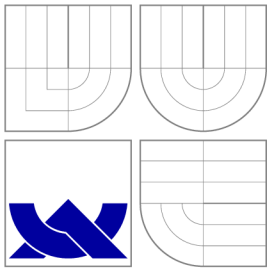
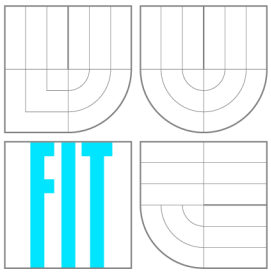


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

IMPLEMENTUJTE ANDROID APLIKACI PRO STAHOVÁNÍ A POSLECH AUDIO PODCASTŮ

IMPLEMENT ANDROID APPLICATION FOR DOWNLOAD AND LISTENING TO AUDIO PODCASTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP BAJANÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IGOR SZŐKE, Ph.D.

BRNO 2016

Zadání bakalářské práce/18060/2015/xbajan01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Bajaník Filip**
Obor: Informační technologie
Téma: **Implementujte Android aplikaci pro stahování a poslech audio podcastů**
Implement Android Application for Download and Listening to Audio Podcasts
Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte teorii a praxi k podcastům, najděte a porovnejte podobné aplikace.
2. Navrhněte a implementujte podcastovací aplikaci. V aplikaci se zaměřte na UX (uživatelskou zkušenost) a oboustranou komunikaci se službou Audeliver.com.
3. Aplikaci otestujte na vhodném vzorku beta-testerů. Zveřejněte aplikaci na Google Play.
4. Pokračujte v implementaci a změnách GUI pro co nejlepší UX. Získejte zpětnou vazbu od uživatelů.
5. Zhodnoťte výsledky a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a část bodu 3 ze zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto bakalárska práca sa zaoberá implementáciou Android aplikácie pre odoberanie a počúvanie audio podcastov. Aplikácia je zameraná na prívetivé užívateľské rozhranie a integráciu služby Audeliver. Prvá časť práce sa zaoberá teoretickými východiskami, rozborom existujúcich riešení, z ktorých vychádza aplikácia. Ďalšie časti sú venované návrhu, implementácií a testovaniu výsledného riešenia. V závere je výsledná aplikácia zhodnotená a je naznačený jej ďalší potencionálny vývoj.

Abstract

This bachelor thesis deals with implementation of an Android application for downloading and listening to the audio podcasts. Application is mainly focused on a friendly user interface and integration of the podcast service Audeliver. The first part of this thesis contains theoretical specifications of the platform and analysis of the existing solutions. The second part is concentrated into the actual application design, implementation and testing. The final part reviews the progress of the final application and presents possibilities of the potential development.

Kľúčové slová

Android, aplikácia, Audeliver, podcast, audio, Java

Keywords

Android, application, Audeliver, podcast, audio, Java

Citácia

BAJANÍK, Filip. *Implementujte Android aplikaci pro stahování a poslech audio podcastů.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Příjmení Jméno.

Implementujte Android aplikaci pro stahování a poslech audio podcastů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Igora Szőkeho, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Filip Bajaník
18. mája 2016

Poďakovanie

Rád by som sa poďakoval pánu Ing. Igorovi Szőkemu, Ph.D. za vedenie tejto bakalárskej práce, odbornú pomoc a cenné rady, ktoré viedli k dosiahnutiu cieľa.

Obsah

Kapitola 1

Úvod

Hudobný priemyseľ dnes zažíva revolúciu. Z predaja fyzických médií sa prechádza na online distribúciu a celé odvetvie začína byť unášané práve týmto prúdom. Formy šírenia online audio diel sú však rôzne. V roku 2000 vznikol koncept tzv. podcastingu, ktorý začal šírenie audio diel vo forme online epizód na pokračovanie. Užívateľ sa môže prihlásiť na odber týchto podcastov a pohodlne odoberať podcast svojej obľubenej rádiostanice, po ceste domov autom sa môže venovať vzdelávaniu alebo počúvať svoj obľúbený hudobný podcast.

Najpohodlnejšou cestou ako odoberať audio podcast je prostredníctvom mobilnej aplikácie. Táto aplikácia má poskytnúť možnosť, ako čo najjednoduchšie získavať a počúvať svoje obľúbené podcasty a vyhľadávať v epizódach, ktoré užívateľov zaujímajú.

Cieľom tejto bakalárskej práce je implementácia Android aplikácie pre vytváranie, sťahovanie a počúvanie vlastných audio podcastov, s využitím webovej služby Audeliver. Mobilný operačný systém Android bol zvolený práve kvôli rozšírenosti. V súčasnosti je jeho podiel na trhu približne 57%[1]. Aplikácia by mala byť zameraná na prívetivé užívateľské rozhranie a integráciu spomínanej služby Audeliver. Mala by odstrániť bariéry pri používaní podcast aplikácií a v konečnom dôsledku byť užitočnou.

Práca je rozčlenená do siedmich kapitol. Kapitola 2 popisuje analýzu a špecifikáciu požiadavkov na aplikáciu, nastieňuje problematiku samotného podcastingu, podcastov a ich dátových štruktúr. Taktiež sa venuje službe Audeliver a popisom jej aplikačného rozhrania. Kapitola 3 sa zaoberá návrhom užívateľského rozhrania a databázy aplikácie. Kapitola 4 sa venuje základným teoretickým princípom vývoja pre platformu Android, jeho architektúrou a základným popisom jeho komponentov potrebných pri implementácií aplikácie. Kapitola 5 rieši implementáciu funkčného, ale aj grafického riešenia. Kapitola 6 je venovaná testovaniu výslednej aplikácia. Poslednou kapitolou je záver, ktorým práca zhrnie dosiahnuté výsledky a nastieni možný vývoj.

[1] Podiel na trhu - http://marketshare.hitslink.com/ (12.1.2016)

Kapitola 2

Analýza a špecifikácia požiadavkov

Čo je to podcat a aké aplikácie na prijímanie podcastov poznáme? Táto otázka bude rozobraná na nasledujúcich riadkoch spolu s analýzou. Analýza poslúži ako základ pre špecifikáciu požiadavkov vlastnej aplikácie. Taktiež poskytne náhľad na existujúce a aktuálne populárne riešenia aplikácií. Pri každej aplikácií bude popis ich plusov a mínusov. Zo zhrnutých poznatkov analýzy sa vyvodí záver a vytvorí špecifikácia požiadavkov.

2.1 Podcasty

Podcasty sú jedna z foriem multimediálneho záznamu, ktorý je distribuovaný online vo forme **epizód**. Na podcasty sa užívatelia prihlasujú k odberu s cieľom konzumovať audio, video, rádio podcast alebo epizódu textového dokumentu **PDF** alebo **ePub**. Autori tieto podcasty uverejňujú vo formáte webového **RSS feedu**[1]. Konkrétnou štruktúrou dát sa bude zaoberať sekcia 2.2. Cieľov tejto práce je implementovať aplikáciu pre odoberanie audio podcastov a integráciou konkrétnej služby **Audeliver**[2].

Výhody podcastu?

- **Efektívnejšie využitie času** – v dnešnom uponáhľanom svete sa čas na vzdelávanie alebo osobný rozvoj hľadá obtiažnejšie. Podcasty však predstavujú ideálny prostriedok ako vyplniť aj hluché miesta, kedy človek napríklad cestuje autom, verejnou dopravou alebo pri cvičení a má tak možnosť dozvedieť sa niečo nové, inšpirovať sa alebo sa pobaviť.
- **Dostupnosť** - podcasty môžu byť po stiahnutí do zariadenia dostupné aj bez internetového pripojenia, čo značne rozširuje ich využitie. Podcasty sa dajú prirovnať k rádiovým vysielaniam, len s tým rozdielom, že si ich môžete vypočuť kedykoľvek a kdekoľvek.
- **Osobnejšia forma predávania informácií** - audio je osobnejšou formou predávania informácií oproti napríklad čítaniu kníh a internetových článkov. Dôvodom je, že človek počuje hlas a medzi poslucháčom sa vytvára osobnejší vzťah. Pri niekoľko hodinových záznamoch vzniká pocit, že je poslucháč súčasťou konverzácie, napríklad úspešných ľudí. Tento pocit je jeden z dôvodov, prečo sú podcasty tak populárne.

