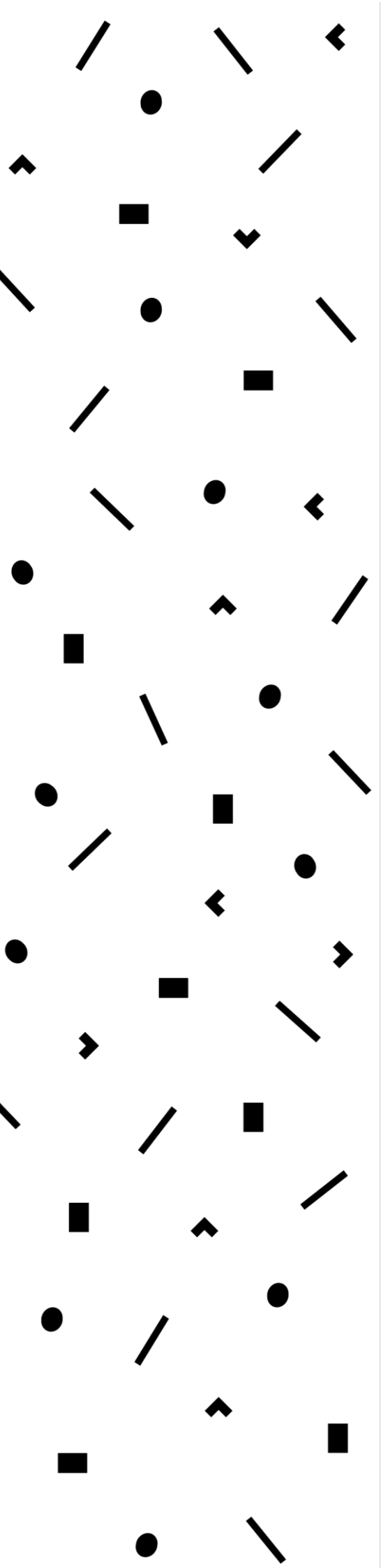

LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

MOBILE BACKEND AS A SERVICE: THE PROS AND CONS OF PARSE

Case company: SuperApp Oy

LAHTI UNIVERSITY OF APPLIED SCIENCES
Degree program in Business Information Technology
Bachelor's Thesis
Autumn 2016
Phu Nguyen

Lahti University of Applied Sciences
Degree Program in Business Information Technology

NGUYEN, PHU

Title: Mobile Backend as a Service: The Pros and Cons of Parse
Case company: SuperApp Oy

Bachelor's Thesis in Business Information Technology
40 pages, 2 pages of appendices

Autumn 2016

ABSTRACT

Using a pre-built backend for an application is an affordable and swift approach to prototyping new application ideas. Mobile Backend as a Service (MBaaS) is the term for pre-built backend systems that developers can use. However, it is advisable to understand the pros and the cons of an MBaaS before deciding to use it.

The aim of the thesis was to determine the advantages and disadvantages of using Parse, a provider of mobile backend as a service, in application development. Parse's definition and other studies about the topic are also discussed.

The theoretical part of this study is based mostly on digital resources and articles related to the topic. On the other hand, the data was empirically collected from project management tool logs of a web application development and interviews with developers who have different amounts of experience in programming. The conclusion was then derived from data analysis and data comparison between the literary and empirical findings.

The conclusion indicates that the advantages of Parse in application development appear to be: cost and time saving; simplicity in implementation; rapidity in learning; rich features in addition to backend tasks; solid data management tools, APIs and SDKs; user experience oriented development. Despite the good characteristics, the disadvantages are limitations in custom backend function and dependencies on the service. However, because of the scope of the case company, the findings are suitable for companies or individuals that develop prototypes or applications for a small group of people.

Keywords: Parse, mobile backend as a service

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF IMAGES

LIST OF ABBREVIATIONS

APIs	Application Program Interfaces
DIY	Do It Yourself
MBaaS	Mobile Backend as a Service
SDKs	Software Development Kits
SQL	Structured Query Language

1 INTRODUCTION

1.1 Thesis background

The twenty-first century is the century of social technology (Kozma & Schank 1998, 3) in which the internet and mobile devices are more accessible. The number of people who have access to the internet is nearly 3.5 billion, and approximately 6 million smartphones and tablets are sold daily (Website live stats 2016). This probably results in a surge in the number of mobile applications and web traffic.

On the other hand, more people are getting interested in the creation of hi-tech such as web developing or software engineering. As Avram notes, the number of software developers increased by 5.3% from 2011 to 2014 from about 10.5 million developers to about 11 million (2014). Furthermore, according to the US Bureau of Labor Statistics, it is estimated that the number of application software developers will be approximately 664.5 thousand in 2020, increasing from 520.8 thousand in 2010 (Statista 2016).

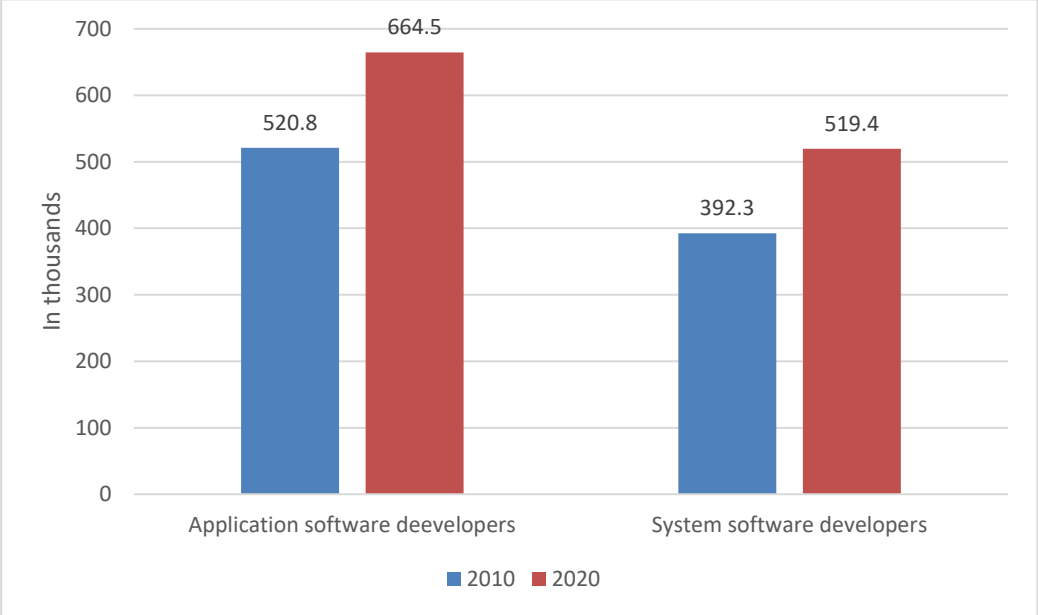

FIGURE 1. Employed software developers in the United States in 2010 and 2020 (Statista, 2016)

The two above-mentioned reasons can possibly lead to a rise in application development. However, a cross-platform application, a mobile application, or a web application, requires a backend system. A solution for developers who intend to create applications without having to think much about backend issues, is to use a Mobile Backend as a Service (MBaaS). Parse is one of the MBaaSs.

1.2 Methods and general structure

This paper explains the advantages and disadvantages of using Parse. The answer to the research problem is resolved by applying a qualitative research method. Specifically, data from interviews and a web application development case will be used for the method. This data is then compared and analysed with findings from the literature review.

This thesis consists of eight parts:

- introduction
- research problem
- research methodology
- theoretical framework
- literature review
- data analysis
- results
- discussion.

1.3 Research objectives and research questions

Defining the research questions and objectives is vital for any study (Yin 2009, 10) and the purpose of a research affects the plan applied to achieve goals that the authors head to (Eriksson & Kovalainen 2008, 27). The purpose of this paper is to provide an understanding of the term Mobile Backend as a Service (MBaaS), and to present information about

the advantages and disadvantages of Parse, an MBaaS provider, in application development.

The research question: *What are the advantages and disadvantages of Parse in application development?*

Listed below are the sub-questions formed to create a deeper understanding of the main question:

1. What are the advantages of an MBaaS and Parse?
2. What are the disadvantages of an MBaaS and Parse?

The nature of the main question is descriptive because its answer will portray the characteristics and features of the subject. By the same token, both sub-questions are descriptive.

2 RESEARCH METHOD

This chapter discusses research methods, how the study was conducted and data was collected.

2.1 Methodology

A paper needs a valid research method. Eriksson and Kovalainen have mentioned that describing a research methodology in a study will back the author up and provide him or her with the techniques of creating a general strategy for the research. Specifically, the method specifies the approaches and steps to finding results for the research question. (Eriksson & Kovalainen 2008, 25-26.)

