Spatial Filters Implementation for Geo2Tag LBS Platform

Vasilii Romanihin, Mark Zaslavskiy, Kirill Krinkin
Saint-Petersburg State Electrotechnical University
Open Source and Linux Lab
Saint-Petersburg, Russia
{bac1ca89, mark.zaslavskiy, kirill.krinkin}@gmail.com

Abstract

Location-based services became more and more widespread because of popularity mobile phones with positioning functions. Today support of 3d objects became main trend in such services development. Navigation systems and search services for huge buildings is only one example of using this feature. This article describes 3d objects and filters support implementation in Geo2Tag platform.

Index Terms: location-based services, spatial filters, spatial objects.

I. Introduction

Today location-based services (LBS) market grows very fast because now you can't find new mobile phone without any positioning module [7]. At the same time accuracy of positioning modules also increases [8] and this fact provides new opportunities for location-based services.

One of the basic trends in LBS is support of spatial objects and 3d maps. Google added support of altitude and search inside the buildings at Google Maps for mobile in 3D project since 29 November 2011. Similar functionality based on OpenStreetMaps API was implemented in experimental OSM-3D project.

Geo2Tag is a universal open source platform for LBS. It provides API for interaction with 2d geodata. Adding support of spatial objects will allow increasing number of services that can be implemented on this platform (in-buildings search and navigations) and it will be the first open source LBS solution with spatial data support.

Part II describes Geo2Tag architecture before adding 3d objects support and requirements for filters. Part III describes what architecture of filters was selected, how they were added into platform and implementation details about client library for Android.

II. Overview

A. Problem statement

Our goal was to add into Geo2Tag support of spatial objects (supporting altitude) and interfaces for spatial filtration. To reach this goal we should solve such tasks:

- Support in the platform of the three geographic coordinates;
- Three-dimensional marking of content;
- Two-dimensional spatial filters;
- Three-dimensional spatial filters.

B. Geo2Tag Architecture

Geo2Tag is centralized system with server, that storage all information and provide access to it by REST API. It has following main advantages – it is open source and it doesn't depend on concrete web-server or DB type. Server consists of two main parts – server application and database (DB). Server application is a web application written by FastCGI framework. It main task is processing clients requests to REST API. Database contains information about users and their geodata.

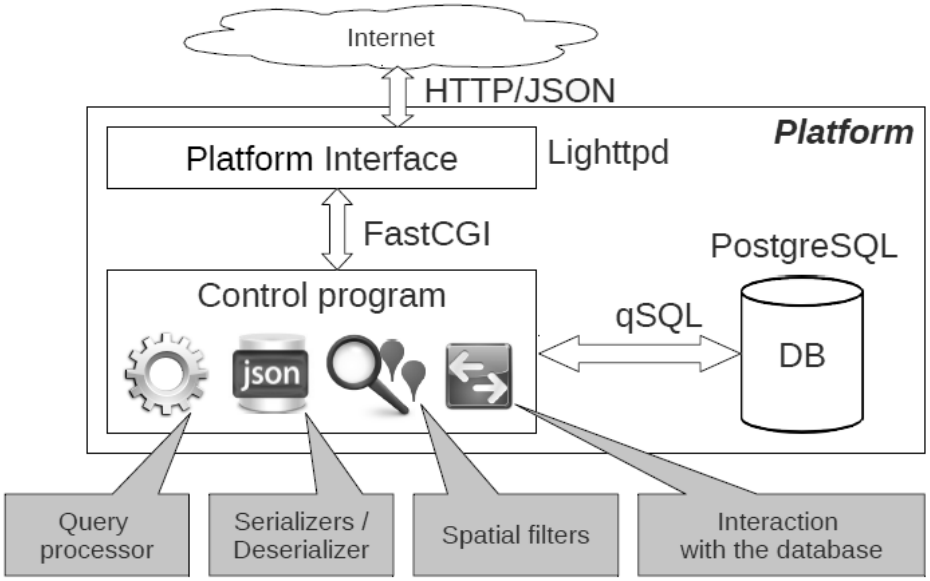

Fig. 1. Platform architecture scheme

Platform use following basic data types:

