

ISCTE  **IUL**
Instituto Universitário de Lisboa

Department of Information Science and Technology

Find Me! – an Indoor Location System

Stuart Costa Martinho

Master in Telecommunications and Computer Engineering

Supervisor

João Carlos Ferreira, Assistant Professor,
ISCTE-IUL

Co-Supervisor

Ricardo Pontes Resende, Assistant Professor,
ISCTE-IUL

October 2018

Abstract

This work presents an approach of combining location information from beacons and local building information to give location and guidance to a user inside a complex building. This information can help user's orientation into unknown buildings through a smartphone. Beacons are installed on the building and emit signals that are converted to user current position, related with Bluetooth's range. The building data is generated by Building Information Modeling (BIM) model which is going to be used as input data of this project. This system is based on a mobile App (*Find Me!*), for Android OS (Operating System), which captures the Bluetooth Low Energy (BLE) signals, coming from the beacon(s), and shows, through a map, the location of the user 's smartphone, and his destination, and guides him to the desired destination.

Keywords: Guidance, Beacons, BIM Model, User location, Mobile App, BLE, destination.

Resumo

Este trabalho visa abordar uma combinação entre localização, proveniente de informação retornada por Beacons, com informação de edifícios de larga escala de modo obter a localização do utilizador e de seguida conseguir guiá-lo, dentro do edifício em que este se encontra. Deste modo, este projeto conseguirá ajudar estes utilizadores que se encontram dentro de um edifício que desconhecem, com orientações, através de um smartphone. São instalados Beacons no edifício, estes emitem sinais que, após recebidos pelo smartphone, irão ser convertidos em localização atual do utilizador. Este processo está relacionado com o raio de cobertura de sinais Bluetooth. Os dados respetivos ao edifício são gerados pelo modelo Building Information Modeling(BIM), estes irão ser usados como dados de entrada deste projeto. Este sistema baseia-se numa aplicação móvel (*Find Me!*), desenvolvida em Android como sistema Operativo, que captura sinais Bluetooth Low Energy (BLE), proveniente dos beacons, e apresenta, através de um mapa, a localização e o destino do utilizador, guiando-o até ao destino pretendido.

Palavras-chave: Orientação, Beacons, modelo BIM, Localização do utilizador, Aplicação Móvel, BLE, destino.

Acknowledgments

I would like to express my very great appreciation to my Supervisor, João Carlos Ferreira, and to my Co-Supervisor, Ricardo Pontes Resende, who provided great guidance and objective reviews to this project.

My special and sincerely thanks to my mother and father, Orquidea and Francisco, who always worked a lot to give me all the tools that I needed to finish my graduation. Without them, it would not be even possible to conquer my objectives.

To my girlfriend who gave me enthusiastic encouragement in every moment that I needed it. Also, to my colleague Vicente Pereira, and students from Architecture that gave me precious contributions to this project.

My sincere thanks to ISCTE-IUL Maintenance and ISCTE-IUL Technology and Information departments who also gave me invaluable help in our ISCTE-IUL pilot tests.

Finally, a huge thanks to Ricardo Vilas Boas that made possible the reconcile between my job and the available time to finish this project.

Contents

| | |
|---|-------|
| Abstract..... | i |
| Resumo | iii |
| Acknowledgments | v |
| Contents | vii |
| List of Figures | xi |
| List of Tables..... | xvi |
| List of Acronyms..... | xviii |
| Chapter 1 Introduction..... | 1 |
| 1.1 Motivation and Framework | 1 |
| 1.2 Objectives | 2 |
| 1.3 Project Outline | 2 |
| Chapter 2 Literature Review | 3 |
| 2.1 Indoor Location Technologies..... | 3 |
| 2.1.1 QR Code..... | 3 |
| 2.1.2 NFC | 4 |
| 2.1.3 GPS..... | 5 |
| 2.1.4 WiFi Triangulation | 5 |
| 2.1.1 Beacon | 6 |
| 2.1.2 IL Technologies Comparison..... | 7 |
| 2.2 Building Information Models | 8 |
| 2.2.1 Maps for orientation | 9 |
| 2.3 Mobile Application Development..... | 10 |
| 2.3.1 Swift..... | 11 |
| 2.3.2 Android Studio | 11 |

| | | |
|-----------|--|----|
| 2.3.3 | Mobile Application Development Verdict..... | 12 |
| 2.4 | The A* Path Finding Algorithm | 13 |
| 2.5 | Database | 14 |
| 2.5.1 | External Database..... | 14 |
| 2.5.2 | Local Database | 14 |
| 2.5.1 | Relational Database Management Systems | 14 |
| 2.5.2 | Data Storage Verdict..... | 15 |
| 2.6 | Related Projects | 15 |
| 2.6.1 | ILS based on QR Code technology | 15 |
| 2.6.2 | ILS based on NFC technology | 16 |
| 2.6.3 | ILS based on WiFi triangulation | 17 |
| 2.6.4 | ILS based on Beacon technology | 18 |
| Chapter 3 | <i>Find Me!</i> Conceptual Modules..... | 21 |
| 3.1 | Find Me! Overview | 21 |
| 3.1.1 | User Roles | 23 |
| 3.2 | System Architecture | 28 |
| 3.2.1 | Beacon Installation Process | 30 |
| 3.2.2 | Mobile App Back End | 31 |
| 3.2.3 | Mobile App Front End..... | 32 |
| Chapter 4 | <i>Find Me!</i> Implementation | 33 |
| 4.1 | Beacon Installation Process | 33 |
| 4.1.1 | Beacons | 33 |
| 4.1.2 | Beacon Physical Placement | 35 |
| 4.1.3 | BIM Model..... | 49 |
| 4.2 | Find Me! Mobile App | 51 |
| 4.2.1 | Mobile App Back End | 52 |
| 4.2.2 | Mobile App Front End..... | 59 |

| | |
|--|----|
| 4.2.3 Entities Relation | 64 |
| Chapter 5 Implementation at the ISCTE-IUL Campus | 65 |
| 5.1 Beacon Installation Process | 65 |
| 5.2 Implementation Results | 68 |
| 5.2.1 Setup | 69 |
| 5.2.2 Find Me! Tests and Results..... | 70 |
| Chapter 6 Conclusion | 91 |
| 6.1 Future Work..... | 92 |
| References..... | 93 |

List of Figures

| | |
|---|----|
| Figure 2-1 - QR code example with "Hello World" information as content. | 3 |
| Figure 2-2 - NFC chip example [4]. | 4 |
| Figure 2-3 - Beacon device. | 6 |
| Figure 2-4 - Estimote Proximity Beacon (left) and BlueCats AA Beacon (BC-313) (right). | 7 |
| Figure 2-5 - 3D section views of the BIM model of ISCTE-IUL's Building 1 – Sedas Nunes. | 9 |
| Figure 2-6- Hand-drawn floor plan of an apartment (left) and a Schematic cross- section of a multilevel building (right). | 10 |
| Figure 2-7 - The Android software stack/architecture [15]. | 12 |
| Figure 2-8 - A* algorithm process [16]. | 13 |
| Figure 2-9 - Positioning by QR_STU. | 16 |
| Figure 2-10 - ILS App's Flow Chart [20]. | 17 |
| Figure 2-11 - The test bed of the experiment. | 17 |
| Figure 3-1 - Find Me! use cases diagram. | 23 |
| Figure 3-2 - Destination insertion mode options. | 24 |
| Figure 3-3 - FIND ROOM option. | 24 |
| Figure 3-4 - FIND FAST options. | 25 |
| Figure 3-5 - Orientation View example. | 25 |
| Figure 3-6 - Elevator button selected. | 26 |
| Figure 3-7 - FIND PHOTOS view. | 27 |
| Figure 3-8 – Screenshot example of Current Floor (left) and Destination Floor (right). | 27 |
| Figure 3-9 - Find Me! architecture. | 29 |
| Figure 4-1 - NearestBeacons scan values example. | 36 |
| Figure 4-2 - Floor no 0, 1 and 2 (left to right) with placed beacons. | 37 |
| Figure 4-3 – Beacon Placement practical example. | 37 |
| Figure 4-4 - Test case 1 layout. Floor 0 (left), floor 1 (right). | 38 |
| Figure 4-5 - Test Case 1 values, Floor 0, user position (left), NearestBeacons App results (right). | 39 |
| Figure 4-6 - Beacon Placement with a steel plate on the top example. | 39 |
| Figure 4-7 - Test case 2 layout. Floor 1 (left), floor 2 (right). | 40 |

| | |
|--|----|
| Figure 4-8 - Test Case 2 values, Floor 2, user position (left), NearestBeacons (right). | 40 |
| Figure 4-9 - Test case 3 layout. Floor 0 (left), floor 1 (middle) and floor 2 (right). | 41 |
| Figure 4-10 - Test Case 3 values, Floor 1, user position (left), NearestBeacons (right). | 41 |
| Figure 4-11 - Test Case 3 values, Floor 2, user position (left), NearestBeacons (right). | 42 |
| Figure 4-12 - Test Case 3 values after configuration changes, Floor 0(left) and Floor 1(right), NearestBeacons. | 42 |
| Figure 4-13 - Test Case 3 values after configuration changes, Floor 2, NearestBeacons. | 43 |
| Figure 4-14 - Test case 4 layout. Floor 1 (left) and Floor 2 (right). | 43 |
| Figure 4-15 - Test Case 4 values, Floor 1, user position (left), NearestBeacons (right). | 44 |
| Figure 4-16 - Test Case 4 values, Floor 2, user position (left), NearestBeacons (right). | 44 |
| Figure 4-17 - Beacons Placement rules diagram. | 45 |
| Figure 4-18 - Beacons Distance Rules diagram. | 47 |
| Figure 4-19- Creating Estimote Account. | 47 |
| Figure 4-20 - Beacon Name configuration | 48 |
| Figure 4-21 - Configure Protocol and Broadcast Packets | 49 |
| Figure 4-22 - Main Dynamo script for beacon and door/room tables production. | 50 |
| Figure 4-23 - Floor plans generated from the BIM model for user visualization (left); internal app pathfinding algorithm support with walkable areas in red (right). | 51 |
| Figure 4-24 - Estimote dependency. | 52 |
| Figure 4-25 – Estimote API, Connect method. | 53 |
| Figure 4-26 - Estimote API, Monotoring Beacons method | 53 |
| Figure 4-27 - Room and Beacon tables structure. | 55 |
| Figure 4-28 - Room CSV data example. | 55 |
| Figure 4-29 - Beacon CSV data example. | 56 |
| Figure 4-30 - Maps floor storage. | 57 |
| Figure 4-31 - setAppropriatedMap method. | 57 |

| | |
|---|----|
| Figure 4-32 - Example of Map floor with marked walkable path in red..... | 58 |
| Figure 4-33 - Map view screen description | 60 |
| Figure 4-34 – Image View set process. | 61 |
| Figure 4-35 - Choose Destination options | 62 |
| Figure 4-36 – FINDROOM option..... | 63 |
| Figure 4-37 - FINDFAST option | 63 |
| Figure 4-38 - Sequence Diagram of Orientations Process..... | 64 |
| Figure 5-1 - Maps of floors 0,1 and 2 of the pilot building - Edificio 1..... | 65 |
| Figure 5-2 - Floor 0 with beacons placement. | 66 |
| Figure 5-3 - Floor 1 with beacons placement. | 67 |
| Figure 5-4 - Floor 2 with beacons placement. | 67 |
| Figure 5-5 - Smartphone Huawei P8 gra-l09 | 69 |
| Figure 5-6 - Estimote Proximity Beacon..... | 70 |
| Figure 5-7 – Test 1 - Floor 1 map with current (yellow) and destination (red) locations marked..... | 71 |
| Figure 5-8 - Test 1 - Floor 1 with paths possible options marked. | 72 |
| Figure 5-9 - Test 2 - floor 1 with the current location (yellow)..... | 73 |
| Figure 5-10 - Test 2 - floor 2 with the destination (red)..... | 73 |
| Figure 5-11 - Test 2, subtest 1 - floor 1 with path options. | 74 |
| Figure 5-12 - Test 2, subtest 1 - floor 2 with path options. | 75 |
| Figure 5-13 - Test 2, subtest 2 - floor 1 with path option..... | 76 |
| Figure 5-14 - Test 2, subtest 2 - floor 2 with path option..... | 76 |
| Figure 5-15 - Test 3 - Floor 1 map with current (yellow) and destination (red) locations marked..... | 77 |
| Figure 5-16 - Results of Test case 1, Beacon ED1P1E intersection. | 78 |
| Figure 5-17 - Results of Test case 1, Beacon ED1P1F intersection..... | 79 |
| Figure 5-18 - Results of Test case 1, Beacon ED1P1G intersection..... | 80 |
| Figure 5-19 - Results of Test case 1, Beacon ED1P1H intersection..... | 81 |
| Figure 5-20 - Results of Test case 2, Subtest1, Beacon ED1P1D intersection... | 82 |
| Figure 5-21 - Results of Test case 2, Subtest1, Beacon ED1P1C intersection... | 83 |
| Figure 5-22- Results of Test case 2, Subtest1, Beacon ED1P2C intersection... | 84 |
| Figure 5-23 - Results of Test case 2, Subtest1, Beacon ED1P2B intersection... | 85 |
| Figure 5-24 - Results of Test case 2, Subtest2, Beacon ED1P2D intersection... | 86 |

| | |
|---|----|
| Figure 5-25 - Results of Test case 2, Subtest2, Beacon ED1P1E(left), ED1P1F(middle) and ED1P1G(right) intersection..... | 86 |
| Figure 5-26 - Results of Test case 2, Subtest2, Beacon ED1P2G (left), ED1P2H (middle), ED1P2A (right) and ED1P2B (bottom) intersection..... | 87 |
| Figure 5-27 - Results of Test case 3, Beacon not intersected message. | 88 |
| Figure 5-28 - Results of Test case 3, Beacon ED1P1H intersection. | 89 |

List of Tables

| | |
|--|----|
| Table 2-1 - Pros and Cons of different ILS devices [10]..... | 7 |
| Table 2-2- Benefits of MYSQL and Oracle [19] | 14 |
| Table 4-1 – Broadcasting Power values converted to Maximum Range in meters.[24]..... | 34 |
| Table 5-1 - Beacons configuration values | 68 |

List of Acronyms

| | |
|-------|---|
| 3G | 3 rd Generation |
| 4G | 4 th Generation |
| 5G | 5 th Generation |
| AOA | Angle of Arrival |
| AP | Access Point |
| API | Application Program Interface |
| BIM | Building Information Model |
| BLE | Bluetooth Low Energy |
| CSV | Comma-Separated Values |
| DB | DataBase |
| GPS | Global Positioning System |
| ID | Identifier |
| IDE | Integrated Development Environment |
| IL | Indoor Location |
| ILS | Indoor Location System |
| INS | Indoor Navigation System |
| IPS | Indoor Positioning System |
| ISCTE | Instituto Superior de Ciências do Trabalho e da Empresa |
| IUL | Instituto Universitário de Lisboa |
| NFC | Near Field Communication |
| OS | Operating System |
| PFA | Path Finding Algorithm |
| PNG | Portable Network Graphics |
| QR | Quick Response |
| RDBMS | Relational Database Management System |
| RSS | Received signal strength |
| RSSI | Received Signal Strength Indicator |
| SDK | Software Development Kit |
| TOA | Time of Arrival |
| URL | Uniform Resource Locator |
| WiFi | Wireless Fidelity |

Chapter 1 Introduction

1.1 Motivation and Framework

In modern buildings, people easily and frequently get lost, mostly because the building layout is complex, orientation signs are insufficient in number and clarity, and the environmental cues such as sunlight are missing. People usually resort to wander and explore on their own, then ask for directions and sometimes get lost again. This situation is more usual in big and complex buildings such as hospitals, airports, shopping's, and museums. This people's life problem can be easily solved by an Indoor Location System (ILS).

ILS, also known as Indoor Positioning Systems, serve the purpose of finding an electronic device inside a building, typically with a simple smart device: phone, tablet or watch. The method of finding the device can be implemented via Bluetooth, infrared, magnetic field and/or WiFi. When the smart device connects to some of the Indoor Location devices, the system can store that data and takes many conclusions about that. An example of an Indoor Location technology is Beacon Technology. A beacon is a device that broadcasts a Bluetooth Low Energy (BLE) signal in a limited and configurable range. This signal can be interpreted as the location of the person inside of a building, without the need of Internet.

Most people have their own smartphone these days so the best and easiest way to help with the orientation is through a mobile App such as the *Find Me! App*. This software's function is to show on a map, where the user is and how to get to the selected destination, guiding him until he reaches it. The most important and crucial parts of Find Me! App is the user's current location (from the beacon) and the desired user's destination. Having these two locations, as data, a Path Finding Algorithm (A* Search Algorithm Type [2]) calculates the shortest way between these points and draw it on a map with the goal of guiding the user to the selected destination. When the user intersects another beacon region, the drawn path is updated with a new user's current location.

1.2 Objectives

The objective of this project is to create an ILS mobile App that gives guidance to the user, giving the right directions until the user reaches the desirable point. The proposed work uses BLE beacons location information, indoor maps in BIM format floor plan, and routing algorithms in a mobile device. This proposal is applied to ISCTE-IUL university campus as a validation approach, and there is an intention of real usage of this work.

1.3 Project Outline

This project is organized as follows:

- **Chapter 1 Introduction:** describes the motivation and the defined objectives of this project are described;
- **Chapter 2 Literature Review:** provides the validation of state of the art technologies, and their descriptions;
- **Chapter 3 Find Me! Conceptual Modules:** describes the user roles and the system architecture of this project;
- **Chapter 4 Find Me! Implementation:** provides the implementation process of the system architecture mentioned in Chapter 3;
- **Chapter 5 Implementation at the ISCTE-IUL Campus:** evaluates the Find Me! project, applied to the ISCTE-IUL Campus;
- **Chapter 6 Conclusion:** concludes this project approach and the features that can be done/improved in the future.

Chapter 2 Literature Review

In this section, we present the technologies and approaches applied to this project, which include the description of available Indoor Location Technologies; the description of the BIM model; the presentation of available technologies to develop a mobile App; an overview of different types to implement a Data Base; a description of Path Finding Algorithm, which will be applied to this project approach; and finally an explanation about different ways and IL technologies used in related ILS projects.

2.1 Indoor Location Technologies

In this section, we describe the available IL devices [1] that can be integrated into our project proposal. Firstly, we start with a brief description of the technology and, based on that, we conclude it with a verdict.

2.1.1 QR Code

The Quick Response code (QR code) [2], invented in 1994 in Japan, is a two-dimensional barcode that holds a high capacity data storage. One QR code can store more than 7089 characters, which, compared to a regular barcode that only stores 20 digits, is much superior. It can be read/scanned by a smart device through his camera on both dimensions, vertically or horizontally. If a QR code is partially destroyed/degraded (but not more than 30% of its content) can be read as well. Figure 2-1 shows an example of a QR code.

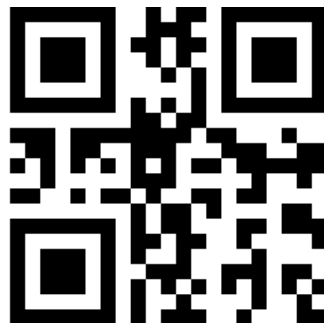


Figure 2-1 - QR code example with "Hello World" information as content.

Each structure of each QR code depends on the string that the user wants to return when read by a smart device camera. Note that the code matches with an exact

combination of characters. For example, if the user creates a QR code with the string “AB” and another code with just “A”, both of them will be completely different.

An ILS system applied with this technology is based on QR codes identification, which is located in several places in the building. Having these QR codes located at different positions in the building, allows users to determine, by reading through a smart device camera, their current location.

QR code Verdict

For the perspective of the building management, QR code technology is cheap because the codes can be simply printed in a paper. This technology holds high accuracy in terms of Indoor Location. In terms of User usability, the user needs to switch on the smart device camera and look for the closest printed QR code to know his current location – needs to do a manual location recognition. Comparing to other devices, this technology has the worse usability.

2.1.2 NFC

Near Field Communication (NFC) technology [3] is based in tags that have chip content and NFC readers. The communication of these two components consists of radio waves transmission (of 13.56 MHz) between NFC readers and NFC tags. Usually, an NFC tag holds a small distance radio-frequency, and thus, to scan a tag successfully, the NFC reader needs to be close to it. A smart device work as a NFC reader. NFC chip can store and also send, to the NFC reader, approximately 8Kb of data. Figure 2-2 shows a NFC chip example.

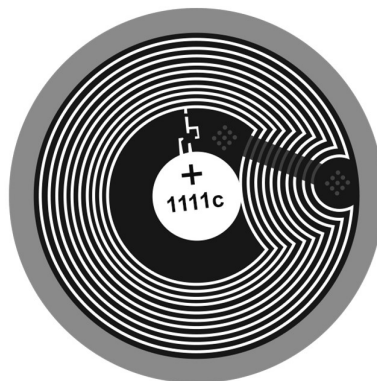


Figure 2-2 - NFC chip example [4].

NFC Verdict

The data transmission between NFC reader and the NFC tag is fast and the price of each tag can be cheap so the budget for the ILS concept is not expensive neither hard to implement. To get the user's current location, there is a need to get the smartphone close to the NFC tag which decreases the user usability, when using NFC technology in an ILS approach.

2.1.3 GPS

Global Position System (GPS), developed by the USA, in the middle of 1973, to improve the navigation systems. It is based on radio navigation signals that come from 24 GPS satellites. All the satellites send radio signals that provide location, status and precise time to the GPS receiver. A device with GPS receiver/reader enabled, can interpret the signal and calculate the distance from each satellite and translate into a distance from four satellites (at minimum). Then uses geometry to determine the device location on earth. This technology applies to both Indoor and Outdoor Location device.

GPS Verdict

GPS is the best location technology for outdoors systems, but for ILS has the negative way of not having a precise location. Inside of the building, it cannot locate where is the device in a specific room.

2.1.4 WiFi Triangulation

Geometric properties of triangles, consists in hot-spots that are physical implemented in several fixed places. Each of them provides the received signal strength (RSS), the angle of arrival (AOA) and the time of arrival (TOA). When a device intersects one or more of these hot-spots, calculates the distances between the target place and the hot-spots and determine, by triangulation, the position of the device. The Position calculation method requires a data service that accesses the packets triangulation sent by each hotspot and determines the final position of the device.

WiFi Verdict

In WiFi triangulation, it is challenging to have good precision and requires a background algorithm to manage the triangulations which sometimes is not useful and can be

confusing to handle. Having internet access and data transfer as the main required points, when one of these requirements is not available, the entire ILS system is compromised.

2.1.5 Beacon

A Beacon is a small Bluetooth Radio Transmitter that repeatedly broadcasts synchronous Bluetooth Low Energy (BLE) [5] signal in a restrict region/area. Each signal contains configurable data that can be received by a smart device. A beacon example can be observed in Figure 2-3 .



Figure 2-3 - Beacon device.

Estimote Beacon

It was implemented by *Estimote, Inc* - a company founded in 2012 by Jakub Krych and Łukasz Kostka, graduates from Jagiellonian University and AGH University of Science respectively. Nowadays, Estimote sells five types of products [6]: Location UWB Beacon, Location Beacon, Proximity Beacon, Sticker Beacon, and Video Beacon.

The most appropriated *Estimote* product to use in an Indoor Location concept is the Proximity Beacon. This device holds an average of 2-3 years of battery life and a configurable range of maximum 70 meters in open field.

BlueCats Beacon

In 2011, *BlueCats* was founded by Cody Singleton, Kurt Nehrenz and Nathan Dunn in USA and Australia. Indoor location devices are the core of this company. There are 3 types of beacons available on the marked: AA Beacon(BC-313) [7], Coin Beacon[8] and USB Beacon[9].

The most designated *BlueCats* product for complement this project is the AA Beacon (BC-313). This indoor location device holds waterproof and a battery life of maximum 5 years depending on the device performance.

Beacon Verdict

Beacons are the most recent indoor location technology, it is easy to configure and to maintain. It has its own API for the developers that want to implement an ILS. The security and Privacy of the users are safe, and the cost of each equipment is not expensive.

Both beacon types, *Estimote* and *BlueCats*, have very similar characteristics so we are sure that both would fulfill all the requirements of this project. Since the price and the characteristics of both beacons are similar, we chose the *Estimote* beacon because the design of the product holds a better look. Figure 2-4 shows the design of both products, *Estimote* Proximity Beacon and *BlueCats* AA Beacon (BC-313).



Figure 2-4 - *Estimote* Proximity Beacon (left) and *BlueCats* AA Beacon (BC-313) (right).

2.1.6 IL Technologies Comparison

In this section, we discuss the pros and cons of an ILS using the different indoor location devices that were referred in section 2.1. Table 2-1 outlines the pros and cons, considering the ILS characteristics of each IL device.

Table 2-1 - Pros and Cons of different ILS devices [10]

| | QRCode | NFC | GPS | WiFi | Beacon |
|---------------------------|--------|-----------|-----------|--------|--------|
| Accuracy of Tracking | High | High | Low | Low | High |
| Effort to Maintain | Easy | Difficult | Difficult | Easy | Easy |
| Industry Uptake | Medium | Medium | Low | High | High |
| Security and Privacy | Medium | High | High | Medium | High |
| Install, Maintenance Cost | Low | Low | High | High | Medium |

In this table, we are evaluating the following categories:

- **Accuracy of Tracking:** with a scale of Low, Medium, and High, this parameter shows the location precision that this ILS device can return to the system;
- **Effort to Maintain:** with a scale of Easy, Medium, and Difficult, this parameter shows the difficulty of maintaining the equipment;

- **Industry Uptake:** with a scale of Low, Medium, and High, this parameter shows how much the ILS is acquired by customers in the global market;
- **Security and Privacy:** with a scale of Low, Medium, and High, this parameter shows how much the level of Security and Privacy that this ILS can provide to the system;
- **Install, Maintenance Cost:** with a scale of Low, Medium and High, this parameter shows the installation and maintenance cost level that the ILS needs.

