Mendelova univerzita v Brně
Provozně ekonomická fakulta

Mobilná aplikácia na generovanie farebných schém z fotografií

Bakalárska práca

Vedúci práce:

Ing. Mgr. Jana Dannhoferová, Ph.D.

Ivan Mudronček

Brno 2016

Čestné prehlásenie

Prehlasujem, že som túto prácu: **Mobilná aplikácia na generovanie farebných schém z fotografií**
vypracoval samostatne a všetky použité zdroje a informácie sú uvedené v zozname použitej literatúry. Súhlasím, aby moja práca bola zverejnená v súlade s § 47b zákona č. 111/1998 Sb., o vysokých školách v znení neskorších predpisov, a v súlade s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Som si vedomý, že sa na moju prácu vzťahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brne má právo na uzatvorenie licenčnej zmluvy a použitie tejto práce ako školského diela podľa § 60 odst. 1 Autorského zákona.

Ďalej sa zaväzujem, že pred spísaním licenčnej zmluvy o využití diela inou osobou (subjektom) si vyžiadam písomné stanovisko univerzity o tom, že predmetná licenčná zmluva nie je v rozporu s oprávnenými záujmami univerzity.

V Brne dňa 6. decembra 2016 _____

Abstract

Mudronček, I. Mobile application for generating color schemes from photograph. Bachelor thesis. Brno: Mendel University, 2016.
The objective of this thesis was to analyze development process for Google Android platform and selection of appropriate development tools. Also, learning basics of color theory and methods for creating color harmony. Lastly, development of mobile application for android mobile devices for automatic creation of harmonious color schemes from photography.

Keywords

Color, colors, color scheme, schemes, mobile, application, Google, Android, development.

Abstrakt

Mudronček, I. Mobilná aplikácia na generovanie farebných schém z fotografií. Bakalárska práca. Brno: Mendelova univerzita v Brne, 2016.
Predmetom práce je analýza možností tvorby aplikácii pre mobilnú platformu Google Android, výber vhodného vývojového prostredia a nástrojov. Ďalej oboznámenie sa s teóriou farieb a metódami tvorby harmonických farebných schém a nakoniec implementácia aplikácie, ktorá bude tvorbu a generovanie farebných schém umožňovať.

Kľúčové slová

Farba, farby, farebná schéma, schémy, mobilná, aplikácia, vývoj, Google, Android.

Obsah

1 Úvod a cieľ práce 13

1.1 Úvod 13

1.2 Cieľ práce 13

2 Operačný systém Android 14

3 Aplikácie pre platformu Android 15

3.1 Architektúra platformy Android 15

3.1.1 Aplikačný rámec 16

3.1.2 Prostredie pre beh aplikácie 16

3.1.3 Knižnice 16

3.1.4 Linuxové jadro 16

3.2 Komponenty aplikácie pre Android 17

3.2.1 Activity 17

3.2.2 Intent 19

3.2.3 Broadcast Receiver 19

3.2.4 Services 20

3.2.5 Content Provider 21

3.3 Programovací jazyk 21

4 Vývojové prostredie 23

4.1 Eclipse 23

4.2 NetBeans 24

4.3 Android Studio 24

4.4 Emulátor Androidu 25

5 Základy aplikácie 26

6 Design aplikácie 28

6.1 Výška a tiene 28

6.2 Animácie 29

6.3 Plávajúce tlačidlo 29

6.4 Farby ... 30

7 Farebné schémy 31

7.1 Delenie farebných schém ... 32

7.1.1 Achromatická sada ... 32

7.1.2 Monochromatická sada ... 32

7.1.3 Analogická sada malých rozdielov ... 32

7.1.4 Analogická sada stredných rozdielov ... 33

7.1.5 Trojitá sada ... 33

7.1.6 Komplementárna sada ... 33

7.1.7 Rozdelená komplementárna sada ... 33

7.1.8 Dvojitá komplementárna sada ... 34

7.2 Využitie fotografie pre tvorbu farebnej schémy ... 34

8 Implementácia aplikácie 35

8.1 Zoznam farieb ... 35

8.1.1 Zoznam vytvorených schém ... 36

8.1.2 Export a zdieľanie ... 37

8.2 Ukladanie a externý prístup ... 39

8.3 Import fotografie ... 41

8.3.1 Import z galérie ... 41

8.3.2 Import z fotoaparátu ... 41

8.4 Pixelizácia ... 42

8.4.1 Priemerovanie farieb ... 42

8.5 Tvorba farebných schém ... 44

8.6 Prezentácia aplikácie ... 46

8.7 Nápoveda ... 47

8.8 Konkurencia ... 49

9 Záver 50

10 Literatúra 51

Zoznam obrázkov

Obrázok 1 Systémová architektúra **15**
Obrázok 2 Životný cyklus aplikácie **18**
Obrázok 3 Navigácia zámeru medzi aktivitami **19**
Obrázok 4 Prijímač v systéme Android **20**
Obrázok 5 Životný cyklus služby **20**
Obrázok 6 Poskytovateľ obsahu v systéme Android **21**
Obrázok 7 Vývojové prostredie Eclipse **23**
Obrázok 8 Vývojové prostredie Android Studio **24**
Obrázok 9 Emulované zariadenie Google Nexus 6 **25**
Obrázok 10 Proces kompilácie a zostavenia aplikácie pre systém Android **26**
Obrázok 11 Kruh dvanástich farieb so svetlými a tmavými odtieňmi **31**
Obrázok 12 Úvodná obrazovka so zoznamom uložených farieb (vľavo), detail konkrétnej farby so zoznamom schém (v strede) a úvodná obrazovka so zoznamom schém triedených podľa primárnej farby (vpravo) **36**
Obrázok 13 Vyexportovaná farebná schéma **38**
Obrázok 14 Pôvodný obrázok (vľavo), preskakovanie pixelov (v strede) a priemerovanie pixelov (vpravo) **43**
Obrázok 15 Postup tvorby farebných schém z fotografie **44**
Obrázok 16 Ikona aplikácie (vľavo) a štartovacia obrazovka (vpravo) **47**
Obrázok 17 Nápoveda a niektoré jej segmenty, líšiace sa dĺžkou **48**

1 Úvod a cieľ práce

1.1 Úvod

Trh s mobilnými telefónmi v posledných rokoch rapídne rastie a neustále sa vyvíja. Staré mobilné zariadenia s obmedzenými schopnosťami sú menené za nové a pokročilé zariadenia s podporou širokého spektra služieb. Mobilná aplikácia je software určený pre použitie na smartfóne, tablete a iných mobilných zariadeniach. Popularita mobilných aplikácii neustále stúpa spolu so zvyšujúcim sa počtom aktívnych používateľov a stáva sa preferovaným spôsobom interakcie, napríklad na rozdiel od internetových služieb. Dopyt verejnosti po mobilných aplikáciách a dostupnosť sofistikovaných vývojárskych nástrojov, knižníc a frameworkov spravili vývoj mobilných aplikácii ľahký, rýchly a produktívny. Napriek tomu sú na trhu s mobilnými aplikáciami stále medzery, ktoré súčasná ponuka úplne nepokrýva.

1.2 Cieľ práce

Cieľom práce je oboznámiť sa so spôsobmi tvorby mobilných aplikácii pre platformu Google Android a analyzovať vhodné nástroje a aktuálne trendy týkajúce sa designu mobilných aplikácii pre túto platformu. Ďalej sa oboznámiť s princípmi generovania farebných schém z fotografie a navrhnúť mobilnú aplikáciu, ktorá túto funkcionalitu poskytne používateľovi.

2 Operačný systém Android

Android ako platforma je open source a je navrhnutý pre beh na mobilných zariadeniach. Používaný je spoločnosťou Google a vlastní ho Open Handset Alliance. Cieľom tejto aliancie je zrýchlenie inovácii v mobilnej výpočtovej technike a poskytnutie lepšieho a zároveň lacnejšieho zážitku používateľovi a Android je cesta k dosiahnutiu týchto cieľov. Android je operačný systém založený na Linuxe, určený hlavne pre fungovanie na mobilných zariadeniach ako sú mobilné telefóny a tablety, jeho použiteľnosť ale nie je obmedzená iba na tieto typy zariadení. Práve vďaka svojej otvorenosti a možnosti prispôsobenia je používaný v rade zariadení, ako sú napríklad osobné počítače, smart televízory, fotoaparáty, hodinky, herné konzoly. Operačný systém Android nie je závislý na konkrétnom hardware a funguje na zariadeniach rozličných výrobcov, na rozdiel od systémov ako je napríklad iOS (produkty spoločnosti Apple), ktoré sú licencované a kontrolované konkrétnymi spoločnosťami. Aktuálne je Android dominantná mobilná platforma s podielom 68,54% na svetovom trhu s mobilnými zariadeniami [1].

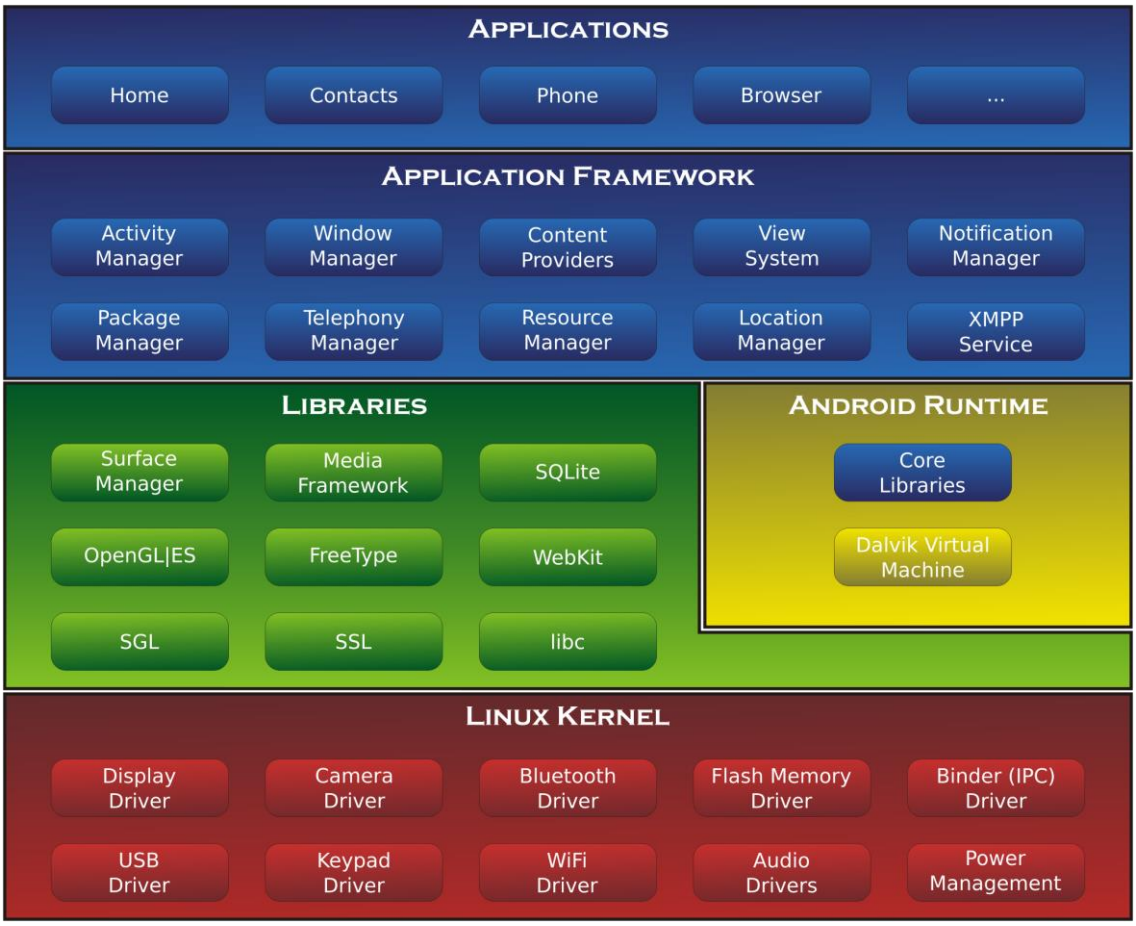
Android je plnohodnotný operačný systém a prostredie pre mobilné zariadenia. Jeho API obsahujú bohatú sadu systémových služieb v intuitívnych súboroch tried, ktoré poskytujú ľahký prístup k funkcionalitám ako sú napríklad určovanie polohy, web, telefonické služby, WiFi, multimédiá, fotoaparát. Všetky nástroje, frameworky a software potrebný pre vývoj mobilných aplikácii pre operačný systém Android sú dostupné zdarma.

