



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

MULTIPLATFORM PHOTO ORGANIZER

MULTIPLATFORMNÍ APLIKACE PRO ORGANIZACI FOTOGRAFIÍ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR

AUTOR PRÁCE

SUPERVISOR

VEDOUCÍ PRÁCE

MAGDALÉNA URMÍNOVÁ

Ing. JIŘÍ HYNEK, Ph.D.

BRNO 2020

Bachelor's Thesis Specification



Student: **Urmínová Magdaléna**
Programme: Information Technology
Title: **Multiplatform Photo Organizer**
Category: Information Systems

Assignment:

1. Get acquainted with the problem of management and organization of photos. Study photo metadata formats. Analyze the existing photo organizers.
2. Study the principles of development of multiplatform applications and design of usable user interfaces. Find and analyze user requirements and associate the requirements with the existing photo organizers.
3. Design a multiplatform application which can be used to organize photos. Consider the user requirements. Focus on the analysis of duplicate photos.
4. Implement the designed application.
5. Perform usability testing, evaluate the results and propose further extensions.

Recommended literature:

- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers/Elsevier, 2010, ISBN: 978-0-12-375030-3.
- Preece, J.: *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, 2015, ISBN: 978-1-119-02075-2.
- Electronjs.org: *Electron Documentation* [online]. 2019 [cit. 2019-10-13]. Available at: <https://electronjs.org/docs>.

Requirements for the first semester:

- Items 1 to 3.

Detailed formal requirements can be found at <https://www.fit.vut.cz/study/theses/>

Supervisor: **Hynek Jiří, Ing., Ph.D.**
Head of Department: Kolář Dušan, doc. Dr. Ing.
Beginning of work: November 1, 2019
Submission deadline: July 31, 2020
Approval date: October 23, 2019

Abstract

This bachelor's thesis focuses on creating a multi-platform desktop application for photo organisation and management. It analyses user requirements and explains the concept of photography metadata. Later, it studies the principles of usable multi-platform application design and describes three used frameworks for multi-platform development. Electron is introduced, and its architecture is illustrated. The process of the application implementation is described, along with a list of used tools and libraries. The evaluation of the application was determined both by user testing and performance testing.

Abstrakt

Táto bakalárska práca sa zaoberá vytvorením multiplatformnej desktopovej aplikácie na organizáciu a manažment fotografií. Analyzuje požiadavky užívateľov a vysvetľuje koncept metadát fotografií. Následne študuje princípy dizajnu použiteľnej multiplatformnej aplikácie a popisuje tri nástroje na multiplatformný vývoj. Predstavuje Electron a popisuje jeho architektúru. Proces implementácie je vysvetlený spolu s vymenovaním použitých nástrojov a knižníc. Vyhodnotenie aplikácie bolo určené na základe užívateľského testovania a testovania výkonu.

Keywords

Electron, React, JavaScript, photography management, photography metadata, desktop application, multi-platform application, IndexedDB

Kľúčové slová

Electron, React, JavaScript, organizácia fotografií, metadáta fotografií, desktopová aplikácia, multiplatformná aplikácia, IndexedDB

Reference

URMÍNOVÁ, Magdaléna. *Multiplatform Photo Organizer*. Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Hynek, Ph.D.

Rozšírený abstrakt

Práca sa zaoberá vývojom aplikácie na triedenie a organizovanie fotografií. Cieľom práce je navrhnúť aplikáciu tak, aby spĺňala podmienky užívateľov a bola dostupná na čo najviac operačných systémoch.

V prvej časti popisujem organizáciu fotiek ako takú. Zaoberám sa štatistikami z rôznych zdrojov ohľadne najpoužívanejších formátov fotiek spolu s konceptom toho, koľko fotografií užívateľa ročne vytvorí. Následne analyzujem možnosti úložiska týchto fotografií. Popisujem výhody a nevýhody využívania virtuálnych vzdialených diskov v porovnaní s lokálnym ukladaním. Následne hodnotím systém, ktorým užívatelia svoje fotky organizujú online či offline spolu s vymenovaním najpoužívanejších aplikácií určených na tento účel. Rozoberám ich silné stránky a nedostatky. V poslednej sekcii prvej kapitoly analyzujem spôsoby detekcie duplikátov vo fotkách a bližšie popisujem využívané existujúce riešenia.

V nasledujúcej kapitole predstavujem pojem metadáta fotografií. Stručne objasňujem ich históriu a vývoj do dnešnej podoby. Dnes sa využívajú tri formáty metadát. Popisujem ich štruktúru a účel spolu s príkladom, ako metadáta konkrétnej fotky môžu vyzeráť.

Aby som mohla začať navrhovať danú aplikáciu, bolo potrebné stanoviť princípy návrhu použiteľnej multiplatformnej aplikácie. Práve týmto princípom sa venujem v tretej kapitole. Každý operačný systém má jedinečné vlastnosti, ktoré musí multiplatformná aplikácia zohľadniť, napríklad práca so súborovým systémom, vzhľad aplikačného rozhrania alebo rôzne cesty ku konfiguračným súborom. Tieto problémy je možné čiastočne alebo úplne vyriešiť použitím nástroja na vývoj multiplatformných aplikácií. Porovnávala som výhody a nevýhody troch – Haxe, Electron a NW.js. Zároveň som predstavila techniky, ktoré som využila na tvorbu použiteľnej aplikácie. Ide o techniky na získavanie požiadaviek užívateľov, vytvorenie empatie s nimi a definovanie prípadov užitia aplikácie.

Po objasnení všetkých pojmov som mohla následne vykonať analýzu celkového problému organizácie fotiek. Požiadavky užívateľov som získala pomocou dotazníka, rozhovorov a následného vytvorenia osoby. Na základe získaných informácií som vytvorila požiadavky na systém a porovnávala ho s existujúcimi aplikáciami. Zistila som, že požiadavky užívateľov nie sú úplne splnené, a tým sa naskytla príležitosť navrhnúť a implementovať novú aplikáciu pre tento účel.

Vďaka všetkým získaným dátam som vytvorila návrh aplikácie, popísaný v ďalšej kapitole. Aplikácia si musí uchovávať dáta o jednotlivých fotkách. Po ich analýze som navrhla výslednú štruktúru dát, ktoré som modelovala použitím ER diagramu. Funkcie užívateľského rozhrania som prezentovala na diagrame prípadov užitia a následne som všetky tieto funkcie jednotlivo popísala. Aplikácia má byť zostavená zo siedmich hlavných okien, ktoré reprezentujú sedem hlavných funkcionalít – import fotografií, ich zobrazenie na časovej osi, zobrazenie na mape, rozdelenie fotiek do albumov, správa zariadení, detekcia duplikátov a vyhľadávanie fotiek.

V nasledujúcej kapitole popisujem proces implementovania získaného návrhu. V prvej časti definujem architektúru projektu. Z popísaných nástrojov na tvorbu multiplatformnej aplikácie som vybrala Electron. Vysvetľujem jeho architektúru a princíp komunikácie medzi užívateľským rozhraním a hlavným procesom Electronu. Na implementáciu užívateľského rozhrania som sa rozhodla použiť React, ktorý v tejto časti tiež popisujem. Súčasťou architektúry projektu je aj výber databázy. Po preskúmaní možností bola vybraná IndexedDB. V druhej časti kapitoly popisujem všetky použité knižnice – spôsob extrahovania metadát z fotiek, zobrazenie máp a iné knižnice použité pre užívateľské rozhranie. V poslednej časti kapitoly popisujem implementačné detaily zaujímavých častí. Demonštru-

jem algoritmus na detekciu duplikátov a vyhľadávanie fotiek podľa užívateľom zadaných parametrov.

Testovanie aplikácie som rozdelila na dve časti. Prvou je testovanie výkonu aplikácie a analýza časovo náročných procesov. Pri importovaní fotiek som merala čas potrebný na čítanie metadát, vytvorenie miniatúry fotky a jej následné uloženie v databáze. Pri vyhľadávaní duplikátov som sledovala závislosť medzi počtom fotiek v knižnici a časom potrebným na detekciu duplicitných fotiek. Pri hľadaní fotiek podľa zadaných parametrov som sledovala závislosť počtu fotiek v knižnici s počtom zadaných parametrov a výsledným časom potrebným na operáciu. Aplikácia bola zároveň testovaná na viacerých operačných systémoch. Druhou časťou testovania je užívateľské testovanie. Prebiehalo formou plnenia úloh a následnými rozhovormi o vykonaných akciách. Sledovala som hlavne jasnosť a intuitívnosť užívateľského rozhrania spolu s použiteľnosťou aplikácie. Pripomienky a nápady užívateľov boli zhodnotené a pridané k ďalšiemu vývoju.

Budúci vývoj aplikácie môže zahŕňať automatickú detekciu tvárí z fotiek, detekciu podobných fotiek na základe viacerých kritérií alebo prispôsobiteľnú farebnú tému užívateľského rozhrania.

Multiplatform Photo Organizer

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the academic supervision of Ing. Jiří Hynek, Ph.D. All the relevant sources, which were used during the preparation of this thesis, are properly cited and included in the list of references.

.....
Magdaléna Urmínová
July 30, 2020

Acknowledgements

I would like to thank my supervisor Ing. Jiří Hynek, Ph.D. for all his patience and guidance, which he always provided me with. Also big thanks to my little sister, who helped me with choosing the right name for the final application.

Contents

1	Introduction	3
2	Photography management	5
2.1	Photo storage	6
2.1.1	Online storage	6
2.1.2	Offline storage	7
2.2	Organisation of photos	7
2.2.1	Managing photos offline	7
2.2.2	Managing photos online	8
2.2.3	Photo organising applications	8
2.3	Detection of photo duplicates	13
2.3.1	Tools for duplicate detection	13
2.3.2	Photo management with duplicate detection	14
3	Photography metadata	15
3.1	History of photography management and metadata	15
3.2	Metadata today	15
3.2.1	Metadata formats	16
3.2.2	Metadata properties	17
4	Principles of usable multi-platform application design	19
4.1	Principles of multi-platform development	19
4.2	Development of multi-platform applications	20
4.2.1	Software platform possibilities	20
4.3	Design of usable application	22
4.3.1	Terminology	22
4.3.2	Used techniques	22
5	Problem analysis	25
5.1	User requirements	25
5.1.1	Question form	25
5.1.2	Interviews	26
5.1.3	Personas	27
5.1.4	System requirements	28
5.2	Existing solutions	29
5.3	Summary	30

6	Solution design	31
6.1	Data structure design	31
6.1.1	Data analysis	31
6.1.2	Final data structure	31
6.2	UI design	32
6.2.1	Use-case diagram	32
6.2.2	Structure of the application UI	33
7	Implementation	39
7.1	Project architecture	39
7.1.1	Electron	39
7.1.2	React	40
7.1.3	Data storage	40
7.2	Used libraries	41
7.2.1	Leaflet	41
7.2.2	Exiftool	41
7.2.3	Imp and ImageMagick	42
7.2.4	Semantic and Material-UI	42
7.2.5	Timeline component	42
7.3	Implementation details	42
7.3.1	Working with metadata	42
7.3.2	Duplicate detection	43
7.3.3	Device management	45
7.3.4	Searching	45
7.4	Packaging and used developer tools	46
7.4.1	Developer tools	46
8	Testing	47
8.1	Performance testing	47
8.1.1	Import	47
8.1.2	Duplicate detection	48
8.1.3	Search	48
8.1.4	Multiplatform testing	49
8.2	User testing	49
8.2.1	Clarity of the user interface	49
8.2.2	Application usability	49
9	Conclusion	50
	Bibliography	51
	A Questionnaire	53
	B CD content	55

Chapter 1

Introduction

Taking photos has never been easier. Nowadays, everyone carries a camera in their pocket. It is not rare that people take many photos and never even look at them. When the time comes, and space on their phone storage or memory card is running out, they just move them to their laptop or external hard drive and take more and more photos with their phone or camera. It is really easy to get lost in the photos when they are all over the place, not sorted, just recklessly stored where they fit. The problem starts when a person such as one mentioned above, is trying to find one particular photo. Alternatively, when something inconvenient happens, and suddenly lots of pictures are stored many times in different locations under various names, and the duplicates are just too much to manage manually.

That is why people use many applications to make the process easier. To manage photos taken using various devices, they usually use a cloud service that provides the advantage of real-time synchronisation and access to everything they have taken so far, mostly sorted. However, these services come with two main disadvantages.

First, accessing a photo stored in the Cloud requires a connection to the internet. Additionally, quality photos can take up much space. A cloud service can compress user's photos, or a lot of data will be spent to load or download the photos from the virtual storage.

The other disadvantage of using cloud services is that people are not often familiar with privacy issues. Many cloud services (mostly those that are “free”) use the users' stored data for machine learning or advertisements. Those clouds that provide secured services are usually quite expensive.

The purpose of this paper is to analyse users' needs and requirements in terms of photo management and organisation, cover existing solutions and their main advantages and disadvantages, and design and implement a free and private solution that is easy to use and understand, and functional on many platforms. Based on user requirements analysis, I will attempt to show that there is a need for an application just like this one among users.

Chapter 2 describes photography management as a process of storing and organising photos and identifies different options of storage. In addition, existing solutions for photo management are listed and described. Lastly, the analysis of photo duplicates detection is defined, and available tools for duplicate detection are listed.

Chapter 3 is focused on defining the term photography metadata and explains the history of photo management, which led to metadata attributes that are used nowadays. Moreover, the existing metadata formats are described.

In Chapter 4, the principles of developing and designing a usable multi-platform application are described. The possible software frameworks and tools used for multi-platform development are compared. In the last section, the principles for developing a usable application are explained, and the used techniques are listed.

In Chapter 5, the problems concerning photo management are analysed in depth. The first section is focused on understanding user requirements based on questionnaire and interviews. Based on the collected data, two personas are created. In the second section, the existing solutions described in Chapter 2 are compared with user requirements and their advantages and disadvantages are named. In the last section, the final requirements for a new application are presented.

Based on Chapter 5, the design of the new application is described in Chapter 6. It contains a design of the data structure and the user interface design of the final application.

In Chapter 7, the process of implementation is described. It demonstrates application architecture and description of used technologies and libraries. Additionally, it explains the selected parts of the implementation.

Finally, the last Chapter 8 is focused on the evaluation process and testing – both performance and user testing.

Chapter 2

Photography management

Not only professional photographers, but many ordinary people or amateurs in photography struggle with the process of photo management: the number of digital photographs they possess and choosing the right software with the possibility to invest in more advanced tools. Moreover, some struggle with the preference of privacy and easy access—whether to keep their images private and offline or to synchronise all their devices using an online tool and share one library or the Cloud. The list of possible software options is filled with applications of various kinds.

Before phones had built-in cameras, or even before owning a film camera became popular, people did not own many photographs. The ones they had, they stored in boxes or sorted in photo albums. A small amount of photos is quite easily manageable. However, with the process of digitalisation, the number of photos increased gradually. Nowadays, as presented in Figure 2.1, the number of digital photos in the world is counted in trillions.

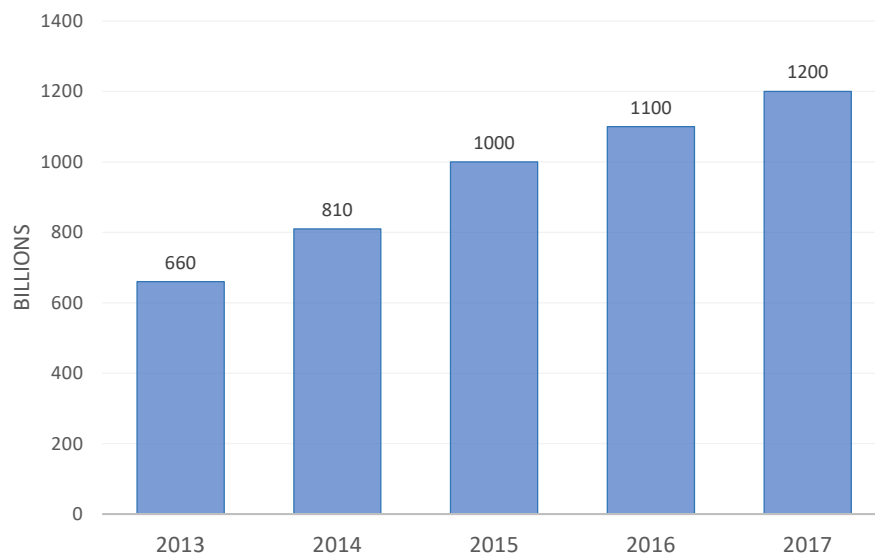


Figure 2.1: The number of photos taken worldwide each year 2013-2017, redrawn from online source².