[1] RSS - http://www.rssboard.org/rss-specification
[2] Audeliver - http://audeliver.com/

- **Rôznorodosť** - podcasty nie sú len diskusné relácie, ale aj hudobné stanice. Mnohí umelci si robia z podcastov svoje hudobné rádiá. Taktiež aj niektorí bloggeri alebo lektori využívajú podcasting a sprístupňujú svoje myšlienky a diela nahovorením svojich článkov.

2.2 Štruktúra dát RSS feedu

Táto sekcia popisuje štruktúru dát RSS feedu a hlavné položky, ktoré budú využité pri programovom spracovávaní. RSS je textový formát, ktorý spadá do rodiny XML. Dátové štruktúry tvoria XML **tagy**, ktoré môžu byť párové alebo nepárové. Samotný RSS feed môže obsahovať väčší počet tagov, no pre potreby aplikácie sa budem sústredovať iba na tie najzaujímavejšie.

Item

Tag, ktorý reprezentuje jednu epizódu podcastu. Obsahuje aj špeciálny *itunes namespace[2]* s tagmi ako napríklad **duration**. Jeho štruktúra je znázornená v tabuľke 2.1.

title	krátky názov epizódy
itunes:subtitle	podtitulok, skrátený popis epizódy
description	dlhší opis danej epizódy
link	adresa dátového súboru vo formáte `mp3`
enclosure	okrem adresy obsahuje aj formát a veľkosť dátového súboru v `bitoch`
itunes:duration	trvanie epizódy v časovom formáte `"HH:mm:ss"`
pubDate	dátum publikovania epizódy vo formáte EEE, `"dd MMM yyyy HH:mm:ss Z"`
itunes:keywords	kľúčové slová, podľa ktorých sa daná epizóda bude dať vyhľadať

Tabuľka 2.1: Štruktúra **item** tagu z RSS feedu

Channel

Základné údaje o danom podcaste sú obsiahnuté v tagu Channel. Potomkovia tohto tagu tvoria epizódy, ktorých môže podcast obsahovať viac. Tento tag sa v dokumente vyskytuje iba jeden krát. Jeho štruktúra je znázornená v tabuľke 2.2. Ukážka podcast feedu sa nachádza v príklade 2.1.

title	názov celého podcastu
description	krátky popis podcastu
atom:link	unikátna adresa reprezentujúca adresu RSS feedu
itunes:image	obsahuje adresu k hlavnému obrázku podcastu
lastBuildDate	dátum posledného zostavenia RSS feedu vo formáte `"EEE, dd MMM yyyy HH:mm:ss Z"`[3]
itunes:category	kategórie oddelené čiarkou, pod ktoré spadá daný podcast

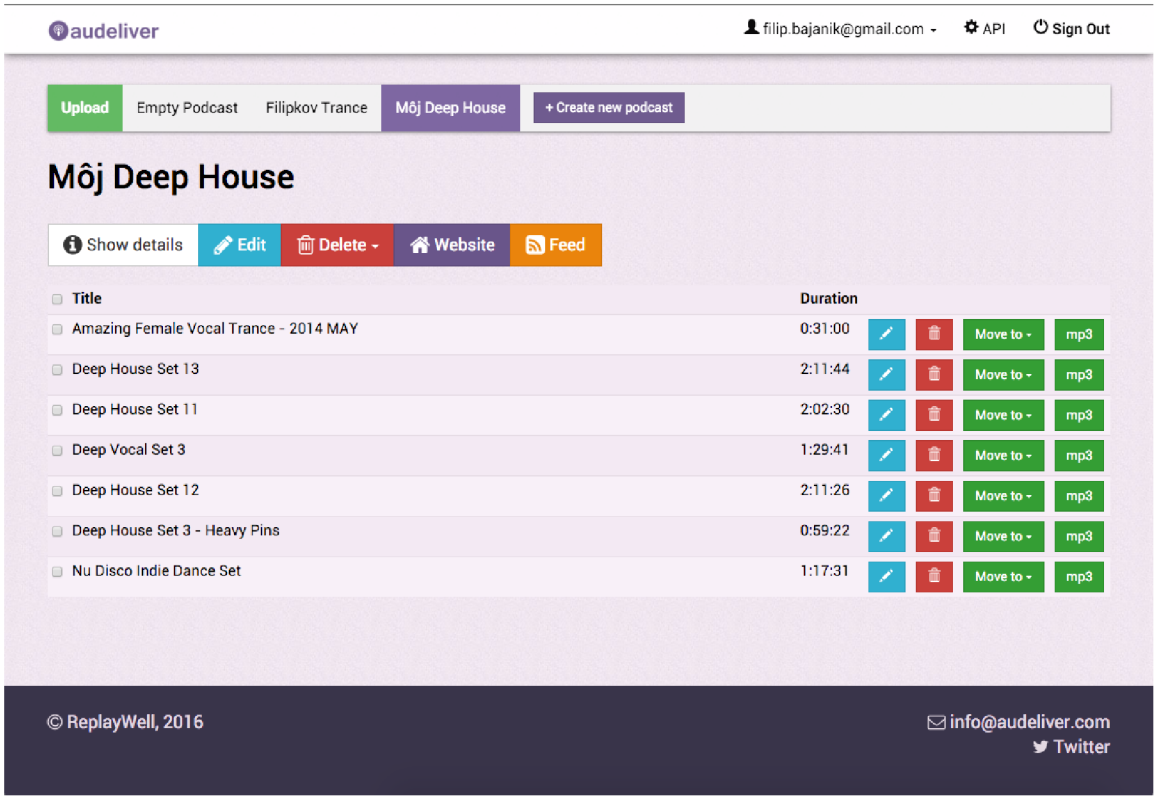
Tabuľka 2.2: Štruktúra **channel** tagu z RSS feedu

```
<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd" xml:lang="de
    " version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
<channel>
<title>My TED podcasts</title>
<link>https://audeliver.com/</link>
<atom:link href="http://audeliver.com/podcast/RPxRMixzeQGJ/feed.xml" rel="
    self" type="application/rss+xml" />
<description></description>
<generator>Audeliver - http://audeliver.com</generator>
<lastBuildDate>Mon, 1 May 2016 16:18:49 +0200</lastBuildDate>
<language>en</language>
<copyright>Copyright 2016 Filip Bajanik. All Rights Reserved.</copyright>
<itunes:image href="http://audeliver.com/imagecloud/podcast0eeed3c.jpg" />
<image>
        <url>http://audeliver.com/imagecloud/podcast0eeed3c.jpg</url>
        <title>My TED podcasts</title>
        <link>https://audeliver.com/</link>
</image>
<itunes:summary></itunes:summary>
<itunes:subtitle></itunes:subtitle>
<itunes:author>Filip Bajanik</itunes:author>
<itunes:owner>
        <itunes:name>Filip Bajanik</itunes:name>
        <itunes:email>filip.bajanik@gmail.com</itunes:email>
</itunes:owner>
<itunes:category text="Technology">
        <itunes:category text="Podcasting" />
</itunes:category>
<itunes:explicit>yes</itunes:explicit>
<item>
        <title>The Best Stats Youve Ever Seen | Hans Rosling</title>
        <itunes:subtitle></itunes:subtitle>
        <itunes:summary><![CDATA[ ]]></itunes:summary>
        <description></description>
        <link>http://audeliver.com/mp3cloud/VcKLpIZHRAMzrK.mp3</link>
        <enclosure url="http://audeliver.com/mp3cloud/VcKLpIZHRAMzrK.mp3"
            length="19780369" type="audio/mpeg"/>
        <guid>http://audeliver.com/mp3cloud/VcKLpIZHRAMzrK.mp3</guid>
        <itunes:duration>0:20:36</itunes:duration>
        <itunes:keywords></itunes:keywords>
        <itunes:explicit>no</itunes:explicit>
        <pubDate>Sun, 1 May 2016 21:22:14 +0200</pubDate>
</item>
</channel>
</rss>
```

Príklad 2.1: Ukážkova štruktúra podcastu v XML

2.3 Služba Audeliver

Služba Audeliver poskytuje užívateľovi jednoduchý spôsob ako si vytvoriť svoj vlastný podcast. Podcasty sú jednoducho manažovateľné a každý užívateľ si môže vytvoriť svoju vlastnú formu internetového rádia, ktoré bude zdielať s priateľmi alebo sám používať.



