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 :

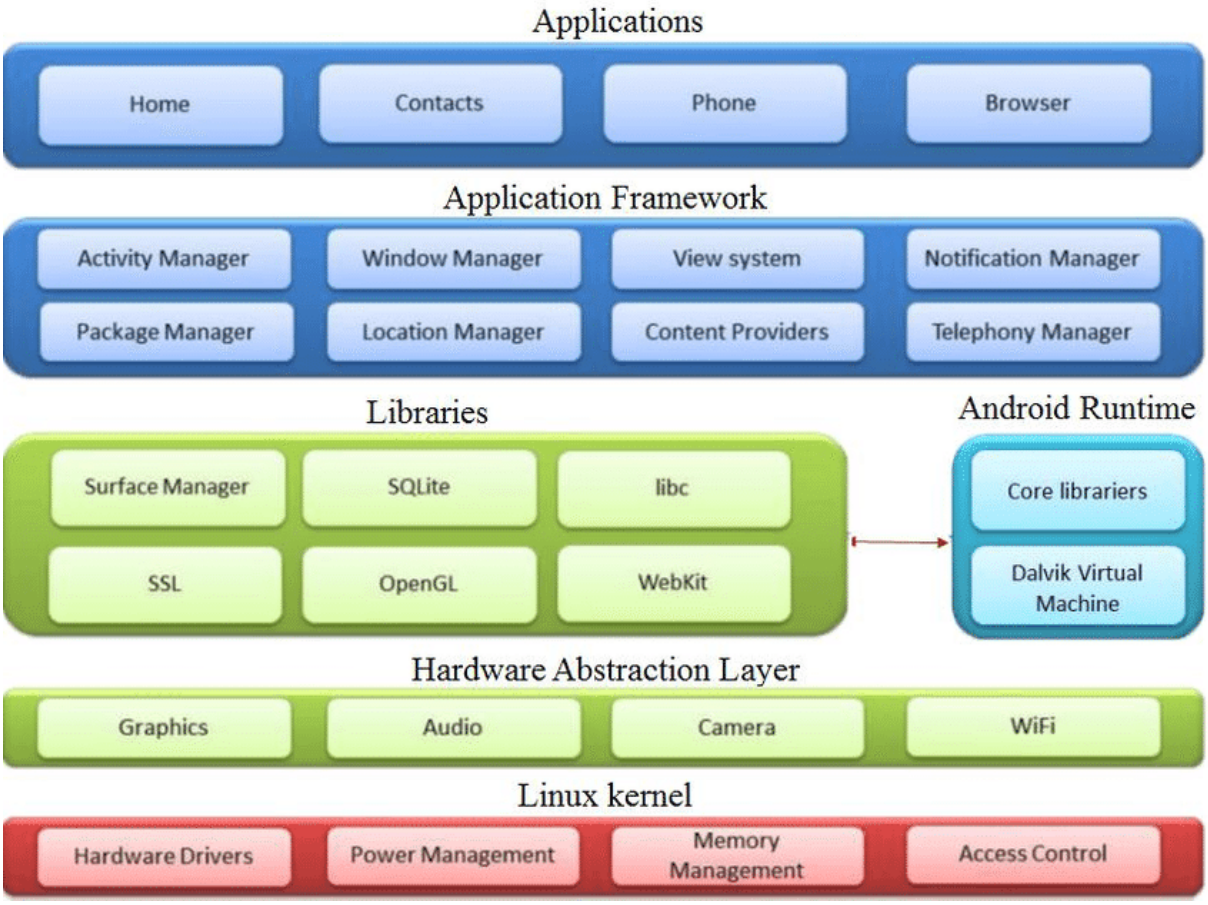

FIGURE 1.2 - Architecture d'Android [11].