3 Aplikácie pre platformu Android

Aplikácia pre Android je mobilný software vytvorený pre fungovanie na platforme Google Android. Takáto aplikácia môže byť vytvorená za použitia viacerých programovacích jazykov. Aplikácia, ktorej tvorbe sa venuje táto práca je tvorená v programovacom jazyku Java, okrem toho sa ale spolieha aj na množstvo vnútorných knižníc písaných v C++. Tvorcovia aplikácie majú možnosť použitia obrovského množstva systémových prostriedkov, nástrojov a knižníc pre použitie v aplikácii. [2]

3.1 Architektúra platformy Android

Operačný systém Android je postavený nad Linuxom. Linux je základ pre architektúru všetkých programov v Androide. Prenositeľnosť, bezpečnosť, práca so sieťami, skvelý management pamäte a procesov a podpora pre zdieľané knižnice sú niektoré z mnohých dôvodov pre použitie Linuxu ako základu pre systém Android.



Obrázok 1 Systémová architektúra

3.1.1 Aplikačný rámec

Umožňuje autorovi aplikácie pristupovať k rôznym službám, napríklad:

- Tvorba používateľského rozhrania.
- Využitie hardware zariadenia.
- Spustenie inej aplikácie na pozadí.

3.1.2 Prostredie pre beh aplikácie

Táto vrstva slúži pre beh aplikácii písaných v programovacom jazyku Java. Výsledný byte kód po konverzii na Dalvik Executable je spustený na virtuálnom stroji Dalvik, ktorý je navrhnutý tak, aby každá aplikácia bol samostatný proces s vlastnou iteráciou virtuálneho stroja. Vrstva okrem toho obsahuje aj knižnice programovacieho jazyka Java a knižnice pre zariadenia so systémom Android.

3.1.3 Knižnice

Natívne knižnice sú zdrojové kódy prenesené z iných open source projektov. V základe ide o kód v jazyku C a C++ potrebný pre zostavenie aplikácie pre systém Android. Existuje množstvo knižníc dostupných v balíkoch. V závislosti od typu zariadenia, na ktorom beží systém Android, môžu byť natívne knižnice odobrané alebo upravené, ak je to potrebné. Medzi dôležité natívne knižnice patria nasledujúce:

- Multimediálne kodeky umožňujúce prehrávanie multimediálnych súborov v rôznych formátoch.
- Databázový systém SQLite, ktorý poskytuje správu relačnej databázy.
- Jadro internetového prehliadača WebKit pre rýchle zobrazenie HTML stránok.
- OpenGL API pre zobrazenie 2D a 3D grafiky.

3.1.4 Linuxové jadro

Na najnižšej vrstve architektúry sa nachádza jadro operačného systému zabezpečujúce komunikáciu medzi hardwarom a softwarom. Základ tejto komunikácie tvoria ovládače. Okrem toho poskytuje ďalšie základné služby ako napríklad spravovanie pamäti, procesov a napájania.

3.2 Komponenty aplikácie pre Android

Tvorca aplikácie používa určité základné komponenty pre vytvorenie aplikácie pre platformu Android. Tieto komponenty pomáhajú rozdeliť tvorbu na menšie prvky, na ktorých môže tvorca aplikácie pracovať nezávisle a nakoniec ich zložiť do finálneho produktu. Existuje päť komponentov nevyhnutných pre tvorbu aplikácie pre Android. Pochopenie týchto aplikačných komponentov je pre tvorcu aplikácie nevyhnutné, pretože riadia všetky hlavné akcie v aplikácii ako sú napríklad prepínanie medzi obrazovkami a aplikáciami, manipulácia s databázou, spúšťanie udalostí a preberanie notifikácii. [3]

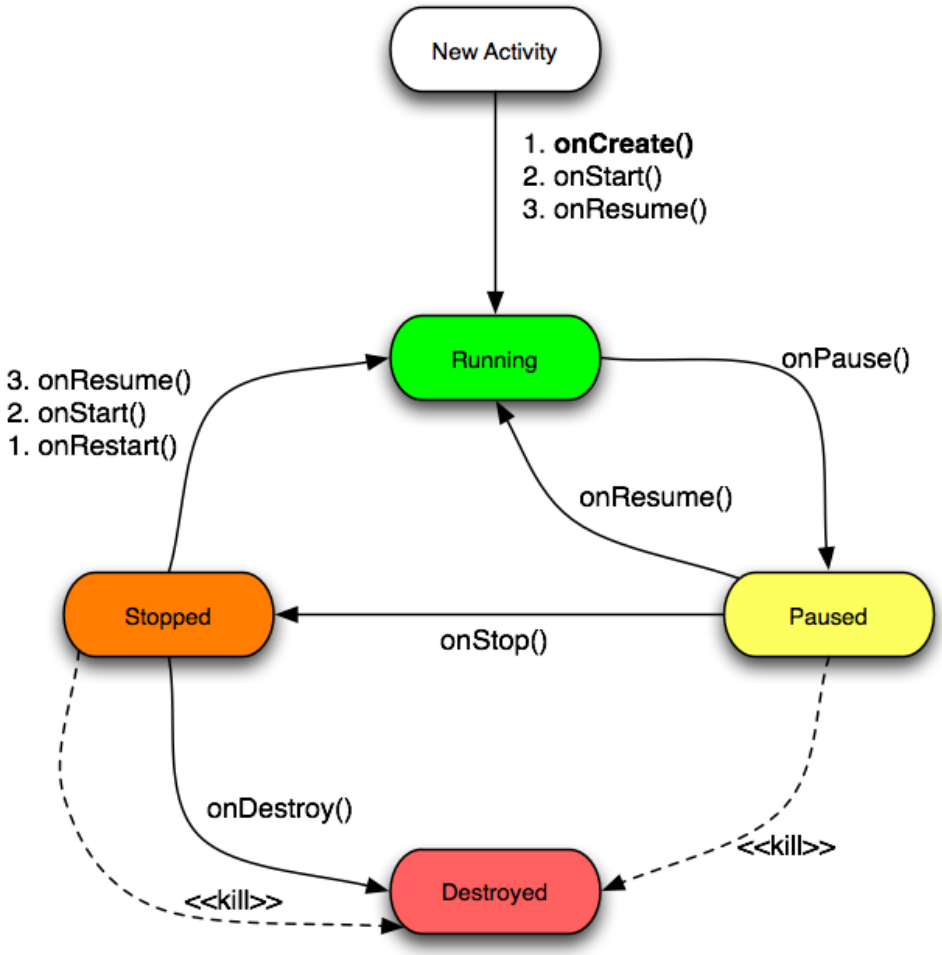
3.2.1 Activity

Aktivita je komponent aplikácie, ktorý poskytuje obrazovku, s ktorou môže používateľ aplikácie pracovať za účelom prevedenia špecifickej úlohy ako môže byť vytočenie telefónneho čísla, vytvorenie fotografie, odoslanie emailu alebo zobrazenie mapy. Jedna aplikácia môže obsahovať viacero aktivít, medzi ktorými sa používateľ aplikácie presúva. Spustenie aktivity je nevyhnutná časť procesu vývoja aplikácie pre systém Android. Aktivita je poskytnutá frameworkom Androidu, ktorý poskytuje širokú škálu funkcionalít, ako je zobrazenie používateľského prostredia, vytvorenie nového procesu a alokovanie pamäte pre prvky používateľského prostredia. Aplikácia pre operačný systém Android má štandardne jednu hlavnú aktivitu, ktorú používateľ vidí pri spustení aplikácie a je možné ju použiť pre navigáciu k ostatým aktivitám ak je to potrebné. Jedna aktivita môže spustiť a ukončiť inú aktivitu pre vykonanie rôznych akcii v rámci aplikácie. Keď používateľ spustí novú aktivitu, predchádzajúca aktivita je zastavená a jej proces je uchovaný systémom. K tejto aktivite je možné sa vrátiť stlačením na to určeného tlačidla, pokiaľ používateľ skončil svoju prácu s aktuálnou aktivitou. Android má veľmi podrobne definovaný životný cyklus aktivity a ako operačný systém spravuje procesy aktivít menením ich stavu. Pri prechádzaní medzi jednotlivými stavmi systém volá callback metódy.

- onCreate() sa volá pri prvom spustení aktivity. Inicializuje sa v nej používateľské rozhranie a vykonajú sa všetky operácie, ktoré nie sú závislé na ďalšom spôsobe použitia aktivity a ktoré je nutné vykonať iba jedenkrát. Táto metóda sa volá aj v prípade že bežala a zmenili sa jej prostriedky, využívané k jej činnosti a je nutné ich znova vytvoriť, na-

príklad pri zmene orientácie obrazovky, keď je potrebné upraviť používateľské rozhranie.

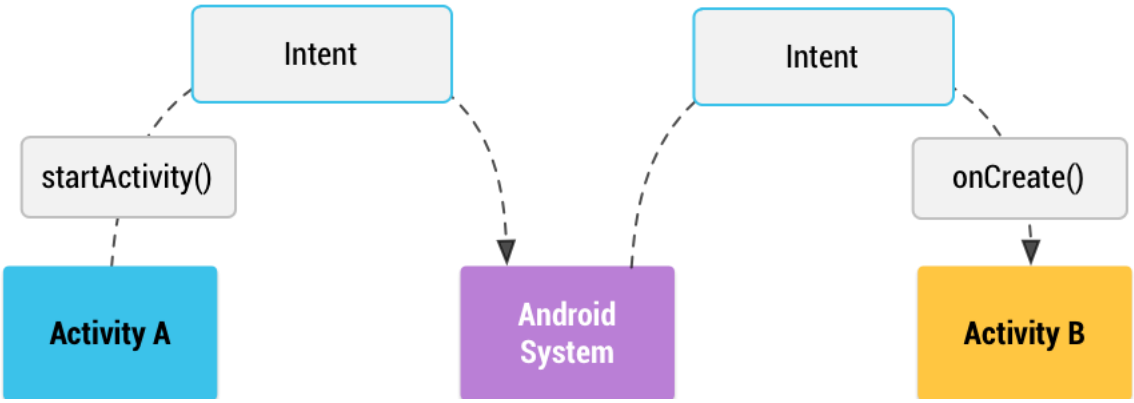
- onDestroy() sa volá v prípade že je aktivita ukončovaná. Využíva sa na uvoľnenie prostriedkov získaných pri vytváraní aktivity. Metóda sa volá vždy, keď je aktivita ukončená volaním funkcie finish(). Toto ale nemusí nastať pri násilnom ukončení, napríklad z dôvodu vyčerpania prostriedkov ako je napríklad voľná pamäť.
- onRestart() je volaná v prípade, že je aktivita zastavená a má byť spustená znovu.
- onStart() je volaná pred tým, ako sa stane aktivita viditeľnou.
- onResume() je volaná pred presunom aktivity do popredia. Je to možné využiť napríklad pre určitú zmenu v používateľskom rozhraní na základe predchádzajúcej akcie.
- onPause() sa volá v prípade, že sa do popredia dostáva iná aktivita. V tomto momente je vhodné napríklad ukončiť procesy okrem hlavného vlákna a uvolniť prostriedky vyžiadané aktivitou.
- onStop() se volá v momente, keď sa skončí viditeľnosť aktivity.



Obrázok 2 Životný cyklus aplikácie

3.2.2 Intent

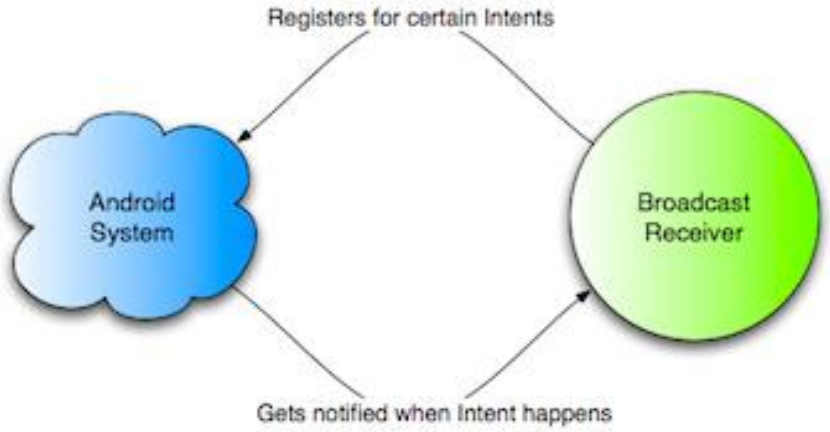
Zámer reprezentuje akciu alebo udalosť, ktorá spôsobila spustenie alebo zastavenie aktivity, spustenie alebo zastavenie služby a prenos v aplikácii. Zámery sú asynchrónne správy posielané medzi hlavnými komponentami aplikácie. Aktivita môže poslať jeden alebo viac zámerov do iných aplikácii, aby vykonali určitú úlohu, napríklad otvorili internetovú stránku alebo prehrali multimediálny súbor. Systém ale môže obsahovať viaceré aplikácie, ktoré sú schopné splniť danú úlohu. V takom prípade je používateľ vyzvaný systémom Android, aby si medzi aplikáciami vybral a nastavil aplikáciu, ktorá bude v danej situácii použitá ako primárna.