²<https://focus.mylio.com/tech-today/heres-how-many-digital-photos-will-be-taken-in-2017-repost-oct>

Based on statistics [4] from 2018, a representative person captured 20.2 photos per day in the United States, on average. In contrast, Europe was in last place, with 4.9 photos per day. It is not just the number of photos people deal with, but also the increasing file size. With the always improving quality of cameras, one image captured by a high-resolution camera in the RAW format can take up to 60 MB of space. However, average photo size varies usually between 3 to 15 MB, depending on resolution, format and compression. Based on the survey [10] from 2018, people store their photo and video data in units of terabytes. With the ever-growing library of pictures, if they are not sorted regularly, people eventually run out of storage. That is why the question of where and how to store our data is fundamental. The most common image file type is JPEG. Other widely used formats are PNG, TIFF, GIF and RAW formats. In article [12], Jeff Kabachinski describes formats of pictures in more detail.

2.1 Photo storage

The most significant question users ask before choosing their right way of managing photos is whether to store the data locally or to use a cloud service. The main advantages and disadvantages are presented in the following paragraphs.

2.1.1 Online storage

Probably the biggest downside of using online storage for practically any data is the necessity of internet connection to import and access the files. Digital photos can take up much space. Uploading quality photos to the Cloud takes much time, especially when the internet connection is slow. Moreover, accessing or migrating remotely stored data in the future would require downloading them. However, pictures that are already online are more comfortable to share with other users on social media. Although, the requirement of constant internet connection is not an issue for some users.

The pertinent reason for people to use cloud storage for storing their photo is multi-device synchronisation and often automatic backups. As long as they are connected, the library of photos is accessible from phones, laptops, tablets, and any other device the user chooses and signs in on.

One of the disadvantages might be the lack of control over the user's data. Since the data is held offsite by a company which users do not control, they sometimes lack the ability to customise their storage. However, it is not an issue for moderate cloud costumer.

Unlike customisation, security and privacy have been an important factor for users. Cloud storage means handing over the control of confidential information to a third party company. There were some affairs concerning cloud companies in the past—for instance, in June 2011, TechCrunch reported that all Dropbox accounts could be accessed without password for four hours³. Another example is an incident called “Celebgate”, occurred during 2014 when a hacker leaked celebrities' photos from Apple's iCloud service. Apple later reported that the victims' iCloud account information was obtained using “a very targeted attack on user names, passwords and security questions”, such as phishing and brute-force attack guessing⁴.

³<https://techcrunch.com/2011/06/20/dropbox-security-bug-made-passwords-optional-for-four-hours/>

⁴<https://www.theverge.com/2014/9/2/6098107/apple-denies-icloud-breach-celebrity-nude-photo-hack>

There are many cloud services on the market. Some of them provide free limited storage. Many cloud services claiming to be free use their customers' data for training their machine learning models and personalised advertisements. Security and privacy are provided usually by more expensive clouds. If the user interface of a cloud application is unclear, occasionally, some customers can get confused in the automatic synchronisation process and create duplicates by mistake.

2.1.2 Offline storage

Whereas using cloud service may evoke a lack of control, storing data locally is the opposite. Data stored in local storage are always accessible, even without an internet connection. Photos can be easily transferred between devices without necessity to download them first. The data are not shared with any third party, which imply total privacy. Only individual users are responsible for their data.

There are two primary options for storing photos offline: keeping the data on the phone or laptop or using external storage. While keeping all the data in the phone's or laptop's hard drive means immediate access, storing photos may take up too much space on a single device. At some point, the device will reach its maximum space. Taking many quality photos requires much space. Secondly, losing a phone or having a laptop stolen would result in the loss of all the precious data. Furthermore, users do not usually require immediate access to older photos. That is why many people tend to use external storage for photographs to archive them. However, the maintenance of multiple external storage devices can get complicated. Users should keep track of which devices are used for what type of data. Typically used are external hard drives, SD memory cards and USB flash drives. Less frequent are CDs/DVDs and NAS (Network-attached storage).

2.2 Organisation of photos

The process of organisation and management of photos varies. The main difference is the storage choice—whether the data is stored online or offline.

2.2.1 Managing photos offline

The way data is distributed throughout devices matters. Splitting data into folders is a good way to keep things organised. However, it is not always the best approach, especially for photos. Some images may fall into more than one category, which makes the folder system ineffective. Furthermore, a person can easily get lost in folders of multiple devices. Operating systems' default applications usually do not offer a view of data simultaneously from more than one device (in one window), which means that the content of every device can be managed only separately. There are applications available for a better way of displaying data on multiple devices, but they do not provide methods for sorting and organising photos.

Default tools of the operating system usually do not offer features such as editing photograph's metadata, adding tags to photos or filtering photos based on various criteria. For this reason, users tend to use third party photo organisation software. Offline applications differ based on which operating systems they are available for.

2.2.2 Managing photos online

It is not only important where the data is stored, but also how the images are organised within the Cloud and whether the customer can quickly sort and filter their photos. Some cloud services offer built-in image recognition and face recognition, which can speed up the search time for a specific photograph. The particular services are named and described in the following section.

2.2.3 Photo organising applications

Before looking for suitable software, the user should consider many criteria. Whether they are willing to invest, or they prefer free applications. Whether they are looking for something easy to use or more complex because sometimes they can unintentionally delete some photos without even realising, so they have to consider their computer skills. They should also take into consideration the amount of data that they have and plan to generate. Another important factor is how often they need to access their data.

Google Photos

The best known online photo manager and organiser is Google Photos⁵. It offers unlimited storage as long as uploaded photos do not exceed 16 megapixels resolution and videos are reduced to 1080p quality. If the conditions are not met, the storage is restricted to 15GB. It automatically uploads and backups every photo on a device when connected to the internet. It also supports image recognition, face recognition and sorting based on dates and places they were taken. Google Photos includes several editing options, such as colour adjustment, cropping, resizing and more. Customers can also create albums that can be shared publicly or privately with specific Google users. The storage can be extended up to 30 TB of data.

While many customers are satisfied with Google's services, many question their privacy policy⁶:

“...We also collect the content that you create, upload or receive from others when using our services. This includes things such as email you write and receive, photos and videos that you save, docs and spreadsheets you create and comments that you make on YouTube videos.”

However, Google has already stated that they will not use their users' photos for commercial or promotional purposes without obtaining explicit permission⁷. The user interface (first picture in Figure 2.2) has an intuitive timeline of photos on the right side and photos are sorted by the date by default. On the left side, there is the main menu, that does not take too much space, which means the focus is primarily on the photos in the middle. Pictures can be easily added to albums with a customisable name.

iCloud

iCloud Photos⁸ is an application designed for Apple devices. It has a shared library for all Apple devices synchronised with one account. It provides content recognition and organises

⁵<https://www.google.com/photos/about/>

⁶cited from <https://policies.google.com/privacy>

⁷<https://www.businessinsider.com.au/google-photos-privacy-2015-5>

⁸<https://support.apple.com/en-us/HT204264>

photos and videos into years, months and days. The photo collection is uploaded using Wi-Fi or cellular data. After signing up, the user automatically gains 5GB of free storage. There is also a possibility to buy more storage. It keeps all photos and videos in their original, high-resolution version.

Mylio

Mylio⁹ is an offline photo organiser and manager which integrates user's devices (both desktop and mobile) and online sources into one catalogue of photos. Mylio helps to manage and organise photos without any remote servers, thus ensuring privacy. As long as all devices are using the same Wi-Fi network and Mylio is running on all of them, photos are synchronised across all signed devices. Mylio does not display pictures on a timeline, like Google Photos. Instead, it displays a grid calendar with photos assigned to it. It offers a world map with markers on locations where imported photos were taken. Example of the user interface is in Figure 2.2 (the second picture). There is a free and paid version for Mac, Windows, iOS, and Android. The free version provides import of 25.000 photos to Mylio library and synchronisation between up to 3 devices. The paid version offers an unlimited amount of photos and devices and RAW photos editing for a subscription fee of \$10 a month.

Flickr

Flickr¹⁰ is an online photo management and sharing application with two main goals: to make content available to people of user's choosing, and to organise photos and videos. It is often used as a photography portfolio or as a blog because photos stored there can be marked as public as well. Example of the user interface is in Figure 2.2 (last picture). Under the image, there are views and comments on the left side and the image metadata on the right side. The EXIF data are displayed very clearly and attractively.

Free and the pro version is available. The free version is limited to 1.000 uploaded pictures. Pro version is \$5 per month and offers unlimited storage, advanced statistics and other features. One of the disadvantages is a collection of user's information (location, photo metadata (EXIF), device information, log information).

⁹www.mylio.com

¹⁰www.flickr.com

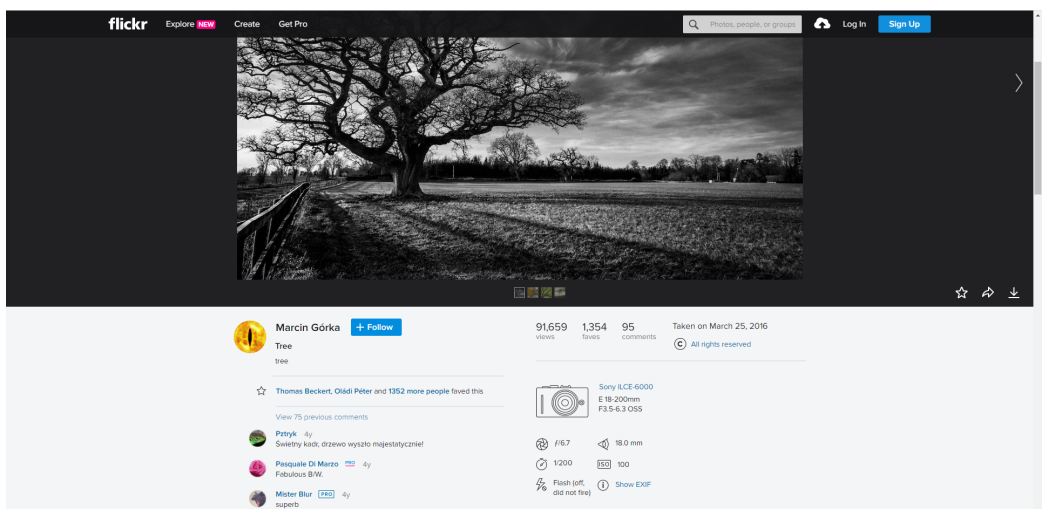
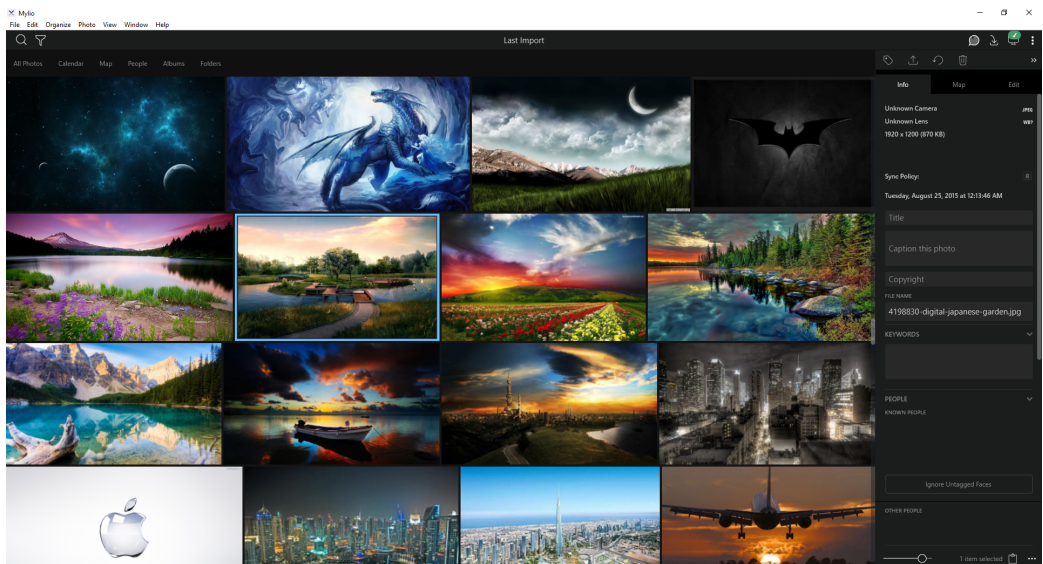
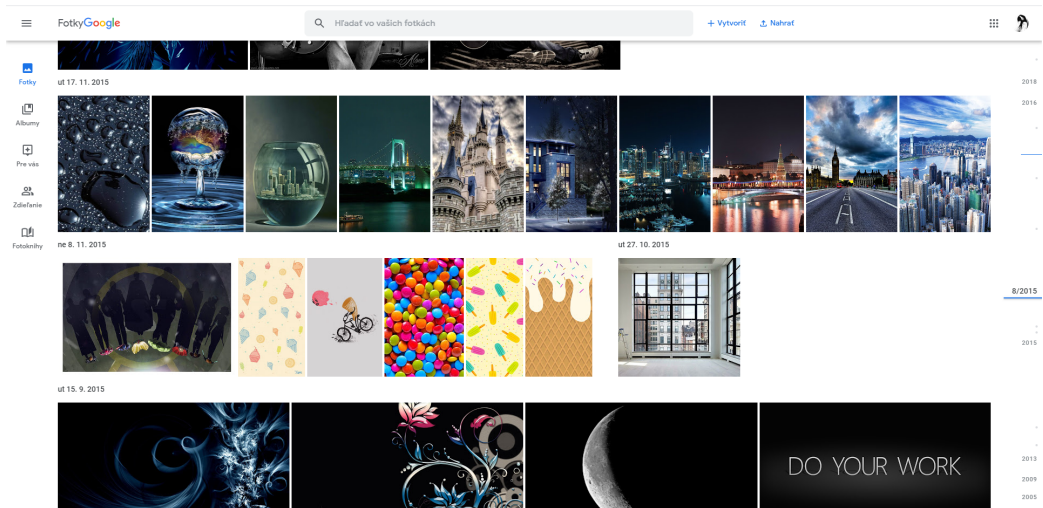


Figure 2.2: Examples of user interfaces of Google Photos(1), Mylio(2) and Flickr(3).

MAGIX Photo Manager

MAGIX¹¹ is a desktop application that allows users to browse and sort photos and videos easily. It supports a wide range of formats, filtering by tags, location, date, categories and rating. It sorts out poor shots, duplicates and blurry photos. It offers a light and dark theme and a possibility to create a travel route animation to show the most important highlights of the vacation. However, the process of creating a travel route is manual. The user interface is shown in the first picture of Figure 2.3. MAGIX is only available for Windows. It offers a free version with limited functions and paid full version for \$50.

Phototheca

Phototheca¹² is a photo organising, editing, and sharing software for a computer. This photo management software makes it easy to view, sort, organise, edit, and share thousands of digital photos and videos. Users can import from cameras, memory cards, hard drives, network shares and even iOS devices. Tag photos and videos with keywords, arrange them into albums, remove duplicates. Phototheca automatically recognises people and cats and gathers all photos of the same person into an album with a name. It supports duplicate detection and colour correction tools. However, it is only available for Windows. The user interface is usually considered outdated. An example is in Figure 2.3, picture in the middle. It comes in a free and paid version.

ACDSee Photo Studio

ACDSee is a desktop application for editing and management with face detection and facial recognition. Managing, labelling and organising of photos is mostly manual, and the application is available for Windows only. However, it offers many features, including photo editing. Example of the user interface is on the bottom picture in Figure 2.3.

Lightroom

Adobe's Lightroom¹³ is a professional photo editor that offers some organising features. It can modify metadata, work with RAW images, edit pictures, give them tags, ratings, labels and more. Unfortunately, there is no free plan—the application has a subscription fee of \$10 for a month.