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 :

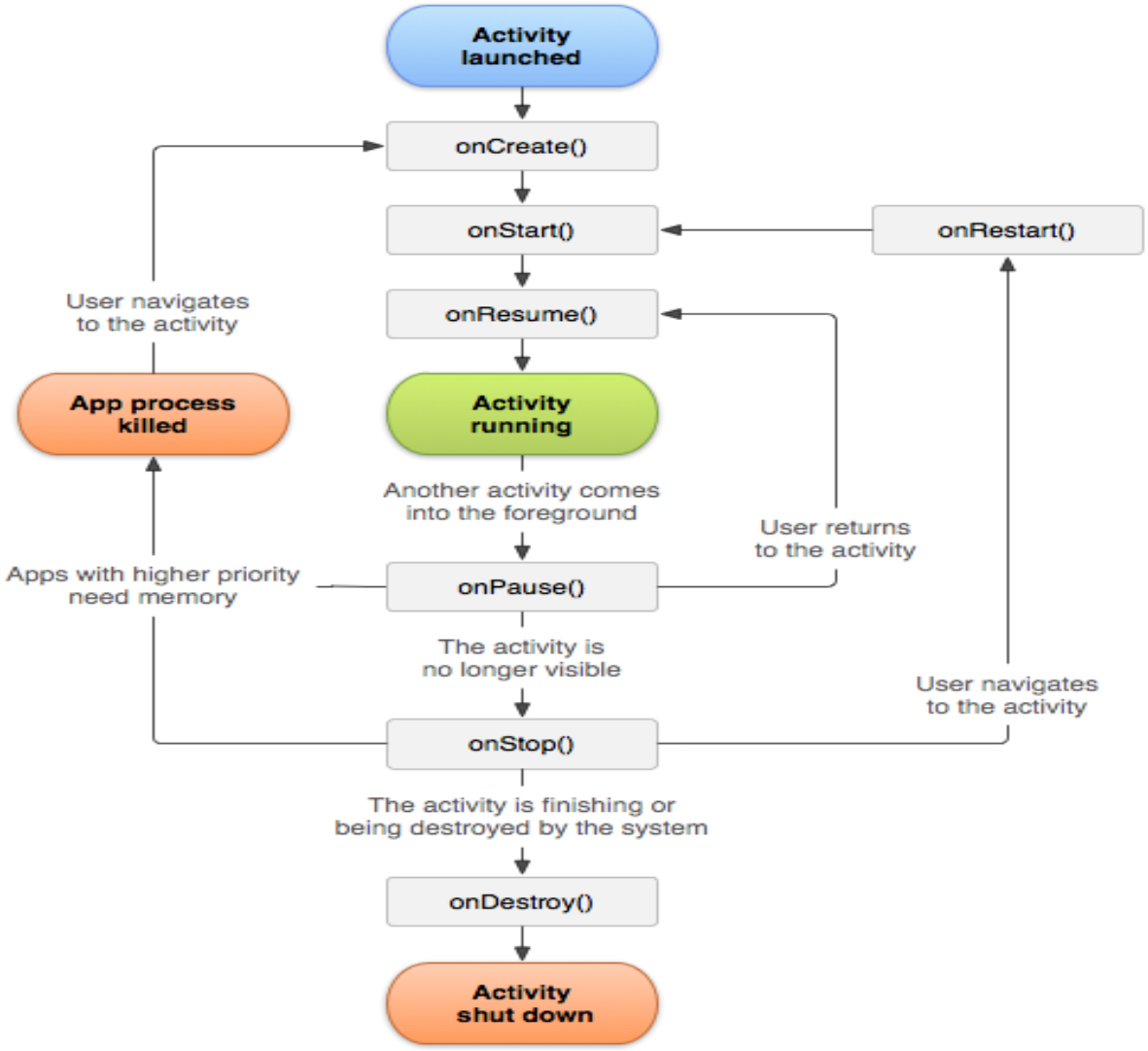

FIGURE 1.3- Activity lifecycle[16]

