# Web Mapping Service for Mobile Tourist Guide

Nikolay Teslya
SPIIRAS
St.Petersburg, Russia
teslya@iias.spb.su

*Abstract*—**Development of context-aware systems depends on the context definition and the context components. The entity location was always one of the main context components. For using location, a map service that provides possibilities of working with geographical information and showing results on the map is needed. The paper describes an implementation of a web mapping service for a mobile tourist guide, which is a context-aware service developed for supporting travelers before, during and after the trip. The mapping service provides possibilities of map showing, routing, geocoding, and has minimal license restrictions. The paper provides an analysis of existing web mapping systems such as Google, Microsoft, Yandex and describes implementation of free web mapping service for the mobile tourist guide based on OpenStreetMap, Leaflet, PostGIS, pgRouting, and Nominatim projects.**

## I. Introduction

Nowadays there are many mobile applications and services developed for information processing with possibilities to consider the context. The definition of the context always includes the location of the considered object. In particular, A.K. Dey et al. provided the most detailed definition of information system context, in 2001. They define it as any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [1]. The examples of location-aware applications are navigation applications, tourist guides, advertisement applications, sport trackers and others. All these applications can determine the user location by using signals from GPS/GLONASS satellites as well as signals from the mobile network base stations. After the information has been processed, results should be shown on the map. For this purpose, the composition of map-based applications is used. This composition includes geoinformation sources, applications for map rendering, user interfaces for map showing, and applications for geoinformation processing. In the case of location-aware application the geoprocessing module can implement the following functions: routing for the high amount of routes, routes showing with additional information, such as start and end points, moving time, route length, geocoding and reverse geocoding, and others.

This paper describes implementation of the web mapping service for the mobile tourist guide [1], [2]. The mobile tourist guide aims to provide a comprehensive up-to-date information and assistance in the travel-related decision making, especially during the trip to enrich tourist experiences. The system accumulates tourist's interests and context-related information and searches for information about points of interest (POI), attractions, and available transportation means based on this information [2], [3]. Transportation means include public transport, taxis, and ridesharing in the region. For attractions and POI searching, routing and visualization of results in the mobile tourist guide, a web mapping service that satisfying the following requirements is needed: location and routing support, no limitations for route requests amount, address searching support, maximal coverage area, and data accuracy.

The rest of the paper is structured as follows. Section 2 provides description and comparison of the main well-known web mapping services: Google Maps, Yandex Maps, Microsoft Bing, and OpenStreetMap with CloudMade. Section 3 gives description of the OpenStreetMap-based instruments for geographic information processing and map rendering with the additional information on the map. Section 4 provides descriptions of the routing and geocoding utilities. Section 5 shows the implementation of web mapping service for mobile tourist guide. Section 6 describes performance of the built service based on the routing time evaluation.

## II. Web geoinformation services

The finding of the accurate and detail geographic information is not a real problem nowadays. Major world Internet-searching corporations additionally provide map services that allows to link search results to the place on the Earth. These services are provided by, for example, Google and Microsoft. Also, Yandex provides map service on the Russian and post-USSR Internet-market. All services provide high quality map with many additional functions, but there are serious limitations to the maps usage such as copyrights and restrictions on servers loading.

Besides proprietary services there is a free map provider called OpenStreetMap. OpenStreetMap is a collaborative, Wikipedia-like project to create a free editable map of the world. The data about the roads are usually obtained from the tracks recorded by GPS-receivers and imported by special programs, which further to convert it into the OSM geodata. The community members add additional information such as buildings, points of interests, road signs, traffic lights, etc. This project provides only map data, without any additional services, but there are many applications, developed by OpenStreetMap community, that can be used to analyze and manipulate these data.

All services can be used as a geographic information system in different projects. Each service provides API for developers that allow including map into applications, web pages, manipulating data on the map, showing additional information over the main map layer, etc. There are some benefits and restrictions for using the services. A comparative overview of main map providers is presented below. The comparison has been made by the following characteristics that are important for the mobile tourist guide application:

- *Coverage* describes geographical area provided by service, data precision and detail.
- *Routing* describes functions and API provided by service for routing and navigation.
- *Geocoding* is the process of finding associated geographic coordinates (often expressed as latitude and longitude) from other geographic data, such as street addresses, or ZIP codes (postal codes). Reverse geocoding is the process of converting geographic coordinates into a human-readable address. This characteristic describes functions and API provided by service for geocoding and reverse geocoding.
- *Restrictions and limits* describes licensing and limitation of service usage.

