UBILocus - Framework for location aware application development

Nuno Sousa
nuno.ribeiro.sousa@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—Nowadays the number of location-aware applications and services is increasing. The development of these applications relies on the Operating System provided APIs or in libraries provided by specific positioning providers. The use of multiple positioning providers in the same application (for instance GPS and Beacons based indoor positioning) requires the use and integration at the application level of various (and sometimes incompatible) functions and data types. Furthermore, the use of complementary services requires the integration with respect to the coordinate system to use.**

In this paper, is proposed a middleware, called UBILocus, that allows the integration and abstraction of multiple positioning providers. Is also defined the library interface, the application data types, describe its implementation and on usage application. Besides being a positioning provider (using multiple technologies), UBILocus also integrates a route generation module usable by the mobile application.

UBILocus was implemented using the Xamarin Platform, allows the use of two positioning providers (GPS and Aruba Beacons[1]), provides two routing subsystems and seamlessly presents in the application private maps. The implemented UBILocus was tested in an Android application.

I. Introduction

The development of location-aware mobile services requires a myriad of services to calculate the location of the device, provide routes, present maps and provide added value to the user[2]. These usual services are presented in Figure 1.

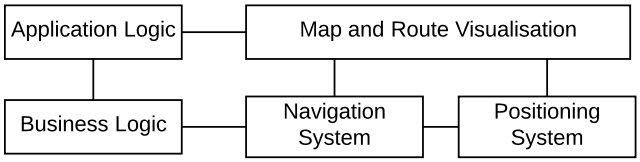

Fig. 1. Services in location-aware application

APIs and libraries to access these services (map visualization, navigation and positioning) are widely available and allow the development of complex systems and applications to fulfil most stakeholders requirements. These libraries are provided by the Operating System (for instance positioning services based on GPS, Wi-Fi and network cells) or by other companies which are developing additional indoor positioning protocols and services.

The development of location-awareness applications and services are crucial in areas such as tourism[3], smart cities[4] or even retail[5], where accurate position and efficient routing should be provided to the users and services. For instance, in these areas the integration of indoor/outdoor positioning is crucial, but until now the development and integration of various location related services incur problems:

- the application should integrate the various positioning providers APIs and data types, which could be impossible if the number of positioning providers grows;
- the application should correctly select the positioning service to use at each time;
- in the case of navigation, the application should be compatible with different coordinates in order to process and correctly display the route;
- the coordinates of the points-of-interest (POIs) should also be compatible with the positioning providers.

The problem of using multiple location services can also be seen in large campus navigation where a user can walk outdoor, using the GPS for location, or walk indoor where is necessary to use a more precise and adequate position system [6].

For the mitigation of the presented problems, is proposed a middleware that allows the integration of multiple positioning services, routing services and mapping services that could be accessed by a common API.

UBILocus is a framework that is extensible, allowing the integration of multiple positioning providers, from different vendors that could work in indoor or outdoor environments. It is also a flexible framework, allowing the simultaneous use of those position systems (hybrid solution). The details of those systems (running locally on the mobile device or requiring a remote server) will be hidden from the application developer. Furthermore, the developer will not be aware of the transition of the environment (indoor/outdoor) since the selection of the suitable provider will be handled by UBILocus.

Another functionality of UBILocus is to integrate multiple navigation systems. The access to this component will also be hidden from the developer by means of an API that will allow different routing requests.

This integration will require the definition of a high level location object (usable in the application and compatible with the various components) and the definition of a distributed architecture:

- Library linked to the mobile application;
- Service running on the mobile device and responsible for forwarding location or routing requests;
- Remote server responsible for the integration of the various routing subsystems.

The remainder of the paper is organized as follows. In Section II, is explained some key concepts regarding navigation systems, positioning systems and mapping services. In Section III is explained the proposed framework. In Section IV is explained the implementation details. Finally, are stated the conclusions and future work in Section V.

II. Related Work

In order to unify, simplify and abstract the development of location-aware applications that require ubiquitous services, multiple frameworks proposals are available.

A. Existing frameworks

Springer [7] presents a framework that integrates multiple positioning systems providing a common API for accessing the positioning information. This framework also provides a seamless handover between different technologies (indoor and outdoor) without the developer awareness. Despite this, this framework do not integrate the mapping and navigation complementary systems which does not corresponds to the requirenments of the development of location-aware applications according to Dhar and Varshney[8].

