BABES-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

HYBRID MOBILE APPLICATIONS IN THE CONTEXT OF ART, PHOTOGRAPHY AND COMMUNICATION

Scientific supervisor
Lect. Ph.D. Radu D. Găceanu

Author
Ungur Nicoleta

2020

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ENGLEZĂ

LUCRARE DE LICENȚĂ

Aplicații mobile hibride în contextul artei, fotografiei și al comunicării

Conducător stiințific
Lect. Ph.D. Radu D. Găceanu

Absolvent
Ungur Nicoleta

2020

Abstract

This thesis introduces the study of *Hybrid mobile applications* that help developers to create cross-platform applications in an efficient way and in a short amount of time. The application developed for this research is called Photograma and the main purpose of it is to easily put photographers and people in contact when their contribution is needed.

Photograma application is developed in React Native and Ruby on Rails and the decision to use these technologies was taken after a careful analysis, considering the cost and the time, but also the mobile devices market and the latest development tools which are used world-wide these days. In terms of application structure/architecture and storage, we rely on the PostgreSQL database and AWS S3 for image storage, which is an external custom made server. Another interesting decision that we made was to integrate Custom Vision (from Azure) to create an image classification model, that was used to predict the photographer's style. To this end, we wanted to emphasize the importance of cloud-storage in the context of a mobile application in 2020, in particular for our purpose of storing images and getting the predictions for the images in real-time, but we discussed this in more details in this paper. We have treated each technology individually in this thesis and we tried to explain why we have chosen each one.

With Photography and Influencers' field growing so much and becoming a powerful economic branch and among the few areas that receive innovative technologies with open arms (in order to promote themselves), we were thinking of a way of structuring in a single application more functionalities needed to satisfy the needs of customers and their different demands. Photograma application aims to improve the process of finding a good photographer in the proximity of the user, matching their interests and photography styles by providing an application that treats elements of each category.

Taking into account that the application is developed for people who are working in art/photography and are very creative, we have considered the design of the application a very important point. Not only than hybrid applications have to be as intuitive, fast, and responsive as a native one, but we have also taken very seriously the problem of visual elements.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Ungur Nicoleta

Contents

List of Figures

List of Tables

Chapter 1

Mobile Applications Systems in Photography, Society and Communication

1.1 Introduction

In our interconnected world and moreover in the context of a new software application, it raises a question ”Which will be the impact of our product and how it will innovate the quality of life for its users?”. This question leads us to a different way of thinking, a deeper analysis of the product and its functionalities.

In this paper, we tried to create a bridge between creative fields and tech world, but also show that an application can be both useful and user-friendly, with an emphasis on UI/UX design. To achieve its goal, the first chapter of this paper ”Mobile Applications Systems in Photography, Society and Communication” presents the importance of Photography in our world and how we started to integrate technology in our day to day lives.

The second chapter called ”Developing Mobile Applications” presents the differences between three categories of mobile applications: Native mobile applications, Web mobile applications and Hybrid mobile applications. From a developer’s point of view, the characteristics of each type of application are important to be considered in terms of time, cost and the needs of the client.

The main point of chapter three, ”Technologies, Storage and Thread Management in the context of the Photograma Application”, is to introduce the technologies used to develop Photograma application. In this chapter we tried to explain the benefits that came along with React Native, Ruby on Rails, PostgreSQL database, but also the importance of using cloud services to store data or to compute various operations. We dive in a bit to the Machine Learning field and tried to make the concept of Image Classification clear and to detail the purpose of classifying images.

The fourth chapter called ”Photograma Application” is dedicated to application functional description. We described how Photograma works, what is the architectural model of it, how we have integrated the tools and what makes our application special.

In the end, we summarised the conclusion, how we understood the importance of mobile applications in 2020 and why we think that Photograma is an original idea (something that was missing from the market) and the future improvements that we planed for our application.

In the last decade, after the internet burst, it became clear that most of the traditional services would be replaced by more accessible online applications. As a consequence, people find themselves in a rush of finding the best application that also matches their requirements. To be user-friendly (easy to use), to be useful, to be a gate for communicating with other people. If we go deeper, the needs of the user depends on its goal. Will it be the application more business oriented or will it help us growing our own business? When we start building an application we have to answer briefly to those questions in order to choose the technologies that will lead us to the final product and will also be time and cost efficient.

If we take a look at what is happening worldwide, smartphones have become a formality, not a luxury. If 30-20 years ago it was a dream to own a telephone or a personal computer, the reality nowadays differs. But if we go only 10 years back, when the smartphones' era has begun, we will realize that people were a bit skeptical to integrate them into their lives. Maybe it was the fear of change, maybe the comfort of what was before then, but we know a thing for sure, it was just a matter of time until everybody has considered them as a revolutionary piece of technology. This thing was also influenced by the high availability and affordability of the new mobile devices. From that moment people started to see smartphones as a necessity and they get used to the idea of carrying a small personal computer in their pocket.

As studies and global surveys show [1] "in 2020, the number of smartphone users in the world is 3.5 Billion, which translates to 45.04% of the world's population owning a smartphone. In total, the number of people that own a smart and feature phone is 4.78 Billion, making up 61.51% of the world's population." and another study [3] shows that most of the internet traffic comes from mobile devices.

In the following sections of this paper, we will present the differences between mobile applications, how we can classify them, how they can influence the user or the development process and also we will catch a glimpse over the different approaches of developing a mobile application: the business perspective and the user perspective when dealing with particular problems.

1.2 Photography and Communication in Society

In a society that encourage free speech and promote free movement, art and photography have become a way of expressing ones feelings and emotions. For usual people, photography is a useful tool in their day to day life, but for artists, photographers and nowadays influencers, photography is seen more like a business, than just an instrument. In order to satisfy everybody's' needs, mobile applications like Facebook and Instagram have become extremely popular.

When it comes to art and photography, it is said that a photo is equivalent to a thousand words. We keep them in our pockets, in our smartphones, we post them on social media and now, with the technology that is accessible to almost anyone all around the globe, we capture memories easier than ever. We share photos with our friends, family, and colleagues and they have become a part of our day to day life. If we see it as the "method of recording the image of an object through the action of light, or related radiation, on a light-sensitive material"[2] we can understand just a part of the process, the theoretical one. But the entire process of taking good photography is much more than that.

From the object of making, which is the camera, until the actual product, which is photography, there is a dense process which is for a lot of people pure art. If we look at the basic definition of the camera "is a light-tight container carrying a lens, a shutter, a diaphragm, a device for holding (and changing) the film in the correct image plane, and a viewfinder to allow the camera to be aimed at the desired scene. The lens projects an inverted image of the scene in front of the camera onto the film in the image plane. The image is sharp only if the film is located at a specific distance behind the lens."[2] we will recognize that behind that object that we are so used to, it's an entire science.

Following a recent study [4] about the most popular social networking mobile applications in the United States from September 2019 shows us that the first most accessed application is Facebook, with 169.76 million mobile users accessing monthly the application, followed by Instagram, with 121.23 million users. Other applications from that study along with the first ones are Twitter, Pinterest, Snapchat and WhatsApp, but there are also some that are not so popular as the ones stated before, but they are used and known in this field. We will call some of them: Vero Social, Youpic, Exposure.co, Flickr and others. We can see that all of them share 2 particular things: visual images and free communication, because the contact information for each user is available to anyone. Perhaps we can conclude that the last can be a short description of the 21st century. Our social media accounts have become our psychological profiles.

1.2.1 Transition to mobile from web applications

The need of people to share their private lives was exploited by big companies. Because humans are social creatures, it is proved that we tend to suffer from social anxiety from time to time, but in 21st century this feeling looks like it is amplified. We use photography as a way to express life itself and to capture moments, and later to share our happiness with people. We don't have to care about war or food anymore, like the people that lived no more that 100 years ago and this is one of the main reasons why technology and the used of it became so popular.

The transition from web applications to mobile applications is strongly related to the apparition of smartphones, of WiFi infrastructure and mobile data from providers. It was an entire machinery that was involved, and a lot of people who were working to let us use our favorite applications on our devices. That transition was beginning in the late 2008-2009, even if it was overlapping with a global crisis. The companies probably saw that crisis as an opportunity to change something, and they started to develop technologies and frameworks for mobile app development.

Only in 2014 and 2015 the big giants were coming out to some initial stable version of their products. Swift (Apple) was the first one released, on June 2, 2014 (5 years ago) followed by React Native (Facebook) and in 2016 by Kotlin (Jet-Brains) and Flutter (Google) in May 2017. These four ones are the biggest competitors on app development market at the moment.

1.2.2 Conclusion

Looking at the situation that is happening right now in a society that was promoting continuously free movement, border-less borders and flights from one end of the globe to the other, it is true that something is going to change, but we also have seen that the people were still connected during this pandemic due to technology, especially applications (mobile/web) with media content.

When evaluating the weight of current mobile applications, we note that every desktop application is now available also as a mobile application or a web application (in case it is not listed in any app store, but is responsive). Now, more than ever before, people buy smartphones for several reasons, other than just for their initial purpose (being a cell phone). With schools having online lessons and content, families that can not afford to buy a laptop or a personal computer for each member of the family, have found mobile phones market a good option. From the user point of view, smartphones are affordable and they are a fast alternative. They were also one of the way for communicating with other people, sending photos and checking with our friends. The below illustration express the current situation and how technology has become a bridge for our social lives.

The market and the economy in 2020 was for sure affected on all levels by the pandemic, and the mobile phone market made no exception. If we were used to have a new apparition from a big tech company every two-three months, the situation in 2020 was a bit different even tho the technology was used this year at its full capacity. In this uncertain situation, we can only wait to see how will fluctuate the tech industry.

(a) Current situation 2020

Figure 1.1: Current situation 2020

Chapter 2

Developing Mobile Applications

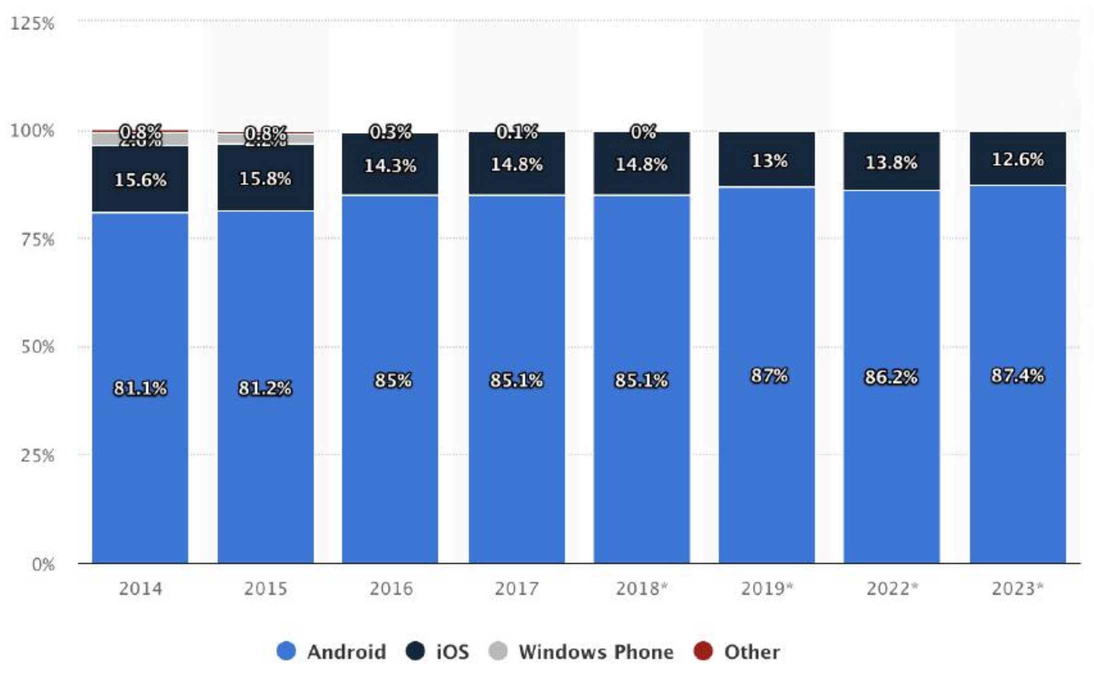
2.1 Mobile Applications

In this section, we will discuss about different types of mobile applications, how they differ from one another (native and hybrid applications), we will make a comparison between them and web applications, and how our lives were affected/changed by mobile application development. More specifically, we'll be focusing on mobile applications for the two largest mobile operating systems: iOS and Android.

2.1.1 Context and definition

First, we have to integrate mobile apps in their context, because we can not refer them without talking about mobile phones. As Cambridge Dictionary defines it, a mobile phone is "a phone that is connected to the phone system by radio instead of by a wire, and can be used anywhere its signals can be received"[5]. This is the main functionality of it, but as technology started to advance from the beginning of the 21st century, so does the development of mobile phones. 1992 was additionally the year that IBM divulged the absolute first cell phone. Quick forward two years and the gadget authoritatively hit the market at a cost of 1,100 US dollars. What's more, after just a half year, it had sold more than 50,000 units.[7]. It was clear that the world was open to this type of technology.

From that moment, all the big players, who practically dictated new trends in phones at that time (Nokia, Motorola, BlackBerry, Palm and Samsung) have started to find some ways of improving that model and so the true metamorphosis of smartphones happened in early 2000 with the appearance of the first BlackBerry phones that had QWERTY external keyboards and made it feasible to get to web pages, send emails and write messages instantly. But 2 of the most important events from this list of "firsts" were the apparition of the iPhone and Android phones. In 2006, when the launch of Apple's first iPhone was announced, the Microsoft CEO from then, Steve Ballmer, said that "There's no chance that the iPhone is going to get any significant market share"[6], but things took a dramatic turn in 2007 when the apparition of Steve Jobs' brand new phone was shaking all mobile rankings and rapidly winning the hearts of people. Apple came with a minimalist and innovative design, a large touchscreen with multi-touch gestures and on-screen QWERTY keyboard and the option of downloading stand-alone applications made available by the online distribution network App Store.



