PEOPLE’S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF MOHAMED SEDDIK BEN YAHIA JIJEL

FACULTY OF EXACT SCIENCES AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

DISSERTATION

IN VIEW OF OBTAINING THE MASTER'S DEGREE IN COMPUTER SCIENCE

SPECIALITY : NETWORK & SECURTY

Design and development of a cab booking android application

Submitted by :

Sellemna Abdellatif

Supervised by :

Melit Leila

– Session 2020 –

* *Acknowledgements**

Above all, I thank my Almighty God, who gave me strength, faith, health, will, and guidance to accomplish this humble work.

I wish to express my sincere appreciation to my supervisor, Mrs Melit Leila, who has the substance of a genius: she convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without her persistent help, the goal of this project would not have been realized.

Abstract

Online taxi booking is nowadays a considerable asset for people who no longer want to waste their time waiting for public transport to be available, especially at times of congestion. It is also an advantage for people who wish to benefit from the use of cabs at any time (24/7). In this context, our work consists of designing and implementing an Android mobile application for geolocation and cab reservation intended for both customers and cab drivers. We enter the information about our trip, and the driver moves to the exact place where we are. Also, thanks to our application, we can see the cab's position on a geographical map that displays its movement in real-time. So there is no need to wait outside or watch through the window. The client can see the location of the cab on the map.   Besides, he will be notified when the taxi arrives. At the end of the trip, the application offers the customer the possibility to note the cab driver. To realize our application, we used Android Studio, Android SDK, Flutter, and some programming languages such as Dart and JSON.

Keyword: Cab booking, Mobile application, Android, Flutter, Dart, JSON, Firebase.

Résumé

La réservation de taxi en ligne présente de nos jours un atout considérable pour les personnes ne voulant plus gaspiller leur temps en attendant la disponibilité du transport public, surtout aux moments des encombrements, c'est aussi un avantage pour les personnes qui veulent bénéficier de l'utilisation des taxis à tout moment (24h/jour 7j/semaine). Dans ce contexte, notre travail consiste à concevoir et réaliser une application mobile android de géolocalisation et de réservation de taxi destinée, à la fois, aux clients et aux chauffeurs de taxis. On rentre les informations de notre trajet et le chauffeur se déplace à l'endroit exact où on se trouve. En plus, grâce à notre application, nous pouvons voir la position du taxi sur une carte géographique qui affiche son déplacement en temps réel. Alors pas besoin d'aller attendre dehors ou surveiller par la fenêtre, le client voit clairement l'emplacement du taxi sur la carte.   En outre, il sera notifié quand le taxi arrivera chez lui. A la fin du trajet, l'application offre au client la possibilité de noter le taxieur. Pour réaliser notre application nous avons utilisé Android Studio, Android SDK, Flutter et quelques langages de programmation tel que Dart ,JSON.

Mot-clé : Réservation de taxi, Application mobile, Android, Flutter, Dart, JSON, Firebase.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Since the Internet appeared, the world's way of thinking and living has been changed. Allowed consumers to make transactions and perform their tasks without having to move around physically. About ten years later, this innovation was followed by the appearance of mobile technology, which took an essential place in our society, PDAs, cell phones, smartphones, tablets etc. The means of connection, such as wireless networks (Wifi, GPRS, and others), have made it possible to track and access the information we need wherever there is network coverage. This is being done using mobile applications. Nowadays, having a mobile device is necessary for every individual, which has made us pushed and motivated to think about a useful mobile application.

Taxi services play a vital role by providing personalised vehicles in the urban transportation system. The biggest challenge of this system is the mismatch of passenger demand and taxi service. It is difficult for passengers to get a taxi on time and mostly vacant taxis waste too much time for finding a passenger which causes traffic blockage. To overcome this issue a new system is introduced among the common people to efficiently utilize the perfect combination of their smartphones and internet to book a cab.So our job is to design and develop a mobile application under Android for online cab booking. This application is designed for both customers and cab drivers and will be installed on mobile terminals. The functionalities of our system are :

- Instant geolocation for both the client and the cab driver.
- Instant and future trip booking.
- An estimate of fares.
- Rate the cab drivers.

Thesis planning

In addition to the introduction and conclusion, our brief is structured in three chapters. In the following we detail the contents of the different chapters.

- **Chapter 1:** This chapter focuses on mobile applications and their types as well as the different existing mobile operating systems and we have explained some existing cab booking apps.
- **Chapter 2:** This chapter will be focused on the analysis and design of the system by presenting the different diagrams.

- **Chapter 3:** This chapter presents the environment and development tools used for the realization and implementation of our application with screenshots of the proposed system.

This paper ends with a general conclusion by presenting a summary of the context of our work and the perspectives we envision for completing this work.

CHAPTER 1: ANDROID APPLICATIONS

1.1. Introduction

Nowadays, the massive adoption of mobile devices to perform our tasks in life is behind the exponential growth in the use of mobile applications as well as in the number of mobile users. In 2019, users downloaded over 200 billion apps and spent more than $120 billion in app stores worldwide, and will surpass that mark this year.[1]

In this chapter, we will start by introducing mobile applications, their definitions, and features in section 2, then we will talk about the Android platform in section 3 before detailing the Android applications, and then we will discuss cab booking apps with an example in section 4. Section 5 concludes the chapter.

1.2. Mobile applications

Mobile applications arrived in the 1990s. They are linked to developments in Internet and telecommunications, agent technologies [2], wireless networks, and the emergence and popularisation of mobile terminals. Over time, the mobile domain has undergone considerable development, particularly since 28 November 2007 when it became a fully-fledged IT domain [3].

In this section, we will describe the main basic concepts of mobile applications.

1.2.1. Definition

A mobile application or "Apps" is application software developed to be installed on a mobile electronic device, such as a PDA, mobile phone, smartphone, or personal digital player. Such an application may be installed on the device at the time of its design or, if device allows it, downloaded by the user through an online store, such as Google Play or the App Store. [4] [5]

1.2.2. Characteristics of mobile applications

- From the target point of view: mobile applications are referred to as electronic devices such as :

  - Smartphones, tablets, personal digital assistants (PDAs)... etc.

- From the hardware context point of view: to be able to say that a mobile application is successful, it must have some very important criteria which are the following [6] :

  - Lower resources: CPU, RAM, DD, ROM.
  - Resource consumption must be minimal e.g.: Power consumption; Optimization of the application process to ensure efficient use of energy.
  - An adapted Graphical User Interface (GUI): a good display quality, a criterion to seduce the user.

- From a software context point of view: mobile applications can be [7]:

  - Connected applications: it is an application that requires an internet connection for good functioning.
  - Non-connected applications: called independent applications, these are applications that work without the need for an internet or telephone connection, such as a contact list, calculator, calendar, walkman, console...
  - Localized applications: GPS navigation, geo-localized works...

1.2.3. Types of mobile applications

Currently, there are three (3) types of a mobile application that any user may encounter: web application, native application, and hybrid application as shown in the figure 1.1 :

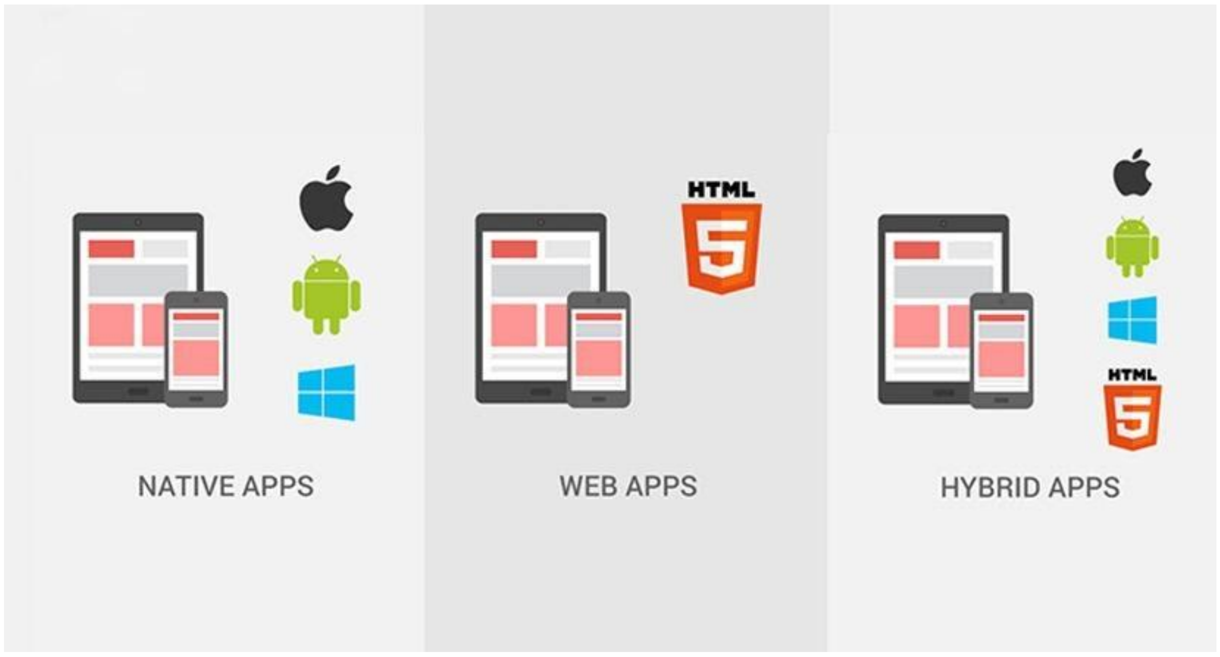


FIGURE 1.1 - Types of mobile applications [8].

1.2.3.1. Native or embedded application: is based on the language of the platform that hosts it. They are therefore programs developed to execute certain functionalities and be deployable in particular mobile platforms or devices.

A native application is therefore a mobile application developed for an operating system used by smartphones and tablets (IOS, Android, Windows phone). It is developed with the language specific to its operating system. Indeed, the development of the native application requires the use of the smartphone's memory without omitting the options related to the targeted operating system (GPS, Camera...) [8][6].

- **Advantages**

  - Better performance through the use of valid software and hardware in the mobile device.
  - Since these applications are already installed in the device and use the data already in existence, they can be run offline without the need for the Internet.
  - Each application is distinguished by its logo to attract the attention of the user.

- **Disadvantages**

  - The major disadvantage of this type of application is that they are strongly related to the device in which they are installed, which will result in the impossibility to evolve or exploit new technologies.

- Making native applications deployable in different types of devices/platforms (i.e. rewriting code in different languages) is a time-consuming task.
- The exclusivity of native applications for a well-designed mobile device type will reduce the number of users and the gain in sales.

1.2.3.2. Mobile web application: these types of applications work like websites, they are mainly developed using web technologies such as HTML5 or CSS3 thanks to HTML5 support and designed specifically for mobile-optimized display [8] Indeed all devices with a web browser can use this type of application. A web application does not take into account the different models of operating systems and brands of smartphones, it is not always ergonomic and moreover it does not use the embedded memory of the smartphone which places it at a disadvantage compared to the native application.

- **Advantages**

- The most important advantage of mobile web applications is the ability to deploy on multiple platforms regardless of device type mobile.
- Cheaper, easy, and quick to develop.
- Easy access using a simple URL without the need for installation or the downloading the supplements.

- **Disadvantages**

- Traditional browsers outperform browsers on mobile devices.
- Mobile web applications cannot exploit the functionality offered by mobile device software and hardware.
- The performance of mobile web applications depends on the speed and state of the Internet connection.

1.2.3.3. Hybrid App or Hybrid App: is the combination of native applications and mobile web applications. The synergy between these two types of applications results in a reduction in time, development effort, price, and maintenance. These applications can be downloaded via online shops, installed on the device, and run from a simple icon like native applications. Hybrid mobile applications are created to run on multiple platforms .[8][6]

1.3. Android

There are different mobile operating systems, in this section, we will focus on the platform we have chosen to develop our application. We start with the definition of the Android platform, its benefits, features, and architecture, and then we detail the basic concepts of Android applications.

1.3.1. Android operating system

The android operating systemis a software platform and operating system developed by Google (2007). Android is based on the Linux kernel in order to be exploited by a wide variety of mobile devices. The popularity of Android is due to the fact that it can be found on a range of devices from different manufacturers including, Samsung, Motorola and HTC, so Android is increasing its technological lead over much cheaper mobiles. [9].

1.3.1.1. Advantages of the Android platform

We chose the Android operating system for the following reasons :

- No license to obtain, no expense for distribution and development.
- To develop location-based applications using GPS.
- Use geographical maps with Google Maps.
- Receive and send SMS, send and receive data on the mobile network.
- Record and playback images, sound, and video.
- Shared data storage tools (SQLite in Sandbox version).
- Hardware acceleration for 2D and 3D.
- Services and applications that can run in the background: that can react during an action, at your position in the city, at the time it is, depending on the identity of the caller.
- A development platform that promotes the reuse of components software and the replacement of the applications provided.

