Smart Places

Samuel Filipe Capucho Mendes Coelho
samuel.coelho@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—Proximity-based applications engage users while they are in the proximity of points of interest. These apps are becoming popular among the users of mobile devices. Proximity based apps are triggered when the user is at specific geographic coordinates, but more interestingly, they can be triggered by tagged objects. We introduce the concept of Smart Place which is a physical place where tags are placed in order to provide a service. Users with a mobile device capable of detecting such tags can use the provided service when they are in proximity of these tags. Several technologies can be used to create tags.**

In this dissertation, we analyze several location technologies in order to choose one that best fits the concept of Smart Place. We have created a solution to develop proximity-based applications, based on this concept. Also, two examples of Smart Places were built: the Smart Restaurant and Smart Museum. The Smart Places apps allows anyone with a mobile device capable of detecting tags, to have access to any proximity-based service, based on our concept of Smart Place and created using the tool we developed. This app was evaluated in terms of energy consumption and the results show that the battery drain can be acceptable making it a viable technical approach.

***Index Terms*—proximity-based; mobile apps; smart place; location based applications**

I. Introduction

Nowadays, mobile devices are equipped with many sensors such as light sensor, Global Positioning System (GPS) and accelerometer. Also, these devices have access to multiple data sources such as the user's activity on social networks and calendar. The applications (apps) that the user can install have access to this data provided by these sensors and data sources. Using this data, the apps can adjust settings and allow users to perform tasks according to it. The mobile apps that use this information, which is called *context*, are named context-aware applications. For instance, an app that puts the phone in silent mode when the user is in a meeting is a context-aware app. The user's location is a particularly useful context information because it allows apps to offer the user special functionalities when they are in a specific place. The focus of this present work is location based applications.

Proximity-based is a particular type of context-aware that takes into account users' location. These apps offer different possibilities, to the users, when they are in the proximity of a given point of interest. We can find multiple examples of such apps. Swarm[1] allows users to check-in in a given place or point of interest, according to user's location. For instance, if the user is in a restaurant, he can use the app to perform a check-in in that restaurant and share his/her location on social networks. These apps are becoming popular. Besides these examples, it is possible to automate tasks based on location. For instance, send a message to someone when we arrive at a given location or turn on the lights when we arrive at home. Services such as If This Then That (IFTTT)[2] allows users to make these kind of automations. These are just examples of what is possible to do with context-aware and more specifically with location based applications. With the right tools and applications we can benefit from our mobile devices being aware of context and location.

To develop proximity-based apps we need to, somehow, get the device's location. However, we need to choose a technology to get this information and be able to associate data to points of interest, which we will refer as tags. Besides location, we need a backend to store the data about each tag. Developers have to develop the mobile apps, the backend and choose the right technology to get the location.

Besides development of proximity-based services another question arrises. How can an owner of a given place offer such services, without having to develop everything him/herself? For instance, a store owner wants to advertise some promotion in the customers mobile devices when they approach the store. How can he/she creates this promotion if there is no programming background? Also, if there are multiple promotions in different stores, customers need to install one app for each store. They should be able, by installing one app, to discover these promotions, or any other proximity-based service while they are

[1]http://www.swarmapp.com
[2]http://ifttt.com

walking, instead of being aware of these services and know which apps they need to install.

We created a tool to develop proximity-based services, removing the need to build a backend and to handle the location technology. Here we introduce the concept of Smart Place. The idea is to have a physical place, which offers a proximity-based service due to existence of tags placed on objects that are relevant. For instance, the example of the store can be considered a Smart Place. The owner place some tags, inside the store, that will trigger a notification in the customers' mobile devices, such as smartphones.

Next section describes all the concepts needed in order to understand our Smart Places solution. Section III presents related work about existing proximity-based applications and tools to develop such applications. Then, we introduce our solution in section IV. Section V explains the evaluation of the solution being presented here. Finnaly, section VI concludes this paper.

II. Background

We built a framework to develop proximity-based applications. However, we need to define the concept of proximity-based application. These are a particular kind of context-aware applications. An application is context-aware when it takes into account the context, such as, location, device's orientation, temperature, etc. Based on this definition, proximity-based applications are context-aware applications that take into account the user's location. They engage the user while they are on proximity of a given point of interest, which we will call a Tag. Someone installs tags in a given space. Then, users can interact with those tags when they are nearby them.