In order to provide ubiquitous services Ranganathan et al.[9] presents a framework that provides ubiquitious location (integrates multiple positioning systems, which could be indoor and outdoor) and ubiquitious mapping. Despite these features, this framework proposal doesn't consider a ubiquitous navigation system which is essential in areas such retail[10][5][11].

Aware of the services requerinmets (Positioning, Mapping and Navigation) of location-aware applications development according to Dhar and Varshney[8], Tóth[12] propose a framework for an indoor positioning, mapping and navigation system. Although this system can be easily extended with different navigation, mapping and positioning systems, it lacks one of the main features of location-aware applications that is the capability of working in multiple environments providing a real ubiquitous positioning, mapping and navigation experience.

The creation of a framework that abstracts the type of positioning and navigation systems, as well as mapping services, involves different research topics. In the present section, is explained the challenges of each of this topics.

B. Positioning Systems

A Positioning system is a system that retrieves the position of an object, device or person in a given place. In this work, the positioning systems are divided with respect to the type of environment in which will be used:

- **Outdoor Positioning Systems** In Outdoor Positioning Systems are included Systems like the Global Positioning System (GPS)[13], the GLONASS[13] and Galileo [13]. These systems are composed of a constellation of satellites and a receiver. The receiver is able to calculate its own location by measuring the time-of-flight of the signals received from, at least, 3 satellites. After this measurement, the device, based on a trilateration algorithm, calculate its position. To the device operate properly, the satellites need to be in line-of-sight. In urban environment, these systems could obtain an error of 28 meters along a street 95% of the time[14]. This error is due to the line-of-sight requisite not been guaranteed in urban areas.
- **Indoor Positioning Systems** Nowadays, the standard Global Positioning System (GPS) provides a *de facto* standard Outdoor Positioning System, whereas doesn't exist a consensual accepted equivalent for an Indoor Positioning System (IPS). In the past few years all kinds of IPSs[6] appeared based on, for example, Wi-Fi[15], RFID[16], Zigbee, Bluetooth Low Energy(BLE), GSM, Light encoding[17]. More recently are appearing technologies that use a combination of several smartphone sensors, such as, Wi-Fi and Bluetooth[18]. All these indoor positioning technologies, normally, are based on proprietary infrastructure and makes use of custom APIs making the application development harder.

Due to the lack of a Positioning System that covers indoor and outdoor environment simultaneously, UBILocus allow the integration of multiple positioning systems at low cost to developers by providing a common API to all positioning systems. The selection of the most appropriate subsystem is out of the scope of this paper.

C. Coordinate Systems

A topic that is tightly coupled with the Positioning System is the Coordinate System. A Coordinate System is essential to uniquely determine the position of a device. This position can be of three different types[19]:

- **Absolute position** represents a position on the surface of the earth that is characterised by coordinates whose origin is the centre of the earth. The standard coordinate system in use is the World Geodetic System established in 1984 and also known as WGS84;
- **Relative position** represents a position that is relative to a referential. For example, in the case of the IPSs, normally the user position is given as a position on a given map, *i.e*, this position is relative to the centre of that map. Normally the map is provided as an image thus the coordinate system is centred in the top left of the image and the units of the coordinate system are pixel. Note that a relative position is always represented by a coordinate but this coordinate is relative to a fixed origin (center of the map);
- **Symbolic position** represents a location characterised by a name and an identifier. At the application level, this location doesn't need an coordinate associated.

Note that in Outdoor Positioning Systems the most common type of position is Absolute position whereas, in IPSs, the position could be represented in many different ways. For example, in some IPSs with room level precision, *i.e*, only is identified the room where the user is, is common to use Symbolic position that identifies unequivocally the current room. In the Section III-D will be explained how this three

types of position can be combined to provide an abstract positioning system.

D. Mapping services

Mapping Service is defined as a service that provides information about the world. This information should contain a representation of the world or more commonly known as a map. This map helps to provide a contextualized experience providing landmarks (features), paths and a visual representation to the user. In the research, multiple mapping services were identified such as Meridian (used by Aruba Beacons), Google Maps and OpenStreetMaps.

In Table I are summarized some properties on the selected mapping services.

