Dublin Institute of Technology
Department of Computing

Mobile Back-end as a Service iOS

Timothy Barnard
Student number: C13720705
Supervisor: Edina Hatunic Webster

Submitted in part fulfilment of the requirements for the degree of
Computer Science, 4th April 2017

Abstract

Text of the Abstract.

Acknowledgements

I would like to express (whatever feelings I have) to:

- My supervisor
- My second supervisor
- Other researchers
- My family and friends

Declaration

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Timothy Barnard 4th April 2017

‘Quote text here.’

Guy Quoted

Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction

1.1 Overview of the project

Back-end as a service or BaaS is best described by a tech analyst who refers to it as "turn-on infrastructure" for mobile and web apps. Basically it's a cloud computing category that's comprised of companies that make it easier for developers to setup, use and operate a cloud backend for their mobile, tablet and web apps [1].

The idea behind my project is to improve the development of modern mobile applications for new and experienced developers. This is achieved by providing services to help with the different phases of development.

1.2 Project Objectives

The project aim is to create a complete package that will enable new and experience developers speed up mobile application development, testing and run time. This will be accomplished by creating a list of services for each section.

1. Development
    - Database storage
    - Push Notifications
    - Sprint board
    - Backup
2. Testing

- A/B Testing
- Exception catching

3. Run
   - Remote Configuration
   - Analytics

To accommodate these services, the project will include three deliverables.

1. Dashboard

   The dashboard is a control panel that simplifies configuring applications. It also has data visualization to help improve the user experience when using the applications.

2. Mobile Back-end as a Service

   This is a model for developers to link their applications to the backend cloud storage and application programming interfaces (API's) exposed to provide the communication with the list of services above.

3. iOS Framework

   The services above need a way to communicate from back-end to the mobile apps. This is accomplish by providing a custom software development kits (SDKs).

1.3 Project Challenges

The key challenge of the project was to find what developers functional requirements looking for when choosing a mobile backend as a service what current providers do not offer. This required setting up meetings with professional mobile developers and get their opinion on the project.

The big challenge faced was to find a way to enhance the developing stage of any application, giving more power to the developer when the application has been published. This lead to another challenge to see how far Apple would allow applications to be configured after published.

1.4 Chapter Walk-through

Chapter 2 Research This chapters discusses the alternative existing solutions to the problem. It also goes into the potential technologies that could be used in the project along with the resultant findings. The

technologies discussed will be what web server to use, the programming language and what the dashboard be development in. This chapter will include surveys and out-sourced discussions with professional mobile developers.

Chapter 3 Design This chapters goes into the design of the projects. What methodologies will be used?, and go into detail of the list of features already mentioned in project objectives section.

Chapter 4 Architecture The architecture chapter will show the overview diagram of the complete project, along with diagrams of the individual services.

Chapter 5 Implementation Implementation chapters explains how to use different deliverables included in the project. Along with code snippets which show how to set up the system and use the SDK.

Chapter 5 Testing/Evaluation This chapter will give an overview of all testing carried out including test-driven approach and unit service testing. The evaluation section discusses the review given from the out-source professional developers. The review not only includes their feedback using the system but also their recommendations where to from here.

Chapter 6 Conclusion The conclusion chapter contains a summary of the project overall and ends with a reflection of the project.

Chapter 2

Research

2.1 Introduction

The main focus of the project is to provide new and experienced mobile developers with three deliverables. The deliverables will bring something new to the table, a new way of developing applications. These deliverables are as follows:

- Mobile Back-end as a Service
- Mobile framework (SDK)
- Dashboard

This chapter will not only cover the background research required for the deliverables, but also research from mobile developers and general mobile users. It will also cover the technologies required along with the current similar solutions.

2.2 Background

2.2.1 Project deliverables

Mobile Back-end as a Service

BaaS is an approach for providing mobile app developers a way to connect their application to back-end cloud storage and processing while also providing common features such as user management, push notifications,

storage, and other features that mobile users demand from their apps these days. The objective of any developer is to get their product finished and publish as quickly as possible. By providing a MBaaS to developers, it reduces time and resources needed to develop an app. As a research paper from Kinvey [1] states these points regarding what BaaS delivers:

- Efficiency Gains - Reducing overhead in all aspects of mobile app development, increasing efficiency at all stages of development
- Faster Times to Market - Reducing the obstacles to take a mobile app from idea to production and overhead with operations once in production
- App Delivery With Fewer Resources - BaaS supports development with fewer developers and supporting data and IT resources
- Optimize for Mobile and Tablets - BaaS providers have put a lot of time and resources into optimization of data and network for mobile apps, and reduce fragmentation problems across multiple platforms and devices.
- Secure and Scalable Infrastructure - BaaS provides a bundled infrastructure that deals with scalability, security, performance and other operational headaches, leaving developers to do what they do best
- Stack of Common API resources - BaaS brings common and essential 3rd party API resources into a single stack, preventing developers from having to go gather them separately

After reviewing why developers should use a MBaaS, the research paper goes on to discuss a pattern of building blocks being to emerge. These are the basic services that any MBaaS provider should be offering.

- User Management - It all starts with a user, the ability for users to sign up and log in.
- Storage - a central location for all app data to be stored to connect all users together.
- Rest API - the link between the back-end services and client applications.
- Communication - feature such as push notifications to keep users connected live with other users.

In the technologies section will go into what web-server will be used along with programming language required.

Mobile framework

Mobile frameworks helps with dealing with complex project such as integrating the MBaaS in apps. This is accomplish by using a dependency manager, which is a tool that manages all of the libraries in a meaningful

and logical manner. By dependencies, its the libraries to make the application work with the MBaaS. There are a few of iOS dependency managers which can be used which will discussed in the technologies section.

Dashboard

2.2.2 Mobile developers

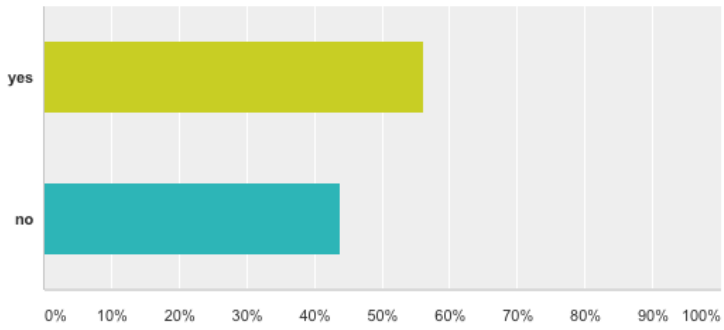
The key part of this project is to research from experienced developers, asking them in person and in forums. I have put up multiple questions in different forums regarding the functionality required for mobile Baas; What advantages and disadvantages do you find with current third party mobile back-end as a services?, What features do you want to be included in these services that you do not see at the moment? and Where do you see the future of third party services going, are they required or should mobile developers implement their own back-end? I have not been quite successful with these forums questions as most questions have to be problem specific but I have had some feedback.

