Budgetize:

Money tracking Android Application

Bachelor thesis

A close up of a logo

Description automatically generated

Robert Marian Roland CIOTOIU

Supervisor:

Prof. dr. eng. Marius MARCU

# Introduction

## Motivation

Most people are usually neglecting how they spend their money. As in the past decades the card payments are continuing to grow in popularity and people are tending to spend more. This is due to the impression that our money is only a number on the screen. This fact is leading us to spend more and to care less about their importance.

All over the world, too many people with different income amounts, are struggling to manage their budgets. I am also seeing around me, that the majority of people are only thinking about how to make more money to solve their financial issues, but they don’t realize this is redundant if their debts will increase proportionally.

Nowadays, mobile phones are part of our everyday life. We carry them everywhere we go, we check them on average 60 times a day and we spend almost 3 hours every day on the phone [1]. According to Statista [2],3.5 billion people are owning a smartphone and this number is growing with 300 million people per year. This tool, which we have in our hands every day, it is capable to do a lot of things which makes our life easier. From checking the weather, reading news, making bookings, calls, meetings to doing payments, checking balance and even help us to improve the way we are managing our budget.

Banks are starting to incorporate into their mobile applications tools like “Analytics” in order to improve transactions tracking, money saving and budget administration. This is very useful to use in case of having bank accounts only to one bank. But things get more complicated when we have multiple types of bank accounts (e.g. credit, debit, savings) at multiple banks. Also, bank applications usually do not allow users to add transactions to those analytics as for example to track cash payments.

What defines 2020 is the power of getting information. We are now one click distance to find an answer for what we need. In bank industry, this thing is very new. Banks used to hold the data very secured, only for them or eventually for partner banks. Today, thankfully to the initiative of Open Banking, we can have third party applications made any organization, which are able to display to the end users, data about their bank accounts and even allow them to make payments.

## The problem

Firstly, the root cause of all our financial issues is the lack of financial education. We are not used to write all the incomes and expenses down to a paper, to eliminate the unnecessarily monthly payments and to learn how to avoid useless acquisitions. Surely, at one point of our life, we tried that, but this is not easy to do and maintain, and most of the people quit as soon as they start.

Even if we are determined to do that, in nowadays, it is almost impossible to keep that tracking on a paper. We would need to use special software, which is generally hard to use for the average user and which usually require a monthly subscription. Writing down every transaction we made is hard and requires a lot of time. To keep the tracking of a family expenses is even harder.

Simply writing down the transactions on the mobile phone every time we make one and sum up at the end of the day with reports gathered from each bank account will never work. If every member of the family has more than one account at more than one bank, the tracking and the ability to improve the way we spend our money would require too much effort.

## The solution

This overwhelming process of keeping the track of expenses can be simplified a lot using the power of information.

Mobile phones and Internet are enablers for digitalization of almost everything. In our case, the incomes and expenses which used to be written down on a paper. The most important think we have nowadays, is the information. How we get that information, how we process it and, the most important thing for a user, is how we display it.

From already 5 years, we can get that information from any bank. We can process information from any bank account and decide how to display it in a manner which helps the end users to evaluate their spending in relation to their income.

The solution I came with, in order to solve all the above issues, is an Android Application in which any end user of a smartphone can add any type of transaction, set-up recursive transactions, categorize those in virtual wallets and link real bank accounts to them. By linking the bank account to a wallet, the application will assure automatic synchronization of transactions. All these functionalities are intuitive and easy to use by any Android user. Also, the application will offer all these functionalities for free as the income will be generated by applying the Freemium Business Model. It will contain only ads with minimal impact on user experience and some usage limitations on the key functionalities of the application.

I chose to develop the application on the Android platform because it has 70.68% [3] of smartphone market share, it is free to develop application and it is very cheap to publish on the Android Marketplace.

The application will enable user’s interest to manage their finances and will cover the lack of financial education we are currently have in our society.

