TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

MP3 PŘEHRÁVAČ PRO TABLET

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Ladislav Hofman**
Vedoucí práce: Ing. Zbyněk Mader, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

MP3 PLAYER FOR TABLET

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Ladislav Hofman**
Supervisor: Ing. Zbyněk Mader, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2013/2014

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ladislav Hofman**
Osobní číslo:	**M10000150**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**MP3 přehrávač pro tablet**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s vhodným vývojovým prostředím s možností vývoje aplikací pro operační systém Android 4.0.
2. Vytvořte aplikaci mp3 přehrávače určenou pro tablet, aplikace používá funkce "drag-and-drop" pro tvorbu aktuálního playlistu mp3 souborů ze složek a změnu pořadí přehrávaných mp3 souborů.
3. Mp3 přehrávač kromě běžných funkcí (play, stop, pause atd.) umožní grafické zobrazení odehraného času přehrávané písně a její předčasné ukončení uživatelem.
4. Pro aplikaci vytvořte vhodné GUI, které splní požadavky na přívětivé uživatelské ovládání pro profesi Diskžokej (DJ).

Rozsah grafických prací:	**Dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 30 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Grant Allen, Android 4 - Průvodce programováním mobilních aplikací, Computer Press, ISBN 978-80-251-3782-6**

[2] **Reto Meier, Professional Android 4 Application Development, 2012, ISBN-10: 1118102274**

Vedoucí bakalářské práce:	**Ing. Zbyněk Mader, Ph.D.** Ústav informačních technologií a elektroniky
Datum zadání bakalářské práce:	**12. září 2013**
Termín odevzdání bakalářské práce:	**16. května 2014**

V. Kopecký

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

Technická univerzita v Liberci - 14 -

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Děkuji svému vedoucímu práce, Ing. Zbyňku Maderovi, Ph.D., za výbornou komunikaci, Miloši Mackovi za doporučení použít fragmenty a drobné rady, Oldřichu Tauferovi za drobné rady a rodině za podporu.

Abstrakt

Práce pojednává o možnostech vývoje aplikací pro operační systém Android. Cílem je vytvořit uživatelsky přívětivou aplikaci hudebního přehrávače určenou pro tablety. Aplikace bude kromě běžných funkcí přehrávačů umožňovat jednoduchou tvorbu playlistu. Ta bude realizovaná pomocí funkce drag&drop, která bude fungovat nejen mezi souborovým systémem a playlistem, ale i v rámci samotného playlistu, kde bude sloužit pro změnu pořadí jednotlivých skladeb.

Při tvorbě aplikace byly využity dostupné prostředky k vývoji aplikací pro platformu Android a vývojové prostředí Android Studio. Největší prostor je v práci věnován způsobu, jakým bylo dosaženo funkčnosti jednotlivých komponent přehrávače.

Vytvořil jsem funkční aplikaci a oproti zadání ji rozšířil o možnost uložení a opětovného načtení playlistu. Věřím, že moje práce bude přínosná nejen pro zadávajícího, ale i pro běžné uživatele.

Klíčová slova:

Google Android, Hudební přehrávač, Drag&Drop, Tablet, mp3

Abstract

The thesis deals with the development of Android applications. The aim is to create a user friendly music player application for tablets. Besides other common functions, the application allows the user to create simple playlists, which is realized using the drag & drop function. Moreover, this function will work not only between the file system and playlist, but also in the playlist itself in order to allow changing the order of single tracks.

To create the application I used the available means and resources for the Android platform and Android Studio development environment. In particular, the thesis describes the way in which were the single functional components produced.

I succeeded in creating a working and functional application which allows the user also to save and load his or her own playlist. I believe that the thesis will be useful for the consultant as well as for common users.

Keywords:

Google Android, music player, Drag&Drop, Tablet, mp3

Obsah

Seznam obrázků

1 Úvod

Když jsem hledal vhodné zadání pro svoji bakalářskou práci, název práce MP3 přehrávač pro tablet mě okamžitě zaujal. Jsem totiž velký nadšenec pro mobilní technologie a vývojem aplikací pro operační systém Android bych se rád zabýval i v profesním životě.

Android je operační systém cílený na mobilní zařízení. Nejčastěji jej můžeme naleznout na mobilních telefonech a tabletech, které rok od roku zvyšují svůj podíl na trhu.

Domluvil jsem si tedy s vedoucím práce schůzku, na které jsem se dozvěděl jeho představu o výsledné podobě přehrávače a na schůzce vznikl i prvotní nákres rozhraní. Šlo o to přiblížit ovládání známé z rozhraní přehrávačů na osobním počítači do rozhraní tabletu. Na počítači je například možné táhnutím myší po obrazovce přenést vybraný mp3 soubor ze složky do fronty přehrávače. Toto se projevilo i v nákresu aplikace. Obrazovka tabletu bude rozdělena na poloviny, kde na jedné straně bude fronta přehrávače a na straně druhé souborový systém. Pod tímto pak budou následovat klasické ovládací prvky přehrávače.

Funkce drag&drop by měla fungovat tak, že uživatel jednoduše najde v souborovém systému nějakou skladbu, mp3 soubor, a jednoduše jen přetáhne prstem po obrazovce na libovolné místo ve frontě přehrávaných skladeb. Ve frontě skladeb by pak drag&drop umožňovala měnit pořadí přehrávaných skladeb.

Mezi klasické ovládací prvky přehrávače řadím tlačítka umožňující zapnout vybranou skladbu, pozastavit ji, přepnou na následující skladbu ve frontě, případně na předcházející skladbu. Dále pak posuvník signalizující průběh skladby a odehraný čas.

Mezi nezvyklé ovládací prvky aplikace bych zařadil funkci pro předčasné ukončení skladby. Ta by měla být reprezentována dalším posuvníkem. Ten by fungoval tak, že pokud by jej uživatel nastavil například na polovinu, tak by se přehrávání dané skladby v polovině jejího času ukončilo.

Vzhledem k tomu, že mě zaujal originální koncept přehrávače prezentovaný vedoucím práce, rozhodl jsem se toto zadání bakalářské práce vypracovat. Zakoupil jsem si tablet Google Nexus 7 model 2013 a otestoval jsem zhruba 2 desítky nejpoužívanějších hudebních přehrávačů na trhu. K mému překvapení se ani jeden z nich nepřibližoval zadanému konceptu.

Zakoupil jsem si knížku Android 4 - průvodce programováním mobilních aplikací od autora jménem Grant Allen a začal se pomalu blížit k cíli své práce.

2 Podrobnější představení problematiky

2.1 Systém Google Android

Android je open source platforma šířená pod licencí *Apache 2.0* vyvíjená konsorciem *Open HandsetAlliance (OHA)*. *OHA* sdružuje technologické firmy od výrobců mobilních telefonů a firem podnikajících v telekomunikacích až po Google. Cílem *OHA* je vytvořit kvalitní mobilní platformu, která by byla k dispozici bez licenčních poplatků.

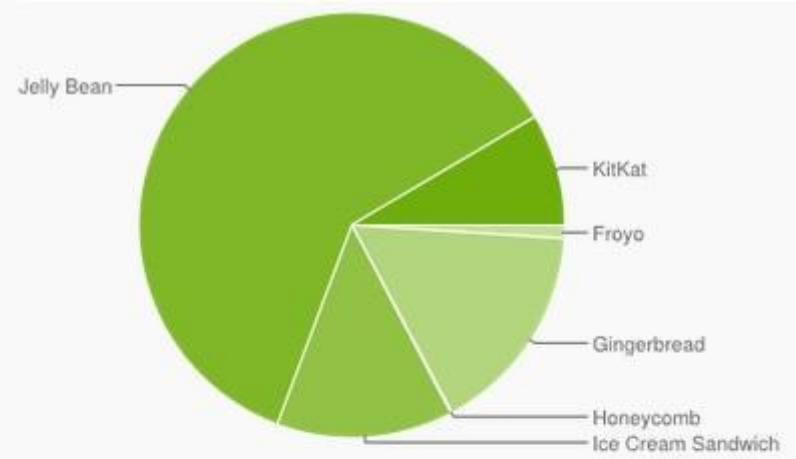
Platforma Android je založena na Linuxu, přičemž využívá jeho jádra pro správu paměti a procesů.

Aplikace určené pro běh na Androidu jsou nejčastěji programovány v jazyku Java. Nicméně existují vývojová prostředí, ve kterých se používá C# (například projekt *Xamarin* určený pro hromadný vývoj pro všechny mobilní platformy).

Mezi nejčastější zařízení, na kterých můžeme nalézt systém Android, patří v dnešní době mobilní telefony a tablety. Android je však připraven i k běhu na chytrých televizích, případně na chytrých hodinkách, notebooku apod.

2.2 Zastoupení verzí Androidu na trhu

Systém Android se postupně vyvíjí a za svoji historii prošel několika změnami. Každou další verzí přicházejí nové funkce pro uživatele i pro programátory. Systém tak není úplně zpětně kompatibilní. Nicméně pokud programátor napíše aplikaci například na verzi 2.3.3, tak je možné tuto aplikaci spustit i na zařízení se systémem Android ve verzi 4.4.



Version	Codename	API	Distribution
2.2	Froyo	8	1.0%
2.3.3 - 2.3.7	Gingerbread	10	16.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	13.4%
4.1.x	Jelly Bean	16	33.5%
4.2.x		17	18.8%
4.3		18	8.5%
4.4	KitKat	19	8.5%

Obrázek 1: Podíl verzí Androidu na trhu

Z obrázku je patrné, že více než 80 % zařízení využívá verzi 4.0.3 a vyšší. Z tohoto důvodu jsem se rozhodl využívat API verze 15 a starší zařízení nepodporovat.