1.3.1.2. Characteristics of Android

There are many Android operating system features. Among these features are [13] :

- **Web browser**: Web browser based on the Webkit rendering engine.
- **Graphics**: 2D graphics library, 3D graphics library based on OpenGL ES 1.0. Hardware acceleration possible.
- **Storage:** SQL database: SQLite is used for data storage.
- **Media**: Android supports the following audio/video/image formats: MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF

- **Connectivity**: gsm, edge, 3G, Bluetooth, wifi.
- **Hardware Support**: Android is able to use a Camera, GPS, accelerometer
- **Development Environment**: Android has a complete development environment containing: an emulator, a debugger, a memory, and a performance analyzer.

1.3.1.3. Android Architecture

The Android Operating System was initially developed by Android Inc. in 2003 and has been modified and optimized by Google since 2005 when they bought the company. Thanks to Google and the Open Handset Alliance (OHA), Android OS was the first open and free mobile Operating System. It has become the most popular smartphone operating system in the world. The Android OS is built over the Linux Kernel and consists of four different layers that perform a variety of tasks that make Android a powerful operating system.[10]

The following diagram (Figure 1.2) illustrates the main components of the system Android operating system. Each section is described in more detail below :

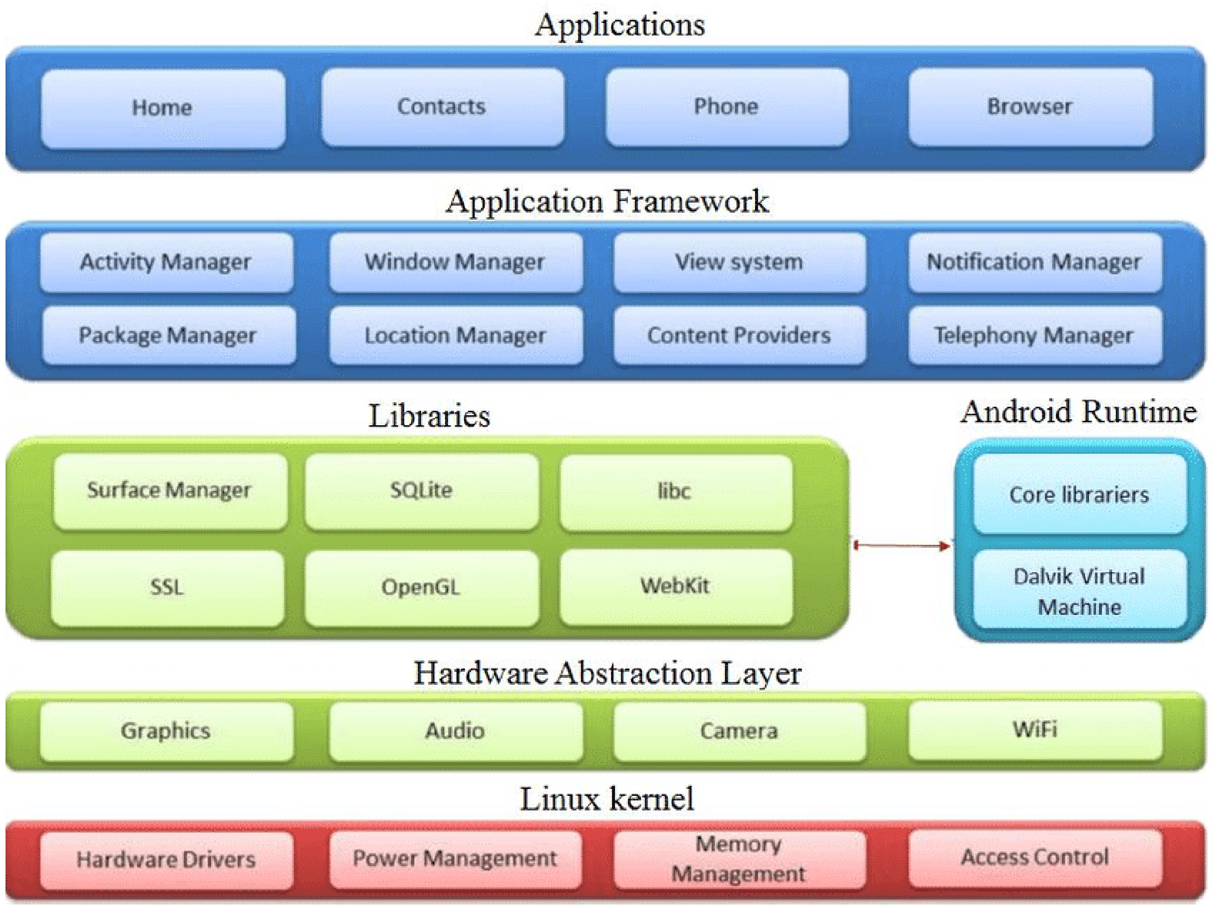


FIGURE 1.2 - Architecture d'Android [11].

a) Linux Kernel

Android is based on a Linux kernel (version 2.6) which manages the system services, such as security, memory and process management, network stack, and drivers. It also acts as an abstraction layer between the hardware and the software stack. [11]

b) Android Runtime Engine

Android includes a set of libraries that provide most of the functionality available in the basic libraries of the Java programming language. Each Android application runs in its own process, with its own Dalvik virtual machine instance. Dalvik VM is an implementation of a virtual machine that has been designed to optimize the multiple executions of virtual machines.

It executes bytecode dedicated to it: the bytecode dex. (Format which is optimized for a minimum memory footprint). This particularity of Android makes it a unique system, far from the traditional Linux systems that many people have encountered before [9][10].

c) Libraries

Internally, Android includes a set of C and C++ libraries used by many components of the Android platform. These libraries are actuallyaccessible to developers through the Android Framework. Indeed, the Android framework makes, internally, calls to C/C++ functionsmuch faster to execute than standard Java methods. The Java Native Interface (JNI) technology allows exchanges between Java code, C, and C++ code [10].

(d) Application Framework

By providing an open development platform, Android gives developers the ability to create extremely rich and innovative applications. Developers are free to take advantage of peripheral hardware, access location information, run background services, set alarms, add status bar notifications, and much, much more. [10] [11]

e) Applications

Android comes with a basic set of programs (also called native applications) to access features such as email, SMS, phone, calendar, photos, maps, web, to name a few. These applications are developed using the Java programming language. For the end-user, this is the only accessible and visible layer. [10][11]

1.3.2. Android application

Android app is a mobile software application developed for use on devices powered by Google's Android platform.

An Android application can be written in several different programming languages.

1.3.2.1. Applications Component

Android apps are organized as a set of components. There are four types of components, and the applications can be composed of one or more of each type. A dynamic instance of one component corresponds to a subset of applications that can be run independently of the others. Thus, many Android applications can be considered as a set of interacting components. The Android application components come in four flavors: [14]

- **Activities**. User-facing components that implement display and input capture.
- **Services**. Background components that operate independently of any user-visible activity.
- **Broadcast receivers**. A component that listens for and responds to system-wide broadcast announcements.
- **Content providers**. Components that make application data accessible to external applications and system components

a) Activities

An activity component implements interactions with the user. Activities are typically designed to manage a single type of user action, and multiple activities are used together to provide complete user interaction. [14][15]

Each activity can start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack. When a new activity starts, it is pushed into the back stack and takes user focus. When the user is done with the current activity and presses the Back button, it is popped from the stack and destroyed and the previous activity resumes. Activities must be declared in the manifest file in order to be accessible to the system. This manifest file presents essential information about the app to the Android system. Information like: minimum SDK (Software Development Kit) version, permissions, activities, services…

i. Activity Lifecycle

Before we deep dive into the lifecycle, we should know that an activity has four states:

State	Descriptions
Active	If an activity is in the foreground of the screen (at the highest position of the topmost stack), it is *active* or *running*. This is usually the activity that the user is currently interacting with.

Paused	If an activity has lost focus but is still presented to the user, it is *visible*. It is possible if a new non-full-sized or transparent activity has focused on the top of your activity, another activity has a higher position in multi-window mode, or the activity itself is not focusable in current windowing mode. Such activity is completely alive (it maintains all state and member information and remains attached to the window manager).
Stopped	If an activity is completely obscured by another activity, it is *stopped* or *hidden*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
Destroyed	The system can drop the activity from memory by either asking it to finish or simply killing its process, making it *destroyed*. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

TABLE 1.1-The different states of an activity [16]

In order to manage the lifecycle of our activity, we need to implement the callback methods. these methods are explained in the table below :

Methode	**description**
onCreate()	Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart()
onRestart()	Called after your activity has been stopped, prior to it being started again. Always followed by onStart()

onStart()	Called when the activity is becoming visible to the user. Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.
onResume()	Called when the activity will start interacting with the user. At this point, your activity is at the top of its activity stack, with user input going to it. Always followed by onPause().
onPause()	Called when the activity loses foreground state, is no longer focusable, or before the transition to stopped/hidden or destroyed state. The activity is still visible to the user, so it's recommended to keep it visually active and continue updating the UI. Implementations of this method must be very quick because the next activity will not be resumed until this method returns. Followed by either onResume() if the activity returns back to the front, or onStop() if it becomes invisible to the user.
onStop()	Called when the activity is no longer visible to the user. This may happen either because a new activity is being started on top, an existing one is being brought in front of this one, or this one is being destroyed. This is typically used to stop animations and refreshing the UI, etc. Followed by either onRestart() if this activity is coming back to interact with the user, or onDestroy() if this activity is going away.
onDestroy()	The final call you receive before your activity is destroyed. This can happen either because the activity is finishing, or because the system is temporarily destroying this instance of the activity to save space.

TABLE 1.2 - Callback methods[16]

Figure 1.3 illustrates the life cycle of activity components with callback methods :

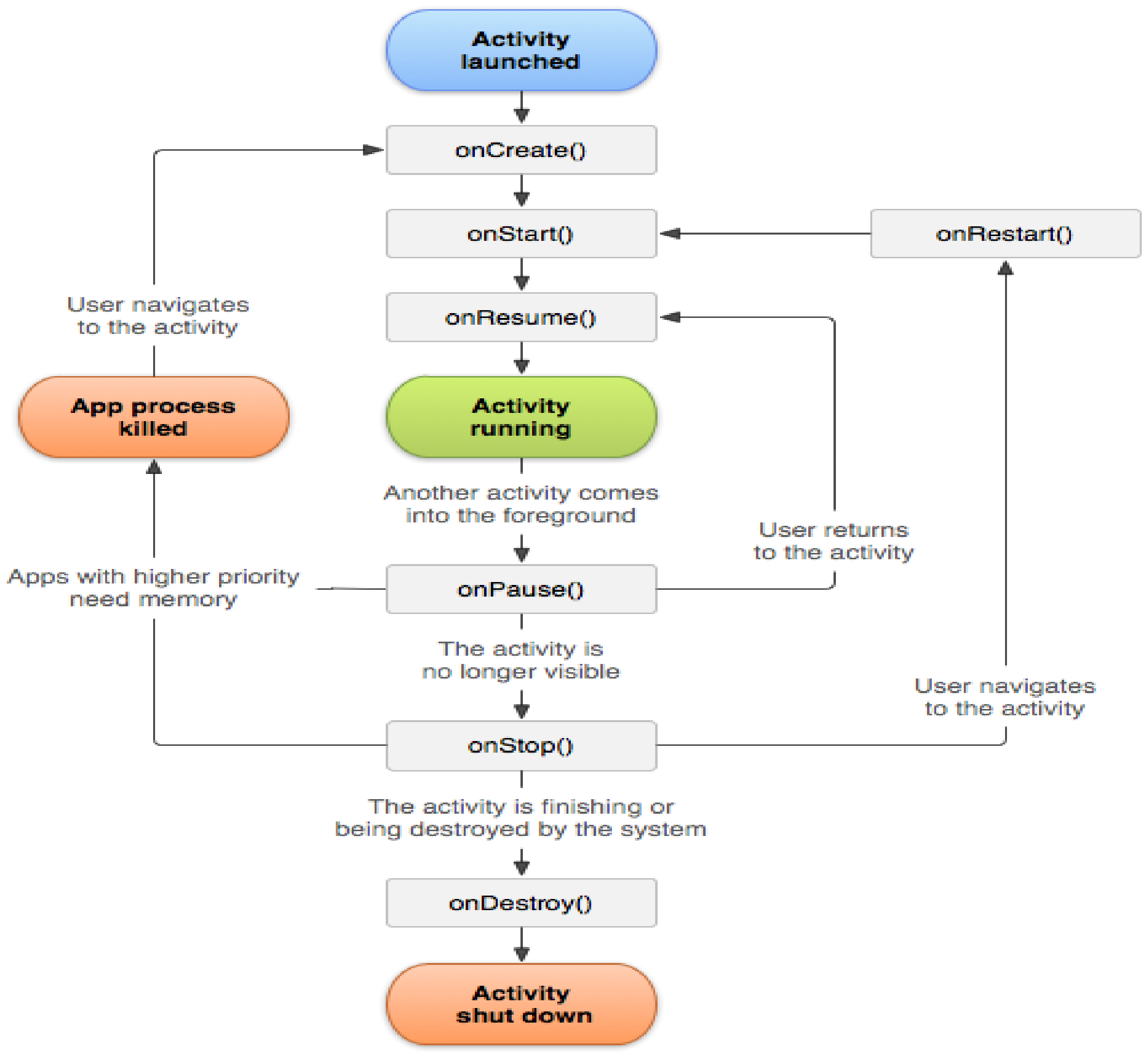

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

FIGURE 1.3- Activity lifecycle[16]

b) Service

Long-running or background components that do not directly interact with the user are expressed as service components. For example, I/O operations that are initiated by activity may not complete before the user-facing activity disappears. In this instance, a service component can be used to carry out the I/O task, independent of the lifetime of the UI elements that initiated it. Services define and expose their own interfaces, which other components bind to in order to make use of the service. As is common with UI elements in GUI environments, services typically launch their own threads in order to allow the main application process thread to make progress and schedule threads associated with other components.[14]

i. Types of Services [16]

The three different types of services are :

- **Foreground:** A foreground service performs some operation that is noticeable to the user. For example, an audio app would use a foreground service to play an audio track. Foreground services must display a Notification. Foreground services continue running even when the user isn't interacting with the app.
- **Background:** A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.
- **Bound:** A service is *bound* when an application component binds to it by calling **bindService():** A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

ii. The lifecycle of a service [12]

The service lifecycle from when it's created to when it's destroyed can follow either of these two paths:

1. **A started service**

   The service is created when another component calls **startService ().** The service then runs indefinitely and must stop itself by calling **stopSelf ().**Another component can also stop the service by calling **stopService ().**When the service is stopped, the system destroys it.

