E-Commerce Mobile Application

Final Year Project
B.Sc.(Hons) in Software Development

BY
Darren Regan

May 1, 2020

Advised by Martin Hynes
Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

Chapter 1

Introduction

My project is a Native Android E-Commerce Application built in Android Studio using Kotlin/Java programming languages. The general concept of my application is to create a e-commerce app that mirrors Amazon or Alibaba.
A specific goal of this project was to learn how native development differs from hybrid development(Xamarin, Ionic, React Native etc). Hybrid development was a topic that had multiple course projects, but i never got to experience building a project based on native development in Android or iOS.

A focus i made on my application is integrating different technologies such as Firebase, Picasso and Retrofit. Using these technologies made certain aspects of my application easier to implement. For example Picasso which is a image downloading and caching library can manage many aspects of image processing in an android environment such as handling ImageView recycling, complex image transformations with minimal memory use and automatic memory and disk caching which all have a heavy amount of coding involved to be implemented manually.

Firebase is another technology i use which has a great number of features and products such as Cloud Firestore which stores and syncs data between users and devices - at global scale - using a cloud-hosted, NoSQL database.
Authentication to manage users in a secure way by offering authentication through email, password and third-party providers like GitHub, Google, Facebook and Twitter.
Realtime Database which is an efficient, low-latency solution for mobile apps that require synced states across clients in realtime.

1.0.1 Objectives for project

Use a new programming language and framework
One of the objectives i had set for this project is to use a new language and framework, i made the choice of android and kotlin as it met all my objectives for my project. The main goal of my project was to build a native mobile application and compare it to hybrid applications that i have built throughout my course.

Create a Native Mobile Application

Use new and useful technologies ss
Include social Media Integration ss

Create a Native Mobile Application

1.0.2 Sections

Methodology After setting objectives for my project, i set out to make a plan for development using Kanban boards. I used an agile approach for my development setting specific goals for each week. Validation and Testing was done using Junit.

Technology Review In this section i explain the Technologies used in my project, technologies such as Firebase, Kotlin, Android Studio are explained in great detail. I will also review programming in android with kotlin compared to my experience through out my four years in college. Comparisons such as Native development vs Hybrid development, use cases for different programming languages, language features such as more use of lambda expressions and more Kotlin features that make it a great alternative to java for android development.

System Design In this section i provided an explanation of the overall system architecture using UML class, sequence and interaction diagrams as well as screenshots of UI components such as how each view is formed with ImageView, TextBox etc.

System Evaluation In this section i evaluate my project against the objectives i set out.

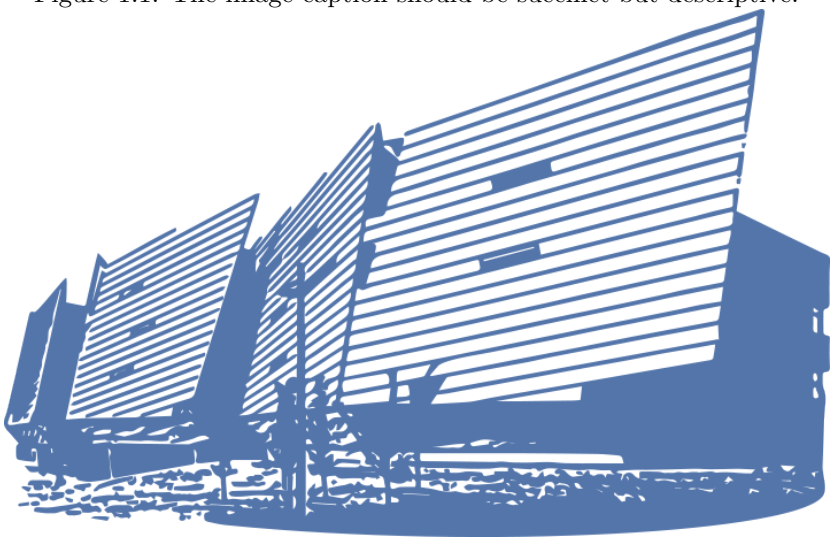
Conclusion
In my conclusion i evaluate my overall goals for my project and list outcomes of the project. I go over discoveries made, what I've learned and what i can improve on in the future

Github Repository `https://github.com/DarrenRegan/Final-Year-Project`

My GitHub Repository contains my Dissertation, APK file which are both available for download with a click of a button at the top of the README.

- The README contains a quick explanation of the project as well as an installation guide, Devices used in testing as well as resources used along with links to research material.
- The code for my project is located at Final-Year-Project/app/src/main/ Java folder contains the code for activities and models
- Res folder contains the code for UI elements of the project, main/res/layout contains XML for all activities

Figure 1.1: The image caption should be succinct but descriptive.



Chapter 2

Methodology

This chapter covers the various methodologies that were implemented in this project, this includes Research methodologies, Software development methodologies, project management, supervisor meetings, developments tools, testing and source control.

An overview of methodologies used in this project; **Software development methodologies** which includes Agile Development, Continuous Delivery, Test Driven Development, Feature Driven Development, Extreme Programming etc. **Project management** which includes GitHub Kanban board and supervisor meetings. **Development tools** Android studio, Firebase. **Source Control** GitHub, Overleaf, Badge/Shield

2.1 Project management

Project Management is a key element of any task to ensure that the project is laid out correctly and each component party is complete at a given date. Because this project is a solo project i used the GitHub Kanban board to first pick features that i may want to implement and then broke them down into separate cards that i could work on individually. This helped brake down tasks which helped in development.

