Universidade do Minho
Escola de Engenharia
Departamento de Informática

Rafael Machado da Silva

Mobile application for tracking cycling activity

June 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Rafael Machado da Silva

Mobile application for tracking cycling activity

Master's Dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Rui João Peixoto José

June 2021

COPYRIGHT AND TERMS OF USE FOR THIRD PARTY WORK

ACKNOWLEDGEMENTS

I would like to thank my parents for the opportunity they gave me to obtain a higher education degree, for all the patience, dedication and motivation they gave me throughout these years, because without them I would not be the person I am today, and I would not be where I am at this moment. I also leave here a special thanks to my brother and my godmother for helping me in what I needed and for contributing with their time and attention. Without forgetting my grandmother for all the good advices she gave me.

To all the teachers of the Integrated Master's Degree in Computer Engineering, who actively contributed along my academic path, without their help I would not have reached the objectives set in obtaining this higher degree, because this dissertation is the fruit of the work and knowledge that I acquired thanks to them.

To my supervisor Professor Rui João Peixoto José for all his time, dedication, motivation, values and knowledge he had with me, as it is with his help that this dissertation was developed.

To my friends for understanding my absence.

A very special affection and thanks to Adriana, for all the things she has done and will continue to do.

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project no 039334; Funding Reference: POCI-01-0247-FEDER-039334]

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

In recent decades there has been significant growth in cycling and the use of bicycles as a means of transport and beyond recreational use. Technological advances along with smartphones and web platforms can help cyclists as they consider and plan their routes. This brings several social, economic and environmental benefits.

Technology is now present in all aspects of modern life, and that includes cycling. This *"technological revolution of cycling"* is a phenomenon that attracts more and more attention and a variety of interests, fuelled both by the processes of transition to intelligent mobility and by a growth in attention to cycling in cities around the world.

This is where this dissertation comes in, which aims to develop the prototype of a mobile application for cycling urban mobility, more specifically to be used by cyclists as a digital tool to support the various aspects of this form of mobility. This application prototype will have the opportunity to introduce itself in a context that is not much explored yet, being able to allow the cyclist to make a manual tracking or autonomous tracking of his travels, thus making that mobility in that city can be investigated and at the same time improved.

It is about thinking of an application that has been designed with a global purpose but has a local behaviour, thus being able to operate in any region that is being used, making it a versatile application without any kind of restriction.

KEYWORDS: Cycling, Mobile Application, Smart Cities, Safety, Navigation, Routes;

RESUMO

Nas últimas décadas tem havido um crescimento significativo no ciclismo e na utilização de bicicletas como meio de transporte e para além do uso recreativo. Os avanços tecnológicos juntamente com os telemóveis e plataformas web podem ajudar os ciclistas enquanto consideram e planejam as suas rotas. Isto traz vários benefícios sociais, económicos e ambientais.

A tecnologia está agora presente em todos os aspectos da vida moderna, e isso inclui o ciclismo. Esta *"revolução tecnológica do ciclismo"* é um fenômeno que atrai cada vez mais a atenção e uma variedade de interesses, alimentada tanto pelos processos de transição para a mobilidade inteligente como por um crescimento de atenção ao ciclismo nas cidades de todo o mundo.

É aqui que entra esta dissertação que visa o desenvolvimento do protótipo de uma aplicação móvel para a mobilidade urbana ciclável, mais especificamente para ser usada pelos ciclistas como ferramenta digital de apoio a diversas vertentes dessa forma de mobilidade. Este protótipo de aplicação terá a oportunidade de se introduzir num contexto que ainda não é muito explorado, podendo permitir ao ciclista fazer um rastremento manual ou autônomo das viagens do mesmo, fazendo assim com que a mobilidade naquela cidade possa ser investigada e ao mesmo tempo melhorada.

Trata-se de pensarmos numa aplicação que tenha sido pensada com um propósito global mas que tenha um comportamento local, sendo assim capaz operar em qualquer região que esteja a ser utilizada, fazendo assim da mesma uma aplicação versátil e sem qualquer tipo de restrição.

PALAVRAS-CHAVE: Ciclismo, Aplicação Móvel, Cidades Inteligentes, Segurança, Navegação, Rotas;

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

ACRONYMS

AWS Amazon Web Services. 64, 65

CO_2 Carbon Dioxide. 1, 7, 12, 14, 15, 91

DSR Design Science Research. 18, 22

EC European Commission. 11, 12
ECF European Cyclists' Federation. 2
EU European Union. 1, 13

IoT Internet of Things. 4, 7, 8, 9

JSON JavaScript Object Notation. 42, 44, 59, 60, 62

NoSQL Not Structured Query Language. 23, 43

OECD Organisation for Economic Co-operation and Development. 9, 10

SQL Structured Query Language. 23

UI User Interface. 60, 61
UML Unified Modeling Language. 34, 39
UN United Nations. 11

1

INTRODUCTION

In this chapter I will present and explain the contextualisation (section 1.1), the motivation of this dissertation (section 1.2), the problem and it's main challenges of the same (sections 1.3, 1.4), then the objectives of this dissertation (sections 1.5), and finally the structure of this dissertation (section 1.6).

1.1 CONTEXTUALISATION

In these times we have the theme of reducing the use of cars to be used by big cities [1], i.e. for the purpose of people to start using bicycles as a means of transport, in order to reduce congestion and people to adopt a more sustainable mode of transport.

This dissertation it is inserted in a sub-project called *"connected 2-wheelers"*, which is also part of the *"Easy Ride"* project. These two projects aim to improve urban mobility and transport, in the case of the *"connected 2-wheelers"* sub-project, its focus is on two-wheeled vehicles, and how to improve their mobility and their interaction in a smart city, thus promoting alternative and sustainable means of transport.

Cycling is assuming an increasing role in sustainable mobility policies. Leading cities and central governments all over the world are making significant investments to bring cycling, and other micro-mobility modes, to the forefront of their mobility strategies [2]. This transition is being fuelled by a combination of sustainability [3] in public health [4, 5], urban life [6], and economic agendas [7, 8].

This has become a popular theme in the last decade because of rising Carbon Dioxide (CO_2) levels and countries want to contribute to an evolution towards alternative means of transport, which benefit the environment as well as sustainability. Examples of this are the European Union (EU) which is more receptive to such ideas, with the aim of reducing our ecological footprint in the world.

This has also been seen with the significant growth in recent decades of cycling and the use of bicycles as an alternative means of transport [9], with this in mind we can then take advantage of the technology at our disposal today to make the most of this new trend by helping cyclists in their daily lives.

To make the most of this technology it is necessary that it is an extension of the bicycle, making it easy and quick to use, we have several examples where the application helps cyclists to trace a route they choose, as well as avoiding areas of great congestion and we even have cases of the use of this technology helping cyclists to make daily travels where other cyclists have access to them, transforming a city, into an smart city where we have all its inhabitants connected to each other, enabling the exchange of information, as well as the creation of an advanced civilization that can transform simple life on a big city.

An example of an application to be used by cyclists, is *"Greenbikenet"* [10] which has the function of the cyclists to connect to each other and share information with each other, also have the possibility to geologically track the cyclists and show useful information for them. These platforms have a responsibility to maintain the safety of cyclists as well as to assist them in their daily life, focusing on the concept of using the resources available in that particular city to make urban cycling a more attractive activity.

1.2 MOTIVATION

As explained above, urban cycling is an activity that is attracting more and more people, due to this increase in demand we have seen a political will to create smart cities, these aim to make the life of a citizen simpler and more technological. This makes urban and suburban centres become connected for the first time, collaborating with each other and constantly learning, in this case smart cities improve the collective intelligence of a city, in this case with the implementation of cycling in intelligent and interconnected transport networks of the future. This makes data extraction important for the dynamism of a smart city, as it is due to this data that we can have a better "picture" of the mobility's within it.

These smart cities will be driven by technology, which, if properly implemented, has the power to introduce behavioural change, such change is necessary for the widespread adoption of new concepts and more intelligent and cleaner modes of transport, thus creating a new paradigm denoted ***smart mobility***, organisations like European Cyclists' Federation (ECF) are trying to spread the word and encourage this new paradigm to succeed and grow in the coming years [11].

Taking this into account we can say that the way we move around the city, in this case the cycling, will also be impacted by this new emerging paradigm of ***smart mobility***, this new paradigm is a main pillar of any society that wanted to *"build"* an smart city. Cycling should be included in this new emerging system. To do so, the bicycle itself must evolve into a more ***smarter cycling***, the cyclist can thus perform various services, or be always up to date with new information and events within their own city.

1.3 PROBLEM

With the emergence of the new paradigm of ***smart mobility***, we have to think about how in our case we can use it to benefit urban cycling. This makes us think about developing a mobile application that can help cyclists, where it can be a shared and collaborative application, even working with data in real time.

The problem with such an application is that users, in this case cyclists, are mostly anonymous and will have to compete with other forms of transport, and most of them are not heard by society, this makes urban transport planners have little precise and comprehensive way of counting, monitoring and taking into account the real-time movement of citizens cyclists through the urban environment. This means that they have little knowledge about how to improve the situation. So we need to move from the mechanical and individual cycling of today to the cycling of tomorrow, as best know as ***smart cycling***.

Another aspect in consideration is the development of the application regarding the location of the cyclist who will use it, because we need to create an application that works every where, because this application will be used to collect vital and crucial information of a certain city, so we have to think about a global application, in which it has a local behaviour and can be used in any location at any time, focusing on the concept of *"GLocal"* that will be explained later.

Taking what was said above into consideration it is also important to think not only in a simple cycling application, but also in an adaptive application, which depends on where it is used and depends on the ecosystem connections made by its user.

1.4 MAIN CHALLENGES

Taking what has been said above as a starting point, we can soon take a number of challenges that this problem may bring. As explained above, the problem presented here is in an area where little research or development work has been done, i.e. I will not have much supporting material or a guide on how to proceed. We can also think of the application in question as being able to function anywhere and also to function autonomously, that is, without the cyclist having to interact with it. These will be the main challenges that the subject of this dissertation may have, then I will address its objectives and what is the purpose of this dissertation, which are based on the prototyping of a new concept of mobile application for urban and sustainable mobility.

This problem brings with it its challenges, the most important being how to incorporate such an application into a developing smart city, and how to do it in such a way that it works not only in one city, but using the concept of *"GLocal"*, where it can be used in any city and work equally. This way of thinking is already a great challenge, because for it to

have this functionality, and that the application has to assume an autonomous behaviour regarding the tracking of the travels of the user.

A major challenge will also be to find out what services a given city has in mind and/or wants to create, for the application to work correctly and cohesively, this already has to do with the development part of the smart city, and in which sense local entities want the application to help them improve the quality of life and mobility in the city.

1.5 OBJECTIVES

With this in mind we can associate this new era with the increase of objects in our lives that are connected and that also produce, share and process data, also known as the Internet of Things (IoT), taking this as a principle we can consider that we will be creating a cycling network [12].

With this dissertation topic it is intended to do a research work as well as to develop a prototype of a mobile application. This prototype application is mainly intended as a research tool for a deeper understanding of the integration between *smartphones* and cycling practices, seeking to discover opportunities that offer the best fit between them. The main result of the work will be a specification and prototype code base to support the development of a public mobile application with a set of selected features.

Here we have the objectives summarise in the context of the present topic in order to corroborate the scope of this dissertation:

- Objective 1 - Development a prototype of a mobile application;
  - Using the smartphone's ability to collect the cyclist's current position in real time.
  - Improving sustainable mobility.
- Objective 2 - Manual and Autonomous Tracking;
  - The user will be able to choose which of their travels they wish to collect and its travels can be collected autonomously.
  - There may be challenges in the autonomous tracking.
- Objective 3 - Data collect;
  - The data collected here provides insight into cyclists' mobility research;
  - They could be used by local authorities to improve cycling mobility;
  - Helping cyclists to improve their day to day lives.

1.6 STRUCTURE OF THE DISSERTATION

- Chapter 1 – Introduction;
  - In this chapter, which is the present chapter where this section is located, the following topics about this dissertation will be explained: (1) the contextualisation, (2) motivation, (3) problem on which it is based, (4) the main challenges of this dissertation, (5) the objectives of this dissertation and finally (6) the planning of this dissertation.
- Chapter 2 – State of the Art;
  - In this chapter all the research work on this dissertation is shown. It will be divided into several sections, and in the first part I will talk about concepts and important terms for this dissertation, then I will show some examples of prototypes of applications similar to the one intended to be developed in this dissertation.
- Chapter 3 – Methodologies and Technologies;
  - This chapter serves to show the methodologies that exist in the research world, as well as in the development world, it also serves to show the methodology that was followed. Next I will also talk about the technologies that will be used during the development of this prototype.
- Chapter 4 – Analysis and proposed approach;
  - This will be the first section of the second part of this dissertation, in this one I will expose the requirements and their survey, as well as the software modulation that will contain a use cases diagram, a domain model, and finally a class diagram. Finally I will have a section where I will explain the approach that was taken here to develop the application prototype.
- Chapter 5 – Development of the prototype;
  - In this section the whole development process will be explained, it was divided into three parts: (1) the database part, where it will be explained the same, and how it was made and developed, (2) I will explain the Back-End part and how it was developed and applied, and finally (3) I will talk about the prototype of the application and about its development process that is in the treated in the Front-End section.
- Chapter 6 – Interfaces of the prototype;
  - This section serves to show the main interfaces of the prototype application that were developed, as well as a short explanation and operation of them.

- Chapter 7 – Review and Analysis;
  - In this section an explanation and demonstration of the features that the prototype application has, which are in accordance with the requirements that were elaborated here, is elaborated, then we have a critical analysis and possible aspects that could be improved in the application as well as a general analysis of all the work in the development process.
- Chapter 8 – Conclusions and Future work.
  - Finally, this section will present a conclusion of all the work that has been developed in this dissertation as well as its contributions, and at the end together with a possible work that could be carried out in the future.

2

STATE OF THE ART

One of the main features and advantages of using a mobile application to encourage sustainable mobility and cycling to contribute to ***smart mobility***, is the fact that we are living in the era of great data, because at the moment having data is having knowledge and power, so *smartphones* and software platforms will play a vital role in this theme, because they will be responsible for collecting data from each travel as well as behaviour data, this data collection has been proven in several ways, as being better than the data collected by normal mobility surveys. Studies done on this subject show that this data collection is of great advantage, as they can be visualised on software platforms, thus obtaining a better understanding of them, and at the same time we have the study on a city about its mobility, thus actively contributing to a better sustainable mobility [1].

Overall, measures to improve mobility in large cities are being identified, such as the use of bicycles as a serious and alternative means of transport, as it contributes to the reduction of CO_2 and traffic in the city itself. Local entities themselves are providing infrastructure to encourage the growth of smart cycling, as they will benefit most from this increase.[9]

In this chapter I will address the concept of IoT and ***smart city*** and what this encompasses, and the concept of ***GLocal***.

In the remainder chapter the following different topics will be explained: (1) the prototypes or applications for bicycles or urban cycling will be shown, these have been developed for non-profit purposes and are intended to do research work on one of the topics of this dissertation, in which they have been financed by public entities. (2) Next I will explain and expose the applications that are more focused on the use of the bicycle as a means of sport, these applications were created by profit-making companies, where they are based on a market share, as well as providing certain functionalities on which the user will have to pay, and (3) finally I will talk about the applications that are more related and with the theme of this dissertation, these have almost all aspects considered here and were made as a research tool, without any profit-making purpose.

2.1 INTERNET OF THINGS (IOT)

The Internet of Things, better known as IoT, is a growing theme and concept, it focuses on people and networked devices (e.g. smartphones), this concept consists of a large number of objects interconnected in a network, which make them objects of information and communication technologies [13]. This technology allows us to store, send and receive information in ways that can transform our daily lives and how we do things. This is a key area of growth for industry, policy and research, mobility is also of great importance for transport, illustrated by the applications of these technologies for ***smart transport*** and ***smart mobility*** [13].

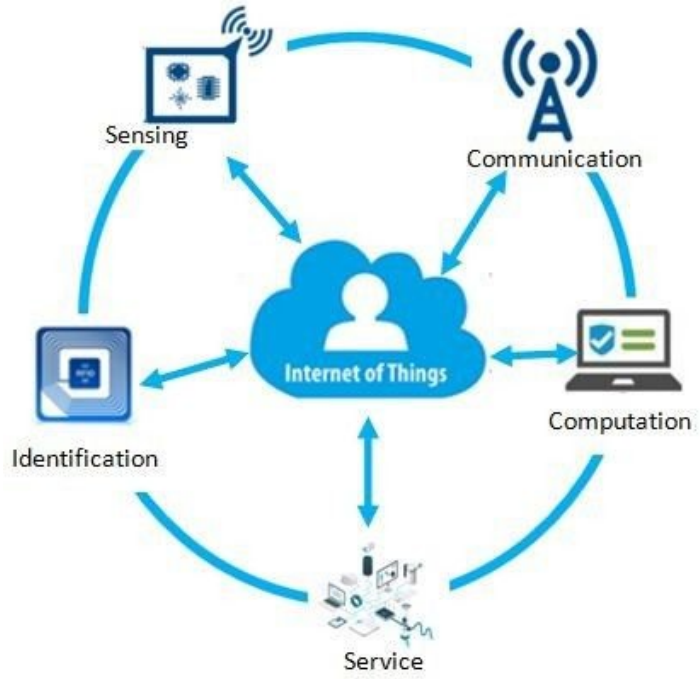


Figure 1: Example model of IoT
(Source: [14])

This concept can be seen as a paradigm that foresees in the near future, that the objects that people use in their daily lives can be equipped with micro-controllers and software platforms, which are capable of carrying out digital communication, allowing them to communicate among themselves and with users, becoming a module of the Internet itself. With this in mind, we can then state that the IoT concept aims to make the Internet more penetrating and immersive, thus allowing easy access and interaction with a large number and variety of digital devices, such as domestic appliances, surveillance cameras, monitoring, sensors, actuators, screens, vehicles, etc [15]. This paradigm is linked to several different domains, among them all is to highlight the concept of smart mobility that has a great impact on smart cities.

Urban IoTs are focused and designed to support a ***smart city*** vision, which aims to investigate and make the most of the communication technologies at our disposal, so that they play a role in city administration services as well as helping citizens by providing a comprehensive survey and information on the possibilities and ideas for developing a smart city [15].

As mentioned above IoT has a lot of attention on the context of smart cities, as this technology is responsible for creating applications that can take advantage of the user as a source of information in the context of smart cities [16], as their help can *"digitize"* the city, i.e. they can be mobile sensors with their smartphones and can thus become headlights in order to provide a lot of data and information relevant to mobility, this makes IoT play an important role in the development of a smart city, as well as in the use of mass cycling to contribute to smart mobility.

This topic it's very important because this mobile application need to be connect to the Internet, and the throw the concept of IoT we can make that.

2.2 SMART CITY

In this section I will talk a bit about the ***smart city*** concept and what it represents, as well as some components that constitute it, such as the concept of ***smart mobility***, ***smart cycling*** and by concept of ***sustainable transport***, for this purpose I will address some concepts explained by Organisation for Economic Co-operation and Development (OECD) in a conference about Smart Cities.

The concept of ***smart city*** initially emerged by encouraging the use of new technologies, taking advantage of the emergence of digital innovation in which we find ourselves to improve the effectiveness and efficiency of urban services within the city, as well as generate new employment and economic opportunities in these [17]. This concept is still growing and is still the subject of several debates. The very definition of ***smart city*** varies across the OECD countries, as they have different policy agendas and different perspectives on the world we live in.

However, in most cases this concept is based on initiatives using digital transformation and software platforms, so that the urban services provided by local authorities become more efficient, so that in the global context the city gains more competitiveness. As has been said, this concept depends on the investment in technological innovation, now this may bring us the question if the investment in new technologies can better the life and well-being of citizens, according to OECD this investment must be made around the human being, because it will transform a city into a ***smart city*** [17].

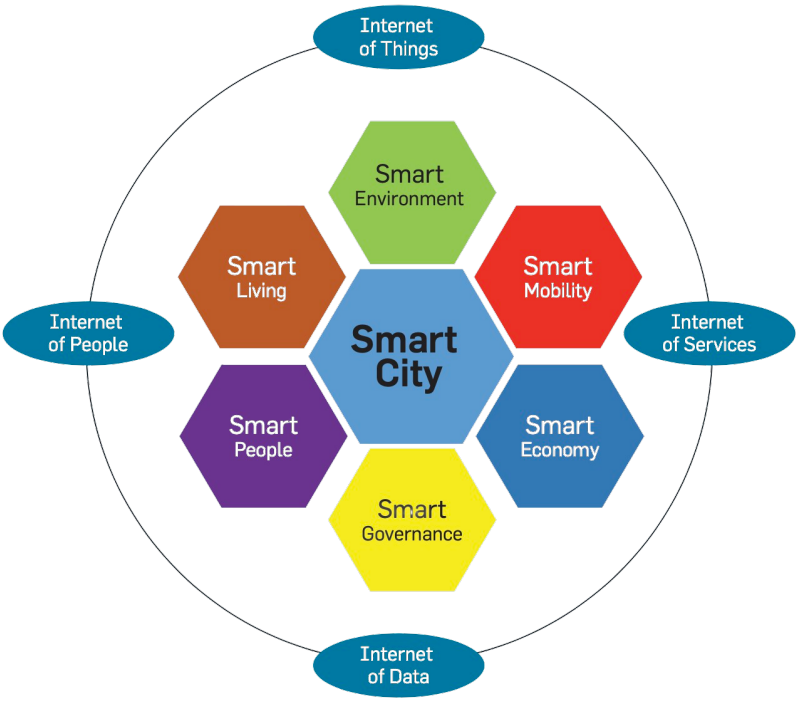


Figure 2: A smart city model
(Source: [18])

The figure 2 shows a possible model of what a ***smart city*** would look like, like all its interconnected components and modules.

OECD tells us that a technology approach in relation must be made with initiatives that drive the process of digital transformation and the use of software platforms in order to create a better quality of life and welfare of citizens and at the same time to provide services and a more sustainable and efficient urban environment so that citizens collaborate with each other, thus making a city interconnected and the progeny of the people who live there [17].

2.2.1 *Smart Mobility*

The term ***smart mobility*** is very much related to the above mentioned theme and explained that it is the case of ***smart city***, because in a city that wanted to become a ***smart city*** will also have to think about how it can become more efficient and intelligent at transport levels, thus achieving the status of ***smart mobility***, there is not a single definition for this term in concrete, below I will explain its various definitions as well as its use.

According to the United Nations (UN), some 54% of the world's population live in urban areas, and by the year 2050 this figure could rise to 60% [19]. With this in mind we can then assume that in 30 years time a large majority of the world's population will be living in urban centres, since we have to prepare these centres for an increase in their means of transport as well as their efficiency, that is where the concept of ***smart mobility*** will come in.

The term ***smart mobility*** can be defined by an intelligent transport ecosystem based on advanced application systems that provide services related to the different means of transport (e.g. road, rail, public transport, cycling, etc.) and traffic management in the city. This definition has evolved over the past decades in line with technological developments in the world and its needs. Taking into account what has been said above, we can then assume that smart mobility has the objective of increasing the quality of life of citizens and at the same time providing a new level of mobility in urban areas [20].

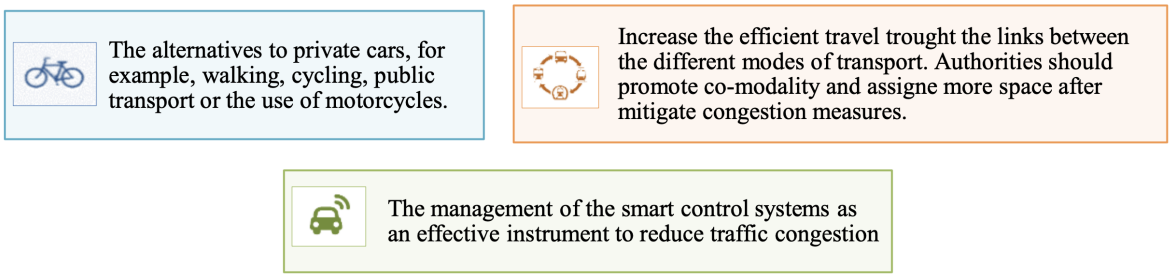


Figure 3: Sketch of European Commission's about Smart Mobility
(Source: [21])

Worldwide most of the references we find when it comes to smart mobility are always interconnected with the improvement of traffic in a city, but this is not the only problem that a smart city will face due to its expansion, because a growing city has to remain interconnected and function as a community, this issue is of great importance in our times, the functioning of the means of transport, its efficiency and sustainability are crucial for a good development of the economy and daily life of a city [21]. The European Commission (EC) establishes different relationships with regard to intelligent mobility as we can see in the figure 3.

2.2.2 *Smart Cycling*

The term ***smart cycling*** comes from the need to transform a means of transport that few people use and that is very basic, and to make it an alternative and attractive means of transport where people can use it in their daily lives, in this case it will be the use of the bicycle as an alternative means of transport, more sustainable and more environmentally friendly.

The entire transport network in a city is already designed to be interconnected, and when I speak of transport networks, I speak of public, individual and goods transport [11], not including the bicycle, taking this into account it is necessary to protect and encourage the use of the bicycle and that is why this module associated with the ***smart city*** was created. Thus, the importance of bicycles is necessary for people to be able to travel in a sustainable, efficient, intelligent and effective way within metropolitan areas, *"Smart velomobilities"* initiatives, policies and an innovation of bicycle use are important to define cycling as an active and sustainable means of transport [16].

Thanks to the technological innovation that has emerged in recent decades, it is possible to improve and transform the experience of cycling from a monotonous to an innovative thing. We see this happening when technology and cycling companies start to create cycling gadgets. This phenomenon can be denoted as *"smartification of cycling"* [22], and thanks to this more and more people are becoming attracted to using the bicycle as an alternative means of transport for their daily lives, taking advantage of the growth in intelligent mobility and increased attention to cycling from cities around the world.

As we can see, the bicycle is increasingly becoming an alternative means of transport in our world, and thanks to this we need to transform it and shape it to the technological world in which we live. The EC has been trying to publicise the positive effects of cycling, especially how cycling improves public health and the environment [13]. Every day we have more and more communities of cyclists growing up in smart cities and being interconnected with each other and with the city, thus enhancing ***smart mobility*** and making the bicycle an intelligent means of transport with the help of the technology at our disposal.

2.2.3 *Sustainable Mobility*

Over the years and due to climate change, the theme of sustainable mobility has become predominant as the world is united in reducing CO_2 emissions to protect the environment and thus combat climate change by making means of transport efficient and environmentally friendly, promoting the concept of sustainable mobility.

This concept comes from the need as smart cities develop, there is a concern to keep them sustainable [23]. As these cities have a large increase in population and a rapid urbanisation, make them highly dependent on the use of private cars, this raises a concern about the welfare of people and our existence in the long term. Measures will have to be taken to counter these consequences in order to introduce more sustainable and environmentally friendly mobility practices into society. In view of this, we can say that the concept of sustainable mobility has a major impact on large metropolitan areas in the long term, reducing pollution in them as well as traffic.

With this in mind we should focus on measuring the performance of a city through the development of sustainable cities, but also on bringing together various ideas or suggestions on how to make some urban changes so that it leads to sustainable cities, especially when it comes to mobility and transport, so that policy and city planners can help and adjust their policies on this issue [16].

The concept of ***smart city*** it's important in this dissertation because it's set the foundation of the development of this mobile application, because this platform will be use in this context, with the help of the three concepts that belongs to a smart city.

2.3 GLOCAL STRATEGY

This term emerged in the development of the ***smart city*** concept, it portrays a smart city as a *"GLocal"* phenomenon, i.e. it considers global and local aspects at the same time, because of this we can then associate smart cities with this term, because they have spread throughout the world over the past few years, having similar characteristics and interdependent globally, however smart cities are a local phenomenon, because each city is independent of each other and unique in its individual characteristics, having different problems and different solutions to solve them [24].

As it was said before, we can say then that this concept is of great importance in what concerns the development of smart cities, and in this problem in question, because it has a very arousing and easy to understand definition, which is when thinking about developing a service, it is always important to think globally and not locally, only this way we can be sure that this service will work worldwide in any city, according to its needs and working in the same way, for example in Lisbon and/or Toronto.

With this we can have an idea how to make this mobile application work every where.

2.4 URBAN-CYCLING APPLICATIONS/PLATFORMS (NON-PROFIT)

In this section I will talk a little about the existing applications/platforms that are research tools that aims to promote sustainable mobility.

As one would expect there are already some approaches taken on this topic, one of them can be found by the **CIVITAS** initiative, this is a programme co-financed by the EU, some tools incorporating this initiative are ***"Bike Citizens App"***, ***"BIKLIO"***, ***"ByCycling"***, ***"TRACE-Walking and Cycling Tracking Services"***, and ***"Bicycle Citizens"***, I will not go into too much detail in each of these applications, I will only explain that both have the function of following the routes made by cyclists and the collection of information about shops in the city and parking places in it, ie that these applications collect data that can be used by the

city itself, these applications take a position on the side of the user, in this case the cyclist [16].

An example not related to the theme of cycling directly, but to the encouragement of sustainable mobility, has been developed by **CEiiA**, which consists of a change-of-game platform, which aims at reducing CO_2 emissions, this is obtained through user rewards in the form of **AYR** credits (where AYR is the name of the platform), and can exchange them for services provided in other applications [16]. Such initiatives aim at a reduction of car use in a city and encourage people to think more about sustainable mobility.

Applications of the kind exposed above could be merged into a single application, i.e. an application that would focus services, so that the user, in this case the cyclist, would be better served.