¹¹<https://www.magix.com/us/free-download/photo-manager/>

¹²www.lunarship.com

¹³<https://www.adobe.com/products/photoshop-lightroom.html>

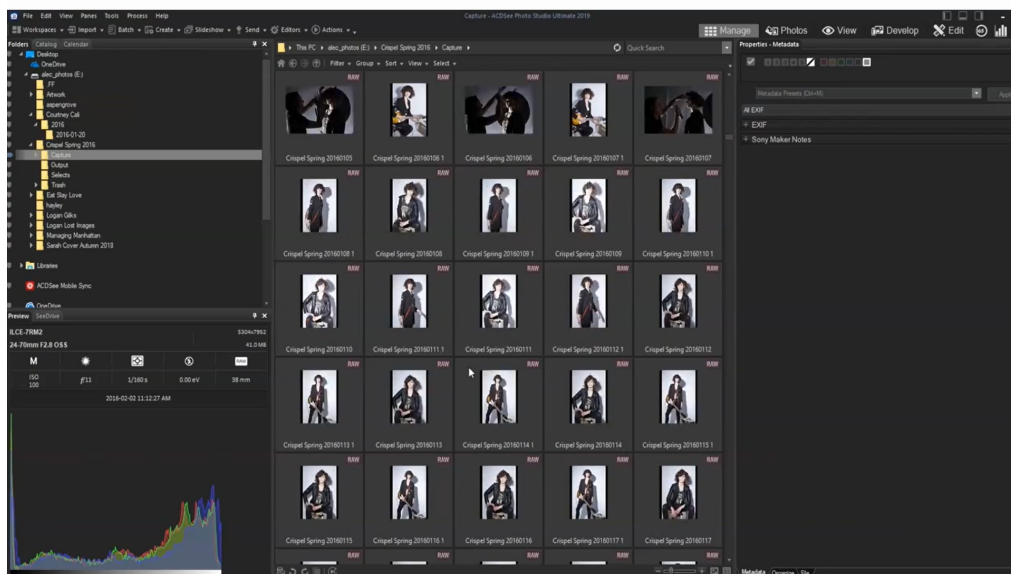
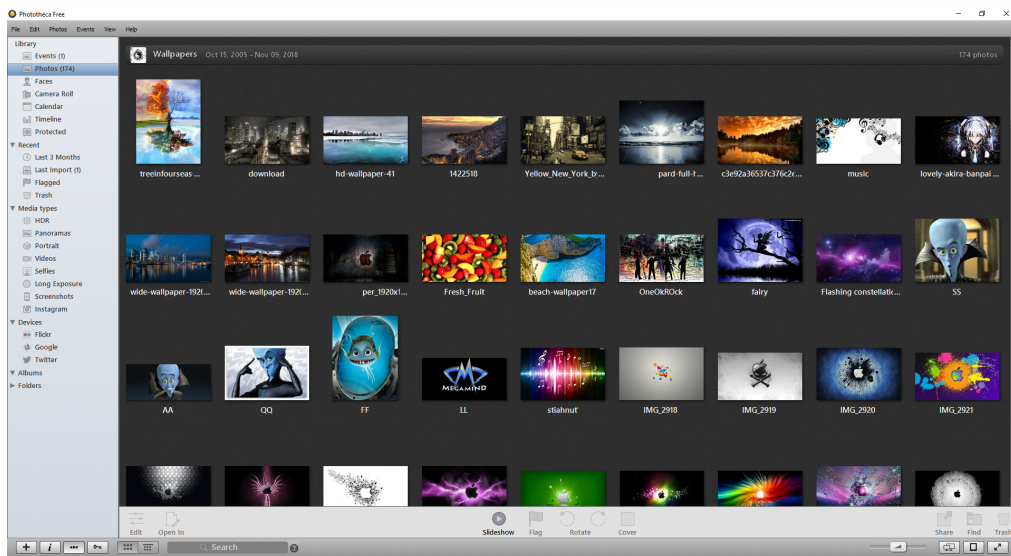


Figure 2.3: Examples of user interfaces of MAGIX Photo Manager (1), Phototheca (2) and ACDSee Photo Studio (3).

2.3 Detection of photo duplicates

When it comes to photos, a definition of word *duplicate* varies. Two images can be identical pixel-by-pixel, or one of them might have been resized, cropped, or colour-corrected. Furthermore, many cameras support multi-shot mode that can take bursts of images that are not identical, but very similar, all taken within a few seconds on the same location. It could be relatively easy to identify such „duplicates“ manually and delete them one by one when there are only a few of them in the library. But it is more common for people to have multiple copies of image folders across their devices, and to find and delete duplicates manually, in this case, would be most certainly very time-consuming and overwhelming.

To automatise this process, many tools are available. However, only a few of them support a wide range of image formats, and some of them detect only identical images, not similar. They also differ based on the operating system they are designed for.

2.3.1 Tools for duplicate detection

There is a group of tools that work in a similar manner. A user usually needs to import photos they want to scan. After some computation time, the list of found duplicates is presented with an option to mark which ones to delete. It might be convenient for some users. However, for the process of photo management and photo duplicates scan, there is a need for two different software solutions.

Duplicate Photos Fixer Pro

Duplicate Photos Fixer¹⁴ is usually number one in users' satisfaction evaluations for duplicate photos manager. It is available for Windows, Mac, Android and iOS. After adding files or folders the user desires to scan, the search engine scans photos and detects duplicates. It lets users review and mark the photos they want to delete before the deletion. The Fixer works with a wide range of image formats. It can identify duplicates based on certain parameters, such as quickly-shot series of photographs (Burst photos), slightly edited/touched-up images or similar photos taken within 24 hours. However, the free version has many limitations — removing only 15 files and blocked premium features. The Pro version price differs based on the OS, from \$7 for iOS to \$72 for Windows.

Other applications

Other applications worth mentioning are:

- Remo Duplicate Photos Remover¹⁵
- Duplicate Cleaner¹⁶
- Duplicate Photo Cleaner¹⁷

¹⁴<https://www.duplicatephotosfixer.com>

¹⁵<https://www.remosoftware.com/remo-duplicate-photos-remover>

¹⁶<https://www.digitalvolcano.co.uk/>

¹⁷<https://www.duplicatephotocleaner.com/>

2.3.2 Photo management with duplicate detection

The most convenient way for users to organise their photo library is a photo management application with built-in duplicate detection functions. Some of the previously mentioned applications from Section 2.2.3 offer duplicate detection.

Google Photos

Google Photos automatically detects an attempt of importing an image that is already in the library and will not upload it again. The duplicate will be detected even if the name of the file is changed. However, when the metadata of a photo is changed (for example date of creation), it does not detect duplicates. There is no feature as „find duplicates“ implemented in Google Photos.

MAGIX

MAGIX offers a feature „search my files and folders for duplicates“. But it detects only identical photos.

Phototheca

When importing files to Phototheca, a dialogue window will appear if duplicate was detected. It lets the user check what photos are duplicated and where the copies are located.

Mylio

When importing photos to Mylio, there is a checkbox with an option to „Exclude suspected duplicates“. With the checkbox turned on, Mylio will exclude identical images, even if they were renamed. Nevertheless, after even a small change in the photo’s metadata (for example the device it was taken with or rating of the photo), Mylio will not identify such an image as a duplicate.

Lightroom

In the process of importing to Lightroom, users can see which photos from the device are already uploaded. Uploading an identical photo is not possible. However, when the same picture is copied to another device and renamed, Lightroom does not recognise it as a duplicate.

Chapter 3

Photography metadata

Metadata, often referred to as “data about data”, provides interesting information that supplements the primary content of digital documents [2]. As stated in the document [13], metadata has become a powerful tool to organize and search through the growing libraries of image, audio and video content. This is especially important in the area of digital photography where, despite the increased quality and quantity of sensor elements, it is not currently practical to organize and query images based only on the millions of image pixels. Instead, it is the best option to use metadata properties that describe what the photo represents and where, when and how the image was taken.

In this chapter, the term photography metadata is described in detail, along with the journey from historical photo metadata to today’s standards.

3.1 History of photography management and metadata

Before the world’s digitization—and even during the early stages of it—metadata were the bits of information written, stamped, typed or printed on slide mounts, envelopes or the backs and borders of photographic prints, usually concerning the author of the photograph, date or a description of its content. It was a piece of information about the picture that must have been communicated through text because it was not evident in the picture itself.

As described in book [16], in the early days of digitization, photos were digitized only for transmission purposes and the digital version was of little consequence because it disappeared after transmission. Photographers transmitting the images typed the photo metadata (cutlines, bylines, dates, locations, etc.) on a strip of paper, attached the strip to the margin of the print and transmitted the verbal information as part of the photo. Essential data and the photos stayed together. Nevertheless, as digital technology developed, keeping digital and written data in one place became increasingly difficult.

3.2 Metadata today

In the modern world, photo metadata is key to protecting images’ copyright and licensing information online. It is also essential for managing digital assets. Detailed and accurate descriptions about images ensure they can be easily and efficiently retrieved via search, by

users or machine-readable code. This results in smoother workflow within organizations, more precise tracking of images, and increased licensing opportunities¹.

3.2.1 Metadata formats

There are three metadata formats widely used in the industry:

- EXIF – Exchangeable image file format
- IPTC-IIM – IPTC Information Interchange Model
- XMP – Extensible Metadata Platform

They are described in more depth in the article [18].

EXIF

A standard for image file formats, jointly managed by the Japan Electronics and Information Technology Industries Association (JEITA) and Camera and Imaging Products Association (CIPA). In particular, the EXIF image format defines a set of TIFF tags that describe photographic images and is widely used by digital cameras.

EXIF attributes contain for example:

- information about the camera – *Camera Model Name*, *Camera Serial Number*,
- photo’s exposure values – *Exposure Time*, *Shutter Speed Value*, *Exposure Index*,
- flash settings information – *Flash*, *Light Source*,
- resolution of the photo – *X Resolution*, *Y Resolution*, *Image Width*, *Image Height*, *Image Size*,
- GPS coordinates – *GPS Latitude*, *GPS Longitude*, *GPS Position*,

and other data, based on camera manufacturers. Example of metadata of a picture taken with a GoPro camera is shown in Figure 3.2. EXIF metadata can be found in TIFF, JPEG, and PSD files. [13]

IPTC-IIM

Multimedia metadata standard developed by International Press Telecommunications Council (a consortium of the world’s major news agencies, news publishers and news industry vendors). The first IPTC standard was text-only and defined to protect the interest of the telecommunications industry (1979).

Later, in 1991, a new standard, the “Information Interchange Model” (IIM), was created. IIM is an envelope format for transmitting news text documents and photos, and it defines the so-called “IPTC headers” which now exist in many photo files. Users can easily insert the data, specifically photo description, keywords, labels, etc. by software concerning photography metadata.

¹Cited from <https://iptc.org/standards/photo-metadata/>

XMP

Adobe’s Extensible Metadata Platform² is a labelling technology that allows embedding metadata into a file itself. XMP allows each software program or device along the way to add its own information to a digital resource, which can then be retained in the final digital file. XMP is serialized in XML and stored using a subset of the W3C Resource Description Framework (RDF). Therefore, customers can easily define their custom properties and namespaces to embed arbitrary information into the file.

3.2.2 Metadata properties

Every metadata format defines its metadata properties. In the simplest scenario, a given metadata property is only defined in a single metadata format. This is, for example, true for the rating property — it should always be read and written into the corresponding XMP (xmp:Rating) field. No further reconciliation is necessary. Also, there are other properties that are unique to the container and will not be reconciled amongst the other formats.

However, photography metadata can be defined in more than one format. Dealing with more than one metadata format makes it challenging to determine the correct behaviour for handling the particular property values.

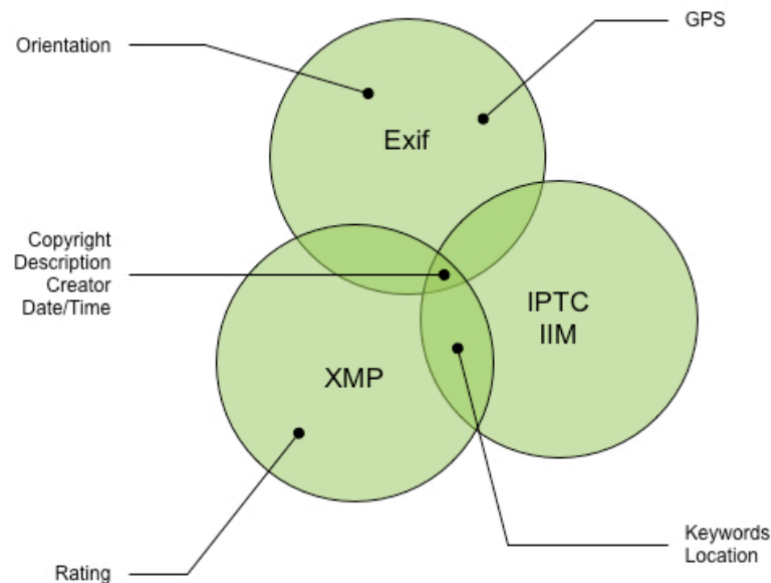


Figure 3.1: Simplified view of metadata defined in more than one format [13]

Based on Figure 3.1, there are only a few properties defined in more than one metadata format. Only four of them are available in EXIF, IPTC-IIM and XMP (Copyright, Description, Creator and Date/Time). To ensure interoperability between software solutions, there is a metadata reconciliation guidance on how to handle the different metadata properties in the context of the actor/role definitions described in section 2 of chapter 3 of this document [13].

²<https://www.adobe.com/products/xmp.html>

⁴<https://exiftool.org/>

```

C:\Users\Kiwinka> exiftool D:\Pictures\GOPR0016.JPG
ExifTool Version Number      : 11.84
File Name                    : GOPR0016.JPG
Directory                    : D:/Pictures
File Size                    : 4.5 MB
File Modification Date/Time   : 2019:03:25 14:22:37+01:00
File Access Date/Time        : 2019:03:26 17:15:42+01:00
File Creation Date/Time      : 2019:03:25 14:22:37+01:00
File Permissions              : rw-rw-rw-
File Type                    : JPEG
File Type Extension          : jpg
MIME Type                    : image/jpeg
Exif Byte Order               : Big-endian (Motorola, MM)
Image Description             : DCIM\100GOPRO\GOPR0016.JPG
Make                         : GoPro
Orientation                   : Horizontal (normal)
X Resolution                  : 72
Y Resolution                  : 72
Resolution Unit               : inches
Software                     : HD7.01.01.70.00
Modify Date                   : 2019:03:25 13:22:37
Exposure Time                 : 1/8
F Number                      : 2.8
ISO                           : 200
Exif Version                  : 0221
Date/Time Original            : 2019:03:24 19:25:25
Create Date                   : 2019:03:24 19:25:25
Shutter Speed Value           : 1/8
Aperture Value                : 2.8
Max Aperture Value           : 2.8
Subject Distance              : 0 m
Metering Mode                 : Average
Light Source                  : Unknown
Flash                         : No flash function
Focal Length                  : 3.0 mm
Exif Image Width              : 4000
Exif Image Height             : 3000
Interoperability Version     : 0100
Exposure Index                : 2147483595
File Source                   : Digital Camera
Scene Type                    : Directly photographed
Custom Rendered               : Normal
Exposure Mode                 : Auto
Digital Zoom Ratio            : 1
Focal Length In 35mm Format   : 15 mm
Scene Capture Type            : Landscape
Gain Control                  : None
Contrast                      : Normal
Saturation                    : Normal
Serial Number                 : C3281325991476
GPS Latitude Ref              : North
GPS Longitude Ref             : West
GPS Altitude Ref              : Above Sea Level
GPS Time Stamp                : 19:25:25
GPS Date Stamp                : 2019:03:24
Compression                   : JPEG (old-style)
Thumbnail Offset              : 44032
Thumbnail Length              : 18379
Creator Tool                  : HD7.01.01.70.00
Metadata Date                 : 2019:03:25 13:22:37Z
Photographic Sensitivity     : 200
Video Frame Rate              : 0.000000
Video Frame Size W            : 4000
Video Frame Size H            : 3000
Video Frame Size Unit        : pixel
Coded Character Set           : UTF8
Application Record Version    : 4
Date Created                  : 2019:03:24
Time Created                  : 19:25:25
MP Image Flags                : Dependent child image
MP Image Format                : JPEG
MP Image Type                 : Large Thumbnail (VGA equivalent)
Total Frames                  : 1
Device Name                   : Photo Global Settings
Firmware Version              : HD7.01.01.70.00
Camera Serial Number          : C3281325991476
Camera Model Name             : HERO7 Black
Auto Rotation                 : Up
Digital Zoom                  : No
Pro Tune                      : On
White Balance                 : 2800K
Sharpness                     : HIGH
Color Mode                    : FLAT
Auto ISO Max                  : 200
Auto ISO Min                  : 100
Rate                          : 4_1SEC
Photo Resolution              : 12MP_W
HDR Setting                   : OFF
Image Width                   : 4000
Image Height                  : 3000
Encoding Process              : Baseline DCT, Huffman coding
Bits Per Sample               : 8
Color Components              : 3
Aperture                      : 2.8
Image Size                    : 4000x3000
Megapixels                    : 12.0
Scale Factor To 35 mm Equivalent : 5.0
Shutter Speed                 : 1/8
GPS Altitude                  : 55.5 m Above Sea Level
GPS Date/Time                 : 2019:03:24 19:25:25Z
GPS Latitude                  : 38 deg 38' 36.84" N
GPS Longitude                  : 9 deg 14' 26.28" W
Date/Time Created             : 2019:03:24 19:25:25
Circle Of Confusion           : 0.006 mm
Field Of View                  : 100.4 deg
GPS Position                  : 38 deg 38' 36.84" N, 9 deg 14' 26.28" W
Hyperfocal Distance           : 0.53 m
Light Value                    : 5.0

```

Figure 3.2: Example of a photograph's metadata extracted by using Phil Harvey's Exiftool⁴.

Chapter 4

Principles of usable multi-platform application design

This chapter is focused on describing the principles of developing a multi-platform application that is considered usable. In the first section, the principles of multi-platform development are listed and described. In the second section, there is a list of possible solutions for developing a multi-platform desktop application with their main advantages and disadvantages portrayed. In the third section, the term *usability* is defined, and the steps of developing a usable user interface design are illustrated.

