Location-Based Services in Public Buildings: LIST for N900

Denis Bespalov, Gregory Serebryakov
Lobachevsky State University of Nizhni Novgorod
Nizhni Novgorod, Russia
wl.maemo@yandex.ru

Abstract

The LIST mobile service is based on the idea that an existing (already formed) offline local community, united by a common territory, common affairs, common procedures and common interests needs tools that make it possible for their members to interact effectively with infrastructure objects (i.e. the university campus) and communicate with each other. The service allows community members to get interesting local content and to exchange location-based information connected with off-line services and facilities. Also, this tool can simultaneously be an interface between the providers of online services and products aimed exactly at the target audience and this audience.

To achieve this target, the service must be available at as wide range of platforms, as possible. Also it should provide optimal performance on mobile devices. In this release of LIST service's Qt client have been redesigned such resource-demanding part of application, as XML to local form translators. We've also added abilities, provided with GPS module to the application and created a MeeGo port of it.

Index Terms: Location-based services, Mobile services, LBS, Social networks

I. Introduction

The LIST mobile service is based on the idea that the existing (already formed) offline local community, united by a common territory, common affairs, common procedures and common interests needs tools that make it possible for their members to interact effectively with the infrastructure object (i.e. the university campus) and communicate with each other. The service allows community members to get interesting local content and to exchange location-based information connected with off-line services and facilities. Also, this tool can simultaneously be an interface between the providers of online services and products aimed exactly at the target audience and this audience.

Users of the LIST service have an opportunity to participate in the mobile social network, uniting them in the local community and allowing them to generate relevant and interesting local content. For local information service at the university campus the target audience is primarily youths and key vendors are ones who provide goods and services targeted at young people.

The service provides access to a diverse local content associated with the campus UNN (University of Nizhni Novgorod). Most of the content is unique and not available through other channels (UNN sites, social networks).

The first version of the LIST service is intended for students, staff and visitors of university campuses. So, it allows users to get access to many specific abilities, namely:

- Orientation on campus and inside buildings
- Find places and people on campus
- Find and order books in the library
- Access to the schedule, the ads, etc.

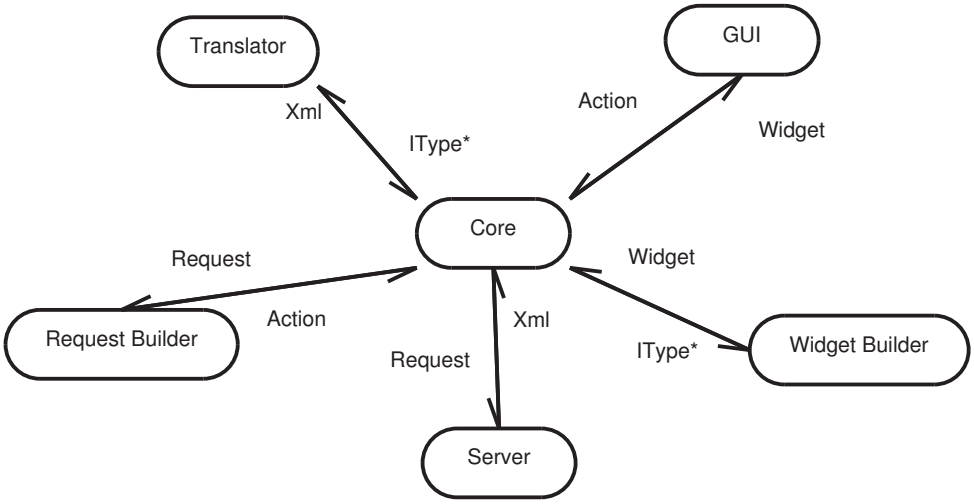

Fig. 1. Application's architecture

The content available through the service consists of two main parts: content created by administrators of service (interactive map of the campus and plans for buildings, information about places and people in the UNN, timetables, official announcements from the university administration and the deans' offices) and the user-generated contents. Users have an opportunity to comment on any information placed in the service, to post messages on their mobile mini-sites and message boards, to communicate in thematic chats, to create location-linked notes. In this model we'll get self-updating and self-growing service.

The concept of service: at the university, business center, shopping mall, park etc. people often need in:

- finding the right room, the person object or service;
- understanding where they are in the building (especially if they come to this place for the first time);
- obtaining the information on the objects' local geography. And, of course, they often just want to talk to someone who, like them, is fated to look through long list with many links lead to information to find associated with this place.