2.3 Dalvik VM

Vzhledem k omezenému výkonu mobilních zařízení není v OS Android implementován *Java VirtualMachine*, ale *DalvikVM*, který byl inženýry z Google vyvinut kvůli potřebě kvalitní optimalizace výkonu. *Dalvik VM* je tedy vlastní virtuální stroj vlastní OS Android.

Vygenerované *.class* soubory jsou pomocí *Dalvik VM* konvertovány do *.dex(DalvikEXecutable)* a *.odex (OptimizedDalvikEXecutable)* před samotnou instalací v zařízení. V těchto souborech se *Dalvik VM* snaží o co největší efektivitu například vynecháváním duplicit kódu.

Android OS verze 4.4 představil experimentální virtuální stroj *ART*, který využívá *ahead-of-time (AOT) compiler*. V tomto případě je při instalaci bytecodepředkompilován do strojového jazyka daného zařízení. Mezi výhody patří rychlejší běh aplikací. Mezi nevýhody potom pomalejší instalace aplikace a její větší nároky na místo v paměti.

2.4 Vývojové prostředí Android Studio

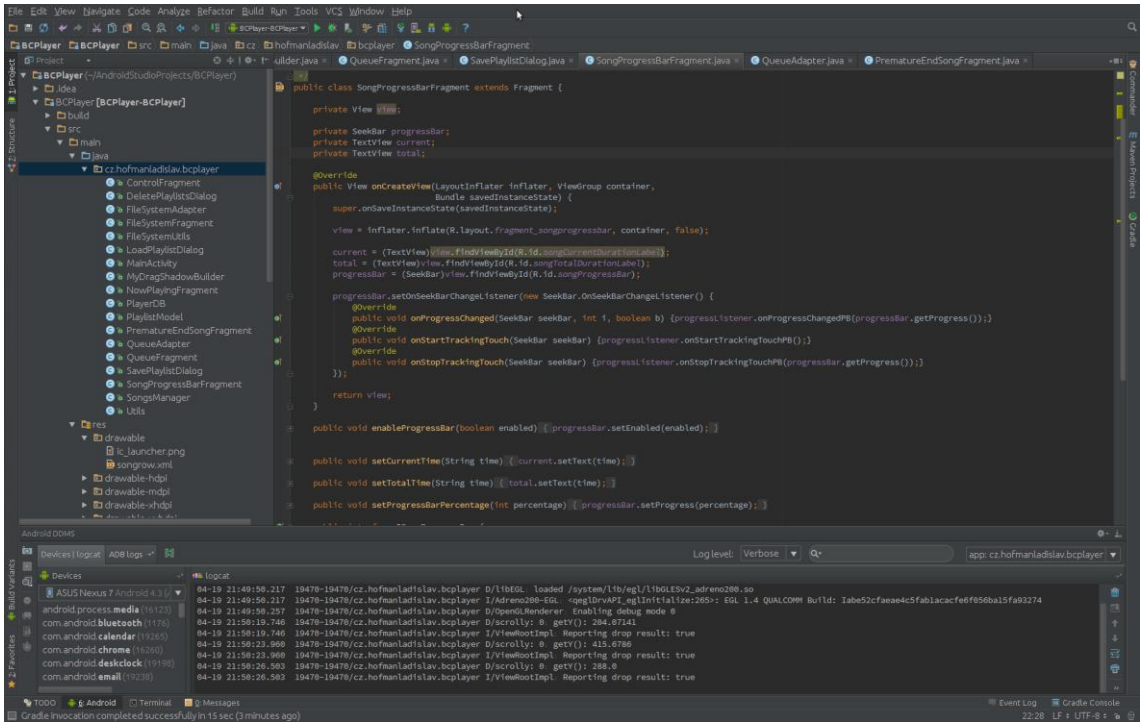
Mezi vývojová prostředí podporovaná společností Google v současné době patří Eclipse s pluginem*ADT (Android DevelopmentTools)* a Android Studio postavené nad *Community* verzí prostředí *JetBrainsIntelliJ IDEA*. Díky tomu získává všechny ty možnosti práce s kódem (navigace v kódu, našeptávání, refaktoring, analýza kódu atd.), ve kterých je *IDEA* špičkou v oboru. Přestože je většina produktů od firmy *JetBrains* placená, Android Studio je zcela zdarma. Poprvé bylo představeno veřejnosti na Google I/O konferenci 16. května 2013 a do budoucna se s ním počítá v roli hlavního vývojového prostředí pro OS Android.

Mezi hlavní výhody Android Studia patří jednoduchá instalace na všech platformách. Jednoduše stačí stáhnout si baliček pro svoji platformu (k dispozici pro Linux, Windows i Mac OS), rozbalit a spustit.

Součástí instalace je:

- samotné Android Studio IDE
- *Android SDK Tools*
- kompilátor Android
- základní emulátory s plnohodnotným systémem Android

Pro vývoj aplikace jsem si po chvíli testování zvolil Android Studio a to i přes jeho časnou verzi, kde se daly očekávat různé problémy, pády aplikace atd., a dřívější zkušenosti s Eclipse a *ADT*.



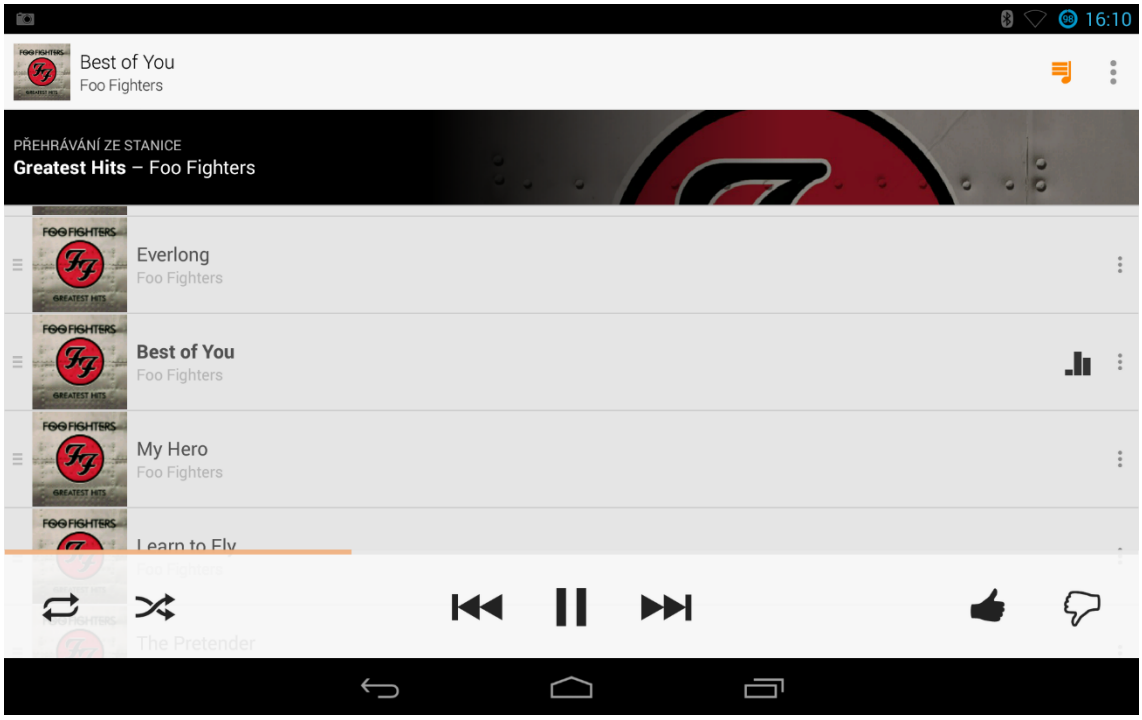
Obrázek 2: Vývojové prostředí Android Studio

Android Studio umožňuje vývoj, ladění a testování aplikací v přehledném funkčně bohatém uživatelském prostředí, na které jsem si velice rychle zvykl.

Naprogramovanou aplikaci je možno přeložit a spustit buď na fyzickém přístroji s OS Android, nebo lze využít emulátoru *AVD (Android VirtualDevice)*, který je schopen naemulovat téměř libovolné podmínky pro následný běh aplikace. Vzhledem k tomu, že vlastním tablet Nexus 7 (2013), podporovaný přímo společností Google, spouštím vyvíjené aplikace přímo na něm.

