TALKING TO THE CITY: AN INTERNET OF THINGS PLATFORM FOR INTERACTION BETWEEN CITIZENS AND PUBLIC SPACES

CONVERSANDO CON LA CIUDAD: UNA PLATAFORMA DE INTERNET DE LAS COSAS PARA INTERACCIÓN ENTRE CIUDADANOS Y ESPACIOS PÚBLICOS

Trabajo Fin de Máster
Curso 2019-2020

Autor
YURI ALEXSANDER TAVARES PEREIRA

DIRECTOR
Rafael Caballero Roldán

Máster en Internet de las Cosas
Facultad de Informática
Universidad Complutense de Madrid

TALKING TO THE CITY: AN INTERNET OF THINGS PLATFORM FOR INTERACTION BETWEEN CITIZENS AND PUBLIC SPACES

CONVERSANDO CON LA CIUDAD: UNA PLATAFORMA DE INTERNET DE LAS COSAS PARA INTERACCIÓN ENTRE CIUDADANOS Y ESPACIOS PÚBLICOS

TRABAJO DE FIN DE MÁSTER EN INTERNET DE LAS COSAS
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

AUTOR
YURI ALEXSANDER TAVARES PEREIRA

DIRECTOR
RAFAEL CABALLERO ROLDÁN

CONVOCATORIA: SEPTIEMBRE 2020
CALIFICACIÓN: 10 (SOBRESALIENTE)

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

7 DE SEPTIEMBRE DE 2020

For those who lost someone important during this tough year of 2020.

ACKNOWLEDGEMENTS

I would like to thank my parents for their support and presence during the important moments of my life. If there is a personal stage like this being completed today, I recognize their efforts in the past to make my path possible. I would like to thank my brother and sister for their love and support expressed in many different ways. I would like to thank Helena Ramos for agreeing to embark with me on this adventure of living abroad and returning to university. These were days of love and companionship that could not be expressed in words. I would like to thank Professor Rafael Caballero Roldán for all the knowledge shared during classes, for the attention, patience and kindness of having accepted to guide me in the development of this project. I would like to thank all the friends who have collaborated in some way for the development of this work, especially Artur Cordeiro, and all the friends that Spain introduced me on this journey.

ABSTRACT

TALKING TO THE CITY: AN INTERNET OF THINGS PLATFORM FOR INTERACTION BETWEEN CITIZENS AND PUBLIC SPACES

This work presents the development of a technological platform for cities whose objective is to allow citizens and tourists to “communicate with it” in an interactive way, through voice commands and without a predetermined script. These characteristics allow users to guide their own learning process in order to obtain information and better understand the history of the cities where they live or visit. More than a strictly technological approach, the project invites the user to explore the different narratives about cities and to positively occupy their public spaces.

The platform consists of four technologies that work together: a web application, a mobile application, IoT devices and cognitive assistants that operate with Artificial Intelligence. Different concepts, programming languages and tools were combined to create the project, such as software development frameworks, databases, *Bluetooth Low Energy* protocol-based devices and cloud computing services. For this first version of the project, the idea is that interested users or institutions can add devices to the platform – and the mechanism for this is presented throughout the chapters of this work.

Keywords

Internet of Things, Smart Cities, Platform, Collaboration, Interaction, Artificial Intelligence, Cloud Computing, Bluetooth Low Energy, Beacon, Mobile Application, Web Application

RESUMEN

CONVERSANDO CON LA CIUDAD: UNA PLATAFORMA DE INTERNET DE LAS COSAS PARA INTERACCIÓN ENTRE CIUDADANOS Y ESPACIOS PÚBLICOS

Este trabajo presenta el desarrollo de una plataforma tecnológica para ciudades cuyo objetivo es permitir que ciudadanos y turistas "se comuniquen con ella" de forma interactiva, a través de comandos de voz y sin un guion predeterminado. Estas características permiten a los usuarios orientar su propio proceso de aprendizaje para obtener información y comprender mejor la historia de las ciudades donde viven o que visitan. Más que un enfoque estrictamente tecnológico, el proyecto invita al usuario a explorar las diferentes narrativas sobre las ciudades y a ocupar positivamente sus espacios públicos.

La plataforma consta de cuatro tecnologías que funcionan juntas: una aplicación *web*, una aplicación móvil, dispositivos *IoT* y asistentes cognitivos que operan con Inteligencia Artificial. Diferentes conceptos, lenguajes de programación y herramientas fueron combinados para crear el proyecto, como *frameworks* de desarrollo de software, bases de datos, dispositivos basados en el protocolo *Bluetooth Low Energy* y servicios de computación en la nube. Para esta primera versión del proyecto, la idea es que los usuarios o instituciones interesados puedan agregar dispositivos a la plataforma – y el mecanismo para esto se presenta a lo largo de los capítulos de este trabajo.

Palabras clave

Internet de las Cosas, Ciudades Inteligentes, Plataforma, Colaboración, Interacción, Inteligencia Artificial, Computación en Nube, Bluetooth Low Energy, Beacon, Aplicación Móvil, Aplicación Web

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 – Introduction

1.1 Ciudad Vestigio at a Glimpse

The project presented in this document, which it is going to be referred to as *Ciudad Vestigio* for identification purposes, is an idea that proposes the creation of a platform for cities with which its citizens and tourists can learn and get information about them in an interactive way. The project combines different technologies and concepts, such as *Internet of Things* protocols, web development frameworks, mobile development frameworks and Artificial Intelligence solutions from cloud computing services to create a remarkable experience of "talking to the city".

Imagine the situation: a young couple walks through a city discovering its unique spaces and suddenly finds a square they never saw before. As people interested in history and architecture, they love what they see in this newly found place and want to understand better its background of construction. To achieve this, the young couple picks up a smartphone, opens the *Ciudad Vestigio Mobile Application* and starts a conversation with the square, asking dozens of questions directly to it. The square, in turn, responds to them according to the questions asked, explaining details about its own history and memories of events that took place inside it. Once there is no script of subjects to follow, no "correct order" or even fear of making "dumb questions" to the city, the users can guide their own learning process, as in a conversation with a trusted friend. The example described along the last few lines uses a square, but it could be a building, a street, a statue or any other space of public interest. But how can a square, a building or other "inanimate element" in a city talk to a person? What is the mechanism used by *Ciudad Vestigio*?

The project uses a group of four different technologies that work together to carry out its main purpose, each one with specific details that are going to be presented throughout the chapters of this work. Just like the project, called *Ciudad Vestigio*, each of these technologies will be referred to by an individual title for identification purposes: the *Ciudad Vestigio Device*, the *Ciudad Vestigio Web Application*, the *Ciudad Vestigio Mobile Application* and the *Ciudad Vestigio VoiceBot*.

The first of them, the *Ciudad Vestigio Device*, is responsible for identifying the elements in the city with which it is possible to interact within the *Ciudad Vestigio* project. The device is a *Bluetooth Low Energy (BLE*[1]*)* module configured as a *beacon* [1] and for each interactive element in the city a corresponding device is required. The device is positioned next to the corresponding element we want to create an "interactive personality" and this device is configured to emit signals which are captured and properly interpreted by other of the four designed technologies: the *Ciudad Vestigio Mobile Application*. But before writing about this specific piece, it is important to mention that the idea behind the use of these devices is that they can be deployed collaboratively, which means that a citizen or an institution can follow the steps to configure a device and run it inside the platform, sharing knowledge about their own city with other citizens and tourists in a more horizontal process. This non-centralization is reflected in some important decisions of the project - which are going to be described soon - like choosing a *Bluetooth Low Energy* module that corresponds to a group of criteria.

The second technology that is part of the platform, which is strongly connected to this mentioned step of configuring a new device by the user, is the *Ciudad Vestigio Web Application*. Developed with the *Python*[2] programming in combination with the *Flask*[3] *Framework*, it is responsible for the registration of users who want to add a device in the platform, as well as for the registration and management of these devices, which are linked to their respective users/owners. The data registered with the *Ciudad Vestigio Web Application* is stored in a *MongoDB Atlas*[4] database and this data guide the performance of the third of the four technologies: the *Ciudad Vestigio Mobile Application*.

The *Ciudad Vestigio Mobile Application* is aimed at the user who wants to consume content from the platform, that is: the person interested in talking to the city in order to learn and get information about it. It is important to note that this user does not

[1] The use of "BLE" to refer to the Bluetooth Low Energy protocol will be recurrent in this work.

[2] https://www.python.org

[3] https://flask.palletsprojects.com

[4] https://www.mongodb.com/cloud/atlas

need to make a registration in the platform with the *Ciudad Vestigio Web Application* to start using it. The mentioned step is exclusive to institutions or citizens who want to add devices to the platform and share content. For the user at the other end of the process, the idea is that the step of consuming information should be the least bureaucratic and the most inviting as possible. Developed with the *JavaScript* programming language in combination with the *React Native*[5] *Framework*, the *Ciudad Vestigio Mobile Application* is responsible for scanning the environment periodically for devices registered in *Ciudad Vestigio* using the *Bluetooth Low Energy* protocol. If one of these devices is found and the user selects it, the application activates the fourth technology involved in the project: the *Ciudad Vestigio VoiceBot* that corresponds to the chosen device.

The *Ciudad Vestigio VoiceBot* is directly responsible for the feature of "creating a personality" attached to an inanimate element in the city, such as a building, a statue, a street, etc. Actually, as mentioned, these "personalities" are coupled with their respective *BLE* devices, allowing users to ask questions with voice and receive answers according to these questions also with voice, as in a conversation with a person. The core of the technology is created using *Watson Assistant*[6], a cloud computing service provided by *IBM*, whose details of implementation will be presented in chapter 6. In terms of usability, it is true that the *Ciudad Vestigio VoiceBot* is "officially" part of the *Ciudad Vestigio Mobile Application*, once it is only reachable from the other technology. However, it is also correct to affirm that the *Ciudad Vestigio VoiceBot* has a whole own dynamic of configuration behind it, what makes possible and reasonable to highlight it in this research as one of the four technologies that forms the *Ciudad Vestigio* platform[7].

---

[5] https://reactnative.dev/

[6] https://www.ibm.com/cloud/watson-assistant/

[7] Once the four technologies were presented, it is important to make a note about the use of the term "platform" in this work, since this expression will be used many times throughout the chapters. The use of the term "Ciudad Vestigio platform" as a way to refer to the project means, in this work, a group of individual technologies with individual purposes that work together in an "orchestrated way" to achieve a functionality that would not exist without any of its parts.

1.2 Motivation

It is not possible to write about the motivations behind this project without addressing my personal background, since many references are connected to the experiences I carry from my place of origin. I am from a Brazilian city called *Fortaleza*, where I lived for almost twenty-five years before starting my professional career in other contexts. *Fortaleza* is one of the main tourist destinations in *Brazil*, a place known for its beaches, natural landscapes, gastronomy and culture, but it is also a city with many serious problems, such as high social inequality [2] [3], poverty and high rates of violence [4] [5].

From all the hard and complex consequences of social inequality, one of them is reflected in the silent process of urban occupation of the city. With a very uneven concentration of wealth, it is possible to observe that the city "does not develop so much as a function of the real needs of its population, but rather as a function of the political and economic interests of the class or fraction of the class that holds power" [6]. As a consequence, the private action and the public action are closely interrelated in the production of space [6].

Under the arguments of "development" or "modernization", the city usually faces episodes of demolition of historic/iconic buildings from the most different periods for private economic reasons [7] [8], where little by little the memory of the city is buried without any reasonable public debate. Similar movements happen with environmental areas of economic interest [9] and decisions that impact poor families with dislodge [10].

On the part of us, citizens, how can we value our own history if we do not know it or do not teach it with the necessary importance over the generations? How can we collectively debate, protect and preserve our cultural heritage if we do not know the historical facts of our own space or even who are the people whose names inspire the title of our streets? The project presented with this document was created under this context of experiences that is specific, but that, unfortunately, is not exclusive to my hometown. It is also understood that this context is a much more complex scenario, addressed in a brief and personal way along the last lines.

During the first designing steps of *Ciudad Vestigio*, when analyzing possible positive impacts brought by its development, the process of imagining the target audience is also understood as a matter of motivation behind the project. From a first perspective, there are three main identified user groups. As a tool for promoting material and immaterial memory, all citizens are potential users of the project, being able to interact with their own city and learn about the space they collaborate to build daily. As a tool for cultural promotion, given the proper focus on tourism and local economy, all tourists interested in learning more about the city they visit are also potential users of the project. In a fictional context where the project is introduced as a local public policy and the most significant content is provided in an organized way by the public administration, the platform has the potential to amplify the experience of a tourist in the city and generate data that can guide strategic decisions for the benefit of the community. The version of *Ciudad Vestigio* presented in this work does not cover this data-driven approach, but the subject is addressed with more details in chapter 7, dedicated to future work. A third group of possible users that the project could bring a positive impact are from the educational field, connecting teachers and students with interdisciplinary outdoor activities about history, culture, heritage, urban planning, geography, etc. While schools can provide a different learning experience to younger students about their own city, the universities and its students from courses related to the subjects of the project can actively collaborate with the production of knowledge that will be consumed by citizens, tourists and other students.

This mentioned perspective of "involving the community" in the process of collaboration and production of content can be interpreted as "utopian" or "dreamy" at first sight, but this motivation is strongly connected with my professional background just before starting the *Master's Degree in Internet of Thing at Complutense University of Madrid (UCM)*. As one of the coordinators in *Fab Lab Livre SP* [11] [12] - project created by the *São Paulo City Hall* and the first Brazilian public policy based on open digital fabrication laboratories for any citizen - I had the opportunity to collaborate with the operation of a project that runs labs in twelve neighborhoods of a city like *São Paulo*, mobilizing people who never had contact with that kind of technology, teachers, students, hackers, community leaders, schools, universities, multinational companies and social institutions around collaborative projects and educational activities. Thousands of people are impacted by this public policy every year and it cultivates a universe of references that - if not necessarily adopted in this project - bring some kind of motivation, such as democratization of technology, *Collaborative Learning* [13], *Open Source* [14], *Do-It-Yourself* mindset and all values around the so-called *Maker Movement* [15].

1.3 Objectives

The main objective of this work is to demonstrate the development, operation and technical choices behind the creation of the project addressed in this introduction. The project, in turn, has as its main objective the creation of a platform for cities with which its citizens and tourists can learn and get information in an interactive way - as in a conversation - combining different technologies. This project must be capable of providing functionalities that match the purposes of two kinds of users.

The first kind is the user who wants to provide content on the platform, which means configuring a *Ciudad Vestigio Device* with its correspondent *Ciudad Vestigio VoiceBot* before placing the device around the city. This user can be an individual person or an institution that maintains an element of public interest, like a museum, a cultural center, a historic building, etc. The project must provide this user with the means - infrastructure and knowledge from the documentation presented in this work - to be able to perform these tasks. To achieve this objective, two technologies are developed and a third one is configured to be used. The first technology is the *Ciudad Vestigio Device*, a *Bluetooth Low Energy* module which is configured as a *beacon* [1] to emit the right signals and run properly in the platform. The steps involving this configuration are meant to be reproduced by the user with the same results described in this work. The second technology is the *Ciudad Vestigio Web Application* that works as the official website of *Ciudad Vestigio* and that is also responsible for the registration of users and devices. This application is managed by the project team. The third technology involved is the Ciudad Vestigio VoiceBot, whose core is created with *IBM Cloud* and its *Artificial Intelligence* based service *Watson Assistant*. This is a third-party service that the user must configure, deploy and attach to the respective *Ciudad Vestigio Device* during the registration process with the *Ciudad Vestigio Web Application*. This work must provide the user with the necessary information to achieve the same result presented as expected operation.

When designing the technologies to be used by this first kind of user in the project, the main objective is to make these steps as easy and as cheap as possible. From the chosen *Bluetooth Low Energy* module to the cloud computing platform, this objective is reflected in five criteria: the community of users has to be active for easy troubleshooting; the documentation (official or not) has to be vast; the price has to be cheap (or even free); the technology has to be easy to find, buy or acquire; and, lastly, the technology must be connected with the state of art in its field. Since collaboration is an important value for the project by design, the idea behind these criteria[8] is to turn the experience of running a new device in *Ciudad Vestigio* more friendly and inviting for this kind of user.

The second kind of user that the project must match the expectations is the user who wants to consume content from the platform in order to learn and get information about the cities. To achieve this objective, the *Ciudad Vestigio Mobile Application* is developed to let users scan devices registered in the platform that are nearby and start a voice conversation with them, if desired. When designing this technology, the main objective is to make this experience as easy as possible, where no registration is required and the user just needs to run the application to be able to use all the features inside it. This detail may not be completely explicit, but this search for easiness is also reflected in the choice of the *Internet of Things* protocol involved. The idea behind choosing the *Bluetooth Low Energy* protocol is that the users do not need to buy an external device to interact with the platform. They can use their own smartphone with a simple application installed to amplify the experience of exploring the narratives about the city.

Lastly, once defined the idea and the meaning of the project, one of the main objectives was to design an entire platform that would correspond to the initial motivation. This means that the project presented here is not just a mobile app, or just a *BLE* device, or just a web application or anything half of the way. The project proposes, from the first step to the last necessary step, all the individual technologies, protocols and

[8] In fact, these five criteria guide the decisions behind all the four technologies, as will be discussed over the next chapter.

concepts that work together to achieve a combined functionality that would not exist without any of its parts.

1.4 Plan of Work

The development of the project was divided into four parts, one for each technology that forms the *Ciudad Vestigio* platform. The sequence of development for these technologies followed the order of presentation made in this introduction, starting with the *Ciudad Vestigio Device* and finishing with the *Ciudad Vestigio VoiceBot*. In each technology, the development process consists of four stages: investigation of possible scenarios, study of the chosen tool, development of the solution and integration of the solution into the platform.

The objective of the first stage is to investigate the universe of tools and frameworks available to guide the decisions behind the development of the solution in question - in this case, each of the four technologies. The investigation process involves analyzing positive and negative aspects of the tools from different perspectives - from the five criteria presented in the last section to important details such as prior knowledge of the tool, level of difficulty and time of development. The most significant details of this stage are covered in the next chapter, which presents a "state of the art" discussion for each technology developed.

The second stage - the study of the chosen tool - aims to seek the necessary knowledge for the development of the solution. More than an abstract and conceptual research of the tool, it is a study process very directed to the development of the solution within the time available and necessary quality. This research is done in books, videos and other types of documents.

Based on the knowledge obtained, the third stage - the development of the solution - aims to develop the technology in question within the established objectives and matching the expectations of each user presented. The most significant details of this stage for each technology has an individual chapter: 3, 4, 5 and 6.

The last step - the integration of the solution to the platform - aims to promote the "coupling" of newly developed technologies to the platform so that they can work in combination with the others. This step is not always necessary for all technologies and the most significant details are also covered in chapters 3, 4, 5 and 6 when applicable.

This work presents a fully functional version of the project - with all technologies working as expected and presenting the expected results - but it is understood that there is still a lot of work to be done. In this context, chapter 7 presents a series of different possible improvements that should guide future work to enhance the objectives behind the project, which are intended to be implemented in the near future. Lastly, chapter 8 presents the conclusions drawn from the entire project development process.

The codes developed for this version of the project are available at https://github.com/vstgio/cvestigio. They are licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Chapter 2 – State of the Art

This chapter proposes a discussion about the context of theories and available technologies that support this project and the decisions made about which paths to follow in terms of viability. It is divided into two parts. The first one presents projects that, in some way, dialogue with the purposes of *Ciudad Vestigio*. As it is a project that combines different technologies in a very particular arrangement, it is not always possible to present other projects with solutions that can be considered similar and comparable in an equivalent way. But that is not the goal either. The idea is to present technological projects that have "points of contact" with *Ciudad Vestigio* - whether for thematic or technological reasons - and discuss how these solutions were implemented, what are their strengths and weaknesses, as well as other details that can contribute to the development of *Ciudad Vestigio*. The focus will be on those projects that intersect technology, city and participation, preferably within the context of *Internet of Things*. No photos will be added to this document to illustrate the selected projects, but it is strongly recommended to visit each one's official webpage to learn more about these very creative projects. Four projects from different locations were selected: one from *Europe*, one from *Latin America* and the other two from *North America*.

The second part of the chapter is denser and proposes an overview discussion about the state of the art for each one of the four technologies that form the *Ciudad Vestigio* platform - the *Ciudad Vestigio Device*, the *Ciudad Vestigio Web Application*, the *Ciudad Vestigio Mobile Application* and the *Ciudad Vestigio VoiceBot*. In general, this second part discusses what options of languages, tools and frameworks are available to be used, what advantages and disadvantages exist, what decisions were made and how they fit the criteria presented in the introduction of this document. For organizational purposes, this second part of the chapter has four sections - one for each technology - and each section in turn has two subsections. The first one discusses the universe of available tools and contextualizes the reasons behind the decisions made for the specific technology. The second subsection focuses on the technical details of the tool chosen to develop the technology in discussion.

2.1 Related Projects

It is not possible to provide an overview of initiatives that have some kind of connection with *Ciudad Vestigio* without mentioning a project developed in *Brazil* in 2017 called *A Voz da Arte*[9]. In the year that the company *IBM* celebrated 100 years of operation in the country, the project was carried out at *Pinacoteca de São Paulo*[10] - one of the most important art institutions in *Brazil* - and used cognitive computing to make the user experience within the museum more interactive and personalized. "The company created a cognitive assistant that answers questions from visitors about seven works of art from the *Pinacoteca* collection. The guided tour with *IBM Watson* technology was open to the public", explains one of the official texts of the initiative, which was created as an advertising campaign signed by the *Ogilvy* agency [16] and was reported by important Brazilian magazines and television programs [17] [18].

Upon entering the building of *Pinacoteca de São Paulo*, the visitor received headphones connected to a smartphone with the specific application of *A Voz da Arte* already running. When walking through the space and approaching one of the seven works that were part of the project, the application was programmed to change its state and to allow the visitor to talk to the piece of art - asking questions and receiving answers without a specific script of subjects [19]. From the projects presented in this section, this is the only one whose existence was known before starting the development of *Ciudad Vestigio* and the initiative brings some important references. However, until then, no technical details behind the project were known that went beyond the mobile application presented to visitors during the exhibition.

A text written by an *IBM Technical Leader* on the social network *LinkedIn*[11] clarifies how the project works [20], but without many technical details. The most interesting information for the context of *Ciudad Vestigio* is to confirm that the development team

[9] The Voice of the Art, in free translation.

[10] https://pinacoteca.org.br

[11] https://www.linkedin.com

also used *Bluetooth Low Energy modules* as *beacons*[12] to activate changes in the state of the mobile application, a detail that during the experience of visiting the exhibition was not explicit. The text does not provide any model specification or other details about the *beacon* device used, nor about how the mobile application was developed.

It is unfortunate that this project with great potential did not went beyond its advertising function and became a real initiative within *Pinacoteca de São Paulo*. A detail of this project that was not intended to be reproduced at *Ciudad Vestigio* is the fact that it was not possible to download the mobile application on a visitor's smartphone, since the app was only installed and available on devices belonging to the museum. In other words, to experience the initiative in a day with many visitors, it was necessary to wait for a device to be available.

A positive experience of this project that is intended to be reproduced in future versions of *Ciudad Vestigio* is the possibility to capture feedback from users in order to improve the quality of the content - both to evaluate responses given by the platform and to understand where there are voids of unanswered questions. This issue is discussed in chapter 7 (about future work) and the strategy for improving the content of the project is discussed in the conclusions of this work.

A second initiative that dialogues with the ideas behind *Ciudad Vestigio* is the *Hello Lamp Post* project, developed in its first version in 2013 in *Bristol*, *England*. The project proposes the creation of a "playful platform that lets you chat to street objects" [21]. This interaction with objects on the street can be done through text messages via *SMS* [22], *Facebook Messenger*[13] or *WhatsApp*[14]. The use of two famous applications worldwide as communication tools for the project is a very interesting detail, but the fact that it is not possible to use voice commands is understood as a feature that could be added.

---

[12] This subject will be covered in the next section.

[13] https://www.facebook.com/messenger

[14] https://www.whatsapp.com

One of the most interesting details of this project is the fact that it uses the unique identification codes that public objects[15] have for administration purposes as part of their solution. In other words, the initiative uses the city's existing infrastructure without the need to add devices or other technological elements to the process. However, there is no guarantee that other cities and public administrations around the world use the same system (or similar) as in England. For this reason, the use of *IoT* devices is better suited to the context of *Ciudad Vestigio*, once the idea is to create a platform that can be used in any city, regardless of the country.

From a practical point of view, when walking through the city and finding one of these objects, the user can send a message with a model predetermined by the platform to "wake up" the object. From that moment the conversation begins. An interesting feature of the project in its first version [23] - which ran exclusively with *SMS* technology using *Twilio* [24] - is the possibility for users to interact with each other through the object, leaving messages and information for those who will interact with the object later. In its most recent version, the official webpage of the project points out that the users can now learn about the area in which the object is located and share opinions about the city from questions asked by the objects. "Insights from interactions are anonymously shared with organizations such as local authorities to enable better informed decision making", explains the website. This is an interesting approach when considering the use of the project as a public policy tool, respecting by design the users' privacy.

The third initiative presented is the *Amateur Intelligence Radio* project [25], created by the Canadian art and design studio *Daily tous les jours* in 2014. The project's connection with *Ciudad Vestigio* is in the idea of "giving life" to elements considered inanimate, in this case the building of *Union Depot Station* in the city of *Saint Paul*, *United States*. One of the most creative details of this initiative is the fact that the building is the host of its own radio show, which the visitors can hear in different places of the station that were projected for this purpose.

[15] Such as poles, fire hydrants, bus stops, mailboxes, etc.

> *Buildings are not only made of bricks. They are also made of the emotions and stories of the people who lived in them. Amateur Intelligence Radio (AIR) is a "quasi-artificial" intelligence inhabiting a historical building. Contrary to the robotic voices heard in many transitory public spaces, the host of this installation transforms the environment by expressing its feelings in a daylong radio show.* [25]

About the radio program presented by the building, an interesting aspect is the fact that it is created dynamically with real-time data, such as weather conditions, the number of people in the room (captured by a camera and processed with *Computer Vision* [26]), the trains' arrivals and departures, local news and even the mood of the building [25]. Station visitors and program audiences could interact with the system through a website. "Whether it is to send a declaration of love, an apology, a birthday wish, or give advice, each participation influences the content and personality of the radio", explains the webpage of the studio [25]. In the context of *Ciudad Vestigio*, the use of these different external data to create the radio program is one of the most powerful features within the *Amateur Intelligence Radio* project. This approach could be used, for example, to compose the responses of the interactive elements in the city in a more dynamic way.

The fourth and last of the projects presented has a connection with *Ciudad Vestigio* in the sense that both initiatives aim to disseminate information about the city and its places in order to strengthen the idea of community. Called *Soofa* [27], the startup founded in 2014 in the *United States* proposes a "neighborhood news feed that connects a community with screens everyone can see and anyone can use". Large screens powered by solar energy and developed with the "electronic paper" technology [28] are spread throughout the city, working in combination to distribute local information to pedestrians about the neighborhoods where they are. This information ranges from cultural agenda events to traffic data, weather, public information, etc. Users can interact by publishing content - reviewed by the project team - through an application called *Soofa Talk* [29] and companies can publish on the platform through a subscription plan. In the context of the *SARS-CoV-2* pandemic, the startup worked with the city of

Boston, in the *United States*, and other cities around it to provide governmental guidance on the situation to citizens [30].

2.2 Analysis of each technology used

Once the projects were presented, the chapter enters its second part. This part is divided into four sections – one for each technology of the platform: the *Ciudad Vestigio Device*, the *Ciudad Vestigio Web Application*, the *Ciudad Vestigio Mobile Application* and the *Ciudad Vestigio VoiceBot*. Each of these sections in turn are divided into two subsections. The first subsection provides an overview of possible scenarios for developing the technology. It aims to discuss questions like which tools exist that could be used to implement the solution; which are the positive aspects of them; which are the negative aspects of them; how they fit the criteria presented in the introduction of this work; and what were the reasons for using the selected tool. The second subsection focuses on the specific tool chosen for developing the solution, providing a technical description of its features that are important in the context of the project. The only technology that differs from the model of two subsections is the *Ciudad Vestigio Web Application* because it also promotes a discussion about the chosen database system.

2.2.1 The Ciudad Vestigio Device

2.2.1.1 Available technologies, possible scenarios

Once defined the purpose of the project and the mechanism for the operation of *Ciudad Vestigio* in abstract layers, several technical questions begin to appear. As it is a project that arises from academic research on *Internet of Things* and its multiplicity of available protocols, the first one is: which communication protocol to use? There are many options and acronyms competing for space and market today, such as *Wi-Fi, BLE, LoRa, Sigfox, MQTT, 6LowPAN, CoAP, ZigBee*, etc.

The answer to that first question is found quickly when we think about the end user of the project: that person who wants to learn more about the city where he/she lives or the city he/she is visiting. The first criterion for choosing a protocol was to reject those that would require the user to purchase equipment just to run that specific protocol on the project. The idea is to make the experience of using the platform easy and inviting, within the feasibility criteria. In this context, the smartphone - which many of us already carry for other reasons – appears as the most interesting option as interaction tool with the platform and narrows the scope of protocol possibilities.

Unlike projects that offer virtuality experiences to their users - such as, for example, museum exhibitions that are adapted in many different ways to the digital environment - *Ciudad Vestigio* proposes an invitation to positively occupy the public spaces and get to know its elements in person. This means that part of the "fun" in using the platform is discovering these spaces around the city, either deliberately or when the user least expects them during a walk. In this context, the use of the smartphone should be conditioned to this experience of "adventure through the city" and interaction with it, which leads us to a second question: how to concretize this mechanism from the technological point of view with the possibilities that smartphones in general offer?

One of the most famous technologies in smartphones, which deals exactly with the issue of positioning and is widely used in the many different services is the *GPS*, an acronym for *Global Positioning System*. The *GPS* is a satellite-based navigation system that was developed by the *United States Department of Defense (DoD)* in the early 1970s [31, p. 1]. Although it was originally designed as a military tool, it has revolutionized the navigation field with different civil applications. Decades later, it is almost impossible to think about our lives today without certain location-based services, such as ordering a taxi with an app or using a map application to find a destination in the city.

The *GPS* provides continuous positioning and timing information, anywhere in the world under any weather conditions [31, p. 1]. Its operation is possible thanks to a group of satellites - at least 24 operational satellites [32] - that transmit radio signals. For the system to work, every satellite has very accurate clocks on board and it is necessary to know the position of each of them very precisely [33, p. 653]. The satellites send an electromagnetic signal and the user has a receiver equipped with a clock, in this case the smartphone. The position of the satellites is known at each instant of time and the position of the receiver is not. In the ideal case, three satellites are sufficient to locate the receiver, but it is not exactly the case [33, p. 654].

> *However, to achieve this, the clock at P would have to have the same quality as those carried aboard the satellites, and this would be impossible because such clocks are cumbersome and very expensive. A fourth satellite is thus needed to make up for the inaccuracy of this clock.* [33, p. 654]

The basic principle of *GPS* exploits the intersection of spheres, where each sphere has a radius of one receiver-satellite distance and is centered on the particular satellite that the smartphone receives the signal. The smartphone will be located at the intersection of these three spheres with a fourth satellite to account for the receiver clock offset [31, p. 8].

