Route optimization and customization using real geographical data in Android mobile devices

Cira Cuadrat Seix[1] and Antoni Pérez Navarro[2]

***Abstract*— Nowadays, there are several services and applications that allow users to locate and move to different tourist areas using a mobile device. These systems can be used either by internet or downloading an application in concrete places like a visitors centre. Although such applications are able to facilitate the location and the search for points of interest, in most cases, these services and applications do not meet the needs of each user. Users can lose the internet connection or the results offered may not fit their schedules or their own personal tastes.**

This paper aims to provide a solution by studying the main projects, services and applications, their routing algorithms and their treatment of the real geographical data in Android mobile devices, focusing on the data acquisition and treatment to improve the routing searches in off-line environments.

***Index Terms*— Android, OpenStreetMap, OSMAnd, routing, optimization**

I. Introduction

AS defined in reference [1], the term Geographical Information Systems (GIS) involves a wide range of hardware, software and geographic data organized for managing and analyzing spatial information referenced with the aim to solve complex problems of planning and management, but also used for individual purposes.

With the advent of GPS software and the fast expansion and great acceptance for smartphones in the market such as IPhones, Blackberries or HTCs, and other mobile devices such as tablets and PDAs, has increased the number of GIS-based applications that deal with geographic information bringing it to the general public and promoting the creation of applications with functions for leisure and entertainment. These new applications provide information and services such as points of interest (POI's), routes of all types, navigation and augmented reality [2]. Currently, only in the Android Market it is possibly to find over 400 applications related to location-based systems (LBS) and GPS [3]. Over half of these applications are used in particular title, focusing mainly on the creation of routes especially for the preparation of tourist routes or location of POI's.

Tourism is an area that has experienced tremendous growth in recent years thanks to the emergence of geographic information accessible to everybody and free of charge [4]. Web applications like GoogleMaps [5], OpenStreetMaps [6] or Wikiloc [7] have their own versions to be used in smartphones and other mobile devices. Moreover, there is a wide range of applications that use these information systems providing specific services: audio tourist guides, default tourist routes, etc.

However, most mobile devices have many limitations to effectively manage geographic information, in most cases using a continuous Internet connection to run these applications on the server. The main problem lies in 3G or HSDPA (3.5G) coverage which only covers the most densely populated areas as shown in the image:

Fig.1. Areas with 3G coverage in Spain [8].

In addition, this service is expensive and often has limited bandwidth consumption per month.

Making the execution of these applications on the server is not a viable option because of the large number of areas where there is not coverage that most of the times are areas important for tourist purposes. Running GIS applications on mobile devices independent of the server is necessary. However, as pointed in [9] limitations in display performance and storage capabilities of these devices hinder this option: small screens, low working memory, limited data entry, etc. Therefore finding a solution that allows mobile users to have only the amount of information they really need is necessary. That

Manuscript received January 19, 2012. This paper is the Master Final Work in the Official Master in Free Software of the Universitat Oberta de Catalunya and it was supported in part by project AVANZA2 (TSI-020110-2009-442).

[1] C. Cuadrat Seix is a postgraduate student in the Universitat Oberta de Catalunya (e-mail: ccuadrat@uoc.edu).

[2] A. Pérez Navarro is a professor and the director of this work at the Universitat Oberta de Catalunya, Department of IT, Multimedia and Telecommunications, Barcelona, Spain (e-mail: aperezn@uoc.edu).

means allowing the users to filter the data according to their needs and their preferences. Thus, the known client-server architecture would be valid, even though, would only be required for short time periods because its goal is customizing the GIS for the user: the information sent would be different according to the interests of the user (architecture, nature) or according to other factors such as the number or types of users, or if the route is done in a vehicle (car, bicycle, walking). To achieve this goal will require considering several factors: to improve the hardware of mobile devices with more working memory and computing power, to have more ability to get input data, to better display and moreover to improve the running data, to create standardized ontologies that help to optimize the user search, and furthermore, to create and refine algorithms that allow us to consider all this information and create routes for each user's needs or tastes.

This paper focuses on the study and the improving of routing algorithms to facilitate customization and meet the needs of users considering factors such as connectivity and the data processing in the routing algorithms. To carry this out, it is necessary to know the efforts that have been done in this subject by the mobile device markets.

The paper is organized as follows: a review of the main devices and the technologies involved is provided in the section 2. In section 3 are described Android, the OS selected to develop the study, and its main APIs for routing. In section 4, there is a summary about the OpenStreetMap project and the presentation of some remarkable applications based on it for mobile devices. Section 5 provides a review of graph theory issues which are used in navigation and routing. In section 6, the four Android projects selected are reviewed to know how they solve the main routing problems. In the following sections, 7 and 8, the improvements from the ideas on the studied projects, are presented, implemented and tested. Finally, the paper closes with the key outcomes from the paper and a brief discussion of the future challenges and direction of this research.

II. MAIN DEVICES AND TECHNOLOGIES INVOLVED

A mobile device is a small, hand-held computing device, typically having a display screen with touch input and/or a miniature keyboard and less than 2 pounds (0.91 kg). Early pocket sized ones were joined in the late 2000s by larger but otherwise similar tablet computers. As in a personal digital assistant (PDA), the input and output are often combined into a touch-screen interface. Nowadays, smartphones and tablets are popular amongst those who wish to use some of the powers of a conventional computer in environments where carrying one would not be practical. Mobile devices such as the iPhone, iPad, Android and others are revolutionizing the way information can be disseminated as explained in [10].

The most common mobile operating systems (OS) used by modern smartphones include Google's Android, Apple's iOS, Microsoft's Windows Phone, Nokia's Symbian, RIM's BlackBerry OS, and embedded Linux distributions such as Maemo and MeeGo. Such operating systems can be installed on many different phone models, and typically each device can receive multiple OS software updates over its lifetime. It is important to know which OS gives more features and facilities to develop this study. In order to choose one, the most important factors to be considered are explained below.

The distinction between smartphones and feature phones can be vague and there is no official definition for what constitutes the difference between them. One of the most significant differences is that the advanced application programming interfaces (APIs) on smartphones for running third-party applications can allow those applications to have better integration with the phone's OS and hardware than is typical with feature phones. In comparison, feature phones more commonly run on proprietary firmware, with third-party software support through platforms such as Java ME or BREW. An additional complication in distinguishing between smartphones and feature phones is that over time the capabilities of new models of feature phones can increase to exceed those of phones that had been promoted as smartphones in the past.

So, it's obvious that the kernel of a SO is the software responsible for providing secure access to the hardware device, responsible for managing resources, and that's why it's important to know which kernel uses each the device. Both Android and Palm are based on Linux. Blackberry is based on a proprietary kernel. Iphone is based on OS X. S60 is based on Symbian and Windows Mobile is based in Windows CE. Both of them are all proprietary software. The main difference between a private or free distribution kernel, is that the kernel which is based in open source software will have a large community of developers, thanks to which it is possible to quickly find errors, make improvements and resolve problems facilitating the rapid adaptability to the needs of end users.

Another important factor to consider is the adaptability of the platform to different terminals. In this item, Android has great adaptability since it is being increasingly used in more mobile devices and not just smartphones.

Finally, the last important factor to consider is connectivity. It's quite important having Wi-Fi Internet access and 3G connectivity to improve data acquisition. All current smartphones offer this possibility with the only limitation of the network are connected and the distributor of phone services.

Android is the platform chosen to carry out this project, for the significantly increased demand in the market of Android devices which is the 43% according to [11]-[12], (figure 2), the availability of the applications, the fact that it is an OS non-dependent on a specific hardware, and in addition, this platform has an enormous community that offers many tools for the development.

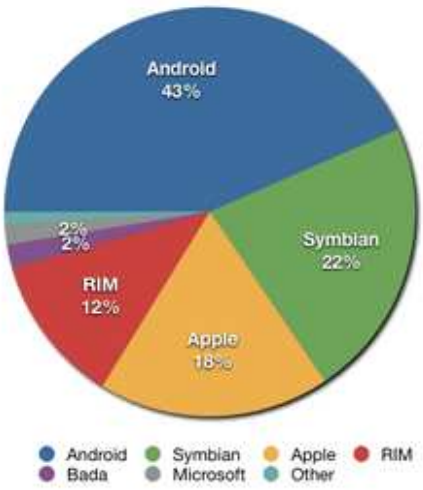

Fig.2. Share of worldwide 2011 smartphones sales to end users by operating system, according to [11]-[12]

III. ANDROID AND ITS MAPPING API'S

Android is a Linux based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance led by Google. Nowadays, Android is spreading their market for other devices such as MP3 players, netbooks, computers, TV's and e-book readers.

Programming on Android is based on APIs. An API (Application Programming Interface) consists of a set of class libraries that can be used quickly and easily. These elements and features for applications can be used and developed without having to start from scratch when starting a new project. There are 2 great API's for programming maps on Android: Android Maps API [13] and OpenStreetMap [14]. In order to carry out this study, the features and characteristics of both API's must be studied to choose which one gives more facilities and better characteristics. Therefore, a little description of both API's is presented here.

Android Maps API is a web mapping service application and technology provided by Google, free (for non-commercial use), that powers many map-based services, including the Google Maps website [5], Google Ride Finder [14], Google Transit [15], and maps embedded on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, car, bike or public transport and an urban business locator for numerous countries around the world. Google Maps satellite images are not updated in real time; they are several months or years old.