Comparing all the evaluated categories from Table 2-1, Beacon technology, which holds the best average between Pros and Cons, is going to be chosen to complement this ILS approach.

2.2 Building Information Models

The Building Information Model, or BIM, is a 3D description of buildings which associate information with the geometry of the building and its contents such as fixings, furniture or spaces. As an example, a parametrized BIM model is aware that a specific door is a double door, glass made, has level 1 fire-rating, connects corridor 1S to room 1S04, opens to the room, on the left-hand-side, etc. ISCTE-IUL's facility management office has been developing a BIM model which is being used to feed maps, room listings, and locations, as well as beacon's locations. This proves a significant advantage because since the BIM models are data-based, can be queried and updated (currently not in real time) with information to and from the App. Moreover, BIM is not only a model but also a new process of work in Architecture, Engineering, and Construction (AEC). This very traditional and conservative industry is very siloed: information does not flow between stakeholders, such as Architects and Engineers, and between the stages of buildings' lives: conception, design, construction, operation, rehabilitation and decommissioning. In the present case, geometrical information on the building that was generated during design, updated construction, and room utilization information which is stored on the university's IT systems is not available or is in a non-usable format for the path-finding App development.

The BIM methodology depends on common, interoperable formats, but also on the controlled information sharing that breaks down information silos. In the current case, a BIM model of the campus is being developed by the campus Facilities Management team [10]. This model, shown in Figure 2-5, includes the complete geometrical description of the building including all room names. To make it useful to this application

more information was added, namely: elevators and stairs were modeled, beacons were modeled with all relevant location, all areas of the building were parametrized as walkable and non-walkable. The software used for the development of the BIM model was Autodesk's Revit version 2019.

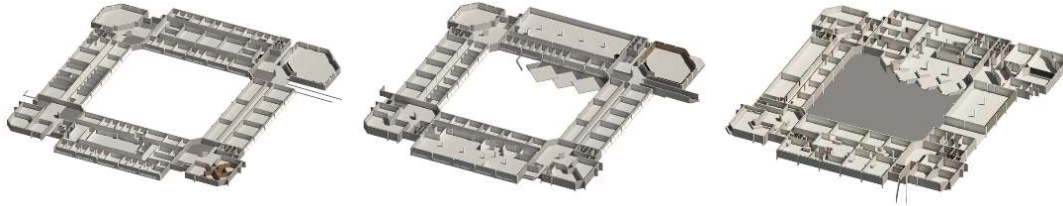


Figure 2-5 - 3D section views of the BIM model of ISCTE-IUL's Building 1 – Sedas Nunes.

To extract information in the format demanded by the application, the Dynamo scripting language [11] was used. This visual programming language gives access to Revit API which allows for the manipulation and export of information in XML format, namely a list of rooms with their coordinates, name, floor and building; a list of beacons with coordinates and id information and the coordinates of the nearest stair and elevator, and automatically generated images of the walkable regions in each floor.

2.2.1 Maps for orientation

The navigation space in large buildings such as schools, public buildings or shopping centers is frequently complex, comprising rooms and corridors, stairs, escalators, lifts and ramps. Traditional architectural drawings such as floor and site plans, elevations and cross sections and more natural perspectives can be used, with adaptations, to aid in the navigation.

A floor plan, Figure 2-6, is a rectangular projection from above showing the arrangement of spaces of a level of a building, as seen if a horizontal section is cut through a building typically at 1.20 m above floor level. It shows anything that could be seen below that level: the floor, walls, windows and door openings, stairs from the floor until the section level and sometimes furniture. This is the most commonly used kind of representation in orientation. In general, the public is used to this kind of drawing, but a considerable proportion of users have difficulty in reading plans.

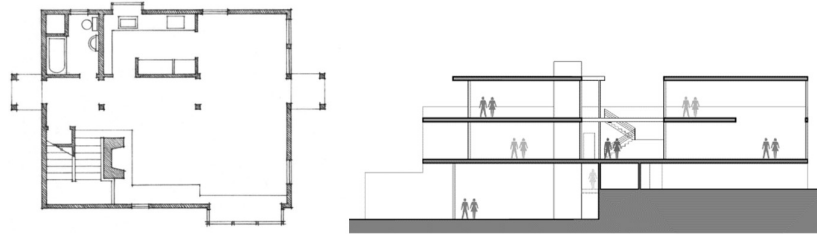


Figure 2-6- Hand-drawn floor plan of an apartment (left) and a Schematic cross-section of a multilevel building (right).

Elevations and cross sections, Figure 2-6 (right) as an example, are rectangular projections in the horizontal direction. The difference between them is the while elevations display the exterior of the building, cross sections show the interior, as seen as if a vertical plan cuts open the building. They are a valuable way to illustrate the relationship between levels of the building, but not very often used in navigation.

A more sophisticated category comprises the several kinds of three-dimensional projections. These are more easily read by lay users since they present a more natural way of showing the interior of buildings, walls, doors, and windows, passages, and furniture. Finally, the first-person view, as popularized in action games, allows users to relate what they see on the screen with reality. It is increasingly being employed, but it is much harder to implement since it demands a more precise location, a more detailed, 3D, description of the building and more powerful 3D graphics processing.

The choice of which elements are represented in the drawings and their level of detail is important in navigation. Most structural and construction details, materials and element thickness are not valuable in this scenario, but prominent decorative features or floor and wall colours or lining materials may be. On the other hand, visual aids such as human figures, shadows, transparency, exploded perspectives, enlargement of evident features or animations, even if not rigorous, give the observer a better grasp of the environment and direction.

2.3 Mobile Application Development

In this section, we describe 2 different types of programming languages to develop a mobile App. In the end, we conclude what is the most appropriate programming language to this project approach, with a verdict.

2.3.1 Swift

Developed by Apple in 2014, this programming language is general-purpose, multi-paradigm, compiled programming language. It is possible to create Apps to iOS, macOS, watchOS and tvOS, most of the equipment produced by Apple. It is an open source language. “Swift is the result of the latest research on programming languages, combined with decades of experience building Apple platforms. Named parameters brought forward from Objective-C are expressed in a clean syntax that makes APIs in Swift even easier to read and maintain” [12].

2.3.2 Android Studio

Based in Linux [13], Android is also an Open Source operative system bought by Google in 2005. This platform able the developers to create and implement applications to smart devices, like tablets, smartwatches or smartphones. A comparative study made in 2016 [14], mentioned that Android OS shares 86.2% on the whole market. It means that most of the people use Android as OS in their smartphones.

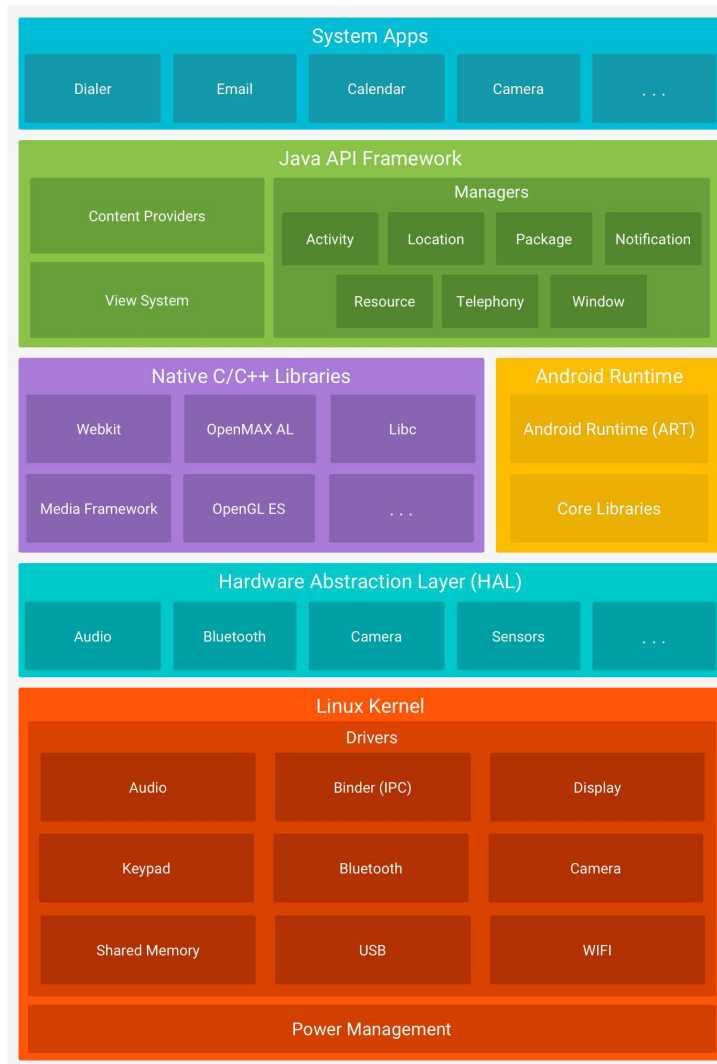


Figure 2-7 - The Android software stack/architecture [15].

As an Open Source software, Android made available his own Integrated Development Environment (IDE) – Android Studio – to all the people that need/want to create their own mobile Apps. Holds its own dedicated Software Development Kit (SDK) and development tools – Android Developer Tools (ADT). Several companies are adopting this platform because there are many people using smart devices with Android, the budget to develop applications it is usually low and, and has high potential for customization (Open Source).

2.3.3 Mobile Application Development Verdict

Developing in Android Studio would avoid a budget investment of an Apple laptop/desktop and because it is an open source, and available to several OS types. A normal laptop/desktop with Windows OS is enough to do it (which is the equipment that

is available right now to build this project proposal). Also, as it is referred in section 2.3.2, most of the people, that holds a smartphone, use Android as a mobile operating system. For these reasons, Android Studio is going to be our choice to do the mobile App development.

2.4 The A* Path Finding Algorithm

A star, or A*, hold the objective of calculating lowest route/path cost from a current/initial point which is called node, to the destination node out of one or more possible nodes. It is based on an evaluation function: $f(n) = g(n) + h(n)$. The $h(n)$ is optimal path cost estimate from node n to the destination node and $g(n)$ is described as the current cost from the current/initial node to any node n , with other words, the optimal path cost finding.

A* traverses the map, it follows the path with the lowest cost while keeping alternative nodes in a sorted priority queue. If a node being traversed has a higher cost than another encountered node at any point, it discards the node with the higher-cost and traverses the lower-cost node instead. This process continues until the goal is reached.

The provided map can be configured with two types of components: nodes and obstacles. To each node will be provided a cost, depending on the initial and destination node but for obstacles, it is not provided any cost so the A* does not consider keeping it on its queue. Figure 2-8 describes, in an objective way, the basic concept of A* algorithm as a flowchart.

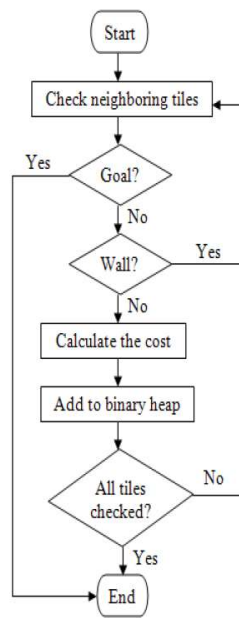


Figure 2-8 - A* algorithm process [16]

2.5 Database

In this project, there is a need to have a database (DB) to store all the information that is needed for this proposed ILS. The database can be implemented in one of the two available options: External or Local.

2.5.1 External Database

The DB is implemented on a physical Server or in a Cloud. The advantage of having an external DB is that every information is not stored locally so it won't affect the device performance. The disadvantage is that creates a dependency of the internet need to the mobile App and can be compromised when there is no WiFi or 3G/4G/5G access.

2.5.2 Local Database

The DB is implemented locally on the device. The advantage is that it doesn't need internet connection to request data from the mobile App to the DB. The main disadvantage affects the performance of the device substantially and could overload the device memory if there is a huge amount of data to store.

2.5.1 Relational Database Management Systems

There are several Relational Database Management Systems (RDBMS), the most relevant on the market are MySQL [17] and Oracle [18]. Considering the Table 2-2, *My SQL* has more advantages than *Oracle* so it is going to be used as RDBMS.

Table 2-2- Benefits of MySQL and Oracle [19]

| MySQL | VS | ORACLE® |
|-----------|--------------------|-----------|
| ★ ★ ★ ★ ★ | NUMBER OF FEATURES | ★ ★ ★ ★ ★ |
| ★ ★ ★ ★ ★ | PRICE | ★ ★ ★ ★ ★ |
| ★ ★ ★ ★ ★ | FLEXIBILITY | ★ ★ ★ ★ ★ |
| ★ ★ ★ ★ ★ | RELIABILITY | ★ ★ ★ ★ ★ |
| ★ ★ ★ ★ ★ | TECH SUPPORT | ★ ★ ★ ★ ★ |

2.5.2 Data Storage Verdict

When having **External Server Database**, there is a need for always having a data transfer/internet access, and if there is no internet access for some moment, all the ILS can be compromised. For this reason, a **Local Database** is our choice to manage the storage part in this proposed ILS. The Database is just going to be populated when the user installs the Find Me! App.

2.6 Related Projects

In this section, it is going to be analyzed projects that implemented an ILS in a complex building with the different available IL technologies.

2.6.1 ILS based on QR Code technology

Ilkovičová, Ľ., Erdélyi, J. and Kopáček, A. proposed an Indoor Location System using QR codes. The QR codes were positioned on several places of the Block A, Faculty of Civil Engineering building - Slovak University of Technology in Bratislava. It was developed a Java mobile application for Android OS named QR_STU, besides the download and installation of this App, the user needs to find the closest QR code and scan it with QR code reader that is already configured in QR_STU App. The scanned QR code returns an URL that allows identification of the user's location. With this URL, the application redirects to an external server, where an interactive map is stored where is already the painted current location of the user. To know how to get to the intended destination, the user just needs to select the room that he wants to go, after that a path is drawn from current user location to the selected destination and able the user to choose some icons to know more information about the destiny room. In Figure 2-9, there is an example of QR_STU working, where the destination is the room 202.

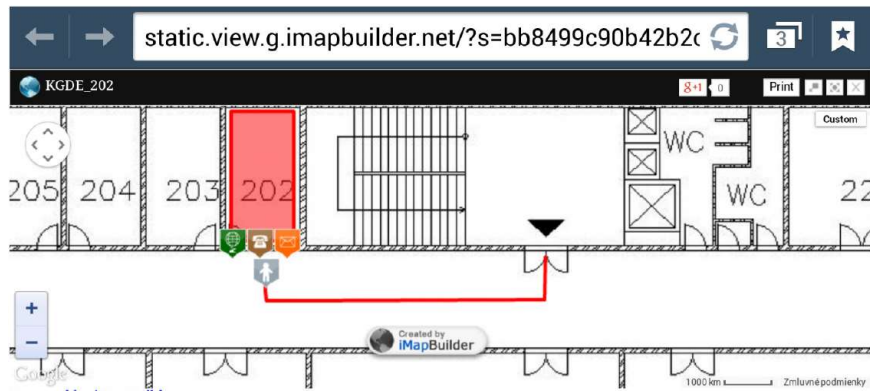


Figure 2-9 - Positioning by QR_STU

With this concept, it is not possible to continuously give orientation to the user because it only shows the user the first current location and the destination, if the user gets lost need to scan another QR code to know where he is, which can be a significant problem as User of usability this ILS.

2.6.2 ILS based on NFC technology

Busra OZDENIZCI, Kerem OK, Vedat COSKUN, Mehmet N. AYDIN planned an Indoor Location System applied with NFC tags [20]. The NFC tags are placed and distributed for all over the building, and a smartphone is used as an NFC reader. The ILS user interface is presented on a mobile App. Inside the building, the user seeks for the NFC tag and read the it with the Indoor Navigation App, using his smartphone camera. The read tag returns an URL that will be sent to a Map server that will interpret it and responds the map floor that corresponds to the tag position, which is the same as the user's current location. The map is processed and converted to a link-node model with topological relationships - composed by roads, corridors, ways, the path between buildings, rooms, halls, stairs, lifts, and doors - on the mobile App side. Rendering the map with the user's location on it, the user is allowed to choose the destination. Having these two locations, a Dijkstra's pathfinding algorithm calculates the shortest path between them, as a background process. In the end, it is shown the map with the calculated way between the destination and the position of the tag. If the user wants to update his current location of the map, needs to read another NFC tag. Figure 2-10 represents what is written above in this section.

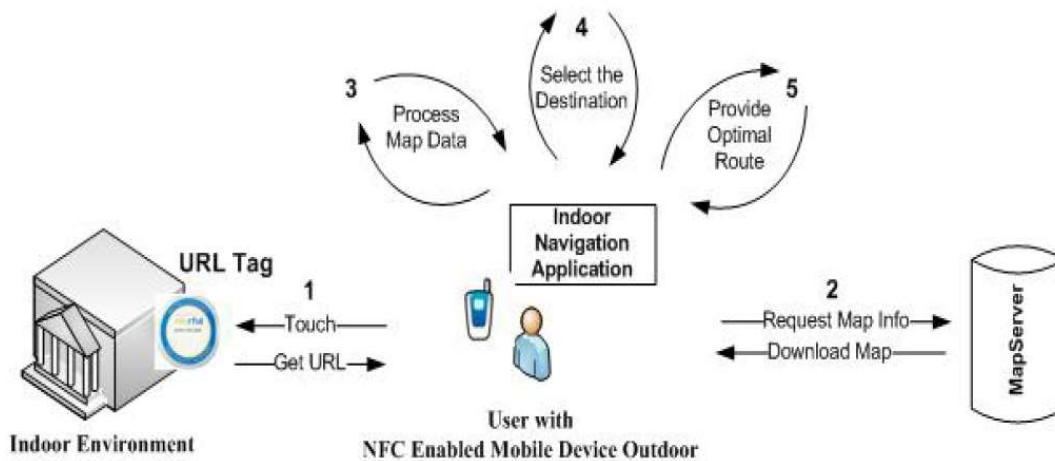


Figure 2-10 - ILS App's Flow Chart [20]

The negative part of implementing NFC tags technology is that the user needs to get his smartphone very close to the tag (which decrease the user usability). The orientation path is always the same from the first moment that he reads the tag until he reaches the destination. If he wants to confirm if he is in the right way, needs to read tag by tag to update it.

2.6.3 ILS based on WiFi triangulation

A paper published by Zhao Kai Li Binghao and Andrew Dempster in 2013 consisted of an ILS using WiFi Triangulation to get a location of, in this case, a laptop. It was tested in on the 4th floor of the Electrical Engineering Building at UNSW (University of New South Wales), Sydney.

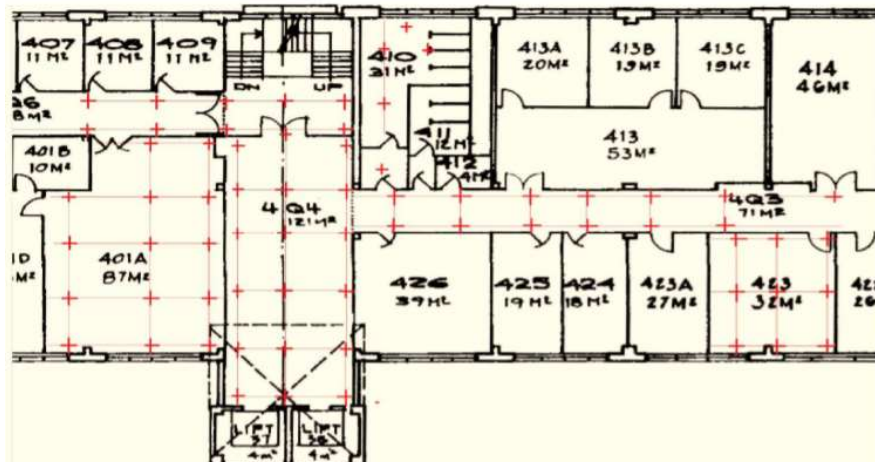


Figure 2-11 - The test bed of the experiment.

As the name describes, the WiFi triangulations consists on a location estimation of a device based on geometric properties of triangles. Several WiFi Access Points (AP) are implemented on the building, the laptop can receive the WiFi signal - the time of arrival (TOA), the angle of arrival (AOA) and the received signal strength (RSS) - from each WiFi AP that is in its range. With the content of each WiFi signal, is possible to calculate the distances between the laptop location and the WiFi APs. Having the location of three or more WiFi APs, the location of the laptop can be calculated/estimated by triangulation.

The precision and the building structure are important factors that can determine the quality of the ILS using WiFi. Internet Access and data transfer are needed, so when the bandwidth is low, it can compromise all the system and make it unpredictable or even useless.

2.6.4 ILS based on Beacon technology

A paper [21], made by Xin-Yu Lin, Te-Wei Ho, Cheng-Chung Fang, Zui-Shen Yen, Bey-Jing Yang and Feipei La, describes an Indoor Location System, using Beacon technology, applied to a Hospital Building. This ILS is divided into three main parts:

- **Patients' Mobile App side:** The patients need to install a mobile App and after that, they will be able to be located, through the implemented beacons that cover all over the hospital building. Depending on the patient location, the intersected beacon sends data to the mobile App which sends it directly to the server that will convert it to a Location;
- **Server Side:** This server is like a middle tier that makes the connection between the two users side: Patients and Medical Staffs. This side holds a that Database that storages all the patient's data and all the information from the location of the beacon as well. When received the location data from the App, does a search in its database looking for a beacon that matches the data sent from the App and returns a location. Having the location, directly send it to Medical Staffs Devices;
- **Medical Staffs Devices side:** Besides showing all the patients data (like condition, name, age, etc.) it also shows the patients current location through the hospital map.

Having this ILS implemented, certainly allows the doctors, and probably security staff as well, to always know where the patients are.

This example of ILS, applied with Beacons Technology, is very similar to what it intended to do in this project but holds a difference – only shows the position/location of the user. In our project, instead of having just a current location of the user, we have the destination, the current location and the calculated shortest path between them. The positive way of using Beacon technology is that is always possible to receive data from the beacons (if building holds beacon coverage) which allows the implementation of an ILS with “live” and synchronized information.

Chapter 3 *Find Me!* Conceptual Modules

Based on the defined objectives in section 1.2, the primary research idea in this project is to use a simplified approach where the location is based on beacons location and defined range. The location indeed can be combined with map information to give a more accurate indoor position. Also, BIM models can allow 3D views, which can be helpful information.

3.1 **Find Me! Overview**

A mobile application, named *Find Me!* App, was developed to integrate beacons and BIM models in order to show the user location and the path that he/she needs to do to the destination, through a smart device. The most important and crucial parts of *Find Me!* App are the user's current location and the inserted user's destination. Having calculated these two locations, a Path Finding Algorithm (A* Search Algorithm Type [22]) calculates the shortest path between these two points and draw it on a map with the objective of guiding the user to his destination. When the user intersects another beacon region, the drawn path is updated with a new user's current location.

Giving a brief description of what is intended to do in this App, when the user runs the *Find Me!* App, it will automatically identify the user's location. This background process will be implemented with Beacons Technology: when the user is inside of a beacon region will be able to collect the data from the beacon. The received data will be translated as location and will show it on a map as a current location of the user. Having the user's current location, the user has the option to insert the destination room and based on that, the mobile App calculates the shortest way between the current location and the destination, using a Path Finding Algorithm. The calculated way is shown on the map and is updated step-by-step (when intersected a new beacon) to increase the usability of App-User until he reaches the intended room. If the user chooses not to follow the given path and claim another option, the App will recalculate another way to the chosen destination. This mobile App also holds some extra features like:

- **User Locomotion Option:** Depending on user condition, it can be chosen, when the current location is on a different floor than the destination, the option of going by elevator or by stairs (default option);

- **Intersect Orientation Photos:** after calculating the shortest way between the current location and destination, it is going to be available, on the *Find Me!* screen, some photos of that path that the user needs to do to reach the destination.

This current work aims to help people with the complexity of the layout of many buildings problems/doubts. Implementing an ILS with input from Building Information Models complemented with BLE beacons can satisfy the user's needs. The BIM model entity will be responsible for generating rooms, beacon location data and map floors to the mobile App that was developed – *Find Me!* App. Beacons are going to be configured with specific data, and placed, based on geometrical concepts, to provide a current live location to the user, through the mobile App.

This project will have three kinds of users in order to make it functional. So before beginning the application description, we need to describe each of them to have a brief knowledge, in order to understand what comes next. Each user roll holds an outstanding job that is dependent on the other user's type to implement a functional ILS.

- **End Users:** any person who has installed the Find Me! App, and to do it, just need to have a smartphone with OS Android to launch the mobile application;
- **System and Database Administrators:** this kind of users are responsible for generating Maps and input data to the mobile App, like rooms and beacon location data. They also need to configure, for the first time, each beacon and physically implement each of them. These users will build or reuse a 3D model of the building, in our test case – ISCTE-IUL building. Having the 3D model done, it is possible to aggregate attributes to all kind of components that are present in the model. In our solution, it will be added attributes to entities as follows:
 - **Rooms:** name of the room, Building Number, Floor Number, and Location Room coordinates;
 - **Stairs:** location coordinates, building, and floor;
 - **Elevator:** location coordinates, building, and floor;
 - **Beacons Location:** MAC address, *IBeacon* protocol specification (UUID, Major, and Minor), Location coordinates, the radius of Beacon Region (broadcasting power in dBm unity), nearer stairs and nearer elevator;
 - **Paths:** which paths are walkable or not and which are permitted for some kind of users' type. All the aggregate attributes can be generated from the BIM model, as well as maps. These will be used in our project to generate the input to the Back End Mobile application;

- **Developers:** develop and do the maintenance of the mobile App – Find Me! code.

