STADT MOERS

Praxissemesterbericht

Projekt: Open Data und Schule

Marc Macziejek, Holger Lieske

18.08.2014

Fakultät: Kommunikation und Umwelt

Betreuer: Prof. Dr. Frank Zimmer

Inhaltsverzeichnis

1. Einführung

Die Stadt Moers – in diesem Projekt in der Rolle des Auftraggebers – ist eine der führenden Kommunen der Open Data-Bewegung. Sie gehört zu den ersten Städten die, nicht unter die Geheimhaltung fallenden Daten für die Öffentlichkeit zugänglich machen. Die unter dem Zugang http://www.offenedaten.moers.de/ erhältlichen Datensätze sind derzeit jedoch aufgrund ihres Formats (maschinenlesbare Datensätze) praktisch nur für Experten nutzbar. Aus diesem Grund beschäftigen sich Open Data-Anhänger derzeit damit, die Datensätze in Form von Webseiten oder Applikationen so aufzubereiten, dass sie für jeden verständlich werden. Ein gutes Beispiel für den Nutzen von maschinenlesbaren Daten, ist der Bundeshaushalt von 2006-2011 unter http://bund.offenerhaushalt.de/, der für den Bürger mit Vergleichsmöglichkeiten und grafischer Darstellung verständlich gemacht wurde.

In ähnlicher Form möchte die Stadt Moers nun erproben, inwiefern die Datensätze für den Schulunterricht visualisiert werden können und diese Visualisierung den Lehrplan unterstützen. Dies soll im Rahmen des Projekts „Open Data und Schule“ untersucht werden.

1.1 Projektziel

Ziel des Projektes war es, den Nutzen von Open Data für den schulischen Unterricht zu verdeutlichen und damit folgenden Auszug einer Studie des Fraunhofer Fokus für die Stadt Köln[1] zu untersuchen:

„Der grundsätzlich breite Themenraum, den offene Verwaltungsdaten zu verschiedenen fachlichen Bereichen wie Geographie, Biologie oder Umweltthemen abstecken können, kann die Zielgruppe Bildung in ihrer Arbeit unterstützen, indem diese Daten für die Wissensvermittlung an Schulen genutzt werden können. Akteure des Bildungssektors haben durch offene Verwaltungsdaten die Möglichkeit, aus einer vertrauenswürdigen Quelle aktuelle Daten, wie etwa die Verteilung von Bodenschätzen oder Bevölkerungsdaten, interaktiv in den Unterricht einzubinden und so über den Tellerrand des zumeist eher statischen Unterrichtsmaterials zu blicken.“[2]

Claus Arndt[3] – Ansprechpartner der Stadt Moers – stand bereits im Vorfeld dieses Projektes jahrelang in Verbindung mit Thomas Nolte vom Gymnasium Adolfinum in Moers. Gemeinsam entwickelten sie die Idee, in Zusammenarbeit mit Studierenden der Hochschule Rhein-Waal eine Projektgruppe zu bilden und den Nutzen von Open Data für den schulischen Unterricht anhand der veröffentlichten Datensätze der Stadt Moers nachzuweisen. So wurde es zur Aufgabe der Studierenden Marc Macziejek und Holger Lieske, gemeinsam mit Lehrpersonal und Schülern, die aktuellen Lehrpläne zu untersuchen und

[1] Informationsseite: http://www.offenedaten-koeln.de/blog/open-data-studie (Stand:15.8.14).
[2] Quelle: http://www.offenedaten-koeln.de/sites/default/files/2013-01-09-open_data_koeln_120918_mit_anhang.pdf Seite 8, Abschnitt 2.2. Zielgruppenanalyse. (Stand:15.8.14).
[3] Fachdienst 3.5 – „Zentrale Dienste“ der Stadt Moers. Fachdienstleiter und Referent des Bürgermeisters.

Themen zu finden, die durch Open Data unterstützt werden können. Darüber hinaus sollten die Datensätze für den Einsatz im Schulunterricht aufbereitet und den Lehrkräften für die Verwendung bereitgestellt werden, um ihren Nutzen nachweislich zu erproben.

1.2 Projektbeteiligte

Name	Rolle
Claus Arndt	Ansprechpartner der Stadt Moers
Thomas Nolte	Lehrer am Adolfinum Moers
Dr. Axel Stender	Statistiker/Rohdatenbereitstellung
Marc Macziejek	Konzept/Technische Umsetzung
Holger Lieske	Konzept/Technische Umsetzung
Thorsten Magerstedt	Technische Umsetzung
Udo Krüger	Konzept

Tabelle 1: Projektbeteiligte

Den Kern der Projektgruppe bildeten der Projektleiter Claus Arndt, Thomas Nolte als Ansprechpartner des Adolfinum sowie die beiden Studierenden Marc Macziejek und Holger Lieske. Herr Dr. Axel Stender[4] stellte die maschinenlesbaren Datensätze zur Verfügung. Thorsten Magerstedt[5] unterstützte die Studierenden bei der technischen Umsetzung.

1.3 Zeitliche Strukturierung

Der im folgenden Kapitel beschriebene Ablauf des Projekts ist in zwei zeitlich getrennten Abschnitten zu betrachten.

Im Rahmen der Veranstaltung „New Public Management“[6] im Studiengang „E-Government“ an der Hochschule Rhein-Waal wurde der Anstoß zu diesem Projekt gegeben. Die beiden Studierenden Marc Macziejek und Holger Lieske erhielten die Aufgabe, in Zusammenarbeit mit Herrn Arndt, Herrn Nolte und Herrn Krüger Konzepte für mögliche Anwendungsfälle zu erstellen und diese in Form eines Berichts als Prüfungsleistung für das Modul einzureichen. Herr Krüger schrieb parallel zu diesem Projekt an einer Bachelorarbeit, die zunächst die Benutzertauglichkeit und nach einiger Entwicklungszeit die Visualisierung der späteren Softwareanwendungen thematisiert. Er war daher an der Erstellung der ersten Konzepte beteiligt.

Während des Praxissemesters[7] bei der Stadt Moers, führten die Studierenden das Projekt mit der technischen Umsetzung und der Weiterentwicklung der Konzepte fort.

Die beiden Abschnitte werden in den Kapiteln *2 Planung* und *3 Technische Umsetzung* im Detail beschrieben.

---

[4] Fachdienst 3.3 – „Organisation“ der Stadt Moers. Statistiker.
[5] Fachdienst 3.5 – „Zentrale Dienste“ der Stadt Moers. Zuständig für E-Government.
[6] Deutsch: Öffentliche Reformverwaltung. Eine Verwaltungsreform, die auf der Übernahme von privatwirtschaftlichen Managementtechniken in die öffentliche Verwaltung beruht.
[7] 20-wöchiges Pflichtpraktikum im sechsten Semester, von März bis Juli 2014.

2 Planung

2.1 Sichtung der Daten

Als eine der ersten Kommunen Deutschlands startete die Stadt Moers am 15.02.2013 mit einer eigenen Open Data-Plattform. Unter http://www.offenedaten.moers.de/ werden seitdem verschiedene Datensätze, die nicht unter Geheimhaltung fallen, in maschinenlesbaren Formaten zur Verfügung gestellt. Bis Juli 2014 veröffentlichte die Stadt Moers auf Ihrer Plattform insgesamt 113 Datensätze in folgenden Kategorien:

- Allgemein
- Bevölkerung
- Finanzen
- Freizeit und Tourismus
- Geodaten
- Internet
- Jugend und Soziales
- Kultur und Bildung
- Ratsinformationssystem
- Verwaltung
- Wahlen

Vor den ersten Meetings mit allen Projektbeteiligten wurden die bis dahin veröffentlichten Datensätze von den Studierenden gesichtet. Dabei wurde eine Recherche über die Struktur der Datenformate, deren Einsatzgebiete und verarbeitungsfähige Software durchgeführt. Außerdem wurde eine Liste für das Lehrpersonal des Gymnasium Adolfinum erstellt, die alle Datensätze sowie eine kurze Erläuterung umfasste.

2.2 Kick-Off Meeting

Nach der Sichtung der Datensätze wurde ein Kick-Off Meeting[8] durchgeführt. Dabei kamen erstmals alle Projektbeteiligten zusammen, um den Ablauf des Projekts und die Vorgehensweise zu besprechen.

Zunächst wurde die Fachschaft Sozialwissenschaften für die Lehrplanuntersuchung und die Unterrichtsbesuche festgelegt. Herr Thomas Nolte bot einen Politikkurs in der neunten Jahrgangsstufe und zwei Sozialwissenschaftenkurse in der elften und zwölften Jahrgangsstufe für Unterrichtsbesuche an. Außerdem wurde geklärt, zu welchem Zeitpunkt die Unterrichtsbesuche stattfinden konnten, ohne den laufenden Unterricht nach Lehrplan zu stören.

Die Studierenden präsentierten beispielhaft eine der ersten Open Data-Visualisierungen unter http://www.offenerhaushalt.de/. An diesem Beispiel wurden Möglichkeiten für den Schulunterricht erläutert und erste Untersuchungen auf die Integration in die bestehenden Lehrpläne durchgeführt.

[8] Auftaktveranstaltung zu Beginn eines Projektes.

Darüber hinaus wurde die Liste der Datensätze an Herrn Nolte übergeben. Diese sollte auf für die Fachschaft Sozialwissenschaften interessante Themenfelder hin untersucht werden.

2.3 Vorbereitung der Unterrichtsbesuche

Die Unterrichtsbesuche wurden durch die Projektgruppe vorbereitet.

Während der Besuche wurden die Schüler durch eine Präsentation von Claus Arndt in das Thema Open Data eingeführt und über das Projekt aufgeklärt. Danach sollten die Schülerinnen und Schüler unter Anleitung der Studierenden Ideen für Open-Data-Anwendungen generieren.

2.4 Unterrichtsbesuche

Die Unterrichtsbesuche fanden jeweils in der zwölften und elften Jahrgangsstufe in einem Sozialwissenschaften-Kurs und in der neunten Jahrgangsstufe in einem Politikkurs statt.

Die Schülerinnen und Schüler aller besuchten Kurse wurden durch Herrn Arndt in die Thematik Open Data eingeführt. Er erläuterte den in der Verwaltung stattfindenden Paradigmenwechsel:

> „Bisher: Alles ist geheim, was nicht ausdrücklich als öffentlich gekennzeichnet ist“

> „Jetzt: Alles ist öffentlich, was nicht ausdrücklich als geheim gekennzeichnet ist“

und die damit verbundenen Problematiken. Es wurde darüber aufgeklärt, dass die zuständigen Stellen der Verwaltung in vielen Fällen selbst nicht einschätzen können, wie interessant ihre Daten für Entwickler bzw. Bürger sind. Als Beispiel für die sogenannten „Schnarchdaten“ gab Claus Arndt die beliebtesten Vornamen der Stadt Moers an. Ein Entwickler erstellte mithilfe dieser Datensätze von mehreren Städten die App „Babybenamsen“, bei der die beliebtesten Vornamen Deutschlands für werdende Eltern zusammengestellt wurden. Darauf folgten die Prinzipien von Open Data[9]. Offene Datensätze müssen u.a. strukturiert, maschinenlesbar, lizenzfrei und vollständig sein.

Die genannten Prinzipien wurden anhand des Beispiels „offener Haushalt“ verdeutlicht. Herr Arndt stellte einen Vergleich zwischen dem alten tabellarischen Haushalt einer Stadtverwaltung und der Visualisierung auf http://www.offenerhaushalt.de/ an. Mit diesem Beispiel konnte den Schülerinnen und Schülern das Potenzial der maschinenlesbaren Datensätze veranschaulicht werden.

Im Anschluss daran machte Claus Arndt explizit auf die Chancen und den Nutzen aber auch auf die Risiken und Probleme von Open Data aufmerksam. Siehe dazu genauer *Anhang 1*.

[9] Quelle: http://opendata-network.org/2010/02/8-open-government-data-principles-vollstaendigkeit/ Letzter Aufruf: 15.08.2014.

Nach der einführenden Präsentation wurden die Schülerinnen und Schüler in Arbeitsgruppen eingeteilt und erhielten von den Studierenden die Aufgabe, Ideen für Open Data-Anwendungen zu sammeln. In der elften und zwölften Jahrgangsstufe lief dies ohne feste Vorgaben nach Brainstorming-Methode ab. Die Arbeitsgruppen kamen dabei zu folgenden Ergebnissen.

	A	B	C	D	E
1	**Frage / Anliegen**	**Benötigte Daten**	**Design**	**Effekt / Nutzen**	**Gibt es das schon?**
2	Univergleich	Daten von versch. Universitäten / Daten der Unis über Studenten Professoren etc.	Mehrere Möglichkeiten (z.B. App, Website)	Entscheidung, wo man sich bewirbt	Ja! Uni-Vergleich.de
3	Freizeitgestaltung	Öffentliche Veranstaltungen		Mehr Beteiligung	
4	Hilfe zur Steuererklärung	Wissen der Steuerfachangestellten	Virtuelle Steuererklärung, wo man drauf klicken kann & Erklärung folgt -> Mustererklärung !	1) Geld einsparen 2) Zeit einsparen 3)Aufwand verringern	Nur in Ansätzen, Erläuterungsbögen in Papierform beim FA
5	Wahlbeteiligung und Orientierung	Wahlergebnisse / politische Umfragen	geographisch, App, Webanwendung	Rückschluss auf sozialen Stand, Zusammenhang mit Einkommen	Teilweise
6	rote Ampel-Statistik	Anzahl der Ampeln und ihre Phasen	App, Navi	Kommt schneller an sein Ziel, weniger Spritverbrauch	
7	Fahrpläne für Busbahn, Zug u. S-Bahn				VRR-App, Pendel Panda / Verbesserung: Verspätung von Bussen anzeigen

Abbildung 1: Ergebnisse des Brainstormings

Zwischen den Unterrichtsbesuchen in der Oberstufe und dem Besuch in der neunten Klasse, lies Herr Nolte der Projektgruppe eine Liste über die aus seiner Sicht interessanten Datensätze der Stadt Moers zukommen. Auf dieser Liste befanden sich Wahldaten, Haushaltsdaten, Geodaten und Sozialdaten. Daraus entstand erstmals die Idee, Wahlergebnisse der Stadt Moers auf einer Karte zu visualisieren und diese über Bevölkerungsdaten in einen Kontext mit der sozialen Struktur in den Stimmbezirken zu bringen. Die Schülerinnen und Schüler der neunten Klasse erhielten aus diesem Grund die Aufgabe, Konzepte für eine Anwendung zu skizzieren, die in der Lage sein soll, Wahlergebnisse auf einer Karte zu visualisieren. Dabei war den Arbeitsgruppen freigestellt, ob sie die Anwendung als Desktop-Software, Website oder App konzipieren.

2.5 Auswertung der Unterrichtsbesuche

Im Nachgang zu den Unterrichtsbesuchen wurden die Ergebnisse[10] durch die Studierenden ausgewertet. Bei dem Brainstorming wurden von den Arbeitsgruppen einige Anwendungen für Datensätze vorgeschlagen, die nicht der Stadtverwaltung gehören. Die Stadt Moers verfügt nicht über Daten zu Hochschulen, Steuererklärungen oder Fahrplänen von Bussen und Bahnen. Die beiden Ideen für eine Veranstaltungs- und eine Rote-Ampeln-App waren nicht in den Lehrplan zu integrieren. Diese Konzepte wurden in diesem Projekt nicht weiter verfolgt. Mit dem Lehrplan und den zur Verfügung stehenden Daten war daher nur die Idee zu einer Anwendung vereinbar, die Wahlbeteiligung und Orientierung der Wähler ausgibt.

Die aus dem Unterrichtsbesuch in der neunten Jahrgangsstufe hervorgegangenen Konzepte überschnitten sich aufgrund der Aufgabenstellung alle thematisch mit der reali-

[10] Siehe Anhang 2 – Unterrichtsbesuche.

sierbaren Idee aus dem Brainstorming. Die Schülerinnen und Schüler erstellten mehrere detaillierte Papierprototypen von Smartphone-Apps und Webseiten, die Wahlergebnisse auf einer Karte anzeigen. Die Arbeitsgruppen die sich für eine Smartphone-App entschieden, waren jedoch bei der Anzahl der Funktionen und der aufrufbaren Daten aufgrund des Platzmangels auf dem Bildschirm eingeschränkt. Insgesamt ergaben die Prototypen der Webanwendung ein runderes System in Bezug auf Funktionalität und Übersichtlichkeit.

2.6 Konzept Wahlergebnisplattform

Aus den Ergebnissen der Unterrichtsbesuche und der durch Herrn Nolte erstellten Liste der interessanten Datensätze, ergab sich das erste von zwei Konzepten, die während dieses Projekts technisch realisiert werden sollten.

Bei diesem Konzept handelte es sich um eine Wahlergebnisplattform, auf der Wahlergebnisse in Form von CSV-Tabellen hochgeladen und auf einer Karte visualisiert werden konnten. Um die Funktionalität und den Aufbau der Webanwendungen besser mit den Lehrkräften kommunizieren zu können, erstellte die Projektgruppe einen Mock-Up[11].

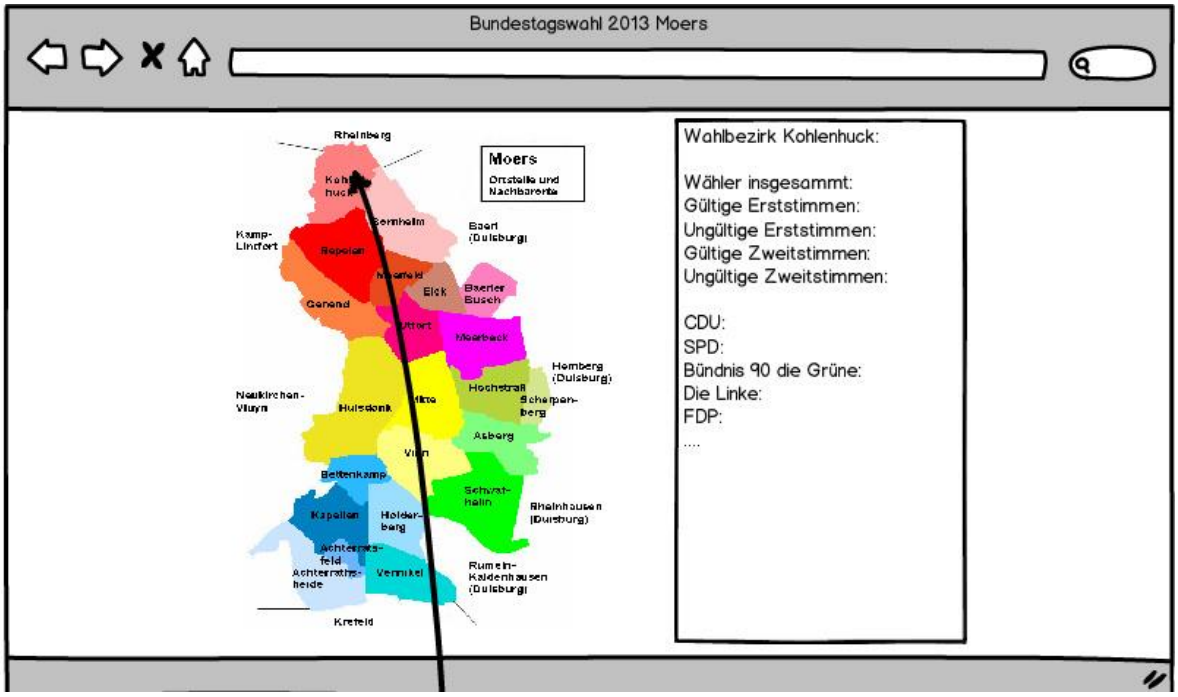


Abbildung 2: Mock-Up Bild 1

Die Webseite sollte eine Umrisskarte der Stadt Moers zeigen, auf der die Stimmbezirke des Stadtgebiets in der Farbe der Gewinnerpartei eingezeichnet sind. Neben der Karte werden die Wahlergebnisse für das Stadtgebiet eingeblendet.

