REM

Mobile Software Engineering in a Business Context

RELATÓRIO DE ESTÁGIO DE MESTRADO

Ricardo Alexandre Gonçalves Pereira
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

novembro | 2018

Mobile Software Engineering in a Business Context

RELATÓRIO DE ESTÁGIO DE MESTRADO

Ricardo Alexandre Gonçalves Pereira
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
Pedro Filipe Pereira Campos

Mobile Application Software Development

Ricardo Alexandre Gonçalves Pereira

Constituição do júri de provas públicas:

Karolina Baras, (Professora Auxiliar da Universidade da Madeira), Presidente

Diogo Nuno Crespo Ribeiro Cabral, (Professor Auxiliar da Universidade da Madeira), Vogal

Pedro Filipe Pereira Campos, (Professor Associado com Agregação da Universidade da Madeira), Vogal

Janeiro, 2019

Funchal – Portugal

Internship Report

Mobile Application Software Developer

Internship Report Submitted to University of Madeira for the Purpose of Obtaining the Degree of MSC Software Developer

Ricardo Alexandre Gonçalves Pereira

Funchal, Portugal

November 2018

To my family for all the support and motivation

Given to me.

And to everyone else, who helped me directly and indirectly.

Thankful To

First and foremost, my family, they supported me all the way and never doubted me even when I doubted myself, I would not be here if not for them. My never-ending appreciation goes to them.

Secondly to all my class-mates, even the ones from a different course, we helped each other the best we could, and I thank you for it.

A special thanks to Equilab for the opportunity, and for all the knowledge they poured into me, it was without a doubt a great first step into my professional career. Also, to *Vaquinha* and *Nº2*, both bars I worked in and where I learned a lot of the real world, these lessons shall accompany me the rest of my life.

And finally, to all my professors and advisor who poured their love of the field into me, they helped me have great respect and love for everything I learned and for all I am going to learn.

Abstract

The smartphone application industry has been on a steady rise all over the world, creating new jobs every year, in one of these companies I realised a 5-month internship in Sweden, developing a new feature for their app using React Native and Firebase. In this internship not only did I improved my programming skill but also gained invaluable insight on the industry, agile environments, how to better work as a team and the responsibility of developing a new feature for an established app on the market.

In this dissertation I shall go over the history of the app, some stats of the industry, the agile methodology, the tools, frameworks and Services used to develop the feature I was trusted with, the day-to-day life of an agile developer in Equilab and how we overcame some issues.

Resumo

A indústria de aplicações moveis tem estado num crescimento constante por todo o mundo, criando novos postos de trabalho todos os anos, foi numa destas empresas que realizei meu estágio profissional de 5 meses na Suécia, desenvolvendo uma noca característica para aplicação utilizando maioritariamente React Native e Firebase. Durante o estágio aperfeiçoei não só minhas habilidades como programador, mas também ganhei conhecimento da indústria, trabalhei num ambiente Agile, trabalhar melhor em equipa e lidar com a responsabilidade de desenvolver uma nova característica numa aplicação líder do seu mercado.

Nesta dissertação eu irei abordar a história das aplicações, algumas estatísticas sobre a indústria, a metodologia Agile, as ferramentas, frameworks e serviços usados para desenvolver a característica com a qual eu fui confiado em realizar, também irei falar como era o dia a dia na vida de um desenvolvedor Agile na empresa Equilab e como superamos alguns dos problemas encontrados

Keywords

App

Developer

Mobile

React Native

Firebase

Agile Development

Palavras-Chave

App

Programador

Telemóveis

React Native

Firebase

Agile development

Acronyms

API - Application Programming Interface

ASP - Application service Provider

BaaS – Backends-as-a-service

CSS - Cascading Style Sheets

IT - Information Technology

OS - Operating System

RN – React Native

SaaS - Software as a Service

Index

1- Motivation

Sometimes while on my academic formation the question of “how will my job be?” arose multiple times, specially every time something new was learned, it seemed like a whole world of knowledge was always ahead of me, with more new things coming up every semester making a relative small world view at first seem to be growing larger, “how will I be able to keep all this knowledge?” this in turn made the possibility of making an internship grow bigger and bigger to the point it seemed to be the best option, to be sure I had what it takes to make it on a global business.

The idea came from a few friends, since I didn’t even know of its existence, to go abroad and do an internship with help from ERASMUS+[1] program. It sounded like a great idea the moment I heard it, I could test my skills professionally and I could test them in a new environment

And so, after the results for the application to the ERASMUS+ scholarship came out, the second phase began, and that was to search for a company that were in need for an intern in their ranks. Most of the search went on the Erasmus intern site[2], but also on training Experience[3], ESPA[4] and ISPO[5]. It was easily more than 30 applications sent to all over Europe, with most of the focus being on the UK, Germany, Spain and Slovakia. Eventually the lucky draft was for Sweden, a small company focused on a single phone APP, Equilab[6]. Before arriving I had already been told I was going to be working on a new feature for the app and it was going to be developed using React Native[7], and also to use Firebase[8] as the database. Equilab is also an app that lets you keep track of the gait, beat, stride, distance made by the horse and other features were being developed at the time of my internship. While running the APP the phone should be on a tight place, like an armband or your pants pocket and let the APP do its magic, the information can be used by both *joyriders* and professionals, students and teachers, and even by people using horse only to pull caravans and other vehicles like that.

Nothing in the world can take the place of persistence. Talent will not; nothing is more common than unsuccessful men with talent. Genius will not; unrewarded genius is almost a proverb. Education will not; the world is full of educated derelicts. Persistence and determination alone are omnipotent. The slogan Press On! has solved and always will solve the problems of the human race.

Calvin Coolidge

2- State of the Art

Phone App

Phone App History

Prior to 1973, “mobile phones” were limited to cars and other vehicles, until Motorola produced the first handheld mobile phone and making the first call on April 3, 1973, but the real “father” of smartphones and consequently the rise of app’s was thanks to IBM when in 1994 launched IBM Simon[9]. Fast forward to 2012 and phone calls are placed in fifth as list of things done on smartphones, ranking behind browsing the Internet, using Social Media, listening to music and playing games, and texting taking the seventh position on the list, staying behind sending emails[10]. So, what happened that made a device with the sole purpose of making phone calls make that very act be left behind in the list?

Figure 1 - Time spent on average on smartphones[10]

Well the phrase “There’s an APP for that” probably had a thing to do with it, but how did we even get here? Who was the person behind it? Was it a company? Was it a bloody revolution? Well nothing as romantic as that I’m afraid, but it did spur from someone's idea/dream, that “infected” so many others in a profession so young, yet so powerful. I think four big categories are important to understand the life of APP.

First, we shall see how thing were in the 25 years prior to the launch of the App store, who had the foresight or capability of taking a small glimpse to the future and how little by little the APP

was going to be born.

Then we move onto the full-blown Information Appliance era, who laid the foundation for the APP to spread like wildfire not only to mobile devices mind you.

We go through the Home screen era, where the APP “cemented” itself in the house and how it conquered the world from here, reaching numbers probably unimaginable a decade before.

And finally, the “current” era, the APP’s as a service era, for a few years it’s been noted the “death” of a develop and release kind of APPs, most users are expecting their apps to be constantly being updated and with more functionalities to be added constantly, otherwise most APP will just lose their user’s interest, as new APP are being released.

Before apps, predictions and origins 1983-2007

Who was the mastermind behind that which is revolutionizing our world, who is the one responsible for giving humans (at least those capable of buying a smartphone) an almost unlimited access to information, the capacity to use a device for recording audio and video, security, the list goes on and on.

Well we can't really attribute this only to one person or company, even though it sounds pretty cool, but Steve Jobs sure played a great role in it, his vision almost 35 years ago was detrimental to having it as it is, even though using his own time schedule it was supposed to take only one decade instead of two or three.

- June 1983 Steve Jobs predicts a software distribution system like a record store and also smartphones *"Apple's strategy is really simple. What we want to do is we want to put an incredibly great computer in a book that you can carry around with you and learn how to use in 20 minutes. … And we really want to do it with a radio link in it so you don't have to hook up to anything and you're in communication with all of these larger databases and other computers."*.[11]
- January 1987 Psion EPOC, an early handheld computer with an Operating System which had basic applications such as a diary.[12]
- January 1996 The Palm OS (also known as GarnetOS), although it was not the first PDA, it was the one that launched an industry, penetrating popular culture and making the way for new devices, it was designed for ease of use with a touchscreen-based graphical user interface. Later versions of the Operating System have been extended to support smartphones. [13]
- December 1997 The nokia 6110, even though snake is mid-1970 and was called Blockade, it was with nokia that Snake got into people's phones and hence their pocket.[14]
- October 1999 WAP (wireless application protocol), a specification for a set of communication protocols to standardize the way that wireless devices can be used for access to internet and instant messaging, because different manufacturers used different technologies interoperability was always an issue, with the use of WAP it wasn't anymore. [15]
- October 2001 iPod First Generation is presented, with the capability of having a thousand CD-quality songs on your pocket represents a step into the future of portable music.[16]
- April 2003 iTunes music store is launched with capability of buying individual songs

cheap and without subscriptions fees[17], it sold over one million songs in the first week, *"In less than one week we've broken every record and become the largest online music company in the world,"*[18] thus creating the first complete solution to the digital music age.

- June 2007 the release of the iPhone, the first phone whose software was designed with the user in mind, with swiping, pinching and tapping a screen, gestures that made different outputs hence making it feel smoother to the user. And also letting a few selected Third party developers the capability of creating native apps for the iPhone.[19]
- March 2008 iPhone software Roadmap, among the plans is increased Third Party support giving full access to the SDK, make it more affordable and make the iPhone reach new countries and a future place where you can acquire apps from other developers that are not Apple related.[20]
- July 2008 the App store is launched with an initial of 500 applications available to the user, and giving the option of how to monetize the apps to the Third party developer between the choices are making the apps completely free, free with in-app purchases or paid, making it the start of a new era for both the smartphones and the users. [21]

The “Information Appliance” model 2008-2010

Although there is nothing new about what phones could do that wasn't able to be done by some other device, what really set it off was the ever increasing computational power coupled with its portability, and although mobile phones were already quite amazing in connecting people with simply the ability of relatively cheap calls, when the ability to connect to the internet occurred and the rise of both functionality and popularity of apps it was just a matter of time that it would be part of most people's lives, “In 1997 the nearly three-quarters of the world’s population who lived in low-income and lower-middle-income economies accounted for just 5% of the world’s population with Internet access By 2005, they accounted for just over 30%” and “In addition, by the end of 2008 more than half the world’s population is expected to have access to a mobile phone.”[22] were both some amazing statistic forecasting a great future.

- July 2008 in the first weekend after the launch of the App Store apple announces that it tops 10 million app store downloads, *“Developers have created a wide array of innovative mobile applications ranging from games to location-based social networking to medical applications to enterprise productivity tools”.* [21]
- September 2008, with more than 3000 applications available in the store, and 90% being priced at less than $10 and over 600 apps being free, its announced by apple to have reached over 100 million app store downloads in less than 60 days. [23]
- October 2008 Android market is launched and the HTC Dream phone, the first smartphone to use the Android mobile operating system, an open competitor to other major smartphone platforms of the time, such as Symbian operating system, BlackBerry OS, and iOS. [24]
- January 2009 “*there’s an app for that*”, an Apple campaign that made this a common catchphrase and later tries to trademark it. [25]
- April 2009, even though blackberry users were capable of downloading third party app for years, from manufacturers official sites and other stores like Handango[26] but since the success of the App store the desire of a centralized store grew on users of other smartphones and so the Blackberry world is launched, and becomes the third major distributor of mobile applications. [27]
- November 2009, WhatsApp launched and company established.[28]
- December 2009, Angry Birds is launched and becomes the number-one paid app on iTunes[29], also being one of the first apps that lead to the creation of many spin-offs, leading also to the creation of a market for merchandise featuring its characters, a

televised cartoon series and a feature film.

- January 2010, ZunZuneo Cuban app, a text-based social network (or microblogging) that allows Cubans to communicate with each other amid government-imposed internet blackout, it was later revealed in 2014 by the Associated Press that it was created by the US government in the hopes of triggering something like the "*Arab Spring*". ZunZuneo or "*cuban twitter*" was able to reach 40k subscribers and was later closed around 2012 without previous notice or any kind of explanation[30]
- October 2010, Windows Phone store (previously as windows Phone Marketplace) is launched at the same time as Windows Phone 7[31], and becomes the fourth major distributor of mobile applications, notably developers needed to have a membership and pay an annual fee of 99$.[32]
- December 2010, Android Geinimi Malware one of the most sophisticated malware that displays botnet-like capabilities, with the possibility of remotely sending commands from a server to a user's phone and control it, presumably the only way of the phone contracting this malware was through unofficial Chinese app stores, also it bears reminding that not all malware focusing on sensible data is contracted this way, and that attacks can be non-OS specific and also phishing attacks which can be contracted simply by clicking a link[31], [32]

The home screen era 2011 – 2013

If you started using smartphones after 2011 you would think that phones always had this type of layout, you would have a screen (and later on multiple screens) and your most favourite or most used apps showing right on the screen for fast access, but it wasn't always so, in fact many attribute part of the success on this relatively small change, like almost always, simplicity is key.

This era was a bit short, but it proved an important point, specifically the ability of other media leaking into the phone app, and that app's could leak into other areas, specifically the service provider ones that I will touch upon in the next chapter.