Note that, the OpenStreetMaps is an open source solution that allows to have private data. In the case of a navigation application in a museum, this feature allows to maintain the map information only available to the users that access the museum.

E. Navigation Systems

A navigation system should provide a route between two or more given locations. Nowadays navigating outside is a relatively common task, where exists services like Google Maps (with Google Directions API) and Bing Maps. Navigating inside, where exits few commercial services, such as, Meridian and indoor.rs, is a relatively new task, although recent observations said that 80% - 90% of the people daily life is spent inside[20]. This fact can be explained by some of the limitations of the IPSs and because, only in the last years, we assist to the generalisation of the use of the smartphones which is used by the majority of the IPSs[21].

During the research, were identified multiple routing systems but few of them could, out of the box, create a route between a set of locations where some locations could be indoors and others outdoors. The main problem in the identified routing systems is that the provided APIs are not prepared for indoor routing because these systems only works with coordinates that are 2D (a latitude and a longitude). In the indoor case, a coordinate composed only by a latitude and a longitude could identify multiple places because at the same location could exist multiple floors. In Table II are summarised some properties on the selected navigation systems.

Note that, the GraphHopper routing system uses the OpenStreetMap data. As GraphHoper is Open Source and easily extensible, is possible to make modifications to the routing engine in order to make it compatible with indoor environment.

III. UBILocus

From the study of existing platforms, can be concluded that the various systems provide most of the requirements for the application developers, but if it is necessary to integrate various technologies several problems arise, with respect to the development and change of the application, or even with respect to the compatibility of data formats used. The objective of UBILocus is to provide to the developers a framework that hides the implementation and data details of the positioning, routing and mapping services (presented in Figure 1), offering ease of use, interoperability and extensibility.

UBILocus will be called by the application and business logic in order to provide added value to the users/service provider.

A. Requirements

In the context of a mobile application, the Positioning component of UBILocus will allow the application, and any existing backend, to calculate the device location. This location can be used both by the Application or by the Business Logic. The Positioning component should hide the differences between indoor and outdoor environment, as well as, allow the use of multiple and complementary positioning techniques (hybrid positioning). Multiple positioning systems can even be used simultaneously depending on the environment of use.

The main objective of Navigation component is to provide a route between a source location A and a destination location B. This navigation system component should allow the definition of constraints and requirements when calculating the optimal route. For instance, a route should contain intermediate locations, also known as waypoints. Besides allowing the integration of various routing algorithms/subsystems, this component should also be capable of handling the location provided by the various positioning services.

The mapping widgets should also be configurable (to allow changes in implementations) and compatible with the various data types and formats used in the application.

The overall requirements can be summarized as:

- Single API;
- Positioning, routing and mapping systems should be decoupled to allow easy exchange to better reflect the stakeholders requirements (Open-Source vs Closed-Source or Paid vs Free);
- integration and simultaneous use of: multiple positioning subsystems and multiple routing systems;
- use of private/supplied maps in the mapping subsystem;
- interoperability between the various positioning, routing and mapping subsystems.

B. Architecture

Figure 2 presents the architecture of UBILocus. The various components fulfil the various functionalities and requirements presented. Although not yet referred, the UBILocus should rely on a server to implement several of its functionalities depending on the adopted technologies (positioning and routing) and to host the Business logic of the application (features, logs and routing policies).

Besides the library, that exports methods, UBILocus is composed by a runtime on the mobile phone and a server component running co-located with the databases (features and maps) and with the Business logic.

Split between the mobile phone and the server lives the visualization, positioning and navigation components.

Mapping Services	Supported Environments	Open Source	Public and Documented API	Include Navigation System	World-Wide Coverage	Map Information
Meridian	Indoor and Outdoor	No	No	Yes	No	Private, proprietary data types
OpenStreetMaps	Indoor and Outdoor	Yes	Yes	No	Yes	Public or private, Open Source and easily extended data types
Google Maps	Indoor and Outdoor	No	Yes	Yes	Yes	Public, proprietary data types

TABLE I
COMPARISON BETWEEN DIFFERENT MAPPING SERVICES

Navigation Systems	Supported Environments	Open Source	Public and Documented API	World-Wide Coverage
Meridian Routing	Indoor and Outdoor	No	No	No
GraphHopper	Outdoor	Yes	Yes	Yes
Google Directions	Indoor and Outdoor	No	No for indoor case	Yes