Another example of an application is the *"Minha Freguesia"* of BSB, this application allows us to report problems to our nearest location, check public transport schedules, and is linked to services provided by local entities, such as municipal bureaucracies [16].

Having this in mind, I state again that it is of great importance to create an application, where all these services are included, and by this quick search we can confirm that we have that failure in the market, where there is no application, with these integrated services.

This example of application are very important because it show as different types of concepts of platforms to research of explore the main concept of this dissertation.

2.5 Bicycle-related applications/platforms

Some existing applications can enter this section, some of these applications have been references above those that were founded by the **CIVITAS** initiative, but these applications have been financed by a public entity, i.e. that their main objective is not to make money, but as a research tool.

Next, I will briefly show and explain some applications that are on the market, and that have as their business model the sale of them, and their advertising, being applications that return funds to their creators.

We can soon talk about two applications/platforms that are the most used either in Apple or Play Store, these are the **Endomondo** and **Strava** application. These are applications that have as functionality to map a certain route that the user performs live, like all the data that this brings. It is also possible to analyse users' physical activity, namely calories burned, average and maximum speed, distance covered, and heart rate through the use of items such as smartwatch. Another known application is **Garmin Connect**, this one has the same features as the ones above, however it needs a specific Garmin device to analyse and register any type of data collected during the journey [16, 1].

Other applications not as complete as those mentioned above are for example, **Map My Ride**, which is similar to **Endomondo** and was created by the same company, but it does not have as many features. Applications like **Komoot** allow us to create routes and plan routes that we can take before going cycling, always providing the user with information related to touristic points, as well as its duration, can be compared to Google Maps.

A different application from all this has the name **Biklio**, this is an application that recognises if bike users are making the city a better place, if that is the case, the same reward them with benefits in establishments the local services. This is an application that is already available in 11 European cities, 4 of which are Portuguese. The application is only available in a few cities because they have to really embrace the concept so that "spots" can promote their offers and discounts which are available to cyclists using the application [25].

We can then conclude that these applications focus on urban cycling as a sport and do not provide opportunities or features that could attract more users, in this case users who want to cycle without the need to do so in a sport.

With the application that were explain above we have a new way of looking at a this, because this type of mobile application are making the urban cycling has a sport, and by looking into this application we can see differences features or qualities that we can improve when development our mobile application.

2.6 similar applications/platforms

In this section I will show some applications and prototypes similar to the work done in this dissertation, I will explain in what each one consists as well as its functionalities and purposes.

2.6.1 *Application Prototype - Minha Freguesia [16]*

This prototype application was created as a research tool, to be used in the city of Braga in Portugal. It has several functionalities among these are: The tracking of each travel, the connection with social networks, data related to each travel, such as the duration of the travel, the average speed, the calories burned, the amount of CO_2 that was avoided, the distance of the travel and the route itself, these are the main functionalities. It should be noted that it is intended to connect the city in which it is inserted, ie we have the possibility to consult the available transport and compare them with each other, we also have the possibility to access news and make surveys in the chosen location well with reporting occurrences that cyclists may find during their journey. This makes this application have a similarity to the theme that is being exposed in this dissertation, because it actively aims to encourage sustainable mobility, through the use of the bicycle, and in addition has the possibility of connecting to

services of the city in which it is inserted, making it an application that can connect with the city, focusing on the theme of smart city. You can see the figure of the app in appendix A.1.

2.6.2 *Application Prototype - SMART app [1]*

This application has been developed jointly by Mobidot and the municipality of Enschede, it gives the opportunity to monitor the cyclists who use it, making it a kind of living laboratory, it was also created for research purposes. It has the ability to track travel behaviour data to provide detailed travel data, it also has the ability to provide traffic data to cyclists. It is based on the incentive business model, investigating whether they have any effect on cyclists. We have the following features: the first is the travel information, this can be real traffic information where users are warned in case of works or events on the way, the second is based on this traffic information the application can provide alternative routes in order to help the user with their travel plans, the third can be considered as feedback by the application of the travel history pattern, this feedback serves to show the user whether to change their behaviour or not, and it is up to the user to decide whether to change, the fourth enters the field of challenges and incentives that are given to the user or to what he commits himself. This is an application that encourages sustainable mobility within a city. You can see the figure of the app in appendix A.2.

2.6.3 *Application - GreenBikeNet [26]*

This application is designed to provide services to cyclists so that they can improve their cycling experience in smart cities. The application uses smartphones and ZigBee low power technology for *ad-hoc networks*. It is embedded in mobility research applications in smart cities. Its purpose is to connect the cycling community, making them connect to each other and visualize each other, that is, that two cyclists who are connected to the same network will be able to see the location of both. This application is good for those who like to do shared cycling, making group travels for greater safety. Cyclists can talk to each other, even if they have different languages, because the application is capable of translating into any language, for this purpose the cyclist must wear headphones. The application also provides real location data and speeds for all cyclists who are connected to the same network. An important feature of this application is the ability to provide an emergency system if necessary, thus protecting cyclists and ensuring their safety, in addition all their commands are done vocally. This application ensures the protection of the cyclist using the group as a means to this end. You can see the figure of the app in appendix A.3.

2.6.4 *Application - BeCity [27]*

Since the data provided by urban cyclists is very useful, and these are collected by *smartphone* sensors, these sensors can enable the next generation of collective knowledge to improve the quality of life and cycling mobility. This application even allows this, which consists in allowing low competition cyclists, i.e. cyclists who use the bicycle for their daily life, to share their routes and their comments, this makes this application a kind of distributed data collection system. It also has the ability to recommend courses, considering several factors among these are: (1) distance, (2) bicycle routes, and (3) attractiveness of these routes. You can see the figure of the app in appendix A.4.

With this different type but similar application are were explain above we can narrow it down the similarity between our work and the work that has been already done in the community, and by doing this we can make a better application, fixing some problems or issues that previous work have to face. Another important aspect is to see the different types of approach that this mobile application choose to approach and how to solve their different problems.

3

METHODOLOGIES AND TECHNOLOGIES

In this chapter the following concepts will be discussed: (1) it will be exposed and briefly explained what research methodologies consist of and a brief example of a methodology within that field, (2) two software development methodologies will be shown, (3) the methodology that I will follow will be briefly explained and finally (4) the technologies on which this dissertation will be based will be exposed as well as a brief explanation of them.

3.1 RESEARCH METHODOLOGIES

Research methodologies are responsible for assisting research projects, as is the intention of this dissertation. These are a set of organised and systematic research techniques, that is, a simple guide to research and how it will be conducted throughout its life cycle [28]. These methodologies describe methods of analysis, and focus on the limitations and resources it may have, in ways that clarify its assumptions and consequences, thus making visible the frontier of knowledge we have about research.

3.1.1 *Design Science Research*

To ensure that this dissertation or any research work, is recognised as sound and relevant, both in the academic field and by society, it must show that it has been rigorously developed and is debatable and verifiable. It is based on the above that it is imperative and indispensable to acquire a robust research method for the successful conduct of a study [29]. It is in this context that the Design Science Research (DSR) research methodology comes in, it aims that to highlight the reliability in the research results, it is necessary a set of care and rigorous procedures that minimize the bias in the results obtained.

The DSR methodology is composed of 4 distinct phases [30], each phase has it's on process and steps. These phases and steps are presented below, as well as a brief explanation of the purpose of each one of them (you can see on figure 4):

1. **Problem identification** - In the first phase of the research process, a problem is identified. It has to be ensured that the problem has practical relevance or might be of relevance once solved;
    - Identify problem: A problem has to be identified. It will be refined in the course of the current phase, to assure its relevance and understanding;
    - Literature research – Part I: To identify a problem, literature research can be used. Unsolved problems could be mentioned in scientific publications as well as in practitioner reports;
    - Expert interviews: Interviews with practitioners and experts in the field can be conducted to identify relevant and addressed problems;
    - Pre-evaluation relevance: Once a suitable problem is identified, a pre-evaluation on the relevance has to be conducted. This includes creating a general research hypothesis in the form of a utility theory.
2. **Solution design** - In the second phase, the solution is designed. It is divided into the steps *"artefact design"* and supporting *"literature research"*. After identifying a problem and pre-evaluating its relevance, a solution has to be developed in the form of an artefact;
    - Design artefact: Artefact design is a creative engineering process;
    - Literature research – Part II: Like during the first phase, the existing knowledge base including state-of-the-art has to be taken into account to ensure research rigour.
3. **Evaluation** - Once the solution reaches a sufficient state, its evaluation can be started. It is possible to iterate back to *"design artefact"* or even *"identify problem"* if necessary;
    - Refine hypothesis: Usually, the general research hypothesis is difficult to evaluate as a whole;
    - Case study/action research: Case studies and/or action research should be performed on the general hypothesis or at least on an important refined hypothesis;
    - Laboratory experiment: Refined hypotheses are evaluated using laboratory experiments or if possible field experiments.
4. **Summarise results** - At the end of the research process, results are summarised and published.

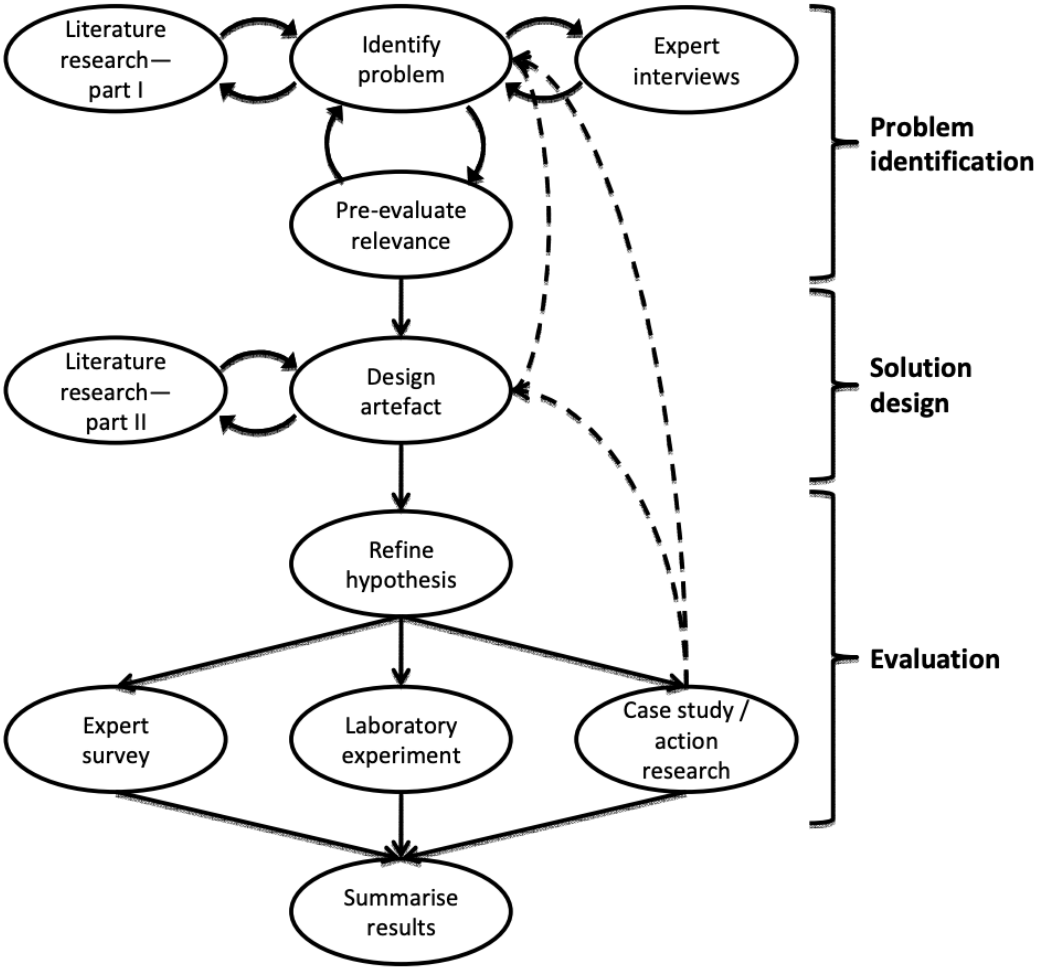


Figure 4: Design Science Research methodology
(Source: [30])

3.2 Software Development Methodologies

A software development methodology is a way to manage a software development project [31], no methodology is better than the other, both have their positive and negative points, the use of them also depends on the project itself, the company and who is using it.

3.2.1 *Waterfall*

This methodology is the original one in software development, it approaches this concept in several steps where one depends on the previous one, and to pass to the next step it is necessary that the previous one is 100% completed [31]. This method is very linear, as once

one stage is completed, you move on to the next. This gives us an advantage as the project is well developed and planned, as each stage is carried out in a methodical manner, thus avoiding setbacks that may increase the cost of development. These projects tend to have a good basis and are well documented projects. However, there is a disadvantage that is the changing requirements or characteristics in the middle of development, this in the software world often happens because some problems are discovered in the middle of development [31].

This methodology is more suitable for more complex and large projects, and which have several teams working on the same project [32], it predicts many failures that are caused by several teams working on the same project.

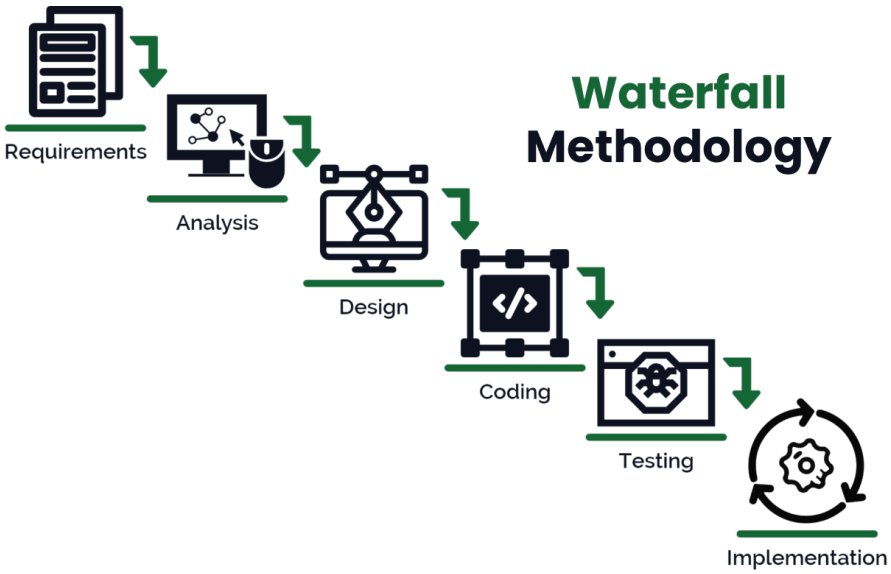


Figure 5: Waterfall methodology
(Source: [33])

3.2.2 *Agile*

This methodology is intended to correct a waterfall failure that is the changes in characteristics throughout development, thus making software development more agile and at the same time providing quality software while minimising development costs. The projects that use this methodology are considered as projects that will be built along the development process, with new requirements to be added during the process. This methodology is used more by small teams, as it requires daily interaction and among the team that will discuss issues about the project. An advantage of this methodology is the great importance that the tests constitute in the development of the project [31]. However a disadvantage is that this methodology does not work very well on large projects or large numbers of teams, as

they last for years to develop, and they already have a rigorous documentation and a well defined design from the beginning.

The agile methodology creates a means in which focuses on a rapid value change and response [32], thus improving the development of small projects, is also bringing a new thing that is the active interaction with the client who is part of the team, this always brings more knowledge to the team, because it interacts with the client.

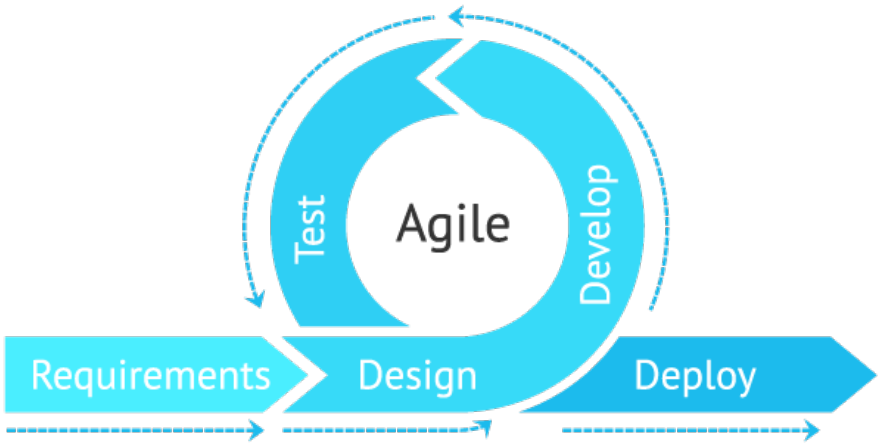


Figure 6: Agile methodology
(Source: [34])

3.3 METHODOLOGY USED

In this section I will give a brief explanation of the methodology that was used, we must think that the work of this dissertation has two components, and the first component of research, this being an important part of which will leave a concrete idea of what is intended with this research, as this phase is of great importance opted to use the methodology explained above 3.1.1, denoted as DSR, this methodology meets what is intended with this dissertation in its initial phase.

In the second component would already be a part of development, since this I will have to opt for a software development methodology, as only I will be working in this dissertation, and for greater speed and solidity in the development process of the prototype application, I will opt for the methodology of **Agile** software development, 3.2.2, because it will go in favour of what is intended in this component.

3.4 TECHNOLOGIES

The choice of technologies for the development of this application is of great importance, because they will support the final solution that in this case will be the prototype of a mobile

application. For this purpose in this section will be exposed and explained the type of *database* that will be used, as well as the architecture of the application itself and its different components, such as the *Back-End* and *Front-End*.

3.4.1 *Databases*

Databases in the software environment are responsible for storing information. They are useful in the context of software development because they make available and keep up to date all the information. There are several types and forms of database, among these we can highlight relational databases (SQL), and non-relational databases (NoSQL).

- **Relational databases** - In this type of database, better known as Structured Query Language (SQL) databases, is used to store data in the format of tables that are related to each other through primary and foreign keys. These tables are composed of columns, these columns define the content of the respective table, the respective information that belongs to that table is defined by rows, which correspond to each existing record within that table [35], these databases are known for their properties of atomicity, consistency, isolation or durability, these properties have the acronym **ACID**;
- **Non-relational databases** - When it comes to non-relational databases better known as Not Structured Query Language (NoSQL), these have dynamic *schemas* and different types of data, being possible to add or remove data with great ease. This type of database is composed of collections, within these collections we have the data itself, and these data can still contain other collections within them, here we have a big difference to SQL databases because these NoSQL databases are not normalized [35], this leads to faster data access, however it becomes more difficult to ensure a correct structure of the same database and the data contained therein, to combat this deficiency, you can always check if the data we want to save are in accordance with the rules that we want to define.

Taking into account what was said above and the subject of this dissertation, it was decided to choose the **non-relational database**, this choice was made because it will only be necessary to save data related to the cyclist, more specifically personal data and data of his location, tracking and travel of the same, thus being this choice to improve, because it allows easy access to the stored data, as well as a junction of all data for a better view of them.

MongoDB Atlas

Within the world of non-relational databases, we chose to use the most common which is *MongoDB*, this type of database allows us a complete management of all data. As *MongoDB*

is a tool already for some time to be used in the world of software development we chose to have a database online, and not local, so we can manage it from anywhere, and have easy access to it, for this purpose we used the inline tool MongoDB Atlas [36], is a tool type cloud and cluster that allows us to always have the database online and available.

3.4.2 *Back-End*

The term ***Back-End***, can be translated in other words to *"the part that is hidden the application"*, it has a great responsibility, because this is the function of putting the core of the application in operation. We can call the term *Back-End* as the architecture module that the application will need. This module will also have micro-services that will support the services that the application will provide.

Among the existing technologies in the *Back-End* area, the one that stands out the most for its stability, efficiency, security and performance is the ***Node.js framework in JavaScript*** [37], as it allows to have a development environment based on *JavaScript* on the server side. Over the years, as large applications require robust servers, these technologies have become more popular, as well as the *JavaScript* programming language [38], which is why this language has risen above the others. The decision was made to use *Node.js* as a *Back-End* technology because it also allows the addition of packages that are needed during the development of the software.

Using this *framework*, it is possible to create a ***RESTful API***, which allows communication between the server and the client. For that, methods such as GET, POST, PUT, DELETE will be used, and it is through these methods that the communication between the mobile application and the *RESTful API endpoints* takes place, in order to perform all necessary operations. You can see in figure 7 an general architecture.

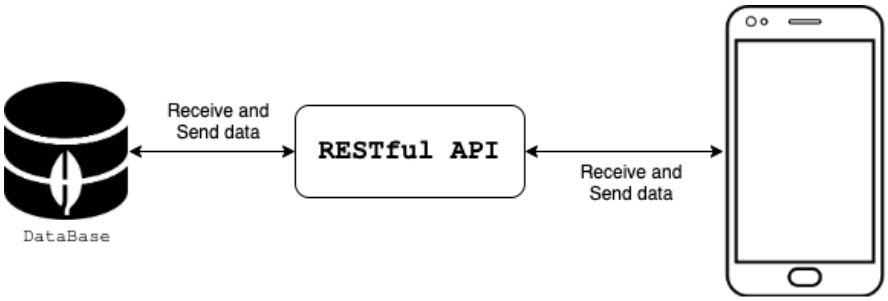


Figure 7: General Architecture
(Source: own)

3.4.3 *Front-End*

As mentioned above we have defined a module that deals with data processing and management, now we only need a module that shows them in a simple and direct way to the end user, this is where the term ***Front-End*** comes in, this has as a definition of the *"known part of the application"*, i.e. that it is the layer of the application that is visible to users. Its purpose is to provide an interaction layer with the application, making this interaction easy and *"user friendly"* so that the end user can access the application from anywhere and consult the data it contains.

In this project the *Front-End* module will be developed and based on a mobile application, so that you can use it to install it on your *smartphone* and use it anywhere and on any platform, making this a versatile software platform capable of working in any environment and thus making it possible to assist in the resolution of this dissertation.

For this purpose as it is a mobile application it must be able to work on the two major operating systems that we currently have available on the market, these are *iOS* [39] and *Android* [40], given this we need a technological tool that allows us to do this. Among the technological tools in this field are *Flutter*, *React Native*, *Ionic* [41], these are *hybrid frameworks*. Taking into account already developed projects using ***React Native***, makes this more accessible for the development of the prototype for mobile application, being this the chosen for this process.

4

ANALYSIS AND PROPOSED APPROACH

The documentation and specifications are aspects of great importance in the development of software [42]. The analysis of requirements to survey them is the first step to take in software development, as defined and said in the *Agile* methodology 3.2.2. It is through the analysis of the same that it is possible to document the needs of the project in question and the constraints of the same, thus being able to formulate a good software documentation for future software users and that should be considered throughout the process of the same.

In this chapter the first section is presented the requirements and all that they encompass, from its survey to its formulation, then based on the survey of requirements will be made a software modeling, showing the diagram of use-cases and classes. Finally, and taking into account all the sections defined above, it is shown proposed approach for solving the development part.

4.1 REQUIREMENTS

"Basic and necessary condition to obtain something or to achieve a certain purpose" [43], this can be a possible definition of requirement among the dozens that exist. The requirements are of great importance all over the world, but in this case more specifically in software development, these can be classified by functional requirements and non-functional requirements.

As mentioned above there are two types of requirements, we have the functional and non-functional, the first aim what the software to be developed can provide to the user, ie, its features and information, these are independent of any design and aspects of software implementation. The non-functional requirements are related to the quality, performance, usability, reliability and robustness of the software to be developed [44].

To acquire these requirements several techniques are used, among them we highlight the interviews, it is due to this that we can obtain a large part of the requirements that the software to be developed will have, we can also use other techniques such as introspection, and an analysis of the domain of the problem in question [45].

In order to create the requirements, it is necessary to pay attention to how they will be displayed and classified. For this purpose we used the **Volere model**, which is one of

the most respected to make this preparation, is that it already has its own table for the description of requirements, and this table is denoted and *requirement shell* [46], below you will find my adaptation of this table and a brief explanation of it (table 1).

Table 1: Requirement shell adaptation
(Source: Own)

Requirement No.:	**Requirement Type:**
Event/Use Case:	
Description:	
Rationale:	
Fit Criterion:	
Customer satisfaction rating:	**Customer dissatisfaction rating:**
History:	**Source:**

Below is a brief explanation of each individual component of this table.

- **Requirement No.** – The number of that particular requirement;
- **Requirement Type** – The type of that requirement;
- **Event/Use Case** – The event of use case belong of the requirement;
- **Description** – Clear and explicit description of the requirement;
- **Rationale** – Justification of the existence of the requirement;
- **Fit Criterion** – Criterion that fits the requirement;
- **Customer satisfaction rating** – Requirement importance criterion;
- **Customer dissatisfaction rating** – Requirement not importance criterion;
- **History** – When the requirement was made;
- **Source** – How create the requirement.

4.1.1 *Techniques Used for Requirements Survey*

Different techniques were used to extract and formulate the requirements, I will then explain each of them and what information was obtained from them. These techniques are of great help for a good formulation of requirements [45].

Interviews

This technique is the most common to be used in the world of Requirements Engineering, because it is thanks to it that we have information regarding the user who will use the application and what are his ideas and thoughts about what an application of this kind

should be like, as an aid in this process I will rely on a scientific article that contains the information I need to proceed [47].

This scientific article aims to achieve a new space of opportunities for digital tools and applications designed for urban cycling, trying to discover current practices associated with existing tools and the discovery of new information, even that which cyclists cannot fully express, or realise they need [47].

In order to explore this topic, the authors of this scientific paper conducted two focus group sessions, and ten interviews with cyclists, containing the same information needed and precise to formulate the requirements that are necessary to develop the prototype of the mobile application. I will not go into much detail about the sessions or the interviews, I will briefly explain what they consisted of and what information was taken from them.

- Focus Groups: A total of 10 participants (all men) participated in the group sessions, their age is between 24 and 59 years, these participants were divided into two groups of 5 elements. The information that was extracted from these focus groups was as follows: 2 participants were cyclists and used the bicycle for travel, 4 participants use the bicycle only for leisure and the remaining four use it for both contexts. For most participants cycling travels last less than an hour [47]. Important things were collected during the focus groups, the participants, in this case the cyclists all converge to various common needs, among this are: (1) the application would be a navigation system with tracking, (2) it could also have social characteristics, where they could share and gather information and (3) the application could integrate services within the city.

- Interviews: A total of 10 participants (7 men and 3 women) aged between 23 and 53 were used for the interviews. The information that was taken from these interviews was the following: all participants except one cyclist use the bicycle to travel, and for those who travel is generally short (less than one hour). These interviews were semi-structured and focused on several topics: (1) the use of digital technologies for cycling, and (2) the ideal mobile application for urban cyclists [47].

After aggregation of the content analysis of both the focus groups and interviews, the following main final themes emerged: (1) Current practices in digital tools, wearables, and sensors, (2) Technological difficulties and needs, and (3) Useful features of a mo- bile application [47].

Introspection

Although this is the most basic technique, it was necessary to use this identification of secondary requirements which would be easily identifiable on the basis of my technological

fields and the use of already existing applications in this area. Examples are the registration and permissions requirements to store and collect the information.

Domain Analysis

I also carried out a domain analysis but without much success, due to the idea being innovative and not having any documentation of similar systems. Anyway, to better understand the domain of the system and all the frontiers of the application prototype, I decided to study the scope of the system and together we made a domain model of our problem presented in the appendix B.1.

