





# **UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY**

**Escuela de Ciencias Matemáticas y Computacionales**

## **TÍTULO: Reserve System for Institutional Autobuses**

Trabajo de integración curricular presentado como requisito para  
la obtención del título de Ingeniero en Tecnologías de la  
Información

### **Autor:**

Henry Fabricio Caraguay Ordoñez

### **Tutor:**

Ph. D. Francesc Antón Castro

Urcuquí, Julio 2020

**SECRETARÍA GENERAL**  
**(Vicerrectorado Académico/Cancillería)**  
**ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**  
**CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN**  
**ACTA DE DEFENSA No. UITEY-ITE-2020-00019-AD**

A los 24 días del mes de marzo de 2020, a las 14:30 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

<b>Presidente Tribunal de Defensa</b>	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.
<b>Miembro No Tutor</b>	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.
<b>Tutor</b>	Dr. ANTÓN CASTRO , FRANCESC , Ph.D.

El(la) señor(ita) estudiante **CARAGUAY ORDOÑEZ, HENRY FABRICIO**, con cédula de identidad No. **1150218418**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **MOBILE RESERVE SYSTEM FOR INSTITUTIONAL BUSES.**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

<b>Tutor</b>	Dr. ANTÓN CASTRO , FRANCESC , Ph.D.
--------------	-------------------------------------

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Tutor	Dr. ANTÓN CASTRO , FRANCESC , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.	8,5
Miembro Tribunal De Defensa	Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.	7,0

Lo que da un promedio de: **8.5 (Ocho punto Cinco)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

*Certifico que en cumplimiento del Decreto Ejecutivo 1017 de 16 de marzo de 2020, la defensa de trabajo de titulación (o examen de grado modalidad teórico práctica) se realizó vía virtual, por lo que las firmas de los miembros del Tribunal de Defensa de Grado, constan en forma digital.*

CARAGUAY ORDOÑEZ, HENRY FABRICIO  
**Estudiante**

Dr. CUENCA LUCERO, FREDY ENRIQUE , Ph.D.  
**Presidente Tribunal de Defensa**

Dr. ANTÓN CASTRO , FRANCESC , Ph.D.  
**Tutor**

Dr. CUENCA PAUTA, ERICK EDUARDO , Ph.D.  
**Miembro No Tutor**

TORRES MONTALVÁN, TATIANA BEATRIZ  
**Secretario Ad-hoc**

# Autoría

Yo, **Henry Fabricio Caraguay Ordoñez**, con cédula de identidad 1150218418, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor(a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urququí, Diciembre 2019.



---

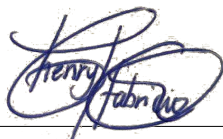
Henry Caraguay O.  
CI: 1150218418

# Autorización de publicación

Yo, **Henry Fabricio Caraguay Ordoñez**, con cédula de identidad 1150218418, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Diciembre 2019.



---

Henry Caraguay O  
CI: 1150218418

# Dedication

*“To my parents and siblings that have given me all the support through these five years in college. I will never be able to compensate for all the help given. ”*

# Acknowledgements

*“ First of all I thank my family, that have been always my main source of support. To my adviser Francesc Castro, for its guidance during the development of this work. To my co-adviser Francisco Hidrobo, that always was there available for meeting with me and solve any of my doubts, without his help this work would not have been possible. Finally, to all my classmates and friends that gave me their help, both in and out of the classroom. ”*



# Resumen

El transporte es uno de los servicios más importantes a los que la población necesita tener acceso permanentemente. Los integrantes de nuestra universidad, al estar esta situada en un cantón pequeño y alejado de la provincia de Imbabura, hacen uso del servicio de transporte de forma diaria y constante. Pero el servicio de transporte interno ofrecido por la universidad en la actualidad presenta inconvenientes a la hora de usarlo. Desde largas colas al momento de abordar el bus hasta imprecisiones en el horario del servicio, que provocan malestar en los usuarios.

Para solucionar este inconveniente el presente proyecto muestra el desarrollo e implementación de una aplicación móvil bajo el sistema operativo Android, que intenta solventar algunos de estos inconvenientes. Esta aplicación móvil lleva el nombre de YTDrive y permitirá a los usuarios acceder al servicio de transporte desde su smartphone. La aplicación tiene dos versiones, una destinada para los pasajeros y una destinada para los conductores, con funciones específicas para cada tipo de usuario.

**Palabras clave:** transporte, aplicación, servicio, solución.

# Abstract

Transport is one of the most important services to which the population needs to have permanent access. The members of our university, being located in a small canton and far from the main roads of Imbabura, make use of the transport service on a daily and constant basis. However, the internal transport service offered by the university currently presents problems at the time of using it. These problems vary from long queues at the time of boarding the bus to inaccuracies in the service schedule, which cause discomfort among the users.

To solve this problem, this project shows the development and implementation of a mobile application under the Android operating system, which attempts to solve some of these issues. This mobile application is called YTDive and will allow users to access the transport service from their smartphone. The application has two versions, one intended for passengers and one intended for drivers, with specific functions for each type of user.

**Keywords:** transport, application, service, solution.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Problem Statement . . . . .	8
1.2	Motivation . . . . .	8
1.3	Objectives . . . . .	8
1.4	Overview . . . . .	9
<b>2</b>	<b>Theoretical Background</b>	<b>10</b>
2.1	Computerized reservation systems . . . . .	10
2.1.1	Airplanes . . . . .	11
2.1.2	Buses . . . . .	11
2.2	Tools for mobile application development . . . . .	12
2.2.1	Back-end and Front-end . . . . .	12
2.2.1.1	Back-end . . . . .	12
2.2.1.2	Front-end . . . . .	13
2.2.2	Development Framework . . . . .	13
2.2.2.1	Ionic Development Framework . . . . .	13
2.2.3	Programming Languages . . . . .	14
2.2.3.1	Back-end . . . . .	14
2.2.3.2	Front-end . . . . .	15
2.2.4	Database Management Systems . . . . .	16
2.2.4.1	PostgreSQL Database . . . . .	16
2.3	Software Engineering Concepts . . . . .	16
2.3.1	Methodology for Mobile Applications . . . . .	16
2.3.2	Architecture of Mobile Applications . . . . .	18
2.3.2.1	Self-contained applications . . . . .	18
2.3.2.2	Applications with internet connection . . . . .	18
<b>3</b>	<b>Technological Proposal</b>	<b>20</b>
3.1	Functional Requirements . . . . .	20
3.2	Non-functional Requirements . . . . .	21
3.3	Application Tutorial . . . . .	22
3.3.1	General Functions . . . . .	22
3.3.1.1	Log in Page . . . . .	22
3.3.1.2	Register Page . . . . .	22

3.3.1.3	Regulations Page . . . . .	23
3.3.1.4	Settings Page . . . . .	24
3.3.1.5	About Page . . . . .	25
3.3.2	Passengers' Functions . . . . .	25
3.3.2.1	User Menu . . . . .	25
3.3.2.2	Reserve Page . . . . .	26
3.3.2.3	Bus Location Page . . . . .	28
3.3.2.4	History Page . . . . .	30
3.3.2.5	Schedule Page . . . . .	31
3.3.3	Drivers' Functions . . . . .	31
3.3.3.1	Driver Menu . . . . .	31
3.3.3.2	Add Shift Page . . . . .	32
3.3.3.3	Bus Tracking Page . . . . .	33
3.3.3.4	Add Bus Page . . . . .	34
3.3.3.5	Add Stop Page . . . . .	34
3.4	Application Architecture . . . . .	35
3.5	Use Case Diagrams . . . . .	36
3.5.1	Use Case Diagram for Users . . . . .	36
3.5.2	Use Case Diagram for Drivers . . . . .	38
3.6	Database Entity Relationship Diagram . . . . .	40
3.7	Activity Diagrams . . . . .	43
3.7.1	General functions . . . . .	43
3.7.2	User Functions . . . . .	44
3.7.3	Driver Functions . . . . .	45
<b>4</b>	<b>Deployment and testing</b> . . . . .	<b>46</b>
4.1	Deployment . . . . .	46
4.1.1	Front-end configuration . . . . .	46
4.1.1.1	Building both presentations of the app . . . . .	46
4.1.1.2	Connecting API REST and network settings . . . . .	47
4.1.2	Back-end configuration . . . . .	48
4.1.2.1	Configuring Node.js server in Heroku . . . . .	48
4.1.2.2	Migrating our database to Heroku . . . . .	48
4.2	Testing . . . . .	49
4.2.1	Login and register . . . . .	50
4.2.2	Reserve . . . . .	51
4.2.3	History and Bus Schedule . . . . .	52
4.2.4	Bus Location and Tracking . . . . .	53
4.2.5	Settings . . . . .	54
4.2.6	Add shift, bus and stop . . . . .	55
4.2.7	Regulations, Help and About . . . . .	55

**5 Conclusions and Future Work** **57**

5.1 Conclusions . . . . . 57

5.2 Future Work . . . . . 57

**References** **59**

# List of Figures

- 2.1 Components of an application with internet connection . . . . . 19
- 3.1 Log in Page . . . . . 22
- 3.2 Register Page . . . . . 23
- 3.3 Regulations Page . . . . . 24
- 3.4 Settings Page . . . . . 24
- 3.5 About Page . . . . . 25
- 3.6 User Menu . . . . . 26
- 3.7 Error in Reserve Page . . . . . 27
- 3.8 Ready to Reserve . . . . . 27
- 3.9 Delete reservation option . . . . . 28
- 3.10 Bus Location Error . . . . . 29
- 3.11 Bus Location Map . . . . . 29
- 3.12 History Page . . . . . 30
- 3.13 Schedule Page . . . . . 31
- 3.14 Driver Menu . . . . . 32
- 3.15 Add Shift Page . . . . . 32
- 3.16 Driver not sharing his or her location . . . . . 33
- 3.17 Driver sharing his or her location . . . . . 33
- 3.18 Add Bus Page . . . . . 34
- 3.19 Add Stop Page . . . . . 35
- 3.20 Application Architecture Diagram . . . . . 36
- 3.21 Use case for Users . . . . . 36
- 3.22 Use case for Drivers . . . . . 38
- 3.23 Database Entity Relationship Diagram . . . . . 40
- 3.24 Activity diagrams for login (A), register (B) and settings (C) functions . . . 43
- 3.25 Activity diagrams for reserve (A), bus location(B), history (C) and bus  
schedule (D) functions. . . . . 44
- 3.26 Activity diagrams for add shift (A), bus location(B), add bus (C) and add  
stop (D) functions. . . . . 45
- 4.1 Function which defines the page to where the user will be redirected in the  
login . . . . . 47
- 4.2 Adding allowed subdomains . . . . . 47
- 4.3 File with specific version of our Node.js dependencies . . . . . 48

- 4.4 Accessing to Heroku using CLI . . . . . 49
- 4.5 Database credentials . . . . . 49
- 4.6 Register - Login Process . . . . . 50
- 4.7 Database with new user registered . . . . . 50
- 4.8 Status of mobile application/database before reserve . . . . . 51
- 4.9 Status of mobile application/database after reserve . . . . . 52
- 4.10 History and Bus Schedule . . . . . 52
- 4.11 History and Bus Schedule not available . . . . . 53
- 4.12 Bus Tracking not available . . . . . 53
- 4.13 Bus Tracking in execution . . . . . 54
- 4.14 Entering our new account information. . . . . 54
- 4.15 Database before and after update account information. . . . . 55
- 4.16 Filling forms for adding a shift, bus or stop. . . . . 55
- 4.17 Checking database. . . . . 55
- 4.18 Informative functions . . . . . 56

# List of Tables

3.1 Use case description . . . . . 37  
3.2 Use case description . . . . . 39



# Chapter 1

## Introduction

Yachay Tech University is located in a rural sector of the Imbabura province, and as a proper characteristic of these sectors, its transportation service suffers from several issues, especially considering that the university campus is located at 2.5 km of the cantonal capital. To supply transportation, the university provides a service of institutional buses that cover the Yachay Tech residences - Ibarra route and vice versa, with seven intermediate stops: capitular room, institute “17 de Julio”, the cemetery of Urcuqui, Santiago del Rey, the “Los Antojitos de Colombia” bakery, the “Laguna Mall” mall, and “Parque de la Madre”. There also exists an internal service, that performs a shorter journey, using just three stops: capitular room, institute “17 de Julio” and the cemetery of Urcuqui.

However, the small amount of available units provided for each turn makes it difficult for students, administrative personnel and teachers to occupy the seats in an ordered way, producing disorder and inconveniences. Besides, it is a frequent situation that users waiting to use the transport service at bus stops different from the initial ones (“Parque de la Madre” for Ibarra - Urcuquí journey, and capitular room for Urcuquí - Ibarra journey) lose their time because the bus would arrive without available seats. Also, there are some units of the public transport available, those offered by the Urcuquí transport cooperative, but the lack of information about the schedule of these units makes this service inconvenient, since the passenger has to be waiting so much time to use this service.

Therefore, this project proposes the creation of a reserve system for institutional buses, that will allow to reserve seats and check availability of the institutional units in an ordered way. The proposed system will obey the most of established rules of the transport service, for instance: a person cannot reserve extra seats and the reserve system is strictly personal. Also, the reserve system will have spatial restrictions, allowing its use only when the user is located at a certain distance of the transport unit. This last condition is implemented to avoid unnecessary reservations, that is, they will not be occupied later, this is very frequent considering that the service is provided by free.

Another advantage of implementing this reservation system on the campus of the university, is that drivers can get rid of the responsibility of forgotten objects on the bus, that is, it will be known which person was driving the bus when the object was lost. This information can be obtained using registers, that will be created at the moment of

reserving a seat. Also, it will save time at the moment of entering the bus, since users will not have to register manually in the passengers list. In order to obtain these advantages, the system must be mandatory.

The main objective of this project is to implement the aforementioned reserve system in the campus, but it could be expanded to other universities where it could be useful, and in general, to any institution facing a similar problem: a high concurrency of people who need to improve their use of their internal transportation system.