In 2006, Google introduced a Java application called Google Maps for Mobile, intended to run on any Java-based phone or mobile device. Many of the web-based site's features are provided in the application. One of its important features is the introduction of a GPS-like location service that does not require a GPS receiver. The "my location" feature works by using the GPS location of the mobile device, if it is available. This information is supplemented by the software determining the nearest wireless networks and cell sites. The software then looks up the location of the cell site using a database of known wireless networks and cell sites. The Cell-site location method is used by triangulating the different signal strengths from different cell transmitters and then using their location property (retrieved from the online cell site database) to aid "My Location" in determining the user's current location. Wireless network location method is calculated by discovering the nearby Wi-Fi hotspots and using their location property (retrieved from the online Wi-Fi database, in the same way as the cell site database) to further discover the user's location. The order in which these take precedence is:

-- GPS-based services.
-- WLAN-, Wi-Fi-based services.
-- Cell transmitter-based services.

The software plots in blue the streets that are available with a yellow icon and a green circle around the estimated range of the cell site based on the transmitter's rated power (among other variables). The estimate is refined using the strength of the cell phone signal to estimate how close to the cell site the mobile device is.

Google Maps Navigation for Android 2.0 is free. The main features provided in the application are:

-- Search in plain English.
-- Search by voice.
-- Traffic view.
-- Search along route.
-- Satellite view.
-- Street View.
-- Car dock mode.

Although, GoogleMaps API is an extension of the Android SDK that provides easy access for Android applications of the Google services geographic data, it has two important problems. First, for proper operation is necessary to use a continuous connection to the Internet as it's stated in [16]. Second, the terms of the license: when using Google Geocoding API, the application of map viewing is subject to a limit of 2500 queries geocode requests per day as explained in [17]. This limit is fixed in order to prevent abuse of the service, after that the API stops working for 24 hours, making it impossible to view the maps.

However, the powerful Google Maps infrastructure has a major limitation: users cannot interact with the geographic data. Instead, they are shown pictures of the data, and any content they add is a separate, and weakly linked, layer [18].

OpenStreetMap (OSM) is a collaborative project to create a free editable map of the world. Two major driving forces behind the establishment and growth of OSM have been restrictions on use or availability of map information across much of the world and the advent of inexpensive portable GPS devices.

The maps are created using data from portable GPS devices, aerial photography, other free sources or simply from local knowledge. Both rendered images and the vector dataset are available for download under a Creative Commons Attribution-ShareAlike 2.0 license. All the contributors of the project must register on the project and must commit to provide data in a Creative Commons BY-SA 2.0 license or choose a compatible license as explained in [19].

The OpenStreetMap approach to mapping was inspired by sites such as Wikipedia; the map display features a prominent 'Edit' tab and a full revision history is maintained. Registered users can upload GPS track logs and edit the vector data using free GIS editing tools like JOSM (Desktop Java editor). Various mobile applications also allow contribution of GPX tracks to the OSM project.

Due to its simplicity for creating maps and generating routes even without network connection, the free access to the source code and the real geographic data and finally, its use in multiple mobile devices applications, this work will be implemented in the OSM environment.

IV. OSM PROJECT

As mentioned in [20], "OSM follows the peer production model that created Wikipedia; its aim is to create a set of map data that's free to use, editable, and licensed under new copyright schemes". This project was born at the University College of London in July 2004. The main purpose is, as stated in [19]: "The project was started because most maps you think of as free actually have legal or technical restrictions on their use, holding back people from using them in creative, productive, or unexpected ways".

A considerable number of contributors edit the world map collaboratively using the OSM technical infrastructure, and a core group, estimated at approximately 50 volunteers, dedicate their time to create and improve OSM's infrastructure, including maintaining the server, writing the core software that handles the transactions with the server, and creating cartographical outputs. There's also a growing community of software developers who develop software tools to make OSM data available for further use across different application domains, software platforms, and hardware devices. However, the main project is the OSM Web site, which can be seen in figure 3 below.

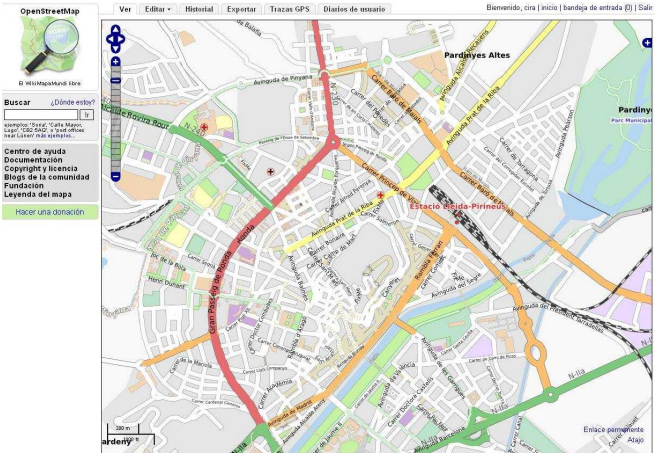

Fig.3. Capture of the OSM Web site with the basic information of the town of Lleida.

Furthermore, new projects have been developed in OSM with special functionalities such as: GreenGPS [21], a sensing fuel-efficient maps application; the STRIVE OSM Project [22], an application to deliver location-based environmental information in Ireland; and UN Spatial Data Infrastructure for Transportation [23], a collaborative mapping for emergency routing for disaster logistics used in Haiti earthquake and UN portal for Africa.

A. *Editing Tools*

User-contributed geographical information is a core part of OSM, and the OSM developer community has made a great effort to implement tools to facilitate user contributions to the database.

For most casual contributors, the OSM Web site offers a lightweight online Flash-based editor, Potlatch 2 [24], which lets users add, update, or delete geographical features through a relatively easy-to-use interface. As can be seen in figure 4, user can choose between lists of tags, ordered by type, select one, then select its position on the map and edit the appropriate information.

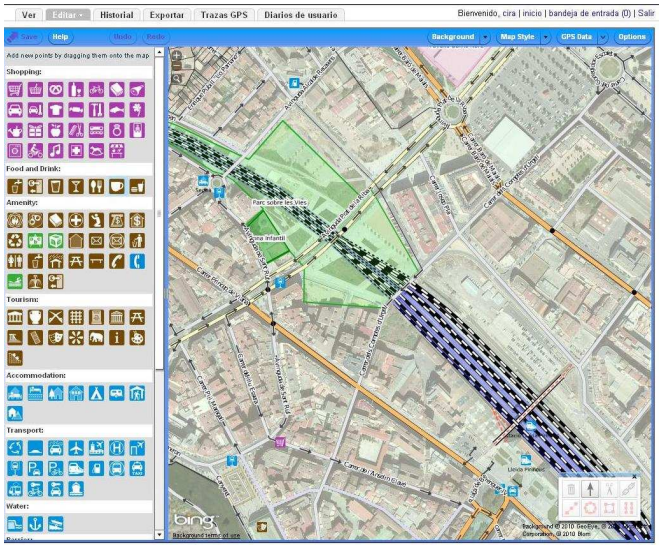

Fig.4. Capture of the web interface of Portlach 2 Edition Tool.

So the interface is kept deliberately simple, with more advanced functionality provided through keyboard shortcuts; Potlatch gives extensive guidance to users by providing predefined tagging schemas for frequently occurring features (such as motorways or primary roads). Potlatch also lets users upload and integrate GPX tracks recorded from handheld GPS units.

More experienced OSM contributors also use the JOSM, an editing suite with an interface similar to traditional GIS packages. The application lets users import, edit, and tag OSM data offline and allows uploads of OSM updates through the OSM API.

Moreover, apart from individual user contributions from GPS tracks and the digitizing of aerial imagery, OSM has also taken advantage of the availability of free geographical information in certain parts of the world. For example, the most important cases explained in [20] which are the commercial navigation information provider AND (Autonomous Navigation Data) and the Isle of Man's Department for Local Government and the Environment that donated the entire street map information of the Netherlands and the Isle of Main to the project.

B. *Technical Infrastructure*

The heart of OSM's technical infrastructure lies in the central database holding the live data, which is implemented in MySQL. The database schema is designed to support wiki environments, such as versioning and rollbacks, and keeps copies of modified and deleted features indefinitely.

All geographical entities are recorded as points (nodes), which contain the latitude and longitude coordinates along with user name and timestamp information. Linear and area features are defined by reference to a list of ordered nodes, called ways. Area features are not explicitly defined in the database schema, rather, they're defined implicitly by the condition of a way that's closed (the first node of a way is the same as the last one) and explicit tagging conventions (using the *tag area=yes*).

As explained in [25], "OpenStreetMap's internal files are lists of nodes, ways and relations, which can be tagged with information about the respective map element". Any user is free to introduce its own tags, but it is recommended to use existing tags and only have new ones if they are not already covered by the existing ones. The tags of the map elements are represented as (key, value) pairs. An element of the map may have multiple tags as can be seen, for example, in line 6 of figure 5, where the key (k) is "addr:city" and the value (v) is "Bremen":

```
<node id="834034642"
  lat="53.0871310" lon="8.8091071"
  version="7" changeset="6027662"
  user="Kerridge" uid="324245"
  timestamp="2010-10-13T09:51:39Z">
  <tag k="addr:city" v="Bremen" />
  <tag k="addr:country" v="DE" />
  <tag k="addr:housenumber" v="20" />
  <tag k="addr:postcode" v="28215" />
  <tag k="addr:street" v="Theodor-Heuss-Allee" />
  <tag k="amenity" v="charging_station" />
  <tag k="name" v="Elektrotankstelle swb" />
  <tag k="note" v="telephone reservation necessary" />
  <tag k="opening_hours" v="Mo-Fr 6:00-18:00; Sa off; Su off" />
  <tag k="operator" v="swb" />
  <tag k="phone" v="+49 421 3593186" />
</node>
```

Fig.5. Example of an OSM node with its tags in an XML representation as seen in [25]

On the other hand, as we can see in the example, the OSM uses a topologic schema of data. The basic elements in the cartography are:

-- Nodes: points that include the geographic location given.

