DynamicPOI - Middleware for developing exhibition navigation applications

Diogo Salgueiro
diogo.m.neves.salgueiro@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—Museums and exhibition venues have been growing in the last decades, allowing a greater diversification both on their layouts and on the displayed artwork. On the other hand, the mobile application market growth allowed for these venues to provide better exhibition support and navigation applications to their visitors. However, each application is developed specifically for a single exhibition and the developer must implement a suitable exhibition structure and configure the necessary location systems.**

One contribution of this work is the definition of a generic schema that allows the definition of entire exhibition structures by establishing relationships between each of their points-of-interest, ranging from physical areas to displayed artwork items.

The other contribution is a middleware that improves the development process of exhibition navigation applications. This middleware aggregates several location systems transparently to the developer and automatically determines the most relevant POI of the exhibition, according to the visitor's location. Furthermore, an automatic services' management was also included to guarantee only the necessary location services are active during an entire visit.

The middleware was implemented in the Android environment by creating a library package and separate location service classes. Its operation was evaluated by testing its features during demonstration visits and the energy savings of an application using this middleware were also determined.

I. Introduction

Cultural related venues, *e.g.* museums and exhibitions, have been growing steadily in the last decades. This growth was accompanied by a greater diversification on the types of cultural heritage available to the general public. Furthermore, the technological evolution, particularly of mobile devices, allowed for a better exhibition experience [1].

Considering the growing concern with improving exhibition experience and visitor interaction, many museums have developed tour guides for their exhibitions [2]. These guides can have multiple technologies and purposes, ranging between simple audio guides to fully developed applications implementing several services (*e.g.* artwork information or augmented reality).

Regarding the diversity of exhibition applications, two main problems can be outlined:

- Each exhibition must have suitable representation for the venues' artwork, POIs and areas. The nonexistence of a common solution that supports multiple types of exhibitions implies that each venue must answer this requirement independently. As a result, when a programmer creates a guide or support application to each venue, it must obey and adapt to the representation guidelines previously defined. In addition, these types of representations are not scalable or do not have any type of hierarchy between their elements.
- In order to determine the user's position relative to an exhibition, one or more location services must be configured and accessed. The existence of different types of venues and areas inside those venues implies that a given location service that suits some areas may not be used accurately on others. Again, when developing solutions that need to determine the visitor's position inside a heterogeneous exhibition, the programmer must configure each location service individually and combine the information of multiple services. After that, the programmer must take the visitor's location and find the nearest POI by accessing the POI structures described above.

As a consequence to the problems outlined for an exhibition's helper application, identified on the previous section, there are two main objectives defined for this work.

The first objective is the definition of an generic representation for POI's structures. The main requirement for this representation is to be able to describe a large variety of POI structures, independently of their size, complexity and nature of the venues (indoor, outdoor or both). This implies that several location types must be considered as the most suitable one varies with each locale characteristics and location accuracy requirements. Other desired feature is not to be restricted to a single platform or data type or, in other words, to be able to easily be loaded into any relevant type of platform (*e.g.* web or mobile applications). This representation model must also be extensible with respected to the supported location types. At the same time, each POI structure that obeys to this representation's guidelines must be able to easily add, delete or modify its elements.

The second objective for this work is the development of a middleware that aims to simplify the development process of exhibition helper applications. The main purpose of this middleware is to determine the most relevant POIs identifier and make it accessible to any application that implements the middleware. This module must combine the information of an exhibition structure with the necessary location services' callbacks in order to determine the nearest POI at any given time. This modules' operations should be completely concealed not only from the application layer (that receives the nearest POI updates) but also from the layer that provides all the sensor information needed for the different location services.

II. Related work

A. Exhibitions

Nowadays, exhibitions are capable of displaying a large variety of the physical cultural heritage to the general audience. The scope of an exhibition may vary tremendously as it can span between the artwork of a single person to artifacts of an entire civilization [3]. Furthermore, exhibitions venues can have different dimensions, ranging from single rooms to geological parks.

On the other hand, the smartphone market growth allowed museums and other exhibition venues to personalize and diversify the visitors' experience inside their venues [4] [5].

Prior to the use of mobile phones inside exhibitions, visitors only had access to predefined routes (via maps or audio guides). Nowadays, several museums have their own applications, capable of displaying different types of tour information directly on the visitors' phone.

Utilising the smartphones' multimedia capabilities, exhibition visitors can now have on-demand access to videos and audio descriptions. One of the most recent technologies to be used to enhance the visitor's experience is Augmented Reality [6]. This technology allows live rendering of 3D objects in the smartphone's viewfinder while the user pans around an area.

Lastly, gamification and storytelling frameworks can be used to further improve the experience and make the user feel like he is an active part of the exhibition and not just a spectator [7].

B. Location identifiers

The basis for any location system is the type of location identifier it uses to describe its determined locations. Three categories of location identifiers are described below:

- **Geographical or absolute coordinates** - the location is described by a set of coordinate system whose origin point is the Earth's center. The most commonly used coordinate systems are ellipsoidal coordinate systems, which describe locations using two angles relative to two orthogonal reference planes: latitude, the angle relative to the Equator plane and longitude, the angle relative to the Greenwich meridian plane [8];
- **Relative or local coordinates** - This type of coordinates differ from the geographical ones by recurring to an referential other than the Earth's center. Therefore, these coordinates are accompanied by which referential they are relative to, *e.g.* a map or a predetermined physical point;
- **Symbolic identifiers** - a location is described by an identifier or human understandable name, *e.g.* locating an object simply as "inside a room", not specifying its position with discrete coordinates.

