Ivo Miguel da Cunha Sousa

Analysis and Improvement of Urban Mobility Platform Based on Smartphones

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
June 2020

Ivo Miguel da Cunha Sousa

Analysis and Improvement of Urban Mobility Platform Based on Smartphones

Masters Thesis

Supervisor: Michel Celestino Paiva Ferreira

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
June 2020

Abstract

Taxi-Link is the main electronic platform that works with taxis in Portugal, and in an era where mobility platforms have been growing exponentially redefining door to door mobility, it is important to improve our platform in order to not fall behind. Our aim is to study these platforms and, build a passenger application. We will also try to improve the platform as a whole with the generated data. A very important step will also be to develop ways to migrate clients that use phone calls to request taxi services to start using the passenger application instead.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the past years, mobility platforms such as Uber, Lyft, Free Now, among others, have been growing exponentially redefining door to door mobility in a great number of cities throughout the world[16]. These platforms generate a huge amount of geo-referenced data that allows the study of driver's behavior and passenger's mobility profiles. The final goal of these platforms is to connect a passenger and a driver application, based essentially on location but also making use of other data, such as evaluations or other information made available by both drivers and passengers.

In this thesis, we proposed to analyze and improve the main electronic platform that works exclusively with taxis in Portugal, Taxi-Link which is partnered with more than 4000 taxi drivers and has more than 2 million users.

Taxi-Link has a taxi dispatching system which is used in partnership with many taxi fleets all over Portugal, but it was missing a passenger application which would bring great value to the platform as a whole and help to improve the overall service.

In this thesis, our main objective was to build that passenger application and, try to collect data that would be valuable in further analysis of the platform allowing us to improve it as a whole. Another objective that was also very important was to investigate and develop techniques on how to migrate clients that request taxis by phone call to use the smartphone application paradigm instead.

This thesis is organized as follows: we have got seven chapters. First, we will talk about the history of the taxi industry explaining how taxi calling and dispatching techniques have been evolving over the years. Then we will analyze some of the main electronic mobility platforms to learn how they operate, learn what we should do and find ways to differentiate ourselves from them. Then we will explain the Taxi-Link architecture to give context on how the platform works as a whole. After this we will explain how we built the passenger application, explaining what we needed from it, how we chose our development tool and, explaining how we used the Google Maps API (Application Programming Interface) to our advantage. Then follows the migration chapter where we explain the phone call to smartphone application migration techniques used and present an analysis of the results achieved by those techniques. Following that up we have got a discussion where we present challenges we faced during the process of developing this thesis, discuss a bit the migration results, and present ways in which we could improve the platform further in the future. And to finalize, we give our conclusion about this project.

Chapter 2

State of the Art

2.1 Taxi Transportation

Taxis are a means of transportation that is based on employing a vehicle with a driver that can be used by a single passenger or a small group of passengers. This type of transportation offers great flexibility on routes, being able to provide clients with a trip from a point A to a point B of their choice, as opposed to other public transportation methods like buses or subway which offer only fixed routes. This method of transportation differs from country to country, sometimes even differs between cities, on aspects like type of cars, tariffs, regulation, dispatching, payment methods and others.

2.2 Evolution of Taxi Calling and Dispatching Techniques

Taxi calling and especially dispatching techniques have evolved substantially in the past years. An important step for **taxi services dispatching** was the use of **call boxes**[18]. A call box is a box containing a special-purpose direct line phone (see figure 2.1) or other telecommunication devices which can be used by different services. Since taxis would not need to be in a dispatch center to answer services from phone calls, this had a great impact. It allowed taxi companies to distribute groups of taxis throughout different regions in what are called taxi stands, instead of having them stay at the dispatch center to provide these services. In consequence, clients would be served quicker. This is a big advantage of having a distributed fleet instead of having it centralized in one place. Call boxes would then be placed at the taxi stands, and when the dispatch operator had a service he would first try to determine which stand would be the best to serve it, and then would contact said stands call box. To determine which taxi would get to serve the dispatched service, drivers would queue up by order of arrival, and the first one to have arrived to the stand would get the service.

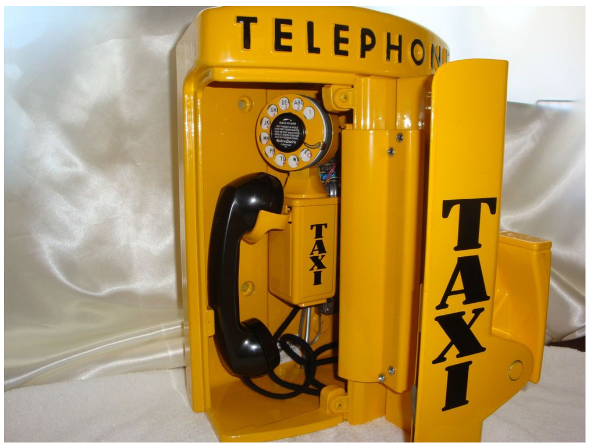


Figure 2.1: Taxi call box [1]

After came the **radio-to-radio communication**. This allowed each taxi to have a radio, which allowed them to be contacted by the dispatch central at anytime, even when cruising. Although it was an improvement for service delivery, it was still limited to the number of channels available. This caused the zone distribution to be restricted by the number of channels assigned to the dispatch company. When a service was received by the dispatch center, the operator would have decide to which channel he should relate the service to, and would then contact said channel via their own radio. Then to select which taxi got the service, instead of the first taxi to arrive to a certain zone, as with the call boxes, it was given to the first one to answer the radio. With this system, taxi drivers had to manually switch channels when crossing to a different zone according to their location. The manual switching of channels was a negative point, since taxi drivers had to distract themselves when switching, which occasionally happened while travelling, even with passengers in, and would cause a decrease in safety.

Nowadays we have **electronic dispatch systems**. With Global Positioning System

(GPS) receivers and internet access both becoming increasingly common in many cities, with it a new way to dispatch taxis has emerged. It is now possible for taxis to automatically inform an electronic system of their current GPS position and status. Which means it is possible to build such a system that uses that information to manage the dispatching in an optimal way. This increases efficiency since automated systems are quicker in decision making and have the ability of making better choices, assigning services to taxis or areas optimally. By radio dispatch, the services were broadcasted to a channel and the first taxi to respond would get it, as said before, even despite the possibility of not being the best fit for said service. This increase also helped to reduce waiting times and increased the number of fulfilled requests. With electronic systems, service information that had to be relayed to the drivers, such as the service details, can be sent via data communications and shown in a receiver device with an electronic display. This new way of relaying information eliminated the problem of voice communication which created a lot of noise, being disturbing for the passengers, and that would lead to misunderstandings which in turn also impacted pickup times. This also contributed to improve efficiency, allowing taxi drivers to complete more services daily[19].

Figure 2.2: Radio box [2]

Regarding **taxi calling** techniques, the most common manner of calling a taxi used to be by **phone call**, with the exception of hailing a taxi in the street. Even though phone calls are still a reliable way of getting a taxi, there are issues such as line availability, the reason being that each dispatch center has a fixed number of lines that limit how many calls can be held simultaneously, sometimes the sound is unclear and even with clear sound the clients instructions might not be readily understood by the phone operator. All these factors delay the requesting process, which ultimately delays the service as a whole. In the past years, akin to the dispatch methods, there as been a surge of **electronic platforms** that also allow clients to request taxis, be it via a website or a mobile application, making it so that clients are not restrained by the limitations of the phone service, and openening a great deal of opportunities for companies that implement this system. If clients have accounts you can give them a set of default and quick access options, such as default payment method or a set of common destinations, improving usability and simplifying the process of requesting a taxi for their clients.

2.3 Electronic Platforms

With the rise of technology, there has also been an increase in the number of electronic platforms for private transportation services, in this section we will analyze some of those platforms passenger applications to know what features some of these have, to understand what is available and what is being done by those platforms.
From all of the electronic platforms available, we mainly focused on Free Now, since their main focus is on taxis and they also operate in Portugal, we also focused on Uber since nowadays they are one of the biggest companies worldwide and one of the main non-taxi private passenger transportation services available in Portugal [17].

In our analysis of both of these applications we found that they have many common features, it is normal since the flow of information for the requesting process in this kind of applications has the same set of core steps, such as picking the route, selecting car characteristics and finally service tracking as it is happening. We will enumerate the features we actually looked at closer and which influenced our application, and while doing that we'll talk about which problems each of those features resolves.

Both applications require their clients to register an account, after that clients login into their account, and are presented with a home screen. In the home screen the client is presented with a map and the application will try to get the clients location (using any location system available by your smartphone Operating System (OS)), having acquired the clients location the application will center the map on it, indicating this location with a marker. Still in the home screen, Free Now allows the client to visually choose the pickup point by dragging the marker, and while Uber also gives you that option later in the process, at first it locks the map in the clients location, assuming they want the car to meet them at their location. We felt these were great approaches to tackle this problem, because clients usually request the taxis in the moment they need it which means they are usually near the desired pickup location. Sometimes they might not even know where they are located, so having the map pin-pointing their location resolves that problem, simplifying the pickup location choice.

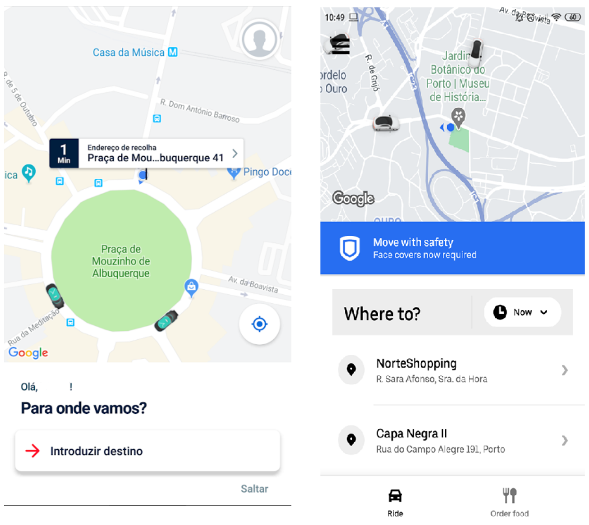


Figure 2.3: Screenshots from both the Free Now and Uber passenger applications

There are also clients that are not near the pickup point and would prefer to insert it by text. Here both applications use an auto complete service with geological information. This feature also works for destination picking, which is its main use. When using it, as the user is writing the desired address into a text prompt these services provide suggestions based on that input. The suggested addresses are part of an object that also contains the a pair of coordinates corresponding to the address. These coordinates are the most useful information for the system since route calculation and subsequent navigation are based on geographical coordinates. Regarding destination picking, Free Now which works with taxis also has the interesting option of skipping the destination location pick. This is because it is still usual for taxis to get directions directly from the client in the car, and for some clients it might be the desired way of making a request, since it helps to speed up the process.