2.5 Nejpoužívanější hudební přehrávače na systému Android

2.5.1 Google Play Music

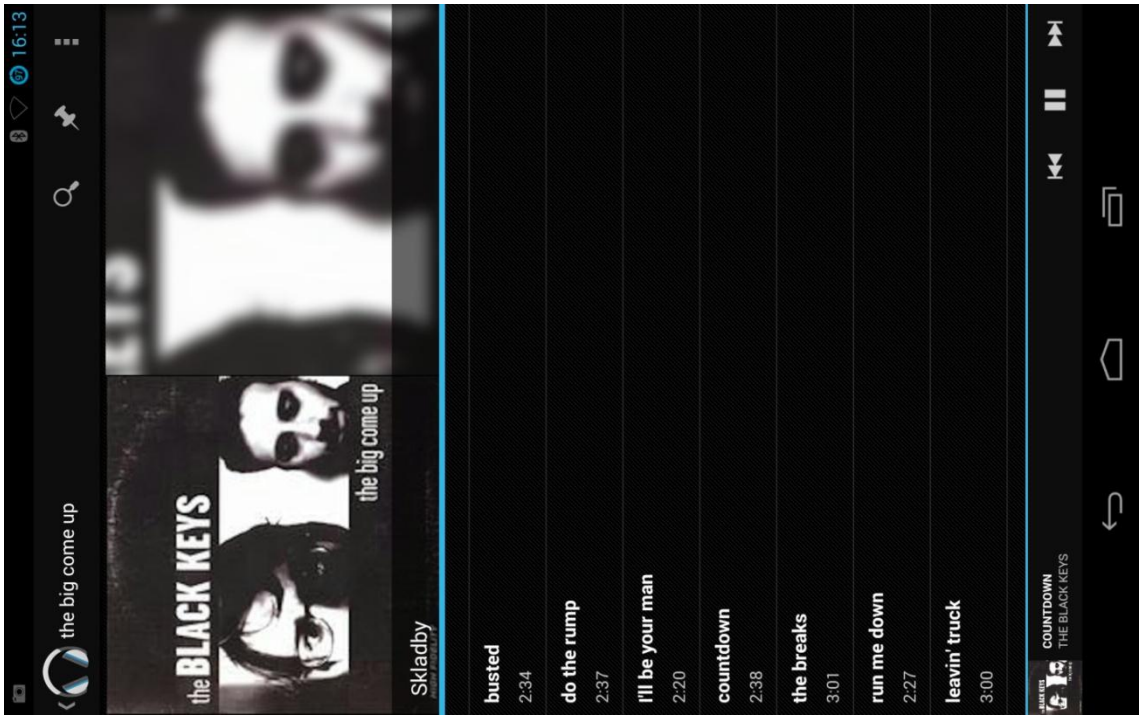


Obrázek 3: Přehrávač Google Play Music

Google Play Music je hudební služba, která nabízí streamování hudby do mobilních telefonů, tabletů a osobních počítačů. Služba nabízí 2 možnosti. Jednou je mít za měsíční paušál cca 130 korun přístup k přehrání veškeré hudby, kterou Google na svém obchodu nabízí. Druhou možností je nahrání až 20 000 hudebních souborů na svůj Google účet a mít k nim přístup zdarma odkudkoliv.

Samotný přehrávač pak nabízí pouze základní funkcionalitu tvořenou přehráváním, ukládáním seznamů skladeb a dělení skladeb dle alba. Pro účely požadované v zadání bakalářské práce je tedy tento přehrávač nevyhovující.

2.5.2 Apollo

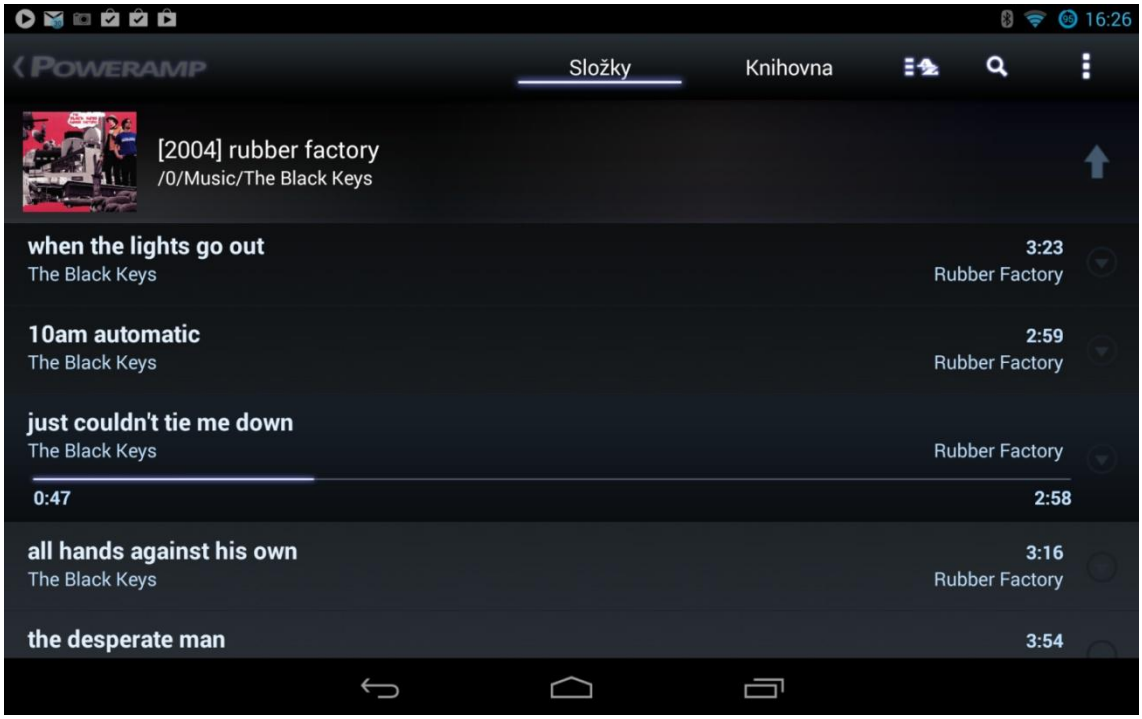


Obrázek 4: Přehrávač Apollo

Hudební přehrávač dodávaný společně s *CyanogenMod* (*Cyanogenmod* je upravený firmware pro řadu telefonů založených na otevřeném operačním systému Android. Nabízí funkce, které nejsou běžně dostupné v standardním Androidu.).

Tento přehrávač patří mezi nejpřehlednější přehrávače, které jsem testoval. Bohužel opět není uzpůsoben realitě tabletů a dá se rozumně používat pouze ve vertikálním naklonění telefonu/tabletu. Dále neumí zobrazit skladby po složkách, ale závisí na správně vyplněných informacích o skladbě (tzv. Id3 tagy). Navíc vzhledem k obecnému zvyku lidí držet tablet v horizontálním (vodorovném) režimu tak nesplňuje zadání.

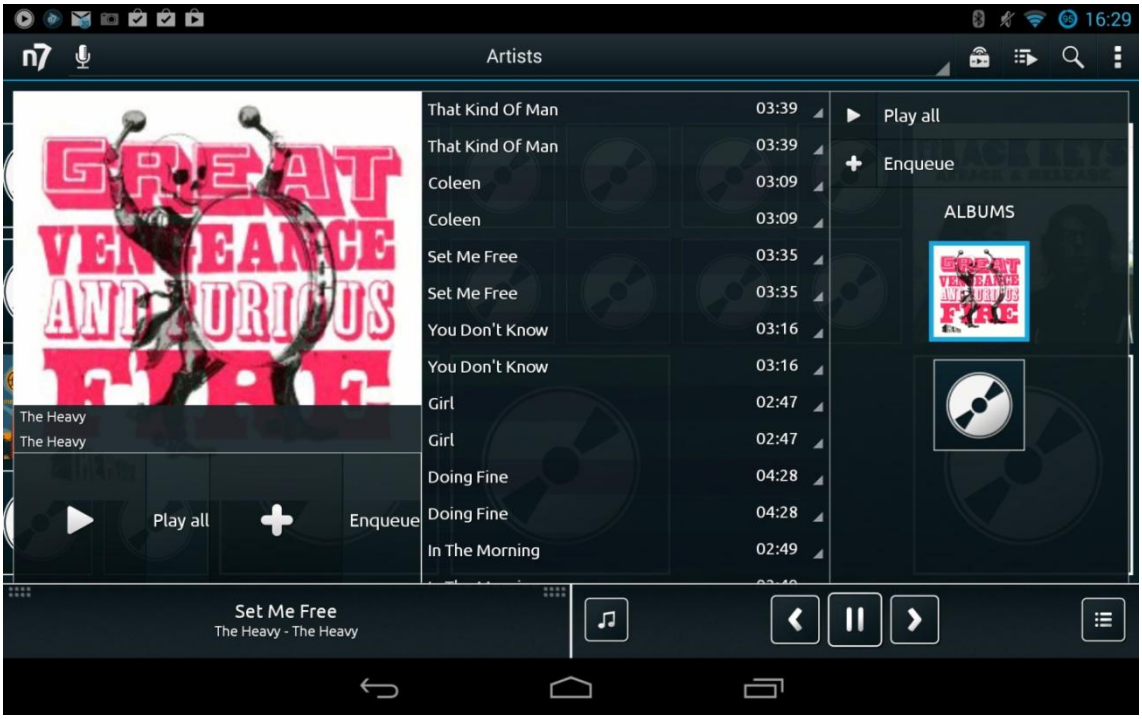
2.5.3 Poweramp Music Player



Obrázek 5: Přehrávač Poweramp Music Player

Přehrávač PowerAmp patří s více jak 10 000 000 stažení mezi nejúspěšnější aplikace. Kromě základních funkcí přehrávače nabízí například velice propracovaný ekvalizér, nebo zobrazení textu skladby. Tvůrci však nepočítali s během aplikace na větším displeji, což se na přehlednosti v případě provozu na tabletu znatelně podepsalo. Některé ovládací prvky jsou tak příliš malé a ty zbylé naopak přehnaně veliké. Pochválit by se dal systém ovládacích gest. Ani tuto aplikaci tedy nepovažuji za vyhovující zadání bakalářské práce.

2.5.4 n7player Music Player



Obrázek 6: Přehrávač n7player Music Player

Tento hudební přehrávač používám na svém mobilním telefonu a jsem s ním velice spokojen. Na tabletu si však zachoval rozměry tlačítek a nabídek z prostředí mobilního telefonu. Tlačítka jsou velice malá a ovládání vzhledem k zadání bakalářské práce zbytečně nepřehledné. To je bohužel daní za funkčně nejobsáhlejší přehrávač.

3 Cíl práce

Jedním z cílů práce je seznámit se s vhodným vývojovým prostředím a možnostmi vývoje pro systém Android 4.0. V předmětu Programování mobilních zařízení jsme využívali vývojové prostředí Eclipse s doplňkem *ADT* pro vývoj aplikací pro systém Android. Vzhledem k tomu, že se o tento obor zajímám, neuniklo mi představení vývojového prostředí Android Studio na konferenci Google I/O konané dne 16. května 2013. Jelikož jsem předem rozhodnut toto vývojové prostředí využít, je mým cílem i cílem práce naučit se s ním efektivně pracovat.