Obrázok 3 Navigácia zámeru medzi aktivitami

3.2.3 Broadcast Receiver

Prijímač je verejný mechanizmus zameraný na príjem informácie v systéme Android. Tento komponent aplikácie umožňuje používateľovi zaznamenať systémové udalosti a prijať upozornenia v prípade, že je registrovaná udalosť spustená, napríklad ide o SMS notifikáciu alebo stav batérie. Prijímač je súčasť aplikácie, ktorá sa aktivuje v prípade, že je registrovaná udalosť spustená. Systém neustále vysiela udalosti, ktoré môžu spustiť akékoľvek množstvo prijímačov. Vysielaná informácia môže byť zasielaná medzi komponentami aplikácie ale aj do úplne odlišnej aplikácie. Prijímač nemá grafickú reprezentáciu ani aktívne nebeží v pamäti.



Obrázok 4 Prijímač v systéme Android

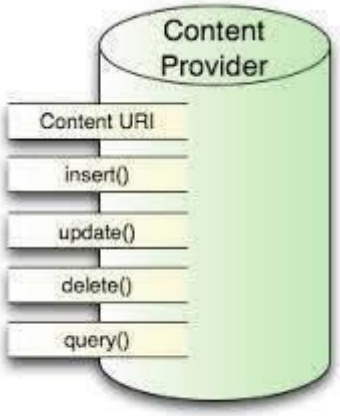
3.2.4 Services

Služby sú komponenty aplikácie, ktoré vykonávajú dlhodobo bežiace operácie na pozadí. Tieto komponenty nie sú pre používateľa viditeľné a aktualizujú zdroje dát, viditeľné aktivity a spúšťajú upozornenia. Služba je komponent aplikácie, ktorý dokáže fungovať na pozadí a v situácii, kde používateľ prepína medzi mobilnými aplikáciami. Operačný systém Android poskytuje a spracováva preddefinované systémové služby, ktoré boli deklarované v každej aplikácii pre systém Android.

Obrázok 5 Životný cyklus služby

3.2.5 Content Provider

Poskytovateľ obsahu je komponent aplikácie, ktorý je použitý na správu a zdieľanie aplikačných databáz. Viaceré aplikácie môžu zdieľať tie isté dáta rozličnými spôsobmi, závisiacimi práve na type dát. Viaceré aplikácie môžu pristupovať k tým istým dátam naraz. Poskytovateľ obsahu je preferovaný spôsob zdieľania dát medzi aplikáciami. Sám systém Android obsahuje natívny poskytovateľ obsahu, ktorý spravuje dáta ako sú audio súbory, obrázky, video a informácie o kontaktoch.



Obrázok 6 Poskytovateľ obsahu v systéme Android

3.3 Programovací jazyk

Java je všeobecne využiteľný, štruktúrovaný, multiplatformový a objektovo orientovaný programovací jazyk. Práve v tomto jazyku sú písané aplikácie pre systém Android a samotné aplikácie sú hlboko založené na základoch tohto jazyka. Java obsahuje množstvo užitočných funkcii a knižníc z jazykov ako C a C++. Dôvody prečo zvoliť jazyk Java ako natívny jazyk pre vývoj aplikácie v systéme Android sú:

- Jednoduchosť na pochopenie a na naučenie sa.
- Nezávislosť na platforme.
- Objektová orientácia.
- Kompilácia a spustenie kódu virtuálnym strojom. [4]

XML (Extensible Markup Language) je značkovací jazyk obsahujúci veľmi jednoduchý, ale flexibilný formát textu, ktorý je prispôsobený pre čítanie človekom, ale aj počítačom. Definuje súbor pravidiel pre kódovanie dokumentu a jeho použitie cez internet. XML je všeobecne rozšírený a používaný formát. Ľahko sa delí a strojovo spracováva a manipuluje. Systém Android vopred pripraví a spracuje zdrojové XML súbory do komprimovaného binárneho formátu a uloží ho v zariadení. Väčšina rozložení používateľského rozhrania a elementov na obrazovke je deklarovaná v XML súboroch. [5]

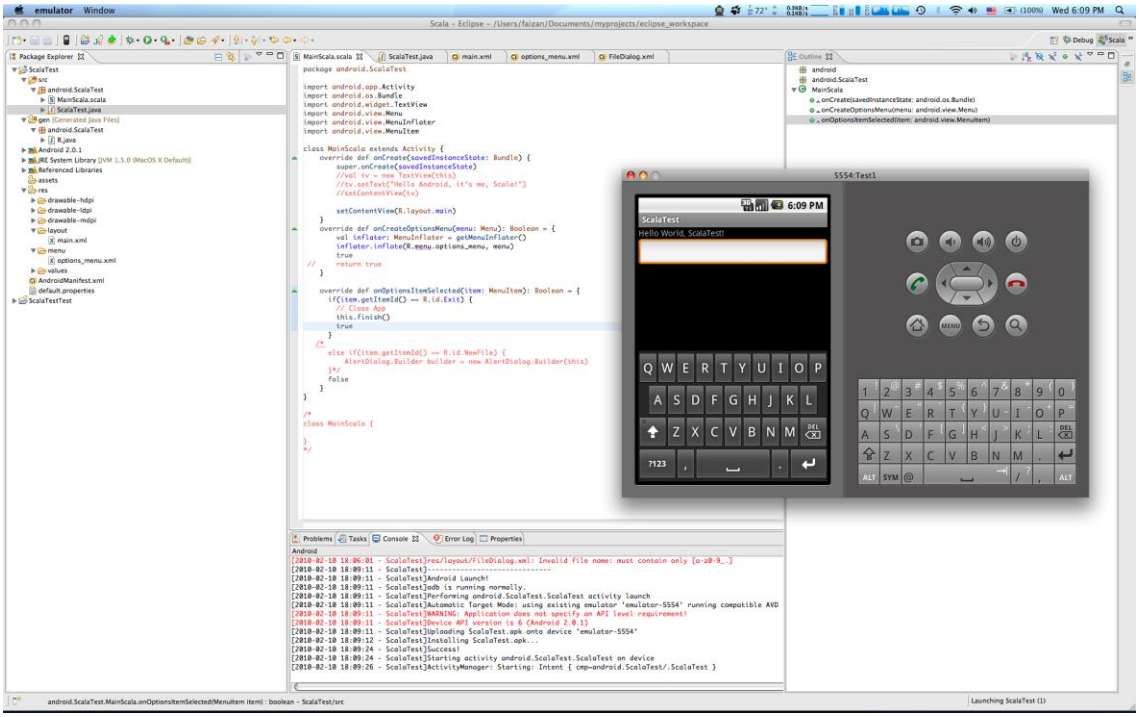
JSON (JavaScript Object Notation) je prenosný textový formát založený na syntaxi JavaScriptu pre popis objektov, zároveň ale nie je závislý na konkrétnej platforme. Knižnice pre spracovanie JSON súborov sú dostupné pre množstvo rôznych programovacích jazykov. Je ľahký na čítanie a zápis pre autora aplikácie a zároveň pre zariadenia so systémom Android ľahký na generovanie a spracovanie. JSON je odvodený zo skriptovacieho jazyka JavaScript aby reprezentoval jednoduchú dátovú štruktúru a asociatívne polia, ktoré sú často adresované ako JSON objekt. [6]

4 Vývojové prostredie

Nevyhnutným krokom pri tvorbe aplikácie je výber vhodného vývojového prostredia (IDE). Dôležitosť tohto kroku spočíva v uľahčení práce a zvýšení produktivity pri správnom výbere prostredia.

4.1 Eclipse

Eclipse IDE sa používa predovšetkým pre programovací jazyk Java. Spoločnosť Google ho dlhodobo podporovala ako jediné prostredie pre vývoj aplikácií pre platformu Android. Vývoj aplikácií pre Android v Eclipse vyžaduje doinštalovanie pluginu Android Development Tools (ADT), ktorý poskytuje sadu nástrojov a ich integráciu do Eclipse, čo možní prístup k množstvu funkcií. Okrem toho obsahuje grafické používateľské rozhranie, obsahujúce prístup k SDK nástrojom príkazového riadku a nástroje pre návrh a budovanie používateľského rozhrania aplikácie. Výhoda tohto vývojového prostredia oproti prostrediu Android Studio je podpora NDK. [7]



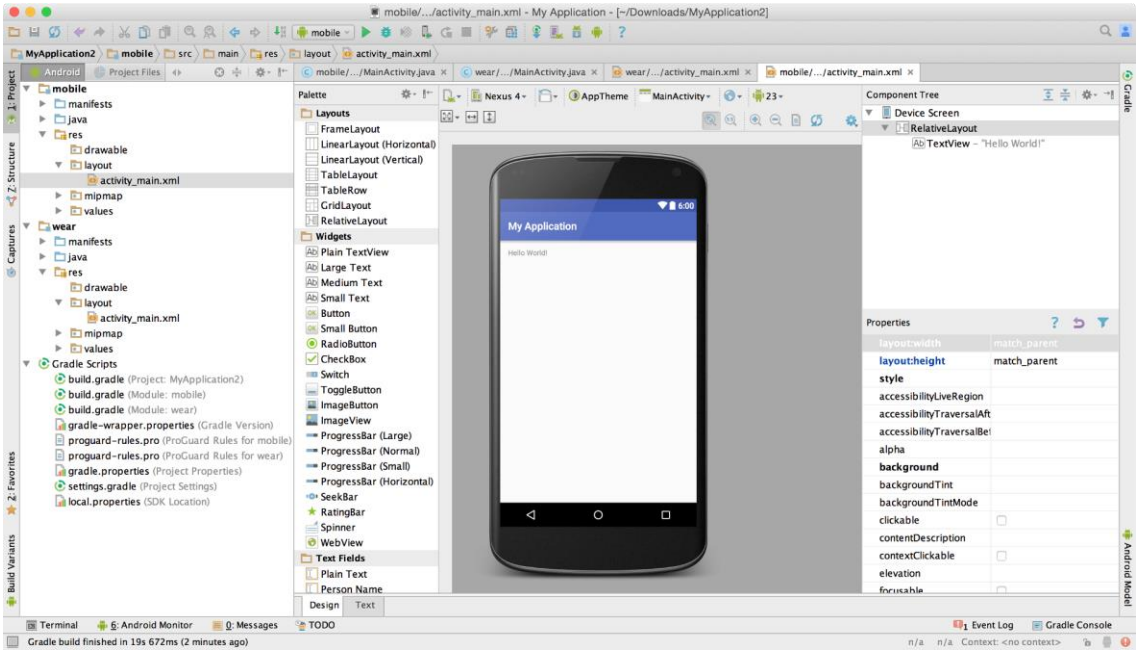
Obrázok 7 Vývojové prostredie Eclipse

4.2 NetBeans

Vývojové prostredie NetBeans je, rovnako ako Eclipse IDE, používané hlavne na vývoj v jazyku Java, v ktorom je aj vytvorené. Okrem toho podporuje množstvo ďalších programovacích jazykov ako sú napríklad C++ alebo PHP. Vývojové prostredie NetBeans vyžaduje, rovnako ako prostredie Eclipse, plugin pre vývoj na platformu Android a to konkrétne plugin NBAndroid, ktorý je však neoficiálny a tak mu chýba podpora spoločnosťou Google.

