VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MULTIMEDIÁLNÍ PŘEHRÁVAČ PRO ANDROID
MULTIMEDIA PLAYER FOR ANDROID

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jan Benedikt**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Štěpán Grabovský**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Jan Benedikt ***ID:*** 173609
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Multimediální přehrávač pro Android

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je realizace knihovny přehrávače multimediálního obsahu s podporou pokročilých služeb jako reklamních formátů, odesílání statistik, zabezpečení obsahu atd. V práci nejdříve definujte vybrané pokročilé služby přehrávače a proveďte analýzu již dostupných řešení. Na základě zjištěných informací poté realizujte funkční knihovnu přehrávače, která bude implementována v jednoduché testovací aplikaci. Celý projekt musí být vytvořen v jazyce Java/Kotlin.

DOPORUČENÁ LITERATURA:

[1] PHILLIPS, Bill, Chris STEWART a Kristin MARSICANO. Android programming: the Big nerd ranch guide. Third edition. Atlanta, GA: Big nerd ranch, 2017. ISBN 978-0-13470-605-4.

[2] SCHILDT, Herbert a Milan DANĚK. Mistrovství - Java. Brno: Computer Press, 2014. ISBN 978-80-251-4145-8.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: Ing. Štěpán Grabovský
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Diplomová práce se zabývá vytvořením pokročilého multimediálního přehrávače, podporující adaptivní stream, zabezpečení multimediálního obsahu a podporu přehrávání reklam. V první části jsou vysvětleny použité technologie. Následuje část s rozborem dostupných řešení a jejich vzájemné porovnání.
Teoretické poznatky jsou následně uplatněny ve vytváření knihovny přehrávače a zkušební aplikaci. Knihovna aplikace vychází z dostupného přehrávače ExoPlayer. Celá aplikace je vytvořena pro operační systém Android pomocí jazyka Java.

KLÍČOVÁ SLOVA

Android, MP4, Adaptivní stream, HLS, MPEG-DASH, DRM, Widewine, FairPlay, Playready, HLS-AES, Google Analytics, VMAP, VPAID, VAST, ExoPlayer

ABSTRACT

This thesis aims to create an advanced multimedia player with support of an adaptive stream, securing of the multimedia content and support for playing advertisement. The first part explains the used technologies followed by an analysis of available solutions and the comparison between the two.
The theoretical framework is then used to create a player library and a testing application. The player library is based on the framework of ExoPlayer application. Whole application is created in Java for the operation system Android OS.

KEYWORDS

Android, MP4, Adaptive stream, HLS, MPEG-DASH, DRM, Widewine, FairPlay, Playready, HLS-AES, Google Analytics, VMAP, VPAID, VAST, ExoPlayer

BENEDIKT, Jan. *Multimediální přehrávač pro Android*. Brno, 2019, 87 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Štěpán Grabovský

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Multimediální přehrávač pro Android“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Štěpánovi Grabovskému za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále děkuji společnosti SVSys s.r.o. za vypsání této diplomové práce.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Kapacita linek přístupových sítí se za posledních několik let skokově zvýšila, a to umožnilo vznik nových typů služeb. Jednou z nich je i poskytování multimediálního obsahu. Multimediální obsah je pomocí sítě streamován do mnoha nejrůznějších druhů zařízení, která umožňují připojení k síti Internet. Jednou z nejdynamičtěji rozvíjejících se platforem jsou chytré mobilní telefony. Díky chytrým mobilním telefonům je dnes možné konzumovat multimediální obsah v podstatě odkudkoliv. A díky tomu stále více roste obliba streamování multimediálního obsahu přímo k uživateli. Dokazují to nové služby online video půjčoven (Netflix, HBO GO, ...), hudební streamovací služby (Spotify, Deezer, Tidal, ...), video servery s vlastní produkcí obsahu (Seznam TV, Mall.tv, ale i Youtube) a poskytovatelé internetové televize (Kuki, O2 TV, Lepší TV, ...). Všechny služby spojují technologie umožňující poskytování obsahu skrze síť Internet. Většina jmenovaných služeb nabízí vlastní řešení přehrávače ve formě aplikace na telefon. Přehrávače disponují pokročilými funkcemi, které zajišťují snadnou obsluhu, uživatelské pohodlí a zejména zabezpečení vysílaného obsahu proti jeho duplikování a dalšímu šíření. Standardem je dnes i přizpůsobení přehrávání podmínkám internetového připojení. Z hlediska poskytovatele je výhodné, když přehrávač disponuje funkcionalitou vytváření statistik užívání přehrávače a v některých případech zobrazováním reklamního obsahu.

Tato práce se zabývá návrhem a vytvořením pokročilého multimediálního přehrávače pro platformu Android. Jejím výstupem je knihovna přehrávače podporující přehrávání streamovaného obsahu, implementuje požadované ochrany vysílaného obsahu a funkce pro přehrávání reklamního obsahu.

První kapitola práce se věnuje technologiím, které jsou využívány pro streamování multimediálního obsahu v síti Internet. Následují technologie zabezpečení obsahu, kde je popsáno několik dostupných řešení od různých vývojářů. Kapitola pokračuje popisem služby Google Analytics, včetně popisu systémů pro doručování reklamního obsahu klientovi.

Druhá kapitola se zabývá popisem nalezených frameworků multimediálních přehrávačů pro operační systém Android. Na základě popisu podporovaných technologií je nejvhodnější z frameworků vybrán a dále je podrobně popsán.

V rámci třetí kapitoly je v textu vytvořen konkrétní návrh multimediálního přehrávače, který vychází ze zvoleného frameworku.

Následuje kapitola obsahující obecný popis publikování aplikace v obchodě Play Store. Ve čtvrté části jsou uvedeny podmínky pro uvedení a postup přípravy aplikace k vydání.

V poslední kapitole je popsán vývoj knihovny přehrávače společně s vývojem testovací aplikace. Poslední kapitola je rozdělena na části podle funkcionality přehrá-

vače. Kapitola začíná částí o přípravě vývoje. V rámci přípravy je uvedena základní terminologie vývoje pro operační systém Android a struktura projektu ve vývojovém prostředí Android Studio. Kapitola pokračuje popisem vývoje knihovny přehrávače. Sekce o vývoji knihovny je rozdělena od základních funkcí po pokročilejší. Po popisu vývoje knihovny přehrávače pokračuje kapitola sekcí o vývoji testovací aplikace. Kapitola je ukončena návodem k provázání knihovny přehrávače s hostující aplikací.

1 Technologie multimediálního přehrávače

V následujících několika sekcích budou popsány technologie, které úzce souvisí s řešením problematiky proudového vysílání audio a video obsahu. První pasáž technologické části bude věnována video formátům a jejich způsobu přenosu sítí Internet. Následovat bude část zaměřená na zabezpečení obsahu v rámci multimediálních dat. Poslední pasáž teoretické části bude věnována metodě zpracování statistiky užívání aplikací v systému Google Analytics a metodám poskytování reklamních formátů.

1.1 Formáty videa a jejich způsoby přenosu počítačovou sítí

1.1.1 MP4

Jedná se o formát multimediálního kontejneru. Tento formát je definovaný v standardu ISO/EIC 14496-14:2003, nyní v aktualizované verzi ISO/EIC 13818-1:2018 [1], dále také uváděný jako MPEG-4 část 4[1]. Základ tohoto multimediálního kontejneru tvoří MPEG-4 12. část[1], vytvořený skupinou Moving Pictures Experts Group. Výhodou kontejneru MP4 je jeho podpora pro více zvukových stop a zároveň více stop s titulky v rámci jednoho multimediálního souboru. Tento kontejner je také vhodný pro proudové vysílání.

Multimediální kontejner MP4 nejčastěji využívá kompresi MPEG-4 pro obraz i pro zvuk. Předností tohoto kontejneru oproti starším formátům (např. AVI) je lepší práce s kompresí obrazového a zvukového obsahu. Dále je schopen pojmout mnoho rozličných druhů multimediálního obsahu, jako jsou titulky, informace o obsahu kontejneru a uživatelské informace[2]. Pro zvukové stopy jsou využívány kodeky jako jsou například MP3 nebo AAC[1].

Soubor typu MP4 je rozdělen na stopy. Každá stopa představuje časovou posloupnost prezentačních jednotek. Prezentační jednotka se v každé stopě nazývá vzorek. Vzorek v souboru může představovat například rámec informací o videu, zvuku, nebo o metadatech[2].

Vizuální data v kontejneru MP4 vychází nejčastěji z MPEG-4. Tento formát je rozšířením původního standardu MPEG-1[3]. Videoklip určité délky obsahuje skupiny snímků, neboli sekvence. Tyto sekvence obsahují samotné snímky. Snímky můžeme rozdělit do čtyř typů. Jedná se o snímky typu: I, P, B a D. Snímky obsahují tzv. slice. Slice je dále tvořena makrobloky. Makrobloky jsou skupinou bloků.[3]

Není však pravidlem, aby kontejener MP4 obsahoval pouze videodata typu MPEG-4. Samotný výrobce software může svobodně zvolit obsah kontejneru a libovolně ho upravovat pro vlastní potřeby.

1.1.2 MPEG-TS

Multimediální kontejner s názvem MPEG-TS, dále také uváděný jako MTS nebo TS, je definovaný ve standardu ISO/EIC 13818-1[1]. Multimediální kontejner MPEG-TS byl vytvořen pro přenos audia, videa a programových a systémově informačních protokolů (*Programové a systémově informační protokoly – Program and System Information Protocol* (PSIP)). Je vhodný pro *Digitální televizní vysílání – Digital Video Broadcasting* (DVB) používané v Evropě, *Pokročilé normy televizních systémů – Advanced Television Systems Committee standards* (ATSC) používané na Americkém kontinentu (konkrétně v Kanadě, Spojených Státech Amerických a Mexiku) a *Televize přes internetový protokol – Internet Protocol television* (IPTV).

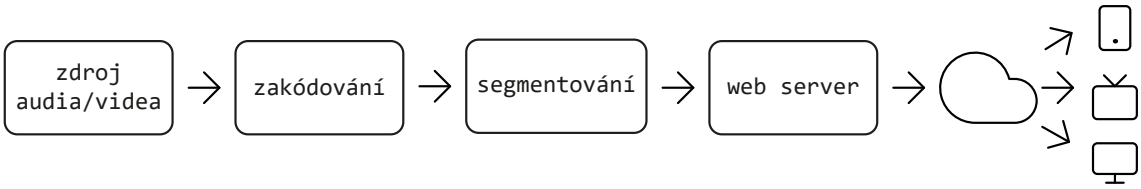
Kontejner MPEG-TS zapouzdřuje elementární streamy. Je tedy možné mít v jednom kontejneru více video a audio stop, včetně elektronického programového průvodce (EPG). MPEG-TS je dále vybaven korekcí chyb, které nastaly během přenosu a synchronizací stop pro udržení integrity přenosu.[4] Tento standard je navržen pro méně spolehlivé přenosy médií. Jedná se například o dříve zmíněné pozemní vysílání (DVB nebo ATSC) a o televizní vysílání skrze IP protokol.[4]

1.1.3 HLS

HTTP Live Streaming (HLS) je protokol pro streamování multimediálních dat. Pro přenos po síti využívá protokol HTTP. Protokol HLS byl vyvinut společností Apple a byl vydán jako součást multimediálního balíku QuickTime[5]. Protokol HLS pracuje na principu segmentace multimediálních dat a pomocí webového serveru umožňuje jejich distribuci.

Výhodou protokolu HLS je možnost použití klasického webového serveru pro distribuci streamu do počítačové sítě.

Multimediální obsah vstupuje do enkodéru, kde je zakódován do MPEG-2 transportního streamu. Tento stream je dále rozdělen na segmenty. Segmenty jsou podle pořadí uspořádány do tzv. playlistu. Playlist obsahuje seznam všech dostupných segmentů streamu, včetně jejich adresy na webovém serveru. Klientská aplikace na základě získaného playlistu dotazuje webový server na jednotlivé segmenty multimediálního obsahu. Celý popis lze vidět na obr. 1.1. Protokol HLS poskytuje funkci videa na vyžádání (video on demand). Tato funkce poskytuje klientovi možnost přeskakovat ve video obsahu na různá místa v čase. Klient si jednoduše vyžádá segment z určitého časového bodu ve video obsahu a webový server od tohoto bodu zahájí stream.[5]



Obr. 1.1: Postup přípravy streamu HLS

1.1.4 MPEG–DASH

Jedná se o adaptivní stream skrze protokol TCP. Hlavní rysy protokolu MPEG–DASH jsou podobné jako u protokolu HLS. I zde je multimediální vstup rozdělen na části, které následně webový server na základě dotazů klienta vysílá na klientskou stranu.[6] Samotný stream je dostupný v mnoha verzích s různou velikostí datového toku. Protokol MPEG–DASH následně na základě propustnosti trasy k uživateli upravuje výběr zdroje se specifickou velikostí datového toku. Celý tento výběr se děje na klientské straně. Aby k této akci mohlo dojít, serverová strana musí nabídnout adekvátní alternativy různých velikostí datového toku.[6] MPEG–DASH je prvním streamovacím protokolem s adaptvní úpravou vysílání, který byl uznán jako mezinárodní standard[7]. Na rozdíl od HLS je MPEG–DASH nezávislý na kodeku, který přenáší. Je tedy schopný přenášet jakýkoliv kodek jako: H.264, H.265, VP9, apod. . . [8].

1.2 Zabezpečení multimediálního obsahu

1.2.1 DRM

Správa digitálních práv – Digital Rights Management (DRM) je systém na ochranu multimediálních dat. Tento systém vychází z autorského práva, či licence, kdy autor chce ochránit své dílo před neoprávněným šířením a jeho produkcí.[9][10] Jedná se o soubor technických postupů, jejichž účelem je ovládat, či omezovat užívání multimediálních dat.[9][10] V rámci této práce se autor zaměří na použití DRM u videostreamů.[9][10] Nejčastěji využívané jsou následující tři standardy:

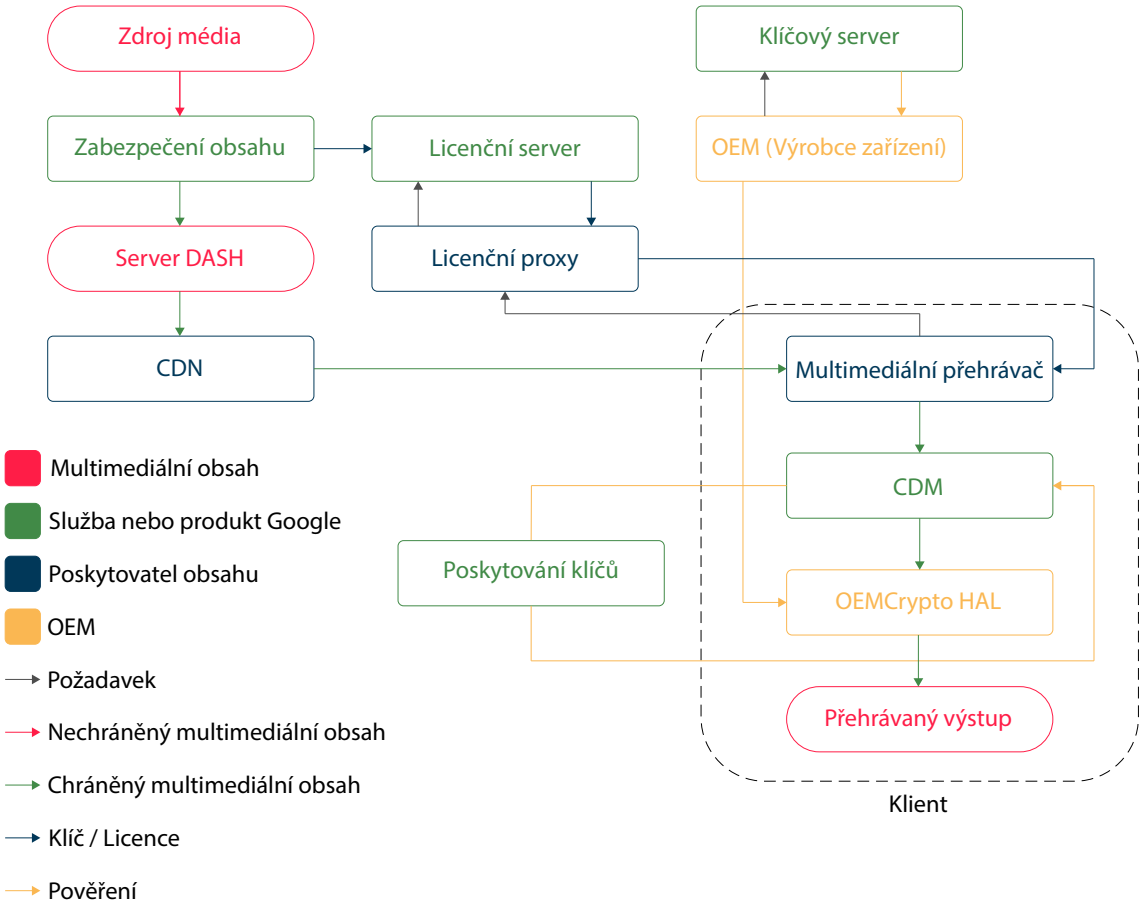
1. widevine,
2. playready,
3. fairplay.

Každý standard pochází od jiného výrobce a jeho použití je mírně odlišné. V základu jsou však všechny tři standardy stejné. Aby byla umožněna kompatibilita napříč spektrem zařízení a software, mají všechna řešení v základu stejná, standardizovaná

pravidla šifrování s adaptivním streamem. U prvních dvou je implementována společná strategie šifrování streamu. Jedná se o využití standardu MPEG–DASH s použitím společného šifrování (MPEG-CENC). U MPEG-CENC je využíván *Standard pokročilého šifrování – Advanced Encryption Standard* (AES) s velikostí 128 bitů. Využívá se v jednom z provozních režimů blokové šifry: *Čítač – Counter* (CRT), nebo *Řetězení šifrových bloků – Cipher Block Chaining* (CBC). U třetího (Fairplay) není využit standard MPEG-CENC, ale pouze využívá šifrování typu AES, opět s velikostí 128 bitů v módu CBC.

Widevine

Řešení Widevine bylo vyvinuto společností Google, Inc. Jeho podpora je implementována do většiny jeho produktů, které umožňují přehrávat některou z variant multimediálního obsahu (například Chromium, Android, apod...). Druhá výhoda tohoto standardu je podpora u běžně dostupných „chytrých“ televizorů (jmenovitě: LG, Panasonic, Philips, Samsung, Sharp Aquos a Sony Bravia).[11]

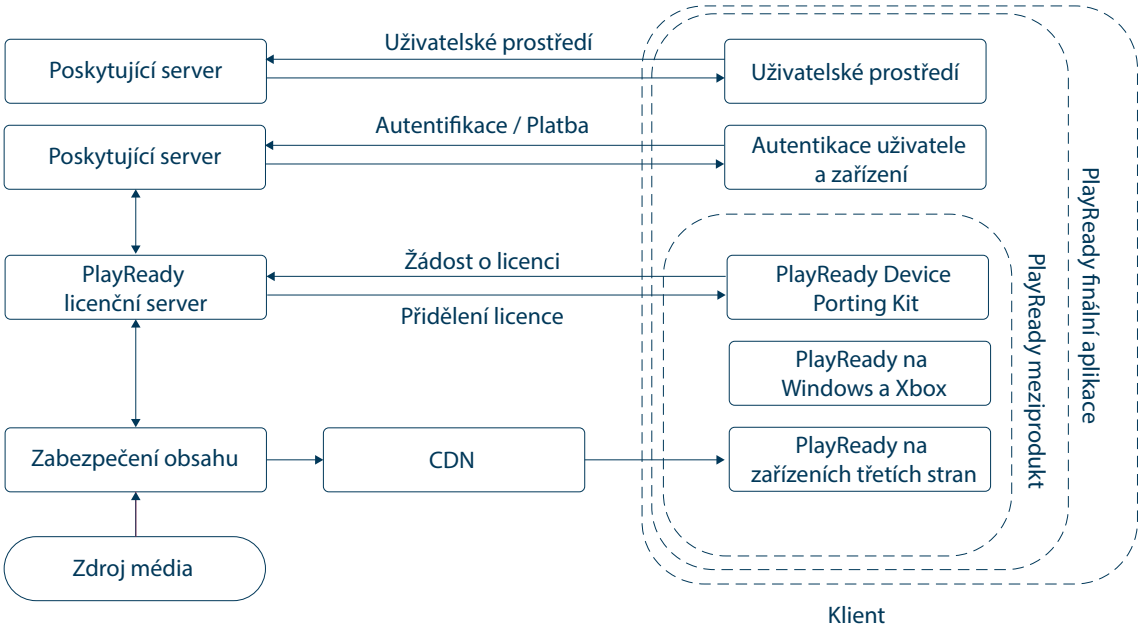


Obr. 1.2: Ukázka komunikace klienta s licenčním serverem Widevine[11]

Z diagramu, na obr. 1.2 vyplývá, že k přehrání multimediálního obsahu je zapotřebí minimálně tří zúčastněných stran. Jedná se o poskytovatele multimediálního obsahu, *Původní výrobce dále prodávaného produktu – Original Equipment Manufacturer* (OEM) a poskytovatele technologie Widevine (v tomto případě společnost Google, Inc.). Na začátku procesu se nachází nechráněný multimediální obsah. Tento obsah je zakódován pomocí klíče z licenčního serveru. Zakódovaný obsah je dále zpracován technologií MPEG–DASH. Pomocí *sítě pro doručování obsahu* (CDN) je distribuován ke klientovi. Zakódovaný obsah je předán multimediálnímu přehrávači, který je autorizován u licenčního serveru skrze licenční proxy. Na základě této autorizace je přehrávači poskytnuta licence, která se společně se zakódovaným obsahem předá *správě dekódování obsahu* CDM. Následně je připojen unikátní klíč od výrobce OEM a dále je zabezpečený obsah rozkódován. Po úspěšném rozkódování je možné obsah přehrát.

