Jörg Wachner, 28666
Jonas Eismann, 28588

Kartographie & Geomatik
Fachsemester: 7

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Dokumentation der Projektarbeit

Praktikum Datenbanken und Informationssysteme

bei Prof. Dipl.-Math. Hans F. Kern

Stadtinformationssystem Brüssel

Wintersemester 11/12

Inhaltverzeichnis

1 Einleitung 3
2 Aufgabenstellung und Ziel der Studienarbeit 3
2.1 Aufgabenstellung 3
2.2 Ziel der Studienarbeit 6
3 Vorbereitung und Datenrecherche 6
3.1 Stadtauswahl 6
3.2 Beschaffung der Daten 6
3.3 Erstellen der Datenbank 18
3.4 Entity-Relationship-Modell 23
4 Realisierung des Informationssystem 24
4.1 Erstellung der HTML-Seiten 24
4.2 Erstellung eines Stylesheets mittels CSS 27
4.3 Grundlagen der Funktionalität 31
4.4 Kurzbeschreibung der Abfragen 32
4.5 Abfrage 10 im Detail 36
4.6 Abfrage 19 im Detail 41
5 Fazit 48
6 Bewertung der Arbeitsanleitung 48

Abbildungsverzeichnis

Abbildung 1: OpenStreetMap Datenbeispiel 7
Abbildung 2: Ausleseprogramm für OpenStreetMap Dateien 9
Abbildung 3: Entity-Relationship-Modell 23
Abbildung 4: Fehlerhafte (links) und korrekte (rechts) Darstellung einer Infobubble 32

1 Einleitung

Die folgende Ausarbeitung dient zur Erklärung und Arbeitsschritte und Beschreibung des Ausbaus des Projektes im Rahmen der Vorlesung im Praktikum Datenbanken und Informationssysteme Bei Prof. Dipl.-Math. Hans F. Kern. Im Folgenden wird die Schritte und Aufgaben im Einzelnen eingegangen, die nötig waren um die Studienarbeit erfolgreich realisieren zu können.

2 Aufgabenstellung und Ziel der Studienarbeit

2.1 Aufgabenstellung

PDB K6B42, K6D15, K5D42 und K6D62

Der Leistungsnachweis zu diesem Praktikum besteht für K6B42 (PO 1), K6D15 (PO 2) und K6D62 (PO 4) aus einer nicht benoteten, für K6B62 (PO 3) und K6D62 (PO 4) aus einer benoteten Studienarbeit, die in Form einer Projektarbeit zu erbringen ist. Das Projekt verfolgt einen integrativen Ansatz, indem die Inhalte vorangegangener Lehrveranstaltungen wie Datenbanken und Informationssysteme, Betriebssysteme, Programmieren, Integrale Kartographie und Mediendesign und Integration zur Lösung der Projektaufgaben herangezogen werden.

Als Grundlage dienen entweder insgesamt neu erhobene Daten oder es können alternativ Daten vorangegangener Projekte genutzt werden. Es handelt sich dabei um Verkehrsnetze größerer Städte mit zusätzlichen touristischen Informationen. Das Verkehrsnetz darf nicht sternförmig sein. Die Schrift des Landes, in dem die Stadt liegt, muss diakritischen Zeichen wie ä, ř, ç enthalten.

Aufgaben

--Führung eines Word-Dokuments "Tagesberichte", in dem Sie die Arbeit am Projekt unter Datumsangabe stichwortartig festhalten. Die zeitlich neueren Angaben stehen dichter am Anfang.

--Projektkonzeption mit Festlegung und Nachweis der erforderlichen Daten, Aufstellung des Datenkatalogs und des Entity-Relationship-Modells und Formulierung von Abfragen. Alternativ: Sichtung und Bewertung der vorhandenen Daten (Projektanalyse). Sie erhalten die Daten eines vorangegangenen Projekts und eine Mustervorlage für Ihr eigenes Projekt.

--Erfassung der Daten. Alternativ: Ergänzung und/oder Vervollständigung der vorhandenen Daten. Es müssen in beiden Fällen Daten mit geographischer Referenz in Form geographischer Koordinaten erfasst werden; z.B. Sehenswürdigkeiten, Religiöse Stätten, Preiszonen, Restaurants, Fahrpläne. Die Güte der Koordinaten muss geprüft werden.

--Erstellen einer SQL-Ladedatei. Laden der Daten in eine MySQL-Datenbank. Realisierung von 20 Abfragen an die Datenbank mit Datum-, Uhrzeit- und Entfernungsberechnungen, mit Gruppenfunktionen, mit Subsubselektionen, mit Inner-, Outer- und Auto-Joins, mit Union, Intersect und Minus. Die sql-Abfragen sollen in je einer Datei isoliert dargestellt werden. In den SELECTs dürfen dabei nur benutzerdefinierte Konstanten enthalten sein.

--Installation der Software. Sie brauchen Notepad++, Apache, MySQL und PHP, Perl, JSP oder ASP. Wenn Sie XAMPP installieren, stehen Ihnen Apache, MySQL, PHP und Perl zur Verfügung.

--Verwendung der Kenntnisse in HTML (XHTML strict!); insbesondere Formulare (Eingabefeld, Eingabebereich, Auswahlliste, Checkbox, Radiobutton, Datei-Upload-Feld, Klick-Button, Absenden- und Zurücksetzen-Button) und Cascading Style Sheets.

--Konzeption des Layouts für die Site mit CSS ohne Frames.

--Einsatz einer serverseitigen Skriptsprache, z. B PHP. Realisieren und dokumentieren Sie zwei sehr kleine (!) Beispiele Ihrer Wahl: 1) die Kommunikation zwischen Client und Server bei der Verwendung von Formularen, 2) wie 1) aber mit Abfrage an eine Datenbank.

--Client-Server-Realisierung der genannten 20 Abfragen an die Datenbank mit Validierung der css-Datei und aller erzeugten HTML-Seiten. Die eror_log-Datei und die Fehlerkonsole müssen leer sein.

--Verwendung der JavaScript-Kenntnisse zur interaktiven Darstellung von Punkt- und möglichst auch Linienobjekten mit einem Internet-Kartendienst (z.B. GoogleMaps, OSM).

--Dokumentation der Projektarbeit mit Zusammenfassung, Fazit und Bewertung der Arbeitsanleitung.

Anregung

Dieses Aufgabenblatt ist ein Original und als solches Bestandteil der Studienarbeit. Für die Dokumentation verwenden Sie bitte Word in möglichst professioneller Weise (Formatvorlagen, automatische Verzeichnisse, INCLUDES). Bei den einzelnen Abfragen sollen Sie dann auf die inhaltlichen und technischen Aspekte eingehen. Die Eingaben und Ausgaben sollen vollständig dokumentiert sein. Die Zuordnung von Ausgabe zu Eingabe soll leicht erkennbar sein. EDV-Ein- und Ausgaben sind in Courier wiederzugeben. Die Programmierbeispiele stellen Sie in Word bitte in den Farben wie bei Notepad++ dar.

Elegant ist es, wenn Sie dafür sorgen, dass Ihre Applikation leicht in eine andere Interface-Sprache gebracht werden kann. Dafür ersetzen Sie alle deutschen Text-Strings durch Variable und erstellen ein Dictionary mit den Zuordnungen der Variablen zu ihren deutschen Strings. Das Dictionary wird mit include eingebunden. Man kann es leicht in eine andere Sprache bringen und einbinden.

Fremdes Eigentum, wie Texte, Bilder und Graphiken, bitte kennzeichnen.

Bitte beachten Sie die Hinweise in der Arbeitsanleitung.

Für die Bewertung sind die Vollständigkeit der Dokumentation, die technische Komplexität, sowie die Gestaltung der Unterlagen wichtig (CD und Druckausgabe im Hochformat). Dokumentieren Sie differenziert Ihren Zeitbedarf. Bitte keine Klarsichthüllen verwenden.

Sie können in Gruppen von höchstens zwei Personen arbeiten. Jedes Gruppenmitglied gibt eine vollständige Dokumentation ab.

Bei der Abgabe der Studienarbeit in einem späteren Semester gilt die Aufgabenstellung des späteren Semesters.

Termine

Einführung	18.3.2011
Vorlage der Ladedatei	15.4.2011
Vorlage des Layouts der Seite	29.4.2011
Vorlage des Arbeitsberichts	29.4.2011
Vorführung am Rechner	ab 24.6.2011

Abgabe der Studienarbeit als Ausdruck und auf CD spätestens am 31.6.2011

Literatur

Stefan Hinz u. Michael Seeboerger-Weichselbaum: MySQL 5 GE-PACKT, Redline, 2006

Michael Kofler: MySQL 5 Einführung, Programmierung Referenz, Addison-Wesley, 2007

Stefan Mintert (Hg.): XHTML, CSS & Co, Addison-Wesley, 2003

Chuck Musciano u. Bill Kennedey: HTML-Das umfassende Referenzwerk, O´Reilly, 1997

Rasmus Lerdorf u. Kevin Tatroe: Programmieren mit PHP, O´Reilly, 2003

Rasmus Lerdorf: PHP kurz und gut, O´Reilly, 2003

David Sklar u. Adam Trachtenberg: PHP Kochbuch, O´Reilly, 2003

David Flanagan: JavaScript, O´Reilly, 1997

Michael Purvis, Jeffrey Sambells u. Cameron Turner: Google Maps Anwendungen mit PHP und AJAX, Redline, 2007

2.2 Ziel der Studienarbeit

Bei dieser Studienarbeit soll ein Informationssystem einer ausgewählten Stadt konzipiert und erstellt werden, welches Auskunft über den kompletten Schienenverkehr der Stadt gibt. Das Verkehrsnetz sollte dabei nicht sternförmig sein und wenn möglich Namen enthalten in denen diakritische Zeichen wie ä, é, ç vorkommen. Das Auskunftssystem besteht später aus einer selbsterstellten Datenbank, welche die erhobenen Daten über die Haltestellen oder Linien mit den dazugehörigen Zusatzinformationen wie Koordinaten und Fahrtzeiten enthält. Zusätzlich werden die Daten durch touristische Informationen, wie z.B. Sehenswürdigkeiten und POIs ergänzt. Das Ergebnis dieses Projekts soll eine vollwertige und grafisch ansprechende Webseite sein, die neben den sonstigen Informationen mindestens 20 Abfragen auf die erstellte Datenbank enthält.

3 Vorbereitung und Datenrecherche

3.1 Stadtauswahl

Zunächst muss eine Stadt für die Umsetzung des Projektes gewählt werden, welche noch in keiner vorherigen Studienarbeit verwendet wurde. Bei dieser Projektarbeit fiel die Wahl auf Brüssel, da diese Stadt kein sternförmiges Liniennetz besitzt und bei vielen Namen der Bahnstationen Sonderzeichen beinhalten (Bsp. Étangs Noirs oder Trône).

3.2 Beschaffung der Daten

Die Daten für die Realisierung des Projekts wurden aus dem Open-Source Produkt www.openstreetmap.org entnommen. OpenStreetMap ist ein freies Projekt, das für jeden frei nutzbare Geodaten sammelt. OSM Daten können dort im "Export" Tab mittels der Angabe von minimalen und maximalen Längen- und Breitengrade des gewünschten Gebietes und dem Format "OpenStreetMap XML Data" als XML Datei exportiert werden.

Es können per Datei maximal 10.000 Knotenpunkte heruntergeladen werden. Da das Stadtgebiet von Brüssel weit mehr als 10.000 Punkte hat, wurde das Stadtgebiet in 9 Teilgebiete unterteilt. Dabei wurde darauf geachtet das sich die

Gebietsgrenzen großzügig überschneiden um sicherzustellen das das komplette Stadtgebiet abgedeckt ist.

Die geographische Größe der einzelnen Teilgebiete ist recht variabel was durch die zum Teil stark abweichende Datendichte in OSM zurückzuführen ist.