- January 2011, App voted as "word of the year", the runner up was cookie monster's "nom nom nom" phrase, widely used in chats, tweet[35]s and text-friendly syllable that connotes "yummy food", some critics said that the word app became omnipresent in that year.[35]
- March 2011, Amazon App store for android operating system is launched, the store included a "Free app of the Day"[36] feature, frequently a game, on the day of release the game was "Angry Birds Rio". The store also had a "Test Drive" feature which allowed users to try an application in the web browser for half an hour, this feature was decommissioned in 2015 because of the decline of its use.
- June 2011 Zynga games goes on a steady fall of users, reaching a loss of 63% of its total player base by 2014, in what people later attributed to the rise of the mobile game industry which in turn replaced the so called "Facebook gaming era", but also many critics from inside the company mentioned the management's lack of long-term vision of the company and the business, things like not being able to innovate on the art and gameplay of previous success. [37], [38]
- September 2011, App store bans satirical game "*Phone Story*", reigniting the debate about how Apple treats apps differently from music, books and films, "Apple has never hidden the fact that it has a stricter censorship policy for iOS apps than for other forms of entertainment. The debate about whether this is unfair is heating up once more this week, however, after it removed an iPhone game called Phone Story from its App Store."[39] and "*"an educational game about the dark side of your favorite smartphone," sets its targets directly on Apple, as you follow a new iPhone's release from mines in the Congo, through the oppressive Foxconn factories and to planned obsolescence in a gadget-obsessed West*".[40]
- December 2011, The app store and job creation, "*the app revolution has added more*

than 291,250 iOS jobs to the U.S. economy since the iPhone introduction in 2007".[41]

- March 2012, Android Market is renamed to Google Play store and with it a sweeping revamp of app, book, music, and video stores, even though all of this was sort of under the same umbrella (Android Market) it would now be officially a 1 place-stop for all buyable and free content for the phone. [42]
- April 2012, Facebook acquires Instagram, with a price tag of 1 billion dollars [43], the highest ever paid for an app company to that date, some say the tops reasons for the buy is both to acquire more data, which in turn would translate to better mobile ads and because Facebook was losing its younger generation of users. [44]
- May 2012, angry birds reaches 1 billion cumulative downloads pricing at somewhere around 99 cents a game, further cementing the notion that the future is on mobile gaming. [45]
- November 2012, Candy Crush is released on iOS, and becomes the most downloaded iOS app of 2013 finishing above Facebook, angry birds, and YouTube, while Minecraft was topping the iPad competition, worth mentioning all 10 of the top 10 apps for iPad were games while for the iPhone it was 9 out of 10, and the dating service match.com being the only non-game on top 10. [46]
- May 2013, Apple store's 50B downloads vs google Play's 48B download, even though Apple had a head start we start to see google closing the gap between them, even though "*...doesn't take into account paid vs. free apps, or how much revenue each makes from ads and other sources. But as you can see from the graph, it marks one area at least where Google used to trail considerably but is now catching up.*"[47]
- May 2013, the rise and fall of Flappy Birds, it was downloaded over 50M times before being pulled down by its creator, when asked why he removed it he said: "*Flappy Bird was designed to play in a few minutes when you are relaxed. But it happened to become an addictive product. I think it has become a problem. To solve that problem, it is best to remove Flappy Bird. It is gone forever.*"[48]
- June 2013, Universal's "Despicable Me: Minion Rush" app is launched, it makes the top 10 apps and in less than 3 months since release reaches 100 million downloads, just like in the past with conventional gaming, we see the movie industry seeping into this relatively new industry and displaying amazing numbers.[49]
- October 2013, 1 million apps in the App store, 1 billion songs played on iTunes and a total of 13 billion being paid out to developers over the years, signalling a multibillion-dollar industry.[50]
- November 2013, Snapchat rejects 3 billion bid in cash from Facebook, seen as a way of stopping the bleeding that is the declining engagement among the teenage users "The

kids who are losing their passion for Facebook are pretty much exactly the same ones who are flocking to Snapchat by the millions. While it has never revealed its user count, my colleague J.J. Colao says it likely has around 26 million users in the U.S., if the results of a recent Pew survey are anything to go by. More impressive, 26% of adults between the ages of 18 and 29 say they use it".[51]

App as a service layer 2014

Software as a Service (SaaS) is something used by many companies and individuals, but it's not something new, it was a tried and failed business model well over a decade ago, so what changed to make it do a 180 degree turn in revenue or profitability? What changed in cloud computing to make it work now?

Now we know it as the "dot com boom" or as it was called back then "Application Service Provider" (ASP), but the way of doing it was the same, users pay for a subscription to an application accessed via website.

Even though most companies failed under the ASP model a few survived and evolved like, Great Plains Accounting system (it became Microsoft Dynamics)[52] and salesforce[53]. The premises were almost the same back then as it is now: "*low cost, easy deployment, painless upgrades. So was the deployment model: remote full-featured applications over the Internet.*"[54] the only difference being SaaS keep up their end of the bargain, ASP didn't. Well that and the fact that SaaS these days can better take advantage of Virtualization and cloud-based scalability, with virtual machines, it's easy to put up new instances of for each customer, granted virtualization dates back to 1960s mainframes, back in the ASP days vendors had to setup extra physical servers to meet user demand, now imagine somewhere in the world the client needs their staff to have their own logins, requiring more instances of an application to be made, with time zones being an inescapable reality and the need to back then physically setup extra servers when the customers notice came in...well I think you can imagine how bad that would go, and to be honest this little detail is nothing compared with the reality that most ASP companies made custom development for their apps rather than spawned images, "*The economy of scale is what killed a lot of hosting providers back in the ASP days and ran them out of business,*"[54] and *"They were just doing an implementation for every customer, as opposed to a single implementation that can now be used by multiple customers – personalized and managed. The people who use the application run and use it differently, but the implementation is pretty much the same for all customers.*"[54]

And let's not forget the final nail in the coffin of the dot com era, internet connection and speed that was reachable back then, "*In 2000, the Federal Communications Commission (FCC) reported that there were 2.8 million "high-speed" Internet lines. But they defined "high-speed" as faster than 200 kilobits per second. Not megabits, kilobits. Today, the average U.S. Internet speed is 21 megabits per second*."[54] talk about a great idea that came to soon.

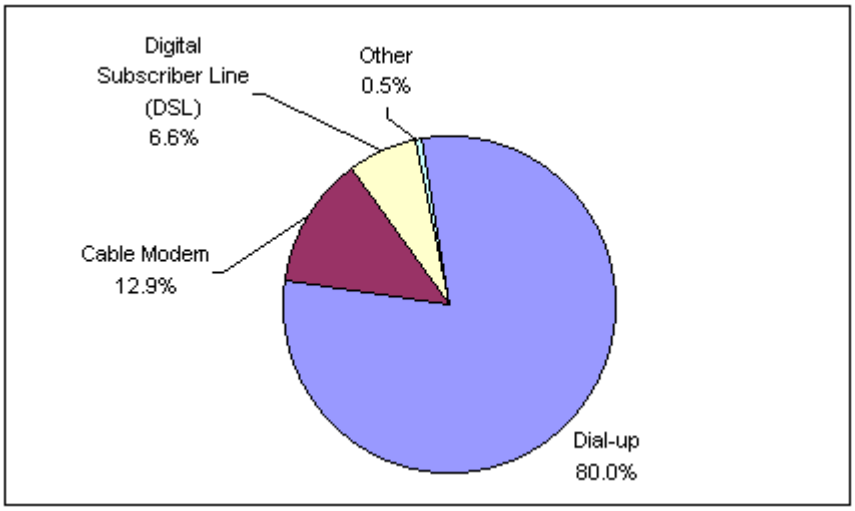

Figure 2 - Home internet Connection Type, 2001 as a percentage of individuals Using the Internet at Home[54]

- January 2014, prediction of 1.75 billion smartphones for 2014, while the 1 billion marker was hit back in 2012, the amount of users using normal mobile phones is said to be 4.55 billion by the end of 2014 worldwide, worth mentioning that the new markets rising in the developing regions of Asia-Pacific, Middle East and Africa can further increase the predicted number.[55]
- January 2014, Apple announces that sales on the App Store on 2013 was 10 billion dollars with December alone amassing more than 1 billion dollars making it the best year of the App Store ever.[56]
- February 2014, another multibillion-dollar deal is made on the app business world, Facebook acquires WhatsApp for 19 billion in cash and stocks, some say the deal is overpriced since the annual revenue of WhatsApp is around 20 million annually, while others say Facebook was looking at other ways of maximizing this revenue. [57], [58]
- March 2014, Android grows to 87% of the share of the global smartphone market but also has 97% of global mobile malware but almost all of it comes from unregulated third party in Asia and middle east, and it's said that only 0.1% of all malware comes from Play Store, and furthermore the shelf-life of the ones found on the official store tends to be extremely short[59], the top reasons for android to have such a high number of malware are the repackaging and trojanizing of popular apps and the customizable nature of the OS, which helped make it so popular, also prevents consistent security measures.[60]
- March 2014, Facebook releases new messaging app Messenger (0.5 billion downloads in six month) that is now mandatory for sending messages in Facebook app, but the traditional "we know best" attitude doesn't go well with many people, the now

inevitability of having to download a second app to do something you previously were able to do with only one app, not to mention low end smartphones will struggle with this new “requirement”, furthermore Facebook disregards the customers previous decision of privacy, by automatically attaching the current location with each message sent. [61]

- May 2014, Gmail reaches 1B downloads, almost two years after becoming the most widely used mail service on the planet, and it’s said to have had over 425 million active monthly users back in 2012.[62]
- June 2014, App Store reaches 75 billion downloads celebrating its sixth anniversary and calculated to generate 15 billion for the 9 million registered developers, App Store is so successful that it’s hard to fathom now that the iPhone was first launched without it.[63]
- June 2014, Wut Wut is launched, it’s messenger-type app with a little twist of anonymity in conjunction with the idea that things shouldn’t last forever “*The messages are also not saved. You can only see as many notifications as your phone’s screen can fit without scrolling down. The experience is more like seeing passing chatter from friends that disappears quickly, rather than diving into a never-ending feed of content.*”[64]
- August 2014, Celebrity iCloud hack (also known as The Fappening), 500 private celebrity pictures hacked, the hack was successful thanks to a brute force attack via Find My iPhone, lots of debate goes on trying to figure out who the blame goes to, Apple who was not able to secure its customers data or maybe of being somewhat negligent if they knew of the issue and did nothing, or if the blame was to fall solely on the hacker that stole and publicly shared the photos. [65]
- September 2014, Game apps are the most popular, 20.3% are games, 10.36% are educational and 1.9% social networking.[66]
- October 2014, The BBC launches a service that provides audio, text messages alert and images through WhatsApp in West Africa to help people get the latest public health information to combat the spread of Ebola, to subscribe to the information users just had to send a text message with ‘JOIN’ to a certain number and ‘STOP’ when ready to unsubscribe, also at first the service was only available in English and in French languages. [67]

Phone App business

In the summer of 2018 the mobile app ecosystem turns 10 years old, with millions of developers working for billions of smartphone users it's easily one of the biggest industries on the planet, a business that is mostly run by two companies, Apple and Google, and to a lesser degree, Amazon and Microsoft. The App Store functionality for all these big names was certainly a great contributor to the success and reach it has on everyone's lives, but as almost everything, it has its pros and cons.

As far as the pros goes, even if it only had this one pro going on for itself it was actually so big it could tackle dozens of cons just by himself, the app store of each company is responsible in having a much bigger reach on customers for apps, it gives the customer a sense of trust by saying that the app most likely won't have any malware in it since for it to be on the official store in the first place it goes through a bunch of "certificate" loops, and also the security of the money transactions taking place, it is something that is always of major importance and is being looked at and improved upon in the official store, since only having one or two transactions going poorly can dramatically sever the stores reputation.

Since the pro mentioned above is so big almost all cons are somewhat related to it, the major con specially to new apps being developed, is the discoverability issue that is inherently associated to "big stores", the phenomenon that creates this problem in the first place is known as "Positive Feedback Loop"[68] and is a concept very well known in Science and Engineering, but also included in biology, chemistry and cybernetics. Even though it has positive in its name it does not always produces a positive outcome in general, it's a process in which the effects of a small disturbance on a system include an increase in the magnitude of the perturbation. Simply put, **A** produces more of **B** which in turn produces more of **A**, this might be good for the specific app able to push through the rest in the category, in which it will have a very sizeable number of downloads which in turn makes it into the popular downloads app, which in turn increases the number of downloads, but for all the others app it's hard to push through, even if the new app is better than the old ones in existence. And lastly and in general terms lack of software trial support.[69]

Then there is the model used by each company for the apps to be displayed on the store, for the Apple App Store, sometime in 2015 they changed the way they promote new and popular apps on the store, it implies Apple's control over apps quality and their compliance to certain standards, this also marked the shift from algorithmically generated list to editorially curated content, "*the "Games" section, for example, no longer has "New," "What's Hot," or "All iPhone (Free & Paid)" categories, which have been replaced with editor curated lists such as "Best New Games" and "More Games You Might Like," the latter based on a customer's purchase*

history."[70] This seems like the best possible solution to overcome one of the cons of the App Stores, but it has been claimed that some app developers have seen between 30% to 90% fewer organic app downloads since the changes were made to this model, although it seems the change is positive for the majority of the developers, which just goes to show how hard it might be to overcome the “Positive feedback loop”.

For Google’s Play Store, the app publication process implies much less strict guidelines for app developers to comply. They are able to reach such analysis speed thanks to their automatic viruses and malware scans, image analysis systems for detecting apps that include sexual content as well as those who infringe on other applications copyright[71]. Only then comes the human to make its final approval. This isn’t perfect of course and creates the well-known problem that sometimes some apps with malware end up on the Play Store, which is something that google is always on it.

App Download Statistics

"On the following graph we see almost 50 billion app downloads jump this year, 197 billion in 2017 versus 149 billion in 2016. According to a projection, by 2021 the total app downloads number will jump to a stunning 352 billion."[69]

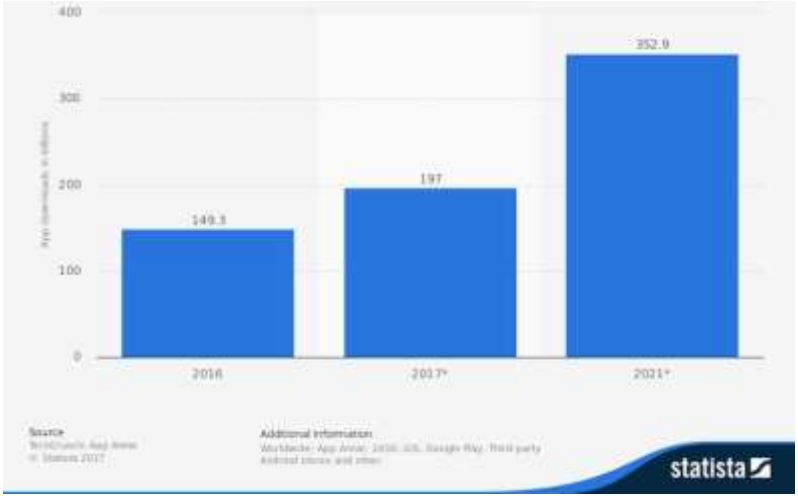

Figure 3 - Number of mobile app downloads worldwide, 2016, 2017 and 2021[69]

Sometime between 2013 and 2014 Google surpassed Apples app download number, and has continued to dominate the field of number of downloads, this was attributed to the fact that only iPhones and iPads devices are the ones able to download from the Apple's App Store, while many 3rd Party Smartphone company creators can use Google's Android OS, and in turn use their app store for acquiring more apps.

