FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	iOS aplikace pro simulaci analogového focení
Student:	Lucia Čahojová
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Analogová fotografie je specifická tím, že obraz zaznamenává na klasický světlocitlivý materiál, který může mít různé vlastnosti odrážející se v podobně barev, ostrosti a dalších parametrech. Řada současných aplikací pro úpravu digitálních fotografií nabízí filtry napodobující vizuálně analogové fotografie. Cílem této práce je vyvinout mobilní aplikaci pro iOS, která nebude upravovat již pořízené fotografie, ale přímo simulovat analogové fotografování. Postupováno bude v souladu s tradičními postupy softwarového inženýrství:

• Analyzujte problematiku analogového a digitálního fotografování, charakteristik analogových fotografií a filmů, procesu digitalizace a možností úprav digitálních fotografií na vzhled analogových.
• Proveďte stručnou rešerši existujících aplikací umožňující tvorbu či úpravu digitálních fotografií do vizuální podoby analogových.
• Sestavte požadavky na vlastní aplikaci a případy užití.
• Navrhněte aplikaci pro iOS po stránce UI i architektury. Při návrhu zohledněte uživatelskou přívětivost a budoucí rozšiřitelnost aplikace (např. o další filtry/filmy).
• Implementujte aplikaci dle návrhu a zdůvodněte výběr technologií.
• Otestujte aplikaci po stránce funkčnosti i uživatelské přívětivosti.
• Zhodnoťte přínosy aplikace, její použitelnost a rozšiřitelnost a navrhněte možnosti dalšího rozvoje.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 25. června 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

iOS aplikace pro simulaci analogového focení

Lucia Čahojová

Katedra softwarového inženýrství
Vedúci práce: Ing. Marek Suchánek

11. mája 2023

Poďakovanie

Rada by som poďakovala svojmu vedúcemu práce, pánovi Ing. Markovi Suchánkovi, za odborné vedenie, pomoc, trpezlivosť a cenné rady pri písaní tejto práce. Moja veľká vďaka taktiež patrí mojej rodine a priateľom, ktorí ma podporovali ako pri písaní práce, tak aj pri celom bakalárskom štúdiu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracovala samostatne a že som uviedla všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržovaní etických princípov pri príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, najmä skutočnosť, že České vysoké učení technické v Prahe má právo na uzatvorenie licenčnej zmluvy o použití tejto práce ako školského diela podľa §60 odst. 1 citovaného zákona.

V Prahe 11. mája 2023

Odkaz na túto prácu

Čahojová, Lucia. *iOS aplikace pro simulaci analogového focení.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Táto bakalárska práca sa zaoberá vývojom mobilnej aplikácie pre simuláciu analógoveho fotenia pre platformu iOS, s dodržaním klasických postupov softwarového inžinierstva. Najskôr je vykonaná analýza problematiky digitálneho a analógového fotografovania a spolu s ňou rešerš konkurenčných aplikácií. Na základe analýzy sú zostavené požiadavky na aplikáciu, podľa ktorých je vytvorený návrh. Na návrh nadväzuje implementácia a testovanie aplikácie. Práca je zakončená zhodnotením výsledkov a uvedním možného rozvoja aplikácie.

Výstupom práce je funkčná mobilná aplikácia, ktorá umožňuje zhotovovanie fotografií s efektom analógového filmu.

Kľúčové slová mobilná aplikácia, iOS, analógová fotografia, úprava fotografií, Swift, SwiftUI

Abstract

This bachelor's thesis deals with development of a mobile application for analog photography simulation for iOS platform, following classical software engineering practices. First, an analysis of digital and analog photography is conducted, along with investigation of similar applications. Based on the analysis, application requirements are formed, according to which the design is created. Design is followed by implementation and testing of the application. The thesis is concluded with an evaluation of the results and possible future development options for the application.

The outcome of the thesis is a functional mobile application that allows taking photos with the effect of an analog film.

Keywords mobile application, iOS, analog photography, photo editing, Swift, SwiftUI

Obsah

Zoznam obrázkov

Úvod

Zovšednenie prítomnosti mobilných telefónov v našich životoch zmenilo nielen spôsob akým komunikujeme, pracujeme, ale aj ako sa zabávame. Záležitosti, ktorých vybavenie kedysi trvalo dni, dnes trvá sekundy. Tak, ako boli popri tejto digitálnej revolúcií listy nahradené telefonátmi, chvíľu sa zdalo, že digitálna fotografia úplne vytlačí fotografiu analógovú. V posledných rokoch sa však ukazuje, že opak je pravdou a ľudia po celom svete znovu objavujú charakteristické čaro a estetiku filmových fotoaparátov.

Dnes už takmer každý nosí vo vrecku výkonný digitálny fotoaparát zabudovaný v svojom mobilnom telefóne. Preto sa javí ako prirodzené využiť pokročilé možnosti fotoaparátov smartfónov a spojiť ich s pôvabom analógovej estetiky. Používateľ tak prostredníctom mobilnej aplikácie simulujúcej vlastnosti analógového filmu dokáže využiť to najlepšie z oboch svetov. Prenosnosť a dostupnosť smartfónov a charakteristcký vzhľad analógovej fotografie.

Aj napriek tomu, že na trhu sa vyskytuje mnoho aplikácií s podobnou funkcionalitou, ich prevedenia neposkytujú dostatočné uspokojenie. Existujúce aplikácie sú často spoplatené, ponúkajú len jeden druh filtra, nedostatočne simulujú vzhľad fotografie odfotenej na film alebo im jednoducho chýba autentickosť analógového fotografovania.

Hlavným zameraním tejto bakalárskej práce je vývoj mobilnej aplikácie pre platformu iOS, simulujúcej analógové fotografovanie s čo najväčším dôrazom na autenticitu. Práca sa zaoberá analýzou, návrhom a implementáciou danej aplikácie, podľa tradičných postupov softwarového inžinierstva.

Prácu tvorí sedem kapitol. Prvá kapitola sa venuje zadefinovaniu cieľov práce. V kapitole druhej je vykonaná analýza digitálneho a analógového fotografovania, popísaný prieskum trhu a zostavenie požiadavkov na aplikáciu. Kapitola tretia sa venuje návrhu aplikácie, zahrňuje popis architektúry, používateľského rozhrania a popis vývoja aplikácií pre platformu iOS. Štvrtá kapitola popisuje implementáciu a piata proces jej testovania. V šiestej kapitole sú zhnuté výsledky práce a načrtnuté možnosti budúceho rozvoja aplikácie.

Kapitola 1

Cieľ práce

Hlavným cieľom práce je vyvinutie mobilnej aplikácie pre simuláciu analógového fotografovania, určenej pre operačný systém iOS za použitia programovacieho jazyka Swift.

Pre uskutočnenie hlavného cieľu je potrebné splniť viaceré čiastkové ciele, určené tradičnými postupmi softwarového inžinierstva, zahrňujúcimi analýzu, návrh, implementáciu a testovanie.

Prvým cieľom je prevedenie analýzy problematiky analógového a digitálneho fotografovania pre popísanie oboch druhov fotografie. Nadväzujúcim cieľom je definovanie možností úprav digitálnych fotografií na vzhľad analógových.

Ďalším cieľom je preskúmanie existujúcich aplikácií pre operačný systém iOS, umožňujúcich úpravu digitálnych fotografií do vizuálnej podoby analógových.

Nasledujúcim cieľom je zostavenie požiadavkov a prípadov použitia výslednej aplikácie. Na to nadväzuje návrh aplikácie po stránke užívateľského rozhrania a architektúry, vyhovujúci zadefinovaným požiadavkom. Podľa návrhu je potrebné aplikáciu naimplementovať a následne otestovať.

Posledným cieľom je zhodnotenie prínosov, použiteľnosti a rozšíriteľnosti výslednej aplikácie.

Kapitola 2

Analýza

Neoddeliteľnou súčasťou softwarového inžinierstva je analýza. Zhromažďuje a usporadúva informácie popisujúce potreby a požiadavky daného softwarového produktu. Vďaka správnemu zadefinovaniu požiadavkov je možné vytvoriť návrh mobilnej aplikácie, ktorá tieto požiadavky a ciele splňuje. [1]

Táto kapitola sa zaoberá analýzou problematiky analógového a digitálneho fotografovania, pre pochopenie základných pricípov oboch druhov fotografie. Taktiež popisuje existujúce aplikácie s podobnou funkcionalitou. na základe toho definuje funkčné a nefunkčné požiadavky, ktoré následne upresňuje pomocou modelu prípadov použitia a doméhového modelu.

2.1 Problematika analógového a digitálneho fotografovania

Základy fotografovania prístupného pre širokú verejnosť sú postavené na analógovej fotografii a vďačíme za to Georgeovi Eastmanovi, ktorý v roku 1888 predstavil a patentoval Kodak No. 1 [2], zobrazený na Obrázku 2.1a. O viac ako 100 rokov neskôr, v roku 1994 zverejnila firma Apple prvý farebný fotoaparát produkovaný komerčne pre veľké masy ľudí [3]. Fotoaparát niesol názov Apple QuickTake 100 a je vyobrazený na Obrázku 2.1b.

Tieto dva druhy fotografie sa líšia už v samotnej postate zachytenia a uloženia obrazu. Ako zdôrazňuje Christian Roemer [6], pre pochopenie rozdielu medzi analógovou a digitálnou fotografiou je potrebné vysvetliť, čo vôbec termín „analóg“ a „digitál“ znamenajú.

Analógové médium je médium, ktoré nepotrebuje byť nijakým spôsobom interpretované alebo prekladané, aby ho bolo možné zobraziť. Je ním napríklad vinylová platňa, či vytlačená fotografia. Obe stačí len „prečítať“.

Naopak, digitálne médium je potrebné uchovávať na pevných diskoch, CDC či DVD nosičoch vo forme núl a jednotiek v kóde. Pri pokuse o prečítanie tohto kódu voľným okom nie je zrejmé, že sa jedná o fotografiu. Pre zobrazenie

(a) Ukážka fotoaparátu Kodak No. 1 [4]

(b) Ukážka fotoaparátu Apple QuickTake 100 [5]

Obr. 2.1: Ukážky prvých fotoaparátov

danej fotografie je preto nutné tieto jednotky a nuly poskladať do jednotlivých pixelov fotky, ktorú je potom možné zobraziť. [6]

Aj napriek 100ročnému časovému rozdielu, sú oba druhy fotografie stále veľmi obľúbené a používané, preto diskusia o otázke, ktorý druh fotografie je lepší ako ten druhý už roky neutícha. Nedá sa na ňu však s určitosťou odpovedať, preto sú v nasledujúcich podkapitolách zadefinované základné termíny fotografie, obe technológie a ich vplyv na vlastnosti a charakter výsledných fotografií.

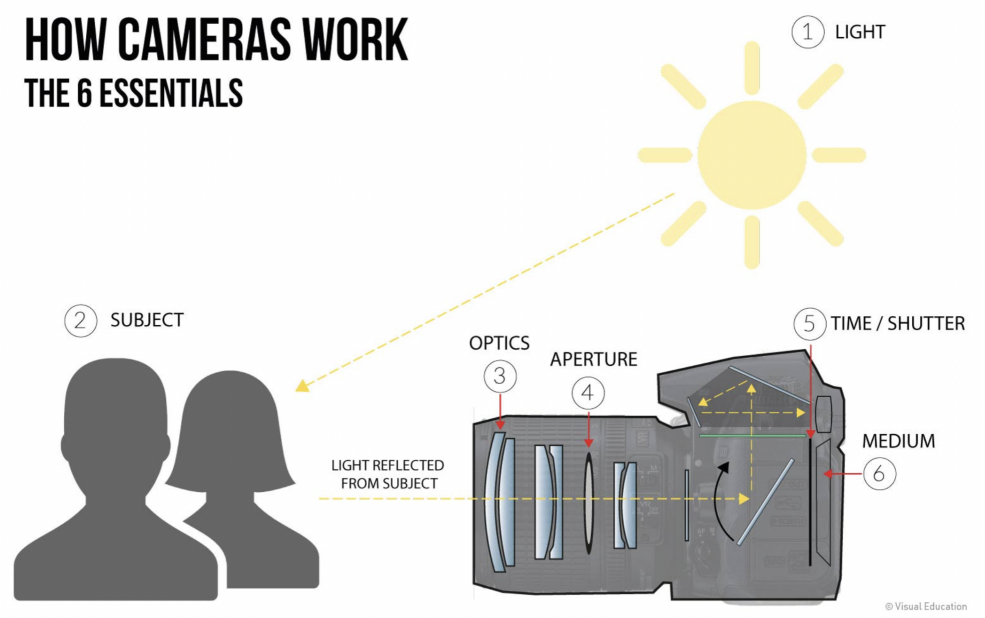
2.1.1 Ako funguje fotoaparát

Až po bezprostredné zachytenie a uloženie obrazu na určené médium sa procesy vyhotovenia fotografie zhodujú. Táto podkapitola sa venuje stručnému vysvetleniu toho, čo sa deje pred uložením fotky a čo všetko ovplyvňuje jej výsledný vzhľad.

2.1.1.1 Svetlo

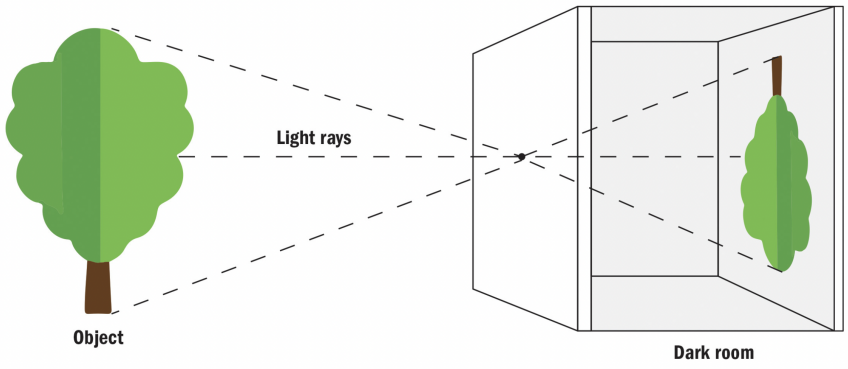
Základom každej fotografie je svetlo. Ak by sa človek pokúšal odfotiť fotku v miestnosti, kde je úplná tma, určite by neuspel. Avšak ak zasvieti baterkou na určité miesto, svetlo z baterky sa odrazí od osvieteného objektu priamo do fotoaparátu, čo mu dovolí objekt odfotiť [7], ako je znázornené na Obrázku 2.2.

Rovnakým spôsobom fungujú všetky lúče svetla navôkol. Lúč vyrazí zo zdroja a priamočiaro putuje k objektu, od ktorého sa odrazí v rovnakom uhle ako dopadol. Po odraze putuje priamočiaro ďalej, takže všade okolo nás sa odrážajú svetlené lúče v rôznych smeroch. Táto definícia je samozrejme zjednodušená, keďže rôzne materiály a povrchy môžu svetlo v rozličných mierach pohlcovať, lomiť alebo meniť jeho smer. [9]



Obr. 2.2: Ukážka ako funguje fotoaparát [8]

Priamočiarosť svetelných lúčov využíva aj dierková komora (lat. *camera obscura*), využívaná k projekcii obrazu. Funguje na princípe svetla prechádzajúceho malým otvorom do tmavej miestnosti, kde sa na ploche nachádzajúcej sa pred otvorom vytvorí prevrátený a zrkadlovo obrátený obraz, ako je vidieť na Obrázku 2.3.



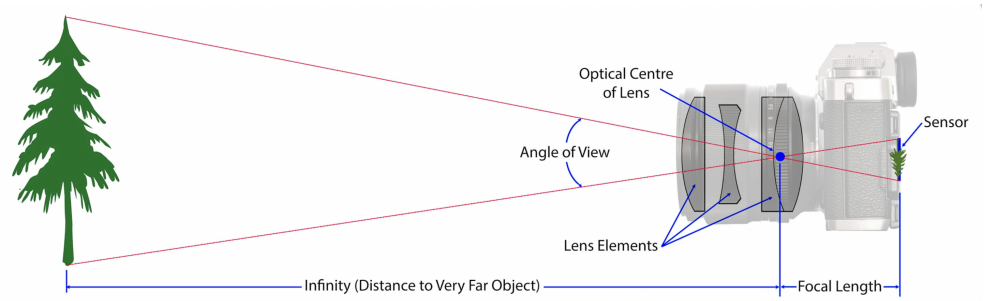
Obr. 2.3: Ukážka dierkovej komory [10]

Tento jav bol v histórií pozorovaný už od pradávna a používaný rôznymi spôsobmi, od pozorovania zatmenia Slnka, cez zobrazovanie predmetov, až po obkreslovanie obrazov maliarmi. Neskôr ľuďom napadlo dať na plochu nachádzajúcu sa pred otvorom svetlocitlivý materiál, ktorý bol schopný pre-

vrátený obraz zachytiť. Tým boli položené základy fotografie, kde sa tento princíp v sofistikovanejšej forme používa dodnes. [7, 11]

2.1.1.2 Optika

Svetelný lúč odrazený od objektu putuje do objektívu fotoaparátu, kde najskôr prejde šošovkou. Šošovka je typicky tvorená sústavou viacerých povrchov, ktoré lúč rôznymi spôsobmi ohýbajú a tým menia jeho smer. Je navrhnutá tak, aby sa svetelné lúče zbiehali v určitom bode za šošovkou, nazývanom ohniskový bod, zobrazenom na Obrázku 2.4. Tento bod sa nachádza na optickej osi, ktorá prechádza stredom šošovky a používa sa na opis svetelných lúčov.



Obr. 2.4: Ukážka fotoaparátu a ohniskového bodu [12]

Po tom, ako sa svetelné lúče stretnú v ohniskovom bode sa opäť odklonia od optickej osi a prevrátený zachytený obraz sa zobrazí na optickej rovine. Optická rovina je plocha rovnobežná s rovinou senzoru alebo filmu vo fotoaparáte. Zachytený obraz sa z optickej roviny dostáva na požadovaný snímač, no zväčša je najskôr prevrátený, aby sa javil tak, ako v realite.

Vzdialenosť medzi ohniskom a snímačom, nazývaná ohnisková vzdialenosť, do veľkej miery ovplyvňuje výsledný zachytený obraz, jeho ostrosť, či mieru priblíženia, avšak tieto detaily práca nepopisuje. [13]

2.1.1.3 Clona

Clona funguje ako nastaviteľný otvor v šošovke fotoaparátu, cez ktorý prechádza svetlo predtým, ako dosiahne obrazové médium, ako je naznačené na Obrázku 2.2. Pomocou clony je možné regulovať, koľko svetlných lúčov prenikne do senzoru. Čím väčší je otvor, tým viac lúčov clona prepustí a tým svetlejší obrázok vznikne.

Často je reprezentovaná pomocou clonového čísla, označovaného aj f-stop a zapísovaného f/N, kde f predstavuje ohniskovú vzdialenosť a N priemer vstupnej šošovky. Platí, že čím väčšia je clona, tým menšie je clonové číslo.

Na hodnote clony závisí aj hĺbka ostrosti, ktorá predstavuje rozsah vzdialeností objektov, ktoré sa na obrázku javia prijateľne ostré. Menšia clona (väčšie

clonové číslo) vytvára väčšiu hĺbku ostrosti, čo v praxi zaistí zaostrenú väčšiu časť scény. Väčšia clona (malé clonové číšlo) má za následok malú hĺbku ostrosti, takže zaostrená bude menšia časť scény. Ak by sme teda chceli zhotoviť umelecký portrét s fotografovanou osobou v popredí a rozmazaným pozadím, potrebovali by sme clonu čo najväčšiu. Naopak, napríklad na fotografiu krajiny by sa nám hodina clona malá. [13, 14]

2.1.1.4 Čas

Expozičný čas, či rýchlosť uzávierky sú pojmy označujúce dobu, počas ktorej je svetlo prepúšťané na senzor fotoaparátu. Expozičný čas reguluje uzávierka, mechanická súčiastka fotoaparátu, ktorá sa otvára a zatvára. Rýchlosť uzávierky teda určuje, ako dlho zostane uzávierka otvorená, aby sa svetelné lúče dostali k snímaču.

Zvyčajne sa vyjadruje v zlomkoch sekundy, napr. $\frac{1}{60}, \frac{1}{125}, \frac{1}{250}$...Vyššia rýchlosť uzávierky (napr. $\frac{1}{1000}$ sekundy) znamená, že uzávierka je otvorená na veľmi krátku dobu, čím sa do fotoaparátu dostane menej svetla. Je vhodná napríklad pri fotografovaní pohybu, keďže pohybujúci sa objekt „zamrazí“ v čase a vý-sledná fotografia je bez rozmazania.

Nižšia rýchlosť uzávierky (napr. $\frac{1}{30}$ sekundy) udrží uzávierku otvorenú dlhší čas, čo umožní, aby do fotoaparátu preniklo viac svetla. Výsledná fotografia sa teda môže javiť rozmazane a neostro, je ale svetlejšia.

Z pohľadu svetelného lúča rýchlosť uzávierky priamo neovplyvňuje ani nemení vlastnosti samotného svetelného lúča. Namiesto toho ovplyvňuje množstvo svetla, ktoré sa dostane na senzor fotoaparátu, čo v konečnom dôsledku ovplyvňuje svetlosť, rozmazanie pohybu a ostrosť výsledného obrazu. [15]

2.1.1.5 ISO

ISO je parameter, ktorý odkazuje na citlivosť média na svetlo, takže citlivosť filmu alebo digitálneho senzora. Umožňuje tak vytvárať svetlejšie fotografie za slabého osvetlenia alebo pri rýchlejších rýchlostiach uzávierky. Naopak, zníženie hodnoty ISO znižuje citlivosť digitálneho senzora alebo filmu na svetlo, čo vedie k tmavším fotografiám.

Typické hodnoty ISO zahŕňajú 100, 200, 400, 800, 1600, 3200 a vyššie. Nesprávne nastavené ISO môže mať za následok podexponované alebo preexponované fotografie, či zvýšený šum alebo zrnitosť. Spolu s clonou a časom tvoria takzvaný fotografický trojuholník, pričom všetky ovplyvňujú množstvo svetla, ktoré dopadne na snímač fotoaparátu. [16]

2.1.2 Charakteristika analógového fotografovania

Na začiatku druhého tisícročia sa vďaka rastúcej popularite a dostupnosti digitálnych fotoaparátov zdalo, že koniec fotoapáratov na film sa neodkladne

blíži. Nasvedčovala tomu nielen obľúbenosť u zákazníkov, ale aj čísla z predajov jednotlivých firiem. [17, 18]

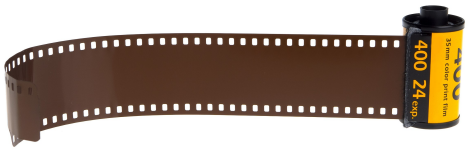
Avšak v posledných rokoch, hlavne počas pandiémie Covid-19 sa vo svete fotografie stretávame čoraz intenzívnejšie s návratom analógových fotoaparátov. Podľa New York Times a viceprezidenta online obchodnej platformy eBay, sa počet predaných fotoaparátov na film v roku 2021 oproti predošlému roku zdvojnásobil. [19]

Nasvedčujú tomu aj úspechy spoločnosti Kodak, ktorá v roku 2012 podala žiadosť o ochranu pred veriteľmi v súlade s americkým zákonom o ochrane pred bankrotom, čo je forma ochrany pre spoločnosti, ktoré nemôžu splácať svoje dlhy. Táto istá spoločnosť skončila rok 2020 s hotovostným zostatkom 196 miliónov amerických dolárov, informuje Guardian [20].

Čím si teda analógová fotografia zaslúžila tak rapídne zvýšenú popularitu? Nasledujúce podsekcie sa venujú pre priblíženie tejto problematiky venujú charakterizácií filmu a filmovej fotografie.