In the context of *Ciudad Vestigio*, one of the main positive details of using the *GPS* technology would be that there is no need to add devices or other resources around the city to make the platform work. With a simple mobile application that knows in advance where the interactive elements are located in the city, it is already possible to trigger communications between users and elements when necessary. However, given the expectations that the users can “discover elements” around the city, *GPS* poses the problem of not always having the desired accuracy. In general, *GPS*-enabled smartphones are accurate to within 4.9 meters, which is a totally acceptable value for the project's needs. The problem is that multiple external factors can influence this accuracy such as signal blockage (due to buildings, bridges, trees, etc.), atmospheric conditions, "multipath" (when signals are reflect in buildings or walls) and the receiver quality [34]. From a practical point of view, it may happen that the user is facing an element of the city with which he/she wants to interact and the application points out other elements because the *GPS* of the smartphone is indicating a different location. This accuracy also decreases within closed spaces [34] and the idea is that the functionalities of *Ciudad Vestigio* cannot be limited to open spaces only. A large train station can have several different spaces and objects of public interest, for example, which means that the use of the platform inside closed spaces is within the scope of possibilities, if necessary.

For these reasons, the *GPS* technology was not selected for this specific purpose within the project, but, as will be discussed in chapter five, it has a very important function within the logic of the *Ciudad Vestigio Mobile Application*.

Another common and highly available technology in smartphones is *Wi-Fi* technology. The *Wi-Fi* is a wireless networking technology that allows devices such as computers, smartphones and other gadgets to interface with the internet. "On the technical side, the IEEE 802.11 standard defines the protocols that enable communications with current Wi-Fi-enabled wireless devices, including wireless routers and wireless access points" [35]. The internet connectivity occurs through a wireless router. A wireless router is sometimes referred to as a wireless local area network (WLAN) device and a wireless network is also called a *Wi-Fi* network. Given the demands of the *Ciudad Vestigio* project, one of the advantages of using *Wi-Fi* is the high availability of wireless routers and access points inside buildings - whether public or private - which would facilitate the use of the platform in situations like this. Problems arise when we think about other elements of interaction that are not buildings, such as squares, streets or even urban objects, such as statues, for example. In these situations, the implementation of wireless routers or access points becomes complicated and disproportionately expensive for the purpose wanted to be achieved.

A third technology that is easy to use and highly available on smartphones is the reading of *QR Codes* from the device's camera. The term *QR* means "Quick Response" and the first *QR Code* system was implemented in 1994 by the Japanese company *Denso Wave*, a subsidiary of *Toyota*, to accurately track vehicles and parts during manufacturing [36]. The first mobile phones containing built-in *QR* readers were marketed in *Japan* in 2002 and currently the technology is used in different contexts and for different purposes, such as online account authentication [37], to send and receive payment information [38], etc.

The *QR Code* stores data like a barcode and these data can include website URLs, phone numbers, or up to 4,000 characters of text. The three large squares at the edges of the code are important to identify the element as a *QR Code*. The reader then analyzes the code "by breaking the whole thing down to a grid. It looks at the individual grid squares and assigns each one a value based on whether it is black or white. It then groups grid squares together to create larger patterns" [36]. The patterns within *QR Codes* represent binary codes and they are interpreted to reveal the data stored by the code.

The advantage of using this technology in the context of *Ciudad Vestigio* is its simplicity. To the user who wants to make content available within the project, it would be enough to publicly present a code generated by the platform, without the need to configure and add devices to the city space. A printed sheet of paper would already work. On the other hand, this would significantly transform the user experience initially designed for the person who wants to interact with the city, since it would be necessary to find the code to be scanned before triggering the interaction (instead of approximation). In the context of a street with many visual stimuli, a massively busy tourist spot or a very large space, the task of trying to find the code - or to know about its existence - can become a negative point. Another very important issue - addressed in chapter seven - is the intention to improve the usability of *Ciudad Vestigio* to include features for visually impaired people. In this context, the use of a resource such as *QR Code* that demands a visual dynamic to find the code and to point the camera at it is not in accordance with the guidelines for the future of the project.

The fourth technology to be presented - and chosen as the ideal solution for this project - is the *Bluetooth Low Energy (BLE)* protocol. The *BLE* started as part of the *Bluetooth 4.0 Core Specification* [39]. Despite sharing a name with classic *Bluetooth*, *BLE* has an entirely different lineage and design goals [40, p. 1]. The two wireless communication standards are not directly compatible and *Bluetooth* devices from any specification version prior to 4.0 cannot communicate with a *BLE* device. The *BLE* standard was introduced in 2010 and in 2016 the *Bluetooth 5* was released [41]. At the time of writing this work the *Bluetooth Core Specification* is in version 5.2 [42].

The *Bluetooth Low Energy* uses the same frequency range as the classic *Bluetooth*, from 2.4000 to 2.4835 GHz, but this range is divided into 40 channels[16] of 2 MHz: 37 of them are used for connection data and the last three (37, 38 and 39) are used as advertising channels to set up connections and send broadcast data [40, p. 16]. The modulation

[16] Instead of the 79 channels of the classic version.

rate of *BLE* radio is set by the specification at a constant 1 Mbps[17], but this is considered a theoretical upper limit. In actual terms, this limit is lowered by a variety of factors.

> *So, in practice, a typical best-case scenario should probably assume a potential maximum data throughput in the neighborhood of 5-10 KB per second, depending on the limitations of both peers. This should give you an idea of what you can and can't do with Bluetooth Low Energy in terms of pushing data out to your phone or tablet and explain why other technologies such as Wi-Fi and classic Bluetooth still have their place in the world.* [40, p. 8]

Although 10 KB/s is not considered a high value for a world where everything becomes faster every day, this is more than enough to use with common wireless gadgets and for the purposes of the project. In terms of operating range[18], it depends on different factors, from environment to device orientation. The transmission power is typically measured in dBm and it is possible to create and configure a *BLE* device that can reliably transmit data 30 meters or more, but it consequently consumes more energy and reduces the lifetime of a battery cell [40, p. 8]. In a generic context, a typical operating range with *BLE* is "probably closer to 2 to 5 meters, with a conscious effort to reduce the range and save battery life without the transmission distance becoming a nuisance to the end user" [40, p. 8]. For this first version of *Ciudad Vestigio* this work assumes that the users and institutions have conditions to power the devices into the outlet[19], protected from unauthorized human manipulation and weather variations. The solution presented in the next chapter starts from this premise and shows how to properly power the chosen device

[17] The *Bluetooth* 5 allows the physical layer to operate at 2 Mbps [195].

[18] The documentation of *Bluetooth* 5 claims it has four times more range than the previous version [195].

[19] This is easily achievable for devices positioned in buildings. Even in contexts of devices positioned outdoors (in a square for example), the public administration can use the infrastructure of an available lamppost to position the device.

into the outlet. This brings the advantage to increase or decrease the signal strength according to the needs of the situation, without having to worry about the battery life.

A *Bluetooth Low Energy* device can communicate with the outside world in two ways: connecting to other devices or broadcasting data without a connection. This is a very important detail to understand the solution used in *Ciudad Vestigio*. If the project needs a periodic data transmission between two devices in both directions, a connection will be necessary. Connections involve two separate roles: a central device and a peripheral device. To establish a connection, a central device first starts scanning to look for advertisers that are currently accepting connection requests. The advertising packets can be filtered by *Bluetooth* address or based on the advertising data itself. When a suitable advertising peripheral is detected, the central device sends a connection request packet and, with the response, the connection is established.

The roles do not impose restrictions and data can be sent by both devices during each connection event. In version 4.0 there was a restriction that peripheral devices could have only one single connection to a central device. This was removed in version 4.1 [40] and now there are much more freedom: a peripheral device can be connected to multiple central devices; a central device can be connected to multiple peripherals; and a device can act as a central and as a peripheral at the same time. The use of connections is the mechanism behind the communication between a notebook and a *Bluetooth* speaker when we listen to music, for example, or a smartphone and a gadget such as a smartwatch.

While most devices advertise to get a connection established, there are situations where they only need to advertise data. This is the case of a project in which a device needs to send data to many others at the same time. This is only possible with advertising packets, since the data sent and received when a connection has been established is only visible to the participants involved.

This example is exactly the case of *Ciudad Vestigio*, since the devices positioned around the city do not need to establish connections with other devices. They just need to advise their presence to the scanning smartphones with the *Ciudad Vestigio Mobile Application*, which in turn knows everything it needs to do in the presence of that specific device. This is the second mentioned way a *BLE* device can communicate with the outside world: using connectionless broadcasting, a good choice when it is necessary to push only a small amount of data on a fixed schedule to multiple devices. There are two roles involved: the *broadcaster* that "sends non-connectable advertising packets periodically to anyone willing to receive them"; and the *observer* that "repeatedly scans the preset frequencies to receive any non-connectable advertising packets currently being broadcasted" [40, p. 9].

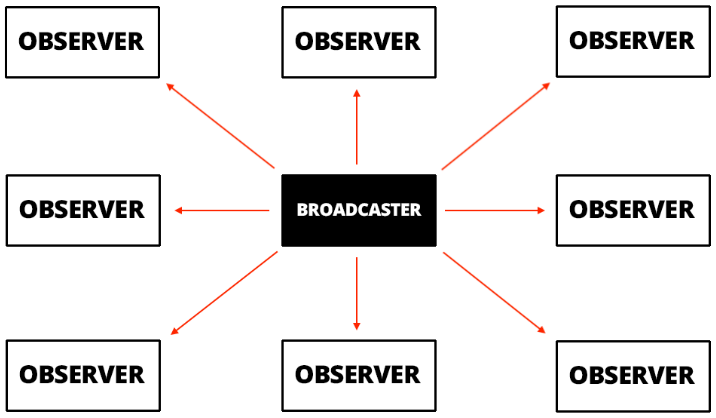


Figure 2-1. Representation of the broadcast topology. The communication is unidirectional.

The idea then is to broadcast data out by taking advantage of the advertising features of *BLE*. Since we are not working with sensitive or private data, the model meets the demands of the *Ciudad Vestigio* project. The standard advertising packet established by the protocol has 31 bytes of payload, used to send data about the device and its capabilities. However, this space can also be used to include custom data. If this 31-byte standard is not enough, *BLE* supports a secondary packet called *Scan Response*. It allows a device (*observer*) that have detected a *broadcaster* to request a second advertising packet with another 31 bytes of payload. In the case of *Ciudad Vestigio* this feature is not necessary and will not be used.

Now that the characteristics of the *Bluetooth Low Energy* protocol are known and the model used to communicate data with other devices has been presented - unidirectional broadcasting - it is important to add another very important concept in the context of the *Ciudad Vestigio* project: *beacons*. *Beacons* are *BLE* devices like any other, but with the specificity of being aimed at working only with the broadcasting function. For this, they are configured to operate with a specific protocol. *Beacon* technology was introduced with *iBeacon* by *Apple* [1], a simple protocol that allows *BLE* devices to transmit very small bits of data. The technology was introduced with the *iOS 7* (2013) and the official documentation for its 1.0 version is from June 2014 [43]. A year later, in 2015, *Google* came out with its own protocol for proximity beacons called *Eddystone*[20]. Since then, both protocols have played an important role in the universe of beacons.

Based on this overview of characteristics, the *Bluetooth Low Energy* protocol was the technology chosen to develop the *Ciudad Vestigio Device* solution. As discussed over the last few pages, it is a protocol that has been around for a decade and has a strong daily presence in the universe of smartphones. The protocol allows the user to adjust the signal strength of the devices, an important feature to adapt the solution for different contexts - indoor or outdoor - and to guarantee the experience of "discovering the city" when approaching elements registered on the platform. In addition, the communication model based on advertising packets is ideal for the needs of the project, making it possible to use specific protocols to perform this function - such as *iBeacon* and *Eddystone* - developed by two big companies like *Apple* and *Google*. It is a protocol based on low energy consumption, which - although not covered in this work - is important for contexts where it is not possible to plug the device into the outlet. The devices in general are not expensive and are configurable in a relatively simple way if the necessary information is made available to the user - which is intended to be done with *Ciudad Vestigio*. And once the devices were mentioned, a final technical question arises: which device to use as a solution for the *Ciudad Vestigio Device*?

2.2.1.2 The HM-19 BLE Module

When investigating the *BLE* device to be used as standard in the *Ciudad Vestigio* project, some assumptions helped to guide the decision. As previously discussed, the

[20] https://developers.google.com/beacons

project does not require a device that does data processing, since its main function is to inform interested observers (the smartphones) about its presence. Once the data transmitted by the devices is not private or sensitive from the security point of view - after all, the project is using the broadcasting model – it allows us to direct the search to lower cost devices.

In the universe of rapid prototyping, there are famous development boards based on *ESP32*, the microcontroller unit (MCU) created by the Chinese company *Espressif Systems* that has *Wi-Fi* and *Bluetooth* connectivity [44]. At the time of writing this work, a development board like this can be purchased online with sellers located in *Spain* for around ten euros and a quick search reveals that it is possible to configure it to work with the *iBeacon* protocol. However, we would be using a board that is "too powerful" for the specific function it needs to perform in the project. There are options available that are more in line with the demands of *Ciudad Vestigio*.

One of them, the one chosen for the project, is the *HM-19* module produced by the Chinese company *Jinan Huamao Technology*. The device is part of a set of low cost *BLE* modules that are famous in the universe of prototyping with microcontrollers, especially in the context of development with *Arduino*. This is an important detail due to the large amount of documentation available online to configure the device, which includes information for troubleshooting. The module is based on the *CC26040R2F MCU* from *Texas Instruments* [45] - which means it supports version 5.0 of the *BLE* - and costs an average of nine euros with online vendors located in *Spain*. It is a very small module (1.8 cm x 1.3 cm x 0.2 cm) with a signal range of up to 100 meters according to its documentation. The *HM-19* operates with 6.5mA in active mode and from 50uA to 1200uA in sleep mode. The steps on how to set these details will be covered in the next chapter[21]. The module has support to operate with the *iBeacon* protocol and one of the most important things to know about it is the format defined for the advertising packets. An *iBeacon* advertisement provides the following information via *Bluetooth Low Energy*:

[21] Chapter 3 opens a series of chapters that explain the implementation details for each technology that is part of the *Ciudad Vestigio* platform.

FIELD	**SIZE**	**DESCRIPTION**
UUID	16 bytes	Application developers should define a UUID specific to their app and deployment use case.
Major	2 bytes	Further specifies a specific iBeacon and use case. For example, this could define a sub-region within a larger region defined by the UUID.
Minor	2 bytes	Allows further subdivision of region or use case, specified by the application developer.

Table 2-1. List of the data stored by an iBeacon protocol packet. The next chapter presents the configuration of each of these fields in the Ciudad Vestigio Device. The data in this table were taken from the official document of version 1.0 of the protocol.

The *UUID*, *Major* and *Minor* values provide the identification data for the *iBeacon* device. "Generally speaking, this information is hierarchical in nature with the major and minor fields allowing for subdivision of the identity established by the *UUID*", explains the protocol document [43]. According to the previous table, in the context of a project like *Ciudad Vestigio*, the guideline is to have a default *UUID* to identify devices that are part of the platform. The other values, *Major* and *Minor*, are used to organize them into the correct subdivisions.

The term *UUID* stands for *Universally Unique Identifier*. It contains 32 hexadecimal digits split into five groups ("00000000-0000-0000-0000-000000000000"). Characters should be numbers from “0” to “9” and letters from “A” to “F”. A group can be made entirely of just numbers or letters or a combination of both [46]. The *Minor* and *Major* values are unsigned integer values between 0 and 65535.

These values presented are not determined by the creator of the protocol, but by whoever is deploying the *beacon* device. That is: *UUID*, *Major* and *Minor* values are not registered with *Apple* and there is no centralized management of these values. In the context of *Ciudad Vestigio*, the management is done entirely by the platform and the user or institution interested in adding devices to the project must use the values provided. The calculation of this value is presented in chapter 4, which explains in detail the implementation of the *Ciudad Vestigio Web Application*.

2.2.2 The Ciudad Vestigio Web Application

2.2.2.1 Available technologies, possible scenarios

As presented in the introduction, the *Ciudad Vestigio Web Application* is responsible for the registration of users, devices and their respective cognitive assistants inside the platform. It is important to mention that this is not a one-way process, since the *Ciudad Vestigio Web Application* returns to the user the *Universally Unique Identifier (UUID)* of the device within the platform. This *UUID* must be used by the user to configure the *HM-19* as a *Ciudad Vestigio Device*. This is a very important step for the project because all the data registered by the *Ciudad Vestigio Web Application* - including the generated *UUID* - is stored in a database, which will later be consulted by the *Ciudad Vestigio Mobile Application*. With this data in hand, the mobile solution knows how to interact with the *BLE* signals received and it knows how to reach the correspondent *Ciudad Vestigio VoiceBot* of a specific device when necessary. The details of this mechanism are discussed in chapters 4 and 5.

When analyzing the main tasks to be performed by the *Ciudad Vestigio Web Application*, it is possible to affirm that the development of this software can be done in several ways using different programming languages and different frameworks for web development. After all, the core of the system are CRUD operations [47] aimed at registering users and devices[22], all in combination with a chosen database. A number of precautions need to be taken regarding issues such as security, but these topics are very well addressed by different options of available frameworks. Regarding them, there are strong communities behind languages like JavaScript, Python, Ruby, PHP, using tools like Express, jQuery, React, Angular, Vue.js, Django, Flask, Rails, Laravel, Drupal, etc. The field of web development nowadays is a universe of tools, knowledge and possibilities in constant transformation and innovation.

Virtually all of the mentioned options are in accordance with the criteria set out in the introduction of this work - strong community, large documentation, low price, high availability and connection with the state of the art in software development. The

[22] The calculation of the *UUID, Major and Minor values* is part of this process.

decision on which programming language and framework to use was strongly influenced by the criterion of previous personal experience with the tool. The recognizable benefit for the project and main argument is the speed of development and consequent gain of time when using a technology already known. It is important to remember that this project involves the development of four completely different technologies and time optimization is an important element throughout the whole process. In this context, the choice was between the *Python* based framework *Flask* in combination with a MongoDB database, or the *JavaScript* based so-called *MERN* stack [48] - a combination of *MongoDB*, *Express*, *React* and *Node.js* working together. The final decision was to use the *Flask* framework.

A common and important detail in both options presented is the presence of *MongoDB* as the chosen technology for the database. As with the framework for web development, the choice is also based on previous personal experience with the technology, but more important in this case is the connection between the demands of *Ciudad Vestigio* and what *MongoDB* provides in terms of functionalities and philosophy. As a non-relational database, the most important feature for the project in this first prototype and proof of concept stage is the flexibility provided. In the context of agile development where the project can often change quickly, the absence of up-front commitments to data models - as could happen with other solutions - and the possibility of redesign certain pieces of the database with no big side effects is a very important resource. This and other characteristics are discussed in more depth in the following sections of this chapter. Another important feature related to the flexibility of non-relational databases such as *MongoDB* is the ability to handle many different kinds of data [49] in the same document. As will be addressed in chapter 4, this detail is connected to the demands of *Ciudad Vestigio* because a device entry in the database gathers different data types such as strings, numbers, images, objects, etc. When both mentioned features are combined, even in a future context where more data starts being collected by the project, the database can absorb these new data points properly [49]. This is an important resource in the age of *big data* [50]. Lastly, another important feature in favor of *MongoDB* as the chosen database is the ease to perform geospatial queries [51]. This is done by creating *2dsphere Indexes*, as will also be described in chapter 4 [52]. In chapter 5 this work describes how the *Ciudad Vestigio Mobile*

Application uses the device's location coordinates to get only nearby devices from database.

2.2.2.2 Flask

Flask[23] is a lightweight web application framework. It is designed to make the process of creating an application easy and quick, with the possibility to scale it up to more complex stages [53]. With just a couple lines of code it is possible to create a basic application:

```
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

Figure 2-2. A basic example of Flask application

The previous code was saved in a file called "hello.py" and - with at least *Python* and *Flask* installed - it is possible to run this file in a terminal window with the first line below:

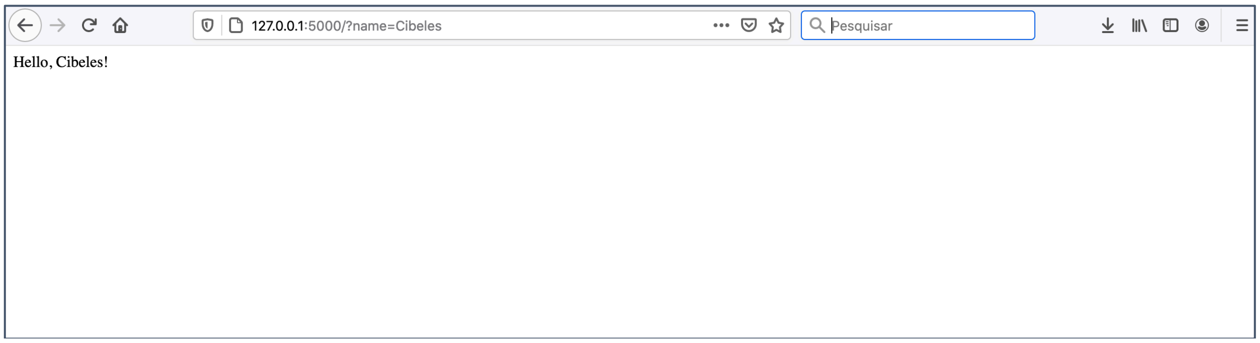
```
$ env FLASK_APP=hello.py flask run

 * Serving Flask app "hello.py"
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 2-3. How to run the example code in a terminal window and the respective response

The response to the command shows that the application is running at "http://127.0.0.1:5000". If we try to access the address through a web browser, we will see

[23] https://flask.palletsprojects.com

a "Hello World!" in the upper left corner of the screen. If we try to access the same address by sending a value to "name" - like "http://127.0.0.1:5000/?name=Cibeles" - we get a different result.



```
Hello, Cibeles!
```

Figure 2-4: The result of accessing "http://127.0.0.1:5000/?name=Cibeles" from a web browser

When analyzed within the context of web development, an interesting detail about *Flask* is the common use of the term "microframework" to refer to it. The subject is discussed in the official webpage of the project:

> *"Micro" does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The "micro" in microframework means Flask aims to keep the core simple but extensible. Flask won't make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don't.* [54]

The "decisions it does make" mentioned by the text are related to the fact that *Flask* bridges to *Werkzeug* to implement a *WSGI*[24] application and to *Jinja2* to handle

---

[24] The term *WSGI* stands for *Web Server Gateway Interface* and it is a "specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request" [190]. *WSGI* is a *Python* standard described in *PEP 3333*, the "Python Web Server Gateway Interface v1.0.1" [191].

templating [55]. *Flask* also binds to a few common standard library packages, such as logging, but everything else is up for extensions. In other words, the idea behind *Flask* is to build a "good foundation" for all applications without forcing preferences. The specific requirements, decisions about design or advanced patterns that a developer wants to make need to take advantage from the extensions [55].