2.1.1 Agile

This project used Agile project management methodologies. An iterative approach was taken. This meant that each week i had certain tasks laid out to be completed be that either a feature for the app or documentation. I also had short weekly/bi-weekly meetings with my supervisor where i would update him, and asks for advice on my dissertation, app, etc.

Agile Road-map

An agile road-map in agile development helps state the goals of the project from the outset. It helps break down the tasks which are then further broken down for sprints. This is usually a high level view of the project. It is used often in industry and it helps to understand a project as a whole and learn what other teams are contributing to the overall project. The roadmap also shows how the project can grow in the future.

Planning and Development

During the meeting and planning phase, the milestones & objectives for the project were identified and broken down into simpler, and more manageable

tasks. The tasks are were then grouped into sprints or iterations lasting one to three weeks depending on the task. These are the tasks which mostly completed before each meeting with my supervisor. With each meeting i had a task somewhat completed and i could then know the following task which i could talk with my supervisor about before hand. So each weekly meeting with the supervisor started a sprint and the aim was to complete most of that sprint before the next meeting. The plans for the project and the sprints are taken from the User stories i created on the Kanban project and Design document. Through this iteration i convert the plans into working code.

Design Phase

Throughout the agile development life cycle design is a step in every iteration/sprint. The design is gradually built on. The design is never defined at the beginning of the project. The gradual evolution of the design enables taking advantages of new technologies that come on stream as well as meeting any new requirements brought forward by clients. The user experience is key when design a project/application. Every design must be user friendly. The type of end user of the item must be consider when designing. This comes to play in android development often as there is many different tools in development to create different features. You need to test them to know whats best for the end product.

Testing

Testing was carried out continuously with Android Studio features. Not only those Android Studio have a huge amount of features the Emulator can test a massive array of devices. So testing for this application was a continuous process throughout every build.
In a real-world environment The user tests would be carried out when meeting with the client. The client using the application would then comment on any changes that should be or could be made.

2.2 System Integration

Producing this project as a solo person meant i used different Software Engineering techniques to development my app. The main approaches used are test driven development(TDD), feature driven development(FDD), and continuous delivery

2.2.1 Test Driven Development

Test Driven Development is based predominately on unit testing. However they are very different because test driven development is when the unit tests are written before the code has been created. This means the developers are using the unit tests as a gild and the tests alone break down the overall task needing

to be completed. The tests will initially fail and the developers write minimal code in order for the tests to pass. This process is continuously repeated in order for the application to pass each test. These tests are commonly automated, in Android Studio a huge amount of testing is automatically completed for you without any manual interference.

2.2.2 Feature Driven Development

Feature Driven Development (FDD) is an agile framework that, as its name suggests, organizes software development around making progress on features. Features in the FDD context, though, are not necessarily product features in the commonly understood sense. They are, rather, more akin to user stories in Scrum. In other words, "complete the login process" might be considered a feature in the Feature Driven Development (FDD) methodology.[9]

FDD planning

CP prepares iteration planning meeting by grouping the right amount of relevant features into a work package. Normally the team has a number of CPs, each of them conducts iterations independently. Each iteration does not have to include the whole 10 development team. Instead, the new iteration team is formed for each iteration. CP selects iteration team members based on availability. Each developer receives a subset of features; strong class ownership is encouraged. The iteration team goes through the list of features, which should be familiar from Process One. If needed, the team contacts the customer to clarify any obscure features. In the complex cases the team may create sequence diagrams. The planning meeting also sets timeline for iteration milestones.[10]

Lifecycle of FDD

1. **Develop an overall model** - FDD project starts with high-level walk through of the scope of the system and its context. Next, detailed domain models are created for each modelling area by small groups and presented for peer review.

2. **Build a features list** - Knowledge gathered during the initial modeling is used to identify a list of features by functionally decomposing the domain into subject areas. Subject areas each contain business activities, and the steps within each business activity form the basis for a categorized feature list.

3. **Plan by feature** - After the feature list is completed, the next step is to produce the development plan and assign ownership of features (or feature sets) as classes to programmers.

4. **Design by feature** - A design package is produced for each feature. A chief programmer selects a small group of features that are to be developed within two weeks. Together with the corresponding class owners, the chief programmer works out detailed sequence diagrams for each feature and refines the overall model. Next, the class and method prologues are written and finally a design inspection is held.

5. **Build by feature** - After a successful design inspection for each activity to produce a feature is planned, the class owners develop code for their classes. After unit testing and successful code inspection, the completed feature is promoted to the main build.

Coding in FDD

Coding process in FDD is not as exciting and challenging as it is in XP. This happens because by the coding time the features have been extensively discussed during Process One, iteration kick-off meeting, design review meeting. Classes and methods are 13 defined by now, their purpose is described in code documentation. Coding often becomes a mechanical process. Unlike XP FDD strongly discourages refactoring. The main argument against refactoring here is that it takes time and does not bring any value to the customer. The quality of code is addressed during code review meetings. FDD encourages strong code ownership. The main idea is that every developer knows the owned code and better realizes the consequence of changes.[10] FDD fights the problem of leaving team members from the different angle:

- Sufficient code documentation simplifies understanding somebody else's code
- Developers know what other people's code does, since they reviewed the design
- Developers will look at each other's code during code review

2.2.3 Continuous Delivery

Continuous delivery is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button. In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem. [8]

Continuous Delivery in an Android Environment

In the context of mobile development, Continuous Delivery is the ability to produce valid builds that are ready to be published in an app store. Continuous Deployment is the final status of the process, which ensures that any of the changes applied to the code is automatically deployed to the production environment.