[11] Modell zu Präsentationszwecken.

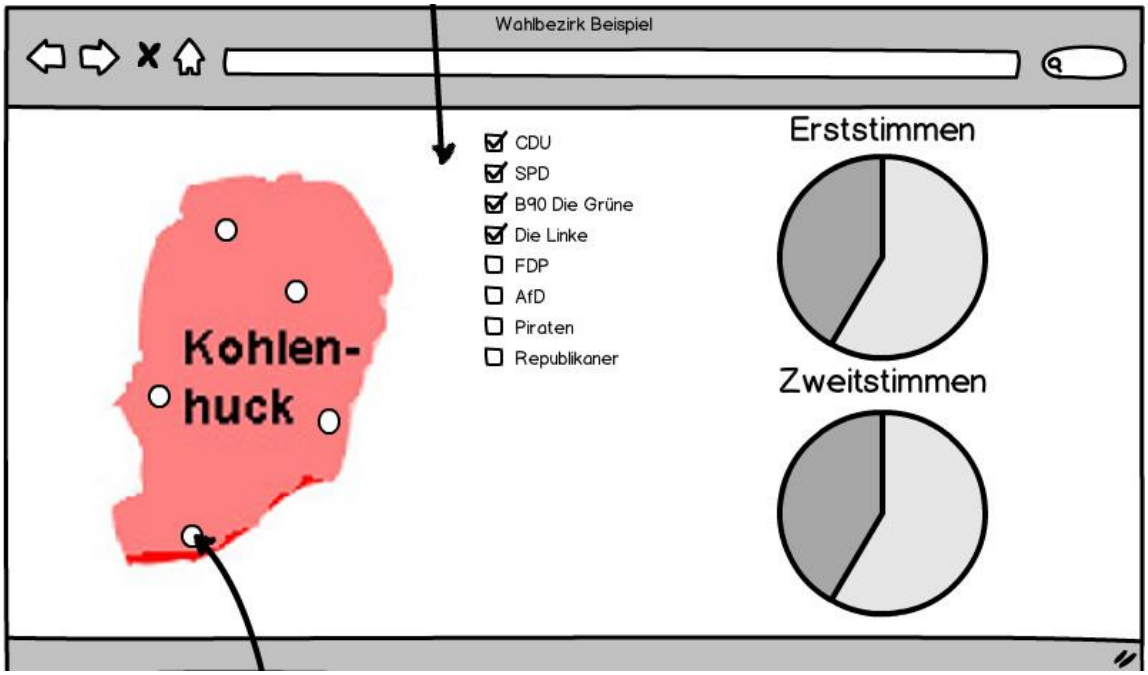


Abbildung 3: Mock-Up Bild 2

Durch Klicken auf einen Stimmbezirk, erreicht der Nutzer eine neue Ansicht. Wie in *Abbildung 3* zu erkennen ist, wurde hier eine separate Einblendung des Stimmbezirks eingeplant. In dem ausgewählten Stimmbezirk sollten nun auch die jeweiligen Wahllokale erkennbar gemacht werden. Neben der Karte werden erneut die Wahlergebnisse eingeblendet. Der Detailgrad der Anzeige sollte über eine Checkbox veränderbar gemacht werden.

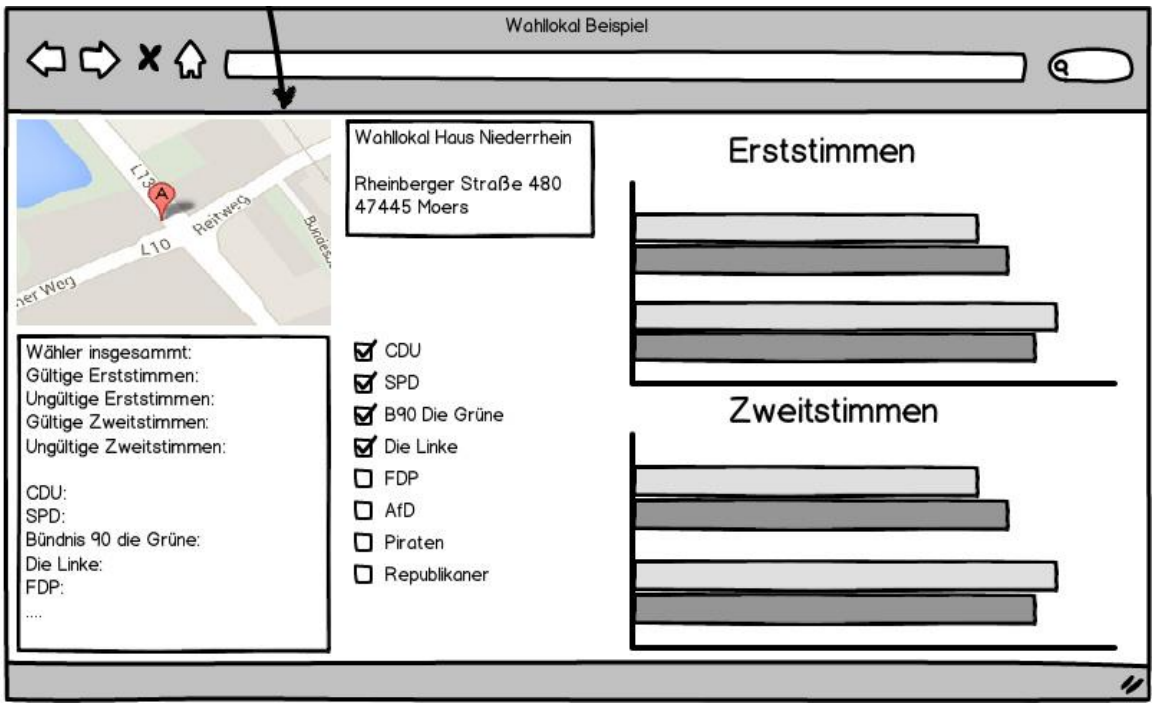


Abbildung 4: Mock-Up Bild 3

Durch Klicken auf ein Wahllokal gelangt der Nutzer zu der Detailansicht. Hier werden Name und Adresse des Wahllokals ausgegeben. Außerdem werden hier die Wahlergebnisse in gewohnter Struktur für das jeweilige Wahllokal angezeigt.

2.7 Konzept ‚offener Haushalt'

Während der Unterrichtsbesuche machte Herr Arndt auf die Visualisierung des Bundeshaushaltes[12] aufmerksam. Das Feedback zu dieser Umsetzung von Seiten der Schülerinnen und Schüler sowie der Lehrkräfte fiel positiv aus, da Finanzplanungen von Bund, Ländern und Kommunen Teil des Lehrplans im Fach Sozialwissenschaften waren. Aus dieser Idee entstand das zweite Konzept.

Angelehnt an die bestehende Visualisierung des Bundeshaushaltes, sollte nun der Haushalt der Stadt Moers auf ähnliche Weise visualisiert werden. Die Entscheidung darüber, mit welchem Tool der Haushalt visualisiert werden soll, fiel auf https://openspending.org/[13]. Hier ist es möglich, Finanzbewegungen von Städten und/oder Ländern mit einer ansprechenden und simplen Visualisierung darzustellen. Dies bietet gegenüber der tabellarischen Form den Vorteil der Übersichtlichkeit. Es lässt sich am besten am Beispiel des bereits erwähnten Bundeshaushalts verdeutlichen. Die Darstellung ist kachelartig, wobei jede der Kacheln ein Ministerium bzw. einen Bereich innerhalb der Verwaltung darstellt. Die Größe der jeweiligen Kachel verdeutlicht den Anteil des Bereichs an den Gesamtausgaben. Unter jeder Ebene befindet sich eine Tabelle, in der die Kacheln nochmals einzeln aufgeführt sind und deren explizite Ausgaben sowie die prozentuale Beteiligung an den Gesamtausgaben zu sehen ist. Durch einen Klick auf die einzelnen Bereiche gelangt man auf die nächste Ebene, bei der die Ausgaben nun den verschiedenen Aufgabenbereichen zugeteilt sind. In der dritten Ebene, die ebenfalls durch Klicken zu erreichen ist, sieht man detailliert, wofür die Mittel aufgewendet wurden. In allen Ebenen bleibt die Tabelle mit der Auflistung aller Ausgaben erhalten. So wird eine einfache aber wirkungsvolle Visualisierung erreicht, welche für den Schulunterricht brauchbar ist, da man die Veränderung der Ausgaben bzw. Einnahmen über mehrere Jahre hinweg erkennen kann. Auf dieser Grundlage sollte der Haushalt der Stadt Moers visualisiert werden, wobei die Ebenen die Fachbereiche, die Fachdienste und die Produkte darstellen sollten.

3. Technische Umsetzung

3.1 Open Spending

Die Projektgruppe stieß bei der Visualisierung des Haushalts auf Open Spending auf Probleme. Die Website benötigt eine CSV[14]-Datei in einer bestimmten Formatierung, um die Ebenen (genannt „Dimensions"), darzustellen. In den Tutorials auf der Seite sind Beispiele und Texte zur Formatierung von CSV-Dateien zu finden. Der Haushalt der Stadt Moers liegt allerdings nur in XML[15]-oder PDF[16]-Format vor. Das XML-Format wurde aufgrund der Richtlinien zu Open Data gewählt, da es maschinenlesbar ist. So musste die XML-Datei in eine CSV-Datei konvertiert werden. Nach Recherchen im Internet fan-

[12] Siehe: http://bund.offenerhaushalt.de/
[13] Open Spending ist eine von der Open Knowledge Foundation bereitgestellte Seite, um Finanzbewegungen von Ländern oder Kommunen mithilfe von Open Data zu visualisieren.
[14] Comma-separated values (eine bestimmte Form einer Excel-Tabelle).
[15] Extensible Markup Language (Auszeichnungssprache für hierarchische Strukturen).
[16] Portable Document Format (plattformunabhängiges Dateiformat für Dokumente).

den sich einige sogenannte „Converter“, mit denen dies möglich sein sollte. Mehrere davon wurden ausgiebig getestet, wobei es bei jedem zu Unstimmigkeiten im Ergebnis kam. Bei den Tests war es nicht möglich, alle in der XML-Datei vorhandenen Daten zu extrahieren und in die CSV-Datei zu übertragen. Nach Rücksprache mit Herrn Arndt wurden die in der Open Data-Community aktiven Experten für maschinenlesbare Formate um Rat gefragt. Daraufhin gab es einige Rückmeldungen bezüglich der Converter und des Problems. Von besonders engagierten Mitgliedern wurde extra für diesen Zweck ein Python[17]-Script geschrieben, welches die vorhandene XML-Datei in eine CSV-Datei konvertieren sollte. Das Produkt aus diesem Script war allerdings auch nicht für Darstellung des Haushalts auf Open Spending geeignet, da während des Konvertierungsprozesses Inhalte verloren gingen, oder nicht richtig übernommen wurden. Die entstandene CSV-Datei war auch nicht mit der von Open Spending vorgegebenen Struktur konform, da sie teilweise Leerzeilen enthielt oder das Zahlenformat nicht korrekt war. Dazu kam es aufgrund der komplizierten Verschachtelung der XML-Datei, da sie den kompletten Haushalt der im Normalfall als PDF-Datei vorliegt enthält.

```
@@ -0,0 +1,178 @@
+#coding: utf-8
+import os
+import sys
+from lxml import etree
+import dataset
+
+def show_path(el):
+    def get_path(el):
+        p = el.getparent()
+        if p is None:
+            return [el.tag]
+        return get_path(p) + [el.tag]
+    print get_path(el)
+
+def to_float(n):
+    try:
+        return float(n)
+    except:
+        return None
+
+def to_int(n):
+    try:
+        return int(n)
+    except:
+        return None
+
+AMOUNTS = {
+    'ANS_PLJ': {
+        'art': 'Ansatz',
+        'jahr': lambda y: y
+    },
+    'ANS_VJ': {
+        'art': 'Ansatz',
+        'jahr': lambda y: y - 1
+    },
+    'PLAN_FJ1': {
+        'art': 'Plan',
+        'jahr': lambda y: y + 1
+    },
+    'PLAN_FJ2': {
+        'art': 'Plan',
+        'jahr': lambda y: y + 2
+    },
+    'PLAN_FJ3': {
```

Abbildung 5: Ausschnitt[18] aus dem von der Community entwickelten Python Script[19] zur Konvertierung der XML-Datei.

Aufgrund dieser Fehlversuche wurde auf das Thema Konvertierung während der „Arbeitsgruppe Open Data“ im Kommunalen Rechenzentrum Niederrhein von Herrn Arndt angesprochen. Es wurde daraufhin mit Hilfe der Mitarbeiter des KRZN eine Lösung für das Problem gesucht, welche aber aufgrund der technischen Möglichkeiten nicht gefunden wurde. Das Hauptproblem bestand dabei in der von Open Spending geforderten Struktur innerhalb der Datei.

---

[17] Programmiersprache (siehe: https://www.python.org/).
[18] Der komplette Code ist nachzulesen unter: https://gist.github.com/pudo/8563444/revisions.
[19] Autor: Friedrich Lindenberg alias pudo.

Die letzte Möglichkeit bestand darin, die benötigte Struktur händisch zu erreichen, d.h. die benötigten Daten aus der PDF-Datei mit dem Haushalt auszulesen und in eine Excel-Tabelle einzutragen. Die Tabelle kann dann als CSV-Datei abgespeichert werden. Die beiden Studenten nahmen die funktionierende CSV-Datei der Stadt Bonn als Vorlage, um sich an der Struktur zu orientieren. Zum Testen[20] wurde ein Fachdienst (in der Tabelle „Department“ genannt) ausgelesen und eingetragen. Die einzelnen Produkte stellen hierbei die Ausgaben für das jeweilige Jahr dar. Beim Eintragen galt es drauf zu achten, dass sich keine Leerzeilen in der Tabelle befinden und dass das Datum sowie die vorhandenen Zahlen in dem von Open Spending vorgegebenen Format[21] übertragen werden.

	A	B	C	D	E	F	G	H	I	J
1	id,date,group,main-department,department,produkt,budget									
2	1,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Steuern und Ã¤hnliche Abgaben,0									
3	2,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Steuern und Ã¤hnliche Abgaben,0									
4	3,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Steuern und Ã¤hnliche Abgaben,0									
5	4,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Zuwendungen und allgemeine Umlagen,0									
6	5,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Zuwendungen und allgemeine Umlagen,0									
7	6,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Zuwendungen und allgemeine Umlagen,0									
8	7,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Sonstige TransferertrÃ¤ge,0									
9	8,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Sonstige TransferertrÃ¤ge,0									
10	9,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Sonstige TransferertrÃ¤ge,0									
11	10,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Ã–ffentlich-rechtliche Leistungsentgelte,0									
12	11,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Ã–ffentlich-rechtliche Leistungsentgelte,0									
13	12,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Ã–ffentlich-rechtliche Leistungsentgelte,0									
14	13,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Privatrechtliche Leistungsentgelte,1593									
15	14,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Privatrechtliche Leistungsentgelte,60									
16	15,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Privatrechtliche Leistungsentgelte,62									
17	16,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Kostenerstattungen und Kostenumlagen,11199									
18	17,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Kostenerstattungen und Kostenumlagen,10674									
19	18,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Kostenerstattungen und Kostenumlagen,10674									
20	19,2012,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Sonstige ordentliche ErtrÃ¤ge,0									
21	20,2013,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Sonstige ordentliche ErtrÃ¤ge,0									
22	21,2014,Innere Verwaltung,Verwaltungssteuerung und Service,Politische Gremien,Sonstige ordentliche ErtrÃ¤ge,0									

Abbildung 6: Teil der händisch erstellten funktionierenden CSV-Datei.

3.1.1 Die Modellierung

Nach dem Übertragen der Daten in die Tabelle wurde auf Open Spending ein Account sowie ein neuer Datensatz[22] für die Stadt Moers angelegt. Daraufhin wurde die Tabelle hochgeladen. Dabei ist zu beachten, dass man die CSV-Datei über den Link direkt aufrufen konnte und man nicht in Unterverzeichnisse navigieren musste. Auch dazu war auf Open Spending ein Tutorial zu finden. Nachdem diese Vorrausetzungen erfüllt waren, wurde die Tabelle hochgeladen, wobei die Seite automatisch die Spalten erkannte, sofern sie in der richtigen Struktur vorlagen. Nachdem die Seite keine Fehlermeldung ausgegeben hatte, wurden die sogenannten „Dimensions[23]“ für die spätere Visualisierung festgelegt. Auch dafür gelten spezielle Vorgaben, die eingehalten werden müssen, damit

[20] Aus zeitlichen Gründen wurde zu Testzwecken lediglich ein Fachdienst ausgewählt.
[21] Im Falle des Datums bedeutet dies: entweder „YYYY-MM-DD“ oder „YYYY“, die Zahlen dürfen keine Kommata enthalten. (die gesamte Formatierung ist nachzulesen auf: http://community.openspending.org/help/guide/en/formatting-data/)
[22] Datensatz bedeutet in dem Fall, dass ein neuer Eintrag für die Stadt Moers angelegt wurde.
[23] Dimensionen, welche in der Visualisierung die einzelnen „Kacheln“ repräsentieren und zur Identifikation dienen.

es keine Probleme gibt. Es gibt drei Kern-Dimensionen, die immer in der Tabelle enthalten sein müssen:

1. Die „**Time-Dimension**“, enthält das nach den Richtlinien formatierte Datum der jeweiligen Transaktion.
2. Die „**Amount-Dimension**“, enthält das Budget des jeweiligen Produktes, also die eigentliche Finanzbewegung.
3. Die „**ID-Dimension**“, dient zur Identifikation der Spalten. (Nummeriert diese von 1-n durch)

Ohne diese drei Dimensionen ist es nicht möglich eine Visualisierung zu erstellen. Nach dem Festlegen der Kern-Dimensionen, wurden die anderen Spalten als Dimensionen verarbeitet. Wichtig ist hierbei, dass jede Spalte eine eigene Dimension zugewiesen bekommt. Zur eindeutigen Identifizierung wird jeder Spalte noch ein Unique-Key hinzugefügt.