Figure 4 - Cumulative Number of iOS and Android (Google Play only) App Downloads, 2014-2016[69]

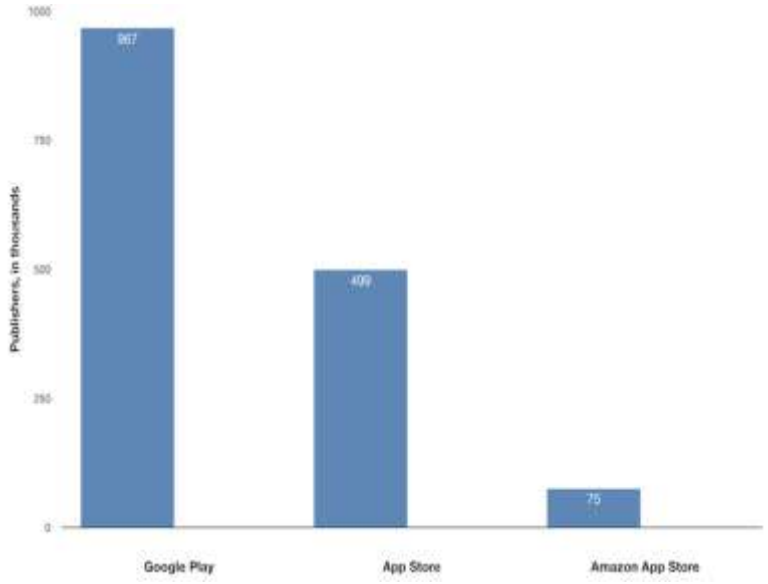

Figure 5 - Number of App Publishers, by App Store, in thousands[69]

Apple's App Store Statistics

Even though it started with a mere 500 apps on its store, the enthusiasm of a mobile app software community propelled the industry like no one could predict. Although there are talks about a slowdown in the store growth which probably correlates to its maturity and saturation, which lead Apple to a new mindset, "... iOS App Store growth is driven not by quantity but quality, each new smartphone and tablet hardware update brings new possibilities for app developers to innovate.[69]

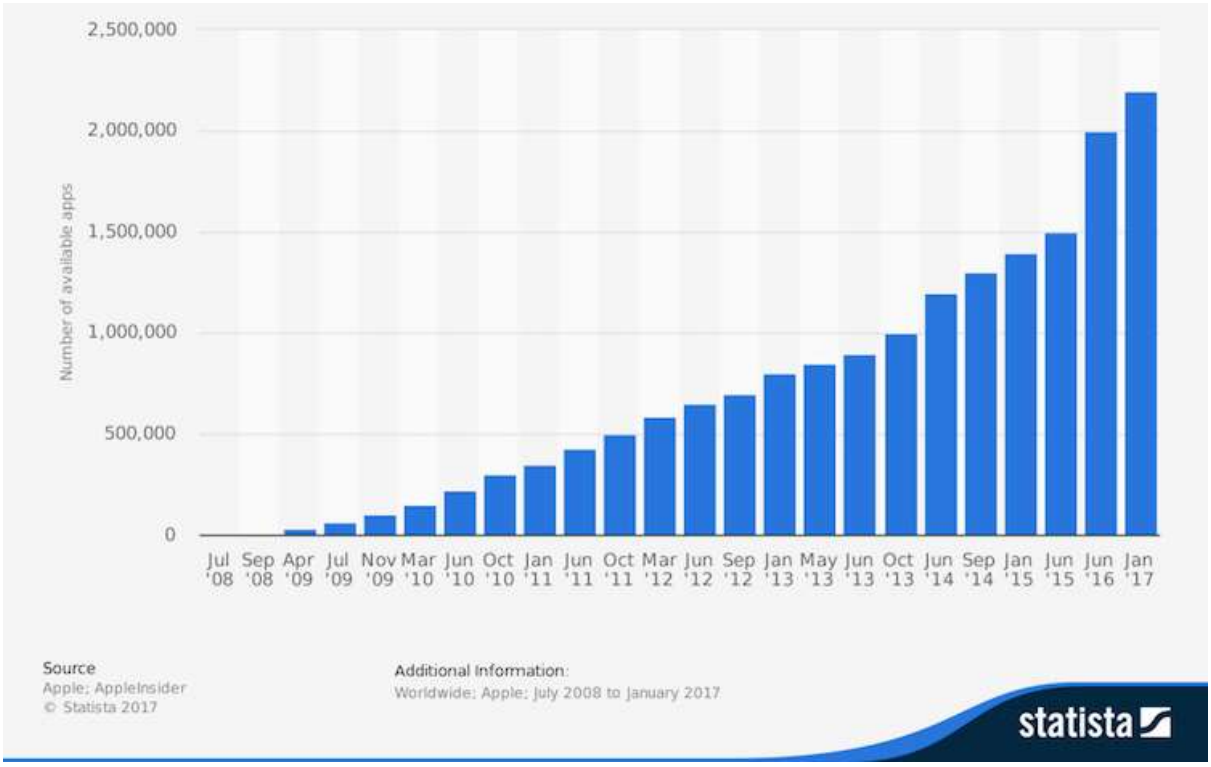

Figure 6 - Number of available apps in the Apple App Store, 2008-2017[69]

Google's Play Store Statistics

With the launch of then known as "The Android Market", launched 3 months after the App Store, it took 4 years for the Play Store to match its number of apps to its rival, the App Store, and has been on top since then, it currently boast a growth of over 1300 apps a day, the open source license given by google sure helped it be the kind that is today, at least on number of apps available that is.[69]

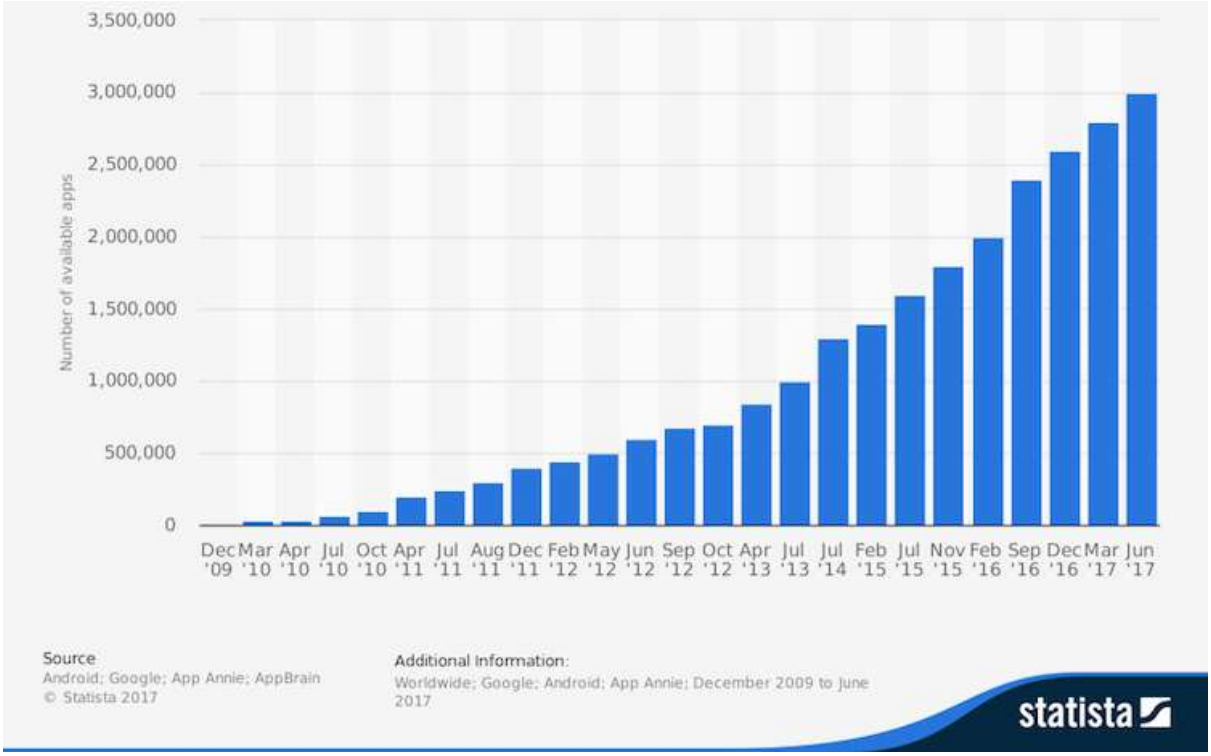

Figure 7 - Number of available applications in the Google Play Store, 2009-2017[69]

Agile Software Development

Developing software, is just writing a list of things to be done, but for some reason when developing software, it's not as straight-through as it seems, **especially** when doing it as a team. Just imagine how it would go to try and write a book like game of thrones be written by multiple people, even if all the writers came from the same university doing the same courses with the same professors and on the same time interval, it would still probably not go as good as we would like. Not only that imagine that each writer must talk to a different number of publishers to get a sense of what is needed to best publish a book, and get illustrators to see what a good cover would be like… I think you get the picture of where I´m going with this.

Add to this the fact that prior to 2000 the way costumer and developers made business in the IT sector was pretty much a onetime meeting (waterfall model[72]), and with enormous time lag between business requirements and the delivery of those needs it invariably led to the cancelling of many projects.

"*In 2000, a group of seventeen "thought leaders," including Jon Kern, Kent Beck, Ward Cunningham, Arie van Bennekum, and Alistair Cockburn, met first at a resort in Oregon and later, in 2001, at The Lodge at Snowbird ski resort in Utah. It was at the second meeting where the Agile Manifesto and the Twelve Principles were formally written.*"[73]

Agile Software Development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer/end user. It's a strong advocate of adaptive planning, evolutionary development, early delivery, continual improvement, and it encourages rapid and flexible response to change.

This manifesto lead to the four foundational values which I will tackle a little bit more in depth and give a bit of my opinion on them in the chapter "Values and concepts of Agile Manifesto", and to twelve supporting principles to those values which in my view are a bit more straight through and don't require much of an explanation or deep diving required, so I shall only list them.

The twelve principles of agile development include:

1 - Customer satisfaction through early and continuous software delivery – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.

2 - Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.

3 - Frequent delivery of working software – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.

4 - Collaboration between the business stakeholders and developers throughout the project – Better decisions are made when the business and technical team are aligned.

5 - Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.

6 - Enable face-to-face interactions – Communication is more successful when development teams are co-located.

7 - Working software is the primary measure of progress – Delivering functional software to the customer is the ultimate factor that measures progress.

8 - Agile processes to support a consistent development pace – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.

9 - Attention to technical detail and design enhances agility – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.

10 - Simplicity – Develop just enough to get the job done for right now.

11 - Self-organizing teams encourage great architectures, requirements, and designs – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.

12 - Regular reflections on how to become more effective – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

Values and Concepts of Agile Manifesto

On the Agile Manifesto that sparked all of this, they noticed that some concepts were more valuable than others, concepts that at first ran a little bit against the current narrative that was rigid and objective, to something a bit more fluid and subjective, here are a few:

"**Individuals and interactions** over processes and tools"[73]

When we allow processes and tools to manage the product development and everything related to it, we indirectly make people and the way they approach the work to conform to these processes and tools, and of course conformity makes the accommodation of new ideas, new requirements, and new thinking, quite hard. Agile approaches in contrast, value people over the processes and tools, this emphasis on individuals and teams makes their focus, energy, innovation and problem-solving skills to come up with new and usually better ideas for the product. The processes and tools are still used in agile project managements of course, but they are intentionally streamlined and support product creation. When people are the focus of the software development, the result is usually a leap in productivity. An agile environment is human-centric and participatory and can be readily adapted to new ideas and innovations.[74]

The focus on people over processes come with these kinds of benefits:

- Communication is clear, effective, quick and efficient.
- Teamwork becomes strong as people work together.
- Development teams can self-organize, have more chances to innovate, and gain deeper job satisfaction.
- Development teams can customize processes as necessary and take personal ownership of the project.

Alas, development team members must have the capacity to be involved, responsible and innovate and might need to let go of ego to work well within the team, but even so, these disadvantages are minor compared to those that follow the opposite value:[74]

- People may over-rely on processes instead of finding the best ways to create good products.
- One process doesn't fit all teams nor all projects — different people have different work styles.
- Communication can be ambiguous and time-consuming.

Next, we move onto the second value of the manifesto, and this one might look a little bit like a counter-intuitive value at first, but again like mentioned above, just because we value the first more than the later, doesn't mean the latter has no value.

"Working Software over comprehensive documentation"[73]

For starters we need to acknowledge that we live in a different business climate now than we used to 20 or even 10 years ago, the needs for a project change very rapidly and that obviously implies the team needs to adapt to these changing requirements.

With that being said, writing comprehensive documentation for a system, has many drawbacks, for instance, if your team spends time documenting the requirements, analysis, design and test cases there is a high chance that what was documented will be outdated by the time is done, which obviously implies that more time will be needed to be spent on correcting the documentation, and if anyone has done something like this before should know that it's pretty hard to correct things than to write them right from the get go.[75]

Another issue is documenting things that might not be needed in the end, naturally, teams want to account for all the features that they think might be needed, they will waste effort and time on documenting these features that might not even make the light of the day on release, so this just leads to more wasted time and frustration on a business that is becoming so dynamic, which in turn results in lost opportunity cost because you delay earning a return on your investment.

The best documentation that provides value is tests for the system. This is executable documentation that remains accountable to the realities of the working system. There are different types of tests and each one focus on a different aspect of the system, those are the Unit test, functional tests and integration tests, and they all result in less defects and more confidence in the system, keep in mind these are meant for the developers.[76]

Unit tests validate the behaviour of individual routines within the system, by testing individual units of the source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.[77]

Functional tests validate how the different routines in the system will interact with each other, Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered. Functional testing usually describes *what* the system does.[78]

Integration tests (also called integrating and testing) verify that all of the parts of the system work together as expected, Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and then analyse the output.[79]

Together all these tests provide accurate, living documentation of the system and this

documentation is updated as the system evolves.

Just like the first value was not directly connected to code itself, this one follows something similar, but migrates from inside the “working lab” and the developers to the customer or the internal customer representative like a product manager.

“**Customer Collaboration** over contract negotiation”[73]

When it comes to today’s marketplace, it is more important to understand the customer’s needs than what was negotiated in the contract. (or like I jokingly like to say, the customer knows what they want not what they need)

Historically, a project management approach usually involved customers at three key points:[80]

- **Project start:** When the customer and the project manager — or another project team representative — negotiate contract details.
- **Any time scope changes during the project:** When the customer and the project manager negotiate changes to the contract.
- **End of a project:** When the project team delivers a completed product to the customer. If the product doesn’t meet customer expectations, the project manager and the customer negotiate additional changes to the contract.

This historical focus on negotiation discourages potentially valuable customer input and can even create an adversarial relationship between customers and project teams, and even though competition in almost all businesses and markets is good, it’s not really that great when customer and developer do it.

To develop software that meets customer needs requires regular communication along the entire development process.

Customer collaboration is especially key to determining, developing and deploying a successful experience design for your customers and end-users.[81]

Agile Processes handle project and product changes in beneficial ways, for example:

- Agile projects create an opportunity for increased customer satisfaction and return on investment by handling change effectively.
- Changes can be incorporated into subsequent iterations routinely and smoothly.

And last but not least, we arrive at our final value, this one in my belief is harder to uphold than the rest of them, because this is something that we as a species have been fighting for ages, the fight against change, obviously not all change is bad, which in turn makes it true the statement that not all change is good, so let's dig deeper into this.

"Responding to Change over following a plan"[73]

Change has been part of human history since...well history itself, but how to be able to make things accommodate with stuff that does not exist now or maybe not be able to predict?

Traditional project management approaches have always attempted to wrestle the change monster, this is an after effect of project teams blindly following the plan and missing opportunities to create more valuable products, to avoid this many companies in the past used to try to implement many things that were not even part of the initial requirements to avoid the pain of adding changes later.[82]

The new Agile software development processes accomplish this through different levels of frequent planning and re-evaluation, they start with a "release planning", with this, teams agree on features to include in a release, the features themselves remain very large and abstract, almost like a painting made with very large strokes. Once these large strokes are grouped up into meaningful releases, we proceed to prioritize each release and then break them down into smaller time periods known as iterations.

At the beginning of each iteration the team agrees on which features they can complete, then they are broken down into stories, which have levels of prioritization in a list, by working like this the team has the ability to adjust every week or two, thus making the incorporation of new features much easier in the next iteration, since no amount of thinking can replace hands-on experience with the real system, this way of working gives great benefits.

This way as the team works on iterations, the stories are pushed from one stage to the next, the stages might vary from company to company but generally include development, testing and deployment. Once an iteration is completed, we can take some time to reflect on what was done, it's almost like a retrospective meeting, which it's discussed what went well and what not, and what other changes should be made going on into the future, this meeting helps learn from mistakes and make an ongoing commitment to improvement.

Also, with this mindset came to be a new development process, the Lead Software Development, it's a process which focuses on limiting the work in progress based on the constraints of the team. This is done to prevent any one stage in the process from completing more work that the other stages can process, in a sense it limits the work in progress to the slowest stage in the process.

This way, teams can identify bottlenecks in the process, which can be relate to bureaucracy,

time, resources, distance or whatever. Making it easier to remove the bottlenecks and make it function at the optimum level.