-- Routes (Ways): ordered list of nodes representing a polyline or polygon (when it begins and ends at the same point).

-- Relations (Relations): groups of nodes, paths and other relationships that can be assigned to the particular properties.

-- Labels (tags) can be assigned to nodes, paths or relationships and consist of a key and value. For example, highway = trunk.

C. *Mapping Outputs*

The main cartographic output from the OSM information is presented on the OSM Web site as a Google Maps-like interface, named "Slippy Map", which uses the open source AJAX library OpenLayers to automatically update the map display and allow interaction with users. As users drag the map, the visible extent is updated and new map tiles are requested in the background without reloading the entire HTML page.

A search function, implemented as an external Web service, lets users to quickly find cities, villages, or other POI's in the database. There is also added an export tab, that lets users to quickly generate map images, PDF files, and raw data downloads of custom bounding boxes. The default set of tiles on the main OSM Web site is rendered using Mapnik, an open source library for generating high-quality map images. It uses a weekly database dump as the source for the rendering of map tiles, given that live rendering of tiles on the clients request would be too computationally expensive to be practical; map tiles are rendered for all zoom levels and saved on the server so that they can be served rapidly as static images.

Thanks to the open source nature of all the tools needed for map rendering, several OSM contributors have developed custom map tile sets that cover specific needs of use, for example, a tile set that highlights cycle-path networks and other relevant information to cyclists.

D. *OSM Routing*

OSM data includes information for routing by many modes including car, foot, bicycle and horse. There are many offline, embedded and web-based routing services using OSM data. One of the most remarkable is OpenRouteService (ORS) [26]. This project has been the base for most projects in routing in OMS. ORS is a route service operating on OSM data that was launched in April 2008. This service available on web, implement open standards of the Open Geospatial Consortium (OGC), mainly OpenGis Location Services 1.1. ORS was the first national route planner for pedestrian or bicycle routes even before the companies like Google.

OSR's routing algorithms are based in the implementation of the combination of A* and Dijkstra's algorithms. However, its main purpose is giving routing services for travelling and navigation and other information such as POI's, is not considered. The data processing is done in the web server using the database information. In figure 6 [26], the main components and structure of the ORS are shown, divided by typology: the viewer tools (OpenLayers), the services involved in all the procedure (OpenLS and the Geoserver) and finally, the renderer (Databases and OSM data).

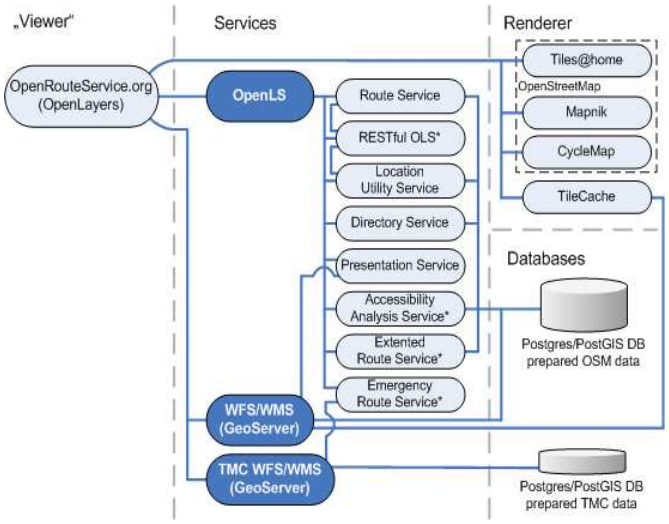

Fig.6. Schema of the components in ORS as seen in [26].

OSR presents some problems, which also affect many OSM routing projects, in order to offer a reliable and successful routing service. The core of each routing application is the routing graph. The routing graph represents the street network as a model of nodes and edges. It is important that the graph is built from a topologically correct dataset: junctions are represented as nodes and streets are represented as edges between them. Junctions are only recognized as such, if the crossing streets have a common node at their intersection. During the collaborative work, as explained in [27] the intersections and junctions were not well defined, so the existing topology of OSM data is examined regarding the occurrence of street intersections with common nodes, junctions, and at those common nodes, streets are divided into individual edges or ways. The number of edges in the routing graph is usually higher than the number of streets in the original dataset. In figure 7 an example of this situation is shown: OSM data on the left shows 2 streets divided into 9 different nodes; and the ORS data, on the right shows only 4 nodes to define the same 2 streets.

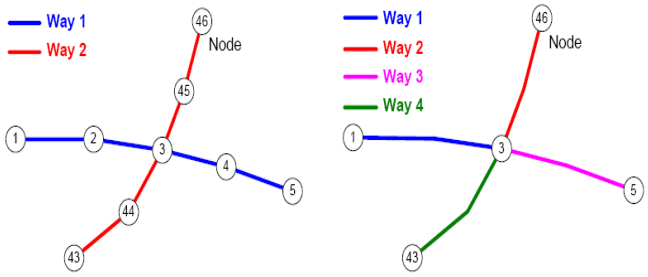

Fig.7. Example of building the routing graph from OSM data taken from [27]: on the left there is the original OSM data, on the right the modified OSM data.

Moreover, another problem is the inconsistency in the OSM data. Unfortunately, in many regions the data are almost nonexistent or not well defined, because of the use of abbreviations or different spelling for the same street or POI. For this reason, the results are not detailed enough and are not fully reliable, given that often lack basic information such as the number of streets and the position of the numbers. However, the increasing amount of the volunteers and contributors in the OSM project, are improving the data quality and moreover, increasing the amount of information in most areas.

E. OSM Projects for Mobile Devices

Currently there are multiple projects that use OSM developed on Android and also provide searches that can be performed without Internet connectivity. Some of the most popular are: Navit [28], GpsMid [29], OSMAnd [30] and VGPS [31], as seen in [19].

Navit is a multiplatform project that mainly focuses on navigation functions performed by vehicle. Navit is one of the most successful applications because it offers the functions of a GPS even having no connectivity and it is also translated to 49 languages, including voice directions.

GpsMid project is also multiplatform, including Android, that offers navigation and also the option to view and search for POI's. It does not only allow the navigation but it also offers the option to edit maps and customize them, and thus actively collaborate with OSM. It is licensed under GPLv2.

OSMAnd is a fully open OSM-based navigation application for Android. Its main features are: display vector OSM maps and tile maps (using Mapnik), supports layers for several types of POI's and let the addition of them into OSM databases, gives navigation with or without connection, voice navigation and uses ORS to perform the online routing, and YOURS or CloudMade for offline routing. It also allows edition and customization of the maps. Moreover, not only it serves navigation and routing by vehicle, also for pedrestrians.

Finally, VGPS is also a multi-platform project that allows addition of GPS off-line navigation and also allows geocatching but only for personal use. However, the routing is focused in travelling by car or any other motor vehicle.

Studying the different technologies used in these projects, the work and the treatment of data with different search algorithms (mainly A* and Dijsktra algorithms) and finally, the way the impedances are considered to favor the search for different user preferences, is the key to improve the search algorithms involved in the routing to get better and customized routes.

V. GRAPH THEORY ISSUES

The routing algorithms on OSM are mainly based on search algorithms in graphs. A graph is an abstract representation of a set of objects where some pairs of the objects are connected by links. The interconnected objects are represented by mathematical abstractions called vertices, and the links that connect some pairs of vertices are called edges. Typically, a graph is depicted in diagrammatic form as a set of dots for the vertices, joined by lines or curves for the edges. An example of a graph can be seen in figure 8, with 6 vertices and 7 edges.

Graphs are one of the objects of study in discrete mathematics.

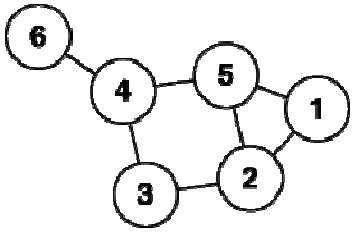

Fig.8. Example of a scheme of a graph in [32]

The typical structure in which the information is stored in a graph depends on the graph itself and the algorithm used to manipulate it. Some structures are simple and use lists and arrays or matrix, although, as pointed in [33] there is often a combination of these structures.

The two algorithms used in most applications based on OSM routing algorithm is A* and Dijkstra's algorithms, although the application Navit, also used Dijkstra's algorithm with Fibonacci as seen in [19].

A* is an algorithm widely used in path finding and graph traversal, the process of plotting an efficiently traversable path between points, called nodes (denoted as *n*). The main problem with some search algorithms in graphs, such as the greedy algorithm, which is guided exclusively by the heuristic function, is that may not indicate the path of lowest cost, or the actual cost of travel to one node to another (such as climbing algorithms) and can be given if necessary to make a move to achieve greater cost solution. Therefore it is quite intuitive that a good algorithm research should take into account two factors, the heuristic value of the nodes and the actual cost of the tour.

To avoid that problem, A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals).

It uses a distance-plus-cost heuristic function (usually denoted *f(n)*) to determine the order in which the search visits nodes in the tree. The distance-plus-cost heuristic is a sum of two functions:

-- the path-cost function, which is the cost from the starting node to the current node (usually denoted *g(n)*)

-- and an admissible "heuristic estimate" of the distance to the goal (usually denoted *h(n)*).

The *h'(n)* part of the *f(n)* function must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, *h(n)* might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.