After the route selection, clients are presented with a suggested route (drivers might take a different route when performing the service) and several car type options where they can choose the one that fits their needs or preferences. These options can be

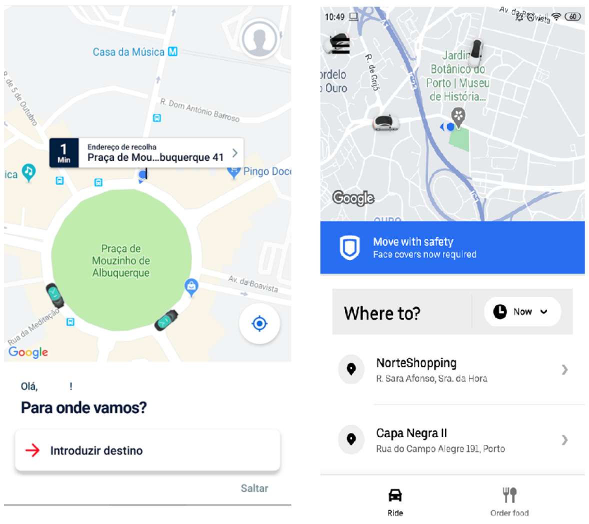


Figure 2.4: Screenshots from both the Free Now and Uber passenger applications

a car with cheaper fare, with a specific number of seats, with handicap support, an ecologic car, among others. These depend on car availability for each platform and region. Each of these options has the corresponding service cost and also an estimate of the pickup time attached to them, which is important information for the user when making the request. In terms of cost, it varies between different platforms, Uber for example presents the client with a static cost before you call the car, while Free Now only presents a range, presenting the actual cost only after the trip ends, except for their lite option which does have a static fare.

When the clients select the type of car they want, they may request the service. Following the request they are presented with a waiting for assignment screen, which is then followed by a pickup screen where they can track the car picking them up. In the pickup screen, clients are provided with information about the car such as model and plate, in junction with information about the driver such as their name and a picture, so they can identify the car performing the service. After being picked up the clients are also able to track the trip on a map. Finally, after the service has terminated they get a screen with a summary of the service and the opportunity to evaluate it. Service evaluations are very interesting for the platforms as it gives them information about user satisfaction and helps them to build the drivers profiles.
There are also some minor features such as the option of defining their home and work addresses which help as quick access addresses when choosing locations and both applications also have a service history, which is also very useful so that clients can keep track of their services.

Chapter 3

Taxi-Link(s) Electronic Expedition System Architecture

3.1 Vehicle Location and GPS

With the surge of GPS devices a great opportunity arised, it was now possible to know with precision where a certain taxi (or anything with a GPS receiver) was located. The GPS system belongs to the United States of America (USA) and is free to use, it is comprised of a total of twenty four satellites orbiting the Earth disposed in such a way that there are always five in view from every point on Earth. This location information was great in this context and led to numerous improvements in many different systems, such as the dispatch system, and had other applications, for instance, with the help of an analysis of mobility patterns it was possible to optimize the distribution of taxis throughout a region. To get a taxi's position taxis must be equipped with a GPS receiver device, this device receives and sends signals to the satellites orbiting the Earth and, to calculate a GPS receiver's location a technique called trilateration is used. Receivers need to be able to communicate with at least three satellites so that this technique can work. When a GPS receiver gets a signal from a satellite, the distance between them is measured by the time it took for the signal to travel. Using the distance to those three satellites the receiver is able to determine its exact location, which is based on the satellites positions themselves[15].

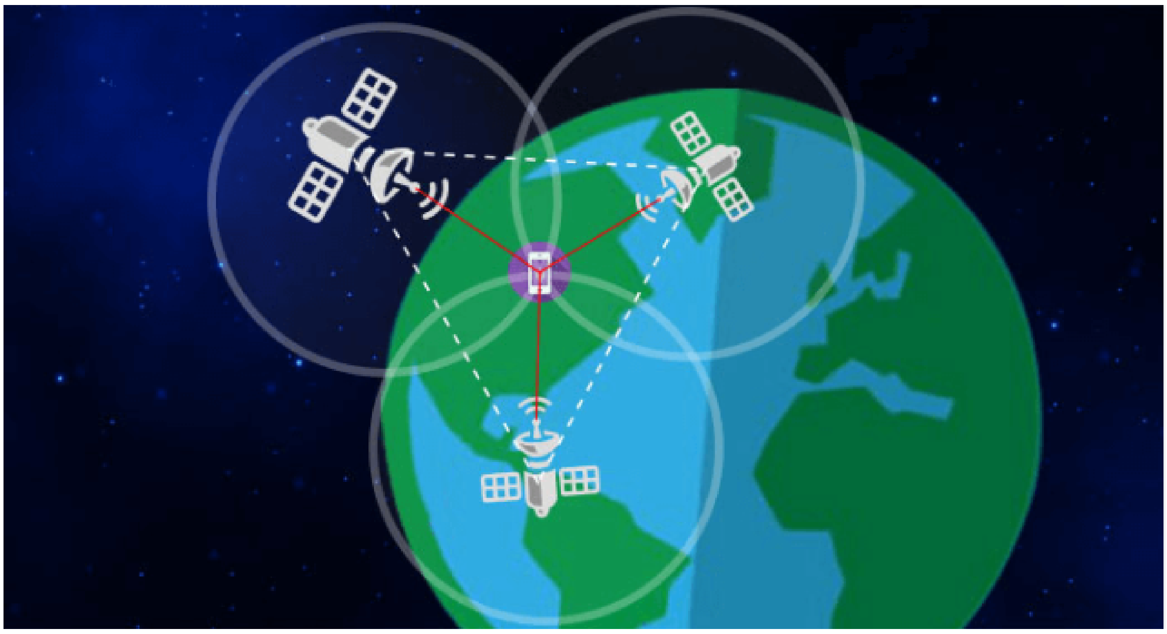


Figure 3.1: GPS Trilateration [9]

3.2 Vectorial Cartography, Toponomy and Route Calculation

In order to build our dispatch system we needed a complementary system that would allow us to query geological data. Its implementation would need to provide the ability to make efficient decisions and provide accurate information through the dispatch system. To fulfill this we acquired a cartography system containing this type of information and functions that complement our objectives. This system uses the following types of objects to represent a map, the main ones are the segments of road, these segments are comprised of two sets of coordinates that identify where and how the segment extends in the map, they also have information of which segments they are connected to and, the directions where it is possible to travel on themselves. These segments represent all the roads (represented by a set of segments) where taxi transit is allowed, including bus lanes. This system also includes POIs (Points Of Interest) that represent a specific geographic point, these are used to mark certain points in the map such as restaurants, shoppings, police stations, hospitals, etc.
The system has a reverse geocoding feature. It works by approximating a given coordinate to a segment in the map by finding its closest point considering all the points that comprise the segments. This feature is useful because it converts a coordinate like those given by a GPS enabled device in a point with meaning in the segment map,

without the need of human processing.
Another feature this system provides is a geocoding functionality. It works in the following way, when it is given a set of words a search is made to the database to try to match those words to a certain location, and each query returns a set of best results. The most usual use case in our system is when a phone operator is answering a clients call and inputs the addresses the client indicates into the system, it is also important that the phone operator has understood to which region (in our case municipality) that address belongs to, since adding that information to the query will narrow down the query results, making them more accurate.
Route calculation is another function the system provides. When it receives two sets of coordinates as input, the system will try to approximate each of those two sets to the best corresponding road segments (using reverse geocoding), then it uses a search algorithm to find the best set of steps (comprised of road segments) that link both segments. This allows us to do route calculation, which is very important when deciding which is the best taxi to assign to a certain request, as we will explain in the next section.

3.3 Vehicle Selection, Taxi Stands and Zoning

In order to decide to which taxi to assign to a given service there are several criteria that guide said decision. The first step is to filter the taxis that meet the criteria of the client, be it number of seats or other characteristics such as having air conditioner, an ATM terminal, etc. From that selection of vehicles we then have to check which taxi stands are close to the chosen pickup point have those kinds of taxis available and decide to which taxi the service is going to be assigned to. After selecting a taxi stand, the next step is deciding to which taxi the service should be granted. In taxi stands there is the concept of stand position, this means that taxis that have been parked in that stand for a longer time have priority in receiving services. This being said, the service is then offered to the taxi with highest priority. It is possible for him to reject it, in which case the service will be offered to the next taxi with highest priority. This system tries to make it so that taxis do not go too long without completing a service. It is important to note that while this is the case for most cities, there are cities in which the rules are such that service attribution does not take in consideration the time taxis have been without service and always sends the one that is the closest. (figure 3.2)
While the dispatch method above only offers the service to one taxi at a time, there is

also another system for offering services available in our infrastructure. In this method while choosing eligible taxis works in the same way as the single offer one, the service is offered not to one but many taxis at a time, having no limit in the number of taxis to which the service is offered. The taxis themselves get a prompt to apply for said service. After X seconds our server will check which taxis applied for assignment and will decide which one is best to execute that service. Here the metric that determines which taxi should get the service is the distance to the client, the distance for each taxi is determined by a route that is calculated by our server (section 3.2), and the taxi with the shortest route is the one that gets assigned to the service. Here, again, we do not assign priority to taxis with more time without completing a service, but this approach allows us to provide a better service for the client. Since the taxi with the shortest route is the one assigned, the pickup time is guaranteed to be the best (not considering traffic information). Also with this method the time to assign a taxi is consistent, as opposed to the single taxi offer, because in that model there is a possibility where a service can be rejected by many taxis and be bounced around, sometimes taking up to ten minutes to assign a taxi to a service, when usually it takes less than one minute. This difference is because in this offer system there is a much lower chance that a service will not be assigned to a taxi in the first offer round since it already offers the service to a number of taxis, rather than one at a time. It is important to note that the reason why the first method is still used for some fleets is that the multiple taxi offer was only introduced in a later version of the platform and many drivers have difficulties adapting to new methods. (figure 3.3)