## *A. Google Maps*

*1) Coverage:* All world. The accuracy and detail of the data are different in regions. For example, whole USA map has a lot of details and very high accuracy, but Russia is detailed only in cities. All text information, such as street and object names, is presented with the current UI language, local language and with Latin transliteration. Routing is available in the half of the presented countries. The detail information about coverage and functions in different countries is available in the table of coverage, provided by Google Code Page [4].

*2) Routing:* Google routing API (Google calls it directions API) provides search for directions for several modes of transportation, include transit, driving, walking or cycling using an HTTP request. Directions may specify start point, end point and waypoints either as text strings (e.g. "St. Petersburg, Russia" or "Darwin, NT, Australia") or as latitude/longitude coordinates. This service is generally designed for calculating directions for static (known in advance) addresses for placement of application content on a map; this service is not designed to respond in real time.

For the real-time calculations, Google provides JavaScript Directions service and Distance Matrix service. Google's Distance Matrix service computes travel distance and journey duration between multiple origins and destinations using a given mode of travel. This service does not return detailed route information. It is provided by the Directions service based on the directions API.

*3) Geocoding:* The Google Geocoding API provides a direct way to access these services via an HTTP request to geocode on the server-side, but this service is not designed to respond in real time.

For the real-time geocoding on client-side, Google provides JavaScript geocoder class for geocoding and reverse geocoding dynamically from user input.

*4) Restrictions and limits:* The Directions API has the following limits:

- Users of the free API can request 2 500 directions per 24 hour period with up to 8 waypoints allowed in each request. Waypoints are not available for transit directions.
- Business customers can request 100 000 directions requests per 24-hour period with 23 waypoints allowed in each request. Waypoints are also not available for transit directions.

Directions API URLs are restricted to approximately 2000 characters, after URL Encoding.

The Distance Matrix service has the following usage limits:

- Maximum of 25 origins or 25 destinations per request;
- At most 100 elements (origins times destinations) per request.

Requests are also rate limited. If too many elements are requested within a certain time period, an OVER_QUERY_LIMIT response code will be returned.

In addition, the Directions API and JavaScript Directions service and Distance Matrix service may only be used in conjunction with displaying results on a Google map; using Directions data without displaying a map for which directions data was requested is prohibited.

The Google Geocoding API (server-side) has the following limits in place:

- Users of the free API can do 2 500 requests per 24 hour period.
- Business customers can do 100 000 requests per 24 hour period.

Client-side geocoding will not face a quota limit unless a batch of geocoding requests within a user session is performed.

Like the directions API, the Geocoding API may only be used in conjunction with a Google map; geocoding results without displaying them on a map are prohibited.

Complete details on allowed usage are described at the Maps API Terms of Service License Restrictions [5].

## B. *Microsoft Bing Map*

*1) Coverage:* All world. The accuracy and detail of the data are different in regions. The detail information about coverage and functions in different countries is available in the table of coverage provided by Bing MSDN page [6]. In addition, there is a coverage analyzer tool that allows to view and discover areas of Bing Maps coverage [7]. This tool is developed by OpenStreetMap community to compare coverage of Bing Maps and OSM Map. According to the tool, the best coverage has USA and Western Europe. Other world has a various precision and detail. All text information, such as street and object names, is presented with the current UI language and with Latin transliteration.

*2) Routing:* Bing Maps Routes API allows to create a route that includes two or more locations and to create routes from major roads using an HTTP request. Users can create driving or walking routes. Driving routes can also include traffic information. The URL used to get route is built by specifying a series of waypoints. A waypoint is a specified geographical location defined by longitude and latitude that is used for navigational purposes. The route includes information such as route instructions, travel duration, travel distance or transit information. A set of route points can also be requested.

*3) Geocoding:* Microsoft Bing Maps provides Location API for geocoding and reverse geocoding. It works over HTTP and HTTPS protocols. Works in all regions, presented in Bing Maps, but with varying precision.

