source Projekt und stellt sowohl den eigenen Quellcode als auch die gesammelten Daten über eine Schnittstelle zur Verfügung.

2.4.4 Ear-Phone

Ein weiteres *Participatory Sensing* Projekt aus dem Jahr 2010 nennt sich Ear-Phone [RCK10]. Eine mobile Applikation für die Geräte Nokia N95 und HP iPAQ dient als Sensor und misst den Schalldruckpegel in regelmäßigen Intervallen. Gemeinsam mit den GPS Koordinaten und einem Zeitstempel werden die Daten bei bestehender WiFi Verbindung an einen zentralen Server übertragen. Basierend auf der Skriptsprache PHP wird eine Rekonstruktion der Lücken zwischen den gemessenen Werten durchgeführt. Die Aufbewahrung erfolgt mittels *MySQL* Datenbank. Für Zugriffe auf die Daten wird im Prototyp eine Lookup-Tabelle verwendet. Im Rahmen des Projekts erfolgte eine Visualisierung mit Hilfe der *Google Maps API* (siehe Abbildung 2.18). Die Präsentation der Messwerte wurde an Hand farbiger Routen realisiert. Für die Darstellung wurden zehn Intervalle zwischen 52 und 70dB in unterschiedlichen Farben gewählt. Durch eine umfangreiche Evaluation sowie mehreren Simulationen kamen die Autoren zum Ergebnis, dass die Messungen der mobilen Geräte eine hinreichende Genauigkeit besitzen [RCK10]. Eine öffentliche Visualisierung der erhobenen Daten ist im Zuge des Projekts nicht realisiert worden.

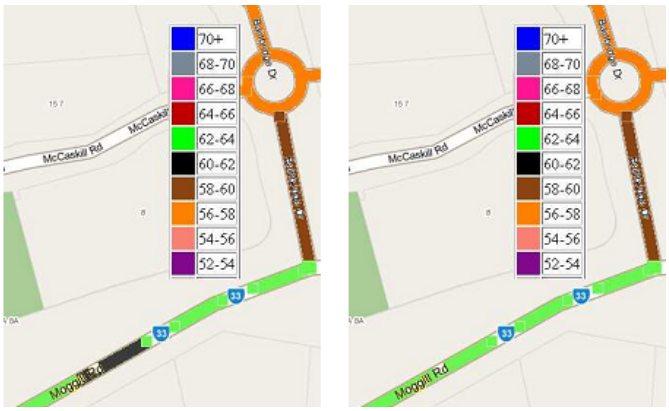

Abbildung 2.18: Screenshot der EarPhone Map mit Daten von vier (li.) und sechs Personen (re.) (Quelle: [RCK10])

2.4.5 Da_sense

Die TU Darmstadt hat 2011 ein Projekt initiiert, welches eine mobile Applikation genannt „NoiseMap“ sowie eine Webplattform mit dem Namen „da_sense“ umfasst. Erstere ist für *Android* und *iOS* verfügbar und überträgt den Schalldruckpegel sowie die momentane Position als JSON String. Die übertragenen Daten und deren Sichtbarkeit können von den Benutzern über die Webplattform verwaltet werden [SBS11]. Da_sense visualisiert alternativ zur Lautstärke auch Helligkeit, Temperatur und Luftfeuchtigkeit. Die Map basiert auf der *Leaflet API* und Kartenmaterial von *Mapbox*. Die Messungen werden durch ein Heatmap Overlay als farbige Hexagone von grün (leise) bis rot (laut) dargestellt.

Zusätzliche Informationen können sowohl via Mouseover angezeigt werden, als auch per Doppelklick auf eine der Repräsentationen. Diagramme geben einen Überblick über die Entwicklung der Lautstärke über die Zeit und lassen auch die Auswertung über ein bestimmtes Areal zu. Dem Open Data Prinzip folgend bietet da_sense seine Daten nicht nur frei verfügbar an, sondern ermöglicht auch den Upload über die mobile Applikation an fremde Plattformen [SBS11].

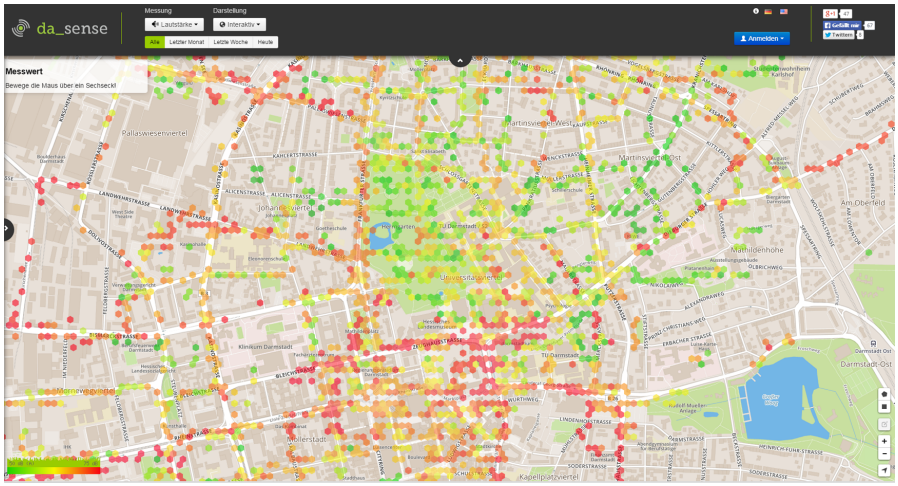

Abbildung 2.19: Screenshot der Lärmkarte da_sense (Quelle: `www.da-sense.de` abgerufen April-2015)

Kapitel 2 bildet eine Basis zum Verständnis der weiteren Arbeit und beinhaltet zu jeder der drei Säulen der Noise Map theoretische Konzepte aus der Literatur. Im ersten Teil wurden Grundbegriffe der Lärmmessung sowie der Ansatz des *Participatory Sensing* eingeführt. Des weiteren wurden die Prinzipien des Representational State Transfer (REST) beschrieben, nach denen die Schnittstelle der Java Web Applikation definiert ist. Im Zentrum der Konzepte rund um die Noise Map steht Visualisierung, beginnend mit einer allgemeinen Einführung in den Bereich bis zum Prozess des Web Mappings. Die Beschreibung inhaltlich verbundener Projekte bildete den Abschluss der Theorie. Auf diesen Konzepten aufbauend, wird nun im Folgenden die technische Umsetzung des Prototyps beschrieben.

KAPITEL 3

Implementierung

Dieses Kapitel behandelt Entwurf und Umsetzung der drei in Kapitel 1.2 beschriebenen Säulen der Noise Map. Eingangs werden im Kapitel 3.1 Konzept und Architektur des Prototyps vorgestellt. Auf die technischen Erweiterungen der Noise-O-Meter Applikation wird in Kapitel 3.2 eingegangen. Die Java Web Applikation, in deren Zentrum die Schnittstelle für Empfang und Verarbeitung der Daten steht, wird in Kapitel 3.3 beschrieben. Im Fokus des letzten Kapitels 3.4 steht die Noise Map selbst.

3.1 Entwurf

Im Folgenden werden zuerst jene Benutzergruppen beschrieben, welche ein Interesse an der Noise Map haben. Auf dieser Basis definiert der Abschnitt 3.1.2 notwendige Eigenschaften des Prototyps. Mit Hilfe eines Anwendungsfalldiagramms werden in Abschnitt 3.1.3 die konkreten Interaktionsmöglichkeiten für den Benutzer beschrieben. Den Abschluss bilden die Architektur des Prototyps sowie eingesetzte Entwurfsmuster in Abschnitt 3.1.4.

3.1.1 Stakeholder

Im vorliegenden Projekt werden verschiedene Stakeholder unterschieden, die Interesse an den durch die Noise Map visualisierten Informationen haben.

- *Benutzer* - Der Prototyp der Noise Map ist als Web Applikation frei zugänglich und daher für Besucher mit unterschiedlichsten Vorkenntnissen und Kompetenzen erreichbar. Da Lärmbelastung grundsätzlich jeden betreffen kann, ist die Noise Map potenziell für alle Menschen nützlich. Von besonderem Interesse könnten Orte mit persönlicher Relevanz sein, wie der eigene Wohnort oder Arbeitsplatz. „Live“ zu beobachten, wie neue Lärmquellen auf der Karte erscheinen und dadurch Zeuge eines Ereignisses zu sein, welches durch einen anderen (anonymen) Menschen initiiert wurde, kann für diese Benutzergruppe ebenfalls reizvoll sein.

- *Mobile Benutzer* - Mobile Benutzer sind Teilnehmer, die mit dem Noise-O-Meter Messungen durchführen und diese zur gemeinsamen Nutzung teilen. Sie besitzen durch die Applikation bereits Vorkenntnisse über die Thematik und sind dadurch von den normalen Benutzern zu unterscheiden. Für diese Benutzergruppe könnte die Motivation besonders hoch sein, die eigenen Messungen auf der Karte nachzuvollziehen und das Ergebnis der kollektiven Bemühungen zu erforschen. Darüber hinaus ist vermutlich die Live-Visualisierung von besonderem Interesse, da hier Teile des durch das *Participatory Sensing* entstehenden Sensornetzwerks dargestellt werden. Des weiteren könnte die Noise Map als Verwaltungszentrale für die hochgeladenen Daten der mobilen Benutzer dienen.

- *Experten und Business-Stakeholder* - Experten und Business-Stakeholder haben ein Interesse, das über das bloße Erforschen der Visualisierung hinaus geht. Deskriptive Statistik sowie die Filterung der Daten nach bestimmten Kriterien sind für diese Benutzergruppe wichtig. Eine Live-Visualisierung der neu hinzukommenden Lärmquellen könnte bei der Analyse der Daten eher störend oder sogar hinderlich sein. Besondere Ansprüche an die Qualität der gemessenen Daten machen es notwendig, ausgewählte Teilnehmer aus der anonymen Datenmenge identifizieren zu können.

Diese Beurteilung der Erwartungen und Ansprüche der Stakeholder sind Teil der Anforderungsanalyse, die im folgenden Abschnitt beschrieben wird.

3.1.2 Anforderungen

Die Anforderungen an den Prototyp ergeben sich aus den Bedürfnissen der Stakeholder sowie aus domänenspezifischen Standards und Normen. Man unterscheidet einerseits funktionale Anforderungen, die festlegen was der Prototyp können muss, und andererseits nichtfunktionale Anforderungen, welche Eigenschaften des Prototyps festlegen. Jede Anforderung wird zur Bearbeitung im Laufe des Projekts mit einer ID versehen.

3.1.3 Anwendungsfälle

Anwendungsfälle definieren die Interaktionen eines Benutzers mit dem System, die notwendig sind, um ein bestimmtes Ziel zu erreichen. Im vorliegenden Projekt stellen der Noise-O-Meter und die Noise Map die Schnittstellen für den Benutzerzugriff dar. Anwendungsfälle können unterschiedlich detailliert beschrieben werden. Im Folgenden sollen Detailinformationen über Systemprozesse zu Gunsten der Lesbarkeit nicht abgebildet werden. Das Anwendungsfalldiagramm in Abbildung 3.1 stellt die Interaktionsmöglichkeiten der zuvor definierten Akteure mit dem Prototyp dar.

Funktionale Anforderungen	
Id	Anforderung
01	Der Prototyp soll Lärmmessungen in einer Datenbank speichern
02	Die Lärmmessungen müssen einem Benutzer eindeutig zuordenbar sein
03	Der Prototyp soll eine Schnittstelle für die Verarbeitung der Daten zur Verfügung stellen
04	Der Prototyp soll eine aggregierte Visualisierung der gesammelten Messungen enthalten
05	Eine grafische Repräsentation soll einzelne Lärmquellen darstellen
06	Dem Benutzer sollen mehrere Interaktionsmöglichkeiten zur Erforschung der Daten angeboten werden
07	Es soll eine statistische Auswertung der Daten für einzelne Benutzer zur Verfügung gestellt werden
08	Durch kurze Updatezyklen soll der Eindruck einer Echtzeit-Lärmkarte entstehen

Tabelle 3.1: Funktionale Anforderungen im Prototyp

Nichtfunktionale Anforderungen	
Id	Anforderung
01	Der Prototyp soll durch den Benutzer intuitiv zu bedienen sein (Benutzbarkeit)
02	Der Prototyp soll in der Lage sein, eine hohe Anzahl gleichzeitiger Anfragen zu bewältigen (Leistung und Effizienz)
03	Der Prototyp soll über das Hypertext Transfer Protocol kommunizieren (Unterstützung von Standards)
04	Der Prototyp soll Maßnahmen zur Wahrung der Datenintegrität unterstützen (Sicherheitsanforderung)

Tabelle 3.2: Nichtfunktionale Anforderungen im Prototyp

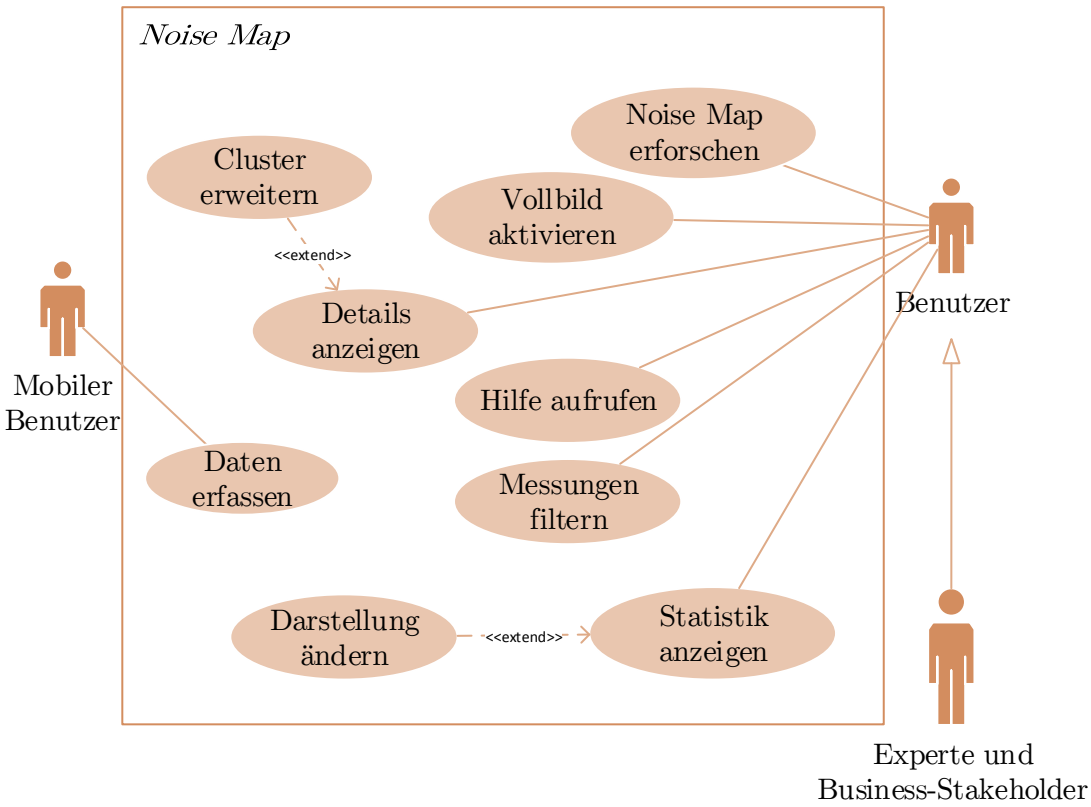

Abbildung 3.1: Anwendungsfälle im Prototyp

Noise Map erforschen	
Akteur	Benutzer
Geltungsbereich	Noise Map
Vorbedingung	Die Web Applikation ist vollständig geladen
Auslöser	Der Benutzer verändert den Kartenausschnitt
Kurzbeschreibung	Der Benutzer erforscht die Karte, indem er die zur Verfügung stehenden Werkzeuge (Zoom, Verschieben) einsetzt
Erweiterungen	Durch die Live-Visualisierung können sich die Daten während des Erforschens verändern
Technologie	Die Map Bibliothek reagiert auf Klick- und Drag-Events sowie Veränderungen des Datensatzes durch Angular-Pusher