The information referred above about each user role can be observed in Figure 3-1.

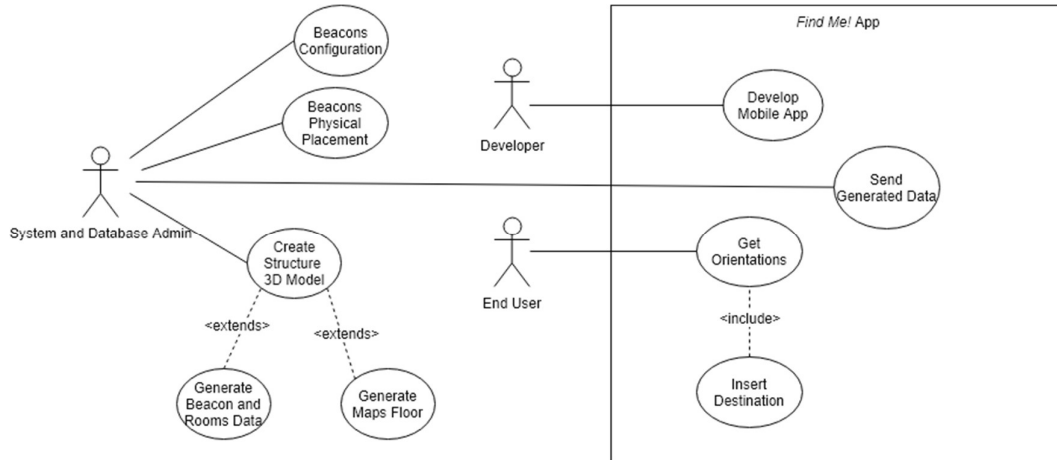


Figure 3-1 - Find Me! use cases diagram.

3.1.1 User Roles

The proposed ILS has the following users type: End User, developers and System and Database Administrators. In this section, we describe the role of each user.

End User

This ILS concept is intended for users that have a smart device, such as tablets or smartphones. The smart device needs to have an Android Operating System (OS) with Android 4.3 (JELLY BEAN MR2) as a minimum software development kit (SDK) target to install Find Me! App. This mobile App is developed for outsider and insider people of the entity (in our case of implementation example – ISCTE-IUL), so there is no *Log In* or *Sign In* need.

The **Get Orientations** action (see Figure 3-1) consists in using the developed mobile App – *Find Me!*, so the next content is going to describe the features of the mobile App. When the user runs the *Find Me!*, it scans for the closest beacon and after that, the user has two options to insert the destination (**Insert Destination** action, see Figure 3-1), as we can observe in Figure 3-2.



Figure 3-2 - Destination insertion mode options.

Choosing *FIND ROOM* mode, the user must write the destination as is implicit in Figure 3-3.

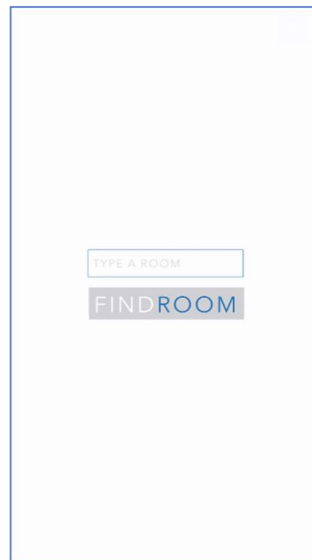


Figure 3-3 - *FIND ROOM* option.

Otherwise, if the user chooses the *FINDFAST* option, is able to choose quick spots like closest WC, food (canteens and bars), ATM, academic services and security station. As can be seen in Figure 3-4.

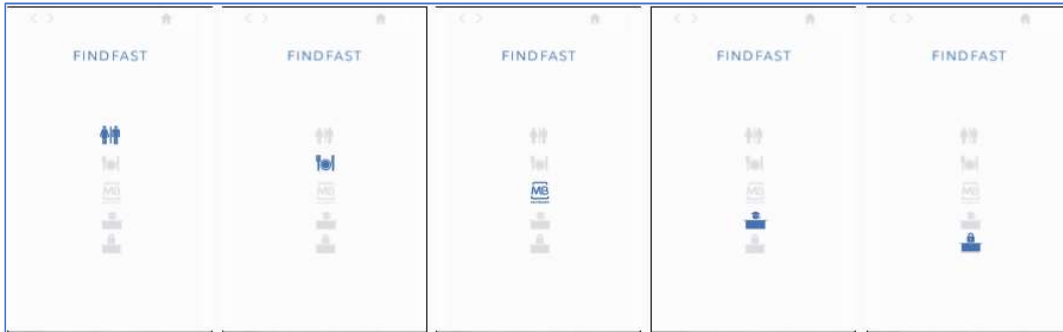


Figure 3-4 - FIND FAST options.

After having the two locations, current location, and destination, it will be shown to the user, the Orientations View, Figure 3-5 is an example of an experiment.

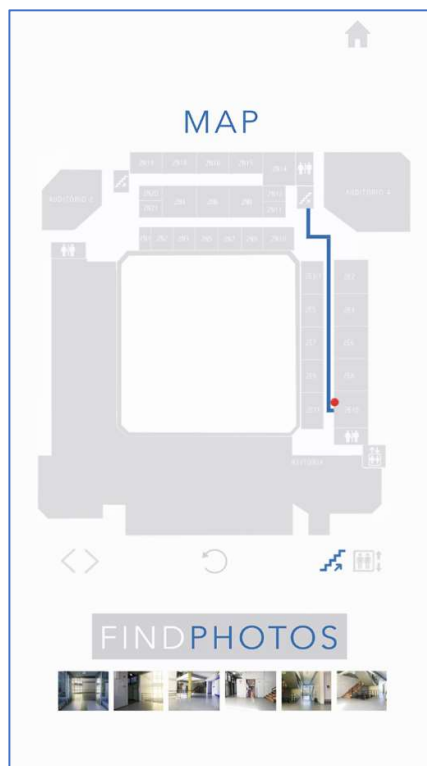







Figure 3-5 - Orientation View example.

In this activity the user is able to use the next features:

- **Path** : The path that the user needs to do;
- **Current Location** : To identify where the user is;
- **Destination** : To identify where is the user's inserted destination;
- **Rotate** : To rotate the map, in a way to help the user to get a better perspective of the map floor. Each time the user presses this button, the map will rotate 45 degrees;

- **Stairs or elevator** : When the destination is located on a different floor, or even in a different building that needs to use elevator or stairs, these two buttons are going to be shown on this screen. By default, the Find Me! redirects automatically to the closest stairs. If the user wants to go by elevator, just need to press the elevator button, and it shows a newly calculated route to the closest elevator as we can see in Figure 3-6;

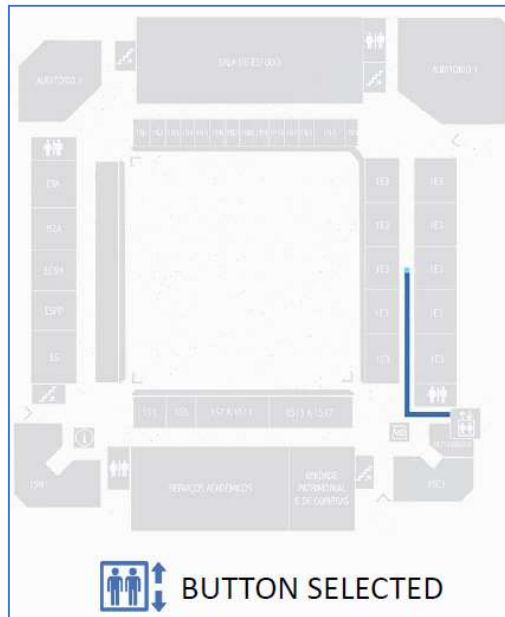





Figure 3-6 - Elevator button selected.

- **Home** : when selected, returns to the insertion mode activity (see Figure 3-2), with the objective of changing the room destination;
- **Orientation Photos** : With this feature, it is possible to check some important photos of the path that the user needs to do to reach the destination. In this way, the user can confirm that is on the right way. Pressing this button changes the layout to get the same photos but with the ability to zoom the pressed photo as we can verify in Figure 3-7. Pressing  button will redirect to Orientations View screen again;

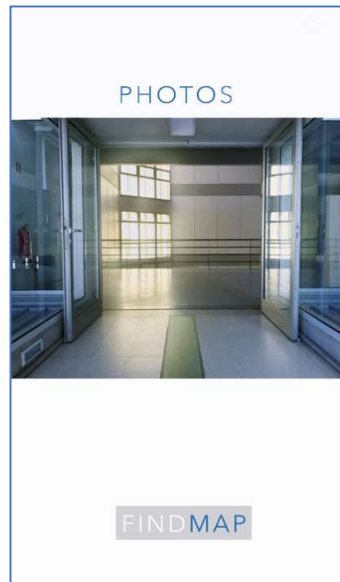


Figure 3-7 - FIND PHOTOS view

- Next/Previous < > : These buttons only appear when the current location floor is different from the destination floor. Clicking on the next button, the user can check where is the inserted destination on the corresponding floor. Clicking previous does the opposite, returns to current location floor map as the Figure 3-8 shows.

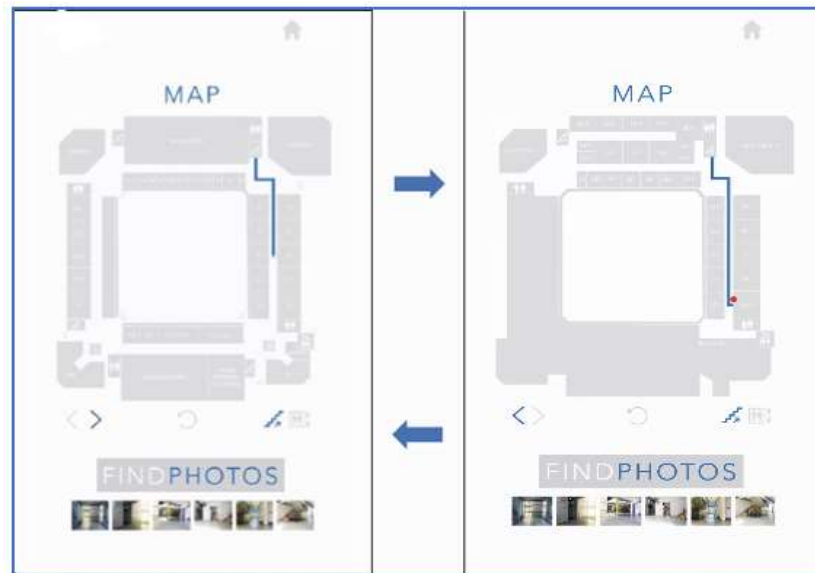


Figure 3-8 – Screenshot example of Current Floor (left) and Destination Floor (right).

Developer

This user has the role of developing the mobile App (**Develop Mobile App** action, see Figure 3-1) that is going to be used by the End Users. The App is named as Find Me!,

and was developed for Android operating system as a target. All the internal entities present in Front End and Back End Mobile App modules were implemented by this user role. When there is a need to fix a bug or add new features, this user type is going to do that kind of work.

System and Database Administrators

This user type holds an important role in this project: is responsible for implementing the external module of this project: Beacon Installation Process (see Figure 3-9).

In the BIM model entity, **Create Structure 3D model** action (see Figure 3-1), a 3D model of the building is generated, and attributes are aggregated to each room, beacon, stairs, and elevators (or more objects if needed). Having added all the required attributes, it is possible to extract that aggregated data in the form of tables and generate map floors (as .PNG files) to the Find Me! App. This process is represented as **Generate Beacon and Rooms Data** action and **Generate Maps Floor** action in Figure 3-1. Every time that a new element is added to the building (i.e. a room number changes, a new door is inserted) this user needs to add it in BIM model (and its attributes as well) and generate the data tables and PNG files. In other words, this user's role also includes the maintenance of this entity. This part is explained with more details in section 4.1.3.

To implement the Beacon entity, this user needs to configure and place the beacons (see section 0 to see how it is done) which are the represented **Beacons Configuration** and **Beacons Physical Placement** action in Figure 3-1. After the implementation of this module, this user is also responsible for the maintenance of each placed beacon.

3.2 System Architecture

In this section, the system architecture of this project is described. The primary objective of this project is to develop a mobile application that will provide on-time orientations in order to satisfy and increase orientations quality to the user inside of a complex building like hospitals, museums, airports or faculties.

The Find Me! project can be described as three main modules, represented in Figure 3-9, where is identified the main system modules:

- **Beacon Installation Process:** This process is used as an initial installation process, where we try to find the number of beacons to use into the building, taking into account building information extracted from the BIM model platform.

- **Front-end Mobile App:** This is the user interface (UI) side, in this module the user is able to observe dynamic orientations elements like: Map with the optimized path to reach the destination; Crucial Orientation Photos to confirm if the user is in the right way; Destination insertion; Options to help the user to get the orientations more understandable.
- **Back-end Mobile App:** This module holds a Local Database that stores the beacons and the rooms data, this kind of data is populated from .CSV files and it is stored locally when the user installs *Find Me! App*. A Beacon Manager that scans for the closest beacons and manages the data that comes from each beacon that is intersected by the smart device. A Map Manager, as the name indicates, manages which stored map(s) floor is going to be returned to the user interface. The map floors, generated by BIM model, are also stored locally when *Find Me!* the App is installed on the user's smartphone. Also has a Path Finding algorithm that calculates the shortest path between two locations.

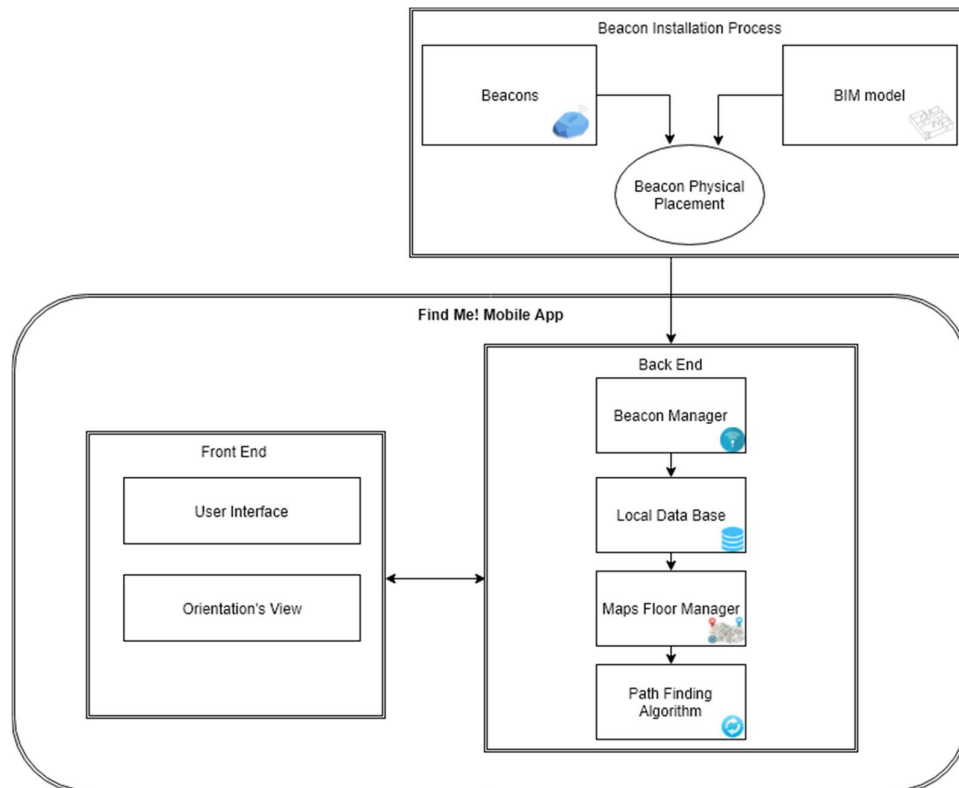


Figure 3-9 - *Find Me!* architecture.

3.2.1 Beacon Installation Process

In this section, we describe the Beacon Installation Process module which includes Beacons, BIM model entities, and the Beacon Physical Placement process.

Beacons

These BLE devices are responsible for sending data that is going to be after received by the **End User** smartphone (as a BLE signals receptor) that is going to be after converted as user current location by the Find Me! App. To validate the source of each BLE signal, each beacon needs to be configured and placed separately by **System and Database Administrators** in order to distinguish each beacon separately. As long as the **End User** smartphone receives a different BLE signal when walking through the building, the user current location is updated in *Find Me!* App.

Beacon Physical Placement

The role of this process is to get the quantity of beacons needed, the position where each one is going to be placed (in each floor) and the broadcasting power configuration needed in order to avoid BLE signals overlap and achieve a good ILS performance. If there is BLE signals overlap the correct user location cannot be assured. An external mobile App was developed to test the beacons Behaviour inside of a building, named *NearestBeacons* App (see section 0 – NearestBeacons App).

In BIM Model entity, it is possible to identify all the exits/entrances, halls/path intersections, stairs and elevators, and corridors centre. With that information, it is possible to apply Beacon Placement rules and Beacons Distance Rules (see section 0 – Beacon Location Rules) to have a successful Beacon Installation Placement process.

BIM Model

This is a 3D model of the building that includes all the elements that are present in the building. In this model, per each walkable and not walkable path, rooms, elevators, stairs and beacons positions, different attributes are set, depending on each type of the object. Having all this data set, it is possible to export all the object locations, attributes and interrelationships in alphanumeric format and also each map floor as PNG files. This exported data is going to be used in Find Me! App as input data.

A proper API is used to extract data from available BIM models. When selecting the building, as a parameter, and the information that we want to extract (Beacons

positions, rooms, paths, and map floors), this API exports the beacons and rooms as .CSV files and the map floors as PNG files.

3.2.2 Mobile App Back End

In this section we discuss all the implemented background logic, developed in Android Studio. The current location, destination, returned map floor and the shortest path, are calculated in this module and returned to the **Front End Mobile App** module.

Beacon Manager

It holds the objective of managing all the beacons that are intersected by the receptor. This entity calls an Estimote API that organizes the intersected beacons by RSSI from the highest value to the lowest, for other words, order the signals from the closest beacon to the furthest. It also extracts the data that comes from each BLE signal that is sent from the closest beacon. Having this feature from the Estimote API, there is no need to develop a handler for the intersection BLE signals.

The intersected beacon data is after sent, directly, to the Local Database entity to run a query and convert it, as the user's current location.

Local Database

This entity receives the Rooms and Beacons .CSV files from the **BIM Model** entity and locally stores each file content in the corresponding table – Room and Location respectively. This storage process, only happens when the user runs the mobile App for the first time.

The Room table holds all the required data to identify a room as a destination, and Location table is where all the beacon data/configuration values are stored in order to give the current location of the user. Receiving an inserted destination string from the user, or beacon data, it is possible to run a query and return a destination location, or a current user location in case of beacon data.

Maps Manager

The maps floor, that come from **BIM Model** entity (the .PNG files), are stored locally on the smartphone memory. The reason to have it locally stored, is to avoid the internet signal dependency to this project. The Maps Manager has the goal of deciding which maps floor is going to be returned to the Path Finding Algorithm Entity and which map is going to be after showed to the **End User** on the smartphone. This decision is based on the current

location and destination of the user. Having decided which map floor needs to be returned, sends it, with the current location and destination as well, to the **Path Finding Algorithm** entity.

Path Finding Algorithm (PFA)

This is a standard implementation of A* algorithm, that receives, as input, the current user location, the user destination and the map floor that sent by Maps Manager entity and calculates the shortest path between the two given locations. After having calculated it, the shortest path is returned to **Orientation's View** entity.

3.2.3 Mobile App Front End

This module presents all the orientation elements that are shown to the user, through the smartphone screen. It was also developed in Android Studio. It's here where it's possible to see (through the smartphone screen) the orientations that the user needs to reach the destination - the map with the current location, destination and the shortest path that the. In addition, the orientation photos and the activity where the user can choose the destination.

Orientation's View

This entity holds the objective of rendering all the orientation elements to help the user to find out the inserted user destination. It receives, from **Maps Manager**, the map floor, the shortest path from **PFA** entity and renders this both input parameters in a single element – an *ImageView*. With the given shortest path, this entity can also calculate the orientations photos - the FINDPHOTOS elements. The FINDPHOTOS shows to the user the spots that he/she needs to pass/go through to reach the destination.

User Interface

A simple entity, asks to insert the user's destination, or just select a FINDFAST destination (see section 4.2.2 – User Interface, to learn what is FINDFAST option). This information is after sent to **Local Database** entity to be converted as a location.

Chapter 4 *Find Me!* Implementation

In this section, it is going to be described how both entity types (external and internal) of the *Find Me!* App were implemented and what is the dependency of each one, and also their role in this developed ILS.

4.1 Beacon Installation Process

This section explains how the Beacon Installation Process (Beacons and BIM entities, and Beacons Physical Placement process) are implemented and how they provide, to *Find Me!* App, information/data to support user localization.

4.1.1 Beacons

To understand how the beacon works it is essential to explain its characteristics.

BLE signal parametrization

- **Broadcasting Power and Range:** Broadcasting Power is the power with which beacon broadcasts its signal. The range is described as the area where the BLE signal can be intersected/received by other smart devices. Broadcasting Power directly impacts the signal range. High power values mean that the range is going to be bigger/longer. The Broadcasting Power can be set from -40 dBm (minimum) to +4 dBm (maximum) – corresponding to a minimum range of 2 meters and maximum range is 70 meters, without obstacles between the Beacon and the receiver;
- **Advertising Interval:** The beacon's transmission packets can be configured in a restrict interval, this interval is the time when the beacon is "sleeping", which means how long the beacon will be freeze until sending another IBeacon packet. For example, if the interval configured is 100ms, it means that the beacon will broadcast its signal once every 100ms (or 10 times per second). It can be set from 100 ms to 2000 ms. Choosing the interval can affect the battery life of the equipment: when the interval is low, more packets are sent, which means that the battery life will be shorter.

Beacon Identification Parametrization (using the iBeacon Protocol)

The iBeacon protocol [23] was developed by Apple in 2014. This protocol enables the configuration of the data that is going to be sent in each BLE signal. In each iBeacon packet it is possible to configure the following fields:

- **UUID:** “16 bytes, usually represented as a string, e.g., “B9407F30-F5F8-466E-AFF9-25556B57FE6D”;
- **Major number:** 2 bytes, or an “unsigned short”, i.e., a number from 1 to 65,535;
- **Minor number:** 2 bytes, same as Major.

This protocol is suitable for this project because, in each BLE packet, three fields are sent so it is possible to have more beacon data combinations and also be more specific when identifying the beacons signal origin.

BLE Signal Characteristics

The Received Signal Strength Indicator (RSSI), is the strength of beacon’s signal as received on the smart device. This is related to distance and Broadcast Power, i.e., the strength depends on the distance and broadcasting power values. Considering the maximum Broadcasting Power, +4 dBm, the Received Signal Strength Indicator range from approximately -26 (close distance, few meters) to -100 (40-50 meters). The RSSI can be used to estimate the distance between the device and the beacon. Table 4-1 shows the approximate values of the distance, in meters, corresponding to broadcasting power values in dBm unity.

Table 4-1 – Broadcasting Power values converted to Maximum Range in meters.[24]

| Broadcasting Power (dBm) | Maximum Range (meters) |
|--------------------------|------------------------|
| -40 | 2 |
| -20 | 3,5 |
| -16 | 7 |
| -12 | 15 |
| -8 | 30 |
| -4 | 30 |
| 0 | 50 |
| 4 | 70 |

4.1.2 Beacon Physical Placement

Before the beacons configuration, there is a need to evaluate the building(s), as a structure, in order to get the number of beacons that we need and where these are going to be placed. This process holds the objective of getting a good correlation between a good ILS performance and a low beacons budget.

Beacon Propagation tests

These tests investigate the BLE signal propagation in concrete situations of a building with traditional reinforced concrete slabs, beams, columns and concrete, masonry and light-construction walls, as well as architectural features such as mezzanines, stairs, etc.

The main questions being investigated in this section are:

- is there BLE signal overlap in the same floor?
- is there BLE signal overlap between floors?
- is there BLE signal overlap in long corridors?
- is there BLE signal overlap between floors with mezzanines?

These tests allow us to make conclusions to achieve good location performance from the BLE signals returned by the placed beacons. To perform the tests a mobile application was developed, as described in the following section.

NearestBeacons App

A mobile App called *NearestBeacons* was developed by the author based on the Estimote API [25]. This auxiliary application has the objective of evaluating the BLE signal, depending on the position of the user, as a monitor of BLE signals intersection. This App, when scanning and detecting one or more BLE signals, print the following fields of each signal **order by signal strength (RSSI)** on the smartphone screen:

- **Major and Minor:** to identify which beacon we are evaluating;
- **Measured Power:** “indicates what is the expected RSSI at a distance of 1 meter to the beacon.” [26];
- **RSSI:** Signal strength value that depends on distance and broadcasting power [26].