Find answers online, using Yandex, Google, Rambler or another search engine via your mobile browser is not a good idea, because you will get an information that is not needed in a particular place and at the current time, or which is scattered across many documents and Web sites to download and view. So the user will waste a lot of time and money. A lot of information in modern web was designed for leisurely analysis at your home or office and only occasionally can be helpful in field conditions. Ironically, the old paper guidebooks, handbooks, and, simply, a notebook in a pocket in these situations can be beneficial to the Internet.

Now we can say, that the concept of the LIST service has been widened a lot since the moment of it's realization start. Now users of service can use all the power of their desktop systems to collect data. As a result, they get something, that we name "photohistory" - an aggregation of text and information, which can be tagged, and associated with some position on the interactive map. Then, after uploading such photohistory on server, it becomes accessible from all kinds of devices, on which LIST service client is implemented.

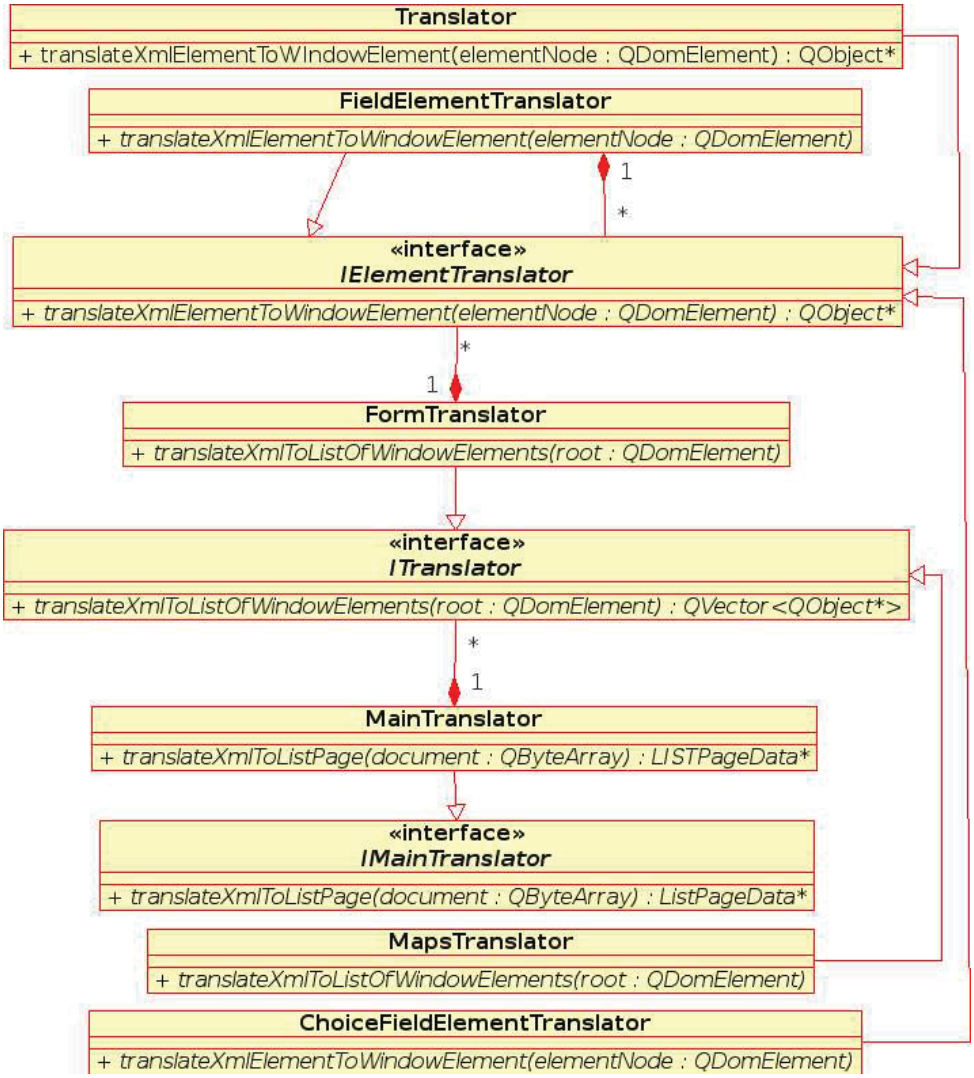

Fig. 2. Old translators

II. Main part

A. System Architecture Fundamentals

LIST is built on the basis of client-server architecture. The client is implemented for J2ME and Qt platform. Of course, Qt-based version provides some additional abilities, compared to J2ME version, but base functionality is identical. The main server (LIST) is implemented with Ruby on Rails. Client and server exchange data is carried out via XML-files.