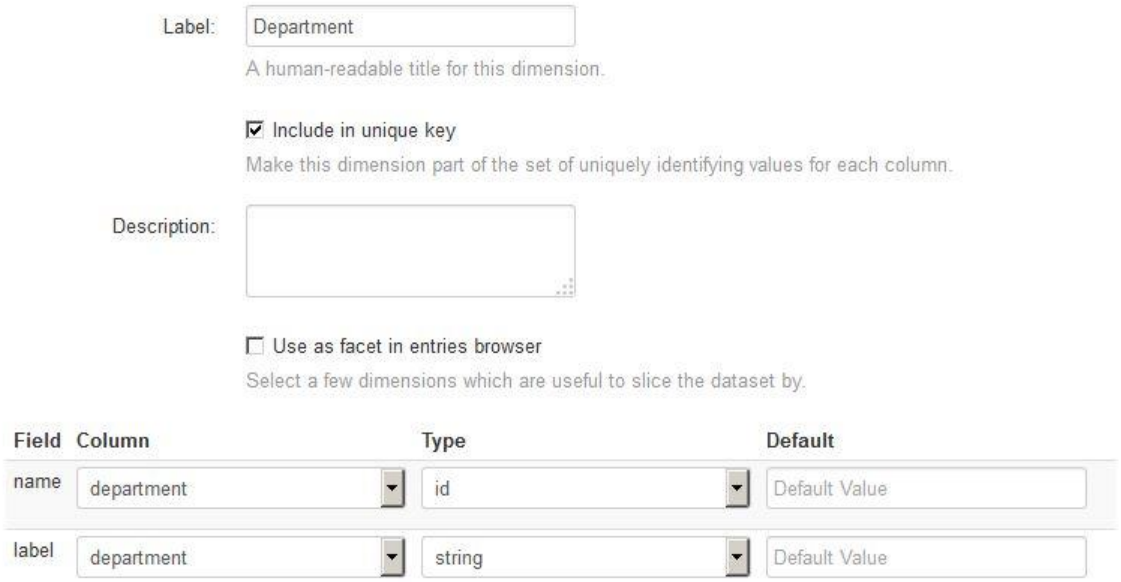


Abbildung 7: Erstellen der „Department“-Dimension auf Open Spending

In der *Abbildung 7* ist zu sehen, wie die Dimension für den Fachdienst (gen. „Department“) erstellt wird. Die Spalte hat einen Unique-Key und es wird ein Name sowie ein Bezeichner für die Spalte festgelegt, der als String[24] übergeben wird. Dies wird in ähnlicher Weise für alle anderen in der späteren Visualisierung gewünschten Spalten durchgeführt.

Nachdem alle Dimensionen festgelegt und gespeichert wurden, mussten sie geladen werden. Dies geschah zunächst durch einen Testlauf, wobei die Seite nochmals die Formate sowie die Unique-Keys überprüfte. Nach erfolgreichem Testlauf wurde die Visualisierung erstellt.

[24] Zeichenkette.

3.1.2 Die Visualisierung

Bei der Visualisierung repräsentiert jede Spalte eine eigene Ebene innerhalb des Modells. Die Zusammenhänge der Spalten werden durch die jeweils tieferen Ebenen dargestellt. Die Detailstufe des Modells hängt von der Verschachtelung der CSV-Datei ab.

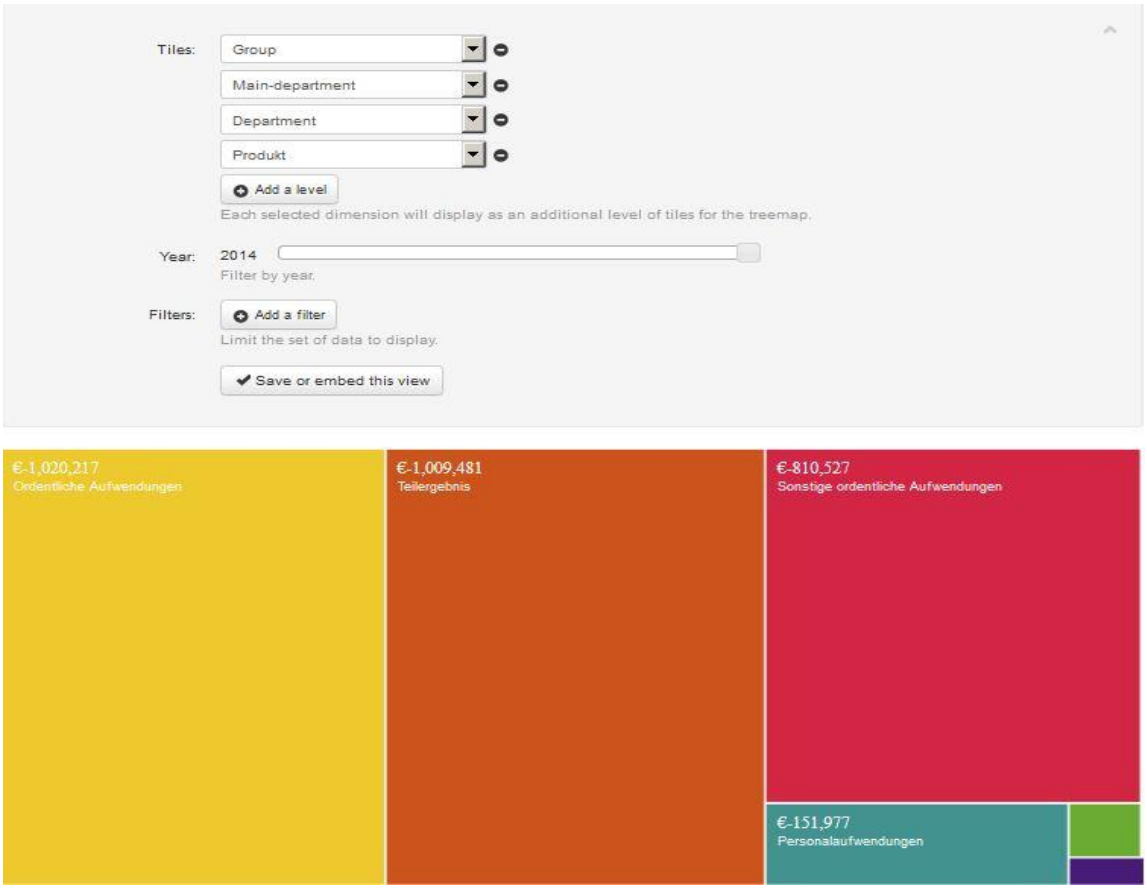


Abbildung 8: Erstellung der Visualisierung auf Open Spending

Auf der Abbildung sieht man das Menü zur Erstellung einer Visualisierung. Gut zu erkennen ist die Verschachtelung der einzelnen Tiles[25]. Die Seite berechnet automatisch den Anteil des Elements am Gesamtumfang und teilt die Ebene dementsprechend ein. Nach der Modellierung kann man das Modell speichern und veröffentlichen. Das von den Studierenden erstellte Testmodell wurde ebenfalls veröffentlicht und ist unter https://openspending.org/moers_test5/views/test zu finden.
Durch eine derartige Visualisierung können die Finanzbewegungen einer Kommune einfach dargestellt und aufgrund der freien Verfügbarkeit auch im Schulunterricht verwendet werden. Mit den Offenen Daten der eigenen Kommune bietet sich auch noch die Möglichkeit, mit aktuellen und interessanten Daten zu arbeiten und die Verbundenheit gegenüber der eigenen Stadt zu stärken.

[25] Ebenen der Visualisierung

3.2 Wahlergebnisplattform

3.2.1 Marktforschung

Vor der Umsetzung auf der eigentlichen Webseite gab es zahlreiche Gespräche, um die beste Möglichkeit zu finden, die Karte und die Einfärbung darzustellen. Durch die Open Data-Community wurde den Studierenden das Programm TileMill empfohlen. Hier ist es möglich, GeoJson[26]-Dateien zu importieren und darauf dann mithilfe der „Mustache Template Language[27]“ verschiedene Layer und auch die Wahllokale darzustellen. Um weitere Informationen über die Modellierung von Geodaten, insbesondere Shapefiles zu erhalten, hielten die Studierenden Rücksprache mit dem Fachbereich 07 – Vermessung und Bauaufsicht der Stadt Moers. In dem Gespräch stellte sich heraus, dass vor der Einfärbung der Karte in TileMill, die sogenannte Attributtabelle des Shapefiles bearbeitet werden musste. Um diese Tabelle zu bearbeiten, benötigten die Studenten ein geographisches Informationssystem. Die Studierenden entschieden sich dazu unter Absprache mit dem gerade erwähnten Fachbereich für das Open Source-Programm Qgis[28]. In diesem Programm ist es möglich Shapefiles zu importieren und diese dann beliebig zu bearbeiten. Unter diese Bearbeitung fällt auch die Modifikation der Attributstabelle.

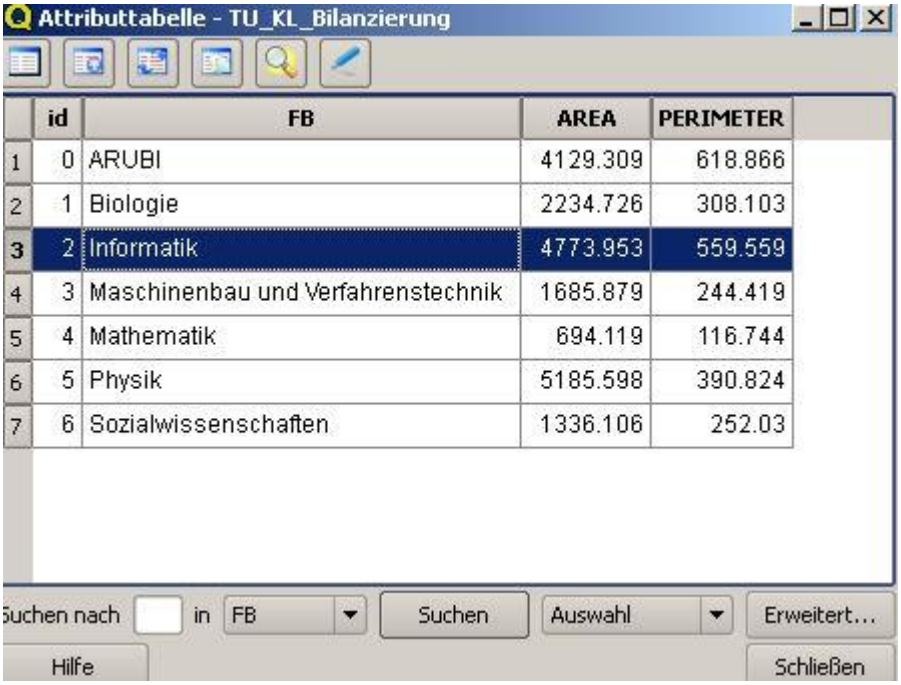


Abbildung 9: Beispiel einer Atributtabelle in Qgis[29]

Auf der *Abbildung 9* sieht man eine Attributtabelle in Qgis. Um mit dem Shapefile arbeiten zu können wurde von den Studierenden eine neue Spalte in der Tabelle hinzugefügt. In dieser Spalte wurden die Gewinner des jeweiligen Wahlbezirkes eingetragen. Nach dem händischen Eintragen des Gewinners wurde das Shapefile in GeoJson konvertiert. In diesem Format lässt sich die Karte in TileMill importieren und bearbeiten.

---

[26] Format zur Notation und Interpretation von Geodaten.
[27] Eine an HTML angelehnte Sprache zur Modellierung von digitalen Inhalten.
[28] Siehe: http://www.qgis.org/de/site/.
[29] Quelle: http://speyererraumplaner.files.wordpress.com/2009/07/bilanzierung.png (Stand 22.7.14).

Nach dem Importieren in TileMill erkennt das Programm automatisch die Koordinaten des Layers und setzt ihn an die richtige Position. Durch die neu hinzugefügte Spalte und der „Mustache Template Language“ lässt sich nun ein Filter erstellen mit dem die Karte eingefärbt wird.

```
21
22  [MEISSTIMM="SPD"] {
23    polygon-fill:#ff0000;
24   }
25 [MEISSTIMM="CDU"]{
26    polygon-fill:#000;
27   }
```

Abbildung 10: Der Filter zum Einfärben in TileMill

Durch den auf der *Abbildung 10* zu sehenden Filter werden die einzelnen Bezirke der Karte in der richtigen Farbe eingefärbt. Die von den Studierenden in der Attributtabelle angelegte Spalte heißt „MEISTSTIMM“ und enthält den Gewinner des jeweiligen Bezirkes. Der Filter erkennt den Gewinner und färbt die Polygone demensprechend in der richtigen Farbe ein. Diese Filterung lässt sich beliebig auf andere Faktoren erweitern.

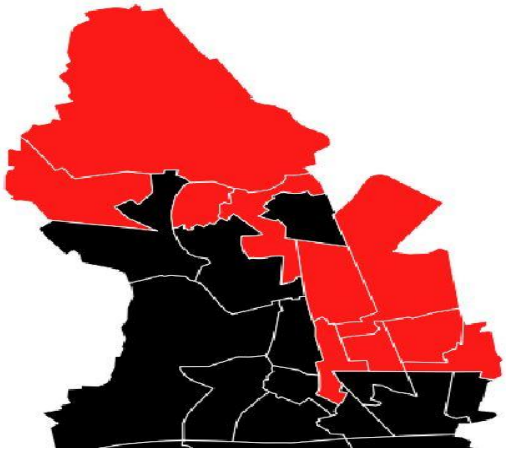


Abbildung 11: Oberer Teil der Karte nach der Einfärbung durch den Filter

Nach der Einfärbung der Karte durch den Filter, wurden die Wahllokale auf die Karte gebracht und modelliert. In TileMill ist es möglich CSV-Dateien zu importieren. Die Studierenden benutzten die bereits vorhandene CSV-Datei mit den Wahlergebnissen und den Geokoordinaten der Wahllokale. Durch das Importieren werden die vorhandenen Geokoordinaten automatisch auf der Karte als Punkte dargestellt. Durch die Mustache Template Language lassen sich nun die Pop-Ups bei Mouseover und die Legenden die bei einem Klick erscheinen, erstellen und modifizieren.

Die Daten aus der Tabelle konnten dazu genutzt werden, um die Pop-Ups insofern zu bearbeiten, dass man beim mouseover den Namen und die Straße des Wahllokals angezeigt bekommt und bei Klick die Ergebnisse aus dem Wahllokal in einer Legende ausgegeben bekommt. Zunächst einmal ging es daran den sogenannten „Teaser“ zu bearbeiten. Der „Teaser“ erscheint bei Mouseover auf dem jeweiligen Wahllokal und er sollte nur die wichtigsten Informationen enthalten.

Content to be shown on hover or first tap (mobile).

```
<b>{{{WAHLLOKAL}}}</b>
<br>{{{STRASSE}}} {{{HAUSNUMMER}}}</br>

<br><br>Stimmbezirk: {{{STIMMBEZIRK}}}</br></br>
<br>    Ratswahlbezirk: {{{RATSWAHLBEZIRK_k}}} </br>

<br><br>Ergebnisse bei <u>Klick</u></br></br>
```

Abbildung 12: Programmierter Teaser in TileMIll

Der in der *Abbildung 12* zu sehende „Teaser“ gilt sowohl für Mouseover („hover“) als auch für den ersten Klick bei einem mobilen Endgerät. Er enthält die wichtigsten Informationen zu dem Wahllokal, wie z.B. den Namen, die Straße, die Hausnummer, den Ratswahl- und den Stimmbezirk. Des Weiteren enthält er einen Hinweis darauf, dass es die Ergebnisse für das Wahllokal per Klick auf das Wort „Klick“ gibt. Er wird auf der Karte als eine Art Sprechblase über dem jeweiligen Wahllokal dargestellt. Unter dem Reiter „Full“ ist es möglich die Legende zu bearbeiten, welche bei Klick erscheint. Bei der Modellierung der Legende stehen einem alle Spalten der Tabelle zur Verfügung, sodass man frei entscheiden kann welche Parteien man darstellen will und welche nicht.

```
<b>{{{WAHLLOKAL}}}</b>

<br>Wahlberechtigte: {{{WAHLBERECHTIGTE}}}</br>
<br>gueltige Stimmen: {{{Z_GUELTIG}}}</br>
<br>ungueltige Stimmen: {{{Z_UNGUELTIG}}}</br>
<br></br><br></br>
<b>Stimmverteilung: </b>
<br>Stimmen SPD: {{{Z_SPD}}}</br>
<br>Stimmen CDU: {{{Z_CDU}}}</br>
<br>Stimmen FDP: {{{Z_FDP}}}</br>
<br>Stimmen Die Gruene: {{{Z_GRUNE}}}</br>
<br>Stimmen Die Linke: {{{Z_DIELINKE}}}</br>
<br>Stimmen Piraten: {{{Z_PIRATEN}}}</br>
<br>Stimmen Afd: {{{Z_AfD}}}</br>
<br>Stimmen Big: {{{Z_BIG}}}</br>
<br>Stimmen Buen,21Rrp: {{{Z_Buen,21RRP}}}</br>
```

Abbildung 13: Modellierungsfenster für die Ergebnisse

Bei der Modellierung wurde darauf geachtet, dass nicht zu viele Informationen in dem Pop-Up stehen. Hierbei gilt: Alles was in {} steht wird durch den Inhalt der Spalten aus der Tabelle ersetzt und der Rest wird als String angezeigt. So sieht man in der *Abbildung 13* z.B., dass der Name des Wahllokals als Überschrift dargestellt wird. Darauf folgen die Spalten der Wahlberechtigten, gültigen sowie ungültigen Stimmen. Anschließend folgen die einzelnen Stimmen der Parteien. Hinter den absoluten Stimmen wäre es durchaus denkbar, auch die prozentualen Werte darzustellen, welche dann in einer entsprechenden Spalte in der Tabelle abgelegt werden müssten. Gut zu erkennen sind die HTML-

Elemente[30] in der „Mustache Template Language“. So wird z.B. durch ein „<b>“ ein Wort fett geschrieben oder durch ein „
“ ein Zeilenumbruch eingeleitet. Man kann in dieser Sprache auch andere HTML-Tags zum stylen von Text verwenden. Wenn man die Modellierung der Karte, des Teasers und der Legende abgeschlossen hat, kann man die Karte in verschiedene Formate exportieren. Unter anderem in verschiedene Bildformate, dabei ist allerdings zu beachten, dass dadurch die Teaser- und Full-Elemente verloren gehen und nur die Karte exportiert wird. Eine andere Möglichkeit ist, die fertige Karte auf Mapbox[31] hochzuladen. Dies ist direkt in TileMill möglich und benötigt lediglich einen Mapbox-Account. Durch das Hochladen kann die Karte veröffentlicht werden und ist dadurch öffentlich aufrufbar und nutzbar, was im Falle der Studierenden bedeutet, dass die Schule mit der Karte arbeiten kann. Für eine einmalige Visualisierung einer Wahl bieten die hier gezeigten Tools eine sehr einfache und schnelle Alternative.

