Winter 2-2017

iOS application for Urban Traffic Prishtina (U.T. Prishtina)

Blerina Berisha

Computer Science and Engineering

iOS application for Urban Traffic Prishtina (U.T. Prishtina)

Bachelor

Blerina Berisha

February / 2017
Prishtinë

Computer Science and Engineering

Dissertation paper
Academic Year 2012 - 2013

Blerina Berisha

iOS application for Urban Traffic Prishtina (U.T. Prishtina)

Mentor: PhD Cand. Naim Preniqi

February / 2017

This paper has been prepared and submitted in partial fulfillment of the requirements for the Bachelor degree

Declaration

Prishtinë, 15/02/2017

I, Blerina Berisha, student of Computer Science and Engineering at the College of Business and Technology, hereby declare that the following dissertation is written and presented for the purpose of attaining my Bachelor's Degree (BSc). I confirm that this dissertation is my own work, is not copied from any other person's work with the exception of information, definitions and projects for which, each source has been cited and referenced according to the Works Cited regulations. These citations and references can be found under 'References' section. I also confirm that I fully understand what plagiarism is and in any case of plagiarism detected, I understand the consequences of it.

Signature

_____

ABSTRACT

Nowadays, mobile applications have a broad range of utility, be it connecting people via social networks, accessibility applications or location based applications. Being able to receive data through location recognition in your mobile phones offers us opportunities applicable in an even wider range. One opportunity I decided to take advantage of, is to provide a solution to my co-citizen's daily necessity for public transportation information on the go such as bus routes, itineraries and pricing. I intended this by developing an iOS application employing my skills that I learned and developed throughout my study years. U.T. Prishtina has been developed using the latest iOS technologies including the latest programming language by Apple, Swift 3.0.

ACKNOWLEDGMENTS

I would like to use this section to thank my professors at the College for Business and Technology, especially my dissertation mentor PhD Cand. Naim Preniqi who has provided guidance and advice. Furthermore, I thank my friends and colleagues who have contributed with their help and support.

CONTENTS

LIST OF FIGURES

GLOSSARY OF TERMS

U.T. – Urban Traffic

UI – User Interface

API – Application programming interface

URL – Uniform Resource Locator

LLVM – Low Level Virtual Machine

1 INTRODUCTION

Information Technology, as an expanding field everyday, holds a pivotal position in our modern lives. As such, mobile applications that comprise a wide variety of platforms, help us have continuous connection to information and are considered a necessity. Having multiple platforms that host these wide-ranging applications, has made it easier for us to connect through social networks and complete daily tasks with the help of the mobile technology.

In the era of every person owning a mobile device, location-based applications, which provide data tied to a location where activities take place, represent a major number of all the application available in mobile stores. Location-based applications offer service(s) to identify a location and make an action with that information depending on the purpose of the application.

The aim of this thesis is to make use of the location services offered by Apple and Google and implement those services in an iOS application. The main idea of the application is to show information about busses, bus lines and bus map of Prishtina. First, based on the data gathered from Prishtina Buses website [1], the application shows bus information such as a bus line source and destination points. Second, it shows bus stops throughout the line and last, it contains additional data on the timelines. This project targets all the citizens of Prishtina that use the urban traffic services and those that are new to using the public transportation system.

The application is developed using Swift programming language, which was first introduced by Apple and has been out since June, 2014. Moreover, the application is developed for the iOS platform.

Data used in this application is second hand, collected from a web application which is sufficient for presentation purposes and a beta version. Moreover, the thesis will show the tools needed to develop such an application, technology used within the application and the final step which is deployment.

The application is subject to update regarding the information because the Municipality of Prishtina has announced that there will be changes to the public transport within the next year.

2 LITERATURE REVIEW

2.1 iOS

iOS is a mobile operating system developed and distributed by Apple Inc. It has extended support of Apple devices such as iPhone, iPad, iPod, Apple TV and WatchOS. Steve Jobs, chief executive officer of Apple Inc., unveiled iPhone in 2007 and as of February 2017, ten generations of iPhones have been released. With each generation, an iOS operating system has been accompanied. [2]

2.1.1 Platform

iOS operating system is controlled by Apple and as such is carefully crafted and well-designed. On top of that, Apple has given great effort into making its products both designable and usable. Key features of the iOS platform include:

- Consistent interface convections
- Minimalist hardware design
- Human Interface Guidelines
- Cocoa Touch
- Curated and closely controlled app distribution system [3]

There are standard conventions set by Apple which are recommended to be used in applications that implement information-oriented design. Information-oriented design is a development method that focuses on information rather than solely data. The difference between the two is that data is raw which can be any character or text. When it is processed and formatted into something that is understood and meaningful to the person that receives it, it is known as information.

Across different devices Apple stays consistent regarding user interface design. Information-based applications tend to follow the standard conventions while game application usually create custom UI controls.

iOS devices are known for the one-button design and the applications have to handle different functions such as going back or forward to different screens due to the main button in the hardware being used to show the main screen of the iOS device.

Human Interface Guidelines, which are software development documents, provide developers with information that contain recommended design approaches. The aim of these documents is to help create applications that are intuitive and consistent.

Cocoa Touch is a UI framework that provides developers with access to the high level application programming which enables access to the rich set of UI controls defined in Cocoa Touch. Information about key frameworks of Cocoa can be found in chapter 2 section 3.4, page 9.

2.1.2 Programming Languages

The iOS operating system itself is written in C, C++, Objective-C and lately Swift, whereas iOS applications are written either in Objective-C or Swift.

Prior to Swift, iOS applications were developed exclusively in Objective-C. Even though Swift was introduced in June 2014, Objective-C remains a vital and a widely used programming language due to the long-standing usage and development of most existing iOS applications.

For the development of this project, I have selected Swift 3.0, which is the latest version of Swift as of February 2017. The main reason that Swift is used to develop U.T. Prishtina is in regard to my experience gained from previous projects as an iOS developer. While being so, I put my skills and knowledge to use as means of developing the application.

2.1.3 Platform Usage

Considering that the most used mobile operating systems are iOS and Android, which combined capture 98.4% share of the smartphone market, iOS accounts for 11.7% of that share worldwide. However, in the United States, iPhone dominates the market share with 43.3%. [4][5]

iPhone 6 and iPhone 6s are the best-selling devices for iOS and with the launch of iPhone 7 and iPhone 7 Plus, the market share is expected to raise during the year 2017.

2.2 iOS Programming Language

2.2.1 Objective-C

Prior to the introduction of Swift, Objective-C was used as the main language for developing software for iOS and OS X. OS X is the Macintosh operating system that Mac devices run on. [6]

The programming language Objective C was developed in the 1980s which is a superset of the C programming language with added libraries that support object-oriented programming. What is more, Objective-C requires programmers to maintain two code files (header .h and implementation files .m). This poses additional work for programmers which is eliminated in Swift.

Furthermore, Objective-C offers two methods for memory management. The first one is Maintain Retain Release – MRR, through which programmers explicitly manage memory. The second one is Automatic Reference Counting – ARC, which is used in newer projects and automatically calls memory management methods. [7]

Both Objective C and Swift are based on LLVM Compiler Infrastructure and use the same iOS Software Development Kit – SDK.

2.2.2 Swift

Swift has first been introduced in June 2014 by Apple Inc. It is a multi-paradigm language thus it supports more than one programming paradigm. A paradigm is a general approach to build a structure of a computer program. This conveys that programmers using Swift have available opportunities to work in a variety of styles in multi paradigms like Imperative, Declarative, Functional, Object Oriented and more. Different problems require different paradigms to be used.