C. Location systems

Nowadays, mobile phones have several available location systems. A list of those systems is presented below:

1) Global Navigation Satellite Systems (GNSSs): Location systems that are comprised of a group (constellation) of satellites and ground control stations. The currently operating GNSSs are GPS, GLONASS, Galileo and the Beidou Navigation Satellite System (BDS). All of these systems share a common operating principle for locating client devices: By having several satellites visible to any client device, the distance between each satellite and a client device can be determined by measuring the signal travel times. The final position is calculated as the intersection of the "imaginary" spheres based on the distances to each satellite.

When clear line-of-sight is guaranteed between a group of satellites and the receivers (*e.g.* open areas) the GNSSs can produce very accurate positioning. Several GNSSs can be used simultaneously to further increase the accuracy [9]. On the other hand, on places with a large amount of tall buildings and indoors, the accuracy of these systems plummets and can even stop operating under these conditions.

2) Cellular networks: Complex mobile networks that provide wireless communication to devices inside its constituent cells. These networks' Base Stations allow for several positioning techniques of the connected Mobile Stations (*e.g.* mobile phones): cell identification, Angle of Arrival (AOA), and Time of Arrival (TOA)/Time Difference of Arrival(TDOA) [10]. A brief performance comparison between some of these techniques was presented by Adusei *et al.* [11]. These networks' positioning capabilities are almost exclusively used on outdoor environments and can only determine locations with a maximum accuracy of several meters. Thus, to improve location accuracy, cellular networks based location systems are usually paired with GNSSs.

3) WiFi: One of the most relevant technologies for both wireless communication and mobile devices location. One simple approach for determining device locations is through Access Points (AP) databases such as Wigle [12] . Generally at a fixed position, if a device is in range of a known AP, its location can be estimated. Other complex approaches include fingerprinting methods [13], which create signal maps for known locations during an "offline" phase and the location of a device estimated by matching algorithms on a "online" phase.

4) BLE: Protocol built from the original Bluetooth specifications, aimed at reducing energy consumption while maintaining a similar range of communication [14]. This reduced energy consumption allowed the creation of a set of devices called beacons. Mobile devices can estimate distances by evaluated the RSSI values of a set of beacons. However, due to complex signal propagation models (particularly indoors) and great interference from other devices on the same wave spectrum (*e.g.* WiFi), fingerprinting methods are often used to achieve better accuracy [15] [16].

5) NFC: Bidirectional wireless communication technology, that enables information exchange between devices when placed a few centimeters apart. NFC tags are small passive chips that can store data. If the position of a NFC tag is know *a priori*, the location of a mobile device can be determined after scanning this tag (given the inherent proximity of this protocol). Ozdenizci *et al.* [17] proposed an indoor navigation system using NFC tags.

6) QR codes: Bi-dimensional bar codes that can encode information in a machine-readable format. As a mean for encoding information, QR codes can be used to locate a user by two approaches:

- A QR can directly encode its position, *e.g.* geographical coordinates;
- A QR code can be an unique identifier of an item inside a venue. If the position of the item is known, one can assess the device's location when this identifier is scanned.

D. Points of Interest

The "Point of Interest Group" of the World Wide Web Consortium (W3C) proposed a set of guidelines to represent POIs [18]. Despite allowing a large set of properties for a POI, this data model has a limited set of features to describe POI relationships: only single level lists can be defined and the "Relationship" element only allows for spatial relation between two POIs.

The FIWARE open source initiative also provided a data model for POI representation [19]. This model, besides the core properties of a POI, allows for its physical representation to be a GEOJson location and/or an address description. Nonetheless, this data model lacks any kind of explicit relationship between POIs.

POI data models are the core of POI-based applications. On the context of Smart Cities, the CitySDK's Tourism API [20] is based on the W3C's POI data model and provides a unified access for tourism solutions and their stakeholders. Nitti *et al.* [21] proposed a IoT based architecture for tourism applications, taking advantage of IoT devices placed at POIs to provide relevant real-time information. Regarding POIs inside exhibitions, Hashemi *et al.* [22] studied a POI recommendation system, demonstrating how good POI structuring is fundamental on this type of applications.

Taking into consideration the three POI-based applications presented above, one of the outcomes is the ability to have both indoor and outdoor POIs but rarely combining these two types inside an application. Furthermore, these applications didn't present complex relationships between their POIs.

E. Programming challenges

In the first place, the exhibition's physical properties must be mapped to suitable data structures by the programmer. These properties can refer to the exhibition's areas, buildings and rooms but also where the different items are placed within them. This task becomes more tiresome if the developer needs to outline different kinds of exhibitions as a suitable format for a venue might not be appropriate for another.

The developer must also guarantee that all the necessary location systems are integrated into the application, allowing the location of both areas and their items. However, if a new location system is to be added, the developer has to openly integrate this new system which might possibly resulting of code changes on the entire application.

Lastly, the developer is also responsible for managing all the location systems in these applications, *i.e.* determine when they should be active and with which parameters should they operate.

III. Solution Design

A. Requirements

When considering an application for exhibition navigation, one can consider three different concerned user classes: the exhibition curator - who defines the exhibition layout; the developer - who programs the application; and the visitor - the final user of the application and visitor of the exhibition. This categorization is important as each of these user classes has a different role and, therefore, can have different requirements:

Exhibition curator

1) Define relationships between the constituent areas of an exhibition and their items;
2) Assign location identifiers to areas and their items, supporting a variety of location systems;
3) Modify a previously defined exhibition structure.

Developer

1) Develop applications that allow information access about items and areas;
2) Modify the exhibition structure without recompiling the application;
3) Easily access to exhibition's identifiers;
4) Integrate several location systems with future inclusions without rewriting the application;
5) Seamless management of the necessary location services during a visitation;
6) Have mechanisms to automatically display widgets.

Visitor

1) Automatic notifications for needed user actions;
2) Seamlessly turns on the necessary location services during a visitation.