Für das Projekt der Studierenden erwiesen sich diese Tools allerdings als nicht ausreichend, da sie nicht dynamisch sind. Immer wenn eine neue Wahl visualisiert werden soll, müssten alle Schritte durchlaufen werden und man müsste für jede Wahl eine einzelne Karte erstellen und neu veröffentlichen. Vergleiche zwischen mehreren Wahlen würden dadurch erschwert. Die Studierenden entschieden sich daher dazu, eine eigene Website mit einer interaktiven Map zu programmieren, die nachhaltig in der Lage ist, Wahlergebnisse aus einer Datenbank zu lesen und darzustellen. Dieses Vorgehen würde zwar mehr Programmieraufwand und Datenbankplanung als bei der Verwendung eines Tools bedeuten, jedoch die nötige Basis für eine gelungene Wahlergebnisplattform schaffen, die von den Schulen verwendet werden kann.

3.2.2 Datenbanken

Nachdem die Entscheidung zur Erstellung einer Webseite festgelegt wurde, ging es darum ein Konzept zu entwickeln, wie genau die vorhandenen Daten gespeichert und verwendet werden können. Die Wahl fiel dabei auf eine MySQL[32]-Datenbank. Die benötigten Daten sind dabei in einzelnen Tabellen angelegt, welche in einer Datenbank zusammengefasst sind. Das hat den Vorteil, dass die Datenbank mit Hilfe von PHP[33] in die Webseite eingebunden werden kann. Ein weiterer Vorteil von MySQL ist, dass es sich dabei um eine Open Source[34]-Software handelt und es aufgrund der Verbreitung wichtige Sicherheitsupdates gibt. Wichtig war es, dass Aufgrund des Codes die Tabellen bestimmte Namen und bestimmte Strukturen enthielten, sodass sie von dem Script gelesen werden können. Dabei gibt es vier verschiedene Tabellenarten. Im nächsten Kapitel wird auf die vier Arten genauer eingegangen.

---

[30] Es handelt sich dabei nicht um Elemente aus der Aktuellen HTML-Version 5.
[31] Mapbox ist eine quelloffene Plattform zum Erstellen und präsentieren von Karten. (siehe: https://www.mapbox.com).
[32] Eines der Weltweit verbreitetsten Datenbankverwaltungssysteme. Siehe: (http://www.mysql.de).
[33] „Hypertext Preprocessor“. Eine Scriptsprache zur Erstellung dynamischer Webseiten.
[34] Open-Source beschreibt eine quelloffene und kostenlose Software.

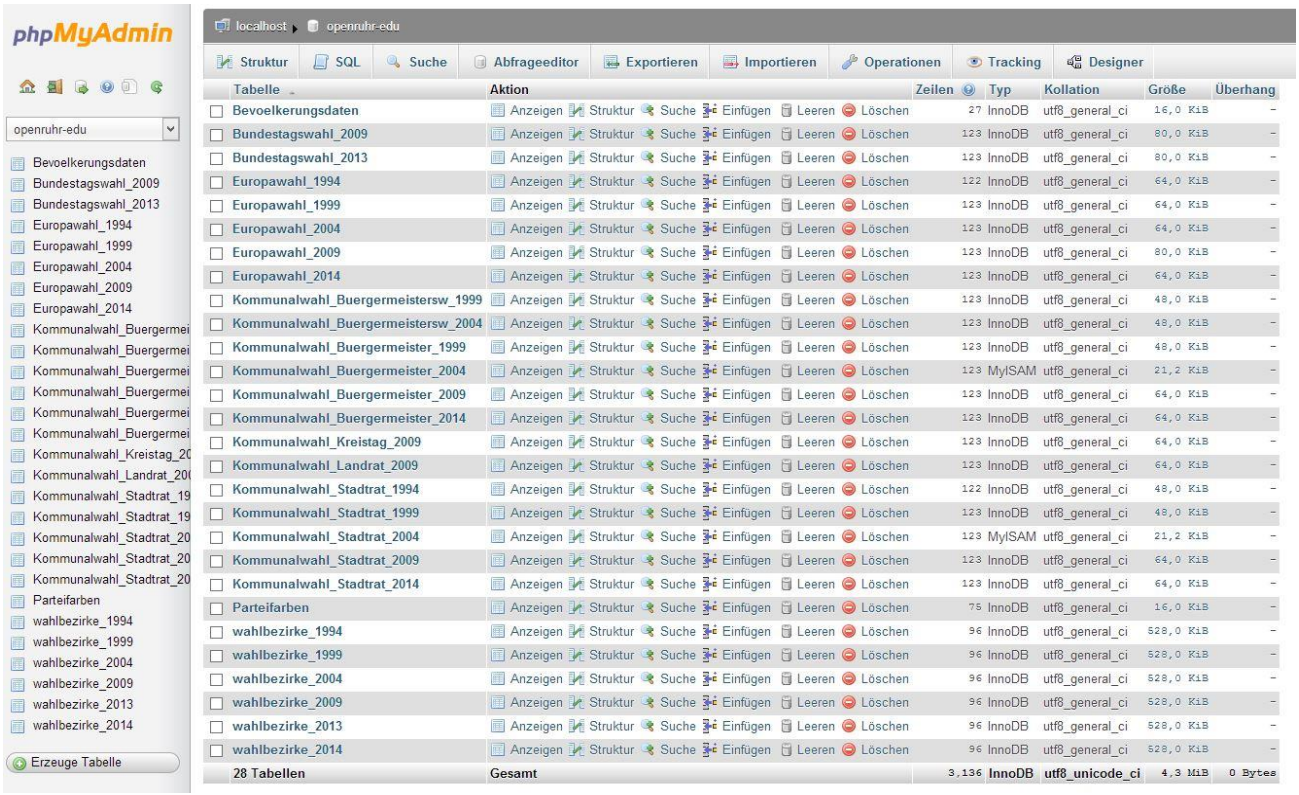


Abbildung 14: Die PhpMyAdmin Ansicht der MySQL-Datenbank

Auf der *Abbildung 14* ist das von den Studierenden genutzte Tool „phpMyAdmin“ zur Datenbankverwaltung zu sehen sowie der aktuellste Stand zum Zeitpunkt der Dokumentation der Datenbank mit allen vorhandenen Tabellen. Jede der Tabellen liegt zunächst als CSV-Datei vor und wird dann in die Datenbank importiert.

Die erste Tabellenart beinhaltet die Wahldaten zu der jeweiligen Wahl. Hier ist es wichtig, dass es für jede Wahl eine eigene Tabelle gibt, welche die nötigen Daten enthält. Die Tabelle benötigt eine bestimmte Struktur, um von dem Script ausgelesen werden zu können. Deshalb mussten die Tabellen, welche zunächst als CSV-Datei vorlagen, vor dem Importieren bearbeitet und in die richtige Struktur gebracht werden. So mussten z.B. einige Zeilen gelöscht werden[35] sowie andere Zeilen hinzugefügt werden.

STADTTEIL_k	RATSWAHLBEZIRK	RATSWAHLBEZIRK_k	STIMMBEZIRK	BEZIRKSART	WAHLLOKAL	STRASSE	HAUSNUMMER	Long	Lat
Moers	110	110 Huelsdonk	1101	Urnenwahlbezirk	Verwaltungsgebaeude ENNI	Am Jostenhof	7	6.608549	51.452390
Moers	110	110 Huelsdonk	1102	Urnenwahlbezirk	Staedtischer Kindergarten	Ruettgersweg	25	6.608160	51.448180
Moers	110	110 Huelsdonk	1103	Urnenwahlbezirk	Grundsch. Moers-Huelsdonk R.2	Ruettgersweg	19	6.608240	51.447760
Moers	110	110 Huelsdonk	1104	Urnenwahlbezirk	Kinder-u. Jugendbuero	Rathausplatz	1	6.626834	51.453460
Moers	110	110 Huelsdonk	1109	Briefwahlbezirk	Briefwahl Huelsdonk	Rathausplatz	1	6.626834	51.453460
Moers	111	111 Stadtmitte-Nord	1111	Urnenwahlbezirk	Autohaus Minrath	Rheinberger Strasse	61	6.630120	51.457990
Moers	111	111 Stadtmitte-Nord	1112	Urnenwahlbezirk	Gymn. Adolfinum. Pausenhalle	Wilhelm-Schroeder-Strasse	4	6.629067	51.453638
Moers	111	111 Stadtmitte-Nord	1113	Urnenwahlbezirk	Gebrueder-Grimm-Schule. Forum	Landwehrstrasse	51	6.633610	51.454160
Moers	111	111 Stadtmitte-Nord	1119	Briefwahlbezirk	Briefwahl Stadtmitte-Nord	fuer den Bezirk 111.9	2067	6.640814	51.451604
Moers	112	112 Stadtmitte-Altstadt	1121	Urnenwahlbezirk	Sparkasse Huelsdonk. Bereich 1	Huelsdonker Strasse	55	6.613801	51.451018
Moers	112	112 Stadtmitte-Altstadt	1122	Urnenwahlbezirk	Sparkasse Huelsdonk. Bereich 2	Huelsdonker Strasse	55	6.613801	51.451018
Moers	112	112 Stadtmitte-Altstadt	1123	Urnenwahlbezirk	Gerhard-Tersteegen-Haus	Haagstrasse	11	6.626522	51.450431
Moers	112	112 Stadtmitte-Altstadt	1124	Urnenwahlbezirk	Gymnasium in den Filder Benden	Zahnstrasse	43	6.620240	51.442520
Moers	112	112 Stadtmitte-Altstadt	1129	Briefwahlbezirk	Briefwahl Stadtmitte-Altstadt	fuer den Bezirk 112.9	3135	6.640814	51.451604
Moers	113	113 Stadtmitte-Sued	1131	Urnenwahlbezirk	Heinr.-Pattberg-	Uerdinger	74	6.635450	51.446140

Abbildung 15: Erste Hälfte einer Wahldatentabelle

Auf der *Abbildung 15* sieht man die erste Hälfte der Wahldatentabelle zur Bundestagswahl 2013. Die hier vorhandene Struktur findet sich ebenso in jeder anderen Wahldatentabelle wieder. Die Tabelle beinhaltet alle relevanten Daten wie den Stadtteil, die Ratswahlbezirksnummer, den Ratswahlbezirk, die Nummer des Stimmbezirks und die Bezirksart. Des Weiteren enthält sie den Namen des Wahllokals sowie die Straße und die Hausnummer. Die nächsten beiden Spalten sind für die Positionierung der Wahllokale auf der Karte besonders wichtig, denn sie enthalten die Geokoordinaten[36] der jeweiligen Wahllokale. Diese wurden mit Hilfe eines von Thorsten Magerstedt entwickelten Excel-Makros via Google Maps aufgezeichnet. Auf das Makro wird im nächsten Abschnitt genauer eingegangen.

[35] Die gelöschten Zeilen waren für das Projekt nicht von Wichtigkeit, so wurden z.B. Die Telefonnummer des Wahllokals oder der Name des zuständigen Hausmeisters gelöscht.
[36] Position des Wahllokals auf Längen- und Breitengrad.

WAHLBERECHTIGTE	WAEHLER/INNEN	E_UNGUELTIG	E_GUELTIG	E_CDU	E_CDU_%	E_SPD	E_SPD_%	E_FDP	E_FDP_%	E_GRUNE	E_GRUNE_%
854	505	10	495	230	46.46	219	44.24	8	1.62	7	1.41
1156	761	10	751	314	41.81	326	43.41	15	2.00	43	5.73
496	245	1	244	109	44.67	96	39.34	10	4.10	8	3.28
1144	645	10	635	238	37.48	281	44.25	15	2.36	33	5.20
0	702	2	700	313	44.71	272	38.86	28	4.00	37	5.29
956	479	5	474	151	31.86	236	49.79	9	1.90	15	3.16
661	343	5	338	100	29.59	150	44.38	8	2.37	19	5.62
997	538	11	527	164	31.12	265	50.28	13	2.47	24	4.55
0	413	7	406	157	38.67	161	39.66	6	1.48	43	10.59
735	422	13	409	140	34.23	193	47.19	4	0.98	22	5.38
716	409	5	404	163	40.35	169	41.83	15	3.71	16	3.96
819	460	6	454	191	42.07	176	38.77	13	2.86	30	6.61
806	486	5	481	238	49.48	173	35.97	16	3.33	21	4.37
0	564	2	562	256	45.55	201	35.77	16	2.85	24	4.27
841	458	4	454	182	40.09	199	43.83	9	1.98	13	2.86

Abbildung 16: Zweite Hälfte der Wahldatentabelle

In der zweiten Hälfte der Wahldatentabelle sind die Stimmen der einzelnen Parteien enthalten. Die Spalten „Wahlberechtigte“ sowie „Waehler/innen“ wurden von den Studierenden per Hand eingefügt, da es sich bei den Namen der Spalten vorher um statistische[37] Kennziffern gehandelt hat. Alle Spalten die mit einem „E_“ beginnen weisen auf die Erststimmen hin. Für die Zweitstimmen beginnen die Spalten mit einem „Z_“. Mit Hilfe der gültigen Stimmen und der Stimmen für die jeweilige Partei wurden in der Spalte „%“ die Anteile errechnet. Diese Daten werden auf der Webseite in Form von Balkendiagrammen dargestellt. Die Struktur ist notwendig, damit das Script die Tabelle lesen und interpretieren kann. Deshalb muss bei jeder neuen Wahl eine Tabelle mit dieser Struktur erstellt und hochgeladen werden. Der Name der Tabelle setzt sich dabei aus dem Namen der Wahl einem „_“ und dem Wahljahr zusammen. Der Name spielt beim Zugriff der Webseite auf die Datenbank eine wichtige Rolle. Darauf wird im Abschnitt *3.2.4 Programmierung* genauer eingegangen.

Das Makro ist innerhalb einer Excel Tabelle programmiert. Um die Geokoordinaten aufzuzeichnen benötigt es lediglich die Straße, die Haunummer und die Stadt. Daraufhin sucht es mit Hilfe der Google Maps API[38] die richtigen Geokoordinaten heraus und kann durch einen zweiten Klick auch einen Gegentest machen.

[37] Die Tabellen stammen ursprünglich von dem Statistiker der Stadt Moers.
[38] Application programming interface, eine Programmierschnittstelle.

	A	B	C	D	E	F	G	H	I	J
1	STRASSE	HAUSNUMM	PLZ	Stadt	Lat	Long				
2	Am Jostenhc	7		Moers	51.4523904	6.6085491				
3	Ruettgerswe	25		Moers	51.4481800	6.6081600				
4	Schwanenrir	5		Moers	51.4462974	6.6144741				
5	Unterwallstr	15		Moers	51.4533117	6.6269245				
6	Rathausplatz	1		Moers	51.4534603	6.6268341				
7	Rheinberger	61		Moers	51.4579900	6.6301200				
8	Wilhelm-Sch	4		Moers	51.4536388	6.6290676				
9	Landwehrstr	51		Moers	51.4541600	6.6336100				
10	fuer den Bez	2067		Moers						
11	Huelsdonker	55		Moers	51.4510180	6.6138019				
12	Meerstrasse	3		Moers	51.4515119	6.6248556				
13	Haagstrasse	11		Moers	51.4504310	6.6265225				
14	Zahnstrasse	43		Moers	51.4425200	6.6202400				
15	fuer den Bez	3135		Moers						

Hol mir die Daten von Google!

Mach den Gegentest

Abbildung 17: Tabelle zu Aufzeichnen der Geokoordinaten

```
Private Sub CommandButton1_Click()

    Dim response As String
    Dim erg As Variant
    Dim stadt As String

    Dim Strasse As String
    Dim PLZ As String
    Dim x As Integer
    Dim Ende As Long
    Dim Zellenummer As Long


    Ende = 1
    Do Until Cells(Ende + 1, 1) = ""
        Zellennummer = Ende + 1
        If Sheets(1).Range("E" + CStr(Zellennummer)).Value = "" Then

            counter = 0

            Set objHTTP = CreateObject("Microsoft.XMLHTTP")
            Strasse = Replace(Replace(Replace(Replace(Sheets(1).Range("A" + CStr(Zellennummer)).Value, "ß", "ss"), "ä", "ae"), "ö", "oe"), "ü", "ue")
            Strasse = Strasse & " " & Sheets(1).Range("B" + CStr(Zellennummer)).Value
            If InStr(LCase(Strasse), "fuer den") = 0 Then

                stadt = Sheets(1).Range("D" + CStr(Zellennummer)).Value
                PLZ = Sheets(1).Range("C" + CStr(Zellennummer)).Value
                URL = "http://maps.googleapis.com/maps/api/geocode/xml?address=" & Strasse & ",+" & stadt & ",+" & "germany&sensor=true"

                objHTTP.Open "Get", URL, False, "", ""
                objHTTP.setRequestHeader "Content-type", "application/x-www-form-urlencoded"
                Call objHTTP.send   'Request an WebServer übergeben!
```

Abbildung 18: Auszug aus dem Quellcode des Makros

In der oberen *Abbildung 17* sieht man die ausgefüllte Tabelle mit den beiden Buttons und den bereits aufgezeichneten Geokoordinaten. In der unteren *Abbildung 18* sieht man einen Auszug aus dem Quellcode des Makros, geschrieben in VBA[39].

In dieser Tabelle werden sämtliche Farben der einzelnen Parteien notiert, um diese dann auf der Website als Balkendiagramm verwenden zu können. Dafür war eine Recherchearbeit nötig, vor allem bei älteren, nicht mehr existenten Parteien. Da die Studierenden auch ältere Wahlen abbilden, mussten auch die Farben dieser Parteien in die Tabelle übertragen werden.

[39] Visual Basic for Applications, eine Scriptsprache um Abläufe innerhalb von Microsoft-Office-Programmen zu steuern.

PARTEI	FARBE	Parteiname
Candan	#eeeeee	Candan
DKP	#ff0000	DKP
50Plus	#f2f2f2	50 Plus
AUFBRUCH	#0000fd	Aufbruch
AUF	#003e7c	AUF
BP	#0099ff	Bayernpartei
CM	#6498ca	Christliche Mitte
DIEFRAUEN	#ff9600	Die Frauen
DIEGRAUEN	#394a94	Die grauen Panther
EDE	#7fb93e	EDE
FBI	#fb2742	FBI
DGP	#0000fe	DGP
Newropeans	#195da2	New Europeans

Abbildung 19: Ausschnitt aus der Parteifarbentabelle

Der Aufbau der Parteifarbentabelle ist relativ simpel und besteht aus den in der *Abbildung 19* zu sehenden drei Spalten. In der ersten Spalte befindet sich der Name der Partei und zwar exakt so, wie er auch in der Wahlergebnistabelle zu finden ist, da der Code nach diesem Namen in der Tabelle sucht (genauer unter *3.2.4 Programmierung* nach zu lesen). Die zweite Spalte enthält die Farbe der Partei im HTML[40]-Farbcode. Der Farbcode wird von dem Browser interpretiert und dann als Farbe dargestellt. In der letzten Spalte befindet sich der Anzeigename der Partei, welcher auf der Webseite dargestellt wird. Diese Tabelle muss nur ergänzt werden, wenn eine neue Partei an einer Wahl teilnimmt. Wenn dies der Fall ist, muss die neue Partei in die Tabelle mit Farbe und Anzeigename eingetragen werden. Sollte bei einer Wahl keine neue Partei antreten, so kann diese Tabelle universell für jede Wahl verwendet werden.