4.3 Android Studio

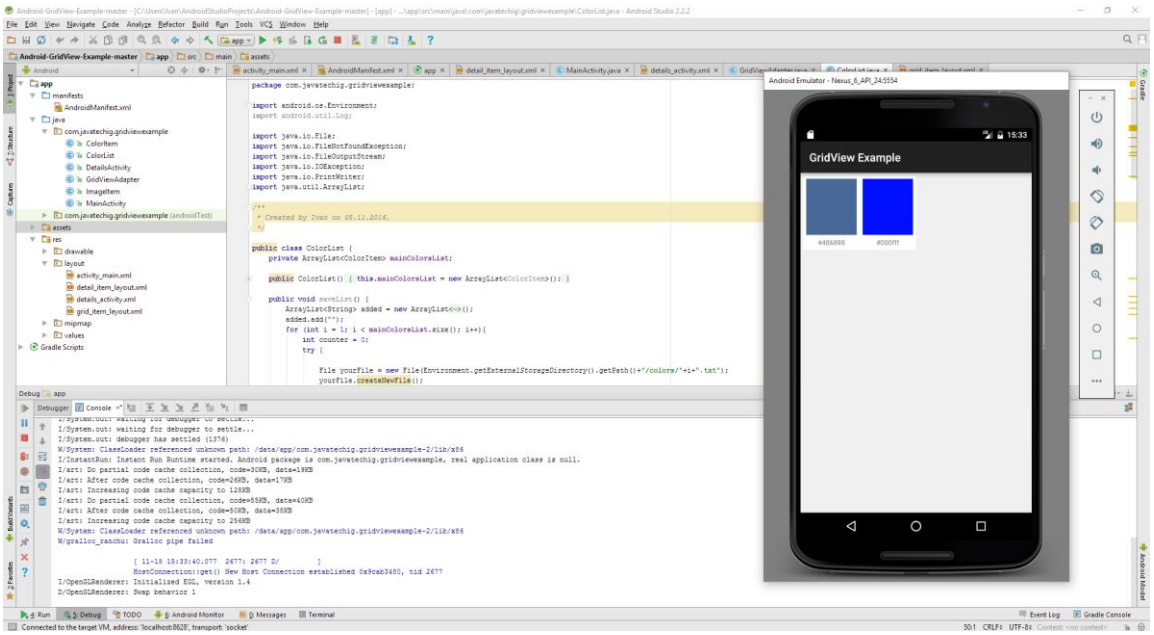
Android Studio je vývojové prostredie, ktoré je od roku 2014 podporované ako primárne vývojové prostredie pre platformu Android spoločnosťou Google. Práve jeho predchodcom bolo vyššie spomínané prostredie Eclipse. Vývoj pluginu ADT bol ale už ukončený. Android Studio IDE je založené na IDE pre programovanie v jazykoch ako Java, Groovy IntelliJ IDEA. Inštalácia tohto vývojového prostredia je veľmi jednoduchá, inštalačný balík obsahuje a stiahne všetky potrebné súčasti. Okrem toho je k inštalácii potrebná Java spoločnosti Oracle. Vývoj aplikácie prebiehal práve v tomto prostredí a to z dôvodu jeho oficiálnej podpory spoločnosťou Google a množstva integrovaných funkcií. [8]



Obrázok 8 Vývojové prostredie Android Studio

4.4 Emulátor Androidu

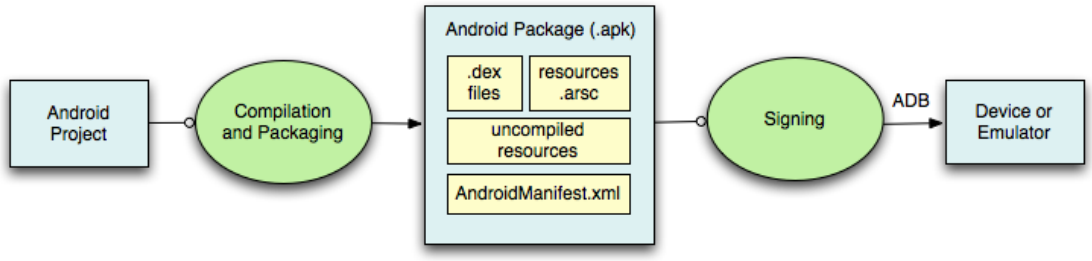
Počas procesu implementácie a ladenia aplikácie je nutné, aby bola aplikácia spustená na zariadení so systémom Android. Vývojár môže využiť zariadenie pripojené cez USB s aktivovaným vývojárskym režimom alebo emulované prostredie Androidu. Emulátor Androidu je virtuálne zariadenie so systémom Android spustené na počítači. Emulátor predstiera všetky softwarové a hardwarové funkcie typického mobilného zariadenia okrem schopnosti uskutočňovať skutočné telefonické hovory. Emulátor Androidu umožňuje tvorcovi aplikácie testovanie fungovania behu aplikácie na rôznych úrovniach API bez potreby použitia skutočných fyzických zariadení. Android Virtual Device (AVD) je konfigurácia zariadenia, ktorá je spustená v emulátore a poskytuje virtuálne prostredie špecifického zariadenia umožňujúce inštaláciu a spustenie aplikácií. Je možné nastaviť množstvo parametrov, medzi ktoré patrí napríklad rozlíšenie obrazovky, veľkosť pamäte alebo existencia fotoaparátu, ale aj softwarové parametre ako je úroveň API. Emulátor systému Android je pre vývoj nevyhnutný, nakoľko by použitie potrebného množstva fyzických zariadení s rozličnými konfiguráciami na ladenie bolo značne nepraktické.



Obrázok 9 Emulované zariadenie Google Nexus 6

5 Základy aplikácie

Aplikácia pre systém Android je skompilovaná a zabalená do jediného súboru, ktorý obsahuje kód aplikácie a zdroje. Každá časť Java kódu, písaného pre aplikáciu v systéme Android, je označená ako Dalvik executable code a všetko ostatné sú zdroje, ako napríklad ikony, animácie alebo XML súbory. Dalvik je implementácia Java Virtual Machine na zariadeniach so systémom Android a z toho dôvodu je jeho neoddeliteľnou súčasťou. Aplikácia je bežne písaná v jazyku Java a kompilovaná do Java byte code, ktorý je pred inštaláciou do zariadenia konvertovaný z .class súborov Java Virtual Machine na .dex (Dalvik Executable) súbory. Formát Dalvik Executable je kompaktný a navrhnutý pre zariadenia s obmedzeniami vo veľkosti pamäte a rýchlosti procesoru. Android SDK kompiluje a balí aplikácie pre systém Android do .apk súborov. Skompilovaný balík obsahuje všetky súbory potrebné pre fungovanie aplikácie na zariadení alebo v emulátore, ako sú skompilované .dex súbory, binárna verzia AndroidManifest.xml súboru a ostatné zdrojové súbory aplikácie.



Obrázok 10 Proces kompilácie a zostavenia aplikácie pre systém Android

Okrem zdrojového kódu v jazyku Java musí aplikácia pre systém Android obsahovať ďalšie nevyhnutné súbory. Treba z nich spomenúť dva, a to R.java a AndroidManifest.xml.

AndroidManifest.xml je základ každej aplikácie pre systém Android a nachádza sa v koreňovom adresári aplikácie. Na základe tohto súboru systém vie, čo a ako volať a ako môže komponent spolupracovať s okolím. Sú v ňom deklarované všetky aktivity, služby a povolenia. AndroidManifest.xml predstavuje esenciálne informácie o aplikácii, ktoré musí systém Android poznať pred spustením akéhokoľvek kódu aplikácie.

R.java je automaticky generovaný súbor, ktorý spája kód v jazyku Java so zdrojovými súbormi. Súbor R.java obsahuje všetky identifikátory (ID) zdrojov priradených k zdrojom, ako sú rozloženia alebo ikony. Toto priradenie zabezpečuje autor aplikácie alebo Android SDK.

Príklad XML súboru AndroidManifest.xml, ktorý je súčasťou vyvíjanej aplikácie a obsahuje povolenie pre zápis a čítanie externej pamäte.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javatechig.gridviewexample">

    <uses-permission andro-
id:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/splashScreenTheme"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category andro-
id:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".DetailsActivity"
            />
        <activity
            android:name=".AllColorsGrid"
            />
        <activity
            android:name=".UserColorSelector"
            ></activity>
    </application>