Since *Werkzeug*[25] and *Jinja2*[26] are core elements for *Flask*, it is important to dedicate a few lines to them. *Werkzeug* is "a comprehensive *WSGI* web application library" [56] that, in the past, began as a simple collection of various utilities for *WSGI* applications and now has become an advanced *WSGI* utility library. *Jinja* is a template engine for *Python*. It is inspired by the templating system of *Django*[27], has full Unicode support and runs on different versions of *Python*. It contains an optional integrated sandboxed execution environment and optional automatic escaping for applications where security is important [57]. In practical terms, special placeholders allow writing code similar to *Python* syntax in the template, a *HTML* file. Then the template receives data to render the final document [58]. In the code below the template receives a variable called "user" and the iteration allows the creation of an unordered list with different elements - deducted from the expressions "<ul>" and "<li>".

```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

Figure 2-5. Piece of code from a Flask template. The variable "user" is used to render the final document.

[25] https://werkzeug.palletsprojects.com

[26] https://jinja.palletsprojects.com

[27] https://www.djangoproject.com

Once created a basic *Flask* application, there are different extensions available in the community to add. According to the official website, the *Flask* core team "reviews extensions and ensures approved extensions do not break with future releases" [54]. There are extensions for different purposes, from database integration to open authentication technologies. It is up to the developer to choose the tools and libraries to use [53].

Although it is not the most famous framework for web development based in *Python* - since *Django* is a strong competitor - *Flask* is being used by important companies and institutions, such as *Netflix* [59], *Uber* [60], *Reddit* [61], *Trivago* [62], the *Massachusetts Institute of Technology* [63], *Lime* [64], etc.

2.2.2.3 MongoDB and NoSQL

Before writing about *MongoDB*, it is important to contextualize it as a *NoSQL* solution. The term *NoSQL* stands for "not only SQL" [65], but it also can be understood as a term for referring to non-relational databases. This is an approach to database design that provides "flexible schemas for the storage and retrieval of data beyond the traditional table structures found in relational databases" [65]. The term relational refers to the relational model of data management first described by *Edgar F. Codd* in 1969 and popularized in different database systems over the years starting with *System R* [66]. A relational database usually stores data in tables containing specific elements and types of data. "For example, a shop could store details of their customers' names and addresses in one table and details of their orders in another. This form of data storage is often called structured data" [49]. In this model, the tables usually consist of columns and rows. When data is inserted to a database, new records are added to existing tables or new tables are created. One important detail is that relationships between two or more tables can be established.

> *A table is referred to as a relation in the sense that it is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts, and the ability to retrieve related data from a table is the basis for the term relational database. A Database Management System (DBMS) handles the*

way data is stored, maintained, and retrieved. In the case of a relational database, a Relational Database Management System (RDBMS) performs these tasks. [67]

Back to the terms that started this section, *NoSQL* and *SQL*, the second term stands for *Structured Query Language*. It is not the unique possible language for implementing query with the relational model, but it is the most famous [66]. The development of *SQL* goes hand-in-hand with the relational databases and its simplicity is a big responsible for the popularity of the relational model [68]. *SQL* and relational databases have been the industry standard since the late 1970s with popular databases such as *Oracle*, *MySQL*, *PostgreSQL*, *DB2*, *SQL Server*, etc.

Although they have been around since the 1990s [66], *NoSQL* databases have become more popular over the last years in the era of *cloud computing*, *big data* and high-volume web and mobile applications [65]. The main difference between it and the relational databases is how they store the information. "A non-relational database stores data in a non-tabular form, and tends to be more flexible than the traditional, SQL-based, relational database structures. It does not follow the relational model provided by traditional relational database management systems" [49]. There are different ways to classify these systems, with categories that often overlap. In this work, a division of four primary data models with which a *NoSQL* database can manage information is used [65]: the key-value based, the document based (*MongoDB*, *CouchDB*), the column based (*Apache Cassandra*) and the graph based (*Neo4j, JanusGraph*). Since *MongoDB* is the chosen database for this project, the focus will be on document-based systems.

Instead of tables with specific pieces of data and relationships between tables, a document can be "highly detailed while containing a range of different types of information in different formats. This ability to digest and organize various types of information side-by-side makes non-relational databases much more flexible than relational databases" [49]. The non-relational databases can be used in contexts when large quantities of complex and different data need to be organized. In the context of *big data*, non-relational databases often perform faster because a query does not have to view several tables to deliver an answer, as relational databases often do [49]. Another detail about documents is their flexibility. "Document databases are designed for flexibility. They aren't typically forced to have a schema and are therefore easy to modify. If an application requires the ability to store varying attributes along with large amounts of data, document databases are a good option" [65].

Based on the previous statement, this important characteristic of non-relational databases fits very well the demands behind the development of *Ciudad Vestigio* in its first version - considering it a prototype and part of a research. Most of these systems do not require to make up-front commitments to data models and existing schemas can often be changed on the fly. The relational databases, in turn, have rigid data models that require careful design planning to ensure proper performance and resist evolution [66]. Another important characteristic of *NoSQL* systems is the ease of horizontal scalability. Both characteristics working together mean that these databases can address "large volumes of rapidly changing data, making them great options for agile development, quick iterations and frequent code pushes" [65].

Now that document-based non-relational databases were presented, this work will focus on *MongoDB* and the tools used for the development of this project: *MongoDB Atlas* and *MongoDB Stitch*. *MongoDB* is an open source [69], non-relational database management system (DBMS) [70] that uses flexible documents to process and store various forms of data [71]. Instead of tables from relational databases, a *MongoDB* database stores data in collections. A collection stores one or more *BSON* documents, which stands for *Binary JSON* [72] and are based on *JSON* [73]. As previously addressed in this section, documents are analogous to records or rows in a relational database table [74] and each document contains one or more fields. These fields, in turn, are similar to the columns in a relational database table and have value pairs. The values of fields may include data from recognizable types (numbers, strings, etc.), but also other documents, arrays and arrays of documents [75]. Some advantages of using documents are the fact that they correspond to native data types in many programming languages and also the reduction of expensive joins when using embedded documents and arrays.

The decision of using *JSON* as representation of data structures in *MongoDB* is related to its success within the software development context. However, there are issues that compromise the use of *JSON* inside a database. First, it is a text-based format and text parsing is very slow. Second, its readable format is not space efficient. Third it supports a limited number of basic data types [72]. To keep the connection between *MongoDB* and *JSON*, but generating high-performance and general-purpose, *BSON* was invented: a "binary representation to store data in *JSON* format, optimized for speed, space, and flexibility" [72]. In terms of security, the use of *BSON* is a good option against *SQL/Query Injection* [76], since *MongoDB* builds the binary when a client application makes a query, not a string [74]. When the developer uses the *MongoDB* driver with a programming language of choice, he/she is accessing *BSON* data through the abstractions available in the chosen language [72].

As mentioned earlier with non-relational databases, one of the main characteristics and attractions for this kind of model is the dynamic schema they provide when compared to the tabular and rigid model established by relational databases. In *MongoDB,* fields can vary from document to document in the same collection and it is not necessary to declare the structure of the documents to the database. The data type for a field can differ across documents in a collection as well [74]. The idea is that the documents are self-describing. Considering that data is not static these days, if a new field needs to be included in a document, it can be done without affecting the entire collection, without updating a central system control or taking the database offline [72]. To change the structure of documents in a collection, it is only necessary to update them to the new desired shape - adding new fields, removing some of them or updating values of fields to a new type [74]. In the opposite direction, this flexibility with documents can be interpreted as a problem. In situations where more control over the data structure and content is necessary, most document databases give developers the task of controlling this within the application code. Since its version 3.2 [77] MongoDB provides tools for enforcing document validation [72].

Lastly, *MongoDB* offers a *Community* and an *Enterprise* version of the database. The first one is the "source available and free to use" edition of *MongoDB*. The second is available as part of *MongoDB Enterprise Advanced* subscription [78] and includes several enterprise-focused features. *MongoDB* also offers *MongoDB Atlas* [79], a fully managed *Database as a Service (DBaaS)* [80] solution for *MongoDB* that requires no installation and offers a free tier for prototyping and early development [78]. This service was the chosen tool for the development of this first version of *Ciudad Vestigio*, in combination with another resource called *MongoDB Stitch*.

2.2.2.4 MongoDB Atlas and MongoDB Stitch

MongoDB Atlas [79] is the core of *MongoDB Cloud* [81] and provides a service to host and manage data with *MongoDB* in the cloud. It is a fully managed *Database as a Service (DBaaS)* solution [82] that allows the deployment of a database with the necessary features in minutes. In a context of innovation and agile development, typical on-premise deployments can last days during the stage of configuration. In turn, *MongoDB Atlas* handles the complexity of deploying, managing and healing database systems [83]. It works in combination with famous cloud service providers - *Amazon Web Services*[28], *Microsoft Azure*[29], *Google Cloud Platform*[30] - and it is available in more than 70 regions [81]. Since it works with more than one partner, it is possible to choose the best cloud provider, location and ecosystem for each project.

One important detail about the service is that *MongoDB Atlas* works with clusters. According to its documentation, they can be a replica set or a sharded cluster [84]. In the first case, the cluster is a group of *MongoDB* servers that maintain the same dataset, providing redundancy and high availability [85]. The second case is a method for distributing data across multiple machines, used to support deployments with very large datasets and high throughput operations [86]. Besides working with command line interface [87], *MongoDB Atlas* has a web interface that helps to manage clusters and databases, to visualize data within collections and make important operations like handling backup. It also brings metrics and allows to create alerts when the cluster reaches a possible threshold that can affect its performance.

[28] https://aws.amazon.com

[29] https://azure.microsoft.com

[30] https://cloud.google.com

The service has a free tier that fits the objectives of this project: a shared cluster with 512 megabytes of storage size [88]. The free tier allows 100 maximum connections, 100 maximum datasets, 500 maximum collections and its dashboard brings metrics for connections, logical size, network and opscounter [89]. In terms of security, it has end-to-end encryption and role-based action control [84]. A developer cannot enable backups with the free tier, but it can be achieved with the commands "mongodump" and "mongorestore" [87] as in an on-premise deploy.

The second mentioned tool, *MongoDB Stitch*, is part of *MongoDB Realm* [90]. It is a serverless platform [91] which allows client applications to access in a simple and secure way data and services, like *MongoDB Atlas*. The platform provides four services: *Stitch QueryAnywhere*, *Stitch Functions*, *Stitch Triggers* and *Stitch Mobile Sync* [92]. As will be addressed in chapter 5, it was used to securely query data from the database created with *MongoDB Atlas* by the *Ciudad Vestigio Mobile Application*. As well as *MongoDB Atlas*, *MongoDB Stitch* has a free tier where the first 25 gigabyte of data transfer per month is free; the minimum of 1.000.000 requests and 100.000 gigabyte-seconds compute per month are also free [92]. The configuration of the platform is made on the same *MongoDB Cloud* web application which includes *MongoDB Atlas*, detail that facilitates the integration of both tools. In the context of this project, once configured the *MongoDB Stitch* backend, it is possible to access the services directly from different frontend applications using one of *Stitch*'s native *SDKs* for *iOS*, *Android* and *JavaScript*.

2.2.3 The Ciudad Vestigio Mobile Application

2.2.3.1 Available technologies, possible scenarios

There are different ways to create a mobile application nowadays. A first option is to develop a native app for the chosen operating system. For *iOS* from *Apple* [93], for example, this can be done using the *Swift* or *Objective-C* programming languages with the *iOS SDK* and the *XCode IDE* [94]. For *Android* from *Google* [95], this can be done using the *Java* or *Kotlin* programming languages with the *Android SDK* and an *IDE* like *Android Studio* [96]. The main advantage of choosing this option is the possibility to create a native app exploring all the capabilities that the operating system and its software development kit can deliver in combination with the hardware, in this case the smartphone. Another advantage is related to the performance of the application, since the developer is using the native language and the tools provided by the chosen platform for this purpose. It also brings access to all the native pre-built user interface components, such as structured layout objects and user interface controls [97]. On the other hand, the main disadvantage of building a native mobile application is the necessity to develop one application for each chosen operating system. It has an important impact in time and costs involved, since the situation demands a bigger or more experienced team that can handle the development of all the necessary applications separately.

A second option are *Web Apps* or *Progressive Web Apps (PWA)*. This second term is a concept introduced by *Google* for "websites that took the right vitamins" [98]. They are not "real" mobile applications, but web apps that provide an "installable, app-like experience on desktop and mobile that are built and delivered directly via the web" [99]. Since they are executed in the web browser, they can easily be reached from different operating systems and devices with a single codebase, what simplifies the whole process. Another advantage is the freedom to exist outside the dynamic and rules of app stores [100], which can be bureaucratic and charge money from the developer [101]. The main difference between the *PWA* and the common pages is that the first has offline capabilities and can access resources from the smartphone like push notifications or hardware features. However, it is important to mention that the possibility to use these features is not comparable to what a native application can do. Since the *PWA* is not a real mobile application, this is not an approach desired for *Ciudad Vestigio* and this option was discarded for developing the project.

Over the past few years a third option gained strength: the use of mobile development frameworks for the creation of cross-platform applications. In general, they allow the compilation of mobile applications for different operating systems from one common codebase. There are different options of tools available in the market - such as

Apache Cordova[31], *PhoneGap*[32], *Flutter*[33], *React Native*[34], *Xamarin*[35], *Ionic*[36] - each one with a particular approach when creating the applications. This universe is also referred to as *Hybrid App Development*, since the solutions presented by these tools are typically a blend of native and web solutions. The official website of *Ionic* explains the mechanism:

> *The heart of a hybrid-mobile application is still just an application that is written with HTML, CSS, and JavaScript. However, instead of the app being shown within the user's browser, it is run from within a native application and its own embedded browser, which is essentially invisible to the user. For example, an iOS application would use the WKWebView to display our application, while on Android it would use the WebView element to do the same function.* [100]

But since the core of the application is written using web technologies, how can they take advantage of all the capabilities of the hardware? The answer lies in the use of plugins with which the apps can have full access to the hardware's features [100].

Due to the maturation of these frameworks, the expansion of their communities, the availability of documentation, the benefits of saving time and the ease of developing applications for different operating systems with just one common codebase, this is the approach chosen for the development of the *Ciudad Vestigio Mobile Application*. But since there are many good options available, each with particular details, choosing one as the "official tool" for the project can become a complex task. As addressed in the introduction of this work, there are some criteria for determining the use of technologies in the project - strong community, large documentation, low price, high availability and

[31] https://cordova.apache.org

[32] https://phonegap.com

[33] https://flutter.dev

[34] https://reactnative.dev

[35] https://dotnet.microsoft.com/apps/xamarin

[36] https://ionicframework.com

connection with the state of the art in software development. When analyzing different scenarios based on these criteria, the process of deciding which framework to use reached two options: *React Native* and *Flutter*. Both are very young technologies, the first is from 2015 and the other is from 2017, but both were created and are used by two giant companies, *Facebook* [102] and *Google* [103] respectively, which means they have some of the most talented teams of engineers in the world overseeing them, maintaining them, pushing them forward and adding new features [104, p. 10]. Besides this, both are technologies with recognizable growth over the last years. This growth trend can be confirmed from the data that *Stack Overflow*[37] collects and publishes every year. For almost ten years [105] the platform conducts a survey with its community of developers that ask questions about important topics, including demographic issues [106] [107]. The 2020 survey was taken by nearly 65.000 people from 186 countries.

When analyzing the data on mobile development, it is possible to verify that *React Native* went from a technology that didn't even appear in the ranking in 2018 to the first position in 2019 and 2020 among the mobile development frameworks. The first table shows a piece of data from the "Other Frameworks, Libraries and Tools" category for the years 2019 and 2020. It compares the most popular tools for different purposes, not only mobile programming. In 2019 *Flutter* was behind *Cordova* and *Xamarin*.

	2019		**2020**	
	All Respondents	**Professional Developers**	**All Respondents**	**Professional Developers**
React Native	10.5%	10.8%	11.5%	11.8%
Flutter	3.4%	3.2%	7.2%	6.6%
Cordova	7.1%	7.4%	6.0%	6.4%
Xamarin	6.5%	6.5%	5.8%	6.0%

Table 2-2. The most popular tools among mobile development frameworks according to Stack Overflow Annual Developer Survey (2019 and 2020).

[37] One of the most important online communities for developers [192] nowadays.

When analyzing the "most loved" and "most wanted" tools in the last two years, *Flutter* stands out. The "most loved" means the percentage of users who are developing with the technology and have expressed interest in continuing developing with it. The "most wanted" means the percentage of users who are not developing with the technology but have expressed interest in developing with it [107].

	2019		**2020**	
	Most Loved	**Most Wanted**	**Most Loved**	**Most Wanted**
React Native	62.5%	13.1%	57.9%	14.0%
Flutter	75.4%	6.7%	68.8%	10.7%

Table 2-3. Comparison between React Native and Flutter in the categories "most loved" and "most wanted" tools according to Stack Overflow Annual Developer Survey (2019 and 2020).

On the other hand, when analyzing the "most dreaded" tools, *Cordova* and *Xamarin* stand out negatively and *Flutter* stands out positively. The "most dreaded" means the "percentage of developers who are developing with the technology but have not expressed interest in continuing to do so" [107].

	Most Dreaded (2019)	**Most Dreaded (2020)**
Cordova	63.3%	71.3%
Xamarin	51.7%	54.6%
React Native	37.5%	42.1%
Flutter	24.6%	31.2%

Table 2-4. Comparison between four mobile development frameworks in the category "most dreaded" according to Stack Overflow Annual Developer Survey (2019 and 2020).

Although the presented data is not an absolute assessment tool, it confirms a personal observation as a developer: *React Native* is currently the most promising framework for creating cross-platform mobile applications and *Flutter* is a rising competitor to watch for the next years. When analyzing another source of information, *GitHub*, *Flutter* has - at the time of writing this document - more stars than *React Native* (97.7k against 89k), but *React Native* has more forks than *Flutter* (19.8k against 13.5k) [108] [109]. In 2018, *React Native* had the second highest number of contributors for any repository in *GitHub* [110] and in 2019 it was sixth [111]. *Flutter*, in turn, moved from a technology that didn't even appear in the 2018 ranking of *GitHub* to second place in 2019. In the case of *React Native*, it is important to mention that it uses *JavaScript* for development - one of the main programming languages in use nowadays [107] - and it is conceptually connected with another famous framework created by *Facebook*: the *React.js*. *Flutter*, in turn, uses the *Dart* programming language, developed by *Google*.

React Native is also known for its performance and for the fact that the applications created with it will render using real mobile *User Interface* components, not *webviews* [112, p. 1], what make them look, feel and perform like any other native app [113]. As explains the official webpage: "In Android development, you write views in Kotlin or Java; in iOS development, you use Swift or Objective-C. With React Native, you can invoke these views with JavaScript using React components. At runtime, React Native creates the corresponding Android and iOS views for those components" [113]. After all the debates promoted throughout this section, the framework chosen to develop the *Ciudad Vestigio Mobile Application* is *React Native*.

2.2.3.2 React Native

React Native is a framework for building mobile applications in more than one operating system - *iOS* and *Android*. It uses the *JavaScript* programming language with the *React*[38] library for building user interfaces and *JSX*. The *JSX* is a syntax extension of *JavaScript* that looks similar to a template language like *XML*, but, as explains the official webpage of *React*, "it comes with the full power of JavaScript" [114]. The code below,

[38] https://reactjs.org

for example, declares a variable called "name" and then use it inside *JSX* by wrapping it in curly brackets:

```
const name = 'Cibeles';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Figure 2-6. An example of embedding expressions in JSX. It is possible to put any valid JavaScript expression inside the curly braces in JSX.

The use of *JSX* is based on the idea that rendering logic is inherently coupled with how events are handled, how the state changes over time and how the data is prepared for display. "Instead of artificially separating technologies by putting markup and logic in separate files, React separates concerns with loosely coupled units called 'components' that contain both" [114]. It is important to mention that *React* and *React Native* does not require using *JSX*, but its use is encouraged by making the code more readable and easier to maintain.

The *React* library referred to more than once is an open source *JavaScript* library also created by *Facebook*. It was originally used to build user interfaces for web applications, with a particular approach for *DOM* manipulation [115]. "Traditional DOM manipulation is slow and expensive in terms of performance and should be minimized. React bypasses the traditional DOM with something called the virtual DOM: basically, a copy of the actual DOM in memory that only changes when comparing new versions of the virtual DOM to old versions of the virtual DOM. This minimizes the number of DOM operations required to achieve the new state" [104, p. 8]. In the case of *React Native*, this mechanism for handling element updates is part of the process.

> *The update cycle in React Native is the same as in React: when props or state change, React Native re-renders the views. The major difference between React Native and React in the browser is that React Native does this by leveraging the UI libraries of its host platform, rather than using HTML and CSS markup.* [112, p. 2]

As mentioned briefly in the last section, *React Native* allows the developer to build applications with native views and access to native platform-specific components. The fact that it renders using its host platform's standard rendering *APIs* makes its approach different from other cross-platform application development tools.

> *This sets React Native apart from other hybrid app frameworks like Cordova and Ionic, which package web views built using HTML and CSS into a native application. Instead, React Native takes JavaScript and compiles it into a true native application that can use platform-specific APIs and components. Alternatives like Xamarin take the same approach, but Xamarin apps are built using C#, not JavaScript. Many web developers have JavaScript experience, which helps ease the transition from web to mobile app development.* [104, p. 4]

Despite the approach of using *JavaScript*, *HTML*, and *CSS* to render interfaces using *webviews* works, it comes with drawbacks, especially around performance [112, p. 2]. In terms of experience, once these processes usually do not have access to the set of native UI elements from the host platform, the result sometimes mimics these native UI elements. In this context, the "reverse-engineering" of things like animations takes big amounts of effort and they can quickly become outdated [112, p. 2]. In contrast, because the application created with *React Native* renders native components and *APIs* directly, performance and user experience are considered much better [104, p. 15].

However, a common drawback pointed out about *React Native* is the fact that it is an abstraction built on top of existing platform *APIs* [104, p. 15]. When updates are released for the host platform - for example, a new suite of features in a new version of *Android* or *iOS* - there will be a lag before they are fully implemented and supported in *React Native* [112, p. 4]. In these cases, the options for the developers are to implement custom support for missing *APIs* themselves or wait until *React Native* reaches the parity with the new release.

Lastly, in order to understand better the code created and available for the *Ciudad Vestigio Mobile Application*, it is important to address the lifecycle of an application built with *React Native*, as well as its building blocks: *Components*. "The entry point of an application is a component that requires and is made of other components. These components may also require other components, and so on" [112, p. 5]. There are two main types of *React Native* components: stateful and stateless. The example below presents a stateful component:

```
class Example extends React.Component {
  constructor() {
    super()
    this.state = { name: "Cibeles" }
  }

  render () {
    return (<OtherComponent />)
  }
}
```

Figure 2-7. An example of stateful component with two lifecycle methods: "constructor" and "render".

The code below presents an example of a stateless component.

```
function Example(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Figure 2-8. An example of stateless component with props, the correct way to render data with them.

The main difference between the two components is that the stateless do not "hook into any lifecycle methods and hold no state of their own, so any data to be rendered must be received as properties (props)". The last code, a function, is a valid *React* component because it accepts a single property object argument with data and returns a *React* element. "We call such components 'function components' because they are literally JavaScript functions" [116]. Important to mention that props are read-only, the developer must never modify them [117].

About the *React Native* lifecycle, when a class is created, a series of "methods are instantiated that you can hook into" [112, p. 6]. In the example of the stateful component presented, the methods are "constructor" and "render", but there are a few more with their specific use cases [118]. Another very important method in the context of the *Ciudad Vestigio Mobile Application* is "componentDidMount". Among them, the only required lifecycle method is "render", the others are optional. However, since they work in sync, these lifecycle methods are very useful to manage the state of components as well as execute code at each step of the way [112, p. 6].

2.2.4 The Ciudad Vestigio VoiceBot

2.2.4.1 Available technologies, possible scenarios

Over the past few years, voice assistant technologies have become increasingly popular. In smartphones with *Android* and *iOS*, for example, *Google Assistant*[39] and Siri[40] are both available voice assistants that many people carry every day in their pockets, using them or not. Currently, as voice recognition technologies mature [119] and as the interest in the concept of smart home grows [120], products like *Echo*[41] from *Amazon* or *Google Home Mini* are increasingly common in people's daily lives. When analyzing the topic from a developer's perspective, the tools for creating applications with cognitive assistants are also more present, especially in the world of cloud computing services: *IBM Cloud* has *Watson Assistant*[42], *Microsoft Azure* has *Virtual Assistant*[43], *Amazon Web*

[39] https://assistant.google.com

[40] https://www.apple.com/siri

[41] The product comes with the *Alexa* assistant.

[42] https://www.ibm.com/cloud/watson-assistant

[43] https://microsoft.github.io/botframework-solutions/overview/virtual-assistant-solution/

Services has *Amazon Lex*[44], *Google* has *DialogFlow*[45]. All of these services can be combined with other services from these same providers to perform the functions of "text-to-speech" and "speech-to-text".

Given this context, the first important decision before starting the development of the *Ciudad Vestigio VoiceBot* was whether it would be better to use a cloud computing service or to develop the technology totally from scratch. Both options have advantages and disadvantages. The use of a cloud computing platform - like those mentioned - brings a huge benefit in ease and speed of development in comparison to the second option. All details of the infrastructure are managed by the service provider and the developer does not have to worry about most of the difficult system setup and maintenance procedures [121]. In the context of *Ciudad Vestigio*, the main disadvantage of using a cloud computing service is the fact that the platform "loses control" of the content shared by citizens or institutions, since the core of the *Ciudad Vestigio VoiceBot* is built with an external service and there is no way to influence this mechanism. This observation is not related to content control - otherwise the platform would not be collaborative on this point - but to ensure that unwanted uses of the project are not allowed, such as using it to disseminate hate speech. However, a promising solution to this issue is addressed in the conclusions of this work.

The second option of scenario - which proposes the development of the *Ciudad Vestigio VoiceBot* totally from scratch - has the advantage of allowing the user who wants to share content to do everything in one place, without the need to use external services. In this context, for example, the configuration of the *Ciudad Vestigio VoiceBot* could be done within the *Ciudad Vestigio Web Application,* together with the *Ciudad Vestigio Device* registration. This approach would also make it possible to deal with the issue of "platform misuse" in more detail, if necessary. This scenario could even compose a monetization strategy for the project, considering that some institutions may need to scale services due to the large volume of requests. On the other hand, this second option adds a huge pressure in terms of effort and time behind development, as well as the

[44] https://aws.amazon.com/lex

[45] https://cloud.google.com/dialogflow/

possibility of not working as expected with the time available for this research. The development of a *machine learning* model like this goes further than the objectives and scope of the project for now.

Based on the arguments presented, when comparing advantages and disadvantages for both scenarios, it was decided that the *Ciudad Vestigio VoiceBot* would be built using cloud computing services. As a first version and as a research project, the main objective is to create a fully functional platform within the established parameters - and time is one of the most important.

Once defined this first detail, a second important question arises: which cloud computing platform should be used when there are many good options available? As addressed over the introduction of this work, *IBM Cloud* was the chosen one. The decision of using this platform was based on three arguments. The first one is related to a short prior knowledge about the platform, which considerably reduces the time needed to study the tool before starting development. The second argument is related to the amount of documentation available - official or made by the community - about creating and running a chatbot with *IBM Watson Assistant*. There are different hands-on tutorials on how to integrate it with other technologies, which is very useful in the context of the project. The third argument is related to the price. As will be shown in the next section, *IBM Cloud* has a free plan that perfectly suits the current demands of *Ciudad Vestigio* and its possible users.

2.2.4.2 IBM Watson Assistant

The *IBM Watson Assistant* is a service that combines "machine learning, natural language understanding, and an integrated dialog editor" to create conversation flows between the apps and its users [122]. The assistant itself is a cognitive chatbot that uses machine learning models [123] built from data provided by the user or institution that hires the service. The objective is to deliver accurate answers in real time for interactions [124] in different applications, devices or channels [125].

When a user queries the system, the assistant tries to identify the intent[46] behind it and routes the query to a skill, which provides the appropriate response. There are two types of skills: the "dialog skills" and the "search skills". The "dialog skill" uses provided information about subjects that users are interested in and how they ask about them. With this data the service dynamically creates a machine learning model made to understand the same or similar requests made [121]. With a graphical editor it is possible to create a script of responses for the assistant to use when interacting in a dialog with users. This is the type of skill used to create the *Ciudad Vestigio VoiceBot*. The "search skill" is only available for plus or premium users. It uses "corporate knowledge bases" or other "collections of content authored by subject matter experts" that are configured for this purpose. The objective is to find a relevant answer for unanticipated or "more nuanced" user queries [121] inside these collections of content.

There are different options to integrate the assistant with other systems [126]. It can be achieved by publishing the assistant directly to an existing social media messaging platform, such as *Facebook Messenger*, as well as embedding a chat inside a website. A third option, which corresponds to the context of *Ciudad Vestigio*, is the integration of the assistant in a custom application, such as a mobile app or a robot with a voice interface.

One important detail about *IBM Watson Assistant* is that the service provides a text-based solution, not a voice-based solution (as might be expected in the context of the *Ciudad Vestigio VoiceBot*). By integrating the service with others - such as *Watson Speech to Text* and *Watson Text to Speech* - it is possible to create the expected voice-based interaction [121]. However, as addressed in chapters 5 and 6, these services were not used mainly due to performance reasons. In the end, other solutions were adopted and the consequences of this decision are discussed in the conclusions of this work.

Lastly, in terms of costs, it is possible to create a *Watson Assistant* service with the free plan called "Lite" and there is no need to add credit card details as some kind of

[46] If the assistant does not identify an intent with high confidence, it can return a message saying that it did not understand the question or that it is in doubt. The mechanism of creating dialog flows will be presented in chapter 6.

"guarantee". The plan offers 10.000 messages per month for up to 1.000 active users [127] per month. It is possible to create up to five dialog skills, each one with 100 dialog nodes [128]. Since the idea behind the project is that users and institutions can add devices to the platform and share content about the, the detail of having a free service option and not having to provide credit card data facilitates the process. In addition, considering that this is a first version of the project, the services offered by the free plan seem to be minimally adequate with the needs of a user or institution newly entered on the platform. As demand increases, it is possible to invest in other plans.

Chapter 3 – Device: System Development

This chapter opens a series of four chapters with the common purpose of presenting the development details for each one of the technologies that forms the *Ciudad Vestigio* platform. This first chapter focuses on the *Ciudad Vestigio Device*, built with the already presented *HM-19 Bluetooth Low Energy* module.

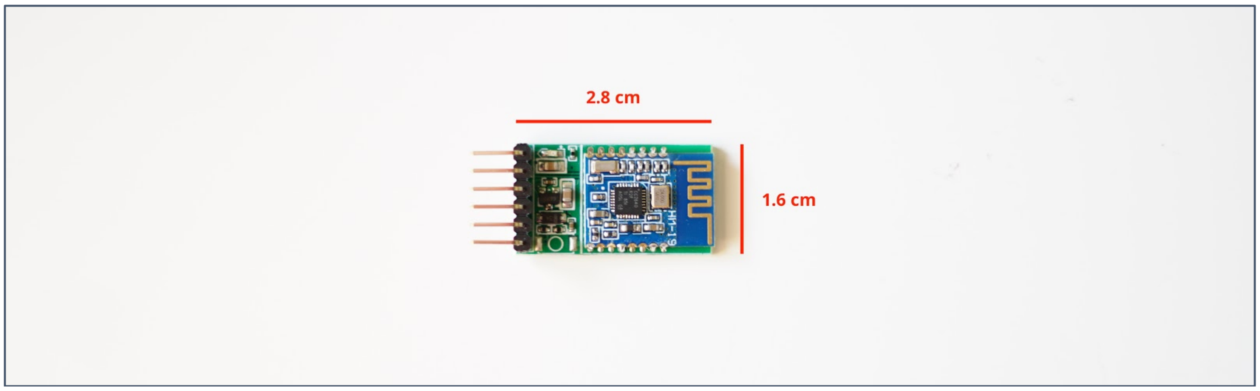


Figure 3-1. Image of the HM-19 module with information about its size.

It is important to note that the device used is a *HM-19* module from the company *Jinan Huamao* as the core module - the blue part - with a six pin base board from another manufacturer, in this case *DSD TECH*. This arrangement of the *HM-19* module is very common and easily available for sale on the market. It is more convenient to connect the module with other boards with no need for welding, which is the case of the project.

The two necessary steps to start working with the device serve different purposes but are strongly connected: how to properly power the device and how to establish a communication with it in order to send commands and carry out its configuration according to the demands of the project. The first task does not demand a complex investigation, since the board itself gives an explicit indication of the necessary voltage to achieve it.

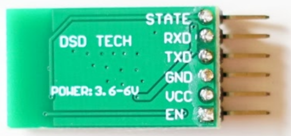


Figure 3-2. Image of the back of the HM-19 module where it is possible to see (in addition to the function of each pin) the correct voltage to power the board.

The indication that a value between 3.6V and 6V (volts) is acceptable brings good news, because it is possible to use different approaches to combine both tasks in one solution during the stage of development. The *HM-19* module itself requires a power supply that can range from 1.9V to 3.7V. The printed indication of 3.6V to 6V is related to the 6 pin base board which has a voltage regulator to deal with this issue. This is an important detail to consider. Back to the options to start working with the device - since it is necessary to power the *HM-19* module and communicate with it - this document presents two alternatives: using an *Arduino Uno R3* board or using a *FTDI FT232RL* board.

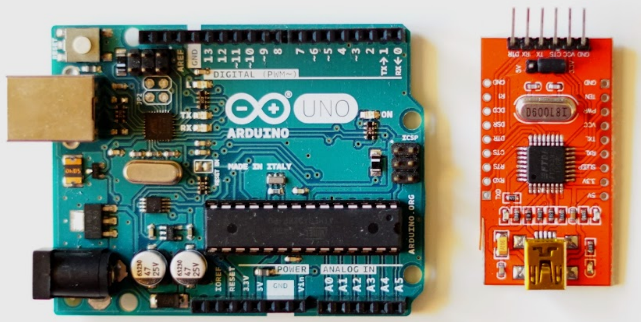


Figure 3-3. Image of the Arduino Uno R3 board and the FTDI FT232RL board. Both can be used to power the HM-19 module and allow communication with it in order to send configuration commands.

Arduino Uno is a microcontroller board based on the *ATmega328P* and the most used and documented board of the whole *Arduino* family [129]. It is currently in its "Revision 3", but in the past *Arduino Uno* was chosen to mark the release of the *Arduino Software (IDE)*. "The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform" [129]. Although its main features will not be used in this project, the *Arduino* board is able to read inputs - such as a "light on a sensor, a finger on a button, or a Twitter message" - and turn them into outputs - "activating a motor, turning on an LED, publishing something online". As explained on the official website: "You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing" [130].

The *FT232RL* board, in turn, is a *USB* to serial *UART* interface device which "simplifies USB to serial designs and reduces external component count by fully integrating an external EEPROM, USB termination resistors and an integrated clock circuit which requires no external crystal into the device" [131]. It has been "designed to operate efficiently with a USB host controller by using as little as possible of the total USB bandwidth available" [131]. The necessary drivers to work with it can be obtained in the official website [132].

Both possibilities can deliver 5V - value inside the ideal range - and both can establish a proper communication with the *HM-19* module in order to configure it according to the *Ciudad Vestigio* specifications. The purpose behind mentioning two options is related to the idea of giving more than one possible way to configure the device within the project. Choosing one or the other is more a matter of taste, costs or availability.

For both options - *Arduino Uno R3* and *FTDI FT232RL* board - the connection to the *HM-19* is represented below using the *Fritzing* software tool [133]. The chosen option, in turn, needs to be connected to a computer using the *USB* port. This dependency on a computer is necessary only during the configuration step. Once finished, a different solution will also be presented in the end of this chapter to show how it is possible to power the device by plugging it into the energy outlet.

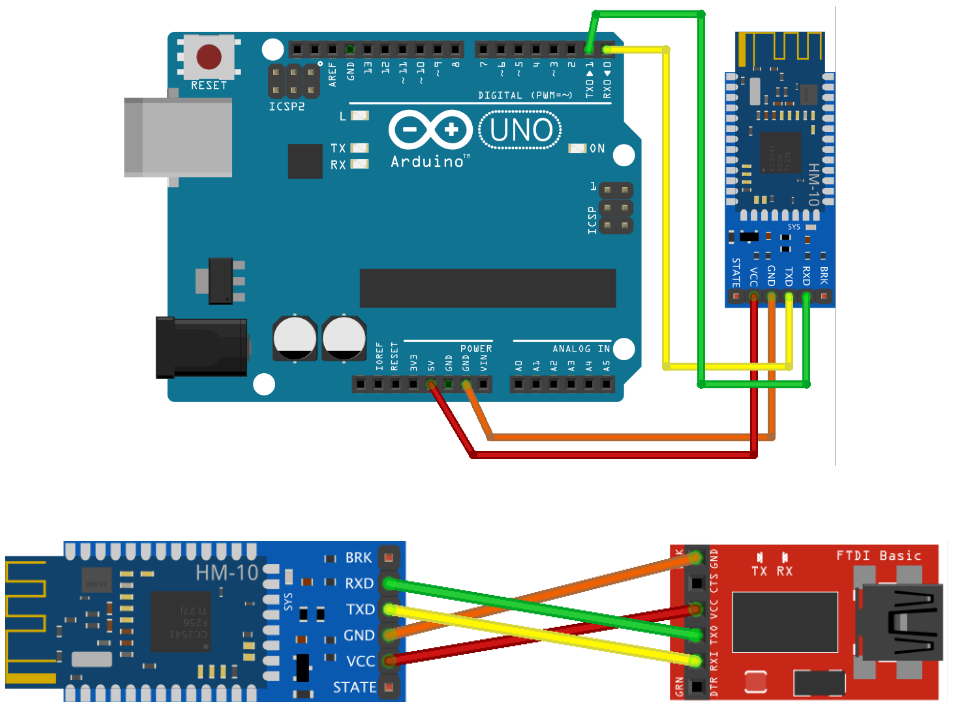


Figure 3-4. Image produced with Fritzing representing the connections of the HM-19 module with the Arduino board (above) and with the FTDI FT232RL board (below). The image of the HM-19 module is represented by a "sister board", the HM-10. There are no problems since the order of the pins is the same.

The first task of supplying power to the *HM-19* is done. The second is to establish the communication between the computer and the module. In order to do this, the *Arduino Software (IDE)* will be used, regardless of the option chosen - *Arduino* or *FTDI FT232RL*. The *Arduino Software (IDE)* can be downloaded from the official website of the project[47]. At the time of writing this document, the software is in its version 1.8.13.

The *Arduino Integrated Development Environment* - or *Arduino Software (IDE)* - contains "a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus", as explains the official website [134]. It has the important task of allowing the connection to the *Arduino* hardware in order to send commands and upload software to it. Programs written using the software are called sketches and these sketches are saved with the file extension ".ino".

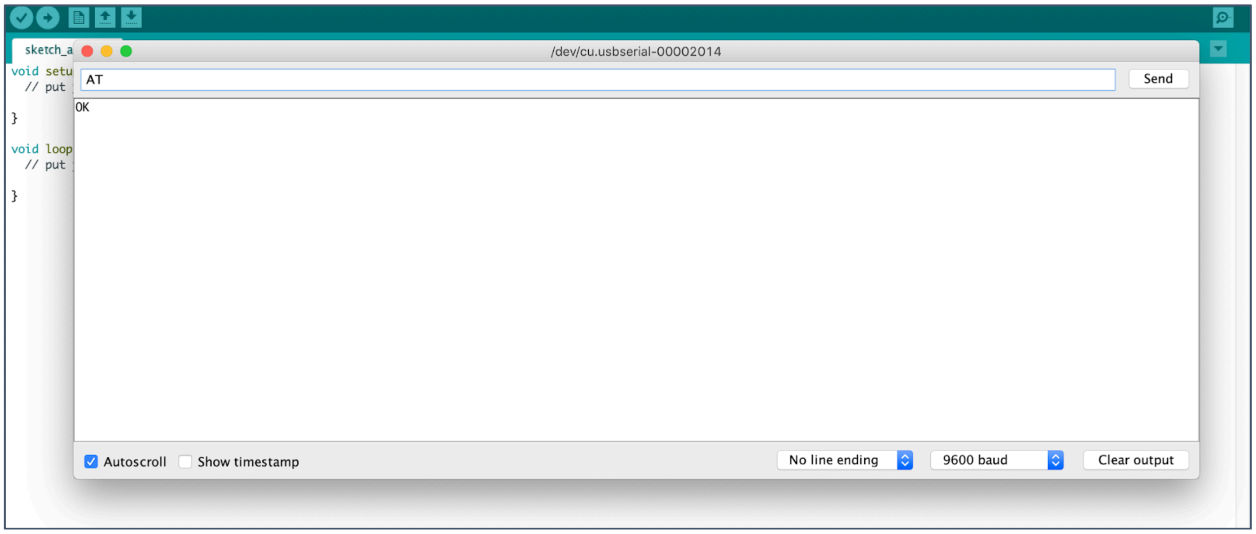
[47] https://www.arduino.cc/en/Main/Software

With the *Arduino Software (IDE)* opened and the board connected via *USB* port, it is necessary to select the port of the *Arduino Uno R3* board or *FTDI FT232RL* board. This can be done in the "Tools" menu, within the option "Port".

Although the *Arduino Software (IDE)* has a famous text editor - where developers create code for different boards - the tool that will be used to configure the *Ciudad Vestigio Device* is the *Serial Monitor*. Inside the universe of *Arduino*, the *Serial Monitor* allows the communication with the board [135, p. 3] in both directions: it can display serial data sent from the *Arduino* [136, p. 95] and it lets the user send data to the *Arduino* board by entering text in the textbox before pushing the "send" button [136, p. 89]. The *Serial Monitor* is extremely helpful when debugging a program [135, p. 3]. In our case, it will be used to communicate with the *HM-19* module, using one of the presented options as medium. The *Serial Monitor* will be used with a *baud rate* of 9600, the speed at which data is transmitted (measured in bits per second) [136, p. 89]. It is approximately 1.000 characters per second [136, p. 96]. The tool will also be used with the option "No line ending" selected to automatically send a carriage return or a combination of a carriage return and a line at the end of each message sent when clicking the "send" button.

To communicate with the *HM-19* module, *AT Commands* will be used. The *AT Commands* are a standard set of commands created by *Dennis Hayes* [137] in the context of his company, the *Hayes Microcomputer Products*, an important manufacturer of modems, as explains a 1998 article from *The New York Times*. "If you're on the Internet, you probably have this company to thank for it: Hayes — or rather, its founder, Dennis Hayes — invented both the personal-computer modem and some of the basic standards that all of today's modems still answer to, according to the company and on-line experts" [138].

A first communication check can be done with the *Serial Monitor* by sending the command "AT", which receives an "OK" response from the *HM-19*.