I met with Trust5 [2] and Tapadoo [3], two mobile software developing companies based in Dublin, Ireland to discuss my project proposal. The general feedback was positive, seeing the potential and powerful tool it could be. I had a working prototype iPad app to demonstrate the remote configuration service as part of my project which updates a client mobile app interface which I will explain later. See appendix for the case study of each company.

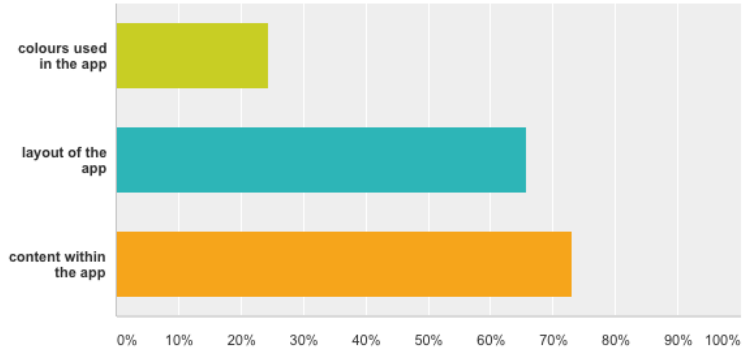
2.2.3 Mobile users

Surveys were created to help get mobile users feedback using apps. The reason for this to be part for the research is that they are the end-user. They are the ones we want to keep happy, keep using our apps. The survey contained a number of questions which along with the results are as follows:

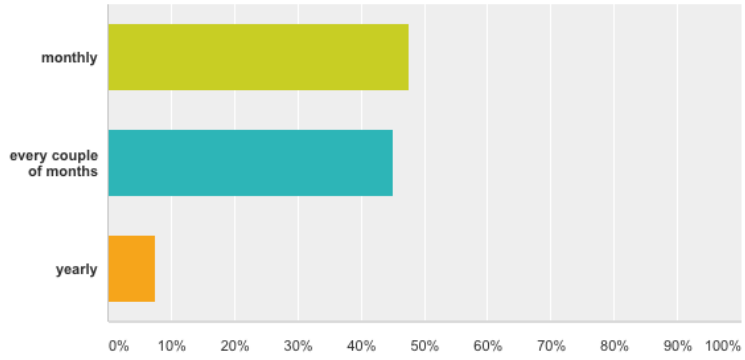
1. Would you delete an app if it crashes?



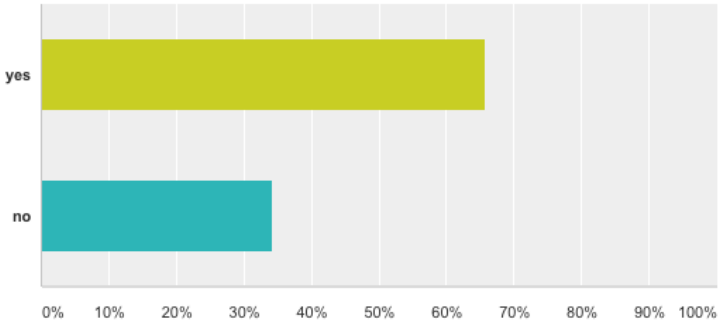
2. If you had a choice between two of the same type apps, Do you choose based on?

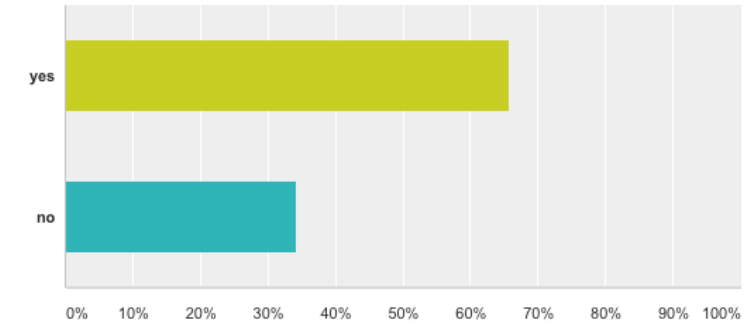


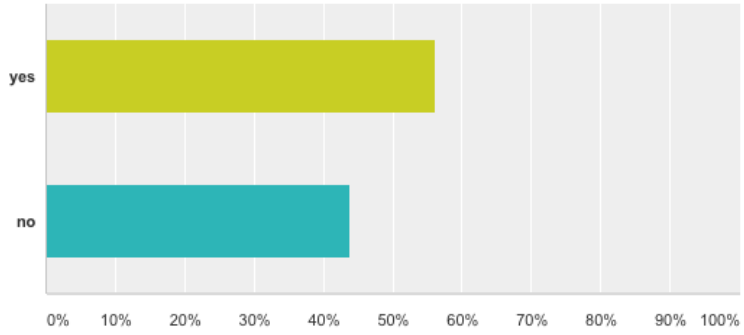
3. How often do you like app updates?



4. Would you prefer the user interface to be kept up to date and fresh more often?







5. Would you prefer being given the option for language choice (eg. English)?

6. Do you mind apps collecting analytics in the background to help improve the app ? (not personal information)

The above survey gives an understanding on what users feel when using apps, also some insight into what they want to have more often. By asking a users these questions, it has given a meaningful reason to implement one of the key features to be implemented in the project. A feature that enables developers to keep the app updated quicker for users to feel like they are not forgotten, to give the app a new look and feel. But also a quick way to stop the user from deleting the app because it crashes. This will be further discussed in the design chapter.

2.3 Technologies

2.3.1 Programming Languages

Objective-C

Objective-C [4] is an object-oriented programming language developed in early 1980s. It was used to develop on NeXT OS which later became OSX and iOS. Objective-C is a super set of the C programming language meaning that it is possible to compile any C program with an Objective-C compiler.

Swift

Swift [5] is Apples latest open-source programming language used mainly for developing on iOS, macOS, watchOS and tvOS applications. Swift it an alternative to the Objective-C language, but is sometimes referred to Objective-C without the C. In contrast to Objective-C, it does not expose pointers to refer to object instances. Swift does retain Objective-C concepts including protocols, closures and enums. It was designed to make writing and maintaining programs easier for developers.

Python

Python [6] is a high level programming language. It is designed to allow programmers to express concepts in fewer lines of codes than possible such as Java or C++. It is the perfect language to write programs on both a small and large scale. It supports object-oriented (OOP) and functional programming. Python was first created in the late 1980s by Guido van Rossum as a successor to the ABC language.

2.3.2 Web Frameworks

Perfect (Server-side swift)

Perfect [7] is a web server and toolkit for developers using the swift programming language to build applications and other REST services. It can be deployed on macOS and Linux server.