(a) Global evolution - statistics

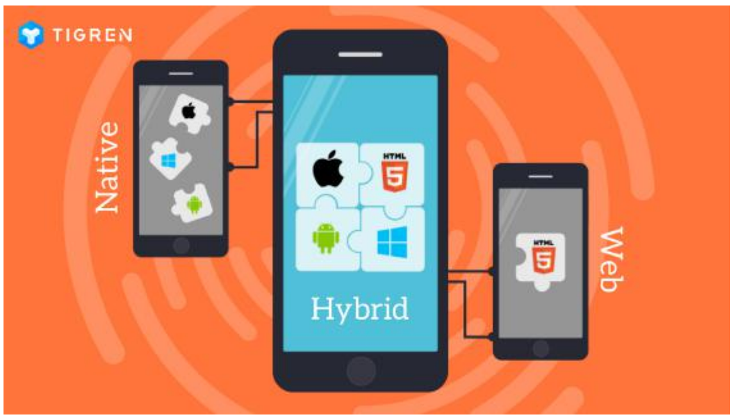
Figure 2.1: Global evolution
Source: `https://www.statista.com`

A year later, in September 2008, Google launched the first commercial Android device which became the number one competitor of the iOS operating system and future most used mobile operating system worldwide (including now). Apple and Android smartphones popularized the world of devices based on a large capacitive touchscreen and lead to the demise of systems focused on the earlier, mouse, and keypads. Fast forward almost 10 years from then, we are all used to own a smartphone, because it behaves like a PC, but we can carry it in our pocket and we can access information from the internet nearly instant(depending on the connectivity and the internet speed). More innovative designs come from the major companies to extend the device as far as possible, shorten the edges and create a full-screen interface. The most impressive one was the notch (the upper portion of the panel that contains the microphone and the front camera), and then, the designers were trying to remove it by adding the pop-up camera or fully slipping mobile. Smartphones often come with enormous batteries up to 5000mAh that tend to sustain the user for up to 15 hours of intense usage as well as quick charging methods. If the legacy has been marked by a number of controls, rigid, tiny displays, antennas, and pixel cameras, the present smartphones are reliable, elegant, compact, durable, and with almost fluid displays.

Let's understand now what is a mobile application from its definition: it is "a software program that runs on a mobile phone"[5]. This is the simplest, but the easiest definition in order to be understood by everybody. It is true that nowadays mobile applications are much more complex than the applications that run more than 10 years ago, but the definition holds for every generation.

2.1.2 The big picture

Applications are commonly little, singular programming units with constrained capacities, for example, a game, calculator, or web browser. In spite of the fact that applications may have abstained from performing various tasks in light of the restricted equipment assets of the early cell phones, their explicitness is presently part of their attractive quality since they permit customers to hand-pick what their gadgets can do.



(a) Types of mobile apps

Figure 2.3: Mobile applications
Source: `https://www.tigren.com/`

As indicated by a study made by eMarketer [8] look into, clients go through two hours and 11 minutes out of every day utilizing versatile applications, yet only 26 minutes perusing the web on a cell phone. Application time made up about 20 percent of all-out media time in 2017 and without precedent, the 2019/2020 individuals invest more energy with their cell phones than they do sitting in front of the TV. Cell phones represent 70 percent of that versatile time and that rate continues developing. Applications are setting down deep roots and, as an ever-increasing number of organizations walk out on the conventional portable web, purchasers' inclination for applications is just going to increment. With those premises, the developers had to find solutions to that demand and they started to develop new technologies and a lot of applications.

For an unprecedented number of nearly 2.87 billion mobile phone users in 2020, the smartphone applications sector is of interest to both consumers and suppliers as well. This market is fast-growing, with companies developing up to three new telephone models a year. For this reason, we are experiencing rapid breakthroughs in technology that transform the way we communicate with the phones. We moved from phones with basic functionalities into true computing beasts able to cope with laptops or PCs in 10 years' time.

In this paper, we will focus more on developing Hybrid applications, but we have to present also the other two types of mobile applications in order to understand the advantages and disadvantages of each of them. We will start in a logical manner and in chronological order, from the web apps, to native based apps and to the newest type, hybrid apps. We will also try to draw a parallel between the three types and explain the main features that developers look at when choosing which technology to use for their future applications based on their needs.

2.2 Web Applications

Mobile Web applications are similar to what we already on our PC via a normal browser. They are web sites developed utilizing common technologies such as HTML5(when HTML5 came around, people realised that they can obtain native-like functionality in the browser through web apps), CSS and JavaScript, but increasingly intuitive, instinctive, designed with a compact first way of coping with and working excellent on mobile phones(responsiveness), but are not written specifically for the device. Web apps are viewed through the internet browser and can adjust to whatever screen you display it on. They are not native to a specific device, and therefore can not be updated or installed. They do also look and work a lot like mobile apps because of their responsive design.

Building a web applications deliver many business benefits because they are easy to maintain (they have a common code-base regardless of mobile platform), will update themselves automatically (no need to update the app from App Store or Google Play) and the user doesn't need to download it and they are quicker and easier to build than mobile apps. Also something very important: web applications do not require app store approval, so can be launched instantly, but this can be seen also as a bad thing, because quality and security is not always guaranteed.

This leads us to some of the disadvantages of building web applications (for the business and also for the users). Because they are not listed in an app store they may not be as discoverable as mobile apps. Web apps are slower, as run over the internet (and moreover, they have to run on different browser and different versions of the browsers) and the internet connection is always 100% available.

2.3 Native Mobile Applications

The most growing form of app is ***Native Mobile Applications***. These are developed for different networks, and are written in languages approved by the platform. For example, for native iOS apps Swift and Objective-C, and for native Android apps, Java or Kotlin. Native apps are often designed to the chosen operating systems utilizing the unique Integrated Development Framework (IDE). Both Apple and Google provide app developers with their own development tools, interface elements, and SDK. Most companies will invest in native mobile app development because of the multitude of benefits offered in comparison to other types of apps.

Let's understand whisch are the benefits of using such an application. Native apps deliver the best performance of all three development approaches, receive complete support from app stores and the overall app marketplace (Distribution in app stores helps with discoverability), interactive, intuitive, and run more smoothly in terms of user input and output. They allows developers to access the full feature set of the selected operating system and the user experience of native apps is far superior to web apps or hybrid apps. To the user, the flow is more natural because of each mobile operating system's specific UI guidelines and standards. [9]

There are also some drawbacks because native apps are using challenging programming languages which often require experienced developers, and expenses are more costly for native apps than those for web or hybrid apps. For sure, they are not one of the best options when talking about a simple, cost efficient mobile application.

With native mobile applications, you ought to build different applications for different operating systems, however, developers have new programming technologies that may sometimes be classified as other types of native app development (which is just half true, because they are used to build hybrid mobile applications - we will explain this concept of hybrid apps in the following section) and those technologies help developers building cross-platform applications.

2.4 Hybrid Mobile Applications

Hybrid, by nature, is something originating from heterogeneous sources, or consisting of various or incongruous types of materials. A hybrid mobile application is an application that is written using the same technology used for websites and mobile web implementations and is installed in a native container utilizing a WebView Mobile model. This device shows online content while the app is used due to the usage of modern technology (CSS, JavaScript, HTML5). It also presents web pages from a mobile website which are customized to a WebView interface. The site material may either be viewed as soon as the device is accessed. Hybrid app development is essentially a web app that incorporates additional native features. Including native features is possible when you deploy a wrapper to act as a bridge between platforms.A hybrid app consists of two parts:

1) The backend code. Hybrid code is written in languages like HTML, CSS, or JavaScript.
2) A native shell which is downloadable and loads the code using a webview.

The advantages that came along with those new technologies and this new paradigm of developing mobile applications have been seen as hybrid applications have started to be used by as many companies as possible. Among them we mention that hybrid apps don't need a web browser like web apps and have access to a device's internal APIs and device hardware.

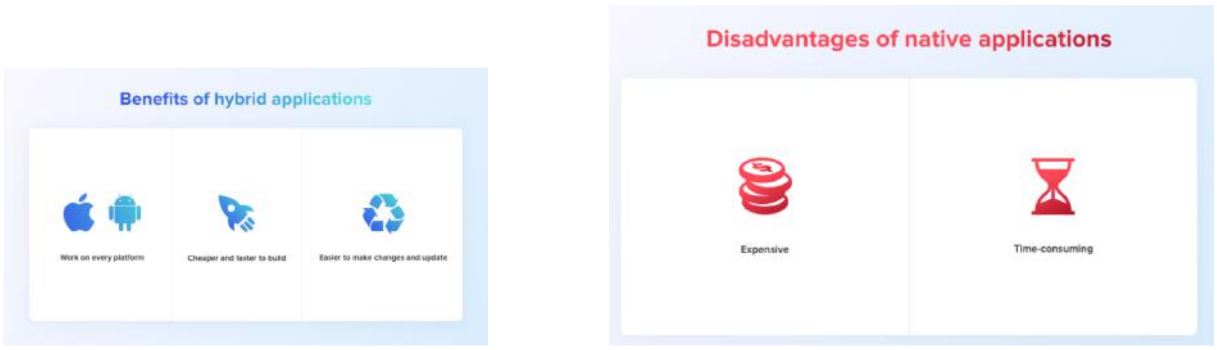

Benefits of hybrid applications
Work on every platform
Cheaper and faster to build
Easier to make changes and update
Disadvantages of native applications
Expensive
Time-consuming

Figure 2.6: Advantages/Disadvantages
Source: `https://www.cleveroad.com/`

From the user experience, it have been seen that mobile applications are commonly subject to limitations due to lack of offline support, one of the benefits of the hybrid App, the offline accessibility feature helps to overcome this challenge. As a result, end-users have uninterrupted access to the app's data without performance glitches. Statistics show that 70% of users are leaving the app because it takes too long to load.From the business perspective, the main advantage of the hybrid applications is that the entire process is easier and quicker to develop than the native apps. It enables businesses to leverage their existing talent pool for software development to lead the mobile market. The development team requires knowledge of JavaScript and HTML. But once the code is written, the application can be run on both operating systems (iOS and Android).

Hybrid App Development provides accurate and flawless user experience (which is one of the most important part when it comes to developing a new app) along iOS, Android platforms. Hybrid applications are about making it work for mobile devices by embracing the web. In return, the lightweight hybrid UI app helps quickly load content and graphics. Apps adapt quicker to different platform screens for faster display of information and seamless streaming of data. The excellent UI experience even improves the chances of the application being approved in the app store.

For sure, there are some disadvantages also for using those new technologies for building hybrid applications and the developers have to consider them when they start building a new application and choosing the stack of it. Due to the enormous complexity of the architecture of hybrid mobile apps they rely on plugins to access a device's built-in features. The downside of such a system is that some plugins can be obsolete or ineffective. In addition, developers can need to build plugins on their own if there are no ready-made solutions that allow you to access a certain part of the functionality of the device you need. On the other hand, the low performance of hybrid applications is the highest disadvantage of them, precisely because it runs on Web View. So, at this moment is a problem that is trying to be solved by growth the performance of the Web View component of our mobile devices. However, the performance of native applications has not yet been reached. It also put more effort into developing a hybrid application that is just as intuitive and interactive as a native one and to provide the user with the best the experience.

Conclusive, as long as our implementation satisfies the user's expectations, suits the required budget and resources, and is a friendly and usable interface for the users, the technology behind it is only one aspect that remains at developers' disposal. Mobile applications are selected based on company needs and consumer preferences. While web-based and native apps have several benefits, hybrid mobile apps have gained popularity. It's because of their adaptability across different platforms. This principle comes down to the creation of a single application that operates seamlessly across multiple platforms.

Chapter 3

Technologies, Storage and Thread Management in the context of the Photograma Application

In the following chapter, we will describe each technology used to develop the mobile application and we will show which are the benefits that come along with them. We will include information about Ruby programming language (in particular Rails framework), why we chose to use PostgreSql for our database management system and why can it be considered one of the best open-source DBMS, but we will present the technology that helped us to use the hybrid mobile app, React Native, which is framework developed by Facebook.In the section "Redux and state management" we will talk in more details about the state management of an application and what options have the developers nowadays to handle it.

3.1 Ruby on Rails, PostgreSql and Heroku

Ruby is a dynamic programming language with a complex but expressive grammar and a core class library with a rich and powerful API. Ruby draws inspiration from Lisp, Smalltalk, and Perl, but uses a grammar that is easy for C and JavaTM programmers to learn. Ruby is a pure object-oriented language, but it is also suitable for procedural and functional programming styles. It includes powerful meta-programming capabilities and can be used to create domain-specific languages or DSLs.[**?**]

Ruby on Rails (or just "Rails" for short) is a free and open-source web development framework written in the Ruby programming language. In its infancy, Ruby on Rails quickly became one of the most popular tools for building dynamic web applications. Rails is used by companies as diverse as Airbnb, SoundCloud, Disney, Hulu, GitHub and Shopify, as well as countless freelancers, independent development stores and startups.[14]

In 1995, Yukihiro Matsumoto(from Japan) who is best known as Matz, developed Ruby as an object-oriented programming from programming languages, such as Perl and Lisp, thus putting great emphasis on "trying to make Ruby natural, not just simple".A major philosophical underpinning of Rails that keeps code short and read-able is the ***DRY*** principle, which stands for ***Don't Repeat Yourself***. Basically, the idea was to create a programming language which was designated for programmers(humans) not just for computers. Interpreted as Perl and Python and object-oriented as Java and Ada, Ruby manages to create a perfect balance between performance and simplicity and those were the facts that were revolutionizing the tech industry at that moment. In 2003, another developer called David Heinemeier Hansson, created 'Ruby on Rails' (RoR) which is a rapid development framework of Ruby. The officially released was in July 2004 as an open source project. What makes Rails very unique is the fact that with only a few lines of codes you can add a lot of language. This pragmatism is one of the most important elements that support its popularity. Editing can be implemented easily without damaging the core and avoiding the usual time consumption associated with the standard process of web development growth.