The string format that is printed on the screen, for each intersected BLE signal, follows the next nomenclature: [incremental intersected BLE signal counter value] + “ – Major:” + [Major value] + “[Minor:” + [Minor value] + “[MeasuredPower:” + [Measured Power value] + “dbm” + “ |Rssi:” + [RSSI value] + “***”. Figure 4-1 shows an example

of NearestBeacons App showing two intersected beacons ordered by RSSI value, with the corresponding fields referred above.

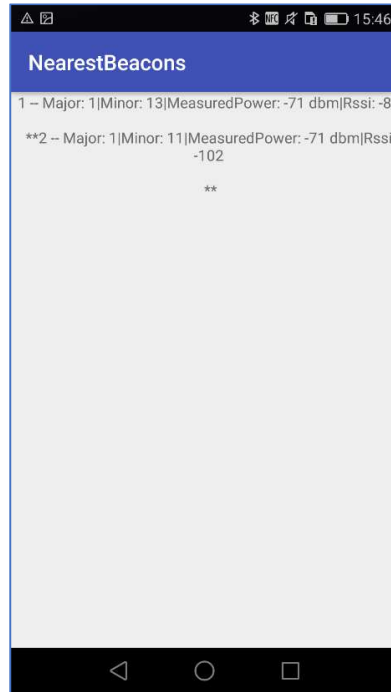


Figure 4-1 - NearestBeacons scan values example.

BLE Signal Propagation Test Results

In this section, we evaluate the beacons behaviour in an indoor environment. With this BLE signal propagations tests, we expect to make conclusions about BLE signals overlap situations and which external factors act as, and offer, BLE resistance. Figure 4-2 shows where each beacon was placed over the floors of ISCTE-IUL's *Edificio 1* in order to evaluate different beacons behaviour situations.

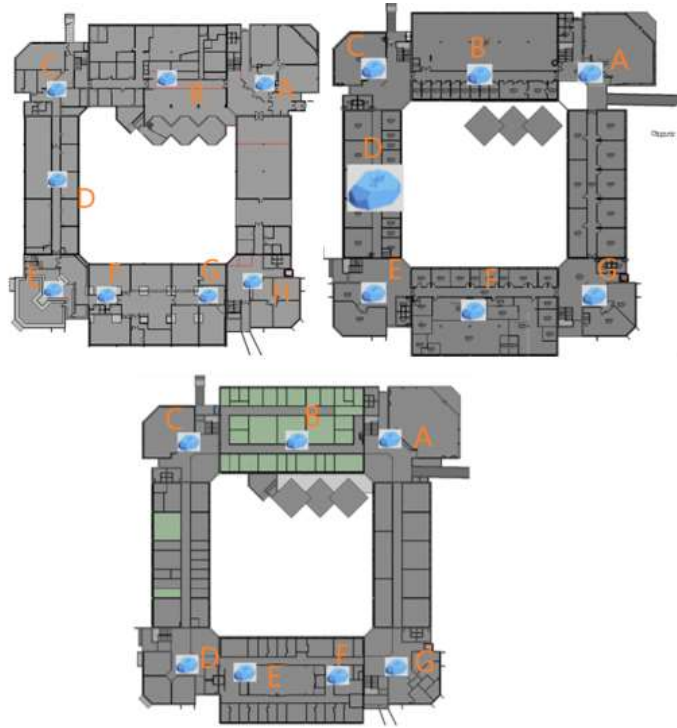


Figure 4-2 - Floor no 0, 1 and 2 (left to right) with placed beacons.

In this first approach, we physically placed 22 beacons and every beacon was configured with a broadcast power of -16 dBm, which means that holds approximate of 7 meters of radius coverage. The beacons were placed on the ceiling of each floor. Figure 4-3 shows a beacon placement practical example.



Figure 4-3 – Beacon Placement practical example.

Case 1: External factors – Slams density

This case holds the objective of getting conclusions about external factors as density of the slam between floors. In Figure 4-4, there is a blue rectangle that identifies the zone that we are focusing in this test case.



Figure 4-4 - Test case 1 layout. Floor 0 (left), floor 1 (right).

In this test case, we evaluate the BLE signal of ED1P0D (Major: 1 and Minor: 13) and ED1P1D (Major: 1 and Minor: 11) that are placed on the ceiling of floor 0 and floor 1 respectively. The signal strength values, taken from the *nearest beacon* App, were taken in 2 positions one from each floor (0 and 1) in a vertical line above each beacon.

Figure 4-5 shows where the user was, when taking the values (left) and also shows a *NearestBeacons* App screenshot(right) on floor 0. Observing these values, it is possible to confirm that the density of slam between floors does not block the BLE signal because we are getting both beacons BLE signals. The API used in the *NearestBeacons* App is the same used in *Find Me!*, so it organizes the signal from the highest RSSI values to the lowest, in this case, ED1P0D comes on the top of the list, and the next one is ED1P1D which is what is intended.

With these facts, it is possible to conclude that there is a need, if the density between floors has no effect on BLE signal, to have one beacon for each floor position to make sure that the *Find Me! the App* does not get BLE signal from the wrong floor, which implicit a wrong given current location of the user.

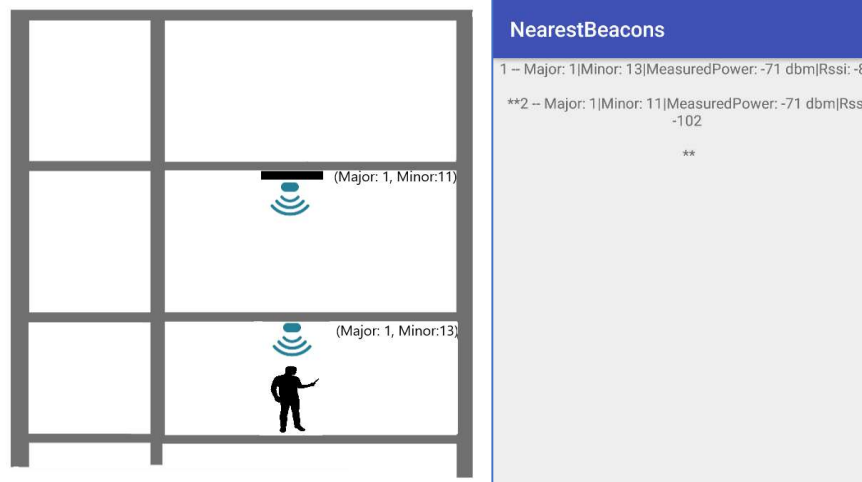


Figure 4-5 - Test Case 1 values, Floor 0, user position (left), NearestBeacons App results (right).

Case 2: External factors – Steel plate as a BLE blocker.

In this test case, we evaluate the steel as a BLE blocker. We placed the beacon ED1P1D (Major: 1 and Minor: 11) on the floor 1 ceiling with a steel plate (20x20cm) square with 5mm of thickness on the top. Figure 4-6 shows a practical example of the material mentioned in this Case 2.

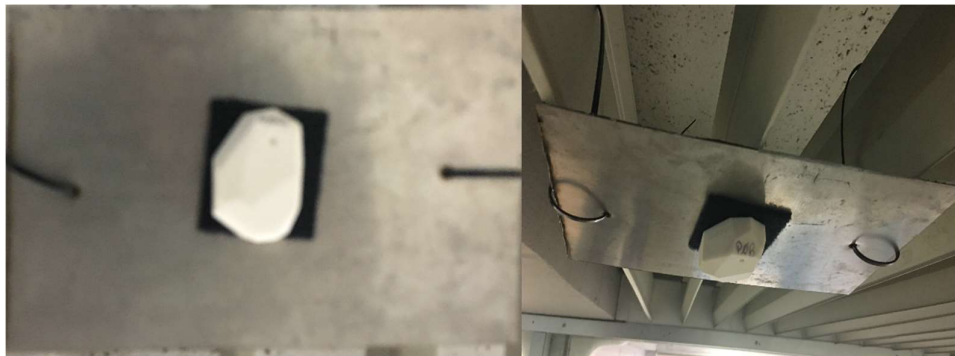


Figure 4-6 - Beacon Placement with a steel plate on the top example.

Figure 4-7 shows the zone that we are studying in blue color. Keep in mind that no beacon was placed on floor 2 for this test case.



Figure 4-7 - Test case 2 layout. Floor 1 (left), floor 2 (right).

Figure 4-8 shows where was the user, when taking the values (left), and *NearestBeacons* App screenshot (right). The position was on floor 2 vertically align with the beacon ED1P1D, in floor 1. As we can observe, *NearestBeacons* App didn't intersect any BLE signal, so it is possible to assume that the steel plate square worked successfully as a BLE blocker between floors.

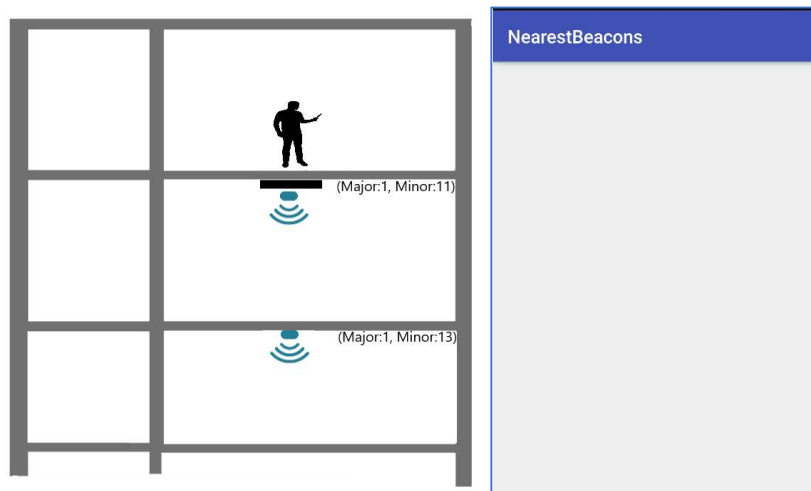


Figure 4-8 - Test Case 2 values, Floor 2, user position (left), *NearestBeacons* (right).

Case 3: Testing BLE overlaps with mezzanines.

In this test case, we evaluate what is the behaviour of 3 beacons (placed on the ceiling of each floor), when having mezzanines between floors. Each beacon is vertically aligned in the same position. Due to mezzanines, there is a hole that crosses the 3 floors, which can be checked in Figure 4-9 in yellow and the zone that we are studying in blue colour.



Figure 4-9 - Test case 3 layout. Floor 0 (left), floor 1 (middle) and floor 2 (right).

The beacons applied in this test were: ED1P0A (Major: 1, minor: 3), ED1P1A (Major: 1, minor: 1) and ED1P2A (Major: 1, minor: 5). The values presented in Figure 4-10 and Figure 4-11 were taken from floor 1 and 2 respectively and in the same position on both floors.

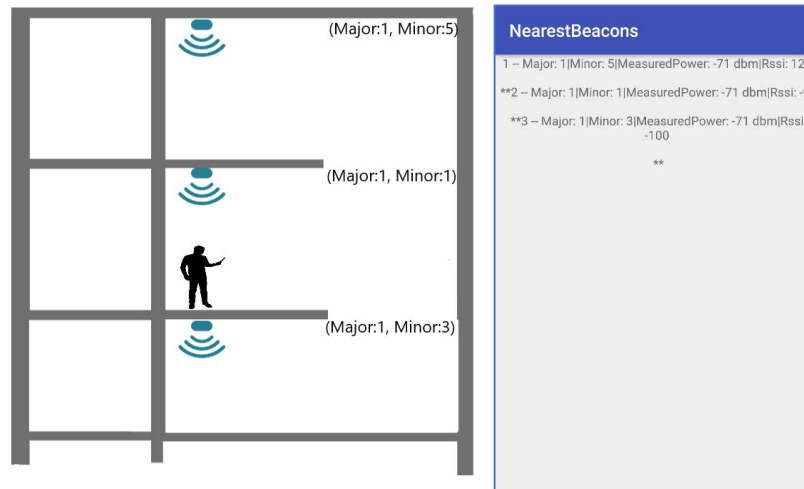


Figure 4-10 - Test Case 3 values, Floor 1, user position (left), NearestBeacons (right).

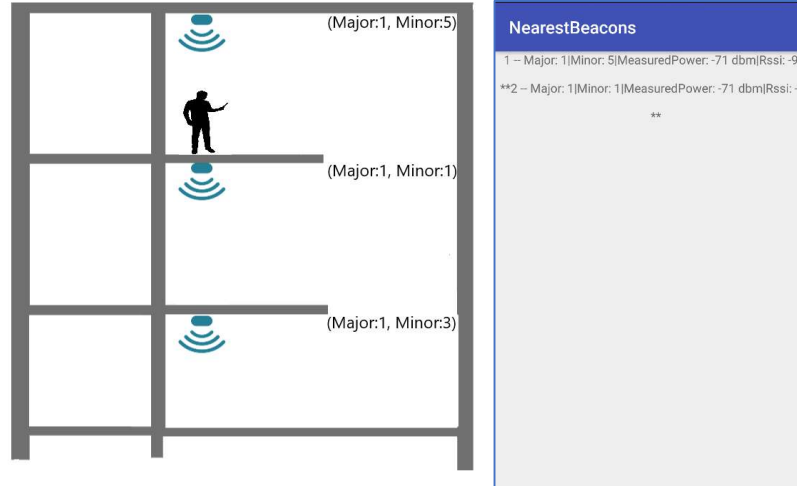


Figure 4-11 - Test Case 3 values, Floor 2, user position (left), NearestBeacons (right).

As we can observe in Figure 4-10, and because of the hole, there is a BLE overlap of signals between ED1P1A and ED1P2A. The *NearestBeacons* App printed the ED1P2A as the highest signal strength (RSSI) on floor 1, instead of the expected ED1P1A beacon signal. This situation means that it can be translated (from *Find Me!* App) as a wrong current location of the user, for other words, the mobile App would show that the user was on a wrong floor. To solve this problem, the best approach here was to set the 3 beacons with a broadcast power from -16 dBm to -20 dBm (see Table 4-1 to see the maximum range in meters). Figure 4-12 and Figure 4-13 show the values, taken in each floor, after this configuration in the same previous positions.



Figure 4-12 - Test Case 3 values after configuration changes, Floor 0(left) and Floor 1(right), NearestBeacons.

| NearestBeacons | |
|----------------|---|
| 1 | – Major: 1 Minor: 5 MeasuredPower: -76 dbm Rssi: -93 |
| **2 | – Major: 1 Minor: 1 MeasuredPower: -76 dbm Rssi: -100 |
| | ** |

Figure 4-13 - Test Case 3 values after configuration changes, Floor 2, NearestBeacons.

As we can confirm, this approach worked when setting the broadcast power to lower values. Another solution, to avoid the broadcast power decrease, could be a steel plate square on the of each beacon where we are in a situation of mezzanines.

Case 4: Testing BLE overlaps with 2 beacons on one floor and another one on a different floor.

In this case, we evaluate the behaviour of having two beacons in floor 2 and just one beacon on floor 1 in different positions. The following beacons were used in this test: ED1P1F (Major: 1, Minor: 18), ED1P2E (Major: 1, Minor: 20) and ED1P2F (Major: 1, Minor: 21). Figure 4-14, shows where the three beacons were placed. In red, the user position and in blue, the zone of this test case.

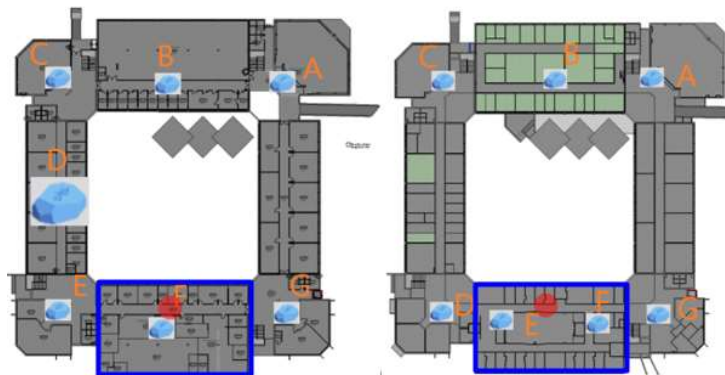


Figure 4-14 - Test case 4 layout. Floor 1 (left) and Floor 2 (right).

In Figure 4-15 and Figure 4-16, we can observe the user position and the returned values from the *NearestBeacons* App for each corresponding floor. In both floors values, it is possible to confirm, because there is no intersected BLE signal from the other floor beacon, that the concrete slab, dividing the floors, blocked the BLE signals. In floor 2, it is only possible to see the values from beacon ED1P2E because the position where the *NearestBeacons* App was taken, was a little bit closer to this beacon and because we

configured both beacons with a maximum of 10 meters of radius range, the ED1P2F beacon was not intersected. Therefore there wasn't BLE signal overlap.

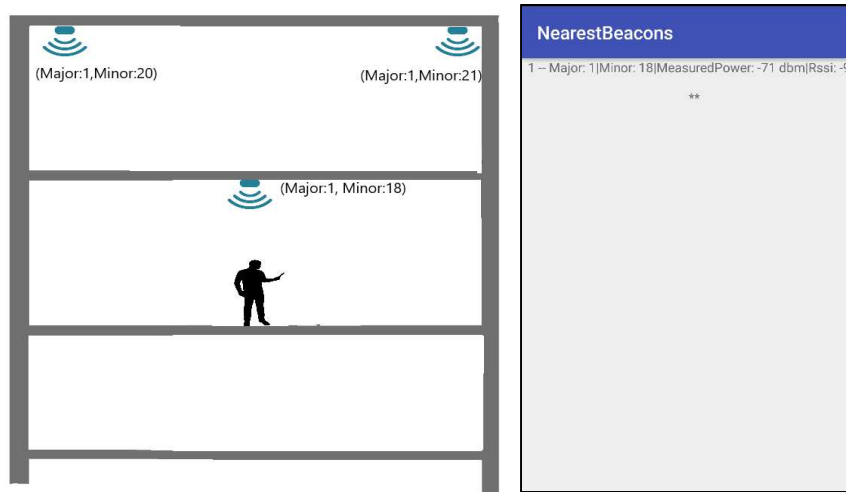


Figure 4-15 - Test Case 4 values, Floor 1, user position (left), NearestBeacons (right).

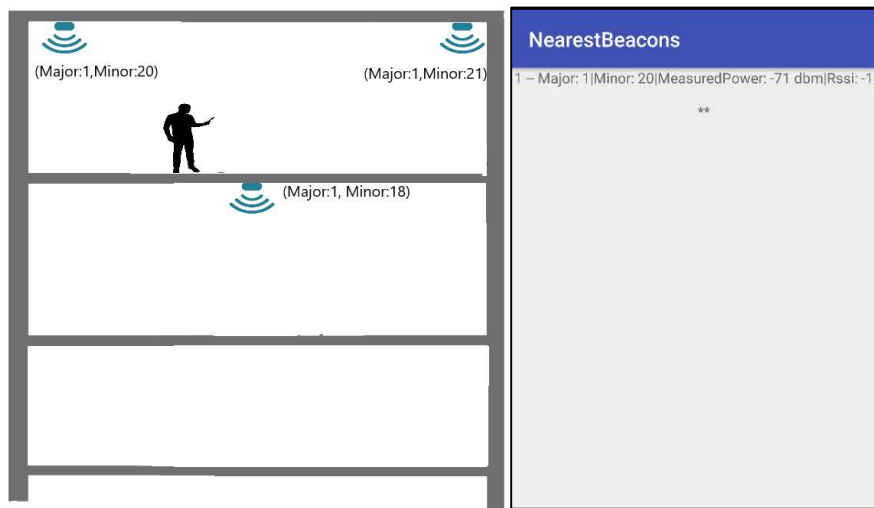


Figure 4-16 - Test Case 4 values, Floor 2, user position (left), NearestBeacons (right).

The four test cases referred above, and their conclusions were used in order to get the behaviour of the BLE signal from beacons in several situations/positions. With this kind of test cases, we can confirm that is possible to place the beacons, following Beacons Physical Placement rules (mentioned in section 0 – Beacon Location Rules), without having BLE signal overlaps.

Beacon Location Rules

The diagram in Figure 4-17 describes the rules that must be followed to get a good relationship between beacon quantity and location. It is a generic algorithm that can be applied to all complex buildings when having a BIM model already implemented. With

the BIM models we can extract the building data needed to this algorithm and with that data, we can minimize the number of needed beacons, which will have a positive impact on the system's budget. The first step of the diagram is to get information from the BIM model of the area of implementation of the ILS. For each building, we have the floor's description which includes the identification of entrances/exits, circulation areas, paths intersections, stairs and elevators, and corridors centre. This identification process ensures that no user can enter or leave the building without being intersected by a beacon. Having identified the beacons positions/locations a Beacons Placement Floor template (a map floor with beacons positions) is generated, and this algorithm skips to the next floor. Having iterated all floors from that building, it skips to the next building and repeats the process until there are no buildings left to evaluate. After that, the generated Beacons Placement Floor templates are analyzed to set the coverage of the radio (broadcasting power) for each one.

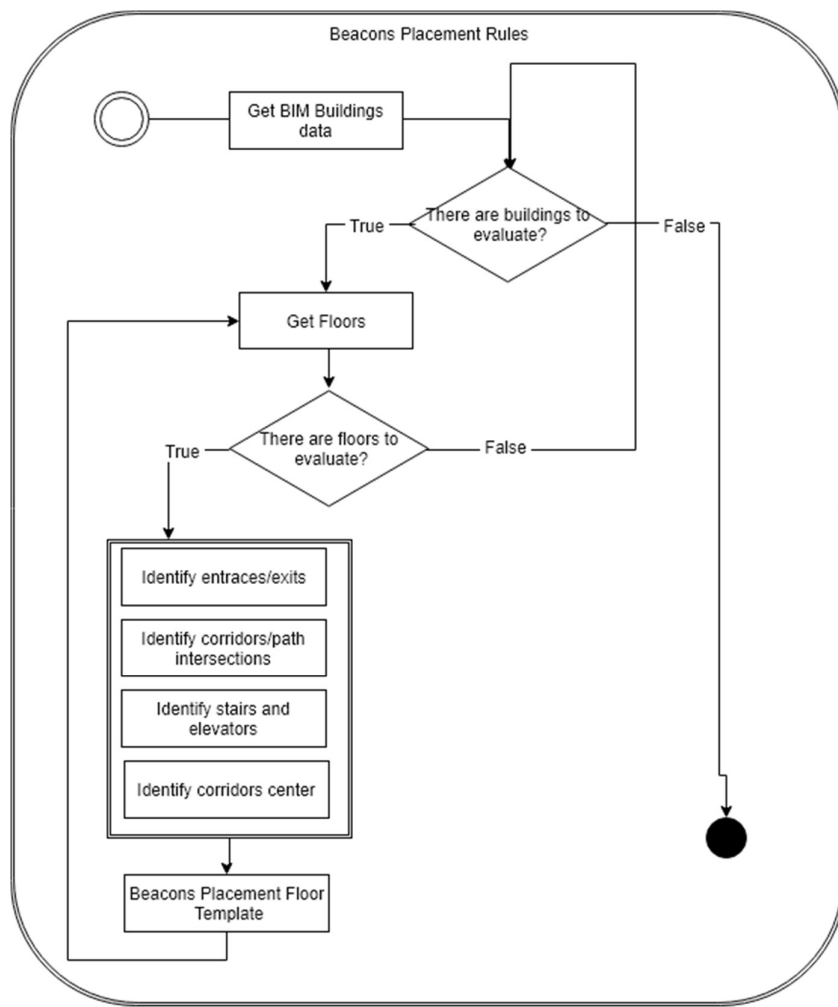


Figure 4-17 - Beacons Placement rules diagram.

After getting each Beacon Placement Floor template, it is time to set the broadcasting power (radio coverage) of each beacon depending on the distance between beacons. In Figure 4-18, we have a Beacons Distance rules that are applied to each beacon. In this rule, for each Beacons Placement Floor (generated by Beacon Placement rules), we pick a beacon and get the distance from its closest beacon neighbor. Having the distance value, we validate:

- **If the distance is less than 4 meters:** beacon broadcasting power is set to -40 dBm, which is 2 meters of radius, approximately;
- **If the distance is between 4 and 7 meters:** beacon broadcasting power is set to -20 dBm, which is 4 meters of radius, approximately;
- **If the distance is more than 7 meters:** beacon broadcasting power is set to -16 dBm, which is 7 meters of radius, approximately.

This process is repeated until there are no beacons to set the broadcasting power. The reason why we didn't implement more validation steps above 7 meters, it is because if we keep incrementing the broadcasting power, the beacon radius is going to be bigger and could possibly overlap the BLE signal from the beacons of the superior or inferior floor and can cause a wrong provided location.

After having performed the referred process above, the beacon position and radius information is included in the BIM model. With this approach, we expected to avoid the BLE signal overlap between beacons. Next topic is going to explain how to configure a beacon (including the broadcasting power).