*4) Restrictions and limits:* It can be specified up to 25 waypoints for a route. Each set of waypoints creates a separate route leg. Between any two waypoints, it can be up to 10 intermediate viaWayPoints. ViaWaypoints define the route path and do not create route legs.

One user may not exceed 100 000 total requests or 20 requests per asset, whichever is greater, of forward geocoding transactions, sessions or routing requests, all measured as an average over any 24 hour period [8].

## C. *Yandex Maps*

*1) Coverage:* All world. The map has high accuracy and detail in the Russia and post-USSR region, as well as Turkey. The accuracy and detail of the data are different in regions. All text information, such as street and object names, is presented with the Russian language and with Latin transliteration.

*2) Routing:* Routing is provided by JavaScript Yandex Map API. The route between a start point and end point is calculated automatically, and an unlimited number of *waypoints* and *throughpoints* can be set on the route. A route is built via the route function, which is passed an array of points for the route to go through, and additional route construction options, if necessary. A route can be built either with or without consideration for traffic jams.

*3) Geocoding:* The geocoder can be accessed either over the HTTP protocol or over using the JavaScript API. When accessing the geocoder over HTTP, the response may be formed either as an XML document in YMapsML format, or in JSON format. With the JavaScript Yandex.Maps API the search can be performed across the entire world map, or in a specified rectangular area. In addition, the rectangular area can apply strict limitations (the search will be performed only across objects inside the area), or flexible ones. In the latter case, the search will be performed across the entire map, but the closer a found object is to the center of the area, the higher it will be ranked in the results.

For reverse geocoding, the type of object to find can be specified (such as building, street, neighborhood, town, or subway station).

*4) Restrictions and limits:* The Yandex.Maps API cannot be used for fee-based cartographic services or services that restrict third-party access in any other way.

The number of geocoding queries is limited to 25 000 per Web site or mobile application a day. There is no information about routing API limitation. [9]

## D. *OpenStreetMap (OSM)*

*1) Coverage:* All world. The accuracy of data depends only on the community activity in the region.

*2) Routing:* OpenStreetMap provides only the data. The different libraries and applications, developed by OSM community, implement routing functions. The examples are pgRouting, GraphHopper, pyRoute, etc. [10].

*3) Geocoding:* Geocoding functions are also provided by applications, developed by community.

*4) Restrictions and limits:* OpenStreetMap data is free for everyone to use.

Due to the fact that map data is available for free download and use, developers can organize own services with all required features and designed for planned loading. As an example of such service, it can be mentioned one of the most powerful OSM-based platform for software development, called CloudMade. The company provides a range of web services, mobile and desktop libraries that let developers to build powerful geo applications quickly and easily. CloudMade makes extensive use of OpenStreetMap data to provide mapping services. It was founded in 2007 by OSM founder Steve Coast and long-term OSM contributer, Nick Black.

CloudMade geocoding API provides possibilities of address geocoding and reverse geocoding, local search, and finds geo objects by structured query. First 100 000 queries free each month. Then it is needed to pay 15$ per 100 000 transactions.

Cloudmade routing API allows calculating route times, distances and more, vehicle, bicycle or pedestrian routing, and real-time, turn-by-turn routing. First 10 000 transactions free each month. Then it is needed to pay 15$ per 100 000 transactions. In addition, CloudMade provides SSL access to their servers for $5 per 100 000 requests.

The rest part of the paper describes how to organize own mapping service for mobile tourist guide application by using the existing open-source OSM-based software. This service allows building routes, showing the results on the self-draw map on mobile device. The choice of own OSM-based mapping service building can be explained by strong limitation to the number of queries, license issues of query results usage in the other map services and performance limitations of the OSM-based public services such as an amount of queries limitation, hardware performance and service loading limitations. At the same time, only ridesharing service from the work of A. Smirnov et al. "Smart Logistic Service for Dynamic Ridesharing" [11], which is a part of the mobile tourist guide, has to implement about 12000 routing queries to find matching path for 20 passengers and 30 drivers. The less value of users is not enough to the service working. The amount of queries for the ridesharing service exceeds all limitations of services, described above, and requires either buying the business account for using existing services or own web map service implementation.