Kitura (Server-side swift)

Kitura [8] is a web server and web framework for Swift 3 developed by IBM. It is similar to Perfect, using core Swift technologies.

Django

Django [9] is an open-source high level Python web framework which follows the Model-View-Controller (MVC) pattern. It consists of an object-relational mapper (ORM) that maps models (M) defined as Python classes to a relational database, a system for processing HTTP request to a web view (V). It focuses on rapid development and the principle of dont repeat yourself and scalable. It was born in 2009 by few web developers and began to use Python to build applications. Django looks after authenticating the user when signing up, signing in and signing out. Python is used throughout the development even for setting files and creating models. Popular site such as Pinterest, Instagram and Bitbucket use Django for their web framework.

Flask

Flask [10] is a micro web framework written in Python and initially released in 2010. Applications such as Pinterest and LinkedIn use this framework. It is called micro framework as it does not require any dependencies to run, as well as that it does not have extra layers such as database but supports extensions that can be added to create extra features. Flask is popular among Python enthusiasts and was the most popular Python web framework on Github.

Node.JS

Node.JS [11] is an open-source cross platform environment originally released in 2009. It is used to create a variety of tools and applications such as GoDaddy, Groupon and Paypal. It is driven by events such as when a consumer purchases an item. Node.js has been optimized for web applications with many input/output operations, as well as real-time communication.

2.3.3 Dependency Managers

CocoaPods

CocoaPods [12] is the de facto standard of package management for iOS. It has the largest community and is officially supported by almost every open-sourced iOS library. It contains over twenty eight thousand libraries which are used in over 1.8 million apps.

Carthage

Carthage [13] is intended to be the simplest way of add frameworks to apps. It was the first dependency manager for macOS and iOS, created by group of developers from Github. It was also the first dependency manager to work with Swift. It exclusively uses dynamic frameworks instead of static libraries, this is the only way to distribute Swift binaries that are supported by iOS 8 and up.

2.3.4 Databases

MySQL

MySQL [14] is an open-source relational database management system (RDBMS) initially released in 1995. Previously owned by a swedish company MySQL AB and now owned by Oracle Corporation. MySQL works

on many system such as macOS, Windows, Linux, FreeBSD so making it a common choice and reviews are positive.

MongoDB

MongoDB [15] is also a free and open-source database type program, but is a document-oriented. Classified as a not-only SQL (NoSQL) which uses JSON like documents to store objects. Features includes indexing, replication, load balancing and the list goes on. The main difference of MongoDB is that it is not a relational database. Objects that are related somewhat together are stored together in one file, improving retrieval of data.

PostgreSQL

PostgreSQL [16] is an object-relational database with additional object features. It differs itself support for highly required and integral object-oriented and relational database functionality, such as complete support for reliable transactions. It also free and open-sourced, yet very powerful with the capabilities of storing procedures.

2.3.5 Evaluation

Table 2.1: Findings

Technology	Version	Area
Swift	3.0.1	Client Framework/Back-end Language
Linux	16.0.4 X64	Server OS
MongoDB	3.2.11	Cloud Storage
Flask	0.1.1	Test Web Framework
CocoaPods	N/A	Dependency Manager

Programming Language

Chosen technologies as a result from the research for client side application is Swift for both the framework and dashboard app. I decided to go with Swift for client side programming for few different reasons but one that stood out was a blog from 9To5Mac website [17]. At the beginning of 2016 Swift took over Objective-C making it the 14th most popular programming language. Protocol Oriented Programming (POP) is an interesting concept mainly used in Swift. It is starting to become commonly used instead of Object Oriented Programming (OOP) because objects are things that encapsulate complexity. So an object does it in a certain way, but protocols can provide objects with doing it different ways. With POP, the interface being what the user interacts with it the main and only concern.

Dependency Managers

A-Coding blog review goes into the pros and cons of both CocoaPods and Carthage [18]. CocoaPods has a large community behind it, so problems can be easily help and resolved. There are a large number of available libraries and are listed on their website: https://cocoapods.org. It has a centralised file that manages all libraries required for the app along with version. But because it is centralized, if it goes down you will be affected.

Carthage is not centralised, each dependency is fetched from their original repositories. It can be very slow managing all dependencies, especially if the app requires a large number of them. Some find CocoaPods invasive as it modifies the Xcode project by building its own workspace. However Carthage does not, it is up to the developer to add the required libraries into their project.

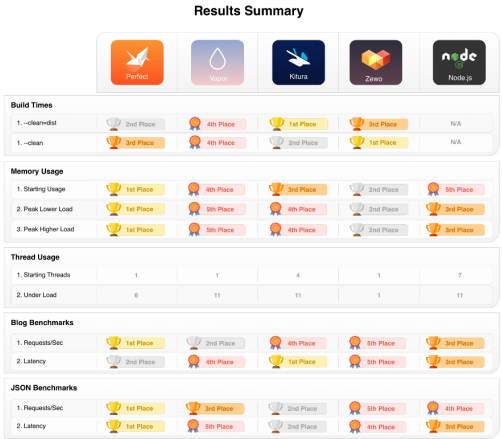
Due to extensive help from the community, and the project is all about helping the developer with reducing the amount of development, the project will be using the CocoaPods dependency manager for the MBaaS framework.

Web Framework

The project is aiming to be open-source, so that developers can contribute towards it. So without needing to learn a different language like Python for Django, the choice is a web-server using Swift language. As swift is becoming more popular and has become open-source, it is reasonable conclusion that the near future will include more back-end servers written in swift.

Now the web framework choose it based on Swift programing language, the choice is between Perfect and Kitura. A bench-marks was conducted by Ryan Collins, a web developer. He posted a blog about the outcome [19] which clearly shows that not only is both Perfect and Kitura faster than Node.JS but that Perfect is the leader. The blog goes into detail of the test that was run, the server setup and the number of requests made.

Figure 2.1: Benchmark



The result as stated in the blog, is that Swift is more capable of taking on the established server side frameworks. So regarding what Swift web framework to use, the decision is Perfect with over 10,000 stars on the Github repository. [20]

Database

The database choice is MongoDB which is a NoSQL. It provides a dynamic way of storing data where object properties name and type are unknown until run time. This allows the system to be implemented with different types of data being sent and gives the developers less stress without having to worry about what models need to be created at design time. From the article When to use MongoDB rather than MySQL [21] the reasons for choosing MongoDB is when:

- Expect a high load amount of data
- Need to grow big
- Do not have a database admin

The above 3 reasons are compelling enough to choose MongoDB because when designing a back-end as a service for applications of any type, then freedom and flexibility are key. Another valid reason is when you do not know the database structure and when providing a service to developers to give them tools to create any database structure is another strong reason to go with MongoDB.

