VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIDEO MONTÁŽ NA PLATFORMĚ ANDROID

VIDEO MONTAGE ON ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MIROSLAV JANSKA**
AUTHOR

VEDOUCÍ PRÁCE — **MARTIN KOLÁŘ, M.Sc.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Janska Miroslav**
Program: Informační technologie
Název: **Video montáž na platformě Android**
Video Montage on Android
Kategorie: Softwarové inženýrství

Zadání:

1. Seznamte se s možnostmi editování videa na platformě Android
2. Vytvořte základní aplikaci s jednoduchým načítáním videa
3. Seznamte se s problematikou editování barev a tónů ve videu
4. Vytvořte ukázkovou aplikaci pro editování videa na Android, včetně stříhání a editování barev
5. Proveďte uživatelské testy srovnávající vytvořené rozhraní s existujícími aplikacemi

Literatura:

- Mednieks, Z.R., Dornin, L., Meike, G.B. and Nakamura, M., 2012. *Programming android*. " O'Reilly Media, Inc.".
- https://developer.android.com/reference/android/media/MediaCodec.html
- https://github.com/googlesamples/android-VideoPlayer

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kolář Martin, M.Sc.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 6. listopadu 2019

Abstrakt

Cílem této práce je implementovat aplikaci umožňující editování videa na platformě Android. Při implementaci byl použit framework FFmpeg umožňující manipulaci s multimediálním obsahem. Během implementace byla aplikace iterativně vylepšována a přizpůsobována požadavkům uživatelů. Výsledná aplikace umožňuje střih, spojování několika videí a aplikování efektů. Díky aplikaci může potencionální uživatel sestříhat natočená videa během pár minut.

Abstract

The main goal of this thesis is to implement an application enabling video editing on the Android platform. The FFmpeg framework was used in the implementation. It enables manipulation with multimedia content. The application was iteratively improved and adapted to user requirements during implementation. The resulting application allows video cutting, merging several videos, and applying effects. Thanks to the application a potential user can edit recorded videos in a few minutes.

Klíčová slova

video editor, Android, FFmpeg, JavaCV, ExoPlayer

Keywords

video editor, Android, FFmpeg, JavaCV, ExoPlayer

Citace

JANSKA, Miroslav. *Video montáž na platformě Android*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Kolář, M.Sc.

Video montáž na platformě Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Martina Koláře. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Miroslav Janska
27. května 2020

Poděkování

Tímto bych chtěl poděkovat Martinu Kolářovi M.Sc. za vedení této práce a za rady, které mi poskytl. Také bych chtěl poděkovat všem lidem, kteří se podíleli na testování aplikace.

Obsah

1 Úvod **3**

2 Platforma Android **4**
2.1 Operační systém 4
2.1.1 Architektura 5
2.2 Základní součásti aplikace 6
2.3 Aktivita 7
2.4 Služba 8
2.5 Fragment 8
2.6 Ukládání dat aplikace 9

3 Možnosti přehrávání a editování videa na platformě Android **11**
3.1 Přehrávání videa 11
3.1.1 MediaPlayer 11
3.1.2 ExoPlayer 12
3.1.3 Srovnání přehrávačů 12
3.1.4 Přehrávače třetích stran 14
3.2 Editování videa 15
3.2.1 Android Media API 15
3.2.2 FFmpeg 15
3.2.3 Knihovny zapouzdřující funkcionalitu frameworku FFmpeg 17

4 Editování barev ve videu **19**
4.1 Základní pojmy 19
4.2 Editování barev pomocí nástroje ffmpeg 20

5 Existující řešení **22**
5.1 VidTrim 22
5.2 WeVideo 23
5.3 Quick 23

6 Cíle **25**

7 Návrh **26**
7.1 Analýza a specifikace 26
7.1.1 Cílová skupina 26
7.1.2 Sběr a analýza požadavků 26
7.1.3 Případy užití 27

7.2 Návrh uživatelského rozhraní . . . 28
7.3 Návrh struktury databáze . . . 31
7.4 Návrh operací s videem . . . 32
7.4.1 Proces exportování . . . 32
7.4.2 Časové optimalizace . . . 32
7.5 Návrh testování . . . 33

8 Implementace **34**
8.1 Použité technologie a postupy . . . 34
8.1.1 Knihovny pro práci s videem . . . 34
8.1.2 Zvolená architektura . . . 35
8.1.3 Data Binding . . . 35
8.1.4 Live Data . . . 36
8.1.5 DBFlow . . . 36
8.2 Uživatelské rozhraní . . . 36
8.2.1 Změny po uživatelském testování . . . 36
8.2.2 Implementace uživatelského rozhraní . . . 37
8.3 Implementace exportování . . . 39
8.4 Výsledky testování . . . 40
8.4.1 Výsledky uživatelského testování . . . 40
8.4.2 Výsledky testování funkčnosti . . . 40

9 Závěr **41**

Literatura **42**

A Obsah CD **44**

Kapitola 1

Úvod

Již několik let je nejpopulárnějším mobilním operačním systémem Android[1]. Většina uživatelů vlastnící zařízení se zmíněným systémem ho používá pro komunikaci, sledování videí či procházení webu, ale kromě toho také pro natáčení videí[2]. Po natočení videí přichází potřeba jejich následné úpravy. Mnoho uživatelů však nepotřebuje obrovské množství funkcionality, které nabízí profesionální editory. Jejich záměrem je co nejrychlejší úprava natočených videí.

Cílem této práce je implementovat Android aplikaci, která bude umožňovat editování videa. Aplikace bude nabízet možnost střihu či spojení několika videí nebo možnost aplikování obrazových efektů. Kapitola 2 seznamuje s platformou Android a klíčovými komponenty použitými při vývoji. Kapitola 3 obsahuje informace o možnostech nabízejících se při vývoji aplikace umožňující přehrávání a editování videa. Kapitola 4 se věnuje editování barev a tónu ve videu. Kapitola 5 poskytuje informace k již existujícím řešením, které vznikly s podobným záměrem jako cílová aplikace. Kapitola 6 obsahuje shrnutí stanovených cílů. Kapitola 7 je zaměřena na návrh uživatelského rozhraní, databáze, operací s videem a testování. Kapitola 8 obsahuje informace k implementaci aplikace. Nejprve kapitola popisuje technologie a postupy, které byly při implementaci použity. V další části obsahuje popis implementace uživatelského rozhraní a procesu exportování. Nakonec se kapitola věnuje výsledkům testování.

[1] Viz https://gs.statcounter.com/os-market-share/mobile/worldwide

[2] Viz https://www.gadget-cover.com/blog/what-are-the-most-popular-reasons-why-people-use-their-smartphones-every-day

Kapitola 2

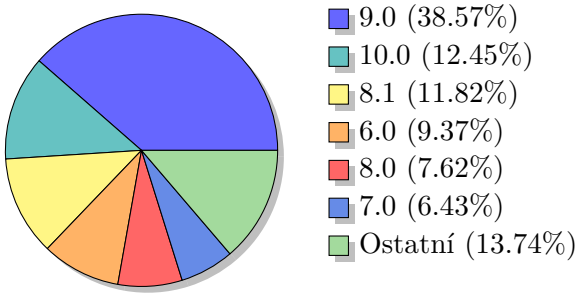
Platforma Android

Tato kapitola nejprve seznámí čtenáře s operačním systémem Android. V další části kapitola poskytuje informace o klíčových komponentech použitých při vývoji cílové aplikace. V závěru se kapitola věnuje oprávněním nutným pro běh aplikací a způsobům perzistence dat.
Pokud není explicitně uvedeno, jsou informace v celé kapitole čerpány z knihy Mistrovství - Android [15] a oficiální dokumentace [8]. Ukázky kódu jsou uváděny v jazyce Java.

2.1 Operační systém

První telefon s operačním systémem Android spatřil svět již 23. září roku 2008. Jednalo se o HTC Dream[1], na kterém běžel Android 1.0. Od té doby se stal Android nejpopulárnějším operačním systémem na mobilních zařízeních. Kromě chytrých telefonů se nachází na tabletech, hodinkách, televizích a například na i chytrých ledničkách.

Vzhledem k tomu, že se jedná o *open-source* platformu, umožňuje výrobcům přidávat vlastní rozšíření. V dnešních dnech mají chytré telefony často stejné hardwarové komponenty, a tak je to jedna z mála možností jak se odlišit od konkurence.



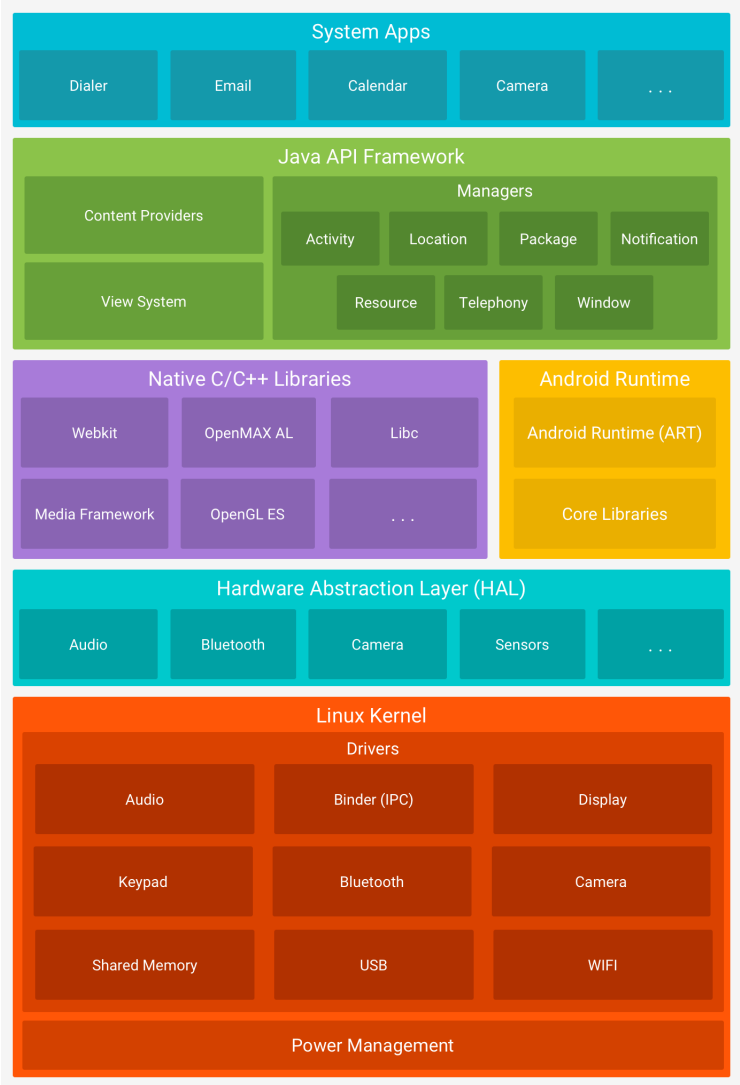
Obrázek 2.1: Zastoupení verzí operačního systému.

Při vývoji aplikace je důležité sledovat podíl jednotlivých verzí systému na trhu, aby nebyla aplikace spustitelná pouze na malém procentu zařízení. Z grafu na obrázku 2.1, který obsahuje data z března roku 2020[2], lze vypozorovat, že většina zařízení běží na verzi 6.0 a novější.

[1] Viz https://www.gsmarena.com/htc_dream-2665.php
[2] Viz https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide

2.1.1 Architektura

Architektura operačního systému Android se skládá z 5 vrstev. Posloupnost těchto vrstev lze vidět na obrázku 2.2.



Obrázek 2.2: Architektura operačního systému Android. Obrázek pochází z oficiální dokumentace.

Nejnižší vrstvu představuje **linuxové jádro**. To je zredukováno o nepotřebné funkce a přizpůsobeno běhu na mobilních zařízeních. Jeho primárním účelem je interakce s hardwarovými komponenty zařízení. V omezené míře umožňuje vykonávání linuxových příkazů [23].

Nad jádrem se nachází vrstva **HAL** (*Hardware Abstraction Layer*). Ta poskytuje vyšším vrstvám rozhraní pro komunikaci s hardwarem. Každý modul, tvořící tuto vrstvu, zapouzdřuje funkcionalitu jedné hardwarové komponenty. Vývojáři díky ní nemusí znát hardwarové specifikace všech zařízení.

Následuje vrstva tvořená **nativními knihovnami** a **běhovým prostředím**. Funkcionalita knihoven, které jsou napsány v jazycích C a C++, je aplikacím zpřístupněna pro-

střednictvím aplikačního frameworku. Knihovny je možné využít při vývoji aplikací, které vyžadují částečnou implementaci v nativním kódu. Pro informace k běhovému prostřední je vhodná publikace Mistrovství - Android [15].

Aplikační framework, který tvoří další vrstvu, poskytuje přístup k často využívaným funkcionalitám systému. Jedná se o nejdůležitější vrstvu pro vývojáře. Její nejpodstatnější části jsou:

- **View System** - nabízí základní prvky uživatelského rozhraní jako jsou tlačítka, ikony, prvky na zobrazení textu a mnoho dalších,
- **Content Provider** - poskytuje přístup k datům aplikace či datům jiných aplikací,
- **Resource Manager** - umožňuje přístup k souborům statického charakteru jako jsou například řetězce (*strings*), bitmapy nebo definice rozložení (*layouts*),
- **Notification Manager** - umožňuje zobrazení upozornění na notifikační liště zařízení,
- **Activity Manager** - spravuje životní cyklus aplikací a zajišťuje uchování dat při přechodu mezi aktivitami.

Nejvyšší vrstvu tvoří zabudované systémové aplikace jako jsou prohlížeč, kalkulačka, kalendář, kontakty apod. Uživatel není nijak omezen používat tyto aplikace jako výchozí. Má možnost je nahradit aplikacemi třetích stran.

2.2 Základní součásti aplikace

Součástí každé aplikace je soubor *AndroidManifest.xml*. Jsou v něm definovány informace jako základní aktivita, orientace, ikona aplikace a základní komponenty. Základními komponenty jsou:

- **Aktivita** (*Activity*) - reprezentuje jednu obrazovku aplikace, bližší informace poskytuje podkapitola 2.3,
- **Služba** (*Service*) - umožňuje na pozadí provádět dlouhotrvající operace jako například exportování videa, více informací poskytuje podkapitola 2.4,
- **Broadcast receiver** - slouží pro příjem všesměrového vysílání a nabízí možnost zpětné reakce,
- **Content provider** - vytváří kompaktní rozhraní pro přístup k datům.

