Maier, Gunther

Article

OpenStreetMap, the Wikipedia Map

REGION

Provided in Cooperation with:
European Regional Science Association (ERSA)

Suggested Citation: Maier, Gunther (2014) : OpenStreetMap, the Wikipedia Map, REGION, ISSN 2409-5370, European Regional Science Association (ERSA), Louvain-la-Neuve, Vol. 1, Iss. 1, pp. R3-R10,
http://openjournals.wu.ac.at/ojs/index.php/region/article/view/70

This Version is available at:
https://hdl.handle.net/10419/110935

REGION
The Journal of ERSA
Powered by WU

Volume 1, Number 1, 2014, R3–R10 journal homepage: region.ersa.org

OpenStreetMap, the Wikipedia Map

Gunther Maier[1]

[1] WU, Vienna University of Economics and Business, Vienna, Austria (email: gunther.maier@wu.ac.at)

Received: 9 December 2014/Accepted: 11 December 2014

Abstract. This paper presents OpenStreetMap and closely related software as a resource for spatial economic research. The paper demonstrates how information can be extracted from OpenStreetMap, how it can be used as a geographical interface in web-based communication, and illustrates the value of the tools by use of a specific application, the WU campus GIS.

1 Introduction

The open digital map, OpenStreetMap (OSM), is potentially a very valuable resource for spatial economic research. Currently (early December, 2014), the whole dataset is 39GB in size, and contains 2.6 billion nodes. All of this information is publicly available and individual users – who drive forward the project in a shared effort – have collected most of it.

The digital map (www.openstreetmap.org, see figure 1) is the central resource around which a range of open and commercial applications, toolsets and services has developed. For an overview, see the list of OSM-based services. Here, we will sketch a few of its research related options.

2 OSM as a source of information

Increasingly, analyses in regional science deal with disaggregated, point data information. The rapid growth in spatial econometrics applications has raised issues about neighborhood characteristics, accessibility of certain sites, and spatial proximity, among others. OSM contains information about many such points of interest through geocoded locations. Although there is no guarantee about the quality and completeness of the information, the quality of the project's data is quite good in many areas because of its reliance on voluntary contributions. In any case, when using OSM in a specific application, one should always validate that the information exists for the respective area in the required quality. One major advantage of OSM is that due to its many contributors, changes are usually reflected much faster than in more bureaucratic, alternative sources.

A number of servers provide a user interface for accessing the information in OSM. One of these is "mapquestapi.com", which communicates via the Xapi web service. The query URL typically supplies a bounding box for the map area to be queried and specifies the type of information requested. For example, the URL,

http://open.mapquestapi.com/xapi/api/0.6/node[amenity=restaurant][bbox=16.40,48.21,16.41,48.22]