2. **A bound service**

   The service is created when another component (a client) calls **bindService().**The client then communicates with the service through an **IBinder** interface. The client can close the connection by calling **unbindService().**Multiple clients can bind to the same service and when all of them unbind, the system destroys the service. The service does *not* need to stop itself.

Figure (1.4) illustrates the service lifecycle

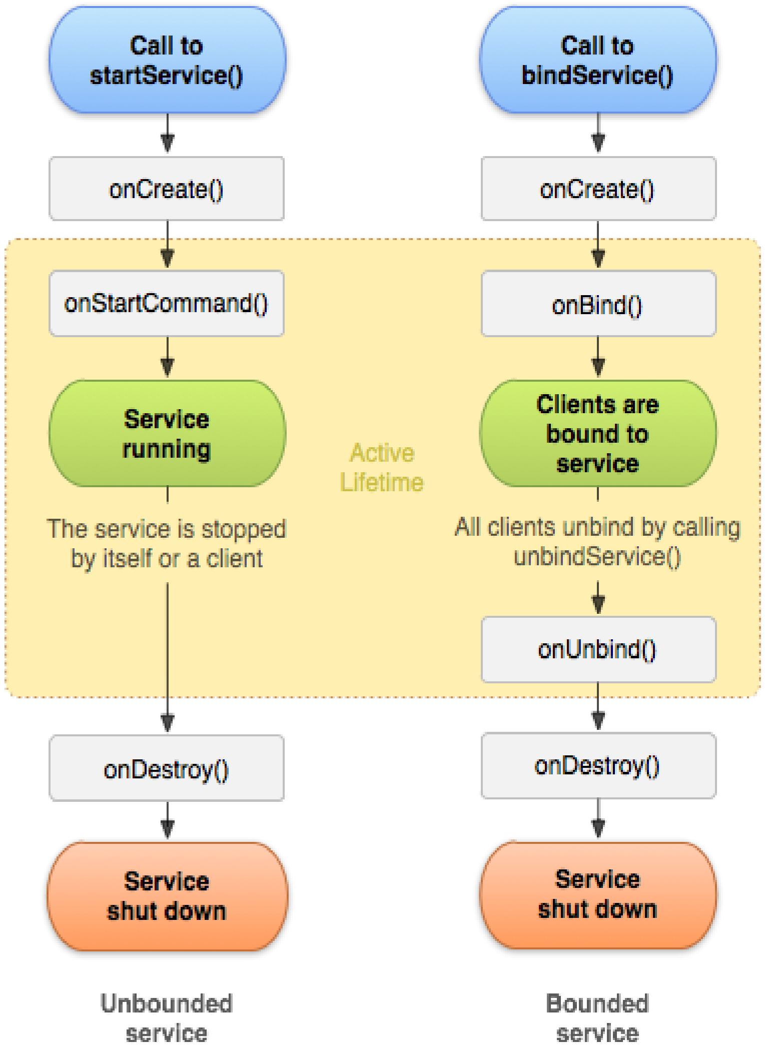


FIGURE 1.4 - Service lifecycle [17]

c) Broadcast receivers

A Broadcast Receiver is an intent-based public subscribe mechanism in Android. This application component allows users to register system events and receive a notification when the registered event is triggered such as SMS notification, battery life, and so on. The receiver is simply a stack of code in the application that becomes activated when a subscribed event is triggered. The system broadcasts events all the time and the broadcasted events can trigger any number of receivers. Broadcasts can be sent from one part of the application to another or to a totally different application. Broadcast Receivers themselves do not have graphical representation, nor do they actively run in memory.

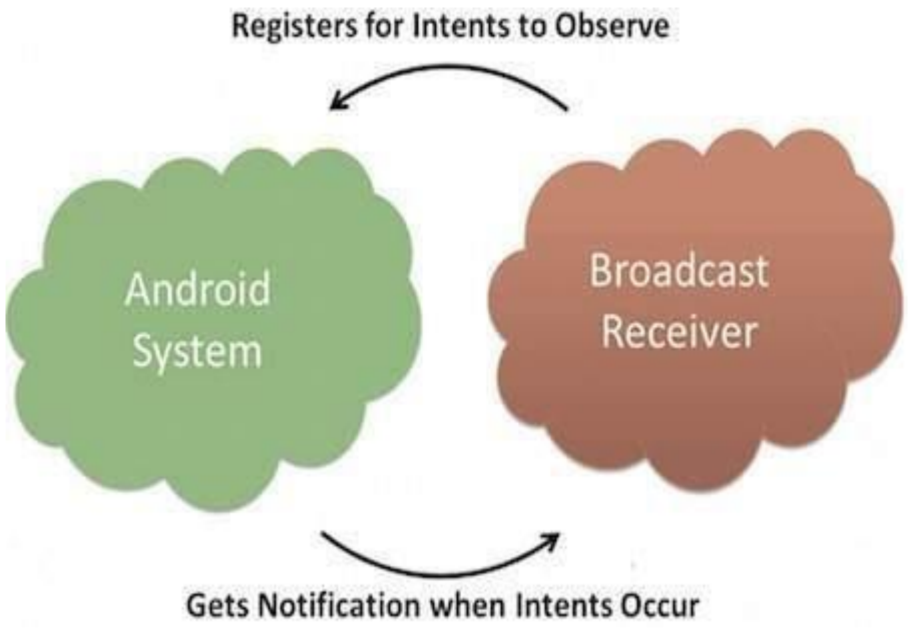

Registers for Intents to Observe
Android System
Broadcast Receiver
Gets Notification when Intents Occur

FIGURE 1.5- Broadcast Receiver [17]

d) Content providers

Components that provide access to an application's data are content providers. Base classes are provided in the Android SDK for both the content provider (that is, the content provider component must extend the base class) and the component seeking access. The content provider is free to store the data in whatever back-end representation it chooses, be it the file system, the SQLite service, or some application-specific representation (including those implemented via remote web services).[14]

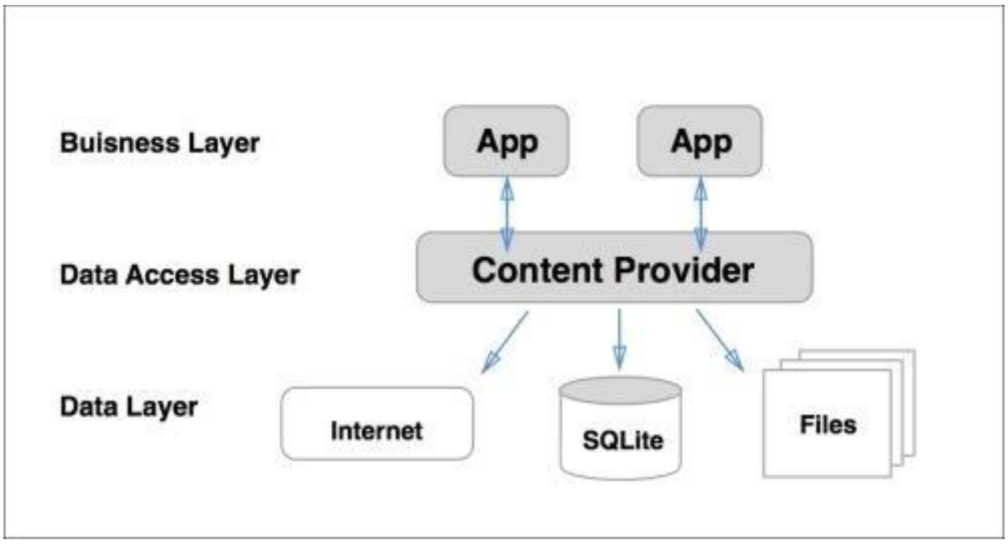


Figure 1.6 – Content providers [14]

1.3.2.2. Additional components

There are additional components which will be used in the construction of the above-mentioned entities, their logic, and wiring between them.

component	Description
Fragments	Represents a portion of the user interface in an Activity.
Views	UI elements that are drawn on-screen including buttons, lists forms, etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Intents are asynchronous messages that name the activity, service, operation, or resource being requested. Intents are a dynamic binding mechanism that enables applications to specify what operations they want to be performed (optionally with some input data), without having to explicitly specify what component will carry out the operation.
Resources	External elements, such as strings, constants, and drawable pictures.
Manifest	Configuration file for the application. The manifest file provides the following information. • **Component description**. The manifest explicitly identifies the activities, services, broadcast receivers, and content providers in the application. For each of these, it names the Java classes that implement each component and describes the Intent messages they can handle. Each component declaration can include a specification of which process should be used to host the component. • **Permissions**. The manifest itemizes the permissions required to execute the application and its components. It also specifies what permissions are required of other applications in order to use this application’s components. Permissions include access to contacts, network I/O, and the file system. • **API version**. The manifest describes the minimum version of Android required to execute the application.

Table1.3- Additional components[18][14]

1.4. Cab booking applications

Since the rise of digital marketing has risen us so up, a lot of people now prefer booking cabs over online applications rather than taking autos or taxis.
Reasons why users prefer online cab booking services :

- Easy to book cabs before the departure time.
- No negotiation with the cab driver about the price of the ride.
- Attractive offers for the users.

We can call an app a cab booking app if it offers some basic functionalities, that are :

- User-friendly UI with location tracking.
- Booking flexibility.
- Estimated time of arrival and distance.
- Fare precision.
- Transparent booking process.
- Cab confirmation.

1.4.1. Some Existing solutions

There are many apps that take place in this market like : uber , lyft ,Grap,Ola,LeCab but we will go with the most efficient :

1.4.1.1. Uber

Uber Technologies Inc. is an American company that develops and offers the Uber mobile application, the application that connects potential passengers with drivers who use their own vehicles. The company was founded in San Francisco in 2009 and started marketing the free mobile application in 2011.[19]

FIGURE 1.7- Uber logo[19]

1.4.1.2. Uber services

Those services are: [19]

- UberT – Potential passengers can hail the official taxi service in that particular town. In New York City, for example, those are "yellow" taxi cabs with a medallion and Boro taxi cabs. Uber charges the application usage and the passenger pays the driver himself.

- UberX – The most famous Uber service. Usually cheaper than the official taxi cabs for 15-20%.

- UberPop – A service that connects potential passengers with unlicensed drivers that have a contract with Uber and have passed their background check. This service is the cause of great controversy which escalated into riots in the city of Paris.

- UberPOOL – Launched in 2014, this is Uber’s most affordable service. It allows ride-sharing with strangers who intend to go the same route. Fare savings can reach up to 40% and if the application cannot find another passenger the sole passenger gets a 10% discount.

- UberMOTO – A low-cost motorcycle transport service launched in February 2016 in Bangkok. Passengers can pay the cab fare in cash or with a credit card.

- UberBlack – The original Uber service which includes luxury vehicles.

- UberSUV – Passenger transport with spacy vehicles.

- UberXL – Passenger transport for large groups.

1.4.1.3. How to use the Uber app [20]

- A passenger opens the app: The passenger enters his destination in the "Where to?" and review each travel option to see the vehicle size, price, and estimated delivery time. Then choose the desired option and confirm the collection.
- A driver is assigned to the passenger: A nearby driver sees the passenger's travel request and decides to accept it. When the driver's vehicle is approximately one minute away, the passenger is automatically notified.
- The driver picks up the passenger: The driver and passenger mutually verify their names and destination. Then the driver begins the journey.
- The driver takes the passenger to the destination: The application offers the driver the option to access step-by-step directions.

- The driver and passenger leave ratings and opinions: At the end of each trip, drivers and passengers can rate each other with a score of 1 to 5 stars. Passengers also have the option to congratulate the driver directly in the app

In the algerian market there are already some existing solutions like Yassir , Temtem and Coursa.