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

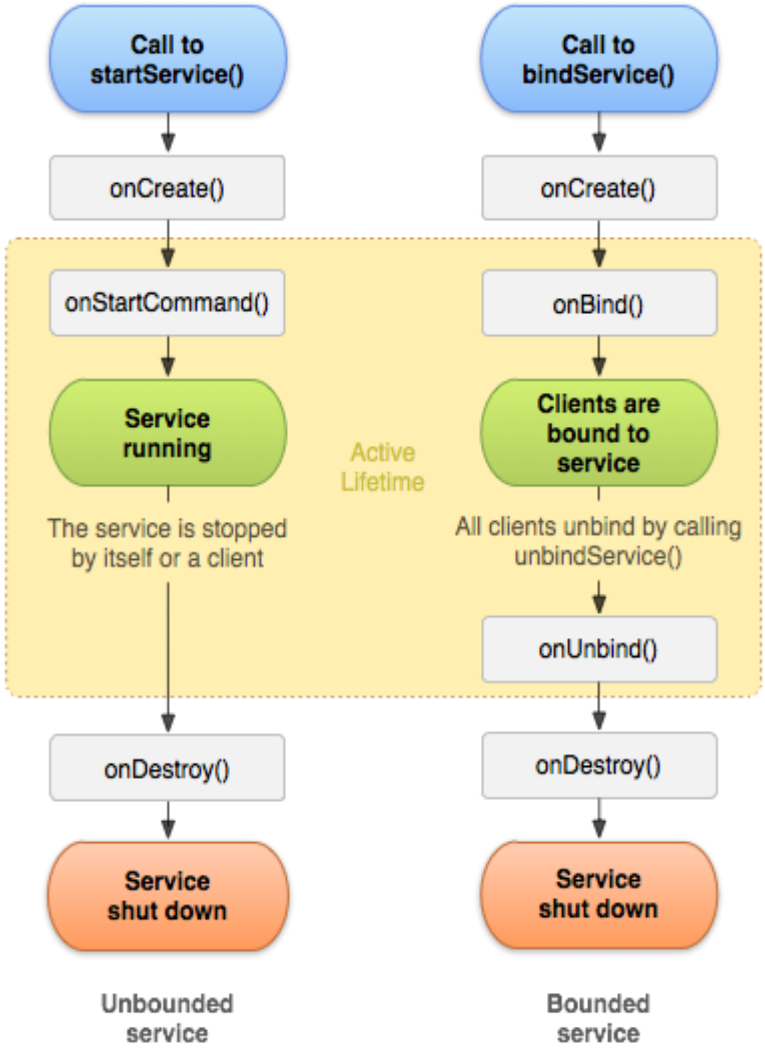

FIGURE 1.4 - Service lifecycle [17]

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.

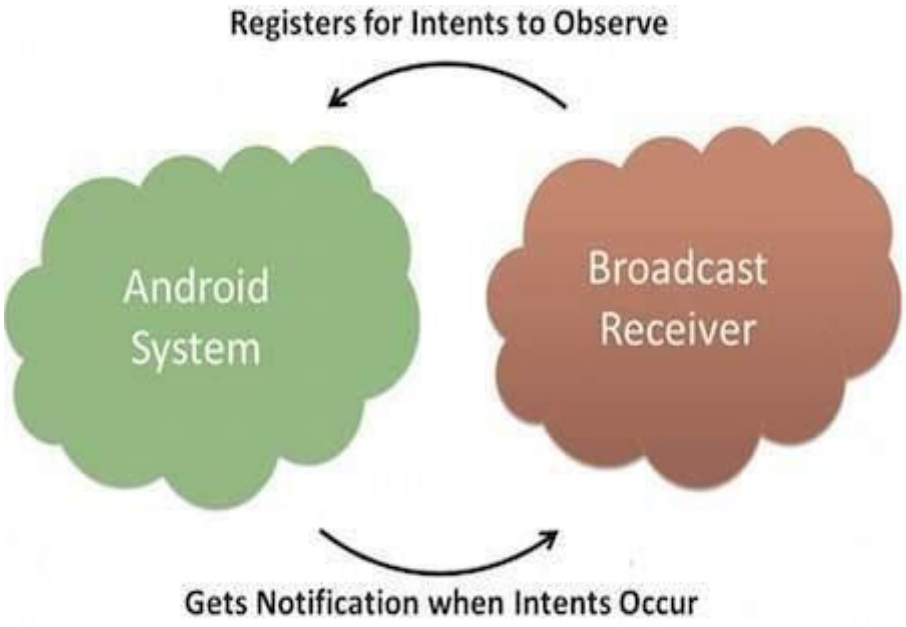

FIGURE 1.5- Broadcast Receiver [17]