Tabelle 3.3: Anwendungsfall - Noise Map erforschen

Vollbild aktivieren	
Akteur	Benutzer
Geltungsbereich	Noise Map
Vorbedingung	Die Web Applikation ist vollständig geladen
Auslöser	Der Benutzer klickt auf den Vollbild-Button
Kurzbeschreibung	Die Karte nimmt den gesamten Bildschirm ein
Technologie	Die Leaflet Fullscreen Bibliothek verändert mittels CSS die Dimensionen der Map

Tabelle 3.4: Anwendungsfall - Vollbild aktivieren

Hilfe aufrufen	
Akteur	Benutzer
Geltungsbereich	Noise Map
Vorbedingung	Die Web Applikation ist vollständig geladen
Auslöser	Der Benutzer klickt auf den Hilfe-Button
Kurzbeschreibung	Ein Hilfe Dialog erscheint, welcher animierte Grafiken zur Bedienung der Statistikwerkzeuge enthält
Technologie	Ein Angular-Bootstrap Modal Window enthält animierte GIF Dateien

Tabelle 3.5: Anwendungsfall - Hilfe aufrufen

Details anzeigen	
Akteur	Benutzer
Geltungsbereich	Noise Map
Vorbedingung	Die Web Applikation ist vollständig geladen
Auslöser	Der Benuter fährt mit der Maus über einen Marker
Kurzbeschreibung	Ein Info-Panel in der rechten oberen Ecke der Karte zeigt zusätzliche Informationen über die Messung(en)
Erweiterungen	Ein Klick auf einen Cluster (Anhäufung von Markern) führt zu einer Übersicht der enthaltenen Marker
Technologie	Event-Handling und die Spiderfy-Funktion der Leaflet Markercluster Bibliothek

Tabelle 3.6: Anwendungsfall - Details anzeigen

Messungen filtern	
Akteur	Benutzer
Geltungsbereich	Noise Map
Vorbedingung	Die Web Applikation ist vollständig geladen
Auslöser	Der Benutzer klickt auf den Filter-Button
Kurzbeschreibung	Eine Sidebar wird eingeblendet, die zwei Datepicker und eine Dropdown-Liste zur Datenmanipulation enthält
Technologie	Leaflet Sidebar Bibliothek und Angular-Bootstrap Datepicker sowie Filter zur Manipulation des Datensatzes

Tabelle 3.7: Anwendungsfall - Messungen filtern

Statistik anzeigen	
Akteur	Benutzer
Geltungsbereich	Noise Map
Vorbedingung	Die Web Applikation ist vollständig geladen
Auslöser	Der Benutzer selektiert ein Areal auf der Karte, in dem sich Marker befinden
Kurzbeschreibung	Eine Sidebar wird eingeblendet, die ein Diagramm sowie statistische Informationen enthält
Erweiterungen	Der Benutzer kann per Klick zwischen Balken- und Tortendiagramm umschalten
Technologie	Leaflet Sidebar Bibliothek, Leaflet Draw um Areale zu markieren, ChartJS zur Darstellung der Diagramme

Tabelle 3.8: Anwendungsfall - Statistik anzeigen

Daten erfassen	
Akteur	Mobiler Benutzer
Bereich	Noise-O-Meter
Vorbedingung	Die Applikation ist installiert und gestartet
Kurzbeschreibung	Der mobile Benutzer startet eine Messung
Auslöser	Der Benutzer berührt den „Messung starten“ Button
Technologie	Schalldruckpegel wird in festgelegtem Intervall gemessen und gespeichert

Tabelle 3.9: Anwendungsfall - Daten erfassen

Bevor auf die technische Umsetzung im Detail eingegangen wird, soll im folgenden Abschnitt die Architektur des Prototyps vorgestellt werden.

3.1.4 Architektur und Entwurfsmuster

Der Entwurf resultiert in einem „Bauplan“ des zu entwickelnden Systems. Ein wesentlicher Bestandteil dieses Bauplans ist die Architektur. Die drei Säulen der Noise Map lassen sich in zwei Klassen unterteilen. Abbildung 3.2 zeigt den Noise-O-Meter und die Noise Map als Clients (Kunden), welche Anfragen an den Server (die Java Web Applikation) stellen und von diesem Antwort erhalten.

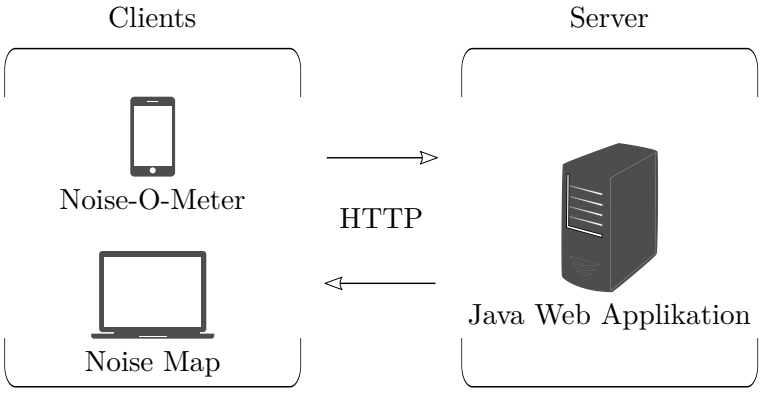

Abbildung 3.2: Client Server Architektur

Um eine klare Trennung der Zuständigkeiten innerhalb des Prototyps zu erreichen, kommt eine sogenannte Schichtenarchitektur zum Einsatz. Diese Struktur ermöglicht es, Abhängigkeiten im Prototyp zu reduzieren, indem eine lose Kopplung zwischen den Schichten sowie eine hohe Kohäsion innerhalb der Schichten forciert wird. Vorteile sind die erhöhte Wartbarkeit und Skalierbarkeit innerhalb des Softwaresystems. Abbildung 3.3 zeigt insgesamt vier Schichten:

1. *Präsentation* - Darstellung der Daten auf der Noise Map beziehungsweise durch das Noise-O-Meter Interface
2. *Security* - In der Sicherheitsschicht erfolgt die Authentifizierung eingehender Anfragen
3. *Programmlogik* - Die Programmlogik setzt sich aus der Kommunikationsschnittstelle sowie den zur Verfügung gestellten Funktionen zusammen
4. *Datenzugriff* - Über die Datenzugriffsschicht erfolgen alle Datenbankaufrufe

Zugriffe zwischen den Schichten erfolgen ausschließlich „top-down“, sodass in einer vertikalen Anordnung (siehe Aufzählung) eine höhere Schicht nur auf eine niedrigere

Schicht zugreifen kann. Die Trennung von Präsentation, Steuerung und Daten bezeichnet man auch als Model View Controller (MVC) Entwurfsmuster.

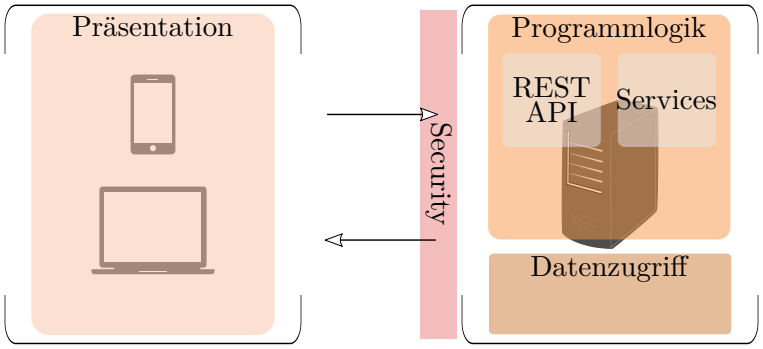

Abbildung 3.3: Schichtenarchitektur

Dependency Injection

Eine weitere Maßnahme, um die Kopplung zwischen Komponenten niedrig zu halten und gleichzeitig eine hohe Wiederverwendbarkeit des Programmcodes innerhalb der Anwendung zu gewährleisten, ist die sogenannte Dependency Injection (DI). Dieses Muster ist Teil vieler Anwendungsframeworks und ermöglicht das Injizieren von Objekten über Konstruktoren, Setter oder Interfaces. So können zum Beispiel Datenquellen, Data-Access-Objekte oder Utility-Klassen eingesetzt werden, ohne diese statisch in einem Service zu verankern.

Data Access Object

Das Data Access Object (DAO) Muster hat zum Ziel jeglichen Datenzugriff zusammenzufassen und über ein Interface zur Verfügung zu stellen. Diese Kapselung bewirkt, dass der Datenzugriff klar geregelt (Trennung der Zuständigkeiten) und versteckt abläuft. Das für eine bestimmte Datenquelle definierte DAO wird durch die Definition des Interfaces austauschbar. So können bei geänderten Anforderungen leicht Adaptionen in der Datenschicht vorgenommen werden.

Im Folgenden wird auf die technischen Umsetzung der drei Säulen der Noise Map, beginnend mit dem Noise-O-Meter eingegangen.

3.2 Noise-O-Meter

Die Noise-O-Meter Applikation, entwickelt von SOPHISYSTEMS und der TU Wien bildet den Ausgangspunkt für die Entwicklung des vorliegenden Prototyps. Die Applikation musste im Zuge des Projekts um drei Funktionen erweitert werden.

3.2.1 Identifikation

Um die Verwaltung der übertragenen Messungen zu ermöglichen, muss jede Messung ihrem Besitzer eindeutig zugeordnet werden können. Zu diesem Zweck wird gemeinsam mit der Messung eine eindeutige Identifikationsnummer (ID) in die Datenbank gespeichert. Zur Erzeugung dieser eindeutigen ID wird die statische Methode *randomUUID()* aus der *java.util.UUID*-Klasse eingesetzt. Bei der ersten Übertragung wird auf dem Smartphone eine Datei erzeugt, welche die erzeugte ID beinhaltet. Bei allen weiteren Übertragungen wird auf die ID aus der Datei zurückgegriffen. Diese Möglichkeit der Identifikation besteht solange der Benutzer das gleiche Gerät benutzt. Die Messungen können dadurch einer Noise-O-Meter Installation zugeordnet werden, obwohl der Benutzer selbst anonym bleibt. Die Privatsphäre wird geschützt, da nicht auf sensible Daten wie IMEI oder die *Android* ID zurückgegriffen wird. Des weiteren legt diese Methode den Grundstein für zukünftige Funktionen, welche es den Noise-O-Meter Benutzern ermöglichen könnten, ihre Daten online zu verwalten.

3.2.2 Standortbestimmung

Für den Einsatz des Noise-O-Meters zur Beurteilung der eigenen Lärmbelastung über die Zeit ist die Standortbestimmung nicht zwingend notwendig. Möchte der Benutzer aber zur Noise Map beitragen oder seine eigenen Messungen auf der Noise Map nachvollziehen, muss zusätzlich zum Messergebnis der Standort erhoben werden. Zu diesem Zweck wird die in Kapitel 2.1.3 beschriebene *Google Location API* eingesetzt.

Es müssen zwei Voraussetzungen erfüllt sein, um den Standort erfolgreich zu erheben. Einerseits muss der Benutzer *Google Play Services* auf seinem Gerät verwenden und andererseits muss die Standortbestimmung auf dem *Android*-Gerät erlaubt sein. Beim Start der Applikation wird über die Klasse *GooglePlayServicesUtil* die Nutzung von *Google Play Services* auf dem jeweiligen *Android*-Gerät überprüft. Verläuft dieser Vorgang erfolgreich wird ein *LocationClient*-Objekt erstellt und verbunden. Um die Standorteinstellungen des Benutzers auf seinem Gerät abzufragen, muss zu einem Umweg über das *Android Location Framework* gegriffen werden. Unter Einsatz der *android.location.LocationManager*-Klasse kann festgestellt werden, ob und in welchem Maße der Benutzer eine Standortbestimmung auf seinem Gerät zulässt. Ist diese gänzlich deaktiviert, so erscheint eine Aufforderung in Form eines *android.app.Dialog* Fensters, die den Benutzer auf Wunsch direkt zu den Standorteinstellungen weiterleitet. Die Methode *startPeriodicLocationUpdates()* wird ausgeführt sobald der *LocationClient* erfolgreich verbunden ist. Innerhalb dieser wird ein neuer *LocationRequest* definiert, ein Intervall für die Aktualisierung sowie die gewünschte Genauigkeit festgelegt. Die Noise-O-Meter Applikation versucht während des Betriebs stets einen möglichst genauen Standort bereit zu halten, um diesen im Fall einer Messung bereitstellen zu können. Aktualisierungen werden von einem *BroadcastReceiver* empfangen und als *android.location.Location*-Objekt „currentBestLocation“ gespeichert.

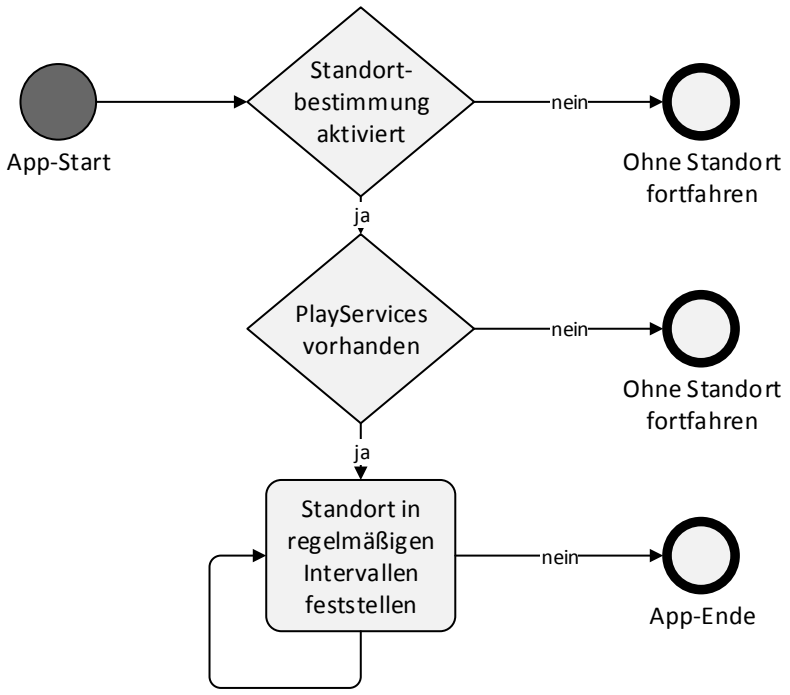

Abbildung 3.4: Lifecycle-Diagramm der Standortbestimmung

3.2.3 Datenübertragung

Die Datenübertragung wird einmal beim Start der Applikation und danach in einem fest definierten Intervall ausgeführt. Die *android.app.AlarmManager*-Klasse wird eingesetzt um halbstündig einen *android.app.PendingIntent* auszulösen, welcher mit Hilfe eines *android.content.BroadcastReceiver* empfangen wird. Wird ein *PendingIntent* im *BroadcastReceiver* entdeckt, so wird das *android.app.Service* UploadService aktiviert. Abbildung 3.5 zeigt die Prozesse, welche in diesem Service durchlaufen werden. Upload-Task ist eine Erweiterung der *android.os.AsyncTask*-Klasse und wird eingesetzt, um Aufgaben im Hintergrund vollständig abzuarbeiten. Der Upload selbst erfolgt mittels des *org.apache.http.client.HttpClient*, der die notwendigen Methoden zur Übertragung über HTTP zur Verfügung stellt. Die Daten selbst werden in JSON gebracht und als String versendet. Um den Übertragungsaufwand so gering wie möglich zu halten, werden die Messungen zu einem Array zusammengefasst. Zusätzlich zu den Messungen werden im „Authorization Header“ an jede Übertragung auch die verschlüsselten Benutzerdaten angefügt. Nach Ausführung der HTTP POST Anfrage wird die Rückmeldung des Servers (der Java Web Applikation) empfangen. Lautet diese „201 Created“ wird eine erfolgreiche Übertragung angenommen und die übertragenen Messungen entsprechend gekennzeichnet. Bei allen anderen Rückmeldungen verbleiben die Messungen als „ungesendet“ im System.