4.1.2 *Functional Requirements*

Table 2: Functional Requirement #1

Requirement No.: 1	**Requirement Type:** Functional
Event/Use Case: Application Operations.	
Description: The cyclist can register in the application.	
Rationale: As the main user of the mobile application, the cyclist should be able to register in order to obtain all the information associated with it and be identified.	
Fit Criterion: The user will be able to register in the application.	
Customer satisfaction rating: 5	**Customer dissatisfaction rating:** 5
History: 15/02/2021	**Source:** Introspection

Table 3: Functional Requirement #2

Requirement No.: 2	**Requirement Type:** Functional
Event/Use Case: Application Operations.	
Description: The cyclist can login in the application.	
Rationale: As the main user of the mobile application, the cyclist should be able to login in order to obtain all the information associated with, and access to the application.	
Fit Criterion: The user will be able to login with valid credentials.	
Customer satisfaction rating: 5	**Customer dissatisfaction rating:** 5
History: 15/02/2021	**Source:** Introspection

Table 4: Functional Requirement #3

Requirement No.: 3	**Requirement Type:** Functional
Event/Use Case: Navigation operations.	
Description: Inform their real-time position.	
Rationale: The cyclist should be able to see on a map their real-time position, so that they can see where they are.	
Fit Criterion: The cyclist could be able to see his location on a map.	
Customer satisfaction rating: 5	**Customer dissatisfaction rating:** 5
History: 15/02/2021	**Source:** [47]

Table 5: Functional Requirement #4

Requirement No.: 4	**Requirement Type:** Functional
Event/Use Case: Navigation operations.	
Description: Plan and choose the route.	
Rationale: The cyclist should be able to choose and plan the route he wants to follow in a certain navigation that he can make.	
Fit Criterion: The cyclist could be able plan routes.	
Customer satisfaction rating: 3	**Customer dissatisfaction rating:** 3
History: 15/02/2021	**Source:** [47]

Table 6: Functional Requirement #5

Requirement No.: 5	**Requirement Type:** Functional
Event/Use Case: Navigation operations.	
Description: Receive alerts of dangerous situations.	
Rationale: The cyclist should be able to receive updates and alerts of the dangerous on the route.	
Fit Criterion: The cyclist could be able to receive alerts.	
Customer satisfaction rating: 3	**Customer dissatisfaction rating:** 1
History: 15/02/2021	**Source:** [47]

Table 7: Functional Requirement #6

Requirement No.: 6	**Requirement Type:** Functional
Event/Use Case: Navigation operations.	
Description: Cyclist can evaluate route.	
Rationale: Cyclist may be able when finishing a route to evaluate it, this evaluation is made to characterise the route.	
Fit Criterion: Cyclists on finish a register can evaluate that.	
Customer satisfaction rating: 4	**Customer dissatisfaction rating:** 3
History: 15/02/2021	**Source:** Introspection

Table 8: Functional Requirement #7

Requirement No.: 7	**Requirement Type:** Functional
Event/Use Case: Navigation operations.	
Description: Cyclist can upload/edit/delete route.	
Rationale: Cyclist may be able when finishing a route to upload it, edit, or delete the route.	
Fit Criterion: Cyclists can management their registers.	
Customer satisfaction rating: 4	**Customer dissatisfaction rating:** 3
History: 15/02/2021	**Source:** Introspection

Table 9: Functional Requirement #8

Requirement No.: 8	**Requirement Type:** Functional
Event/Use Case: Cyclist operations.	
Description: Cyclist can add/update your profile.	
Rationale: The cyclist should be able to update and add you personal profile on the application, so that the application can have data about that.	
Fit Criterion: Cyclists can management their profile.	
Customer satisfaction rating: 5	**Customer dissatisfaction rating:** 4
History: 15/02/2021	**Source:** Introspection

Table 10: Functional Requirement #9

Requirement No.: 9	**Requirement Type:** Functional
Event/Use Case: Cyclist operations.	
Description: Cyclist can add/update your profile cyclist.	
Rationale: The cyclist should be able to update and add you cyclist profile on the application, so that the application can have data about that.	
Fit Criterion: Cyclists can management their cyclist profile.	
Customer satisfaction rating: 4	**Customer dissatisfaction rating:** 2
History: 15/02/2021	**Source:** Introspection

Table 11: Functional Requirement #10

Requirement No.: 10	**Requirement Type:** Functional
Event/Use Case: Application operations.	
Description: Autonomous tracking.	
Rationale: Application may be able to detect when user is cycling, and trigger an automatic tracking, so that the cyclist don't need to start tracking it self.	
Fit Criterion: Application can begin autonomous tracking.	
Customer satisfaction rating: 2	**Customer dissatisfaction rating:** 1
History: 15/02/2021	**Source:** Introspection

Table 12: Functional Requirement #11

Requirement No.: 11	**Requirement Type:** Functional
Event/Use Case: Bicycle operations.	
Description: Tutorials on regular check-ups and minor repairs.	
Rationale: The cyclist will be able to access quick tutorials on good bicycle maintenance, as well as repairs and tips on the bike.	
Fit Criterion: The cyclist can view tutorials.	
Customer satisfaction rating: 2	**Customer dissatisfaction rating:** 1
History: 15/02/2021	**Source:** [47]

4.1.3 *Non-Functional Requirements*

Usability

Table 13: Non-Functional Requirement #12

Requirement No.: 12	**Requirement Type:** Non-Functional
Event/Use Case: Application operations.	
Description: User can change language of the application.	
Rationale: The application has to be able to have at least two languages available in case the user wants to change the language of the application.	
Fit Criterion: Multi-language feature.	
Customer satisfaction rating: 3	**Customer dissatisfaction rating:** 2
History: 15/02/2021	**Source:** Introspection

Table 14: Non-Functional Requirement #13

Requirement No.: 13	**Requirement Type:** Non-Functional
Event/Use Case: Application operations.	
Description: Application can work offline.	
Rationale: The application must be able to work without the need for an Internet connection.	
Fit Criterion: Application could be able to work on all conditions.	
Customer satisfaction rating: 3	**Customer dissatisfaction rating:** 2
History: 15/02/2021	**Source:** Introspection

Legal

Table 15: Non-Functional Requirement #14

Requirement No.: 14	**Requirement Type:** Non-Functional
Event/Use Case: None.	
Description: Personal information should be implemented to match with the new European Union data protection laws.	
Rationale: All data collected by the system should be from in accordance with EU data protection law, with a view to privacy and confidentiality of patients.	
Fit Criterion: Application could be able to protect personal data.	
Customer satisfaction rating: 4	**Customer dissatisfaction rating:** 3
History: 15/02/2021	**Source:** Introspection

Operational

Table 16: Non-Functional Requirement #15

Requirement No.: 15	**Requirement Type:** Non-Functional
Event/Use Case: Navigation operations.	
Description: Application store routes on memory.	
Rationale: The application must be able to store the routes made by the cyclist at least one hour after the end of the routes.	
Fit Criterion: Application could store registers on memory.	
Customer satisfaction rating: 4	**Customer dissatisfaction rating:** 3
History: 15/02/2021	**Source:** Introspection

Table 17: Non-Functional Requirement #16

Requirement No.: 16	**Requirement Type:** Non-Functional
Event/Use Case: Application operations.	
Description: The mobile application recovers the last session in case you have not logged out.	
Rationale: This means that you do not need to log in every time you open the application..	
Fit Criterion: Close the application with the login and restart it.	
Customer satisfaction rating: 4	**Customer dissatisfaction rating:** 3
History: 15/02/2021	**Source:** Introspection

4.2 SOFTWARE MODULATION

In this section I will show and explain, one of the most important components in software development, this component is denoted of diagrams Unified Modeling Language (UML), these consist of lists or diagrams of software specifications, as well as functionalities of the same and diagrams of execution and instructions of the same, those diagrams are based on the UML. They are based on a detailed analysis of the requirements that were defined and specified in the previous section 4.1.

To draw and to represent these diagrams, I will use the program *Visual Paradigm* [48], I proceeded to this choice for the experience that I possess in the same, the same is a tool that was created with this purpose, and for the simple use of the same, being possible to aggregate all the diagrams of the project in a single place being able to consult the same ones later of a simpler and organized form.

Software development diagrams are represented using the UML language as mentioned above. The software development includes is normally divided into stages, in which each of these stages involves different UML diagrams, each of these diagrams is dedicated to a different aspect of the project [49], this makes it possible to have a general design specification of the system. Ahead will be shown two UML diagrams, (1) use case diagram 4.2.1 and (2) the class diagram 4.2.2.

4.2.1 *Use Case Diagram*

The use cases diagrams are of great importance in the context of software development and in relation to UML, these tell and show an abstract story about the system behaviour, can also be used to specify communications between the various participants/actors of the system. Through a diagram with all the possible use cases it is possible for us to show relationships, or possible scenarios that could happen in software development [50].

Due to the use of these diagrams it is easier the communication between the development team and the customers, because they are easy to understand, even for people who are not used to this kind of technologies. This leads to easier software development, which leads to a reduction in the time of granting the software.

First a general use case diagram was formulated, this was prepared so that it is possible to identify the actors that will be part of the system, as well as the respective interaction groups.

In a general way, we can verify in the figure 8, in which we can indicate that the software that will be developed will contain two users, the **Cyclist** and the **Administrator**, being that the main one will be the Cyclist. Next we can ascertain that we have 5 groups being these: (1) Application Operations, (2) Navigation Operations, (3) Cyclist Operations, (4) Bicycle

Operations and finally (5) Manage Users. The first 4 groups belong to the cyclist who will have interaction with the system, and the last group (5) will only be the interaction with the system administrator. We can see these iterations in the figure 8.

In the figures 9, 10, 11, 12, 13, the use case diagrams of each of the groups mentioned above will be presented in more detail, as well as the set of use cases belonging to them. In these diagrams it will be possible to analyse and obtain a general information of the functionalities that the software will make available for each of the users.

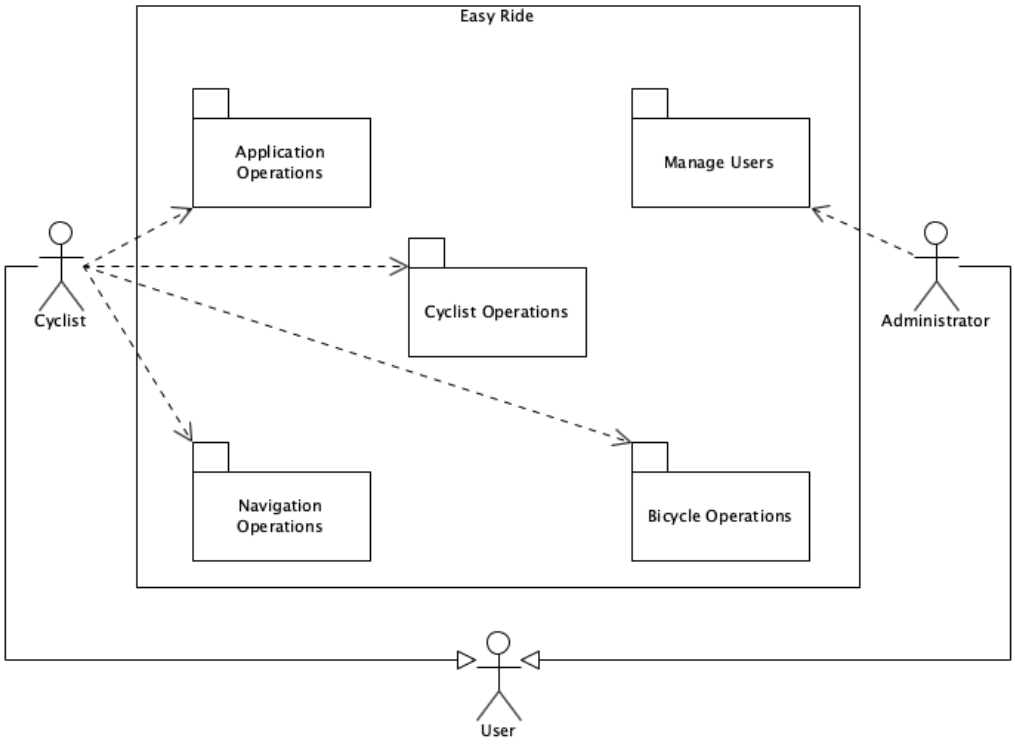


Figure 8: Overall use case diagram
(Source: Own)

Regarding the group of operations of the application seen in figure 9, the user in this case the cyclist, should be able to register in the application, this allows the same to save their data in the application, then the same can log into it allowing the same enter the application and access and use it, allowing the cyclist to interact with it. For a better convenience in the use of the application, the same can also change the language of the same, for now these languages will be Portuguese and English. And finally the same can turn on or off the autonomous tracking if you want, this will allow the application to detect if the cyclist is using the bike and start tracking it. These operations concern the use cases diagram in relation to the application itself.

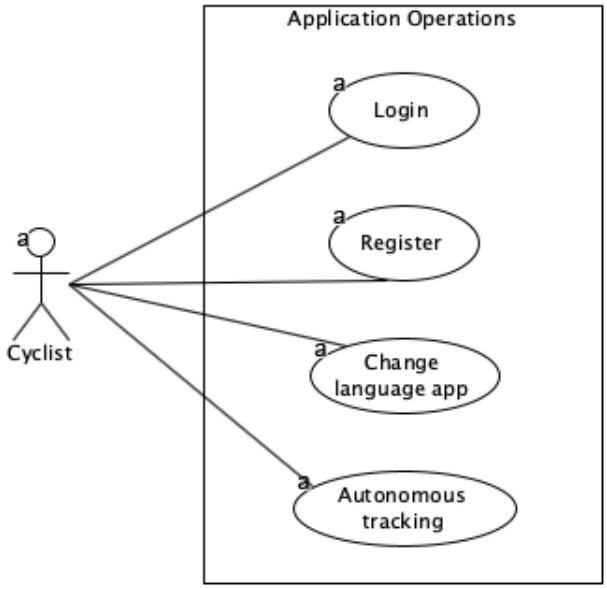


Figure 9: Application operations use case diagram
(Source: Own)

An important group are the navigation operations, you can see the use cases diagram in figure 10. With the help of this diagram we can then verify that the cyclist as the main user will be able to choose a route that he wants to follow, this helps in his navigation when using the application, after that he will be able to have control over his records, that is he will be able to delete them, edit them, or save them in the database. After the conclusion of a route or of a travel, he/she will be able to evaluate that route, to save particular information about that route, he/she will also be able to start a trace if he/she wants and be able to visualize it, finally he/she will be able to consult his/her real position on a map, this way he/she will be able to find out his/her location in real time.

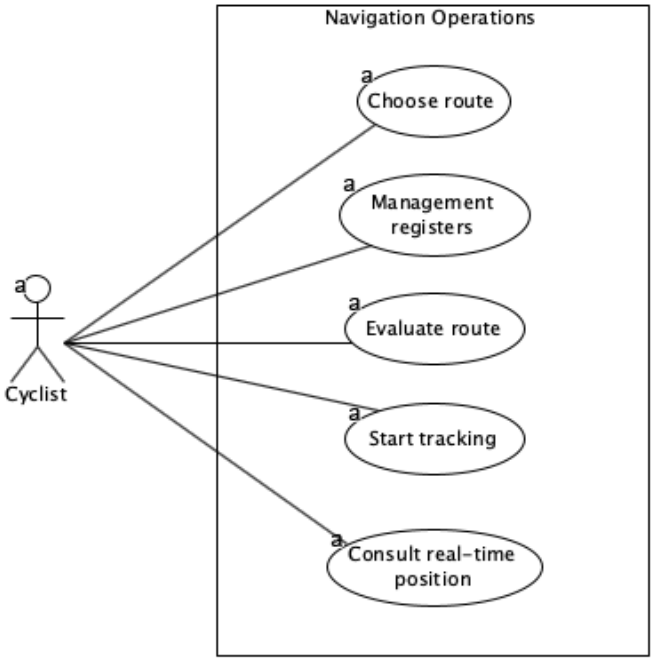


Figure 10: Navigation operations use case diagram
(Source: Own)

Regarding the cyclist operations group, that you can see in figure 11, the same can have a control in two profiles, the personal profile and the cyclist profile. This allows you to add the two profiles so that we have a more complete information of the same, and finally the same will be able to consult data relative to the same.

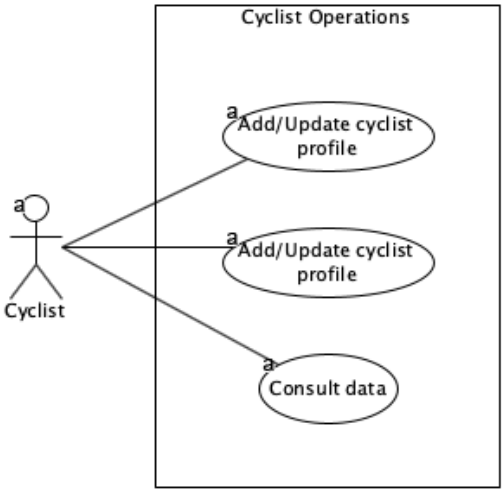


Figure 11: Cyclist operations use case diagram
(Source: Own)

The bicycle operations group seen in figure 12, allows in this case the cyclist to be able to access maintenance tutorials, or other kinds of tutorials about his bicycle, and finally it may be possible to connect to the bicycle itself.

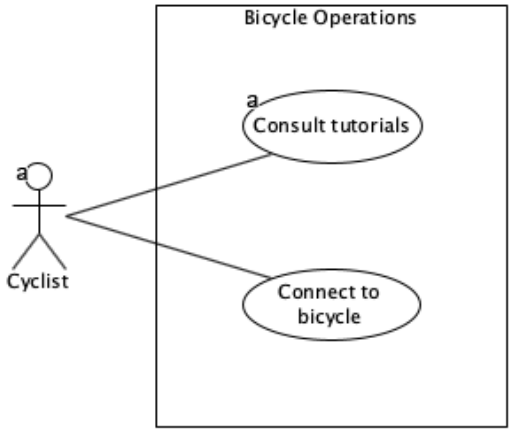


Figure 12: Bicycle operations use case diagram
(Source: Own)

Finally we have the manage users group, as you can see on figure 13, this one is more related to the control of the application, by an administrator, where he can have control over the system, the users and the registers that exist in it.

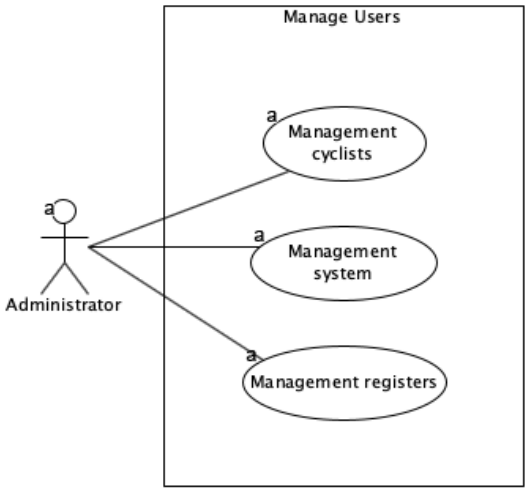


Figure 13: Manage users use case diagram
(Source: Own)

4.2.2 *Class Diagram*

In this section I will show and explain the class diagram that was created and on which the software development will be based. Being the class diagram the most common to be used in UML diagrams, it is of great importance to have the same in this software development process.

The class diagram is used to model a static design of a system view, this same can also be seen as a kind of primary artefact before the object oriented software development [51]. We can then think of the class diagram as a central component for the representation of a solution domain of a given platform and that serves as a basis for the generation of specific details that will be necessary for the generation of the software code.

A class diagram describes the static structure of the classes in the system and the relationships between these entities. Thus, a class diagram represents the structure of the system, since the summarised essence, the basis for the operation of the system, but not the the very dynamics of the system. Here being important to highlight the key word **"class"**.

Figure 14: Class diagram overall
(Source: Own)

You can check through figure 14 the class diagram in a more general way, I decided to be like that because the application development will focus on two different aspects that I decided to separate by packages, the first will be the Front-End package and the following will be the Back-End package those will be explain and expose in figures 15 and 16. I decided to make this choice for being simpler to explain each of these components. Focusing now on the main class called Easy Ride, this class will have 4 attributes associated with it with their respective *getters* and *setters*, these will be (1) user, (2) token, (3) travels and (4) language, these will serve for a good functioning of the software, and finally have the initial constructor of the Easy Ride class itself.

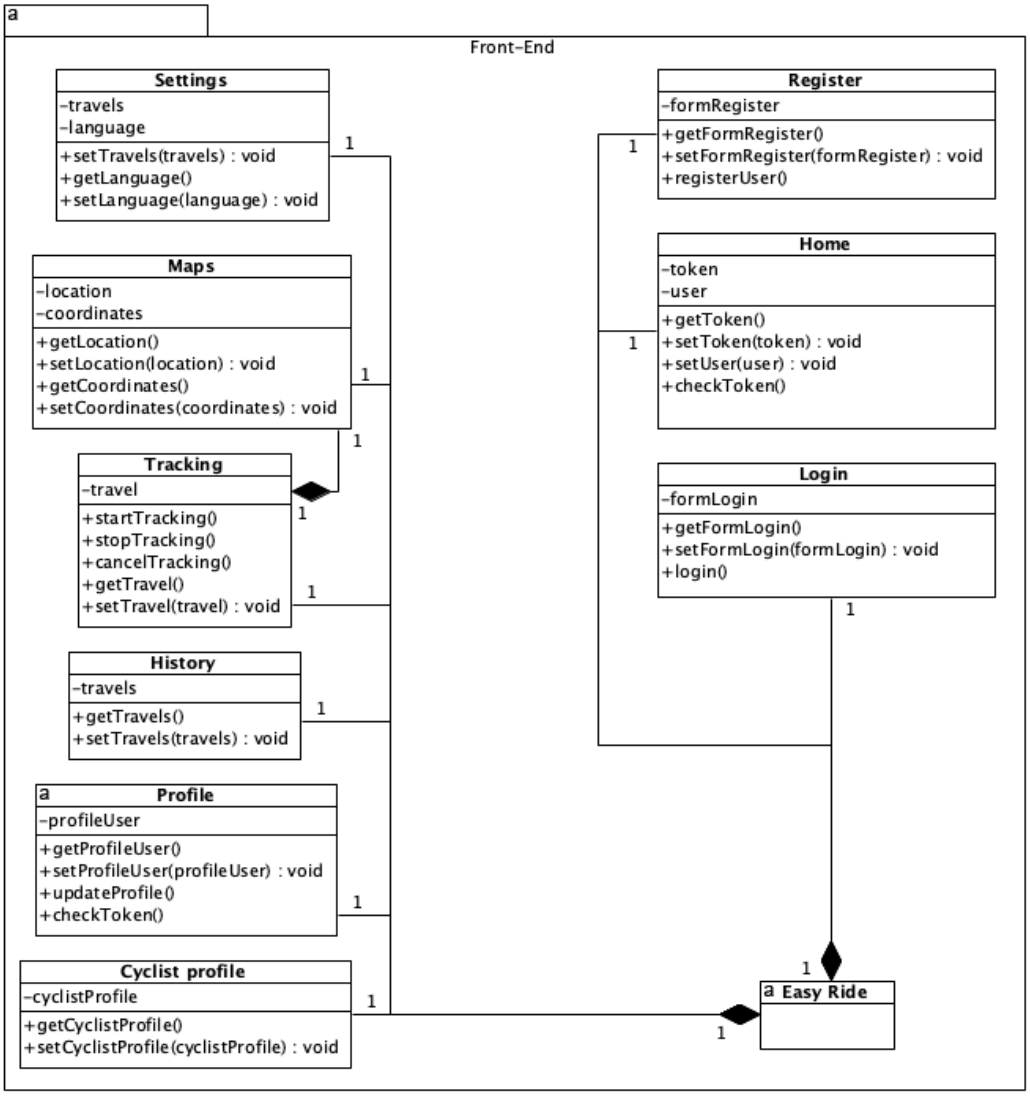


Figure 15: Front-End class diagram
(Source: Own)

As you can see in figure 15, shows the class diagram corresponding to the Front-End package, I will give a brief explanation of it.

This can be separated in two sections, the first being the authentication section in the application, this being composed by the classes: (1) *Register*, which is responsible for allowing the user registration, (2) *Login*, which is responsible for the user access to the application and finally (3) *Home*, which will be the first class to be loaded and shown if the case to the user. The second component will be the main part of the application, this will be composed by: (1) *Settings*, responsible for configuring the application as the language, (2) *Maps*, this will be responsible for all the management when it comes to navigation, (3) *Tracking*, will be the class responsible for the travel tracking, (4) *History*, class responsible for showing the

user the travels that the same made, (5) *Profile*, will be responsible for showing the user his personal profile, (6) *Cyclist Profile* will be responsible for the cyclist's profile.

These are the main classes that will be defined in the Front-End package.

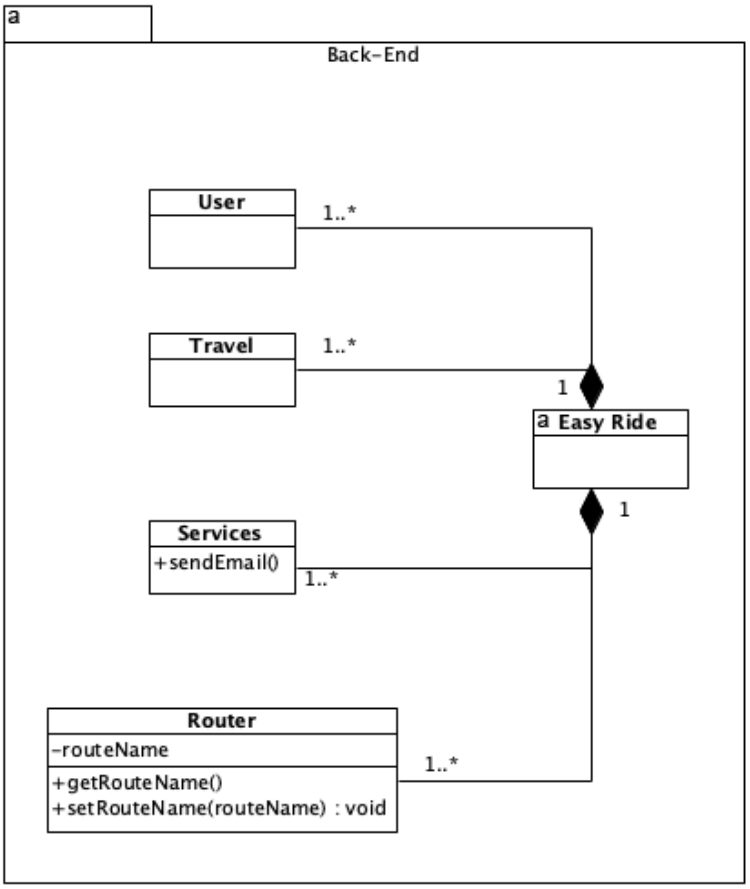


Figure 16: Back-End class diagram
(Source: Own)

By the figure 16, we can see the Back-End package, I will not go into detail about it, because it will be better explained in section 5.2, it's just to point out that it will have 4 important classes, the first two which are respectively *User* and *Travel*, will be the classes associated with the application records. Then we have the *Services* and *Router* classes that will be focused on providing services and routes to our package respectively.

4.3 PROPOSED APPROACH

In order to respond to what was said and explained in section 4.1, and section 4.2, which are respectively the requirements and the modulation of the software and also taking into account the database that will be used and defined, it is necessary to plan and propose an approach, mediating the criteria that were defined above.

We can start with the Back-End, which will be developed using the *Node.js framework*, it is crucial to understand how it performs the communication between the database and the Front-End, as well as its final structure.

In what concerns the database connection, is as it is a database *"Document-oriented storage"*, it will have a synchronous connection and used queries in *JavaScript Object Notation (JSON) structure* to extract and modify data from it.

The communication between the two main modules, the Back-End and the Front-End, will be done through a *RESTful API*, which will use CRUD operations (*create, read, update, delete*), allowing a dynamic access to the data, since this application involves the manipulation of many data and not only static data, making it a dynamic application.

So we can say that there is a separation between the server and the client, which is possible the evolution of them independently, you can see in figure 17, the final architecture of the application.

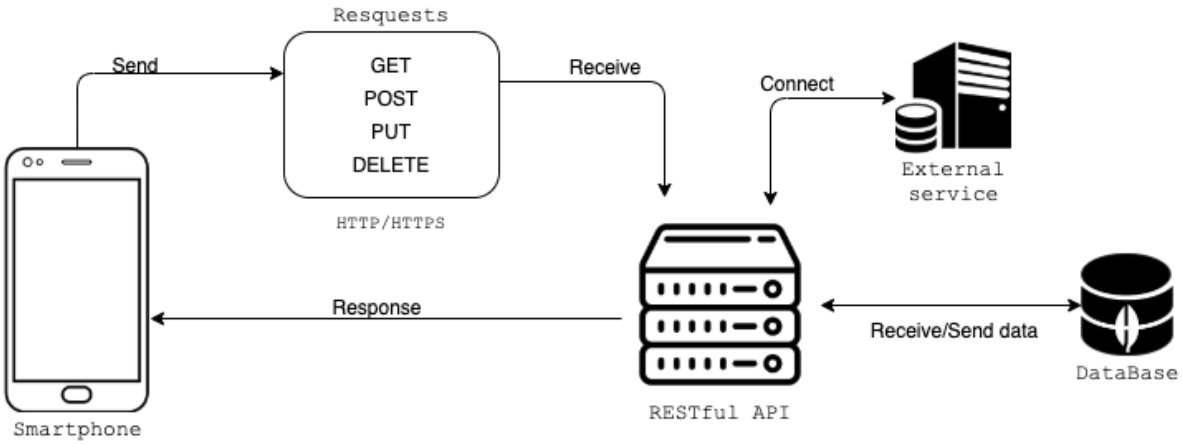


Figure 17: Architecture
(Source: Own)

5

DEVELOPMENT OF THE PROTOTYPE

In this chapter I will explain the development process of the mobile application prototype, this will be done taking into account the proposed approach that was defined in the section 4.3.

In the next's sections all the different components and parts of the prototype development as well as its entire structure will be presented and explained.

In the following three sections, I will first explain how the database was developed and how it was structured, as well as the models that compose it and its final scheme. In the second and third sections respectively, the Back-End and Front-End structures will be explained, as well as the different components that these encompass, their main packages used in their development, and details of the same.

5.1 DATA BASE

As mentioned in the last chapter of this dissertation, the database that was chosen and that will be used to support the data layer is of the non-relational type, better known as NoSQL, using MongoDB for that.

Due to the requirements survey, and the elaboration of the domain model, use cases diagram and finally the class diagram, and these approved by the stakeholders, the database models and schema were then defined.

During the preparation of this database, collections that will make up the final database were defined. The choice of each attribute in these collections was chosen so as to be able to store the data we intend to collect.

In the following sections I will show each scheme that makes up the database as well as its final diagram, and finally I will show where and how it was allocated.

5.1.1 *Schemes*

The schemes that were elaborated will support the software's data layer, each model represents an object belonging to the respective data collection, as well as we can have models within models, thus making it possible to have knowledge within each object. Next we will explain the different models and their components that will compose our final database schema. To represent these schemes the *JSON* notation was used, because MongoDB relies on this notation to compose its databases.

User Schema

In the listing below you can see the data that belongs to the user schema, which is responsible for store the data regarding a user.

```
{
    "name": String, // Name of the user
    "email": String, // Email of the user
    "password": String, // Password of the user
    "accessCode": String, // Access code to confirm email
    "role": String, // Role of the user in db
    "emailIsVerified": Boolean, // Boolean to confirm if email was verify
    "isLogged": Boolean, // Boolean to know if user login for the first time
    "requestResetPassword": Boolean, // Check if the user request a reset of the
        ↪ password
    "date": Date, // Date when the user was create on db
    "profile": ProfileSchema, // Profile of the user
    "cyclistProfile": CyclistProfileSchema // Cyclist profile of the user
}
```

Listing 5.1: User Schema

As you can see above, we can see that the user's schema has its different attributes, it should be noted that this schema uses two auxiliary schemes, which are the *profile* and the *cyclistProfile*, these two attributes use auxiliary schemes to represent their knowledge, which will be explained next, respectively.