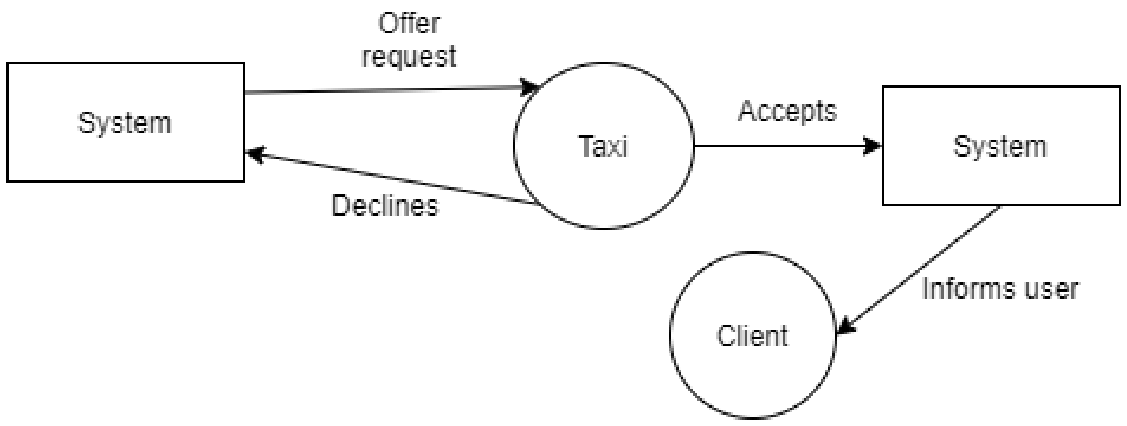


Figure 3.2: Diagram showing the single offer circuit

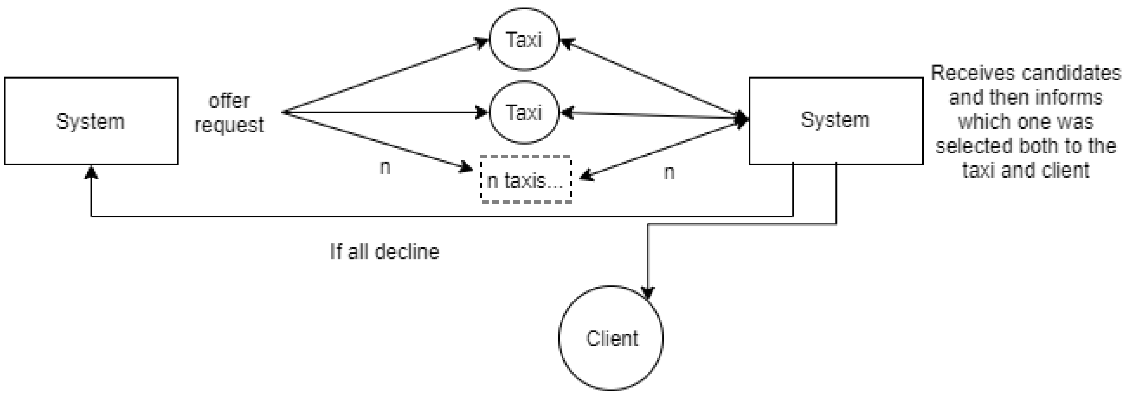
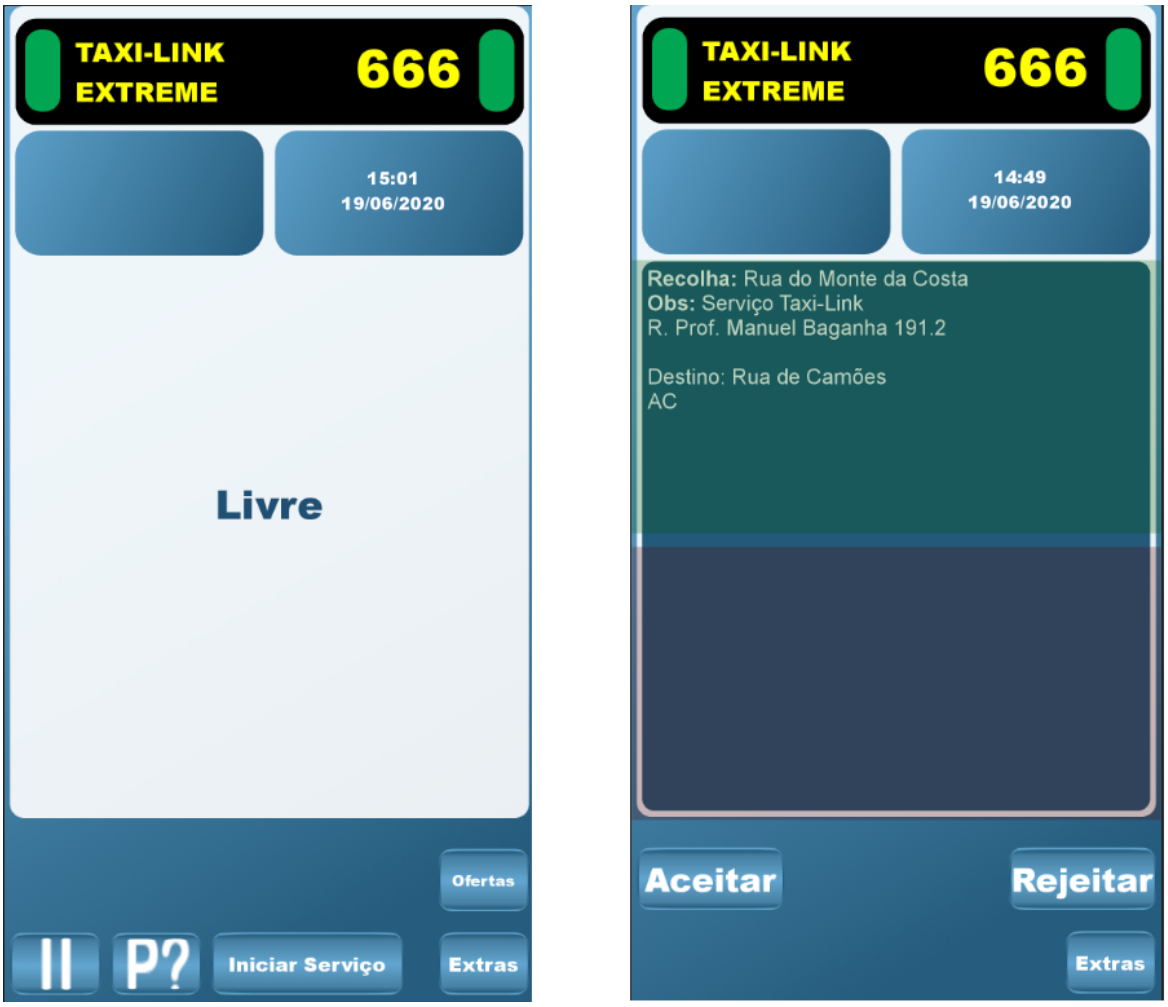

System
offer request
Taxi
Taxi
n
n taxis...
n
System
Receives candidates and then informs which one was selected both to the taxi and client
If all decline
Client

Figure 3.3: Diagram showing the multi offer circuit

3.4 Driver Application

Our driver application is the medium through which we communicate with our partnered drivers. And it is only available on Android. Each driver has a different login that lets them access the application features according to their profile. The driver has the option to set his state, such as busy, parked or in pause, accordingly. This information along with position updates are sent to the server. This makes it so that the server is always aware of each taxi's state, which is very important when it comes to the dispatching, since the dispatch system makes use of that information to make its decisions. One very important detail is that the application is always running even when it is in the background, this is so that it keeps communicating with the server in order for the data to be updated in both ends at all times.
The driver application also works has an indicator of service information for the driver, it displays information such as the pickup address and observations, the observations range from a message from the client or flags enumerating service characteristics such as air conditioned, payment method or if it's a special service (special services are credited to an entity instead of the actual client) (see figure 3.4). It also features a navigate button that opens a navigation app sending the coordinates automatically to the chosen application. And also contemplates some other options like generating and emitting invoices for the client, connecting to a bluetooth paired printer in order to print said invoices so to provide them to the clients.



Recolha: Rua do Monte da Costa
Obs: Serviço Taxi-Link
R. Prof. Manuel Baganha 191.2

Destino: Rua de Camões
AC

Figure 3.4: Screenshots from our driver application

3.5 Car-Server Communication Architecture

Every taxi carries a smartphone with our driver application installed and configured. For our Car-Server communication we use a bidirectional communication protocol. This is imperative because both agents have to relay information to one another when specific events occur in both ends. One of these events is the state management, as mentioned in the section above, it is really important that this information is relayed so that taxis can be eligible for services and so that the dispatch system has an accurate knowledge of which taxis are available.
In case of a communication failure, it is important to have a recover system, for example if a taxi is in a taxi stand in a certain position, if he has to reboot his device, without a recovery system the taxi would lose its place in the stand's queue. In order

to avoid this we have in place a recover system that grants that a taxi maintains its position even if there is a communication failure during a certain time frame, until it starts communicating again.

It is also crucial that the taxi informs the system of its location, for this we have got in place a system that uses GPS to determine what are its location coordinates. When the taxis GPS coordinates are determined they are sent to the system, this happens in intervals of X seconds. This is used for some reasons, the dispatch center needs to know where taxis are located to make good dispatch decisions, it is also used for taxi tracking during a service, allowing the users to see where their assigned taxi is at a given moment, the tracking can be done through the use of an application (either a mobile or web application).

Chapter 4

Passenger Application

Building a passenger application was the main goal for this project. Our objective with this application was to allow our clients to easily call a taxi with their smartphone. To do that we needed to understand what our requirements were, for that we looked at other electronic platforms in section 2.3, and what we came up was the following. First, we wanted the users to have register to be able to use this service, we will go into more depth about this in section 4.2. Then there is the process of calling the taxi, here we would need a way for clients to choose their pickup, and optionally their destination locations (it is possible to skip this step, see figure's 4.4 components 6 and 7), then we wanted to show them a suggested route and the taxi types available for that service along with cost information. Payment method selection would also be an important point. After those steps the clients need to be able to request a taxi, when they request a taxi we also wanted to show them feedback about the service status, as well as the possibility to track the taxi assigned to the service. At the end of the service, we also wanted to give them the option to evaluate said service. In figure 4.1 we have got a diagram showing this circuit. In the next sections, we will explain how we reached our goal and implemented these features.

4.1 Framework of Development: React Native

4.1.1 What is a Framework and why use one

To build the application we had to chose a framework for the development, a framework is a tool that provides a foundation on which software can be developed. They usually

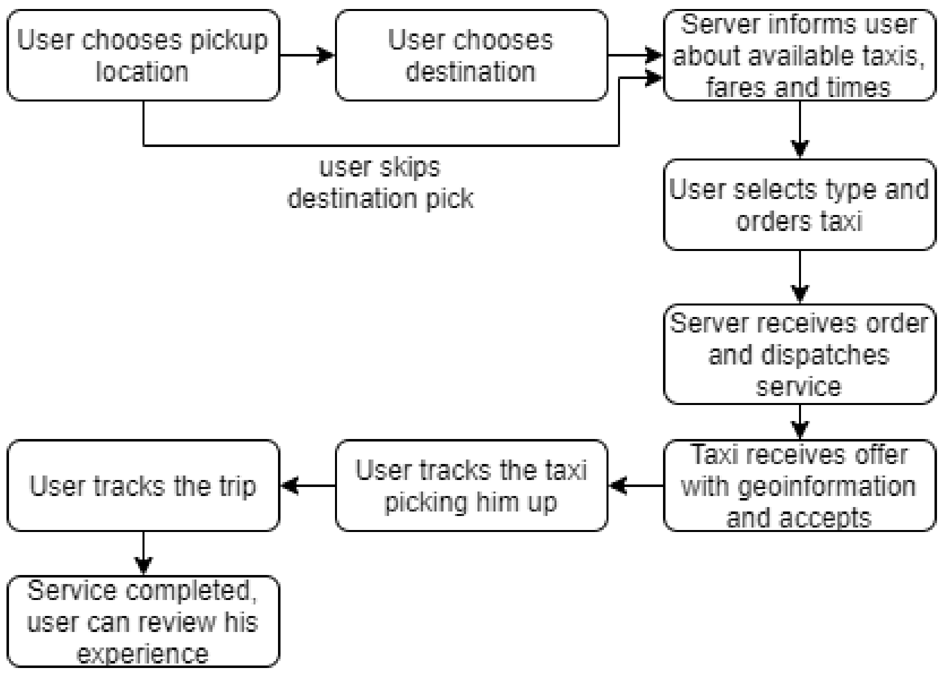

User chooses pickup location
User chooses destination
Server informs user about available taxis, fares and times
user skips destination pick
User selects type and orders taxi
Server receives order and dispatches service
Taxi receives offer with geoinformation and accepts
User tracks the taxi picking him up
User tracks the trip
Service completed, user can review his experience

Figure 4.1: Main use case diagram

include predefined classes and functions that can be used to process input, manage hardware, and interact with the system software. This foundation greatly improves the process of developing a new application since they remove the necessity of creating a lot of code that is common between applications, enabling the development to be more focused and on a higher level. While frameworks may seem like libraries, there is a difference, when you use a library you are the one importing the library and using its methods, whereas in a framework the difference is that it imports and calls your code, a good way to explain this is that a framework is like a program with holes that you fill with your code.