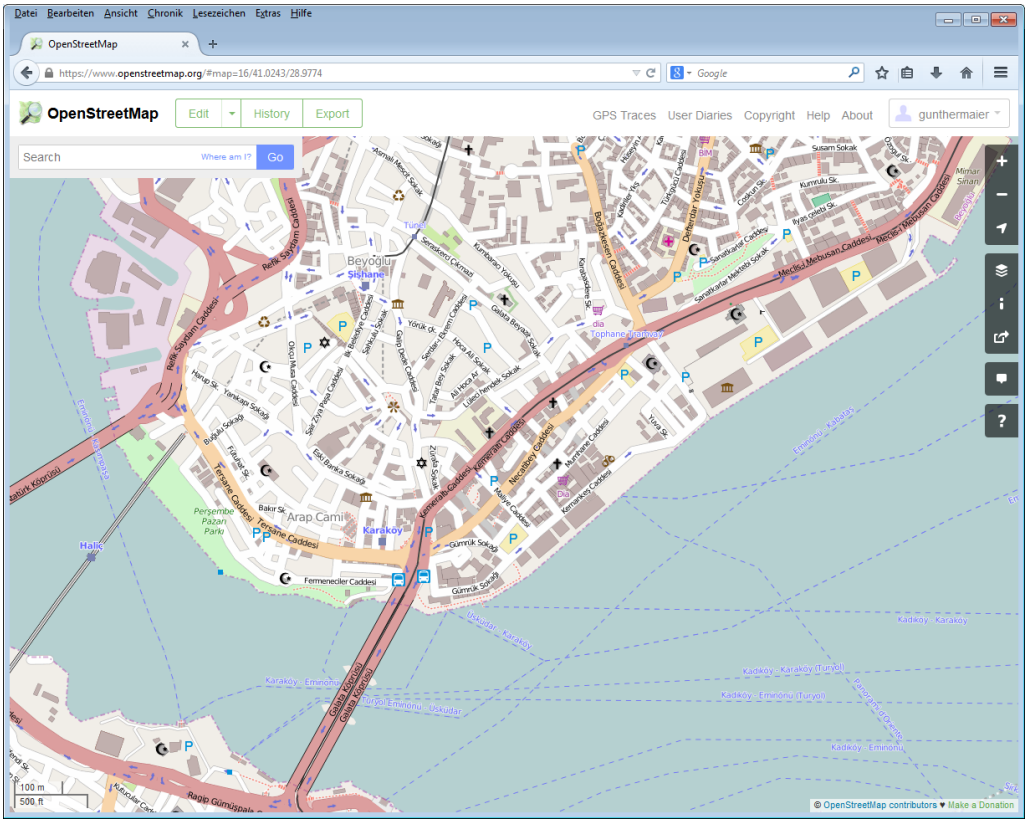

Figure 1: OpenStreetMap homepage

asks the server to return all the nodes that characterize a "restaurant" (identified by the tag "node[amenity=restaurant]") at and around the new campus of WU (specified by the bounding box "[bbox=16.40,48.21,16.41,48.22]"). The result is the XML-file shown in figure 2. For each restaurant, we get – among other information – its latitude and longitude, and then a list of "tags" that characterize the place. All returned nodes contain the tag "amenity=restaurant," because this is the tag we searched for. For some restaurants, we get little additional information; for others the list of tags is quite long and detailed. When a program issues such queries, their results can be processed directly by the software, used in analysis, stored in a database, or just saved to the hard-drive.

Many other servers and interfaces exist for such tasks. A well-documented alternative is Overpass. For an overview, see the OSM Wiki.

3 Use of OSM for geocoding and routing

The digital map contains all the information needed for geocoding and routing. For example, we could use the above-mentioned querying option to search for a specific address and extract the latitude-longitude coordinates from the respective result. A specialized OSM-based service for geocoding (finding latitude/longitude of an address) and reverse geocoding (finding an address from latitude/longitude) is Nominatim. It can be used via a webpage (nominatim.openstreetmap.org) or directly through a search request. A more detailed description of Nominatim appears in the Wiki.

For routing and calculating distances along a road network, development seems to concentrate on web-based services (see the list of OSM-based services mentioned above). We could not find any open server that allows for requesting routing services through an API. The project OSRM seems to be the closest: It offers open source software for setting up and running routing servers.