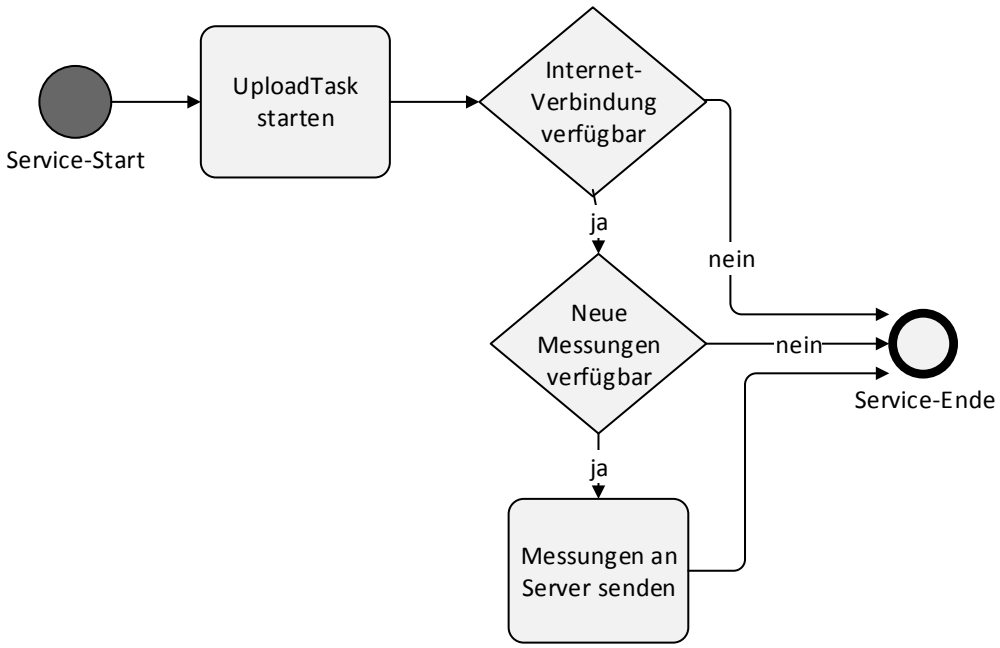

Abbildung 3.5: Lifecycle-Diagramm der Datenübertragung

3.3 Java Web Applikation

Die Java Web Applikation (Backend) bildet die Schaltzentrale des Prototyps und trennt die Programmlogik von den für den Benutzer sichtbaren Elementen (Frontend). Die Bereitstellung einer Schnittstelle nach den in Abschnitt 2.2 beschriebenen Prinzipien ermöglicht die Kommunikation der in Abbildung 3.2 dargestellten Komponenten. Zu den Aufgabenbereichen des Backends gehören:

- Empfang der Daten von den mobilen Clients (Noise-O-Meter Benutzern)
- Aufbewahrung der Daten in einer Datenbank
- Verarbeitung der Daten (Serialization etc.)
- Zurverfügungstellung der Daten für die Noise Map

Die Implementierung des Backends erfolgt in der Programmiersprache Java unter Verwendung des *Spring Web Application Frameworks*. Für die Schnittstelle ist das *Jersey RESTful Web Services Framework* eingesetzt worden, welches die Umsetzung der REST Architektur ermöglicht. Die Persistenz erfolgt mit Hilfe der *Java Persistence API* (JPA), dem *Hibernate Framework* sowie einer *PostgreSQL* Datenbank.

Im Folgenden soll detailliert auf die Umsetzung wichtiger Bereiche des Backends eingegangen werden. Im Fokus des ersten Abschnitts steht die REST Schnittstelle, gefolgt von der Datenzugriffsschicht. Diese umfasst sowohl die Definition der Datenbank als auch jeglicher Klassen, welche darauf Zugriff haben. Abschließend werden Maßnahmen bezüglich Autorisierung und Authentifizierung beschrieben.

3.3.1 REST Schnittstelle

Die Definition der REST Schnittstelle verläuft in insgesamt vier Schritten.

1. Identifikation der Ressourcen
2. Modellierung der Uniform Resource Identifier
3. Definition des Datenformats
4. Zuweisung der HTTP Methoden

Unter Ressourcen versteht man Datenstrukturen, welche über die Schnittstelle übertragen werden sollen. Im vorliegenden Prototyp sind dies Datenstrukturen, die einerseits die Messungen und andererseits die Benutzer darstellen.

Damit das Prinzip der Adressierbarkeit (siehe 2.2.1) erfüllt ist, muss jede Ressource über eine eindeutige Adresse URI ansprechbar sein. Zu diesem Zweck wird folgendes Schema eingeführt, durch das alle Ressourcen über eine Adresse bestehend aus den Teilen **/api** (engl. Application Programming Interface) und **/v1** (Version 1) erreichbar sind.

```
/api/v1/measurements
/api/v1/measurements/{id}
/api/v1/users
/api/v1/users/{id}
```

Durch das Anhängen der ID **{id}** an die Adresse, lassen sich einzelne Ressourcen adressieren.

Das Format in dem die Ressourcen übertragen werden ist vom jeweiligen Kommunikationspartner abhängig. Da die Noise Map auf einem JavaScript Framework basiert wurde für den Prototyp die JavaScript Object Notation gewählt. Eine Messung sieht im JSON Format folgendermaßen aus:

```
{
"id":1,
"value":40,
"lon":16.361014,
"lat":48.214553,
"timestamp":1419025055,
"owner": 33
}
```

Abschließend muss festgelegt werden, welche Operationen für die Ressourcen angeboten werden sollen. Durch die Beschränkung auf die HTTP Methoden GET, POST, PUT, DELETE ist die Forderung nach einem uniformen Interface (siehe 2.2.1) erfüllt. Der Prototyp soll die Möglichkeit bieten, alle Messungen auf einmal abzurufen, um diese beispielsweise auf der Noise Map darzustellen. Zu diesem Zweck wird die *GET* HTTP-Methode an die **/measurements** Adresse definiert. Darüber hinaus sollen mehrere Messungen auf einmal empfangen werden können, wenn eine Anfrage mittels *POST* HTTP-Methode an die **/measurements** Adresse erfolgt. Für die Verwaltung der Daten ist es zudem hilfreich, wenn einzelne Messungen angesehen oder gelöscht werden können. Tabelle 3.10 zeigt alle im Prototyp realisierten HTTP Operationen.

Angebotene Schnittstellen		
Ressource	URI	Methode
Alle Messungen abrufen	/measurements	GET
Eine einzelne Messung abrufen	/measurements/{id}	GET
Messung(en) hinzufügen	/measurements	POST
Eine einzelne Messung löschen	/measurements/{id}	DELETE
Alle Benutzer abrufen	/users	GET
Einen einzelnen Benutzer abrufen	/users/{id}	GET
Einen einzelnen Benutzer löschen	/users/{id}	DELETE

Tabelle 3.10: HTTP Methoden im Prototyp

Implementierung

Java-Klassen bilden die zuvor identifizierten Ressourcen im System ab. Abbildung 3.6 zeigt das Objektmodell der Klassen *User.java* und *Measurement.java*. Diese Datenstrukturen bilden die Basis für das sogenannte *ObjectMapping*, bei dem Java-Objekte automatisch in JSON übersetzt werden können und umgekehrt.

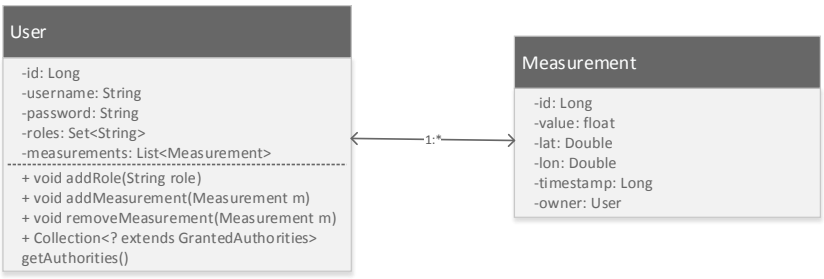

Abbildung 3.6: Objektmodell der identifizierten Ressourcen

Die Klassen *UserResource.java* und *MeasurementResource.java* setzen die zuvor definierten Schemata zur Adressierung der Ressourcen um und enthalten die in Tabelle 3.10 aufgeführten Operationen. Dies geschieht mit Hilfe sogenannter *Annotations*, welche das *Jersey Framework* zur Verfügung stellt. Zusätzliche Anmerkungen in den Java-Klassen erlauben die Definition von Pfaden (*@Path*), Operationen (*@GET*) und Datenformaten (*@Produces*, *@Consumes*). Folgender Programmcode überträgt auf eine GET-Anfrage an den **/measurements** Pfad eine Liste aller Messungen als JSON String:

```
@Path("/measurements")
public class MeasurementResource {

  @GET
  @Produces(MediaType.APPLICATION_JSON)
  public String getAllMeasurements(...) { ... }

  ...

}
```

Die Repräsentationen der Ressourcen werde über die Annotations *@Produces* und *@Consumes* direkt an den Java Methoden definiert. Dadurch wird festgelegt, welches Format von der Methode erwartet beziehungsweise zurückgegeben wird. Benötigt man eine Darstellung, welche von der Struktur der Modelle abweicht, müssen benutzerdefinerte *Serializer* eingesetzt werden. Dies ermöglicht die flexibel Übertragung von Messungen in JSON als auch in GeoJSON. *Serializer* werden den Ressourcen-Klassen über die *Spring Dependency Injection* zur Verfügung gestellt.

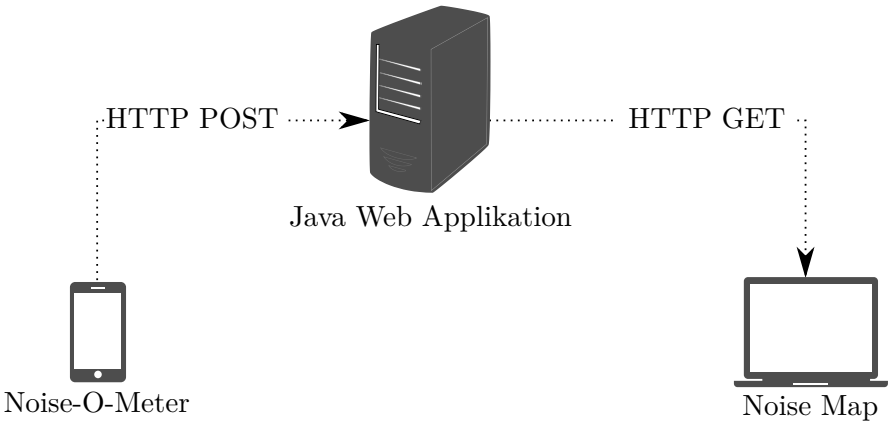

Abbildung 3.7: HTTP Kommunikation im Prototyp

Die Ressourcen-Klassen enthalten wenig Programmlogik und delegieren die meisten Aufgaben an die Datenzugriffsschicht. Diese wird im folgenden Abschnitt beschrieben.

3.3.2 Datenzugriffsschicht

Der Prototyp erfordert eine Datenbank, welche eine hohe Anzahl gleichzeitiger Datenübertragungen seitens der Noise-O-Meter Benutzer sowie gleichzeitige Datenabfragen über die Noise Map bewältigen kann. Zu diesem Zweck wird die etablierte open-source Datenbank *PostgreSQL* eingesetzt. Diese bietet großteils SQL konformen Zugriff, Mechanismen zur Behandlung gleichzeitiger Anfragen sowie zahlreiche Erweiterungsmöglichkeiten.

Datenbankmodell

Abbildung 3.8: Datenbank ER-Diagramm

Der Einsatz der *Java Persistence API* ermöglicht die objektrelationale Abbildung von Java-Objekten in die Datenbank. Den bereits erwähnten Klassen *Measurement.java* und *User.java* kommt dabei eine Mehrfachfunktion zu. Sie dienen sowohl als Modell für die REST Schnittstelle als auch als Modell für die Datenbank. Abbildung 3.8 zeigt das Ergebnis dieses Abbildungsprozesses.

Erneut werden Annotations eingesetzt, um direkt im Programmcode zusätzliche Funktionen zu definieren. Der folgende Programmcode zeigt den Einsatz der Annotations in der *Measurement.java*-Klasse.

@Entity definiert, dass es sich bei dieser Java-Klasse um ein Datenbank-Objekt handelt, welches in die in *@Table* definierte Tabelle in der Datenbank abgebildet werden soll. *@NamedQueries* lässt die Definition diverser Datenbankabfragen direkt im Java-Objekt zu. Über die Annotations *@Id* und *@GeneratedValue* wird festgelegt, dass es sich beim Feld *Id* um einen Primärschlüssel handelt, dessen Inhalt automatisch generiert werden soll. Dadurch wird jedem neuen Measurement-Objekt, dass in die Datenbank geschrieben wird automatisch eine eindeutige fortlaufende ID zugeteilt, über die es angesprochen und abgerufen werden kann. Jedes Measurement-Objekt ist genau einem User-Objekt zugeordnet. Ein User-Objekt kann jedoch in Beziehung zu vielen Measurement-Objekten stehen. Diese *@ManyToOne* Beziehung wird in der Datenbank abgebildet, indem jede Messungen ein Feld (*@JoinColumn*) mit der Id ihres Besitzers hat.

```
@Entity
@Table(name = "measurements")
@NamedQueries({
  @NamedQuery(name = "Measurement.findAll",
      query = "SELECT m FROM Measurement m")
})
public class Measurement implements BaseEntity {

  @Id
  @GeneratedValue
  private Long id;
  public Long getId() { return id; }
  protected void setId(Long id) { this.id = id; }

  ...

  @ManyToOne
  @JoinColumn(name="owner")
  private User owner;

}
```

Data Access Objects

Alle Zugriffe auf die Datenbank erfolgen aus der Datenzugriffsschicht, welche den anderen Schichten die Datenbankfunktionen über ein Interface zur Verfügung stellt. Dies wird mit Hilfe des Data Access Object (DAO) Entwurfsmusters gewährleistet. Java-Klassen vereinen Methoden für die Datenbankzugriffe mit dem Ziel, eine Schnittstelle für den Datenbankzugriff bereitzustellen. Dies ermöglicht einerseits die Trennung von Programmlogik und Datenbankoperationen und schafft andererseits eine Unabhängigkeit gegenüber der Datenquelle, deren Austausch einfach möglich wäre. Die Datenbanktransaktionen selbst werden mit Hilfe des *Hibernate Frameworks* abgewickelt. Über sogenannte *Sessions* werden die Datenbankoperationen realisiert. Die abstrakte Klasse *HibernateDao.java* enthält die Implementierung der CRUD Operationen (Create, Read, Update, Delete). Sowohl für Messungen *MeasurementDaoImpl.java* als auch für Benutzer *UserDAOImpl.java* gibt es Implementierungen die *HibernateDao.java* um eigene Datenbankoperationen erweitern.

Die DAO werden den Ressourcen über *Spring Dependency Injection* zur Verfügung gestellt. Ein Zugriff auf die REST Schnittstelle bedeutet in der Folge auch einen Zugriff auf die Datenbank. Daher befasst sich der folgende Abschnitt mit Authentifizierung und Autorisierung.

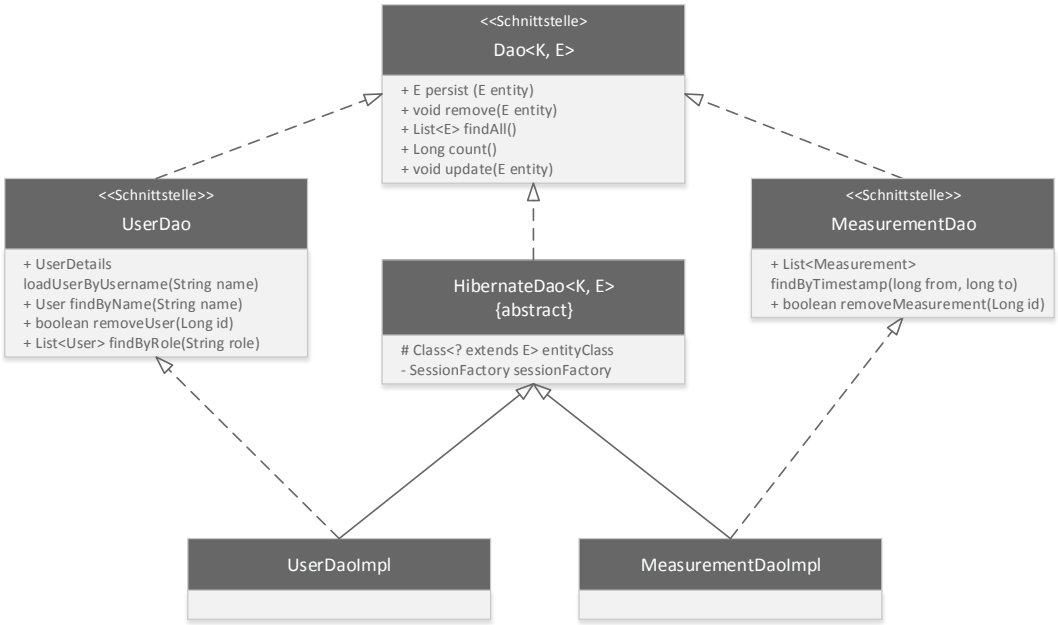

Abbildung 3.9: Klassendiagramm der Data Access Objects

3.3.3 Security-Layer

Durch das Security-Paket des *Spring Frameworks* sind im Prototyp erste Maßnahmen zur Datenintegrität und Sicherheit implementiert. Die Security Konfiguration regelt Zugriffe auf die Schnittstelle. Es werden folgende Szenarien unterschieden:

- *Nicht authentifiziert* - Unbekannte Zugriffe sind lediglich auf die GET Methode und den **/api/v1/measurements** Pfad möglich
- *Authentifiziert als User* - Authentifizierte Zugriffe mit der Rolle „User“ sind berechtigt Anfragen mit der POST Methode an **/api/v1/measurements** zu stellen
- *Authentifiziert als Admin* - Zugriffe durch Administratoren sind auf alle implementierten Operationen möglich

Diese Unterscheidung soll gewährleisten, dass alle Benutzer Messungen einsehen können (durch die Noise Map), jedoch nur authentifizierte Benutzer Messungen übertragen können (durch den Noise-O-Meter). Zu jeder Übertragung seitens des Noise-O-Meters wird automatisch eine verschlüsselte Benutzername-Passwort-Kombination hinzugefügt, die das Gerät zu einem authentifizierten Gerät mit der Rolle „User“ macht. Die Authentifizierung verhindert, dass Übertragungen angenommen werden, die nicht vom Noise-O-Meter stammen und soll dadurch die Datenintegrität schützen.