The main server (LIST) contains all the local content and user information. During the interaction with the server client behaves like a browser — displays the resulting page, which are described by using a specific XML markup language. Simplified architecture of client application is provided on the Fig. 1.

The functionality of system:

- Displaying XML pages generated on the server as an application screens
- Navigation and positioning with the use of raster maps on the basis of OpenStreetMaps and GoogleMaps

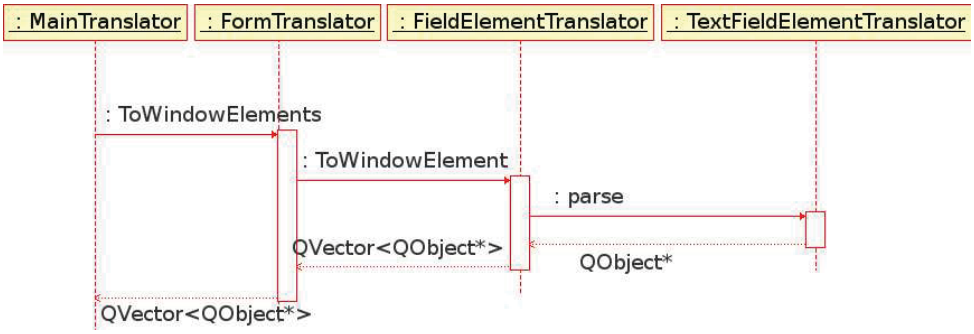

Fig. 3. Sample translation process

- Display maps of the campus and plans of premises using vector maps
- Creating notes linked with the specified position on the map and with the date
- Including additional services into the application

B. Processing XML pages on the client

Our application uses a markup language screens based on XML. When an user cause some event (choice of menu items, etc.) the client sends a HTTP request to the server, receives a response with a description of the next screen and displays it. Client supports both GET and POST methods, latter is used for transmitting photos.

Markup language supports several types of screens:

- main screen
- info screen
- photo's screen
- menu screen
- entry forms' screen
- map's screen

The architecture of XML's translator module has been changed a lot since last release. After doing some benchmarking and performance tests we abandoned the the complex tree of translators, which you can see on the Fig. 2.

Under the name 'Translator' is hide many specific types of translators, which we won't describe here, but you can see them on the Fig. 4

Now we have only few translators, as it shown on the Fig. 5.

Translation process, that you can see on Fig. 3, due to this redesign hasn't changed a lot.

C. Maps

Client supports raster maps. We are not using vector maps due to rendering of large maps (plots of cities) on the client side is not possible because of the platform's performance lack, so we use raster maps server CloudMade, obtained on the basis of vector maps OpenStreetMaps ([2]), nevertheless it can easily be changed to to another service, using Mercator projection of the globe ([1],[3]) and working with ellipsoidal approximation of the Earth form (e.g. Google Maps).

At a greater length you can read about Maps implementation in Denis Bespalov's article. Here we'll write only about some specific abilities, introducted in this version. Now, more and more devices, supporting GPS navigation, appears every year. So, we think. that providing