In addition, two non functional requirements can be considered for the proposed solution:

- **Efficient battery consumption** - regarding the high energy consumption of several device's location systems, particularly GNSSs and cellular networks, it is desirable to minimize the up time of these services;
- **Minimize impact on application size** - The incorporation of the middleware must not result in a noticeable increase on the final application size.

B. Architecture

Considering the requirements presented in section III-A, the two main components of the proposed solution are a schema for the definition of POI structures and a middleware that simplifies the development process of exhibition navigation applications. Figure 1 presents a generic architecture of the solution and where these two components are integrated.

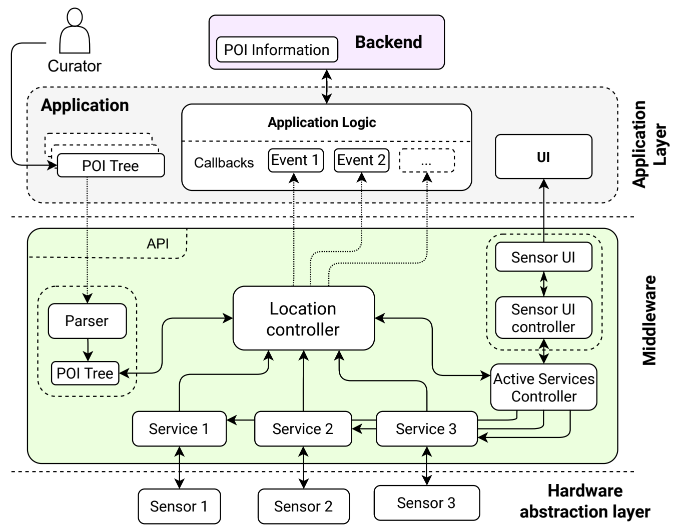


Fig. 1. Generic architecture for a middleware aimed at exhibition navigation applications

POI Tree The generic model for this POI Tree will be described on section III-C. A set of these structures must be loaded into the middleware by the application via a provided API.

Parser After a POI structure is loaded into the middleware, it needs to be parsed into runtime data in order to be easily accessible by the other middleware components. Straightforwardly, the Parser module has decoding mechanisms that obtain data objects from the externally loaded structures.

Location controller The main purpose of the Location controller is to determine the most relevant POI to the application's user (visitor). The controller analyzes the runtime POI structure and continuously maps the location responses from several location services into a position inside the exhibition.

Furthermore, upon changing to a new "most relevant" POI, an event is sent to the application and the Location controller also determines the necessary location services for the newly determined position inside the exhibition. This information is then passed to the Active Services controller.

Active services controller The Active Services controller manages the activity of all the location services configured on the middleware. Upon knowing from the Location controller which services are necessary to be active, it starts (or maintains) only those and stops the unnecessary ones. Contrary to the majority of the location services, some services need explicit user interactions to operate properly. In these cases, a separate controller must be notified in order to request the necessary interactions.

Sensor UI This component of the middleware is responsible for displaying widgets and prompt the user to perform specific actions when needed. The Sensor UI controller receives requests from the Active Services controller and invokes the correspondent UI resource that will be displayed to the user. The user interaction's result is sent back to the Sensor UI controller and will be used by the Location controller to assess necessary location updates.

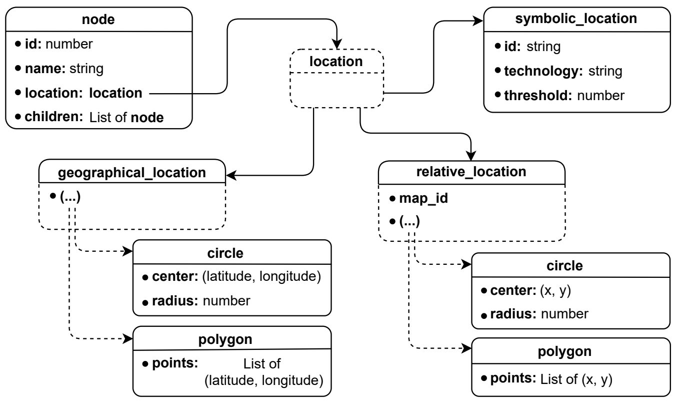


Fig. 2. Generic POI data model diagram

C. POI data model

The proposed data model for representing POI structures is illustrated on figure 2.

1) POI nodes: To achieve a hierarchical representation for a group of diverse POIs, a recursive approach was taken. The "node" entity, that represents a POI, is the fundamental element in any POI structure.

It has the "identifier" and "name" properties, as well as a property for describing its location (presented on the next subsection). As some POIs can have other POIs inside them, the "children" property allows the possibility for a "node" to have a list of inner "nodes" (with an identical structure). Therefore, using a set of recursive "nodes", a tree of POIs can be constructed, with a varying depth accordingly to the corresponding physical layout of the POI group.

2) Location: As stated above, each POI node has a "location" property which stores its positioning information. POIs can have different types of location identifiers, generally dependent on the chosen referential. In addition, an area of "significance" must be defined for each POI, *i.e.* an region i is relevant, visitor outside, not relevant. Considering this two guidelines, the "location" property can be one of three types: absolute, relative or symbolic and each one has properties that define an active region.

Geographical locations allow to locate anything on Earth, using geographical coordinates, such as latitude and longitude. In order to support geographical areas for POIs, the "geographical location" on the proposed data model can be defined by a circle, with a single point and radius, or a polygon, with an array of three or more points.

Relative locations have a similar principle to geographical location but the set of coordinates is relative to a specific referential, such as a building or room. Therefore, instead of using pairs of latitude and longitude, relative locations use 2D coordinates and an identifier for the chosen referential.