3.3.4 Technologie

Java Web Anwendungen werden durch den Einsatz sogenannter *Servlets* umgesetzt. Eine Java-Klasse implementiert die Schnittstelle *javax.servlet.Servlet* um HTTP-Methoden verarbeiten zu können. Damit eine Server Komponente auf diese Klasse zugreifen kann, muss der Pfad im sogenannten *Deployment Descriptor*, der *web.xml* Datei spezifiziert werden. So eine Server Komponente kann beispielsweise ein *Servlet Container* wie *Apache Tomcat* oder auch ein vollwertiger J2EE-Server wie *JBoss* sein. Da im vorliegenden Projekt alle *Servlet*-Anfragen über das *Jersey Framework* abgewickelt werden sollen, ist im *Deployment Descriptor* eine *Jersey Servlet*-Klasse definiert. Tabelle 3.11 stellt die wichtigsten Abhängigkeiten des Backends dar.

Java Frameworks und Bibliotheken		
Paket	Version	Zweck
Jersey-Spring3	2.11	Java REST Framework
Jackson JAXRS Provider	2.4.1	JSON ObjectMapping
Javax Servlet API	2.5	Java Servlet Funktionalität
Spring Framework	4.0.6	Java Web Anwendungs Framework
PostgreSQL	9.1-901.jdbc4	PostgreSQL Treiber
Hibernate Entitymanager	4.3.5	ORM Framework
Spring Security	3.2.5	Security Framework
Rest-Assured	2.3.2	Test Funktionalität für die REST Schnittstelle
JUnit	4.8.1	Java Test Framework
Pusher-Rest-Java	0.9.0	Pusher Funktionalität

Tabelle 3.11: Java Web Applikation Abhängigkeiten

3.4 Noise Map

Dieses Kapitel beschreibt die technische Umsetzung der Noise Map in deren Mittelpunkt die kartografische Visualisierung der Lärmmessungen steht. Ihre Aufgaben als Frontend sind die Präsentation der Daten sowie den Benutzer zu Interaktion und Exploration anzuregen.

Das Grundgerüst der Noise Map basiert auf dem von Google entwickelten open-source JavaScript Framework *AngularJS*. Dieses ermöglicht das dynamische Binden von Daten an HTML Elemente und dadurch ein hohes Maß an Interaktivität und Flexibilität. Die Web Map basiert auf der open-source JavaScript Bibliothek *Leaflet* und den Karten der *OpenStreetMap* und der *Basemap*.

Der folgende Abschnitt 3.4.1 beschreibt die wichtigsten Eigenschaften von *AngularJS* sowie deren Einsatz in der Noise Map. Abschnitt 3.4.2 erläutert den gesamten Prozess der Visualisierung vom Designprozess über Clustering bis hin zu den zur Verfügung gestellten

Interaktionen. Der letzte Abschnitt 3.4.3 gibt eine Übersicht über die verwendeten Technologien.

3.4.1 AngularJS Web Applikation

Für die Präsentation dynamischer Inhalte im Web sind Technologien notwendig, die über bloßes HTML hinausgehen. Das *AngularJS* Framework bietet durch *Directives* und *Data-Binding* Features, welche besonders für die Entwicklung von *Single Page Applications* geeignet sind. Darüber hinaus forciert das Framework das MVC Entwurfsmuster und damit eine Trennung von Zuständigkeiten. Abbildung 3.10 zeigt wie Model, View und Controller in einer *AngularJS* Web Anwendung zusammenarbeiten.

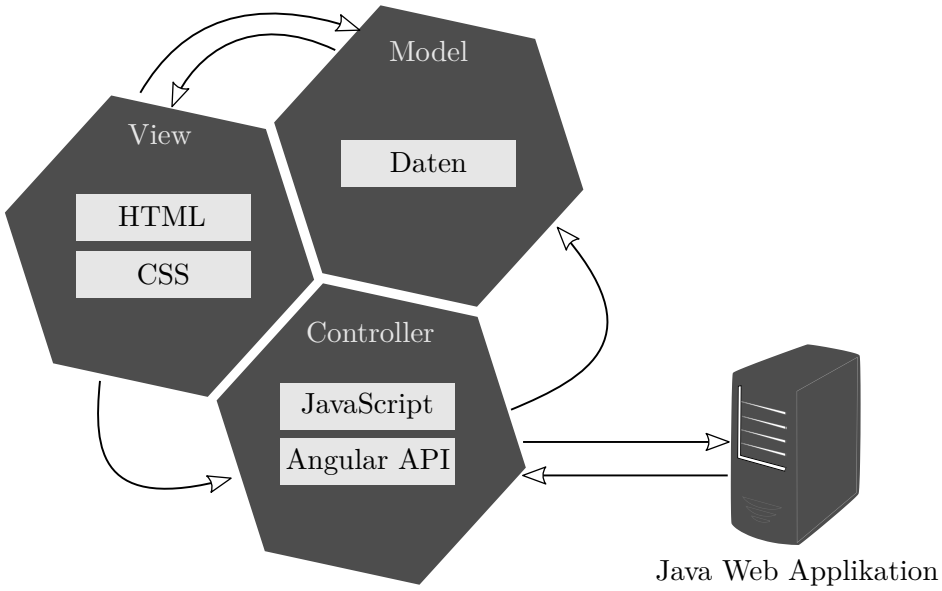

Abbildung 3.10: MVC in AngularJS

Das *Model* beinhaltet sowohl die Daten der Web Anwendung als auch die entsprechende Geschäftslogik. Dies könnte beispielsweise eine Liste von Lärmmessungen in JSON sein.

Der *View* beschreibt jenen Bereich der Web Anwendung, den der Benutzer beim Aufruf zu sehen bekommt, und daher den Ist-Zustand des *Models*. Er ist das Ergebnis des gerenderten *Document Object Model*, welches in *AngularJS* über *Directives* manipuliert und gleichzeitig strukturiert wird. Diese machen es möglich, benutzerdefinierte DOM Elemente zu erstellen sowie bestehende DOM Elemente mit speziellem Verhalten auszustatten. Durch das bereits erwähnte *Two-Way Data-Binding* ist es möglich, dass Änderungen, die der Benutzer im *View* vornimmt, sich unmittelbar auf das *Model* auswirken.

Die *Controller* werden als JavaScript-Klassen realisiert und enthalten die gesamte Programmlogik. Sie stellen das Bindeglied zwischen *Model* und *View* dar und verarbeiten Interaktionen seitens des Benutzers. Darüber hinaus beziehen sie im vorliegenden Prototyp Daten über die REST Schnittstelle.

Aufbau

Die *index.html* Datei bildet das *AngularJS Template* und legt das HTML Grundgerüst mit Sektionen für dynamisch geladene *Views* sowie die entsprechenden *Controller* fest. Eine *Bootstrap* Navigationsleiste erlaubt das „Springen“ zwischen der Map und einem kurzen Text, welcher das Projekt beschreibt. Web Anwendungen mit dieser Form der Präsentation werden auch als *One-Page-Scroller* bezeichnet, da das Auswählen eines Menüpunkts zu einem CSS-animierten Scroll-Effekt führt. Die übrigen *Views* und *Controller* werden als Komponenten der Visualisierung im folgenden Abschnitt beschrieben.

AngularJS erscheint nicht nur wegen seiner Möglichkeit, einfach *Single Page Applications* erstellen zu können sinnvoll, sondern auch auf Grund der guten Integration der *Leaflet* Map Bibliothek. Die *Angular-Leaflet-Directive* ermöglicht es die Vorteile von *Data-Binding* und *Directives* auch für Web Mapping zu nutzen.

3.4.2 Kartografische Visualisierung

Im *Map-View* sind HTML Defintionen für die Map selbst sowie für das wichtige Präsentationselement *Sidebar* definiert. Letzteres ist ein verstecktes *Div-Element*, welches ausgelöst durch Eingaben des Benutzers sichtbar wird und wichtige Informationen oder zusätzliche Bedienelemente enthält. Auch die *Sidebar* wird dynamisch mit Inhalten befüllt. Abbildung 3.11 zeigt die Ansicht die der Benutzer beim initialen Aufruf der Noise Map vorfindet.

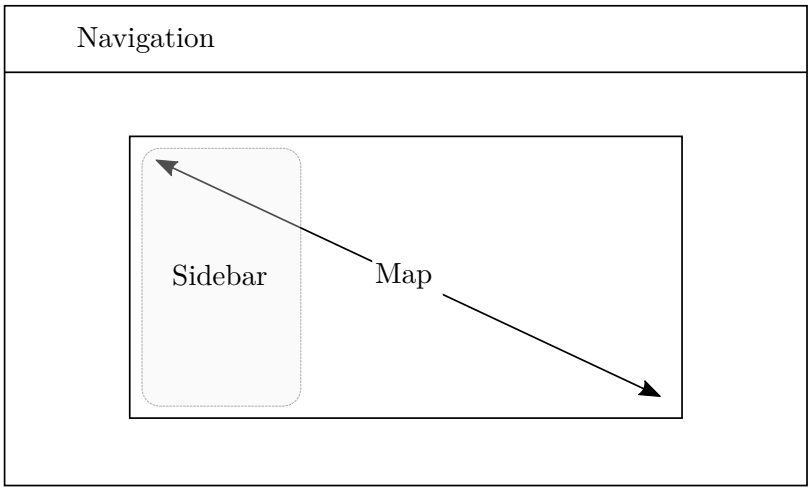

Abbildung 3.11: Noise Map Wireframe

Die gesamte, die Map betreffende Programmlogik, ist im *Map-Controller* zusammengefasst. Dieser enthält unter anderem die Definition der initialen Map-Parameter und des Kartenmaterials. Die weitere Beschreibung des Visualisierungsprozesses orientiert sich an den sieben Schritten der Datenvisualisierung [Fry08], dargestellt in Abbildung 3.12.

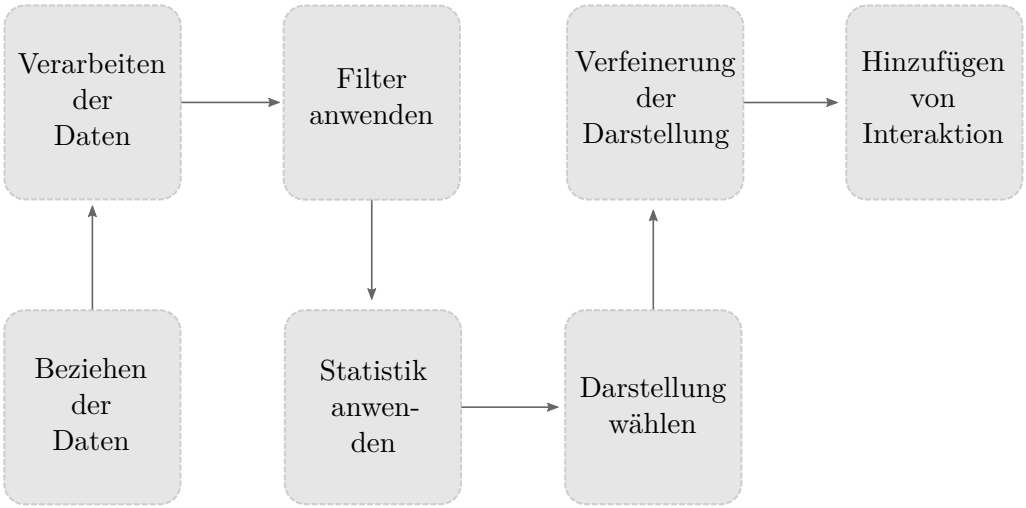

Abbildung 3.12: Die sieben Schritte der Datenvisualisierung

Beziehen der Daten

Die *Angular-Resource* Bibliothek bietet die Möglichkeit, mit externen Datenquellen über REST Schnittstellen in Interaktion zu treten. Eine *Factory* Funktion retourniert ein *Resource*-Objekt mit der entsprechenden Antwort des Servers. Dadurch können im *Map-Controller* die Lärmmessungen mit *Measurement.query()* einfach abgerufen werden.

Verarbeiten der Daten

Dieser Schritt wird im vorliegenden Projekt bereits im Backend durchgeführt. Durch das entsprechende *ObjectMapping* können die übertragenen Daten durch einen *Query-Parameter* entweder in JSON oder in GeoJSON angefordert werden. *Leaflet* kann mittels GeoJSON große Datenmengen komfortabel kartografisch abbilden.