</manifest>
```

6 Design aplikácie

Material Design je takzvaný design language. Ide o špecifikáciu vizuálnych parametrov prvkov, ktoré majú byť pre danú aplikáciu a platformu typické. V tomto prípade ide o jazyk vyvinutý spoločnosťou Google. Predstavuje špecifikáciu ako navrhnúť a implementovať grafické používateľské rozhranie na mobilných zariadeniach so systémom Android. Použiť ho je ale možné aj na iných typoch zariadení ako sú napríklad televízory, hodinky alebo internetové stránky. V súčasnosti bolo množstvo aplikácií prepracovaných do tohto designu. Medzi prvé patrili aplikácie spoločnosti Google, a to emailový klient Gmail, YouTube a ďalšie. Vývoj Material Designu mal za cieľ vytvorenie vizuálneho jazyka spájajúceho moderný design a dostupné technologické možnosti. Je to univerzálna grafická špecifikácia nezávislá na platforme, spôsobe ovládania zariadenia alebo jeho veľkosti. Material Design bol inšpirovaný materiálmi ako papier a atrament, ktoré tvorili základný podklad pre vývoj tejto špecifikácie. Materiály sa v grafickom používateľskom rozhraní aplikácie chovajú rovnako ako v skutočnosti a zachovávajú si fyzikálne zákony. Elementy používateľského rozhrania by mali tvoriť hierarchiu a ich postupnosť by mala dávať zmysel. Aplikácia by mala mať kompaktný vzhľad so zjednoteným pôsobením. Z toho dôvodu sa odporúča používať iba obmedzenú farebnú paletu, množstvo textových fontov a ich veľkosť. Dôležitá je jasnosť a sýtosť farieb a jednotný štýl naprieč všetky obrazovky aplikácie. [9]

6.1 Výška a tiene

Do príchodu Material Designu sa prostredie systému Android interpretovalo iba v dvoch rozmeroch. Napriek pridaniu tretieho rozmeru sa však z komponentov nestávajú priestorové objekty. Komponenty grafického používateľského rozhrania je možné umiestňovať do rôznej výšky v závislosti od ich dôležitosti. Samotná výška každého komponentu je ale 1dp a mala by byť nemenná. Jednotlivé komponenty sú umiestnené v rozdielnych výškach. V závislosti na tejto výške vrhajú tieň na nižšie položené objekty a svoj podklad. Čím vyššie sa objekt nachádza, tým väčší a zároveň rozmazanejší tieň vrhá. Toto navádza priestorový dojem vrstvenia papiera. Objekty majú ďalej možnosť meniť svoj tvar, veľkosť, rotovať a pohybovať sa. Jeden objekt však nemôže zdieľať priestor s iným objektom a teda prekrývať sa v rovnakej výške ani pri pohybe. Zmena výšky objektu je štandardne následok interakcie používateľa aplikácie. Ide napríklad o posun tlačidla pri jeho stlačení, čo poskytuje

vizuálnu odozvu o úspešnom vykonaní akcie. Vďaka tomu používateľ lepšie rozozná skutočnosť, že sa mu podarilo tlačidlo úspešne stlačiť a aplikácia pracuje na adekvátnej akcii. [10]

6.2 Animácie

Používateľ aplikácie by si mal byť v každom momente vedomý, kde v aplikácii sa momentálne nachádza. Žiadna používateľova interakcia s aplikáciou by nemala spôsobiť jeho zmätenie. K tomuto účelu slúži široká paleta animácií. Väčšina z ich je ale dostupná iba vo verzii API 21 a novších. Zmysel animácií je vo vytvorení plynulého prechodu medzi obrazovkami aplikácie. Napríklad vytvoriť previazanie so stiskom tlačidla postupným rozšírením novej obrazovky práve z polohy tlačidla. Počas prechodu medzi obrazovkami aplikácie je treba rátať s tromi druhmi objektov:

- Objekty miznúce z pôvodnej obrazovky.
- Objekty objavujúce sa na novej obrazovke.
- Objekty zdieľané oboma obrazovkami.

Skrytie a zobrazenie objektov je možno manuálne nastaviť pre každý objekt samostatne. Primárne sa treba ale zamerať na prehľadnosť. Animácia by mala mať zmysel a dôvod svojej existencie. Prvky by sa mali pohybovať naraz a synchronizovane, najlepšie v rovnakom smere. Tento pohyb by mal upútať pozornosť používateľa a naviesť ho k dôležitým údajom na obrazovke. Objekty, ktoré sú zdieľané, je možno využiť práve pri prechode zo zoznamu položiek na konkrétne vybranú položku a jej detail. Je možné vytvoriť prepojenie medzi prvkami, ktoré sa nachádzajú na oboch obrazovkách. Takýmto spôsobom sa dá používateľovi uľahčiť prehlaď o navigácii a dianí v rámci aplikácie. [11]

6.3 Plávajúce tlačidlo

Ide o prvok, ktorý v aplikácii s Material Designom zaujme na prvý pohľad. Je to kruhové tlačidlo s plnou farebnou výplňou a ikonou vo svojom strede, predstavujúcou jeho funkciu. Štandardne je umiestnené v pravom dolnom rohu obrazovky aplikácie. Pri posune je tlačidlo statické a jeho poloha sa nemení. Podľa pravidiel by aktivita mala obsahovať iba jedno takéto tlačidlo a to pre aktiváciu najpoužívanejšej akcie na danej obrazovke ako je napríklad vytvorenie

novej správy v aplikácii Gmail. Okrem toho nesmie mať akcia vyvolaná týmto tlačidlom deštruktívny charakter ako je napríklad odstraňovanie. Umiestnenie tlačidla v pravom dolnom rohu zároveň umožňuje jednoduchý prístup a ovládanie na mobilných zariadeniach. [12]

6.4 Farby

Farby v Material Designe by mali spĺňať jednotný dojem aplikácie. Vhodné je preto používať iba obmedzené množstvo farieb a vizuálne zhodné prvky naprieč aplikáciou ako je rovnaký tvar, veľkosť a farba všetkých tlačidiel, ohraničení a písem. Okrem bielej, čiernej a odtieňov šedej farby, ktoré sa používajú na text a pozadie, sa štandardne používajú iba dve ďalšie farby. Primárna farba je udaná v troch odtieňoch. Ide o hlavný odtieň viditeľný napríklad na hlavnom paneli aplikácie. Tento odtieň je v aplikácii dominantný. Ďalšie dva odtiene sú svetlejší a tmavší variant primárnej farby. Druhá farba je doplnková a je dobré ju použiť na zvýraznenie dôležitých prvkov, ktoré majú upútať pozornosť používateľa aplikácie. Používa sa napríklad na plávajúce tlačidlo. Mala by byť dostatočne kontrastná voči primárnej farbe a z tohto dôvodu nie je vhodné použiť iný odtieň primárnej farby. Nie je vhodné aby sekundárna farba bola použitá príliš, pretože by odvádzala pozornosť od dôležitých prvkov. Všetky farby by mali byť dostatočne jasné a sýte. V oficiálnej dokumentácii poskytovanej spoločnosťou Google sa nachádzajú konkrétne farby vhodné na použitie, čo autorovi aplikácie uľahčuje výber. [13]

7 Farebné schémy

Analýzou spôsobov usporiadania farieb môžeme zistiť, že prevažuje usporiadanie do kruhu. Ide o užitočnú pomôcku pri výbere kombinácií farieb. Kruh farieb znázorňuje vzťahy medzi farebnými tónmi, či už ide o podobnosť alebo odlišnosť. Preto ide o užitočný nástroj, vhodný pre zostavenie farebných schém. Základom farebného kruhu tvoreného dvanástimi farbami sú tri primárne farby, a to žltá, modrá a červená. Ide o farebné tóny, ktoré nemožno získať miešaním iných farieb. Naopak všetky ostatné farebné tóny vznikajú miešaním týchto primárnych farieb. Sekundárne farby vnikajú kombináciou dvojice farieb primárnych. Ide o oranžovú, zelenú a fialovú. Bývajú tiež označované ako farby komplementárne, nakoľko tvoria protiklad farieb primárnych. Nachádzajú sa na kruhu farieb oproti sebe a ako bolo spomenuté v predchádzajúcich kapitolách, je vhodné takúto farebnú kombináciu použiť v Material Designe ako farby používateľského rozhrania aplikácie. Medzi primárnou a sekundárnou farbou sa na kruhu farieb nachádza farba terciárna, ktorá vniká práve miešaním farby primárnej a sekundárnej. Ďalším miešaním farieb sa postupne dostávame ku všetkým farbám farebného spektra.

Obrázok 11 Kruh dvanástich farieb so svetlými a tmavými odtieňmi

7.1 Delenie farebných schém

Farebné schémy je možné deliť podľa určitých kritérií, ako je počet základných farebných tónov alebo podľa vzdialenosti a príbuznosti farebných tónov v kruhu farieb. [14]

7.1.1 Achromatická sada

Sada farieb kombinujúca čiernu, bielu a odtiene šedej, ktorá pôsobí neutrálne, harmonicky a seriózne. Výhoda tejto sady spočíva v možnosti byť doplnená akoukoľvek pestrou farbou spektra bez negatívneho vplyvu na jej súlad. V opačnom prípade môže táto sada pôsobiť nevýrazne a až nudne. Achromatickú sadu s farebnou dominantou je možné interpretovať ako kombináciu achromatickej a monochromatickej sady.

7.1.2 Monochromatická sada

Je založená iba na jednom farebnom tóne a jeho kombinácii s vlastnými farebnými odtieňmi. Tieto odtiene sú dosiahnuté prechodom zvoleného tónu k farbe bielej alebo čiernej. Farby v sade majú medzi sebou úzky vzťah s jasnou súvislosťou. Celá sada v tom prípade pôsobí súvisle a harmonicky a jej prirodzenosť vychádza zo skutočného sveta a pôsobenia svetla a tieňov na farby. Môže nastať nevýhoda spojená s malou rozmanitosťou tónov oproti iným sadám. Je jednoduché ju vytvoriť, ale nedosahuje výrazný farebný kontrast pre zvýraznenie prvkov alebo upútanie pozornosti.

7.1.3 Analogická sada malých rozdielov

Pozostáva z málo rozdielnych farebných tónov farieb susediacich vo farebnom kruhu. Je podobná s monochromatickou zostavou, ale poskytuje väčší rozsah farebných tónov. Táto sada pôsobí upokojujúco a nenáročne na ľudský zrak, nakoľko je tvorená svetlom s podobnou vlnovou dĺžkou. Združuje farby, ktoré sa málo odlišujú svojou svetlosťou a farebnosťou. Oproti kontrastnejším sadám vyvoláva menšie napätie, chýba jej však rozdielnosť farebných tónov. V analogickej sade je možné kombinovať aj čisté farebné tóny s odtieňmi. Jedna sýta farba pôsobí ako dominantná a sú k nej pridružené farby v svetlejších alebo tmavších odtieňoch. Je však vhodné sa vyhnúť analogickej zostave kombinujúcej susediace teplé a studené farebné tóny.

7.1.4 Analogická sada stredných rozdielov

Tieto farby spolu v kruhu farieb priamo nesusedia. Napriek tomu je možné s nimi vytvoriť farebnú harmóniu. Výhodou je výraznejší farebný a svetelný kontrast aj keď sa stráca jednostrannosť farebnej nálady. Z dôvodu malej blízkosti farebných odtieňov je vhodné pridať prirodzený prechod farieb, ktorý má vzťah k obom predchádzajúcim farbám. Tento medzistupeň ich spája a zároveň je s oboma farbami v súlade.

7.1.5 Trojitá sada

Ide o tri farby, ktoré sú umiestnené v tretinách farebného kruhu. Táto trojica farieb sa nachádza na vrcholoch rovnostranného trojuholníka. Všetky farby zo zostavy sa kombinujú v zhodnej čistote a sýtosti, čo vyvoláva dojem celkovej živosti. Sada je plná energie a poskytuje možnosti pre tvorbu kontrastu a farebných dominánt. Trojitú sadu je možné zostrojiť z primárnych, sekundárnych a aj terciárnych farieb.

7.1.6 Komplementárna sada

Kombináciou dvoch protiľahlých tónov dosahujeme najväčší farebný kontrast a najväčšie napätie. Hovoríme o komplementárnych farbách a ich kombinácia tvorí komplementárnu sadu. Pre túto sadu je charakteristický veľký tepelný kontrast, nakoľko sada vždy spája teplý a studený farebný tón. Výhodou môže byť výrazná rozmanitosť farebných tónov pôsobiacich živo. Výberom dvoch sýtych farieb do zostavy v rovnakom pomere môže spôsobiť prílišnú tvrdosť kontrastu. Táto situácia sa však dá zmierniť, napríklad použitím jednej farby ako dominantnej a druhej ako akcentu.

7.1.7 Rozdelená komplementárna sada

Je tvorená tromi farbami, a to základným farebným tónom a farbami susediacimi s farbou komplementárnou k základnému tónu. Poskytuje dostatočný kontrast, ktorý však nie je tak ostrý, ako pri komplementárnej sade a pôsobí preto na ľudské oko príjemnejšie. Poskytuje okrem toho väčší priestor na vyváženie teplých a studených farieb.

7.1.8 Dvojitá komplementárna sada

Skladá sa z dvoch základných farieb a dvoch ich komplementov. Kombinuje teda štyri farby, ktoré sú väčšinou rozmiestnené v štvrtinách kruhu farieb. V prípade veľkého rozstupu však môže vyvolávať až príliš agresívny dojem na pozorovateľa. Menší rozstup medzi farbami zmierňuje napätie. V takom prípade je vhodné zvoliť dve susediace farby a ich susediace komplementy.

7.2 Využitie fotografie pre tvorbu farebnej schémy

Farebná schéma sa dá vytvoriť aj inak ako na základe pravidiel kruhu farieb. Zásady pre prácu s farbami vychádzajú práve z prírodných zákonitostí. Z toho dôvodu je možné získať farebnú schému z digitálnej fotografie. V prípade potreby získania schémy so živými a teplými farbami je vhodné fotografovať napríklad jesennú krajinu, západ slnka alebo kvitnúce rastliny. Problém nastáva v momente, keď je treba z fotografie, obsahujúcej milióny farieb, získať práve minimum farieb, ktoré sú žiaduce.

Jedným zo spôsobov, ako z fotografie získať harmonickú farebnú schému je pixelizácia. Ide o proces, v ktorom sa zo zvolenej fotografie vytvorí mozaika, kde každá jej súčasť obsahuje iba pixely rovnakej farby. Výhody tohto procesu spočívajú v odstránení šumu spôsobeného málo kvalitným fotoaparátom zariadenia alebo nevyhovujúcimi svetelnými podmienkami, ktorý by mohol mať nepriaznivý vplyv na výslednú farebnú schému. Okrem toho nastane určité spriemerovanie množstva mierne odlišných odtieňov farieb vo fotografii. V konečnom dôsledku sa dosiahne obrázok, ktorý umožní jednoduchšie a presnejšie vytvorenie harmonickej farebnej schémy a zároveň umožní jednoduchšiu navigáciu používateľovi aplikácie pri výbere konkrétnej farby z fotografie.

8 Implementácia aplikácie

Na vývoj aplikácie bolo použité vývojové prostredie Android Studio, ktoré je v súčasnosti podporované ako primárne vývojové prostredie pre platformu Android a to z dôvodu jeho oficiálne podpory spoločnosťou Google, množstvu integrovaných funkcií a jednoduchosti použitia.

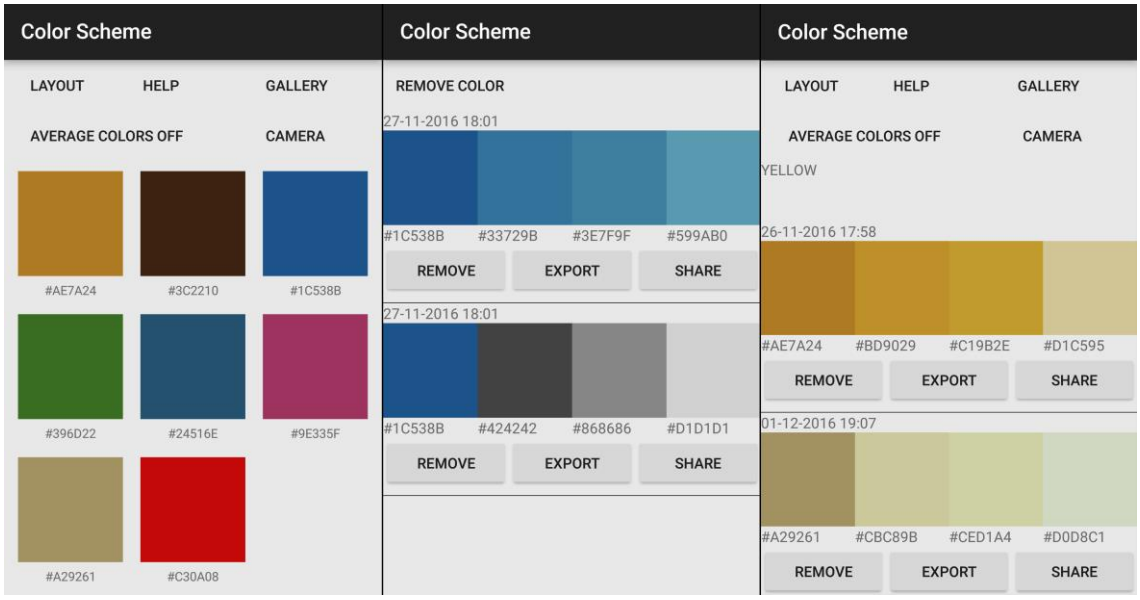
8.1 Zoznam farieb

Prvá obrazovka, ktorú používateľ aplikácie vidí po spustení, je zoznam všetkých uložených farieb, ktoré boli použité na vygenerovanie farebných schém. Je možné meniť spôsob zobrazenia úvodnej obrazovky tlačidlom (LAYOUT). V prvom prípade sú farby zobrazené v mriežke, každé políčko zobrazuje danú farbu a pod ňou jej hexadecimálne vyjadrenie. Každá z týchto položiek sa dá zvoliť a používateľ je presunutý na novú obrazovku s vytvorenými farebnými schémami danej farby. Na úvodnej obrazovke sa nachádza tlačidlo umožňujúce pridať novú farbu z fotografie (CAMERA), tlačidlo pre pridanie novej farby z galérie (GALLERY), tlačidlo obsahujúce nápovedu (HELP) a prepínač funkcie priemerovania farieb (AVERAGE COLORS ON/OFF). Druhý spôsob zobrazenia obsahuje všetky spomínané ovládacie prvky, ale zobrazuje všetky farebné schémy triedené podľa primárnej farby a rozdelené do skupín.

Z dôvodu existencie veľkého množstva farieb na hlavnej obrazovke aplikácie nastáva rozpor s pravidlami Material Designu, zameranými na použitie farieb. V tejto situácii by sa plávajúce tlačidlo s kontrastnou sekundárnou farbou stratilo a nespĺňalo by svoju funkciu upútania pozornosti k najdôležitejšej funkcii danej obrazovky, konkrétne pridanie farby z fotografie. Práve opačný prístup, a to použitie neutrálnej farby a voľného priestoru, lepšie upúta pozornosť používateľa aplikácie v tejto situácii a umožní jednoduchšiu navigáciu. Tak isto je vhodné použiť neutrálnu farbu v záhlaví aplikácie, konkrétne tmavý odtieň šedej, aby nenastal konflikt so zoznamom uložených farieb.

8.1.1 Zoznam vytvorených schém

Na túto obrazovku je používateľ aplikácie presmerovaný pri zvolení konkrétnej farby na úvodnej obrazovke. Zobrazuje všetky vytvorené farebné schémy k danej farbe v zozname pod sebou. Každá položka obsahuje dátum a čas vytvorenia farebnej schémy, jednotlivé farby v rade za sebou a pod každou farbou jej hexadecimálnu reprezentáciu. Okrem toho sa na tejto obrazovke nachádza tlačidlo pre odstránenie danej farby a tlačidlá umožňujúce export ako portable network graphics a zdieľanie pre každú farebnú schému.



Obrázok 12 Úvodná obrazovka so zoznamom uložených farieb (vľavo), detail konkrétnej farby so zoznamom schém (v strede) a úvodná obrazovka so zoznamom schém triedených podľa primárnej farby (vpravo)

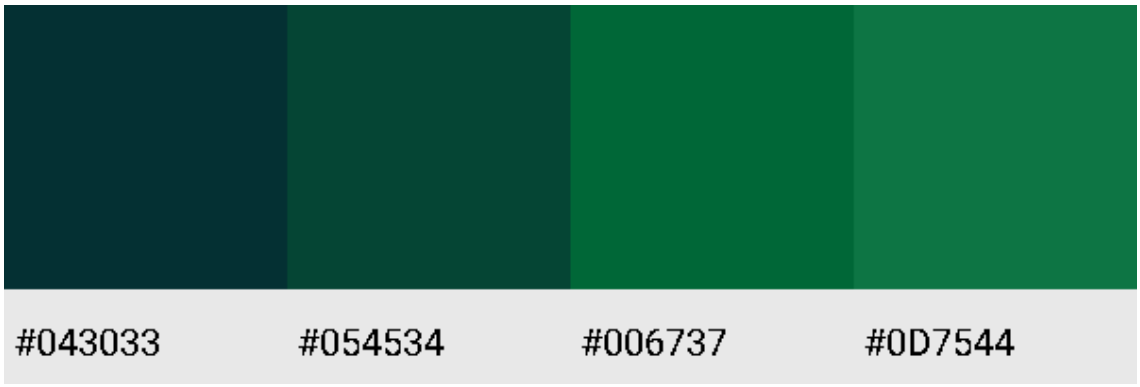
8.1.2 Export a zdieľanie

Pri stlačení tlačidla na exportovanie (EXPORT) sa daná farebná schéma prevedie na obrázok vo formáte PNG a uloží sa do priečinku v externej pamäti zariadenia. Toto umiestnenie je poskytované systémom Android a nachádza sa v časti pamäte, ku ktorej je umožnený prístup z počítača a nezáleží, či samotné mobilné zariadenie so systémom Android umožňuje použitie externých pamäťových kariet. Názov súboru ja tvorený hexadecimálnym vyjadrením primárnej farby schémy a dátumom a časom exportu. Exportovaný obrázok má šírku 600 pixelov a výšku 200 pixelov. Obsahuje farby danej schémy a pod nimi ich hexadecimálne vyjadrenie.