Kromě zmíněných informací soubor *AndroidManifest.xml* obsahuje specifikaci oprávnění, které aplikace potřebuje pro přístup k zabezpečeným částem systému. Od verze 7.0 je vyžadováno získat tyto oprávnění i za běhu aplikace interakcí s uživatelem. Tabulka všech elementů, které se v souboru mohou vyskytovat je dostupná v oficiální dokumentaci[3].

[3]Tabulka elementů: https://developer.android.com/guide/topics/manifest/manifest-intro

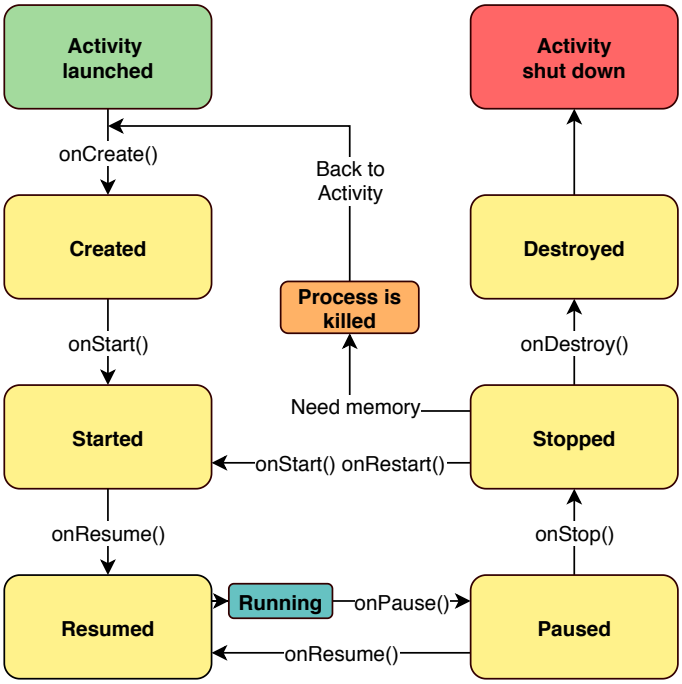
2.3 Aktivita

Aktivita je komponent reprezentující jednu *obrazovku* aplikace, která je zobrazena uživateli. Aplikace je většinou tvořena několika aktivitami. Výjimku mohou tvořit aplikace navržené podle návrhového vzoru *Single Activity*. Pro více informací je vhodný článek [19] porovnávající klady a zápory využití zmíněného návrhového vzoru.

Hlavním účelem aktivity je zobrazovat informace uživateli a reagovat na vyvolané události. Každá aktivita prochází svým životním cyklem, který lze vidět na obrázku 2.3, během kterého se dostává do třech hlavních fází.

Pokud dochází k interakci s uživatelem a aktivita se nachází na vrcholu zásobníku aktivit, jedná se o fázi **na popředí** (typicky ve stavu *Running* či *Resumed*). Do fáze, kdy je aktivita **pozastavená**, ale je stále částečně viditelná, se nejčastěji dostane při vyvolání dialogového okna, během kterého dochází k přenesení fokusu z aktivity na dialog. Ve fázi, kdy je aktivita **zastavená**, nastává překrytí jinou aktivitou a pro uživatele již není viditelná. Aktivita je stále uložená v paměti, proto není problém se k ní v případě potřeby vrátit.

Vývojáři je umožněno metody, které jsou volány v závislosti na změně stavu životního cyklu aktivity, v případě zájmu přepisovat.



Obrázek 2.3: Životní cyklus aktivity. Informace použité pro vizualizaci pocházejí z oficiální dokumentace.

Během průchodu životním cyklem jsou volány následující metody:

- **onCreate()**
  - volání během prvního spuštění aktivity či změně orientace zařízení
  - vyskytují se zde inicializační metody a funkce
- **onStart()**
  - volání během přípravy přechodu aktivity na popředí

- **onResume()**
  - volání během uvedení **na popředí**
  - po zavolání může docházet k interakci s uživatelem
  - často se zde vyskytují zpětné alokace uvolněných zdrojů
- **onPause()**
  - volání během toho co aktivita opouští popředí a je **pozastavena** (může být stále viditelná)
  - mohou se zde vyskytovat metody nebo funkce pro uvolňování zdrojů, která mají nízkou časovou náročnost
- **onStop()**
  - aktivita již není viditelná a je **zastavena**
  - měly by se zde vyskytovat časově náročnější operace uvolňování zdrojů, které nejsou potřeba během toho, co je aktivita na pozadí
- **onDestroy()**
  - volání během ukončení životnosti aktivity

2.4 Služba

Služba je komponent umožnující vykonávání dlouhotrvajících operací na pozadí. Těmi může být exportování videa či přehrávání hudby. Na rozdíl od aktivity nepotřebuje uživatelské rozhraní a není pozastavena, přestože dojde k opuštění aplikace.

Při svém spuštění služba nevytváří vlastní vlákno, ale je spuštěna v hlavním vlákně hostitelského procesu. Z toho důvodu je důležité pro náročnější operace vytvořit nové vlákno. Pokud by byly operace prováděny v hlavním vlákně aplikace, mohlo by dojít k blokování interakce s uživatelem nebo by mohla aplikace přestat reagovat.

Pokud má k vykonávání operací docházet jen při interakci uživatele s aplikací, je lepší volbou vytvoření nového vlákna než využití služby.

2.5 Fragment

Fragment je komponent umožnující modulární rozdělení aktivity. Není nutnou součástí aplikace, ale jeho znovupoužití v několika aktivitách umožňuje zjednodušení vývoje.

Musí být vždy připojen k hostitelské aktivitě, protože bez ní jeho existence není možná. Aktivita silně ovlivňuje jeho životní cyklus. Pokud je aktivita pozastavena, jsou pozastaveny i její fragmenty apod. Nicméně i za běhu aktivity lze přidávat nebo odstraňovat fragmenty, díky čemuž lze vytvořit aplikaci, která obsahuje několik fragmentů, ale pouze jedinou aktivitu. Prostřednictvím hostitelské aktivity mohou mezi sebou jednotlivé fragmenty komunikovat.

Životní cyklus fragmentu má kromě stejných metod jako aktivita navíc 4 další metody, které jsou stejně jako u aktivity v případě nutnosti přepisovatelné. Za zmínku stojí metoda **onAttach()**, která je volána během připojení fragmentu k hostitelské aktivitě, a metoda

onCreateView(), jenž je volána během prvního vykreslení fragmentu. Podrobný popis životního cyklu fragmentu se nachází v oficiální dokumentaci[4].

2.6 Ukládání dat aplikace

Android nabízí několik možností pro ukládání dat aplikace, jenž mají být dostupné i po opětovném spuštění aplikace. Těmi jsou:

- **Sdílené preference** - umožňují ukládaní primitivních datových typů[5] ve tvaru klíč-hodnota,
- **Interní úložiště** - umožňuje ukládání souborů, ke kterým nemají ve výchozím stavu přístup jiné aplikace,
- **Externí úložiště** - umožňuje ukládání souborů, ke kterým mají přístup i ostatní aplikace,
- **Databáze** - umožňuje ukládání strukturovaných datových typů v privátní databázi aplikace.

Objektově relační mapování (ORM)

Databáze je jedním z nejvíce využívaných způsobů pro zajištění perzistence dat. Z toho důvodu Android poskytuje možnost využití *SQLite*[6] databáze. S tím ale přichází nevýhoda spojená se znalostí jazyka SQL potřebného pro práci s daty.

Řešení poskytuje objektově relační mapování. To se stará o konverzi mezi záznamy relační databáze a objekty objektově orientovaného jazyka. Umožňuje vývojáři pracovat s daty uloženými v databázi jako s objekty bez nutnosti napsání jediného dotazu v jazyce SQL.

Knihoven nabízejících zmíněnou funkcionalitu je celá řada. Za zmínku stojí například knihovna *Room*[7], která je součástí skupiny komponent Android Jetpack[8], nebo knihovna *DBFlow* [11], která je jednou z nejvyužívanějších knihoven třetích stran.

Rozdíl mezi dotazem bez využití objektově relačního mapování a dotazem s jeho využitím lze vidět na ukázce 2.4.

[4] Viz https://developer.android.com/guide/components/fragments
[5] Primitivní datové typy v Javě: http://kubu.wz.cz/help/java/datatype.html
[6] Viz https://developer.android.com/training/data-storage/sqlite
[7] Viz https://developer.android.com/training/data-storage/room
[8] Viz https://developer.android.com/jetpack

Čisté SQLite:

```
ArrayList<String> names = new ArrayList<String>();

Cursor result = database.rawQuery( "SELECT `name` FROM `Employee` WHERE
    `salary`<50000 AND `salary`>25000", null );

result.moveToFirst();
while(result.isAfterLast() == false) {
   names.add(result.getString(result.getColumnIndex("name")));
   result.moveToNext();
}
```

Dotaz pomocí knihovny DBFlow:

```
ArrayList<String> names = new ArrayList<>();

List<Score> employeeList = SQLite.select()
   .from(Employee.class)
   .where(Employee_Table.salary.lessThan(50000))
   .and(Employee_Table.salary.greaterThan(25000))
   .queryList();

foreach(Employee employee: employeeList) {
   names.add(employee.getName());
}
```

Obrázek 2.4: Rozdíl mezi dotazem bez využití objektově relačního mapování a dotazem s jeho využitím. Pomocí obou ukázek jsou získána jména zaměstnanců, kteří mají plat mezi 25 a 50 tisíci.

Kapitola 3

Možnosti přehrávání a editování videa na platformě Android

Tato kapitola má za cíl seznámit s možnostmi, které se nabízí při vývoji aplikace umožňující přehrávání a editování videa. První část obsahuje obecný přehled možností pro přehrávání videa. Druhá část kapitoly seznamuje s knihovnami umožňující editování videa. Na konci kapitoly je podrobněji popsán framework FFmpeg.

3.1 Přehrávání videa

Tato podkapitola je zaměřená na přehrávače, které poskytuje společnost Google. Srovnává jejich klady a zápory a obsahuje porovnání s přehrávači třetích stran. Pro využití ve vyvíjené aplikaci musí přehrávač splňovat jisté požadavky. Jedná se zejména o možnost plynulého přehrávání videa a podpora dostatečného množství formátů.

3.1.1 MediaPlayer

Jedná se o třídu, která je součásti multimediálního frameworku systému Android. Umožňuje přehrávání audio a video souborů uložených v interní či externí paměti zařízení. Dále také umožňuje přehrávání multimediálního obsahu přes síťové připojení, pokud k tomu má aplikace patřičné oprávnění. Informace jsou převzaty z dokumentace [10].
Příklad některých podporovaných kodeků a mediálních kontejnerů:

Video: **H.263** (3GPP, MP4), **H.264** (3GPP, MP4, MPEG-TS), **MPEG-4 SP** (3GPP), **VP8** (WebM, Matroska), **VP9** (WebM, Matroska)

Audio: **AAC LC** (3GPP, MP4, MPEG-TS), **FLAC** (FLAC), **MP3** (MP3), **Opus** (Matroska), **Vorbis** (Ogg, Matroska)

*Příklady jsou uváděny ve formátu **kodek** (kontejner/y).*

Kromě audia a videa umožňuje zobrazovat obrázky. Jedná se o soubory typu BMP, GIF, JPEG, PNG, WebP a HEIF. Nicméně tato funkcionalita nebude ve vyvíjené aplikaci využita.

Seznam všech podporovaných kodeků a formátů je uveden v oficiální dokumentaci[1].

3.1.2 ExoPlayer

Jedná se o *open-source* přehrávač, který umožňuje přehrávání multimediálního obsahu. Jeho součásti využívají MediaCodec API[2] a je vyvíjen společností Google, která ho využívá ve své službě YouTube. Je distribuován pod licencí *Apache 2.0*, proto existuje mnoho aplikací a knihoven, které přehrávač ExoPlayer využívají. Představuje vhodnou alternativu k MediaPlayer API a oproti němu nabízí více funkcionality. Zejména se jedná o adaptivní streaming[3], práci s titulky, přehrávání reklam a rozšíření o framework FFmpeg. Informace jsou převzaty z dokumentace přehrávače [9].

Seznam podporovaných kontejnerů:
MP4, **M4A**, **FMP4**, **WebM**, **Matroska**, **MP3**, **Ogg**, **WAV**, **MPEG-TS**, **MPEG-PS**, **FLV**, **ADTS**
Některé funkcionality, jako například skok ve videu, nemusí být podporovány u všech z nich.

Díky rozšíření pomocí frameworku FFmpeg[4] podporuje velké množství audio kodeků:
Vorbis, **Opus**, **FLAC**, **ALAC**, **PCM μ-law**, **PCM A-law**, **AMR-NB**, **AMR-WB**, **AC-3**, **E-AC-3**, **DTS**, **DTS-HD**, **TrueHD**

Podporované formáty titulek:
WebVTT, **TTML/SMPTE-TT**, **SubRip**, **SubStationAlpha (SSA/ASS)**

3.1.3 Srovnání přehrávačů

Podle výše zmíněných informací to vypadá, že vhodnou volbou pro využití v aplikaci je jeden i druhý přehrávač, opak je ale pravdou.

Minimální množství kódu pro zprovoznění je jedna z výhod přehrávače MediaPlayer. Pro přehrávání stačí dva řádky kódu, které se nacházejí v ukázce na obrázku 3.1.

```
1 MediaPlayer player = MediaPlayer.create(context, sourceUri, display);
2 player.start();
```

Obrázek 3.1: Inicializace a spuštění přehrávače MediaPlayer.

Na pozadí je ale mezitím prováděno velké množství operací, nad kterými nemá vývojář přímou kontrolu. I to je jeden z důvodů, které vedly k vzniku přehrávače ExoPlayer.
V ukázce na obrázku 3.2 lze vidět inicializaci přehrávače ExoPlayer. Zdrojové kódy jsou převzaty z cílové aplikace. Informace k jejich popisu pocházejí z dokumentace přehrávače a diplomové práce Jana Benedikta [3].