## III. Own map rendering

To reduce the load on the main OpenStreetMap site, developers are highly encouraged organizing their server for rendering and distributing map to the end users. For this purpose the main page of the OpenStreetMap project uses *PostgreSQL* DBMS with *PostGIS* extension as a main datasource, tile generator *Mapnik* for map rendering and a special module — *mod_tile* — for Apache web server for tiles caching and sharing. PostGIS extension provides functions to work with geographical data stored in PostgreSQL database. The map is shown to the user with using a JavaScript Leaflet library. This combination of software is well tested and longtime checked software that provides high-speed work and efficient works on high load. All the software is open source and can be used by anybody for own map services organization.

### *A. Map data source organization*

OpenStreetMap project delivers the map data in the OSM XML-file. This file contains a description of the properties of all the elements that can be displayed on the map with additional information about when the element has been added, who has add it, etc. The file structure allows manipulating data while map style is configured and create layers showing an additional information over the main map layer. However, the use of the file directly from the file system is difficult due to the great file size and a huge amount of information held in it. For example, file that holds data about whole planet requires more than 450 Gb HDD space (29 Gb compressed, pbf format). Only St. Petersburg requires about 330 Mb of free disk space. It is very hard to search through the similar amount of information that is presented with one file. It is also needs a lot of computational power and time.

To resolve the problem of a big file size the model of holding map data in the PostgreSQL DBMS has been developed by OSM community. Osm2pgsql schema [12] has historically been the standard way to import OSM data for use in rendering software such as Mapnik. The import to the PostgreSQL database with PostGIS extension is handled by the osm2pgsql software, which has two modes of operation, slim and non-slim, which control the amount of memory used by the software during import and whether it can be updated. Slim mode supports updates, but time taken to import is highly dependent on disk speed and may take several days for the full planet, even on a fast machine. Non-slim mode is faster, but does not support updates and requires a vast amount of memory.

The import process is lossy, and controlled by a configuration XML-file in which the keys of elements of interest are listed. The values of these "interesting" elements are imported as columns in the points, lines and polygons tables. (Alternatively, values of all tags can be imported into an "hstore" type column.) These tables can be very large, and care must be paid to get good indexed performance.

### *B. Map Rendering*

For the map rendering the Mapnik toolkit is used. It can draw map from many sources: directly from OSM-XML file, from the data stored in the database, from shapefiles and others. This toolkit is also a main software for map rendering on the main page of OpenStreetMap project. Mapnik is written in C++ and can be scripted using binding languages such as Javascript (Node.js), Python, Ruby, and Java. It uses the Anti-Grain Geometry rendering library and offers anti-aliasing rendering with subpixel accuracy. It can read ESRI shapefiles, PostGIS, TIFF rasters, .osm files, any GDAL or OGR supported formats, CSV files, and more. Pre-built packages are available for OS X and Windows and can be found at Mapnik.org/download. Many Linux distributions provide packages for installing Mapnik.

Mapnik can output map images to a variety of graphics formats — PNG, JPEG, SVG, and PDF, of any defined size, any area in different geographic projections. OpenStreetMap and other services primary use of Mapnik involves rendering thousands of 256 × 256 pixel tiles (Fig. 1), which are displayed with a JavaScript "Slippy Map" interface [13].

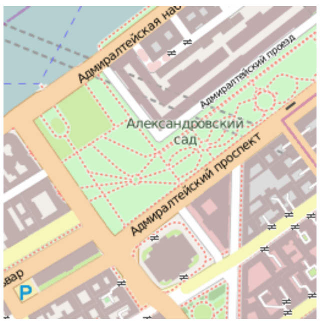

Fig. 1. An example of a tile in St. Petersburg at zoom level 15

At the server side, the mod_tile module for Apache web server is used. This module uses Mapnik for tiles generation. It provides a dynamic combination of efficient caching and on the fly rendering. Due to its dynamic rendering, only a small fraction of overall tiles needs to be kept on disk, reducing the resources required. At the same time, its caching strategy allows for a high performance serving and can support several thousand requests per second.

### C. *Map displaying*

There are two main JavaScript libraries that combine individual tiles in whole map: Openlayers and Leaflet. These libraries also provide API to include additional information layers over the main map layer. The JavaScript code with "Slippy Map" can be included into html web page as widget.