```
void setu
  // put
}

void loop
  // put
}
```

```
AT
```

```
OK
```

Figure 3-5. Screenshot of the Arduino Software that shows a communication check with the HM-19 module using the Serial Monitor.

Once the communication is established correctly, it is possible to start the most important stage: the configuration of the device as a *beacon*. To perform this task, the official documentation of the *HM-19* module will be used and it can be obtained from the manufacturer's official website [139].

Since there is no official *UUID*, *Major* or *Minor* values calculated by the *Ciudad Vestigio Web Application* so far to include in the device[48], fake data will be used because the objective for now is to configure the *HM-19* module as a *beacon* and verify its correct operation. The sequence of executed *AT Commands* are based on the official documentation [140].

ORDER	**AT COMMAND**
1	AT
2	AT+RENEW
3	AT+RESET
4	AT

[48] This mechanism will be introduced in the next chapter.

5	AT+IBE012345678
6	AT+IBE112345678
7	AT+IBE212345678
8	AT+IBE312345678
9	AT+MINO0x2019
10	AT+MARJ0x2020
11	AT+ADVI5
12	AT+ADTY3
13	AT+IBEA1
14	AT+DELO2
15	AT+PWRM0
16	AT+RESET

Table 3-1. List of commands needed to configure the HM-19 module as a beacon. Fake UUID, Major and Minor data were used.

Once all the commands are presented, it is necessary to analyze them one by one. As already mentioned, the first and third commands, just "AT", are responsible for verifying the communication with the module. The "AT+RENEW" command restores all set values to factory setup and the next command, "AT+RESET", restarts the module. The sequence of four commands that start with "AT+IBE" is responsible for setting the *beacon UUID*. The 32 hexadecimal digits are divided into four segments and each one is set with its own command. In this first test, all segments were set to "12345678". The next commands, which start with "AT+MINO" and "AT+MARJ" respectively, set the *Minor* and *Major* values of the device. Note that these values are sent in hexadecimal, as can be seen by the presence of "0x" before them. The values can range from 0x0001 to 0xFFFE for both. The next command, "AT+ADVI5", sets the advertising interval of the device in milliseconds. This value ranges from "0" to "F", representing the information below:

VALUE	INTERVAL
0	100 ms
1	252.5 ms
2	211.25 ms
3	318.75 ms
4	417.50 ms
5	546.25 ms
6	760.00 ms
7	852.50 ms
8	1022.5 ms
9	1285 ms
A	2000 ms
B	3000 ms
C	4000 ms
D	5000 ms
E	6000 ms
F	7000 ms

Table 3-2. List of possible time intervals that the device can use to advertise packets.

From the command table it is possible to verify that a 546.25 milliseconds interval is being used. This is an important command when creating a low power *beacon*, as the increase of the advertising interval reduces the amount of energy used by the module. For this first version *Ciudad Vestigio* this is not exactly a concern, because this work considers as a premise that the devices are plugged into the outlet.

The next command, "AT+ADTY3", sets the advertising type. There are four options of parameters available.

VALUE	RESULT
0	Advertising, ScanResponse, Connectable
1	Only allow last device to connect in 1.28 seconds
2	Only allow Advertising and ScanResponse
3	Only allow Advertising

Table 3-3. List of possible ways to configure the type of advertising with the device.

The value "3" as parameter is the most appropriate for the project for two reasons. The first one is related to the fact that connections and scan responses are not features designed for the *Ciudad Vestigio Device*. The second reason is related to the decrease in the amount of energy used by the module by canceling connections and scan responses.

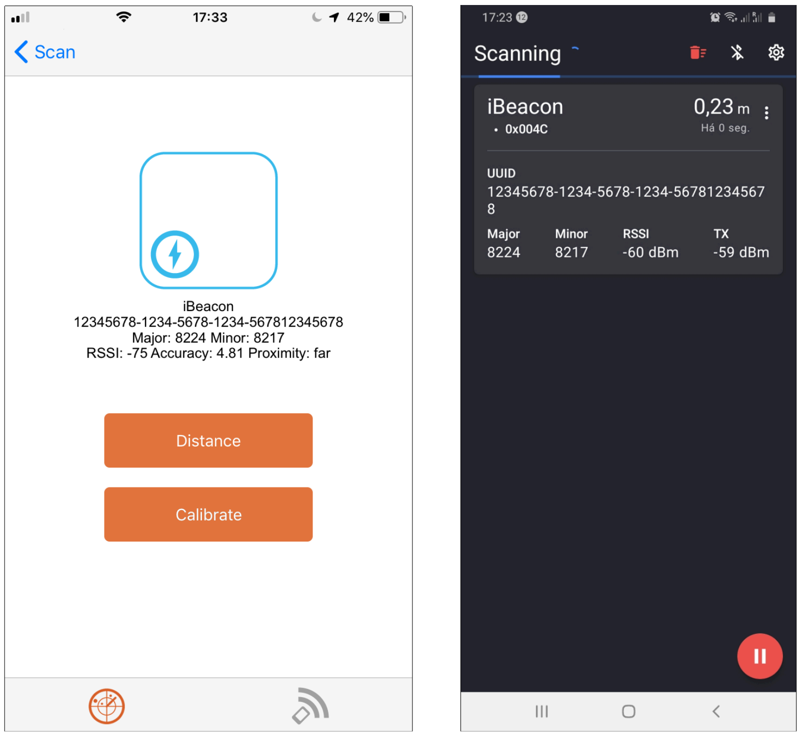
The next command, "AT+IBEA1", turn the *beacon* on or off. In this case the command sends a "1" value turning it on. The next command, "AT+DELO2", sets the *beacon* deploy mode. The value "1" configures it with the broadcast and scanning functions, while the value "2" configures it with the broadcast function only. Since the *beacon* devices in *Ciudad Vestigio* have the exclusive mission of advertising their *UUID*, *Major* and *Minor* values there is no reason to turn on another feature that increases the amount of energy used.

The last command before restarting the module again is "AT+PWRM0" and it is responsible for setting a sleep type for the module. The value "0" allows the device to enter into "auto sleep" mode and the default value "1" does not allow it. This is an important command since it is another feature that collaborates to reduce the power consumption by the device. According to the documentation, the module works with 6.5mA when in active state and in sleep state this value goes from 50uA to 1200uA. It is important to mention that the *HM-19* will no longer respond to *AT Commands* after sending the "AT+PWRM0" command and restarting the module. If the user wants to keep working with the device to set more details, this command should be the last step. In the case where the module is already in sleep mode, a simple way to wake it up is sending a long string to the module via *Serial Monitor*. After waking up the module is available to send and receive *AT Commands*.

It is important to note that every command presented in the reference table receives a response by the module via *Serial Monitor*. There are also commands to verify the current values set in the board, but they will not be covered in this work. These commands can be obtained from the official documentation of the module [140].

The next step is to test and confirm the correct functioning of the device as a *beacon*. To achieve this objective, external applications aimed at interacting with *beacon* devices will be used, one option for *iOS* and other for *Android*. For *iOS* the application used is *Locate Beacon* [141] and for *Android* the application used is *Beacon Scanner* [142].

As can be seen in the image below, both applications found the device with the defined *UUID*, *Major* and *Minor* values, proving its correct configuration.



iBeacon
12345678-1234-5678-1234-567812345678
Major: 8224 Minor: 8217
RSSI: -75 Accuracy: 4.81 Proximity: far

iBeacon 0,23 m
• 0x004C
Há 0 seg.
UUID
12345678-1234-5678-1234-56781234567
8

Major	Minor	RSSI	TX
8224	8217	-60 dBm	-59 dBm

Figure 3-6. Screenshot of the two mobile apps used to prove the correct functioning of the HM-19 module as a beacon. The image on the left shows an iOS app, the image on the right shows an Android app.

It is important to mention that this configuration step will be recalled in the next chapter, when this work presents the development of the *Ciudad Vestigio Web Application*. With the *UUID*, *Major* and *Minor* values of the device calculated by the application, it will be possible to make the device recognizable within the platform.

So far, the *HM-19* module has been powered by a computer, using an intermediate board[49] *connected to a USB* port of this computer. Once finished the *Ciudad Vestigio Device* configuration stage, it is no longer necessary to communicate and send commands to the device. This situation allows us to think of other ways to power the device before placing it in the "real world", once using a *USB* port does not seem to be a viable model. As already discussed, in this first version of the project it is considered that users and institutions interested in adding devices to the platform are able to plug these devices into an energy outlet.

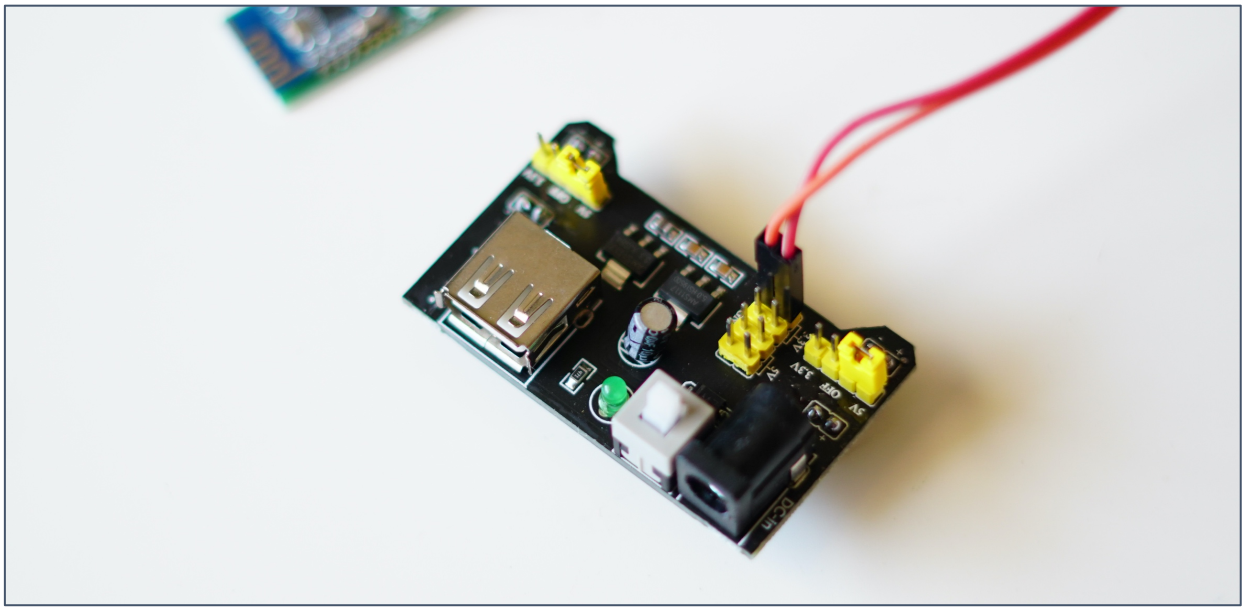


Figure 3-7. Image of a MB102 module in detail. It has a USB port on the left and a Barrel Jack connector on the right.

Given this premise, a *MB102* module will be used as an option to power the *HM-19*. It costs an average of 3 euros in *Spain* and has two standard inputs: a *USB* port and a

[49] Either the Arduino Uno or the FTDI FT232RL board.

Barrel Jack. The module itself can be powered with a value between 7V and 12V and has two voltage regulators that allow 3.3V and 5V outputs. Thus, this detail allows the user to power the *MB102* module through the *Barrel Jack* input and supply the *HM-19* module with the 5V output. The image below shows the connected parts.

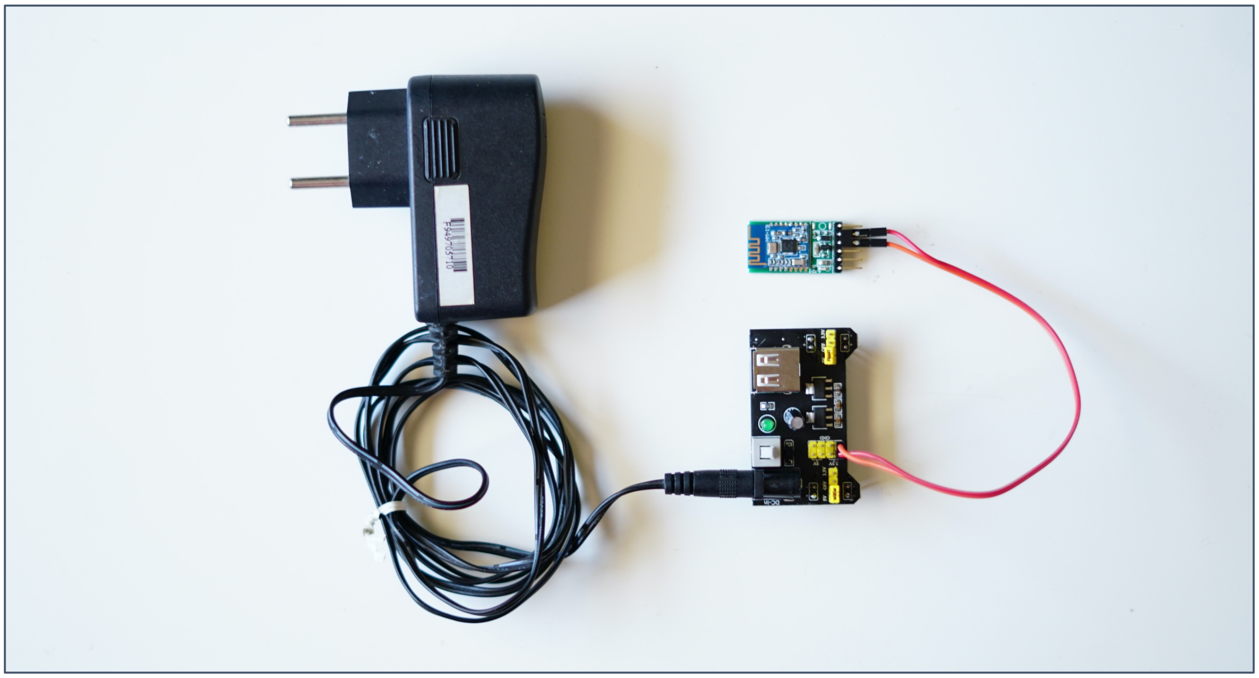


Figure 3-8. An option to power the HM-19 module into the outlet using an MB102 module. The only two pins used are VCC and GND. The power adapter used has a 9V, 1A output.

3.1 BONUS: The easiest way to configure a Ciudad Vestigio Device

Although explained step by step, it is understood that the configuration of the *Ciudad Vestigio Device* can be challenging for those who do not have a more solid computer background. Aware of this situation, this work presents the easiest way to configure a *Ciudad Vestigio Device*: using a smartphone with an app that simulates the signal of a *beacon* device. This approach is not ideal or encouraged - after all you would use a powerful hardware for a very "simple" activity - but this project is more about encouraging collaboration than creating technical barriers to users or institutions who want to share content. From these reflections, this "super easy" option is presented as an alternative. For *iOS* the app *BLE Scanner 4.0* [143] and for *Android* the app *Beacon*

Simulator [144] are both able to simulate *beacon* devices which are recognizable by the *Ciudad Vestigio Mobile Application*. After registering the device with the *Ciudad Vestigio Web Application*, the user just needs to add the *UUID*, *Major* and Minor values to the app before turning on the *Bluetooth* signal. It is important to remember that the smartphone must have at least version 4.0 of *Bluetooth*.

Chapter 4 – Web Application: System Development

The *Ciudad Vestigio Web Application* is responsible for the registration of devices within the *Ciudad Vestigio* platform. With this technology, the institution or user interested in sharing content about a city in the project can create an account and, from this user registration, start to add and manage devices. The process of registering devices involves not only the information about the hardware itself, but also information about the respective *Ciudad Vestigio VoiceBot* associated with it, as will be addressed in this chapter and in chapter 6.

This mechanism of registration is a two-way operation. In the same process that the *Ciudad Vestigio Web Application* stores the data provided about a device in a *MongoDB Atlas* database, the application also calculates the *UUID*, *Major* and *Minor* values that the user must send to the device in order to make it recognizable by the *Ciudad Vestigio* platform. This is a very important step in the dynamics of the platform because it connects the hardware layer – by returning to the user the calculated values – with the mobile layer - since the *Ciudad Vestigio Mobile Application* consults the same database of the *Ciudad Vestigio Web Application* to check which devices are part of the platform[50]. In this way, the *Ciudad Vestigio Web Application* is the official gateway for devices to become part of the *Ciudad Vestigio* platform.

As discussed in chapter 2, the code available for the technology was developed using the *Python* programming language in combination with the *Flask* microframework and the data generated by this application is stored in a database created with *MongoDB Atlas*. To run the available code, the first step is to install the requirements of the project. It can be done using *pip*, the package installer for *Python* [145], with the line[51] below. It must be executed in a terminal window inside the project folder. It is important to mention that the project was developed and executed using the *venv* module [146] to create virtual environments. If this is the chosen option, it is necessary to create it before installing the requirements.

[50] This mechanism will be covered in more detail in the next chapter.

[51] The steps were done using a computer with *macOS* operating system.

```
pip install -r requirements.txt --upgrade
```

Figure 4-1. How to install the project requirements.

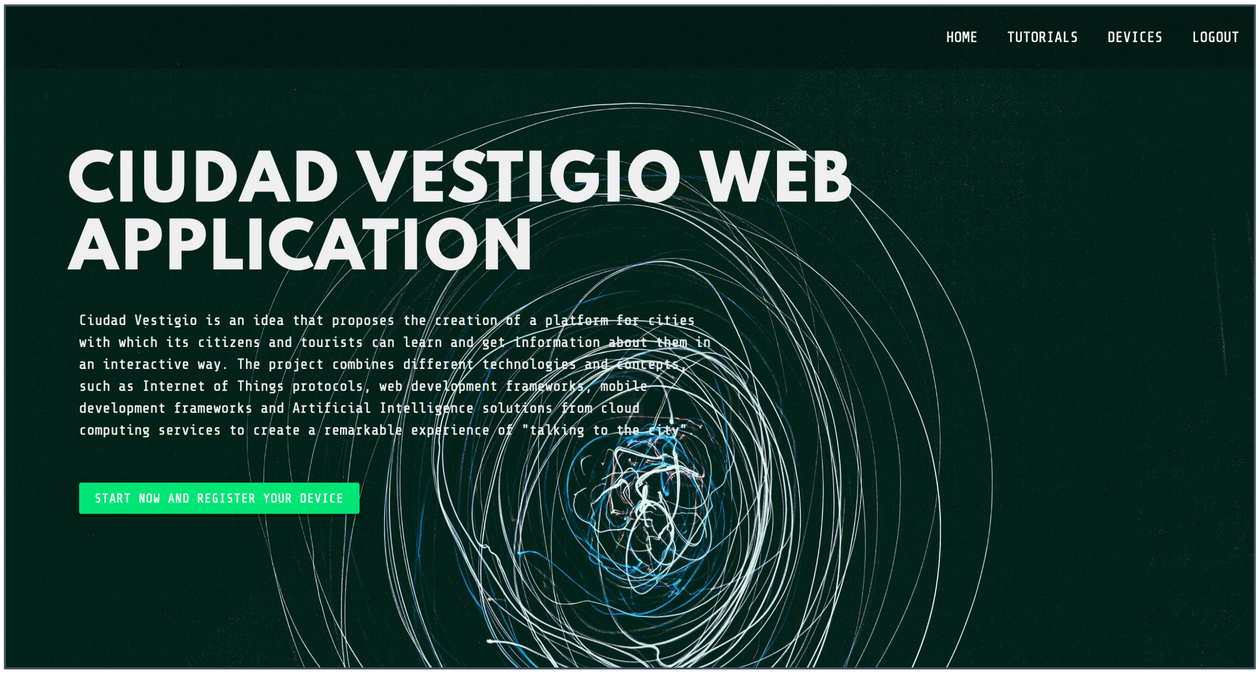
After the installation, it is possible to run the project with the line below, which also must be executed in the terminal window inside the project folder. The code available does not come with the original data from the "config.cfg" file because it contains sensitive information about database credentials and host server. However, it can be easily filled with the corresponding values.

