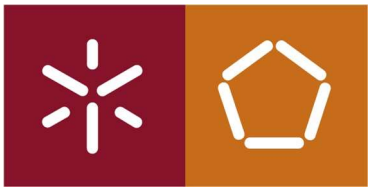


University of Minho School of Engineering Department of Informatics

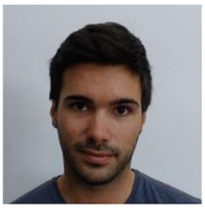
Practical work

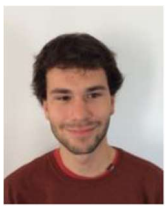
Shift 2 Group 2

Integrated Master in Computer Engineering

Informatics Laboratories IV


A67746
Bruno Freitas


A55872
João Marques


A81931
Luís Duarte


A84010
Ulisses Araújo

Braga, July 5, 2020

RESUME

The elaboration of this project focused on the development of an application directed to mobile devices, with the main focus on the management of tickets that allow the attendance of public access services, allowing their control and collection. The central source of inspiration for the design of this application, arose due to the empirical and exhaustive waiting for the collection of tickets, which aim at accessing services that are constantly being requested.

Due to the specific character of this work, it became essential to present a detailed presentation of the steps that led to its construction. In a first phase, the problem whose application it was proposed to solve was analyzed, followed by extensive exploration in the market and, finally, the construction of a logo prototype for the brand. The second phase comprised the survey of requirements to be answered, which are divided into functional, non-functional and business requirements. In a third phase, modeling was indispensable, so that it could serve as a guiding thread for the subsequent implementation of the project. In the fourth phase, the database and server were generated, until the application conceived on Android and IOS. The fifth phase focused on cost analysis and economic projection, whose results reflect a considerable profitable benefit, in a future and long term perspective. In the last phase, the need for future planning emerged, as well as for carrying out numerous tests with the ultimate goal of guaranteeing the maximum viability and efficiency of the developed system and the fulfillment of all the features promised in it. Finally, it should be noted that, as it is an iterative and incremental process where the generated application was being shaped to better serve its users, it can be designed and thus gain its deserved space in the market. as well as carrying out numerous tests with the ultimate objective of guaranteeing the maximum viability and efficiency of the developed system and the fulfillment of all the features promised in it. Finally, it should be noted that, as it is an iterative and incremental process where the generated application was being shaped to better serve its users, it can be designed and thus gain its deserved space in the market. as well as carrying out numerous tests with the ultimate objective of guaranteeing the maximum viability and efficiency of the developed system and the fulfillment of all the features promised in it. Finally, it should be noted that, as it is an iterative and incremental process where the generated application was being shaped to better serve its users, it can be designed and thus gain its deserved space in the market.

Index

Picture Index

1. Introduction

The following project was carried out in the scope of the curricular unit Laboratórios de Informática IV, present in the third year of MIEI, and has as main objective, the development of an application that responds to a certain problem. The relentless search for the best way to satisfy the end users of this application has become the basis for its design.

As previously mentioned, the application developed aims at the control and collection of tickets for a given service and this report illustrates the decisions and phases made. To begin, still in this chapter, the definition of the product to be developed is carried out, followed by how it was implemented, in the later chapters.

1.1. Problem definition

The first process in the realization of a computer application must always be the identification of a problem present in society, so that the application is useful and solves it. Proof of this is the scenario facing the software industry at the moment. This industry has experienced an explosion over the past few years and is increasingly being exploited for profit. In this way a lot of software is made every day, and a lot of it is not even used. Now, huge amounts of human and monetary resources cannot be wasted in this way at all, when they could be redirected to solve a real problem, from which a large part of humanity would benefit from this resolution.

In order to reinforce this idea, a diagram with statistics on the use of developed software is presented below:

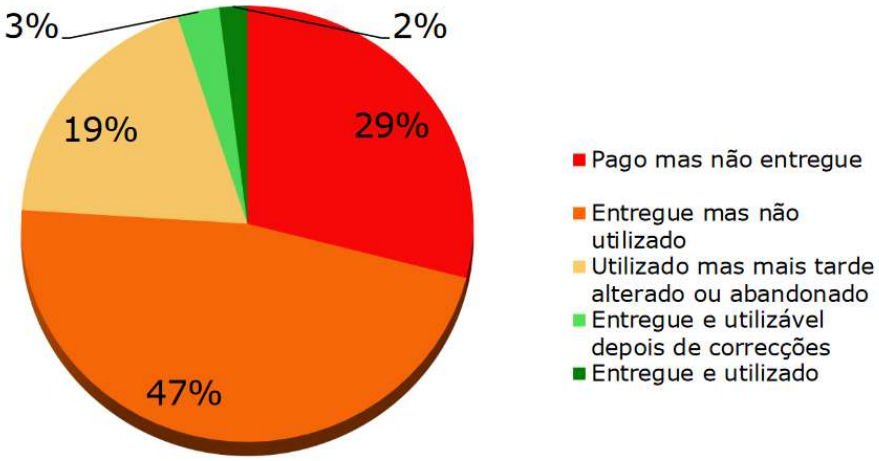


Figure 1: Graph of the use of developed software

Therefore, the group first started by identifying a problem / theme that would indeed be an asset. This process is described below.

1.2. Problem

What problem can one then choose to solve? In fact, it is a simple and complex question at the same time, as all of us, in our daily lives, face situations that we would certainly like to be resolved. However, to what extent could an application extinguish it? This is the type of problem that is sought so that it can be solved.

Thus, the problem analyzed and which we propose to solve, is as follows: "Disorder caused by waiting lines to access services". Basically any public access service, from the supermarket butchers to the social security services, are based on queues to control the priority in their access, which means that any citizen has to wait in a virtual or physical queue, wasting time , which could not only be used for another activity, but also annoys the person concerned.

In this way, we will try to carry out an application that bypasses this, and the whole process will be detailed and explained later.

1.3. Title and Framework

The name of the project is “IQueue, smart ticket management” and it is a mobile application that fits into the priority and access control, in services based on queues.

The application will include not only a service, but all those who want / allow the application to cover. In this way, a user can, in just one application, control access to various services, so that it is decentralized from any service / activity.

Figure 2: Application logo

1.4. Goals

The main purpose of the application is mask the upheaval caused by waiting lines leading to greater user satisfaction. In this way, it is intended that the application allows intelligent and easy control by the user, in turn, when accessing a system. An example of this is the control of the ticket that a user makes when he wants to take advantage of the Academic Services of the University of Minho, be it from the part of removing a ticket in person, to the management that the user has to do with it, that is, to be constantly attentive to the ticket that has priority at the moment, so that when your turn comes, it will enjoy what you want.

Another fundamental objective is to allow the user to know the congestion of the service without having to be in person in it, together with the provision of various statistics that allow the user to decide the best time to enjoy this service.

1.5. Market Study

Another fundamental aspect to explore before diving into the development of the application is the current market status. In this topic there are two key issues to take into account. The first is: Does the application have the potential to enter the market? Considering what was previously mentioned and analyzing the problem it solves, we believe it is, as it is useful and beneficial for society.

The second important aspect is to understand if it has already been developed by another entity, and if so, if there is scope to do better. In this context, a research was carried out and three similar applications were identified. However, all of these have their limitations presented below:

"Ticket manager": This first competing application is American, and focuses on the commercial tickets market, making it easy to manage them, from being able to store them, to using the tickets through the application. However, it does not allow queuing control or provide service statistics. It is only intended to store the commercial tickets of the user who uses it. Example of the type of useful tickets to keep in this application, are train tickets or tickets to go to see a football game. "CitizensMap": This application works in the field of Portuguese social security. Allows the user to withdraw a ticket to a store of their choice. But, it presents serious problems, since not having a simplified use until some functionalities are constantly not working. "App Continente":

Finally, there is an application targeted for use in "Continente" supermarkets. This application already allows you to take tickets to access services such as butchers, and has an acceptable and simple operation. However, in addition to not providing detailed statistics, it is too centralized, in the sense that it only works for "Continente" supermarkets, and nowadays, having an application for only one service becomes obsolete, as the user for each service he wants having access would have to have a specific application, resulting in a huge number of applications unnecessarily.

2. Survey and requirements analysis

Requirements gathering is vitally important in software development, being the first technical activity to be carried out. It aims to define what is expected for the application, and ensure the satisfaction of the target audience. This step includes not only the definition of the features to be implemented, but also the characteristics that the software must meet and business requirements.

These are then presented, divided into functional requirements, which represent the functionalities of the system, non-functional, which represent how the system should be, and finally, business, which represent how services can use the application.

2.1. Functional Requirements

1st: The application must be able to support three different types of accounts, these being the common user, service manager and employee of a service. Each of these types contains different functionality.

2nd: Each action produced by an entity in the application, such as answering a ticket, must update all necessary information automatically, making it available immediately.

2.1.1. Functional Requirements for all account types

3rd: In order to access all the functionalities related to the type of account he has, the user has to login, using his own credentials.

4th: Similarly, each user can log out.

2.1.2. Functional requirements for the type of common user account

5th: A common user can create an account providing their personal data including, necessarily, their mobile phone number to ensure that a user does not have

multiple accounts. The system must also have a form of authentication in order to ensure that the mobile phone number really belongs to the user.

6th: If you wish, the user can check your account data at any time, and you can also edit your password

7th: The user can still remove his account if he wishes

8th: The user must be given the ability to consult the list of all services present in the application, filtering them by different criteria or searching by keyword

9: Together with the list of services, the user must be able to consult a specific service, being able to observe general information

10th: Each user must have a rating according to the ticket rate he / she wasted. In this way, critical services can choose to let access to the functionality of obtaining a ticket, only users with a certain minimum rating

11: The user must be able to withdraw a ticket related to a service present in the application, as long as he has the minimum rating necessary for the service he wants

12th: Complementing the option to remove a ticket in a simple way, there is yet another. It is the possibility, if the user wishes, to withdraw a ticket depending on the location where he is, or depending on the time he claims to arrive at the service, in order to circumvent the possibility of tickets being wasted because the user has not yet find on site. If you wish to withdraw depending on the location, the application calculates the expected time to reach the location, and merging with the data you have, removes the ticket at the right time, so that when the user arrives at the location and waits as little as possible to be served . The check-out feature works in the same way, but instead of taking into account the location, it calculates with the time of arrival at the place.

13th: At any time, the user can view the tickets he currently has, as well as a list of those he has used. It must be possible to filter this list according to the date

14th: Complementing the functionality of removing tickets, the user can cancel any of the tickets he currently has, if he does not intend to use them. In this way, you will not lose reputation as wasted tickets will not be considered.

15th: If the user withdraws a ticket and does not use it or cancel it, his reputation must automatically decrease. The classification should not, however, decrease if this ticket is not picked up in a simple way, but, depending on arrival time or location, and is wasted by system error when picking up at the wrong time.

16th: The reputation of the user must also increase according to the number of tickets that he withdraws and uses successfully, and also according to the time frame in which he does not waste any ticket that he withdraws.

17th: Each user can maintain a list of favorite services, for faster access, being able to add or remove a service from their same list.

18th: The system must also maintain information on the list of services recently used by a user.

19th: The user can access real-time statistics information for each service, such as the current congestion, the estimated service time or the estimated waiting time to be served.

20th: In addition to real-time statistics, the user can access for each service the information of his congestion history, service time and waiting time, this information being grouped by week, month and year.

21st: Finally, the user must be able to consult information about the team that designed and implemented the application, to clarify potential doubts

2.1.3. Functional Requirements for the type of manager and employee account

22nd: Both the manager and the employee must have access to the list of tickets to attend to their service, and if they select one, they can choose to attend to it

2.1.4. Functional requirements for the type of manager account

23rd: The manager must have access to the list of employees referring to the service.

24th: You can also create an account for a new employee you employ, as well as remove an account if you fire someone.

25th: Still referring to the employees he employs, the manager must be able to consult the list of tickets he has attended for each one, which can be filtered by date.

26th: The manager must be able to activate / deactivate the functionality to withdraw tickets for his service at any time.

27th: In addition to the ability to collect tickets, the manager must also have features that allow him to change the service hours, or the minimum rating required by the service for users to collect tickets

2.2. Non-functional requirements

These types of requirements are further divided into requirements regarding the Back-End or the Front-End of the application to be developed.

2.2.1. Non-functional requirements for the Front End (Application present in user's mobile phone)

1st: The interface must be simple and user-friendly to use.

2nd: Since easy use is the main objective when it comes to the subject of the interface, it must still be appealing to the public.

3rd: The developed application must support Android and IOS systems.

2.2.2. Non-functional requirements for the Back-End

4th: The entire system must be secure in order to be reliable.

5th: The system must be available 24 hours a day, 7 days a week.

6: A fast, flawless distributed system must be built.

7: The system must have a database ready to deal with the inherent competition, and capable of storing all the necessary data in a persistent way, making it available whenever necessary.

2.3. Business Requirements

The business requirements are as important as the previous ones, since they define how the final product can be used, and by whom.
With the development of the proposed application it is expected that any private business or public service can be present in the application, as long as it is desired. For this, they will only have to contact the people responsible for the application, who will handle all technical issues. They will also have to pay a certain monthly fee, depending on what they want, explained at the end of this report.

3. Planning

Any project, regardless of the area in which it operates, such as industry, marketing or scientific, deals with human resources, since these are the fundamental part in its realization. This project is in no way different. For this very reason, effective planning is essential with the aim of achieving an advantageous employment relationship in favor of better performance in carrying out the project. It is intended to achieve goals such as motivation, productivity, communication, organization, among others. Thus, making good planning from start to finish is as important as any other task. With this goal in mind, the following Gantt diagram was made to express in addition to the tasks to be performed, who will do them:

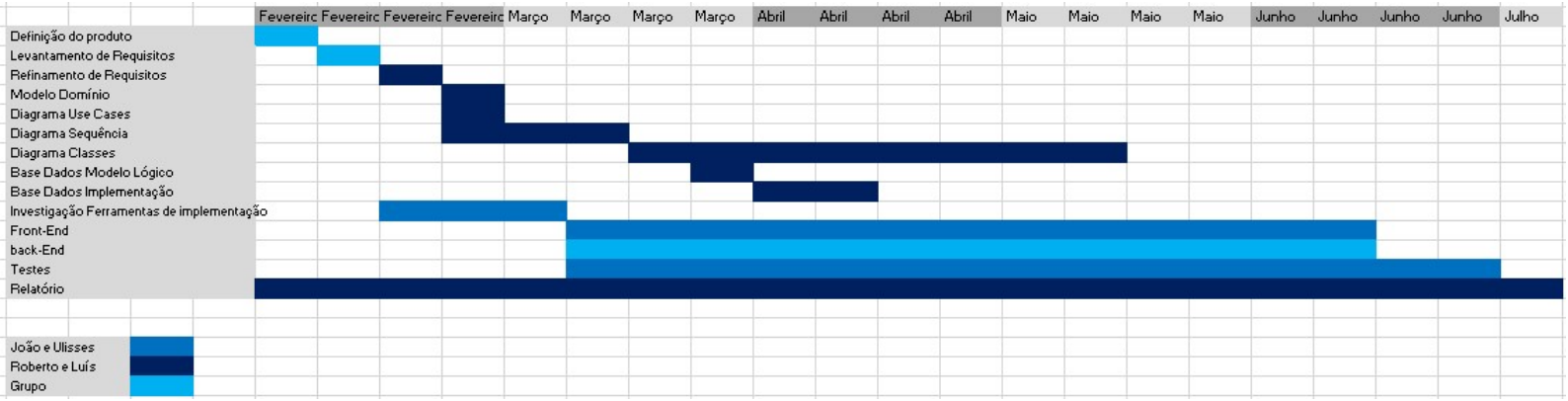


Figure 3: Gantt diagram

4. Modeling

The first big step towards the implementation of the proposed application was to perform the respective modeling. Although part of the population that programs do not adopt this phase in the design of their projects, it is in fact as fundamental a part as the programming itself, and so it was decided that it would be carried out.

The objectives that were set for this phase are to take advantage of everything that modeling has to offer to our project, such as, for example, the usefulness that a good model has to help frame and understand the problem to be solved correctly. In this way, the models are simplifications of reality and abstract representations of a system, which will allow the communication of ideas in a simplified way, either within the team designed for the implementation of this project, or between that same team and the application client. . In addition to ideas, through models, it will be possible to document all development decisions.

Next, the models made will be presented, and it was decided to use the UML language for their realization.

4.1. Domain Model

In order to capture relevant information about the project domain, the domain model was developed.

This model makes it possible to represent the framework of the problem and to reason about it. In this way the entities of the same were represented, and the relationship between them. The result is shown below:

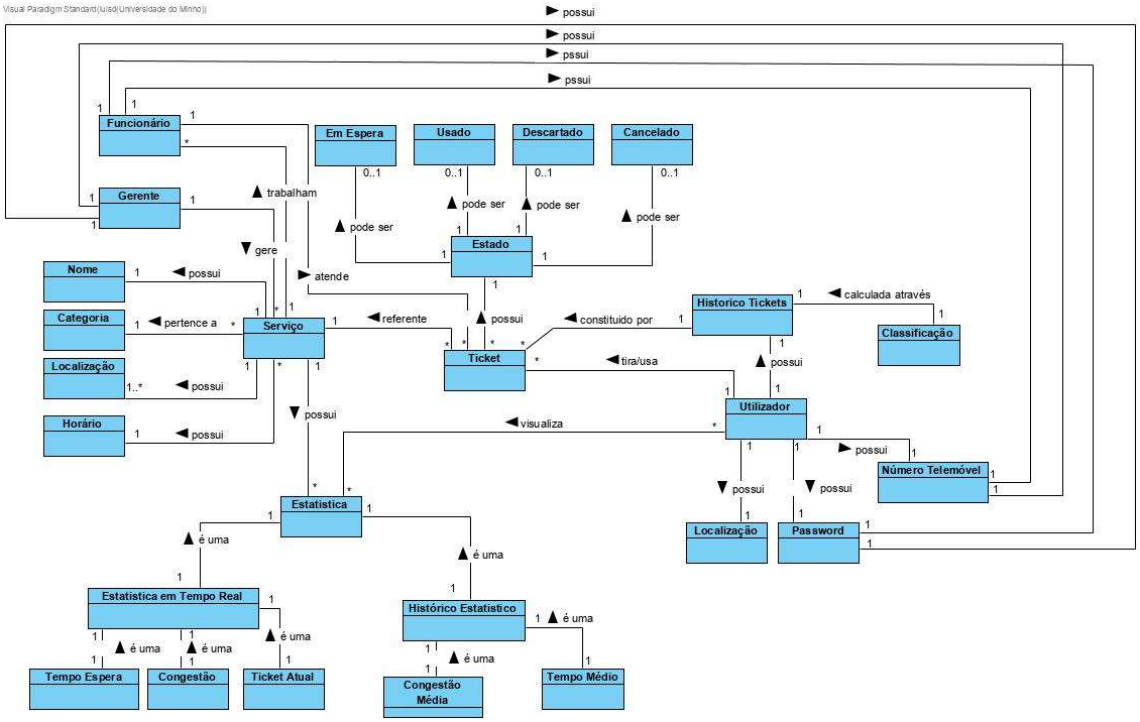


Figure 4: Domain Model

4.2. Diagram Use Cases

The use cases represent the functional requirements of the system, which generally describe the interactions between the system and the user, being basically a narrative of how the system is used.