4.1 Principles of multi-platform development

When developing a multi-platform application, there are some rules to keep in mind. Firstly, operating systems use various file systems. When working with file systems, the application should detect which one is in use and have a valid approach implemented. Also, a path to system data is different for every platform. While Microsoft Windows keeps information about applications in the *Appdata* folder, OS X usually uses *~/Library/Application Support* and Linux equivalent depends on the specific application – for instance, Chrome data is usually in *~/.Chrome* folder.

The second issue concerning different operating systems is the title bar of their windows. The title bar¹ is a horizontal bar located at the top of a window in a GUI. It displays the title of the software, name of the current document, or other text identifying the contents of that window. The placement of minimise, resize, and close buttons varies. While Windows has these buttons always on the right side, OS X keeps them on the left side. On Linux, placement of them and window GUI is customisable and depends on the chosen desktop environment. The application should try to keep them placed where the user expects them to be.

Another problem may appear when the application needs to run third-party software. Each operating system has its own way of launching an application within an application. Last but not least, detailed and comprehensive testing is crucial, on as many platforms and their versions as possible.

¹<https://www.computerhope.com/jargon/t/titlebar.htm>

4.2 Development of multi-platform applications

There are different ways of approaching the problem of writing a cross-platform application program. One of them is to create multiple versions of the same program in different source trees. This means that the Microsoft Windows version of a program would have one set of source code files while the Linux or Mac version might have another. Even though this is a straightforward approach to the problem, it has the potential to be noticeably more expensive in development cost and development time. Furthermore, creating two or more than two different programs that are supposed to behave similarly may lead to more difficulties, because the two different source trees may have different programmers and possible problems with bug tracking and fixing.

To be platform-independent is the desired quality of many applications. Modern development offers multiple methods to simplify the implementation process. One of them is called cross-platform software development. This development process allows writing a single code-base to create applications for multiple platforms. There are many options of frameworks or available tools to choose from.

4.2.1 Software platform possibilities

In this section, the alternatives of frameworks are listed, and their advantages and disadvantages are described. There are many possible options available, for example Haxe, Electron, NW.js, 8th, Kivy, Enyo and more. In this section, Haxe, Electron and NW.js are described.

Haxe

Haxe² is an open-source high-level strictly-typed programming language with a fast optimizing cross-compiler. It allows compilation of programs, written using an ECMAScript-oriented syntax, to multiple target languages. Currently, there are nine supported target languages, including JavaScript, PHP, Python, C++, C# and Java. The first version came out in 2006.

Because the Haxe Language can compile to many different platforms, it is useful in a wide variety of domains. It is popular with game creators because it is fast, has many useful libraries, and can target iOS, Android, Web and Desktop easily.

Haxe can be used for web development, for client-side as well as server-side. The most popular libraries are **Haxe JS Kit** and **Haxe React**. Furthermore, it is popular for desktop applications as well. When aiming for apps with a native look, users can use the HaxeUI library, which uses several ways to build native UI for each platform. There is support for WxWidgets when targeting C++, which provides a real native look that lets developers create applications for Windows, Mac OS X, Linux. Moreover, Haxe offers full access to JavaScript/Node APIs.

However, there are several disadvantages, one of them being not very detailed documentation and only a few books on Haxe, many of which are outdated. Many users claimed to have troubles with compatibility when adding third-party APIs.

²www.haxe.org

Electron

Initially built for the Atom code editor, Electron is an open-source framework developed by GitHub. It is based on Chromium core and Node.js, which allows developers to write cross-platform desktop user interfaces with popular web technologies: HTML, CSS and JavaScript. It provides all advantages of a native desktop application, such as access to the file system or the use of system notifications.

Electron has two separate contexts (main process and render process) that are completely separated. The script that runs in the main process can display a GUI by creating web pages. An Electron app always has only one main process. Each web page in Electron runs in its own process, which is called the renderer process. These processes can communicate in several ways, for example using `ipcRenderer` and `ipcMain` modules for sending messages. Electron architecture is described in more depth in Section 7.1.1.

Front-end relies entirely on web standards and technologies, which makes it possible to integrate modern front-end frameworks, such as ReactJS³ or Angular⁴. For desktop applications, it provides various core functionalities, such as automatic updates, crash reporting, installer creator, debugging and system-specific features. Moreover, it offers detailed documentation [8] and has a large community. Another advantage is that applications written using Electron are easily transferable to the browser as a result of using web technologies. One of the disadvantages is that there is no built-in Model-View-Controller architecture provided by Electron, and platforms for Chrome are not fully supported yet.

NW.js

NW.js (previously known as Node-WebKit) allows developing cross-platform software using modern web technologies like HTML, CSS3 and JavaScript, including WebGL. As well as Electron, NW.js is based on Chromium and Node.js. It provides support for all Node.js APIs and most of the third-party modules. It calls Node.js modules directly from DOM and Web Workers. It was created in the Intel Open Source Technology Center.

NW.js has a generous list of demo applications and video games and provides great community support with easily searchable answers. Some of its functions are more feature-rich and mature than those of Electron. However, some features that are available in Electron, such as auto-updater and crash-reporting, do not come built-in with NW.js. That said, building with NW.js requires more efforts and extra modules. Unlike in Electron, where its two contexts are separated, NW.js offers an option of separate contexts and mixed contexts. More information is provided in the documentation of NW.js [14].

³<https://reactjs.org/>

⁴<https://angular.io/>

4.3 Design of usable application

The quality that every application interface is trying to reach is being usable. But the term *usability* could be understood differently.

4.3.1 Terminology

ISO defines the term *usability* as „extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use“ in the international standard ISO 9241-210 [17]. It consists of the 6 subcharacteristics – appropriateness recognisability, learnability, operability, user error protection, user interface aesthetics and accessibility.

Usable design is also focused on responsivity. As Jeff Johnson writes in the book *Design with the Mind in Mind* [11]:

To be perceived by users as responsive, interactive software must follow these guidelines:

- Acknowledge user actions instantly, even if returning the answer will take time; preserve users' perception of cause and effect.
- Let users know when the software is busy and when it is not.
- Free users to do other things while waiting for a function to finish.
- Animate movement smoothly and clearly.
- Allow users to abort (cancel) lengthy operations they do not want.
- Allow users to judge how much time lengthy operations will take.
- Do its best to let users set their own work pace.

4.3.2 Used techniques

To fulfil the requirements for a usable user interface, the interaction with product users is inevitable. There exist a variety of methods and techniques for analysis and evaluation of user requirements. Jenny Preece in book *Interaction Design* [15] describes five methods of data gathering for requirements: interviews, questionnaires, observation, researching similar products and studying documentation. Another used technique to gather information and empathise with users is creating personas. To visualise and summarise the data, the use-case diagram can be used.

Questionnaire, interviews and documentation studying

A questionnaire is a simple technique for gathering information from respondents. More personal approach, however, is an interview. According to [15], there are two types of interviews – structured and unstructured ones. Structured interviews consist of predetermined questions, while in unstructured interviews „the questions posed by the interviewer are open, meaning that there is no particular expectation about the format or content of answers. A benefit of unstructured interviews is that they generate rich data that is often interrelated and complex, i.e. data that gives a deep understanding of the topic.“

Later, the author states „*Studying documentation is good for understanding legislation and getting some background information on the work. It also does not involve stakeholder*

time, which is a limiting factor on the other techniques.“. The documentation about photography metadata [13] was studied, and the main information was described in Chapter 3.

Personas

Personas are fictional characters, which are created based on research in order to represent types of users that will use an application, product or service. Creating a persona is an important step to understanding the mindsets of potential customers. It helps designers to gain a similar perspective and identify with the users and to build empathy. A persona is usually presented as an individual person with a name, face, and a backstory.

„Personas usually explain that person’s needs, values, goals, frustrations, and desires. Personas are made as human as possible to further enhance the sense that this is a real person with a messy life and quirky ways of coping with very recognisable human situations.“

- Leah Buley, *The User Experience Team of One* [5]

Use-case diagram

An important step before designing software architecture is to define all the requirements and assign them to roles. One of the options to do so is to create a use-case diagram. The purpose of a use case diagram in Unified Modeling Language (UML) is to demonstrate the different ways that a user might interact with a system. In the UML, a use-case diagram can summarise the details of the system’s users (in this context known as actors) and their interactions with the system [7]. An effective use case diagram can help to see different scenarios or to help clarify the goals of the system. As stated in [3], actors can communicate with the system for many reasons, including:

- To start a use case. Use cases are always started by actors.
- To ask for some data stored in the system.
- To change the data stored in the system by means of a dialogue with the system.
- To report that something has happened in the system’s surroundings that the system should be aware of.

The use cases, actors, and their associations can be shown on use-case diagrams. The diagrams are used to convey who the actors are, what the use cases are, and how they are related.

Testing

As written in the book *Interaction Design* [15], collecting data about users’ performance on predefined tasks is a central component of usability testing. There are numerous methods for the data collection, for example, video recording of the users including their facial expressions and logged keystrokes and mouse movements, asking participants to think out loud while carrying out tasks or a satisfaction questionnaire to find out how users actually feel about using the product. Structured or semi-structured interviews may also be conducted with users to collect additional information about what they liked and did not like

about the product. Usually, the tasks that are given to users include searching for information, navigating through the application. Sometimes, their performance is measured, for example, the time to complete a task, a number of errors per task or the success rate of completing the task.

Chapter 5

Problem analysis

In this chapter, problems concerning photo management are analysed. In the first section, the user needs and requirements are described using three different methods. After that, the advantages and disadvantages of existing solutions are listed. In the last section, there is a summary of what was being said and a conclusion that leads to the solution design of a new application.

5.1 User requirements

„A great product experience starts with a good understanding of your users. Not only do you want to know who they are, but you want to dive deeper into their motivations, fears, mentality, and behaviour. But how do we know what our users really want?“

- Chris Bank, Jerry Cao, *Guide to UX Design Process and Documentation* [6]

Well defined and understood user requirements can help make decisions about design and development. Leah Buley wrote that connecting with users is so essential that it's one of the core tenets of user experience: design products *for* and *with* users [5]. Later he explains that he prefers calling users just *people* because it reminds him that it's the people all around us that we're designing for.

5.1.1 Question form

The first method I used for user analysis was a question form containing nine simple questions to specify and verify my assumptions. There was a need to determine whether the planned photography management application is actually wanted or needed. The form was published using social media in groups of international students who keep photos from all their travels and also in groups of high school and university students. It was answered by 65 people. The questionnaire is in Appendix A. The main outcomes were:

- 97% of asked people claimed to take pictures using their smartphone, of which 25% also used a camera.
- 29% of asked do not back up their photography data at all, 33% use external hard drives, USBs, SD cards or a computer for backups and 38% use cloud services such as Google Photos or iCloud.

- 60% of asked claimed to have troubles with duplicates in their storage, and 84% of them delete duplicates manually.
- Some of the asked wrote that they keep some duplicates intentionally to store them into more than one category and search for them more easily.
- Most of the asked responded that they are concerned about their privacy when using cloud services, or they do not use them.
- A significant part of respondents answered that they would like to organise their photos if they had the time because they consider the process excessively time-consuming.

The main conclusion of the questionnaire was that asked people do not have a comprehensive way of storing and managing photos which are stored on many devices.

5.1.2 Interviews

Next approach was more personal and oriented towards users as individuals. I spent a few days interviewing various people that could become potential users of the final product. Interviewed people can be divided into two main groups. The first is a group of people aged between 18 and 25, mostly college students or young adults. The second group is made up of slightly older or middle-aged adults, who collected a large number of digital photos during their life so far and they are not as technically proficient to be working with complicated software. During the interview, I focused on these issues:

- What their current photo storing and organisation system looks like.
- How much data they store and how many devices they use.
- Which formats of photos they mostly work with.
- Whether they use device synchronisation.
- How often people go through their old photos.

Thanks to the interviews and discussions, I gained a lot of useful information and a wide perspective on different users' problems.

Conclusions

Some people tried to keep their photo library tidy and organised from the start—from their first phone with a camera or their first digital photo camera. Photos divided into albums are clearer, however, albums tend to grow and are challenging to maintain. It would be very time consuming for them to rebuild the system of albums that was kept before. It is also hard to find the motivation to sort photos into albums. Moreover, if a person spent many hours on organising photos into albums on a mobile device and after some time they change a phone brand, it might not be compatible. The new phone will ignore the hierarchy of photos and would store them differently. All the work that was put into it would be useless.

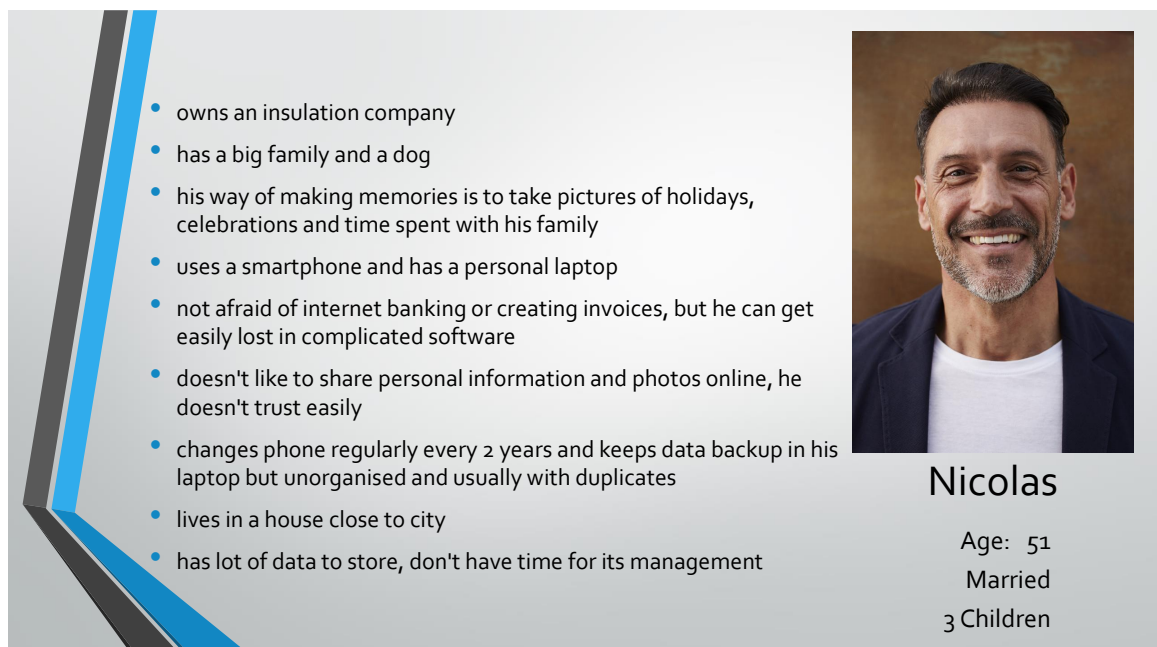
Even though not always organised, users try to keep their photos in one place, usually in a computer or an external hard drive. Many people that use Apple devices, as well as devices with a different operating system, had issues with compatibility. Most of the interviewed

claimed that they try to back up their data regularly. Many of them encountered a problem with duplicates. For example, a person backups their photos from phone to computer, and keeps photos in both places. They organise their photos in a computer into albums. However, when they back up their photos next time, they do not know how to store them into created folders, and they get lost in all the data. This leads to a lack of motivation for users to organise photos because it would take them too much time. As a result, they just copy all the photos to the computer again, causing duplicates and using more space of data storage. The amount of data people store differs substantially. Some keep only a few gigabytes, while people with better quality cameras can store up to hundreds of gigabytes.

Based on the questionnaire and the interviews, it was found that the most used format of photos among asked users is JPG. However, people who focus more on photography and photo editing answered that they keep their photos in RAW format, which requires more space. It is not just the photos people take with their cameras, but also photos they download from the internet or screenshots from their computer or phone. These pictures are usually in PNG or GIF format. One of the asked questions was how often people go through their old photos, and the results were unexpected. Usually, people often go back only to their recent photos. Pictures taken more than a year before present are visited only a few times a year, sometimes even less often. Among the reasons why people revisit their old pictures are refreshing their memory of a particular event or finding a specific picture.