Lean software development processes do not use fixed time periods like iterations. Instead it continues to produce features at the rate that is capable of doing it and when the team determines that is appropriate, thanks to this they can use value stream mappings to determine what are the most important parts of a feature to the users, allowing to get even quicker feedback and validating what is being done as building a desirable product.[83]

Another reason why we went with agile development over something like waterfall which is the same regarding steps in the iteration process but unlike waterfall, agile is less rigorous when moving through all the steps which is something ill go over with in a later chapter demonstrating this very point.

Because of these reasons, and since we were committed in delivering constant and almost tailor-made experiences to the users of the app, we adhered to these values and over the next pages I shall go over many examples that prove these were always part when developing the final product.

3- Technical Background

So before going any further, we should go over the main tools and frameworks that I used in my internship, worth mentioning I used many other tools but to some extent I already worked with them or they were pretty much a staple among all IT companies, things like a chat tool (we used slack[84]), or a version control tool (GitHub[85]), so for this topic I shall only focus on the two pillar my work rested upon, the Framework that my code was being run over, and the Database that was used.

React Native

React Native (RN) is the baby brother of React, React is an open source JavaScript library designed to build user interfaces. The need for the existence of React Native came from the popularization of app's, and the exhausting need to write more than one code/project to be able to run on different OS systems.

In the past developers of an App had to create an android studio project to be able for Android phone users to use their app, and another Xcode project to run the very same app for iPhone users, and even a third time to run it for windows phone users, this can easily be identified as a problem for developers that want to build apps quickly to reach the maximum amount of users.

Since Android and iOS have very different codebases, business and start-ups struggled to be able to hire developers for both, with RN this problem is gone, and only need one developer to write across different mobile operating systems.

The promise of a framework that will be able to "Write once, run anywhere" isn't new, and is known as mobile web apps, HTML5 apps or hybrid app. It's basically a responsive webpage loaded in a WebView. Unfortunately, when dealing with different OS and these frameworks we tend to lose the native look and feel of the mobile platforms, since the performance of the application decreases, transitions, animations and clicks all take a toll and become slower making users not wanting to use the applications anymore.

Basically React is used to create web application, while RN is used to build Mobile app's both use the Reactive Core part of the architecture, but then they fork between React and RN, RN is one of the best options because it uses Webkit's Javascript VM JavaScriptCore and platform specific UI components for each platform to not only do the job is supposed to do, but also deal with the drawbacks of most of the frameworks that promised and didn't really delivered. RN architecture is comprised of 3 modules:[86]

- **The core**: Includes the business logic and the state of the application. This component can be implemented using the Redux framework. Code reuse is done via this module.
- **The mobile App**: Implemented as a standard React Native application, using containers and components.
- **The web application**: Implemented as a standard React application, using containers and components.

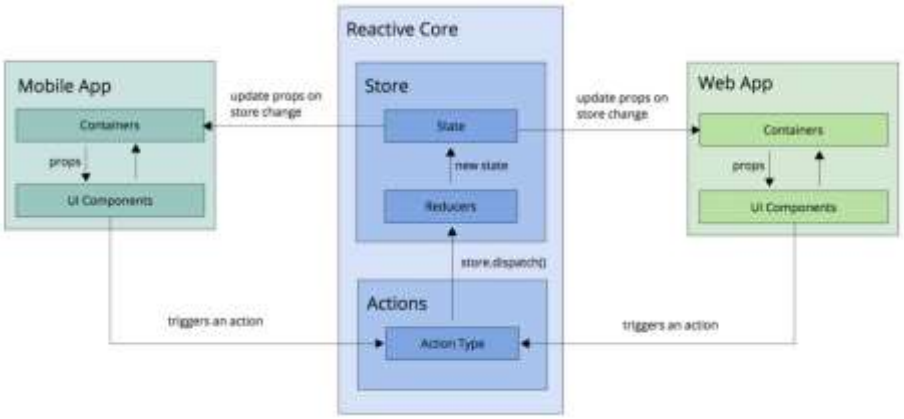

Figure 8 - Diagram of the Architecture of React[86]

So, let's analyse both pros and cons of using RN:[87]

Pros:

- **The community**, it's growing quickly and offers a great network of experienced developers either through the updated documents on their official page and neutral forums like stack overflow.
- **Faster Development**, it's obvious that since you only need to write the code once and it will run on multiple OS it will take less time to build a final project, but it's more than that, the fact that you don't need to brainstorm on how to build it for one environment or the other, the time spent exchanging crucial information between different developers and so on, it's just time that can be better spent on more crucial matters in such a cutthroat business.
- **Closer Teams,** this one is a sub-pro from the faster development but still, with React Native bringing both iOS and Android developers together, you'll most likely be working in closer teams. This should make working together and making decisions a lot easier and faster.

Cons:

- **still improving,** it doesn't have all the modules that the native development environments have, but they are still working on it adding new libraries and functionalities to speed up the development process, but even when no official module to a problem exist there are quite a

few solutions there made by the community, you'll just have to spend time looking for it.

- **Still technical,** RN eases the development of many aspect of coding with its pre-packaged elements, but you still need a good developer to take care of some technical issues. Things like incorporating the camera accessibility, or even push notifications and some other sophisticated data handling.
- **will it last?** there is a small concern since it was developed by Facebook, that it could kill the project at any time, making you waste time on something without a future.

Firebase

Building app's in this day and age is all good and dandy, with dozen or even hundreds of guides to help you get started for almost everything imaginable, but dealing with the back-end of things was always a sort of different beast altogether, you had to deal with the authentication, you had to design the database, how to host the page, it was an absurd amount of things that had to be done, just to be able to show your idea for the app or webpage.

Enter Firebase, the back end as a service (BaaS) it deals with all of the aforementioned problems with an incredible ease and it even has plenty more functionalities, one of its most known abilities is the Realtime Database, which stores and synchronizes data between users and devices in real time on the cloud, and the best part is that the data persist even when the device or app goes offline. It has an extensive number of guides to help you get started with the service, it works with Android, iPhones and for regular web pages, it uses Java and JavaScript as its main language, but some of the functionalities of firebase like cloud functions and Authentication can also operate in C++ and Unity.

It uses the NoSQL types of database contrary to the conventional SQL ones, basically it means it is a non-relational Database, this kind of systems store and manage data in ways that allow for high operational speed and great flexibility on the part of the developers, also unlike SQL systems many of the NoSQL systems can be scaled horizontally across hundreds of servers. Of course, this gain in scalability and speed comes at the cost of data consistency for the reliable transactions which is a must for certain types of apps or web pages like for instance banking.[88]

Syntax-wise SQL is a lightweight declarative language. It's deceptively powerful, and has become an international standard, although most systems implement subtly different syntaxes. NoSQL databases use JavaScripty-looking queries with JSON-like arguments but can also use XML-like files! Basic operations are simple, but nested JSON can become increasingly convoluted for more complex queries.[89]

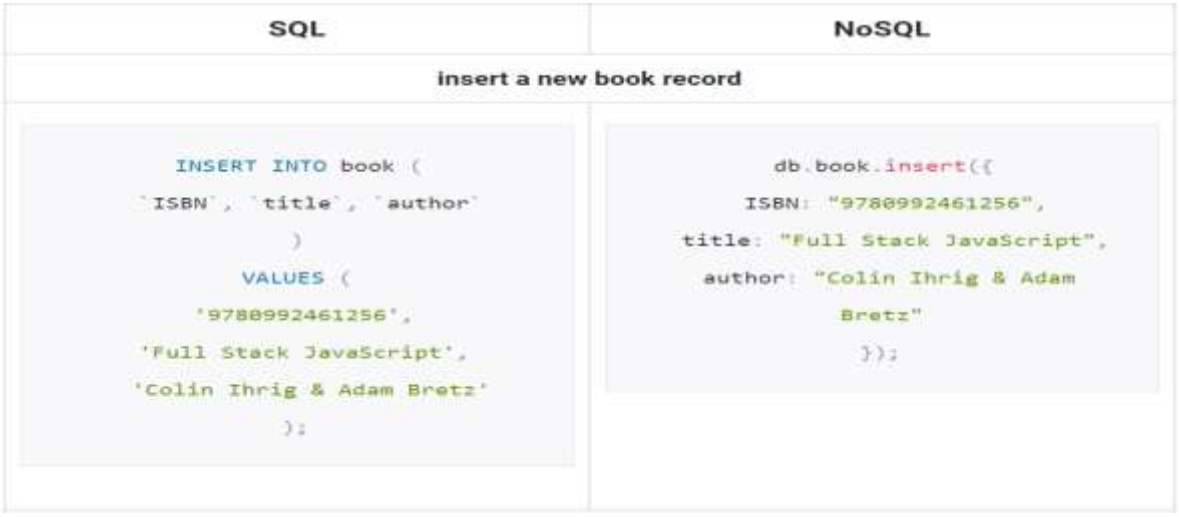

Figure 9 - Differences on inserting a new item, between SQL and NoSQL[89]

All in all it would boil down to these reason when to use SQL or NoSQL:[89]

Projects where SQL is ideal	**Projects where NoSQL is ideal**
logical related discrete data requirements which can be identified up-front.	unrelated, indeterminate or evolving data requirements
data integrity is essential	simpler or looser project objectives, able to start coding immediately
standards-based proven technology with good developer experience and support.	speed and scalability are imperative.

Regarding Firebase this is a good summary of its pros and cons:[8]

Pros:

- A wide range of authentication, ranging from the standard of email & password, to all the new ones like GitHub, Google or even Facebook.
- Realtime data.
- Plenty of ready-made API for most of its functionalities.
- Built in security at the data node level.
- File storage backed by Google Cloud Storage.
- Static file hosting
- Capability to build highly scalable applications by treating data as streams.

Cons:

- Limited Query abilities due to Firebase data stream model.
- Traditional relational data models are not applicable to NoSQL.
- No on-Premise installation.

4- Role and Context in the company

Before continuing it is important to introduce a small summary of how I got into the place and position I was in the company I interned with.

After applying to many company across a myriad of countries in Europe through quite a couple of websites for that same purpose, I finally got into the part of doing the interviews, the first ones were mostly to get to know each other, talk about the subjects that I learned about in my academic career, to see what was the frameworks and languages used in the respective companies and what would presumably be my role in the company for the internship.

The lucky pick was a relatively new company which created a mobile phone app named Equilab, it was close to achieving the 100k downloads on the Apple's App Store and Google's Play Store combined (number achieved close to 2 months in my internship).

The next skype calls and emails exchanged was about how everything was going to be solved, the back and forth of papers regarding the Erasmus+ institution, the sending Institution (Universidade da Madeira) and the receiving institution (Equilab AB) and to also go over the "Intellectual Property Rights" and the Secrecy demanded for the Internship.

When all the "red tape" was done we finally moved to the role I was going to play inside the company, I was given the chance to choose between developing for iPhone or Android which I then choose to work with android since I had a previously worked with it for a course in my Academic career, although when I arrived there it was decided that I was actually going to start working on the new framework that was going to start to be used in the company React Native and with the BaaS Firebase, I was going to be working on the new feature which basically was going to work like a goals/sub goals for the users of the App so they could write their desired goals and keep track of which ones were completed, we were also to display a badge-like system of rewards that would be tied down to their completed goals, all of this was to be done with the user in mind, with constant testing not only with other developers but also with regular users of the app, some of the techniques used were the Usability testing, and also a form of A/B testing with not only users of the app but also with fellow developers that were not in the know-how of the new feature being developed, so to ensure the highest quality software was being delivered to the users.

From the get-go I was instructed to read the Material Design[90] that was developed in 2014 by google, Material Design makes more liberal use of grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows. The reason for the creation of this design is "unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch."[91]
With this we have all the references and background needed to analyse my work within Equilab.

5- Development Process

Agile Methodologies & Human Centered Design

So, let's see how we put to practice what was being preached in not only our academia classes but also on many companies that say they are agile but then fail to deliver on their promises.

As previously stated the business market has a tendency of being pretty dynamic in contrast to stale-like which leads to companies with fast reaction to all kind of changes be more successful and stay relevant in the market, if you factor in the IT business then you can pretty much multiply this reality tenfold, because of this we at Equilab were following mainly two types of meeting in the SCRUM/Agile methodologies and those were the weekly Sprint Planning meeting and daily Stand Up meeting.

The weekly Sprint Planning usually happened on Mondays and were used both to analyse how the last weeks plans went and to set up the plans for the week, all the developers were on this meeting and sometimes even the marketing members were part of this, these meetings would last between 1 to 2 hours long, we would go over the needs of the user, the stakeholders, for each task on the backlog. For tracking this information, we first used Trello[92], Trello is a collaboration tool that organizes projects in boards, this way with a quick lance we can see who is working on what and also see how everything is and on what part of the plan it's going. In our "whiteboard" we had four main columns to which all task would either belong to one of them at any single time, those were the "Current Sprint" column that was updated weekly with the plans for the coming week, the "In progress" column to add the particular objectives being worked on this column could either have still a few left objectives from previous sprint or move the newly added sprint task to this column when a developer started working on them. The "completed" column is as the name suggest the task that were completed and to be analysed to see if they were to the standards the company had set or if it had to be reworked on and if so move it back again into the "In progress" column, and lastly the "planned" column this one was mostly to report bugs and state possible fixes to a few inconsistencies found, and to be analysed on the next Sprint and decide how to move forward with it.