Thus, identifying them is another very important step in the development of quality software. For that reason, a diagram is presented below that shows them:

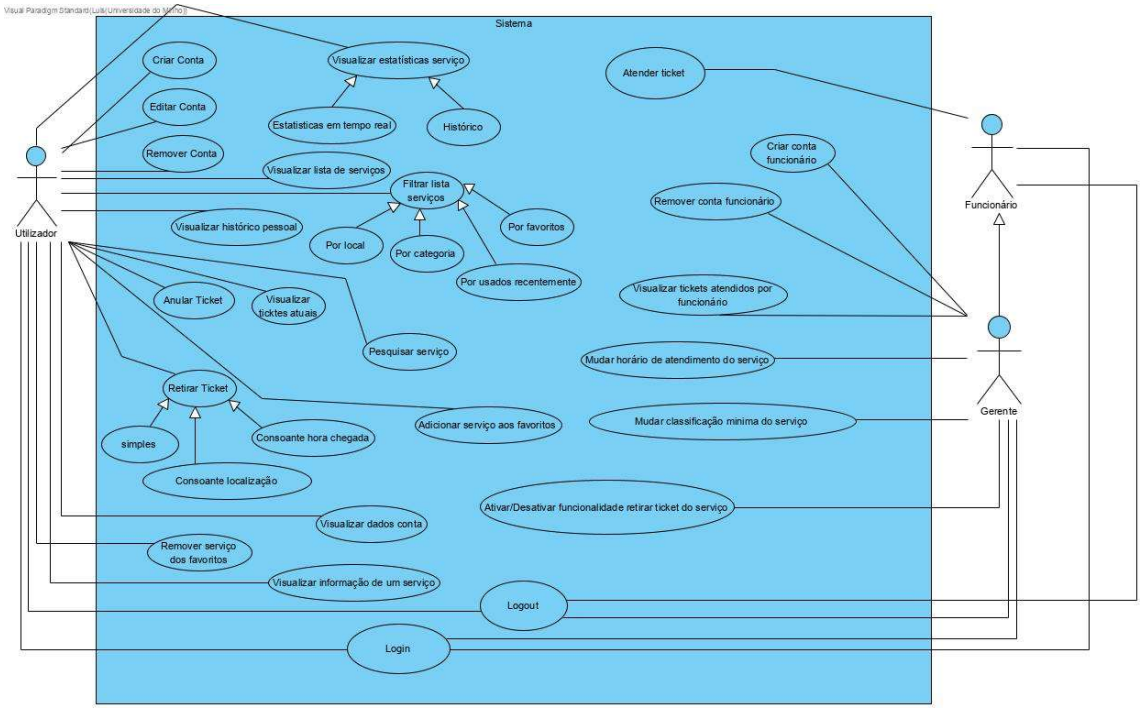


Figure 5: Use Cases diagram

4.3. Sequence Diagrams

For each of the features to be developed, a System Sequence Diagram was performed to assist in the implementation of the new phase of the project.

These diagrams allow a better understanding of the necessary flow, and the necessary interactions, for the functioning of the proposed application, with the focus being the temporal ordering.

4.3.1. Create an account

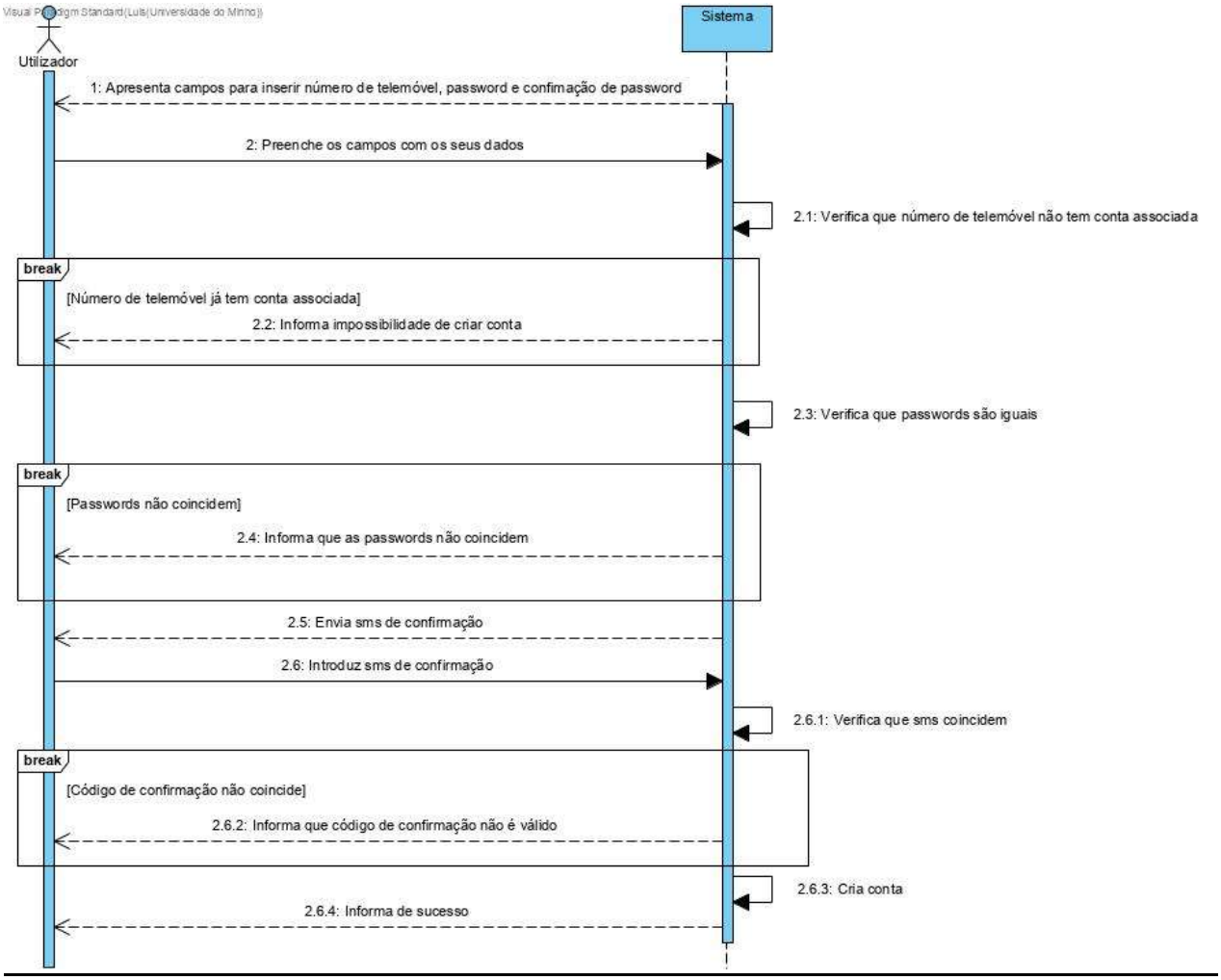


Figure 6: DSS Create Account

The “Create Account” is the use case that allows users who are not registered in the system to register, so that in the future, they can take advantage of it.

4.3.2. Remove Account

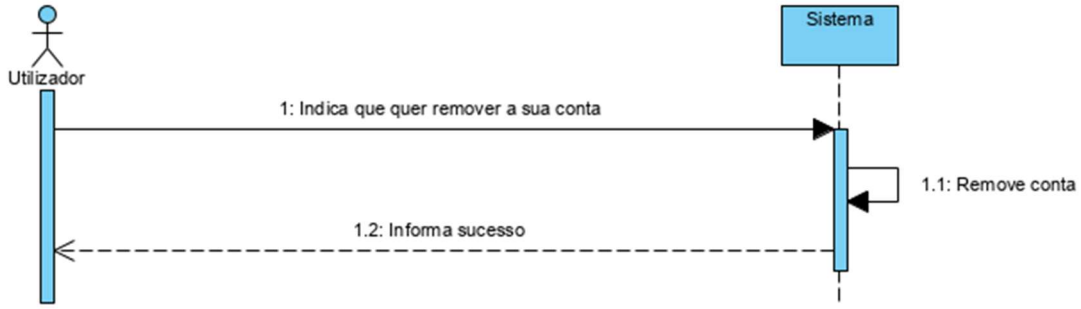


Figure 7: DSS Remove Account

The "Remove Account" represents the functionality that allows users who are on the system to remove their account.

4.3.3. Edit data

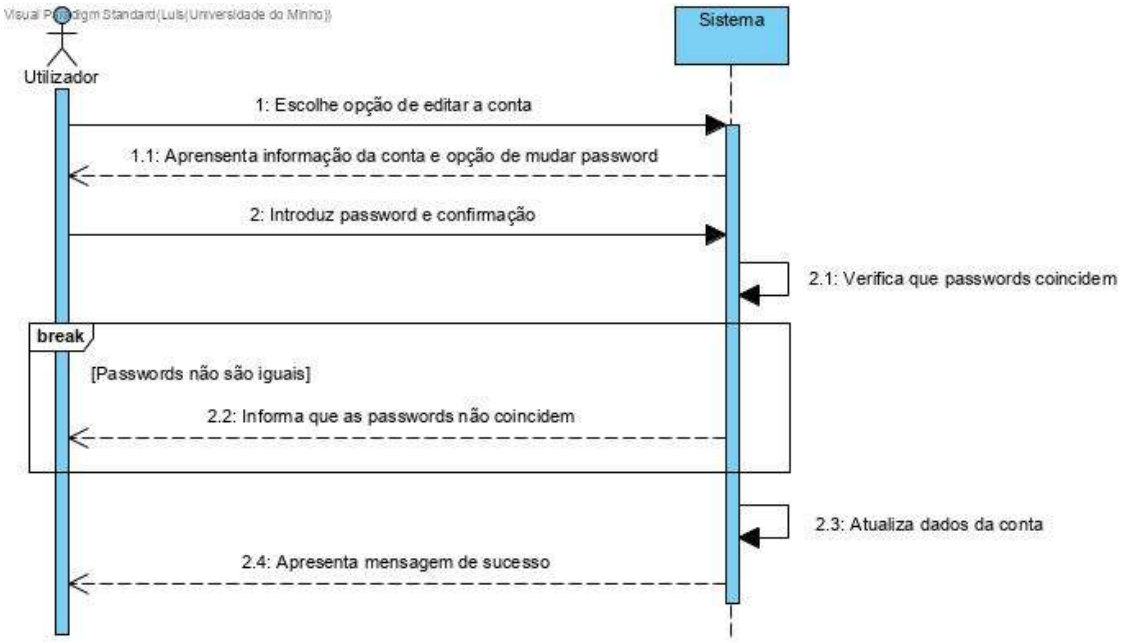


Figure 8: DSS Edit Data

The “Edit Data” is a use case that allows users who are authenticated in the system to change their password.

4.3.4. View account data

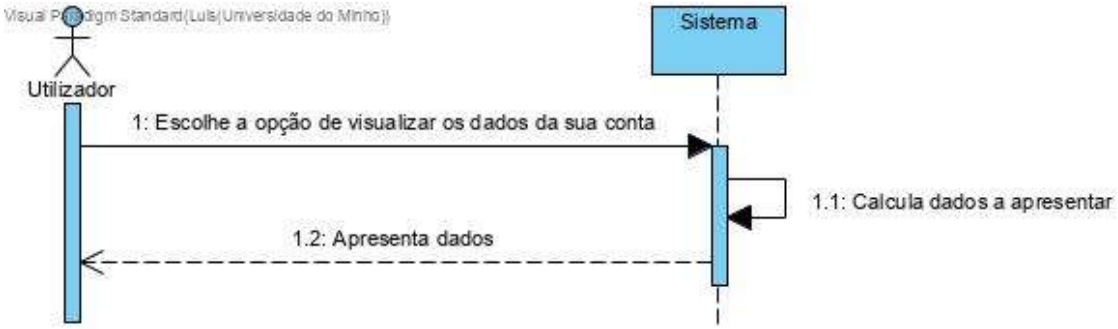


Figure 9: DSS View account data

The “View account data”, allows the user to check his personal data, such as the mobile phone number or his current reputation

4.3.5. Remove Simple Ticket

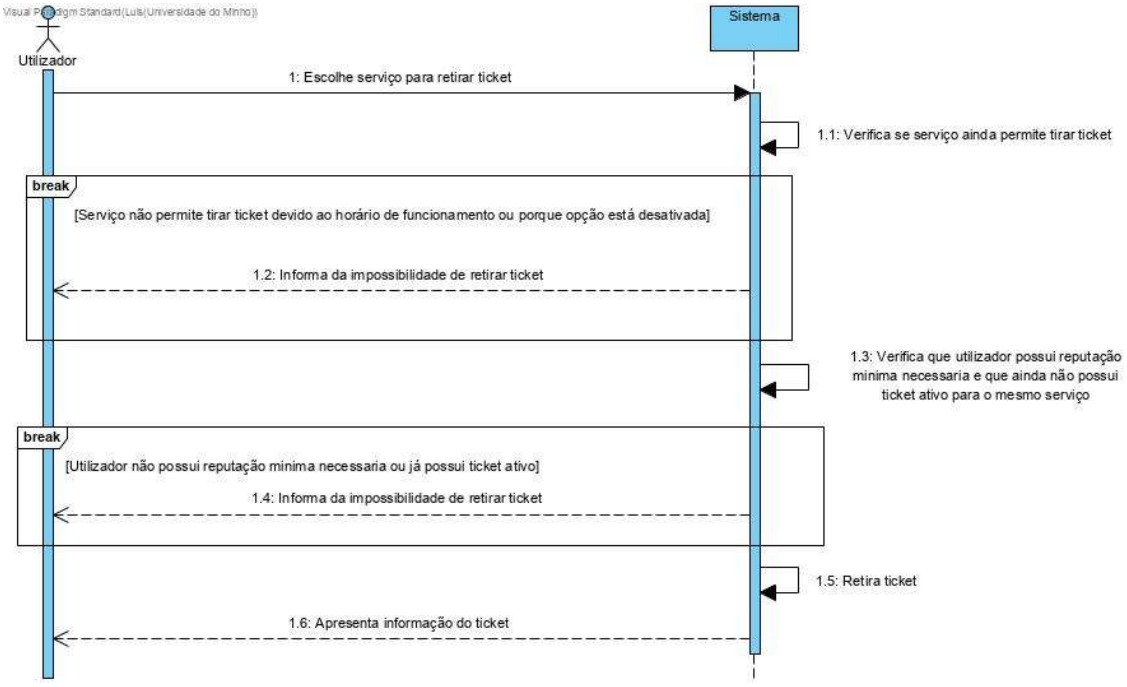


Figure 10: DSS Withdraw Simple Ticket

The “Withdraw Simple Ticket” is a feature that allows system users to withdraw a ticket in a simplified way for a certain service.

4.3.6. Remove Ticket Depending on Location

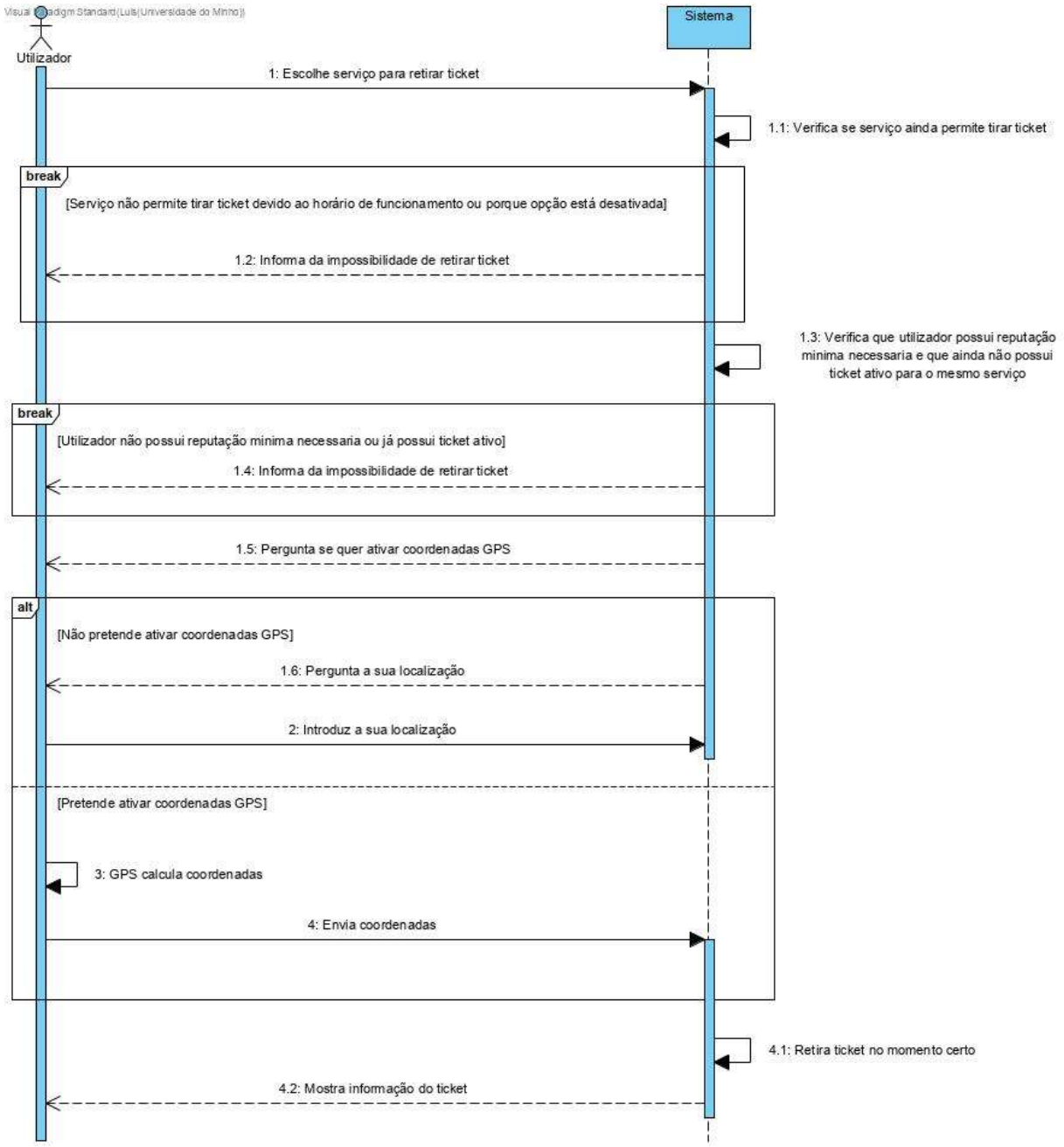


Figure 11: DSS Withdraw Ticket Depending on Location

The “Withdraw Ticket Depending on Location” is a use case that allows the user to withdraw tickets depending on the location, that is, allows it to be withdrawn automatically.

4.3.7. Withdraw Ticket Depending on Arrival Time

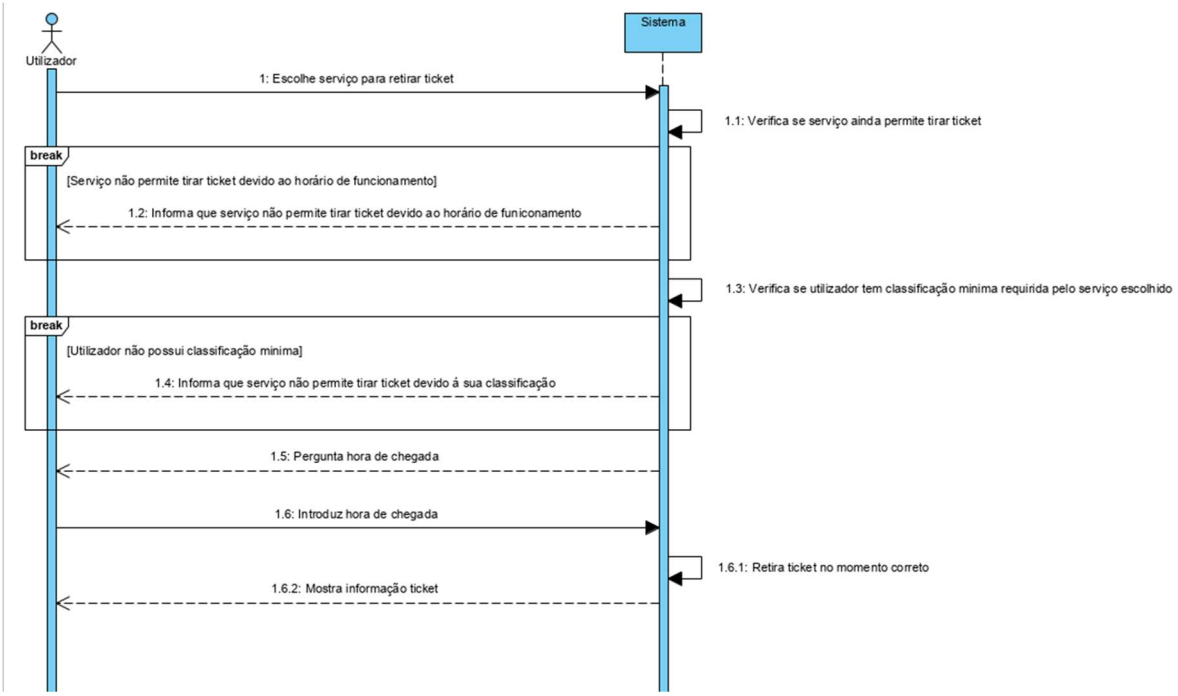


Figure 12: Withdrawing Ticket Depending on Arrival Time

The “Withdrawing Ticket According to Arrival Time” is a use case that allows the user to withdraw a ticket automatically depending on the time they expect to arrive at the service.