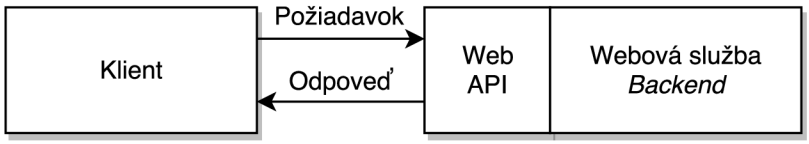
Obr. 2.1: Užívateľské rozhranie služby Audeliver

Audeliver podporuje nahratie svojej vlastnej hudobnej knižnice alebo audio podcastu vo formátoch `mp4`, `mp3`, `mpeg`, `wav`, `ogg`, `wmv`, `flac`, `mp2`, `mpeg` a mnohé ďalšie. Okrem iného ponúka Audeliver sťahovanie audio súborov z populárnych služieb akými sú napríklad `Youtube`, `Vimeo`, `SoundCloud` a iné služby.

2.4 Popis Audeliver API

Súčasťou zadania bakalárskej práce je aj obojstranná komunikácia so službou Audeliver. Klient, resp. mobilné zariadenie, bude komunikovať prostredníctvom **REST API**[4] protokolu. REST API je webové aplikačné programové rozhranie (**API**), pracujúce s protokolom **HTTP**[5], ktoré poskytuje funkcie k uľahčeniu komunikácie medzi aplikáciami, umožňujúc vzájomnú výmenu dát. Schéma komunikácie je znázornená na obrázku 2.2.

[3]Formát použitého dátumu - http://www.w3.org/Protocols/rfc822/#z28
[4]https://www.ibm.com/developerworks/webservices/library/ws-restful/
[5]HTTP - https://tools.ietf.org/html/rfc2616



Obr. 2.2: Komunikácia klienta so serverom

Základné princípy[8] **REST API:**

- Stav aplikácie a jej chovanie sa vyjadruje tzv. zdrojom. Každý zdroj musí mať unikátny identifikátor (URL[6], URN[7]).
- Stav aplikácie je určený pomocou URL. Ďalšie stavy môžme získať pomocou adries URL, ktoré klient dostane v odpovedi zo serveru.
- Je definovaný jednotný prístup pre získanie a manipuláciu so zdrojom v podobe operácií (**PUT**, **GET** a **POST**, **DELETE**).
- Zdroj môže mať rôzne reprezentácie (XML, HTML, JSON, SVG, PDF), klient teda nepracuje priamo so zdrojom, ale s jeho reprezentáciou.

API využíva s výnimkou registrácie pri všetkých HTTP dotazoch **Basic Access Authentication**[3]. Táto autentifikácia obsahuje zašifrované prihlasovacie údaje k účtu uživateľa, ktoré sa nazývajú **autentifikačný token**. Tento token sa pripája do hlavičky výsledného dotazu. Kladná odpoveď vráti požadované dáta vo formáte XML, pričom neautentifikovaným užívateľm príjde záporná chybová odpoveď znázornená na príklade 2.2.

```
<result>
        <error>
                <status>401</status>
                <message>Unauthorized.</message>
        </error>
</result>
```

Príklad 2.2: Správa pre nepovolený prístup

Zoznam funkcí Audeliver API je znázornený v tabuľke 2.3 a umožňuje vytvárať, editovať a mazať podcasty spolu s epizódami. Taktiež je možné získavať informácie o prihlásenom užívateľovi alebo registrovať nového uživateľa. Záporná odpoveď je už v spomínanom formáte chybovej hlášky z príkladu 2.2, no môže sa líšiť obsahom chybového čísla **status** alebo obsahom správy položky **message**. Reprezentácia príjmaných dát je podobná s formátom uvedeným v tabuľke 2.1 a tabuľke 2.2, preto nebudú znovu popisované.

[6]Uniform Resource Locator - jednotná adresa zdroja
[7]Uniform Resource Name - jednotný identifikátor mena

Uživateľ		
Metóda	**Cesta zdroju**	**Popis**
GET	/user	Vráti uživateľské dáta
POST	/user	Vytvorí nového uživateľa
PUT	/user	Upraví uživateľské dáta
Záznamy epizód		
Metóda	**Cesta zdroju**	**Popis**
GET	/recordings	Vráti všetky záznamy
POST	/recordings	Vytvorí nový záznam
DELETE	/recordings/{id}	Vymaže záznam
GET	/recordings/{id}	Vráti konkrétny záznam
PUT	/recordings/{id}	Upraví záznam
Podcasty		
Metóda	**Cesta zdroju**	**Popis**
GET	/podcasts	Vráti zoznam podcastov
POST	/podcasts	Vytvorí nový podcas
DELETE	/podcasts/{id}	Vymaže podcast
GET	/podcasts/{id}	Vráti konkrétny podcast
PUT	/podcasts/{id}	Upraví podcast

Tabuľka 2.3: Popis API služby Audeliver

Atribút **id** je súčasťou cesty k zdroju a určuje konkrétnu cestu k dotazovanému podcastu alebo epizóde. Okrem atribútov, ktoré sa nachádzajú v ceste dotazu sú aj atribúty, ktoré sa posielajú formulárom. Tieto atribúty sú typu **formData**[8] a skladajú sa zo sady dvojíc **kľuč/hodnota**. Príklad dát, posielaných pri vkladaní novej epizódy, je v tabuľke 2.4.

Atribút	Povinné	Popis	Dátový Typ
url	**X**	URL média súboru	string
title		Názov epizódy	string
subtitle		Podnázov epizódy	string
keywords		Kľučové slová oddelené čiarkou	string
description		Popis epizódy	string
podcastId		Podcast ku ktorému sa má epizóda pridať	string

Tabuľka 2.4: Tabuľka znázorňuje dáta, ktoré sa posielajú pri **POST** dotaze na server služby.

2.5 Analýza existujúcich riešení

Táto sekcia obsahuje zoznam analyzovaných aplikácií. Popis sa bude týkať funkčného, ale i grafického riešenia. Opisujú sa však iba vybrané rysy aplikácie, nakoľko majú všetky aplikácie veľa spoločných funkčných, ale aj grafických rysov. Obrázok 2.3 znázorňuje najpoužívanejšie aplikácie. Analýza bude použitá na vytvorenie špecifikácie požiadavkov, ktorá bude združovať najlepšie vlastnosti vybraných aplikácií.

[8] https://developer.mozilla.org/en-US/docs/Web/API/FormData

Podkicker

Medzi najväčšie prednosti tejto aplikácie patrí účelný a jednoduchý design. Podkicker však skrýva aj množstvo nastavení, ktoré využijú hlavne skúsenejší užívatelia, no nie všetky dôležité akcie sú vždy na dosah jedného kliknutia. Veľký plus je aj v obrovskej databáze podcastov a možnosť pridávania podcastov z rôznych zdrojov.

Aplikácia však nedisponuje možnosťou tvoriť playlisty a občasne sa po návrate do aplikácie vyskytne problém so streamovaním[9] epizód. Táto chyba môže byť pre užívateľov, ktorí si streamujú podcasty, závažná. Funkcia plánovaného vypnutia prehrávania je skrytá hlboko v nastaveniach.

Beyondpod

Aplikácia Beyondpod už disponuje možnosťou tvoriť vlastné playlisty a označovať obľúbené epizódy. Taktiež má užívateľské rozhranie prispôsobené pre ovládanie v aute. Toto rozhranie sa však aktivuje kliknutím na kompaktný prehrávač, ktorý obsahuje len jedno tlačidlo na prehranie a pozastavenie podcastu, viď. obrázok 2.3f.