2.1.2.1 Film

Keďže je film vyrobený zo svetlocitlivého materiálu, je potrebné ho skladovať v svetlotesnej nádobe, aby sa zabránilo náhodnému osvetleniu, ako je znázornené na Obrázku 2.5. Po zachytení fotografie je potrebné film manuálne alebo automaticky pretočiť, aby bolo možné odfotiť ďalšiu snímku.

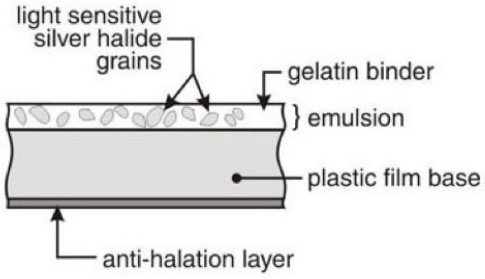


Obr. 2.5: Ukážka analógového filmu [21]

Časti filmu Analógový film sa skladá primárne z troch častí, ako je zobrazené na Obrázku 2.6. Podľa [22, 23, 24] sú nimi:

- **Emulzia** predstavuje najdôležitejšiu časť filmu, keďže je zodpovedná za zachytávanie obrazu. Skladá sa z mikroskopických kryštálov halogenidu striebra zasadených do želatínového materiálu.

  Farebný film obsahuje týchto vrstiev viacero, pre jednotlivé farebné zložky (červená, zelená a modrá). Každá vrstva emulzie je citlivá na určitú vlnovú dĺžku svetla a tak sú schopné vytvoriť farebný obraz.



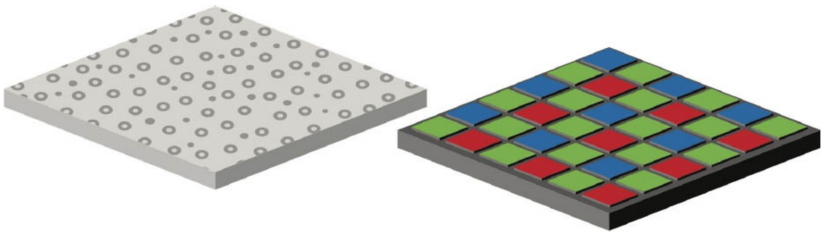
Obr. 2.6: Ukážka prierezu materiálu filmu [25]

Pri dopadnutí svetla na túto vrstu prebehne reakcia medzi striebornými halogenidmi a svetlom, prčiom sa strieborné ióny (Ag+) menia na strieborné atómy (Ag). Tieto atómy sa potom zhromaždia v okolí zrnka (malého kryštálika strieborného halogenidu), a tým vytvoria tmavý bod. Tento bod predstavuje mikroskopickú časť obrazu a je záznamom jasu a farebnej informácie. Pri opakovaní tohto procesu na viacerých kryštáloch sa vytvára komplexný obraz, ktorý je zachytený na filme.

Tvar, veľkosť a distribúcia kryštálov halogenidu striebra v želatínovom materiále určujú vzhľad zrna na výsledom obrázku. Menšie kryštály produkujú jemnejšie zrno a vyššie rozlíšenie, no vyžadujú viac svetla na správnu expozíciu. Väčšie kryštály sú citlivejšie na svetlo, čím dokážu zachytiť snímky v horších svetelných podmienkach, no zároveň vytvárajú viditeľnejšie zrno.

V porovnaní s digitálnou fotografiou, pravidelné usporiadanie pixelov na digitálnom snímači spôsobuje, že na obrázku sa objavujú diagonálne čiary. Sú zobrazené ako zubaté, drobné hrany – tento jav sa nazýva *aliasing*. Aliasing sa stáva výraznejším, keď sa uhol diagonály približuje usporiadaniu na snímači. Pri farebnom snímači môže aliasing spôsobiť *moiré* efekt alebo miestne skreslenie farieb v obraze. Filmová fotografia s organickou distribúciou zrnitosti, nie je na takéto pravidelné vzory citlivá, ako je možné vidieť v porovnaní na Obrázku 2.7.

Výrobcovia digitálnych fotoaparátov používajú rôzne metódy na zabezpečenie čistého zobrazenia takýchto štruktúr: do fotoaparátov pridávajú takzvaný antialiasingový filter, ktorý mierne rozmazáva obraz počas snímania. Týmto spôsobom sa zabráni vzniku zubatých hrán na úkor ostrosti obrazu. Preto je často potrebné digitálne obrázky zaostriť v poslednom kroku pri následnom spracovaní. Filmová fotografie tento krok nevyžaduje.



Obr. 2.7: Ukážka porovnania zrna (naľavo) a pixelov (napravo) [22]

- **Báza** je pružný, priehľadný plastový materiál, ktorý poskytuje oporu pre ostatné vrstvy filmu. Zvyčajne sa vyrába z acetátu celulózy alebo polyesteru. Musí byť stabilná a odolná voči roztrhnutiu alebo poškodeniu.

- **Antihalačná vrstva** sa nachádza pod vrstvou emulzie alebo na zadnej strane bázy. Jej úlohou je zabraňovať odrazu svetla späť do vrstvy emulzie. Zvyčajne sa skladá z čierneho farbiva alebo pigmentu v želatínovej vrstve.

Formát a počet snímkov Formát filmu sa vzťahuje na veľkosť a tvar filmu, pričom rôzne veľkosti majú rôzne pomery strán. Súvisí s nimi aj počet snímkov, ktoré je na film možné zachytiť. Najčastejšie sa podľa [26, 27] rozdeľujú na:

- **110 film** je najmenší vyrábaný formát, pričom veľkosti jeho strán sú 13x17mm. Nejakú dobu sa vôbec nepredával, no dnes sú tieto filmy opäť dostupné vďaka firme Lomography. Je možné naň nafotiť 12, 20 alebo 24 snímkov.

- **35mm film** (odvodené zo šírky filmu v milimetroch), sa tiež nazýva *kinofilm*, či 135 film a je najpopulárnejším fotografickým filmom. Veľkosti jeho strán sú 24x36mm a oproti strednému formátu ponúka menej detailov a viac zrna. Tento formát taktiež ponúka 3 možnosti počtu snímkov – 12, 24, či 36.

- **Stredoformátový film** alebo zvitkový film, či 120 film, označuje film so šírkou cca 61 mm – formát teda nesúvisí s jeho šírkou. Zvitok sa používa u stredoformátových fotoaparátov a je možné ho podľa formátu fotoaparátu nafotiť v rôznych pomeroch. Je možné naň nafotiť 8, 10, 12 alebo 16 snímkov.

- **Veľkoformátový film** označuje akýkoľvek formát obrazu o rozmeroch 9x12cm alebo väčší. Hlavnou výhodou je vyššie rozlíšenie, ktoré umožňuje fotiť aj pri slabšom osvetlení. Od ostatných formátov sa odlišuje taktiež tým, že na každú fotku používa individuálny film namiesto súvislého kotúča.

2.1.2.2 Od natiahnutia filmu k výslednej fotografii

Ako je teda možné premeniť odrazený svetelný lúč na fyzickú fotografiu, ktorá zachytí daný moment prakticky navždy? Tento proces je možné vďaka [22, 28] zhrnúť v nasledujúcich krokoch:

1. **Vloženie filmu do fotoaparátu** je síce očividný, ale veľmi dôležitý krok, nakoľko bez vloženého filmu by nebolo možné snímky zachytávať. Je potrebné sa uistiť, že film je vložený správne, aby sa dal posúvať a tak fotiť každú fotografiu na novú prázdnu časť filmu.

2. **Zachytenie snímkov** s požadovanými nastaveniami fotoaparátu, korespondujúcimi so svetelnými podmienkami. Po každom snímku je potrebné film manuálne alebo automaticky (podľa možností fotoaparátu) pretočiť.

3. **Previnutie filmu** po dofotení všetkých snímkov. Niektoré fotoaparáty dokážu film previnúť naspäť do plastovej časti automaticky. Dôležité je film chrániť pred svetlom, aby sa nezničili vyfotené fotografie.

4. **Vyvolanie filmu** za použitia špecifických chemikálií. Do tohto bodu fotografie na filme nie sú vidieť, tvoria len takzvaný latentný obraz. Vyvolanie filmu pozostáva z nasledovných krokov, vykonávaných v absolútne zatemnenej miestnosti:

    a) **Vloženie filmu do vývojnice**, čo je zvyčajne plastová nádoba, v ktorej celý proces prebieha. Najskôr je ale potrebné film v tme navinúť na cievku, odstrihnúť od plastového dielu a až tak vložiť do vývojnice.

    b) **Naliatie vývojky do nádoby.** Vývojka slúži na premenu strieborných halogenidov na kovové striebro. Nádobu s vývojkou a navinutým filmom na cievke je potrebné preklápať zo strany na stranu, pričom je dôležité sledovať dobu, počas ktorej strieborné halogenidy s vývojkou reagujú. Táto doba je určená typom vývojky a filmu. Po jej uplynutí je potrebné vývojku z nádoby vyliať.

    c) **Naliatie prerušovacieho roztoku do nádoby.** Prerušovací roztok zastavuje proces vyvolávania a neturalizuje vývojku. Keby sme nepoužili prerušovací roztok, vývojka by regovala so striebornými halogenidmi príliš dlho a fotografiu zničila. Nádobu je opäť potrebné pretáčať zo strany na stranu počas určenej doby a potom roztok vyliať.

    d) **Použitie ustálovača**, ktorý odstraňuje nevyvinuté strieborné halogenidy a stabilizuje obraz, rovnakým postupom ako v predošlých krokoch.

e) **Vyčistenie filmu vo vode**, ktoré zaručí odstránenie ustálovača a prebytkových chemikálií.

f) **Vybratie filmu z nádoby a jeho usušenie.**

V každom zo spomenutých krokov je možné zmenami časov, alebo zloženiami jednotlivých látok rôzne ovplyvňovať výzor výslednej fotografie. Výsledkom tohto kroku je negatív, na ktorom už je síce vidieť výsledný obrázok, ale s prevrátenými farbami. Takže svetlé časti sa javia tmavo a vice versa.

5. **Vytlačenie fotografií** zo vzniknutého negatívu na svetlocitlivý papier, pomocou zväčšovacieho prístroja. Do zväčšovacieho prístroja sa umiestni vybraný negatív a pod prístroj fotocitlivý papier, ktorý je následne ožiarený svetlom. Podobne ako na filme, najskôr na papieri vznikne latentný obraz.

6. **Vyvolanie fotografií** z latentného obrazu na fotocitlivom papieri. Prebieha podobne ako vyvolanie filmu za použitia vývojky, prerušovacieho roztoku a ustálovača. Po dokončení je opäť potrebné hotové fotografie umyť a usušiť.

2.1.3 Charakteristika digitálneho fotografovania

Nasledujúce podsekcie sa venujú definícií digitálneho senzoru, ako aj popísaniu procesu zachytenia fotografie na daný senzor.

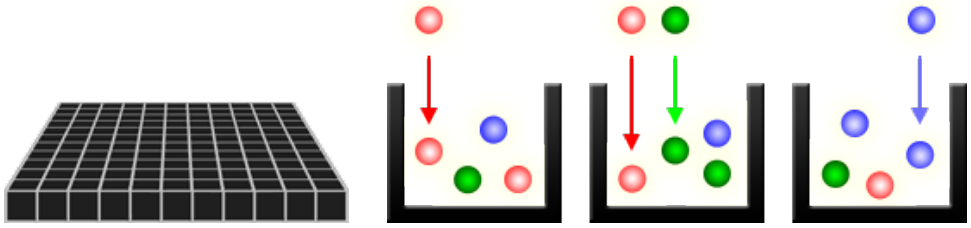
2.1.3.1 Snímač

Obrazový snímač, nazývaný aj senzor je polovodičové zariadenie, ktoré zachytáva svetlo potrebné na vytvorenie digitálneho obrazu. Zjednodušene sa dá povedať, že keď sa otvorí uzávierka, snímač zachytí fotóny (elementárne častice, zodpovedné za prenos elektromagnetického žiarenia) a prevedie ich na elektrický signál, ktorý procesor vo fotoaparáte prečíta a interpretuje ako farby. Tieto informácie sa potom spoja do jedného obrázku.

Zložitejšia odpoveď je, že snímač sa skladá z miliónov dutín nazývaných „fotostránky“, zobrazených na Obrázku 2.8a. Tieto fotostránky sa otvoria pri otvorení uzávierky, zachytia fotóny a zatvoria sa po dokončení expozície. Počet fotostránok je rovnaký ako počet pixelov, ktoré má daný fotoaparát.

Fotóny, ktoré zasiahli jednotlivé fotostránky, zobrazené na Obrázku 2.8b, sa interpretujú ako elektrický signál, ktorého sila sa mení podľa toho, koľko fotónov bolo skutočne zachytených v dutine. Ak by sme sa ale pozreli na obrázok, ktorý bol nasnímaný len s elektrickými údajmi zo snímača, obrázky by boli v skutočnosti v odtieňoch šedej.

Na zachytenie farebných obrázkov musí byť nad každú fotostránku umiestnený filter, ktorý prepúšťa len určité farby svetla. Prakticky všetky súčasné



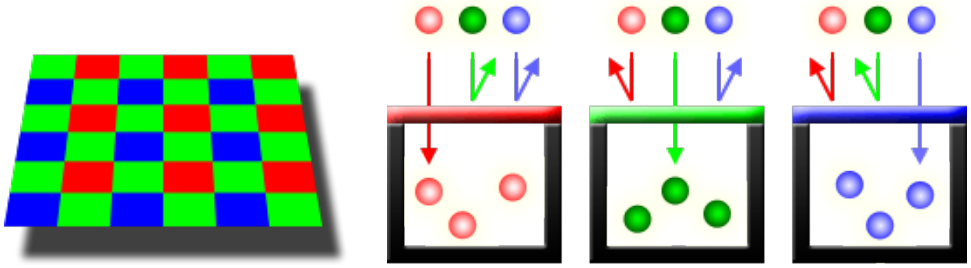
(a) Ukážka poľa fotostránok tvoriacich snímač [29]

(b) Ukážka jednotlivých fotostránok s fotónmi [29]

Obr. 2.8: Ukážky fotostránok

digitálne fotoaparáty dokážu zachytiť iba jednu z troch základných farieb v každej dutine a neprepúšťajú približne dve tretiny prichádzajúceho svetla.

To má za následok, že fotoaparát musí aproximovať ďalšie dve základné farby, aby mal každý pixel plnú farbu. Najbežnejší typ poľa farebných filtrov sa nazýva pole Bayerových filtrov, naznačeného na Obrázku 2.9a. Bayerov filter je farebný filter umiestnený na vrchu každej fotostránky, ako je zobrazené na Obrázku 2.9b. Používa sa na určenie farby obrazu na základe toho, ako sa merajú elektrické signály zo susedných fotoschránok.



(a) Ukážka poľa Bayerových filtrov [29]

(b) Ukážka jednotlivých fotostránok s Bayerovým filtrom [29]

Obr. 2.9: Ukážky fotostránok s farebným filtrom

Filre môžu byť červenej, zelenej a modrej farby, s pomerom jedna červená, jedna modrá a dve zelené v každej sekcii štyroch fotostránok, keďže ľudské oko je senzitívnejšie na zelenú farbu. Nadbytok zelenej vytvára menej zašumený obraz s jemnejšími detailami, ako by sa dalo dosiahnuť, keby sa s každou farbou zaobchádzalo rovnako.

Obrazový snímač teda primárne zachytáva svetlo a premieňa ho na elektrický signál, čo je analógový proces. v tejto fáze sú teda informácie stále v analógovej forme. Po vygenerovaní analógového signálu je potrebné previesť ďalšie ktory, ktoré z neho vytvoria digitálny obraz. [29, 30]

2.1.3.2 Od fotónov k výslednej fotografii

Po zachytení elektrického signálu s farebnou informáciou, je potrebné ho spracovať do formy digitálneho obrázku, pozostávajúceho z jednotlivých pixelov. Tento proces je podľa [31] možné zhrnúť do nasledovných krokov, znázornených aj na Obrázku 2.10:

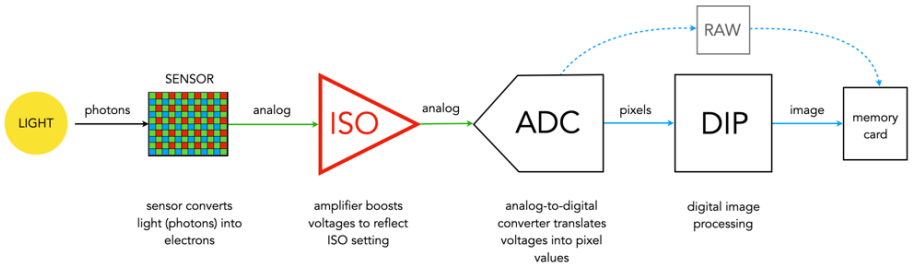
1. **ISO zosilňovač upraví signál** na základe ISO nastavení. Používa násobiaci faktor M, na vynásobenie počtu elektrónov na základe nastavenia ISO fotoaparátu. Pre vyššie ISO bude M vyššie a vyžaduje menej elektrónov.

2. **ADC (analógovo-digitálny prevodník) konvertuje analógové signály na diskrétne digitálne hodnoty**, ktoré už sa dajú označiť za pixely. Analógové signály klasifikuje do úrovne jasu (v podstate do matice pixelov).

   Ak je fotoaparát nastavený na RAW, tak sa snímka uloží vo formáte RAW na pamäťovú kartu. Ak je nastavenie RAW+JPEG alebo JPEG, môžu byť pixle ešte ďalej spracované.

3. **DIP systém, alebo aj systém digitálneho spracovania obrazu, použije algoritmus *demosaicing***, ktorý konvertuje pixely v matici na RGB obrázok. na základe konkrétnych nastavení fotoaparátu je možné použiť aj iné techniky spracovania obrazu, napríklad doostrenie obrazu, redukcia šumu a podobne.

4. **Upravený obrázok je uložený na pamäťovú kartu.**



Obr. 2.10: Ukážka procesu spracovania elektrického signálu do podoby digitálneho obrázku [31]

2.1.4 Výhody a nevýhody

Je takmer nemožné určiť, ktorý typ fotografie je lepší či horší, a v konečnom dôsledku to aj tak závisí na preferenciách fotografa. Najvýraznejšie výhody a nevýhody je ale podľa [32, 33] možné zhrnúť do nasledovných bodov:

- **Cena** digitálneho fotoaparátu sa šplhá častejšie oveľa vyššie ako cena analógového, obzvášť toho z druhej ruky. Avšak náklady na kúpu filmov, vyvolávanie a tlač fotiek môžu filmovú fotografiu z dlhodobého hľadiska predražiť.

- **Vzhľad** filmovej fotografie zaručuje charakteristický dojem, širokú farebnú paletu a jemnosť.

- **Rozlíšenie**, teda level detailu obsiahnutý vo fotke, sa pri digitálnej fotografii zvyčajne meria v megapixeloch. Pri fotení na film je ťažké ho určiť, keďže je zrno vo filme nerovnomerne rozdistribuované a má rôzne veľkosti. Aj napriek tomu, že mnohé analógové filmy ponúkajú vysoké rozlíšenie platí, že digitálne fotoaparáty vo všeobecnosti poskytujú rozlíšenie vyššie.

- **Hmatateľné médium** v podobe negatívov alebo tlačených fotografií pri analógoch poskytuje pocit zadosťučinenia a hlbšie prepojenie s umeleckým dielom. Digitálne fotografie je samozrejme taktiež možné vytlačiť, avšak ľudia častejšie uchovávajú svoje fotografie v digitálnych úložiskách.

- **Spätná väzba** digitálneho fotoaparátu prostredníctvom LCD obrazoviek umožňuje fotografom prezerať snímky a instantne vykonávať úpravy. Pri filmovej fotografií je potrebné najskôr dofotiť film a vyvolať ho, čo zaberá viac času.

- **Úpravy fotografií** sú častejšou praktikou pri fotení na digitálny fotoaparát a existuje množstvo softwarov, ktoré to umožňujú. Na druhej strane, pri správnom nastavení analógového fotoaparátu fotografie zväčša netreba doupravovať, takže fotografovi šetria čas.

- **Flexibilita** digitálnej fotografie umožňuje vyššiu kontrolu nad nastaveniami ako je ISO, rýchlosť uzávierky, clona a vyváženie bielej, čo dovoľuje fotografom prispôsobiť sa rôznym svetelným podmienkam.

- **Všímavosť** fotografa sa jednoznačne prehlbuje pri analógovej fotografií, keďže s každým snímkom sú spojené náklady. To vedie fotografov k vyššiemu sústredeniu sa na prítomný moment a zachytenie okamihu.

- **Archivovanie** filmových negatívov správnym spôsobom môže zaručiť ich životnosť na mnoho desaťročí, dokonca aj viac ako storočie. Vďaka

tomu je filmová fotografia spoľahlivou možnosťou dlhodobého uchovania snímok. Naproti tomu archivácia digitálnych obrázkov môže byť problematická, keďže digitálne pamäťové médiá môžu časom zastarať alebo degradovať, čo vedie k potenciálnej strate fotografií.

2.1.5 Možnosti úprav digitálnych fotografií

Z poznatkov predchádzajúcich kapitol o fungovaní filmu a digitálneho senzoru je zrejmé, že každé médium má iné vlastnosti a prvky, ktoré sa nedajú ľahko nahradiť. Avšak existuje mnoho spôsobov, ako aspoň čiastočne napodobniť vzhľad analógovej fotografie, s teplými tónmi a filmovým efektom na fotografií digitálnej. Je možné tak uskutočniť aplikáciou rôznych efektov a filtrov, popísaných v nasledujúcich častiach práce.

Zosvetlenie odleskov čiernej

Analógový film nie je schopný dosiahnuť čistú čiernu farbu, kvôli hustote emulzie a bázy z ktorých je tvorený. Jemné zosvetlenie výskytov čiernej farby na Obrázku preto dokáže napodobniť túto nedokonalosť filmu a vytvára vyblednutý efekt s nižším kontrastom. [34]

Docieliť tento krok je možné určením prahovej hodnoty pre jednotlivé pixely, ktorá identifikuje tmavé oblasti obrázku, ktoré je potreba upraviť. Pomocou transformačnej funkcie je potom potrebné namapovať identifikované tmavé pixely na vyššie hodnoty, pričom sa jas najtmavších pixelov zvýši viac ako jas pixelov bližie k určenej prahovej hodnote. Pixely nad prahovou hodnotou tak zostanú nezmenené. [35]

Príkladom takéhoto postupu je iterovanie cez jednotlivé pixely a zvyšovanie ich jasu pod prahovou hodnotou o určité percento.

Ztmavenie odleskov bielej

Ztmavením odleskov bielej je možné imitovať viaceré vlastnosti filmovej fotografie, ako napriklad zníženie celkového rozdielu medzi tmavými a svetlými oblasťami. To pomôže zvýrazniť tóny farieb v strede tejto škály, vďaka ktorým dostane fotografia viac nostalgický ráz. [34].

Je možné tak učiniť rovnakým spôsobom ako v predošlom kroku, avšak s opačným postupom.

Rozostrenie

Rozostrením alebo rozmazaním určitých častí obrázku je možné aspoň čiastnočne simulovať jemnejší a organickejší vzhľad analógových fotografií. Nakoľko ale filmy dokážu detailne zachytiť scenérie, je dôležité uchovať detaily obsiahnuté na hranách objetov fotografie. [34]

Docieliť tohto efektu je možné napríklad použitím selektívneho rozmazania, ako je bilaterálny filter, ktorý zachováva okraje, ale znižuje šum. Názov bilaterálny pochádza z podstaty filtru, keďže berie do úvahy dva druhy váh. Postupným prechádzaním pixelov fotografie, sa pomocou týchto váh a váženého priemeru dopočítavajú hodnoty pre daný pixel z okolitých pixelov.