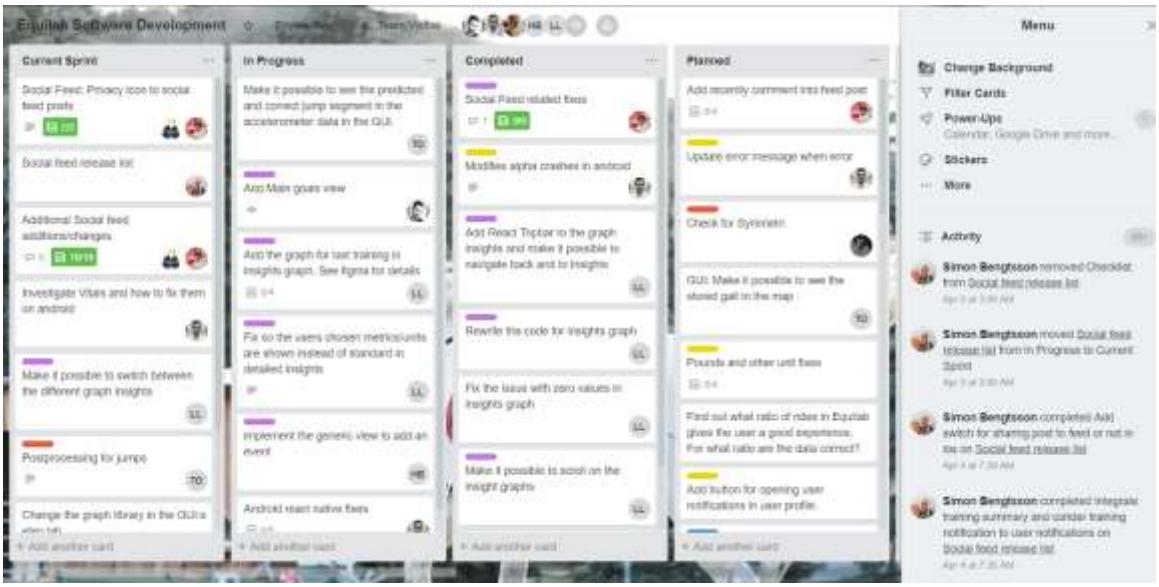

Figure 10 - Equilab Board in April at Trello

We later moved on to another similar tool named JIRA[93], the key differences between both tools is that JIRA is more focused for software teams this lead to a incorporated reporting functionality, issues management and time tracking instead of creating a separate column to deal with bugs and issues found, and finally JIRA had a higher number of 3rd party integration which made us finally transitioning to this new tool.[94]

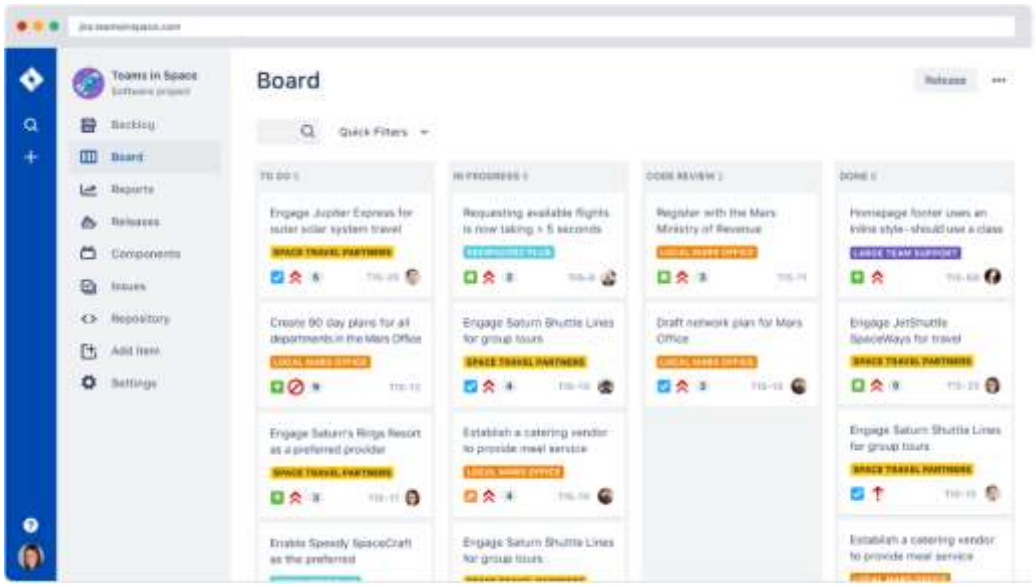

Figure 11 - What a regular Jira Board looks like[93]

The main reason for this change was because at the time, the development team was the one using the most this tool, and it would benefit us greatly to use the new incorporated reporting functionality, specially since we had developers abroad working for us.

Now on to the Stand-up meetings, these usually took less than 30 minutes, and would happen early on the morning and they usually go like this, it would start on a random person in the company and he/she would say how the previous day had gone, what was achieved, what problems were faced and finally what was the plan for the current day. You might think something so simple would have zero value to the company but you would be wrong, you see this meeting was really great for enabling strong visibility and transparency among all teams but it also made easier for developers to share ideas on how things could be better done, to be more scalable or to simply state a few warnings when developing a specific task for a certain framework, as for anyone that has worked with any framework they would know that each one has its quirks and sometimes what works on one doesn't work for another. These meetings in time proved to be invaluable to developing great code, both easy to read and scalable.[95]

Last but not least is the UX design and collaboration tool Figma[96], this is a web-based tool with real-time collaboration giving the ability to share around easily cross-team, it's very worthwhile to be able to easily sketch with a high degree of fidelity a prototype which has such a high degree of resemblance to what it will end up being, and more importantly than being able to do this, is the ability to redo it or simply change a few things and not need to start from scratch, of course there is a lot of tool that do this in some way shape or form, but below are a few of the reason why choose Figma over other similar tools:[96], [97]

- It's super easy to use with a lot of drag and drop options for prototyping.
- a built-in commenting that can be added anywhere in the design and to even tag people.
- The ability to inspect, export, copy CSS, iOS and android code and even download icons and images directly from the project URL.
- A built-in version control history for all collaborators with the ability to roll back or even to fork from a previous version/state.

All in all, Figma just seemed to fit like a glove with the cyclical nature of Agile development and both the stand-up and sprint meetings practiced at Equilab, how we could easily show to everyone and come up with better versions each week it was like a godsend.

Material Design Guidelines & Flexbox

Long are the days when each app wanted to be different and unique...well not entirely, companies still stride to be different and unique of course but not the feel of it, companies just go about it differently now, you see companies now work hard to implement guidelines and rules to all aspects of development, in this particular case I shall approach the User Interface (UI) principles and guidelines.

The principles are generally abstract and have high generality & low in authority and are widely applicable and enduring, while guidelines can guide/advise on how to achieve a principle but usually are narrowly focused. There were three major principles I strived to achieve in my work and they were:

- **Learnability**: the ease with which new users can begin effective interaction and achieve maximal performance.
- **Flexibility**: the multiplicity of ways the user and system exchange information.
- **Robustness**: the level of support provided to the user in determining successful achievement and assessment of goal-directed behaviour.

To achieve this consistently we at Equilab adhered to the guidelines of Material Design (MD), in a way it's a visual language that synthesizes the classic principles of good design with the innovation of technology and science. Since more companies and app developers are following these guidelines, we are all achieving a single underlying system that unifies the user experience across platforms, devices, and input methods. MD achieves this because it inspires itself on the physical world and try to replicate it on the virtual one, this way we can capitalize on the user's knowledge of the real world and use it on the virtual one and by doing so it reduces the learning curve to start using the app.

MD has three main sections to getting around all of this it is:[98]

- **Material System**: an expanded and enhanced design system is unified with material tools and components to improve workflow between design and development.
- **Material Foundation**: Design and strategize how to build your app using Material Design architecture, while learning the principles and theory that underpin Material Design.
- **Material Guidelines**: Customize and deploy a unique Material theme systematically across your product – from design to code.

Now to achieve these principles and guidelines we made use of a little something named Flexbox, and this is a new layout mode in CSS3, it distributes space along a single column or row.

In very simple terms it means we create *flex containers* and we set the direction (it's horizontal when not specified) the *flex items* inside the container are to be placed (you can have containers inside containers). Of Course, this is only the beginning since after this you kind of have full control over it, you can even move the items singularly or by group specific amounts or by dimension, but it's preferable to use one or multiple of the many existing properties for both the behaviour or look of the container or the items inside it, some of the most used properties when working with flexbox is flex-direction which defines what is going to be the main axis inside the container by choosing row they will be placed horizontally from left to right, row-reverse horizontally from right to left, column vertically from top to bottom or column-reverse vertically from bottom-up.

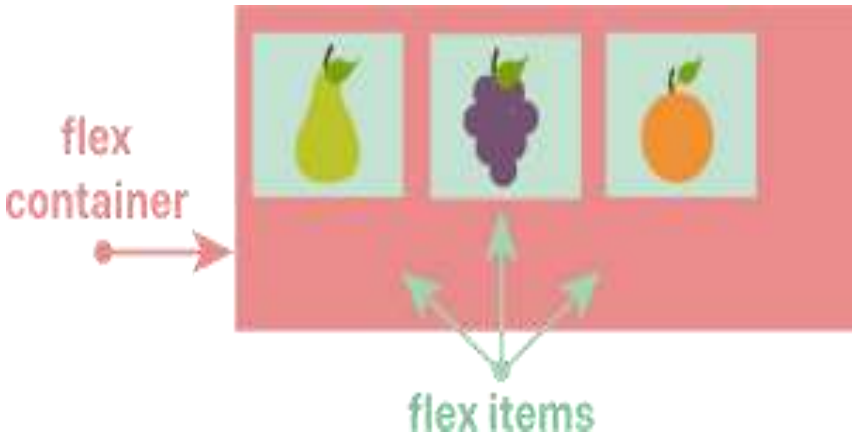

Figure 12 - Flex container and Flex items[99]

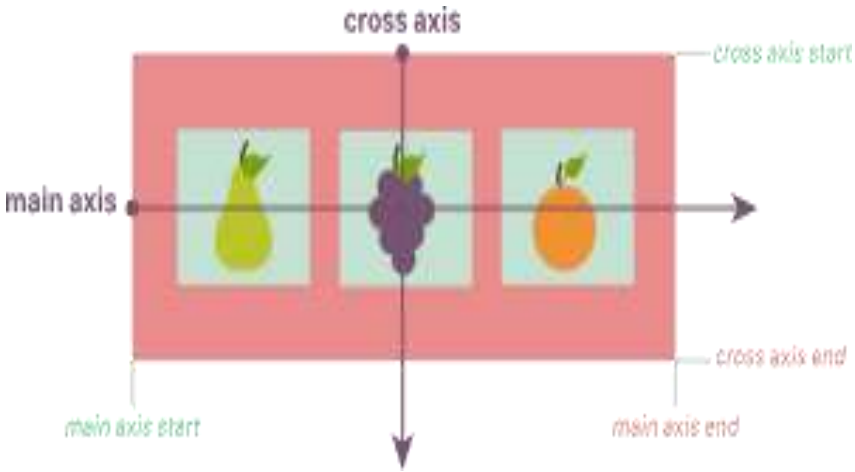

Figure 13 - Axis and cross-Axis of flex-direction row[99]

The property "justify-content" which depending on the main axis chosen with "flex-direction" will let you choose how you want to place out the items inside the container, "flex-start" will put the items at the beginning of the main axis, "flex-end" will place it at the end of the main axis, "center" will centre them out, "space-between" will put the first one in the start of the main axis, the second at the end of the main axis and all the consecutive ones spaced

between, “space-around” is a conjunction of space-between and centre. There is pretty much an unlimited amount of options available with new properties constantly being made available by third parties, bellow is a snippet of part of the flexbox for container and items inside them.

```
export default {
  container: {
    flex: 1,
    backgroundColor: '#F5F5F5',
  },
  containerGoal: {
    flexDirection: "column",
    backgroundColor: 'white',
    minHeight: 55,
  },
  containerGoalHeader: {
    flexDirection: "row",
    justifyContent: 'space-between',
  },
  containerShowGoal: {
    flexDirection: 'row',
    maxWidth: 250,
    justifyContent: 'center',
  },
  mainGoalText: {
    fontSize: 20,
    fontWeight: 'bold',
    paddingTop: 8, //13.5
    paddingBottom: 4,
    paddingLeft: 14, //17
    textAlign: 'left',
    color: '#02020b',
  },
  containerGoalCheckbox: {
    marginLeft: 8,
    marginTop: 8,
    justifyContent: 'flex-start', //'flex-start' || 'flex-end'
  },
  containerSubgoalInput: {
    flexDirection: "row",
    justifyContent: 'space-between',
    paddingLeft: 16,
  },
  containerSubgoalAndPlus: {
    flexDirection: "row",
  },
  containerSubgoalBody: {
    flexDirection: "row",
    justifyContent: 'space-between',
  },
```

Figure 14 - Flexbox snippet for the styles of some of the containers and items

React Native Examples

In this chapter I shall analyse a few snippets of code, made during my internship to bridge the code demonstrated here with how it looked like in the later chapter found in prototype 4. First, remember that we moved from Firebase to Firestore mid development, and in firestore everything is either a Collection or a Document and in our case we always had Collection followed by document sometimes multiple time, so, regarding the first screen on Figure 30, to fetch the data to be displayed in said screen we would make a firestore query to fetch the information on our DB that matched the user that would be in the users collection and to fetch the list of all his goals, then after getting said list we would sort them out by time of creation, and after that we would sort them by if it's either completed goal or an uncompleted goal and return the sorted array, worth mentioning we always added an empty goal as the first item of the array and it shall be explained the reason for the next figure, this whole process is displayed in Figure 15.

```
getItems() {
  const user = firebase.auth().currentUser.uid;
  this.unsubscribe = firebase
    .firestore()
    .collection('users')
    .doc(user)
    .collection('goals')
    .onSnapshot(querySnapshot => {
      let sortedGoals = []
      //added for making the "Add Goal" button appear on the flatlist Component
      sortedGoals.push({ id: '', name: '', subgoals: [] })

      querySnapshot.forEach(doc => {
        let goal = doc.data();
        goal.id = doc.id
        sortedGoals.push(goal)
      });

      sortedGoals.sort((one, two) => {
        if (!one.id) {
          return -1
        } else if (!two.id) {
          return 1
        }

        if (one.completedAt && !two.completedAt) {
          return 1
        } else if (two.completedAt && !one.completedAt) {
          return -1
        }

        return one.createdAt < two.createdAt ? 1 : -1
      })

      this.setState({
        sortedGoals,
        isLoading: false
      })
    }, error => {
      console.log(error) // TODO Log or tell user
      this.setState({
        isLoading: false
      })
    })
}
```

Figure 15 - getItems function, it searches and sorts all goals from the DB

After this sorting process was finished we would go and render said goals to be displayed

like in Figure 30 and to determine if it would have a white (create a new goal), greyed-out (uncompleted goal) or orange (completed goal) badge associated with it and to get this we would run each object on the array through a few if statements, after that and since the array was already sorted the user would always get a view where the first button clickable would be to create a new goal, next display all uncompleted goals and finally display all the completed goals, this can be seen in Figure 16.

```
renderItem = ({ item }) => {
  //making the add goal button

  let image;
  if (!item.id) {
    image = require("./img/add_goal_badge.png")
  } else if (item.completedAt) {
    image = require("./img/goal_badge_18.png")
  } else {
    image = require("./img/unfinished_goal_badge.png")
  }

  let content;
  if (!item.id) {
    content = I18n.t('goals.add_goal')
  } else {
    content = item.name
  }

  return (
    <View style={styles.li}>
      <TouchableOpacity
        activeOpacity={1}
        onPress={() => { this.navigateToGoal(!!item.id, item) }}
      >
        <View style={styles.containerImage}>
          <Image
            source={image}
            style={{
              alignSelf: 'flex-start'
            }}
          />
        </View>
        <View style={styles.containerText}>
          <Text style={styles.textComp}>
            {content}
          </Text>
        </View>
      </TouchableOpacity>
    </View>
  );
}
```

Figure 16 - Function to recognise which icon to give each goal fetched

Another meaningful snippet of code(Figure 17) relates to one of the feedbacks from the user testing mentioned in prototype 4, previously only one Text Input field would display at time and after finishing said input we would save it one by one in the array as a string (or multiple strings), to accommodate this we would instead create an object for each sub goal, and each object would have also an interactive Text Input field associated that could be modified anytime (while previously a button had to be pressed which would take you to a new screen and modify It

like shown in Figure 25 on chapter Prototype 2)

```
renderSubgoalRow = ({ item, index }) => (
  <View style={styles.containerSubgoalRender}>
    <View style={styles.containerSubgoalText}>
      <Icon2 name="circle" style={styles.btnIconCircle} />
      <TextInput
        underlineColorAndroid={'rgba(0,0,0,0)'}
        style={{
          borderColor: 'transparent',
          color: 'black',
          fontSize: 18,
          justifyContent: 'center',
        }}
        autoCorrect={false} //added to deal with underline on text
        placeholderTextColor='grey'
        multiline
        placeholder='Add Subgoal'
        onChangeText={text => {
          item.name = text
          this.setState({ subgoals: [...this.state.subgoals] })
        }}
        value={item.name}
        blurOnSubmit={true}
        onSubmitEditing={() => this.handleSpecificSubgoalInputSubmit(item)}
      />
    </View>
    <TouchableOpacity
      activeOpacity={1}
      onPress={() => this.toggleSubgoalCheckbox(item, index)}
    >
      { (item.completedAt !== null) ? (this.showCheck()) : (this.showUncheck()) }
    </TouchableOpacity>
  </View>
)
```

Figure 17 - Function which renders sub goals

DB Architecture

Like previously stated Firebase is a NoSQL Database, it uses JSON notation to be able to structure the data and store it in the firebase Realtime Database, we need careful planning in how it should be stored as to get the information back in a fast and reliable way. Firebase has a built-in system of creating itself a unique identifying key, but also gives the ability of yourself dealing with that matter. As such for my case we went back and forth quite a lot in this particular case, and I'll go over both of them.