1.5. Conclusion

In this chapter, we have presented in the first part, a small overview of mobile applications. Thus, we have detailed the different features and their types. Then in the second part, we explained the operating system we used in our application. We also described their advantages, their main features, and their architecture, and then we describe the concepts of Android applications. Finally, we made a presentation of some applications of Cab booking under Android that exists.

CHAPTER 2: ANALYSIS AND DESIGN

2.1. Introduction

Transportation is an issue of concern in big cities of many developing countries today. Due to that, we have thought of developing for our final project, an android application for online cab booking. This application is intended for both customers and cab drivers and will be installed on mobile terminals.

In this chapter, we are going to see the design of our application by first presenting the development process used for the realization of our application, and then we will present the UML diagrams and highlight those we have used. We will then present the analysis of our system using the diagrams associated with this step and then the creation of the database.

2.2. Unified Process(UP)

In this section, we will present the development process used, which is essential for any kind of IT project, and for our application, we have chosen the Unified Process (UP).

2.2.1. Definition

A unified process is a software development process built on UML; it is iterative, incremental, architecture-centric, use-case driven, and risk-driven [21]. It is a process pattern that can be adapted to a wide class of software systems, different application domains, different types of companies, different skill levels, and various company sizes.

The purpose of the Unified Process is to guide developers towards the implementation and effective deployment of systems that meet customer needs. [22]

2.2.2. Features of UP

- **UP is iterative and incremental.**

The project is broken down into iterations or short steps that allow for better monitoring of overall progress. At the end of each iteration, an executable part of the final system is produced incrementally (by addition).

Figure 2.1 illustrates the iteration of UP.

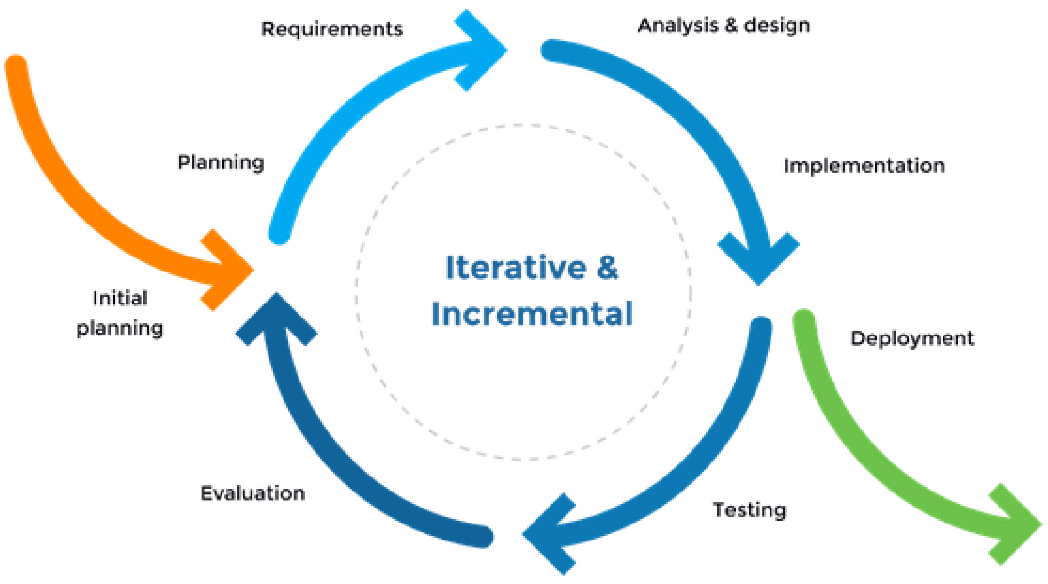


FIGURE 2.1 - The iteration of UP[22]

- **UP is focused on architecture.**

Any complex system must be broken down into modular parts in order to facilitate maintenance and evolution. This architecture (functional, logical, hardware, etc.) must be modeled in UML and not only documented in the text. [22]

- **UP is guided by UML use cases.**

The main purpose of a computer system is to satisfy customer needs. The development process will be accessed on the user. The use case allows us to illustrate these needs. They detect and then describe the functional needs, and together they constitute the use case model that dictates the full functionality of the system. [23]

- **UP is driven by risks**

  The major risks of the project must be identified as soon as possible, but above all, they must be removed as quickly as possible. The measures to be taken within this framework determine the order of iterations.

2.3. Unified Modeling Language (UML)

Before programming the application and starting to write the code, you must organize the ideas, document them, then organize the realization by defining the modules and the stages of the realization. This process, which takes place before writing, is called modeling. Modeling consists of creating a virtual representation of a reality in such a way as to highlight the points of interest. In our project, we used the UML methodology for modeling different diagrams.

2.3.1. Definition

« The Unified Modeling Language (UML) is defined as a graphical and textual modeling language for understanding and describing requirements, specifying and documenting systems, sketching software architectures, designing solutions, and communicating viewpoints. UML is a language with a well-defined syntax and rules that try to achieve writing goals through a graphical representation made of diagrams and textual modeling that enriches the graphical representation. » [24].

2.3.2. List of UML Diagram Types

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment.[25]

These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams. (see figure 2.2).

Structural UML diagrams

- Class diagram.
- Package diagram.
- Object diagram.
- Component diagram.
- Composite structure diagram

- Deployment diagram.

Behavioral UML diagrams

- Activity diagram.
- Sequence diagram.
- Use case diagram.
- State diagram.
- Communication diagram.
- Interaction overview diagram.
- Timing diagram.

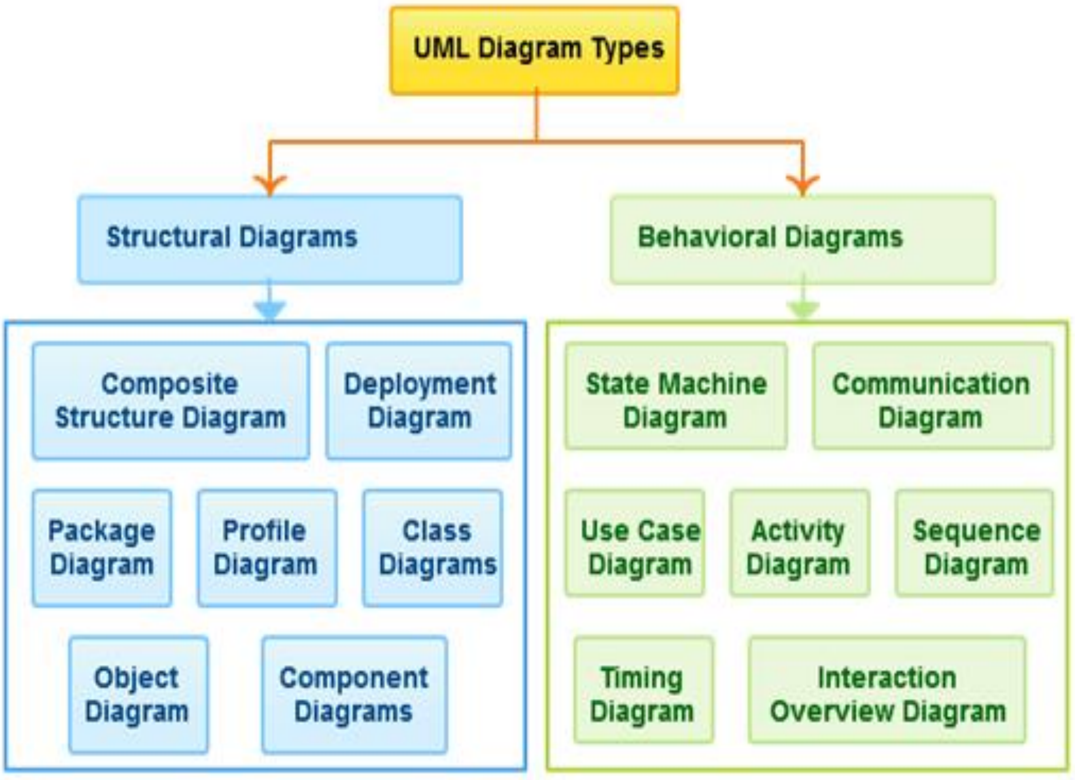


FIGURE 2.2 - UML diagram types[25]

The diagrams we are going to present are the diagrams we used in our application :

2.3.2.1. Use Case Diagram

As the most known diagram type of the behavioral UML types, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors, and how these different functions interact.

It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system. [26]

5.3.2.2. Class Diagram

Class diagrams are the main building block of any object-oriented solution. It shows the classes in the system, attributes, and operations of each class, and the relationship between each class.

In most modeling tools, a class has three parts. Name at the top, attributes in the middle, and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.[26]

5.3.2.3. Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors.[26]

2.4. Requirements Specification

The specification of requirements is the starting phase of any application to be developed in which we will identify the needs of our application. We distinguish between needs functionalities that present the expected functionalities of our application and our customers' needs.

2.4.1. Functional requirements :represent the actions that the system must perform, it becomes operational only if it satisfies them. This application must cover mainly the following functional requirements :

- Location of the user.
- Instant Booking
- Future Booking
- An estimate of time and distance.
- An estimate of the fare.
- Manage Booking requests.
- Review the booking history.
- Registration.
- Rate the driver.

2.4.2. Non-functional requirements: they are needs in terms of performance, type of material, or design type. For this, our future system must meet the following characteristics :

- Provide good UX (user experience).
- Security.

2.5. Requirements analysis

The first step in design is to analyze the situation to take into account the constraints, risks, and any other relevant elements and ensure work or process that meets the needs of the client. In this section, we define the actors of our system and present the use case diagram.

2.5.1. Use case diagram

The representation of a use case involves three concepts: the actor, the use case, and the interaction between the actor and the use case [27]. In this section, we present these three concepts for our approach**.**

2.5.1.1. Identifying Actors

An actor is a person, organization, local process (e.g., system clock), or external system that plays a role in one or more interactions with the system.

The actors within our approach are :

- Passenger.
- Driver

2.5.1.2. Identifying use cases

A use case represents a system functionality. This functionality is accomplished by a user (Actor)[27].

The following table shows the different use cases associated with our system :

Function	**Use case**	**Actor**
Sign up with a phone number	User Authentication	Passenger /Driver
Sign up with a Google account		
Sign in with phone number		

Sign in with a Google account		
View trips history	View trips history	Passenger /Driver
Accept request	Manage booking requests	Driver
Refuse request		
Set availability	Set availability	Driver
Specify the beginning and the end stations	Instant booking	Passenger
Specify the beginning and the end stations	Future booking	Passenger
Indicate time		
Rate Driver	Rate Driver	Passenger

Table 2.1 - The different use cases

2.5.1.3. Illustration of use case diagram

Figure 2.3 illustrates a use case diagram with the actors: the driver and the passenger, as well as use cases.

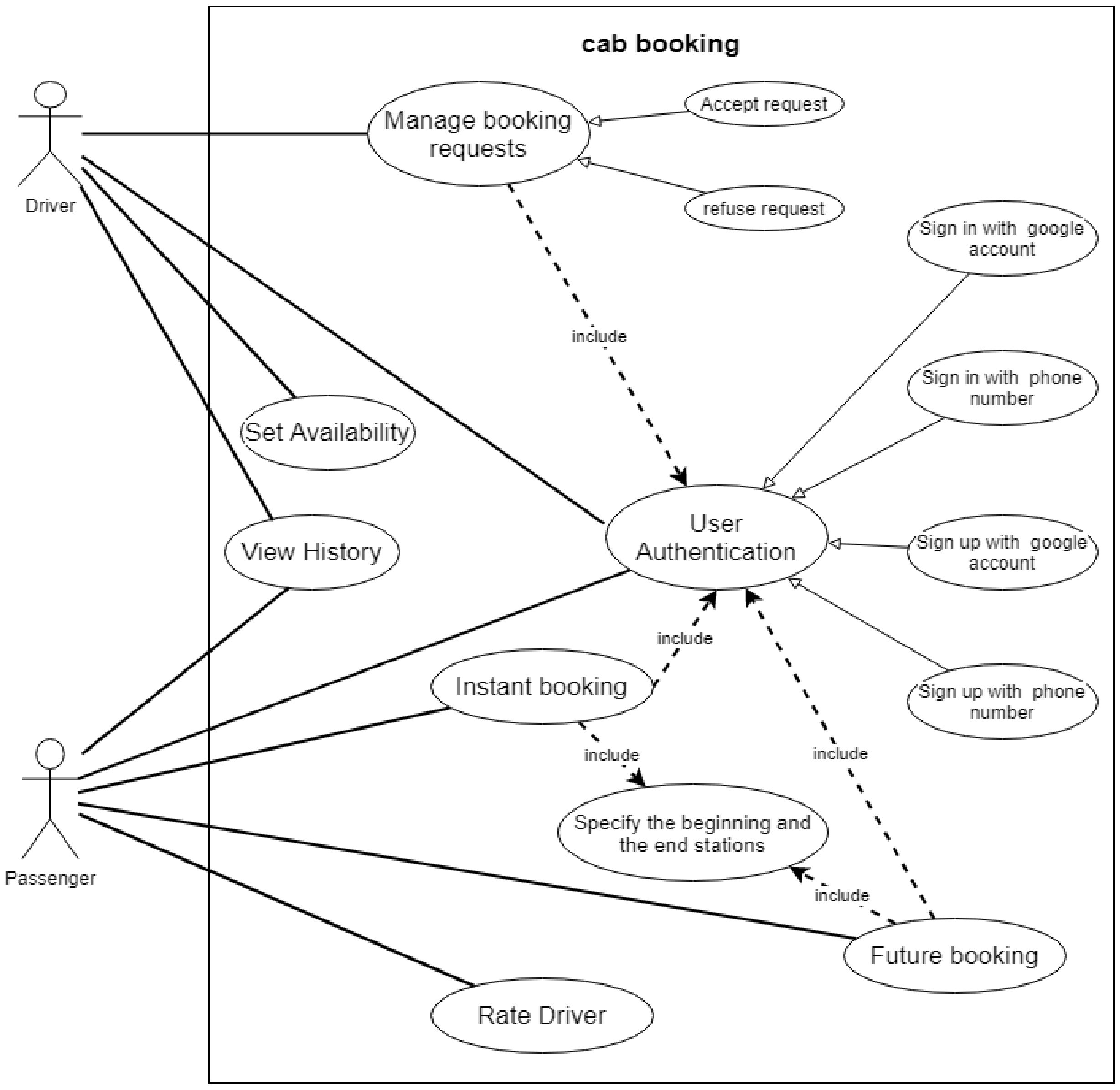


FIGURE 2.3 - Use case diagram.