Prvá váha je založená na priestorovej blízkosti susedných pixelov. Aplikuje sa v rámci definovaného okolia (ako aj iné rozmazávacie filtre), kde priradí vyššiu váhu pixelom bližšie k pixelu v strede. Vďaka tomu umožňuje bilaterálnemu filtru vyhladiť oblasti s podobnou farbou.

Druhá váha berie do úvahy podobnosti intenzity okolitých pixelov. Pixely s podobnými intenzitami ako pixel v strede dostanú vyššiu váhu, zatiaľ čo pixely s veľkými rozdielmi intenzít dostávajú menšiu váhu alebo sú z priemerovacieho procesu úplne vylúčené. Táto váha založená na intenzite pomáha zachovať hrany a ostré prechody v intenzite. [36]

Úprava farebných tónov

Mnoho populárnych filmov, ako napríklad Kodak Gold má charakteristickú farebnú paletu, kde tiene majú teplejší podtón a svetlé časti obrázku naopak studený. Aplikáciou tohto efektu je možné taktiež evokovať dojem hĺbky v obraze, takže sa fotografia javí menej plochá. Táto vlastnosť sa často spája práve s analógovými fotoaparátmi.

Simulovať tento efekt sa dá rôznymi spôsobmi, napríklad pomocou vyhľadávacích tabuliek. Tieto tabuľky obsahujú výsledné hodnoty jednotlivých zložiek pixelov, na ktoré sú namapované ich vstupné hodnoty.

Zvýšenie teploty pixelu sa dá docieliť zvýšením hodnoty jeho červenej a zelenej zložky. Zvýšenie červenej zložky by ale malo byť výraznejšie ako zelenej. Modrú zložku je vhodné naopak znížiť. Zníženie teploty pixelu je možné obratením popísaného postupu. [37]

Zrno

Pridanie efektu zrna do digitálnej fotografie vychádza z prirodzených vlastností filmu a kryštálov halogenidu strieborného obsiahnutých v jeho materiáli. Okrem estetického prínosu môže zrno taktiež zamaskovať digitálny šum. [34]

Na simuláciu zrna je vhodné vytvoriť novú vstvu obrázku o rovnakej veľkosti ako je pôvodná fotografia. Pre každý pixel tejto vrstvy je potom potrebné náhodne vygenerovať hodnotu intenzity zrna, z čoho pri vhodnom výbere distribúcie náhodných hodnôt vznikne vzor podobajúci sa zrnu. Túto vrstvu je následne potrebné aplikovať na pôvodný obrázok, napríklad prenásobovaním jednotlivých pixelov s určenou váhou.

Vinetácia

Aj napriek tomu, že pri novších analógových filmoch a fotoaparátoch sa vinetácie už moc často neobjavovala, stále patrí k charakteristickým znakom tejto fotografie. Vznikala predovšetkým kvôli obmedzeniam a nedokonalostiam skorých objektívov fotoaparátov. Prejavuje sa ako postupné stmavnutie obrazu smerom k rohom a okrajom.

Je možné jú docieliť vytvorením novej vrstvy obrázku o rozmeroch pôvodnej fotografie, ktorej hodnoty pixelov reprezentujú mieru ztmavnutia výsledného pixelu. na okrajoch fotografie by sa teda mali nachádzať tmavšie pixely a v strede naopak bledšie, poprípade úplne bez zmeny. Prechod zo stredu na okraje by mal byť postupný a prirodzený. Túto vrstu je následne opäť potrebné aplikovať na pôvodný obrázok, napríklad prenásobením pixelov. [38]

Svetelné odlesky

Svetelné odlesky, či úniky svetla sú spôsobené neúmyselným osvetlením filmu, vytvárajuce rôzne farebné pruhy, zahmlenie alebo posuvy farieb. Sú výsledkom nedokonalostí tela fotoaparátu, antihalačnej vrstvy alebo nesprávnej manipulácie s filmom.

Simulovať ich je možné vytvorním a aplikáciou novej vrstvy obrázku s požadovanými odleskami. Odlesky je možné vytvoriť pomocou vygenerovania náhodného vzoru tvarov ako sú napríklad kruhy a elipsy, poprípade ich kombináciou. Tieto vzory by mali mať jemný, difúzny vzhľad, napodobňujúci spôsob, akým by sa svetlo objavilo pri nechcenom úniku svetla na film. [39]

2.2 Rešerš existujúcich aplikácií pre iOS

Analýza a rešerš existujúcich konkurenčných aplikácií simulujúcich analógovú fotografiu je dôležitá pre popísanie ich slabých a silných stránok. Pomocou nich je možné lepšie zadefinovať požiadavky na výslednú aplikáciu, ktoré sa inšpirujú kladnými aspektami analyzovaných aplikácií a eliminujú identifikované nedostatky.

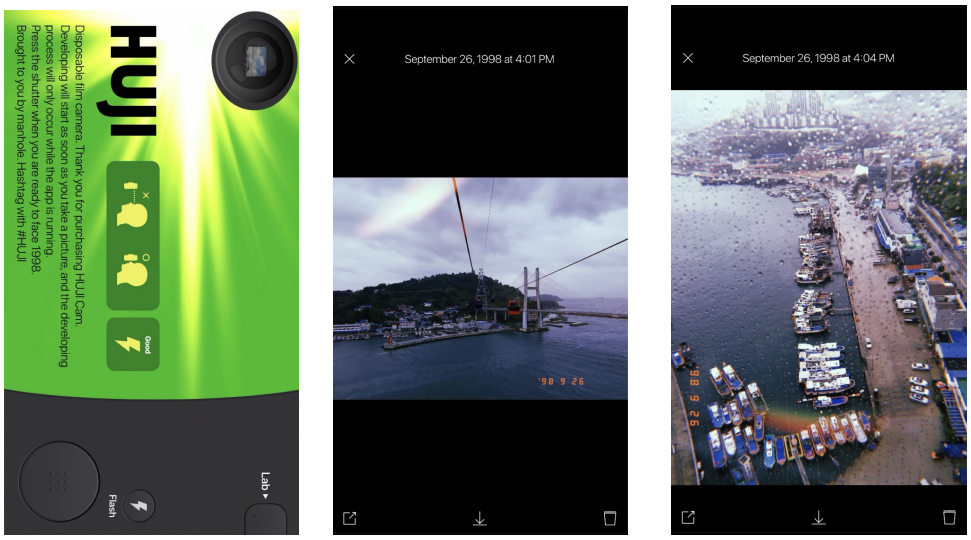
Nasledujúce podsekcie sa venujú trom aplikáciám s podobnou funkcionalitou, ich popisu a zadefinovaniu slabých a silných stránok.

2.2.1 Huji Cam

Aplikácia Huji je jednou z najpopulárnejších aplikácií napodobňujúcich jednorázové analógové fotoaparáty z konca 90tych rokov. Disponuje len jedným filtrom, ktorý používa silné svetelné odlesky a výrazne upravuje farby do oranžových tónov. Výsledný vzhľad fotografie potom evokuje jednorázové analógové fotoaparáty, ale nezachováva si teplo, jemnosť a hĺbku klasických filmov.

Fotografiu je možné odfotiť buď cez malý alebo veľký hľadáčik, čo pridáva pocit hľadáčiku na reálnom fotoaparáte. Po odfotení fotografie je možné si ju hneď prezrieť v gálérií. Nie je ale možné do aplikácie nahrávať už existujúce fotografie, takže si zachováva funkcionalitu fotoaparátu a nie aplikácie na úpravu fotiek.

Celkovo aplikácia pôsobí prívetivo pre užívateľa a až na veľkosť niektorých ikoniek je veľmi intuitívna. Je možné ju používať v bezplatnej verzii ale ponúka aj Premium balíček, ktorý rozširuje jej funkcionalitu.



Obr. 2.11: Ukážky aplikácie Huji Cam [40]

2.2.2 Gudak

Aplikácia Gudak má platenú aj neplatenú verziu aplikácie a tieto verzie fungujú samostatne. Zachováva koncept fyzických filmov tak, že pri začiatku fotenia je nutné zvoliť druh filmového efektu, ktorý je následne aplikovaný na 24 budúcich fotografií, tak ako pri natiahnutí filmu do fotoaparátu. Týchto 24 fotografií ale musí užívateľ vyfotiť v priebehu jednej hodiny a aplikáciu nesmie zavrieť, inak fotografie stratí. Tým aplikácia popiera vlastnosti filmu, keďže reálny film je možné mať natiahnutý vo fotoaparáte aj niekoľko mesiacov.

Vzhľad efektu sa ale približuje vzhľadu analógového filmu, takže hlavnú funkcionalitu – simuláciu analógu – spĺňa.

Pre simuláciu procesu vyvolania filmu má pevne stanovených 72 hodín, počas ktorých nemá užívateľ prístup k výsledným fotografiám. Touto vlastnosťou výrazne obmedzuje užívateľa a opäť popiera vlastnosti filmu, keďže pri domácom ale aj profesionálnom vyvolávaní fotografií je možné ich mať odfotené aj vyvolané v priebehu jedného dňa.

Vzhľad aplikácie je jednoduchý, avšak prechod do galérie neintuitívny pre iOS zariadenia.

Obr. 2.12: Ukážky aplikácie Gudak [41]

2.2.3 Hipstamatic

Mobilná aplikácia Hipstamatic ponúka široké spektrum filtrov analógových filmov, pričom pre odomnkutie všetkých možností je potrebné použitie platenej verzie. Aplikácia dovoľuje užívateľom fotografovať fotky priamo v aplikácií, ale aj nahrávať do nej fotky už odfotené.

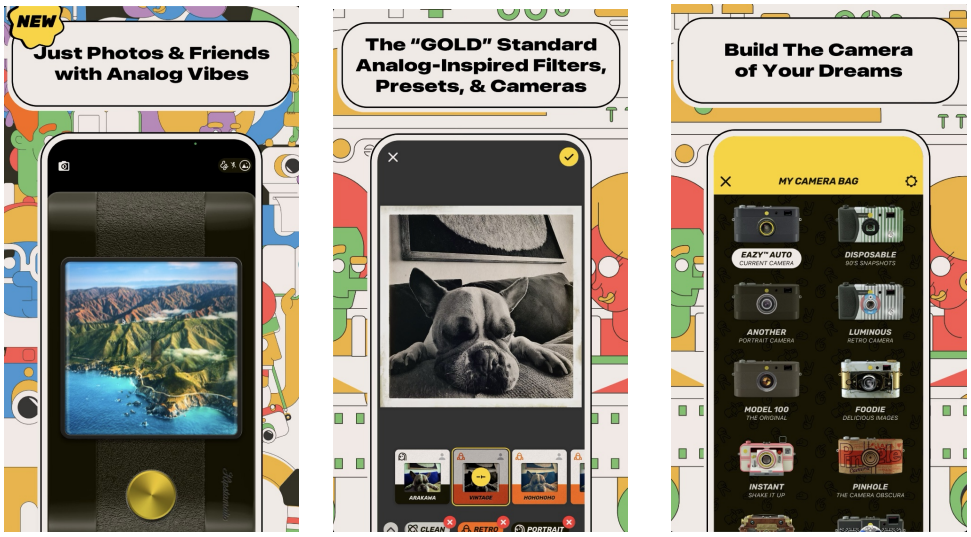
Funguje čiastočne aj ako sociálna sieť, kde je možné pridávať svoje fotky do verejného profilu a spájať sa s ostatnými užívateľmi. Túto funkcionalitu je možné považovať aj za plus, keďže rozširuje atraktivitu aplikácie, ale aj za mínus, pretože môže spôsobovať zmätok, keď chce používateľ len zachytiť fotku.

Fotografie zhotovené pomocou Hipstamatic majú štvorcový formát bez možnosti jeho zmeny. Filtre simulujúce film sú výrazne exponované, s vysokým kontrastom, takže tak ako pri aplikácií Huji strácajú jemnosť filmu. Sú dostupné hneď po odfotení v galérií.

V aplikácií nie je rozlíšené, ktorá fotografia bola odfotená s akým filmovým efektom, takže koncept fyzických filmov nezachováva a fotografie mixuje dokopy v galérií.

2.2.4 Zhrnutie existujúcich aplikácií

Popísanie slabých a silných stránok konkurenčných aplikácií dovoľuje lepšie zhodnotiť prínosy jednotlivých funkcionalít a vizuálnych prvkov aplikácií.



Obr. 2.13: Ukážky aplikácie Hipstamatic [42]

Uvedená rešerš teda poslúži ako podklad pre analýzu požiadavkov uvedených v nasledujúcej sekcii.

2.3 Požiadavky

Pred začiatkom vývoja aplikácie je dôležité zadefinovanie požiadavkov na danú aplikáciu, pre vymedzenie hraníc systému, odhad pracnosti a celkové vyjasnenie zadania. Identifikácia požiadavkov ovplyvňuje všetky nasledovné fázy vývoja, takže je dôležité ich správne a detailne určiť hneď na začiatku.

Jednotlivé požiadavky môžu mať uvedené rôzne informácie pre ich špecifikáciu, v tejto práci sú k požiadavkám uvedené nasledovné údaje, podľa [43]:

- **Identifikátor** uľahčuje orientáciu a odkazovanie na daný požiadavok.
- **Názov** stručne popisuje požiadavok.
- **Popis** tvorí najdôležitejšiu časť, v ktorej je požiadavok popísaný presnejšie.
- **Priorita** splnenia požiadavku určuje jeho dôležitosť pri vývoji, pri nízkej priorite môže byť z rôznych dôvodov nesplený. Prioritu požiadavkov je možné definovať pomocou metódy MoSCoW, ktorá požiadavky delí do štyroch kategórií [44]:

  1. **Must have** požiadavky musia byť splnené za každých okolností. Bez ich splnenia by výsledná aplikácia nefungovala.

2. **Should have** požiadavky sú taktiež veľmi dôležité pre projekt, avšak ak na ne nezostane čas či peniaze, môžu byť pridané do budúcich verzií systému.
3. **Could have** požiadavky nie sú nevyhnutné pre základnú funkčnosť aplikácie, takže sa spracúvajú len ak sú splnené predošlé dve kategórie.
4. **Will not have** požiadavky sú voliteľné a zväčša špecifikujú budúce verzie systému.

Správne zadefinovaný požiadavok by mal podľa [43] ďalej spĺňovať nasledujúce tri vlastnosti:

- **Jednoznačnosť** požiadavku zaručuje jeho jasnosť a konkrétnosť a vystihnutie jeho cieľ.
- **Splniteľnosť** zaručuje, že požiadavok je možné realizovať.
- **Overiteľnosť** zodpovedá za to, že je požiadavok možné otestovať.

Jednotlivé požiadavky sa podľa [45] delia do dvoch základných kategórií:

- **Funkčné požiadavky** definujú základné vlastnosti a funkčnosť aplikácie a určujú, čo má systém robiť. Taktiež popisujú akými výstupmi ma systém reagovať na dané vstupy.
- **Nefunkčné požiadavky** špecifikujú, akým spôsobom má systém fungovat pre splnenie určenej funkcionality. Majú dopad na celkový výkon systému.

Na Obrázku 2.14 sa nachádza diagram požiadavkov, v ktorom sú zhrnuté funkčné a nefunkčné požiadavky na vyvíjanú aplikáciu. Nasledujúce podsekcie sa venujú ich detailnejšiemu popisu a prioritizácií.

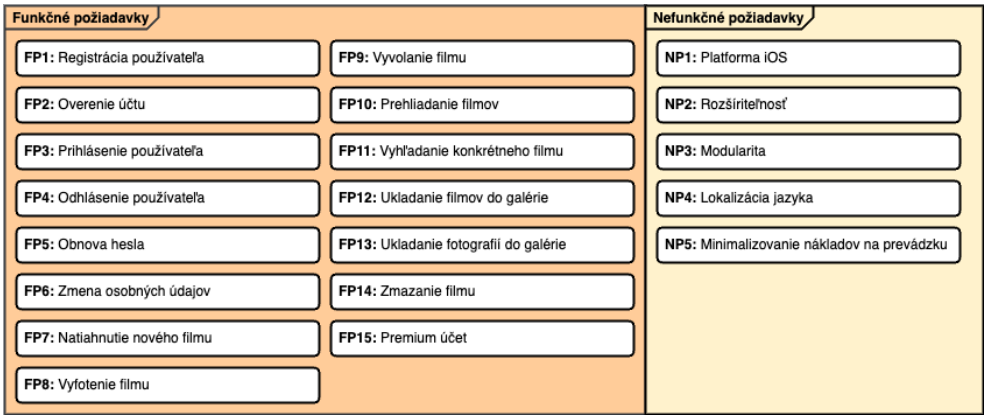
2.3.1 Funkčné požiadavky

FP1 – Registrácia používateľa

- **Popis:** Užívateľ si môže v aplikácií vytvoriť nový účet.
- **Priorita:** Must Have

FP2 – Overenie účtu

- **Popis:** Pre úspešné dokončenie registrácie musí používateľ overiť svoju e-mailové adresu.
- **Priorita:** Could Have



Obr. 2.14: Diagram požiadavkov, podľa [43]

FP3 – Prihlásenie používateľa

- **Popis:** Používateľ sa môže do aplikácie prihlásiť pomocou e-mailu a hesla.
- **Priorita:** Must Have

FP4 – Odhlásenie používateľa

- **Popis:** Prihlásený používáteľ sa môže z aplikácie odhlásiť.
- **Priorita:** Must Have

FP5 – Obnova hesla

- **Popis:** Používateľ si môže pri zabudnom hesle pri prihlísení heslo obnoviť.
- **Priorita:** Should Have

FP6 – Zmena osobných údajov

- **Popis:** Používateľ môže zmeniť svoje osobné údaje, zahŕňajúce meno a priezvisko.
- **Priorita:** Should Have

FP7 – Natiahnutie nového filmu

- **Popis:** Používateľ môže nastaviť, na aký typ a veľkosť filmu bude fotiť fotografie.
- **Priorita:** Must Have

FP8 – Vyfotenie filmu

- **Popis:** Používateľ môže na natiahnutý film vyfotiť daný počet fotografií.
- **Priorita:** Must Have

FP9 – Vyvolanie filmu

- **Popis:** Používateľ môže natiahnutý film v aplikácií vyvolať.
- **Priorita:** Must Have

FP10 – Prehliadanie filmov

- **Popis:** Používateľ si môže prehliadať vyvolané filmy a fotky, ktoré obsahujú.
- **Priorita:** Must Have

FP11 – Vyhľadanie konkrétneho filmu

- **Popis:** Používateľ môže prehľadávať filmy podľa mena.
- **Priorita:** Could Have

FP12 – Ukladanie filmov do gálerie

- **Popis:** Používateľ môže ukladať celé filmy z aplikácie do galérie telefónu.
- **Priorita:** Must Have

FP13 – Ukladanie fotografií do gálerie

- **Popis:** Používateľ môže ukladať jednotlivé fotografie z aplikácie do galérie telefónu.
- **Priorita:** Should Have

FP14 – Zmazanie filmu

- **Popis:** Používateľ môže zmazať celý film z aplikácie.
- **Priorita:** Must Have

FP15 – Premium účet

- **Popis:** Používateľ môže aktivovať alebo deaktivovať Premium účet.
- **Priorita:** Will Not Have

2.3.2 Nefunkčné požiadavky

NP1 – Platforma iOS

- **Popis:** Aplikácia je vyvíjaná pre iOS platformu.
- **Priorita:** Must Have

NP2 – Rozšíriteľnosť

- **Popis:** Aplikácia umožňuje jednoduchú rozšíriteľnosť v rámci pridávania nových typov filmových filtrov.
- **Priorita:** Must Have

NP3 – Modularita

- **Popis:** Aplikácia je zložená z viacerých modulov a zmena jedného modulu má minimálny dopad na ostatné.
- **Priorita:** Should Have

NP4 – Lokalizácia jazyka

- **Popis:** Aplikácia dokáže používať slovenský aj anglický jazyk.
- **Priorita:** Could Have

NP5 – Minimalizovanie nákladov na predvádzku

- **Popis:** Aplikácia využíva primárne bezplatné služby a platformy, pre minimalizovanie nákladov na jej vývoj a zaručenie toho, že výslednú aplikáciu je možné používať aj zadarmo.
- **Priorita:** Must Have

2.4 Model prípadov použitia

Model prípadov použitia detailnejšie popisuje a vysvetľuje požadovanú funkcionalitu. Zaoberá sa teda upresnením funkčných požiadavkov a ich vysvetlením na konkrétnych príkladoch – prípadoch použitia. Samotný prípad použitia predstavuje interakciu medzi používateľom (človekom alebo strojom) a systémom [43, 46].

Model prípadov použitia tvorí zoznam aktérov, diagram prípadov použitia a zoznam prípadov použitia [43], zadefinovaných a popísaných v nasledujúcih podsekciách.

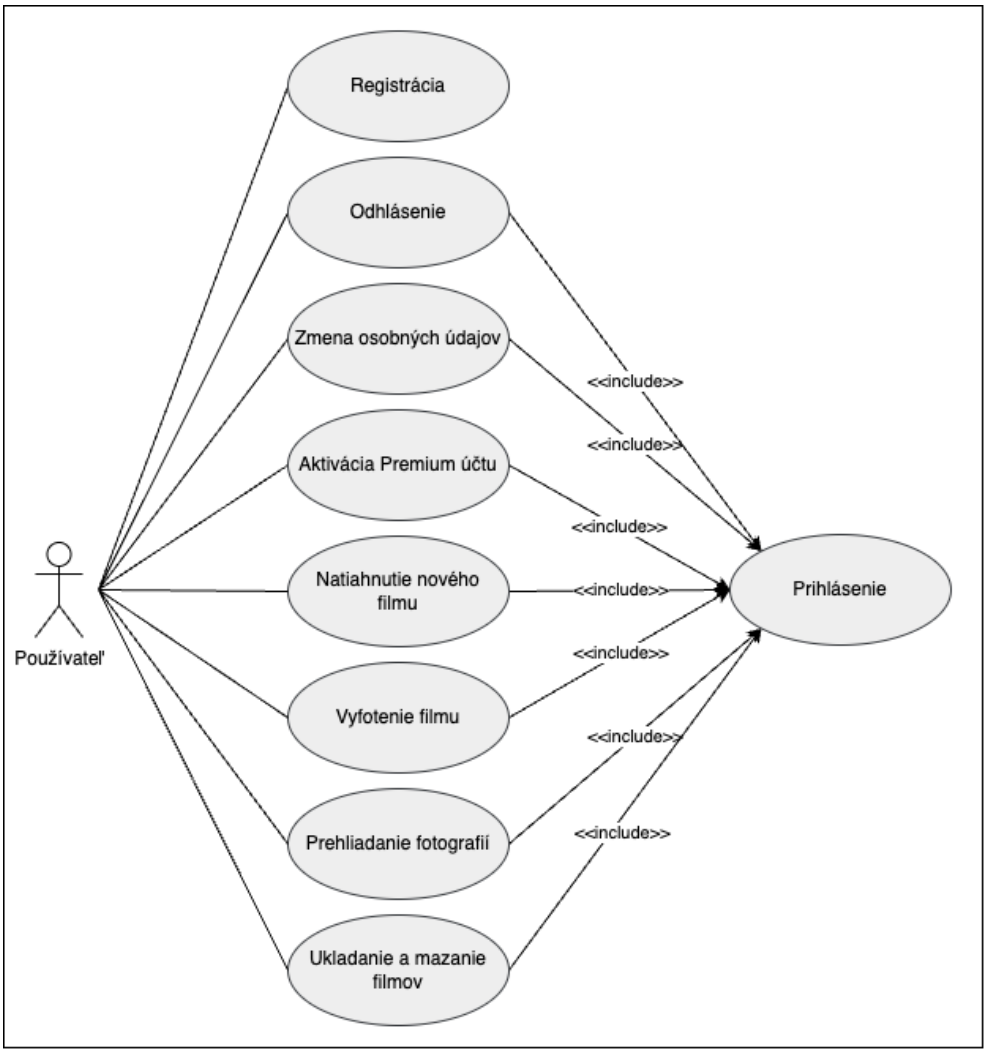
2.4.1 Zoznam aktérov

Zoznam aktérov identifikuje všetky prvky, ktoré interagujú s vyvíjaným systémom [46].