Dalším cílem je vytvořit aplikaci mp3 přehrávače určenou primárně pro tablet, která používá funkci drag&drop pro tvorbu aktuálního playlistu mp3 souborů ze složek souborového systému. Stejným způsobem poté umožnit změnu pořadí přehrávaných mp3 souborů neboli skladeb. Drag&drop je velmi častá funkce v témě každém operačním systému a umožňuje tahem myší přetáhnout nějaký objekt z jednoho místa na druhé. Tuto funkci jsem již v několika aplikacích na systému Android objevil, a proto věřím, že její využití bude i v mé práci možné.

Mp3 přehrávač by měl kromě běžných funkcí, jakými jsou *play, stop, pause* atd. umožnit i grafické zobrazení odehraného času přehrávané skladby a její předčasné ukončení uživatelem. Mezi běžné funkce si dovolím zařadit i možnost přepnutí na následující, případně předchozí, skladbu. Grafické zobrazení odehraného času je téměř výhradně v přehrávačích tvořeno obdobou posuvníku a informacemi o aktuálním a celkovém čase přehrávané skladby. Na tomto řešení ve své práci nebudu nic měnit, jelikož mi přijde osvědčené a uživatelsky přívětivé.

Možnost předčasného ukončení skladby bude po diskusi s vedoucím práce realizováno jako druhý posuvník s časovými údaji aktuální skladby. Pokud uživatel posuvníkem pohne, čas se přepočítá a jakmile aktuální skladba tohoto času ve svém přehrávání dosáhne, bude automaticky ukončena.

Posledním cílem práce, který však bude nutné realizovat již při samotném návrhu aplikace, je vytvoření vhodného GUI. Toto GUI by mělo splnit požadavky na přívětivé uživatelské ovládání pro profesi Diskžokej (DJ). Znamená to splnit určité požadavky na velikost grafických prvků, aby nedocházelo k nechtěným přehmatům apod.

4 Rozbor řešení

4.1 Widget

Všechny nástroje pro vytváření uživatelského rozhraní nabízejí nějaké základní widgety(1). Toto platí i pro systém Android, kde jich existuje relativně mnoho, a v případě potřeby se dají upravovat.

4.1.1 Popisek

Popisek je asi nejjednodušší widget dostupný uživateli a programátorovi. Je o prosté zobrazení textového řetězce. V Androidu jej reprezentuje třída *TextView*. U popisku lze měnit téměř vše od stylu písma, přes jeho barvu a velikost až po průhlednost. Veškeré definice lze zadat jak programově, tak přímo v xml definici rozložení.

Například zapsáním `android:textsize="20dp"` je nastavena velikost písma 20 dp, tedy takzvaných pixelů nezávislých na hustotě pixelů na displeji zařízení. Dále lze velikost nastavit například v pixelech, nebo v milimetrech.

4.1.2 Tlačítko a obrázkové tlačítko

Dále se v systému Android velice často používají tlačítka. Realizované konkrétně třídou *Button* a *ImageButton*, kde první zmíněné slouží k obyčejnému tlačítku s popiskem a druhé k tlačítku ve tvaru nastaveného obrázku. Tlačítku lze nastavit barvu pozadí a textu, velikost, akci, co se provede po kliknutí a spoustu dalšího.

4.1.3 Textové pole

Textové pole slouží k uživatelskému zadávání dat. V tomto případě textu. Textové pole má na starosti třída *EditText*. Specifikovat lze například možnost zadávat pouze číselné hodnoty (`android:digits`), nebo možnost nahrazování zadávaného textu tečkami pro potřebu zadání hesla (`android:password`). Možností je daleko více a obecně je doporučováno specifikovat je v definici xml.

4.1.4 Posuvník

Posuvník je další widget, který budu využívat ve své práci. Podobá se studiovému posuvnému potenciometru a v Androidu je reprezentován třídou *SeekBar*. Já jej budu využívat jako indikátor odehraného času skladby a její ovládání. Například lze programově danému *SeekBaru* nastavit pozici, neboli aktuální posun, metodou `setProgress(intpercentage)`. Dále je možné reagovat na uživatelské akce, jako je například posun na jinou procentuální pozici.

4.2 Kontejnery

Kontejnery obecně organizují kolekce vybraných widgetů, případně dalších podřízených widgetů, do různých struktur. Pokud například chceme vytvořit formulář s textovým polem a tlačítky zarovnanými k určité straně, budeme potřebovat kontejner. Kontejnej je tedy kořenový element, do kterého můžeme vkládat widgety, případně skupiny widgetů, či fragmenty. Definice rozložení prvků se provádí v xml souborech.

Nejpoužívanějším kontejnerem, který budu využívat ve své aplikaci, je *LinearLayout*. Třída *LinearLayout* řadí widgety, či vnořené kontejnery za sebe do řádku, či sloupce. Funguje tedy podobně jako třída *FlowLayout* nástroje *Java/Swing*.

Tomuto layoutu lze nastavit například orientaci (vertikální, horizontální), mezery mezi widgety, nebo například gravitaci. Gravitace určuje, jakým směrem se budou vložené prvky zarovnávat. Standardně je nastavena vlevo nahoru.

Dalšími kontejnery jsou například *RelativeLayout*, *AbsoluteLayout* nebo *TableLayout*. Způsob jejich fungování lze vyčíst již z jejich názvu.

4.3 Aktivita

Aktivity jsou základními stavebními bloky uživatelského rozhraní. Lze si je představit například jako okno internetového prohlížeče. Mezi jednotlivými aktivitami lze jednoduše přepínat, případně z nich otevírat další aktivity. Návrat do předchozí aktivity je poté možný pomocí systémového tlačítka zpět podobně jako ve webovém prohlížeči.

Aktivita se v průběhu svého života může nacházet ve třech základních stavech - může být na popředí a mít uživatelský vstup, může být pouze viditelná (třeba jen částečně) nebo může být pozastavená na pozadí. Při přechodu aktivity mezi stavy jsou volány systémová zpětná volání, které vymezují jednotlivé stavy aktivity a definují její životní cyklus.

- `onCreate()` - Volána v momentě vytvoření aktivity. Zde se standardně provádí inicializace aktivity a nastavuje obsah uživatelského rozhraní.
- `onStart()` - Zavolána, když se aktivita stane viditelnou pro uživatele. Nemusí být plně viditelná - její část může být zakryta nebo může být vidět pod poloprůhledným pozadím jiné aktivity.
- `onResume()` - Volána, když aktivita začne dostávat uživatelský vstup.
- `onPause()` - Volána těsně před ztrátou uživatelského vstupu.
- `onStop()` - Zavolána, když aktivita přestává být viditelná pro uživatele.
- `onDestroy()` - Volána před zrušením instance aktivity.
- `onRestart()` - Tato funkce je volána těsně předtím než aktivita, která byla zastavena funkcí `onStop()`, je znovu nastartována - ve funkci `onStart()`.

4.4 Fragment

Fragment je volitelná vrstva, kterou lze vložit mezi aktivity a widgety a která je navržena tak, aby pomáhala měnit konfiguraci aktivit za účelem podpory malých i velkých obrazovek. Fragmenty slouží k agregaci widgetů a kontejnerů. Poté je lze umísťovat do aktivit, přičemž někdy i několik fragmentů do jedné aktivity, jindy pouze jeden fragment do jedné aktivity.

Při použití fragmentů je každá oddělená komponenta uživatelského rozhraní umístěna do svého fragmentu. Aktivity, kterých se tato implementace týká, pak na základě velikosti obrazovky určují, která komponenta bude do fragmentu umístěna a která nikoliv. Aktivity, které se nemění, pak nemusí fragmenty používat vůbec.

Fragmenty mají svůj životní cyklus podobný aktivitám, který je ale přímo řízen aktivitou, ve které se daný fragment nachází.

4.5 ListFragment

ListFragment je jednou z podtříd třídy Fragment. Tato třída obaluje do třídy Fragment kontejner *ListView* a je navržena tak, aby mohla snadno zobrazovat seznamy. Při použití třídy stačí zavolat metodu `setListAdapter()` pro vybraný a nakonfigurovaný objekt *ListAdapter* a přepsat metodu `onListItemClick()`, která obsluhuje klepnutí do řádku seznamu.

4.6 MediaPlayer

Třída *MediaPlayer* může být v Androidu použita k přehrávání audia a videa. Dále umožňuje toto přehrávání dále kontrolovat a ovládat pomocí různých funkcí. Následující výběr funkcí používám ve své práci.