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]

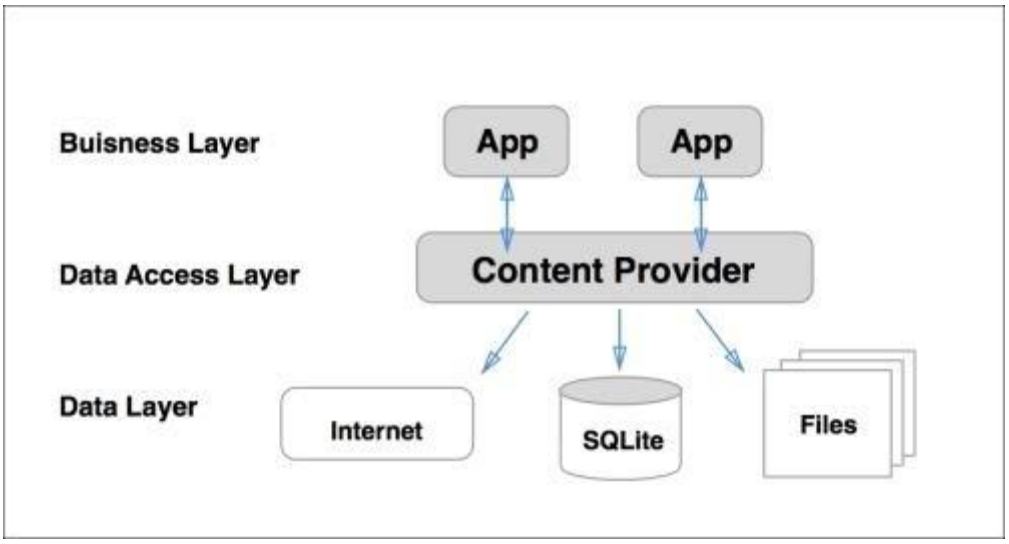

Figure 1.6 – Content providers [14]

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.

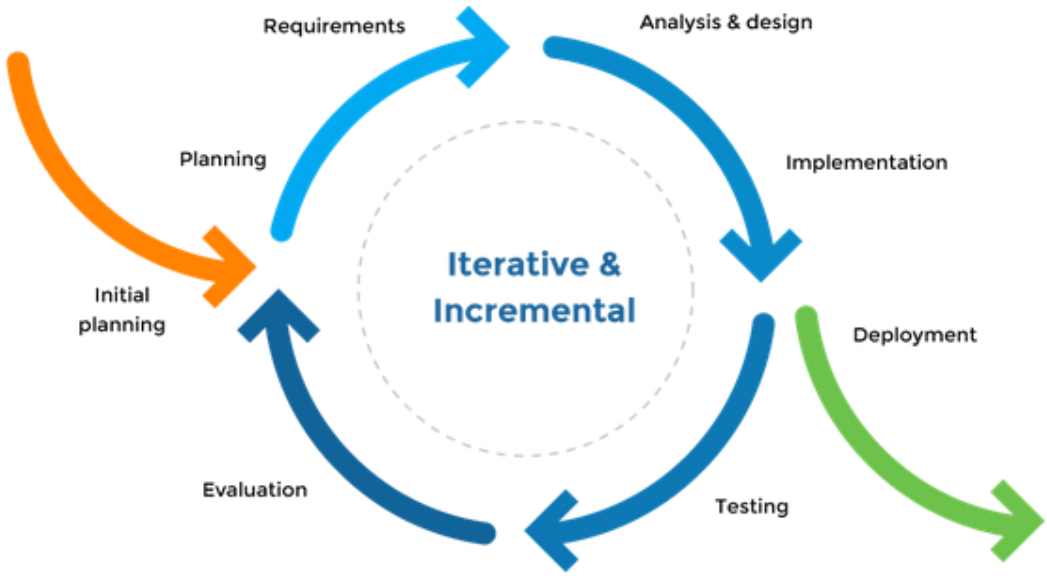

FIGURE 2.1 - The iteration of UP[22]

- **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]