A. Smart Places

Our solution is based on the concept of Smart Place. Place's owner installs tags and those tags offer some service to the users when they are nearby. For instance, a store owner could use these tags to advertise some promotion when the customers are nearby the store. However, Smart Places are not just about stores. This concept can be used to any kind of service that requires the users to be nearby tags. For instance, it is possible to build a Smart Restaurant, where tags have the information about the table's number and the customers are allowed to call a waiter without requiring to type the table's number.

Multiple kinds of people are involved in Smart Places. There are users, owners and developers. The end users interact with tags, that are installed in Smart Places, using their mobile devices. Owners are responsible for managing and installing tags in their places, where they want to offer a proximity-based service. Developers are needed to develop these services, for instance, the Smart Restaurant or the store promotions.

B. Location

In order to understand what a Smart Place is, we need to have a good insight about concepts related to location. We have used a taxonomy[1] to classify some properties about location. These properties will allow us to have a better understanding about the concept of a Smart Place and the location technologies that can be used. It is possible to classify location systems in terms of techniques, physical position or symbolic location, absoulute or relative position, location computation, scale, recognition, cost and limitations.

There are three main techniques to compute the object's location, Triangulation, Proximity and Scene Analysis. Triangulation can be done via lateration, which uses distance measurements between two well known points and angulation, which uses measurements of angules relative to known points. Proximity measures how close the object is to a known point. In Scene Analysis we examine a view from a given point.

There is physical and symbolic location. Physical location is a set of coordinates that, identifies, unequivocally, a place where the object is. Using Symbolic location is not to identify where the object is. For instance, we could say that an object is in the kitchen or in the office but not in a given street and city.

There are two ways of computing an objects' location The object being located might have means to compute its own location or it can delegate that computation to an external infrastructure

C. Location Technologies

As previously mentioned, a Smart Place has tags and the users can interact with them using their mobile devices. Somehow, the mobile device needs to be able to detect the presence of these tags. Multiple technologies can be used. Some of them require the user interaction. Others require the devices to be equipped with extra hardware. We can consider the following technologies:

- Global Positioning System (GPS)[2] is a location system that uses 24 sattelites plus three backups.

Receivers send signals and sattelites answer back. Measurements are taken from this signals in order to receivers be able to calculate their own location.
- Quick Response (QR) Codes is a type of two dimensional barcode. The user just needs an app that reads these codes. Whenever the user sees one of these codes, he/she opens the Quick Response (QR) code reader app, scan the code and see the content provided by it.
- Near Field Communication (NFC)[3] is a short distance radio communication technology. The two devices communicating need to be of 10 cm or less distant from each other.
- Google Maps Indoor allows the user to navigate inside a building using Google Maps.
- Bluetooth Low Energy (BLE)[4] is a short range wireless communication technology, developed by Bluetooth Special Interest Group (SIG)[3]. Unlike classic Bluetooth, it is focused on low power consumption. It is a feature of Bluetooth 4.0[5]

Some technologies are tag based, such as QR Codes, Near Field Communication (NFC) and Bluetooth Low Energy (BLE) that can be used to provide symbolic location. Others, such as GPS and Google Maps Indoor, are used to provide physical location but can be augmented to provide symbolic location. However, GPS does not work properly indoors and Google Maps Indoor requires that buildings are mapped first by a team from Google™. Each technology was classified in terms of techniques, type of location (physical or symbolic), computation, scale, recognition of individual receivers, costs and limitations.

GPS and Google Maps Indoor do not require extra hardware. However, they were created to provide physical location and not symbolic as it is needed for the concept of Smart Place. GPS does not work properly indoors and not all buildings are mapped in Google Maps Indoor. QR codes can be used for symbolic location. They do not need extra hardware and there tools to generate these codes[4]. However, they require the user interaction. He/she needs to be aware of the existence of these codes and use his/her mobile device to scan the code and get access to the content it provides.

The final decision was to pick BLE using the iBeacon[5] protocol, which was developed by Apple™, because it has the best tradeoff between cost and kind of location it provides. Using this technology we have symbolic location. With the adequated infrastructure it is possible to associate any kind of information to each tag. It requires extra hardware but this is the responsability of who manages the place where the tags will be deployed. The user does not anything else but a mobile device with Bluetooth, version 4.0, or later. We have used three beacons, from Estimote™, in the implementation of our solution.