In addition, the A* algorithm maintains two auxiliary data structures that can be called open, implemented as a priority queue (ordered by the value *f (n)* of each node), and closed, which keeps information nodes already visited. At each step of the algorithm, the node is expanded first to open, and if that is not a goal node, calculate *f (n)* of all of its children and insert them to open and mark the evaluated node to closed.

The search algorithm is a combination of the breadth-first search (BFS) and the depth-first search (DFS): while *h'(n)* tends to first in depth, *g (n)* tends to first in width. This way, the way of research changes whenever nodes are most promising. An example of the A* search algorithm is shown in figure 9 below, where the green square marks the starting point, the red squares show the path followed, the grey ones mark the obstacle, and finally, the blue square denotes the destination.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

Fig.9. Schema of a search using A* algorithm as shown in [32]

If the heuristic *h* satisfies the additional condition $h(x) <= d(x,y)+h(y)$ for every edge *x*, *y* of the graph (where d denotes the length of that edge), then *h* is called monotone, or consistent. In such a case, A* can be implemented more efficiently because no node needs to be processed more than once, and A* is equivalent to running Dijkstra's algorithm with the reduced cost $d'(x,y) = d(x,y) - h(x) + h(y)$.

The space required for A* algorithm to be executed is the biggest problem. As is well possible to store all nodes in each state, the amount of memory required is exponential with respect to the size of the problem. To solve this problem several variations of this algorithm have been proposed and its performance combined with the Dijkstra's algorithm.

Dijkstra's algorithm, as defined in [33], also called the shortest path algorithm, is an algorithm to determine the shortest path given a source vertex to other vertices in a graph with weights on each edge. Its name refers to Edsger Dijkstra, who was first in describing it in 1959.

The idea behind this algorithm is to go exploring all paths shorter from the vertex origin and leading to all other vertexes, and when the shortest path from source vertex to other vertexes, that build the graph, are obtained the algorithm stops. The algorithm is a specialization of the uniform cost search, and as such, does not work on graphs with edges of negative cost (in always choosing the node with less distance may be excluded from the search nodes in future iterations will decrease overall cost of the path passing through an edge with negative cost).

The algorithm would work as follows: given a weighted directed graph of *N* nodes not isolated, *X* is the initial node, a vector *D* of size *N* will store at the end of the algorithm the distances from *X* to other nodes, and the following steps will continue:

-- Step 1: Initialize all distances in *D* with an infinite value on the top because they are unknown, except for *X* that has to be placed at 0 due to the distance from *X* to *X* is 0.

-- Step 2: Let $a = X$ (to take as the current node).

-- Step 3: Travel all the nodes adjacent to a, except the nodes that have been already treated or completed, calling these nodes $v_{i.}$

-- Step 4: If the distance from *X* until it is stored in *D* is greater than the distance from *X* to *a*, added to the distance from *a* to v_i; then it is replaced with the new distance, this is: if $(D_i > D_a + d\ (a, v_i))$ then $D_i = D_a + d\ (a, v_i)$.

-- Step 5: Mark as completed node *a*.

-- Step 6: Take the current node as the next node with the lower value in D (this can be done storing the values in a priority queue) and return to step 3 while there are nodes which are not marked.

Once the algorithm has finished, D is completely full.

In figure 10 below, an example of the Dijkstra's algorithm completed route is shown. In the schema, every node is noted with a letter and the distance between the nodes, is already calculated with a value number. The problem lies in knowing which is the shortest path between nodes *a* and *z*. After the execution of the Dijkstra's algorithm, the shortest path is ADCBFEZ, marked in red, and the total distance is 23.

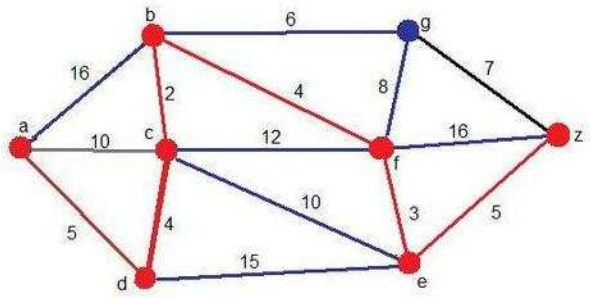

Fig.10. Schema of a search using Dijkstra's algorithm as shown in [32]

The combination of the A* algorithm and Dijkstra's algorithm, get on reducing the high computational cost of the A* algorithm while minimizing some cases search are simpler to use the Dijkstra's algorithm because the road is almost direct. So, the combination of both algorithms is perfect for routing in mobile devices because of their actual limitations of the hardware: low memory and low CPU.

VI. STUDY AND ANALYSIS OF THE PROJECTS

As seen in previous sections, there are several problems in routing and navigating in mobile devices. These are mainly the limitations of the hardware of these devices, the lack of connection in some areas, the quality of the data and finally the data acquisition and treatment to develop the routes. From now and on, the following sections will focus in solving the main problems that directly are involved in the routing calculation to improve and optimize the searches. In order to achieve this goal, a previous study of the solutions offered by the main projects in Android mobile devices, is necessary.

The main issues that directly affect the calculation of routes are the connectivity, the quality of data and the data acquisition and treatment in the routing algorithms. The use of the combination of Dijkstra's and A* algorithms solve the hardware limitations problem, for this reason, a modification or new combination is not proposed. However, the way in which data is acquired and treated by those algorithms, is an importance subject to improve the routing and customize searches.

The four projects selected are Navit, GpsMid, VGPS and OSMAnd because of their popularity and importance as pointed in [3] and [19]. For the connection issue and the quality of data, all four projects provide a similar solution. However, the acquisition and treatment of the data, differs in each project in some aspects.

An important aspect to be considered is the purpose of each project. Both VGPS and Navit, are applications designed to calculate routes for vehicles, to travel by car or any other motor vehicle. These projects only include POI's to be displayed on the maps, but there is no real treatment of that information in the routing algorithms. For that reason, these two projects will only be considered for the connectivity problem.

A. *The Connectivity Problem*

An Internet connection is a requirement for most of the routing systems. The mobile device running this type of applications must be connected to the Internet and enable GPS features in order to give the exact position and control the movement during the route. However, neither the internet nor the GPS coverage, are not always available. Moreover, the long term connection also implies the waste of the battery which is quite limited in these devices.

To solve this issue, all four projects, allow the off-line routing. To achieve it, user must download the maps of the area that is going to visit and store it in the device memory.

Nevertheless, these maps include the information of big regions, even full countries, becoming big files which must be stored in the limited storage memory of the mobile device. For example, OSMAnd maps can be downloaded from the project web [30], but as seen in figure 11 below, every file includes the information of a full country, making it too heavy to be stored in the device main memory.

Filename ▾	Summary + Labels ▾	Uploaded ▾	Size ▾
Benelux_europe_1.obf.zip-2	Map, POI, Transport, Address index for Benelux europe {13.11.2011 : 454.7 MB} Testdata	Nov 13	190 MB
Benelux_europe_1.obf.zip-1	Map, POI, Transport, Address index for Benelux europe {13.11.2011 : 454.7 MB} Testdata	Nov 13	190 MB
Italy_europe_1.obf.zip-1	Map, POI, Transport, Address index for Italy europe {14.11.2011 : 236.6 MB} Testdata	Nov 15	190 MB
Netherlands_europe_1.obf.zip-1	Map, POI, Transport, Address index for Netherlands europe {14.11.2011 : 347.6 MB} Testdata	Nov 15	190 MB
Czech_republic_europe_1.obf.zip	Map, POI, Transport, Address index for Czech republic europe {14.11.2011 : 184 MB} Testdata	Jan 6	184 MB

Fig.11. Capture of the maps repository of OSMAnd project in its website [30].

Another example is the repository of GpsMid project, shown in figure 12, which even offering the maps by city extracts, the weight can be of 47MB.

City extracts

Area	Files	Last mod.	Size
Amsterdam	jad - jar - zip	Monday December 12, 2011	13M
Amsterdam-Rotterdam	jad - jar - zip	Monday December 12, 2011	42M
Berlin	jad - jar - zip	Monday December 12, 2011	27M
Boulder	jad - jar - zip	Sunday December 11, 2011	2.8M
Greater London	jad - jar - zip	Monday December 12, 2011	45M
Hamburg	jad - jar - zip	Monday December 12, 2011	28M
Helsinki	jad - jar - zip	Monday December 12, 2011	30M
Karlsruhe	jad - jar - zip	Monday December 12, 2011	16M
Koeln/Bonn	jad - jar - zip	Monday December 12, 2011	45M
Krakow/Katowice	jad - jar - zip	Monday December 12, 2011	20M
Leipzig	jad - jar - zip	Monday December 12, 2011	14M
Lodz	jad - jar - zip	Monday December 12, 2011	5.0M
Los Angeles	jad - jar - zip	Sunday December 11, 2011	41M
Madrid	jad - jar - zip	Sunday December 11, 2011	21M
Milano/Bergamo	jad - jar - zip	Monday December 12, 2011	25M
Moscow	jad - jar - zip	Monday December 12, 2011	42M
Muenchen	jad - jar - zip	Monday December 12, 2011	47M

Fig.12. Capture of the maps repository of GpsMid project in its website [29].

In order to not collapse the main memory of the mobile device, each project offers different solutions.

GpsMid and VGPS obtain the data in several ways:

-- downloading it from Planet OSM [34], which is a web repository of all OSM data in one file. There are also files called *Extracts* which contain OSM data for individual continents, countries and important metropolitan areas. However, it is not recommended because of the big weight of the files and the repository is quite big, so user can get lost in it.