[1] Seznam podporovaných formátů: https://developer.android.com/guide/topics/media/media-formats
[2] Viz https://developer.android.com/reference/android/media/MediaCodec
[3] Technika používaná při streamování multimédií po síti. Je detekována šířka pásma a CPU uživatele. Podle získaných informací se nastavuje kvalita mediálního proudu.
[4] Repozitář rozšíření: https://github.com/google/ExoPlayer/tree/release-v2/extensions/ffmpeg

```
1   //1. krok
2   TrackSelector trackSelector = new DefaultTrackSelector();
3   //2. krok
4   DefaultLoadControl loadControl = new DefaultLoadControl.Builder().
        setBufferDurationsMs(5000, 10000, 1500, 3000).
        createDefaultLoadControl();
5   //3. krok
6   SimpleExoPlayer player = ExoPlayerFactory.newSimpleInstance(context,
        trackSelector, loadControl);
```

Obrázek 3.2: Inicializace přehrávače.

- V **1. kroku** je vytvořen `TrackSelector`, který má na starost volbu nejvhodnější video a audio stopy. Zvolený `DefaultTrackSelector` je vyhovující pro většinu případů užití. Pro volbu nejvhodnější video a audio stopy v případě adaptivního streamování je vhodné využít `BandwidthMeter`, který poskytuje třídě `TrackSelector` informace o aktuální šířce pásma a ta na základě toho volí stopu, která má optimální bitrate[5].

- V **2. kroku** je inicializován `DefaultLoadControl`, který má na starost vyrovnávací paměť. Vyrovnávací paměť umožňuje plynulé přehrávání videa i přes krátkodobé výpadky sítě. Nevyhovující výchozí parametry vyrovnávací paměti lze jednoduše modifikovat pomocí metody `setBufferDurationMs()`.

- A ve **3. kroku** je možné inicializovat `SimpleExoPlayer`, který poskytuje rozhraní pro ovládání přehrávání videa.

V okamžiku, kdy je přehrávač inicializován, mu může být předán zdroj dat. Obrázek 3.3 obsahuje ukázku nastavení zdroje dat a spuštění přehrávání.

```
1       DataSource.Factory dataSourceFactory = new
            DefaultDataSourceFactory(context,Util.getUserAgent(context, "
            video_editor"));
2       MediaSource videoSource = new ProgressiveMediaSource.Factory(
            dataSourceFactory).createMediaSource(sourceUri);
3       player.prepare(videoSource);
4       player.setPlayWhenReady(true);
```

Obrázek 3.3: Nastavení zdroje dat a spuštění přehrávání.

- Nejdříve je nutné vytvořit instanci třídy `DataSource`, která umožňuje vytvořit objekt `MediaSource`, jenž obsahuje veškeré informace potřebné pro přehrávání videa. Posléze může být objekt předán přehrávači a může být spuštěno přehrávání pomocí metody `setPlayWhenReady()`.

[5] Množství dat přenesených za jednotku času. Nejčastější jednotkou je počet bitů za sekundu.

3.1.4 Přehrávače třetích stran

Kromě zmíněných přehrávačů, které poskytuje společnost Google, existuje velké množství alternativ. Jako ukázka jsou zde uvedeny dva přehrávače se zajímavou funkcionalitou.

Pokud je při vývoji aplikace požadováno přehrávání co největšího množství formátu, existující vhodnější alternativy než přehrávače MediaPlayer a ExoPlayer. Využití těchto alternativ nicméně přináší i svá negativa. Mohou jimi být velmi strohá dokumentace nebo v některých případech dokonce ukončený vývoj. Tyto okolnosti mohou hrát zásadní roli v časové náročnosti vývoje aplikace.

Fanplayer

Fanplayer [12] je přehrávač, který je šířený pod licencí *GNU Lesser General Public License verze 3*[6] a využívá framework FFmpeg. S přestávkami je vyvíjen jediným člověkem, jímž je Chen Kai z Číny. Přesto se jedná o plnohodnotný přehrávač videa s nízkým využitím CPU a paměti. Podporuje velké množství formátů. Jejich výčet lze vidět v tabulce 3.4

Jeho velkou nevýhodou je dokumentace pouze v čínštině. Jedinou možností jak získat informace je studiem zdrojového kódu. Dále pak neumožňuje jednoduchou aplikaci obrazových efektů. Pro vývoj aplikací sloužících pro editování videa se proto příliš nehodí.

Pro více informací o mnoha jiných funkcionalitách je vhodné navštívit repozitář přehrávače.

GiraffePlayer2

GiraffePlayer2 [25] je přehrávač, který vychází z jiného velmi komplexního přehrávače IJK[7]. I tento přehrávač využívá framework FFmpeg, díky čemuž podporuje velké množství formátů. Poskytuje mnoho zajímavých funkcionalit. Je jimi například přehrávání videa v plovoucím okně nebo ovládání hlasitosti a jasu pomocí gest.

Nenabízí možnost jednoduchého aplikování obrazových efektů. Díky možnosti umístění videa do `RecyclerView`[8] a možnosti přehrávání videa v plovoucím okně se spíše hodí pro vývoj aplikací podobných službě YouTube.

[6] Viz https://www.gnu.org/licenses/lgpl-3.0.html
[7] Viz https://github.com/bilibili/ijkplayer
[8] Viz https://developer.android.com/jetpack/androidx/releases/recyclerview

3.2 Editování videa

Tato podkapitola obsahuje přehled možností, které lze využít při vývoji aplikace umožňující editování videa. Konkrétně se jedná o dvě nejvíce využívané možnosti, kterými je je balík *android.media* a framework FFmpeg. Na konci kapitola obsahuje bližší seznámení s knihovnami, které zapouzdřují funkcionalitu frameworku FFmpeg.

3.2.1 Android Media API

android.media [7] je standardní balík systému Android. Poskytuje třídy, které lze využít pro přehrávání a střih multimediálního obsahu. Tyto třídy umožňují pracovat s jednotlivými snímky videa.

Je vhodnou volbou spíše pro jednodušší střih videa. Díky své jednoduchosti je pro implementaci složitějších operací vhodnější dále zmíněný framework FFmpeg, který navíc podporuje větší množství formátů.

3.2.2 FFmpeg

FFmpeg [6] je multiplatformní multimediální framework obsahující velké množství nástrojů a knihoven, které umožňují nahrávání, konverzi, filtrování a streamování videa či audia. Je vyvíjen pod systémem Linux, ale lze ho zkompilovat i pro jiné platformy jako je například Android, na kterém poté mohou být spouštěny dále zmíněné nástroje. Je distribuován pod licencí *GNU LGPL* nebo *GNU GPL*. Licence je závislá na tom, s jakými součástmi je framework zkompilován. Informace v této části pocházejí z dokumentace frameworku.

Hlavní součásti

FFmpeg se skládá z části, které dohromady vytvářejí mocný nástroj pro editování videa.

Nástroje příkazového řádku:

- **ffmpeg** - nástroj, který zapouzdřuje funkcionalitu knihoven, které jsou zmíněny dále (například umožňuje konverzi formátu, změnu vzorkovací frekvence, změnu rozlišení apod.),
- **ffplay** - multimediální přehrávač založený na knihovnách frameworku FFmpeg využívající pro přehrávání knihovnu SDL[9],
- **ffprobe** - nástroj pro analýzu multimediálního obsahu podobající se aplikaci MediaInfo[10].

Knihovny:

- **libavutil** - knihovna obsahující pomocné funkce pro ostatní částí frameworku (jedná se například o generátory náhodných čísel nebo matematické či kryptografické funkce),
- **libavcodec** - jedna z hlavních knihoven obsahující audio a video kodéry,
- **libavformat** - knihovna určená pro demultiplexing a multiplexing multimediálního obsahu,

[9] Viz https://www.libsdl.org/
[10] Viz https://mediaarea.net/en/MediaInfo

- **libavdevice** - knihovna umožňující komunikaci se vstupními a výstupními zařízeními přes multimediální rozhraní (Video4Linux2, VfW, DShow, ALSA),
- **libavfilter** - knihovna obsahující filtry multimediálního obsahu,
- **libswscale** - knihovna poskytující funkce pro změnu rozlišení a barevného modelu,
- **libswresample** - knihovna umožňující optimalizované převzorkování a změnu formátu audia.

FFmpeg v příkazovém řádku

Jak již bylo zmíněno, framework FFmpeg lze zkompilovat i pro platformu Android. Nástroj **ffmpeg** lze poté používat pro úpravu videa stejným způsobem jako na systému Linux.

Ukázka použití:

```
$ ffmpeg -i vstup.mp4 vystup.mkv
```

- Konverze vstupního souboru s formátem MP4 na formát MKV.

```
$ ffmpeg -i vstup.mp4 -vf scale=1920:1080 vystup.mp4
```

- Změna rozlišení vstupního videa na 1920x1080 využitím filtru, jenž je zadáván pomocí parametru -vf.

```
$ ffmpeg -i vstup.mp4 -ss 10 -to 20 vystup.mp4
```

- Vyříznutí částí vstupního videa od 10. po 20. sekundu.

```
$ ffmpeg -i vstup.mp4 -r 30 vystup.mp4
```

- Změna snímkovací frekvence vstupního videa na 30. Některé snímky videa jsou při konverzi duplikovány nebo zahazovány. Z toho důvodu má video stále stejnou časovou délku.

```
$ ffmpeg -i vstup.mp4 -ss 10 -to 20 -vf scale=1920:1080 -r 30 vystup.mkv
```

- Kombinace předchozích příkazů

Pokud mají videa stejný formát, audio kodek a video kodek. Lze je jednoduše spojit do jednoho videa pomocí následujícího příkazu.

```
$ ffmpeg -f concat -safe 0 -i videa.txt -c copy vystup.mkv
```

- Soubor `videa.txt` obsahuje seznam souborů, jenž se mají spojit. Jeho obsah musí být v následujícím formátu:

```
file '/home/videa/vystup0.mkv'
file '/home/videa/vystup1.mkv'
file '/home/videa/vystup2.mkv'
```

Pro více informací o nástroji **ffmpeg** je vhodné navštívit dokumentaci frameworku.

Formáty	**Video kodeky**	**Audio kodeky**
AVI	H.261–H.265	AAC LC
FLV	WMV1, WMV2, WMV3	FLAC
MP4	QuickTime	MP3
Matroska	RealVideo	Opus
Ogg	AV1	TrueHD
IFF	Intel Indeo	Vorbis
OMA	VP9	AMR-NB, AMR-WB

Obrázek 3.4: Vybrané formáty a audio/video kodeky podporované frameworkem FFmpeg.

FFmpeg jich podporuje obrovské množství. Jejich tabulku lze nalézt v dokumentaci frameworku[11].

3.2.3 Knihovny zapouzdřující funkcionalitu frameworku FFmpeg

Tato menší podkapitola obsahuje přehled knihoven, které poskytují funkcionalitu frameworku FFmpeg na platformě Android.

MobileFFmpeg

MobileFFmpeg [21] je knihovna poskytující stejnou funkcionalitu jako nástroje **ffmpeg** a **ffprobe**. Kromě platformy Android je distribuována i pro platformy iOS a tvOS. Je vydávána ve dvou verzích, z nichž jedna podporuje již Android 4.1 (SDK[12] 16). Je šířena pod licencí GNU LGPL 3.0 nebo GNU GPL 3.0. Výběr licence závisí na použitých součástech knihovny. Práce s knihovnou je velmi jednoduchá. Ukázku provedení příkazu z předcházející podkapitoly, lze vidět na obrázku 3.5.

[11] Viz https://www.ffmpeg.org/general.html#toc-File-Formats
[12] Software development kit - nástroje pro vývoj aplikací pro platformu Android

```
int rc = FFmpeg.execute("-i vstup.mp4 -ss 10 -to 20 -vf scale
    =1920:1080 -r 30 vystup.mkv ");
if (rc == RETURN_CODE_SUCCESS) {
    Log.i(Config.TAG, "Provedeni prikazu probehlo uspesne.");
}
```

Obrázek 3.5: Ukázka provedení příkazu pomocí knihovny MobileFFmpeg.

JavaCV

JavaCV [2] je knihovna, která obsahuje kolekci adaptérů knihoven[13] poskytujících funkcionalitu v oblasti počítačového vidění. Jsou jimi například OpenCV, FlyCapture, OpenKinect, librealsense, CL Eye Driver, videoInput, ARToolKitPlus a mimo to také framework FFmpeg. Kromě nich poskytuje vlastní třídy nástrojů, které v jisté míře zjednodušují používání funkcí zmíněných knihoven a vytváří mezi nimi a aplikací jakousi mezivrstvu. Kolekce adaptérů je opravdu veliká, proto lze importovat pouze některé z nich. Licence knihovny se může lišit podle použitých součástí. Nevýhodou použití knihovny je téměř neexistující dokumentace. Jediný způsob získání informací o jejím použití je z několika málo příkladů nebo ze sekce *Issues* jejího repozitáře na službě GitHub. To práci s knihovnou činí časově náročnou. Jak již bylo zmíněno, knihovna poskytuje vlastní třídy nástrojů umožňující editaci multimediálního obsahu. Jedná se zejména o třídy `FrameGrabber`, `FrameFilter`, `FrameConverter` a `FrameRecorder`. **FrameGrabber** je třída, která má mnoho podtříd (`FFmpegFrameGrabber`, `OpenCVFrameGrabber` aj.). Je vhodná pro získání jednotlivých snímků videa. **FrameFilter** je třída, která má také několik podtříd (`FFmpegFrameFilter` aj.). Nabízí možnost použití filtrů na video či audio podobně jako s využitím parametru -vf u nástroje **ffmpeg**. **FrameConverter** je pomocná třída umožňující jednoduchý převod obrazových dat mezi jednotlivými API. Jedná se například o převod struktury *Mat* knihovny OpenCV[14] na Android Bitmap. Poslední zmíněná třída **FrameRecorder** má také několik podtříd (`FFmpegFrameRecorder`, `OpenCVFrameRecorder`). Umožňuje ze získaných snímků vytvořit nové video. Na obrázku 3.6 lze vidět ukázku práce s knihovnou.

```
FFmpegFrameGrabber grabber = new FFmpegFrameGrabber("home/videa/vstup
    .mp4");grabber.start();
FFmpegFrameRecorder recorder = new FFmpegFrameRecorder("home/videa/
    vystup.mkv",1920,1080);recorder.start();

while(true){
    Frame frame = grabber.grabFrame();
    if(frame==null)break;
    recorder.record(frame);
}
```

Obrázek 3.6: Konverze formátu a změna rozlišení.