First we went with a very basic DB structure, firstly it was separated from the company DB, also we still weren't sure as to what was going to be stored in particular since the questionnaires (which was the responsibility of another colleague) for the users weren't finished yet but the information itself was not as important at this point in time, we just had to be sure that the information could be store and displayed!

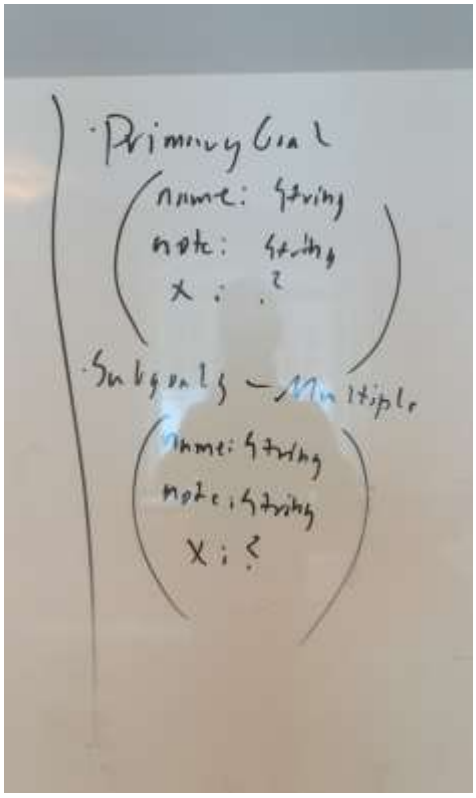

Figure 18 - First sketch of the database

The identifying keys at this point were being generated automatically by firebase, it also aggregated posterior sub goals into the primary goal which lead to a very clunky way of fetching information associated with a specific goal, particularly it searched every sub goal in the DB and looked for a specific field which contained the information of the primary goal it was associated with, of course this consumes lots of unnecessary resources for the server and makes the process

take way longer than it needs to be, early on this just proved how important it is to plan ahead and very thoroughly all the details when dealing with DB, even the NoSQL ones.

Worth mentioning that not all changes to the DB were to accommodate or deal with problems regarding the backend, some changes were to accommodate the way of how things were going to be displayed in the app itself. Also we made a small migration from firebase to firestore, firestore is pretty much like firebase except it's more appropriate for mobile apps that possess a more complex query system and most importantly it's made of either collections or documents[100], the first thing was to adapt the goal and sub goals to collections and each particular goal and sub goal to have an associated key and be a document part of said collection. This way every time we were dealing with a goal, we automatically got the associated array of sub goals with it, this improved the speed of fetching the information from the DB considerably. The latest version of our DB used timestamps as the key of each goal and sub goal, since all of these inputs were to be made by users in their phones, it would be pretty much impossible for one user to input two goals or sub goals in the same millisecond, this way this field worked not only as a identifying value but as a sorting value when choosing a parameter to display information by the timestamp itself.

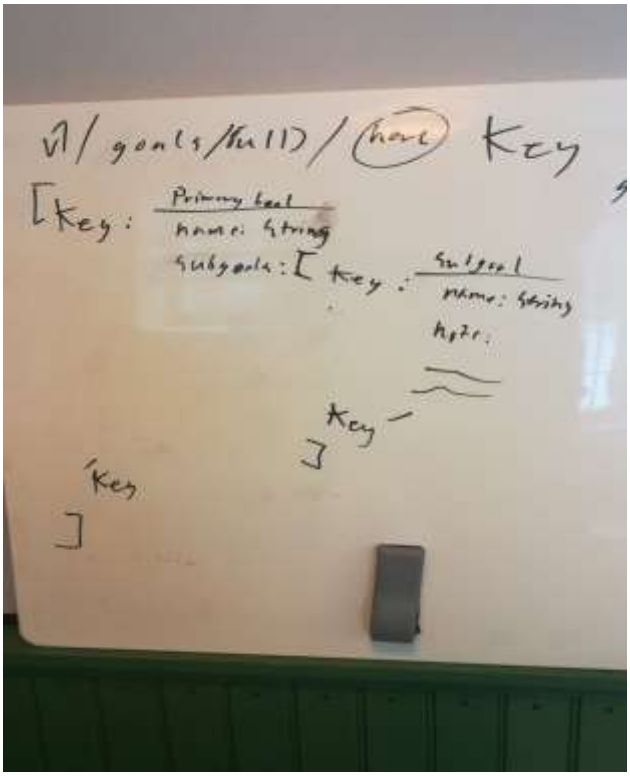

Figure 19 - Sketch of how the DB should look-like on firestore

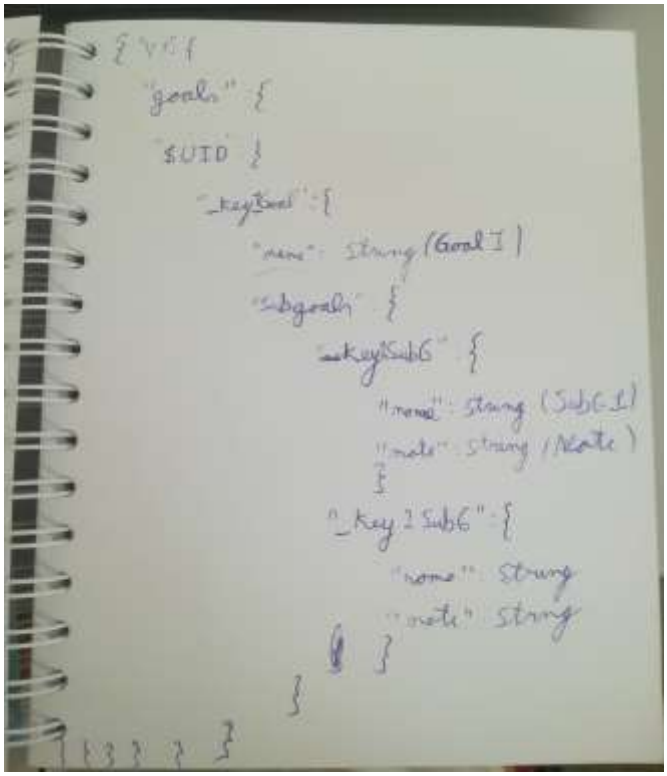

Figure 20 - How it looked like with an example

Brainstorming & Sketching

In this section the actual implementation details are revealed. Starting by how we went from acknowledging and accepting the principles of Agile Development as our guiding framework, how we used the framework's and BaaS features to build the best possible app for our users.

Firstly let's go about how the Brainstorming went, specially how the first idea for the feature I was to develop came about, as previously stated Equilab is an App for horse riders, mostly tailored for joyrides more than professional riding, even though we provide quite the functionally for professionals, and among the riders inside the company the talks about how someone was using another app to write down challenges to be achieved while riding and using the data provided by Equilab to track down the completion of the goal, this was the spark that ignited all the work I was going to do.

After the idea had settled in, came the work of how it would look and work, in total and in the period of five month the UI and even some parts of the backend of the feature changed to a grand total of 5 times, some of the changes were part of the following of the material design guidelines, some came about the feelings from fellow users of the app inside the company, and ultimately came the change that was thanks to users of the app with no previous knowledge of this new feature to test and give their opinions, worth mentioning we were conducting our very own unmoderated user testing and looking for all the "hiccups" on the user while testing the feature, one of these hiccups was a button of confirmation that was supposed to be pressed when something had to be saved, but we noticed in quite a few of the users they pressed it before anything was even written, of course pressing it with nothing to be saved was the same as pressing an empty space on the phone so no harm there, but we noticed this and we removed the button and made it appear after something was ready to be saved, this in turn led people to understand the functionality of the button without telling them.

I'll go over all the versions and the changes that occurred in them in a later chapter. This of course are the big moments of the development, but many small changes came pretty much weekly or even daily, thanks to the spring meetings with all the members of the company.

6- Prototypes

UI Evolution

Usually the cycle would go a bit like this:

- **Planning**: Firstly, we would go about how we thought the process of writing down goals and sub goals would go about, what the user wanted to do, how the user should go about doing it, and even what would be important to see at each step of the process.
- **Design**: Then to design the UI of the feature to be developed, the main program used to sketch our idea was Figma and while doing it respecting the material.io guidelines.
- **Develop**: Most of my job was spent here, making all our ideas and objectives come to life, trying to merge the agile methodology and the feature into one.
- **Test**: We would do internal testing and external testing.
- **Evaluate**: We would go around many points of interest to evaluate, the speed of performing one task or multiple ones, the feel of the navigation and the animations of each developed aspect, the aesthetics of the “empty screen”, and even how to make each part distinct and meaningful.
- **Meet**: The meet was made mostly with my superior and we would go over the results obtained in the Test and Evaluate parts of the cycle and decide how to make the next iteration.

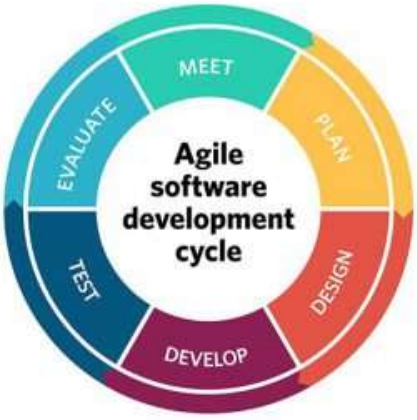

Figure 21 - The life cycle of Agile Software Development[101]

Of these 6 steps, I only worked on the design, develop, test and evaluate parts, while someone on the team did the other two, but like mentioned previously, one of the benefits of Agile over something like waterfall is the “possibility” of skipping one ore more steps of the wheel, in our case sometimes we would make a “cycle” with only the design, develop and testing steps, leading to some changes of the prototype with only this sub-cycle.

Testing

This was probably one of the most important aspects of the entire cycle by a long-shot, specially since our job was to develop a tool to be used by a wide range of users, it's of the most importance to continually do testing's with the people using the product, this part was mostly done with two different sets of users, first we would do almost weekly internal testing with two people that were both developers and horse joyriders, these were the ones that helped with testing the "big load" of content being developed, things like the general flow of inputting data, consulting said data, and the overall flow of task to be done for the functionality to do it's proper work. This kind of testing were the major factor of deciding the first changes that lead to making the next set of testing.

Next, we move to probably the most important set of testing, the user testing, we would either take phones containing the developer version of the app being worked on to a horse stable and ask people to test the new changes or take people to HQ and do the testing in there. These testing were the most important because, generally they were the ones responsible for the biggest changes on the overall look of the app (and in one case even making us change the back-end of both the app and the database to accomplish the new feedback). I shall go over a few of these cases in each Prototype version discussed below.

Prototype 1

First, let me explain what was expected the user to achieve with this new functionality, generally for many horse riders, both professional and joyrides alike, they usually have goals to achieve while riding their horse, these can be quite different from user to user, one user send an email to us saying something like: “I want to have the possibility to write on the app that during this summer I want to ride all the horses of the stable I go to, and see which horse I have already done this” and during our testing of other functionality of the app we heard other similar request, so we analysed these kind of feedback and came up with an abstract structure of what our users seemed to ask of us, and we came up with a Goal and Sub-goal list, which lead us to the prototypes displayed bellow.

All these images are from the site Figma used to make the prototypes that guided my development. The goals and sub goals for the users and analyse the reasoning used for the decisions and see what we can learn from the entire journey and discover the shortcuts to employ for future work. First lets state how was the flow of inputs the user had to make to fully write a goal and sub goal.

On its very own profile we were to add a new button, the “Add Goal” button which lead to a new screen (this screen was developed from scratch and later redone multiple times as we’ll see), the first thing the user had to do was to write down the name of the goal, then and only then would the ability to write down the name for the sub goal would unlock after which a new screen would automatically open where it would show on the header the name of the goal and a list of sub goals pertaining to said goal, by clicking this sub goal would open up a third screen to write down notes and training and even choose a date to be the limit for the completion of said sub goal, this part of choosing a date was going to be a new feature and was being developed in parallel with another member of the team, to which was only going to be fully developed once the basics of every feature was fully done. Upon all checkboxes being ticked a new button would show up giving the user the ability to fully complete the goal and be congratulated with a new screen celebrating this new achievement, also worth mentioning we wanted a clear separation from completed sub goals and uncompleted sub goals.

As we can see here this first version was rooted mostly on logic and straight line action (on the later version I’ll show how we open up the way of doing this very act in in different order) and there was mostly only one way of doing it, also we later delved into how to merge the logic with all the Agile knowledge and Material Design.

Figure 22 - Profile, Goal, Sub Goal and Extra's Screens

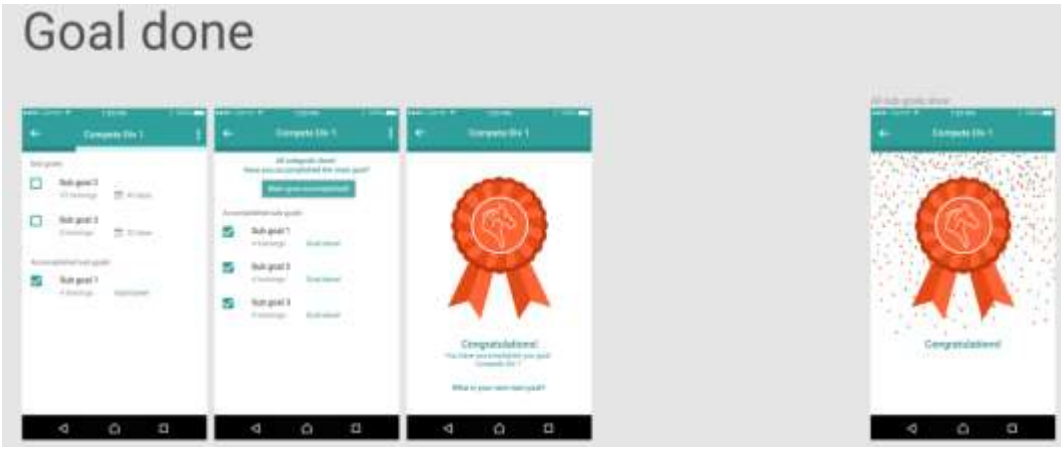

Figure 23 - Uncompleted Sub Goals, Completed Sub Goal and Congratulatory Screens

Prototype 2

Let’s go over what changed between these versions, we were still stuck on our ways of preventing our users from doing things in an order we did not want, so we separated the input of the main goal from the sub goal, by making a screen for only writing the name of the goal.