Profile Schema

In the listing below you can see the data that belongs to the profile schema, which is responsible for store the data regarding the profile of the user.

```
{
    "city": String, // City of the user
    "postalCode": String, // Postal code of the user
    "gender": String, // Gender of the user
    "birthDate": Number // Birth date of the user
}
```

Listing 5.2: Profile Schema

Cyclist Profile Schema

In the listing below you can see the data that belongs to the cyclist profile schema, which is responsible for store the data regarding the cyclist profile of the user.

```
{
    "cyclistType": String, // Type of cyclist
    "bicycleType": String, // Type of bicycle that user use
    "frequentUseBike": String, // Frequency of the use of the bicycle
    "mainTransport": String, // Main transport of the user
    "bicycleBrand": String // Bicycle brand
}
```

Listing 5.3: Cyclist Profile Shhema

These two schemes shown above use a *flag* so that MongoDB doesn't assign *ids* to both since they belong to the user schema.

Travel Schema

In the listing below you can see the data that belongs to the travel, which is responsible for store the that regarding the travels of the user.

```
{
    "location": { // Object regarding the city and country that the travel take place
        "country": String,
        "city": String
    },
    "duration": { // Object regarding the duration of the travel
        "hour": Number,
        "minutes": Number,
        "seconds": Number
    },
    "distance": Number, // Distance in (km) of the travel
    "co2Saved": Number, // CO2 in (g) saved in the travel
    "averageSpeed": Number, // Average speed of the travel
    "routeSpeed": [Number] // Different speeds along the travel
    "routeCoordinates": [ // Array of arrays regarding the coordinates of the travel,
        ↪ this notaion is in GeoJSON
        [
            longitude, latitude, altitude, timestamp
        ]
    ],
    "date": Date, // Date that the travel was made
    "idUser": String, // ID of the User that this travel belongs
    "evaluaion": EvaluationSchema // Evaluation of the travel
}
```

Listing 5.4: Travel Schema

Above you can check the schema that was created to save the travel made by a user, two things should be highlighted, the first has to do with the ***idUser*** attribute, this is the user id that MongoDB created when the user was inserted in the database, so we have this travel linked to the user who made it and the second thing is that this schema will use an auxiliary schema called *Evaluation Schema* to save the evaluation of the respective travel, this schema will be explained below. The notation that was use for the attribute *"routeCoordinates"* is based on GeoJSON [52].

Evaluation Schema

In the listing below you can see the data that belongs to the evaluation schema, which is responsible for store the data regarding the evaluation of the travel.

```
{
    "traffic": Number, // Evaluation between 1-5 of the traffic
    "route": Number,  // Evaluation between 1-5 of the route
    "pollution": Number,  // Evaluation between 1-5 of the pollution
    "global": Number,  // Evaluation between 1-5 of the global appreciation
    "comment": String,  // Comment regarding the travel
}
```

Listing 5.5: Evaluation Schema

This schema shown above use a *flag* so that MongoDB doesn't assign *id* to it, because it's belong to travel schema.

5.1.2 *Diagram*

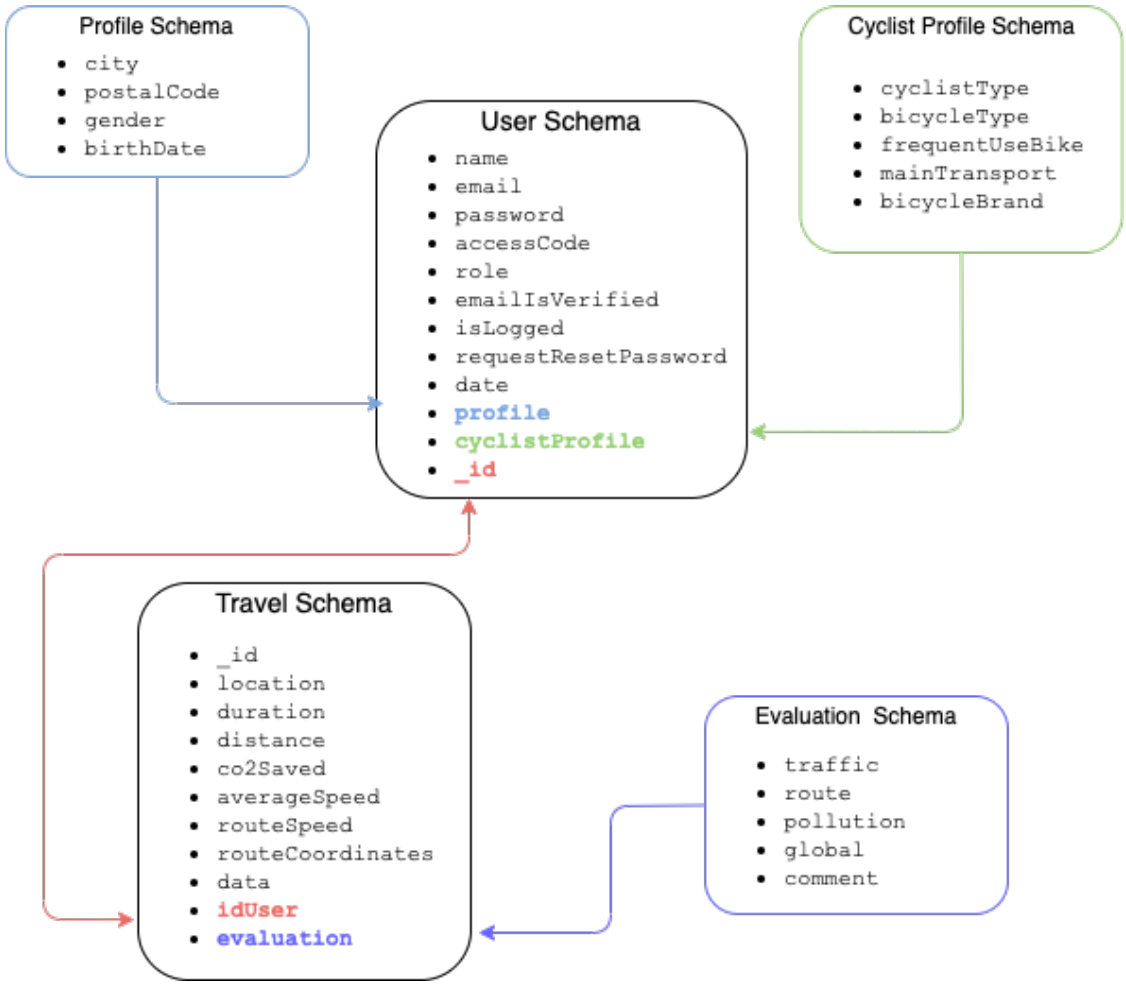


Figure 18: Diagram database
(Source: Own)

As you can see on figure 18, it shows the database diagram that was created and implemented, there are two schemes that are respectively *Travel Schema* and *User Schema*, these two will create two collections in our database where the data will be stored, these collections will be called **travels** and **users** respectively.

5.1.3 *Deployment*

In this section I will explain where the database was allocated and how it is represented in this service using images to demonstrate it.

The same as mentioned above was allocated in MongoDB Atlas [36], this is a tool that allows the creation of clusters and perhaps and creation of database within these clusters, thus allowing to have the database being online and the same being easy to query.

Allocating the database in this service allows us to have greater control over it, as well as the consultation of data in it, and we have a more secure database in which it can never go down or be hacked.

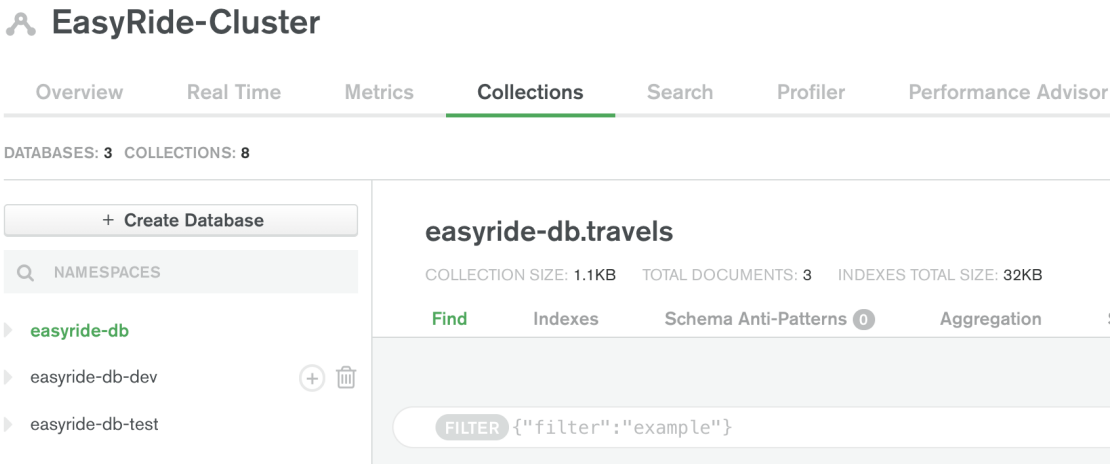


Figure 19: MongoDB Atlas
(Source: Own)

5.2 BACK-END

In this chapter I will explain and show the structure that makes up the Back-End, the main frameworks and packages that were used in its development, I will also show how the documentation of the Back-End was generated and presented, as well as how it is protected and how its deployment was performed.

The same was started using the *npx framework* [53] with the *express package*, as this combination we have initially built a template of a Back-End based on *Node.js* and *Express*.

5.2.1 *Structure of Back-End*

The Back-End is divided into 10 fundamental parts. These parts will be explained later. In the figure below you can check these different components.

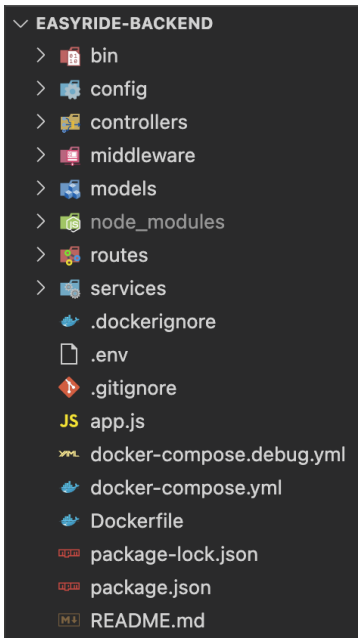


Figure 20: Structure of Back-End
(Source: Own)

Next, each of these components will be exposed and explained as well as their respective function in the Back-End.

bin folder

- The directory contains a configuration file named *www*, this file will be responsible for creating and starting the server;
- It is responsible for setting the server port, as well as all the *http protocol* logic.

config folder

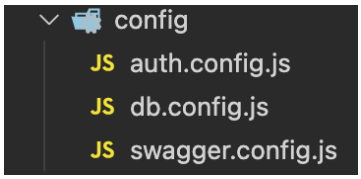


Figure 21: Structure of config folder
(Source: Own)

The directory contains 3 configuration files, these files will be explained next:

- *auth.config.js* – this file is responsible for storing and configuring the authentication part in the Back-End, this process will be explained in section 5.2.4;
- *db.config.js* – this file is responsible for storing and configuring important data, such as the name of the database and its username and password, to then access it and access settings;
- *swagger.config.js* – this file is responsible for making a configuration of the Back-End documentation itself, this documentation is generated with the help of the *swagger package* that will be explained later.

controllers folder

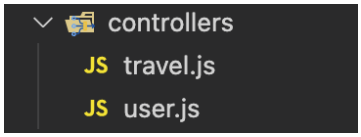


Figure 22: Structure of controllers folder
(Source: Own)

In this directory there are two files, the *travel.js* and the *user.js*, both are responsible for the communication between the information that comes to the Back-End through the routes and the database, they contain functions to do database operations, below you can see an example of these functions. The *user.js* file is responsible for operating on the **users** collection, and the *travel.js* file is responsible for operating on the **travels** collection.

```
/*  Function to save User */
const saveUser = async (user) => {
    const salt = await bcrypt.genSalt(10);
    const hashPassword = await bcrypt.hash(user.password, salt);
    user.password = hashPassword;
    const newUser = new User(user);
    return newUser.save();
}
```

Listing 5.6: JavaScript - Function that save a user

You can see more examples on appendix D.1.1

middleware folder

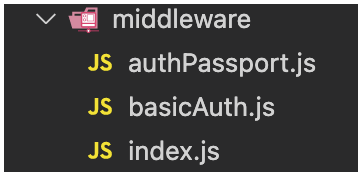


Figure 23: Structure of middleware folder
(Source: Own)

In this folder we can see that there are two important files, these being *authPassport.js* and *basicAuth.js* respectively, these files will be explained next. The *index.js* file is only for importing this directory.

- *authPassport.js* – this file is responsible for configuring the *passport package* which will be explained further on. In this file we have functions that have the function of protecting the internal routes of the Back-End, that is, routes after the user logs into the application;

```
/* Config Cyclist */
passport.use('isCyclist', new JWTstrategy({
    secretOrKey: authConfig.secretKey,
    jwtFromRequest: ExtractJWT.fromExtractors([extractFromSession]),
    passReqToCallback: true
}, async (req, token, done) => {
    try {
        if(token.user.role === 'CYCLIST') return done(null, token.user, null);
        return done(null, false, { message: 'Unauthorized.' });
    } catch (error) {
```

```
        Sentry.captureException(error);
        return done(error);
    }
}))
```

Listing 5.7: JavaScript - Function that protect intern routes

- *basicAuth.js* – in these files are functions and settings to protect the application's external routes, i.e. routes such as login and registration, these will be protected using the *Basic Auth strategy* that will be explained later.

models folder

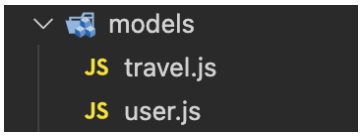


Figure 24: Structure of models folder
(Source: Own)

In this directory you can find two important files, these being *user.js* and *travel.js*, which will be explained briefly below:

- *user.js* – is responsible for representing and creating the *user* schema and the *users* collection, in it you can find information regarding these two components;
- *travel.js* – is responsible for representing and creating the *travel* schema and the *travels* collection, in it you can find information regarding these two components.

```
const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    min: 6,
    max: 255
  },
  email: {
    type: String,
    required: true,
    min: 6,
    max: 255
  },
```

```
  password: {
    type: String,
    required: true,
    min: 6,
    max: 1024
  },
  profile: {
    type: ProfileSchema,
    default: {}
  },
  cyclistProfile: {
    type: CyclistProfileSchema,
    default: {}
  },
  role: {
    type: String,
    required: true
  },
  emailIsVerified: {
    type: Boolean,
    default: false
  },
  requestResetPassword: {
    type: Boolean,
    default: false
  },
  isLogged: {
    type: Boolean,
    default: false
  },
  accessCode: {
    type: String,
    default: ""
  },
  date: {
    type: Date,
    default: Date.now
  },
}, { versionKey: false });
...
module.exports = mongoose.model('User', UserSchema, 'users');
```

Listing 5.8: JavaScript - User schema - file *user.js*

As you can see on the listing on top, this is an example of a model from the databases, and how it's represent using *mongoose package*. You can see more on appendix D.1.2.

routes folder

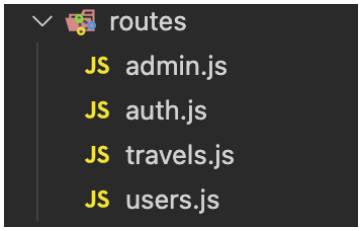


Figure 25: Structure of routes folder
(Source: Own)

In this directory you will find 4 files, these are *admin.js*, *auth.js*, *travels.js*, *users.js*, these files are responsible for controlling their respective routes and the flow of traffic and requests coming into the Back-End, these will also allow the operations of a *RESTful API*, which are GET, POST, DELETE and PUT.

- *admin.js* – this file will be responsible for the routes **/admin**, these routes will be management functions of the Back-End, the same route is protected by the *Basic Auth* strategy, with a configuration unique to this route;
- *travels.js* – this file is responsible for controlling the **/travels** routes, these routes have the function of controlling all the operations related to the *travels* collection, for example, adding a travel to the database;
- *users.js* – in this file we can find all the logic associated with the **/users** route, this route has the function of managing the *users* collection;
- *auth.js* – finally this file is responsible for the login or registration process in the application, as well as functions related to authentication processes, this file controls the **/auth** routes.

Below you can see two examples of functions belonging to these files, you can see more on appendix D.1.3.

```
router.post('/user', async (req, res) => {
  try {
    if(req.body) {
      const travel = req.body;
      const validTravel = Travel.validateTravel(travel);
      if (validTravel.error) {
```

```
        return res.status(400).jsonp({ message: validTravel.error.details[0].message
            });
      }
      const result = await Travel.saveTravel(travel);
      return res.jsonp({ message: 'Success on saving Travel.' });
    } else {
      return res.status(400).jsonp({ message: 'Travel invalid.' });
    }
  } catch (error) {
    Sentry.captureException(error);
    return res.status(500).jsonp({ message: 'Error on saving Travel.' });
  }
});
```

Listing 5.9: JavaScript - Route POST **/travels/user** - file *travels.js*

```
router.get('/users', passport.authenticate('isAdmin', { session: false }), async (req,
     res) => {
    User.listAllUsers()
      .then(data => res.jsonp({ data }))
      .catch(error => res.status(500).send('Error on get all Users: ' + error))
});
```

Listing 5.10: JavaScript - Route GET **/admin/users** - file *admin.js*

services folder

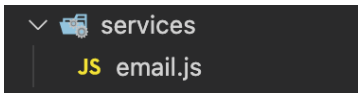


Figure 26: Structure of services folder
(Source: Own)

This directory is responsible for providing external services to the Back-End, in this case we only have one, in this case it is the *email.js* file, this file is responsible for giving the Back-End the ability to send emails to the client side.

app.js file

This file is the main Back-End file, in other words it is the core of the Back-End.

This is where all the necessary configurations and operations for the server to operate are done.

In this file we can find the connection to our database, then the *sentry framework* configuration that will be explain later, as well as the configuration of the automatic generation of documentation and also the definition of each of the routes that this server will provide, below I will briefly explain each important operation performed in this file.

- Database connection – this file is responsible for the connection to the database, you can see the code below;

```
/* Connection to MongoDB */
let db_name;
if(app.get('env') == 'development') {
  db_name = dbConfig.MONGO_DB_DEV;
} else {
  db_name = dbConfig.MONGO_DB;
}
const DB_URL = `mongodb+srv://${dbConfig.MONGO_USERNAME}:${dbConfig.
    MONGO_PASSWORD}@${dbConfig.MONGO_HOSTNAME}/${db_name}?retryWrites=true&w=
    majority`;
console.log('> Waiting connection to MongoDB...');
mongoose
    .connect(DB_URL, dbConfig.options)
        .then(() => {
            console.log(`> Successfully connect to MongoDB <${db_name}>...`);
        })
        .catch((error) => {
            Sentry.captureException(error);
            console.log('MongoDB connection refused: ' + error);
        })
```

Listing 5.11: JavaScript - Connection to DB

- Configure *sentry framework*, this was use to track and throw errors on the production server, you can see that in code below;

```
const Sentry = require('@sentry/node');
...
/* Config sentry */
if(app.get('env') == 'production') {
  Sentry.init({
    dsn: 'https://7fb5cb514e824b5da13941c32018aaa8@o536457.ingest.sentry.io
        /5655086',
    tracesSampleRate: 1.0,
  });
}
```

Listing 5.12: JavaScript - Configuration of sentry

- Configure and protection to routes – this file is responsible for secure and configure the routes that the server offer.

```
const travelsRouter = require('./routes/travels');
...
app.use('/travels', passport.authenticate('isCyclist', {session: false}),
    travelsRouter);
```

Listing 5.13: JavaScript - Routes configuration and protection

package.json file

This file is of importance for the Back-End, because this is where the name of the server is defined, its version, description, its author, as well as the possible scripts to run on it and finally a list of *npm packages* installed as dependencies of the project. This file is automatically created when the project is generated for the first time.

.env file

In this file we can find the static environment variables that will be used by the server, in this case the port on which it will be available.

Docker related files

We can see that there are about 4 files related to Docker, these files will help with the Back-End deployment. These files are *.dockerignore*, *docker-compose.debug.yml*, *docker-compose.yml* and *Dockerfile*, these files will be explained later.

5.2.2 *Frameworks and Main Packages Used*

Mongoose

Mongoose is of great importance to the Back-End, because it's because of this that all the management and communication with the MongoDB database is possible. Besides allowing the connection to the database, it also allows you to build the schemas and collections that define the database, as well as providing methods that can help, such as queries or data handling in the database, and imported methods in the package itself. This package is compatible with *Node.js*, and is used for all databases made in MongoDB [54].

Express

The Express express framework it's a framework of *Node.js*, is used to create abstractions of routes and middleware, so that it is possible to create *RESTful API's*. This framework is capable of working with different types of protocols, as well as different libraries that have direct integration with it [55].

Passport

Passport is authentication middleware for *Node.js*. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based application [56]. A comprehensive set of strategies support authentication using a username and password, or JSON Web Tokens.

JSON Web Tokens

The *Node.js* package JSON Web Tokens is used to generate and verify tokens in order to ensure the security of the intern routes [57].

sentry framework

This framework has the function of tracking and monitoring errors that may happen in software development [58], in this case this framework is being used in the production environment so that we always have access to possible errors or exceptions that may happen, if I had not used this framework I would not know how the Back-End would be working, nor if it would be reporting errors or not, so thanks to this framework it is possible to monitor it in real time.

5.2.3 *Back-End Documentation*

Swagger

Swagger is an interface description language for describing *RESTful API's* expressed using JSON. Swagger is used along with a set of open source software tools to design, build, document, and use *RESTful Web services* [59].

Express Swagger Generator

This *npm package* is responsible for generating the documentation User Interface (UI). With this package it is possible for us to comment and define each of the routes that our server will provide and then query those routes. Below you can see an excerpt of how this package works and how it generates the swagger documentation for the *RESTful API* [60].

```
const expressSwagger = require('express-swagger-generator')(app); // Connect the app
    to the swageer
...
/* Route Swagger */
app.use(basicAuth(authConfig.optionsAuthDocs)); // Protect the route /api-docs with
    Basic Auth
expressSwagger(swaggerConfig.optionsSwagger); // Configuration of the swagger
...
```

Listing 5.14: JavaScript - Swagger configuration

```
/**
 * This function is responsable for check if the token is valid
 * @route POST /auth/token
 * @param {string} body.token.required - Token
 * @group Auth - Operations about authetication
 * @produces application/json
```

```
 * @consumes application/x-www-form-urlencoded
 * @returns {string} 200 - Token is valid
 * @returns {string} 400 - Token was not send
 * @returns {string} 401 - Authorization issues - Token invalid
 * @returns {string} 500 - Error on check token
 * @security basicAuth
 */
router.post('/token', async (req, res, next) => {
    if (req.body.token) {
        try {
            const decoded = await jwt.verify(req.body.token, authConfig.secretKey);
            const user = decoded.user;
            const token = jwt.sign({ user }, authConfig.secretKey, { expiresIn: '90
                days' });
            return res.jsonp({
                token,
                user
            });
        } catch(err) {
            return res.status(401).jsonp({ message: 'Token invalid.' });
        }
    }
    return res.status(400).jsonp({ message: 'Token was not provide.' });
});
```

Listing 5.15: JavaScript - Example of the syntax to generate swagger documentation

We can verify through the code excerpts and the syntax of the swagger documentation that is made, that it is possible for us to generate a UI that will access the *RESTful API* documentation. You can see it in appendix C.1

5.2.4 *Authentication Process*

The authentication process is of great importance in the development of a *RESTful API*, as it is due to this process that we can protect it, thus being able to secure the data it processes as well as the entire application's Back-End. In this authentication process there are two techniques/protocols that stand out among the others, which are *Basic Auth* and *Passport with JWT*, respectively, those will be explain in next.

Basic Auth

This technique is the most used in *http* protocols, because it is based on the username and password technique to access a *RESTful API* [61]. These fields will be passed and encrypted in the headers of the requests, thus being possible to perform authentication, in this project this type of technique was used in the authentication routes, i.e., the external routes of the server, being these composed by the endpoint **/auth**. Below you can see an excerpt of code how it was used.

```
/* Protect routes with Basic Auth */
app.use('/auth', middleware.basicAuth.verifyBasicAuth, authRouter);
...
/* Middleware to validate Basic Auth */
const verifyBasicAuth = (req, res, next) => {
    const credentials = auth(req);
    try {
        if (!credentials || !checkCredentials(credentials.name, credentials.pass)) {
            return res.status(401).jsonp({ error: "Access denied." });
        } else {
            next();
        }
    } catch (err) {
        res.status(400).jsonp({ error: 'BasicAuth is not valid.' });
    }
};
```

Listing 5.16: JavaScript - Example of a Basic Auth authentication and it's configuration

Passport with JWT

To protect the internal routes of the *RESTful API*, the *passport-jwt framework* was used, it's in the purpose of authenticating the end-points of a server, using the JSON Web Tokens technique, and it's also possible to assign roles to users, so it's easier to configure the protection and access to these routes [62]. The routes that will be protected by this technique are, (1) **/users** and (2) **/travels**, next I will show an excerpt of the code of the configuration and implementation of this technique as well as a simple diagram of how it works.

```
/* Login strategy with passport-jwt */
passport.use('login', new LocalStrategy({
    usernameField: 'email',
    passwordField: 'password'
}, async (email, pass, done) => {
```

```
    try{
        const user = await User.findOne({ email: email });
        if(!user) return done(400, false, null);
        const validPassword = await user.isValidPassword(pass);
        if(!validPassword) return done(400, false, null);
        delete user.password;
        user.password = undefined;
        delete user.emailIsVerified;
        user.emailIsVerified = undefined;
        return done(null, user, { message: 'User authenticate.' });
    }
    catch(error){
        Sentry.captureException(error);
        done(erro);
    }
}));
```

Listing 5.17: JavaScript - Example of passport-jwt authentication

Above in the code example we can see how the *passport-jwt package* handles the login process, and in figure 27 we can see a small diagram of how this process occurs.

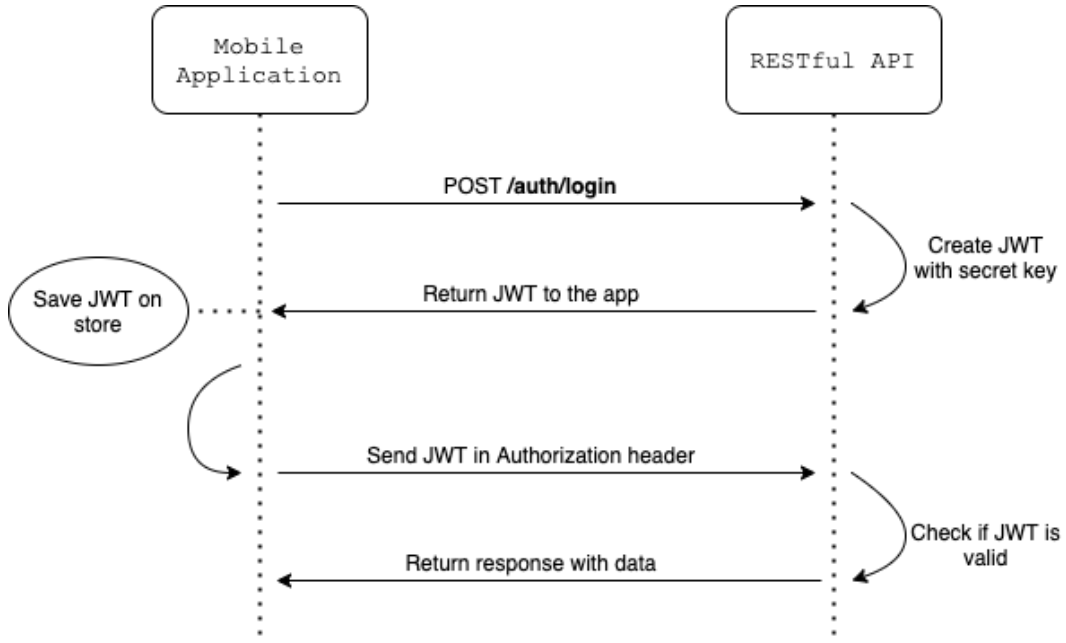


Figure 27: JWT authentication process
(Source: Own)

5.2.5 *Deployment*

A Back-End deployment is very important in software development, because it is with this that it will be possible to have a server in production and the same always online and available to be used anywhere in the world and with any type of connection.

To perform the deployment process two technologies were used, the first was *Docker* and the second was *Amazon Web Services (AWS)*, both will be explained next.

Docker

This technology is widely used in the *DevOps* world, because it is with it that most software applications are launched in production. The purpose of Docker is to create containers [63], in which within these are software images previously mounted or installed, so that it is possible to give a deployment of these images created. With this in mind then is to emphasize the importance of Docker in this case, to deploy the server Back-End, below you can see an excerpt of the *Dockerfile* that was used in this project.

```
FROM node:12.18-alpine
ENV NODE_ENV=production
# Create app directory
WORKDIR /usr/src/app
# Install app dependencies
COPY ["package*.json", "./"]
RUN apk add --no-cache make gcc g++ python && \
    npm ci --only=production --silent && \
    npm rebuild bcrypt --build-from-source && \
    apk del make gcc g++ pytho
# Bundle app source
COPY . .
EXPOSE 5001
CMD ["npm", "start"]
```

Listing 5.18: Dockerfile that was use

You can see more examples of Docker related code snippets in the appendix D.1.4.