-- downloading it with JOSM [35], which allows the selection and extraction of OSM data of the user selected areas. This process includes the installation of JOSM and exporting the data to an XML file.

-- downloading it directly in the OSM site [6], first searching the area and then exporting the data into an XML file.

Once the user has the XML file, this has to be converted in the formats that fit for each application. As explained in [31], VGPS converts the files in java libraries using the VGPS Map Generator. As seen in figure 13 below, this process is a little bit complicated for novel users, because VGPS Map Generator works using commands in the terminal.

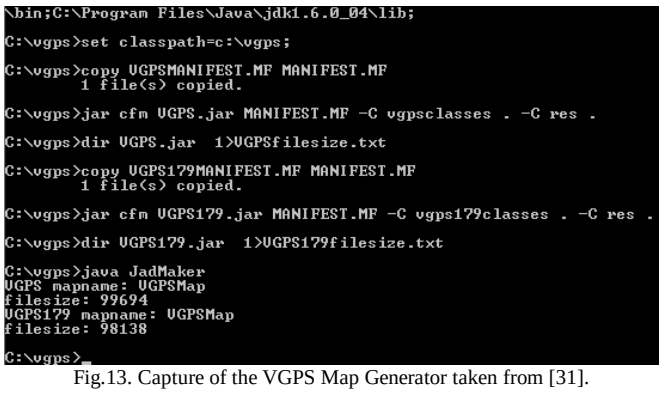

Fig.13. Capture of the VGPS Map Generator taken from [31].

GpsMid does not need any conversion and accepts OSM data files in both *.xml* and *.osm* extensions.

Navit's map files can be downloaded from Garmin site [35] or OSM site [6] in XML format, as GpsMid or VGPS, no conversion is needed. Garmin is a private company that develops consumer, aviation, and marine technologies for GPS. This company has the intellectual property of all their maps. However, according to [28], this company has given access to some of their maps, like the MetroGuide Europe which is open and can be used freely.

OSMAnd project offers their maps in .obf extension, compressed into ZIP files, in the repository of the project site [30]. User can also download directly the data from OMS webpage in XML format, and use the OSMAndMapCreator application to convert the files into .obf. Morever, this project allows to charge GPX files which is quite useful to make routes offline. The main problem is the big weight of the files, for example, if a user wants to download the map data of a city of Spain, must download the full Spain map, and its weight's, even compressed into a ZIP file is 43.1 MB. To minimize the problem, OSMAnd allows choosing to store the data in the memory of the device or in the SD target and moreover, the place in which download it. This facilitates the storage because user can download the maps both in a computer and then insert the SD target to the device, or directly in the mobile device, and then, decide if the maps must be stored in the SD target or in the main memory.

B. The Quality of Data

The availability and accuracy of spatial data is important for these applications to work efficiently, effectively, and correctly. As explained in previous sections, OSM coverage is not uniform across all the territory and the best mapped zones are cities and towns. The lack of spatial data in a particular region can lead to these systems not performing well.

Nevertheless, the popularity of OSM is growing quickly and the diversity and quantity of the points of interest provided offer new opportunities and challenges in creating customized and detailed visualization of cities.

To collaborate in the project and moreover improve the quality and quantity of the OSM data, both GPSmid and OSMAnd implement the functionality to set data and track routes to be uploaded to the OSM project automatically. The only limitations are the tags provided in both applications. So if the users do not find a predefined tag that matches the information they want to store, the only way is saving the track and the position in the mobile device and upload it in the OSM web page.

The process of recording a track is quite easy in both applications. The user only has to select the "record track" option on the menu and follow the instructions. The edition of every point is quite easy, as seen in the figure 14, where first user selects the coordinates and then, specifies the information to edit and choose the appropriate tags.

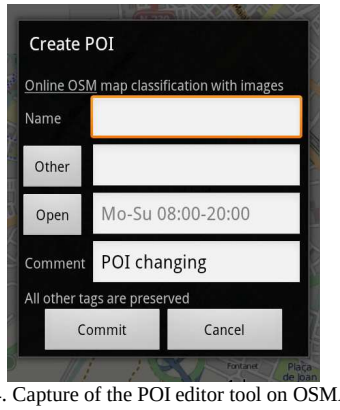

Fig.14. Capture of the POI editor tool on OSMAnd.

C. *The Data Acquisition and Treatment*

In both projects, the searches can be done in two different ways. The first one is a simple text-based interaction for free search, similar to the one existing in tools like Google Maps or OpenStreetMap, where user selects "Search" option and writes down then name of the address or POI.

In this case of search, the text input by the user needs to undergo a process of analysis which extracts from the query the concepts which are matched. There are several problems with that type of searching. The main problem is that in this direct search depends on how the tags have been formalized in the OSM project. For example, in the region of Catalonia, there are two official languages: Spanish and Catalan. If a user searches for a specific street, has to write it in Catalan if it was introduced in Catalan, and the search in Spanish does not offer any results. Moreover, the tags may not be well defined. For example, if a user does not write it in the same way it's written and defined, there may not be a good result. This can be seen in figure 15: if the user searches for "*Passeig de Ronda, Lleida, Cataluña, Spain*", there are no results. But if the user searches for "*Gran Passeig de Ronda, Cataluña*", user can choose between the following results: "*Gran Passeig de Ronda, Parc, Gualda, Llivia, Cataluña, 25171, España*" and "*Gran Passeig de Ronda, Parc, Gualda, Llivia, Cataluña, España*", which are the same street in Lleida, but instead of using the name of the town, appears the name of two neighborhoods (Gualda and Llivia).

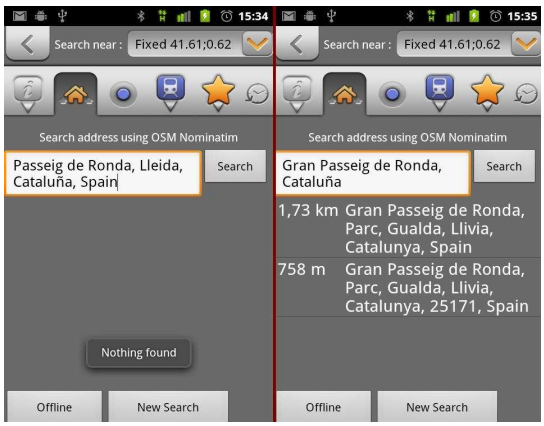

Fig.15. Capture of the two search cases in OSMAnd.

Another problem is the lack of data, explained in the previous sections of this paper. And the other problem is that the data has not yet been indexed. The main responsible for indexing the data in OSM project is Nominatim [37]. Nominatim is s a tool to search OSM data by name and address and to generate synthetic addresses of OSM points. Usually, the indexing process takes about 2 days. However, there is no update for some areas; for example, the town of Lleida (Catalonia, Spain) has not been updated since 2007 [37]. For this reason, if user searches for a specific street or POI using internet connection, may not find any result. However, using the offline mode, user can update its area file and then have the actual data.

The second type of search provides a better structuring of the query with the help of a tag selection. As seen in figure 16, user selects the option search by POI's and can choose between a big list that includes: the search by name, predetermined data groupings of POI's (car assistance, for tourists, etc.), the nearest POI's from the location and of course, by the OSM categories: Transport, Education, Nature, etc.

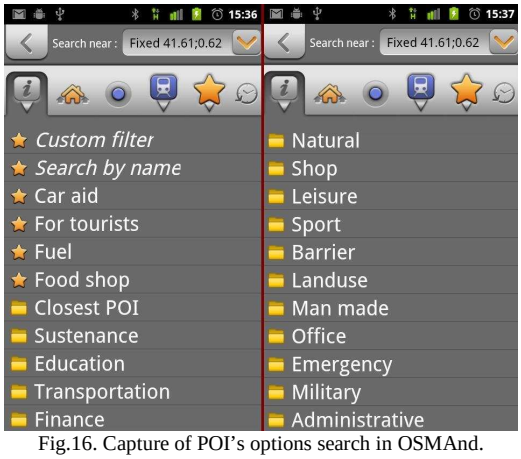

Fig.16. Capture of POI's options search in OSMAnd.

So, while the first interface is more intuitive, the second has the advantage that it is easier to relate with the concepts in the ontology of the POI's. Moreover, user only has to select the correct name of the POI instead of writing it, which eliminates any problem of the nomenclature or definition.

In both projects, once the point of search has been defined (address or POI) the way in which the search is done, differs. On one hand, if it's an online search using the connectivity characteristics of the device, internet connection and GPS, the process will involve a continuous exchange of data between the server and the application in the mobile device. As seen in figure 17, the application will send the data to the server for both types of search, the text input search named *type 1* and the POI's selection search, named *type 2*. The search will be done on the OSM server, as well as the routing calculation. The result and the calculated route information will be sent to the devices and shown in the application map.

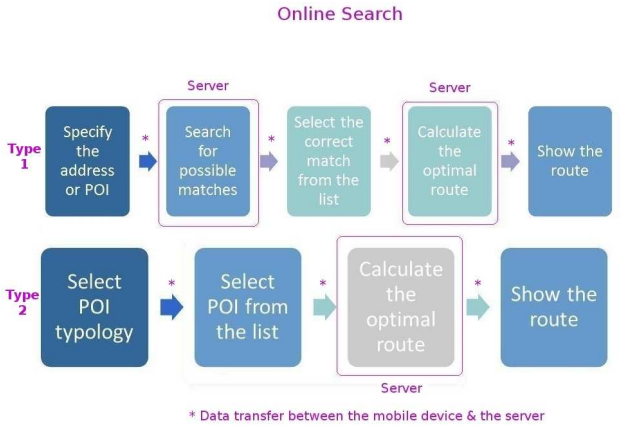

Fig.17. Schema of the online search process considering both typologies of search. The squared procedures are completely performed on the server.