4.1.2 Considered Frameworks and Ultimate Choice

First and foremost we had to decide what framework we were going to use to build this mobile application. The first step in this process was deciding if we should build the application using native frameworks or with a cross-platform framework. Since we had as a key objective to build the application for the two leading mobile OSs, Android and iOS [11], if we were to use the native frameworks we would have had to have two different projects, with different technologies. With native software development kits (SDKs) we would have had more control over the OS and would also be able to achieve better performance, despite that, there is a big problem related to development time, since developing two projects with two different technologies is much more time-consuming when compared to one project with one technology. With cross-application frameworks there is the big advantage of a single code base for both projects, even though sometimes some platform-specific code is required, it does not have a great impact on development time. There is a great number of mobile development frameworks nowadays and many of them are already able of achieving great performance and have support for the features that we needed. Having analyzed both options we then decided to use a cross-platform framework, the main reason being the development time advantage, and also because it would not hinder our ability to implement the features we aimed to implement.

As we said, there are several cross-platform mobile development frameworks available nowadays, here we will enumerate the ones we considered while giving a brief explanation on how they work, and some advantages and shortcomings that we found relevant for us during this analysis:

1. **Ionic**[10] is a cross-platform UI framework that uses Web Views to render its content, a Web View in mobile devices allows your application to render web code (HTML, Javascript, and CSS). Ionic also has integrated support for web frameworks such as Angular or React. An advantage for us was that we had previous experience with it, which would speed up the development process. But there are some negatives which outweigh this, achieving good performance is hard with Ionic since it does not render native components and uses web technologies to render its screens, and since it is web-based, for us to use the Google Maps Native SDK we would have to go through some trouble.

2. **Xamarin**[14] is owned and supported by Microsoft, the languages used for development are C# and XAML, it renders native components which is great for performance and also has support to use platform-specific SDKs. One negative point is that support for 3rd party libraries was not great and we had no previous knowledge of the languages.

3. **React-Native**[12] is developed and supported by Facebook, it is based on the javascript framework React, it renders its UI components using native components which as said for Xamarin, is great for performance, has support for platform-specific SDKs. It has hot reloading, which is a feature that speeds up the debugging process greatly since changes made to the Javascript code are immediately shown while the application is running.

Figure 4.2: Ionic, Xamarin and React Native logos [12] [10] [14]

After evaluating these options we chose React-Native, the main reason was previous knowledge of Javascript, paired with its capabilities, like the native rendering of components, support for native SDKs, and great debugging tools. Another point in favour is that it also has a large community which is great for when problems are faced during development and also means substantial support for third party modules.

4.2 Registration and Login with Third Party Platforms

4.2.1 Registry and Login

The registry is a very important step, you need to be registered and verified to access the applications main feature (requesting a taxi), our reasoning to add the registration was that if the application could be used by anyone with anonymity there would be a higher risk of dummy requests and also having client accounts allows us to create a more personalized and richer experience for the user, as we will talk about in the next sub-section.

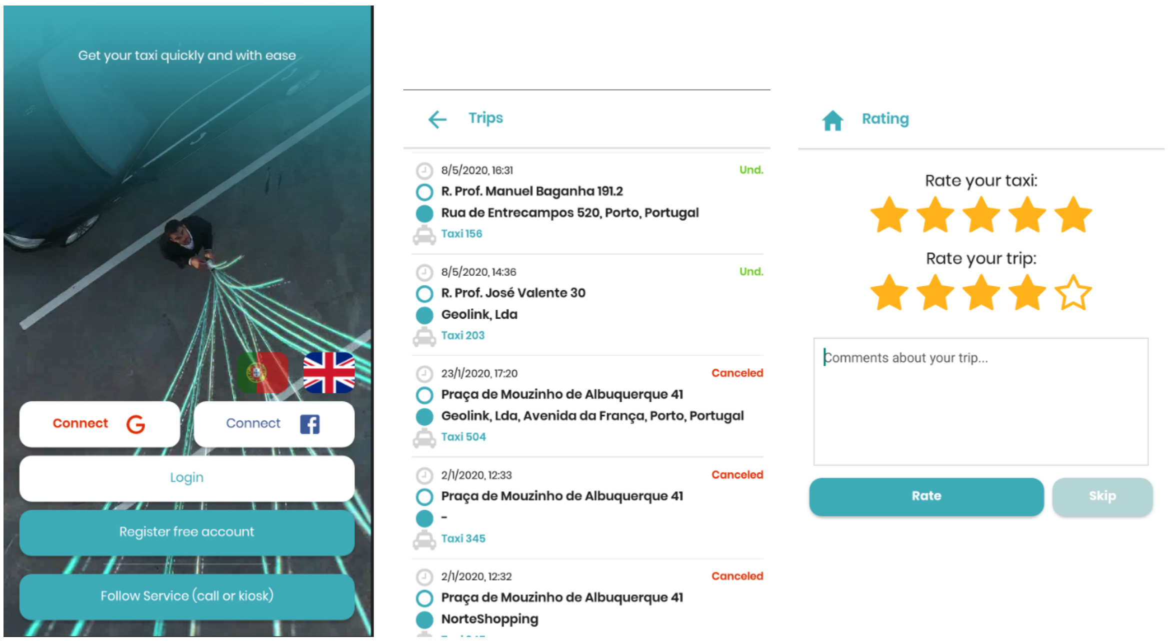
For the registration, clients have to insert their phone number which is the primary key with which we identify them in our system. We use the phone number because in the Taxi-Link ecosystem services might be generated by other means such as phone calls, so this allows us to link services requested by the same clients independently of the system used. Then they have to insert their name and insert a password. To complete the process, for security purposes in order to try to grant that the phone number belongs to the client registering, we generate a verification code that is sent to the user's phone by SMS. This verification also improves the value of the user base by reducing the number of spam registrations due to the need of having a valid phone number. If they do not verify their account the registration process is halted and that client does not have access to the application. After the users are registered and verified, they are automatically logged in and granted an auth token, which is used to preserve the log in each time they open the application, this token has an expiration date that can be renewed and is stored in the application data. We also have third party platforms login, with Facebook and Google for both the Android version and iOS version of the application (see figure 4.3), and for the iOS version, we have added Apple sign-in. While the user will still have to provide the mandatory information asked for in the normal registration process, the login process gets simplified since he will not have to insert his phone number and password combination every time they need to log in.

4.2.2 Client Account Options

The client account was kept simple in terms of options, basically what we have in the user area are the following sections, the profile where they can change their name, e-mail, password, and the selected language. A section where they can define their work and home locations, these will be displayed in the destination location choosing process as quick choice options. We also have different payment options, those being, cash in the taxi, card payment in the taxi, card payment by application, and also there is an option to add themselves as a credit client. Credit clients are clients where their company has a contract with a fleet and the clients themselves do not have to pay, instead, their company will be billed for the service. Another very important area for the user is the service history page (see figure 4.3), here the users are able to see all services that he made with our application, with some information about the service along with the option for the users to delete a service from their history or to rate it if they have not rated it previously (subject to a time frame).

4.3 Google Maps API

In our project the Google Maps API had a very important role, it grants the capabilities of Google Maps to your application. As it stands it offers a very complete set of functionalities, allowing us to implement all the features we envisioned for our application. It has a substantial number of feature-rich and very detailed services that get constant updates, its pricing is reasonable, though optimal usage is important since non-optimal usage can easily bump up the payment balance by a significant margin, another great aspect is its great documentation[7] which is very detailed and covers a


Get your taxi quickly and with ease
Connect
Connect
Login
Register free account
Follow Service (call or kiosk)
Trips
Rating
Rate your taxi:
Rate your trip:
Comments about your trip...
Rate
Skip

8/5/2020, 16:31 Und.
R. Prof. Manuel Baganha 191.2
Rua de Entrecampos 520, Porto, Portugal
Taxi 156

8/5/2020, 14:36 Und.
R. Prof. José Valente 30
Geolink, Lda
Taxi 203

23/1/2020, 17:20 Canceled
Praça de Mouzinho de Albuquerque 41
Geolink, Lda, Avenida da França, Porto, Portugal
Taxi 504

2/1/2020, 12:33 Canceled
Praça de Mouzinho de Albuquerque 41
-
Taxi 345

2/1/2020, 12:32 Canceled
Praça de Mouzinho de Albuquerque 41
NorteShopping

Figure 4.3: Screenshots of the Taxi-Link passenger application (login, history and rating screens)

lot of what a developer needs to know to use the API.
We will now explain what services we used from this API and how we used them, giving concrete examples of what advantages they brought us and also explaining our reasoning for using them as we did.

4.3.1 Visualization of Information on Top of the Map

Google provides us with a map object like the one used by Google Maps. This object renders the map and has many customization options available. One of those options is a way to add your own map markers, with the only requirement being the set of coordinates where it is to be placed, then you have some other options such as using a custom component instead of the default red pin icon, these custom components can be just like any other component in an application, meaning you can add a button, an image, text, or others, this becomes a great tool that gives us the possibility of customizing the map experience showing what we need on top of it. We used these markers in some scenarios such as to show taxis in the map. There are two different instances where we need to show taxis, one is in the home screen where we show X taxis near the selected pickup point's location. To achieve this effect we created an image depicting a taxi as seen from above. The markers also have a heading property that determines their orientation, we calculate each taxi marker's heading by the taxi's current position in relation to its previous positions, when it comes to taxis parked in a taxi stand we calculate the heading for one of them and treat them as a group (see figure 4.4). The other instances where we show taxis is when tracking them, here instead of showing the taxis in a stationary state we need to represent the taxi's movements. The movements are determined by a set of positions that we get from an update received every X seconds. At first what we did in order to represent movement was to simply change the taxi marker's coordinates to the updated ones (updated the heading as well) and while this worked well enough, the movement was not very smooth since the taxi seemed to jump from coordinate to coordinate, even with updates every second. To make the movement smoother we tried calculating ten intermediate positions between two consecutive positions and updating them each tenth of a second. This technique did achieve the impression of a much smoother movement. We also made use of these markers to show other information, including the selected points (pickup and destination) with a box with the address and expected pickup time and in the home screen in addition to the taxis near the selected pickup point we also show the near X taxi stands (figure 4.4), this also helps to contextualize the users with their surroundings and to give them an idea of what to expect in terms of service responsiveness for that location.

Another tool that we can use to display information on top the map is the polyline component, this component receives an array of point coordinates, and it renders a line on top of the map connecting all the points provided in that array. There are also some customizing properties such as stroke width and color, which we also used in order to improve visibility and to match our design choices. We are using polylines in two scenarios, one is when we want to show the suggested route after the user picks his the desired pick-up point and destination (see figure 4.7), we talk about how we get the route in the sub-section 4.3.3. The other use is to indicate the trail of the taxi's movements during tracking (see figure 4.5), the coordinates we use to build the array are the same from the taxi position updates that we get from our own services, which we elaborate on in section 4.5.