TABLE II
COMPARISON BETWEEN DIFFERENT ROUTING SYSTEMS

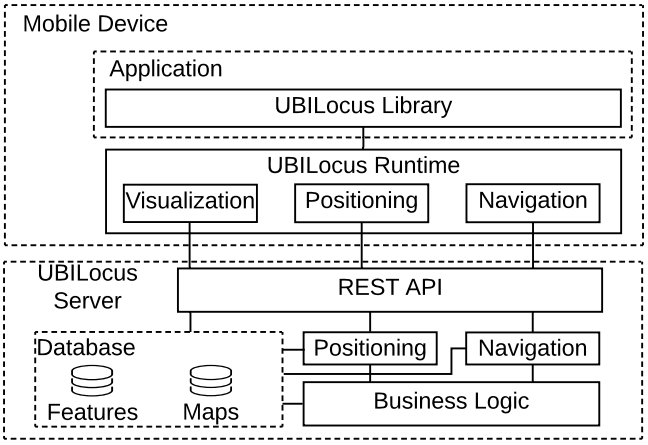

Fig. 2. Architecture of UBILocus.

Although the **Visualisation** component runs on the mobile device to present the user with maps, routes and features, it needs to contact the UBILocus server to fetch the corresponding maps. This maps can be hosted in a private or public service. The limits and the corresponding coordinate system of each map allows the seamless transition between maps as the user moves.

The **Positioning** component will allow the interaction with the installed positioning subsystems (for instance GPS and Beacons). Since a common API is provided (section III-C), this component is responsible for the selection of the more appropriate positioning algorithm/subsystem and for the transformation of each coordinate system to the one defined in UBILocus data types (section III-D). Although GPS works standalone on mobile devices, other more complex positioning systems (for instance based on beacons) can rely on a server component. To fulfil these cases, the UBILocus runtime will also be able to interact with the server to define and calculate the mobile device location.

The UBILocus **Navigation** component is also used to hide and abstract all the complex routing algorithms. This computation is offloaded to the server due to power constrains and necessary information (beacons, available routes, accessibility), the mobile runtime will be mostly responsible for the communication to the server and processing on the responses. Since the routing system should handle indoor and outdoor environments simultaneously, *i.e.*, the selected routing system should handle the different types of locations as described in III-D.

C. Library

The library will hide all the details of the various subsystems providing at the same time, the functionalities required by most of the location-aware applications. The defined functions are presented next.

Positioning System API Taking into consideration the current programming paradigms, the location retrieving will be based on events and callbacks. The API has two methods to configure the location updates: `StartPosListening` (this function receives a parameter corresponding to the minimum time/space interval between location updates), and `StopPosListening`. Each time the location changes (w.r.t. the configuration) an event is launched. This event contains the current location information.

Routing System API The objective of the Routing component is to calculate a route that connects a set of locations. The mobile application does not need to know the algorithm or subsystem and only needs to call the `RequestRoute`

method. This method receives as input a list of locations, and a JSON parameter that defines the constraints (for instance, type of route or information about impaired mobility). The function returns a route that the Business logic best defines.

Map and Route Visualization API Since the application map should display the current user location, routes and features. The API is as follow: `CenterInLocation` (Centers the view of the map in a given location), `DrawFeature` (Receives as parameter a location and additional information and plot the feature in the map), `CleanFeature` (Removes one feature from the map), `DrawRoute` (Receives as parameter a Route as returned by `RequestRoute` and plots it in the map), and `CleanRoute` (Removes the current route).

D. Data types

In order to propose a framework and a API that are extensible and allow the interoperability of various subsystems, it is necessary that the used data types follow a common and rich format. The proposed data types for the APIs are as follow and known by each of the UBILocus components. These data types are used to provide a seamless data transfer between components providing a loosely coupled and extensible system.