Imperative programming handles control flow explicitly where, through commands, computations take place step by step. On contrary, declarative programming handles control flow implicitly; programs state what the result should be, not how to get it. Object oriented programming is based around objects that store data and types of operations that allow to manipulate the data. [8]

Swift offers a modern syntax and its objective is to be a high-performance programming language that is memory safe. Memory safe is a technical term that is used when software doesn't run in any memory access errors. Like Objective-C, Swift uses ARC to manage memory, which automatically inserts memory management calls at compile time.

In addition, Swift is based on C and Objective-C and is fully compatible with Objective C. Both languages can be used in one program. xCode, a macOS development environment, builds a *bridging header* which is a file that imports all of the Objective-C symbols. As a result, Swift offers the capability to access code written in Objective-C.

2.3 Tools

2.3.1 xCode

xCode is the official Integrated Development Environment – IDE, which is a software application that provides facilities for software development. xCode is used for developing software for iOS, macOS, watchOS and tvOS.

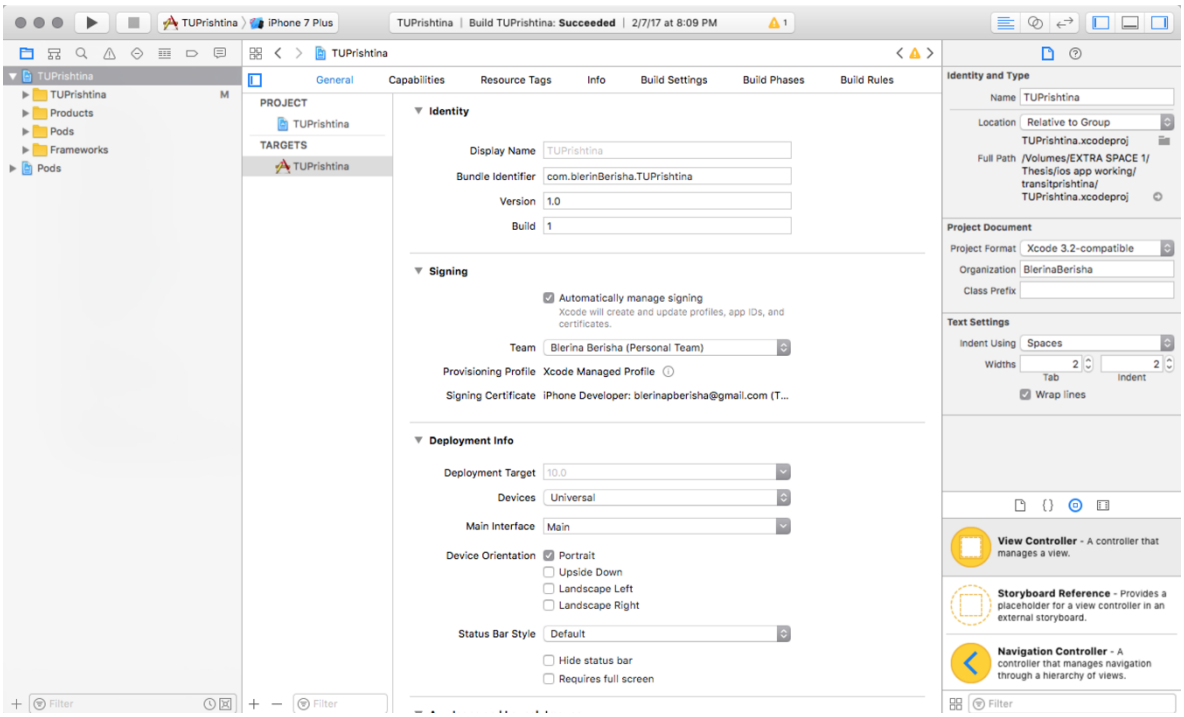


Figure 1. xCode overview

Developers can download xCode free of charge via the Apple Developer website [9]. It includes all the facilities needed for software development such as simulators, professional editor and debugging tools. It builds, installs, runs and debugs any applications that are based on macOS.

Also, it includes Apple's developer documentation (iOS Developer library) which is easily accessible without leaving xCode and allows browsing the documentation provided, for libraries in the SDK.

Installed as a part of xCode, there are built-in devices that can be used for testing and simulating the application, called simulators. Pre-installed simulators of different devices and versions simulate a real device in different environments. For example, in the common case of not having the opportunity to have the all various devices hands-on, simulators enable us virtual environments. This makes early testing significantly easier. However, the simulator doesn't emulate an iPhone processor, disk drive and other specifications unlike an emulator. It uses the Mac's device resources and power. On contrary, an emulator emulates both the software and the hardware; however, xCode doesn't provide iOS emulators. What is more, xCode requires a macOS device in order to run.

2.3.2 iPhone

iPhone is a device designed by Apple, first publicly introduced in 2007. As of February 2017, Apple has released ten generations of iPhones with the last one being iPhone 7 and iPhone 7 Plus. Moreover, iPhone was one of the first smartphones to use a touchscreen interface.

2.3.3 Playgrounds

Swift Playgrounds are part of xCode that can be used for running chunks of Swift code as part of testing that specific part of code. They were launched along side Swift in 2014.

These Playgrounds allow developers to see the result of a piece of code in real time and as such are widely used for testing out algorithms.

2.3.4 Cocoa

Application Programming Interface – API is a set of definitions, protocols and tools for building application software. [10]

API offers functions to the applications accessing them while providing the information requested. The main purpose of APIs is for developers to use certain technologies when building various applications. Given that, Cocoa is the native API of Apple's operating system, macOS; correspondingly Cocoa Touch is the API used in iOS, watchOS and tvOS. Cocoa Touch consists of several frameworks. Key frameworks are the following:

- Foundation Kit Framework
- UIKit Framework
- GameKit Framework
- MapKit Framework
- Store Kit Framework

The Foundation framework provides Objective-C basic classes which implement the root class called NSObject. The prefix NS stands for NeXTSTEP. NEXT was a software company founded by Steve Jacobs while NeXTSTEP was an operating system based in a computer operating system developed in the 1970s named UNIX [12]. NSObject is the root class for most Objective-C class hierarchies where those classes inheriting from NSObject get the ability to behave as Objective-C objects. The foundation framework presents classes with different functionalities. The most important ones are object-oriented storage types such as NSData, NSValue and NSNumber all of which store simple C data values. Moreover, it provides with NSString which stores text strings, NSDate a class that stores time and more storage types. With the introduction of Swift, the prefix NS was omitted and instead of e.g. NSString it is written as String only.

UIKit Framework defines the core components of an iOS application consisting of labels, buttons, table views and navigation. These are crucial components needed to construct

a fully working iOS application. While the Foundation Framework defines classes and functions, UIKit framework is exclusively for iOS, tvOS and watchOS user interface.

GameKit Framework offers features for developing games. MapKit Framework presents classes for embedding maps into iOS applications. StoreKit Framework enables applications to process financial transactions within the app.

3 PROBLEM DEFINITION

Smartphones allow people to improve their daily lives even by just saving small fractions of time and effort. In times of rapid information technology development, going out for a walk and noticing that most of the people own and have in hands a smartphone, shows that the mobile application development is swiftly expanding.

Although the mobile technology has already advanced a lot, the main problem remains, as such the city lacks a mobile application that shows information about the public transportation in the city. The information is available only through a website; however, it is not always convenient to access it or ready on the go.

This project aims to solve this problem by offering an iOS application that is straight forward and very easy to use. It consists of three main views with the first one being a map of Prishtina, showing all lines in different colors. The second view provides a list of all the bus lines which are a click away from the detailed view of the lines. The third view presents a list of information that incorporates the information regarding different traveling schedules each bus line has on weekdays and weekends.

As a significant property of the application is its offline availability which is suitable for people that don't constantly have internet access on their mobile phones. This is possible by saving all data locally on the device.

Predicting that Prishtina is attracting more tourists everyday, the application is very handy for those that don't speak the language and need to get to a specific desired location using the public urban traffic.