Symbolic locations differ from the previous two types by the absence of a unequivocally position on a referential. Instead, they identify an area, generally indoors such as rooms or hallways. This limitation does not hinder its usefulness, as some POIs might not require more precise location descriptions. The proposed data model for this location type has an ”identifier” property for the corresponding area and ”technology” and ”threshold” properties for possible technologies differentiation and a threshold condition.

D. API

In order to an application use the proposed library, a communication mechanism must be implemented between the two components. The library itself must have a publicly accessible API, composed of several methods calls that the application integrating the library could execute. On the other hand, the library will also have to provide callbacks or event responses in order to send information back to the application.

1) Method calls: The available API calls are presented below.

- `init` - This method initiates the runtime components of the library by providing a POI structure. This structure has to comply to the predefined set of rules in order to be initialize correctly;
- `getNearestPOI` - request the library for the nearest POI relative to the visitor;

2) Callbacks/events: The set of events from the library to the application using it are described below.

- `enteredPOI` - notifies the application that the visitor entered a new POI region;
- `exitedPOI` - notifies the application that the visitor had exited a previously entered POI region, *i.e.* deviated from its defined region;
- `nearestPOI` - returns the identification of the nearest POI to the application;

IV. Implementation

A. JSON schema

A JSON Schema was built using Schema guidelines [23] to implement the POI data model described on section III-C. The schema definition is based on a set of ”definitions” that assure the entire POI structure validation, from each of its elements properties to the multiple possible ”location” structures.

B. Android middleware implementation

The proposed solution was implemented on the Android environment, using the Kotlin programming language. Figure 3 provides an overview of the implemented solution.

The middleware component was implemented inside a single library package and by external location services. The orange colored components are runtime components that enable all the middleware's functionalities. The green colored components correspond to location services, each one implemented as a Service class (described on section IV-C).

The library's core class is named PoiManager. This class is responsible for aggregating several components, with varying levels of complexity but, most importantly, is the contact point between the middleware and the application layer. Two interfaces are defined to establish a bidirectional communication between the middleware and the application.

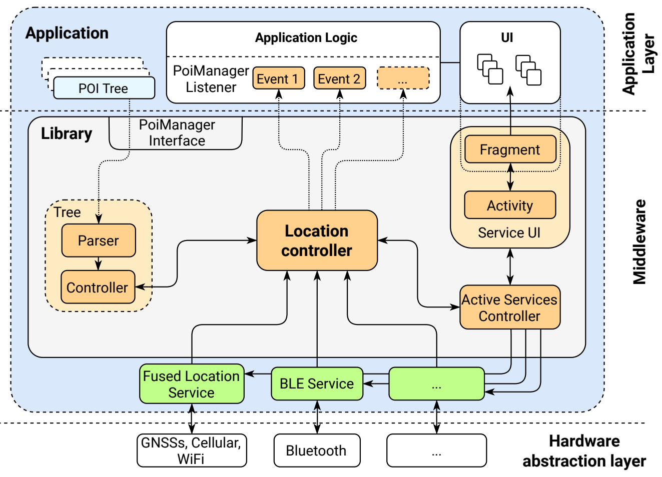


Fig. 3. Diagram of the implemented Android middleware

PoiManagerInterface The purpose for this interface is to define which are the publicly accessible methods for the middleware and their respective parameters:

- **init(jsonContent)** - responsible for the middleware initialization process. A POI tree is provided via the ”jsonContent” parameter, which will be parsed into runtime objects;
- **retrieveCurrentPoi()** - requests the middleware for the most relevant POI, considering the current visitor's position.

PoiManagerListener This interface specifies the callbacks sent by the middleware to the application layer:

- **enteredNode(nodeId)** - triggered event each time a new node location condition is met, updating the current POI;
- **exitedNode(nodeId)** - triggered event each time the determined visitor's location invalidates a previously active location condition, *i.e.* when a node ceases to be the most relevant for the visitor;
- **currentPOI(nodeId)** - returns the current POI's identifier, if it exists.

JSON parsing Upon receiving the JSON content via the init() API call, it needs to be parsed into memory objects through a set of JSON adapters. As part of a JSON parsing library, JSON adapters where used to convert JSON objects into Kotlin data class. Some JSON elements could be converted directly, others needed the definition of custom adapters, e.g. the ”location” properties. After defining the direct and custom adapters for all expected JSON properties, a ”Tree” class object is obtained and stored in memory while the library is operating.

Tree controller The Tree object is saved inside the Tree controller, implemented through a simple class. The TreeController class is responsible not only for store the

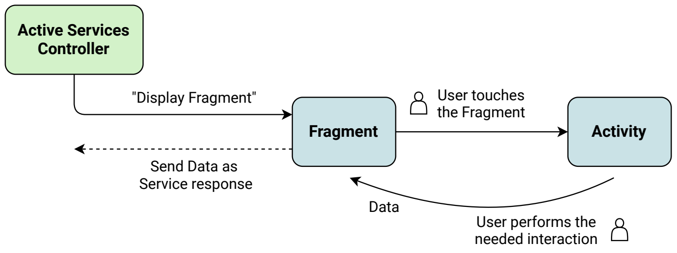

Active Services Controller
"Display Fragment"
Fragment
User touches the Fragment
Activity
Send Data as Service response
Data
User performs the needed interaction

Fig. 4. Interaction between Service UI components

runtime content of the POI, but also a linked list containing the path between the tree's root and the "most relevant POI node" or current node. Furthermore, some methods are available to easily access and modify this linked list while determining the most relevant node.

Service UI While the majority of the location services can operate seamlessly in the application's background, some services rely on user interactions to to operate properly, *e.g.* NFC and QR codes. Given these required interactions, some UI elements must be defined, in this case, Fragments and Activities. Figure 4 describes the interaction between these components.