On the other hand, if the search is done without connectivity, the mobile device will be responsible of all the process, as seen in figure 18. Firstly, the data for the searching will be taken from the data files stored in the memory of the device. Secondly, the calculation route will be generated by the application. And finally, the application will show the routing search result in the map of the application.

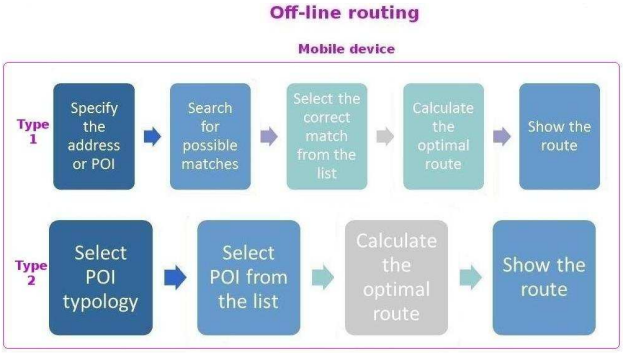

Fig.18. Schema of the offline search process considering both typologies of search.

VII. PROPOSALS FOR IMPROVEMENT

As seen in section 6, all four projects (Navit, GpsMid, VGPS and OSMAnd) have done great efforts to solve the main problems. However, there are several points, in which improvements can be made, in order to get faster, more customized and more optimized routes.

These solutions involve the 3 main problems: the connectivity problem, the quality of data and finally, the data acquisition and treatment. The main goal to be achieved is solving and minimizing the problems, considering aspects such as time, CPU load, memory use and the quality of the results per search. Besides, the quality of the results will also include the improvements in order to achieve the user's needs. These proposals for improvements are explained below for every problem.

A. The Connectivity Problem

The lack of connectivity is already solved using preloaded data which is been downloaded from every project site or directly from the OSM web. However, the files obtained are quite big to store them in the mobile devices main memory. Moreover, its weight also influences negatively in the searches because of the great amount of data that must be considered and treated before getting the correct match. The more data involved, the more time is needed to locate the point, the more memory is used and of course, the more CPU is also used.

To solve this problem, data files must be reduced. For example, instead of having all the information of a full country or continent in one file, this information can be divided into small parts involving only the information of a particular city or region.

It is important to note that this solution does not provide any improvement when the search is completely carried out with connectivity. However, the results obtained may be profitable and can be taken into account for future developments that work out how the information is stored on the servers.

B. The Quality of Data

As explained in previous sections, OSM coverage is not uniform across all areas. The lack of spatial data in a particular region can lead to the system not performing well. For example, the town of Lleida (Catalunya, Spain), has a lack of basic information such as streets names or POI's. Moreover, there are areas that are not well defined: incorrect names of streets, squares or POI's, incorrect locations of POI's and streets, etc. So there is a need of having available and correct data.

In order to solve that problem, there are mainly two ways. The first one involves the use of the Portlach Edition Tool (or other editing tool depending in the expertise of the user) in the OSM webpage. The web user interface of this tool is quite easy to use and quite intuitive. Moreover, the changes can be seen immediately into the OSM website.

Nevertheless, improving the data in the OSM website may not be enough: depending on the area, the information is only stored but not indexed for the searching tools, as seen in previous sections. So even if the appropriate changes are made, searches will not be optimized.

The second way to solve the problem of coverage is to optimize the data using the mobile application. GpsMid and OSMAnd projects allow creating POI's and tracks and then uploading them in the OSM project. And what is more important, the new data is stored in the projects maps and updated regularly. This information will be taken into account in the off-line searches. However, the depuration and correction of the data is not an available and direct option from the mobile applications. To solve that problem, editing the map information and correcting it, is the only way.

C. The Data Acquisition and Treatment

In mobile devices, as seen in section 6, searches acquire and process the information depending on two aspects: the available connection and the type of information which is being searched. The lack of 3G or HSDPA (3.5G) coverage can not be solved because depends on the service provider. However, in both methodologies, the availability and correctness of the data stored and indexed in the server and

stored in the mobile device, can be solved as seen in the previous subsections.

Nevertheless, there are some aspects that can be considered to be improved related to the data. These aspects consider the customization and optimization of the searches to cover the final user needs: the ontology used and the groups of tags by activity. As seen in section 6.c, OSMAnd and GpsMid projects use the ontology formalized in the OSM project and offers the POI's search organized by it.

However, this classification of data is not well structured and furthermore, is not really intuitive. For example, the *History* and *Tourist* themes can be confusing: a user may expect to find in the *Tourist* theme, the historical monuments of the area, instead of information of hotels or tourist information offices. Another aspect to be considered is the lack of valuable information about the POI's. Information such as the schedule of the POI should be offered. This way the user does not waste time going to a place which can be closed. So this information is quite important in order to cover user needs and moreover, to optimize searches. Thus, there is not only a saving in user's time, but also there is a saving of searches in terms of exchange of information (data acquisition, queries and displays) between the server and the mobile device (in available connection searches) or a saving of the resources in the mobile device (in offline searches).

As well as the typology can not be easily modified because depends on the OSM project, the improvement of the tags information stored in the map and POI files of some projects can be optimized. For example, OSMAnd offers the option of storing the schedule information of POI's, as can be seen in figure 14. However, this information is not used by the application and not even filled for most of the POI's. So, filling the information of these tags and the implementation of the functionality that considers them is the improvement proposed to this problem.

D. Methodology Implementation

In order to implement and test the improvements proposed in the three previous sections, there are some methodology issues that must be considered and explained.

To be able to carry out the proposed improvements and their testing, the development is done in a clone application of one of the projects studied in the previous sections. This clone application includes the changes proposed to optimize and customize the searches. These improvements involve:

1. Reducing the existing data files size.
2. Creating and editing quality data files for searching.
3. Developing the functionalities to carry out the testing.
4. Developing the functionalities to customize the searches.

The Android project selected between the studied projects in previous sections is OSMAnd project [30]. This selection is due to the fact that it's the only fully open OpenStreetMap-base navigation application for Android. Other projects such as GpsMid are working on the migration to Android, but there are several bugs and problems that have not been solved yet and do not work properly in Android platforms. Moreover, OSMAnd allows OSM data contributions and works the POI's data, not only displaying it but editing and searching by it. Another advantage is that the project has the goal to create comfortable navigation/routing application for Android mobile devices that have limitation with internal, operating memory and processor resources. Also application has a goal to economy internet usage or do not use it at all (using preloaded data), that's why offline features are more prioritized than online. As most of the improvements proposed are going to be displayed using offline characteristics, this project is the one that fits the main needs of the improvement proposals.

The implementation and testing will be done in an Android mobile device. The selected model to perform all the improvements is an HTC Desire, running in Android OS version 2.2. The main hardware characteristics of this device as seen in [38] are:

-- CPU: 1 GHz.
-- ROM Memory: 512 MB.
-- SD Memory Target: 4 GB.
-- Screen: AMOLED 3.7 inches and resolution of 800x480.
-- Other characteristics enabled: GPS, 3G, Wi-Fi and Bluetooth connectivity.

The connection between the mobile device and the computer is done by USB.

The development environment used is the SDK Android [39] and its tools for developing in Android 2.2 Platform.

The environment to develop the source code is Eclipse Indigo for Java Developers [40] performed in the OS Linux distribution Ubuntu [41] version Oneiric.

For compiling the OSMAnd clone project, the programming languages have to be specified. Mainly the source code is written in Java, but there are some features that include C++ language. Moreover, the features to treat the data stored in *.obf* format are mainly implemented in JavaEE and XML.

The OSMAnd clone project can be downloaded from the web site of the project using *git* [42], a version control system. Once downloaded and compiled, the clone project can be exported to the Eclipse SDK to work with the source code, as seen in capture 19 below.

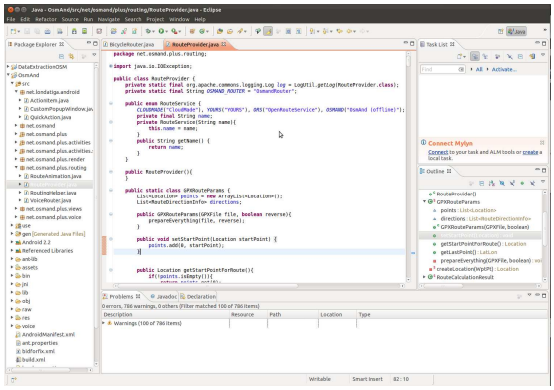

Fig.19. Capture of the OSMAnd source compiled in Eclipse.

As explained in previous sections, OSMAnd data is stored in .obf (OSMAnd binary format) format for the mapping data and the POI's data. This extension file can be edited using SQlite database [43] or the OSMAnd application. All the instructions to create, edit and upload the data sources are available in [44].

As pointed in previous sections, OSMAnd enables routing offline if previously the track is stored in the SD memory card. This type of search is based in POI's to locate the places. So the tracks have been recorded and stored in the SD memory card of the device for the testing in GPX format. These tracks have been recorded using OSMAnd.

The OSMAnd maps available online for the area of the city of Lleida are included in the files named *Europe_Spain.obf* (166MB) and *Catalonian_wiki.obf (156MB)*. *Catalonian_wiki.obf* is reduced to the area of the province of Lleida. Moreover, some POI's of the town of Lleida have been edited and corrected. The weight of the final file, named Lleida.obf, is about 29 MB, almost a 19% less than the original file.