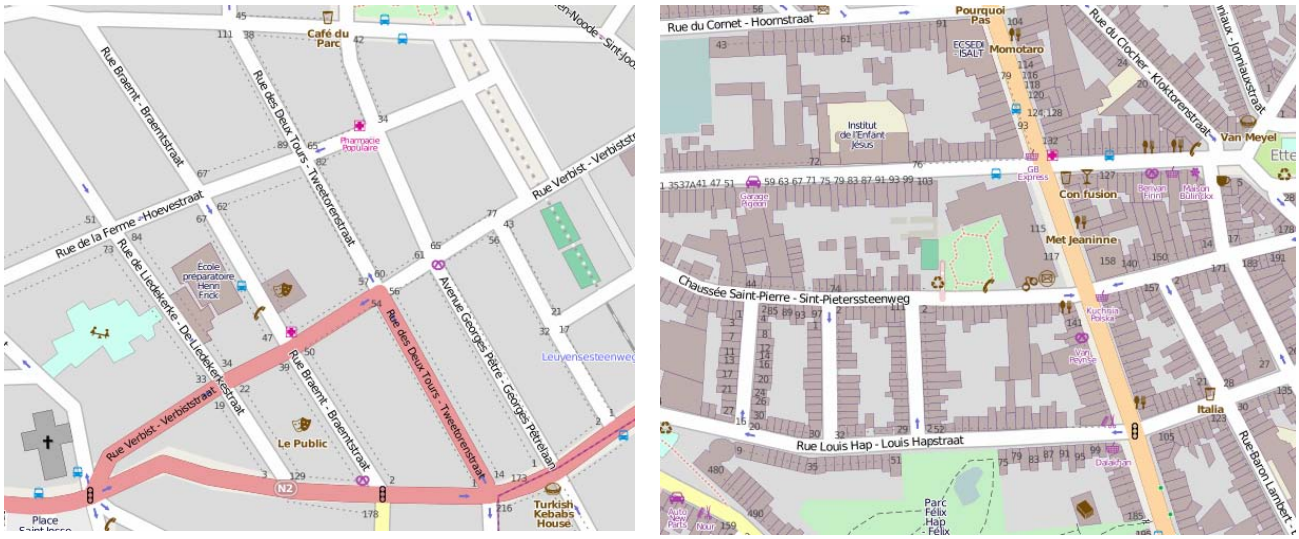

Abbildung 1: OpenStreetMap Daten im gleichen Maßstab von zwei ca. 1 km voneinander entfernten Stadtgebieten

Mit einem selbstentwickeltem Programm (OSMreader.exe) können die für dieses Projekt notwendigen Informationen ausgelesen werden. Die von OpenStreetMap exportierten Daten werden als OSM-Datei lokal gespeichert und mit Hilfe von OSMreader.exe werden die herausgelesenen Daten als Excel-Datei gespeichert. Da diese Daten sehr komplex (bis so 15000 Zeilen) sind werden hier nur Auszüge gezeigt, da dies sonst den Rahmen der Dokumentation sprengen würde.

Sekundär wurde die französische Wikipedia Seite der Brüsseler Straßenbahnen zur Festlegung der angefahrenen Stationen der einzelnen Linien verwendet (http://fr.wikipedia.org/wiki/Tramway_de_Bruxelles). Dies wurde notwendig da es sich als nicht praktikabel herausgestellt hatte diese aus den OSM Daten zu erfassen.
Zuletzt wurde Google Maps verwendet um die Lage einzelner, in OSM nicht vorhandener Stationen zu bestimmen. Dies war aber nur bei unter 5% der Gesamtanzahl aller Stationen nötig.

Auszug einer exportierten OSM-Datei:

```
<node id="503885987" lat="50.8768412" lon="4.3282397" user="moyogo"
```

```
uid="246" visible="true" version="1" changeset="2554872" timestamp="2009-
09-21T09:10:34Z">
  <tag k="name" v="Lenoir"/>
  <tag k="railway" v="tram_stop"/>
  <tag k="shelter" v="yes"/>
 </node>
<node id="503885987" lat="50.8768412" lon="4.3282397" user="moyogo"
uid="246" visible="true" version="1" changeset="2554872" timestamp="2009-
09-21T09:10:34Z">
  <tag k="name" v="Lenoir"/>
  <tag k="railway" v="tram_stop"/>
  <tag k="shelter" v="yes"/>
 </node>
<node id="867006879" lat="50.8721035" lon="4.3257498" user="moyogo"
uid="246" visible="true" version="1" changeset="5528348" timestamp="2010-
08-18T16:46:59Z">
  <tag k="addr:housenumber" v="590"/>
  <tag k="addr:street" v="ChaussÃ©e de Jette - Jetsesteenweg"/>
  <tag k="amenity" v="pub"/>
  <tag k="name" v="The Ascot"/>
 </node>
```

Wichtige Informationen sind hier:

- Breitengrad: lat="**50.8842091**"

- Längengrad: lon="**4.3378033**"

- OSM ID: uid="**246**"

- Stationsname: tag k="name" v="**Ernest Salu**"

- Typ: tag k="railway" v="**tram_stop**"

Optionale Informationen sind das Vorhandensein von Überdachungen (shelter) und Rollstuhlrampen (wheelchair), welche aber nur selten in den OSM Daten vorhanden sind.

Dies sind beispielhafte Auszüge der exportierten OSM-Datei, beinhaltet sind z.B. Informationen zu Straßen, Parks, Haltestellen, jegliche Arte von POIs, etc., da nur ein Teil der Daten verwendet werden und auch eine übersichtliche Excel Datei benötigt wird, wurde ein eigenes Programm verfasst, welches die wichtigen Daten (Koordinaten, Name der Haltestelle/POI, etc.) ausliest und als Excel Datei speichert (OSMreader.exe).

Zum Auslesen der Daten wurde ein Programm in der Programmiersprache C# erstellt. C# (welches von der Syntax sehr ähnlich wie Java ist) wurde verwendet

da man im Praxissemester mit diesen (und seiner Entwicklungsumgebung Microsoft Visual Studio) sehr viel Erfahrung gesammelt hatte.

Das Programm hat den Namen OSMreader.exe und befindet sich im Verzeichnis "\www\OSMreader", sein kommentierter Sourcecode im Verzeichnis "\www\OSMreader\Sourcecode".

Die .sln Datei ist die Microsoft Visual Studio Projektdatei. Ist dieses nicht vorhanden kann der Programmcode auch mit Notepad++ geöffnet werden.

Die Programmlogik befindet sich in der ein Verzeichnis tiefer gelegenen Form1.cs Datei. In Node.cs wird ein in Form1.cs verwendetes Objekt definiert. Die restlichen Dateien sind von Visual Studio automatisch generierte Bausteine des Formulars.

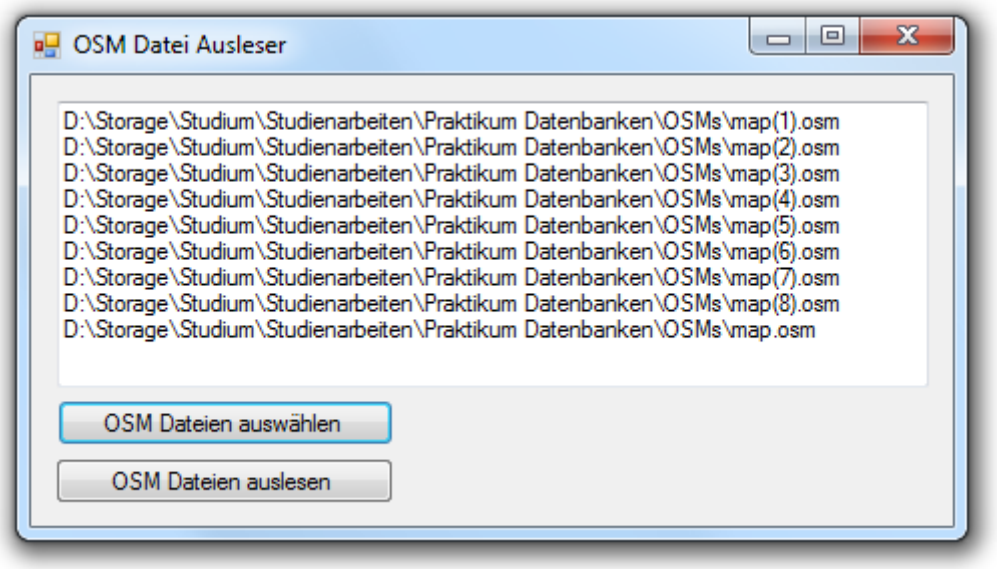

Abbildung 2: Ausleseprogramm für OpenStreetMap Dateien

Bei Klick auf das obere "OSM Dateien auswählen" Button öffnet sich ein Auswahlfenster in dem man die auszulesenden OSM Dateien markieren kann (mit gedrückter Shift Taste eine Mehrfachauswahl tätigen).

Diese werden nach dem Drücken auf "Öffnen" wie im Bild gezeigt im Textfenster angezeigt.

Bei Klick auf das untere "OSM Dateien auswählen" Button werden die ausgewählten OSM Dateien ausgelesen und eine Excel Datei für die Haltestellen und eine für die POIs erstellt.

Dies kann etwas dauern, der Button wird daher während die Erstellung läuft deaktiviert und sein Text zu "Excel Dateien werden erstellt" geändert. Mit den Daten von Brüssel und dem heimischen Desktop beträgt die Erstellungszeit etwa 30 Sekunden, bei schwächeren Rechnern kann dies aber durchaus länger dauern.

Die Excel Dateien werden im Verzeichnis der OSM Dateien erstellt. Daher können keine OSM Dateien von einem nicht beschreibbaren Medium (wie z.B. von einer CD) ausgelesen werden. Vor dem Auslesen müssen diese auf die Festplatte kopiert werden. Auch hat das Programm nur einfache Schreibrechte, daher dürfen die OSM Dateien nicht in Bereichen mit eingeschränkten Schreibrechten liegen.

Das Soll-Ergebnis des Auslesens sind zwei Excel Dateien. "Haltestellen" mit den Informationen der UBahneingänge sowie den Straßenbahn- und Bushaltestellen, "POIs" mit Informationen der Dienstleistungen und Geschäfte. Diese sind im Verzeichnis "\www\OSMreader\Ausgabedateien" abgelegt.

OSMreader.exe:

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Diagnostics;
using Excel = Microsoft.Office.Interop.Excel;

namespace OSMreader
{
    public partial class Form1 : Form
    {

        //
############################################################################
        /// <summary>
        /// Globale Variablen
        /// </summary>
        //
############################################################################

        /* Pfad der Verzeichnisses aus dem zuletzt OSM Dateien geöffnet
wurden */
        private string sFileDirectoryPath = "";
```

```
        /* Liste des zu einem String konvertierten Inhalts der einzelnen
OSM Dateien */
        private List<string> listOSMxmls = new List<string>();

        /* Liste von "Node" Objekten (siehe Node.cs) */
        private List<Node> listNodes = new List<Node>();

        /* OSM ID eines Nodes für die Suche nach doppelten Einträgen */
        private int nID = 0;


        //
##############################################################################
        /// <summary>
        /// Konstruktor
        /// Initialisiert das Form
        /// </summary>
        //
##############################################################################
        public Form1()
        {
            InitializeComponent();
        }


        //
##############################################################################
        /// <summary>
        /// Ereignisbehandlung wenn das "OSM Dateien auswählen" Button
geklickt wird:
        /// Öffnet eine oder mehrere OSM Dateien
        /// Liest deren Inhalt aus und speichert ihn in listOSMxmls
        /// Speichert ihr Verzeichnis in sFileDirectoryPath
        /// Zeigt den Pfad und Namen der eingelesenen OSM Dateien im
Textfenster des Forms an
        /// </summary>
        //
##############################################################################
        private void bFileChooser_Click(object sender, EventArgs e)
        {
            OpenFileDialog Dialog = new OpenFileDialog();
            Dialog.Multiselect = true;
            Dialog.Filter = "OSM XML Daten|*.osm";
            Dialog.InitialDirectory = "C:";
            if(sFileDirectoryPath != "")
            {
                Dialog.InitialDirectory = sFileDirectoryPath;
            }
            Dialog.Title = "OSM Daten auswählen";
            if (Dialog.ShowDialog() == DialogResult.OK)
            {
                sFileDirectoryPath =
System.IO.Path.GetDirectoryName(Dialog.FileNames[0]);
                txtOSMfiles.Clear();
                foreach (string sPath in Dialog.FileNames)
                {
                    listOSMxmls.Add(System.IO.File.ReadAllText(sPath));
                    txtOSMfiles.Text += sPath + "\r\n";
                }
            }
        }
```