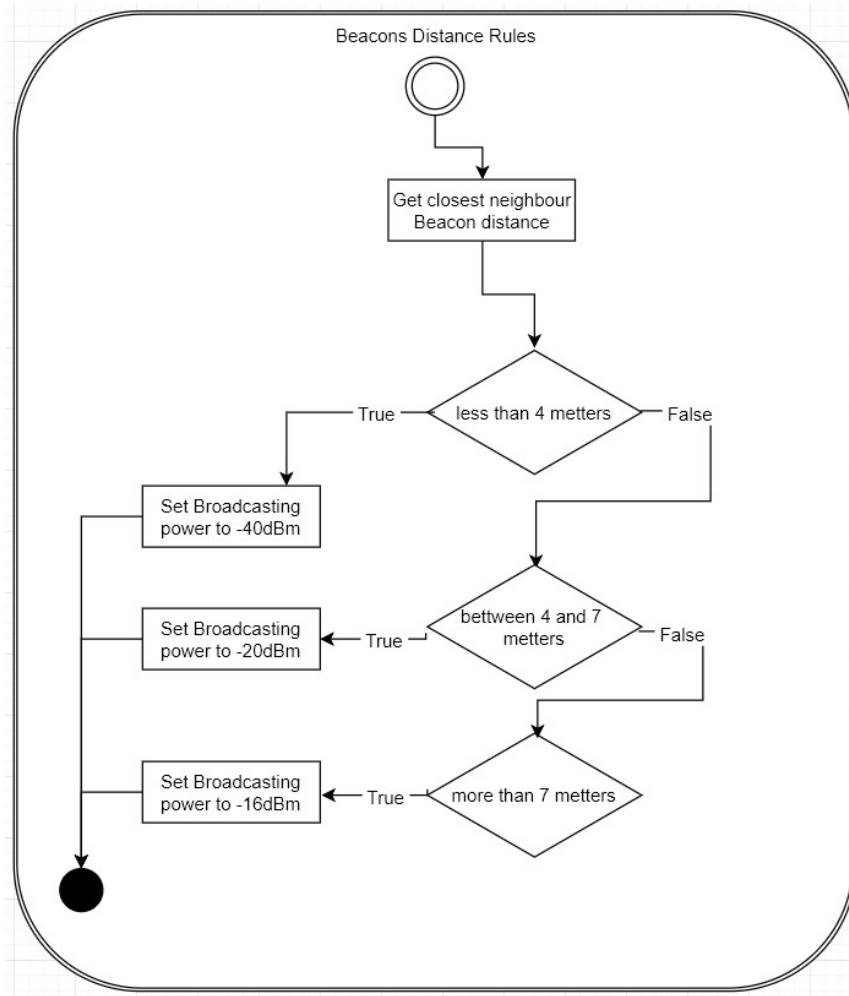


Figure 4-18 - Beacons Distance Rules diagram.

Beacon Configuration

As it is mentioned in section 3.1.1 – System and Database Administrators, one of their roles is to configure, place and manage the beacons entity on this project. Every beacon must be pre-configured to be after placed. To be able to do it, first, and only on this user role, there is a need to create an account on *Estimote* Cloud – example on Figure 4-19.

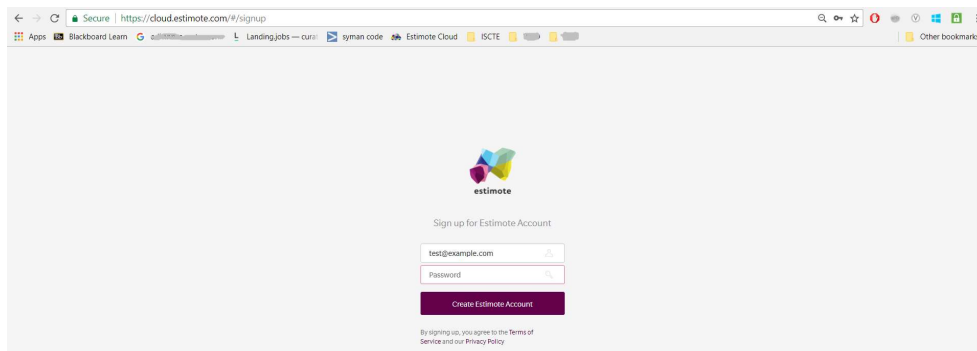


Figure 4-19- Creating Estimote Account

After having created an account, only for the first time, the administrator(s) needs to install the Estimote mobile App, switch on the Bluetooth and mobile data, and claim all the intended beacons to the previously created account. The smartphone works as a gateway between the cloud and the beacons, to update each newly transferred beacon to the own cloud. After having claimed all the beacons, it is time to configure Beacons Name: This field follows the ED + [Building number] + P + [floor number] + [Alphabet letter] nomenclature. Having set each beacon name field, it is easy to identify each beacon outside of the mobile application. An example of a beacon name configuration can be observed in Figure 4-20.

The screenshot shows the 'Edit Settings' window for a beacon. The left sidebar has 'Device' selected. The main content area is titled 'Name' and shows 'ED1P2B'. Below this are several settings: 'Geo Location' (dropdown), 'Tags' (dropdown), 'Access Mode' (dropdown, set to 'Deployed & Protected'), 'Smart Power Saving' (dropdown, set to 'Off'), 'Motion Only Broadcasting' (dropdown, set to 'Off'), 'Motion Only Broadcasting Delay (0 to 300 000 ms)' (input field, set to '0'), 'Flip to Sleep' (dropdown, set to 'Off'), 'Automatic Firmware Update' (dropdown, set to 'Off'), and 'Estimote Secure Monitoring' (dropdown, set to 'Off'). At the bottom right, there are 'Cancel' and 'Save Changes' buttons.

Figure 4-20 - Beacon Name configuration

Protocol and Broadcast Packets

In this stage, we enable the iBeacon protocol, the BLE signal parametrization, and the BLE Identification Parametrization (see section 4.1.1). In this project, all the beacons are configured with the same UUID to create a private beacons mesh. Having a private mesh prevents the addition of an external beacon by other unauthorized users. Major and minor fields are going to differentiate each beacon inside of that mesh.

On Broadcast Packets section, it is possible to configure the Advertising Interval (section 4.1.1 to see description) - this parameter can have consequences in beacons battery life if the interval value is low. The user is also able to configure the Broadcasting Power (section 4.1.1 to see description).

The Figure 4-21 shows how to enable IBeacon Protocol, configure UUID, Major and Minor and the Broadcast Packets.

The screenshot displays the 'Edit Settings' window for an IBeacon. The left sidebar lists various settings categories, with 'iBeacon' selected. The main area contains several configuration options:

- Enabled:** A dropdown menu set to 'On'.
- Proximity UUID:** A text field containing the value '7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF'.
- Major (1 to 65535):** A text input field with the value '1'.
- Minor (1 to 65535):** A text input field with the value '9'.
- Secure UUID (?):** A dropdown menu set to 'Off'.
- Custom Advertising Interval:** A dropdown menu set to 'On'.
- Interval (100 to 10000 ms):** A text input field with the value '300'.
- Broadcasting Power (dBm):** A dropdown menu set to '-40'.

At the bottom right, there are 'Cancel' and 'Save Changes' buttons.

Figure 4-21 - Configure Protocol and Broadcast Packets

After all the configuration above, the administrator needs to use the smartphone to run the *Estimote* mobile App and update the configuration changes in each beacon.

4.1.3 BIM Model

All building related data that provides the Find Me! App with physical context is extracted from the pilot building BIM model. An Autodesk Revit model containing the geometry of the pilot building was made available by ISCTE-IUL's facility management office. The model was upgraded to Revit version 2019[27] and a few adaptations were made to the geometry, namely correction and reparameterization of stairs and elevators and insertion of missing doors. Classrooms and office information was imported into the model from the university's academic management system using dedicated Dynamo scripts, and identification of all remaining areas, such as circulation, technical were manually added to the model. A Revit family was created for the beacons containing all their parameters, and beacons were individually inserted into the model, along with their alphanumeric information. A Dynamo routine that scans each door in the model, retrieves the identification of the room the door serves (opens to) and writes that id in the door "Mark" parameter was developed and applied to the model. This way navigation is facilitated since when a user wishes to navigate to a room he is led not to the centre of the

room but to one of its doors. These modelling tasks were performed by a trained BIM modeler in about two days, not counting with Dynamo routines development.

The next step is then to integrate beacons, doors, stairs, elevators locations and their interrelationships into the *Find Me!* App, as well as two kinds of floor plans: one for user visualization and other for internal App navigation. This analysis and export is performed by a set of Dynamo routines that analyze the model and generate two CSV files for the building and two 2D PNG files per floor.

A first Dynamo script extracts stair and elevator location (coordinates and building floor) and the floors each of them serves. Then two similar scripts, shown in Figure 4-22, extract beacon and door location and internal parameters and determines the stair and elevator closer to each beacon and door and generate the CSV files.

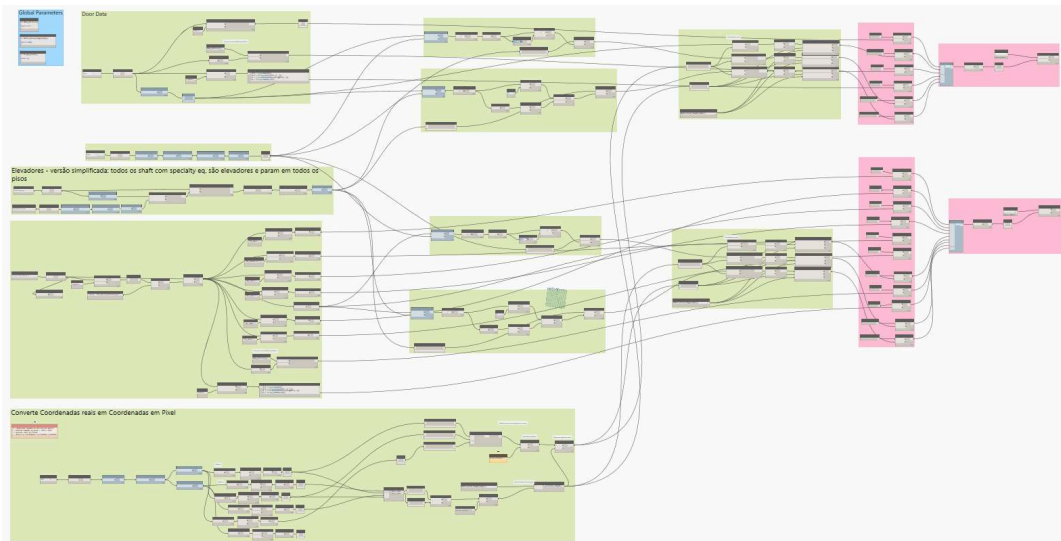


Figure 4-22 - Main Dynamo script for beacon and door/room tables production.

All coordinates are converted from meters to a 1000x1000 pixel grid. In the pilot building this corresponds to approximately 10 pixel/meter, which is a good compromise between location precision/image quality and file size. The origin of the grid is coordinated with the PNG files view of the floor plans.

The Rooms .CSV comes with the following fields:

- **Sala:** room's name;
- **Coordenadas_Porta:** room location;
- **Edificio:** building number;
- **Piso:** floor number;
- **Coordenadas_Escada_Proxima:** closest stairs location;
- **Coordenadas_Elevador_Proximo:** closest elevator location;

The Beacons .CSV comes with the following fields:

- **Name:** Beacon’s name;
- **UUID:** UUID value;
- **Major:** identifies the building number (in this case always 1);
- **Minor:** integer which identifies the beacon inside the building (starting at 1);
- **Coordenadas_Beacon:** Beacon’s location;
- **Edificio:** Building number;
- **Piso:** Floor number;
- **Coordenadas_Escada_Proxima:** closest stairs location;
- **Coordenadas_Elevador_Proximo:** closest elevator location.

Figure 4-23 displays examples of the 1000x1000 pixel 2D PNG files. All info is generated directly from the BIM model, including room identification. In the navigation plan, the red area indicates zones where navigation is allowed. The allowed areas correspond to “rooms” in the BIM model that are tagged with a “Circulation”, “Stair” or “Elevator” parameter (as opposed to “Classroom”, “Technical Area” or “Meal Area” and other tags).

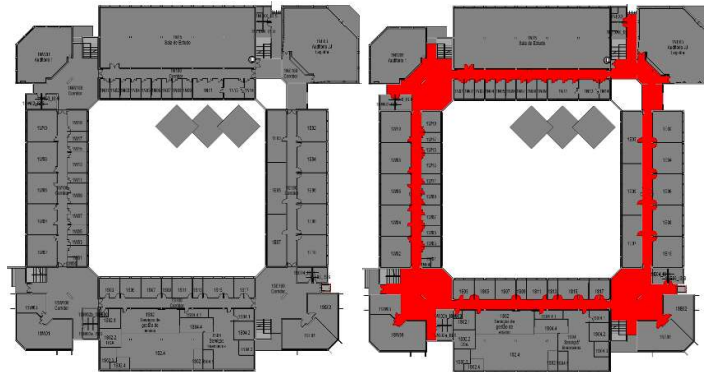


Figure 4-23 - Floor plans generated from the BIM model for user visualization (left); internal app pathfinding algorithm support with walkable areas in red (right).

4.2 Find Me! Mobile App

In this section, we describe how the Back End and Front End Mobile App entities were implemented in this developed ILS. Both sides, Back End and Front End were developed in Android Studio programming language.

4.2.1 Mobile App Back End

In this section, we explain how all the entities, from the Back End Mobile App module, mentioned in section 3.2.2, were implemented.

Beacon Manager

This entity is responsible, as the name says, to manage the beacons that the smartphone intersects, even if the user is walking or not. *Estimote* provides an SDK that offers an API to handle the BLE connection, so there is no need to process all the packets that are nearby the smartphone because this API fulfills the requirements of this entity.

To import this SDK, we need to add it as a dependency of the Android Studio project. In Gradle Scripts, into the *build.gradle*, we just need to add the dependency, as the Figure 4-24. Until now, version 1.4.0 is the most viable to use[28], so that is the one that we choose to work.



Figure 4-24 - Estimote dependency

After adding the dependency, the Android Studio compiler automatically synchronizes the gradle and the *Estimote* API is ready to use in our code.

To start monitoring the beacons, there is a need to call, every time the user runs the application, the *connect()* method to switch on the receiver of the smartphone for BLE packets.

Besides that, in this method it is possible to set filters to which kind of beacons we want to intersect, creating our own Beacon Region instance. In this developed ILS, we want to intersect beacons from the private mesh (see **section 0 – Beacon Configuration** to check what is this private mesh and the UUID value) that we created with the UUID value: 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF. It is intended to get all of the beacons in this mesh, so we set the third and fourth argument to null in order to be able to intersect beacons with different major and minor values. The first argument is just the name that we call to the created Beacon Region. Figure 4-25 shows the way to call this method properly.


```

beaconManager.connect(new BeaconManager.ServiceReadyCallback() {
    @Override
    public void onServiceReady() {
        beaconManager.startMonitoring(new BeaconRegion(
            "monitored region", UUID.fromString("7AE702E0-E1A7-
EEA9-E127-4C304EC7D4DF"),
            null, null));
    }
});|

```

Figure 4-25 – Estimote API, Connect method.

After having called the `connect()` method, now we are able to start receiving BLE packets in our mobile application. To do it, just need to call the method `BeaconMonitoringListener()`, this method can be triggered in two cases: When intersecting a beacon region and when we are leaving the intersected beacon region. The `onEnteredRegion` triggers when we receive more than one BLE packet (in terms of quantity). After that, it is assumed that we entered in a beacon region if we keep receiving BLE packets. In this method, we also have access to an intersected beacons list (ordered by the closest beacon to the farthest, the RSSI value). In each intersected beacon, we can extract the UUID, Major, and Minor values. In our case, we use that values and send it to the DB as a query, to know where the user’s current location is. This part will also be explained in **Local Database**, along with this section. If our receptor, the smartphone, stops receiving that kind of packets it will trigger the `onExitedRegion`, and it is assumed that we left the Beacon Region. All of what is mentioned above can be observed in Figure 4-26.

```

beaconManager.setMonitoringListener(new
BeaconManager.BeaconMonitoringListener() {
    @Override
    public void onEnteredRegion(BeaconRegion region, List<Beacon>
beacons) {

        //INTERSECTED A BEACON REGION

    }
    @Override
    public void onExitedRegion(BeaconRegion region) {
        //LEFT A BEACON REGION
    }
});

```

Figure 4-26 - Estimote API, Monitoring Beacons method

The Beacon Manager, when initialized, works like a thread that always looks for BLE packets. Since the moment that the user runs the App until he/she closes it, the Beacon Manager always works in the background in order to find beacons, this feature allows the refresh of the path “on time”, when intersecting a new beacon.

Local Database

To avoid internet issues, which could compromise the main objective of this project, it was decided to implement a local database instead of having an external database. The maps floor, rooms, and beacons data, that comes from the BIM Model entity, are stored locally. This storage process only happens when the user installs the mobile App.

As it is referred in section 4.1.3, after that BIM model entity builds a 3D model (in our implementation example: ISCTE-IUL, Edificio 1), it is possible to generate 2D maps floor and with that, it is possible to export the rooms and the beacons attributes, that are present in the 3D model building, as data. BIM Model generates this data as .CSV files with Rooms and Beacons information. The tables, in Local Database entity, that are populated with .CSV file data, holding the following structure:

Room Table

- **Id:** This field is unique and holds the objective of differentiating the rooms;
- **Name:** Name of the room;
- **Building:** Number of the building;
- **Floor:** Number of the Floor;
- **Location:** Coordinates of the location of the room on that specific floor.

Beacon Table

- **Id:** This field is unique and differentiates the beacons;
- **Building:** Number of the building;
- **Floor:** Number of the floor;
- **Major:** IBeacon values;
- **Minor:** IBeacon values;
- **Stairs:** Coordinates of the nearest stairs on that specific floor;
- **Elevator:** Coordinates of the nearest elevator on that specific floor;
- **Location:** Coordinates of the location of the beacon in that specific floor.

Figure 4-27 shows the structure fields of Room and Beacons, mentioned above, as tables.

| Room | | | Beacon | | |
|----------|------------|----|----------|------------|----|
| Id | Integer | PK | Id | Integer | PK |
| Name | Varchar(6) | | Building | Integer | |
| Building | Integer | | Floor | Integer | |
| Floor | Integer | | Major | Integer | |
| Location | varchar(8) | | Minor | Integer | |
| | | | Stairs | Varchar(6) | |
| | | | Elevator | Varchar(6) | |
| | | | Location | Varchar(6) | |

Figure 4-27 - Room and Beacon tables structure

An example of a Room .CSV file can be observed in Figure 4-28. The Id is an incremental field so there is no need to read that value from the Room .CSV file, it is going to be managed by the database side. The Name field is read from the first Room CSV column – Sala. Building and Floor field are going to be read from column 3 (Edificio) and 4 (Piso) CSV file respectively. Finally, Location field is read from Coordenadas_Porta column, besides this field comes with (xx, yy, zz) coordinates format, we are only reading and converting to (xx, yy) coordinates format to be after used in 2d map floors need. The Fields Coordenadas_Escada_Proxima (closest stairs location) and Coordenadas_Elevador_Proximo (closest elevator location) are not read as an input of this project, but we keep them for future needed features.

| |
|--|
| Sala,Coordenadas_Porta,Edificio,Piso,Coordenadas_Escada_Proxima,Coordenadas_Elevador_Proximo |
| 0NW07,(208;275;71),1,0,(265;156;87),(842;744;117) |
| 0NW08,(270;231;71),1,0,(265;156;87),(842;744;117) |
| 0NW06,(208;294;71),1,0,(265;156;87),(842;744;117) |
| 0NW04_IS,(173;269;71),1,0,(265;156;87),(842;744;117) |
| 0NW03_IS S,(139;269;71),1,0,(265;156;87),(842;744;117) |
| 0W09,(208;359;71),1,0,(404;343;68),(842;744;117) |
| 0W07,(208;409;71),1,0,(404;343;68),(842;744;117) |

Figure 4-28 - Room CSV data example.

Figure 4-29 shows an example of Beacon data that comes from the corresponding CSV file. The Id is an incremental field and it is going to be handled by the DB side. The Building and Floor field are going to be read from .CSV column 7 (Edificio) and 8 (Piso) respectively. Major and Minor fields are going to be populated from the .CSV columns 3 (Major) and 4 (Minor). Stairs and Elevator fields are loaded with the data from CSV columns 8 (Coordenadas_Escada_Proxima) and 9 (Coordenadas_Elevador_Proximo). The Location field is read from Coordenadas_Beacon column. Note that, the CSV columns 5, 8 and 9 comes with (xx, yy, zz) coordinates format but we are only reading and converting to (xx, yy). UUID CSV column is an unused part of this data, but we keep it to future features.

| Name,UUID,Mayor,Minor,Coordenadas_Beacon,Edificio,Piso,Coordenadas_Escada_Proxima,Coordenadas_Elevador_Proximo |
|--|
| ED1P2F,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,21,(213;237;169),1,2,(265;156;124),(842;744;117) |
| ED1P2G,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,22,(208;492;170),1,2,(133;713;124),(842;744;117) |
| ED1P2D,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,7,(212;764;168),1,2,(133;713;124),(842;744;117) |
| ED1P2A,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,5,(742;764;167),1,2,(690;845;124),(842;744;117) |
| ED1P2B,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,9,(747;468;167),1,2,(690;157;124),(842;744;117) |
| ED1P2C,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,10,(741;236;166),1,2,(690;157;124),(842;744;117) |
| ED1P2E,7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF,1,22,(477;208;169),1,2,(453;336;68),(842;744;117) |

Figure 4-29 - Beacon CSV data example.

Having the database populated with data (loaded from the .CSV files), this entity is ready to fulfil all the requests/query from the Find Me! App. An example of that request is when the smartphone intersects one beacon, the **Beacon Manager** entity extracts the major and minor values and sends a request (named `getCurrentLocation` query) to the DB to after returns the intersected beacon location, considering the major and minor as input. The same happens when the user inserts the room name, the Find Me! App sends a request (named `getDestination` query) with the room name as input to the DB in order to get the room location.

Maps Manager

As it was referred above in section 3.2.2, this entity is responsible for managing all the map floors (.PNG files) that need to be returned to the Path Finding Algorithm, according to current location and destination of the user. The Map Manager extracts, from the returned query (`getCurrentLocation` and `getDestination`) in **Local DataBase** entity, the building and the floor number of each Location. After that, a string is created with the extracted values, and the final format is going to be: “edificio” + [building number] + “piso” + [floor number] – example: edificio1piso1. Having created both strings (one for each location), this entity validates if both strings are equal. The Map Manager follows a rule to make sure that the returned map floor is the correct one. This rule is based in two important aspects:

- **Both strings hold the same value:** It means that the user is in the same building and floor than the destination, so the Map Manager just needs to return the map floor that corresponds to that particular string value;
- **Different values:** It means that the destination is on a different floor of the current Location of the user. Considering that every floor is connected by stairs or elevators, this entity returns 2 map floors - the corresponding current location and destination map floors.

Figure 4-30, shows where the map floors .PNG files are stored with file names example, considering what was mentioned above. Having both strings format, a method from Map Manager code is called to get them .PNG files from the res folder.

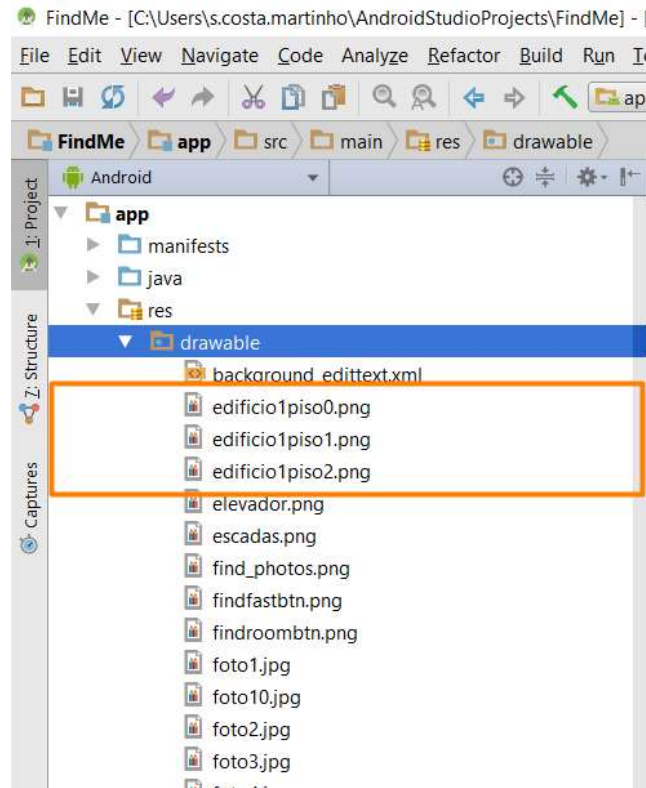


Figure 4-30 - Maps floor storage

Figure 4-31 shows the *setAppropriatedMap* method to complement the information that is described in this implemented entity. This code is responsible for returning the image to render depending on the building and floor values. Creates a string, and based on that, set on the element (*Image View*) the corresponding map floor .PNG file.

```
private Bitmap setAppropriatedMap(ImageView mapImage, BitmapFactory.Options myOptions, int building, int floor) {  
    strMapToShow = "EDIFICIO"+ building + "PISO" + floor;  
  
    switch (strMapToShow) {  
        case "EDIFICIO1PISO1":  
            mapImage.setImageResource(R.drawable.edificio1piso1);  
            mapImage.setVisibility(View.VISIBLE);  
            return BitmapFactory.decodeResource(getResources(), R.drawable.edificio1piso1, myOptions);  
        case "EDIFICIO1PISO2":  
            mapImage.setImageResource(R.drawable.edificio1piso2);  
            mapImage.setVisibility(View.VISIBLE);  
            return BitmapFactory.decodeResource(getResources(), R.drawable.edificio1piso2, myOptions);  
        case "EDIFICIO1PISO0":  
            mapImage.setImageResource(R.drawable.edificio1piso0);  
            mapImage.setVisibility(View.VISIBLE);  
            return BitmapFactory.decodeResource(getResources(), R.drawable.edificio1piso0, myOptions);  
        default:  
            break;  
    }  
  
    return null;  
}
```

Figure 4-31 - setAppropriatedMap method.