[40] Hypertext Markup Language, Auszeichnugssprache zur Darstellung von digitalen Inhalten.

Die Tabelle dient zur grafischen Darstellung der Bezirke auf der Karte der Webseite. Sie enthält alle nötigen Informationen um die einzelnen Bezirke auf die Karte zeichnen zu können.

BEZIRK	POLYGONE
1101	[[[6.604580685584836, 51.469168396761724], [6...
1102	[[[6.608419974554615, 51.451091272963332], [6...
1103	[[[6.609349877439452, 51.448462054109228], [6...
1104	[[[6.627852741664498, 51.462935238429516], [6...
1111	[[[6.636526430138319, 51.46507602678733], [6....
1112	[[[6.629962579591415, 51.454889452685379], [6...
1113	[[[6.638292148814103, 51.45998486318561], [6....
1121	[[[6.611873175148427, 51.451207720152894], [6...
1122	[[[6.621545358341627, 51.452884896828195], [6...
1123	[[[6.626857635435026, 51.45318251683647], [6....
1124	[[[6.623334611167013, 51.445407837860742], [6...
1131	[[[6.635249146417599, 51.449994821394093], [6...
1132	[[[6.629563267239106, 51.449813946803992], [6...
1133	[[[6.640793428913033, 51.451948899034868], [6...
1141	[[[6.639910198173197, 51.460974769590024], [6...

Abbildung 20: Tabelle zum Zeichnen der Wahlbezirke

Die Tabelle zur Darstellung der Wahlbezirke besteht aus zwei Spalten. Die erste Spalte enthält fortlaufend die Bezirksnummer, wie sie auch schon in der Tabelle mit den Wahldaten zu sehen war. In der zweiten Spalte sind die Polygone[41] zur Darstellung der Bezirke zu sehen. Sie enthalten die jeweiligen Eckpunkte des Bezirkes, sodass dieser dann mit Hilfe des Scripts auf der Karte dargestellt werden kann. Bei der Wahlbezirkstabelle war zu beachten, dass sich in der Stadt Moers die Wahlbezirke im Laufe der Zeit verändert haben, d.h., dass die Abmessung mancher Bezirke oder die Nummern einzelner Bezirke verändert wurden. Um mit den Geokoordinaten nicht zu kollidieren, war unbedingt darauf zu achten, dass die Bezirke auch denen des darzustellenden Wahljahres entsprachen. So gibt es in der Datenbank sechs verschiedene Bezirkstabellen. Jede der Tabellen enthält die passenden Bezirke für das dargestellte Wahljahr. So muss also bei jeder neuen Wahl überprüft werden, ob die Bezirke aktuell sind und daraufhin eine neue Tabelle erstellt werden.

Die letzte Tabellenart enthält die Bevölkerungsdaten der Stadt Moers. Sie ist eingeteilt in Ratswahlbezirke. Dies wurde vorgenommen, da sonst die Körnung[42] zu detailreich wäre und gegebenenfalls Rückschlüsse auf einzelne Wähler gezogen werden könnte. In diesem Fall würde die Visualisierung gegen die Vorschritten zum Datenschutz verstoßen.

[41] Mit Polygonen sind hier Dreiecksnetzte gemeint, welche sich gut zur schnellen Darstellung von Oberflächen auf einem Computer nutzen lassen.
[42] Detailstufe innerhalb der Bezirke.

RATSWAHLBEZIRK	m_0_bis_18	m_18_bis_25	m_25_bis_35	m_35_bis_45	m_45_bis_60	m_60	w_0_bis_18	w_18_bis_25	w_25_bis_35	w_35_bis_45	w_45_bis_60	w_60	anwohner
110	310	165	230	271	584	612	324	161	235	258	628	825	4603
111	221	110	245	231	360	437	201	158	251	218	427	628	3487
112	247	124	247	239	482	585	202	125	230	266	492	789	4028
113	236	128	262	225	411	488	220	133	265	233	452	781	3834
114	303	191	255	228	451	438	320	178	269	220	481	538	3872
115	291	185	188	204	401	419	309	159	201	207	428	465	3457
116	302	179	254	246	510	521	292	173	210	232	524	627	4070

Abbildung 21: Tabelle mit den Bevölkerungsdaten (Teil1)

Im ersten Teil der Tabelle sieht man die Anwohner des jeweiligen Bezirkes unterteilt nach Altersgruppen und Geschlecht. Nach dem Auswerten der Ergebnisse der Unterrichtsbesuche aus den Jahrgangsstufen elf und zwölf sowie nach Rücksprache mit Schülerrinnen und Schülern der neunten Jahrgangsstufe entstand diese Tabelle mit Bevölkerungsdaten. So lassen sich mit Hilfe dieser Tabelle im Unterricht z.B. Rückschlüsse auf das Wahlverhalten ziehen oder andere interessante Ereignisse erkennen. Vor allem die Einteilung nach Bezirken ist für den Schulunterricht sehr interessant. Auf der Website wird die Tabelle in Bezirke aufgeteilt und durch Klicken ein- oder ausgeblendet. Die Einteilung der Altersgruppen wurde mit Hilfe von Hr. Dr. Stender erstellt und entspricht damit der gängigen Einteilung bei Bevölkerungsdaten.

auslaender_0_bis_18	auslaender_18_bis_25	auslaender_25_bis_35	auslaender_35_bis_45	auslaender_45_bis_60	auslaender_60	migrant_0_bis_18
18	17	30	55	43	31	144
35	44	77	74	73	60	159
24	22	61	73	74	72	160
67	40	67	72	73	41	195
47	64	85	119	119	134	283
72	37	78	89	93	112	295
50	49	84	102	114	71	230
19	19	44	54	59	41	125

Abbildung 22: Tabelle mit Bevölkerungsdaten (Teil2)

Der zweite Teil der Tabelle enthält Daten zum Ausländer- und Migrantenanteil der jeweiligen Ratswahlbezirke. Die Einteilung der Altersgruppen entspricht der vorherigen. In der Tabelle wird zwischen Ausländer und Migranten unterteilt, allerdings nicht nach Geschlecht. Diese Einteilung ist wichtig, da Ausländer ohne deutschen Pass kein Wahlrecht haben und somit nicht an der Wahl teilnehmen dürfen. Migranten hingegen verfügen über einen deutschen Pass und sind wahlberechtigt und deshalb auch in der Statistik erfasst. Dies ist vor allem bei dem Vergleich der Wahlberechtigten und gültigen Stimmen sowie bei der Wahlbeteiligung der einzelnen Bezirke zu beachten, da es sonst zu Abweichungen kommen kann. Die von den Studierenden verarbeiteten Bevölkerungsdaten stammen aus dem Jahr 2013. Daher ist zu beachten, dass die soziale Struktur bei älteren Wahlen nicht zu 100% mit den Daten in der Tabelle übereinstimmen

wird. Um neue Bevölkerungsdaten anzulegen, wird lediglich eine Tabelle mit identischer Struktur benötigt.

Dadurch, dass das Script der Studierenden nicht mit festgelegten Werten, sondern mit Variablen arbeitet, ist es dynamisch. Somit wird es nachfolgenden Benutzern einfach gemacht, neue Daten in die vorhandene Karte einzubringen. Es werden nur CSV-Tabellen mit der beschriebenen Struktur benötigt. Der einzige Bearbeitungsaufwand im Script sind damit die zu ergänzenden Menüpunkte in der Navigation, welche sich aber aufgrund des Namens der Tabellen einfach erstellen lassen[43]. Diese Art wurde von den Studenten gewählt, da es auch anderen Städten, Schulen oder Studenten möglich sein soll das Script zu nutzen und neue Wahlen darzustellen.

3.2.3 Mögliche Weiterentwicklung der Datenbankstruktur

Im Nachgang zu der vorhandenen Arbeit oder auch als neues Projekt wäre es denkbar, die Datenbanken zu normalisieren. Das bedeutet, man legt Primary-[44] und Foreign[45] Keys fest. Mit Hilfe dessen erstellt man Verweise und Verbindungen der einzelnen Tabellen. Das hat den Vorteil, dass man z.B. nicht für jede Wahl eine eigene Tabelle braucht und das man mehrere Städte auf einer Karte zusammen darstellen kann. Des Weiteren dient es der Übersichtlichkeit, Wartbarkeit und Performance[46], denn MySQL ist darauf ausgelegt untereinander verknüpfte Tabellen abzurufen und zu bearbeiten. In dem momentanen Konstrukt könnte es irgendwann zu Problemen mit dem Abrufen der Tabellen kommen, da sich möglicherweise zu viele Tabellen in der Datenbank befinden.

Abschließend kann man sagen, dass die momentane Datenbankstruktur ausreicht, um die vorhandenen Wahlen darzustellen, ohne Performance-Probleme zu verursachen. Beim Weiterführen des Projekts sollte allerdings die oben angesprochene Normalisierung in Betracht gezogen werden, um ein flüssiges Zusammenspiel zwischen Datenbank und Webseite gewährleisten zu können und zwar auch dann noch, wenn mehrere Städte auf der Karte vertreten sind.

---

[43] Die Menüpunkte lassen sich innerhalb des Codes aus dem Aufbau der vorhandenen ableiten.
[44] Ein Primary Key dient zur eindeutigen Identifizierung einer Tabelle innerhalb eines MySQL-Datenbanksystems.
[45] Ein Foreign Key ist ein Verweis auf einen Primary Key einer anderen Tabelle.
[46] Geschwindigkeit der Tabellenverarbeitung.

3.2.4 Programmierung

Im folgenden Abschnitt wird auf die Funktionalität der Wahlergebnisplattform eingegangen. Dazu werden die wichtigsten Abschnitte des Quellcodes in Zusammenhang mit der Datenbankstruktur und dem Nutzererlebnis gestellt.

Am Ende des Praxissemesters bei der Stadt Moers, wurde ein vorläufiger Prototyp durch die Studierenden veröffentlicht. Durch die Kooperation mit Ernesto Ruge[47] ist dieser Prototyp unter der Subdomain http://edu.openruhr.de/ auf dem Open Ruhr Server zu erreichen.

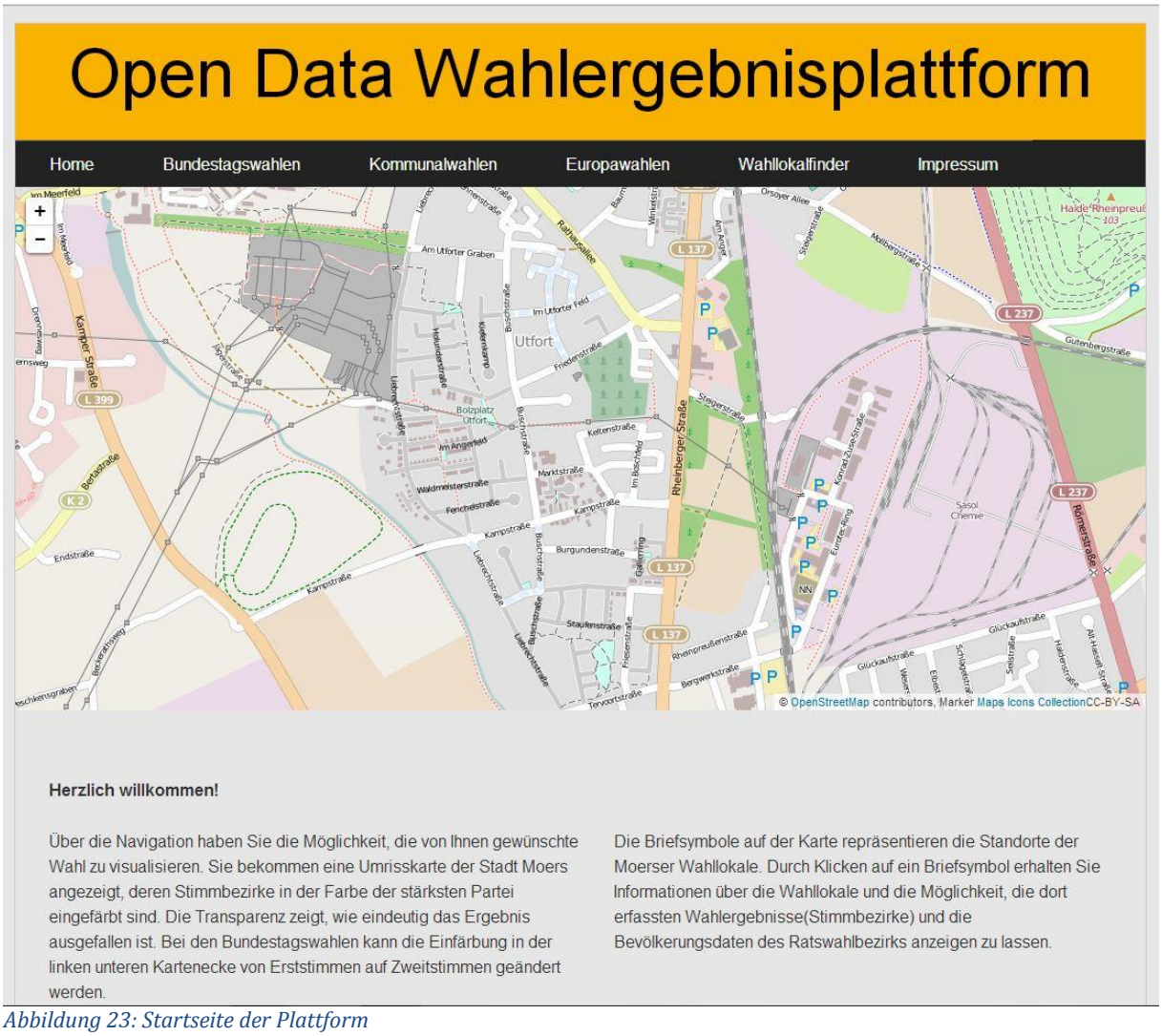


Abbildung 23: Startseite der Plattform

Der Besucher bekommt zunächst eine leere Karte angezeigt, die über dem Moerser Stadtgebiet zentriert ist. Unterhalb der Karte wird kurz beschrieben, wie die Plattform zu verwenden ist.

[47] Ernesto Ruge studiert an der TU Dortmund Physik, ist Geschäftsführer beim ruhrmobil-E e.V., einem Verein für Elektromobilität im Ruhrgebiet und freiberuflicher Web-Entwickler und Server-Betreiber mit einer Passion für Open Data. Er ist der redaktionelle und technische Ansprechpartner der Plattform http://openruhr.de Quelle: http://openruhr.de/author/infinity/ (Stand:15.8.14).

```
<!-- Map Style&Script -->
    <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.css" />
    <script src="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.js"></script>

    <script>
// create a map in the "map" div, set the view to a given place and zoom
    var map = L.map('map').setView([51.47211, 6.627610000000004], 15);

// add an OpenStreetMap tile layer
    L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
        }).addTo(map);

    map.attributionControl.setPrefix(false);
    map.attributionControl.addAttribution("Marker <a href=\"http://mapicons.nicolasmollet.com/\">Maps Icons Collection</a>CC-BY-SA");
    </script>
```

Abbildung 24: Codesegment Map Style&Script

Bei der Karte handelt es sich um eine Open Source JavaScript-Bibliothek namens Leaflet. Diese wird durch das oben gezeigte Codesegment auf der Plattform eingebunden. In den Zeilen 131 und 132 wird das für die Darstellung nötige Stylesheet und das für die Funktionalität nötige JavaScript von der Leaflet-Webseite verlinkt. Um die Karte sichtbar zu machen, wird in Zeile 136 eine JavaScript-Variabel deklariert, auf der Leaflet-eigene JavaScript Objekte genutzt werden, um eine neue Karte zu erstellen. Das vordefinierte Objekt „L.map()“ erwartet den Parameter „map“, um einen neuen Kartenlayer zu erstellen. Durch die „.setView()“-Einstellungen, wird die Karte mit dem Zoomfaktor 15 über den Koordinaten 51.47211, 6.627610000000004 (Moers-Utfort) zentriert. Über die Funktion in den Zeilen 139-141 wird dieser Karte das Copyright-Attribut hinzugefügt.

Abbildung 24: Navigation

Über die Navigation kann der Besucher die gewünschten Wahlergebnisse aufrufen. Durch einen Mouse-Over über den Schaltflächen Bundestagswahl, Kommunalwahl und Europawahl, werden die Wahljahre chronologisch, beginnend mit der aktuellsten Wahl angezeigt. Bei der Bundestags- und bei der Europawahl genügt ein Klick auf die Jahreszahl um zur Visualisierung zu gelangen. Bei der Kommunalwahl ist bei einem Mouse-Over über die Jahreszahl eine detaillierte Auswahl nötig, da hier Wahlergebnisse zum Bürgermeister, zum Stadtrat, zum Kreistag und zum Landrat auswählbar sind. Dabei ist zu beachten, dass nicht alle Wahlen im selben Zyklus stattfinden und daher unter manchen Jahreszahlen nur die Wahl zum Bürgermeister und zum Stadtrat zu finden sind.

```
        <li>
            <a href="#">Kommunalwahlen</a>
            <ul>
                <li>
                    <a href="#">2014</a>
                    <ul>
                        <li><a href="index.php?wahl=Kommunalwahl_Buergermeister_2014">Bürgermeister</a></li>
                        <li><a href="index.php?wahl=Kommunalwahl_Stadtrat_2014">Stadtrat</a></li>
                    </ul>
                </li>
```

Abbildung 25: Auszug aus dem HTML-Code der Navigation

Der Auszug in *Abbildung 25* zeigt die HTML-Listenelemente für die Kommunalwahlen im Jahr 2014. Klickt der Besucher auf die Fläche Kommunalwahl, oder auf die Jahreszahl

2014 im Untermenü, wird durch „href="#"“ nur die aktuell aufgerufene Seite neu geladen. Bei den Listenelementen Bürgermeister und Stadtrat wird der URL[48] jedoch ein Parameter angehängt und eine neue Seite geladen, wie die folgende Abbildung verdeutlicht.