Ruby on Rails has an active and lively culture. Rails has three fundamental concepts that have been followed given the rapid evolution of the base Rails code. Here are some of the characteristics that set Rails apart from other frameworks:

- Convention Over Configuration
- Metaprogramming
- Three Default Environments
- Active Record Framework
- Scaffolding

Another interesting and useful thing that comes out of the box in RoR are the gems. Let's understand what gems are and how they are used inside a Rails project. Each gem has a name, version, and platform (for example, the rake gem has a 0.8.7 version, Rake's platform is Ruby, which means it works on any platform Ruby runs on). Platforms are based on the CPU architecture, operating system type and sometimes the operating system version. Examples include "x86-mingw32" or "java". The platform indicates the gem only works with a ruby built for the same platform. RubyGems will automatically download the correct version for your platform. Inside gems are the following components: code (including tests and supporting utilities), documentation, gemspec. Each gem follows the same standard structure of code organization. [17]

3.1.1 Storage

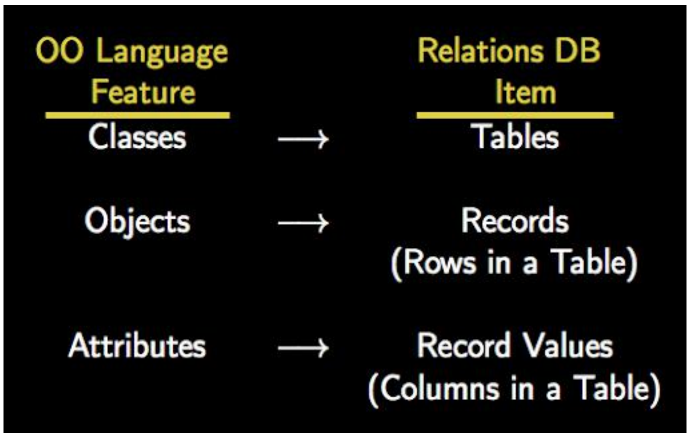
An amazing benefit that comes with rails is the integration with a lot of databases. The system has an excellent Object Relational Mapping, called ActiveRecord, which enables developers to access

Ruby on Rails can be integrated seamlessly with Database Management Systems such as PostgreSQL. To create a Rails app configured for Postgres, run this command:

```
        rails new myapp --database=postgresql
```

This generates a directory named ”myapp” that houses an app called ”myapp” (you can name it whatever you want when the above command is run). Rails requires the user’s credentials for the connection, but if needed, those can definitely be changed. Rails use the database.yml file to link to the respective database for the current Rails system, using YAML, a format for data serialization. In different environments there are a few databases listed here; development, test, and production. By default Rails would allow each environment to have a different database. This is useful, since the test database is drained and rebuilt every time you run Rails tests, for example. Make sure the username and password match the username and password that you gave your Postgres user to for every account.

Object Relational Mapping, commonly known as its abbreviation ORM, is a technique that links an application’s rich objects to tables in a relational database management system. Use ORM, the properties and relationships of the objects in an application can be conveniently stored and retrieved from a database without writing explicitly SQL statements and with less general access code to the database.



OO Language Feature		Relations DB Item
Classes	⟶	Tables
Objects	⟶	Records (Rows in a Table)
Attributes	⟶	Record Values (Columns in a Table)

Figure 3.1: Rails Active Record
Source: `https://www.slideshare.net/`

It general, it might be necessary to write plenty of configuration code when writing applications using other programming languages or frameworks. In general this is particularly valid also for ORM frameworks. Nevertheless, if you obey the conventions adopted by Rails, when constructing Active Record models, you'll need to write very little configuration (in some cases no configuration at all). The theory is that this will be the default way if you configure the programs in the same way most of the time. Therefore, specific configuration will only be necessary in those cases where the standard convention is not comprehensible. It is very easy to create Active Record models. Running the commands from below, a model User will be created and it will be mapped to a database table of users. By doing so, you will also be able to map the columns of each row in that table with the attributes of your model's instances.

```
        rails g model User email:string password:string
        rake db:migrate
```

3.1.2 Provided Authentication

In mobile apps, the authentication process is important as it gives us knowledge about the identity of the user communicating with the application. First of all, we're referring to security and data privacy when we're talking about authentication, but also we do have to look at the customized interface the app has to offer to the signed user. These days, mobile applications usually concentrate on the user, connecting the data to user profiles.

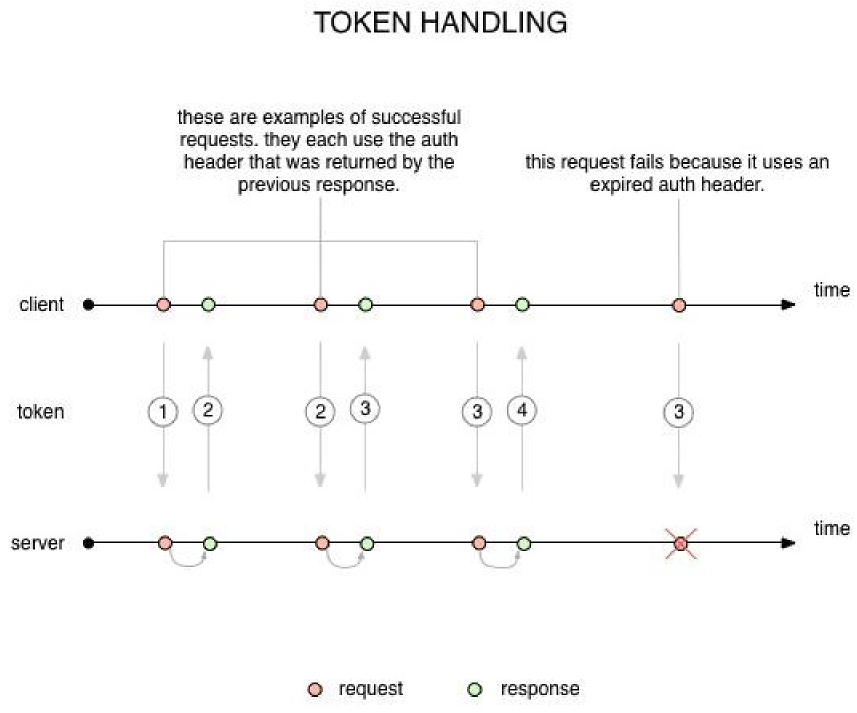

TOKEN HANDLING
these are examples of successful requests. they each use the auth header that was returned by the previous response.
this request fails because it uses an expired auth header.
client
time
token
1
2
2
3
3
4
3
server
time
request
response

Figure 3.2: Authentication + Authorisation
Source: `devise-token-auth.gitbook.io`

For this project, the authentication and authorisation part was implemented using 2 ruby gems: devise and devise-token-auth. The first one is used for the admin authentication and the second one is used for user authentication on mobile devices(the mobile application). The last one is a gem that provide an easy, multi-client, and stable token-based Rails authentication. When building mobile applications that need authentication, developers have to use tokens, not cookies (which can be used on a web browser). On each request, this gem refreshes the tokens, and expires them in a short time, so the system is protected and secure. It also holds a session for every client / device, so that you can have as many sessions as you want.

The devise-token-auth gem comes also with an out of the box implementation for OmniAuth authentication (OmniAuth provides auth using GitHub, Facebook, Google, Apple and others) and a compatibility with anothe gem (rake-cors) which allows the access to the API routes from mobile devices. We will present shortly what is CORS and which is its usage.

Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell browsers to give an application(web or mobile) running at one origin, access to selected resources from a different origin. An application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, or port) from its own.[20]

3.1.3 Deployment - Heroku

Heroku is a cloud platform that lets companies build, deliver, monitor and scale apps — it is a way to go from idea to URL, bypassing all those infrastructure headaches.[18] Heroku helps developers to make deployment, configuration, scaling, and application management workflows as simple and effective as possible, so developers can concentrate on what is most essential: building great applications that delight and engage customers. Deploying and managing software should be friction-free, and these skills should be part of the DNA of a company. Heroku relies on Git, a distributed source control management tool, for deploying projects. After the git and Heroku are configured (commands and articles related to the configuration can be found on their official site [18]) the developer only have to run the following commands:

```
        git add .
        git commit -m "Deploy first app on Heroku"
        git push heroku master
```

The information from the above sections was related to server (back-end) technologies and what happens behind the scenes. In the next sections we will talk about front-end technologies. Front-end technology deals with setting up the UI elements along with the functionality of an application. Everything that can be seen when navigating around the app, from fonts and colors to drop down menus, sliders and so on.

The first version of Photograma was deployed to Heroku, but as we started to develop more, we faced the payment problem, so for the purpose of this paper we choose to keep the server running only on our local machine. We kept this section in this paper, because we wanted to show how easy is to deploy a Rails application on Heroku.

3.2 React Native and Expo

As of 2019, react is one of the most successful JavaScript framework, developed and maintained by Facebook. When it was first launched it took web development by storm and its popularity has since increased among the developer community. When first launched in 2015, React Native took this one step further and helped create Native iOS apps with common knowledge of web technologies. React Native has become a major player in mobile growth in only a few years, and increasing its support for both Apple's iOS and Google's Android has been a game changer.

In 2018, ***React Native***[11] had the second highest number of contributors for any repository in GitHub. Today, React Native is supported by contributions from individuals and companies around the world including Callstack, Expo, Infinite Red, Microsoft, and Software Mansion.[21]

In this paper, we will focus more on developing React Native applications with Expo. Why is Expo so famous and what differentiates it from other competitors?

Expo is a suite of tools, libraries, and resources that can be used to develop iOS and Android apps more quickly than ever.

The ***Expo SDK*** is a snippet of natively written libraries for each platform that provides JavaScript access to the system functionality of the computer (things like camera, push notifications, contacts, local storage, and other hardware and operating system APIs). The SDK is designed to flatten out platform gaps as much as possible, which makes your project extremely versatile because it can run in any native environment that contains the Expo SDK. The only development tools needed are Expo CLI, the latest version of NodeJs and an expo client app (it can be downloaded from the play store for Android and from the app store for iPhone). Running the following commands will set the environment and will create and will start the first expo project :

```
        npm install -g expo-cli
        expo init app_name
        cd app_name
        expo start
```

After successfully following the above commands, a QR code will be generated and that can be scanned from the Expo App in android and from the Camera App in the iOS.

After this step, the application will be up and running on the users' devices. For more information and snippets of code, check their official website. [22]

3.2.1 Redux and State management

Because in modern applications the idea of sharing information through the entire application became pretty common, the developers faced some problems implementing the solution for this need. React came with some in place solutions like using Props and AsyncStorage, but those solutions were causing developers headaches because they had to take care of every Component and its Props. Much more, managing state when building complex tasks, means in a plain application using only react and Props, to send the props from parent to child hierarchically and to send the callbacks the same way. Communication between two components that do not have a parent-child relationship is discouraged in React (and other frameworks too). The Facebook developers are recommending that the global event system should be designed according to Flux pattern — and that's where Redux comes in.

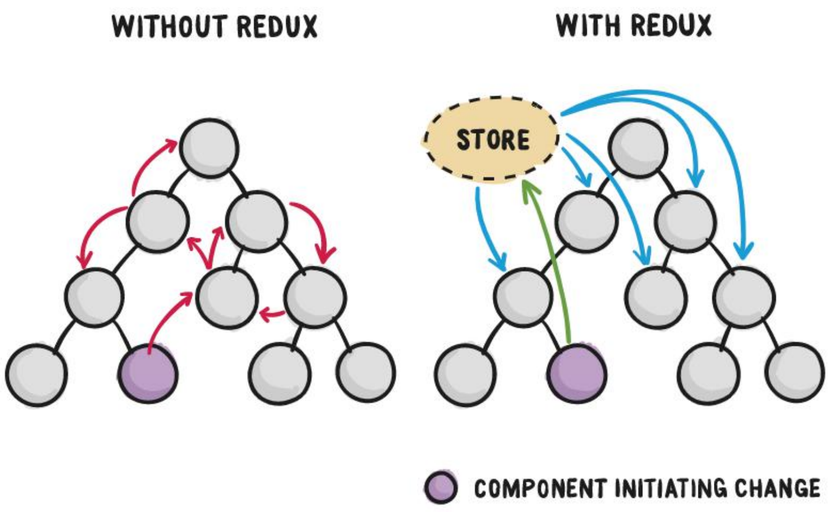


Figure 3.3: Component hierarchy in React Native
Source: `https://css-tricks.com`

3.2.2 What is Redux? What are the benefits of using it?

Redux is a generic JavaScript state container and comes as an additional library that can be used in parallel with front-end frameworks such as React, Angular, Ember.js, Backbone etc. Redux is most commonly used in combination with React though. Redux comes at the table with several benefits that are commonly discussed:

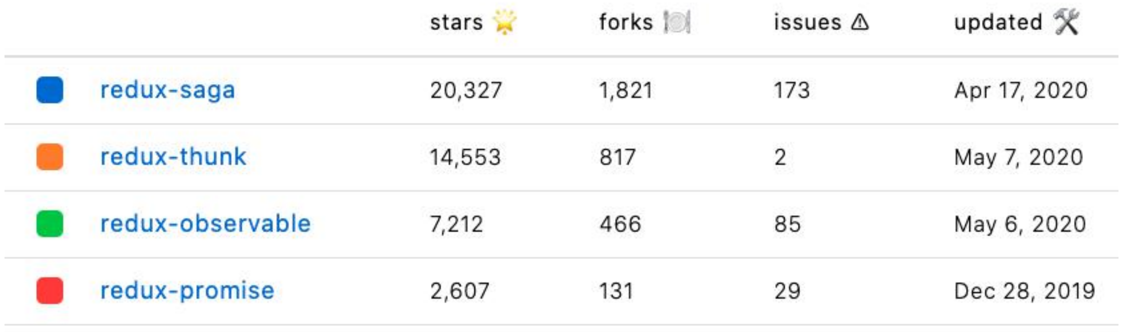
- Predictable state updates - easier to understand how the data flow works
- The use of ”pure” reducer functions - making reasoning more easy to check
- Centralised state - allows things like logging data changes or persistent data between page refreshes easier to implement

3.2.3 How Redux works?

Redux guarantees that each and every component of the application has full access to the application’s state without having to send props down to child components or use callback functions to send back data to a parent.

The store can be thought of as a ”mediator” in the application for all state changes. Involved by Redux, components do not interact directly amongst themselves, but instead all changes in state must go through the single source in fact, the store. It’s obvious, with Redux, that all components are receiving their state from the store. It’s also clear where components will submit changes to their state — the store, too. The component that initiates the change is only about dispatching the change to the store and doesn’t have to care about a list of other components that need the change of state. This is how Redux makes it easier to think about the data flow. The only problem appears along with asynchronous tasks. Redux manages the application state synchronously. The solution to this is to integrate a middleware. A middleware provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.