4.3.2 Google Places and Address Picking

Since our goal is to connect passengers and taxis, it is important that the passengers can indicate their desired pickup and destination locations. In our application there are two different methods to indicate it, the first method uses Reverse Geocoding[6]

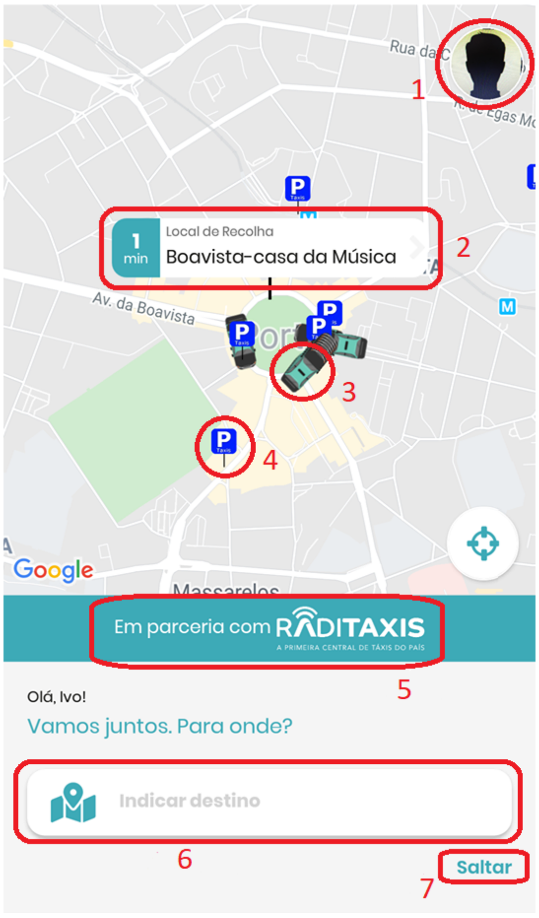


Figure 4.4: Screenshot of the Taxi-Link applications home screen

to resolve the address, and the second one matches text with a location.
The **Reverse Geocoding** method, is used when navigating the map in the home screen, which first starts by trying to get the user's location centering the map in their location and using this method to resolve the address, after having their location users are able to move the map freely. We use a fake marker, a fake marker is a UI only component (not a Google Maps marker) to indicate the center of the map and to show the resolved address corresponding to the map (see figure 4.4). Using the map's OnRegionChange event we defined a callback function which uses the current map coordinates, also corresponding to the center of the map, and we use those coordinates as arguments for the Google Maps Reverse Geocoding API. This API's response returns an exact address for those coordinates, then we simply use that address in the fake marker to show to the user the current pick-up point's address and later on we also use this address and coordinates to inform the driver where the pickup point would be. Altough we have got a reverse geocoding service in our platform (section 3.2) we use the address returned by Google since we found it to be more accurate and

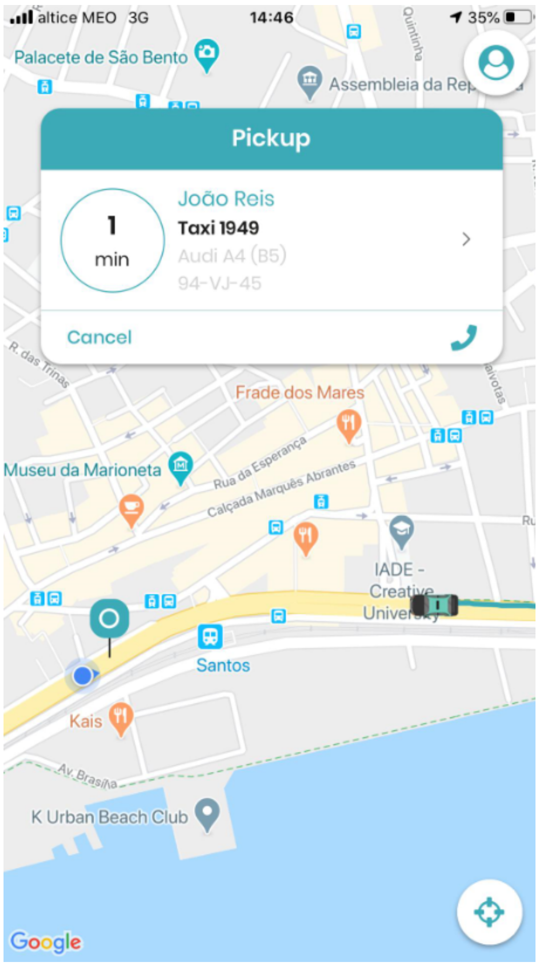


Figure 4.5: Screenshot of the Taxi-Link applications pickup screen.

more updated.
The other way to pick addresses is indicating **locations by text**, this was important to us as it is the only way to indicate the desired destination location, being also an option for the pickup location indication. The best option we found to achieve this goal was to use a service that provided an address autocomplete feature with geological information, and we chose to use the Google Places Autocomplete API[4]. It has great documentation on how to use it and provides a lot of useful information that we can use for our advantage. The utilization of this API is not complicated as it only has one required argument which is the text we want to resolve, but there are some optional arguments that if defined correctly are very important as they can improve or deteriorate the results dramatically depending on our utilization. Getting these arguments right results in getting more accurate matches with the least keystrokes needed from the users, improving their experience. From all the optional arguments

available, we are going to enumerate the ones we did use to optimize the results and the reasoning behind those choices:

1. Location - here we send the chosen pickup point (when defined), this along with the radius argument has a great impact narrowing down the results.
2. Radius - we set it at 10.000 meters, while trips can be farther than 10.000 meters, from our testing we found that the query yielded better results when compared to using larger values like 15.000 meters or 20.000 meters.
3. Components - here we used the country option, and limited the results to Portugal, this is because at the moment it is the only country where we operate and removes other countries locations from appearing in the results. This option is also flexible and you can add more countries if you need to get results from a number of specific countries at once.

Our approach with keystrokes was to make it so that after two characters, and after every following character, we make a new request with the updated text to the API. The returned results contain an array with up to five different locations objects, and each object has a number of informations about the corresponding location. We take these results and show the addresses in a list (see figure 4.6). Each item in this list is a clickable button which the user can then click to select the desired address, on said click we also have to finalize the Autocomplete API request with a Places Details request[8], this Places request only requires the place_id parameter, which is provided in the Autocomplete response's object. We make this Places request for optimal financial usage of these APIs, it is important to bundle up all the requests for a given session using what is called a session token, it is suggested that this token is generated by using a version 4 UUID[13] generator and its to be sent in the "sessiontoken" arguement of the Autocomplete requests and also in the Places request, if done correctly all these requests will be bundled together and will be billed as one. In cases where the user might not make a place request but does more than six autocomplete requests (at the time of writing this was the number of requests that surpassed the cost of completing a session with a Places request) we also complete those sessions.

During the pickup point selection it is also important that we check within our services if the pickup location has a corresponding fleet, this is to make sure that we can provide our services in said location, if the location is not valid we warn the users they have to choose another pickup location if possible, blocking them from going on with their request.

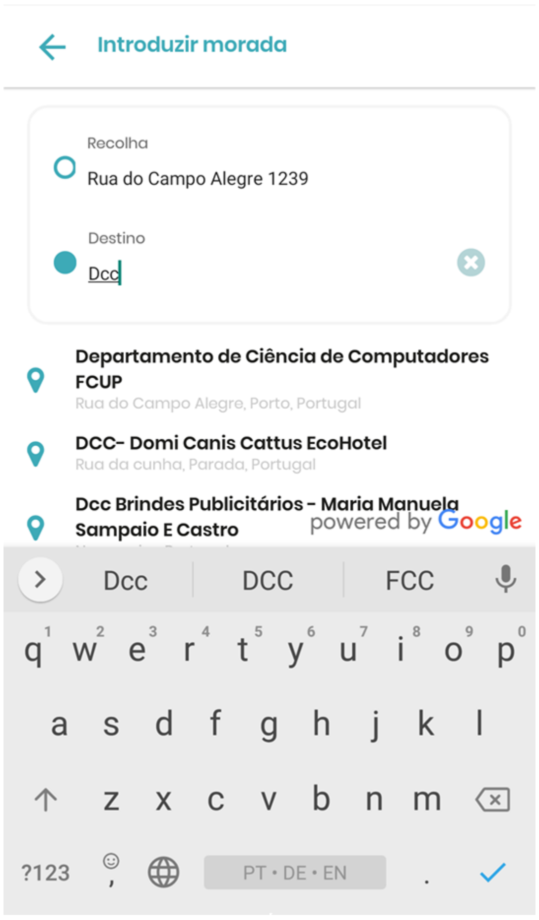

Introduzir morada
Recolha
Rua do Campo Alegre 1239
Destino
Dcc
Departamento de Ciência de Computadores FCUP
Rua do Campo Alegre, Porto, Portugal
DCC- Domi Canis Cattus EcoHotel
Rua da cunha, Parada, Portugal
Dcc Brindes Publicitários - Maria Manuela Sampaio E Castro
powered by Google
Dcc
DCC
FCC
?123
PT • DE • EN

Figure 4.6: Screenshot of the Taxi-Link applications address choosing screen

4.3.3 Route Calculation

As was mentioned before, if the client selects both a pickup and destination location, we then show a screen in which we show a suggested route (see figure 4.7). To show a route we needed to calculate it. To accomplish this effect we used another Google service that is called the Directions API[5]. This API provides us with routes between two given sets of coordinates, which are the only required arguments. The returned routes are sent compressed [3] and we have to decode them in our end in order to have the actual array of coordinates, along with the routes it also provides us more information about these, from which we are using the duration estimate and route length. Using this API we make use of some of the optional arguments, one being the mode of travel which in our case is driving and also the alternatives option which we set to true, the latter being an especially important setting, because what this argument does is, when set to false, the request returns only the route that Google finds optimal, while with it set to true the API will respond with three different routes.

This is important for us in regards to cost as we will talk about in the next section. It is important to understand what is an optimal route for Google [5] to explain why it is not the one we base our pricing on every time. Google's primary route optimization target is travel time, but as explained in the APIs documentation it also takes in account other factors such as distance, number of turns and more[5]. Sometimes the optimal route for Google is not the one that taxis take, we came to this conclusion by talking to some of our client fleet's contacts and we understood that taxis usually try to take the shortest route possible, which might not be the optimal one for Google. This is because in some instances the shortest route may actually take longer or go through bad roads, among other factors. This way we take the three alternatives presented to us and make our choice based on the following reasoning, first we discard the routes that take longer than 20 minutes than the fastest one, this is because if the difference in time is this large then the taxi will most of the time choose the quicker route, then with the remaining routes we choose the shortest one to present and make the cost calculation with.

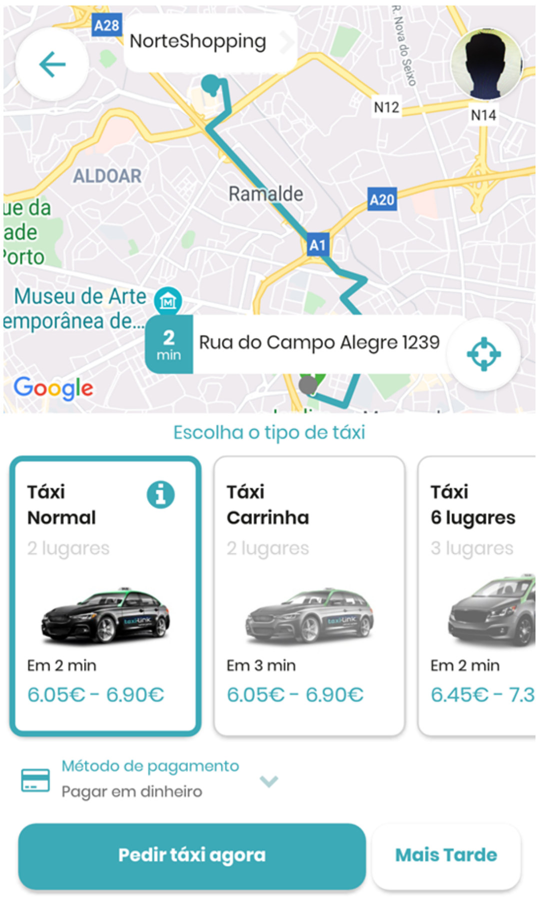


Figure 4.7: Screenshot of the Taxi-Link applications order screen

4.4 Cost Simulator

This was a hard feature to get right, but it is also a major feature in a transport application that we wanted to add so that our clients could have a good idea of what they are getting into in terms of the cost before calling a taxi. The cost of a taxi service is defined by legislation and has a convention which is given by the following parameters, ”bandeirada”, it is the minimum cost of a service, the call cost, that is also accounted for in an application service, and any other supplementary costs such as extra-baggage or pet transportation. These are the elements that contemplate the initial price of a service. Then there is the amount that the trip itself adds to the cost, this amount is calculated by an accumulation of impulses during the length of service, these impulses are triggered by either distance or time, and each impulse is assigned a cost depending on the tariff in place for the corresponding service, and as we can see in figure 4.8 there are different tariffs for different types of services, time of day also influences their cost, and taxi types be it by different passenger numbers or if they have taxi badge or not also influences the tariff. It is important to note that the night time tariff is higher for all types of service and it is active from 9 pm to 6 am, weekends and holidays also use the night tariff. If applicable the client will also have to pay for tolls for both ways, or parking but we are not considering them in this simulator at the moment as we do not have this information, and the client might make a deal with the taxi driver. So based on all this information and having the tariff charts for each fleet, we built the simulator. Following we present the pseudo-code for this simulator:

```
def cost_simulation(route, fleet_id, taxi_type):
    total_cost = 0
    total_cost += bandeirada
    total_cost += call_supplement
    polyline = decode_polyline(route)
    last_point = null
    distance = 0
    for point in polyline:
        if last_point != null:
            distance += get_distance(point, last_point)
            if (dist) > IMPULSE_DIST:
                total_cost += get_impulse_cost(taxi_type, fleet_id, dist, point)
                distance = 0
        last_point = point
    aux_range = total_cost * 0.15
    return (total_cost - aux_range, total_cost + aux_range)
```