4.3.8. Cancel Ticket

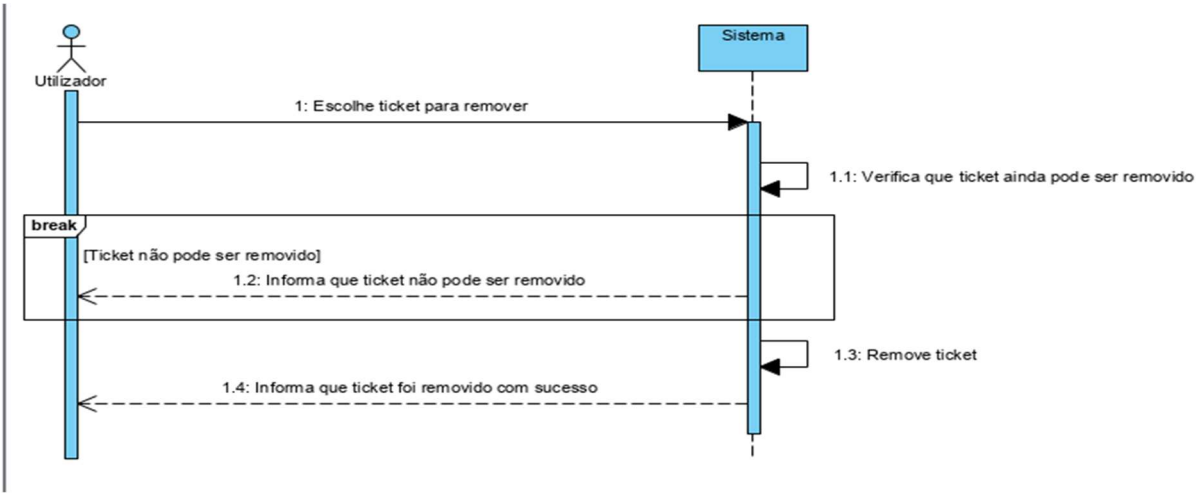


Figure 13: DSS Cancel Ticket

The “Cancel Ticket” allows the user to cancel one of their tickets

4.3.9. Search Service

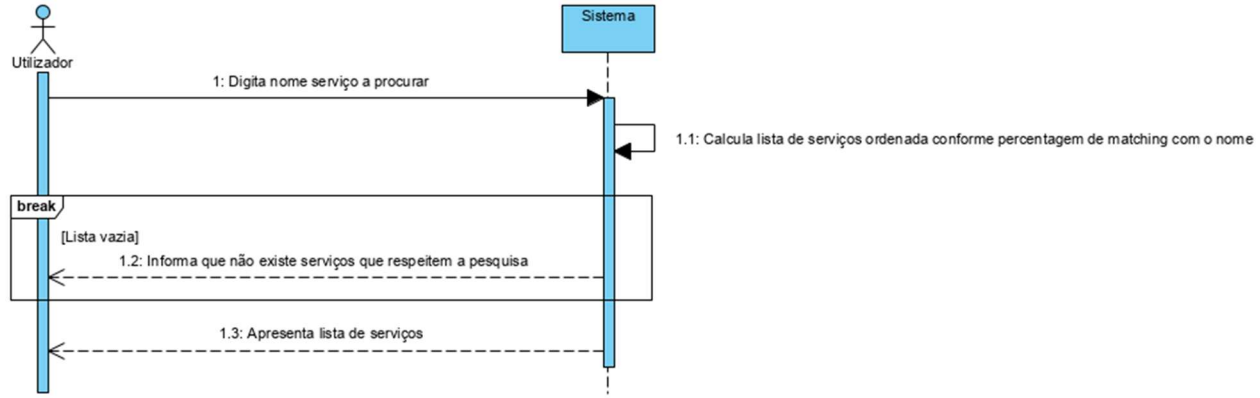


Figure 14: DSS Search for Service

The “Search Service” is a use case that allows the user to search for a service using a keyword. This word can be a service name, location, etc.

4.3.10. Add Service to Favorites

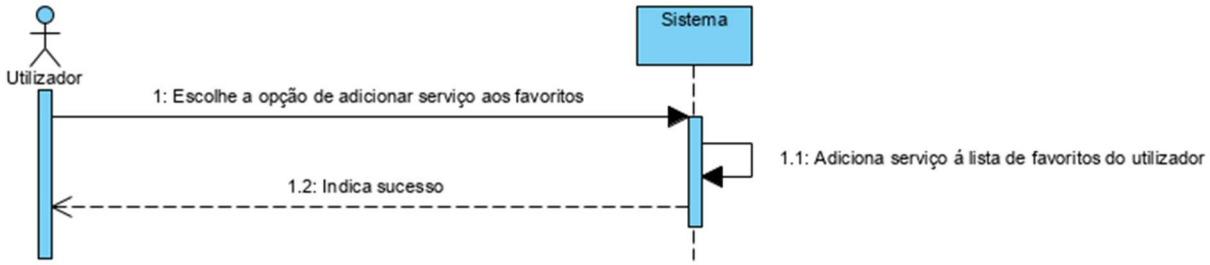


Figure 15: DSS Add Service to Favorites

“Add Service to Favorites” is a feature that allows the user to add a service to their list of favorites.

4.3.11. Remove Service from Favorites

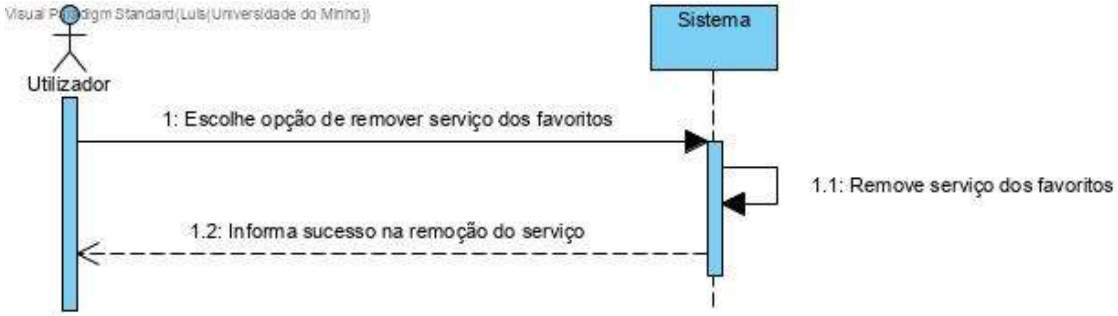


Figure 16: Remove Service from Favorites

The “Remove Service from Favorites” that allows the user to remove a service from his list of favorites.

4.3.12. View information about a service

Figure 17: Viewing information about a service

The “View service information” allows the user to collect significant information about the chosen service

4.3.13. View list of services

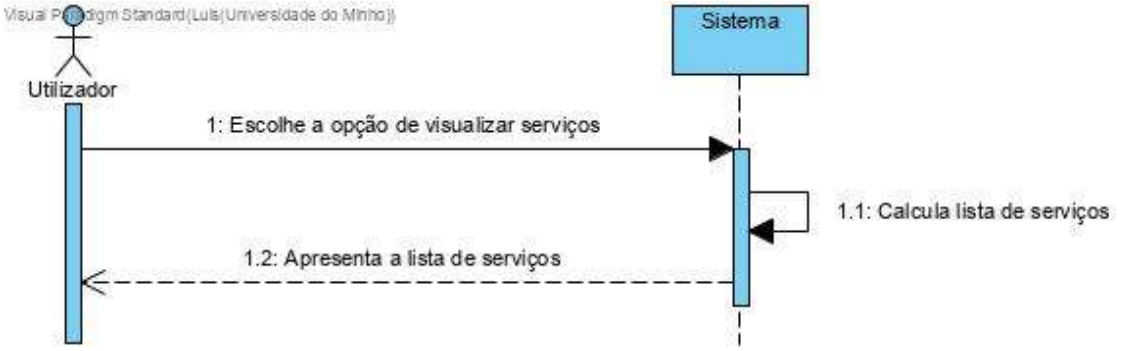


Figure 18: View list and services

The “View the list of services” is a use case that allows the user to access all services

4.3.14. Filter Recently Used Services

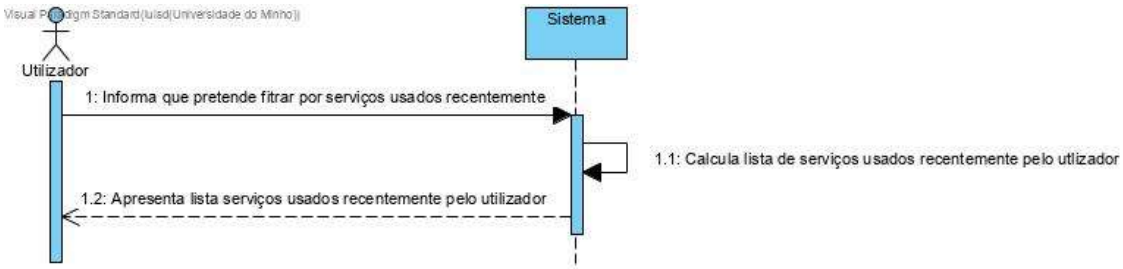


Figure 19: DSS Filter Recently Used Services

The “Filter Recently Used Services” is a use case that allows the user to obtain the list of services recently used by him.

4.3.15. Filter Services by Category

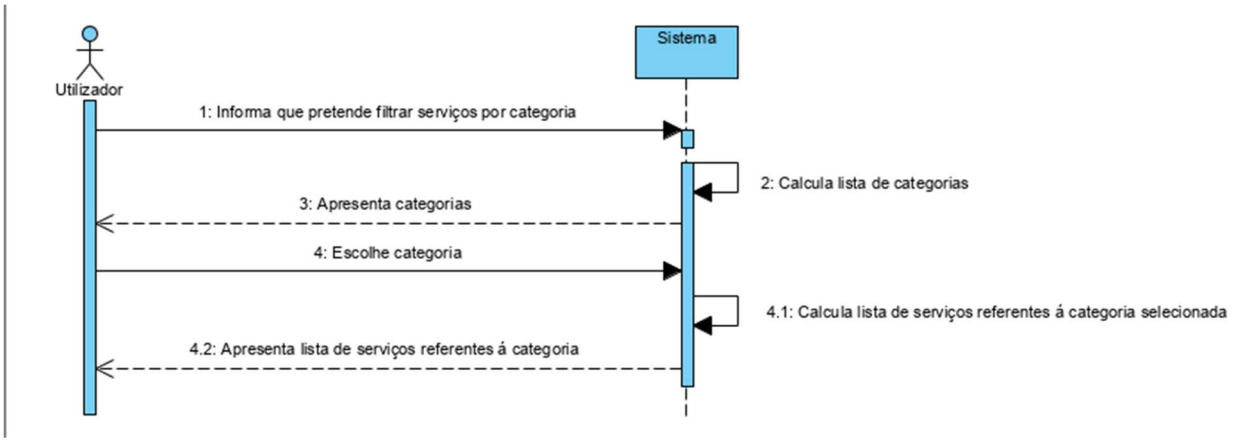


Figure 20: DSS Filter Services by Category

The “Filter Services by Category” allows the user to filter the services belonging to a category

4.3.16. Filter Favorite Services

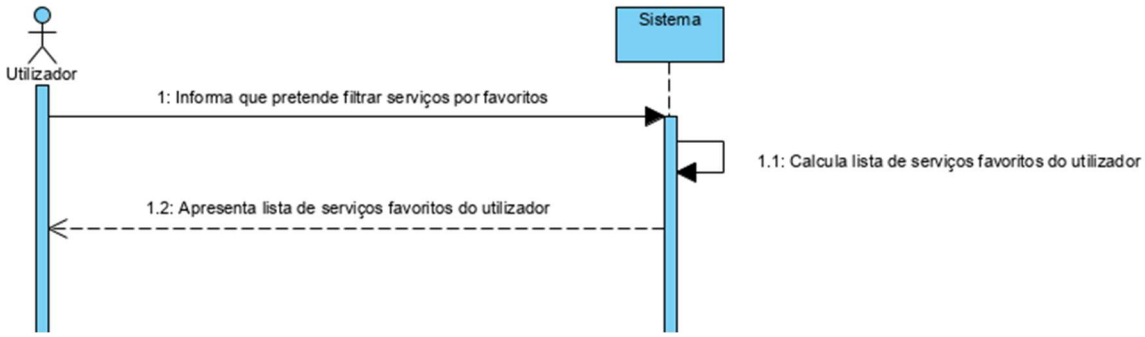


Figure 21: DSS Filter Favorite Services

The “Filter Favorite Services” is a feature that allows the user to obtain the list of services that belong to their favorites

4.3.17. Filter Services by Location

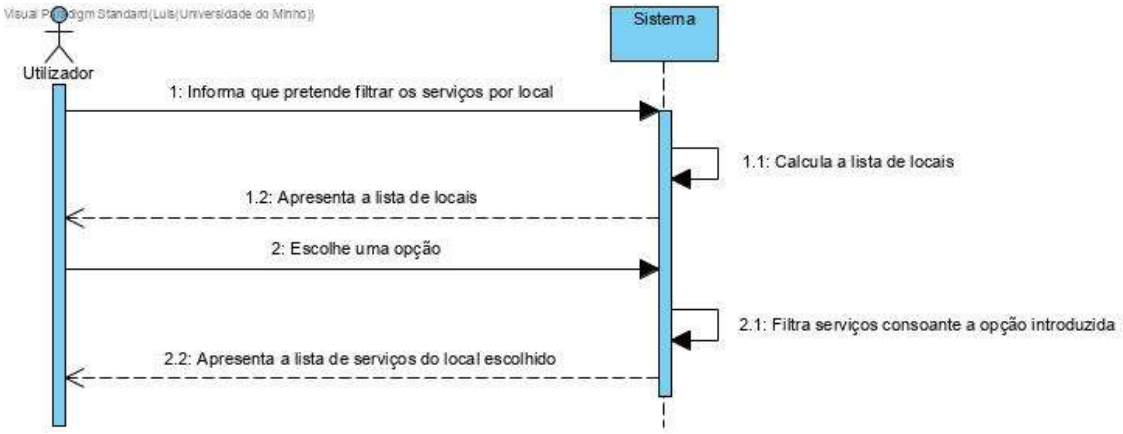


Figure 22: DSS Filter Services by Location

The “Filter Services by Location” is a use case that allows the user to obtain the list of services from a specific location

4.3.18. Real-Time Statistics

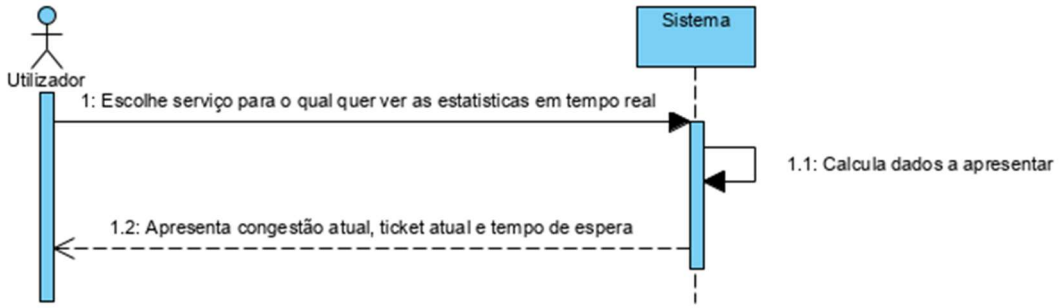


Figure 23: DSS Real-Time Statistics

“Real Time Statistics” is a feature that allows the user to see relevant information about congestion if a service in real time

4.3.19. View Service History

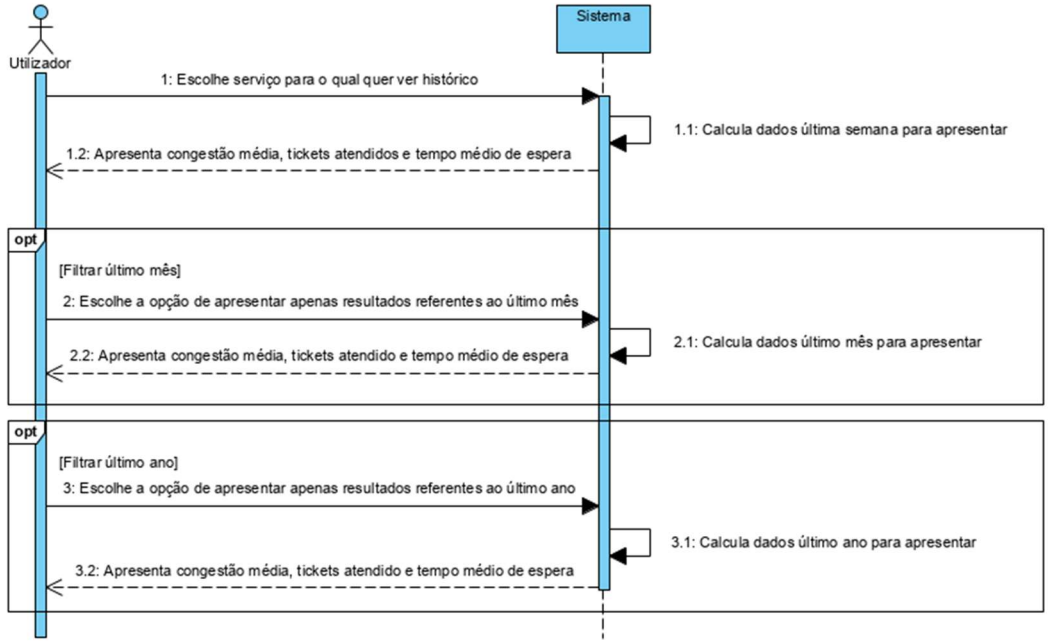


Figure 24: DSS Service History

The “View Service History” is a use case that allows the user to consult past statistics for a given service