Some of the most popular middlewares with redux side effects are redux-saga, redux-thunk, redux-observables, and redux-promise. If we look through npm trends and compare all of the widespread middleware redux side effects, we can see that redux-saga has the highest number of stars and forks and that it is highly maintained. This will be the middleware that we will talk about in this paper. In the next section, we will walk you through the benefits of using redux-saga.



		stars	forks	issues	updated
🟦	redux-saga	20,327	1,821	173	Apr 17, 2020
🟧	redux-thunk	14,553	817	2	May 7, 2020
🟩	redux-observable	7,212	466	85	May 6, 2020
🟥	redux-promise	2,607	131	29	Dec 28, 2019

Figure 3.4: Redux popular middlewares
Source: `https://www.npmtrends.com/`

3.2.4 Redux-Saga

Redux-saga is a library that aims to make application side effects (i.e. asynchronous things like data fetching and impure things like accessing the browser cache) easier to manage, more efficient to execute, easy to test, and better at handling failures.[23]

The theoretical model is that, in the application, a saga is like a different thread that is directly responsible for the side effects. Redx-saga is a redux middleware which implies how a thread can be started, paused and terminated with normal redux actions from the main application, it also has access to the entire redux app state and can also dispatch redux actions. The most important thing that differentiates redux-saga it's the fact that it uses generators to make asynchronous flows easy to read, write, and test.

Installing this middleware into an application is straight forward. Running the following command in the terminal in the project structure will set it.

```
        yarn add redux-saga
```

Listing 3.1: Login Method implemented with Redux-Saga

```
export function* login({email, password}) {
    try {
        yield put(UserActions.loginLoading(true));
        const response = yield call(axios.post, '/v1/auth/sign_in.json',
            {email, password});
        if (response.status === 200) {
            const uid = response.headers['uid'];
            const client = response.headers['client'];
            const accessToken = response.headers['access-token'];
            const expiry = response.headers['expiry'];
            const avatar = response.data.data.avatar.url;
            const name = response.data.data.name;
            const username = response.data.data.username;
            const id = response.data.data.id;
            const phone = response.data.data.phone;
            console.log(response.data);
            yield put(UserActions.loginSuccess({
                ...response.data
            }));
        }
        yield put(UserActions.loginLoading(false));
    } catch (error) {
        yield put(UserActions.loginError('Invalid username or password!'));}}
```

3.2.5 Publishing a React Native application with Expo

When developers create any type of application, they will test if it is working on their local machines in the beginning. That was exactly the process that was followed by me while developing Photograma. The only problem is that when the machine stops, or the bundle is closed, so does the application. Expo CLI and React Native packager are running on developers' machine and bundle up all the source code making it available from a URL. Expo CLI uses exp.direct which is a domain for tunneling, so that any device on the internet having the URL of the project may access it, even if it runs behind a VPN or a firewall. This makes opening a project on a real phone much simpler. The idea remains the same as the above, when the machine stops, the application will no longer be available from that URL.

Expo makes the entire process of publishing the application to be straight forward, just a command away.

```
    expo publish
```

No other setup is required, so if the developer just created an expo project that is running, he can run this command and it will be published/deployed. If he follows the above steps, the packager will minify the whole code and will generate two versions of it (one for iOS and one for Android). Expo will provide a link that will allow anyone to access the project

Deploying to the App Store and Play Store When it comes to sharing the application with the whole world, then things get complicated. The programmers have to be sure that the code is running properly on multiple devices. Even if a hybrid application by definition is an application built to be cross-platform, there are also some drawbacks and a developer has to be aware of them (there will be cases when the native code will be written separately for each platform).

Because deploying an application is not free (at least not for deploying the iOS app), in this paper we will not focus that much on the entire process, even if it is a rigorous one. All the information is explained in detail in expo's documentation.[22]

3.3 AWS S3 and Cloud Services

First of all, I would like to explain what Amazon Web Services are, in particular AWS S3 that was integrated into this project. ***Amazon Web Services (AWS)*** provides over 175 completely integrated applications from data centers, rendering it the world's most robust and broadly accepted cloud service. Millions of users , from startups to large companies and leading organisations. They use AWS to cut costs, be agile and innovate more rapidly. [25].

3.3.1 What makes AWS so popular?

In the last years, AWS have provided a large variety of services with a lot more features than any of its competitors or any other cloud provider. Furthermore, they managed to deliver all these services and integrated features in a fast and cost-efficient way. This was the main reason that powered them on top of the pyramid. Some of their services are: infrastructure technologies like compute, storage, and databases–to emerging technologies, such as machine learning and artificial intelligence, data lakes and analytics, and the Internet of Things.

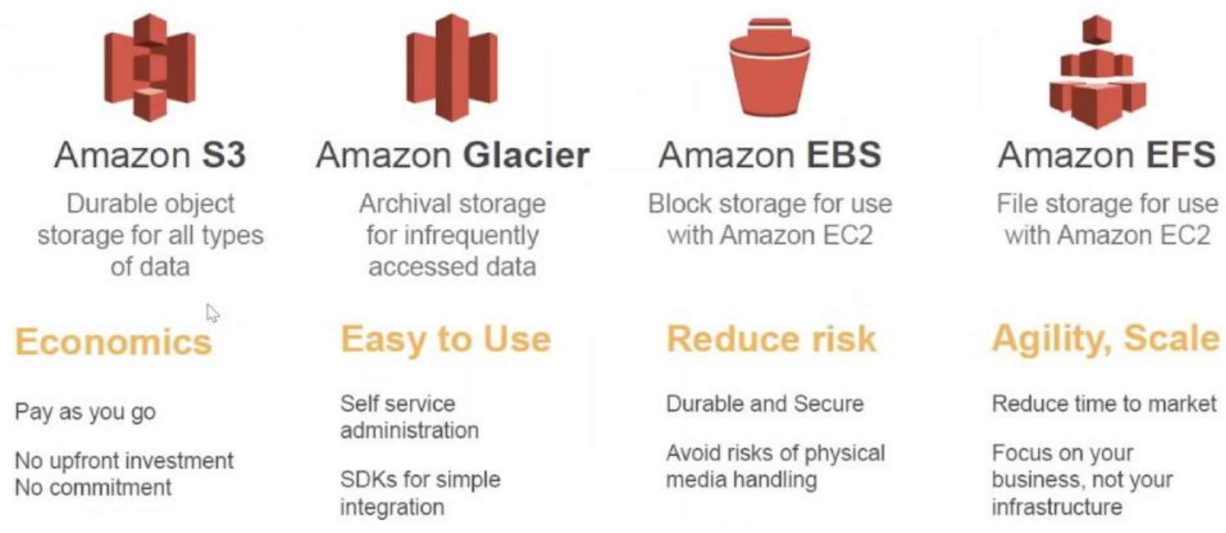

Amazon S3
Durable object storage for all types of data
Amazon Glacier
Archival storage for infrequently accessed data
Amazon EBS
Block storage for use with Amazon EC2
Amazon EFS
File storage for use with Amazon EC2
Economics
Pay as you go
No upfront investment
No commitment
Easy to Use
Self service administration
SDKs for simple integration
Reduce risk
Durable and Secure
Avoid risks of physical media handling
Agility, Scale
Reduce time to market
Focus on your business, not your infrastructure

Figure 3.5: AWS - Storage choices
Source: `https://www.youtube.com/watch?v=AP4JWtzIAt0`

The ***Amazon Simple Storage System (AWS S3)*** is an Amazon Web Service (cloud storage system), which provides strong scalability and efficiency. The customers pay only for storage they are using without minimum fees or initial costs.[24]

Amazon S3 it's a perfect fit for hosting multimedia content and for image and video sharing. They also provide easy ways to integrate this product with a large variety of technologies (in my case Ruby on Rails). I took advantage of the fact that Amazon has been always involved in the educational sector and it has offered free cloud-based storage for students and researchers, thus I have integrated their products into my thesis with no fees.

Amazon S3 stores data as objects in a bucket. The bucket has a folder structure in which each object is a file. The bucket is like a repository of objects and you can have more buckets. In the beginning, in order to have access to those buckets, the user has to create one. When creating a bucket, the user can decide the access that will be given to that bucket (who can execute CRUD operations). After the creation of a bucket, Amazon will provide some secret credentials for it, and with those crypto-credentials the bucket will be accessed outside the AWS (from the application in which it will be integrated).

In my application, I have used S3 alongside carrierwave and fog-aws gems, which provides a simple way to upload files from Rails applications.

```ruby

CarrierWave.configure do |config|
  config.storage = :fog
  config.fog_provider = 'fog/aws'
  config.fog_credentials = {
    provider: 'AWS',
    aws_access_key_id: ENV['AWS_ACCESS_KEY'],
    aws_secret_access_key: ENV['AWS_SECRET_KEY'],
    region: ENV['AWS_REGION']
  }
  config.fog_directory = ENV['AWS_BUCKET']
  config.fog_public = true

end
```

3.4 TensorFlow - image classification

In 2020 image classification is a common utensil in mobile and web applications (in special those who are photography and visual oriented) and a really popular and world wild used tool is TensorFlow. Because we are developing the application in a JavaScript framework, for this project we were using TensorFlow.js, which is a subclass of TensorFlow.[26]

Before diving into understanding what image classification is, we will explain another concept, Machine learning. The first google search will get you to the simple definition of ML, the one that we will use also in this paper(in order to keep it simple and easy to be understood by anyone).

Machine learning is an application of ***artificial intelligence (AI)*** that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.[27]

3.4.1 What is image classification?

A common use of machine learning is to identify what an image represents.The task of predicting what an image represents is called image classification. A model for the image classification is trained to recognize different image classes. For example, a model could be trained to recognize photos of three different types of people: a man, a woman, and a child.[28]

The model to be trained receives images and labels for those images during the training process. Each label has a name/tag that is different depending on the concept that has to be learned.

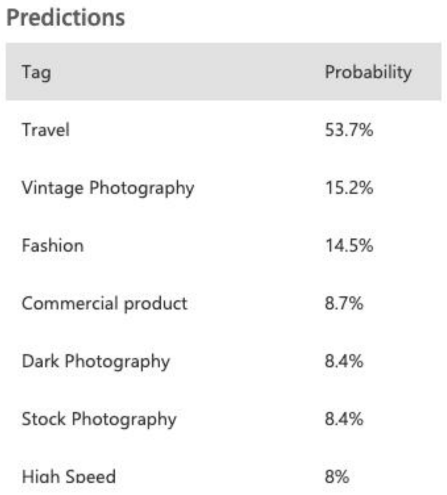
The process of prediction has a good accuracy only if sufficient training data (often hundreds or thousands of images per label) were given to the model. Image classification models can learn to predict whether new images belong to any of the classes it has been trained on.

3.5 Custom Vision

For training the model we have used Custom Vision AI, a cloud platform from Microsoft Azure. We have chosen this platform because it has some important features: customization to your scenario, intuitive model creation flexible deployment, built-in security.

The process of training our model was handled entirely by the Azure platform, the only thing that was needed was to provide the set of data/images and the labels. After the model was trained, we have integrated it into our mobile application through an API call. The picture below shows how the predictions for an image look like after the image classification model was trained with our images.

(a) Photo to be classified



Predictions

Tag	Probability
Travel	53.7%
Vintage Photography	15.2%
Fashion	14.5%
Commercial product	8.7%
Dark Photography	8.4%
Stock Photography	8.4%
High Speed	8%

(b) Predictions

Figure 3.6: Predictions

3.6 Geo-location for Mobile Devices

In 2020 it is hard to imagine going to the street and asking people for coordination, but that was a common thing 20 years ago. Being a traveler was in the early '90s was forcing you to be sociable and to politely ask other people for guidance. Nowadays it is a bit weird to do that because almost everyone has a smartphone in their pocket with a guidance application (something like Google Maps) and there are great sources for localization that help users: Global Positioning System (GPS), WiFi positioning system, location-based on mobile networking.

The new generation of smartphones comes with localization applications already installed and they are connected to the built-in sensors from the device. However, the orientation phase is not feasible for all these sensors. The sensors that can be used are an accelerometer, magnetic field, gyroscope, and orientation. The power output for them: 0.25 mA, 7.0 mA for the magnetic field, 13.13 mA for orientation, and 6,1 mA for gyroscope depending on their different characteristics.

Sensors have a wide coverage diameter because they can simultaneously process information from various sources. The precision of the sensor depends on the type of detector used, meaning that accurate data can not be obtained in some cases. However, sensors can improve the accuracy of results obtained via the positioning system using a WiFi signal or through mobile network location according to Google declarations.

Figure 3.8: Global Positioning System

3.6.1 Global Positioning System

GPS, or the Global Positioning System, is a global navigation satellite system that provides location, velocity and time synchronization.[29] This system has grown a lot in the last decade, thus the mobile devices communicate and receive information from 30 international satellites distributed in different parts of the globe. The built-in receiver from a device collects data and the GPS can identify the position of the device after performing a set of operations and calculations based on the intersection point of overlapping spheres.

GPS is very less remarkable when taken alone. GPS recipients supply only coordinates, headings, and some other statistics that are not helpful when viewed independently, but GPS technology is really valuable when combined with maps.

The old generation of phones was using GPS data only to locate the user when an emergency call was made (911 US/ 112 RO), but now sophisticated telephones may obtain and show maps of position for viewing and offering directions. Such turn-over systems typically operate with JAVA, and function with the cellular servers of the user's provider.

Chapter 4

Photograma Application

The hybrid mobile application developed and chosen as an example of what we have discussed throughout this paper it's called ***Photograma*** and it is an application that mainly addresses a given category of users: photographers and people who love photography and art.

The application is composed as a bridge between photographers and people who need good photos, who have a particular style (depending on their preferences or personality) and who would like to hire a photographer (for a hourly rate chosen by the photographer) for a few hours, at a given location for a photo shooting. This application must be patented as an instance of the class of social and entertainment applications, as well as part of the hybrid applications. With a fresh and modern UI, this application is designed to be user-friendly, useful and to cover a particular need for its users.

4.1 Application Functional Description

Photograma is a hybrid application developed in React-Native, with Ruby on Rails support for data authentication and back-end services and PostgreSql for storage, using the Google API for navigation and other third-party systems like AWS S3 to store photos and TensorFlow.js for image classification.

The diagram for the user use case (see Figure 4.1) covers and gives an overview of the functional part of the application, more exactly the user flow and the photographer use case does the same, but from its flow. We have three main actors in the application, the users, the photographers and the servers: rails server, AWS S3 servers, TensorFlow.js for classification and Google servers(for the Map View).