There are two possible approaches for conducting researches: deductive and inductive. While an inductive research is based on exploring the phenomenon and collecting data to form a concept or a theory, a deductive research provides arguments and results based on available theory, concepts from books or academic materials, and data collected (Saunders et al. 2009, 61). In short, the inductive method and deductive method are opposite approaches.

In this research about Parse, the author used the deductive method. Another name for the deductive way of researching a study is “waterfall” or “top-down” method. Figure 2 demonstrates the general steps in the “waterfall” method.

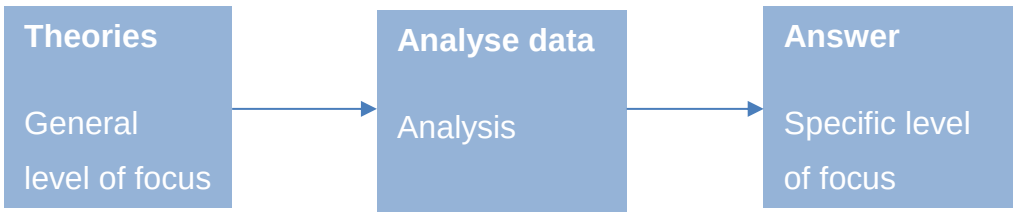

FIGURE 2. General steps in “waterfall” method

Moreover, this study applies the qualitative method. This is because with this method, the phenomenon can be understood and studied from a practical perspective. The approach can also be understood more clearly by looking at Creswell's explanation (2009, 4) that qualitative research is:

> A means for exploring and understanding the meaning individuals or groups ascribe to a social or human problem (Creswell 2009, 4).

Theories used in this method concerned Mobile Backend as a Service (MBaaS), especially Parse. In short, MBaaS is a service for cloud storage and other functionalities for web or mobile applications. Furthermore, Parse is one of the MBaaS providers taken into further study in this paper.

2.2 Study framework

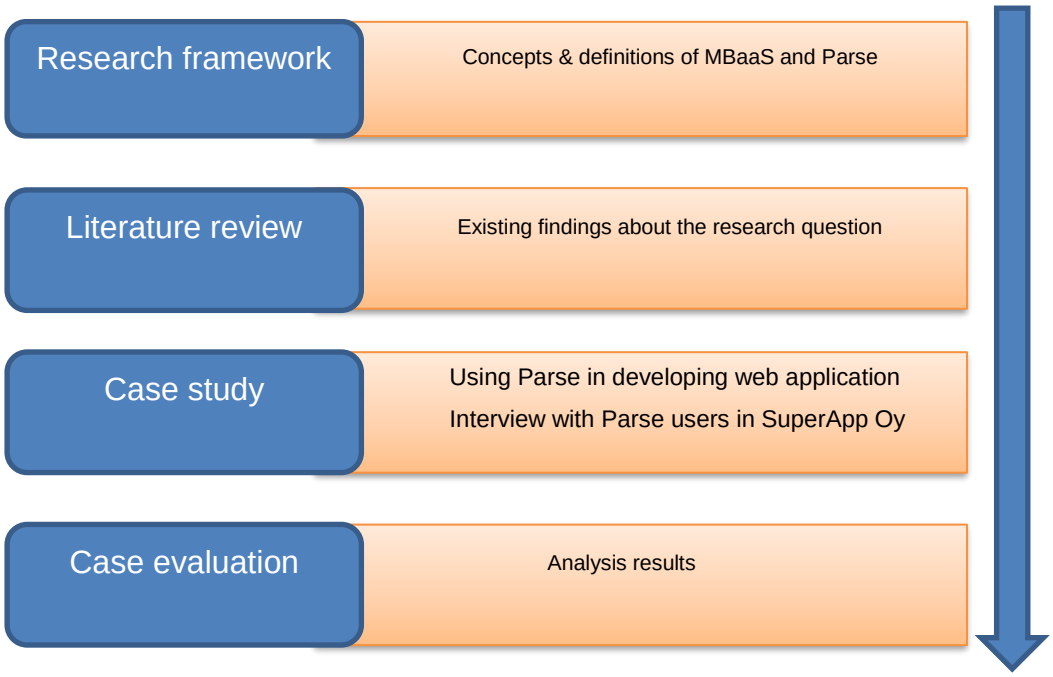

FIGURE 3. Study framework

Figure 3 describes the framework of this study including the following: a research framework, a literature review, a case study, data collection, and a case evaluation. To create a research framework, both the concept of Mobile Backend as a Service and Parse were explained. After finding basics information on the subject, the literature review was compiled based on existing findings regarding the research question. The case

study was made based on SuperApp's web application by addressing some of the pros and cons in using Parse. Moreover, data was gathered by interviewing SuperApp's developers who have used Parse. Finally, the data collected in the case study was analysed and then compared with the literature review findings.

2.3 Data collection

SuperApp Oy, a start-up company in Lahti, Finland, was chosen for the case study. The company was selected because the author has been working for them. Not only did he use the web application, the development of which he took part in, but his position was also beneficial in the interviews as the interviewees were his colleagues.

Data from case study was collected using the participant method and semi-structured interview. Regarding the participant method, data was gathered from the author's working diary written during the development phase of the web application. In addition, data was collected from interviews with the colleagues through semi-structured and in-depth interviews.

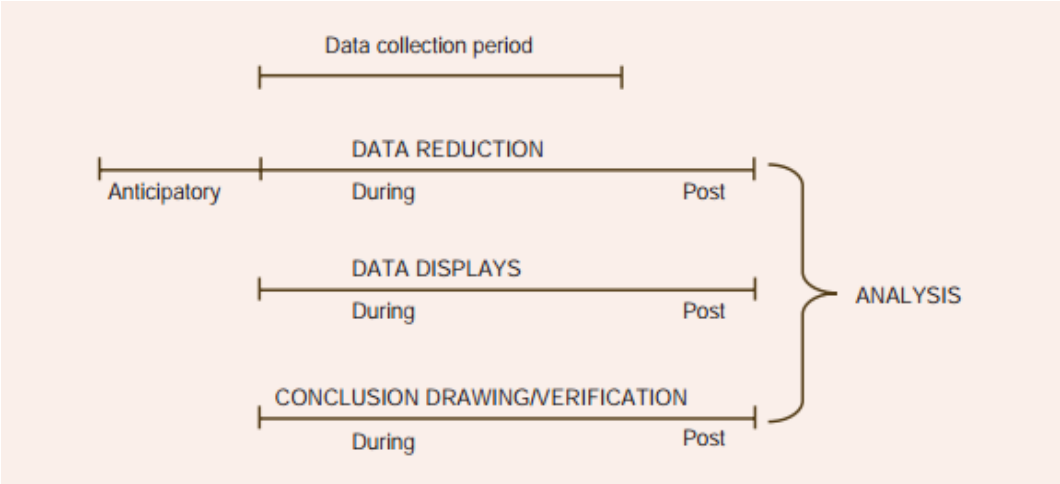

FIGURE 4. Data analysing steps (Engel & Schutt 2012, 306)

Figure 4 demonstrates the steps in analysing the data that the author used in this thesis. The first step is the documentation of the data, also known as the data collection period. Data was gatherd from project tracking tool

logs and interviews with SuperApp Oy’s developers. The second step is organising or categorising the collected data into concepts. This stage is called data reduction. After data reduction, data was displayed by connecting it to the literature review findings. The final stage was the legitimisation of data. This stage was done by evaluating the results and then compiling the findings.

3 RESEARCH FRAMEWORK

In this section, the author describes the research framework that considers MBaaS and Parse. The following concepts are defined: backend, MBaaS, SDK, and API. In addition, Parse's definition, history, and features, and the technology behind it will be presented.

3.1 Mobile Backend as a Service

3.1.1 Backend

To understand what an MBaaS is, the definition of a backend will be mentioned first. Figure 5 shows what a backend is:

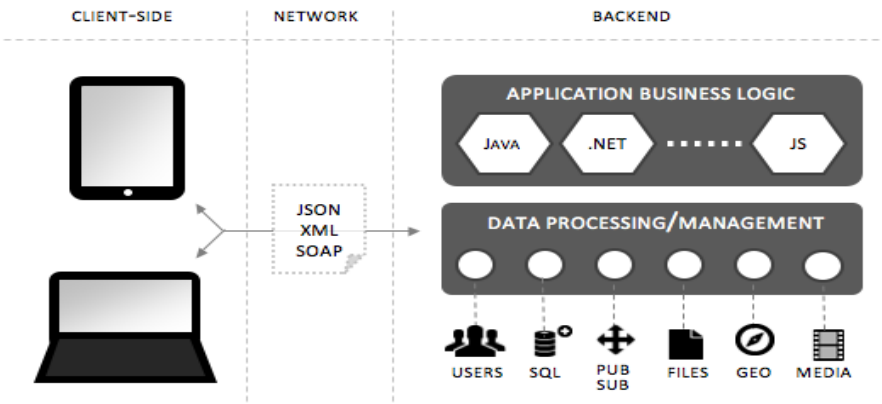

FIGURE 5. Backend (Backendless, 2016)

A backend consists of application business logic and data processing/management. The data processing or management sectors are made of users, SQL, files, pub/sub, geo, and media. (Backendless 2016.)