4.3.20. View Personal History

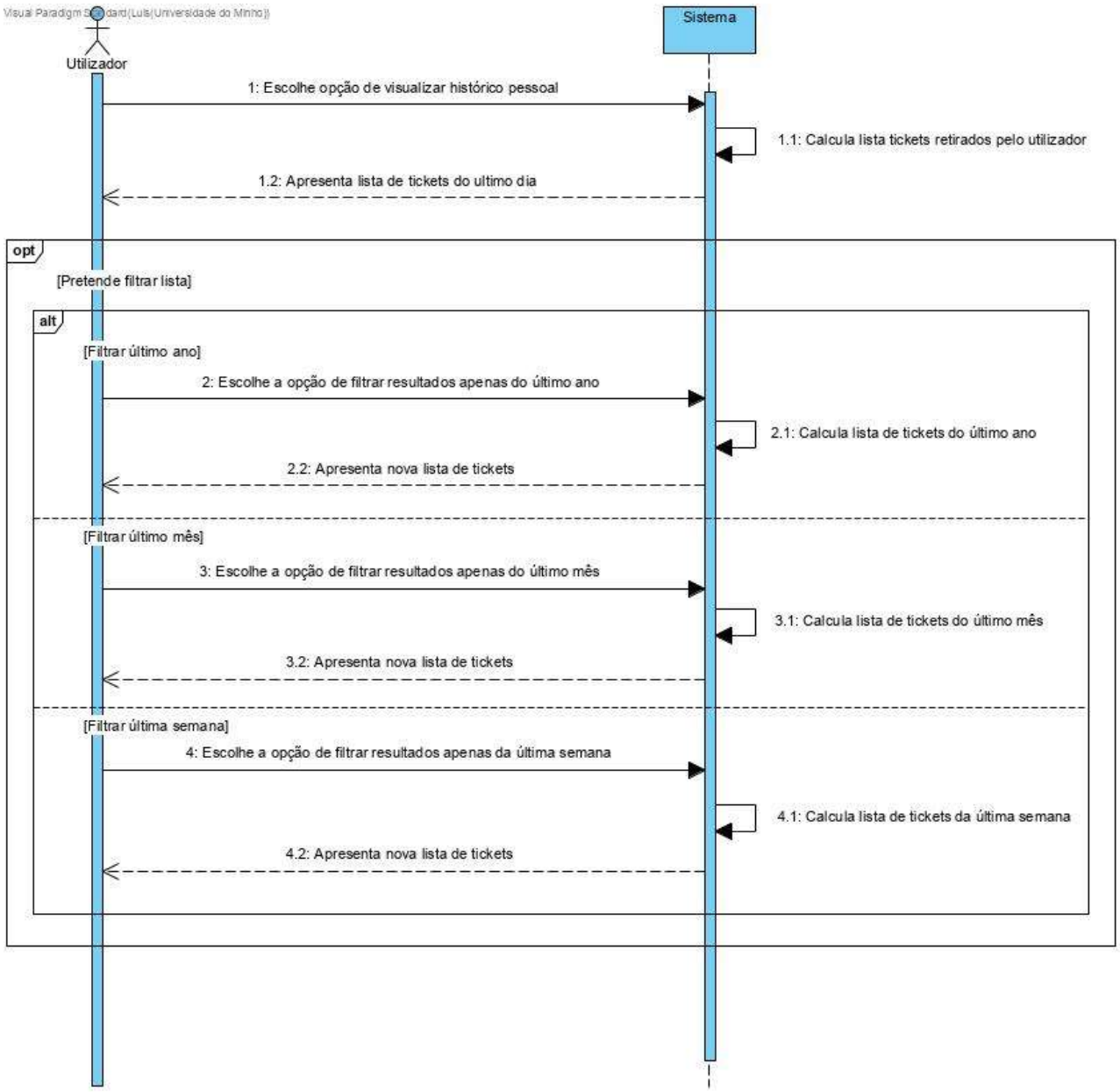


Figure 25: DSS View Personal History

The “View Personal History” is a use case that allows the user to check the tickets he has used in the past

4.3.21. View Current Tickets

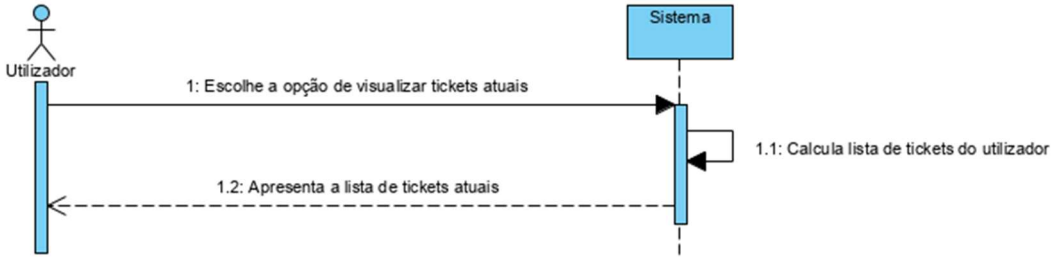


Figure 26: DSS View Current Tickets

The "View Current Tickets" allows you to consult your active tickets

4.3.22. Create Employee Account

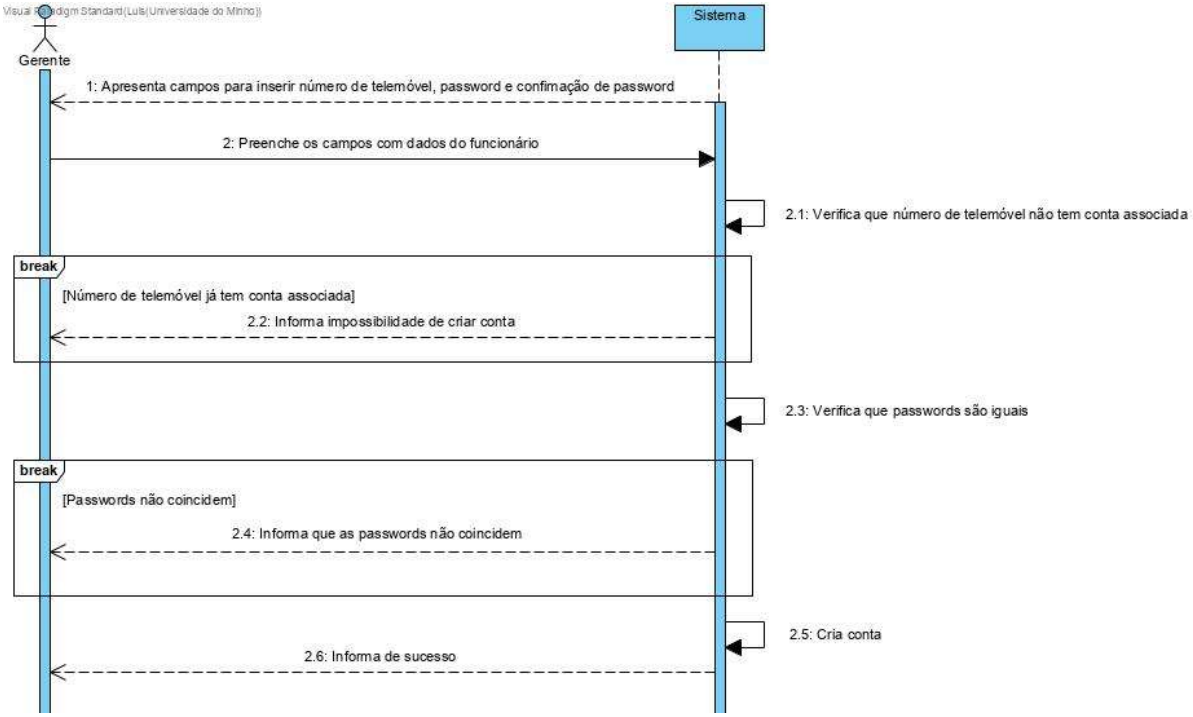


Figure 27: DSS Create Employee Account

The “Create Employee Account” is a use case that allows the service manager to create system access accounts for their employees

4.3.23. Remove Employee Account

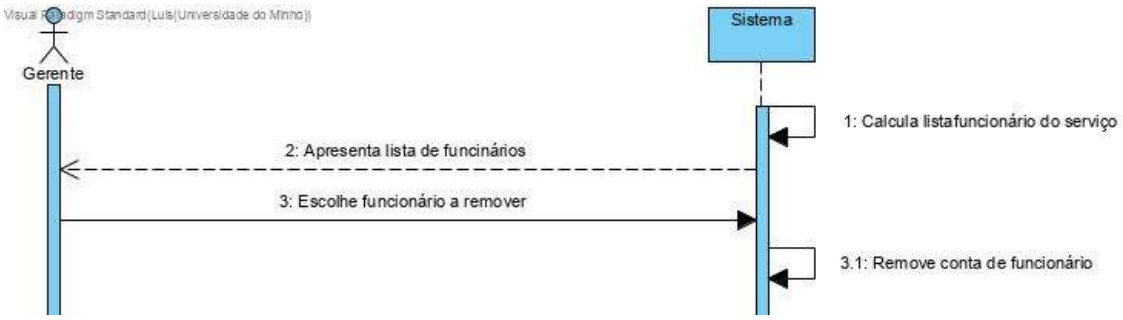


Figure 28: DSS Remove Employee Account

“Remove Employee Account” is a feature that allows the manager to delete his employees' accounts.

4.3.24. Modify Service Hours

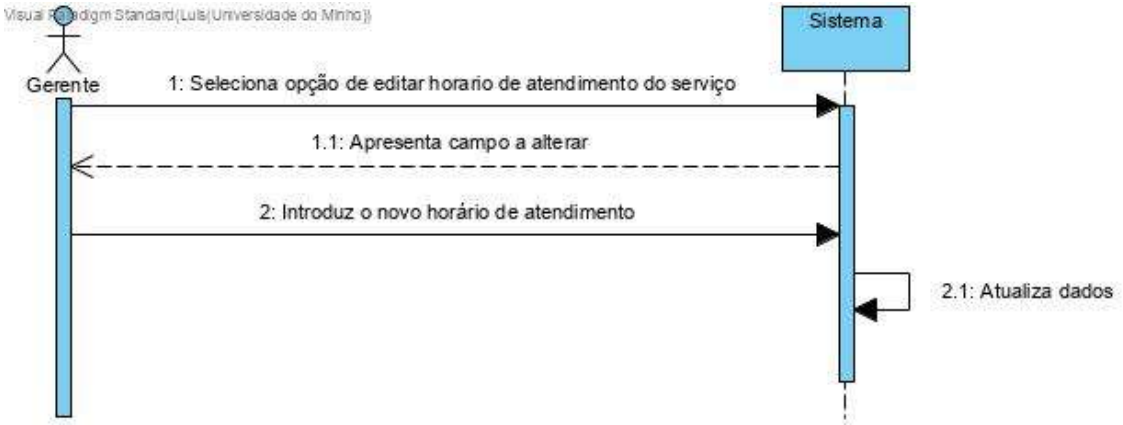


Figure 29: DSS Modify Service Hours

The “Modify Service Hours” is a use case that allows the manager to modify the hours of his service

4.3.25. Changing the Minimum Classification of a Service

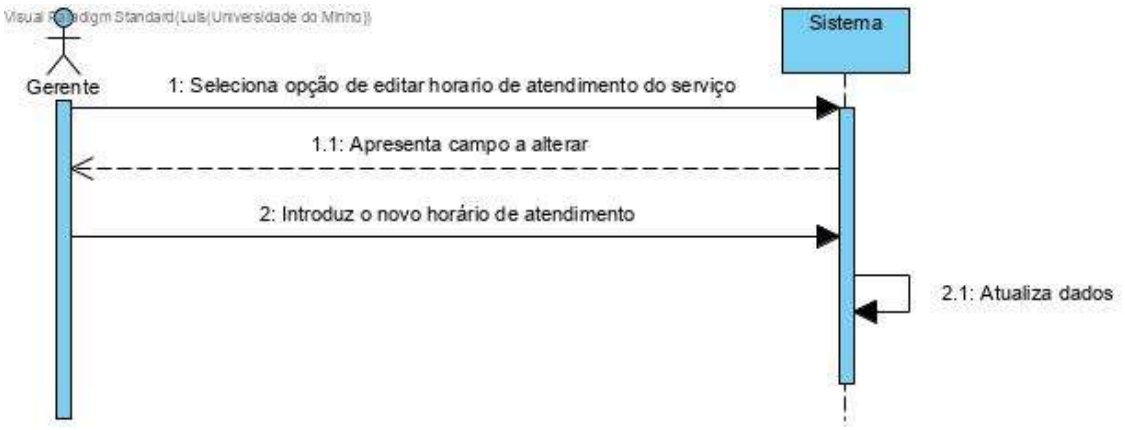


Figure 30: Changing the Minimum Rating for a Service

The "Change the Minimum Classification of a Service" allows the manager to modify the minimum classification of his service so that tickets can be withdrawn

4.3.26. Enable or disable Withdraw Feature Ticket

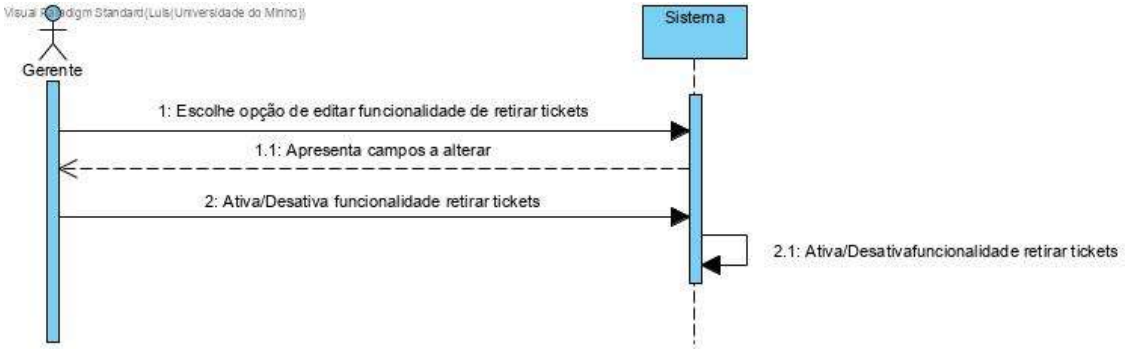


Figure 31: DSS Enable or Disable Withdraw Ticket Functionality

The “Activate or Deactivate Ticket Withdrawal Functionality” is the functionality that allows the manager to activate or deactivate the functionality to withdraw tickets from his service

4.3.27. View Tickets Served By Employee

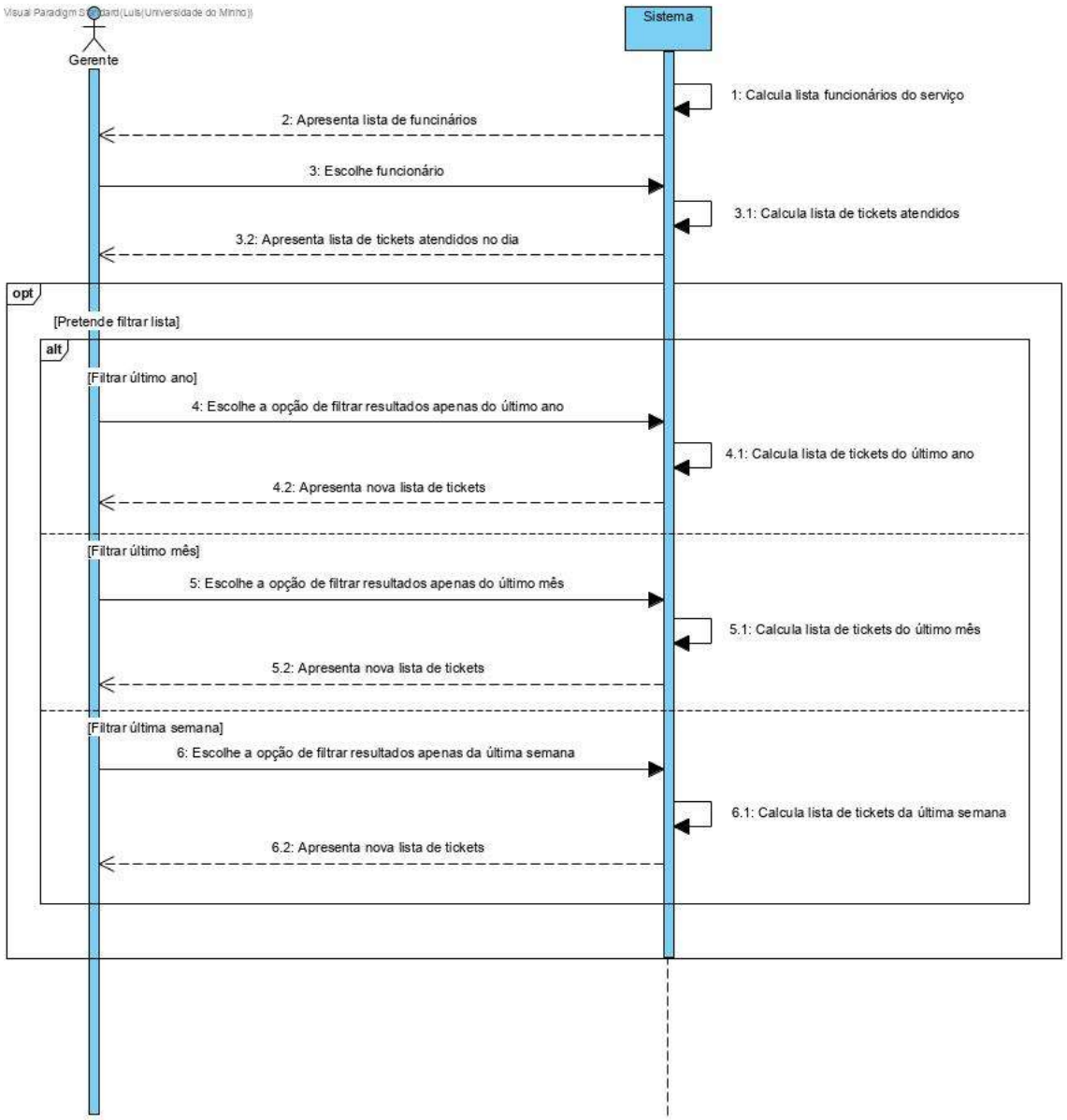


Figure 32: DSS View Tickets Served by Employee

The “View Tickets Served by Employee” is a use case that allows the manager to see the tickets served by a specific employee. The list can be filtered by time

4.3.28. Answer Ticket

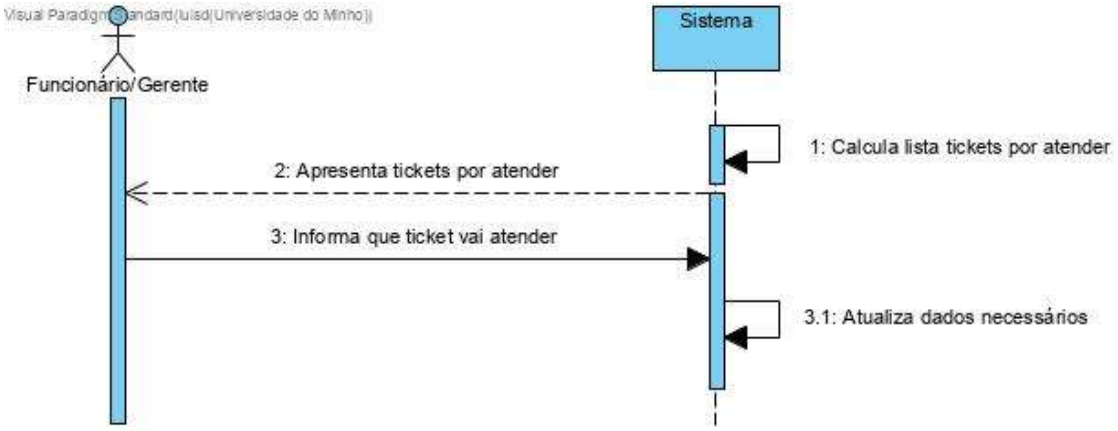


Figure 33: DSS Answering Ticket

The "Answer Ticket" allows an employee or service manager to answer tickets

4.3.29. Account Login

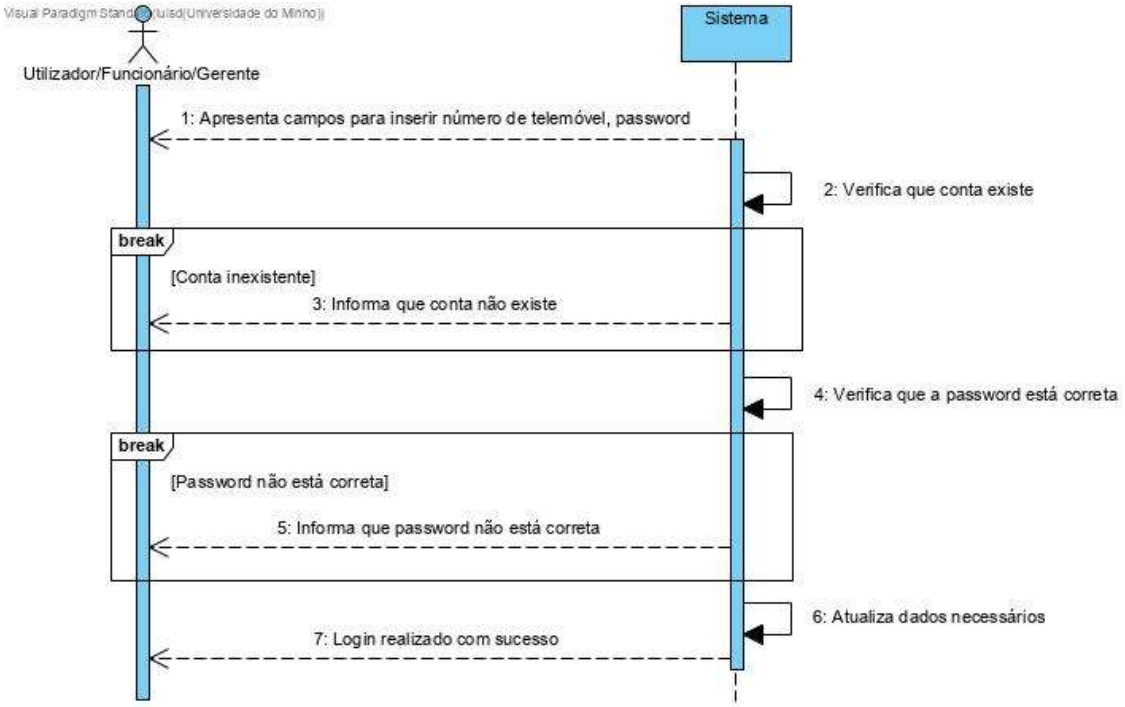


Figure 34: DSS Account Login

The "Account Login" is a use case that allows a manager, user or employee to access the content and features of your account

4.3.30. Logout

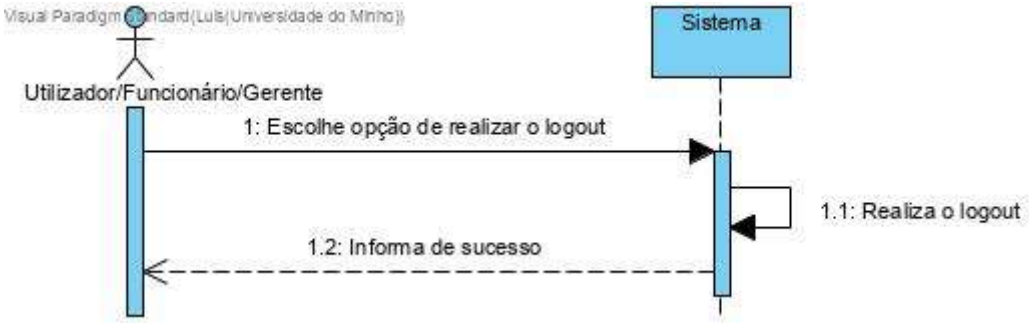


Figure 35: Logout DSS

The "Logout" is a use case that allows a manager, user or employee to leave their session

4.4. Prototype Interfaces

In order to test the system concepts, explore alternatives, and start thinking about how the system communicates with the user, employee and manager in particular, an interface prototype was developed, in which we had as main concerns trying to create an appealing interface to the user, easy to use, fast and intuitive for manager, employee and user.