Zložitosť ovládania a príliš vela skrytých možností môže odradiť potencionálneho užívateľa už po nainštalovaní aplikácie. V rôznych zoznamoch epizód chýba vyhľadávanie a vypočuté epizódy sa automaticky vymazávajú.

Podcast Addict

Uživateľovi poskytuje aplikácia prehľadné filtrovanie podľa kategórií a zoznam rozpočúvaných podcastov. Výhodou je aj možnosť sťahovania a streamovania epizód iba počas dostupnosti WiFi a limit na počet stiahnutých epizód.

Aplikácia má podľa dizajnových pravidiel[10] od Google neaktuálny design a taktiež je svojou zložitosťou nevhodná pre začiatočníkov.

Podcast Republic

Medzi výhody tejto aplikácie patrí história prehrávania, ktorá môže užívateľovi pomôcť nájsť už vypočutý podcast. K navigácií medzi epizódami prispieva jednoduché filtrovanie podľa zvolených parametrov. Filtrovaniu však sekunduje nie veľmi dobre navrhnuté vyhľadávanie, pri ktorom užívateľ nevidí dôležitý obsah a to ani pri väčších uhlopriečkach (5.5”). Zaujímavosťou aplikácie je česká lokalizácia a zabudovaný ekvalizér.

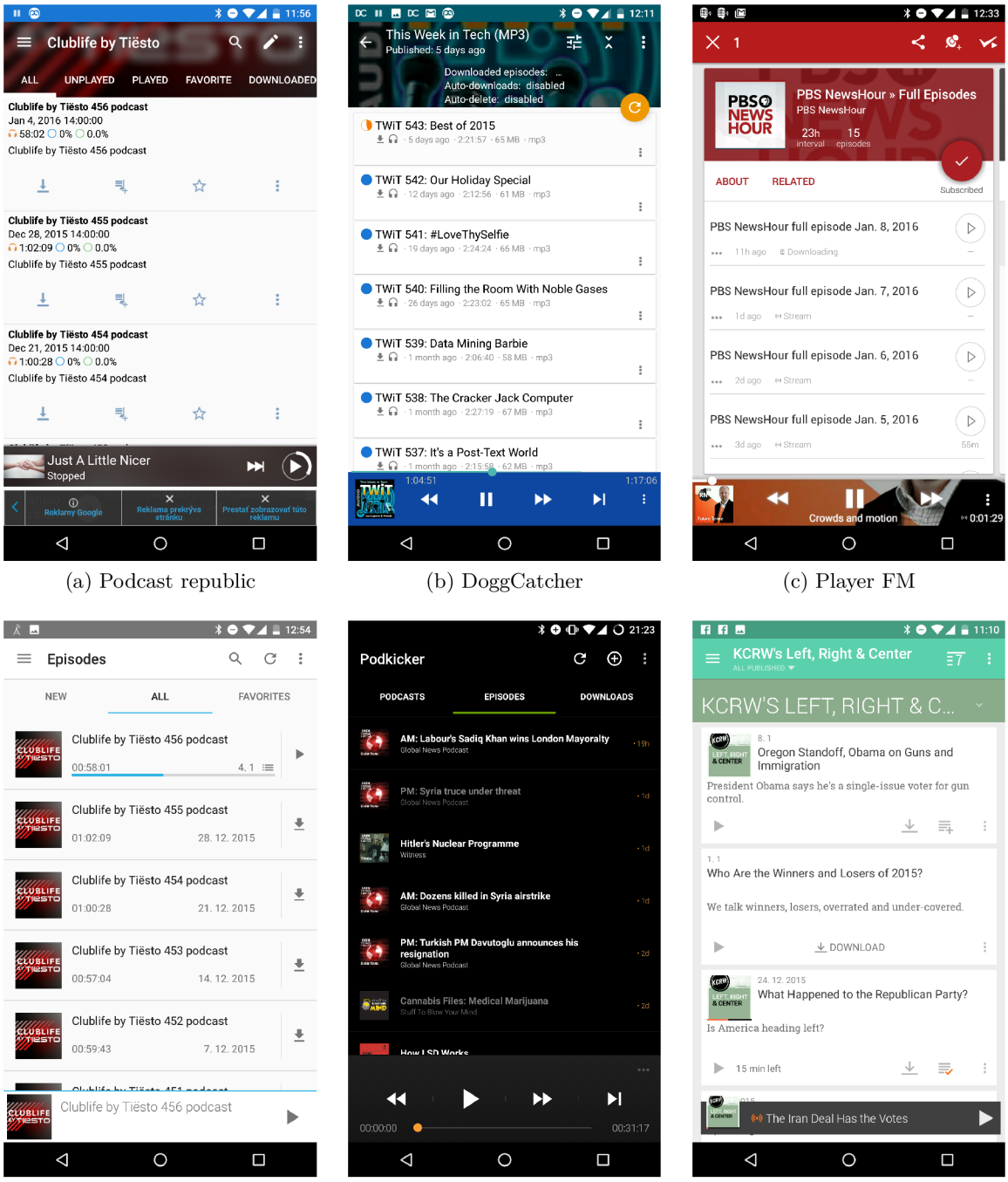
Bohatú funkčnosť aplikácie zhoršuje jej nestabilita a chaotické užívateľské rozhranie. To je charakterizované duplicitami nastavení, kde sú rovnaké možnosti zobrazené na hlavnej obrazovke, v nastaveniach, v hlavnom menu a vo vysúvacom menu.

Dogg Catcher

Kompaktný prehrávač aplikácie, ktorý je k dispozícií na každej obrazovke, obsahuje všetky základné akcie na navigáciu pri prehrávaní. Plusom je aj popis ku každej položke v nastavení.

[9] Streaming - technológia kontinuálneho prenosu audiovizuálneho materiálu medzi zdrojom a koncovým užívateľom

[10] Material design - https://www.google.com/design/spec/material-design/introduction.html



(a) Podcast republic (b) DoggCatcher (c) Player FM

(d) AntennaPod (e) Podkicker (f) Beyondpod

Obr. 2.3: Zobrazenie typického štruktúrovania zoznamu epizód, rozloženie najdôležitejších elementov a základné ovládacie prvky prehrávačov vybraných aplikácií. Aplikácie predstavujú výber toho najpoužívanejšieho na platforme Android.

Hlavné menu aplikácie je však zbytočne preplnené možnosťami, ktoré nepatria pod bežné prípady užitia takéhoto typu aplikácie (Ohodnodte nás, Pomoc, Licencie, O aplikácií, Ukončiť aplikáciu).

Player FM

Player FM prináša synchronizáciu s webovým účtom, čo môže byť pre aktívných užívateľov veľkou výhodou, nakoľko môžu s prehrávaním pokračovať vo webovom rozpraní. Vítaná je aj možnosť označiť epizódy na prehratie neskôr. Tieto epizódy sa po pripojení k WiFi automaticky stiahnu. Zaujímavosťou aplikácie je aj možnosť zvolenia akcie po spárovaní sa s bluetooth zariadením (automatické prehratie podcastu).

Veľkým deficitom aplikácie je jej reakčná odozva. Príkladom je akcia pozastavenia prehrávania, kedy prehrávanie pokračuje približne 1.5s po stlačení tlačidla. Player FM neobsahuje možnosť tvoriť vlastné playlisty a ani označovať epizódy ako obľúbené.

AntennaPod

Jedná sa open-source prehrávač s jednoduchým rozhraním. Má však občasné problémy s prehrávaním alebo streamovaním. Taktiež sa po čase používania prestane obnovovať ukazateľ času a užívateľ tak netuší, v akej fáze prehrávania sa nachádza. Problémy zmiznú po reštarte aplikácie. Ďalším problémom je, že neposkytuje žiadnu možnosť ako v zozname epizód filtrovať alebo vyhľadávať epizódy.