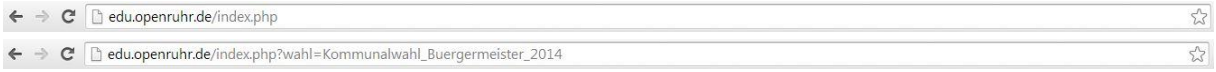


Abbildung 26: URL-Parameter

Der URL-Parameter „wahl“ ist nicht nur für die eindeutige Zuordnung der Unterseitenlinks wichtig, sondern wird auch dazu verwendet den Inhalt des Header-Elements zu verändern und die Wahlergebnisse aus der richtigen Tabelle der Datenbank abzurufen, wie später genauer erläutert wird. Dazu wird der Parameter auf einer PHP[49]-Variabel abgelegt.

```
156     $parameter = $_GET['wahl'];
157     $stimme = $_GET['stimme'];
158
159     if ($stimme == "") {
160         $stimme = "e";
161         }
162
163     if ($parameter ==""){
164         echo"var parameter = ' Open Data Wahlergebnisplattform  ';";
165         }
166     else {
167         echo"var parameter = '" . $parameter ."';";
168         echo"var  stimme = '" . $stimme ."';";
```

Abbildung 27: Parameter-Variabel

Der Inhalt des Parameters wird in Zeile 156 aus dem PHP-Get-Array[50] ausgelesen und auf der PHP-Variabel „$parameter“ abgelegt. Der Parameter bzw. die Variabel „$stimme“ wird für die Darstellung der Bundestagswahlen benötigt und ist daher bei Kommunalwahlen und Europawahlen standartmäßig leer. Mit der Bedingung in Zeile 159 wird der Parameter in diesen Fällen mit dem Inhalt „e“ belegt, um saubere Datenbankabfragen zu gewährleisten. Wozu dieser Parameter nötig ist, wird zu einem späteren Zeitpunkt genauer erläutert. Befindet sich der Besucher der Plattform auf der Startseite, ist die Variabel „$parameter“ ebenfalls leer und wird mit dem Inhalt „Open Data Wahlergebnisplattform“ belegt.

```
745     document.getElementById('header').innerHTML='<h2> ' + parameter.replace(/_/g," ") + '</h2>';;
746     document.title = parameter;
```

Abbildung 28: Headerinhalt und Dokumententitel ändern

In Zeile 745 wird der Parameter als Inhalt in das DIV-Element „header“ geschrieben. Dabei werden die Unterstriche durch Leerzeichen ersetzt. In Zeile 746 wird der Parameter auch als Dokumententitel festgelegt. In der folgenden Abbildung werden die Verän-

[48] Uniform Resource Locator. Identifiziert eine Ressource (z.B. eine Webseite) über die Zugriffsmethode und den Ort.
[49] Siehe Fußnote Nr.26.
[50] Deutsch: Feld. Eine Datenstruktur zu Verwendung vieler gleichartiger Datentypen.

derung auf der Plattform verdeutlicht die eintreten, wenn der Besucher von der Startseite auf die Kommunalwahl 2014 (Bürgermeister) wechselt.

Abbildung 29: Änderung des Dokumententitels und des Header-Inhalts

In den folgenden Abschnitten werden die Datenbankabfragen und die Darstellung der Wahlergebnisse beschrieben. Auf die Verbindung zur Datenbank wird nicht genauer eingegangen, da es sich nur um eine Testumgebung handelt. Der gesamte Quelltext ist im *Anhang 3* nach zu lesen.

```
175 //Abfrage der MySQL-Tabelle "Parteifarben"
176         $abfrage = "SELECT * FROM Parteifarben";
177         $ergebnis = mysqli_query($db, $abfrage);
178
179 //In der Tabelle vorhandene Parteien und die zugehörigen Farben in einem Array "$Parteifarbe" speichern
180         while($row = mysqli_fetch_object($ergebnis)){
181            $partei = $row->PARTEI;
182            $farben = $row->FARBE;
183            $wahl = $row->WAHLLOKAL;
184            $anzeigename = $row->Parteiname;
185
186            $parteifarbe[$partei] = $farben;
187            $parteiname[] = $anzeigename ;
188
189            $parteiindex = array_keys ($parteifarbe); //Auf diesem Array werden die Namen der Parteien (Indizes) abgelegt
190
191            } //while zu
192
193         $arraylaenge = count($parteiindex); //Array-Länge auf "$Arraylaenge" speichern um festzustellen, wie viele Partei
```

Abbildung 30: Abfrage der Tabelle „Parteifarben“

Zunächst wird der gesamte Inhalt der unter Punkt *3.2.2 Datenbanken* beschriebenen Tabelle „Parteifarben“ abgefragt. In der While-Schleife[51] werden die Inhalte der Spalten „PARTEI“, „FARBE“ und „Parteiname“ auf ihrem Namen entsprechenden PHP[52]-Variablen abgelegt. Darauf folgt die Erstellung der beiden Arrays[53] „$parteifarbe[]“ und „$parteiname[]“. Das Array „$parteifarbe[]“ erhält bei jedem Schleifendurchlauf den Inhalt der Spalte „PARTEI“ als Index. Der Index wird später dazu verwendet um die richtigen Spalten aus den Wahlergebnistabellen anzusprechen. Außerdem wird passend zum Index der Hexadezimal-Farbcode aus der Spalte „FARBE“ in das Array gelegt. Im zweiten Array „$parteiname[]“ werden die Anzeigenamen aus der Spalte „Parteiname“ gesammelt, die später auf der Plattform für die Besucher sichtbar werden. In Zeile 189 werden mit „array_keys“ alle Schlüssel des Arrays „$parteifarbe[]“ an die Variable „$parteiindex“

[51] Schleife innerhalb einer Programmiersprache, welche solange durchlaufen wird, wie eine Bedingung zutrifft.
[52] Siehe Fußnote Nr.33.
[53] Siehe Fußnote Nr.50.

geliefert. Außerdem wird nach der While-Schleife in Zeile 193 die Länge der „array_key“-Liste auf der Variabel „$parteiindex“ ausgezählt und wiederrum auf der Variabel „$arraylaenge“ gespeichert. Wozu dieser Schritt nötig ist, wird ebenfalls zu einem späteren Zeitpunkt erläutert.

Im Folgenden wird auf die Abfrage der Wahlergebnisse auf Stimmbezirksebene, die Errechnung der Gesamtergebnisse, die Färbung der Geolayer auf der Karte und die Darstellung der Ergebnistabellen auf der Plattform eingegangen. Dabei ist zu beachten, dass die Tabellen der Bundestagswahlen Ergebnisse zu den Erststimmen und den Zweitstimmen beinhalten. Diese werden durch die Vorzeichen „E_“ für Erststimmen und „Z_“ für Zweitstimmen unterschieden. Bei den Bundestagswahlen muss daher für beide Vorzeichen ein Abfrageblock durchlaufen werden. Da diese Abfrageblöcke abgesehen vom Spaltenvorzeichen identisch sind und die Tabellen der Kommunal- und Europawahlen nur Spaltennamen mit dem Vorzeichen „E_“ enthalten, wird hier nur auf den Aufbau der Erststimmen-Abfrage eingegangen. Die Darstellung der Erst- und Zweitstimmen auf der Plattform wird jedoch genauer beschrieben.

```
195 //Abfrage aller Wahlergebniss aus der Tabelle Wahlergebnisse
196         $abfrage2 = "SELECT * FROM " . $parameter;
197         $ergebnis2 = mysqli_query($db, $abfrage2);
```

Abbildung 31: Abfrage der Wahlergebnistabelle

Wie bereits weiter oben beschrieben, wird der URL[54]-Parameter dazu genutzt, die richtige Tabelle in der Datenbank anzusprechen. Mit der Abfrage in Zeile 196 wird also die Tabelle angesprochen, deren Name dem Inhalt der Variabel „$parameter“ entspricht.

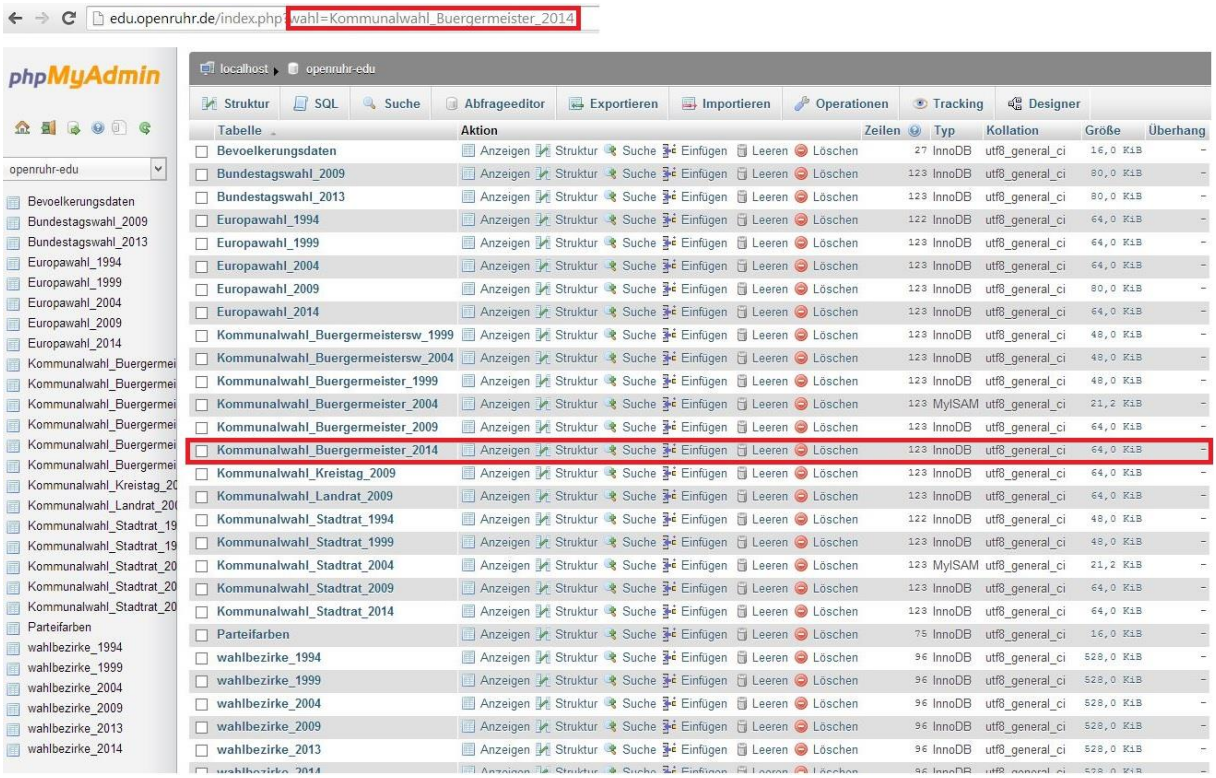


Abbildung 32: Zusammenhang zwischen Parameter und Tabellenname

In diesem Beispiel (*Abbildung 32*) befindet sich der Besucher auf der Unterseite der Bürgermeisterwahl im Jahr 2014, also wird die Tabelle „Kommunalwahl_Buergermeister_2014“ angesprochen.

```
201        while($row = mysqli_fetch_object($ergebnis2)){
202            $bezirk = $row->STIMMBEZIRK;
203            $wahl = $row->WAHLLOKAL;
204            $wahllokal = $row->WAHLLOKAL;
205            $wahllokal = str_replace($wahllokal, '"', '!');
206            $bezirk = (string) $bezirk;
207            $html = "";
208            $html2= "";
209            $max = 0;
210            $maxP = "";
211            $stimmen = intval($stimmen) + intval($row->Z_GUELTIG);
212            $stimmen2 = intval($stimmen2) + intval($row->E_GUELTIG);
```
Abbildung 33: While-Schleife

Auf die Abfrage folgt eine While-Schleife mit der das Gesamtergebnis der Wahl, die Wahlergebnisse für ein Wahllokal, die Geolayer und deren Einfärbung für die Erst- und Zweitstimmen ausgegeben werden. Für die einzelnen Schritte werden zu Beginn die in *Abbildung 33* zu sehenden Variablen deklariert.

[54] Siehe Fußnote Nr.48.

```
296 for ($i=0; $i < $arraylaenge; $i++){
297     $spaltenname = "E_" . $parteiindex[$i];
298     if(array_key_exists($spaltenname,$row)){
299         $absolut = $row->$spaltenname;
300         $gesamtab2[$parteiindex[$i]] = $gesamtab2[$parteiindex[$i]] + $absolut;
301
302         $spaltenname2 = "E_" . $parteiindex[$i] . "_%";
303         $relativ = $row->$spaltenname2;
304
305         if($absolut !=0){
306             $html2 .= "<tr><td>" . $parteiname[$i] . "</td><td align='right'>" . $absolut . "</td><td align='right'>" . $relativ. "%</td>
307             <td><div style = 'position: relative; height: 15px; width: " . $relativ . "px; background-color:" . $parteifarbe[$parteiindex[$i]] . ";' ></div></td></tr>";
308             }
309
310         if ($i==0){
311             $max = $relativ;
312             $maxP = $parteiindex[$i];
313             }
314         else {
315             if($relativ>$max){
316                 $maxP = $parteiindex[$i];
317                 $max = $relativ;
318                 }
319             }//else
320         }//if(array)
321     }// for zu
```

Abbildung 34: For-Schleife

In *Abbildung 30* wurde auf der Variabel „$arraylaenge“ gespeichert, wie viele Indizes aus der Tabelle „Parteifarben“ in das „$parteifarbe[]“-Array eingetragen wurden. Die in der obigen *Abbildung 34* gezeigte For-Schleife[55] läuft also solange durch, wie Indizes vorhanden sind. Damit ist sichergestellt, dass die Wahlergebnistabellen auf Spalten von allen Parteien durchsucht werden, die in der Tabelle „Parteifarben“ eingetragen wurden. Das in *Abbildung 30* erstellte Array „$parteiindex“ wird mit dem Index „$i“ – der sich bei jedem Schleifendurchlauf um den Wert „1“ erhöht – dazu genutzt eine nachhaltige Abfrage der Ergebnisspalten zu erreichen. Statt zum Beispiel mit „$spaltenname = „E_CDU“;“ und „$spaltenname2 = „E_CDU_%“;“ werden die Spalten wie in *Abbildung 34* gezeigt, über das Array „$parteiindex“ angesprochen.

Zur Erinnerung wird auf *Abbildung 19* verwiesen, die einen Ausschnitt der Tabelle „Parteifarben“ zeigt. Die Inhalte der Spalte „PARTEI“ dieser Tabelle wurden als Indizes auf dem Array gespeichert. Index „0“ ist in diesem Fall also die Partei CDU, Index „1“ die SPD und Index „2“ DIELINKE. Jede Partei in der Spalte stellt einen Index im Array dar. Beim ersten Schleifendurchlauf werden also „$spaltenname = „E_CDU“;“ und „$spaltenname2 = „E_CDU_%“;“ ausgeführt. Beim zweiten Durchlauf wird der nächste Index genutzt, also werden „$spaltenname = „E_SPD“;“ und „$spaltenname2 = „E_SPD_%“;“ ausgeführt. Dieser Vorgang wird wiederholt, bis alle Parteien abgearbeitet wurden.

In Zeile 300 wird ein Array erstellt, das bei jedem Durchlauf die absoluten Wahlergebnisse der Parteien addiert, um das Gesamtergebnis für das Stadtgebiet zu erhalten. In den Zeilen 305 bis 308 werden die Wahlergebnisse in eine HTML-Tabelle geschrieben, die später auf der Plattform ausgegeben wird. Jeder Schleifendurchlauf erzeugt eine neue Tabellenzeile. Diese Tabelle wird auf der PHP-Variabel „$html2“ gespeichert. Auf der Variabel „$html“ wurde zuvor die im Aufbau identische Tabelle der Zweitstimmen abgelegt.

CDU	19494	34.5%
SPD	26924	47.65%
Die Linke	3471	6.14%
Gruene	2388	4.23%
…	…	…

Tabelle 2: Struktur der HTML-Tabelle mit Beispielwerten

[55] Schleife innerhalb einer Programmiersprache, mit der man eine Gruppe von Anweisungen in einer bestimmten Anzahl Wiederholungen ausführen kann.

Von der Zeile 310 bis zur Zeile 318 wird Code ausgeführt, der anhand des relativen Ergebnisses einer Partei den Gewinner für einen Stimmbezirk ermittelt. Dieser Schritt ist für die spätere Einfärbung der Geolayer nötig.

```
$divHtml2[$bezirk] ="<table cellspacing='20' cellpadding='1'><tr><td><b>Partei</b></td><td><b>Stimmen</b></td>
<td><b>(%)</b></td><td></td></tr>" . $html2 . "</table>";

ruch Erststimmen
echo "\n";

kicken
if(strpos($wahl,"Briefwahl") === FALSE){
    echo "divHtml2[\"" . $bezirk . "\"] = \"" . $divHtml2[$bezirk] . "\";";
    }
```

Abbildung 35: Array „$divHtml2“

Im letzten Schritt der Wahlergebnis-Aufbereitung wird das Array „$divHtml2[]“ mit dem Index „$bezirk“ erstellt. Dieses Array trägt eine Tabellenstruktur mit drei Spalten deren Überschriften durch „Partei“, „Stimmen“ und „(%)“ definiert sind. Auf dem Index „$bezirk“ wird bei jedem Schleifendurchlauf die vierstellige Stimmbezirksnummer aus der Wahlergebnistabelle gespeichert. Die passenden Wahlergebnisse der Stimmbezirke wurden zuvor auf der Variabel „$html2“ zwischengelagert und nun in die Tabellenstruktur eingefügt. Auf der Plattform können die Wahlergebnisse aus den Tabellen zu jedem Stimmbezirk ausgegeben werden. Für die Zweitstimmen existiert ein weiteres Array namens „$divHtml[]“. Bei der gewählten Kartendarstellung der Wahlergebnisse ist zu berücksichtigen, dass es neben den Urnenwahlbezirken auch Briefwahlbezirke gibt. Die Briefwahlbezirke sind jedoch nicht wie die Urnenwahlbezirke an einen konkreten geographischen Punkt - dem Wahllokal -gebunden. Bisher wurde keine Möglichkeit ausgearbeitet, mit der die Ergebnisse der einzelnen Briefwahlbezirke auf der Plattform dargestellt werden können. Daher wird am Ende der obigen *Abbildung 35* abgefragt, bei welchen Stimmbezirken es sich um Briefwahlbezirke handelt. Nur wenn der zurück gegebene Wert „FALSE“ entspricht, es sich also nicht um einen Briefwahlbezirk handelt, wird der Bezirk mit Hilfe des „echo“-Befehls ausgegeben. Dabei ist unbedingt zu beachten, dass die in den Briefwahlbezirken erfassten Stimmen jedoch trotzdem in das Gesamtergebnis des Stadtgebiets einfließen und die Wahlergebnisse nicht verfälscht werden. Für die Ausgabe der Gesamtergebnisse wird ebenfalls ein Array erstellt, dessen Systematik dem beschriebenen Array „$divHtml2“ ähnelt. Wie die folgende *Abbildung 36* zeigt (Zeilen 397/398), wurden bei der Erstellung jedoch zusätzlich noch die Prozentanteile der absoluten Gesamtergebnisse manuell errechnet und in das Array aufgenommen.

```
394 for ($i=0; $i < count ($gesamtab2); $i++){
395
396     if($gesamtab2[$parteiindex[$i]]!=null){
397         $gesamtrel2 = $gesamtab2[$parteiindex[$i]] / $stimmen2 * 100;
398         $gesamtrel2 = round($gesamtrel2,2);
399         $ghtml2 .= "<tr><td>" . $parteiname[$i] . "</td><td align='right'>" . $gesamtab2[$parteiindex[$i]] . "</td>
400         <td align='right'>" . $gesamtrel2 . "%</td><td>
401         <div style = 'position: relative; height: 15px; width: " . $gesamtrel2 . "px; background-color:" . $parteifarbe[$par
402         }
403     }// for zu
404
405 $divgHtml2["gesamt"] ="<table cellspacing='20' cellpadding='1'><tr><td><b>Partei</b></td><td><b>Stimmen</b></td>
406 <td><b>(%)</b></td><td></td></tr>" . $ghtml2 . "</table>";
407 echo "divHtml2[\"" . "gesamt" . "\"] = \"" . $divgHtml2["gesamt"] . "\";";
408 echo "\n";
```