```
        //
############################################################################
        /// <summary>
        /// Ereignisbehandlung wenn das "OSM Dateien auslesen" Button
geklickt wird:
        /// Deaktiviert und ändert den Text des Buttons während Methoden
ausgeführt werden
        /// Ruft ReadOSM auf um aus den Strings in listOSMxmls "Node"
Objekte zu erzeugen
        /// Ruft SaveExcel auf um aus den "Node" Objekten Excel Datein zu
erzeugen
        /// </summary>
        //
############################################################################
        private void bReadOSM_Click(object sender, EventArgs e)
        {
            bReadOSM.Enabled = false;
            bReadOSM.Text = "Excel Dateien werden erstellt";
            ReadOSM();
            SaveExcel("Haltestellen");
            SaveExcel("POIs");
            bReadOSM.Enabled = true;
            bReadOSM.Text = "OSM Dateien auslesen";
        }


        //
############################################################################
        /// <summary>
        /// Liest die Strings der OSM Daten aus und erstellt aus deren
Inhalt "Node" Objekte (siehe Node.cs)
        /// Speichert diese als Liste in listNodes
        /// Entfernt doppelte Node Objekte in listNodes (durch
geographische Überlappung der OSM Dateien
        /// kommt es zu doppelten Einträgen)
        /// </summary>
        //
############################################################################
        private void ReadOSM()
        {
            try
            {
                foreach (string sOSMxml in listOSMxmls)
                {

                    string[] aSeparator = new string[] { "</node>\n <way
id=\"" };
                    string[] aOSMxml = sOSMxml.Split(aSeparator,
StringSplitOptions.None);

                    if (aOSMxml.Length == 1)
                    {
                        aSeparator = new string[] { "/>\n <way id=\"" };
                        aOSMxml = sOSMxml.Split(aSeparator,
StringSplitOptions.None);
                    }

                    aSeparator = new string[] {"<node id=\""};
                    string[] aNodes =
aOSMxml[0].Split(aSeparator,StringSplitOptions.None);

                    foreach (string sNode in aNodes)
```

```
                    {
                        if (sNode.Contains("tram_stop") ||
                            sNode.Contains("bus_stop") ||
                            sNode.Contains("subway_entrance") ||
                            sNode.Contains("amenity") ||
                            sNode.Contains("shop"))
                        {
                            Node objNode = new Node();

                            aSeparator = new string[] {"\" lat=\""};
                            string[] aSplit =
sNode.Split(aSeparator,StringSplitOptions.None);
                            objNode.nID = Convert.ToInt32(aSplit[0]);

                            aSeparator = new string[] {"\" lon=\""};
                            aSplit =
aSplit[1].Split(aSeparator,StringSplitOptions.None);
                            objNode.dLatitude =
Convert.ToDouble(aSplit[0].Replace('.',','));

                            aSeparator = new string[] {"\" user=\""};
                            aSplit =
aSplit[1].Split(aSeparator,StringSplitOptions.None);
                            objNode.dLongitude =
Convert.ToDouble(aSplit[0].Replace('.', ','));

                            if (sNode.Contains("tram_stop"))
                            {
                                objNode.sType = "Strassenbahnhaltestelle";
                            }

                            if (sNode.Contains("bus_stop"))
                            {
                                objNode.sType = "Bushaltestelle";
                            }

                            if (sNode.Contains("subway_entrance"))
                            {
                                objNode.sType = "Ubahneingang";
                            }

                            if (sNode.Contains("\"amenity\""))
                            {
                                objNode.sType = "POI - " + Splitter("<tag
k=\"amenity\" v=\"", aSplit);
                            }

                            if (sNode.Contains("<tag k=\"name\" v=\""))
                            {
                                objNode.sName = Splitter("<tag k=\"name\"
v=\"", aSplit);
                            }

                            if (sNode.Contains("\"shop\""))
                            {
                                objNode.sType = "POI - " + Splitter("<tag
k=\"shop\" v=\"", aSplit);
                            }

                            if (sNode.Contains("shelter\" v=\"no"))
                            {
                                objNode.sShelter = "Nein";
                            }
```

```
                            if (sNode.Contains("shelter\" v=\"yes"))
                            {
                                objNode.sShelter = "Ja";
                            }

                            if (sNode.Contains("wheelchair\" v=\"no"))
                            {
                                objNode.sWheelchair = "Nein";
                            }

                            if (sNode.Contains("wheelchair\" v=\"yes"))
                            {
                                objNode.sWheelchair = "Ja";
                            }

                            nID = objNode.nID;
                            listNodes.RemoveAll(FindID);
                            listNodes.Add(objNode);
                        }

                    }
                }
            }

            catch (Exception ex)
            {
                Debug.WriteLine(ex.Message);
            }

        }


        //
############################################################################
        /// <summary>
        /// Extrahiert den Inhalt eines XLM formatierten Strings
        /// </summary>
        //
############################################################################
        private string Splitter(string sSeperator, string[] aSplit)
        {
            string[] aSeparator = new string[] { sSeperator };
            string[] aSplitted = aSplit[1].Split(aSeparator,
StringSplitOptions.None);

            if (aSplitted[1].Contains("\"/>"))
            {
                aSeparator = new string[] { "\"/>" };
                aSplitted = aSplitted[1].Split(aSeparator,
StringSplitOptions.None);
            }

            return aSplitted[0];
        }


        //
############################################################################
        /// <summary>
        /// Durchsucht Nodes nach der in nID gespeicherten ID
        /// </summary>
        //
```

```
##############################################################################
        private bool FindID(Node node)
        {
            if (node.nID == nID)
            {
                return true;
            }
            {
                return false;
            }
        }


        //
##############################################################################
        /// <summary>
        /// Liest aus den Nodes, je nachdem ob sType "Haltestellen" oder
"POIs" ist, die entsprechenden
        /// Werte aus und speichert sie in einer Excel Datei gleichen
Namens im Verzeichnis der OSMs
        /// </summary>
        //
##############################################################################
        private void SaveExcel(string sType)
        {
            try
            {
                Excel.Application ExcelApp;
                Excel.Workbook ExcelWorkbook;
                Excel._Worksheet ExcelWorksheet;
                object misValue = System.Reflection.Missing.Value;
                ExcelApp = new Excel.Application();
                ExcelWorkbook = ExcelApp.Workbooks.Add(misValue);
                ExcelWorksheet =
(Excel.Worksheet)ExcelWorkbook.Worksheets.get_Item(1);

                String sFile = sFileDirectoryPath + "\\" + sType + ".xls";

                ExcelWorksheet.Cells[1, 1] = "ID";
                ExcelWorksheet.Columns["A", Type.Missing].ColumnWidth = 12;
                ExcelWorksheet.Cells[1, 2] = "Name";
                ExcelWorksheet.Columns["B", Type.Missing].ColumnWidth = 45;
                ExcelWorksheet.Cells[1, 3] = "Typ";
                ExcelWorksheet.Columns["C", Type.Missing].ColumnWidth = 23;
                ExcelWorksheet.Cells[1, 4] = "Breitengrad";
                ExcelWorksheet.Columns["D", Type.Missing].ColumnWidth = 11;
                ExcelWorksheet.Cells[1, 5] = "Längengrad";
                ExcelWorksheet.Columns["E", Type.Missing].ColumnWidth = 11;

                if (sType == "Haltestellen")
                {
                    ExcelWorksheet.Cells[1, 6] = "Überdachung";
                    ExcelWorksheet.Columns["F", Type.Missing].ColumnWidth =
12;
                    ExcelWorksheet.Cells[1, 7] = "Rollstuhlrampe";
                    ExcelWorksheet.Columns["G", Type.Missing].ColumnWidth =
14;
                }

                int nLine = 2;
                foreach (Node objNode in listNodes)
                {
                    if (sType == "Haltestellen" &&
```

```
                    (objNode.sType == "Strassenbahnhaltestelle" ||
                     objNode.sType == "Bushaltestelle" ||
                     objNode.sType == "Ubahneingang"))
                {
                    ExcelWorksheet.Cells[nLine, 1] = objNode.nID;
                    ExcelWorksheet.Cells[nLine, 2] = objNode.sName;
                    ExcelWorksheet.Cells[nLine, 3] = objNode.sType;
                    ExcelWorksheet.Cells[nLine, 4] = objNode.dLatitude;
                    ExcelWorksheet.Cells[nLine, 5] =
objNode.dLongitude;
                    ExcelWorksheet.Cells[nLine, 6] = objNode.sShelter;
                    ExcelWorksheet.Cells[nLine, 7] =
objNode.sWheelchair;

                    nLine++;
                }

                if (sType == "POIs" && objNode.sType.Contains("POI"))
                {
                    ExcelWorksheet.Cells[nLine, 1] = objNode.nID;
                    ExcelWorksheet.Cells[nLine, 2] = objNode.sName;
                    ExcelWorksheet.Cells[nLine, 3] = objNode.sType;
                    ExcelWorksheet.Cells[nLine, 4] = objNode.dLatitude;
                    ExcelWorksheet.Cells[nLine, 5] =
objNode.dLongitude;
                    nLine++;
                }
            }

            ExcelWorkbook.SaveAs(sFile,
Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue,
misValue,
                Excel.XlSaveAsAccessMode.xlExclusive, misValue,
misValue, misValue, misValue, misValue);
            ExcelWorkbook.Close(true, misValue, misValue);
            ExcelApp.Quit();

            releaseObject(ExcelWorksheet);
            releaseObject(ExcelWorkbook);
            releaseObject(ExcelApp);
        }
        catch (Exception ex)
        {
            Debug.WriteLine(ex.Message);
        }
    }


    //
############################################################################
    /// <summary>
    /// Entfernt die Excel Objekte nach dem Beenden der Erstellung
einer Excel Datei
    /// </summary>
    //
############################################################################
    private void releaseObject(object obj)
    {
        try
        {

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
            obj = null;
```

```
            }
            catch (Exception ex)
            {
                obj = null;
                MessageBox.Show("Exception Occured while releasing object "
+ ex.ToString());
            }
            finally
            {
                GC.Collect();
            }
        }