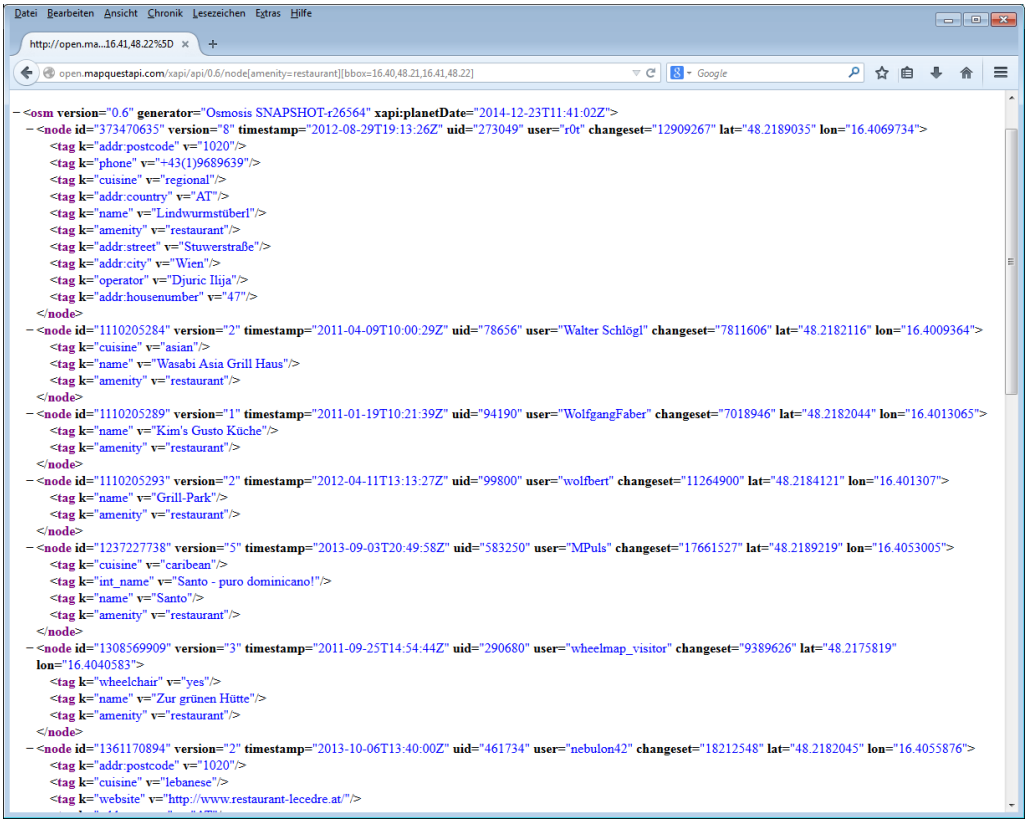

Figure 2: XML-file resulting from a query

4 OSM in a web-frontend

The digital map of OSM is edited and improved by thousands of registered users worldwide. Their edits are stored in the central database and made available to all other users. In research, however, we may have results that will not become features of the central map, but should be displayed just on top of the map. Two elements are necessary for such tasks: 1) a background map, and 2) a mechanism to place our results on top of this map. The first element is provided by renderings that are generated from the OSM database. These are picture files available over the Internet that can link together like tiles to form a base map of any location in the world. These base maps are available in different scales so that the user can zoom in and out.

The JavaScript library, OpenLayers, provides the second element; it is now available in version 3 and allows the user to combine a base map with user generated features, which are typically placed in transparent layers on top of the base map. Figure 3 gives an example of such a map[1]. It shows the city of Brest in France and its vicinity on an OSM-based map, and a small flag placed right underneath the city. In our web browser we can use the plus and minus icons in the upper left hand corner of the map to zoom in and out. While the size of the flag remains constant, its position on the screen is adjusted so that it will always remain on the same coordinates of the map.

Figure 4 shows the HTML page for this example, consisting of HTML and JavaScript code. There are three key elements in every such application:

1. To include the OpenLayer library. This is done in the call of the external JavaScript in the fifth line of code. There we include the library directly from the server "openlayers.org". This line guarantees that we can use the functionality of OpenLayers in the rest of the application.

[1] Use http://openjournals.wu.ac.at/ojs/index.php/region/author/downloadFile/70/186 to open this example in your browser.