4 METHODOLOGY

This research is based on a solely qualitative research. Utilizing a combined set of knowledge and experience gained from different subjects while attending College of Business and Technology and working on many in and out of university projects. Primarily, Software Engineering course provided fundamental background information about software development through which I used different approaches on developing this project. In addition to university courses, my work experience has also aided into putting a lot of new methods which provided adequate optimization of the application.

As an extended research method, I used second-hand data from the Prishtina Buses website which I adapted to my application requirements.

5 RESULTS

5.1 Initial development preparations

In order to start developing iOS applications, a computer with a running macOS is needed with the latest xCode installed. xCode is found in the App Store or in the Apple official downloads website. It includes several built-in templates for developing iOS applications. Different templates are suitable for creating different application views. Included templates are single view, game view, tab-based navigation view and table view. U.T. Prishtina is a tab-based navigation application which consists of three main views. Each view represents the main topics of the application.

To begin, xCode opens a welcome window which pops up a tab that offers to create a new project. After selecting the template, the editor loads the matching view of the template. This project is a tabbed application and as a template a tab-based navigation view is selected. A Tab Bar Controller with a few tab views are loaded ready to be used.

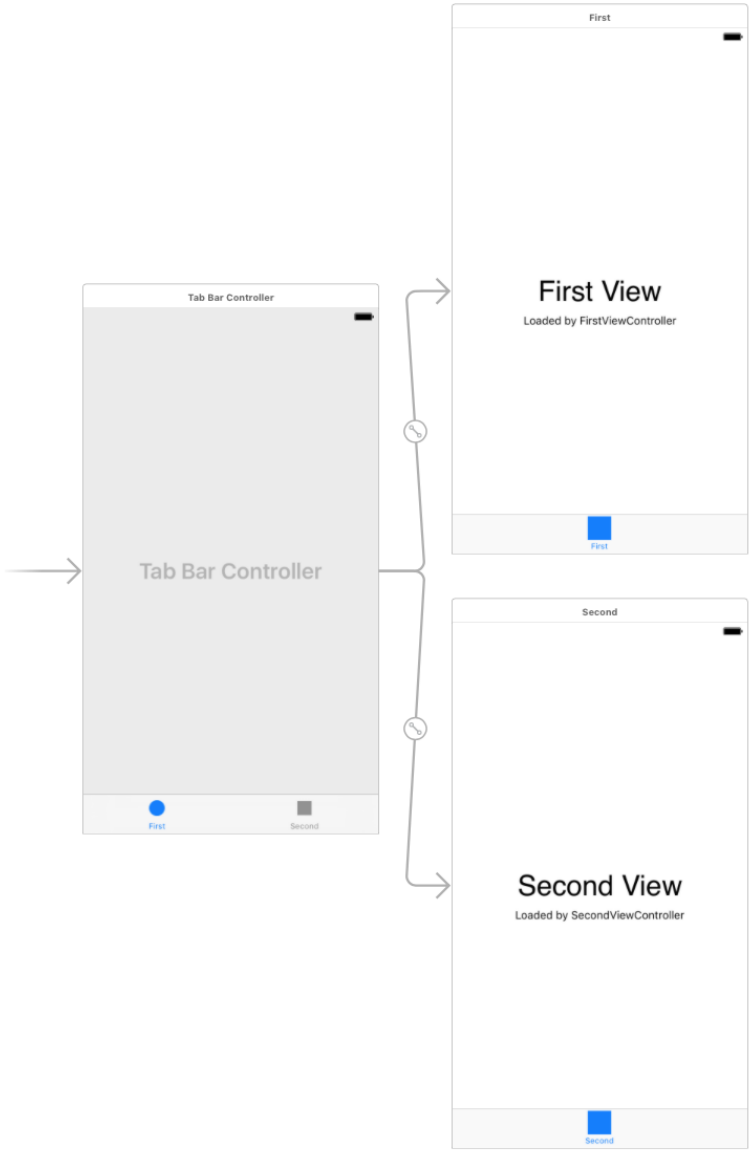


Figure 2. Tab Bar Controller

5.1.1 View Controllers

iOS applications, unlike desktop applications, have limited screen size and are very dynamic. Usually iOS applications show only single displays and to preserve the dynamics it stacks views and swaps them as needed. View controllers are used for managing application's

different interfaces. View controller is an instance of UIViewController class from the UIKit framework that represents the foundation of an iOS application. The UIViewcontroller class defines many methods and properties that help manage the views, events and the transition between one view to another. Every application has one or more view controllers because it manages application's interface and the underlying data.

There are two types of view controllers: content view controllers and container view controllers. Content view controllers are the main type of view controllers that manage the application's content. Container view controller has other child view controllers that manage the presentation or navigation of the content view controllers. In this project, as an initial point, a UITabBarController is used which is a container view controller that contains the main views of the application. Basically, a tab bar controller organizes the application into tabs where each tab is a controller of some kind. Each tab has a name, an optional icon and can communicate with the other tabs in the application that are contained in the tab bar. U.T. Prishtina consists of three tabs.

The first tab is a view controller that contains a map and displays the bus lines. Second and third tab are view controllers that are are embedded into navigation controllers. A navigation controller is an instance of the UINavigationController class. It is a container view controller that is used to manage a stack of view controllers and manage the presentation of those view controllers. In order to transit from one controller to another, the navigation controller helps with the flow using animations. In the second tab, a list of bus lines is shown where with a click another view controller opens containing the details of the requested line. The navigation controller stacks the view controllers and offers a method to go back to the initial view which in this case is a list with bus lines. [13]