The get_impulse_cost function defines how much each fraction of distance adds to the cost. We are only using distance and not time because time only becomes a factor when the taxi is stopped (more specifically, it did not achieve the determined impulse distance in 24 seconds), and the two main instances on how this might happen are being stopped in traffic lights which has a negligible impact on the price, or if there is a lot of traffic, but since we do not have traffic information we are not considering it for our simulator. Given this, the cost these impulses will add, like we said above, depending on the taxi type and also the time of day, and service type, depending on the region where the service is being provided. After all the impulses are calculated and added to the total cost, we then return a range with a 15% reduction and addition to the cost as the bounds, we do this because we ca not be sure our prediction will be 100% correct, since the taxi may take another route or simply not follow the whole of the suggested route, and because we are not considering traffic as a factor for the cost. Although our testing for validation was limited because we do not get the final price for a service sent to us at the moment, we asked feedback from our fleets with in-field testing. This testing helped us define what this interval range should be in order to be more accurate. After some testing, we agreed on the 15% for the range, since it was accurate in most evaluated cases from the feedback provided by the in-field testing.

4.5 Communication with the Server and Tracking in Real Time

With this application we have to pass and gather data to and from the Taxi-Link system infrastructure, for that we needed to define how the application would communicate with the the system. We decided to build two separate services for two different use cases, one service was a RESTful API, and the other one a Web Socket Server. RESTful APIs are based on the Representational State Transfer (REST) architecture that utilizes the request models from HTTP requests (such as GET, PUT, POST, etc) and responses, it uses a low amount of bandwidth which optimizes the internet usage. Another considerable advantage of a RESTful API is that the API calls are stateless, this means that any request is independent of each other and can be made

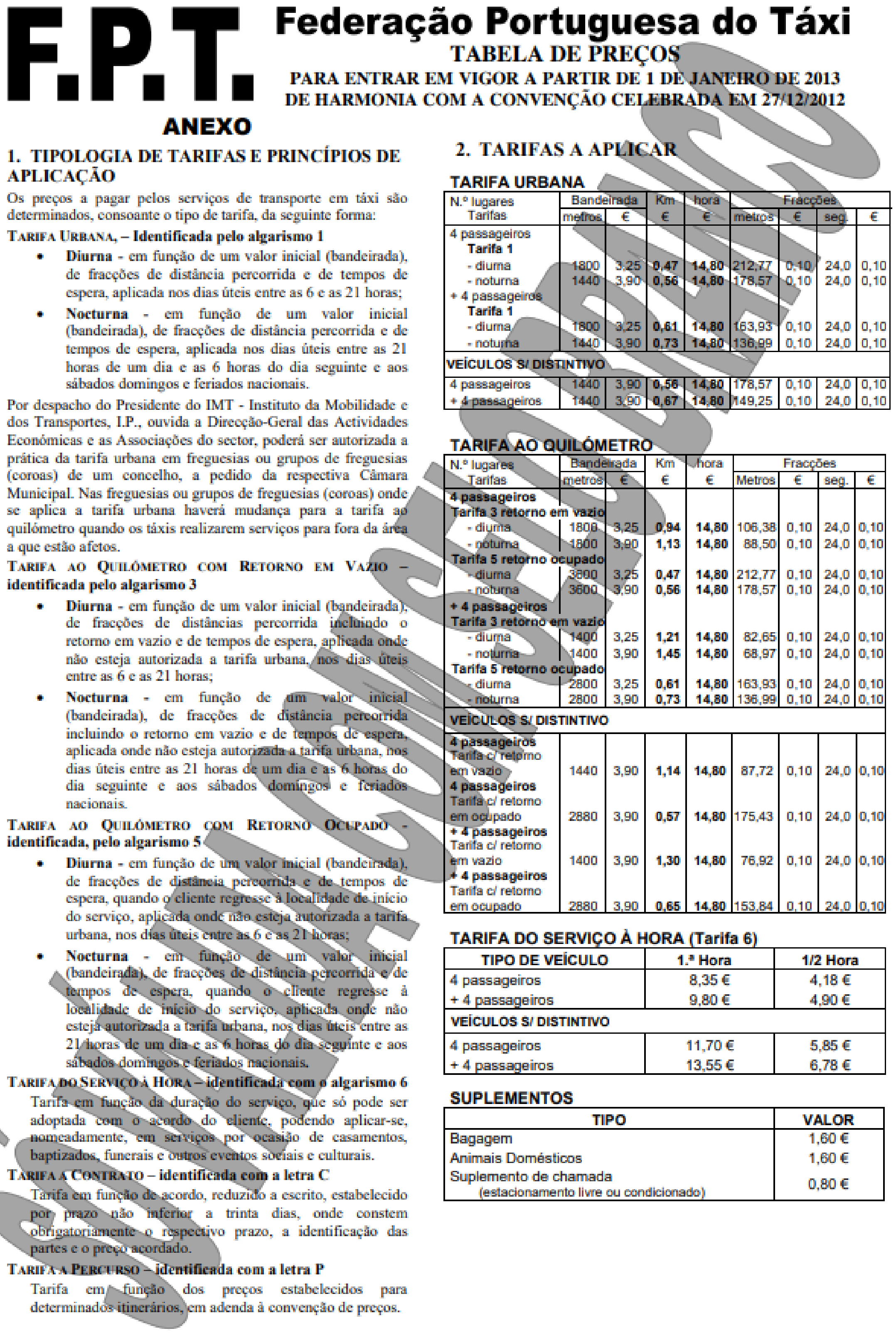
at any point in a session which allows us to be more flexible with the flow and to have a not very complicated service infrastructure.
In figure 4.4 we have some components that are shown according to information fetched from the API, being the following:

- **1** - this is a button that connects to the user profile, the user's profile is fetched from the API according to a unique auth_token,
- **2** - this box has the estimated pickup time if a user would call a taxi from that location, this time is calculated by getting the distance using the same method in section 3.3 and then using a function to convert the distance to time. In this box we also show the address of the selected point, the address comes from Google's Geocoding service (see sub-section 4.3.2), also checking the coordinates in our server, to determine if that is a valid pickup point (showing an error if that is not the case),
- **3** - here we show taxis using the method explained in subsection 4.3.1. The coordinates where we draw the taxis are based on the current pin location, making a call to the API that calculates which X taxis are the ones closest to said location,
- **4** - this works the same way as component three but instead of fetching the closest taxis it fetches the closest taxi stands,
- **5** - we show an image of a fleet according to which one would serve the request if it was made with that location as the pickup point.

A web socket server is a server that will open a communication channel with a client and send him information based on events that are triggered by some mechanism. We chose a web socket server for updating the client with service-related events, such as when a taxi accepts the service, when the taxi has reached the pickup point, among others. We also use this socket server to send the taxi's position during tracking. If these actions were to be resolved by the RESTful API, our client application would have to have a loop making requests from time to time to check the state of the service or the position of the assigned taxi, this would not be optimal, since it the client application would have to make useless requests (when the state has not changed) and also could introduce some delay in delivering the updated information to the user since it would depend on the request interval and we wanted to update the user as soon as possible for better responsiveness, while also avoiding to make useless requests that would be wasting resources. This are the reasons why we chose a web socket server for these cases, this way everytime one of these events happens, the event will trigger a message to be sent to the user by the Taxi-Link system. This allowed us to achieve our goal of updating the user when needed without the client application having to make any requests, granting a better experience.

The web socket system is designed to work in the following way, when a user logs into the application the client application will try and open a socket between the server and itself. Since it is important to check if the connection is up when the client is opened, the client is responsible to check the connection liveness and to recover said connection if it detects that it has failed.
It is also important for this service that when a connection dies, either by a network error or the client application being closed when the connection is re-established, that it recovers the previous state of the connection, for example if there is a service ongoing we need to send the last event that was triggered in the server to the client. It is very important to do so, so that whenever a user opens the application if he is in the midst of a service he gets the screen that he is expecting along with the correct service state. Imagine for example the user was being picked up and the service happened to be canceled by the fleet operator, or if the taxi had arrived, when the user opened the client application he would not have any information about those events. To avaoid this, we made it so that whenever the socket is opened the last event related to the user's account is always sent. We also did not want to flood the user with useless and repeated information, like being notified that your last service ended even though you already knew it, every time you opened the application. So for final events such as canceled service or end of service we save the state of the last one in the application data, so when such events are sent by the socket server we check the service id's in order to not trigger an action already triggered previously.

There is one specific case where the socket being used in conjunction with the RESTful API, which is for the payment. Our system receives payment requests by the taxis, this triggers an event in the socket server, which then informs the client application that there is a payment request. Then the actual payment communication is handled by communicating through the API. This is another example where it is imperative that the server can inform the client application on demand instead of the client application handling all the requests.



F.P.T. **Federação Portuguesa do Táxi**

TABELA DE PREÇOS

PARA ENTRAR EM VIGOR A PARTIR DE 1 DE JANEIRO DE 2013
DE HARMONIA COM A CONVENÇÃO CELEBRADA EM 27/12/2012

ANEXO

1. TIPOLOGIA DE TARIFAS E PRINCÍPIOS DE APLICAÇÃO

Os preços a pagar pelos serviços de transporte em táxi são determinados, consoante o tipo de tarifa, da seguinte forma:

Tarifa Urbana, – Identificada pelo algarismo 1

- **Diurna** - em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, aplicada nos dias úteis entre as 6 e as 21 horas;
- **Nocturna** - em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, aplicada nos dias úteis entre as 21 horas de um dia e as 6 horas do dia seguinte e aos sábados domingos e feriados nacionais.

Por despacho do Presidente do IMT - Instituto da Mobilidade e dos Transportes, I.P., ouvida a Direcção-Geral das Actividades Económicas e as Associações do sector, poderá ser autorizada a prática da tarifa urbana em freguesias ou grupos de freguesias (coroas) de um concelho, a pedido da respectiva Câmara Municipal. Nas freguesias ou grupos de freguesias (coroas) onde se aplica a tarifa urbana haverá mudança para a tarifa ao quilómetro quando os táxis realizarem serviços para fora da área a que estão afetos.