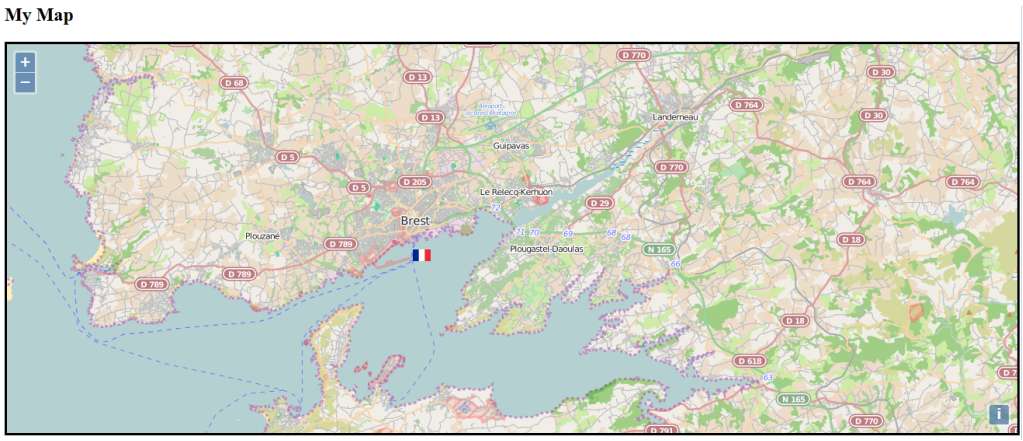

Figure 3: Map with icon generated from OSM

2. To reserve space for the map on the HTML page. This is done in the div statement in the tenth line of code. This div is given the ID "map". This ID will be used later to link the map to this HTML element. The style option of the div statement defines the map area as 500 pixels high, extending over the whole width of the screen and surrounded by a solid, three-pixel-wide, black border.

3. To define the content of the map. This is done in the script statement in the body of the HTML code (starting with <script . . . > in line 11 and ending with </script> in line 35). This JavaScript code will be discussed below.

The code for creating the map is best explained from bottom to top. Here we can only present the general logic. For a more comprehensive presentation of JavaScript and OpenLayers, see the respective webpages or specialized literature (e.g., Flanagan 2001; Crockford 2008; Santiago 2012, 2015; Di Lorenzo and Allegri 2013; Bennett 2010; Gratier et al. 2015).

The last statement in the JavaScript block defines a new ol.Map object with the name "mymap". Its target statement links the map to the previously defined div with the ID "map". As we can see from the layers statement, the map has two layers. The first is an ol.layer.Tile object that is called directly from the OSM server (source: new ol.source.OSM). This is all it needs to link in the tiles of the base map. The second layer is named "vectorLayer," and has been defined earlier in the code block. This layer adds the image of the French flag.

The view statement defines what shall be visible when the page is loaded. The center of the page should be at longitude -4.40 and at latitude 48.38. Since the base map uses projection "EPSG:3857" rather than the latitude-longitude-coordinates, we have to convert our specifications through "transform". By setting the zoom level to 11, we request a rather detailed view of the map.

The vectorLayer for the map statement is defined in the code block starting with "var vectorLayer." This statement only says that the vectorLayer should be created according to the specifications given in the variable "iconFeature." When the variable "iconFeature" is created in the first lines of the JavaScript block, we define that the flag should be placed at coordinates -4.48 / 48.37. Instead of using one statement, we use two: First, we define the position for the icon in a variable "iconGeometry". Again, we use transform to convert to the coordinate system of the map. Second, we use this variable for defining the variable "iconFeature." In another statement starting with "iconFeature.setstyle . . ." we load the image of the flag from the server and link it as an image to the variable "iconFeature."

Going through the JavaScript block from top to bottom, we first define the features of the vector layer, then create the vector layer, and finally, add it to the map for display.