    }
}
```

Die daraus resultierenden Excel Dateien beinhaltet nun wichtige Informationen für das Projekt: Name der Haltestellen/POI, die dazugehörigen Koordinaten, Typ. Mit Typ ist bei den Straßenbahnen das Verkehrsmittel gemeint, bei POIs beschreibt der Typ um welche Kategorie es sich handelt (Krankenhaus, Schule, Restaurant, Sehenswürdigkeit etc.). Auch hier werden aufgrund der hohen Datenmenge nur Auszüge der Datei gezeigt.

Resultierende Excel-Datei für die Straßenbahnen:

ID	Name	Typ	Breitengrad	Längengrad	Überdachung
264634923	Guillaume De Greef	Strassenbahnhaltestelle	50,882582	4,335606	Ja
264635045	Démineurs - Ontmijners	Strassenbahnhaltestelle	50,877892	4,332885	
291777803	Schweitzer	Bushaltestelle	50,86493	4,296821	
319243864	Stade - Stadion	Strassenbahnhaltestelle	50,893604	4,332299	Ja
319243874	Stade - Stadion	Strassenbahnhaltestelle	50,893502	4,33204	Ja
461715389	Cimetière de Jette - Kerkhof van Jette	Strassenbahnhaltestelle	50,879095	4,333836	
461715399	Cimetière de Jette - Kerkhof van Jette	Strassenbahnhaltestelle	50,879614	4,334466	
462754947	Jette Gare - Jette Station	Strassenbahnhaltestelle	50,880129	4,330585	
474388213	Berchem Station	Strassenbahnhaltestelle	50,872437	4,289965	
496270403	Cimetière de Jette - Kerkhof van Jette	Bushaltestelle	50,879721	4,334375	
496270474	Cimetière de Jette - Kerkhof van Jette	Strassenbahnhaltestelle	50,879339	4,333697	
496273939	Guillaume De Greef	Bushaltestelle	50,882543	4,335405	
496273940	Guillaume De Greef	Bushaltestelle	50,883436	4,335388	
496278330	Hôpital Brugmann - Brugmann Ziekenhuis	Bushaltestelle	50,885873	4,33295	
496278334	Crocq	Bushaltestelle	50,885153	4,330495	
496278337	Hôpital Brugmann - Brugmann Ziekenhuis	Bushaltestelle	50,885451	4,33253	
496278358	Kufferath	Strassenbahnhaltestelle	50,889161	4,334153	Ja
496287608	Stiénon	Strassenbahnhaltestelle	50,890874	4,330302	Ja

Resultierende Excel-Datei für die Points of Interest:

ID	Name	Typ	Breitengrad	Längengrad
270839197	CHU Brugmann - Brugmann UH	POI - hospital	50,886206	4,3325393
286252416	Maison communale de Jette - Gemeentehuis van Jette	POI - townhall	50,8757596	4,3245814
344913323	Gemeentelijke Nederlandstalige basisschool	POI - school	50,9077967	4,3083944
344913386	École communale primaire	POI - school	50,907331	4,3117362
460041875	GB	POI - supermarket	50,8731665	4,3269587
493193906	Hôpital des Enfants Reine Fabiola	POI - hospital	50,8885403	4,3288842
493227894	Delhaize	POI - supermarket	50,893613	4,3234183
493683701	Friterie du Miroir	POI - fast_food	50,8728794	4,3253622
503099548	Quick	POI - fast_food	50,8711517	4,2926553
503105913	Carrefour	POI - supermarket	50,8720291	4,2922004
503105916	Lunch Garden	POI - restaurant	50,8716987	4,291428
503105932	Delhaize	POI - supermarket	50,8720562	4,2970327
506113729	Lidl	POI - supermarket	50,9041245	4,3060727
506113730	Carrefour	POI - supermarket	50,9033451	4,3074546
513423749	Colruyt Jette	POI - supermarket	50,8736593	4,3275252
513425540	Universitair Ziekenhuis Brussel	POI - hospital	50,8872628	4,3089256
513425541	Mediamarkt	POI - supermarket	50,8718178	4,294755
513425542	Brico	POI - supermarket	50,8719803	4,2946177
513425543	Pita Mania	POI - fast_food	50,8716972	4,3255996
513425544	Le Delvaux	POI - restaurant	50,8748292	4,3269672
513425545	Delhaize	POI - supermarket	50,8778892	4,328332
513425546	Masson	POI - pharmacy	50,8730907	4,3253278

3.3 Erstellen der Datenbank

Nachdem alle benötigten Daten gesammelt worden sind, kann nun die Datenbank als Grundlage der Projektarbeit erstellt werden. Bevor nun die eigentliche Datenbank realisiert werden kann, werden die Daten zunächst in mehreren Excel Dateien strukturiert und festgehalten. Diese strukturierten Excel Dateien werden hier nicht weiter aufgeführt, befinden sich aber in der beiliegenden CD. Auch die fertige Ladedatei beinhaltet 2500 Zeilen, daher werden nur die ersten zwanzig, sowie die fünf letzten Zeilen der jeweiligen Tabellen gezeigt.

```
DROP DATABASE IF EXISTS bruessel;

SHOW WARNINGS;
CREATE DATABASE IF NOT EXISTS bruessel;
SHOW WARNINGS;
USE bruessel;
SHOW WARNINGS;

DROP TABLE IF EXISTS tramstop;
SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS tramstop (
```

```
      id int(4),
      name varchar(50),
      lat float(15),
      lon float(15), shelter varchar(5),
      wheelchair varchar(5)
);
SHOW WARNINGS;

INSERT INTO tramstop (id, name, lat, lon, ramp, lines) VALUES
(1, "Abbaye", 50.8196713, 4.3705183, ""),
(2, "Acacias", 50.8331381, 4.3923309, ""),
(3, "Albert", 50.8216146, 4.3426237, ""),
(4, "Albert I", 50.839173, 4.32312, ""),
(5, "Alcyons", 50.8675348, 4.2938105, ""),
(6, "Alma", 50.849607, 4.452647, "Ja"),
(7, "Alphonse XIII", 50.7916443, 4.3671229, ""),
(8, "Altitude 100", 50.8166857, 4.33624, ""),
(9, "Amitié", 50.844605, 4.464735, ""),
(10, "Anneessens", 50.8440974, 4.3452831, ""),
(11, "Araucaria", 50.893415, 4.3628651, ""),
(12, "Arsenal", 50.8261445, 4.3967334, ""),
(13, "Arts et Métiers", 50.8474729, 4.3381473, ""),
(14, "Arts-Loi", 50.84555, 4.36861, ""),
(15, "Auderghem-Forêt", 50.8184014, 4.4510344, ""),
(16, "Auderghem-Shopping", 50.816143, 4.426039, ""),
(17, "Aumale", 50.839624, 4.312291, ""),
(18, "Avenue du Roi", 50.8318108, 4.3337334, ""),
(19, "Aviation", 50.8346524, 4.4565258, ""),
(20, "Azur", 50.8666821, 4.2869841, ""),

…

(325, "Yser", 50.857719, 4.350113, ""),
(326, "Zaman", 50.814176, 4.3249651, ""),
(327, "Zaman-Forest National", 50.814176, 4.3249651, ""),
(328, "Mabru", 50.874240, 4.363880, ""),
(329, "Tenreuken", 50.8050046, 4.42349395, ""),

;
SHOW WARNINGS;

DROP TABLE IF EXISTS routes;
SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS routes (
      id int(4),
      name varchar(20),
      hexcolor varchar(7),
      color varchar(10)
);
SHOW WARNINGS;

INSERT INTO routes (id, name, hexcolor, color)VALUES
(1, "Linie 1", "#A020F0", "lila"),
(2, "Linie 2", "#FFE600", "orange"),
(3, "Linie 5", "#FFFF00", "eigelb"),
(4, "Linie 6", "#0000FF", "blau"),
(5, "Linie 3", "#A2C257", "avocado"),
(6, "Linie 4", "#E0427F", "pink"),
(7, "Linie 7", "#FFFF7E", "papaya"),
(8, "Linie 19", "#CD6600", "dunkelorange"),
(9, "Linie 25", "#871F78", "purpur"),
(10, "Linie 31", "#D1E231", "birne"),
```

```
(11, "Linie 32", "#FF66002", "gelb"),
(12, "Linie 33", "#EE82EE", "violet"),
(13, "Linie 39", "#CD6600", "dunkelorange"),
(14, "Linie 44", "#FCD116", "dunkelgelb"),
(15, "Linie 51", "#FCD116", "dunkelgelb"),
(16, "Linie 55", "#FCD116", "dunkelgelb"),
(17, "Linie 81", "#008000", "grün"),
(18, "Linie 82", "#00BFFF", "hellblau"),
(19, "Linie 83", "#A2C257", "avocado"),
(20, "Linie 92", "#CD6600", "dunkelorange"),
(21, "Linie 94", "#FF66002", "gelb"),
(22, "Linie 97", "#871F78", "dunkellila"),
;
SHOW WARNINGS;

DROP TABLE IF EXISTS poi;
SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS poi (
      id int(4),
      name varchar(50),
      lat float(15),
      lon float(15),
      kind varchar(50),
      info varchar(50)
);
SHOW WARNINGS;

INSERT INTO poi (id, name, lat, lon, kind, info)VALUES
(1, "Dexia", 50.8677102, 4.3350208, "Bank", ""),
(2, "ING Ganshoren Basilique", 50.8685603, 4.3124231, "Bank", ""),
(3, "BNP Paribas Fortis Basilique", 50.8677681, 4.3140217, "Bank", ""),
(4, "ING Miroir", 50.8723133, 4.3242156, "Bank", ""),
(5, "Dexia", 50.8717785, 4.3261039, "Bank", ""),
(6, "Delta Lloyd", 50.8725977, 4.3231856, "Bank", ""),
(7, "BNP Paribas Fortis Jette", 50.8729565, 4.3244194, "Bank", ""),
(8, "KBC", 50.872908, 4.3268742, "Bank", ""),
(9, "Citibank", 50.8730367, 4.3258013, "Bank", ""),
(10, "BNP Paribas Fortis Jette-Le Miroir", 50.8728335, 4.3264236, "Bank",
""),
(11, "Dexia", 50.8749209, 4.3010715, "Bank", ""),
(12, "DEXIA", 50.9090745, 4.3075754, "Bank", ""),
(13, "BNP Paribas Fortis", 50.8324425, 4.4496068, "Bank", ""),
(14, "Dexia", 50.8270331, 4.4594335, "Bank", ""),
(15, "Fortis", 50.8625125, 4.464054, "Bank", ""),
(16, "Dexia", 50.8484913, 4.4658134, "Bank", ""),
(17, "Centea", 50.8493145, 4.4653157, "Bank", ""),
(18, "KBC", 50.8448236, 4.4179763, "Bank", ""),
(19, "BNP Paribas Fortis", 50.847737, 4.4079303, "Bank", ""),
(20, "AXA", 50.8464716, 4.4146189, "Bank", ""),

…

(1387, "Théâtre Marni", 50.8289448, 4.3715017, "Theater", ""),
(1388, "Le Petit théâtre Mercelis - Klein Mercelis Theater", 50.8329009,
4.3662431, "Theater", ""),
(1389, "Cirque Pauwels - Pauwels Circus", 50.7880678, 4.3307868, "Theater",
""),
(1390, "De Moelie", 50.7731893, 4.3348195, "Theater", ""),
(1391, "Stalstudio", 50.9154633, 4.4095646, "Theater", ""),


;
```

```
SHOW WARNINGS;

DROP TABLE IF EXISTS routetime;
SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS routetime (
      id int(4),
      name varchar(20),
      endstop varchar(7),
      time varchar(10)
);
SHOW WARNINGS;

INSERT INTO routetime (id, name, endstop, time)VALUES
(1, "Linie 1", "Gare de l'Ouest - Stockel", "27 min"),
(2, "Linie 2", "Simonis (Leopold II) - Simonis (Elisabeth)", "21 min"),
(3, "Linie 5", "Erasme - Herrmann-Debroux", "36 min"),
(4, "Linie 6", "Roi Baudouin  - Simonis (Elisabeth)", "34 min"),
(5, "Linie 3", "Esplanade  - Churchill", "26 min"),
(6, "Linie 4", "Gare du Nord - Stalle P", "30 min"),
(7, "Linie 7", "Vanderkindere - Heysel", "45 min"),
(8, "Linie 19", "De Wand - Groot-Bijgaarden", "25 min"),
(9, "Linie 25", "Boondael Gare - Rogier", "35 min"),
(10, "Linie 31", "Gare du Nord - Marius Renard", "28 min"),
(11, "Linie 32", "Gare du Nord - Drogenbos Château", "32 min"),
(12, "Linie 33", "Bordet Station - Stalle P", "54 min"),
(13, "Linie 39", "Montgomery - Ban Eik", "20 min"),
(14, "Linie 44", "Montgomery - Tervuren Station", "19 min"),
(15, "Linie 51", "Heysel - Van Haelen", "48 min"),
(16, "Linie 55", "Bordet Station - Rogier", "20 min"),
(17, "Linie 81", "Montgomery - Marius Renard", "42 min"),
(18, "Linie 82", "Berchem Station - Drogenbos Château", "35 min"),
(19, "Linie 83", "Berchem Station - Montgomery", "47 min"),
(20, "Linie 92", "Schaerbeek Gare - Fort-Jaco", "45 min"),
(21, "Linie 94", "Stade - Musée du Tram", "86 min"),
;
SHOW WARNINGS;

DROP TABLE IF EXISTS stops_on_routes;
SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS stops_on_routes (
      route_id int(4),
      stop_id int(4),
      timefstart int(4),
      distancefstart int(10),
      sequencefstart int(4)
);
SHOW WARNINGS;

INSERT INTO stops_on_routes (route_id, stop_id, time, distance,
sequence)VALUES


(1, 127, 0, 0, 1),
(1, 29, 1.0669465274977, 0.482875893257244, 2),
(1, 112, 2.67877322406176, 1.21235224077825, 3),
(1, 75, 3.92939657547349, 1.77835611480334, 4),
(1, 264, 5.27058118265393, 2.38534596717578, 5),
(1, 83, 6.04963981411431, 2.73793030282049, 6),
(1, 126, 7.57485327958195, 3.4282074587647, 7),
(1, 221, 8.45808840374877, 3.82793972139128, 8),
(1, 14, 9.46410515177252, 4.2832401730203, 9),
(1, 192, 10.7640877305391, 4.87158291819329, 10),
```

```
(1, 268, 11.698192207684, 5.29433750070714, 11),
(1, 202, 14.4993777250211, 6.56209077981075, 12),
(1, 208, 15.9635130481233, 7.22472534843395, 13),
(1, 165, 16.9061475001226, 7.65134040485271, 14),
(1, 138, 18.0075976033344, 8.14983183695293, 15),
(1, 291, 19.2895531561538, 8.73001595745948, 16),
(1, 256, 21.2139240466399, 9.60094274596596, 17),
(1, 309, 23.1619336787791, 10.4825678948842, 18),
(1, 6, 23.9918416597076, 10.8581654972795, 19),
(1, 79, 24.9323331023926, 11.2838106760987, 20),
(1, 280, 26.9999910480921, 12.2195859485687, 21),


(2, 271, 0, 0, 1),
(2, 216, 1.60225947932812, 0.785107144870779, 2),
(2, 29, 2.50317992399577, 1.22655816275793, 3),
(2, 127, 3.48864093064321, 1.70943405601517, 4),

...