```
public void export(Bitmap bitmap, String name){
    File dir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTO
RY_PICTURES) + "/exported colors/");
    if (!dir.exists()){dir.mkdir();}
    Calendar c = Calendar.getInstance();
    SimpleDateFormat df = new SimpleDateFormat("dd-MM-yyyy_HH:mm:ss");
    String formattedDate = df.format(c.getTime());
    File dest = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTO
RY_PICTURES) + "/exported colors/", name + "_" + formattedDate +
".PNG");

    try {
        FileOutputStream out = new FileOutputStream(dest);
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
        out.flush();
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

V prípade využitia možnosti zdieľania (SHARE) sa vytvorí rovnaký obrázok, ale iba ako dočasný súbor, ktorý je vymazaný vždy pri ďalšom využití funkcie zdieľania. Systém Android umožňuje predať dáta inej aplikácii, v tomto prípade obrázok, a v kóde je zabezpečené, aby sa pri každom zdieľaní zobrazila voľba aplikácie, ktorá bude použitá alebo aby sa zobrazila upozorňujúca hláška, v prípade absencie akejkoľvek kompatibilnej aplikácie.

```
public void share (Bitmap bitmap){
    try {
        File dir = new
File(Environment.getExternalStorageDirectory().getPath()+"/ColTmp/");
        dir.mkdir();
        if (dir.isDirectory())
        {
            String[] children = dir.list();
            for (int i = 0; i < children.length; i++)
            {
                new File(dir, children[i]).delete();
            }
        }

        File file = new File(Environment.getExternalStorageDirectory()
+ "/ColTmp/", "tmp.png");
        FileOutputStream fOut = new FileOutputStream(file);
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, fOut);
        fOut.flush();
        fOut.close();
        file.setReadable(true, false);
        Intent shareIntent = new Intent();
        shareIntent.setAction(Intent.ACTION_SEND);
        shareIntent.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(file));
        shareIntent.setType("image/png");
        startActivity(Intent.createChooser(shareIntent, getResour-
ces().getText(R.string.send_to)));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



Obrázok 13 Vyexportovaná farebná schéma

8.2 Ukladanie a externý prístup

Aplikácia pracuje s dátami uloženými na externom zariadení, aby k nim mal používateľ aplikácie umožnený prístup. Každá farebná schéma je uložená vo vlastnom textovom súbore s jednoduchou štruktúrou obsahujúcou hexadecimálnu hodnotu primárnej farby schémy, názov alebo dátum vytvorenia schémy a jednotlivé farby danej schémy, opäť zaznamenané hexadecimálne. V prípade, že súbor obsahuje viac schém, ako oddeľovač je použitá fráza „<NEXT_SCHEME>“. Z dôvodu jednoduchosti formátu uložených dát nebol použitý žiadny štandardný formát štruktúrovaného textu. Aplikácia najprv vyhodnotí, či existujú súbory s farbami a na základe toho načíta vopred uložené dáta z pamäte. Uložené súbory sú čítané po riadkoch a z dát sú tvorené objekty typu Color, ktoré sú pridané do zoznamu všetkých farieb ColorList.

V prípade ukladania si aplikácia najprv vytvorí pomocný zoznam farieb (ArrayList added) a za jeho pomoci si uchováva informáciu o farbách, ktoré už boli spracované a predpripravené na ukladanie. Aplikácia overí existenciu cesty a na základe toho pokračuje v ukladaní. Potom každú položku zo zoznamu farieb prevedie na textový formát a po riadku zapíše do súboru. Zapisuje všetky farebné schémy primárnej farby do jedného súboru a z toho dôvodu si drží informáciu o už spracovaných farbách, aby nevznikli redundantné dáta.

```
public void saveList() {
    File dir = new
File(Environment.getExternalStorageDirectory().getPath()+"/colors/");
    dir.mkdir();
    if (dir.isDirectory())
    {
        String[] children = dir.list();
        for (int i = 0; i < children.length; i++)
        {
            new File(dir, children[i]).delete();
        }
    }
```

```
    int counter = 0;
    for (Color _item : mainColorsList){
        try {

            File yourFile = new
File(Environment.getExternalStorageDirectory().getPath()+"/colors/"+(c
ounter+1)+".txt");
            Log.e("POKUS O UKLADANIE", String.valueOf(yourFile));
            yourFile.createNewFile();
            FileOutputStream out = new FileOutputStream(yourFile, fal-
se);

            PrintWriter pw = new PrintWriter(out);
            String title = _item.getTitle();
            pw.println(title);
            String date;

            for (int i = 0; i < _item.getColorSchemes().size(); i++){
                date = _item.getColorSchemes().get(i).getDate();
                pw.println(date);
                for (String color :
_item.getColorSchemes().get(i).getColors()){
                    pw.println(color);
                }
                if ((_item.getColorSchemes().get(i).getColors().size()
> 1) && (i != _item.getColorSchemes().size()-1)){
                    pw.println("<NEXT_SCHEME>");
                }
            }

                pw.flush();
                pw.close();
                out.close();
            //}
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        counter++;
    }
}
```

8.3 Import fotografie

Aplikácia umožňuje používateľovi výber z viacerých možností vloženia fotografie pre tvorbu farebných schém do aplikácie. Umožňuje použitie už vopred nasnímanej fotografie alebo obrázku a tak isto umožňuje použitie vstavaného fotoaparátu na nasnímanie fotografie. Pri importe nastáva zmena rozlíšenia fotografie na šírku 640 pixelov a výšku vypočítanú pre zachovanie pomeru strán a vytvorí sa nová aktivita, v ktorej prebieha samotná tvorba farebnej schémy.

8.3.1 Import z galérie

Pri vložení fotografie z galérie sa aktivuje výber z primárnej aplikácie určenej na prezeranie fotografii v zariadení. Samotná funkcia starajúca sa o import iba predá fotografiu ďalej a o nasledujúce spracovanie sa stará metóda onActivityResult.

```
public void loadPhotoFromGallery(View view){
    Intent pickPhoto = new Intent(Intent.ACTION_PICK, andro-
id.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(pickPhoto , 0);
}
```

8.3.2 Import z fotoaparátu

Systém Android umožňuje rovnako ako pri importe z galérie použiť systémovú funkciu, ktorá je ale nevyhovujúca z dôvodu tvorby fotografie v malom rozlíšení a jej neuloženi. Preto je použitý pozmenený postup, ktorý nasnímanú fotografiu ukladá do pamäte. Tak isto ako pri importe z galérie sa o spracovanie stará metóda onActivityResult.