2.3.6 Extra tools required

Xcode

Xcode is an integrated development environment (IDE) containing a suite of software development tools developed by Apple for developing software for macOS, iOS and tvOS. First released in 2003, the latest version 8 is free download via Mac App Store. It supports source code for programming languages C, C++, Objective-C and Swift etc.

Swift Playgrounds

Swift playground also know as just playground is an interactive work environment that allows you see the values in the sidebar for the written code. As and when you make changes to your code the sidebar reflects the changed result. It was introduced in Xcode 6 and enhanced in Xcode 7 that make learning Swift and experimenting

much easier. Instead of having to create a app just to run and test some Swift code, a smaller playground can be made to view the outcome of each test piece.

DigitalOcean server

DigitalOcean is an American cloud infrastructure provider. It provides developers cloud services that help to deploy and scale applications that run simultaneously on multiple computers. As of December 2015, DigitalOcean was the second largest hosting company in the world in terms of web-facing computers. Within a few steps, you can have web-server up and running. The perfect choice for this project, that can set-up a Ubuntu 16.04 server in a matter of minutes.

2.4 Similar Technologies

2.4.1 Overview

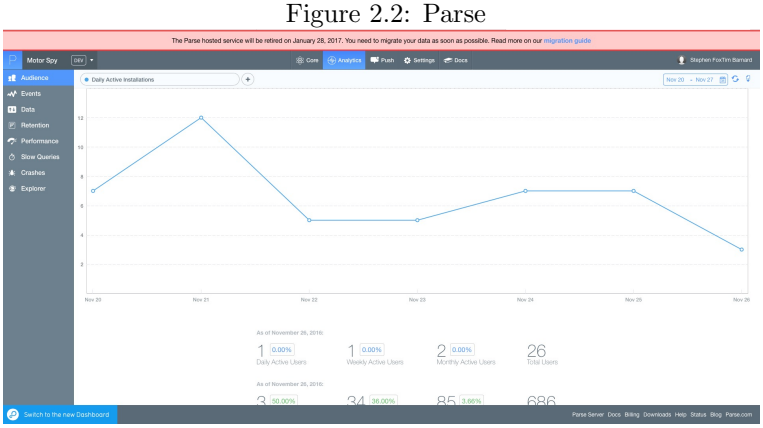
Table 2.2: Alternative Solutions

Service	Parse	Firebase	BassBox	Amazon
Notifications	Yes	Yes	Yes	Yes
Database	Yes	Yes	Yes	Yes
Analytics	Yes	Yes	No	Yes
self hosted	Yes	No	Yes	No
Remote configuration	No	Yes	No	No
Backup	No	No	No	No
User Sign-Up	Yes	Yes	Yes	Yes
A/B Testing	No	Yes	No	Yes

2.4.2 List

Parse

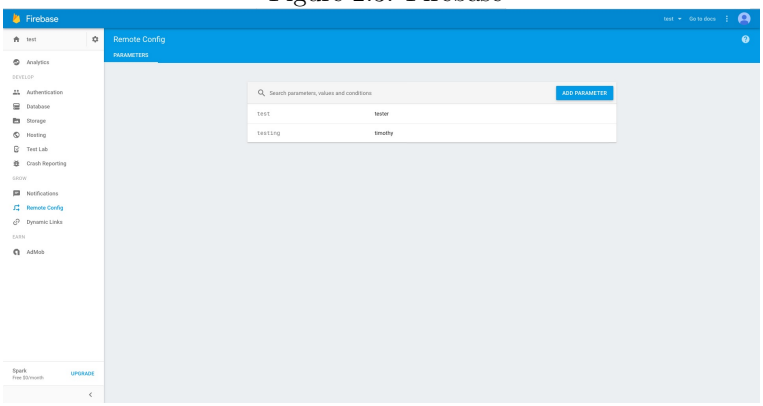
Figure 2.2: Parse



Parse [22] was founded in 2011 by Tikhon Bernstam, Ilya Sakhar, James Yu and Kevin Lacker former Google and Combinator employees. The project kick-started when it raised 5.5 billion dollars in funding in late 2011 and by 2012 over 20,000 mobile developers were using the service. Facebook in 2013 acquired the company for 85 million dollars and continued to grow. By 2014 500,000 apps were using the service, but sadly Facebook in 2016 announced that they are closing down the service in January 2017. They do provide tools and tutorials on how to migrate to your own hosted service. The service when operational was widely used by developers and Fig 2.2 shows the main dashboard page. One of the developers mentioned in the interview that his company Tapadoo used to use Parse before they announced the closure of the service.

Firebase

Figure 2.3: Firebase

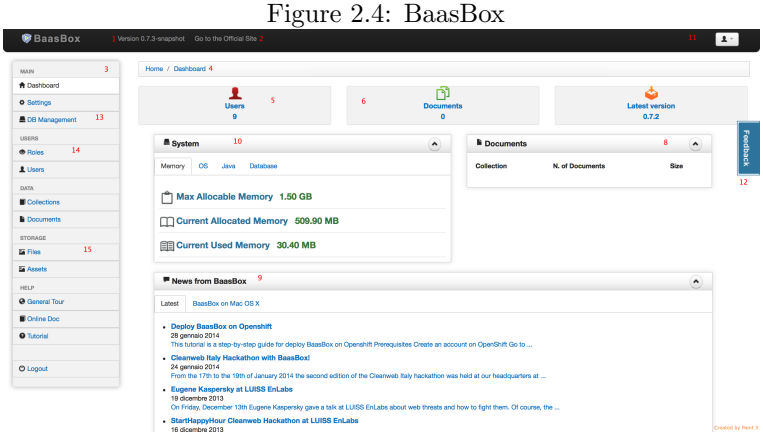


Firebase [23] evolved from Envolve, a start-up founded by Tamplin and Lee in 2011. They provided a service that enabled developers to integrate on-line chat into their web site but found after a while that the service was being used to pass application data that was not chat messages in real time. So the team decided to separate the chat system and real time architecture that powered it. This lead to the founding of Firebase, a separate company. It raised funding in 2013 and in 2014 Google acquired the company for an undisclosed amount. The company provides a list of services as shown in Table 2.2 for free for limited amount of users and storage of 5GB, but as you increased the storage and users using your application then so does the price. If we way in more storage, real-time database space then the price per month is around 200 dollars [24]. The dashboard for managing your project is shown in Fig 2.3, the current page is remote configuration service.