(97, 317, 33.6040612575746, 7.68795352673292, 24),
(97, 134, 36.4918029891005, 8.34861248872591, 25),
(97, 195, 37.8823744347547, 8.66674810239023, 26),
(97, 321, 39.5231138738759, 9.04211727163307, 27),
(97, 97, 40.9825532876104, 9.37600853262892, 28),
;
SHOW WARNINGS;
```

Eine Herausforderung stellte das Abspeichern von Sonderzeichen in der SQL Datenbank dar. Was dieses besonders "interessant" machte war ein unterschiedliches Verhalten der Datenbank je nach Rechner auf dem sie ausgeführt wurde.
Auf dem privaten Rechner wo der Großteil der Webseite entwickelt wurde funktionierte eine Ladedatei ohne besondere Modifikationen problemlos. Auf dem Hochschulrechner kam es mit dieser aber zur einer falschen Darstellung von Sonderzeichen.
In der Anleitung wurde empfohlen die SQL Datenbank auf ein UTF8 Character Set umzustellen. Eine so modifizierte Ladedatei brachte auf den Hochschulrechnern aber auch nicht das gewünschte Resultat.
Erfolg brachte auf dem Hochschulrechner schließlich eine Ladedatei mit einem Binary Character Set.
Das die Webseite aber schließlich auf einem anderen Rechner gespeichert werden soll wo wiederum andere Bedingungen herrschen können haben wir uns dazu entschlossen alle drei Variationen - bruessel.sql, bruessel_binary.sql und bruessel_utf8.sql - mitzuliefern. Mit Gottes Hilfe wird eine funktionieren.

3.4 Entity-Relationship-Modell

Durch einen entsprechenden SQL-Befehl kann die Ladedatei ausgeführt werden. Falls es nun im weiteren Verlauf des Projekts nötig ist, einzelne Daten zu ändern, müssen die Anpassungen in der Ladedatei ausgeführt werden. Dabei wird nach jeder Änderung die Ladedatei wieder ausgeführt, um so die Datenbank zu aktualisieren.
Einen umfassenden Überblick über Struktur, Entitäten und Verknüpfungen der Datenbankbestandteile lässt sich aus folgendem Entity-Relationship-Modell entnehmen.

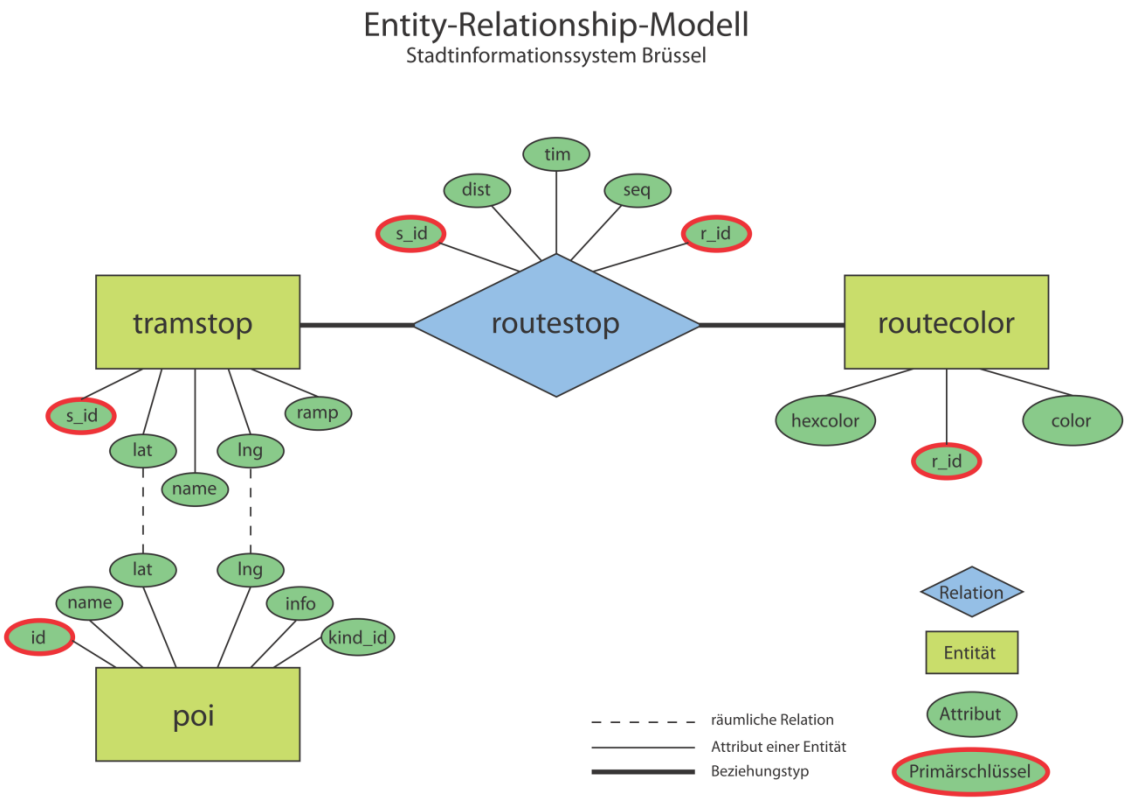

Abbildung 3: Entity-Relationship-Modell

4 Realisierung des Informationssystem

4.1 Erstellung der HTML-Seiten

Nachdem das Konzept für den Aufbau und die Struktur mit Hilfe von Scribbles steht, kann mit der Umsetzung der graphischen Oberfläche in HTML begonnen werden. Eine weitere Seitenprogrammiersprache wie PHP, welche später in auch noch Anwendung findet, kann problemlos integriert werden.
Für die Umsetzung werden Schrift, Hintergrund und weitere Bestandteile der graphischen Oberfläche in sogenannten Stylesheets (siehe 4.2) vordefiniert.
Die Seite besteht aus zwei Bereichen, welche das Grundgerüst für die Seite stellen. Dabei ist im oberen Bereich ein Coulage von unterschiedlichen Bildern aus Brüssel mit dem Namen der Seite. Darunter befinden sich vier Navigations-Buttons Home (Startseite), Abfragen, Downloads und Impressum. Auf der Startseite wird eine kurze Einführung über die Stadt gegeben. Auf der Seite der Abfragen findet die PHP-Programmierung Anwendung, hier stehen 20 statische und dynamische Abfragen zum Liniennetz in Brüssel zur Verfügung. Unter Downloads befinden sich Dateien die zur Realisierung des Informationssystems benötigt wurden.
Im unteren Bereich befindet sich jeweils der Name der Hochschule Karlsruhe, sowie ein Button zur Validierung der Seite führt.
Die verwendeten Bilder wurden aus dem Internet entnommen und evtl. mit Adobe Photoshop und Illustrator bearbeitet worden.
Exemplarisch für den HTML-Code wird die Startseite aufgeführt.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de-de">
      <head>
            <title>Liniennetz Bruxelles</title>
            <meta http-equiv="content-type" content="text/html;charset=utf-
8" />
            <link rel="stylesheet" href="format.css" type="text/css"/>
      </head>
      <body>
                        <table style="margin: 0 auto;" width="100%"
cellspacing="0" cellpadding="0" border="0">

            <tr>
            <td><!--ende aussentab-->
```

```
            <table style="margin: 0 auto;" width="100%"  cellspacing="0"
cellpadding="0" border="0"  id="top"   >
            <tr>
            <td valign="top" id="hpname">Stadtinformationssystem
Br&uuml;ssel</td>
            </tr>
            <tr>

            <td> </td>
            </tr>
            <tr>
            <td valign="bottom"  class="buleiste" ><!--men? oben
(width="100%")-->
            <table    id="menu2" width="100%"  border="0" cellpadding="0"
cellspacing="0"    >
            <tr>
            <td width="25%"><a href="index.php"
onfocus="this.blur()">HOME</a></td> <!--width="20%"-->
            <td width="25%"><a href="query.php"
onfocus="this.blur()">ABFRAGEN</a></td>

            <td width="25%"><a href="download.php"
onfocus="this.blur()">DOWNLOADS</a></td>
            <td width="25%"><a href="impress.php"
onfocus="this.blur()">IMPRESSUM</a></td>

                        </tr>
                              </table>
                  <!-- ende men? oben-->
                        </td>
                  </tr>

            </table>
                  <!--ende header-->

            <table style="margin: 0 auto;"  width="100%"  id="main"
border="0" cellpadding="0" cellspacing="0"  >
            <tr>

            <td valign="top"  id="sp1" height="700">

            <h2>Herzlich Willkommen auf dem Stadtinformationssystem von
Br&uuml;ssel</h2>
            Die Region Br&uuml;ssel-Hauptstadt (oder auch Br&uuml;sseler
Region; frz. Region de Bruxelles-Capitale, ndl. Brussels Hoofdstedelijk
Gewest) ist eine der drei Regionen Belgiens.
            Sie umfasst das "zweisprachige Gebiet Br&uuml;ssel-Hauptstadt",
welches genau wie der gleichnamige Verwaltungsbezirk aus den neunzehn
Gemeinden in und um Br&uuml;ssel besteht.

            <h2>Liniennetz</h2>

            Die Metro Br&uuml;ssel bildet zusammen mit der
Stra&szlig;enbahn das R&uuml;ckgrat des schienengebundenen
&ouml;ffentlichen Personennahverkehrs in der Region Br&uuml;ssel-
Hauptstadt, welche neben dem administrativen Stadtgebiet Br&uuml;ssels 18
weitere selbstst&auml;ndige Gemeinden umfasst, die jedoch ein
zusammenh&auml;ngendes st&auml;dtisches Siedlungsgebiet bilden. Die einzige
Untergrundbahn Belgiens entwickelte sich aus der Untertunnelung von
Stra&szlig;enbahnstrecken, welche stufenweise auf U-Bahn-Betrieb umgestellt
wurden. Die Er&ouml;ffnung der ersten U-Bahnstrecke fand 1976 statt.
Betreiber des Systems mit derzeit vier Linien ist die Soci&eacute;t&eacute;
des Transports Intercommunaux de Bruxelles/Maatschappij voor het
```

Intercommunaal Vervoer te Brussel, kurz STIB/MIVB.

```
            <h2>Metro</h2>
```