So far, we have communicated only from the program to the user (by placing the flag

```
<!doctype html>
<html lang="en">
  <head>
    <link rel="stylesheet" href="http://openlayers.org/en/v3.0.0/css/ol.css" type="text/css">
    <script src="http://openlayers.org/en/v3.0.0/build/ol.js" type="text/javascript"></script>
    <title>OpenLayers 3 example</title>
  </head>
  <body>
    <h2>My Map</h2>
    <div id="map" style="height:500px; width: 100%; border: 3px solid black"></div>
    <script type="text/javascript">

      var iconGeometry = new ol.geom.Point([-4.48, 48.37]).transform('EPSG:4326','EPSG:3857');

      var iconFeature = new ol.Feature({
        geometry: iconGeometry
      });

      iconFeature.setStyle(new ol.style.Style({
        image: new ol.style.Icon({src: '23px-Flag_of_France.svg.png'})
      }));

      var vectorLayer = new ol.layer.Vector({
        source: new ol.source.Vector({
          features: [iconFeature]
        })
      });

      var mymap = new ol.Map({
        target: 'map',
        layers: [new ol.layer.Tile({source: new ol.source.OSM}), vectorLayer],
        view: new ol.View({center: ol.proj.transform([-4.40, 48.38],'EPSG:4326','EPSG:3857'),zoom:11})
      });

    </script>
  </body>
</html>
```

Figure 4: HTML and JavaScript code using OpenLayers to generate the map of figure 3

on the map). Of course, the JavaScript program can also receive input from the user and act on it. Suppose we want to be able to place the flag at another place on the map. A mouse click on the map should move the flag icon to this location. This functionality is very easy to implement. We only need to add the following lines of code to the end of the JavaScript block:

```
mymap.on('singleclick', function (evt) {
    iconGeometry.setCoordinates(evt.coordinate);
});
```

With this code, we add an event-handler to the map, which is executed whenever a single mouse click (i.e., "singleclick") occurs. This event reports among other things the coordinates of the mouse click; these are in variable "evt.coordinate". With the "setCoordinates" method of "iconGeography," the coordinates of the vectorLayer are set equal to those of the mouse click.

5 OSM-based web applications

The licensing used by OSM, OpenLayers and related tools allows developers to use them in their own applications, even when they are commercial. In concluding this discussion, we want to sketch one such example of an OSM-based application: WU's campus GIS. GOMOGI, a small Austrian startup company, designed it with the intention to help employees and visitors find their way around the newly built campus of the university.

As figure 5 shows, the campus GIS uses an OSM base map and overlays it with floor plans for all the floors of the campus buildings. The vertical bar on the right allows the user to switch between the storeys.

Because the tool is linked to the office assignment database, it can offer search and routing functions. When supplied the name of an employee, for example, the campus GIS marks the respective office location on the respective floor layer (see figure 6). This location can be selected as a start, end, or mid point of a route via the popup menu. The