# State-of-the-art

## Overview

On the market, there are a lot of applications available for tracking expenses. Many of them are destinated for businesses and for people which already have some experience on managing their budget. In general, applications designed for an average user, require a monthly subscription in order to use the application and some special functionalities like linking bank account and cloud synchronization. The applications that are free usually neglects the user interface and the user’s experience.

## Main competitors

Researching the Android and IOS Marketplace, I found three main potential competitors at this time. In the next paragraphs, I will make a comparison by exposing the benefits and detriments of using *Budgetize* or any of the following leaders in the Smartphones Marketplace.

One application I found which almost combines all the principles I had in mind when I developed *Budgetize* is *Wally*. *Wally* has been removed from Android Marketplace and it is only available now on IOS. I still want to compare my application to *Wally* because it was the only one I could found, to be able to link my bank account for free and to import my history transactions.

*Wally*is free to use, but also has some paid functionalities. It has a wide range list of supported banks which can be linked to the application, it also allows to organize the transactions on categories and wallets, we can add transactions manually and it is able to display information about the introduced and imported data in a lot of ways, all of those for free. Until there we are sharing the same functionalities, but what differentiates *Budgetize* from *Wally* are features like creating foreign accounts and ability to switch the currency of any wallet which we offer for free. But the most important advantage of *Budgetize* is the user experience.

From the first touch of *Wally*,I have been overwhelmed of the many configurations and views, the lack of animations, crashes in some cases and how unintuitive is the application to use at the first sight. For sure *Wally* has a lot more ways to display the information about the data we introduced/import, has more configurations and it has some interesting features for the premium version, but the target of *Budgetize* is to be appealing to anyone who installs the application, to demonstrate it is very easy to manage the budget and to track the expenses.

Another big competitor is *Spendee*. The strongest element of *Spendee* is the user experience and the user interface. Animations, reports and analytics are very well displayed. The application is easy to use and very intuitive. One important feature of *Spendee*, which *Wally* doesn’t seems to have at least in the free version, is the biometric access to the application. I consider that feature extremely important for an application which is handling sensitive data like bank account information.

Even having all the benefits from above, the downside of the application is that you need to add each transaction manually without having any other possibility in the free version. For importing transactions from bank accounts and create multiple virtual wallets, there are multiple premium versions which require monthly or yearly payment, or a big one-time payment.

Only on Android *Spendee* has over 1.000.000 downloads and over 28.000 reviews with a rating of 4,1 of 5 [4]. Those numbers demonstrate that there is a need in area of financial management.

The last competitor I would like to mention is *Bankin’*. This Android Application also has over 1.000.000 downloads and almost 40.000 reviews with a rating of 4.7 of 5. The biggest advantages of this applications are the ability to link the bank accounts, analytics and multiple ways of displaying the transactions or view them based on their type. All those features are free, also the application have premium functionalities like a personal coach but unfortunately that content is only available in French. Also, other parts of the applications are displayed only in French and this cannot be changed. Another downside is that users cannot track all of their spending because the application doesn’t provide the functionality to manually add new transactions, it only provides the ability to import transactions from bank accounts. Even this may be sufficient for some users, this application has a limited number of banks and I wasn’t able to find any supported bank from Romania. This makes the application to have no usage in some countries or for some users which have accounts at unsupported banks. The design of that application is basic, without having any innovation in my opinion and this design pattern can be found in a lot of other applications, especially for budget tracking.

## Conclusions

*Budgetize* by combining the main key functionalities, for saving user’s time on adding the transactions manually, with easiness of using the application and an appealing user interface. Also, *Budgetize* has been developed with security awareness, offering the end user credibility and safety. At the design part, *Budgetize* combines multiple elements in order to gain a spirit of originality, clearly differentiating from the rest of the applications from the market from its segment.

Having into consideration the above-mentioned features of Budgetize, I definitely think those will create its own place on the Android Marketplace and enable the opportunity of acquiring a lot of users.

ADD HERE A DIAGRAM ON HOW I POSITION ON THE MARKET RELATING TO THE ABOVE DESCRIPTED APPS(PE MODELUL DIN PREZENTAREA DE LA ANTREPRENORIAT)

# Theoretical Foundation

## Concepts

### Object-oriented programming

Object-oriented programming is a programming paradigm consisting of organizing the data into objects. The object represents the instantiation of a class, considering the class as a blueprint.