[13] Viz https://github.com/bytedeco/javacpp-presets
[14] Viz https://docs.opencv.org/trunk/d3/d63/classcv_1_1Mat.html

Kapitola 4

Editování barev ve videu

Cílem kapitoly je nejprve seznámit s pojmy nutnými pro zasvěcení do problematiky a poté poskytnout ukázku aplikování obrazových efektů pomocí nástroje **ffmpeg**. Informace vyskytující se v teoretické části pocházejí z přednášky předmětu IZG [17] a článku publikovaného na webu wikisofia [26].

4.1 Základní pojmy

Barva

Barva je vjem subjektivního charakteru vytvářený na sítnici lidského oka vyvolaný působením světla daného spektra šířeného ze zdroje. Získaná informace je z oka předána do mozku a podle vlnové délky interpretována jako barva.

Základní vlastnosti barvy:

- **Jas** - jedná se o intenzitu světla či světelného zdroje. Umožňuje zesvětlit či ztmavit video.
- **Sytost** - určuje *čistotu* barvy světla. Umožňuje upravit *živost* obrazu.
- **Barevný tón** (odstín) - dominantní vlnová délka světla, určující převládající barvu. Umožňuje upravit barevný nádech videa.

Barvy lze rozdělit na:

- **achromatické barvy** – černá, bílá a šedá – odlišeny pouze úrovní jasu a sytosti,
- **monochromatické barvy** – variace jednoho odstínu na základě hodnot sytosti,
- **chromatické barvy** – ostatní barvy barvy kromě achromatických.

Barevný model

Barevný model popisuje barvy na základě podílu jednotlivých složek. Existuje jich velké množství. Jsou jimi například **RGB** (základní model počítačové grafiky), **CMYK** (základní model pro tištěný výstup) a **YUV** (model využívaný například u televizního vysílání).

Filtrování obrazu

Filtrování obrazu je proces, při kterém dochází k nahrazování jednotlivých barevných složek každého pixelu novou hodnotou. Tyto hodnoty jsou vypočítány pomocí lineárních či nelineárních rovnic.

V případě zájmu o více informací o dané problematice je vhodná publikace Color appearance models [5].

4.2 Editování barev pomocí nástroje ffmpeg

Nástroj **ffmpeg** nabízí několik možností, které lze využít pro úpravu barev. Jednou z nich je filtr `colorchannelmixer`, jenž umožňuje modifikaci jednotlivých barevných složek každého pixelu. Následovat bude ukázka jeho použití.

Snímek použitý pro ukázku byl získán pomocí nástroje **ffmpeg** z testovacího videa, které pochází z webu Internet Archive[1].

Šedotónový filtr

Použitím filtru vznikne šedě tónovaný obraz. Výsledná intenzita I je pro každý pixel vypočítána dosazením aktuálních hodnot jednotlivých barevných složek do rovnice 4.1. R značí červenou, G zelenou a B modrou složku. Vypočítaná hodnota I je posléze dosazena jako hodnota všech barevných složek daného pixelu. Hodnoty konstantních složek rovnice se mohou v jistých implementacích lišit. Na obrázku 4.2 lze vidět výsledek po použití filtru [18].

$$0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad = \quad I \tag{4.1}$$

Ekvivalentní příkaz nástroje **ffmpeg**:

```
colorchannelmixer=.299:.587:.114:0:.299:.587:.114:0:.299:.587:.114
```

Obrázek 4.1: Původní obraz.

Obrázek 4.2: Po použití filtru.

[1] Využité video: https://archive.org/details/SampleVideo1280x7205mb

Sépiový filtr

Použitím filtru vznikne sépiově tónovaný obraz. Na rozdíl od předešlého filtru jsou hodnoty všech barevných složek pixelu počítány samostatně. Značení zůstává stejné. Nové hodnoty jsou vypočítány pomocí rovnic 4.3, 4.4 a 4.4. Na obrázku 4.4 lze vidět výsledek po použití filtru [22].

$$0.393 \cdot R + 0.769 \cdot G + 0.189 \cdot B = nove\_R \tag{4.2}$$
$$0.349 \cdot R + 0.686 \cdot G + 0.168 \cdot B = nove\_G \tag{4.3}$$
$$0.272 \cdot R + 0.534 \cdot G + 0.131 \cdot B = nove\_B \tag{4.4}$$

Ekvivalentní příkaz nástroje **ffmpeg**:

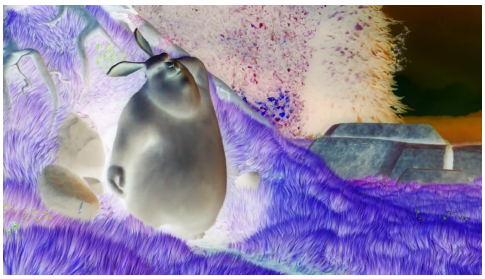
```
colorchannelmixer=.393:.769:.189:0:.349:.686:.168:0:.272:.534:.131
```

Obrázek 4.3: Původní obraz.

Obrázek 4.4: Po použití filtru.

Další možností aplikování efektu je použitím filtru `negate`, jenž invertuje barvy. Filtr nemá žádné parametry. Výsledek jeho použití lze vidět na obrázku 4.6.

Obrázek 4.5: Původní obraz.



Obrázek 4.6: Po použití filtru.

Více informací o dalších možnostech editování barev ve videu se nachází v rozsáhlé dokumentaci[2].

[2] Viz https://ffmpeg.org/ffmpeg-filters.html#Video-Filters

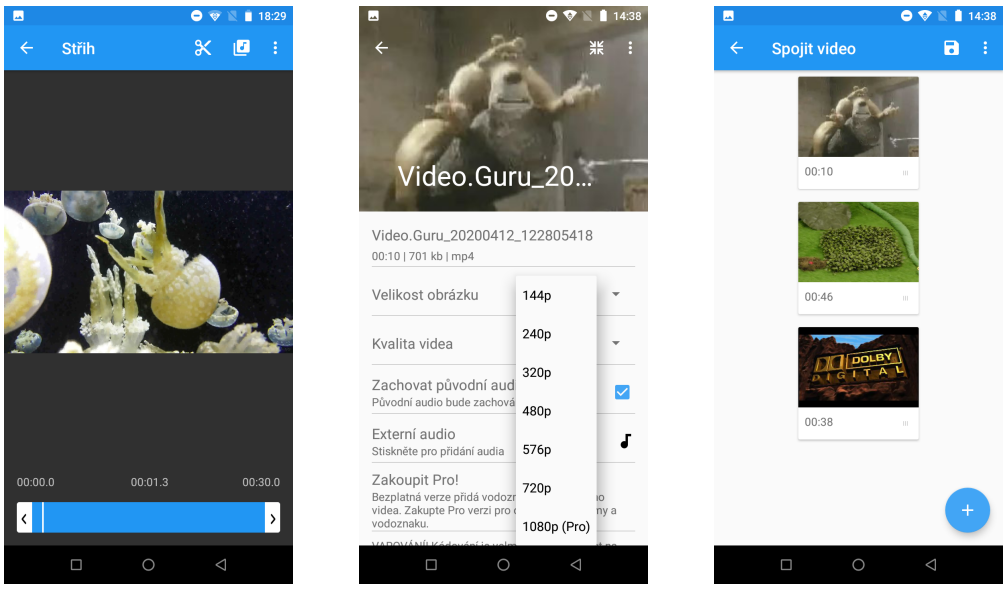
Kapitola 5

Existující řešení

V dnešní době již existuje velké množství aplikací zaměřených na editování videa na platformě Android. Každá z nich má své pro i proti. Následující kapitola obsahuje výběr několika aplikací, které poskytují základní respektive složitější možnosti editování videa. Jedná se o editory, které vznikly s podobným záměrem jako cílová aplikace. Tím je možnost upravení videa během několika málo minut. Informace o počtu stažení jsou aktuální k 1. 5. 2020.

5.1 VidTrim

Aplikace VidTrim[1] poskytuje základní možnosti editace videa. Umožňuje střih videa, spojení několika videí do jednoho (bez možnosti vložení přechodu) a aplikování obrazových efektů. Dále umožňuje překódování videa do několika předdefinovaných rozlišení. Aplikace má ve službě GooglePlay přes 10 milionů stažení a její hodnocení jsou většinou kladné. Existuje také placená verze aplikace. Ta odstraní reklamu z exportovaného videa. Ukázku aplikace lze vidět na obrázku 5.1.



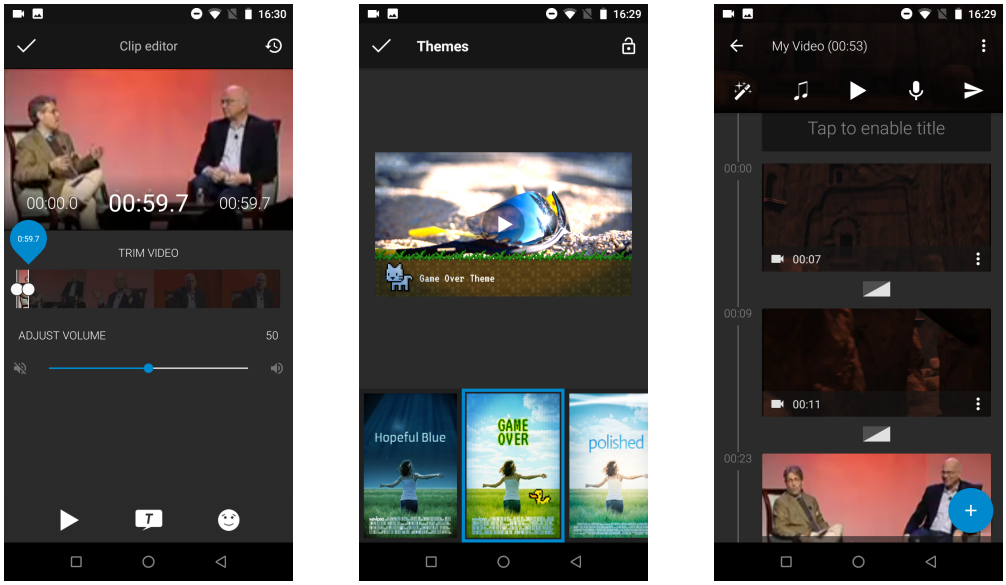
Obrázek 5.1: Aplikace VidTrim.

[1] Viz https://play.google.com/store/apps/details?id=com.goseet.VidTrim

Operace s videem jsou implementovány pomocí frameworku FFmpeg. Přesto aplikace nenabízí žádnou jinou volbu výstupního formátu než je MP4. Jednou z dalších nevýhod je, že neumožňuje vykonání několika operací současně. Například se jedná o sjednocení videí spojené s aplikováním obrazových efektů. Uživatel je nucený video exportovat, aby následně mohl provést další operaci. Uživatelské rozhraní aplikace není příliš vhodné pro střih delších videí, protože neumožňuje zvolení přesného času střihu.

5.2 WeVideo

Aplikace WeVideo[2] je další poměrně jednoduchý editor videa. Kromě většiny základních funkcí poskytovaných i předchozí aplikací nabízí možnost vkládání textu, emotikonu či přechodu mezi videa. Na rozdíl od předchozí aplikace umožňuje vykonávat několik operací současně, díky čemu šetří drahocenný čas uživatelů. Na službě GooglePlay má aplikace přes 10 milionů stažení a je pravidelně aktualizována. Ukázku aplikace lze vidět na obrázku 5.2.



Obrázek 5.2: Aplikace WeVideo.

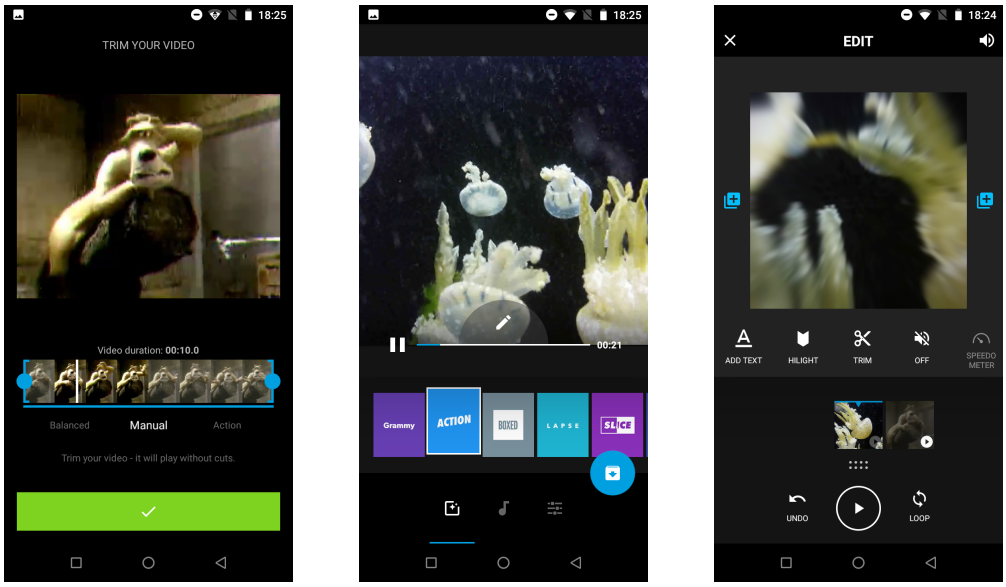
Editor neumožňuje přidání obrazového efektu na konkrétní video. Místo toho nabízí možnost výběru tématu, po jehož zvolení editor automaticky vloží efekty, hudbu a přechody mezi jednotlivá videa. Při exportu videa aplikace ve verzi zdarma nenabízí žádnou možnost výběru výstupního formátu či rozlišení a video obsahuje reklamu. Přestože uživatelské rozhraní aplikace je poměrně zdařilé, není příliš vhodné pro střih delších videí.

5.3 Quick

Aplikace Quick[3], dříve známá pod jménem Replay, je editor videa vyvíjený společností GoPro. Kromě podobných funkcí jako předchozí dvě aplikace nabízí editor například možnost zaostření na určitý bod ve videu. Aplikace si klade za cíl umožnit uživatelům editování videa během několika málo minut.