5.1.3 Personas

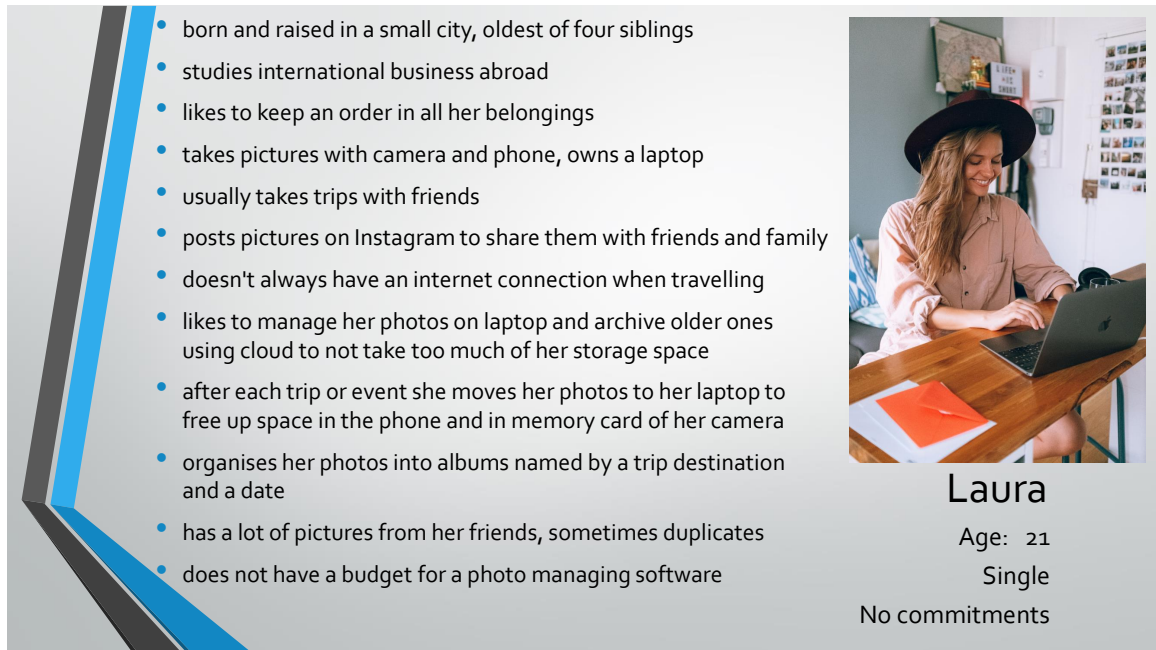
Based on the interviews, I created two personas to represent two main groups of users. First of them is Nicolas (described in Figure 5.1), and the second one is Laura (in Figure 5.2). They are different, but their goal concerning photo organisation is the same, as well as the problems they encounter.



- owns an insulation company
- has a big family and a dog
- his way of making memories is to take pictures of holidays, celebrations and time spent with his family
- uses a smartphone and has a personal laptop
- not afraid of internet banking or creating invoices, but he can get easily lost in complicated software
- doesn't like to share personal information and photos online, he doesn't trust easily
- changes phone regularly every 2 years and keeps data backup in his laptop but unorganised and usually with duplicates
- lives in a house close to city
- has lot of data to store, don't have time for its management

Nicolas
Age: 51
Married
3 Children

Figure 5.1: Persona Nicolas as a representation of older users, with not organised way of photo management and no professionally advanced computer skills.



- born and raised in a small city, oldest of four siblings
- studies international business abroad
- likes to keep an order in all her belongings
- takes pictures with camera and phone, owns a laptop
- usually takes trips with friends
- posts pictures on Instagram to share them with friends and family
- doesn't always have an internet connection when travelling
- likes to manage her photos on laptop and archive older ones using cloud to not take too much of her storage space
- after each trip or event she moves her photos to her laptop to free up space in the phone and in memory card of her camera
- organises her photos into albums named by a trip destination and a date
- has a lot of pictures from her friends, sometimes duplicates
- does not have a budget for a photo managing software

Laura
 Age: 21
 Single
 No commitments

Figure 5.2: Persona Laura as a representation of younger users, with a lot of photos from all around the world and no persistent photo management, but more advanced computer skills.

Nicolas represents older users. They prefer clean and intuitive design. They collected a huge amount of photos over their life but are too busy to manage them manually. Another required feature is duplicate detection. They usually take photos with only a few devices, but store photos across many.

Laura, on the other hand, represents younger users who are familiar with more complicated software. They usually take photos with more than one device and have many shared photos with friends. They require a way to organise photos across multiple devices. They are not equipped with constant access to the internet. They have gathered many photos from across the world and would probably appreciate a map view with the locations of their pictures.

5.1.4 System requirements

Based on everything that has been said, the main system requirements are presented. Users value their privacy. They appreciate automatic device synchronisation and online backing up, but only of photos of their choosing. However, the easiest way of using cloud services (described in Section 2.2.3) is to upload all photos automatically. That means that users would have to organise and sort out their photos before uploading them to the cloud, which often leads to a decrease in efficiency.

The interviewed people use multiple devices with a variety of operating systems. Commonly, the one device connecting most of their photos is a laptop or a computer. In other words, the most suitable application for photography management should be available for computer operating systems such as Windows, macOS, as well as Linux distributions. The photo managing software should provide an option for detecting and deleting duplicates, including very similar photos. Moreover, a lot of asked people would appreciate

the facial recognition of people displayed in the photos. A map displaying photos taken on different places would be an excellent feature for many of the asked users.

The most used method of photo management is to divide photos into custom albums, so the system should not skip this feature. Also, it should allow users to label their pictures with tags, names and descriptions for easier search. Searching for a particular photo in a library should not take too much time, and filtering pictures should be intuitive. Interviewed people suggested filtering photos based on date, location, tags, faces, albums, occasions, type (landscape or portrait), a device they were taken with etc.

5.2 Existing solutions

The existing solutions described in Section 2.2.3 cover several user requirements. Google Photos is the best solution for people who have constant internet access, are not concerned about their private pictures and do not mind quality restrictions. For them, it is a free and available powerful tool to help them automatically organise their photo library and synchronise their devices. When using an Apple device, the best solution to organise photos is using iCloud and iPhotos applications. However, only 5 GB of cloud data is for free. Users that do not wish to pay for extra storage usually choose different software.

People who do not prefer to upload their photos to a cloud can choose from a variety of options depending on their operating system. The most popular ones for Windows are ACDSee Photo Studio, Phototheca and MAGIX Photo Manager. They all come in free and paid versions, however, the free version is very limited in all cases. The most interesting features were listed in Section 2.2.3. The ACDSee Photo Studio seems chaotic for users at first glance. It offers many features, nonetheless, it is overwhelming for a user concerned only about photo management. It has a steep learning curve, and since most of the functions are manual, it is taking a significant amount of time to organise photos in this program. However, when the user is already familiar with the environment, it is convenient for them to continue using this application. The Phototheca application has an outdated user interface and lacks the ability to show pictures on a map. On the other hand, it offers a clear view of the data distribution in different devices. Also, it offers media type division (left side panel in the middle picture of Figure 2.3).

The best option for professional photographers who want to work with RAW formats of photos and powerful editing options is Lightroom. However, this solution is not recommended for a layman, for the reason that it is expensive for essential photo organising purposes.

The last existing solution described in this paper is Mylio. It is an offline tool, but it requires an online user's registration. It has a modern user interface and offers many features, including displaying photos on a map (using OpenStreetMap¹). Even though the interface looks quite clear and intuitive on a first glance, it is actually quite complex, and it takes time to learn its mechanisms. Another downside is that the application is not available for Linux and that the free version is limited. Furthermore, it lacks a feature for searching for duplicates.

¹www.openstreetmap.org

5.3 Summary

Based on all the user requirements listed in this chapter and their mapping to already existing solutions for photo management, there is an opportunity for a new application. Users lack an application that can work on multiple operating systems, including Linux. They want an application that is private and free, with an opportunity to manage their own data the way they desire with absolute control of it. Many users would prefer customisable UI or a possibility of light and dark mode. They would appreciate seeing a world map with markers of places where they took their photos, but the map should not have too many details. The default order of the photos should be in a simple looking and easy to navigate timeline.

One of the main desires is to be able to edit metadata of photos, but not each photo separately, but to be able to change an attribute of metadata on multiple selected photos at the same time. However, the most important feature is to be able to clearly see which photos are stored where and on which device, with the option to manage and organise them like they are all in one folder. They demand to be able to tag people on their photos with a possibility of automatic face recognition. They lack automatic duplicate detection and the option to search through their library and find similar or duplicate photos. They want to be able to create albums with custom names, descriptions and thumbnail photos.

All these features cannot overload the design of the application because users prefer simple looking and intuitive applications.

Chapter 6

Solution design

In this chapter, the design of the application is described. The first section is focused on the analysis of the data structure, and the data solution is presented. The second section is focused on the UI design of the final application. The use-case diagram is created, and the application is divided into sections. Each section is described, and the user interface design is presented.

6.1 Data structure design

Photography metadata contains a lot of information. Most of them are not needed in case of managing and organising them. However, some are crucial. The application needs to store a piece of information about each of the photos, to enable the filtering and sorting based on this data. Particularly name, path, date, resolution, location, a device it was taken with, a device where it is stored, size, tags and additional data that users need. Each photo is provided with an ID. Additionally, the miniature of the photo is required for faster loading of the images.

6.1.1 Data analysis

Photos in the library need to be divided by the device they are stored in. Each device is provided with the number of photos it currently holds. Photos need to keep their metadata and tags. Every photo has a unique ID and its data. The global number of images in the library is required as well.

6.1.2 Final data structure

The design of the data structure is demonstrated using an Entity-relationship diagram, displayed in Figure 6.1. Each device in the library has a unique ID, and the count of photos it currently holds is computed. It can contain zero to an unlimited amount of images. Each photo is distinguishable by its ID and contains all the required information. Additionally, tags can be added to it, and any number of people can be tagged on each photograph.

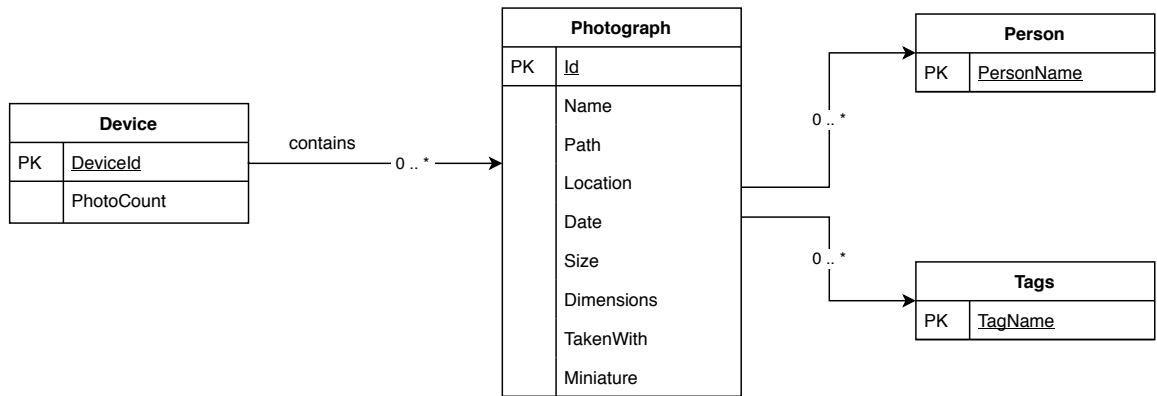


Figure 6.1: Entity-relationship model of the application's data structure design.

6.2 UI design

In this section, the user interface design is described. First, based on the requirements from the previous chapters, the use-case diagram is created. Then the structure of the user interface and the decisions concerning the design are described.

6.2.1 Use-case diagram

In the presented use case diagram, there is only one figuring actor—the user. The diagram was created for UI design purposes, not for implementation. The final use case diagram is in Figure 6.2.

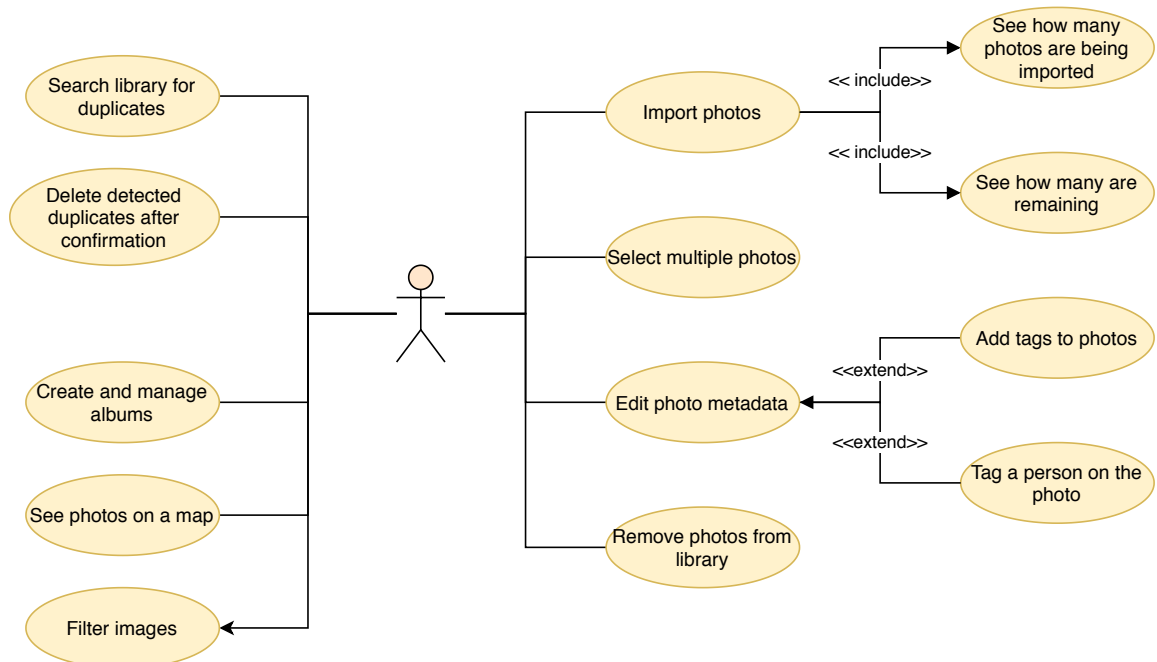


Figure 6.2: The final use case diagram.

It contains the main actions that the user can execute in the application. The first use case any customer will face is importing photos into the library. To maintain the application's responsiveness, the remaining time and number of imported photos are shown. Then, the user is able to manage his photos—to edit their metadata, create and manage albums, display them on a map or remove them. Another use case is to filter images based on various criteria. Detecting duplicates in the library is a separate use case, and it should be divided from other functionalities. User needs to see detected duplicates before he can remove them, or choose which ones to remove.

6.2.2 Structure of the application UI

Based on the application requirements and the use-case diagram, the application should have a variety of functionalities. In order to maintain clarity, the application is divided into sections, which all have their own context view and are accessible from the main menu. The sections' design and functionalities are described below.

Navigation menu

To access all the functionalities and quickly orientate among them, the application is provided with the navigation menu (sketch is presented in Figure 6.3). The menu is located on the top, to provide more space in width. It should contain only the most important links, thus avoiding overloading the user.

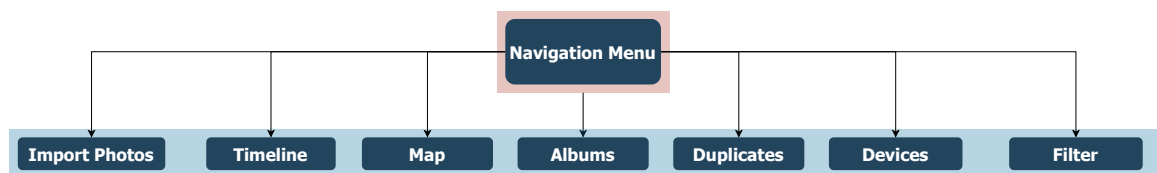


Figure 6.3: Sketch of navigation menu and layout of elements.

Import of Photos

Importing photos into the library is the first step in using the application. User is able to add more photos at any time—therefore the import button should be always visible and not difficult to locate. That is the reason why it has been placed in the navigation menu.

While photos are being imported, a user should be able to see how much time remains and how many photos were successfully imported into the library. Also, they should be able to cancel the operation at any time in the process.

Timeline

In the Timeline view, photos from the library are sorted by the date and displayed chronologically. The presented data described in Chapter 4 shows that people focus more on recent photos than older ones, so the newest photos in the Timeline view are placed on the top.

However, since there can be thousands of photos taken in a given year, letting users scroll down to see different years implies impracticality. A possible solution could be to display only a preview of one year's photos and click on the year to display all of them. To determine when the photo was taken, photos taken in one year could also be divided into months and be displayed accordingly.

Photos displayed in the timeline should be clickable and selectable. After choosing a particular photo, the information about it should be displayed, namely its size, name, tags and other metadata. The displayed data are meant to be editable. To edit multiple photographs at once, they need to be selectable. Users are used to selecting multiple items using keyboard shortcuts, namely *Shift* for selecting a line of items, or *Ctrl* to select each item individually. To maintain usability, there should not be new practises introduced. On the other hand, benefit from what users are already used to improves the application's learnability and operability. Selected items are distinguished by an outline that contrasts with the background. The first design of the Timeline view is displayed in Figure 6.4.