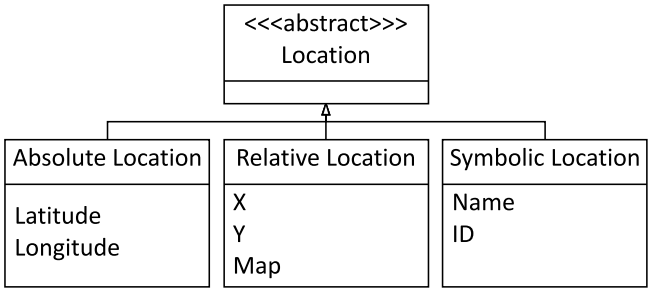

Fig. 3. Location data types

A **Location** is a generic data type that unequivocally identifies a point in space. Since various subsystems use different location representations, is necessary to define a generic Location type that can be extended to various representations. Figure 3 presents the various sub-classes of locations supported by UBILocus.

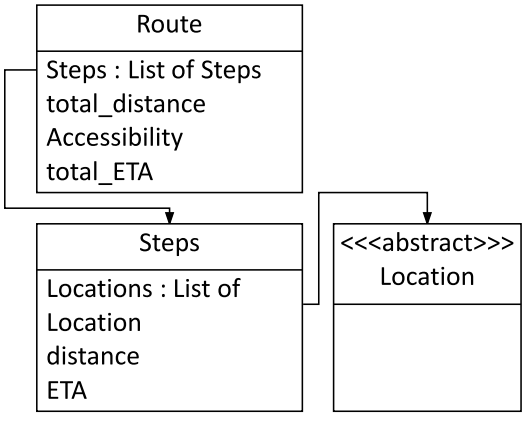

Fig. 4. Route data type

Absolute and relative Locations are similar in data but differ in use, since a Relative Location is associated with a Map (for instance building, or floor). Examples of symbolic Locations are Rooms (with textual identifies) or generic facilities (such as WCs). For the application, these generic facilities do not need a coordinate, but internally will have a actual physical place. Locations represented in different types (e.g. absolute and symbolic) will be matched in the route graphs and displayed in different maps.

Routes (Figure 4) represent the list of Locations that allows a user to go to a set of certain Locations. The way to calculate these routes is hidden from the mobile application developer and the only information presented are the actual points (initial, final and intermediary), the estimated distance and time of arrival.

A **Feature** represents a Point of Interest relevant to the application and is composed of a location, an identifier and additional data.

Since Features, Routes and Locations can be local to an institution, building or private area, **Maps** should also be considered as a UBILocus data type. They will be used in the mobile application to visualize data but will also provide context to the relative Locations. Maps information includes: a Name, a Floor Number, a graphical representation (image) and a unique identifier.

IV. Implementation

The first version of the UBILocus library is targeted at the Xamarin Platform. It consists of a set of C# files and binary libraries. This implementation allows the execution of the applications compiled to various mobile platforms (Android and IOS). The runtime is included in the library and linked to the application.

The UBILocus server implements a set of REST APIs (for the various components) and was implemented in Python. This server aggregates the actual implementation/subsystem for the maps storage, routing definition and positioning calculations, as described next.

Positioning System The current version of the positioning component implements two distinct positioning algorithms. This component allows the access to the local GPS service, but also provides access to the indoor Aruba Beacons positioning service. Using these complementary technologies is possible to provide the position of the device indoors and outdoors.

In order to accommodate these two subsystems the positioning component is implemented as described in Figure 5.

This component has a wrapper that receives requests from the application and redirects them to the suitable subsystem. Since these two systems use different coordinate systems, the positioning component is responsible for the needed standardization (Coordinate Standardization in figure 5).

Another important component is the Positioning System Selector, which is responsible to automatically select the Positioning System that best represent the position of the user.

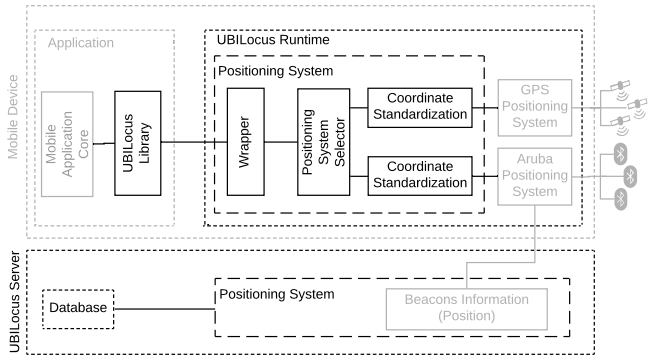

Fig. 5. Positioning System.