AWS

This type of services allows the deployment of services or software, so that it is always online and available, AWS provides hundreds of services to the community, among these the chosen one was **Amazon EC2** [64], this is a virtual machine where any type of operating

system can be installed, and in this case *Ubuntu 20.04* was installed, thanks to this service that AWS provides us, it is then possible with the integration of Docker to do with our *RESTful API* that was created and developed to be allocated in this virtual machine, so we can access it via the Internet.

5.3 FRONT-END

In this section I will explain the entire development process of the Front-End, this refers to a prototype of a mobile application, is as previously stated will be developed using the *React Native framework* [65], this allows us to develop mobile applications for both *Android* and *iOS* hat both are the most used by the community. Next I will explain how the Front-End structure was separated and developed, as well as the different components of it, then I will explain how the *autonomous tracking* was developed, next I will highlight some important packages and frameworks that were used during the development of the software.

5.3.1 *Structure of Front-End*

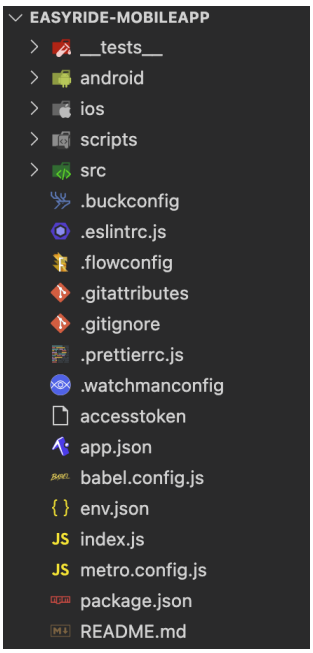


Figure 28: Front-End structure
(Source: Own)

As you can see in figure 28, we have a Back-End structure divided into 7 main components, then these components will be exposed and their respective internal layers. We can soon see that two components will be the **ios** and **android** directories, then it will be the **src** directory,

is where all the code and logic of the application, then we have a **scripts** directory, this will be responsible for additional code for the application, then we have important configuration files for the framework, then we have files related to the modules and configuration for *Node.js* and *NPM packages* then we have the **index.js** file.

ios folder

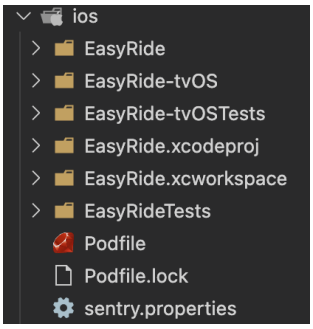


Figure 29: Structure of ios folder
(Source: Own)

As we can see in figure 29, this directory is composed of several *Xcode* [66] configuration files, as well as project components so that it is able to generate an app to be installed on the *iOS* operating system. Files such as **Podfile** are responsible for linking the project's dependencies so that they can be passed to *Xcode* to generate our mobile app. Finally we have the file *sentry.properties*, this is responsible for integrate *sentry framework* on the app to monitoring and tracking errors.

android folder

We can check through figure 30 the structure of the android directory, it is composed of configuration files based on **gradle** [67], these serve to be possible to generate the *apk* [68] of the mobile application so that it can be installed on the *Android* operating system. Next I will briefly explain the contents of the **app** directory within this same, because it is of great importance to generate the project, because it is within it that we get most of the configuration files.

This **app** directory contains, again, *gradle* files that will have the same functions as defined above. To generate the *apk*, *Android* needs to generate a private key, so that it can be associated with the application and only with it can a release *apk* be generated again, so that it can replace an update of the application on the smartphone, for example. Within this directory we can also find the assets of the application, such as images and icons of

the application and configuration files in *xml* of great importance so that the application can have access to various important services within the smartphone itself [69]. Finally we have the file *sentry.properties*, this is responsible for integrate *sentry framework* on the app to monitoring and tracking errors.

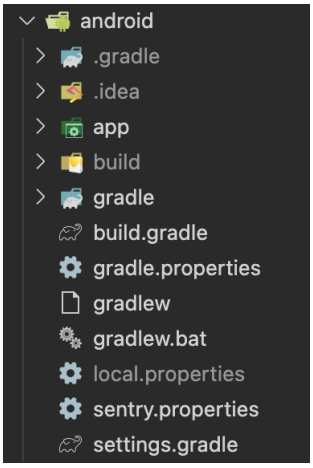


Figure 30: Structure of android folder
(Source: Own)

scripts folder

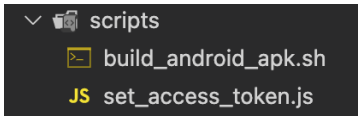


Figure 31: Structure of scripts folder
(Source: Own)

In this directory we find two files of the *script* type, these will be files that will serve as support during the development of the application, as functions that will be executed or *bash* code to be executed when needed, next I will explain the two files that make up this directory.

- *build_android_apk.sh* – this file as the name suggests, has as function to run a *bash script* that will allow the *apk* release to be generated, due to the complexities of these commands as well as the size of them I decided to put them together in one file, so it's only necessary to execute this file;
- *set_access_token.js* – as is known this application will use a map library, and to access to them, it will be necessary to configure an access token to them, the information will be

stored in a configuration file further explained and the access to these maps will also be explained further on, and it is with that purpose that file was defined.

src folder

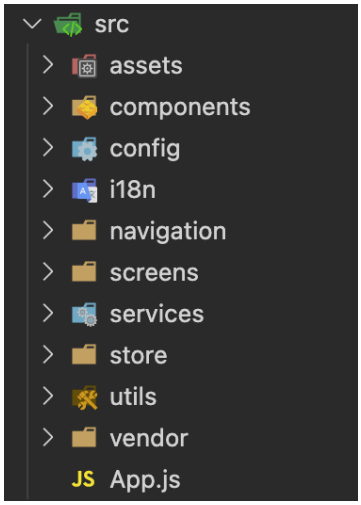


Figure 32: Structure of src folder
(Source: Own)

As we can see in figure 32, this board is responsible for containing all the code that is developed from the mobile application, it consists of 11 internal components that have their respective function during software development, in other words, it is in this board that we can find the source code of the mobile application. Next I will expose each of these components, as well as their relevance within the project and its components if necessary due to its complexity.

1. *assets folder* – We can see through the figure 33 that the components that make up this directory are: (1) the images folder, will be images that the application will use, then we have (2) the styles folder, this will be responsible for defining *CSS* styles that the application may need to define its screens of the application.

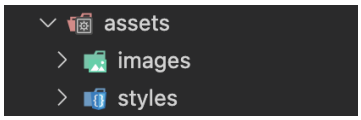


Figure 33: Structure of assets folder
(Source: Own)

2. *components folder* – In this directory we can find the components that are part of the application, these serve as an aid to compose the application, or main screens

of the application. They are seen as components of the application, not as classes or main screens, an example of this we can see by the file *WifiNotice.js*, which has the responsibility to show the user if he is connected to the internet or not, and the other files that make up this directory have the same preposition, which is to provide additional components and functionality that screens or that the application needs.

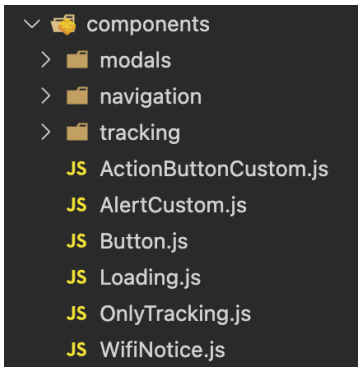


Figure 34: Structure of components folder
(Source: Own)

3. *config folder* – In this directory we can find configuration files, which in this case refer to the documentation of the generated code, as well as the *npm axios package*, which will be its purpose later on in this context.

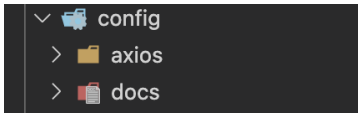


Figure 35: Structure of config folder
(Source: Own)

4. *i18n folder* – This directory is of great importance, because as we can see by the figure 36, this will configure and give the application the ability to support multi language, and we can see the files *en.js* and *pt.js*, which contain the texts to be used by the application, either in English or in Portuguese, thus giving the ability to have the application to use two languages or more. In this directory we also have the configuration file of *i18n* itself, further ahead I will talk about how this procedure was done as well as the packages that were used. You can see the structure on the figure 36. An example the code used can be seen on appendix D.2.3.

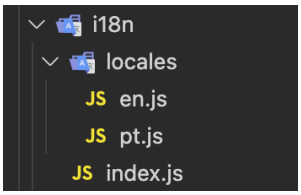


Figure 36: Structure of i18n folder
(Source: Own)

5. *navigation folder* – This directory has all the logic of the mobile application navigation, this means that this is where the different routes or screens that the application will have are differentiated and imported. As we can see in figure 37 the application will be divided into two different components, the first will concern the authentication screens of the application, i.e. the initial screens of the same, and secondly the internal screens of the application, these screens will compose the core of the application, and the directories are *auth* and *app* respectively, and the file *index.js* responsible for the global configuration and separation of these two components.

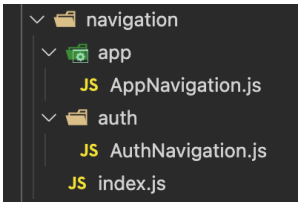


Figure 37: Structure of navigation folder
(Source: Own)

```
...
import AuthNavigation from './auth/AuthNavigation';
import AppNavigation from './app/AppNavigation';
...
const SwitchNavigator = createSwitchNavigator(
  {
    Auth: AuthNavigation,
    App: AppNavigation,
  },
  {
    initialRouteName: 'Auth',
  },
);
...
```

Listing 5.19: Example of the *index.js* file on navigation folder

6. *screens folder* – As you can see on figure 38, this directory will contain all the screens that the application will use, as it was said before, these will be separated by two different components, the *auth* and the *app*. As you can see in the figure below, the *auth* directory contains the *Home.js*, *Login.js*, *Register.js*, *ForgotPassword.js* and *RegisterOptional.js* screens, which will be the application's initial screens, and at last the app directory contains the application's internal screens, these screens will be the application's core, because they contain all the application's logic to be developed.

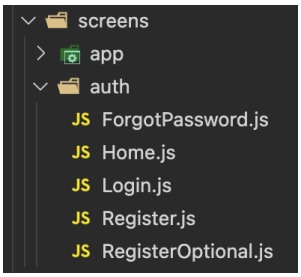


Figure 38: Structure of screens folder
(Source: Own)

7. *services folder* – In this board as we can see by the figure 39, we have the services that the application will use these services are those that are connected with the application's Back-End, in this case will be our *RESTful API*, which was explained in the previous section (5.2 section), with the files *travels.js* and *users.js*, we can then separate the shop by their respective endpoints, the first concerns the operations with the management of travels and the second with the management of the user itself, as login functions and so on. Below you can see an excerpt of the code taken from the *travels.js* file.

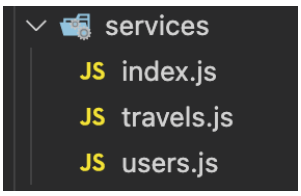


Figure 39: Structure of services folder
(Source: Own)

```
import axios from 'axios';
import { store } from 'store';
const saveTravel = (travel) => {
    const token = store.getState().authReducer.token;
    const headers = { 'Authorization': `Bearer ${token}` };
    return axios.post('/travels/user', travel, { headers });
};
```

Listing 5.20: Example of the *travels.js* file on services folder

8. *store folder* – This is of great importance for the mobile application, because it's thanks to these that will allow the application to have internal storage and how it is defined and constituted and operations in relation to it. We can see by the figure 40, that this is divided into three important parts, the first one being the types of actions that we'll have when we handle our store, these will be defined by action types, secondly we have the reducers, these are responsible for storing the information we want and also make a separation of it if necessary, within this directory of reducers, we have our main one called *rootReducer.js*, this will combine all the other reducers into one, then we have the other two in this case the *authReducer.js* and the *storeReducer.js*, the first one will be responsible for working with data regarding authentication processes and the second one will be in storing the desired information. The third part is the *index.js* file where all the store logic will be composed, this logic will be explained further ahead. We can see in appendix D.2.1 code excerpts taken from these files.

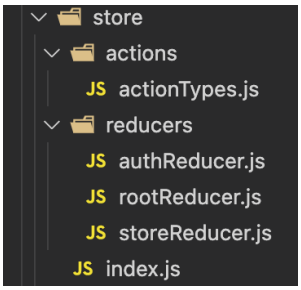


Figure 40: Structure of store folder
(Source: Own)

9. *vendor folder* – In software development we have the term *vendor* as a thing that is not ours, that is, in this folder we can find packages or code that was developed by other entities and that was reused and adjusted so that it is possible to integrate it in the application, because in an initial state it would not be possible to integrate it, so these packages or pieces of code were placed in this folder because they were not developed by me. These will be the *cities* folder, which contains all the cities in the world and finally we have the *react-native-popup-select package*, which was created with the intention of being a pop up in the application, but that in an initial state was not possible to integrate due to maintenance problems.

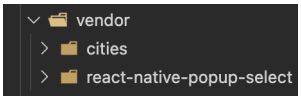


Figure 41: Structure of vendor folder
(Source: Own)

10. *utils folder* – As the name of this directory says, this is responsible for providing features of the utility part of the application, this is composed of a range of files that have the functionality to provide new functions, or external functions to the application, these include the *location.js*, *mapBox.js* and *navigationService.js*, the first being auxiliary functions when it comes to location operations or tracking, then we have the file that is responsible for configuring the library of maps that we use in the application and these are explained later and finally we have the file that concerns the navigation of navigation itself, being able to create new navigation operations and access the same from anywhere in the application. You can see its structure in figure 42.

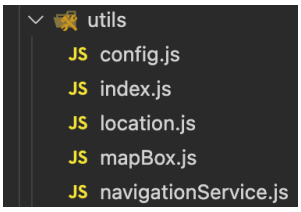


Figure 42: Structure of utils folder
(Source: Own)

11. *App.js file* – This file is of great importance for the application because it is the starting point for all the application logic. In this file we can find the initial configuration of the application as well as all the application to be called and its global components. You can see on appendix D.2.2 an excerpt of the code that is in this file.

config related files

- *package.json* – As said above in the Back-End section, this file is important for the framework to be used in this case of *NPM* and *react-native*, because it's in this file where all the project's dependencies are stored as well as the author, the version and executable scripts;
- *babel.config.js* – This file is a configuration file for the *Babel JavaScript compiler* [70], which in this case is responsible for simplifying the path of the local file imports, so that they can be simpler and easier to import and use;
- *app.json* – This file contains the name of the application and the name that will be displayed when installing it on an operating system, either on *Android* or *iOS*;
- *env.json* – This file is responsible for saving environment variables, or global variables that can be very useful during the development of the application.

index.js file

This is the main file in the execution of the application, it's this file that is called at application startup and that will proceed to register the application, in this case the *App.js* file that was explained above, as well as the association of the application name that is present in the *app.json* file, below we can see the code that makes it up.

```
import React from 'react';
import { AppRegistry } from 'react-native';
import App from './src/App';
import { name as appName } from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

Listing 5.21: *index.js* file of Front-End

5.3.2 *Autonomous Tracking Development*

Here I will briefly explain how the autonomous tracking of travels is done, and how this part of the application was developed. To perform this task were used two frameworks of great importance, being that the first [71], is responsible for making a background tracking of the trip when it is requested, and the second framework [72] is responsible for recognising if the user is practising the cycling activity. Below you can see code excerpts and their explanation and purpose.

```
...
const detectionIntervalMillis = 1000;
ActivityRecognition.start(detectionIntervalMillis);
if(Platform.OS === 'android') {
    await this.startForegroundService();
}
this.unsubscribe = ActivityRecognition.subscribe(async detectedActivities => {
    const mostProbableActivity = detectedActivities.sorted[0];
    if(mostProbableActivity.type == 'ON_BICYCLE' || mostProbableActivity.type == '
        CYCLING') {
        if(!flagTracking) {
            flagTracking = true;
            this._startTracking();
        }
    } else {
        this._stopTracking();
    }
```

```
}).bind(this);
...
```

Listing 5.22: Tracking the activity of the user

We can see from the code excerpt above that it is here that we make an autonomous tracking of the activity that the user of the application is at the moment practising, and as we can see all this logic is introduced in a background function of the application, so that it is always running. From the code we can see that when the cycling activity is detected the application starts tracking the trip.