When the defined Fragments receive requests from the Active Services Controller, they become visible (or hidden) on the application's UI. Upon user interaction, the Fragment will invoke the Activity to obtain the required information. Lastly, this information is sent back to the runtime library components and used as a regular location update callback.

To include these components into an application, the developer simply has to define a placeholder on the application's UI layout indicating where the desired Fragment will appear and instantiate its class as a regular UI component. The remaining logic is already defined on the middleware.

C. Location services

Two location services were defined on the middleware, one for obtaining geographical location callback and other to detect the nearest BLE beacon. Both services share a common operating principle: independent service classes, initiated/stopped by the Active Services Controller which send a set of Broadcast events (containing the location data) which will be captured by the Location Controller of the middleware through BroadcastReceivers.

1) FusedLocationProviderService: The Google Location Services API [24] have a "Fused Location Provider" that aggregates the GNSSs, cellular networks and WiFi location systems of a mobile device. A FusedLocationProviderClient was set up inside an external service class by configuring two objects: LocationRequest - for operational parameters such as update frequency and accuracy - and LocationCallback - defining what actions should be performed after a location is obtained. On the LocationCallback, a broadcast event was defined to send the determined geographical coordinates, which will be captured by a BroadcastReceiver configured at the PoiManager class. These coordinates will then be used to assess the most relevant POI to the visitor.

2) BeaconService: A BLE beacon detection service was also defined an external service class. Using the AltBeacon's library and its APIs [25], this service can estimate the distance to a set of beacons and determine which one is the nearest to the visitor. A thread was set up to provide regular update events. Each of these updates contains the nearest beacon identifier and an estimate for its distance to the device, or an empty message if no beacon is found. Similarly to the previous service, these updates are broadcast and captured by a BroadcastReceiver on the PoiManager. Once again, this information will be used on a set of POI assessment algorithms.

D. POI algorithms

Upon receiving a location update from a location service, a set of algorithms must search the POI tree for the visitor's most relevant node, by evaluating their location conditions.

1) Geographical nodes: Considering the location callback obtained for geographical nodes, *e.g.* a set o coordinates, two observations can be made:

- Only geographically described nodes can be evaluated;
- A single pair of geographical coordinates can match several POI nodes' area conditions.

With this conditions in mind, three algorithms (two of them recursive) were created for determining the most relevant POI from a single pair of coordinates, navigating through the entire POI tree. Every time a location match is found, a recursive downwards search algorithm attempts to find the lowest level that still match that condition. If no matches are found downwards, other recursive algorithm is invoked to search the tree upwards (an then downwards is a new match is found). Regarding the middleware events, each time a new node is determined to be the most relevant, a corresponding "enteredNode" event is triggered. On the other hand, if no matches are found, an "exitedNode" event is sent to the application.

2) Beacon nodes: The search algorithms for beacon identifier and distance callbacks share some similarities with the geographical ones, having also three connected search algorithms and the same events. However, in this case, each match condition is unique on the entire tree, *i.e.* only a single POI node will match the received beacon identifier. In order to avoid searching the entire tree trying to find a single match, a few search rules were defined. These rules were shaped to allow a common practical case on exhibitions: easily search neighbor beacon nodes (indoor rooms) as common children of a single geographical node (a building) while the visitor walks through the building exhibition.

E. Active services' management

The purpose of the Active Services Controller is to ensure only the necessary services are activated, given the current visitor's location. Considering the previously described services'

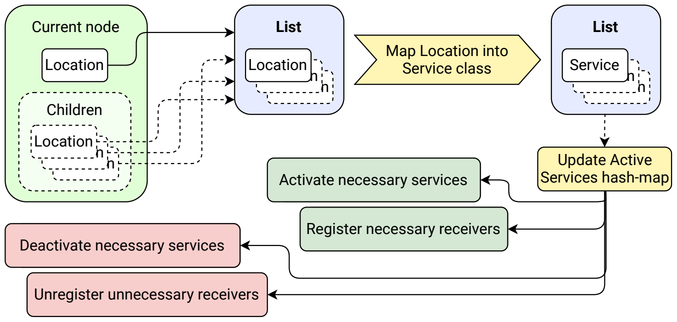

Current node
Location
Children
Location
n
n
List
Location
n
n
Map Location into Service class
List
Service
n
n
Update Active Services hash-map
Activate necessary services
Register necessary receivers
Deactivate necessary services
Unregister unnecessary receivers

Fig. 5. Active services' controller operation

implementation, this controller must start (and stop) the services' classes and register (and unregister) the correspondent receivers. This controller's operation is described on figure 5.

The first requirement for this controller is to store which services are currently active. Secondly, some mapping algorithms must be defined in order to obtain the correspondent service class for a given location type. The final step is to determine which location types are relevant given the current node. In order to be able to detect the current node's exit condition and its children (if exist) enter conditions, a list containing the current both node's and its children location types is built and then mapped to a list containing the corresponding service classes which will be used to only activate the necessary services and their respective receivers.

F. QR code integration

As a demonstration for the middleware feature to also support user interaction dependent location services, a QR code scanning service was integrated. Two connected UI components were defined:

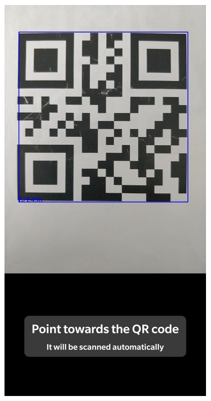
- **IndicatorFragment** - indicator shown to the user when a QR code is available. When the user touches this Fragment, it launches the QRCodeActivity, expecting it to return a read value. This component;
- **QRCodeActivity** - camera activity that automatically scans QR codes. After a successful scan, the scanned value is returned to the Indicator Fragment.