Eine Lärmmessung in GeoJSON hat einen Typ „Feature“, ein Geometrie-Objekt, welches den Typ sowie ein Array mit Koordinaten enthält und ein Properties-Objekt, das Aufschluss über den Messwert, den Zeitstempel sowie die ID des Urhebers gibt.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [
      16.3360479,
      48.2279154
    ]
  },
  "properties": {
    "value": 61.54524,
    "timestamp": 1415796548000,
    "owner": 16
  }
}
```

Filter anwenden

Die Daten könnten theoretisch sowohl serverseitig durch das Übertragen von zusätzlichen Parametern an die REST Schnittstelle als auch clientseitig durch die Verwendung der implementierten Filterfunktion selektiert werden. Zum Zeitpunkt dieser Arbeit sind jedoch nur die clientseitigen Filterfunktionen im Prototyp implementiert. Diese machen es möglich, das in Schritt eins angeforderte *Resource*-Objekt nach dem Zeitstempel sowie der ID des Benutzers zu filtern. Die entsprechenden Funktionen finden sich im *Map-Controller* wieder. Die Filter werden automatisch ausgelöst, wenn die Noise Map zum ersten Mal geladen wird und wenn sich das Datenarray während der Betrachtung ändert (Live-Aktualisierung). Der Benutzer kann die Filter über die *Sidebar* verändern. In dieser befinden sich zwei *AngularUI Datepicker* zur Auswahl des Beginn- und des Enddatums sowie ein *Dropdown-Menü* zur Auswahl der Benutzer ID.

Statistik anwenden

Methoden der deskriptiven Statistik können vom Benutzer mit Hilfe der Statistik-Werkzeuge angewandt werden. Eine Modifikation der *Leaflet.draw* Bibliothek macht es möglich Areale auf der Karte zu markieren und auszuwerten. Die Funktionsweise der Modifikation wird im Folgenden beschrieben:

Anhand der selektierten Punkte werden Diagramme erstellt und dem Benutzer in der *Sidebar* präsentiert. Die Lärmentwicklung im markierten Areal kann am Liniendiagramm (Abbildung 3.13) abgelesen werden, welches die letzten zwölf Messungen zeigt. Das Kuchendiagramm (Abbildung 3.14) zeigt eine Auswertung aller selektierten Messungen und unterteilt diese in die Kategorien ruhig, mittel und laut.

Die Diagramme werden mit Hilfe der *Chart.js* Bibliothek und einer entsprechenden *AngularJS Directive* realisiert. Der *Statistik-View* wird bei Beendigung der Benutzereingabe

Algorithm 3.1: Modifikation der *leaflet.draw* Funktionalität

Data: Array of all points
Result: Array of selected points

```
1  listen to draw:created events;
2  if draw:created then
3      if type == circle then
4          for each data point;
5          if | center - point | < radius then
6              add to selected points;
7          end
8      else
           /* type is polygon or rectangle                    */
9          for each data point;
10         if point in polygon then
               /* calculated with leaflet-pip                 */
11             add to selected points;
12         end
13     end
14 end
```

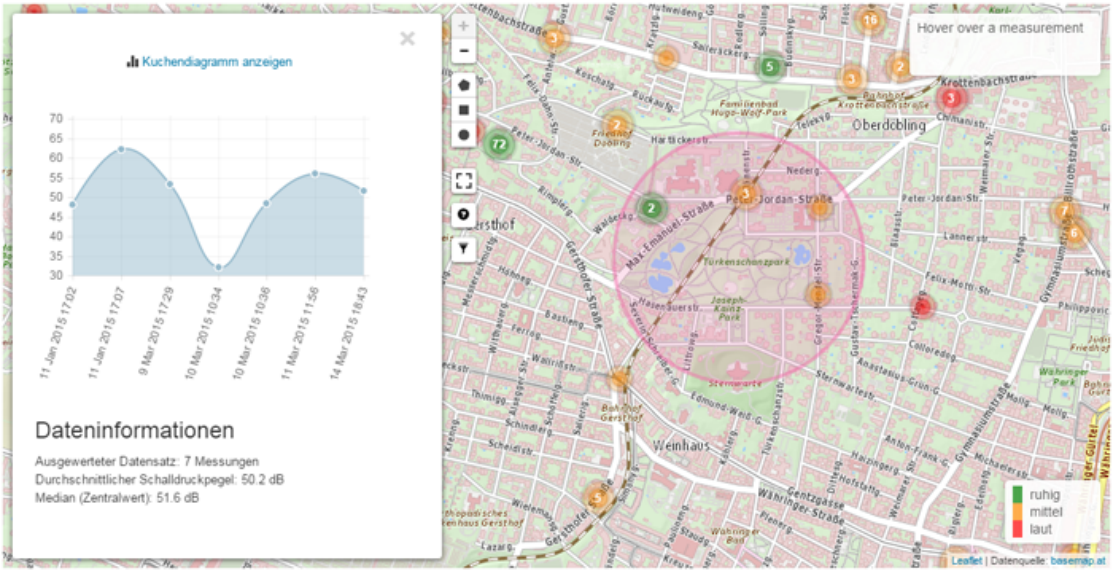

Abbildung 3.13: Statistische Auswertung an Hand des Liniendiagramms im Prototyp

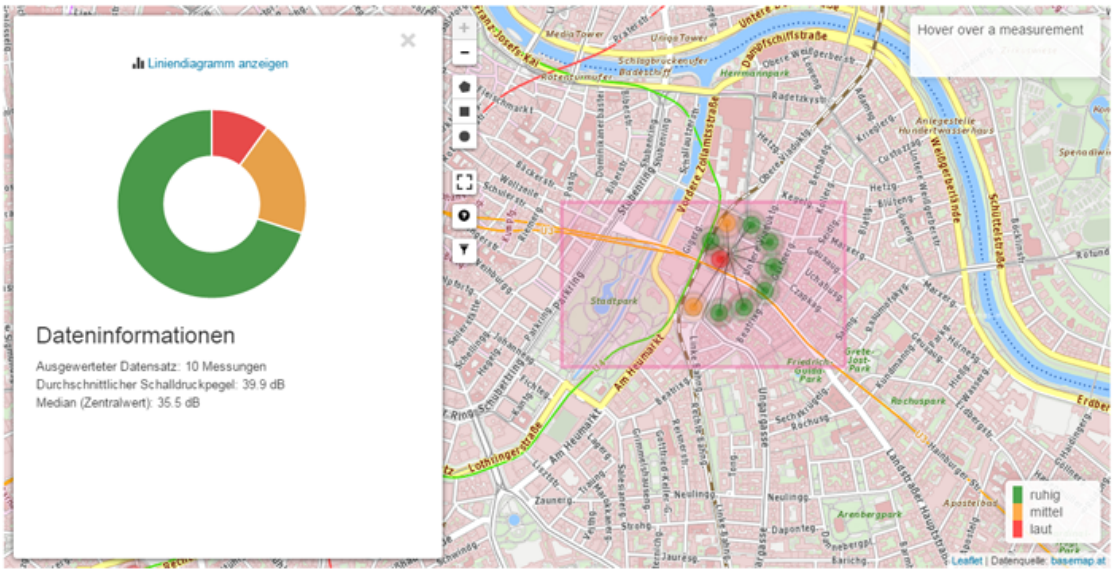

Abbildung 3.14: Statistische Auswertung an Hand des Kuchendiagramms im Prototyp

(dem Markieren des Areals) dynamisch in die *Sidebar* geladen. Kuchen- und Liniendiagramme sind nur zwei der vielen zur Verfügung stehenden Varianten die *Chart.js* bietet. Zusätzlich werden dem Benutzer die Anzahl der markierten Messungen, der Mittelwert und der Medien ausgegeben. Diese Funktionen sind ebenfalls im *Map-Controller* umgesetzt.

Darstellung wählen

Das Kartenmaterial für die Noise Map bilden einerseits die *OpenStreetMap* und speziell für den österreichischen Raum die *Basemap*. *Leaflet* ermöglicht den Einsatz mehrerer Karten die dynamisch gewechselt werden können. Die Evaluation (Kapitel 4) hat ergeben, dass die Grundkarte der *Basemap* die beliebteste Darstellungsvariante ist, weshalb diese im Prototyp der Standard ist.

Die Darstellung der Messungen wird durch animierte Scalable Vectorgraphics (SVG) realisiert. Kreisförmige grafische Repräsentationen stellen einzelne Lärmquellen auf der Noise Map dar. Um mehr Aufmerksamkeit zu generieren und gleichzeitig optisch an die Ausbreitung von Schall zu erinnern wird mittels Synchronized Multimedia Integration Language (SMIL) eine Animation erzeugt. Die insgesamt drei Kreise vergrößern langsam ihren Radius wodurch der Eindruck einer wellenförmigen Ausbreitung entsteht. Die Repräsentationen sind in grün, orange und rot gehalten um mit den Ampelfarben drei Schalldruckpegelintervalle zu beschreiben.

1. Grün: 0 dB - 45 dB
2. Orange: 46 dB - 65 dB

3. Rot: 66 dB und höher

Der folgende Code beschreibt einen der drei animierten Kreise, die gemeinsam die grafische Repräsentation bilden.

```
<circle id="wave" stroke="black" fill-opacity="0.4"
cx="25" cy="25" r="3" fill="blue" stroke-width="5"
stroke-opacity="0.1">

<animate id="anim1" attributeName="r" attributeType="XML"
from="0" to="7" begin="0s" dur="2s" fill="freeze"
repeatCount="indefinite" />