```
...
BackgroundGeolocation.on('location', (position) => {
    const { routeSpeed, routeCoordinates, distanceTravelled } = this.state;
        const { latitude, longitude, altitude, time } = position;
        const timestamp = time
        const newCoordinate = { latitude, longitude, timestamp };
        ...
    });
    BackgroundGeolocation.checkStatus(status => {
        if (!status.isRunning) {
          BackgroundGeolocation.start();
        }
    });
    ...
});
...
```

Listing 5.23: Background Geolocation logic

With the code excerpt above we can make an autonomous tracking of the journey of the user who is using the application, through this logic we can extract vital and important data from the route in question.

5.3.3 *Frameworks and Main Packages Used*

React-Native MapBox-gl/maps

As the application will need maps for its proper functioning, and so that it is possible to view them, I had to choose which entity would be the one to provide, such entity was chosen **MapBox** [73], this is an American supplier of maps. Having this part chosen the next choice is what would be the way to do this integration in the prototype of the mobile

application, after a search came up the *NPM package* named **react-native-mapbox-gl/maps** [74], it is thanks to this package that the integration is possible, it has direct connection with the map provider.

React-Native Geolocation-Service

This *NPM package* was used to make it possible to track the cyclist, as well as the cyclist's exact location. The purpose of this package is to work with geolocation services [75], so it is possible to know the coordinates of the cyclist and more data related to it. The package has features that allow tracking the cyclist, as well as knowing the exact location of the cyclist.

React-Native Activity-Recognition

This *NPM package* is of great importance when it comes to autonomous tracking, because it is due to this that it is possible to know which activity the user is most likely to be doing [72], in this case the activity in question is cycling, and when the same is detected the application will start an automatic tracking of the travel in question

React-Native Core Components

During the development of this application, some components provided by the React Native framework were used [76]. These components are then adjusted and transformed so that they are compatible with the operating system in which the application itself is installed and running, transforming them into active code using the asynchronous communication technique [77].

axios

This application will be in contact with the *RESTful API* that was developed, for this communication to happen we used this package *NPM* called *axios*, it gives the application the ability to make http or https requests to the server that was developed [78], so that the information becomes dynamic. You can see the configuration of this package on appendix D.2.4.

i18next

This package will give the application the ability to have multi language support [79], it is thanks to this that it is possible to choose which texts will be visible in the application if a certain language has been chosen. Therefore, it is possible that the application has at least two or more possible translations.

redux

With the help of the *redux package*, it is possible for the application to have internal storage [80], this allows the application to be reactive, so that it changes state according to certain actions, as well as giving it the ability to save internal data, thus making it possible for it to work offline.

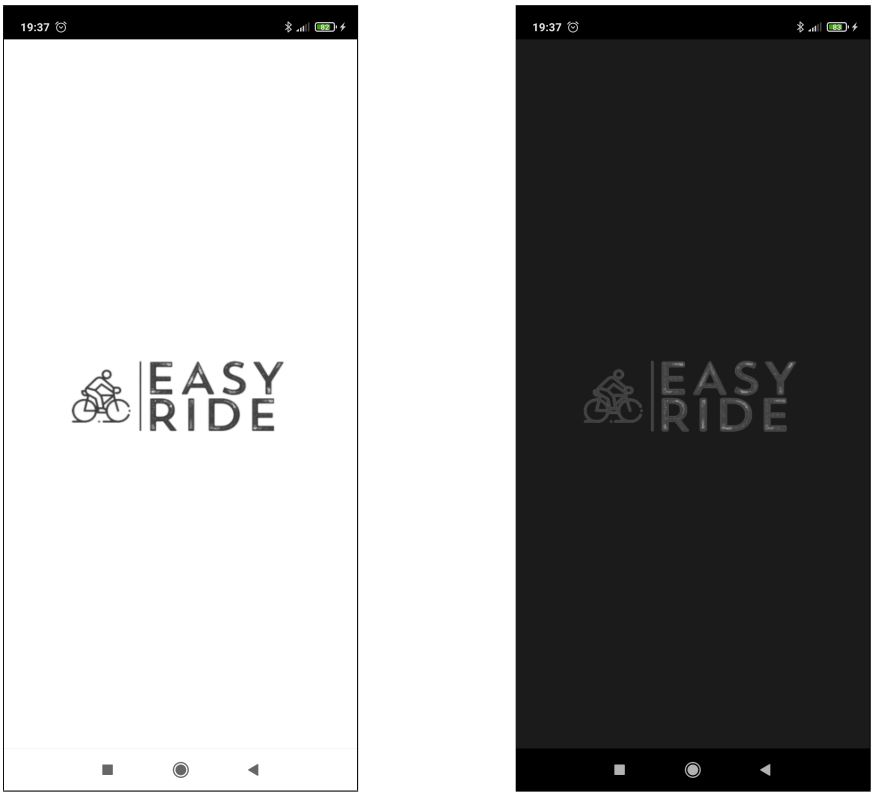
Next we have the chapter 6, where I will show the interfaces of the mobile application prototype and some of these functionalities will be shown in those interfaces, as well as their use.

6

INTERFACES OF THE PROTOTYPE

In this chapter will be presented and explained the main screens of this application, the pictures that will be shown next will be taken using the screen capture, these captures were taken using a *Xiaomi Redmi Note 9 Pro of 6.67 inches*. The order of them will be based on the navigation criteria of the application itself.

6.1 SPLASH SCREEN

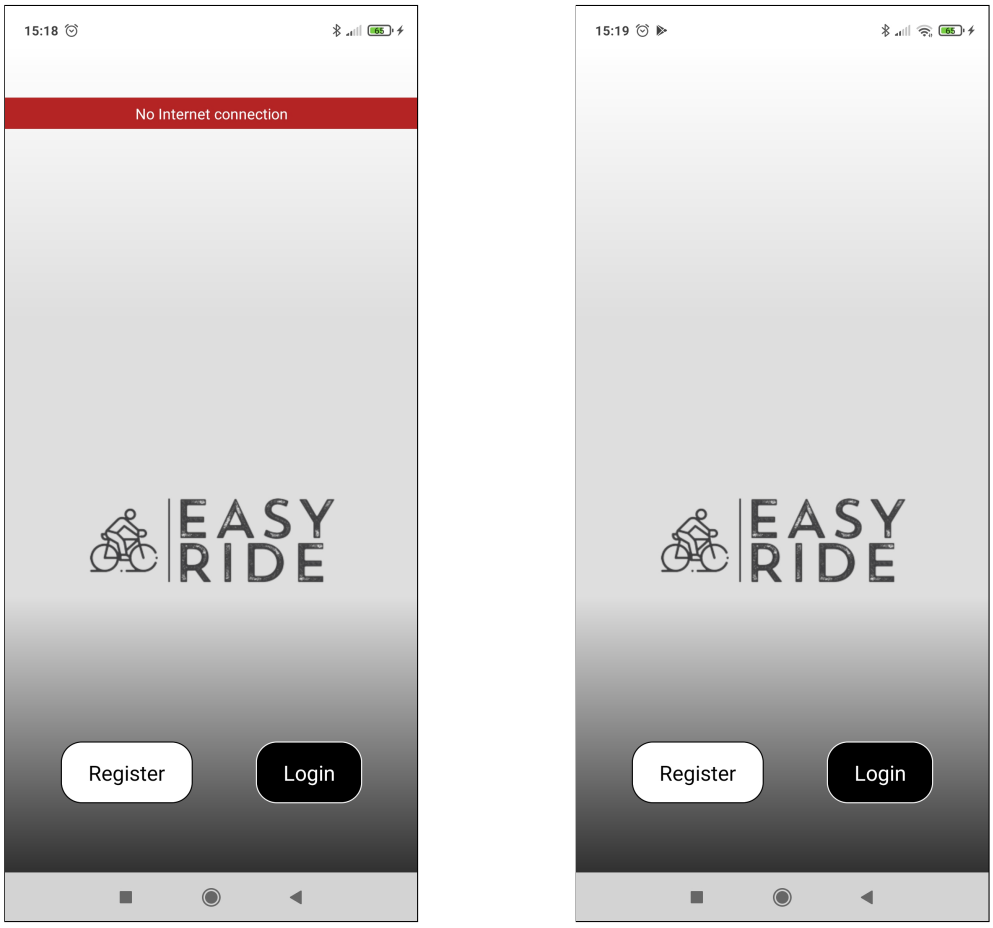


(a) Splash screen light (b) Splash screen dark

Figure 43: Splash screen

As you can see on figure 43, we have the application's **Splash screen**, this has two purposes, the first being when the application is opened, all its logic and configuration must take place and when this process happens, the Splash screen is shown, the same can also be used when we want to restart the application, when we want to change the application's language, for example. We can also see that as mentioned above the application is adaptable according to the theme of the selected operating system, having a dark mode or a light mode, as you can see on figure 43b and 43a respectively. The next prints of the screen will be made using the light theme.

6.2 HOME SCREEN



(a) Home screen no WiFi (b) Home screen with WiFi

Figure 44: Home screen

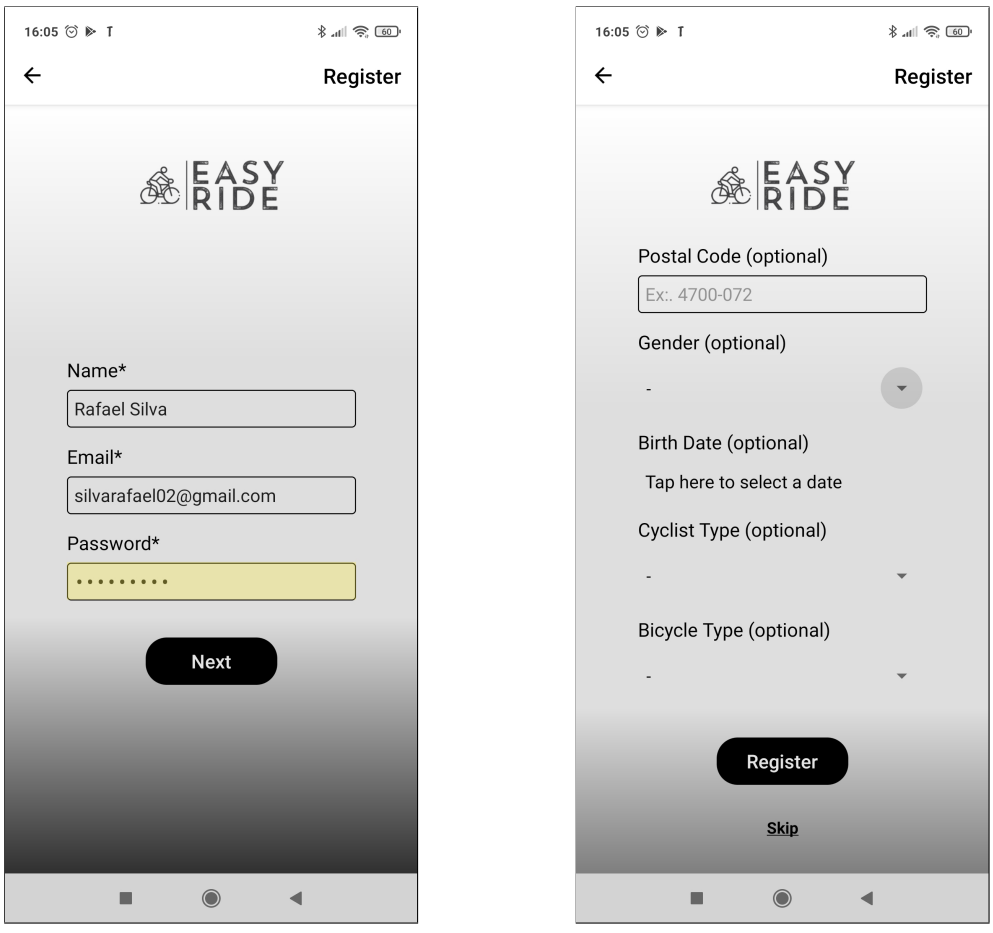
In figure 44 we can see the home screen of the application, this screen is shown in two scenarios, (1) when the application is opened for the first time, this is the screen that the user will see, (2) when the user logs out of the application. As you can see from figure 44a, it shows that the user is not connected to the internet, this component will be shown

throughout the application if the user is not connected to the internet, and from figure 44b we can see that it disappears when the user is connected, but before disappearing it informs the user that he is connected to the internet. Next, from this screen we can navigate to the application Registration or Login, this navigation can be done with the help of navigation buttons, or by simple swipe left or right depending on the operation to do.

6.3 Authentication screens

The authentication screens were divided into two parts, the first composed by the Registration screens that can be seen on figure 45b and the second part composed by the Login and Reset Password screens that can be seen on figure 46.

6.3.1 *Register screens*

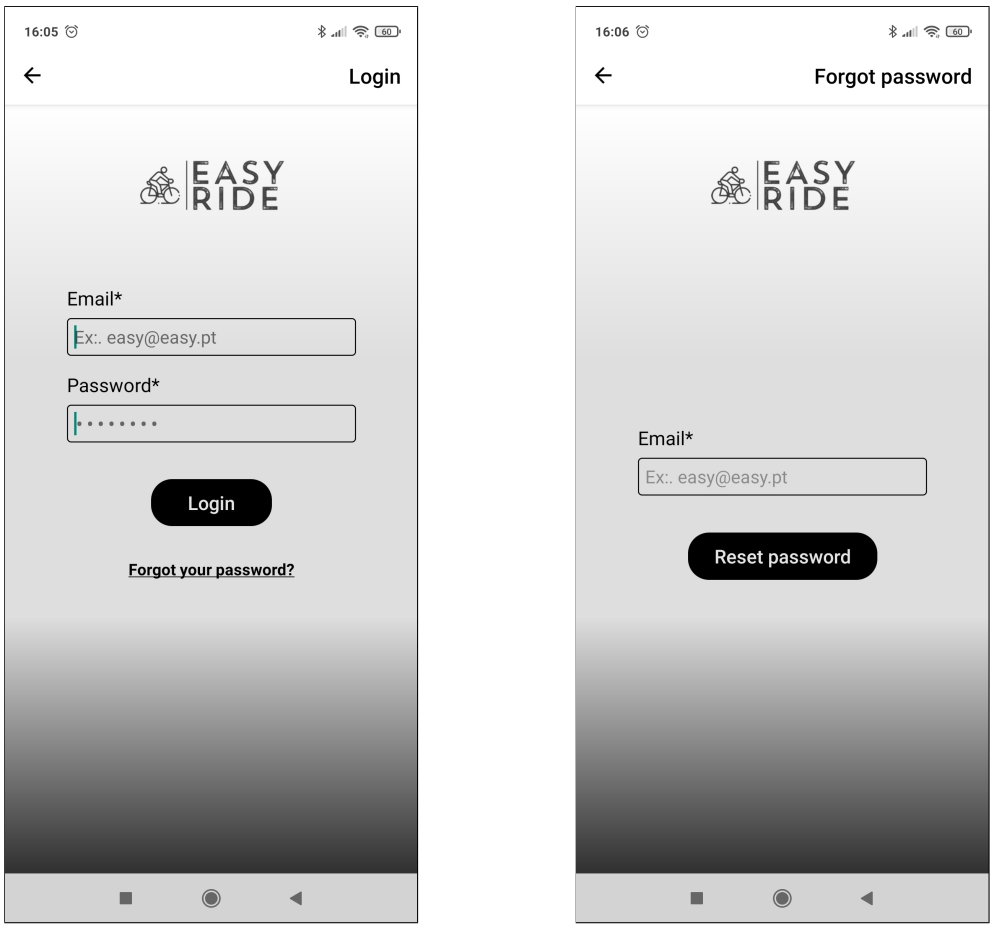


(a) Register screen (b) Register Optional screen

Figure 45: Register screens

By the figure 45 we can see that the registration process in the application has two steps, and the first step contains only three fields to fill that are mandatory, we can see that by the figure 45a, in which the fields to be filled are the **Name**, **Email** and finally the **Password**, if this registration is valid the user is redirected to the next screen that we see in the figure 45b, this screen has optional fields, in which the user may fill or not, these being the following: the postal code, the gender, the date of birth, the type of cyclist and finally the type of bicycle, after the user's registration is successful the user is redirected to the Home screen.

6.3.2 *Login and Forgot Password screens*

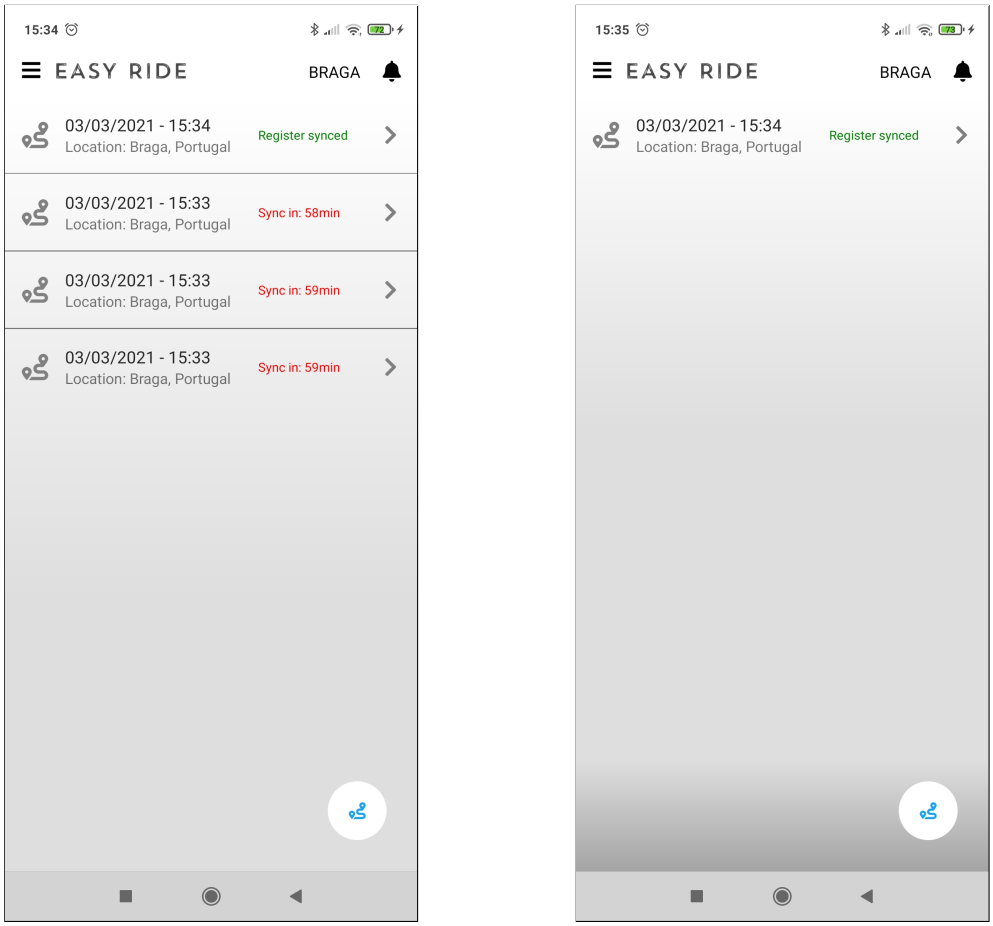


(a) Login screen (b) Forgot Password screen

Figure 46: Login and Forgot Password screens

By the figure 46 we have the two screens, the Login screen and the password reset screen, we can see that the first is requested the Email and Password, if the login is successful the user is forwarded to the internal part of the application. On the second screen only the Email is requested, which is to send an email to reset the user's password.

6.4 HISTORY'S SCREENS



(a) History last hour

(b) History Top 10

Figure 47: History's screens

Through figure 47 we can see that we have two screens, these are related to the user's travel history, and the figure 47a is related to the user's last hour history, in other words, the user's recent history, in which we can see that in this case one record has already been synchronised with the database, and we have 3 that have not yet been synchronised, if the user allows the hour to pass, these will be synchronised automatically and autonomously with the database, this was thought so that the user can choose which records he wants to see saved and which he does not want. Next by the figure 47b we can see a similar record, only that this one is relative to the records that are already in the database, being a kind of top 10 of your records.

We can also see that the application in its internal screens will have a button on the bottom right card, this button will have several functions depending on the screen where the user is, in this case it will be to navigate to the tracking screen. Finally we can see that

the application shows the city where the user is using the application in the header of the application.

6.5 NAVIGATION DRAWER

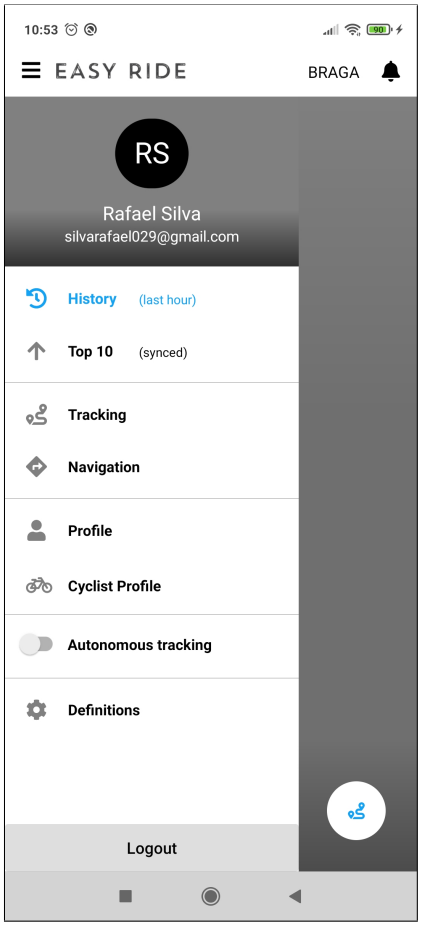
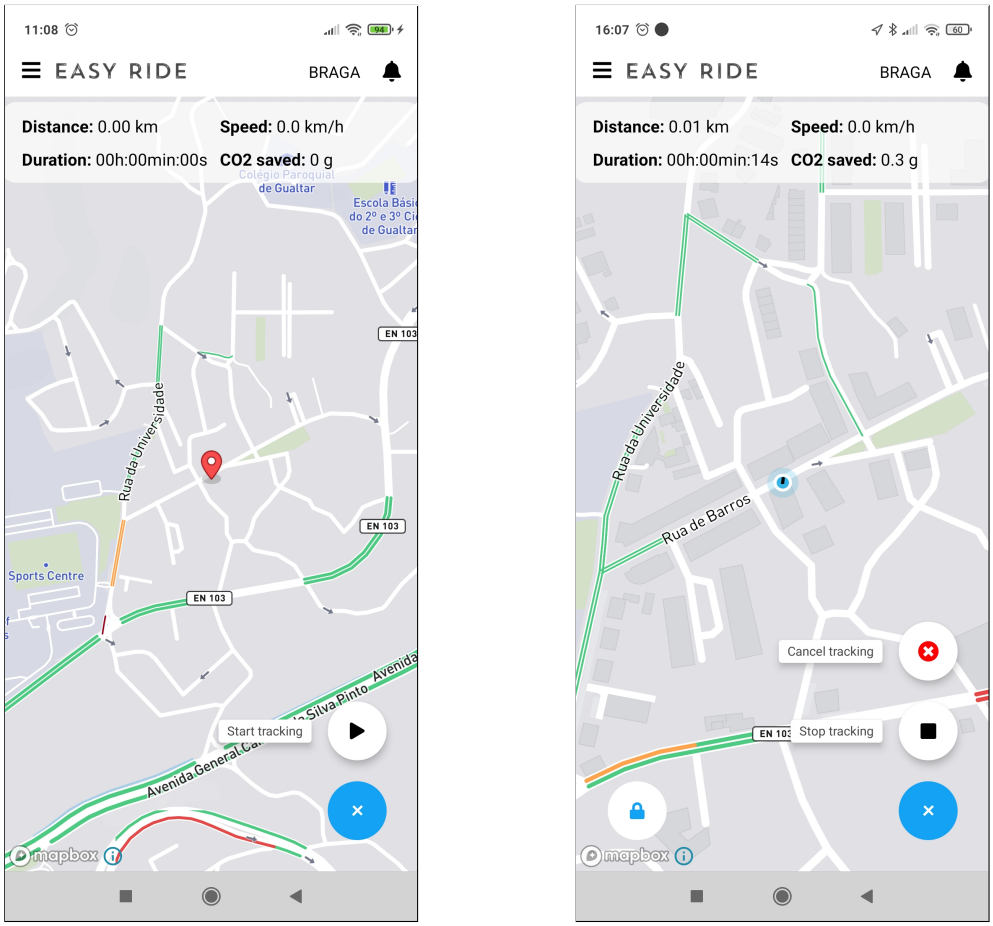


Figure 48: Navigation drawer

In order to allow an internal navigation in the application it was chosen a navigation drawer, as you can see in figure 48, this can be shown by swiping right on any screen, or clicking on the application name, or in the three bars that are in the application header. Here we can find the name and email of the user who is logged into the application as well as the different screens that it has, which are: **History last hour**, **History synced**, both were explained in the section above, then we have screens related to navigation, which are respectively the **Tracking screen** and the **Navigation screen**, and then we have screens related to user information, which are the **User Profile** and the **Cyclist Profile**. Finally we have the possibility to activate the **autonomous tracking** of the application, and then we have the screen relating to the **Settings** of the application itself. At the end of this navigation

menu we have a button if the user wants to **logout** of the application. Next I will be shown and explained these missing screens.

6.6 TRACKING SCREEN



(a) Start tracking

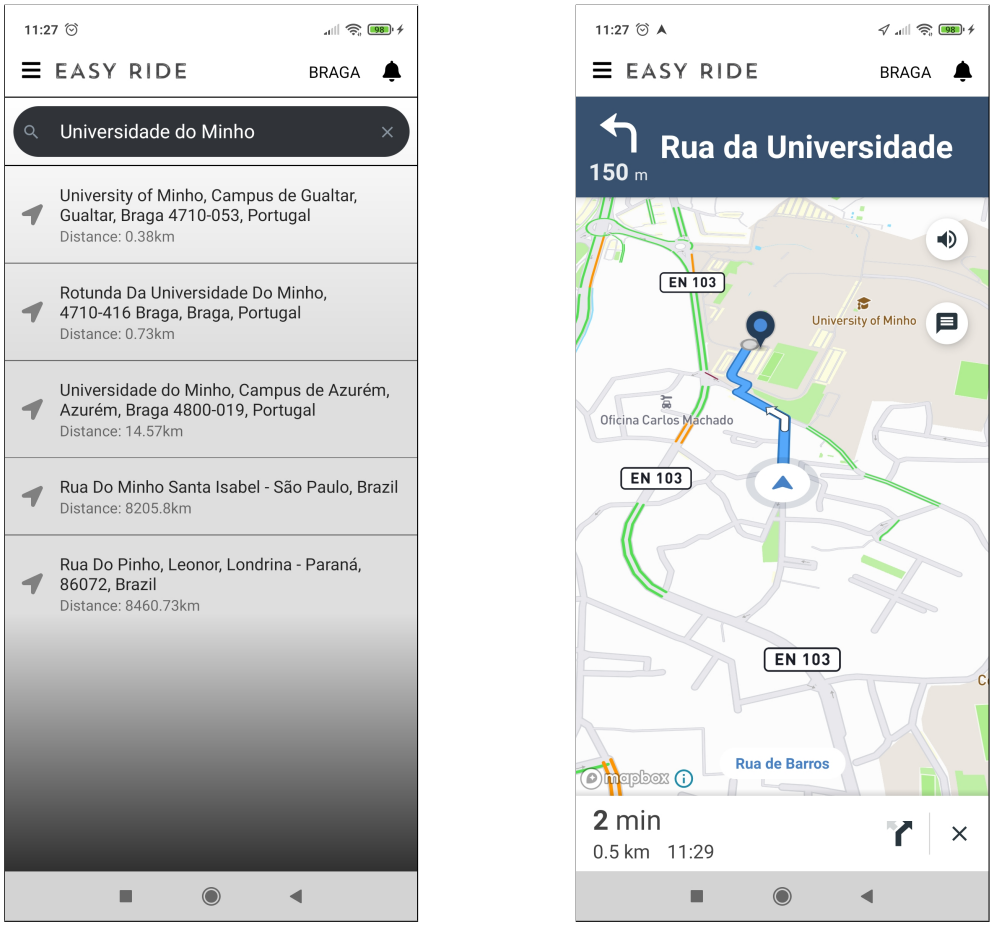
(b) On tracking

Figure 49: Tracking screen

In figure 49 we can see the tracking screen in two modes, and the first mode shown in figure 49a is the mode in which the user has not yet started the tracking, we can see that the button in the lower right corner will assume other functions as was said above, the user can start your tracking by clicking the play button, the same can also consult the map without having started the navigation. In figure 49b we can see the second mode of the screen that is when the user is already performing the tracking, we can see that the button in the lower right corner immediately takes on other functions, and the same are to stop scanning or cancel the tracking, we can also see that the top of the screen is transmitted to us information on the tracking, such as distance, speeds, duration and co2 that was saved, we can also see that the user has the ability to check their position in real time and the places where it will

pass, you can also Lock the phone if you want. Also the styles of the maps will change by hour, in other words, the application have maps by day and maps by night, you can see more figures about this screen on appendix E.1.

6.7 NAVIGATION SCREEN



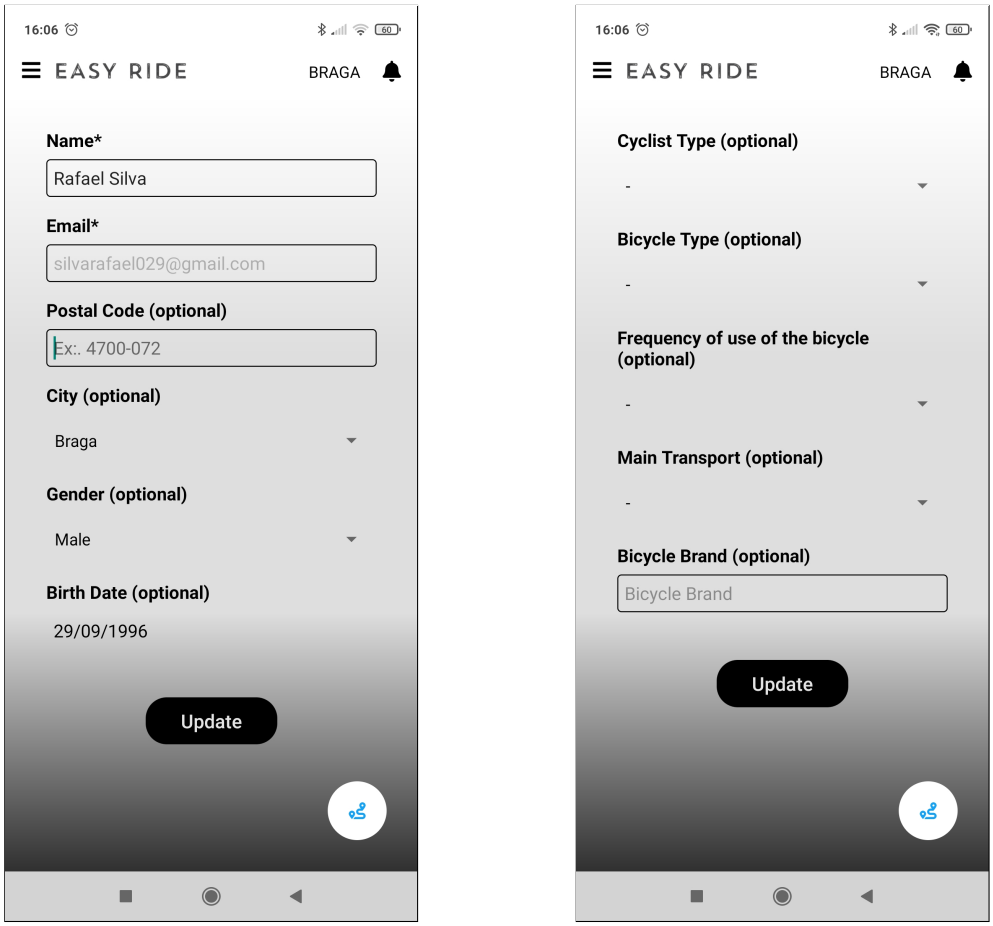
(a) Search place (b) On navigation

Figure 50: Navigation screen

This screen is responsible for the navigation system of the application, i.e., it helps users to get to the place they want. As you can see in figure 50 this screen has two modes, the first mode is the search for the place where you want to travel, we can see in figure 50a, where the user writes the place where he wants to travel and the application provides the 5 best places that fit your search and shows the distance of them in relation to the location of the user, this search is done reactive, ie, the user just start to write the place and the application will automatically load the sites that resemble it. Then the user can click on the place to which he wants to travel and we enter the navigation mode which is shown in figure 50b where we can see the instructions and navigation route, as well as the time it takes to get

there and the distance, the same navigation can have sound instructions if you prefer, the same navigation can also be cancelled if you prefer.

6.8 PROFILES SCREENS

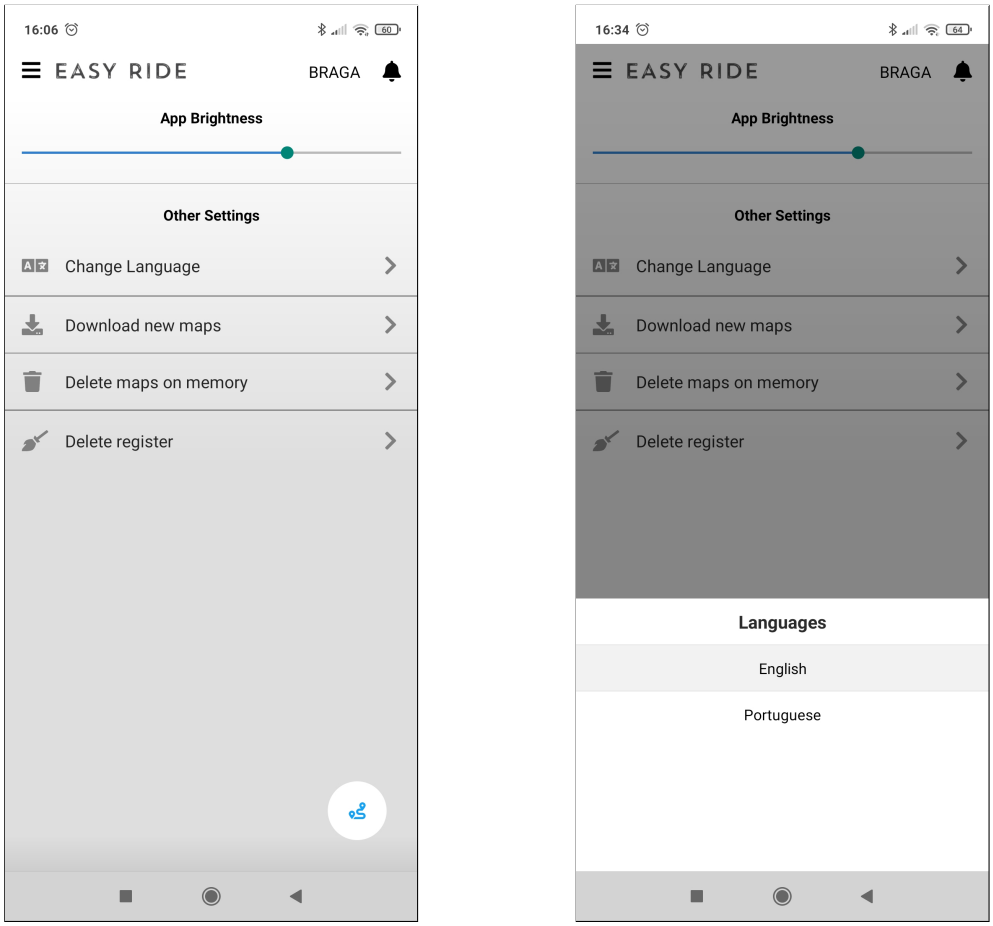


(a) User profile (b) Cyclist profile

Figure 51: Profiles screens

By the figure 51 we can see that we have two profiles, the first being the user profile that we can see by the figure 51a, and the second the cyclist profile by the figure 51b. The first profile, which in this case is the user's, allows us to have personal information about him, in this case will be his name, his birth date, his city, hist postal code and finally his gender. The cyclist profile has more to do with the cycling component, where the user can say what type of cyclist he is, the type of bicycle he uses, how often he uses it, his main means of transportation and finally the brand of the bicycle.

6.9 SETTINGS SCREEN

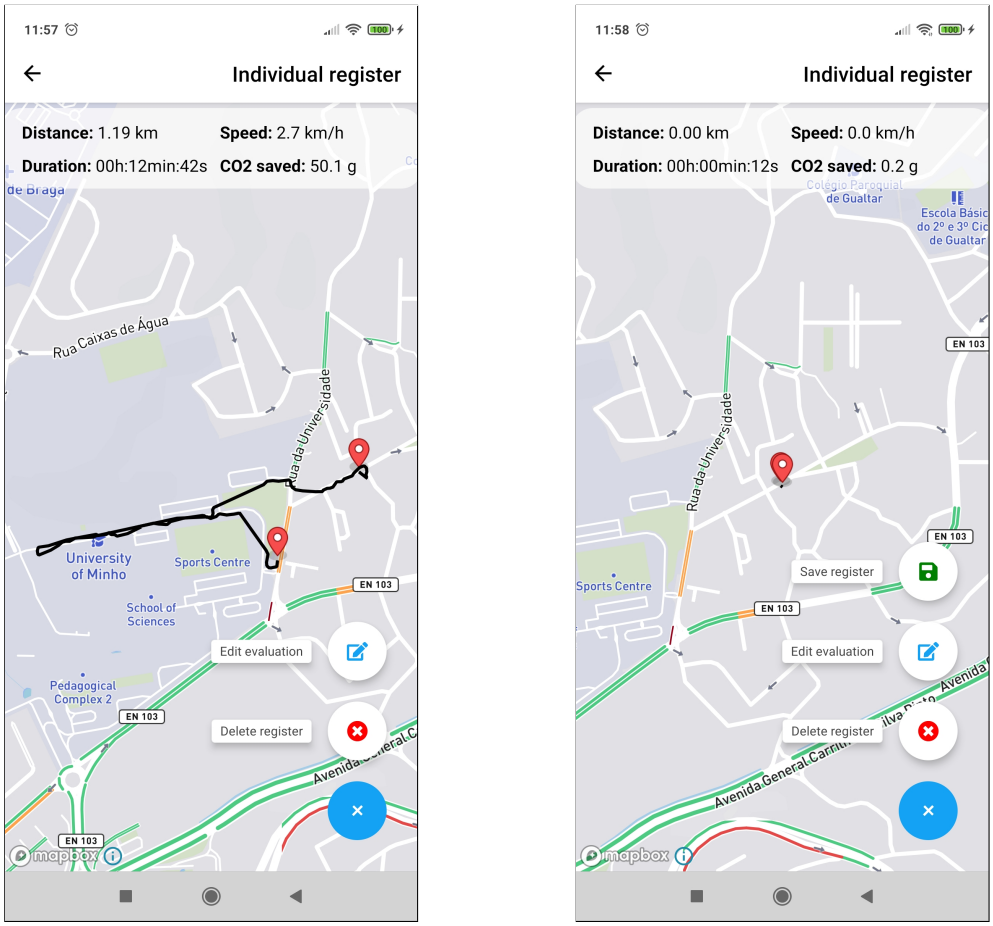


(a) Definitions (b) Change language

Figure 52: Settings screen

On this screen we can find the application settings, the same can be seen by the figure 52a, where we have the following settings: (1) change the application language, (2) download maps to the application, (3) delete the maps saved in the application memory, and (4) delete all the records that have not been synchronised yet. You can also change the brightness of the app. In figure 52b we can see the menu to change the application language, where the user chooses the language that he/she wants and the application will restart with the chosen language.

6.10 INDIVIDUAL REGISTER SCREEN



(a) Register synced (b) Register not synced

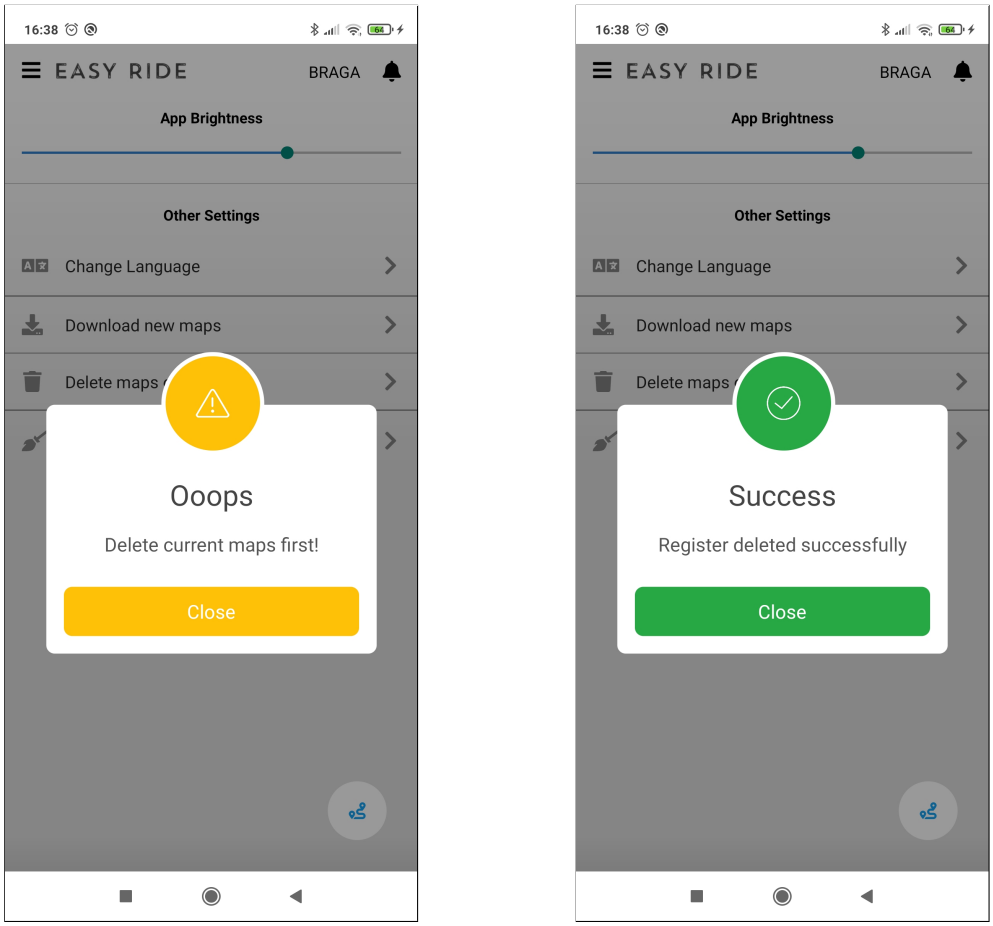
Figure 53: Individual register screen

On this screen we can see the information corresponding to each individual record that the user has in the application, or in the database associated to them. Figure 53a shows the record that has been synchronised, where we can see that operations can be performed on this record. Figure 53b shows the record that has not yet been synchronised where we have similar operations, but we have a new operation, which is to save the record in the database.

6.11 COMPONENTS

In this section I will briefly talk about auxiliary components that were used in the different screens that were exposed here. These include the alerts that make up the application, and the modals, which are a kind of auxiliary screen to be used.

6.11.1 *Alerts*



(a) Alert warning (b) Alert success

Figure 54: Alerts

As you can see from figure 54, these alerts have two modes, the warning mode and the success mode, these alerts serve to notify information to the user, for a better experience using the application. By figure 54a we have the warning alert and by figure 54b the success alert, these alerts also contain internal messages, which can be changed according to the situation, more examples of these alerts can be seen in appendix E.2.

6.11.2 *Modals*

These components are used so that it is not necessary to make a new screen, i.e., it is a popup window that can be opened and closed as needed. These are used when the user first logs into the application, it corresponds to the modal of the cyclist profile that you will be able to see in figure 65a. The second concerns the evaluation of the travel itself, which can be seen in figure 65b.

7

REVIEW AND ANALYSIS

In this chapter I will speak in the final part of software development that is the review and analysis of the software that was developed, in this case will be the final version of the prototype application that developed throughout this dissertation, as well as the project as a whole joining the two components of the project, ie, the Back-End and Front-End. This serves to have a first review and critical analysis of the work that was developed as well as have a feedback of possible improvements in the application, as well as an analysis of it, to know if it is fulfilling the purpose for which it was developed.

Then I will have a section of the features that the application has, this will expose all the features that the application has. Finally I will have an analysis section, where here I will have possible improvements in the application, as well as features that in the future may be added to the application, this section will also make a critical analysis of the work collecting my opinions about the application as well as a small number of user testers of this prototype.

7.1 FEATURES OF THE PROTOTYPE

In this section I will show and briefly explain some functionalities and features that were incorporated in this prototype application, these represent the requirements that were exposed in section 4.1.

1. Application is capable of registering a user as well as logging him in;
2. Application has internal storage in case it is needed;
3. The application saves the session, this allows the user to login only once, and the first time all the information is saved, so when the user accesses again the application is not necessary to login again, this is due to the use of *redux framework*;
4. The use without internet connection is possible when the first login is made, that is, it is possible to use the application without the need to be connected to the internet;

5. The application allows the user's profile and cycling profile to be modified;
6. One of the main functionalities of the application is the tracking of the user himself, he can do this and at the same time consult his position in real time, as well as check his current route, within this tracking he can consult the following data:
   - Speed;
   - Distance;
   - Time of the travel;
   - CO_2 saved.
7. At the end of a travel you will be able to evaluate the travel according to the criteria that have been defined if you wish to do so;
8. The travels made by the user will be saved in the internal memory of the application for a period of one hour so that the user can delete or edit data from them, after that hour the application synchronises the travels with the database automatically if the application is connected to the internet, otherwise wait until there is a connection;
9. The user can consult a top 10 of his travels already saved in the database;
10. The application allows autonomous tracking by the cyclist, that is, if the cyclist wants the application can make an autonomous and automatic tracking of their travels, this will only be activated if the application detects the use of bicycle;
11. The application allows navigation within it, the user can search for a place to which they want to travel, and after selecting the place the application shows the route and helps in navigation to the given place, thus allowing the user to have access to navigation;
12. The application allows the maps to be saved on internal memory for use without an Internet connection;
13. The application selects the styles of maps to be shown according to the time at which it is being used, this means that we can have maps made to be used at night, or maps for the day, for a better view of them;
14. The application is reactive thanks to the *redux framework*, this means that it can change states dynamically, making it more *user friendly* to use;
15. The application can also adapt to different internal themes of the operating system itself, that is, it changes according to the theme chosen by the smartphone user, that is, either dark mode or light mode;