Implementovaná aplikácia má aktéra jednoho a je ním samotný koncový používateľ, ktorý používa aplikáciu na zachytávanie fotografií.

2.4.2 Diagram prípadov použitia

Diagram prípadov použitia vizuálne zachytáva interakcie medzi aktérmi a systémom a teda zobrazuje funkčné požiadavky systému v kontexte cieľov používateľa. Je znázornený na Obrázku 2.15.



Obr. 2.15: Diagram prípadov použitia, podľa [43].

Samotný diagram prípadov použitia ale pre správne popísanie funkčných požiadavkov nie je dostatočne detailný. Jednotlivé prípady použitia je preto potrebné slovne popísať a vysvetliť, čomu sa venuje nasledujúca podsekcia.

2.4.3 Zoznam prípadov použitia

UC1 – Registrácia

- **Popis:** Používateľ klikne na tlačidlo „Registrácia“ a zobrazí sa mu registračný formulár. Používateľ vyplní polia potrebné pre registráciu, zahrňujúce meno, priezvisko, e-mail, heslo a potvrdenie hesla. Ak užívateľ vyplní všetky údaje správne, s korektným formátom e-mailu a heslom splňujúcim bezpečnostný štandart a klikne na tlačidlo „Zaregistrovať sa“, zobrazia sa mu inštrukcie pre overenie účtu pomocou e-mailu. Po overení a aktivácií účtu je užívateľ zaregistrovaný a prihlásený do aplikácie.

UC2 – Prihlásenie

- **Popis:** Používateľ klikne na tlačidlo „Prihlásenie“ a zobrazí sa mu prihlasovací formulár. Vyplní polia potrebné pre prihlásenie, zahrňujúce e-mail a heslo. Po kliknutí na tlačilo „Prihlásenie“ je pri zadaní správnych údajov prihlásený do aplikácie, v opačnom prípade je vyzvaný ku korekcii údajov.

  Ak si používateľ nepamätá heslo, ktoré zadal pri registrácií, aplikácia mu povolí ho obnoviť. Po klinutí na tlačidlo obnovy vyplní formulár s e-mailom a klikne na obnovenie hesla. Do e-mailovej schránky mu príde odkaz, pomocou ktorého heslo zmení. Po zmene hesla sa môže opäť prihlásiť už popísaným postupom.

UC3 – Odhlásenie

- **Popis:** Možnosť odhlásenia má iba príhlasený používateľ. Zo záložky „Profil“ môže kliknúť na tlačidlo „Odhlásenie“, ktoré ho vyzve svoju voľbu potvrdiť. Po kladnej odpovedi na výzvu je používateľ odhlásený a presmerovaný na štartovaciu obrazovku aplikácie. V opačnom prípade používateľ zostane prihlásený.

UC4 – Zmena osobných údajov

- **Popis:** Prihlásený používateľ má možnosť v záložke „Profil“, pod kolónkou „Zmena údajov“ zmeniť svoje osobné informácie zahrňujúce meno a heslo. Po kliknutí je presmerovaný do formuláru, kde môže údaje upravovať. Úpravy musí potvrdiť tlačidlom „Uložiť“, inak nie sú zaznamenané.

UC5 – Aktivácia Premium účtu

- **Popis:** v záložke „Profil“, pri kliknutí na tlačidlo „Premium plán“ je prihlásený užívateľ presmerovaný na obrazovku, na ktorej je mu umožnené zakúpenie Premium účtu, ktorý mu odomkne všetky druhy filmov v ponuke aplikácie. Pre aktiváciu môže zadať údaje z karty alebo použiť Apple Pay.

UC6 – Natiahnutie nového filmu

- **Popis:** V záložke „Fotoaparát“ má používateľ natiahnutý film daného typu. Ak je kapacita filmu vyčerpaná, používateľ je pomocou notifikácie vyzvaný zadať údaje k novému filmu. Po kladnom potvrdení notifikácie je presmerovaný na obrazovku, kde zadá meno, počet snímkov a typ filmu. Po potvrdení výberu je navrátený späť do fotoaparátu, kde môže na zvolený film fotiť fotografie.

UC7 – Vyfotenie filmu

- **Popis:** Používateľ môže na natiahnutý film fotiť ním zadaný počet fotografií. Môže tak učiniť v záložke „Fotoaparát“, kliknutím na tlačidlo v strede na spodku obrazovky, simulujúce spúšť fotoaparátu. Po dofotení daného počtu fotografií je v záložke „Galéria“ možné film vyvolať kliknutím na tlačidlo „Vyvolať“. Po kliknutí sa zobrazia odfotené fotografie a je možné ich prezerať.

UC8 – Prehliadanie fotografií

- **Popis:** Prehliadanie fotografií je možné len pri filmoch, ktoré už sú vyvolané. Používateľ si môže v záložke „Galéria“ prezerať filmy buď v malom náhľade filmu, alebo si film rozkliknúť na samostatnej obrazovke dedikovanej danému filmu a prezerať ich na celej obrazovke. Filmy je taktiež možné podľa mena prehľadávať, kliknutím na vyhľadávanie v hornej časti záložky „Galéria“.

UC9 – Ukladanie a mazanie filmov

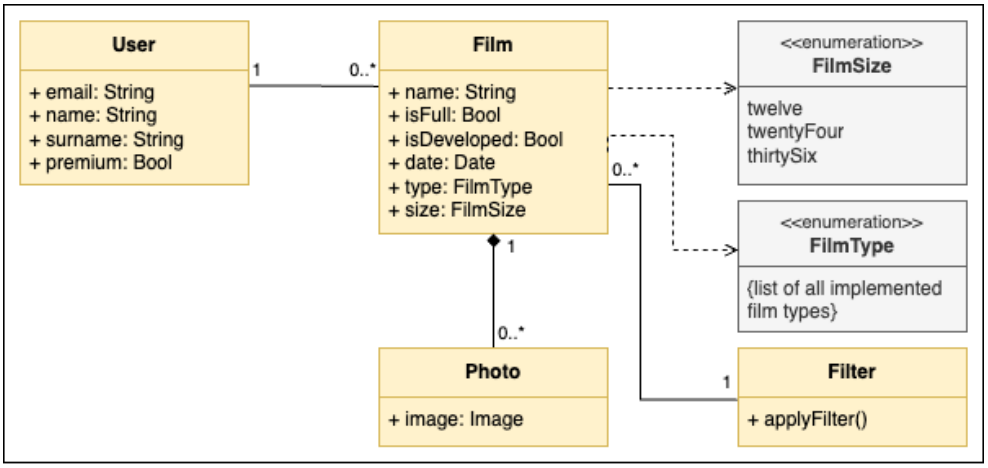
- **Popis:** Vyvolané filmy si môže používateľ uložiť do galérie svojho zariadenia, alebo ich z aplikácie úplne vymazať. Môže tak učiniť v záložke „Galéria“, kde zvolí a rozklikne daný film. V pravej hornej časti obrazovky klikne na tlačidlo indikujúce úpravu, ktoré mu dá na výber dve možnosti – „Zmazať“ a „Uložiť“. Pri možnosti „Uložiť“ sa uložia všetky fotografie do galérie zariadenia, pri možnosti „Zmazať“ sa používateľovi zobrazí notifikácia pre potvrdenie zmazania. Po kladnej odozve je film vymazaný, v opačnom prípade zostane aj naďalej v aplikácií.

Po rozkliknutí jednotlivých fotografií je možné ukladať len samostané fotografie, taktiež v pravom hornom rohu obrazovky danej fotky.

2.5 Doménový model

Doménový model pomáha identifikovať kľúčové entity a vzťahy medzi nimi v rámci špecifickej oblasti. Pomáha pochopiť základné koncepty systému, ktoré nie sú zrejmé požiadavkov a prípadov použitia, čím uľahčuje návrh systému. [43]

Konkrétny doménový model pre aplikáciu je možné vidieť na Obrázku 2.16.



Obr. 2.16: Doménový model, podľa [43].

2.5.1 Používateľ

Entita `User` predstavuje jedného používateľa aplikácie. O každom používateľovi sú zaznamenávané jeho osobné údaje, zahrňujúce e-mailovú adresu, meno, priezvisko a záznam o tom, či má zakúpený Premium plán k svojmu účtu, čomu slúži atribút `premium`.

Väzba medzi entitami `User` a `Film` značí, že používateľ môže mať vo svojej galérií 0 až N filmov, ale film môže patriť len jednému používateľovi.

2.5.2 Film

Entita `Film` reprezentuje analógový film, ktorý má svoje meno, dátum vyvolania, atribút isDeveloped na poznačenie, či bol daný film už vyvolaný používateľom a taktiež atribút `isFull`, ktorý značí, či už je naplnená jeho kapacita. Okrem toho má ešte dva špeciálne enumerované atribúty, kde jeden určuje typ filmu a druhý počet snímkov, ktoré sa na film zmestia.

Ku každému filmu navyše patria fotografie, ktoré sú naň zachytené. Každý film môže obsahovať 0 až N fotografií, pričom každá fotografia môže patriť len k jednému filmu. Keďže entita `Film` skladá z fotografií – je z nich zkomponovaná – je medzi nimi väzba typu kompozície.

2.5.3 Fotografia

Entita `Photo` predstavuje samotnú fotografiu zhotovenú v aplikácií. Jej atribút `image` preto obsahuje obrázok, ktorý je možné v aplikácií zobraziť. Každá fotografia môže byť priradená len k jednému filmu.

2.5.4 Typ filmu

Entitu `FilmType` je možné nazvať aj enumerovaná entita, alebo entita výčtového typu, keďže predstavuje konečnú a dobre definovanú množinu všetkých možných typov filmu. Pri pridaní implementácie nového typu filmu je potrebné ju o daný typ rozšíriť.

2.5.5 Veľkosť filmu

Entita `FilmSize` je taktiež enumerovaná entita, ktorá obsahuje množinu všetkých možných veľkostí filmu. Keďže aplikácia simuluje 35mm film, entita obsahuje tri veľkosti – 36, 24 a 12 snímkov na film.

2.5.6 Filter

`Filter` predstavuje rozhranie, ktoré obsahuje jednu funkciu `applyFilter`. Pridávaním nových implementácií tohto rozhrania je možné rozširovať aplikáciu o nové filtre na úpravu fotografií. Platí, že jeden filter môže byť použitý na akýkoľvek počet filmov, ale filmu môže priliehať len jeden filter.

Kapitola 3

Návrh

Požiadavky na aplikáciu vymedzené v analýze jasne definujú, čo má vyvíjaná aplikácia robiť. Ďalším krokom úspešného vývoja je špecifikácia spôsobov, ako toho docieliť [47]. Tejto časti vývoja sa venuje návrh. Nasledujúce sekcie sa zaoberujú popisom zvolenej platformy a programovacieho jazyka, návrhom architektúry aplikácie ako aj návrhom používateľského rozhrania.

3.1 iOS vývoj

iOS (iPhone Operating System) je mobilný operačný systém spoločnosti Apple Inc., používaný na zariadeniach iPhone a iPod Touch. iOS platforma kladie dôraz na konzistentnosť a jednoduchosť v dizajne, čo používateľom zaručuje jednoduchú navigáciu v zariadení a poskytuje intuitívny a esteticky príjemný používateľský zážitok.

Prvá verzia operačného systému iOS bola predstavená v roku 2007, spolu s prvým mobilným tefónom iPhone. Od roku 2007 bolo vydaných niekoľko verzií systému, pričom najnovšia je iOS 16, predstavená v septembri roku 2022. [48]

V nasledujúcich podsekciách je popísané prostredie, v ktorom je možné aplikácie pre operačný systém iOS implementovať, spôsob ich distribúcie, ako aj operačný jazyk a možnosti tvorby používateľského rozhrania.

3.1.1 Vývojové prostredie

Jediný oficiálny nástroj pre vytváranie a publikovanie aplikácií pre zariadenia od spoločnosť Apple Inc. je *Xcode*. Xcode je integrované vývojové prostredie (IDE) určené pre vývoj aplikácií pre všetky platformy spoločnosti Apple a je bezplatné pre všetkých jej používateľov. Okrem toho Xcode podporuje mnohé programovacie jazyky vrátane Swift, Objective-C, Objective-C++, C, C++, Java, Python a ďalšie.

Xcode je dostupný iba na zariadeniach Mac, s operačným systémom macOS. Pre vývoj aplikácií pre iOS platformu musí teda vývojár vlastniť zariadenie Mac [49].

Aj napriek tomu, že existujú alternatívy ku Xcode, väčšina z nich sa spolieha na kompiláciu pomocou Xcode. Publikovanie aplikácie je na tomto IDE taktiež závislé, síce nemusí byť spustené, ale musí byť nainštalované [50]. Preto je pre implementáciu aplikácie vybrané IDE Xcode.

3.1.2 Distribúcia aplikácie

Distrubúcia aplikácií je možná len so zakúpeným *Apple Developer Program*. Tento program je založený na ročnom členskom preplatnom vo výške $99 a poskytuje vývojárom prístup k rôznym zdrojom, nástrojom, dokumentácii a samotnej publikácií aplikácie v obchode App Store. Apple Developer Program je možné zakúpiť priamo na webovej stránke spoločnosti Apple.

Hotovú aplikáciu je možné otestovať medzi koncovými používateľmi pomocou služby *TestFlight*, poskytovanej spoločnosťou Apple. TestFlight umožňuje distribúciu predbežnej verzie aplikácie obmedzenému počtu testerov na beta testovanie pre získanie spätnej väzby ešte pred vydaním aplikácie. Služba ponúka funkcie, ako sú aktualizácie v aplikácii, analytika aplikácií a iné. Pre použitie služby TestFlight je taktiež potrebné mať zakúpený Apple Developer Program.

Na zverejnenie aplikácie do App Store je možné použiť Xcode, alebo webovú stránku App Store Connect. Je potrebné poskytnúť informácie o aplikácii vrátane jej názvu, popisu, snímok obrazoviek a prípadných cenách. Pred odoslaním žiadosti o zverejnenie je potrebné sa uistiť, že aplikácia spĺňa všetky pokyny spoločnosti Apple, aby sa predišlo zamietnutiu zverejnenia aplikácie počas procesu kontroly.

Proces kontroly môže trvať od niekoľkých dní do niekoľkých týždňov v závislosti od zložitosti aplikácie. Spoločnosť Apple si vyhradzuje právo odmietnuť akúkoľvek aplikáciu, ktorá nespĺňa ich pokyny, takže je nevyhnutné pred odoslaním na kontrolu zabezpečiť, aby aplikácia spĺňala všetky požiadavky. Po schválení je aplikácia zverejnené v App Store, kde si ju môžu používatelia nainštalovať a začať používať. [51]

Na účely tejto bakalárskej práce nebol zakúpený Apple Developer Program, takže popísané poznatky slúžia k budúcemu rozvoju aplikácie.

3.1.3 Programovací jazyk

Pre natívny vývoj aplikácií pre platformu iOS existujú dve možnosti voľby programovacieho jazyka – *Swift* a *Objective-C*.

Oba sú podporované a odporúčané spoločnosťou Apple na vytváranie natívnych aplikácií pre iOS, no každý z nich ponúka odlišné vlastnosti a funkci-

onalitu. Ich charakteristika a výhody a nevýhody sú popísané v nasledovných dvoch podsekciách.

3.1.3.1 Objective-C

Objektovo orientovaný programovací jazyk Objective-C bol vytvorený okolo roku 1980. Jednou z hlavných výhod je kompatibilita s akoukoľvek verziou systému iOS, takže sú vďaka nemu podporované aj staršie projekty. Vďaka dlhoročnej prítomnosti na trhu je veľmi stabilný, má rozsiahlu dokumentáciu a taktiež disponuje rôznymi nástrojmi uľahčujúcimi vývoj.

Napriek tomu už sa v nových projektoch takmer nepoužíva, pretože má oproti moderným programovacím jazykom zložitejšiu syntax, takže je vývoj v tomto jazyku často časovo náročnejší. [52]

3.1.3.2 Swift

Programovací jazyk Swift bol predstavený v roku 2014, a vďaka modernejšej a zjednodušenej syntaxi je jednoduchšie mu porozumieť už na prvý pohľad.

Disponuje automatickou správou pamäte s automatickým počítaním referencií a je menej náchylný na chyby. Vďaka stručnej sytaxi a optimalizovanému kompilátoru je rýchlejší nielen pre programovanie, ale aj výkonovo.

Keďže je Swift relatívne nový jazyk stále sa vyvíja a mení, takže je nekompatibilný so staršími verziami. Preto je potrebné staré projekty vyvýjať aj naďalej v Objective-C. [52]

Zvolený je jazyk pre implementáciu aplikácie je na základe predošlých poznatkov jazyk Swift a jeho najnovšia verzia Swift 5.8.

3.1.4 Tvorba používateľského rozhrania

Používateľské rozhranie tvorí vizuálnu časť aplikácie, s ktorou môžu používatelia interagovať na svojich zariadeniach. Zahŕňa všetky prvky, ktoré používateľovi umožňujú interakciu s aplikáciou, ako sú tlačidlá, štítky, textové polia, obrázky a ďalšie.

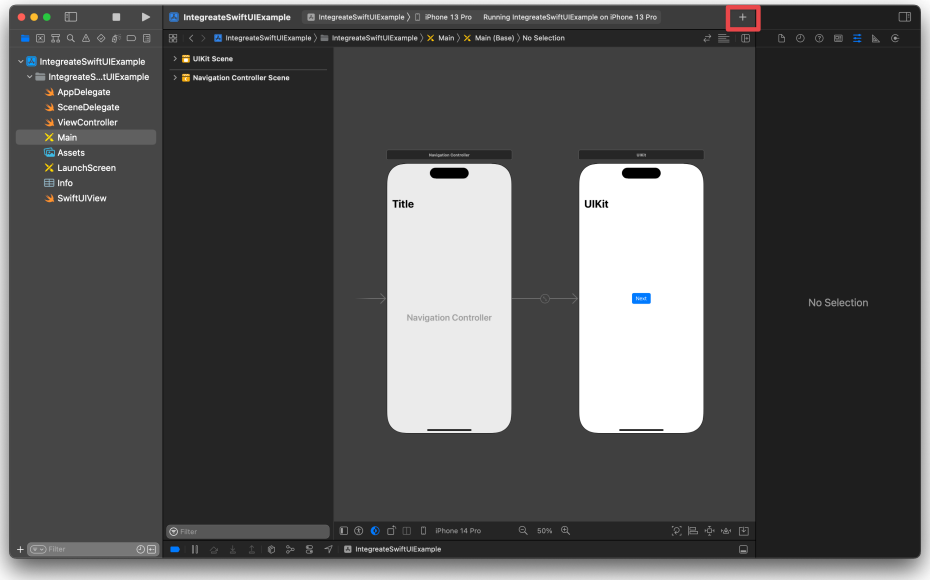
Samotný programovací jazyk Swift neposkytuje funkcionalitu pre tvorbu používateľského rozhrania. Spoločnosť Apple preto ponúka na výber dve možnosti – *UIKit* framework a *SwiftUI* framework. Framework predstavuje špecifickú sadu dopredu definovaných funkcií dostupných pre programátora, ktoré mu uľahčujú prácu a šetria čas.

Nasledujúce dve podsekcie popisujú oba frameworky a určujú, ktorý z nich je pre aplikáciu zvolený.

3.1.4.1 UIKit

UIKit framework bol predstavený v roku 2008, je založený na jazyku Objective-C a ponúka vysokú spätnú kompatibilitu so staršími verziami iOS a stabilnú a overenú funkcionalitu.

Užívateľské rozhranie (UI) je pomocou frameworku UIKit možné vytvoriť pomocou kódu, alebo pomocou grafického rozhrania zvaného Interface Builder. Interface Builder umožňuje skladať užívateľské rozhrania pomocou Storyboards, zobrazených na Obrázku 3.1, pričom každá Storyboard predstavuje jednu obrazovku aplikácie. Používateľské rozhranie sa dá preto veľmi jednoducho nadefinovať niekoľkými kliknutiami, ktoré vygenerujú kód.



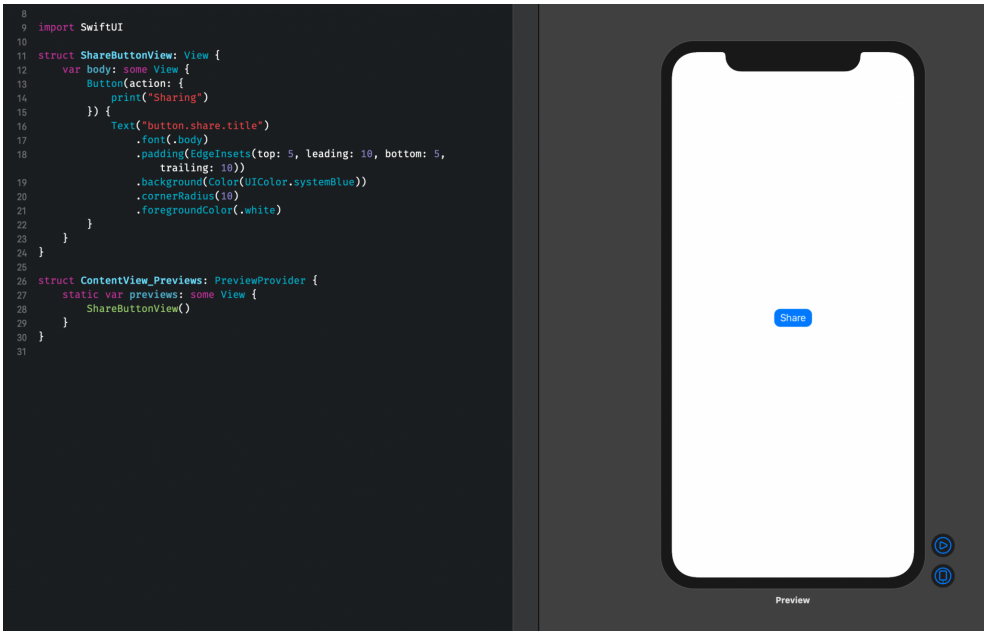
Obr. 3.1: Ukážka Storyboard [53]

Pri vynechaní rozhrania Interface Builder je potrebné UI definovať prostredníctvom kódu, ktorého tvorba zvykne byť náročnejšia. Keďže UIKit používa viac imperatívnu syntax programovania, je potrebné presne špecifikovať čo má program robiť, kedy sa má aktualizovať, ako sa majú jednotlivé elementy správať, ako majú vyzerať a kde majú byť umiestnené. Aj napriek tomu, že tento prístup je zdĺhavejší, ponúka programátorom väčšiu moc nad výzorom výslednej aplikácie, preto je výhodný pre aplikácie so špecifickým UI. [54]

3.1.4.2 SwiftUI

Framework SwiftUI bol predstavený v roku 2019, a podporuje len zariadenia s operačným systémom iOS 13 a vyššie. Keďže je na trhu len krátko, veľmi rýchlo sa mení, vyvíja a dopĺňa svoju funkcionalitu.

Narozdiel od UIKit neumožňuje použitie Interface Builder, tento fakt ale kompenzuje možnosťou použitia Live Preview – živého náhľadu, zobrazeného na Obrázku 3.2. Tento náhľad poskytuje vývojárom grafické vykreslenie písaného kódu už počas jeho tvorenia. Živé vizuálne ukážky dovoľujú ihneď skontrolovať naimplementovanú časť kódu, keďže sa v nich nezrovnalosti zobrazia okamžite. [54]



Obr. 3.2: Ukážka Live Preview [55]

SwiftUI využíva deklaratívnu syntax, ktorá umožňuje vývojárom deklarovať, ako by malo používateľské rozhranie vyzerať a správať sa a framework sa postará o samotnú implementáciu. Tento fakt tvorí jeden z hlavných rozdielov medzi SwiftUI a UIKit, pretože pri UIKit je všetky tieto informácie potrebné naprogramovať pomocou sady procedurálnych krokov. Výsledkom je, že vývoj používateľského rozhrania pri SwiftUI je jednoduchší a rýchlejší, ale možnosti prispôsobenia sú obmedzenejšie [56].