These UI components are shown on figure 6. In order to integrate them on the middleware runtime operations, a helper QRCode service class was created which is activated by the Active Services Controller each time the current node has one or more QR code children. This service sends broadcast messages to a receiver included on the IndicatorFragment, requesting it to be visible or invisible on the UI. When the Fragment receives the QR code value from the Activity, it broadcasts an event to a POIManager BroadcastReceiver (like any other location service receiver). Therefore, using the already defined structures of the middleware, a complex location system was integrated in a similar fashion as the others.

G. Demonstration application

In order to correctly configure and integrate the proposed middleware, a demonstration application was developed. The logic diagram of the application is illustrated on figure 7

(a) Indicator-Fragment



(b) QRCodeActivity

Fig. 6. UI elements presented to the visitor for QR code scanning.

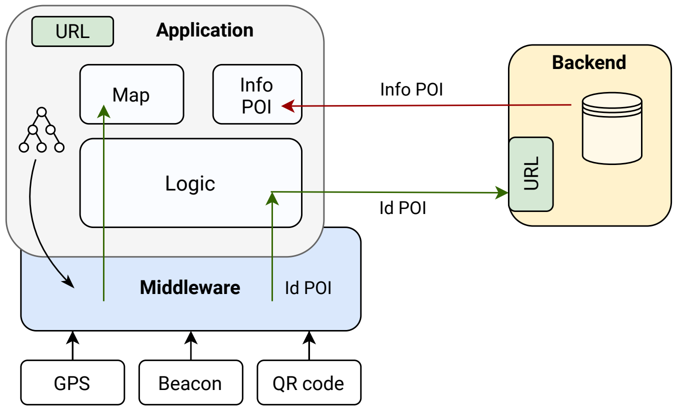


Fig. 7. Application logic diagram

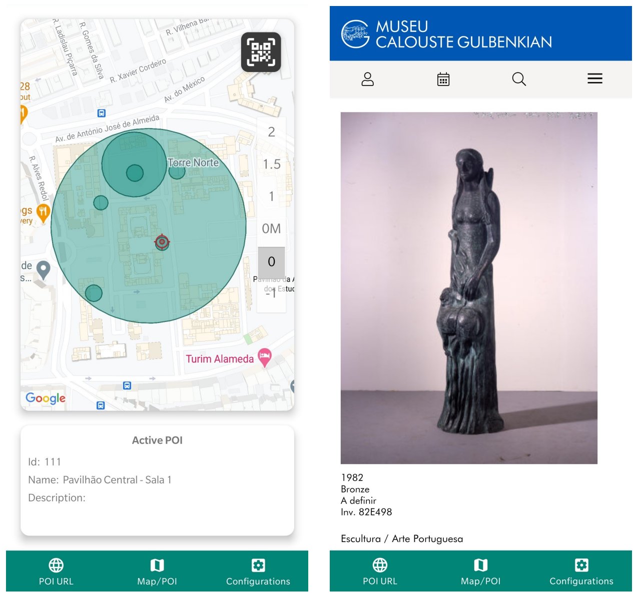
The application starts by instantiate the middleware by loading a POI tree contained on a JSON file. The middleware is configured to use three location systems: GPS, BLE and QR codes. These systems will provide location callbacks to the middleware which will determine the most relevant POI and send its identifier to the application. In order to obtain further information about the POIs, the application has an URL reference to a backend to which it will send the POI identifiers. Lastly, after receiving the POI information from the backend, the application will display this information to the visitor.

The demonstration application has two main screens: one for displaying a map and the current POI's information and other to display web pages (when the current POI has an associated URL). Both of this screens are displayed on figure 8. Figure 8a displays the previously defined QR code IndicatorFragment.

V. Evaluation

A. Functional requirements validation

Regarding the identified functional requirements on section III-A, the following topics demonstrate how the implemented



Active POI
Id: 111
Name: Pavilhão Central - Sala 1
Description:

1982
Bronze
A definir
Inv. 82E498

Escultura / Arte Portuguesa

(a) Map/POI screen (b) WebView screen

Fig. 8. Main screens of the demonstration application

middleware features fulfilled those requirements:

Exhibition curator

1) The proposed JSON schema provides a set of guidelines which allow the definition POI tree structures. In these structures, a wide variety of POIs is supported, enabling not only relationships between exhibition areas but also items inside them;
2) The proposed JSON schema has a location property which supports three types of location identifiers: geographical, relative and symbolic. These identifiers will be supported by corresponding location services;
3) In order to make changes to an already defined exhibition structure, it only necessary to modify the corresponding JSON file and load it into the application.

Developer

1) The implemented middleware greatly simplifies the development process of an exhibition navigation application, by providing area or item identifiers to the application using it;
2) The JSON defined POI trees are not included into the application's code. If a POI tree needs to be modified, the application will remain unchanged;
3) The proposed middleware provides POI identifiers the application;
4) The developed middleware abstracts the whole location services logic from the developer as they are defined inside the middleware and solely managed by it. when a location service is added or modified, it is only necessary to compile a new version of the middleware and include it on the application (no need to change its code);
5) Accordingly to the visitor's location at any given time, the middleware disconnects the location services that serve no purpose for such location, ensuring only the necessary services are kept running;
6) the developed middleware lays the foundations for presenting buttons on the application's UI to activate specific location services.

Visitor

1) The implemented middleware includes a set of components which enable location services that rely user interactions;
2) As the management of the active location services is automatically performed by the middleware, the visitor does not need to manually turn on or off each one while visiting an exhibition.