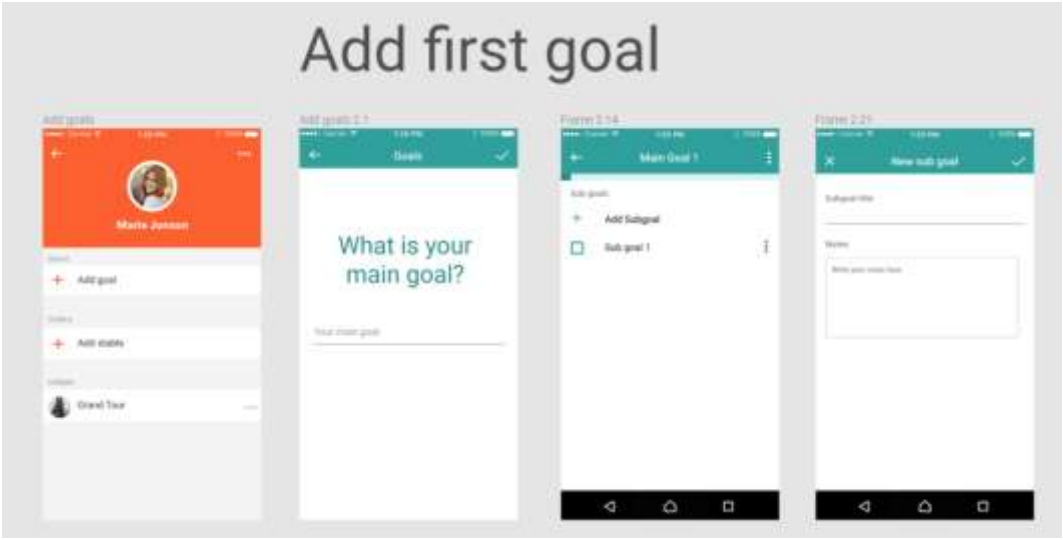

Figure 24 - New Goal and Extra's screen

After that step was done the sub goals screen would show up giving the ability to write new sub goals and also display the list of sub goals both completed and uncompleted, notice how this time around there were no sections separating the completed and uncompleted sub goals, we came to this change because in our internal testing we did not like how the sub goals would “jump” around mid-clicking and instead we went with greying out the completed ones, this way it would still give the user the sense of differentiation between both of these sub goals and make it less confusing.

Figure 25 - New Sub Goals screen also with the new edit buttons

Last but not least we changed the screen of inputting notes and training and dates for each sub goal, we decided that we wanted to make a stand-alone version of both the goal and calendar and to connect them at a later date, so on the screen displayed in Figure 23 we added a “more vertical” icon in front of each sub goal, that would give you the option of either deleting the sub goal, or to edit it, and this option would take you to a new Extra screen where you could add notes and to rename the sub goal (on the image we see a typo that was later corrected on the developed version, Goal Title to Sub goal title)

Figure 26 - Extra Screen

Prototype 3

The third prototype, this one was the version that suffered the most changes overall, especially at the backend part of it, requiring the redevelopment of the code from scratch to accommodate the possibility of trying to do this task in different orders, also the introduction of something new to the company, showing sponsored content.

Figure 27 - Revamped Goal and Sub Goal screens, Extra's mostly intact

Something very peculiar came to us amidst our internal testing, one of the developers observed that iOS app's uses mostly circular checkboxes to the left of the content, while in android it's mostly square checkboxes to the right of the content, this took us by surprise that something so simple eluded us for so long, we then proceeded to spend an entire day looking for something that would feel natural to both iOS and Android users alike, we finally landed on circular checkboxes like iOS but on the right side of the content like android, a middle ground to try and please both user bases.

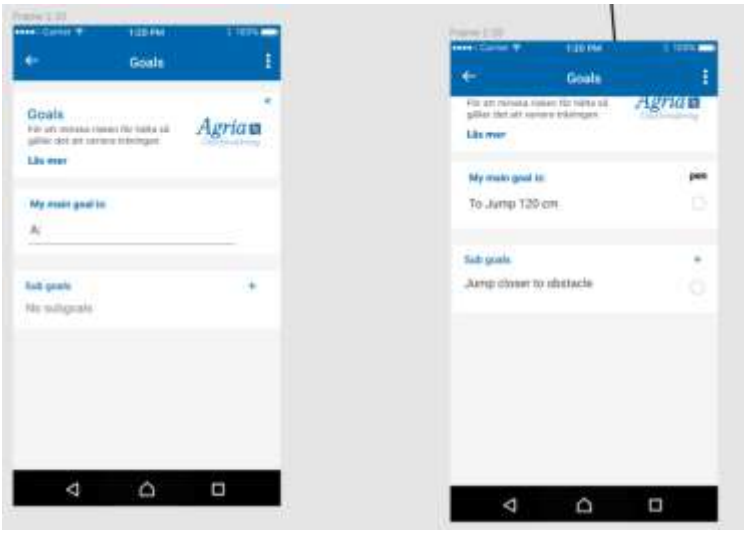

Figure 28 - Goal screen, closer inspection

Finally, we tried to come up with a more rustic way of showing up the list of accomplished goals, we felt it would go well with our users since they are accustomed to this kind of background in real life, we tried to connect both realms once again. We also fiddled around multiple way of displaying the moment the user finishes up a goal how he would go back to the list of all completed goals.

Figure 29 - The rustic screen of Accomplished Goals

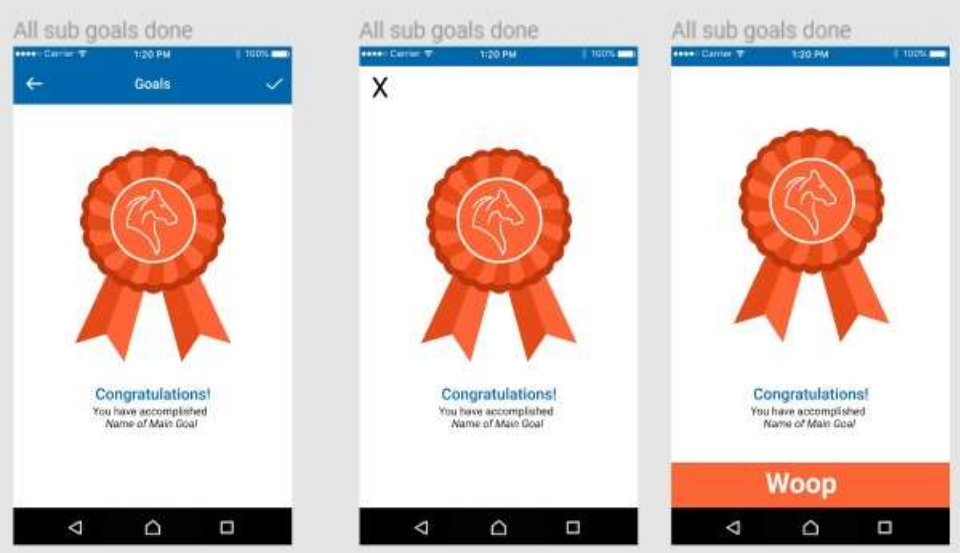

Figure 30 - Different ways of navigating back

Prototype 4 (Final version)

We are getting very close to the final version, most of the changes are just re-position of text and icons or resizing and moving around small stuff. The biggest change was creating a new screen and in it merge the previous button that was on the profile screen of the user with the old completed list of goals screen, into one single screen where it display a list of icons where you can create a new goal, select unaccomplished goal or revisit an accomplished goal, and even though the shape for all three of them are the same, the colouring is the one that helps make the distinction among them and also merged the block separating goal and sub goals

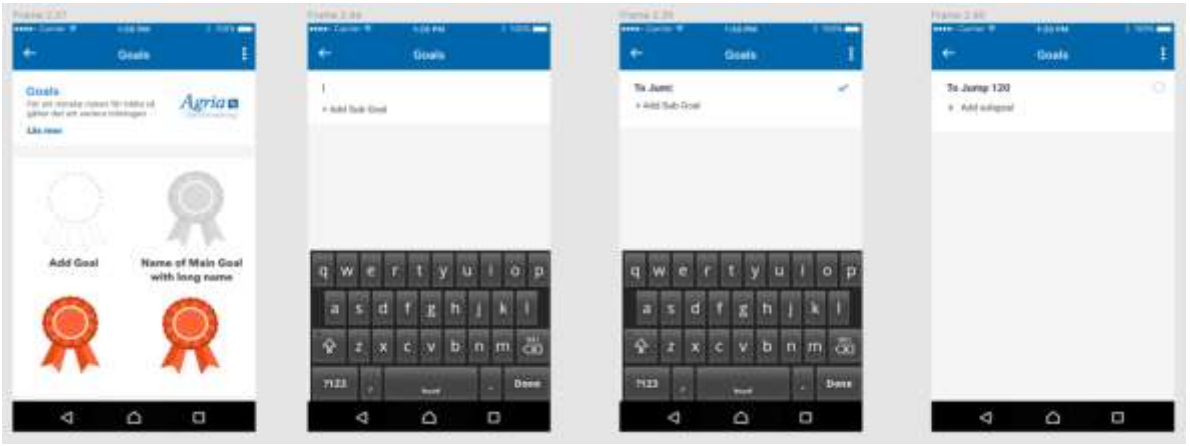

Figure 31 - The new List of Goals screen and Goal/Sub Goal screen

We also implemented a feature requested by some testers, that the moment of writing one sub goal is finished(by pressing enter on the phone keyboard) to open up a new text field to write another, so to give the ability to input multiple sub goals without the need of constant tapping on the screen, while on the past every time a new sub goal had to be inputted a button was needed to be pressed first.

Figure 32 - How the flow of inputting new sub goals looked like

7- Lessons Learned

All in all this internship was a great experience for me and not only in a professional way to be honest, but I have seen that it's also pretty hard to transition from student to worker, and that even when you do make that transition, you never stop the "learning" aspect of the student life, in fact I'd wager that the learning is even more important at the working phase than when studying since failing to adapt in a competitive market can spell doom for your life.

One of the first things I would argue to create at least a simple "Developer Documentation", after a while the project starts getting pretty big, and you will get different people working on different aspects of the code, and after that if you are still in business you will probably get to a point where you need to get more people involved who did not accompanied the whole process from scratch, and they need to do something very similar to what was previously done… well I did come up with this particular scenario and you would think it is not of any particular degree of difficulty to resolve it, but after a while when you have way over hundreds of files in a project you tend to blur up a bit of where and when things were done and even for what specific purpose any method was developed for, this is where a good Documentation comes in, or any documentation for that matter.

Also, that the constant use of users for testing your developments cannot be overstated, its so easy to overlook so many simple details over the most diverse range of topics (like the difference between checkboxes among Android and iOS users), that it becomes almost impossible for one or a small team of developers to keep in mind of everything, also like many developers know when learning agile, that it's so much easier and cost-efficient to make big changes early on than to make them later on, so better safe than sorry.

8- Conclusions

The internship as a whole served more than just the “final nail” in my master’s degree, not only did it answered all the initial questions posed in the motivation chapter, but the answer were all very positive, I am now certain that whichever area I end up working with I will do my best at it and it will be more than enough, that if something goes wrong I and my group shall fix it, that to make something happen all you need is determination and time, and finally I know now that I love what I do because I get a kick out of fixing problems, and there’s no shortage of that in this area as a whole.

Multiple types of research was made, among them download and usage of apps that gave the option to write objectives or to do list (Google Keep being a big factor), internal and external testing on all the prototypes made, to which each iteration proved we could answer both our own expectation and those of our users which kept us ever more engaged with our audience, and even though I was not fully responsible for either the backend and frontend I was still fortunate to work on both of them, which gave me a great opportunity to test me and reassure myself, once again that all you need is determination and perseverance.

I believe React Native will become a major framework soon, mostly because of third party developer’s enthusiasm and community sense among them, with a high degree of helping new and old developers and the ever-increasing number of library and components with easy use and functionality the amount of RN enthusiast will surely rise, also Firebase with it’s easy to use and built-in functions ready to deploy will ensure that companies can spend their time working on what really matters for their users.

References

[1] «Erasmus EF», *Erasmus Educação e Formação*. [Em linha]. Disponível em: http://erasmus.dev. [Acedido: 17-Jul-2018].

[2] «ERASMUSINTERN». [Em linha]. Disponível em: https://erasmusintern.org/. [Acedido: 17-Jul-2018].

[3] «Erasmus Internship», *Training Experience*. [Em linha]. Disponível em: https://www.trainingexperience.org/. [Acedido: 17-Jul-2018].

[4] «ESPA», *ESPA*. [Em linha]. Disponível em: https://www.espauk.com/. [Acedido: 17-Jul-2018].

[5] «International Student Placement Office, Aiding the international development of UK Businesses». [Em linha]. Disponível em: https://www.ispo.co.uk/. [Acedido: 17-Jul-2018].

[6] Equilab, «Equilab, track your exercises, created for passionate riders», *Equilab*. [Em linha]. Disponível em: https://equilab.horse/. [Acedido: 17-Jul-2018].

[7] «React Native · A framework for building native apps using React». [Em linha]. Disponível em: https://facebook.github.io/react-native/. [Acedido: 16-Out-2018].

[8] C. Esplin, «What is Firebase?», *How To Firebase*, 24-Out-2016. [Em linha]. Disponível em: https://howtofirebase.com/what-is-firebase-fcb8614ba442. [Acedido: 29-Ago-2018].

[9] «IBM Simon: World's first smartphone is now 20 years old - Android Authority». [Em linha]. Disponível em: https://www.androidauthority.com/ibm-simon-birthday-134255/. [Acedido: 26-Jan-2019].

[10] S. Richmond, «Smartphones hardly used for calls», 29-Jun-2012.

[11] C. Chan, «This Lost Speech from 1983 Will Make You Think That Steve Jobs Was from the Future», *Gizmodo*. [Em linha]. Disponível em: https://gizmodo.com/5948434/this-lost-speech-from-1983-will-make-you-think-that-steve-jobs-was-from-the-future. [Acedido: 19-Jul-2018].

[12] «EPOC (operating system)», *Wikipedia*. 28-Mai-2018.

[13] «Palm SO | Palm Source - Palm Software and Palm OS». [Em linha]. Disponível em: https://www.palmsource.com/palmos/. [Acedido: 30-Out-2018].

[14] «*Blockade* (video game)», *Wikipedia*. 30-Mar-2018.

[15] «What is WAP (Wireless Application Protocol)? - Definition from WhatIs.com», *SearchMobileComputing*. [Em linha]. Disponível em: https://searchmobilecomputing.techtarget.com/definition/WAP. [Acedido: 19-Jul-2018].

[16] «Apple Presents iPod», *Apple Newsroom*. [Em linha]. Disponível em: https://www.apple.com/newsroom/2001/10/23Apple-Presents-iPod/. [Acedido: 19-Jul-2018].

[17] «Apple Launches the iTunes Music Store», *Apple Newsroom*. [Em linha]. Disponível em: https://www.apple.com/newsroom/2003/04/28Apple-Launches-the-iTunes-Music-Store/. [Acedido: 19-Jul-2018].

[18] «iTunes Music Store Sells Over One Million Songs in First Week», *Apple Newsroom*. [Em linha]. Disponível em: https://www.apple.com/newsroom/2003/05/05iTunes-Music-Store-Sells-Over-One-Million-Songs-in-First-Week/. [Acedido: 19-Jul-2018].

[19] B. X. Chen, «June 29, 2007: iPhone, You Phone, We All Wanna iPhone», *Wired*, 29-Jun-2009.

[20] K. M. Thursday, March 06, 2008, e 01:40 pm PT, «Apple posts iPhone Software Roadmap event stream», *AppleInsider*. [Em linha]. Disponível em: https://appleinsider.com/articles/08/03/06/apple_posts_iphone_software_roadmap_event_stream. [Acedido: 30-Out-2018].

[21] «iPhone App Store Downloads Top 10 Million in First Weekend», *Apple Newsroom*. [Em linha]. Disponível em: https://www.apple.com/newsroom/2008/07/14iPhone-App-Store-Downloads-Top-10-Million-in-First-Weekend/. [Acedido: 20-Jul-2018].