3.1.2 MBaaS's definition

According to Carney (2013), Mobile Backend as a Service (MBaaS) is a service that provides a way to connect the application to backend cloud storage for people who develop web and mobile applications. On the other hand, an MBaaS also delivers features like user management, push

notifications, and social networking service integration. The package of services is offered using SDKs and APIs. Backendless 2016 (2016) provides a figure (figure 6) that describes how a typical MBaaS works:

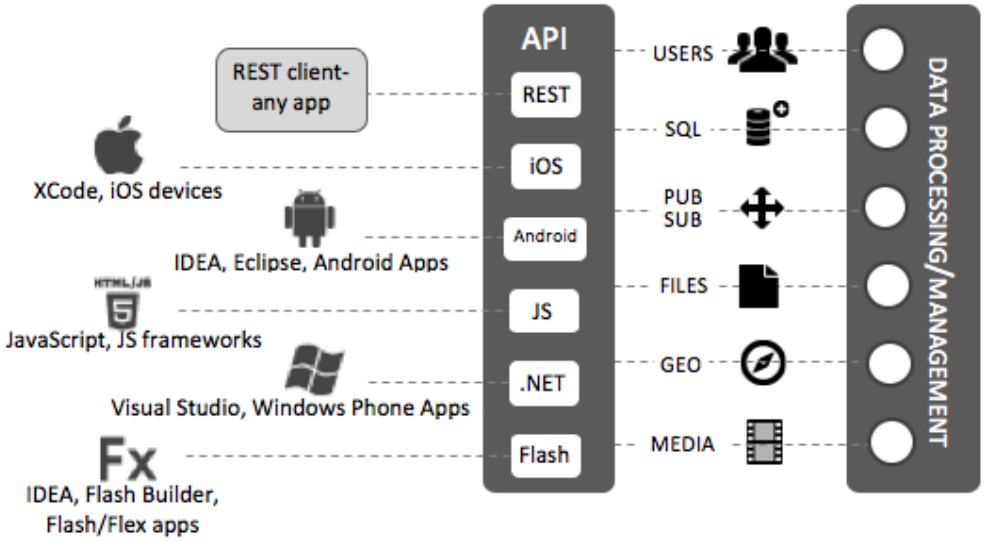

FIGURE 6. MBaaS (Backendless 2016)

3.1.3 SDKs & APIs

According to Beal, an application programming interface (API) is a package of routines, functions, protocols, and tools for building applications. A Software development kit is a set of one or more APIs, programming tools, and documentation that allows a developer to program an application in a platform. (Beal 2016.)

3.2 Parse

3.2.1 Definition

In short, Parse is an MBaaS provider (Jain 2013). More specifically, according to Sukhar (2013), one of the founders of Parse, Parse is a package of cloud services for developers that includes SDKs for different platforms.

3.2.2 History

Parse was founded in 2011 in San Francisco, California by Tikhon Bernstam, James Yu, Ilya Sukhar, and Kevin Lacker. The company continued to grow thanks to investments and fundraising. By August 2011, the company received $1.5 million, funded by 11 organisations and individual investors (Kincaid 2011b). At the end of 2011, from a funding session created by Ignition Partners, $5.5 million was raised for the development of Parse (Kincaid 2011a). In 2013, the San Francisco-based startup was acquired by Facebook for $85 million. However, at the end of January 2016, Facebook put a bullet on Parse that it would be shutting down on January 28, 2017 (Lardinois & Constine 2016). The company has released open source SDKs and tutorials on how to migrate application data to other servers to compensate for their service shutting down (Parse 2016).

3.2.3 Features

Parse has three main features: Parse Core, Parse Analytics, and Parse Push. Parse Core is about data management, social network integration, user management, and server-side logic creation with Cloud Code. Parse Push is notification push feature that helps create messages or notifications and send them to users within a few lines of code. In addition, Parse Analytic tracks any data point in an app in real time. (Parse 2016.)

Parse works many platforms: mobile phones, desktops, web, and embedded systems. On mobile platforms, Parse can be utilised for iOS and watchOS, Unity, Android, Xamarin, Windows, and React. For desktops and webs, Parse has APIs for OS X, JavaScript, .Net, tvOS, Unity, Windows, and Parse. Lastly, Parse supports Qualcomm, Texas Instruments, Atmel, C++, Broadcom, and Intel for the internet of things.

3.2.4 Technology behind Parse

To build the service, Parse uses Amazon Web Services, Ruby, Rails, MongoDB, Redis, MySQL, Chef, Nagios, Cacti, Capistrano, and HAProxy (Rose 2012).

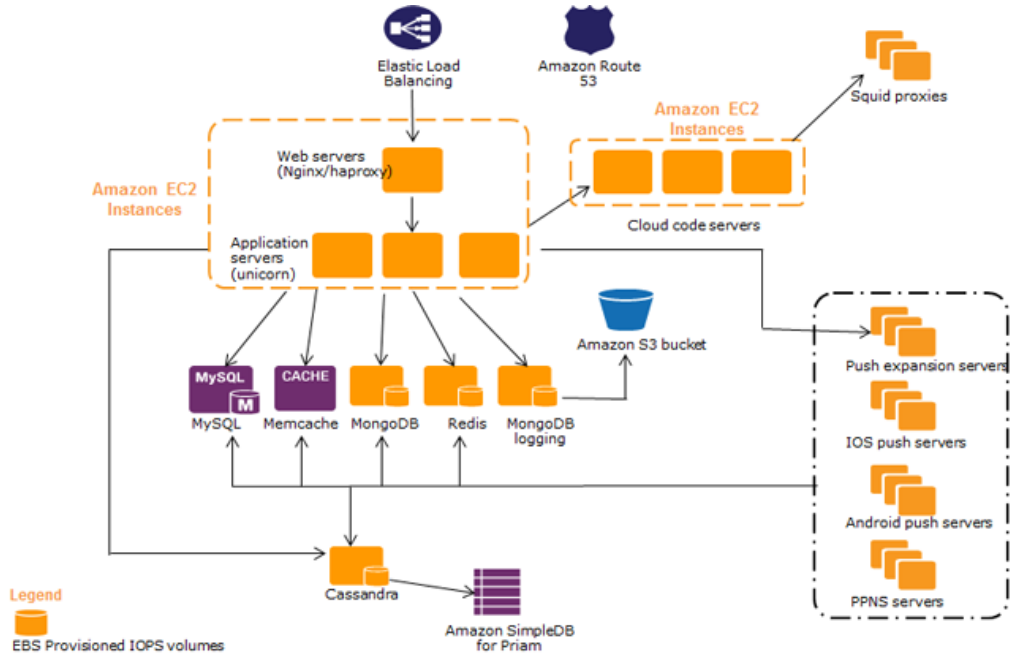

FIGURE 7 Technology behind Parse (Amazon, 2016)

Figure 7 demonstrates the environment that Parse is built on. In this environment, there are striped 1000 Provisioned IOPS volumes for MongoDB clusters running on Amazon Elastic Compute Cloud instance. The Parse engineers also make use of Elastic Load Balancing at the top of the stack to connect to web servers (Amazon 2016).

4 LITERATURE REVIEW

This chapter delves into the literature on the research topic of Mobile Backend as a Service and Parse. The author of this thesis will address the advantages and disadvantages of MBaaS, and follows up with the same approach with Parse. At the end, there will be a summary of the sensible findings used in data collection and data analysis.

4.1 Process of literature review

To create a literature review, this section follows Creswell's guidelines (2009, 29-30). Accordingly, the review is made according to these stages:

- identifying keywords: "mobile backend as a service", "backend as a service", "baas", "mbaas", "parse", and "parse.com"
- searching for appropriate journals and articles with the keywords
- reviewing and construct the findings
- summarizing the findings

While in 4.2 and 4.3, the theories are reviewed and constructed, section 4.4 assembles the findings that fit into the study.