Continuous Delivery for mobile development can be tricky. For Android app development requires a key to sign off the production build before it can move to the Google Play Store. Keeping keys and passwords on a Version Control Software (VCS) is a bad practise, because VCS is available to any member with access to the project, which includes any temporary 3rd parties. It's common practise therefore to create release build on a local computer, but this erodes one of the great advantages to Continuous delivery: having a scalable online platform and the ability to easily deploy any stages of the Continuous Integration at any time.[2]

Because of this you need to secure the signing keys of the app, building a Continuous Delivery around a mobile development environment can be tricky. For Android app development, you can improve this by implementing Google Play App Signing, which applies a second key when your app is uploaded to the Google Play Store. But even after employing this method, there will still be at least a key and a couple passwords required to build a release (signing key, key and upload password). To get past this you can use a Encrypted Cloud Storage service for this problem. They can keep the keys safe and are only limited by the integration capabilities of your chosen Continuous Delivery system.

Implementing Continuous Delivery

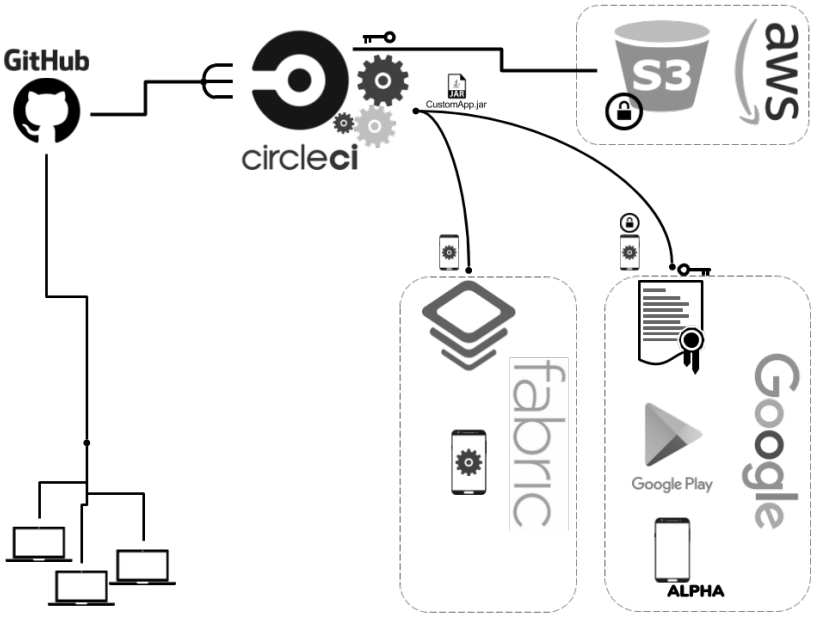
Implementing Continuous Delivery, it requires you pick the right tools for the job. So choosing your Continuous Integration system is important. For this project because i was solo i could just use my local computer for deployment and tool integration inside Android Studio is so good, i didn't need to implement these tools, but in a real-world environment working with a team this implementation is mandatory

Some examples of Continuous integration systems:

- GitLab CI
- CircleCI
- CodeShip
- Github Actions
- AWS Pipeline

Using these systems with a Version Control System such as GitHub is an easy feat to accomplish and results in a final system as shown below

Figure 2.1: Continuous Delivery Setup for an Android App [2]



2.3 Development tools

In order to manage this project i used different development tools and Source Control tools. This was very useful as it enables working on different branches for features that i want to implement and leaves the master branch as a working state of the project. This could be used for industry standard as continuous delivery. For this project it's meant that after each sprint there was a working version to present to the customer. Although the working version does not mean it doesn't need changes in next sprints. I was solo this project so i did not take advantage of branches as much as a group would, as each member would have there own branch to work on.

2.3.1 Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.
The integration between Android Studio and Firebase made Android Studio the most superior development platform to use.

2.3.2 Firebase

Firebase is Google's mobile and web application development platform that helps you build, improve, and grow your application. Firebase frees developers to focus on crafting excellent user experiences. You don't need to manage servers. You don't need to write APIs. Firebase is your server, your API and your database, everything is written generically so that you can modify every-thing to suit most needs.

2.4 Source Control

2.4.1 GitHub

Github provides hosting for software development version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. GitHub offers plans free of charge, and professional and enterprise accounts. The Github kanban board was a feature i used in this project to track sprints.

2.4.2 Overleaf

Overleaf is an open-source online real-time collaborate LaTeX editor. It is hosted at http://www.overleaf.com, but can also run on your own local version, and contribute to the development of Overleaf.

2.4.3 Badge/Shield

Badge/Shield.io is a service for concise, consistent, and legible badges in SVG and raster format, which can easily be included in GitHub READMEs or any other web page. The service supports dozens of continuous integration services, package registries, distributions, app stores, social networks, code coverage services, and code analysis services. Every month it serves over 470 million images. [5] This is used for easy access to a copy of the APK and Dissertation at the top of my README, you can further customize these badges to show Beta versions, stable versions, downloads, ratings, code coverage etc.

Chapter 3

Technology Review

This chapter discusses the different technologies used in throughout the project. It discusses the the advantages and disadvantages of each technology and why certain technologies were used over others. It also discusses hybrid applications compared to native applications, advantages, disadvantages, uses at different business structures and other topics.