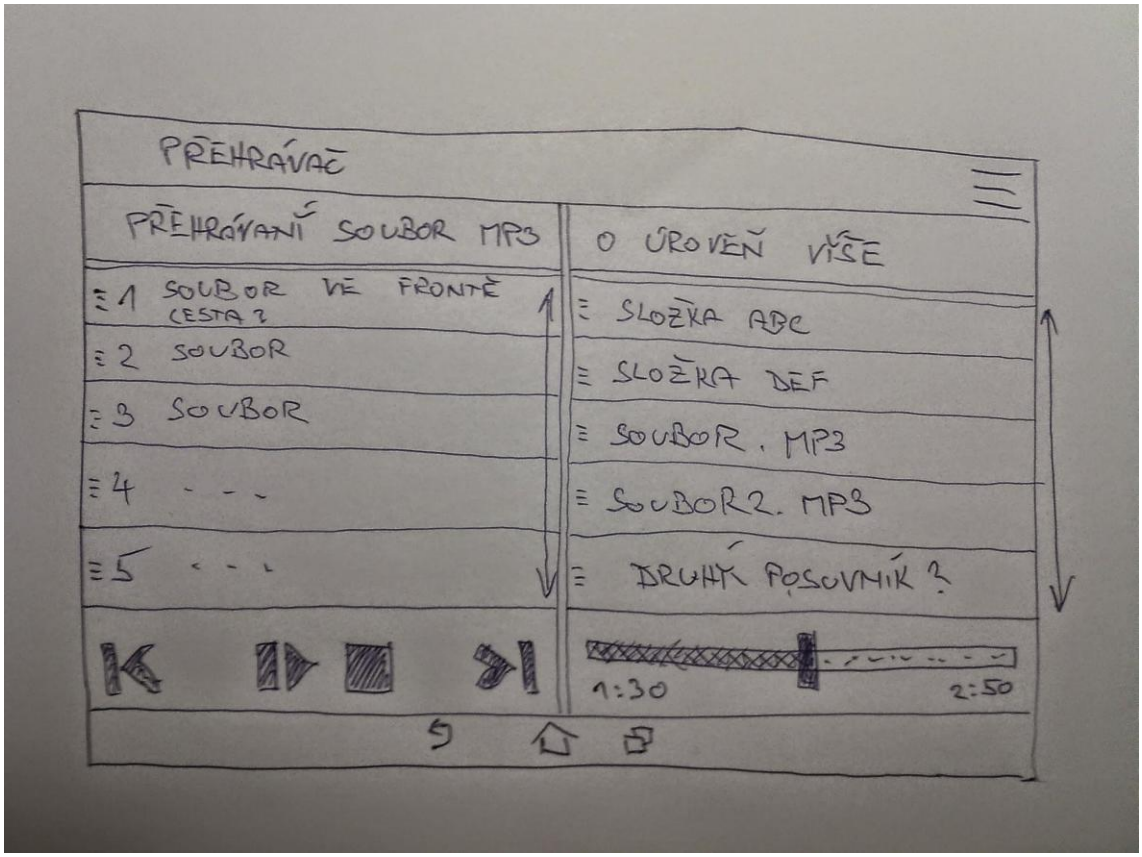
- `MediaPlayer mp = newMediaPlayer()` - Nová instance třídy *MediaPlayer* (1)
- `setDataSource("/sdcard/path_to_song")` - Nastavení zdroje dat, cesta k hudebnímu souboru
- `mp.start()` - Přehrání skladby
- `mp.pause()` - Pauza
- `mp.reset()` - Reset instance třídy *MediaPlayer*
- `mp.getDuration()` - Získání délky skladby v milisekundách
- `mp.getCurrentDuration()` - Získání aktuální pozice přehrávané skladby v milisekundách
- `mp.seekTo(positon)` - Posunutí místa přehrávání skladby na určitý čas zadaný s milisekundách
- `mp.isPlaying()` - Kontrola, zda skladba hraje, nebo ne. Vrací `true` nebo `false`
- `mp.setVolume(floatleftVolume, floatrightVolume)` - nastavení hlasitosti přehrávání

4.7 SQLite databáze

SQLite je vestavěná databáze Androidu, která obsahuje čisté rozhraní SQL a vyniká svojí rychlostí. V systému Android jej může využívat každá aplikace. Díky SQL rozhraní je práce s ní jednoduchá, jelikož je toto rozhraní běžně používáno ve všech databázích založených na jazyce SQL. Pro manipulaci s daty se tak používají příkazy typu *SELECT*, *INSERT* apod.

4.8 Grafický návrh

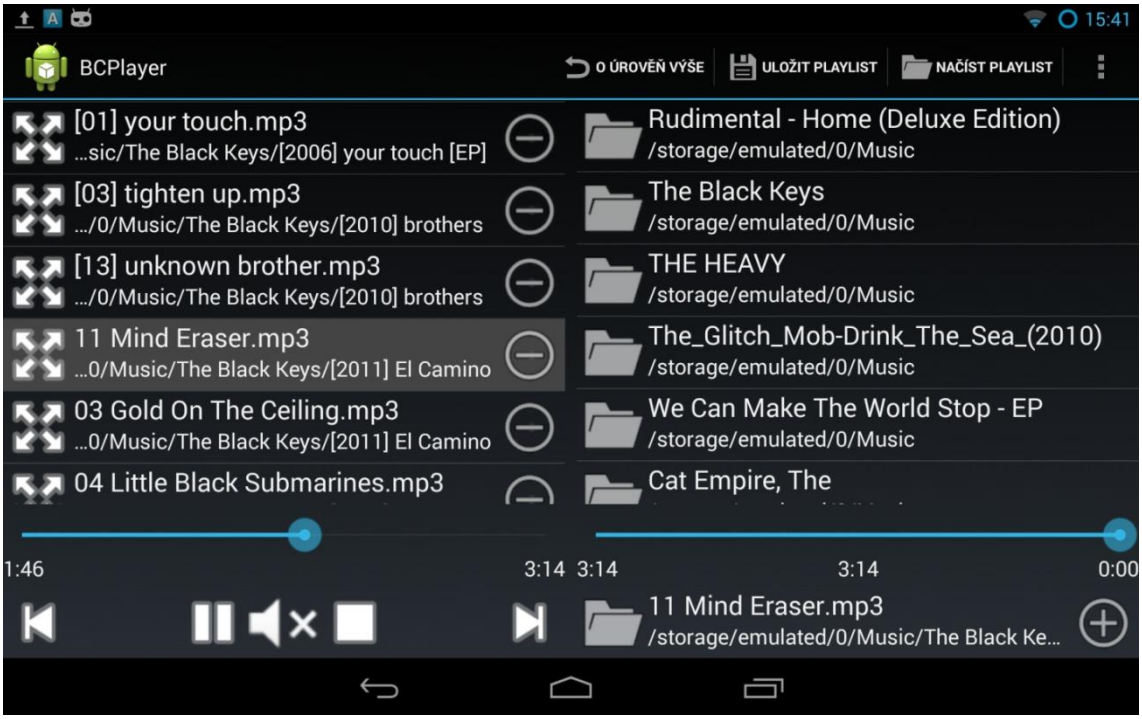
Po konzultaci s vedoucím práce vznikl hrubý nákres podoby přehrávače, ze kterého jsem následně vycházel při návrhu jednotlivých fragmentů v aplikaci přehrávače.



Obrázek 7: Hrubý grafický nákres aplikace

5 Realizace řešení

5.1 Výsledné uživatelské prostředí přehrávače



Obrázek 8: Výsledná grafická podoba přehrávače

Výsledné uživatelské prostředí do sebe integruje všechny potřebné funkce.

Na levé straně se nachází fronta skladeb s vyznačenou právě přehrávanou skladbou. Na pravé straně souborový systém s výpisem složek, případně hudebních souborů. Vybraná složka se otevře, pokud se jí uživatel dotkne. O úroveň výše v souborovém systému se uživatel dostane kliknutím na tlačítko nad souborovým systémem se šipkou zpět a popiskem O úroveň výše.

Dále jsou zde zastoupeny tlačítka a posuvníky obvyklé pro většinu moderních přehrávačů, které jsou popsány podrobněji v následujících kapitolách.

5.2 Fronta skladeb

Frontu skladeb jsem realizoval pomocí *ArrayList*, ve kterém jsou jednotlivé skladby vložené v podobě `HashMap<String, String>`. *ArrayList* umožňuje nejen přidávat další položky (skladby) na konec fronty, ale i na libovolné místo ve frontě, což následně využiji při drag&drop. Do *HashMap* ukládám cestu k hudebnímu souboru a jeho název.

K frontě dále patří celočíselná proměnná `actualSongIndex`, ve které je uložena pozice právě přehrávané skladby ve frontě.

5.3 Ovládací tlačítka přehrávače

Obrázek 9: Hlavní ovládací tlačítka přehrávače

Mezi hlavní ovládací tlačítka přehrávače patří (zleva) Přehrát předchozí skladbu, přehrát/pozastavit aktuální skladbu, tlumené pozastavení, stop a přehrát následující skladbu.

Tlačítko přehrát předchozí skladbu sníží hodnotu proměnné `actualSongIndex` o jedna (nejméně však na nulu) a zavolá metodu `playSong(intposition)`, které předá na místě parametru `position` proměnnou `actualSongIndex`. Podle toho se instanci třídy *MediaPlayer* nastaví zdrojová cesta skladby.

Tlačítko přehrát/pozastavit reaguje dle stavu instance mp třídy *MediaPlayer*. Pokud není žádná skladba v tuto chvíli přehrávaná, nebo pozastavená, tak zavolám metodu `playSong()` jako v předchozím případě. V případě, že skladba již hraje a uživatel použije tlačítko přehrát/pozastavit, tak se zavolá metoda `mp.pause()` a skladba se zastaví na aktuální pozici. Dalším klepnutím na toto tlačítko se zavolá metoda `mp.start()` a skladba začne hrát z místa, kde byla zastavena.

Tlačítko pro tlumené pozastavení plynule sníží hlasitost přehrávání na nulu během necelých 2 sekund a skladbu pozastaví. Toto je docíleno metodou třídy *Handler*`postDelayed(Runnable r, long delayMillis)`, kde prvním parametrem je instance třídy *Runnable* a druhým zpoždění v milisekundách. Zpožděním se určuje, v jakých intervalech se bude spouštět kód v přepsané metodě `run()` třídy*Runnable*, tedy prvního parametru metody.

V mém případě mám nastaveno zpoždění 60 ms a v metodě `run()` plynule snižuji hlasitost o 3 % až na nulu, kdy zavolám metodu instance třídy *Handler*`removeCallbacks(Runnable r)` s parametrem daného *Runnable*, která opakované spouštění metody `run()` zruší.

Hlasitost bylo nutné v každém kroku plynule snižovat lineárně, abych zabránil zprvu pozvolnému snižování hlasitosti a následnému prudkému snížení, k čemuž jsem využil vzorec:

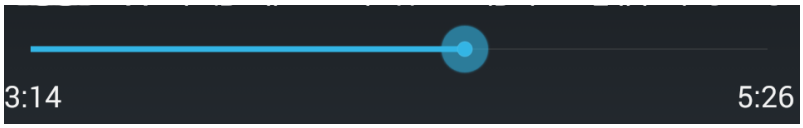
$$hlasitost = 1 - \frac{\log(maximální\ hlasitost - požadovaná\ hlasitost)}{\log(maximální\ hlasitost)}$$

Poté stačilo hlasitost nastavit pomocí `mp.setVolume(hlasitost, hlasitost)`.