In the use of the software, by the employee and manager, we chose not to prioritize the aspect of the interface, but rather, in the design of a simple and clear one, which allows the system to be configured as quickly and efficiently as possible. In the case of the user, much emphasis was placed on the visual aspect, as it is important for questions of market satisfaction and marketing.

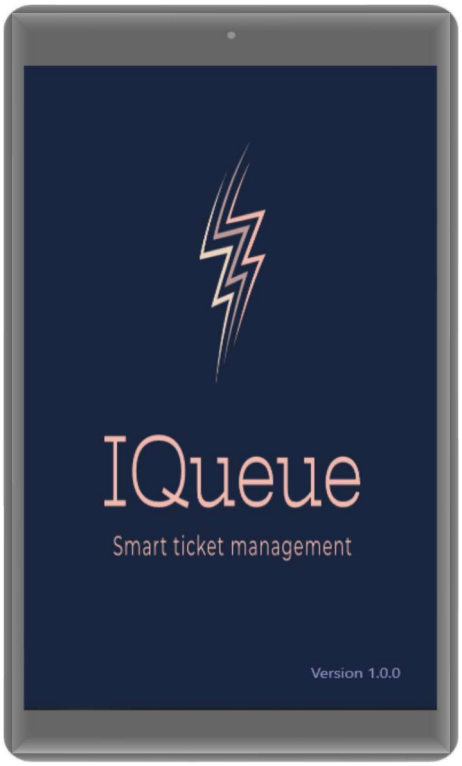


Figure 36: Loading interface

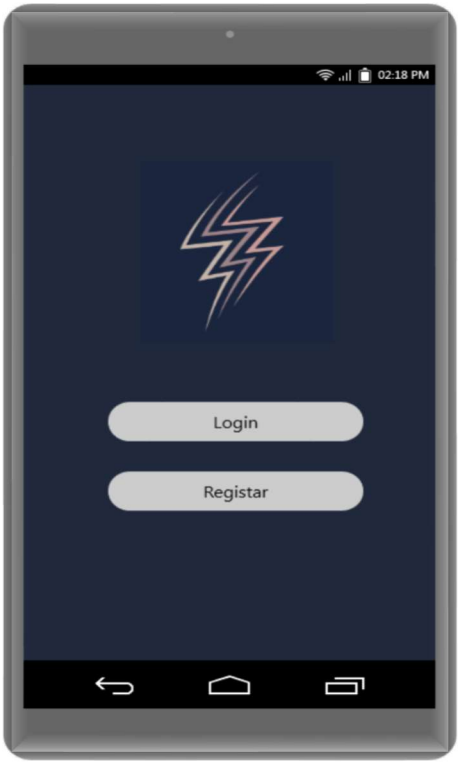


Figure 37: Home menu interface

Figure 38: Login interface

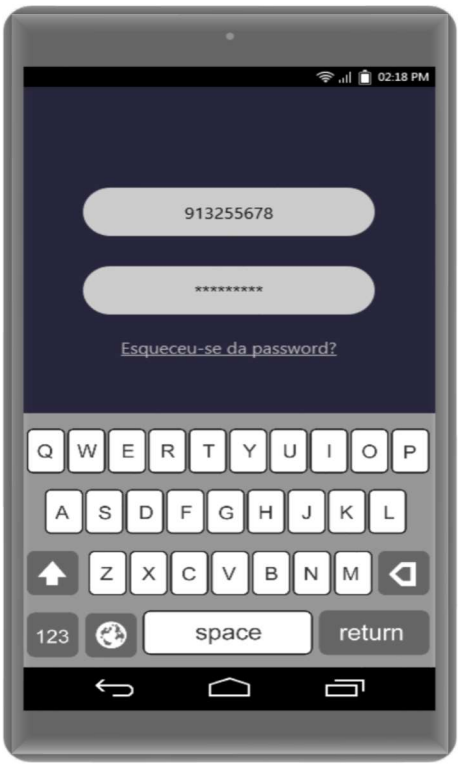


Figure 39: Inserting interface login credentials

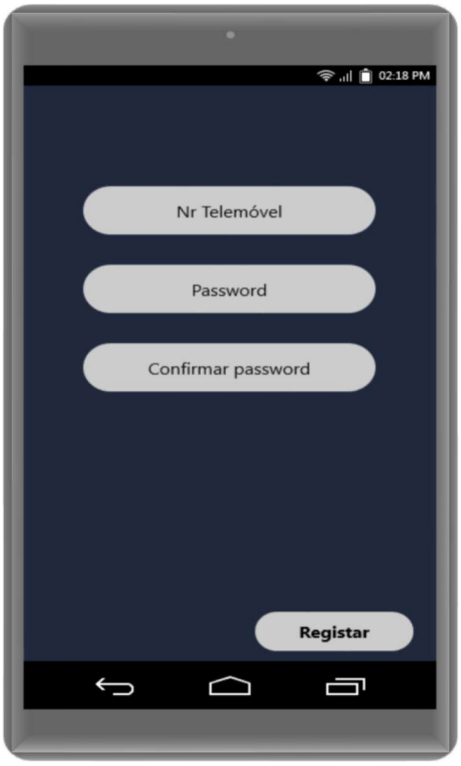


Figure 40: Register interface

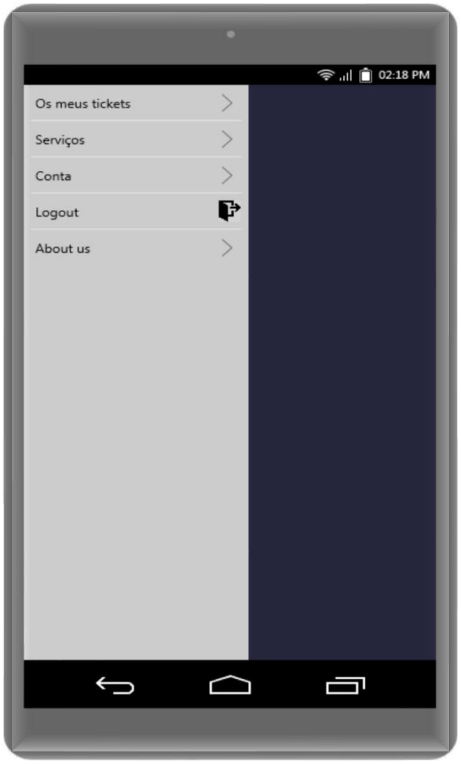


Figure 41: Home menu interface

Figure 42: Interface of my tickets

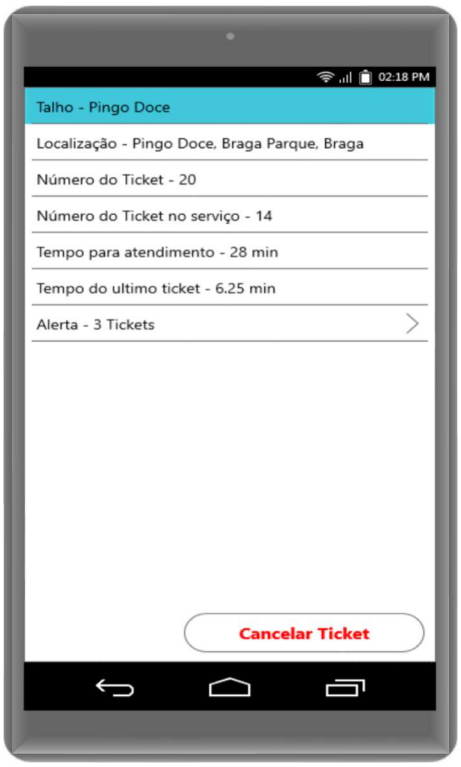


Figure 43: Interface of an active ticket

Figure 44: Canceling an interface ticket

Figure 45: Service interface

Figure 46: Statistics interface in real time

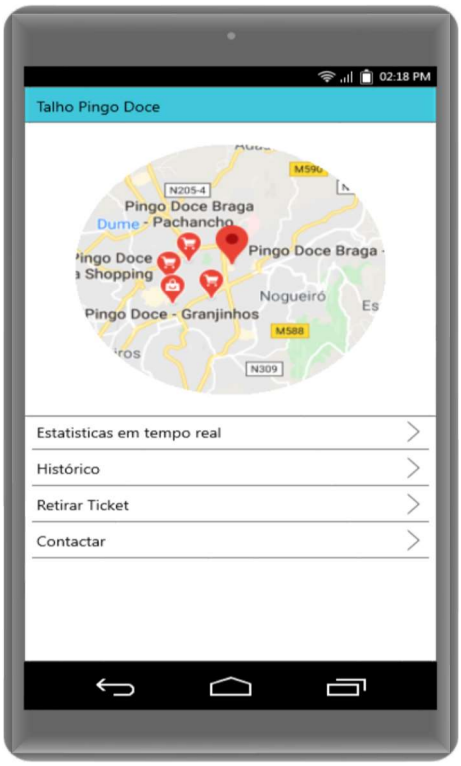


Figure 47: Active service interface

Figure 48: History interface on choice of service

Figure 49: Remove ticket interface

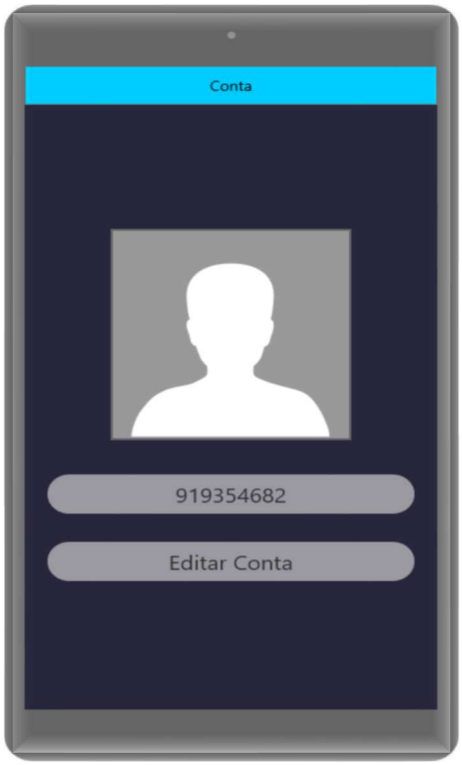


Figure 50: Account data interface

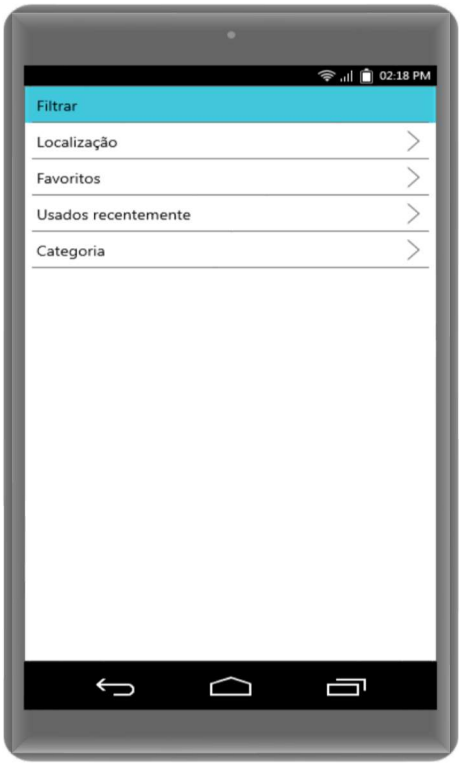


Figure 51: Filter interface

Figure 52: Logout interface

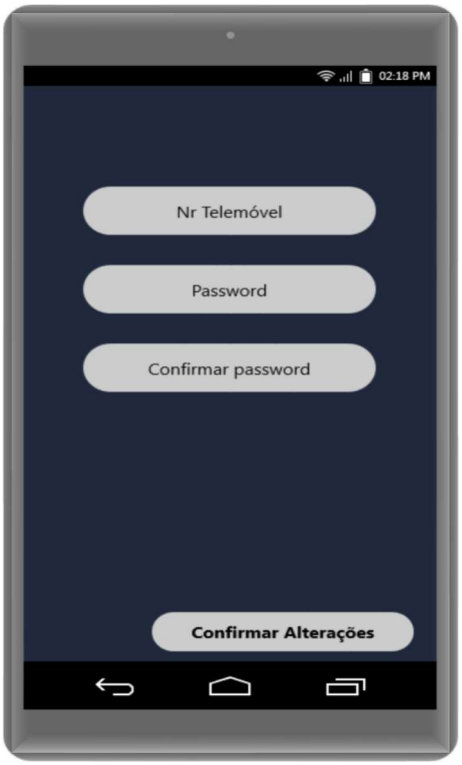


Figure 53: Edit account interface

Figure 55: Interface about us

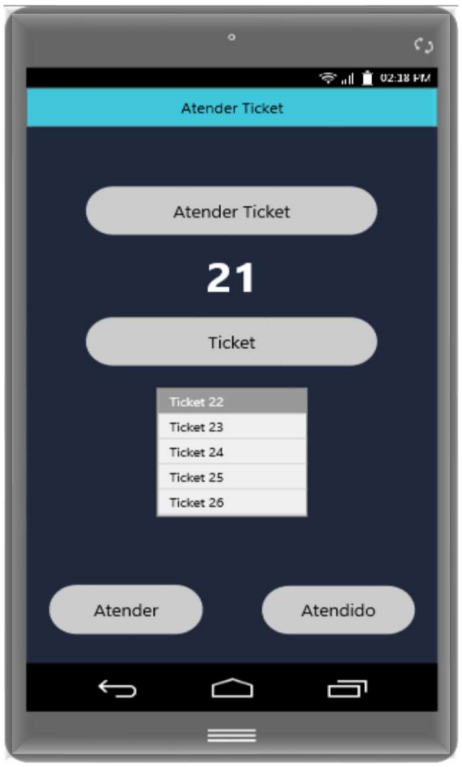


Figure 54: Employee / manager interface Answering Ticket

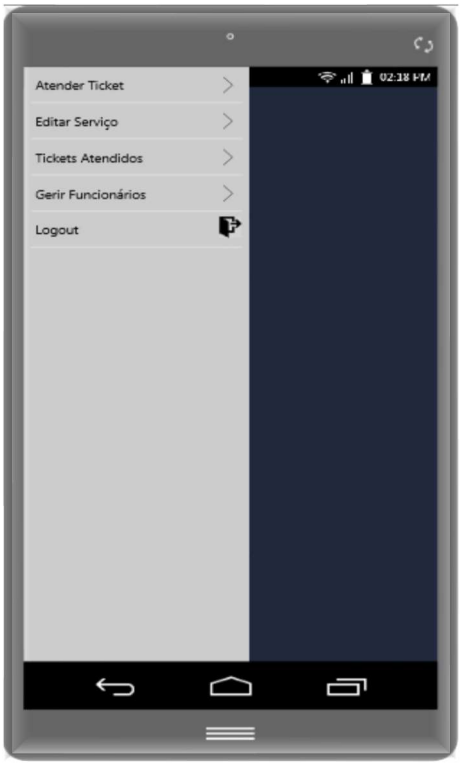


Figure 57: Manager interface edit service

Figure 58: Menu manager interface Initial

Figure 59: Manager interface Choose of employee

Figure 56: Manager interface choice of statistics

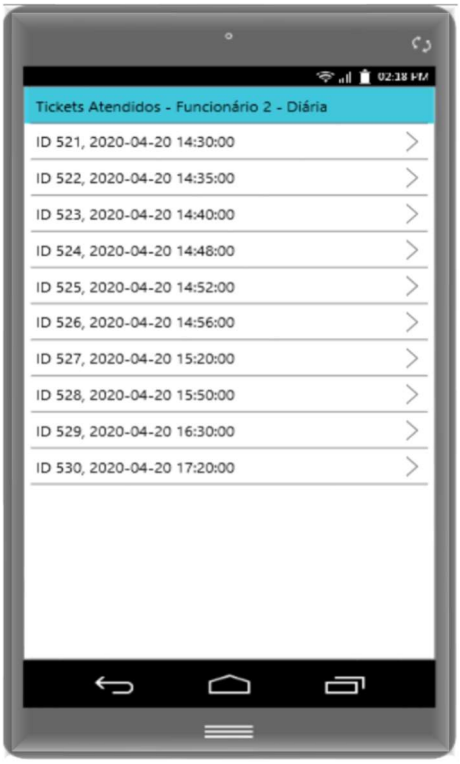


Figure 60: Manager interface, tickets attended

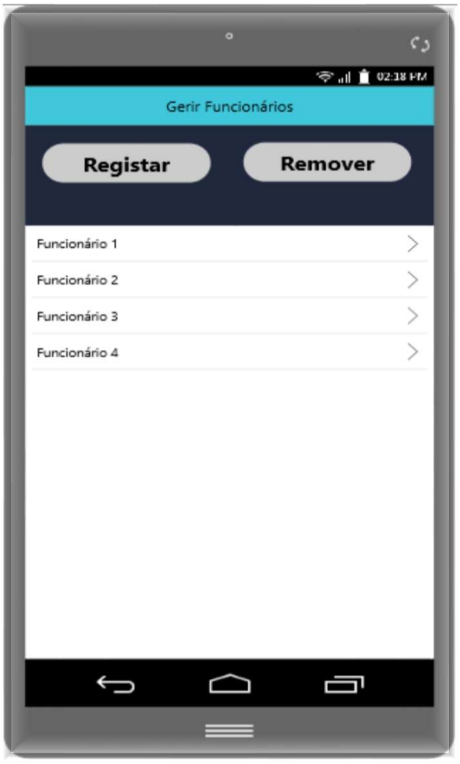


Figure 61: Manager interface, manage employees

Figure 62: Manager interface add employee

Figure 63: Manager interface remove employee

5. Implementation

In this sub-chapter, the successive stages of the application design will be presented, from the program developed to meet the requirements imposed on the application BackEnd, consisting of a program constantly running on an online server, and in a distributed database, up to the design of the Front- End, consisting of a developed program, which will run on users' mobile devices. All these parts, although developed independently, are interconnected with each other, allowing to reach the final goal, of a fully functional application.