3.1 Overview

This project is a Native android app built with Kotlin in Android Studio, Fire-base is used for the database, statistics, verification etc.
Topics:

- Kotlin/Java comparison
- Fire-base
- Picasso
- Android Studio/IntelliJ
- Native Applications
- Hybrid Applications
- Hybrid vs Native comparison

3.2 Main Technologies

This section will discuss the main technologies currently in use in the android application.

3.2.1 Kotlin

Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to inter-operate fully with Java, and the JVM version of its standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript or native code (via LLVM). Language development costs are borne by JetBrains, while the Kotlin Foundation protects the Kotlin trademark.
Kotlin is the preferred language for Android app developers as of May 2019, since the release of Android Studio 3.0 in October 2017, Kotlin has been included as an alternative to the standard Java compiler. The Android Kotlin compiler targets Java 6 by default, and lets programmers choose between Java 8 to 14 for optimization purposes.
Kotlin originated at JetBrains, which is the company behind IntelliJ IDEA. Kotlin has been open source since 2012 and has a large team of full-time developers working on it, there is also the Kotlin project of GitHub which has more than 370 contributors.

Advantages

Kotlin has many advantages, many are quite serious improvements in readability and workflow which was noticeable when creating my project

- **Less code combined with greater readability** - Spend less time writing code and working to understand the code of others.
- **Mature language and environment** - Kotlin has developed continuously over the years not only as a language but as a whole ecosystem with very robust tooling. Its seamless integration with Android Studio, makes it actively used by companies to develop Android applications.
- **Kotlin support of Android Jetpack and other libraries** - KTX extensions adds kotlin language features, such as coroutines, extension functions, lambdas, and named parameters, to existing Android libraries.
- **Interoperability with Java** - You can use Kotlin along with the Java programming language in your applications without needing to migrate all your code to Kotlin.
- **Support for multi-platform development** - You can use Kotlin for developing not only Android but also iOS, back-end, and web applications by sharing the common code among the platforms.
- **Code safety** - Less code and better readability lead to fewer errors. The Kotlin compiler detects the remaining errors, making the code safe.
- **Easy to Learn** - Kotlin is very easy to learn, especially for any Java experienced developers.
- **Large community** - Kotlin a great support and many contributions from the community, which is growing all over the world. According to Google, over 60% of the top 100 apps on the Google Play Store use Kotlin. Many startups and Fortune 500 companies have already developed Android applications using Kotlin and more and more companies are prioritizing Kotlin Native application development over other options due to the robust toolkit and optimizations that make your applications the best that they can be.

Disadvantages

- **Shift from Java to Kotlin** - Kotlin is an amazing programming language and there is a reason why leading lead companies have started using kotlin, but at their core their two different languages. Developers won't be able to quickly shift from one to another without taking time to learn Kotlin. Therefore company's have to consider different approaches to Android app development as additional expenses are required on training a team of developers.

- **Hard to find experienced developers** - There is a high demand for specialists in Kotlin as Google made it the preferred language for Android development in 2019, but there is still a very large amount of Java programmers on the market compared to Kotlin developers. This means on average the Kotlin developers may be younger meaning less senior developers available for hire. This is quite a large disadvantage, but will quickly fade away as many leading tech companies have switched which creates a ripple effect down the chain of companies.

- **Limited learning resources** - Although the number of Android app developers who use Kotlin instead of Java increase everyday, there is still a limited number of resources in the market compared to Java. Many College courses will teach Java over Kotlin as both are so similar, meaning most Kotlin developers come from a background in Java and learn to code in Kotlin themselves.

Kotlin Syntax

Kotlin syntax is familiar to any programmer that is from a OOP domain and be be more or less understood from the get-go. There are differences from Java such as primary and secondary constructors, val & var variable declarations and more.

Below you can see the basic structure of a class in kotlin

```
class Foo {

    val b: String = "b"      // val means unmodifiable
    var i: Int = 0           // var means modifiable

    fun hello() {
        val str = "Hello"
        print("$str World")
    }

    fun sum(x: Int, y: Int): Int {
        return x + y
    }

    fun maxOf(a: Float, b: Float) = if (a > b) a else b

}
```

String Interpolation - A smarter and more readable version of Java's String.format() that is built into the language

```
val x = 5
val y = 10
print("sum of $x and $y is ${x + y}")  // sum of 5 and 10 is 15
```

Type Inference - Kotlin will infer your types wherever you feel it will improve readability

```
val a = "abc"                          // type inferred to String
val b = 4                              // type inferred to Int

val c: Double = 0.7                    // type declared explicitly
val d: List<String> = ArrayList()      // type declared explicitly
```

Smart Casts - The Kotlin compiler tracks logic and auto-casts types if possible, which means you do not need to use instanceof checks followed by explicit casts

```
if (obj is String) {
    print(obj.toUpperCase())      // obj is now known to be a String
}
```

When Expression - The switch case is replaced with the more readable and flexible when() expression

```
when (x) {
    1 -> print("x is 1")
    2 -> print("x is 2")
    3, 4 -> print("x is 3 or 4")
    in 5..10 -> print("x is 5, 6, 7, 8, 9, or 10")
    else -> print("x is out of range")
}

// It also works as an expression or a statement
// with or without an argument
val res: Boolean = when {
    obj == null -> false
    obj is String -> true
    else -> throw IllegalStateException()
}
```

Setter & Getter behavior - You can make custom set & get behaviors that are added to public fields, which means getter & setters won't bloat your code

```
class Frame {
    var width: Int = 800
    var height: Int = 600

    val pixels: Int
        get() = width * height
}
```

Data Classes - We frequently create classes whose main purpose is to hold data. In such a class some standard functionality and utility functions are often mechanically derivable from the data. In kotlin, this is called a data class and is marked as data. Its a Plain Old Java Object so its complete with toString(), equals(), hashCode() and copy().

```
data class Person(val name: String,
                  var email: String,
                  var age: Int)

val john = Person("John", "john@gmail.com", 112)
}
```

3.2.2 Kotlin vs Java

blah..

Advantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

Disadvantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

3.2.3 Firebase

Firebase is Google's mobile and web application development platform that helps you build, improve, and grow your application. Firebase frees developers to focus on crafting excellent user experiences. You don't need to manage servers. You don't need to write APIs. Firebase is your server, your API and your database, everything is written generically so that you can modify everything to suit most needs. Firebase has a huge amount of features, real time databases, cloud storage, hosting, machine learning, authentication, statistics, analytics and more.

Firebase products are setup in three different area's.

1. **Build better apps**
2. **Improve app quality**
3. **Grow your business**

Build better apps

Firebase lets you build more powerful, secure and scalable apps, using world-class infrastructure. There are seven different products focused on building a better app. My project takes advantage of Authenication, Realtime Database and Cloud storage. Almost all of Firebase products are extremely useful and are worth mentioning as they could be implemented into the project at some point. I will first summarize the products, and then go into further detail on the specific products i used in my project. How the code is implemented, why i used it etc.

Products for building better apps

- **Cloud Firestore** - Store and sync data between users and devices - at global scale - using a cloud-hosted, NOSQL database. Cloud Firestore gives you live synchronization and offline support along with efficient data queries. Its integration with other Firebase products enables you to build truly serverless apps.

- **ML Kit** - Bring powerful machine learning features to your mobile app whether you're new or experienced in ML. Get started easily by using our ready-to-use APIs for common mobile use cases, or import your own custom models which can be hosted and served to your apps by Firebase. ML Kit APIs can run on-device or in the cloud, depending on the functionality, and some give you both choices.

- **Cloud Functions** - Extend your app with custom backend code without needing to manage and scale your own servers. Functions can be triggered by events, which are emitted by Firebase products, Google Cloud services, or third parties, using webhook

- **Authentication** - Manage your users in a simple and secure way. Firebase Auth offers multiple methods to authenticate, including email and password, third-party providers like Google or Facebook, and using your existing account system directly. Build your own interface, or take advantage of our open source, fully customizable UI.

- **Hosting** - Simplify your web hosting with tools made specifically for modern web apps. When you upload your web assets, we automatically push them out to our global CDN and give them a free SSL certificate so your users get a secure, reliable, low-latency experience, no matter where they are.

- **Cloud Storage** - Store and share user-generated content like images, audio, and video with powerful, simple, and cost-effective object storage built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

- **Realtime Database** - Realtime Database is Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime. We recommend Cloud Firestore instead of Realtime Database for most developers starting a new project.

Improve app quality

Firebase gives you insights into app performance and stability, so you can channel your resources effectively. These products weren't used in the project, but are worth mentioning as they are quite valuable in a commercial development environment where app performance and crashing have a huge impact on user engagement and app performance on the Google Play Store.

Products for improving app quality

- **Crashlytics** - Reduce your troubleshooting time by turning an avalanche of crashes into a manageable list of issues. Get clear, actionable insight into which issues to tackle first by seeing the user impact right in the Crashlytics dashboard. Realtime alerts will help you stay on top of stability even on the go. Crashlytics is the primary crash reporter for Firebase.
- **Performance Monitoring** - Diagnose app performance issues occurring on your users' devices. Use traces to monitor the performance of specific parts of your app and see a summarized view in the Firebase console. Stay on top of your app's start-up time and monitor HTTP requests without writing any code.
- **Test Lab** - Run automatic and customized tests for your app on virtual and physical devices hosted by Google. Use Firebase Test Lab throughout your development lifecycle to discover bugs and inconsistencies so that you can offer up a great experience on a wide variety of devices.
- **App Distribution** - Firebase App Distribution allows developers to send pre-release versions of their app to trusted testers from the console or using command line tools, as well as manage testers in one place.

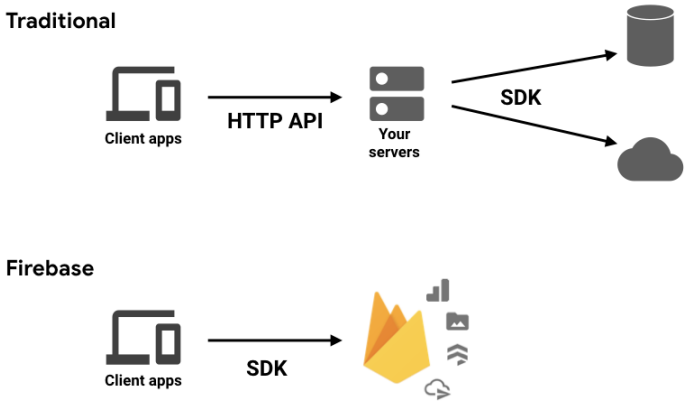
Grow your business

Firebase helps you grow to millions of users, simplifying user engagement and retention.