Abbildung 36: Array der Gesamtergebnisse

Mit diesem Schritt ist die reine Aufbereitung der Wahlergebnisse abgeschlossen. Nun folgt die Erstellung der Geolayer, mit denen die Stimmbezirke auf der Leaflet-Karte visualisiert werden können.

```
if($max>50){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 1.0};";
   }
elseif($max>45){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 0.85};";
   }
elseif($max>40){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 0.7};";
   }
elseif($max>35){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 0.55};";
   }
elseif($max>30){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 0.4};";
   }
elseif($max>25){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 0.25};";
   }
elseif($max>20){
   echo "var farbe = {\"color\": \"" . $parteifarbe[$maxP] . "\", \"weight\": 1, \"fillOpacity\": 0.1};";
   }
```

Abbildung 37: Farbe und Transparenz der Geolayer

Bevor die Polygone für die Stimmbezirke aus der Datenbank abgefragt werden, muss zunächst deren Farbe und Transparenz ermittelt werden. Auf der Variabel „$max“ wurde zuvor der Prozentwert des höchsten erzielten Ergebnis aus dem Stimmbezirk und auf der Variabel „$maxP“ der zugehörige Index aus dem Array „$parteiindex“, also der Name der Partei, die gewonnen hat, gespeichert. Der auf der Variabel „$max“ abgelegte Wert wird in der obigen *Abbildung 37* mit Abstufungen in fünfer Schritten verglichen. Entspricht der Wert einer der Bedingungen, wird eine neue Variable „farbe“ erstellt, die Eigenschaften für die Polygone trägt. Durch dieses Vorgehen wird die Transparenzeinstellung vorgenommen. Je niedriger der auf „$max“ abgelegte Prozentwert, desto niedriger ist die „fillOpacity“ der Polygone. Die Farbe wird mit Hilfe des Arrays „$parteifarbe“ und der Variabel „$maxP“ ermittelt. Die auf „$maxP“ abgelegte Partei wird dem Array „$parteifarbe“ als Index gesetzt. Gewinnt in einem Stimmbezirk zum Beispiel die CDU, wird der Inhalt – also die Farbe – dieses Indizes in das Attribut „color“ eingefügt. Wird der Code in diesem Fall ausgeführt steht dort also „color: #000000“.

Auf die Vorbereitung der Farb- und Transparenzeinstellungen folgt die Abfrage und Ausgabe der Polygone für die Stimmbezirk-Layer. Die Zuteilung der Haushalte zu bestimmten Stimmenbezirken unterliegt strengen Kriterien. So darf ein Stimmbezirk beispielsweise eine bestimmte Anzahl an Wahlberechtigten nicht unterschreiten. Die Stimmbezirke müssen daher gegebenenfalls für neue Wahlen angepasst werden. Deswegen wurden für die Plattform Wahlbezirk-Tabellen für alle verarbeiteten Wahljahre von 1994 bis 2014 angelegt.

```
$abfrage3 = "SELECT * FROM wahlbezirke_" . substr($parameter,-4) . " WHERE BEZIRK=\"" . $bezirk . "\"";
$ergebnis3 = mysqli_query($db, $abfrage3);

while($row2 = mysqli_fetch_object($ergebnis3)){
   $bezirksname = $row2->BEZIRK;
   $koordinaten = $row2->POLYGONE;
   $wahl = $row->WAHLLOKAL;

der Wahlbezirke mit den Koordinaten aus der Datenbank
   echo "var layer = {\"type\": \"FeatureCollection\",\"features\": [{ \"type\": \"Feature\", \"properties\":
   { \"BEZIRKSNUM\": \"" . $bezirksname . "\" }, \"geometry\":{\"type\": \"Polygon\",
   \"coordinates\": " . $koordinaten . "}}]};";

   }

der Map hinzufügen
if($stimme =="e"){
   echo "var myLayer = L.geoJson(layer,{style: farbe}).addTo(map);";
   echo "\n";
   }
```

Abbildung 38: Abfrage und Ausgabe der Polygone

Bei der Abfrage der richtigen Wahlbezirk-Tabelle wird erneut der URL-Parameter verwendet. In der Abfrage werden die letzten vier Zeichen des Parameter extrahiert und als Jahreszahl an den Tabellennamen „wahlbezirke_“ angehängt. Anschließend werden aus der Tabelle die Bezirksnummer und die entsprechenden Polygone für die Layer abgefragt und mit Hilfe der JavaScript-Bibliothek von Leaflet in einer „Feature-Collection“ verarbeitet. Diese „Feature-Collection“ beinhaltet die Bezirksnummer und die Geokoordinaten für die Layer und wird auf der JavaScript-Variabel „layer“ abgelegt. Um die einzelnen Layer mit den vorbereiteten Style-Attributen zu verknüpfen und als Gesamtkonstrukt der Karte hinzuzufügen, wird in Zeile 374 auf der Variabel „myLayer“ ein „L.geoJson“-Objekt[56] erzeugt, auf dem die Layer mit der Style-Variabel „farbe“ verknüpft und über „addTo(map)“ der Karte hinzugefügt werden.
Wie bereits weiter oben in diesem Kapitel erwähnt, gibt es neben dem URL-Parameter „wahl“ ebenfalls einen URL-Parameter „stimme“, auf dem grundsätzlich der Wert „e“ für Erstimmen abgelegt wird. Dieser Parameter wird dazu genutzt, um dem Benutzer auf den Unterseiten der Bundestagswahl die Möglichkeit zu geben, die Zweitstimmen ebenfalls auf der Karte zu visualisieren.

Abbildung 39: Checkbox und Parameter „Stimme“

Auf den Unterseiten der Bundestagswahl wird in der linken unteren Kartenecke die in *Abbildung 39* zu sehende Checkbox eingeblendet. Wird „Erststimmen“ ausgewählt, wird der URL „stimme=e“ angehängt und bei Auswahl von „Zweitstimmen“ entsprechend „stimme=z“. Daraus ergibt sich zum einen ein eindeutiger Unterseitenlink und zum anderen die Möglichkeit unterschiedliche Layer auf der Karte zu laden. Da die Wahlergebnisse und damit auch die Sieger in den Stimmbezirken bei Erst- und Zweitstimmen unterschiedlich ausfallen können, ist es nötig für beide Stimmen Layer zur Verfügung zu stellen.

```
373  if($stimme =="e"){
374      echo "var myLayer = L.geoJson(layer,{style: farbe}).addTo(map);";
375      echo "\n";
376      }
```

```
290  if ($stimme =="z"){
291      echo "var myLayer1 = L.geoJson(layer1,{style: farbe}).addTo(map);";
292      echo "\n";
293      }
```
Abbildung 40: Ausgabe der Erst- oder Zweitstimmen selektieren

[56] Erzeugt ein neues GeoJSON-Layer, dass bearbeitet und der Karte hinzugefügt werden kann.

Der Layer für die Zweitstimmen wird im Abfrageblock der Zweitstimmen nach identischem Muster aufgebaut. Bei der Selektierung welches Layer ausgegeben wird, genügt also den Inhalt der Parameter-Variabel „$stimme“ abzufragen (Zeile 373 bzw. 290).

Die letzte fehlende Komponente für eine erfolgreiche Visualisierung der Wahlergebnisse sind die Wahllokale, die mit Hilfe der JavaScript-Bibliothek als Briefsymbole auf der Karte angezeigt werden.

```
529  echo "var LokalMarker = [";
530
531     $abfrage4 = "SELECT * FROM " . $parameter;
532     $ergebnis4 = mysqli_query($db, $abfrage4);
533
534     $out = "";
535
536     while($row = mysqli_fetch_object($ergebnis4)){
537        $wahl = $row->WAHLLOKAL;
538        $stimmbezirk = $row->STIMMBEZIRK;
539        $lang = $row->Long;
540        $breit = $row->Lat;
541
542        if(strpos($wahl,"Briefwahl") === FALSE){
543           if(!isset($geodaten[$lang . $breit])){
544              $geodaten[$lang . $breit] = $stimmbezirk;
545              }
546           else {
547              $geodaten[$lang . $breit] = $geodaten[$lang . $breit] ."-" . $stimmbezirk;
548              }
549           }// if zu
550        }// while zu
551
552     $abfrage5 = "SELECT * FROM " . $parameter;
553     $ergebnis5 = mysqli_query($db, $abfrage5);
554
555     while($row = mysqli_fetch_object($ergebnis5)){
556        $wahl = $row->WAHLLOKAL;
557        $straße = $row->STRASSE;
558        $stimmbezirk = $row->STIMMBEZIRK;
559        $hnummer = $row->HAUSNUMMER;
560        $lang = $row->Long;
561        $breit = $row->Lat;
562        $link = "http://www.moers.de";
563
564        if(strpos($wahl,"Briefwahl") === FALSE){
565
566           $out = $out . "{\"type\": \"Feature\",\"properties\": {\"popuptype\": \"wahllokal\",\"type\": \"wahllokal\", \"name\": \"". $wahl . "\",\"link\": \"" . $link ."\",
567           \"BEZIRKSNUMM\": \"" . $geodaten[$lang . $breit] . "\", \"straße\": \"" . $straße . "\", \"hausnummer\": \"" . $hnummer . "\", \"popupContent\": \"test\"},\"geometry\":
568           {\"type\": \"Point\",\"coordinates\": [" . $lang . "," .$breit . "]},";
569
570           }
571        }// while zu
```

Abbildung 41: Aufbau der „Lokal-Marker“

Wie auch bei den Wahlergebnissen wird der URL-Parameter „wahl“ dazu genutzt, die richtige Tabelle anzusprechen. Im ersten Schritt (Zeile 536 bis 550) werden die Längen- und Breitengrade sowie die zugehörigen Bezirksnummern aus der Tabelle in das Array „$geodaten[]“ geschrieben. Längen- und Breitengrad stellen dabei den Index für die jeweilige Bezirksnummer als Inhalt der Array-Schubladen dar. Im zweiten Schritt (Zeile 555 bis Zeile 571) werden alle in der Tabelle enthaltenen Informationen über die Wahllokale auf Variablen abgelegt. In Zeile 566 wird dann ein String mit einer „Feature-Collection“ aufgebaut, in der die Variablen an Attribute geknüpft werden. Hier wird das zuvor erstellte Array „$geodaten[]“ benötigt, um die Bezirksnummern mit deren Hilfe die richtigen Wahlergebnisse abgerufen werden können, eindeutig zuzuordnen. Am Ende der Abfrage wird dieser String ausgegeben.

```
659   function onEachFeature(feature, layer) {
660      coords = feature.geometry.coordinates
661      area.push([coords[1], coords[0]])
662      if(feature.properties.BEZIRKSNUMM.match("-")==null){
663         layer.bindPopup('<b>' + feature.properties.name + '</b></br>' + feature.properties.straße + ' ' + feature.properties.hausnummer + '</br></br>Stimmbezirk ' + feature.properti
664         } else {
665            var bezirkn = feature.properties.BEZIRKSNUMM;
666            bezirkn = bezirkn.toString();
667            var bezirk = bezirkn.split("-");
668            var output ="";
669            output ='<b>' + feature.properties.name + '</b></br>' + feature.properties.straße + ' ' + feature.properties.hausnummer;
670
671            for(var i=0; i<bezirk.length; i++) {
672               output = output + '</br></br>Stimmbezirk ' + bezirk[i] + ':</br><a id="myLink" title="Wahlergebnisse anzeigen" href="#anker" onclick="DivZeiger(' + bezirk[i] + ')">Wah
673               }
674
675            layer.bindPopup(output);
676         }
677      }// function ende
678
679  function pointToLayer(feature, latlng) {
680     var option = options[feature.properties.type];
681     return L.marker(latlng, option)
682     }
683
684  L.geoJson(LokalMarker, {
685     onEachFeature: onEachFeature,
686     pointToLayer: pointToLayer
687     }).addTo(map);
```

Abbildung 42: „onEachFeature“-Funktion und hinzufügen der Karte

In der *Abbildung 42* wird die „onEachFeature“-Funktion der Leaflet-Bibliothek dazu genutzt, die Pop-Ups der einzelnen Lokal-Marker mit Inhalt zu füllen. Auf die genauere Funktionalität soll hier nicht genauer eingegangen werden, da diese detailliert in der Leaflet-Dokumentation nachvollziehbar ist[57]. Den Inhalt der Pop-Ups stellen die zuvor im String verarbeiteten Informationen zu den Wahllokalen dar. In den Zeilen 684 bis 687 werden die Marker, wie auch zuvor die Bezirk-Layer auf einem „L.geoJson()“-Objekt der Karte hinzugefügt.

Abbildung 43: Kartendarstellung auf der Plattform

Die bisher beschriebenen Abfragen aus der Datenbank und Ausgaben auf die Leaflet-Karte führen zu der in *Abbildung 44* gezeigten Kartendarstellung. Die Stimmenbezirke wurden eingefärbt und als Geolayer über die Karte gelegt. Die Briefsymbole stellen die Wahllokale des Moerser Stadtgebiets dar. Klickt der Nutzer auf ein Briefsymbol, öffnet sich ein Pop-Up, das den Namen des Wahllokals sowie die Adresse beinhaltet. Außerdem befindet sich unter der Bezirksnummer der Link „Wahlergebnisse anzeigen“ auf dessen Funktion im Folgenden genauer eingegangen wird.

```
600  function DivZeiger(text1){
601     if (text1 == "gesamt"){
602        document.getElementById("ergebniscontainer").innerHTML = "<b>Gesamtergebnis Erststimmen"  + " :</b>"  + divHtml2[text1];
603        document.getElementById("ergebniscontainer2").innerHTML = "<b>Gesamtergebnis Zweitstimmen"  + " :</b>"  + divHtml[text1];
604        document.getElementById("ergebniscontainer").style.display = 'inline';
605        //Bevölkerungsdaten-Div mit Bevölkerungsdaten füllen
606        text1 = text1.toString().substring(0,3);
607        document.getElementById("bedaten").innerHTML = "<b>Bevölkerung im Moerser Stadtgebiet:</b> (Stand: 2013)"  + bedaten[text1];
608        } else {
609           document.getElementById("ergebniscontainer").innerHTML = "<b>Erststimmen in Stimmbezirk " + text1 + " :</b>"  + divHtml2[text1];
610           document.getElementById("ergebniscontainer2").innerHTML = "<b>Zweitstimmen in Stimmbezirk " + text1 + " :</b>"  + divHtml[text1];
611           document.getElementById("ergebniscontainer").style.display = 'inline';
612           //Bevölkerungsdaten-Div mit Bevölkerungsdaten füllen
613           text1 = text1.toString().substring(0,3);
614           document.getElementById("bedaten").innerHTML = "<b>Bevölkerung im Ratswahlbezirk " + text1 + " :</b> (Stand: 2013)"  + bedaten[text1];
615        }
```

Abbildung 44: DivZeiger-Funktion

[57] Siehe http://leafletjs.com/reference.html#layergroup (Abschnitt GeoJson (Stand:15.8.14)).

Die „DivZeiger“-Funktion nimmt den Parameter „text1“ entgegen. Die Funktion wird „onload“ aufgerufen, also wenn der Besucher die Plattform öffnet und kann durch Klicken auf den Link „Wahlergebnisse anzeigen“ in den Pop-Ups der Lokal-Marker erneut durch den Besucher aufgerufen werden. Unterhalb der Karte sollen die vorbereiteten Wahlergebnistabellen für das Gesamtergebnis und für die einzelnen Stimmbezirke angezeigt werden. Um den Besucher nicht mit zu vielen Tabellen zu überfordern, wird jedoch immer nur eine Tabelle bzw. jeweils eine für Erst- und eine für Zweitstimmen angezeigt. Im Fall des „onload“-Aufrufs der Funktion, ist der Inhalt des Parameters „text1=gesamt“, also werden den beiden DIV-Elementen „ergebniscontainer“ und „ergebniscontainer2“ die Arrays „divHtml2[gesamt]“ bzw. „divHtml[gesamt]“ - in denen die Tabellen für das Gesamtergebnis abgelegt wurden – hinzugefügt (Zeilen 601 bis 604). Auf der Plattform entsteht diese Ausgabe:

Gesamtergebnis Erststimmen :

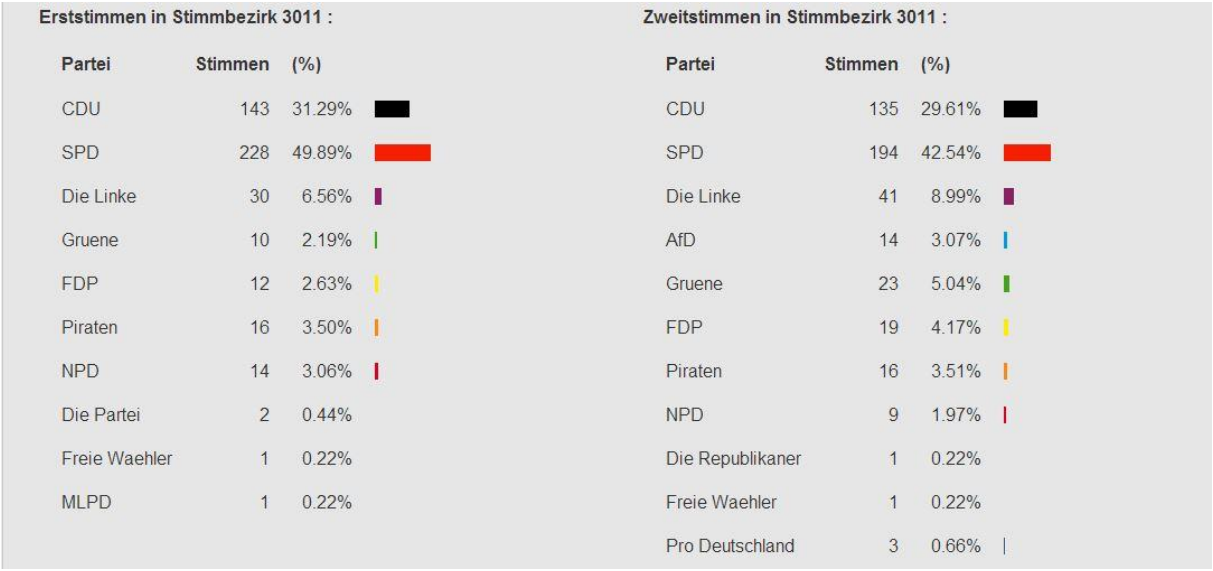
Partei	Stimmen	(%)
CDU	19494	34.5%
SPD	26924	47.65%
Die Linke	3471	6.14%
Gruene	2388	4.23%
FDP	1197	2.12%
Piraten	1296	2.29%
NPD	1148	2.03%