Tarifa ao Quilómetro com Retorno em Vazio – identificada pelo algarismo 3

- **Diurna** - em função de um valor inicial (bandeirada), de fracções de distâncias percorrida incluindo o retorno em vazio e de tempos de espera, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 6 e as 21 horas;
- **Nocturna** - em função de um valor inicial (bandeirada), de fracções de distância percorrida incluindo o retorno em vazio e de tempos de espera, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 21 horas de um dia e as 6 horas do dia seguinte e aos sábados domingos e feriados nacionais.

Tarifa ao Quilómetro com Retorno Ocupado - identificada, pelo algarismo 5

- **Diurna** - em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, quando o cliente regresse à localidade de início do serviço, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 6 e as 21 horas;
- **Nocturna** - em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, quando o cliente regresse à localidade de início do serviço, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 21 horas de um dia e as 6 horas do dia seguinte e aos sábados domingos e feriados nacionais.

Tarifa do Serviço à Hora – identificada com o algarismo 6

Tarifa em função da duração do serviço, que só pode ser adoptada com o acordo do cliente, podendo aplicar-se, nomeadamente, em serviços por ocasião de casamentos, baptizados, funerais e outros eventos sociais e culturais.

Tarifa a Contrato – identificada com a letra C

Tarifa em função de acordo, reduzido a escrito, estabelecido por prazo não inferior a trinta dias, onde constem obrigatoriamente o respectivo prazo, a identificação das partes e o preço acordado.

Tarifa a Percurso – identificada com a letra P

Tarifa em função dos preços estabelecidos para determinados itinerários, em adenda à convenção de preços.

2. TARIFAS A APLICAR

TARIFA URBANA

N.º lugares Tarifas	Bandeirada		Km	hora	Fracções			
	metros	€	€	€	metros	€	seg.	€
4 passageiros								
Tarifa 1								
- diurna	1800	3,25	**0,47**	**14,80**	212,77	0,10	24,0	0,10
- noturna	1440	3,90	**0,56**	**14,80**	178,57	0,10	24,0	0,10
+ 4 passageiros								
Tarifa 1								
- diurna	1800	3,25	**0,61**	**14,80**	163,93	0,10	24,0	0,10
- noturna	1440	3,90	**0,73**	**14,80**	136,99	0,10	24,0	0,10
VEÍCULOS S/ DISTINTIVO								
4 passageiros	1440	3,90	**0,56**	**14,80**	178,57	0,10	24,0	0,10
+ 4 passageiros	1440	3,90	**0,67**	**14,80**	149,25	0,10	24,0	0,10

TARIFA AO QUILÓMETRO

N.º lugares Tarifas	Bandeirada		Km	hora	Fracções			
	metros	€	€	€	Metros	€	seg.	€
4 passageiros								
Tarifa 3 retorno em vazio								
- diurna	1800	3,25	**0,94**	**14,80**	106,38	0,10	24,0	0,10
- noturna	1800	3,90	**1,13**	**14,80**	88,50	0,10	24,0	0,10
Tarifa 5 retorno ocupado								
- diurna	3600	3,25	**0,47**	**14,80**	212,77	0,10	24,0	0,10
- noturna	3600	3,90	**0,56**	**14,80**	178,57	0,10	24,0	0,10
+ 4 passageiros								
Tarifa 3 retorno em vazio								
- diurna	1400	3,25	**1,21**	**14,80**	82,65	0,10	24,0	0,10
- noturna	1400	3,90	**1,45**	**14,80**	68,97	0,10	24,0	0,10
Tarifa 5 retorno ocupado								
- diurna	2800	3,25	**0,61**	**14,80**	163,93	0,10	24,0	0,10
- noturna	2800	3,90	**0,73**	**14,80**	136,99	0,10	24,0	0,10
VEÍCULOS S/ DISTINTIVO								
4 passageiros Tarifa c/ retorno em vazio	1440	3,90	**1,14**	**14,80**	87,72	0,10	24,0	0,10
4 passageiros Tarifa c/ retorno em ocupado	2880	3,90	**0,57**	**14,80**	175,43	0,10	24,0	0,10
+ 4 passageiros Tarifa c/ retorno em vazio	1400	3,90	**1,30**	**14,80**	76,92	0,10	24,0	0,10
+ 4 passageiros Tarifa c/ retorno em ocupado	2880	3,90	**0,65**	**14,80**	153,84	0,10	24,0	0,10

TARIFA DO SERVIÇO À HORA (Tarifa 6)

TIPO DE VEÍCULO	1.ª Hora	1/2 Hora
4 passageiros	8,35 €	4,18 €
+ 4 passageiros	9,80 €	4,90 €
VEÍCULOS S/ DISTINTIVO		
4 passageiros	11,70 €	5,85 €
+ 4 passageiros	13,55 €	6,78 €

SUPLEMENTOS

TIPO	VALOR
Bagagem	1,60 €
Animais Domésticos	1,60 €
Suplemento de chamada (estacionamento livre ou condicionado)	0,80 €

Figure 4.8: Taxi tariff chart

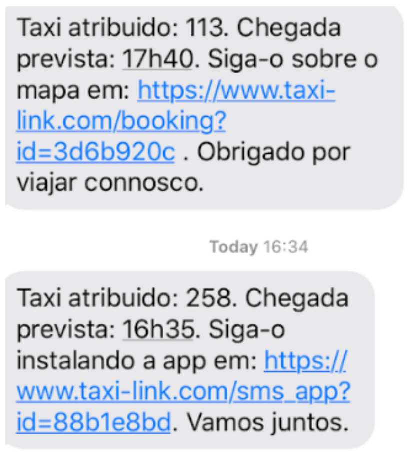
Chapter 5

Technological Migration of Passengers

Taxi-Link already has a large number of clients. In the middle of the conception of this project, we have had a discreet launch of our passenger application, discreet because it was still in active development, and although we did have some clients registering in the application, normally, most of our clients still use a phone call or even kiosks to call a taxi. Our aim in this chapter is to illustrate our view on how we can take advantage of current mobile technology and integrate it with our current dispatch system to inform them about this application and, to motivate these phone call clients to make a shift of paradigm, migrating them from requesting taxis by a phone call to instead start using the passenger application.

First, we will explain how the process of calling a taxi through the Taxi-Link system works. When you call a dispatch center you speak with a phone operator, you inform the operator where you want the taxi to pick you up and where you will want to go. Then you specify your needs, be it the number of seats, payment method, wheelchair compatibility, and other ones. The operator will then send a request for a taxi through the Taxi-Link dispatch system, and the call ends. When the system assigns a taxi to the client, the client receives an SMS from the system. The contents of this SMS are comprised of the following fields, an estimate for the pickup time, the taxi's badge number, and also an URL. This URL would take the client to a webpage where it is possible to track the taxi on a map while it is in the pickup state. Since this SMS and the phone-call itself are the only points in the process where we have a direct communication line to the client, we wanted to do something with one of them. We chose the SMS to achieve to reach our goal.

Our idea was to change the webpage sent in the SMS's URL when viewed on a mobile device. We thought of integrating this webpage with the application, having it serve as a bridge for the client to meet the latter, but we also needed to offer an immediate advantage for the client to want to get it. With that objective in mind, we made it so that the tracking of the taxi was now to be done via the application instead of the website itself, giving the users said reason as to why they would want to install the application.


Taxi atribuido: 113. Chegada prevista: 17h40. Siga-o sobre o mapa em: https://www.taxi-link.com/booking?id=3d6b920c . Obrigado por viajar connosco.

Today 16:34

Taxi atribuido: 258. Chegada prevista: 16h35. Siga-o instalando a app em: https://www.taxi-link.com/sms_app?id=88b1e8bd. Vamos juntos.

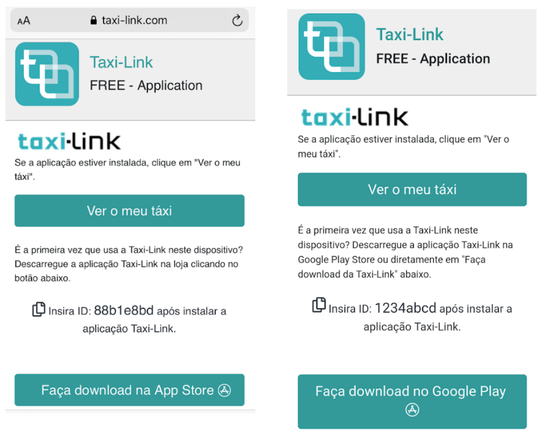
Figure 5.1: SMS example (old SMS on top and new SMS in the bottom).

5.1 Connecting the APP and the SMSs

One thing that is important to any kind of user experience is to make it as simple as possible for the user. With this in mind we wanted our new page to facilitate the user experience, how did we achieve this?

First of all, we had to create a new section in the app where you could track the phone call services, and we also needed a way for the users to access these services. To do so we started associating with each service a service token, this token is a unique alphanumeric eight-character code, which when inserted in the application opens a new web socket connection. This connection works just like the web socket connection for the normal services as explained in section 4.5.

So what did we want from this new webpage? We needed it to show the generated token for each service, and we also wanted it to have two buttons, one that would connect to the application and automatically open the web socket, and another one that would redirect you to your OSs corresponding application store. The second button was rather simple, we only needed to identify which OS the user was on, which the browser can identify, and then having it redirect them to the corresponding application store. Regarding the first button, we were able to achieve the desired effect by using what is called **deep links**. What is a deep link? In the web deep links are simply hyperlinks that instead of taking you to the home page of a certain website have some information that, for example, would take you to another page inside said website, that information is contained in the URL. In the mobile ecosystem, deep links work a bit differently, in this context we can define a certain URL prefix (example: tlapp://) that makes it so that when you try to path to said URL having the application installed on your device it will know to open your application. When the application is opened this way, be it a cold or warm start (if the application was closed or open previously), we have access to the whole URL, this means we can define arguments and build a parser and define different actions for those arguments. So we defined the URL for our app and when the deep link URL is generated in the webpage we append a "token=XXXXXXXX" argument to that URL. If the application is opened this way having said argument, it will automatically try to establish a web socket using that token. Also of note, if the user has to first download the application, after the download he can still go back to the webpage and click the deep link button, making it much easier for him to follow his service. When these services terminate the final screen is shown and there we enumerate several features that come with using the application for calling taxis instead of a phone call, along with a button that takes them to the registration screen, as well as a go-to the beginning button (see figure 5.3).



AA taxi-link.com

Taxi-Link
FREE - Application

taxi·Link

Se a aplicação estiver instalada, clique em "Ver o meu táxi".

Ver o meu táxi

É a primeira vez que usa a Taxi-Link neste dispositivo? Descarregue a aplicação Taxi-Link na loja clicando no botão abaixo.

Insira ID: 88b1e8bd após instalar a aplicação Taxi-Link.

Faça download na App Store

Taxi-Link
FREE - Application

taxi·Link

Se a aplicação estiver instalada, clique em "Ver o meu táxi".