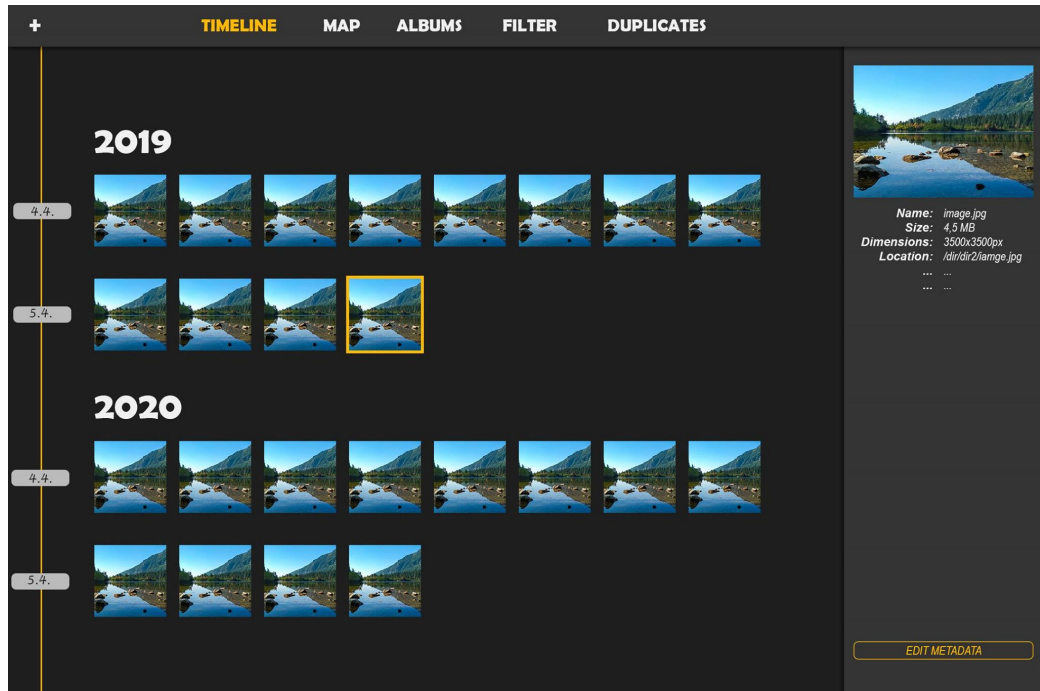


Figure 6.4: Proposed application design of the Timeline view (designed using Adobe Photoshop).

Map

Seeing photos displayed in locations where they were taken was one of the requirements by users. In the Map view, there is a zoomable map of the world with marks representing the photos. The colour scheme of the map should not be too intense or expressive since the focus should be more on the photos. Nonetheless, having too many photo markers in one location would look confusing. That is the reason why their clustering was suggested. While zoomed out, the user can see how many photos were taken on specific locations. When zooming in, the marker will de-cluster into specific positions with the correct photo marker. The photo is displayed on the map in such a way, that its location is in the middle of the image. The layout design is in Figure 6.5.

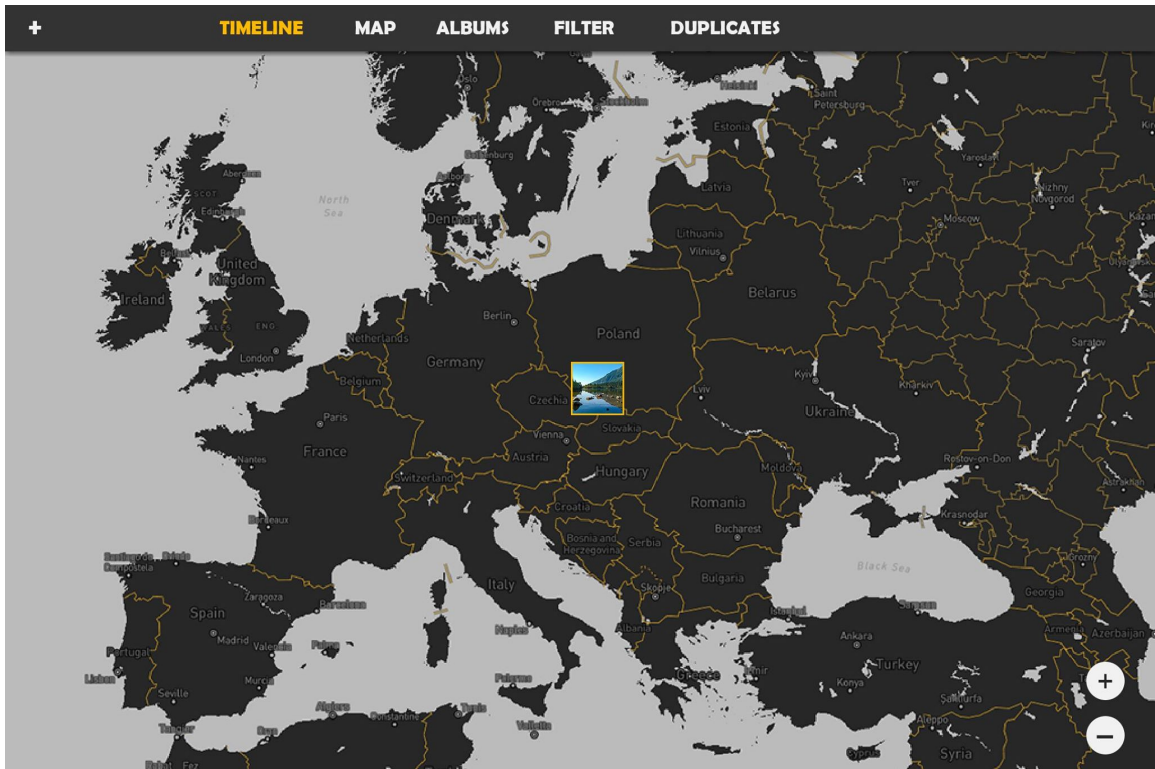


Figure 6.5: Proposed application design of the Map view.

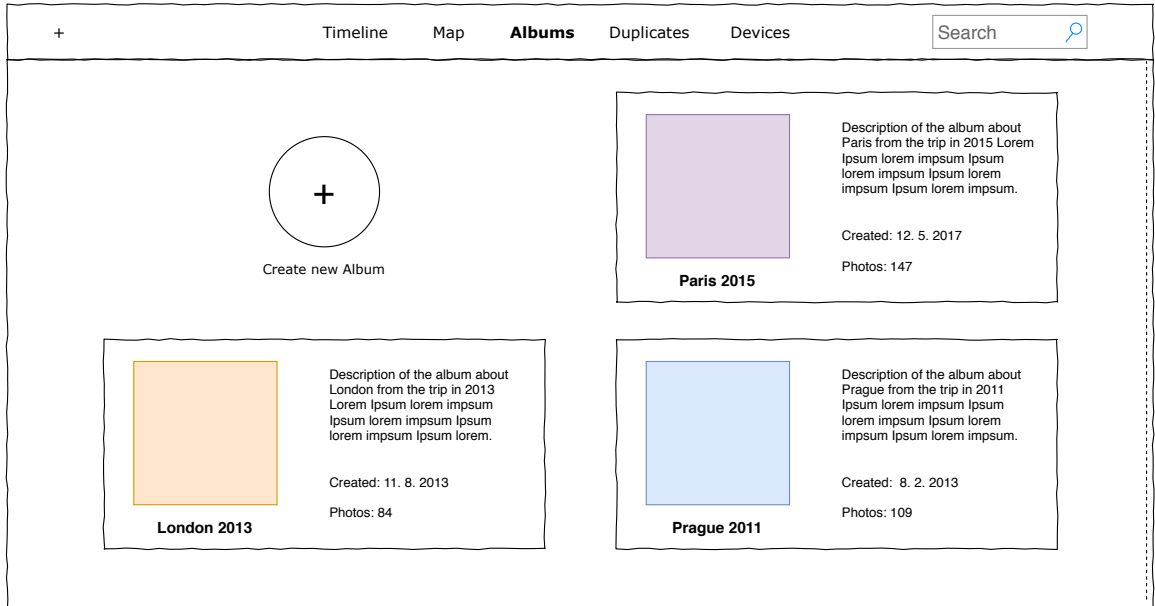


Figure 6.6: Proposed design of the Album section. Albums with name, thumbnail, description, creation date and number of photos.

Albums

Creating an album of photos is one of the key features of any photo organising application. Defining an album only by its name can be misleading, users may forget what they meant by the name and would have to look inside to see what kind of photos they put there. To avoid such situations, an album should be defined not only by its name but also by a thumbnail and description. Thumbnail image is what the user's eye focuses on the most. The layout design is shown in Figure 6.6.

Duplicates

When clicking on the Duplicates section, the user is presented with an option to start the process of duplicate detection. He can choose from 3 options: to detect exact duplicates, to detect possible duplicates or similar photos, or to detect both. After the process is finished, the detected photos are displayed. As shown in Figure 6.7, they are displayed in a row, each photo with a name and device it is stored in, with an option to be deleted.

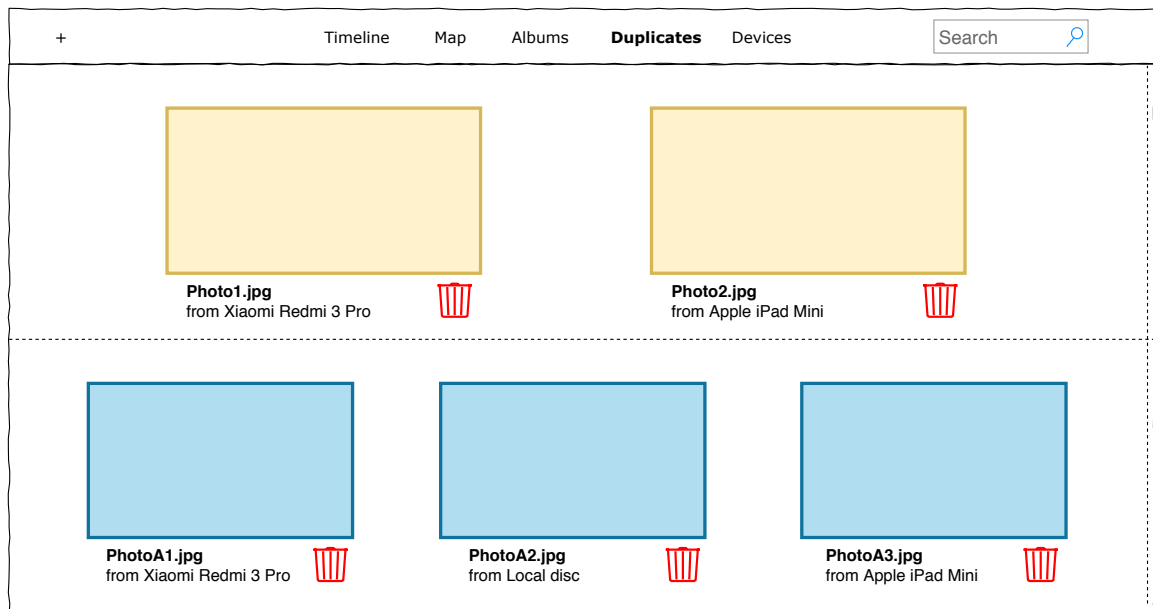


Figure 6.7: Proposed design of the Duplicates section. Detected duplicates are displayed next to each other with an option to be deleted.

Devices

The user is able to import photos from as many devices as he uses. To keep the order of photos imported from different devices, the *Devices view* was created. Even after disconnecting a storage device from the computer, the user can see information about his photos in the library. The disconnected devices are distinguished from those that are connected. Users are also able to remove the device from the library and all photos from that device with it.

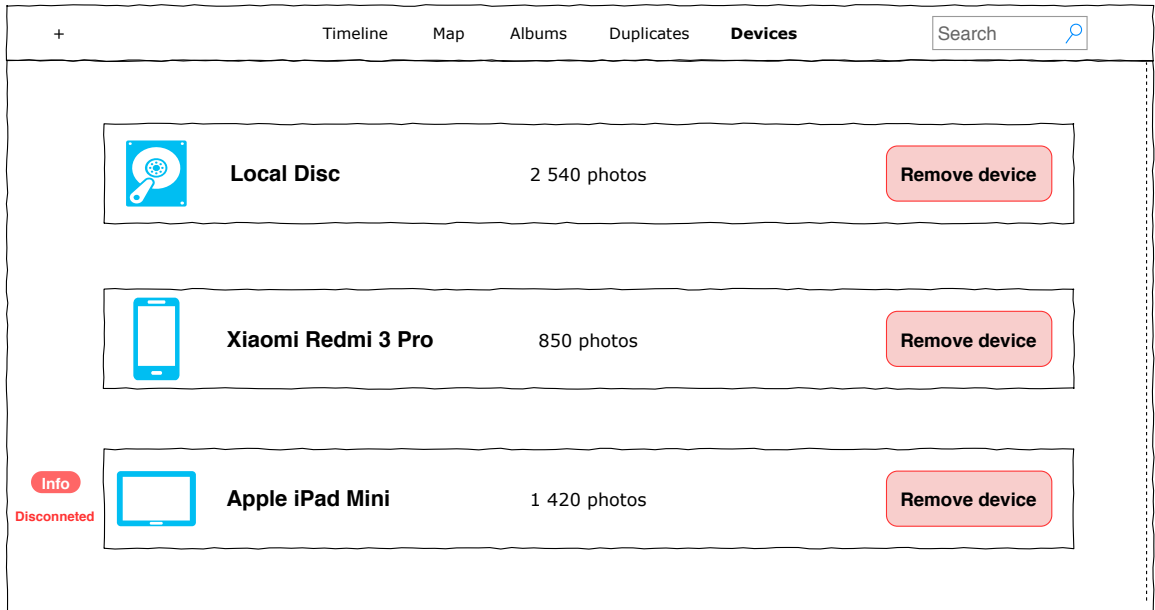


Figure 6.8: Layout design of the Devices section. Every device has a name and a number of photos imported in the library. Disconnected devices are distinguished by distinctive colored information.

Filter

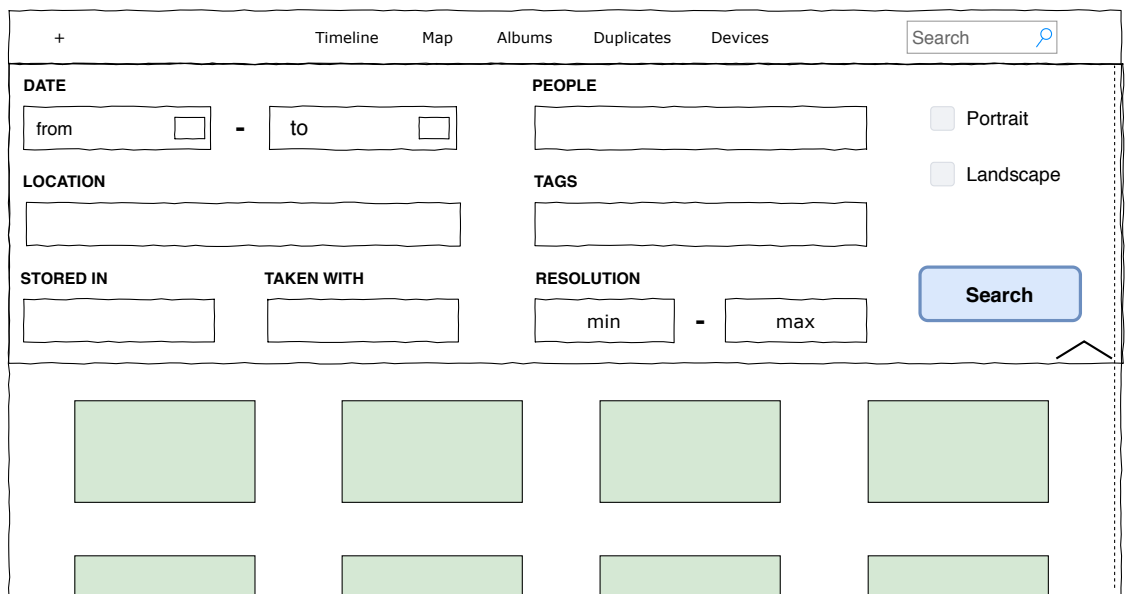


Figure 6.9: Layout design of the Filter section.

Last but not least, there is the Filter view. Based on the results of question 6 from the form in Appendix A, photos should be filterable by these criteria: date, location, device it was taken with, tags, people on the photo and whether it is a landscape picture or a portrait. Additionally, users can filter photos based on which device they are stored. Another added feature is to filter images depending on their resolution.

To choose a date, the user can use a date-picker or type a date manually. For people, a list of devices or tags, users can pick from a dropdown menu with available options in each field. After selecting desired attributes, pressing the button *Search* will present the results. To provide more space for filtered photos, the filtering bar can be minimised using an arrow at the right bottom (as presented in Figure 6.9).