*1) OpenLayers* library has been developed by OSM community to display a map on the main page of the OpenStreetMap project It is very powerful library that allows to configure all the properties of the map. It can display map tiles loaded from any sources as well as additional objects such as markers, lines, polygons, popups, etc. At the current version of OpenLayers library (2.13.1) the syntax is quite hard and realisation of several features needs a lof of code to be written. Moreover, some elements of the interface have problems with scaling when running on the mobile devices. These issues are planned to be resolved at the OpenLayers 3.0 release.Also it is planned to easing the use of the library by providing a new syntax.

*2) Leaflet* library has been developed by CloudMade project community. Implements most of the OpenLayers functions and has an easy to use API with simple syntax. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while still being accessible on older ones. It is used for the main OSM website map, as well as on Flickr, Wikipedia mobile apps, foursquare, craigslist, IGN, Washington Post, The Wall Steet Journal, Geocaching.com, City-Data.com, StreetEasy, Nestoria and Skobbler among others [14].

## IV. ADDITIONAL FUNCTIONS PROVIDED BY LIBRARIES

As it has been mentioned, the main goal of the OpenStreetMap project is to provide high quality geographical information. There are a lot of applications and libraries using to work with this information. All of these applications are developed by OpenStreetMap community. The most used functions in every GIS are routing and geocoding.

### A. *Routing libraries*

OpenStreetMap data includes information about roads for routing by many modes including car, foot, bicycle and even a horse. There are many routing applications and libraries, developed by the OSM community. The most interesting from them are pgRouting [15] and GraphHopper [16].

*1) pgRouting:* is an extension for the PostgreSQL DBMS. It allows manipulate through the map data imported to the PostgreSQL DBMS with PostGIS extension. PgRouting searches for path based on a weighted graph that built based on information about roads in OpenStreetMap. The main library code is written using C++ program language with wrappers on SQL.

pgRouting provides functions for:

- All pairs shortest path, Johnson's Algorithm;
- All pairs shortest path, Floyd-Warshall Algorithm;
- A* shortest path;
- Bi-directional Dijkstra shortest path;
- Bi-directional A* shortest path;
- Dijkstra shortest path;
- Driving Distance – allows finding nodes that are laying in the defined distance from the starting node;
- K-Shortest path, multiple alternative paths;

- K-Dijkstra, one-to-many shortest path allows finding shortest paths from one origin to several destinations;
- Traveling sales person;
- Turn Restriction Shortest Path (TRSP);
- Shortest Path Shooting Star. Searches for the shortest path from edge to edge (not from vertex to vertex).

*2) GraphHopper*: is a fast and memory efficient Java routing engine. Designed for the server, desktop, as well as for Android. Works out of the box with OpenStreetMap (osm and pbf) but can be adapted to use another data.

GraphHopper provides functions for:

- A* Shortest Path;
- Dijkstra Shortest Path;
- Bi-directional Dijkstra Shortest Path;
- Bi-directional A* Shortest Path;
- One-to-many shortest paths.

### *B. Geocoding libraries*

*1) Tiger Geocoder:* is a PL/pgSQL based geocoder written to work with the TIGER (Topologically Integrated Geographic Encoding and Referencing system) / Line and Master Address database export released by the US Census Bureau. Geocoder is packaged with the PostGIS source code and can be used with PostgreSQL/PostGIS map database. Although it is designed specifically for the US, a lot of the concepts and functions are applicable and can be adapted to work with other country address and road networks.

*2) Nominatim:* is another geocoder for PostGIS, gaining in popularity and more suitable for international use. It uses OpenStreetMap gazeteer formatted data. It requires osm2pgsql for loading the data, PostgreSQL 8.4+ and PostGIS 1.5+ to function. It is packaged as a webservice interface and seems designed to be called as a webservice. Just like the tiger geocoder, it has both a geocoder and a reverse geocoder component. From the documentation, it is unclear if it has a pure SQL interface like the tiger geocoder, or if a good deal of the logic is implemented in the web interface.

## V. WEB MAPPING SERVICE IMPLEMENTATION

As it has been mentioned, web mapping service for mobile tourist guide should implement the following functions: location and routing support, address searching support, maximal coverage area, and data accuracy and also has no limitations for queries amount. According to these requirements and descriptions of the services, presented in Section 2 and Section 3, the following combination of project is used for GIS building on the server side (Fig. 2):

- OpenStreetMap as map data source,
- PostgreSQL database with PostGIS extension for map storing and data processing,
- Apache mod_tile module with Mapnik-based renderd rendering core for map rendering,
- pgRouting library for routing,
- Nominatim library for geocoding.