Pocket Casts

Možnosti tejto aplikácie sú veľmi bohaté. Disponuje prehľadným menu s možnosťou zobraziť neprehrané epizódy, vytvárať vlastné playlisty, označovať obľúbené epizódy ale aj tvoriť vlastné filtre.

Aplikácia však postráda možnosťou prepínať sa tlačidlami medzi epizódami a jedinou možnosťou je aktuálnu epizódu posúvať v čase. Zmenšený prehrávač obsahuje iba tlačidlo na pozastavenie.

2.6 Špecifikácia požiadavkov

Táto sekcia zhrňuje zistené poznatky z predchádzajúcej analýzy existujúcich riešení. Zároveň kladie dôraz na osobné preferencie, ktoré budú do požiadavkov taktiež zaradené.

Podcasty

Po otvorení aplikácie by užívateľovi nemala aplikácia poskytnúť len strohý zoznam vytvorených podcastov, ako to je v konkurenčných aplikáciách, ale aj dodatočné informácie, ktoré mu uľahčia zorientovanie a samotný výber. Užívateľ by mal mať prehľad o posledných nevypočutých epizódach a o aktuálnosti podcastov.

Epizódy

Pohľady pre zoznamy epizód by mali byť ľahko filtrovateľné. K filtrovaniu patrí zoraďovanie a vyhľadávanie, ktoré by malo byť dosiahnuteľné na jedno kliknutie. Nie každá z analyzovaných aplikácií ponúkala užívateľovi možnosť, ako sa dostať napríklad k stiahnutým

epizódam alebo si zoradiť epizódy podla dĺžky záznamu. Ďalšou dôležitou vlastnosťou je efektívne využitie priestoru na obrazovke, aby užívateľ videľ čo najviac obsahu, či už má mobilné zariadenie s malou uhlopriečkou alebo veľkou.

Dôležitou súčasťou aplikácie je aj tvorba playlistov[11]. Väčšina konkurenčných aplikácií túto možnosť neponúkali. Užívateľ by mal byť schopný rýchlo vytvárať a mazať playlisty a taktiež tieto zoznamy napĺnať.

Prehrávač

Prehrávač je prvok užívateľského rozhrania, ktorý by mal byť dostupný z každej obrazovky a obsahovať najfrekventovanejšie funkcie pre ovládanie prehrávania. Tieto funkcie zahŕňajú prehratie a pozastavenie, prechod na nasledujúcu a predchádzajúcu epizódu, posunutie v čase epizódy. Tieto funkcie sa zdajú byť samozrejmosťou, no nie každý z analyzovaných prehrávačov túto podmienku spĺňal.

Väčšina aplikácií ponúkala prehrávač, ktorý bol na celú obrazovku. Pre vyhľadanie nasledujúcej epizódy alebo iného podcastu bolo treba prehrávač minimalizovať, čo sú zbytočné kroky navyše. Preto by mal byť prehrávač dostupný z každej obrazovky.

Čo sa týka funkčnosti, prehrávač by mal byť schopný epizódy počas napojenia na WiFi aj streamovať a túto možnosť mať dostupnú na jedno kliknutie. Vhodnou funkciou pri počúvaní je aj časovač, ktorý po vybranom čase prehrávanie pozastaví.

Mnohí užívateľia počúvajú podcasty počas cestovania autom, preto je vhodné implementovať aj komunikáciu s Bluetooth zariadeniami, ktorá uľahčí ovládanie za jazdy a poskytne napárovanie multimediálnych tlačidiel auta na ovládanie prehrávača v aplikácií.

Material design[12]

Material design je designovou filozofiou od spoločnosti Google, predstavená v roku 2014. Je to design inšpirovaný skutočnými materiálmi, ktorý má svoju hĺbku, animácie a farby. Dôkazom využitia tohto designu sú aplikácie, ktoré po implementácií tohto designu závratne zvýšili svoju predajnosť. Príkladom je aplikácia **Wego**[13], ktorá zvýšila návratnosť užívateľov o 300%. Využitie tohto designu v aplikácií je samozrejmosťou.

Integrácia služby Audeliver

Zdroj audiozáznamov pre podcasty v aplikácií budú tvoriť dáta zo služby Audeliver, kde sa užívateľ z aplikácie registruje alebo prihlási. Preto bude dôležité prepojiť túto službu s aplikáciou a zaručiť ich obojstrannú komunikáciu. Aby nebol užívateľ nútený pridávať si epizódy a vytvárať podcasty prostredníctvom webovej stránky Audeliveru, budú tieto akcie integrované priamo v aplikácií. Prostredníctvom zdielania z ostatných aplikácií, bude môcť užívateľ epizódu importnúť priamo do podcastu aplikácie.

[11] Playlist - vlastný zoznam multimediálnych záznamov
[12] https://design.google.com/
[13] https://plus.google.com/+AndroidDevelopers/posts/VFm99qWWFHK

Kapitola 3

Návrh riešenia

Nasledujúci text sa zaoberá návrhom aplikácie a jej uživateľského rozhrania. Návrh sa zameriava na splnitelnosť požiadavkov definovaných v sekcií pre špecifikácie požiadavkov 2.6. Budú rozobrané konrétne elementy návrhu spolu s odôvodnením. Pri návrhu bolo dbané na jednoduchosť používania, no dôležitým faktorom bola aj funkčnosť aplikácie.

3.1 Užívateľské rozhranie

Pri návrhu užívateľského rozhrania je vhodné začať prípadom užitia. Ten odhalí, pre koho je aplikácia určená a aké akcie sa od aplikácie očakávajú. Z prípadu užitia sa vytvorí návrh obrazoviek a návrh navigácie medzi obrazovkami. Tieto obrazovku budú v jednotlivých sekciách detailnejšie popísané.

Diagram prípadu užitia

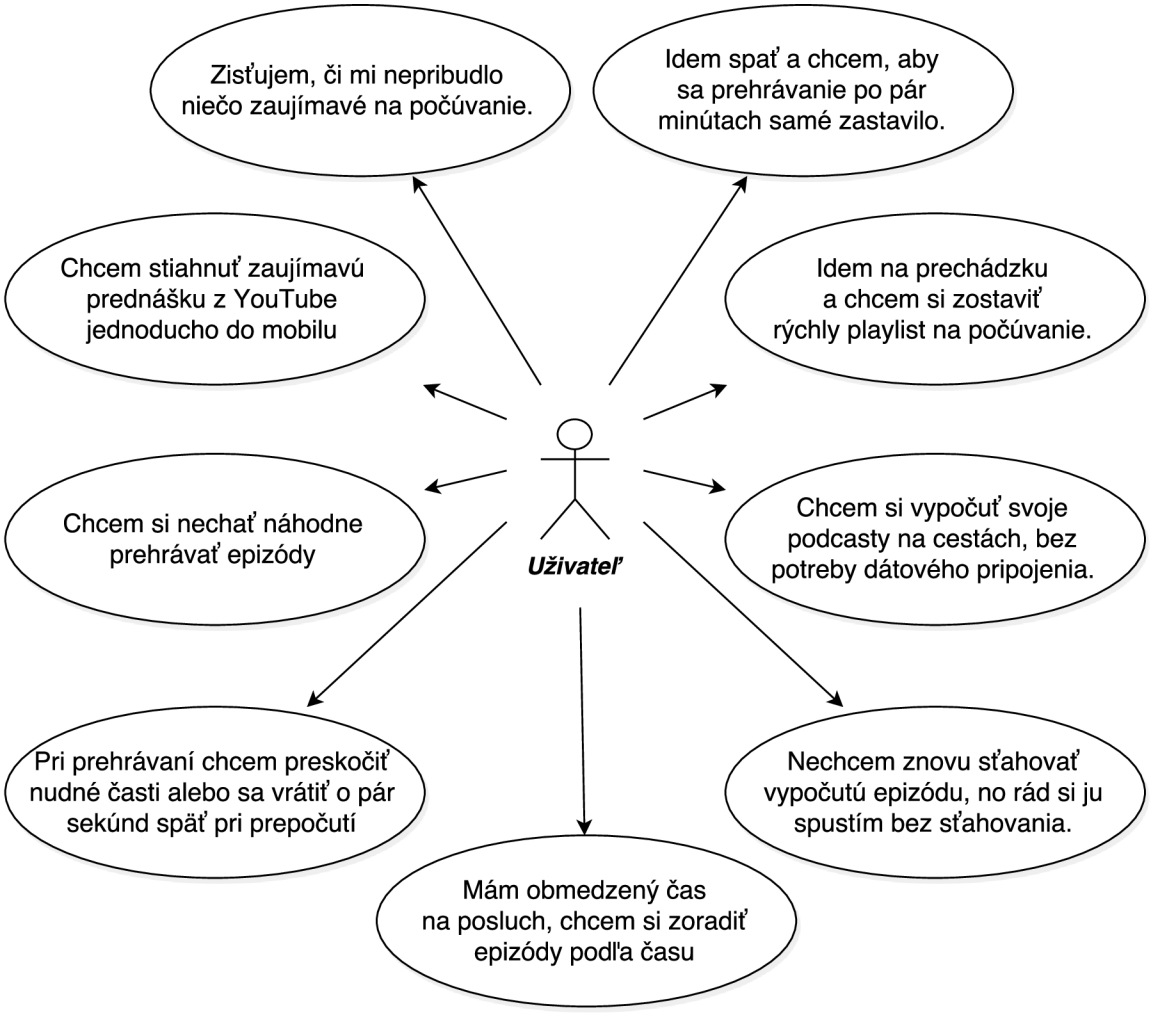
Diagram prípadu užitia zobrazený na obrázku 3.1 poskytuje prehľad hlavných rysov a funkcií aplikácie. Aplikácia bude určená pre bežného užívateľa, ktorý rád počúva podcasty a nechce sa zaťažovať zbytočnými funkciami, no zároveň požaduje jednoduchú a účelnú navigáciu. Užívateľom môže byť taktiež **power user**[1], filtrujúci podcasty podľa seba, ktorého zaujíma velkosť epizódy alebo ktorý si často vytvára rôzne playlisty.