Gesamtergebnis Zweitstimmen :

Partei	Stimmen	(%)
CDU	18617	32.84%
SPD	22623	39.9%
Die Linke	4001	7.06%
AfD	2386	4.21%
Gruene	3673	6.48%
FDP	2265	3.99%
Piraten	1196	2.11%
NPD	808	1.43%
Die Republikaner	95	0.17%
Big	77	0.14%

Abbildung 45: DIV-Elemente mit den Gesamtergebnissen

In jedem anderen Fall (Zeilen 608 bis 611), also wenn der Besucher auf den Link „Wahlergebnisse anzeigen“ klickt und der Funktion als Parameter eine Bezirksnummer übergibt, werden die DIV-Elemente mit den passenden Tabellen für den Stimmbezirk gefüllt. Klickt der Nutzer also beispielsweise auf den in *Abbildung 43* zu sehenden Link, werden die Arrays „divHtml2[3011]“ und „divHtml[3011]“ mit den Tabellen für diesen Stimmbezirk in den DIV-Elementen angezeigt. Dadurch entsteht auf der Plattform diese Ausgabe:



Erststimmen in Stimmbezirk 3011 :

Partei	Stimmen	(%)
CDU	143	31.29%
SPD	228	49.89%
Die Linke	30	6.56%
Gruene	10	2.19%
FDP	12	2.63%
Piraten	16	3.50%
NPD	14	3.06%
Die Partei	2	0.44%
Freie Waehler	1	0.22%
MLPD	1	0.22%

Zweitstimmen in Stimmbezirk 3011 :

Partei	Stimmen	(%)
CDU	135	29.61%
SPD	194	42.54%
Die Linke	41	8.99%
AfD	14	3.07%
Gruene	23	5.04%
FDP	19	4.17%
Piraten	16	3.51%
NPD	9	1.97%
Die Republikaner	1	0.22%
Freie Waehler	1	0.22%
Pro Deutschland	3	0.66%

Abbildung 46: DIV-Elemente mit den Ergebnissen des Stimmbezirks 3011

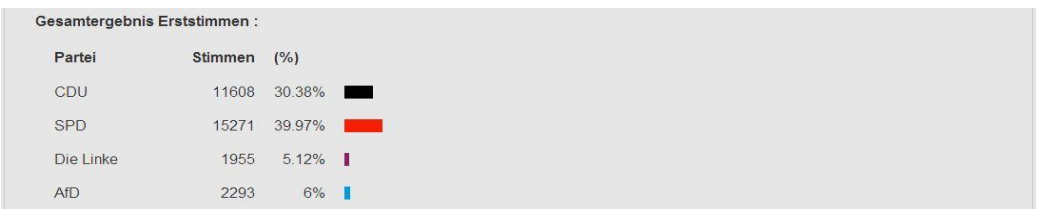
Die Funktion „DivZeiger“ wird außerdem dazu genutzt um die Bevölkerungsdaten auszugeben (Zeilen 606/607 bzw. 613/614). Darauf wird nach der Beschreibung der Aufbereitung der Bevölkerungsdaten genauer eingegangen.

```
594  var parameterlaenge = parameter.length;
595  var unterstrich = parameter.search ("_");
596  var kuerzung = parameterlaenge - unterstrich;
597  var parameterkurz = parameter.substring(0,parameter.length-kuerzung);
```

```
617  if (parameterkurz == "Kommunalwahl") {
618     document.getElementById("ergebniscontainer2").style.display = 'none';
619     } else if (parameterkurz == "Europawahl") {
620        document.getElementById("ergebniscontainer2").style.display = 'none';
621        }
622     }// function ende
```

Abbildung 47: Ausblenden des „ergebniscontainer2“

Da es nur bei den Bundestagswahlen Zweitstimmen gibt, ist der Inhalt des Arrays das dem „ergebniscontainer2“-Element hinzugefügt wird, nicht definiert. Um die Ausgabe „undefined“ im „ergebniscontainer2“ auf den Unterseiten der Kommunal- und Europawahlen zu verhindern, wird dieses DIV-Element dort ausgeblendet. Um die Unterseiten nachhaltig für den Code erkennbar zu machen, wurde der URL-Parameter in den Zeilen 594-597 so gekürzt, dass auf der Variabel „parameterkurz“ nur noch die Wahlart (Bundestagswahl, Kommunalwahl oder Europawahl) steht. Mit der Abfrage in den Zeilen 617-622 kann dadurch das DIV-Element in den Fällen „Kommunalwahl“ und „Europawahl“ ausgeblendet werden. Auf der Plattform ist nur das DIV-Element „ergebniscontainer“ zu sehen:



Gesamtergebnis Erststimmen :

Partei	Stimmen	(%)
CDU	11608	30.38%
SPD	15271	39.97%
Die Linke	1955	5.12%
AfD	2293	6%

Abbildung 48: Wahlergebnisansicht bei Europa- und Kommunalwahlen

```
406 //Datenbankabfrage Bevölkerungsdaten
407         $abfrage6 = "SELECT * FROM Bevoelkerungsdaten";
408         $ergebnis6 = mysqli_query($db, $abfrage6);
```

Abbildung 49: Abfrage der Bevölkerungsdatentabelle

Für die Abfrage der Bevölkerungsdaten ist der URL-Parameter zum Zeitpunkt der Prototypen-Veröffentlichung nicht nötig, da nur ein Datensatz vorhanden ist und dieser für alle verarbeiteten Wahljahre verwendet wird.

```
444 //While Schleife
445         while($row = mysqli_fetch_object($ergebnis6)){
446             $ratswahlbezirk = $row->RATSWAHLBEZIRK;
447             $ratswahlbezirk = (string)$ratswahlbezirk;
448             $anwohner = $row->anwohner;
449 
450             $ganwohner += $anwohner;
451 
452 //Altersgruppen männlich
453             $alterm1 = $row->m_0_bis_18;
454             $alterm2 = $row->m_18_bis_25;
455             $alterm3 = $row->m_25_bis_35;
456             $alterm4 = $row->m_35_bis_45;
457             $alterm5 = $row->m_45_bis_60;
458             $alterm6 = $row->m_60;
459 
460 //Altersgruppen männlich (Gesamt)
461             $galterm1 += $alterm1;
462             $galterm2 += $alterm2;
463             $galterm3 += $alterm3;
464             $galterm4 += $alterm4;
465             $galterm5 += $alterm5;
466             $galterm6 += $alterm6;
467 
468 //Altergruppen weiblich
469             $alterw1 = $row->w_0_bis_18;
470             $alterw2 = $row->w_18_bis_25;
471             $alterw3 = $row->w_25_bis_35;
472             $alterw4 = $row->w_35_bis_45;
473             $alterw5 = $row->w_45_bis_60;
474             $alterw6 = $row->w_60;
475 
476 //Altergruppen weiblich (Gesamt)
477             $galterw1 += $alterw1;
478             $galterw2 += $alterw2;
479             $galterw3 += $alterw3;
480             $galterw4 += $alterw4;
481             $galterw5 += $alterw5;
482             $galterw6 += $alterw6;
```

Abbildung 50: Auszug While-Schleife/Berechnung der Gesamtwerte

Während der in *Abbildung 50* gezeigten While-Schleife[58] werden die Bevölkerungsdaten auf Ratswahlbezirksebene sowie auf Stadtgebietsebene aufbereitet. Der Ausschnitt zeigt, wie für die sechs männlichen und sechs weiblichen Altersgruppe Variablen mit den Daten für die Ratswahlbezirke aus der Tabelle belegt werden. Außerdem wird für jede Gruppe eine Gesamtwert-Variabel erstellt, deren Inhalt sich bei jedem Schleifendurchlauf um den abgefragten Wert aus den Ratswahlbezirken erhöht. Dieses Vorgehen wird für die Daten zu Ausländern und Migranten wiederholt.

```
//Tabelle mit Daten für den Ratswahlbezirk füllen
        echo "bedaten[\"" . $ratswahlbezirk . "\"] = \"<p>Anwohner: " . $anwohner . "
        <a id='detailein' href='#anker' onclick=bedatenzeiger();>Details einblenden</a>
        <a id='detailaus' style='display: none' href='#anker' onclick=bedatenzeiger()[b];return false;[/b]>Details ausblenden</a>
        </p><table id='bevtabelle'><tr><th width='10%'><b>Altersgruppen</b></th><th width='10%'><b>Männlich</b></th><th width='10%'>
        <b>Weiblich</b></th><th width='10%'><b>Ausländer</b></th><th width='10%'><b>Migranten</b></th></tr><tr><td>0-18</td><td>" . $alterm1 . "</td>
        <td>" . $alterw1 . "</td><td>" . $auslaender1 . "</td><td>" . $migrant1 . "</td></tr><tr><td>18-25</td><td>" . $alterm2 . "</td><td>" . $alterw2 . "</td>
        <td>" . $auslaender2 . "</td><td>" . $migrant2 . "</td></tr><tr><td>25-35</td><td>" . $alterm3 . "</td><td>" . $alterw3 . "</td>
        <td>" . $auslaender3 . "</td><td>" . $migrant3 . "</td></tr><tr><td>35-45</td><td>" . $alterm4 . "</td><td>" . $alterw4 . "</td>
        <td>" . $auslaender4 . "</td><td>" . $migrant4 . "</td></tr><tr><td>45-60</td><td>" . $alterm5 . "</td><td>" . $alterw5 . "</td>
        <td>" . $auslaender5 . "</td><td>" . $migrant5 . "</td></tr><tr><td>060</td><td>" . $alterm6 . "</td><td>" . $alterw6 . "</td>
        <td>" . $auslaender6 . "</td><td>" . $migrant6 . "</td></tr></table>\";";

        echo "\n";

        }//while zu
```

Abbildung 51: Array „bedaten“ für einen Ratswahlbezirk

Ähnlich wie bei den Wahlergebnissen werden auch die Bevölkerungsdaten in einer Tabelle angeordnet und in einem Array einsortiert. Das Array „bedaten“ aus *Abbildung 52* trägt dabei die Ratswahlbezirke als Indizes, einen ein-/ausblende-Button und die Wertetabellen für die Ratswahlbezirke.

```
//Tabelle mit Daten für das gesamte Moerser Stadtgebiet füllen
        echo "bedaten[\"ges\"] = \"<p>Anwohner: " . $anwohner . " <a id='detailein' href='#anker' onclick=bedatenzeiger();>Details einblenden</a>
        <a id='detailaus' style='display: none' href='#anker' onclick=bedatenzeiger()[b];return false;[/b]>Details ausblenden</a></p><table id='bevtabelle'>
        <tr><th width='10%'><b>Altersgruppen</b></th><th width='10%'><b>Männlich</b></th><th width='10%'><b>Weiblich</b></th><th width='10%'><b>Ausländer</b></th>
        <th width='10%'><b>Migranten</b></th></tr><tr><td>0-18</td><td>" . $galterm1 . "</td><td>" . $galterw1 . "</td><td>" . $gauslaender1 . "</td>
        <td>" . $gmigrant1 . "</td></tr><tr><td>18-25</td><td>" . $galterm2 . "</td><td>" . $galterw2 . "</td><td>" . $gauslaender2 . "</td>
        <td>" . $gmigrant2 . "</td></tr><tr><td>25-35</td><td>" . $galterm3 . "</td><td>" . $galterw3 . "</td><td>" . $gauslaender3 . "</td>
        <td>" . $gmigrant3 . "</td></tr><tr><td>35-45</td><td>" . $galterm4 . "</td><td>" . $galterw4 . "</td><td>" . $gauslaender4 . "</td>
        <td>" . $gmigrant4 . "</td></tr><tr><td>45-60</td><td>" . $galterm5 . "</td><td>" . $galterw5 . "</td><td>" . $gauslaender5 . "</td>
        <td>" . $gmigrant5 . "</td></tr><tr><td>060</td><td>" . $galterm6 . "</td><td>" . $galterw6 . "</td><td>" . $gauslaender6 . "</td>
        <td>" . $gmigrant6 . "</td></tr></table>\";";
```

Abbildung 52: Array „bedaten“ mit den errechneten Gesamtwerten

Für die Gesamtwerte des Moerser Stadtgebiets erhält das Array „bedaten“ den Index „ges“ und trägt neben dem ein-/ausblende-Button die errechneten Gesamtwerte in einer identisch aufgebauten Tabelle.

Die Bevölkerungsdaten werden, wie bereits erwähnt, mit Hilfe der „DivZeiger“-Funktion aus *Abbildung 44* in die Tabelle geladen. Beim „onload“-Aufruf der Funktion wird das in *Abbildung 52* aufgebaute Array mit dem Index „ges“ auf der Plattform geladen. Beim Laden der Tabellen für den Ratswahlbezirk ist jedoch eine Besonderheit zu beachten. Die bei den Wahlergebnissen verwendeten Stimmbezirksnummern sind vierstellig, während die Ratswahlbezirksnummern nur durch dreistellige Ziffernblöcke definiert werden. Die letzte Zahl eines Stimmbezirks ist die Identifikationsnummer innerhalb eines Ratswahlbezirks. Das bedeutet, dass die ersten drei Ziffern eines Stimmbezirks mit den drei Ziffern des Ratswahlbezirks übereinstimmen. Der Stimmbezirk „3011“ aus *Abbildung 43* ist also der erste Stimmbezirk im Ratswahlbezirk „301“. Klickt der Besucher auf den Link „Wahlergebnisse anzeigen“ und ruft dabei die Funktion „DivZeiger“ auf, muss zum Laden der richtigen Bevölkerungsdaten aus dem Array „bedaten“ nur das letzte Zeichen des übergebenen Parameters „3011“ entfernt werden. So kann der passende Index, der durch die Nummer des Ratswahlbezirks definiert ist, aufgerufen werden.

[58] Siehe Fußnote Nr.51.

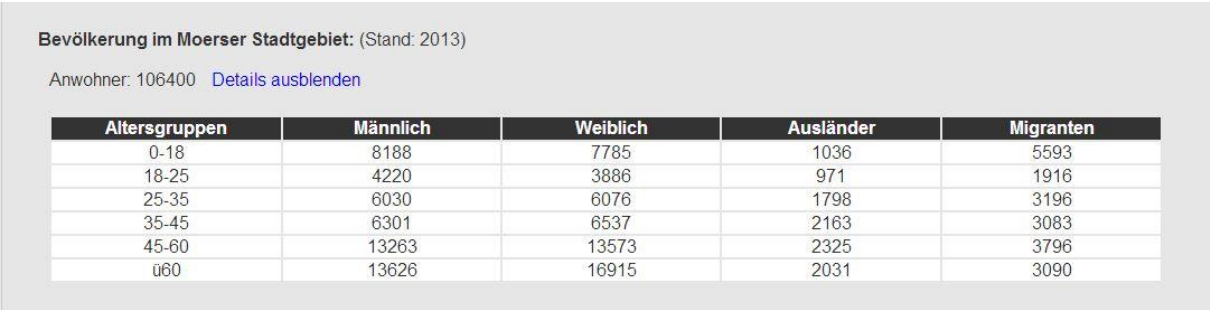
Abbildung 53: Ausgeblendete Bevölkerungsdaten

Damit der Besucher der Plattform sich nicht direkt zu Beginn mit zu vielen Daten konfrontiert sieht, bleibt die Tabelle mit den Altersgruppen zu männlichen und weiblichen Anwohnern sowie Ausländern und Migranten ausgeblendet. Lediglich die Anwohnerzahl des Stadtgebiets bzw. des gewählten Ratswahlbezirk wird angezeigt. Klickt der Besucher nun auf den Link „Details einblenden“, wird folgende Funktion ausgeführt:

```
634  function bedatenzeiger(){
635      if (document.getElementById("bevtabelle").style.display =='inline') {
636         document.getElementById("bevtabelle").style.display ='none';
637         document.getElementById("detailein").style.display = 'inline';
638         document.getElementById("detailaus").style.display = 'none';
639         } else {
640            document.getElementById("bevtabelle").style.display ='inline';
641            document.getElementById("detailaus").style.display = 'inline';
642            document.getElementById("detailein").style.display = 'none';
643            }
644         }// function ende
```

Abbildung 54: Funktion „bedatenzeiger“

Diese Funktion fragt ab, ob das TABLE-Element „bevtabelle“ bereits eingeblendet ist. Trifft dieser Fall zu, wird die Tabelle ausgeblendet und der Link von „Details ausblenden“ auf „Details einblenden“ geändert. Ist das Element „bevtabelle“ bei Aufruf der Funktion nicht eingeblendet, wird es sichtbar gemacht und der Link „Details einblenden“ durch den Link „Details ausblenden“ ersetzt. Bei eingeblendeten „bevtabelle“-Element bekommt der Besucher auf der Plattform folgendes ausgegeben:



Altersgruppen	Männlich	Weiblich	Ausländer	Migranten
0-18	8188	7785	1036	5593
18-25	4220	3886	971	1916
25-35	6030	6076	1798	3196
35-45	6301	6537	2163	3083
45-60	13263	13573	2325	3796
ü60	13626	16915	2031	3090

Abbildung 55: Eingeblendete Bevölkerungsdaten

In diesem Kapitel wurde auf die wichtigsten Code-Segmente eingegangen, die für die Visualisierung der Wahlergebnisse und die Weiterentwicklung der Plattform von Bedeutung sind. Auf die restlichen Codesegmente soll in dieser Dokumentation nicht genauer eingegangen werden, da deren Funktionalität nicht ausschlaggebend für die weiter Entwicklung des Projekts sind. Bei der Navigation griffen die Studierenden auf eine vorgefertigte responsive[59] Navi zurück, die unter http://osvaldas.info/drop-down-navigation-

[59] Reaktionsfähig d.h. die Webseite skaliert bei Veränderung der Bildschirmgröße mit und ist somit universell für jedes Endgerät eisetzbar.

responsive-and-touch-friendly zu finden ist. Das Buch „Responsive Webdesign“[60] legte den Grundstein für das restliche Seitenlayout. Alle angelegten-Codedateien sind in *Anhang 3* in lauffähiger Fassung zu finden.

[60] Siehe: http://responsive-webdesign-buch.de (Internetseite zu dem Buch Stand:15.8.14)).

Abbildungsverzeichnis

Tabellenverzeichnis