When the object is instantiated, it can be “personalized” by giving values to the constructor. In general, classes can contain variables, constructors, destructors, getters and setters, methods, inner classes, etc. but those can differ from a programming language to another. The behavior of the class is defined through its methods.

The main principles of object-oriented programming are: Encapsulation, Abstraction, Inheritance and Polymorphism [5]. Those principles are used in order to build clear and modular structures and to have reusable code and more flexibility [6].

The encapsulation is realized by setting the access modifier to the class, constructor, variable method, or data member. Encapsulation is simplifying the maintenance of the application, makes the application easier to understand and it is also reducing the human errors [7].

Abstraction is very important because it reduces the complexity of the code, it is avoiding code duplication, increases the usability and it is increasing the security of the application. In Java for example, the abstraction is realized by using abstract classes and interfaces [8].

Inheritance is a concept which helps the developer into organizing the code in a hierarchy. This is assuring the reusability of the code and maintains a defined structure of the objects [5].

Polymorphism is an important concept because it is unlocking the ability to Override and Overload. It is realized using inheritance and child objects can Override and Overload parent’s methods. This concept consolidates the structure of the program and helps to reuse code.

### Android Activity Lifecycle

Every Android Activity has a lifecycle. A lifecycle means that the Android Activity can go through different stages as Figure 3-2 illustrates. These stages are defined in order to save resources like random access memory. Another reason is to have applications with different priority. This is empowering the Android Operating System to know which Activities can be destroyed in critical situations or which Activities are not unused anymore and still consume a lot of memory or computational power.

Whenever the activity is changed, it can be paused or destroy. The developer must be aware of this lifecycle in order to avoid application crashes. This can be done by using the android activity’s class callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy() [9]. We are able to override every of these methods in our Activity and correctly handle what happens with Activity content in order to avoid crashes or memory leaks. Overriding onCreate() is mandatory in order an activity to start and work.



Figure ‑. A simplified illustration of the activity lifecycle. (reference: [9])

### Model-View-ViewModel (MVVM)

The Model-View-ViewModel is an architectural pattern where the frontend and backend are separated. This pattern is widely used in mobile and desktop applications because it is moving the complicated logic from the View into the View Model. This allows the View only to manage the way it is displaying the data read from the View Model.

This pattern is very important to be used in Android development in order to avoid “God” Activities. In Model-View-ViewModel, it is very important for the View Model to do not know anything about the View. The View Model role is to get data from the Model, process it and to offer methods available to provide this processed data to the View. The View must know how to use the View Model, what it can request and what it will receive. Its purpose is just to display the provided data without doing a lot of process with it. In the other way, the View can also be responsible to send the data to the View Model in order to process it, save it and even return a response. Communication between the View Model and the View can be realized using bindings, commands, or events (Figure 3-1).



Figure ‑. Model-View-ViewModel diagram (reference: [10]).

In Android, the View from Figure 3-1 is represented by the Activity. A very important property of the ViewModel in Android is that it will be created on the onCreate() method of the Activity and destroyed only when the Activity is finished(Figure 3-3). It is not affected by actions like screen rotations, where an activity can be destroyed for instance.

Moving the logic of the application in the ViewModel, is a very good choice in order to have a well-organized code. Adding the backend logic in the Activity, the programmer must make all that logic lifecycle aware, which will lead to adding a lot more code to the activity and complicating the initial logic very much. This can be easily avoided by using the Model-View-ViewModel architectural pattern.