2.5.1.4.Use Cases Descriptions

Each use case must be associated with a textual description of the interactions between the actor and the system.

The description of a use case is written in six points :

- Objective
- Actor(s)
- *Pre-conditions*: the conditions that must hold for the use case to begin.
- *Post-conditions*: the conditions that must hold once the use case has been completed.
- Main success scenarios(Basic Flow): use case in which nothing goes wrong.
- Alternative paths (Alternative Flow): these paths are a variation on the main theme.

- **Use case <<User Authentication>>**

The following table is a textual description of the use case " **User Authentication** ".

Use case title	**User Authentication**
Objective	Authenticate user
Actor(s)	Driver/ Passenger
Basic Flow	1- launch the app 2- the user choose between sign-in /up with a phone number or with a google account 3-after successful Authentication, the user will be redirected to the home screen.
Alternative Flow	**1-If the user chooses to go with a google account :** -If the user enters an invalid google account, an error message is shown -If an error occurs during parsing information of a google account, an error message is shown. **2-If the user chooses to go with the phone number :** -if the user enters an invalid phone number, an error message is shown If the user enters an invalid OTP code, an error message is shown. **3-If the user login with account that not exist :** if a user enters a not valid account, an error message shaw indicates that the account does not exist.

Table 2.2 - User Authentication's description.

- **Use case << Manage booking requests>>**

The following table is a textual description of the use case " **Manage booking requests** ".

Use case title	**Manage booking requests**
Objective	Manage booking requests that were made by Clients
Actor(s)	Driver
Pre-conditions	The driver must have already opened the app and already authenticated. Booking exists
Basic Flow	1- The driver receives the trip booking information 2- The Driver swipe right to accept the reservation 3- The status of The cab driver becomes automatically unavailable
Alternative Flow	The Driver swipes right to refuse the reservation, so the passenger keeps waiting for another driver to accept
Post-conditions	Booking is confirmed or rejected

Table 2.3 - Manage booking requests' description

- **Use case << Instant booking>>**

The following table is a textual description of the use case " **Instant booking** " :

Use case title	**Instant booking**
Objective	Booking a Driver for an immediate trip
Actor(s)	Passenger
Pre-conditions	The Passenger should be already opened the app and already authenticated.

Basic Flow	1- The system detects the location of the passenger with all nearby driver. 2- The client specifies the beginning and end stations. 3- The system estimates the duration, the distance and the fare of the trip. 4- The client receives a message indicating that the trip has been confirmed. 5-The client will be redirected to the trip screen.
Alternative Flow	If the waiting period runs out, the customer will receive a message that there is no driver who has responded to his request on time.
Post-conditions	The booking is made.

Table 2.4 - Instant booking' description

- **Use case << Future booking>>**

The following table is a textual description of the use case " **Future booking** " :

Use case title	**Future booking**
Objective	Booking a Driver for a future trip.
Actor(s)	Passenger.
Pre-conditions	The passenger must be already authenticated.
Basic Flow	1- The client specifies the beginning and end stations. 2- The system will estimate the duration , the distance and the fare of the trip 3- The client Indicates the date and time. 4- when the date of the trip is equal to the current date, the client receives a notification.
Alternative Flow	If the app was closed or in the background, the client receives a notice to remind him.
Post-conditions	The booking is made.

Table 2.5 - Future booking's description

- **Use case << Rate driver >>**

The following table is a textual description of the use case " **Rate driver** " :

Use case title	**Rate driver**
Objective	Rate the trip’s driver
Actor(s)	Passenger
Pre-conditions	The passenger must be already authenticated.
Basic Flow	1- The client rates the driver by stars

Table 2.6 - Rate driver’s description

- **Use case << Set Availability >>**

The following table is a textual description of the use case " **Set Availability**" :

Use case title	**Set Availability**
Objective	Set the availability of the cab driver
Actor(s)	Driver
Pre-conditions	The driver must be already authenticated.
Basic Flow	1- The driver chooses his status

Table 2.7 – Availability‘s description

- **Use case << View trips history >>**

The following table is a textual description of the use case " **View trips history**" :

Use case title	**View trips history**
Objective	View trips history
Actor(s)	Driver / Passenger
Pre-conditions	The driver / passenger must be already authenticated.
Basic Flow	1- The driver / passenger View his trips history

Table 2.8 -View trips history's description

2.6. Conception

This part will be dedicated to the design of our system, we will use the class diagram to represent the entities manipulated by the users.

2.6.1. Class diagram

2.6.1.1. Data Dictionary

Table 2.7 below represents the data dictionary of the class diagram :

Class	**attribute**	**description**	**Type**
passenger	uid	passenger’s identifier	String
	account	passenger’s account (email/phone)	String
	fullname	passenger’ full name	String
Driver	uid	Driver’s identifier	String
	fullname	Driver’s full name	String
	contact	Drivers’ contact(email/phone)	String
	rate	Driver’s rate	Double
	cab_num	cab number	String
	nrate	Number of the clients who rated the driver	Integer
	available	The availability of the driver	Boolean
	cordinate	The coordinate of the driver(location)	Geopoint
Reservation	id	Reservation’s id	String
	uid_r	passenger’s identifier	String
	uid_d	Driver’s identifier	String
	cost	Reservation’s cost	Integer
	Time	Duration of the trip	String
	distance	Reservation’s distance	Double
	date	Reservation’s date	Datetime
	source	Reservation’s source	String

	destination	Reservation’s destination	String
	rate	The rate that was given for the driver by a client	double
	picked_up	The booking was accepted or not	Boolean

Table 2.9 - Data dictionary

2.6.1.2. Class diagram illustration

Figure 2.4 represents class diagram of our system :

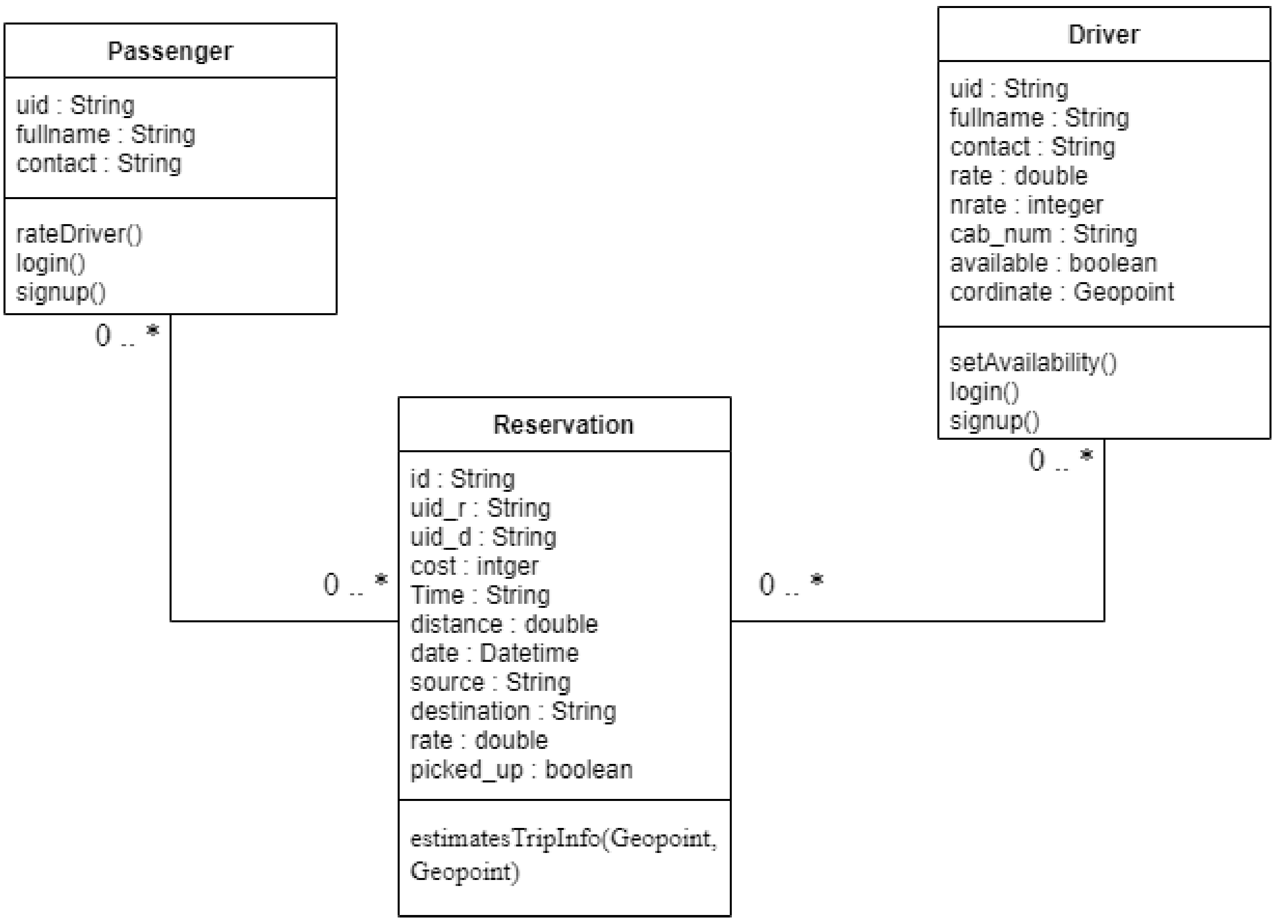


FIGURE2.4 - Class diagram

2.6.2. Creation of a database

Our application is based on distributed systems and manages data hosted on servers that change in real time, so we need to store our data in real time and access it just as quickly, and traditional relational databases cannot provide a satisfactory response time, which is why we opted to use NoSQL databases.

We will present in the following, the main concepts of the NoSQL database.

2.6.2.1. NoSQL

NoSQL databases are those databases that are non-relational, open-source, distributed in nature as well as having high performance in a linear way that is horizontally scalable. A Nonrelational database does not organize its data in related tables (i.e., data is stored in a non-normalized way). NoSQL databases are open-source; therefore, everyone can look into its code freely, update it according to his needs, and compile it. Distributed means data is spread to different machines and is managed by different machines so here it uses the concept of data replication.[28]

2.6.2.2. NoSQL features

NoSQL databases are an excellent fit for many modern applications such as mobile, web, and gaming that require flexible, scalable, high-performance, and highly functional databases to provide great user experiences.[29]

- **Flexibility**: NoSQL databases generally provide flexible schemas that enable faster and more iterative development. The flexible data model makes NoSQL databases ideal for semi-structured and unstructured data.[29]
- **Scalability**: NoSQL databases are generally designed to scale out by using distributed clusters of hardware instead of scaling up by adding expensive and robust servers. Some cloud providers handle these operations behind-the-scenes as a fully managed service.[29]
- **High-performance**: NoSQL database is optimized for specific data models and access patterns that enable higher performance than trying to accomplish similar functionality with relational databases.[29]
- **Highly functional**: NoSQL databases provide highly functional APIs and data types that are purpose-built for each of their respective data models.[29]

2.6.2.3. NoSQL datastore types

NoSQL databases are divided into a number of databases. There are four new different types of data stores in NoSQL :

1. **Key-value databases**: The key-value database name itself states that it is a combination of two things that are key and a value. It is one of the low profile

(traditional) database systems. Key-Value (KV) databases are the mother of all the databases of NoSQL. The Key is a unique identifier to a particular data entry. The key should not be repeated if one used that it is not duplicate in nature. Value is a kind of data that is pointed out by a key.[28]

2. **Wide column/column family**:are NoSQL databases that store data by column instead of saving data by row (as in relational databases). Thus, some rows may not contain part of the columns, offering flexibility in data definition and allowing them to apply data compression algorithms per column. Furthermore, columns that are not often queried together can be distributed across different nodes.[30]

3. **Graph-oriented**: These databases aim to store data in a graph-like structure. Data is represented by arcs and vertices, each with its particular attributes. Most Graph-oriented databases enable efficient graph traversal, even when the vertices are on separate physical nodes. Moreover, this type of database has received a lot of attention lately because of its applicability to social data.[30]

4. **Document**: A document is a series of fields with attributes.Most databases of this type store documents in semi-structured formats such as XML(eXtensible Markup Language), JSON (JavaScript Object Notation), or BSON (Binary JSON). They work similarly to Key-Value databases, but in this case, the key is always a document's ID and the value is a document with a pre-defined, known type (e.g., JSON or XML) that allows queries on the document's fields.[30]

As the saying goes, don't bring a knife to a gunfight. Using the right tool for the job is one of the most important things to reach the most desirable results. So we've decided to pick Firestore, which is a **Document database**, for these reasons:

- Offers great performance for mass read and write requests.
- Documents organized into collections.
- Assign an ID to each document which makes it easy to retrieve and manage.
- The documents are in the format of the JSON format (JavaScript Object Notation), which allows us great interoperability.