Products for Growing your business

- **In-App Messaging** - Engage and nurture your active users with targeted and contextual messages that encourage them to complete meaningful actions within your app. You have the power to trigger messages based on user behavior and interests. You can also customize the design of in-app messages to fit your brand. In-App Messaging supports a variety of use cases and formats.
- **Google Analytics** - Analyze user attributions and behavior in a single dashboard to make informed decisions on your product roadmap. Gain realtime insights from reports, or export your raw event data to Google BigQuery for custom analysis.
- **Predictions** - Harness the power of Google's machine learning to get insight into which segments of users are likely to churn or spend (or complete another conversion event). Use these smart predictive segments for targeting in other products like Remote Config, Cloud Messaging, and In-App Messaging.
- **A/B Testing** - Improve your app by running product and marketing experiments, without worrying about setting up the infrastructure to run A/B tests. Customize experiments to suit your goals. Test a variety of updates to your app, like message copy or new features. Then, only roll-out changes proven to move the needle on your key metrics.
- **Cloud Messaging** - Send messages and notifications to users across platforms—Android, iOS, and the web—for free. Messages can be sent to single devices, groups of devices, or specific topics or user segments. Firebase Cloud Messaging (FCM) scales to even the largest apps, delivering hundreds of billions of messages per day.
- **Remote Config** - Customize how your app renders for each user. Change the look and feel, roll out features gradually, run A/B tests, deliver customized content to certain users, or make other updates without deploying a new version—all from the Firebase console. Monitor the impact of your changes and make adjustments in a matter of minutes.
- **Dynamic Links** - Use Dynamic Links to deliver a customized user experience for iOS, Android, and the web. You can use them to power mobile web to drive native app conversions, user to user sharing, social and marketing campaigns, and more. Dynamic Links provides you with the attributions you need to better understand your mobile growth.

Traditional Architecture vs. Serverless Architecture

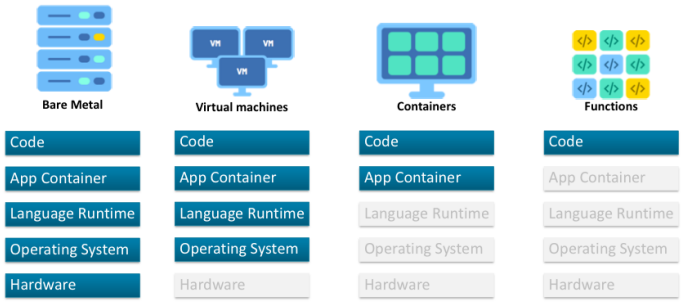


Firebase and other serverless architectures have rapidly emerged as a new technology concept in recent years. Using this Architecture, developers can create a variety of applications for various industries. Many enterprises have already started adopting serverless products, serverless architecture provides computation-light, highly-flexible, stateless applications and more. Developers are constantly on the lookout for more effective ways to maintain the software development lifecycle, doing so introduces new technologies that are accompanied with an increase in productivity.
Serverless architecture was introduced to help businesses focus on application development. With serverless, businesses no longer have to worry about server infrastructure, reducing development costs and shortening the development cycle.

A traditional architecture approach is frequently a large computer or cluster of computers that is accessed over the internet to provide access to information or servers. The machines are commonly located at specific area's around the world depending the size of the company. A traditional approach requires a team to manage these servers that which starts to become a problem what different scales of companies. As you get towards Enterprise sized servers they can become increasingly expensive. Not only do they cost Hundreds of Thousands to millions, you need to have space to house them, a staff of engineers to maintain them. In addition, the average life cycle of a given server is about 5 years. Which means not only do you have to potentially replace your machines, but in the interim you are constrained by the limitations of that machine.
An example of these would be if a websites server does not have the capacity to handle it's increased load, the only traditional solution is to purchase more servers to handle extra volume, or replacing existing servers with a better model. Both have exceptionally high cost and come with large downsides.

Serverless Architectures aims to solve these problems. Rather than having one machine to host a job, you can now utilize servers, such as Firebase, Amazon AWS and other cloud hosted databases. All have a huge array of additional features that you opt into to spec out a server that fits your needs and removes the additional costs and headaches of a traditional server.

Figure 3.1: Serverless Evolution



Advantages of Serverless

Advantages Serverless architectures has are the following:

1. Providers scale and manages the required resources
2. Rapid provision of resources in real-time, even for unforeseen peak loads and disproportionate growth
3. Highly scalable and flexible architecture
4. Users only need to pay for the resources they use
5. High error tolerance thanks to flexible hardware infrastructure in the provider's computer centers

Disadvantages of Serverless

Disadvantages Serverless architectures has are the following:

1. No access to virtual machines, operating system or runtime environments
2. Implementing serverless structures is very labor-intensive
3. Lock-in effect – for example, when changing provider, you generally have to recode all event-based functions
4. Relatively complex monitoring and debugging process, as in-depth performance, and error analyses are generally not possible

As shown Firebase is an excellent option for any developer interested in creating a mobile application, finally i will go over the Pros and Cons of choosing Firebase

The Pros of Firebase

- **Databases** - Depending on your budget, Google offers robust databases to use with your apps. Both realtime and Firestore databases can be scaled in terms of size, suggesting a fully secure managed solution, that still provides you easy access to your data via firebase console. Data updates and offline access makes databases usable for real time application, as well as keeping multiple databases in sync

- **Wide selection of products** - Firebase suggest a lot of products to make your application work. You can choose between realtime databases and firestore, store data in the cloud and build serverless applications with the integrated Cloud functions.

- **Free to use for small developers** - Firebase pricing[16] is free as you start with it. This will allow you to understand whether it fits your application and understand all the peculiarities. Once you reach certain amount of database memory or need a specific product, you can choose a different plan around that product or memory you need. You can do this by using the Blaze plan calculator provided on firebase pricing website[16]

- **Excellent Documentation** - The whole firebase platform is extremely well documented. Good technical documentation, API documentation, SDK references, which makes all the products easier to use and accessible for the user. The firebase products page contains all the required information concerning the integration's, available platforms, guidance's, docs, guides and lists supported technologies.