Path Finding

This entity holds an important role in this project. It has the objective of calculating the shortest path between two locations. This algorithm receives, as input, a map floor, a current location and the destination of the user. After evaluating each combination of pixels of the map floor (converted as a pixels matrix), returns an optimized path(an array of pixels).

The first step of this algorithm is mapping which areas/paths are walkable, in order to avoid forbidden areas or even impossible building parts (like walls and closed areas that is not possible to humans to go through). The map floor, given as an input, already comes with pixels marked as red. These red pixels help the mapping process, discovering which pixel is going to be a walkable pixel. Basically, the map floor is converted to a matrix of pixels, and each pixel of this matrix will be verified if it is red. If it is true, the pixel is set as walkable. Figure 4-32 has an example of a map floor with marked walkable paths.

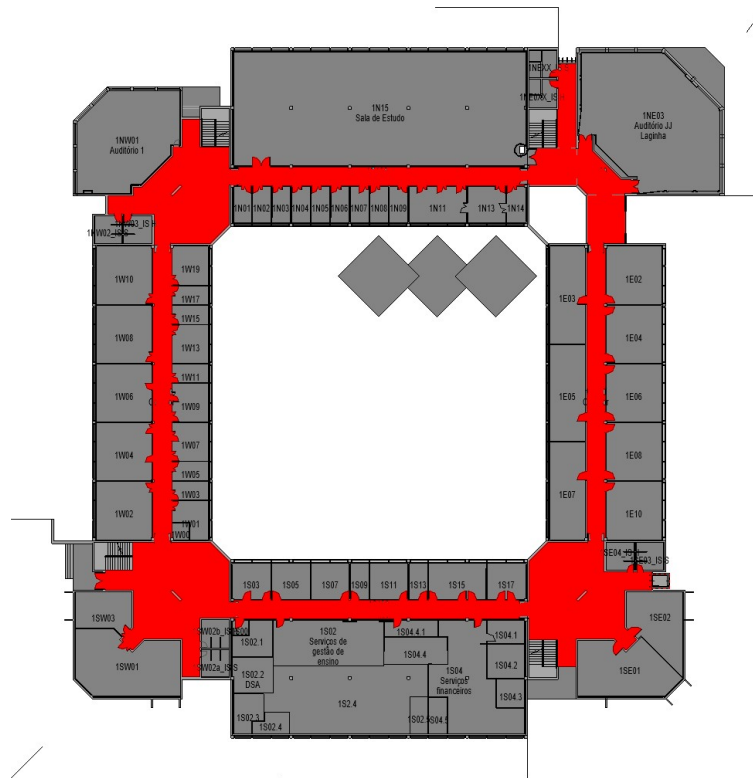


Figure 4-32 - Example of Map floor with marked walkable path in red.

This mapping process is after considered when calculating the shortest path between the two given locations. Having the locations (which are converted to pixels too)

and the possible walkable paths already calculate, it is time to get the shortest path between these two pixels.

The A* star algorithm that we are using [29], adapted from an available code developed by pawelc [30], starts to calculate multiple routes to reach the destination pixel. Having into account the X and Y from the current location and destination into pixels matrix (converted from map floor), validates all the possible routes from the start pixel to the final pixel, always considering the walkable paths already set on Mapping process. Having calculated all the possible routes, it is picked up the route that holds the lowest cost [16], which means that is also the shortest path/route between current location and destination. In the end, the lowest cost path is returned to Map View entity to be after painted above the selected map floor to show to the user the path that he/she needs to do on that floor to reach the destination. In case that the Path Finding Algorithm returns an empty array of pixels, it means that there is no possible path to reach the destination.

4.2.2 Mobile App Front End

In this section, we explain how all the entities, from the Front End Mobile App module, mentioned in section 3.2.3, were implemented. Both entities are related to everything that is shown to the user through the Find Me! App screen.

Orientation View

It is in this entity where all the orientations are provided to the user. These orientations are dynamic so every time that the user's smartphone intersects another beacon, this entity will be updated with new orientations. The presented screen, that is shown to the user on Find Me! App, can be observed in Figure 4-33 with numerical elements to be described on the following topics:

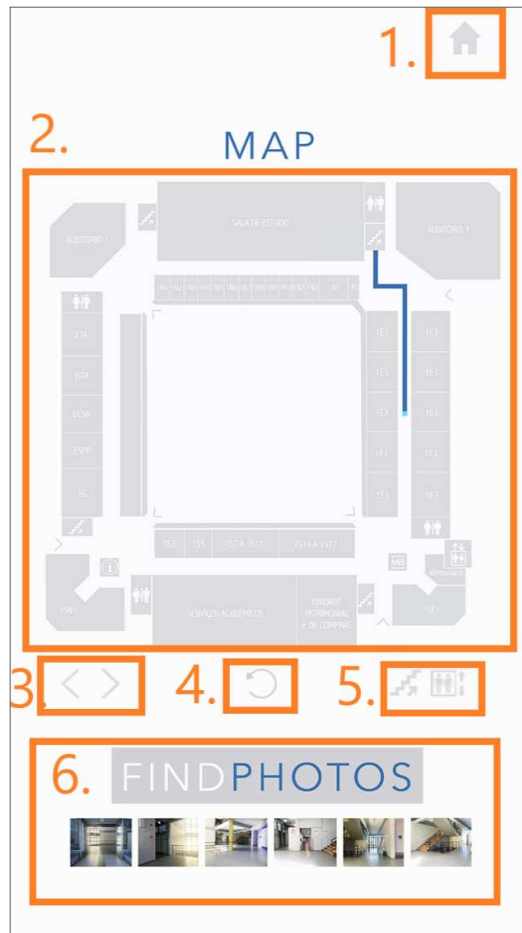


Figure 4-33 - Map view screen description

- **Home (1):** This button holds the objective to go back to the screen, where is possible to change the inserted/chosen destination, every time that is selected;
- **Map View (2):** It is an Image View element that shows the user the path that he/she needs to do to reach the destination. This entity receives a PNG image from Map Manager (see section 4.2.1 - Map Manager) and an array of pixels from Path Finding Algorithm (see section 4.2.1 – Path Finding). The PNG image it is converted to a Bitmap and a method from Orientation’s View code is called to match the array of pixels with Bitmap, in order to get which pixels are needed to be painted. Having this step done it is possible to paint the path above the PNG image, creating a new PNG image that will be set on this Image View element. Figure 4-34 shows a scheme to get more understandable what was mentioned above. Keep in mind that the pixels array are just values calculated from Path Finding Algorithm, there is no painted colour associated on the first step, so the first step is represented on this way just to get more easy to understand.

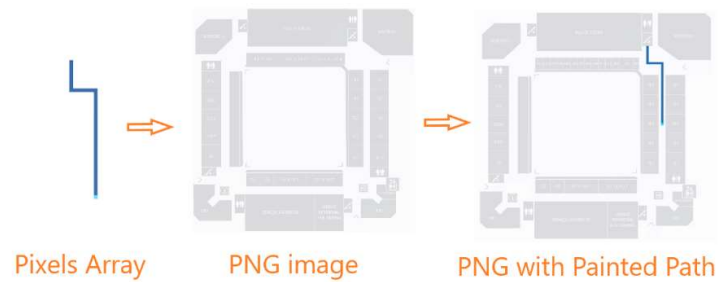


Figure 4-34 – Image View set process.

- **Change Floor Arrows (3):** This buttons group will only appear when the user's current location floor is different from the destination. This feature holds the objective, of getting to the user, the ability to check what is the path he/she needs to do on the destination floor to reach the destination. Selecting the right arrow button requests to Map Manager, the map floor image that corresponds to the destination. After that, follows the normal system flux that passes through Path Finding Algorithm that calculates the shortest path between stairs or elevator location (depending on User option, **Elevator and Stairs (5)**) and the destination. In the end, shows it on the **Map View (2)**. When Selecting left arrow, goes back to the preview image on **Map View (2)** – the current Location map floor;
- **Rotate (4):** Each time the user selects this button rotates the **Map View (2)** element 45° degrees, in order to help the user to get a better perspective of the map;
- **Elevator and Stairs (5):** This buttons group only appears when the user's current location floor is different from the destination. This feature allows the user to choose between going by elevator or stairs to the destination floor. By default, the stairs option is selected. Changing to the elevator does a request to Path Finding Algorithm to calculate, instead of return the path from the current location to the closest stairs, the shortest path between the current location and the elevator location. If the user selects the right arrow (**Change Floor Arrows (3)**) the shortest path will be calculated from the destination floor elevator location to destination location;
- **Find Photos (6):** This horizontal image scroll image view gives, to the user, crucial photos of the path that he/she needs to see as a confirmation that the user is on the right way. The Find Photos is populated according to the path that the user needs to do on the respective floor that the user is currently on. If the user

selects 3. **Change Floor Arrows (3)**, the Find Photos will be updated, depending on the path, to the destination floor also. As it is referred above, this entity receives an array of pixels from Path Finding Algorithm entity with X and Y points. To populate Find Photos element, the *getImageScroll* method is called to check if some crucial photos location (X and Y) is contained into the array of pixels.

User Interface

This entity is responsible for asking the user for the room that he/she wants to reach into the building – the destination. The user is able to pick one of two options to choose a destination, selecting a quick room or manually type a destination. Figure 4-35 shows the screen that will appear to the user with the objective to choose one of the two mentioned options.



Figure 4-35 - Choose Destination options

FINDROOM, this screen shows to the user, a text box, and a button. In the text box the user is able to text the room name that he/she wants to reach and when finished just need to select the button. Figure 4-36 shows an example of the screen.

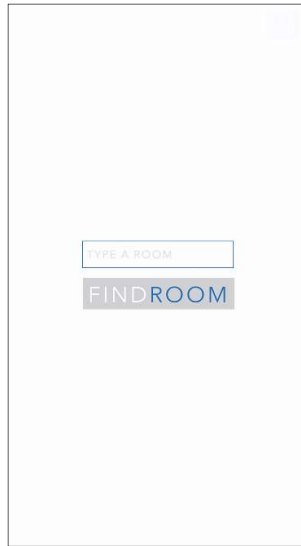


Figure 4-36 – FINDROOM option.

FINDFAST, this option offers the ability to choose a quicker destination without text need. This quickly spots can be configured by the developer, but for the first approach, and picking up the implementation example, we chose to give the user places like WC, restaurants, ATM, academic services and security station as quick spots.

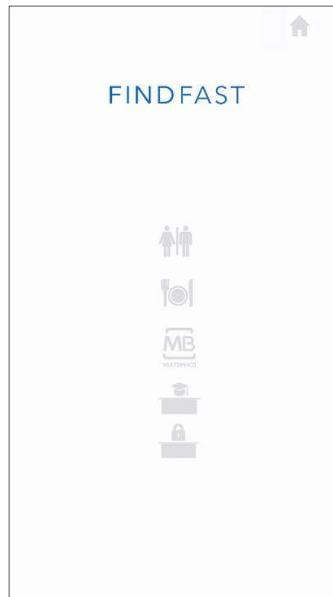


Figure 4-37 - FINDFAST option

After choosing the FINDROOM or FINDFAST, a string of the typed/picked room name is sent to the Database entity to convert it as a location.

4.2.3 Entities Relation

Having all the entities described in detail in 4.2.1 and 4.2.2 sections, Figure 4-38, shows how the different entities communicate with each other in order to give the right orientations to the user as a sequence diagram. Note that the BIM model entity is not implicit on this diagram, because this entity is only needed when there are changes on the input data, like rooms or beacons information.

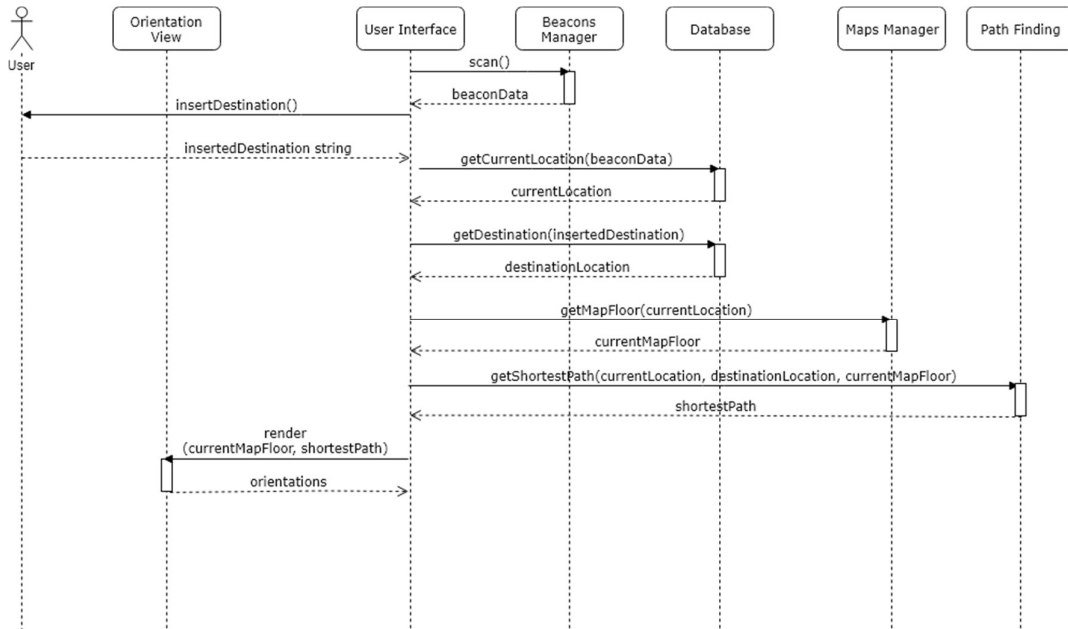


Figure 4-38 - Sequence Diagram of Orientations Process

When the *Find Me!* App runs, automatically scans for a beacon. If a beacon is intersected, Beacon Manager entity returns the intersected beacon data (UUID, Major, and Minor values) and triggers the next activity/screen of the mobile App – Insert Destination. The user types the room destination. After that, `getCurrentLocation` and `getDestination` queries run on the database that returns the current location and the destination location. Having these two locations, the Map manager is called in order to provide the corresponding maps floor. Having current location, destination location and the respective map floor, the Path Finding entity runs with these given arguments and returns the shortest path between the two locations. The Orientation View receives that information and renders it. While the user is walking, reaching the destination, the App keeps scanning for other beacons with the objective of getting an updated current location of the user. After that, the *Find Me!* App follows the normal flux until the user reaches the destination.

Chapter 5 Implementation at the ISCTE-IUL Campus

Every year, ISCTE-IUL receives an average of 1150 new student distributed in sixteen graduate courses and approximately 200 to Postgraduate, Master and Ph.D. degrees. A high percentage of these students do not know how the facilities are organized as a physical structure and when the moment to go to a specific room comes, they need to ask how to get there.

After meetings with ISCTE-IUL Technology and Information department, it was decided and confirmed that ISCTE-IUL is going to be used as validation approach to this concept.

5.1 Beacon Installation Process

We have chosen one building of the three that exists in ISCTE-IUL to confirm if the presented solution would work in a practical case. The chosen building is the Edificio 1, holds three floors (0, 1 and 2), 4 main entrances/exits, 12 stairs (4 for each floor) and 164 rooms. Each room name, excluding the academic services rooms of the faculty, holds an ISCTE nomenclature of [Floor Number] + [cardinal point orientation] + [Incremental number of the room], for example 2E10 - which means that this room is located on the 2nd floor on the east aisle of the building and sequentially is the 5th on corridor. Figure 5-1 shows each map floor of the Building 1 - Edificio Sedas Nunes. The building has three floors, has an 80 m long square shape and contains approximately 350 rooms.

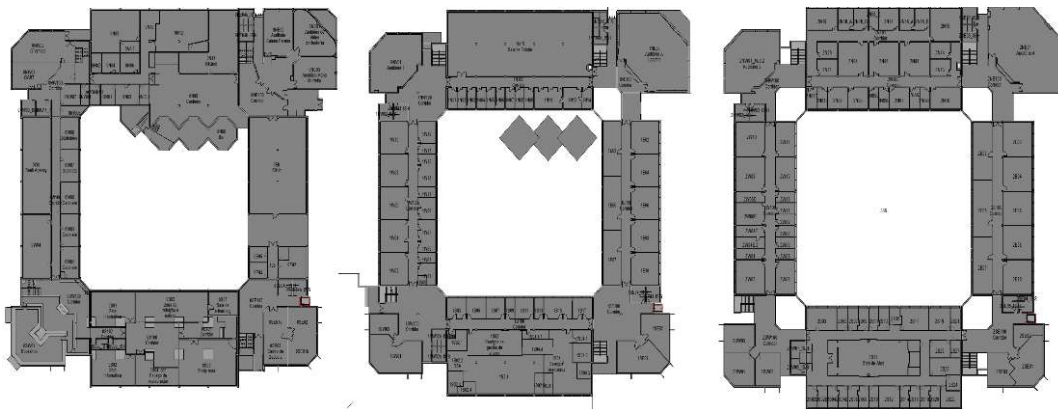


Figure 5-1 - Maps of floors 0,1 and 2 of the pilot building - Edificio 1.

Applying the Beacons Placement rules and Beacons Distance Rules in ISCTE-IUL *Edifício 1*, Figure 5-2, Figure 5-3 and Figure 5-4 show where the beacons are going to be placed in floor 0, floor 1 and floor 2 respectively. In each floor, we have the stairs and the elevator on the same zone that the paths intersect and also the same zone where are the entrances/exits, so there is no need to triplicate the beacons, one per each zone is enough. The rest of the beacons are placed on the center of the corridors, following Beacons Placement rules. Considering Beacons Distance Rules, the beacons ED1P2B, ED1P2C, ED1P2H and ED1P2G were set to -20 dBm of broadcasting value. The rest of the beacons, because the closest neighbor beacon, of each beacon, has more than 7 meters of distance, were set to -20 dBm.

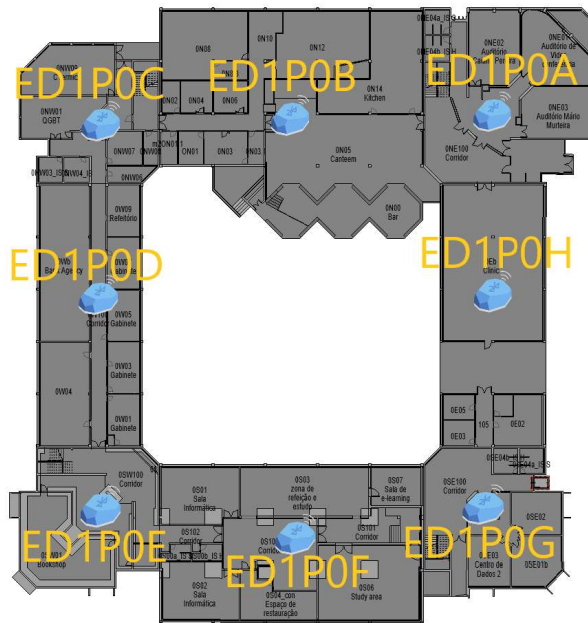


Figure 5-2 - Floor 0 with beacons placement.

Table 5-1 - Beacons configuration values

| Name | UUID | Major | Minor | Broadcasting power (Dbm) |
|--------|--------------------------------------|-------|-------|--------------------------|
| ED1P2A | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 5 | -16 |
| ED1P2B | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 9 | -20 |
| ED1P2C | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 10 | -20 |
| ED1P2D | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 7 | -16 |
| ED1P2E | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 20 | -16 |
| ED1P2F | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 21 | -16 |
| ED1P2G | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 22 | -20 |
| ED1P2H | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 23 | -20 |
| ED1P2I | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 24 | -16 |
| ED1P2J | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 25 | -16 |
| ED1P1A | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 1 | -16 |
| ED1P1B | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 2 | -16 |
| ED1P1C | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 6 | -16 |
| ED1P1D | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 11 | -16 |
| ED1P1E | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 8 | -16 |
| ED1P1F | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 18 | -16 |
| ED1P1G | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 19 | -16 |
| ED1P1H | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 26 | -16 |
| ED1P0A | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 3 | -16 |
| ED1P0B | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 4 | -16 |
| ED1P0C | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 12 | -16 |
| ED1P0D | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 13 | -16 |
| ED1P0E | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 14 | -16 |
| ED1P0F | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 15 | -16 |
| ED1P0G | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 16 | -16 |
| ED1P0H | 7AE702E0-E1A7-EEA9-E127-4C304EC7D4DF | 1 | 17 | -16 |

5.2 Implementation Results

In this section, we evaluate the performance of the *Find Me!* App. In this evaluation, it is going to be described which smartphone was used (and its specifications), how many and what beacons were used and where were physically placed, and, in the end, the beta tests that were chosen to evaluate the mobile App in different environments.

5.2.1 Setup

In this section, it is going to be described, in detail, the equipments that were used in this implementation at the ISCTE-IUL Campus.

Mobile device

This evaluation was composed of two types of equipment's, a smartphone and a set of 26 *Estimote* beacons.



Figure 5-5 - Smartphone Huawei P8 gra-109

Smartphone specifications are as follows

- **CPU:** Octa-Core, 2 processors: 2Ghz Quad-Core ARM Cortex-A53 and 1.5Ghz Quad-Core ARM Cortex-A53 [31];
- **GPU:** ARM Mali-T628 MP4;
- **Random-access Memory (RAM):** 3GB;
- **Internal storage:** 16 GB;
- **Screen:** 5.2 inches;
- **Battery:** Lithium-ion Polymer (LiPo) - Non-removable;
- **Operating System (OS):** Android 5.0.2 (Lollipop [32]).

Estimote Beacons



Figure 5-6 - Estimote Proximity Beacon

- **Default Battery Life:** 3 years;
- **Maximum Battery Life:** 5 years;
- **Max. Range:** 70 meters;
- **Thickness:** 25 mm.

5.2.2 Find Me! Tests and Results

Considering the beacon installation process in section 5.1, 26 beacons were placed in *Edificio 1* and are going to be part of the following Test cases using Find Me! App. First, we describe each test (and our expectations) in detail. In the end, we analyze the results provided by the mobile App.

Test Cases

To perform these tests, we needed to choose where would be the current position/location of the user in *Edificio 1* and also pick the final destination. But before that, we decided to split these tests into two approaches: **the current location floor is the same of the destination** and the other one, **the current location floor is different from the destination floor**. Also, in the end we tested the situation of **running the mobile App in a position that is not covered by a beacon region**.

Test 1 – The current location floor is the same as the destination

In this test, the current location floor is the same as the destination floor so there is no need to use stairs or elevator, so we expected that the App only renders one map floor. The user is going to choose the room 1E02 as a destination, and his current location is

going to be on the west side of the floor on the left bottom. Each location can be confirmed in Figure 5-7, the current location in yellow and destination in red color.

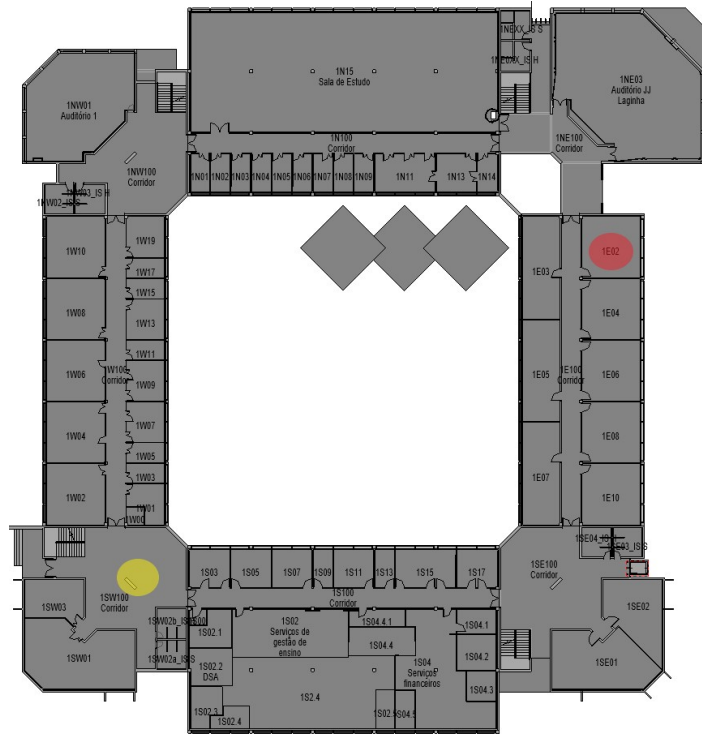


Figure 5-7 – Test 1 - Floor 1 map with current (yellow) and destination (red) locations marked.

Having this current location and destination on the same floor, the *Find Me!* App can calculate two possible paths to reach the destination. One is through the south corridor and the other through the east corridor. Figure 5-8 represents the two expected possible path options and in Figure 5-3 we can confirm where are placed the beacons in this floor, calculated through the Beacons Placement rules diagram (see section 0 – Beacon Location Rules).