Dalším tlačítkem je stop. V tomto případě se pouze zavolá metoda instance třídy *MediaPlayer*`mp.stop()`, která aktuální skladbu ukončí.

Tlačítko přehrát následující skladbu zvýší proměnnou `actualSongIndex` o 1 a zavolá metodu `playSong()` jako v případě tlačítka pro přehrání předchozí skladby.

5.4 Zobrazení a ovládání průběhu skladby



Obrázek 10: Posuvník pro ovládání průběhu skladby

K zobrazení a ovládání průběhu skladby využívám widget jménem *SeekBar* (posuvník, obdoba posuvného studiového potenciometru) a dva *TextView* (zobrazení textu) pro aktuální čas skladby a celkový čas skladby.

Stejně jako v případě tlačítka na tlumené pozastavení skladby i zde využívám metody tříd *Handler* a *Runnable*. Pokud skladba hraje, tak každých 100 milisekund zjistím aktuální čas přehrávané skladby pomocí metody `mp.getCurrentDuration()` a ten vydělím celkovým časem skladby (`metoda mp.getDuration()`). Tuto hodnotu dále vynásobím 100krát, čímž získám procentuální poměr odehraného času, který nastavím metodou `setProgress(intpercentage)` danému *SeekBaru*.

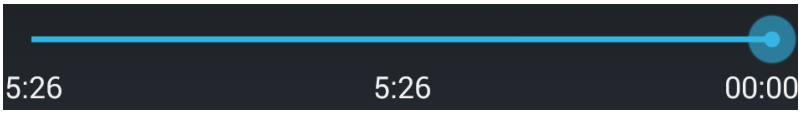
Ve stejném cyklu nastavuji i aktuální čas skladby. Metodě `milliSecondsToTimer(long timeInMiliseconds)` ve třídě *Utils* předám hodnotu získanou v předchozím kroku a výsledný *String* ve formátu *hh:mm:ss* pomocí metody `setText(String text)` nastavím danému *TextView*.

Celkový čas skladby je nastavován pouze jednou a to v okamžiku začátku přehrávání skladby v metodě `playSong()` zmiňované výše.

Pomocí *SeekBaru* je možné i ovládat pozici přehrávané skladby. K tomu je nutné přepsat metody v `SeekBar.OnSeekBarChangeListener()` a to konkrétně `onProgressChanged()`, `onStartTrackingTouch()` a `onStopTrackingTouch()`. Jak již jejich název napovídá, první metoda se volá při každé změně *SeekBaru* (a to i programovému, čehož využívám pro předčasné ukončení skladby popsané o kapitolu dále). Druhá v okamžiku začátku tahu (posunu) *SeekBaru* a třetí v okamžiku uvolnění tahu (v tuto chvíli se zavolá i první metoda, jelikož došlo ke změně).

V okamžiku začátku tahu zavolám metodu `removeCallbacks()` popsanou výše, protože je nutné vypnout nastavování stavu *SeekBaru* při tažení. Jakmile uživatel ukončí tah, je z nové pozice vypočítán metodou z třídy *Utils*`progressToTimer(intprogress, inttotalDuration)` nový požadovaný čas. Tento čas je poté nastaven *TextView* způsobem popsaným výše. Dále je zavolána metoda `mp.seekTo(inttime)`, která posune přehrávání skladby na požadovaný čas. Nakonec je opět obnoveno automatické posouvání *SeekBaru* a času v *TextView* zavoláním metody `postDelayed()` popsané v předchozí kapitole.

5.5 Předčasné ukončení skladby



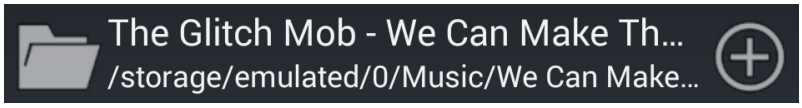
Obrázek 11: Posuvník pro předčasné ukončené skladby

Zde podobně jako v předcházejícím případě využívám *SeekBar* a tentokrát tři *TextView* pro zobrazení konečného času, celkového času a času od konce skladby.

Jakmile uživatel změní hodnotu *SeekBaru*, tak je dle celkového času skladby a nastavené hodnoty vypočten konečný čas. Odečtením konečného času od celkového získáme zbývající čas do konce skladby. Tyto hodnoty jsou následně přepočítány a nastaveny pro daná *TextView*.

Předčasného ukončení skladby je docíleno pomocí porovnávání hodnot obou *SeekBarů* v přepsané metodě `onProgressChanged()` z předchozí kapitoly. Jakmile je hodnota obou *SeekBarů* stejná, nebo u *SeekBaru* z této kapitoly nižší, program přepne na následující skladbu, jelikož přehrávání dosáhlo bodu požadovaného ukončení skladby.

5.6 Zobrazení aktuálně přehrávané skladby



Obrázek 12: Grafická podoba aktuálně přehrívané skladby

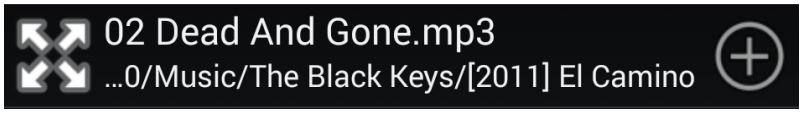
Aktuálně přehrávaná skladba se nachází v pravém dolním rohu přehrávače. Skládá se ze dvou tlačítek (*ImageButton*) a dvou řádků textu (*TextView*).

Tlačítko vlevo, ve tvaru adresáře, otevře na místě souborového systému složku, ve které se nachází aktuálně přehrávaná skladba.

Tlačítko na pravé straně, ve tvaru znaku + přidá danou skladbu na konec fronty.

Textové řádky jsou pouze informativní a zobrazují uživateli v horním řádku název skladby a ve spodním cestu, kde se daná skladba nachází v souborovém systému zařízení.

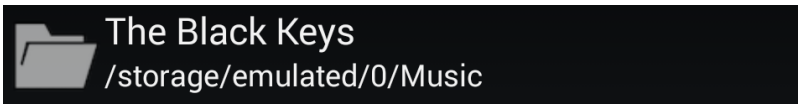
5.7 Formát řádku ve frontě a v souborovém systému



Obrázek 13: Grafická podoba řádku fronty a souborového systému

Zobrazení skladby ve frontě a v souborovém systému je téměř shodné se zobrazením aktuálně přehrávané skladby. Rozdíl je ve funkci tlačítek. Jakmile je skladba ve frontě, tak tlačítko úplně vpravo mění svoji funkci a vzhled. Namísto znaku + je zobrazen znak - signalizující, že kliknutím bude daná skladba z fronty odebrána.

Tlačítko vlevo, ve tvaru čtyř šipek slouží k aktivaci funkce drag&drop. Skladbu je poté možné přetáhnout po displeji na libovolné místo ze souborového systému do fronty skladeb, případně pouze v rámci fronty skladeb. Celý proces drag&drop popisuji v samostatné kapitole.



Obrázek 14: Grafická podoba složky v souborovém systému

V souborovém systému je vhodné od sebe graficky oddělit hudební soubory a složky. K tomu využívám metodu třídy *File*`isDirectory()`, která vrací `true` nebo `false` v závislosti na tom, zda předaná cesta je složkou, či nikoliv. V závislosti na výsledku této metody měním defaultní ikonu čtyř šipek na ikonu adresáře a současně s tím zakazuji funkci drag&drop. Dále v pravé části programově odstraňuji tlačítko na přidání do fronty, neboť v tomto případě postrádá smysl.

5.8 Menu a ActionBar

Obrázek 15: Zobrazení menu v ActionBaru

Menu se v systému Android od verze 4 v aplikacích standardně zobrazuje v pravém horním rohu a má tvar tzv. trojtečky. Zdrojový xml soubor obsahující informace o jednotlivých položkách menu se v projektu nachází v adresáři *res/menu/main.xml* a je možné mít těchto souborů více, pro různé stavy aplikace.

Ve zdrojovém souboru lze mimo popisků a ikon nastavit i jakým způsobem se bude daná položka uživateli zobrazovat. Například pokud položce menu nastavím `android:showAsAction="ifRoom|withText"`, tak se daná položka v *ActionBaru* zobrazí i s popiskem, ale za předpokladu, že pro ni bude na displeji dostatek místa. V opačném případě bude ukryta pod tlačítkem se symbolem trojtečky. V mém případě zde skrývám položku pro odstranění vybraných seznamů skladeb.

Menu je aktivitě přiřazeno přepsáním metody `onCreateOptionsMenu(Menu menu)` ve které je nutné přiřadit správný zdroj podoby menu. To se provede pomocí `getMenuInflater().inflate(R.menu.main, menu)`, kde `R.menu.main` odpovídá definici *main.xml* z adresáře *res/menu/*.

V přepsané metodě `onOptionsItemSelected(MenuItemitem)` je pak možné porovnat ID dané položky menu (*item*) s ID zadané v xml souboru. Tím se dá snadno zjistit, na kterou položku uživatel klikl a podle toho provést zadanou operaci.

První položkou menu je *O úroveň výše*. Pokud na tuto položku uživatel klikne, tak se souborovému systému nastaví nadřazená složka a list se naplní všemi soubory a složkami v ní umístěné. Poté se položky z listu vypíší na displej pomocí metody `notifyDataSetChanged()`, která uvědomí adaptér, že má zpracovat aktuální data.

Dalšími položkami v menu je uložení playlistu (seznamu skladeb), načtení playlistu a odstranění vybraných playlistů. Funkce těchto položek je popsána v kapitole Dialogy.