Navigácia v aplikácií

Užívateľská skúsenosť (UX) je pri aplikácií zásadným faktorom. Uživateľská skúsenosť predstavuje ľudské vnímanie a reakcie, ktoré sú výsledkom z používania alebo predpokladaného používania produktu, služby alebo systému[6].

Na obrázku 3.2 je znázornená navigácia v aplikácii od prvého spustenia. Pri prvom spustení sa aplikácia dotáže na prihlasovacie údaje uživateľa alebo ho vyzve k registrácií v službe Audeliver. Samotná registrácia môže potencionálnych uživateľov odradiť od používania. Bolo by teda ideálnejšie, ak by aplikácia ponúkala možnosť prihlásenia sa pomocou demonstračného účtu. Demonstračný účet však zatiaľ služba neponúka. Jediným spôsobom ako začať aplikáciu používať je momentálne vytvorením účtu alebo jeho vlastnením a prihlásením.

[1] Power user - skúsený uživateľ



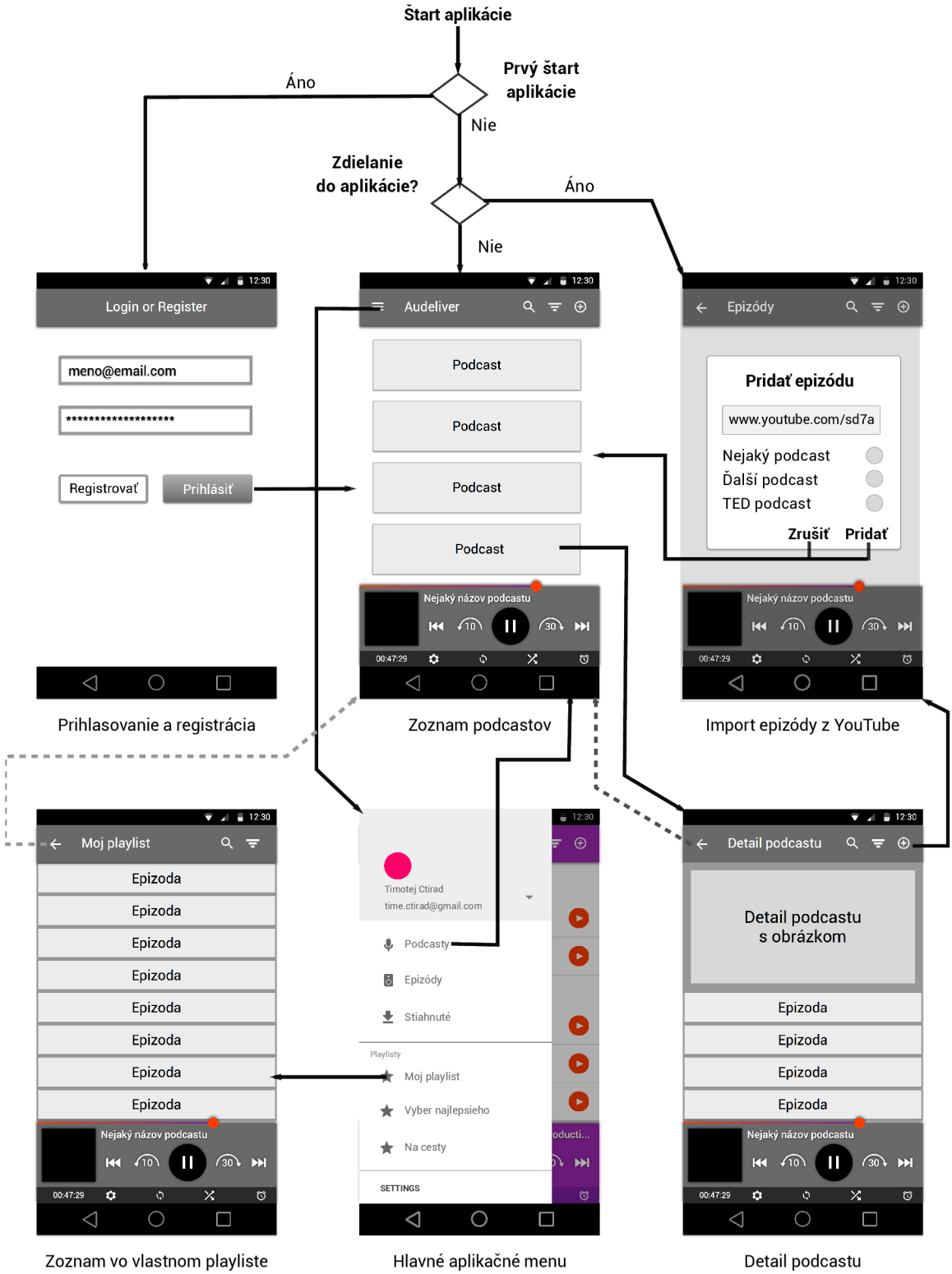
Obr. 3.1: Diagram prípadu užitia

Po prihlásení sa užívateľ dostane do hlavnej obrazovky, ktorá zobrazuje zoznam podcastov. Po stisknutí menu ikony v ľavom hornom rohu sa vysunie hlavné aplikačné menu, ktoré slúži ako hlavná navigácia naprieč celou aplikáciou. Toto menu sa dá vysunúť z každej obrazovky potiahnutím prstu z lavej časti obrazovky, smerom doprava alebo kliknutím na spomínanú ikonu. Toto menu sa však zobrazuje iba v hlavnom zozname podcastov, pretože táto obrazovka slúži ako obrazovka východzia. Pri ďalších obrazovkách sa vo vrchnej časti aplikácie zobrazuje šípka, ktorá vedie naspäť na východziu obrazovku.

Prehrávač

Prehrávač v dolnej časti obrazovky na obrázku 3.3 znázorňuje návrh kompaktného prehrávača, ktorý bude dostupný z každej obrazovky. Prehrávač integruje všetky potrebné ovládacie prvky pre manipuláciu so skladbou.