Chapter 7

Implementation

This chapter describes the process of application implementation. Implementation was based on the design described in Chapter 6. First of all, this chapter focuses on the project architecture and description of chosen technologies for the final product. Later, it lists interesting implementation problems and their solutions.

7.1 Project architecture

The most crucial step in the early stages of implementation was to choose a fitting framework. In Section 4.2.1, three of the well-known frameworks are described. Based on their advantages, disadvantages and overall usage, I decided to use Electron for my application implementation. It was chosen mainly because of its built-in functionalities, such as `electron-builder`, automatic updates and crash reporting. Furthermore, it includes a detailed and easy-to-read documentation [8].

To make the application interface responsive, the front-end part of the implementation was done using React. Integrating React with Electron might become complicated, by cause of security settings and mutual permissions and relative paths to source files and assets. That is the reason why a `secure-electron-template`¹ was used as a starting point. It offers mentioned integration with React and Redux, with the addition of a best-practices security setting.

7.1.1 Electron

What Electron is and its comparison with Haxe and NW.js was described in Section 4.2.1. In this section, I will focus more on Electron architecture. All further mentioned facts about Electron are cited from its official documentation [8].

As far as development is concerned, an Electron application is essentially a Node.js application. The starting point is `package.json` file that is identical to that of any Node.js module. Electron apps are developed in JavaScript using the same principles and methods. The lifecycle of the application is managed through `electron.app` class and application windows are created using `BrowserWindow` class. By default, the content of the application is loaded from `index.html`. Electron keeps the back-end JavaScript state separate from that of the front-end. This isolation is one of the differences between Electron and NW.js. That is why data sharing between back-end and application windows is handled via inter-

¹Template by reZach available here <https://github.com/reZach/secure-electron-template>

process communication (sometimes called “message passing”). There are several ways to communicate (share data) between the main process and renderer processes. In my case, the `ipcRenderer` and `ipcMain` modules were used. This communication is presented in Figure 7.1.

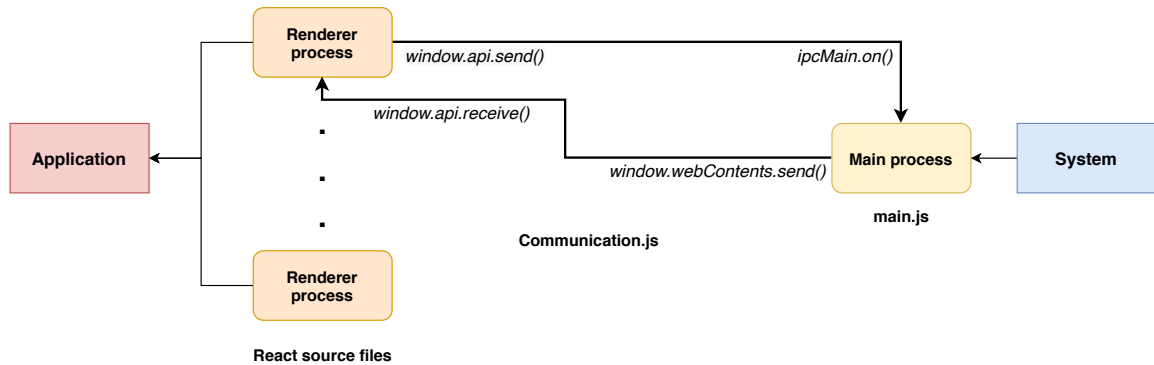


Figure 7.1: Communication between Main process and renderer processes in Electron. Each renderer process has a communication channel with the main process.

7.1.2 React

React is a JavaScript library for building user interfaces. It allows creating interactive UIs built from components, which are efficiently updated and rendered only when the data changes. These encapsulated components are reusable, which makes the code easier to read and debug. There are two main approaches when using React: Class approach and React Hooks. My application interface uses the React Hooks approach. More about React is available in its documentation [9].

7.1.3 Data storage

The data analysis from Section 6.1 shows that there is a need to store application data. To prevent forcing a user to install third-party software for a database and to keep all the data offline, I tried to store the application metadata in the JSON format using LowDB². LowDB is a small JSON database for Node, Electron and the browser. The data was distributed using a tree structure. However, this approach showed performance issues concerning searching the database. If there was a request to find a photo in a library based on a tag, the data would have to be searched linearly. Moreover, the whole JSON structure would become hard to maintain. For these reasons, the data storage was remade.

IndexedDB

IndexedDB is a transactional database system, like an SQL-based Relational Database Management System (RDBMS). However, unlike SQL-based RDBMS, which uses fixed-column tables, IndexedDB is a JavaScript-based object-oriented database. It is a database that is built into a browser and stores multiple kinds of values. It uses indexes to enable high-performance searches of data. By its documentation, it is intended more for offline than online applications. However, the API is quite complicated to use with not so easy-to-read documentation and with a steep learning curve. There are multiple wrappers to make

²<https://github.com/typicode/lowdb>

the development with IndexedDB more “programmer-friendly”. Dexie.js³ is a minimalist wrapper for IndexedDB with high performance and thoroughly explained in the detailed documentation. The first step was to design the structure of the database. The final structure is presented in Figure 7.2.

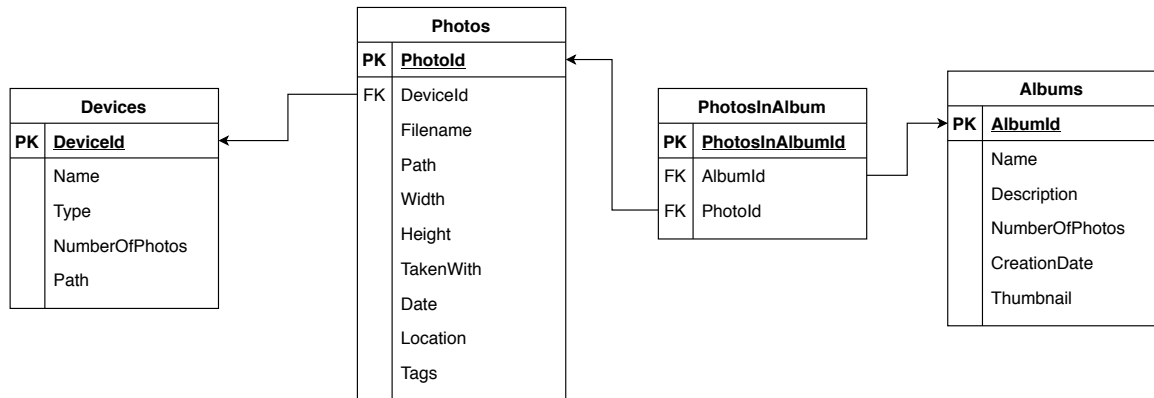


Figure 7.2: Structure of the final database implemented using Dexie.js.

Each table is added to the database by its name and primary key, followed by the list of indexed properties. Unlike SQL, not all properties need to be specified, only the ones that are meant to be indexed. Dexie.js offers many useful methods for tables and collections. Transactions are asynchronous and return *promises*⁴.

7.2 Used libraries

In this section, existing libraries used for application implementation are listed, and the reason they were used is explained.

7.2.1 Leaflet

One of the desired functionalities was a map with photo locations. There was no need to implement a new map from a scratch because there are already functional and free libraries for this handling. I needed an interactive map interface with the possibility of custom markers and clustering of them. For that purposes, I used Leaflet [1]. It also offers component approach for React⁵. For photo clustering, I used a `Leaflet.markercluster` plugin⁶.

7.2.2 Exiftool

ExifTool⁷ is a platform-independent Perl library plus a command-line application for reading, writing and editing meta information in a wide variety of files. ExifTool supports many different metadata formats. It is powerful, fast, flexible and customisable. It was used to extract metadata from the image in Figure 3.2. To be able to use it in Electron

³<https://dexie.org/>

⁴More about Javascript promises is in the documentation <https://javascript.info/promise-basics>

⁵<https://react-leaflet.js.org/>

⁶<https://github.com/Leaflet/Leaflet.markercluster>

⁷<https://exiftool.org/>

application, an interface or a wrapper was required. `Node-exiftool`⁸ is a Node.js interface to the Exiftool command-line application. Its methods were used for working with Exiftool.

7.2.3 Jimp and ImageMagick

To make loading of pictures faster on pages such as Timeline or Map—where there is no need for them to be displayed in full quality—I decided to create miniatures (thumbnails) of photos. They are created when the photos are imported to the library.

JavaScript Image Manipulation Program (Jimp)⁹ is an image processing library for Node.js written entirely in JavaScript, with zero native dependencies. It offers many ways to work with images, however, I used only a few methods to create miniatures. This optimised application UI performance, but the process of creation of the miniatures was too slow (further described in Section 8.1.1). For this reason I used ImageMagick¹⁰. It is a powerful photo editing tool, but its many features do not affect the length of processing time that much.

The location of miniatures depends on the operating system that runs the application. The path is identical to those mentioned in Section 4.1. A filename of the miniature is `id` of the original photo. Each photo has a path to its miniature saved in the database.

7.2.4 Semantic and Material-UI

Semantic UI React¹¹ and Material-UI¹² are both libraries with React components for faster and easier web development. They were used for multiple reusable components throughout the user interface implementation. Semantic was used for forms and sidebars, while Material-UI was used more for buttons, icons and date pickers. They both provide detailed documentation with clear examples, which made them uncomplicated to integrate.

7.2.5 Timeline component

`React-vertical-timeline-component`¹³ is a React component used in a main page of the application to display photos chronologically in a timeline. The single-column version was used. It was customised to match the theme of the application and buttons for specific years were added to display each year's photos separately.

7.3 Implementation details

This section is focused on more problematic parts of the implementation. Some of the functionalities need a deeper description and explanation of why they were implemented in such a way.

7.3.1 Working with metadata

After the photo's metadata is read, particular attributes are saved in the database. User is able to edit photo's metadata after clicking button "edit metadata". All the changes are

⁸<https://github.com/Sobesednik/node-exiftool>

⁹<https://github.com/oliver-moran/jimp>

¹⁰<https://www.npmjs.com/package/imagemagick>

¹¹<https://react.semantic-ui.com/>

¹²<https://material-ui.com/>

¹³<https://github.com/stephane-monnot/react-vertical-timeline>

presented only in the application database. The original picture stays untouched. This way, users can not accidentally damage their pictures—they work only with a representation of them.

7.3.2 Duplicate detection

A desired functionality of the application was to detect duplicates in the library. There are two possible approaches available: The first one is to detect a duplicate when it is about to be imported and the second one is an action initiated by the user to search the library for duplicates.

Importing

When a selected photo is being imported, its metadata is read, and a certain part of it is saved in the database. In this process, each photo is determined by its path and name. When importing, a database is searched whether there is an existing photograph with that path, thus it is not possible to import the same picture twice. Importing of the photo is done using `importEachPhoto()` asynchronous function (presented code example is in Listing 1). Since `path` is an indexed attribute of table `Photos`, searching is accelerated. The function returns a number of successfully imported photos.

```
async function importEachPhoto(photos, deviceId) {  
  
  let counter = 0;  
  
  await db.transaction('rw', db.photos, db.devices, async () => {  
    for (let i = 0; i < photos.length; i++) {  
      await db.photos.get({ path: photos[i].path }).then((photo) => {  
        // if the photo is not already in the library  
        if (photo === undefined) {  
          counter++;  
          db.photos.put(photos[i]);  
        }  
      })  
    }  
  })  
  // get current number of photos in device  
  const actualNumber = db.devices.get(deviceId, (device) => {  
    return device.numberOfPhotos;  
  })  
  // update counter of photos for device  
  db.devices.update(deviceId, { numberOfPhotos: actualNumber + counter });  
  
  return counter;  
}
```

Listing 1: Asynchronous Javascript pseudocode representing function used for importing pictures to the database. Before photo is saved, a check is done to see if it already exists in the database (particularly in table `Photos`). After importing, it updates `numberOfPhotos` attribute of the device.

User-initiated action

User can not import the same picture twice. Nevertheless, having duplicated photos with different names across multiple devices is a possible situation. For that reason, the user is able to initiate action for searching the library and detect groups of same photos, even with different names and paths. This action is executed after clicking button “Detect duplicates” in page Duplicates. It initiates an algorithm described in Listing 2.

```
async function findDuplicates() {

  const arrayOfArrays = []; //final structure
  await db.transaction('r', db.photos, async () => {
    let previousPhoto = {}; // saved to compare with current
    let readingFirstTime = true; // to skip first comparison with previousPhoto
    let duplicateDetected = false; // to control a chain (more than 2)
    let arrayOfDuplicates = [];

    //photos are ordered by date, it is the most important parameter
    db.photos.orderBy('date').each((photo) => {
      if (!readingFirstTime) {
        // multiple parameters of photos are compared
        if (photo.parameters === previousPhoto.parameters) {
          // if there is not a chain of duplicate photos
          if (!duplicateDetected) {
            arrayOfDuplicates.push(previous, photo);
            duplicateDetected = true;
          } else { // if it is a chain, store only current photo
            arrayOfDuplicates.push(photo);
          }
        } else {
          //the previousPhoto and current photo are not duplicates
          duplicateDetected = false;
          // if the previous photos were a chain a duplicates,
          // store the group globally and empty the arrayOfDuplicates
          if (arrayOfDuplicates.length !== 0) {
            arrayOfArrays.push(arrayOfDuplicates);
            arrayOfDuplicates = [];
          }
        }
      }
      readingFirstTime = false;
      previousPhoto = photo;
    }
  )
  return arrayOfArrays;
}
```

Listing 2: Algorithm written in JavaScript pseudocode to demonstrate `findDuplicates()` function. The algorithm is iterating through sorted photos by date and comparing each one with the previous. If there is a match, it stores them in `arrayOfDuplicates`. It can detect chain of duplicate photos. When the chain is broken, photos from `arrayOfDuplicates` are saved in `arrayOfArrays` and the first array is emptied.

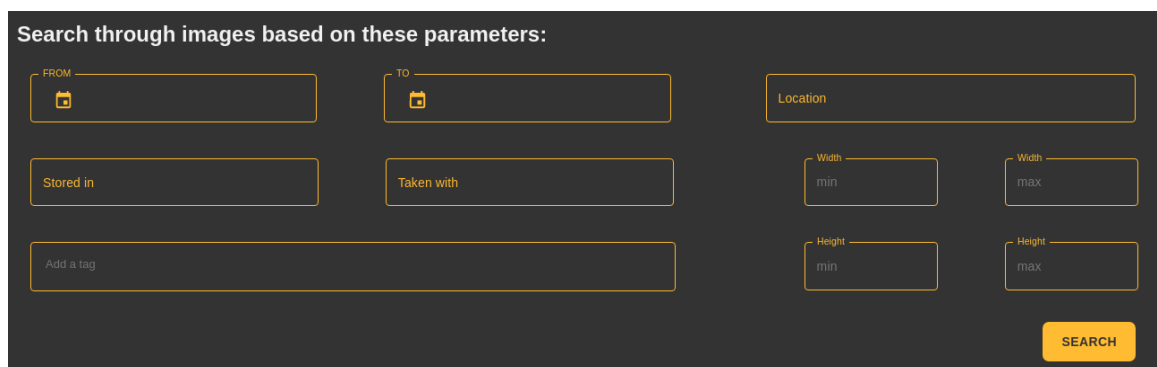
7.3.3 Device management

Users are not only able to import photos from the local drive, but also any device connected to the computer. In the application, there is a representation of every device. It contains a name, type, and a path to the root directory of the device. Photos imported from a certain device have a relative path bounded with their parent devices. The reason for this is the application portability. User can export and afterwards import their database of pictures to another computer. After specifying new paths to their devices, the whole library is functional without the need of re-importing the photos, because the photo paths are dependant on the device path. Removing a device from the library would mean deletion of all photos imported from it.

User can specify the path to a certain device when the devices are created (in the page Devices). Afterwards, when photos are imported, it is necessary to select which device they come from.

7.3.4 Searching

One of the key functionalities of the application is to filter images based on attributes specified by the user. The final parameters of searching are displayed in Figure 7.3.



The screenshot shows a dark-themed search interface. At the top, it says "Search through images based on these parameters:". Below this, there are several input fields arranged in a grid. The first row has "FROM" and "TO" (both with folder icons) and "Location". The second row has "Stored in", "Taken with", "Width min", and "Width max". The third row has "Add a tag", "Height min", and "Height max". A yellow "SEARCH" button is at the bottom right.

Figure 7.3: The final parameters that users can use to filter their images. This image is a screenshot of the application.