2.6.2.4. Our database schema

The figure below 2.5 illustrates the driver’s document:

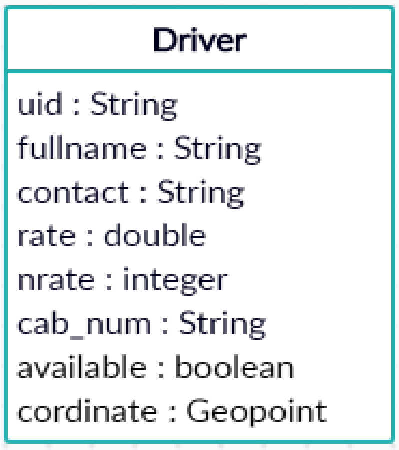


FIGURE 2.5 - Drivers document

The figure 2.6 below illustrates the Passenger’s document :

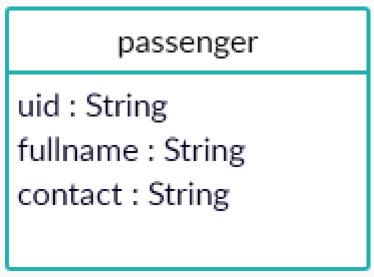


FIGURE 2.6 – Passenger’s document

The figure 2.7 below illustrates the Reservation’s document :

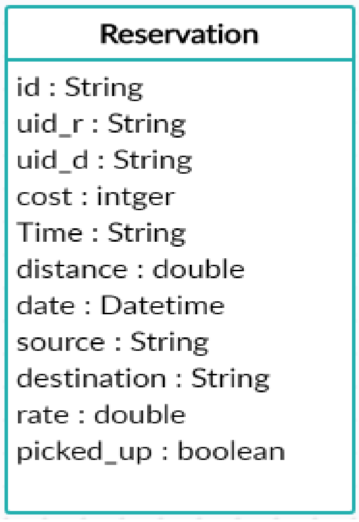


FIGURE 2.7 – Reservation’s document

2.6.3. Component diagram

One key element to develop a taxi booking application is the GPS. When it comes to GPS service there's only one name that rules. It's Google Maps. The figure 2.8 below illustrates the Component diagram of our system:

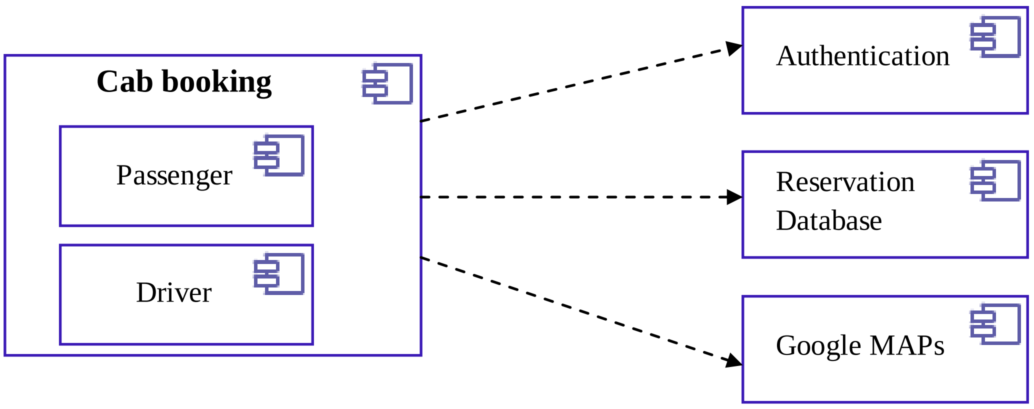


FIGURE 2.8- Component Diagram

2.7. Conclusion

This chapter introduced the design phase of our system using the Unified Process approach and the appropriate UML diagrams, the ones we found indispensable to our work. A description of the functional requirements of the different application actors and non-functional requirements as well as the UML design of the application are also presented.

CHAPTER 3: REALIZATION

3.1. Introduction

We continue with the last stage of our work, which is the realization phase by specifying the development tools used, programming languages and web services. These last ones allowed us to reach the desired result, which let us present the different services offered by our application.

3.2. Development languages

3.2.1. Dart

Dart is a programming language that was developed by Google. The first version of Dart was released on November 14, 2013, and version 2.0 was released in August 2018. It's open-source, object-oriented, strongly typed, a class defined, and uses a C-style syntax, which is to say, it's like many other modern programming languages, including Java or C#, and to some extent, even JavaScript. [31]

3.2.2. Dart features

- Owned by Google.
- Dart supports both just-in-time (JIT) compiling and ahead-of-time (AOT) compiling.[flutter in action].
- It can transpile to JavaScript to maximize compatibility with web development.[31]

Dart

FIGURE **3.1**- Dart’s logo[31]

3.2.3. JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.[32]

3.3. Development environment

3.3.1. Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA.[33] .

3.3.2. Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor that runs on your desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go, Dart) and runtimes (such as .NET and Unity).[34]

3.3.3. Android SDK

The Android SDK (software development kit) is a set of development tools used to develop ap[35] :

- Required libraries.
- Debugger.
- An emulator

- Relevant documentation for the Android application program interfaces (APIs).
- Sample source code.
- Tutorials for the Android OS.

3.3.4. Flutter

Flutter is Google’s UI toolkit for building beautiful, natively compiled mobile, web, and desktop applications from a single codebase. It aims to make development as easy, quick, and productive as possible. Things such as Hot Reload, a vast widget catalog, excellent performance, and a solid community contribute to meeting that objective and makes Flutter a pretty good framework.[36][37].

FIGURE 3.2 - Flutter’ logo[36]

3.3.4.1. Flutter's approach

Compared to other solutions, flutter performs much better because the application is compiled AOT (Ahead Of Time) instead of JIT (Just In Time) like the JavaScript solutions. Flutter eliminated the bridge and the OEM platform and uses Widgets Rendering instead of working with the canvas and events, which is up to its rendering engine. And it uses Platform Channels to use the services. Besides, it is not difficult to use platform APIs with an asynchronous messaging system, which means if you need to use a specific Android or iOS feature, you can do it quickly. Flutter also makes it possible to create plugins using channels that can be used by every new developer. So, to put it simply: code once and use it everywhere! [37]

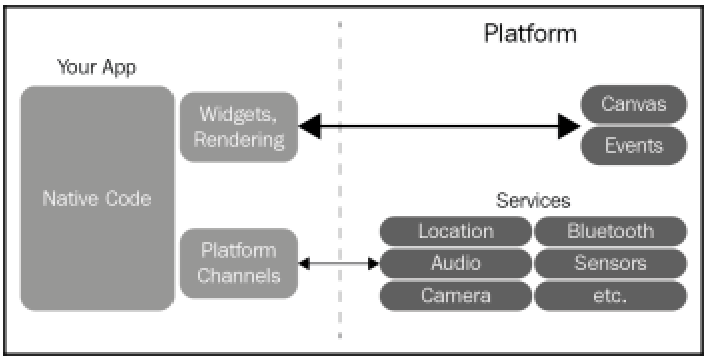


FIGURE 3.3 - Flutter's approach[37]

3.4. Firebase and web services used

3.4.1. Firebase

Firebase is a mobile and web application development platform created in 2011 by Firebase.Inc and then acquired by the company by Google in 2014 to be integrated into its Cloud services offering (Google Cloud Platform). Firebase's primary objective is to free you from the complexity of creation and maintenance server architecture while ensuring you rock-solid scalability and ease of use.

Firebase has several services ready to use and those that we picked to use in our application :

1. **Cloud Firestore:** Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It keeps the data in-sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity.[38]
2. **Authentication:** Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook, Twitter, and more.[39]
3. **Cloud Messaging:** is a cross-platform messaging solution that lets you reliably send messages at no cost. Using FCM, you can notify a client app that a new email or other data is available to sync. You can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.[40]

3.4.2. Web services used

Web services are self-descriptive, modular, and weakly coupled applications that provide a simple, standards-based model for programming and deploying applications and running through the web infrastructure.
There are two types of Web service :

- Web service based on the SOAP protocol.
- Web service based on the REST architectural style.

Considering the REST style's simplicity, we opted to invoke existing REST web services with JSON messages. As part of our application, we used some existing web services:

3.4.2.1. Google Map API

With the Google Map API, we can add maps based on Google Maps data to our application. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. We also use API calls to add markers, polygons, overlays to a basic map, and change the user's view of a particular map area.[41]

3.4.2.2. Distance Matrix API

The Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations. The API returns information based on the recommended route between start and endpoints, as calculated by the Google Maps API, and consists of rows containing duration and distance values for each pair.We used this for retrieve distance and duration of the trips [42].

3.4.2.3. Places API

The Places API is a service that returns information about places using HTTP requests. The Places API lets you search for place information using various categories, including establishments, prominent points of interest, and geographic locations. You can search for places either by proximity or a text string. A Place Search returns a list of places along with summary information about each place; additional information is available via a Place Details query.We used this api for retrieve places addresses[43]

3.4.2.4. Directions API

The Directions API is a service that calculates directions between locations using an HTTP request.[44]

3.4.2.5. Geocoding API

Google Maps Geocoding API is a service that performs geocoding and reverse geocoding of addresses.

3.5. Presentation of the graphical user interfaces

Our Project contains two applications, one reserved for the client and another reserved for the cab driver. In the following, we would like to present the interfaces of our cab booking system:

3.5.1. Login UI(user interface)

This Screen is shared between the passenger and the driver apps. The user has to choose to sign in within a Phone number or google account. If the user decides to use a phone number, he will receive an OTP code by SMS that must be entered on Otp UI to sign in.

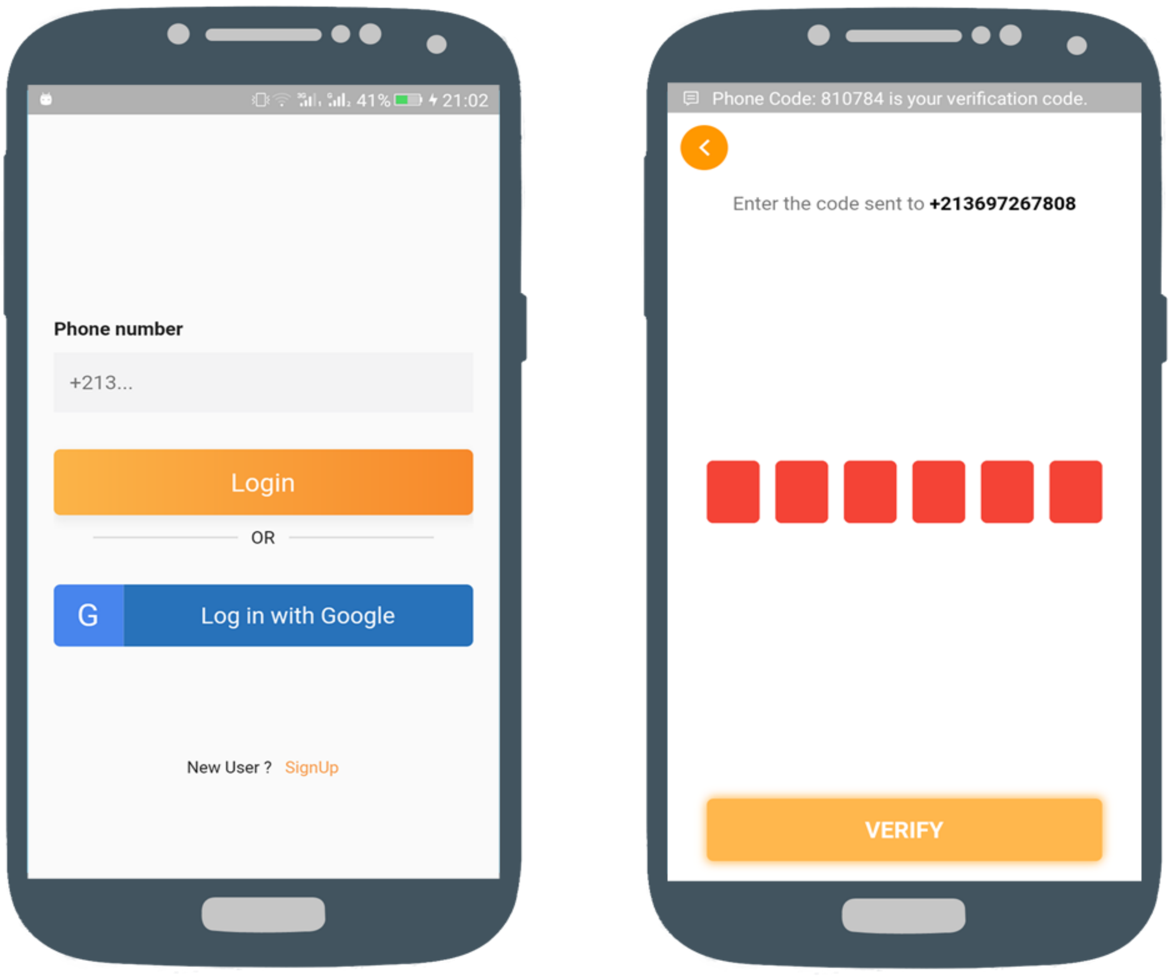

41% 21:02
Phone number
+213...
Login
OR
G
Log in with Google
New User ? SignUp
Phone Code: 810784 is your verification code.
Enter the code sent to +213697267808
VERIFY

FIGURE 3.4- Login and Otp Screens

3.5.2. Sign up UI(user interface)

This Screen is common between the passenger and the driver apps. The user has to enter his full name and choose to register within a Phone number or google account. If the user decides to use a phone number, he will receive an OTP code by SMS that must be entered on Otp UI to sign up.