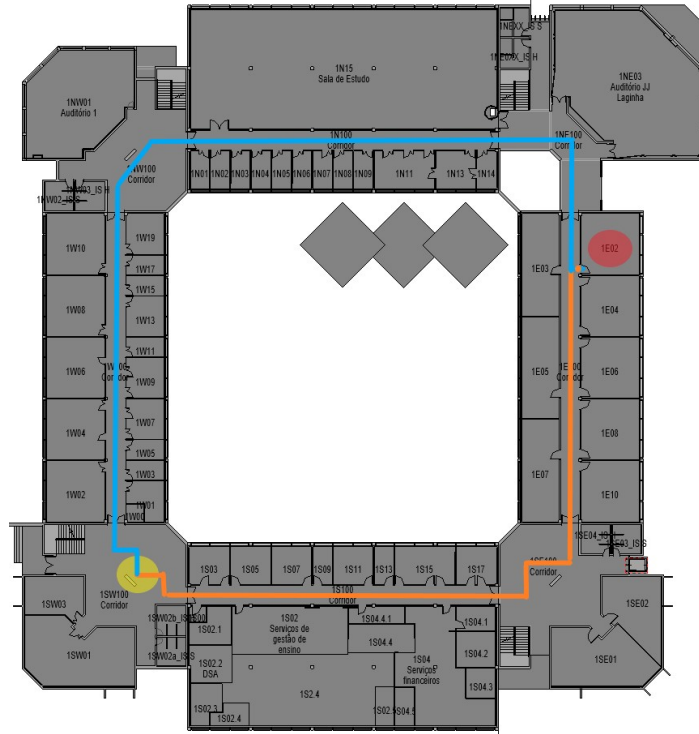


Figure 5-8 - Test 1 - Floor 1 with paths possible options marked.

Test 2 - The current location floor is different than the destination floor:

This test was created to evaluate the performance of *Find Me! App* when the user needs to change floors, to reach the destination. To change floors, the user has one of two options: going by stairs or take the elevator. So for that reason, we decided to split this test into two subtests: **reach the destination by stairs** and **reach the destination by elevator**. In both subtests, the current location is going to be on the west side of floor 1, and the destination is going to be the room 2N05E. The current location is marked as yellow, and the destination is marked as red, both can be confirmed in Figure 5-9 and Figure 5-10 respectively.

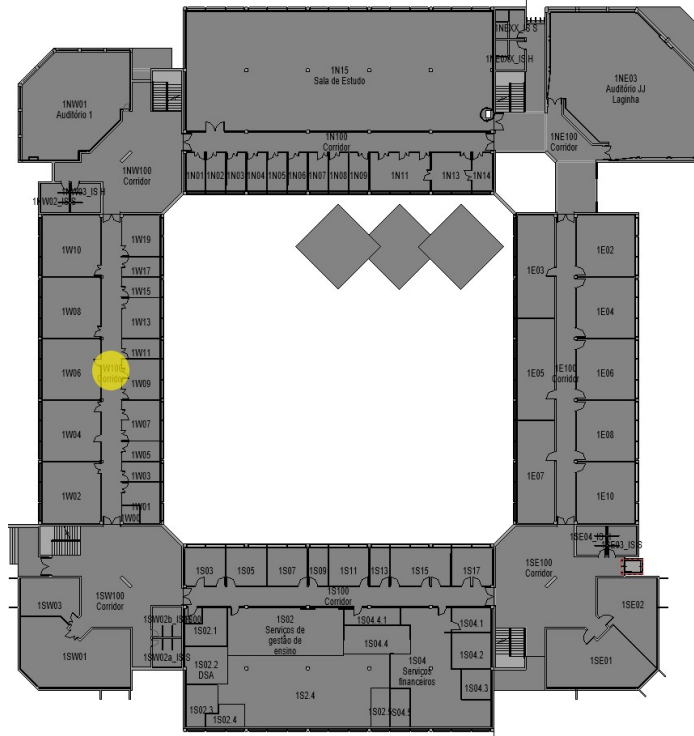


Figure 5-9 - Test 2 - floor 1 with the current location (yellow).

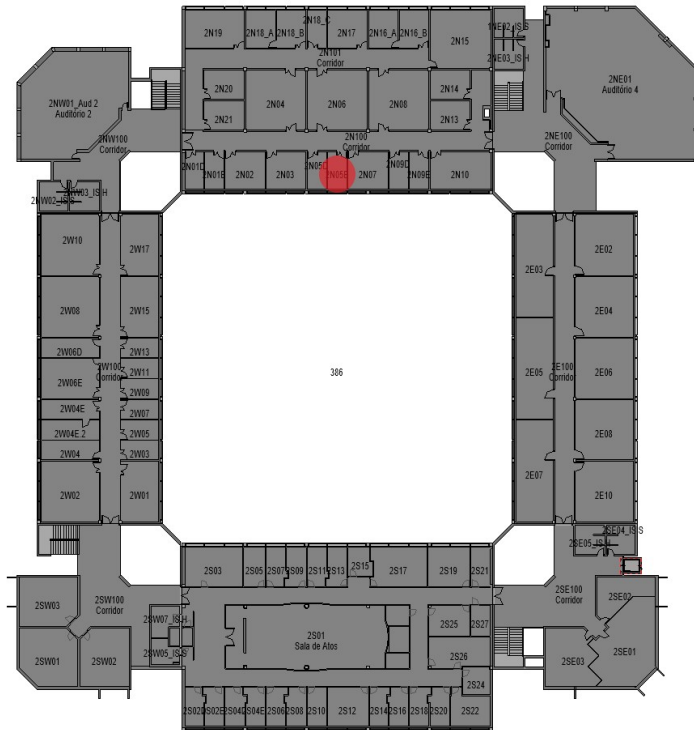


Figure 5-10 - Test 2 - floor 2 with the destination (red).

Test 2 – subtest 1 reach the destination by stairs:

In this subtest, we want to evaluate the mobile App when the user, reaches the destination going by stairs (which is the default selected option when the current location floor is

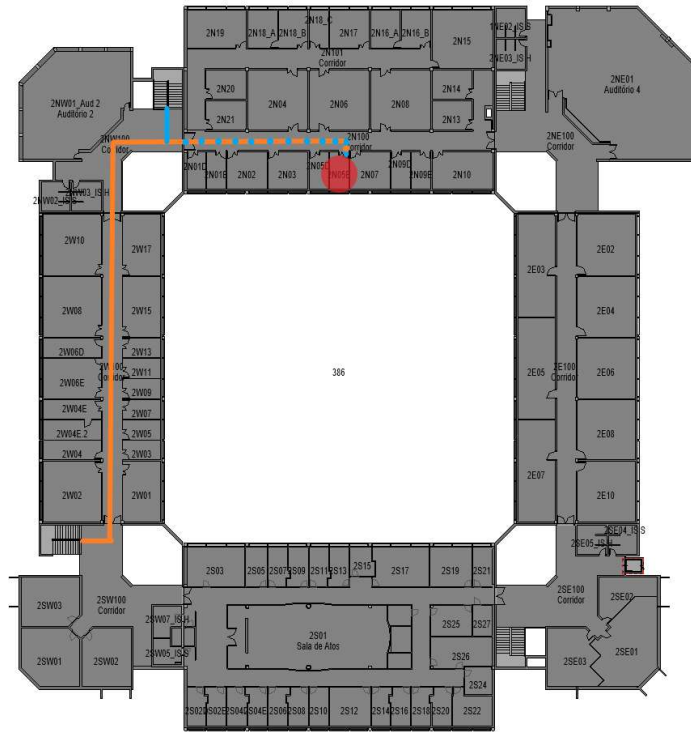


Figure 5-12 - Test 2, subtest 1 - floor 2 with path options.

Test 2 – subtest 2 reach the destination by elevator:

In this subtest, we evaluate the user reaching the destination by taking the elevator. To pick up this option, the user needs to select the elevator button to recalculate the shortest path. Considering the current position of the user, Figure 5-13 shows the path that is expected to show on *Find Me!* App to reach the elevator on floor 1. In Figure 5-3 we can confirm where are placed the beacons on floor 1.

Test 3 – Running the mobile App in a position that is not covered by a beacon region

This test holds the objective of evaluating the Find Me! App behaviour when was running in a building position that wasn't covered by a beacon region. The chosen floor was the floor 1 and the destination room was 1E02. This destination and the user current location can be confirmed in Figure 5-15 and we can confirm where are placed the beacons of floor 1 in Figure 5-3.

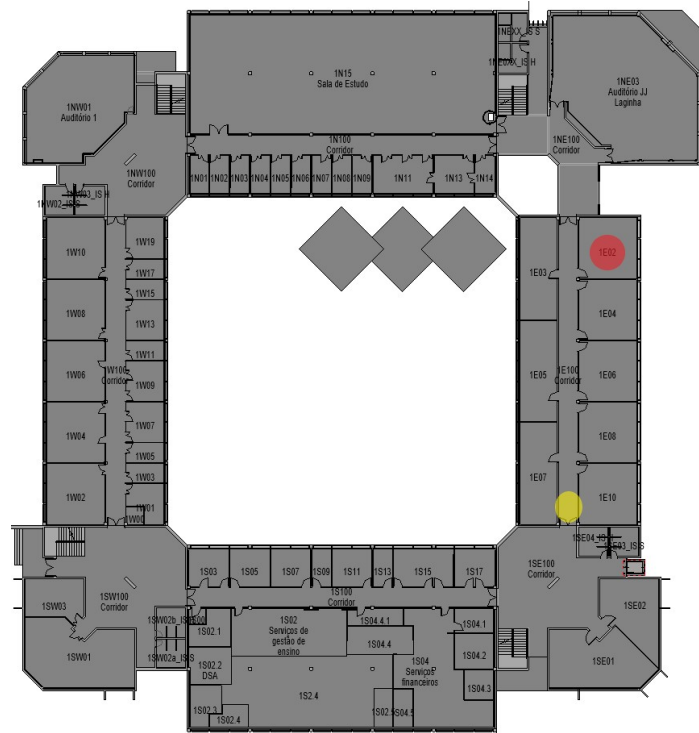


Figure 5-15 - Test 3 - Floor 1 map with current (yellow) and destination (red) locations marked.

Results

In this section, we evaluate the results of the described tests above and make conclusions about each one. We are going to start with Test 1 as the first pilot test.

Test 1 – The current location floor is the same as the destination

In this test, the user typed the room 1E02 as a destination. The position where he started to use the Find Me! (see Figure 5-7 to check the current user and destination position), was covered by a beacon ED1P1E, so the mobile App could easily calculate his current location. Figure 5-16 shows a Find Me! App screen of where the user was, when started using the App. The previous figure also shows that the Find Me! App chooses the orange path option (see Figure 5-8) as the shortest path to reach the destination.

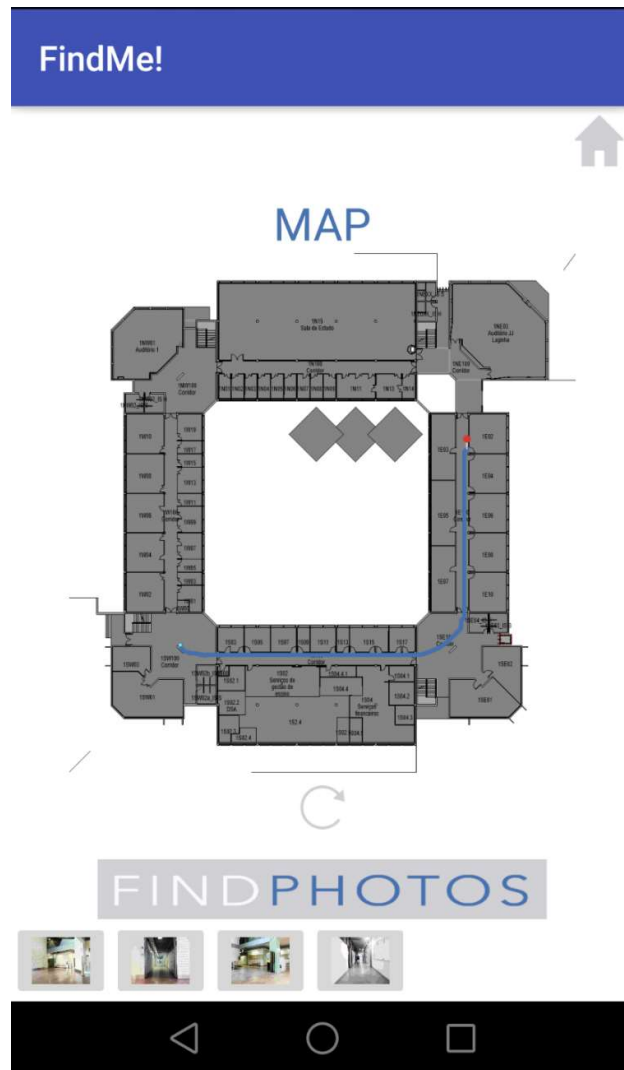


Figure 5-16 - Results of Test case 1, Beacon ED1P1E intersection.

As long as the user was walking, following the given orientations from the Find Me! App, intersects new beacons so the path and the FINDPHOTOS were also updated. Figure 5-17, Figure 5-18 and Figure 5-19 show the screenshots of the App when intersected ED1P1F, ED1P1G and ED1P1H beacons respectively.

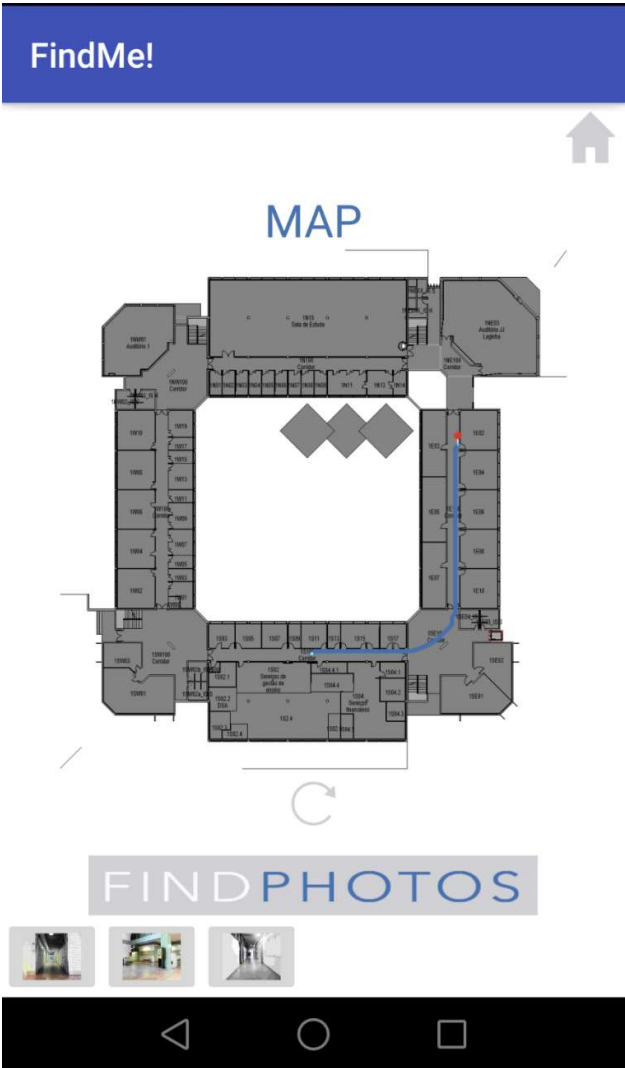


Figure 5-17 - Results of Test case 1, Beacon ED1P1F intersection.

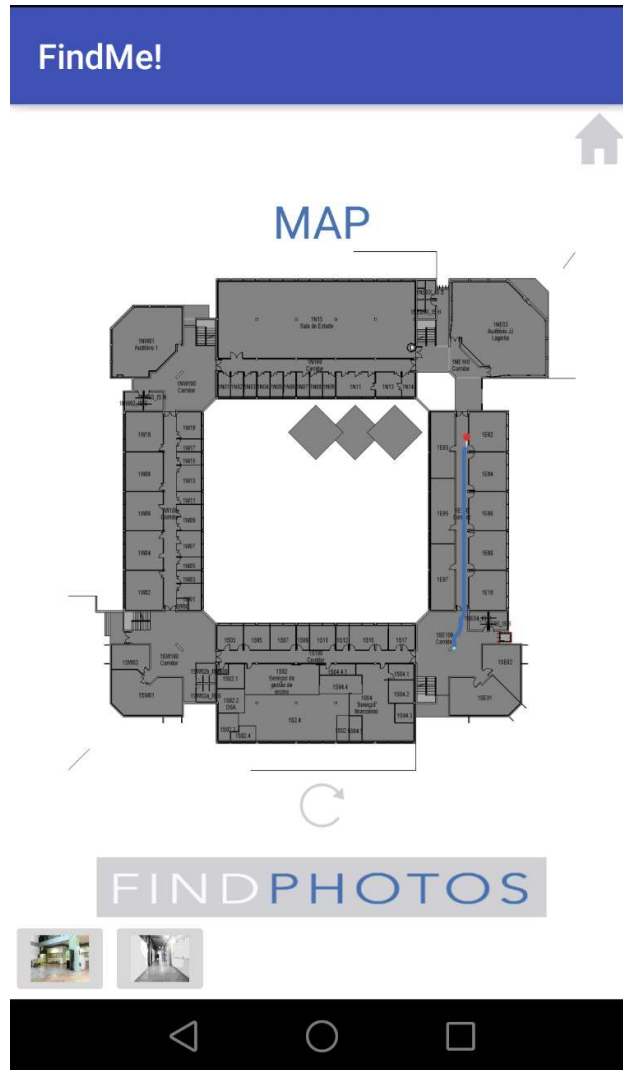


Figure 5-18 - Results of Test case 1, Beacon ED1P1G intersection

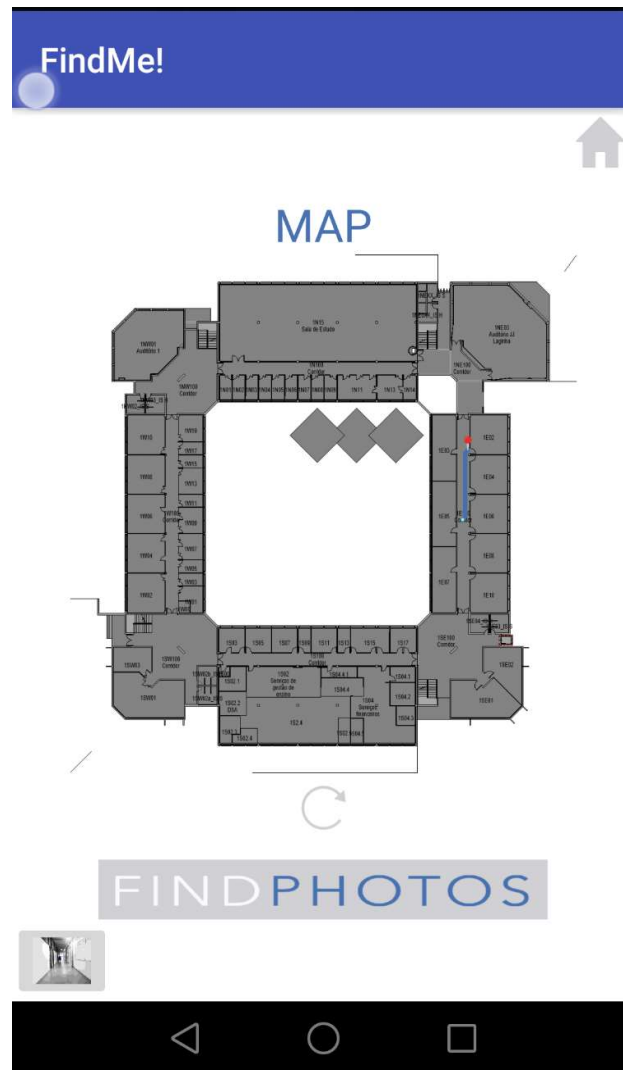


Figure 5-19 - Results of Test case 1, Beacon ED1P1H intersection.

The user reached the destination, but we detected a kind of limitation of Find Me! App. Besides the user position is the same as the destination, when the user finally reaches it, the mobile App cannot get that kind of information to assume that the user already gets the room. This point is going to be analyzed and improved as future work if we conclude that is a requirement.

Test 2 - The current location floor is different than the destination floor:

In this test, the user typed the room 2N05E as a destination. The position where he started to use the Find Me! (see Figure 5-9 to check the current user and Figure 5-10 destination position), was covered by a beacon ED1P1D

Subtest 1 - reach the destination by stairs:

The stairs option is chosen by default so when the Find Me! App intersected the beacon ED1P1D signal, having 2N05E as a destination, it automatically calculated the shortest path between the user current location to the nearest stairs and Figure 5-20 demonstrates it. Also, the App considered the nearest stairs as the blue option and not the orange option (see Figure 5-11 to watch the expected options).

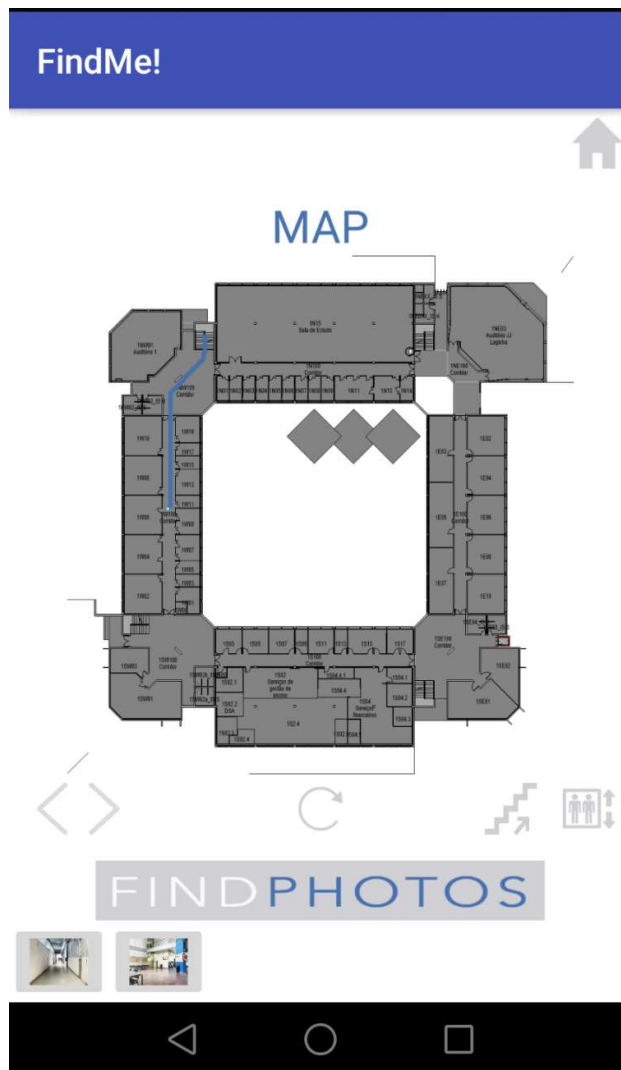


Figure 5-20 - Results of Test case 2, Subtest1, Beacon ED1P1D intersection.

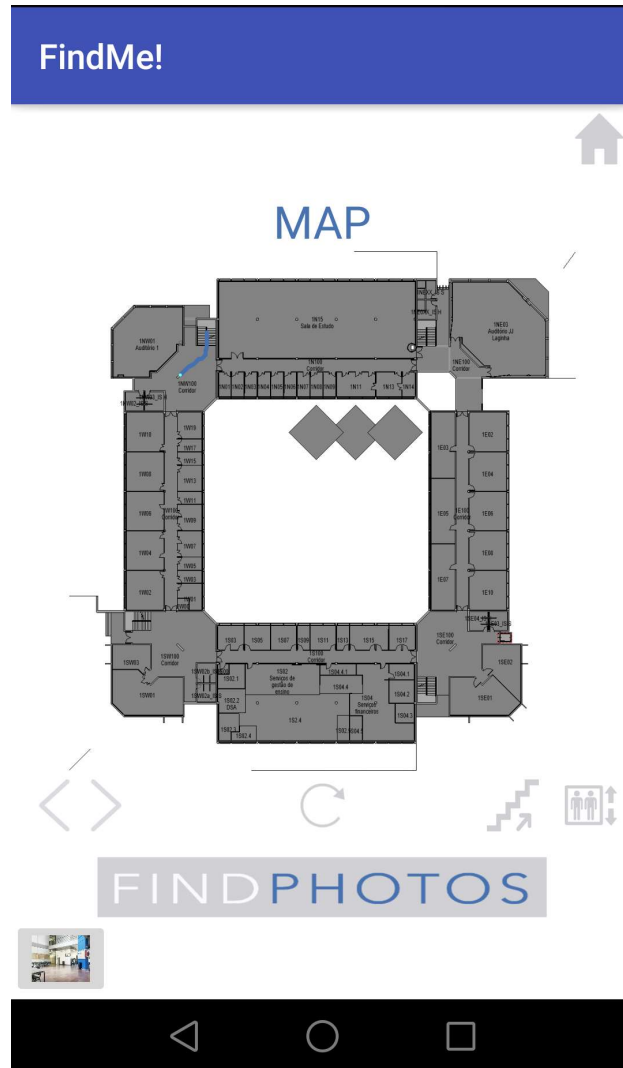


Figure 5-21 - Results of Test case 2, Subtest1, Beacon ED1P1C intersection.

When the user went upstairs, intersected the ED1P2C beacon and automatically the map floor, from floor 2, was rendered as is shown in Figure 5-22.