5.1. Data base

The database is one of the most crucial parts in the design of the final application. It will allow you to keep all records persistently, obtain information quickly and available, and allow scalability and data security.

Thus, in this sub-chapter, all the phases of its design are illustrated, having chosen the language “MySQL”, present in the paradigm of the relational database, and the “Microsoft Azure” platform used to host the developed database online. .

5.1.1. Conceptual Model

The first phase of the creation of the database consisted of the realization of the conceptual model, since it shows the first physical representation of the database to be developed, helping to understand the reasoning that is intended to be implemented, from the identification of the different entities and attributes, even your relationships. The result is shown below:

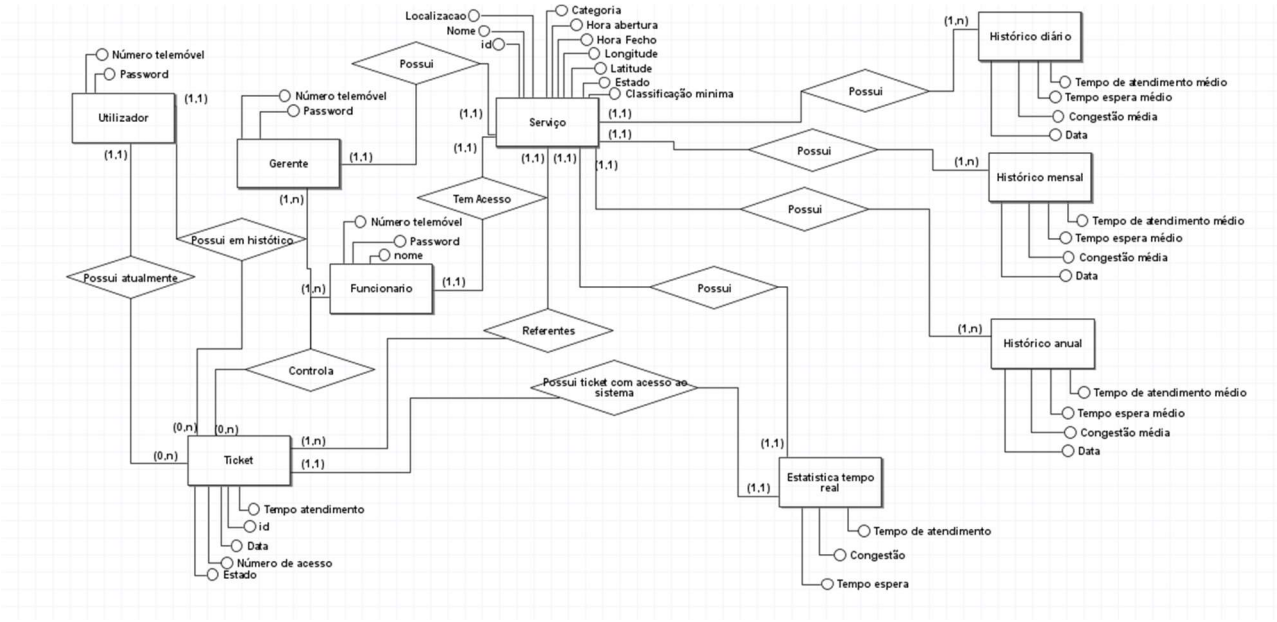


Figure 64: Conceptual Model

5.1.2. Logical Model

The logical model presented below is essentially the result of the interpretation of the conceptual model data.

The result is presented and explained below:

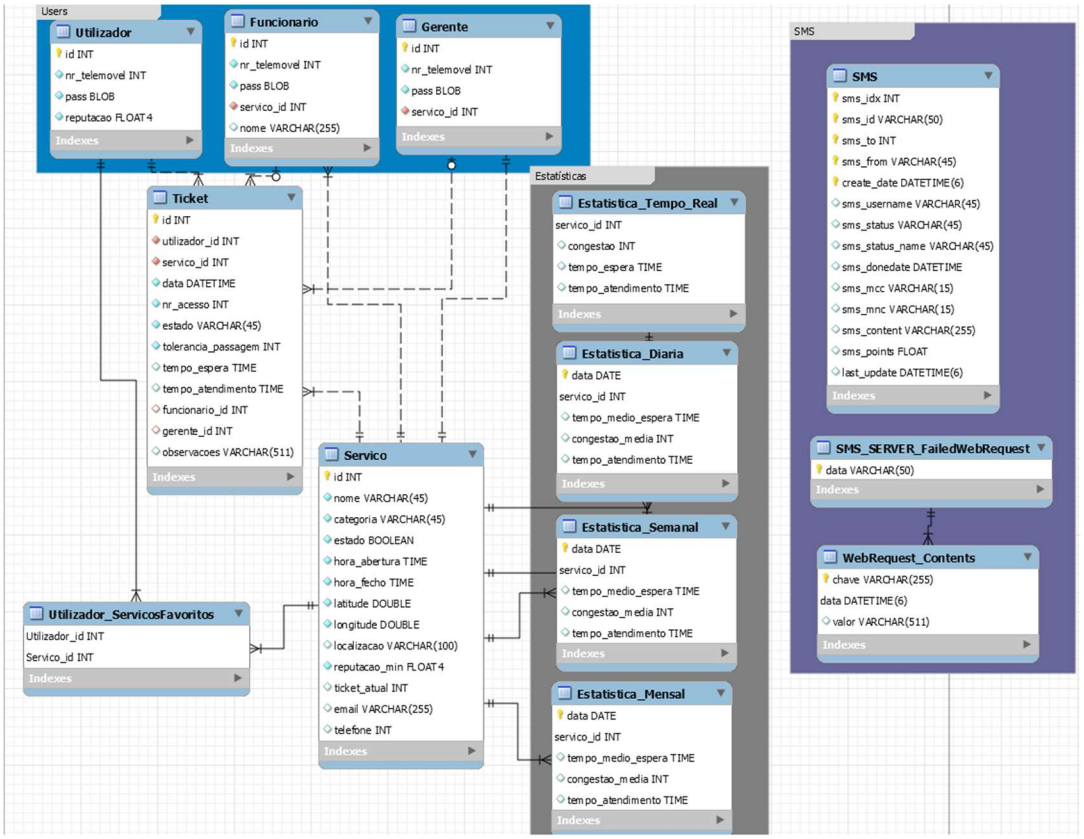


Figure 65: Logical Model

The three tables in the "Users" subdomain allow you to store the three types of user accounts existing in the application, together with all the important data related to these accounts, from the mobile phone number to the password. It was decided to have a unique account number as the primary key, and not the mobile phone number, in order to allow a mobile phone number to be used more than once, if it is transmitted between people.

Then, the tables “Service” and “Ticket”, serve in a similar way, to store the information related to the services present in the application and the tickets generated, respectively.

The “Statistics” subdomain refers to information about congestion issues and waiting times in each of the services. The table “Estatistica_Tempo_Real”, stores the numbers of the moment in each moment, so that the users are able to have in real time useful information about the affluence to a specific service. The remaining tables store information passed from different time intervals. Having only the “Estatistica_diaria” table would be sufficient to present the necessary information to users, as the other two are calculated and updated according to this. However, in order to decrease the data transfer traffic between the database and the user, these two additional tables are maintained, reducing the processing outside the database and the traffic at all times.

The tables in the “SMS” subdomain are used to store information about authentication SMS sent to users, for security and financial management of the company.

It only remains to explain the table "User_ServicesFavorites", which as the name implies, results from the relationship between the table "User" and the table "Service", keeping the favorite services of each of the users.

5.1.3. Type of Users and Connections

For security reasons in accessing the database, four types of users were created, in which each one has only access to the functionality and data strictly necessary for its normal operation. These four types are: guest, user, employee and manager. As for connections, these are made with the mobile number of each user, and each different account has a

own connection. This connection can only be used by a maximum number of devices simultaneously, to prevent denial of service attacks on the database, also taking into account the fact of connecting via Wi-Fi and mobile data at the same time. The type of guest connection is only for the user to log in to the application, and later, if the login is successful, the user uses his own connection to communicate with the database. (In fact, it is the server that uses this connection, as the user never communicates directly with the database, as will be demonstrated later).

5.1.4. Procedures

Also for several reasons, listed below, all access to the database, whether to consult, change, add or remove data, is done through calls, by the server, of “stored procedures” implemented in the database. . The reasons for this decision were as follows:

- Security issues:
    - Each procedure has access control, and only accounts of a certain type can use certain procedures, increasing security
    - Better automatic protection, that is, done by the database, in order to protect yourself from data injection attacks
- Efficiency issues:
    - Each procedure call does not need to transfer all code from the server to the database, resulting in less traffic generated
    - Having the procedures stored in the database, you are able to temporarily keep results in cache, increasing efficiency in future orders
    - Parameters can be used as output in order to return response data more efficiently
    - The database is easier to optimize its operation using its own procedures

5.1.5. Triggers and Events

The last step in the design of the database was to add "triggers" and "events" to update information in certain tables automatically and effectively. These are presented and explained below:

Name of Events:	Objective:
myeventEstatisticaRealHoraria	Responsible for updating the statistics in time real, every 3 minutes
myeventEstatisticaRealDiaria	Responsible for updating daily statistics, every 24 hours
myeventEstatisticaRealWeekly	Responsible for updating weekly statistics, every 24 hours
myeventEstatisticaRealMensal	Responsible for updating monthly statistics, every 24 hours
myeventTicketNumZero	Responsible for updating the ticket for each service to zero, at midnight, every day
myeventAddReputacaoFree	Responsible for ensuring that a user with less than 1.5 reputation stars will earn 0.5 stars if they do not waste any tickets in a week
myeventDescartaTickets	Responsible for ensuring that if a ticket is not answered within 36 hours, it is canceled

Figure 66: Database Events

Triggers name:	Objective:
checkNextTicket	Responsible for updating the current ticket on a service
after_turnoffService	Responsible for ensuring that if the manager closes the service, all unanswered tickets are discarded

Figure 67: Database triggers

5.2. Back-End

In this sub-chapter, the main decisions made in the design of the application Back-End will be explained, more specifically the software layer that interconnects the database developed with each of the mobile customers. (Denote that the database is also part of the Back-End and has already been implemented before).

5.2.1. VS REST API Distributed System

The first major decision to be made in the realization of this software layer, was which architecture / paradigm would be used in its design. For this, we identified two different alternatives that we could choose from, and will be explained below, with a critical analysis of each one, including its advantages and disadvantages, and finally, the one chosen.

The options identified were the design of a REST API or a distributed System, more specifically, a server that served the mobile devices as its clients.

A REST API essentially consists of an online interface that allows you to feed the requests of those who use it, without them depending on how it is implemented. It makes use of the HTTP protocol, of the application layer, and allows exchange of information, built in a pre-defined format, using languages such as JSON for this. The main advantage of this type of implementation is its ease and widespread use, already having a well-defined prototype and methods already made and ready to be reused.

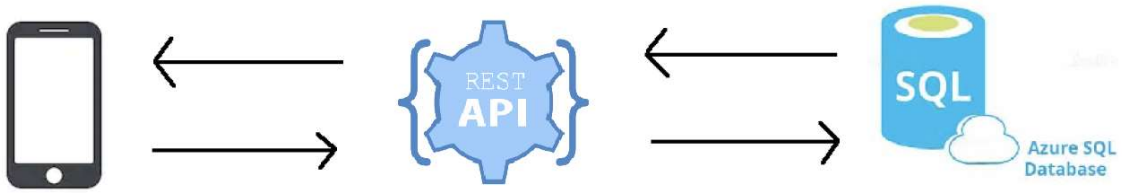


Figure 68: Rest API

The second alternative represents the implementation of a system distributed from the root, so that all customers (mobile devices) communicate with a

server, owned and programmed by the work group, and it communicates with the database. This paradigm also has its advantages. The first, and one of the main, is that the cost of communication is lower, so it is more efficient to deal with large volumes of data. The second is freedom of implementation, since something that is already well defined and implemented is not used, as in a REST API, which makes use of a pre-defined “template”.

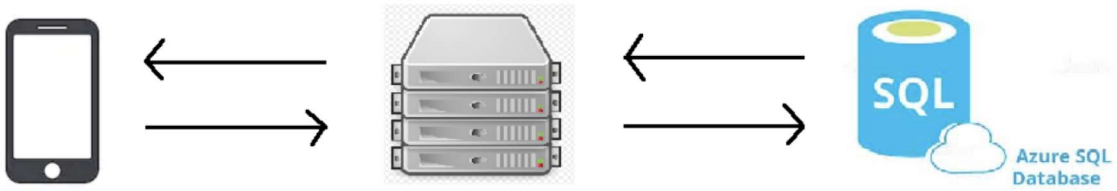


Figure 69: Distributed System

Finally, any of the alternatives would be a completely viable option for the implementation of the intended. In this way, the working group chose to use a distributed system, mainly because this paradigm is somewhat more free and powerful, and unique features can be implemented. The C # language was used to achieve this assumption, and all communications are based on the TCP protocol, to ensure reliability in communication, that is, there are no losses in data transmission.

5.2.2. Class diagrams

In this subchapter, for the sake of modeling and documentation, the class diagram resulting from the development of the server is presented.

For reasons of presentation and legibility, the “facade” class does not have the signature of the methods it contains, but in reality it has a signature for each of the methods of the classes that compose it, so that customers only communicate with it, being the “entry point” of the server.

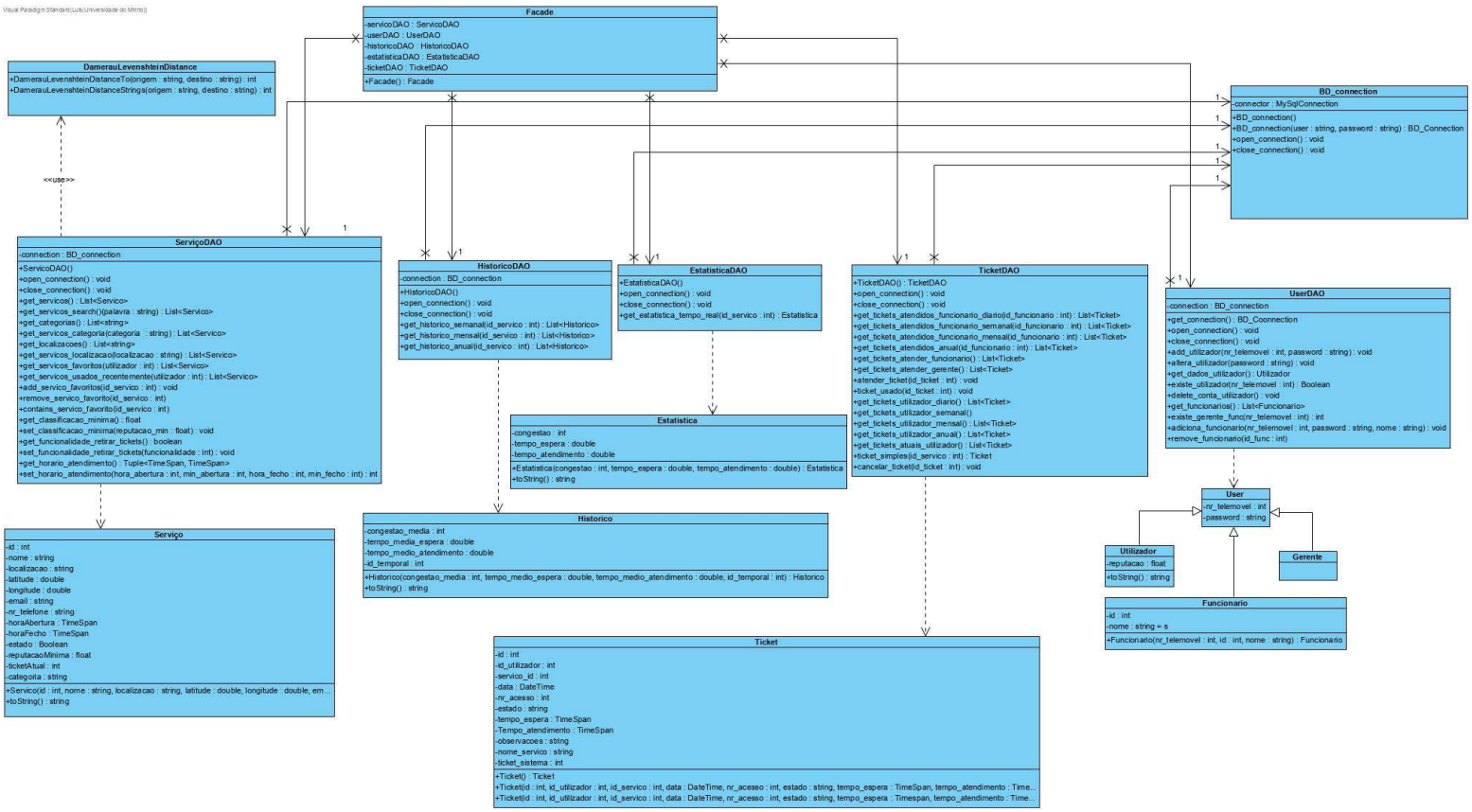


Figure 70: Class diagram

Then, and also for readability reasons, the second part of the diagram is presented, consisting of the classes responsible for handling the sending of SMS to users, and the functionality to collect tickets depending on the location, or depending on the time of arrival

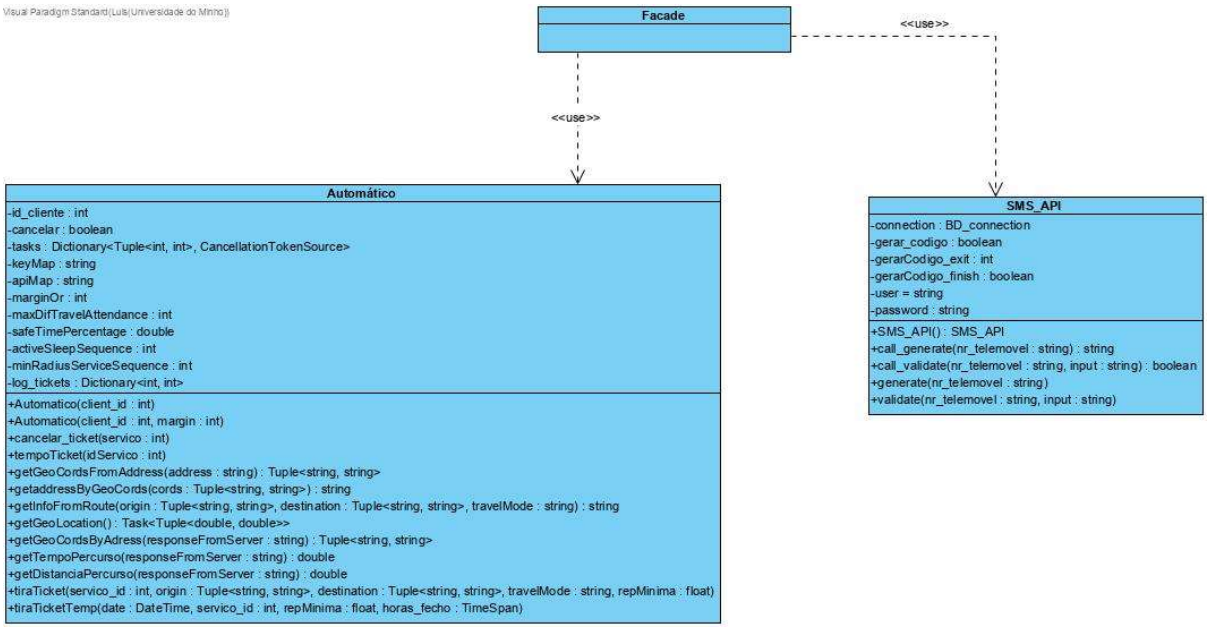


Figure 71: Class Diagram Continued

5.2.3. External API's

In order to guarantee authentication and reliability in the use of the application, that is, that fake users are not created, guaranteeing its veracity and avoiding the use of other more sensitive data, such as citizen card number or emails (easily forged), it was then decided to use the typical SMS for sending access codes (authentication via SMS). To this end, a company that provided such functionality was contacted. After a few attempts it was achieved with a European company, based in Poland, called (interestingly) SMS-API. Regarding the API, it is quite powerful, and it is also possible to use more resources available in the future. Currently, only the “SMS Authenticator” service was used, which sends a text formatted by the group, together with a verification code generated by the external API. This verification code is valid for 3 minutes. However, since we already have the code and, with each new order, the previous code becomes obsolete, coupled with the verification cost (since it needed a new call to the service, in this case the code verification functionality that implies a monetary cost), it was decided to keep it on the developed server.