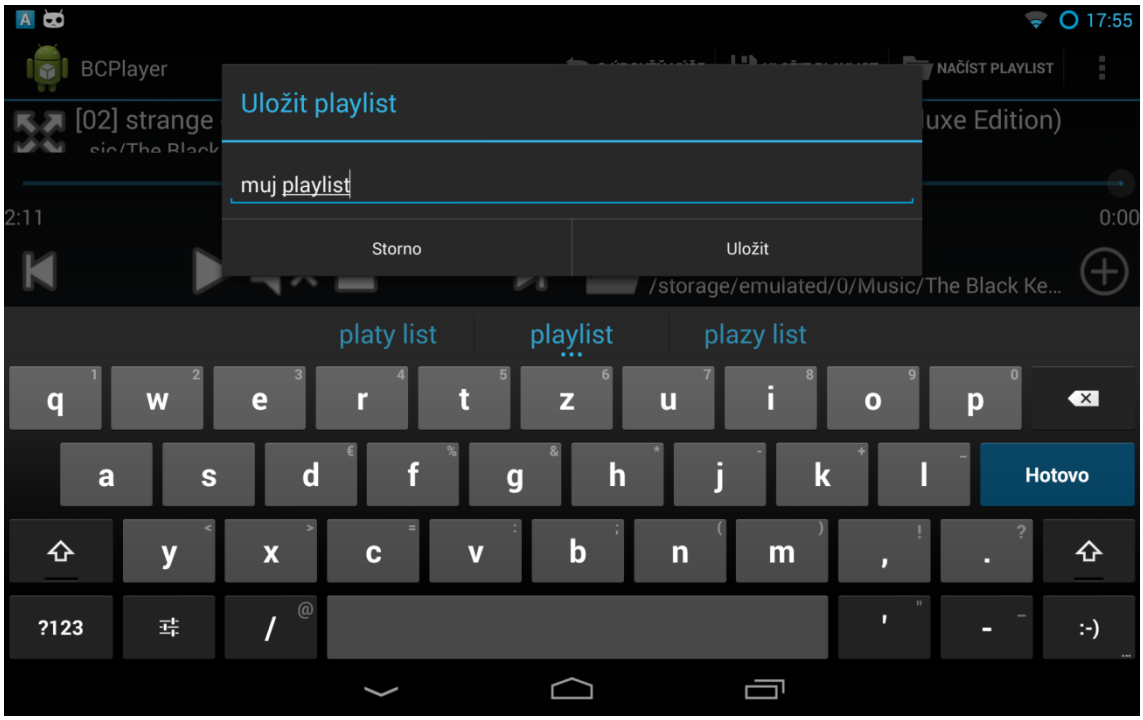
5.9 Dialogy

Dialogy neboli vyskakovací okna známá i z jiných operačních systémů, sloužící k potvrzení nějaké uživatelské akce, či informující uživatele o nějakém stavu aplikace, našlo mnohá využití i v systému Android.

Od Androidu verze 3.0 se pro správu dialogů používá třída *DialogFragment*. Tuto třídu stačí podědit a v metodě `onCreateDialog()` nastavit vše, co je v daném dialogu potřeba zobrazit. K dispozici jsou až 3 ovládací tlačítka, popisek dialogu a prostor mezi tím může být naplněn například textem, seznamem položek, kolonkou pro vepsání textu, případně dalšími prvky, které však ve své aplikaci nevyužívám.

Zobrazení dialogu se provádí pomocí metody `instanceDialogu.show(getFragmentManager(), "Popis Dialogu")`, v tu chvíli systém zjistí, jak je daná instance dialogu definovaná ve své třídě a dialog zobrazí uživateli.

5.9.1 Uložení seznamu skladeb



Obrázek 16: Dialog pro uložená playlistu

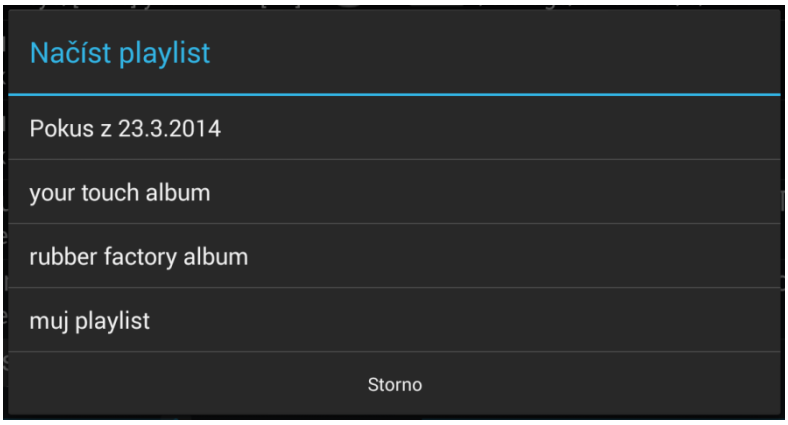
V tomto případě je uživateli zobrazen dialog s *EditText*, kam je možné napsat požadovaný název playlistu. Automaticky je zobrazena klávesnice a celou akci je možné potvrdit kliknutím na tlačítko Uložit, případně zrušit kliknutím na tlačítko Storno či mimo okno dialogu.

Název playlistu je předán přes rozhraní (*Interface*) třídě *MainActivity*, ve které zavolám metodu třídy *PlayerDB*`savePlaylistToDB()`, které předám seznam skladeb a uživatelem zvolený název.

V databázi mám 2 tabulky. Do jedné ukládám názvy playlistů a ID, do druhé pak cesty k jednotlivým skladbám, ID a ID playlistu. Mám tak vytvořenou praktickou relaci 1:n, která mi umožňuje jednoduše porovnat, jaké skladby k danému playlistu patří.

V případě výše popsané metody `savePlaylistToDB()` uložím metodou `insert()` do tabulky playlist název playlistu. Výhodou je, že tato metoda vrací ID vloženého řádku. Toto ID použiji v následujícím cyklu, ve kterém postupně vkládám všechny skladby ze seznamu do tabulky song.

5.9.2 Načtení seznamu skladeb



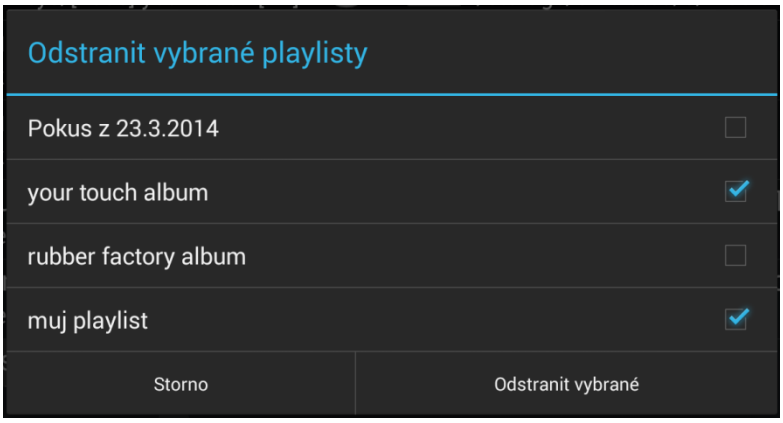
Obrázek 17: Dialog pro načtení playlistu

Po kliknutí na položku menu Načíst playlist je zobrazen seznam playlistů a pokud si uživatel nějaký vybere, je načten na místo aktuální fronty skladeb.

Instanci tohoto dialogu je předán seznam playlistů, získaný metodou `getPlaylists()` z třídy *PlayerDB* a poté je zobrazen v listu. Kliknutím na položku z listu získám její pozici a tu zpětně porovnám s předaným seznamem playlistů, ze kterého dále získám ID playlistu, které využiji v následujícím kroku.

Metodě `getPlaylistByID(int id)` z třídy *PlayerDB* předám získané ID a to v následném dotazu na databázi porovnám se všemi položkami tabulky song. Vybranými položkami poté naplním `ArrayList<HashMap<String, String>>` ve stejném formátu, jako u fronty skladeb a aktuální frontu skladeb přepíši touto, právě načtenou.

5.9.3 Odstranění vybraných seznamů skladeb



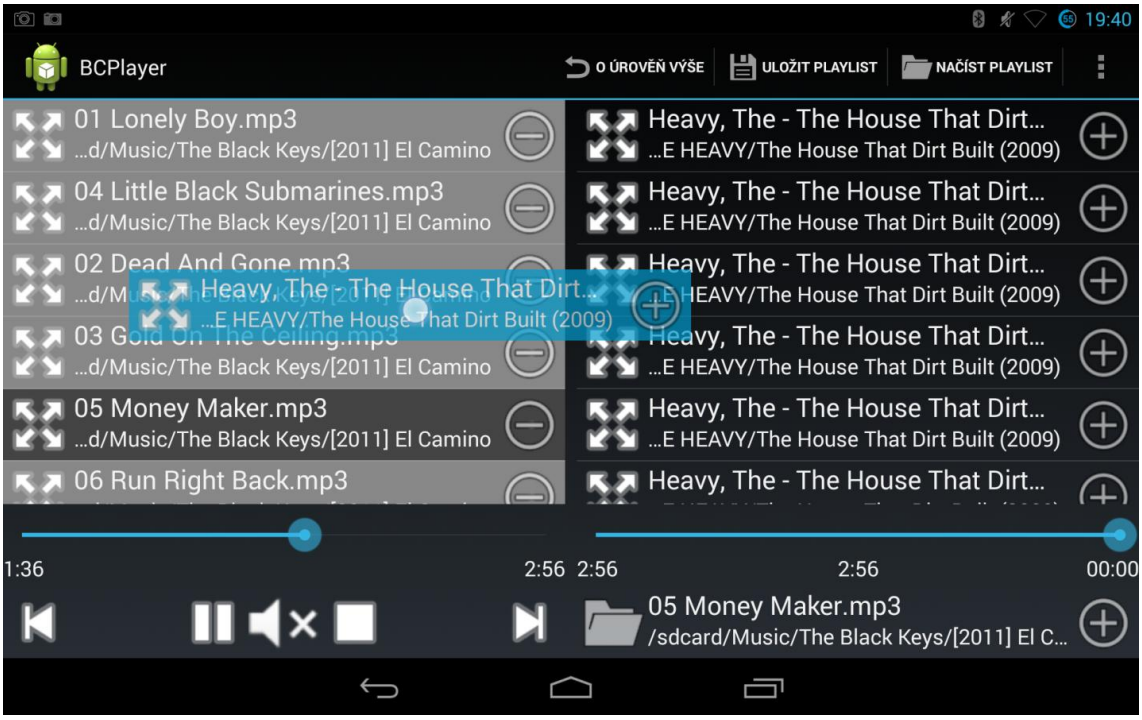
Obrázek 18: Dialog pro odstranění vybraných playlistů

V tomto případě je uživateli nabídnut seznam uložených playlistů, přičemž má možnost označit ty, které si přeje odstranit.

Zde je postup částečně stejný, jako v případě načtení playlistu. Instanci dialogu je před jeho zobrazením předán seznam playlistů, které zobrazí při svém vyvolání. Na rozdíl od předchozího případu je zde možné zvolit více možností a ty pak najednou potvrdit. K zjištění, které položky uživatel odstranil, jsem zvolil *ArrayList*. Pokud uživatel klikne na položku, vložím její pozici do *ArrayListu*, pokud na stejnou položku klikne podruhé, z *ArrayListu* ji odstraním, jelikož to znamená, že označení položky zrušil. Pokud uživatel potvrdí akci kliknutím na tlačítko Odstranit vybrané, tak předám seznam vybraných položek třídě *MainActivity*, opět za využití rozhraní.

Ve třídě *MainActivity* poté procházím celý seznam vybraných položek, z každé z nich získám ID playlistu a to předávám v parametru metodě `deletePlaylistFromDatabase(int id)` třídy *PlayerDB*. Tato metoda poté odstraní všechny položky z databáze, které mají shodné ID, čímž se odstraní daný playlist.

5.10 Drag&Drop



Obrázek 19: Zobrazení aktivní funkce drag&drop

Na obrázku je zobrazena funkce drag&drop. Umožňuje uživateli táhnutím přesunout položku ze souborového systému na jakékoliv místo ve frontě skladeb, případně měnit pořadí skladeb ve frontě. Funkce se aktivuje dlouhým kliknutím na položku. Bílá tečka u tažené skladby na obrázku znázorňuje pozici prstu na displeji a jde o vývojářskou funkci Androidu, v aplikaci není přítomna.

Standardně pro aktivaci funkce dlouhého kliku na položku stačí pro daný list nastavit `setOnItemLongClickListener()` a v metodě `onItemLongClick()` napsat vše potřebné pro danou akci. Pokud ovšem není využíván standardní list ve tvaru jednoho textu na jeden řádek, ale vlastní podoba řádku, tak překladač neví, čemu má *LongClick* přiřadit. Proto je potřeba to specifikovat v rámci deklarace layoutu. To se provádí pomocí `android:longClickable="true"`, který jsem nastavil pro celý Layout. *LongClick* je tedy možné vyvolat z kteréhokoliv místa dané položky.

V metodě `onItemLongClick()` do proměnné typu *ClipData* vložím v případě tažená z fronty pozici skladby, kterou bude taženo. Samotný posun je pak započat zavoláním metody `startDrag()` na dané view, v tomto případě na řádek. Metodě v parametrech předám pozici ve formátu *ClipData* a samotnou skladbu jako *Object*. Tyto data pak využiji v dalším kroku.

Dalším krokem bylo nastavit jak má fronta reagovat, pokud se bude tažený objekt nacházet v jejím prostoru. K tomu slouží `setOnDragListener()` a metoda `onDrag()`. Tělo této metody běží v neustálé smyčce, dokud uživatel táhne objekt po displeji. Podle události právě uložené v instanci třídy *DragEvent* lze určit akci, která se má provést. Například `ACTION_DRAG_ENTERED` a `ACTION_DRAG_EXITED` znamená, že uživatel přetáhl objekt nad frontu, případně mimo. V tomto případě měním pozadí fronty na šedou, resp. zpět na průhlednou. Pokud nastane událost `ACTION_DROP`, znamená to, že uživatel přetáhl objekt nad frontu skladeb a uvolnil jej.

Tuto událost zpracovávám v metodě `processDrop(DragEventevent)`. Z proměnné *event* získám informace o dané skladbě a souřadnice, na kterých uživatel uvolnil objekt. Ze souřadnice y a informace o délce odrolování fronty získané přes metodu `getTop()` volanou na prvního potomka listu je vypočtena pozice, na kterou si uživatel přál vložit skladbu. Na toto místo je tedy skladba do *ArrayListu* vložena a zároveň je přepočítán index právě přehrávané skladby. Tím je tato funkcionalita vyřešena.

6 Vyhodnocení řešení

Nainstalovaná aplikace zabírá pouze 208 kB v paměti zařízení. Její velikost se zvyšuje úměrně s tím, kolik má uživatel uložených seznamů skladeb v databázi. Vzhledem k tomu, že jsou ukládána pouze nejnutnější data pro opětovné načtení playlistu, není celková náročnost na paměť nikterak velká.

Využití aplikace sice umožňuje použít pouze jednu definici rozložení a jednu aktivitu, znamenalo by to však nepřehledný kód, možné problémy s dalším rozšířením aplikace (jeden layout má omezený počet povolených uživatelských prvků) a problémy s výkonem na slabších zařízeních.

Aplikace se skládá z celkem 18 tříd a 12 xml definic rozložení. Celkem 1750 řádků kódu ve třídách a 450 řádků kódu xml.

Nad rámec zadání jsem vyřešil i ukládání playlistů. K tomu se dá přistupovat několika způsoby. Jedním z nich je například ukládat je ve formátu m3u do zařízení, případně na paměťovou kartu. Zde jsem ale narazil na problém, který souvisí s tím, že bych pokaždé musel od uživatele zjišťovat, na jaké místo si přeje playlist uložit. Dalším problémem je obrovská časová náročnost zápisu na paměťovou kartu. Proto jsem se rozhodl přistoupit k jednoduchému a výkonnému řešení. Tím je ukládání playlistu do vnitřní SQLite databáze Androidu.

Grafické uživatelské rozhraní jsem, troufám si tvrdit, vytvořil dostatečně jednoduché a přehledné v porovnání s konkurenčními přehrávači. Využil jsem standardní ikony Androidu a nakopíroval je do zdrojových souborů aplikace. Díky tomu bude aplikace napříč verzemi Androidu vypadat stále stejně. Do budoucna je nicméně možné GUI zcela předělat.

Možné problémy vidím ve využití vláken pomocí třídy *Runnable* a *Handler*. V systému Android pro toto využití existuje speciální třída *AsyncTask*, která by se pro tyto účely měla používat.

7 Závěr

Po bližším obeznámení s vývojovým prostředím Android studio a funkcí jednotlivých komponent systému Android se mi postupně podařilo splnit všechny body zadání.

Drag&drop funguje přesně dle zadání. Je tedy možné měnit pořadí přehrávaných hudebních souborů ve frontě. Hudební soubory lze táhnutím přenést ze souborového systému do fronty přehrávače.

K běžným funkcím přehrávače (*play, stop, pause*) jsem přidal funkci pro postupné utlumení hlasitosti a zastavení aktuálně přehrávané skladby. Dále možnost přepnutí na následující, případně předchozí, skladbu ve frontě. Zobrazení odehraného času a předčasné ukončení skladby jsem realizoval dvěma posuvníky a textovými popisky nesoucí informace o celkovém a aktuálním času skladby.

Věřím, a reakce okolí mi to potvrzují, že jsem pro uživatele vytvořil jednoduché a přehledné grafické prostředí.

Nad rámec zadání jsem ještě doplnil možnost ukládání a načítání seznamů skladeb. K tomu jsem využil zabudovanou databázi Androidu SQLite.

Naučil jsem se využívat dostupné možnosti vývoje aplikací pro operační systém Android. Aplikaci považuji za přínosnou a mám zájem ve vývoji nadále pokračovat i mimo rámec bakalářské práce.

Myslím, že jsem vytvořil přehrávač, který nemá na trhu konkurenci ve svém konceptu snadnosti ovládání a přehlednosti grafického prostředí.

Do budoucna by se dala na aplikaci například vylepšit funkce drag&drop.Co se týká přesnějšího zobrazení, na jaké místo bude tažená skladba vložena. Dále je možné doplnit na místo souborového systému například řazení skladeb dle autora, žánru, alba apod. Vzhledem k tomu, že jsem na možnost rozšíření aplikace při vývoji myslel a využil tzv. fragmenty, tak je možné nové funkce v budoucnu přidat bez většího zásahu do aktuálního zdrojového kódu aplikace. K jednotlivým skladbám ve frontě by bylo například vhodné přidat informaci o délce dané skladby. Dalším možným řešením celé aplikace by mohl být například ekvalizér.

Použitá literatura

1. **Allen, Grant.***Android 4 - Průvodce programováním mobilních aplikací.* Brno : Computer Press, 2013.
   978-80-251-3782-6.

2. **Tamada, Ravi.** Android Building Audio Player Tutorial. *Android Hive.* [Online] 5. 3 2012. [Citace: 12. 4 2014.] http://www.androidhive.info/2012/03/android-building-audio-player-tutorial/.

A Obsah přiloženého CD

- Text bakalářské práce
  – bakalarska_prace_2013_Ladislav_Hofman.pdf
  – bakalarska_prace_2013_Ladislav_Hofman.docx
- Zdrojový kód aplikace
  – zdrojovy_kod.zip
- Instalační soubor aplikace
  – bcplayer.apk