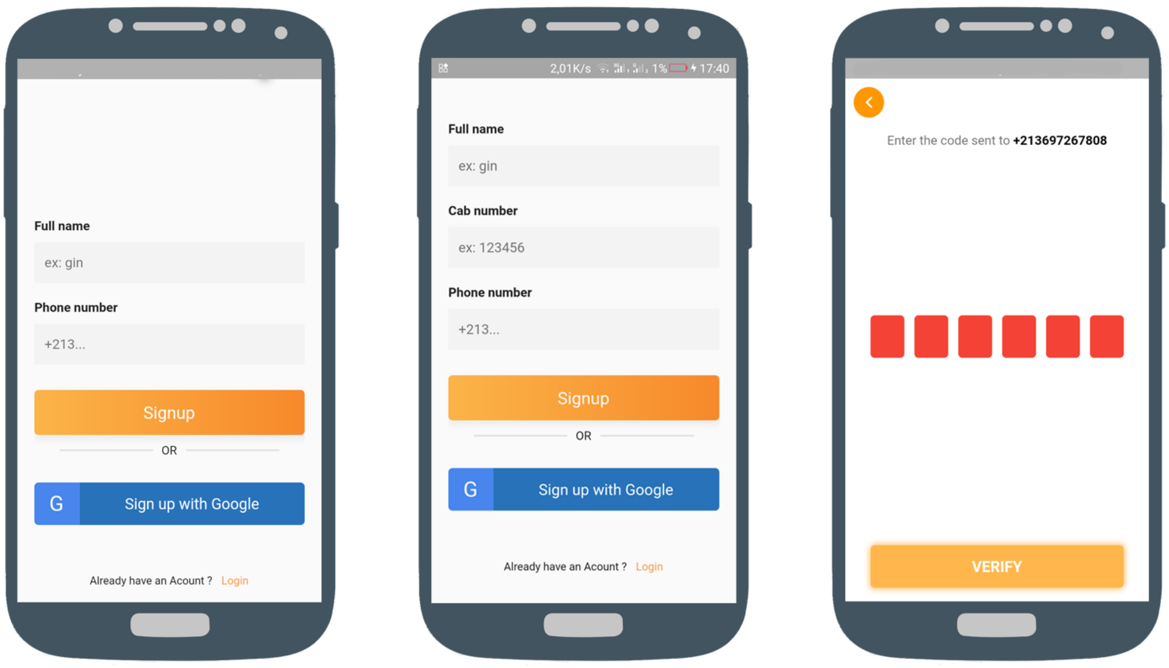


FIGURE 3.5- Signup and Otp screens

3.5.3. Menu Screen

When the driver and client access the application, he will be able to consult its menu, as shown in the following figure :

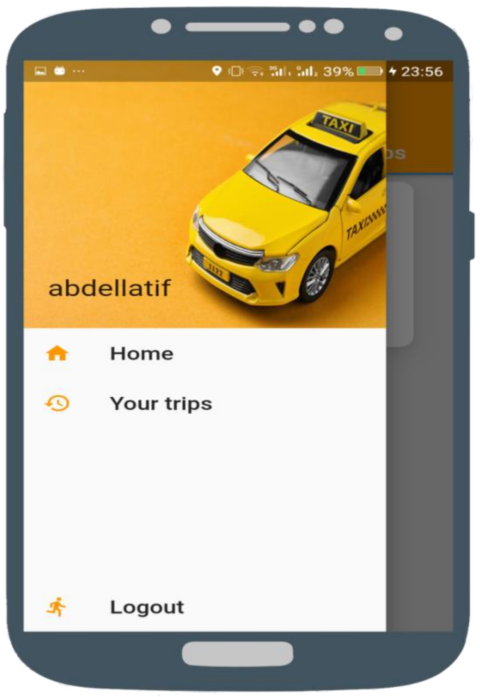
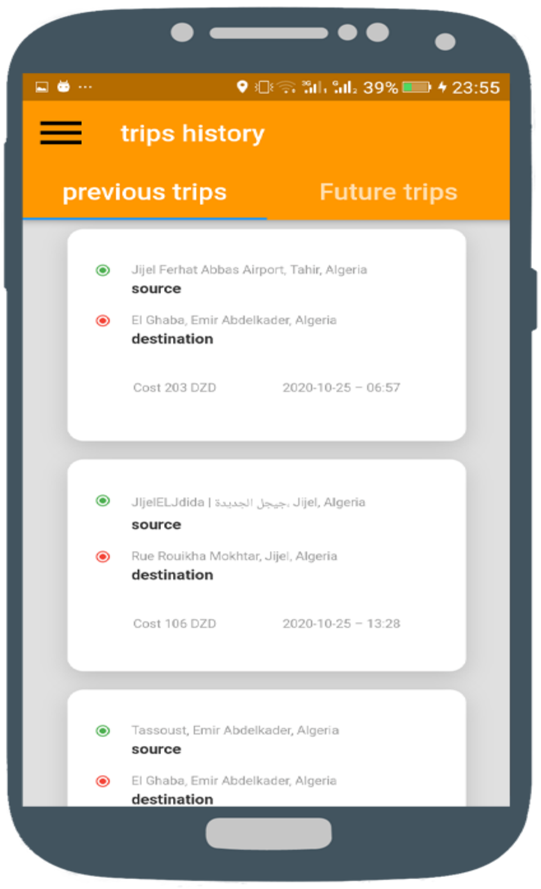


FIGURE 3.6 - Menu Screen

3.5.4. History Screen

This screen is unique for the driver app and the client app

- **Client :** the screen has tabs to show the previous and futter trips , as shown in the following figure :


23:55

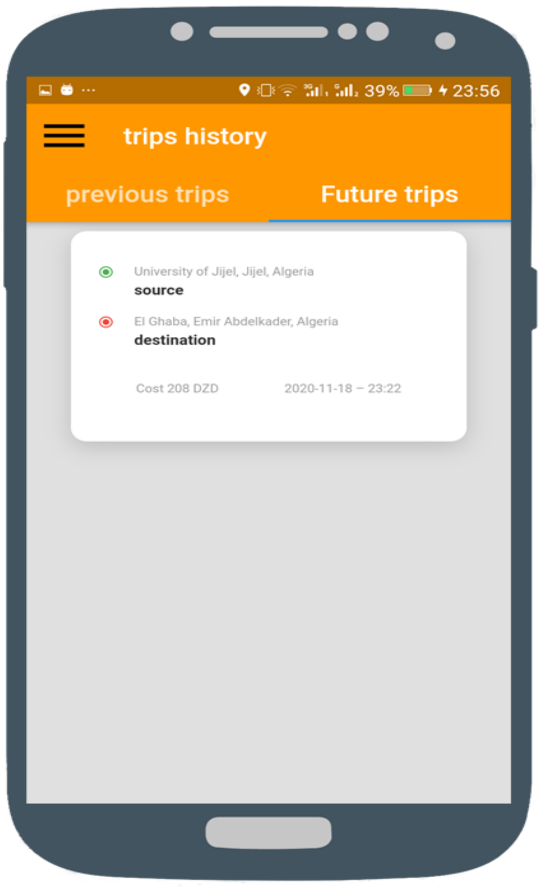
trips history

previous trips | Future trips

Jijel Ferhat Abbas Airport, Tahir, Algeria
source
El Ghaba, Emir Abdelkader, Algeria
destination
Cost 203 DZD 2020-10-25 – 06:57

JljelELJdida | جيجل الجديدة، Jijel, Algeria
source
Rue Rouikha Mokhtar, Jijel, Algeria
destination
Cost 106 DZD 2020-10-25 – 13:28

Tassoust, Emir Abdelkader, Algeria
source
El Ghaba, Emir Abdelkader, Algeria
destination


23:56

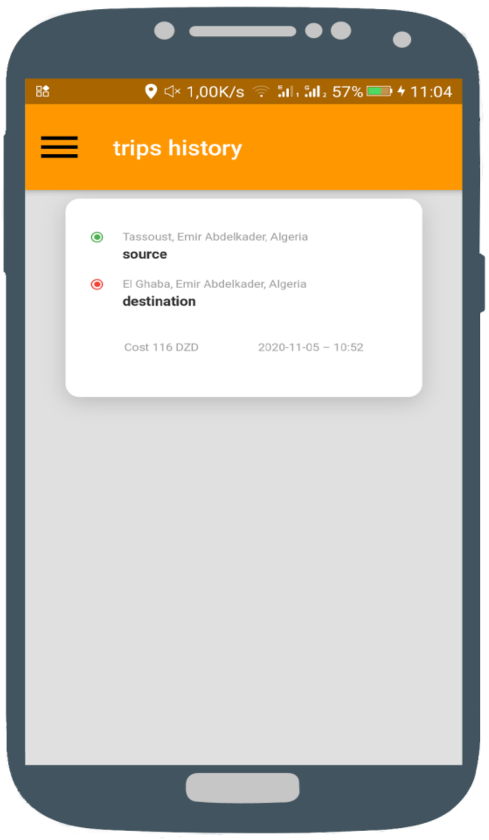
trips history

previous trips | Future trips

University of Jijel, Jijel, Algeria
source
El Ghaba, Emir Abdelkader, Algeria
destination
Cost 208 DZD 2020-11-18 – 23:22

FIGURE 3.7 - Client history screens

- **Driver :** The driver app shows all the trips that were carried by this driver.



Tassoust, Emir Abdelkader, Algeria
source

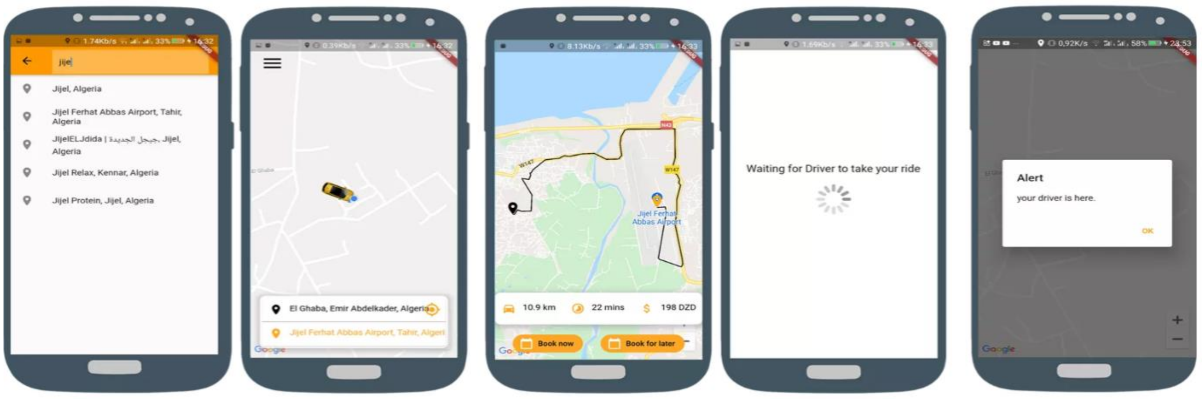
El Ghaba, Emir Abdelkader, Algeria
destination