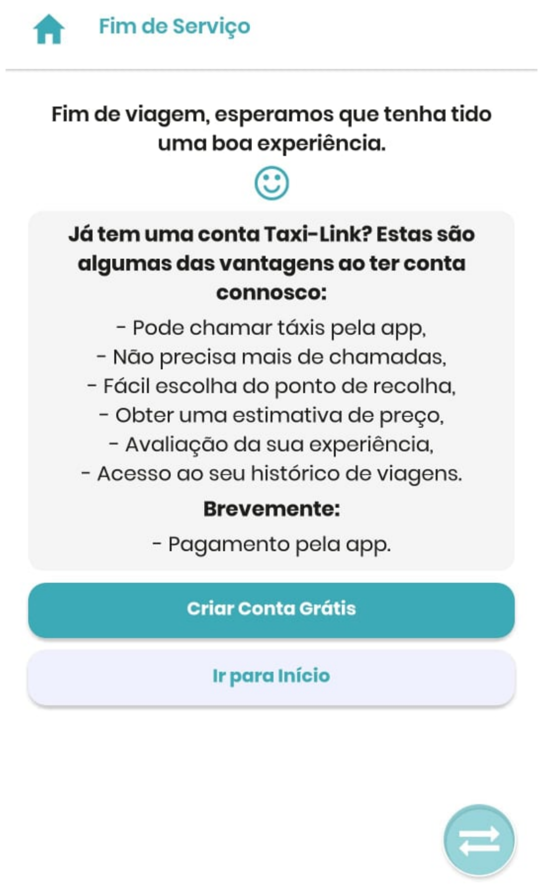
Ver o meu táxi

É a primeira vez que usa a Taxi-Link neste dispositivo? Descarregue a aplicação Taxi-Link na Google Play Store ou diretamente em "Faça download da Taxi-Link" abaixo.

Insira ID: 1234abcd após instalar a aplicação Taxi-Link.

Faça download no Google Play

Figure 5.2: The new webpage for both iOS and Android smartphones.



Fim de Serviço

Fim de viagem, esperamos que tenha tido uma boa experiência.

Já tem uma conta Taxi-Link? Estas são algumas das vantagens ao ter conta connosco:

- Pode chamar táxis pela app,
- Não precisa mais de chamadas,
- Fácil escolha do ponto de recolha,
- Obter uma estimativa de preço,
- Avaliação da sua experiência,
- Acesso ao seu histórico de viagens.

Brevemente:

- Pagamento pela app.

Criar Conta Grátis

Ir para Início

Figure 5.3: Screenshot of the Taxi-Link applications end of service screen, when tracking a SMS service.

5.2 Migration Results

When this system was complete we started rolling it out for some regions, and we will now present the results achieved by this strategy. This is only a preliminary analysis where the data used for analysis ranges from the start of the rollout at the beginning of May to mid-June. The information that was logged to help us in this analysis was the following, we can check in which services the link was clicked and if the corresponding token was used to open a socket. Opening the link tells us that the clients were subject to the publicity about the passenger application and, knowing the ones that tracked their service grants us that those clients did install the application. After defining this, in order to know how many of these clients registered a Taxi-Link account, we checked the phone number of new registrations and compared them against the contacts associated with these phone services. Then we also analyzed how many of these clients already completed requests using the passenger application and how many services in total where made by them, while also comparing the data generated by these clients against the rest of the user base.

Fleet	**SMS Services**	**Link Clicked**	**Tracked**		**Unique Clients**	**Unique Clicks**	**Unique Tracked (Unique Installs)**
Porto	10084	417	113		4673	290	52
Lisboa	7003	912	143		3926	749	92
Sintra	7006	199	20		3810	183	18
Braga	1674	49	10		1352	43	8
Aveiro	256	34	4		185	32	3
Madeira	194	41	10		157	36	8
Matosinhos	49	2	1		43	2	1
Total	**26266**	**1654**	**301**		**14146**	**1335**	**182**

Table 5.1: Number of SMS services shown in correlation with link clicks and tracked services. For both the total of services and unique clients only.

In the table 5.1 our analysis tells us information about these services with the new system, telling us how many services were done by a fleet, like we said before some fleets had this service for longer than others, Matosinhos for example only had it for two days previous to this analysis. We can also see here in how many services the clients clicked those links and, also how many of them actually started tracking their service. To make more sense of this data we have also distinguished how many unique

clients did request these services, and also how many unique clients did click the link and started tracking their service.
We can take some insights on how the system is doing, 9.4% are clicking the link at least once. We can see that it happens way more often in Lisboa than in Porto and Sintra for example, an explanation for this is that in Lisbon pickup times are usually higher, which can lead to a desire to track the service. From these 9.4%, 13.6% track the services and have installed the application.
We have also checked how many of these users that clicked and tracked the services actually installed the application, and this was the result:

- Clicked (tracked or not) - 127 (9.5% of total unique clicks, 0.9% of total unique clients),
- Tracked - 55 (30% of total tracked)

So almost 10% of the clients that click the link register an account, representing almost 1% of the total unique clients that got the SMS at least once. Clients that do track the service made an account 30% of the time, indicating that actually having them using the application might increase the rate of registration.

In the next table, we will analyze how many of these registered clients actually requested services through the application.

Requested Services	**Successful Services**	**Canceled Services**		**Migrated Clients Services**	**Migrated Clients Successful Services**
1084	646	438		242	176
100%	∼60%	∼40%		∼22% of total	∼27% of successful total

Table 5.2: Table representing quantitative analysis of services done by normal costumers and migration ones.

Here it is important to consider that from our total of 1374 registered users (with activated accounts), the migrated users represent only 9.2% (127 users) of this total. As we can see from table 5.2, these users make up 22% of the total requests (canceled or fulfilled), and, when analyzing only successful services we can see that these users are responsible for 27% of our total of successful services. By these numbers, we can see really how important this migration process is since these clients are starting to have a positive impact on the platform. Even though these users only represent a fraction of our current user base, they have already had a big impact on the number of services.

Chapter 6

Discussion

In this section, we will discuss our results, problems we have encountered along the way while also talking about future work. One thing that delayed us was the covid-19 pandemic, which did have a considerable impact on our platform, there was a reduction of about 80% to 90% in the number of services our platform served, which did also impact the analysis of the results since the volume of services was not the expected one.

Although as stated in section 5.2 the time to generate data to evaluate the migration system was relatively short, we think the results from that analysis were promising and expect better results when the number of services starts going back to normal.

In terms of future work, there are some ideas we think might be interesting to explore. One area that is of great importance is to protect our drivers from careless clients that cancel services after being assigned a taxi, especially when it is done several minutes into the pickup stage. If a taxi is in a taxi stand, when it leaves it its position is lost, which is a relevant factor in having priority for the allocation of services, and when they are picking up a customer they are also wasting time and gas (energy in the case of electric cars), so in this perspective, it would be very interesting to build a penalization system that would aim to discourage clients from canceling services, though this is a delicate problem to handle. At the moment what we have is, depending on the stage of the service, we show a message warning the clients of how their cancellation negatively impacts the taxi driver.

Assignment time is also very relevant, as it directly correlates with client satisfaction, and while it is subject to some factors like availability which we cannot control, there are some cases in our system where drivers do not accept certain services without having a good reason, this makes it so that services bounce from driver to driver delaying the assignment, as stated in section 3.3. Therefore building a punishment system for drivers who reject services, would also be interesting to improve assignment speed, though like the penalization system for clients it is a delicate issue that has to be carefully handled.

Adding more options to the passenger application to improve the user experience and user fidelity is also something we can think of doing. Here we have developed some different ideas, like having clients having the option to choose one or more preferred drivers, which would impact the service dispatch for that specific client. Suggesting these drivers could be based on previous successful services, on the service ratings or usual locations. Another point that would also be interesting would be to actually make more use of the rating system. For example having it impact the taxi selection when assigning services by giving priority for higher-rated drivers, looking to improve taxi driver and client interactions as it would be an extra factor encouraging the driver to provide a good experience.

Chapter 7

Conclusion

Building a passenger application was our main goal, and it was achieved with success, it complements the Taxi-Link system as we expected it to, being a very important step to fully automatize the whole process of taxi calling and dispatching. React Native ended up being a great choice like we assumed when selecting it. During the development and learning processes, we did not find any major roadblocks, always finding a solution to every problem or necessity we had. In the end, we had a functional and performant application that ticked all the features we wanted to have at the conception of this project. While there were some crashes and problems at first after our discreet launch, especially for some models (mostly regarding the Android ecosystem where many models and versions are being used) all of them were corrected when identified, having the application free of known bugs at the completion stage of this project. All in all our experience with React Native was very positive and was in line with what we expected after doing our analysis.

The Google API was also very helpful, all the services we used work very well. It has great documentation that really helps in figuring out the best fit for our needs and also has great tips about implementing their services in an optimal way. So, in the end, it was also a great experience to develop this application making use of Google services.
The generated data will help the platform to improve in the future, it will help in the creating of user and driver usage and behavior profiles and, it will also help to discover flaws where the service is not being provided as expected to allow us to possibly understand where they might be failing and how we can fix or improve it.

References

[1] Call box image.
https://www.pinterest.pt/pin/367958232038897826/, Last accessed on 15.06.2020,.

[2] Call box image.
https://www.olx.pt/anuncio/rdio-de-taxi-vhf-teltronic-p-2500-IDz21Ez.html, Last accessed on 15.06.2020,.

[3] Encoded polyline algorithm format.
https://developers.google.com/maps/documentation/utilities/polylinealgorithm, Last accessed on 22.05.2020.

[4] Google autocomplete api developer guide.
https://developers.google.com/places/web-service/autocomplete, Last accessed on 03.06.2020.

[5] Google directions api developer guide.
https://developers.google.com/maps/documentation/directions/intro, Last accessed on 01.06.2020.

[6] Google geocoding api developer guide.
https://developers.google.com/maps/documentation/geocoding/intro, Last accessed on 12.05.2020.

[7] Google maps platform documentation.
https://developers.google.com/maps/documentation, Last accessed on 08.06.2020.

[8] Google place details api developer guide.
https://developers.google.com/places/web-service/details, Last accessed on 03.06.2020.

[9] Gps trilateration image.
https://electricalfundablog.com/global-positioning-system-gps/, Last accessed on 14.06.2020.

[10] Ionic framework.
https://ionicframework.com/, Last accessed on 10.12.2019.

[11] Mobile operating system market share worldwide.
https://gs.statcounter.com/os-market-share/mobile/worldwide, Last accessed on 08.01.2020.

[12] React-native.
https://reactnative.dev/, Last accessed on 8.06.2020.

[13] Rfc 4122: A universally unique identifier (uuid) urn namespace.
https://tools.ietf.org/html/rfc4122, Last accessed on 22.05.2020.

[14] What is xamarin?
https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin, Last accessed on 10.12.2019.

[15] R. Bajaj, S. L. Ranaweera, and D. P. Agrawal. Gps: location-tracking technology. *Computer*, 35(4):92–94, 2002.

[16] Judd Cramer and Alan B. Krueger. Disruptive change in the taxi business: The case of uber. *American Economic Review*, 106(5):177–82, May 2016.

[17] Gonçalo Ferreira. Uber, bolt e outras novas operadoras já com um terço das empresas e motoristas de táxis em portugal. https://observador.pt/2019/07/16/uber-bolt-e-outras-novas-operadoras-ja-com-um-terco-das-empresas-e-motoristas-de-taxis-em-portugal/, Last accessed on 03.01.2020,.

[18] Dr James Cooper John Nelson, Ray Mundy. *Taxi! Urban Economies and the Social and Transport Impacts of the Taxicab}. Ashgate Publishing Limited, 1st edition.*

[19] Z. Liao. Taxi dispatching via global positioning systems. IEEE Transactions on Engineering Management*, 48(3):342–347, 2001.*