4.2 Mobile Backend as a Service (MBaaS)

In this section, the advantages and disadvantages of using a mobile backend as a service will be examined.

4.2.1 Pros of an MBaaS

Rice, a Kinvey (an MBaaS platform) employee answered on Quora (2012) that the main advantage of an MBaaS is that a developer who has implemented his or her database over an MBaaS can spend more time on developing user experience, functionality, and frontend. The reason is that the MBaaS takes over the complexity of backend. Lane (2012) mentioned that using one of the MBaaS providers eases the setup, operation, and

maintenance process of a backend system. This is because traditional development process involving the backend development increases the development time and application expense (Birani 2013, 8-9).

Using an MBaaS to develop an application saves time and money (Rice 2012). Figure 8 is taken from Kinvey website (2016a). It demonstrates that with MBaaS, the duration of an application development project can be cut down by three-fourths.

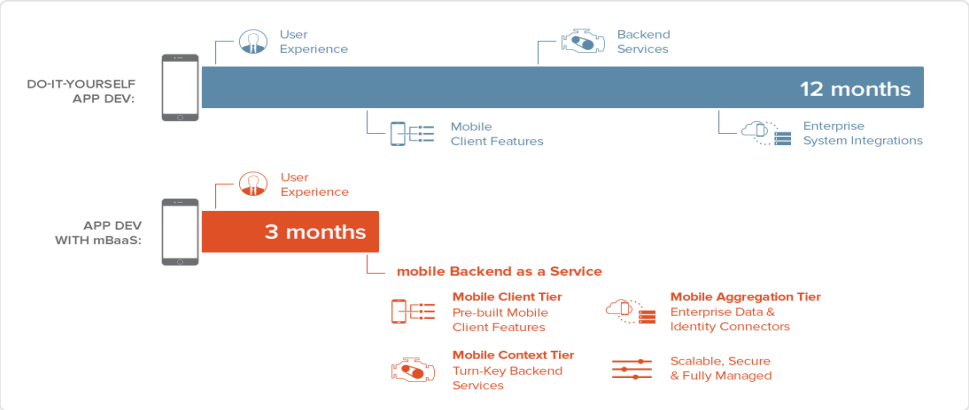

FIGURE 8. DIY and MBaaS duration comparison (Kinvey 2016a)

Furthermore, a calculator comparing the cost, personnel, and work duration of the same project created with both a self-made backend system and an MBaaS can also be found on the Kinvey website. The example in figure 9 is taken from a business-to-customer app. The Android app is built using internal employees to develop. In addition, the application has these characteristics: simple location data, 3-8 screens, a polished user experience and a complete level of security.

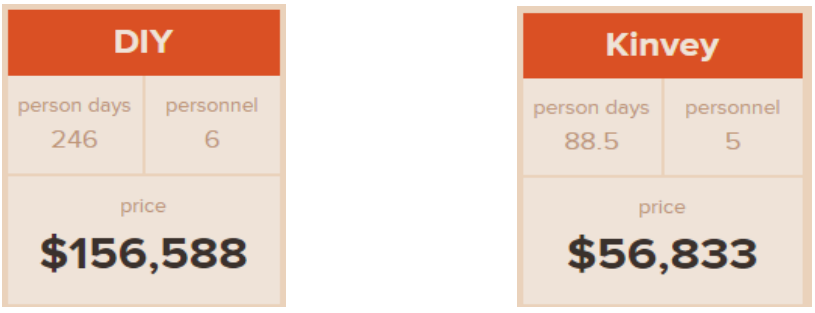

FIGURE 9. DIY and MBaaS price comparison (Kinvey 2016b)

Additionally, a bigger organisation that wants to create a large-scale application does not have to hire a cloud provider like Amazon Web

Service (AWS) or backend developers to setup, operate, and maintain the app's backend system if they implement the application on an MBaaS (Rice 2012). Furthermore, ease and speed of use included in every MBaaS (Balkan 2012) are the reasons for time and money efficiency.

Additionally, an MBaaS does not only consist of data storage, it is also a package of functionalities. Multiple mobile functions, namely geolocations, social networks, notification push, user management, etc., can be found with most MBaaS providers in the market (Rice 2012).

Finally, using an MBaaS helps the organisation focus on its frontend or user experience as the service eases the setup, operation, and maintenance of the backend. This leads to the fact that the user interface, the user experience, and the functionalities, which make an application unique and outstanding, are polished and well-functioned (Rice 2012).

4.2.2 Cons of an MBaaS

When it comes to disadvantages of a service, these are the categories: control, scalability, and shutting-down probability. Firstly, when an organisation has an application running over an MBaaS, there is no complete control over its infrastructure and software stack (APIs, SDKs) (Rice 2012). Secondly, Balkan (2012) mentioned, features of MBaaS are not suitable for large-scale applications. Lastly, the risk that an organisation takes when they use MBaaS, is that the service might get taken down and they will face a severe amount of data migration to a new backend system (Balkan 2012).

4.3 Parse

By the same token of the previous section, Parse's advantages and disadvantages will be delved into.

4.3.1 Pros of Parse

According to Birani (2013, 8-9), the following are the advantages that Parse offers:

1. **Rapid application development**: Parse allows developers to develop applications rapidly with flexible modules.
2. **UX-rich applications**: Parse allows developers to focus on offering a great user experience and not concern about backend maintenance and complex infrastructure.
3. **Powerful data management**: Parse handles everything a developer needs to keep and maintain data securely and efficiently over the cloud. Data types include string, text, photo, or audio, etc. There is a dashboard which includes a graphical user interface to ease the use of Parse.

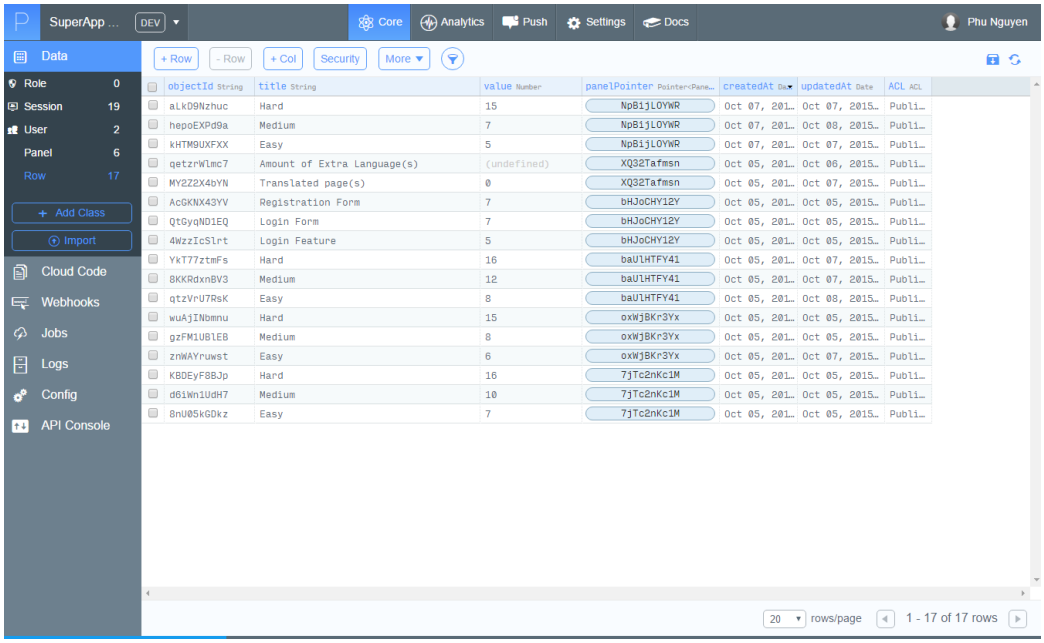

IMAGE 1. Parse dashboard

4. **Make the app social:** developers can connect users via social media sites (such as Facebook and Twitter) with just a few lines of programming code. Developers will not have to care about linking accounts across networks, resetting passwords, and keeping everything safe and secure, because Parse will do it.

5. **Instant push notification**: Parse eases the effort of adding real-time push notifications to an application. It is without much effort for Parse users to send millions of notifications daily.
6. **Run custom app code**: users can add custom logic to the app's backend with Parse's Cloud Code.
7. **One backend for all**: Parse offers a rich compilation of SDKs. This eases the effort to create stunning and powerful apps for different devices and environments.