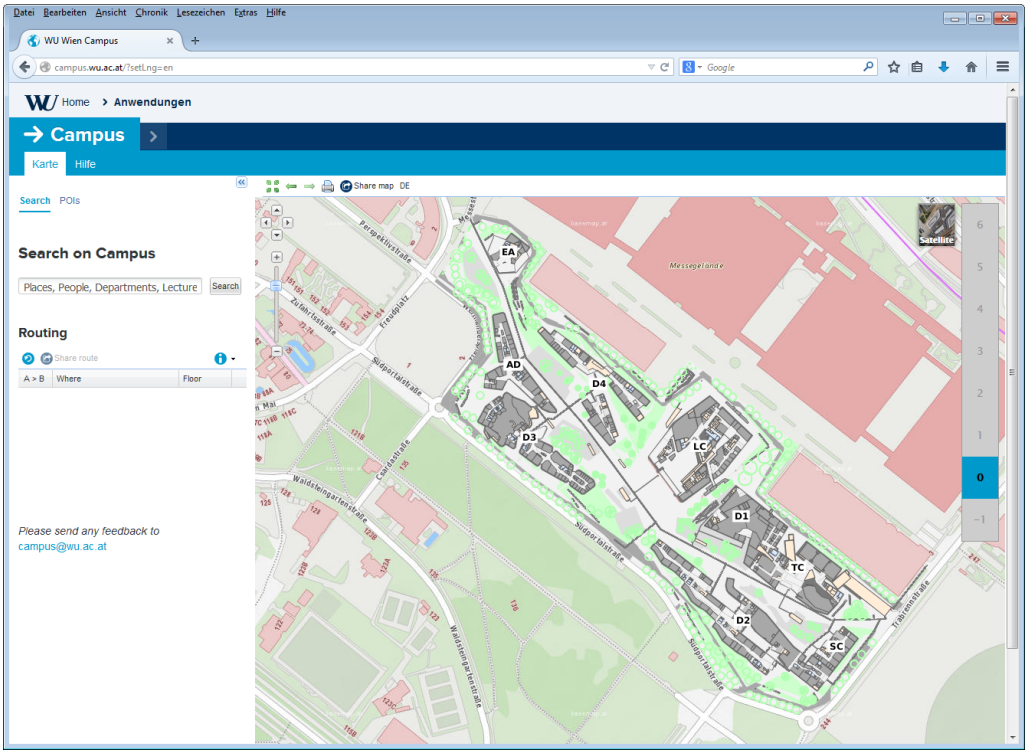

Figure 5: The WU campus GIS

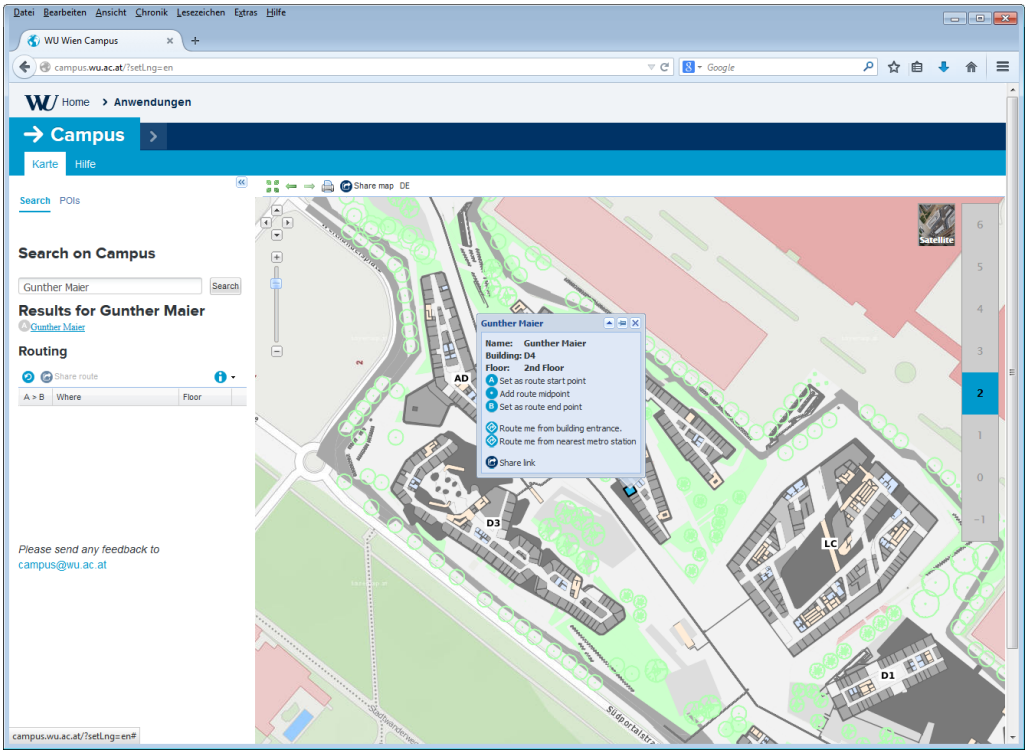

Figure 6: Search result in the WU campus GIS

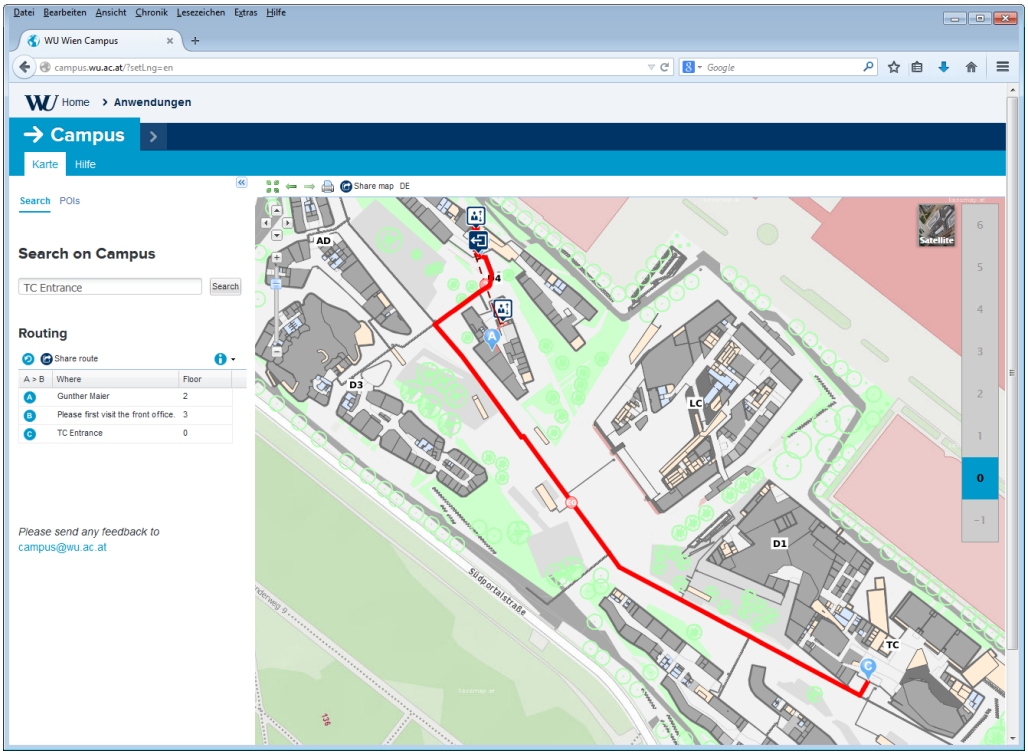

Figure 7: Routing information in the WU campus GIS

routing information provided by the tool links up the different floor layers if necessary and can be investigated floor by floor (see figure 7).

6 Conclusion

As was demonstrated in this short paper, OpenStreetMap is a valuable tool for spatial economic research. It can help with geolocation and routing tasks and offers a wealth of open information about the location of facilities, offices, points of interest and such. In combination with OpenLayers, OpenStreetMap can also serve as a geographical interface for web-based communication of research results or as a tool for data collection. As the last section has demonstrated, the tools can also be used for the development of professional GIS-oriented services.

Links mentioned in the text

Service	Weblink
OpenStreetMap Homepage	http://www.openstreetmap.org