16. The user can choose the language in which the application is translated for his comfort, and can change the brightness of the application.

7.2 ANALYSIS

As we can soon notice, this prototype application is a proof of concept, i.e., it is not a tool that can be used by the general public for their daily use, but a tool that proves an idea, in this case this dissertation. Note that this application is only a prototype and is not the final version of it, because in this application there are not all the final functionalities that could be added during the development.

We can consider this application as a means of obtaining data, i.e., it is a tool to collect *"metadata"*, which in the times in which we live are very important, because as was said in the first chapter of this dissertation it is of great importance to have an idea of the cyclists who exist in a particular city and what are their usual routes. With this data collection it is possible to build a virtual map of the cycling mobility within a city, and how this can improve it. This prototype application enters this sector, as it gives us the ability to collect data in real time, as well as the collection of personal data of each user, and this data collection is not intrusive for the user in question.

To point out an important aspect in this application that is denoted of *"auto-tracking"*, with this functionality the application will be able to track the movement of cyclists in an autonomous way, being always ready to start a tracking when the same will ride the bike, this is of great importance because this way the user does not have to be worried in activating the application constantly, and this way the application has an *"autonomous"* way of working without any iteration being necessary with it. This is a good aspect because it increases the amount of data collected by the application.

In a general analysis I think that what was developed here is of great importance, as it is a tool that can be used in any city to collect cycling data and with these data build a better city for cycling mobility.

7.2.1 *Autonomous/Manual tracking*

In this sub-section I will take a more critical and in-depth look at the different states that tracking can be in, and how these different states can conflict with each other if not well planned.

Here we can verify the introduction of two agents, the first being the cyclist and the second being autonomous tracking by the application, both have the ability to start a travel and end it, thus making possible conflicts may exist in the interaction of these two agents. Having this logic of autonomous and manual tracking in the app at the same time is complex, because

we always have to pay attention to both so that there are no conflicts between them and so that we can always collect viable data. Another aspect to note is that when a journey has not been stopped automatically, either by autonomous tracking or by the user with respect to manual tracking, this can mislead the data that has been collected.

In figure 55 we can then see a simple state diagram in which we can find all the logic that contains the manual and autonomous tracking, then I will briefly explain these different states and why they are represented here in this diagram.

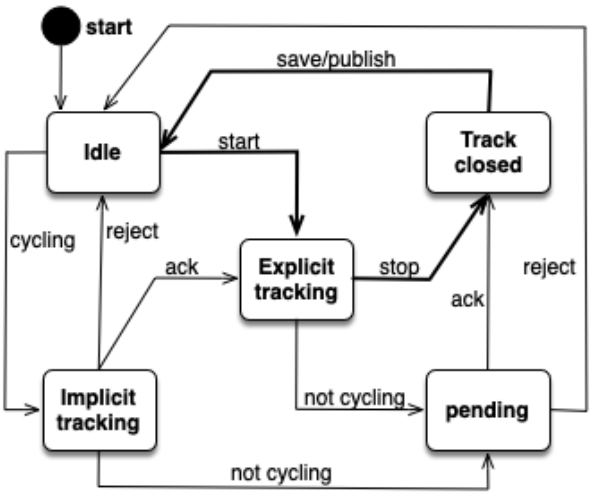


Figure 55: Tracking state diagram

- We start with the **"idle"** state, this state represents the initial state, where the mobile application is not doing any tracking;
- Next we have two fundamental states, which are:
  - **"implicit tracking"** - this state concerns when the application detects that the user is practising cycling, if the user is already doing a manual tracking this will not be active, finally when the application detects that the activity has ended this enters the **"pending"** state;
  - **"explicit tracking"** – this state concerns manual tracking, if the application is doing autonomous tracking this is stopped, and when the user finishes their tracking this returns to the **"track closed"** state.
- The **"pending"** state is when we have, for example, a travel that has not been stopped by the user or the application, and its validation is pending action by the user;
- Finally we have **"track closed"** state that ends the cycle of a tracking by saving the travel and returning to the **"idle"** state.

8

CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusion of all the work carried out. In a first moment, the main contributions are mentioned and then topics for future work are addressed.

8.1 MAIN CONTRIBUTIONS

With this dissertation several themes were raised that were exposed in section 1.3, 1.4 and 1.5, in these respective chapters several aspects that constitute the foundations of this dissertation were exposed, these contribute actively to the development of this mobile application prototype that was developed and explained here, this application prototype aims to improve the cycling mobility in a smart city. Through this application is possible a data collection of a particular city so that local entities can know where are critical areas of mobility in their city and how to improve mobility and promote sustainable mobility. With this prototype application it is possible to have an initial proof of concept and an initial version, i.e. a "beta" version of how an application of this type could be, and thus try to innovate a sector that lacks solutions to the problem of cycling mobility.

All this work here aims to make it easier for local authorities to collect data so that they can improve their city in a sustainable way.

We can then answer here certain questions that may be raised at the end of the development of this dissertation.

1. How could the app improve cycling and sustainable mobility?
    - The application's first function is to collect metadata so that this can later be used by local authorities, for example for the construction of new cycle paths;
    - The same application also encourages cycling by cyclists, as it shows the cyclist useful information that they can view while cycling or if they want to check their history.

2. How does the mobile app help cyclists in their daily lives?
    - As seen above the application provides two screens as interactive maps on which the cyclist can choose to only start tracking and thus consulted important information on that tracking in real time;
    - Then the cyclist will be able to choose a destination to which they want to travel and application will provide a possible route to the destination;
    - The application has the ability to work in offline mode, this is of great advantage, because then it is not necessary that the cyclist is connected to the internet;
    - Finally, thanks to the *"auto-tracking"* feature, it is possible for the cyclist to be monitored autonomously without needing access to the application.
3. This mobile application work in every city?
    - As we have seen this application does not depend on an expected city to work, it can work in any city without any problem;
    - This allows local entities to know what is happening in their city as the use of this application;
    - You can improve the city because we have data for each city, independent of each other.

8.2 FUTURE WORK

All the work initially proposed was developed, however there are points of improvement for future work, such as improvement points for future work, such as:

- Use of *HTTPS* to ensure an additional layer of security;
- Improve the responsiveness of some components to provide a good user experience on any size of screen;
- Construction of a dashboard for application management by an Administrator;
- Integration with social networks;
- Integration with local events, so that cyclists can participate in them;
- Possible integration of shared routes between cyclists;
- Development of a Web Interface for cyclists to access their data, and heats maps of their travels.

BIBLIOGRAPHY

[1] B. Huang, T. Thomas, B. Groenewolt, T. Fioreze, and E. van Berkum, "How to use smartphone apps to encourage cycling: Clues from a living lab with smart in enschede," in *ITS European Congress 2019*, 2019. 13th ITS European Congress 2019 : Fulfilling ITS promises, ITS Europe 2019 ; Conference date: 03-06-2019 Through 06-06-2019.

[2] United Nations, "The 2030 agenda for sustainable develpement." https://sdgs.un.org, 2015. Accessed 2020-12-23.

[3] J. Pucher and R. Buehler, "Cycling towards a more sustainable transport future," *Transport Reviews*, vol. 37, no. 6, pp. 689–694, 2017.

[4] J. Pucher and R. Buehler, "Walking and cycling for healthy cities," *Built Environment*, vol. 36, 12 2010.

[5] P. Oja, S. Titze, A. Bauman, B. de Geus, P. Krenn, B. Reger-Nash, and T. Kohlberger, "Health benefits of cycling: a systematic review," *Scandinavian Journal of Medicine & Science in Sports*, vol. 21, no. 4, pp. 496–509, 2011.

[6] G. Salat, Serge; Ollivier, "Transforming the urban space through transit-oriented development : The 3v approach.," in *World Bank, Washington, DC.*, 2017.

[7] D. Arancibia, B. Savan, T. Ledsham, and M. S. Bennington, "Economic impacts of cycling in dense urban areas: Literature review," 2015.

[8] T. Blondiau, B. van Zeebroeck, and H. Haubold, "Economic benefits of increased cycling," *Transportation Research Procedia*, vol. 14, pp. 2306–2313, 2016. Transport Research Arena TRA2016.

[9] N. Stamatiadis, G. Pappalardo, and S. Cafiso, "Use of technology to improve bicycle mobility in smart cities," in *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 86–91, 2017.

[10] A.-S. O.M.F. and D. Z., "Greenbikenet: an intelligent mobile application with green wireless networking for cycling in smart cities," in *Mobile Networks and Applications*, p. 352–366, 2016.

[11] European Cyclists' Federation, "Towards a smarter cycling." https://ecf.com/what-we-do/cycling-new-technologies/towards-smarter-cycling, 2020. Accessed 2020-12-23.

[12] F. Behrendt, "Mobility and data: cycling the utopian internet of things," *Mobilities*, vol. 15, no. 1, pp. 81–105, 2020.

[13] F. Behrendt, "Why cycling matters for smart cities. internet of bicycles for intelligent transport," *Journal of Transport Geography*, vol. 56, pp. 157 – 164, 2016.

[14] H. Bouazza, L. Zohra, and B. Said, *Integration of Internet of Things and Social Network: Social IoT General Review*, pp. 312–324. 12 2019.

[15] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *Internet of Things Journal, IEEE*, vol. 1, 01 2012.

[16] M. Meireles and P. Ribeiro, "Digital platform/mobile app to boost cycling for the promotion of sustainable mobility in mid-sized starter cycling cities," *Sustainability*, vol. 12, p. 2064, 03 2020.

[17] A. Matta, K. Fritz, B. Kim, S.-J. Kim, A. Akhmouch, and P. Philip, "Smart cities and inclusive growth," in *OECD Roundtable on Smart Cities and Inclusive Growth*, 2020. 1st OECD Roundtable on Smart Cities and Inclusive Growth (9 July 2019, OECD Headquarters, Paris, France).

[18] R. Khatoun and S. Zeadally, "Smart cities: Concepts, architectures, research opportunities," *Commun. ACM*, vol. 59, p. 46–57, July 2016.

[19] United Nations, "2018 revision of world urbanization prospects." https://population.un.org/wup/Publications/Files/WUP2018-Report.pdf, 2018. Accessed 2020-12-23.

[20] I. Šemanjski, S. Mandžuka, and S. Gautama, "Smart mobility," in *2018 International Symposium ELMAR*, pp. 63–66, 2018.

[21] R. Arce-Ruiz, N. Baucells, and C. Moreno Alonso, "Smart mobility in smart cities," in *XII Congreso de Ingeniería del Transporte*, 06 2016.

[22] A. Nikolaeva, M. te Brömmelstroet, R. Raven, and J. Ranson, "Smart cycling futures: Charting a new terrain and moving towards a research agenda," *Journal of Transport Geography*, vol. 79, p. 102486, 2019.

[23] E. Trindade, M. Hinnig, E. Costa, J. Marques, R. Bastos, and T. Yigitcanlar, "Sustainable development of smart cities: A systematic review of the literature," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 3, 12 2017.

[24] R. P. Dameri, C. Benevolo, E. Veglianti, and Y. Li, "Understanding smart cities as a glocal strategy: A comparison between italy and china," *Technological Forecasting and Social Change*, vol. 142, pp. 26 – 41, 2019. Understanding Smart Cities: Innovation ecosystems, technological advancements, and societal challenges.

[25] F. Van Schaik, "Designing a mobile user interface for road cyclists," 2013.

[26] O. M. F. Abu-Sharkh and Z. Dabain, "GreenBikeNet: an Intelligent Mobile Application with Green Wireless Networking for Cycling in Smart Cities," *Mobile Networks and Applications*, vol. 21, no. 2, pp. 352–366, 2016.

[27] S. Torres, F. Lalanne, G. del Canto, F. Morales, J. Bustos-Jimenez, and P. Reyes, "Becity: sensing and sensibility on urban cycling for smarter cities," in *2015 34th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–4, 2015.

[28] C. Igwenagu, *Fundamentals of research methodology and data collection*. 04 2016.

[29] D. Lacerda, A. Dresch, A. Proença, and J. A. V. Antunes Júnior, "Design science research: A research method to production engineering," *Gestão & Produção*, vol. 20, pp. 741–761, 12 2012.

[30] P. Offermann, O. Levina, M. Schönherr, and U. Bub, "Outline of a design science research process," 01 2009.

[31] D. Young, "Software development methodologies," *White paper*, 08 2013.

[32] W. Van Casteren, "The waterfall model and the agile methodologies : A comparison by project characteristics - short," *White paper*, 02 2017.

[33] Rezaid, "Waterfall methodology." https://rezaid.co.uk/sdlc-waterfall-model, 2020. Accessed 2020-12-23.

[34] Devcom, "Agile methodology." https://devcom.com/how-we-work, 2020. Accessed 2020-12-23.

[35] C. Győrödi, R. Gyorodi, and R. Sotoc, "A comparative study of relational and non-relational database models in a web- based application," *International Journal of Advanced Computer Science and Applications*, vol. 6, 11 2015.

[36] "Mongodb atlas." https://www.mongodb.com/cloud/atlas, 2021. Accessed 2021-02-15.

[37] H. Sun, D. Bonetta, C. Humer, and W. Binder, "Efficient dynamic analysis for node.js," pp. 196–206, 02 2018.

[38] S. Tilkov and S. Vinoski, "Node.js: Using javascript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[39] "ios 14." https://www.apple.com/pt/ios/ios-14/, 2021. Accessed 2021-02-24.

[40] "Android." https://www.android.com/intl/pt_pt/, 2021. Accessed 2021-02-24.

[41] OPTASY, "What hybrid app development framework should you use for your projects in 2020? top 3," 2020.

[42] S. Souza, N. Anquetil, and K. Oliveira, "A study of the documentation essential to software maintenance," 01 2005.

[43] Dicio, "Requirement definition by dicio." https://www.dicio.com.br/requisito/, 2021. Accessed 2021-02-15.

[44] J. M. Fernandes and R. J. Machado, *Lecture Notes in Management and Industrial Engineering:: Requirements in Engineering Projects*. Springer, Cham, 2016.

[45] M. Bokhari and S. Siddiqui, "A comparative study of software requirements tools for secure software development," *BIJIT*, vol. July-December, 2010 Vol.2 No.2, 04 2010.

[46] S. Robertson and J. Robertson, *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley Professional, 3rd ed., 2012.

[47] I. Fortes, D. Pinto, J. Vieira, R. Pessoa, and R. José, "The perspective of cyclists on current practices with digital tools and envisioned services for urban cycling." 2021.

[48] "Visual paradigm." https://www.visual-paradigm.com, 2021. Accessed 2021-02-15.

[49] M. N. Alanazi, "Basic rules to build correct uml diagrams," in *2009 International Conference on New Trends in Information and Service Science*, pp. 72–76, 2009.

[50] R. Klimek and P. Szwed, "Formal analysis of use case diagrams," *Computer Science*, vol. 11, pp. 115–131, 01 2010.

[51] O. Nikiforova, J. Sejans, and A. Cernickins, "Role of uml class diagram in object-oriented software development," *J. Riga Technical University*, vol. 44, pp. 65–74, 01 2011.

[52] "Geojson." https://geojson.org/, 2021. Accessed 2021-02-22.

[53] "Npx framework." https://github.com/npm/npx, 2021. Accessed 2021-02-22.

[54] "Mongoose docs." https://mongoosejs.com/docs/, 2021. Accessed 2021-02-21.

[55] "Express framework." https://expressjs.com/, 2021. Accessed 2021-02-21.

[56] "Passport authentication in js." http://www.passportjs.org/, 2021. Accessed 2021-02-21.

[57] "Json web token packages." https://github.com/auth0/node-jsonwebtoken, 2021. Accessed 2021-02-21.

[58] "Sentry: Application monitoring and error tracking software." https://sentry.io/, 2021. Accessed 2021-02-27.

[59] "Swagger." https://swagger.io/, 2021. Accessed 2021-02-21.

[60] "Express swagger generator." https://github.com/pgroot/express-swagger-generator, 2021. Accessed 2021-02-21.

[61] "Authentication http." https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Authentication, 2021. Accessed 2021-02-23.

[62] "passport-jwt." http://www.passportjs.org/packages/passport-jwt/, 2021. Accessed 2021-02-23.

[63] "Docker." https://www.docker.com/, 2021. Accessed 2021-02-23.

[64] "Amazon ec2." https://aws.amazon.com/pt/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc, 2021. Accessed 2021-02-23.

[65] "React native." https://reactnative.dev/, 2021. Accessed 2021-02-24.

[66] "Xcode." https://developer.apple.com/xcode, 2021. Accessed 2021-02-24.

[67] "Gradle build tool." https://gradle.org/, 2021. Accessed 2021-02-24.

[68] "Apk definition." https://pt.wikipedia.org/wiki/APK, 2021. Accessed 2021-02-24.

[69] "Android manifest in xml." https://developer.android.com/guide/topics/manifest/manifest-intro, 2021. Accessed 2021-02-24.

[70] "Babel compiler." https://babeljs.io/, 2021. Accessed 2021-02-27.

[71] "React native background-geolocation." https://github.com/darron1217/react-native-background-geolocation, 2021. Accessed 2021-02-27.

[72] "React-native activity recognition." https://github.com/Aminoid/react-native-activity-recognition, 2021. Accessed 2021-02-27.

[73] "Mapbox." https://www.mapbox.com, 2021. Accessed 2021-02-27.

[74] "Mapbox maps sdk for react native." https://github.com/react-native-mapbox-gl/maps, 2021. Accessed 2021-02-27.

[75] "React-native geolocation service." https://github.com/Agontuk/react-native-geolocation-service, 2021. Accessed 2021-02-27.

[76] "React-native core components and apis." https://reactnative.dev/docs/components-and-apis, 2021. Accessed 2021-02-27.

[77] "Understanding the react native bridge concept." https://dev.to/mfrachet/understanding-the-react-native-bridge-concept-1k90, 2021. Accessed 2021-02-27.

[78] "Axios." https://github.com/axios/axios, 2021. Accessed 2021-02-27.

[79] "i18next." https://www.i18next.com/, 2021. Accessed 2021-02-27.

[80] "Redux." https://redux.js.org/, 2021. Accessed 2021-02-27.

A

RELATED APPS

A.1 APPLICATION PROTOTYPE - MINHA FREGUESIA

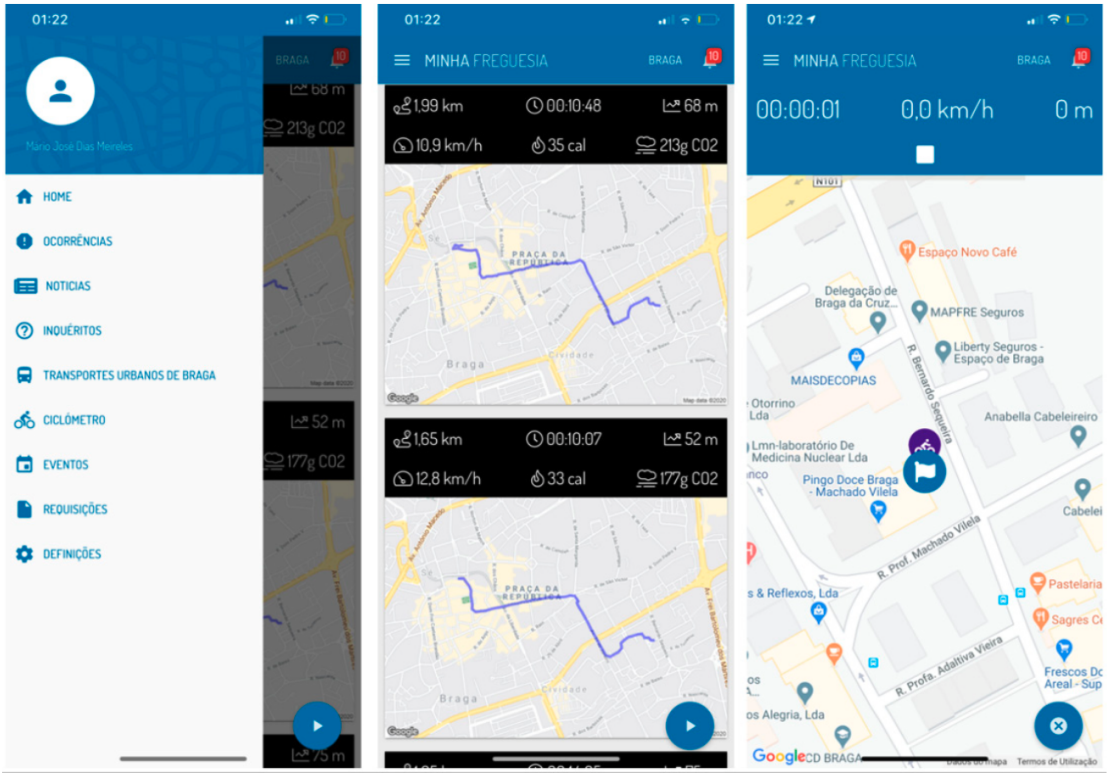


Figure 56: Application Prototype with the Bicycle Module
(Source: [16])

A.2 APPLICATION - SMART APP

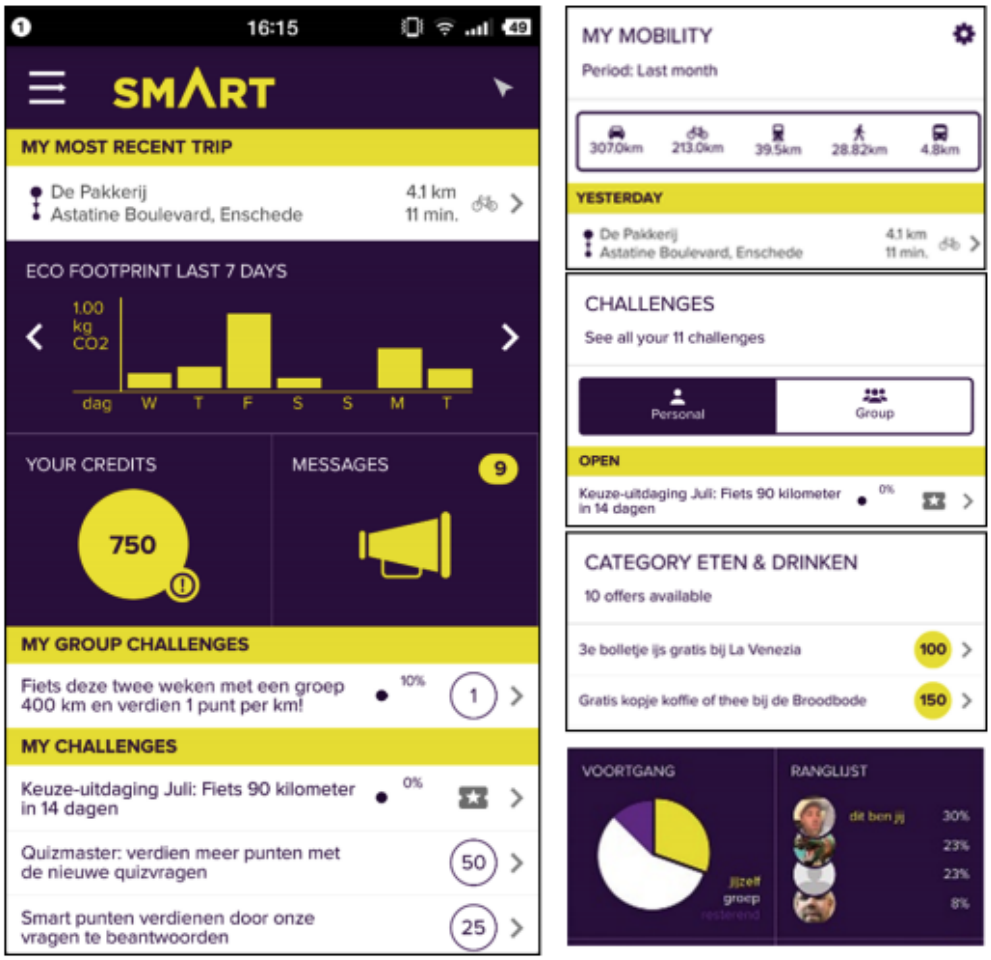


Figure 57: Application SMART app
(Source: [1])

A.3 APPLICATION - GREENBIKENET

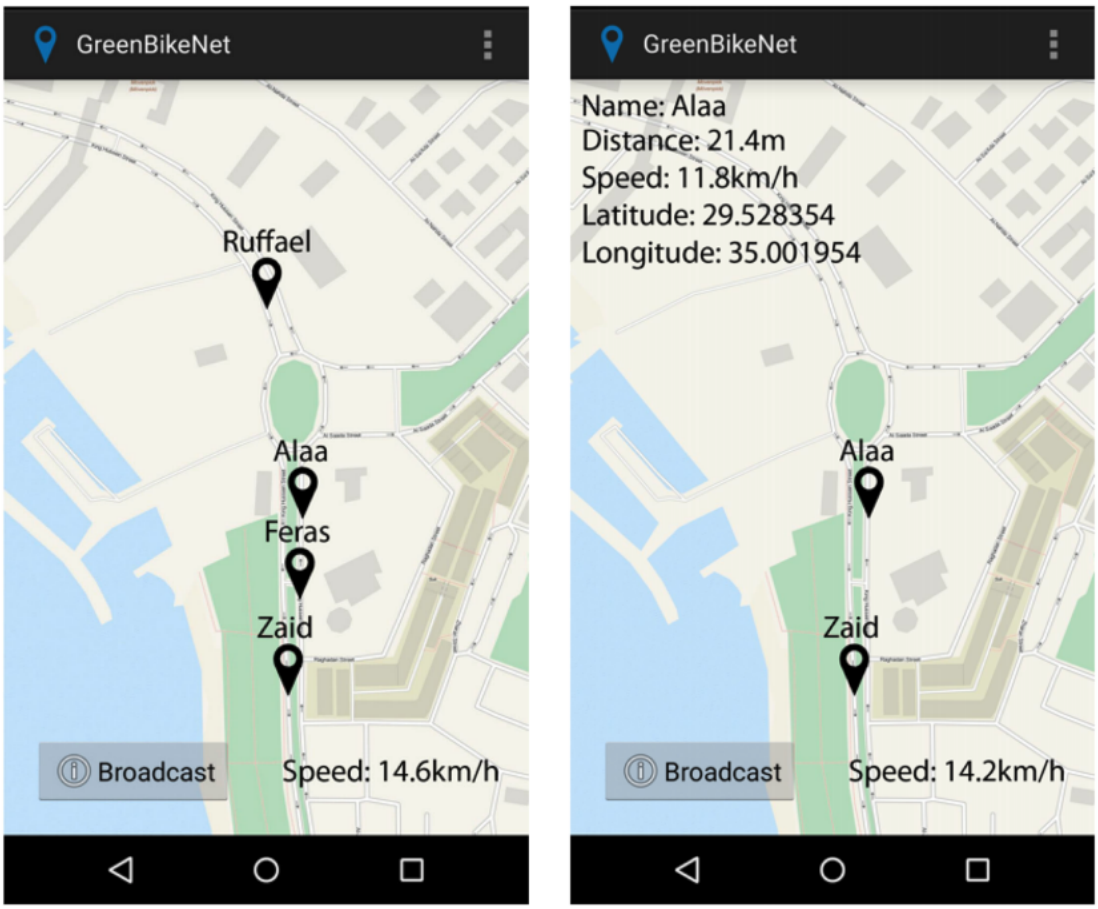


Figure 58: Application Prototype GreenBikeNet
(Source: [26])

A.4 APPLICATION - BECITY

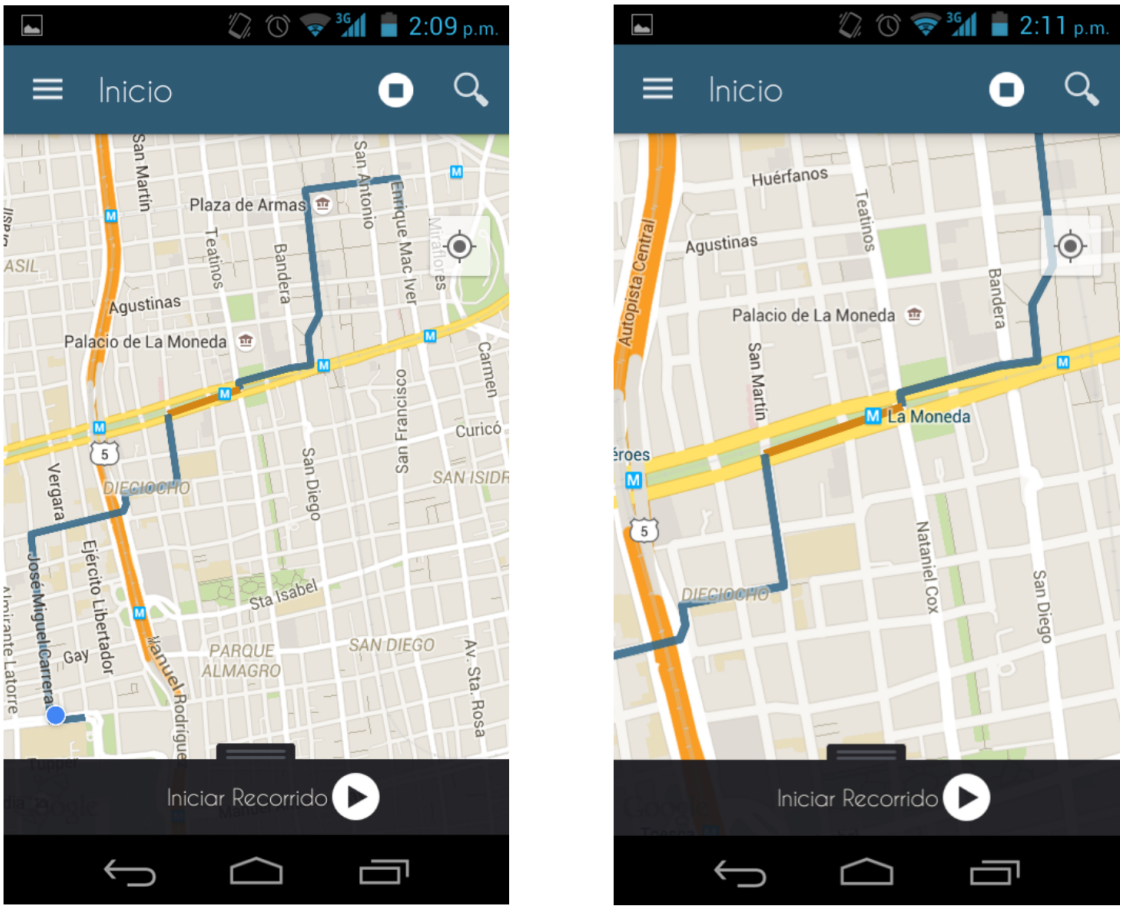


Figure 59: Application Prototype BeCity
(Source: [27])

B

DIAGRAMS

B.1 DOMAIN MODEL

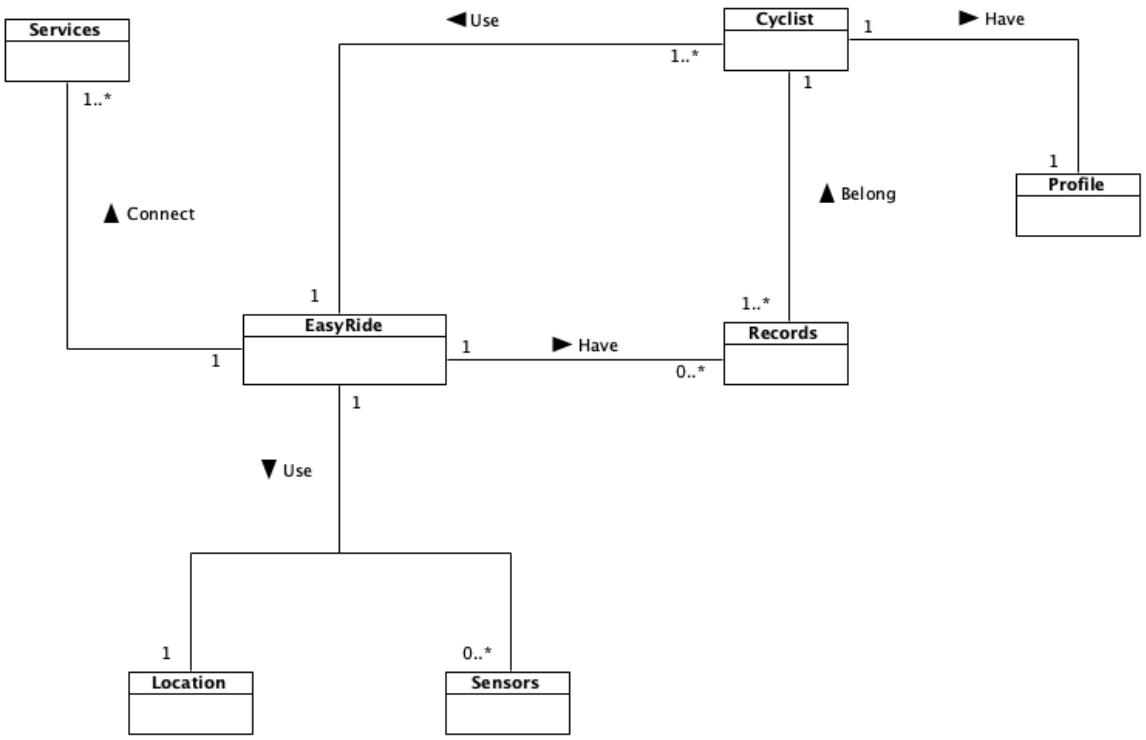


Figure 60: Domain Model
(Source: Own)

C

DOCUMENTATION

C.1 SWAGGER DOCUMENTATION

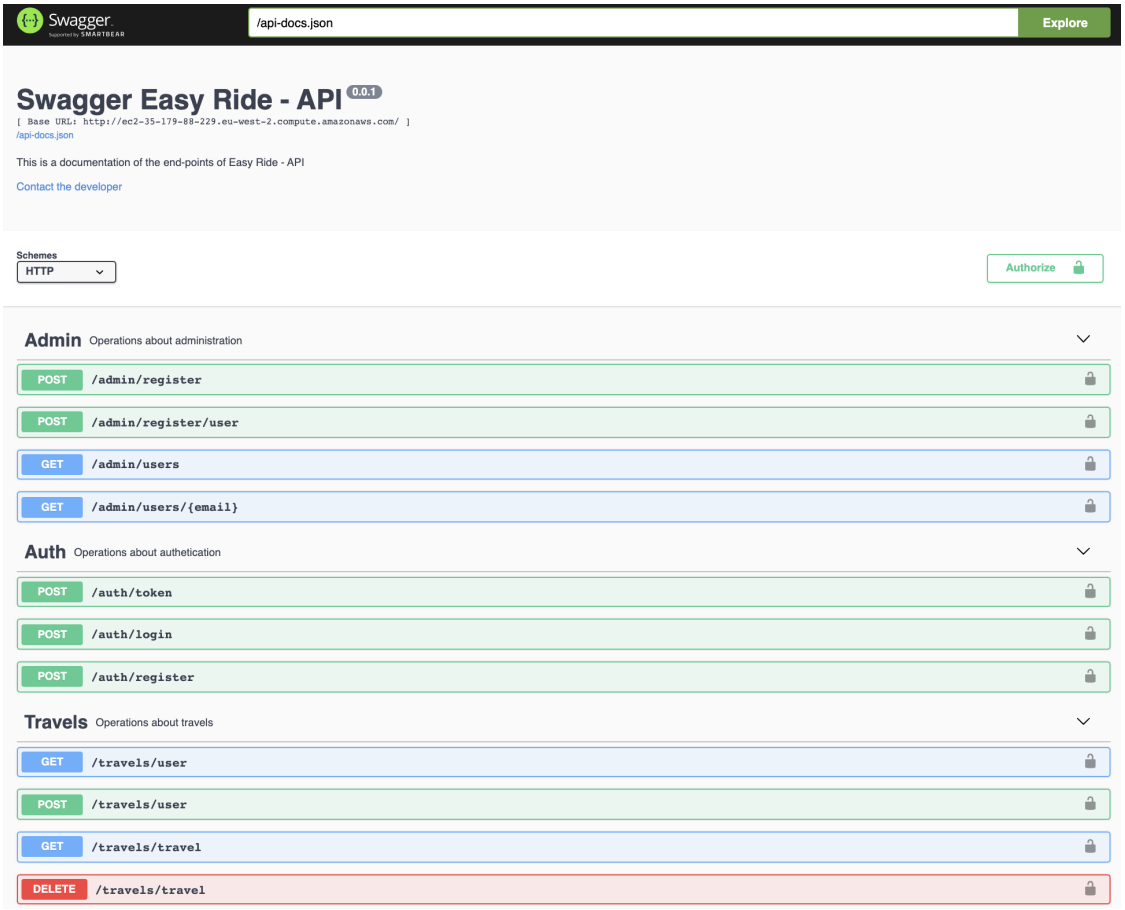


Figure 61: Swagger documentation (excerpt 1)
(Source: (Own))

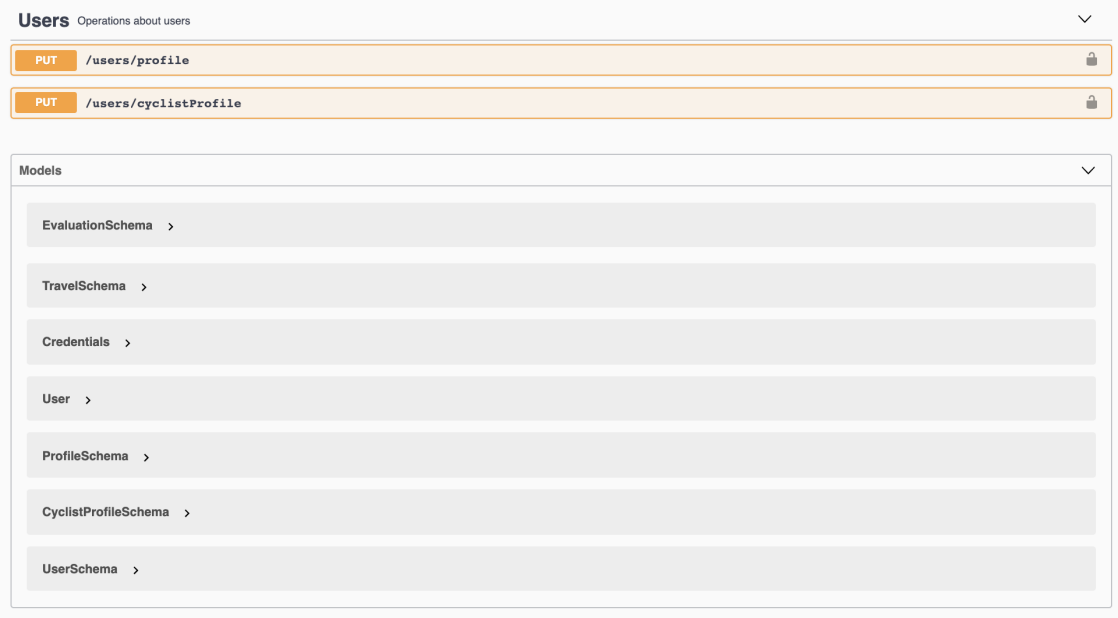


Figure 62: Swagger documentation (excerpt 2)
(Source: (Own))

D

LISTINGS

D.1 BACK-END FUNCTIONS

D.1.1 *controllers folder functions*