Príkladom je posuvník, ktorý má rozumné rozmery pre uchytenie a posuv, no zároveň nezaberá zbytočné miesto. U dedikovaných tlačidiel pre posuv v skladbe bol zvolený skok o 30 sekúnd dopredu alebo 10 sekúnd dozadu. Užívateľ tak má možnosť jedným dotknutím preskočiť nudnú pasáž alebo sa vrátiť naspät iba o jednu vetu v rozhovore. Pre jednoduchšiu identifikáciu práve prehrávanej epizódy bol do prehrávaču integrovaný aj obrázok podcastu.

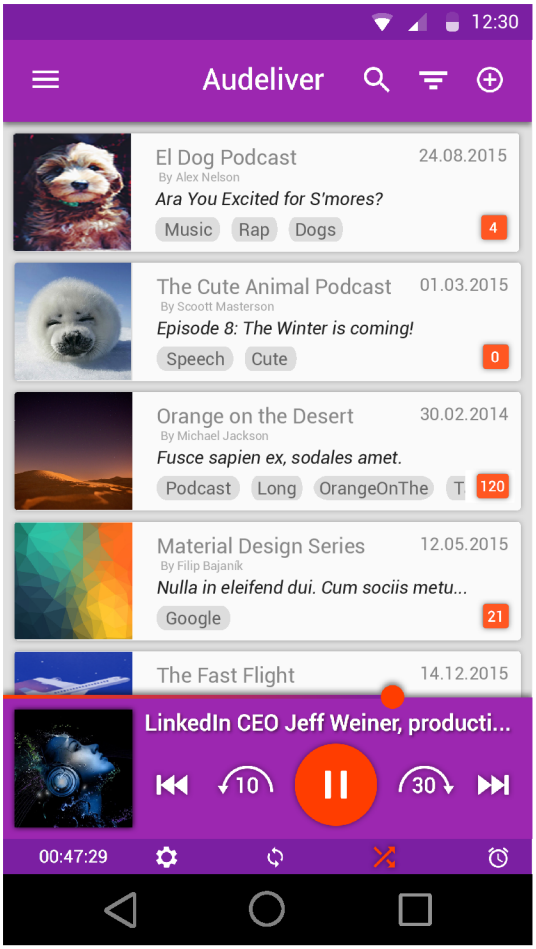


Obr. 3.2: Navigácia v aplikácií

Pod ním sa nachádza aktuálna časová pozícia v skladbe. V neposlednom rade obsahuje prehrávač tlačidlá, ktoré sprístupnia zvukové efekty, časovač, opakovanie alebo zamiešanie skladieb. Užávateľ tak má dostupné všetko v jednom kompaktnom prehrávači a nemusí vstúpiť do zanorených nastavení.

Hlavná obrazovka

Hlavná obrazovka, ktorá je znázornená na obrázku 3.3 obsahuje zoznam odoberaných podcastov spolu s obrázkami daných podcastov. Sústredil som sa hlavne na zobrazenie čo najviac relevantných dát, aby užívateľ nemusel navštevovať detail daného podcastu.



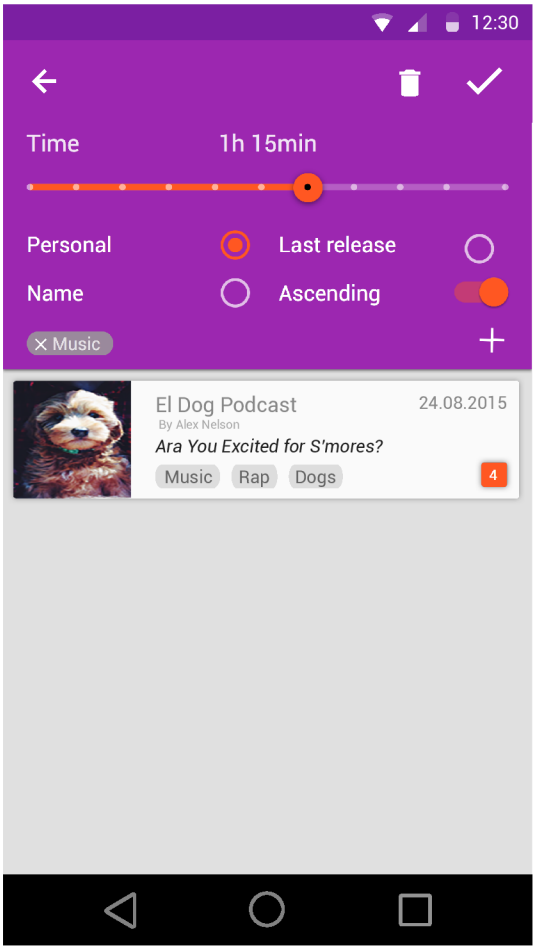
Obr. 3.3: Návrh hlavnej obrazovky

Každá položka podcastu obsahuje samozrejme nadpis a podnadpis podcastu. Je však vhodné zobraziť aj počet neprehraných epizód, aby užívateľ videl hneď z hlavnej obrazovky, či mu nepribudla nová epizóda. V položke podcastu je taktiež možné vidieť názov a dátum poslednej epizódy. Uznal som však za vhodné, že pre jednoduchšiu orientáciu pri výbere podcastu sa užívateľovi môžu hodiť kategórie, pod ktoré spadá podcast. Tieto kategórie sú zobrazené vo forme značky v tvare elipsy.

Filtrovanie

Z detailu podcastov alebo zo zoznamu epizód sa užívateľ dostane k filtrovaniu. Pri zozname podcastov by sa filtrovali hlavne podcasty, ale je možné, že sa táto funkcionalita obmedzí iba na filtrovanie epizód, nakoľko bežný užívateľ neodoberá tak veľké množstvo podcastov, aby potreboval filtrovanie.

Pri filtrovaní je možnosť vlastného zoradenia, zoradenia podľa dátumu posledného vydania a podľa mena. Všetky tieto možnosti sa dajú prepínačom prepnúť do režimu zoraďovania vzostupného alebo zostupného. Filtrovanie obsahuje aj voľbu filtrovania podľa značiek, ktorých môže byť neobmedzené množstvo.

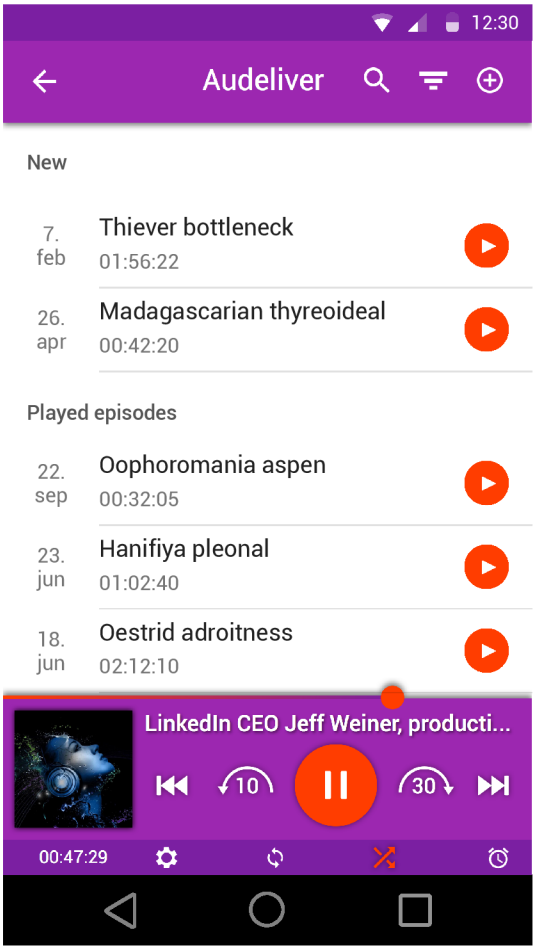


Obr. 3.4: Návrh filtrovania na hlavnej obrazovke - zobrazenie podcastov

Zoznam detailu podcastov

Zoznam epizód bol navrhnutý, aby obsahoval základné informácie o epizódach a akcie akými sú sťahovanie a streamovanie. Medzi základné informácie, ktoré zaujímajú používateľov patrí dátum zverejnenia, názov epizódy a trvanie epizódy. Epizódy sú členené do dvoch kategórií. Prvou kategóriou sú epizódy nové, ktoré ešte neboli užívateľom prehrané a druhú kategóriu tvoria epizódy, ktoré boli manuálne označené ako prehrané alebo ich užívateľ naozaj prehral. Užívateľ má možnosť epizódu stiahnuť alebo ju začať streamovať online. Po