List of OSM based services	http://wiki.openstreetmap.org/wiki/List_of_OSM_based_Services
Mapquest search interface	http://open.mapquestapi.com/xapi/
Overpass search interface	http://wiki.openstreetmap.org/wiki/Overpass_API
OSM wiki	http://wiki.openstreetmap.org/wiki/Main_Page
Nominatim geocoding interface	http://nominatim.openstreetmap.org/
Nominatim wiki	http://wiki.openstreetmap.org/wiki/Nominatim
Open Source Routing Machine	http://project-osrm.org/
WU campus GIS	http://campus.wu.ac.at

References

Bennett J (2010). OpenStreetMap, Packt Publishing.

Crockford D (2008) JavaScript: The Good Parts, O'Reilly Media / Yahoo Press.

Di Lorenzo A, Allegri G (2013) Instant OpenLayers Starter, Packt Publishing.

Flanagan D (2001) JavaScript: The Definitive Guide, 4th Edition, O'Reilly Media.

Gratier T, Spencer P Hazzard E (2015) OpenLayers 3 Beginner s Guide, Packt Publishing.

Santiago A (2012) OpenLayers Cookbook, Packt Publishing.

Santiago A (2015) The book of OpenLayers 3: Theory & Practice, Leanpub, https://leanpub.com/thebookofopenlayers3.