Aj napriek tomu, že oba frameworky sú oficiálne podporované, spoločnosť Apple sa snaží o presadenie SwiftUI [54].

Na tvorbu používateľského rozhrania je na základe popísaných informácie využitý framework SwiftUI, keďže môže vývoj značne zrýchliť, zjednodušiť a zaručiť aplikáciu s prirodzeným iOS dizajnom. Aplikácia zároveň nepotrebuje podporovať staršie verzie iOS.

3.2 Firebase backend

Zaznamenávanie údajov o identite jednotlivých používateľov umožňuje bezproblémové prihlásenie na viacerých zariadeniach. Keďže jedným z požiadavkov na aplikáciu je možnosť zakúpenia Premium plánu, použitie backendovej služby umožní vývojárom spravovať nielen informácie o identite užívateľov, ale aj o zakúpení tohto balíčka.

Na tento účel je vhodné použitie *Firebase* od spoločnosti Google, ktorá predstavuje platformu na vývoj webových a mobilných aplikácií. Poskytuje služby ako autentizácia, cloud úložisko, web hosting, či strojové učenie. Pre splnenie požiadavkov postačuje použitie dvoch služieb – *Authentication* a *Firestore* – popísaných v nasledujúcich podsekciách. Výhodou použitia Firebase pre túto prácu je taktiež fakt, že ponúka bezplatný plán pre jej použitie.

3.2.1 Authentication

Vďaka službe Authentication je možné registrovať alebo prihlásiť používateľa do aplikácie pomocou osobných údajov, ako je napríklad e-mail a heslo. Pre zjednodušenie prihlásenia Authentication však ponúka aj možnosť prihlásenia pomocou Google, Facebook, či Twitter. Keď používateľ tieto údaje zadá do aplikácie, stačí ich len pomocou Firebase Authentication Software Development Kit (SDK) predať Firebase a ich backendové služby sa postarajú o všetko ostatné. Používateľa je samozejme možné aj odhlásiť, alebo využiť rôznu inú funkcionalitu ponúkanú touto službou.

Po úspešnom prihlásení je možné pristupovať ku základným informáciám používateľa a taktiež kontrolovať jeho prístupy do ostatných služieb Firebase. [57]

3.2.2 Cloud Firestore

Cloud Firestore predstavuje fexibilnú a škálovateľnú NoSQL databázu, vhodnú na vývoj aplikácie na mobilné zariadenia, web, či server. V reálnom čase udržiava údaje synchronizované medzi klientskymi aplikáciami ale ponúka aj offline podporu.

Keďže je dátový model Cloud Firestore založený na NoSQL databáze, údaje sú uložené v dokumentoch, ktoré obsahujú polia namapované na hodnoty, ktoré sú uložené v kolekciách. V dokumentoch je možné vytvárať podkolekcie a hierarchické dátové štruktúry, ktoré sa menia s rastom databázy. Dotazovanie v Cloud Firestore efektívne a flexibilné a tiež umožňuje získavať údaje na úrovni dokumentu bez toho, aby musela byť načítaná celá kolekcia s dokumentami. [58]

Vďaka tejto funkcionalite je teda možné pomocou Firestore zaznamenávať detailnejšie osobné údaje o používateľoch ako je meno, priezvisko či dátum

narodenia, ako aj informáciu o zakúpení Premium plánu, čo by iba pomocou Authentication nebolo možné.

Spojením Firestore a ďalšej služby ponúkanej Firebase – Firebase Storage, by bolo možné okrem osobných údajov o používateľovi ukladať aj jeho zhotovené fotografie a filmy. Používateľ by mal tak fotografie a filmy zosynchornizované na všetkých zariadeniach. Využitie tejto služby ako úložisko fotografií by však spotrebovalo veľké množstvo úložného priestoru, čím by boli prekročené limity bezplatného plánu Firebase, použitého pre túto aplikáciu. Bolo by teda potrebné zakúpenie plateného plánu, ktorý je ale finančne náročnejší a aplikácia sama o sebe by tým pádom musela byť platená. Keďže je ale jedným z nefunkčných požiadavkov minimalizovanie nákladov na prevádzku aplikácie, aby bola k dispozícií každému, autorka práce sa rozhodla Firebase Storage pre ukladanie fotografií nevyužiť.

3.3 Architektúra

Po správnom zadefinovaní požiadavkov na aplikáciu a výbere vhodných technológií je možný výber architektúry aplikácie. Architektúra predstavuje návrh štruktúry systému, určuje ako spolu jednotlivé časti interagujú a taktiež špecifikuje pravidlá organizácie kódu a celkovej komplexity projektu.

Výber vhodnej architektúry je jedným z hlavných nosníkov úspešného vývoja aplikácie, keďže vie zaručiť sprehľadnenie projektu, jeho rozšíriteľnosť a celkovú udržateľnosť. Neexistuje jedno univerzálne riešenie vhodné pre všetky aplikácie, preto je dôležité zvážiť rozsah implementovanej aplikácie, požadovanú funkcionalitu a vymedzený čas.

Neoddeliteľnou súčasťou architektúry sú návrhové vzory, ktoré ponúkajú riešenia často sa vyskutujúcich problémov. Nie sú zavislé na jazyku alebo technológií, ale predstavujú všeobecný prístup k riešeniu daného problému.

Nasledujúce podsekcie sa venujú popísaniu možností architektúry, zvolenej arichitektúre, dôvodom jej výberu a výhodám a nevýhodám jej voľby.

3.3.1 Možnosti štrukturizácie projektu

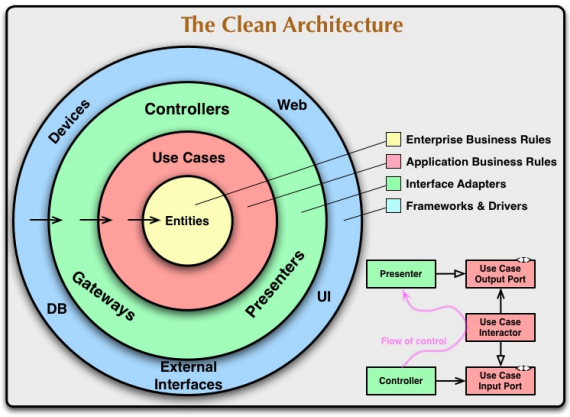
3.3.1.1 Clean Architecture

Často používanou architektúrou pre iOS vývoj je *Clean Architecture*, ktorá sa zameriava na oddelenie povinností jednotlivých častí, modularitu a udržateľnosť. Činí tak organizáciou kódu do štyroch osobitných vrstiev, zobrazených na Obrázku 3.3:

- **Entities** tvoria najvnútornejšiu vrstvou, ktorá obsahuje základné dátové štruktúry.
- **Use Cases**, v preklade prípady použitia popisujú správanie aplikácie.

- **Interface Adapters** prekladá údaje medzi vonkajšou vrstvou a základnými vrstvami.
- **Frameworks and Drivers** pozostávajú z rámcov, nástrojov a externých závislostí.

Vďaka tento štruktúre sú na sebe komponenty nezávislé a ľahko zameniteľné, aplikácia je jednoduchšie testovateľná, prispôsobivejšia a ľahšie sa udržuje či rozširuje [59].



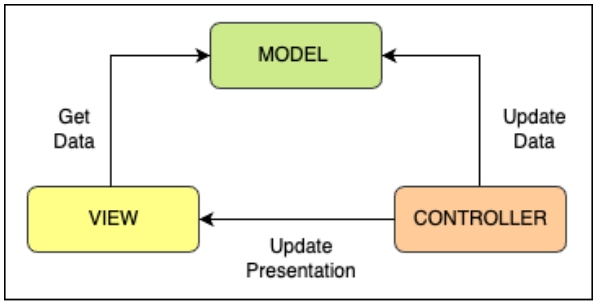
Obr. 3.3: Ukážka vrstiev Clean Architecture, podľa [60]

Vzhľadom na rozsah aplikácie je ale Clean Architecture nevhodná, pretože pri menších projektoch toto rozdelenie pridáva zbytočnú réžiu a tak môže viesť k zvýšenej zložitosti, predĺženiu času vývoja a nadbytočnému úsiliu. Pre menšie projekty sa preto pre štrukturizáciu iOS aplikácií taktiež používajú návrhové vzory *MVC* a *MVVM*, popísané v nasledovných častiach.

3.3.1.2 MVC

Návrhový vzor MVC, celým názvom Model-View-Controller, rozdeľuje aplikáciu do troch rovnomenných komponent, ako je zobrazené na Obrázku 3.4, komponenty sú nasledovné:

- **Model** reprezentuje dáta a ich funkcionalitu.
- **View** predstavuje používateľské rozhranie a slúži na vizualizáciu dát a príjem vstupov, avšak nenachádza sa tu žiadna logika aplikácie.
- **Controller** funguje ako sprostredkovateľ medzi Modelom a View. Spracúva vstupy používateľov z View, zodpovedajúcim spôsobom na ne reaguje a aktualizuje Model.



Obr. 3.4: Ukážka návrhového vzoru MVC, podľa [61]

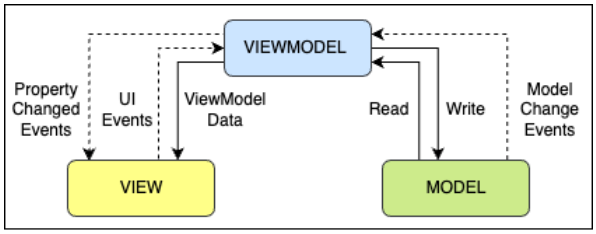
Návrhový vzor MVC by bol vhodnou voľbou pri použití UIKit na tvorbu používateľského rozhrania, keďže podstata UIKit stojí na MVC, ako je uvedené v oficiálnej dokumentácií od Apple [62]. Beh aplikácie využívajúcej UIKit je postavený na ViewControllers, ktoré riadia životný cyklus jednotlivých Views čím tvoria obrazovku.

Keďže bol ale pre implementáciu používateľského rozhrania zvolený framework SwiftUI, návrhový vzor MVC nie je úplne výhodné použiť.

3.3.1.3 MVVM

MVVM, celým názvom Model-View-ViewModel, sa skladá z nasledujúcich troch komponent, zobrazených aj na Obrázku 3.5:

- **Model** reprezentuje dáta a ich funkcionalitu.
- **View** predstavuje používateľské rozhranie a slúži na vizualizáciu dát a príjem vstupov, avšak nenachádza sa tu žiadna logika aplikácie.
- **ViewModel** sprostredkúva komunikáciu medzi View a Modelom, pomocou *bindings* alebo iných akcií. Rovnako je v ňom uložený aktuálny stav dát, ktoré sú zobrazované pomocou *View*.



Obr. 3.5: Ukážka návrhového vzoru MVVM, podľa [61]

Špeciálnu rolu v MVVM majú už spomínané bindings, ktorým sa dá zjednodušene rozumieť ako akýmsi spojeniam medzi dátami a príslušným View, ktoré dáta zobrazuje. [63]

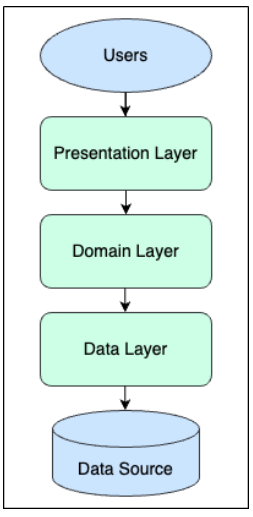
Bindings je možné veľmi prirodzene využiť v SwiftUI, keďže samotný jazyk ponúka takzvané *property wrappers*, ktoré dokážu rozšíriť premenné a prepojiť ich s jednotlivými Views. Vďaka nim sa vždy po zmene hodnoty premennej obalenej do property wrapper táto zmenená hodnota premietne aj do príslušného View.

Aj napriek jeho výhodám a natívnemu fungovaniu so SwiftUI, prosté použitie MVVM pre aplikáciu nie je postačujúce, pretože nerieši jej modularitu, ktorá je jedným z nefunkčných požiadavkov. Keby bolo neskôr rozhodnuté napríklad o zmene Firebase na iný typ databázy, prechod by tak bol obtiažny.

3.3.2 Trojvrstvová architektúra

Trojvrstvová architektúra, ako naznačuje názov a Obrázok 3.6, rozdeľuje aplikáciu do troch vrstiev. Samotná vrstva predstavuje abstraktnú časť aplikácie, ktorá pomaháha s jej rozdelením a modularitou. Vrstvy architektúry sú podľa [64] nasledovné:

- **Prezentačná vrstva** zobrazuje dáta prostredníctom grafického používateľského rozhrania (GUI), takže s ňou priamo interaguje samotný používateľ.
- **Doménová vrstva**, ktorej sa taktiež hovorí aj stredná vrstva zahrňuje jadro funkcionality aplikácie.
- **Dátová vrstva** je nezávislá na aplikácií a sprostredkováva dáta z databázových serverov.



Obr. 3.6: Ukážka trojvrstvovej architektúry, podľa [64]

Vďaka zvýšeniu modularity a oddeleniu dátovej vrsty od zbytku aplikácie je Trojvrstvová architektúra vhodným adeptom na výber pre aplikáciu. Upresnenie jej použitia je popísané v nasledujúcej podsekcii, venujúcej sa zvolenej architektúre.

3.3.3 Zvolená architektúra

Ako už bolo spomenuté, pri návrhu architektúry aplikácie je dôležité brať do úvahy potreby konkrétnej aplikácie a preto je pre aplikáciu zvolená Trojvstvová architektúra v kombinácií s návrhových vzorom MVVM.

Prezentačná vrstva je tvorená Views a ViewModels, pričom View definuje používateľské rozhranie a ViewModel udržuje zobrazované dáta. ViewModels taktiež používajú aj ostané dve vrstvy.

Doménová vrstva obsahuje entity (zvyčajne Models) danej domény a ich pridruženú funkcionalitu a business logiku.

Pre kominukáciu a získavanie informácií z rôznych dátových zdrojov potom ViewModels používajú funkcionalitu dátovej vstvy. Obvykle sa na dátovej vrstve jedná o volanie API, použitie databáze či iOS systému na správu súborov a podobne.

Zvolená architektúra popísaná v tejto podsekcii teda zaručí rozšíriteľnost, prehľadnosť, udržateľnost a taktiež modifikovateľnosť aplikácie.

3.4 Návrh používateľského rozhrania

Posledná časť celkového návrhu aplikácie sa venuje návrhu používateľského rozhrania, ktorý sa riadi *Human Interface Guidelines* (HIG) od firmy Apple. HIG predstavuje súbor pravidiel a doporučení pre vyvíjanie produktov na akúkoľvek Apple platformu a pri ich dodržaní zaručuje prehľadnosť, intuitívne ovládanie a maximálne uspokojivý zážitok pri používaní aplikície. [65]

Nasledujúce podsekcie sa venujú popísaniu jednotlivých častí a obrazoviek aplikácie a vysvetleniu ich funkcionality. Sú doplnené o *wireframes*, ktoré predstavujú zjednodušené vizuálne zobrazenia aplikácie a poslúžia ako záchytný bod a predloha pri implementácií.

3.4.1 Autentifikácia

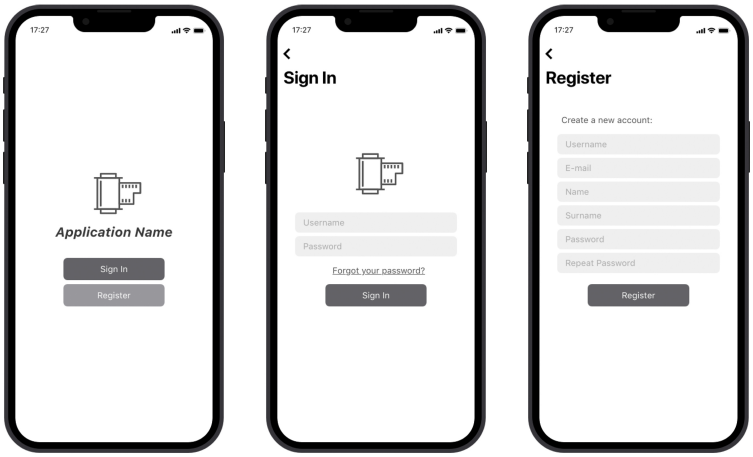
Po spustení aplikácie sa neprihlásený používateľ ocitne na štartovacej obrazovke pre autentifikáciu, zobrazenej na Obrázku 3.7a. Z tejto obrazovky si môže zvoliť, či sa chce prihlásiť, alebo zaregistrovať, ak ešte nemá vytvorený používateľský účet.

Pri voľbe tlačidla pre prihlásenie je presmerovaný na obrazovku prihlásenia, zobrazenej na Obrázku 3.7b, kde je vyzvaný zadať svoje prihlasovacie údaje zahŕňajúce e-mailovú adresu a heslo. Ak si heslo nepamätá, má možnosť si ho obnoviť.

Pri voľbe tlačidla pre registráciu je používateľ presmerovaný na obrazovku registrácie, zobrazenej na Obrázku 3.7c, kde je mu po vyplnení osobných údajov umožnené sa prihlásiť.

Pre spätnú väzbu je aj obrazovka prihlásenia aj obrazovka registrácie responzívna, takže ak zadá používateľ e-mail v zlom formáte alebo zadá nesprávne heslo, aplikácia používateľa upozorní a vyzve k oprave.

Po úspešnej registrácií alebo prihlásení je používateľ vpustený do aplikácie, ktorá je rozdelená na tri časti pomocou záložiek v spodnom paneli obrazovky.



(a) Štartovacia obrazovka (b) Prihlásenie (c) Registrácia

Obr. 3.7: Ukážky wireframes Autentifikácie, vytvorené pomocou [66]

3.4.2 Galéria

Prvú záložku tvorí záložka galérie, v ktorej je používateľovi umožnené prezerať a prehľadávať odfotografované filmy v menších náhľadoch, ako je zobrazené na Obrázku 3.8a. Pre zobrazenie detailu celého filmu môže používateľ kliknúť na jednotlivé náhľady, čím je presmerovaný na obrazovku detailu filmu, ako je zobrazené na Obrázku 3.8b. Na tejto obrazovke je používateľovi umožnené taktiež film zmazať alebo uložiť.

3.4.3 Používateľský profil

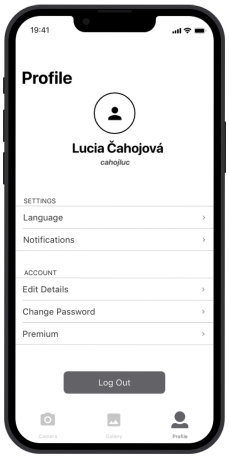
Druhá záložka predstavuje používateľský profil, ako je znázornené na Obrázku 3.9. na tejto obrazovke si môže používateľ zmeniť jazyk aplikácie, upraviť svoje osobné údaje, aktivovať Premium plán, či sa odhlásiť zo svojho účtu.

Pre dodržanie nefunkčného požiadavku o minimalizovaní nákladov na aplikáciu definovanom v analýze, sú jednotlivé filmy ukladané len do lokálneho

(a) Galéria (b) Detail filmu

Obr. 3.8: Ukážky wireframes Galérie, vytvorené pomocou [66]

úložiska telefónu. Preto je pri odhlásení používateľ notifkovaný o ich zmazaní, ak sa rozhodne dokončiť proces odhlásenia.



Obr. 3.9: Ukážka obrazovky Profilu, vytvorené pomocou [66]

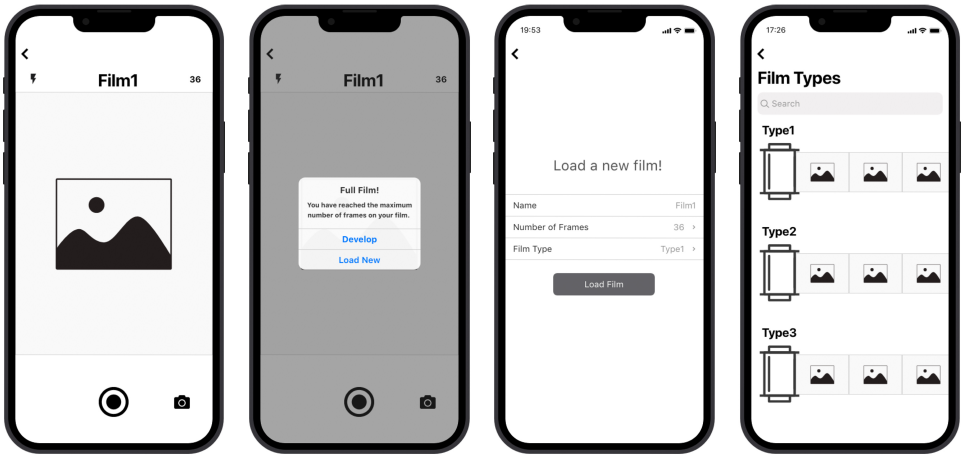
3.4.4 Fotoaparát

Posledná záložke predstavuje hlavnú časť aplikácie a je ňou samotný fotoaparát, zobrazený na Obrázku 3.10a. Je navrhnutý tak, aby sa čo najviac približoval dizajnu systémové fotoaparátu zariadení iPhone, aby zachoval intuitívne ovládanie.

V hornej časti obrazovky sa nachádza lišta, na ktorej je možné ovládať blesk, zobrazuje ale tiež meno momentálne natiahnutého filmu a počet zostá-

vajúcich snímkov. V dolnej časti obrazovky sa nachádza spúšť a tlačidlo na zmenu orientácie fotoapátu, ktore umožňuje prepínať medzi prednou a zadnou kamerou.

Pri vyčerpaní kapacity natiehnutého filmu je používateľ vyzvaný natiahnuť nový film prostredníctvom notifikácie zobrazenej na Obrázku 3.10b. Pri kladnom potvrdení je presmerovaný na obrazovku, kde môže zadať meno filmu, počet snímkov a vybrať si typ filmu, ako je zobrazené na Obrázku 3.10c. Aplikácia pri zadávaní názvu filmu požívateľa upozorní, ak zadá už použitý názov filmu.



(a) Fotoaparát (b) Notifikácia o plnom filme (c) Natiahnutie nového filmu (d) Typy filmu

Obr. 3.10: Ukážky wireframes Fotoaparátu, vytvorené pomocou [66]

Kapitola 4

Implementácia

Vďaka úspešnému prevedeniu analýzy, zadefinovaniu požiadavkov na aplikáciu a zostaveniu návrhu je možné na základe nadobudnutých poznatkov a navrhnutého riešenia prejsť do implementačnej fázy práce.

Táto kapitola sa preto venuje popísaniu procesu implementácie aplikácie s názornými ukážkami použitého kódu a taktiež opísaniu obštrukcií, na ktoré bolo počas implementácie narazené.

4.1 Continuous Integration

Zavedenie použitia *Continuous Integration* (CI), bolo jedným z kľúčových krokov na začiatku implementácie. CI je softwarový postup, ktorý zahŕňa spravovanie a integráciu kódu v zdieľanom úložisku. Taktiež umožňuje aplikáciu v prostredí zdielaného úložiska testovať a tak zaručiť kompatibilitu novopridaného kódu. CI pomáha včas zachytiť chyby v kóde a tak poskytnúť rýchlu spätnú väzbu vývojárom. Vďaka automatizovaniu určitých procesov vie taktiež zvýšit produktivitu. [67]

Pre zavedenie CI do projektu bolo najskôr zvolené úložisko GitLab, avšak pri jeho nastavovaní autorka narazila na niekoľko prekážok a nakoniec bolo zvolené úložisko GitHub. Proces zavedenia je popísaný v nasledovných podsekciách aj spolu s popisom identifikovaných problémov a ich riešení.