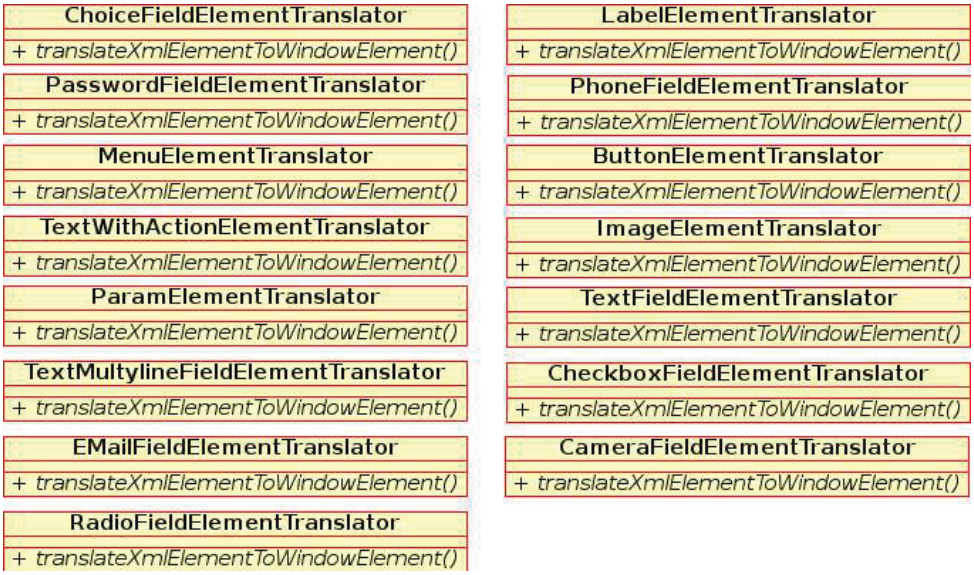

Fig. 4. Translator types

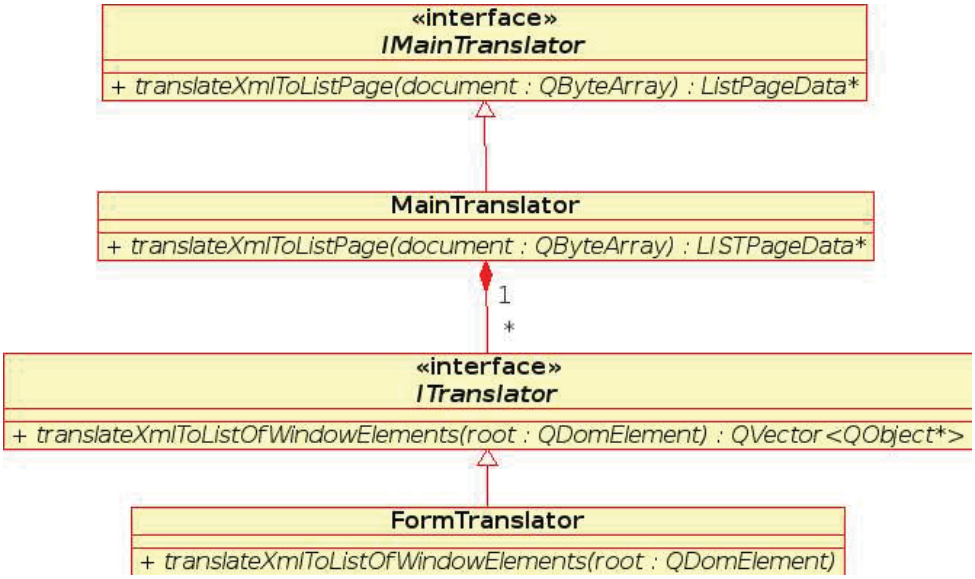

Fig. 5. New translators

the access to this feature from LIST client is very important task. This problem consists of two parts:

- imaging the position in real time
- processing of the saved tracks, loaded from third-party devices.

The first of them is solved using the Qt Mobility interfaces to native GPS api. So, the second is more interesting. Among the wide range of the devices, named GPS trackers, that are used for recording your route, the majority can export data in KML([4]) format. So, adding the KML parsing support into the LIST client is a consistent decision. Google, as the developer of the KML standart, provides the library for it's computing, and firslty we've thought about using it. But on the mobile devices one the most actual problems is a problem

of memory. Unfortunatelly, Google's libkml is to huge, especcially comparing to the size of our application, and we have to write our own KML parser, based on Qt XML library.

III. Conclusion

Coming to the conclusion, we should say that almost all aims, which were set on the stage of the project's concept developing, have been reached. Now the service LIST successfully works on the base of the Nizny Novgorod State University (UNN) and will be started at some more object soon.

The work under the support of GPS navigation and offline packages is in process now and we sure it'll be finished soon.

References

[1] Mercator projection http://en.wikipedia.org/wiki/Mercator projection
[2] Cloud Made API description http://developers.cloudmade.com/projects/tiles/documents
[3] J.P. Snyder,"Map Projections - A Working Manual", *U.S. Geological Survey Professional Paper 1395*, United States Government Printing Ofce, Washington, D.C., 1987
[4] KML description http://code.google.com/intl/en-US/apis/kml/
[5] libkml, Google's library for to parsing, generating and operating on KML http://code.google.com/p/libkml/