## 1.1 Problem Statement

The use of the transport service in our university campus is not adequate and presents several inconveniences. We can name some: people waiting for a bus that will arrive full, people making large queues and pushing at the moment of getting in the bus, people who lost the bus although there are passengers who will leave the bus in the next stop, and so on. These are some of the problems that can be solved or, at least, in some way can be eased by implementing a mobile application for the internal transportation system.

## 1.2 Motivation

Considering that an internet service is available throughout the campus, and the majority of the university population has access to a mobile device (smartphone), building a mobile application to make the transport service more efficient, thus alleviating the problems described earlier, would be useful.

Besides, considering that the number of technologies and resources available to develop applications has increased and become more accessible, becomes feasible the objective of creating a mobile application to manage the internal transportation system.

## 1.3 Objectives

The general objective of this project is **to design and develop a reserve system for improving the service offered by institutional buses at Yachay Tech.**

To achieve this, a set of specific objectives has been defined:

- Comprehend the functioning scheme of the institutional transport service.
- Design a virtual reserve system intended to improve the scheme of service and to avoid inconveniences in the use of institutional transport, according to current regulations.
- Build a mobile app to access the reserve system using a development framework.

## 1.4 Overview

This thesis is organised as follows.

Chapter 2, “Theoretical Background” describes the main theoretical concepts that are needed to understand how the project was built.

Chapter 3, “Technological Proposal” describes the main functional and non-functional aspects of the mobile application in a technical sense.

Chapter 4, “Deployment and testing” presents a guide about how to perform a installation of the mobile application and how it was tested to prove its functionality.

Finally, Chapter 5, “Conclusions ad Recommendations” presents a set of remarks about the project achievements and some suggestions for the future.

# Chapter 2

## Theoretical Background

This chapter describes the main concepts that are needed in order to have a good understanding of the thesis and all the tools that it includes. Also, we present a concise review of how computerized reservation systems work through the world and some relevant facts about them.

### 2.1 Computerized reservation systems

The computerized reservation systems (CRS's) were designed to allow users to contract and confirm reservations for scheduled space in transportation services. Often, this service is limited by terms and conditions established previously by the provider, restrictions that are usually related to the schedule of such passenger space [1].

If we analyze the public transportation systems, it can be seen that CRS's have not been completely applied yet. For instance, let us consider the urban transport system in Bogota, the collection system is based in the sale of cards, each card giving the passenger access to one, two or ten travels. However, the system has demonstrated to be insufficient, some flaws are: queues for buying cards are very common, generating considerable delays; the cards cannot be read correctly sometimes, producing congestion; also the 80% of cards are those corresponding to one and two travel, while the cards of ten travels just have the 20% of the market share [2].

G. Perez [3] classified the public transport systems according to their payment method. This classification is not relevant, but details the objects frequently used: cards, electronic cospel, and cash itself. These alternatives have their own advantages/disadvantages, but the conclusion is that the evolution of these systems is leading to the m-commerce, that is defined as small mount transactions using mobile devices, a kind of service whose functioning is pretty similar to that of a CRS. This system offers improvements in terms of security, efficiency and operation cost with respect to traditional systems, but its main hassle is that it requires technological devices that could not be accessible for everyone.

Presently, with the boom of mobile devices with high capabilities called smartphones, and the increasing access to Internet all over the world, computerized reservation systems have evolved, and can be accessed from a mobile app, making the experience more interactive to the user.

The capabilities of computerized reserved systems have been expanded and are not limited to the transportation sector, we can now buy and reserve restaurant seats (ex. Eatigo, BigDish, OpenTable, Quandoo, etc), cinema seats (GMovies, TodayTix, Cinemax, etc), among other kinds of services. These apps work by showing offers from different providers, acting as a third agent and some are created by the owner of the organization that gives the service.

### 2.1.1 Airplanes

CRS's were created originally for airlines companies, to cover their own organization needs, and later the use of these services was applied to travel agencies as an associated sales channel. Since CRS's were created in the late 60's, their functioning and implementation required hardware considered very expensive for the epoch. The functions provided by the first CRS's were: information about scheduling of flights, rates, seats availability/reservation and tickets emission.

Before the use of CRS's, the process done by the travel agent to get one plane ticket was very cumbersome. First, the agent had to read airline schedules from either OAG (Official Airline Guides) or printed directories that were published in a monthly basis. Second, he or she had to call the desired airline to reserve and confirm passenger flights and annotate physically the tickets. After the implementation of CRS's, a travel agent had access to computerized travel data banks. Thus, he or she had the capability to reserve/confirm/transact airline, hotel or car rental reservations using a keyboard, a ticket printer, and a display terminal connected to the centralized reservation service [4].

In the airlines sector, we can find apps that allow us to search flights and to choose the best in terms of price, scheduling, airline company, etc. There are lots of options, such as: Kiwi, Hipmunk, Skiplagged, Hopper, Despegar, etc.

### 2.1.2 Buses

There exists a growing market for booking systems in autobuses. In many countries, bus tickets are expended at the time of boarding the unit. This could be harmful for tourists that do not speak the native language, who might end paying an extra charge. However, in some countries like Vietnam, there exists a service provided by high-end autobuses, that offer a comfortable and spacious vehicle, air conditioning, WiFi, sleeper seats, and even a toilet. Usually, these units are interprovincial. In this case, we have a booking system, that allows users to book tickets from a website. Also, it is frequent that we can book tickets from several websites, choosing lower prices.

In the software market there are a lot of alternatives to implement booking systems for autobuses. Provab Technosoft for instance, provides a reservation system, that includes supplier interface, route scheduling, seat selection by travelers, price and availability management. Also, it offers a software 100 % mobile ready, that would work perfectly on desktops, tablets and mobile phones [5]. Another solution is Bus Booking Software, provided by Brolmo company. It allows to organize bus routes and tours with ease. The software allows you to plan, schedule, and manage routes specifying start, end, and stop

destinations. Also, it allows to configure each bus with different ticket rates and discounts. An online payment method via PayPal is implemented in the system, too. It is based on a web page, since Brolmo is specialized in web apps [6]. In Vietnam, Ma Linh Express and Hoang Long are two of the most reputed operators that run these services across the country. Also, a 10 % discount is given, if the booking took place one week in advance [7].

## 2.2 Tools for mobile application development

This section will describe some of the available technologies and tools involved in mobile applications development. If we take into consideration the technique with which a mobile application was created we can find native applications and hybrid applications.

A native application is the one developed using the native programming language of the device, e.g. Objective C/Swift for iOS and Java for Android. Some advantages of this approach includes getting the best possible performance, designing the user interface following the guidelines of the platform and obtaining access to all device capabilities [8]. There are some tools for developing native applications, e.g., Android Studio and Xcode.

A hybrid application is built using a native container such as Phonegap/Cordova and using programming languages usually focused on web programming (HTML, CSS and JavaScript). The native container gives access to device capabilities and does not take into account the target operating system. Some of the best frameworks are: Flutter, Ionic, Xamarin, React Native and NativeScript [9].

### 2.2.1 Back-end and Front-end

In software engineering, the terms front-end and back-end refer to the different layers composing a system. Usually, they are the presentation layer (front-end), and the data access layer (back-end).

In the popular client–server model, an application is partitioned in tasks or workloads, between the providers of a resource, called servers, and service requesters, called clients [10]. In this model, the client is considered the front-end and the server is considered the back-end.

#### 2.2.1.1 Back-end

The back-end programming can either be Object Oriented (OPP) or functional.

The Object Oriented Programming is a technique focused on the creation of objects. In this paradigm, the statements must be executed in a particular order. The Functional Programming uses a declarative language, which means that statements could be executed in any order. In this project, an Object Oriented paradigm was used, specifically a server running under Node.js.

Node.js is a software tool, specifically a runtime, it is asynchronous and event-driven designed to build network applications. It has support for HTTP, then it is suitable for multimedia purposes since it allows streaming and has low latency. It makes Node.js a

good option to develop a web library. Unlike pure JavaScript code, the code written in Node.js is executed in the server [11].

Node.js only supports natively JavaScript, but there are many compile-to-JS languages, giving access to write Node.js applications using CoffeeScript, Dart, TypeScript, ClojureScript, and others. Node.js is used to build network programs such as Web servers. The difference between Node.js and PHP is that the latter blocks functions until completion, this means that commands only execute if the previous ones have finished, while Node.js functions are non-blocking, then they can be run concurrently or even in parallel [12].

### 2.2.1.2 Front-end

A front-end system is the part of an information system that can be directly interacted by the user to receive back-end services of the host system. It allows user to access and request features of the back-end. A front-end system can be either a software application, or a combination of hardware, software and network resources. The parts that constitute a front-end system could be a graphical user interface (GUI), and a front-end client application that is connected to the back-end system. Usually a front-end has very limited computational processing capabilities, and its functioning is based on the data and functions provided by the back-end in the host system [13].

## 2.2.2 Development Framework

A framework is a term used to describe a Software Development Kit (SDK) that can help to solve the complex problem of developing solutions for Data Integration projects such as: Data Warehousing, Data Consolidation, Data Migration, and System Integration. Among the tools included in a framework we can find: prebuilt reusable components, customisable templates and some standard practices for solving determined problems [14].

Some advantages of using a development framework are: they have good support and documentation, since a lot of programmers are using it; the most popular ones are open-source, then they are free to use; another advantage is their efficiency, many different applications have the same underlying code, and frameworks help to reduce the writing of that code again, saving time and money [15].

### 2.2.2.1 Ionic Development Framework

Ionic Framework is an open source UI toolkit for developing mobile and desktop apps using web technologies (HTML, CSS, and JavaScript). It can be used in the front-end part of a software system. It provides a lot of tools to improve the user experience in GUI matters (e.g., controls, interactions, gestures and animations).

Ionic can be integrated easily with other libraries or frameworks, or if the developer prefers, it can be used standalone. Officially, it has integrations with Angular and React, and soon it will have support for Vue [16].

Besides, Ionic relies on Apache Cordova, which is a powerful framework that allows the web or mobile application to use the device's native capabilities and then, to become a

native application. Combining these tools, makes Ionic a perfect platform to build hybrid applications [17].

### 2.2.3 Programming Languages

This section describes the main programming languages used in back-end and front-end programming.

#### 2.2.3.1 Back-end

Some popular languages used in back-end programming are:

- **Java**  
One of the most versatile languages. It can be used from computers to smart cards. Its versatility is based on the Java Virtual Machine (JVM). This Java Virtual Machine acts like a middle layer that allows one to execute code on any computer, regardless of where that code was compiled [18].
- **PHP**  
PHP is one of the most popular server-side programming languages. Since it is dynamically typed, it makes that the same portion of code can have a totally different meaning depending on the context, which makes programs written in PHP difficult to scale. It is recommended for beginners since it does not have a strong syntax, allowing code with errors to compile until that problematic zone is reached, also it has an abundance of resources for the language due to the large community [18].
- **Python**  
Python is the fastest-growing programming language. It is very accessible due to its clear and simple syntax. It is dynamically typed, open source, and object-oriented. It can be used in projects involving cross-platform shell scripting, quick automation, simple web development, and more [18].
- **SQL**  
Structured Query Language or SQL (pronounced ‘sequel’) is the most common query language. It is a declarative language, that allows developers to ‘declare’ the results they would like to obtain, without needing to specify the steps or process for that result. Usually, it is used together with development frameworks that allow raw queries to the database. It is very important in the data market, then learning SQL can benefit to financial analysts, data-driven marketers, and online entrepreneurs [18].
- **JavaScript**  
JavaScript can be used for both front-end and back-end. It is functional and dynamically typed, that makes it so flexible, but also it means that objects created using this language will be slow. It is also difficult to maintain and scale. The



advantages for beginners is that it needs just a little setup, since the code can be executed in a browser [18].

### 2.2.3.2 Front-end

This section describes the programming languages related to Ionic Framework, since it was the tool used in the front-end programming of this project.

- TypeScript

The TypeScript programming language is a strict superset of JavaScript, that adds a set of useful tools, being optional types one of the most important. Being a superset of JavaScript involves that any code written in JavaScript can be compiled by TypeScript. Adding optional types has many benefits for the programmer, for instance, the compiler can quickly catch obvious coding blunders, and editors are enabled to provide code completion through technologies such as Intellisense [19].

- HTML

The HyperText Markup Language is a language created to build web pages. It is relatively easy to learn, being accessible to most people, also, it is powerful, allowing to create a web page using just some tens of code lines. It is based on hyperlinks, that are a special text which allows to navigate among pages, using just a mouse click.

The code written in HTML, can be run directly in a web browser, and it is just necessary to dispose of a rudimentary text editor to start coding in it. Another special feature of HTML are tags, that can be recognized because they are written between triangle brackets `<>`. They allow to add images, tables, formatted text, and so on [20].

- CSS

Cascading Style Sheets (CSS) enable websites designers to separate layout from content in a web page. It allows one to customize the look and feel, changing the interaction of the final user with the software, all this without having to edit all pages conforming the site [21]. Using CSS to build websites has many advantages such as: consistency, since by making one change to your CSS style sheet, you can change the appearance of the whole website, saving time, specially in large pages; bandwidth reduction, when CSS separates your website's contents from its design layout, you dramatically reduce your file transfer size, reducing web hosting times and making your page load faster; and browser compatibility, since CSS stylesheets increase your website's adaptability and ensure that more visitors will be able to visit and use your page, in the way you intended [22].

## 2.2.4 Database Management Systems

A database is a collection of organized information that can be easily accessed, managed and updated. Typically, a computer database contains aggregations of data records or files organized into rows, columns and tables. [23].

A database allows one to store the information used by a mobile application, thus giving the possibility to process it and obtain aggregated data at any time. Having the data stored in registers allows one to separate a mobile application design from its content.

The database manager usually provides users with the ability to control read/write access, specify report generation and analyze usage. Some databases offer ACID (atomicity, consistency, isolation and durability) compliance to guarantee that data are consistent and that transactions are complete [24].