BaasBox

BaasBox [25] is an open sourced project founded in 2013, it is an application that acts a database and application server combined. It provides developers with an API as a back-end to store data for their mobile and web applications. It is the first back-end as a service to be open source and free to download. BaasBox also provides

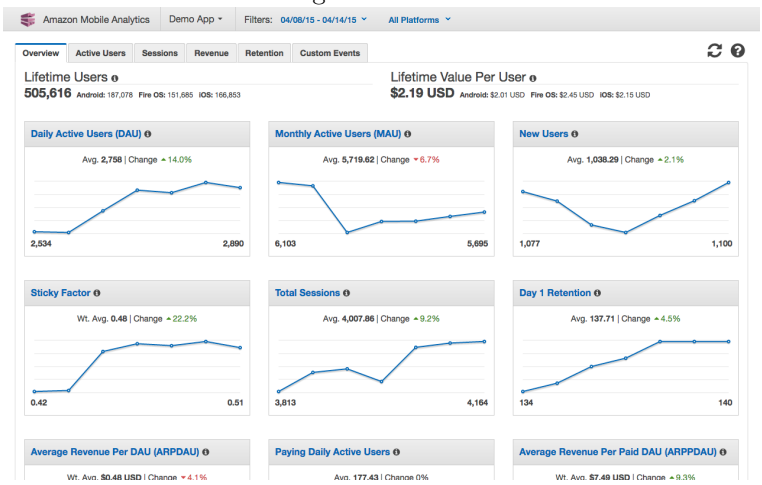
Figure 2.4: BaasBox



cloud services so instead of hosting the back-end on your own server. Fig 2.4 shows the dashboard main page, on the left panel are a list of services they provide such as database.

AWS

Figure 2.5: AWS



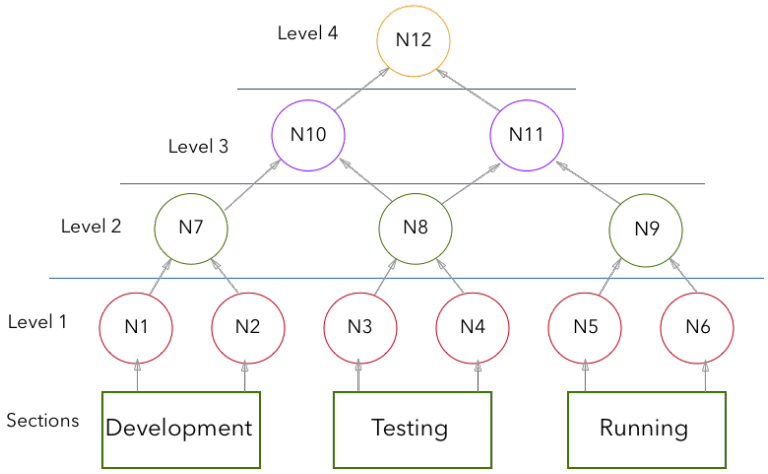
Amazon Web Services(AWS) [26] is a subsidiary of Amazon.com which offers a suite of cloud computing services launched in 2006. By 2007 amazon claimed that more than 180,000 developers had signed up to use AWS Amazon Web Services. One of its services to AWS Mobile Services, a way to provide help to develop mobiles apps than can scale to hundreds of millions of users. Popular companies such as AirBnB and Netflix uses AWS mobile services to power their applications. Like other MBaaS providers, they do offer a limited 12 month free tier which includes 5GB of standard storage, 20,000 get requests and 2,000 put requests. Then outside of these limits the price does rise. The list of services they provide include analytics shown in Fig 2.5.

Chapter 3

Design

3.1 Methodology

Figure 3.1: Tree-Shape



After doing research, I decided to implement my own methodology called "Tree-Shaped". The idea behind this methodology is to en-corporate the functionality into different sections, and prove each functionality at a lower, early stage. This benefits by setting out the goal into sections and filling in what functional requirements are need to reach the goal. The proving of functionality part is to able to fix any potential compatibilities at an earlier stage or remove if necessary.

The methodology has three parts

1. Sections

   The sections stage is first step in using the "tree shaped" methodology. This part is where we set out what we are trying to achieve. Then incorporate each functionality into their respective sections.

2. Functionality

   After we define our sections including their respective functionalities, tests are to be written out for each functionality. This are to show that they still output the correct result, being what they are supposed to do.

3. Levels

   Levels stage is where we are proving/testing the functionalists. As we move up the tree into each node, the functionalists are combined together. Thus solving compatibilities issues at an earlier stage of the project development.

This methodology is based on the test-driven development (TDD) process that relies on the repetition of very short development cycles. At each level, the nodes which holds the application are tested. These tests are from the functionality part, when combining the previous nodes, we want to make sure they still output the correct result.

The methodology although sounds good from a theoretical point of view, in a real world it has its drawbacks. Each node (circle in each level) requires making new test application, combing the previous two node applications and potential re-factoring. This takes time where some projects do not have, but we reduce the number of bugs found but re-factoring at each level. To overcome this, the methodology allows to skip one level to reduce testing times.

3.2 Functional Requirements

Table 3.1: Functional Requirements

ID	Section	Name	Description	Priority
1	Development	Database storage	Create, Read, Update, Delete objects	High
2	Development	Push notifications	Send push notifications to devices	Medium
3	Running	Analytics	Measure users in app activities	High
4	Running	Backup	Backup database to remote site	Low
5	Running	Self hosted	Host the MBaaS on developers server	High
6	Running	Remote Configuration	In app live updates	High
7	Running	A/B Testing	Testing different variations	High
8	Development	Live Database	Update objects without user refreshing	Low
9	Development	Dashboard	Interface for developers manage apps	High
10	Testing	Exception catching	Interface for developers manage apps	High

3.2.1 Development

Self hosted

Allow the developer to host their own system to have full control on when up and running, and not having to worry if the provider is going to shut down the system. By giving the developer a way to host their own back-end then this will keep the cost down of not having to paying for third party services.

Backup

This feature gives the developer a piece of mind that the applications data is constantly backup to local or remote location.

Sprint board

Challenge faced when creating an application, is trying to keep a list of features or changes to be made for a version. Also knowing history of features already implemented. Their are already different sprint board applications out there, but I wanted a way to en-corporate it all in one single dashboard app, that can be used with teams.

3.2.2 Testing

Crashes

3.2.3 Running/Live

Database

The API and framework will provide a service to send and retrieve objects to the database. These tools will not only be able to post and get objects but also to set out filters. Each collection of objects such as users can be viewed in a table format in the dashboard application explained later.

A/B Testing

A/B testing also known as split testing is comparing two versions of a page to see which performs better. Currently this popular with web pages but my plan is to bring this to mobile applications. All mobile applications no matter what services they provide all have one goal; a reason to exist. A/B testing allows you to make more

out of your existing traffic. This is achieved by sending our to variants (A and B) to similar visitors at the same time and use analytics to provide us what version wins. Included in my research, I did a survey to see how end-users respond to different applications. How they look?, Are they concerned on the aesthetics of the app?. At stated end-users do choose whether or not they will continue to use an app. So by using A/B testing service we can quickly find out what they do and do not like. So how can we implement this service in mobile apps? This leads on to my next service.