</circle>
```

Der leicht transparente Kreis wird über einen Zeitraum von zwei Sekunden vom Radius 0 zu einem Radius Größe 7 transformiert. Dieser Vorgang wiederholt sich für die anderen drei Kreise mit sukzessive größeren Radien wobei im letzten Kreis zusätzlich eine „fade out“ Animation eingesetzt wird um flüssigere Übergänge zu erreichen. Durch den Einsatz animierter Repräsentation soll laut Kraak [Kra03] visuelles Denken gefördert werden. Abbildung 3.15 zeigt die drei verschiedenfarbigen Icons, welche die Lärmquellen auf der Map darstellen.

Abbildung 3.15: Icons der Noise Map

Darstellung verfeinern

Die Grundansicht der Noise Map zeigt kreisförmige Lärmquellen auf dem Kartenmaterial der *Basemap*. Um auch große Datenmengen am selben Ort oder sich überlappende Lärmquellen sinnvoll abbilden zu können, müssen mehrere Lärmquellen zusammengefasst dargestellt werden. Datengruppen, sogenannte Cluster, können auf der Basis verschiedener Kriterien zusammengefasst werden. Auch in diesem Punkt gibt es sowohl serverseitige als auch clientseitige Methoden die Daten zu gruppieren. Der realisierte Prototyp verwendet ausschließlich clientseitige Methoden und setzt dabei auf die *Leaflet-Markercluster* Bibliothek. In einem agglomerativen hierarchischen Clusteringverfahren bildet zunächst jeder

Punkt einen eigenen Cluster und wird dann schrittweise über die Berechnung der Distanz zu größeren Clustern zusammengefasst. Zu diesem Zweck setzt die *Leaflet-Markercluster* Bibliothek auf ein sogenanntes *Distance Grid*, welches den gesamten Kartenbereich in Quadrate vordefinierter Größe unterteilt. Die Implementierung des Prototyps sieht einen maximalen Clusterradius von 40 Pixel vor. Um bei diesem Prozess keine Informationen zu verlieren, wird die *iconCreateFunction()* modifiziert, sodass der Cluster den Durchschnittswert der enthaltenen Lärmmessungen annimmt sowie den Zeitpunkt der aktuellsten Messung. Zu diesem Zweck werden über alle Cluster iteriert, die erwähnten Werte berechnet und abhängig vom Resultat das entsprechende Icon aus Schritt vier ausgegeben. Dieses Icon enthält zusätzlich in seinem Zentrum eine Zahl, welche die Anzahl der enthaltenen Messpunkte angibt und als Unterscheidungsmerkmal zu einzelnen Messungen dient.

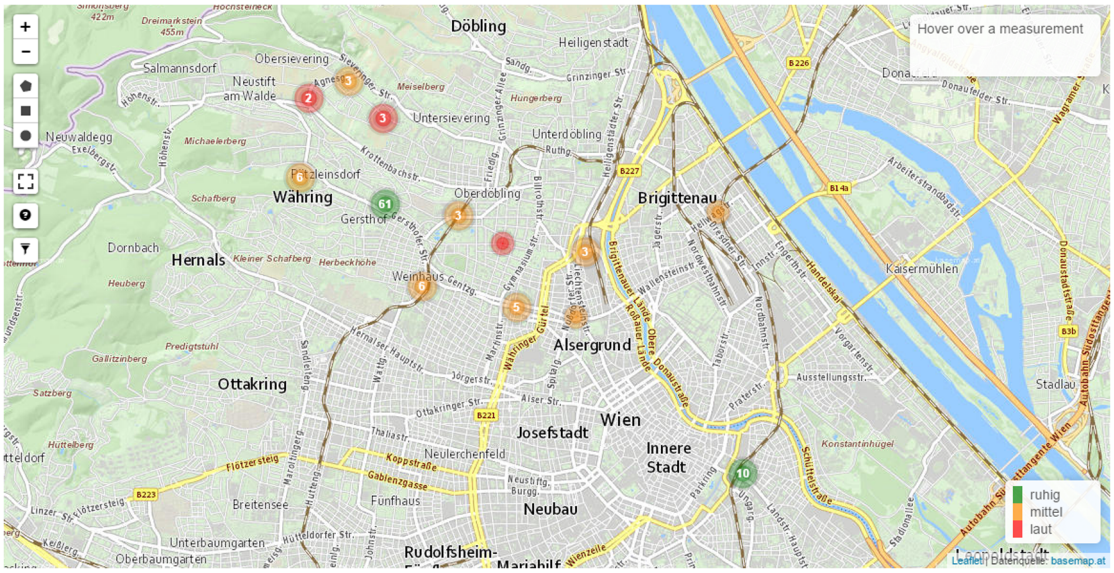

Abbildung 3.16: Noise Map mit Clustern

Der gesamte Vorgang hat eine ungefähre Laufzeit von O(n) wobei n die Anzahl der zu clusternden Punkte ist. Die *Spiderfy-Funktion* erlaubt es, per Mausklick die einzelnen Messungen eines Clusters zu betrachten (in Abbildung 3.14 zu sehen).

Die Darstellung der Lärmmessungen wird zusätzlich durch eine zeitliche Komponente verfeinert. Die *Angular Pusher API* ermöglicht Live-Aktualisierungen, wenn sich die zu Grunde liegenden Daten verändern. Dass bedeutet, dass der Benutzer, ohne die Web Anwendung neu laden zu müssen, die aktuellsten Daten sieht und beobachten kann, wie neue Lärmpunkte hinzukommen. *Anguler Pusher* funktioniert nach einem Publisher/Subscriber Modell, bei dem Nachrichten über Kanäle ausgetauscht werden können.

Wenn ein mobiler Benutzer einen neuen Datensatz hochlädt, wird über die *Pusher API* eine Nachricht an die Noise Map gesendet. In der Folge werden die aktuellsten Daten

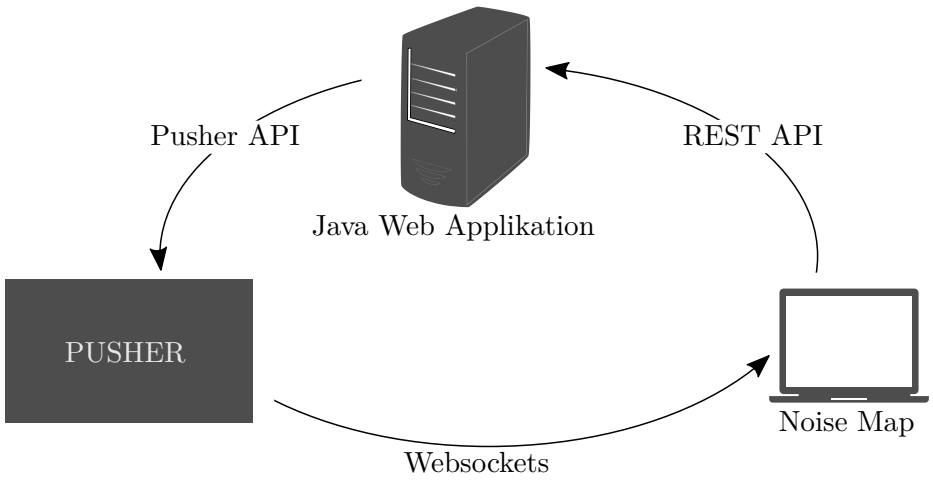

Abbildung 3.17: Pusher in der Noise Map

vom Server geladen und aktualisiert auf der Karte dargestellt. Für den Betrachter wird ein neuer Lärmpunkt sichtbar während er gerade die Daten erkundet.

Interaktion hinzufügen

In Kapitel 2.3 wurden vier Kategorien mit Interaktionsmöglichkeiten nach Crampton [Cra02] vorgestellt. Die Noise Map bietet insgesamt sieben der in Abbildung 2.8 dargestellten Möglichkeiten.

1. Zoom
2. Navigation
3. Hervorheben
4. Verschiedene Ansichten
5. Statistik (Daten-Mining)
6. Markieren und Auswerten
7. Filtern

Benutzer können den **Zoomlevel** der Map sowohl über die Buttons (Plus/Minus) am linken oberen Rand als auch über das Mausrad regulieren. Der initiale Wert *13* ist über den *Map-Controller* festgelegt und stellt etwa einen Maßstab von *1:70.000* dar. Insgesamt werden *15* Stufen zwischen den Maßstäben *1:500 Mio* und *1:15.000* angeboten.

Navigation in der Noise Map ist durch Klicken und Ziehen mit der Maus möglich. Benutzer können entweder durch *Drag and Drop* oder durch Doppelklick den Kartenausschnitt nach ihren Wünschen verändern und dadurch die Daten erforschen.

Das **Hervorheben** von einzelnen Lärmpunkten oder Clustern ist mittels *Mouseover* möglich. Eine Infobox in der rechten oberen Ecke der Map zeigt Zeit und Datum der letzten Messung, den durchschnittlichen Schalldruckpegel sowie die Anzahl der Messungen am gewählten Ort an. Dies ist auch für einzelne Lärmpunkte eines Clusters möglich, sofern dieser zuvor durch einen Klick „aufgefächert“ wurde.

Die Noise Map bietet drei verschiedene Karten, die vom Benutzer via Mausklick gewechselt werden können. Diese sind die *OpenStreetMap* für eine weltweite Darstellung sowie die *Basemap* in der Standardansicht und in einer reduzierten grauen Variante für den österreichischen Raum. Darüber hinaus ist es den Benutzern möglich, die Karte und alle ihre Werkzeuge im Vollbildmodus zu verwenden. Eine flächendeckende **Ansicht** bietet zusätzlich Übersicht und Fokus.

Statistik kann durch eines der drei zur Verfügung stehenden Werkzeuge abgerufen werden. Zu diesem Zweck wählt der Benutzer einen der drei Buttons (Polygon, Rechteck, Kreis) und markiert ein gewünschtes Areal. Die enthaltenen Lärmpunkte werden ausgewertet und für den Benutzer sowohl als Diagramm als auch in Form deskriptiver Statistik dargestellt (Abbildung 3.13 und 3.14). Über einen Hilfe-Button können animierte Grafiken hervorgerufen werden, die den Einsatz dieser Werkzeuge vorführen. Darüber hinaus wird die Verwendung durch Tooltips unterstützt.

Über den **Filter**-Button können die angezeigten Lärmpunkte angepasst werden. Zwei *Datepicker* ermöglichen es dem Benutzer, ein Startdatum und ein Enddatum zu wählen. Darüber hinaus können über ein *Dropdown Menü* nur die Messungen einer bestimmten Benutzer ID angezeigt werden. Die beschrieben Buttons, über welche der Benutzer mit der Noise Map in Interaktion treten kann, sind in Abbildung 3.16 ersichtlich. Abbildung 3.18 zeigt die gesamte prototypisch realisierte Web Anwendung zum Zeitpunkt dieser Arbeit.

3.4.3 Technologie

Die Noise Map ist mit *Yeoman* erstellt, einem open-source Entwicklungs-Toolkit, welches sowohl Generatoren, Build Management als auch Dependency Management inkludiert. Die Generatoren bilden eine Projektstruktur für die Web Anwendung und installieren die gewünschten *AngularJS* Pakete. Alle weiteren Pakete werden über den Paketmanager *Bower* installiert und verwaltet. Der *Build-Prozess* wird mit Hilfe des Toolkits *Grunt* automatisiert. Wichtige Features sind die Zusammenführung aller Bibliotheken zu einer gemeinsamen Datei sowie die Komprimierung von Grafiken und Programmcodes. Diese Prozesse führen zu weniger externen Zugriffen und dadurch zu einer besseren Performanz der Web Anwendung. Tabelle 3.12 gibt eine Auflistung aller in der Noise Map zum Einsatz kommenden JavaScript Frameworks und Bibliotheken.

JavaScript Frameworks und Bibliotheken		
Paket	Version	Zweck
AngularJS	1.3.2	Framework für Web Anwendungen
Leaflet	0.7.3	Mapping Bibliothek
Angular-Leaflet-Directive	0.7.10	Directive für Leaflet Maps
Bootstrap	3.2.0	Responsive Web Framework
Angular-Resource	1.3.0	REST-Support für Angular Web Anwendungen
Angular-Route	1.3.0	Directive für Single-Page-Apps
AngularUI	0.11.2	Bootstrap-Komponenten als Angular Directives
Angular-Pusher	0.0.13	Echtzeitdaten über Websockets
Chart.js	1.0.1	JavaScript Diagramm-Bibliothek
Leaflet-Pip	0.1.0	Berechnet ob ein Punkt in einem Polygon liegt
Leaflet-Markercluster	0.4.0	Cluster-Funktionalität für Leaflet
Startbootstrap-Scrolling-Nav	1.0.0	One-Page-Scroller Template
Leaflet.Fullscreen	0.0.3	Leaflet Vollbild Erweiterung
Leaflet.Draw	0.2.3	Erweiterung zum Zeichnen von geometrischen Formen
Leaflet-Sidebar	0.1.6	Dynamische Sidebar für Leaflet
L.EasyButton	*	Erweiterung für Custom Buttons

Tabelle 3.12: Noise Map Abhängigkeiten

In Kapitel 3 wurde die technische Umsetzung der drei Säulen der Noise Map, beginnend beim Entwurf und der Architektur bis zur Noise Map selbst, beschrieben. Der Aufbau des Kapitels orientiert sich an den Aufgaben der Komponenten. Diese sind die Datenerfassung durch den Noise-O-Meter, die Datenverarbeitung durch die Java Web Applikation sowie die Präsentation durch die Noise Map. Es beschreibt detailliert die prototypische Realisierung und beweist zugleich die Machbarkeit dieses Ansatzes zur Lärmvisualisierung. Im folgenden Kapitel 4 wird die Evaluation des Prototyps beschrieben, deren Erkenntnisse bereits in der Refinementphase in den Prototyps eingeflossen sind.

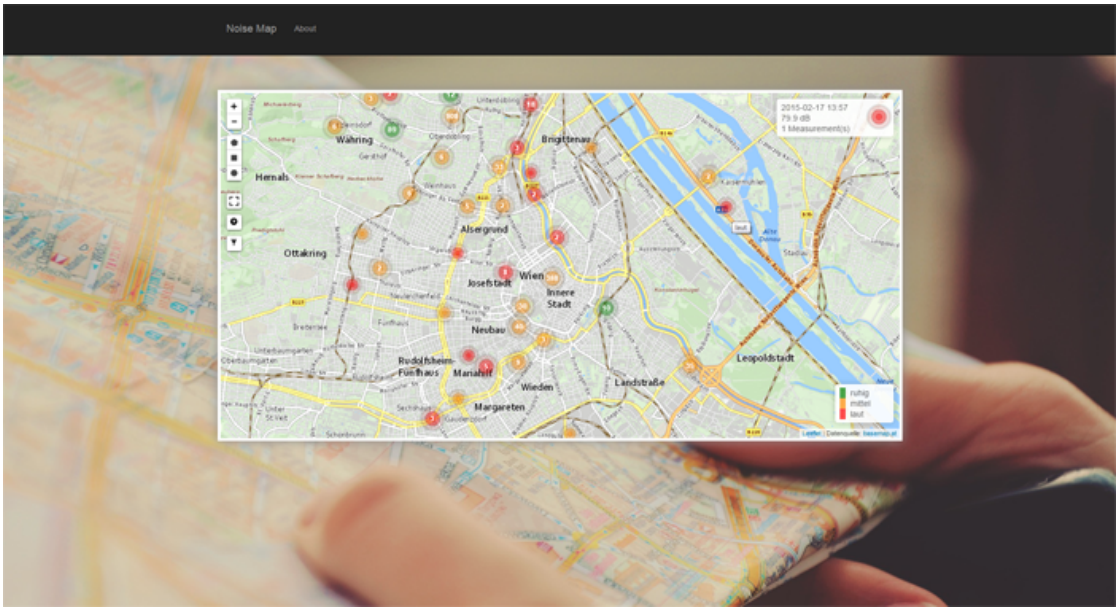

Abbildung 3.18: Noise Map Prototyp

KAPITEL 4

Evaluation

Die hohe Verfügbarkeit von Datenmaterial und Werkzeugen führt dazu, dass Visualisierungen immer populärer werden. Aus dieser Entwicklung entsteht die Notwendigkeit Visualisierungen, insbesondere wenn sie in der Forschung zum Einsatz kommen, auf ihre Eignung hin zu beurteilen. Die Herausforderungen liegen einerseits darin, den Fokus der Evaluation richtig zu setzen sowie andererseits die richtigen Fragen zu stellen [KSFN08]. Im Mittelpunkt dieser Evaluation steht die Noise Map in ihrer Funktion, Informationen in Daten sichtbar zu machen. Da in der Informationsvisualisierung sowohl Aspekte aus der Mensch-Computer Interaktion als auch wahrnehmungspsychologische Gesichtspunkte eine wichtige Rolle spielen, sind die Fragen und Aufgaben in der vorliegenden Evaluation auf die Bereiche Usability und Wahrnehmung gerichtet.

- *Usability* - Mit Aufgaben in diesem Bereich soll die Bedienbarkeit der Visualisierung überprüft werden. Diese können sich zum Beispiel auf Funktionen wie Zooming, Filtering oder den Zugriff auf Detailinformationen beziehen.
- *Wahrnehmung* - Aufgaben in diesem Bereich gehen der Frage nach, ob geeignete Repräsentationen für die darzustellenden Daten gewählt wurden. Durch Fragen zur Wahrnehmung und dem Verständnis wird die richtige Kodierung sowie die Lesbarkeit der Visualisierung überprüft.

4.1 Zielsetzung

Das primäre Ziel der Evaluation ist es, herauszufinden ob die Noise Map Einsichten in die Daten geben kann. Dies ist dann der Fall, wenn es seitens der Probanden zu einer adäquaten Erfassung der Visualisierung kommt. Zusätzlich soll unter Anwendung qualitativer Methoden Potenzial für Verbesserungen aufgedeckt und gegebenenfalls in den Prototyp eingearbeitet werden. Die Resultate der Evaluation dienen somit der Optimierung des Prototyps und dessen Bedienbarkeit.

4.2 Ausgangslage

Der Ablauf der Evaluation erfolgt in zwei Phasen. In der ersten Phase wird eine Stichprobe von zehn Teilnehmern ausgewählt und mit der Noise-O-Meter Applikation ausgerüstet. Diese werden gebeten, über einen Zeitraum von mindestens sieben Tagen möglichst viele Messungen durchzuführen. Im Rahmen des Erstkontakts unterzeichnen sie eine Einverständniserklärung, welche die Vorgehensweise sowie die Rahmenbedingungen der Evaluation festlegt. Diese umfasst die Beschreibung des Projekts, den Einsatz der Android Applikation, die Rechte des Probanden, Erläuterungen betreffend der zweiten Evaluationsphase sowie Kontaktinformationen. Die zweite Phase erfolgt nach Abschluss der mindestens siebentägigen Messperiode. In einem experimentellen Setting müssen die Probanden Aufgaben am Prototyp der Noise Map durchführen. Durch Beobachtung und Dokumentation des Verlaufs sowie der Anwendung des Think-Aloud-Protokolls sollen Eindrücke und Gedanken zur Noise Map erfasst werden. Die Zusammenstellung der Aufgaben erfolgte mit der Zielsetzung, Erkenntnisse über die Qualität der Werkzeuge, der grafischen Repräsentationen sowie der gesamten Darstellung zu erhalten. Darüber hinaus wurde auf eine ausgeglichene Verteilung über folgende Kategorien aus den Bereichen Wahrnehmung und Usability Wert gelegt:

- Bedienbarkeit - Beurteilung der Interaktionsmöglichkeiten
- Kodierung - Beurteilung der eingesetzten Kodierung
- Lesbarkeit - Beurteilung der eingesetzten Repräsentationen

Insgesamt werden jedem Probanden dreizehn Aufgaben zur Noise Map vorgegeben, auf die im Folgenden einzeln eingegangen wird.

1. Was denken Sie wird hier dargestellt?
   Ziel(e): Kodierung, Lesbarkeit
   Aufgabe(n): Assoziieren
   Den Probanden wird zunächst die Grundansicht der Noise Map vorgegeben, welche die grafischen Repräsentationen der Lärmmessungen der letzten fünf Monate umfasst. Sie werden gebeten ihre Assoziationen zur Darstellung zu beschreiben.

2. Bitte beschreiben Sie, was Ihrem Eindruck nach die farbigen Punkte bedeuten.
   Ziel(e): Kodierung, Lesbarkeit
   Aufgabe(n): Assoziieren, Kategorisieren
   Die Probanden beschreiben ihre Assoziationen zu den grünen, orangen und roten Punkten.

3. Ist Ihnen ein Größenunterschied zwischen den Punkten aufgefallen?
   Ziel(e): Kodierung, Lesbarkeit
   Aufgabe(n): Vergleichen, Unterscheiden