```
const Joi = require('@hapi/joi');
/* Function to validate user */
const validateUser = (user) => {
    const schema = Joi.object({
        name: Joi.string()
                  .pattern(/^[a-zA-Z]+(([ ][a-zA-Z ])?[a-zA-Z]*)*$/)
                  .min(3)
                  .max(45)
                  .required(),
        email: Joi.string()
                  .email({ minDomainSegments: 2 })
                  .required(),
        password: Joi.string()
                     .pattern(/[a-zA-Z]|[\-_@#!*]|[0-9]$/)
                     .required(),
        role: Joi.string()
                  .pattern(/(OWNER|CYCLIST|ADMIN)$/)
                  .required(),
    }).unknown(true)
    return schema.validate(user)
}
```

Listing D.1: JavaScript - Function that validate a user

```
const Joi = require('@hapi/joi');
const Travel = require('../models/travel');
...
/* Function to save Travel */
const saveTravel = async (travel) => {
    const newTravel = new Travel(travel);
    return newTravel.save();
}
```

Listing D.2: JavaScript - Function that save a travel

D.1.2 *models folder functions*

```
const mongoose = require('mongoose');
...
const ProfileSchema = new mongoose.Schema(
  {
    city: {
      type: String,
      min: 6,
      max: 255
    },
    gender: {
      type: String,
    },
    age: {
      type: Number,
    },
  },
  {
    _id: false
  }
);
```

Listing D.3: JavaScript - Profile schema - file *user.js*

```
const bcrypt = require('bcrypt');
...
UserSchema.methods.isValidPassword = async function (password) {
  const user = this;
  const compare = await bcrypt.compare(password, user.password);
  return compare;
}
```

Listing D.4: JavaScript - Function to compare password - file *user.js*

```
const mongoose = require('mongoose');
const TravelSchema = new mongoose.Schema({
  location: {
    type: Object,
    required: true,
  },
  duration: {
    type: Object,
    required: true,
  },
  distance: {
    type: Number,
    required: true,
  },
  co2Saved: {
    type: Number,
    required: true,
  },
  averageSpeed: {
    type: Number,
    required: true,
  },
  routeSpeed: {
    type: [Number],
    required: true
  },
  routeCoordinates: {
    type: [Object],
    required: true
  },
  idUser: {
    type: String,
    required: true
```

```
  },
  date: {
    type: Date,
    required: true
  },
  evaluation: {
    type: EvaluationSchema,
    default: {}
  }
}, { versionKey: false });
module.exports = mongoose.model('Travel', TravelSchema, 'travels');
```

Listing D.5: JavaScript - Travel schema - file *travel.js*

D.1.3 *routes folder functions*

```
router.post('/register', async (req, res) => {
    if(req.body.role === 'ADMIN' || req.body.role === 'OWNER') return res.sendStatus
        (401);
    const user = {
        name: req.body.name,
        email: req.body.email,
        password: req.body.password,
        role: req.body.role,
    }
    const validUser = User.validateUser(user);
    if (validUser.error) {
        return res.status(400).jsonp({ message: validUser.error.details[0].message });
    }
    const userAux = await User.consultUser(req.body.email);
    if(!userAux) {
        try {
            const result =  await User.saveUser(user);
            return res.jsonp({ message: 'Success on saving User.' });
        } catch (error) {
            return res.status(500).jsonp({ message: 'Error in saving User.' });
        }
    }
    return res.status(409).jsonp({ message: 'Email already in use.' });
});
```

Listing D.6: JavaScript - Route POST **/auth/register** - file *auth.js*

```
router.put('/profile', async (req, res) => {
  try {
    if(req.body) {
      const valid = User.validateUserUpdate(req.body);
      if (valid.error) {
        console.log(valid.error.details[0].message)
        return res.status(400).jsonp({ message: valid.error.details[0].message });
      }
      const result = await User.updateUser(req.body,req.user._id);
      let user = await User.consultUserID(req.user._id);
      delete user.password;
      user.password = undefined;
      const token = jwt.sign({ user }, authConfig.secretKey, { expiresIn: '90 days' })
          ;
      req.user.token = token;
      req.session.token = token;
      req.session.save();
      return res.jsonp({
        token,
        user
      });
    }
    return res.status(400).jsonp({ message: 'Empty request.' })
  } catch (error) {
    return res.status(500).jsonp({ message: 'Error on saving User Profile.' })
  }
});
```

Listing D.7: JavaScript - Route PUT **/users/profile** - file *users.js*

D.1.4 *Docker related code*

```
version: '3.4'
services:
  easyride-backend:
    image: easyride-backend
    build:
      context: .
      dockerfile: ./Dockerfile
    environment:
      NODE_ENV: production
    ports:
      - 5001:5001
    volumes:
      - .:/usr/src/app
      - /usr/src/app/node_modules
    container_name: easyride-backend
```

Listing D.8: docker-compose file that was use

D.2 FRONT-END FUNCTIONS

D.2.1 *store folder related code*

```
const LOGIN = 'LOGIN';
...
const ADD_TRAVEL = 'ADD_TRAVEL';
...
export {
    LOGIN,
    ...
    ADD_TRAVEL,
    ...
}
```

Listing D.9: Example of *actionTypes.js* file that was use

```
import {
    LOGIN
    ...
} from '../actions/actionTypes';
```

```
...
const initialState = {
    token: undefined,
    user: undefined,
}

export default function authReducer(state = initialState, action) {
    switch (action.type) {
        case LOGIN:
            return {
                ...state,
                token: action.payload.token,
                user: action.payload.user,
            };
            break;
        ...
    }
}
```

Listing D.10: Example of *authReducer.js* file that was use

```
import { combineReducers } from 'redux';
import { reducer as network } from 'react-native-offline';

import storeReducer from './storeReducer';
import authReducer from './authReducer';

export default combineReducers({
    storeReducer,
    authReducer,
    network
});
```

Listing D.11: Example of *rootReducer.js* file that was use

D.2.2 *App.js excerpt code*

```
import React, { Component } from 'react';
import { Provider } from 'react-redux';
import { PersistGate } from 'redux-persist/integration/react'
import { ModalPortal } from 'react-native-modals';
import { ReduxNetworkProvider } from 'react-native-offline';
```

```
import * as Sentry from '@sentry/react-native';
...
import { store, persistor } from 'store';
...
/* Config Sentry only in production */
if(!__DEV__) {
  Sentry.init({
    dsn: 'https://a80a4419d351446a9d03fe549e34fa48@o536457.ingest.sentry.io/5655080',
    enableAutoSessionTracking: true,
  });
}
...
import AppContainer from 'navigation';
import WifiNotice from 'components/WifiNotice';
import ActionButtonCustom from 'components/ActionButtonCustom';
import OnlyTracking from 'components/OnlyTracking';
...
class App extends Component {
  constructor(props) {
    super(props);
  }
  ...
  render() {
    return (
      <Provider store={store}>
          <PersistGate
            persistor={persistor}
            loading={null}
          >
            <ReduxNetworkProvider pingInBackground={true}>
              <StatusBar backgroundColor='#ffffff' barStyle='dark-content'/>
              <AppContainer
                onNavigationStateChange={(prevState, newState) => {
                  this._getCurrentRouteName(newState);
                }} />
              <ModalPortal />
              <WifiNotice />
              <ActionButtonCustom />
              <OnlyTracking />
            </ReduxNetworkProvider>
        </PersistGate>
      </Provider>
    );
```

```
  }
}
export default withNamespaces()(App);
```

Listing D.12: Example the *App.js* file

D.2.3 *i18n related code*

```
...
const getLanguageByDevice = () => {
  const language = Platform.OS === 'ios'
    ? NativeModules.SettingsManager.settings.AppleLocale ||
      NativeModules.SettingsManager.settings.AppleLanguages[0]
    : NativeModules.I18nManager.localeIdentifier;
  return normalizeTranslate[language];
}
const languageDetector = {
  type: 'languageDetector',
  async: true,
  init: () => { },
  detect: async (callback) => {
    const lng = await AsyncStorage.getItem(STORAGE_KEY);
    if(lng) {
      return callback(lng);
    } else {
      const lang = getLanguageByDevice();
      AsyncStorage.setItem('language', lang);
      return callback(lang);
    }
  },
  cacheUserLanguage: () => { },
};
..
i18n
  .use(languageDetector)
  .use(reactI18nextModule)
  .init({
    initImmediate: false,
    resources: resources,
    react: {
      wait: true
    },
```

```
    fallbackLng: 'en',
    keySeparator: '.',
    ns: ['common'],
    defaultNS: 'common',
    debug: true,
    interpolation: {
      escapeValue: false
    }
  });
```

Listing D.13: Example the *index.js* file on *i18n* folder

D.2.4 *axios related code*

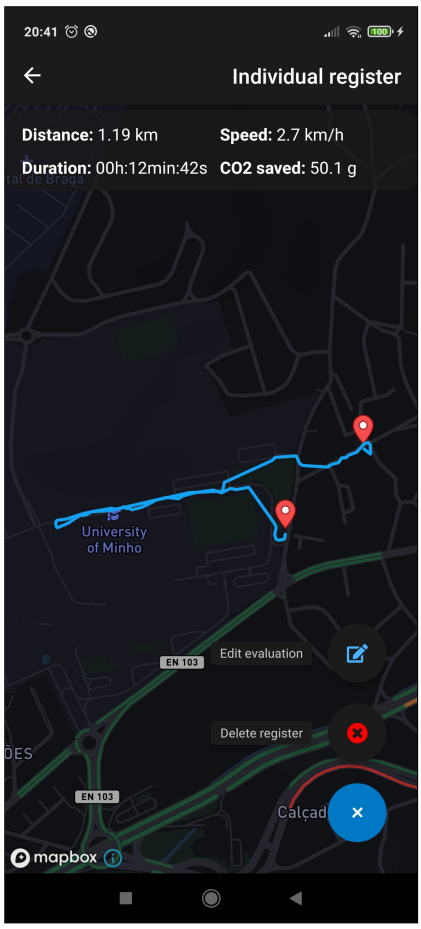
```
import axios from 'axios';

const configAxios = () => {
    if(!__DEV__) {
        axios.defaults.baseURL = 'http://ec2-35-179-88-229.eu-west-2.compute.amazonaws
            .com';
    } else {
        axios.defaults.baseURL = 'http://192.168.1.165:5001';
    }
    axios.defaults.headers.common['Content-Type'] = 'application/json;charset=UTF-8';
}

module.exports = {
    configAxios,
}
```

Listing D.14: Example the *axios* configuration

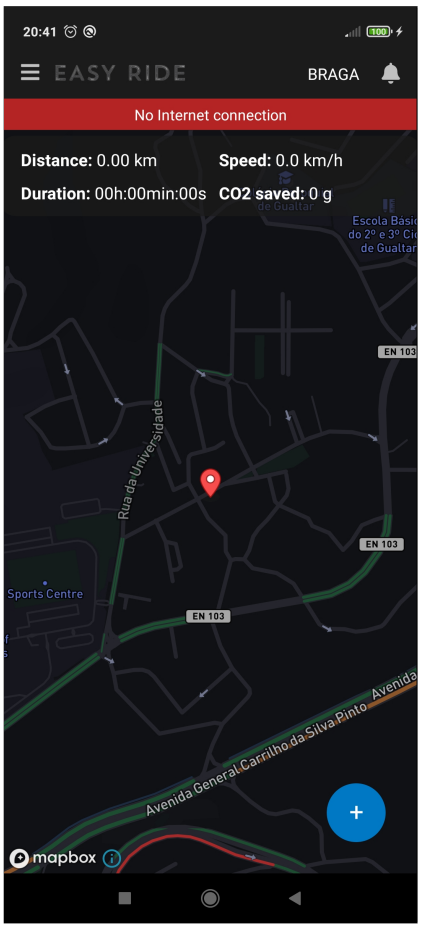
E

INTERFACES

E.1 MAPS SCREENS



(a) Night mode individual register



(b) Tracking screen night mode

Figure 63: Maps screens

E.2 ALERTS

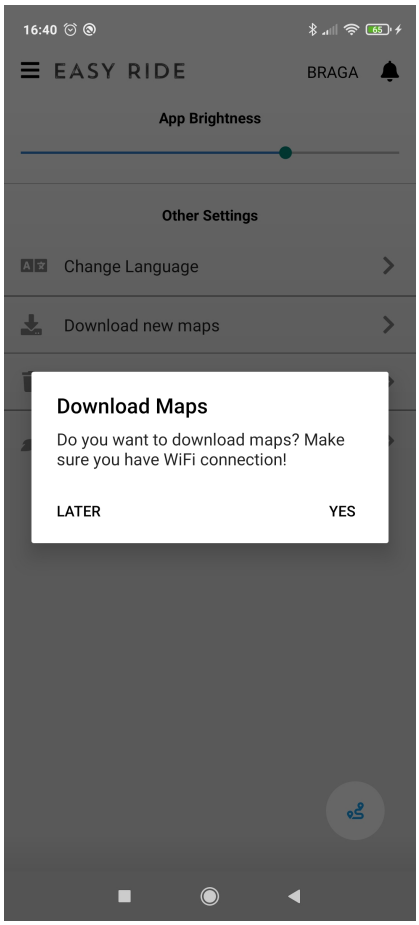
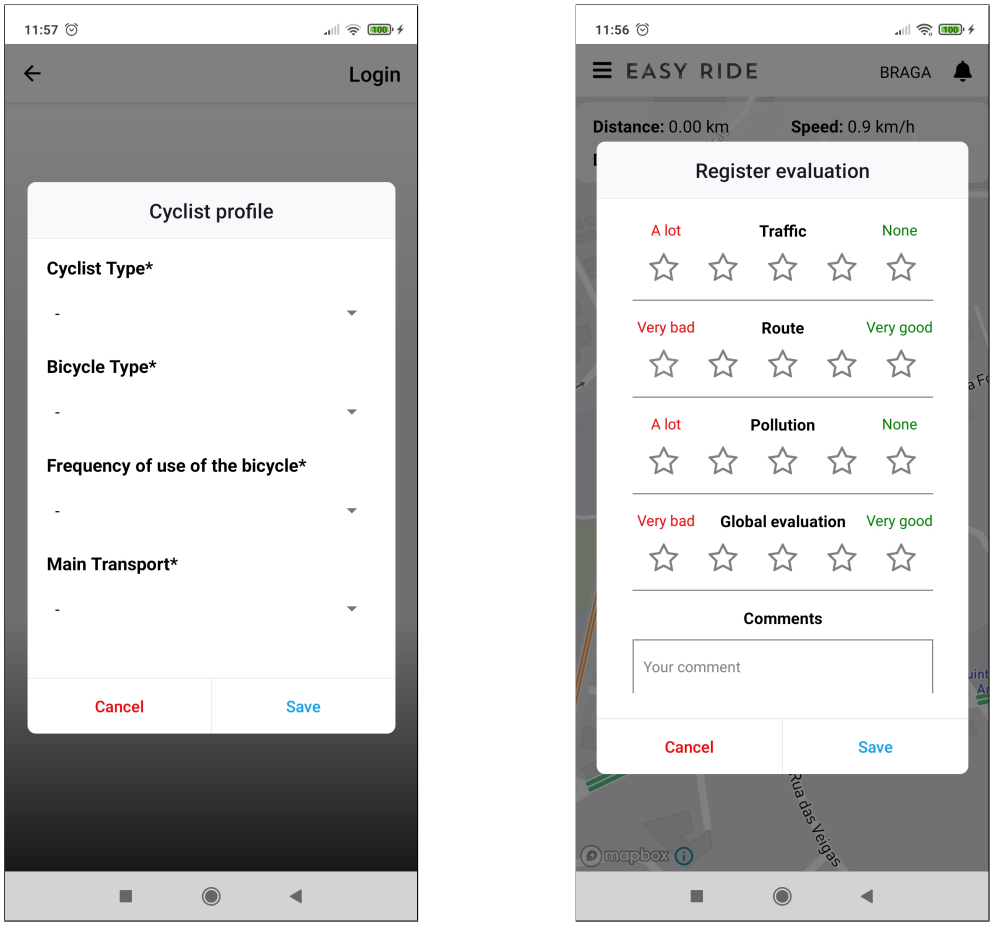


Figure 64: Simple alert

E.3 MODALS



(a) Cyclist profile modal

(b) Travel evaluation modal

Figure 65: Modals