[2]Viz https://play.google.com/store/apps/details?id=com.wevideo.mobile.android
[3]Viz https://play.google.com/store/apps/details?id=com.stupeflix.replay

Po výběru z paměti telefonu či jiného úložiště jsou videa analyzována a je z nich sestaveno výsledné video obsahující zajímavé momenty. Uživatelům je následně umožněno jednotlivá videa doupravit manuálně. Aplikace má přes 100 milionů stažení na GooglePlay a její hodnocení jsou víceméně jen pozitivní. Ukázku aplikace lze vidět na obrázku 5.2.



Obrázek 5.3: Ukázka aplikace Quick.

Aplikace bohužel nenabízí žádnou volbu výstupního formátu a nabízí pouze výběr ze dvou rozlišení. Je patrné, že cílem aplikace není vytvářet konkurenci profesionálním editorům. Jejími typickými uživateli jsou spíše lidé, kteří chtějí rychle sestříhat natočená videa a popřípadě je sdílet s dalšími uživateli. Bližší informace o aplikaci pocházejí z článku publikovaného v magazínu Svět Androida [13].

Kapitola 6

Cíle

Cílem je vytvoření aplikace, která bude poskytovat základní operace nutné pro upravování videa. Jedná se zejména o střih a spojení několika videí. Dále by měl editor umožňovat aplikování jednoduchých obrazových efektů. Uživatelé by měli mít možnost si přehrát upravená videa ještě před samotným exportováním. V dnešní době si uživatelé mohou do svého zařízení stáhnout videa z mnoha zdrojů v jakémkoli možném formátu, proto by jich aplikace měla podporovat co nejvíce.

Nevýhodou mnoha aplikací je téměř minimální možnost volby při exportování výsledného videa. Proto by měla aplikace poskytovat více výstupních formátů, rozlišení a například i volbu počtu snímku za sekundu.

Typickými uživateli by měli být lidé, kteří chtějí v co nejkratším časovém úseku sestříhat svá natočená videa. Kritickým faktorem pro takové uživatele je doba, po kterou musí interagovat s uživatelským rozhraním aplikace. Proto by cílová aplikace měla obsahovat pouze nutnou funkcionalitu a uživatelské rozhraní by mělo být přizpůsobeno rychlé práci s videem. A i přesto, že úpravou videa by měli uživatelé strávit pouze minimální množství času, tak by se veškeré jimi provedené úpravy měly průběžně ukládat. Z důvodu toho kdyby se k nim chtěli později vrátit.

Rekapitulace stanovených cílů:

- Poskytovat operace jako je střih a spojení několika videí.
- Podporovat co největšího množství formátů videa.
- Umožňovat aplikaci obrazových efektů.
- Poskytovat možnost volby výstupních parametrů.
- Přehrávat upravená videa bez jejich předchozího exportování.
- Aplikace bude obsahovat pouze nezbytnou funkcionalitu.

Kapitola 7

Návrh

Následující kapitola obsahuje informace týkající se kompletního návrhu vyvíjené aplikace. Nejprve se kapitola zabývá specifikací a analýzou požadavků na výsledný produkt. V další části se kapitola zabývá návrhem uživatelského rozhraní, návrhem struktury databáze a návrhem operací s videem. Konec kapitoly obsahuje návrh testování srovnávajícího navržené uživatelské rozhraní s jinými aplikacemi. Diagram případů užití a navržené obrazovky aplikace byly nakresleny pomocí nástroje draw.io[1].

7.1 Analýza a specifikace

Tato podkapitola nejprve seznámí s cílovou skupinou aplikace a procesem zjišťování požadavků. V druhé části se podkapitola věnuje diagramu případů užití vytvořeného na základě analýzy požadavků.

7.1.1 Cílová skupina

Uživatelé, kteří by měli tvořit cílovou skupinu vyvíjené aplikace, jsou obyčejní lidé, kteří potřebují v co nejkratším čase upravit natočená videa. Věkové rozpětí potenciálních uživatelů není nijak omezeno a mezi cílovou skupinu patří i lidé pouze s minimální úrovní technické zdatnosti.

7.1.2 Sběr a analýza požadavků

Sběr požadavků na vyvíjenou aplikaci probíhal s potenciálními uživateli formou dialogů. Následně proběhla korektura duplicitních, nesmyslných a protichůdných požadavků.

Na základě analýzy jednotlivých požadavků byl sestaven diagram případů užití. Poté proběhla zpětná konzultace s uživateli, která měla za cíl najít případné chyby v sestaveném diagramu.

Uživatelé již během vývoje testovali prototypy aplikace a ukázalo se, že některé případy užití nebyly formulovány dostatečně přesně. Proto byly následně doplněny.

[1] Viz https://drawio-app.com/

7.1.3 Případy užití

Na základě analýzy požadavků vznikl diagram případů užití. Diagram je složený z aktéra a případů užití. Aktérem je uživatel, který interaguje s aplikací. Diagram obsahuje i případy užití, které přibyly až během vývoje aplikace.

Obrázek 7.1: Diagram případů užití vyvíjené aplikace.

Prvním případem užití je **zobrazení seznamu projektů**. Po jeho zobrazení může uživatel **vytvořit**, **smazat** nebo **zobrazit konkrétní projekt**. Po zobrazení projektu je uživateli nabídnuta celá řada dalších možností. První z nich umožňuje **přidat video**. Pokud

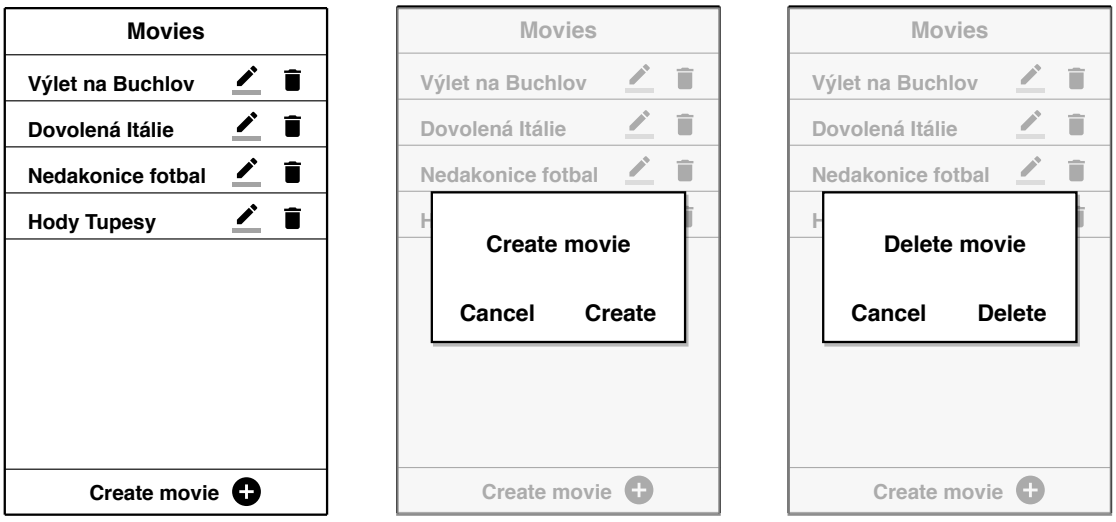
se v seznamu videí již nějaké video nachází, tak má uživatel možnost volby konkrétního místa pro přidání videa. Pokud je uživatelem nebo automaticky aplikací **vybráno video**, tak může uživatel **zobrazit informace o videu** a pokud je vybrané video podporováno, tak jej může **přehrát**. Dále může video ze seznamu **odstranit** nebo **přesunout** na jiné místo. Důležitým případem užití je možnost **úpravy parametrů vybraného videa**, která v sobě skrývá možnost oříznutí času videa nebo výběr obrazového efektu, který bude na video aplikován jak během přehrávání tak při exportování projektu.

Pokud se uživatel rozhodne může zobrazený **projekt exportovat** či modifikovat nevyhovující nastavení **výstupních parametrů**.

Kromě uvedených základních případů užití se uživateli nabízí možnost duplikování videa, rozdělení videa na dvě a nebo vrácení smazaného projektu.

7.2 Návrh uživatelského rozhraní

Na základě vytvořeného diagramu případu užití následně vznikl první návrh uživatelského rozhraní. Vzhledem k cílové skupině obsahuje návrh pouze ovládací prvky, které jsou nutné pro zajištění základní funkcionality.



Obrázek 7.2: Obrazovka zobrazující seznam projektů

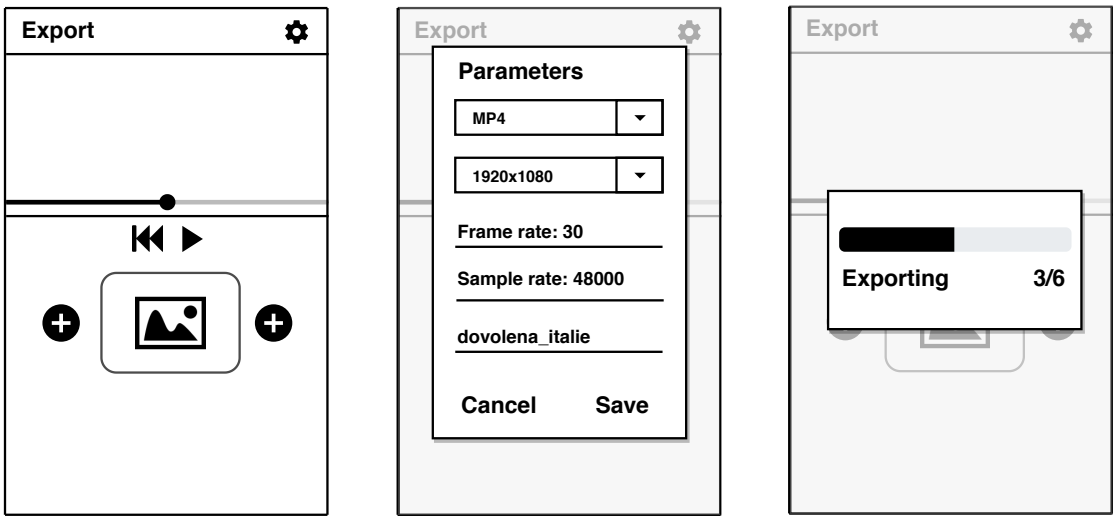
Na obrázku 7.2 lze vidět návrh obrazovky, která bude zobrazována po startu aplikace. Respektive bude zobrazována až jako druhá, protože v průběhu implementace před ni bude navržena `loading` obrazovka.

Uživatel zde uvidí seznam svých projektů, které jsou pracovně v návrhu nazvány `Movies`. Projekt bude reprezentován svým názvem a u každého z nich budou zobrazeny tlačítka pro smazání a editaci názvu projektu. Pro lepší přehled bude seznam seřazený podle data poslední editace. Po kliknutí na položku seznamu, kromě míst kde se nacházejí zmiňovaná tlačítka, bude uživateli zobrazen konkrétní projekt.

Dále se na této obrazovce bude nacházet tlačítko `Create movie`, jehož prostřednictvím bude moci uživatel vytvořit nový projekt. V návrhu lze vidět, že operace jako vytvoření, smazání nebo editace názvu projektu budou řešeny pomocí dialogů.

Po zobrazení konkrétního projektu bude uživateli zobrazena nejdůležitější obrazovka aplikace, jejíž návrh lze vidět na obrázku 7.3. Bude sloužit pro editování multimediálního obsahu.

V horní navigaci se budou nacházet tlačítka nezahrnují funkcionalitu spojenou s videi, ale s celým projektem. Jedná se o tlačítka umožňující úpravu výstupních parametrů a exportování projektu. Jak lze vidět v návrhu na obrázku 7.3, uživatel bude mít na výběr z několika předem definovaných formátů a rozlišení. Ostatní parametry kromě názvu souboru budou voleny automaticky. Při exportování projektu bude uživateli zobrazován počet exportovaných dílčích částí, díky čemu bude mít uživatel lepší přehled o probíhajícím procesu.



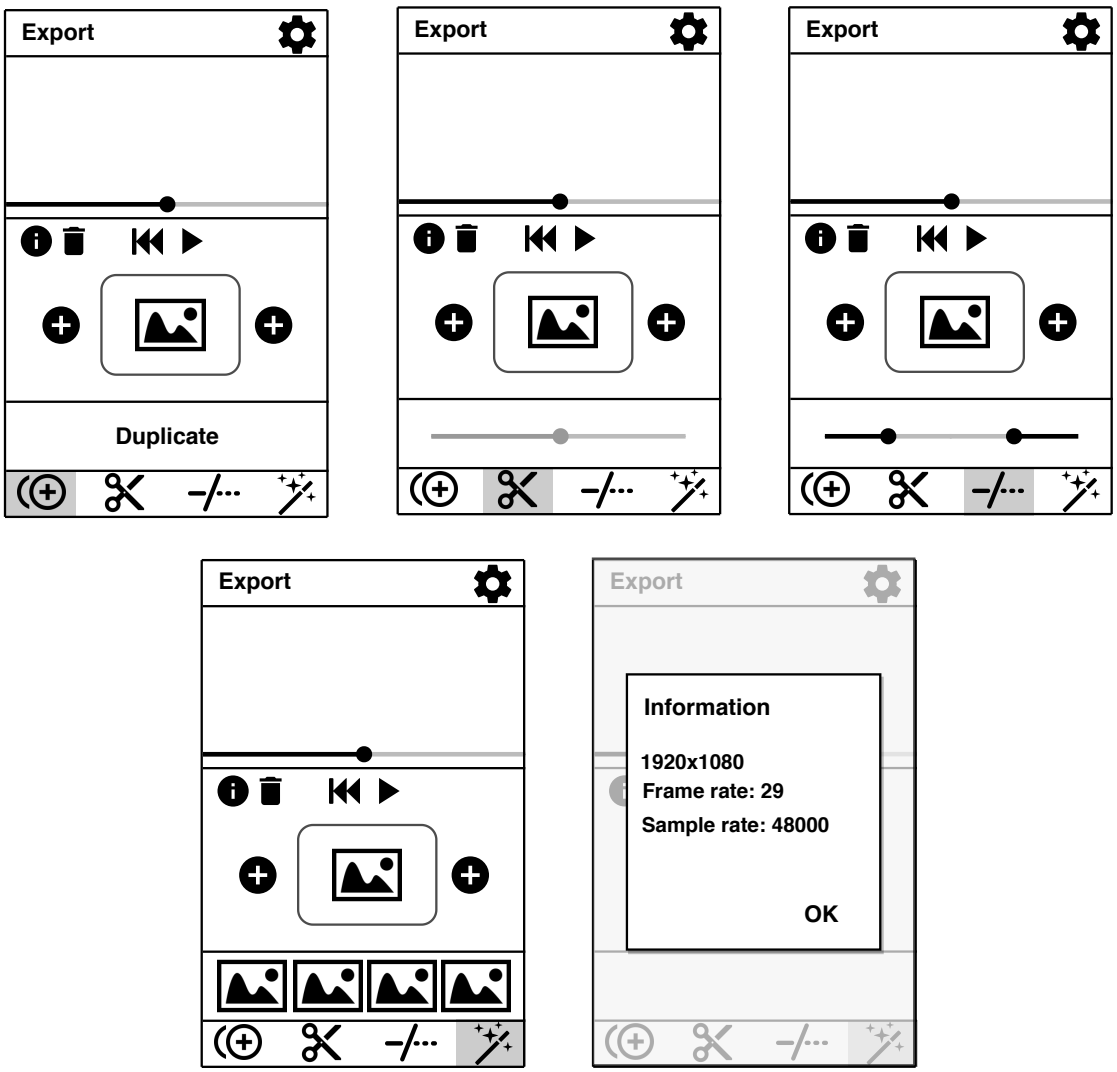
Obrázek 7.3: Obrazovka zobrazující seznam projektů

Pod horní navigací se bude nacházet náhled videa. V něm bude uživateli zobrazováno upravené video před jeho exportováním. Díky tomu nebude muset uživatel video exportovat jenom kvůli tomu, aby viděl výsledek aplikování obrazového efektu. Dále se zde bude nacházet posuvník, jímž se bude moci uživatel posouvat v čase aktuálně přehrávaného videa. Uživatel bude mít možnost ho skrýt či znovu zobrazit kliknutím na náhled videa. Přímo pod náhledem videa se bude nacházet tlačítko umožňující zastavit či spustit přehrávání videa.