PlayReady

Řešení PlayReady je vyvíjeno společností Microsoft Corporation. Nejlepší podporu tedy lze očekávat v jejich produktech. Zejména se jedná o operační systém Microsoft Windows, herní konzole XBox a internetový prohlížeč Microsoft Edge a starší předchůdce Microsoft Internet Explorer. Výrobce dále deklaruje podporu u operačního systému Google Android a Apple iOS. Řešení PlayReady je také podporováno prohlížeči Google Chrome a Mozilla Firefox. Řešení PlayReady na rozdíl od společných znaků s ostatními technologiemi DRM nabízí navíc implementaci standardů jako jsou: *Formát aplikace společného média – Common Media Application Format* (CMAF), *Ekosystém digitálního obsahu zábavy – Digital Entertainment Content Ecosystem* (DECE LLC) a *Protokol o autorizaci multimédií online – Online Multimedia Authorization Protocol* (OMAP).[12]

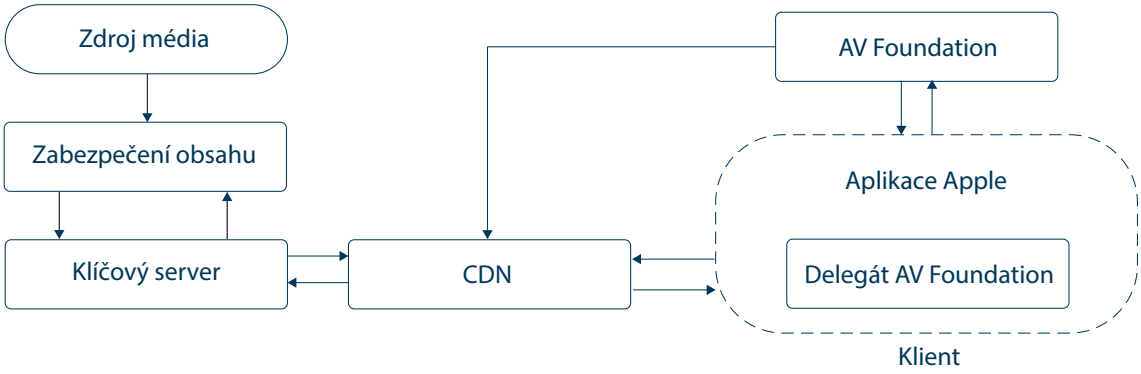


Obr. 1.3: Ukázka komunikace klienta s licenčním serverem PlayReady[12]

Jak lze vyčíst z obr. 1.3, řešení je podobné předešlému konkurenčnímu řešení Widevine od společnosti Google, Inc. I zde zde hraje hlavní roli licenční server, který komunikuje s poskytujícím serverem. Poskytující server má za úkol „komunikaci“ s uživatelem – ověřuje licenci, nebo platbu a poskytuje data pro výběr přehrávaného obsahu. Při výběru obsahu proběhne kontrola licence/platby a na základě kladného výsledku je kontaktován modul „zabezpečeného obsahu“, který spustí kódovaný stream. Na klientské straně modul PlayReady stream rozkóduje a předá přehrávači k přehrání.[12]

FairPlay

Posledním řešením, na které se autor zaměřil, je produkt společnosti Apple, Inc. Jedná se o řešení s názvem FairPlay. Znovu je zde podpora především vlastních produktů, zejména operačního systému iOS a macOS. Z historického hlediska FairPlay bylo poprvé využito pro službu iTunes. Tato služba sloužila k distribuci hudebního, video a knižního obsahu. V dnešní době již tuto ochranu DRM k distribuci nevyužívá[13]. Jako jediné řešení nevyužívá MPEG–DASH, ale HLS, které společnost Apple také vyvinula. I u řešení FairPlay je využit šifrovací algoritmus AES s délkou klíče 128 bitů, odkud vychází standard HLS-AES, který bude popsán později. Pro přenos je využit multimediální kontejner MP4 a zvuková stopa AAC. Klíč pro rozšifrování obsahu je nazván jako „uživatelský klíč“. Při registraci nového zařízení do služby iTunes je tomuto zařízení vygenerován právě tento „uživatelský klíč“. Při žádosti o přehrávaný obsah dojde ke stažení šifrovaných dat, která jsou následně pomocí „uživatelského klíče“ rozšifrována.



Obr. 1.4: Ukázka komunikace klienta s licenčním serverem pomocí technologie FairPlay[14]

Celý postup je následující a vychází z obr. 1.4: Klient zažádá „AV Foundation“ o přehrání určitého obsahu. „AV Foundation“ stáhne manifest ve formátu *.m3u*, který obsahuje „klíč obsahu“. Dále dojde k žádosti od delegáta k „AV Foundation“ o tento klíč. Po poskytnutí klíče dojde k žádosti na streamovací server. Vygenerovaný manifest je společně s klíči zaslán na „klíčový server“. Server odpovídá streamem, který je zakódovaný. Tento zakódovaný stream je následně pomocí „uživatelského klíče“ rozšifrován a přehrán klientovi.[14]

1.2.2 HLS-AES

Stejně jako u předchozích tří ochran streamovaného obsahu, i u nativního HLS lze využít šifrování obsahu. V principu šifrování funguje na principu přidání další

vrstvy na straně serveru a klienta, která se stará o šifrování, respektive o rozšifrování multimediálního obsahu. Oproti výše zmíněným ochranám DRM nepotřebuje žádný licenční server. Server šifruje vzorky multimediálního obsahu pomocí AES s klíčem o velikosti 128 bitů. Tento šifrovaný obsah je dále distribuován ke klientovi. Mimo zabezpečený multimediální obsah je nutné, aby klient získal klíč, pomocí kterého zabezpečený obsah dešifruje. Pro distribuci tohoto klíče je vhodné zvolit dostatečně zabezpečený komunikační kanál, aby nedošlo k úniku klíče k neoprávněné osobě.[15]

Přenos zabezpečeného obsahu může probíhat ve třech variantách.[15] První variantou je využití stejného šifrovacího klíče pro celý multimediální stream. Klíč tedy zůstává společný pro všechny vysílané vzorky po celou dobu streamování. Tato varianta je ta nejméně bezpečná.[15] Následuje varianta, kdy je pro každý vzorek použit nový šifrovací klíč. Nevýhodou varianty s unikátním klíčem pro každý vzorek je vysoká režie na straně serveru i na straně klienta. Při každém vzorku je nutné bezpečně dopravit klíč ke klientovi.[15] Třetí varianta spočívá ve variabilní změně šifrovacího klíče a je tedy kombinací dvou předešlých variant. Šifrovací klíč je společný pro n vzorků a po době m se změní. Doba m je dána nastavením streamovacího serveru, kdy odpovídá hodnotě n odeslaných vzorků o délce t.[15] Distribuce klíče je zajištěna pomocí playlistu (viz výpis 1.1), kde je v sekci „#EXT-X-KEY:“ uveden odkaz k jeho získání. Dále lze z výpisu vypozorovat, že je uvedena i metoda šifrování obsahu, a to AES-128.[15]

Výpis 1.1: Ukázka playlistu s klíčem typu m3u pro HLS-AES[16]

```
1  #EXTM3U
2
3  #EXT-X-TARGETDURATION:10
4
5  #EXT-X-KEY:METHOD=AES-128,
6  URI="https://keys.source.test/key-1.key"
7
8  #EXTINF:10.0,
9  http://media.source.test/video1/sequence-1-segment-1.ts
10 #EXTINF:10.0,
11
12 #EXTINF:10.0,
13 http://media.source.test/video1/sequence-2-segment-1.ts
14 http://media.source.test/video1/sequence-2-segment-2.ts
15
16 #EXT-X-ENDLIST
```

1.3 Statistiky užívání a poskytování reklamních formátů

1.3.1 Google Analytics

Pro pohodlnou správu aplikace je vhodné dostávat zpětnou vazbu od uživatelů. Zpětná vazba může být jak aktivní, tak pasivní.

Aktivní zpětná vazba spočívá v přímém kontaktu s uživatelem, který může například pomocí e-mailu hlásit nedostatky aplikace a případně její chyby. Toto chování lze vyvolat pomocí chybových hlášení, kdy je uživatel nabádán k odeslání chybové zprávy pomocí již výše zmíněného e-mailu, či jiné formy přímého, elektronického komunikačního kanálu.

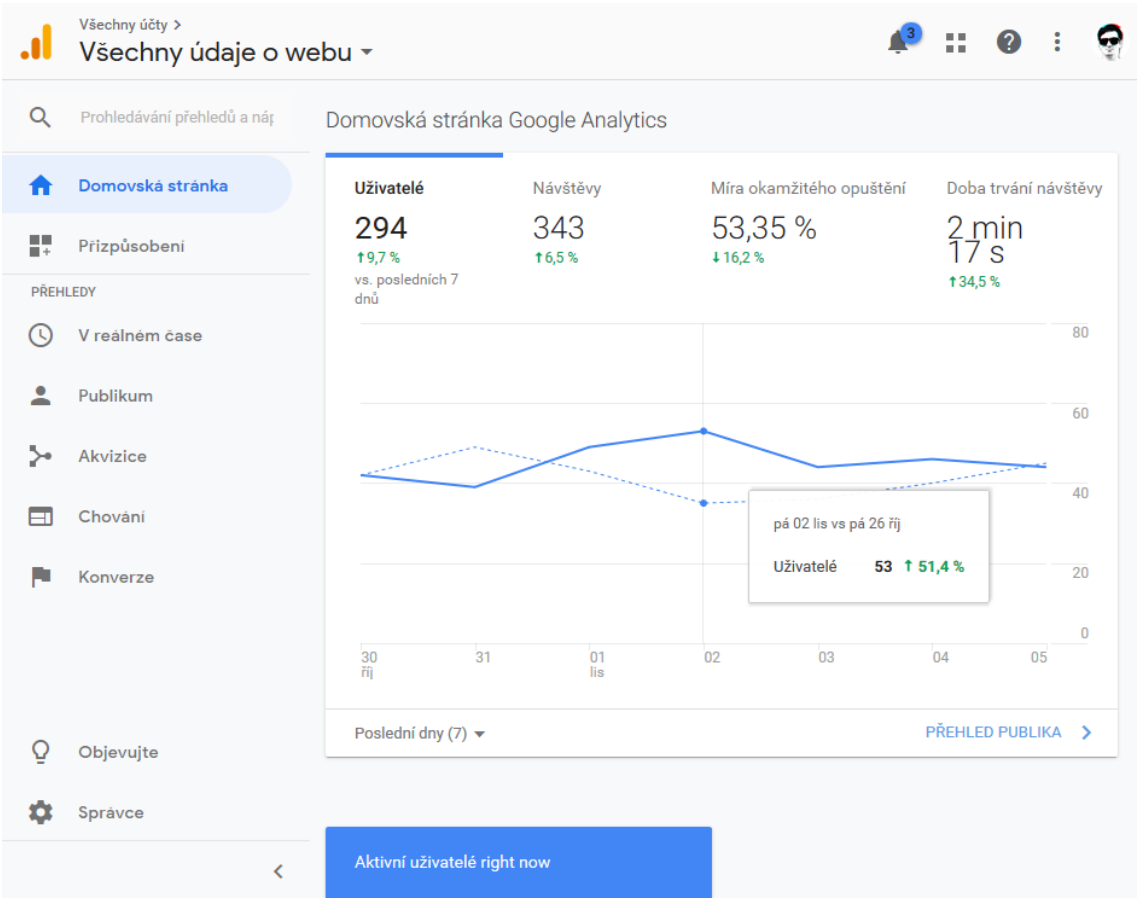
Pasivní zpětná vazba funguje na základě automatického zaznamenávání chování uživatele v aplikaci. Lze tedy zaznamenávat akce, které uživatel vykonává a jejich výsledky odesílat na server, kde probíhá jejich vyhodnocení. Jako nejvhodnější server s touto funkcionalitou bylo vybráno řešení od společnosti Google, Inc. s názvem Google Analytics. Výhodou tohoto řešení je připravenost na formát dat, který bude aplikace produkovat. Další výhodou vybraného řešení je podpora operačního systému Android.

Popis fungování systému Google Analytics

Služba Google Analytics byla primárně vytvořena pro sbírání dat o chování uživatelů na webových stránkách. Protože jsou obě platformy (Android i Google Analytics) produktem společnosti Google, Inc, byla do služby Google Analytics integrována i podpora pro sběr dat z aplikací běžících v operačním systému Android.[17]

Objem dat, který je do služby Google Analytics zasílán závisí na míře integrace této služby do aplikace nebo webové stránky. Princip však spočívá v odesílání událostí, které se udály. Událost může mít mnoho charakterů. Jedná se například o přepnutí do jiného okna, klepnutí na tlačítko, ale i různé chybové stavy.[17][18]

Provázání stránky nebo aplikace se službou Google Analytics probíhá pomocí vygenerované trackovací *Identifikace – Identification* (ID). Pomocí *Aplikačního programového rozhraní – Application Programming Interface* (API) jsou data odesílána na server, kde jsou následně automaticky zpracována do grafů a tabulek [17][18]. Ukázkové zpracování dat ve službě Google Analytics lze vidět na obr. 1.5.



Obr. 1.5: Ukázka hlavní stránky Google Analytics se zpracovanými daty.

1.3.2 VAST

Specifikace VAST předepisuje šablony pro reklamní systémy poskytující různé druhy reklamního formátu. Jedná se zejména o video obsah, je však možné poskytovat i formu statického obrazu. VAST je (*Rozšiřitelný značkovací jazyk – Extensible Markup Language* (XML)) šablonou, která stanovuje formát zobrazení reklamy na klientském zařízení. Celý tento systém usnadňuje zobrazování a zpracování reklam. V rámci multimediálního přehrávače je možné určit pozici reklamy v přehrávaném obsahu, její délku, odkázat zákazníka na promovaný obsah, atd ... VAST lze momentálně využít ve třech vydáních, která jsou rozepsána dále. VAST je prvním systémem, který sjednotil systém poskytování škálovatelného reklamního obsahu. Jeho použití umožňuje zobrazovaní reklamního obsahu od vícero dodavatelů. Do přehrávače je tedy nutné pokaždé implementovat proprietární reklamní formáty. První verze reklamního formátu VAST byla poprvé představena orgranizací *Interactive Advertising Bureau* (IAB) v roce 2008.

VAST 3.0

Momentálně nejstarší používané vydání specifikace VAST. Ve verzi 3.0 byla poprvé deklarována možnost volby podpory různých formátů reklamy. Jedná se zejména o:

- lineární reklama,
- nelineární reklama,
- lineární reklama s možností přeskočení,
- lineární reklama s doprovodnou reklamou,
- sekvenční skupina reklam.

Je tedy možné na úrovni přehrávače zvolit skupinu podporovaných reklam.[19]

VAST 4.0

Ve verzi 4.0 došlo k mnoha vylepšením jak na straně klienta, tak i na serverové straně. Níže budou popsány všechny podstatné změny.

Ve verzi 4.0 došlo k oddělení video a interaktivního souboru. Tento krok byl zvolen na základě chování přehrávače, kdy multimediální soubor lze bez větších problémů přehrát, avšak interaktivní soubory nelze vždy korektně zobrazit. Z tohoto důvodu přibyla v souboru VAST značka, která definuje oddělení těchto dvou souborů. Také byla přidána možnost přehrávat videa s vysokým rozlišením. Jako další je možnost využít informaci o dalších videích určených ke zpracování, což přináší divákovi větší komfort při sledování – videa se načítají při přehrávání předešlého.[20]

Jako velkou novinku ve verzi 4.0 lze považovat implementaci univerzálního reklamního ID. Univerzální ID slouží k identifikaci sledujícího uživatele napříč zařízeními a službami. Slouží také pro personalizaci reklamního obsahu.[20]

Došlo i k úpravě ověřování přehrání reklamy a interakce s uživatelem. Byly včleněny kategorie, které ulehčují oddělení reklamního obsahu od konkurenčního. Implementace kategorií přispěla k vyšší bezpečnosti. [20]

VAST 4.1

Oproti předchozím verzím došlo u verze 4.1 k následujícím změnám. Nyní je možné ověřovat i architekturu, která není VPAID a je oddělená od multimediálního souboru. V rámci této změny došlo k zahrnutí podpory pro práci s „Open Measurement“. Šablona pro doručování audio reklam byla sloučena do VAST. Byla přidána podpora „skrytých titulků“ pro neslyšící a dále byla zjednodušena práce s AdServingID. V neposlední řadě byly upraveny vlastnosti serverové části.[21]

1.3.3 VPAID

Služba *Definice rozhraní zobrazování reklam pro přehrávač videa – Video Player Ad-Serving Interface Definition* (VPAID) vytváří společné rozhraní mezi přehrávačem videa a reklamní jednotkou. To umožňuje interakci s reklamami.[22]

Pomocí VPAID je možné zachytávat podrobnosti o přehrávání reklam a interakce uživatele. Jedná se o standardizovaný formát monitorování výše uvedených akcí v přehrávači. Jedná se o rozšíření služby VAST. Samotná služba VAST disponuje pouze jednoduchou funkcionalitou, která v době vzniku neodpovídala požadavkům inzerentů. Například vůbec neobsahuje funkce pro sledování interakce uživatele s reklamou.[22]

Společné rozhraní umožňuje přítomnost protokolu pro komunikaci mezi přehrávačem videa a reklamním serverem. Je tedy možné získat více informací o dosahu reklamy, jejím stavu a informace o uživatelské aktivitě. Přítomnost VPAID umožňuje v přehrávači zobrazovat interaktivní reklamní obsah, který samotná služba VAST není schopná zpracovat.[22]

1.3.4 VMAP

Specifikace *Playlist s video reklamou – Video Multiple Ad Playlist* (VMAP) je XML šablona, která předepisuje strukturu pro vlkádání reklam do reklamního inventáře. Využívá se při situaci, kdy majitel video obsahu nemůže kontrolovat nebo spravovat samotný video přehrávač.[23]

VMAP dokáže ovládat reklamní inventář a tím definovat pozice reklam v přehrávaném video obsahu. VMAP však neumožňuje zasahovat do reklam samotných. Reklamy definuje VAST a VMAP dokáže pouze určovat jejich pozici a počet v přehrávaném multimediálním obsahu. VMAP tedy pouze definuje reklamní pozici, počet reklam v jedné reklamní pozici a jejich celkový počet v přehrávaném videu. Tímto se také zajišťuje integrita celého reklamního inventáře.[23]

2 Analýza dostupných řešení

V následující části bude vypracován návrh multimediálního přehrávače pro systém Android. V úvodu bude provedena analýza dostupných řešení. Dále bude vybráno vhodné řešení, které bude nejvíce odpovídat požadovaným vlastnostem. Z vybraného řešení bude vytvořen návrh přehrávače. Jako část návrhu bude využit *Unifikovaný modelovací jazyk – Unified Modeling Language* (UML).

2.1 Dostupná řešení

2.1.1 Multimediální framework založený na technologii Gstreamer

Veřejně dostupných řešení frameworku multimediálního přehrávače se na Internetu nachází celá řada. Z nalezených lze uvést například „An Android Multimedia Framework Based on Gstreamer“. Tento přehrávač disponuje podporou formátů MPEG-2, MPEG-4, *Windows Média Video – Windows Media Video* (WMV), *Média (*MPEG-4) Video – Media (MPEG-4) Video (M4V), *Formát QuickTime – QuickTime File Format* (QT) a formát *Formát RealMedia – RealMedia File Format* (RM)[24]. Multimediální framework postavený na technologii Gstreamer však nedisponuje potřebnou podporou pro streamované vysílání typu MPEG–DASH a HLS[24]. Není tedy možné jej využít.

2.1.2 Android Media

Další nalezený framework je součástí systému Android. Jedná se o nativní multimediální přehrávač. V programovém prostředí Android se jedná o třídu Media. Tato třída disponuje základní funkcionalitou pro přehrávání médií. Tato funkcionalita zahrnuje samotné jádro přehrávače, které se nazývá MediaPlayerService. Dále je součástí této funkcionality také třída pro práci s kodeky médií a vlastním grafickým rozhraním přehrávače.[25] Stejně jako výše uvedený framework založený na technologii Gstreamer, ani tento framework nedisponuje funkcionalitou pro přehrávání streamovaného obsahu ve formátech MPEG–DASH a HLS[26].

2.1.3 ExoPlayer

Při studování technické dokumentace k operačnímu systému Android bylo nalezeno třetí řešení. Poslední nalezené řešení nese název ExoPlayer. Tento framework je vyvíjen společností Google, Inc. pro jejich vlastní aplikace Youtube a Google Movies. Výhodou tohoto frameworku je doporučení samotnou společností Google, Inc. pro

vývoj aplikací založených na streamovaném obsahu[26]. Další nespornou výhodou tohoto frameworku je velká komunitní podpora. Poslední výhoda spočívá v aktivním vývoji. O vývoj se stará společnost Google, Inc. a kód, který je dostupný ve službě GitHub je udržován a jeho části jsou několikrát do měsíce aktualizovány. Na základě těchto vlastností byl framework ExoPlayer zvolen pro následný vývoj přehrávače.

2.2 Popis frameworku ExoPlayer

Jedná se o knihovnu, či framework pro přehrávání multimediálního obsahu. Celý framework je napsaný v jazyce Java a je dostupný jako open source, pod licencí Apache 2.0 [27]. Z výše zmíněných řešení pro přehrávání multimediálního obsahu je framework ExoPlayer postaven na základu multimediálního přehrávače v operačním systému Android. Využívá tedy funkcionality přímo tohoto přehrávače. Základní funkcionalitu systémového přehrávače rozšiřuje o další funkce, jedná se zejména o adaptivní stream, práce s titulky, rozšířená podpora multimediálních kontejnerů a práce s frameworkem FFmpeg. Všechny podporované multimediální kontejnery, formáty titulků a FFmpeg rozšíření jsou uvedeny v tab. 2.1. Podpora formátů adaptivního streamu a DRM zabezpečení multimediálního obsahu jsou uvedeny v tab. 2.2.

Multimediální kontejnery	**Titulky**	**FFmpeg rozšíření**
FMP4	TTML	Vorbis
WebM	WebVTT	Opus
MPEG-TS	SMPTE-TT	FLAC
MKV	SubRip	ALAC
MP3	SubStationAlpha (SSA)	PCM μ-law
MP4	ASS	PCM A-law
M4A		AMR-NB
OGG		AMR-WB
MPEG-PS		AAC
WAV		AC-3
FLV		E-AC-3
ADTS		DTS, DTS-HD
		TrueHD

Tab. 2.1: Seznam podporovaných multimediálních kontejnerů, formátů titulků a FFmpeg rozšíření.

DRM	Adaptivní stream
WideVine	MPEG–DASH
PlayReady	HLS
	SmoothStreaming

Tab. 2.2: Seznam podporovaných DRM zabezpečení a adaptivních streamů.

Jak je z tabulek 2.1 a 2.2 patrné, framework Exoplayer disponuje širokou škálou podporovaných formátů. V úvodu této práce byly nastíněny požadavky pro přehrávač, který bude na základech frameworku Exoplayer zhotoven.

2.2.1 Podpora formátů adaptivního streamování

Pro adaptivní streaming je ve frameworku implementována podpora multimediálních kontejnerů, které jsou uvedeny tab. 2.3.

MPEG–DASH	SmoothStreaming	HLS
FMP4	FMP4	MPEG-TS
WebM		FMP4
Matroska		ADTS (AAC)
		MP3

Tab. 2.3: Seznam kontejnerů podporovaných v adaptivním streamování.[28]

MPEG–DASH v současné době nepodporuje multimediální kontejner MPEG-TS. Jeho podpora do budoucna není plánována.[28] U technologie MPEG–DASH jsou všechny multimediální kontejnery podporovány pouze v demuxovaném režimu. Demuxovaný režim je také podmínkou multimediálního kontejneru FMP4 u technologie SmoothStreaming.[28] Technologie HLS nevyžaduje u kontejnerů vypsaných v Tab. 2.3 žádné speciální vlastnosti.

2.2.2 Podpora systému ochrany multimediálního obsahu

Framework ExoPlayer podporuje dohromady čtyři různé metody šifrování obsahu DRM. Všechny jsou vypsány v tab. 2.4.

Zvolená DRM ochrana je závislá na verzi systému Android. U MPEG–DASH je technologie Widevine dostupná od verze *Aplikační programové rozhraní – Application Programming Interface* (API) 19, pro schéma „cenc“ a od verze 25 pro schémata „cbcs“, „cbc1“ a „cens“[28]. Dále technologie PlayReady SL2000 je u technologie MPEG–DASH dostupná pouze u Android TV. Poslední uvedená technologie pro MPEG–DASH je ClearKey, kterou je možné využít od verze API 21.

MPEG–DASH	SmoothStreaming	HLS
Widevine	PlayReady SL2000	AES-128
PlayReady SL2000		Widevine
ClearKey		

Tab. 2.4: Seznam DRM ochran podporovaných v adaptivním streamování.[28]

U streamovací technologie SmoothStreaming je dostupný systém zabezpečení obsahu pouze typ PlayReady SL2000. Systém je stejně, jako u streamovací technologie MPEG–DASH dostupný pouze u verze Android TV.

Poslední streamovací technologie HLS podporuje dvě metody zabezpečení obsahu. U metody AES-128 není její použití omezeno verzí systému Android. U druhé metody Widevine je omezení podobné jako u streamovací technologie MPEG–DASH. Rozdíl je pouze u schémat „cbc1“ a „cens“, která nejsou podporována. U zbylých dvou schémat zůstávají verze API systému Android stejné, tedy pro „cenc“ schéma je požadováno API 19 a vyšší, dále pro „cbcs“ 25 a vyšší.

2.2.3 Podpora funkcionalit napříč verzemi OS Android

Následuje výčet funkcionalit a minimální přípustné verze operačního systému Android pro plnou funkční podporu. Tabulka č. 2.5 byla převzata a přeložena ze zdroje [29].

2.2.4 Struktura frameworku ExoPlayer

Framework ExoPlayer je složen z částí, které jsou vůči sobě kódově nezávislé. Je tedy možné přizpůsobit ExoPlayer vlastní potřebě a vynechat nepotřebné části. Části, či dále balíčky, jsou dvou typů. První typem je knihovna (library). Knihovna poskytuje základní funkcionalitu. Jedná se například o práci s multimediálním streamem HLS, MPEG–DASH nebo SmoothStream. Knihovnou je dokonce i jádro samotného frameworku ExoPlayer. Tuto knihovnu je nutné použít, ostatní knihovny je možné využívat libovolně. Druhým typem balíčku je rozšíření (extension). Rozšíření přináší doplňkovou funkcionalitu, která není k základní práci přehrávače potřeba. Jako rozšíření je například podpora reklamního obsahu. Dále jsou k dispozici rozšíření pro práci s většinou uvedených multimediálních kontejnerů.

Framework využívá k implementaci knihoven a k rozšíření automatizační nástroj Gradle, který slouží k automatickému sestavování programu. Tento nástroj má vlastní konfigurační soubor, kde jsou definovány závislosti, které se během kompilace programu automaticky zavádějí do programového kódu. Při vývoji se tedy používá

Funkce	**Verze Android**	**Android API**
Přehrávání audia	4.1	16
Přehrávání videa	4.1	16
MPEG–DASH (bez DRM	4.1	16
MPEG–DASH (Widevine schéma „cenc“)	4.4	19
MPEG–DASH (Widevine „cbcs“, „cbc1“, „cens“)	7.1	25
MPEG–DASH (ClearKey)	5.0	21
SmoothStreaming (Bez DRM)	4.1	16
SmoothStreaming (PlayReady SL2000)	Android TV	Android TV
HLS (Bez DRM)	4.1	16
HLS (AES-128)	4.1	16
HLS (Widevine schéma „cenc“)	4.4	19
HLS (Widevine schéma „cbcs“)	7.1	25

Tab. 2.5: Podpora funkcí v závislosti na verzi API a OS Android[29]

funkcionalita knihoven a není nutné se starat o provázání kódu. Výhodou implementace pomocí nástroje Gradle je možnost importování vzdálených balíčků, které se fyzicky nenacházejí přímo v projektu, ale například v repositáři služby GitHub. Další výhodou je logicky oddělený kód, který lze udržovat bez nutnosti zásahu do projektu. Ve výpisu č. 2.1 je ukázána část kódu pro zavedení frameworku ExoPlayer do projektu. V prvním řádku ukázky je definován základní balíček pro vytváření aplikací pro systém Android. Na řádcích 6 a 7 je zajištěna kontrola verze prostředí Java, v tomto případě se jedná o verzi 1.8. Dalším důležitým parametrem je určení cílové verze API. Tento parametr je nastaven na řádku číslo 10 a také 14. Nejdůležitější část ukázky začíná na řádku číslo 25. Zde se nachází definice závislostí. Závislosti jsou ony balíčky, které byly popsány v předešlé části této kapitoly. V ukázce je uvedena pro názornost pouze jedna závislost. Jedná se o zavedení jádra ExoPlayeru do vlastního projektu.

Výpis 2.1: Ukázka sestavovacího souboru pro nástroj Gradle

```
apply plugin: 'com.android.application'

android {
    android {
        compileOptions {
            sourceCompatibility JavaVersion.VERSION_1_8
            targetCompatibility JavaVersion.VERSION_1_8
        }
    }
    compileSdkVersion 28
    defaultConfig {
        applicationId "cz.janbenedikt.advancedplayer"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
        }
    }
}

dependencies {
  implementation fileTree(dir: 'libs', include: ['*.jar'])
  implementation project(':exoplayer-library-core')
  ...
}
```

2.2.5 Struktura balíčku v projektu ExoPlayer

Balíček v projektu ExoPlayer je samostatným projektem. Splňuje tedy všechny základní zásady projektu, které jsou v projektech pro operační systém Android nezbytné. Projekt je tvořen základní adresářovou strukturou, která obsahuje adresář „manifest“ a „java“. Jako volitelné adresáře (závisí na povaze projektu) lze v projektu nalézt také „resources“ a „assets“. Složka „manifest“ obsahuje tzv. manifest soubor. Obecně jsou v souboru manifest uvedeny všechny potřebné informace o aplikaci pro operační systém Android. Jedná se například o seznam oprávnění

k externím funkcím (internet, volání, přístup k úložišti, apod ...), verze aplikace, minimální verze API a hlavní třída aplikace [30]. Následuje výpis obrazovek aplikace (layouts). Uvedení souboru manifest je povinné [30]. Ve složce „java“ se nachází zdrojové soubory samotné aplikace, respektive balíčku. Umístění zdrojových souborů je stejné, jako v klasickém projektu. Hlavní složka definuje balíček (balíčkem se v tomto případě myslí tzv. „package“ projektu psaného v jazyce Java). V hlavní složce jsou následně umístěny samostatné zdrojové soubory tříd a funkcí. Zbylé dvě složky „resources“ a „assets“ nejsou v projektové struktuře povinné. První nepovinná složka „resources“ obsahuje externí soubory jako jsou například obrázky, ikony apod. Poslední nepovinná složka „assets“ je určena pro externí soubory, které jsou aplikací využívány pro její práci. V případě balíčku pro ExoPlayer se může jednat například o playlist, testovací videa a hudební soubory [30]. Poslední a hlavní součást každého balíčku je konfigurační skript pro sestavení aplikace. Balíčky využívají stejně, jako celý framework, balíčkovací systém Gradle.

2.2.6 Popis programového kódu

Komponenta přehrávače ve frameworku ExoPlayer je navržena tak, aby nezáleželo na formátu přehrávaného média, kde je médium uloženo a jak bude vykresleno. Všechny konkrétní nastavení jsou delegovány skrze komponenty přehrávače. Delegace komponent probíhá při vytváření přehrávače, nebo při přípravě přehrávání. Dále budou popsány společné části všech verzí třídy ExoPlayer.[31]

1. **MediaSource** – definuje médium, které má být přehráno. Toto médium je do MediaSource načteno a je předáno přehrávači, který z tohoto zdroje médium na začátku přehrávání převezme. Moduly knihovny poskytují podporu pro streamy typu HLS (HlsMediaSource), MPEG–DASH (DashMediaSource), SmoothStream (SsMediaSource) a pro běžné zdroje médií, jako je například MP4 (ExtractorMediaSource). Další implementace MediaSource spočívá v podpoře doplňkových médií typu titulky (SingleSampleMediaSource). Jako poslední jsou různé kombinace popsaných MediaSource, kdy je možné mediální zdroje slučovat do následujících zdrojů: MergingMediaSource, ConcatenatingMediaSource, LoopingMediaSource a ClippingMediaSource.[31]
2. **Renderers** – starají se o zpracování předaného média (jeden z typů MediaSource) a vytvářejí z něj jednotlivé komponenty. Knihovna obsahuje následující implementace pro běžné typy médií: MediaCodecVideoRenderer, MediaCodecAudioRenderer, TextRenderer a MetadataRenderer. Po předání MediaSource jej zpracuje a předá přehrávači. Tento úkon se děje při vytváření přehrávače.[31]
3. **TrackSelector** – má na starost zvolení jedné ze stop dostupných v médiu (na-

příklad zvuková stopa dabingu). TrackSelector zvolenou stopu předá každému z dostupných Renderů. Popsaný úkon vykonává objekt DefaultTrackSelector. [31]

4. **LoadControl** – řídí načítání více médií (pokud některý z výše popsaných druhů MediaSource obsahuje více video, či audio stop) a určuje kolik jich bude uloženo do mezipaměti. Hlavní LoadControl objekt je dostupný jako DefaultLoadControl a je vhodný pro většinu případů. LoadControl je využit po vytvoření přehrávače.[31]

Hlavní prvky při stavbě přehrávače ExoPlayer

Následovat bude popis prvků použitých při vytváření instance přehrávače. Tyto prvky jsou při jejím vytváření požadovány, a proto bude níže nastíněn jejich účel.

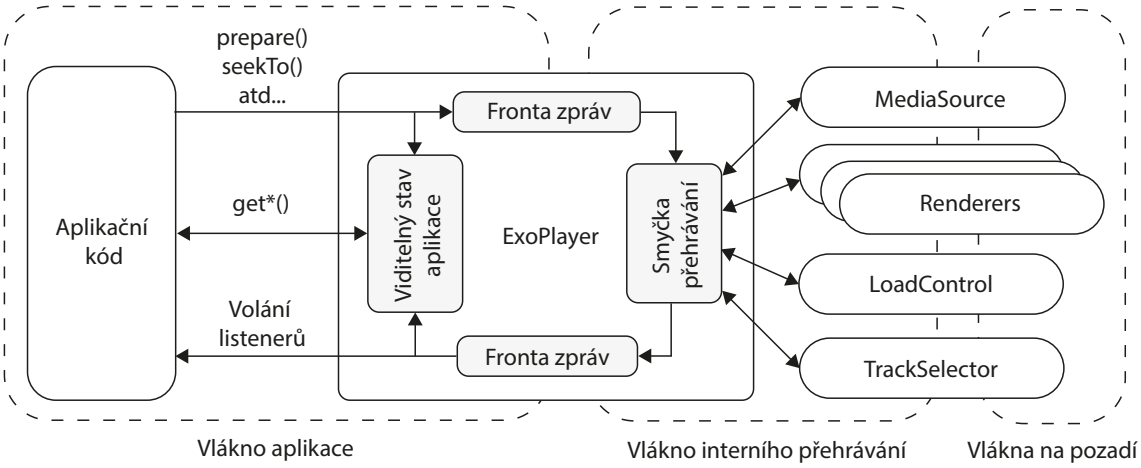
1. **Context** – Rozhraní globálních informací o aplikaci. Je to abstraktní třída a o její implementaci se stará systém Android. Z kontextu je možné získat různé zdroje, které jsou specifické pro danou aplikaci (například přístup do složky „assets“ a „resources“). Dále je kontext aplikace využíván pro komunikaci v rámci různých obrazovek aplikace.
2. **RenderersFactory** – Rozhraní pro budování přehrávače. Obsahuje listeners pro události video a audio renderů.
3. **BandwidthMeter** – Poskytuje informace o šířce pásma linky. Tímto je přehrávač informován o prostředcích, na jejich základě může upravovat bitrate přehrávaného videa (pouze pokud je rozdílný bitrate k dispozici).
4. **Looper** – Nahrazuje klasickou třídu v jazyce Java – Threads. Třída Looper je doplněna o funkcionalitu komunikace mezi vlákny přizpůsobenou pro operační systém Android.

Hlavní třídou v přehrávači ExoPlayer je SimpleExoPlayer. Vychází z rozhraní ExoPlayer, které poskytuje základní funkcionalitu. Jakožto rozhraní jej není možné inicializovat jako objekt a je tedy nutné metody přetížit a využít vlastního objektu. Třída SimpleExoPlayer je inicializována pomocí konstruktoru, kterému jsou předávány následující objekty Context, RenderersFactory, BandwidthMeter a Looper. Po předání těchto parametrů dojde k inicializaci přehrávače. Po inicializaci přehrávače je možné vytvořit MediaSource a ten nastavit v přehrávači jako zdroj média.

Aby byl výstup z přehrávače viditelný, je nutné jej předat do zobrazovací třídy s názvem SimpleExoPlayerView. Po inicializaci objektu z této třídy dojde pomocí konstruktoru navázání objektu na layout s přehrávačem. Pomocí metody *setPlayer()* dojde k provázání s logickou částí přehrávače a tím je vše připraveno k přehrání multimediálního obsahu.

Model přehrávače z pohledu vláken systému

Model přehrávače ExoPlayer z pohledu vláken v systému lze vidět na obr. 2.1. Z obr. 2.1 lze vyčíst, že přehrávač nepracuje pouze v jednom vlákně. To ani systém Android neumožňuje [30]. Přehrávač je rozdělen na tři druhy vláken. Každé vlákno má zde svou funkci. První sada vláken je zodpovědná za hlavní část aplikace a její vlastní kód. Dále jsou zde vlákna zodpovědná za interní přehrávání. Vlákna interního přehrávání mají na starost komunikaci ExoPlayeru se zdroji média a jejich kontrolu. Třetím typem vláken jsou vlákna na pozadí, která mají na starost načítání média a dále i jeho další zpracování.[31]



Obr. 2.1: Ukázka fungování ExoPlayer z pohledu vláken.[31]

3 Návrh pokročilého přehrávače

3.1 Stanovení požadavků na přehrávač

V úvodu práce byly popsány základní protokoly a funkcionality. Jedná se především o podporu streamovaného obsahu a možnost jeho zabezpečení pomocí DRM. Dále by měl přehrávač umět pracovat s reklamními systémy typu VAST, VPAID a VMAP. Pokud nebude reklamní systém dostupný a vznikne požadavek na přítomnost reklamy, přehrávač by měl být schopen vytvořit reklamu z pouhého odkazu na reklamní video. Výstupem diplomové práce by měla být knihovna přehrávače, která bude poskytovat veřejné metody pro práci s rozhraním přehrávače. Knihovna by měla být koncipována tak, aby bylo možné jednoduše napojit vlastní grafické rozhraní a plně využít funkcionalitu přehrávače.

3.2 Základní funkcionalita

Navrhovaný přehrávač by měl splňovat funkcionalitu definovanou v první kapitole. Jedná se zejména o podporu adaptivního streamu. V rámci adaptivního streamu by měl přehrávač podporovat formáty HLS a MPEG–DASH. V rámci této podpory by měl přehrávač využít potenciál obou protokolů. Potenciálem se myslí přizpůsobení datového toku videa dostupné kapacitě přenosového kanálu, podpora více zvukových a obrazových stop a možnost rychlého posunu na časové ose.

Další důležitou součástí funkčního vybavení přehrávače je podpora zabezpečení obsahu DRM. Při využití frameworku ExoPlayer je zaručená podpora protokolu Widevine. U technologie PlayReady SL2000 je podpora zaručena pouze u verze Android pro televize. FairPlay není podporováno vůbec.

Přehrávač by měl disponovat možností zaznamenávat akce uživatele a ty dále zasílat službě Google Analytics.

Poslední žádaná podpora se týká implementace poskytovatele reklamních formátů VAST, VPAID a VMAP.

Na obr. A.1 je zobrazen návrh struktury navrhovaného přehrávače. V případě navrhované knihovny je hlavní třídou *PlayerActivity*. Třída *PlayerActivity* bude obsahovat všechny potřebné metody pro zprovoznění přehrávače. Jako vstupní soubor pro konihovnu bude využit „superplaylist“ ve formátu *JavaScriptový objektový zápis – JavaScript Object Notation* (JSON). Hlavní soubor s informacemi o médiích („superplaylist“, příloha C) bude obsahovat seznam videí, který bude syntakticky analyzován a z výsledku bude vytvořeno jednoduché menu ve formě seznamu. O syntaktickou analýzu souboru se bude starat třída *PlayListDownloader*. Zavoláním metody *getPlaylist()* je „superplaylist“ stažen z webu a předán jako parametr

metodě *parseJSON(json : String)*. Adresy multimediálního obsahu jsou uloženy do objektu *Profile*, obsaženém v objektu *Playlist*. Objekt obsahuje potřebné proměnné a je strukturován tak, aby odpovídal požadavkům nastavení přehrávače ExoPlayer. Celá inicializace a nastavení přehrávače ExoPlayer bude probíhat v metodě *initializePlayer()*. Inicializace bude automaticky volána při spuštění aplikace metodou *onCreate()*. V inicializační metodě dojde z playlistu k vytvoření *mediaSource*. Při inicializaci přehrávače bude předán výsledný *mediaSource*. V posledním kroku inicializace bude celý vytvořený přehrávač předán do *ExoPlayerView*.

3.3 Ukázka vlastního modulu přehrávače ExoPlayer

Jako testovací modul bylo zvoleno vytvoření doplňkové funkcionality pro komunikaci se službou Google Analytics. Modul je koncipován jako jednoduchý prostředník mezi aplikací a službou Google analytics. Struktura modulu dodržuje zásady stanovené v předešlé kapitole. Celý balíček zaštiťuje konfigurační soubor *build*, napsaný v jazyce Gradle. V souboru *build* je navíc oproti ukázce výpisu č. 2.1 definován název balíčku a jeho popis. Nezbytností celého balíčku je napojení na knihovnu Google Analytics. Napojení je také definováno v souboru *build*.

Balíček je složen dohromady ze čtyř složek. Jedná se o:

1. manifest,
2. java,
3. generatedJava,
4. resources.

Ve složce číslo jedna je uložen manifest. Stejně jako u projektu aplikace pro systém Android, i v projektu balíčku pro framework ExoPlayer je přítomen manifest. V případě manifestu pro balíček jsou v tomto souboru přítomny pouze oprávnění balíčku v rámci přiřazování prostředků systémem. Definice oprávnění jsou uvedena ve výpisu č. 3.1.

Výpis 3.1: Ukázka souboru manifest pro balíček Google Analytics

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
 xmlns:android=
  "http://schemas.android.com/apk/res/android"
 xmlns:tools=
  "http://schemas.android.com/tools"
 package=
  "cz.janbenedikt.android.exoplayer2.ext.googleanalytics">

 <uses-permission
  android:name="android.permission.INTERNET"/>
 <uses-permission
  android:name="android.permission.ACCESS_NETWORK_STATE"/>
</manifest>
```

V druhé složce s názvem *java* se nachází zdrojové soubory balíčku. Jsou zde uloženy dvě složky. První složka obsahuje zmiňované zdrojové soubory a druhá složka obsahuje zdrojové soubory pro testovaní. V třetí složce s názvem *generatedJava* jsou umístěny soubory, které se dynamicky generují vývojovým prostředím, což je v tomto případě Android Studio. V těchto složkách jsou uloženy soubory poskytující například zdrojové texty pro aplikaci. Poslední složka obsahuje veškerý dodatečný obsah pro balíček. Rozšiřující balíček obsahuje pouze soubor definující uživatelské ID služby Google Analytics a nastavení balíčku. Jeho ukázka je zobrazena ve výpisu č. 3.2.

Výpis 3.2: Ukázka souboru manifest pro balíček Google Analytics

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="ga_sessionTimeout">300</integer>

    <bool name="ga_autoActivityTracking">true</bool>

    <string name="ga_trackingId">UA-XXXXXXXXX-X</string>

    <string name="ga_sampleFrequency">100.0</string>

    <bool name="ga_reportUncaughtExceptions">true</bool>
</resources>
```

V souboru nastavení balíčku jsou uvedeny následující hodnoty:

1. **ga_autoActivityTracking** – slouží pro nastavení doby definující délku ne-

činnosti, po které se vlákno balíčku uspí,

2. **ga_autoActivityTracking** – povolení automatického sledování (aktivní obrazovka, apod..),
3. **ga_trackingId** – ID uživatele vygenerované službou Google Analytics,
4. **ga_sampleFrequency** – počet zasílaných vzorků,
5. **ga_reportUncaughtExceptions** – automatické zaznamenání a odeslání neočekávané výjimky.

Samotný zdrojový kód je tvořen jednou třídou s názvem *AnalyticsApplication*. Její inicializace se provádí metodou *initialize(Context context)*, do které je nutné předat kontext aplikace. Po úspěšné inicializaci probíhá práce s balíčkem pomocí tří metod.

První metoda *trackEvent(String category, String resume, String label)* slouží k vlastní události definované v parametrech metody. První parametr zařadí událost do příslušné kategorie. Následuje parametr, který definuje název zaslané akce. Poslední parametr slouží k podrobnějšímu popisu zaznamenané události.

Druhá metoda *trackException(Exception e)* dokáže zaznamenat výjimku typu Exception.

Z důvodu, že okna jsou v aplikacích pro Android značena strojově, slouží poslední metoda *trackScreenView(String screenname)* ke srozumitelnějšímu pojmenování.

4 Aspekty vývoje aplikace pro operační systém Android

4.1 Publikace aplikace v obchodě Google Play Store

Publikace aplikace v obchodě Play Store se řídí pravidly, která je nutné dodržet. Při nedodržení těchto pravidel, není možné aplikaci do obchodu umístit, případně je stažena přímo společností Google, Inc. Společnost Google, Inc. poskytuje návod pro kontrolu všech pravidel v přehledném seznamu. Samotné zveřejnění aplikace je rozděleno do dvou fází. První fází je příprava aplikace ke zveřejnění. Přípravou se rozumí následující úkony:

1. Úprava kódu, kde je nutné odstranit zápis do systémového logu,
2. přepnutí kompilace z režimu *debug* do režimu *release*,
3. kompilace a podepsání aplikace (například pomocí nástroje Gradle),
4. otestování funkčnosti aplikace,
5. aktualizace všech zdrojů aplikace (obrázky, ikony, atd. . .),
6. příprava serverů, na kterých je aplikace závislá.

Jakmile je výsledná aplikace připravená, po zkompilování je dostupný soubor ve formátu *.apk*. Soubor *.apk* je balíčkem aplikace, který je možné nahrát do obchodu Play Store. V tomto kroku se předpokládá, že vývojář má v Google Konzoli aktivní vývojářské členství. Členství je aktivní po vyplnění údajů o vývojáři (fyzická nebo právnická osoba) a zaplacení poplatku 25 USD.

Jsou-li splněny všechny výše uvedené podmínky je možné přejít k druhému kroku - zveřejnění aplikace v obchodě Play Store. Při publikování nové aplikace v obchodě Play Store je možných několik stavů, kterými může být aplikace označena [32]:

1. **Koncept** - Aplikace zatím není v obchodu publikována.
2. **Připraveno k publikování** - Aplikace zatím není publikována v Google Play, je však připravena pro publikování. Jakmile je vývojář připraven k publikování, je možné aplikaci odeslat ke zpracování.
3. **Čeká na publikování** - Aplikace je ve stavu zpracování společností Google, Inc.
4. **Publikováno** - Aplikace je publikována a je dostupná v Google Play. V tomto stavu je možné vytvářet nová vydání aplikace (aktualizovat stávající verzi).
5. **Odmítnuto** - Stav nastane, pokud vývojář porušil některou ze zásad Google Play. Aplikace je odmítnuta a není publikována. Jakmile je aplikace upravena podle zásad Google Play, je možné ji znovu publikovat.
6. **Pozastaveno** - Publikování a distribuce aplikace bylo pozastaveno z důvodu porušení zásad Google Play. Porušení zásad může vzniknout například ne-

vhodným obsahem aplikace nebo porušením distribuční smlouvy. Vývojář je s tímto stavem obeznámen přes e-mail.

Jakmile je aplikace vydána a je přítomna v obchodě Google Play je stále nutné ji udržovat aktuální. Přechází však do nové kategorie stávajících aplikací. Aplikace v kategorii stávajících aplikací může nabývat následujících stavů [32]:

1. **Publikováno** - Aplikace je publikována v obchodě Google Play. Dostupná je nejnovější verze aplikace. V Google Play je zobrazeno datum publikování nejnovější verze.
2. **Aktualizace zamítnuta** - Aktualizace byla zamítnuta z důvodu porušení zásad Google Play. Poslední publikovaná verze aplikace je na Google Play nadále dostupná, avšak poslední aktualizace nikoliv. Jakmile vývojář v aplikaci provede změny a odstraní nedostatky porušující zásady, je možné aplikaci znovu odeslat k posouzení společností Google, Inc.
3. **Publikování zrušeno** - Aplikace nyní není k dispozici pro nové uživatele. V zařízeních stávajících uživatelů však stále dostupná je. Aplikaci je možné znovu publikovat.
4. **Pozastaveno** - Publikování a distribuce aplikace bylo pozastaveno z důvodu porušení zásad Google Play. Porušení zásad může vzniknout například nevhodným obsahem aplikace nebo porušením distribuční smlouvy. Vývojář je s tímto stavem obeznámen přes e-mail.
5. **Odstraněno** - Ve stavu „Odstraněno“ není aplikace dohledatelná v obchodě Google Play, jak pro nové, tak i pro stávající uživatele. Odstranění může dojít jak z vůle vývojáře, tak i kvůli porušení zásad obchodu Google Play. Dojde-li k odstranění z důvodu porušení je možné chyby napravit a odeslat novou verzi aplikace k posouzení.
6. **Aktualizace čeká na zpracování** - Stav nastane při odeslání aktualizace stávající aplikace. Jakmile započne distribuce aktualizace uživatelům, přejde do stavu „Publikováno“.

4.1.1 Zásady Google Play

Protože obchod Google Play využívá více než jedna miliarda uživatelů, je nutné dodržovat podmínky stanovené v zásadách Google Play. Porušení těchto zásad má za následek stažení aplikace z obchodu. Zásady jsou rozděleny do několika kategorií [33]:

1. **Zakázaný obsah** - Tato kategorie je zaměřena na veškerou propagaci nevhodných a nelegálních věcí. Patří mezi ně například: Obsah ohrožující mravní vývoj dětí a mládeže (Sexuálně explicitní obsah, projevy nenávisti, násilí, šikana a obtěžování), aplikace využívající výkon zařízení pro těžbu kryptoměn,

obchodování s binárními opcemi, hazardní hry a nezákonné činnosti (prodej drog a jiných nelegálních návykových látek, propagace konzumace alkoholu a tabáku nezletilými).

2. **Předstírání jiné identity a duševní vlastnictví** - Jedná se o zákaz aplikací, které využívají jiné známé aplikace, za které se vydávají. Většinou jsou takové aplikace spojeny s určitou podvodnou činností (těžba kryptoměn, sledování uživatele, nebo propagace reklam).
3. **Ochrana soukromí, zabezpečení a klamání** - Jedná se zejména o kontrolu s nakládáním s osobními údaji. V rámci Evropské Unie existují směrnice upravující tyto aktivity.
4. **Zpeněžení reklamy** - Aplikace mohou využívat pro svůj rozvoj příjem z reklamního obsahu. Tento obsah dokáže generovat přímo společnost Google, Inc., provozující službu Google AdSense. Dále některé aplikace, či služby využívají příjem pomocí předplatného. Jedná se například o prodej multimediálního obsahu. Pokud je aplikace navržena nesprávně (například předplatné nejde zrušit), je opět stažena z obchodu.
5. **Spam a minimální funkčnost** - Jedná se zejména o aplikace, které samy bez svolení uživatele navštěvují různé webové stránky s cíle zvýšení návštěvnosti. Dále se také jedná o aplikace, které při každé interakci uživatele zobrazí reklamu.

Všechny výše uvedené body jsou společností Google, Inc. sledovány a testovány. Pokud je nalezeno pochybení, je vývojář se situací obeznámen a je nucen vzniklou situaci napravit.

4.1.2 Vygenerování balíčku aplikace v Android Studiu

Jakmile je projekt přichystám ve stavu „release“, což je označení pro stav projektu k vydání, je možné vygenerovat binární soubor aplikace. Ve většině případů má příponu *.apk*. Ve vývojovém prostředí existují nástroje, které celý proces ulehčí. Pro nahrání balíčku aplikace do obchodu Play je nutné mít balíček podepsaný klíčem, aby nedošlo k jeho záměně. Generování klíče probíhá opět ve vývojovém prostředí. Jakmile jsou klíče připraveny, je možné vygenerovat binární soubor aplikace. Nově je možné využít tzv. Android App Budle.[34] Jedná se o jednodušší generování binárních souborů pro různé verze hardwaru. Verze se liší zejména v rozlišení obrazovky, velikosti paměti a verzi operačního systému Android.

Před samotným vygenerováním verze aplikace do binární formy je nutné zkontrolovat její verzi. Verze aplikace je nastavena v konfiguračním souboru Gradle. Nastavení verze aplikace je zobrazeno na výpisu 4.1. Samotné nastavení verze je prováděno na dvou místech. První místo je proměnná *versionCode*. Jedná se celé

číslo a musí být s každou nahranou verzí v Google konzoli unikátní. Druhá proměnná *versionName* je textové označení verze. Je tedy čistě na vývojáři, zvolí-li označení verze pomocí čísla, nebo pomocí dnes oblíbeného jmenného označení.

Výpis 4.1: Ukázka sekce s nastavením projektu

```
1 defaultConfig {
2         minSdkVersion 20
3         targetSdkVersion 28
4         versionCode 3
5         versionName "0.3"
6         testInstrumentationRunner
             "androidx.test.runner.AndroidJUnitRunner"
7     }
```

Po nastavení verze je možné přejít k samotnému generování. Jak již bylo zmíněno, generování binárních souborů je řešení pomocí nástrojů přítomných ve vývojovém prostředí Android Studio. Nástroj nese jméno „Generate Signed Bundle or APK“. Je doporučené vygenerovat novější „Android App Bundle“ [34]. Nástroji je nutné specifikovat umístění vygenerovaných klíčů a zadat heslo k repozitáři s klíči a heslo použitého klíče. Dále je nutné specifikovat modul aplikace, který bude použit. Jakmile jsou věci nastaveny je v dalším kroku na výběr cesta k vygenerovaným binárním souborům. Jako poslední je nutné specifikovat verzi pro kompilaci (debug/release). Po zadání parametrů v druhém kroku je spuštěna kompilace a výskledkem je balíček, který lze nahrát do Google Konzole.

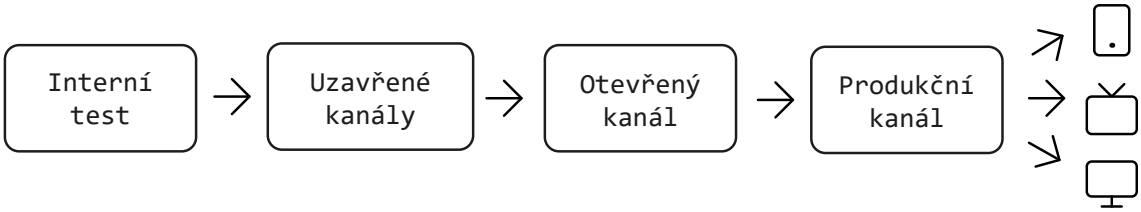
4.1.3 Proces nahrání balíčku aplikace do obchodu Google Play

Veškerá interakce vývojáře s obchodem Google Play je zprostředkována pomocí nástroje Google Console. Nástroj Google Console spravuje veškeré aplikace a jejich verze vývojáře. Je možné vytvářet verze aplikace, testovat aplikaci pomocí interních testerů i veřejného testování. Je-li balíček *.apk* přichystán, je možné vytvořit v konzoli novou aplikaci. Vytváření aplikace v konzoli začíná zadáním názvu aplikace, který bude viditelný v obchodě Play a výchozího jazyka. Po vytvoření aplikace je zobrazena obrazovka s nastavením aplikace. Mezi nezbytné položky k vyplnění patří:

1. **Krátký popis** - Zobrazen v náhledu aplikace. Měl by obsahovat stručné seznámení s aplikací.
2. **Úplný popis** - Celkový popis aplikace a jejich funkcí.
3. **Ikona ve vysokém rozlišení** - Ikona pro identifikaci aplikace v obchodě. Musí být velká alespoň 512 x 512 obrazových bodů. Nově není možné použít ikonu s průhledným pozadím.

4. **Snímky obrazovky** - Snímky aplikace, uvedené v obchodě pro lepší představu uživatele o aplikaci.
5. **Hlavní grafika** - Grafické okno umístěné v záhlaví obchodu.
6. **Typ aplikace** - Definice druhu v obchodě Google Play, kde bude aplikace umístěna (Hry/Aplikace).
7. **Kategorie** - Kategorie aplikace (Finance, Byznys, Jídlo a pití, Hudba a zvuk, Nástroje, apod. . .).
8. **Hodnocení obsahu** - Před zveřejněním aplikace je nutné vyplnit dotazník o hodnocení obsahu. Dotazník obsahuje otázky o formě obsahu v aplikaci. Jedná se o vytvoření doporučeného hodnocení pro věkové skupiny dětí. Obsahuje-li aplikace vulgární výrazy není vhodná pro mladší ročníky, apod. . .
9. **Email** - Email vývojáře pro možnost zasílání zpětné vazby.
10. **Zásady ochrany soukromí** - Vložení vlastních zásad ochrany soukromí.

Po vyplnění potřebných údajů je možné přejít k samotné publikaci. Publikace aplikace je rozdělena do několika vydání. Existuje několik kanálů vydání, které lze v Google konzoli využít. Jak je z Obr. 4.1 patrné, existují čtyři druhy produkčních kanálů.



Obr. 4.1: Produkční kanály vydání

První kanál s názvem „Interní test“ je určen pro interní testování (například v rámci organizace). Zde je aplikace umístěna v prvních verzích, které nejsou dostatečně stabilní pro publikaci mezi veřejností. V rámci kanálu interního testování, není aplikace umístěna do obchodu play a je dostupná pouze z Google konzole.

Následují kanály „Uzavřeného testování“. Tyto kanály odpovídají první alfa verzi aplikace. V těchto kanálech je aplikace již umístěna do obchodu Play, avšak není dohledatelná veřejností. Aplikace je přístupná pouze pomocí vygenerovaného odkazu, skrze kterého je možné se přihlásit do testování. Testování probíhá v rámci schválených uživatelů. Ostatní uživatelé, nejsou oprávněni aplikaci instalovat do svých zařízení.

Třetí typ kanálu nese název „Otevřený kanál“. Kanál otevřeného testování umisťuje aplikaci do veřejného katalogu obchodu Google Play a je možné ji dohledat i jako uživatel, který není oficiálně do testování zapojen. Aplikace je však stále označena jako takzvaná „beta“ verze. To značí, že aplikace není zcela hotová a může

obsahovat chyby, které můžou mít vliv na výkon (jak zařízení, tak samotné aplikace).

Jakmile je aplikace připravena na produkční vydání, je přesunuta do fáze „Produkčního kanálu“. Produkční kanál obsahuje verze, které jsou hotové, otestované a bez chyb. Aplikace je v produkčním kanále dohledatelná v obchodě Play a je možné ji nainstalovat do podporovaného zařízení. O kontrolu kompatibility se stará sám obchod Play, který zamezí instalaci na nepodporovaný hardware.

Všechny kanály mohou obsahovat i podkanály, které mohou při testování sloužit pro rozdělení mezi různé skupiny testujících uživatelů. Rozdělení kanálů slouží i k oddělení verzí aplikace na základě různé geografické polohy. To lze využít například pro verzi aplikace určenou pro Evropu a druhou pro Asii. Dále tímto dělením řešit různé restrikce ve formě zákonného nařízení různých zemí ve světě (Jako příklad lze užít vysokou míru cenzury v Číně).

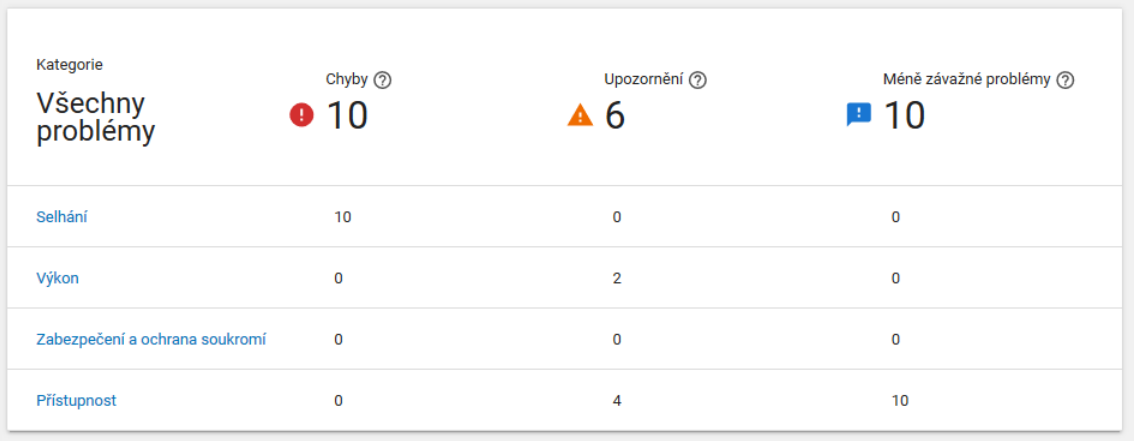
Samotné nahrávání balíčku *.apk* se odehrává v jednom z kanálů, ve kterém bude verze aplikace umístěna.

4.1.4 Testování aplikace

Po každém nahrání nové verze aplikace do Google konzole je aplikace otestována. Jedná se o automatické testy. Testy se provádějí na různých verzích operačního systému Android a různých konfiguracích hardwaru. Testy simulují chování uživatele a sledují různé parametry aplikace. Jedná se zejména o výkon, využití paměti, nebo také rychlost vykreslení obrazovky. Nedojde-li k problému, který by bránil vydání nahrané verze, je aplikace publikována. Tyto testy se snaží odhalit i některé zakázané parametry, jako jsou například vysoká míra spamu a zobrazování reklam, apod... Obecně testy testují výkon aplikace a některé body Zásad Google Play.

Výsledky testů jsou následně vývojáři k dispozici. Pokud aplikace nefunguje správně a například nastávají její pády, je zaznamenán výstup z konzole virtuálního stroje, kde je aplikace testována. Souhrn chyb, varování a problémů lze vidět na Obr. 4.2. Souhrn je reprezentován formou tabulky. Tabulka obsahuje tři úrovně „problému“. První a nejzávažnější úroveň zahrnuje chyby. Jedná se ve většině případů o celkové selhání aplikace s následkem jejího pádu. Následují upozornění, která nezpůsobují u aplikace její pád, avšak mohou mít vliv na její výkon. Poslední úrovní jsou takzvané „Mírně závažné problémy“. Mírně závažnými problémy se myslí vývojářovy „prohřešky“ z hlediska přístupnosti aplikace (optimalizace aplikace pro zrakově indisponované uživatele).

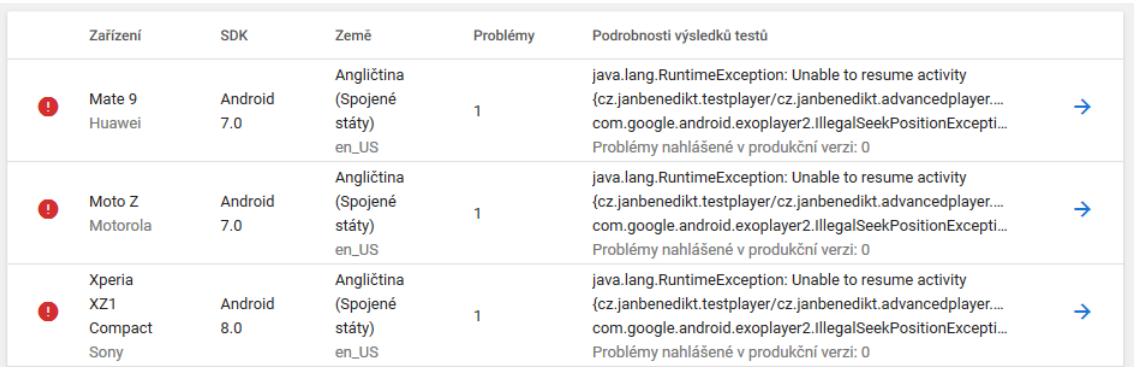
Souhrn se dále dělí na typy problémů. Typy na Obr. 4.2 jsou čtyři. První se týká selhání aplikace, následované jejím pádem. Jde o nejzávažnější problém, který je třeba okamžitě řešit. Následují problémy spjaté s výkonem aplikace. Zde je vhodné



Kategorie	Chyby	Upozornění	Méně závažné problémy
Všechny problémy	10	6	10
Selhání	10	0	0
Výkon	0	2	0
Zabezpečení a ochrana soukromí	0	0	0
Přístupnost	0	4	10

Obr. 4.2: Souhrnné zobrazení chyb a varování zaznamenaných při testech

se vyvarovat například nekonečným smyčkám při čekání na různé druhy událostí. Tyto smyčky můžou mít za následek zacyklení aplikace a její vytěžování systému. Nejde však jen o smyčky, vývojář musí počítat se složitostmi algoritmů, apod. . . Další důležitá kategorie problémů zahrnuje problémy se zabezpečením a ochranou soukromí. V rámci této kategorie je testováno, zdali nejsou osobní data odesílána mimo aplikaci. Poslední kategorií jsou problémy s přístupností. Je doporučené aplikaci přizpůsobit zrakově, či jinak indisponovaným lidem. Jedná se o vysvětlující popisky na ovládacích prvcích apod. Při bližším zkoumání nastalých chyb je k dispozici tabulka



Zařízení	SDK	Země	Problémy	Podrobnosti výsledků testů
Mate 9 Huawei	Android 7.0	Angličtina (Spojené státy) en_US	1	java.lang.RuntimeException: Unable to resume activity {cz.janbenedikt.testplayer/cz.janbenedikt.advancedplayer.... com.google.android.exoplayer2.IllegalSeekPositionExcepti... Problémy nahlášené v produkční verzi: 0
Moto Z Motorola	Android 7.0	Angličtina (Spojené státy) en_US	1	java.lang.RuntimeException: Unable to resume activity {cz.janbenedikt.testplayer/cz.janbenedikt.advancedplayer.... com.google.android.exoplayer2.IllegalSeekPositionExcepti... Problémy nahlášené v produkční verzi: 0
Xperia XZ1 Compact Sony	Android 8.0	Angličtina (Spojené státy) en_US	1	java.lang.RuntimeException: Unable to resume activity {cz.janbenedikt.testplayer/cz.janbenedikt.advancedplayer.... com.google.android.exoplayer2.IllegalSeekPositionExcepti... Problémy nahlášené v produkční verzi: 0

Obr. 4.3: Detailní zobrazení problémů u konkrétního zařízení

s detailním výpisem problémů, které nastaly během testování. Tabulka s těmito výpisy je zobrazena na Obr. 4.3. Tabulka obsahuje informace z virtuálního stroje, kde se aplikace testovala. Je uveden typ zařízení, které bylo simulováno, verze *Systémového vývojového nástroje – Software development kit (*SDK), která odpovídá verzi operačního systému Android, země, počet nastalých problémů a podrobnosti výsledku testů. U vzniklých problémů jsou výsledky reprezentovány výpisem konzole virtuálního stroje Java.

Statistiky dále tvoří grafy výkonu, využití paměti, využití procesoru a další užitečné informace.

5 Vývoj knihovny přehrávače

5.1 Příprava vývoje aplikace

Vývoj začíná vytvořením projektu ve vývojovém prostředí Android Studio. Při vytváření projektu se určují cílové platformy, pro které bude aplikace vyvíjena. Platformou se rozumí tři cílová zařízení, pro která je v současnosti možné aplikace vyvíjet. Hlavním typem zařízení jsou telefony a tablety. Mezi těmito dvěma druhy neexistuje z pohledu vývojového prostředí a výsledného binárního souboru rozdíl. Jedná se pouze o jiné rozmístění ovládacích prvků a přizpůsobení aplikace pro větší obrazovky. Při cílení na tablety je nutné vzít v úvahu i rozlišení obrazovek společně s hustotou obrazových bodů na obrazovce. Je to důležité z hlediska rozlišení obrazových materiálů užívaných v aplikaci. Další typy zařízení jsou chytré hodinky a televize. Pro chytré hodinky nebude aplikace z této práce vyvíjena.

5.1.1 Struktura aplikace

V zadání diplomové práce je požadavek na vytvoření knihovny přehrávače. Aplikace bude rozdělena na dvě části. Část přehrávače, která bude knihovnou a část testovací aplikace. Knihovna přehrávače musí být implementovatelná do jiných aplikací bez nutnosti zásadních zásahů do kódu. Aby bylo možné knihovnu vytvořit bylo nutné se seznámit se strukturou projektu. Struktura je následující:

```
/ ..... Kořenový adresář projektu
├── .gradle .2 .idea .2 app
│   ├── build
│   ├── libs
│   ├── release
│   ├── src
│   │   ├── androidTest
│   │   ├── main ..... Zdrojové soubory aplikace
│   │   │   ├── java ..... Zdrojové soubory aplikace v jazyce Java
│   │   │   ├── res ..... Podpůrné soubory aplikace
│   │   │   └── AndroidManifest.xml
│   │   └── test
│   └── build.gradle
├── captures
├── gradle
├── build.gradle
├── gradle.properties
└── settings.gradle
```

Výše uvedená struktura je pouze orientační. Nejedná se o striktní podobu. Podoba projektové struktury se může mírně lišit.

Kořenový adresář projektu obsahuje různé soubory a složky. Nejdůležitější z nich budou následně popsány. Mezi hlavní složky patří: *build*, *lib*, a *src*. Ve složce *build* jsou uloženy výstupní binární soubory. Jedná se například o soubory s příponou *.apk*. Složka *lib* obsahuje knihovny použité v projektu a složka *src* obsahuje veškeré zdrojové soubory projektu. Mezi hlavní soubory projektu patří *build.gradle*, *gradle.properties* a *settings.gradle*. Je zřejmé, že všechny tři soubory se týkají automatizačního, sestavovacího nástroje Gradle. Nástroj Gradle se stará o celou režii projektu. Pomocí něj je prováděno importování knihoven, jejich nastavování, správa cílové verze SDK, pro kterou bude projekt tvořen a jak bylo zmíněno výše, i verze aplikace.

Složka *src* obsahuje projektové soubory, mezi které patří soubory tříd v jazyce Java, obrazové materiály použité v aplikaci a soubory (takzvané „layouts“), což jsou návrhy obrazovek tvořené v XML. Jeden z nejdůležitějších souborů ve složce *src*, je soubor *AndroidManifest.xml*. Jedná se o hlavní aplikační soubor, kde jsou uvedeny všechny dostupné aktivity aplikace, oprávnění aplikace (oprávnění pro přístup k síti Internet, přístup k místnímu úložišti, apod. . .). Složka s názvem *res* obsahuje obrazové a návrhové soubory. Soubory jsou děleny dále do složek podle jejich použití. Obrazové materiály jsou umístěny ve složce *drawable*, jedná se například o ikony tlačítek. Následuje složka *layouts*, kde jsou umístěny XML soubory s návrhy obrazovek aktivit. Na začátku této kapitoly bylo zmíněno, že lze vytvářet jednu aplikaci pro různá zařízení. K tomuto účelu existují projektové složky s názvy začínajcící spojením *minimap-**. Název vždy pokračuje kódovým označením rozlišení, pro které jsou soubory určeny. Nejčastěji jsou ve složkách s tímto názvem uloženy ikony aplikace a obrazové materiály, které podléhají destrukci při změně rozlišení (rastrové materiály). Poslední složkou je *values*. Poslední složka obsahuje projektové soubory s definovanými konstantami a nastaveními. Nastavení a konstanty ve složce *values* se týkají zejména nastavení barev, stylu aplikace a třeba i slovníky pro lokalizaci aplikace.

5.1.2 Terminologie a chování aplikace v operačním systému Android

Na následujících řádcích bude věnována pozornost termínům specifickým pro vývoj aplikací pro operační systém Android.

Plány a aktivity

Plány (v terminologii vývoje anglicky „layouts“) jsou návrhy obrazovek psané pomocí značkovacího jazyka XML. Příklad plánu je vidět ve výpisu č. 5.1.

Výpis 5.1: Ukázka jednoduchého plánu s tlačítkem

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <androidx.constraintlayout.widget.ConstraintLayout
4  xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      android:id="@+id/SplashScreen"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent">
9
10     <Button
11         android:id="@+id/button"
12         android:layout_width="0dp"
13         android:layout_height="48dp"
14         android:layout_marginStart="167dp"
15         android:layout_marginTop="229dp"
16         android:text="Button"
17         app:layout_constraintStart_toStartOf="parent"
18         app:layout_constraintTop_toTopOf="parent" />
19 </androidx.constraintlayout.widget.ConstraintLayout>
```

Výpis č. 5.1 obsahuje jednoduchý plán s jedním tlačítkem. Jako hlavní plocha návrhu slouží prvek jménem *ConstraintLayout*. Jedná se o prvek z knihovny AndroidX. AndroidX je nejnovější a doporučovaná knihovna prvků pro aplikace. Knihovna vznikla společně s projektem Material Design. [35] Plocha *ConstraintLayout* slouží k rozmístění ovládacích a zobrazovacích prvků v návrhu. V případě výpisu č. 5.1 je na ploše umístěno jedno tlačítko. Část s tlačítkem začíná od řádku číslo deset. První parametr tlačítka *android:id* značí ID prvku. Uvedené značky jsou pro všechny prvky stejné. Stejné nastavení ID bude i u ostatních prvků.

Jakmile je hotový plán obrazovky, je možné z něj vytvořit aktivitu. Aktivita vzniká spojením plánu v XML a souboru s třídou v jazyce Java. Spojení obou souborů vzniká v souboru *AndroidManifest.xml*. V předchozím odstavci bylo uvedeno, že všechny aktivity musejí být uvedeny v souboru *AndroidManifest.xml*. Je to právě z tohoto důvodu, aby bylo možné propojit plán s funkcionalitou. Je pravidlem, aby bylo možné vytvořit aktivity, musí mít třída v jazyce Java stejný název jako vytvořený plán. Ve výpisu č. 5.2 je uvedena ukázka vytvoření nové aktivity.

Výpis 5.2: Ukázka akivity v souboru *AndroidManifest.xml*

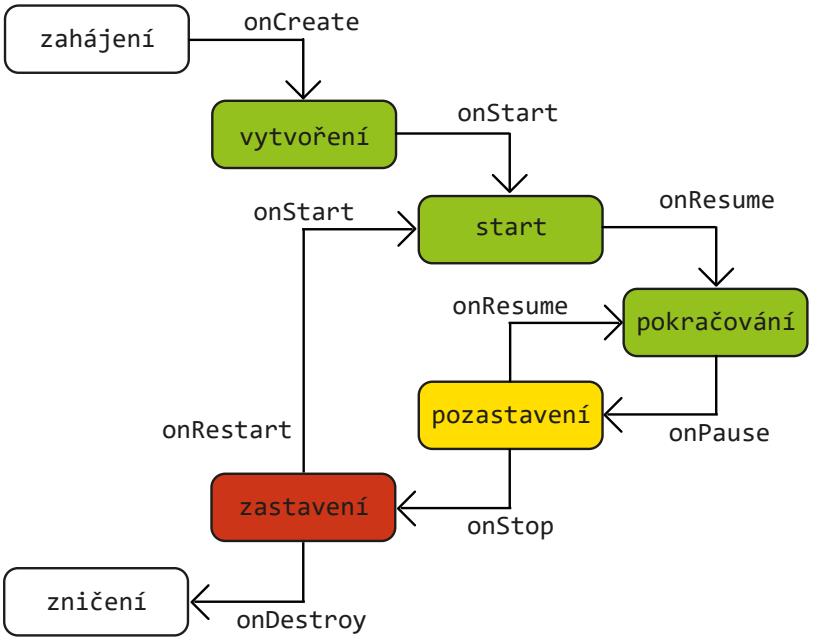
```
1 <activity android:name=".SplashScreen">
2   <intent-filter>
3     <action android:name="android.intent.action.MAIN" />
4     <category
        android:name="android.intent.category.LAUNCHER" />
5   </intent-filter>
6 </activity>
```

Na prvním řádku je uveden název aktivity. V případě výpisu č. 5.2 se jedná o název „SplashScreen“. Na třetím řádku je uvedeno, že se jedná o hlavní aktivitu a bude tedy spuštěna při startu jako první. Čtvrtý řádek označuje aktivitu jako „LAUNCHER“, to z aktivity dělá takzvanou spouštěcí aktivitu. Spouštěcí aktivity jsou zobrazovány při startu aplikace a slouží jako obrazovka během načítání dat pro spuštění aplikace. Jedná se například o stažení dat ze sítě Internet. Přítomnost spouštěcí aktivity není podmínkou, jedná se ale o pohodlnou přípravu dat. Existuje nepsané pravidlo, že spouštěcí aktivita musí být zobrazena co nejkratší časový úsek, aby nedošlo k velkému prodlení špuštění aplikace a uživatel nebyl nucen čekat. Jakmile jsou splněny výše uvedené předpoklady, je možné aplikaci spustit.

Životní cyklus aplikace v operačním systému Android

Aktivity a potažmo celá aplikace v systému Android prochází tzv. „životním cyklem“. Životní cyklus aplikace upravuje chování a volání metod při vytváření aktivit, jejich uspání a zavírání. Celý životní cyklus je zobrazen na Obr. č. 5.1. Celý cyklus začíná fází zahájení. Při zahájení je volána metoda *onCreate*. Metoda *onCreate* je nezbytná a musí být přítomna v každé aktivitě. [36] Jakmile je aktivita vytvořena, je zavolána metoda *onStart*. V metodě *onStart* je možné nastavovat počáteční podmínky aktivity. Po startu je volána metoda *onResume*. Metoda *onResume* slouží hlavně v dalším případě, když je aplikace probuzena. Další stav, pozastavení, je vyvolán, pokud je aplikace minimalizována. Operační systém z důvodu úspory energie veškeré nepotřebné aktivity uspává. Pokud tedy aplikace nemá dostatečná oprávnění, je uspána. Aktivity, které mohou být na pozadí aktivní jsou například ty, které komunikují se senzory telefonu (geolokační senzor, senzor tepu, gyroskop, apod. . .), nebo přehrávající multimédia. Jakmile je aplikace opět vyvolána z minimalizovaného stavu, je opět volána metoda *onResume*. Je tedy vhodná pro nastavení aktivity do původního stavu. Při uspání aktivita ztrácí veškerá neuložená data. Pomocí metody *onResume* je tedy možné získat uložená data zpět. Při zavření aplikace dojde k volání metody *onStop*. Zde je možné zavolat destruktory tříd a jiné metody uvolňující využívané prostředky aktivitou. Speciální případ je restart aktivity, který se provádí

například při selhání. Při restartu se volá metoda *onRestart* a *onStart*.



Obr. 5.1: Životní cyklus aktivity

Předávání dat mezi aktivitami

Aplikace většinou tvoří více aktivit. Z hlediska vývoje je často nezbytné, aby byla umožněna mezi těmito aktivitami komunikace. Pro komunikaci a výměnu dat se využívá třída *Intent* a třída *Bundle*. Hlavní je využití třídy *Intent*. Využívá se pro komunikaci mezi aktivitami.[37]

Výpis 5.3: Ukázka použití třídy *Intent* a *Bundle*

```
1 Intent intent = new Intent();
2 intent.setClass(this, Other_Activity.class);
3 intent.putExtra("UNIQUE_ID", "DATA");
4
5 Bundle bundle=new Bundle();
6 bundle.put("UNIQUE_KEY", "DATA");
7 intent.putExtra(bundle);
8 startActivity(intent);
```

Ve výpisu č. 5.3 je ukázka použití těchto tříd. Na řádku číslo jedna je uvedeno vytvoření třídy *intent*. Na řádku dvě je přiřazení cílové třídy do objektu *intent*. Další řádek obsahuje vložení dat. Data jsou vkládána jako klíč a data. Do objektu *intent* lze vkládat všechny základní datové typy, včetně pokročilejších, jako jsou například

ArrayListy. Je možné vkládat i své vlastní objekty. Podmínkou úspěšného vložení vlastního objektu je implementace rozšíření třídy objektu o třídu *Parcelable*.

Aby byl kód logičtěji členěn je možné vkládaná data dělit do logických svazků. K tomuto účelu slouží třída *Bundle*. Práce s třídou *Bundle* je ukázána ve výpisu č. 5.3 od řádku číslo pět. Vkládání dat do objektu funguje stejným způsobem, jako u třídy *Intent*. Po vložení všech dat je možné vyvolat novou aktivitu. Tato akce je zobrazena na řádku číslo osm.

5.2 Vývoj aplikace

Ze zadání práce vyplývá požadavek stanovující formu přehrávače jako knihovnu aplikovatelnou do jakékoliv další aplikace pro platformu Android. Protože vývoj vlastního základu pro přehrávání multimediálního obsahu by byl na platformě android zbytečný a časově náročný, byl na základě poznatků z sekce č. 2 vybrán framework ExoPlayer.

Základ tedy bude framework ExoPlayer, který bude obstarávat vykreslování obsahu. Výběr řešení ExoPlayer se opíral i o skutečnost, že projekt je spravován společností Google, Inc., početné uživatelské základny a stále aktivního vývoje. Autor této práce čerpal znalosti a inspiraci z ukázkové aplikace, která se nachází přímo v projektu ExoPlayer a je oficiálním zdrojem postupů při práci s kódem přehrávače.

5.2.1 Základní struktura projektu ExoPlayer

Sekce č. 2 dále popisovala, že framework ExoPlayer využívá systém doplňků. Celý framework je tedy poskládán z na sobě nezávislých doplňků, které lze v případě nevyužitelnosti z projektu odstranit. ExoPlayer je tvořen následujícími základními doplňky:

- exoplayer-library
- exoplayer-libraty-core
- exoplayer-library-dash
- exoplayer-library-hls
- exoplayer-library-smoothstreaming
- exoplayer-library-ui

Dále je tvořen rozšířeními, které dodávají přehrávači další funkcionalitu:

- exoplayer-extension-cast
- exoplayer-extension-cronet
- exoplayer-extension-ffmpeg
- exoplayer-extension-flac
- exoplayer-extension-gvr

- exoplayer-extension-ima
- exoplayer-extension-jobdispatcher
- exoplayer-extension-leanback
- exoplayer-extension-mediasession
- exoplayer-extension-okhttp
- exoplayer-extension-opus
- exoplayer-extension-rtmp
- exoplayer-extension-vp9

Mezi hlavní doplňky této práce se řadí zejména doplňky uvedené v prvním seznamu a dále také doplněk pro podporu kolekce FFMPEG a doplněk IMA (*Interaktivní inzerce médií – Interactive Media Advertising* (IMA)) pro podporu reklamního formátu VAST a VMAP. Implementace doplňků do projektu probíhá opět skrze sestavovací nástroj Gradle. Pro vložení doplňku do vlastního projektu se využívá dvou klíčových slov *implementation project*. Ukázka vložení některých doplňků je uvedena ve výpisu č. 5.4.

Výpis 5.4: Ukázka vložení doplňků ExoPlayer do projektu

```
1 implementation project(':exoplayer-library-core')
2 implementation project(':exoplayer-library-dash')
3 implementation project(':exoplayer-library-ui')
```

Vývojář má na výběr dvě možnosti. První možnost je implementace frameworku ExoPlayer pomocí lokálních doplňků. Způsob užití lokálního doplňku je zobrazen ve výpisu č. 5.4. Druhá možnost je využití online knihoven, které se automaticky udržují stále aktuální. Obě možnosti s sebou nesou výhody i nevýhody. Nevýhody první možnosti spočívají v neaktuálnosti knihoven a nutnosti je ručně aktualizovat. Výhodou první možnosti však je to, že vývojář má všechny aktualizace ve vlastní režii a má všechny změny pod kontrolou. Nevýhody druhé možnosti jsou právě v automatických aktualizacích, které probíhají mimo vývojářovu kontrolu. Může tedy nastat situace, kdy dojde k aktualizaci knihovny a třídy, které jsou v projektu použity a projdou při aktualizaci změnou. Tato změna se promítne i do projektu vývojáře a jeho kód začne vykazovat chyby. Výhoda druhé možnosti je opět v aktuálnosti celého projektu. V této práci byla zvolena první popsaná metoda.

Nastavení verze Java

Pro správné fungování frameworku ExoPlayer je nutné nastavit verzi prostředí Java. Opět se vše odehrává v souboru nastavení *build.gradle*. Aktuální použitá verze je verze 1.8. Nastavení verze je ukázáno ve výpisu č. 5.5.

Výpis 5.5: Ukázka nastavení verze Java

```
compileOptions {
  sourceCompatibility JavaVersion.VERSION_1_8
}
```

Práce s přehrávačem ExoPlayer

Přehrávač ExoPlayer využívá třídy *SimpleExoPlayer*. Vytvoření objektu z této třídy se provede pomocí instance třídy *ExoPlayerFactory*. Do této instance se pomocí metody *newSimpleInstance* předá parametr kontextu aplikace. Pomocí uvedené metody se vytvoří přehrávač. [38] Příklad vytvoření přehrávače je uveden ve výpisu č. 5.6.

Výpis 5.6: Ukázka vytvoření přehrávače

```
SimpleExoPlayer player =
   ExoPlayerFactory.newSimpleInstance(context);
```

Přehrávač však existuje pouze na pozadí a v tuto chvíli dokáže přehrávat pouze zvuk. Nemá nastavenu plochu, na kterou může vykreslovat obrazový výstup. V plánu aktivity je tedy nutné vytvořit plátno, které dokáže přebírat výstup z ExoPlayeru. Zmíněné plátno se jmenuje *ExoPlayerView*. Po vytvoření plátna je možné navázat přehrávač se zobrazovacím plátnem pomocí nastavovací metody *setPlayer*. [38] Propojení přehrávače se zobrazovacím plátnem lze vidět ve výpisu č. 5.7.

Výpis 5.7: Ukázka vytvoření přehrávače

```
playerView.setPlayer(player);
```

Nyní je vytvořen přehrávač s možností vykreslování výstupu videa. Stále však přehrávač nemá žádné médium, které lze přehrát. Všechna média k přehrání v přehrávači ExoPlayer, jsou reprezentována objektem *MediaSource*. Pokud se jedná o adaptivní stream, má každý typ streamu svůj vlastní typ objektu. Pro HLS existuje objekt *HlsMediaSource*, pro MPEG–DASH to je *DashMediaSource* a pro SmoothStream - *SsMediaSource*. Pro ostatní klasické multimediální kontejnery lze využít základní typ *MediaSource*.

Výpis 5.8: Ukázka přípravy přehrávače [38]

```
1 // Vytvoří instanci DataSource, skrze kterou jsou média
     načtena
2 DataSource.Factory dataSourceFactory = new
     DefaultDataSourceFactory(context,
3      Util.getUserAgent(context, "AdvancedPlayer"));
4 // Toto je zdroj médií reprezentující, média k~přehrání
5 MediaSource videoSource = new
     ExtractorMediaSource.Factory(dataSourceFactory)
6      .createMediaSource(mp4VideoUri);
7 // Příprava přehrávače se zdrojem multimédií
8 player.prepare(videoSource);
```

Z výpisu č. 5.8 lze vidět základní přidání média do přehrávače. Po přidání média do přehrávače může začít přehrávání. Pomocí výše uvedeného kódu lze vytvořit jednoduchý přehrávač.

5.2.2 Fungování přehrávače

Po propojení knihovny přehrávače s obsluhující aplikací se aktivita přehrávání volá klasicky pomocí objektu *Intent*. Knihovna se skládá ze dvou aktivit. První aktivita je určena pro přípravu přehrávače a druhá je samotný přehrávač. Z rodičovské aplikace je nutné volat aktivitu pro přípravu. Aktivita pro přípravu přehrávače se jmenuje *PlaylistDownloader*. Aktivita *PlaylistDownloader* očekává adresu na soubor s informacemi o přehrávaném médiu ve formátu JSON a objekt nastavení přehrávače (více o nastavení přehrávače v sekci č. 5.2.8).

Ukázka souboru s informacemi je uvedena v příloze B, konkrétně se jedná o výpis č. B.1. Soubor ve formátu JSON má následující strukturu: v hlavním objektu souboru se nachází jméno média a základní popis. Dále se v objektu nachází informace o jednotlivých multimediálních proudech. Proudy mohou být čtyř druhů. Jedná se o typ MP4, MPEG–DASH, HLS a SmoothStream. Objekt s informacemi o multimediálním proudu v sobě nese informace o typu proudu, adresu, pojmenování stop (více stop platí pro adaptivní proud), informace o jazyku zvukové stopy a pokud se jedná o adaptivní proud chráněný technologií DRM, tak objekt s informacemi o ochraně. Obsahem objektu s informacemi o ochraně DRM je typ ochrany, adresa licenčního serveru, token pro autentizaci, certifikát a přesnou verzi ochrany. Hlavní objekt dále pokračuje objektem s informacemi o titulcích. Titulky obsahují název a adresu. Následuje objekt se značkami, které lze využít pro rozdělení média na kapitoly. Každá značka je tvořena její časovou polohou a popisem. Po značkách následuje logický datový typ o povolení přehrávání reklam. Jako poslední se v souboru

nachází pole s odkazy na reklamní obsah.

Jakmile je adresa na soubor s informacemi předána aktivitě, je aktivitou spuštěno paralelní vlákno pro stažení souboru. Paralelní vlákno pro stažení je zapotřebí z důvodu omezení stahování z Internetu v hlavním vlákně aktivity. Omezení stahování v hlavním vlákně je dáno politikou operačního systému Android. [39] Jakmile je soubor stažen je převeden do textového řetězce a ten následně do objektu typu *JSONObject*. V rámci třídy aktivity *PlaylistDownloader* je vytvořeno několik podpůrných tříd, které slouží k uložení informací o médiu ze staženého souboru. Třídy byly vytvořeny pro jednodušší a přehlednější práci s daty v třídě. Jedná se o následující třídy:

- **Playlist** - třída sdružující všechny další třídy. Jedná se o objekt pro celé přehrávané médium,
- **Profile** - třída pro reprezentaci jednoho typu proudu, nebo jeden MP4 soubor,
- **ProfileDRM** - pokud médiu obsahuje zabezpečení obsahu jsou informace uloženy zde,
- **Subtitles** - třída pro informace o titulcích,
- **VideoTimeFlags** - třída pro informace o časových značkách,
- **VideoAds** - třída pro informace o reklamním obsahu.

Po vytvoření objektu *JSONObject* a jeho naplnění jsou data v něm obsažená předána do příslušných objektů. Jakmile jsou data připravena, je vytvořen nový objekt typu *Intent*. Objekt typu *Intent* je naplněn na základě informací z objektu *Playlist* a dalších obsažených objektů. To znamená zejména to, že pokud nějaká z položek není obsažena (reklamy, titulky, časové značky, apod. . .), není tato informace předávána do přehrávače. Po vytovření a naplnění objektu typu *Intent* je zavolána aktivita přehrávače.

V aktivitě přehrávače jsou z objektu *Intent* vyčteny všechny obsažené informace. Během čtení se postupně vytvářejí objekty *MediaSource* a její podmnožiny. Po vyčtení všech informací a vytvoření objektů s odkazy na médium dojde k automatickému spuštění přehrávání.

5.2.3 Základní ovládání přehrávače

Přehrávač jako takový je třeba ovládat. Základní požadavky na ovládání se týkaly klasických funkcí přehrávače. Jednalo se zejména o interakce typu: přehrát, pozastavit, posunout se v čase a přehrát další video. Protože bylo nevyhovující základní ovládání přehrávače, bylo nutné vytvořit vlastní. Vlastní ovládací prvky se vytvářejí stejně jako nový plán pro aktivitu. Rozdíl je v absenci zanesení aktivity do souboru *AndroidManifes.xml* a není třeba vytvářet soubor s třídou. Framework ExoPlayer poskytuje vlastní ovládací prvky, které lze využít. V práci bylo z nabízených vyu-

žito pouze tlačítko pro přehrání a pozastavení. Ostatní tlačítka bylo nutno uzpůsobit vlastním potřebám. Jak již bylo uvedeno, pro vytvoření vlastního ovládacího panelu je nutné vytvořit nový návrh (layout). Návrh byl umístěn do složky *layouts* a pojmenován *custom_playback_control.xml*. Vlastní ovládací prvky obsahují i rozhraní pro přepínání profilů videa, zvukové stopy, zobrazení titulků a přepínání jejich jazyka. Dále je implementováno rozhraní pro práci s hlasitostí audia. Poslední implementovaná funkcionalita vlastního ovládání přehrávače je zobrazení názvu a popis přehrávaného média.

Přehrát a pozastavit

Tlačítka přehrát a pozastavit představují základní ovládací prvky přehrávače. Jako jediné využívají metody frameworku ExoPlayer. Prvek tlačítka je spárován s frameworkem pomocí unikátního ID, které pro funkci přehrát a pozastavit obsahuje *exo_play*, respektive *exo_pause*. Jakmile je prvek správně pojmenován, kód frameworku zajistí prvku funkčnost.

Framework ExoPlayer obsahuje pro pozastavení a pokračování přehrávání jednoduchou metodu. Jedná se o metodu *setPlayWhenReady()* s pravdivostním parametrem. Užití metody *setPlayWhenReady()* lze vidět ve výpisu č. 5.9.

Výpis 5.9: Ukázka zapnutí a vypnutí přehrávání

```
1 //Přehrávání
2 player.setPlayWhenReady(true);
3 //Pozastaveno
4 player.setPlayWhenReady(false);
```

Posunutí přehrávání v čase

Posunutí přehrávání v čase, takzvané „seekování“ byla jedna z funkcionalit, které bylo třeba upravit dle vlastních potřeb. Jedna z funkcí přehrávače je zákaz posunu v čase přehrávání. Zákaz se u ovládacích prvků projevuje jejich zmizením. Pokud však byla tlačítka spravována přímo frameworkem, nebylo možné jim po spuštění dodatečně nastavovat parametry viditelnosti. Proto byly ovládací prvky pro posun v čase přehrávání vytvořeny vlastní. Ovládací prvky mají stejnou funkcionalitu jako původní spravované frameworkem, rozšířenou o možnost jejich skrytí. Vytvoření tlačítka posunu vpřed lze vidět ve výpisu č. 5.10.

Výpis 5.10: Ukázka vytvoření tlačítka pro posunutí v čase přehrání vpřed

```
1  // Vytvoří objekt ImageButton a předá do něj tlačítko
       podle jeho ID
2  ImageButton forward = findViewById(R.id.custom_exo_ffwd);
3  // U~tlačítka nastaví listenner na kliknutí
4          forward.setOnClickListener(v -> {
5              //Nastaví výchozí hodnotu o~kolik se bude
                   pozice přehrávání posunovat v~čase
6              int fastForwardMs =
                   PlayerControlView.DEFAULT_FAST_FORWARD_MS;
7              // Získá celkovou délku přehrávaného média
8              long durationMs = player.getDuration();
9              // Nastaví novou pozici pro přehrávání
10             long seekPositionMs =
                   player.getCurrentPosition() +
                   fastForwardMs;
11             if (durationMs != C.TIME_UNSET) {
12                 seekPositionMs = Math.min(seekPositionMs,
                       durationMs);
13             }
14             // Nastaví přehrávanou pozici v~přehrávači
15             player.seekTo(seekPositionMs);
16         });
```

Další/Předchozí přehrávaná položka

Ovládací tlačítka pro přepínání multimediálního obsahu využívají vlastnosti přehrávače s předáváním adresy na soubor s informace o médiu. Ovládací prvky jsou naprogramované, aby zničily aktuální aktivitu přehrávače a předaly aktivitě *PlaylistDownloader* novou adresu. Obě tlačítka obsahují akci při jejich stisknutí, která odkazuje na metodu *playRandomProductFromPlaylist()*. Výpis metody *playRandomProductFromPlaylist()* lze vidět na výpisu č. 5.11.

Výpis 5.11: Ukázka metody pro přeskočení přehrávání na další multimediální obsah

```
1  private void playRandomProductFromPlaylist(){
2    String randURL = "http://example.org/getnextplay.php";
3    PlayerConfig playerConfig = new PlayerConfig();
4    //Zničení aktivity přehrávače
5    playerActivity.finish();
6    Intent intent = new Intent(this,
7    cz.janbenedikt.advancedplayer.PlaylistDownloader.class);
8    intent.putExtra(Tags.PLAYLIST_URL, randURL);
9    intent.putExtra(Tags.PLAYER_CONFIGURATION,
       playerConfig);
10   intent.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
11   this.overridePendingTransition(0, 0);
12   this.startActivity(intent);
13 }
```

Adresa obsažená v metodě *playRandomProductFromPlaylist()* generuje náhodný soubor s informacemi o médiu a předává ho aktivitě *PlaylistDownloader*. Aby zničení aktivity a její přepnutí nešlo poznat je zde využito vypnutí animací přechodu mezi aktivitami, to lze vidět na řádku č. 10 a 11, ve výpisu č. 5.11.

Ukazatel průběhu

Ukazatel průběhu je dalším prvkem, který využívá funkcionality frameworku ExoPlayer. Protože vznikl stejný problém s dodatečným nastavením prvku, jako u tlačítek pro posun v čase, bylo pro zákaz posunu v čase nutné nastavit celý objekt přehrávače. Více o nastavení přehrávače pro zákaz posunu v čase je uvedeno v podsekci č. 5.2.8.

Kapitoly/záložky na časové ose

V souboru s informacemi o médiu je sekce s časovými značkami pro vyznačení na časové ose. Značky jsou uvedeny ve formátu, který definuje standard ISO 8601. [40] Pro jeho převedení do linuxového formátu časové značky je využita metoda *java.time.Duration.parse()*. Po převedení jsou značky předány do metody *setExtraAdGroupMarkers()*. Metoda je poskytována frameworkem ExoPlayer a je zároveň využívána knihovnou IMA pro označení časového úseku, kde bude přehrán reklamní obsah. To s sebou nese problém ve stejné barvě značek. Při současném použití obou typů značek je není možné od sebe rozlišit. Na tento problém nebylo nalezeno žádné řešení.

Hlasitost a ztlumení

Nedílnou součástí přehrávače je i ovládání hlasitosti. Jedná se o další ovládací prvky, které byly přidány na kontrolní panel přehrávače. Konkrétně se jedná o tlačítko pro ztlumení a posuvník pro nastavení hlasitosti. Tlačítko zároveň funguje jako indikátor úrovně hlasitosti. Indikace je provedena ikonou.

Úroveň hlasitosti je nastavována pomocí objektu *AudioManager*.

5.2.4 Přepínání přehrávaných stop

Přehrávač disponuje možností přehrávat adaptivní streamy definované v kapitole 1.1. Konkrétně se jedná o MPEG–DASH, HLS a dále také SmoothStream. Všechny typy streamů v sobě mohou obsahovat různé profily videa, audia a titulků.

V rámci video profilů se může přehrávač nacházet ve dvou režimech. První režim vybírá nejvhodnější profil na základě dostupné šířky pásma připojení. Při automatickém režimu je z manifest souboru příslušného adaptivního streamu vyčtena šířka pásma potřebná pro stahování daného profilu. Přehrávač zjišťuje dostupnou šířku pásma pomocí objektu *DefaultBandwidthMeter*, který je součástí objektu *DefaultTrackSelector*. Druhý režim spočívá v manuálním režimu, který je aktivován výběrem jednoho z dostupných profilů zobrazených pomocí výběru na kontrolním panelu přehrávače.

Pro přepínání všech profilů (audio, video i titulků), byla vytvořena třída s názvem *TrackChanger*. Systém přepínání profilů se liší na základě toho, zdali jde o adaptvní stream, nebo jde-li o přehrávání MP4 souboru. Pro oba případy se využívá stejná třída.

Adaptivní stream

U adaptivního streamu je využit objekt *DefaultTrackSelector*. Objekt *TrackSelector* v sobě uchovává všechny dostupné profily. Profily jsou rozděleny podle čísla *GroupIndex* a *TrackIndex*. Jak z názvů vyplývá, *GroupIndex* indikuje číslo skupiny. Skupiny jsou tři. První skupina jsou profily videa. Skupiny jsou číslované od nuly, proto první skupina má číslo 0. Následuje druhá skupina. V druhé skupině jsou uložené profily audia a má číslo 1. Poslední skupina obsahuje titulky a má číslo 2. *TrackIndex* určuje jednotlivé profily ve skupině. Pomocí *GroupIndex* a *TrackIndex* vznikne unikátní dvojčíslí, které identifikuje konkrétní profil, ať se jedná o video, audio, nebo titulky.

Aby bylo možné profily přepínat, je nutné nejprve zjistit, jaké profily objekt *DefaultTrackSelector* obsahuje. Profily lze získat pomocí metody *getTrackGroups*. Metoda je poskytována objektem *MappingTrackSelector*, který udržuje informace

o profilech z objektu *DefaultTrackSelector*. Celý proces získání profilů lze vidět ve výpisu č. 5.12.

Výpis 5.12: Ukázka zjištění dostupných profilů.

```
1 ArrayList<Profiles> profiles = new ArrayList<>();
2 MappingTrackSelector.MappedTrackInfo trackInfo =
3     trackSelector == null ? null :
         trackSelector.getCurrentMappedTrackInfo();
4 if (trackSelector == null || trackInfo == null) {
5     return null;
6 }
7 TrackGroupArray trackGroups =
     trackInfo.getTrackGroups(rendererIndex);
```

První je nutné vytvořit objekt *MappingTrackSelector*. Ten lze vytvořit vrácením metody z *getCurrentMappedTrackInfo*, která je poskytována objektem *DefaultTrackSelector*, což lze vidět na řádcích 2 až 3. Jakmile je objekt *MappingTrackSelector* naplněn, je možné z něj získat jednotlivé profily. Pro jednotlivé profily je přichystán *ArrayList* s objekty typu *Profiles*, které byly vytvořeny pro potřeby autora. Jeho inicializace je provedena na řádku číslo 1. *ArrayList* je naplněn na řádku číslo 7. Nyní má třída *TrackChanger* informace o dostupných profilech pro jednu skupinu (audio, video, titulky). Skupina je definována proměnnou *rendererIndex*.

Z dostupných informacích o profilech je nyní možné vygenerovat v ovládacím panelu jednotlivé ovládací prvky pro přepínání profilů.

Jakmile je na kontrolním panelu vybrán konkrétní profil, je na základě jeho indexu vybrán z dostupných profilů v objektu *DefaultTrackSelector*. Celý proces změny profilu je zobrazen ve výpisu č. 5.13. Indexy jsou uloženy v kontrolním prvku *radioButton*, který obsahuje identifikátor jménem *tag*. V *tag* identifikátoru jsou uloženy ID jednotlivých profilů. Při výběru konkrétního *radioButtonu* dojde k vyčtení hodnoty z *tagu* a ta je předána metodě, kde vybrán odpovídající profil. Pokud se jedná o výběr automatického přepínání profilů, je ID vždy stejné. Pro audio je ID automatického výběru 8888 a pro video 9999.

Výpis 5.13: Ukázka přepnutí profilu

```
1  @Nullable DefaultTrackSelector.SelectionOverride override;
2  //9999 a 8888 znamenají automatický výběr
3  if((trackIndex != 9999 && groupIndex != 9999)
4      && (trackIndex != 8888 && groupIndex != 8888)) {
5      override = new
           DefaultTrackSelector.SelectionOverride(groupIndex,
           trackIndex);
6  }else {
7           override = null;
8         }
9
10 DefaultTrackSelector.ParametersBuilder parametersBuilder
     = trackSelector.buildUponParameters();
11 parametersBuilder.setSelectionOverride(trackType,
     trackGroups, override);
12 trackSelector.setParameters(parametersBuilder);
```

O výběr profilu se stará konfigurace objektu *SelectionOverride*. Nastavení profilu se provádí v konstruktoru objektu. Nastavení pomocí konstruktoru je vidět na řádku číslo 5. Zůstane-li objekt typu *null*, jedná se o výběr automatického přepínání profilů.

Po vytvoření objektu *SelectionOverride* je předán společně s číslem skupiny a číslem profilu do nového objektu typu *ParametersBuilder*. Nový objekt *ParametersBuilder* je dále nastaven pomocí metody *setParameters* do objektu *DefaultTrackSelector*.

Přepínání profilů výše uvedenou metodou je společné pro všechny typy profilů (audio, video, titulky).

MP4 soubory

Jedním z požadavků zadaní bylo implementovat funkcionalitu která zajistí, aby bylo možné využívat několika MP4 souborů, podobně jako přepínání profilů u adaptivního streamu. Pouze s rozdílem, že nebude možné profily přepínat automaticky podle šířky pásma připojení.

V souboru s informacemi o předaném médiu v příloze B lze vidět v poli *streamsInfo* objekty s jednotlivými soubory (pouze v případě MP4). Aby celý systém přepínání kvalit a audio stop fungoval korektně, je nutné aby byly k dispozici všechny kvality se všemi audio stopami. Pokud tedy budou k dispozici dvě různé kvality videa a dvě jazykové audio stopy, celkově jsou nutné čtyři MP4 soubory.

Hlavní rozdíl oproti adaptivnímu streamu je v nemožnosti používat objekt *DefaultTrackSelector*. Objekt *DefaultTrackSelector* má totiž vždy k dispozici informace

pouze o jednom aktuálně přehrávaném MP4 souboru. Z tohoto důvodu byly vytvořeny pomocné třídy s názvem *MP4Profile*, které jsou uloženy v *ArrayListu*. Celý *ArrayList* uchovává informace o všech dostupných profilech. Pro každý soubor je vytvořen odpovídající objekt v *ArrayListu*, který představuje jeden profil. Hlavní identifikátory profilu jsou *VideoQuality* a *AudioLocale*.

Profily videa jsou reprezentovány stejně, jako u adaptivního streamu, soustavou ovládacích prvků typu *radioButton*. Stejně jsou reprezentovány i profily audia. Na kontrolním panelu jsou tedy umístěny dvě sady ovládacích prvků typu *radioButton*. Při změně profilu jedné sady se vyčítá aktuálně zvolený profil v sadě druhé. Pro změnu MP4 profilu jsou potřeba oba identifikátory profilu. Ze známých profilů v kolekci *ArrayList* se vybere ten MP4 soubor, který odpovídá oběma identifikátorům (kvalita videa a zvuková stopa). Vybraný profil obsahuje adresu MP4 souboru. Ve chvíli výběru se zaznamená aktuální pozice přehrávání a uloží se. Po nalezení odpovídajícího profilu se zastaví přehrávání a zavře se přehrávač. Pomocí objektu *Intent* se zavolá nová aktivita *PlaylistDownloader* a předá se jí adresa nového MP4 souboru společně s časovým údajem, který reprezentuje pozici přehrávání, kde došlo k přepnutí profilu. Po zavolání aktivity *PlaylistDownloader* se nastaví přehrávač a zapne se přehrávání od časové hodnoty, ve které došlo k přepnutí profilu. Rozdíl tedy spočívá v kompletním „zničení“ přehrávače a jeho novém sestavení oproti pouhému přepnutí profilu u adaptivního streamu.

5.2.5 Podpora DRM - Widevine

Přehrávač podporuje plně DRM technologii Widevine a částečně technologii Playready. Částečná podpora znamená plnou funkčnost pouze v televizích. Na mobilních zařízeních technologie PlayReady podporována není. Třetí typ FairPlay není podporován vůbec. Absence podpory FairPlay je zaviněna samotnou společností Apple, Inc., která podporu většinu svých technologií soustředí pouze na své produkty a pro ostatní platformy je implementace uzavřeného kódu nemožná. [14] Informace o DRM jsou umístěny v souboru s informacemi o médiu (v příloze B) a jsou označené jako „drmInfo“. Jedná se o pole objektů, kde jednotlivý objekt představuje jednu technologii DRM. Informace o DRM jsou uloženy v objektu *Profile*DRM.

Objekt *Profile*DRM obsahuje údaje o: typu DRM, offline token, typ klíčového systému, adresu licenčního serveru a adresu certifikátu. Offline token je využit, pokud manifest adaptivního streamu neobsahuje informace o licenčním klíči.

Po vytvoření příslušných objektů s informacemi o DRM, je vybrán vhodný typ. Nejvyšší prioritu má typ Widevine. Protože přehrávač nebyl pro televize uzpůsoben, není zatím podpora implementována. Je-li přítomen typ Widevine, jsou informace předány přehrávači. Pokud jsou přítomny offline klíče, je nastavena HTTP hlavička,

která je tvořena párem klíč-hodnota, kde klíč je „X-AxDRM-Message“ a hodnota je offline DRM klíč.

Při vytvoření aktivity *PlayerActivity* je z předaných informací vytvořen objekt *DrmSessionManager*, které je ukázáno ve výpisu č. 5.14.

Výpis 5.14: Ukázka vytvoření objektu *DrmSessionManager*

```
1 DefaultDrmSessionManager<FrameworkMediaCrypto>
    drmSessionManager = null;
2 String drmLicenseUrl =
    intent.getStringExtra(Tags.DRM_LICENSE_URL_EXTRA);
3 boolean multiSession =
    intent.getBooleanExtra(Tags.DRM_MULTI_SESSION_EXTRA,
    false);
4 UUID drmSchemeUuid =
    Util.getDrmUuid(intent.getStringExtra(drmSchemeExtra));
5 String[] keyRequestPropertiesArray =
    intent.getStringArrayExtra(
6 Tags.DRM_KEY_REQUEST_PROPERTIES_EXTRA);
7 drmSessionManager =
    buildDrmSessionManagerV18(drmSchemeUuid,
    drmLicenseUrl, keyRequestPropertiesArray,
    multiSession);
```

Vytvoření objektu *DrmSessionManager* se skládá ze tří vstupních proměnných. První je typ technologie DRM, následuje adresa licenčního serveru, pole offline klíčů a poslední zapnutí, nebo vypnutí rotace klíčů. Rotace klíčů je využívána v situaci, kdy se pravidelně mění klíč DRM.

Výpis 5.15: Ukázka vložení objektu *DrmSessionManager* do přehrávače

```
1 SimpleExoPlayer player =
    ExoPlayerFactory.newSimpleInstance(this,
    rendersFactory, trackSelector, drmSessionManager);
```

Ve výpisu č. 5.15 je zobrazeno aktivování podpory DRM v přehrávači. Podpora se zapíná vložením objektu do metody *newSimpleInstance*, objektu *ExoPlayerFactory*. Vkládá se společně kontextem aplikace (this), objektem *DefaultRenderersFactory* a objektem *DefaultTrackSelector*. Pokud je obsah objektu *DrmSessionManager* null, nastavení pro zabezpečený obsah se neuplatňuje a předpokládá se, že médium žádné zabezpečení neobsahuje.

5.2.6 Přehrávání reklamního obsahu

Reklamní obsah využívá standardů VAST a VMAP. Pro podporu zmíněných formátů je využito rozšíření frameworku ExoPlayer jménem IMA. Doplněk IMA poskytuje podporu pro všechny verze standardů. Konkrétně jde o verze: VAST 3.0, VAST 4.0, VPAID 2.0 a VMAP 1.0. Reklamní obsah se dělí do tří kategorií, podle umístění ve videu. První kategorie je „preRoll“. Jedná se o reklamu, která je automaticky spuštěna před samotným spuštěním videa. Následuje „midRoll“, která je umístěn do časového úseku během přehrávání videa. Poslední kategorií je „postRoll“, kde je reklama přehrána po konci videa.

Pro informace o reklamách, získaných ze souboru s informacemi o médiu, je určena třída s názvem *VideoAds*. Třída obsahuje kolekce typu *ArrayList* pro adresy na příslušné reklamy. Kolekce jsou dohromady tři. Každá pro jednu kategorii reklamního obsahu. Pro reklamy typu „midRoll“ je navíc připravena kolekce pro uložení času příslušné reklamy. Index času se musí shodovat s indexem adresy pro reklamu.

5.2.7 Zobrazení informací o médiu

Ve vytvořeném kontrolním panelu jsou umístěny textová pole pro zobrazení informací o videu, jako je například název a krátký popis. Hodnoty jsou převzaty ze souboru s informacemi o médiu (ukázka v příloze B).

5.2.8 Nastavení přehrávače

Přehrávač se nastavuje pomocí třídy *PlayerConfig*. Třída obsahuje údaje o zákazu seekování, výběru výchozí stopy titulků, výběru výchozí audio stopy, pozici začátku přehrávání a výchozí index MP4 souboru, který bude vybrán pro přehrávání.

Objekt *PlayerConfig* je nastaven podle parametrů z hlavní aplikace. Nastavení se neděje v knihovně přehrávače, ale mimo ni. Přehrávače je možné objekt *PlayerConfig* předat pomocí objektu *Intent* přímo s klíčem PLAYER_CONFIGURATION z třídy *Tags*.

5.2.9 Zákaz seekování

Seekování je metoda manuálního posunu v čase přehrávaného média. Seekování je možné provést pomocí tlačítek na ovládacím panelu přehrávače, nebo přímo na časové ose. V rámci konfigurace přehrávače je i možnost zákazu seekování. Nastavení zákazu je reprezentováno proměnnou *seekButtons*. Pro její nastavení třída poskytuje metodu *setSeekButtons*, kde parametrem je pravdivostní proměnná.

Přehrávání média z libovolné časové pozice

Další nastavení přehrávače je při zobrazení přehrávače, začít přehrávat médium z libovolné časové pozice. Pro nastavení se využívají časové značky délky definované v ISO 8601 [40]. Pro nastavení je dostupná metoda *setStartTime* s parametrem ve formátu *String*. Není-li hodnota nastavena, použije se výchozí nastavení – přehrávání od začátku.

5.2.10 Automatické přehrávání dalšího média

Přehrávač obsahuje tzv. „AutoPlayNext“ funkcionalitu. Jedná se o funkci, která na konci přehrávaného média automaticky začne přehrávat další. Funkce se chová stejně jako stisk tlačítek „další“, nebo „předchozí“.

Výpis 5.16: Ukázka odchycení události konce přehrávání

```
1 case Player.STATE_ENDED:
2     logToAnalytics("EventListenerState", "Playback ended!
        | Media name: " + playingMediaName);
3     Switch sw_autoPlay_next =
        findViewById(R.id.autoPlayNext_switch);
4     if(sw_autoPlay_next.isChecked()){
5         playRandomProductFromPlaylist();
6     }
7     player.setPlayWhenReady(false);
8 break;
```

Ve výpisu č. 5.16 je ukázka zachycené události konce přehrávání. Kód je součástí metody, která naslouchá událostem přehrávače. Při události konce přehrávání, která je reprezentována stavem *Player.STATE_ENDED*, je událost nejprve zaznamenána. Záznam události se nachází na řádku číslo 2. Následuje zjištění stavu komponenty typu přepínač z kontrolního panelu. Přepínačem si může uživatel zvolit, zdali chce pokračovat v přehrávání dalšího média. Pokud je přehrání dalšího média povoleno, je zavolána metoda *playRandomProductFromPlaylist*. Jedná se o stejnou metodu, která je volána u tlačítek „další“ a „předchozí“.

5.2.11 Univerzální posílání událostí o stavu přehrávače

Pro zaznamenávání událostí v přehrávači je aplikace propojena se službou Google Analytics. Struktura modulu propojení je popsána v sekci 3.3. Začlenění modulu do aplikace je provedeno v hlavní třídě aplikace.

Aplikace dokáže zaznamenávat tři druhy události. Prvním je obrazovka, na které se momentálně uživatel nachází. Nejedná se však o snímání obrazovky, ale pouze

o zaznamenání jejího názvu. Není tedy nijak narušeno soukromí uživatele. Ukázka zaznamenání otevření aktivity je ve výpisu č. 5.17.

Výpis 5.17: Ukázka zaznamenání otevření aktivity.

```
1 AdvancedPlayer.getInstance().trackScreenView("Player");
```

Druhým typem je záznam výjimky, která nastala. Využívá se v konstrukci „try/catch“, která dokáže zachytávat výjimky. V části „catch“ je zapsáno volání metody *trackException* pro uložení nastalé výjimky. Volání metody je zobrazeno ve výpisu č. 5.18.

Výpis 5.18: Ukázka zaznamenání výjimky.

```
1 AdvancedPlayer.getInstance().trackException(exception);
```

Poslední typ je univerzální zaznamenávání zpráv. Vývojář může zaznamenávat jakékoliv události a dělit je do kategorií. Metoda *trackEvent* obsahuje tři parametry. První parametr obsahuje název kategorie, následuje název akce a tělo zprávy. Přehrávač tento typ zpráv využívá například pro periodické sledování odehraného času média. Ukázka volání metody je uvedena ve výpisu č. 5.19. Využití zaznamenání univerzální metody zasílání zpráv je také uvedeno ve výpisu č. 5.16, na řádku č. 2, použité v metodě *logToAnalytics*.

Výpis 5.19: Ukázka zaznamenání obecné zprávy.

```
1 AdvancedPlayer.getInstance().trackEvent("Kategorie",
    "Akce", "Zpráva");
```

5.2.12 Třída Tag

Pro potřeby komunikace mezi aktivitami byla vytvořena třída *Tag* obsahující značky využívané pro identifikaci proměnných, vkládaných do objektů *Intent*.

Vybrané značky jsou uvedeny v následujícím seznamu:

- **DRM_SCHEME_EXTRA** – typ technologie DRM,
- **DRM_LICENSE_URL_EXTRA** – adresa licenčního serveru,
- **DRM_KEY_REQUEST_PROPERTIES_EXTRA** – offline klíče,
- **DRM_MULTI_SESSION_EXTRA** – povolení rotace klíčů,
- **PREFER_EXTENSION_DECODERS_EXTRA** – rozšíření pro dynamické streamy (využito u MPEG–DASH),
- **PLAYER_CONFIGURATION** – objekt nastavení přehrávače,
- **VIDEO_TIME_FLAGS** – pole časových značek,
- **PROFILES_NAMES** – pole jmen profilů,
- **PLAYLIST_MEDIA_NAME** – název média,
- **PLAYLIST_MEDIA_INFO** – popis média,

- **PLAYLIST_URL** – adresa pro soubor s informacemi o médiu.

Použití značky je zobrazeno ve výpisu č. 5.11, na řádcích číslo 8 a 9.

6 Vývoj testovací aplikace

Pro demonstraci a testování knihovny byla vytvořena testovací aplikace. Aplikace se skládá ze dvou aktivit. První aktivitou je tzv. „SplashScreen“. Aktivita je použita pro načítání dat ze sítě Internet a pro přípravu aplikace. Aktivita typu „SplashScreen“ je také popsána v sekci 5.1.2.

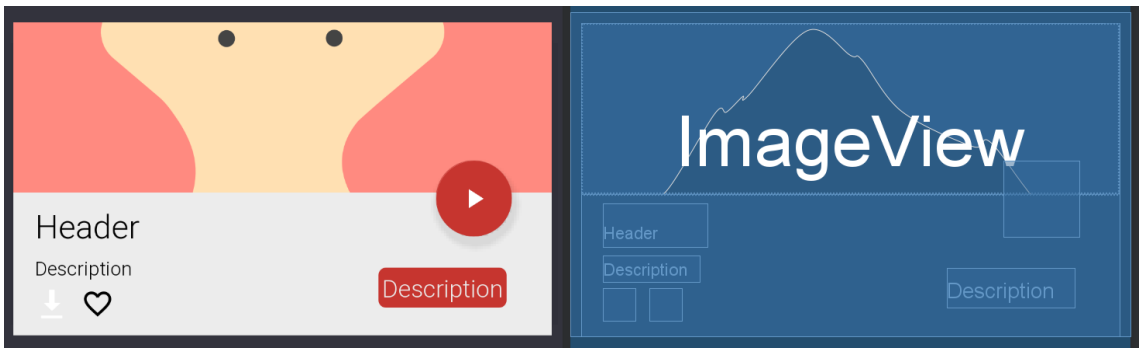
Třída aktivity „SplashScreen“ vytváří paralelní vlákno, které stahuje hlavní soubor s informacemi o dostupných médiích ze sítě Internet (ukázka souboru je v příloze C). Struktura souboru závisí na třídě, která se stará o analýzu dat ze souboru. Není tedy striktně zadána, jako v případě knihovny přehrávače. Po stáhnutí dat je soubor typu JSON zpracován a data z něj jsou uložena do objektů.

Po stažení a zpracování dat bude automaticky zobrazena aktivita s názvem *MainActivity*, která je druhou aktivitou testovací aplikace. Aktivita je tvořena menu s kartami představující jednotlivá média, dostupná k přehrání. Pro účely menu byla vytvořena vlastní komponenta, představující kartu s informacemi o médiu.

6.1 Vytvoření vlastní komponenty

Aby byly vykresleny všechny informace o médiu, včetně jeho náhledového obrázku, byla vytvořena jednoduchá komponenta. Komponenta představuje kartu menu, obsahující náhledový obrázek, název média, krátký popis a spouštěcí tlačítko pro přehrání média. Dále jsou v komponentě prvky, které zatím nemají funkční využití. Jedná se o tlačítko pro stažení média, tlačítko pro přidání média mezi oblíbené a popisek pro rozdělení do kategorií.

Vytvoření vlastní komponenty začíná vytvořením plánu. V plánu jsou umístěny všechny komponenty, které budou na kartě zobrazeny. Návrh je zobrazen na obrázku č. 6.1.



Obr. 6.1: Návrh vlastní komponenty

V návrhu jsou uvedeny dva módy, které lze při navrhování použít. Mód vlevo je ukázka reálného zobrazení komponenty, nebo aktivity v aplikaci. Návrh vpravo se využívá pro umisťování komponent a jejich vzdálenostní vztahy mezi sebou. Komponenty lze umisťovat v závislosti na vzdálenosti jiné komponenty. Vzdálenostní vztahy se využívají například v případě různých rozlišení obrazovek, kde nelze využít pevné usazení komponent. Při pevném usazení by mohlo dojít k posunu komponent na jiné místa, nebo k jejich překrývání.

Komponenty mají dvě třídy. První třída nese název „Adapter“ a druhá „View“. Třída „Adapter“ slouží k nastavování dat jednotlivým komponentům v kartě. Lze ji využít pro nastavení názvu pro textové pole *Header* (viz návrh v obrázku č. 6.1) atd. *RecyclerView.Adapter<VideoCardAdapter.CardViewHolder>* je mateřskou třídou, ze které dědí třída „Adapter". Třída „Adapter"poskytuje rozhraní *recyclerView*, ve kterém jsou karty umístěny. Pro nastavení dat karty je v třídě *MainActivity* vytvořen objekt *adapter*, kam je předán *List* s informacemi o médiích (viz výpis č. 6.1).

Výpis 6.1: Ukázka vytvoření objektu *Adapter*

```
1 VideoCardAdapter adapter = new
     VideoCardAdapter(productList, this);
```

Každá karta obsahuje tlačítko, které volá knihovnu přehrávače. Aby bylo možné nastavit pro každou kartu vlastní volání, je v třídě *Adapter* vytvářena vlastní akce kliknutí tlačítka. Adresa s odkazem souboru s informacemi o médiu je uložena v parametru *Tag* tlačítka. Tím je zaručeno, že každá karta bude obsahovat adresu vlastního média.

Třída *View* slouží k přístupu ke komponentám karty. Obsahuje předávací metody komponent pro nastavování parametrů a pro získávání informací o komponentě.

6.2 Implementace knihovny přehrávače do testovací aplikace

Knihovna se implementuje do projektu pomocí nástroje Gradle. Jako první je nezbytné vložit složku s knihovnou do projektu spouštěcí aplikace. Složku s knihovnou je nutné vložit do kořenové složky projektu, bude-li složka umístěna jinde, je nutné přizpůsobit kód ve výpisu č. 6.2. Jakmile je knihovna umístěna v projektu, je nutné vložit implementační kód z výpisu č. 6.2 do *settings.gradle*.

Výpis 6.2: Ukázka implementace knihovny v souboru *settings.gradle*

```
1 include ':app', ':advancedplayer', ':googleanalytics'
2 gradle.ext.exoplayerRoot = 'advancedplayer/libs/ExoPlayer'
3 gradle.ext.exoplayerModulePrefix = 'exoplayer-'
4 apply from: new File(gradle.ext.exoplayerRoot,
    'core_settings.gradle')
5 project(':googleanalytics').projectDir = new
    File('advancedplayer/libs/googleanalytics')
```

Jakmile je implementační kód vložen a soubor *settings.gradle* uložen, je nutné spustit synchronizaci nástroje Gradle. Během synchronizace je prováděno provázání knihovny s projektem.

Po dokončení synchronizace je posledním krokem importování třídy *PlayerConfig*, která se stará o nastavení přehrávače. Import je proveden kódem, který je uveden ve výpisu č. 6.3.

Výpis 6.3: Ukázka implementace třídy *PlayerConfig*

```
1 import cz.janbenedikt.advancedplayer.PlayerConfig;
```

6.2.1 Předání adresy média a volání aktivity z knihovny

Pro spuštění přehrávače je využit objekt *Intent*. Na řádku číslo 1, výpisu č. 6.4 je uvedeno vytvoření objektu *Intent*, společně s konstruktorem, který obsahuje odkaz na třídu přehrávače. Je vždy nutné volat uvedenou třídu *PlaylistDownloader*. Do objektu *Intent* jsou následně vkládány dvě proměnné. Jako klíč při vkládání je využita třída *Tag* (viz. sekce 5.2.12). Konkrétně jde o Tag.PLAYLIST_URL a Tag.PLAYER_CONFIGURATION. První proměnná je odkaz na soubor s informacemi o médiu. Ukázka souboru je v příloze B. Druhým je objekt *PlayerConfig*. Po naplnění objektu *Intent* je zavolána metoda z aktuální třídy *startActivity*, kde v parametru je předán objekt *Intent*. Po otevření začne automatické přehrávání.

Výpis 6.4: Ukázka volání aktivity přehrávače

```
1 Intent intent = new Intent(mainContext,
    cz.janbenedikt.advancedplayer.PlaylistDownloader.class);
2 intent.putExtra(Tags.PLAYLIST_URL, "https://
    example.org/products/id-001/play");
3 intent.putExtra(Tags.PLAYER_CONFIGURATION,
    item.getPlayerConfig());
4 mainContext.startActivity(intent);
```

7 Závěr

Cílem práce byla realizace knihovny přehrávače multimediálního obsahu s podporou pokročilých služeb. Jednalo se zejména o reklamní formáty, odesílání statistik, zabezpečení obsahu a podporu adaptivního streamu.

V kapitole č. 1 byly popsány technologie využívané v oblasti přehrávání multimédií. Jako první se kapitola věnovala multimediálním kontejnerům MP4 a MPEG-TS. Kapitola pokračovala popisem technologií adaptivního streamu a zabezpečení obsahu DRM. V závěru kapitoly byla věnována pozornost službě Google Analytics a reklamním formátům VAST, VMAP a VPAID.

Po popisu technologií následoval rozbor nalezených řešení. Celkově se jednalo o tři různá řešení. První dvě (přehrávač postavený na technologii Gstreamer a nativní rozhraní přehrávače systému Android) nevyhovovala stanoveným požadavkům. Poslední popsané řešení s názvem ExoPlayer disponovalo podporou všech stanovených požadavků a byl zvolen jako výchozí framework, na kterém bude knihovna přehrávače postavena. Na základě zvoleného frameworku přehrávače byl v kapitole č. 2 vytvořen popis funkcionality, kterou framework disponuje. Kapitola č. 2 pokračovala popisem struktury frameworku a využitím struktur balíčků. V poslední části kapitoly byl popsán programový kód a nástroj Gradle.

Kapitola č. 3 se věnovala návrhu knihovny přehrávače. Jako první byly stanoveny podmínky obsahu technologií a požadované vlastnosti pokročilého přehrávače. Návrh pokročilého přehrávače vycházel z frameworku ExoPlayer. Zvolený framework byl jako předpoklad zohledněn i v UML návrhu pokročilého přehrávače. Po teoretickém návrhu přehrávače byl popsán vytvořený modul pro odesílání statistik o využívání aplikace do systému Google Analytics.

Následuje kapitola č. 4 popisující problematiku přípravy a publikování aplikace v obchodě Play Store.

Vývoj byl rozdělen do dvou kapitol. Kapitola č. 5 se věnovala popisu vývoje knihovny přehrávače. Popis vývoje knihovny je rozdělen do sekcí podle funkcionality přehrávače, které jsou řazeny od základního ovládání, po komplexnější funkce. Následovala kapitola č. 6, obsahující popis vývoje testovací aplikace, určenou k ověření funkčnosti knihovny přehrávače.

Součástí práce je také testovací aplikace, do které byla knihovna implementována. Po implementaci knihovny byla pomocí testovací aplikace ověřena funkčnost knihovny. Testovací aplikace společně s knihovnou byly publikovány do obchodu Play Store, kde splnily podmínky pro publikování společnosti Google, Inc. Aplikace je publikována v testovacím kanálu a není tedy veřejná. Publikováním aplikace do obchodu Play Store byl završen kompletní vývojový cyklus.

Literatura

[1] ISO/IEC 13818-1:2018: *Coding of audio, picture, multimedia and hypermedia information.* 6. Geneva: International Organization for Standardization, 2018.

[2] GOOGLE TECHNOLOGY HOLDINGS LLC; *Patent Issued for MP4 Container File Formats and Methods of Processing MP4 Container Files (USPTO 9185468).* Journal of Engineering [online]. Atlanta: NewsRx, 2015, , 4968 [cit. 01. 10. 2018]. ISSN 1945-8711. Dostupné z URL: <http://search.proquest.com/docview/1734244185/?pq-origsite=primo>

[3] ČÍKA, Petr. *Multimediální služby* [online]. Brno: Vysoké učení technické v Brně, 2012 [cit. 15. 10. 2018]. ISBN 978-80-214-4443-0. Dostupné z URL: <https://www.utko.feec.vutbr.cz>

[4] MAMMI, E., G. RUSSO a P. TALONE. *Television over IP overview. In: Visual Information Processing (EUVIP), 2010 2nd European Workshop on* [online]. IEEE Publishing, 2010, s. 119-124 [cit. 01. 10. 2018]. DOI: 10.1109/EUVIP.2010.5699130. ISBN 978-1-4244-7288-8.

[5] *HTTP Live Streaming Overview. Apple Developer* [online]. Cupertino: Apple, 2016, 01.03.2016 [cit. 15. 10. 2018]. Dostupné z URL: <https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html>

[6] TIMMERER, Christian. *Special Issue on Modern Media Transport—Dynamic Adaptive Streaming over HTTP (DASH).* Signal Processing: Image Communication [online]. Elsevier B.V, 2012, 27(4), 269-270 [cit. 01. 10. 2018]. DOI: 10.1016/j.image.2012.03.006. ISSN 0923-5965. Dostupné z URL: <https://www-sciencedirect-com.ezproxy.lib.vutbr.cz/science/article/pii/S0923596512000653>

[7] *MPEG 98 - Geneva.* The Moving Picture Experts Group [online]. Villar Dora: Dr. Leonardo Chiariglione, 2011 [cit. 15. 10. 2018]. Dostupné z URL: <https://mpeg.chiariglione.org/meetings/98>

[8] *Media presentation description and segment formats:* MPEG Dynamic Adaptive Streaming over HTTP (DASH). MPEG Dynamic Adaptive Streaming over HTTP (DASH) [online]. Villar Dora: Dr. Leonardo Chiariglione, 2011 [cit. 15. 10. 2018]. Dostupné z URL: <https://mpeg.chiariglione.org/standards/mpeg-dash/media-presentation-description-and-segment-formats>

[9] OZER, Jan. *DRM.* Streaming Media Magazine [online]. Medford: Information Today, 2017, , 122-130 [cit. 01. 10. 2018]. ISSN 15598039. Dostupné z URL: <http://search.proquest.com/docview/1891321025/>

[10] ZHAOFENG MA. *Digital rights management: Model, technology and application.* Communications, China [online]. USA: China Communications Magazine Co., 2017, 14(6), 156-167 [cit. 01. 10. 2018]. DOI: 10.1109/CC.2017.7961371. ISSN 1673-5447.

[11] *Getting Started with Widevine DRM. In:* Https://www.widevine.com/ [online]. Mountain View, Silicon Valley: Google, 2018 [cit. 23. 10. 2018]. Dostupné z URL: <https://storage.googleapis.com/wvdocs/Widevine_DRM_Getting_Started.pdf>

[12] *Secure audio/video content against unauthorized use, and help monetize content.* Microsoft PlayReady [online]. Redmond: Microsoft Corporation, 2018 [cit. 23. 10. 2018]. Dostupné z URL: <https://docs.microsoft.com/en-us/playready/>

[13] *Apple Music will now let you store your music library DRM-free.* IMore [online]. Florida: Mobile Nations, 2016 [cit. 23. 10. 2018]. Dostupné z URL: <https://www.imore.com/apple-rolling-out-improved-itunes-match-apple-music-subscribers>

[14] *FairPlay Streaming.* Apple Developer: HTTP Live Streaming [online]. Cupertino: Apple, 2018 [cit. 23. 10. 2018]. Dostupné z URL: <https://developer.apple.com/streaming/fps/>

[15] *MPEG-2 Stream Encryption Format for HTTP Live Streaming.* Apple Developer: HTTP Live Streaming [online]. Cupertino: Apple, 2018 [cit. 06. 11. 2018]. Dostupné z URL: <https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/HLS_Sample_Encryption/Intro/Intro.html#//apple_ref/doc/uid/TP40012862-CH5-SW1>

[16] *Example Playlists for HTTP Live Streaming.* Apple Developer: HTTP Live Streaming [online]. Cupertino: Apple, 2018 [cit. 06. 11. 2018]. Dostupné z URL: <https://developer.apple.com/documentation/http_live_streaming/example_playlists_for_http_live_streaming>

[17] *Nápověda Analytics* [online]. Mountain View: Google, 2018 [cit. 06. 11. 2018]. Dostupné z URL: <https://support.google.com/analytics/>

[18] CUTRONI, Justin. *Google Analytics.* O'Reilly, 2010. ISBN 0596158009.

[19] *Video AD Serving Template (VAST 3.0).* 3.0 [online]. New York, 2012 [cit. 06. 11. 2018]. Dostupné z URL: <https://www.iab.com/wp-content/uploads/2015/06/VASTv3_0.pdf>

[20] *Video AD Serving Template (VAST 4.0).* 4.0 [online]. New York, 2016 [cit. 06. 11. 2018]. Dostupné z URL: <https://www.iab.com/wp-content/uploads/2016/04/VAST4.0_Updated_April_2016.pdf>

[21] *Video AD Serving Template (VAST 4.1).* 4.1 [online]. New York, 2018 [cit. 06. 11. 2018]. Dostupné z URL: <https://iabtechlab.com/wp-content/uploads/2018/11/VAST4.1-final-Nov-8-2018.pdf>

[22] *Video Player - AD Interface Definition (VPAID).* 2.0 [online]. New York, 2012 [cit. 28. 11. 2018]. Dostupné z URL: <https://www.iab.com/wp-content/uploads/2015/06/VPAID_2_0_Final_04-10-2012.pdf>

[23] *Video Multiple AD Playlist (VMAP).* 2.0 [online]. New York, 2014 [cit. 28. 11. 2018]. Dostupné z URL: <https://www.iab.com/wp-content/uploads/2015/06/VMAP.pdf>

[24] WANG, Hai, Fei HAO, Chunsheng ZHU, Joel J. P. C. RODRIGUES a Laurence T. YANG. *An Android Multimedia Framework Based on Gstreamer.* Green Communications and Networking [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 2012, , 51-62 [cit. 22. 10. 2018]. DOI: 10.1007/978-3-642-33368-2_5. ISBN 978-3-642-33367-5. Dostupné z URL: <http://link.springer.com/10.1007/978-3-642-33368-2_5>

[25] *Media: Architecture.* Android Source [online]. Mountain View: Google, 2018 [cit. 22. 11. 2018]. Dostupné z URL: <https://source.android.com/devices/media>

[26] *ExoPlayer.* Android Developer Documentation [online]. Mountain View: Google, 2017 [cit. 22. 11. 2018]. Dostupné z URL: <https://developer.android.com/guide/topics/media/exoplayer>

[27] *ExoPlayer: LICENSE.* GitHub [online]. Mountain View: Google, 2014 [cit. 22. 11. 2018]. Dostupné z URL: <https://github.com/google/ExoPlayer/blob/release-v2/LICENSE>

[28] *Supported formats.* ExoPlayer [online]. Mountain View: Google, 2017 [cit. 26. 11. 2018]. Dostupné z URL: <https://google.github.io/ExoPlayer/supported-formats.html>

[29] *Supported devices.* ExoPlayer [online]. Mountain View: Google, 2017 [cit. 26. 11. 2018]. Dostupné z URL: <https://google.github.io/ExoPlayer/supported-devices.html>

[30] PHILLIPS, Bill, Chris STEWART a Kristin MARSICANO. *Android programming: the Big Nerd Ranch guide. Third edition.* Atlanta, Georgia: Big Nerd Ranch, 2017. ISBN 978-0-13470-605-4.

[31] *Interface ExoPlayer.* ExoPlayer library documentation [online]. Mountain View: Google, 2018 [cit. 05. 12. 2018]. Dostupné z URL: <http://google.github.io/ExoPlayer/doc/reference/com/google/android/exoplayer2/ExoPlayer.html>

[32] *Publikování aplikace.* Nápověda Play Console [online]. Mountain View: Google, 2019 [cit. 27. 03. 2019]. Dostupné z URL: <https://support.google.com/googleplay/android-developer/answer/6334282?hl=cs>

[33] *Centrum zásad pro vývojáře.* Nápověda Google Play [online]. Mountain View: Google, 2019 [cit. 27. 03. 2019]. Dostupné z URL: <https://play.google.com/intl/cs/about/developer-content-policy/index.html#!?modal_active=none>

[34] *Android App Bundle.* Android Developers [online]. Mountain View: Google, 2019 [cit. 30. 04. 2019]. Dostupné z URL: <https://developer.android.com/platform/technology/app-bundle>

[35] *AndroidX Overview.* Android Developers [online]. Mountain View: Google, 2019 [cit. 30. 04. 2019]. Dostupné z URL: <https://developer.android.com/jetpack/androidx>

[36] *Activity.* Android Developers [online]. Mountain View: Google, 2019 [cit. 01. 05. 2019]. Dostupné z URL: <https://developer.android.com/reference/android/app/Activity>

[37] *Intent.* Android Developers [online]. Mountain View: Google, 2019 [cit. 01. 05. 2019]. Dostupné z URL: <https://developer.android.com/reference/android/content/Intent>

[38] *Developer guide.* ExoPlayer [online]. Mountain View: Google, 2019 [cit. 02. 05. 2019]. Dostupné z URL: <https://exoplayer.dev/guide.html>

[39] *Connect to the network.* Android Developers [online]. Mountain View: Google, 2019 [cit. 04. 05. 2019]. Dostupné z URL: <https://developer.android.com/training/basics/network-ops/connecting>

[40] *ISO 8601.* Date and time – Representations for information interchange. 2. Geneva: International Organization for Standardization, 2019.

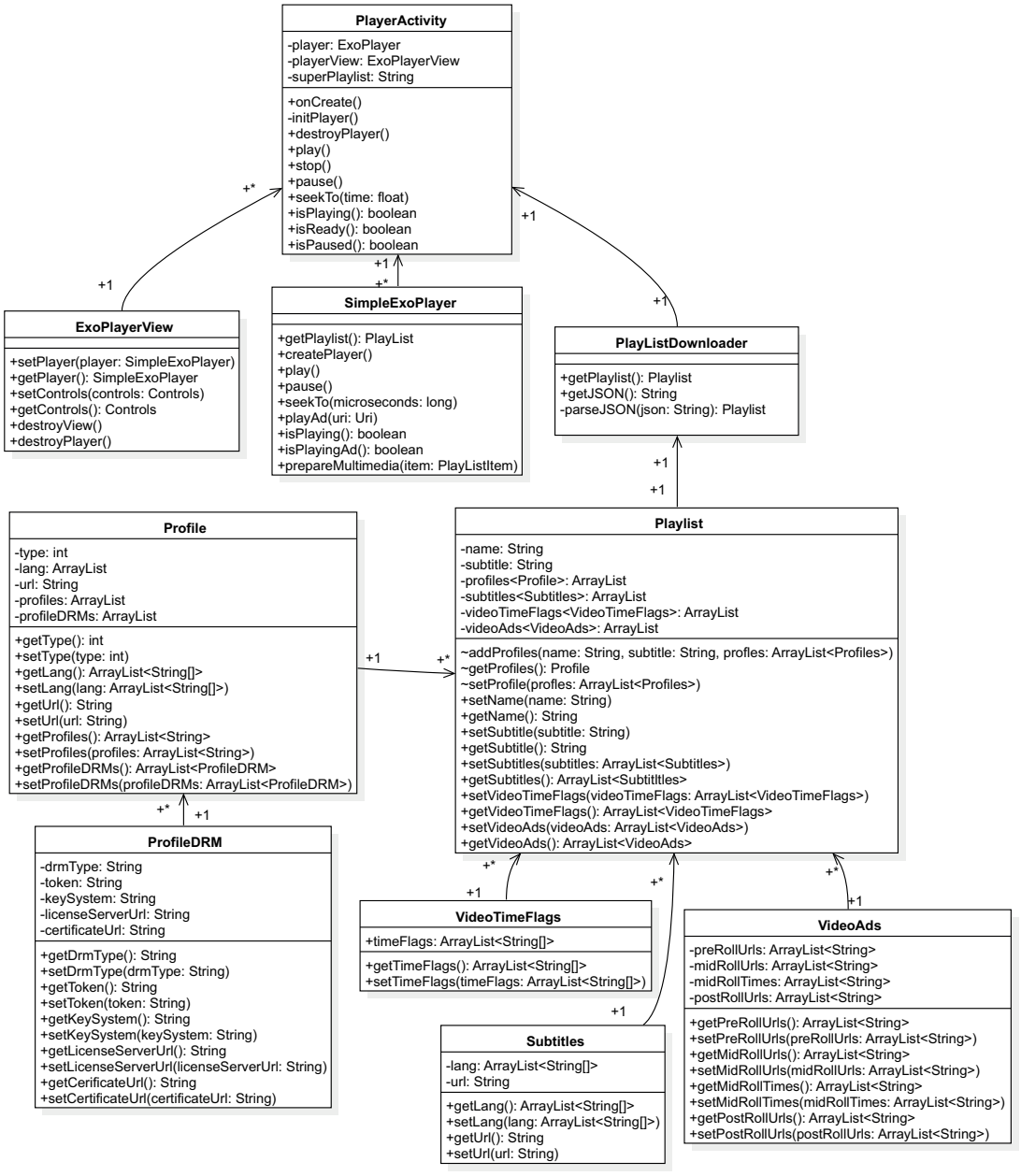
Seznam symbolů, veličin a zkratek

AAC	Pokročilé kódování zvuku – Advanced Audio Coding
ADTS	Přehrávání zvukových dat – Audio Data Transport Stream
AES	Standard pokročilého šifrování – Advanced Encryption Standard
API	Aplikační programové rozhraní – Application Programming Interface
ATSC	Pokročilé normy televizních systémů – Advanced Television Systems Committee standards
AVI	Prokládané audio video – Audio Video Interleave
CBC	Řetězení šifrových bloků – Cipher Block Chaining
CDM	Správa dekódování obsahu – Content Decoding Management
CDN	Síť pro doručování obsahu – Content Delivery Network
CMAF	Formát aplikace společného média – Common Media Application Format
CRT	Čítač – Counter
DECE LLC	Ekosystém digitálního obsahu zábavy – Digital Entertainment Content Ecosystem
DRM	Správa digitálních práv – Digital Rights Management
DVB	Digitální televizní vysílání – Digital Video Broadcasting
EPG	Elektronický programový průvodce – Electronic Program Guide
FLV	Flash video – Flash Video
FMP4	Fragmentované MP4 – Fragmented MP4
HLS	Vysílání v reálném čase pomocí protokolu HTTP – HTTP Live Streaming
HLS-AES	Zabezpečení HLS pomocí AES
HTTP	Hypertextový transportní protokol – Hypertext Transport Protocol
IAB	Interactive Advertising Bureau
ID	Identifikace – Identification
IMA	Interaktivní inzerce médií – Interactive Media Advertising
IP	Internetový protokol – Internet Protocol
IPTV	Televize přes internetový protokol – Internet Protocol television
JSON	JavaScriptový objektový zápis – JavaScript Object Notation
M4A	Média (MPEG-4) Audio – Media (MPEG-4) Audio
M4V	Média (MPEG-4) Video – Media (MPEG-4) Video
MKV	Multimediální kontejner matroška – Matroska Multimedia Container
MP3	MPEG-1 nebo MPEG-2 Audio vrstva III – MPEG-1 or

	MPEG-2 Audio Layer III
MP4	MPEG-4 část 14 – MPEG-4 Part 14
MPEG	Expertní skupina pohyblivého obrazu – Moving Picture Experts Group
MPEG-4	MPEG část 4 - MPEG 4
MPEG-CENC	Společné šifrování MPEG – MPEG Common Encryption
MPEG–DASH	Dynamické adaptivní vysílání pomocí protokolu HTTP – Dynamic Adaptive Streaming over HTTP
MPEG-TS	MPEG transportní proud – MPEG transport stream
MTS	viz. MPEG-TS
OEM	Původní výrobce dále prodávaného produktu – Original Equipment Manufacturer
OGG	Protokolování – Ogging
OMAP	Protokol o autorizaci multimédií online – Online Multimedia Authorization Protocol
PSIP	Programové a systémově informační protokoly – Program and System Information Protocol
QT	Formát QuickTime – QuickTime File Format
RM	Formát RealMedia – RealMedia File Format
SDK	Systémový vývojový nástroj – Software development kit
TCP	Transmission Control Protocol
TS	viz. MPEG-TS
UML	Unifikovaný modelovací jazyk – Unified Modeling Language
VAST	Šablona pro zobrazování digitálních videoreklam – Digital Video Ad Serving Template
VMAP	Playlist s video reklamou – Video Multiple Ad Playlist
VPAID	Definice rozhraní zobrazování reklam pro přehrávač videa – Video Player Ad-Serving Interface Definition
WAV	Windows Audio Video – Windows Audio Video
WMV	Windows Média Video – Windows Media Video
XML	Rozšiřitelný značkovací jazyk – Extensible Markup Language

Seznam příloh

A Návrh přehrávače v UML



Obr. A.1: UML návrh přehrávače.

B Soubor s informacemi o médiu ve formátu JSON

Výpis B.1: Ukázka souboru s informacemi o médiu

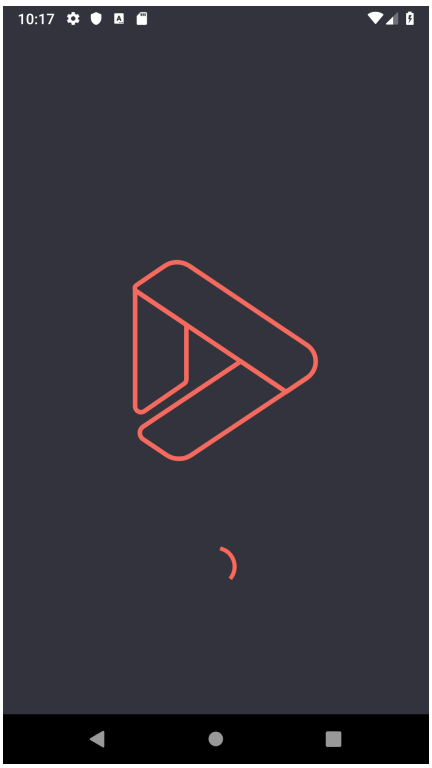
```
{ "title": "Medium name",
  "subtitle": "Media description",
  "streamInfos": [
      {"type":"MP4", "lang":{
         "key":"en","title":"English"},
         "url":"https://example.org/video.mp4",
         "profiles":["ao"]},
      {"type":"HLS", "lang":{"key":"fi",
         "title":"Suomi"},
         "url":"https://example.org/video.m3u",
         "profiles":["sd1","sd2","sd3"],
        "drmInfo": [{
           "drmType":"FAIRPLAY","drmFairPlayInfo":{
                    "token":"abcdefghijklmnopqrtuvwxz",
                    "keySystem":"com.apple.fps.1_0",
                    "licenseServerUrl":
                    "https://example.org/AcquireLicense",
                    "certificateUrl":
                    "https://example.org/fairplay.cer"}}]
      }],
  "subInfos":[{"lang": {"key": "en", "title": "English"},
               "url": "https://example.org/sub_en.vtt"}],
  "videoTimeFlags": [{"time":"PT0S", "text":"Quater"}],
  "adsEnabled": true,
  "preRollUrls": ["https://example.org/vasts/sop.xml",
                  "https://example.org/vasts/mop.xml"],
  "postRollUrls": ["https://example.org/vasts/mop.xml",
                   "https://example.org/vasts/sop.xml"],
  "midRollPosToUrls": {
    "PT4M30S": ["https://example.org/vasts/sop.xml",
                "https://example.org/vasts/mop.xml"],
    "PT7M30S": ["https://example.org/vasts/sop.xml",
                "https://example.org/vasts/mop.xml"]}
}
```

C Hlavní soubor s informacemi o dostupných médiích ve formátu JSON

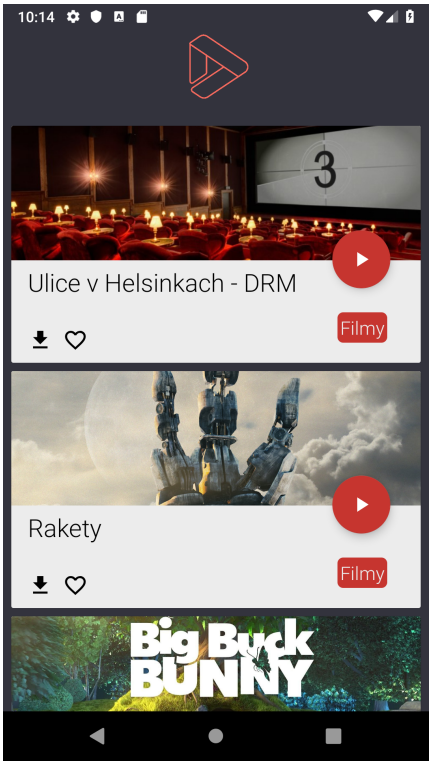
Výpis C.1: Ukázka hlavního souboru s informacemi o dostupných médiích

```
{ "categories": [
    { "id": "001",
      "name": [
        {"lang": "en", "text": "Movies"},
        {"lang": "cs", "text": "Filmy"}],
      "product": [
        { "id": "001",
          "name": [
            {"lang": "en", "text": "Tears of Steel"},
            {"lang": "cs", "text": "Rakety"}
          ],
          "pictures": {"1920":
             "https://example.org/pic001/1920x1080.jpeg"},
          "playlist":
             "https://example.org/products/id-001/play",
          "conf": {
            "seekButtons": false,
            "preferredLanguageVideo": "dm",
            "preferredLanguageSubtitles": "cs",
            "startTime": "PT5M"}}]
    },
    { "id": "002",
      "name": [
        {"lang": "en", "text": "Music"},
        {"lang": "cs", "text": "Hudba"}],
      "product": [
        { "id": "999",
          "name": [
            {"lang": "en", "text": "Hootenanny"},
            {"lang": "cs", "text": "Hootenanny"}],
          "pictures": {"1920":
             "https://example.org/pic999/1920x1080.jpg"},
          "playlist":
             "https://example.org/products/id-999/play"}]
    }]
}
```

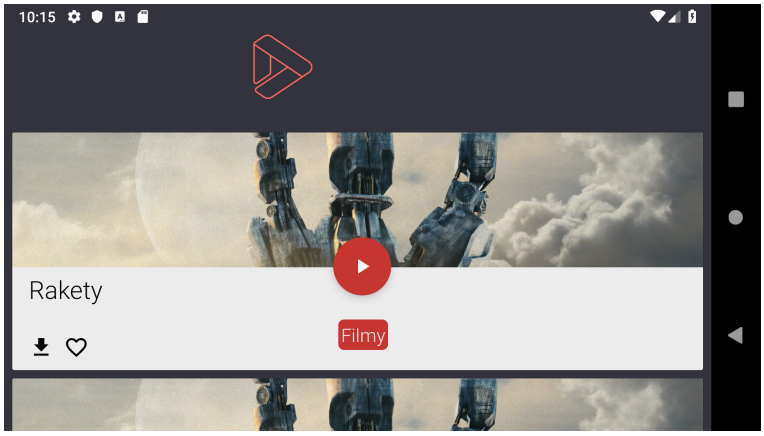
D Ukázky testovací aplikace



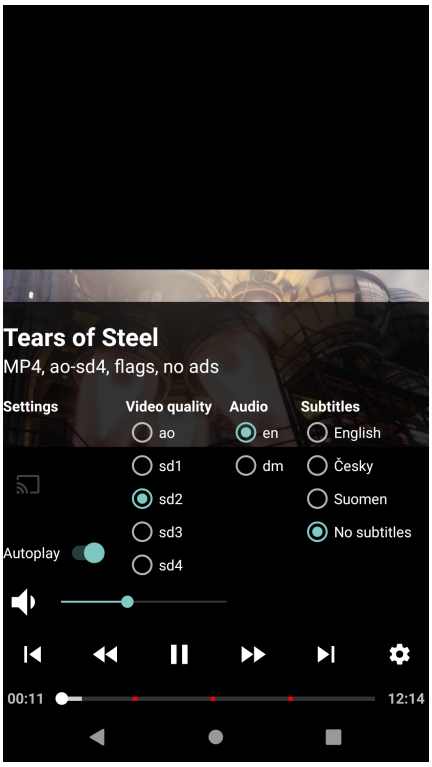
Obr. D.1: Spouštěcí obrazovka



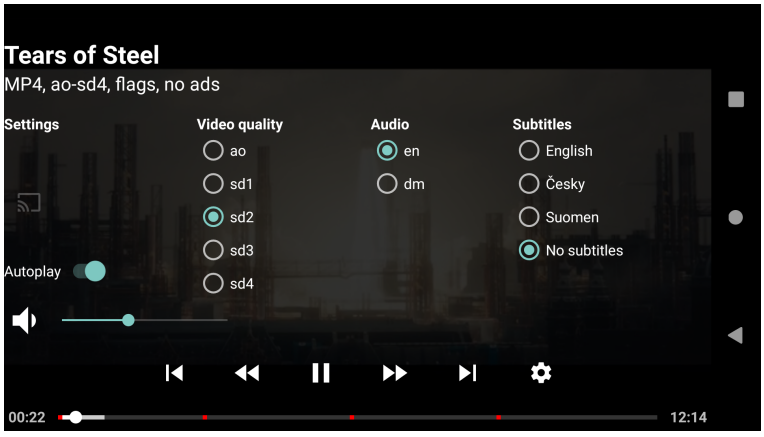
Obr. D.2: Menu testovací aplikace



Obr. D.3: Menu aplikace otočeno na široko



Obr. D.4: Kontrolní panel knihovny přehrávače



Obr. D.5: Kontrolní panel knihovny přehrávače otočeno na široko

E Obsah přiloženého CD

- `/` ..... Kořenový adresář CD
  - `Advanced_Android_Player_-_Benedikt.pdf` .... Elektronická verze dokumentace
  - `player_library` ..... Složka projektu s knihovnou přehrávače
    - `build`
      - `...`
    - `libs`
      - `ExoPlayer` ..... Zdrojové soubory frameworku ExoPlayer
      - `googleanalytics` ..... Zdrojové soubory modulu Google Analytics
    - `src`
      - `main`
      - `...`
    - `build.gradle`
    - `advancedplayer.iml`
  - `testPlayer` ..... Složka projektu s testovací aplikací
    - `advancedplayer` ..... Implementovaná knihovna přehrávače
    - `app`
      - `build`
      - `libs`
      - `src`
      - `...`
    - `build`
    - `gradle`
    - `build.gradle`
    - `settings.gradle`
  - `binaries` ..... Vygenerované balíčky testovací aplikace
    - `TestPlayer.abb` ..... Podepsaný aplikační balíček typu Bundle
    - `TestPlayer.apk` ..... Podepsaný aplikační balíček typu APK
  - `readme.txt` ..... Pokyny pro použití projektů a balíčků