```
python3 run.py
```

Figure 4-2. Command to execute the available project code.

In this first version, the *Ciudad Vestigio Web Application* was designed with the layout shown in the image below. To improve the front-end development of the application, the *Materialize* CSS framework [147] was used. It is based on the *Material Design*[52] created by *Google*. Back to the system, it is fully functional, only the textual content has not been totally created. It is important to mention that the most important functions and rules of the *Ciudad Vestigio Web Application* can be consulted in the *app.py* file of the available code.

[52] https://material.io

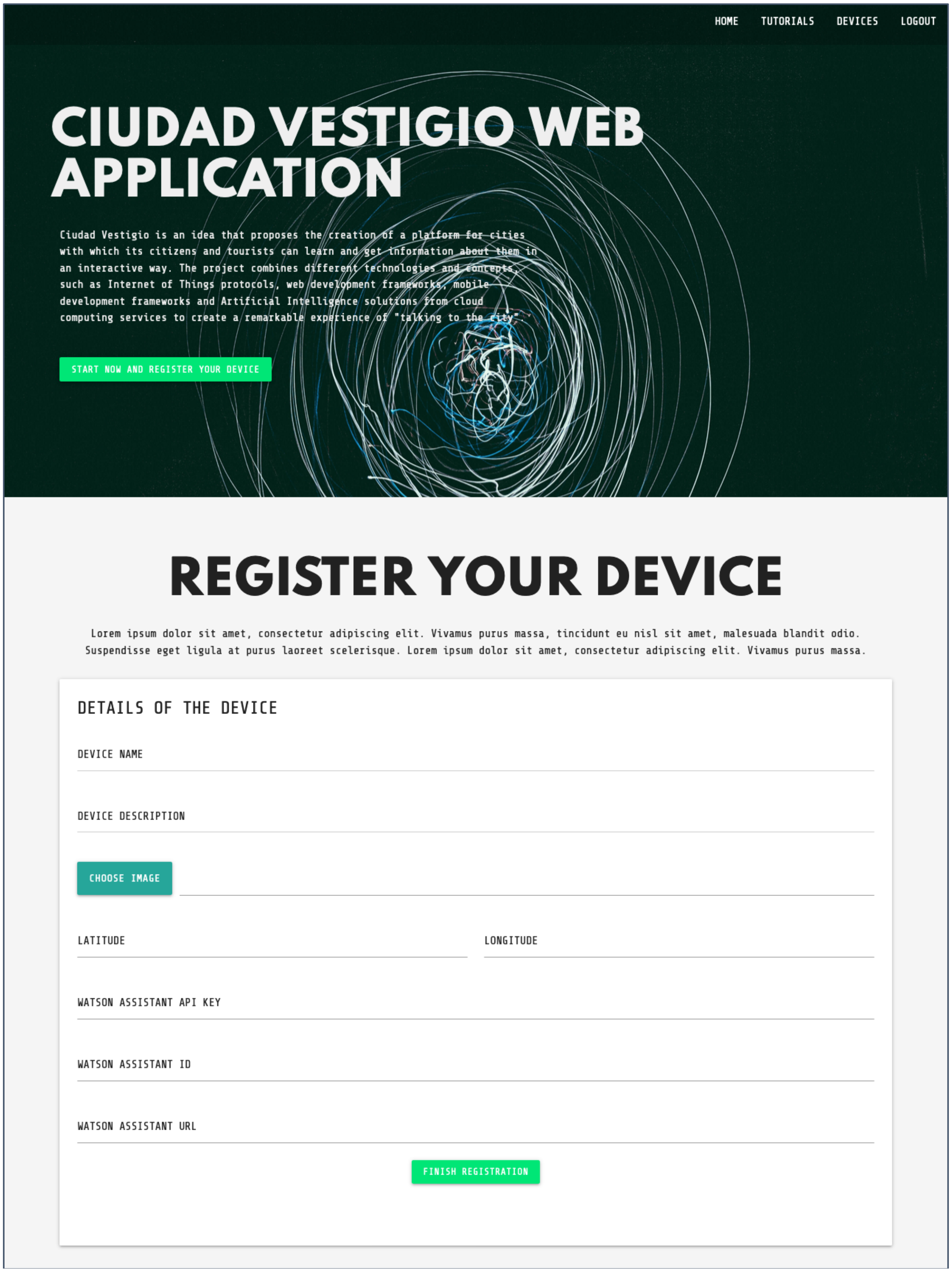

HOME
TUTORIALS
DEVICES
LOGOUT
CIUDAD VESTIGIO WEB APPLICATION
START NOW AND REGISTER YOUR DEVICE

Ciudad Vestigio is an idea that proposes the creation of a platform for cities with which its citizens and tourists can learn and get information about them in an interactive way. The project combines different technologies and concepts, such as Internet of Things protocols, web development frameworks, mobile development frameworks and Artificial Intelligence solutions from cloud computing services to create a remarkable experience of "talking to the city".

Figure 4-3. Screenshot of the Ciudad Vestigio Web Application homepage. After the login, new content is revealed and the menu options change.

If the user is not logged in, the page shows the following sections: *home*, *tutorials*, *sign up* and *login*. The home page aims to bring the "welcome message" of the *Ciudad Vestigio* project with a short "about" text.

The logic behind user registration and login was implemented using a *Flask* extension called *Flask-Security* [148]. It manages the necessary routes ("/register" and "/login"), procedures and communication with the database according to safety standards. If the user creates an account and logs in with the correct credentials, the system reveals the device registration section in the same main page and a new menu option called *devices* to manage registered devices.



CIUDAD VESTIGIO WEB APPLICATION

Ciudad Vestigio is an idea that proposes the creation of a platform for cities with which its citizens and tourists can learn and get information about them in an interactive way. The project combines different technologies and concepts, such as Internet of Things protocols, web development frameworks, mobile development frameworks and Artificial Intelligence solutions from cloud computing services to create a remarkable experience of "talking to the city".

REGISTER YOUR DEVICE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa, tincidunt eu nisl sit amet, malesuada blandit odio. Suspendisse eget ligula at purus laoreet scelerisque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa.

DETAILS OF THE DEVICE

DEVICE NAME

DEVICE DESCRIPTION

LATITUDE

LONGITUDE

WATSON ASSISTANT API KEY

WATSON ASSISTANT ID

WATSON ASSISTANT URL

Figure 4-4. Screenshot of the Ciudad Vestigio Web Application homepage after logging in. The menu at the top right changes and the page reveals (in a very direct way) the device registration form.

The registration of devices requires the following data: *name*, *description*, *image*, *latitude* and *longitude* of where the device will be positioned in the city. The other three data to be filled are related to the *Ciudad Vestigio VoiceBot* created with *IBM Cloud*: *Watson Assistant API* key, *Watson Assistant ID* and *Watson Assistant URL*. All of these values are provided after the creation of a cognitive assistant, a process which will be covered in detail in chapter 6.

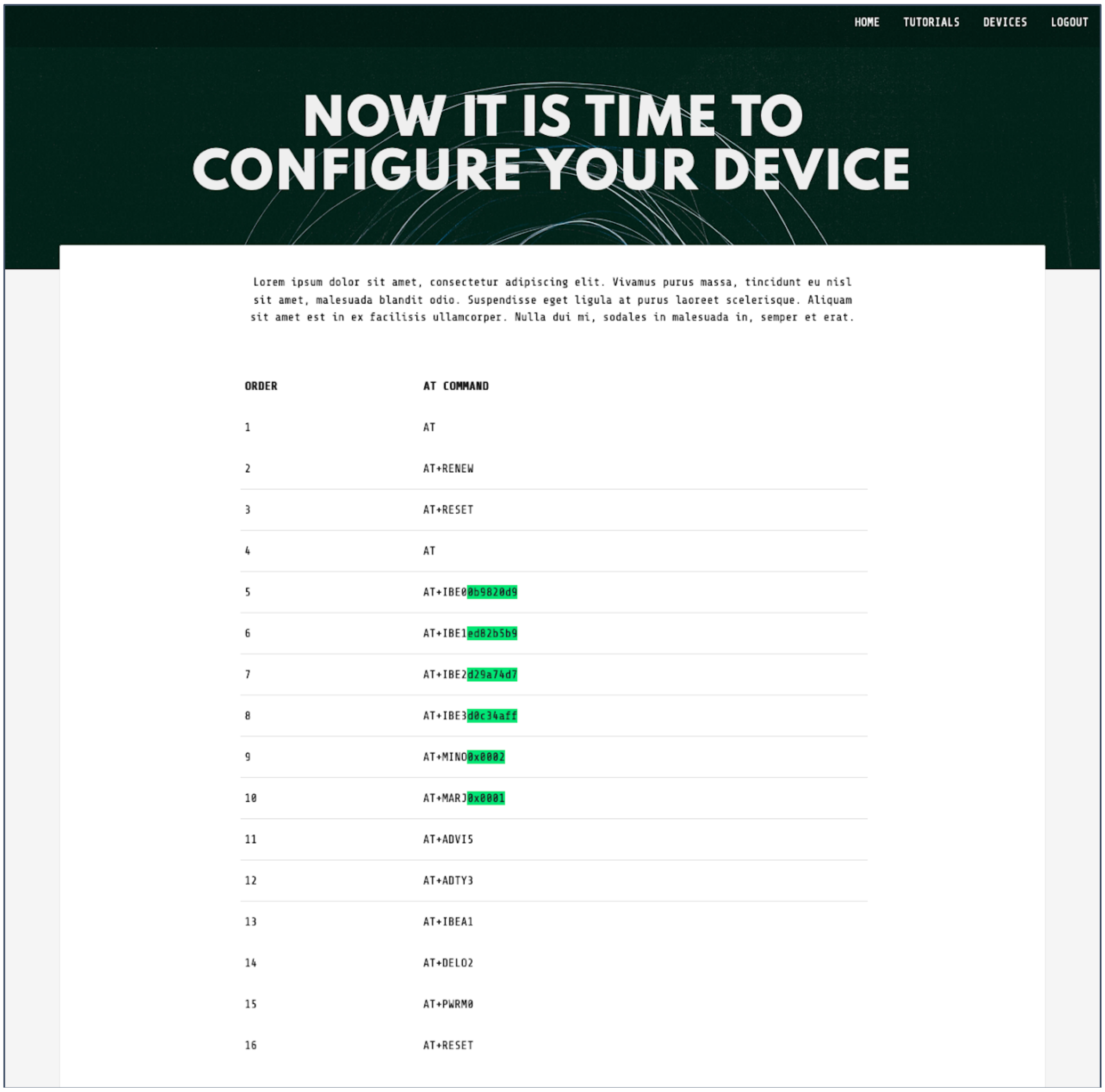
As will be presented in the next chapter, the first three elements of the form - *name*, *description* and *image* - compose the cards of found devices which are presented by the *Ciudad Vestigio Mobile Application* to the users. The latitude and longitude data are extremely important because they act as indexes [52] [149] for the geospatial queries made by the mobile application to the *MongoDB database*. As will be presented, the application initially queries for registered devices which are in a radius of 3 kilometers from the user[53]. For the user or institution interested in registering a device, the task of obtaining the latitude and longitude values is not complex, since famous tools such as *Google Maps* can help with this [150].

If the registration process is completed correctly by the user, the system shows a confirmation page. It is important to mention that all fields on the form are indicated as "required" in the data mapper for *MongoDB* used to manage the connection with the database. The tool is called *MongoEngine* [151] and the steps involving the creation of the *MongoDB Atlas* database will be covered shortly.

The confirmation page is an important element because it presents the *UUID*, *Major* and *Minor* values that the user needs to send to the device in order to make it recognizable by the *Ciudad Vestigio* platform. The table in the image below shows the same *AT Commands* addressed in the last chapter, where the digits highlighted in green are included dynamically by the system and correspond to the correct data generated by the calculation performed for a fictional example user. As previously mentioned, beyond the table the textual content has not been created, but the idea is to give a

[53] This prevents the application from receiving more values from the database than necessary, generating useless data traffic and losing performance.

step-by-step tutorial of how to configure the *Ciudad Vestigio Device* properly. In a real context, this is the moment where the user or institution can repeat the steps presented in the previous chapter, sending the sequence of *AT Commands* and testing the correct functioning of the device with one of the suggested mobile applications.



NOW IT IS TIME TO CONFIGURE YOUR DEVICE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa, tincidunt eu nisl sit amet, malesuada blandit odio. Suspendisse eget ligula at purus laoreet scelerisque. Aliquam sit amet est in ex facilisis ullamcorper. Nulla dui mi, sodales in malesuada in, semper et erat.

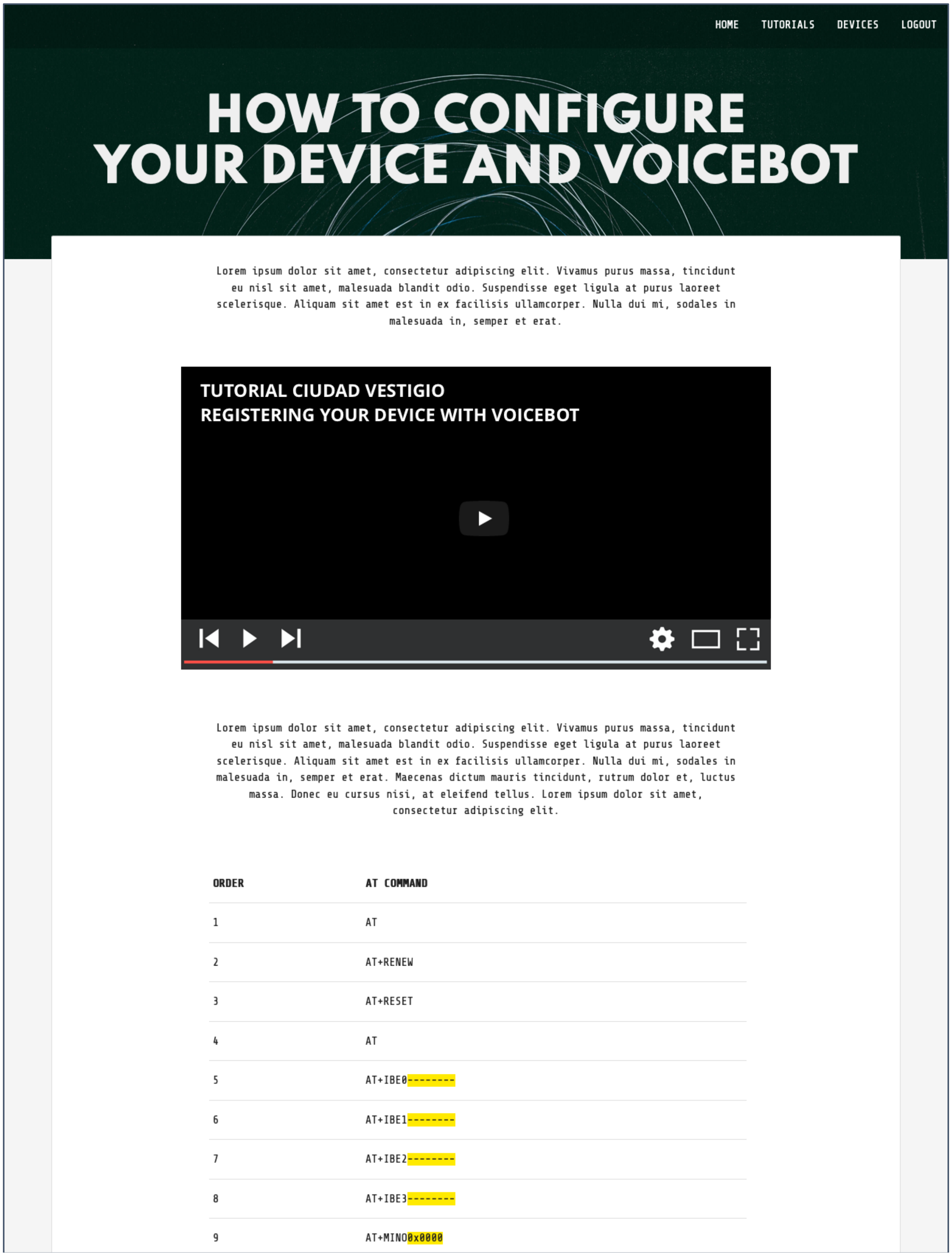
ORDER	AT COMMAND
1	AT
2	AT+RENEW
3	AT+RESET
4	AT
5	AT+IBE00b9820d9
6	AT+IBE1ed82b5b9
7	AT+IBE2d29a74d7
8	AT+IBE3d0c34aff
9	AT+MINO0x0002
10	AT+MARJ0x0001
11	AT+ADVI5
12	AT+ADTY3
13	AT+IBEA1
14	AT+DELO2
15	AT+PWRM0
16	AT+RESET

Figure 4-5. Screenshot of the "confirmation" page. It is shown to the user after completing the device registration process. The data highlighted in green is the actual result of an example calculation of UUID, Major and Minor values that, in a real context, the user must add to the device.

As discussed in chapter 2, *Apple*'s guidelines for the *iBeacon* protocol indicate that application developers should define a specific *UUID* to their app and deployment use case. This means that all devices that are part of the *Ciudad Vestigio* platform must share the same *UUID* and what differentiates these devices from each other are the values of *Major* and *Minor*. The *UUID* contains 32 hexadecimal digits, split into 5 groups, separated by hyphens. In the example shown with the previous image, the *Ciudad Vestigio* platform UUID would be "0b9820d9-ed82-b5b9-d29a-74d7d0c34aff". In the example the *Major* values is "0x0001" and the *Minor* value is "0x0002".

For this first version of the *Ciudad Vestigio Web Application*, the system associates a *Major* value for each registered user, which means that all devices of the same user share the same *Major value*. Then for each device registered by the user an incremental *Minor* value is associated with it. The *Major* and *Minor* are unsigned integer values between 0 and 65535. This means that a single institution or user can add more than sixty thousand devices. This logic of calculating and assigning the *UUID*, *Major* and *Minor* is just a first approach for this first prototype of the project, and it will be optimized in future versions. The function responsible for assigning these values is located in the *app.py* file of the available code.

The next section of the website, *Tutorials*, aims to provide the necessary guidelines in text, images and video for the correct configuration - from start to end - of the *Ciudad Vestigio Device*. This goes from the configuration of the device itself - as seen in the previous chapter - to the configuration of the *Ciudad Vestigio VoiceBot* associated with this device, a topic which will be covered in chapter 6. Once again, the textual content has not been created since the focus is on the development of the system. The idea behind this section is to turn it into a reference spot in terms of documentation for those users and institutions interested in configuring devices properly. The development of this documentation is part of the future work addressed in chapter 7.



HOW TO CONFIGURE YOUR DEVICE AND VOICEBOT

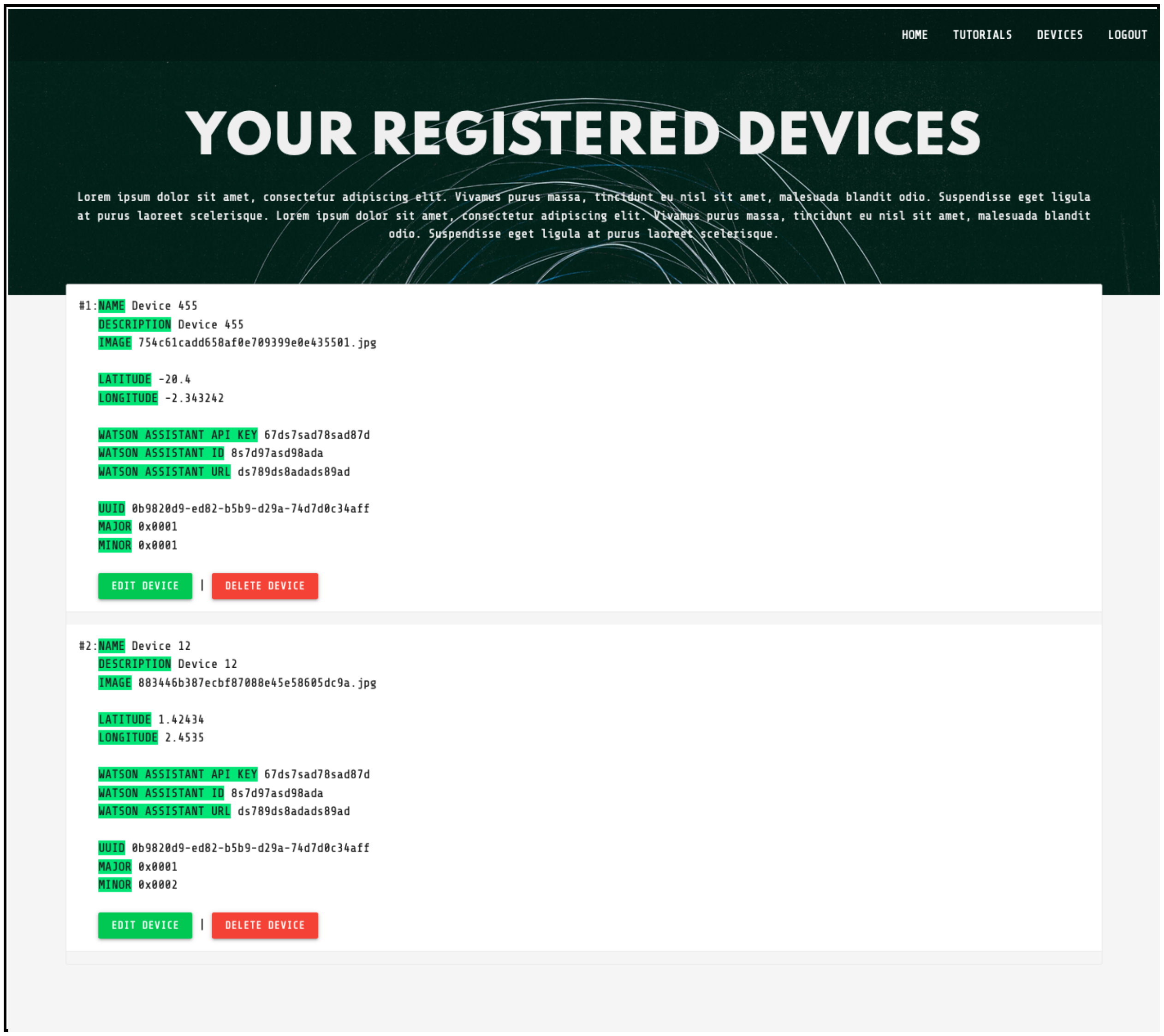
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa, tincidunt eu nisl sit amet, malesuada blandit odio. Suspendisse eget ligula at purus laoreet scelerisque. Aliquam sit amet est in ex facilisis ullamcorper. Nulla dui mi, sodales in malesuada in, semper et erat.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa, tincidunt eu nisl sit amet, malesuada blandit odio. Suspendisse eget ligula at purus laoreet scelerisque. Aliquam sit amet est in ex facilisis ullamcorper. Nulla dui mi, sodales in malesuada in, semper et erat. Maecenas dictum mauris tincidunt, rutrum dolor et, luctus massa. Donec eu cursus nisi, at eleifend tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

ORDER	AT COMMAND
1	AT
2	AT+RENEW
3	AT+RESET
4	AT
5	AT+IBE0--------
6	AT+IBE1--------
7	AT+IBE2--------
8	AT+IBE3--------
9	AT+MINO0x0000

Figure 4-6. Screenshot of the "Tutorials" page in a simplified version. Once the content created, the idea is that this page gathers all the necessary documentation for the users.

The last section to introduce is called *Devices*. This is an administration area where the user can view the set of registered devices, edit details and delete them if necessary. Each card corresponds to a device and shows the information given during the registration process: *name*, *description*, *image filename*, *latitude*, *longitude* and the credentials used by the *Ciudad Vestigio Mobile Application* to connect to the *IBM Cloud API*. The cards also bring the group of data calculated by the system for consultation: *UUID*, *Major* and *Minor*. The image below shows the devices of the example user.



YOUR REGISTERED DEVICES

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa, tincidunt eu nisl sit amet, malesuada blandit odio. Suspendisse eget ligula at purus laoreet scelerisque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus purus massa, tincidunt eu nisl sit amet, malesuada blandit odio. Suspendisse eget ligula at purus laoreet scelerisque.

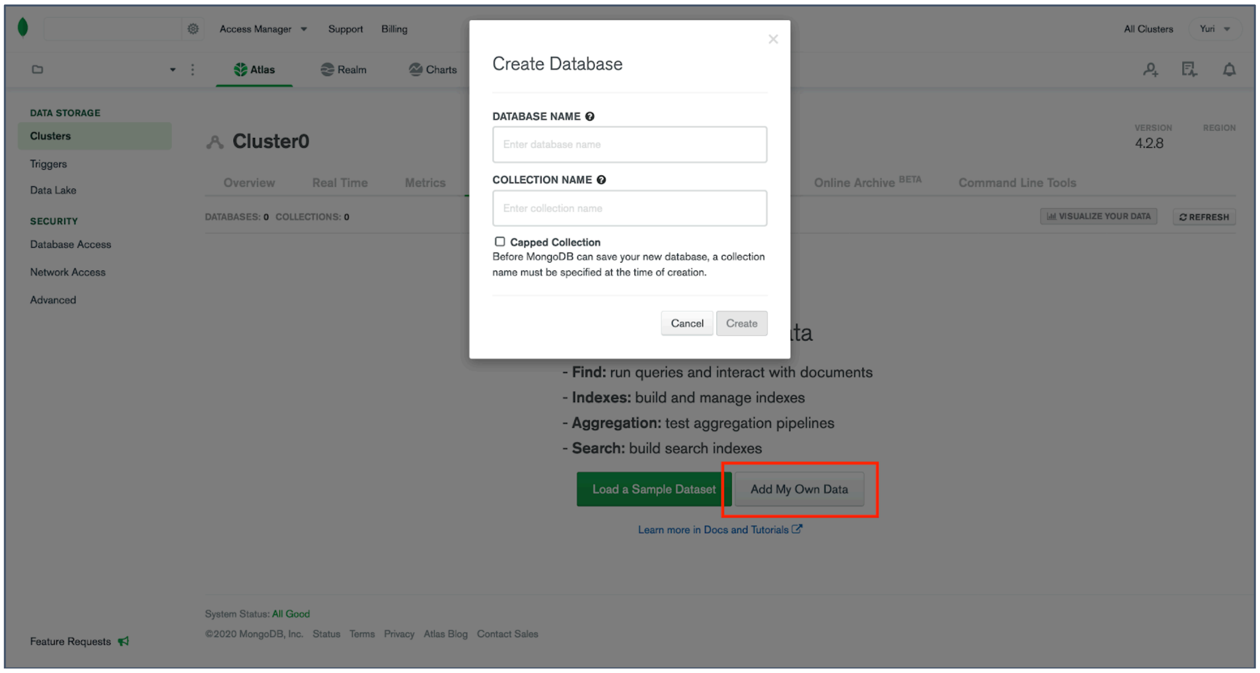
```
#1:NAME Device 455
   DESCRIPTION Device 455
   IMAGE 754c61cadd658af0e709399e0e435501.jpg

   LATITUDE -20.4
   LONGITUDE -2.343242

   WATSON ASSISTANT API KEY 67ds7sad78sad87d
   WATSON ASSISTANT ID 8s7d97asd98ada
   WATSON ASSISTANT URL ds789ds8adads89ad

   UUID 0b9820d9-ed82-b5b9-d29a-74d7d0c34aff
   MAJOR 0x0001
   MINOR 0x0001

#2:NAME Device 12
   DESCRIPTION Device 12
   IMAGE 883446b387ecbf87088e45e58605dc9a.jpg

   LATITUDE 1.42434
   LONGITUDE 2.4535

   WATSON ASSISTANT API KEY 67ds7sad78sad87d
   WATSON ASSISTANT ID 8s7d97asd98ada
   WATSON ASSISTANT URL ds789ds8adads89ad

   UUID 0b9820d9-ed82-b5b9-d29a-74d7d0c34aff
   MAJOR 0x0001
   MINOR 0x0002
```

Figure 4-7. Screenshot of the "Devices" page, where the user or institution can see the registered devices and manage them by editing the information or deleting the device from the platform.

4.1 MongoDB Atlas Basic Configuration

As presented in chapter 2, this entire process of adding, editing, deleting and retrieving devices is made in communication with a *MongoDB Atlas* cluster. The process of creating a cluster is very straightforward. After registering an account on the official page of *MongoDB Atlas*[54], the first step is to create a project and then create a cluster. As previously mentioned, the free plan will be used which works with shared clusters [152]. In this process it is possible to choose the service provider: *Amazon Web Services*, *Google Cloud Platform* or *Microsoft Azure*. For no special reason, the first option was chosen with the *eu-west-1* region, located in Ireland.

From that moment it is already possible to create databases and collections by clicking on "collections" inside the cluster page and pressing the "Add My Own Data" button. A popup window will appear as shown in the image below.



Create Database

DATABASE NAME

Enter database name

COLLECTION NAME

Enter collection name

Capped Collection

Before MongoDB can save your new database, a collection name must be specified at the time of creation.

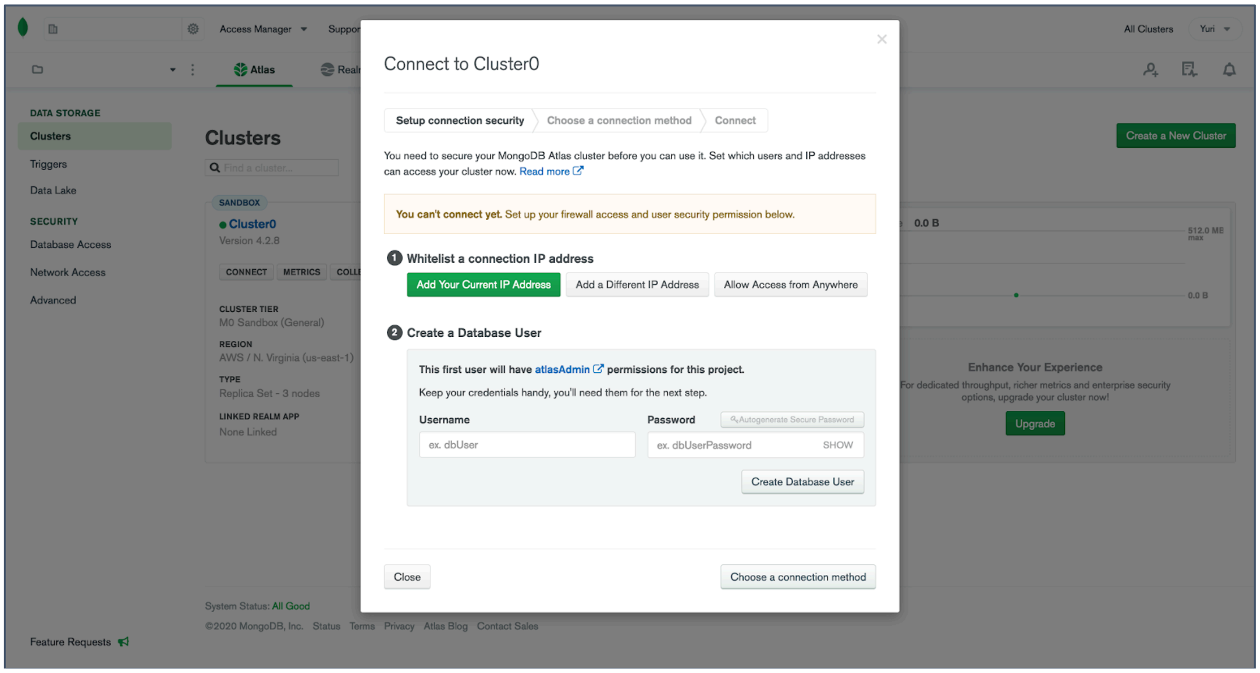
Cancel Create

- **Find:** run queries and interact with documents
- **Indexes:** build and manage indexes
- **Aggregation:** test aggregation pipelines
- **Search:** build search indexes

Figure 4-8. Screenshot of the MongoDB Atlas webpage that shows the first step to create a database after creating the cluster.

54 https://www.mongodb.com/cloud/atlas

The *Flask* extension used to interact with the database is *Flask-MongoEngine*[55], which, in turn, provides integration with the already mentioned *MongoEngine*[56]. After the cluster and database are created, *MongoEngine* can use a "connection string" to establish the connection between the *Ciudad Vestigio Web Application* and *MongoDB Atlas*. To obtain this string, it is necessary to press the "connect" button on the main page of the cluster. A popup window will appear and from this window it is necessary to set which users and IP addresses can access the cluster.



You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

You can't connect yet. Set up your firewall access and user security permission below.

This first user will have atlasAdmin permissions for this project.

Keep your credentials handy, you'll need them for the next step.

For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!

System Status: All Good

©2020 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Figure 4-9. Screenshot of the MongoDB Atlas webpage that shows the necessary steps to connect the cluster to external applications, in our case the Ciudad Vestigio Web Application. It is possible to indicate allowed IPs (or allow access from anywhere) and it is necessary to create credentials for the database.

After pressing the "Choose a connection method" button it is possible to obtain the "connection string" to establish the communication between the cluster and the *Ciudad Vestigio Web Application*. This string is added to the "config.cfg" file according to the *Flask-MongoEngine* requirements [153].

---

[55] http://docs.mongoengine.org/projects/flask-mongoengine/en/latest

[56] http://mongoengine.org

It is important to mention that *MongoEngine* allows to define schemata for documents as this helps to "reduce coding errors, and allows for utility methods to be defined on fields which may be present" [154]. This resource was used in this project, as can be seen in the *models.py* file of the available code. Any need for change is easily handled by changing this file.

If the steps are performed correctly it will be possible to establish the connection between the *Ciudad Vestigio Web Application* and the database created with *MongoDB Atlas*. From the application it is possible to create collections, documents, etc. Any operation made with the *Ciudad Vestigio Web Application* involving users or devices on the platform are stored in the cluster and this same data will be consulted by the *Ciudad Vestigio Mobile Application* with the help another resource, *MongoDB Stitch*. This mechanism will be covered in more detail in the next chapter.

Chapter 5 – Mobile Application: System Development

As presented in the introduction of this work, the *Ciudad Vestigio Mobile Application* is responsible for scanning the environment periodically for devices which are registered as part of the *Ciudad Vestigio* platform. If one of them is found and the user decides to interact with it, the application activates the *Ciudad Vestigio VoiceBot* that corresponds to the chosen device. Then the experience of talking to the element of the city begins. To recap some important topics, these *Bluetooth Low Energy* devices are configured as *beacons* and have a unique combination of *UUID*, *Major* and *Minor values* which identify them within the project. As addressed in the last chapter, these devices need to be registered in the platform with the *Ciudad Vestigio Web Application*, which is why these devices are known for the *Ciudad Vestigio Mobile Application*.

In chapter 2, the *React Native* mobile development framework was presented as the chosen tool for developing the *Ciudad Vestigio Mobile Application*. Although the framework allows the creation of apps for more than one system, this work focuses on the development of the *Ciudad Vestigio Mobile Application* for only one operating system. The main objective, after all, is to develop the entire platform with its four technologies - not to demonstrate the potential of *React Native*. Given this context, the operating system chosen to develop the *Ciudad Vestigio Mobile Application* in this first version is *Android*. The reason is directly related to the reach it has in the smartphone market, used in at least 80% of the smartphones in the world [155].

Considering that the application is not yet available on *Google Play*[57], it is necessary to run the available code in an *Android* smartphone in order to test it. It is not possible to run the application in the *Android Emulator*, since the app uses *Bluetooth* features that cannot be performed by the *Android Emulator* [156]. To run the application in an *Android* smartphone, the device must be with the "USB Debugging" option enabled inside "Developer options" [157] and it needs to be connected to a computer via *USB*. When connecting both, the *USB* debugging must be authorized in the smartphone. To

[57] Digital platform for distribution of Android mobile applications: https://play.google.com/store

develop this project, two smartphones from "different generations" were used in order to test the *Ciudad Vestigio Mobile Application*: a *Samsung A30s* released in 2019 and a *Samsung Galaxy Note 4* released in 2014.

Additionally, it is necessary to set up a development environment in the computer in order to run the code. The *React Native CLI* was used, which requires *Android Studio*[58] to get started. Even if a developer wants to use another code editor to develop an app, which was the case, the installation of *Android Studio* is required in order to set up the necessary details to build the *React Native* app for the *Android* operating system. It is important to mention that the instructions for setting up the development environment are different depending on the computer's operating system and whether the developer wants to start the development for *iOS* or *Android*. In the case of *Android*, the common dependencies for *macOS*, *Windows* and *Linux* are currently *Node.js*[59], the *React Native* command line interface, a *Java Development Kit* [158] and the already mentioned *Android Studio*. The *macOS* specifically demands the installation of *Watchman* - a tool created by *Facebook* for watching changes in the filesystem, recommended for better performance - and *Windows* demands the installation of *Python 2* [159]. Regardless of the operating system, the details for each case are presented in the official website of *React Native* [160], where the installation instructions are very clear and easy to follow.

The first step to run the available code of the *Ciudad Vestigio Mobile Application* is to start *Metro*, the *JavaScript* bundler that ships with *React Native*. *Metro* "takes in an entry file and various options, and gives you back a single JavaScript file that includes all your code and its dependencies" [161]. According to the *React Native* documentation:

> *If you're familiar with web development, Metro is a lot like webpack - for React Native apps. Unlike Kotlin or Java, JavaScript isn't compiled - and neither is React Native. Bundling isn't the same as compiling, but it can help improve startup performance and translate some platform-specific JavaScript into more JavaScript.* [160]

[58] https://developer.android.com/studio

[59] https://nodejs.org

To start *Metro*, it is necessary to run the code below in the terminal inside the project folder of the *Ciudad Vestigio Mobile Application*.

```
npx react-native start
```

Figure 5-1. Command to start Metro. It must be executed in a terminal window inside the project folder.

To finally run the *Ciudad Vestigio Mobile Application*, it is necessary to open a new terminal window inside the project folder and run the line below.