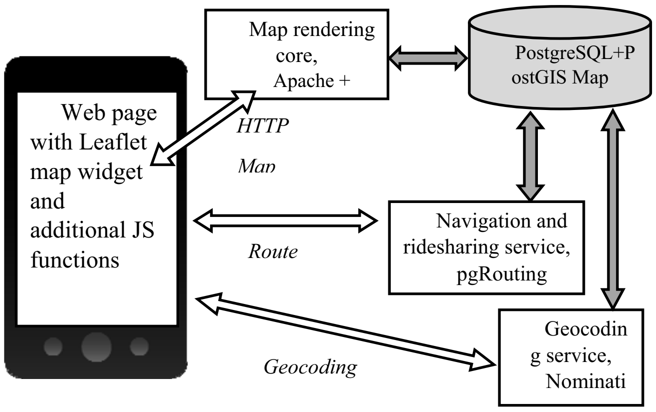
Map rendering core, Apache +
PostgreSQL+P ostGIS Map
Web page with Leaflet map widget and additional JS functions
HTTP
Map
Navigation and ridesharing service, pgRouting
Route
Geocodin g service, Nominati
Geocoding

Fig. 2. Web mapping service architecture for the mobile tourist guide

On the client side, the Leaflet library for map showing is used. In addition, client provides the device location to the server using device-specified geolocation API.

For the information importing, two main utilities has been used: osm2pgsql and osm2pgrouting. First utility imports the full OSM XML file to the database tables, except information about changes, versions, timestamps, etc. It creates the table structure, imports file and builds indexes for the fast searching.

Osm2pgrouting utility has been used to build routable database. It has created all needed tables, imported information about selected types of road and created topology over these roads.

On the client side the developed GIS is presented by the web page with included “slippy map”, provided by Leaflet library. When the web page view opens by the user, the map is initializing. Map contains of a set of tiles with defined zoom level. These tiles are rendered by the renderd daemon at the server, which is a part of Apache mod_tile module. For the server usage optimization rendered tiles are saved in the cache directory. The structure of cache directory looks like follows:

```
shared_cache_folder
     |- z
        |- x
           |- y.png,
```

where '*shared_cache_folder'* is a name of shared folder on the web-server, '*z'* is a name of subfolder in '*shared_cache_folder'*, associated with a map zoom level, '*x'* is name of subfolder in '*z'*, associated with a first map's tile index, and '*y'* is a file in '*x'*, associated with a second tile's index (see Fig. 3). When the web page with "slippy

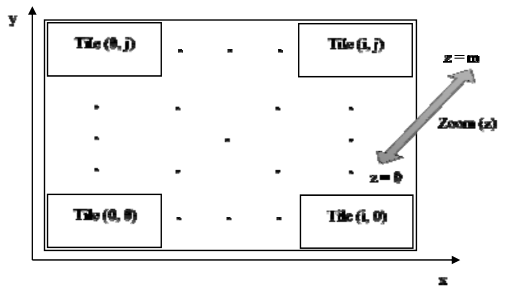

Fig. 3. "Slippy" Map view structure

map" requests tile, it constructs an http request by the following scheme:

```
http://path_on_server/{z}/{x}/{y}.png
```

If there is no tile in the destination path, renderd is started to render it and save in cache. Else the tile is loaded from the cache and adds to the "slippy map" view on the web page.

In addition to the Leaflet API, several functions has been written with JavaScript to provide possibilities of showing advanced information such as single routes, routes with ridesharing, markers with additional information and layers descriptions. These functions can be accessed from the web page as well as from the application.

Navigation and ridesharing service is developed based on the pgRouting library. The service allows building shortest paths and searching the transportation means to reach the places of interest. It uses the following pgRouting functions:

- shortest path A* for finding shortest paths for tourists travelling by car or preferring foot walks;
- driving distance for possible meeting points search in the case of ridesharing;
- K-Dijkstra, one-to-many shortest path for the optimal ridesharing route finding.