Seznam videí, který se bude nacházet pod náhledem, bude kromě jednotlivých videí obsahovat tlačítka umožňující přidání nového videa na konkrétní místo seznamu. U každého videa bude zobrazen náhledový snímek, který poskytne uživateli lepší přehled o tom, jak bylo video oříznuto a o jaké video se přímo jedná. V seznamu videí se bude moci uživatel jednoduše posouvat a během přehrávání bude seznam posouván automaticky. Důvodem je, aby měl uživatel přehled o tom, kde v seznamu se aktuálně přehrávané video nachází.

Po výběru videa bude zastaveno přehrávání a uživateli bude zobrazena spodní navigace, kterou lze vidět na obrázku 7.4. Jejím účelem bude zajištění snadné dostupnosti jednotlivých operací. Pomocí ní bude moct uživatel jednoduše přepínat mezi panely, které budou poskytovat možnost duplikování, rozdělení či oříznutí videa. Posledním z panelů bude panel obsahující seznam náhledových snímků po aplikování obrazových efektů. Z důvodu toho, aby nebyl uživatel rozptylován, bude navigace i s panely během přehrávání skryta.

Vybrané video bude moci uživatel jednoduše přesunout pomocí operace `drag and drop`[2]. Po vybrání videa budou uživateli kromě spodní navigace zobrazeny další tlačítka umožňující zobrazit informace o zvoleném videu nebo ho odstranit ze seznamu.



Obrázek 7.4: Obrazovka editoru po výběru konkrétního videa.

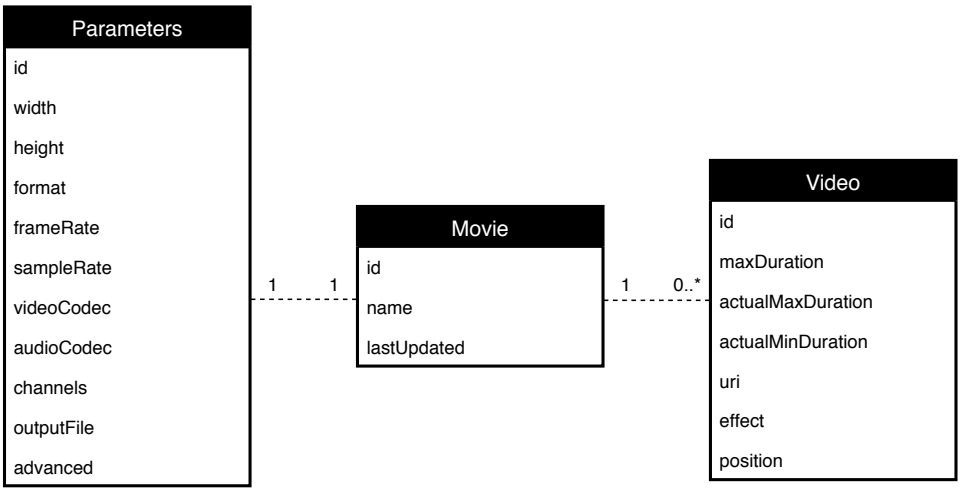
Přestože bylo rozmístění ovládacích prvků voleno již během návrhu s ohledem na cílovou skupinu, ukázalo se rozmístění některých z nich jako nevhodné a bylo v průběhu implementace a testování upraveno. Více informací k výslednému návrhu poskytuje podkapitola 8.2.1.

[2]Operace během níž uživatel uchopí konkrétní video a přetažením ho přesune na jiné místo.

7.3 Návrh struktury databáze

Pro uchovávání perzistentních dat aplikace byla pro svou komplexnost zvolena databáze. Její strukturu budou tvořit 3 tabulky, ve kterých budou ukládána data jednotlivých projektů. Detailnější popis tabulek se nachází pod ER diagramem 7.5, jenž znázorňuje jednotlivé tabulky databáze.

Z důvodu následného publikování zdrojového kódu na platformu GitHub jsou tabulky i jejich atributy pojmenovány anglicky, jelikož se na platformě vyskytují i vývojáři, kteří nemluví česky. Diagram obsahuje rovněž atributy, které byly doplněny až v průběhu implementace.



Obrázek 7.5: ER Diagram databáze video editoru.

- **Tabulka Movie**
  Tabulka bude reprezentovat jeden vytvořený projekt. Kromě atributu `id`, jenž je primárním klíčem s vlastností `auto-increment`[3], obsahuje tabulka atribut `name`, který bude sloužit pro uchování názvu projektu. Posledním atributem je pak `lastUpdated`, který bude obsahovat datum poslední editace projektu. Jeho hlavním účelem bude možnost přehlednějšího zobrazení seznamu projektů.

- **Tabulka Parameters**
  Tabulka bude obsahovat výstupní parametry každého projektu. Mohla by být sloučena s tabulkou `Movie`, ale z toho důvodu, že k tabulce `Movie` bude přistupováno neúměrně častěji, jsou tabulky navrženy samostatně. Díky tomu bude zvýšena rychlost zápisu a čtení do obou tabulek. Rovněž tato tabulka má primární klíč v podobě atributu `id`. Výstupní rozlišení bude ukládáno v atributech `width` a `height`, formát v atributu `format`, snímkovací frekvence v atributu `frameRate`, vzorkovací frekvence v atributu `sampleRate`, video kodek v atributu `videoCodec`, audio kodek v atributu `audioCodec`, počet audio kanálu v atributu `channels` a výstupní soubor v atributu `outputFile`. Pomocí atributu `advanced` bude ukládána informace o tom, zda se uživatel nepřepnul do rozšířeného módu zadávání parametrů.

- **Tabulka Video**
  Tabulka bude obsahovat informace spojené s videi přidanými k projektům. Kromě

[3] Automatické generování jedinečného čísla po přidání záznamu do tabulky.

primárního klíče v podobě atributu `id` obsahuje tabulka atribut `maxDuration`, ve kterém bude kvůli snížení četnosti volání knihovny ukládána původní délka videa a atributy `actualMaxDuration` a `actualMinDuration`, pomocí kterých bude ukládána upravená délka videa. V tabulce se dále nachází atribut `uri`, který bude sloužit pro specifikaci zdroje multimediálního obsahu, atribut `effect`, který bude poskytovat informaci o zvoleném obrazovém efektu a atribut `position`, který bude uchovávat informaci o umístění videa v seznamu videí.

7.4 Návrh operací s videem

Původní myšlenkou byla existence pouze jediné třídy `VideoProcessor`, která by poskytovala veškerou funkcionalitu spojenou s videem. Třída měla využívat pouze knihovnu `JavaCV`. Tato volba se ale při experimentech ukázala jako nevhodná, protože knihovna neumožňovala jednoduše spojit několik videí s různou snímkovací frekvencí s tím, že by byla zachována jejich původní délka. Z toho důvodu bylo nutné navrhnout proces exportování jinak.

Místo zmíněné knihovny byla pro proces exportování vybrána knihovna `MobileFFmpeg`, která umožňuje vykonání příkazů nástroje `ffmpeg`. Ukázku příkazů obsahuje již podkapitola 3.2.2. Pro jejich generování byla navržena třída `CommandGenerator`, která bude pro vstupní parametry generovat odpovídající příkazy, které bude možné vykonat pomocí nástroje `ffmpeg`.

Kromě zmíněných tříd byla navržena třída `Effects`, která bude umožňovat přidání nového efektu pomocí jediného řádku kódu. Pro přidání efektu bude stačit filtr nástroje `ffmpeg` a filtr knihovny `ExoPlayerFilter`, která bude zajišťovat aplikaci efektu na video během přehrávání a je dále zmiňována v podkapitole 8.1.1.

Pro ostatní operace se ukázala knihovna `JavaCV` za vyhovující. Bylo ale potřeba brát v potaz to, že při prvním zavolání knihovny probíhá načítání jejich součástí napsaných v nativním kódu. Výsledkem toho může být až 4 sekundy trvající odezva. Z toho důvodu byla navržena `loading` obrazovka. Ta bude zobrazována ještě před seznamem projektů a během jejího zobrazení budou načteny součásti knihovny.

7.4.1 Proces exportování

Exportování projektu bude probíhat v několika krocích. Nejprve budou pro vstupní parametry vygenerovány příkazy nástroje `ffmpeg`. V následujícím kroku budou příkazy samostatně vygenerované pro každé video vykonány. V posledním kroku bude vykonán příkaz, kterým budou všechna videa spojena. Pokud během procesu exportování dojde k přerušení, budou všechny dočasné soubory smazány a bude zastaveno vykonávání aktuálního příkazu.

7.4.2 Časové optimalizace

Experimenty s knihovnou `JavaCV` bylo zjištěno, že proces získání náhledového snímku videa trvá několikanásobně déle než aplikování obrazového efektu. Z toho důvodu byla navržena optimalizace v podobě ukládání získaných náhledových snímků do struktury `Bitmap`. Díky ní nebude trvat načtení seznamu s náhledovými snímky po aplikování efektů i několik sekund, ale bude trvat v řádu milisekund.

7.5 Návrh testování

V dnešní době existuje již celá řada podobných aplikací. Je tak důležité zjistit jakým způsobem aplikace obstojí při testování použitelnosti v porovnání s již existujícími aplikacemi, protože hlavním cílem je hlavně to, aby byla aplikace používána.

Testování bude probíhat na cílové skupině uživatelů a bude pro něho využit prototyp poskytující základní navrženou funkcionalitu. Uživatelé budou postupovat podle předem vytvořeného scénáře, který je složen ze sady úkolů. Ty jsou vybrány s ohledem na proveditelnost ve všech srovnávaných aplikacích, kterými bude vyvíjená aplikace, aplikace *Quick*[4] a aplikace *WeVideo*[5]. Během testování bude zaznamenáváno jejich chování a následně po jeho ukončení bude zjištěna zpětná vazba pomocí připravených otázek.

Sada úkolů vykonávaných během testování

- **Úkol 0** - Ujistěte se, že máte v telefonu přinejmenším 2 videa. Alespoň jedno z nich s minimální délkou 20 minut.
- **Úkol 1** - Vytvořte nový projekt.
- **Úkol 2** - Přidejte do vytvořeného projektu 2 videa. Jedno s minimální délkou 20 minut.
- **Úkol 3** - Ořízněte 20 minutové video na 10 vteřin.
- **Úkol 4** - Upravte výstupní rozlišení.
- **Úkol 5** - Přesuňte video z konce seznamu na začátek.
- **Úkol 6** - Exportujte projekt.
- **Úkol 7** - Smažte projekt.

Otázky dotazované po ukončení testování

- **Otázka 1** - Na které aplikaci se vám provádění úkolů zdálo nejvíce intuitivní.
- **Otázka 2** - Které úkoly se vám řešily lépe pomocí již existujících aplikací a které pomocí vyvíjené aplikace?
- **Otázka 3** - Dělalo vám problém řešení některého ze zadaných úkolů během testování vyvíjené aplikace?

Návrh testování čerpá z článku doktorky Amy Schade [20], který se věnuje hodnocení konkurenční použitelnosti, a z prezentace agentury Aitom [1], jenž obsahuje informace k uživatelskému testování.

[4] Viz https://play.google.com/store/apps/details?id=com.stupeflix.replay
[5] Viz https://play.google.com/store/apps/details?id=com.wevideo.mobile.android

Kapitola 8

Implementace

Tato kapitola seznámí s nejdůležitějšími částmi implementace. Na začátku kapitola představí, některé ze zvolených technologií a postupů. Poté se kapitola zaměří na implementaci uživatelského rozhraní, přičemž seznámí se změnami, které nastaly oproti původními návrhu. V další části se bude věnovat způsobu implementace exportování videa. Nakonec seznámí s výsledky testování, jenž bylo navrženo v předcházející kapitole.

8.1 Použité technologie a postupy

Následující podkapitola představí technologie a postupy, které byly použity během implementace.

8.1.1 Knihovny pro práci s videem

Pro zajištění potřebné funkcionality bylo použito několik knihoven. Cílem následující sekce je s nimi seznámit. Informace o knihovně Data Binding a třídě Live Data pocházejí z oficiální dokumentace [8].

JavaCV

Knihovna JavaCV byla vybrána pro zajištění většiny funkcionality spojené s manipulací s video soubory. Jedná se například o získávání náhledového snímku či informací o videu. Z důvodu, jenž byl zmiňován již v návrhu, nebyla zvolena pro exportování.