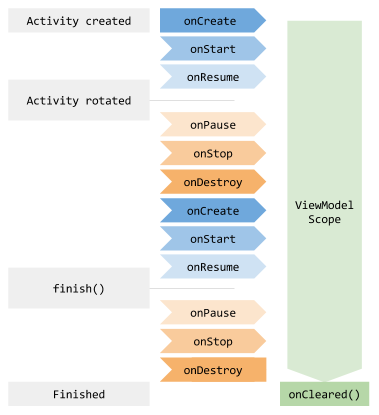


Figure ‑. Illustration of ViewModel Lifecycle (reference: [11])

Model-View-ViewModel is making the code reusable, modular, maintainable, and most important, it enables the possibility of collaborative work. Different types of programmers can work on the same screen for example, one very skillful on the user-interface design and one very good at the implementation of backend logic. Any layer from Figure 3-1 can be replaced easily, because of the modularity that Model-View-ViewModel enables. Because the frontend logic is separated to backend logic, we can easily maintain the code, refactor, add new functionalities, UI elements or even change the provider of data.

### Android LiveData

LiveData is a lifecycle-aware data holder which follows the publisher-subscriber architectural pattern. It is used together with an Observer, as LiveData is notifying the Observer when the data is changed. We can override the onChanged() method of the Observer and there we can implement what to happen when the data is changed. In this way, we can keep the Views from the UI permanently updated.

Being lifecycle-aware, it means that the LiveData will notify the Observer, from the Activity, only when the Activity is in an active state. When the Activity is Destroyed, the Observer is automatically unsubscribed to the live data. Those actions are making LiveData to avoid crashes in the case the Activity gets destroyed. Also, because the Observers gets unbound automatically, they are cleanup from the memory in order to avoid memory leaks. As all this measurements are done automatically by the LiveData, there is no more manual lifecycle handling needed [12].

We can communicate between View, ViewModel and Model using LiveData very easily, with lifecycle-awareness. So, by using the Model-View-ViewModel pattern in combination with LiveData, we will get a clear architecture, easy to maintain and easy to develop on.

### SQL

Structured Query Language (SQL) is a programming language used and designed to work with relational databases. With SQL we can manipulate data from database. SQL can contain clauses, expressions, predicates, queries, statements, and insignificant whitespaces [13].

### RESTful Web Service

A RESTFul Web Services are web services build on the REST architectural style. This architectural style is based on a client-server communication, using HTTP as a communication protocol. The main philosophy of REST is to build lightweight, simple, and fast applications.

Using REST, servers can expose services in order to be accessed by various clients. Those services are called “endpoints”. REST, as well as SQL and HTTP, follows the CRUD paradigm [14]. CRUD is an acronym for the operations create, read, update, and delete. Those are the essential operations to work with data. In REST, those operations are named the same as in HTTP: PUT, GET, POST and DELETE [15].

Every REST endpoint is a mapping to one of the four enumerated operations. Using REST, we can easily build and use APIs. REST eases the way we transmit and use data.

### RESTful API

API stands for Application Programming Interface. A REST API is a collection of endpoints which are deserving together as a computing interface. [16]

## Used Technologies

### JAVA Programming Language

Java is one of the most popular programming languages in present according to GitHub [17]. It was released in 1995, developed by Sun Microsystems. Java is an Object-Oriented Programming language, where classes and objects are the core of the language. Java is mostly known because of its Virtual Machine, which is able to run on any computer architecture. This allow us to write Java applications which can run on any machine and any operating system [18]. In order to run Java application, Java Runtime Environment (JRE) needs to be installed on the machine. When the Java code is compiled, it results in Java Bytecode which is executed by the Java Virtual Machine. Some platforms are able to directly execute Java Bytecode, without the need of installing JRE, as the hardware is built in this manner.

An important principle of Java is to be simple, object oriented and familiar [19]. Java follows the C/C++ syntax, but it a simplified form, and it has less low-level access.

A java important feature is the automatic memory management handled by the garbage collector. In other Object-Oriented Languages, we have to manually deallocate objects from memory after we used them. In Java, this job is done automatically by the garbage collector. This kind of thing has some advantages and disadvantages. One advantage is that the programmer doesn’t have to worry about deallocation of the objects or about possible crashes if the deallocation has been done too early, thankfully to the automation of the memory management. This is speeding up the development process and keeps a cleaner code. A downside is that we cannot deallocate from memory objects which we are sure we are not using them anymore and they may occupy a lot of memory. This may lead to performance issues. Having more flexibility in C++ offers us the ability improve application’s performance but with the tradeoff that we have to write much more code and be more careful to memory. [20]

Android applications are build on Java. Another big advantage of Java is that it is continuously updated, and new versions are released constantly. They also release some versions with long term support.