At least one parameter needs to be filled for searching to perform. The `search()` function is executed in one transaction. In Listing 3, there is Javascript pseudocode to represent the logic of the searching algorithm. Only the first parameter is read from the database, because Dexie.js does not offer chaining of 'where' clause. When this feature is added to Dexie, search function will be optimised.

```

async function search(parameters) {
  let noParameterRead = true;
  let finalArray = [];
  await db.transaction('r', db.photos, async () => {
    forEach (parameter) {
      if (parameter is not empty) {
        if (noParameterRead) {
          //this is the first parameter we search for
          let res = db.photos.where(paramName).equals(paramValue).toArray();
          finalArray.push(res);
          //at least one parameter was read
          noParameterRead = false;
        }
        else {
          forEach(item in finalArray) {
            // search finalArray
            if (item.parameter != parameter)
              // remove items which do not meet conditions
              finalArray.remove(item);
          }
        }
      }
    }
  }
  return finalArray;
}
}

```

Listing 3: Asynchronous Javascript function used for searching through photos based on parameters filled by users.

7.4 Packaging and used developer tools

By using `electron-builder` mentioned in Section 7.1, it became noticeably easy to package and build a ready for distribution Electron app for macOS, Windows and Linux. The application needs to be compiled for each of the platforms. A compiled executable version for these three platforms is part of the CD content listed in Appendix B. They were also shared online¹⁴ and given users for evaluating purposes.

7.4.1 Developer tools

The application was developed using JetBrains WebStorm¹⁵ with school licence. It was implemented and tested on Ubuntu 18.04. The project was versioned and backed up using the Github repository.

¹⁴<http://www.stud.fit.vutbr.cz/~xurmin01/Memotheca/index.html>

¹⁵<https://www.jetbrains.com/webstorm/>

Chapter 8

Testing

This chapter summarises testing methods performed to prove the application’s usability and functionality. Firstly, it describes tests of performance, with an emphasis on testing on multiple platforms. Secondly, the reactions of users are described.

8.1 Performance testing

The most time-consuming actions in the application are importing of photos, duplicate detection and their filtering by user-specified parameters. This is the reason why the most significant emphasis was on them in testing.

8.1.1 Import

When a photo is being imported, its metadata is read (by Exiftool), a miniature is created, and extracted data is written to the database. All of these three processes are asynchronous. When testing the application the first time, this process took approximately 2.3 seconds to import a photo. Therefore, the application had to be tested to see which of these processes took the most time. Each process was debugged and tagged with a timestamp of start and end. As a result, miniature creation took the longest. The Jimp library significantly extended the time of importing photos. For this reason, I decided to analyse my options further. After thorough research, `Imagemagick` was used.

In Table 8.1, performance results are shown. The import was tested multiple times, always importing 236 photos, both JPG and PNG files, together 631,3 MB large. The biggest photo had 11 MB, the smallest 17 KB.

Testing of import [s]			
Reading metadata	Miniature creation	Storing data to DB	Overall
37.31	10.384	0.312	48.006 s
31.84	11.112	0.322	43.274 s
33.24	10.587	0.314	44.141 s
34.89	12.741	0.308	47.939 s
32.12	10.107	0.354	42.581 s

Table 8.1: Results of import testing. Three main operation were tested and their dependence on time compared. The average time to import 236 photos is 45.188 seconds, that is 191 milliseconds per photo.

The results show that the largest action is reading of metadata—it takes approximately $\frac{3}{4}$ of the import time. Creation of the miniatures takes around $\frac{1}{4}$ of the time and storing the data in the database is really fast (1.3 ms per photo).

I also tried importing a lot more photos. For example, 1300 photos were imported in 3 minutes and 56 seconds, which makes 180 milliseconds per photo. This could be optimised by not extracting metadata from each photo separately, and do it in one process. However, this solution will require a deeper examination of Exiftool.

8.1.2 Duplicate detection

User can search their library for duplicates. To test the speed of this feature, I repeated the process of detection multiple times with a variation of photos in the library. The results are presented in Table 8.2.

Duplicates Testing [ms]	
Number of photos in library	Average time
100	97 ms
260	122 ms
550	187 ms
800	301 ms
1000	340 ms
2000	635 ms

Table 8.2: Results of duplicate detection testing. The detection was executed with a variety of photos in the library, the average time changed depending on number of photos.

The results undoubtedly show that the more photos in the library, the longer the duplicate detection takes. However, the time complexity is not linear.

8.1.3 Search

The last time-consuming action in the application is searching. Users can search their library based on many parameters. In this testing, I focused on the time required to get results based on the number of photos currently in the library and the number of parameters to search by. The results of these measurements are presented in Table 8.3.

Search Testing		
Number of photos in library	Number of parameters	Average time
850	1	70.9 ms
850	2	93.5 ms
850	3	90.3 ms
850	4	86.1 ms
2000	1	128.3 ms
2000	2	160.5 ms
2000	3	170.4 ms
2000	4	176.8 ms

Table 8.3: Results of search testing. The table shows dependence of the search speed on the number of parameters and the number of photos in the library.

Based on the measurements, it is obvious that not only the number of photos but also the number of search parameters influences the length of search time. However, it is possible that many specified parameters will shorten the computing time because the array which the algorithm works with will contain fewer values.

8.1.4 Multiplatform testing

The final application is supposed to be functional on multiple platforms. To verify this claim, it was tested on Ubuntu 18.04, Kali 2019.1 and Windows 10. The application has been successfully launched and ran without any problems in all three cases. Integration with Exiftool and IndexedDB worked in all the platforms.

8.2 User testing

To verify the usability of the user interface, a narrow circle of testers was selected, whose task was to test the application in order to identify deficiencies of the interface. The application was sent to 7 volunteers, who started to use the software for the purpose of organisation and management of their photos. Five of the seven users tested the application on Windows, two on Linux. They installed it without troubles. I used *controlled settings involving users* method of evaluation. This method is further described in Chapter 13 of this book [15].

8.2.1 Clarity of the user interface

I was present when they tested the user interface. The feedback took the form of an interview. They were asked to complete tasks and to comment on the process. As a result, valuable information has been obtained. Several users had suggestions on how to improve the application. Some users lacked an intro home page, where it would be explained, what is the application capable of with a description of its functions. It was pointed out, that icon of a plus (for import) and icon of a magnifying glass (for search) are not intuitive enough, and should be replaced with the page name. Some users expected that the application could create a slideshow of photos and going through them on full screen. This idea was added to further future extensions. Another required functionality was to be able to create a new device on the Import page (not only on the Devices page). That would fasten the process of import to a new device.

8.2.2 Application usability

Six out of seven users stated that they would personally like to start using the application after its publication. However, they would appreciate if they could use keyboard shortcuts for more actions (for example *Esc* for closing sidebar). They would like to have an option to edit the colour theme of the application. The seventh volunteer commented that he is satisfied with the features of Lightroom and does not need another application. All comments from the users will be considered and taken into account in further development.

Chapter 9

Conclusion

The goal of this thesis was to implement a multi-platform application for organising and managing photos. The application is focused on the management of photos from multiple devices, duplicate detection and filtering of photos based on user-specified parameters.

Based on the analysis of user requirements and research of existing solutions, application functionalities, along with user interface, were designed. To understand extracting data from photographs, photo metadata were studied. After the analysis of principles of usable multi-platform application design, the software was implemented using Electron and React.

The final application offers many functionalities: hierarchical order of photos on the timeline, location of photos displayed on a map, device-based photo management, distribution of photos into albums, editing metadata of photos, user-specified parameter search, and duplicate detection.

This work can be further developed in the future by implementing automatic face recognition. Additionally, another way to further develop this work is to implement an algorithm for similar photos detection, based on multiple criteria.

The final application was tested by users, and its performance was examined on multiple operating systems. The project was open-sourced on GitHub¹ in July 2020.

¹<https://github.com/Kiwinka/Memotheca>

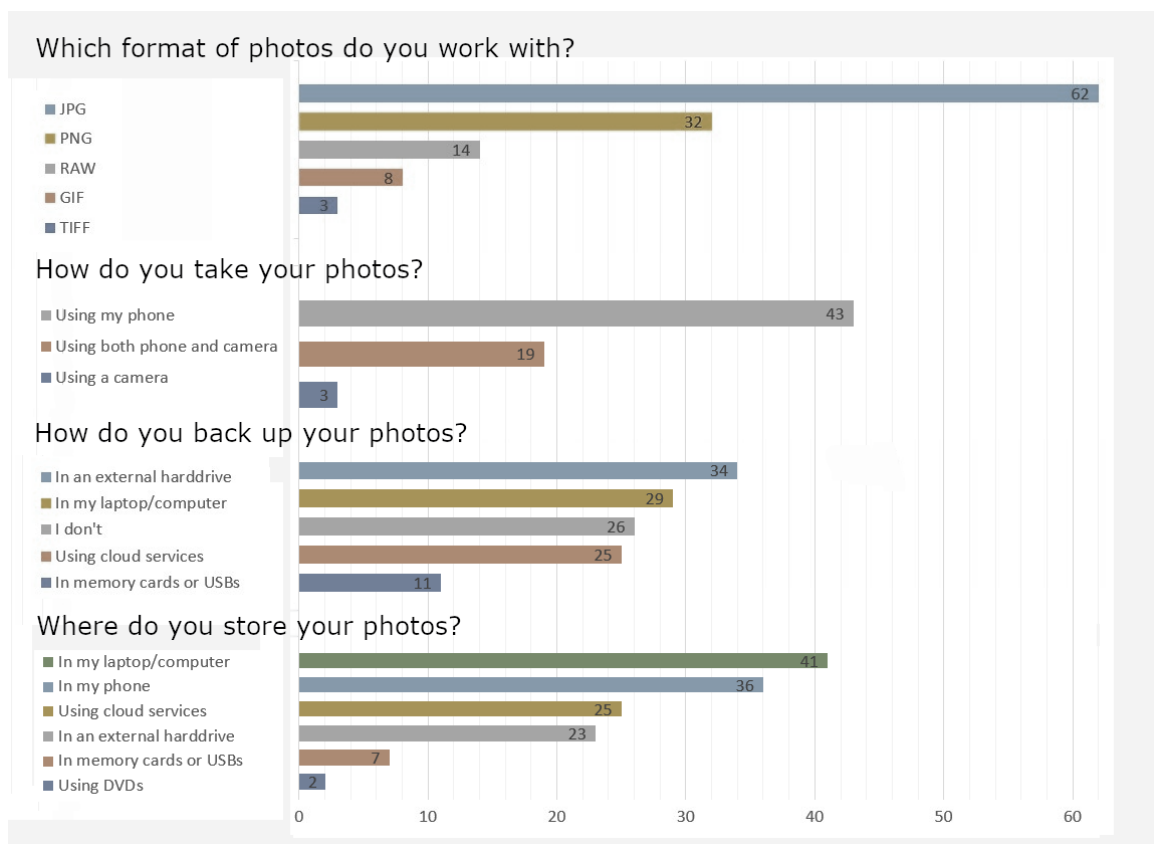
Bibliography

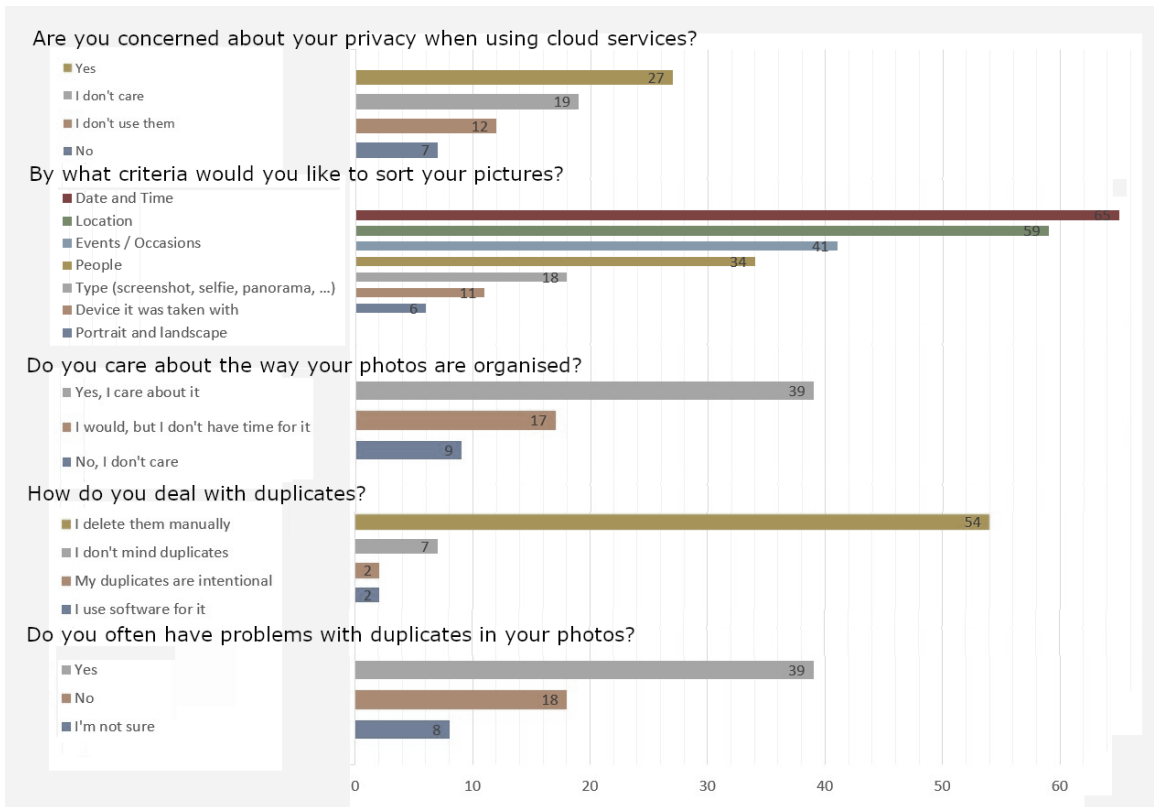
- [1] Vladimir Agafonkin: *Leaflet API Documentation*. 2020.
Retrieved from: <https://leafletjs.com/reference-1.6.0.html>
- [2] Baca, M.: *Introduction to metadata*. Los Angeles: Getty Publications. third edition. 2016.
Retrieved from: <http://www.getty.edu/publications/intrometadata>
- [3] Bittner, K.; Spence, I.: *Use Case Modeling*. Addison-Wesley. 2003. ISBN 0-201-70913-9.
- [4] Brogie, M.: Average Number of Photos Taken Per Day Around the World. Available at <https://www.repsly.com/blog/field-team-management/field-data-insight-average-number-of-photos-taken-per-day-worldwide>.
- [5] Buley, L.: *The User Experience Team of One: A Research and Design Survival Guide*. Brooklyn, N.Y. : Rosenfeld Media. 2013. ISBN 9781457102943.
- [6] Chris Bank, J. C.: *The Guide to UX Design Process & Documentation*. UXPin Inc.. 2015.
Retrieved from: <https://www.uxpin.com/studio/ebooks/guide-to-ux-design-process-and-documentation/>
- [7] Daoust, N.: *UML Requirements Modeling for Business Analysts*. Technics Publications LLC. 2012.
- [8] ElectronJS: *Electron Documentation*. 2020.
Retrieved from: <https://www.electronjs.org/docs>
- [9] Facebook: *ReactJS Documentation*. 2020.
Retrieved from: <https://reactjs.org/docs/>
- [10] Goldstein, J.: How Much Photo Video Data Do You Have Stored? Available at <https://www.backblaze.com/blog/how-much-photo-video-data-do-you-have-stored/>.
- [11] Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers/Elsevier. 2010. ISBN 978-0-12-375030-3.
- [12] Kabachinski, J.: TIFF, GIF, and PNG: Get the picture? *Biomedical instrumentation & technology*. vol. 41, no. 4. 2007: pp. 297–300.
Retrieved from: <https://www.aami-bit.org/doi/pdf/10.2345/0899-8205%282007%2941%5B297%3ATGAPGT%5D2.0.CO%3B2>

- [13] Metadata Working group: *Guidelines for Handling Image Metadata*. 2010.
Retrieved from: https://web.archive.org/web/20120131102845/http://www.metadataworkinggroup.org/pdf/mwg_guidance.pdf
- [14] NW.js: *NW.js Documentation*. 2020.
Retrieved from: <https://nwjs.readthedocs.io/>
- [15] Preece, J.; Rogers, Y.; Sharp, H.: *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons. 2015. ISBN 978-1-119-02075-2.
- [16] Robl, E. H.: *Organizing your photographs*. Amphoto. 1986. ISBN 978-0817453008.
- [17] International Organization for Standardization (ISO), I. O.: *ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. 2019.
- [18] Tesic, J.: Metadata practices for consumer photos. *IEEE MultiMedia*. vol. 12, no. 3. 2005: pp. 86–92.

Appendix A

Questionnaire





Appendix B

CD content

readme.txt

File describing structure of the content.

xurmin01-thesis.pdf

PDF thesis.

xurmin01-thesis-print.pdf

Print version of PDF thesis.

thesis-src/

L^AT_EX source code.

app/

Source files of the application.

dist-linux/

Executable application for Linux.

dist-windows/

Executable application for Windows.

dist-mac/

Executable application for Mac OS.