   Falls ja, werden die Probanden gebeten, ihre Assoziationen betreffend des Größenunterschieds zu beschreiben. Falls nein, wird darauf hingewiesen und so wie bei einer positiven Antwort fortgefahren.

4. Finden Sie die Animation der Punkte ansprechend?
   Ziel(e): Kodierung, Lesbarkeit
   Aufgabe(n): Einstufen
   Die Probanden werden gebeten, ihren Eindruck auf einer Skala von eins bis sechs zu beschreiben, wobei eins für „gar nicht“ und sechs für „sehr“ steht.

5. Versuchen Sie sich bitte einen Überblick über Wien zu verschaffen und beschreiben Sie dabei die einzelnen Schritte.
   Ziel(e): Bedienbarkeit
   Die Probanden sollen diese Aufgabe mit der Maus auf der Karte lösen. Dabei wird dokumentiert, welche Werkzeuge zum Einsatz kommen und wie die Probanden vorgehen.

6. Bitte wählen Sie einen großen Punkt aus und beschreiben Sie was Sie wahrnehmen. Bitte klicken Sie auf den Punkt - was sehen Sie jetzt zusätzlich? Ist diese Repräsentation verständlich?
   Ziel(e): Bedienbarkeit, Kodierung, Lesbarkeit
   Die Probanden sollen mit dem Cursor einen Cluster auswählen und diesen durch einen Klick erweitern. Mittels Mouseover sind in der linken unteren Ecke Informationen zum Cluster als auch zu den einzelnen Repräsentationen zu lesen.

7. Bitte zeigen Sie nur Messungen aus Dezember 2014 an.
   Ziel(e): Bedienbarkeit
   Die Probanden werden gebeten, die Filterfunktionen zu benutzen. Das Vorgehen der Probanden wird dokumentiert.

8. Bitte erkunden Sie die Symbole am Rand der Karte und beschreiben Sie Ihren Eindruck. Sind sie selbsterklärend? Gibt es Unklarheiten?
   Ziel(e): Bedienbarkeit, Kodierung, Lesbarkeit
   Aufgabe(n): Assoziieren, Unterscheiden
   Die Probanden sollen die zur Verfügung stehenden Werkzeuge ausprobieren und beschreiben. Dabei wird dokumentiert, wie intuitiv die Bedienung erfolgt und wo mögliche Probleme auftreten.

9. Vergleichen Sie bitte die durchschnittliche Lärmbelastung von Alsergrund mit der von Innere Stadt.
   Ziel(e): Bedienbarkeit
   Aufgabe(n): Vergleichen
   In dieser Aufgabe sollen die Probanden die vorher kennengelernten Statistikwerkzeuge einsetzen. Dabei wird die Vorgehensweise sowie das Resultat dokumentiert.

10. Bitte beschreiben Sie kurz den Eindruck über die zwei verschiedenen Darstellungen der Karte.
    Ziel(e): Bedienbarkeit, Lesbarkeit
    Aufgabe(n): Vergleichen

Sowohl die Basemap als auch die OpenStreetMap werden den Probanden vorgeführt und deren Assoziationen dokumentiert.

11. Entsprechen die Lärmmessungen Ihrem subjektiven Eindruck?
    Ziel(e): Bedienbarkeit, Kodierung
    Aufgabe(n): Vergleichen
    Die Probanden werden mit ihren eigenen Messungen konfrontiert und beschreiben ihren subjektiven Eindruck der einzelnen Situationen.

12. Haben Sie Anregungen, Verbesserungsvorschläge oder Ideen für die Noise Map?
    Die Anregungen und Vorschläge der Probanden werden dokumentiert.

13. Bitte schildern Sie Ihre Erfahrungen mit der Noise-O-Meter Applikation (Akku, Bedienbarkeit, Probleme).
    Die Erfahrungen der Probanden werden dokumentiert.

4.3 Durchführung

Die Interviews werden vom Evaluationsleiter arrangiert und in einem kontrollierten Setting durchgeführt. Dieser ruft dafür den online zur Verfügung stehenden Prototyp der Noise Map auf und achtet darauf, dass eine Maus mit Mausrad zur Verfügung steht. Für die Evaluation wird der Filter so eingestellt, dass die Messungen der letzten fünf Monate angezeigt werden. Während der Aufgaben beobachtet und dokumentiert der Evaluationsleiter den Verlauf sowie die Aussagen der Probanden. Tabelle 4.1 bietet eine Übersicht über die Probanden und deren Geräte.

Teilnehmer an der Noise Map Evaluation			
Id	Geschlecht	Alter	Device
p0	weiblich	33	Samsung Galaxy S3
p1	männlich	18	Galaxy Nexus i9250
p2	männlich	29	Sony Xperia T3
p3	weiblich	33	Huawei Ascend Y300
p4	männlich	28	HTC One Mini 2
p5	weiblich	57	Huawei Ascend Y300
p6	männlich	32	Samsung Galaxy S3 Mini
p7	männlich	32	Galaxy Nexus i9250
p8	weiblich	30	Samsung Galaxy S4
p9	männlich	59	HTC Desire 816

Tabelle 4.1: Geschlecht und Alter der Probanden sowie die verwendeten Android Geräte

Die vorliegende Stichprobe spiegelt großteils eine Altersklasse wider, die üblicherweise mit Technik gut vertraut ist. Da für den Testbetrieb der Noise-O-Meter Applikation

die privaten Geräte der Probanden herangezogen werden mussten, ist dies von hoher Relevanz.

4.4 Resultate

Im Folgenden werden die Ergebnisse dargestellt. Dies sind einerseits die Daten (insgesamt 1863 Messungen), welche aus der ersten Phase der Evaluation resultieren (siehe Tabelle 4.2) und andererseits die transkribierten Evaluationsbögen, welche mit Hilfe thematischer Analyse und farblicher Kodierung auf wiederkehrende Muster untersucht wurden. Die Erkenntnisse aus den insgesamt dreizehn Aufgaben des Bogens werden zusammengefasst dargestellt, um daraus Schlussfolgerungen hinsichtlich der Qualität des Prototypen (Lesbarkeit, Bedienbarkeit und Kodierung) ziehen zu können.

Gesammelte Daten der Probanden				
Id	Anzahl Messungen	Min. SPL	Max. SPL	Avg. SPL
p0	43	23,44 dB	74,29 dB	43,25 dB
p1	15	57,78 dB	78,69 dB	66,80 dB
p2	12	6,90 dB	72,77 dB	37,32 dB
p3	125	20,88 dB	79,28 dB	51,95 dB
p4	172	23,06 dB	79,35 dB	60,39 dB
p5	86	20,88 dB	80,31 dB	44,42 dB
p6	12	34,96 dB	77,49 dB	58,94 dB
p7	1342	33,92 dB	80,31 dB	50,67 dB
p8	32	40,37 dB	80,29 dB	67,62 dB
p9	24	25,12 dB	73,71 dB	47,87 dB

Tabelle 4.2: Anzahl der durchgeführten Messungen mit niedrigstem, höchstem und durchschnittlichem Schalldruckpegel pro Proband

Auf Grund der bereits vorhandenen Vertrautheit der Probanden mit dem Projekt, diente die erste Aufgabe hauptsächlich als Einstieg in die Evaluation. Das Beschreiben der Darstellung fungiert als Ausgangspunkt für die weiteren Fragen und konnte von allen hinreichend ausgeführt werden.

Die Assoziationen zu den Lärmpunkten ähneln sich in der Wahrnehmung der Intensität (grün = ruhig/gut, orange = weniger ruhig/mittel, rot = laut/schlecht), die verwendeten Begriffe unterscheiden sich jedoch. So assoziierten etwa 40% der Probanden alle drei Repräsentationen ausschließlich mit Verkehrssituationen, während 60% von Grünflächen, Kaffeehäusern und anderen Alltagssituationen sprachen (Abbildung 4.1). Mehrere Aussagen, lassen darauf schließen, dass innerhalb der drei Pegelintervalle trotz der Dezibel-Angabe im Info-Panel kaum noch differenziert wird. Ein roter Punkt wird als sehr laut empfunden auch wenn der gemessene Wert an der Grenze zur orangen Kategorie liegt.

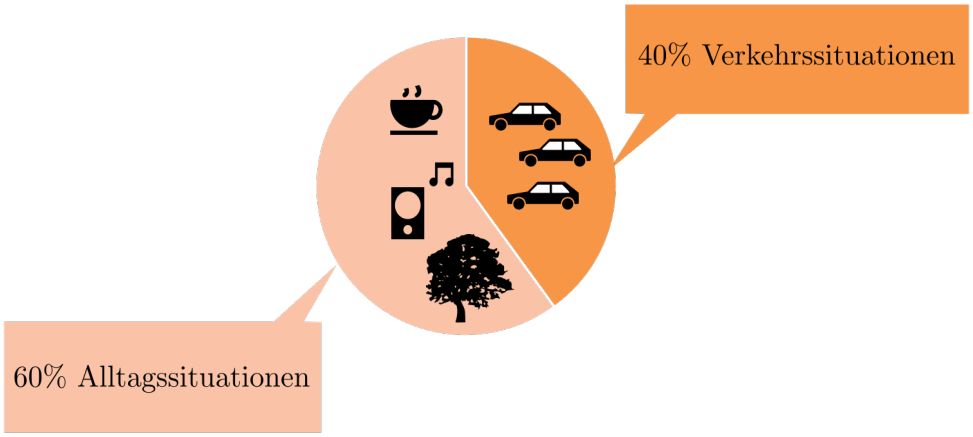

Abbildung 4.1: Darstellung der unterschiedlichen Assoziationen der Probanden

Auffallend ist, dass der Größenunterschied zwischen den einzelnen Lärmmessungen und den Clustern (siehe Abbildung 4.2) von nur zwei Probanden auf Anhieb wahrgenommen wurde. Auch nach einem entsprechenden Hinweis interpretieren nur vier von zehn Teilnehmern diesen als eine Häufung von Messungen. 40% der Teilnehmer vermuten fälschlich, dass durch ein größeres Symbol ein besonders lauter Ort dargestellt wird.

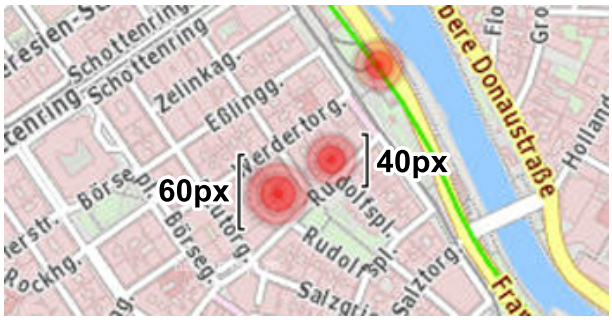

Abbildung 4.2: Zeigt Cluster (li.) und eine Einzelmessung (re.)

Die Animation der grafischen Repräsentationen wird als durchwegs positiv empfunden (Abbildung 4.3). Auf einer Skala von 1 bis 6 (1 = gar nicht ansprechend, 6 = sehr ansprechend) wurde diese als ansprechend bis sehr ansprechend bewertet. Positiv wurde der Hinweischarakter sowie die erhöhte Aufmerksamkeit durch die pulsierenden Punkte hervorgehoben. Die Animation wird als lebhaft und alarmierend beschrieben.

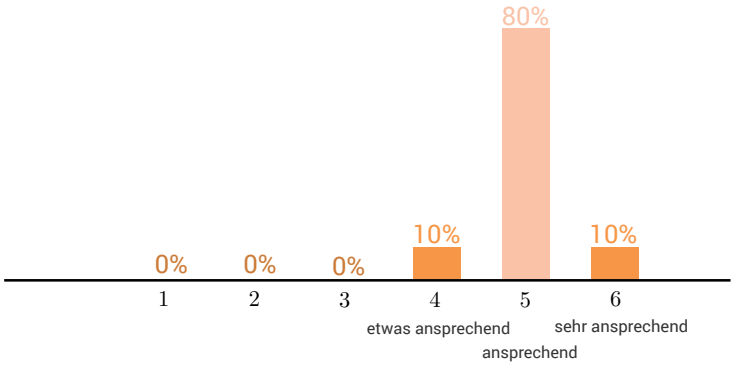

Abbildung 4.3: Bewertung der grafischen Animation durch die Probanden

Aufgabe fünf wurde von allen Teilnehmern erfolgreich absolviert. Ein Großteil der Probanden war mit den angebotenen Navigationsmöglichkeiten innerhalb einer Karte bereits vertraut. Geringere Erfahrung spiegelt sich in vermehrtem Einsatz der angebotenen Zoom-Buttons wider, wogegen erfahrene Benutzer den Zoomlevel hauptsächlich mit dem Mausrad regulieren und die Karte via drag & drop bewegen. Die Häufigkeiten mit denen die unterschiedlichen Navigationsmöglichkeiten eingesetzt wurden, werden in Abbildung 4.4 dargestellt.

Die Noise Map bietet sowohl grafische Repräsentationen als auch Werkzeuge, um zusätzliche Informationen über das Datenmaterial zu erlangen (siehe Abbildung 4.5). Die Anzeige für Detailinformationen am linken unteren Rand der Karte gibt Aufschluss über den Zeitpunkt der letzten Messung, den gemessenen Schalldruckpegel sowie die Anzahl der Messungen. Ein Klick auf einen Cluster führt zu einer Aufschlüsselung der einzelnen darin befindlichen Messungen und damit zu zusätzlichen Informationen. Diese Ansicht wurde von 80% der Probanden als verständlich beschrieben.

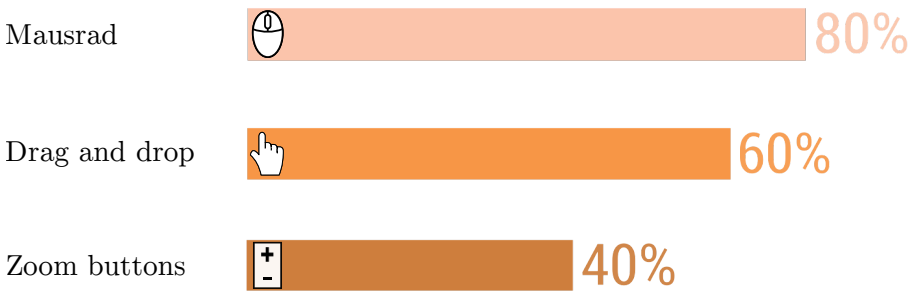

Abbildung 4.4: Eingesetzte Navigationsmöglichkeiten nach Häufigkeit geordnet

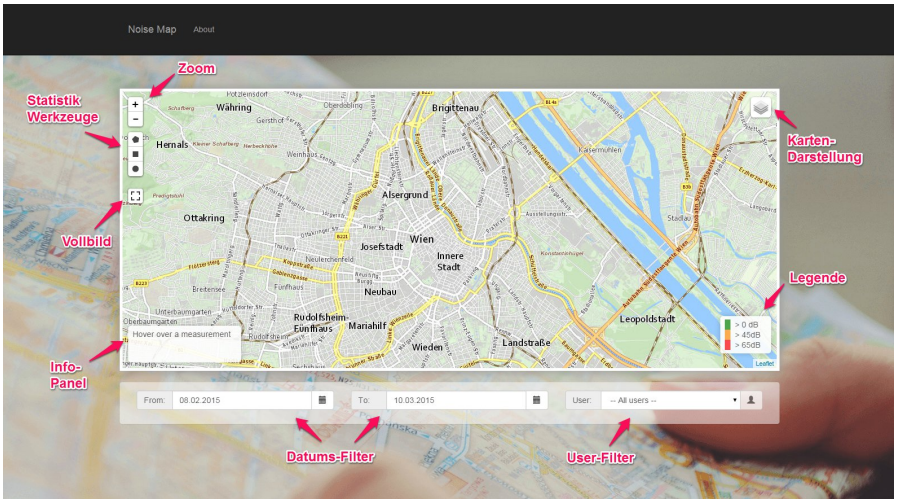

Abbildung 4.5: Prototyp der Noise Map, welcher für die Evaluation eingesetzt wurde

Neun der Probanden waren auf Anhieb in der Lage Messungen mit Hilfe der Filterwerkzeuge zeitlich zu selektieren. Zoom und Vollbild wurden von allen entweder nach dem Betrachten des Tooltips oder nach dem ersten Ausprobieren als verständlich empfunden. Die Anzeige für Detailinformationen (Info-Panel) wurde von 50% der Probanden spät oder gar nicht wahrgenommen, was einen schwerwiegenden Informationsverlust bedeutet. Zwei Personen äußerten zudem den Wunsch nach einer auffälligeren grafischen Darstellung. Die Legende wurde von allen als selbsterklärend beschrieben. Auf Nachfrage wurde jedoch deutlich, dass die Verwendung des mathematischen Symbols $>$ größer nicht für alle sofort klar war. Die Statistikwerkzeuge (Polygon, Kreis und Rechteck) stellten für alle eine gewisse Herausforderung dar. Besonders das Polygon, welches über das Setzen von Punkten ein „Abstecken“ bestimmter Areale ermöglicht, wurde trotz Tooltip als zu kompliziert empfunden. Mehrere Personen wünschten sich eine Anleitung zu diesem Bereich.

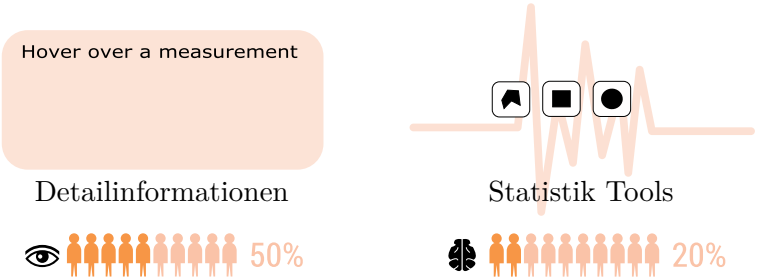

Abbildung 4.6: Mängel an Lesbarkeit und Bedienbarkeit bei der Anzeige für Detailinformationen sowie den Statistik Werkzeugen

Trotz der Schwierigkeiten benutzten 50% von ihnen das Polygon für zumindest einen der beiden zu vermessenden Bezirke. Diese Aufgabe wurde von allen zufriedenstellend bewältigt. Das Diagramm, welches die Lärmquellen im markierten Areal darstellt, war häufig nicht in der Lage, die Menge an gewählten Messungen darzustellen. Dies resultiert daraus, dass der Prototyp sowohl die Expertenansicht als auch die aggregierte Version vereint.

Das Symbol für die verschiedenen Darstellungsvarianten der Karte wurde von allen Probanden nach dem ersten Ausprobieren verstanden und eingesetzt. Von den zwei angebotenen Kartenversionen (Basemap, OpenStreetMap) sprachen sich sechs Probanden für die Basemap aus, da diese übersichtlicher und daher für die Darstellung der Lärmpunkte besser geeignet sei. Drei sahen in beiden Varianten Vor- und Nachteile und einer bevorzugte die OpenStreetMap.

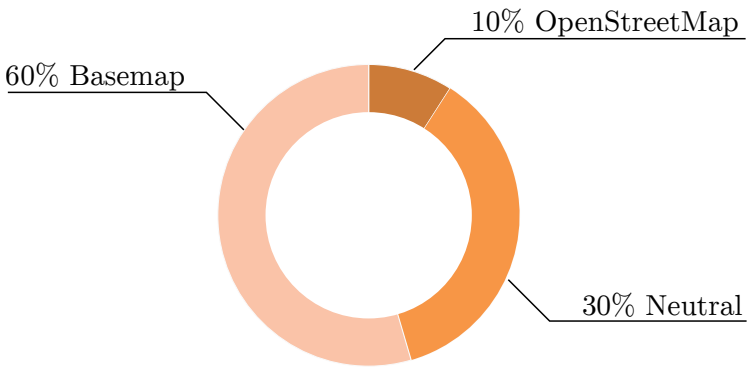

Abbildung 4.7: Zeigt die bevorzugte Darstellungsvariante der Probanden

Auf die subjektive Empfindung zu den eigenen Lärmmessungen befragt, berichteten 50% der Teilnehmer eine Übereinstimmung mit den eigenen Eindrücken. Die anderen 50% empfanden ihre visualisierten Messungen zumindest teilweise als zu laut. Eine Person äußerte die Vermutung, dass einerseits Wind und andererseits die Reibung seiner Jacke die Messung verfälscht haben könnten. Bei einzelnen Geräten der Probanden, welche ihre Messungen als zu laut empfunden haben, konnte eine Abweichung der Messung im Vergleich zum Referenzmodell (Huawei Ascend Y300) beobachtet werden.

Befragt nach ihren Erfahrungen im Umgang mit der Noise-O-Meter App berichteten vier Teilnehmer von erhöhtem Energieverbrauch bei intensiver Nutzung. Fünf nahmen keine verstärkten Akkuverbrauch wahr. Positiv wurde die einfache Handhabung hervorgehoben, sowie die Möglichkeit, ohne vorherige Anmeldung beitragen zu können.

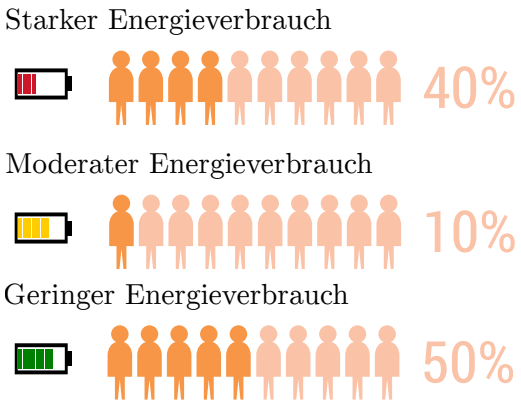

Abbildung 4.8: Beurteilung des Energieverbrauchs durch die Tester

4.5 Schlussfolgerungen

Aus den Erkenntnissen der Evaluation ergeben sich folgende Maßnahmen bezüglich des Prototyps:

- Modifikation der grafischen Cluster-Repräsentationen
- Umgestaltung und Neupositionierung des Info-Panels
- Verzicht auf mathematische Symbole in der Legende
- Anleitung zu den Statistik-Werkzeugen
- Effizienterer Einsatz der Standortbestimmung, um Energie zu sparen
- Gleichzeitige Darstellung des markierten Areals und des Diagramms

Cluster wurden mit einer Zahl im Zentrum der grafischen Repräsentation versehen, welche die Anzahl der zusammengefassten Lärmpunkte angibt. Diese dient als zusätzliches Unterscheidungsmerkmal zu einzelnen Lärmmessungen.

Das Info-Panel wurde von der linken unteren Ecke in die rechte obere Ecke versetzt und zeigt nun zusätzlich zu den Detailinformationen eine grafische Repräsentation jener Lärmquelle über der sich der Mauszeiger des Benutzers gerade befindet. Durch die bessere Position und die farbliche Veränderung soll mehr Aufmerksamkeit für dieses wichtige Informationselement erzeugt werden.