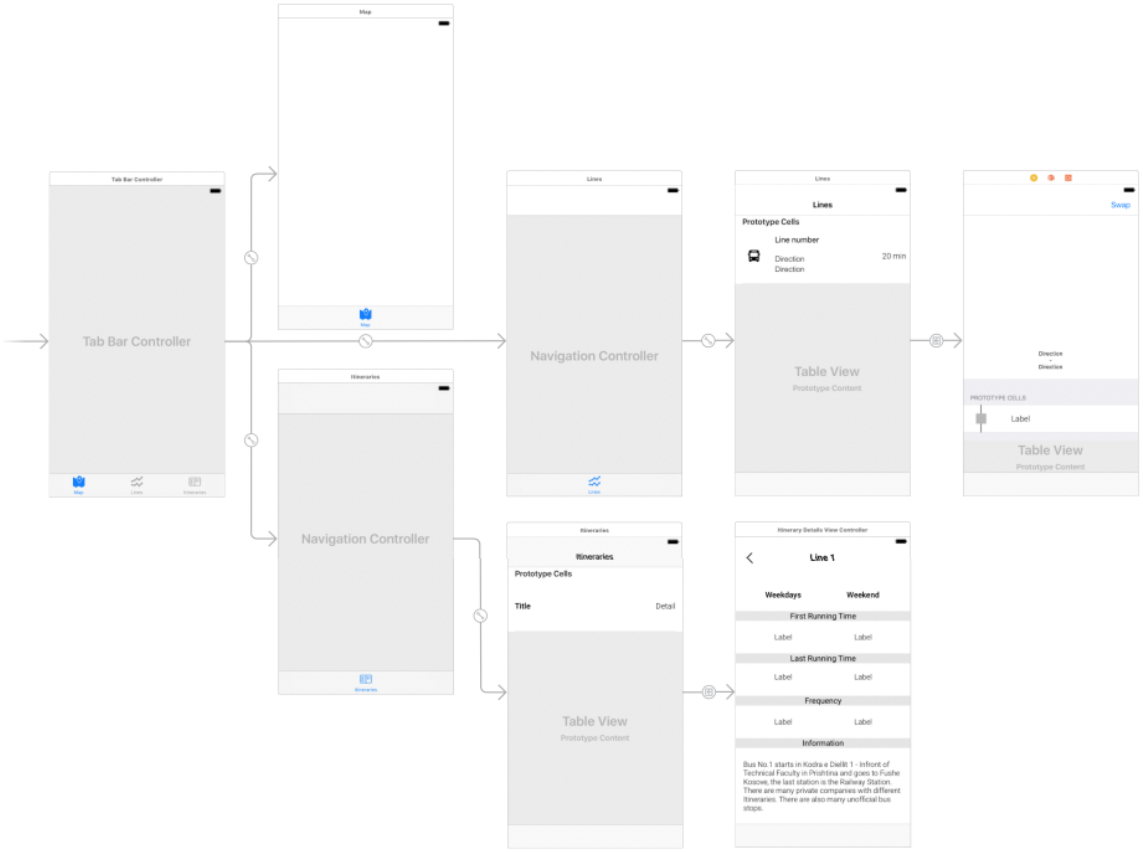


Figure 3. View Controllers overview

5.1.2 View controller life cycle

UIViewController class is the most used view controller in applications. From its creation to its destruction it goes through several states. Different methods are invoked at different states in the lifecycle of the view controller in different orders depending on the state of the view. Lifecycle events are as following:

- loadView – this method is almost never managed manually as it is automatically called when views are created through the interface builder.

- viewDidLoad – it is called once when view controller is loaded in the memory and is used to create any views that will live for the entire lifecycle of the view controller. Views created are not yet visible on the screen.

- viewWillAppear – it is called when the view gets added to the hierarchy and is presented on the screen. It gets called whenever the view re-appears e.g. coming from a back button.

- viewDidAppear – it is called after the view is added to the hierarchy and it is typically used for starting animations.

- viewWillDisappear – whenever the view is about the be removed from the view hierarchy this method gets called.

- viewDidDisappear – it is called when the view is already removed from the hierarchy.

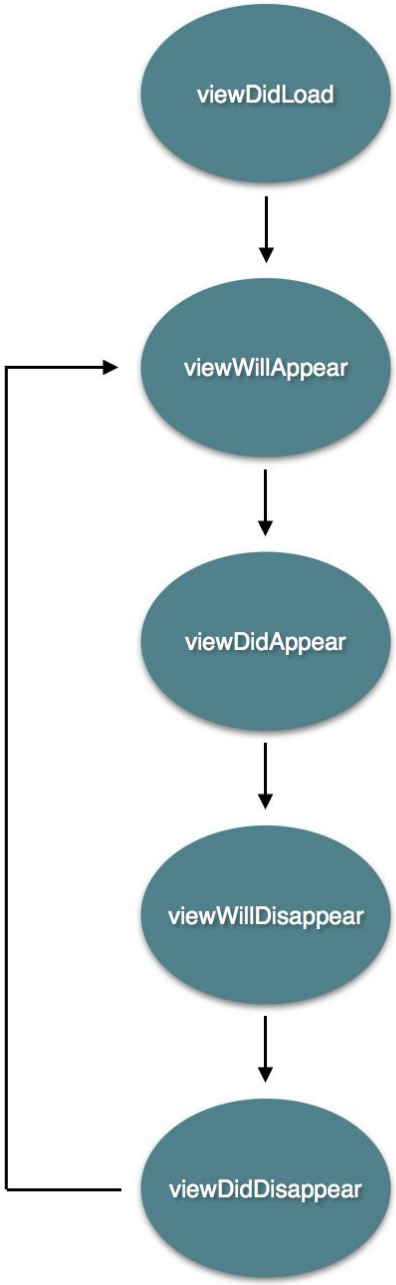


Figure 4. iOS lifecycle

5.1.3 Storyboard

An iOS application is composed of many views which the user can navigate with finger touch control. The relationships with these views are visualized in storyboards. xCode's interface builder offers storyboards that visualize the views and the flow of the application.

Storyboard offers most of the UI elements including all the controllers which are navigation controller, tab bar controller, view controller, table view controller and more. All the UI elements used in an iOS application can easily be utilized by dragging and dropping the elements into the storyboard.

An important feature in xCode is called Auto Layout. Auto Layout is a constraint-based approach that calculates the position of all the views based on constraints set. Constraints represent the relationship between two views and through auto layout the size and position of the views is calculated. Storyboard makes it easy to use Auto Layout offering only control-dragging option between views and configuring the constraints. It also helps with the design of the application because it offers many screen sizes and automatically adjust each element in a view in order to show how it would look in different environments. Adding constraints can be done through control clicking on a view and dragging to the other view.

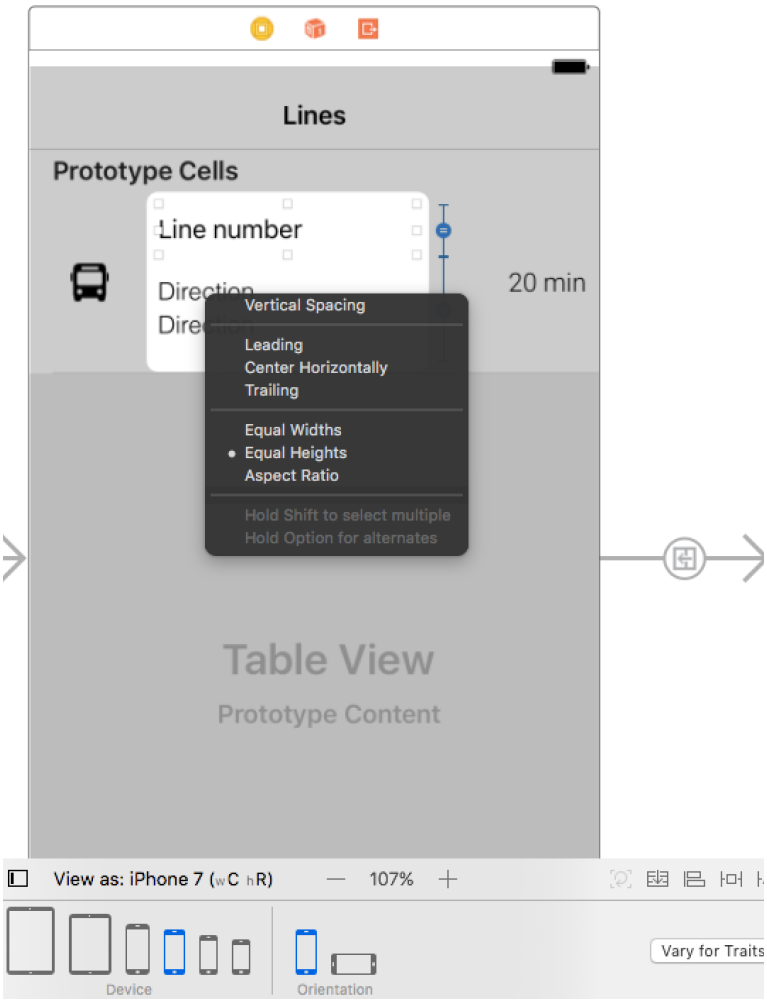


Figure 5. Control-dragging a constraint

5.2 MVC pattern

A design pattern is a template that helps making code easily manageable and offers reusable solutions to problems in software design. Model View Controller is the most-used design pattern in iOS development which separates the application into the following three components: [14]

- Model – models are mostly used for retrieving and string data into the database. It is an object that holds that data.

- View – views display the application's UI that visually represent the model and the components that the user can interact with.
- Controller – controllers are the mediator that coordinate the work between models and views where they handle and process the user's input and interaction which correspondingly updates the model and the view.