TABLE I
NODE TYPING AND CORRESPONDING TIME SPENT FOR EACH DEMONSTRATION EXHIBITION

	Demo 1		Demo 2	
Type of node	Number of nodes	Time spent (minutes)	Number of nodes	Time spent (minutes)
Geographical	5	15	7	20
Beacon	20	60	14	42
QR code	53	-	45	-
Total time		75		62

B. *Performance*

In order to evaluate the middleware's operation and its energy consumption, two demonstration exhibitions were created to mirror real exhibitions and possible paths followed inside them by a visitor. A JSON file was built for each exhibition and both had similar layout: one geographical root node containing the entire exhibition venue, some geographical nodes corresponding to different buildings and indoor rooms identified through BLE beacons. Lastly, some QR nodes were added to each room to simulate displayed artwork. Table I presents the number of nodes for each location type and the time spent on each node type during the defined visit for each exhibition.

The geographical nodes of both exhibitions were placed inside the Instituto Superior Técnico campus. A Pycom [26] LoPy board was used to simulate the different BLE beacons. For simulating the retrieval of POI information from a web service, the Gulbenkian Foundation [27] artwork database was used and the QR codes on each room (defined by a beacon) were given identifiers for this database.

After defining the simulated exhibition visits, the next step was to define which were the scenarios for evaluating the middleware's performance. The chosen four scenarios were:

- **Idle** - for providing baseline measurements, the device was kept only with the screen on during the visits' duration;
- **Optimized without QR code** - middleware performs an active services' management, *i.e.* it deactivates unnecessary location services according to the visitor's location, without scanning any QR codes;

TABLE II
MEASURED ENERGY CONSUMPTION DURING THE DEMONSTRATION VISITS

	Demo 1 (75 min)		Demo 2 (62 min)	
	Estimated consumption (mAh)	Percentage drop	Estimated consumption (mAh)	Percentage drop
Idle	102	5%	98	6%
Optimized w/o QR code	194	11%	162	10%
Optimized with QR code	252	12%	170	10%
All Services On	425	18%	302	13%

- **Optimized with QR code** - middleware performs an active service management and the available QR codes are scanned;
- **All Services On** - all location services are kept active by the middleware and available QR codes are also scanned.

On all the scenarios, except for the ”Idle” one, the demonstration application was kept open and only the middleware managed the location services. The user followed the pre-defined path for each demonstration visit and simulated the entering and exiting of rooms by changing the advertised beacon identifier.

In order to evaluate the energy consumption in the different test cases, two indicators were used. The first is the battery percentage drop given by the device’s operating system and the other is determining the energy consumed by the device. This latter measurement was performed by an external current meter.

Analyzing the obtained energy consumption measures on table II, the ”Idle” scenario obtained the lowest energy consumption, as expected. On the other hand, the highest energy consumption was obtained on the scenario where all the location services were kept active during the visits. Comparing this scenario with the scenario where the middleware performed an active service management (”Optimized with QR code”), the energy savings were about 40%. Given the majority of the simulated visits duration is spent inside rooms identified by BLE beacons, having the GPS service to be turned off in these areas achieved a significantly lower energy consumption.

Considering the two scenarios when the active service management is performed by the middleware but differ on the scanning of the available QR codes, one can observe the scanning process resulted on a measurable impact on the energy consumption. The increased consumption is justified not only by repeatedly accessing the camera but also by the several network requests to obtain the items’ details.

In order to evaluate the impact of the middleware on the size of an application that incorporates it, a comparison was made between the demonstration application (developed on section IV) and the same application without importing the middleware’s classes. The Android Studio APK analyzer was used to compare the final sizes of both applications. Table III contains the obtained size differences.

Regarding the total size of the APK, where the entirety of the application is compressed on a single file, one can asses that the inclusion of the middleware increased the file size by less than 350 KB. Considering the size of the application’s classes, when including the middleware into the application, the uncompressed size of this classes increased 753.8 KB. Therefore, the inclusion of the middleware did not have a significant impact on both the classes’ and the APK’s sizes, which suggests the majority of the application classes correspond to core libraries of the Android environment.

TABLE III
SIZE COMPARISONS BETWEEN AN APPLICATION WITH OR WITHOUT INCLUDING THE MIDDLEWARE

	Without Middleware	With Middleware	Difference
APK total size (compressed)	2.1 MB	2.5 MB	335.4 KB
Application classes size (uncompressed)	1.3 MB	2.0 MB	753.8 KB

VI. CONCLUSIONS

A. Accomplishments

This work starts by consider a wider definition for the POIs and how hierarchical relationships can be established between them. In other words, not only relevant areas can be treated as POI but also smaller areas and relevant items contained inside them as well. In order to properly describe the arrangement of a wide diversity of POIs, a JSON schema was developed. This schema enforces a set of guidelines for defining these structures while supporting several location description techniques for each element.

The defined POI structures were defined to be external to any exhibition navigation application. In order to access an exhibition layout, this type of applications simply need to load the corresponding POI structures at run-time. Thus, if future modifications are made on the physical layout of the exhibition, the POI can be updated independently of the application and, consequently, it is not necessary to recompile it.

The other main contribution of this work is a middleware that simplifies the development process of exhibition navigation applications. By receiving a compliant structure to the JSON schema defined above and aggregating the location callback of several location services, the middleware is able to determine which is the most relevant POI at any given time for an exhibition visitor.

One the main abstractions provided by the middleware is the integration of several location services. Therefore, during the development process of applications that utilize this middleware, the developer does not need to integrate and configure any location service. Furthermore, the middleware also is able to perform an automatic management of the running services.

By analysing the visitor's location relative to an exhibition's layout (using its POI representation), the middleware only activates the necessary location services for that location. This management not only further decouples the application developer from the services by also allows for significant energy savings while using exhibition navigation applications. Considering the scenario of an exhibition application that does not have any kind of active service management, the inclusion of this middleware can obtain energy savings of about 40%, without requiring any extra effort to the application developer.