#### 2.2.4.1 PostgreSQL Database

PostgreSQL is an enterprise-class relational database management system, it is not just a database, it is also an application platform. It is very advanced, and it is a direct alternative to the best proprietary database systems: Oracle, Microsoft SQL, and IBM DB2. It has a great performance, in benchmarks, PostgreSQL either exceeds or matches in performance to the best database systems in the market, both open source and proprietary.

Some of the many advantages PostgreSQL offers to the final user are:

- Define additional data types to solve your problem. One can create complex numbers, triangles, duodecimal data types and so on. PostgreSQL also handles automatically data type creation on table definition.
- It has compatibility with numerous programming languages. It has inbuilt support for C, SQL, and PL/pgSQL, but you can enable support for additional languages such as PL/Perl, PL/Python, PL/JavaScript, PL/Ruby, and PL/R. It allows to manage and solve problems choosing the language that best fits to the problem.
- It has several options in administration tools, specifically psql, pgAdmin, phpPgAdmin, and Adminer. They all have a community of developers that adds functionalities with every new version, but PostgreSQL is not limited to these four tools. There are a lot more, both open source and proprietary [25].

## 2.3 Software Engineering Concepts

This section describes concepts involved with the process of developing a mobile application, and how it is composed.

### 2.3.1 Methodology for Mobile Applications

In software development, a methodology establishes a disciplined process upon software development with the objective of making software development easier and more efficient [26]. It is possible to group the software development methodologies in two types: traditional and agile. The traditional methodologies are plan driven, where first we establish and document a complete set of requirements, and next, we perform a high level/architectural design development. Due to these exigent requirements, these kind of methodologies are considered heavyweight, and do not respond well when changes

are even low. To satisfy software developments based on iterative enhancements, a new technique was created in 1975, and any methodology that uses this approach is called an agile methodology. The agile methodologies are more focused on people, interaction, working software, customer collaboration and changes, while traditional methodologies focus more on processes, tools, contracts and plans [27].

For this project, I used an agile approach as a reference methodology because it is more adaptive to future changes in system requirements, and the planning is not very detailed, then I could save time in the process. Also, as a strong characteristic of agile methodologies is the interaction with the clients, that in my case, would be two main groups: people providing the service (bus drivers, transport manager) and users (students, teachers, administrative staff). Then, I added features to the system even when the development process was already started, when they suggested any, without having to rely exclusively on the initial documentation as in traditional methodologies [28].

In this project, the methodology used as a model was the XP methodology. Specifically, the PXP, a version of XP that stands for Personal Extreme Programming, and as its name suggests, adapts the concepts of XP to a work team of one person.

The XP methodology is one of the most popular “agile” software methodologies. It is a set of values, principles and practices for fast development of high-quality software that tries to suit the needs of the customer in the fastest way possible. XP is extreme in the sense that it tries to impulse many software development “best practices” to their logical extreme [29].

I used PXP as my reference methodology because my teamwork was composed just by myself.

The methodology process in PXP can be decomposed in the following steps [30]:

- Planning

In this phase, the requirements statement are obtained. It can be shown using user stories, using some UML software products (e.g., Visual Paradigm), to depict them. Then, each user story is decomposed into features.

After this, features are grouped into meaningful groups denominated FeatureSets. After this, an estimation of FeatureSet size will be done to estimate the development time to give to each set. Finally, a FeatureSet-PriorityList can be created to sort the order of coding of FeatureSet.

Products: Complete schedule of the coding.

- Development

In this phase, the developing is started. Every FeatureSet is coded according to the order given by the FeatureSetPriority-List. It is important to consider that each FeatureSet will be decomposed into tasks, and this set of tasks will have a priority order too. Then, when these priorities have been established, the development can start. After each task is correctly implemented and tested, it is needed to update the refactor baseline and upload it to a control version manager, e.g., GitHub.

After finishing each FeatureSet implementation, it is needed to perform an integration test, and if no errors are detected, the code can be integrated into the production baseline. This process is repeated for any new FeatureSet added to the project,

until all the requirements have been fulfilled.

Products: Final beta version of the app.

- **Post Mortem**

In this final phase, the acceptance testing for code in production baseline will be finished. Then, the software is ready to be publicly released.

Products: Final building of the app.

It is important to remark that the PXP methodology was used as a reference, and this work does not reflect all the elements and products described by the methodology. It will be explained later in the proposal chapter.

## **2.3.2 Architecture of Mobile Applications**

The architecture used in a mobile application development depends on the type of information that will be displayed. In this sense, we can divide mobile applications in two groups: self-contained applications and applications with Internet connection [31].

### **2.3.2.1 Self-contained applications**

In this type of applications, the content is static. It means that its images, information, menus never change or rarely. Usually, this type of applications are developed natively, due to its structure. The development of a self-contained application can result simple, but if the information is subject to many changes, using this model could become non-viable. Some examples of this type of applications are: calendar, chess games, video players, etc.

### **2.3.2.2 Applications with internet connection**

In this type of applications, the information is stored in a back-end, which has a database that stores the information. These data are accessed by the application front-end through an API (application program interface), that is in charge of communicating the front-end with the back-end. A clear example of this type of applications is social media applications such as Facebook, Instagram, Twitter, etc.

The development of this type of applications is more complex, since it is necessary to build both the front-end and the back-end, and it is also necessary to rent a host to store the back-end in the cloud, then these factors increase the final cost of the application. The benefits are that the information can be dynamic, and it can be altered by any person that has access to the database. A scheme describing this type of applications can be seen in Figure 2.1.



Figure 2.1: Components of an application with internet connection

# Chapter 3

## Technological Proposal

This chapter describes both the functionality and architecture of the proposed mobile application. It includes the requirements that the mobile application fulfills and a set of UML diagrams that are useful to understand the application behavior.

### 3.1 Functional Requirements

These functional requirements were obtained after a process of research and consulting with the users to whom the mobile application is intended for. An interview with students and drivers lead us to identify the characteristics that the mobile application should incorporate. After considering both the aforementioned interviews and the internal guidelines, the functional requirements were written.

The functional requirements are:

1. When opening the app, it must show the main menu with the main options:
  - Reserve a seat
  - See bus location
  - History
  - Check bus schedule
  - Read internal transport regulations.
  - Settings
  - Help
  - About
2. When 'Reserve a seat' is selected, the app will show the number of available seats, transport unit that will make the journey and the departure time, but some constraints apply:

- The user location must be approximately  $x$  meters of the bus stop, where  $x$  is a distance measured in meters defined by the system administrator. This restriction allows to ensure that the person is going to use its seat, otherwise its space will be wasted.
- The reserve system will be enabled  $t_1$  minutes before departure time.
- The reserve system will be disabled  $t_2$  minutes before departure time.

The values for  $t_1$  and  $t_2$  will be defined by the system administrator, and these times are established to allow the bus owner dispose of the remaining seats for those persons that do not have the mobile application, or they could not use it by any circumstance. In this sense, the driver will allow people to use those seats in the last time lapse (between  $t_2$  and departure time).

3. When ‘See bus location’ is selected, the user will be able to see the real-time location of the transport unit doing the journey. For this first version of the mobile application it is considered that there is only one bus offering the service for each shift.
4. When ‘History’ is selected, the journeys in which the user has been will be shown.
5. When ‘Read internal transport regulations’ is selected, it will open the current guidelines for the transport service, so that users know how the service works, because these guidelines are the foundation for the app.
6. When ‘Settings’ is selected, it must be possible to adjust the user profile: name, birth date, account password, etc.
7. When ‘Help’ is selected, a small tutorial about how to use the app will be shown, to assist users that are starting to use the system.
8. When ‘About’ is selected, software version will be displayed.

## 3.2 Non-functional Requirements

The non-functional requirements are established according to the hardware conditions the application needs to run correctly. If a device does not meet any of these requirements, the system will not work as intended.

The non-functional requirements are:

1. The device must have GPS. In the case of passengers, it needs to be active at the moment of reserving a seat, and when asking for bus location. For the bus driver (if the bus GPS is not available), it should be active throughout all the journey.
2. The mobile application requires internet connection in order to be able to connect to the application server.

3. The device must run Android 5.1 or higher, with at least a 800 MHz processor and 1 GB RAM. A 4" screen is recommended.

## 3.3 Application Tutorial

The application has two presentations, due to there are two different types of users: passengers (either students, teachers or administrative personnel) and drivers.

This is because each type of user will enjoy a specific menu with different functions. For instance, there are specific functions for drivers (ex. add a stop), and for passengers (ex. reserve a seat).

### 3.3.1 General Functions

These functions are for both type of users passengers and drivers.

#### 3.3.1.1 Log in Page

The user can log in the system using his/her e-mail and password. The password is checked in the server, and will redirect the user to the Home Page if everything is correct

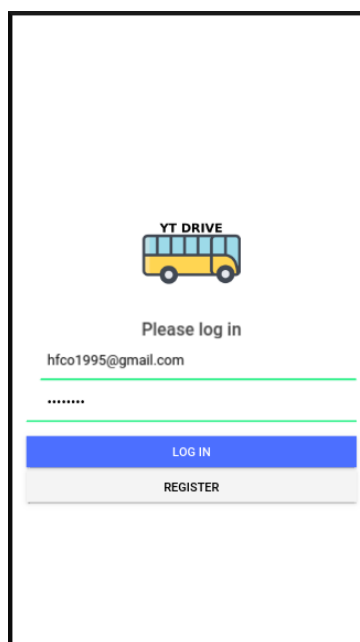


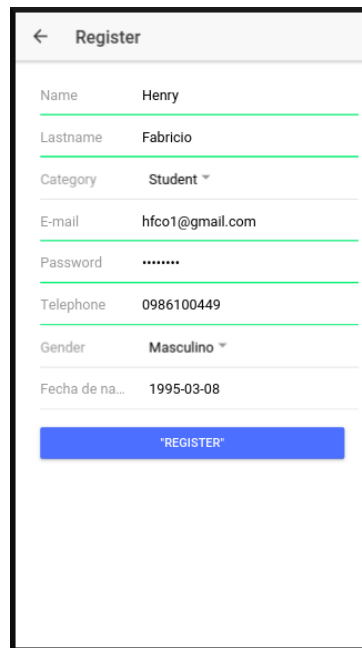
Figure 3.1: Log in Page

#### 3.3.1.2 Register Page

If the user is using the application for the very first time, he or she has to register by clicking on a button in the Log in Page. If a user wants to register in the system, he or



she must indicate some personal information such as: name, lastname, category, e-mail, password, telephone, gender and birthdate.



Field	Value
Name	Henry
Lastname	Fabricio
Category	Student
E-mail	hfco1@gmail.com
Password	.....
Telephone	0986100449
Gender	Masculino
Fecha de na...	1995-03-08

Figure 3.2: Register Page

Once a user is registered in the database, the application will redirect him to the Log in Page, so that he can start using the application.

### 3.3.1.3 Regulations Page

This page has an informative nature, here we can find a description of the current regulations in the internal transportation service. Also, we can find the list of stops enabled for the service. Finally, we can check the updated schedule for buses.

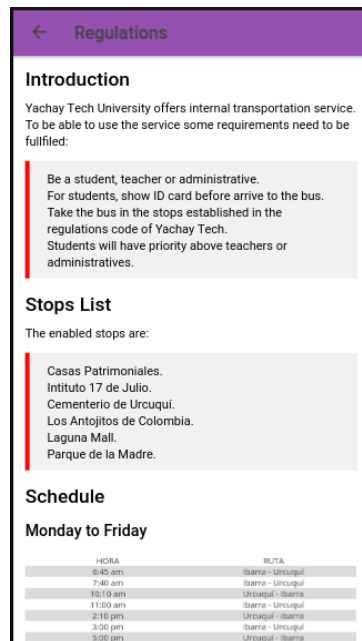


Figure 3.3: Regulations Page

### 3.3.1.4 Settings Page

In this page, we can modify some information about our profile. In the first section, we can change our name, lastname, email and telephone. In the second section, we can change our password.

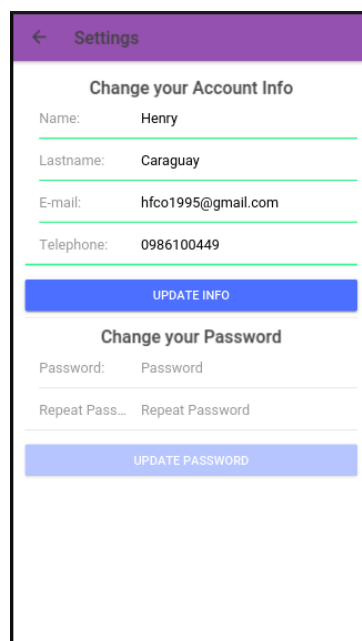


Figure 3.4: Settings Page

### 3.3.1.5 About Page

In this page, we have information about the application. The version number and developer's name.

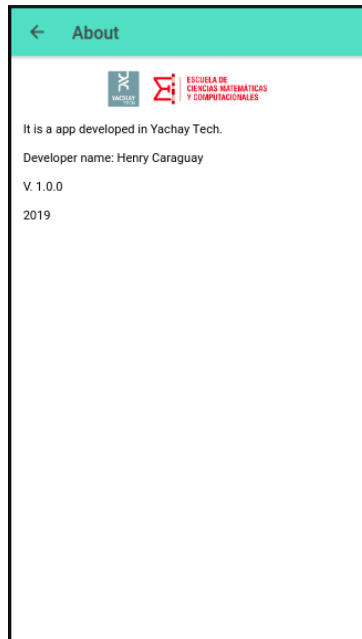


Figure 3.5: About Page

## 3.3.2 Passengers' Functions

These functions are those intended for the passengers of the bus, described here as users. A user can be either a student, a teacher, or a person from the administrative personnel.

### 3.3.2.1 User Menu

This is the user interface that a user sees when he or she has just logged in the system. It is composed of eight shortcuts to the functions of the app. The last four (third and fourth row) are general for both passengers/drivers, but the first four (first and second row) are specific to a passenger (either a student, teacher or administrative personnel).