4.3.2 Cons of Parse

An article about limitations of Parse was written by Dagelić in 2014. Dagelić (2014) mentioned a considerable amount of limitations. Most of these were found from tests, which were examined on a larger scale of context. Here are some of his findings:

1. API requests per minute are limited to 160 for the entire application for free users
2. Parse switches to “approximate count” in counting objects with more than 1000
3. Client-side functions get a timeout error after 3 seconds
4. Server-side functions get a timeout error after 7 seconds
5. Limitations in concurrency tasks
6. Live logging skips logs
7. A limitation of 1000 objects in a request
8. Delay in push notifications
9. Inability to drop a whole class (table) programmatically
10. Parse system gets unavailable from time to time
11. Cloud code limitation with no debugger tool, no option for complex queries

4.4 Literature review findings for analysis

The literature review above has mentioned some of the previous findings of the research question and its sub-questions. They are the advantages and disadvantages of using an MBaaS as well as Parse.

For an MBaaS, the pros are:

- It saves time and money
- It has multiple features in addition to doing backend tasks
- It helps developers focus on frontend development.

The cons of an MBaaS are:

- Limitations in scalability
- Dependency

Specifically, in Parse, the literature review comes up with these pluses:

- Fast to implement
- Convenient data management
- Easy to learn
- Custom server code
- Great SDKs and APIs

For all its good qualities, Parse has also issues with:

- A limitation of requests per minute for free users
- A limitation in programmatical removal of a class
- Cloud code limitation

In the data collection and data analysis, not all the mentioned findings are selected to be in the model lists and evaluated. This is because the unaddressed ones are not relevant to the scope of the chosen case company.

5 DATA COLLECTION AND ANALYSIS

5.1 Case overview

The cases chosen for this study are from SuperApp Oy, including the web application development and three interviews of Parse's users.

5.1.1 SuperApp Oy and interviewees

SuperApp Oy is a small-sized company founded in Lahti in summer 2015. The company focuses on web and mobile technology. Specifically, their products include: prototypes, web applications, mobile applications, and premium websites (SuperApp Oy 2016).

These people were interviewed in this study: Nikula, J., Räihä, M. and Nguyen, T.

Nikula is the chief technology officer in SuperApp Oy. He has experience in programming for 19 years since he was 7, and in creating websites for 17 years since he was 9. At SuperApp Oy he is responsible for research and development, executing customer projects and supervising and consulting other developers and trainees. Although Nikula is young, his experience makes him a very wise and trustworthy man in technical issues.

Räihä has been a web developer in SuperApp Oy since May 2016. Despite the young age of 21 years, he already has 9 years of experience with programming.

Nguyen is a development trainee in SuperApp Oy. He just started working in the company a few weeks before the interview was conducted. He is the least experienced developer in the list as he has only 2 years of programming.

The interviews were conducted on 22nd September, 2016. Because of the interviewees' different backgrounds and knowledge, the amount of time

spent varied from 15 minutes to 30 minutes. In those interviews, the participants were asked about their thoughts of Parse.

5.1.2 Interview questions

The interview questions (listed in appendix 1) are divided into 6 sections:

- general information and experience
- usability, learning progress, and effectiveness
- Parse API
- Parse dashboard usability
- Parse cloud code
- other limitations.

While the purpose of the first section is to get an overview of the interviewees as well as their experience in the topic, the rest of the questions regarding Parse are formed from the theory findings mentioned in subchapter 4.4. First, the “usability, learning progress, and effectiveness” part issues the pros for fast implementation, easy use and learn, cost and time effectiveness, and user experience oriented development. Second, the pros for the great SDKs and APIs are related to the interview questions in “Parse API”. Third, the “Parse dashboard usability” addresses the pros of convenient data management and one of the multiple features in addition to doing backend tasks. Fourth, the “Parse Cloud code” section issues the Cloud code limitations. Finally, the “Limitations” group up the rest of the cons: a limitation of requests per minute for free users, limitations in a multitude of data, requests or queried objects, a limitation in programmatically remove of a class and the dependency.

5.1.3 Website Price Calculator (WPC)

The Website Price Calculator (WPC) case was chosen because this application made use of Parse. In addition, the reason also is that the author is the main developer of this application throughout its development phases. During the development phases of WPC, the author was working as a web development trainee in SuperApp Oy.

At the beginning of the project, Website Price Calculator was a web application developed to serve as a tool for customers who want a website created by SuperApp. Customers can input the wanted categories and see the hours and expenses SuperApp needs to create their desired product.

This case is separated into two subcases accordingly to the two main phases of the development: WPC version 1 (v1) and WPC version 3 (v3). WPC version 2 is not mentioned because it is a fully functional offline version before Parse integration.

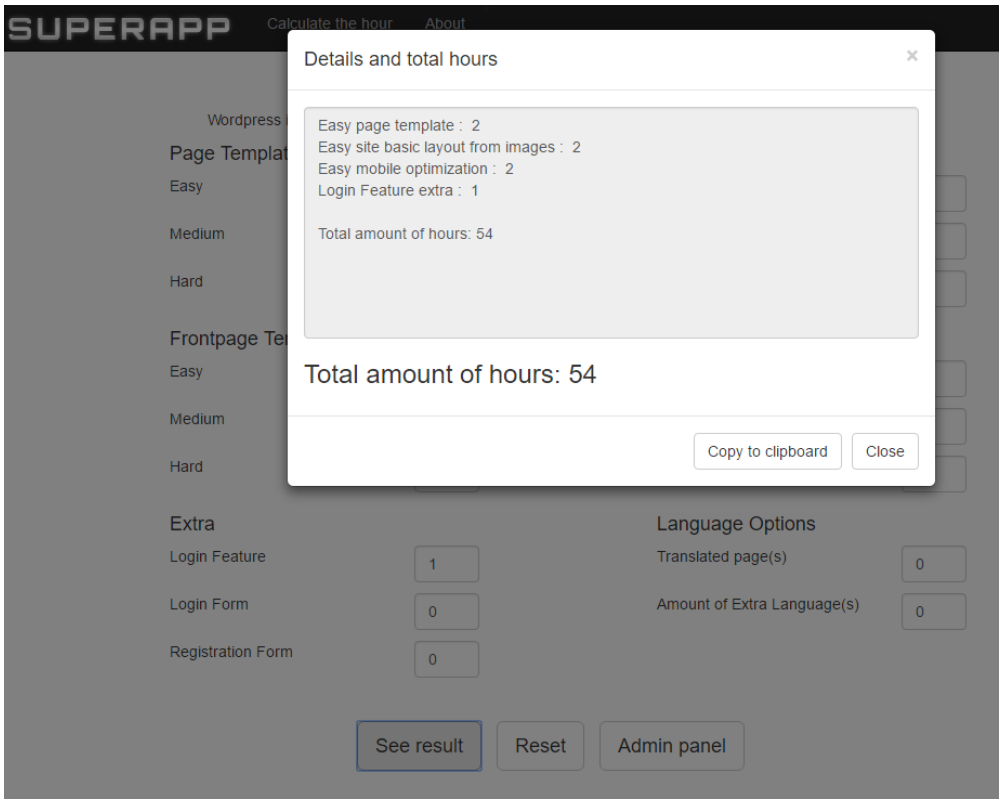

IMAGE 2. WPC v1 – result view

The main difference between two versions is the customisation capability. In WPC v1 it is very limited for an admin to add custom checkboxes, options, lists, or drop-downs. However, in WPC v3, users can create their own calculator with a logo and a description. Additionally, they can then add, edit, and delete input fields, dropdowns, checkboxes, and customisable logic fields.

IMAGE 3. WPC v3 – result view

As the project was developed using Agile Development Method, documentation from the development is saved and recorded through a project management tool named Trello. This tool resembles Agile Development Method, with several phases (sprints) such as Todo, Doing, Done, etc… Members can create, edit, and move the items (tasks) from one phase to another. On the other hand, the information of the phases a task belongs to is also recorded within an item.