4.1.1 GitLab

GitLab je open-source nástroj na správu úložisiek, ponúkajúci rôznu funkcionalitu zahŕňajúcu kontroly kódu, správu *issues* ale aj Continuous Integration a Continuous Deployment, ktoré ma priamo zabudované. Môže byť hosťovaný na cloude ale aj na vlastnom serveri.

Pre CI používa GitLab takzvané *pipelines*, ktoré predstavujú množinu úloh, spustených vo vopred určenom poradí. Samotná pipeline je definovaná v `.gitlab-ci.yml` súbore, ktorého ukážka je zobrazená vo Výpise 4.1. Pipline

sa spustí len ak je prevedný príkaz `push` do vývojovej vetvy `develop`, čo je definované v časti `only` v každej fáze pipeline v súbore `.gitlab-ci.yml`.

Jednotlivé úlohy predstavujú fázy pipeline a sú zadefinované v časti `stages` v súbore `.gitlab-ci.yml`. Vo Výpise 4.1 sú definované fázy `build` a `test`, pretože ako bolo už spomenuté, aplikácia nebude nasadená do App Store. Fázy sú spustené postupne, podľa ich bližších špecifikácií zadefinovaných nižšie v kóde.

```
image: swift:5.7

stages:
  - build
  - test

build:
  stage: build
  tags:
    - macOS
  script:
    ...
  working-directory: ./src
  only:
    - develop

test:
  stage: test
  tags:
    - macOS
  script:
    ...
  only:
    - develop
```

Listing 4.1: Ukážka súboru .gitlab-ci.yml

Jednotlivé úlohy spracováva *runner*, ktorému sa hovorí aj agent. Existujú dva druhy runnerov – zdielané alebo vlastné. Na fakultnom nástroji GitLab, ktorý bol použitý pre projekt, je k dispozícií len jeden runner s operačným systémom Linux [68]. Avšak ten je, ako už bolo spomenuté v Sekcii 3.1.1, pre vývoj aplikácie pre iOS platformu nevhodný, nakoľko je možné aplikáciu zostaviť len v prostredí operačného systému macOS.

Ďalšou možnosťou bolo vytvorenie a použitie vlastného runneru za použitia *Docker container*, avšak vždy pri pridávaní kódu na GitLab a spustení pipline by sa musel vývojár uistiť, že je daný Docker container spustený. GitLab ponúka macOS runner v Beta verzii, avšak iba používateľom s Premium účtom [69]. Keďže popísané kroky boli vyhodnotené ako réžia navyše, bolo

autorkou zvolené použitie *GitHub* a všetky vetvy a commity boli zmigrované.

4.1.2 GitHub

GitHub je webová platforma pre správu verzií a spoluprácu na softwarových projektoch. Čo sa týka CI/CD funkcionality, GitHub disponuje platformou zvanou *GitHub Actions*, pomocou ktrej je možné aplikáciu zostaviť, otestovať a aj nasadiť.

V neplatenej verzii majú súkromné repozitáre pridelené 2000 minút využívania GitHub Actions zadarmo na mesiac. Pri použití macOS runner sa využité minúty počítajú desaťnásobne, takže pri použií 1 minúty GitHub Actions sa z 2000 voľných minút odráta 10.

Pipeline projektu na dobehnutie potrebovala približne 30 minút, takže v praxi je možné ju za mesiac spustiť približne šesťkrát. Pre potreby práce je to postačujúce, pretože pri vyvíjaní jednotlivých častí aplikácie sú používané pre každú časť samostatné vývojové vetvy. Do vetvy `develop` sú teda pridávané len väčšie hotové celky častí aplikácie. Vďaka rozsahu aplikácie je teda pridelených 2000 minút na mesiac postačujúci. V opačnom prípade by bolo možné repozitár označiť ako verejný, pri ktorom toto obmedzenie neplatí.

GitHub taktiež využíva runners, avšak narozdiel od GitLab ponúka aj zdieľané macOS runners. Tieto runners sú virtuálne prístroje, na ktorých je spustená pipeline zadefinovaná v `/.github/workflows/ios.yml` súbore, zobrazenom na Výpise 4.2. To, kedy je spustená je zadefinované v časti `on`, takže na príkaz `push` alebo `pull_request` do `develop` vývojovej vetvy.

Jednotlivé fázy pipeline sú definované pod `jobs`, konkrétne v `steps`. Krok `Checkout` naklonuje kód z repozitára do prostredia runneru. `Build` ho zostaví a `Test` otestuje.

Ďalším krokom by v budúcnosti mohlo byť aj zavedenie *Continuous Deployment* (CD), čo by zaručilo aj automatizované nasadenie aplikácie do produkcie. CD však nie je použíté, pretože aplikácia nemá zakúpený Apple Developer Program a tak nasadenie nie je možné. Pri použití CD by bolo potrebné pozmeniť aj celkové nastavenie GitHub Actions a využiť vlastný runner, keďže v neplatenej verzii nie je zahrnuté CD. Pre účely tejto bakalárskej práce je však použitie bezplatnej verzie GitHub Actions postačujúce, keďže je to najrýchlejšia a najvýhodnejšia verzia.

4.2 Návrhové vzory

Táto sekcie popisuje dva vybrané návrhové vzory – *Repository* a *Dependency Injection*, využité v implementácií vo viacerých prípadoch. Vďaka zakomponovaniu týchto vzorov do implementácie je zvýšená modularita a celková udržateľnosť kódu.

```yaml
name: iOS workflow

on:
  push:
    branches: [ "develop" ]
  pull_request:
    branches: [ "develop" ]

jobs:
  build:
    name: Build and Test scheme using iPhone 14 simulator
    runs-on: macos-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Set Default Scheme
        run: |
          ...
        working-directory: ./src
      - name: Build
        env:
          platform: ${{ 'iOS Simulator' }}
        run: |
          ...
        working-directory: ./src
      - name: Test
        env:
          platform: ${{ 'iOS Simulator' }}
        run: |
          ...
        working-directory: ./src
```

Listing 4.2: Ukážka súboru ios.yml

4.2.1 Repository Pattern

V rámci dodržania nefunkčného požiadavku o modularite aplikácie a využitia trojvrstvovej architektúry z návrhu, je vhodné zakomponovanie návrhového vzoru Repository do implementácie. Návrhový vzor Repository zaručuje izoláciu dátovej vrstvy od ostatku aplikácie [70]

V jazyku Swift je možné Repository pattern definovať pomocou rozhrania `protocol`, a implementovať jednotlivými triedami. Názorné použitie je možné vidieť vo Výpise 4.3. Tento príklad predstavuje rozhranie založené na *CRUD* operáciach (Create, Read, Update, Delete). Je vďaka nemu možné vytvárať, načítať, aktualizovať a mazať jednotlivých používateľov v databáze.

Vďaka takto definovanému rozhraniu je možné kedykoľvek ľahko zmeniť

samotnú implementáciu rozhrania, a tak napríklad zmeniť databázu, či spôsob ukladania fotiek.

```
protocol UserRepository {
    func create(user: User)
    func read(id: Int) -> User?
    func update(id: Int, name: String, email: String)
    func delete(id: Int)
}

class UserRepositoryImplementation: UserRepository {
    func create(user: User) {
        ...
    }

    func read(id: Int) -> User? {
        ...
    }

    func update(id: Int, name: String, email: String {
        ...
    }

    func delete(id: Int) {
        ...
    }
}
```

Listing 4.3: Ukážka použitia návrhového vzoru Repository v jazyku Swift

4.2.2 Dependency Injection

Na využitie funkcionality implementácie jednotlivých Repositories je využívaný návrhový vzor *Dependency Injection*. Tento návrhový vzor zaručuje, že ak komponenta potrebuje využívať externé zdroje alebo iné závislosti, tieto závislosti sú do nej predané cez konštruktor alebo metódy nastavujúce hodnoty. Vďaka tomu si komponenta tieto závislosti nemusí vytvárať alebo spravovať sama.

Pre príklad, ak komponenta A závisí na komponente B, namiesto toho, aby A vytvárala inštanciu B sama, inštancia B je A predaná pomocou externého systému. Komponenta A tak nevie, ako je B vytvorená, čo uľahčuje nahradenie rôznych implementácií B bez nutnosti meniť kód A.

Pre implementáciu Dependency Injection je využitý framework *Resolver*, pomocou ktorého sú jednolivé implementácie Repositories zaregistrované a využívané vo `ViewModels` pomocou `@Injected` property wrapper, ako je znázornené vo Výpise 4.4.

```
import Resolver

class UserManager {
    @Injected private var userRepository: UserRepository

    func createUser(name: String, email: String) {
        let id = Int.random(in: 1...10000)
        let user = User(id: id, name: name, email: email)
        userRepository.createUser(user)
    }
}
```

Listing 4.4: Ukážka použitia frameworku Resolver pre návrhový vzor Dependency Injection

Vďaka využitiu Repositories a Dependency Injection je nielen oddelená dátová vrstva od zvyšku aplikácie, ale je taktiež možné jednotlivé implementácie ľahko nahradiť za iné a je dodržaná modularita jednotlivých komponent.

4.3 Spojenie s Firebase

Pre zakomponovanie a prvotné prepojenie aplikácie s platformou *Firebase*, je potrebné splniť nesledovné kroky, aby bolo možné *Firebase* začať využívať:

1. **Vytvorenie nového projektu vo Firebase Console**.
2. **Pridanie Firebase do projektu**. Tento krok je podrobne rozpísaný vo Firebase Console a zahŕňa nasledujúce časti:
    a) **Zaregistrovanie aplikácie** pomocou *Apple Bundle ID*, ktoré je možné získať v záložke hlavného *target* v Xcode, pod *Bundle Identifier*.
    b) **Stiahnutie `GoogleService-Info.plist` súboru** z Firebase Console.
    c) **Pridanie `GoogleService-Info.plist` súboru** do koreňovej zložky projektu v Xcode.
    d) **Pridanie Firebase SDK** (Software Development Kit) do projektu, pomocou *Swift Package Manager*.
    e) **Nakonfigurovanie Firebase** v kóde pomocou funkcie `configure()`, ako je znázornené vo Výpise 4.5.

```
class AppDelegate: NSObject, UIApplicationDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
        FirebaseApp.configure()
        return true
    }
}
```

Listing 4.5: Ukážka konfigurácie Firebase

Po splnení krokov je možné ihneď začať využívať jednotlivé služby Firebase. V nasledujúcej sekcií je okrem iného priamo popísané aj využívanie dvoch služieb definovaných v návrhu – Authentication a Cloud Firestore.

4.4 iOS aplikácia

Táto sekcia má za úlohu poskytnúť detailnejší pohľad na implementačné časti aplikácie, ktoré sa riadia architektúrou stanovenou v návrhu a sú dôležitou súčasťou funkcionality aplikácie. V rámci tejto sekcie sú prezentované príklady kódu, ktoré slúžia ako konkrétne ilustrácie týchto detailov. Sekcia je rozdelená na tri časti – podľa jednotlivých vrstiev architektúry.

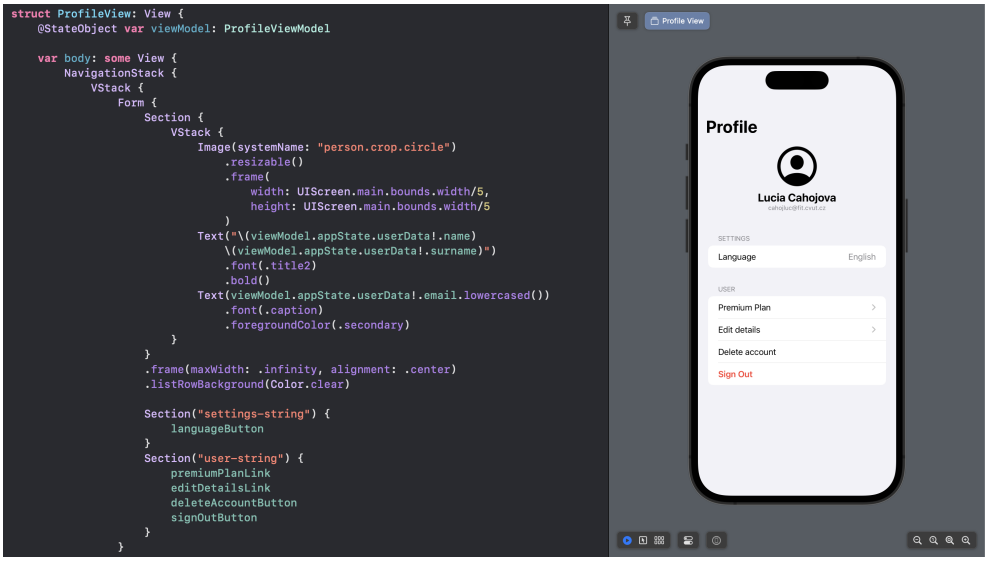
4.4.1 Prezentačná vrstva

Ako je definované v návrhu, prezentačná vrstva je tvorená jednotlivými Views a ViewModels. Views zobrazujú používateľské rozhranie a taktiež slúžia na interakciu s používateľom. Na ľavej časti Obrázku 3.2 je vidieť časť kódu View pre obrazovku Profil, ktorá demonštruje deklaratívnu syntax frameworku SwiftUI.

Jednotlivé časti UI, ako tlačidlá, textové polia, obrázky a podobne podliehajú rozhraniu View. Sú hierarchicky zadefinované a vďaka tomu, že všetky predstavujú vlastne samostatné Views ich SwiftUI pomocou svojho *rendering engine* dokáže jednoducho vykresliť na obrazovku.

Pre náhľad jednotlivých obrazoviek simultánne počas ich programovania je využívažívaný nástroj Live Preview, popísaný v časti návrhu, zobrazený v pravej časti Obrázku 3.2.

Na tom istom Obrázku je zároveň znázornené, že pre zobrazovanie potrebných dát `ProfileView` využíva svoj vlastný `viewModel`. Ak tieto dáta definované vo `viewModel` zmenia, framework obrazovku automaticky aktualizuje s novými dátami. Vďaka tomu je nie potrebné funkcionalitu pre aktualizáciu



Obr. 4.1: Ukážka využitia Live Preview a kódu obrazovky Profil

dát programovať samostatne a celková tvorba používateľského orzhrania je značne urýchlená.

4.4.2 Doménová vrstva

Doménovú vrstvu tvoria entity, ktoré majú svoj vlastný súbor atribútov či metód relevantných pre požiadavky aplikácie. Tieto entity tvoria jadro business logiky.

Okrem klasických modelov obsahujúcich jednotlivé atribúty danej entity je v doménovej vrstve zahrnuté aj rozhranie pre samotné filtre simulujúce analógový film. Toto rozhranie, ako aj jeho možnosti implementácie sú popísané v nasledujúcej časti práce.

4.4.2.1 Filtre

Pre implementáciu filtrov simulujúcich výzor analógového filmu je využitý framework CoreImage, výkonný nástroj na úpravu fotografií v jazyku Swift. CoreImage ponúka viac ako 200 filtrov na úpravu farieb, rozmazania, zaostrenia rôznych a iných. Tieto filtre je možné taktiež reťaziť, takže je vhodný na aplikovanie úprav popísaných v analýze tak, aby aplikácia spĺňala funkcionalitu definovanú v návrhu.

Kroky potrebné pre pridanie nového filtra simulujúceho analógový film pomocou frameworku CoreImage sú podrobne popísané v nasledovnej časti.

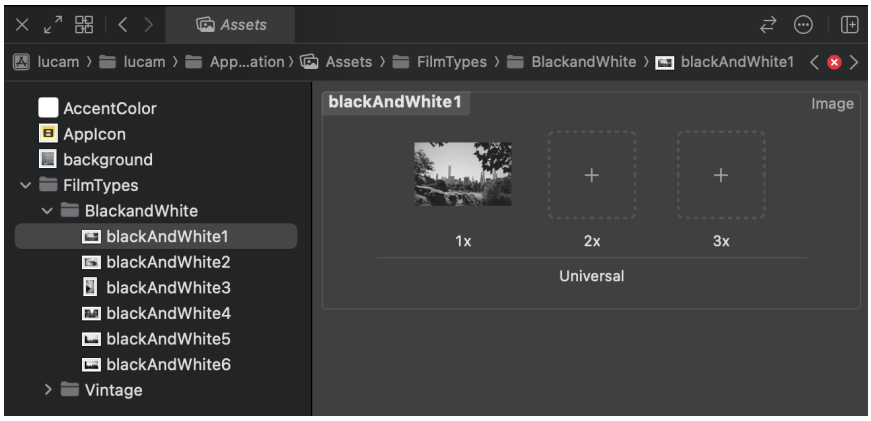
Pridanie filtra Jednotlivé filtre silmulujúce rôzne druhy analógových filmov sú v aplikácií realizované ako štruktúry implementujúce rozhranie `Filter`,

ktoré disponuje jednou funkciou – `applyFilter()`.

Pre zjednodušenie pridávania nového filtra a oddelenie závislostí jednotlivých tried, ako aj pre vyhnutie sa potreby inštanciovania jednotlivých tried je na implementáciu filtrov využitý vzor Dependency Injection, za použitia frameworku Resolver.

Nasledujúce kroky detailne popisujú proces pridania nového filtra s názvom BlackAndWhite do aplikácie:

1. **Pridanie náhľadov použitia filtra vo forme fotografií do zložky `Assets.xcassets`.** Ako je naznačené v návrhu, pri natiahnutí nového filmu si môže používateľ prezerať jednotlivé typy filmov a ich použitia na náhľadoch. Tieto náhľady sú čerpané zo zložky `Assets.xcassets`, ktorá je zkompilovaná vo fáze zostavenia projektu a jej štruktúra nie je zachovaná vo výslednom balíčku aplikácie. Pre optimalizáciu je zreorganizovaná aby bolo možné z nej načítavať jednolivé súbory čo najefektívnejšie. Z tohto dôvodu nie je možné programovo načítať jednotlivé zložky, podzložky a ich obsah. Preto je dôležité dbať na správnosť názvov fotografií pridaných do tejto zložky, ako je zobrazené na Obrázku 4.2.



Obr. 4.2: Ukážka pridania náhľadov použitia BlackAndWhite filtra do zložky `Assets.xcassets`

2. **Pridanie nového typu filmu do dátového typu enum s názvom `FilmType`.** Pomocou tejto enumerácie je možné zadefinovať meno filtra, ktoré je zobrazené pri výbere. Taktiež je v enume definovaný zoznam názvov náhľadových fotografií zo zložky `Assets.xcassets`, aby bolo možné k nim pristupovať za behu programu.

```
enum FilmType: String, CaseIterable, Codable {
    case blackAndWhite = "Black and White"

    var imageNames: [String] {
        switch self {
        case .blackAndWhite:
            return ["blackAndWhite1", ... ]
        }
    }
}
```

Listing 4.6: Ukážka enumerácie `FilmType`

3. **Pridanie implementácie rozhrania `Filter` pre daný typ filmu.**

```
protocol Filter {
    func applyFilter(image: inout UIImage)
}

struct BlackAndWhiteFilter: Filter {
    func applyFilter(image: inout UIImage) {
        ...
    }
}
```

Listing 4.7: Ukážka pridanie implementácie rozhrania `Filter`

4. **Pridanie typu filmu a príslušnej implementácie do `filterTable` tabuľky.** Jednotlivé implementácie rozhrania `Filter` z tabuľky sú najskôr prostredníctvom frameworku Resolver s pomocou ich názvu zaregistrované, ako je zobrazené na Výpise 4.8. Vďaka tejto registrácií s menom je možné za behu rozhodovať o použití konkrétnej implementácie.

```
extension Resolver: ResolverRegistering {
    static var filterTable: [(filmType: FilmType, filter: () ->
    ↪  Filter)] {[
        (.blackAndWhite, { BlackAndWhiteFilter() }),
        (.vintage, { VintageFilter() })
    ]}

    public static func registerAllServices() {
        for (filmType, filter) in filterTable {
            register(name: Resolver.Name(filmType.rawValue)) {
                filter()
            }
        }
        ...
    }
}
```

Listing 4.8: Ukážka registrácie jednotlivých implementácií rozhrania `Filter`

Takto pridaný nový filter simulujúci analógový film je po zostavení aplikácie ihneď k dispozícií pri natiahnutí nového filmu.

Ukážka filtra Ako už bolo spomenuté, vďaka využitiu frameworku CoreImage, ktorý dokáže jednotlivé filtre reťaziť a disponuje veľkým množstvom jednotlivých filtrov, bolo možné zjednodušene špecifikovať, ako má výsledný obrázok vyzerať.

Vo Výpise 4.9 je možné vidieť implementáciu farebného filtra, ktorá využíva možnosti úprav z časti analýzy tejto práce. Pre využitie filtra je potrebné zadefinovať jeho presný názov a v parametroch špecifikovať hodnoty pre jeho aplikáciu.

Pre príklad, filter `CIHighlightShadowAdjust` frameworku CoreImage je špecifikovaný dvomi parametrami – `inputShadowAmount` a `inputHighlightAmount`. Hodnota *highlights* zostáva nezmenená, zatiaľ čo hodnota *shadows* je zvýšená, a tak sú oblasti tieňov vo výslednom obrázku svetlejšie.

V aplikácií sú implementované dva druhy filtra, čiernobiely a farebný. Ukážka ich aplikovania na fotografie je zobrazená v Prílohe B.

4.4.3 Dátová vrstva

Dátová vrstva implementovanej aplikácie je zodpovedná za riadenie a komunikáciu so zvolenými úložiskami. V nasledujúcej časti je popísaný spôsob správy jednotlivých používateľov v aplikácií za použitia platfotmy Firebase a taktiež správa fotografií za použitia triedy *FileManager*.

```
// CIHighlightShadowAdjust
let paramsCIHighlightShadowAdjust: [String: AnyObject] = [
    "inputShadowAmount": NSNumber(value: 0.2),
    "inputHighlightAmount": NSNumber(value: 1.0)
]
...
func applyFilter(image: inout UIImage) {
    if var imageCI = CIImage(image: image) {
        applyCIFilter(name: "CIHighlightShadowAdjust", parameters:
        ↪  paramsCIHighlightShadowAdjust, image: &imageCI)
        applyCIFilter(name: "CIToneCurve", parameters:
        ↪  paramsCIToneCurve, image: &imageCI)
        applyCIFilter(name: "CIWhitePointAdjust", parameters:
        ↪  paramsCIWhitePointAdjust, image: &imageCI)
        applyCIFilter(name: "CIColorCrossPolynomial", parameters:
        ↪  paramsCIColorCrossPolynomial, image: &imageCI)
        applyGrain(image: &imageCI)

        let processedCGImage = CIContext().createCGImage(imageCI, from:
        ↪  imageCI.extent)
        image = UIImage(cgImage: processedCGImage!, scale: image.scale,
        ↪  orientation: image.imageOrientation)
    }
}
```

Listing 4.9: Ukážka implementácie funkcie `applyFilter()`

4.4.3.1 Správa Firebase

Komunikácia so službami Firebase predstavuje dôležitú súčasť dátovej vrstvy zvolenej trojvrstvovej architektúry, popísanej v návrhu. Pre jej implementáciu je použitý návrhový vzor Repository, ktorý zabezpečuje oddelenie zdrojov dát od zvyšku aplikácie. Využitím tohto vzoru je zabezpečená flexibilita aplikácie, pretože je kedykoľvek možné prejsť z platformy Firebase na akúkoľvek inú platformu, bez vplyvu na zvyšok aplikácie. To umožňuje ľahkú zmenu zdrojov dát bez nutnosti zmeny logiky aplikácie.