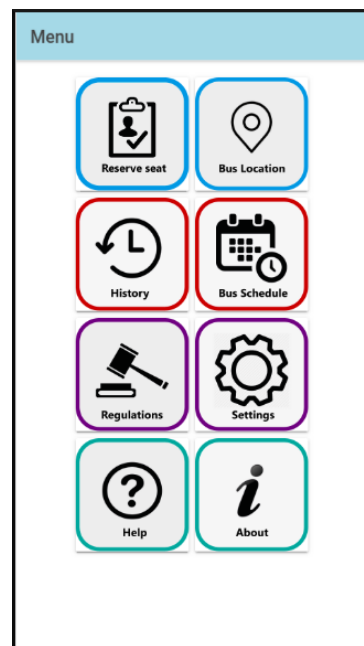


Figure 3.6: User Menu

### 3.3.2.2 Reserve Page

In this page, the user can reserve a seat for the next shift in the day. Two constraints are considered: a distance constraint, where the system asks the user to be inside a determined distance; and a time constraint, because the user needs to reserve its seat  $t_1$  minutes before departure time until  $t_2$  minutes before departure time,  $t_1$  and  $t_2$  being established by the system administrator.

The distance constraint is established to avoid unnecessary reservations in the system, due to the service is free. Then, if a passenger is near to a bus stop, we know he or she is going to use his or her seat.

The time constraint is established in order to enable the reserve system in the mobile application. For instance, the reserve system will be enabled between  $t_1$  and  $t_2$  minutes before departure, and the last time lapse between  $t_2$  minutes before departure and departure itself, will serve to the driver to give access to those passengers who could not use the mobile application for any reason.

If the user does not fulfill both of the constraints, he or she will see the screen 3.7:

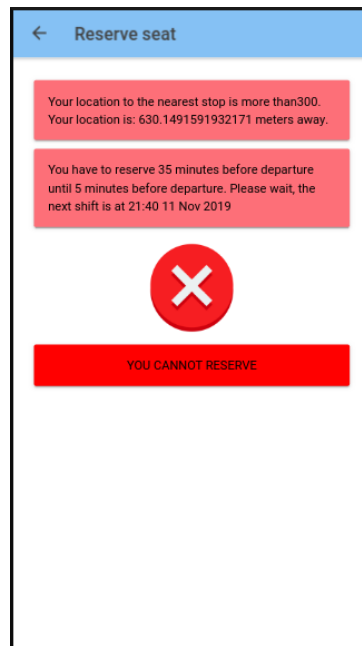


Figure 3.7: Error in Reserve Page

If the user has met both constraints, he will be enabled to reserve a seat, and then the screen 3.8 will appear, where the user can see the start time of the shift, approximated duration of the shift, the initial stop and the final stop. Also the user can see the bus assigned for the shift and the remaining seats in the bus.

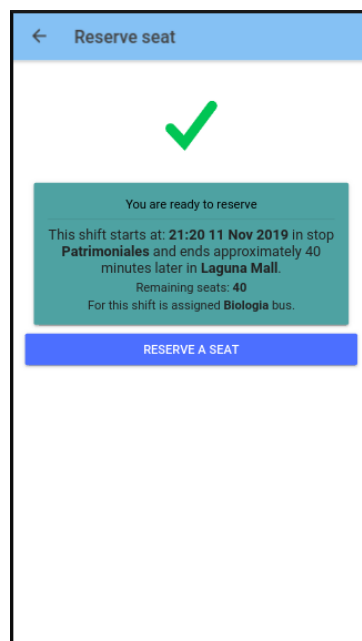


Figure 3.8: Ready to Reserve

Once a seat is booked, the user still has the option to cancel the reservation, as shown

in the screen 3.9.

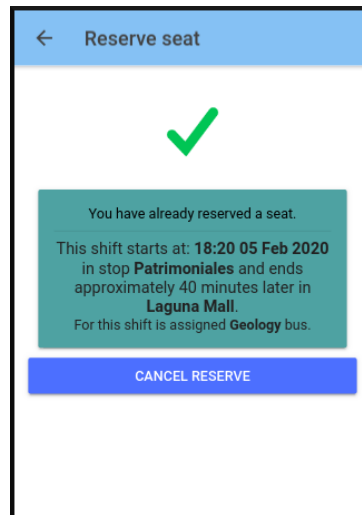


Figure 3.9: Delete reservation option

### 3.3.2.3 Bus Location Page

In this first version of the mobile application, we are considering that there is only one bus per shift. Since it is very difficult to obtain the information from the GPS sensor inside the vehicle, we are using the GPS information of the driver's phone.

It is very difficult to obtain the information directly from the bus due to two factors. The first factor is that due to administrative issues this information cannot be shared directly to third parties, but it is exclusively used by the university's transportation department, and the second factor is that even if we could obtain access to this information, in terms of programming, it is easier to use the GPS sensor of the mobile phone, since the programming framework can communicate directly with it.

In this page, the user can see the real-time location of the bus, but if the driver doing a shift does not send his location, the user will see the screen 3.10.



Figure 3.10: Bus Location Error

However, if the driver is sending his location, the user will be able to see the bus position in a map. This location will be updated every one minute until the driver finishes the location sharing, as shown in screen 3.11.

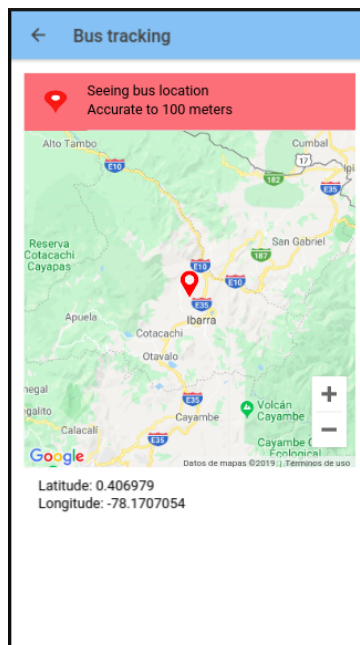


Figure 3.11: Bus Location Map

A marker indicating the position of the bus will be displayed every minute, making a trace in the map, thus the user can be aware of the position and direction of the bus.

This is useful for passengers that are waiting for the next bus, because they can see if the bus has or has not passed.

### 3.3.2.4 History Page

In this page, the user can check the history of his or her shifts. The user can see the date, time, and departure stop of the shifts he or she has taken, as shown in screen 3.12. The departure stop is taken from the moment the user books a seat, the arrival stop is not shown in the system, because it would require either a constant tracking of user's device location or a manual register done by the user, both being unfeasible.

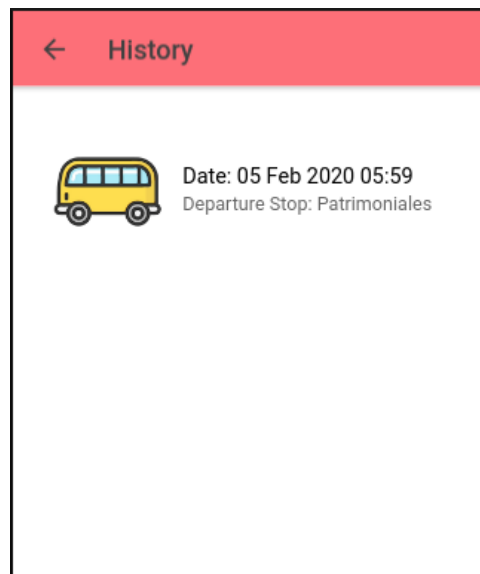


Figure 3.12: History Page



### 3.3.2.5 Schedule Page

In this page, the user can see the bus schedule in the current day. For each shift, the user can see the assigned driver, the time and date for departure, and the assigned bus to that shift.

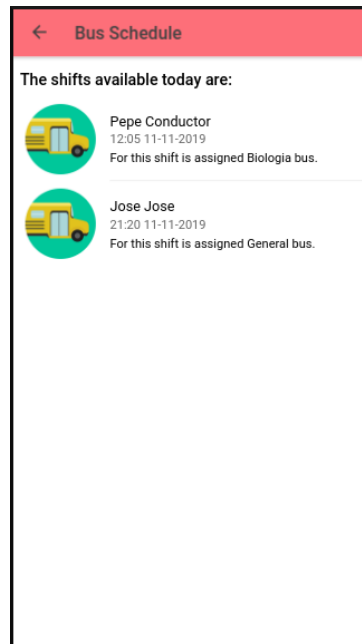


Figure 3.13: Schedule Page

### 3.3.3 Drivers' Functions

These functions are those intended for the drivers of the internal transportation service. These are mostly administration functions.

#### 3.3.3.1 Driver Menu

This is the user interface that a driver sees when he or she has just logged in the system. It is composed of eight shortcuts to the functions of the app. The last four (third and fourth row) are general for both passengers/drivers, but the first four are specific to a driver.

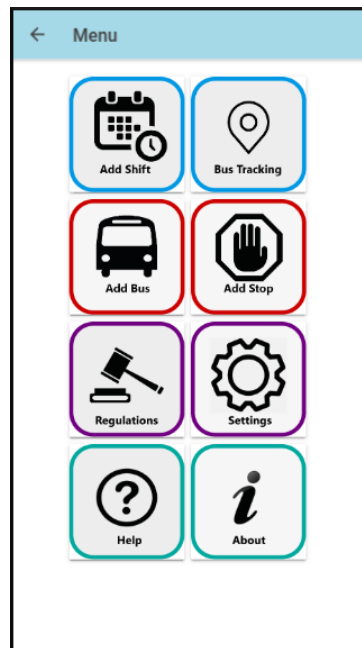


Figure 3.14: Driver Menu

### 3.3.3.2 Add Shift Page

In this page, the driver can add a shift in the system. The driver can specify the assigned bus, assigned driver, start time and finish time for the shift. This function is intended for those extra shifts that are not in the regular schedule.

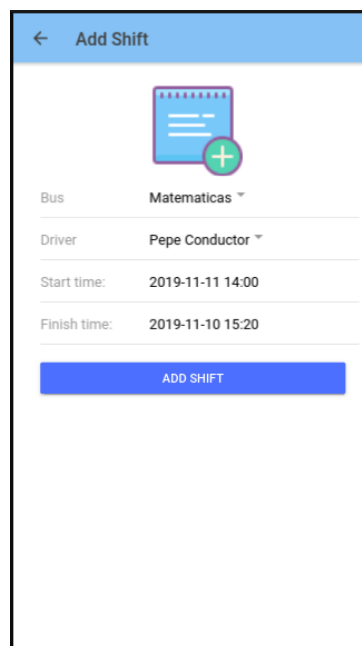


Figure 3.15: Add Shift Page

### 3.3.3.3 Bus Tracking Page

In this page, the driver can share his location as shown in Figure 3.16. It will be uploaded to the database, so that the user can start checking the bus location based on the driver's position. The driver can see in the application when he or she is sharing his or her position and also he or she has a button to stop sharing it, as indicated in Figure 3.17

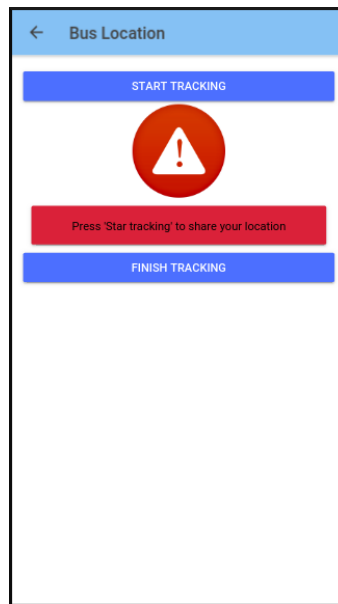


Figure 3.16: Driver not sharing his or her location

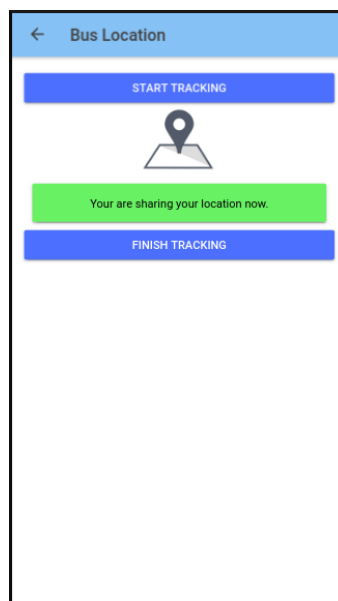


Figure 3.17: Driver sharing his or her location

### 3.3.3.4 Add Bus Page

In this page, the driver can add a bus in the system. The driver can specify the school that owns the bus, the color of the bus, its license plate, a boolean indicating if the bus is operative and the capacity of seated passengers. The responsible of the transport service can delegate this task to some driver, while there is not a designated system manager . This function has an auxiliary character, since the acquisition of new buses is not frequent.

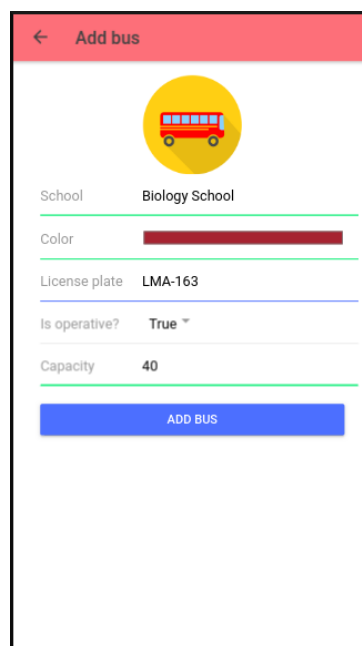


Figure 3.18: Add Bus Page

### 3.3.3.5 Add Stop Page

In this page, the driver can add a stop in the system. The driver needs to indicate the name, latitude and longitude coordinates of the new stop. This data can be obtained easily using an application server of web maps (ex. Google Maps). It is also an auxiliary function, and its use might not be frequent.