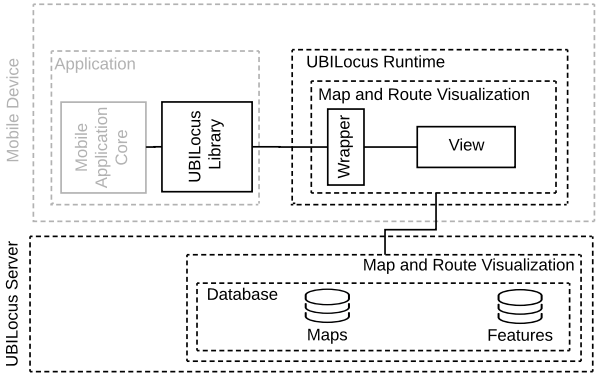

Fig. 7. Map and Route Visualisation.

The implementation of such component/algorithm is *per se* a topic of research and out of the scope of this work. Therefore, the current implementation used a simple selection algorithm: if a valid GPS position is calculated with a accuracy of 20 meters or greater, this position is used by the positioning system component. Otherwise, if a valid position is returned by the Aruba Beacons IPS, this position is used.

Routing System The Routing System runs completely on the server (as presented in figure 6), the mobile runtime only forwards requests to the server.

The current implementation allows the use of the Meridian[22] or GraphHopper[23] route generator engines, and provide a placeholder for the ordering of locations to visit and engine selection. The main function of the reordering system is to reorder, if necessary, the list of Locations based on a predefined strategy. After the reordering, the list of Locations is passed to the routing engine.

Visualization The implemented Visualisation System uses the open-source OpenLayers JavaScript library (openlayers.org) to display the suitable map: floor plant, features, route and the device position. This JavaScript library is wrapped around a web-view container as presented in Figure 7. This system also performs the download from the server of the suitable map to present in the application

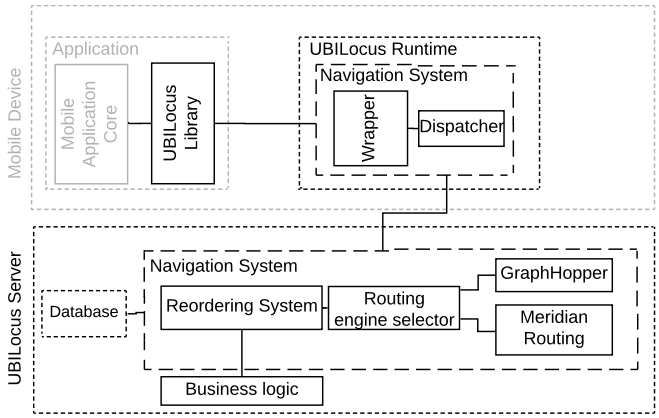

Fig. 6. Routing System

A. Validation

UBILocus was validated with the implementation of the Xamarin ILocate application, allowing users to navigate and get directions in an organization composed of several buildings using indoor (Aruba Beacons) and outdoor (GPS) positioning systems. Figures 8 and 9 present an example of a single route, in yellow, that is composed by a indoor path and outdoor path. Depending on the location UBILocus selected the suitable location mechanism and presented the correct map.

UBILocus also allowed the use of two routing systems, the GraphHopper routing system and the Meridian routing system. The GrapHhopper routing engine was extended with the development of a simple "Business Logic" which generate intermediate locations that the user should visit while going to its destination.

The proposed requirements (simultaneous positioning, routing, mapping subsystems and single API) were all accomplished with the design of UBILocus and demonstrated with its implementation and use in the development of mobile application.

1) User feedback: In order to further validate the UBILocus framework in production environment a public test was conducted using smartphones with Android OS(versions: 5.0.1, 7.0 and 7.1). In this test the users answered to a survey containing a set of questions regarding the the performance of each of the UBILocus systems (Positioning, Visualization and Navigation). According to the users feedback all the systems work as expected, except in the following topics,

- The accuracy and refresh rate of the indoor positioning system in use (Aruba Beacons) is not sufficient for indoor tasks in the conducted experiment. These problem is not a result of the use of the UBILocus, since the Position System of UBILocus function as expected (selecting correctly the indoor positioning system in the indoor environment).
- The transition between indoor and outdoor works flawlessly according to users (˜5 seconds) but the transition between outdoor and indoor experiences some delay (˜20 seconds). As explained in Section IV the implementation of the Positioning System selector is, *per se*, a topic of research in the bibliography. Therefore, further research