```
npx react-native run-android
```

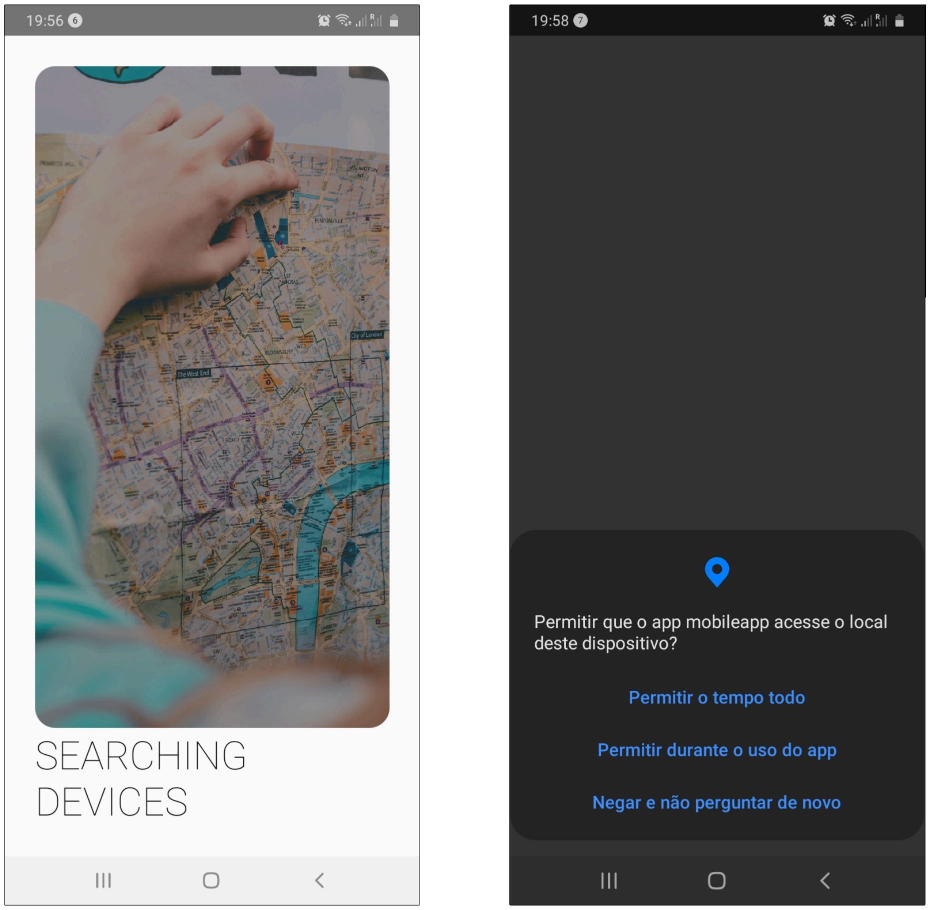
Figure 5-2. Command to run the Ciudad Vestigio Mobile Application.

If everything is set up correctly, the application should run on the *Android* smartphone shortly. In both mentioned commands it is important to explain the presence of the term "npx".

> *React Native has a built-in command line interface. Rather than install and manage a specific version of the CLI globally, we recommend you access the current version at runtime using npx, which ships with Node.js. With npx react-native <command>, the current stable version of the CLI will be downloaded and executed at the time the command is run.* [160]

Whereas *React Native* is a young technology with updates that can be released in relatively short intervals, the use of *npx* during the development stage of this project was an important detail to force the application to always comply with the latest stable version of the *React Native CLI*.

As presented in the introduction of this document, the *Ciudad Vestigio Mobile Application* is aimed at the user who wants to consume content from the platform, that is: the person interested in interacting with the city in order to learn and get information about it. Based on this statement, the user journey presented on the following pages will be fully connected to the experience of this specific user.



Permitir que o app mobileapp acesse o local deste dispositivo?

Permitir o tempo todo

Permitir durante o uso do app

Negar e não perguntar de novo

Figure 5-3. On the left, screenshot of the Ciudad Vestigio Mobile Application home screen. On the right, the application requests access to the device's location if this is the first time the app is being executed.

The *Ciudad Vestigio Mobile Application* starts with a device search screen, as can be seen in the previous image, left side. The photo flashes slowly in order to indicate that the search is in progress, an effect that cannot be reproduced by this document. If this is the first time the user has opened the application, it first asks for authorization to use the device's location, as can be seen in the previous image, right side.

The use of the device's *Bluetooth* signal is an essential feature behind the proper operation of the *Ciudad Vestigio Mobile Application*. If the user does not have the feature activated, the application shows a different screen informing that it is necessary.

To simplify the number of screens designed, the same screen layout is used in case the user has not authorized the location of the device.

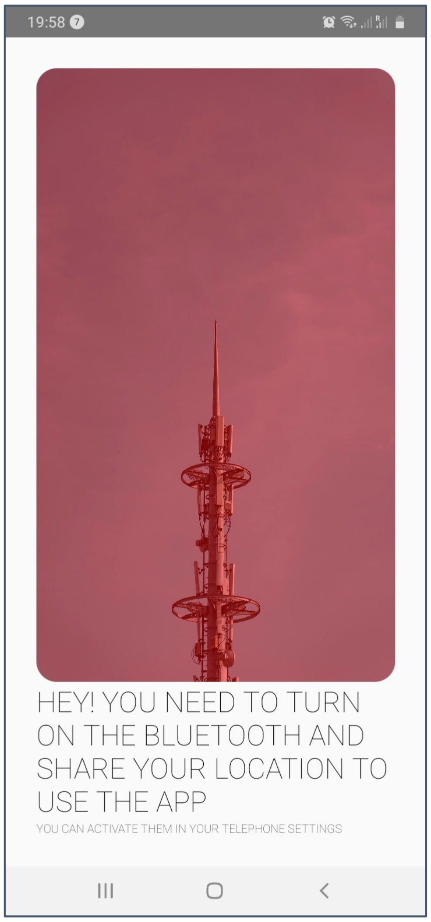

19:58
HEY! YOU NEED TO TURN ON THE BLUETOOTH AND SHARE YOUR LOCATION TO USE THE APP
YOU CAN ACTIVATE THEM IN YOUR TELEPHONE SETTINGS

Figure 5-4. Screenshot of the layout presented if the user does not have the smartphone's Bluetooth signal activated or if the user has not authorized access to the smartphone's location.

The *Bluetooth* usage verification is done using an external library, called *react-native-bluetooth-status* [162], obtained from *npm*. The term *npm* stands for *Node Package Manager* and means two different but connected things. The first one is an online repository for the publishing of open-source *Node.js* projects. The second one is a command-line utility for interacting with the mentioned repository, which aids in "package installation, version management, and dependency management" [163]. As a company, *npm* was founded in 2014 and was acquired by *GitHub* in 2020, which is now part of *Microsoft*. The project has an important role in the *JavaScript* community and "helps support one of the largest developer ecosystems in the world" [164]. The libraries and applications published on *npm* can be searched on the official website [165] and with a single command-line command it is possible to install a package. Another important use for *npm* is dependency management - which can be used when testing the *Ciudad Vestigio Mobile Application*. When there is a *Node.js* project with a *package.json* file, it is possible to run the command "npm install" inside the project directory to install all the dependencies listed in that file.

Back to the *Bluetooth* library used in the application, it allows the developer to add a listener to verify the use of the feature by the smartphone. It is an important detail because if the user disables the *Bluetooth* signal at some point during the application's execution, the warning screen is displayed. This immediate screen change is the result of a change in a state variable of the component [166]. In the case of the device's location, React Native's *PermissionsAndroid* [167] was used. Both features - the use of the device's location and the use of *Bluetooth* - are declared in the *AndroidManifest.xml* file of the available code.

With the authorization to use the device's location, the *Ciudad Vestigio Mobile Application* queries the *MongoDB Atlas* database using *MongoDB Stitch*. This is done using the official *MongoDB* library available in *npm* for this purpose - called *MongoDB Stitch React Native SDK* [168] - and the steps to set the integration between *React Native* and *MongoDB* are available in the official blog of *MongoDB* [169]. The implementation is done in the *mongodbstitch.js* file of the available code: using the latitude and longitude values of the smartphone, the function *getDevices* queries the database for *Ciudad Vestigio* Devices within a 3 km radius from the user. There are two possible results for this operation: the database is successful in finding devices registered on the platform within a radius of 3km or the database does not find anything in that region where the user is. If this is the case, a different new screen is presented to inform the user that there are no devices registered on the *Ciudad Vestigio* platform near the location. The message brings an invitation to keep moving to try to find them. The application, in turn, makes periodic queries to the database every 5 minutes to update the list of nearby devices[60].

[60] This is an initial approach to the feature that must be optimized in future versions.

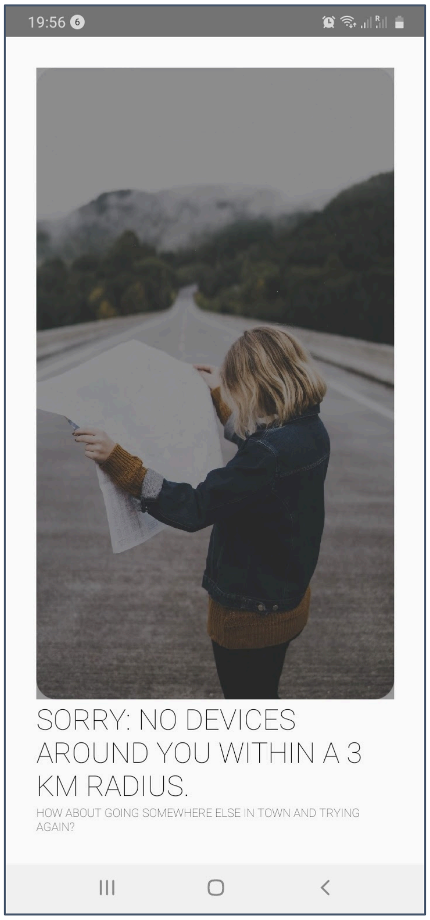
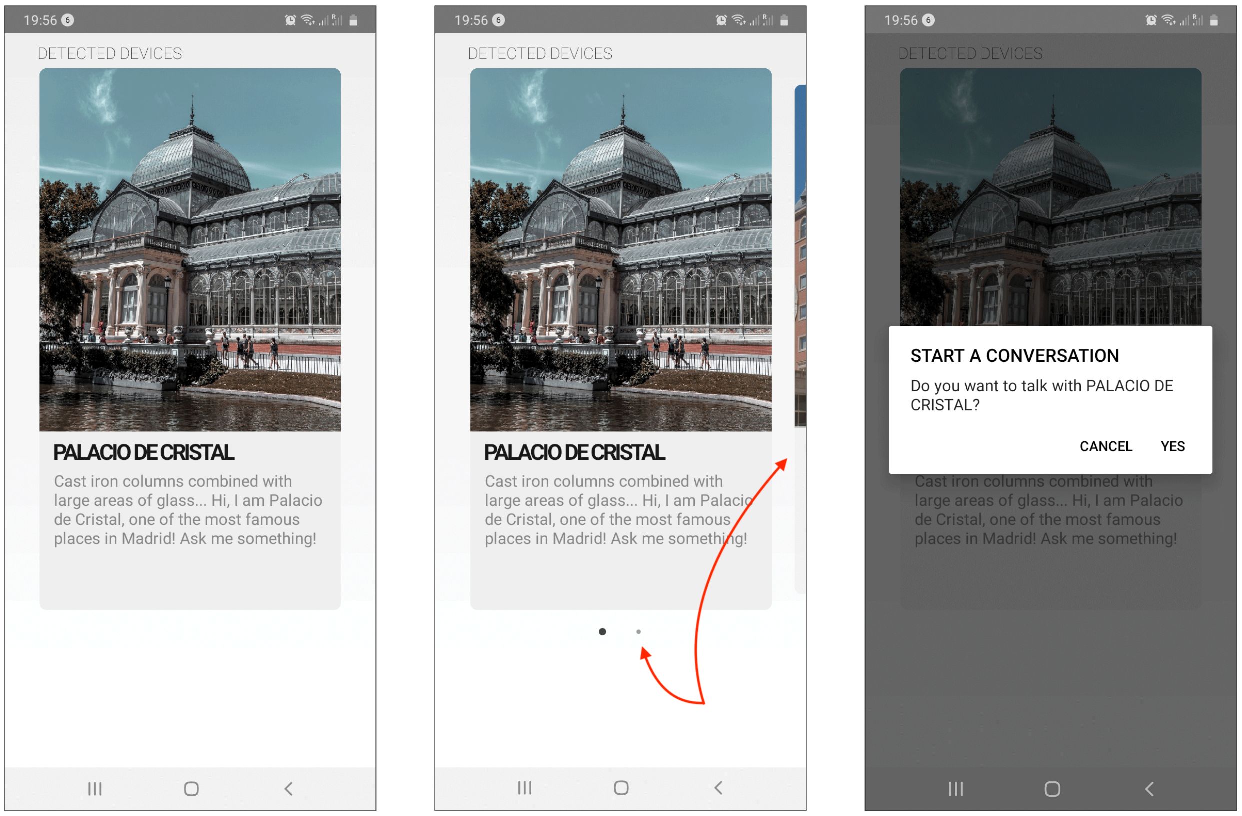

19:56
SORRY: NO DEVICES AROUND YOU WITHIN A 3 KM RADIUS.
HOW ABOUT GOING SOMEWHERE ELSE IN TOWN AND TRYING AGAIN?

Figure 5-5. Screenshot of the layout presented to the user if there are no devices of the project within a radius of 3 km. The information about nearby devices is obtained by the application when consulting the database created with MongoDB Atlas.

If the database is successful in finding devices nearby the user, the *Ciudad Vestigio Mobile Application* initializes the functions to work with the *Bluetooth Low Energy* protocol, in order to recognize signals from the devices registered in the platform. This is done using an external library available in *npm*, called *react-native-beacons-manager* [170], which provides an *API* with a group of methods to configure and start the process of detecting *beacon* devices.

If the user is near a *Ciudad Vestigio Device* and its *Bluetooth Low Energy* signal is recognized by the *Ciudad Vestigio Mobile Application*, a new screen is shown to present the found device to the user. As it is possible for the user to be in the presence of more than one device from the platform at a time, it was decided to use a "carousel" of cards for user interface to present the devices nearby, as can be seen in the image below, in the screenshot positioned in the center. Each card presents the photo of the element that the device corresponds to, the name and the description, pieces of data that were registered by a user or institution before with the *Ciudad Vestigio Web Application*. This carousel and card structure were made using the external library *react-native-snap-carousel* [171], also available in the *npm* repository.

This is the screen with which the user can select the elements found in the city to start an interaction. The user just needs to click on the desired element and the application triggers a confirmation alert, as seen with the image below, right side.



Cast iron columns combined with large areas of glass... Hi, I am Palacio de Cristal, one of the most famous places in Madrid! Ask me something!

Cast iron columns combined with large areas of glass... Hi, I am Palacio de Cristal, one of the most famous places in Madrid! Ask me something!

Do you want to talk with PALACIO DE CRISTAL?

Cast iron columns combined with large areas of glass... Hi, I am Palacio de Cristal, one of the most famous places in Madrid! Ask me something!

Figure 5-6. Three screenshots of the moment when one or more devices are found by the app. The image on the left shows a situation where only one device was found. The image on the center shows a situation where two devices have been found and the user can slide the screen to see the next card. The image on the right shows the confirmation alert when a card is selected.

If the user confirms, the *Ciudad Vestigio Mobile Application* finally enters the interaction screen of the selected *Ciudad Vestigio VoiceBot*. Upon entering the screen, the user receives a welcome voice message from the *VoiceBot*, according to the message configured by those who created the element with *IBM Watson Assistant*, as will be presented in the next chapter. The screen presents a picture of the chosen element, the name of the element and a button to trigger interactions with it. When the user presses the button, a subtle sound indicates that the voice interaction has started. If this is the first time using the *Ciudad Vestigio Mobile Application*, the app will request authorization to use the smartphone's microphone, similarly to what happens with the device's location. This request is also made using React Native's *PermissionsAndroid* [167].

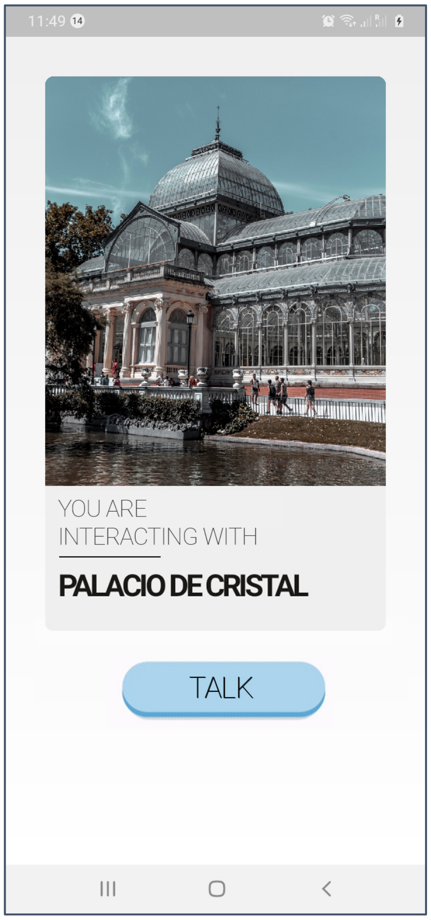


Figure 5-7. Screenshot of the layout used to interact with the chosen Ciudad Vestigio VoiceBot.

When the user finishes saying something (or saying nothing), a new subtle sound indicates that the *Ciudad Vestigio VoiceBot* has finished capturing audio. It then starts the process of interpreting what was said to give an answer as soon as possible. The audio recorded by the application is converted in text by an external library called *React Native Voice* [172]. It is very important to discuss this step because, at first, the idea was to execute it with the chosen cloud computing service for the development of the *Ciudad Vestigio VoiceBot*. Among the services offered by *IBM* - the chosen provider - there are *IBM Watson Speech to Text (STT)* [173] to do the first conversion and *IBM Watson Text to Speech (TTS)* [174] to perform the reverse procedure. In the middle of them there is *IBM Watson Assistant*, the cognitive piece of the process, which will be addressed in the next chapter. But once the existence of a library maintained by the community of *React Native* was discovered for the first "speech to text" conversion, after testing and getting good results, it was decided to make both conversions using *React Native* libraries to improve the application's performance and decrease the waiting time between questions and answers.

Considering the context in which the user is in the middle of the streets with access to the internet via 3G or 4G, the act of sending an audio to the cloud and waiting for another audio through the cloud whenever something is said could add an inconvenient latency to the user experience. Given this context, the use of libraries that fulfill the function locally on the smartphone was identified as the best solution and this was an important decision made during the development process of the *Ciudad Vestigio Mobile Application*.

Thus, in its current version, when the user says something to the *Ciudad Vestigio VoiceBot*, the recorded audio is converted to text locally by the mentioned *React Native* library. When converting an audio to text, the library generates a series of options of what may have been said by the user. The text of the option with the more confidence created by the library is then sent to *Watson Assistant* which also returns the answer in text format. The last step is to convert this response from text to audio and reproduce it to the user with the smartphone. This is done using another *React Native* library, called *React Native TTS* [175]. All the implementation and functions related to the *Ciudad Vestigio VoiceBot* inside the *Ciudad Vestigio Mobile Application* are in the *voicebot.js* file.

Regarding the journey of this specific user, it is important to note that no registration is required. The idea, as addressed in the introduction of this work, is that the process of consuming content should be as less bureaucratic as possible, where the users feel invited to use the platform as a way to learn and obtain information about the cities they live or visit.

Chapter 6 – VoiceBot: System Development

Among the four technologies presented in the introduction, the development of the *Ciudad Vestigio VoiceBot* was, at first sight, the most challenging. Even in a system with so many different technologies working together - web, mobile and IoT devices - the mission of creating a technology that works with audio and text in an "interpretive manner" within a flow of conversation seemed a complex case. In practice, this process proved to be simpler than the expected with the ease of configuration and deployment provided by the available cloud computing services on the market. In the case of the *Ciudad Vestigio VoiceBot*, the amount of documentation available on the chosen platform, the *IBM Cloud*, has greatly accelerated this process of development.

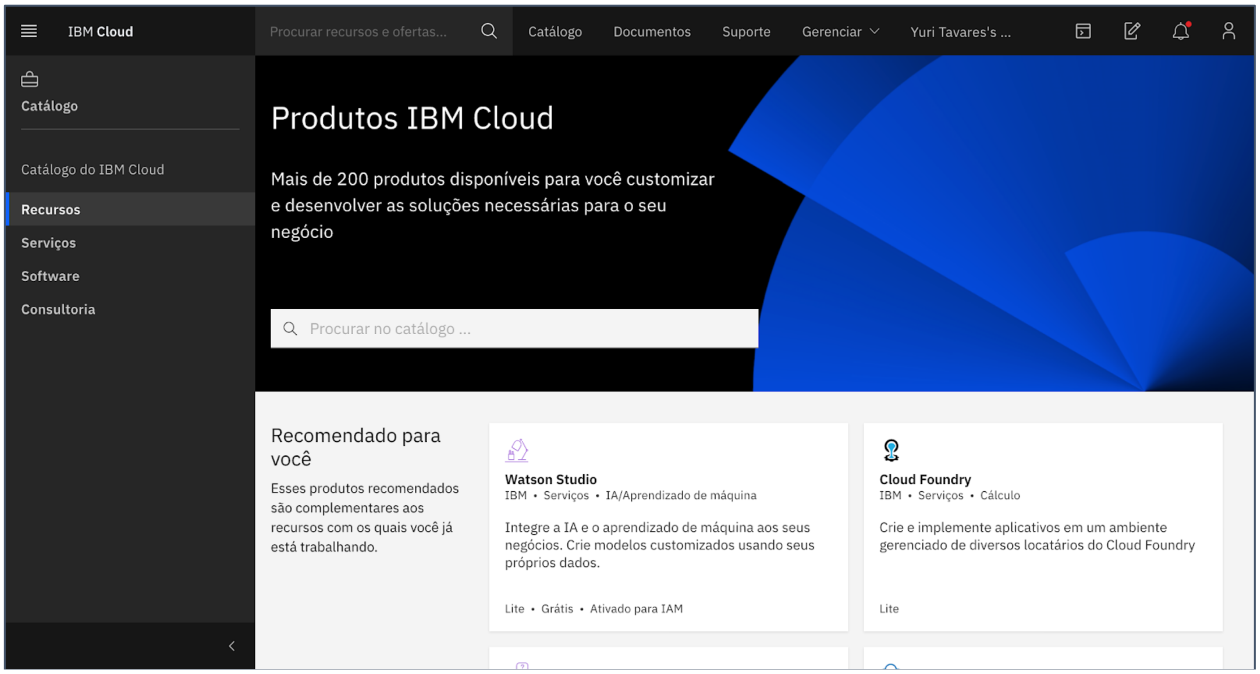
Since there is no code available for this technology - unlike the *Ciudad Vestigio Web Application* or the *Ciudad Vestigio Mobile Application* - this chapter has a profile more similar to chapter three: a short but informative walkthrough on how to correctly configure the *Ciudad Vestigio VoiceBot* to work in the *Ciudad Vestigio* platform. After all, both chapters can serve as a basis for the user who wants to add devices and share content within the project. The development of the *Ciudad Vestigio VoiceBot* in particular has the interesting detail of allowing institutions to think about the content strategy for each registered device, since it is possible to preview questions, answers and dialogs that users can start with the corresponding element in the city.

The set of steps necessary to create the *Ciudad Vestigio VoiceBot* can be summarized in three: create an account and an instance of the *Watson Assistant* service in the *IBM Cloud*; develop a skill with a set of possible questions made by users and their respective answers; create an assistant, add the skill created to it and get the credentials to finish the registration in the *Ciudad Vestigio Web Application*.

The first step in this process is to create an account on the official *IBM Cloud*[61] page. After logging in with the account, the user needs to click on the "Create Resource" button from the "Dashboard" screen. It takes the navigation to a next page that contains

[61] https://www.ibm.com/cloud

the catalog of services offered by *IBM Cloud* and where it is possible to find the service chosen for this project - *Watson Assistant* - with the search tool.



Produtos IBM Cloud

Mais de 200 produtos disponíveis para você customizar e desenvolver as soluções necessárias para o seu negócio

Recomendado para você

Esses produtos recomendados são complementares aos recursos com os quais você já está trabalhando.

Watson Studio
IBM • Serviços • IA/Aprendizado de máquina

Integre a IA e o aprendizado de máquina aos seus negócios. Crie modelos customizados usando seus próprios dados.

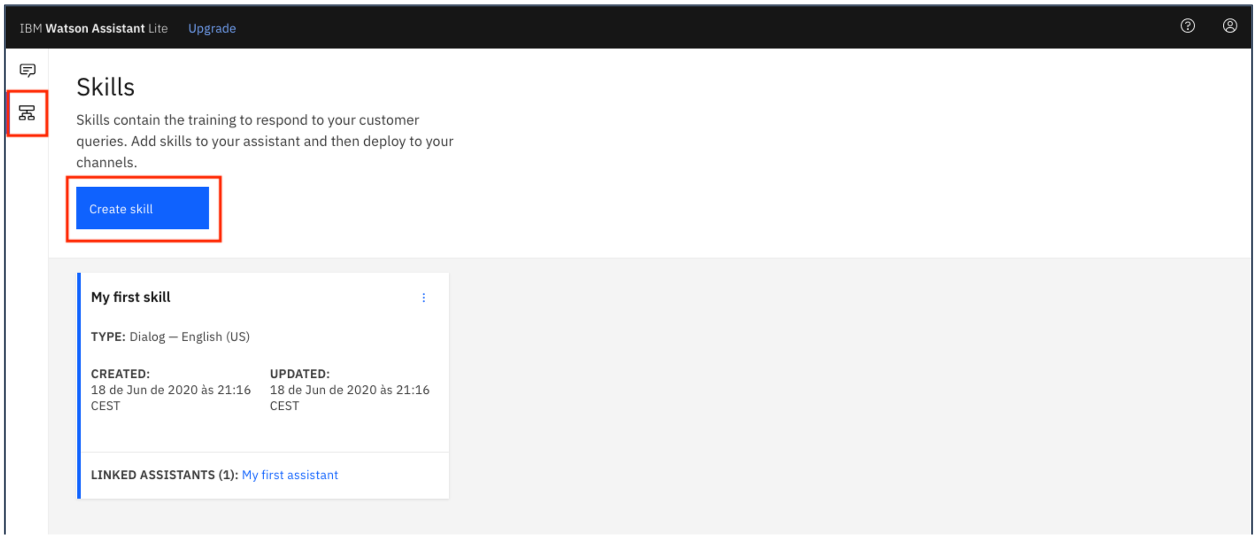
Cloud Foundry
IBM • Serviços • Cálculo

Crie e implemente aplicativos em um ambiente gerenciado de diversos locatários do Cloud Foundry

Figure 6-1. Screenshot that shows the catalog of products offered by IBM Cloud. From the search bar it is possible type the service which is going to be used: Watson Assistant.

The user needs to choose a name for the service, a region to host it and a plan of payment. As discussed in chapter 2, the "Lite" plan is the chosen one. It is free, it provides sufficient service to the project in this first version and it does not require a credit card registration to create the service. If all steps were performed correctly, the *Watson Assistant* service instance will be created. At this point, however, there is no assistant created inside the service and no ability to handle any form of user interaction. It is necessary to add meaningful content to be used by the future assistant.

To start adding content, it is necessary to launch *Watson Assistant* from its main page and, in the "Skills" tab, press the "Create Skill" button.



Skills

Skills contain the training to respond to your customer queries. Add skills to your assistant and then deploy to your channels.

My first skill

TYPE: Dialog — English (US)

CREATED: 18 de Jun de 2020 às 21:16 CEST

UPDATED: 18 de Jun de 2020 às 21:16 CEST

LINKED ASSISTANTS (1): My first assistant

Figure 6-2. Screenshot that shows two items highlighted in red: the "Skills" tab and the "Create Skill" button. There is a skill already created by default, but it will not be used.

After choosing the "Dialog Skill" option, it is necessary to give it a name and select the language with which the skill will process the queries. This means that, in a context where multiple languages are supported, for each language that the institution or the user wants to make a content available, a different corresponding skill must be created. For this first release, the *Ciudad Vestigio Mobile Application* does not support multiple languages - only English - but this is one of the features planned for new versions, as discussed in the next chapter of future work.

The next step is to teach *Watson Assistant* how to handle interactions and questions from users of the *Ciudad Vestigio* platform. This is addressed by the *IBM Cloud* as the "intents" of the user. An intent is a "collection of user statements that have the same meaning. By creating intents, you train your assistant to understand the variety of ways users express a goal" [176]. In other words, it is necessary to create intents about different subjects and, for each intent, add examples of interactions related to them. The more examples that are added, the tendency is that the conversation will be processed better.

Inside the created skill, in the "Intents" tab, the next step is to click on the "Create Intent" button[62]. The example developed for this work will be a *VoiceBot* associated with a device for the *Palacio de Cristal* in *Madrid*. It is an important tourist place for the city built in 1887 [177] and located within a famous park called *Parque del Retiro*.

For this skill example, three intents will be created: *#history*, *#creator* and *#activity*. The nomenclature including hashtag (#) is part of the template used by *IBM Cloud* [176]. The first intent is formed by interactions of users interested in the history of the *Palacio de Cristal*; the second is formed by interactions of users interested in the creator of the place, *Ricardo Velázquez Bosco*; and the third one gathers interactions of users interested in the activities that take place in the *Palacio de Cristal*, which today is one of the exhibiting sites of the *Museo Nacional Centro de Arte Reina Sofía*[63]. Many other types of interactions can be created about an important space like this, but the goal here is to create a simple example situation.

For the first intent, *#history*, a set of eight examples of possible interactions made by users were created and added. Since the idea is to provide the experience of talking to the city and the elements which are part of it, many of the user examples are produced with questions as in a conversation with a person.

USER EXAMPLES
What is your story?
How were you born?
When were you created?
When you were born?
Can you tell me your story?
I want to know your story.

[62] It is possible to import intents from a CSV file. The mechanism can be very useful for when there is a high number of data to add.

[63] https://www.museoreinasofia.es

What were the events behind your creation?
What is the story behind your birth?

Table 6-1. Examples of user interactions for the #history intent.

For the second intent, called *#creator*, a set of eight examples of interactions were created and added.

USER EXAMPLES
Who created you?
Who designed you?
Who is your father or mother?
Who is the architect behind the design of the Palacio de Cristal?
Who is the creator of the Palacio de Cristal?
Who is the author of the Palacio de Cristal?
Who created the Palacio de Cristal?
Who designed the Palacio de Cristal?

Table 6-2. Examples of user interactions for the #creator intent.

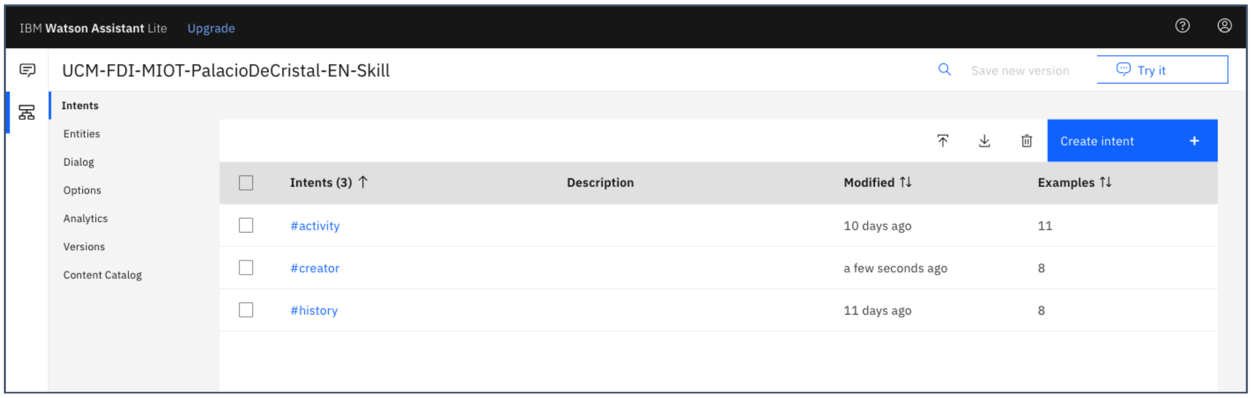
Finally, for the third intent, called *#activity*, a set of eleven examples of user interactions were created and added.

USER EXAMPLES
What activities are carried out here?
What kind of activities take place here?
Do you currently receive any kind of activity?
What is your role today?
Are you part of a project?

What are you used for?
What happens here?
What is your purpose?
What do people do here?
What kind of things are done here?
This space is very large.

Table 6-3. Examples of user interactions for the #activity intent.

The image below shows the three basic intents chosen for this work after the described configuration.



Intents (3) ↑	Description	Modified ↑↓	Examples ↑↓
#activity		10 days ago	11
#creator		a few seconds ago	8
#history		11 days ago	8

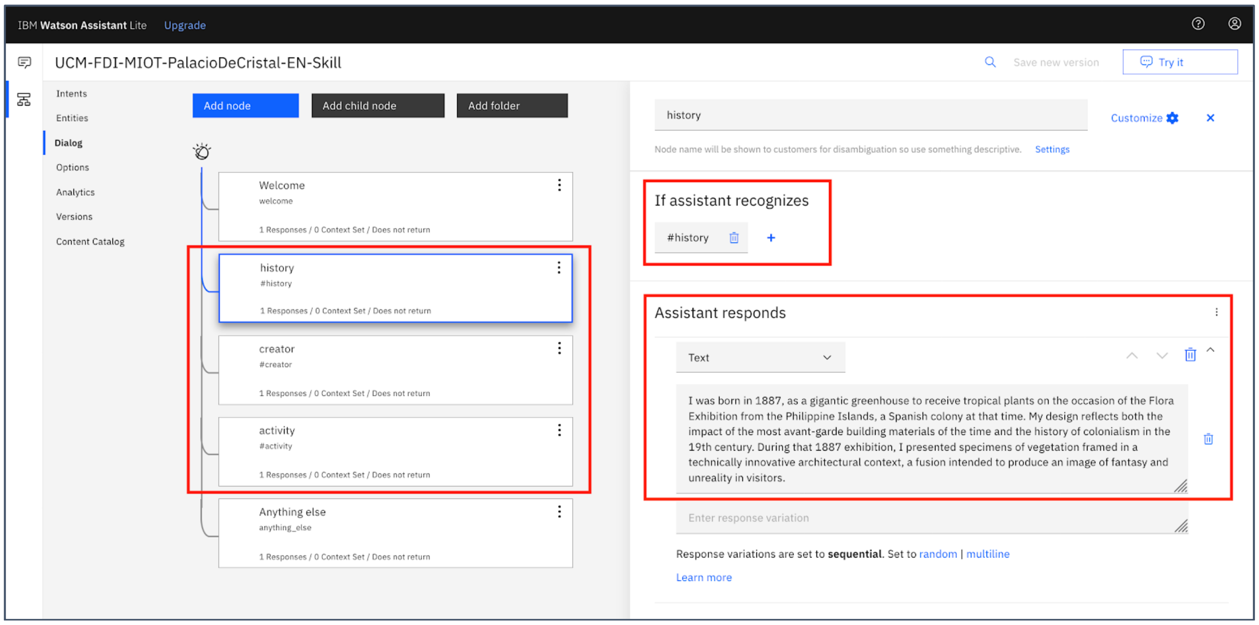
Figure 6-3. Screenshot that shows the result after setting the intents.

The next step is to configure the dialogs and responses which will reproduce a real flow of conversation. Inside the main page of the created skill, in the "Dialog" tab, there are two nodes created by default: the "Welcome" and the "Anything else" nodes. The first one contains a "greeting that is displayed to your users when they first engage with the assistant" [178]. In the context of the *Ciudad Vestigio Mobile Application*, this is the message automatically sent by the *Ciudad Vestigio VoiceBot* as soon as the connection is established between them. This occurs when the user confirms the interest in talking to the element by selecting it from the graphical interface of the application, as seen in the last chapter. By default, the assistant is programmed to send the message "Hello. How can I help you?", but this can be customized for a message that best suits the context of the element in the city, in this case the *Palacio de Cristal*. The second node, the "Anything else", is a very useful resource and contains "phrases that are used to reply to users when their input is not recognized". There are three default messages for this node and it also can be customized to fit better the context of the element which is being configured. In both cases it is important to lead the users to the context of the things in the city with which they are communicating.

A corresponding dialog node will be created for each of the three intents configured and they will contain the possible responses of the assistant to the set of user examples developed in the previous step. The "history" node will be used to illustrate the example and it corresponds to the *#history* intent. The first step is to establish the condition that determines the assistant's entry at that specific node during the flow of conversation. In this case, the intent *#history* is chosen, that is: if a user of the application makes any interaction identified with confidence as related to the "history theme", the dialogue flow goes to that node and verify what to do. It is important to mention that this condition can be concatenated with others using "and" and "or" operators, but it will not be the case in the example. Once the condition is set, it is necessary to define the answer (or the answers) that will be given by the *Ciudad Vestigio VoiceBot* for interactions about the history behind the *Palacio de Cristal*. The following short text has been added as a response, created from official material about the space [177].

> *I was born in 1887, as a giant greenhouse to receive tropical plants on the occasion of the Flora Exhibition from the Philippine Islands, a Spanish colony at that time. My design reflects both the impact of the most avant-garde building materials of the time and the history of colonialism in the 19th century. During that 1887 exhibition, I presented specimens of vegetation framed in a technically innovative architectural context, a fusion intended to produce an image of fantasy and unreality in visitors.*

The image below shows some of the elements discussed over the last paragraphs. The step performed with the "history" node needs to be performed for each of the other nodes. The "Try it" button in the upper right corner of the screen is an important feature because it allows the developer to test conversations as the nodes are created, edited and the model is retrained.


IBM Watson Assistant Lite
Upgrade
UCM-FDI-MIOT-PalacioDeCristal-EN-Skill
Save new version
Try it
Intents
Entities
Dialog
Options
Analytics
Versions
Content Catalog
Add node
Add child node
Add folder
Welcome
welcome
1 Responses / 0 Context Set / Does not return
history
#history
1 Responses / 0 Context Set / Does not return
creator
#creator
1 Responses / 0 Context Set / Does not return
activity
#activity
1 Responses / 0 Context Set / Does not return
Anything else
anything_else
1 Responses / 0 Context Set / Does not return
history
Customize
Node name will be shown to customers for disambiguation so use something descriptive.
Settings
If assistant recognizes
#history
Assistant responds
Text
Enter response variation
Learn more

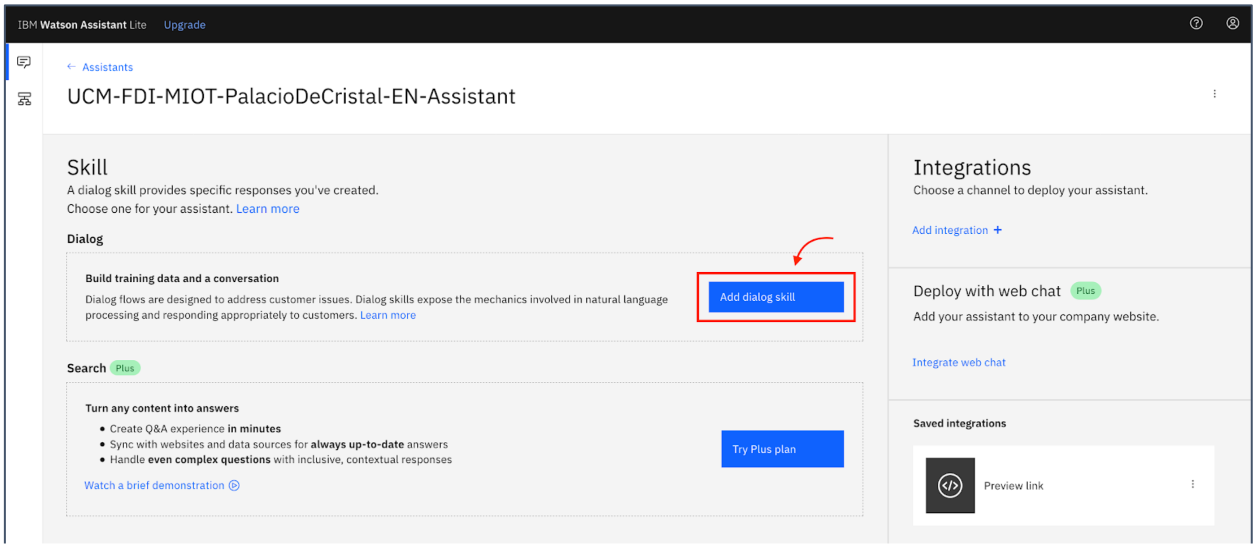
I was born in 1887, as a gigantic greenhouse to receive tropical plants on the occasion of the Flora Exhibition from the Philippine Islands, a Spanish colony at that time. My design reflects both the impact of the most avant-garde building materials of the time and the history of colonialism in the 19th century. During that 1887 exhibition, I presented specimens of vegetation framed in a technically innovative architectural context, a fusion intended to produce an image of fantasy and unreality in visitors.

Response variations are set to **sequential**. Set to random | multiline