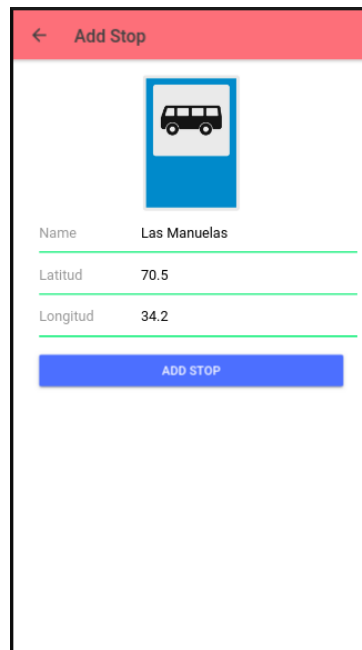


Figure 3.19: Add Stop Page

## 3.4 Application Architecture

In this project, we will use a Client - Server architecture. It is composed of two main components: a mobile android application (front-end) and a Node.js server (back-end).

The mobile application is developed using the Ionic framework, that is based on languages such as TypeScript (a JavaScript's super language), HTML and CSS. The mobile application communicates through a REST API, implemented in a Node.js server, with the database. The system information is mounted in PostgreSQL, an open source database management system.

In the Figure 3.20, the application layer defines the protocols that are used to interchange data between the Node.js server, the GPS sensor (that communicates via Cordova environment), and the Ionic Framework. Finally, the mobile client (a.k.a. mobile application) interacts with final users.

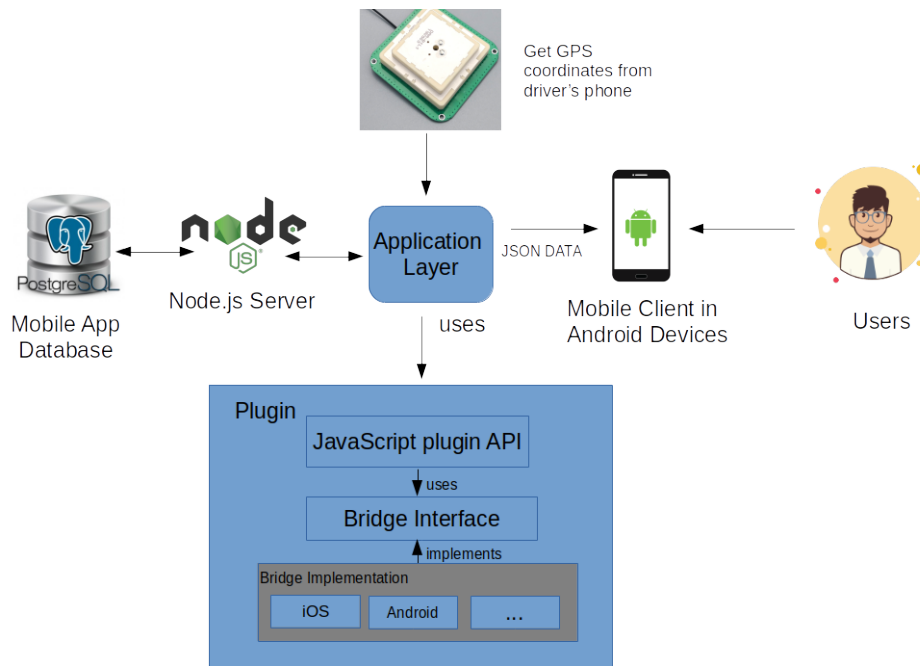


Figure 3.20: Application Architecture Diagram

### 3.5 Use Case Diagrams

This diagram shows the relation between actors and application from a general perspective.

#### 3.5.1 Use Case Diagram for Users

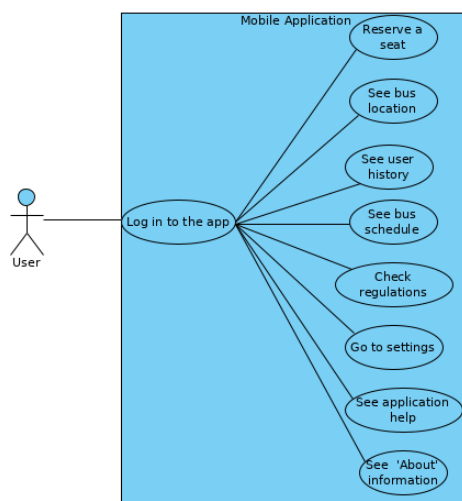


Figure 3.21: Use case for Users

<b>Use Case</b>	Application Diagram
<b>Actor</b>	User
<b>Description</b>	The user needs to enter his e-mail and password in order to start session in the system and use the application functionalities.
<b>Precondition</b>	The application must be installed in a smartphone running Android and must have access to internet.
<b>Activities</b>	1. Reserve a seat: allow to reserve a seat in the next bus shift.
	2. See bus location: allow to check the bus position in real time ( $\pm 1$ min)
	3. See user history: allow to check all shifts taken by the user.
	4. See bus schedule: allow to check all shifts in the current day.
	5. Check regulations: allow to see the guidelines for the internal transportation service
	6. Go to settings: allow to update personal information and modify password.
	7. See application help: a small tutorial to learn how to use the application.
	8. See about information: version number, and other minor details.
<b>Alternative Flow</b>	If the user cannot remember his credentials, he has an unlimited number of attempts, or he can ask help to the system manager to reset his password.

Table 3.1: Use case description

### 3.5.2 Use Case Diagram for Drivers

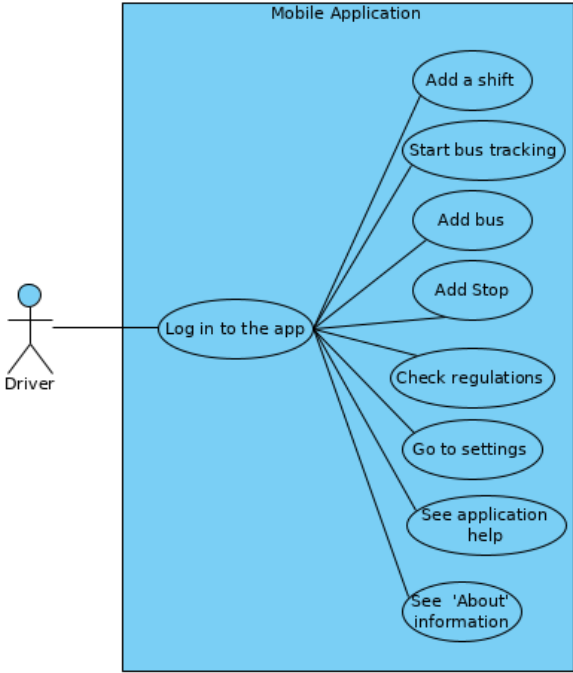


Figure 3.22: Use case for Drivers



<b>Use Case</b>	Application Diagram
<b>Actor</b>	Driver
<b>Description</b>	The driver needs to enter his e-mail and password in order to start session in the system and use the application functionalities.
<b>Precondition</b>	The application must be installed in a smartphone running Android and must have access to internet.
<b>Activities</b>	1. Add a shift: allow to register a new shift, selecting a specific date and hour.
	2. Start bus tracking: allow the driver to send his position to the database, then users can track the bus position based in the driver's smartphone position.
	3. Add bus: allow to add a bus using a form where bus information must be added (school, color, license plate, etc).
	4. Add stop: allow to add a stop indicating its name and coordinates.
	5. Check regulations: allow to see the guidelines for the internal transportation service
	6. Go to settings: allow to update personal information and modify password.
	7. See application help: a small tutorial to learn how to use the application.
	8. See about information: version number, and other minor details.
<b>Alternative Flow</b>	If the driver cannot remember his credentials, he has an unlimited number of attempts, or he can ask help to the system manager to reset his password.

Table 3.2: Use case description

### 3.6 Database Entity Relationship Diagram

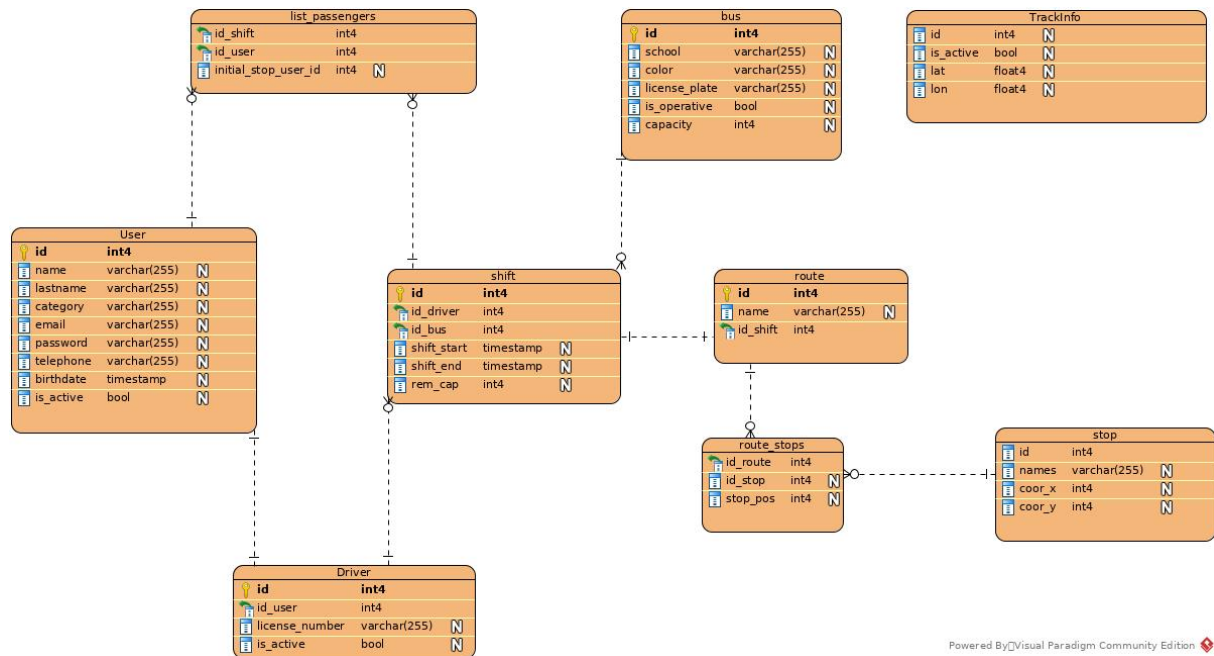


Figure 3.23: Database Entity Relationship Diagram

The Database Entity Relationship diagram (DERD) shows the relationships of entities stored in a database, with each entity having a group of attributes. For this project, the DERD can be seen in Figure 3.23.

- **User**: this table stores the personal information for all users in the system.
  - id**: A number that identifies each user in the system. It is a primary key.
  - name**: The first user's name.
  - lastname**: The first user's last name.
  - category**: This specifies if a user is a student, teacher or if it is part of the administrative personnel.
  - email**: User's e-mail.
  - password**: User's hashed password.
  - telephone**: User's phone.
  - birthdate**: User's birthdate.
  - is\_active**: A boolean to indicate if a user is active in the system, for administration purposes.
- **List\_passengers**: this table stores information about passengers in each shift.
  - id\_shift**: The id of each shift that has been made. It is a foreign key.
  - id\_user**: The id of the user that took the shift. It is a foreign key.
  - initial\_stop\_user\_id**: The id of the stop where the user entered to the bus, taken

by the system at the moment of reserving a seat.

- **Driver:** this table stores specific information about drivers.
  - id:** The id of each driver. It is a primary key.
  - id\_user:** The driver is also an user in the system, then to avoid repeating information, he or she is linked to the table User using this id. It is a foreign key.
  - license\_number:** The driver's license.
  - is\_active:** A boolean to indicate if the driver is enabled to offer the service. Useful for later administration purposes.
- **Shift:** this table stores the information for each shift.
  - id:** The id of each shift. It is a primary key.
  - id\_driver:** The id of the driver assigned for each shift. It is a foreign key.
  - id\_bus:** The id of the bus assigned for each shift. It is a foreign key.
  - shift\_start:** The date and time for shift's start.
  - shift\_end:** The date and time for shift's end.
  - rem\_cap:** The remaining capacity for each shift.
- **Bus:** this table stores the information for each bus.
  - id:** The id of each bus. It is a primary key.
  - school:** The school where each bus belongs to.
  - color:** The color of the bus. Useful for identification purposes.
  - license\_plate:** The license plate of the bus.
  - is\_operative:** A boolean indicating if the bus is available or not.
  - capacity:** The available capacity, this is the number of seats, for each bus.
- **Route:** this table stores the information for each route. A route is a set of stops in an specific order.
  - id:** The id of each route. It is a primary key.
  - name:** A descriptive name for each route, e.g., "Ibarra - Urcuquí".
  - id\_shift:** The id of the shift(s) that are assigned to a route. It is a foreign key.
- **Route\_stops:** this is an auxiliary table that allows to specify the order of each stop in a route.
  - id\_route:** The route's id.
  - id\_stop:** The stop's id.
  - stop\_pos:** The order position of the stop in the route.
- **Stop:** this table stores the information about each bus stop.
  - id:** The id of each stop. It is a primary key.

**names:** A name to identify each stop, e.g., “Laguna Mall”  
**coord\_x:** The latitude coordinate of the stop.  
**coord\_y:** The longitude coordinate of the stop.

- **Trackinfo:** this table stores the information about driver position (used as bus position) in the current shift.  
**id:** The id of each tracking information.  
**is\_active:** A boolean to know if the driver is sharing his or her position.  
**lat:** The latitude coordinate of the driver’s position.  
**lon:** The longitude coordinate of the driver’s position.

### 3.7 Activity Diagrams

The following diagrams describe dynamic aspects of the mobile application. It can be seen as an advanced version of flow chart.