- **Accessible UI and ease of Integration** - Firebase requires minimal programming language knowledge, and suggests integration's via its user interface, it also is integrated into Android Studio which further increases ease of use. While eliminating the need for complex configurations, anyone can set up the application.

- **Google products** - Firebase comes with a Content Delivery Network (CDN) in-built with Google Cloud platform, while also having integration with all google products. A product owned by google also has the advantage of unlikely hood of the company collapsing or not supporting the product frequently.

The Cons of Firebase

The cons of firebase are limited and depends on the scope of you project and the size of your company. Some of the cons are currently being fixed through new features that are currently in Beta. For example Firebase Realtime Database, which i used in this project has limitations of not being able to do complex query's and Data Modeling. Google aims to fix this with its new product Firestore.

- **Firebase Realtime Database limitations** - Realtime Database is used as the main storage for my project, which has cons. One of the mains problems with it, is the limited querying capabilities. Realtime database provides no way to filter capabilities, because the whole Database is a huge JSON file, which makes it pretty difficult to make complex queries.
- **Data Modeling** - Firebase Realtime Database and it data modeling as a problem where because of "Database as a single JSON file" structure,m you can't implement relations between data items.
- **Locked into a vendor** - This is a wide problem with BaaS solutions in general, not having the ability to migrate data to another platform can be considered a con.
- **Support for iOS** - While firebase is a cross-platform product, it concentrates more on Android mobile platform being that its a google product. Android Studio easily integrates all Firebase products such as Test Lab, Auth etc. While iOS devices are lacking behind on such integration's.

3.2.4 Picasso

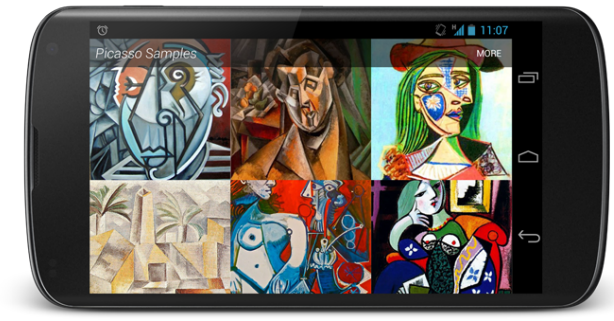
Picasso is a powerful image downloading and caching library, it is widely used and requires very little code to implement. This is used because working with images in android is difficult, you need to work with network requests, caching, background threads and decoding & encoding of image which uses a lot of memory. Picasso hides all this and adds additional features like resizing images while being memory efficient.

Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!

```
Picasso.get().load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

Many common pitfalls of image loading on Android are handled automatically by Picasso:

- Handling ImageView recycling and download cancellation in an adapter.
- Complex image transformations with minimal memory use.
- Automatic memory and disk caching.



Features

ADAPTER DOWNLOADS

Adapter re-use is automatically detected and the previous download canceled.

```
@Override public void getView(int position, View convertView, ViewGroup parent)
  SquaredImageView view = (SquaredImageView) convertView;
  if (view == null) {
    view = new SquaredImageView(context);
  }
  String url = getItem(position);

  Picasso.get().load(url).into(view);
}
```

IMAGE TRANSFORMATIONS

Transform images to better fit into layouts and to reduce memory size.

```
Picasso.get()
  .load(url)
  .resize(50, 50)
  .centerCrop()
  .into(imageView)
```

Can also specify custom transformations for more advanced effects.

```
public class CropSquareTransformation implements Transformation {
  @Override public Bitmap transform(Bitmap source) {
    int size = Math.min(source.getWidth(), source.getHeight());
    int x = (source.getWidth() - size) / 2;
    int y = (source.getHeight() - size) / 2;
    Bitmap result = Bitmap.createBitmap(source, x, y, size, size);
    if (result != source) {
      source.recycle();
    }
    return result;
  }

  @Override public String key() { return "square()"; }
}
```

Pass an instance of this class to the transform method.

PLACE HOLDERS

Picasso supports both download and error placeholders as optional features.

```
Picasso.get()
    .load(url)
    .placeholder(R.drawable.user_placeholder)
    .error(R.drawable.user_placeholder_error)
    .into(imageView);
```

A request will be retried three times before the error placeholder is shown.

RESOURCE LOADING

Resources, assets, files, content providers are all supported as image sources.