III. Related Work

Here we discuss some related work in order to have a good insight, about the possibilities and existing solutions, for the problem we are trying to solve. First, we take a look at some examples of proximity-based apps, in order to see what can be done when the user is in proximity of a given point of interest. Since we tried to create a solution that provides an easy way of creating proximity-based services, there is a need to look at the state of the art of existing solutions of frameworks to develop context-aware applications. As already mentioned, proximity-based applications are a particular category of context-aware applications that take into account the user's location.

We present some applications, where BLE beacons were used, to get good insights about the potential use cases of this technology:
- BlueSentinel[6]: This is a occupancy detection system, for smart buildings, that uses BLE Beacons to detect the presence of people. The concept of a smart building is similar to Smart Place, due to the existence of sensors and actuators. It is focused on the power efficiency of the building. The idea is to optimize energy consumption according to people's presence. For instance, if there are no people in a given room, the heating system can be turned off.
- ContextCapture[7]: In this work, the authors try to use context-based information to allow users to add more information to their status updates in the main social networks, such as Facebook[6] and Twitter[7]. The authors implemented a mobile app and a server integrated with Facebook and Twitter. Context information comes from the smartphone itself, from its sensors and from the nearby devices through Bluetooth. Devices can be other smartphones or BLE Beacons, which are used for indoor location.

Since we have created a framework to develop proximity-based services, the state of the art of ex-

[3]http://www.bluetooth.org
[4]http://goqr.me
[5]http://developer.apple.com/ibeacon

[6]http://www.facebook.com
[7]http://twitter.com

isting frameworks, that deliver context information to the apps will be presented:

- Frameworks for developing distributed location-based applications: There are frameworks to develop location-based applications. In the work presented in Krevl et al.[8], a framework was developed to allow developers to build location-based apps. Location information can come from any source, such as GPS receivers, Bluetooth receivers and Wireless Fidelity, Wireless Internet (WiFi) receivers. The authors discuss some benefits and limitations of several technologies for getting the user's positioning. The mobile device get geographical coordinates from any source and send that data, in a Simple Object Access Protocol (SOAP)[9] message, to the appropriate web service in the Application Server. This server, communicates with the Database Server to query the database, which sends back a response with location information, if there is any, for that particular group of geographical coordinates. The authors did not evaluate the system. This solution offers abstractions for the location information sources. Geographical coordinates can come from any source. The authors do not take into consideration constraints in terms of resources, such as lack of Internet connection and battery. Since most users have limited data plans for their smartphones and SOAP messages can grow, in size, due to its Extensible Markup Language (XML) format, a more efficient message encoding could be used instead.
- Dynamix[10]: is a framework to develop mobile native and web apps that allow them to receive context information, for instance, position and device's orientation. This framework has plugins that get one or more sensor's raw data and turn that into event objects, that contain more high-level information. This framework supports many kinds of context information and it is possible to develop more plugins to allow the apps to generate additional events that are not already supported.

In order to get a good insight about the potential of proximity-based services, we analyzed three applications.

We introduced a framework to develop distributed location-based applications and a more generic one that targets the development of context-aware applications. For each one, we described the idea, its main components, in order to get an overview about their limitations, which we try to circumvent in our solution. However, most of them only helps developers to write native apps, that is, apps written for a specific platform that will run only on it. This leads to a situation where the user needs to install one app, in his/her mobile device, for each proximity-based service he/she wants to use. In order to circumvent this limitation, this solution allows developers to use the same technologies that are used for any web application such as HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and Javascript to build Smart Places allowing them run on a web browser that can be embedded in a mobile app. This way, users only need to install one app to discover and use proximity-based services following the Smart Places approach.

The framework described in Krevl et al.[8] offers abstractions for the location information sources. Geographical coordinates can come from any source. It is a good approach for mobile native apps but it does not support web apps. The authors do not take into consideration constraints in terms of resources, such as lack of Internet connection and battery. Since most users have limited data plans for their smartphones and SOAP messages can grow in size due to its XML format, a more efficient message encoding could be used instead, for instance Representational State Transfer (REST) using JavaScript Object Notation (JSON).

To achieve our goal, our framework could be just a plugin for Dynamix. The plugin would need to get the beacon's raw data and turn that into a more high-level information using a backend. In this framework, the user needs to install an app that manages the service that runs in background and needs to define some security policies such as which information the app can have access to or which sensors it can use. This could mean a big overhead since we are more focused on developing proximity-based applications that do not require such complex security policies because in this kind of applications, there is only need to access the device's sensors that could provide positioning data to the applications.