Figure 6-4. Screenshot that shows on the left the three created nodes – history, creator and activity – each corresponding to an intent. On the right it is possible to see how the history node was configured.

It is very important to note that this developed example shows the simplest possible approach for configuring the *Watson Assistant* as the *Ciudad Vestigio VoiceBot*, but that it is still effective in the context of the *Ciudad Vestigio* platform. The tool has much more complex features to build conversational flows with action triggers that the developer can explore from the available documentation and tutorials [179].

Now that the skill and its dialog flow are ready, it is time to bind it to an assistant and finally get the credentials. Within the "Assistants" tab, the next step is to click on the "Create Assistant" button and choose a name for it. The next screen shows the "Add dialog skill" button with which it is possible to connect the configured skill to the newly created assistant.


IBM **Watson Assistant** Lite Upgrade

← Assistants

UCM-FDI-MIOT-PalacioDeCristal-EN-Assistant

Skill

A dialog skill provides specific responses you've created.
Choose one for your assistant. Learn more

Dialog

Build training data and a conversation

Dialog flows are designed to address customer issues. Dialog skills expose the mechanics involved in natural language processing and responding appropriately to customers. Learn more

Add dialog skill

Search Plus

Turn any content into answers

- Create Q&A experience **in minutes**
- Sync with websites and data sources for **always up-to-date** answers
- Handle **even complex questions** with inclusive, contextual responses

Watch a brief demonstration

Try Plus plan

Integrations

Choose a channel to deploy your assistant.

Add integration +

Deploy with web chat Plus

Add your assistant to your company website.

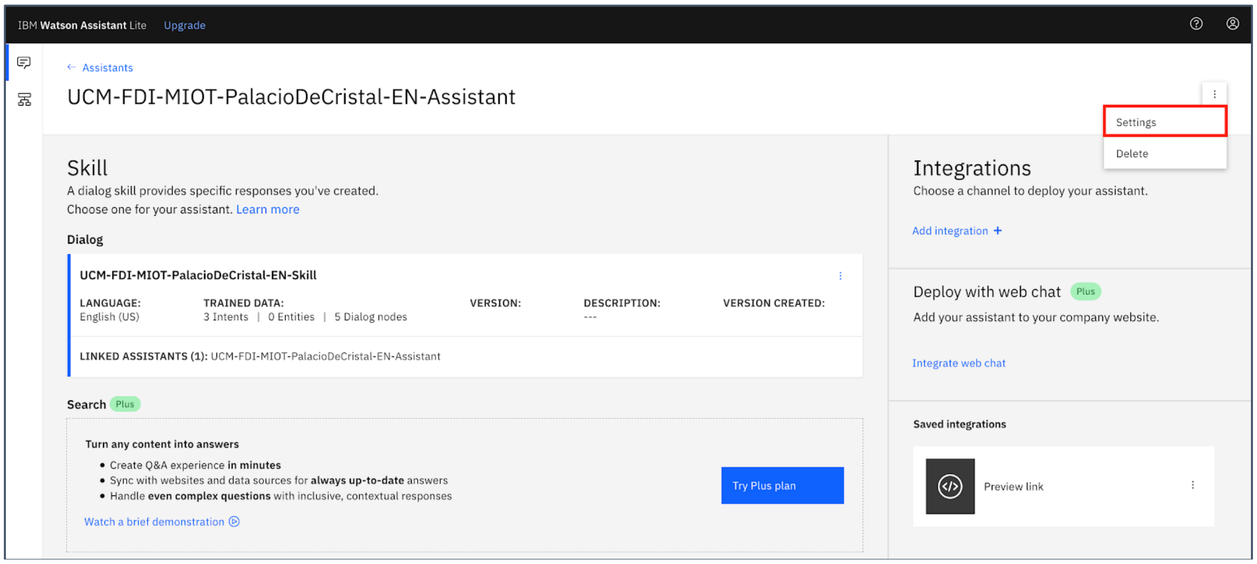
Integrate web chat

Saved integrations

Preview link

Figure 6-5. Screenshot that shows how to connect the created assistant to the skill set in the last steps.

After selecting the skill, it is necessary to click on the three dots located in the upper right corner of the page, then click on "Settings". In the left bar, the "API details" menu presents all the necessary credentials to configure the *Ciudad Vestigio VoiceBot* within the *Ciudad Vestigio Web Application*.


IBM **Watson Assistant** Lite Upgrade

← Assistants

UCM-FDI-MIOT-PalacioDeCristal-EN-Assistant

Settings

Delete

Skill

A dialog skill provides specific responses you've created.
Choose one for your assistant. Learn more

Dialog

UCM-FDI-MIOT-PalacioDeCristal-EN-Skill

LANGUAGE:	TRAINED DATA:	VERSION:	DESCRIPTION:	VERSION CREATED:
English (US)	3 Intents | 0 Entities | 5 Dialog nodes		---	

LINKED ASSISTANTS (1): UCM-FDI-MIOT-PalacioDeCristal-EN-Assistant

Search Plus

Turn any content into answers

- Create Q&A experience **in minutes**
- Sync with websites and data sources for **always up-to-date** answers
- Handle **even complex questions** with inclusive, contextual responses

Watch a brief demonstration

Try Plus plan

Integrations

Choose a channel to deploy your assistant.

Add integration +

Deploy with web chat Plus

Add your assistant to your company website.

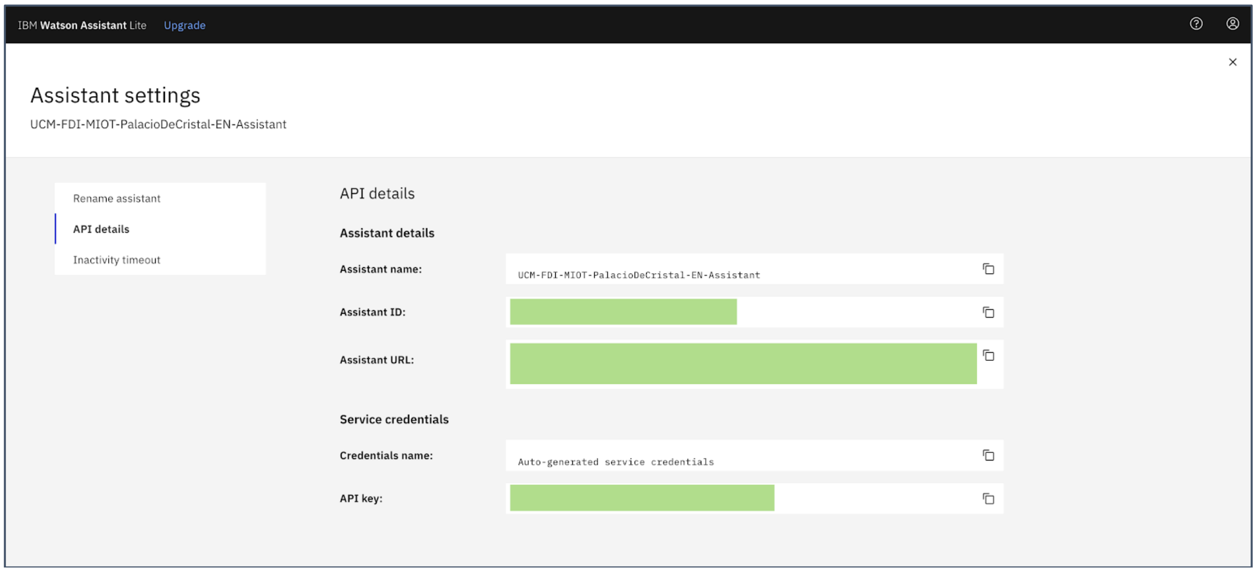
Integrate web chat

Saved integrations

Preview link

Figure 6-6. Screenshot that shows the first step to get the credentials of the assistant.

As previously discussed in chapter 4, three data are required: the *Watson Assistant API Key*, the *Watson Assistant ID* and the *Watson Assistant URL*. These three pieces of data are covered in green in the image below.



Assistant settings

UCM-FDI-MIOT-PalacioDeCristal-EN-Assistant

API details

Assistant details

Assistant name: UCM-FDI-MIOT-PalacioDeCristal-EN-Assistant

Assistant ID: [redacted]

Assistant URL: [redacted]

Service credentials

Credentials name: Auto-generated service credentials

API key: [redacted]

Figure 6-7. Screenshot that shows the three data necessary to register the Ciudad Vestigio VoiceBot together with its corresponding device in the Ciudad Vestigio Web Application.

After copying each one, this data can be added in the process of registering a device with the *Ciudad Vestigio Web Application*, as seen in chapter four. If all the steps[64] are performed correctly, the device should be properly integrated into the platform with the expected interaction features. On the part of the *Ciudad Vestigio Mobile Application*, the integration between the two technologies involved - the *React Native Framework* and the *IBM Cloud API* - can be verified in the file "voicebot.js" of the available code.

[64] Creating the *Ciudad Vestigio VoiceBot*; registering the *Ciudad Vestigio Device* with the respective assistant in the *Ciudad Vestigio Web Application;* and configuring the device with credentials returned by the web application.

Chapter 7 – Future Work

One of the most interesting aspects of the project in its current state is the diversity of directions it can take from now on. Some of them are listed below, with decisions and impacts that do not exclude each other.

The first and more important idea of future work is related to expanding the capabilities of the platform as a tool of accessibility and empowerment for people with visual impairment. New features of usability would be necessary (and a lot of study to achieve it), but since the project uses the voice as its main mechanism of interaction, no "strange" or "forced" adaptations would be necessary to include more users and promote a better experience in the city for more people. A whole new layer of features could be added to what the platform does at this moment and the more devices in a city, the more a person can get information, gain autonomy and experiment with more confidence the public space.

A second idea of improvement is related to the idea of "exploring the city" addressed over this work. In addition to the current features of the *Ciudad Vestigio Mobile Application*, the idea is to build a map with which the user can visualize the position of other devices in the city and go search for them in person. The implementation of a map could bring many different possibilities to the project. The app could, for example, indicate routes of interactive elements based on the user's location and/or interest. The user, in turn, could maintain a list of visited devices and another list for elements he/she wants to visit. A gamification strategy could be implemented, creating missions to be done, achievements, rankings, etc. Related to this theme of "exploring the city", another simple feature that needs to be implemented are notifications. If the user is not using the app in the foreground and passes through an interactive element, a notification could warn him/her under certain conditions (the list of elements to be visited, for example). The *react-native-beacons-manager* library used to handle the *BLE* demands of the *Ciudad Vestigio Mobile Application* have functions to achieve this objective.

A third idea is related to the collection and analysis of data transferred through the *Ciudad Vestigio Mobile Application*. Once the project connects so many different subjects - such as culture, history, urbanism, tourism - the universe of information that can be obtained is very interesting and could be used for the benefit of the community. It is very important to mention that the objective here is not to collect personal data from users in order to generate profiles and usage patterns. The application does not even ask for a login or other kind of personal information from the person who is using it. With this data, a public administration could understand, for example, where are the main spots of interest in a city and create strategies to highlight other regions; or understand, for example, which foreign languages are the most used with the *Ciudad Vestigio Mobile Application*. And once the topic "foreign languages" has been addressed, another idea of new feature is to add support for more than one language with the *Ciudad Vestigio Mobile Application and VoiceBot*. It would be necessary to create content for more than one language, but, since the project is collaborative and this detail cannot be controlled, the use of libraries and external services to translate textual content could be a solution.

A fifth idea of new features would be to develop the platform's own cognitive assistant. This is a more "bold" and long-term proposal due to the amount of work and time it would require. The first advantage would be to concentrate all the steps of configuring a device with its corresponding cognitive assistant in the same place. The second, and more important, advantage would be the possibility to generate revenue for the project through the service provided with this own assistant. This would be a service for institutions and users that need a high number of queries.

A sixth line of future work is related to the development of documentation about the project for the potential community of users. It is understood that the process of configuring a device is not so simple for people who do not have a more solid experience with computers and programming. It is very important to create content in different languages and formats with all the necessary steps. This documentation could also give more options of devices to use in the platform.

Another idea of future work is the possibility to capture feedback from users in order to improve the quality of the content. This could be used to evaluate responses given by the platform and to understand, from the unanswered questions, where the possible content voids are.

Lastly, currently the project suggests a specific device (*HM-19* in combination with a *MB102* module) to be used as the *Ciudad Vestigio Device*. A last idea would be the creation of an open source "official case" for the device, in order to give a common appearance to devices that are part of the platform. This case could be shared as a file with the “.STL” file extension to be fabricated with a 3D printer.

Chapter 8 – Conclusions

We live in an interesting and, at the same time, frightening period of the world: on the one hand, the dystopia of surveillance and the precariousness from the "platform capitalism" [180] are stronger than ever; on the other, we live the possibility - often just a possibility - of human emancipation in fields not always imagined. Two of those universes that carry the "complexity of the times" - between good things and not so good things - are the concepts of *Internet of Things* and *Smart Cities*. For those who follow the debates about tendencies in technology, these concepts have been on the "tech trends" lists for years [181] [182] [183] [184] [185]. But why don't we see them being implemented with the same urgency that they are announced as "the next big thing"? Where are they in the "materiality" of our daily lives?

The intention with this chapter of conclusions is not to try to answer these questions, but to promote some reflections. Concepts such as *Internet of Things* and *Smart Cities* are so complex and without appeasement for so many years that we can only define them as two ideas in dispute. Ask an engineer, an urban planner, a politician, an activist and a citizen what a Smart City is and you will fail to try to accommodate a unified definition. If in the field of abstraction we are already dealing with so many perspectives and stakeholders in dispute of narrative, it is justifiable that their existence in the field of materiality is even more complex. Years have passed and the episode of the *Smart City* project in *Toronto* - headed by *Sidewalk Labs*[65] - comes to support this point of view [186].

The project presented in this work – which was called *Ciudad Vestigio* throughout the pages and chapters – is born in the light and shadow of all these reflections and complexities. The project dialogues with the *Internet of Things* concept by making use of its technical repertoire of protocols, devices and possible combinations to achieve its main objective. In the same context, the project also dialogues with the concept of *Smart Cities* by proposing a "technological layer" to imagine a possibility of city. However, instead of the narrative often focused on "functionality" and "efficiency" about how cities can be transformed - automated houses, improvements in car traffic, efficiency in

[65] Sidewalk Labs is a Google's sister firm.

the use of resources, energy efficiency and "improvements in security" - this work tries to stick its roots into a slightly different sphere: in the dimension of human subjectivities and sensitivities.

The project's initial objective was to create a platform for cities with which citizens can "communicate with it" interactively, using voice commands and without a predetermined script. Throughout this document, different perspectives of the project were presented: possible scenarios for each technology, their development challenges, how these technologies combine with each other and different step-by-step guides on how to add a device with the cognitive assistant to the platform.

When reflecting on the technical decisions made during the development of the project, the two most difficult ones were choosing the mobile development framework for creating the *Ciudad Vestigio Mobile Application*; and choosing the two *React Native* libraries to perform the *speech-to-text and text-to-speech* functions of the *Ciudad Vestigio VoiceBot*.

The challenge with the first one – choosing the mobile development framework to be used - was related to the fact that there was no prior knowledge about any of the options discussed in chapter 2. Because it is a technology that converges the work of the other three, it was necessary to guarantee that the chosen framework, *React Native*, would be able to respond with the necessary quality and speed to the development challenges. As a consequence, the design process of the solution required a more extensive prior study before starting to code and prior research about available libraries was an essential step to guarantee the results.

Regarding the second of the decisions – using the *React Native* libraries to execute functions that were originally designed for use with cloud computing services – it is understood that it was one of the most interesting decisions of the project. In addition to the obvious consequences of performance and reducing the possibility of errors[66], the

[66] Having the speech-to-text and text-to-speech functions inside the Ciudad Vestigio Mobile Application reduces the number of steps that a user needs to perform to add a device to the platform and, consequently, the possibility of errors that can happen along the way.

decision generates another very important benefit: it allows to assign a little more control over the content to the *Ciudad Vestigio* platform.

This can be a controversial statement, since the project was born under the sign of collaboration by inviting users and institutions to share content about their cities. However, one of the details that most concerns about this model is the misuse of this relationship of trust. After all, it is the user who sets up the cognitive assistant with possible questions and answers for interactions, without any interference by the platform. By adding these two essential steps – the *speech-to-text* and *text-to-speech* functions - under the responsibility of the *Ciudad Vestigio Mobile Application*, it is possible to do a minimal processing of the responses that come from the *Watson Assistant* to check for any types hate speech, expressions that hurt the platform's politics, etc. This is not implemented in this first version, but it is important to note that a decision that was initially guided by technical performance issues also became a possible solution to a topic that was pending on the project's radar.

Still on this theme - sharing content in a collaborative way - the investigation of related projects - especially the first one presented, called *A Voz da Arte* - brings very interesting observations. In the text wrote by the *IBM Technical Leader* involved with the initiative, it is argued that the best content strategy is incremental and iterative. "That is, to release as soon as possible a version of the system created with few examples. Collect all user interaction with the application and add the new questions asked in the set of examples. Periodically, you must retrain the model and update the application with the new version of it" [187]. Although not implemented in this first version, this is a strategy that seems to make sense in the context of *Ciudad Vestigio* and which can be tested to verify eventual positive results. Once again, having the *speech-to-text* and *text-to-speech* functions under the responsibility of the *Ciudad Vestigio Mobile Application* brings an advantage to test this scenario. The platform could help collaborating users and institutions by presenting insights to improve engagement with their interactions and cover potential content gaps.

When reflecting on the feasibility of implementing the project in real scenarios, it is believed that it is possible to mobilize different actors - institutions and citizens – around a project that can grow collectively. In this context, community activation activities, such as talks, device configuration workshops and partnerships with different actors in a city can play a fundamental role. However, it is important to highlight a perspective presented during the introduction of this work: the potential that the project carries if adopted as a public policy of culture and/or tourism by the public administration of a city. It is an extremely cheap project for a municipal budget[67] that can generate very interesting returns for the community. The project can positively impact citizens by strengthening the collective knowledge about cities and their histories, cultures, heritage and other forms of narratives. As already mentioned, the platform can be used by schools to provide their students with different learning experiences about the city. If also implemented as a tourism policy, the project can indirectly impact the local economy by improving the experience of tourists in the city - who will have an open-air audio guide, without any itineraries. With the anonymous data collected, the local administration can develop strategies for public policies from a data-driven approach. Finally, the project also has the potential to positively impact visually impaired people in their experience around the city. This is a whole world of features and possibilities that can be developed, but also a challenge that demands a lot of research.

Finally, it is important to end these conclusions by stating that one of the main motivations behind this project is the idea of "humanizing the city", even if slightly, by creating a platform with which people can learn, generate positive debate and strengthen their personal sense of belonging to a place that many times can be oppressive and violent in many different ways. *Ciudad Vestigio* proposes an invitation to celebrate the public spaces and the "vestiges" of history that our cities keep in every corner.

[67] The solution presented in this work uses a combination of devices (HM-19, MB102) that cost around 12 euros in common vendors from Spain. This value can be decreased considerably as devices are purchased in quantity and from other countries. The other main cost depends on the volume of data used from the cloud computing services.

BIBLIOGRAPHY

[1] kontakt.io, «What is a beacon?,» kontakt.io, [En línea]. Available: https://kontakt.io/beacon-basics/what-is-a-beacon/. [Último acceso: 28 04 2020].

[2] I. Paula, «Fortaleza é a nona cidade mais desigual no acesso ao emprego, aponta estudo do Ipea,» Jornal o Povo, 17 01 2020. [En línea]. Available: https://www.opovo.com.br/noticias/economia/2020/01/17/fortaleza-e-a-nona-cidade-mais-desigual-no-acesso-ao-emprego--aponta-estudo-do-ipea.html. [Último acceso: 28 08 2020].

[3] G1, «Fortaleza é uma das cidades mais desiguais da América Latina, diz ONU,» 22 08 2012. [En línea]. Available: http://g1.globo.com/ceara/noticia/2012/08/fortaleza-e-uma-das-cidades-mais-desiguais-da-america-latina-diz-onu.html. [Último acceso: 28 08 2020].

[4] C. V. Mota, «Como Fortaleza se tornou a região metropolitana mais violenta do Brasil,» G1, 07 03 2019. [En línea]. Available: https://g1.globo.com/ce/ceara/noticia/2019/03/07/como-fortaleza-se-tornou-a-regiao-metropolitana-mais-violenta-do-brasil.ghtml. [Último acceso: 29 08 2020].

[5] Jornal O Povo, «Fortaleza é a 9ª cidade mais violenta do mundo,» Jornal O Povo, 09 04 2019. [En línea]. Available: https://www.opovo.com.br/jornal/farol/2019/04/08/fortaleza-e-a-9---cidade-mais-violenta-do-mundo.html. [Último acceso: 28 08 2020].

[6] W. Rosner y U. Vilsmaier, «Planificación urbana versus autoregulación: la ciudad de Fortaleza (Brasil) como tablero de juego de diferentes agentes,» *Scripta Nova,* vol. 5, pp. 79-104, 2001.

[7] E. d. Lima, «Quem lucra com uma cidade sem memória?,» Jornal O Povo, 02 07 2015. [En línea]. Available: http://blogdoeliomar.com.br/2015/07/02/quem-lucra-com-uma-cidade-sem-memoria/. [Último acceso: 03 09 2020].

[8] Jornal O Povo, «Mansão na Aldeota com jardins de Burle Marx é tombada provisoriamente após demolição inciada,» Jornal O Povo, 02 06 2020. [En línea]. Available: https://www.opovo.com.br/noticias/ceara/2020/06/02/mansao-na-aldeota-com-jardins-de-burle-marx-e-tombada-provisoriamente-apos-demolicao-inciada.html. [Último acceso: 03 09 2020].

[9] M. Facundo, «Conselho Gestor aprova início de estudo para empreendimento em 50 hectares de área preservada nas dunas da Sabiaguaba,» Jornal O Povo, 09 07 2020. [En línea]. Available: https://www.opovo.com.br/noticias/fortaleza/2020/07/09/dunas-da-sabiaguaba-conselho-gestor-aprova-construcao-em-area-preservada.html. [Último acceso: 03 09 2020].

[10] V. K. -. V. m. e. https://copadomundo.uol.com.br/noticias/redacao/2012/05/17/vlt-de-fortaleza-tem-estudo-de-impacto-ambiental-reprovado-por-tce-e-pode-parar.htm?cmpid=copiaecola, «VLT de Fortaleza tem falhas em estudo de impacto; conselheira do TCE quer parar obra... - Veja mais em https://copadomundo.uol.com.br/noticias/redacao/2012/05/17/vlt-de-fortaleza-tem-estudo-de-impacto-ambiental-reprovado-por-tce-e-pode-parar.htm?cmpid=cop,» UOL, 17 05 2012. [En línea]. Available: https://copadomundo.uol.com.br/noticias/redacao/2012/05/17/vlt-de-fortaleza-tem-estudo-de-impacto-ambiental-reprovado-por-tce-e-pode-parar.htm. [Último acceso: 03 09 2020].

[11] Prefeitura de SP, «Fab Lab Livre SP,» Prefeitura de SP, 03 09 2018. [En línea]. Available: https://www.prefeitura.sp.gov.br/cidade/secretarias/inovacao/inclusao_digital/index.php?p=194463. [Último acceso: 20 07 2020].

[12] R. Oliveira, «Fábricas de criatividade revolucionam aprendizado (e já há delas no Brasil),» El País Brasil, 17 11 2016. [En línea]. Available: https://brasil.elpais.com/brasil/2016/11/16/politica/1479318738_264761.html. [Último acceso: 20 07 2020].

[13] Center for Teaching Innovation, «Collaborative Learning,» Center for Teaching Innovation, [En línea]. Available: https://teaching.cornell.edu/teaching-resources/engaging-students/collaborative-learning. [Último acceso: 28 08 2020].

[14] K. Finley, «The WIRED Guide to Open Source Software,» Wired, 24 04 2019. [En línea]. Available: https://www.wired.com/story/wired-guide-open-source-software/. [Último acceso: 28 08 2020].

[15] M. Hatch, The Maker Movement Manifesto: Rules for Innovation in the New World of Crafters, Hackers, and Tinkerers, McGraw-Hill, 2013.

[16] Ogilvy, «A Voz da Arte,» Ogilvy, [En línea]. Available: http://www.ogilvy.com.br/cases/a-voz-da-arte. [Último acceso: 24 08 2020].

[17] Metrópolis, «Metrópolis: Pinacoteca Watson,» 19 04 2017. [En línea]. Available: https://www.youtube.com/watch?v=2jjs7guPX2g. [Último acceso: 24 08 2020].

[18] L. Agrela, «Inteligência artificial da IBM ajuda você a entender Portinari,» Exame, 06 04 2017. [En línea]. Available: https://exame.com/tecnologia/inteligencia-artificial-da-ibm-ajuda-voce-a-entender-portinari/. [Último acceso: 24 08 2020].

[19] M. mg, «A Voz da Arte - IBM Watson,» 03 04 2017. [En línea]. Available: https://www.youtube.com/watch?v=1rOAgvCnZpw. [Último acceso: 24 08 2020].

[20] F. Barth, «Alguns detalhes técnicos e desafios do projeto "A voz da arte – IBM | Watson",» LinkedIn, 05 04 2017. [En línea]. Available: https://www.linkedin.com/pulse/alguns-detalhes-t%C3%A9cnicos-e-desafios-do-projeto-voz-da-fabr%C3%ADcio-barth/. [Último acceso: 24 08 2020].

[21] Hello Lamp Post, «Hello Lamp Post,» Hello Lamp Post, [En línea]. Available: https://www.hellolamppost.co.uk/. [Último acceso: 24 08 2020].

[22] Twilio Docs, «SMS: Character Limits and Images in Text Messages,» Twilio Docs, [En línea]. Available: https://www.twilio.com/docs/glossary/what-is-an-sms-short-message-service. [Último acceso: 24 08 2020].

[23] Pan Studio, «Hello Lamp Post,» Pan Studio, [En línea]. Available: http://panstudio.co.uk/project/hello-lamp-post/. [Último acceso: 24 08 2020].

[24] Twilio, «Build the new normal for contactless delivery with Twilio SMS,» Twilio, [En línea]. Available: https://www.twilio.com/. [Último acceso: 24 08 2020].

[25] Daily Tous les Jours, «Amateur Intelligence Radio: A radio show hosted by a building,» Daily Tous les Jours, [En línea]. Available: https://www.dailytouslesjours.com/en/work/air. [Último acceso: 24 08 2020].

[26] Intel, «What Is Computer Vision?,» Intel, [En línea]. Available: https://www.intel.com/content/www/us/en/internet-of-things/computer-vision/overview.html. [Último acceso: 25 08 2020].

[27] soofa, «Connect here: The neighborhood news feed that connects a community with screens everyone can see and anyone can use,» soofa, [En línea]. Available: http://www.soofa.co/. [Último acceso: 25 08 2020].

[28] T. Nardi, «Will 2020 (finnaly) be the year of electronic paper?,» Hackday, 13 04 2020. [En línea]. Available: https://hackaday.com/2020/04/13/will-2020-finally-be-the-year-of-electronic-paper/. [Último acceso: 25 08 2020].

[29] soofa, «The Neighborhood News Feed,» soofa, [En línea]. Available: https://talk.soofa.co/. [Último acceso: 25 08 2020].

[30] R. Walrath, «How One Startup Is Getting COVID-19 Information to Boston's Pedestrians,» The Business Journals, 16 03 2020. [En línea]. Available: https://www.bizjournals.com/boston/inno/stories/news/2020/03/16/how-one-startup-is-getting-covid-19-information-to.html. [Último acceso: 25 08 2020].

[31] A. El-Rabbany, Introduction to Gps: The Global Positioning System, Artech house, 2002.