For the geocoding possibilities, the Nominatim library is using. Because of Nominatim public server maintainers ask not to provide a heavy usage of theirs servers, the own instance of Nominatim has been configured for working with Apache web server. When user tries to find an address, the following request is generated on the client side:

```
http://path_to_nominatim/search?format=jso
n&q=queried+address
```

This request returns found places with coordinates in JSON format that interpreted by the client application and is showed to the user. It can be several results from one query and user should select one. In the case of reverse geocoding, when user clicks on the map, client application generates another request to the Nominatim:

```
http://path_to_nominatim/reverse?format=js
on&lat=latitide&lon=longitude&zoom=zoom_leve
l
```

This request returns a place address that depending on the zoom level. At higher zoom levels there are more details found by reverse geocoding. For example, at zoom level 10 and the coordinates `lat=59.93404`, `lon=30.30599`, the geocoder finds Admiralteysky District of St. Petersburg, Russia. At the same coordinates at zoom level 18 there is St Isaac's Cathedral.

## VI. Performance test

Since the routing functions are the most resource-intensive in presented structure, the test is based only on the routing time evaluation. All evaluations have been performed in the virtual machine with Debian 7.2 operating system, running under the Hyper-V hypervisor. For the virtual machine 4 cores of the Intel® Xeon(R) CPU E5620 2.40GHz and 3 Gb DDR3 RAM have been allocated. In addition, files of the virtual machine with map database are held on RAID 1 to improve read performance of the database.

Routing functions work over the built topology and run rather fast. For example, it is need about 401 ms to find route from the most northern point of St. Petersburg to its most southern point (path consists of 363 points, total length is 31 km) with A* shortest path function. In this example path has been searched through the all types of roads, including walk roads, footways, steps and other road types which are cannot been ridden by car. After the excluding roads that cannot been passed by car, search time has reduced to 172 ms (366 points, total length is 31.3 km). Fig. 4(a) and Fig. 4(b) provide an example of how the searched path depends on the available road types. At the Fig. 4(a) a route has been built through the all road types with the following query:

```
SELECT * FROM pgr_astar(
     'SELECT gid AS id,
      source::integer,
      target::integer,
      length::double precision AS cost,
      reverse_cost,
      x1, y1, x2, y2
     FROM ways', 535, 44115, true, true)
```

In this SQL query, the shortest path A* function is used. In this function, the inner SQL selects the roads that will be used for routing; first number is a start point ID, second number is an end point ID, and Boolean values set the road graph is directed and edges has reverse cost. This query returns 135 path points with total distance 8.4 km at 342 ms. With this path driver should ride a car through yards, footpaths, etc.

All roads in OpenStreetMap have a classification tag that describes the road class (highway, motorway, residential,

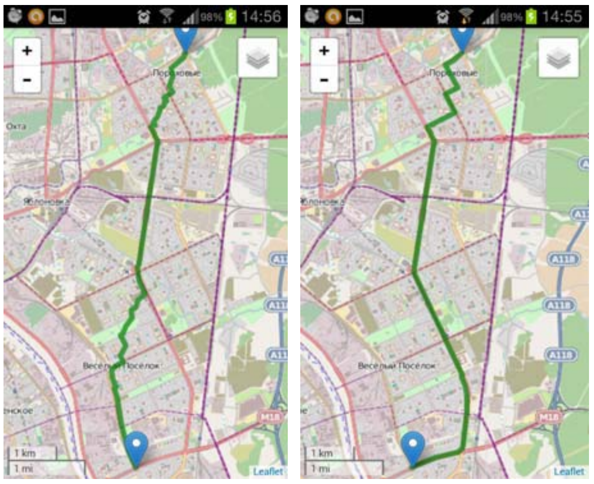

Fig. 4. Route (a) without the road type restrictions (left)
(b) with restriction of road types that cannot been passed by car (right).

footway, etc.), maximal available speed, road cover (asphalt, gravel, etc.). These tags converts to the road class ID while importing to the database and can be used for roads filtering. The Fig. 4(b) shows path with restriction of roads types that cannot been passed by car. In the imported database, these roads have class ID less or equal *110*. The request that takes into consideration the road restrictions is presented below:

```
SELECT * FROM pgr_astar(
     'SELECT gid AS id,
      source::integer,
      target::integer,
      length::double precision AS cost,
      reverse_cost,
      x1, y1, x2, y2
     FROM ways WHERE class_id <= 110’,
535, 44115, true, true)
```

This request returns path with 112 points with total distance 8.8 km at 130 ms.