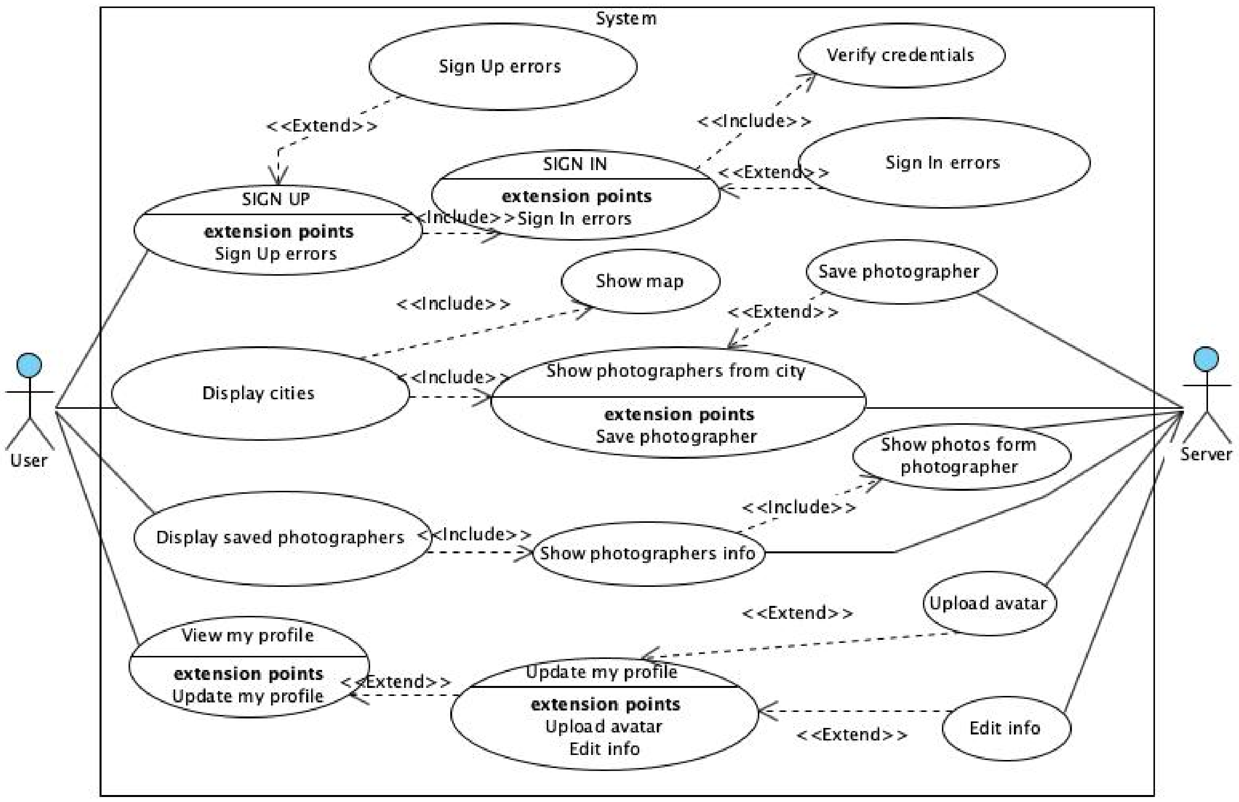


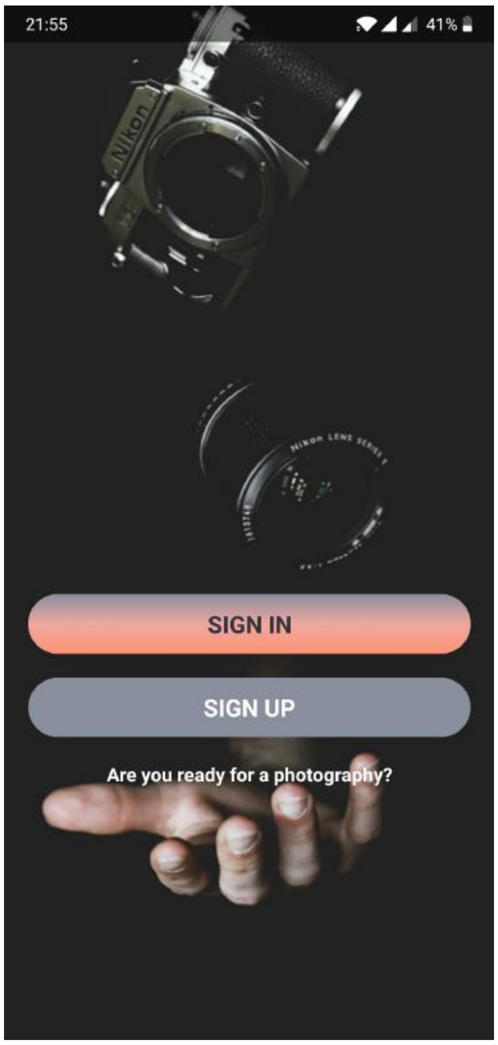
Figure 4.1: Use case diagram for Photograma

The simple user first's interaction with the application, according to the use case is the SIGN IN Screen. The user has to enter valid credentials in order to get the authentication token from the server, which will process the user logging. Because Photograma is an application made for artists, we have considered UI/UX design an important aspect of the application, that's why the SIGN IN Screen has an animation, made with the use of "react-native-reanimated" library.

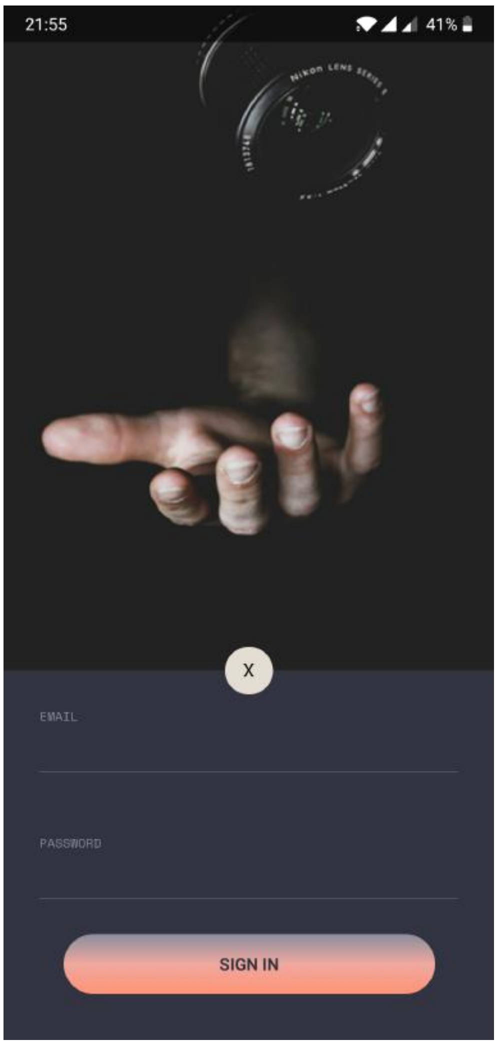
Listing 4.1: Rotation Method

```
function runTiming(clock, value, dest) {
    const state = {
        finished: new Value(0),
        position: new Value(0),
        time: new Value(0),
        frameTime: new Value(0)
    };
    const config = {
        duration: 1000,
        toValue: new Value(0),
        easing: Easing.inOut(Easing.ease)
    };
    return block([
        cond(clockRunning(clock), 0, [
            set(state.finished, 0),
            set(state.time, 0),
            set(state.position, value),
            set(state.frameTime, 0),
            set(config.toValue, dest),
            startClock(clock)
        ]),
        timing(clock, state, config),
        cond(state.finished, debug('stop clock', stopClock(clock))),
        state.position
    ]);
}
```

If the user is new to Photograma, he has the option to Sign Up for an account and he will be asked to enter his information. After this step, the credentials related to his account will be stored in PostgreSql and will appear in the admin Dashboard. The user will have an associated ID, a unique email and a username. The entire login process is implemented for security purposes and the authentication and authorisation are based on Bearer tokens. The token generated will be used whenever a resource is requested from the server. In order to verify if the call was made from a secure source, we will send in headers the auth token, which will have an expiry date. If the token has expired, the user has to login again.



(a) Login

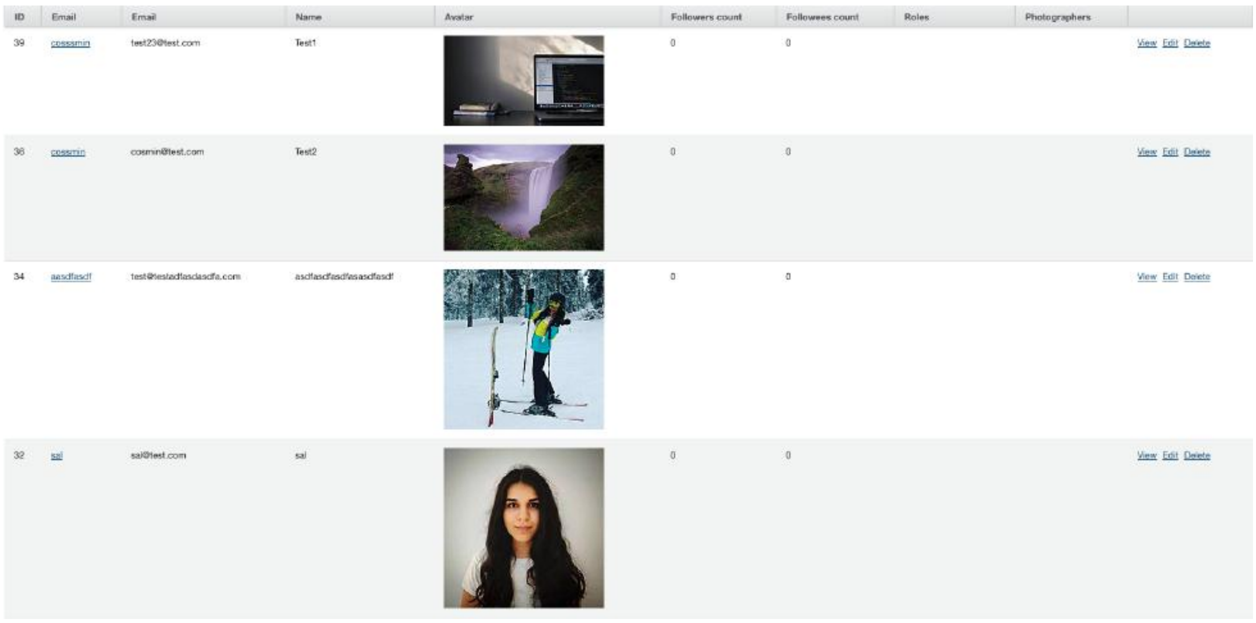


(b) Login Animated

Figure 4.2: LOGIN

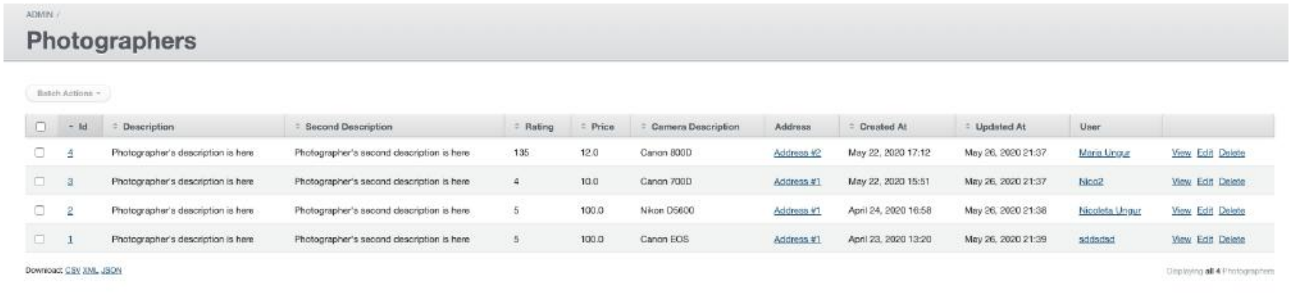
When the user is successfully logged in, it will be redirected to the Dashboard/Home page of the application. A list of cities will be shown there (theoretically the cities where the application is used, the idea that is used by Uber, because only some cities have Uber drivers) and the user can choose from it in order to see some photographers from there.

The admin dashboard with all users (see figure 4.4) and with all photographers (see figure 4.5) will show the users information, but only for those who have admin credentials. The actors for the mobile applications are the users and the photographers, but there is a third type, the admin users, who ensure that everything is working properly. They have access to all the operations that will produce a result in the application. The idea to have an admin dashboard is pretty common in the industry and also in this particular field of mobile development. Moreover, we have developed a login system also for these users that is different than the one used for photographers and simple users. Practically the application developed for the admin users is an independent web application, but with less styling and design. The idea is that all the operations from Photograma can be performed also from that web application, but only if the user has the admin rights.