Remote Configuration

Remote configuration is a service that lets you change the behaviour and appearance of the app without requiring the users to download an app update. When using this service, you create the default appearance such as how a button looks, whats the title etc. Then later on we can update this values via the dashboard to configure these. So how does the app know when to get the new version? Included in each request done within the app is the configuration object at the top. This object will tell you, what version is available to download. So why use remote configuration?

- Quickly roll out changes to your app
- Customise your app depending on the version they are running
- Use this along with A/B testing to find improvements

Notifications

Allowing the developer to reach their users and perform tasks in the background. These allows the developers to bring the attention of the users to their app if for example a message comes in. A powerful tool to keep the app in real time and the user connected to the app. Ability to add notification certificates relating to each applications and to send individual or bulk notifications.

Analytics

Analytics to give the developer real time information on how their app is doing, how users are using their app. Custom tool to configure what data they want to analyse for example, if the applications is based on vehicles then it can be configured to show list of top vehicle type, make or model. Each configuration will have a graph on the main screen. Analytics will also be used for AB Testing service explained later.

3.2.4 Non Functional Requirements

3.2.5 Security

Security is a big part to any cloud based applications. Users personal information being sent up to cloud, where potential hacks could expose these. Not along has able enforcing developers to user HTTPS request, to secure data transmission, I have also taken into account some security measures.

Keys

The systems supports two keys authentication, there is one key that allows requests to access the web server services. The next key allows data back and fourth to the database.

Database

Instead of all applications the developer owns being contained in one database, each application created gets their own database. These are only allowed access once the key gets authenticated.

3.3 Project Components

Chapter 4

Architecture

4.1 Overview

Figure 4.1: Overview

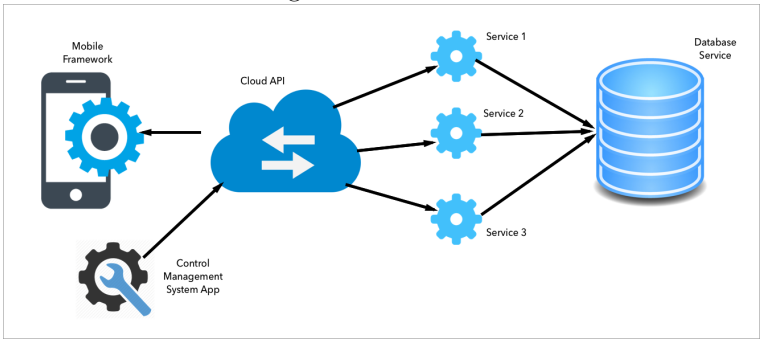
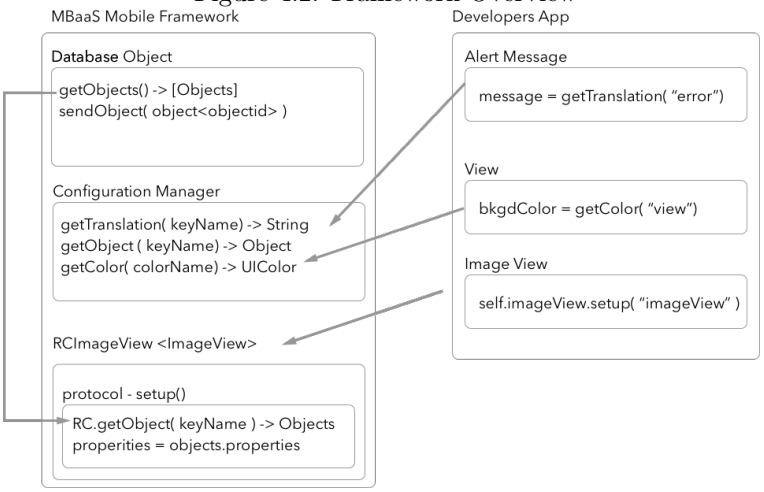


Fig 4.1 shows the complete Mobile Back-end as a Service system. It shows the mobile framework used by developers application to send HTTPS request using a REST API such as get request to retrieve objects from the database layer using a service layer. The service layer is where the database, notifications and remote configuration services sit. So when a request is made to the Cloud Rest API framework such as send object to the server , the API will call the database service class and will enter the object into the database and return the result back to the client (framework). The diagram also shows the Control Management System (CMS) app using a dashboard to accomplish such tasks as send the configuration files updates to the server, which will send HTTPS request to the Cloud API and have its own service class to accomplish these tasks. The diagram only displays 3 service class, but can be as many as required.

Mobile frameworks separates the back-end API by functionality, making it easier for developers to create apps without having to learn how the API works. It provides class functions and protocols to which can be used to interact with the server API. An example of this is in Fig 5.3 where the framework has the database object that

Figure 4.2: Framework Overview



makes HTTPS request to retrieve the objects, but the developer can use the inherited function call getObjects. The remote configuration requests are also in the diagram to get values from the JSON files stored on the mobile phone app directory.

4.2 Diagram

4.3 Development Concepts

Chapter 5

Development

5.1 Introduction

As explained in the design chapter, this project includes three deliverables. So the development will include; a Mac app dashboard, Perfect web-server and CocoaPods framework. Using the "tree-shaped" methodology the web-server was split up into their separate sections which include development, testing, and run. Then each service was design, developed and testing before moving up the tree. The development then broken up into phases, this way the project can be kept on track on what has been completed and whats left to do.

1. Services Development
2. Integrate into Live App
3. Dashboard Development
4. CocoaPod Framework

5.2 Services Development

Each service had its own flask application along with Playground app. This made is easy to decide whether or not it was possible to implement each service into the project.

Database storage

The database storage required an object role model (ORM) which is a powerful method for designing and querying database models at the conceptual level, where the application is described in terms easily understood

by non-technical database developers. It is a technique for converting data between incompatible type systems in object-oriented programming languages. The ORM was developed using Playground tool, where the creation of objects and parsing into JSON objects. The functionality of the ORM is to create a new object, parse it and send to the server, and be able to bring all objects back from the server. Extra functionality was added to be able to send filters.