Finally, all the screenshots are taken using the default screenshot application on Ubuntu. The screenshots on the HTC device are taken using Dalvik Debug Monitor [45] (DDM). This monitoring tool is also the main responsible to show the HTC hardware use. In figure 20 below, the DDM is monitoring the Memory use of the device.

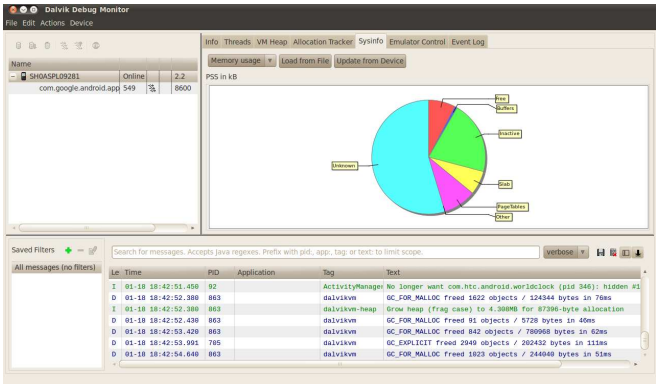

Fig.20. Capture of the Dalvik Debug Monitor, monitoring the HTC memory use.

The development in the source code involves two different objectives:

1. The functionalities to carry out the testing.
2. The functionalities to customize the searches.

For the first objective, the main changes in the source code and data files are made in order to visualize the improvements proposed considering aspects such as time, CPU load, memory use and the quality of the results per search.

The system device characteristics can be monitored by the DDM tool. However, the time and the quality of the search can not be monitored with this tool. For this reason, a function which calculates time has been added to the source code. This functionality is called when there is a searching of two different ways:

-- when user makes a search by POI's: the function is called since the user selects a typology or group of POI's and ends when the selected POI is shown in the map.

-- when user makes a search for routing: the function is called when user selects the option "Navigate to a point" and finishes when the route is showed to the user in the map.

Once it has finished, a message in shown in the device screen, specifying the time that has been spent in the search, as can be seen in figure 20. This process gives us a clear idea if the changes made in the data files, are correct.

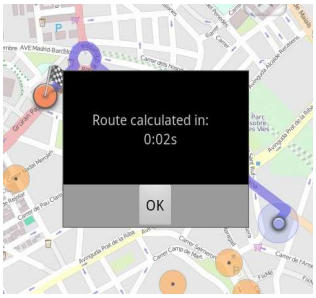

Fig.20. Capture of the dialog showing the time calculation of a route online search (in the testing section specified as Case 2).

In order to be able to know the quality of the routing searches, the route described in the mobile device is compared to the same search done in the ORS webpage [26], as shown in figure 21, using the same coordinates in both. Also both searches are performed in the same travelling display: pedestrian.

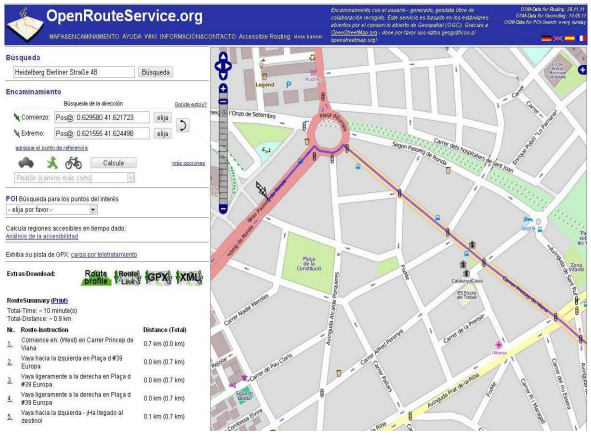

Fig.21. Capture of the OSR comparative of the search used in testing Case 2.

For the second objective, customizing and facilitating the search of the user, the data of some POI's has been optimized including the schedule time. OSMAnd does not offer any information about the schedule even though it offers to set that information. For this reason, a little functionality that considers this aspect has been included. This function checks the day of the week and the time in the device mobile and compares it to the data of the POI. As shown in figure 22, if the selected POI is closed in that day in that time, then shows the information to the user, and asks if the user wants to go on with the search.

Fig.22. Capture of the screen telling the POI selected, *la Llotja* is closed.

With this functionality, users won't waste time going to a POI that is closed. Moreover, this functionality improves the user satisfaction and helps in customizing the routes even though do not affect in the routing optimization and treatment of the data and it's not included in the testing.

E. Testing the Improvements in the Android Device

All the testing will be performed in the town of Lleida (Catalonia, Spain). The testing will be done for the following search cases:

- *Case 1*: Search of a particular POI and its location on the map.
- *Case 2*: Search from *my location* option to a specific address.
- *Case 3*: Search from *my location* option to a specific POI.

All three searches will be done with and without connectivity for each file: the *Europe_Spain.obf* file and the simplified and improved file, named *Lleida.obf*. . The online searches are performed with GPS and Wi-Fi enabled in order to not loose connectivity. The offline searches are done using the GPX files tracked before.

The address introduced to the system is *Gran Passeig de Ronda, Lleida* and the POI's selected are, in Case 1 *La Llotja*, the emblematic Congress Center of the town, and in Case 2, *Centre d'Art La Panera*, an art exhibition hall.

The difference between the OSR and the OSMAnd clone search is denoted by a numerical value. This value is the difference of the distance (in meters) of both searches, establishing the same coordinates. The value shows which has given the best route, being the best search the shortest one, and it is expressed like that:

--If *value=0*: both routes are equal.

--If *value<0*: the OSR search is shorter than the OSMAnd clone search.

--If value>0: the OSMAnd clone search is shorter than the OSR search.

Every case of search is performed 3 times for the values of CPU load, memory used and time spend on the search. The values shown in the final results are the average of the three results obtained.

VIII. RESULTS AND DISCUSSIONS

Following the testing and the implementation methodology, explained in the previous section, the results obtained are shown in table I below.

TABLE I
RESULTS

File	Case	Status	CPU	Memory	Time	Quality
Europe_Spain.obf	1	Online	93%	401MB	8s	0
	1	Offline	79%	316MB	5s	
	2	Online	98%	395MB	9s	-14
	2	Offline				
	3	Online	94%	412MB	15s	8
	3	Offline	81%	358MB	13s	
Lleida.odf	1	Online	91%	384MB	7s	0
	1	Offline	89%	315MB	4s	
	2	Online	96%	372MB	5s	-14
	2	Offline				
	3	Online	85%	425MB	12s	8
	3	Offline	79%	374MB	9s	

Table I shows the values monitored in the testing (CPU load, memory use, time per search and the quality of the route) for each case and for each data file considering the connectivity.

These results show the fact that OSMAnd clone is an application implemented thinking on mobile devices characteristics and limitations. Its adaptability to the device main resources can be seen in the figure: it never surpasses the ROM memory capability or collapses the CPU.

It's also shown that offline features are more prioritized than online features. This prioritizing can be seen, for example, in the fact that offline searches can not be performed if the user must type the address, because of the use of Nominatim online tool. For this reason, there are no values in offline searches for *Case 2* in the table.

Analyzing the results between the online and offline searches for every file, there is a significant decrease on time values as well as for the ROM memory use, between the searches. However, the CPU load is quite similar in the 3 cases, in which the lower value obtained is 79% for both files. For the offline searches, still there is little improvement in the CPU load, because even the search is performed in the device, the GPS and Wi-Fi connections, with the system processes and the exchange of data to maintain the connection, are not working.

Comparing the results obtained between the two stored files (*Europe_Spain.obf* and *Lleida.obf*), there is a significant decrease on the time values. CPU load and memory use are lower comparing them with the full Spain file because OSMAnd also uses the stored file information for the online searches, minimizing the exchange of information between the device and the server, directly sending the coordinates of a POI, instead of the query for the name of the POI.

The differences between the OSR and the OSMAnd clone routing searches are minimal: 14 m and 8m. This is due to the fact that the data of Lleida has not been indexed since 2007, as explained in section 6, so they have the same available data. Moreover, ORS gives the total distance value of the route in kilometers, rounding it off up, while OSMAnd clone shows the distance in meters (also in kilometers but does not round off). However, as seen in figures 20-21, showing the same search, OSR makes an optimal path, the shortest one, than OSMAnd clone. OSMAnd clone runs the route by the other side of the round.

To sum up, with the results obtained, it is shown that the weight of the data files and the quality of the data influence the

result. The less the weight, the less the time spent on searching and the less waste in memory and storage of the device.

IX. Conclusion

This work presents the state of the art of the main technologies, projects and services that are being developed for Android mobile devices. From the study of the main services and applications, their routing algorithms and their treatment of the data, this paper describes a methodology working on the data files to improve and customize the routing searches considering the main problems in routing in mobile devices:

1. The lack of connectivity in some areas.
2. The quality of the data involved in the searches.
3. The treatment and acquisition of the data to develop the searches.
4. The lack of services to improve the routing experience.

The solutions presented are the following:

1. The use of stored data files that include only the information of the area, reducing the ones available in the OSMAnd project site.
2. The optimization of the data files, both geographical information and also the characteristics of the POI's, editing and updating the data stored.
3. The treatment of the POI's information in the searches, in order to meet the needs of the users.

These solutions are implemented and tested on an OSMAnd clone application, a fully open OpenStreetMap-base navigation application for Android, specially optimized for offline searches. The testing area is the region of Lleida (Catalonia, Spain).