[22] L. Rainie, Janna, e erson, «Scenario 1: The Evolution of Mobile Internet Communications», *Pew Research Center: Internet, Science & Tech*, 14-Dez-2008. .

[23] «App Store Downloads Top 100 Million Worldwide», *Apple Newsroom*. [Em linha]. Disponível em: https://www.apple.com/newsroom/2008/09/09App-Store-Downloads-Top-100-Million-Worldwide/. [Acedido: 20-Jul-2018].

[24] «Android turns 10: Remembering the first Android phone, the T-Mobile G1 / HTC Dream». [Em linha]. Disponível em: http://www.myjoyonline.com/technology/2018/September-25th/android-turns-10-remembering-the-first-android-phone-the-t-mobile-g1-htc-dream.php. [Acedido: 17-Out-2018].
[25] «Apple Gets A Trademark: There's An App For That™», *Cult of Mac*, 12-Out-2010. .
[26] «Handango – Mobile Content Delivery for the Real World». .
[27] B. S. Segan, 2009 9:20AM EST April 1, e 2009 April 1, «RIM Launches BlackBerry App World», *PCMAG*. [Em linha]. Disponível em: https://www.pcmag.com/article2/0,2817,2344187,00.asp. [Acedido: 20-Jul-2018].
[28] C. Aguilar, «How WhatsApp Acquired 900m Users To Literally BECOME A Word-of-Mouth Giant», *Word-of-Mouth and Referral Marketing Blog*, 12-Nov-2015. [Em linha]. Disponível em: https://www.referralcandy.com/blog/whatsapp-marketing-strategy/. [Acedido: 17-Out-2018].
[29] «The Full List Of The Top iPhone And iPad Apps Of 2010», *TechCrunch*. .
[30] A. Press, «US secretly created "Cuban Twitter" to stir unrest and undermine government», *The Guardian*, 03-Abr-2014.
[31] P. G. October 21 e 2009 Operating systems, «Windows 7 launched by Microsoft», *TechRadar*. [Em linha]. Disponível em: https://www.techradar.com/news/software/operating-systems/windows-7-launched-by-microsoft-643946. [Acedido: 17-Out-2018].
[32] «Windows Phone Store», *Windows Central*, 07-Ago-2014. [Em linha]. Disponível em: https://www.windowscentral.com/windows-phone-store. [Acedido: 17-Out-2018].
[33] «Geinimi Android Trojan horse discovered – Naked Security». [Em linha]. Disponível em: https://nakedsecurity.sophos.com/2010/12/31/geinimi-android-trojan-horse-discovered/. [Acedido: 23-Jul-2018].
[34] «Android Trojan dubbed 'Geinimi' found in legitimate applications». [Em linha]. Disponível em: https://phys.org/news/2010-12-android-trojan-dubbed-geinimi-legitimate.html. [Acedido: 17-Out-2018].
[35] A. P. January 8, 2011, e 6:43 Am, «Word of the Year? "App"!» [Em linha]. Disponível em: https://www.cbsnews.com/news/word-of-the-year-app/. [Acedido: 23-Jul-2018].
[36] «Introducing the Amazon Appstore for Android : Appstore Blogs». [Em linha]. Disponível em: https://developer.amazon.com/blogs/appstore/post/Tx2VH5RWFII9F8S/introducing-the-amazon-appstore-for-android. [Acedido: 17-Out-2018].
[37] S. Dredge, «Zynga has lost 63% of its players. Is NaturalMotion the answer?», *The Guardian*, 20-Mar-2014.
[38] C. Farivar, «How Zynga went from social gaming powerhouse to has-been», *Ars Technica*, 09-Dez-2013. [Em linha]. Disponível em: https://arstechnica.com/information-technology/2013/09/how-zynga-went-from-social-gaming-powerhouse-to-has-been/. [Acedido: 23-Jul-2018].
[39] S. Dredge, «Apple bans satirical iPhone game Phone Story from its App Store», *The Guardian*, 14-Set-2011.
[40] M. Brown, «Apple Bans Phone Story Game That Exposes Seedy Side of Smartphone Creation», *Wired*, 14-Set-2011.
[41] «Job Creation», *Apple*. [Em linha]. Disponível em: https://www.apple.com/job-creation/. [Acedido: 23-Jul-2018].
[42] J. Topolsky, «Hello, Google Play: Google launches sweeping revamp of app, book, music, and video stores», *The Verge*, 06-Mar-2012. [Em linha]. Disponível em: https://www.theverge.com/2012/3/6/2848223/google-play-store-rebranded-android-market. [Acedido: 23-Jul-2018].
[43] E. M. Rusli, «Facebook Buys Instagram for $1 Billion», *DealBook*, 1333991748. [Em linha]. Disponível em: //dealbook.nytimes.com/2012/04/09/facebook-buys-instagram-for-1-billion/. [Acedido: 23-Jul-2018].
[44] K. Hill, «10 Reasons Why Facebook Bought Instagram», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/kashmirhill/2012/04/11/ten-reasons-why-facebook-bought-instagram/. [Acedido: 23-Jul-2018].

[45] D. Thier, «Angry Birds Hits a Billion Downloads», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/davidthier/2012/05/09/angry-birds-hits-a-billion-downloads/. [Acedido: 23-Jul-2018].
[46] S. Dredge, «Minecraft and Candy Crush Saga top Apple's 2013 app charts», *The Guardian*, 17-Dez-2013.
[47] «The App Store's 50B Downloads Vs. Google Play's 48B: Android Closes The Gap», *TechCrunch*. .
[48] «The Rise and Fall of Flappy Bird | Listen & Read | Spotlight English». [Em linha]. Disponível em: https://www.spotlightenglish.com/listen/the-rise-and-fall-of-flappy-bird. [Acedido: 23-Jul-2018].
[49] «"Despicable Me: Minion Rush" Reaches 100 Million Downloads». [Em linha]. Disponível em: https://www.prnewswire.com/news-releases/despicable-me-minion-rush-reaches-100-million-downloads-222987531.html. [Acedido: 23-Jul-2018].
[50] N. Ingraham, «Apple announces 1 million apps in the App Store, more than 1 billion songs played on iTunes radio», *The Verge*, 22-Out-2013. [Em linha]. Disponível em: https://www.theverge.com/2013/10/22/4866302/apple-announces-1-million-apps-in-the-app-store. [Acedido: 23-Jul-2018].
[51] J. Bercovici, «Facebook Tried To Buy Snapchat For $3B In Cash. Here's Why.», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/jeffbercovici/2013/11/13/facebook-wouldve-bought-snapchat-for-3-billion-in-cash-heres-why/. [Acedido: 23-Jul-2018].
[52] K. says, «Can I Still Buy Great Plains Accounting Software?», *ERP Software Blog*, 16-Nov-2015. .
[53] «CRM Software & Cloud Computing Solutions», *Salesforce.com*. [Em linha]. Disponível em: https://www.salesforce.com/eu/. [Acedido: 19-Out-2018].
[54] «The Pre-History of Software as a Service», *SmartBear.com*. [Em linha]. Disponível em: https://smartbear.com/blog/test-and-monitor/the-pre-history-of-software-as-a-service/. [Acedido: 18-Jul-2018].
[55] «Smartphone users worldwide will total 1.75B in 2014». [Em linha]. Disponível em: http://www.microwavejournal.com/articles/21462-smartphone-users-worldwide-will-total-175b-in-2014. [Acedido: 24-Jul-2018].
[56] «App Store Sales Top $10 Billion in 2013», *Apple Newsroom*. [Em linha]. Disponível em: https://www.apple.com/newsroom/2014/01/07App-Store-Sales-Top-10-Billion-in-2013/. [Acedido: 24-Jul-2018].
[57] B. Kepes, «Updated - Facebook Buying WhatsApp - $19B In Cash And Stock», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/benkepes/2014/02/19/breaking-facebook-buying-whatsapp-19b-in-cash-and-stock/. [Acedido: 24-Jul-2018].
[58] P. Olson, «Facebook Closes $19 Billion WhatsApp Deal», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/parmyolson/2014/10/06/facebook-closes-19-billion-whatsapp-deal/. [Acedido: 24-Jul-2018].
[59] G. Kelly, «Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/. [Acedido: 24-Jul-2018].
[60] «5 reasons 97% of all new mobile malware is targeting Android – Safe and Savvy Blog by F-Secure». .
[61] V. Goel, «Facebook Requires Users to Install Separate Messaging App», *Bits Blog*, 15-Abr-2014. .
[62] «Gmail For Android Reaches 1 Billion Downloads, This Event Has Been Archived», *Android Police*, 12-Mai-2014. .
[63] D. E. D. Thursday, July 10, 2014, e 10:55 am PT, «75 billion downloads later, Apple celebrates the App Store's sixth anniversary», *AppleInsider*. [Em linha]. Disponível em: https://appleinsider.com/articles/14/07/10/apple-inc-reaches-sixth-anniversary-of-the-app-store. [Acedido: 24-Jul-2018].
[64] «Anonymish Chat App Wut Raises Seed Round From Foundation, Google Ventures», *TechCrunch*. .
[65] Quora, «Is Apple Responsible For The Hacked Leak Of Private Celebrity Photos Via iCloud?», *Forbes*. [Em linha]. Disponível em: https://www.forbes.com/sites/quora/2014/09/03/is-apple-responsible-for-the-hacked-leak-of-private-celebrity-photos-via-icloud/. [Acedido: 25-Jul-2018].

[66] «Development of Mobile Applications – MIS2101 – Summer 1 2015». [Em linha]. Disponível em: http://community.mis.temple.edu/mis2101sec711summer2015/2015/06/13/development-of-mobile-applications/. [Acedido: 06-Nov-2018].
[67] «BBC launches WhatsApp Ebola service», *BBC News*, 16-Out-2014.
[68] D. Nelson, «Positive Feedback Loop Examples», *Sci. Trends*, Jun. 2018.
[69] «App Download and Usage Statistics», *Business of Apps*. [Em linha]. Disponível em: http://www.businessofapps.com/data/app-statistics/. [Acedido: 26-Jul-2018].
[70] J. Rossignol, «Apple Shifts to Editorially Curated Lists on App Store». [Em linha]. Disponível em: https://www.macrumors.com/2015/06/01/app-store-editorial-curation-shift/. [Acedido: 26-Jul-2018].
[71] «App Submissions On Google Play Now Reviewed By Staff, Will Include Age-Based Ratings», *TechCrunch*. .
[72] Andrew Powell-Morse, «Waterfall Model: What Is It and When Should You Use It?», *Airbrake Blog*, 08-Dez-2016. .
[73] «Comprehensive Guide to the Agile Manifesto», *Smartsheet*, 29-Jul-2016. [Em linha]. Disponível em: https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto. [Acedido: 22-Out-2018].
[74] M. Berteig, «The Agile Manifesto – Essay 2: Individuals and Interactions over Processes and Tools», *Agile Advice*, 13-Jan-2015. .
[75] «Applying Agile Management Value 2: Working Software Over Comprehensive Documentation», *dummies*. .
[76] M. Vincent, C. Williams, M. Vincent, e C. Williams, «Agile Manifesto - Working Software Over Comprehensive Documentation», *Source Allies*, 17-Mai-2013. [Em linha]. Disponível em: https://www.sourceallies.com/2013/05/agile-manifesto-working-software-over-comprehensive-documentation/. [Acedido: 02-Ago-2018].
[77] «Unit Testing», *Software Testing Fundamentals*, 04-Jan-2011. .
[78] «FUNCTIONAL Testing Tutorial: What is, Process, Types, & Examples». [Em linha]. Disponível em: https://www.guru99.com/functional-testing.html. [Acedido: 22-Out-2018].
[79] «Integration Testing», *Software Testing Fundamentals*, 03-Jan-2011. .
[80] «Applying Agile Management Value 3: Customer Collaboration Over Contract Negotiation», *dummies*. .
[81] «Customer Collaboration Over Contract Negotiation», *Sean Van Tyne*, 28-Ago-2016. [Em linha]. Disponível em: http://seanvantyne.com/2016/08/28/customer-collaboration-over-contract-negotiation/. [Acedido: 02-Ago-2018].
[82] «Applying Agile Management Value 4: Responding to Change Over Following a Plan», *dummies*. .
[83] M. Vincent, C. W. Kessler David, M. Vincent, e C. W. Kessler David, «Agile Manifesto - Responding to Change Over Following a Plan», *Source Allies*, 19-Ago-2013. [Em linha]. Disponível em: https://www.sourceallies.com/2013/08/agile-manifesto-responding-to-change-over-following-a-plan/. [Acedido: 06-Ago-2018].
[84] Slack, «Where work happens», *Slack*. [Em linha]. Disponível em: https://slack.com/. [Acedido: 22-Out-2018].
[85] «GitHub», *GitHub*. [Em linha]. Disponível em: https://github.com/github. [Acedido: 22-Out-2018].
[86] C. Aguilar, «Reactive Core architecture for React Native and React applications», *Medium*, 01-Mar-2017. .
[87] C. M. & M. Murphy, «What exactly is React Native and should I learn it?», *Techworld*. [Em linha]. Disponível em: https://www.techworld.com/apps-wearables/what-is-react-native-3625529/. [Acedido: 07-Ago-2018].
[88] S. Yegulalp, «What is NoSQL? NoSQL databases explained», *InfoWorld*, 07-Dez-2017. [Em linha]. Disponível em: https://www.infoworld.com/article/3240644/nosql/what-is-nosql-nosql-databases-explained.html. [Acedido: 29-Ago-2018].
[89] «SQL vs NoSQL: The Differences», *SitePoint*, 18-Set-2015. .
[90] «Design», *Material Design*. [Em linha]. Disponível em: https://material.io/design/. [Acedido: 15-Ago-2018].

[91] «What is Material Design?», *The Interaction Design Foundation*. [Em linha]. Disponível em: https://www.interaction-design.org/literature/topics/material-design. [Acedido: 23-Out-2018].
[92] «Trello». [Em linha]. Disponível em: https://trello.com/. [Acedido: 23-Out-2018].
[93] «Jira | Issue & Project Tracking Software | Atlassian». [Em linha]. Disponível em: https://www.atlassian.com/software/jira. [Acedido: 23-Out-2018].
[94] «JIRA vs Trello - Our Review on Project Management Tools», *Usersnap Blog*, 04-Ago-2016. .
[95] M. Radwan, «Types of Meetings in Scrum and Agile | Automation Planet». .
[96] «Figma: the collaborative interface design tool.», *Figma*. [Em linha]. Disponível em: https://www.figma.com/. [Acedido: 24-Set-2018].
[97] M. Pacifico, «Why We Should Be Using Figma», *Prototypr*, 06-Abr-2018. [Em linha]. Disponível em: https://blog.prototypr.io/why-we-should-be-using-figma-1510d0923be. [Acedido: 24-Set-2018].
[98] «Introduction», *Material Design*. [Em linha]. Disponível em: https://material.io/design/introduction/#principles. [Acedido: 20-Ago-2018].
[99] «What IS Flexbox?», *Space Ninja*, 24-Ago-2015. [Em linha]. Disponível em: https://spaceninja.com/2015/08/24/what-is-flexbox/. [Acedido: 21-Ago-2018].
[100] «Modelo de dados do Cloud Firestore», *Firebase*. [Em linha]. Disponível em: https://firebase.google.com/docs/firestore/data-model?hl=pt-br. [Acedido: 09-Out-2018].
[101] «10 Key Principles of Agile Software Development». [Em linha]. Disponível em: https://project-management.com/10-key-principles-of-agile-software-development/. [Acedido: 24-Out-2018].