```
Picasso.get().load(R.drawable.landing_screen).into(imageView1);
Picasso.get().load("file:///android_asset/DvpvklR.png").into(imageView2);
Picasso.get().load(new File(...)).into(imageView3);
```

DEBUG INDICATORS

For development you can enable the display of a colored ribbon which indicates the image source. Call setIndicatorsEnabled(true) on the Picasso instance.

3.2.5 Native Applications

blah..

Advantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

Disadvantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

3.2.6 Hybrid Applications

blah..

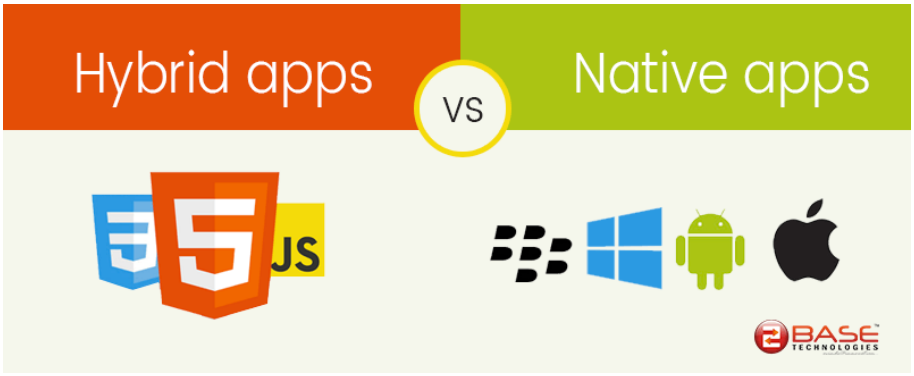
Advantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

Disadvantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

3.2.7 Hybrid vs Native Applications



blah..

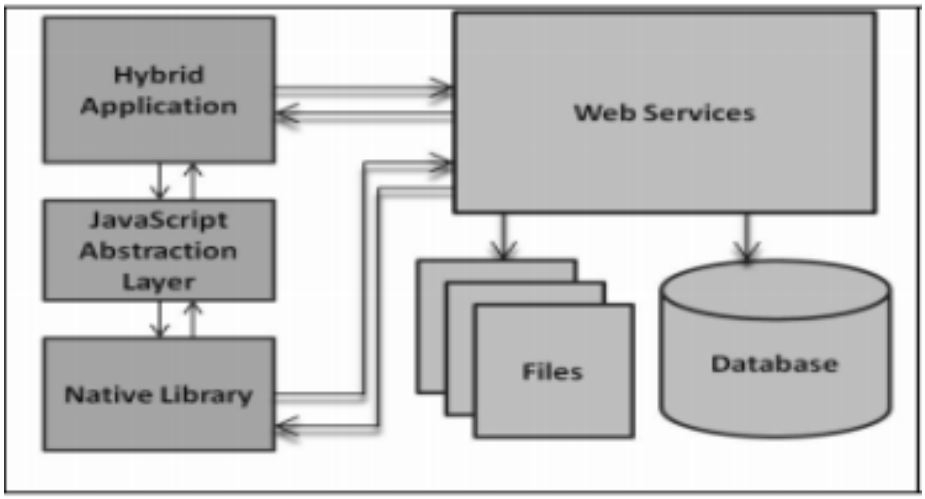
Advantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

Disadvantages

- **Shift from Java to Kotlin** - Kotlin
- **Hard to find experienced developers** - asdas
- **Limited learning resources** - Although

Figure 3.2: Hybrid Application



Chapter 4

System Design

Blah.....

Chapter 5

System Evaluation

Blah.....

Chapter 6

Conclusion

Blah.....

References

[1] *Airbnb moving away from react native.* URL: https://medium.com/airbnb-engineering/react-native-at-airbnb-f95aa460be1c.

[2] *Anatomy of Android Continuous Delivery.* URL: https://medium.com/the-telegraph-engineering/android-continuous-delivery-fb41da63176.

[3] *Android Performance tips.* URL: https://developer.android.com/training/articles/perf-tips.html.

[4] *AWS Pipeline CI/CD tool.* URL: https://aws.amazon.com/getting-started/projects/set-up-ci-cd-pipeline/.

[5] *Badges/Shield.io for Readme's.* URL: https://github.com/badges/shields.

[6] *CircleCI Continuous Delivery Software tool.* URL: https://circleci.com/docs/2.0/about-circleci.

[7] *CodeShip Continuous Delivery Software tool.* URL: https://documentation.codeship.com/basic/quickstart/getting-started.

[8] *Continuous Delivery Article.* URL: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment.

[9] *Feature-Driven Development.* URL: https://en.wikipedia.org/wiki/Feature-driven_development.

[10] *Feature-Driven Development compared to XP.* URL: http://www.featuredrivendevelopment.com/files/FDD_vs_XP.pdf.

[11] *Firebase.* URL: https://firebase.google.com/.

[12] *Firebase Android Documentation.* URL: https://firebase.google.com/docs/android/setup.

[13] *Firebase Android Google tutorial.* URL: https://codelabs.developers.google.com/codelabs/firebase-android/#0.

[14] *Firebase Authenication Docs.* URL: https://firebase.google.com/docs/auth.

[15] *Firebase Cloud Storage.* URL: https://firebase.google.com/docs/storage.

[16] *Firebase pricing*. URL: https://firebase.google.com/pricing.

[17] *Firebase Products*. URL: https://firebase.google.com/products.

[18] *Firebase Realtime Database Docs*. URL: https://firebase.google.com/docs/database.

[19] *Github Actions CI/CD tool*. URL: https://github.com/features/actions.

[20] *GitLab Continuous Delivery Software tool*. URL: https://docs.gitlab.com/ee/ci/.

[21] *Key Signing for Google App Store*. URL: https://developer.android.com/studio/publish/app-signing.

[22] *Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options*. URL: https://developer.salesforce.com/index.php?title=Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options&oldid=51601.

[23] *Picasso Source Code/GitHub*. URL: https://github.com/square/picasso.

[24] *React Native article by a Airbnb, Google Android Developer*. URL: https://medium.com/airbnb-engineering/sunsetting-react-native-1868ba28e30a.

[25] *Smartphone Market Share*. URL: https://www.idc.com/promo/smartphone-market-share/os.

[26] *Technical Documentation in Software Development*. URL: https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/.

[17] [11] [22] [1] [24] [3] [12] [14] [18] [15] [25] [13] [16] [23] [5] [9] [10] [26] [2] [8] [21] [20] [6] [7] [19] [4]