Note that it works with any international number (with specific costs depending on the nationality of the number).

For automatic tickets, it was necessary to use two Google Maps APIs, the "Distance Matrix API" and the "Geocoding API", in which both APIs work through an HTTP request from the client, and one response from google API in JSON format. The first API is used to calculate the travel time from a url, placing the necessary fields, metrics, origins and destinations and the method of transport. The part of the "Geocoding API" works in the same way, that is, through a URI, in which only its fields are changed.

5.3. Front End

The Front-End of the application developed by the work team is the application layer that will be present on the users' mobile phones and will communicate with the developed Back-End, to achieve all the assumptions. This application layer was designed in Xamarim, in order to create a mobile application with Microsoft technology, and that is compatible with the two largest mobile operating systems (Android and IOS). Was

still used, as a form of assistance, several plugins, which allow from the presentation of graphs in an easy way, to obtain the user's location via GPS.

5.3.1. Results

The IQueue team developed the “Smart Ticket Managemet” for several weeks until they were able to obtain the previously designed software product. Then, we will demonstrate and explain the mobile application, presenting screenshots of the final interface, and explaining the product and its functionalities.

For any type of participant that accesses the “Smart Ticket Managemet” system, the possibility will be presented to register in the application or to be registered.

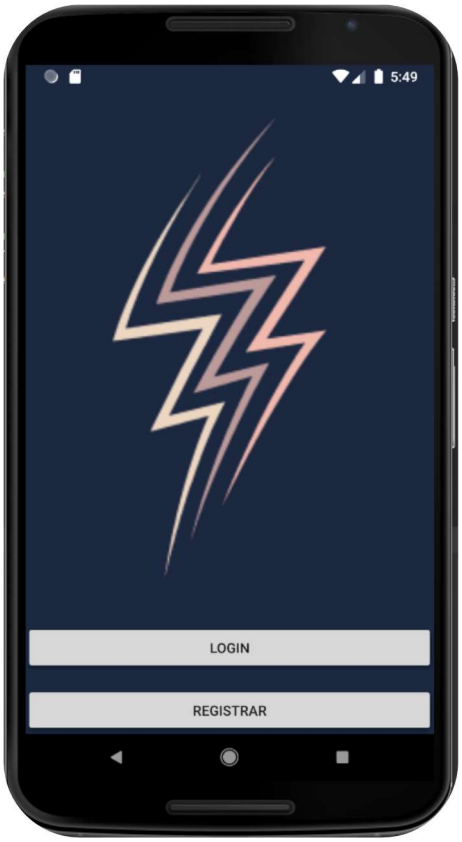


Figure 72: Results, Start Menu

Through the option to choose the registration option, available only to users, they will be redirected to the registration page, where they can submit their data and later register, if everything is within the allowed parameters.

Figure 73: Results, User registration

After filling in the data, the user will have to end the service with an account confirmation message.

Figure 74: Results, Account confirmation

If you want to login, the following page will be displayed, where you can submit the input data, and will guarantee access to the user, employee, or manager menu, depending on the type of account

Figure 75: Results, Login

If the data is incorrectly entered, that is, the account does not exist or the password is wrong, there will be an error pop up display

Figure 76: Results, Account Error

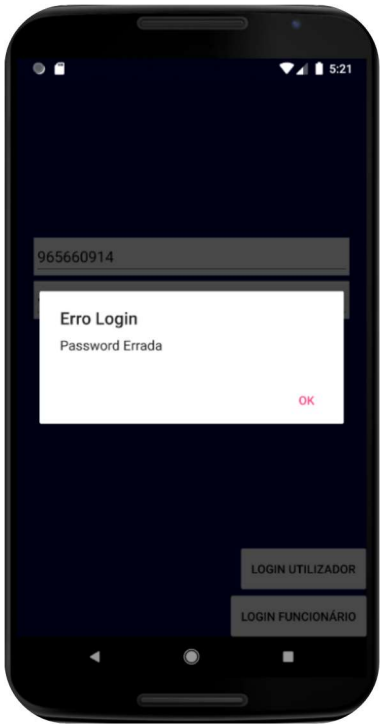


Figure 77: Results, Wrong Password

After successfully logging in to a user account, the window with the menu and the respective features to which the user has access is displayed. Choosing any of the options presented, you will have access to these features.

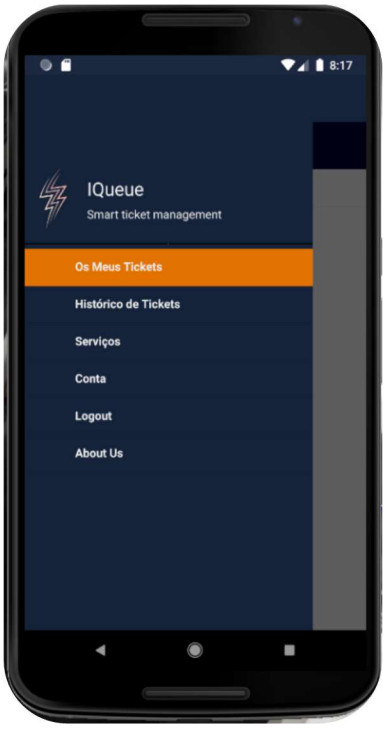


Figure 78: Results, Home menu

If the user chooses the option “My Tickets”, he will have access to the tickets that are active, being able to choose any of them to see their status.

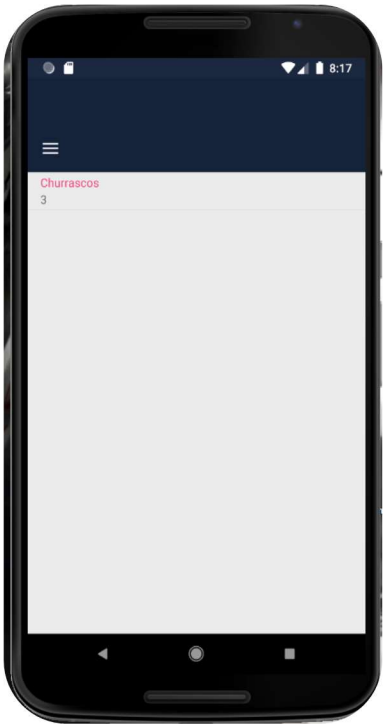


Figure 79: Results, Active tickets

As we can see, when choosing a ticket, the user observes the data of the ticket that was withdrawn, and also has the possibility to cancel the ticket.

Figure 80: Results, Active Ticket

In active tickets, if the user chooses to cancel a ticket, he / she will have a security "pop up", to obtain certainty of cancellation.

Figure 81: Results, Cancel Ticket confirmation

If the user, in the initial menu, chooses the ticket history option, he will be able to observe all tickets taken on the day.

Figure 82: Results, Tickets used history

If the user wants to change the filter for the ticket history, he will be presented with the following options, where he can choose to obtain new results.

Figure 83: Results, Ticket history filter

When the user in the initial menu, chooses the option to choose the option “services”, he will obtain the existing services, being able to apply a filter or to search from an entered name.

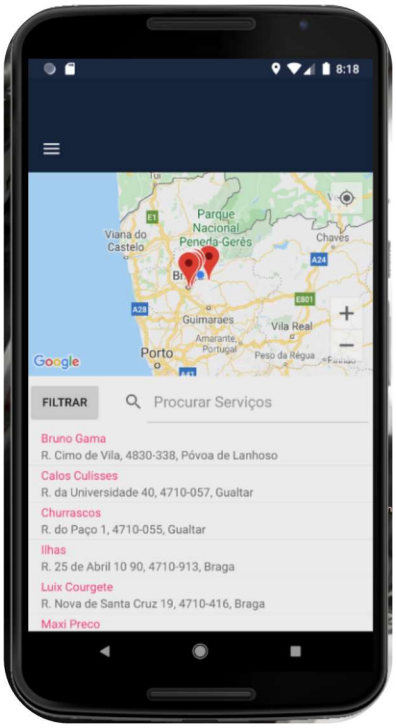


Figure 84: Results, Services

From the moment the user chooses a service, selecting it, an options menu will be available.

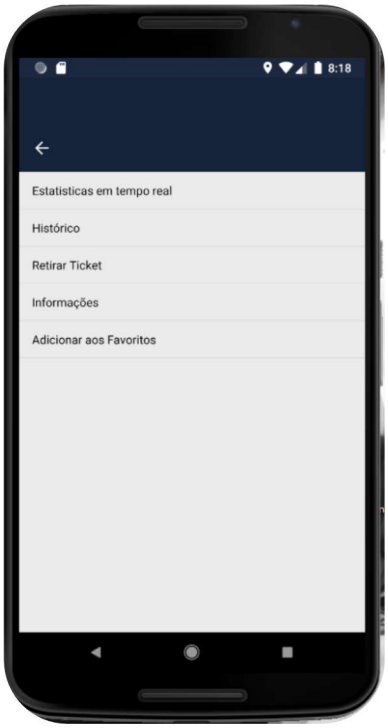


Figure 85: Results, Service menu

In case the user chooses “real-time statistics”, the statistics of the service in question will be made available.

Figure 86: Results, Real-time statistics

In this way, if the user chooses the history, the statistics of the service in question will be made available in graph format.

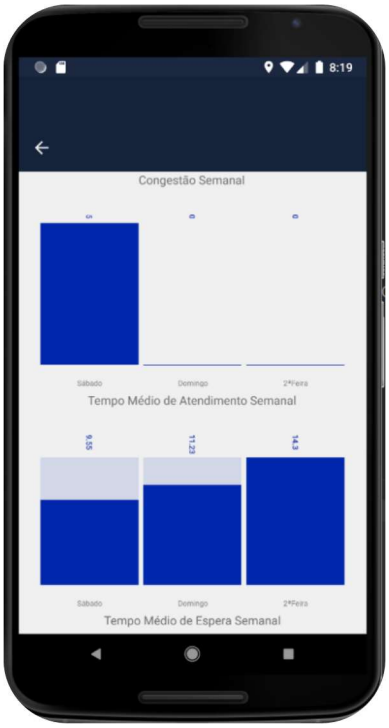


Figure 87: Results, Service history

By choosing the option “withdraw ticket”, the user has three options on how he wants to carry out the process.

Figure 88: Results, Remove Ticket

If you click on “information”, from the service menu, the user will have access to the service data.

Figure 89: Results, Service Data

When choosing "Account", from the menu, the user will have access to his account data and possible changes to it.

Figure 90: Results, User account data

If the user wants to change his password, he will have to click on edit account.

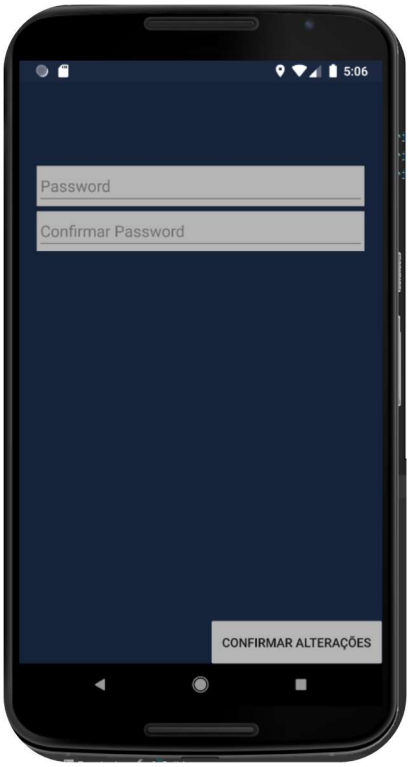


Figure 91: Results, Change password

When choosing the “About Us” option, from the menu, the user will have all the information about the application, as well as an option to contact the company, in case of doubt

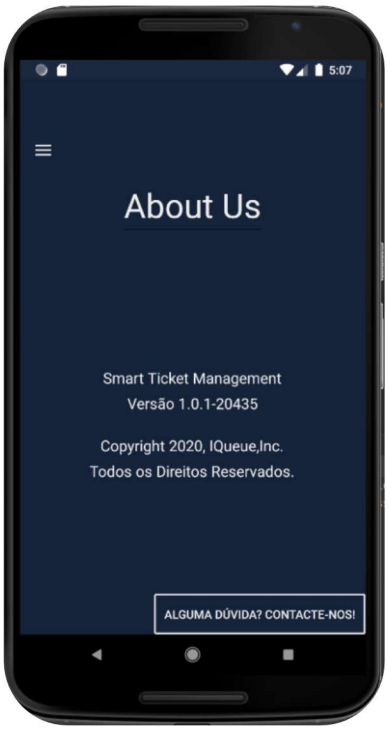


Figure 92: Results, About Us

If the login is successful in an employee account, the “Answer Tickets” window is displayed with the functionality of answering tickets and logout only.

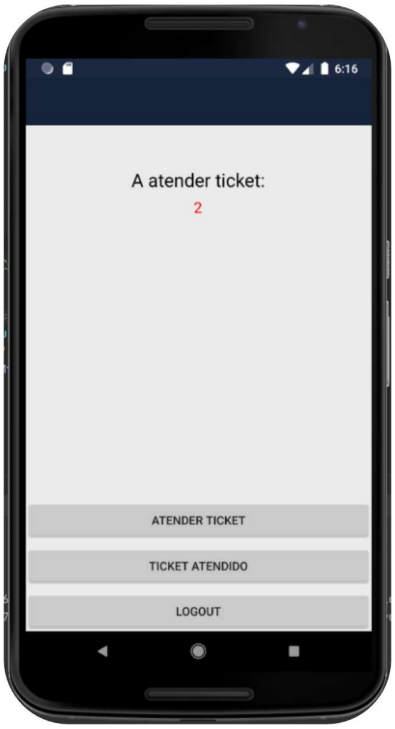


Figure 93: Results, Attending ITcket employee

If the login is successful in a manager account, the Menu window is displayed with the features to which the manager has access. Choosing any of the options presented, you will have access to these features.

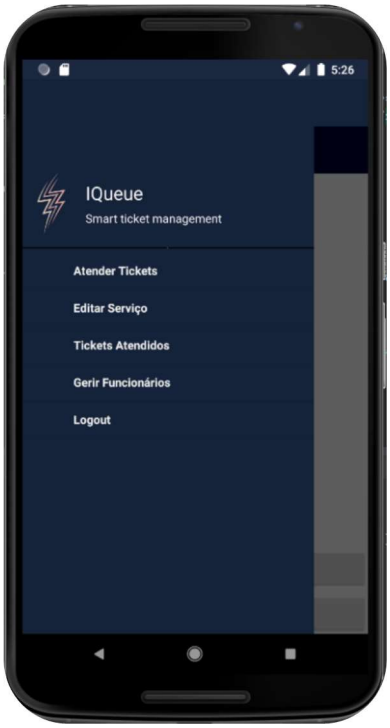


Figure 94: Results, Manager menu

In the option “Answer Ticket”, from the menu, the manager can answer tickets, if he so wishes.

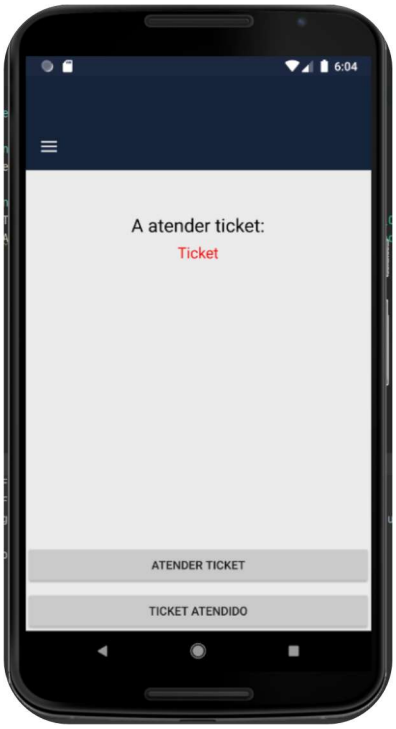


Figure 95: Results, Answering Ticket manager

In the "Edit Service" option, from the menu, the manager can edit the data of the menu, changing if the minimum reputation necessary to take a ticket from the service is desired, edit the service hours and finally activate or deactivate the service.

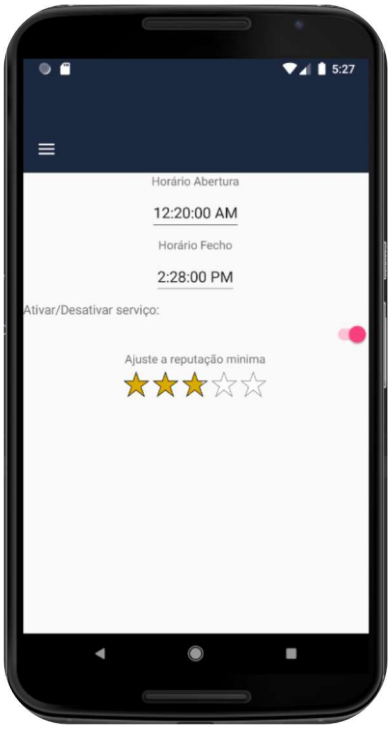


Figure 96: Results, Edit Service data

If you choose “Answered Tickets”, from the menu, the manager will be directed to a page, where you can choose the employee for whom you want to obtain information about the tickets served.

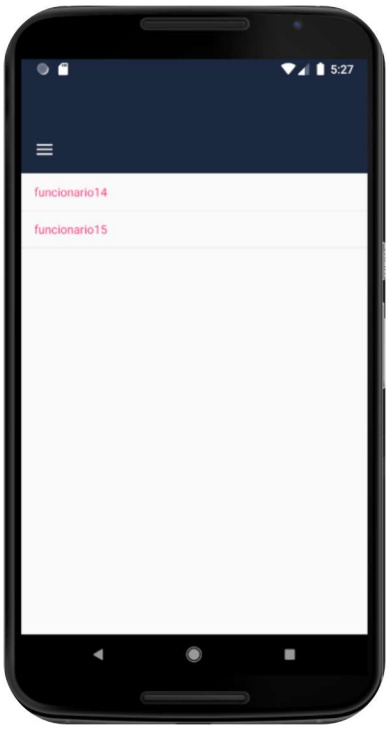


Figure 97: Results, Tickets Used, Employee Choice

From the moment the manager chooses the employee, the list of tickets served will be obtained.

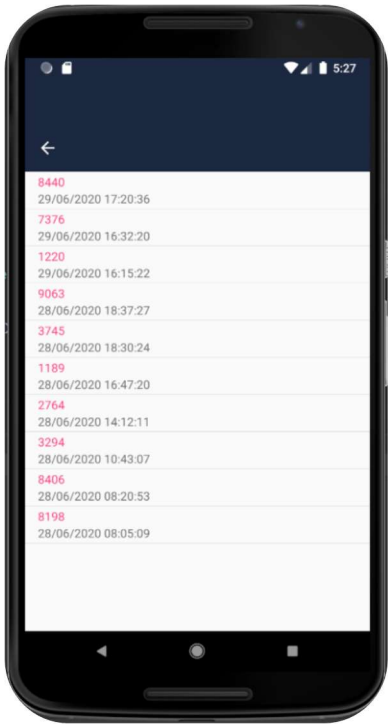


Figure 98: Results, Tickets served

If the manager clicks on one of the tickets, he will have information about it.

Figure 99: Results, ticket information

In the option "Manage employees", from the menu, the manager can add, or remove employees.