Figure 5.1: Storage Sequence Standard

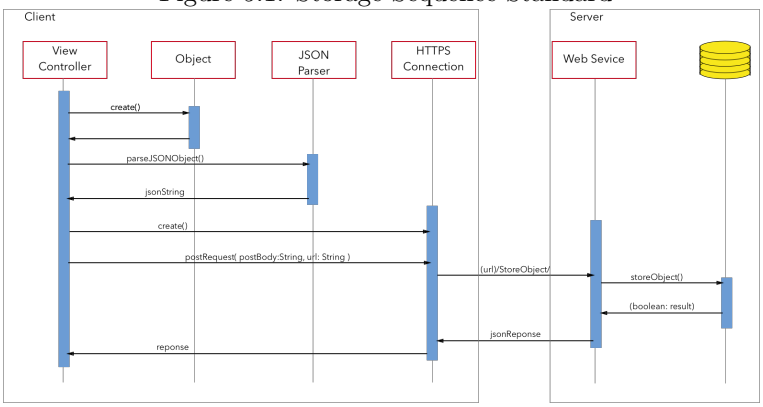
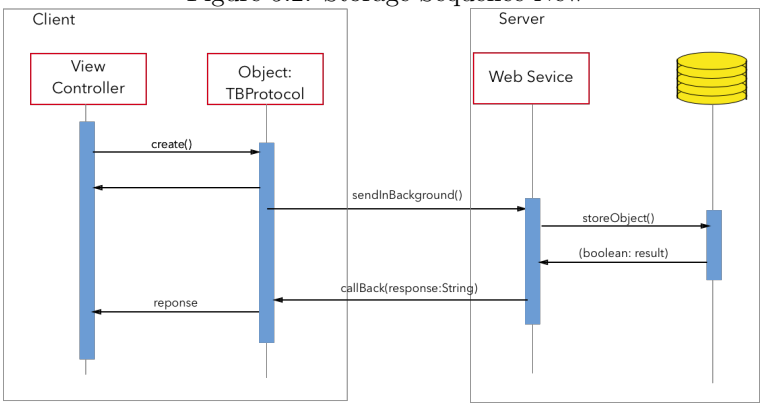


Figure 5.2: Storage Sequence New



The two figures above show the difference to developing the standard way, with the developing having to create a JSON parser and then set-up a HTTPS connection to send it to the server in Fig 5.1 . The new way in Fig 5.2 using the project's storage protocol involves the developer only having to create the object, then using the objects functionality to send it to the server.

Push notifications

Analytics

Backup

Self hosted

Remote Configuration

A/B Testing

Live Database

Exception catching

5.3 Integrate into Live App

For some parts of project, an already developed and published app called DIT-Timetable was used to add in the services, to test if Apple would allow it through. The services include remote configuration and language choice. Due to Apple's strict guidelines, the remote configuration was developed into the DIT-Timetable app into two stages, then each stage had a build and published.

Phase 1

This phase included just the basic remote configuration, with the capability of updating text such as page title, and label values. Apple did approve this phase, and while the app live, using the iPad prototyping app the text values were able to be changed.

Phase 2

Phase 2 gave the ability to adjust user interface values such as text colour, text size and user interaction enabling/disabling. This also been approved by Apple giving it a go ahead to be completely integrated into the project.

5.4 Dashboard Development

The dashboard was originally going to be created as an iPad app, but after some thought that not all mobile developers can be expected to own an iPad, the dashboard was developed as a Mac App.

5.5 CocoaPod Framework

The CocoaPod was left to last to development, as all the test cases used in the Playground could be brought over and adjusted to fit the framework. After the CocoaPod was finished, a test project was created for external professional mobile developers to use and give some feedback.

Chapter 6

Implementation

6.1 Installation Deployment

One of the key components of my project is the back-end web application. As one of services I am providing is self hosted. So developer can host their own back-end on any server of their choosing as long as the operating system is Ubuntu 16.04. The complete system is available on Github.

https://github.com/collegboi/PerfectServer

```
#!/bin/bash

#    name@ip_address
ssh root@100.100.100.100
```

Listing 6.1: Server Login

SSH, or secure shell is a network protocol that provides a secure, encrypted way to communicate with your server. In Figure 6.1 is used to log in to your server. The user-name by default by root but we will change this next and the next field is the IP address of your server.

```
#USERNAME = your new username you want to be called
useradd -d /home/USERNAME -m -s /bin/bash USERNAME
echo "USERNAME ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

#USERNAME = your username you just set
passwd USERNAME
```

Listing 6.2: Setting user-name

In Figure 6.2 we are going to set your new user-name and password. Doing this will help secure your server by moving away from the default root user-name.

```
#pulling the server from github
git clone https://github.com/collegboi/PerfectServer

cd /PerfectServer
./install
```

Listing 6.3: Installing

Last is to pull down the server from Github and install all the necessary packages This is done by running the following commands one at a time in Figure 6.3.

6.2 Development

6.2.1 Add the SDK

```
$ cd your-project directory

#if you do not have cocoapods installed
$ sudo gem install cocoapods

$ pod init
```

Listing 6.4: Init pods

After creating if you don't have an Xcode project, you will want to install the SDK. This will be done using CocoaPods. CocoaPods is a dependency manager for Swift projects. It contains thousands of libraries that can be used in your apps. One of which is MBaasKit. Start off with stepping into your project as seen in Figure 6.4.

Next we will add the MBaasKit pod to the file 6.5.

```
pod "MBaaSKit"
```

Listing 6.5: Pod file

Last part for adding the SDK to your project is to install the pod. This can be done by running the command in Figure 6.6

```
$ pod install
$ open your-project.xcworkspace
```

Listing 6.6: Pod install

6.2.2 Using the SDK

Notifications

To start using notifications, we first need to create the installation object and send that to our server. This is done by adding the following function 6.7 in our AppDelegate file.

```
func application(_ application: UIApplication,
    didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
  let installation = Installation(deviceToken: deviceToken)
  installation.sendInBackground() { ( completed, message) in
    DispatchQueue.main.async {
      if (completed) {
        print(message)
      }
    }
  }
}
```

Listing 6.7: Register for Notifications

Storage

```
struct TestObject: JSONSerializable {

  var name: String!
  init() {}

  init(name:String) {
    self.name = name
  }

  init( dict: [String:Any] ) {
    self.name = dict["name"] as! String
  }
}


var result = [TestObject]()
result.getAllInBackground(ofType:TestObject.self) { (succeeded: Bool, data: [TestObject]) ->
    () in

  DispatchQueue.main.async {
    if (succeeded) {
```

```
21        result = data
22        print("success")
23      } else {
24        print("error")
25      }
26    }
27  }
28
29  let testObject = TestObject(name: "timothy")
30  // if objectID is set then it is updating else inserting
31  testObject.sendInBackground("<objectID>"){ (succeeded: Bool, data: NSData) -> () in
32
33    DispatchQueue.main.async {
34      if (succeeded) {
35        print("scucess")
36      } else {
37        print("error")
38      }
39    }
40  }
```

Listing 6.8: Storing/Retrieving Objects

As you can see in Figure 6.8, when creating a struct and using the protocol JSONSerializable, we can then send and retrieve the objects in the backgrounds using those commands.