Cost 116 DZD 2020-11-05 – 10:52

FIGURE 38- Driver trips history screen

3.5.5. Instant booking steps

The passenger's location is detected automatically; he can see all available drivers within a 3 km radius. He should pick the beginning and the end stations on the home screen. The app calculates the cost automatically using a formula which is defined by this equation: a base fare (100 DZD) +(mileage * 5 DZD + the number of minutes *2 DZD), the app also calculates the trip's duration expected and the distance and shows it with a road trace. Then he will hit the "book now" button, then he will be redirected to the "waiting screen," waiting for a driver to take his ride. When a driver accepts the booking request, the passenger receives a notification also when the cab driver arrives at the start location.


jijel
Jijel, Algeria
Jijel Ferhat Abbas Airport, Tahir, Algeria
JijelELJdida | جيجل الجديدة، Jijel, Algeria
Jijel Relax, Kennar, Algeria
Jijel Protein, Jijel, Algeria
El Ghaba, Emir Abdelkader, Algeria
Jijel Ferhat Abbas Airport, Tahir, Algeri
10.9 km
22 mins
198 DZD
Book now
Book for later
Alert
OK

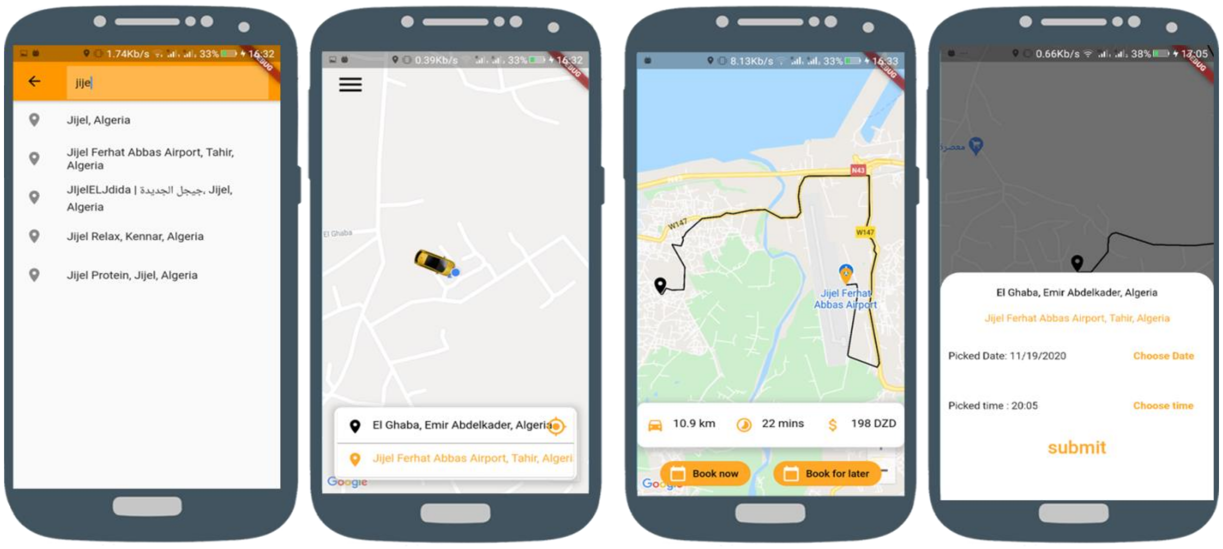
Waiting for Driver to take your ride

your driver is here.

FIGURE 3.9-Instant booking steps

3.5.6. Future booking steps

The passenger's location is detected automatically; he can see all available drivers within a 3 km radius. He should pick the beginning and the end stations on the home screen. The app calculates the cost automatically using a formula which is defined by this equation: a base fare (100 DZD) +(mileage * 5 DZD + the number of minutes *2 DZD), the app also calculates the trip's duration expected and the distance and shows it with a road trace. Then he will hit the "book for later" button, then he should select the time and the day.


jije
Jijel, Algeria
Jijel Ferhat Abbas Airport, Tahir, Algeria
JijelELJdida | جيجل الجديدة، Jijel, Algeria
Jijel Relax, Kennar, Algeria
Jijel Protein, Jijel, Algeria
El Ghaba, Emir Abdelkader, Algeria
Jijel Ferhat Abbas Airport, Tahir, Algeri
10.9 km
22 mins
198 DZD
Book now
Book for later
Choose Date
Choose time
submit

El Ghaba, Emir Abdelkader, Algeria

Jijel Ferhat Abbas Airport, Tahir, Algeria

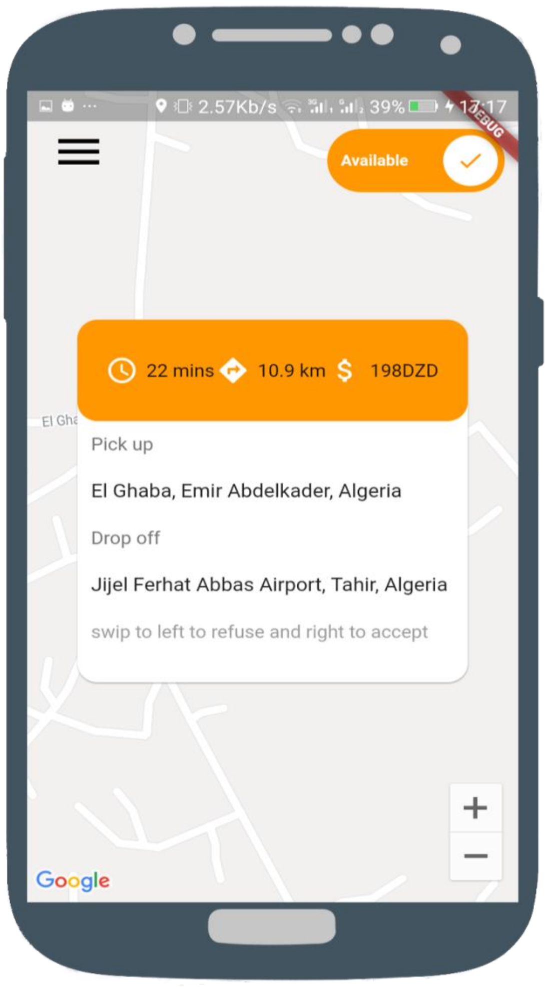
Picked Date: 11/19/2020

Picked time : 20:05

FIGURE 3.10- Future booking steps

3.5.7. Manage booking requests

All the cab drivers near the pickup location receive a notification of the client's reservation request. So the cab driver has to swipe right to accept or left to refuse.



22 mins 10.9 km $ 198DZD

Pick up

El Ghaba, Emir Abdelkader, Algeria

Drop off

Jijel Ferhat Abbas Airport, Tahir, Algeria

swip to left to refuse and right to accept

FIGURE 3.11 - Manage booking requests

3.6. Conclusion

In this chapter, we have described the practical aspects related to the implementation of our application, the environment, the development tools, as well as the languages used, followed by Firebase, without forgetting the web services that we used, then we presented the main interfaces of our mobile application.

CONCLUSION

Mobile applications for Taxi or cab booking is one of the basic needs of the general population nowadays, especially in urban areas since this application picks up significance among the general population. In this context, the objective of our work is the development of an Android application for a cab reservation system.

In order to achieve this goal, we have chosen to model our system with UML formalism based on a unified process approach. We have defined the functional requirements specifications through the use case diagram with the textual description of each use case. Once the system's functionalities were defined, we presented the design of our application using the class and component diagrams. The last step was an overview of all tools used for the realization of our application, followed by a presentation of the main interfaces of our application.

Our cab booking application gives its clients two options while booking, Ride Now or Ride Later, in a few clicks. Registration process is simple and booking process estimates fare and calculates the cost automatically.

Perspectives

In order to improve our application, we have drawn a line of perspective. We mention**:**

- Add in-app payment.
- Improve the UX(user experience).
- Create an IOS version for the app.
- Create an Admin panel to manage the backend.

BIBLIOGRAPHY

[1] https://www.riskiq.com/resources/research/2019-mobile-threat-landscape-report/.visited on 14/7/2020.

[2] Horn, U. et al. (1999). "**Services mobiles interactifs-La convergence de la radiodiffusion et des communications mobiles**." UER-revue technique, Union européenne de radio-télévision(281) : 14-19. ISSN : 1019-6595.

[3] Perchat, J. (2015). « **Composants multiplateformes pour la prise en compte de l'hétérogénéité des terminaux mobiles** » . Thèse de doctorat en Informatique. Valenciennes : Université de Valenciennes et du Hainaut-Cambresis, 08-01-2015, 209 p.

[4] « Mobile app », disponible sur l'url « https://whatis.techtarget.com/definition/mobile-app », visited on 15/7/2020.

[5] R. Minelli and M. Lanza (2013). **" Software analytics for mobile applications--insights & lessons learned**." Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, IEEE.

[6] https://www.hd-motion.com/quels-sont-les-differents-types-dapplication-mobile/ . visited on 16/7/2020.

[7]https://www.taktilcommunication.com/blog/applications-mobile/definition-typologie-applications-mobiles.html, visited on 16 /7/2020.

[8] https://generationmobiles.net/autre/les-differents-types-dapps-mobiles-leurs-avantages-et-inconvenients/.visited on 16 /7/2020.

[9] Android (2020). « **https://www.android.com/**»visited on 17 /7/2020.

[10] Sierra, F., & Ramirez, A. (2015). « **Defending Your Android App.** » Proceedings of the 4th Annual ACM Conference on Research in Information Technology - RIIT '15. doi:10.1145/2808062.2808067

[11] Naing Linn Htun , Mie Mie Su Thwin(2017) « **Proposed Workable Process Flow with Analysis Framework for Android Forensics in Cyber-Crime Investigation**», The International Journal Of Engineering And Science (IJES) ,Volume 6 , PP 82-92, 2017 .

[12] http:// devloper.android.com/guide/components/services. visited on 16/7/2020.

[13] http://igm.univ-mlv.fr/~dr/XPOSE2008/android/ , visited on 18/7/2020.

[14] Barry,P.Crowley.P. **«Modern Embedded Computing Designing Conncted Pervasive, Media-Rich Systems»**. 2012

[15] https://devloper.android.com/guide/components/activities/. visited on 16 /7/2020

[16] https://devloper.android.com/guide/components/services/. visited on 16 /7/2020

[17] https://www.tutorialispoint.com/android/android-broadcast_receivers.htm/. visited on 16 /7/2020

[18] https://www.tutorialispoint.com/android/androidapplication_components.htm/. visited on 16 /7/2020

[19] Slavulj, Marko & Kanižaj, Krešimir & Đurđević, Siniša. « **The Evolution of Urban Transport – Uber** ». (2016)

[20] HTTPS://WWW.UBER.COM/ES/EN/, visited on 17 /7/2020.

[21] ROQUES P et VALLEE F. UML en action de l'analyse des besoins a la conception en JAVA , 2eme Edition EYROLLES, 2003.

[22] JOCOBSON I. et BOOCH G. et RUMBAUGH J. **« Le processus unie de développement (the unied software développement process)** », Edition EYROLLES, 2000.

[23] https://sabricole.developpez.com/uml/tutoriel/unifiedProcess/. consulted on 20/9/2020

[24] PASCAL ROQUES, «**Les cahiers du programmeurs UML2 modéliser une application web** »,EYROLLES, 4e édition, 2008

[25] https://www.smartdraw.com/uml-diagram/. visited on 20/9/2020.

[26] https://creately.com/blog/diagrams/uml-diagram-types-examples/ , visited on 20/9/2020

[27] JOSEF GABAY, DAVID GABAY, « **UML2 Analyse et Conception** », Université de Québec, 1re édition, 2009.

[28] Vatika Sharma, Meenu Dave, « **SQL and NoSQL Databases »,** International Journal of Advanced Research in Computer Science and Software Engineering Research Paper, Volume 2, Issue 8, August 2012.

[29] https://aws.amazon.com/nosql/, visited on 20/9/2020.

[30] Alejandro Corbellinin, Cristian Mateos, Alejandro Zunino, Daniela Godoy, SilviaSchiaffino, « **Persisting big-data: The NoSQL landscape**», Information Systems, Volume 63,2017, Pages 1-23, ISSN 0306-4379.

[31] Alessandria,S. « **Flluter Projcts»**Packet**.**2020

[32] https://www.json.org/json-en.html . visited on 18/10/2020

[33] https://developer.android.com/studio/intro . visited on 18/10/2020

[34] https://code.visualstudio.com/docs. visited on 18/10/2020

[35] https://www.techopedia.com/definition/4220/android-sdk. visited on 18/10/2020

[36]https://flutter.dev/docs. consulted on 18/10/2020

[38] https://firebase.google.com/docs/firestore. consulted on 18/10/2020

[37] Giordano,S.,Mainkar,P. « **Google Flluter Mobile Devlopment Quick Start Guide.**»Packet. ISBN=9781789344967, March 2019.

[39] https://firebase.google.com/docs/auth. visited on 18/10/2020

[40] https://firebase.google.com/docs/cloud-messaging. visited on 18/10/2020

[41] https://developers.google.com/maps/documentation/android-sdk/overview. visited on 18/10/2020

[42] https://developers.google.com/maps/documentation/distance-matrix/overview. visited on 18/10/2020

[43] https://developers.google.com/places/web-service/overview. visited on 18/10/2020

[44]https://developers.google.com/maps/documentation/directions/overview. visited on 18/10/2020.