### XML (eXtensible Markup Language)

XML is a markup language and a file format mostly known for being human and machine readable. XML files have a lot of applications in web area, for code generation, distributing and carry data, etc. It is mainly composed of tags and elements. XML can also have empty tags (tags which doesn’t contain any value between the start-tag and the end-tag). Inside the start-tag, we can have attributes. An attribute is a name-value pair. [21]

XML also has a feature named “XML declaration”, which is describing information about the XML file itself. For example, it can describe the version of the XML and the encoding. [21]

XML are used in programming a lot. There are a lot of XML parsers for each programming language, as well as XML builders. In Java for example we have the well-known SAX Parser and DOM Parser. Also, XML is used for example by the SOAP messaging protocol as message format. XML is also used intensively in Android Development. Using XML, we define the Activity layout, the Fragment layout, the Manifest file, or other files like colors, strings, styles, etc.

### Gradle

Gradle is a high performant build automation tool used to build any type of software. It is open-source and it follows the concepts of Apache Ant and Apache Maven. Regarding syntax, instead of using XML like Apache Maven, it uses Apache Groovy. [22]

Gradle is tightly bound to Java, as Apache Groovy is a Java compatible syntax and it runs on the Java Virtual Machine. This enables the Gradle ability to run on any platform, if the JVM is installed or if the hardware is able to execute Java Bytecode. Even Gradle runs on the Java Virtual Machine, it is not limited to run only Java project. A downside of Gradle is that it is supporting repositories and filesystems only from Apache Maven and Apache Ivy. [23] Gradle is already integrated in some big IDEs like Android Studio, Eclipse, IntelliJ or NetBeans. [24]

### Apache Maven

Maven, as Gradle, is a build automation tool used for multiple types of languages. In order to build a project with Maven, we have to create a Project Object Model (POM) file, where we write all the dependencies we need. POM files are eXtensible Markup Language (XML) files. A very important functionality of Maven is the ability to write and use Plugins. Plugins can be used to perform different tasks like running builds, tests, generating project files etc.

### Android Operating System

Android is the mostly used operating system in the smartphone industry with a market share of 70.68% [3]. It has been launched on 23 September 2008, almost 1 year after the first IOS release, by Google. This operating system was used first on the smartphones, but now we can find it also in watches, smart glasses, cars, cameras, smart TVs, home appliances and many other devices. [25]

Regarding Figure 3-4, the Android Operating System is made of 6 important layers. The core layer of Android is based on a modified version of the Linux Kernel, because it is providing the standard functionalities to communicate with the hardware. That core layer has implementations for handling Drivers and Power Management.

The next layer is called Hardware Abstraction Layer (HAL) and it is an interface between the Kernel and the upper level layers. This layer contains libraries which are used by the upper level Java APIs. Without HAL, we would not be able to use the Bluetooth, Sensors or Camera effortless directly from Java.

The third layer is the Android Runtime. This layer is assuring that each application from Android is running in its own Runtime (an instance of Android Runtime) and in its own process. ART was introduced in Android 5.0 Lollipop, prior to it, Android used the Dalvik process virtual machine. Both ART and Dalvik VM uses the Dalvik bytecode and stores the bytecode into .dex (Dalvik EXecutable) files. ART replaced the Dalvik VM because it is more performant due to its new features [26]. The first new feature is composed of two new compilation methods, Ahead-of-time (AOT) and Just-in-time (JIT), which are in fact just some compile optimizations [27]. The last two features of ART are the improved Garbage collector and better debugging support. [28]

The fourth layer the Android Operating System is represented by the Native C/C++ Libraries used in the Android Runtime layer and Hardware Abstraction Layer. Those, for example, offer support for drawing and manipulating 2D and 3D graphics in the Android Applications. Those libraries can also be accessed by the developers using the Android NDK, in order to implement some functionalities or algorithms directly in C/C++ [29].