Táto podsekcia sa venuje implementačným detailom komunikácie so službami platformy Firebase a konkrétnym príkladom kódu s využitím návrhového vzoru Repository.

Authentication Rozhranie funkcií potrebných pre registráciu používateľa, jeho prihlásenie, odhlásenie, zmazanie používateľa, či obnovu hesla je zobrazené na Výpise 4.10, ako `AuthenticationRepository`. Vo výpise je taktiež zobrazená aj časť implementácie tohto rozhrania využívajúca Firebase Authentication, v štruktúre `FirebaseAuthenticationImplementation`.

```
protocol AuthenticationRepository {
    func registerUser(email: String, password: String) async throws
    func logInUser(email: String, password: String) async throws
    func signOutUser() throws
    func deleteUser() async throws
    func isUserLoggedIn() -> Bool
    func sendPasswordReset(email: String) async throws
}

struct FirebaseAuthenticationImplementation: AuthenticationRepository {
    func registerUser(email: String, password: String) async throws {
        do {
            try await Auth.auth().createUser(withEmail: email,
            ↪  password: password)
        } catch let error as NSError {
            switch error.code {
            case AuthErrorCode.networkError.rawValue:
                throw FirebaseError.networkError
            ...
            default:
                throw FirebaseError.defaultError
            }
        }
    }
    ...
}
```

Listing 4.10: Ukážka rozhrania AuthenticationRepository a jeho implementácie pomocou Firebase

Cloud Firestore Tak ako popisuje návrh, Cloud Firestore je využívaný pre zaznamenávanie detailnejších údajov o používateľovi, ktoré nie je možné uchovaváť použitím samotnej služby Authentication. Pre ukladanie týchto údajov a komunikáciu s Cloud Firesotre je využité rozhranie `UserRepository`, a jeho implementácia `FirebaseUserImplementation`. Pomocou tohto repozitára je možné vytvárať, načítať a mazať údaje o používateľovi.

Spojením služieb Authentication a Cloud Repository v implementácií aplikácie je tak naplnená funkcionalita používateľského rozhrania definovaná v návrhu, konkrétne časť týkajúca sa manipulácie s jednotlivými používateľskými účtami.

4.4.3.2 Správa fotografií

Aby aplikácia splnila nefunkčný požiadavok o minimalizovaní nákladov na jej prevádzku, zhotovené fotografie sú spravované v úložisku len ak je používateľ prihlásený, tak ako je bližšie popísané v návrhu. Autorka sa tak

```
protocol UserRepository {
    func createUser(newUser: UserData) async throws
    func readUser() async throws -> UserData?
    func deleteUser() async throws
}

struct FirebaseUserImplementation: UserRepository {
    private let db = Firestore.firestore()

    func createUser(newUser: UserData) async throws {
        guard let user = Auth.auth().currentUser else { return }
        let userID = user.uid
        let docRef = db.collection("users").document(userID)
        do {
            try await docRef.setData(
                ["email": newUser.email,
                "name": newUser.name,
                "surname": newUser.surname,
                "premium" : newUser.premium]
            )
        } catch {
            throw FirebaseError.defaultError
        }
    }
    ...
}
```

Listing 4.11: Ukážka rozhrania UserRepository a jeho implementácie pomocou Firebase

rozhodla učiniť, aby nebolo potrebné na ich skladovanie používať Firebase (prípadne inú platformu) a platiť za využitý priestor pre ukladanie dát.

Z toho dôvodu je ako úložisko pre fotografie použitý framework *Foundation* a jeho trieda FileManager, ktorá ponúka funkcionalitu na manipuláciu nielen s fotografiami. FileManager je z bezpečnostných dôvodov obmedzený na manipuláciu so súbormi v izolovanom prostredí aplikácie, taktiež zvanom aj *sandbox*.

Pre manipuláciu s jednotlivými fotografiami a filmami je opäť využitý návrhový vzor Repository. Rozhranie na správu fotografií `FilmRepository` a je aj spolu s implementáciou využívajúcou FileManager zobrazené na Výpise 4.12.

Vďaka využitiu návrhového vzoru Repository by v budúcnosti pri zvýšení finančných kapacít na náklady bolo možné jednoducho pridať implementáciu využívajúcu platený Firebase Storage. Vďaka tomu by bolo možné mať prístup k filmom a fotografiám na viacerých zariadeniach a synchronizovať využívanie aplikácie a pridávanie fotografií z týchto zariadení.

```
protocol FilmRepository {
    func createFilm(filmName: String, filmType: FilmType, filmSize:
    ↪  FilmSize) throws
    func loadAllFilms() throws -> [Film]
    func deleteAllFilms()
    func deleteFilm(filmName: String) throws
    func markFilmFull(filmName: String) throws
    func markFilmDeveloped(filmName: String) throws
    func savePhoto(photo: Photo, image: UIImage, filmName: String,
    ↪  filmType: FilmType, filmSize: FilmSize) throws
    func readImage(with id: UUID) throws -> UIImage
    func filmNameExists(filmName: String) -> Bool
}

struct FileManagerFilmImplementation: FilmRepository {
    func createFilm(filmName: String, filmType: FilmType, filmSize:
    ↪  FilmSize) throws {
        var films = (try? loadFilmsJson()) ?? []
        films.append(Film(name: filmName, type: filmType, size:
        ↪  filmSize))
        try saveFilmsJson(films: films)
    }
    ...
}
```

Listing 4.12: Ukážka rozhrania FilmRepository a jeho implementácie pomocou FileManager

Kapitola 5

Testovanie

Táto kapitola sa venuje testovacej fázi vývoja softwaru. Kvalitne prevedené testovanie aplikácie dokáže zaručiť jej kvalitu a spoľahlivosť. Pomocou testov je možné identifikovať chyby a nefunkčné časti aplikácie a ich opravou tak znížiť riziko zlyhania. Detekcia chýb dokáže taktiež ušetriť prostriedky potrebné na vývoj aplikácie, keďže je detekované chyby možné opraviť v čo najskorších fázach a zabrániť tak ich propagácií do iných častí projektu.

Nasledujúce dve podsekcie sa venujú dvom typom testov prevedených na aplikácií – unit testom a používateľskému testovaniu.

5.1 Unit testovanie

Unit testy sú automatizované testy, ktorých úlohou je overovanie správnej funkčnosti kódu, takzvaných *units*, v preklade jednotiek kódu. Pre zhnutie, podľa [71] by mal správny unit test spĺňať nasledovné tri body:

- **Overenie malej časti kódu.**
- **Overenie je rýchle.**
- **Overenie je prevedené izolovaným spôsobom.**

Pre otestovanie aplikácie pomocou unit testov je možné použiť *XCTest framework*. Každá testovacia trieda musí byť podtriedou `XCTestCase` a každá testovacia metóda musí začínť slovom `test`, aby bol XCode schopný testy spustiť. [72]

Pre testovanie tried s rôznymi externými závislosťami je potrebné nasimulovať správanie týchto externých zdrojov. Táto simulácia sa nazýva *mocking* a príklad jej použitia je možné vidiet vo Výpise 5.1, pre simulovanú implemenáciu rozrhania `FilmRepository`. V reálnom použití implementácia využíva úložisko na ukladanie fotiek a manipuláciu s nimi.

```
class FilmRepositoryMock: FilmRepository {
    var films = [Film]()

    func createFilm(filmName: String, filmType: FilmType, filmSize:
    ↪  FilmSize) throws {
        films.append(Film(name: filmName, type: filmType, size:
        ↪  filmSize))
    }
    ...
}
```

Listing 5.1: Ukážka simulácie FilmRepository pre unit testy

Po impelentovaní `FilmRepositoryMock` je možné otestovať samotnú triedu `FilmViewModel`, pomocou testov zadefinovaných vo `FilmViewModelTests`. Vo Výpise 5.2 je možné vidieť funkciu `setUp()`, v ktorej je vytvorená inštancia mock triedy `FilmRepositoryMock`. Táto funkcia je spravovaná samotným frameworkom a je zavolaná pred každým jedným testom.

```
class FilmViewModelTests: XCTestCase {
    var filmRepositoryMock: FilmRepositoryMock!
    var appState: AppState!
    var film: Film!
    var viewModel: FilmViewModel!

    override func setUp() {
        super.setUp()
        filmRepositoryMock = FilmRepositoryMock()
        appState = AppState()
        ...
    }

    func testHandleDevelopFilmButtonTap() {
        XCTAssertFalse(viewModel.film.isDeveloped, "Not developed")
        viewModel.handleDevelopFilmButtonTap()
        XCTAssertTrue(viewModel.film.isDeveloped, "Developed")
    }
    ...
}
```

Listing 5.2: Ukážka unit testov pre FilmViewModel

Správne správanie jednotlivých funkcií je overené pomocou takzvaných *assertov*, vo Výpise 5.2 použitých ako `XCTAssertTrue` a `XCTAssertFalse`. Tieto asserty sú funkcie, ktoré pri nesplnení podmienky vyvolajú chybu a zastavia

beh programu.

Pre zistenia percenta kódu pokrytého testami ponúka Xcode nástroj *Code Coverage*, ktorý nielenže vypíše percento pokrytia, ale aj zvýrazní riadky kódu ktoré boli vykonané počas testovania aplikácie. Vďaka tomu bolo pri implementácií možné ľahko zisťovať, ku ktorým častiam kódu chýbajú testy a následne ich doplniť.

5.2 Používateľské testovanie

Používateľské testovanie prestavuje proces testovania, ktorý zamestnáva ľudí ako účastníkov testovania a zástupcov cieľového publika. Vďaka tomu je možné vyhodnotiť mieru, do akej produkt spĺňa špecifické kritériá použiteľnosti.

V nasledujúcich podsekciách sú popísaní účastníci testovania, scenáre jednotlivých úloh na otestovanie, výsledky testovania a taktiež možné vylepšenia aplikácie, vyplývajúce zo spätnej väzby od účastníkov testovania.

5.2.1 Vybraní používatelia

Pre používateľské testovanie bolo vybraných päť osôb, vo veku od 18 do 57 rokov z blízkeho okruhu autorky práce. Všetky osoby používajú operačný systém iOS na dennej báze, všetci zo zúčastnených osôb buď fotili na analógový film v minulosti, alebo ho pravidelne používajú v súčasnosti. Jedna z obsôb taktiež pravidelne používa aplikáciu podobného charakteru ako testovaná aplikácia a dve osoby využívajú rôzne filtre na úpravu svojich fotografií.

5.2.2 Testovacie scenáre

Testy boli prevádzané na mobilných zariadeniach iPhone 8, iPhone 11 a iPhone 13 Max Pro. Mobilná aplikácia je do zariadení vopred nainštalovaná. Testy boli sústredené hlavne na správnu funkčnosť aplikácie a jej intuitívne ovládanie.

Jednotlivé testy boli vedené autorkou práce, ktorá pri testovaní zadávala používateľom jednotlivé inštrukcie, ktoré bolo potrebné previesť pre splnenie testu. Ak niektoré testy potrebovali prerekvizity ako prihlasovacie údaje, informácie boli poskytnuté používateľom vopred, napísané na papieri a predložené pred nimi. Jednotlivé scenáre sú uvedené v Prílohe A. Každý scenár má svoj identifikátor, názov a jednotlivé kroky potrebné pre jeho splnenie.

Testovacích scenárov je dokopy šesť a testujú procesy registrácie, natiahnutia nového filmu, vyfotenia a vyvolania filmu, uloženia filmu do galérie zariadenia, zmeny jazyka a odhlásenia.

5.2.3 Výsledky používateľského testovania

Všetci používatelia dokázali bez problémov a nápovedy splniť jednotlivé testovacie scenáre. Ich spätná väzba bola kladná, dva používatelia tiež odcenili možnosť zmeny jazyka na slovenský, keďže nehovoria anglicky. Taktiež sa im páčilo zachovanie vyvolávania filmu až po jeho vyfotení, vďaka čomu sa na výsledné fotografie tešili viac.

Čo sa týka funkčnosti aplikácie, používatelia neidentifikovali zásadné problémy alebo nefunkčnosť aplikácie. Pri scenári zahŕňajucom uloženie fotografie do galérie telefónu traja používatelia informovali, že si neboli istí, či sa fotografie naozaj uložili. Táto nejasnosť bola vyriešená prekryvom časti obrazovky s informáciou o uložení, ako je zobrazené na Obrázku 5.1. Tento text sa na krátky čas zobrazí a po chvíľke zase z obrazovky zmizne.

Obr. 5.1: Ukážka upozornenia o uložní fotografie

5.2.4 Možné vylepšenia

Na záver testovania boli používatelia dotazovaní na možné vylepšenia aplikácie a funkcionalitu, ktorá by im používanie aplikácie zjednodušila či sprí-jemnila. Jedna z používateliek by odcenila možnosť pridávať popis k fotografiám a jednotlivým filmom, aby si mohla zaznamenať tiež okolnosti danej fotky.

Ďalší používateľ zhodnotil, že by sa mu pre mazanie filmov páčila možnosť film podržať, pričom by sa mu zobrazili možnosti manipulácie s filmom, tak ako v galérií telefónu.

Kapitola 6

Výsledky a možný rozvoj aplikácie

Záverečná kapitola sa zaoberá zhrnutím výsledkov tejto bakalárskej práce a zároveň prezentuje možnosti ďalšieho možného rozvoja vyvíjanej aplikácie.

6.1 Výsledky práce

Táto sekcia sa zaoberá zhodnotením výsledkov jednotlivých fáz vývoja aplikácie a taktiež poskytuje ohodnotenie miery splnenia funkčných a nefunkčných požiadavkov.

6.1.1 Zhrnutie výsledkov

Analýza poskytla náhľad do problematiky analógového a digitálneho fotografovania s uvedním možností úprav digitálnych fotografií na analógové a taktiež prehľad existujúcich konkurenčných aplikácií. Vďaka dôkladnej analýze bolo potom možné presne stanoviť funkčné a nefunkčné požiadavky na aplikáciu s dôrazom na jednoduché a intuitívne ovládanie a autenticitu analógového fotografovania.

Vybrané vývojové prostredie Xcode sa osvedčilo ako vhodný nástroj na implementáciu a testovanie tejto mobilnej aplikácie. Nástroj Xcode Live Preview v kombinácií s frameworkom SwiftUI značne uľahčilo a zrýchlilo vývoj používateľského rozhrania.

Vďaka zvolenej trojvrstvovej architektúre a vzoru MVVM je aplikácie jednoducho rozšíriteľná o nové filtre a taktiež je možné ľahko zmeniť jednotlivé implementácie repozitárov pre ukladanie fotografií a zvolenou backendovou službou.

V implementačnej časti sa podarilo vyvinúť funkčnú aplikáciu pre platformu iOS, v testovacej časti otestovať jej funkcionalitu a upraviť identifikované nedostatky.

Výsledkom bakalárskej práce je funkčná mobilná aplikácia pre platformu iOS, simulujúca analógovú fotografiu. Ukážky jednotlivých obrazoviek je možné vidieť v Prílohe C.

6.1.2 Splnenie požiadavkov

Výsledná aplikácia spĺňa všetky nefunkčné požiadavky. Je funkčná pre platformu iOS, jednoducho rozšíriteľná o nové filmy, modulárna, poskytuje na výber dva jazyky a náklady na jej prevádzku sú úplne minimalizované.

V rámci nefunčných požiadavkov splňuje všetky požiadavky s prioritou *must have* a *should have* a taktiež niektoré požiadavky s prioritou *could have*.

6.2 Možný rozvoj aplikácie

Táto sekcia sa zaoberá niekoľkými smermi, ktorými sa môže aplikácia uberať v jej budúcom rozvoji.

Pôvodné funkčné požiadavky s prioritou typu *could have* a *will not have*, ktoré neboli implementované sú vhodným adeptom na prvé rozvíjanie funkcionality aplikácie v jej budúcich verziách.

Vďaka spätnej väzbe od používateľov z používateľského testovania boli navrhnuté vylepšenia aplikácie zahrňujúce pridanie popisu k jednotlivým fotografiám a filmom, pre zaznamenanie kontextu danej fotografie. Taktiež bolo používateľmi navrhnuté využívanie viacerých gest na ovládanie aplikácie, napríklad podržanie filmu pre zobrazenie možností jeho manipulácie, či potiahnutie filmu doľava pre jeho zmazanie. Tieto návrhy nijak neovplyvňujú výslednú funkciounalitu aplikácie, avšak spríjemňujú jej ovládanie.

Prirodzeným rozšírením aplikácie je aj pridanie možnosti videozáznamov.

Záver

Hlavným cieľom tejto bakalárskej práce bolo vytvorenie mobilnej aplikácie simulujúcej analógovú fotografiu pre zariadenia s operačným systémom iOS. Vývoj aplikácie sa riadil tradičnými postupmi softwarového inžinierstva, zahrňujúcimi analýzu, návrh, implementáciu a testovanie. Pre splnenie hlavného cieľa bolo potrebné splniť viaceré čiastkové ciele.

V úvodnej časti bola prevedená analýza problematiky analógového a digitálneho fotografovania a rešers konkurenčných aplikácií pre zariadenia s operačným systémom iOS. Nadobudnuté informácie slúžili ako podklad k zostaveniu funkčných a nefunkčných požiadavkov na výslednú aplikáciu. Funkčné požiadavky boli ďalej podporené prípadmi použitia a doménovým modelom.

V časti návrhu boli popísané možnosti vývoja aplikácií pre iOS platformu a využitie Firebase pre backendové služby. V tejto časti bol taktiež na základe predošlej analýzy požiadavkov vytvorený návrh architektúry aplikácie ako aj návrh používateľského rozhrania. Architektúra aplikácie je založená na princípoch trojvrstvovej architektúry, s využitím návrhového vzoru MVVM. Používateľské rozhranie kladie dôraz na intuitívne ovládanie a jednoduchý dizajn a jeho návrh je podložený wireframe náčrtmi jednotlivých obrazoviek.

Ďalšia časť práce sa venovala samotnej implementácií aplikácie. Popisuje využitie kontinuálnej integrácie, použité návrhové vzory, komunikáciu s platformou Firebase ale aj jednotlivé časti trojvrstvovej architektúry tvoriace štruktúru kódu. Jednotlivé sekcie sú pre lepšie porozumenie doplnené o zdrojové kódy.

V nadväzujúcej časti o testovaní je popísané využitie unit testovania počas celého vývoja aplikácie pre pokrytie zdrojového kódu. Na záver prebehlo taktiež používateľské testovanie s koncovými používateľmi aplikácie pre overenie celkovej funkčnosti aplikácie. Na základe výsledkov používateľského testovania boli prevedné menšie zmeny v používateľskom rozhraní pre jeho sprehľadnenie.

V záverečnej časti boli zhrnuté výsledky práce a tiež ďalší možný rozvoj vyvíjanej aplikácie. Tým boli všetky čiastkové ciele a taktiež aj hlavný cieľ splnené a teda došlo k spleniu zadania tejto bakalárskej práce.

Literatúra

[1] Voorhees, D. P.: *Guide to Efficient Software Design*. Springer Nature Switzerland AG, 2020, ISBN 978-3-030-28501-2.

[2] Eastman Kodak Company: George Eastman History[online]. 2023. [cit. 2023-04-18]. Dostupné z: https://www.kodak.com/en/company/page/george-eastman-history

[3] Laing, G.: Apple QuickTake 100 Retro Review [online]. 2022. [cit. 2023-04-18]. Dostupné z: https://www.cameralabs.com/apple-quicktake-100-retro-review/

[4] Science Museum Group: Kodak No 1 Camera [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://collection.sciencemuseumgroup.org.uk/objects/co8083879/kodak-no-1-camera-box-camera-roll-film-camera

[5] The Digital Camera Museum: Apple QuickTake 100 [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://www.digitalkameramuseum.de/en/cameras/item/apple-quicktake-100

[6] Roemer, C.: What is the Difference between Analog and Digital? [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://kodakdigitizing.com/blogs/news/what-is-the-difference-between-analog-and-digital

[7] CreativeLive, Inc.: The Ultimate Guide to Learning Photography: How to use your camera[online]. 2023. [cit. 2023-04-18]. Dostupné z: https://www.creativelive.com/photography-guides/how-does-a-camera-work

[8] Visual Education: How Do Cameras Work? [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://visualeducation.com/photography-course/how-cameras-work/

[9] William Harris, C. F.: How Light Works [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://science.howstuffworks.com/light2.htm

[10] Camera Obscura and World of Illusions Edinburgh: What Is a Camera Obscura? [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://www.camera-obscura.co.uk/article/what-is-a-camera-obscura

[11] Balihar, D.: Co je dírková komora [online]. 2023. [cit. 2023-04-18]. Dostupné z: https://www.pinhole.cz/cz/pinholecameras/whatis.html

[12] Gray, E.: What Is Focal Length in Photography? [online]. 2019. [cit. 2023-04-19]. Dostupné z: https://photographylife.com/what-is-focal-length-in-photography

[13] Adams, A.; Baker, R.: *The Camera*. Little, Brown, 2018, ISBN 9780316485425.

[14] Mansurov, N.: Understanding Aperture in Photography [online]. 2022. [cit. 2023-04-19]. Dostupné z: https://photographylife.com/what-is-aperture-in-photography

[15] Kábele, M.: Co je expoziční čas a jaký kdy použít? [online]. 2017. [cit. 2023-04-20]. Dostupné z: https://mariankabele.cz/365-fotorad/fotorada-4-co-je-expozicni-cas

[16] Peterson, B.: *Understanding Exposure: How to Shoot Great Photographs with a Film Or Digital Camera*. Amphoto Books, 2004, ISBN 9780817463007.

[17] Martin, R.: The End of Film Photography [online]. 2006. [cit. 2023-04-18]. Dostupné z: https://www.nyip.edu/photo-articles/cameras-and-gear/the-end-of-film-photography

[18] Burley, R.: *Disappearance of Darkness: Photography at the End of the Analog Era*. Princeton Architectural Press, 2012, ISBN 9781616890957.

[19] Benveniste, A.: In the Smartphone Era, Millennials Are Buying Film Cameras [online]. 2022. [cit. 2023-04-17]. Dostupné z: https://www.nytimes.com/2022/01/14/crosswords/film-camera-crossword-clue.html

[20] Kars, I.: 'You Only Have One Shot': How Film Cameras Won Over a Younger Generation [online]. 2022, 2022. [cit. 2023-04-17]. Dostupné z: https://www.theguardian.com/technology/2022/jun/05/you-only-have-one-shot-how-film-cameras-won-over-a-younger-generation

[21] Rawpixel: Analog film roll [online]. 2023. [cit. 2023-04-2]. Dostupné z: https://www.rawpixel.com/image/6040055/analog-film-roll-free-public-domain-cc0-image

[22] Marquardt, C.; Andrae, M.: *The Film Photography Handbook: Rediscovering Photography in 35mm, Medium, and Large Format*. Rocky Nook, 2019, ISBN 9781681985275.

[23] Advameg, Inc.: Photographic Film [online]. 2023. [cit. 2023-04-21]. Dostupné z: http://www.madehow.com/Volume-2/Photographic-Film.html

[24] Anchell, S.: *The Darkroom Cookbook*. Focal Press, 2008, ISBN 9780240810553.