#### 3.7.1 General functions

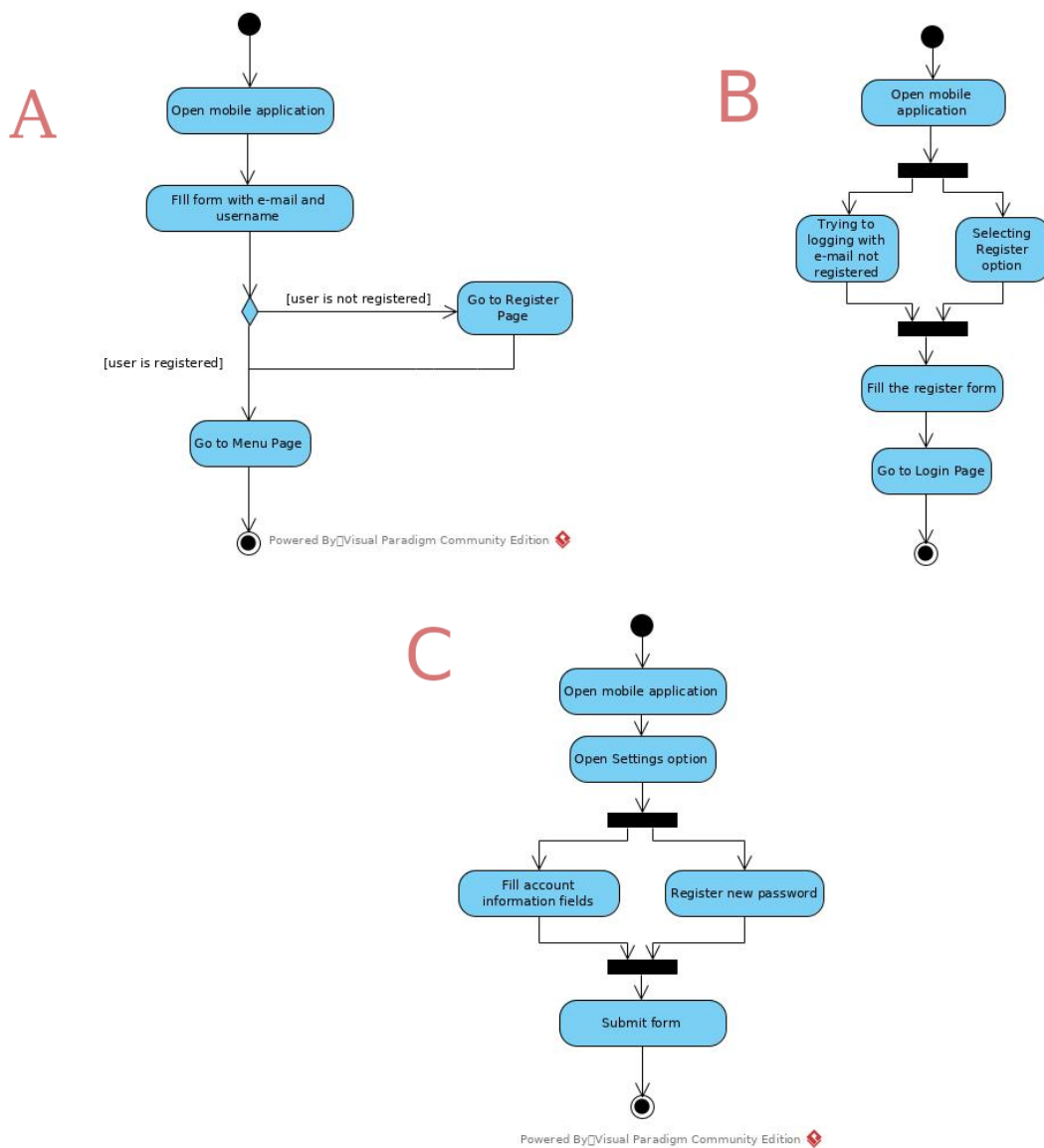


Figure 3.24: Activity diagrams for login (A), register (B) and settings (C) functions .

### 3.7.2 User Functions

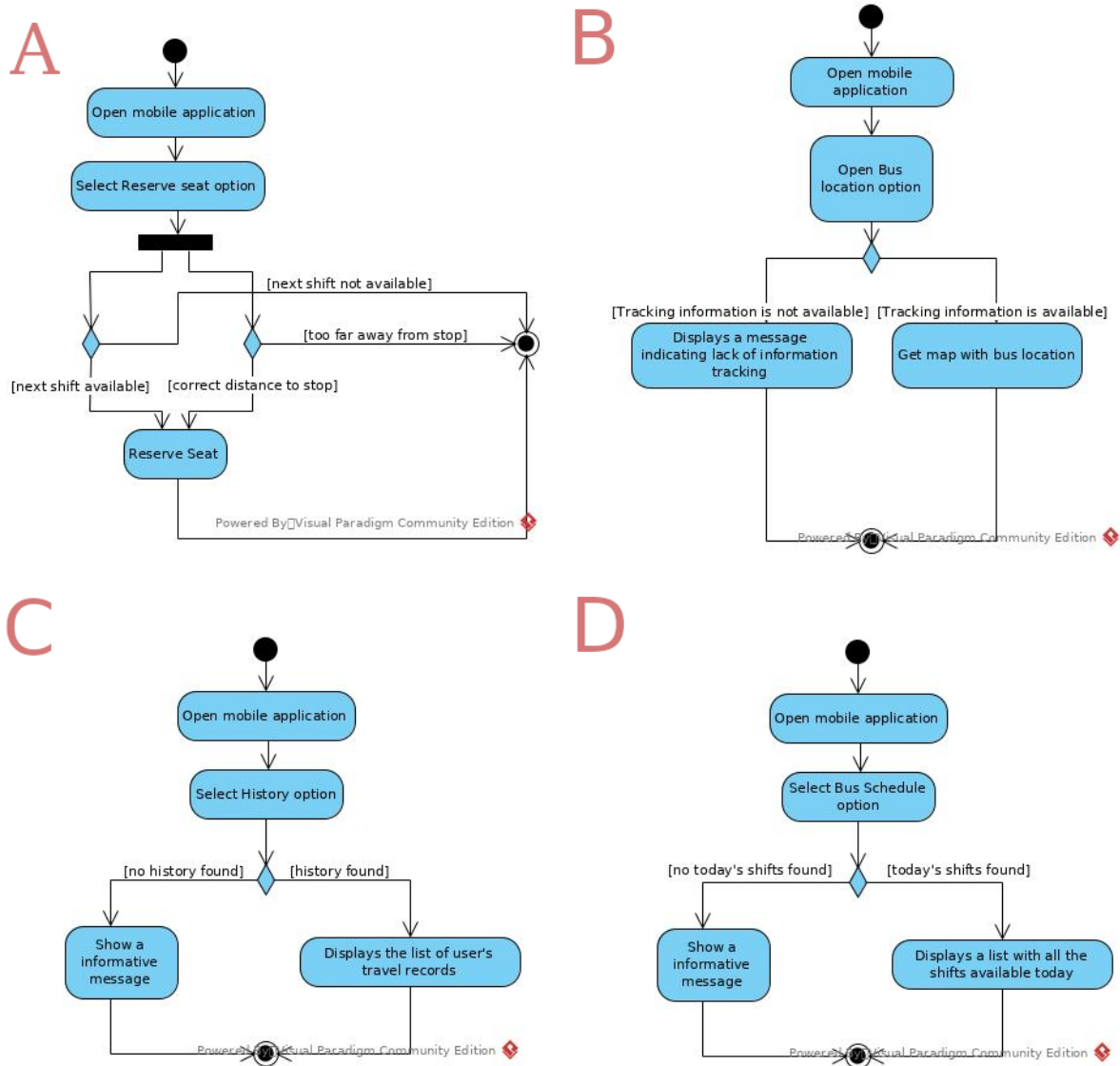


Figure 3.25: Activity diagrams for reserve (A), bus location(B), history (C) and bus schedule (D) functions.

### 3.7.3 Driver Functions

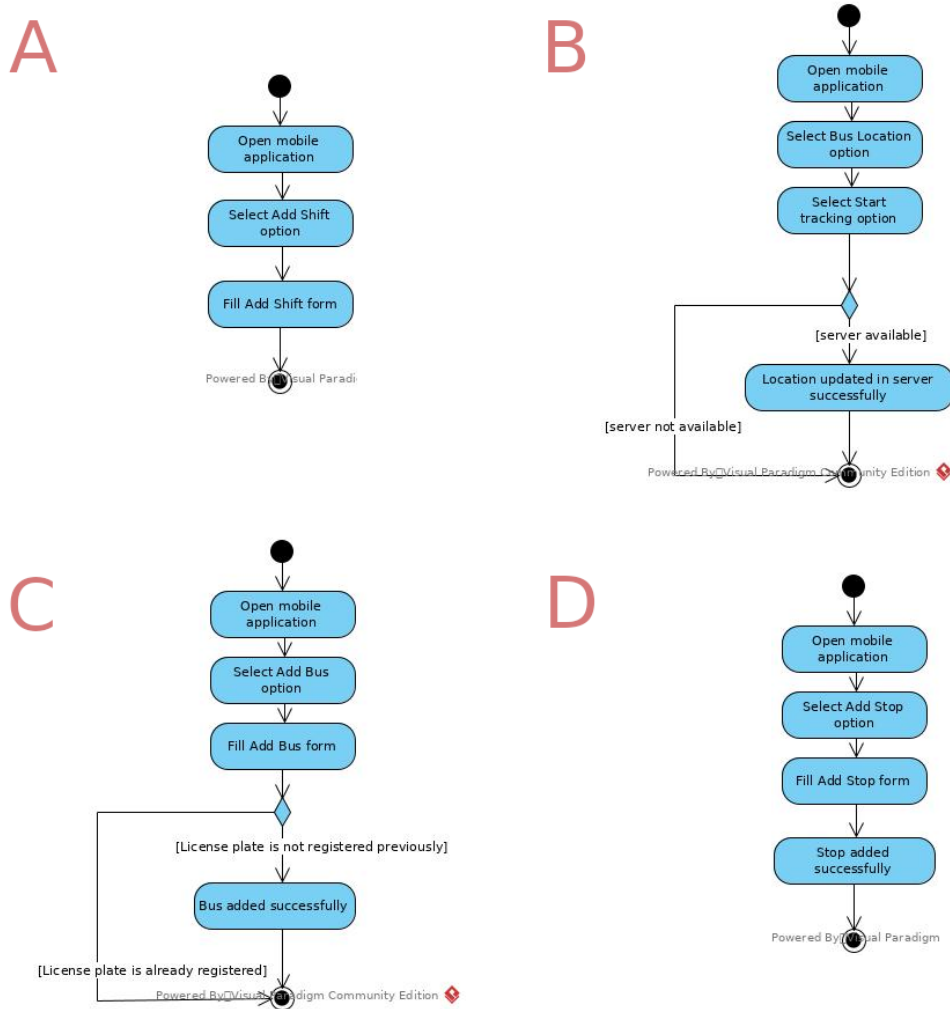


Figure 3.26: Activity diagrams for add shift (A), bus location(B), add bus (C) and add stop (D) functions.

# Chapter 4

## Deployment and testing

This chapter presents an overview of how the application was deployed. Some functionality tests will be performed to assure that the mobile application has the intended behavior.

### 4.1 Deployment

This section describes all the steps required to run the application. It includes from the building of the android mobile application up to the back-end configuration in a cloud platform.

#### 4.1.1 Front-end configuration

This section will show some settings needed in the front-end code, in order to build the app, and connect it flawlessly to the back-end.

##### 4.1.1.1 Building both presentations of the app

Since the application has two presentations, one intended for passengers and another for drivers, each version is built separately.

The front-end has in its source code all the pages for both versions, but depending on which presentation is built, it is necessary to modify the home page to which a user is redirected after performing a login in the system. This setting is located in the login page.



```
// Attempt to login in through our User service
doLogin() {
  this.user.login(this.account).subscribe((resp) => {
    this.navCtrl.push(HomePage); // Users
    // this.navCtrl.push(DrivermenuPage); // drivers
  }, (err) => {
    this.navCtrl.push(RegisterPage);
    // Unable to log in
    let toast = this.toastCtrl.create({
      message: "User not found. Please register",
      duration: 3000,
      position: 'top'
    });
    toast.present();
  });
}
```

Figure 4.1: Function which defines the page to where the user will be redirected in the login

Once the code of the front-end is ready, it can be built running in a console the command `ionic cordova build android` which generates a file with extension apk, that is the file format for installers in android. This installer can be distributed to any user with a device that fulfills the requirements, or can be uploaded to a store such as Google Play, F-droid, etc.

#### 4.1.1.2 Connecting API REST and network settings

After the back-end is configured in a cloud platform, the domain, to which our REST API is connected, must be defined. In this project Heroku was chosen as the cloud platform, for its advantages, these are described in the back-end configuration section 4.1.2.

In the Ionic Framework, the REST API is defined as a provider. It is called 'rest' and define basic CRUD operations such as: get, post, put and delete. The connection url is defined as: `url: string = 'https://ytdrive.herokuapp.com'`.

Now, to terminate the back-end connection, the network security settings of the mobile application must be configured. This element is specific to the Android operating system, it requires explicitly declaring Internet domains to allow connection to the mobile application. Otherwise, there will be no connection to the back-end, or other Internet domains.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config cleartextTrafficPermitted="true">
    <domain includeSubdomains="true">localhost</domain>
    <domain includeSubdomains="true">https://ytdrive.herokuapp.com</domain>
    <!-- <domain includeSubdomains="true">192.168.100.4</domain> -->
    <!-- <domain includeSubdomains="true">172.21.2.56</domain> -->
  </domain-config>
</network-security-config>
```

Figure 4.2: Adding allowed subdomains

## 4.1.2 Back-end configuration

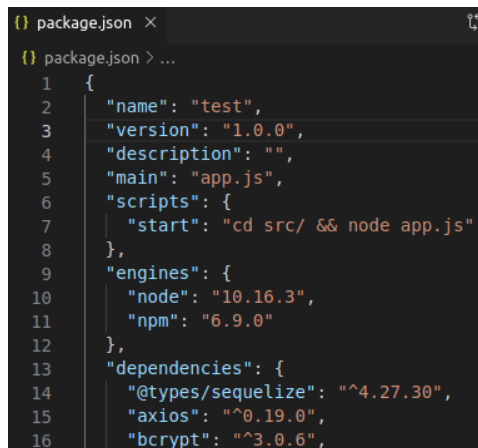
To be able to use the mobile application in a real environment, the back-end must be configured in a cloud. After researching the available cloud platforms, Heroku was chosen by its advantages.

Heroku is a cloud platform that gives facilities to developers to build, deliver, monitor and scale apps. It has free plans, allowing developers to test their apps in a minor scale, rather than having to pay to do so.

Also, Heroku has plugins, that allow to configure our database, send sms and e-mails, edit the content in the case of websites/blogs, and more.

### 4.1.2.1 Configuring Node.js server in Heroku

First, a new project must be created in *heroku.com*. To deploy the Node.js application, a.k.a back-end, it must be stored in Github, a version control management software. Now, in Heroku, you can select the branch of the software to be implemented. To avoid any type of conflict, it is advisable to specify the version used of all engines (node, npm) and dependencies, it is usually done in the file *package.json*.