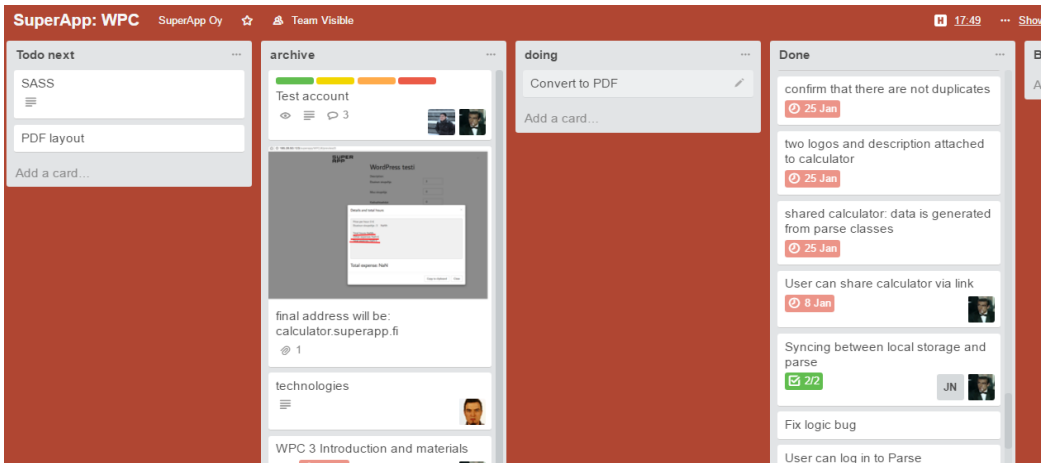

IMAGE 4. WPC Trello board

5.2 Data reduction

5.2.1 Website Price Calculator

In this chapter, five pros of Parse and MBaaS are gathered from the WPC project's management tool. The advantages are: time efficiency, fast implementation, an approved addition feature, development focusing on user experience, and solid APIs.

Progress and duration of tasks are deduced by analysing the logs in Trello. By examining the logs, the author determines that using Parse saves time and implementing Parse is rapid.

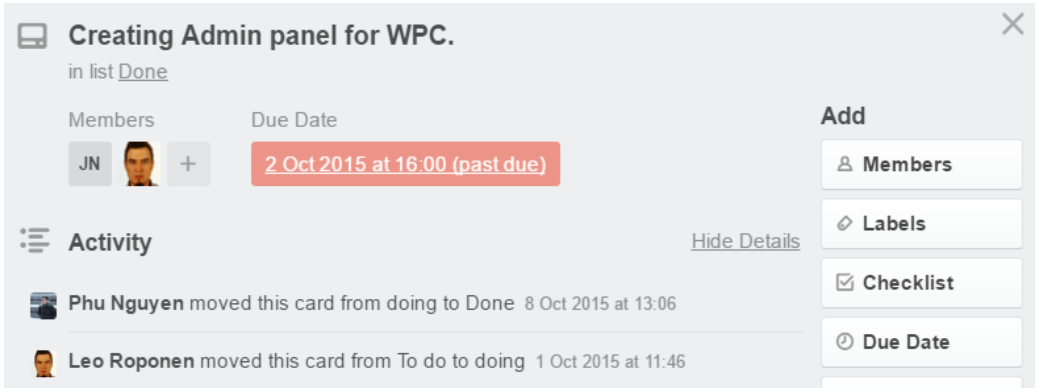

IMAGE 5. WPC v1 creating admin panel card

Regarding the log in image 5, the admin panel (image 6), which can be used to add, edit, and delete data from a server, was developed swiftly in a week. In addition, at that point, the developer had no previous experience in working with Parse.

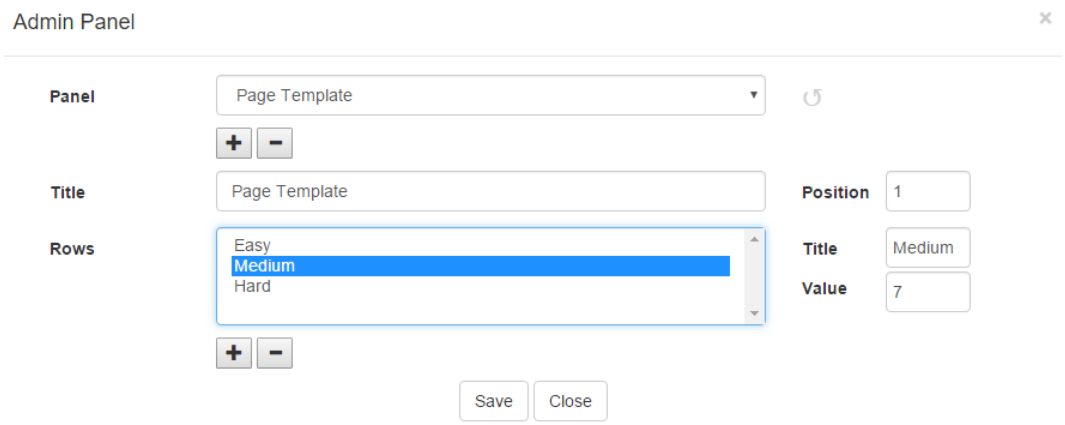

IMAGE 6. WPC v1 admin panel

In WPC v3 it took three weeks for all functionalities to be implemented. The functionalities are login, calculator sharing, and synchronisation and interaction between offline storage and Parse server. Without Parse, it is impossible for the developer to focus only in the functionalities.

In all versions, the developer could integrate Parse authentication with ease. This has worked flawlessly, and with Parse it was nearly effortless to implement user authentication.

Interaction and synchronisation between local storage and server cost most of the time in the development phase of WPC v3, because of the structures of the database and characteristics of JavaScript. In the data structure, there are items and subitems. These objects require asynchronous functions which synchronise an item and its subitems between the server and the application. However, JavaScript does not normally handle asynchronous functions. This means it does not wait for a request from a server to execute, but jumps to next functions.

Parse provides an API feature called “Promise”, easing the attempt to make codes run asynchronously so that the developer could implement the synchronisation between local storage and server data. The code snippet in image 7, taken from a function that downloads data from a Parse server, describes the use of Promise in action:

```
var Calculator = Parse.Object.extend("Calculator");
var query = new Parse.Query(Calculator);
query.equalTo('objectId', $scope.parseID);
query.find().then(function (calculators) //promise use
{
        var calculator = calculators[0];
        calculator.set("name", $scope.calcName);
        calculator.set("price", $scope.calcPrice);
        calculator.set("logo", $scope.logo);
        calculator.set("description", $scope.description);
        console.log(calculator.get('name') + 'Calculator will be saved');
        return calculator.save(); //promise use
}).then(function (calculator) //promise use
{
        if ($scope.items.length != 0){
          $scope.saveItem(0, $scope.items.length, calculator);
        } else {
          saveLocalStorage();
          $('#successSaved').modal('show');
        }
});
```

IMAGE 7. Parse promise in action

In image 7, Parse Promise uses are highlighted in green. Not only do Promise makes programmers’ codes look clearer, but they also make functions easier to follow for code reviewers. Developers using Parse can profit from this feature of the Parse API.

5.2.2 Nguyen’s interview

Nguyen was not able to answer all questions, because his experience in programming, web app development, and Parse is limited. At the time of the interview he said that he had only been programming for two years, programming web apps for one month, and knowing Parse for two weeks. However, he gave decent answers to questions about Parse in usability and learning progress, and the Parse dashboard usability.

As an inexpert developer, Nguyen summed up his experience in the learning progress of Parse:

> *It's basic and easy to use. There is a guide which is perfectly easy to follow then I found it not so difficult. (Nguyen 2016.)*

To be specific with a real task, he had to do a small application that required “Login and Signup” and he said that it was “amazing” that he did that in only two days.

As a new Parse user, Nguyen has some points about the Parse dashboard:

> *Creating a column cannot be easier than ever, then row and absolutely easy class to look up, fix, and delete. It is practical at some points. (Nguyen 2016.)*

In addition, he thought that the dashboard has a “nice UI, easy access and followup”.

5.2.3 Räihä’s interview

Räihä replied most of the questions in the interview, except the section about Cloud code because he did not use it.

Räihä is one of the newer employees SuperApp Oy has hired. He has been coding for nine years and developing web applications for approximately five years. Even though he has been using Parse for only five months, his programming experience and his talents in IT field make his answers very fitting.

As an experienced developer, Räihä thinks that Parse “is very easy to learn”. In addition, it took him only three hours to get to know Parse. Moreover, within one hour of the training, he managed to create an application, that has user authentication, registration, and notifications on the site when a user logs in. Besides, according to Räihä, a developer

would take at least three times longer to create the same application with his own backend system.

In terms of money and time saving, in Räihä's opinion, Parse is "pretty effective" in cost and time effectiveness. He also mentioned:

> *You don't need to do any backend development so you can just focus on the front end. The SDK is very simple to use and to learn. (Räihä 2016.)*

Regarding the Parse API, for Räihä, Parse provides a "very solid, easy to learn and easy to use" API. An example of this is the Parse Promise, which "is awesome", as stated by Räihä, "because you can avoid callback hell (too many nested codes)."