The fifth layer is the Java API Framework. This consists in Content providers, View System and Managers. Those are the APIs offered by Android, which the developers are using a lot for creating Android Applications. For example the Resource Manager is used to get non-code resources like layout files, drawables, strings, and colors.

The final layer is made of the System Applications. Here are the core applications of any Android device, for example the Telephone application, Agenda, SMS, Camera, etc. Those applications come pre-installed to any android smartphone.

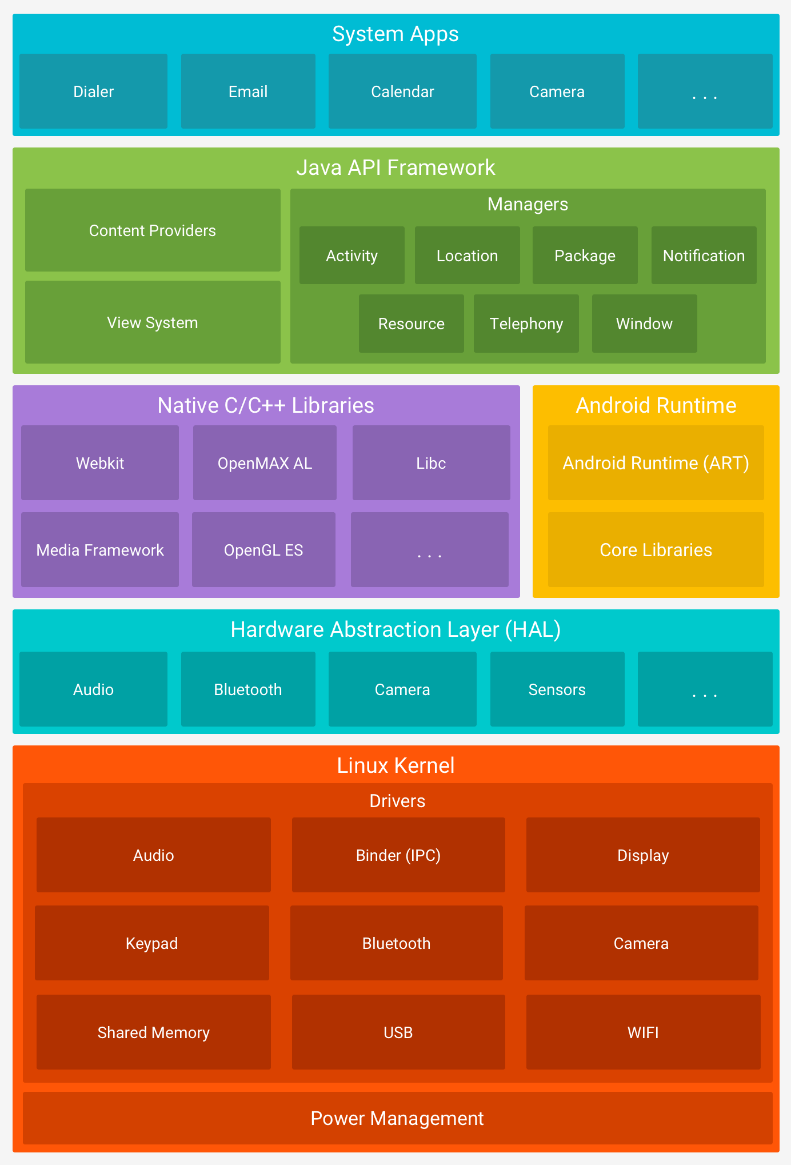


Figure ‑ Android stack

### Spring Boot

Spring Boot is a Java framework developed by Pivotal Team, which help the developers to build micro services. Spring Boot speeds up the process of building a micro service, by incorporating a servlet container like Tomcat, Jetty or Undertow, and being pre-configured and ready to run. Also, it provides mappings to create REST endpoints, easing the process of creating a Java REST API. It also provides well-written documentation, with code snippets, which guides the developer on how to create “Hello, World” applications. [30]