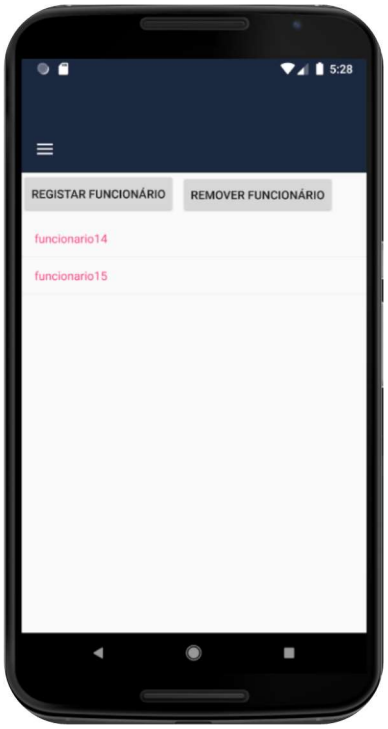


Figure 100: Results, Managing employees

If the manager selects one of the employees, and then click on the remove button, a security message will appear for its removal.

Figure 101: Results, Remove employee

If the manager wants to add an employee, he will be directed to an employee registration page.

Figure 102: Results, Add employee

6. Economic viability

6.1. Cost analysis

Any project in the real world can be seen as a business, and therefore it has to generate economic profitability for those who create and manage it. Thus, it is vital, in addition to creating a usable and friendly application, to have a good financial plan for it to be sustainable. For this reason, then, a financial plan is presented that reflects a balance between expenses and sources of profit, and the plan outlined for the launch of the application.

Point number one is how the application will be launched. It is expected that this will be launched simultaneously in the Android store, “Google Play Store”, and in the IOS store, “App Store”, to reach the largest possible number of users. This will cost a single € 25 to be launched in the Android store, and an annual cost of around € 100 to keep it in the IOS store. It is also expected to generate a marketing and advertising plan for the launch of the application with an initial budget of around € 1000. After launch, the application will always have sources of fixed costs, either expense or profit. These are shown below.

As for expenses, these result mainly from maintaining the database on the “Microsoft Azure” platform, maintaining the server, also using a virtual machine rented on the same platform, and also from two external APIs, used for sending confirmation messages. , and for distance queries between two points, to allow you to take tickets intelligently. To maintain the database, it is necessary to pay a monthly fee depending on the disk space occupied. This cost is shown below in the form of a graph:

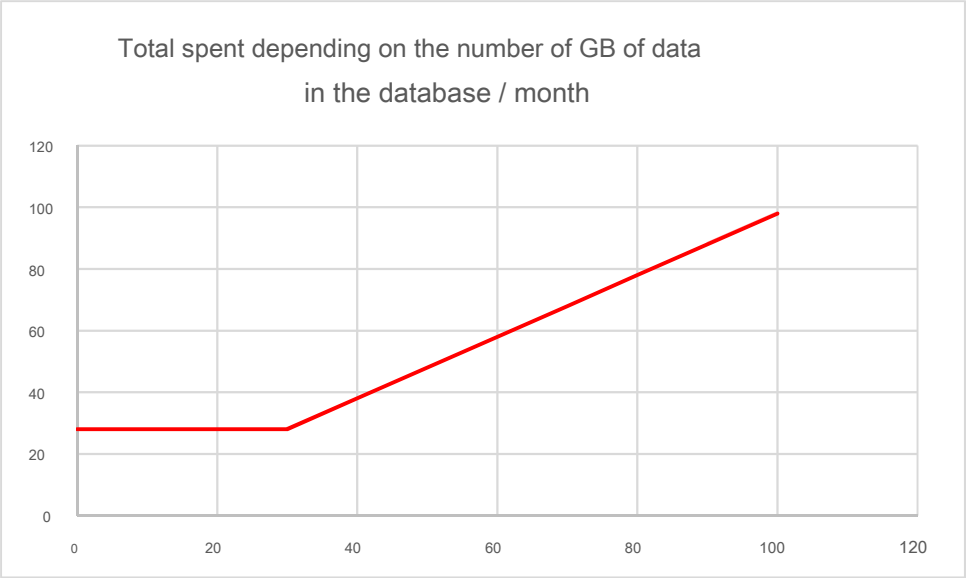


Figure 103: Database expenses graph

As for the rental of the virtual machine, it was decided to rent one with the “Windows 10” operating system and with about 8 GB of RAM, resulting in a fixed monthly cost of around 152 €, representing the largest immediate expense.

Still referring to the costs, each SMS sent, for each new user account created has a cost of about € 0.02, with a graph below showing the costs taking into account the number of users created:

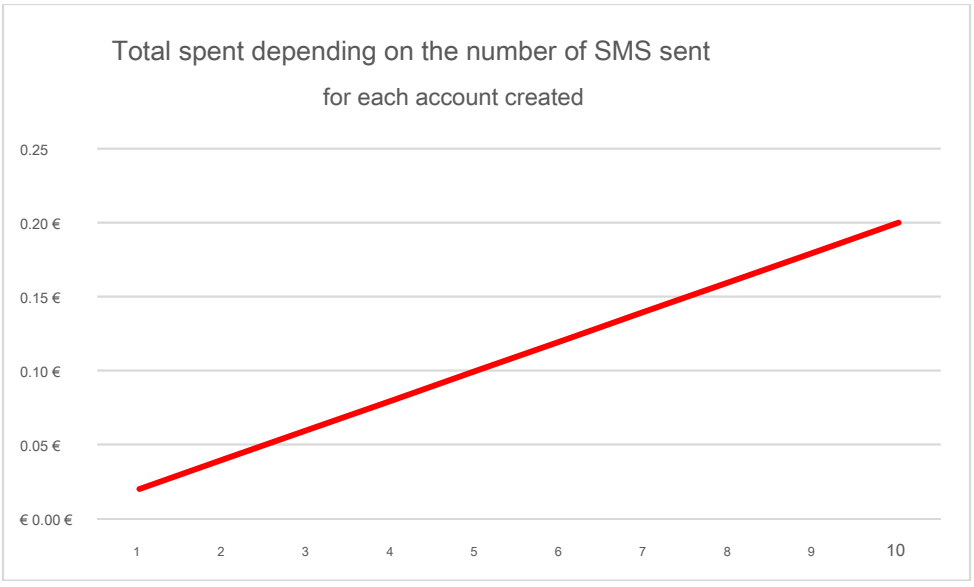


Figure 104: SMS Authenticator expense graph

Also due to the use of another external API, in this case belonging to GOOLE. This API is used for the functionality of removing tickets intelligently, and represent

a cost of € 10 for every 1000 tickets taken. This information is presented in the form of a graph:

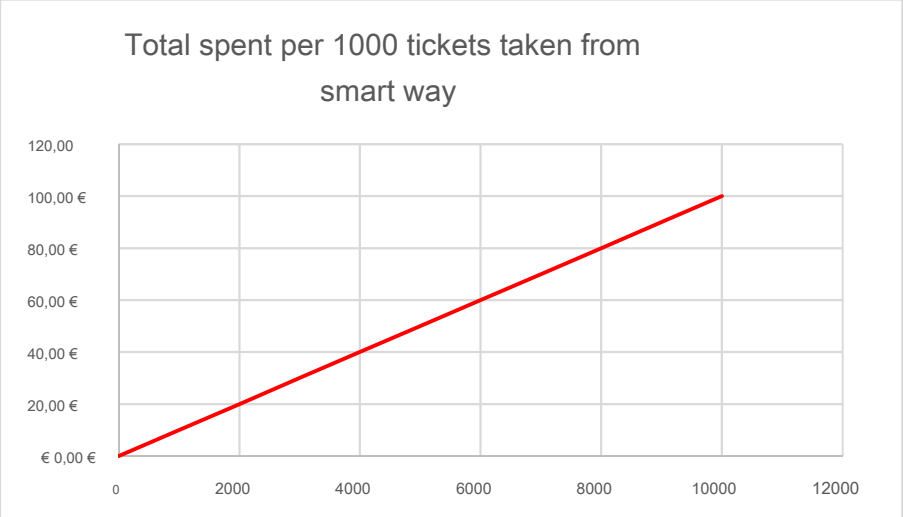


Figure 105: Smart ticket spend graph

Lastly, there is the cost of labor for the workgroup, and it was decided, as it is a company of independent developers, that we would be paid by royalties, that is, we received according to the company's profit, being that each of us holds 15% in royalties.

Now beginning to explore the sources of profit, each has a different origin. The first source, consists of the monthly fee that a service has to pay the company to stay in the application, which is around 50 €. A graph is shown below, showing the relationship between the number of companies and the profit obtained per month:

Figure 106: Monthly company profits

The second source of profit is the payment that each service makes for each ticket that is withdrawn and used on itself. This value is around € 0.03 for each ticket. A projection is shown below:

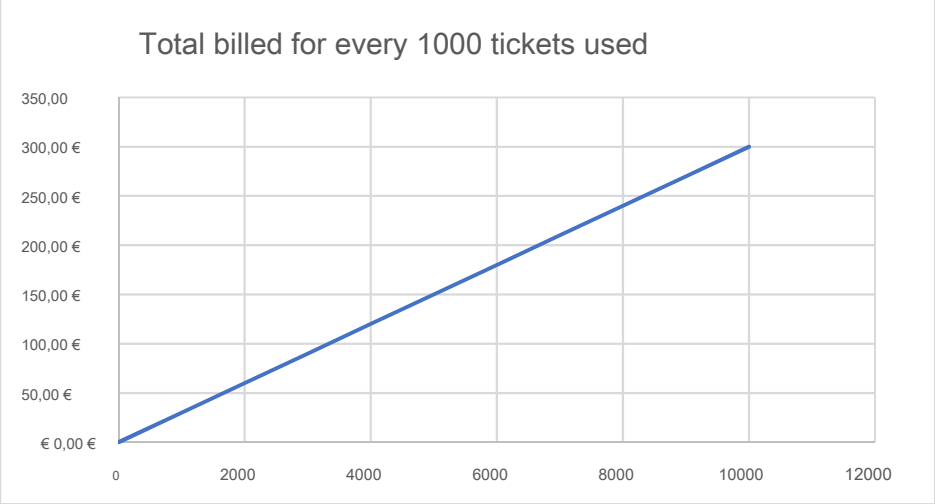


Figure 107: Profits for every 1000 tickets used

The company's main source of revenue is expected to be advertising included in the application itself, with each company wishing to include advertising in the application having to pay around € 500 per month.

Figure 108: Profits from advertising

The last source of profit will be the monthly fee paid by premium users (feature not yet implemented, but it belongs to future work). They will have to pay a monthly loyalty of about 3 €.

Below are general figures as to what is expected for the next 5 years:

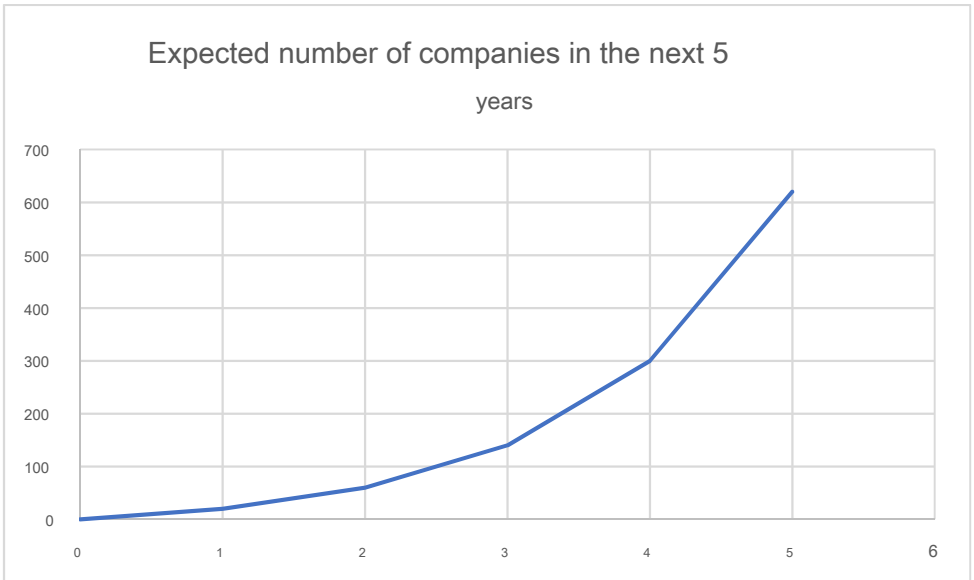


Figure 109: Companies expected in 5 years

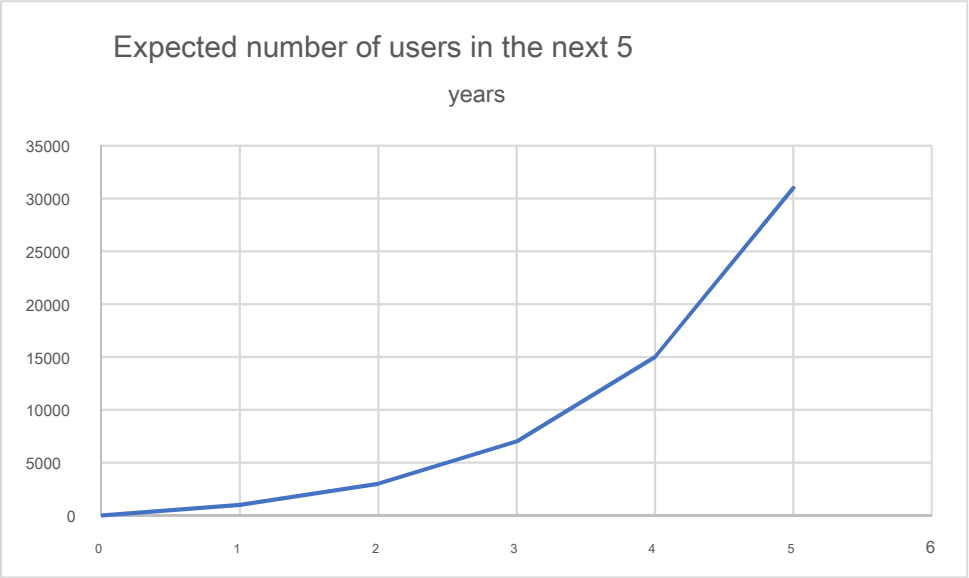


Figure 110: Users expected in 5 years

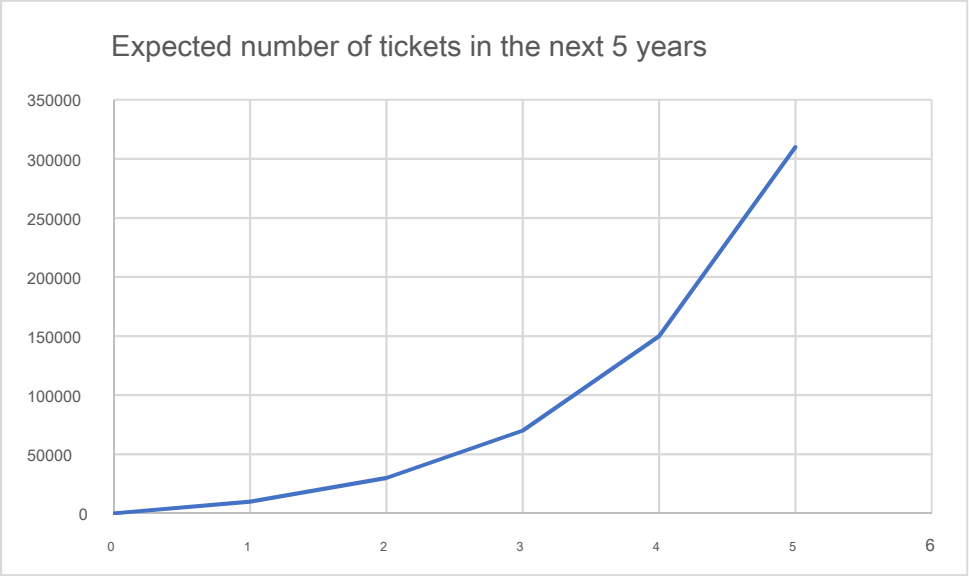


Figure 111: Tickets expected in 5 years

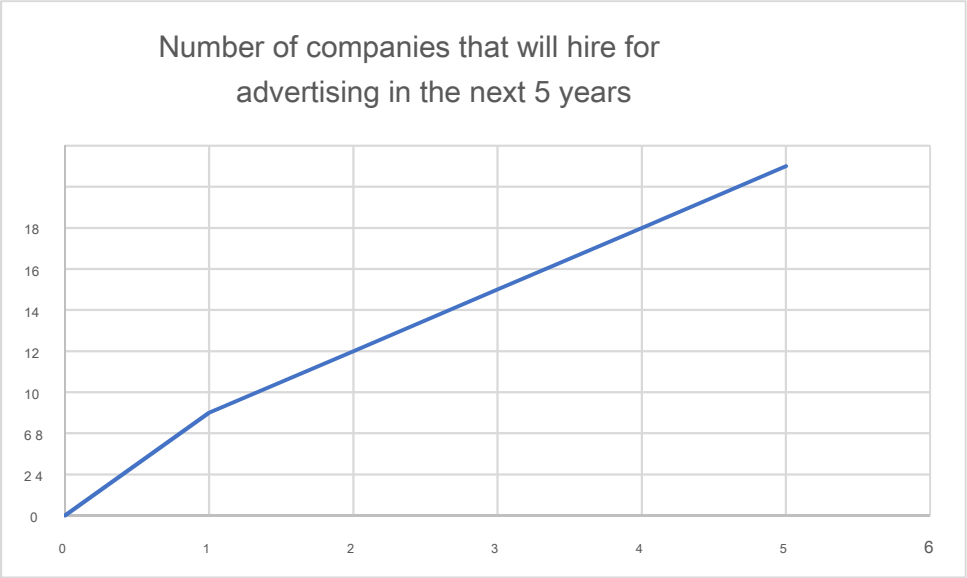


Figure 112: Companies opting for advertising

In conclusion to the financial study, and taking into account the data presented above, the group expects to reach about (provided that 20% tickets withdrawn are smart, and about 100 GB of data are occupied after 5 years):

Total profit over 5 years = Total revenues in 5 years - Total expenses in 5 years = (102000 (advertising revenue) + 9000 (revenue tickets) + 360000 (revenue from companies))

- (6000 (database expense) + 600 (amount spent on SMS) + 600 (tickets intelligently) + 50000 (business and application investment)) = € 413 800.

7. Future Tasks

Like all software systems, this work is no exception, there are always new features that can be added to make the application even more complete.

If the application developed is successful in this first phase of development, the updates that would be made to it are presented below, in order of priority:

- Use of data analysis and Machine Learning to calculate real-time statistics for a service, with greater precision and susceptibility to variances.
- Implementation of advertising support in the application, so that companies that want to hire us to advertise their products can be easily supported.
- Implementation of a new account: premium user, which will make it possible to distinguish between two types of users, and not just the common one. The perks of the premium will be to have no advertising in your application and to be able to obtain an unlimited number of tickets intelligently in the application, while the average user will have a monthly limit.

- Introduction of notifications in the application, which allow better control of tickets.

- Encryption of data between server and clients, using a public key and private key mechanism.
- QRCode implementation for the use of tickets.
- Support for multiple languages when the application reaches an international level.
- Increase the exponentiality of automatic tickets making it possible for multiple services simultaneously.

Figure 113: QRCode

8. Conclusion

The idea of the application conceived at the beginning of this semester could not have been better. It was really a big challenge, but it gave great pleasure to every working group to develop.

In view of the aforementioned, it is also essential to mention that the phenomenal help provided by the Polish company SMS API, to which the group spares no words in thanks, since this was the only company that provided its services free of charge.

The group thinks that it has achieved all the objectives initially proposed for the proper functioning of the application that resulted from the mutual assistance between all the elements, as well as their contributions so that it is worthy of going further. The greatest difficulty felt was based on the use of new technologies with which he was never in contact, a barrier that was quickly overcome when carrying out intensive research on the project at hand. In this way, it was really enriching to be in contact with technologies such as the “Microsoft Azure” platform, API's that we had never worked with before, and still gain confidence with a new language, in this case, C #.

As aspects to improve, consistency in the elaboration of the work is considered of special relevance, directed to the equitable division of the different tasks present in each phase, with the final objective of establishing a balance in terms of quantity to be carried out in each part and moment.

In terms of speed and efficiency, it is understood that the project is in a comfortable position, which can be said to be in a good level with regard to them. In short, the importance of what has been accomplished in this project is notoriously visible, since it turns out to be a small sample of a great future that lies ahead in our area.