The interview also reveals his thoughts about the Parse dashboard:

> *Parse dashboard is clearer, easier to use, and simpler compare to phpMyAdmin (Räihä 2016).*

In Räihä's interview, some questions about Parse's limitations were answered. Firstly, he was not in any situation that required programmatic class removal. About the fact that Parse does not have a function to remove the tables using code, he mentioned:

> *At some cases, it is a big minus. Especially when they need a temporary table or something similar. (Räihä 2016.)*

Secondly, in regards to Parse limitation about the case when there is a limit for requests per minute, his thought about Parse is that:

> *Parse is kind of for small scale applications, it is just enough for development. For production, you will need more. (Räihä 2016.)*

Finally, he does not like the fact that Parse service is closing. If he had apps using the service, it would be a big thing for him. This is because "it is a pain to migrate and handle those". Moreover, he also gave a decent point about the termination of Parse that "it is also good that they release the open source so now people can develop freely what they want."

5.2.4 Nikula's interview

Not only did Nikula complete all the questions, but he also gave thoughtful answers and further information about Parse. Specifically, information about Parse migration plan and opportunities in Parse open source was specified.

Nikula has decent experience with Parse. He has been using Parse for the past two and a half years.

Talking about how fast one can get to know Parse, he pointed out that:

> *This is the best part of Parse. It's very easy with prior knowledge of programming and database because Parse handles everything else such as authentication for you. And it took only a few hours to learn the basics of Parse. (Nikula 2016.)*

The first application he created using Parse was a mobile application based on web technologies. It took him eight hours to build the first version of this application. In addition, he also mentioned that without Parse, it would take one developer with advanced programming skills at least a week to build the same app with a backend with the same level of security and scalability.

Concerning cost and time effectiveness, Nikula had a word for Parse: "miraculous". In addition to that, Parse is high quality:

> *It's 'miraculous' to use Parse in terms of time, money, and also quality. It's very good for businesses. It's now getting cheaper and faster to build applications with Parse or similar backend services. (Nikula 2016.)*

Nikula has a similar thought about the API as the previous interviewees:

> *The API has a very deep tutorial in a very broad range of languages. Basically, it is pretty good. With cloud code, you can even create your own custom APIs, which is very flexible. (Nikula 2016.)*

In addition, he also answered that Parse is a very precious tool because “asynchronous requests are a very big part of using Parse and Parse Promise makes them look more readable.”

Regarding usability points of the Parse Dashboard, he pointed out that:

> *It is simple, readable, very user-friendly. People who aren't tech guys can also use it easily. (Nikula 2016.)*

By the same token, another thing that he likes about Parse is that a user can set class level security from the dashboard. This feature does not yet exist in other backend dashboards, such as phpMyAdmin.

About the Parse cloud code, Nikula showed much interest in this topic. This might be because 70% of his Parse projects use Cloud code. On the other hand, when answering a question about the limitation of the Parse cloud code, he stated that:

> *It is impossible to install extra frameworks into the Parse server (Nikula 2016).*

When talking about other limitations of Parse, Nikula seemed to favor Parse. Firstly, he did not think that the inability to remove a table programmatically is a big minus. Secondly, in his opinion, Parse scalability is one of their best features as:

> *Creators don't have to forecast what they will need when they do it on their own. Everything is handled by Parse service. (Nikula 2016.)*

Thirdly, in Nikula’s point of view, the limit in requests per minute in Parse is also not a big issue. He explained:

> *This isn't a big problem because 160 API requests per minute fit 80% of the projects. It also depends on the project types. If an app is for a company use, there aren't a lot of users then it's good. But if an app is advertised in public, and thousands or millions of people will use that, this isn't a good idea. (Nikula 2016.)*

Lastly, he admitted that he was sad about the fact that Parse is closing their service. However, he also hinted at a positive point in this service termination:

> *They handle it well by providing the open source platform. We don't have to rebuild all the backend; we only need to migrate those. In addition, we can improve the Parse open source to our needs. For example, now I can install my own frameworks in the server so cloud code can handle more. (Nikula 2016.)*

5.3 Data analysis

5.3.1 Summary

In table 1, a summary of issues: which are approved, which are disapproved, which are both, and which are unanswered, are gathered from the data reduction section in 5.2. Approved issues are notified with a thumb up (👍), disapproved issues with a thumb down (👎), both approved and disapproved issues with a neutral emotional icon (😐), and unanswered issues are marked with a dash (-).

			Nguyen	Räihä	Nikula	WPC
Pros	MBaaS	Time & money efficiency	👍	👍	👍	👍
		Additional features apart from backend	👍	👍	👍	👍
		User experience oriented development	👍	👍	👍	👍
	Parse	Swift implementation	👍	👍	👍	👍
		Convenient data management	👍	👍	👍	-
		Cloud code	-	-	👍	-
		Solid APIs and SDKs	👍	👍	👍	👍
Cons	MBaaS	Scalability limitations	-	-	👎	-
		Dependency	-	👍	👍	-
	Parse	Free user limitation	-	👍	😐	-
		Inability to remove a class	-	👍	👎	-
		Cloud code limitation	-	-	👍	-

Approved: 👍, Disapproved: 👎, Neutral: 😐, Unanswered: -

TABLE 1. Data analysis summary

5.3.2 Analysis

In this section, the author goes through the issues mentioned in 4.4 and analyses those with the findings from the collected data.

5.3.2.1 Pros

Most of the pros are fully approved. Some of the questions regarding them were not answered because the interviewees were not experienced in the subjects.

Firstly, time efficiency and cost efficiency are approved in all the cases. Even though the matter of cost was not directly mentioned, it is possible that time efficiency, in any case, will lead to a decrease in costs. Moreover, Nikula also mentioned that Parse added quality into this duo.

Secondly, the additional features that Parse provides apart from doing only backend tasks are partly confirmed. In the interviews and the web application case, the data shows that Parse functions well in user authentication. The other features of Parse were not mentioned because they are out of the company's scope.

Thirdly, the advantage that an MBaaS will help the developers focus on frontend was proven in the research. Apart from Räihä, who directly mentioned this in his interview, the others implied the same. The data also shows that it took the developers a short time to implement the frontend without having to worry much about the backend.

Fourthly, the fast implementation, the convenient data management tool, and solid APIs and SDKs are substantiated in the interviews and in the case.

Finally, the questions about custom backend code were not answered by all the cases. Only one over four cases approved the cloud code feature from Parse.

Besides, Nikula also stated that Parse has a decent scalability. This means users can scale up the application with ease and develop the application in a wide variety of platforms, such as webs, mobiles, and computers.

5.3.2.2 Cons

The chosen disadvantages from the literary findings were not so approved. In addition, the questions concerning them were answered mostly by Nikula and Räihä, the two most experienced developers.

Nikula and Räihä both agreed on the dependency in an MBaaS. The pain of handling the data migration in Parse case is undeniable for the interviewees. By the same token, Nikula also commended on Parse action after their shutdown: Parse “open sources” their backend service.

The interviewees did not agree as much on Parse’s inability to programmatically remove a class. Therefore, this is not counted as a valid disadvantage.

Concerning the limitation in requests per minute for free users, Räihä and Nikula equally agreed and disagreed. Accordingly, this disadvantage is inconclusive.

The final issue of the listed cons, the cloud code limitation, was approved by Nikula.

6 RESULTS

This chapter concludes the findings of the study.

6.1 Sub-question 1: What are the advantages of an MBaaS and Parse?

Based on the case study and three interviews, it can be concluded that MBaaS and Parse have several advantages.

Using an MBaaS is beneficial for companies like SuperApp. A company can save time and money by adopting an MBaaS. Additionally, an MBaaS not only supports basic database tasks, it also offers extra features for users such as user authentication. Moreover, the most beneficial advantage of using an MBaaS is that the developers can focus on developing the utilities and the look of a product without caring much about the backend.

On the other hand, Parse has four main advantages. Firstly, developers with different experience ranges have nearly the same steady and effective learning progress with Parse. Secondly, Parse has a convenient dashboard with some extended features for managing the data. Thirdly, Parse users can create their own APIs using the cloud code. Lastly, the APIs and SDKs Parse offers were proven to be solid, easy to understand and easy to use.

6.2 Sub-question 2: What are the disadvantages of an MBaaS and Parse?

When it comes to cons, the study showed two main issues: service dependency and cloud code limitation.

Regarding MBaaS, a company that uses such a service must rely on its availability. Therefore, it is a problem, if the service is halted as this leads to data migration.

Regarding the cons of Parse, the only confirmed issue was the cloud code limitation. In other words, the users do not have the full control to servers. This means that basic functions can be programmed on the server side with the cloud code but sophisticated programs based on other frameworks on a Parse server cannot be developed.