```
package.json ×
package.json > ...
1  {
2    "name": "test",
3    "version": "1.0.0",
4    "description": "",
5    "main": "app.js",
6    "scripts": {
7      "start": "cd src/ && node app.js"
8    },
9    "engines": {
10     "node": "10.16.3",
11     "npm": "6.9.0"
12   },
13   "dependencies": {
14     "@types/sequelize": "^4.27.30",
15     "axios": "^0.19.0",
16     "bcrypt": "^3.0.6",
```

Figure 4.3: File with specific version of our Node.js dependencies

Finally, there are some final details that must be adjusted. First, a domain to access the back-end server is needed. Heroku gives a domain by default, but it can be changed in the application settings, inside Heroku's control panel. Also, the URL of our database must be declared in the configuration variables, inside Heroku platform, this database also needs to be online.

### 4.1.2.2 Migrating our database to Heroku

At the start of the mobile application development, a local Postgres database was used, in order to use it with the back-end running in Heroku, it must be migrated to the Heroku

platform using the Postgres add-on, this is a plugin that allows one to upload our database using the Heroku CLI, shown in Figure 4.4.

```
henryf3@henryf3:~$ heroku apps
=== hfco1995@gmail.com Apps
bookyt2019
ytdrive
```

Figure 4.4: Accessing to Heroku using CLI

Once the database has been migrated, it can be accessed using an online database manager, in this case Adminer. To have access to the database, one needs to obtain the credentials in the Heroku platform, as shown in Figure 4.5. After that, the timezone in the Heroku database must be adjusted, in order to avoid conflicts when working with time functions.

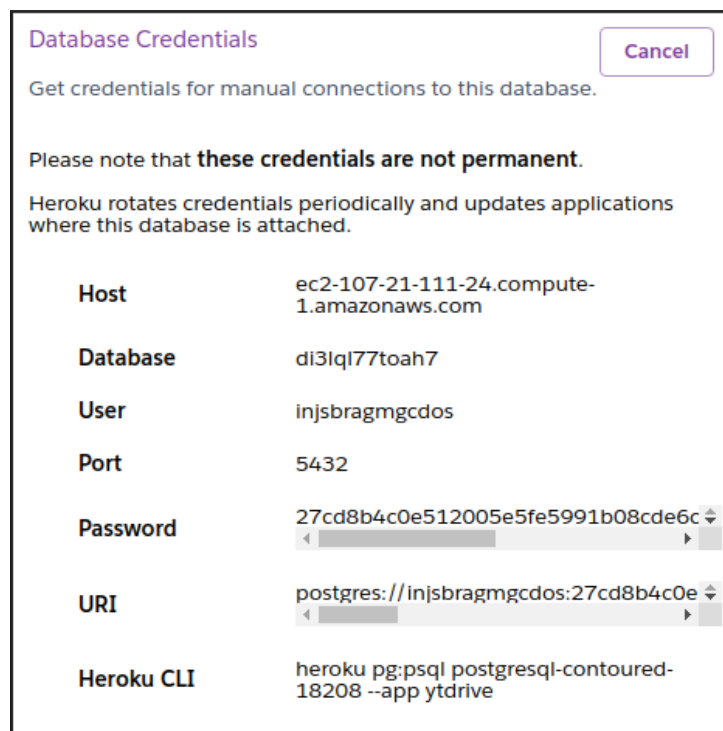


Figure 4.5: Database credentials

## 4.2 Testing

In this section, the functions of the mobile application will be tested, to verify that it is working as expected. In order to check this, after performing some action that involves writing to the database, those changes will be checked.

### 4.2.1 Login and register

In this case, the creation of a new account in the system will be checked.

First, the user needs to open the application and select the 'Register' option, fulfill the form and confirm the register. Finally, the new user can login using the e-mail and password, this process can be seen in Figure 4.6.

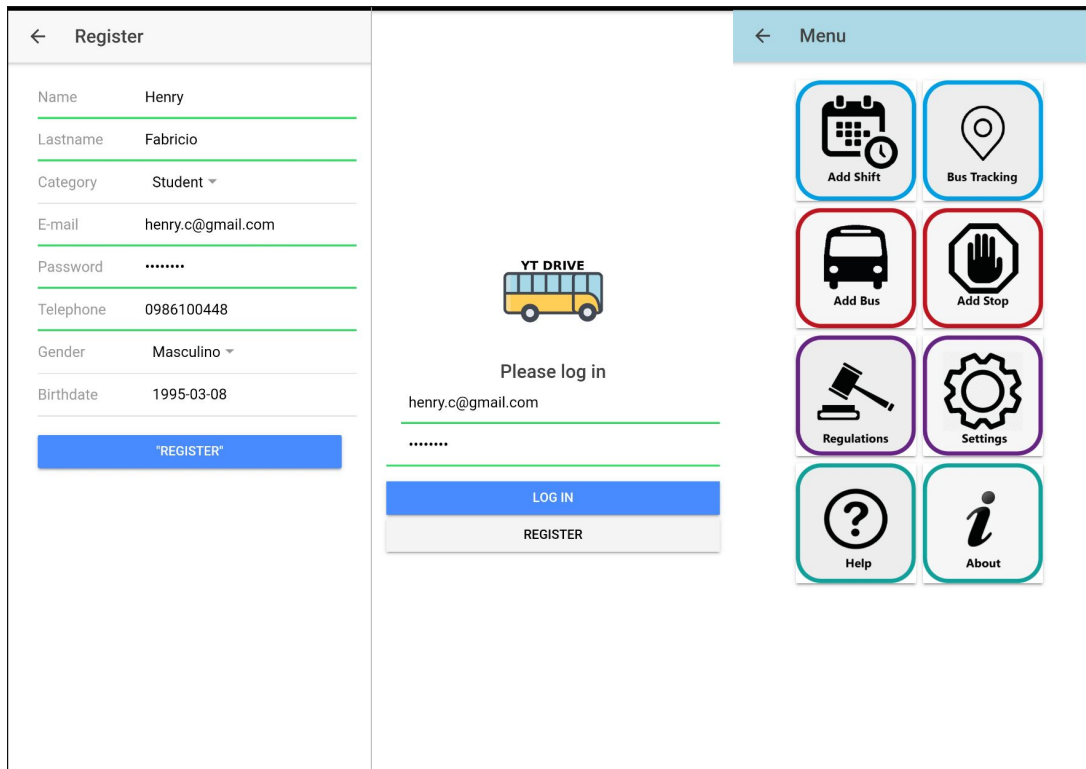


Figure 4.6: Register - Login Process

When verifying the database, a new user has been registered and his password has been stored after passing a hash process, as seen in Figure 4.7.

id	name	lastname	category	email	password
4	Pepe	Conductor	Administrative	pepedrive@gmail.com	\$2b\$04\$FgGGkWq6mFwnk
2	Marta	Higareda	Teacher	mhgreda@gmail.com	\$2b\$04\$FgGGkWq6mFwnk
3	Jose	Jose	Administrative	josejose@gmail.com	\$2b\$04\$FgGGkWq6mFwnk
5	Henry	Fabricio		hfco1@gmail.com	\$2b\$04\$EBk.BDF1KcVsGd0
1	Henry	Caraguay	Student	hfco1995@gmail.com	\$2b\$04\$EBk.BDF1KcVsGd0
6	Henry	Fabricio	Student	henry.c@gmail.com	\$2b\$04\$y9DxbUyQ/fg0NjyU

Figure 4.7: Database with new user registered

### 4.2.2 Reserve

For this function, the behavior of the mobile application is analyzed, first, at the moment of reserving a seat, and after, at the moment of canceling the reservation.

Before reserving a seat, the application will check if the user meets the constraints of time and distance, if so , it will display the information for the next shift and the remaining seats in the shift, as shown in Figure 4.8, also in this figure, the database shows the remaining capacity of the current shift.

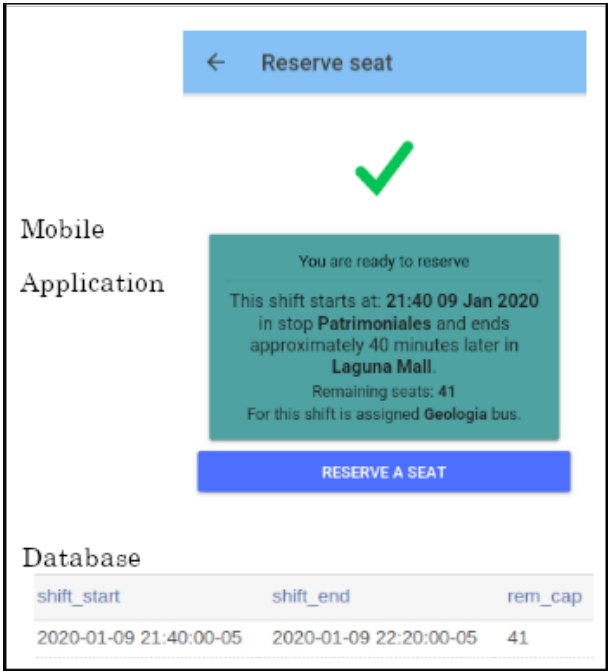


Figure 4.8: Status of mobile application/database before reserve

After performing a reserve, the mobile application shows that a seat has been already reserved, and the remaining seats in the database has also decreased in one, as shown in Figure 4.9.

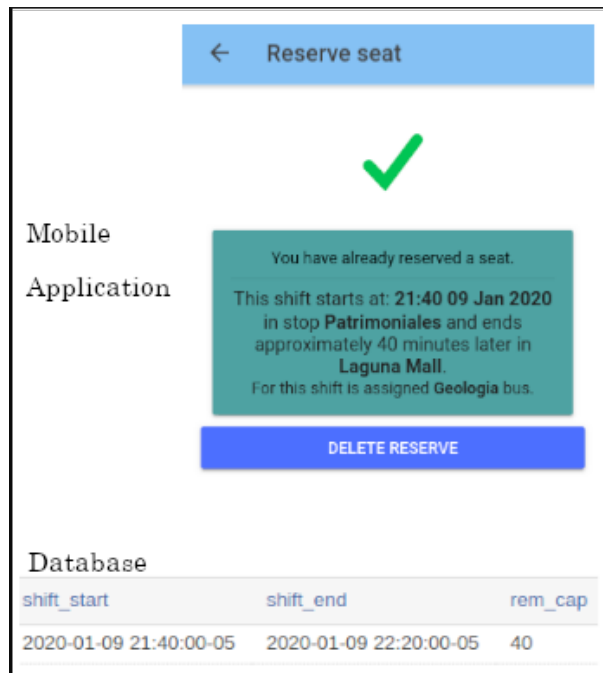


Figure 4.9: Status of mobile application/database after reserve

Also, if the user cancels the reservation, the changes made in the database are undone, and thus, the mobile application is working correctly.

### 4.2.3 History and Bus Schedule

These functions are reading from the database, and if the user have data available, it displays a list with the required information, as shown in Figure 4.10

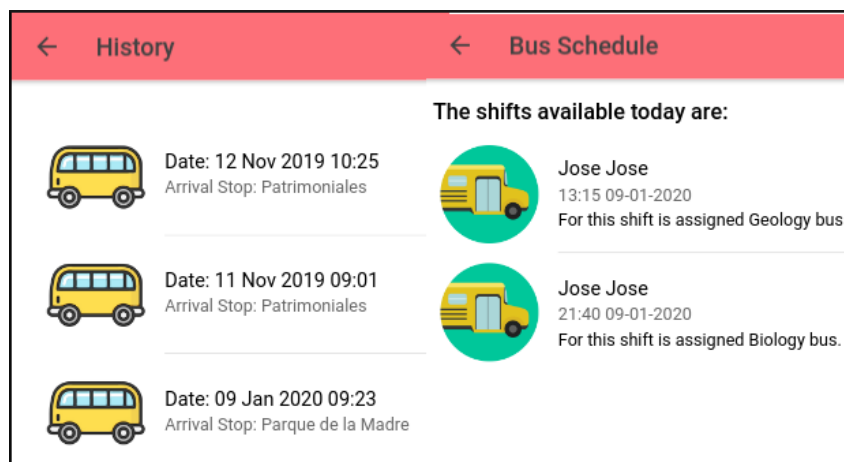


Figure 4.10: History and Bus Schedule

If there was no data available, the application would show a message indicating that, as shown in Figure 4.11

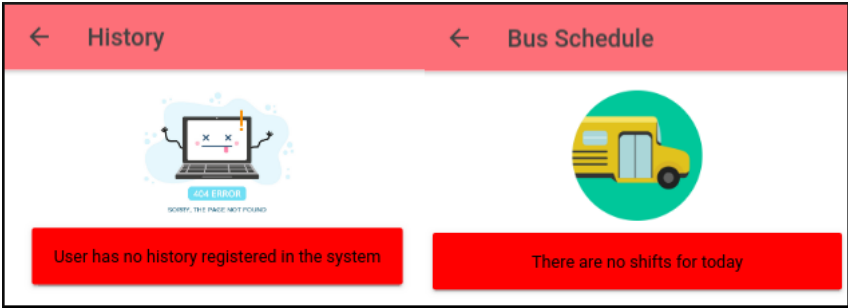


Figure 4.11: History and Bus Schedule not available

### 4.2.4 Bus Location and Tracking

For this function, a driver must be using his phone and propagating his or her location, since it acts as the bus location. If a driver is not sharing his or her location through his or her phone, the user will see a message as shown in Figure 4.12.

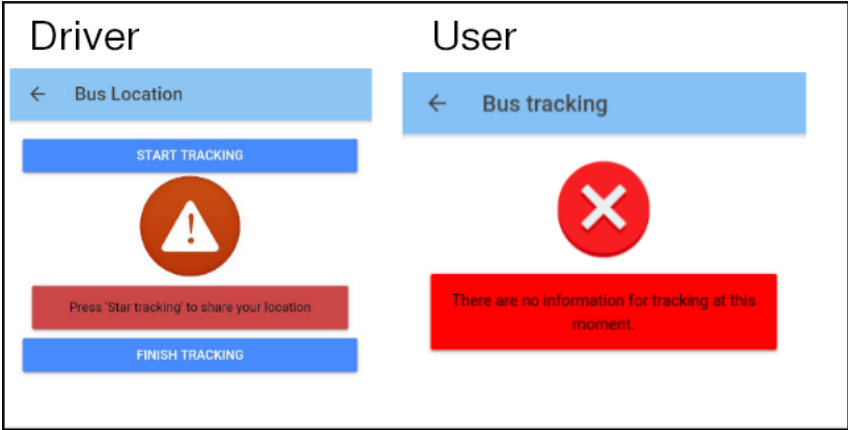


Figure 4.12: Bus Tracking not available

Once a driver is sharing his or her location, it will be uploaded to the database every sixty seconds, and the user finally can see the position of the bus in a map using red markers, as shown in Figure 4.13. Meanwhile, the driver can finish the bus tracking whenever he or she considers it is not relevant. In the tests performed, there were no troubles in using this function.

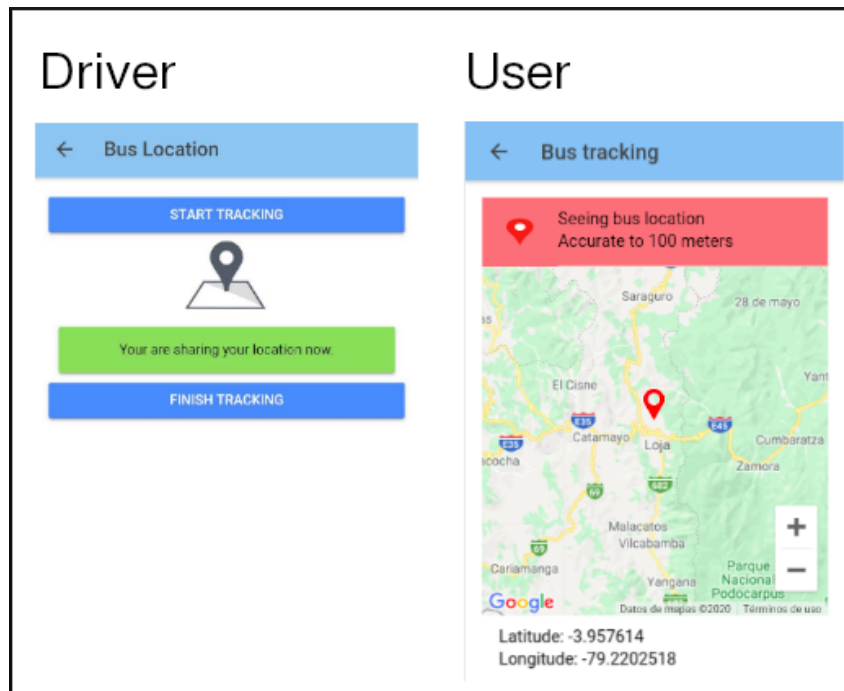


Figure 4.13: Bus Tracking in execution

### 4.2.5 Settings

In this option, the user can change some details of his or her account. As shown in Figure 4.14, the user can change details like the name, lastname, e-mail and telephone. Besides, the user can change his or her password.

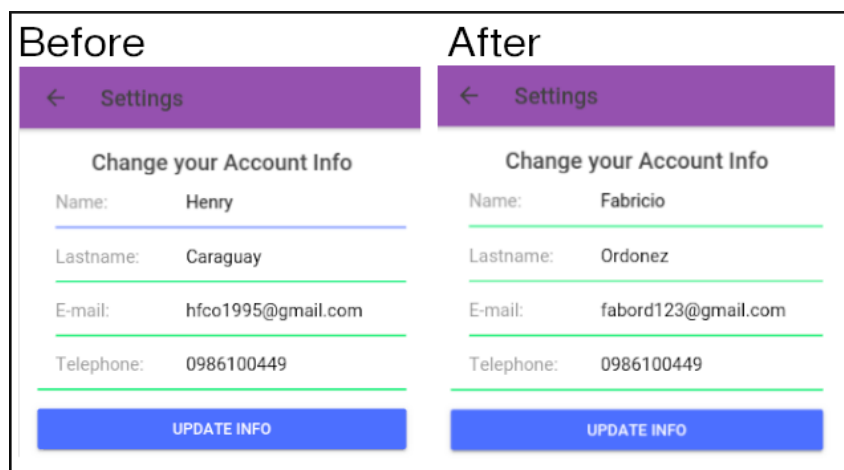


Figure 4.14: Entering our new account information.

The Figure 4.15 shows that the information in the database has been successfully changed, from the former information to the new one. Also, the hashed password has been updated accordingly to the recently entered data.



Before					
id	name	lastname	category	email	password
1	Henry	Caraguay	Student	hfco1995@gmail.com	\$2b\$04\$EBk.BDF1KcVs

After					
id	name	lastname	category	email	password
1	Fabricio	Ordonez	Student	fabord123@gmail.com	\$2b\$04\$AXdgGBw9Ei

Figure 4.15: Database before and after update account information.

### 4.2.6 Add shift, bus and stop

These functions are designed for the driver version of the application. Their functioning is similar, the driver needs to fill some information and confirm the addition, as shown in Figure 4.16. If everything was correct, the application would show a message confirming that the operation was successful. If the database is checked, the changes are confirmed, as shown in Figure 4.17

Figure 4.16: Filling forms for adding a shift, bus or stop.

Add shift						
id	id_driver	id_bus	shift_start	shift_end	rem_cap	
71	2	2	2020-01-10 13:20:00-05	2020-01-10 14:20:00-05	40	

Add bus					
id	school	color	license_plate	is_operative	capacity
56	Math School	#e22525	LBP-163	t	40

Add stop			
id	names	coor_x	coor_y
43	Supermaxi Ibarra	-78.1358814	0.3457668

Figure 4.17: Checking database.

### 4.2.7 Regulations, Help and About

These last three functions are fixed inside the application since its character is purely informative. Then, these functions do not require connection to the back-end, nor to the database. These functions are shown in Figure 4.18.

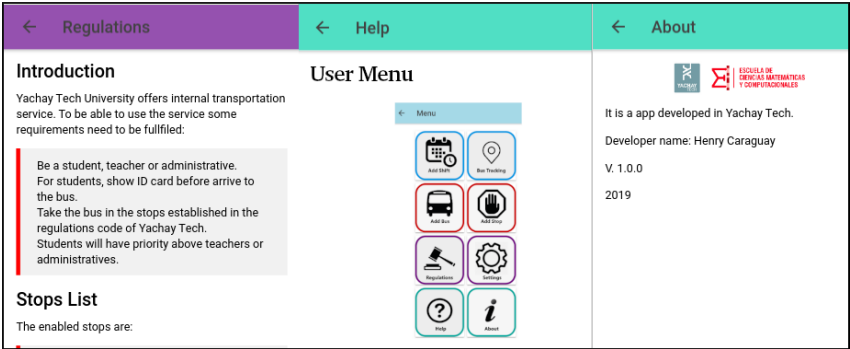


Figure 4.18: Informative functions

# Chapter 5

## Conclusions and Future Work

This chapter gives final insights about the research, what was learned from it, and what could be improved in the future.

### 5.1 Conclusions

- At the end of this work, an application that fulfills the raised requirements exposed in Chapter 1 was built. The application could be tested, and the results were satisfactory. It could be the base for a more complete mobile application that can help to solve the inconveniences that the internal transport service has at the moment of writing this work.
- Through the development of this project, I learned more about mobile applications development, especially about frameworks focused on creating hybrid software, that can be compiled for several platforms. Due to availability issues at the time of testing, I decided to focus on Android.
- Creating the back-end for this application introduced me to servers and their characteristics, how these send and receive data from clients, and how they must be configured in order to retrieve information from a database.
- This project allowed to understand the architecture of a mobile application, including how to perform a deployment, this covers the configuration of the back-end in a cloud platform, so that it gives access through internet to its functionalities.

### 5.2 Future Work

- This thesis covers a first phase of the project, unfortunately because of the complexity of implementing a mandatory system in the transport service and time constraints, it could not be tested in a real environment. In a second phase, one could try to implement the system to check if its use helps to improve the quality of the service,

using variables such as the average waiting time, number of occupied and empty seat, and so on.

- After doing an analysis of the reservation system, we could determine that there might be a need for a third type user, who would be in charge of administrative tasks such as adding buses, drivers, stops, etc. To facilitate his or her work, we could develop a desktop version focused exclusively on these tasks.
- In this first version of the mobile application, the user makes the reservation of the entire route, that is, it is assumed that the user gets on the bus at one of the first three bus stops and gets off at one of the last three, it was done this way to facilitate the implementation of the system at the programming level, but it is not the most optimal. In the future, the mobile application can be improved with a monitoring system, to know more precisely which seats are free and which are not.

# References

- [1] E. Leiseca and R. F. Agüero, “Computerized reservations and scheduling system,” Oct. 12 1993, uS Patent 5,253,165.
- [2] I. Chaparro, *Evaluación del impacto socioeconómico del transporte urbano en la ciudad de Bogotá: El caso del sistema de transporte masivo, Transmilenio*. United Nations Publications, 2002, vol. 48.
- [3] G. Pérez, *Sistemas de cobro electrónico de pasajes en el transporte público*. United Nations Publications, 2002, vol. 45.
- [4] L. B. Lockwood, “Automated sales system,” May 3 1994, uS Patent 5,309,355.
- [5] P. T. Staff. (2018) Travel Technology Solutions. [Online]. Available: <https://www.provab.com/>
- [6] B. Staff. (2019) The web apps you need nothing to install. simply copy and paste to have our web tools on your website! [Online]. Available: <https://www.brolmo.com/>
- [7] F. Beyond. (2019) The Complete Guide to Booking Buses in Vietnam. [Online]. Available: <https://findingbeyond.com/2019/07/24/booking-buses-in-vietnam/>
- [8] InnovaAge. (2011) Apps híbridas vs nativas vs generadas. ¿qué decisión tomar? [Online]. Available: <http://www.innovaportal.com/>
- [9] K. Chavda. (2019) The best mobile app development frameworks for the future. [Online]. Available: <https://dev.to/ketanchavda/the-best-mobile-app-development-frameworks-for-the-future-2dab>
- [10] S. Ahmed and K. Jamsa, “Guide to client server computing,” 2006.
- [11] N. Foundation. (2019) Node.js a javascript runtime. [Online]. Available: <https://www.nodejs.org/en>
- [12] N. Foundation. (2019) Node.js documentation. [Online]. Available: <https://nodejs.org/api/cluster.html>
- [13] Techopedia. (2015) Front-End System. [Online]. Available: <https://www.techopedia.com/definition/3799/front-end-system>

- [14] P. Clareburt. (2010) Development framework what is a development framework? [Online]. Available: <https://it.toolbox.com/blogs/peterclareburt/what-is-a-development-framework-why-use-a-development-framework-042911>
- [15] Opencola. (2016) Development framework advantages and disadvantages of using frameworks. [Online]. Available: <http://www.opencola.com/advantages-and-disadvantages-of-using-frameworks/>
- [16] I. O. Page. (2019) What is Ionic Framework? a brief introduction. [Online]. Available: <https://ionicframework.com/docs/intro#license>
- [17] C. Griffith, *Mobile App Development with Ionic*. O'Reilly Media, Inc, 2017.
- [18] Opencola. (2019) Back End fundamentals the beginner's guide to backend development. [Online]. Available: <https://learntocodewith.me/posts/backend-development/>
- [19] A. Feldthaus and A. Møller, "Checking correctness of typescript interfaces for javascript libraries," *ACM SIGPLAN Notices*, vol. 49, no. 10, pp. 1–16, 2014.
- [20] R. Shannon, "What is html," *Saatavissa: http://www.yourhtmlsource.com/starthere/whatishtml.html [viitattu 21.5. 2014]*, 2007.
- [21] T. Raman, "Cascaded speech style sheets," *Computer networks and ISDN systems*, vol. 29, no. 8-13, pp. 1377–1383, 1997.
- [22] N. Solutions. (2014) 5 Advantages of CSS Web Design. [Online]. Available: <http://www.networksolutions.com/education/css-web-design-advantages/>
- [23] Oracle. (2020) What is a database? [Online]. Available: <https://www.oracle.com/database/what-is-database.html>
- [24] M. Rouse. (2006) Database definition. [Online]. Available: <https://searchsqlserver.techtarget.com/definition/database/>
- [25] R. Obe and L. Hsu, *PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database*. O'Reilly Media, 2017. [Online]. Available: <https://books.google.com.ec/books?id=Xj85DwAAQBAJ>
- [26] A. Irvine and M. McKenzie, "Evaluation of the dsn software methodology," *The Deep Space Network Progress Report*, pp. 42–48, 1978.
- [27] M. Awad, "A comparison between agile and traditional software development methodologies," *University of Western Australia*, p. 30, 2005.
- [28] M. Stoica, M. Mircea, and B. Ghilic-Micu, "Software development: Agile vs. traditional." *Informatica Economica*, vol. 17, no. 4, 2013.

- [29] E. Akpata and K. Riha, “Can extreme programming be used by a lone programmer?” *Systems Integration*, vol. 167, 2004.
- [30] R. Agarwal and D. Umphress, “Extreme programming for a single person team,” pp. 82–87, 01 2008.
- [31] P. Pimienta. (2014) Arquitecturas de aplicaciones moviles. [Online]. Available: [https://https://deideaaapp.org/arquitecturas-de-aplicaciones-moviles/](https://deideaaapp.org/arquitecturas-de-aplicaciones-moviles/)