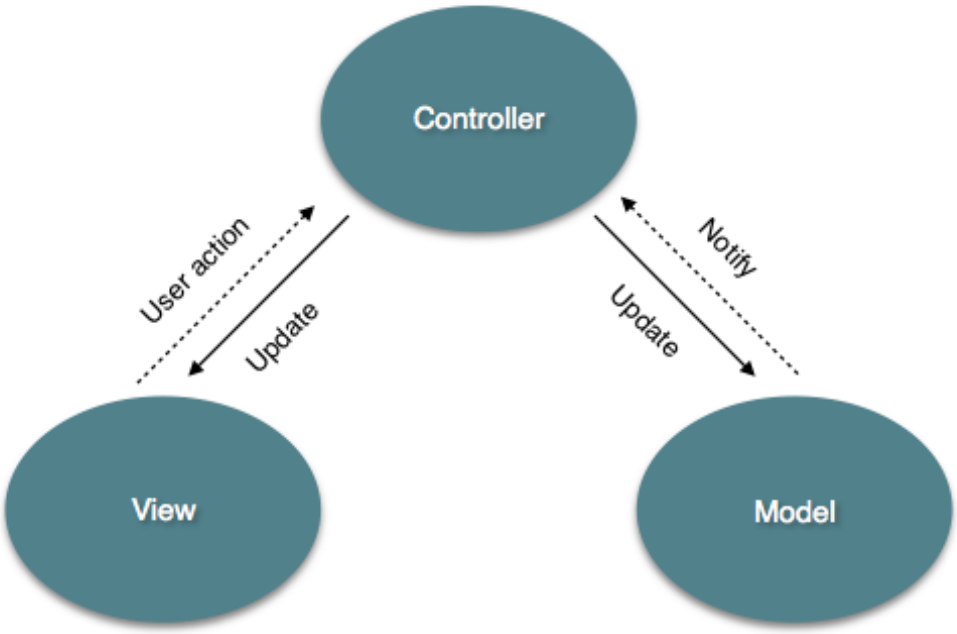


Figure 6. MVC pattern

MVC is the pattern used in this project. Data is retrieved from a JSON file. JSON is a collection of data that is human-readable.

```
{
 "routes": [
        {
        "lines": [
                {
                "name": "Line 1",
                "color": "#e29d81",
                "startLocationLat": 42.6495081925311,
                "startLocattionLng": 21.1660133836129,
                "endLocationLat": 42.6342959385523,
                "endLocationLng": 21.0817190276728,
                "directions": [
                        {
                        "from": "Technical Faculty",
                        "to": "Fushe Kosove",
                        "bus_stops": [
                                {
                                "lat": 42.649649981580964,
                                "lng": 21.16584703872968,
                                "name": "Technical Faculty"
                                },
```

…

All the data is stored in the model classes which are Route, Line, Stop, Direction and Itinerary. Each class retrieves the data from the JSON file and stores it into the corresponding objects. The controller then manipulates with the data and shows it in the matching views in the tabs of the application.

5.3 Cocoapods and Google Maps

U.T. Prishtina is a location-based application that uses maps. Maps used in this project are from Google Maps API which is installed through Cocoapods. Cocoapods is an open-source application manager that has a lot of libraries that can be installed in the project. It is installed using commands in the terminal emulator. Before implementing Google Maps, a Google Maps SDK needs to be installed in the project through Cocoapods. The Google Console offers developers an API key which provides access to the Google Maps, methods and usage statistics.

GMSMapView is the object used to represent maps in the application which is a subclass of the UIView class. A GMSMapView offers necessary functions for managing objects in the map such as polylines and markers. Polyline class is used to draw on the map which is a connected segment on the map. It is an object that contains the latitude and longitude in an ordered sequence. Marker class is used to display a specific location on the map. They can be clicked and offer information about the location.

5.4 Initial View Controller

The starting point in the project is the UITabViewController which holds the main views of the application. The tab bar view controller enables the user to switch between tab views. Each tab has a corresponding view controller.

5.4.1 Map view

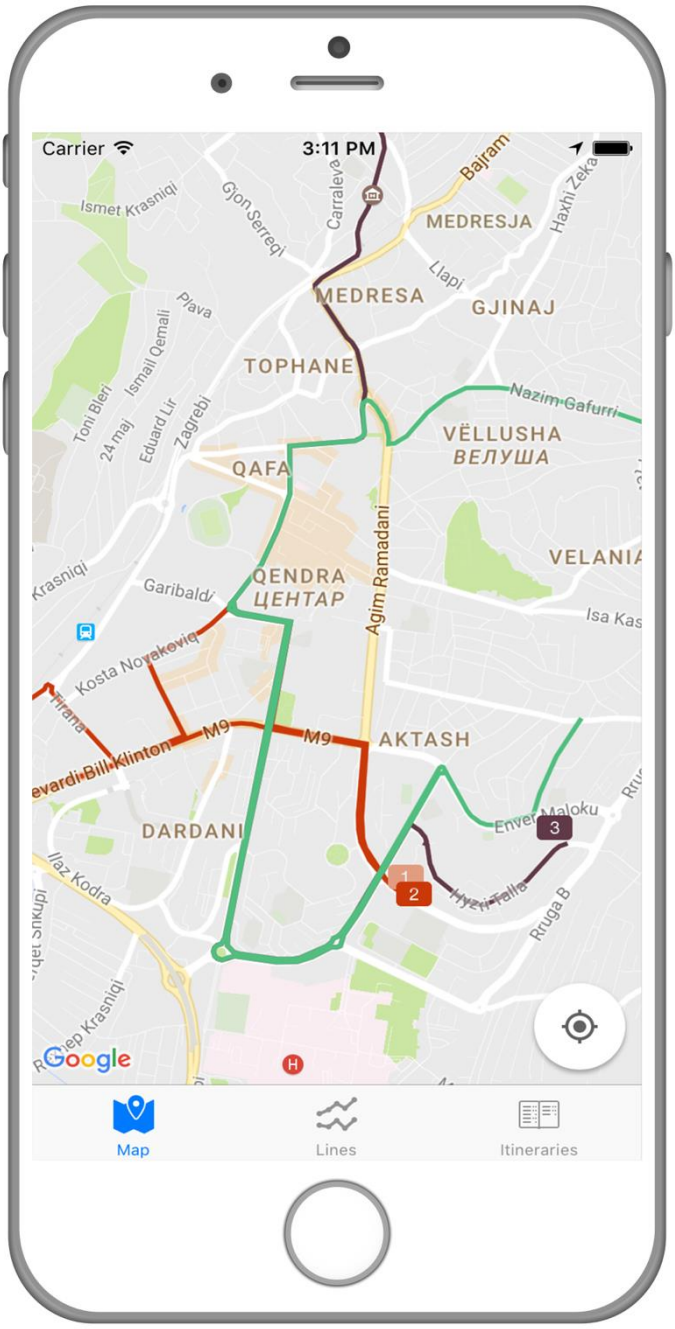


Figure 7. First tab - Map view

MapViewController is the first tab in the tab view controller. Its main purpose is to show an overall view of all the bus lines in different colors.

Google map is saved in a GMSMapView object which is a subclass of UIView. Google map's properties such as camera position and location button are set to Prishtina's latitude and longitude.

Data about bus lines is retrieved through the API engine and saved in the routes object. JSON data is firstly parsed using a third-party library named Unbox [15].

```
BusEngine.instance.request(type: .routes) { (data, error) in
        ...
        let route: Routes = try! Unbox(data:data!)
        ...
}
```

BusEngine is a singleton class that is created with the purpose of making requests in the JSON file. A singleton is a class that allows only one instance of itself to be created throughout the lifetime of the application while it is running. I have used the singleton pattern to create the BusEngine class because the methods about making requests will be used in every tab view hence making it easier and simpler to call by accessing the instance of the engine as shown above.