Das Brüsseler Metrosystem hat eine Länge von 38 Kilometern, verfügt über 59 Stationen und ist von 5:00 Uhr bis 1:00 Uhr im Einsatz. Brüssel hat insgesamt vier U-Bahn-Linien, wobei sich je zwei Linien weite Streckenabschnitte teilen. Das Netz ist mit einem durchschnittlichen Stationsabstand von 650 m recht feinmaschig, was sich dadurch erklärt, dass bei der Umstellung vom Straßenbahn- zum U-Bahn-Betrieb oftmals die Lage der Haltestellen beibehalten wurde. Die Züge fahren zur Spitzenzeit in 6-Minuten-Intervallen, durch die Linienüberlappungen ergeben sich auf vielen Streckenabschnitten 3-Minuten-Intervalle. Die Linien 1 und 5 befahren zusammen die in Ost-West-Richtung führende Strecke durch die Innenstadt vom Westbahnhof bis Merode. Am östlichen Enden dieser Strecke teilt sie sich in zwei Äste auf, die jeweils von einer der beiden Linien befahren wird. Die Linien 2 und 6 verkehren auf dem am 4. April 2009 fertiggestellten "Kleinen Ring". Ergänzt wird das Netz durch unterirdische Straßenbahnstrecken (Prémétro), wobei die von den Tramlinien 3 und 4 befahrene Nord-Süd-Strecke durch die Innenstadt die wichtigste ist.

```
                        <br/><br/><br/>

            <!-- ende inhalt-->
            </td>
                  </tr>

                        <tr>
                  <td  id="fussnote">
                        <a href="http://validator.w3.org/check?uri=referer"
id="validator">
                              <img src="http://www.w3.org/Icons/valid-
xhtml10-blue"
                              class="div_button"
                              alt="Valid XHTML 1.0 Transitional"/>
                        </a>


                        HOCHSCHULE KARLSRUHE - TECHNIK UND WIRTSCHAFT
            </td>
                  </tr>
                        </table>
            <!-- aussentab-->
                        </td>
                  </tr>
            </table>
      </body>
</html>
```

4.2 Erstellung eines Stylesheets mittels CSS

Parallel zur Erstellung der HTML-Seite wird eine CSS-Datei (format.css) angelegt. In dieser werden alle Größen definiert (Schriftgröße, Farbe etc.) Der große Vorteil beim Arbeiten mit Stylesheets ist, dass bei etwaigen Änderung des Layouts der Seite nur das entsprechende Element im Stylesheet geändert werden muss. Die verwendete CSS-Datei sieht wie folgt aus:

```
body
{margin-top: 0px;
margin-left: 0px;
margin-right: 0px;
margin-bottom: 0px;
background-color: #e6e6e6;
}

table,td {font-size: 97%;
line-height: 125%;
font-family: arial, helvetica, tahoma ,verdana,  sans-serif;
color:#000; align: center;
}


h2{ font-family: "Trebuchet MS",arial, helvetica, verdana, calibri, tahoma,
sans-serif;
color:#7F7F7F;
margin-bottom:20px;
letter-spacing:2px;
font-size: 125%;
border-bottom:solid 0px #746b64;
padding-bottom:4px
}

h4{ font-family: "Trebuchet MS",arial, helvetica, verdana, tahoma, sans-
serif;
color:#7F7F7F;
font-size: 100%;
border-bottom:solid 0px #746b64;
padding-bottom:4px
}

/*table.center{ align: center; }*/

.li {
width:14px;
background-image:url(grafics/ra1.jpg);
background-repeat:repeat-y;
vertical-align:top;
border-right:solid 0px #BFBFBF;
}


.re {
```

```
width:14px;
background-image:url(grafics/ra2.jpg);
background-repeat:repeat-y;
vertical-align:top;
border-left:solid 0px #BFBFBF;}

/* hauptgerüst */

#main {
border-bottom:solid 10px #fff;
border-right:solid 10px #fff;
border-left:solid 10px #fff;
height: 600px;
}

#top {background-image:url(grafics/tram_oben3.jpg);
height:198px;
border-top:solid 10px #fff;
border-right:solid 10px #fff;
border-left:solid 10px #fff;
background-repeat:no-repeat;
background-position: center top;
color:#2A3C22;
}

#hpname {height:36px;
padding-left:25px;
padding-top:107px;
text-align:left;
color:white;
letter-spacing:1px;
font-size:20px;
font-family: "Trebuchet MS",arial, helvetica, verdana, tahoma, sans-serif;
font-style:italic;

}

/* menue oben */

.buleiste {

height:36px;
}

#menu2 a, #menu2 a:visited , #menu2 a:active {display: block;
color:#000;

text-decoration:none;
font-family:calibri, sans-serif;
font-size: 13pt;
padding-left: 0px; padding-bottom: 0px; padding-top:0px;
margin-left: 0px; margin-right: 0px;
border-left:solid 1px #fff;
border-bottom:solid 1px #E6E6E6;
border-top:solid 1px #fff;
line-height:36px;
text-align:center;
background-image:url(grafics/mover.jpg);
}


#menu2 a:hover {background-color:#fff; color:#808080; text-decoration:none ;
```

```
border-left:solid 1px #fff;
border-bottom:solid 1px #E6E6E6;
border-top:solid 1px #fff;
}


#sp1 {text-align:justify;
padding-left: 30px;padding-right: 30px;padding-top: 40px;padding-bottom:
50px;
background-color:#fff;width:100%;
background-image:url(grafics/bgmain.jpg);
background-repeat:repeat-x;
border-bottom:solid 1px #bfbfbf;
}


#fussnote {font: normal 14px calibri, sans-serif;
color: #1F1F1F;
padding-top:10px;
height:40px;
text-align:center;
border-top:solid 1px #fff;
background-color:#fff;
background-image:url(grafics/MOVER.jpg);
background-repeat:repeat-x;
}



body
{scrollbar-arrow-color: #737b66; scrollbar-base-color: #F5F8F3;
scrollbar-highlight-color : #737b66; scrollbar-shadow-color : #ffffff;
SCROLLBAR-TRACK-COLOR: #e6e6e6;}


/* menue  */


#menu1 a , #menu1 a:visited , #menu1 a:active {display:block;
background-color:#;
text-decoration:none ;
text-align:center;
font-size: 89%;
line-height: 180%;
font-family: "Trebuchet MS",arial, helvetica, verdana, tahoma, sans-
serif;color:#7B917C;
width:160px;
}

#menu1 a:hover{
background-color:#A5ADAA;
color:#fff;
text-decoration:none ;
}

/* Karte und Legende */

#mapWrapper {
      float:left;
      width:100%;
}
```

```
#map {
      height: 660px;
      border-left:solid 10px #fff;
}

#legend {
      background-image:url(grafics/Legende.png);
      background-repeat:no-repeat;
      background-position: right top;
      background-color: white;
      float:right;
      position:relative;
      width:30px;
      height:660px;
      border-right:solid 10px #fff;
}

#legendMax {
      float:right;
      position:relative;
      width:300px;
      height:640px;
      padding: 10px;
      background-color: white;
    font-family: verdana;
    font-size: 14px;
}

#validator1 {
    float:left;
    padding-left:15px;
}
#validator2 {
    float:right;
    padding-right:15px;
}

/* allgemeine links im text */

a:link, a:visited, a:active{ font-size: 98%;
font-family: arial, helvetica, verdana, tahoma, sans-serif;color:#000}

a:hover{ text-decoration:UNDERLINE;background-color:#C5C8CD;
color:#3F3F3F;}
```

Eine Herausforderung stellte das Abspeichern von Sonderzeichen in der SQL Datenbank dar. Was dieses besonders "interessant" machte war ein unterschiedliches Verhalten der Datenbank je nach Rechner auf dem sie ausgeführt wurde.

Auf dem privaten Rechner wo der Großteil der Webseite entwickelt wurde funktionierte eine Ladedatei ohne besondere Modifikationen problemlos. Auf dem Hochschulrechner kam es mit dieser aber zur einer falschen Darstellung von Sonderzeichen.

In der Anleitung wurde empfohlen die SQL Datenbank auf ein UTF8 Character Set

umzustellen. Eine so modifizierte Ladedatei brachte auf den Hochschulrechnern aber auch nicht das gewünschte Resultat.

Erfolg brachte auf dem Hochschulrechner schließlich eine Ladedatei mit einem Binary Character Set.
Das die Webseite aber schließlich auf einem anderen Rechner gespeichert werden soll wo wiederum andere Bedingungen herrschen können haben wir uns dazu entschlossen alle drei Variationen - bruessel.sql, bruessel_binary.sql und bruessel_utf8.sql - mitzuliefern. Mit Gottes Hilfe wird eine funktionieren.

4.3 Grundlagen der Funktionalität

Generell wurde bei dem Design der Funktionalität darauf geachtet unnötige Benutzereingaben zu vermeiden. So werden Abfragen automatisch durchgeführt sobald ein Benutzer eine Auswahl trifft ohne das er danach noch auf ein "OK" Button klicken muss.
In den Karten wurde der maximale und minimale Zoom eingeschränkt, da ein Zoom größer als der Stadtgebiet Brüssel oder kleiner als die minimale Größe um Überlappungen der Symbole zu vermeiden keinen Sinn macht.
Für die Karten (Query15 - 20) wird die Google Maps JavaScript V3 API verwendet. Neben dieser werden auch Funktionalitäten von zwei weiteren Programmbibliotheken genutzt:

Infobubble

link: http://code.google.com/p/google-maps-utility-library-v3/source/browse/trunk/infobubble/
Ermöglicht die Erstellung optisch und funktionell besserer Infofenster. Bei der verwendeten Version kam es allerdings ab und zu zu fehlerhaften Fensterbreiten. Dieses Problem wurde behelfsmäßig gelöst indem die Fensterbreite in Zeile 1661 von infobubble.js generell um 10 Pixel erhöht wurde.

Abbildung 4: Fehlerhafte (links) und korrekte (rechts) Darstellung einer Infobubble

gmap3

link: http://gmap3.net/

gmap3 ist einen auf jQuery basierende JavaScript API für Google Maps welche einige Funktionen erleichtert bzw. überhaupt möglich macht.

Auch hier wurde der bestehende Sourcecode verändert, allerdings nicht wegen einem Bugfix, sondern um eine Funktionalität zu erweitern. gmap3 Marker Objekte haben ein "Tag" Attribut mit dem Marker in unterschiedliche Klassen eingeordnet werden können. Polylinien hatten dies Attribut allerdings nicht, was eine Auswahl von diesem kompliziert machte. Deswegen wurde in Zeile 1452f von gmap3.js das polyline Objekt um ein Tag Attribut erweitert.

Generell ist zu den Karten noch anzumerken das sich der Programmierstil von Query16 und Query20 von den anderen Karten leicht unterscheidet. Grob gesagt ist bei 16 und 20 praktisch alles in einem großen php Block während in den anderen Karten etwas feiner mit php umgegangen wurde. Dies ist darauf zurückzuführen das 16 und 20 als erste Abfragen erstellt wurden (auch vor den Abfragen 1-14 ohne Karte) und 15, 17-19 als letzte Abfragen. In dieser Zeit hat sich durch den Lerneffekt der Programmierstil etwas geändert, was sich auch im Code bemerkbar macht.

Es kommt dadurch von Query 1 zu Query 14 auch zu einer gewissen Variation (oder Evolution) in der Methodik für die Benennung der Variablen.

4.4 Kurzbeschreibung der Abfragen

Es wird im Folgenden auf alle Abfragen kurz eingegangen. In den zwei folgenden Kapiteln werden zwei davon beispielhaft genauer durchgegangen.

Abfrage 1:

Dynamikstufe 0

Statische Abfrage der Anzahl der Linien.

Abfrage 2:

Dynamikstufe 0

Statische Abfrage der Gesamtlänge aller Linien. Hier ist anzumerken das dies nicht der Gesamtlänge des Streckennetzes entspricht da viele Linien sich Streckenabschnitte teilen.

Abfrage 3:

Dynamikstufe 1

Angabe der kürzesten bzw. längsten Linie, je nach Benutzereingabe. Die Besonderheit dieses (und des nächsten) Queries ist das die php Eingabe ("min" bzw. "max") direkt in eine SQL Abfrage eingebaut wird. Dies ist in diesem Fall unbedenklich, sollte aber normalerweise vermieden werden da es zu Sicherheitslücken führen kann.

Abfrage 4:

Dynamikstufe 1

Angabe der Linie mit der niedrigsten bzw. höchsten Durchschnittsgeschwindigkeit, je nach Benutzereingabe. Abgesehen von einer leicht komplexeren SQL Abfrage in der Funktionsweise identisch mit Query3.

Abfrage 5:

Dynamikstufe 0

Statische Berechnung dreier Werte einer durchschnittlichen Linie durch jeweils eine SQL Abfrage pro Wert.

Abfrage 6:

Dynamikstufe 1

Auslesen aller Werte einer vom Benutzer Ausgewählten Linien mittels einer einzelnen SQL Abfrage.

Abfrage 7:

Dynamikstufe 0

Statische Abfrage der Anzahl der Haltstellen.