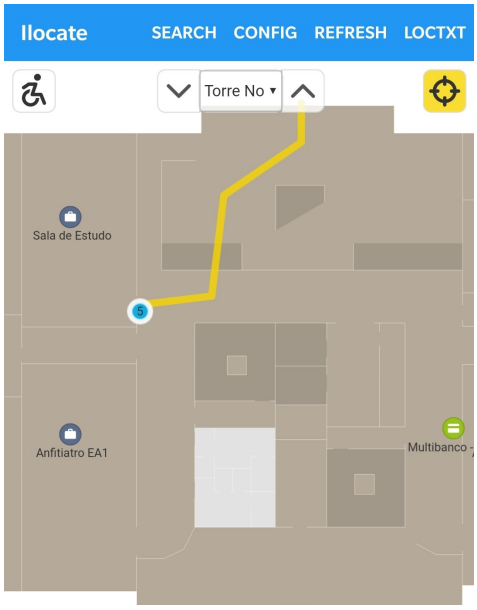

Fig. 8. Indoor View.

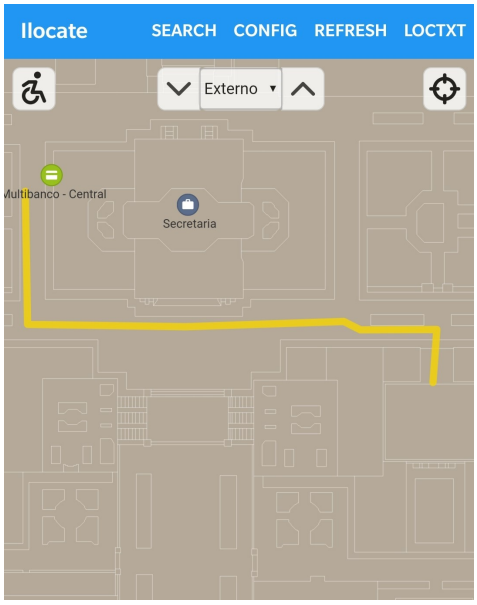

Fig. 9. Outdoor View.

and implementation should be done in this topic in order to accomplish better results in the transition timing.

2) Performance Test: A intensive performance test, regarding power consumption and resource consumption, was not done because the UBILocus middleware (Library and Runtime) is mainly composed by a set of callback methods (˜20 lines of code each) that internally don't have any type of loop or background threads running, *i.e,* the computational effort is low. Therefore the impact of UBILocus in the device performance and power consumption is residual and the main power consumption is mainly from the different Positioning Systems (in this case Aruba Beacons and GPS positioning services) and by the Xamarin Forms services.

V. Conclusions

In this paper, was proposed a middleware, called UBILocus, that allows an easy way to develop location-aware mobile applications that work in indoor and outdoor environments. A single API is provided for accessing the various positioning services. Moreover, UBILocus allows the simultaneous use of multiple positioning subsystems, providing the developer/user a transparent and seamless transition between different environments.

UBILocus also provides complementary location-aware mobile services such as route generator and map service. Both services could be easily exchanged in order to use the underlying technologies that best fit the requirements and the applicable business logic. All the features provided by UBILocus are hidden and abstracted in order to ease the development.

Although tested on Xamarin/Android the UBILocus system (composed by an API/Library, Runtime and Server), is possible to be used by different programming languages at low cost to the developers. This is possible due to layered architecture, isolation and abstraction, only requiring the porting of a new UBILocus Library, suitable for language/environment.

Also, UBILocus can be extended by adding custom components (that implement additional services), due to a well defined core of data types, and rich API. This extensibility feature was tested with the integration of multiple location/routing services and the demonstrative the business logic component.

As future work, further research has to be done on the topic of the reasoning/selection of the best positioning systems.

Acknowledgements: This work was supported by national funds through Funda-ção para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 and by ITEN Solutions – Sistemas de Informação, S.A by providing the Aruba Beacons.