Bus lines data is retrieved through a for-loop, which allows the code to repeated based on a condition, in this case the number of the bus lines. The route model contains an array that stores all bus lines. Looping the array generates the information about lines such as the line number, polyline, polyline's color, starting and ending coordinates. Polyline in Google Maps is a set of latitude and longitude pairs. In the routes model, polyline is stored as a single string which has all the coordinates encoded using a compression algorithm. To decode the string and get all the coordinates a polyline decoder is used. [16]

5.4.2 Bus lines

LinesViewController is the second tab and it is a UIViewController. It stores a table view that lists all the bus lines in order. In addition, table view is a UITableView that displays a list of items in a single column. The routes object that retrieves all the bus data from the BusEngine is used to create the table's cells. A UITableView needs to have an object, in this case the LinesViewController object, that conforms to the UITableViewDelegate and UITableViewDataSource. The delegate is a class that receives notifications from the table view that provides data and behavior to the LinesViewController. It provides the table view the ability to manage selection of the cells. The UITableViewDataSource's methods provide the table view all the data needed to display the bus information.

In the table view, data about bus lines is shown in the order of the bus line numbers. It displays the line number, direction and frequency.

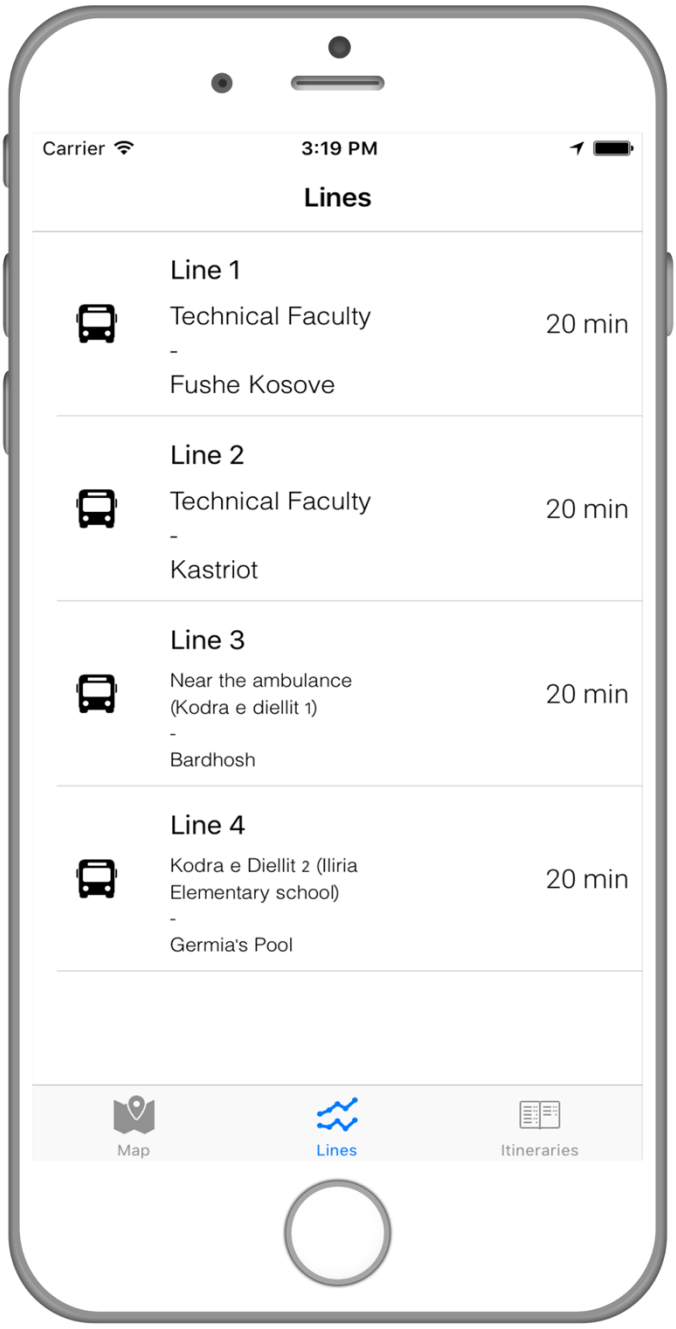


Figure 8. Second tab - Line view

Clicking on a table view cell it transitions from LinesViewController to LineDetailsViewController. This view controller shows details about the specific bus line clicked. At the top it shows a map which has the polyline of the selected bus line. It contains

a table view which scrolls through all the bus stops of the bus line. It has the ability to swap the direction of the bus lines and it adjusts to the corresponding bus polyline and bus stops.