MobileFFmpeg

Knihovna MobileFFmpeg byla vybrána z toho důvodu, že umožňuje vykonávání příkazů nástroje `ffmpeg` na platformě Android. Poskytuje tak možnost implementovat proces exportování způsobem, jakým byl navržen v předcházející kapitole.

ExoPlayerFilter

ExoPlayerFilter [24] je knihovna umožňující aplikaci efektů na video přehrávané pomocí přehrávače `ExoPlayer`, který byl vybrán pro zobrazování náhledu videa. Pro aplikaci efektů knihovna využívá *shadery* knihovny *OpenGL*. Obrazové efekty jsou tak aplikovány pomocí GPU. Díky tomu je přehrávání videí s vyšším rozlišením plynulejší než v případě využití CPU, které by bylo zbytečně zatěžováno (viz kapitola věnující se testování výkonu

v publikaci *Android on x86*[14]). Pro zasvěcení do problematiky použití knihovny *OpenGL* je vhodný web *Learn OpenGL*[1].

8.1.2 Zvolená architektura

Z hlediska návrhu lze vytvořit aplikaci několika způsoby. Prvním z nich je vytvoření pouze za pomocí aktivit a fragmentů, které kromě zobrazování informací a reakcí na uživatelskou interakci, obsahují aplikační logiku. Výsledkem je aplikace, která je obtížně rozšiřitelná, špatně čitelná a testovatelná pouze s obtížemi. Oddělení aplikační logiky umožňují architekturní vzory. Nejoblíbenějšími vzory pro návrh aplikací na Android jsou **MVC** (*Model-View-Controler*), **MVP** (*Model-View-Presenter*) a **MVVM** (*Model-View-ViewModel*). Z nich byl pro implementaci vybrán návrhový vzor MVP [16] [4].

Model-View-Presenter

MVP je návrhový vzor, který vychází ze staršího vzoru MVC. V porovnání s ním nabízí jednodušší vytváření *unit testů*[2] či větší flexibilitu. Stejně jako MVC je složen ze třech částí. Těmi jsou:

- **Model** - jeho úkolem je práce s daty a zajištění business logiky.
- **View** - nejčastěji reprezentováno aktivitou či fragmentem. Jeho úkolem je zobrazovat informace uživateli a přesouvat do presenteru veškerou uživatelskou interakci (například stisknutí tlačítka).
- **Presenter** - tvoří mezivrstvu mezi *Modelem* a *View*. S *View* není připojen přímo, ale komunikace probíhá prostřednictvím rozhraní. Jeho hlavním úkolem je reagovat na uživatelskou interakci či případně komunikovat s *Modelem*.

8.1.3 Data Binding

Knihovna Data Binding[3], která je součástí skupiny komponent Android Jetpack, umožňuje deklarativní propojení prvků uživatelského rozhraní s daty. Výsledkem jejího použití je čitelnější a v první řadě jednodušší kód. Přenáší část logiky aktivity do souboru rozložení. V ukázce na obrázku 8.1 lze vidět množství zbytečného kódu, které je možné díky knihovně zredukovat.

```
1          TextView textView = findViewById(R.id.movie_title);
2          textView.setText(movie.getTitle());
```

```
1         <TextView android:text="@{movie.title}" />
```

Obrázek 8.1: Ukázka kódu bez a s využitím knihovny Data Binding.

[1] Viz https://learnopengl.com/
[2] Unit testy (jednotkové testy) slouží pro automatické testování jednotlivých částí implementace.
[3] Viz https://developer.android.com/topic/libraries/data-binding

8.1.4 Live Data

Live Data[4] je třída, která je součástí kolekce Android Jetpack. Je navržená k uchovávání dat, jejichž změny lze pozorovat. Nejedná se ale o klasickou observable[5] třídu. Na rozdíl od ní je *lifecycle-aware*. To znamená, že respektuje životní cyklus jiných komponentů, jako je aktivita nebo fragment. Díky čemu jsou nejsou upozorňovány neaktivní komponenty. Třídu Live Data je možno jednoduše kombinovat s knihovnou Data Binding.

8.1.5 DBFlow

DBFlow [11] je knihovna vytvářející abstrakční vrstvu nad *SQLite* databází. Její použití dokáže ušetřit velké množství času při vývoji. Příklad jejího použití byl uveden již v podkapitole 2.4. Pro implementaci byla vybrána s ohledem na velikost navržené aplikace a rychlost vykonávání operací[6].

8.2 Uživatelské rozhraní

Tato podkapitola je věnována uživatelskému rozhraní. To bylo implementováno podle návrhu z podkapitoly 7.2. V první části podkapitola seznamuje se změnami, které nastaly oproti původnímu návrhu v důsledku uživatelského testování. Poté ve druhé části blíže seznamuje s implementací finální verze uživatelského rozhraní.

8.2.1 Změny po uživatelském testování

Během uživatelského testování, kterému se bude více věnovat podkapitola 8.4, se ukázalo, že rozmístění ovládacích prvků v prototypu aplikace nebylo nejvhodnější. Uživatelské rozhraní bylo na základě získaných poznatků několikrát upraveno a následně znovu testováno. Nedostatky, které se vyskytovaly v dřívějších verzích, by tak již měly být vyřešené.

Při testování se ukázalo, že někteří uživatelé měli problém s umístěním a velikostí tlačítek sloužících pro smazání a editaci názvu projektu. Na základě těchto poznatků bylo tlačítko pro smazání nahrazeno *swipe* gestem a tlačítko pro editaci názvu bylo nahrazeno dlouhým podržením prstu na názvu projektu.

Další změnou je úprava tlačítka pro vytvoření nového projektu. To se podle uživatelů příliš nehodilo k celkovému designu aplikace. Na základě těchto připomínek bylo s designem aplikace více sladěno. Neméně důležitým požadavkem bylo přidaní času poslední editace projektu. Aplikace dovoluje vytvořit projekt s názvem již existujícího projektu, díky přidání času poslední editace je tak uživatel může jednoduše odlišit.

V návrhu obrazovky editoru nastalo také několik změn. Jednou z nich je úprava ovládacích prvků umožňujících přesný výběr času oříznutí videa. Tlačítka, kterými si mohl uživatel přiblížit komponentu `RangeSeekBar`[7], byly nahrazeny dialogem, jenž nabízí přesný výběr času pomocí komponentu `NumberPicker`[8], který umožňuje výběr čísla z předdefinovaného rozsahu.

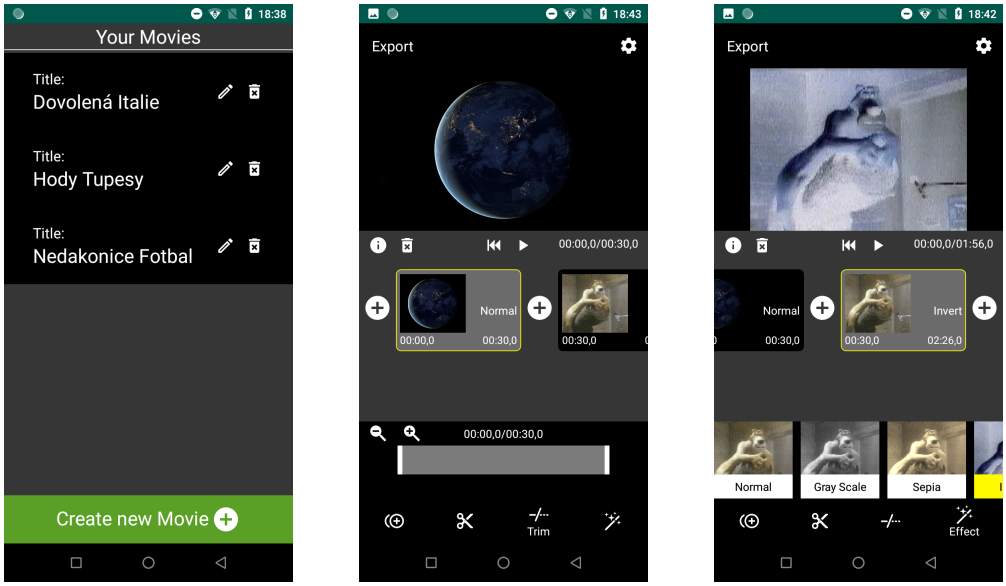
[4] Viz https://developer.android.com/topic/libraries/architecture/livedata

[5] Observer je návrhový vzor využívaný v situaci, kdy má pozorovaný objekt upozorňovat na změny své pozorovatele.

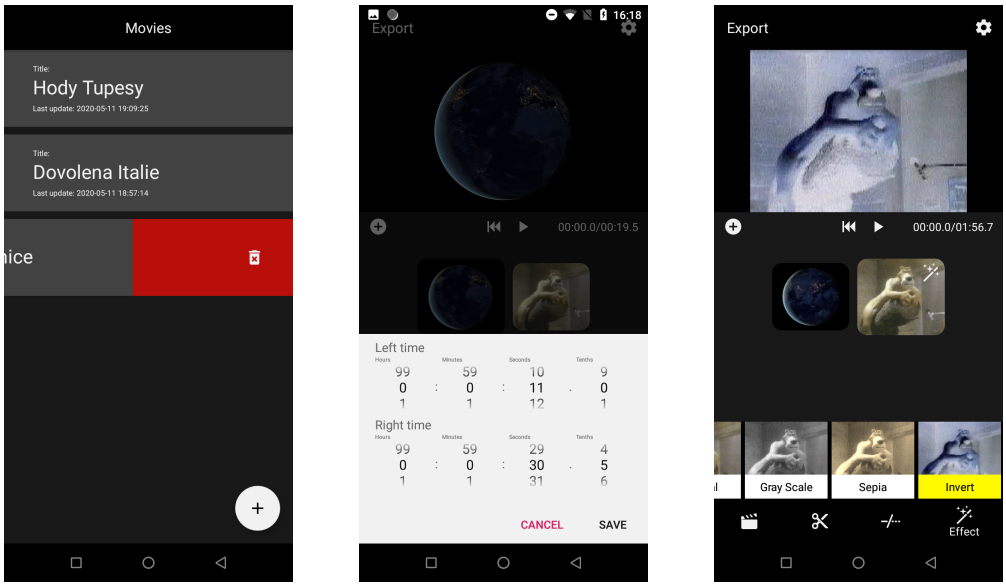
[6] Srovnání výkonnosti databázových knihoven: https://github.com/AlexeyZatsepin/Android-ORM-benchmark

[7] Viz https://github.com/Jay-Goo/RangeSeekBar

[8] Viz https://developer.android.com/reference/android/widget/NumberPicker



Obrázek 8.2: Původní verze uživatelského rozhraní



Obrázek 8.3: Finální verze uživatelského rozhraní

Původní návrh uživatelského rozhraní, jenž lze vidět na obrázku 8.2, byl kromě zmíněných úprav dále modifikován. Některé komponenty byly ve finálním návrhu 8.3 přesunuty či úplně odstraněny. Výsledkem úprav je minimalistická aplikace zachovávající si původní funkcionalitu.

8.2.2 Implementace uživatelského rozhraní

Ukázku finální verze uživatelského rozhraní aplikace lze vidět na obrázku 8.3. Na prvním snímku je vyobrazeno smazání projektu. To bylo implementováno vytvořením třídy `SwipeToDeleteCallback`, která dědí od třídy `ItemTouchHelper.Callback`.

`ItemTouchHelper`[9] je spjat s komponentou `RecyclerView`, v níž jsou zobrazeny všechny projekty. Rozšiřuje ji o možnost použití *swipe* gest a operace *drag and drop*. Při implementaci bylo využito gesta *swipe* doleva. Pokud je zaznamenán pohyb doleva, tak je místo projektu, jenž představuje jednu položku seznamu, zobrazena ikona koše nad červeným podkladem. Pokud si uživatel své rozhodnutí rozmyslí, může jednoduše operace smazání projektu zanechat. Pro smazání musí být položka přetažena minimálně ze 70%. Tato hodnota byla zvolena po experimentu s uživateli, při kterém se ukázalo, že na telefonech s velkým displejem uchopených v jedné ruce, nebylo úplné přetažení nikterak jednoduché. Po smazání projektu je na několik sekund zobrazena komponenta `Snackbar`[10], jenž poskytuje možnost vrácení smazaného projektu, pokud si uživatel své rozhodnutí rozmyslí nebo projekt smaže omylem. Sekvenci akcí lze vidět na obrázku 8.4.

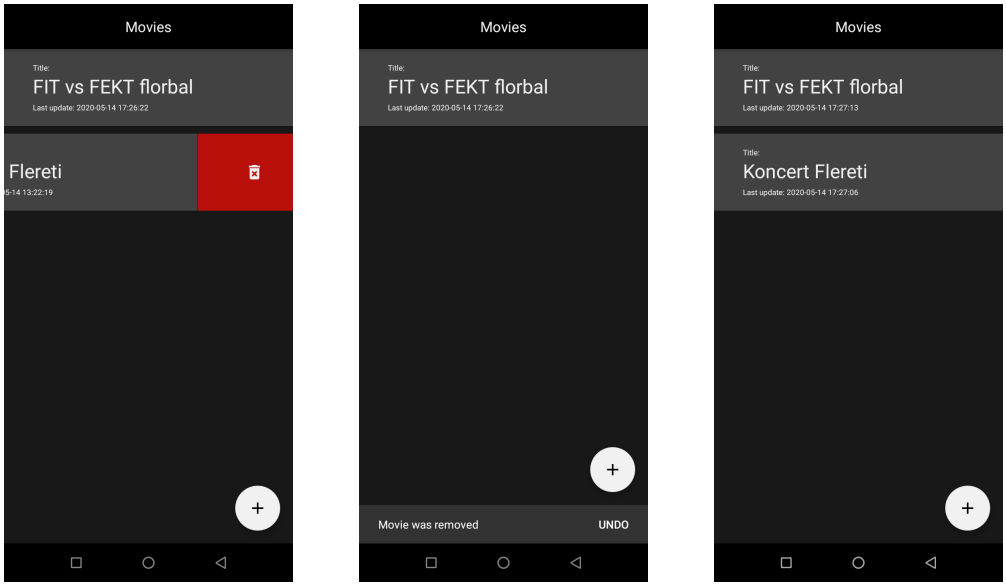
Po výběru projektu se zobrazí obrazovka editoru, ve které má uživatel možnost upravit přidaná videa. Před samotným zobrazením videí v komponentě `RecyclerView` probíhá prostřednictvím navržené třídy `VideoProcessor` proces získání náhledových snímků. Pro jejich získání je využita knihovna `JavaCV`, která při prvním použití třídy `AndroidFrameConverter` alokuje zdroje potřebné pro převedení struktury `Frame` na strukturu `Bitmap`, která je dále využívána. Kvůli tomu je doba prvního načtení projektu vždy delší. V průběhu procesu získání náhledových snímku je uživateli zobrazena animace načítání, která je implementována pomocí komponenty `ProgressBar`[11].