[25] Dungarwal, S.: Film [online]. 2019. [cit. 2023-04-21]. Dostupné z: https://www.shreyansdungarwal.com/blog/2019/4/15/understanding-black-amp-white-film-photography

[26] Urešová, E.: Filmy do klasických analogových fotoaparátů [online]. 2022. [cit. 2023-04-22]. Dostupné z: https://www.instaxstore.cz/blog/filmy-do-klasickych-analogovych-fotoaparatu/

[27] Fialová, K.: Jak vybrat filmový materiál [online]. 2022. [cit. 2023-04-22]. Dostupné z: https://www.fotoskoda.cz/3760-jak-vybrat-filmovy-material/

[28] Doušová, M.: Analog: jak zvětšovat fotky? [online]. 2019. [cit. 2023-04-22]. Dostupné z: https://www.fotoskoda.cz/2407-analog-jak-zvetsovat-fotky/

[29] Cambridge in Colour: Digital Camera Sensors [online]. 2020. [cit. 2023-04-23]. Dostupné z: https://www.cambridgeincolour.com/tutorials/camera-sensors.htm

[30] Gillmor, C.: How Does a Digital Camera Sensor Work? [online]. 2018. [cit. 2023-04-23]. Dostupné z: https://medium.com/tech-update/how-does-a-digital-camera-sensor-work-1342974250fd

[31] SPQR: From Photosites to Pixels [online]. 2021. [cit. 2023-04-23]. Dostupné z: https://pixelcraft.photo.blog/2021/05/10/from-photosites-to-pixels-i-the-process/

[32] Canlas, J.; Kalp, K.: *Film Is Not Dead: A Digital Photographer's Guide to Shooting Film*. Pearson Education, 2012, ISBN 9780132907125.

[33] Doušová, M.: Analog: proč fotit na film? [online]. 2019. [cit. 2023-04-22]. Dostupné z: https://www.fotoskoda.cz/2368-analog-proc-fotit-na-film/

[34] Deguzman, K.: How to Make Photos Look Like Film [online]. 2021. [cit. 2023-04-23]. Dostupné z: https://www.studiobinder.com/blog/how-to-make-photos-look-like-film/

[35] Virk, F. A.: How to Make an Image Brighter [online]. 2023. [cit. 2023-04-23]. Dostupné z: https://www.educative.io/answers/how-to-make-an-image-brighter

[36] Kornprobst, P.; Tumblin, J.; Durand, F.: Bilateral Filtering: Theory and Applications [online]. *Foundations and Trends in Computer Graphics and Vision*, 01 2009, doi:10.1561/0600000020.

[37] Rosenthal, M.: Emulating Film: What are Look Up Tables (LUTs)? [online]. 2015. [cit. 2023-04-23]. Dostupné z: https://www.videomaker.com/article/c01/18284-emulating-film-what-are-look-up-tables-luts/

[38] Mansurov, N.: What is Vignetting? [online]. 2020. [cit. 2023-04-23]. Dostupné z: https://photographylife.com/what-is-vignetting

[39] Calague, C.: A Simple Guide To Create Light Leaks in Photography [online]. 2020. [cit. 2023-04-23]. Dostupné z: https://www.cartoonize.net/create-light-leaks-in-photography/

[40] Manhole, Inc.: Huji Cam [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://apps.apple.com/us/app/huji-cam/id781383622

[41] Screw Bar Inc.: Gudak Cam Lite [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://apps.apple.com/us/app/gudak-cam-lite/id1449791761

[42] Hipstamatic, LLC: Hipstamatic [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://apps.apple.com/us/app/hipstamatic/id1450672436

[43] Mlejnek, J.: Analýza a sběr požadavků [online]. 2022. [cit. 2023-04-24]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/532096/mod_resource/content/7/03.prednaska.pdf

[44] ProductPlan: MoSCoW Prioritization [online]. 2022. [cit. 2023-04-24]. Dostupné z: https://www.productplan.com/glossary/moscow-prioritization/

[45] Gorbachenko, P.: What are Functional and Non-Functional Requirements and How to Document These [online]. 2023. [cit. 2023-04-24]. Dostupné z: https://enkonix.com/blog/functional-requirements-vs-non-functional/

[46] Miles, R.; Hamilton, K.: *Learning UML 2.0: A Pragmatic Introduction to UML*. O'Reilly Media, 2006, ISBN 9780596555221.

[47] Mlejnek, J.: Návrh softwarových systémů [online]. 2022. [cit. 2023-04-26]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/532107/mod_resource/content/6/05.prednaska.pdf

[48] Casserly, M.: Phone OS and iOS: Every Version Released So Far [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://www.macworld.com/article/1659017/ios-versions-list.html

[49] Ekren, E. K.: What Is Xcode and How to Use It? [online]. 2022. [cit. 2023-04-26]. Dostupné z: https://www.netguru.com/blog/what-is-xcode-and-how-to-use-it

[50] Nekrasov, A.: How to Write iOS Apps Without Xcode [online]. 2020. [cit. 2023-04-26]. Dostupné z: https://betterprogramming.pub/writing-ios-apps-without-xcode-89450d0de21a

[51] Apple Inc.: Distributing Your App to Registered Devices [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://developer.apple.com/documentation/xcode/distributing-your-app-to-registered-devices

[52] The Upwork Team: Swift vs. Objective-C: A Look at iOS Programming Languages [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://www.upwork.com/resources/swift-vs-objective-c-a-look-at-ios-programming-languages

[53] Wongpatcharapakorn, S.: How to use SwiftUI as UIView in Storyboard [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://sarunw.com/posts/swiftui-view-as-uiview-in-storyboard/

[54] L., J.: UIKit vs. SwiftUI: How to Choose the Right Framework for Your App [online]. 2022. [cit. 2023-04-26]. Dostupné z: https://getstream.io/blog/uikit-vs-swiftui/

[55] Lee, A. V. D.: SwiftUI Previews: Validating views in different states [online]. 2022. [cit. 2023-04-26]. Dostupné z: https://www.avanderlee.com/swiftui/previews-different-states/

[56] Jayani, P.: UIKit vs. Swiftui [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://bluewhaleapps.com/blog/uikit-vs-swiftui

[57] Google Inc.: Firebase Authentication [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://firebase.google.com/docs/auth

[58] Google Inc.: Cloud Firestore [online]. 2023. [cit. 2023-04-26]. Dostupné z: https://firebase.google.com/docs/firestore

[59] Martin, R.: *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2018, ISBN 9780134494166.

[60] Martin, R. C.: The Clean Architecture [online]. 2012. [cit. 2023-04-28]. Dostupné z: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

[61] Mishra, R.: Difference Between MVC and MVVM Architecture Pattern in Android [online]. 2022. [cit. 2023-04-28]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-and-mvvm-architecture-pattern-in-android/

[62] Apple, Inc.: About App Development with UIKit [online]. 2023. [cit. 2023-04-28]. Dostupné z: https://developer.apple.com/documentation/uikit/about_app_development_with_uikit

[63] Khan, A.: MVVM and SwiftUI [online]. 2023. [cit. 2023-04-28]. Dostupné z: https://www.appypie.com/mvvm-swiftui-how-to

[64] Rubin, D.: The Three Layered Architecture [online]. 2021. [cit. 2023-04-28]. Dostupné z: https://medium.com/@deanrubin/the-three-layered-architecture-fe30cb0e4a6

[65] Apple Inc.: Human Interface Guidelines [online]. [cit. 2023-04-29]. Dostupné z: https://developer.apple.com/design/human-interface-guidelines/guidelines/overview

[66] PROTOIO Inc.: Proto.io [online]. 2023. [cit. 2023-04-29]. Dostupné z: https://proto.io/

[67] Amazon Web Services, Inc.: Continuous Integration Explained [online]. 2023. [cit. 2023-04-30]. Dostupné z: https://aws.amazon.com/devops/continuous-integration/

[68] Jirutka, J.: Runnery [online]. 2023. [cit. 2023-04-29]. Dostupné z: https://help.fit.cvut.cz/gitlab/ci/runners.html

[69] GitLab: SaaS Runners on macOS (Beta) [online]. 2023. [cit. 2023-04-30]. Dostupné z: https://docs.gitlab.com/ee/ci/runners/saas/macos_saas_runner.html

[70] Fowler, M.: *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003, ISBN 9780321127426.

[71] Khorikov, V.: *Unit Testing Principles, Practices, and Patterns: Effective testing styles, patterns, and reliable automation for unit testing, mocking, and integration testing with examples in C#*. Manning, 2020, ISBN 9781617296277.

[72] Lee, A. V. D.: Getting Started with Unit Tests in Swift [online]. 2022. [cit. 2023-05-01]. Dostupné z: `https://www.avanderlee.com/swift/unit-tests-best-practices/`

Dodatok **A**

Ukážky scenárov používateľského testovania

A.1 Registrácia

Identifikátor: TS1

Prerekvizity: Voľná e-mailová adresa, ktorú je možné použiť na registráciu.

Testovací scenár:

1. Zapnite aplikáciu a počkajte na načítanie východzej obrazovky.
2. Kliknite na tlačidlo Registrácia.
3. Do jednotlivých polí formulára zadajte potrebné údaje. Ak nechcete zadávať údaje vlastné, môžete si ich vymyslieť. E-mailovú adresu môžete vyplniť s adresou na papieri pred vami.
4. Stlačte tlačidlo Registrovať, v dolnej časti obrazovky.

A.2 Natiahnutie nového filmu

Identifikátor: TS2

Prerekvizity: Používateľ je prihlásený do aplikácie.

Testovací scenár:

1. Zapnite aplikáciu a prejdite do záložky Fotoaparát.
2. Pokúste sa o odfotenie fotografie.

3. Po zobrazení notifikácie potvrďte natiahnutie nového filmu.
4. Vyplňte údaje o novom filme, zahŕňajúce jeho názov, počet snímkov a typ filmu.
5. Potvrďte natiahnutie filmu stlačením tlačidla Vložiť film v dolnej časti obrazovky.

A.3 Vyfotenie a vyvolanie filmu

Identifikátor: TS3

Prerekvizity: Používateľ je prihlásený do aplikácie a má natiahnutý film s názvom Test vo fotoaparáte.

Testovací scenár:

1. Zapnite aplikáciu a prejdite do záložky Fotoaparát.
2. Odfoťte akékoľvek fotografie a vyčerpajte tak kapacitu natiahnutého filmu tak, aby bola v pravom hornom rohu obrazovky 0, značiaca prázdny film.
3. Prejdite do záložky Galéria.
4. Použitím vyhľadávača nájdite vyfotený film s názvom Test.
5. Kliknite na tlačidlo vyvolať v pravej časti sekcie tohto filmu.

A.4 Uloženie filmu

Identifikátor: TS4

Prerekvizity: Používateľ je prihlásený do aplikácie a v záložke Galéria sa nachádza vyvolaný film s názvom Test.

Testovací scenár:

1. Zapnite aplikáciu a uistite sa, že ste v záložke Galéria.
2. Pomocou vyhľadávania nájdite film s názvom test.
3. Otvorte samostanú záložku filmu kliknutím kamkoľvek na ukážku filmu.
4. V pravej hornej časti obrazovky kliknite na možnosti a vyberte možnosť Uložiť.

A.5 Zmena jazyka aplikácie

Identifikátor: TS5

Prerekvizity: Používateľ je prihlásený do aplikácie.

Testovací scenár:

1. Zapnite aplikáciu a prejdite do záložky Profil.
2. Kliknite na sekciu Jazyk a počkajte na presmerovanie do Nastavení.
3. V nastaveniach zvoľte angličtinu ako jazyk aplikácie.
4. Použite navigátor v ľavej hornej časti obrazovky pre spätné presmerovanie do aplikácie.

A.6 Odhlásenie

Identifikátor: TS6

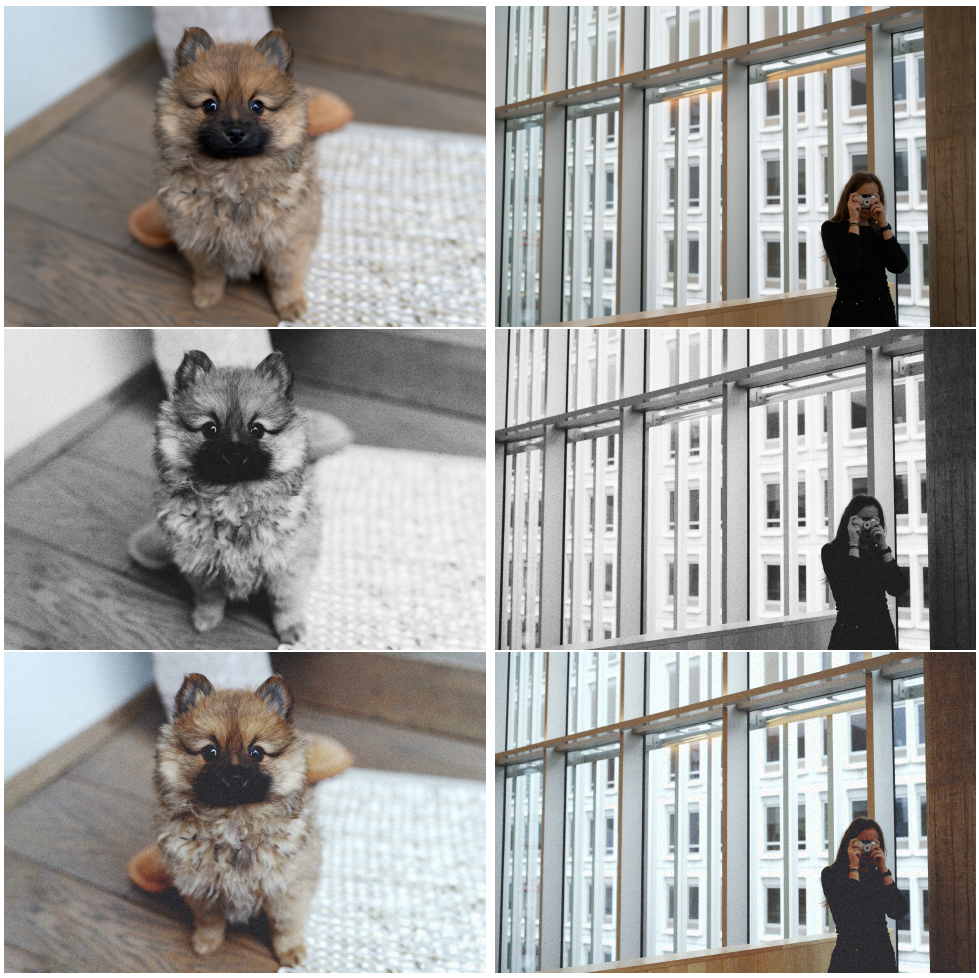
Prerekvizity: Používateľ je prihlásený do aplikácie.

Testovací scenár:

1. Zapnite aplikáciu a prejdite do záložky Profil.
2. Kliknite na tlačidlo Odhlásenie.
3. Po zobrazení notifikácie potvrďte odhlásenie.

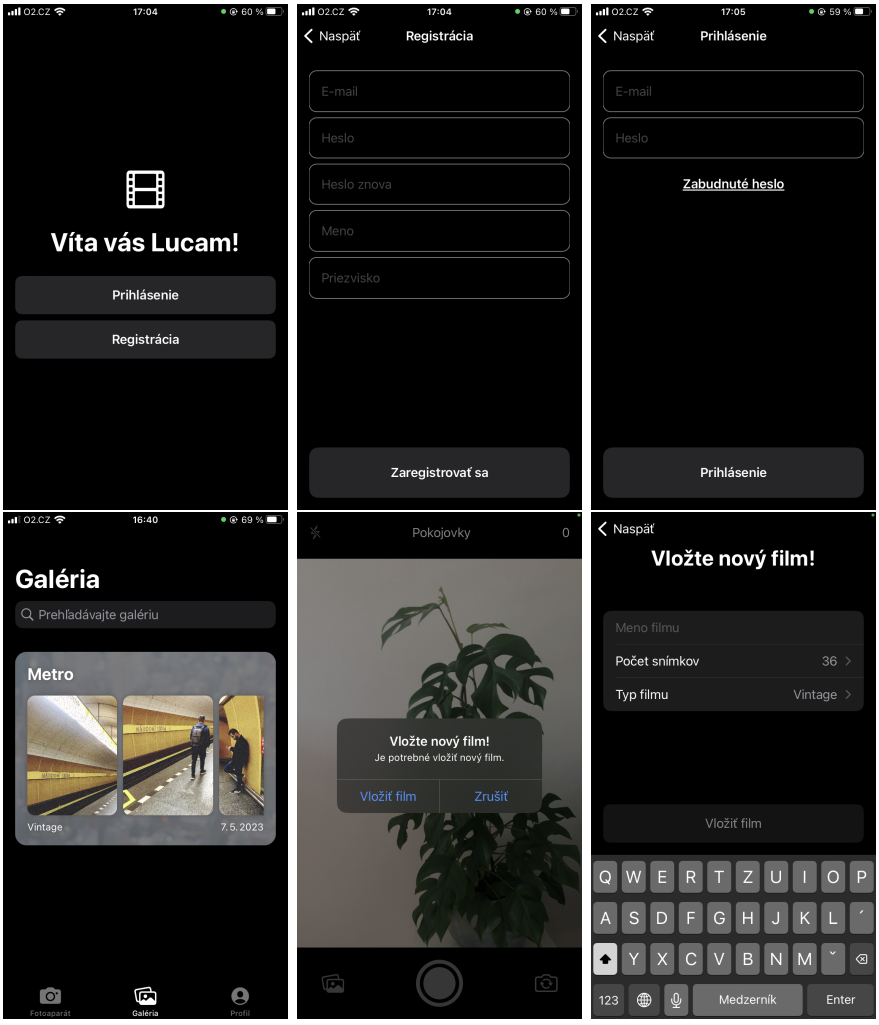
Dodatok B

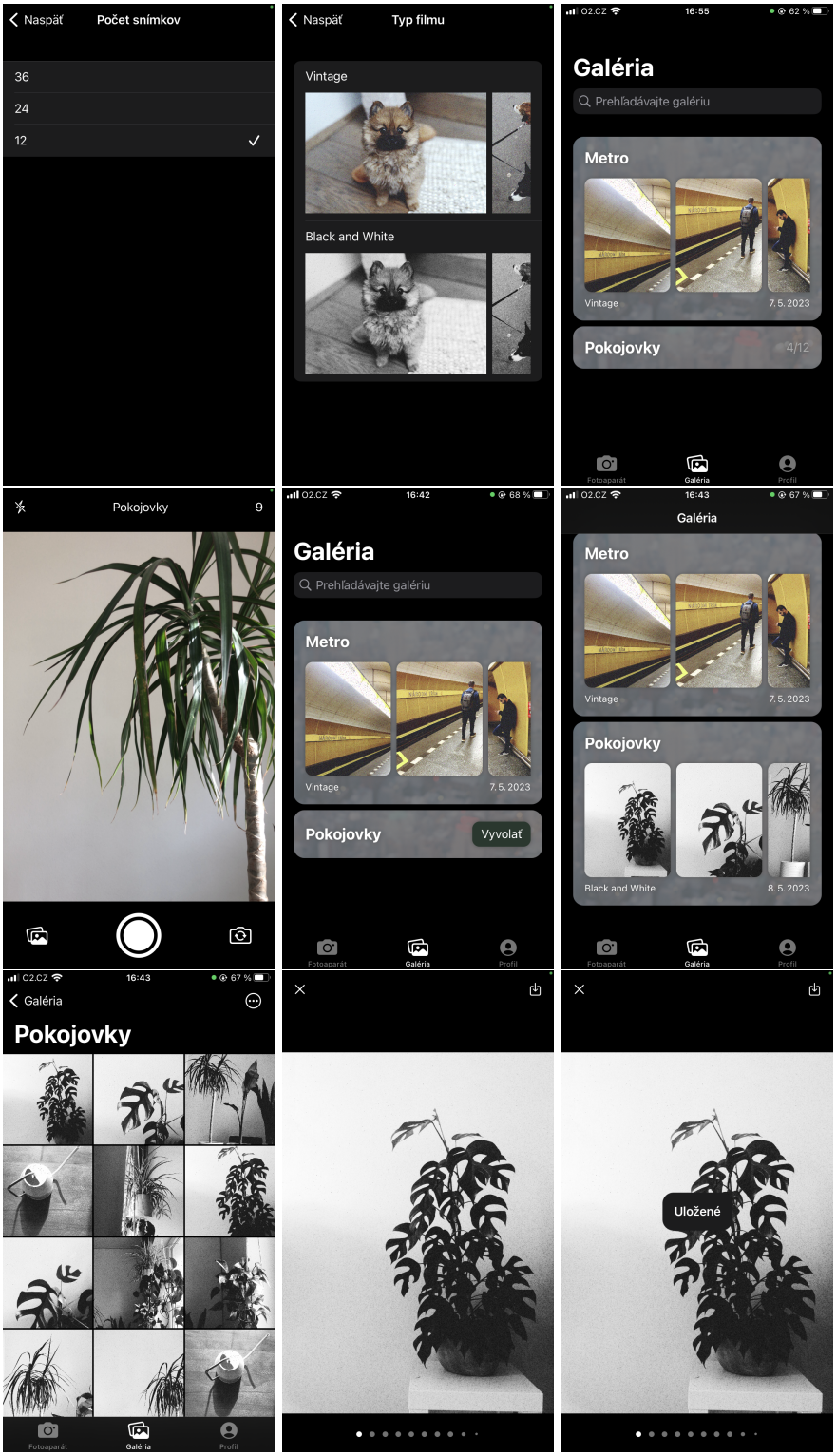
Ukážky filtrov aplikovaných na fotogafie

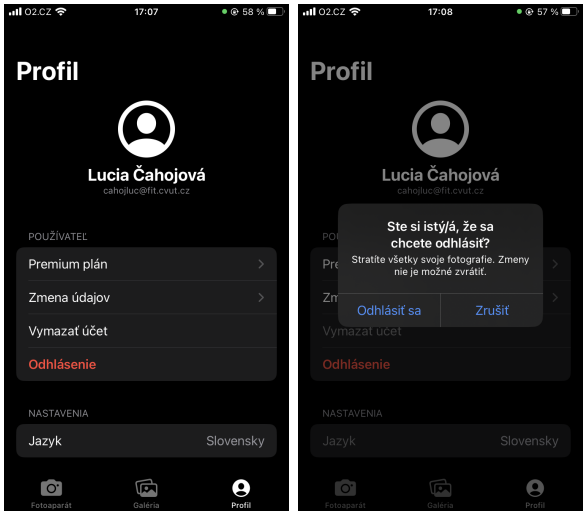


Dodatok C

Ukážky výslednej aplikácie




Naspäť
Počet snímkov
36
24
12
Naspäť
Typ filmu
Vintage
Black and White
Galéria
Prehľadávajte galériu
Metro
Vintage
7. 5. 2023
Pokojovky
4/12
Fotoaparát
Galéria
Profil
Pokojovky
9
Galéria
Prehľadávajte galériu
Metro
Vintage
7. 5. 2023
Pokojovky
Vyvolať
Galéria
Metro
Vintage
7. 5. 2023
Pokojovky
Black and White
8. 5. 2023
Galéria
Pokojovky
Uložené


O2.CZ
17:07
58 %
Profil
Lucia Čahojová
cahojluc@fit.cvut.cz
POUŽÍVATEĽ
Premium plán
Zmena údajov
Vymazať účet
Odhlásenie
NASTAVENIA
Jazyk
Slovensky
Fotoaparát
Galéria
Profil
O2.CZ
17:08
57 %
Profil
Lucia Čahojová
cahojluc@fit.cvut.cz
Ste si istý/á, že sa chcete odhlásiť?
Stratíte všetky svoje fotografie. Zmeny nie je možné zvrátiť.
Odhlásiť sa
Zrušiť
Vymazať účet
Odhlásenie
NASTAVENIA
Jazyk
Slovensky
Fotoaparát
Galéria
Profil

Dodatok D

Zoznam použitých skratiek

CD Continuous deployment

CI Continuous integration

CRUD Create, Read, Update, Delete

GUI Graphical user interface

HIG Human Interface Guidelines

IDE Integrated development environment

MVC Model-View-Controller

MVVM Model-View-ViewModel

MoSCoW Must have, Should have, Could have, Won't have

SDK Software Development Kit

UI User interface

Dodatok E

Obsah prílohy

- `readme.txt` ..... stručný popis obsahu prílohy
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `examples`
  - `screenshots` ..... snímky jednotlivých obrazoviek aplikácie
  - `video` ..... video na ukážku funkčnosti aplikácie
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF