Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Práce s multimédii na platformě Android

Plzeň 2016 Marek Zimmermann

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 21. června 2016

Marek Zimmermann

Poděkování

Rád bych poděkoval Ing. Ladislavu Pešičkovi za trpělivost, vstřícný přístup, cenné rady a věcné připomínky, které mi během tvorby bakalářské práce poskytl.

Abstract

The topic of this bachelor thesis is exploring the state of multimedia support on Android platform, focusing on video and streaming support. The theoretical part deals with situation of the Android platform (general description, latest news, version labeling etc.), explains some basic multimedia terminology and then it describes the level of multimedia (pictures, audio and video) support in Android and how to work with them when programming applications for that platform. The last part contains information about streaming and libraries that can be used for extending multimedia support or streaming support of Android devices. The practical part describes an implementation of application used for streaming video and audio between two Android devices.

Abstrakt

Předmětem této bakalářské práce je prozkoumání možností práce s multimédii v operačním systému Android se zaměřením na video a streamování. Teoretická část se zabývá situací kolem platformy Android (obecný popis, novinky nejnovější verze, značení verzí atd.), vysvětluje základní pojmy v oblasti multimédií a poté se již zaměřuje na podporu a práci s multimédii (obrázky, audio a video) na platformě Android. V poslední části je pak popsáno streamování a knihovny rozšiřující multimediální podporu či usnadňující streamování mezi zařízeními s OS Android. Praktická část popisuje implementaci aplikace pro přenos videa a audia mezi dvěma zařízeními s OS Android.

Obsah

1 Úvod

Android je dnes nejpoužívanějším operačním systémem pro chytré mobilní telefony a tablety. Navíc jej můžeme najít i v další elektronice – v televizích, chytrých hodinkách, autech, herních konzolích, kamerách atd. Jednou z nejčastějších činností na tomto systému pak bývá mimo jiné konzumace různého audiovizuálního obsahu - poslech hudby, sledování filmů, video streamů z internetu atd. Účelem této práce je prozkoumat, jaké možnosti nabízí platforma Android při práci s multimédii včetně jejich streamování.

Teoretická část práce popisuje různé programátorské možnosti práce s videem, obrázky a audiem na platformě Android a také se zabývá streamováním těchto multimédií.

Praktická část poté zahrnuje tvorbu aplikace pro streamování videa mezi dvěma zařízeními s OS Android.

2 Android

Android je mobilní operační systém založený na jádře Linuxu, který je dostupný jako otevřený software (open-source)[1].[1]

Pro rozšíření funkcionality lze pro zařízení s Androidem naprogramovat různé aplikace. Primárním jazykem pro jejich programování je jazyk Java včetně jejích knihoven s výjimkou knihoven uživatelského rozhraní, které jsou nahrazeny vlastními knihovnami Androidu. Primární IDE pro vývoj aplikací je od prosince 2014 *Android Studio*, modifikovaná verze *IntelliJ IDEA*. [2]

Google používá dva způsoby číslování verzí Androidu. První obsahuje číslo označující majoritní verzi následované jedním nebo několika čísly minoritních verzí a poté jménem. Verze jsou odděleny tečkami, například *Android 6.0 Marshmallow*, *Android 4.0.4 Ice Cream Sandwich* atd. Toto číslování je vhodné pro člověka, neboť lze verze mezi sebou snadno odlišit, respektive porovnat.[3]

Druhý způsob zahrnuje číslování dle verze API spojené s danou verzí Androidu – čím vyšší číslo, tím novější verze, například: *API 16*, *API 22* atd. Toto značení je vhodnější z programového hlediska, neboť se lépe porovnává strojově.[4]

Zatím nejnovější verzí Androidu[2] je *Android 6.0 Marshmallow* (*API 23*) oficiálně představený společností Google 5. října 2015. Mezi nové funkce patří například přepracovaný systém udělování práv aplikacím, integrovaná podpora čteček otisků prstů, automatická záloha aplikací, nový RAM Manager, vylepšené fungování úložiště (zejména práce s externími úložišti) atd.[5]

Význam tohoto operačního systému lze navíc vidět při pohledu na podíl na trhu jednotlivých OS. Podle průzkumu společnosti IDC ve druhém čtvrtletí roku 2015 trh s mobilními telefony ovládá právě platforma Android s podílem přes 82%. [6]

[1] Některé části jsou uzavřenou technologii firmy Google, přesto Google tento systém prezentuje jako open-source.

[2] 14. prosince 2015

3 Práce s multimédii na platformě Android

Tato kapitola vysvětluje čtenáři některé ze základních pojmů z oblasti multimédií. Dále popisuje způsoby práce s multimédii na platformě Android (obsahuje popis dělený na práci s audiem, obrázky a videem) a v poslední části probírá streamování těchto multimediálních dat a využití externích knihoven pro rozšíření podpory multimédií či streamování.

3.1 Pojmy

Pro práci s multimédii je třeba nejprve vysvětlit několik pojmů.

Prvním z nich je tzv. *encoding* (česky kódování), což je proces převodu jedné formy dat do druhé. Existuje několik druhů kódování – kódovat můžeme obrázky, audio, video a znaky psaného textu. Tento proces se provádí zejména kvůli úspoře místa na datovém médiu – hlavním účelem bývá tedy tzv. *komprese*. Hlavními dvěma způsoby komprese dat jsou komprese ztrátová a bezeztrátová. U ztrátové komprese je cílem zbavit se nedůležité informace pro snížení výsledné velikosti daného záznamu[1] – příkladem můžou být formáty MP3, AAC, H.264 a další. Bezeztrátová komprese se snaží pokročilými technikami zmenšit výslednou velikost záznamu tak, aby se žádná zaznamenaná informace neztratila – příkladem mohou být formáty FLAC, RAR, LZMA a další.[7]

Multimediální formát lze tedy definovat jako konkrétní způsob, jakým se data enkódují.

Při volbě daného formátu pak rozlišujeme dvě části pro práci s daným multimediálním záznamem – kodér, který slouží pro zakódování dat pomocí zvoleného formátu a dekodér, který slouží pro jejich dekódování z daného formátu. Balíček těchto dvou částí se pak nazývá *kodek* (z anglického *codec*, pocházející ze slovního spojení „COder-DECoder“).[8]

Dalším pojmem popsaným v této podkapitole je tzv. *kontejner* (v angličtině

[1] Příkladem u hudby by bylo odstranění slabých tónů, které díky současně hrajícím silnějším tónům nemůže posluchač slyšet.

container). Ten lze popsat jako *wrapper* (česky „obal“) pro data kódovaná v různých formátech – jeho úkolem je tedy popsat, co je v něm uloženo a jakým způsobem. Příkladem těchto kontejnerů jsou například AVI, MKV aj.[9]

Praktickým příkladem popisujícím všechny výše zmíněné pojmy může být film v kontejneru MKV (soubor tedy bude s příponou „.mkv“) obsahující jednu video stopu kódovanou pomocí x264 (open-source implementace formátu H.264) a dvě audio stopy (český a anglický jazyk) kódované pomocí AAC.

Pro porovnání kvality je poté třeba znát především pojmy *rozlišení*, *datový tok*, *snímková frekvence* a *vzorkovací frekvence*.

Rozlišení (anglicky *resolution*) značí počet rozlišitelných pixelů v obou směrech obsažených ve video souboru. Zpravidla se tento údaj udává jako počet horizontálních pixelů krát počet vertikálních pixelů (například 1920×1080). Obecně platí, že čím větší rozlišení, tím více informací může video obsahovat. Pro správné zobrazení takového videa je pak také třeba vhodné zobrazovací zařízení, které dané rozlišení podporuje. Spolu s rozlišením se také udává tzv. *poměr stran* (anglicky *aspect ratio*, znázorňující poměr počtu horizontálních ku počtu vertikálních pixelů. Dnes nejpoužívanějším poměrem je poměr 16:9.[10]

Datový tok (anglicky *bitrate*) značí počet bitů přenesených za jednotku času. V případě multimediálních souborů jsou jedním z nejzásadnějších parametrů z hlediska kvality - vyšší bitrate umožňuje vytvoření kvalitnějších multimédií[2], zpravidla za cenu větší velikosti výsledného souboru. U multimediálních souborů pak rozeznáváme rovněž tzv. konstantní bitrate (CBR, anglicky *constant bitrate*), tedy stejný bitrate po celou dobu práce s multimédiem, nebo variabilní bitrate (VBR, anglicky *variable bitrate*), značící proměnlivý bitrate v průběhu vytváření či přehrávání multimediálního souboru.[11]

Snímková frekvence (anglicky *framerate*) značí počet snímků za sekundu. Obecně platí, že čím vyšší počet snímků, tím plynulejší „animace“ zachycená na daných snímcích. U současné filmové produkce se nejčastěji používají frekvence 24 či 30 snímků za sekundu, většina současných displejů podporuje snímkové frekvence do hodnoty 60 snímků za sekundu[3].[12]

[2]V případě videa jde například o větší počet detailů. Naopak při příliš nízkém bitrate lze pozorovat chyby v obrazu – tzv. *artefakty*.

[3]Lze samozřejmě najít displeje schopné zobrazovat 144 snímků či více.

Vzorkovací frekvence je počet pořízených vzorků daného zvuku za sekundu popisující jeho digitální vyjádření. Čím vyšší daná frekvence je, tím přesnější digitální reprezentace daného zvuku může být.[13][14]

3.2 Audio

Tato podkapitola popisuje podporu různých audio formátů a poté práci s nimi při jejich přehrávání a nahrávání.

3.2.1 Audio obecně

Android je v základu velmi dobře připraven na práci s audiem.

Podpora konkrétních formátů je znázorněna v tabulce 3.1. Symbol „x“ značí podporu pro přehrávání daného formátu (ve sloupci „Dekodér“) a podporu pro nahrávání nebo jinou manipulaci s daným formátem[4] (ve sloupci „Kodér“). Pokud není podpora daného formátu přítomna ve všech verzích Androidu, je v závorce uvedeno, od které verze je podpora daného formátu přítomna. Pro detailnější informace o podpoře formátů (velikosti datových toků, podporovaných kanálech apod.) viz developer.android.com.[15]

Formát *AAC LC* je formát, který byl vytvořen jako náhrada formátu *MP3*. Formát *HE-AACv1* je rozšířením AAC LC optimalizovaným pro využití v situacích s omezeným datovým tokem (jako příklad bývá uvedené streamování audia). *HE-AACv2* je pak vylepšenou verzí HE-AACv1.[16] Posledním podporovaným formátem z rodiny AAC je *AAC ELD*, který se používá v situacích s extrémně omezeným datovým tokem a potřebou co nejlepšího poměru kvalita/datový tok (typicky u hlasového přenosu).[17]

Formát *AMR-NB* byl vytvořen pro přenos hlasu v sítích GSM. Pracuje v rozsahu 200-3400 Hz s datovým tokem od 4,75 do 12.2 kbps. Je tedy nenáročný pro přenos, ovšem za cenu velmi nízké kvality zvuku.[18] Pro zlepšení kvality zvuku byl pak vyvinut formát *AMR-WB*, který pracuje na šířce pásma 50-7000 Hz a s datovým tokem od 6,6 do 23,85 kbps.[19]

[4]Do audio souboru nutně nemusíme chtít nahrávat – můžeme chtít například upravit již existující záznam pomocí nějakého efektu apod.

Formát/kodek	**Kodér**	**Dekodér**	**Podporované přípony souborů a kontejnery**
AAC LC	x	x	3GPP(.3gp), MPEG-4 (.mp4, .m4a), ADTS raw AAC (.aac), MPEG-TS (.ts)
HE-AACv1	x (4.1+)	x	
HE-AACv2	–	x	
AAC ELD	x (4.1+)	x (4.1+)	
AMR-NB	x	x	3GPP(.3gp)
AMR-WB	x	x	
FLAC	–	x (3.1+)	FLAC (.flac)
MP3	–	x	MP3 (.mp3)
MIDI	–	x	Type 0 and 1 (.mid, .xmf, .mxmf), RTTTL/RTX (.rtttl, .rtx), OTA (.ota), iMelody (.imy)
Vorbis	–	x	Ogg (.ogg), Matroska (.mkv, Android 4.0+)
PCM/WAVE	x (4.1+)	x	WAVE (.wav)
Opus	–	x (5.0+)	Matroska (.mkv)

Tabulka 3.1: Podpora audio formátů v Androidu

FLAC je otevřený bezeztrátový audio formát. Mezi bezeztrátovými formáty patří mezi nejrozšířenější, neboť má dobrý kompresní poměr, rychlé dekódování a je otevřený. Používá se zejména pro hudbu.[20]

Formát *MP3* patří mezi nejrozšířenější ztrátové formáty audia. Používá se především pro hudbu, neboť nabízí uspokojivou kvalitu při velmi malé velikosti souboru (v porovnání s bezeztrátovými formáty a nekomprimovaným audiem) a navíc jeho podporu obsahuje téměř každé zařízení schopné přehrávání hudby, chytré telefony s Androidem nevyjímaje.

MIDI formát je používaný pro uložení sekvence tónů. U těch je zaznamenána notace, výška a další parametry, podle kterých je pak možné přehrát daný soubor na jiném zařízení – i takovém, které má jiné HW vybavení[5]. Výhodou je velmi malá velikost souborů a snadná editace.[21]

Vorbis je ztrátový open-source formát konkurující formátům MP3, AAC

[5]Vidět to lze na starých telefonech s editorem vyzvánění ukládajícím v MIDI – přehrávaný MIDI soubor vydával na 2 různých telefonech jiné zvuky, melodie však zůstala zachována.

či WMA. Nabízí kvalitu podobnou zmíněným formátům, není však tolik rozšířen (ztrácí hlavně z hlediska podpory přehrávačů hudby).[22]

Proprietární bezeztrátový formát *WAVE* byl vyvinutý firmami Microsoft a IBM. Nejčastěji se používá jako formát skladující nekomprimovaná surová (anglicky *raw*) audio data. Formát samotný ale podporuje několik způsobů komprese ukládaného audia. Spíše než na hudbu se používá na ukládání zvuků[6].[23]

Opus je otevřený ztrátový kodek. Zvládá datové toky od 6 do 510 kbps, vzorkovací frekvence od 8 do 48 kHz, velikost rámců od 2,5 do 60 ms, zvládá konstantní i variabilní datové toky, podporuje až 255 kanálů a umí datový tok, vzorkovací frekvenci a velikost rámce dynamicky měnit za běhu. Díky tomu je vhodný pro streamování audia i jeho ukládání.[24]

3.2.2 Přehrávání audia

K přehrávání audia slouží v Androidu třída `MediaPlayer`.[32] Umí přehrávat audio jak z lokálních zdrojů a *URI* (získané například od *Content Resolver*u), tak i audio získané z externích *URL* (tedy například z audio streamů).

Pro přehrání tzv. „raw audio resource“ (zdroje audia, které nebyly žádným způsobem předzpracovány) lze použít metodu `create()` třídy `MediaPlayer` a poté jí spustit metodou `start()` – viz zdrojový kód 3.1.

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.
        sound_file_1);
mediaPlayer.start();
```

Zdrojový kód 3.1: Ukázka kódu pro přehrávání raw zdroje audia

Pro přehrávání audia z jiných zdrojů je postup znázorněn ve zdrojovém kódu 3.2. Po vytvoření instance třídy `MediaPlayer` jí musíme nastavit typ audio streamu přes metodu `setAudioStreamType()` na hodnotu `AudioManager.STREAM_MUSIC`. Poté nastavíme zdroj dat metodou `setDataSource()` (vstupem může být jak vnitřní URI, tak externí URL), audio připravíme metodou `prepare()` a poté můžeme spustit přehrávání metodou `start()` a zastavit metodou `stop()`.

[6]Například zvukové soubory u počítačových her bývají ukládány v tomto formátu.

Je třeba zmínit, že zpracování metody `prepare()` může zvláště u síťových zdrojů trvat velmi dlouho. Protože vše v rámci aktivity běží standardně v jednom vlákně, může toto být příčinou zaseknutí UI či dokonce způsobit ANR[7] chybu. Je tedy lepší použít metodu `prepareAsync()`, která přípravu na přehrání audia provede v jiném vlákně a odchytit dokončení procesu v metodě `onPrepared()` rozhraní `MediaPlayer.OnPreparedListener` a třídu implementující toto rozhraní pak nastavit instanci `MediaPlayer` pomocí metody `setOnPreparedListener()`. Důležitý je také fakt, že instance třídy `MediaPlayer` si může brát významné množství systémových zdrojů - měla by tedy být uvolněna, jakmile to bude možné, pomocí metody `release()`.

Android rovněž umožňuje přehrávat audio na pozadí (přehrávání pak pokračuje, i pokud uživatel zavře zdrojovou aktivitu) tak, že výše zmíněné kroky provede nikoliv v aktivitě, ale v tzv. službě – třída `Service`. Pro tu platí stejná pravidla ohledně zpracování v jednom vlákně jako u aktivity.

Pokud bychom chtěli například vytvořit aplikaci sloužící jako hudební přehrávač, je lepší toto audio přehrávat „na popředí“ (anglicky *foreground service*). Toto přehrávání běží (stejně jako služba na pozadí) i po zavření zdrojové aktivity, ale s tím rozdílem, že má právě oproti službám na pozadí vyšší prioritu danou systémem (služby na popředí bere jako služby z hlediska uživatele důležitější než ty na pozadí).

Rovněž lze implementovat přehrávání i po zhasnutí displeje či zamčení telefonu. K tomu slouží tzv. *wake locks*. S těmi je nutné nakládat opatrně, neboť jejich dlouhé držení zkracuje výdrž telefonu na baterii.

Při běhu se může stát, že v určitou chvíli bude chtít přehrát nějaké audio více aplikací (například pokud během přehrávání hudby chce jiná aplikace upozor-

[7] Application Not Responding – chyba nastávající v případě, kdy si systém myslí, že se aplikace zasekla a nabízí uživateli možnost ji ukončit.

```
String url = "http://www.zdroj.audia.cz...";
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare();
mediaPlayer.start();
```

Zdrojový kód 3.2: Ukázka kódu pro přehrávání audia z URL

nit uživatele na novou SMS) a uživatel by mohl některé tyto zvuky přeslechnout. Proto se při přehrávání zvuku v Androidu řeší ještě tzv. *audio focus* – aplikace se systému dotazuje (respektive od něj dostává informace), zdali může přehrávat zvuk. Je třeba podotknout, že aplikace může toto zcela ignorovat, snižuje se tím však uživatelská přívětivost – uživatel může přeslechnout důležité zvukové upozornění. Dotazování probíhá přes metodu `requestAudioFocus()` třídy `AudioManager`, pro zpracování změny daného stavu lze pak implementovat rozhraní `AudioManager.OnAudioFocusChangeListener` a jeho metodu `onAudioFocusChange`.

Pro detailní implementaci výše zmíněného viz developer.android.com.[33]

3.2.3 Nahrávání audia

Android umožňuje celkem snadné pořizování audia při použití třídy `MediaRecorder`. Ukázka přípravy a začátku nahrávání je znázorněna ve zdrojovém kódu 3.3. Instanci je třeba nastavit zdroj audia metodou `setAudioSource()` (v ukázce je to mikrofon telefonu), nastavit výstupní formát audio souboru metodou `setOutputFormat()` a jméno metodou `setOutputFile()`, nastavit formát nahraného audia metodou `setAudioEncoder()`, připravit se na nahrávání metodou `prepare()` a začít samotné nahrávání metodou `start()`. Přerušit toto nahrávání lze metodou `stop()` a rovněž je vhodné co nejdříve uvolnit zdroje zabrané instancí `MediaRecorder` metodou `release()`.

```
MediaRecorder mRec = new MediaRecorder();
mRec.setAudioSource(MediaRecorder.AudioSource.MIC);
mRec.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
mRec.setOutputFile("rec");
mRec.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
mRec.prepare();
mRec.start();
```

Zdrojový kód 3.3: Ukázka kódu pro nahrávání audia z mikrofonu telefonu do souboru rec.3gp

3.3 Obrázky

V podkapitole se nachází popis práce s obrázky – podporované formáty, jejich zobrazování a pořizování.

3.3.1 Obrázky obecně

Podpora konkrétních formátů je znázorněna v tabulce 3.2. Symbol „x“ značí podporu pro přehrávání daného formátu (ve sloupci „Dekodér“) a podporu pro nahrávání nebo jinou manipulaci s daným formátem (ve sloupci „Kodér“). Pokud není podpora daného formátu přítomna ve všech verzích Androidu, je v závorce uvedeno, od které verze je podpora daného formátu přítomna. Pro detailnější informace o podpoře formátů (velikosti datových toků, podporovaných kanálech apod.) viz developer.android.com.[15]

Formát/kodek	**Kodér**	**Dekodér**	**Podporované přípony souborů a kontejnery**
JPEG	x	x	JPEG (.jpg)
GIF	–	x	GIF (.gif)
PNG	x	x	PNG (.png)
BMP	–	x	BMP (.bmp)
WebP	x (4.0+)	x (4.0+)	WebP (.webp)

Tabulka 3.2: Podpora formátů obrázků v Androidu

Formáty obrázků lze rozdělit do dvou kategorií. První jsou tzv. bitmapové obrázky, kde je obrázek tvořen maticí pixelů s informací o barvě. Druhou jsou tzv. vektorové obrázky, kdy to, co vidíme, je uloženo jako seznam objektů, které jsou popsány pomocí různých křivek. Výhodou vektorových obrázků oproti bitmapovým bývá menší velikost souborů a větší možnosti škálování (zvětšování a zmenšování) obrázků bez viditelných deformací či ztráty kvality. Nevýhodami bývá nemožnost vykreslit některé komplexnější objekty a občas i doba vykreslování v případě některých složitých objektů. Skladovat obrázky jako vektorové se vyplácí v případě jednoduchých křivek či znaků (tedy například logo), naopak složitější objekty (například pořízené fotografie) je lepší skladovat jako bitmapové.

Formát *JPEG* je dnes nejpoužívanějším ztrátovým formátem na ukládání bitmapových obrázků. Používá se zejména pro ukládání fotografií, neboť jeho

komprese výrazně snižuje velikost obrázku a při rozumné úrovni komprese není lidským okem poznat ztráta informací způsobená kompresí. Podporuje až 24bitové barvy a maximální rozlišení 64Kx64K.[25]

GIF je bezeztrátový formát pro ukládání bitmapových obrázků. Jeho hlavním omezením je omezení na 8bitové barvy – tedy v jednom obrázku nemůže být více než 255 barev. Na internetu je však hojně používaný pro jeho podporu jednoduchých animací (formát ji vytváří tím nejjednodušším způsobem – má uložen všechny snímky a poté je jenom zobrazuje za sebou). Díky nízkému počtu barev a LZW algoritmu pro komprimaci mívá tento formát malou velikost – výjimku tvoří animace, které buď ukládají obrázky ve vysokém rozlišení, nebo jsou velmi dlouhé (nebo oboje).[26]

Jako náhrada byl vytvořen formát *PNG* – bezeztrátový formát, rovněž pro bitmapové obrázky. Podporuje až 48bitové barvy, maximální rozlišení až 2Gx2G pixelů, obsahuje rovněž informaci o *alpha* kanálu (s jehož pomocí lze vytvářet průhledné části obrázku) a data komprimuje metodou LZ77[8]. Formát GIF však ve výsledku nenahradil, neboť formát PNG nepodporuje žádný druh animací. Hodí se však pro obrázky, kde by se ztráta informací (například u JPEG komprimace) mohla negativně projevit na obrázku. Toto typicky nastává u obrázku s ostrými přechody – obrázky s textem, screenshoty (kopie stavu, který zobrazuje displej daného zařízení, uložená do obrázku) apod.[27]

Android rovněž obsahuje podporu pro zobrazení formátu *BMP* – formát pro ukládání bitmapových obrázků, rovněž bezeztrátový. Formát ukládá obrázek po jednotlivých pixelech, má podporu pro 1, 4, 8, 16, 24, a 32bitové barvy a nejnovější verze formátu podporuje rozlišení až 2Gx2G. Díky jednoduchosti formátu a nulovém zatížení patenty je tento formát velmi rozšířený a zvládá ho zobrazit i editovat velké množství nástrojů. Tento formát umí komprimovat data pomocí RLE komprese, ale většinou se používá jeho varianta bez komprese. Díky tomu bývají soubory s obrázky tohoto formátu velmi velké a nehodí se tak pro použití na internetu.[28]

Formát *WebP* poskytuje ztrátový i bezeztrátový přístup pro ukládání bitmapových obrázků. Vyvinula ho firma Google úpravou svého formátu *VP8* používaného pro ukládání videa. Formát podporuje průhlednost, je otevřený a snaží se konkurovat formátům PNG a JPEG.[29]

[8]Jedním z důvodů pro nahrazení GIF formátu byl fakt, že jeho LZW algoritmus pro komprimaci byl patentovaný.

Pro práci s obrázky se na platformě Android využívá třída `Bitmap`. Načíst data do instance této třídy lze pomocí třídy `BitmapFactory` několika způsoby:

- z pole bytů pomocí metody `BitmapFactory.decodeByteArray()`,
- z řetězce obsahujícího absolutní cestu k souboru obsahujícím obrázek pomocí metody `BitmapFactory.decodeFile()`,
- ze souboru metodou `BitmapFactory.decodeFileDescriptor()` získaného přes `FileDescriptor`
- z Resources metodou `BitmapFactory.decodeResource()`,
- ze streamu metodou `BitmapFactory.decodeStream()`

Pro zobrazování obrázků má pak Android i patřičné UI komponenty pro jejich zobrazení. Nejčastěji používaná komponenta pro zobrazení obrázků je `ImageView`.

3.3.2 Vytváření obrázků

V systému Android lze získání obrázku pořízeného kamerou telefonu zařídit dvěma způsoby:

- vyvoláním aktivity pomocí tzv. *intent* (česky „záměr“), jenž jako výsledek vrací kód popisující, zda se akce zdařila (viz dále),
- při práci s instancí třídy `Camera`[9] využijeme metody `takePicture()`, která nám vrátí pole bytů, což je právě námi pořízený snímek

Při vytváření snímku přes *intent* musíme jako parametr přibalit *URI* souboru, do kterého chceme vyfocený obrázek uložit. V metodě naší zdrojové aktivity `onActivityResult()` pak můžeme zkontrolovat, zda bylo pořizování snímku úspěšné (při neúspěchu lze navíc rozlišit i chybu od stornování akce uživatelem). Pro detailní ukázku viz developer.android.com.[34]

[9]Práce s kamerou je detailněji rozebrána v kapitole 3.4.3 na straně 15.

3.4 Video

V této podkapitole se nachází popis formátů pro práci s videem, popis přehrávání a nahrávání videa na platformě Android a rovněž se zde zmiňuje i práce s multimediálními soubory.

3.4.1 Video obecně

Podpora konkrétních formátů je znázorněna v tabulce 3.3. Symbol „x“ značí podporu pro přehrávání daného formátu (ve sloupci „Dekodér“) a podporu pro nahrávání nebo jinou manipulaci s daným formátem (ve sloupci „Kodér“). Pokud není podpora daného formátu přítomna ve všech verzích Androidu, je v závorce uvedeno, od které verze je podpora daného formátu přítomna. Pro detailnější informace o podpoře formátů (velikosti datových toků, podporovaných kanálech apod.) viz developer.android.com.[15]

Formát/kodek	**Kodér**	**Dekodér**	**Podporované koncovky a kontejnery**
H.263	x	x	3GPP (.3gp), MPEG-4 (.mp4)
H.264 AVC	x (3.0+)	x	3GPP (.3gp), MPEG-4 (.mp4), MPEG-TS (.ts)
H.265 HEVC	–	x (5.0+)	MPEG-4 (.mp4)
MPEG-4 SP	–	x	3GPP (.3gp)
VP8	x (4.3+)	x (2.3.3+)	WebM (.webm), Matroska (.mkv)
VP9	–	x (4.4+)	

Tabulka 3.3: Podpora video formátů v Androidu

Formát *H.263* byl vytvořen jakožto komprimační formát pro videokonference, kde bylo třeba nízkého datového toku. I přes rozšířenost jeho nástupce (*H.264*) je tento formát stále používán, například pro MMS. Důležité je zmínit, že H.263 se stará výhradně o video, audio tak musí být zpracováno jiným formátem.[30]

Formát *H.264* (také znám jako *MPEG-4 AVC*) je dnes nejrozšířenějším formátem pro práci s videem. Nabízí dobrý poměr kvality ku velikosti videa. Navíc tento formát podporuje drtivá většina videopřehrávačů.[31]

Novinkou je formát *H.265* (známý také jako *HEVC*), který je nástupcem

formátu H.264 a odpovědí na postupný nástup vyšších rozlišení v čele se 4K. V porovnání s předchozím formátem se tento snaží při videu o stejné kvalitě o snížení velikosti o zhruba polovinu, respektive o zachování velikosti při vyšší kvalitě či rozlišení videa. Formát zatím není příliš rozšířený, neboť je oproti předchozím dvěma uvedeným formátům relativně nový a je náročnější na zpracování.

Formát *MPEG-4 SP* (SP značí „Simple Profile“) se používal v případě, kdy bylo k dispozici málo zdrojů – pomalé připojení, nevýkonný HW apod. Příkladem můžou být například některé starší mobilní telefony, videokonferenční zařízení, kamery apod. Kvalita videa tedy není předností tohoto formátu. Dnes se s takovými zařízeními již moc nesetkáme, a tak je v případě potřeby pořízení nenáročného záznamu lepší použít formát H.263.

Formáty *VP8* a *VP9* jsou odpovědí společnosti Google na formát H.264. VP9 je nástupce formátu VP8, vyznačuje se vyšší efektivitou (u videa stejné kvality dosahuje VP9 o 50% menší výsledné velikosti) a rychlejším dekódováním. Důvodem postupného rozšíření těchto formátů je otevřenost těchto formátů – některé techniky formátu H.264 i jeho nástupce jsou patentované.

3.4.2 Přehrávání videa

Pro přehrávání videí v Androidu stačí využít UI komponentu `VideoView`, přes URI videa nastavit komponentě zdroj metodou `setVideoURI()` a spustit přehrávání metodou `start()`.

Protože však uživatel ve většině případů chce mít kontrolu nad tímto přehráváním, tj. chce mít možnost video pozastavit, přetočit, spustit atd., existuje zde třída `MediaController`, která do komponenty pro přehrávání videa přidá ovládací prvky – tlačítka pro přetočení, pauzu/přehrávání, posunutí videa a také posuvník pro skok na libovolnou část videa. Pro zakomponování nastavíme instanci `VideoView` controller metodou `setMediaController()` a instanci `MediaController` ovládaný prvek metodou `setAnchorView()`.

Druhým způsobem přehrávání videa je využití tříd `MediaPlayer`, `SurfaceHolder` a UI komponenty `SurfaceView`. Podrobnosti implementace lze vidět na code.tutsplus.com.[35]

Obrázek 3.1: Ukázka ovládacích prvků videa třídy `MediaController`, zdroj: code.tutsplus.com[35]

3.4.3 Práce s kamerou

Práce s kamerou v systému Android se podobá práci s daty, ke kterým může najednou přistupovat více procesů.

Mobilní zařízení s OS Android obsahují většinou dvě kamery – přední a zadní. Programově jsou odlišeny pomocí ID. Přednost mívá zadní kamera – typicky tedy mívá zadní kamera ID 0 a přední kamera ID 1. Není to ovšem pravidlem.

Pokud chceme s kamerou pracovat, musíme jí nejprve pro sebe „zabrat“. K tomu slouží metoda `open()` třídy `Camera`, jejímž parametrem může být ID kamery[10]. Tímto zabráním nám vznikne instance téže třídy. Pokud již s kamerou nechceme pracovat, můžeme uvolnit zdroje zabrané instancí pomocí metody `release()`.

Pokud chceme námi zabranou kameru použít (například k nahrávání videa), je třeba ještě před použitím kameru odemknout metodou `unlock()`, popřípadě zamknout metodou `lock()` (a umožnit tak dalším procesům ji využívat). Od API 14 (Android 4.0+) je proces zamykání a odemykání automa-

[10]Pokud ID není zadáno, zabere se první kamera – tedy ta s ID 0.

tický s výjimkou selhání přípravy procesu na nahrávání[11].

Od API 21 (Android 5.0 Lollipop) poskytuje pro práci s kamerou Google nové API `Camera2`. Cílem nového API bylo dát větší volnost ohledně samotného manuálního nastavení kamery a rovněž zpřehlednění práce s ní. Práce s kamerou byla rozdělena do těchto pěti tříd:

- `CameraManager` – slouží pro iterování, výpis a výběr kamery, se kterou chceme pracovat,
- `CameraDevice` – jedna instance reprezentuje jednu kameru, se kterou chceme pracovat,
- `CameraCaptureSession` – obsahuje další třídy pro zobrazení výstupů z kamery a umožňuje práci s pořízenými obrázky,
- `CaptureRequest` – řeší nastavení a výstupy pro pořízení jednoho obrázku, vytváří žádosti na vytvoření obrázků ze šablon
- `CaptureResult` – výsledek zpracované žádosti `CaptureRequest` instancí `CameraDevice`

Protože je podíl Androidu verze 5.0 a vyšší stále menšinový[12], preferuje se buď používání staršího API `Camera`, nebo implementace obou a použití vybraného API dle podpory konkrétního zařízení.[3]

Z důvodu výše zmíněné podpory `Camera2` API malým množstvím zařízení, absencí komplexnějších tutoriálů od vývojářů Androidu a využití staršího `Camera` API v praktické části práce se budou další části práce věnovat pouze staršímu API.

3.4.4 Nahrávání videa

Pro nahrávání videa je třeba v Androidu dodržet tento postup:

1. Získat kameru – získat instanci `Camera`, která slouží jako zdroj videa, pomocí metody `open()`.
2. Propojit UI komponentu s kamerou – komponentu (nejčastěji `Surfa-`

[11] Tedy při selhání metody `prepare()` instance `MediaRecorder`.
[12] V současné době (14. 12. 2015) má podíl verzí 5.0+ přesně 30%.

ceView) lze s kamerou spojit pomocí metody `setPreviewDisplay()` instance třídy `Camera`.

3. Spustit zobrazení náhledu – zobrazit uživateli, co kamera vidí, aby věděl, co bude uloženo ve videu – toto spustíme metodou `startPreview()` patřící opět instanci třídy `Camera`.

4. Spustit nahrávání videa – k tomu jsou třeba tyto kroky:

   (a) Odemknout kameru pro použití ve třídě `MediaRecorder` – u instance třídy `Camera` zavoláme metodu `unlock()`[13].

   (b) Nakonfigurovat instanci třídy `MediaRecorder`[14] následujícím způsobem:

      i. Nastavením zdroje (kamery) pomocí metody `setCamera()`.

      ii. Nastavením zdroje audia pomocí metody `setAudioSource()` (typicky `MediaRecorder.AudioSource.CAMCORDER`).

      iii. Nastavením zdroje videa metodou `setVideoSource()` (typicky `MediaRecorder.VideoSource.CAMERA`).

      iv. Nastavením výstupního formátu – od API 8 lze nastavit profilem metodou `setProfile()` (profil získáme z metody `get()` třídy `CamcorderProfile`), jinak lze nastavit přímo výstupní formát metodou `setOutputFormat()`, kódování audia metodou `setAudioEncoder()` a kódování videa metodou `setVideoEncoder()`.

      v. Nastavením výstupního souboru metodou `setOutputFile()`.

      vi. Nastavením, kam se bude zobrazovat výstup z kamery při nahrávání metodou `setPreviewDisplay()` – použijeme stejnou komponentu, jako u zobrazování náhledu před nahráváním.

   (c) Připravit instanci[15] `MediaRecorder` na nahrávání metodou `prepare()`.

[13] Od API 14 (Android 4.0) se o uzamykání a odemykání stará systém automaticky.

[14] Následující kroky musí být vykonány v určeném pořadí. Více v diagramu na developer.android.com[37]

[15] Metoda validuje nastavenou kombinaci parametrů a pak instanci nastaví dle daných parametrů.

   (d) Začít nahrávání metodou `start()`.

5. Zastavit nahrávání – nahrávání se zastavuje pomocí následujících metod zavolaných v daném pořadí:

   (a) Zastavením instance třídy `MediaRecorder` metodou `stop()`.

   (b) Resetováním instance třídy `MediaRecorder` metodou `reset()` – toto odstraní veškeré nastavení této instance.

   (c) Uvolněním zdrojů instance třídy `MediaRecorder` metodou `release()`.

   (d) Zamknutím instance třídy `Camera` (aby ji mohly využívat další instance třídy `MediaRecorder`) metodou `lock()` – navzdory poznámce výše o automatizaci zamykání zde platí jedna výjimka – pokud u instance třídy `MediaRecorder` selže metoda `prepare()`, je třeba instanci kamery zamknout manuálně.

6. Zastavit náhled z dané kamery zavoláním metody `stopPreview()` třídy `Camera`.

7. Uvolnit kameru zavoláním metody `release()` nad instancí třídy `Camera`.

Třídu `MediaRecorder` lze použít i s vynecháním částí, které nastavují náhled. V drtivé většině případů se toho však nevyužívá – uživatel pak nevidí, co nahrává.

Pokud víme, že v dané aplikaci budeme nahrávat videa a nebudeme pořizovat obrázky, můžeme zavolat u instance třídy `MediaRecorder` metodu `setRecordingHint()` s parametrem `true`, což může zrychlit přípravu na nahrávání a může pomoct vyhnout se případným komplikacím. Ideální je nastavování této hodnoty ještě před zobrazením náhledu uživateli, nesmí se však měnit po započetí nahrávání. Nahrávání videa i pořizování fotek bude fungovat s jakýmkoliv nastavením této hodnoty, slouží pouze k informování kamery, co je záměrem této aplikace (z těchto dvou možností).

Instanci třídy `MediaRecorder` lze rovněž nastavit dodatečné parametry[16] (jinak má každý parametr nastavenou výchozí hodnotu):

[16]Toto nastavení dodatečných parametrů se ve výše zmíněném postupu musí odehrát na místě nastavování profilů.

- velikost datového toku videa (metodou `setVideoEncodingBitRate()`),
- rozlišení videa (metodou `setVideoSize()`),
- počet snímků videa za sekundu (metodou `setVideoFrameRate()`),
- velikost datového toku audio (metodou `setAudioEncodingBitRate()`),
- počet kanálů u nahrávání zvuku (metodou `setAudioChannels()`, většinou na hodnotě 1 – mono, nebo 2 – stereo),
- vzorkovací frekvence audia za sekundu (metodou `setAudioSamplingRate()`)

Nahrávání videa pomocí Camera2 API probíhá obdobně – práce s instancí třídy `MediaRecorder` zůstává téměř stejná, liší se hlavně práce s kamerou (viz kapitola 3.4.3). Pro detaily implementace lze nahlédnout do ukázkové implementace od Googlu umístěné na GitHubu.[38]

Detailní informace a ukázka implementace nahrávání pomocí Camera API jsou k dispozici na developer.android.com.[39]

3.4.5 Ukládání multimediálních dat

Pro ukládání multimediálních dat jsou důležité 2 metody:

- `Environment.getExternalStoragePublicDirectory()` – tato metoda nám vrátí instanci třídy `File`, která odkazuje na veřejný adresář. Jako parametr jí můžeme zadat, který veřejný adresář chceme – například s parametrem `Environment.DIRECTORY_PICTURES` nám metoda vrátí adresář, kam se běžně ukládají veřejně dostupné obrázky. Pokud aplikaci odinstalujeme, soubory uložené v tomto adresáři nebudou smazány.
- `Context.getExternalFilesDir()` – tato metoda vrátí adresář asociovaný s ukládáním obrázků dané aplikace. Soubory nejsou nikterak chráněny, takže je mohou vidět a měnit ostatní aplikace. Po odinstalování aplikace jsou smazány i soubory v tomto adresáři. Metoda rovněž přijímá typ parametru zmíněný v bodu výše.

Vytvoření (respektive příprava) souboru tak, abychom do něj mohli ukládat

multimediální obsah, je znázorněno ve zdrojovém kódu 3.4. Je třeba dodat, že na začátku ukázky by bylo vhodné kontrolovat existenci externího úložiště metodou `Environment.getExternalStorageState()` a rovněž není špatný nápad vytvářet unikátní název souboru (v ukázce je kvůli zkrácení délky kódu uvedeno jen „IMG.jpg“), například pomocí data vytvoření souboru.

```
File mediaStorageDir = new File(Environment.
   getExternalStoragePublicDirectory(Environment.
      DIRECTORY_PICTURES), "MyCameraApp");

// Create the storage directory if it does not exist
if (! mediaStorageDir.exists()){
    if (! mediaStorageDir.mkdirs()){
        Log.d("MyCameraApp", "failed to create directory");
        return null;
    }
}

// Create a media file name
File mediaFile = new File(mediaStorageDir.getPath() +
   File.separator + "IMG.jpg");

return mediaFile;
```

Zdrojový kód 3.4: Ukázka kódu pro vytvoření souboru, do kterého budeme ukládat vytvořený obrázek

3.5 Streamování

Podkapitola obsahuje informace o streamovaní a jeho podpoře na platformě Android.

3.5.1 Streamování obecně

Streamování multimédií lze definovat jako metodu, jejímž úkolem je přenést multimediální obsah (většinou audio nebo video) od zdroje poskytujícího „streamovatelný“ obsah ke koncovému uživateli. Od klasického postupu

se liší zejména tím, že se stahuje pouze část obsahu, kterou chceme zrovna „zkonzumovat“ (u videa se tím míní zhlédnutí, u audia poslech atd.).

Streamovat lze jak předem nahraný obsah, tak i živé vysílání.[40] Růst oblíbenosti tohoto typu konzumace multimédií souvisí zejména se zkvalitňováním připojení k internetu ve většině částí světa – nemálo uživatelů má již z domova dostatečnou konektivitu ke sledování filmu, který je přímo při sledování stahován ze serveru ve vysoké kvalitě.

Streamování předem nahraného obsahu je technicky nejjednodušší – velikost odezvy (zde je tím míněn čas mezi stáhnutím dané části a zhlédnutím té samé části) se zde klidně může pohybovat v řádech minut[17]. Příkladem streamování předem nahraného obsahu jsou služby jako Netflix, Voyo, Spotify aj., založené na měsíčním či jednorázovém poplatku za sledování či poslech multimediálního obsahu.

U streamování živého vysílání (kde jeho obsah vzniká zároveň s jeho sledováním) lze nalézt oba druhy – se zpožděním i bez zpoždění (respektive s co nejmenším zpožděním). Příkladem služeb s živým vysíláním jsou například Twitch.tv, Hitbox aj., které jsou založeny na vysílání živého obsahu od zdroje (tito lidé jsou pak nazýváni tzv. streamery) k divákům. Dobu zpoždění si ve většině případů může řídit sám streamer[18]. Příkladem streamování s co nejmenším zpožděním mohou být služby vyžadující nikoliv jednosměrný (jako u všech výše uvedených) ale obousměrný provoz – tedy například video chatovací služby (jako je Skype, Google Hangouts aj.). Zde probíhá snaha o minimalizaci zpoždění, neboť narušuje dojem z výsledného hovoru. Co nejmenší zpoždění je rovněž vyžadováno u sportovních přenosů, zvláště pokud jsou spojené se sázením (například portál tipsport.cz), neboť pro uživatele je důležité rychle reagovat na události v dané sportovní události[19].

V souvislosti se streamováním je třeba zmínit také *RTSP*, což je protokol aplikační vrstvy starající se o přenos streamovaného obsahu, synchronizaci (mezi videem a audiem) a ovládání streamu (pozastavování či spuštění přehrávání atd.). Většina *RTSP* serverů využívají protokolů *RTP* a *RTCP*. Pro streamování je s protokolem *HTTP* využíván nejčastěji.[41]

[17]Uživateli zde nevadí, že si na film musí chvíli počkat, důležité je, aby tato doba zůstala víceméně stejná – při zvětšování by se video zasekávalo při čekání na data.

[18]U Twitch.tv to bývá nastavení v rozmezí 10 sekund až několika minut. Obdobná rozmezí jsou nabízena i u konkurenčních služeb.

[19]Například by nebylo příjemné při nové sázce vidět náhlou změnu kurzu k horšímu poměru a až s minutovým zpožděním zjistit, že změna proběhla kvůli vstřelení gólu.

3.5.2 Streamování na Androidu

Nativní podpora streamování video obsahu po síti v Androidu chybí. Co se týče audia, má Android základní podporu pro streamování audia přes RTP – v balíčku `android.net.rtp` nabízí tyto 4 třídy:

- `AudioCodec` – třída obsahující audio kodeky, které lze použít u třídy `AudioStream`,
- `AudioGroup` – třída fungující jako hub pro mikrofon, reproduktor a přidané audio streamy,
- `AudioStream` – třída starající se o přenos dat přes RTP, její instance symbolizuje jeden kanál (například příjem audia od protistrany při hovoru),
- `RtpStream` – základní třída pro posílání a příjem dat přes RTP, dědí od ní třída `AudioStream`

3.6 Externí knihovny

Podkapitola má za cíl seznámit čtenáře s vybranými multimediálními externími knihovnami, které lze při práci s multimédii či streamováním multimediálního obsahu využít.

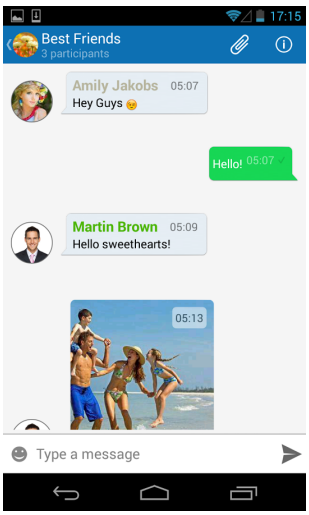
3.6.1 Vitamio

Vitamio je multimediální knihovna s otevřeným zdrojovým kódem pro Android a iOS rozšiřující podporu multimediálních formátu jednotlivých platforem. Za zmínku stojí například možnost přehrávání souborů WMV či FLV – častých, avšak nativně v Androidu nepodporovaných formátů. Knihovna rovněž implementuje podporu pro síťové protokoly pro streamování multimédií, jako *HTTP progressive streaming*, *RTSP* a další. Vitamio využívá knihovnu *FFmpeg* verzi 2.0, pro správné fungování vyžaduje Android ve verzi 2.1 a vyšší (případně verzi 4.3+ u iOS verze) a podporuje procesorové architektury *ARMv6*, *ARM VFP*, *ARMv7*, *ARM NEON*, *MIPS* a *X86*. Firma Vitamio umožňuje pouze experimentování s knihovnou (vývoj aplikace) zdarma,

podmínkou pro jakékoliv použití (komerční i nekomerční) je zakoupení licence (cena na webu není zmíněna, všude je použito sousloví „Business negotiation“).[42]

3.6.2 QuickBlox

QuickBlox je back-end API od stejnojmenné firmy zajišťující komunikaci mezi serverem a klientem. Jeho úkolem je poskytovat možnost komunikace mezi dvěma uživateli. Komunikaci umožňuje několika různými způsoby – pomocí chatu, video hovoru či posíláním souborů. API je rozděleno do několika modulů, autoři pak v dokumentaci popisují, které moduly jsou potřeba při implementaci daných způsobů komunikace[20]. SDK je přístupné pro mobilní platformy Android, iOS, Windows Phone[21] a Blackberry. Zdrojový kód API je uzavřený, knihovny a příklady implementace jsou poskytovány zdarma na GitHubu, zpoplatněno je pronajmutí serverů[22] potřebných pro správný chod (pronájem začíná na ceně $49 za měsíc). Volně přístupné jsou rovněž zdrojové kódy aplikace *Q-municate* – oficiální aplikace implementující všechny funkce QuickBlox API.[43]



Obrázek 3.2: Ukázka aplikace *Q-municate*, zdroj: play.google.com

[20]Například pro implementaci textových zpráv jsou potřeba moduly `Chat` a `Core`.
[21]Pouze pro verzi 7.
[22]Firma nabízí i variantu zdarma, která je omezená počtem odeslaných zpráv, počtem uživatelů a poskytovanou podporou.

3.6.3 RTMP Broadcast Library

RTMP Broadcast Library od firmy *Agilio* je knihovnou umožňující zařízení s Androidem nebo iOS streamovat audio a video na server podporující RTMP – proprietární protokol pro přenos videa a audia přes internet vyvinutý firmou *Macromedia*. Jako formát pro přenos videa byl zvolen H.264, přenos audia je realizován ve formátu AAC. Knihovna podporuje procesorové architektury *i386*, *ARMv7*, *ARMv7s* a *ARM64* a pro svůj běh vyžaduje iOS verze 4.3 až 8.x[23]. Použití knihovny je placené, nabízena je zkompilovaná knihovna (za cenu $1,990) a zdrojový kód, je možné si připlatit za rozšíření funkcionality knihovny (podpora Full HD rozlišení, podpora variabilního datového toku atd.). Oficiální aplikací využívající tuto knihovnu je pak *Broadcast Me*, dostupná pro Android a iOS.[44]

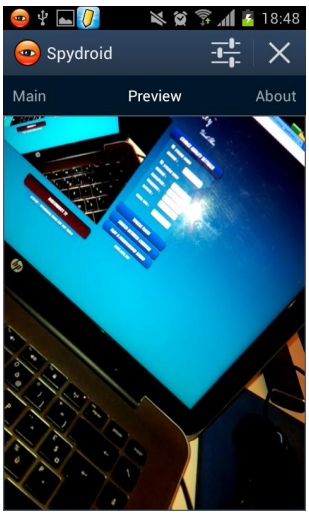
Obrázek 3.3: Ukázka aplikace *Broadcast Me*, zdroj: play.google.com

3.6.4 Knihovna libstreaming

Další popisovanou knihovnou je *libstreaming*. Jde o open-source knihovnu umožňující streamování obrazu z kamery a audia z mikrofonu ze zařízení s OS Android pomocí RTP. Knihovna podporuje video formáty H.264 a H.263 a audio formáty AAC a AMR. Knihovna je nabízena pod GNU GPL licencí nebo pod komerční licencí (pro využití v aplikaci s uzavřeným kódem). Aplikací demonstrující funkčnost knihovny je *Spydroid-ipcamera* pro Android.

[23]O požadované verzi OS Android se na stránce knihovny autoři nezmiňují.

Projekt se zdá být dále neudržovaný, poslední aktualizace proběhla k datu 3. 10. 2015.[45]



Obrázek 3.4: Ukázka aplikace *Spydroid-ipcamera*, zdroj: play.google.com

3.6.5 Red5 Media Server a Red5Pro

Red5 Media Server je serverová open-source aplikace od společnosti *Infrared5* napsaná v jazyce Java, sloužící ke streamování multimediálního obsahu přes *RTMP* a *HTTP* protokol či přes tzv. *Web sockets*.[46]

Red5 Pro je uzavřenou variantou stavějící na open-source variantě. Nabízí navíc HW prostředky pro provozování serveru, SDK pro podporu Android a iOS zařízení a technickou podporu[24]. Je možné využívat *Red5 Pro* zdarma při konektivitě serveru omezené na 10 souběžných streamů a absenci technické podpory. Placená verze pak začíná na ceně $55 za měsíc.[47]

3.6.6 PubNub

PubNub od stejnojmené firmy nabízí řešení pro komunikaci (včetně streamování multimediálních dat) v reálném čase. Firma nabízí SDK s podporou 70 jazyků (zahrnují tedy podporu pro Android, iOS, Windows Phone a další)

[24] *Red5 Media Server* nabízí pouze komunitní podporu.

a pronájem svého cloudu fungujícího jako server (je tedy nezbytný při použití tohoto řešení). Cena za tyto služby začíná na ceně $149 za měsíc. Pro použití existuje i bezplatná varianta omezená počtem 100 aktivních uživatelů za 1 den a omezením maximálního měsíčního počtu zpráv na 1 milión.

4 Tvorba streamovací aplikace

Tato kapitola popisuje tvorbu aplikace pro přenos videa a audia. V jednotlivých podkapitolách rozebírá konkrétní implementaci, vysvětluje návrhy podle kterých implementace vznikla a rovněž obsahuje popis testování aplikace a výsledky testů.

4.1 Návrh aplikace

Součástí této práce je návrh a vytvoření aplikace pro online přenos videa a audia.

Přenos lze realizovat jednosměrně či obousměrně. Zvolen byl jednosměrný přenos mezi dvěma zařízeními – tedy „vysílač“ a „přijímač“. Vysílající zařízení je v práci popisováno jako „server“, přijímající zařízení je popisováno jako „klient“.

Řízení přenosu (myšleno jeho spuštění, zastavení změna parametrů atd.) je konfigurovatelné z obou zařízení. Detaily jsou popsané v podkapitole A.

Jako filtr knihoven zmíněných v podkapitole 3.6 pro využití v této aplikaci byly kladeny následující podmínky:

- Knihovna musí být zdarma alespoň pro využití v nekomerčních aplikacích.
- Knihovna, která ve verzi zdarma omezuje svojí funkcionalitu, může být uznána (záleží na druhu omezení).
- Je žádané, aby knihovna byla otevřená (bylo možno nahlédnout do zdrojového kódu).
- Preferované jsou řešení bez jakýchkoliv uzlů mezi koncovými zařízeními (toto však na rozdíl od ostatních není nutná podmínka).

Knihovna *Vitamio* má otevřené zdrojové kódy, pro jakékoliv použití je však vyžadováno zakoupení licence. Nemožnost využít knihovnu zdarma vylučuje rovněž *RTMP Broadcast Library*. Využití *QuickBlox*, *Red5Pro* a *PubNub*

brání požadavek na otevřené řešení, u posledně jmenovaného je navíc nejasné omezení na 1 milión zpráv měsíčně ve vztahu ke streamování videa – je mezi nimi souvislost? Pokud ano, kolik zpráv zabere stream o definované délce (10 minut, 2 hodiny atd.)?

Ze zmiňovaných knihoven se tak jako nejvhodnější jevila knihovna *libstreaming*. Přenos videa a audia u aplikace *Spydroid* (která pro přenos využívá knihovnu *libstreaming*) se při testování této aplikace nepodařilo zprovoznit[1]. Při experimentování s touto knihovnou v rámci vlastní aplikace se sice podařilo spustit server vysílající přes protocol *rtsp*, nicméně žádný z testovaných klientů nebyl schopen tento stream ani „přečíst“, ani zobrazit.

Protože bylo navíc preferované řešení bez jakéhokoliv uzlu mezi dvěma koncovými zařízeními, bylo nakonec rozhodnuto o vytvoření vlastního řešení pro přenos videa a audia. Pro oddělení logiky programu a zobrazovací části je aplikace rozdělena dvě části: knihovnu a aplikaci. Knihovna obsahuje většinu logiky programu (konkrétně popsanou v podkapitole 4.2), aplikace pak obsluhuje uživatelovy požadavky a zobrazuje výstupy knihovny (stream).

Při navrhování způsobu streamování z jednoho zařízení do druhého byly zvažovány dva postupy:

1. Streamovat výstup z kamery přímo – Android umožňuje výstup z kamery přesměrovat do socketu (s využitím tzv. `ParcelFileDescriptor`, který umožní instanci `MediaRecorder` zapisovat do streamu, jako by to byl soubor)

2. Nahrávat úseky videa (a audia) do souboru a tento řetěz souborů posílat druhému zařízení.

První ze způsobů se na první pohled jeví jako vhodnější, neboť jeho stream je kontinuální (u druhé možnosti budou existovat drobné výpadky způsobené uzavíráním nahrávání jednoho souboru a otevřením dalšího). Zde se však objevuje problém s řešením jakýchkoliv anomálií, neboť pokud se při přenosu vyskytne chyba (například malé zadrhnutí vinou chvilkového slabého signálu WiFi), přehrávač se pokusí danou část videa přehrát znovu – zde nastává problém, protože takovýto stream nemůže být z principu tzv. *seekable* (nemůžeme se nijak přesouvat zpět či vpřed při přehrávání tohoto videa). Tento přístup tedy bude funkční pouze za ideálních podmínek [48]. Z tohoto důvodu byla zvolena druhá metoda – video ze streamu bude obsahovat malé výpadky,

[1] Aplikace běžela, náhle se neukončovala, jenže nepřenesla žádné video ani audio.

mělo by nicméně být odolnější vůči nepříznivým podmínkám (například výše zmíněný problém s poklesem signálu WiFi).

Jako minimální požadované API pro aplikaci a knihovnu bylo zvoleno nejnižší API u testovaných zařízení, tedy *API 16* – minimální potřebná verze Androidu pro spuštění této aplikace je tedy verze *4.1* (pro více informací o testovacích zařízeních viz podkapitola 5.1 na straně 38).

4.2 Popis implementace – knihovna

V této podkapitole je popsána implementace knihovny sloužící pro streamování videa. Knihovna byla nazvána `CamStreamerLib`.

4.2.1 Popis částí knihovny

Obsah knihovny lze popsat dvěma způsoby: programovým (členění ve zdrojovém kódu) a funkčním (členění dle způsobu, jakým spolu jednotlivé části spolupracují).

Z hlediska programového je zdrojový kód knihovny rozdělen do balíčků takto (řazeno abecedně):

- balíček `camera` – část starající se o práci s kamerou – zamykání a odemykání, práci s preview kamery (náhledem) a rovněž nahrávání souborů,
- balíček `client` – klientská část starající se o komunikaci se serverem a hlavní třídou knihovny (tj. `CSStreamManager`),
- balíček `network` – obsahuje pouze výčet všech síťových příkazů, které si mezi sebou posílají server a klient (`NetworkMessage`),
- balíček `server` – serverová část starající se o komunikaci s klientem a hlavní třídou knihovny (tj. `CSStreamManager`),
- balíček `storage` – část zabývající se ukládáním video souborů a přístupem k těmto uloženým souborům[2],

[2]Ukládány jsou instance třídy `File` – jde tedy o ukládání odkazů na dané soubory.

- balíček `utils` – obsahuje pomocné proměnné využívané v celém projektu a globální nastavení aplikace (třída `CSSettings`),
- balíček `video` – obsahuje část implementující přehrávání videa na straně klienta

V kořenovém balíčku (tj. `cz.zcu.fav.kiv.zimmma.camstreamerlib`) se nachází třída a několik rozhraní, skrz která může aplikace využívající tuto knihovnu ovládat stream, popřípadě získávat různé informace (o připojení k síti, o nových událostech/stavech streamu atd.). Konkrétně se jedná o třídu `CSStreamManager` sloužící pro ovládání streamu (jak z pozice klienta, tak z pozice serveru) a o implementovatelné rozhraní `INetworkStateListener`, sloužící pro informování uživatele o stavu připojení k síti a `IServerStateChangeListener` s `IClientStateChangeListener` sloužící k informování uživatele (přes GUI) o změnách, které se udály v rámci streamu, respektive celé knihovny (změna kvality streamu, spuštění streamu, odpojení od serveru aj.). Této třídě a uvedeným rozhraním se detailněji věnuje podkapitola 4.2.2.

Druhý způsob dělení spočívá v rozdělení knihovny na „moduly“, které je abstraktnější a díky kterému lze lépe vysvětlit principy fungování jednotlivých částí knihovny. Knihovnu lze tímto způsobem rozdělit takto:

- Manager,
- Client stream,
- Server stream,
- Storage,
- Camera,
- Video player

Každý modul bude detailně popsán v následujících podkapitolách. Schéma klientských i serverových modulů lze najít v příloze B.

4.2.2 Modul „Manager“

Do tohoto modulu spadá třída `CSStreamManager`, balíčky `client` a `server` a všechna rozhraní v kořenovém balíčku. Úkolem tohoto modulu je zpracovávat požadavky uživatele vyslané přes GUI (start streamu, změna kvality, připojení se k serveru atd.) a naopak informovat uživatele o jakýchkoliv změnách (ukončení spojení z druhé strany, vyžádána změna kvality streamu, nahrávání videa selhalo atd.). Množina stavů, kterými je knihovna schopná informovat GUI (resp. uživatele) o změnách, je popsána ve výčtu `ClientStateChange` pro klienta a ve výčtu `ServerStateChange` pro server[3].

`CSStreamManager` při práci s danými instancemi serveru a klienta pracuje s jejich super třídami `StreamServer` a `StreamClient`. Toto zajišťuje možnost rozšíření – implementaci zcela jiného způsobu streamování videa a audia. Oddděděné třídy `CachedStreamServer` a `CachedStreamClient` pak představují implementaci konkrétního způsobu streamování těchto multimédií (více jsou rozebrány v následujících podkapitolách).

4.2.3 Moduly „Client stream“ a „Server stream“

Cílem těchto modulů je síťová komunikace. Patří sem třídy `CachedStreamClient` a `CachedStreamServer`, respektive privátní třídy v ní obsažené.

Pro práci se sockety je využito třídy `Socket`, respektive pro navázání spojení je na serveru použita třída `ServerSocket`.

Pro síťovou komunikaci jsou použity dva sockety. Komunikace probíhá přes protokol TCP. První, nazvaný „control socket“, slouží k obousměrnému zasílání zpráv, jejichž cílem je změnit stav spojení nebo o nějaké události (uživatel se chce odpojit, změnit kvalitu streamu apod.) informovat protistranu. Zpráva se skládá ze čtyřbytového integeru signalizujícího typ zprávy (například před odpojením se od protistrany je poslána zpráva `DISCONNECT` symbolizovaná posláním integeru s hodnotou `1`). Detailní informace všech typů zpráv lze najít v JavaDoc příslušného výčtu `NetworkMessage` v balíčku `network`.

Druhý socket, nazývaný „data socket“, je využíván jednosměrně pro přenos datových souborů ze serveru ke klientovi. Formát posílaných dat na tomto

[3]Více informací je obsaženo v JavaDoc dokumentaci příslušných výčtů.

socketu je tento: `<file size><file name size><file name><file>`
kde:

- `file size` – integer (4 byty), velikost posílaného souboru v bytech,
- `file name size` – integer (4 byty) velikost posílaného jména souboru v bytech,
- `file name` – řetězec v UTF-8, jméno posílaného souboru,
- `file` – byty, data posílaného souboru

Každý ze socketů je provozován na svém vlastním portu. „Control socket“ spojuje strany na portu 9696, „data socket“ využívá portu 9697. Hodnoty jsou uloženy ve třídě `CSSettings` v balíčku `utils`.

Z pohledu serveru se o příjem a zpracování kontrolních zpráv „control socketu“ stará privátní třída `StreamServerControlReceiver`, která periodicky kontroluje, zda je na vstupu socketu možno číst data. Pokud ano, vlákno přečte zprávu a vykoná potřebné příkazy. Odesílání řeší přímo instance třídy `CachedStreamServer`. Odesílání datových souborů přes „data socket“ má na starosti `StreamServerDataSender`. O navazování spojení se serverem se stará instance privátní třídy `StreamServerConnectionListener`.

Na straně klienta se o příjem a zpracování zpráv stará rovněž privátní třída `StreamClientControlReceiver`, fungující na stejném principu jako třída serveru zpracovávající příchozí zprávy na „control socketu“. Přijímání datových souborů přes „data socket“ řeší instance privátní třídy `StreamClientDataReceiver` a o navazování spojení se serverem se stará instance privátní třídy `StreamClientConnectionHandler`.

Při spuštění serveru jsou na výběr dvě možnosti: *normální* a tzv. *Power save recording* režim. Při spuštění s normálním režimem je okamžitě po startu rovněž spuštěno nahrávání z kamery. Výhodou tohoto režimu je zmenšení načítání („bufferování“) před začátkem přehrávání videa při připojení klienta. Při spuštění s power save recording režimem je nahrávání videa spuštěno až při připojení klienta a je opětovně vypnuto po odpojení klienta. Výhodou tohoto režimu je úspora na straně serveru (zejména úložného místa) a menší zpoždění přehrávaných souborů (neplést s výše zmíněným „bufferováním“, více viz podkapitola 4.2.4).

4.2.4 Modul „Storage“

Funkcí tohoto modulu je přijímat a ukládat video soubory a při různých dotazech je pak ve správném pořadí vracet. Korektní implementaci určuje rozhraní `IVideoFileStorage`, funkční implementací je pak třída `CacheVideoFileStorage`. Prací se soubory je zde myšlena práce s instancemi třídy `File`, která obsahuje informace o daném souboru a skrz kterou je možné k datům souboru přistupovat.

Na straně serveru jsou do instance zmíněné třídy ukládány soubory vytvořené nahráváním kamery (více informací viz podkapitola 4.2.5), na straně klienta jsou ukládány soubory obdržené ze serveru. Úložiště má omezený počet uložených souborů[4]. Při připojení klienta začne server postupně posílat klientovi soubory od nejstaršího k nejnovějšímu.

Nejdůležitější na tomto modulu je přístup k již uloženým souborům metodou `getNextFile()`, která dle předaného parametru umožňuje vrátit jak nejstarší soubor úložiště (když je parametr `null`, potřebné při připojení nového klienta), tak nejbližší novější soubor a rovněž je možné rozpoznat stavy, kdy úložiště nemá žádné soubory (metoda vrátí `null`) a kdy poslední soubor předaný parametrem je v dané době nejnovějším souborem (metoda vrátí instanci předanou v parametru). Pro více informací o chování úložiště viz JavaDoc rozhraní `IVideoFileStorage`.

Podle zaplněnosti úložiště se pak mění doba zpoždění vzniku videa ku shlédnutí videa uživatelem (klientem). Pokud jsou v úložišti dva soubory, pak bude doba zpoždění 10 sekund (videa jsou 5 sekundové). Při maximálním množství (tedy 5 videí) bude zpoždění 25 sekundové.

4.2.5 Modul „Camera“

Do tohoto modulu patří obsah balíčku `camera`. Hlavní třída `CameraManager` se stará o veškerou práci s kamerou a nastavení náhledu kamery pro uživatele. Nejdůležitější částí je privátní třída `CameraRecorderTask`, jejíž instance při spuštění po nastavené době (v aplikaci je nastaveno 5000 milisekund – 5 sekund) zastaví probíhající nahrávání, připraví nové nahrávání a spustí ho. Soubory vytvořené tímto nahráváním pak ukládá přes modul „storage“ (pro

[4] V aplikaci je nastavena fixní hodnota 5 - při vyšším počtu souborů se z úložiště odebere nejstarší soubor.

více informací o ukládání viz podkapitola 4.2.4). `CameraManager` také provádí změny parametrů nahrávaného videa a také hlásí stav nahrávání. Výčet `CameraRecordingState` obsahuje seznam všech stavů, které umí `CameraManager` hlásit. Pro hlášení stavů je třeba implementovat rozhraní `ICameraRecorderStateListener` a zaregistrovat danou třídu do `CameraManager` metodou `addCameraRecorderStateListener()`.

4.2.6 Modul „Video player“

Poslední popisovaný modul slouží na straně klienta k přehrávání přijatých video souborů. Nezbytná funkčnost je zajištěna implementací rozhraní `IVideoPlayer`, referenční ukázkou implementace je pak třída `VideoPlayer`. Ten pro správnou funkčnost vyžaduje komponentu schopnou přehrávat video (implementovaná třída využívá komponentu `VideoView`) a rovněž tzv. *handler* umožňující nastavovat tuto komponentu z UI vlákna[5]. Třída přehrávače poté úzce spolupracuje s úložištěm (pro více informací o úložišti viz podkapitola 4.2.4) – nejprve čeká, než úložiště bude obsahovat dostatek souborů k přehrávání (v aplikaci je požadovaný počet souboru nastavený na minimálně dva v úložišti) a poté je začne přehrávat.

4.2.7 Práce s knihovnou

Pokud chceme využívat služby knihovny v nějaké aplikaci, existují dva způsoby importování v Android Studiu. Je možné přidat odkaz na projekt knihovny:
File > *Project Structure* > kliknout na ikonku + (plus) > v okně *New Module* zvolit *import Gradle Project* > uvést cestu k projektu knihovny
Projekt knihovny je přiložen na CD (viz příloha C).

Druhou možností je stejným postupem importovat soubor AAR, což je ekvivalent JAR souborů pro knihovny na Androidu:
File > *Project Structure* > kliknout na ikonku + (plus) > v okně *New Module* zvolit *import .JAR/.AAR Package* > uvést cestu ke knihovně

Pro to, aby mohla knihovnu využívat aplikace, slouží třída `CSStreamManager` a tři implementovatelné rozhraní.

[5]Android nedovoluje jakoukoliv manipulaci s UI komponentami v jiném než UI vlákně.

Třída `CSStreamManager` je hlavním bodem pro využití funkce knihovny. Při vytváření její instance v aplikaci vyžaduje knihovna tzv. kontext aplikace (`Context`) pro správnou funkčnost vlákna, které ověřuje konektivitu zařízení. Změny ohledně připojení zařízení k síti je možné hlásit uživateli implementováním rozhraní `INetworkStateListener` a registrováním daného posluchače metodou `addStreamStateListener()`.

Pokud chceme ze zařízení vytvořit server, můžeme tak učinit metodou `startStreamServer()`, která vyžaduje následující parametry:

- `powerSaveRecording` – zda má server nahrávat jen při připojení klienta či neustále (více viz podkapitola 4.2.3 na straně 32),
- `context` – kontext aplikace pro správnou funkčnost nahrávání videí,
- `svPreview` – komponentu `SurfaceView` pro zobrazení náhledu kamery,
- `camera` – instanci kamery, která bude použita pro nahrávání,
- `mediaStorageDirectory` – adresář, kam se budou ukládat nahraná videa,
- `serverStreamStateChangeListener` – třída, které se budou hlásit změny stavu serveru

Třída implementující rozhraní `IServerStateChangeListener` bude upozorněna při změnách stavu serveru a může tak na tyto události adekvátně reagovat. Pro ukončení serveru slouží metoda `stopStreamServer()`.

Pokud chceme ze zařízení vytvořit klienta, můžeme tak učinit metodou `startStreamClient()`, která vyžaduje následující parametry:

- `svVideo` – komponentu `VideoView` pro přehrávání přijatých videí,
- `mediaStorageDirectory` – adresář, kam se budou ukládat přijatá videa,
- `clientStreamStateChangeListener` – třída, které se budou hlásit změny stavu klienta,
- `ipHostname` – IP adresa či hostname serveru,
- `context` – kontext aplikace pro správnou funkčnost přehrávání videí

Třída implementující rozhraní `IClientStateChangeListener` bude upozorněna při změnách stavu klienta a může tak na tyto události adekvátně reagovat. K ukončení klienta (odpojení od serveru a zastavení všech běžících vláken) slouží metoda `stopStreamClient()`.

Pokud bychom chtěli při běhu změnit stav streamu, můžeme tak učinit metodou `changeStream()`, jenž vyžaduje dva parametry – hodnotu výčtu `ChangeStreamAction` znázorňující, co chceme změnit, a případné parametry potřebné k této změně[6].

4.3 Popis implementace – aplikace

Aplikace je tvořena celkem třemi aktivitami. První aktivita, označena jako startovní, nabízí pouze dvě tlačítka – kliknutím na „Server“ se přesuneme na aktivitu umožňující vytvořit ze zařízení server a tlačítkem „Client“ se přepneme na aktivitu umožňující připojení se k serveru.

Druhá aktivita slouží k použití zařízení jako server. Zobrazuje informace o stavu streamu a stavu připojení k síti. Uživatel zde má možnost nastavit „Power save recording“ režim před spuštěním serveru, spustit server a také vidí náhled kamery.

Třetí aktivita slouží k použití zařízení jakožto klienta. Zobrazuje informace o stavu streamu a stavu připojení k síti. Uživatel může zadat IP adresu serveru, ke které se chce připojit, a poté může ovlivňovat stream nastavením kvality a zastavením live streamu. Plocha sloužící ke zobrazení videa se objeví po připojení k serveru a při odpojení opět mizí. Protože informace o stavu nejsou při probíhajícím přehrávání důležité, skryjí se, aby plocha určená k zobrazení streamu mohla být větší.

Protože náhled z kamery počítá s používáním telefonu na šířku[7] (tzv. režim *landscape*), jsou aktivity navržené pouze pro režim na šířku pro zachování jednotnosti.

Aplikace si pro své fungování vyžádá na daném zařízení následující povolení (řazeno abecedně):

[6] V současné implementaci je druhý parametr využitý jen u startování streamu z pozice klienta.

[7] Při použití telefonu na výšku (*portrait*) je obraz z kamery posunutý o 90 stupňů.

- `android.permission.ACCESS_NETWORK_STATE` – pro přístup k informacím o stavu připojení,
- `android.permission.CAMERA` – pro využívání kamery (nahrávání videa),
- `android.permission.INTERNET` – pro používání socketů pro přesun dat po síti,
- `android.permission.READ_EXTERNAL_STORAGE` – pro čtení dat na úložišti,
- `android.permission.RECORD_AUDIO` – pro nahrávání audia přes mikrofon
- `android.permission.WRITE_EXTERNAL_STORAGE` – pro zápis dat na úložiště

Více informací o grafické stránce aplikace a jejím ovládání lze nalézt v uživatelském manuálu (příloha A).

5 Testování aplikace

Tato kapitola popisuje testovaná zařízení a provedené testy a seznamuje čtenáře s dosaženými výsledky.

5.1 Testovací zařízení

Aplikace byla primárně testována na třech zařízeních: *Acer Liquid Z530*, *Samsung Galaxy S3 Mini* a *GIGABYTE GSmart Roma R2*. Základní informace o parametrech těchto telefonů lze najít v tabulce 5.1.

	Acer Liquid Z530	**Samsung Galaxy S3 Mini**	**GIGABYTE GSmart Roma R2**
Android	5.1	4.1.2	4.2.2
Uhlopříčka displeje	5”	4”	4”
Rozlišení displeje	1280×720	800×480	800×480
Rozlišení kamery	8 MPx	5 MPx	5 MPx
Procesor	4×1,3 GHz	2×1 GHz	2×1,3 GHz
RAM	1 GB	1 GB	1 GB
Úložiště	8 GB	8 GB	4 GB

Tabulka 5.1: Popis parametrů testovacích zařízení.

5.2 Metodika testování

V této podkapitole jsou popsané testy vykonané v rámci otestování funkčnosti aplikace.

5.2.1 Test velikosti souborů a stabilita

V tomto testu server po určenou dobu nahrával kamerou zvolenou scénu. Po uplynutém čase bylo změřeno kolik místa bylo zabráno. Byly zvoleny intervaly 1 minuta, 10 minut a 60 minut. Poslední z testů byl rovněž testem stability aplikace.

Jako testovací scéna bylo zvoleno nahrávání monitoru, na kterém běžel hodinový sestřih akčních scén. Při testování kratších úseků (1 a 15 minut) se nahrávala pouze testovaná délka – celkově se tedy nahrávalo prvních 60, 15 a 1 minuta.

Protože byly nahrávány akční scény, bylo předpokládáno, že nahrané soubory se budou velikostně blížit spíše horní mezi jejich teoretické maximální velikosti. Lze tedy předpokládat, že při nahrávání klidnějších scén budou přenášené soubory menší velikosti, než u výsledků tohoto testu.

5.2.2 Test výpadků připojení

V tomto testu byla otestována odolnost vůči výpadku připojení k síti. Testován byl jak nečekaný výpadek zdroje WiFi (tedy AP), tak i nechtěné vypnutí uživatelem (vypnutí připojení v daném zařízení), či obecně vzato výpadek připojení pouze na jedné straně.

Cílem by mělo být korektní chování aplikace – neměla by neočekávaně spadnout, při výpadku spojení by mělo být možné se k serveru opětovně připojit.

5.2.3 Test slabého spojení

Bylo otestováno, jak si obě zařízení povedou při snížené kvalitě signálu WiFi. Aplikace nesmí spadnout, přenos dat může být zpožděný.

5.2.4 Test přerušení běhu aplikace

Bylo otestováno přerušení běžící aplikace jinou aplikací. Zvoleno bylo přerušení hovorem – na telefon s běžící aplikací bylo zavoláno z jiného zařízení, hovor alespoň několik sekund probíhal a poté byl ukončen.

5.3 Výsledky testování

Při dlouhodobém testování se aplikace ukázala jako zcela stabilní. Výsledky testování velikostí přenesených souborů jsou znázorněny v tabulce 5.2. Obraz při testování nízké kvality[1] nezabíral mnoho místa i při delším streamování, ztrácely se v něm však některé detaily objektů. Při testování vyšší kvality[2] byl výsledný obraz dostatečně kvalitní pro bližší prozkoumání všech objektů v obraze, nicméně při dlouhodobém pozorování výrazně vzrostly nároky na úložný prostor. Rovněž při streamování ve vyšší kvalitě vzrostl požadavek na lepší kvalitu připojení (především na vyšší rychlost připojení).

Doba streamování	**Nízká kvalita**	**Vysoká kvalita**
1 minuta	1,4 MB	15,2 MB
15 minut	26,3 MB	314,3 MB
60 minut	107,5 MB	1 260,0 MB

Tabulka 5.2: Výsledky testu celkové velikosti přenesených souborů.

Při testování výpadku připojení k síti byla aplikace stabilní, spojení mezi zařízeními bylo přerušeno a obě strany se „resetovaly“ do původního nastavení (odpojeno, připraveno na zapnutí). Stejná situace nastala u testu přerušení běhu aplikace.

Testování slabého připojení bylo také úspěšné – přenos souborů se při velmi nízkém signálu zpomalil, aplikace se však odpojila až při úplné ztrátě signálu.

[1] Byl použit profil s nejnižší možnou kvalitou – `QUALITY_LOW`.
[2] Byl použit profil s názvem `QUALITY_480P`.

6 Možná vylepšení aplikace

V aplikaci je prostor pro vylepšení.

Server by mohl být upraven tak, aby zvládal několik souběžných klientů místo jednoho. Informace spojené s jedním uživatelem by bylo třeba oddělit (do samostatné třídy) a zajistit správný přístup vůči sdíleným zdrojům (kamera a úložiště natočených video souborů).

Správce kamery by mohl být rozšířen o podporu více možností nastavení. Toto by šlo zařídit dvěma způsoby: implementací nastavení konkrétních hodnot definující kvalitu videa (velikost datového toku, rozlišení, snímková frekvence atd.) místo profilů – to by vyžadovalo rozsáhlé testování, zdali je konkrétní kombinace podporovaná kamerou – nebo přidáním více profilů. Pokud by bylo žádané implementovat přenos pouze jedné ze složek (jen přenos videa či jen přenos audia), muselo by se postupovat první metodou, neboť mezi profily Androidu nejsou žádné s tímto nastavením.

Pokud by byla implementována vyšší kvalita streamu, stála by za zvážení implementace „full screen“ přehrávání (přehrávání přes celou plochu displeje) na straně klienta. Vyžádalo by si to přepracování modulu pro přehrávání videa, neboť docílit přehrávání přes celou obrazovku vyžaduje otevření nové aktivity[1] (bylo by tedy třeba vyřešit i ovládací prvky streamu).

Vhodným rozšířením by také mohla být implementace časového nahrávání – klient si na serveru nastaví čas, kdy se má i bez jeho připojení začít nahrávat. Po skončení nahrávání by si pak klient mohl zpětně přehrát stream nahraný v minulosti. Pro toto rozšíření by bylo třeba implementovat další úložiště „nad“ současným úložištěm (které slouží jen jako „cache“) a ošetřit situace, kdy se při nahrávání připojí klient, který chce vidět live stream.

Z testování stability vyplynulo také jedno možné rozšíření – při přerušení spojení by bylo vhodné implementovat možnost automaticky znovu spustit server jakmile by se zařízení opět připojilo k síti.

[1] V současném stavu je přehrávač vázán na aktivitu, ve které byl stream spuštěn.

7 Závěr

Úkolem práce bylo prozkoumat oblast multimédií na platformě Android, vybrat vhodné knihovny pro práci s multimediálními daty, prozkoumat oblast streamování multimédií a navrhnout a vytvořit aplikaci pro streamování videa a zvuku mezi mobilními zařízeními.

Výsledkem práce je aplikace streamující video a audio principem posílání malých video souborů z vysílacího zařízení (serveru) ke klientovi. Proces streamování je konfigurovatelný z klientské i serverové stanice.

Výsledná aplikace je vhodná pro účely monitorování různých objektů či situací, u kterých lze využít jednosměrný přenos s krátkým zpožděním (od 10 sekund do 25 sekund) a nevadí malé výpadky v přenosu (v řádu stovek milisekund).

Seznam použitých zkratek

- AAC – Advanced Audio Coding
- AAC ELD – Advanced Audio Coding Enhanced Low Delay
- AAC LC – Advanced Audio Coding Low-Complexity
- AMR-NB – Adaptive Multi-Rate Narrowband
- AMR-WB – Adaptive Multi-Rate Wideband
- API – Application Programming Interface, rozhraní pro programování aplikací
- AVI – Audio Video Interleave
- BMP – Windows Bitmap
- FLAC – Free Lossless Audio Codec
- GIF – Graphics Interchange Format
- H.263 – algoritmus pro kompresi videa z rodiny H.26X
- H.264 – MPEG-4 AVC (Advanced Video Coding)
- H.265 – viz HEVC
- HE-AAC – High-Efficiency Advanced Audio Coding
- HEVC – High Efficiency Video Coding
- HW – Hardware

- IDE – Integrated Development Environment, vývojové prostředí
- JPEG – Joint Photographic Experts Group, grafický formát, rovněž organizace stojící za tímto formátem
- LZMA – Lempel–Ziv–Markov chain algorithm
- LZW – Lempel-Ziv-Welch
- MKV – Matroska Video
- MMS – Multimedia Messaging Service
- MP3 – MPEG-2 Audio Layer III
- OS – Operační systém
- PNG – Portable Network Graphics
- RAM – Random-access memory
- RAR – Roshal Archive
- RLE – Run-length encoding
- RTCP – RTP Control Protocol
- RTP – Real-time Transport Protocol
- RTMP – Real Time Messaging Protocol
- RTSP – Real Time Streaming Protocol
- SDK – Software Development Kit
- UI – User Interface
- URI – Unified Resource Identifier, jednotný identifikátor zdroje
- VR – Virtual Reality
- WAVE – Waveform Audio File Format

Seznam obrázků

Seznam tabulek

Literatura

[1] The Tech Terms Computer Dictionary *Android Definition* [online], cit[2015-19-12],
`http://techterms.com/definition/android`

[2] Android Developers *Android Studio and SDK Tools* [online], cit[2015-20-12],
`http://developer.android.com/sdk/index.html`

[3] Android Developers *Dashboards* [online], cit[2015-12-14],
`http://developer.android.com/about/dashboards/index.html`

[4] Android Developers *Android Studio and SDK Tools* [online], cit[2015-20-12],
`http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels`

[5] AndroidPIT *Android 6.0 Marshmallow: all the key features explained* [online], cit[2015-20-12],
`https://www.androidpit.com/android-m-release-date-news-features-name`

[6] International Data Corporation *Smartphone OS Market Share 2015, 2014, 2013, and 2012* [online], cit[2015-11-11],
`http://www.idc.com/prodserv/smartphone-os-market-share.jsp`

[7] The Tech Terms Computer Dictionary *Encoding Definition* [online], cit[2015-19-12],
`http://techterms.com/definition/encoding`

[8] The Tech Terms Computer Dictionary *Codec Definition* [online], cit[2015-19-12],

http://techterms.com/definition/codec

[9] Webopedia: Online Tech Dictionary *What is container format?* [online], cit[2015-19-12], http://techterms.com/definition/codec

[10] MediaCollege.com *Video Resolution* [online], cit[2016-20-04], http://www.mediacollege.com/video/resolution/

[11] Encoding.com *Understanding bitrates in video files* [online], cit[2016-20-04], http://help.encoding.com/knowledge-base/article/understanding-bitrates-in-video-files/

[12] TechTarget *What is frame rate?* [online], cit[2016-20-04], http://searchnetworking.techtarget.com/definition/frame-rate

[13] TechTarget *What is sample rate?* [online], cit [2016-20-04], http://whatis.techtarget.com/definition/sample-rate

[14] LI, Ze-Nian, DREW, Mark S., *Fundamentals of multimedia.*, Pearson Prentice Hall, 2004, cit [2016-20-04]

[15] Android Developers *Supported Media Formats* [online], cit[2015-10-12], http://developer.android.com/guide/appendix/media-formats.html

[16] Fraunhofer IIS *HE-AAC, HE-AAC v2* [online], cit[2015-20-12], http://www.iis.fraunhofer.de/en/ff/amm/prod/audiocodec/audiocodecs/heaac.html

[17] Fraunhofer IIS *AAC-ELD Family* [online], cit[2015-20-12], http://www.iis.fraunhofer.de/en/ff/amm/prod/kommunikation/komm/aaceld.html

[18] VoiceAge *AMR (Adaptive Multi-Rate) standard* [online], cit[2015-20-12], http://www.voiceage.com/AMR-NB.AMR.html

[19] VoiceAge *AMR-WB/G.722.2* [online], cit[2015-20-12], http://www.voiceage.com/AMR-WB.G.722.2.html

[20] Xiph.org Foundation *FLAC - Free Lossless Audio Codec* [online], cit[2015-22-12], https://xiph.org/flac/

[21] McGill University *Standard MIDI file format* [online], cit[2015-20-12], http://www.music.mcgill.ca/ ich/classes/-mumt306/StandardMIDIfileformat.html

[22] Xiph.org Foundation *Vorbis audio compression* [online], cit[2015-20-12], https://xiph.org/vorbis/

[23] Digital Preservation (Library of Congress) *WAVE Audio File Format* [online], cit[2015-22-12], http://www.digitalpreservation.gov/formats/fdd/fdd000001.shtml

[24] Xiph.org Foundation *Opus Codec* [online], cit[2015-22-12], https://www.opus-codec.org/

[25] FileFormat.Info *JPEG File Interchange Format* [online], cit[2015-24-12], http://www.fileformat.info/format/jpeg/egff.htm

[26] FileFormat.Info *GIF* [online], cit[2015-24-12], http://www.fileformat.info/format/gif/egff.htm

[27] FileFormat.Info *PNG* [online], cit[2015-24-12], http://www.fileformat.info/format/png/egff.htm

[28] FileFormat.Info *Microsoft Windows Bitmap* [online], cit[2015-24-12], http://www.fileformat.info/format/bmp/egff.htm

[29] Google Developers *A new image format for the Web | WebP* [online], cit[2015-24-12], https://developers.google.com/speed/webp/

[30] Movavi Online *H263 video format* [online], cit[2015-26-12], http://online.movavi.com/format-h263.html

[31] H264info.com *What is H.264* [online], cit[2015-26-12], http://www.h264info.com/h264.html

[32] Android Developers *MediaPlayer* [online], cit[2015-19-12], http://techterms.com/definition/codec http://developer.android.com/reference/android/media/MediaPlayer.html

[33] Android Developers *Media Playback* [online], cit[2015-10-12], http://developer.android.com/guide/topics/media/mediaplayer.html

[34] Android Developers *Camera* [online], cit[2015-12-12], http://developer.android.com/guide/topics/media/camera.html

[35] Envato Tuts+ Code Tutorial *Streaming Video in Android Apps* [online], cit[2015-12-12], http://code.tutsplus.com/tutorials/streaming-video-in-android-apps-cms-19888

[36] Amazon forum *Using Camera2 to replace Deprecated Camera API* [online], cit[2015-14-12], https://forums.developer.amazon.com/forums/thread.jspa?threadID=4836

[37] Android Developers *MediaRecorder* [online], cit[2015-26-12], http://developer.android.com/reference/android/media/MediaRecorder.html

[38] Android Developers *googlesamples/android-Camera2Video* [online], cit[2015-26-12], https://github.com/googlesamples/android-Camera2Video

[39] Android Developers *Camera* [online], cit[2015-26-12], http://developer.android.com/guide/topics/media/camera.html

[40] Technopedia *What is Streaming Media?* [online], cit[2016-05-01], https://www.techopedia.com/definition/14586/streaming-media

[41] Technopedia *What is Real Time Streaming Protocol (RTSP)?* [online], cit[2016-05-01], https://www.techopedia.com/definition/4753/real-time-streaming-protocol-rtsp

[42] Vitamio *Vitamio - The most professional multimedia framework for Android and iOS* [online], cit[2016-06-01], https://www.vitamio.org/en/

[43] QuickBlox *QuickBlox Backend: cloud communication backend API as a service for mobile and web apps* [online], cit[2016-06-01], http://quickblox.com/

[44] Agilio *iOS Android RTMP RTSP Library - Streaming and Broadcasting Libraries* [online], cit[2016-06-01], http://www.realtimelibs.com/

[45] Simon Guigui *fyhertz/libstreaming* [online], cit[2016-06-01],

https://github.com/fyhertz/libstreaming

[46] Infrared5 *Red5 Media Server* [online], cit[2016-18-04], http://red5.org/

[47] Infrared5 *Red5 Pro* [online], cit[2016-18-04], http://red5pro.com/

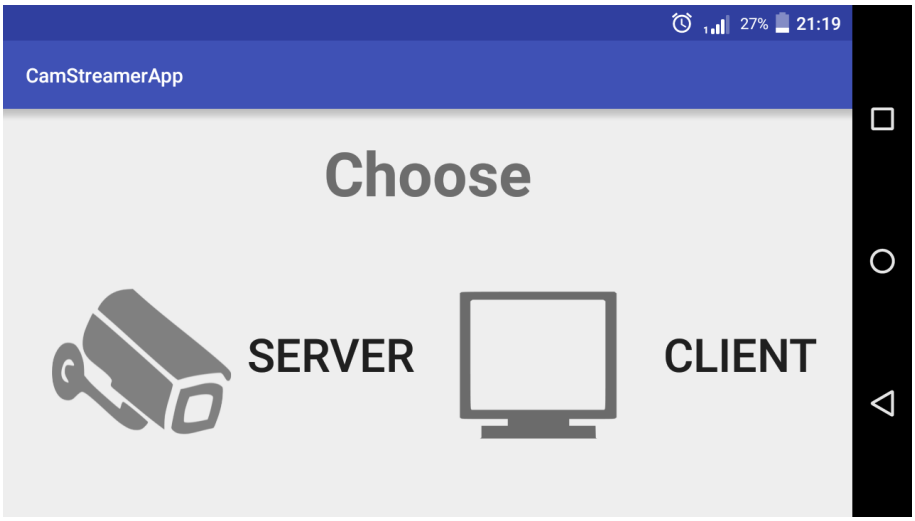
[48] Stack Overflow *Live-stream video from one android phone to another over WiFi* [online], cit[2016-12-06], http://stackoverflow.com/a/14432887

A Uživatelská dokumentace

Pro nainstalování je třeba na daném zařízení povolit instalaci z neznámých zdrojů:
Nastavení > *Zabezpečení* > zaškrtnout *Neznámé zdroje* – Povolit instalaci z neznámých zdrojů Poté lze již bez problémů nainstalovat aplikaci pomocí příslušného APK souboru (viz příloha C).

Aplikaci lze nainstalovat pouze na zařízení s Androidem verze *4.1* a vyšší.

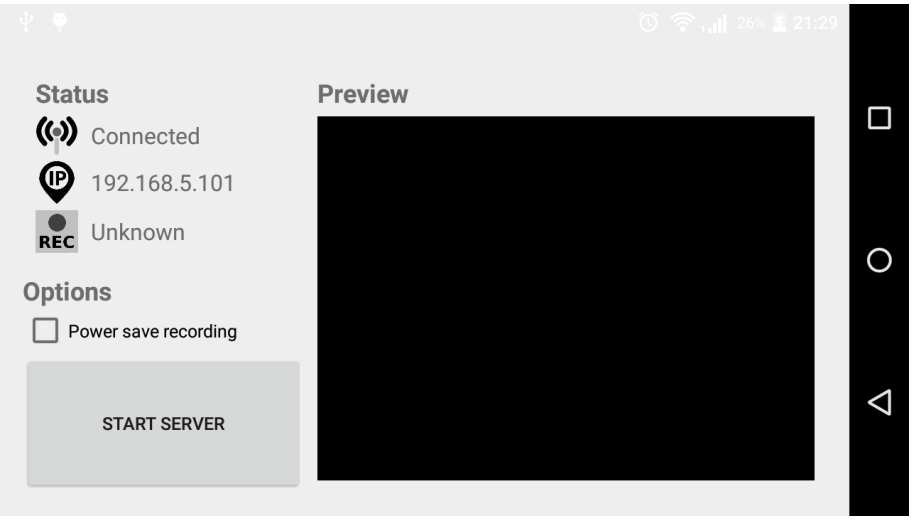
Při spuštění se zobrazí startovací aktivita (viz obrázek A.1), na které si můžeme zvolit, zda bude naše zařízení sloužit jako server, nebo jako klient.



Obrázek A.1: Startovací aktivita aplikace.

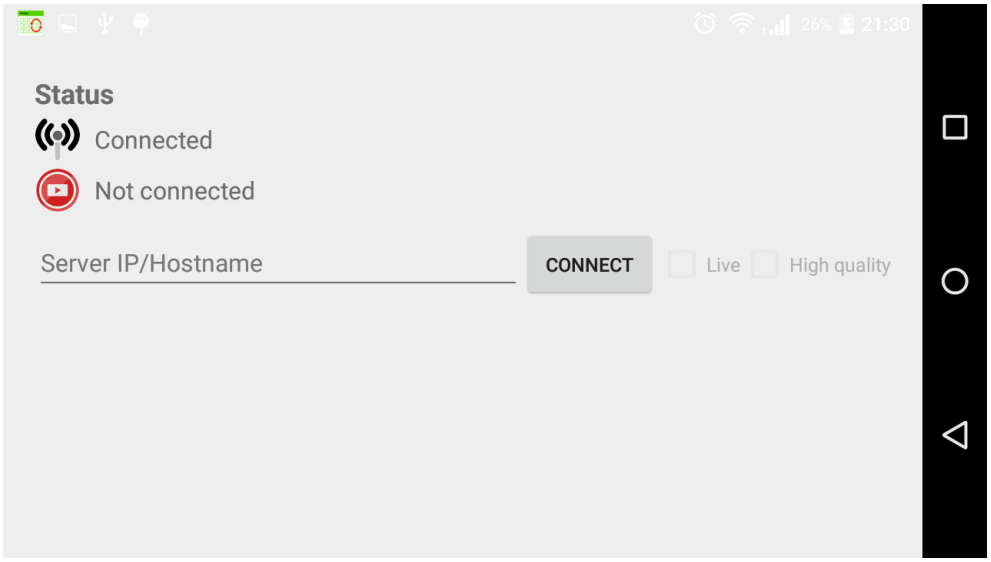
Pokud je vybrán server, zobrazí se nám aktivita (viz obrázek A.2), na které vidíme důležité informace potřebné k jeho spuštění a provozování (zda je zařízení připojeno, jakou má IP adresu a jaký je stav streamu), možnosti nastavení před spuštěním („Power save recording“ – nahrávání pouze pokud je klient připojen k serveru), tlačítko pro spuštění a zastavení serveru a plocha pro náhled kamery.

Po startu serveru se zobrazí náhled kamery (podle zvolené možnosti se začne nahrávat) a čeká se na připojení klienta. Po jeho připojení se automaticky začnou posílat nově nahrané soubory.



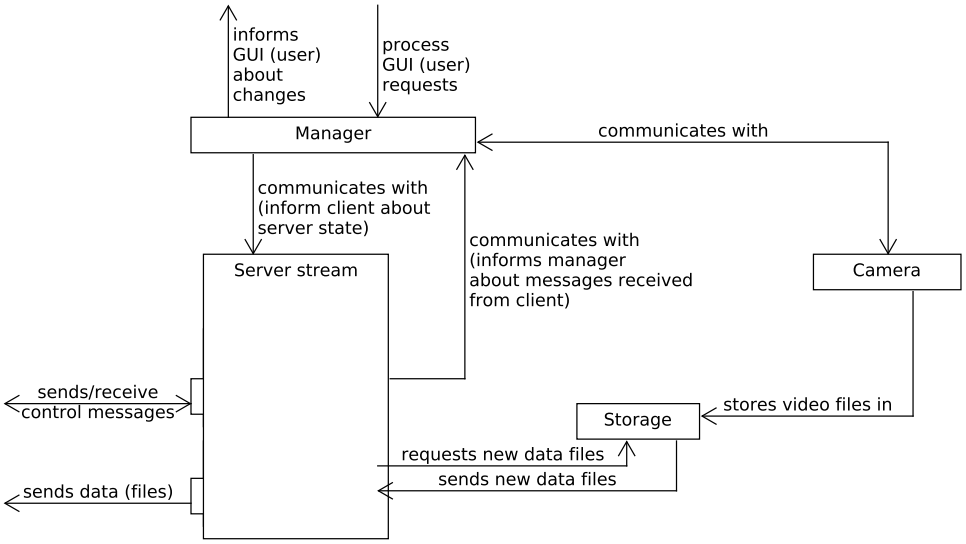
Obrázek A.2: Serverová aktivita aplikace.

Pokud bychom zvolili klientskou část (viz obrázek A.3), zobrazí se nám informace o připojení (připojení k síti a informace o stavu připojení k serveru) a řádka pro vložení IP adresy serveru. Po jeho vložení je třeba kliknout na tlačítko „Connect“ a klient se pokusí připojit k serveru. Pokud se zdaří, stav ve statusu se změní na „Connected“ a zpřístupní se nám prvky napravo od tlačítka určené k řízení přenosu. Zatrhnutím možnosti „Live“ se začnou ze serveru streamovat data. Zatržítko vedle s nápisem „High Quality“ slouží k volbě mezi streamem s nižší kvalitou (nezatrhnuté) a volbou s vyšší kvalitou (zatrhnuté). Volba se neprojeví okamžitě (na serveru se projeví s nahráváním dalšího video souboru).

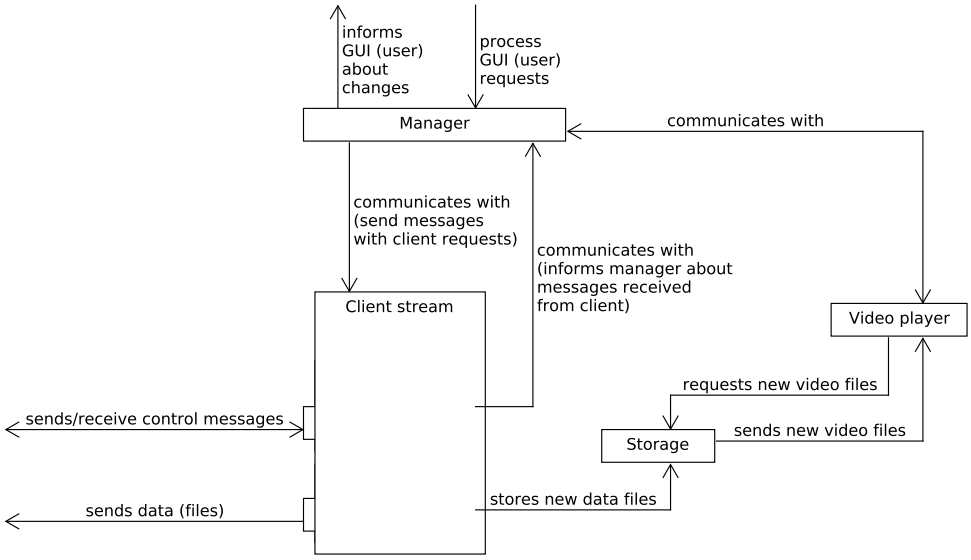


Obrázek A.3: Klientská aktivita aplikace (nepřipojená – bez video layoutu).

B Schéma modulů knihovny



Obrázek B.1: Schéma modulů z pohledu serveru.



Obrázek B.2: Schéma modulů z pohledu klienta.

C Obsah přiloženého DVD

Obsah přiloženého DVD:

- složka *bin* – obsahuje instalační soubor aplikace `CamStreamerApp.apk` a přeložený soubor s knihovnou `CamStreamerLib.aar`,
- složka `doc` – obsahuje `Zimmermann_BPINI.pdf` – dokument bakalářské práce,
- složka `javadoc` – obsahuje JavaDoc dokumentaci ke knihovně a aplikaci (v samostatných složkách),
- složka `src` – obsahuje zdrojové kódy knihovny a aplikace (v samostatných složkách) a rovněž zazipované projekty knihovny a aplikace,
- složka `tex` – obsahuje zdrojové kódy dokumentu bakalářské práce psané v LaTeXu,
- soubor `readme.txt` – zkopírovaný obsah této přílohy