Přesunutí videa pomocí operace *drag and drop* bylo implementováno s využitím již zmiňované třídy `ItemTouchHelper`. Při testování se ukázalo, že výchozí nastavení akcelerace táhnutí položky seznamu nebylo příliš vhodné. Pokud se v seznamu nacházelo větší množství videí, dělalo uživatelům problém položku umístit na přesné místo. Po úpravě je po 500 milisekundách dosažena maximální rychlost táhnutí, která je dále konstantní.

Pokud se uživatel rozhodne smazat z paměti telefonu video, které je přidáno do konkrétního projektu, mohlo by to způsobit pád aplikace. Proto byl navržen snadno implementovatelný způsob kontroly. Ta probíhá v třídě `EditorPresenter`, která implementuje rozhraní `LifecycleObserver`, které třídě umožňuje reagovat na změny životního cyklu aktivity. Samotná kontrola je vykonávána prostřednictvím metody `contentUriExists()` třídy `ContentUriUtils` před procesem získání náhledových snímků a při návratu aktivity na popředí. Zmíněná metoda ověří, jestli nebylo video smazáno či přesunuto.

Spodní navigace, jenž je zobrazována během úpravy videa, byla implementována s využitím komponenty `BottomNavigationView`, jejíž menu může obsahovat až 5 položek. S její pomocí je uživateli umožněno přepínat mezi panely, které poskytují možnosti úpravy videa. Každý panel je reprezentován jedním fragmentem. Fragmenty jsou do aktivity přidávány dynamicky pomocí třídy `FragmentManager`, která umožňuje přidávat, odstraňovat či nahrazovat fragmenty nacházející se v kontejneru typu `FrameLayout`[12], jenž je vhodnou volbou pro zobrazení jednoho `view`[13].

[9] Viz https://developer.android.com/reference/androidx/recyclerview/widget/ItemTouchHelper
[10] Viz https://developer.android.com/reference/com/google/android/material/snackbar/Snackbar
[11] Viz https://developer.android.com/reference/android/widget/ProgressBar
[12] Viz https://developer.android.com/reference/android/widget/FrameLayout
[13] Viz https://developer.android.com/reference/android/view/View



Obrázek 8.4: Sekvence akcí od smazání po vrácení projektu.

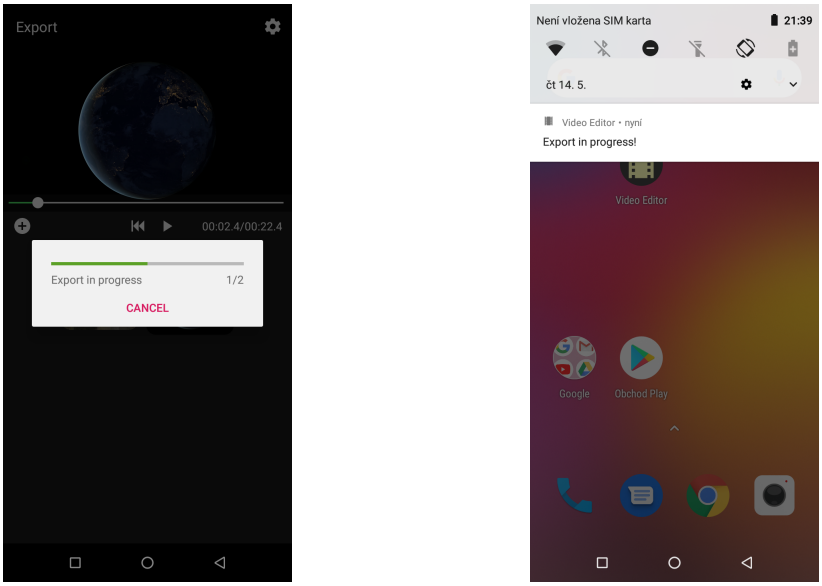
8.3 Implementace exportování

Jak již bylo napsáno v podkapitole 7.4, pro implementaci procesu exportování bylo možné využít jak knihovny `JavaCV` tak knihovny `MobileFFmpeg`. Knihovna `JavaCV` je vyhovující pokud nemají spojované videa rozdílnou snímkovací frekvenci. V případě jejího využití by musel být proces duplikování a zahazování snímků implementován ručně. Využitím knihovny `MobileFFmpeg` se lze ruční implementaci vyhnout. Knihovna nabízí možnost vykonání příkazů nástroje `ffmpeg`, který provede duplikaci nebo zahození snímků automaticky.

Proces exportování je implementován v třídě `VideoProcessor`, jak bylo navrženo v sekci 7.4.1. Pro vstupní parametry je pomocí metody `generateCommand()` třídy `CommandGenerator` pro každé video vygenerován příkaz, jenž je v následujícím kroku vykonán. Poté co jsou všechna videa samostatně exportována, lze provést spojení videí. Při implementaci je využit způsob spojení, o němž se již zmiňuje sekce 3.2.2. Další možné způsoby lze nalézt v dokumentaci[14]. Po vykonání příkazu, jenž spojí videa, je adresář se všemi dočasnými soubory smazán.

Proces exportování je výpočetně náročná operace trvající až několik minut. Z toho důvodu byla implementována třída `ExportService`, jenž dědí od třídy `Service`. Tato třída umožňuje vykonávat dlouhotrvající operace na pozadí a není pozastavena ani poté, co uživatel opustí aplikaci.

[14] Viz https://trac.ffmpeg.org/wiki/Concatenate



Obrázek 8.5: Na obrázku vlevo lze vidět dialog, který zobrazuje postup v probíhajícím procesu exportování. Na obrázku vpravo lze vidět notifikaci, která je uživateli zobrazena během procesu exportování. Díky ní uživatel ví, jestli byl projekt exportován či ještě ne.

8.4 Výsledky testování

Následující sekce obsahuje shrnutí výsledků testování funkčnosti aplikace a uživatelského testování.

8.4.1 Výsledky uživatelského testování

Po implementování prvního funkčního prototypu bylo navržené rozhraní testováno na uživatelích, jak bylo navrhnuto v podkapitole 7.5. Získané poznatky byly po testování vyhodnoceny a některé z nich použity v dalším návrhu uživatelského rozhraní. Většina uživatelů během testování označila vyvíjenou aplikaci za nejvíce intuitivní i přes nevhodné rozmístění některých prvků v původním návrhu. Jejich výpověď mohla být ovšem mírně zkreslená tím, že editory, s kterými byla aplikace porovnávána, obsahují více funkcionality.

V průběhu vývoje proběhly celkem 4 iterace, během kterých byla funkcionalita aplikace vylepšována a testována na uživatelích.

8.4.2 Výsledky testování funkčnosti

Proces exportování byl otestován na několika projektech. Každý projekt se skládal z videí, které měly rozdílný formát, rozlišení a snímkovací frekvenci. Testování bylo provedeno pro všechny formáty a rozlišení, které může uživatel zvolit. Pokud se uživatel přepne do pokročilého módu, tak může nastavit další parametry. Ty nejsou nijak kontrolovány a nebyly ani kvůli množství možných kombinací testovány.

Dobu exportování zde není vhodné uvádět z důvodu silné závislosti na výkonu zařízení.

Kapitola 9

Závěr

Cílem bylo vytvořit aplikaci umožňující střih videa a aplikování efektů. Aplikace byla během vývoje testována na cílové skupině uživatelů a na základě jejich podnětů iterativně vylepšována. Důvodem bylo vytvoření co nejintuitivnějšího uživatelského rozhraní přizpůsobeného rychlé práci s videem. Výsledná aplikace umožňuje stříhat videa, skládat je za sebe a aplikovat na ně efekty. Kromě toho aplikace nabízí možnost přehrávat upravená videa bez předchozího exportování.

Uživatel má při exportování možnost výběru z několika výstupních formátů a rozlišení. V rozšířeném módu má dále možnost volby snímkovací a vzorkovací frekvence. Vytvořené projekty jsou automaticky ukládány a uživateli je umožněno se k rozpracovaným projektům vrátit.

Na rozdíl od některých existujících řešení probíhá proces exportování přímo na mobilním zařízení, díky čemu není nutný přístup k internetu. To ovšem může přinést nevýhodu spojenou s vyššími nároky na výkon zařízení a spotřebou baterie.

V budoucnu se nabízí velký prostor pro rozšíření aplikace. Uživatelé již během implementace a po jejím dokončení přišli s mnoha nápady, jak by mohla aplikace ještě více usnadňovat editaci videa. Příkladem může být automatické sestavení výsledného videa ze zajímavých momentů. Podobnou funkcionalitu má již zmiňovaný Quick. Dalším příkladem pak může být podpora úpravy zvukové stopy a v neposlední řadě se nabízí rozšíření počtu obrazových efektů a podporovaných formátů.

Literatura

[1] Aitom Digital. *Uživatelské testování krok za krokem* [online]. [cit. 2020-04-22]. Dostupné z: https://www.pojdmetestovat.cz/file/16.

[2] Audet, S. *JavaCV GitHub* [online]. [cit. 2020-04-22]. Dostupné z: https://github.com/bytedeco/javacv.

[3] Benedikt, J. *Multimediální přehrávač pro Android*. Brno, CZ, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/118224.

[4] Bilík, D. *Úvod do MVP na Androidu* [online]. Září 2016 [cit. 2020-04-22]. Dostupné z: https://www.ackee.cz/blog/uvod-do-mvp-na-androidu/.

[5] Fairchild, M. D. *Color appearance models*. John Wiley & Sons, 2013. ISBN 978-1-1199-6703-3.

[6] FFmpeg. *About FFmpeg* [online]. [cit. 2020-04-22]. Dostupné z: https://www.ffmpeg.org/about.html.

[7] Google Inc.. *Android.media Summary* [online]. [cit. 2020-04-22]. Dostupné z: https://developer.android.com/reference/android/media/package-summary.

[8] Google Inc.. *Developer Guides* [online]. [cit. 2020-03-29]. Dostupné z: https://developer.android.com/guide.

[9] Google Inc.. *ExoPlayer Documentation* [online]. [cit. 2020-04-22]. Dostupné z: https://exoplayer.dev/.

[10] Google Inc.. *MediaPlayer Overview* [online]. [cit. 2020-04-22]. Dostupné z: https://developer.android.com/guide/topics/media/mediaplayer.

[11] Grosner, A. *DBFlow Documentation* [online]. [cit. 2020-04-22]. Dostupné z: https://agrosner.gitbooks.io/dbflow/content/.

[12] Kai, C. *Fanplayer GitHub* [online]. [cit. 2020-04-22]. Dostupné z: https://github.com/rockcarry/fanplayer.

[13] Kilián, K. *Quik: stříhejte videa jako "jablíčkáři"* [online]. Květen 2016 [cit. 2020-04-22]. Dostupné z: https://www.svetandroida.cz/quik-strihejte-videa-jako-jablickari/.

[14] Krajci, I. a Cummings, D. *Performance Testing and Profiling Apps with Platform Tuning. In: Android on x86.* Apress, 2014. ISBN 978-1-4302-6130-8.

[15] Lacko, L. a Herodek, M. *Mistrovství - Android.* Computer Press, 2017. ISBN 978-80-251-4875-4.

[16] Maxwell, E. *MVC vs. MVP vs. MVVM on Android* [online]. Leden 2017 [cit. 2020-04-22]. Dostupné z: https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/.

[17] Michal Španěl, T. M. *Barvy a barevné modely* [online]. 2019 [cit. 2020-04-22]. Dostupné z: https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_barvy_print_rev2019.pdf.

[18] Michal Španěl, T. M. *Redukce barevneho prostoru* [online]. 2019 [cit. 2020-04-22]. Dostupné z: https://www.fit.vutbr.cz/study/courses/IZG/private/lecture/izg_slide_omezeni_barev_print_rev2019.pdf.

[19] Rabetckiy, D. *A Single-Activity Android Application.* [online]. Leden 2019 [cit. 2020-04-29]. Dostupné z: https://medium.com/rosberryapps/a-single-activity-android-application-why-not-fa2a5458a099.

[20] Schade, A. *Competitive Usability Evaluations: Definition* [online]. Prosinec 2013 [cit. 2020-04-22]. Dostupné z: https://www.nngroup.com/articles/competitive-usability-evaluations/.

[21] Sener, T. *Mobile-FFmpeg GitHub* [online]. [cit. 2020-05-01]. Dostupné z: https://github.com/tanersener/mobile-ffmpeg.

[22] Smith, Z. *Convert images to grayscale and sepia tone using C* [online]. Srpen 2007 [cit. 2020-04-22]. Dostupné z: https://www.techrepublic.com/blog/how-do-i/how-do-i-convert-images-to-grayscale-and-sepia-tone-using-c/.

[23] Sohail, M. *How To Use Linux Terminal In Android* [online]. Dec 2019 [cit. 2020-04-22]. Dostupné z: http://www.linuxandubuntu.com/home/how-to-use-linux-terminal-in-android.

[24] Suda, M. *ExoPlayerFilter GitHub* [online]. [cit. 2020-04-22]. Dostupné z: https://github.com/MasayukiSuda/ExoPlayerFilter.

[25] Tom. *GiraffePlayer2 GitHub* [online]. [cit. 2020-04-22]. Dostupné z: https://github.com/tcking/GiraffePlayer2.

[26] wikisofia. *Reprezentace a vytváření barev* [online]. Květen 2017 [cit. 2020-04-22]. Dostupné z: https://wikisofia.cz/wiki/Reprezentace_a_vytvá%C5%99ení_barev.

Příloha A

Obsah CD

Přiložené CD obsahuje následující adresáře:

- src - zdrojový kód v podobě Android Studio projektu
- thesis - technická zpráva ve formátu PDF
- app - přeložená aplikace VideoEditor.apk
- latex - zdrojový kód technické zprávy