```
public void loadPhotoFromCamera(View view) {
    String timeStamp = new SimpleDateFor-
mat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = timeStamp + ".jpg";
    File storageDir = Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES);
    pictureImagePath = storageDir.getAbsolutePath() + "/" + imageFile-
Name;
    File file = new File(pictureImagePath);
    Uri outputFileUri = Uri.fromFile(file);
    Intent cameraIntent = new In-
tent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
    cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
startActivityForResult(cameraIntent, 1);
}
```

8.4 Pixelizácia

Fotografia musí byť vopred zmenšená, pretože proces pixelizácie pri plnom rozlíšení fotografie bol príliš časovo náročný a vyvolával nestabilitu aplikácie. Používateľ má možnosť si na hlavnej obrazovke aplikácie vybrať možnosť zapnutia priemerovania farieb. V prípade, že je táto možnosť vypnutá sa z fotografie vyberie iba určitý počet pixelov a ostatné sa ignorujú. Tento spôsob je vybraný ako primárny, pretože proces pixelizácie je náročný aj pre výkonnejšie zariadenia.

8.4.1 Priemerovanie farieb

Aplikácia obsahuje triedu ImageData, ktorá si uchováva fotografiu, s ktorou sa práve pracuje a informácie o danej fotografii, ktoré aplikácia vygenerovala. Trieda umožňuje tieto súkromné atribúty použiť inde v aplikácii za použitia funkcii getImage a getImageColorData. Pri vytvorení inštancie triedy je vyžadovaná fotografia ako Bitmap. Ako vhodný počet blokov výslednej mozaiky bolo zvolených osem blokov, nakoľko ide o primerané množstvo, ktoré umožní používateľovi aplikácie ľahký výber a zároveň má matematický vzťah k štandardným pomerom strán digitálnej fotografie, ktoré sú 4:3 a 16:9. Počet riadkov mozaiky je odvodený od výšky fotografie a vypočítava sa v aplikácii na základe rozlíšenia fotografie. Aplikácia si vždy zoberie malú časť fotografie a vytvorí farebný priemer, ktorý zaznamená do dočasného ArrayListu. Na záver je jeho obsah skopírovaný do atribútu triedy imageColorData.

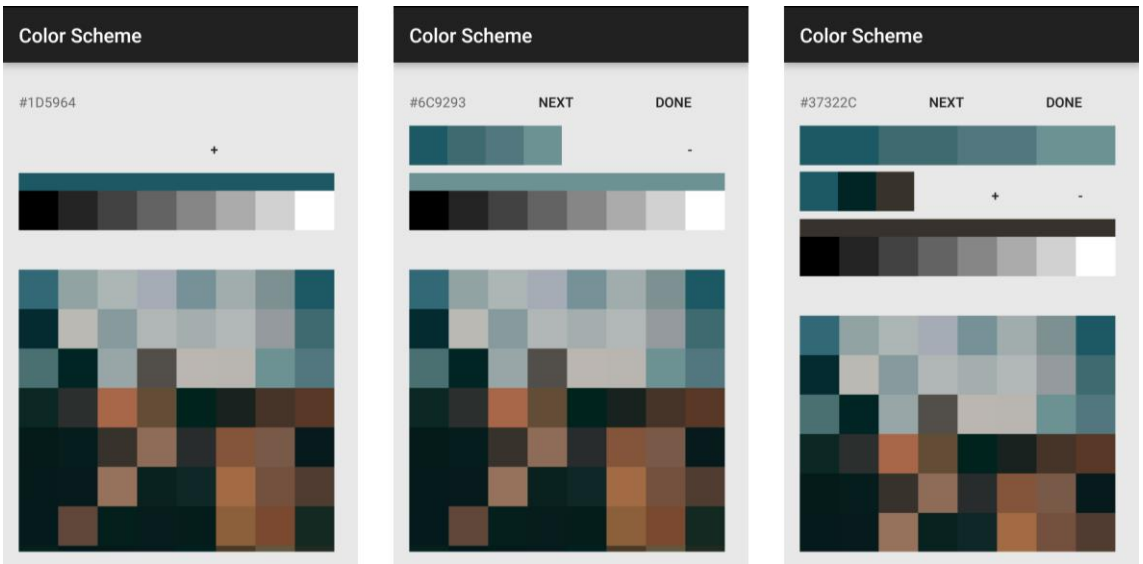
```
public void fillImageColorData(boolean average){

    float width = image.getWidth();
    float height = image.getHeight();
    float tmpNum = (height / width) * 8;
    int rows = Math.round(tmpNum);

    ArrayList<ArrayList<Long>> finalArrayList = new ArrayList<>();

    if (average) {
        for (int y = 0; y < rows; y++){
            ArrayList<Long> emptyArrayList = new ArrayList<>();
            for (int x = 0; x < 8; x++){
                long redBucket = 0;
                long greenBucket = 0;
                long blueBucket = 0;
                long pixelCount = 0;
                for (int yTmp = (y * (image.getHeight()/rows)); yTmp <
(y * (image.getHeight()/rows)) + (image.getHeight()/rows); yTmp++){
                    for (int xTmp = (x * (image.getWidth()/8)); xTmp <
(x * (image.getWidth()/8)) + (image.getWidth()/8); xTmp++){
```

```
                        pixelCount++;
                        redBucket += andro-
id.graphics.Color.red(image.getPixel(xTmp, yTmp));
                        greenBucket += andro-
id.graphics.Color.green(image.getPixel(xTmp, yTmp));
                        blueBucket += andro-
id.graphics.Color.blue(image.getPixel(xTmp, yTmp));
                    }
                }
                int averageColor = android.graphics.Color.rgb((int)
(redBucket/pixelCount), (int) (greenBucket/pixelCount), (int) (blue-
Bucket/pixelCount));
                emptyArrayList.add((long) averageColor);
            }
            finalArrayList.add(emptyArrayList);
        }

        imageColorData = finalArrayList;

    } else {
        for (int y = 0; y < image.getHeight();
y+=image.getHeight()/rows){
            ArrayList<Long> emptyArrayList = new ArrayList<>();
            for (int x = 0; x < image.getWidth();
x+=image.getWidth()/8){
                emptyArrayList.add((long) image.getPixel(x,y));
            }
            finalArrayList.add(emptyArrayList);
        }