[32] GPS.gov, «Space Segment,» GPS.gov, [En línea]. Available: https://www.gps.gov/systems/gps/space/. [Último acceso: 27 08 2020].

[33] M. Capderou, Handbook of Satellite Orbits: From Kepler to GPS, Springer Science & Business, 2014.

[34] GPS.gov, «GPS Accuracy,» GPS.gov, [En línea]. Available: https://www.gps.gov/systems/gps/performance/accuracy/. [Último acceso: 27 08 2020].

[35] Cisco, «What Is Wi-Fi?,» Cisco, [En línea]. Available: https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html. [Último acceso: 01 09 2020].

[36] Kaspersky, «What is a QR Code and how do I scan one?,» Kaspersky, [En línea]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan. [Último acceso: 27 08 2020].

[37] Whatsapp, «Como escanear um código QR do WhatsApp,» Whatsapp, [En línea]. Available: https://faq.whatsapp.com/android/contacts/how-to-scan-a-whatsapp-qr-code. [Último acceso: 27 08 2020].

[38] Nubank, «O que é um QR Code e como funciona essa tecnologia?,» Nubank, [En línea]. Available: https://blog.nubank.com.br/qr-code/. [Último acceso: 27 08 2020].

[39] Bluetooth, «Archived Specifications,» Bluetooth, [En línea]. Available: https://www.bluetooth.com/specifications/archived-specifications/. [Último acceso: 27 08 2020].

[40] K. Townsend, C. Cufí, Akiba y R. Davidson, Getting Started with Bluetooth Low Energy, O'Reilly Media, Inc., 2014.

[41] R. McCormick, «https://www.theverge.com/2016/6/10/11900038/bluetooth-5-announced-double-range-4-times-speed,» The Verge, 10 06 2016. [En línea]. Available: Bluetooth 5 will be announced next week with four times the range and double the speed. [Último acceso: 28 08 2020].

[42] Bluetooth, «Core Specifications,» Bluetooth, [En línea]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/. [Último acceso: 28 08 2020].

[43] Apple Developer, «Getting Started with iBeacon,» Apple Developer, [En línea]. Available: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf. [Último acceso: 01 09 2020].

[44] Espressif, «ESP32,» Espressif, [En línea]. Available: https://www.espressif.com/en/products/socs/esp32. [Último acceso: 28 08 2020].

[45] Texas Instruments, «SimpleLink Bluetooth 5.1 Low Energy Wireless MCU,» Texas Instruments, [En línea]. Available: https://www.ti.com/lit/ds/symlink/cc2640r2f.pdf?ts=1599221652365&ref_url=https%253A%252F%252F. [Último acceso: 01 09 2020].

[46] S. Toulson, «iBeacon Parameters: UUID, Major and Minor,» kontakt.io, [En línea]. Available: https://support.kontakt.io/hc/en-gb/articles/201620741-iBeacon-Parameters-UUID-Major-and-Minor. [Último acceso: 01 09 2020].

[47] Codecademy, «What is CRUD?,» Codecademy, [En línea]. Available: https://www.codecademy.com/articles/what-is-crud. [Último acceso: 30 08 2020].

[48] MongoDB, «MERN Stack,» MongoDB, [En línea]. Available: https://www.mongodb.com/mern-stack. [Último acceso: 12 08 2020].

[49] MongoDB, «What is a Non-Relational Database?,» MongoDB, [En línea]. Available: https://www.mongodb.com/non-relational-database. [Último acceso: 03 09 2020].

[50] Oracle, «What Is Big Data?,» Oracle, [En línea]. Available: https://www.oracle.com/big-data/what-is-big-data.html. [Último acceso: 31 08 2020].

[51] MongoDB, «Geospatial Queries,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/geospatial-queries/. [Último acceso: 03 09 2020].

[52] MongoDB, «2dsphere Indexes,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/core/2dsphere/. [Último acceso: 03 09 2020].

[53] The Pallets Projects, «Flask,» The Pallets Projects, [En línea]. Available: https://palletsprojects.com/p/flask/. [Último acceso: 03 09 2020].

[54] The Pallets Projects, «Foreword,» The Pallets Projects, [En línea]. Available: https://flask.palletsprojects.com/en/1.1.x/foreword/. [Último acceso: 03 09 2020].

[55] The Pallets Projects, «Design Decisions in Flask,» The Pallets Projects, [En línea]. Available: https://flask.palletsprojects.com/en/1.1.x/design/. [Último acceso: 03 09 2020].

[56] The Pallets Projects, «Werkzeug,» The Pallets Projects, [En línea]. Available: https://werkzeug.palletsprojects.com/en/1.0.x/. [Último acceso: 03 09 2020].

[57] The Pallets Projects, «Jinja,» The Pallets Projects, [En línea]. Available: https://palletsprojects.com/p/jinja/. [Último acceso: 03 09 2020].

[58] The Pallets Projects, «Jinja,» The Pallets Projects, [En línea]. Available: https://github.com/pallets/jinja. [Último acceso: 03 09 2020].

[59] F. Allanabanda y V. Veeraraghavan, «Automation as a Service: Introducing Scriptflask,» Netflix Technology Blog , 22 05 2017. [En línea]. Available: https://netflixtechblog.com/automation-as-a-service-introducing-scriptflask-17a8e4ad954b. [Último acceso: 03 09 2020].

[60] Stackshare, «vault.uber.com,» Stackshare, [En línea]. Available: https://stackshare.io/uber-technologies/vault-uber-com. [Último acceso: 03 09 2020].

[61] Stackshare, «Reddit,» Stackshare, [En línea]. Available: https://stackshare.io/reddit/reddit. [Último acceso: 03 09 2020].

[62] Stackshare, «Trivago,» Stackshare, [En línea]. Available: https://stackshare.io/trivago/trivago. [Último acceso: 03 09 2020].

[63] Stackshare, «MIT,» Stackshare, [En línea]. Available: https://stackshare.io/mit/mit. [Último acceso: 03 09 2020].

[64] Stackshare, «Lime,» Stackshare, [En línea]. Available: https://stackshare.io/lime/lime. [Último acceso: 03 09 2020].

[65] IBM Cloud Education, «NoSQL Databases,» IBM Cloud, 06 08 2019. [En línea]. Available: https://www.ibm.com/cloud/learn/nosql-databases. [Último acceso: 04 09 2020].

[66] B. Anderson y B. Nicholson, «SQL vs. NoSQL Databases: What's the Difference?,» IBM Cloud, 18 06 2020. [En línea]. Available: https://www.ibm.com/cloud/blog/sql-vs-nosql. [Último acceso: 04 09 2020].

[67] Oracle, «The Java Tutorials,» Oracle, [En línea]. Available: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html. [Último acceso: 04 09 2020].

[68] MariaDB, «Understanding the Relational Database Model,» MariaDB, [En línea]. Available: https://mariadb.com/kb/en/understanding-the-relational-database-model/. [Último acceso: 04 09 2020].

[69] MongoDB, «MongoDB GitHub Repository,» MongoDB, [En línea]. Available: https://github.com/mongodb. [Último acceso: 04 09 2020].

[70] MongoDB, «Database Management Systems,» MongoDB, [En línea]. Available: https://www.mongodb.com/database-management-system. [Último acceso: 04 09 2020].

[71] IBM Cloud Education, «MongoDB,» IBM Cloud, 27 08 2019. [En línea]. Available: https://www.ibm.com/cloud/learn/mongodb. [Último acceso: 04 09 2020].

[72] MongoDB, «JSON and BSON,» MongoDB, [En línea]. Available: https://www.mongodb.com/json-and-bson. [Último acceso: 04 09 2020].

[73] MDN Web Docs, «Working with JSON,» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON. [Último acceso: 04 09 2020].

[74] MongoDB, «FAQ: MongoDB Fundamentals,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/faq/fundamentals/. [Último acceso: 04 09 2020].

[75] MongoDB, «Introduction to MongoDB,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/introduction/. [Último acceso: 04 09 2020].

[76] OWASP, «SQL Injection,» OWASP, [En línea]. Available: https://owasp.org/www-community/attacks/SQL_Injection. [Último acceso: 04 09 2020].

[77] MongoDB, «Databases and Collections,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/core/databases-and-collections/. [Último acceso: 04 09 2020].

[78] MongoDB, «The MongoDB 4.4 Manual,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/. [Último acceso: 04 09 2020].

[79] MongoDB, «MongoDB Atlas,» MongoDB, [En línea]. Available: https://www.mongodb.com/cloud/atlas. [Último acceso: 04 09 2020].

[80] IBM Cloud Education, «DBaaS (Database-as-a-Service),» IBM Cloud, 14 11 2019. [En línea]. Available: https://www.ibm.com/cloud/learn/dbaas. [Último acceso: 04 09 2020].

[81] MongoDB, «MongoDB Cloud,» MongoDB, [En línea]. Available: https://www.mongodb.com/cloud. [Último acceso: 04 09 2020].

[82] MongoDB, «An explanation of MongoDB Atlas' features and functionalities,» MongoDB, 11 07 2017. [En línea]. Available: https://www.youtube.com/watch?v=zbmMsLwMd-k. [Último acceso: 04 09 2020].

[83] MongoDB, «MongoDB Atlas Documentation,» MongoDB, [En línea]. Available: https://docs.atlas.mongodb.com/. [Último acceso: 04 09 2020].

[84] MongoDB, «Create a New Cluster,» MongoDB, [En línea]. Available: https://docs.atlas.mongodb.com/tutorial/create-new-cluster/. [Último acceso: 01 09 2020].

[85] MongoDB, «Glossary,» MongoDB, [En línea]. Available: https://docs.atlas.mongodb.com/reference/glossary/. [Último acceso: 01 09 2020].

[86] MongoDB, «Sharding,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/sharding/. [Último acceso: 01 09 2020].

[87] MongoDB, «Connect to a Cluster using Command Line Tools,» MongoDB, [En línea]. Available: https://docs.atlas.mongodb.com/command-line-tools/. [Último acceso: 01 09 2020].

[88] D. Petersen, «MongoDB Atlas free tier is now available on Azure,» Microsoft Azure, 12 11 2018. [En línea]. Available: https://azure.microsoft.com/es-es/blog/microsoft-azure-tutorial-how-to-integrate-azure-functions-with-mongodb/. [Último acceso: 01 09 2020].

[89] MongoDB, «Configuration Limitations,» MongoDB, [En línea]. Available: https://docs.atlas.mongodb.com/reference/free-shared-limitations/#atlas-free-tier. [Último acceso: 01 09 2020].

[90] MongoDB, «MongoDB Realm,» MongoDB, [En línea]. Available: https://docs.mongodb.com/realm/. [Último acceso: 01 09 2020].

[91] Red Hat, «What is serverless?,» Red Hat, [En línea]. Available: https://www.redhat.com/en/topics/cloud-native-apps/what-is-serverless. [Último acceso: 01 09 2020].

[92] MongoDB, «MongoDB Stitch & Mobile FAQ,» MongoDB, [En línea]. Available: https://www.mongodb.com/cloud/stitch/faq. [Último acceso: 01 09 2020].

[93] Apple, «iOS 13: Una nueva imagen. Mil nuevas posibilidades.,» Apple, [En línea]. Available: https://www.apple.com/es/ios/ios-13/. [Último acceso: 01 09 2020].

[94] Apple Developer, «Start Developing iOS Apps (Swift),» Apple, [En línea]. Available: https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/. [Último acceso: 01 09 2020].

[95] Android, «What's new this summer,» Android, [En línea]. Available: https://www.android.com/. [Último acceso: 01 09 2020].

[96] Android Developers, «Criar um projeto Android,» Android, [En línea]. Available: https://developer.android.com/training/basics/firstapp/creating-project. [Último acceso: 01 09 2020].

[97] Android Developers, «Interface do usuário e navegação,» Android, [En línea]. Available: https://developer.android.com/guide/topics/ui. [Último acceso: 01 09 2020].

[98] S. Richard y P. LePage, «What are Progressive Web Apps?,» web.dev, 06 01 2020. [En línea]. Available: https://web.dev/what-are-pwas/. [Último acceso: 01 09 2020].

[99] Codelabs Developers, «Introduction PWA,» Codelabs Developers, 30 04 2019. [En línea]. Available: https://codelabs.developers.google.com/codelabs/your-first-pwapp/#0. [Último acceso: 01 09 2020].

[100] C. Griffith, «What is Hybrid App Development?,» Ionicframework, [En línea]. Available: https://ionicframework.com/resources/articles/what-is-hybrid-app-development. [Último acceso: 01 09 2020].

[101] Apple Developer, «Apple Developer Program,» Apple, [En línea]. Available: https://developer.apple.com/programs/how-it-works/. [Último acceso: 01 09 2020].

[102] React Native, «Who's using React Native?,» React Native, [En línea]. Available: https://reactnative.dev/showcase. [Último acceso: 01 09 2020].

[103] Flutter, «Apps take flight with Flutter,» Flutter, [En línea]. Available: https://flutter.dev/showcase. [Último acceso: 01 09 2020].

[104] N. Dabit, React Native in Action: Developing iOS and Android apps with JavaScript, Manning Publications Company, 2019.

[105] Stack Overflow, «Stack Overflow Annual Developer Survey,» Stack Overflow, [En línea]. Available: https://insights.stackoverflow.com/survey. [Último acceso: 01 09 2020].

[106] Stack Overflow, «Developer Survey Results 2019,» Stack Overflow, 2019. [En línea]. Available: https://insights.stackoverflow.com/survey/2019. [Último acceso: 01 09 2020].

[107] Stack Overflow, «Developer Survey 2020,» Stack Overflow, 2020. [En línea]. Available: https://insights.stackoverflow.com/survey/2020. [Último acceso: 01 09 2020].

[108] Flutter, «Flutter GitHub Repository,» Flutter, [En línea]. Available: https://github.com/flutter/flutter. [Último acceso: 01 09 2020].

[109] React Native, «React-native GitHub Repository,» React Native, [En línea]. Available: https://github.com/facebook/react-native. [Último acceso: 01 09 2020].

[110] Octoverse, «Repositories,» Octoverse, [En línea]. Available: https://octoverse.github.com/2018/projects.html#repositories. [Último acceso: 31 08 2020].

[111] Octoverse, «The State of the Octoverse,» Octoverse, [En línea]. Available: https://octoverse.github.com/. [Último acceso: 31 08 2020].

[112] B. Eisenman, Learning React Native, O'Reilly Media, Inc., 2015.

[113] React Native, «Core Components and Native Components,» React Native, [En línea]. Available: https://reactnative.dev/docs/intro-react-native-components. [Último acceso: 01 09 2020].

[114] React, «Introducing JSX,» React, [En línea]. Available: https://en.reactjs.org/docs/introducing-jsx.html. [Último acceso: 01 09 2020].

[115] MND Web Docs, «Introduction to the DOM,» MND Web Docs, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. [Último acceso: 01 09 2020].

[116] React, «Components and Props,» React, [En línea]. Available: https://reactjs.org/docs/components-and-props.html. [Último acceso: 31 08 2020].

[117] React, «Props are Read-Only,» React, [En línea]. Available: https://reactjs.org/docs/components-and-props.html#props-are-read-only. [Último acceso: 31 08 2020].

[118] React, «React.Component,» React, [En línea]. Available: https://reactjs.org/docs/react-component.html. [Último acceso: 31 08 2020].

[119] T. Simonite, «Google May Have Finally Made a Truly Usable Voice Assistant,» WIRED, 05 08 2019. [En línea]. Available: https://www.wired.com/story/google-made-truly-usable-voice-assistant/. [Último acceso: 01 09 2020].

[120] P. L. Austin, «What Will Smart Homes Look Like 10 Years From Now?,» TIME, 25 07 2019. [En línea]. Available: https://time.com/5634791/smart-homes-future/. [Último acceso: 01 09 2020].

[121] IBM Cloud Docs, «Assistants,» IBM Cloud, 21 07 2020. [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-assistants&locale=en. [Último acceso: 01 09 2020].

[122] IBM Cloud, «Watson Assistant v2,» IBM Cloud, [En línea]. Available: https://cloud.ibm.com/apidocs/assistant/assistant-v2. [Último acceso: 01 09 2020].

[123] IBM Cloud Education, «Machine Learning,» IBM Cloud, 15 07 2020. [En línea]. Available: https://www.ibm.com/cloud/learn/machine-learning. [Último acceso: 01 09 2020].

[124] IBM Cloud, «About Watson Assistant,» IBM Cloud, 03 09 2020. [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-index. [Último acceso: 01 09 2020].

[125] IBM Cloud, «Watson Assistant,» IBM Cloud, [En línea]. Available: https://www.ibm.com/cloud/watson-assistant/. [Último acceso: 01 09 2020].

[126] IBM Cloud, «About Watson Assistant,» IBM Cloud, 03 09 2020. [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-index. [Último acceso: 01 09 2020].

[127] IBM Cloud, «User-based plans explained,» IBM Cloud, [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-services-information#services-information-user-based-plans. [Último acceso: 01 09 2020].

[128] IBM Cloud, «Explore Watson Assistant pricing plans to create an omni-channel customer experience,» IBM Cloud, [En línea]. Available: https://www.ibm.com/cloud/watson-assistant/pricing/. [Último acceso: 01 09 2020].

[129] Arduino, «ARDUINO UNO REV3,» Arduino, [En línea]. Available: https://store.arduino.cc/arduino-uno-rev3. [Último acceso: 14 08 2020].

[130] Arduino, «What is Arduino?,» Arduino, [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction. [Último acceso: 14 08 2020].

[131] FTDI Chip, «Future Technology Devices International Ltd. FT232R USB UART IC Datasheet,» FTDI Chip, [En línea]. Available:

https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf. [Último acceso: 01 09 2020].

[132] FTDI Chip, «Future Technology Devices Ltd.: The USB Bringing Solutions Specialists,» FTDI Chip, [En línea]. Available: https://www.ftdichip.com/FTDrivers.htm. [Último acceso: 01 09 2020].

[133] Fritzing, «Fritzing: electronics made easy,» Fritzing, [En línea]. Available: https://fritzing.org/. [Último acceso: 01 09 2020].

[134] Arduino, «Arduino Software (IDE),» Arduino, [En línea]. Available: https://www.arduino.cc/en/guide/environment. [Último acceso: 14 08 2020].

[135] C. Amariei, Arduino Development Cookbook, Packt Publishing Ltd., 2015.

[136] M. Margolis, Arduino Cookbook, O'Reilly Media, Inc., 2011.

[137] Wikipedia, «Hayes command set,» Wikipedia, [En línea]. Available: https://en.wikipedia.org/wiki/Hayes_command_set. [Último acceso: 14 08 2020].

[138] V. Shannon, «The Rise and Fall of the Modem King,» The New York Times, 07 01 1999. [En línea]. Available: https://www.nytimes.com/1999/01/07/news/the-rise-and-fall-of-the-modem-king.html. [Último acceso: 14 08 2020].

[139] Jinan Huamao, «Jinan Huamao Technology Limited Company,» Jinan Huamao, [En línea]. Available: http://jnhuamao.cn/. [Último acceso: 13 08 2020].

[140] Jinan Huamao, «Download Center,» Jinan Huamao, [En línea]. Available: http://jnhuamao.cn/index_en.asp?ID=1. [Último acceso: 01 09 2020].

[141] Apple, «Locate Beacon,» Apple, [En línea]. Available: https://apps.apple.com/us/app/locate-beacon/id738709014. [Último acceso: 31 08 2020].

[142] N. Bridoux, «Beacon Scanner,» Google Play, [En línea]. Available: https://play.google.com/store/apps/details?id=com.bridou_n.beaconscanner&hl=en. [Último acceso: 31 08 2020].

[143] Bluepixel Technologies LLP, «BLE Scanner 4.0,» Apple, [En línea]. Available: https://apps.apple.com/us/app/ble-scanner-4-0/id1221763603. [Último acceso: 01 09 2020].

[144] V. Hiribarren, «Beacon Simulator,» Google Play, [En línea]. Available: https://play.google.com/store/apps/details?id=net.alea.beaconsimulator&hl=en. [Último acceso: 31 08 2020].

[145] PyPI The Python Package Index, «pip 20.2.2,» PyPI, [En línea]. Available: https://pypi.org/project/pip/. [Último acceso: 19 08 2020].

[146] Python, «Creation of virtual environments,» Python, [En línea]. Available: https://docs.python.org/3/library/venv.html. [Último acceso: 19 08 2020].

[147] Materialize, «Documentation Materialize,» Materialize, [En línea]. Available: https://materializecss.com/. [Último acceso: 20 08 2020].

[148] Python, «Flask-Security,» Python, [En línea]. Available: https://pythonhosted.org/Flask-Security/. [Último acceso: 19 08 2020].

[149] MongoDB, «Sparse Indexes,» MongoDB, [En línea]. Available: https://docs.mongodb.com/manual/core/index-sparse/. [Último acceso: 19 08 2020].

[150] Google Maps Platform, « Maps JavaScript API,» Google Maps Platform, [En línea]. Available: https://developers.google.com/maps/documentation/javascript/examples/event-click-latlng. [Último acceso: 19 08 2020].

[151] Mongo Engine, «Mongo Engine,» Mongo Engine, [En línea]. Available: http://mongoengine.org/. [Último acceso: 19 08 2020].

[152] MongoDB, «MongoDB Pricing,» MongoDB, [En línea]. Available: https://www.mongodb.com/pricing. [Último acceso: 20 08 2020].

[153] Mongo Engine, «Flask-MongoEngine: Configuration,» Mongo Engine, [En línea]. Available: http://docs.mongoengine.org/projects/flask-mongoengine/en/latest/#configuration. [Último acceso: 20 08 2020].

[154] Mongo Engine, «Defining a document's schema,» Mongo Engine, [En línea]. Available: https://docs.mongoengine.org/guide/defining-documents.html#defining-a-document-s-schema. [Último acceso: 20 08 2020].

[155] IDC Corporate, «Smartphone Market Share,» IDC Corporate, [En línea]. Available: https://www.idc.com/promo/smartphone-market-share/os. [Último acceso: 01 09 2020].

[156] Android Studio, «Executar apps no Android Emulator,» Android Studio, [En línea]. Available: https://developer.android.com/studio/run/emulator. [Último acceso: 16 08 2020].

[157] Android Developers, «Executar o app,» Android Developers, [En línea]. Available: https://developer.android.com/training/basics/firstapp/running-app. [Último acceso: 16 08 2020].

[158] Oracle, «Java SE Downloads,» Oracle, [En línea]. Available: https://www.oracle.com/java/technologies/javase-downloads.html. [Último acceso: 01 09 2020].

[159] Python, «Python,» Python, [En línea]. Available: https://www.python.org/. [Último acceso: 01 09 2020].

[160] React Native, «Setting up the development environment,» React Native, [En línea]. Available: https://reactnative.dev/docs/environment-setup. [Último acceso: 16 08 2020].

[161] Metro, «Concepts,» Metro, [En línea]. Available: https://facebook.github.io/metro/docs/concepts/. [Último acceso: 16 08 2020].

[162] National Party Month, «React Native Bluetooth Status,» National Party Month, [En línea]. Available: https://www.npmjs.com/package/react-native-bluetooth-status. [Último acceso: 31 08 2020].

[163] Node, «What is npm?,» Node, 26 08 2011. [En línea]. Available: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/. [Último acceso: 17 08 2020].

[164] NPM, «About npm,» NPM, [En línea]. Available: https://www.npmjs.com/about. [Último acceso: 17 08 2020].

[165] NPM, «Build amazing things,» NPM, [En línea]. Available: https://www.npmjs.com/. [Último acceso: 17 08 2020].

[166] React, «State and Lifecycle,» React, [En línea]. Available: https://en.reactjs.org/docs/state-and-lifecycle.html. [Último acceso: 17 08 2020].

[167] React Native, «PermissionsAndroid,» React Native, [En línea]. Available: https://reactnative.dev/docs/permissionsandroid. [Último acceso: 17 08 2020].

[168] NPM, «Mongodb Stitch React Native Sdk,» NPM, [En línea]. Available: https://www.npmjs.com/package/mongodb-stitch-react-native-sdk. [Último acceso: 17 08 2020].

[169] A. Morgan, «Building iOS and Android Apps with the MongoDB Stitch React Native SDK,» MongoDB, 03 12 2018. [En línea]. Available: https://www.mongodb.com/blog/post/building-ios-and-android-apps-with-the-mongodb-stitch-react-native-sdk. [Último acceso: 17 08 2020].

[170] NPM, «React Native Beacons Manager,» NPM, [En línea]. Available: https://www.npmjs.com/package/react-native-beacons-manager. [Último acceso: 17 08 2020].

[171] NPM, «React Native Snap Carousel,» NPM, [En línea]. Available: https://www.npmjs.com/package/react-native-snap-carousel. [Último acceso: 17 08 2020].

[172] NPM, «React Native Community Voice,» NPM, [En línea]. Available: https://www.npmjs.com/package/@react-native-community/voice. [Último acceso: 17 08 2020].

[173] IBM Cloud, «Watson Speech to Text,» IBM Cloud, [En línea]. Available: https://www.ibm.com/es-es/cloud/watson-speech-to-text. [Último acceso: 17 08 2020].

[174] IBM Cloud, «Watson Text to Speech,» IBM Cloud, [En línea]. Available: https://www.ibm.com/es-es/cloud/watson-text-to-speech. [Último acceso: 17 08 2020].

[175] NPM, «React Native Tts,» NPM, [En línea]. Available: https://www.npmjs.com/package/react-native-tts. [Último acceso: 17 08 2020].

[176] IBM Cloud, «Creating intents,» IBM Cloud, 29 07 2020. [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-intents. [Último acceso: 22 08 2020].

[177] Museo Nacional Centro de Arte Reina Sofia, «Sedes y espacios,» Museo Nacional Centro de Arte Reina Sofia, [En línea]. Available: https://www.museoreinasofia.es/museo/sedes. [Último acceso: 22 08 2020].

[178] IBM Cloud, «Getting started with Watson Assistant,» IBM Cloud, 10 08 2020. [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started. [Último acceso: 22 08 2020].

[179] IBM Cloud, «Building a conversational flow,» IBM Cloud, 02 09 2020. [En línea]. Available: https://cloud.ibm.com/docs/assistant?topic=assistant-dialog-overview. [Último acceso: 22 08 2020].

[180] N. Srnicek, Platform Capitalism, John Wiley & Sons, 2017.

[181] P. High, «Gartner: Top 10 Strategic IT Trends For 2015,» Forbes, 07 10 2014. [En línea]. Available: https://www.forbes.com/sites/peterhigh/2014/10/07/gartner-top-10-strategic-it-trends-for-2015/. [Último acceso: 01 09 2020].

[182] Gartner, «Gartner Identifies the Top 10 Strategic Technology Trends for 2020,» Gartner, 21 10 2019. [En línea]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-10-21-gartner-identifies-the-top-10-strategic-technology-trends-for-2020. [Último acceso: 01 09 2020].

[183] L. Goasduff, «The Future City in a Smart World,» Gartner, 19 05 2015. [En línea]. Available: https://www.gartner.com/smarterwithgartner/the-future-city-in-a-smart-world/. [Último acceso: 01 09 2020].

[184] Deloitte, «Smart Cities: How rapid advances in technology are reshaping our economy and society,» Deloitte, 11 2015. [En línea]. Available: https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/public-sector/deloitte-nl-ps-smart-cities-report.pdf. [Último acceso: 01 09 2020].

[185] Deloitte University Press, «Tech Trends 2015: The fusion of business and IT,» Deloitte University Press, 2015. [En línea]. Available: https://www2.deloitte.com/content/dam/Deloitte/mx/Documents/technology/Tech-Trends-2015-FINAL.pdf. [Último acceso: 01 09 2020].

[186] L. Cecco, «Google affiliate Sidewalk Labs abruptly abandons Toronto smart city project,» The Guardian, 07 05 2020. [En línea]. Available: https://www.theguardian.com/technology/2020/may/07/google-sidewalk-labs-toronto-smart-city-abandoned. [Último acceso: 01 09 2020].

[187] F. Barth, «Alguns detalhes técnicos e desafios do projeto "A voz da arte – IBM | Watson",» LinkedIn, 11 04 2017. [En línea]. Available: https://www.linkedin.com/pulse/alguns-detalhes-t%C3%A9cnicos-e-desafios-do-projeto-voz-da-fabr%C3%ADcio-barth/. [Último acceso: 24 08 2020].

[188] L. A. Bucki, Word 2013 Bible, John Wiley & Sons, 2013.

[189] CFI, «Cursos de Formación en Informática,» [En línea]. Available: http://cursosinformatica.ucm.es. [Último acceso: 01 06 2019].

[190] WSGI, «What is WSGI?,» WSGI.org, [En línea]. Available: https://wsgi.readthedocs.io/en/latest/what.html. [Último acceso: 03 09 2020].

[191] Python, «PEP 3333: Python Web Server Gateway: Interface v1.0.1,» Python, [En línea]. Available: https://www.python.org/dev/peps/pep-3333/. [Último acceso: 03 09 2020].

[192] B. Feldman, «The Hidden Power of Stack Overflow: How a Website You've Never Heard of Is Holding the Web Together,» Intelligencer, 24 03 2017. [En línea]. Available: https://nymag.com/intelligencer/2017/03/the-hidden-power-of-stack-overflow.html. [Último acceso: 01 09 2020].

[193] Python, «Hashlib: Secure hashes and message digests,» Python, [En línea]. Available: https://docs.python.org/3/library/hashlib.html. [Último acceso: 19 08 2020].

[194] Python, «Hexdigest,» Python, [En línea]. Available: https://docs.python.org/3/library/hashlib.html#hashlib.hash.hexdigest. [Último acceso: 19 08 2020].

[195] Bluetooth, «Bluetooth 5: Go Faster. Go Further.,» [En línea]. Available: https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf. [Último acceso: 28 08 2020].