IV. Solution

The main goal of our solution is to assist the development of proximity-based mobile applications. There are already some tools available for this purpose (presented in section III). However, some of them are attached to a given platform, that is, developers need to write one proximity-based application for each platform they want to reach. In order to circumvent this limitation, this solution allows developers to use the same technologies that are used for any web application, such as HTML, CSS and Javascript.

Before starting the description of our solution, we need to take a look at three kinds of users that will be part of it:

- End users: Anyone with a mobile device that installs an app to scan for nearby Smart Places;
- Owners: These users are the ones responsible for managing a given place that they want to turn into a Smart Place;
- Developers: The users that develop the code of the Smart Places.

Figure 1 shows the main components of our solution, which are the following:

- Beacons are the small devices that will act as tags, according to our definition of a Smart Place;
- Backend is where all the data is stored, that is, the Smart Places that are available, the Smart Places that each owner has configured, information about each beacon, etc;
- End Users Mobile App is another Android mobile app that allows users, with a mobile device, BLE enabled, to have access to nearby Smart Places and to detect the beacons that belong to those Smart Places;
- Owners Mobile App is an Android mobile app that owners use to select which Smart Places they want to configure. It also allows to configure each individual beacon that belongs to a given Smart Place;
- Developers Application Programming Interface (API) provides the necessary methods that developers can use to create their proximity-based services, based on the concept of a Smart Place.

Smart Places solution has a component that targets each one of the presented type of user. Owners have a mobile app that allow them to turn the places they manage into Smart Places. There is another mobile app to allow anyone, with a mobile device, such as a smartphone, to use the services provided by Smart Places nearby. To develop these services our solution offers an API that developers can integrate in their web applications to make them react to the presence of the user.

To create the backend, we have used Parse[8] Backend as a Service (BaaS). A BaaS is a particular type of Software as a Service, that allows developers to specify which entities and their fields, that they need to store. An BaaS allows developers to not have to be concerned about aspects related to distributed systems,

[8]http://parse.com

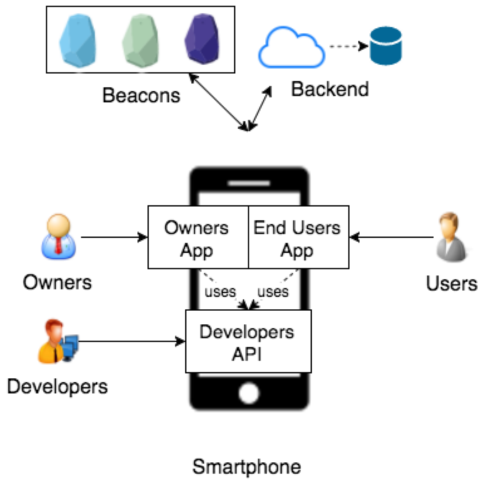

Fig. 1: Overview with the main components of our solution

such as, scalability and security. Since our backend only needs to be able to store and retrieve data, we chose this solution instead of doing an implementation of a REST API from the scratch.

Anyone that have a mobile device, such as a smartphone, can use the services provided by any Smart Place. In our solution, there is an Android app, that notifies the user when he is nearby any Smart Place. When the mobile device approaches any Smart Place, the app notifies the user that he is near a Smart Place. When the user touches these notifications, the app shows an embedded web browser that contains a web page that can react to nearby objects, that is, beacons with meaning to the application.

The Smart Places' owners manage one or more places where they want to provide some service to visitors in their mobile devices. In order to make owners be able to offer this kind of services an Android app designed for them is offered by this solution. This app offers features such as, get a list of all available Smart Places and configure an instance of a Smart Place. In order to configure a Smart Place first, owners need to tag physical objects. They need to deploy beacons in the right places. Then, they use the mobile app to create an instance of a Smart Place following a small set of steps. First, the app shows a list of all available Smart Places. Then, the owner selects one and he can see a text explaining what that Smart Place is about. Finnaly, the owner just needs to type a title and a message, that will appear in the users' mobile devices notifications when they are nearby

Owners configure the data of a Smart Place and end users can interact with objects nearby. But, who will add behavior to these Smart Places? Our solution offers a way for developers to create their Smart Places. Also, we want to avoid the user having to install one mobile app for each Smart Place. The app for end users has an embedded web browser, so they can use any Smart Place as they would use any web application without the need to install one more mobile app. That is why a Javascript library is part of our solution. This way, an existing web application can use this library and make it react to nearby objects tagged with BLE beacons.