- *User* – data that represent human user of Geo2Tag – login and password;
- *Channel* – object that contain name, description and set of tags. User subscription for the channel means that user can read tags from it;
- *Tag* – object that contain URL of multimedia object, name, description, time of creation and coordinates(latitude and longitude);

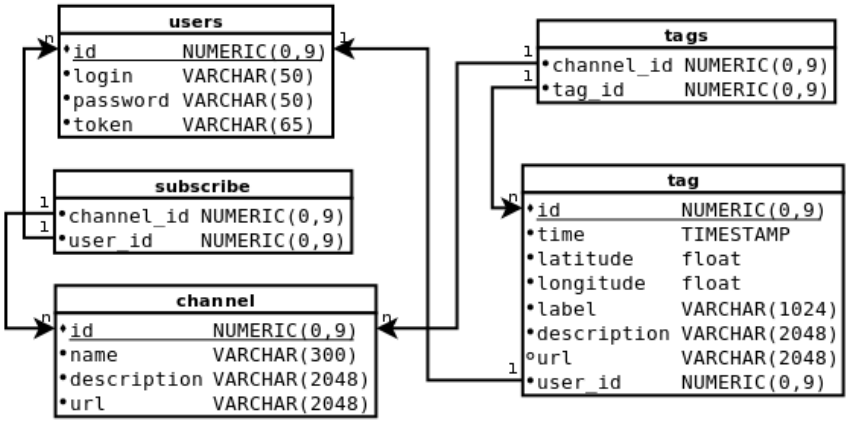

Fig. 2. DB logical scheme

Without 3d objects and spatial filters support REST API was consisting of such requests:

- *login* – get authorization token;
- *addUser* – add new user into platform;
- *subscribe* – subscribe user to the channel;
- *unsubscribe* – unsubscribe user to the channel;
- *channels* – get available in current location channels list;
- *subscribed* – get list of channels that user is subscribed;
- *addChannel* – add new channel into database;
- *writeTag* – add tag into subscribed channel;
- *loadTags* – get tags from all your channels that lies in specified radius near specified location.

More information about platform can be found at related work – [6].

III. IMPLEMENTATION

A. Altitude support

Adding altitude support into platform was the beginning of the spatial filters support. By term "altitude support" we mean that content can be marked with three geographical coordinates – longitude, latitude and altitude. Because only *tag* data type contains coordinates, so only parts of realization associated with it should be modified. This process contains following steps:

- Extension of *tag* relation by *altitude* attribute. This step will allow to store simple 3d objects (3d tags) into platform DB;
- Modification of REST API *WriteTag* request format and *LoadTags* response format – added altitude parameter;
- Modification of server application code for working with altitude:
  - Containers (modify tag entity realization in platform for storage altitude),
  - DB interaction (modify SQL requests for read and modify altitude),
  - *WriteTag* and *LoadTags* query processors (add additional parameter for altitude and allow to add tags with altitude and retrieve tags with altitude from platform).

After these actions any user gets ability to add and manage 3d data using platform API.

B. Spatial filters implementation

By the term *filter* we mean platform REST API request that consists of user's conditions and return all tags that satisfy this conditions. For the realization of spatial filters we need to solve following tasks:

- Filtration by 2d shape (return tags that are inside of specified figure in longitude-latitude plane);
- Filtration by time interval (return tags that was created in specified time interval);
- Filtration by altitude interval (return tags that has altitude inside specified altitude interval);
- Filtration by composite rules (aggregation of all filtration types listed above).

So, using just 2d shapes as filters will give 2d filters, combination of 2d shapes with altitude interval will give 3d filters and addition of time filter will give the most flexible solution.

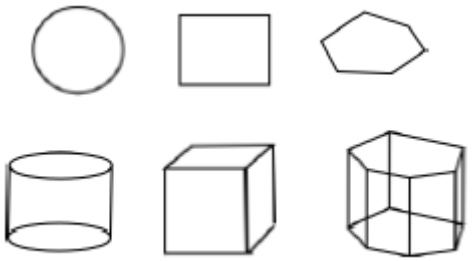

Fig. 3. Basic 2d and 3d filter shapes

As a basic 2d shapes we choose circle, rectangle and polygon, because this shapes are the most common in buildings architecture. Because 3d filters are based on this, basic shapes for them were cylinder, cube and fence.