        imageColorData = finalArrayList;
    }

}
```

Obrázok 14 Pôvodný obrázok (vľavo), preskakovanie pixelov (v strede) a priemerovanie pixelov (vpravo)

8.5 Tvorba farebných schém

Po úspešnom importovaní fotografie je používateľ aplikácie presmerovaný na obrazovku s výberom farieb do schémy. Na obrazovke sa na začiatku nachádza pruh farieb s čiernou, bielou a odtieňmi šedej a pod ním plocha tvorená farebnými štvorcami vytvorenými z fotografie. V tomto momente sa dá stlačiť každé políčko a v pruhu nad odtieňmi šedej sa zobrazí aktívna farba spolu s jej hexadecimálnym vyjadrením v ľavom hornom rohu. Pokiaľ používateľ stlačil ktorúkoľvek z farieb fotografie, zobrazí sa tlačidlo + pre pridanie farby do schémy. Nie je ale možné zvoliť žiadny z odtieňov šedej ako primárnu farbu a preto sa pri ich stlačení tlačidlo na pridanie nezobrazí. Po stlačení pridávacieho tlačidla sa nastaví primárna farba, zobrazí sa prvé políčko pracovnej farebnej schémy a vedľa tlačidla na pridanie farby (+) sa zobrazí tlačidlo na odstránenie farby (-). Do pracovnej schémy je možné pridávať ďalšie farby a aj farby z pruhu odtieňov šedej. V prípade, že pracovná schéma obsahuje minimálne dve farby, zobrazí sa tlačidlo pre pridanie novej pracovnej schémy (NEXT) a tlačidlo pre ukončenie práce (DONE). V prípade stlačenia tlačidla pre pridanie novej farebnej schémy sa predchádzajúca pracovná schéma zobrazí nad pracovnou schémou a je možné začať tvoriť schému novú, alebo ukončiť tvorbu s uložením hotových farebných schém tlačidlom pre ukončenie práce. Pracovná schéma sa neuloží a treba ju presunúť medzi hotové schémy tlačidlom pre pridanie novej pracovnej schémy. Po ukončení práce je používateľ presunutý na hlavnú obrazovku s výberom všetkých farieb.



Obrázok 15 Postup tvorby farebných schém z fotografie

Pod mozaikou farieb sa nachádza tlačidlo aktivujúce automatické generovanie farieb. V prípade, že je generovanie schém aktivované, aplikácia na základe primárnej farby vygeneruje nasledujúce farebné schémy:

- Achromatická – vygeneruje sa v prípade, že ako primárna farba bol zvolený odtieň sivej z fotografie. Vygeneruje sa jedna alebo dve schémy s prechodom od primárnej farby k bielej, čiernej alebo obom. V prípade, že by sa primárna farba nachádzala príliš blízko čiernej alebo bielej, je daná schéma ignorovaná, pretože by jej farby boli príliš podobné a splývali by.

- Achromatická s farebným akcentom – vygeneruje sa v prípade, že je ako primárna farba zvolená farba iná ako odtieň šedej. Podľa toho, aká tmavá je farba, sa vygeneruje jedna schéma s prechodom k bielej alebo k čiernej. Môže nastať situácia, že sa vygenerujú obe schémy. To nastáva v prípade, že sa primárna farba nachádza približne v rovnakej vzdialenosti od bielej a čiernej.

- Analogická malých rozdielov – aplikácia nájde susediace farby na kruhu farieb s rovnakými vlastnosťami vychádzajúcimi z farebného modelu HSV/HSB (hue, saturation, value/brightness) a pridá ich do schémy, ktorá v konečnom dôsledku obsahuje tri farby.

- Analogická stredných rozdielov – funguje na rovnakom princípe ako analogická malých rozdielov, ale na kruhu farieb nepracuje so susednými farbami, ale až s farbami nasledujúcimi.

- Trojitá – rovnaký princíp ako pri analogických schémach, ale k primárnej farbe sa pridávajú farby ležiace v tretinách kruhu, na vrcholoch rovnostranného trojuholníka.

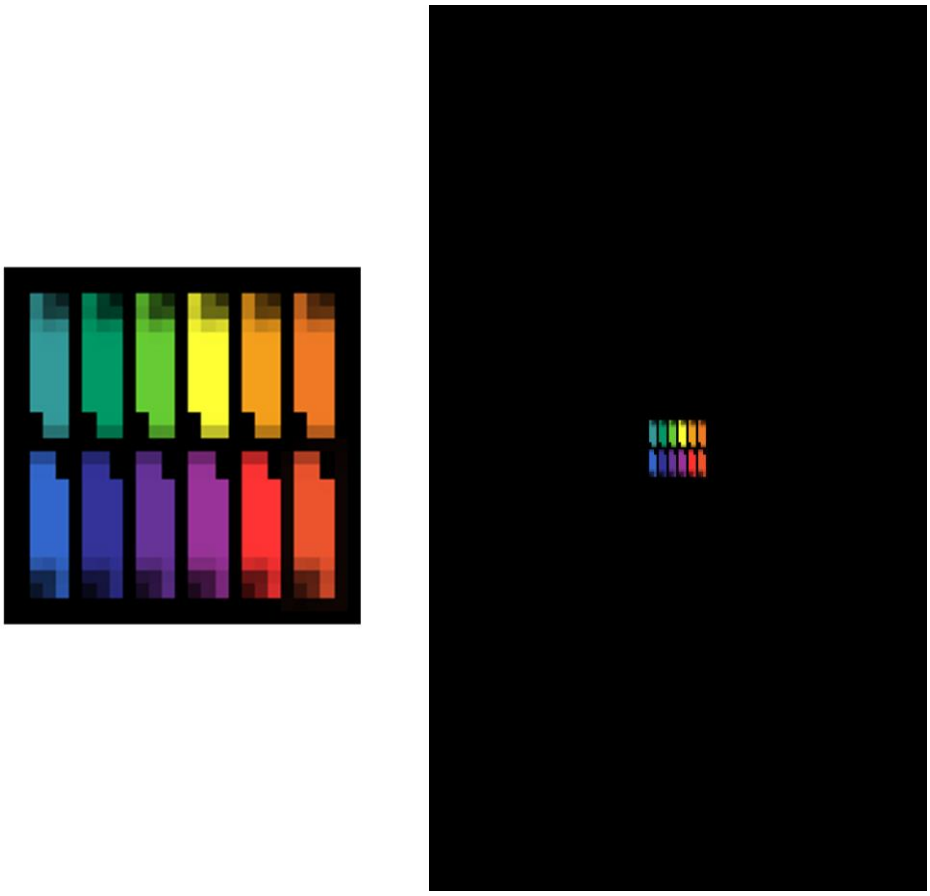
- Komplementárna rozdelená – funguje v princípe ako analogická schéma, ale k primárnej farbe sa pridávajú farby susediace s farbou komplementárnou.

- Komplementárna – k primárnej farbe sa pridá farba protiľahlá na kruhu farieb. Generuje sa najviac schém, ako prvá schéma obsahujúca iba obe farby. Nasleduje schéma s rovnakým zastúpením primárnej a sekundárnej farby (2+2), ktorých môže byť vygenerovaných viac, v závislosti od tmavosti a sýtosti primárnej farby. Za rovnakých pravidiel sa generujú schémy s väčšinovým zastúpením primárnej farby (1+3) a väčšinovým zastúpením farby komplementárnej (3+1).

- Komplementárna dvojitá – schéma je generovaná podobne ako schéma trojitá, ale do úvahy sa berú farby ležiace na štvrtinách kruhu farieb, teda na štvorci.

8.6 Prezentácia aplikácie

Aby aplikácia budila profesionálny dojem, sú potrebné určité drobnosti. Okrem jednotného designu a správne zvolenej farebnej palety je to aj ikona a štartovacia obrazovka (tzv. splashscreen). Ikona je v rôznych rozlíšeniach od 36 pixelov na 36 pixelov po 192 pixelov na 192 pixelov pre rozličné veľkosti a rozlíšenia obrazoviek zariadení. Tvorí ju čierna plocha s obdĺžnikovými objektami. Každý z týchto objektov je vyplnený jednou z dvanástich farieb z už spomínaného kruhu farieb a obsahuje mierny prechod do čiernej v rohu. Dokopy je na ikone okolo 190 rôznych farieb. Ikona evokuje zameranie aplikácie a to je práve práca s farbami. Štartovacia obrazovka sa zobrazí pri spúšťaní aplikácie a vypĺňa prázdny priestor. Znovu evokuje určitý dojem profesionality počas spúšťania procesov aplikácie. Dĺžka jej zobrazenia závisí od výkonu zariadenia. Štartovacia obrazovka je čierna plocha s ikonou aplikácie v strede a znovu jej rozlíšenie závisí od mobilného zariadenia. Aplikácia bola preložená do slovenského jazyka a jazyk sa zvolí automaticky v závislosti od nastavenia mobilného zariadenia, rovnako ako sa volí ikona a štartovacia obrazovka v závislosti od rozlíšenia obrazovky a ako sa mierne mení štýl tlačidiel v závislosti od verzie systému Android.



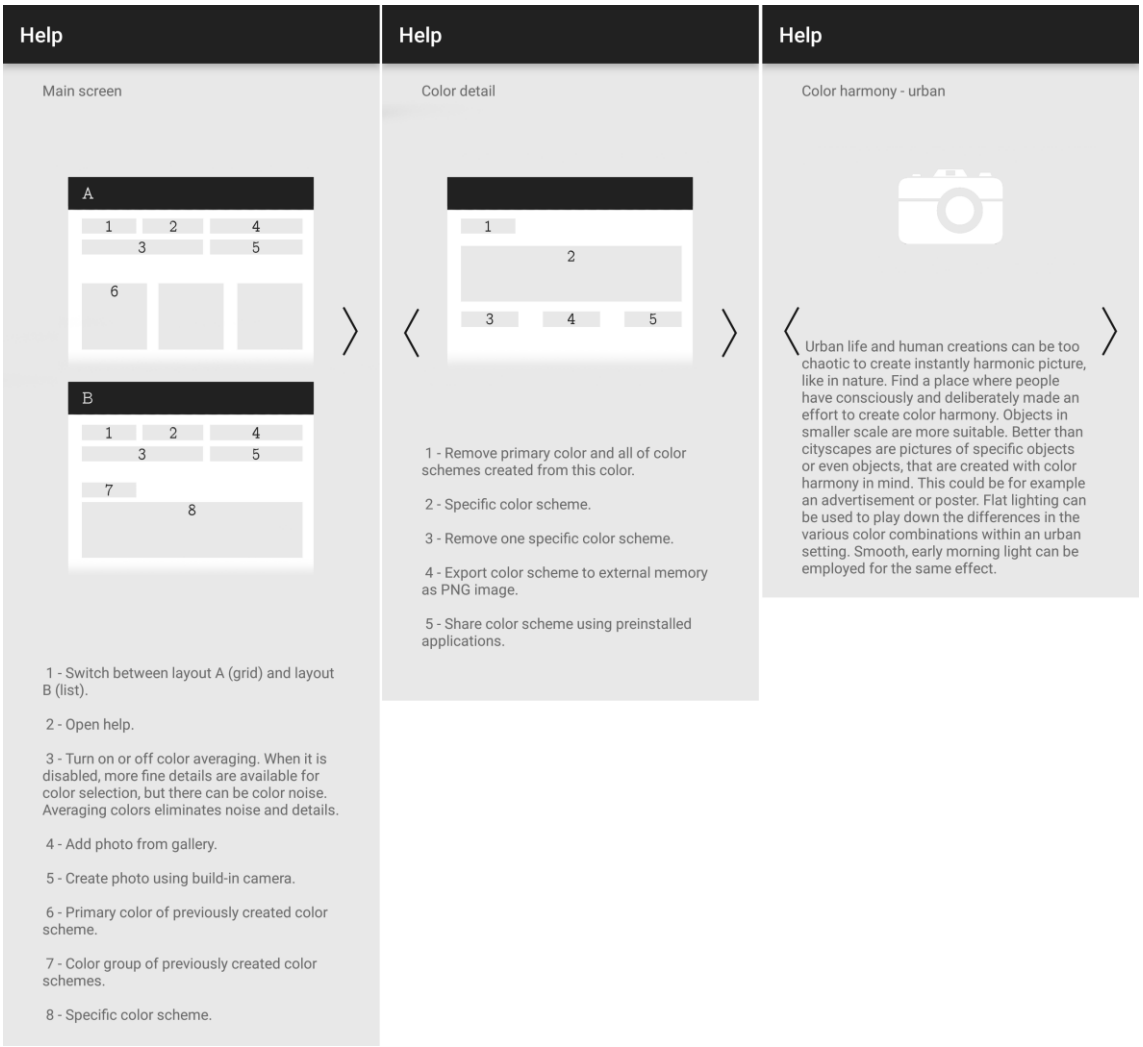
Obrázok 16 Ikona aplikácie (vľavo) a štartovacia obrazovka (vpravo)

8.7 Nápoveda

Sekcia nápoveda (HELP) je zobrazená po stlačení rovnomenného tlačidla na úvodnej obrazovke. Je vytvorená ako Tabbed Activity, čo znamená, že sa medzi jednotlivými stránkami posúva horizontálne. Polohu používateľa v nápovede indikuje existencia šípok na stranách obrazovky. V strede sa nachádza plocha umožňujúca zobrazenie grafiky a textu. Význam nápovedy spočíva v možnosti oboznámiť používateľa aplikácie s jej rozhraním, funkciami a ovládaním. V tomto prípade je využitá aj na tipy, ako nasnímať fotografiu vhodnú pre tvorbu harmonických farebných schém. Konkrétne položky nápovedy sú:

- Úvodná obrazovka – v tejto sekcii nápovedy sa nachádza popis používateľského rozhrania úvodnej obrazovky a jej segmentov.

- Detail farby – nápoveda popisuje obrazovku detailu konkrétnej farby a tak isto položky úvodnej obrazovky v režime zoznamu.
- Výber farby – nachádza sa tu popis jednotlivých prvkov na obrazovke tvorby farebných schém.
- Harmónia farieb (príroda) – táto sekcia nápovedy obsahuje postup tvorby farebne harmonickej fotografie v prírode a to napríklad snímaním rastlín.
- Harmónia farieb (mesto) – v tejto časti sú umiestnené možnosti tvorby harmonických fotografií v mestskom a človekom vytvorenom prostredí.
- Harmónia farieb (portrét a zátišie) – v poslednej časti nápovedy je spomínaná tvorba fotografií, pri ktorých má autor väčšiu kontrolu nad scénou a môže si dovoliť tvoriť farebnú harmóniu, ktorá sa dá inde nájsť ťažko.



Obrázok 17 Nápoveda a niektoré jej segmenty, líšiace sa dĺžkou

8.8 Konkurencia

Na trhu sa samozrejme už nachádzajú aplikácie, poskytujúce používateľovi podobné funkcie zamerané na tvorbu a generovanie farebných schém z fotografie. Pri podrobnejšom skúmaní však každá obsahuje určité nedostatky, ktoré boli práve v tejto práci eliminované. Najčastejšie boli nájdené tieto nedostatky:

- Chýbajúce funkcie – aplikácia neobsahovala možnosť tvorby vlastnej farebnej schémy, umožňovala iba automatické generovanie a opačnú situáciu, kde umožňovala tvorbu schémy, ale bez automatického generovania schém ďalších.
- Prémiový obsah – obsah nebol dostupný v základnej verzii aplikácie distribuovanej zdarma, ale bola možnosť chýbajúce funkcie dokúpiť alebo bola už základná verzia aplikácie spoplatnená.
- Automatický proces tvorby – generovanie farebnej schémy prebiehalo bez akéhokoľvek zásahu používateľa a nebola možnosť tento proces ovplyvniť. V tejto práci boli vytvorené viaceré postupy, ktoré si stále zachovali jednoduchosť, ale poskytujú používateľovi určitú kontrolu nad procesom.
- Sociálne funkcie a export – časť už dostupných aplikácii prekvapivo neobsahovala možnosť zdieľania a akéhokoľvek exportu vygenerovaných schém. Preto bola v tejto práci implementovaná možnosť exportu a zdieľania farebných schém za použitia vopred nainštalovaných aplikácii v mobilnom zariadení.
- Off-line obsah – samostatná kategória konkurencie sú webové služby, ktoré poskytujú rovnakú funkcionalitu ako aplikácia, ale za použitia internetového prehľadávača. Aj keď môžu byť nadradené množstvom funkcií a používateľským prostredím, ich hlavná nevýhoda spočíva v nutnosti internetového pripojenia. To stále na mobilnom zariadení nie je samozrejmosť, či už z finančných dôvodov alebo z dôvodu kvality signálu a pripojenia.

Konkurenčné aplikácie neobsahujú všetky tieto nedostatky naraz, ale ani žiadna z nich nebola schopná ich všetky eliminovať. Práve tieto nedostatky boli odstránené aplikáciou, popísanou v tejto práci, ktorá sa javí ako schopná alternatíva.

9 Záver

Cieľom tejto práce bolo vytvoriť aplikáciu pre platformu Google Android umožňujúcu tvorbu a automatické generovanie harmonických farebných schém. Tieto ciele boli úspešne dosiahnuté a aplikácia bola testovaná v emulátore systému Android a aj na viacerých fyzických zariadeniach s rozdielnymi verziami operačného systému. Aplikácia funguje bez problémov a používateľské prostredie sa automaticky prispôsobuje danému zariadeniu.

Táto práca poskytuje základné informácie a postupy tvorby aplikácie pre systém Android, popisuje vývojové prostredia a potrebné nástroje. Systém Android je open source, takže množstvo zdrojov, vývojových postupov, nástrojov a dokumentácie je možné nájsť zdarma na komunitných fórach.

S programovaním aplikácie pre platformu Google Android som nemal žiadne skúsenosti, takže táto práca bola pre mňa veľkým prínosom, hlavne čo sa týka oboznámenia sa s novými technológiami, nástrojmi určenými na tvorbu aplikácií, ale aj s teóriou fungovania komponentov aplikácie a samotného operačného systému Android.

Ďalší vývoj aplikácie môže byť zameraný v prvom rade na optimalizáciu kódu a niektorých použitých metód. Treba overiť stabilitu pri dlhodobom používaní na širokej škále zariadení a postupovať na základe získaných informácií. Aplikácia je pripravená na preklad do ďalších jazykov a následná lokalizácia je jednoduchá možnosť ďalšieho rozvoja. Tak isto môže byť po zvolení vhodného finančného modelu aplikácia umiestnená do obchodu s aplikáciami Google Play.

10 Literatúra

[1] Operating system market share: Market Share Statistics for Internet Technologies [online]. [cit. 2016-12-14]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1

[2] LACKO, Ľuboslav. Vývoj aplikací pro Android. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[3] ALLEN, Grant. Android 4: průvodce programováním mobilních aplikací. Brno: Computer Press, 2013. ISBN 978-80-251-3782-6.

[4] Java SE | Oracle Technology Network | Oracle [online]. [cit. 2016-12-14]. Dostupné z: http://www.oracle.com/technetwork/java/javase/overview/index.html

[5] Extensible Markup Language (XML) [online]. [cit. 2016-12-14]. Dostupné z: http://www.w3.org/XML/

[6] JSON [online]. [cit. 2016-12-14]. Dostupné z: http://www.json.org/

[7] Google Inc. Eclipse with ADT In: Android Developers [Online]. 2016 [Citace: 8. 11. 2016]. Dostupné z: hhttp : //developer.android.com/tools/help/adt.htmli

[8] Google Inc. Android Studio In: Android Developers [Online]. 2016 [Citace: 8. 11. 2016]. Dostupné z: hhttp : //developer.android.com/tools/studio/index.htmli

[9] Introduction - Material design - Material design guidelines [online]. [cit. 2016-12-14]. Dostupné z: https://material.google.com/

[10] Material motion - Motion - Material design guidelines [online]. [cit. 2016-12-14]. Dostupné z: https://material.google.com/motion/material-motion.html

[11] Choreography - Motion - Material design guidelines [online]. [cit. 2016-12-14]. Dostupné z: https://material.google.com/motion/choreography.html#

[12] Buttons: Floating Action Button - Components - Material design guidelines [online]. [cit. 2016-12-14]. Dostupné z: https://material.google.com/components/buttons-floating-action-button.html

[13] Color - Style - Material design guidelines [online]. [cit. 2016-12-14]. Dostupné z: https://material.google.com/style/color.html

[14] DANNHOFEROVÁ, Jana. Velká kniha barev: kompletní průvodce pro grafiky, fotografy a designéry. Brno: Computer Press, 2012. ISBN 978-80-251-3785-7.