The library was turned into an open-source project, hosted on a github repository[9] and it is available to install using bower[10], which is a tool to manage dependencies in web applications. Then, developers just need to include the library and use the available functions. The library is event-based, that is, the mobile apps, for owners and end users emit events to the library such as, a nearby beacon is detected to the web application running inside a embedded web browser. In this library there is a global object, which is "SmartPlaces" with several methods. All those methods need to receive a callback because, as already mentioned, the library follows an event-based approach.

We have created two examples of Smart Places, the Smart Restaurant and Smart Museum. The Smart Restaurant allows customers of a restaurant to place their orders without the need to wait. The Smart Museum allows museum's visitors to have access to more information, in their mobile devices, about a given object in an enxhibition, when they are in the proximity of that object. We first built the Smart Restaurant. While builiding this example, we wrote Javascript code to handle the events emitted by the mobile apps for owners and end users. From this code, it was possible to create a library that resulted in a complete independent project, from which, the examples depend on. After building the first example, we developed another one, which is the Smart Museum. We defined the Javascript library as a dependency and observed that the same API that fits the Smart Restaurant example, also could be used in the other example.

V. Evaluation

The evaluation focused on two aspects. First, we need to verify if the method to get the distance from a given beacon is reliable. Second, since we have a service running in background scanning for beacons from time to time we evaluated our solution in terms of battery consumption.

In the evaluation process, a smartphone and a set of three beacons from Estimote™ were used. The smartphone was a Motorola™ Moto G[11]. This device has the following specifications:

- Central Processing Unit (CPU): Quad-core 1.2 GHz Cortex-A7[12]
- Graphics Processing Unit (GPU): Adreno 305
- Random-access Memory (RAM): 1 Gigabyte (GB)
- Internal storage: 16 GB
- Screen: 4.5 inches
- Battery: Non-removable Li-Ion 2070 Milliampere-hour (mAh) battery
- Operating System (OS): Android 5.0.2 (Lollipop[13])

A. Nearest Beacon Detection

Our solution relies on a library, which its API allows us to get the distance from a given beacon. However, this value is related to the signal's strength, that comes from the beacon. We performed a set of experiments to verify how reliable was this value and if we can use that value to compute which beacon is the nearest one.

The set of experiments, is summarized in Table I. Figure 2 shows the layout that was used where value d is the distance between beacons. In these experiments, the Smart Musem example was used. In each experiment, it ran for 5 minutes using 10 seconds as the interval between each scan. 10 seconds was chosen because it is a reasonable value to walk in the museum to have enough time to perform any computation, that was needed, after each scan. Running the experiment for 5 minutes, with the mentioned interval between each scan, allowed us to have more than 20 scans. Then, in Android Studio log output, it was possible to check how many times each beacon was detected as the nearest one.

The mobile app for end users scans for beacons but only requests data for the nearest one. To get the nearest one, it has to rely on the signal strength to calculate the distance. We performed 4 experiments in order to try to get the accuracy of the mechanism

[9]http://github.com/samfcmc/smartplaces-js
[10]http://bower.io

[11]http://www.gsmarena.com/motorola_moto_g-5831.php
[12]http://www.arm.com/products/processors/cortex-a/cortex-a7.php
[13]https://www.android.com/versions/lollipop-5-0

Variables	**Experiments**			
	1	**2**	**3**	**4**
Number of beacons	3			
Interval between each scan	10s			
Experiment duration	5m			
Distance between beacons (meters)	0.5	1	1.5	2

TABLE I: Experiments to get the accuracy of the method to get the nearest beacon

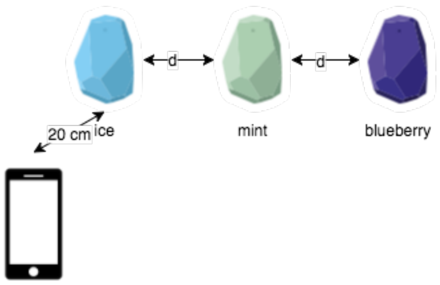

Fig. 2: Layout used for the experiments to get the accuracy of the distance value

that calculates the distance that the mobile device is from a given beacon.

For each experiment, we counted how many times each beacon was detected as the nearest one. Looking at the layout used in these experiments we can see that the app should detect the beacon named *ice* as the nearest one. Figure 3 shows the percentage of times that the beacon *ice* was detected as the nearest one, showing that, as we increase the distance between beacons, the accuracy to detect the nearest beacon also increases. From the results, we can conclude that it is recommended that the beacons are, at least, 1.5m or 2m distant from each other.

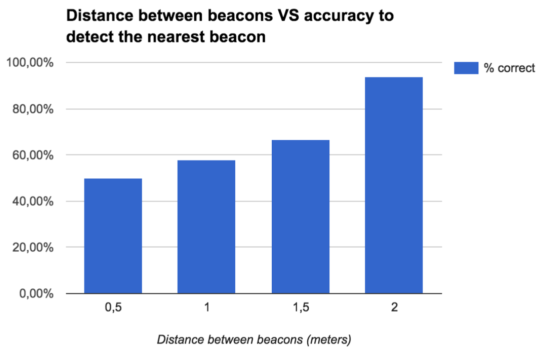

Fig. 3: Relation between distance between beacons and accuracy to detect the nearest beacon

In an environment, where the beacons are close to each other, our solution might not work as expected. For instance, in the previously described Smart Restaurant example, the tables should not be close to each other. This is not always possible because, some restaurants try to optimize space and have tables as much close to each other as possible. In the Smart Museum example, two objects, in a given exhibition, should not be close to each other. Otherwise, the visitor would be notified about an object and he might be looking at another one.

B. Energy Consumptions

Another important aspect of this solution is the battery consumption. Since our mobile app for end users runs on background to scan for nearby beacons, that can have a negative impact on the device's battery. If the user notices that the battery drains too fast, he will not use this solution.

Table II summarizes the experiments performed to evaluate the battery consumption. Figure 4 shows the layout used for this group of experiments. We have used the same beacons as in the experiments described in section V-A. The beacons are equally distant 25 cm from each other. The smartphone is at the same distance from the beacon in the middle, the green one named mint. We performed 4 experiments. In each one the app was turned on and ran for 1 hour in background mode scanning for beacons in order to discover nearby Smart Places. The first two used WiFi data connection. The remaining used Third Generation (3G) mobile network. Different data connection means can lead to different energy consumptions. We need to understand which connection, WiFi or 3G, drains more power. If it is 3G, the user might only use our solution if he/she is connected to a WiFi Access Point (AP). We want our mobile app to be always turned on scanning for nearby Smart Places. Using it only when WiFi is available would make its usage very limited and the user would not take the full advantage of it because he/she needs be aware that a WiFi AP is available and turn the mobile app on again. We tested two values for the interval between each scan, 5 minutes because it is the default value that the beacons library use in background mode, and 2 and half minutes. The second value is half the first in order to see how much more power is drained when we set a smaller value for the interval between each scan. Trying to find a smaller interval is important because it will reduce the probability that the user was not able to discover a nearby Smart Place.

The following scenarios were tested:

Variables	**Experiments**			
	1	**2**	**3**	**4**
Data connection type	WiFi		3G	
Interval between each scan	2m30s	5m	2m30s	5m
Experiment duration	1h			

TABLE II: Summary of experiments to get the battery consumption when the mobile app is scanning for beacons in the background

- When the user stays in the same Smart Place the entire experiment;
- When the user moves from one Smart Place to another. This is done removing existing data about Smart Places already detected to force the app to request this data again from the Backend in each scan process.

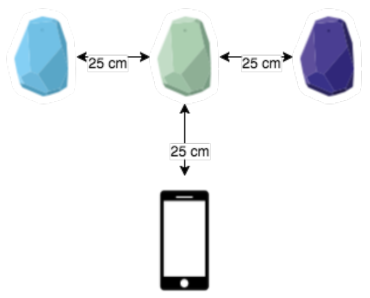

Fig. 4: Layout used for the experiments to get the battery consumption

Data communications, WiFi or 3G, are the major source of battery drain, as suggested in studies, such as [11]. We used Battery Historian[14] to measure the power drain and how much data was transferred (sent and received).

Figure 5 shows the results for the first scenario, where the user does not move. It is possible to see that using WiFi we got no power drain. However, using 3G it drained 0.29% and 0,40% of the total battery available. From this results it is possible to conclude that our solution introduces the most overhead when using 3G as the mean to perform data communications such as, communications with the backend.

After the first set of experiments we tested Facebook[15] since it is one of the popular apps and also has services running in background. It is also known

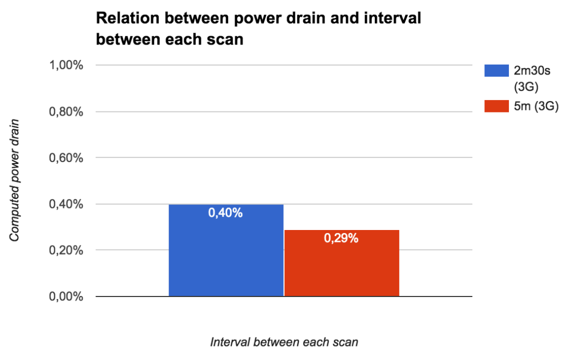

Fig. 5: Relation between power drain and interval between each scan in the scenario where the user stays in the same Smart Place

Variables	**Values**
Data connection type	3G
Experiment duration	1h
Computed power drain (%)	3.77
Data transferred (KB sent and received)	4627.83

TABLE III: Battery consumption of Facebook app

to be one of most power draining apps[16]. We let it ran for 1 hour as we did in our first experiments. While running Facebook, we used it to check our news feed in 5 minutes. The rest of the time the app was running in background. Table III summarizes the conditions and results of the test performed using Facebook app. We used these values as a reference to compare with the power drain in the second scenario, that is, at each scan the user moves from one place to another which implies more requests to the backend.

Then, we performed the second set of experiments, that is, the second scenario where the user moves from one Smart Place to another. Here, the app requests more data from the backend. Figure 6 shows the results of these experiments. Here, we got more power drain than in the previous scenario. These results shows that, the more communication with the backend is required, more power our solution drains. Once more, using WiFi we have got almost zero power drain. However, similar to the results in the previous scenario, there is more power drain using 3G. Using two minutes and half of interval between each scan, we got more 0.71% than using five minutes. As happened before, there is more power drain when we increase the interval between each scan.

[14]https://developer.android.com/tools/performance/batterystats-battery-historian/index.html

[15]http://play.google.com/store/apps/details?id=com.facebook.katana

[16]http://www.forbes.com/sites/jaymcgregor/2014/11/06/facebook-and-instagram-are-killing-your-phones-battery-heres-a-simple-fix

In the worst case we obtained a power drain of 2.71% which is approximately 71% less than using Facebook in the same period of time. Using this app as a reference in terms of apps that run services in background, we can say that the battery consumption is acceptable for a daily basis usage.

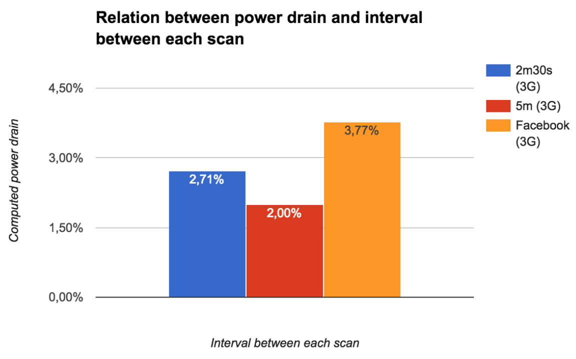

Fig. 6: Relation between power drain and interval between each scan in the scenario where the user moves along multiple Smart Places

VI. Conclusion

We have developed the Smart Places solution which offers a tool, for developers, to allow them to create proximity-based services and an interface for end users and owners. We introduced the concept of Smart Place which is a pyshical space with tags that mobile devices can detect and offer different possibilites to the users, according to each tag. We chose BLE beacons using iBeacon protocol because it is compatible with any smartphone with Bluetooth version 4.0 or above. The owners of Smart Places are responsible to place those beacons in the right place. Users do not need any extra hardware. They only need to download a single app and turn on the Bluetooth receivers of their mobile devices.

We have created a solution that allows developers, of Smart Places, that provide proximity-based services, using web technologies, such as, HTML, CSS and Javascript. With this tool, any web application can react to the presence of tags placed in a given Smart Place. This solution makes the development of Smart Places easier due to the fact that, developers do not need to handle the technology to handle tags neither the backend where the information about Smart Places and their tags is stored. There is a mobile app for end users, anyone with a mobile device, capable of reacting to tags in a Smart Place. Since each Smart Place is a web application they can run inside an embedded web browser in this mobile app, allowing users to have access to any Smart Place withouth the need to install a new native app for each one. There is a mobile app for owners of Smart Places that they can use to choose which proximity-based services they want to provide and configure the necessary tags.