For implementation of such filters idea we choose following architecture (*DataMark* type represent *tag* entity):

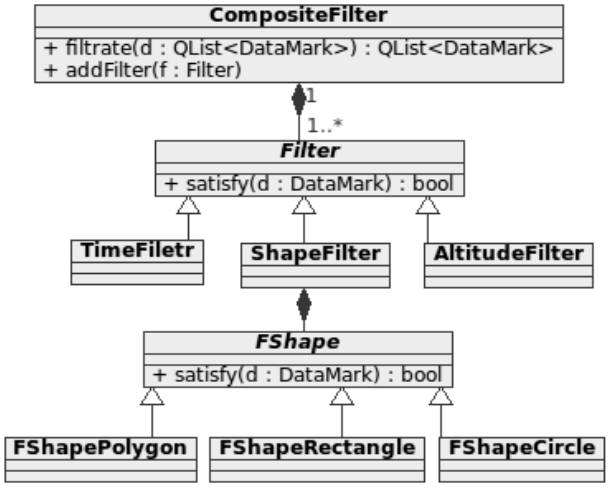

Fig. 4. Spatial filters class diagram

Class *CompositeFilter* aggregates three types of filtration. It main method *filtrate* check input list of tags and return only tags, which satisfy filters conditions. Method addFilter allow adding any child for the *Filter* class into *CompositeFilter* filter list. Each child of *Filter* has method *satisfy* that check filter conditions for one tag. The same method exists in *FShape* class and its children – classes representing different 2d shapes for spatial filters.

CompositeFilter works using simple algorithm described below:

- Apply *TimeFilter* to user *DataMarks***;**
- Apply *ShapeFilter* to *DataMarks* that satisfy *TimeFilter*;
- Apply *AltitudeFilter* to *DataMarks* that satisfy *ShapeFilter*;
- Return *DataMarks* that satisfy *AltitudeFilter*.

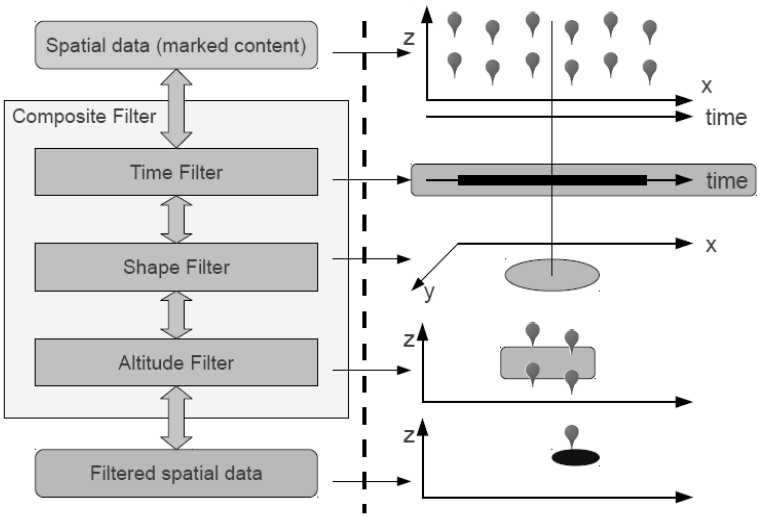

Fig. 5. CompositeFilter work algorithm

On a basement of filters architecture were implemented six spatial filters:

- *filterCircle;*
- *filterBox;*
- *filterPolygon;*
- *filterCylinder;*
- *filterCube;*
- *filterFence.*

Below you can see example of *filterCylinder* query:

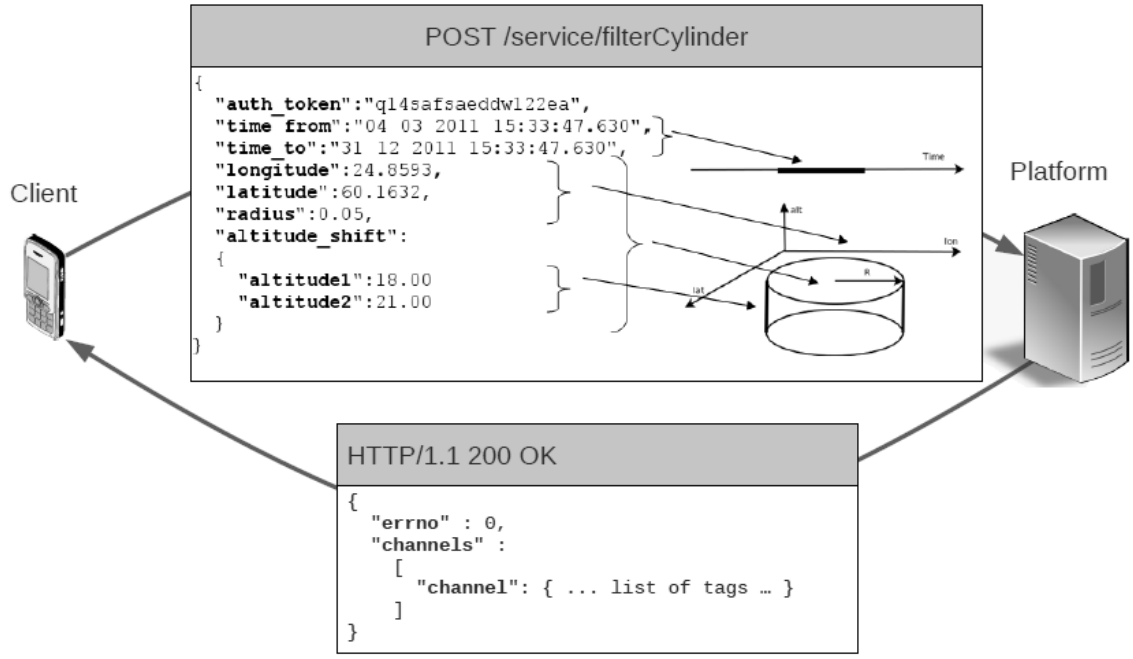

Fig. 6. Example of filterCylinder query

Because 3d filters extend 2d filters by altitude and time restriction they will work slower than 2d filters. In this work no performance investigation was done, but it will be done in next papers.

C. Spatial filters implementation

For the next stage of this project (mobile client implementation) client library for Android OS was written. This platform where chosen because Android devices are now dominant part of the mobile phones market. Library realizes following functions:

- Abstraction from the Geo2Tag platform interface;
- Implement basic entities of platform (user, tag, channel);
- Support basic platform API (login, writeTag, loadTags, addUser, subscribe, unsubscribe, applyChannel, channels, subscribed);
- Support spatial filters (Circle, Rectangle, Polygon, Cylinder, Cube, Fence).

Library was implemented as two packages - *ru.spb.osll.objects* and *ru.spb.osll.json*. First package implements standard Geo2Tag entities - Mark, Channel, User, GeoPoint.

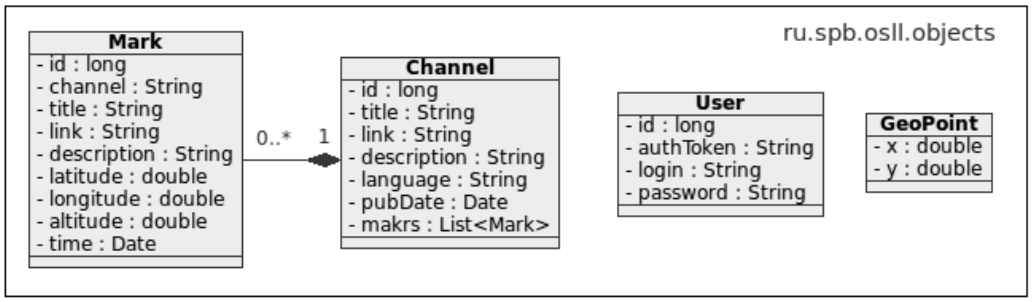

Fig. 7. Class diagram of *ru.spb.osll.objects* package

Package *ru.spb.osll.json* encapsulates client-server interaction and provides collection of serializations/deserializators for platform REST API.