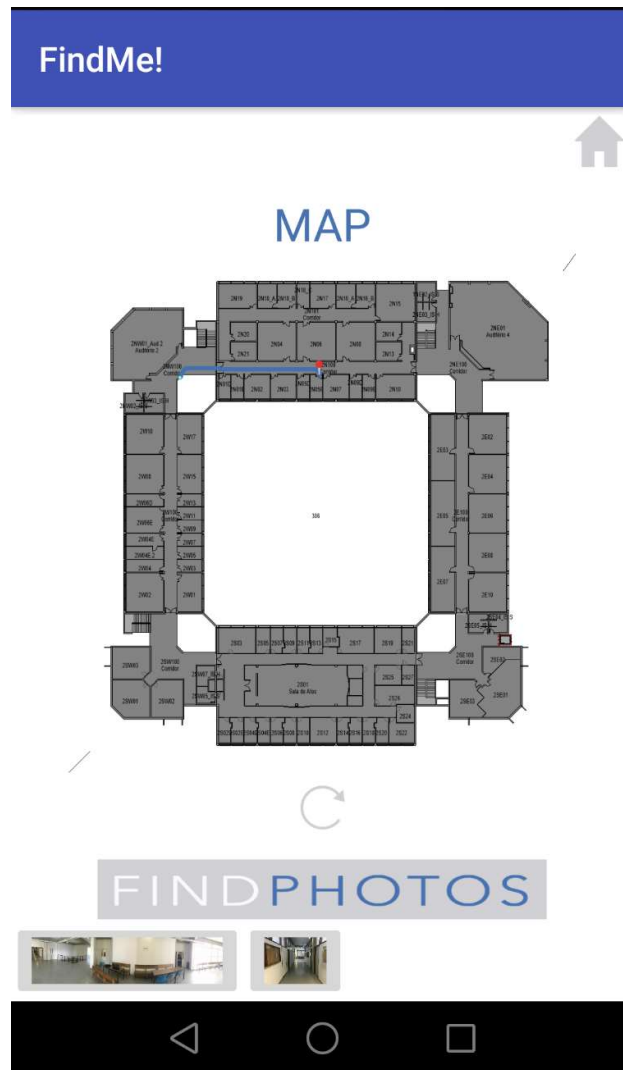


Figure 5-22- Results of Test case 2, Subtest1, Beacon ED1P2C intersection.

The destination room, was right below of beacon location so it was easy to find the room through the Find Me! App. Figure 5-23 shows the beacon ED1P2B intersection.

In this subtest, we think that everything occurred as expected and there were no obstacles.

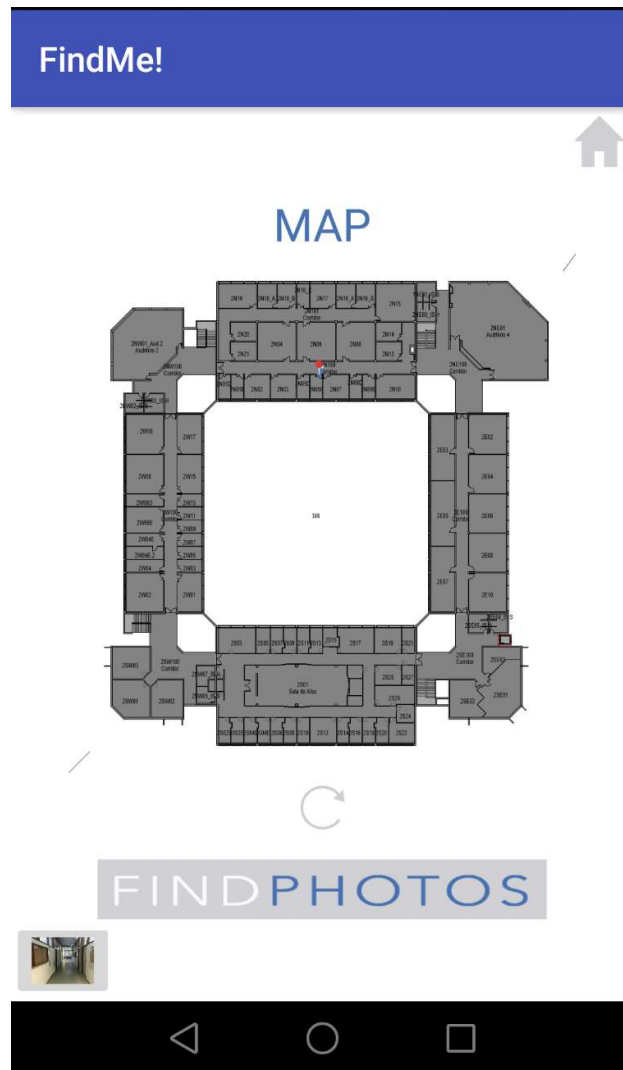


Figure 5-23 - Results of Test case 2, Subtest1, Beacon ED1P2B intersection.

Subtest 2 - reach the destination by elevator:

In this subtest, the initial current location of the user is the same as in subtest 1, because the intersected beacon is also the same – ED1P1D. The elevator button was pressed to calculate the shortest path through the elevator. Figure 5-24 shows a screenshot of the Find Me! with the shortest path, calculated, to the elevator.

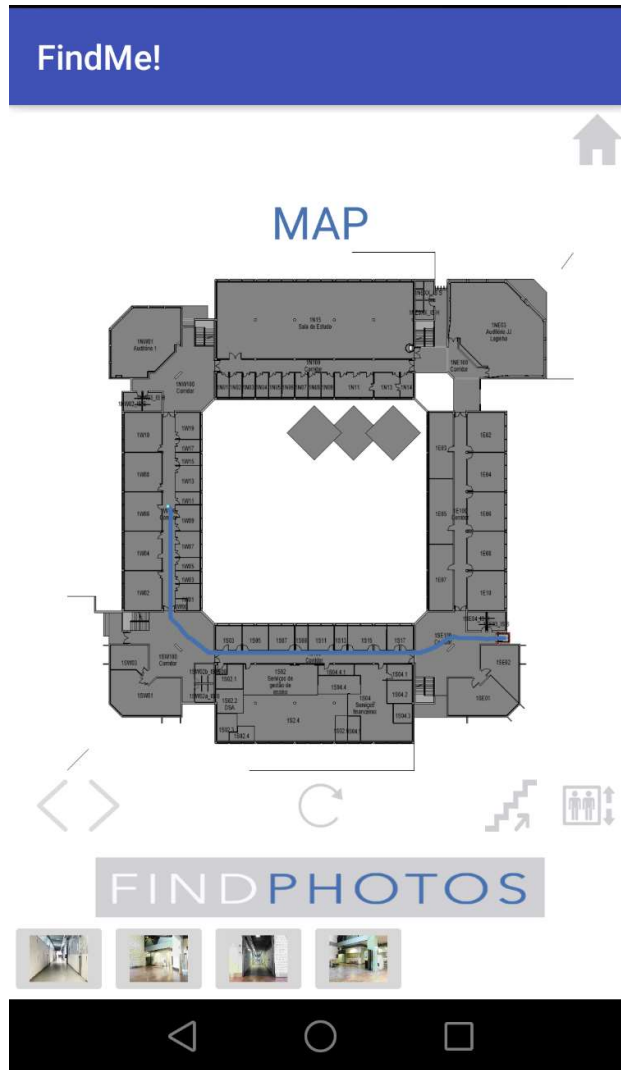


Figure 5-24 - Results of Test case 2, Subtest2, Beacon ED1P2D intersection.

Figure 5-25 shows several states of the mobile App when intersected the beacons ED1P1E, ED1P1F, and ED1P1G until the user reaches the elevator.

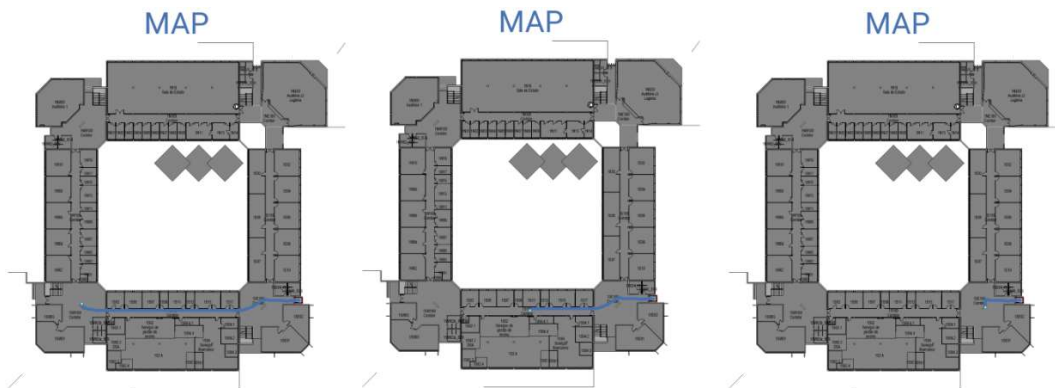


Figure 5-25 - Results of Test case 2, Subtest2, Beacon ED1P1E(left), ED1P1F(middle) and ED1P1G(right) intersection.

After the user took the elevator, in order to go to floor 2, the App intersected the beacon ED1P2G and rendered the map floor, from floor 2, with the new updated path. Figure 5-26 shows the rest of the states that the Find Me! App assumed when intersected the beacons ED1P2G, ED1P2H, ED1P2A, and ED1P2B. Having the same destination and initial current location of the subtest1, we confirmed that Find Me! App worked as expected.

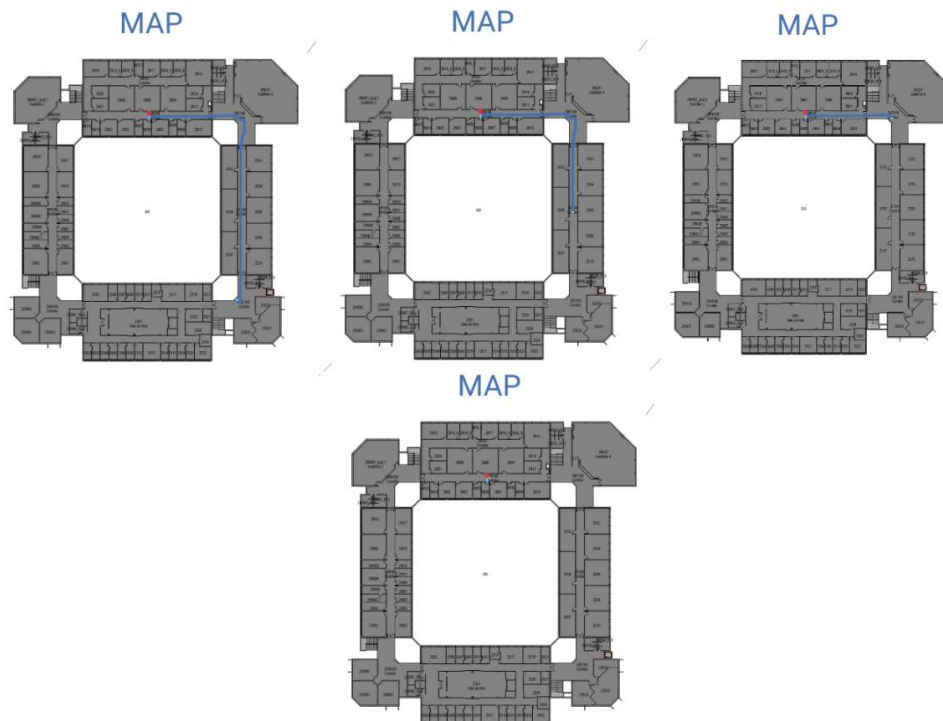


Figure 5-26 - Results of Test case 2, Subtest2, Beacon ED1P2G (left), ED1P2H (middle), ED1P2A (right) and ED1P2B (bottom) intersection.

Test 3 – Running the mobile App in a position that is not covered by a beacon region

The user ran the App and the Find Me! stayed locked in the Insertion Mode options screen, showing the message: “BLE NOT FOUND”. In Figure 5-27 it is possible to see the screenshot of the Insertion Mode options returning the mentioned message.

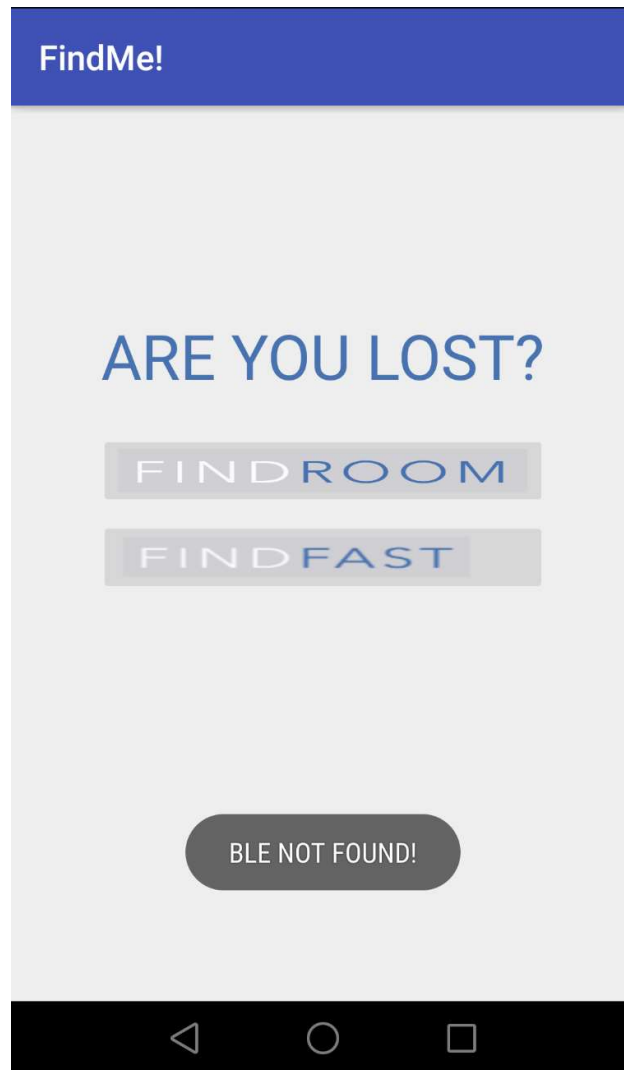


Figure 5-27 - Results of Test case 3, Beacon not intersected message.

The user started walking north, and, a few meters after, intersected the beacon ED1P1H. Having received the BLE signal, the Insertion Mode options screen unlocked the available options and the user typed the room destination. Figure 5-28 shows a screenshot of the beacon intersection moment.

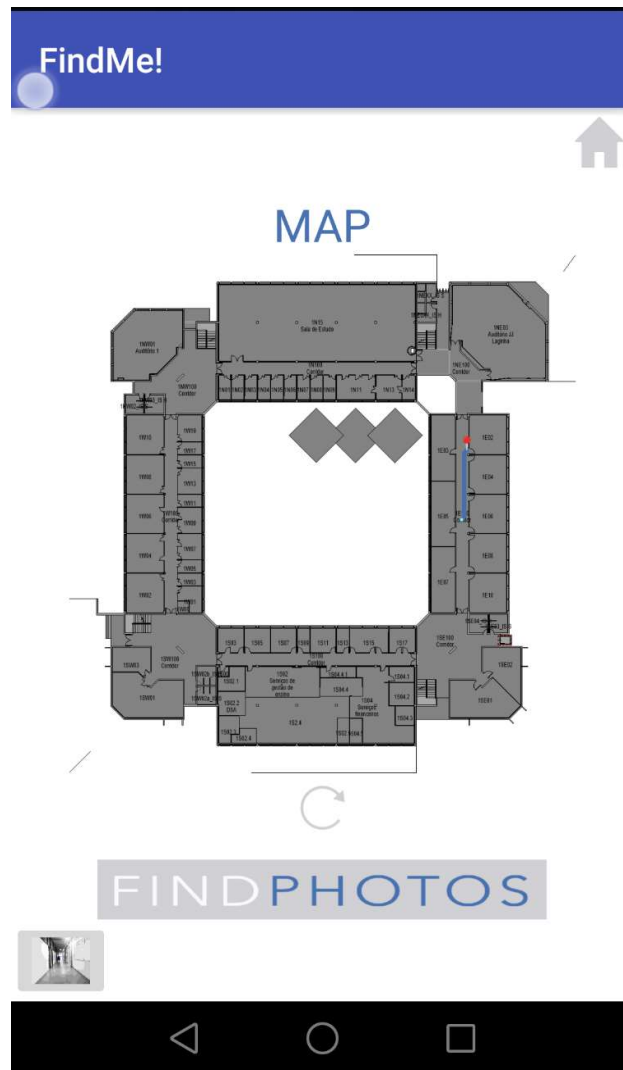


Figure 5-28 - Results of Test case 3, Beacon ED1PIH intersection.

Having a placed beacon per each intersection path and corridor centre worked as “workaround” in this test case, but we think that this is a limitation of this project when there is no coverage in the moment of running the mobile App. This situation needs to be analyzed and after improved as a future work.

Chapter 6 Conclusion

We have developed a generic ILS combined and complemented with BIM and Beacons technology. With this approach, the End Users can have access to updated guidance orientations in their own smartphone which will, certainly, help a lot of people into complex buildings. In this project proposal, we introduced a concept, which it is considered by us, as a simple way of combining the geometry of the map (coming from BIM Model), the beacons and a mobile App without having trouble with signal triangulations, and since we implemented a local database, we have no dependencies to internet access. Also, with BIM models help, it is possible to calculate where each beacon can be placed (Beacon Location Rules) without human calculation. This process can reduce this project budget value without decreasing the ILS performance.

The ILS technology of this project should be the most compatible as possible to the most used smartphones, so we analyzed the ILS devices technology available. Beacon technology was our choice, using the *iBeacon* protocol. We also choose Android OS as a target, so we developed our mobile App in Android Studio IDE.

We picked ISCTE-IUL – Edificio 1, as an implementation test to validate this project proposal. We analyzed the BLE signal, coming from several beacons and their behavior, which allowed us to create the Beacon Location Rules. Having all the beacons placed, with the right position and the correct coverage values, we have made Find Me! App experiments in various situations. In general, our test results fulfilled the expectations.

We have implemented a solution that can be applied and also customized, to every type of complex buildings, with the condition that a 3D model of the building needs to be created for the BIM model to after generating all the building data needed to the mobile Application. Having this requirement fulfilled, just need to place and configure the beacons.

We have published a paper - Find_Me: IoT Indoor Guidance System – in System Proceedings of International Conference on Ambient Intelligence (ISAMI'18), 20th-22nd June 2018 Toledo, Spain [33]. We also participate in the FCT 2018 Science Program, doing a demo of the presented solution, cooperating with the investigation group ISTAR of ISCTE-IUL. A paper to MDPI Sensors journal is in preparation (<http://www.mdpi.com/journal/sensors>) to be submitted in October 2018.

6.1 Future Work

Along with this project implementation, we found some limitations that we want to improve on the next releases of this project.

One of them is when the user is in a position that is not covered by beacon range. This situation can be improved by one of two options: incrementing the number of beacons (which will increase the project budget) or returning, in mobile App, information messages for the user to move along until intersects a beacon region - since we covered every intersection paths, this last Approach would certainly work.

Other needed improvement, when the user reaches the destination there is no way to know it, from the mobile App perspective. On the next release, we expect to analyze if placing NFC tags in each room door, would be a benefit to this project. With these tags, the user would need to pass the smartphone close to the tag in order to receive their content and send it to the Find Me! App as a confirmation of reaching the destination.

Using our solution, it is also possible to implement a Reports Generator, taking into account user information like: most searched rooms, the building zone where the users usually get lost, stairs or elevator preference, most unsearched rooms or other analyzed options. This report can be after analyzed by the building entities and make conclusions about that.

Implement the idea of simulating a path without needing to be physically present in the building. Just to give the option of checking the path before going to the complex building. To make it possible, it will be provided a 3D image of the entire building and the user will be able to choose which is the entrance that he wants as the first current location. After that, the user just needs to insert the destination and the Path Finding Algorithm will calculate the simulating path between these two locations and shows it. It will have almost the same flux of what is implemented right, but in this case, the path won't be updated by de beacons because the user is outside of the building. With this improvement, this App can be used everywhere.

References

- [1] L. i. Inc, “Indoor location technologies compared,” 2017. [Online]. Available: <https://lighthouse.io/indoor-location-technologies-compared/>. [Accessed: 02-Dec-2017].
- [2] L. Ilkovičová, J. Erdélyi, and A. Kopáčík, “Positioning in Indoor Environment using QR Codes,” *Competence Cent. SMART Technol. Electron. Informatics Syst. Serv. ITMS 26240220072*, pp. 117–122, 2014.
- [3] T. Radar, “NFC description.” [Online]. Available: <https://www.techradar.com/news/what-is-nfc>. [Accessed: 08-Aug-2018].
- [4] Onemobiz, “NFC tag.” [Online]. Available: <http://onemobiz.com/nfc-marketing/>. [Accessed: 08-Aug-2018].
- [5] A. Developers, “Bluetooth Low Energy.” [Online]. Available: <https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>. [Accessed: 08-Jan-2018].
- [6] E. Inc., “Estimote Products Specification.” [Online]. Available: https://estimote.com/products/?gclid=Cj0KCQiAyszSBRDJARIsAHAqQ4om91_Co4TM8pBLM44BeJIQAwcc4JNeazH2sXqhaENhVX0f2HOOcaAh51EALw_wcB. [Accessed: 08-Jan-2018].
- [7] BlueCats, “BlueCats AA Beacon.” [Online]. Available: <https://www.bluecats.com/aa-bluetooth-beacon/>. [Accessed: 08-Aug-2018].
- [8] BlueCats, “BlueCats Coin Beacon.” [Online]. Available: <https://www.bluecats.com/coin-mobile-beacon/>. [Accessed: 08-Aug-2018].
- [9] BlueCats, “Blue Cats USB Beacon.” [Online]. Available: <https://www.bluecats.com/usb-beacon/>. [Accessed: 08-Aug-2018].
- [10] R. Resende *et al.*, “Plataforma Web-BIM para Gestão de Instalações de um Campus Universitário,” in *1º Congresso Português de Building Information Modelling*, 2016, pp. 501–511.
- [11] Dynamo, “Dynamo BIM.” [Online]. Available: <http://dynamobim.org/>. [Accessed: 08-Aug-2018].
- [12] Apple, “Swift.” [Online]. Available: <https://developer.apple.com/swift/>. [Accessed: 08-Aug-2018].
- [13] Linux, “Linux OS.” [Online]. Available: <https://www.linux.com/what-is-linux>. [Accessed: 20-Jan-2018].

References

- [14] “Comparative study of Mobile Platforms 2017.” [Online]. Available: <https://android.jlelse.eu/apple-vs-android-a-comparative-study-2017-c5799a0a1683>. [Accessed: 20-Jan-2018].
- [15] D. Android, “Android Platform Architecture.” [Online]. Available: <https://developer.android.com/guide/platform/index.html>. [Accessed: 20-Jan-2018].
- [16] W. Y. Loong, L. Z. Long, and L. C. Hun, “A STAR PATH FOLLOWING MOBILE ROBOT Wong,” *4th Int. Conf. Mechatronics*, no. May, pp. 17–19, 2011.
- [17] M. SQL, “MySQL about.” [Online]. Available: <https://www.mysql.com/about/>. [Accessed: 28-Jan-2018].
- [18] ORACLE, “ORACLE About.” [Online]. Available: <https://www.oracle.com/corporate/index.html>. [Accessed: 28-Jan-2018].
- [19] Itxdesign.com, “Comparing My SQL vs ORACLE.” [Online]. Available: <https://itxdesign.com/mysql-vs-oracle/>. [Accessed: 28-Jan-2018].
- [20] B. Ozdenizci, K. Ok, V. Coskun, and M. N. Aydin, “Development of an indoor navigation system using NFC technology,” *Proc. - 4th Int. Conf. Inf. Comput. ICIC 2011*, pp. 11–14, 2011.
- [21] X. Y. Lin, T. W. Ho, C. C. Fang, Z. S. Yen, B. J. Yang, and F. Lai, “A mobile indoor positioning system based on iBeacon technology,” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, vol. 2015–Novem, pp. 4970–4973, 2015.
- [22] R. Belwariar, “A* Algorithm,” 2017. [Online]. Available: <https://www.geeksforgeeks.org/a-search-algorithm/>. [Accessed: 08-Dec-2017].
- [23] E. Inc., “iBeacon Protocol,” 2016. [Online]. Available: <http://developer.estimote.com/ibeacon/>. [Accessed: 01-Dec-2017].
- [24] E. Inc., “Estimote Cloud Settings Values.” [Online]. Available: <https://cloud.estimote.com/#/beacons/b238d205630607db4bb08ed5703fb201/settings>. [Accessed: 14-Sep-2018].
- [25] E. Inc., “Estimote Monitoring API.” [Online]. Available: <https://developer.estimote.com/android/tutorial/part-2-background-monitoring/>. [Accessed: 08-Aug-2018].
- [26] E. Inc., “Beacon Signal characteristics.” [Online]. Available: <https://community.estimote.com/hc/en-us/articles/201636913-What-are-Broadcasting-Power-RSSI-and-other-characteristics-of-a-beacon-s-signal->. [Accessed: 08-Aug-2018].

- [27] Revit, “Revit Features 2019.” [Online]. Available: <https://www.autodesk.com/products/revit/new-features>. [Accessed: 23-Sep-2018].
- [28] G. Hub, “Estimote,Android-SDK.” [Online]. Available: <https://github.com/Estimote/Android-SDK/releases>. [Accessed: 06-Jun-2018].
- [29] C. Looker, “free software license A*.” [Online]. Available: <http://www.codelooker.com/id/215/1123818.html>. [Accessed: 09-Aug-2018].
- [30] Pawelc, “A* code author.” [Online]. Available: <http://www.codelooker.com/blog/index.asp?username=pawelc>. [Accessed: 09-Aug-2018].
- [31] ARM, “Arm Products.” [Online]. Available: <https://www.arm.com/products>. [Accessed: 08-Aug-2018].
- [32] Android, “Lollipop.” [Online]. Available: <https://www.android.com/versions/lollipop-5-0/>. [Accessed: 08-Aug-2018].
- [33] S. Martinho, J. Ferreira, and R. Resende, “Find_Me: IoT Indoor Guidance System,” in *System Proceedings of International Conference on Ambient Intelligence*, 2018.