Two examples of Smart Places were created, using our solution, a Smart Restaurant and a Smart Museum. The Smart Restaurant had the goal to allow customers, of a restaurant, to place their orders, after they take a sit using their mobile devices. Using this solution, customers do not need to specify which table the orders belong to. Using the tag system through BLE beacons, the app automatically get the table's number and add that information when the customer places the final order. In the Smart Museum we created the experience of a museum where visitors can have access to more information about an object that they are close to.

We evaluated our solution in terms of battery consumption. Users will not use our solution if they run out of battery faster than if they are not using our solution. For each experiment, we have tried two different types of data connection, using WiFi and 3G. Using WiFi the power drain is almost zero. However, using 3G, the power drain was above 2%, in just one hour. Comparing with the power drain of Facebook app running in the same period we can conclude that the power drain is acceptable but there is room for improvement. For instance, we could store geographical coordinates for each tag. When detecting a tag, the app could use these coordinates to fetch data about the other tags in the same area in just one request. This way, we need less requests to the backend which can result in less power drain than the actual solution.

A. *Future Work*

There are aspects of our work that can be improved in the future. Owners of Smart Places need to deploy tags and use the mobile app to configure those tags. However, there is not any interface, that they can use, to register themselves, as the owners of such tags. In the development of this work we introduced the needed data manually, that is, all the associations between tags and their owners. There is one more interface missing to allow developers to register the Smart Places they develop. When owners are configuring their Smart Places, they can choose from a list. The Smart Restaurant and Smart Museum examples were introduced in the backend manually. There is no way for developers to add a Smart Place to this list. A future improvement could be these missing interfaces. One that would allow owners to register the ownership of tags and another for owners that would use it to register the Smart Places they developed. Another limitation is the fact that, using our solution, only web applications can offer proximity-based services. In one side, this is a good fit because this is what allows to have one app to access multiple Smart Places. On the other side, there could be developers that could use our solution to add proximity-based features in their existing native mobile apps. This could be solved having an Software Development Kit (SDK) available for the three most used mobile OSs, iOS, Android and Windows Phone, to allow to integrate our functionality in existing apps.

Proximity-based applications are an important type of context-aware applications since they provide services, to the user, that are relevant in the right place. However, developers need the right tools to create these applications and users need an easy way to have access to such applications. Our work targets not only developers but also users and owners. A complete solution is needed because developers would not create proximity-based applications if there are no users. Also, users would not be engaged if developers do not have the tools to create these applications.

Proximity-based applications can offer several possibilities. For instance, companies that have online and physical stores could achieve more integration of offers such as, customers have access to exclusive discounts when they approach a given tag or a set of tags. Another possibility is to have tags in a supermarket, which could trigger reminders in an app that customers use to store their shopping list. For instance, the customer approaches the section where there is milk and this is on his shopping list. The app would remind him that he needs milk. Our Smart Places solution and its evaluation can be considered a step forward to be able to use, develop and manage all these and many more possibilities.

References

[1] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, 2001.

[2] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global positioning system: theory and practice*. Springer Science & Business Media, 2013.

[3] V. SHARMA, P. GUSAIN, and P. KUMAR, "Near field communication," in *Proceedings of the Conference on Advances in Communication and Control Systems-2013*. Atlantis Press, 2013.

[4] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[5] S. Bluetooth, "Bluetooth core specification version 4.0," *Specification of the Bluetooth System*, 2010.

[6] G. Conte, M. De Marchi, A. A. Nacci, V. Rana, and D. Sciuto, "BlueSentinel: a first approach using iBeacon for an energy efficient occupancy detection system," in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*. ACM, Nov. 2014, pp. 11–19.

[7] V. Antila and J. Polet, "ContextCapture," in *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*. New York, New York, USA: ACM Press, Sep. 2011, p. 585.

[8] A. Krevl and M. Ciglaric, "A framework for developing distributed location based applications," in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2006, p. 6 pp.

[9] S. Seely, *SOAP: Cross Platform Web Service Development Using XML*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[10] D. Carlson and A. Schrader, "Dynamix: An open plug-and-play context framework for Android," in *2012 3rd IEEE International Conference on the Internet of Things*. IEEE, Oct. 2012, pp. 151–158.

[11] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?" in *Proceedings of the 7th ACM european conference on Computer Systems - EuroSys '12*. New York, New York, USA: ACM Press, Apr. 2012, p. 29.