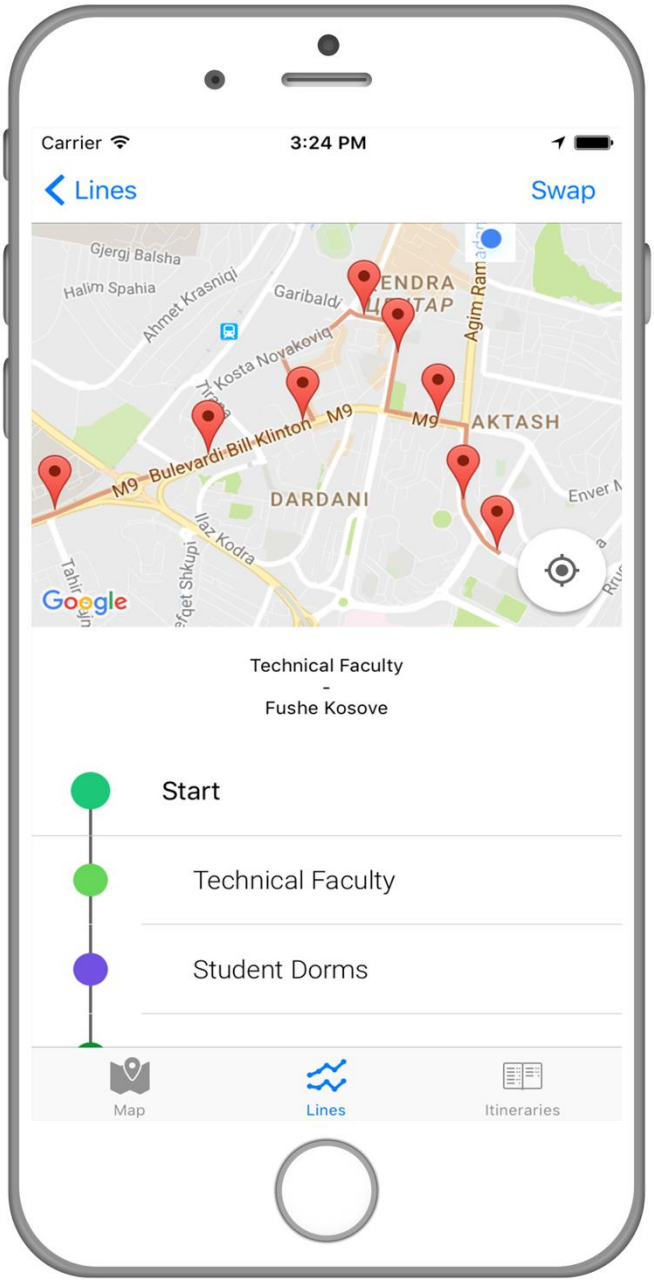


Figure 9. Second tab - Line details view

5.4.3 Itineraries

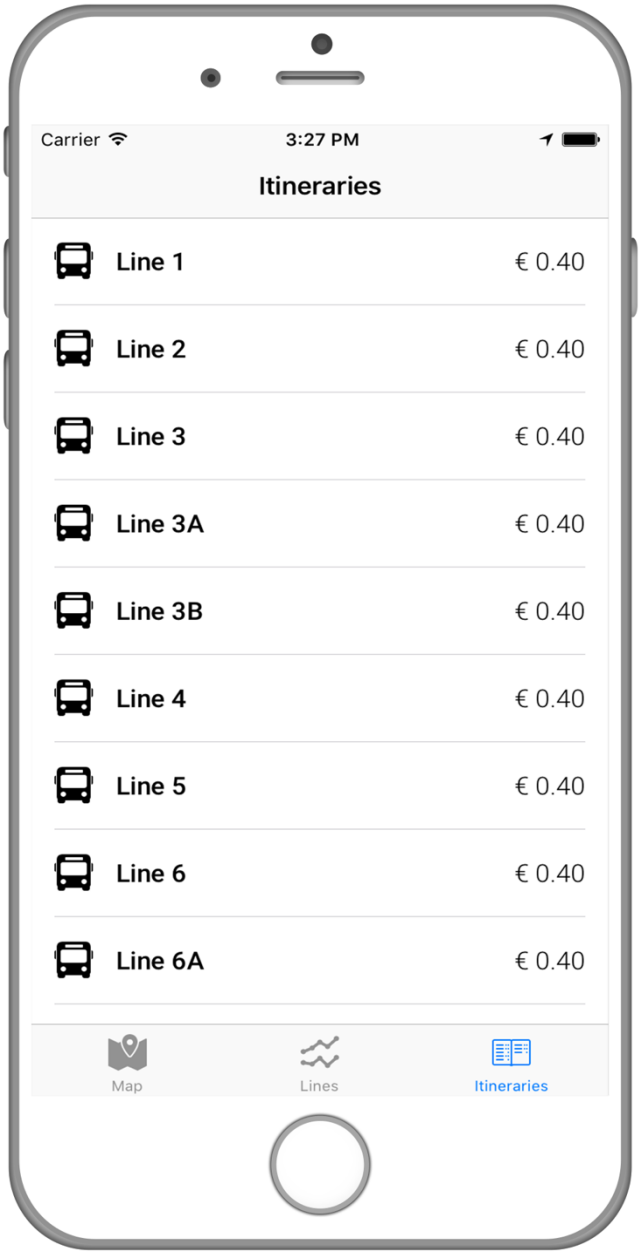


Figure 10. Third tab - Itinerary view

The third tab is a view controller named ItineraryViewController that contains a table view with data about itineraries of the buses. The itineraries object stores the data from the JSON file through the engine API call.

```
BusEngine.instance.request(type: .itineraries) { (data, error) in
        ...
        let itinerary: Itineraries = try! Unbox(data:data!)
        ...
    }
}
```

Each table view cell shows the name of the line and the price of the ticket. Clicking the cell, it transitions to ItineraryDetailsViewController. This view controller contains details about the bus line such as the first and last running times in weekdays and weekends, frequency of the bus line and general information about the bus line.

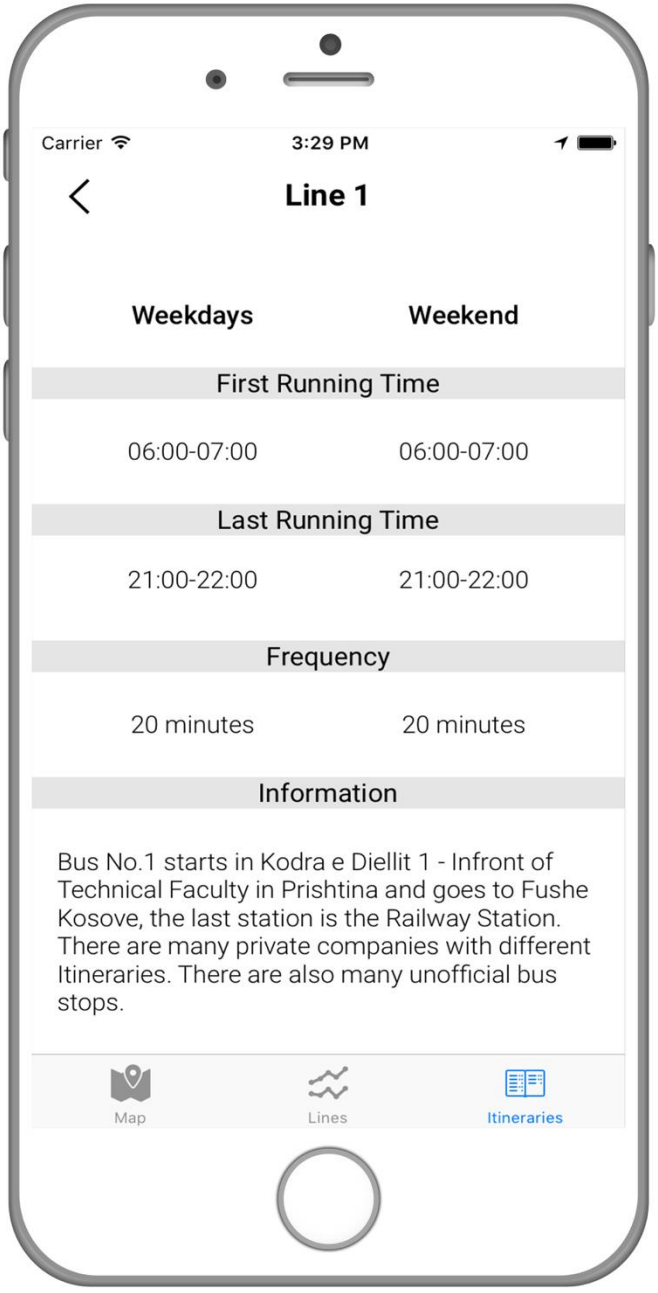
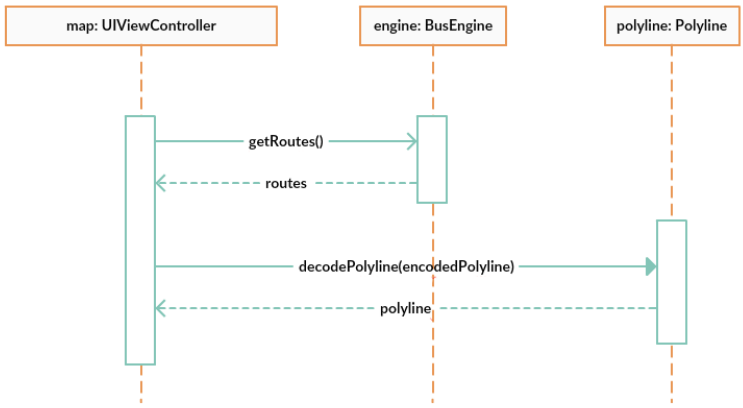
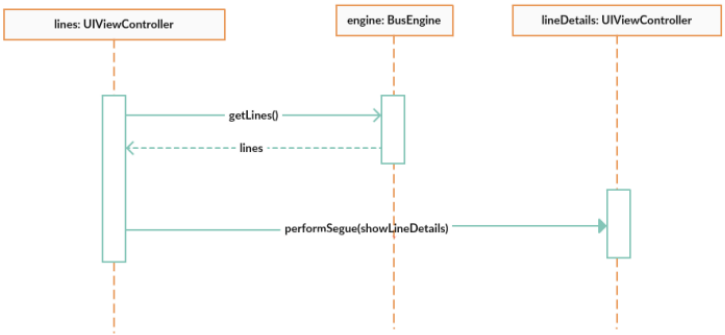


Figure 11. Third tab - Line details view

5.5 Sequence UML Diagram





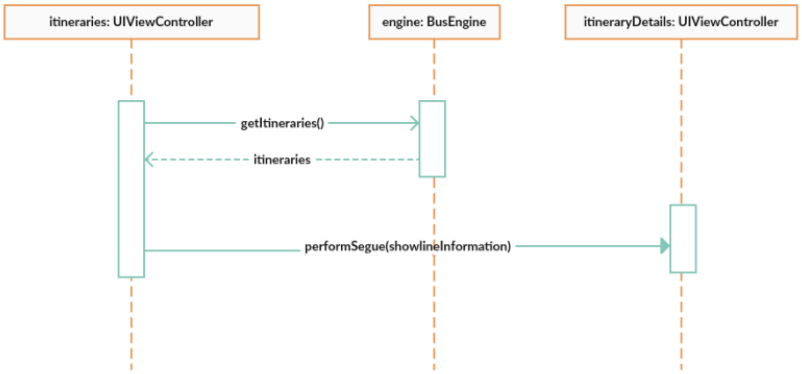


Figure 12. Sequence UML Diagram

5.6 Class and method descriptions

The following classes and methods are used in the project:

Models

- Routes
- Route
- Line
- Stop
- Direction
- Itinerary

Enum

- BusRequestType – Is used in the BusEngine class to differentiate between request. Enum cases are routes and itineraries.