### IntelliJ IDEA

IntelliJ is an in integrated development environment developed by JetBrains, used to developed Java applications mainly. Beside Java, we can also develop applications in Kotlin, Groovy or Scala as all these programming languages uses the Java Virtual Machine and the code is compiled to Java Bytecode. It is written in Java and Kotlin and it is available for Windows, Linux and MacOS. [31]

IntelliJ has been released in January 2001 and currently there are two version available: Ultimate and Community. Community is the free version, with an Apache License 2.0, used to develop mainly Java applications, with less support and less frameworks available. The ultimate version is focused more for enterprise and web applications [32]. In my case, I used IntelliJ only to create a backend Java server, where I am exposing REST services using Spring Boot. IntelliJ community edition fulfilled my needs, and I was able to realize what I purposed with it.

The main advantages of using IntelliJ are the design, the easiness of writing code using the Generate functionality and Smart code completion, the ability to navigate through the hierarchy of classes, the built-in version control system(VCS) and many more functionalities and tools available.

### Android Studio

Android Studio is an integrated development environment, build by Google, with the purpose of easing the process of developing Android Applications. It is mainly based on IntelliJ IDEA made by JetBrains, with modifications and plugins which are used in Android Development. Android Studio has been released by Google on 16 May 2013. It is written in Java, C++ and Kotlin and it is available on Windows, Linux and MacOS.

Android Studio supports the same programming languages as IntelliJ IDEA, but from October 2017, it also supports the programming language developed by JetBrains, Kotlin. From 2019, this is becoming the default Android programming language, replacing Java. The main advantage of Kotlin is that it is fully interoperable with Java.

One of the main advantages of using Android Studio is the Android Virtual Device. This is a well optimized emulator which can simulate an Android smartphone on which we can install our application for testing reasons or debugging. Another important feature of Android Studio is the Layout Editor where we can view the layout in different screen configurations, we can drag and drop UI components offered by Android, and we can easily modify the XML layout files seeing the changes in real-time. Another difference from IntelliJ is that Android Studio has Gradle built in, and we can easily import any library we need from Gradle or Maven.

Summing up the advantages enumerated above with the fact that this application is updated periodically, and I observed great improvements for the emulator, I think Android Studio is the best choice to develop Android Applications. [30]

### MSSQL

Microsoft SQL Server is a management system for relational databases. It was released on 24 April 1989 by Microsoft. [33] It is mainly used to store and retrieve data, using Microsoft’s own version of Structured Query Language (SQL). [34]

In order to connect and access the Microsoft SQL server from Java, we need to use Microsoft’s Java Database Connectivity API(JDBC) driver.

### Room Database (SQLite)

Room database is a component of the Android Jetpack, meant to simplify the development process of an Android Application, launched by Google in 2017. It is an abstraction layer between our application and an SQLite database, helping programmers to avoid boilerplate code. As Room is an Object Relational Mapping (ORM) library, we can easily map Java Objects to SQLite Objects. Another advantage using Room, is that the query errors are detected at compile time.

Room Database is developed to work with LiveData. Using Model-View-ViewModel in combination with Room Database, assures the data is observable directly in the database. This means that every change in the database will reflect in the UI (user interface). Also, it saves a lot of time, avoiding writing a lot of boilerplate code to achieve the same objectives, and most importantly, it maintains a clear structure and an organized code. [35]

In order to communicate with the Room Database, we need to define Data Access Objects and Entities. An Entity is a java class which represent the mapping between the Java object and a table row. In the entity we define the fields and its types of the object, create a constructor and getters and setters. With androidx library, we can define the table name, set primary keys, foreign keys, indexes, and map every member of the class to a column.

The Data Access Object is an interface where, using androidx library, we can define the queries to the database using annotations. The advantage of Room is that, at the compile time, we can see if we have any errors in the query written in the annotations. With those queries we can retrieve, insert, update, or delete or even observe data from the database.