The implementation of the solutions 1 and 2 is made considering aspects such as time, CPU load, memory use and the quality of the results per search. For the third solution, to facilitate and customize the search of the user, the data of some POI's has been optimized including the schedule time and a function has been implemented that considers this aspect.

The main conclusions that can be drawn from the results analysis are:

- The quality of data helps on the searching in order to get better results.
- The weight of the stored files influences negatively in the searches.
- The functionality implemented considering the schedule, improves the user experience and helps on the user's choice, avoiding user's waste of time, and saving time and queries to the servers.

However, there are several problems that should be taken into account. First of all, OSM actual data is not indexed for the search. For example, the data of the area of Lleida has not been updated since 2007 for searching and routing. A study to implement a solution should be considered.

Second, the ontologies of the data in OSM are not well-defined. The research of better information architecture (IA) and its ontologies must be carried out.

Finally, with the improving of the IA, a study of the final user's needs should be considered in order to facilitate real thematic routes and information display on the application according to the user wishes.

Acknowledgment

C. Cuadrat would like to thank Alejandro Reche Pérez for its help and support in the technical and programming issues.

Finally, C.Cuadrat would also express her gratitude to all OSM project and OSMAnd project developers and contributors, and the director of this project, A. Navarro, without their work and advice this paper would not have been possible.

References

[1] J. Gutiérrez Puebla and M.Gould, SIG:Sistemas de Información Geográfica. Madrid (Spain): Editorial Síntesis, S.A., 1994, ch. 1.

[2] L. Descamps-Vila, J. Casas, J. Conesa and A. Pérez-Navarro,"Cómo introducir semántica en las aplicaciones SIG móviles: expectativas, teoría y realidad" presented at the V Jornades SIG Lliure, Girona (Spain), March 24, 2011. Available: http://dugi-doc.udg.edu/bitstream/10256/3380/1/art6.pdf

[3] Android Market Site. [Last search: 27/11/2011]. Available: https://market.android.com/

[4] L. Descamps-Vila, J. Casas, J. Conesa and A. Pérez-Navarro, "Hacia la mejora de la creación de rutas turísticas a partir de información semántica", presented at the V Jornades SIG Lliure, Girona (Spain), March 24, 2011. Available: http://dugi-doc.udg.edu/bitstream/10256/3384/1/art13.pdf

[5] GoogleMaps Site. [Last consulted: 24/10/2011]. Available: http://maps.google.com/

[6] OpenStreetMap Site. [Last search: 20/10/2011]. Available: http://www.openstreetmap.org/

[7] Wikiloc Site. [Last search: 24/10/2011]. Available: http://wikiloc.com/wikiloc/home.do

[8] Comisión del Mercado de las Telecomunicaciones (CMT), Informe Anual 2010 - Infraestructuras, Barcelona (Spain), 2010. Available: http://informeanual.cmt.es/docs/1.2%20INFORME%20SECTOR%20-%20INFRAESTRUCTURAS.pdf

[9] W. Paireekreng and K.W. Wong, "Intelligent Mobile User Profile Classification for Content Personalisation" in IEEE WKDD '10 Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Washington DC (USA), Jan. 9-10, 2010, pp241-244.

[10] C. Davidsson and S. Moritz, "Utilizing Implicit Feedback and Context to Recommend Mobile Applications from First Use", in Proceedings of the 2011 ACM Workshop on Context-awareness in Retrieval and Recommendation (CaRR), New York (USA), February , 2011, pp. 19-22.

[11] Allied Business Intelligence, Inc., Android Overtakes Apple with 44% Worldwide Share of Mobile App Downloads. [Last search: 26/10/2011]. Available: http://www.abiresearch.com/press/3799-Android+Overtakes+Apple+with+44%25+Worldwide+Share+of+Mobile+App+Downloads

[12] Gartner Newsroom, Gartner Says Sales of Mobile Devices in Second Quarter of 2011 Grew 16.5 Percent Year-on-Year; Smartphone Sales Grew 74 Percent. [Last search: 26/10/2011]. Available: http://www.gartner.com/it/page.jsp?id=1764714

[13] Google Code, Google Projects for Android: Google APIs – Obtaining a Maps API Key. [Last search: 20/10/2011]. Available: http://code.google.com/android/add-ons/google-apis/mapkey.html

[14] Google Code, Google Projects for Android: OSM API. [Last search: 20/10/2011]. Available: http://code.google.com/p/modosmapi/

[15] Google Labs, Google Ride Finder. [Last search: 20/10/2011]. Available: http://labs.google.com/ridefinder

[16] Google Labs, Google Transit site. [Last search: 20/10/2011]. Available: http://www.google.com/intl/en/landing/transit/#mdy
[17] I. Lidó Monzón and Dr. S. Machado Sánchez, "Aplicación Android de movilidad para invidentes", M.S.c, IT Department, Universitat Politècnica de Catalunya, Barcelona, Spain, April 30, 2011.
[18] R. Priedhorsky and L. Terveen, "The computational geowiki: what, why, and how", in Proceedings of the 2008 ACM conference on Computer supported cooperative work, San Diego (USA), Nov. 08-12, 2008, pp. 267-276
[19] OpenStreetMap (OSM), Official Wiki of OSM Project. [Last search: 20/10/2011]. Available: http://wiki.openstreetmap.org/wiki/
[20] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps", Pervasive Computing, IEEE Computer Society, Vol. 7, no. 4, pp. 12-18, Oct.-Dec. 2008
[21] R. Ganti, N. Pham, H. Ahmadi, S. Nangia and T. Abdelzaher, "GreenGPS: A Participatory Sensing Fuel-Efficient Maps Application", Annual International conference on Mobile Systems, Applications and Services (MobiSys), San Francisco (USA), June 2010, pp. 151-164. Available: http://domino.research.ibm.com/comm/research_teams.nsf/pages/mes.pubs.html/$FILE/mobisys10.pdf
[22] B. Ciepluch, P. Mooney, R. Jacob and A.C. Winstanley, "PhD Showcase: Using OpenStreetMap to deliver location-based environmental information in Ireland", SIGSPATIAL Special, v.1 n.3, Nov. 2009, pp. 17-22
[23] P. Neis, P. Singler and A. Zipf, "Collaborative mapping and Emergency Routing for Disaster Logistics – Case studies from the Haiti earthquake and the UN portal for Afrika", In: Geospatial Crossroads @ GI_Forum '10. Proceedings of the Geoinformatics Forum Salzburg, 2010. Available: http://koenigstuhl.geog.uni-heidelberg.de/publications/2010/Neis/un-osm-emergency-routing.gi-forum2010.full.pdf
[24] Portlach Edition Tool. [Last search: 08/01/2012]. Available: http://www.openstreetmap.org/edit
[25] M. Codescu, G. Horsinka, O. Kultz, T. Mossakowski and R. Rau, "Osmonto - an ontology of OpenStreetMap tags". In State of the map Europe (SOTM-EU) 2011, 2011. Available: http://www.informatik.uni-bremen.de/~okutz/osmonto.pdf
[26] OpenRouteService website. [Last search: 13/01/2012]. Available: http://openrouteservice.org
[27] S. Schmitz, A. Zipf and P. Neis, "New Applications based on collaborative geodata – the case of Routing", in XXVIII INCA International Congress on Collaborative Mapping and SpaceTechnology, Gandhinagar, Gujarat, India. Available: http://koenigstuhl.geog.uni-heidelberg.de/publications/bonn/conference/cmap2008.cartography-bonn.subm.pdf
[28] Navit project website. [Last search: 02/11/2011]. Available: http://www.navit-project.org/
[29] GpsMid project website. [Last search: 30/12/2011]. Available: http://gpsmid.sourceforge.net/
[30] OSMAnd project website. [Last search: 08/01/2012]. Available: http://osmand.net/
[31] VGPS project website. [Last search: 02/11/2011]. Available: http://www.digitalmobilemap.com/vgps-map-generator
[32] Wikipedia's Graph Theory page. [Last search: 24/10/2011]. Available: http://en.wikipedia.org/wiki/Graph_theory
[33] J. Gimbert, R. Moreno, J.M. Ribó and M. Valls, Apropament a la teoria de grafs i als seus algorismes.Lleida (Spain): Edicions de la Universitat de Lleida, 1998, ch. 1 - ch.2 Annex A.
[34] Planet OSM repository website. [Last search: 11/01/2012]. Available: http://planet.openstreetmap.org/
[35] Java OpenStreetMap Editor (JOSM) website. [Last search: 11/01/2012]. Available: http://josm.openstreetmap.de/
[36] Garmin Corporation website. [Last search: 11/01/2012]. Available: http://www.garmin.com/
[37] Nominatim project web page. [Last search: 14/01/2012]. Available: http://wiki.openstreetmap.org/wiki/Nominatim
[38] HTC Desire main webpage. [Last search: 14/01/2012]. Available: http://www.htc-desire.es/
[39] Android SDK site. [Last search: 16/01/2012]. Available: http://developer.android.com/sdk/index.html
[40] Eclipse Indigo site. [Last search: 16/01/2012]. Available: http://www.eclipse.org/
[41] Ubuntu site. [Last search: 16/01/2012]. Available: http://www.ubuntu.com/
[42] Git Control System webpage. [Last search: 16/01/2012]. Available: http://git-scm.com/
[43] SQLite Database webpage. [Last search: 18/01/2012]. Available: http://www.sqlite.org/
[44] OSMAnd Wiki project site. [Last search: 18/01/2012]. Available: http://code.google.com/p/osmand/wiki/HowToArticles
[45] Android developers: How DDMS Interacts with a Debugger. [Last search: 18/01/2012]. Available: http://developer.android.com/guide/developing/debugging/ddms.html