ID	Email	Email	Name	Avatar	Followers count	Followees count	Roles	Photographers	
39	cosssmin	test23@test.com	Test1		0	0			View Edit Delete
36	cossmin	cosmin@test.com	Test2		0	0			View Edit Delete
34	aasdfasdf	test@testadfasdasdfa.com	asdfasdfasdfasasdfasdf		0	0			View Edit Delete
32	sal	sal@test.com	sal		0	0			View Edit Delete

Figure 4.4: Users and their login information



ADMIN /

Photographers

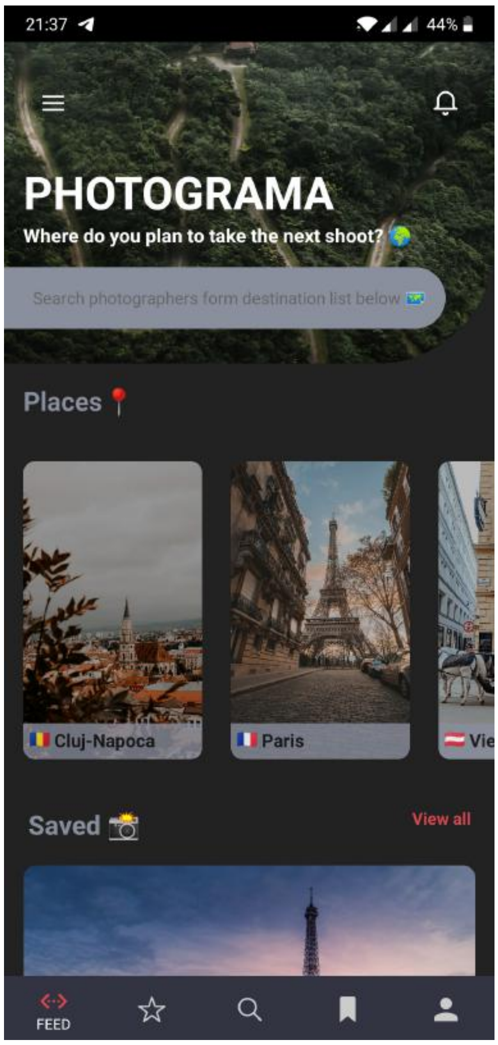
Batch Actions

	Id	Description	Second Description	Rating	Price	Camera Description	Address	Created At	Updated At	User	
☐	4	Photographer's description is here	Photographer's second description is here	135	12.0	Canon 800D	Address #2	May 22, 2020 17:12	May 26, 2020 21:37	Maria Ungur	View Edit Delete
☐	3	Photographer's description is here	Photographer's second description is here	4	10.0	Canon 700D	Address #1	May 22, 2020 15:51	May 26, 2020 21:37	Nico2	View Edit Delete
☐	2	Photographer's description is here	Photographer's second description is here	5	100.0	Nikon D5600	Address #1	April 24, 2020 16:58	May 26, 2020 21:38	Nicoleta Ungur	View Edit Delete
☐	1	Photographer's description is here	Photographer's second description is here	5	100.0	Canon EOS	Address #1	April 23, 2020 13:20	May 26, 2020 21:39	sddsdsd	View Edit Delete

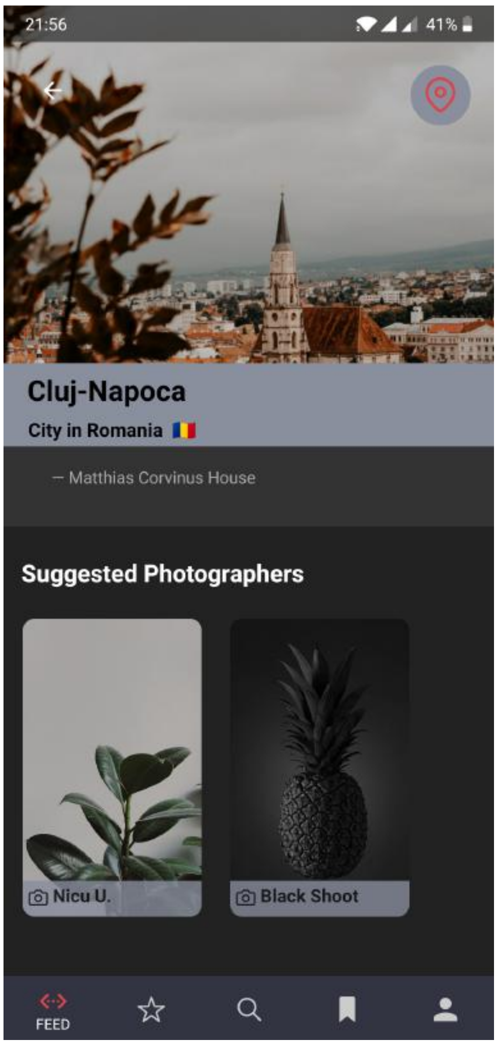
Download: CSV XML JSON

Displaying all 4 Photographers

Figure 4.5: Photographers and their information


21:37
44%
PHOTOGRAMA
Where do you plan to take the next shoot?
Search photographers form destination list below
Places
Cluj-Napoca
Paris
Vie
Saved
View all
FEED

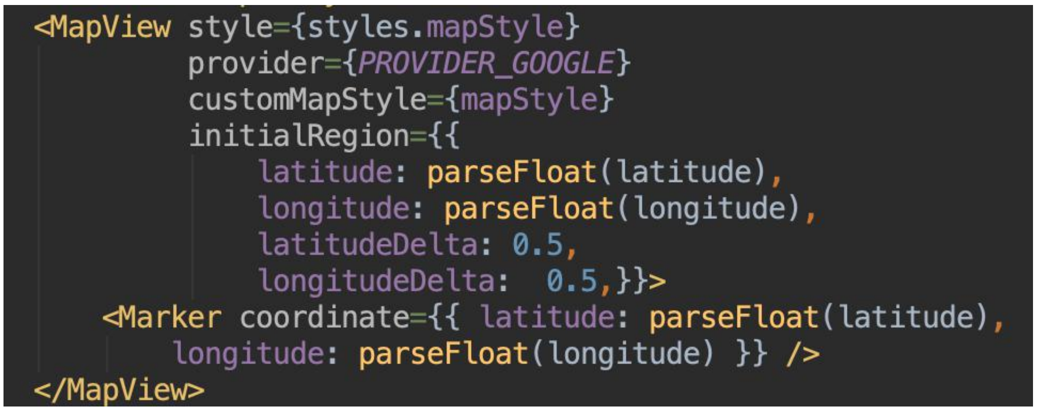
(a) City Screen 2



(b) City Screen 2

Figure 4.6: Home screen and City screen

Another point of interest for our hybrid mobile application, Photograma, is the Home screen with the list of cities. The user can see the most important attractions from a city and can contact and save the photographers that match its style (in the Saved section). There is also a map provided by Google API to the location of the photographer so the user can make an idea about the city and the view for the next shoot. The photographer has attached to its public profile some information like phone and email, where he can be contacted. The application also gives the opportunity to make an appointment in the photographer's agenda.



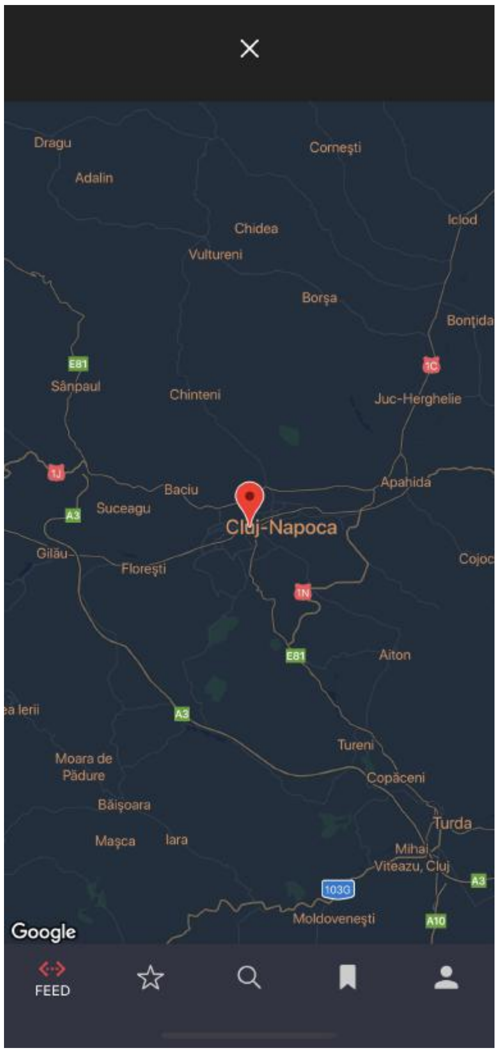
```
<MapView style={styles.mapStyle}
         provider={PROVIDER_GOOGLE}
         customMapStyle={mapStyle}
         initialRegion={{
             latitude: parseFloat(latitude),
             longitude: parseFloat(longitude),
             latitudeDelta: 0.5,
             longitudeDelta:  0.5,}}>
    <Marker coordinate={{ latitude: parseFloat(latitude),
        longitude: parseFloat(longitude) }} />
</MapView>
```

Figure 4.8: MapView Component

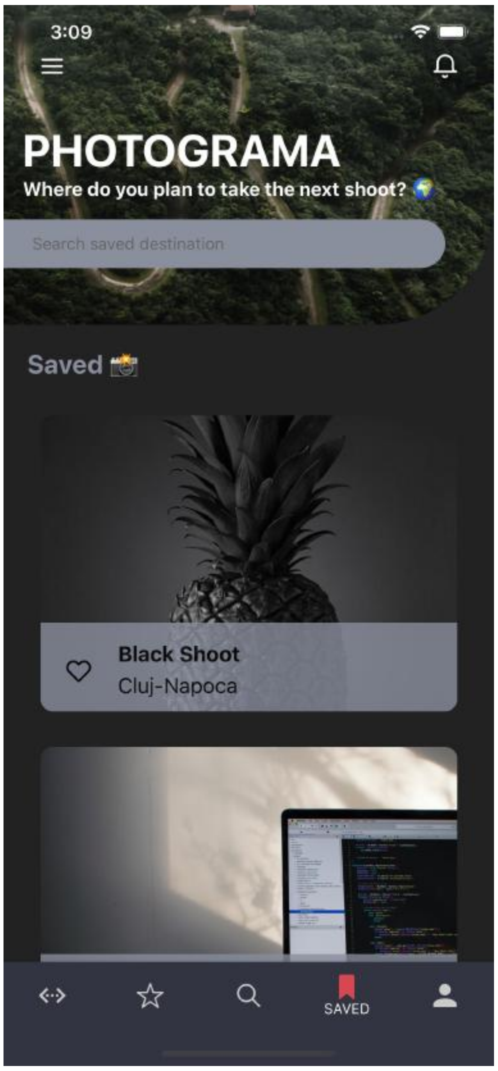
Name	Display photographers from a given city
Actors	The authenticated user/photographer
Basic path	1. The user is located on the *Home* screen 2. The user clicks on the button to go to a city screen 3. The system displays city information and a list of photographers from that city 4. The user views the photographers and their details 5. The user chooses to manifest interest for a photographer 6. The system modifies the number of saved photographers for that user
Entry conditions	The user/photographer has internet connection
Exit conditions	A new photographer is added by the saved photographers

Table 4.1: Use case: Display photographers from a given city

In case the user likes the content and the style of the photographers, he can click the like button (the one from the upper right corner) and save that profile is the saved category as an interest. The Saved screen will show a list that is private and different for each user showing its saved photographers.



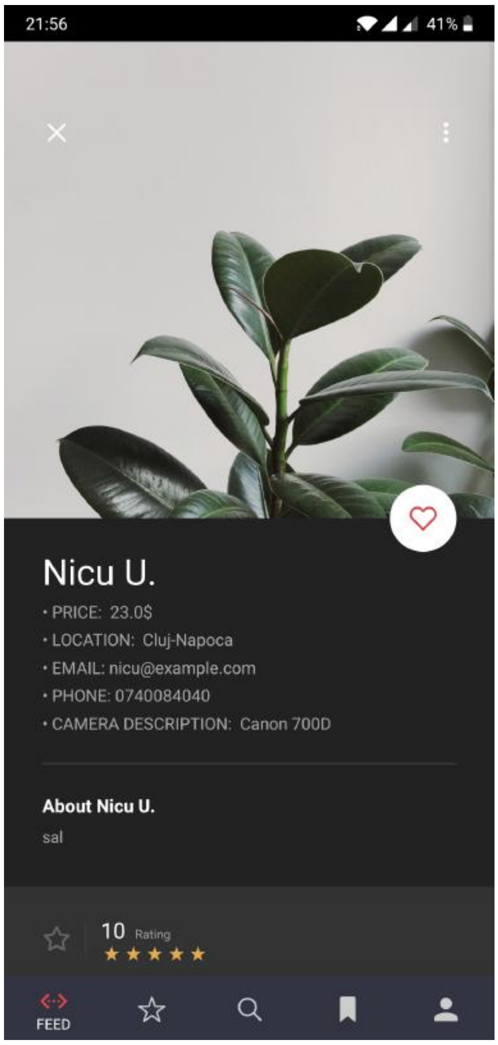
(a) Photographer from city



(b) Saved Photographers

Figure 4.9: Map Screen

The photographer's profile that will be seen by the normal user will have a fixed structure and it's composed by modules: the avatar and the name of the photographer, the save button, some basic information about the photographer (price, location, email, phone number, camera description), two sections for his description (each photographer can present himself in an original way), one section for rating, one section for location (with a map), one section where he can tell the user about the services the he offers and the most important section is called "My style" - the section where he displays the photos that he was taken before, that define his style (from here the name of the module).


21:56 41%

Nicu U.

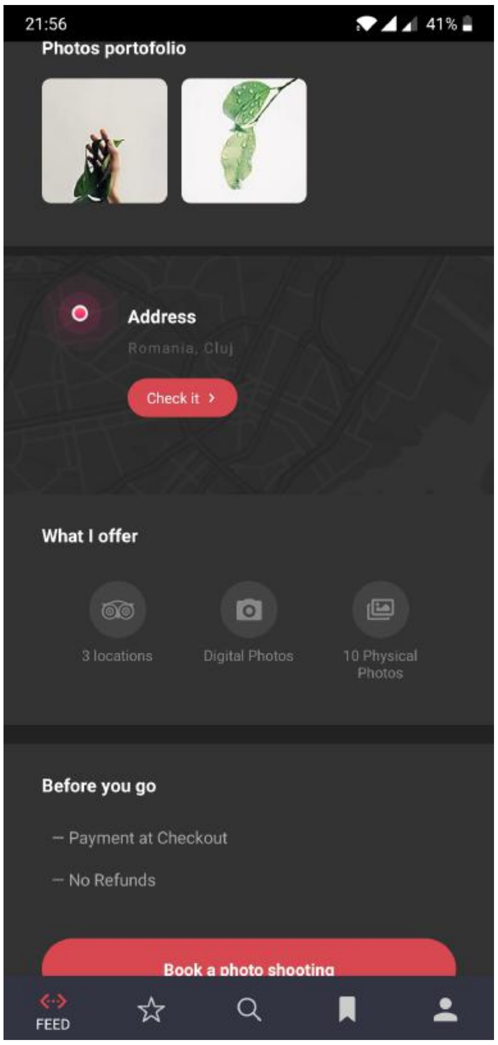
- PRICE: 23.0$
- LOCATION: Cluj-Napoca
- EMAIL: nicu@example.com
- PHONE: 0740084040
- CAMERA DESCRIPTION: Canon 700D

About Nicu U.

sal

10 Rating

(a) Photographer from city


21:56 41%

Photos portofolio

Address

Romania, Cluj

Check it ›

What I offer

3 locations | Digital Photos | 10 Physical Photos

Before you go

– Payment at Checkout

– No Refunds

Book a photo shooting

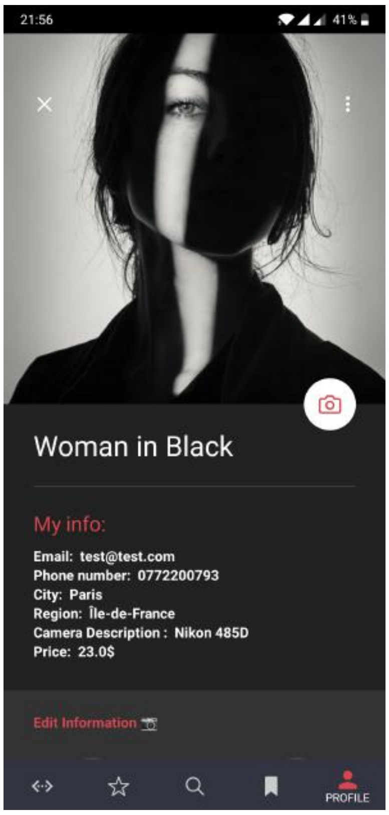
(b) Photographer from city

Figure 4.11: Photographer from city

So far we have described the flow from the point of view of a normal user, but the flow changes a bit when a user is logged in as a photographer. *My Profile* screen will be different, thus the photographer will be able to change the avatar as a simple user, but he can also add a new photo to his *My Style* section that will be shown at his public profile (the one from where a user can make an appointment). He can also edit his information and camera description. The below table is showing the flow of the feature "Add new photo to my gallery".