Engine

- BusEngine – A singleton class that handles requests
    - request(type: BusRequestType, completionHandler: @escaping (_ data: Data?, _ error: Bool) -> Void) - based on the bus request type it creates the request with the appropriate file path.

- createRequest(path: String) -> URL? - accepts the file path and returns the file URL referencing the directory.

- createRoutes(_ request: URL, completionHandler: @escaping (_: Data?, _ error: Bool) -> Void) – retrieves the routes from the file and returns the object.

- createItirenaries(_ request: URL, completionHandler: @escaping (_: Data?, _ error: Bool) -> Void) – retrieves the itineraries from the file and returns the object.

View Controllers

- MapViewController – Shows a Google map with the bus lines

  - set(googleMapView: GMSMapView) – configures the Google map's camera position and location.

  - getRoutes() – retrieves all the routes from the engine API.

  - decode(polyline: String, color: String) – decodes the string that contains the encoded polyline.

  - addPolyline(path: GMSMutablePath, color: String) – it adds the polylines in the map.

  - addMarker(lineNumber: String, startCoordinate: CLLocationCoordinate2D, endCoordinate: CLLocationCoordinate2D) – it adds markers at the end of the line that contain the bus line number.

- LinesViewController – Shows a table view with all the bus lines
    - getRoutes() – retrieves all the routes from the engine API.
    - func prepare(for segue: UIStoryboardSegue, sender: Any?) – when a line is clicked, the corresponding information is passed to the destination view which in this case is the LineDetailsViewController.
    - tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int – it accepts the number of routes and makes a respective number of rows.
    - tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell – it creates cells for the table view.
- LineDetailsViewController – Shows the details for the selected bus line
    - setDirectionLabel(index: Int) – shows the starting and ending location names.
    - swapDirection(_ sender: UIBarButtonItem) – it gets called when the swap button is clicked. It swaps the direction of the bus line and updates the map and bus information.
- ItineraryViewController – Shows the bus line itineraries
    - getItineraries() – retrieves all the itineraries for the bus lines from the engine API.

- prepare(for segue: UIStoryboardSegue, sender: Any?) – when an itinerary is clicked in the table view, the corresponding information is passed to the destination view which in this case is the ItineraryDetailsViewController.

- ItineraryDetailsViewController – Shows the details about the itinerary of a bus line

  - getItineraryInformation() – based on the itinerary clicked, it shows the details about the running times, frequency and general information about the bus line.

6 DISCUSSIONS AND CONCLUSIONS

While living in a moderately underdeveloped city there is a variety of challenges that a person faces going about day-to-day activities. Some co-citizens go to universities, some go grocery shopping, a lot of them myself included have a daily job. All of these activities involve transportation, be it public transportation through busses or personal via cars or vans. Although to more developed countries/cities transportation might not be an apparent challenge, to Prishtina it is a significant one considering the majority of the population prefers using the public transportation due to frequent traffic jams within the city. Even though the public one is preferred, this type, unfortunately, considerably lacks service quality and development. Some of the typical problems that are faced on daily basis are:

- The usage of outdated information regarding bus lines
- Absence of clearly marked bus stops
- Inability to track routes, and
- Deficiency in timetable accessibility and availability

Back in 2011, a group of web developers built a website with the attempt to solve to some extent the public transportation issues regarding the fore-mentioned problems. With the intent of respecting their approach to giving a solution, the website has offered data and information virtually for every bus stop in Prishtina. Although the website has had very few updates along the years it has reliably served as a reference point to building and developing a new solution for the public transportation which is prone to regular updates as the municipality makes changes to the system. In addition to using outdated information, from an online-only available source, a set of problems that are daily faced are the absences of clearly marked bus stops. As the bus stops are long-standing, certain conditions have caused for the bus markings to be removed or destroyed, together with them the timetables and route maps. Due to this absence, tracking bus routes is, as a result, very difficult and passengers usually identify the needed stops by their surroundings.

While offering a general and functional result for the mentioned challenges and in contrast to building an online-only accessible platform, U.T. Prishtina being a mobile application, offers a permanent and most importantly offline solution. Furthermore, this application will contain all information regarding bus lines and which routes/streets they take, pricing, bus stop markings, route tracking and timetables that are constantly available, free of charge, for iOS devices for the moment and soon for Android and other mobile platforms.

Ultimately, while keeping in mind that the city of Prishtina has plans for the purchase of new buses and the re-structuring of the entire public transportation map, U.T. Prishtina will have regular updates to conform to the new map/routes and new bus lines.

7 REFERENCES

[1] "Online Map of Bus Lines in Prishtina." Prishtina Buses. N.p., n.d. Web. 01 Feb. 2017. <http://prishtinabuses.info/>.

[2] Rawlinson, Nik. "Apple Turns 40 Today - Here's Its History from 1 April 1976 to 2016." Macworld UK. N.p., 01 Apr. 2016. Web. 18 Oct. 2016.

[3] Whinnery, Kevin. "IOS Platform Overview." Appcelerator. N.p., 04 Dec. 2014. Web. 10 Oct. 2016.

[4] Rob Van Der Meulen. "Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015." Gartner. Gartner, 18 Feb. 2016. Web. 20 Oct. 2016.

[5] "ComScore Reports January 2016 U.S. Smartphone Subscriber Market Share." ComScore, Inc. N.p., 03 Mar. 2016. Web. 22 Oct. 2016.

[6] Rouse, Margaret. "What Is OS X? - Definition from WhatIs.com." WhatIs.com. N.p., Aug. 2006. Web. 10 Jan. 2017.

[7] "Advanced Memory Management Programming Guide." About Memory Management. Apple, 17 July 2012. Web. 01 Dec. 2016.

[8] "Programming Paradigms." Paradigms. Loyola Marymount University, n.d. Web. 20 Dec. 2016.

[9], [11] <https://developer.apple.com/>.

[10] Proffitt, Brian. " What APIs Are And Why They're Important - ReadWrite." ReadWrite. N.p., 19 Sept. 2013. Web. 28 Nov. 2016.

[12] Ford, Kevin. "The Best of NeXT Computers." What's with All the NeXT Names? N.p., 2008. Web. 5 Jan. 2017.

[13] Matt Neuburg. "Chapter 19. View Controllers." Programming IOS 6. N.p., n.d. Web. 03 Jan. 2017.

[14] "Cocoa Core Competencies." Model-View-Controller. Apple Inc, 21 Oct. 2015. Web. 29 Dec. 2016.

[15] JohnSundell. "JohnSundell/Unbox." GitHub. N.p., 01 Dec. 2016. Web. 20 Jan. 2017. <https://github.com/JohnSundell/Unbox>.

[16] Mor, Raphaël, and Tom Tayler. "Raphaelmor/Polyline." GitHub. N.p., 21 Sept. 2016. Web. 30 Jan. 2017. <https://github.com/raphaelmor/Polyline>.