Abfrage 8:

Dynamikstufe 1

Abfrage aller Linien die eine angegeben Haltstelle anfahren.
Dies kann entweder durch Eingabe des kompletten oder eines Teils des Liniennamens in ein Textfenster geschehen (starten der Suche mit Enter oder Mausklick) oder durch Auswahl des Liniennamens aus einer Dropdown Liste.
Zusätzlich kann beim Textfenster durch eine Checkbox nur nach dem genauen Namen gesucht werden. Z.B. damit man bei Eingabe von "Albert" nur die Daten dieser Haltstellen angegeben bekommt und nicht zusätzlich die von "Albert I".

Abfrage 9:

Dynamikstufe 0

Statische Abfrage aller Haltestellen mit Rollstuhlrampen.

Abfrage 10:

Dynamikstufe 2

Abfrage der Entfernung und Fahrzeit zwischen zwei Haltestellen einer Linie. Im linken Dropdown Menu wird eine Linie ausgewählt worauf das "Haltestelle 1" und "Haltestelle 2" Dropdown Menu mit deren Linien gefüllt wird. Sobald dort bei beiden eine Auswahl getroffen wird werden die Ergebnisse angezeigt.
Diese Abfrage hat noch ein paar Besonderheiten, daher wird sie in 4.5 detailliert durchgegangen.

Abfrage 11:

Dynamikstufe 0

Statische Abfrage der Typen von POIs mit Anzeige ihres Symbols in der Karte.

Abfrage 12:

Dynamikstufe 0

Abfrage der Anzahl von POIs eines vom Nutzer ausgewählten Typs.

Abfrage 13:

Dynamikstufe 1

Abfrage des Namen, Typs und Entfernung von POIs innerhalb eines von Nutzer angegebenen Maximalentfernung eine ausgewählten Haltestelle.

Abfrage 14:

Dynamikstufe 2

Abfrage der die nahesten Haltestelle und ihrer Entfernung zu einem bestimmten POI. Hierzu wird zunächst der POI-Typ ausgewählt worauf das zweite Dropdown Menu sich mit allen Namen der POIs dieses Typs füllt. Wird dort einer ausgewählt werden zu diesem die Werte berechnet.

Abfrage 15:

Dynamikstufe 2

Abfrage der Position einer Haltestelle und Anzeige von dieser auf der Karte. Wenn im "Linie" Dropdown Menu eine Linie gewählt wird werden nur Haltestellen dieser Linie im "Haltestellen" Dropdown Menu aufgelistet, wenn nicht können dort alle Haltestellen ausgewählt werden.

Abfrage 16:

Dynamikstufe 0

Je nach Benutzerauswahl Anzeige einer oder mehrerer Linien und ihrer Stationen auf der Karte. Anzumerken ist hier das die gesamte Information aller Linien und Stationen beim Laden der Abfrage schon abgerufen wird. Die unterschiedlichen Anzeigen geschehen also nicht durch das Laden von neuen Informationen sondern durch das Ausblenden bereits vorhandenen. Es hier daher um eine statische Abfrage, auch wenn sie nicht wirklich so aussieht.

Abfrage 17:

Dynamikstufe 1

Bei Rechtsklick auf die Karte Abfragen der nahesten Haltestelle zu diesem Kartenpunkt.

Anmerkung: Der Test dieser Abfrage verursachte eine kleine Panik, da sie nach 6 Monaten auf einmal nicht mehr funktionierte. Die Ursache dafür war, wie sich herausstellte, eine Änderung des von (wahrscheinlich) Google Maps zurückgelieferten Speicherorts der Koordinaten. Anstatt event.latLng.Ia bzw. .Ja als Speicherort für die Lat bzw. Lng Werte waren diese nun in event.latLng.Pa bzw. .Qa zu finden. Warum diese Änderung stattgefunden hat ist nicht bekannt.

Abfrage 18:

Dynamikstufe 2

Abfrage der Adresse eines bestimmte POIs und Anzeige von diesem in der Karte. Analog wie in Abfrage 15 kann mit Auswahl einer Kategorie die Liste der POIs eingeschränkt werden

Abfrage 19:

Dynamikstufe 1

Nach Auswahl einer POI Kategorie und Rechtsklick auf einen Kartenpunkt wird der naheste POI dieser Kategorie zu diesem Kartenpunkt und eine Route zu diesem angezeigt.

Diese Abfrage wird in 4.6 detailliert durchgegangen.

Abfrage 20:

Dynamikstufe 0

Anzeige aller Sehenswürdigkeiten eines oder mehrerer Kategorien auf der Karte. Wie bei Abfrage 16 handelt es sich hier um eine statische Abfrage wo alle Informationen beim Laden der Seite abgerufen werden und deren Sichtbarkeit mit JavaScript eingeschränkt wird.

4.5 Abfrage 10 im Detail

Die Funktion `selectionChange(form)` im `<head>` Element wird als `onchange` Event von jedem der drei Dropdown Menus ausgelöst und löst ein neues Laden der Seite mit den getätigten Eingaben aus. Dabei werden nur Werte für die Haltestellen weitergegeben wenn dort auch welche ausgewählt wurden.

Eine Besonderheit hier ist das wenn die Linie gewechselt wird die bisher ausgewählten Haltestellen erhalten bleiben wenn sie von der neuen Linie auch abgefahren werden. Ein Beispiel dafür wäre Gare du Nord und Gare du Midi, welche sowohl von Linie 3 und Linie 4 abgefahren werden. Es wird aber dennoch beim Wechsel von Linie 3 zu 4 die neue Zeit angegeben! Sollte das Beibehalten der ausgewählten Stationen (sofern sie von der neu ausgewählten Linie ebenfalls abgefahren werden) nicht wie von uns als Feature sondern als Bug angesehen werden müsste man beim Ändern der Linie die Werte für die Haltestellen zurücksetzen.

```
function selectionChange(form){
      var r_id=form.r_id.options[form.r_id.options.selectedIndex].value;
      var stop1=form.stop1.options[form.stop1.options.selectedIndex].value;
      var stop2=form.stop2.options[form.stop2.options.selectedIndex].value;
```

```
        var location="query10.php?route=" + r_id;
        if (stop1 != "") location += "&stop1=" + stop1;
        if (stop2 != "") location += "&stop2=" + stop2;
        self.location=location;
}
```

In dieser wie auch in allen anderen Abfragen wird am Anfang vom `body` mit der php Methode **`include()`**die für alle Seiten identische Kopfzeile `header.html` und am Ende vom `body` die für alle Seiten identische Fußzeile `footer.html` gelanden.

```
include("header.html");
```

Anschließend wird mittels `dbConnect.php` eine Verbindung mit der Datenbank aufgebaut.

```
require_once("./dbConnect.php");
```

Danach wird geprüft ob Parameter für die Route und die beiden Stationen vorhanden sind und, wenn ja, werden diese geladen. Wenn kein Parameter für die Route vorhanden ist wird diese auf "1" gesetzt.

```
@$r_id=$_GET['route'];
if (@$r_id != null) {
    $r_id = $_GET['route'];
}
else {
    $r_id = 1;
}
@$s_id1=$_GET['stop1'];
if (@$s_id1 != null) {
    $s_id1 = $_GET['stop1'];
}
@$s_id2=$_GET['stop2'];
if (@$s_id2 != null) {
    $s_id2 = $_GET['stop2'];
}
```

Es folgt ein **`echo`** Eintrag für den HTML Code der Abfrage bis zum Dropdown Menu für die Linien.

```
echo
    '<table style="margin: 0 auto;"  width="100%"  id="main"
        border="0" cellpadding="0" cellspacing="0">
    <tr>
    <td valign="top" id="sp1">
    <h2>Abfragen - Haltestellen</h2>
    <b>Abfrage 10:</b>
    <form action="" method="get">
    <div>
        Was ist die Entfernung und Fahrzeit zwischen zwei Haltestellen
        einer Line?
        <br /><br />
```

```
            <select name="r_id" size="1"
                  onchange="selectionChange(this.form)">
    ';
```

Für das Linien Dropdown Menu werden alle in der Datenbank vorhandene Linien ausgelesen und zum Menu hinzugefügt. Die im Parameter `$r_id` gespeicherten Linie wird dabei als selektiert definiert.

```
$query = "SELECT DISTINCT r_id FROM routestop";
$result = mysql_query($query) OR die("Mist! ". mysql_error());
while ($row = mysql_fetch_assoc($result)) {
    if ($row['r_id']==$r_id) {
        echo "<option selected='selected'
        value='".$row['r_id']."'>Linie ".$row['r_id']."</option>";
    }
    else
    {
        echo "<option value='".$row['r_id']."'>Linie
        ".$row['r_id']."</option>";
    }
}
```

Die Haltestellen Dropdown Menüs werden ähnlich erzeugt, es gibt bei diesen aber eine kleine Besonderheit.
Durch die erste leere `if` Bedingung wird eine Station, die im anderen Dropdown Menu schon ausgewählt ist, nicht mit in die Liste der auswählbaren Stationen mit aufgenommen.

```
$query = "SELECT s_id, name, tim, dist FROM tramstop NATURAL JOIN
        routestop WHERE r_id = ".$r_id;
$result = mysql_query($query) OR die("Mist! ". mysql_error());
while ($row = mysql_fetch_assoc($result)) {
    if ($row['s_id']==$s_id2) {
    }
    else if ($row['s_id']==$s_id1) {
        echo "<option selected='selected'
              value='".$row['s_id']."'>".$row['name']."</option>";
    }
    else
    {
        echo "<option
              value='".$row['s_id']."'>".$row['name']."</option>";
    }
}
```

Der php Code für das Dropdown Menu der zweiten Station ist abgesehen davon das `$s_id1 und $s_id2` dort vertauscht sind und der SQL Query nicht erneut ausgeführt wird identisch.
Wenn bei der ersten und/oder die zweiten Haltestelle keine Station ausgewählt ist wird unter den Dropdown Menüs eine Aufforderung zur Auswahl angezeigt.

```
    if ($s_id1 == null || $s_id2 == null){
            echo 'Wählen sie eine Linie und zwei Stationen aus.';
        }
```

Ist in beiden Haltestellen Dropdown Menüs eine Haltestelle ausgewählt wir aus den Resultat des SQL Query für die Haltepunkte die dort mit abgerufene Zeit und Entfernung für die beiden Stationen ausgelesen

```
else {
            $stop1 = "";
            $dis1 = "";
            $tim1 = "";
            $stop2 = "";
            $dis2 = "";
            $tim2 = "";

            mysql_data_seek($result, 0);
            while ($row = mysql_fetch_assoc($result)) {
                if ($row['s_id']==$s_id1) {
                    $stop1 = $row['name'];
                    $dis1 = $row['dist'];
                    $tim1 = $row['tim'];
                }
                else if ($row['s_id']==$s_id2) {
                    $stop2 = $row['name'];
                    $dis2 = $row['dist'];
                    $tim2 = $row['tim'];
                }
            }
```

und aus diesen Werten die Zeitdauer und der Abstand zwischen ihnen berechnet und angezeigt.

```
            echo 'Die Entfernung und Fahrzeit zwischen den Stationen
            <b>'.$stop1.'</b>       und <b>'.$stop2.'</b> der <b>Linie
            '.$r_id.'</b> beträgt   <b>'.ABS(ROUND(($dis1-$dis2)*1000,-2)).'
            Meter</b> und     <b>'.ABS(ROUND($tim1-$tim2)).' Minute';
            if (ABS(ROUND($tim1-$tim2)) > 1) {
                  echo 'n';
            }
            echo '</b>.';
```

4.6 Abfrage 19 im Detail

Im Gegensatz zu Abfrage 10 ist der JavaScript Anteil in Abfrage 19 recht hoch. Anfangs werden im `head` die vier benötigten externen JavaScript Programm Bibliotheken geladen: Die der Google Maps V3 API, die von gmap3 benötigten jQuery 1.4.4 API, die gmap3 APi für das Gros der Kartenlogik und die Infobubble API um bessere Infofenster zu erstellen.

```
<script type="text/javascript"
    src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="js_libraries/jquery/jquery-
    1.4.4.js"></script>
<script type="text/javascript"
    src="js_libraries/gmap3/gmap3.js"></script>
<script type="text/javascript"
    src="js_libraries/infobubble/infobubble.js"></script>
```