We can build a new Room Database for our application by creating a class which has to extend the Room Database class and adding the annotation @Database above the class declaration. In this class we can also define the way to make Migration in case of the schema is changed.

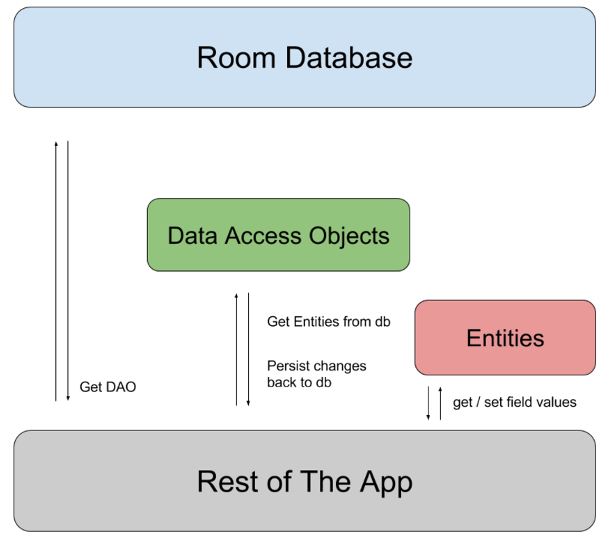


Figure ‑ Components of Room Database (reference: [35])

### Room Explorer

Room Explorer is a 3rd party library taken from GitHub, used mainly to view the data from our application’s Room Database. This library allows us to start an activity through which we can view, insert, delete, or update the content of our tables. We are even able to manually write and execute queries using this explorer.

This library is very useful to use for debugging, to see if data is missing when actions are performed, etc. This library has an Apache License 2.0. [37]

### JSON

### GSON

### OAuth

### OkHttp

### Open Banking Project API

### Google Sign-In API

### Google Biometrics

Google Biometrics is an API available on Android Jetpack. Using this API, we can easily access and use the biometric hardware the phone is providing, in order to validate the user. We can restrict the access to our application, which significantly improves the security.

We can use BiometricManager class in order to find if there is any biometric available on the device. BiometricManager is the successor class of the FingerprintManager [38]. FingerprintManager was released in Android 9, but it was replaced in Android 10 by BiometricManager, as the new biometrics hardware technologies appeared in the smartphone industry. If this validation is successful, and there is a valid or multiple biometrics available on the device, we can forward use the BiometricPrompt class. With BiometricPrompt class, we can show the dialog screen which request biometrics in order to make the authentication, we can validate the authentication, and we can implement follow-ups in case of different authentication statuses. In the Figure 3-6 we can see the architecture of BiometricPrompt and the authentication flow. [39]

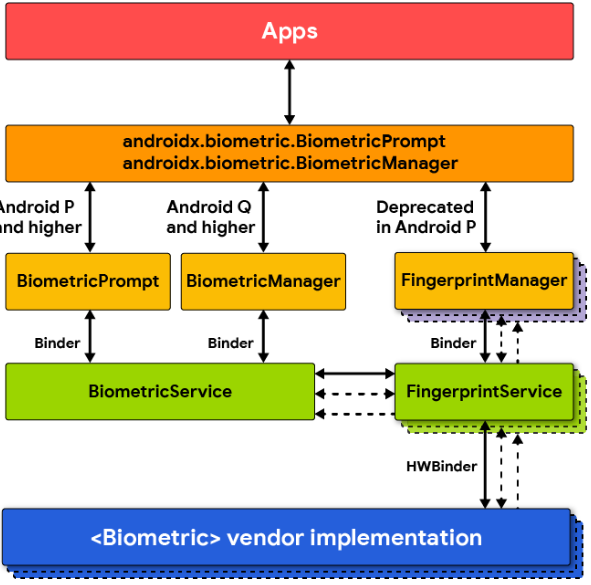


Figure ‑ BiometricPrompt architecture (reference: [39])

# Solution Development

## Requirement specifications

## Solution architecture

## Detailed design

## Implementation

## Testing

# User Guide and Experimental Results

## Screens

## Flows

## Experimental Results

# Conclusions

# References