Die drei in der Legende beschriebenen Lärmintervalle wurden mit den mehrheitlich verwendeten Begriffen der Probanden benannt um möglichen Missverständnissen auf Grund des mathematischen „Größer“-als-Zeichen vorzubeugen. Diese lauten „ruhig“, „mittel“ und „laut“.

Der Prozess der Standortbestimmung wurden so verändert, dass zwischen den Lärmmessungen das Aktualisierungsintervall verringert wird um unnötigen Positionsabfragen vorzubeugen. Dadurch soll ein geringerer Energieverbrauch erreicht werden.

Um statistische Informationen gleichzeitig mit dem markierten Areal anzeigen zu können wurde die bestehende Anzeige in einem *Modal-Window* durch eine dynamische *Sidebar* ersetzt, welche erscheint sobald der Benutzer einen Markierungsvorgang abgeschlossen hat. Dies ermöglicht dem Benutzer eine Verbindung zwischen den statistischen Auswertungen und Diagrammen und den selektierten Lärmquellen.

Im Folgenden Kapitel werden die wichtigsten Punkte dieser Arbeit noch einmal zusammengefasst und weiterführende Aspekte beschrieben.

KAPITEL 5

Fazit und Ausblick

Die vorliegende Arbeit nutzt die Möglichkeiten der Informatik, um sich der Gesundheitsgefährdung durch Lärmverschmutzung und damit einem konkreten Problem des modernen Lebens zu widmen. Die Noise Map hat durch das Sichtbarmachen von Lärm das Potenzial sich positiv auf das Leben von Menschen auszuwirken, und gleichzeitig ein Bewusstsein für die entstandene Problematik zu schaffen. Ziel dieser Arbeit war es, Lärmmessungen so zu visualisieren, dass ein Bild über die Lärmbelastung einer Stadt entsteht.

Entwicklung, Umsetzung und Evaluation des Prototyps liefern eine positive Antwort auf die Frage der Machbarkeit von Lärmvisualisierung durch *Participatory Sensing*. Auf der Basis einer ausführlichen Literaturrecherche sowie der Analyse inhaltlich verwandter Projekte konnte ein Softwareentwurf erstellt werden, welcher drei Komponenten umfasst:

1. Eine mobile Sensorkomponente zur Datenerfassung
2. Eine zentrale Serverkomponente mit standardisierter Schnittstelle als Datenhub
3. Eine kartografische Visualisierung, die Noise Map als Präsentationskomponente mit Interaktionsmöglichkeiten, welche zur Exploration einladen

Mit der aus der Kooperation von SOPHISYSTEMS und der TU Wien entstandenen *Android* Applikation Noise-O-Meter, stand bereits eine Sensorkomponente zur mobilen Messung des Schalldruckpegels auf dem Smartphone zur Verfügung. Sie wurde um die Funktionen zur Standortbestimmung und Datenübertragung erweitert, um die raumzeitliche Darstellung der Messwerte auf der Noise Map zu ermöglichen. Die kollaborative mobile Datenerfassung bildet die Grundlage für das vorliegende Projekt und basiert auf Anreizen durch Information. Im Ausgleich für die Rechenleistung der Smartphones bekommt jeder Benutzer die Möglichkeit, die aggregierten Daten online zu erforschen.

Die Java Web Applikation stellt mit Hilfe des *Jersey REST Frameworks* eine standardisierte Schnittstelle nach den REST Prinzipien zur Verfügung. Übertragene Daten

werden in einer *PostgreSQL* Datenbank gespeichert und für die Präsentation vorverarbeitet. Darüber hinaus reglementiert das *Spring Security Framework* den Zugriff auf die Daten und setzt dadurch wichtige Schritte in Richtung Datenschutz und -integrität. Dies ist wichtig, da die Bereitwilligkeit des Benutzers, freiwillig Daten und vor allem sensitive Metadaten herzugeben, von dessen Vertrauen abhängt. Diese Metadaten (Standort und Zeitstempel) können viel über Gewohnheiten der mobilen Benutzer aussagen und sind daher unbedingt schützenswert.

Eine *Leaflet* Web Map eingebettet in einer *AngularJS* Web Anwendung stellt Lärmquellen mithilfe animierter grafischer Repräsentationen dar. Die farblich reduzierte Karte und clientseitiges Clustering helfen einen schnellen Überblick über die Lärmbelastung einer Stadt zu gewinnen. Einen besonderen Anreiz zur Erforschung der Daten liefert die Live-Visualisierung neuer Messungen. Dadurch entsteht eine Art Verbindung zwischen den Betrachtern der Noise Map und den mobilen Benutzern, die mit ihren Smartphones Messungen durchführen. Statistische Werkzeuge ermöglichen durch das Markieren eines Areals auf der Karte eine weiterführende Analyse der Lärmpunkte. Sowohl deskriptive Statistik als auch verschiedene Diagramme, die beispielsweise die Entwicklung der markierten Messungen über die Zeit zeigen, stehen dem Benutzer zur Verfügung. Die angezeigten Daten können darüber hinaus über Filter modifiziert werden, um ausschließlich die Lärmbelastung einer bestimmten Zeitperiode anzuzeigen. Dies, sowie die Veränderung des Kartenausschnitts ermöglicht einen Vergleich von Belastungsszenarien.

Im Rahmen der Evaluation haben zehn Probanden zwischen 18 und 59 Jahren, über einen Zeitraum von einer Woche, 1863 Messungen mit dem Noise-O-Meter durchgeführt und später auf der Noise Map nachverfolgt. Dadurch konnten Schwächen des Prototyps, insbesondere bezüglich der Darstellung und Bedienbarkeit, an Hand der Evaluationsergebnisse in der Refinementphase behoben werden. Die Darstellung durchwegs positiv bewerteter animierter Repräsentationen zeichnet ein Bild, welches rasch besonders laute sowie besonders leise Plätze einer Stadt erkennen lässt.

Ein wichtiger Faktor für den Erfolg von *Participatory Sensing* Projekten ist eine große Anzahl von Messungen. Die Voraussetzung dafür ist, potenzielle Barrieren bei der Nutzung möglichst gering zu halten. Obwohl die Eigenschaften des Prototyps diesbezüglich in vielen Belangen positiv zu bewerten sind, könnten die weitere Optimierung des Energieverbrauchs sowie der Effizienz der mobilen Applikation sich günstig auswirken. Besonders hinsichtlich sensibler Metadaten sind zusätzliche Vorkehrungen betreffend des Datenschutzes notwendig.

Entscheidend für die Aussagekraft der Visualisierung ist die Datenqualität, welche auch im vorliegenden Prototyp einen wichtigen Stellenwert einnimmt. Trotzdem sei an dieser Stelle erwähnt, dass mittels weiterer Maßnahmen, wie beispielsweise einer Bewertung der Qualität der eingehenden Daten oder einer zusätzlichen Aggregation mit externen Daten, die Glaubwürdigkeit weiter erhöht werden könnte.

Im Vergleich zu inhaltlich verwandten Projekten ist die Noise Map vor allem durch die Live-Visualisierung hervorzuheben. Der Prototyp bietet den Benutzern viele Funktionen und könnte sich in Zukunft abhängig vom Stakeholder-Interesse in verschiedene Richtungen entwickeln. Business- und Experteninteresse könnte zusätzliche Diagramme

und Analysewerkzeuge hervorbringen, während öffentliches Interesse weitere Darstellungsvarianten nach sich ziehen könnte.

Zukünftige Entwicklungen könnten eine zusätzliche Integration der Noise Map in den Noise-O-Meter hervorbringen, sodass auch mobile Benutzer die kartografische Visualisierung einsehen können. Die Datenverwaltung über die Noise Map und damit die Möglichkeit, als Teilnehmer die Kontrolle über seine Daten auszuüben, stellt ebenfalls einen wichtigen nächsten Entwicklungsschritt dar.

Sowohl die untersuchte Literatur als auch die Forschung im Rahmen dieser Diplomarbeit offenbaren die Bedeutung und das Ausmaß von Lärmverschmutzung. Die Umsetzung des vorliegenden Projekts trägt dazu bei, das Bewusstsein für Lärm im Alltag zu schärfen und dadurch die individuelle Lärmbelastung zu verringern.

Literaturverzeichnis

[ARR11] César Asensio, Manuel Recuero, and Mariano Ruiz. Noise mapping. *Applied Acoustics*, 72(8):477–478, July 2011.

[BC05] Michael A. Black and William E. Cartwright. Web cartography and web-enabled Geographic Information Systems (GIS): new possibilities, new challenges. *Proceedings of the 22nd International Geographic Conference*, pages 1–10, 2005.

[BDD+08] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub, and Christopher Schmidt. The geojson format specification. `http://www.geojson.org/geojson-spec.html`, 2008. [Online; abgerufen April-2015].

[BEH06] JA Burke, D Estrin, and M Hansen. Participatory sensing. *Center of Embedded Network Sensing*, 2006.

[Ben10] J Bennett. OpenStreetMap, 2010.

[BfLuF13] Umwelt und Wasserwirtschaft Bundesministeriums für Land und Forstwirtschaft. Lärmkarte - Straßenlärmkartierung 2007. `http://www.laerminfo.at/karten/strassenverkehr/strasse07/24h.html`, 2013. [Online; abgerufen April-2015].

[Bur09] Bill Burke. RESTful Java with JAX-RS, 1st Edition, 2009.

[Cra02] Jeremy W. Crampton. Interactivity Types in Geographic Visualization. *Cartography and Geographic Information Science*, 29(2):85–98, January 2002.

[CRKH11] Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928–1946, November 2011.

[dKS03] Henk de Kluijver and Jantien Stoter. Noise mapping and GIS: optimising quality and efficiency of noise effect studies. *Computers, Environment and Urban Systems*, 27(1):85–102, January 2003.

[Fry08] Ben Fry. *Visualizing data; [exploring and explaining data with the processing environment]*. O'Reilly, Sebastopol, Calif. [u.a.], 1. ed. edition, 2008.

[Goo15] Google. Google location API. `http://developer.android.com/google/play-services/location.html`, 2015. [Online; abgerufen April-2015].

[HKH10] Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. Preserving privacy in participatory sensing systems. *Computer Communications*, 33(11):1266–1280, July 2010.

[Kan09] Eiman Kanjo. NoiseSPY: A Real-Time Mobile Phone Platform for Urban Noise Monitoring and Mapping. *Mobile Networks and Applications*, 15(4):562–574, November 2009.

[Kra03] Menno-Jan Kraak. Geovisualization illustrated. *ISPRS Journal of Photogrammetry and Remote Sensing*, 57(5-6):390–399, April 2003.

[Kra04] Menno Jan Kraak. The role of the map in a Web-GIS environment. *Journal of Geographical Systems*, 6(2):83–93, 2004.

[KSFN08] Andreas Kerren, John T. Stasko, Jean-Daniel Fekete, and Chris North. *Information Visualization Human-Centered Issues and Perspectives*. Springer, 2008.

[LS92] Luqi and R. Steigerwald. Rapid software prototyping. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume ii, pages 470–479 vol.2, Jan 1992.

[Maz09] Riccardo Mazza. *Introduction to information visualization*. Springer, London, 2009.

[MEGK11] Piotr Mioduszewski, Jerzy a. Ejsmont, Jan Grabowski, and Daniel Karpinski. Noise map validation by continuous noise monitoring. *Applied Acoustics*, 72(8):582–589, July 2011.

[MHL09] Jürgen H. Maue, Heinz Hoffmann, and Arndt von Lüpke. *0 Dezibel + 0 Dezibel = 3 Dezibel; Einführung in die Grundbegriffe und die quantitative Erfassung des Lärms; Null Dezibel + 0 Dezibel = 3 Dezibel*. Erich Schmidt, Berlin, 9., neu bearb. u. erw. aufl. edition, 2009.

[Neu08] A Neumann. Web Mapping and Web Cartography. *Encyclopedia of GIS*, pages 1243–1246, 2008.

[Nie09] Maria E Niessen. NoiseTube: Measuring and mapping noise pollution with mobile phones. *Information Technologies in Environmental Engineering (ITEE 2009)*, 2009.

[RCK10] RK Rana, CT Chou, and SS Kanhere. Ear-phone: an end-to-end participatory urban noise mapping system. *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 105–116, 2010.

[RR07] Leonard Richardson and Sam Ruby. RESTful web services, 2007.

[SBS11] Immanuel Schweizer, R Bärtl, and Axel Schulz. NoiseMap-real-time participatory noise maps. *Proc. 2nd Intl Workshop on Sensing Applications on Mobile Phones*, 2011.

[TLC09] Kang-Ting Tsai, Min-Der Lin, and Yen-Hua Chen. Noise mapping in urban environments: A Taiwan study. *Applied Acoustics*, 70(7):964–972, July 2009.

[YKSJ07] Ji Soo Yi, Youn Ah Kang, John Stasko, and Julie Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–31, 2007.

[Zhu01] Xuan Zhu. Developing web-based mapping applications through distributed object technology. *Cartography and Geographic Information Science, 2001, Vol.28(4), p.249-258*, 2001.

Abkürzungen

API Application Programming Interface. 41

BEV Bundesamts für Eich- und Vermessungswesen. 25

BMLFUW Bundesministeriums für Land und Forstwirtschaft, Umwelt und Wasserwirtschaft. xvii, 24, 25

CSS Cascading Stylesheet. 49

DAO Data Access Object. 38, 46

DI Dependency Injection. 38, 43, 46

DOM Document Object Model. 19, 49

GPS Global Positioning System. 3, 12, 21, 25, 27

HTML Hypertext Markup Language. 18, 48–50

HTTP Hypertext Transfer Protocol. 14, 25, 32, 40–42

ID Identifikationsnummer. 11, 25, 32, 38, 42, 51, 52

IMEI International Mobile Equipment Identity. 11, 25, 38

JPEG Joint Photographic Experts Group. 21

JSON JavaScript Object Notation. 14, 21, 28, 40, 42, 43, 51

KML Keyhole Markup Language. 26

MVC Model View Controller. 37, 48

OSM OpenStreetMap. 19–21, 25

PHP Hypertext Preprocessor. 25, 27

PNG Portable Network Graphics. 21

REST Representational State Transfer. xv, 4, 13, 14, 29, 41, 45, 46, 49, 51, 71

SVG Scalable Vectorgraphics. 54

URI Uniform Resource Identifier. 14, 41

URL Uniform Resource Locator. 21

WHO World Health Organization. 2

WWW World Wide Web. 14, 18, 20

XML Extensible Markup Language. 14, 26