Exception Handling

Exception is a problem that arises during the execution of a program. If exceptions are not handled, it can cause the app to crashes and when the app is live, we would have no way of knowing. But by adding the following lines 1 2 in Figure 6.9 we can catch uncaught exceptions and send them to our back-end storage and view with the dashboard. Line 4 is an example of sending a caught exception that can have a message sent along.

```
1  MyException.client()
2  MyException.sharedClient?.setupExceptionHandler()
3
4  MyException.sharedClient?.captureMessage(message: "Caught Exception")
```

Listing 6.9: Storing/Retrieving Objects

Chapter 7

Testing and Evaluation

7.1 Testing

Software testing is an investigation conducted to provide information about the quality of the product or service under test. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. In my project I am using test-driven approach method, this helps eliminate parts of the code that will not work or fix them before adding them to system.

7.1.1 Service testing

Service testing is where we test the individual services to ensure they are working and outcome are what is expected. This is done by creating a small flask server that will take HTTP requests from a client app for each service such as storage etc.

7.2 Evaluation

Chapter 8

Conclusion

8.1 Summary of Project Achievements

8.1.1 Reflective

My project has changed from the beginning within a few months into it. I started out wanting to develop the back-end using Django framework and Python for the programming language. This changed to using Perfect (Swift server side) for the back-end and Swift language for both client and server. The reasons behind this because the project will be open sourced, it will allow developers already developing their apps in Swift to contribute towards it. Having software open source has its benefit which not only include having developers contributing and in return having free software. It can also help with providing more functionality, services and security to the system.

8.1.2 Learning outcomes

So far with the research, the project has potential to be completed without any major issues. There is a need for an open-source mobile backend a service for new and experienced developers, as current systems either are expensive or do not have enough services available. Conclusions I have come across and did not expect the outcome was with pushing the limits of what can be changed remotely in terms of user interface. The outcome is that it can be done, but still not sure how far I can go. As part of the research with talking to developers has been that the project has plausibility. Personal development has been with research, the amount of done with this project out ways any other project done before. Researching my project idea has shown me skills to when developing software to always think ten steps ahead where possible. Instead of starting the development and researching along the way but start with the research has shown a completely new way to creating software.

This way of creating software has advantages such as finding the potential issues and risks early on to overcome. Speaking with professional mobile developers when I did the research has helped with the project with not only validating my idea that it has potential but also giving me constructive feedback. The feedback given has made me do more research regarding other services that companies are doing to look into. One developer I had an interview with also pointed out some areas to be cautious about. The research gone into this project apart from technologies and methodologies being used has shown the need for a mobile backend as a service.

8.2 Applications

Applications.

8.3 Future Work

Future Work.

Chapter 9

Cases

9.1 Research

9.1.1 Tapadoo

Tapadoo [3] gave me more constructive feedback, stating that while the remote configuration service on its own is good it has its drawbacks. A developer's point of view is that updating the user interface will not happen often enough to validate the use of it. However using the remote configuration along with A/B testing (comparing two variations of an app against each other) is a powerful, useful tool for developers and customers. Instead of relying on what the designer or the project manager thinks, they leave it up to the users by viewing the analytics based on the two different variations.

He also mentioned to be careful when using the translation files and allowing the users to choose their own language as this goes against Googles and Apples guidelines. He stated that there are services already implemented called Localization that deals with the displaying the correct translation. He also explained a need for updating content within an app, changing the button title from Pay to Pay Now for example and that my project should include this service.

9.1.2 Trust5

Trust5 [2] had two developers to meet with me, they expressed interest in the idea and said it is a powerful tool for developers to use. They gave me feedback to design it as a white label product, meaning that the product can be used by any company with their logo attached. We discussed the different features already implemented with regards the remote configuration and explained where to concentrate on and what to leave

to last. The configuration is split up into three phases of testing as explained earlier, they talked about leaving the complicated part of converting user interface objects to Apples visual format language to last and just used the constraint object class until the majority of the project was completed.

9.2 Evaluation

9.2.1 Trust5

Bibliography

[1] Kinvery. Backend as a service whitepaper. Available at http://go.kinvey.com/baas-ecosystem-whitepaper/, [cited March 2017].

[2] Trust5. Available at http://www.trust5.com, [cited November 2016].

[3] Home — tapadoo [internet]. Available at https://tapadoo.com, [cited November 2016].

[4] Apple. About objective-c [internet]. developer.apple.com. Available at https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introd [cited October 2016].

[5] Apple. Swift - apple developer [internet]. developer.apple.com. Available at https://developer.apple.com/swift/, [cited October 2016].

[6] Welcome to python.org [internet]. Available at https://www.python.org, [cited October 2016].

[7] Perfect. Available at http://perfect.org/, [cited January 2017].

[8] IBM. Kitura. Available at http://www.kitura.io, [cited January 2017].

[9] The web framework for perfectionists with deadlines — django [internet]. Available at https://www.djangoproject.com, [cited October 2016].

[10] Welcome — flask (a python microframework) [internet]. Available at http://flask.pocoo.org, [cited October 2016].

[11] Foundation n. node.js [internet]. Available at https://nodejs.org, [cited October 2016].

[12] Coocapods. Available at https://cocoapods.org, [cited March 2017].

[13] Carthage. Available at https://github.com/Carthage/Carthage, [cited March 2017].

[14] Available at https://www.mysql.com, [cited October 2016].

[15] A review of ios dependency managers. Available at https://a-coding.com/a-review-of-ios-dependency-managers/, [cited October 2016].

[16] Postgresql: The world's most advanced open source database [internet]. Available at https://www.postgresql.org, [cited October 2016].

[17] Django vs flask: Which is better for your web app? [internet]. Available at http://ddi-dev.com/blog/programming/django-vs-flask-which-better-your-web -app/, [cited November 2016].

[18] A-Coding. A review of ios dependency managers. Available at https://a-coding.com/a-review-of-ios-dependency-managers/, [cited March 2017].

[19] Benchmarks for the top server-side swift frameworks vs. node.js. Available at https://medium.com/@rymcol/benchmarks-for-the-top-server-side-swift-frameworks-vs-node-js-24460cfe0beb.3cmlf3psm, [cited January 2017].

[20] Perfect github repository. Available at https://github.com/PerfectlySoft/Perfect, [cited January 2017].

[21] When to use mongodb rather than mysql (or other rdbms): The billing example - dzone database [internet]. Available at https://dzone.com/articles/when-use-mongodb-rather-mysql, [cited November 2016].

[22] Parse [internet]. Available at https://parse.com/, [cited October 2016].

[23] Google. Firebase — app success made simple [internet]. Available at https://firebase.google.com/, [cited September 2016].

[24] Google. Pricing — firebase [internet]. Available at https://firebase.google.com/pricing/, [cited October 2016].

[25] Baasbox build awesome apps faster [internet]. Available at http://www.baasbox.com/en/, [cited September 2016].

[26] Amazon. Pricing [internet]. amazon web services, inc. 2016. Available at https://aws.amazon.com/ec2, [cited September 2016].