## VII. Conclusion

There are many web mapping services that can be used by developers to build own application with map functions support. Most of them provide possibilities of map showing, routing, geocoding and other functions. At the same time, these services have many restrictions for using their functions. OpenStreetMap project provides geographic information data for free. Developers can build own geographic information services with needed functions based on the OSM data. All needed software is free for use, modification and provides many additional functions to manipulate the OSM data. Therefore, own map service may be implemented. Developers may extend the functionality of this service by using applications and libraries developed by the OSM community, or realize own functions to manipulate the OSM data. The case study that is presented in the paper shows an example of the map service building for tourist support application. Performance tests of the built system show high-speed evaluations of the used functions. It means that the system can work under high load and provide high service quality to users.

## Acknowledgment

This research is a part of grant KA322 «Development of cross-border e-tourism framework for the programme region (Smart e-Tourism)» of Karelia ENPI programme, which is co-funded by the European Union, the Russian Federation and the Republic of Finland. The presented results are also a part of the research carried out within the project funded by grants # 13-07-00336 and 13-01-00286 of the Russian Foundation for Basic Research.

## References

[1] A.K. Dey, D. Salber, and G.D. Abowd, (2001). *"A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications,"* Context-Aware Computing, A Special Triple Issue of Human-Computer Interaction, 16. Retrieved August 13, 2012. [Online]. Available: http://www.cc.gatech.edu/fce/ctk/pubs/HCIJ16.pdf.

[2] A. Smirnov, A. Kashevnik, S.I. Balandin, and S.Laizane, Intelligent Mobile Tourist Guide: Context-Based Approach and Implementation, Internet of Things, Smart Spaces, and Next Generation Networking, *13th Int. Conf., NEW2AN 2013, and 6th Conf., ruSMART* 2013, St. Petersburg, Russia, August 28-30, 2013, Springer, LNCS 8121, pp. 94-106.

[3] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Shilov, M. Shchekotov, N. Teslya, Recommendation System for Tourist Attraction Information Service, *Proc. of the 14th Conf. of Open Innovations Association FRUCT*, State University of Aerospace Instrumentation, pp. 140-147.

[4] Google Map Coverage Filtered [Online]. Available: http://gmaps-samples.googlecode.com/svn/trunk/mapcoverage_filtered.html

[5] Google Maps/Google Earth APIs Terms of Service. [Online]. Available: https://developers.google.com/maps/terms#section_10_12.

[6] Bing Maps Geographic Coverage [Online]. Available: http://msdn.microsoft.com/en-us/library/dd435699.aspx.

[7] Bing/Coverage. OpenStreetMap wiki. [Online]. Available: http://wiki.openstreetmap.org/wiki/Bing/Coverage.

[8] Microsoft® Bing™ Maps Platform APIs’ Terms Of Use [Online]. Available: http://www.microsoft.com/maps/product/terms.html.

[9] Yandex.Maps API Terms of Use [Online]. Available: http://legal.yandex.com/maps_api/.

[10] Routing. OpenStreetMap Wiki. [Online]. Available: http://wiki.openstreetmap.org/wiki/Routing#Developers.

[11] A. Smirnov, N. Shilov, A. Kashevnik, N. Teslya, "Smart Logistic Service for Dynamic Ridesharing," *Internet of Things, Smart Spaces, and Next Generation Networking, 12th International Conf., NEW2AN 2012, and 5th Conf., ruSMART 2012*, St. Petersburg, Russia, August 27-29, 2012, pp. 140-151.

[12] Osm2pgsql/schema. OpenStreetMap Wiki [Online]. Available: http://wiki.openstreetmap.org/wiki/Osm2pgsql/schema.

[13] Slippy Map. OpenStreetMap Wiki. [Online]. Available: http://wiki.openstreetmap.org/wiki/Slippy_Map
[14] Leaflet. An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps [Online]. Available: http://leafletjs.com/.
[15] pgRouting. [Online]. Available:http://pgrouting.org.
[16] GraphHopper. [Online]. Available:http://graphhopper.com/.