B. Future work

Despite the accomplishment of the identified requirements for this work, some future improvements could further improve this solution:

1) Support more complex relationships between POIs;
2) Variable runtime service parameters;
3) Integration of new location services.

References

[1] S. Medic and N. Pavlovic, "Mobile technologies in museum exhibitions," *Turizam*, vol. 18, no. 4, pp. 166–174, 2014.

[2] K. Best, "Making museum tours better: Understanding what a guided tour really is and what a tour guide really does," *Museum Management and Curatorship*, vol. 27, no. 1, pp. 35–52, feb 2012.

[3] B. Lord and M. Piacente, *Manual of Museum exhibitions*. Rowman & Littlefield Publishers, 2014.

[4] X. Wei and Z. Jianping, "Mobile Application Used in Museum Learning and Its Case Study," in *Proceedings - 2015 International Conference of Educational Innovation Through Technology, EITT 2015*. Institute of Electrical and Electronics Engineers Inc., apr 2016, pp. 90–93.

[5] H. Tsai and K. Sung, "Mobile applications and museum visitation," *Computer*, vol. 45, no. 4, pp. 95–98, apr 2012. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6178133

[6] R. Wojciechowski, K. Walczak, M. White, and W. Cellary, "Building Virtual and Augmented Reality Museum Exhibitions," in *Proceedings of the ninth international conference on 3D Web technology - Web3D '04*. New York, New York, USA: ACM Press, 2004.

[7] F. Borrero, P. C. Sanjuán Muñoz, and G. Ramírez González, "Técnicas de gamificación en el turismo, prueba de aplicación, Casa Museo Mosquera," *Sistemas y Telemática*, vol. 13, no. 33, pp. 63–76, 2015.

[8] A. Küpper, *Location-Based Services: Fundamentals and Operation*. John Wiley and Sons, dec 2005.

[9] X. Li, X. Zhang, X. Ren, M. Fritsche, J. Wickert, and H. Schuh, "Precise positioning with current multi-constellation Global Navigation Satellite Systems: GPS, GLONASS, Galileo and BeiDou," *Scientific Reports*, vol. 5, no. 1, p. 8328, feb 2015.

[10] H. Laitinen, S. Ahonen, S. Kyriazakos, J. Lähteenmäki, R. Menolascino, and S. Parkkila, "Project Number: IST-2000-25382-CELLO Project Title: Cellular network optimisation based on mobile location Cellular Location Technology," 2001.

[11] I. K. Adusei, K. Kyamakya, and K. Jobmann, "Mobile positioning technologies in cellular networks: An evaluation of their performance metrics," in *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 2, 2002, pp. 1239–1244.

[12] Wigle.net, "WiGLE: Wireless Network Mapping," 2020. [Online]. Available: https://www.wigle.net/https://wigle.net/ Accessed on 2020-08-07.

[13] A. Khalajmehrabadi, N. Gatsis, and D. Akopian, "Modern WLAN Fingerprinting Indoor Positioning Methods and Deployment Challenges," pp. 1974–2002, jul 2017.

[14] Bluetooth SIG, "Bluetooth Core Specification versiom 5.2," 2019. [Online]. Available: www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc{_}id=478726 Accessed on 2020-08-09.

[15] R. Faragher and R. Harle, "Location fingerprinting with bluetooth low energy beacons," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428, nov 2015.

[16] Y. C. Pu and P. C. You, "Indoor positioning system based on BLE location fingerprinting with classification approach," *Applied Mathematical Modelling*, vol. 62, pp. 654–663, oct 2018.

[17] B. Ozdenizci, V. Coskun, and K. Ok, "NFC internal: An indoor navigation system," *Sensors (Switzerland)*, vol. 15, no. 4, pp. 7571–7595, mar 2015.

[18] W3C, "Points of Interest," 2012. [Online]. Available: https://www.w3.org/2010/POI/wiki/Main{_}Page Accessed on 2020-08-11.

[19] Fiware, "PointOfInterest - Fiware-DataModels," 2020. [Online]. Available: https://fiware-datamodels.readthedocs.io/en/latest/PointOfInterest/PointOfInterest/doc/spec/index.html Accessed on 2020-08-12.

[20] R. L. Pereira, P. C. Sousa, R. Barata, A. Oliveira, and G. Monsieur, "CitySDK Tourism API - building value around open data," *Journal of Internet Services and Applications*, vol. 6, no. 1, pp. 1–13, 2015.

[21] M. Nitti, V. Pilloni, D. Giusto, and V. Popescu, "IoT Architecture for a sustainable tourism application in a smart city environment," *Mobile Information Systems*, vol. 2017, 2017.

[22] S. H. Hashemi and J. Kamps, "Exploiting behavioral user models for point of interest recommendation in smart museums," *New Review of Hypermedia and Multimedia*, vol. 24, no. 3, pp. 228–261, jul 2018.

[23] J. schema org, "JSON Schema — The home of JSON Schema," 2020. [Online]. Available: https://json-schema.org/http://json-schema.org/

[24] Google, "Google Location Services API," 2020. [Online]. Available: https://developers.google.com/android/reference/com/google/android/gms/location/package-summary Accessed on 2020-08-18.

[25] Radius Network, "Android Beacon Library," 2016. [Online]. Available: https://altbeacon.github.io/android-beacon-library/ Accessed on 2020-08-16.

[26] Pycom, "Pycom - Next Generation Internet of Things Platform," 2019. [Online]. Available: https://pycom.io/ Accessed on 2020-09-11.

[27] Fundação Calouste Gulbenkian, "Fundação Calouste Gulbenkian," 2020. [Online]. Available: https://gulbenkian.pt/ Accessed on 2020-08-07.