Danach werden die globalen Variablen definiert: Die Koordinaten des Zentrums beim Start der Karte, die Anfangs, minimale und maximale Zoomstufe und die Variablen unter denen später die Koordinaten eines Rechtsklicks auf die Karte gespeichert werden sollen.

```
var latcenter = 50.85034;
var lngcenter = 4.35172;
var zoomstart = 14;
var zoommin = 11;
var zoommax = 18;
var lat;
var lng;
```

Als nächstes wird ein InfoBubble Objekt mit den gewünschten Parametern (Schattenart, Polsterung, Hintergrundfarbe....) erstellt.

```
var infobubble = new InfoBubble({
    shadowStyle: 1,
    padding: 6,
    backgroundColor: "rgb(220,220,220)",
    borderRadius: 4,
    arrowSize: 10,
    borderWidth: 1,
    borderColor: "#2c2c2c",
    disableAutoPan: true,
    hideCloseButton: true,
    arrowPosition: 30,
    arrowStyle: 2
});
```

Die Funktion `addMarkers` enthält die eigentlich Programmlogik des JavaScript Teils dieser Abfrage. Sie hat 4 Parameter:

`latlngM` sind die Koordinaten des Punktes wo der Nutzer einen Rechtsklick getätigt hat

`latlngP` sind die Koordinaten des Zielpunktes

`nameP` ist der Name und

`iconP` das Icon des Zielpunktes

```
function addMarkers(latlngM,latlangP,nameP,iconP){
```

Wurden der Funktion Koordinaten des Rechtsklicks gegeben wird auf diese ein Marker gesetzt und die Karte auf ihn zentriert.

```
if (latlngM != null) {
    $("#map").gmap3({
        action: "addMarker",
        latLng: latlngM,
        map:{
            center: true
        }
    });
}
```

Wenn Koordinaten, Name und Icontyp definiert sind

```
if (latlangP != "", nameP != "", iconP != "") {
```

wird zuerst ein Icon mit dem Typ entsprechenden Symbol erstellt,

```
var poi_image = new google.maps.MarkerImage("grafics/" +
  iconP +".png", null, null, new google.maps.Point(13,
  13));
```

dann der Karte ein Marker mit diesem Icon auf der in `latlangP` angegebenen Position hinzugefügt

```
$("#map").gmap3({
    action: "addMarker",
    latLng: latlngP,
    marker:{
        options:{
            icon: poi_image
        },
```

und diesem Marker ein MouseOver und MouseOut Event für das Öffnen und Schließen einer Infobubble mit `nameP` als Inhalt hinzugefügt.

```
events:{
    mouseover: function(marker, event){
```

```
                        var map = $(this).gmap3("get");
                        infobubble.open(map, marker);
                        infobubble.setContent(nameP);
                    },
                    mouseout: function(){
                        infobubble.close();
                    }
                }
            }
        },
```

Danach wird als zweite Aktion im `$("#map").gmap3({` Aufruf der Karte eine Route zwischen `latlngM` und `latlngP` hinzugefügt. Als Fortbewegungsart wird "zu Fuß" angegeben.

```
        {
            action:'getRoute',
            options:{
                origin:latlngM,
                destination:latlngP,
                travelMode:
                      google.maps.DirectionsTravelMode.WALKING
            },
            callback: function(results){
                if (!results) return;
                $("#map").gmap3({
                    action:'addDirectionsRenderer',
                    options:{
                        preserveViewport: true,
                        draggable: false,
                        directions:results,
                        suppressMarkers:true
                    }
                });
            }

        });
    }
}
```

Die Funktion `reloadForm()` erledigt das neu Laden das Forms nach Benutzereingaben, also entweder dem Rechtsklick auf die Karte um einen Standort festzulegen oder die Auswahl einer POI Klasse im Dropdown Menu.

```
function reloadForm(){
    var kIDselect = document.getElementById("kID");
    var kID =
          kIDselect.options[kIDselect.options.selectedIndex].value;

    var location = "query19.php?";
    if (lat != null && lng != null) location += "lat=" + lat +
          "&lng=" + lng;
    if (kID != "") location += "&kID=" + kID;
    self.location = location;
}
```

Die letzte JavaScript Funktion im Kopfteil ist der Konstruktor. Diese Funktion wird automatisch beim Laden der Seite ausgeführt.

```
$(function(){
```

Dieser Teil des Konstruktors initialisiert die Karte mit den vorher definierten Werten für das Zentrum und das Anfangs-Zoomlevel. Es wird auch bestimmt welche Kontrollen verfügbar sind.

```
    $("#map").gmap3({
        action: 'init',
        options:{
            center:[latcenter, lngcenter],
            zoom: zoomstart,
            streetViewControl: false,
            overviewMapControl: true,
            scaleControl: true,
            overviewMapControlOptions: {opened: true}
        },
```

Danach werden die Events der Karte definiert, hier die Begrenzung des Zoomlevels und das Auslesen der Kartenkoordinaten und neu Laden des Forms bei Rechtsklick.

```
        events: {
            /* Begrenzt das Zoomlevel */
            zoom_changed: function(map, event){
                if (map.getZoom() < zoommin) map.setZoom(zoommin);
                if (map.getZoom() > zoommax) map.setZoom(zoommax);
            },
            rightclick:function(map, event){
                lat = Math.round(event.latLng.Pa *
                      1000000)/1000000;
                lng = Math.round(event.latLng.Qa *
                      1000000)/1000000;
                reloadForm();
            }
        }
    });
```

Danach werden die Legendenframes über den Frame der Karte gelegt. Die Legende ist bei allen Karten mittels Klick auf die Leiste mit dem vertikalen "Legende" Text minimierbar und wieder maximierbar.
Die funktioniert aber nicht immer perfekt, es kann vorkommen das hinter der legende anstatt der Karte teilweise nur die Hintergrundfarbe ist.

```
        $("#legend").insertBefore("#map");
        $("#legendMax").insertBefore("#map");
        $("#legend").click(function(event) {
            $("#legendMax").toggle();
        });
    });
```

Dies schließt den `script` Teil vom `head` ab. Als nächstes folgt der `body` mit dem `php` Teil. Als erstes wird dort wie in Abfrage 10 der Header eingefügt und eine Verbindung mit der Datenbank erstellt.

```
include("header.html");

require_once("./dbConnect.php");
```

Darauf folgt ein **`echo`** mit dem HTML bis zu Dropdown Menü. Darauf folgend werden die Attribute für die ID der POI Kategorie und des Breiten und Längengrades des Rechtsklicks ausgelesen, sofern vorhanden.

```
@$kID=$_GET['kID'];
@$lat=$_GET['lat'];
@$lng=$_GET['lng'];
```

Danach werden aus der Datenbank alle POI Kategorien abgefragt, damit das Dropdown Menü gefüllt und, sofern eine der Kategorien vorher ausgelesen würde, diese als selektiert markiert.

```
$query = "SELECT kind_id, kind FROM poi_codes";
$result = mysql_query($query) OR die("Mist! ". mysql_error());
while ($row = mysql_fetch_assoc($result)) {
    if ($row['kind_id']==$kID) {
        echo "<option selected='selected'
      value='".$row['kind_id']."'>".$row['kind']."</option>";
    }
    else
    {
        echo "<option
      value='".$row['kind_id']."'>".$row['kind']."</option>";
    }

}
```

Sofern alle drei Werte (Breitengrad, Längengrad, POI Kategorie) ausgelesen werden konnten wird aus diesen der naheste POI dieser Kategorie zu diesen Koordinaten bestimmt.

```
if (@$lat != null && @$lng != null) {
    if (@$kID != ""){
        $query =
            "SELECT
                distances.name, distances.lat, distances.lng,
                  distances.kind, distances.icon, distances.distance
            FROM(
                SELECT
                    poi.name, poi.lat, poi.lng, poi_codes.kind,
                        poi_codes.icon, ROUND(
                        (
                            (
                                Acos(
                                    Sin( poi.lat * Pi() / 180 ) *
                                    Sin( ".$lat." * Pi() / 180 )
                                    +
                                    Cos( poi.lat * Pi() / 180 ) *
                                    Cos( ".$lat." * Pi() / 180 ) *
                                    Cos(( poi.lng - ".$lng." ) * Pi()/180)
                                )
                            ) * 180 / Pi()
                        ) * 60 * 1.1515 * 1.609344 * 1000
                    ,0) AS distance
                FROM
                    poi natural join poi_codes
                WHERE
                    kind_id = ".$kID."
                GROUP BY
                    distance
            ) distances
            LIMIT
                1";
      $result = mysql_query($query) OR die("Mist! ". mysql_error());
```

Das Resultat wird in der Legende angezeigt

```
  while ($row = mysql_fetch_assoc($result)) {
    echo
        '<br /><br />
        Die näheste Sehenswürdigkeit der Kategorie
            "<b>'.$row['kind'].'</b>" ist <b>'.$row['name'].'</b>.<br />
        Die Entfernung beträgt <b>'.round($row['distance'],-1).' </b>
            Meter.
```

und aus den ausgelesenen Parameter werden in einem JavaScript Teil die Marker und die Route zwischen ihnen zur Karte hinzugefügt. Davor muss aber die Seite erst fertig geladen sein, ansonsten wird die Funktion `addMarkers` zum hinzufügen der Marker zur Karte ausgeführt bevor die Karte existiert. Dies wird mittels `$(document).ready(function() { ... });` sichergestellt. Alle Anweisungen in der

geschweiften Klammer werden erst ausgeführt nachdem die Seite fertig geladen ist.

```
        <script type="text/javascript">
        //<![CDATA[
            $(document).ready(function() {
                lat = '.$lat.';
                lng = '.$lng.';
                var latlngM = [lat, lng];
                var latlngP = ['.$row['lat'].', '.$row['lng'].'];
                var nameP = "'.$row['name'].'";
                var iconP = "'.$row['icon'].'";
                addMarkers(latlngM,latlngP,nameP,iconP);
            });
        //]]>
        </script>
```

Ist hingegen keine POI Kategorie ID vorhanden, sondern nur der Längen- und Breitengrad, wird stattdessen nur der Marker für die Position des Rechtsklicks gesetzt. Auch hier wird der Inhalt erst nach dem vollständigen Laden der Seite ausgeführt.

```
else {
    echo
        '<script type="text/javascript">
        //<![CDATA[
            $(document).ready(function() {
                lat = '.$lat.';
                lng = '.$lng.';
                var latlngM = [lat, lng];
                addMarkers(latlngM,"","","");
            });
        //]]>
        </script>';
}
```

Abschließend wird wie in allen Abfragen vor den Ende des `body` der Inhalt vom `footer.html` Element der Webseite hinzugefügt.

```
include("footer.html");
```

5 Fazit

Man kann aus OSM Daten viele Informationen extrahieren, allerdings ist die Dichte und Qualität der Daten stark schwankend. Daher sind sie nur bedingt als primäre Datenquelle in professionellen Anwendungen geeignet. Es spricht nichts gegen ihre Nutzung als sekundäre Datenquelle zur Ergänzung, allerdings muss dann ihre gegebenfalls vorhandene unterschiedliche Datendichte berücksichtigt werden. Auch sind Fehler in den OSM Daten nicht auszuschließen, daher ist eine Kontrolle von kritischen Daten unbedingt erforderlich.

Zur Speicherung von Sonderzeichen in der SQL Datenbank wurde keine befriedigende geräteunabhängige Lösung gefunden. Je nach Host Rechner kann eine unterschiedliche Ladedatei notwendig sein.

Die Verwendung von Stylesheets erleichtert die einheitliche Strukturierung von HTML Seiten erheblich, es muss allerdings beachtet werden, dass Browser die dort enthaltenen Informationen zum Teil unterschiedlich interpretieren.

Bei der Verwendung von PHP, JavaScript und HTML zusammen in der gleichen Datei sollte es vermieden werden diese zu stark zu mischen, da dann die Übersichtlichkeit schnell verloren gehen kann.

6 Bewertung der Arbeitsanleitung

In der Anleitung gibt es auf Seite 15 bei "Dokumentation der Anleitung" einen kleinen Fehler. Der Kommandozeilenaufruf zum Erstellen von lst Dateien

```
\T <path>\<file>.lst
\T
```

funktioniert so nicht. Das letzte "\T" muss in "\t" umgewandelt werden.