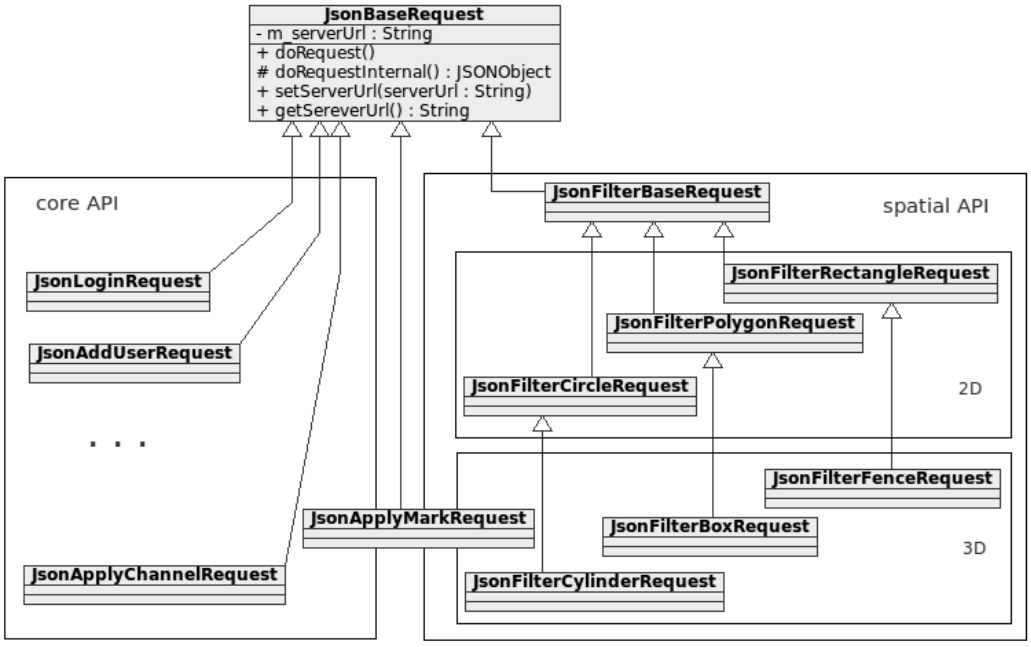

Fig. 8. Class diagram of *ru.spb.osll.json* package

IV. Conclusion

During this project were added support for altitude in geodata, new interfaces for spatial filtration and implemented Android client library. Future plans:

- Implement client libraries for other mobile platforms. We choose Symbian, MeeGo and IOS as priority platforms because Nokia and Apple phones take more than 50% of market in Russia [9];
- Add support of complex 3d objects. At the current moment platform supports only 3d points, we want to add support of objects similar to filter shapes;
- Add support for events system. At this step we want to provide service for user notification about his closeness to some objects;
- Platform performance and load balancing. In this case our first goal is to research vertical and horizontal scalability of platform and try to improve it. Also different filters performance will be investigated.

Acknowledgment

The authors would like to thank Finnish Russian University Cooperation in Telecommunication Program for provided equipment and support.

References

[1] Steiniger S. Foundation of Location Based Services: Lecture Notes on LBS / S. Steiniger, M. Neun, A. Edwardes. - University of Zurich. 28 p.
[2] Google Maps for mobile in 3D. URL: http://www.google.com/mobile/maps/3d/.
[3] OpenStreetMap-3D. URL: http://wiki.openstreetmap.org/wiki/OSM-3D/.
[4] REST. URL: http://en.wikipedia.org/wiki/Representational_state_transfer/.
[5] Fast CGI. URL: http://en.wikipedia.org/wiki/FastCGI/ .
[6] I. Bezyazychnyy, K. Krinkin, M. Zaslavskiy, S. Balandin, Y. Koucheravy Geo2Tag Implementation for MAEMO 7th Conference of Open Innovations Framework Program FRUCT. Saint-Petersburg, Russia 26-30 April 2010.
[7] GPS-enabled phones the next 'must have'. URL: http://www.chinadaily.com.cn/bizchina/2011-01/14/content_11855043.htm/.
[8] GPS Accuracy. URL: http://www.gps.gov/systems/gps/performance/accuracy/.
[9] 2012 Mobile Market Share [Infographic]. URL: http://connect.icrossing.co.uk/2012-mobile-market-share-infographic_7962.