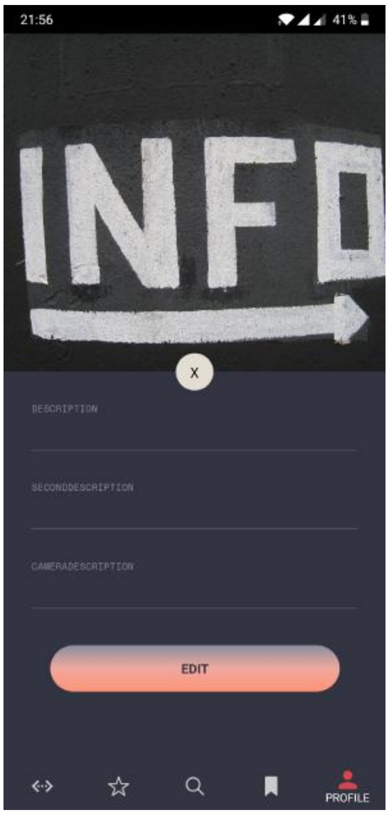
Name	Add new photo to my gallery
Actors	The authenticated photographer
Basic path	1. The photographer is located on the *My Profile* screen 2. The photographer clicks on the button to add new photo 3. The system displays the gallery 4. The photographer adds a photo from the phone gallery 5. The photographer saves and the system saves the photo in AWS S3 6. The system redirects the user back to the *My Profile* screen
Entry conditions	The photographer has internet connection
Exit conditions	A new photo is added by the photographer

Table 4.2: Use case: Add new photo to my gallery



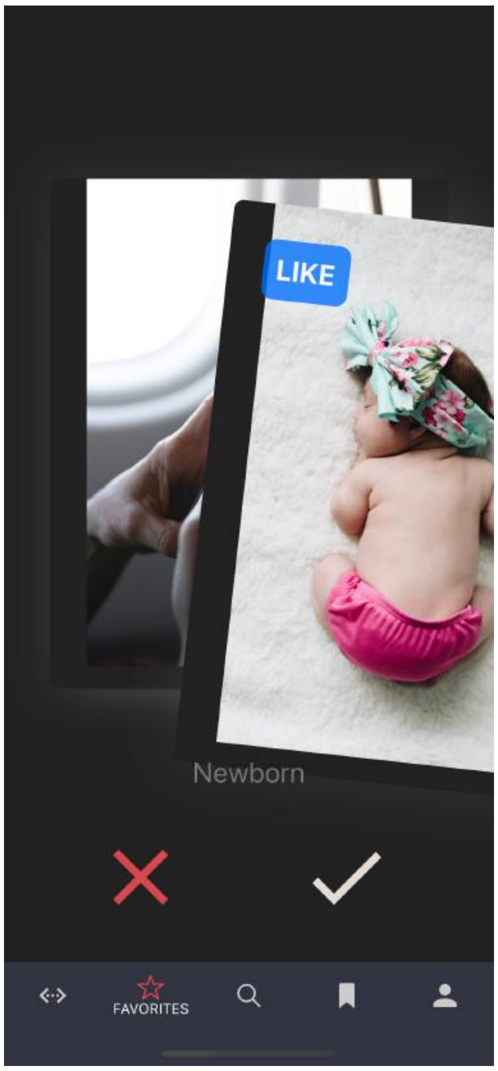
Email: test@test.com
Phone number: 0772200793
City: Paris
Region: Île-de-France
Camera Description : Nikon 485D
Price: 23.0$

(a) My Photographer profile

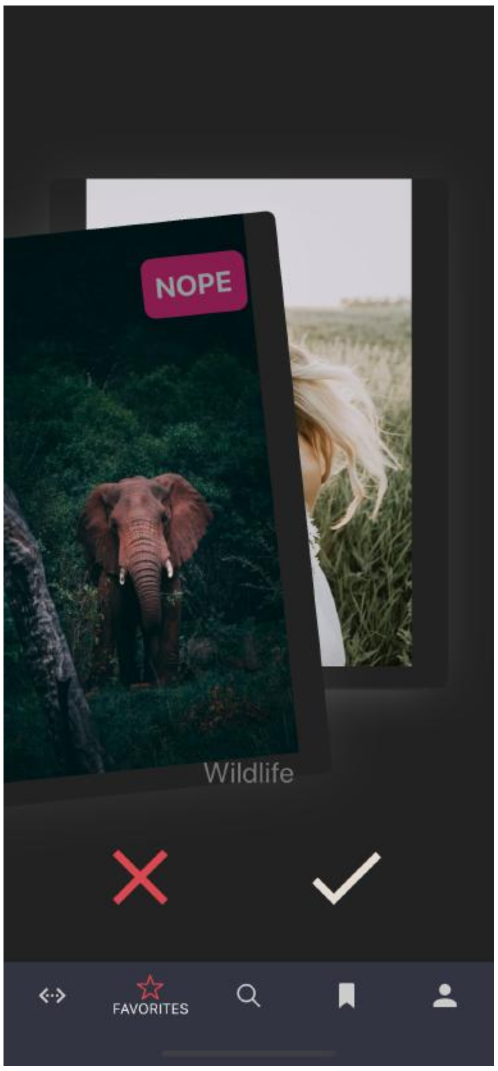


(b) My Photographer profile

Figure 4.13: My profile



(a) My style/Preferences

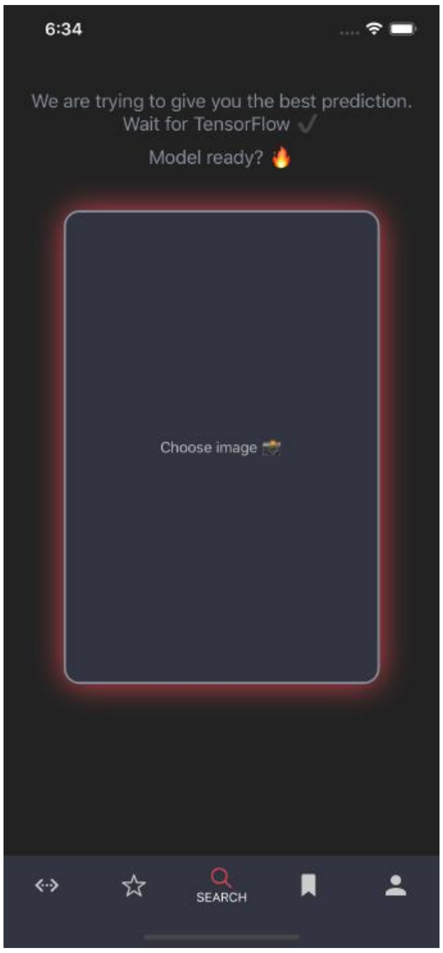


(b) My style/Preferences

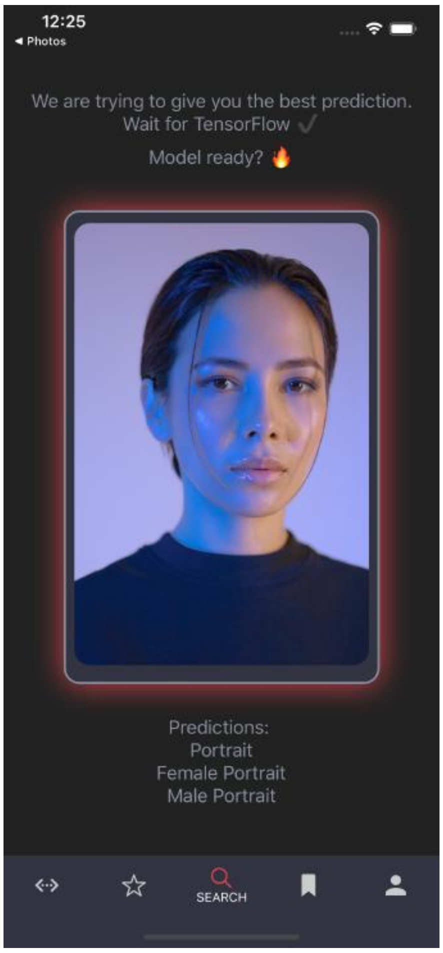
Figure 4.15: My style/Preferences

If the photographer press on the second button from the Bottom navigation, there will be a screen with cards which describes different photography styles and each one can choose his style and he will be able to see photographers that have that style. There will be available 48 types of styles, exactly the ones that were used to train the imagine classification model.

In case a photographer want to test his photography style before adding a new image, he can go to the Model Screen, pressing the middle button from the bottom navigation. He can choose a photo and the model will predict his type. There will be a list with 3 predictions, but the first one is the most accurate one.



(a) Image Classification



Predictions:
Portrait
Female Portrait
Male Portrait

(b) Image Classification

Figure 4.17: Image Classification

Name	Predict photography style
Actors	The authenticated photographer
Basic path	1. The photographer clicks on the button to choose image 2. The system displays the gallery 3. The photographer adds a photo from the phone gallery 4. The system will predict the style
Entry conditions	The photographer has internet connection
Exit conditions	A list of predictions will be shown

Table 4.3: Use case: Predict photography style

4.2 Application Structure and Architecture

Photograma is structured in 2 main parts (that can also be considered standalone applications): ***back-end*** part and ***front-end*** part.

In this section, we will try to explain how we used the stack technologies, how we connected those 2 parts in order to build the final project.

The back-end part was implemented in Ruby on Rails and its main purpose was to receive information from the database and to send it to the front-end part. This application was also handling the admin dashboard and the logic behind it.

We used an MVC pattern for the admin dashboard and for the API part we just created the controllers and the models for each route that we needed.

Listing 4.2: Methods from UsersController implemented in Ruby on Rails

```ruby
module Api
  module V1
    class UsersController < :: Api::V1::ApplicationController
      def index
        @users = User.all
      end

      #GET method to show photographer for a user
      def show
        user = current_user
        if user.update(user_params)
          photographer = user.photographer
          address = user.photographer.address
          render json: {
              data: user,
              my_address: address,
              photographer: photographer
          }, status: :ok
        else
          render json: { message: 'ERROR' }, status: :unprocessable_entity
        end
      end
    end
```

Ruby language, in particular Rails framework, comes with the Active Model module which eases the process of creating classes, objects and tables for the database. If the tables were connected between each other, we added the required foreign keys inside the model class, and with the help of another rails module called "Active Record", the database tables were connected, and the connections were visible from the file "./config/db/schema.rb". The object model evolution refers to the phases of the model until the final state is achieved.

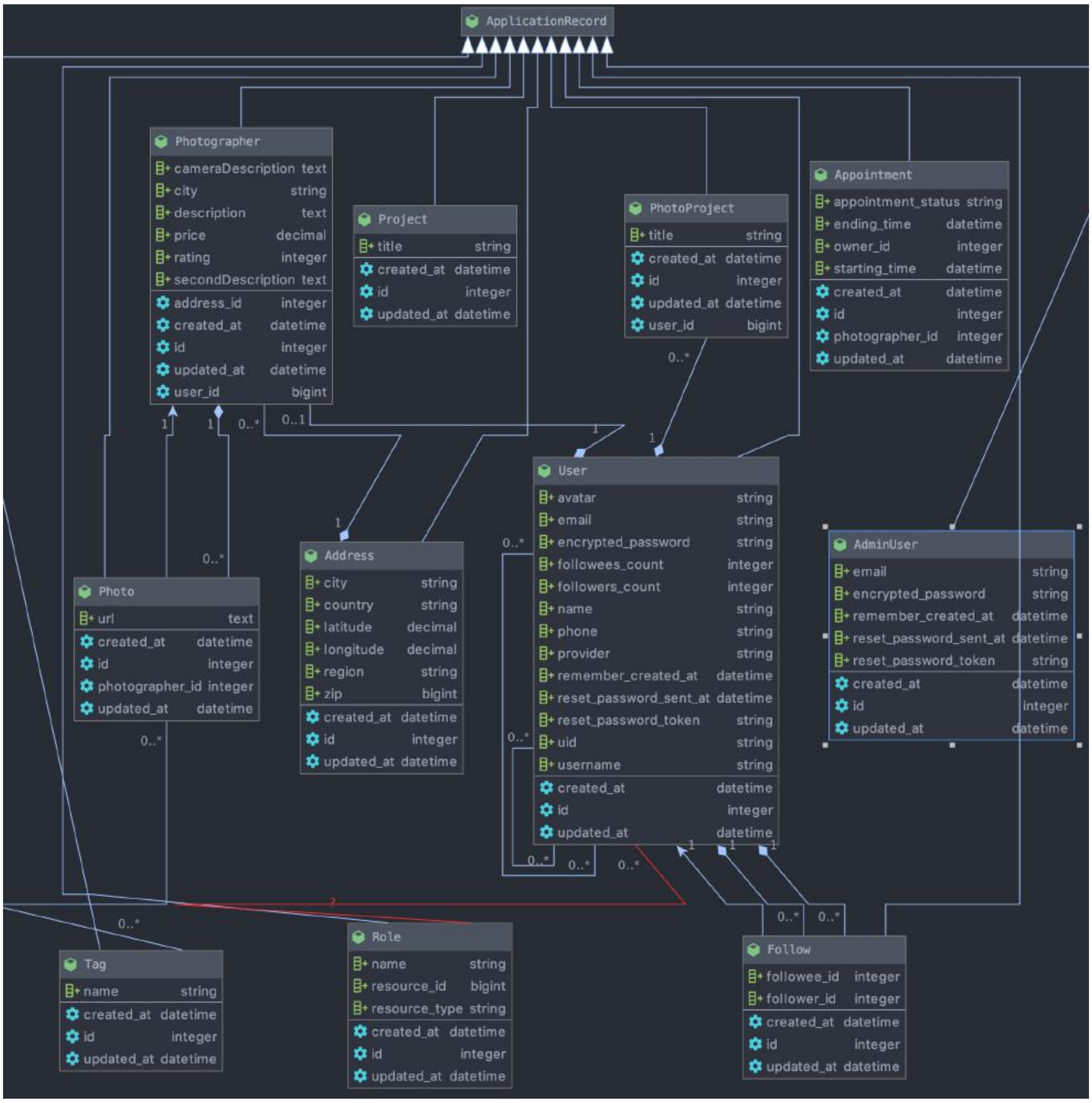


Figure 4.19: Class diagram for the Photograma

In terms of objects, characteristics, associations, and activities, a structural model describes the architecture of the system. Using class diagrams, this is represented in UML. The front-end part was implemented in React Native. This framework developed by Facebook developers is constantly changing and improving. Considering this we have faced also some drawbacks, but we also took advantage of some good parts. React Native gave us the possibility to implement trippingly a modern design for our users and to provide a high-performance user-experience because the asynchronous calls were handled by redux-saga. The below code snippet show how we connected the logic inside Root Saga. Each word written with caps lock is an action that will be triggered at some point in the application. The connected methods can be seen in the "./redux" folder and the "rootSaga" method is placed in the "./config/saga.config.js" file.