6.3 Research question: What are the advantages and disadvantages of Parse in application development?

By using Parse in application development, developers or companies

- save time and money
- can implement the Parse service easily
- can have steady and fast learning progress with Parse
- have more features than just a system that handles backend
- can focus on the frontend without taking much care of the backend
- have solid tools to handle the backend system (APIs, SDKs, and the dashboard)

Parse also has disadvantages which are

- backend code limitation
- dependency on the service.

7 DISCUSSION

In this section will be discussed the limitations, validity, and reliability of this thesis, and suggestions for future studies. Regarding validity and reliability, the study relied on the terms' defnitions mentioned in "Validity, reliability, and generalizability in qualitative research" (Leung 2015).

7.1 Limitations

A limitation of this thesis is the restriction in the scope of the case company. SuperApp Oy is a small sized company which creates prototype software and web applications, and uses Parse only with one framework for web application development. The issues that were chosen to be analysed in the data collection and analysis are not identical to all possible findings in the literature review. An example of this is the restraint in using multitude amount of data and requests, or the advantage of Parse being compatible with various platforms, which where not analysed.

The references in the thesis are mostly digital references that might not be reliable. However, because the topic is new, it was difficult to find books or more reliable sources. In this case, only one book regarding Parse was referenced.

7.2 Reliability

Reliablity in qualitative research refers to the similarity of outcomes albeit the difference in procedures (Leung 2015).

This research is reliable. In the data collection, three different developers with dissimilar programming experience were asked to give opinions about Parse. In the end, they gave similar opinions of certain points in the issues. A good example of this is that they all agree with that Parse is easy to learn and implement.

7.3 Validity

A qualitative research is valid if the methods, stages, and information of it are appropriate. For example, to determine if a research is valid, the appropriateness between the research question and the wanted results or the study framework and research, is accessed. (Leung 2015.)

The paper meets several requirements to be valid. This study's topic concerns an IT service, and the case study chosen is an IT company, which has been using that service. Furthermore, the presumed results, mentioned at the beginning of the research, were the pros and cons of the IT service. The expected findings were concluded at the end of this paper.

7.4 Suggestions

The fact that Parse "open-sources" its platform leads to some topics for further study. Following are the suggested problems regarding Parse, an **open source** backend as a service:

- Migration of an existing Parse backend to a new server using open source Parse backend
- Installation of an open source Parse backend
- The advantages and disadvantages of using an open source Parse backend.

LIST OF REFERENCES

Printed sources

Birani, B. 2013. Application Development with Parse using iOS SDK. Birmingham: Packt Publishing.

Creswell, J. 2009. Research Design Qualitative, Quantitative, and Mixed Methods Approaches. Thousand Oaks: Sage Publication.

Eriksson, P. & Kovalainen, A. 2008. Qualitative Methods in Business Research. Thousand Oaks: Sage Publications.

Engel, R. & Schutt, R. 2012. The Practice of Research in Social Work. Thousand Oaks: Sage publications.

Kozma, R. & Schank, P. 1998. Connecting with the twenty-first century: Technology in support of educational reform 1998, 3-30.

Saunders, M., Lewis, P. & Thornhill, A. 2009. Research Methods for Business Students 5th edition. Harlow: Pearson.

Webster, J. & Watson, R. 2002. Analysing the Past to Prepare for the Future: Writing a Literature Review 6/2002, xiii-xxiii.

Yin, R. 2009. Case Study Research: Design and Methods 4th edition. Thousand Oaks: Sage publications.

Digital sources

Amazon 2016. Parse technology at Amazon Web Services. [accessed 5 March 2016]. Available at http://aws.amazon.com/solutions/case-studies/Parse/

Avram, A. 2014. IDC Study: How Many Software Developers Are Out There? [accessed 03 September 2016]. Available at https://www.infoq.com/news/2014/01/IDC-software-developers

Backendless website 2016. [accessed 13 March 2016]. Available at https://backendless.com/what-is-backend-as-a-service/

Balkan, A. 2012. What are the pros and cons of using a backend as a service? [accessed 25 March 2016]. Available at https://goo.gl/lvgLL0

Beal, V. 2013a. SDK - software development kit definition. [accessed 13 March 2016]. Available at http://www.webopedia.com/TERM/S/SDK.html

Beal, V. 2013b. API – application programming interface. [accessed 13 March 2016]. Available at http://www.webopedia.com/TERM/A/API.html

Carney, M. 2013. AnyPresence partners with Heroku to beef up its enterprise mBaaS offering. [accessed 10 March 2016]. Available at https://pando.com/2013/06/24/anypresence-partners-with-heroku-to-beef-up-its-enterprise-mbaas-offering/

Dagelić, A. 2014. All the limits of Parse. [accessed 7 October 2016]. Available at http://profi.co/all-the-limits-of-parse/

Jain, P. 2013. What is Parse.com? [accessed 3 March 2016]. Available at https://www.quora.com/What-is-Parse-com

Kincaid, J. 2011a. Parse the Heroku for mobile raises 5.5 million series. [accessed 1 March 2016]. Available at https://goo.gl/HEX9Mo

Kincaid, J. 2011b. YC funded Parse – a Heroku for mobile apps. [accessed 2 March 2016]. Available at https://goo.gl/V87p6g

Kinvey 2016a. [accessed 2 March 2016]. Available at http://web.archive.org/web/20160219064959/http://www.kinvey.com/why

Kinvey 2016b. Kinvey mBaaS Savings Calculator. [accessed 4 March 2016]. Available at http://calculator.kinvey.com/mbaas-savings-calculator

Lane, K. 2012. Rise of mobile backend as a service. [accessed 3 March 2016]. Available at https://goo.gl/4HEXQX

Lardinois, F & Constine, J. 2016. Facebook shutters its Parse developer platform. [accessed 7 March 2016]. Available at: https://goo.gl/O5qFqj

Leung, L. 2015. Validity, reliability, and generalizability in qualitative research. [accessed 18 November 2016]. Available at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4535087/

Monroe, M. 2013. The Gospel of MBaaS. [accessed 13 March 2016]. Available at https://goo.gl/e3eAIw

Parse archive 2016. Commercial website [accessed 6 March 2016]. Available at http://web.archive.org/web/20160111030706/http://parse.com/

Rice, K. 2012. What are the pros and cons of using a backend as a service? [accessed 23 March 2016]. Available at https://goo.gl/lvgLL0

Statista 2016. Number of software developers employed in the United States in 2010 and 2020 (in 1,000). [accessed 3 September 2016]. Available at https://goo.gl/Ri2JZT

Rose, S. 2012. What is the technology behind Parse? [accessed 3 March 2016]. Available at https://goo.gl/hhhfid

Sukhar, I. 2013. What is Parse.com? [accessed 3 March 2016]. Available at https://www.quora.com/What-is-Parse-com

Website live stats 2016. [accessed 06 November 2016]. Available at http://websitelivestats.com

Website price calculator v1. [accessed 5 March 2016]. Available at http://77.240.19.15/~play1sueprapp/WPC/

Website price calculator v3. [accessed 6 March 2016]. Available at http://calculator.superapp.fi/

Oral sources

Nikula, J. 2016. Chief technology officer. SuperApp Oy. Interview 22 September 2016.

Nguyen, T. 2016. Trainee. SuperApp Oy. Interview 22 September 2016.

Räihä, M. 2016. Developer. SuperApp Oy. Interview 22 September 2016.

APPENDICES

APPENDIX 1 Semi-structured interview questions

1. **General information and experience**
    a) What's your name?
    b) How long have you been programming?
    c) How long have you been programming web applications?
2. **Usability, learning progress and effectiveness**
    a) When did you first know Parse or any similar backend as a service?
    b) How do you see your learning progress with Parse? How long did it take you to get to know Parse?
    c) What was the first application (or tasks) you do with Parse? How long did it take?
    d) How long does it take to build a web app with Parse and with a Do it yourself backend?
    e) How effective you think it is to use Parse in terms of money and time?
3. **Parse API**
    a) Do you use Parse promise? Do you think Parse promise is a good feature that is included in the Parse API? Why?
    b) What do you think about the API in general?
4. **Parse dashboard usability**
    a) Could you describe the Parse dashboard from the point of usability?
    b) Which things do you like about the Parse dashboard?
5. **Parse cloud code**
    a) How often in a project did you use the Parse Cloud code?
    b) What are the features/characteristics you think Parse cloud code should have had?
6. **Parse limitations**
    a) Have you ever needed to remove a class (table) programmatically

b) Parse does not have a function to remove a class using code, do you think this is a big minus? Why and why not?
c) What do you think about Parse scalability?
d) There is a limitation (160 API requests per minute for a whole app) for free users, do you think it is relevant? Why and why not?
e) What do you think about the fact that Parse is shutting down?