References

[1] Aruba Beacons Validated Reference Design. URL https://community.arubanetworks.com/aruba/attachments/aruba/Aruba-VRDs/69/2/Aruba%20Beacons%20Validated%20Reference%20Guide.pdf. Accessed on 2017-10-09.

[2] E. Kaasinen. User needs for location-aware mobile services. *Personal and ubiquitous computing*, 7(1):70–79, 2003, Springer-Verlag.

[3] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a context-aware electronic tourist guide: Some issues and experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2000.

[4] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu, and L. T. Yang. Trace analysis and mining for smart cities: issues, methods, and applications. *IEEE Communications Magazine*, 51(6):120–126, 2013, IEEE.

[5] D. Black, N. J. Clemmensen, and M. B. Skov. Pervasive computing in the supermarket: Designing a context-aware shopping trolley. *Int. J. Mob. Hum. Comput. Interact.*, 2(3):31–43, 2010, IGI Global.

[6] Y. Gu, A. Lo, and I. Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE*

Communications surveys & tutorials, 11(1):13–32, 2009, IEEE.

[7] T. Springer. Mapbiquitous–an approach for integrated indoor/outdoor location-based services. In *Int. Conf. on Mobile Computing, Applications, and Services*. Springer, 2011.

[8] S. Dhar and U. Varshney. Challenges and business models for mobile location-based services and advertising. *Commun. ACM*, 54(5), 2011, ACM.

[9] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. D. Mickunas. Middlewhere: a middleware for location awareness in ubiquitous computing applications. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Springer-Verlag New York, Inc., 2004.

[10] V. Bourne. The rise of indoor positioning - a 2016 global research report on the indoor positioning market. URL https://www.indooratlas.com/wp-content/uploads/2016/09/A-2016-Global-Research-Report-On-The-Indoor-Positioning-Market.pdf. Accessed on 2017-10-09.

[11] Philips. Where are the discounts? carrefour's led supermarket lighting from philips will guide you, May 2015. URL https://www.philips.com/a-w/about/news/archive/standard/news/press/2015/20150521-Where-are-the-discounts-Carrefours-LED-supermarket-lighting-from-Philips-will-guide-you.html. Accessed on 2017-10-09.

[12] Z. Tóth. Ilona: indoor localization and navigation system. *Journal of Location Based Services*, 10(4), 2016, Taylor & Francis.

[13] B. Hofmann-Wellenhof and et al. *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer-Verlag Wien, 2007.

[14] M. Modsching, R. Kramer, and K. ten Hagen. Field trial on gps accuracy in a medium size city: The influence of built-up. In *3rd workshop on positioning, navigation and communication*. WPNC, 2006.

[15] N. Dinh-Van, F. Nashashibi, N. Thanh-Huong, and E. Castelli. Indoor intelligent vehicle localization using wifi received signal strength indicator. In *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. IEEE, 2017.

[16] S. Ting, S. K. Kwok, A. H. Tsang, and G. T. Ho. The study on using passive rfid tags for indoor positioning. *Intl. journal of eng. business management*, 3:8, 2011, SAGE Publications.

[17] T.-H. Do and M. Yoo. An in-depth survey of visible light communication based positioning systems. *Sensors*, 16 (5), 2016, Multidisciplinary Digital Publishing Institute.

[18] A. Baniukevic, C. S. Jensen, and H. Lu. Hybrid indoor positioning with wi-fi and bluetooth: Architecture and performance. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*. IEEE Computer Society, 2013.

[19] L. Reyero and G. Delisle. A pervasive indoor-outdoor positioning system. *Journal of Networks*, 3(8):70–83, 2008, Academy Publisher.

[20] K.-J. Li and J. Lee. Indoor spatial awareness initiative and standard for indoor spatial data. In *IROS 2010 Workshop on Standardization for Service Robot*, 2010.

[21] D. UK. There's no place like phone Consumer usage patterns in the era of peak smartphone, 2016. URL https://www.deloitte.co.uk/mobileuk/assets/pdf/Deloitte-Mobile-Consumer-2016-There-is-no-place-like-phone.pdf. Accessed on 2017-10-09.

[22] Meridian Platform. URL https://http://meridianapps.com/. Accessed on 2017-10-09.

[23] G. GmbH et al. Graphhopper: Fast and flexible route planning. URL https://github.com/graphhopper/graphhopper. Accessed on 2017-10-09.