Listing 4.3: Root Saga

```
export default function* rootSaga() {
    try {
        // User Sagas
        yield all([takeEvery(UserTypes.LOGIN, login)]);
        yield all([takeLatest(UserTypes.REGISTER, register)]);
        yield all([takeLatest(UserTypes.UPDATE, update)]);
        yield all([takeLatest(UserTypes.EDIT, edit)]);
        yield all([takeEvery(UserTypes.INFO, info)]);
        yield all([takeEvery(PhotosTypes.PHOTOS, photos)]);
        yield all([takeEvery(PhotosTypes.PHOTOS_BY_PHOTOGRAPHER,
                        photosByPhotographer)]);
        yield all([takeEvery(PhotographersTypes.PHOTOGRAPHERS_BY_CITY,
                        photographersByCity)]);
        yield all([takeLatest(MyImageTypes.MY_IMAGE, myImage)]);
        yield all([takeEvery(PhotographerTypes.EDIT_PHOTOGRAPHER,
                        editPhotographer)]);
        yield all([takeEvery(AppointmentTypes.CREATE_APPOINTMENT,
                        createAppointment)]);
        yield all([takeEvery(AppointmentTypes.GET_APPOINTMENTS_FOR_CURRENT_USER,
                        getAppointmentsForCurrentUser)]);
        yield all([takeEvery(HashtagTypes.CREATE_HASHTAG, createHashtag)]);
        yield all([takeLatest(HashtagTypes.GET_HASHTAGS_FOR_PHOTOGRAPHER,
                        getHashtagsForPhotographer)]);
        yield all([takeEvery(LikeTypes.CREATE_LIKE, createLike)]);
        yield all([takeEvery(LikeTypes.DELETE_LIKE, deleteLike)]);
        yield all([takeEvery(SavedTypes.GET_SAVED_FOR_USER, getSavedForUser)]);
    } catch (err) {
        console.log(err);
    }
}
```

4.2.1 Application Integration With Other Systems

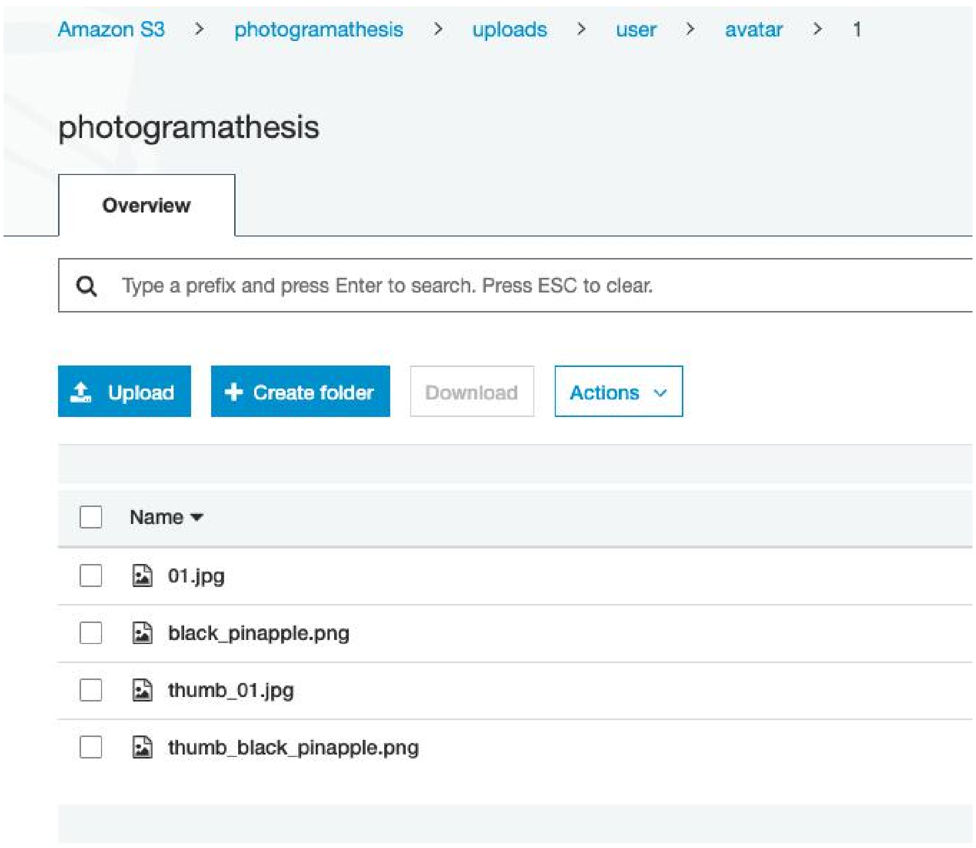
As mentioned before during this paper, the application communicates via REST calls with the rails server to get the details needed in order to function correctly. But our server is not the only server with which the application interacts. There are also some systems, called Third parties, which facilitate the development of the application: Google API for map view and location services, AWS S3 for image storage, Custom Vision from Azure which gives the predictions for the image classification.

We choose AWS S3 because it is one of the most popular cloud services at the moment, it is fast and in terms of cost, for a small business is a very good option (for us it is free, because AWS offers free access for students and researchers). Taking that into account I decided that for my needs, this cloud platform is a good fit for storing photos.

When a photo is uploaded from the mobile application, the information of that photo is sent raw to the Rails server, which will handle the format of the photo and then will send it to the S3 console. Rails will send to S3 the same photo twice, the original photo and a thumb of it (a resized photo). The below code snippet show exactly the process of resizing an image (creating a thumb) that will be further sent to S3. We use the thumb version when the photos are shown in Photographer's page and the original sized photo, for the avatars.

Listing 4.4: Root Saga

```
# Create different versions of your uploaded files:
version :thumb do
  process resize_to_limit: [200, 200]
end
```


Amazon S3 > photogramathesis > uploads > user > avatar > 1
photogramathesis
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions
Name

01.jpg
black_pinapple.png
thumb_01.jpg
thumb_black_pinapple.png

(a) How photos are stored in S3

Figure 4.20: How photos are stored in S3
Source: `https://s3.console.aws.amazon.com`

When we first start to think about image classification as a feature for Photograma application, in our minds raised a question: "Why we classify images?". The answer for our application was simple: we wanted to provide a tool for our users to test their photography type, but Image classification has multiple uses in different domains, including those that have an impact on humanity, like cancer detection and many others. For the purpose of this paper, we will explain only our particular usage of it.

For the image classification model, we have chosen another cloud service, Microsoft Azure, more specifically a product that is designed for model training, Custom Vision AI. We did the exact same research when we started to search for the tool that will cover our needs, as we did for AWS and we choose this product because have it was easy to configure and was free for students. The idea was simple, we needed to have a big set of data/images to train our model, and for each image from that set a different label (or labels). After the set was provided to the model, this was trained in different iterations in order to increase its accuracy. When we were satisfied with the trained model, we downloaded the model from Custom Vision and we integrated it with our React Native application. For a faster response, after downloading the model we also integrated their API in our application (both steps were needed in order to be faster and to properly function).

Chapter 5

Conclusion and Future Work

In this paper, we tried to underline the importance of the mobile applications market in today's world, people's need for communication, and for sharing parts of their day to day lives and also to give readers a novelist view of our tech world. Photograma belongs to the category of hybrid applications and to the category of visual applications (we have put an emphasize on UI/UX design and we have explained the importance of design in this paper).

In the development of the application, we have chosen React Native for developing the Hybrid mobile application and Ruby on Rails for back-end part and PostgreSQL for storage, AWS S3 for image storage and Google API for map view and location. There are a lot of things to be taken in consideration when developing a mobile application and there are always things that can be improved.

For now, Photograma is just a prototype providing a model for future work in the field of mobile applications made for photographers, influencers and communication between its users. One thing that we will implement in the future and will make a big impact is the existence of a chat inside the application, that will ease the communication flow between photographers and simple users.

Other things that are going to be implemented are the Stripe API for payments inside the application and authentication with other systems and integration with different devices like smart watches or smart bracelets. The user will receive a notification each time a new appointment is made. Moreover, each appointment will be synchronised with a Google or Apple calendar.

In conclusion, we tried to make a useful and user friendly application for the persons that need a photographer at a given point in time and in a certain location, but also an application made for the other category of users, the photographers. Each user has something to gain from using this application and to conclude, if the users are happy, we reached our goal.

Bibliography

[1] The Number of Smartphones world-wide
https://www.bankmycell.com/blog/how-many-phones-are-in-the-world [May 2020]

[2] Definition and History of Photography and the photography in the digital age
https://www.britannica.com/technology/photography [April 2020]

[3] [Study] Study - Mobile vs Web applications
https://www.perficientdigital.com/insights/our-research/mobile-vs-desktop-usage-study [April 2020]

[4] Most popular mobile social networking apps in the United States
https://www.statista.com/statistics/248074/most-popular-us-social-networking-apps-ranked-by-audience/ [September 2019]
https://www.statista.com/statistics

[5] The definifion of a mobile application/mobile phone from Cambridge Dictionary https://dictionary.cambridge.org/dictionary/english/mobile-application [April 2020]

[6] Informations about first iPhone
https://appleinsider.com/ [April 2020]

[7] Article about the invention of the first smartphone https://simpletexting.com/where-have-we-come-since-the-first-smartphone/ [May 2020]

[8] A study made to register the time that mobile phones users spend on mobile apps or web apps
https://www.emarketer.com/ [May 2020]

[9] The difference between Native and Hybrid mobile apps
https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/#Native_App_Development

[10] Author: Akshat Paul, Abhishek Nalwaya
Title: "React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications"
Publisher: Apress [Jun 22, 2019]
ISBN: 9781484244531

[11] Author: Devin Abbott Houssein Djirdeh, Anthony Accomazzo, Sophia Shoemaker
Title: "Fullstack React Native The Complete Guide to React Native"
Publisher: Fullstack.io [Jun 12, 2017]

[12] Author: Adam Boduch
Title: React and React Native - Complete guide to web and native mobile development with React"
Publisher: Packt Publishing Ltd [March 2017]
ISBN: 9781789346794

[13] [REST] Representational State Transfer (REST) [May 2020]
`https://dzone.com/refcardz/rest-foundations-restful`

[14] Author: Michael Hartl
Title: Online book - RAILS-TUTORIAL `https://www.railstutorial.org/book`

[15] Authors: David Flanagan, Yukihiro Matsumoto
Title: The Ruby Programming Language (Covers Ruby 1.8 and 1.9)
Publisher: O'Reilly Media [Originally published: December 21, 1995
ISBN 13: 9780596516178
ISBN 10: 0596516177

[16] Authors: Dave Thomas, David B. Copeland, and Sam Ruby
Title Agile Web Development with Rails 5.1
Publisher: Pragmatic Bookshelf [Originally published 2006]
ISBN 10: 1680502514
ISBN 13: 978-1680502510

[17] [Ruby on Rails] Ruby on Rails official site
`https://rubyonrails.org/` [May 2020]

[18] [Heroku]Heroku official site
`https://www.heroku.com/` [May 2020]

[19] [Devise-token-auth] Devise token auth official site
`https://devise-token-auth.gitbook.io/` [May 2020]

[20] [Mozilla Front] Mozilla Front official site
`https://developer.mozilla.org/` [May 2020]

[21] [Redux-Native] React native official documentation
`https://reactnative.dev/` [May 2020]

[22] [Expo] Expo official documentation
`https://expo.io/` [May 2020]

[23] [Redux-Saga] Redux-Saga official documentation
`https://redux-saga.js.org/` [May 2020]

[24] [AWS S3 article]
Author: Santiago Obrutsky, Emre Erturk
Institute: Eastern Institute of Technology, New Zealand
Title: "Multimedia Storage in the Cloud using Amazon Web Services: Implications for Online Education" [August 2016]

[25] Amazon Web Services official site
`https://aws.amazon.com/what-is-aws/` [May 2020]

[26] [TensorFlow.Js] Official documentation of TensorFlow.Js
`https://www.tensorflow.org/js` [May 2020]

[27] [Machine Learning] Machine Learning Definition
`https://expertsystem.com/machine-learning-definition` [June 2020]

[28] [Image Classification] Information about Image Classification
`https://www.tensorflow.org/lite/models/image_classification/overview`

[29] [GPS] Global Positioning System - GPS
`https://www.geotab.com/blog/what-is-gps/` [June 2020]