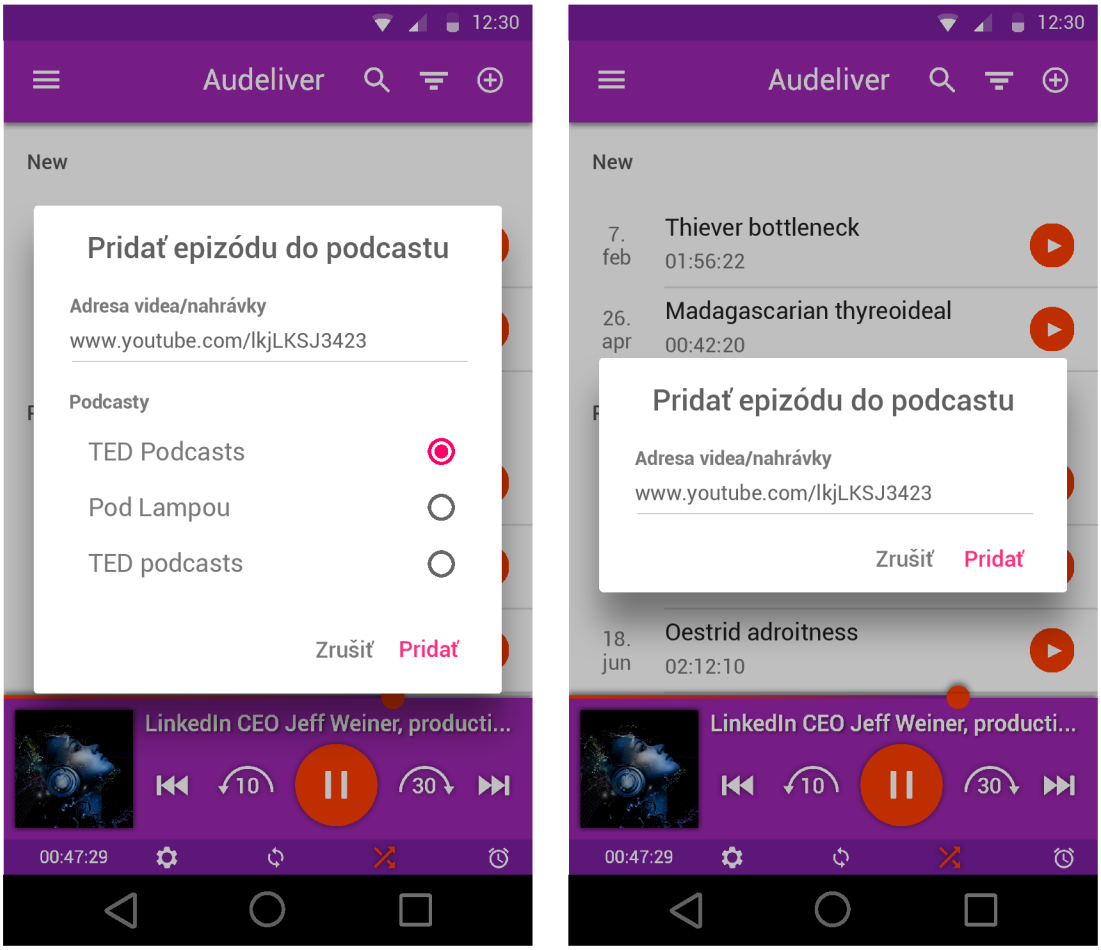
kliknutí na názov epizódy sa užívateľovi zobrazí dlhší popis epizódy.



Obr. 3.5: Návrh detailu podcastov

Pridávanie podcastov

Existujú dva hlavné scenáre pridávania epizód do aplikácie priamo zo zariadenia. Prvý scenár je ten, kedy sa uživateľ nachádza napríklad v aplikácií YouTube. Táto aplikácia, ako aj ostatné aplikácie, ktoré prezentujú audiovizuálne dáta, obsahuje tlačidlo na zdielanie. Týmto tlačidlom sa zvyčajne zdiela odkaz na zdroj. Pri YouTube je zdrojom správa obsahujúca odkaz na video. Otvorí sa okno s výberom, do akej aplikácie sa má táto správa zdielať. Pri výbere popisovanej aplikácie sa aplikácia spustí a zobrazí sa dialógové okno, ktoré je znázornené na obrázku 3.6a. Pokiaľ zdielaná správa obsahuje validný odkaz, políčko na vkladanie adresy sa znefunkční. Pod adresou sa následne zobrazí výber, do akého podcastu sa má epizóda vložiť. Po potvrdení sa vytvorí dočasný záznam, oznamujúci uživateľovi, že epizóda sa spracováva na serveri. Po úspešnom spracovaní dostane uživateľ notifikáciu o úspešnosti a epizódu je možné stiahnuť alebo rovno prehrať.



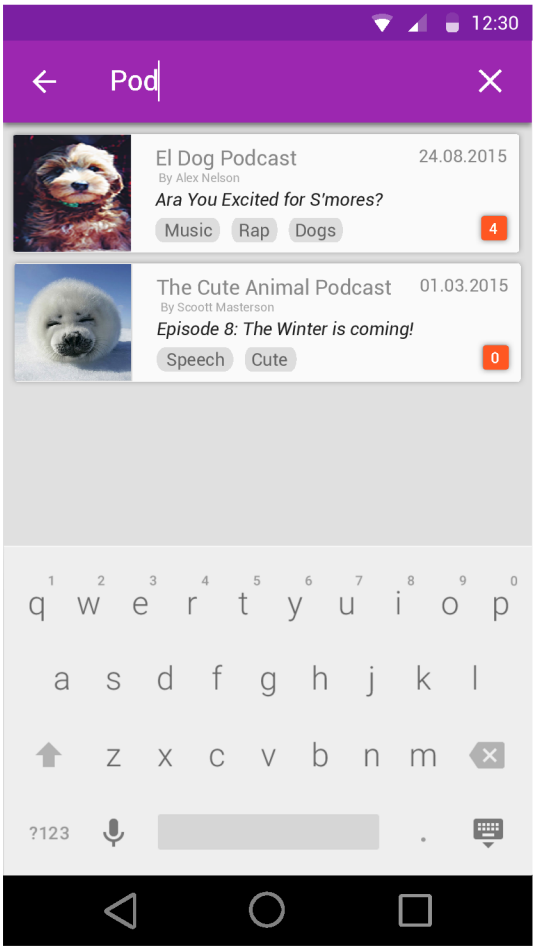
(a) Pridanie epizódy z YouTube app. zdielaním (b) Pridanie epizódy priamo z detailu podcastu

Obr. 3.6: Návrh pridávania epizódy z Youtube

Druhou možnosťou je pridanie novej epizódy z aplikácie. Ak uživateľ klikne z detailu podcastu na ikonu znázorňujúcu znak *plus*, otvorí sa dialógové okno s poľom pre URL adresu. Ak sa v systémovej schránke nachádza správa, ktorá obsahuje validný odkaz, tento odkaz bude automaticky vložený do otvoreného dialógového okna. V inom prípade je potrebné adresu vyplniť ručne. Po potvrdení dialógového okna sa vyvolá akcia, ako v prvom scenári.

Vyhľadávanie

Vyhľadávanie bolo navrhnuté tak, aby v unifikovanom textovom poli mohol používateľ zadať akúkoľvek textovú informáciu, týkajúcu sa podcastov. Môže teda vyhľadávať v názvoch, podnázvoch, popisoch epizód alebo podcastov a kategóriách. Vyhľadávanie je dostupné na obrazovke zoznamu epizód, ale aj na obrazovke detailu podcastu.



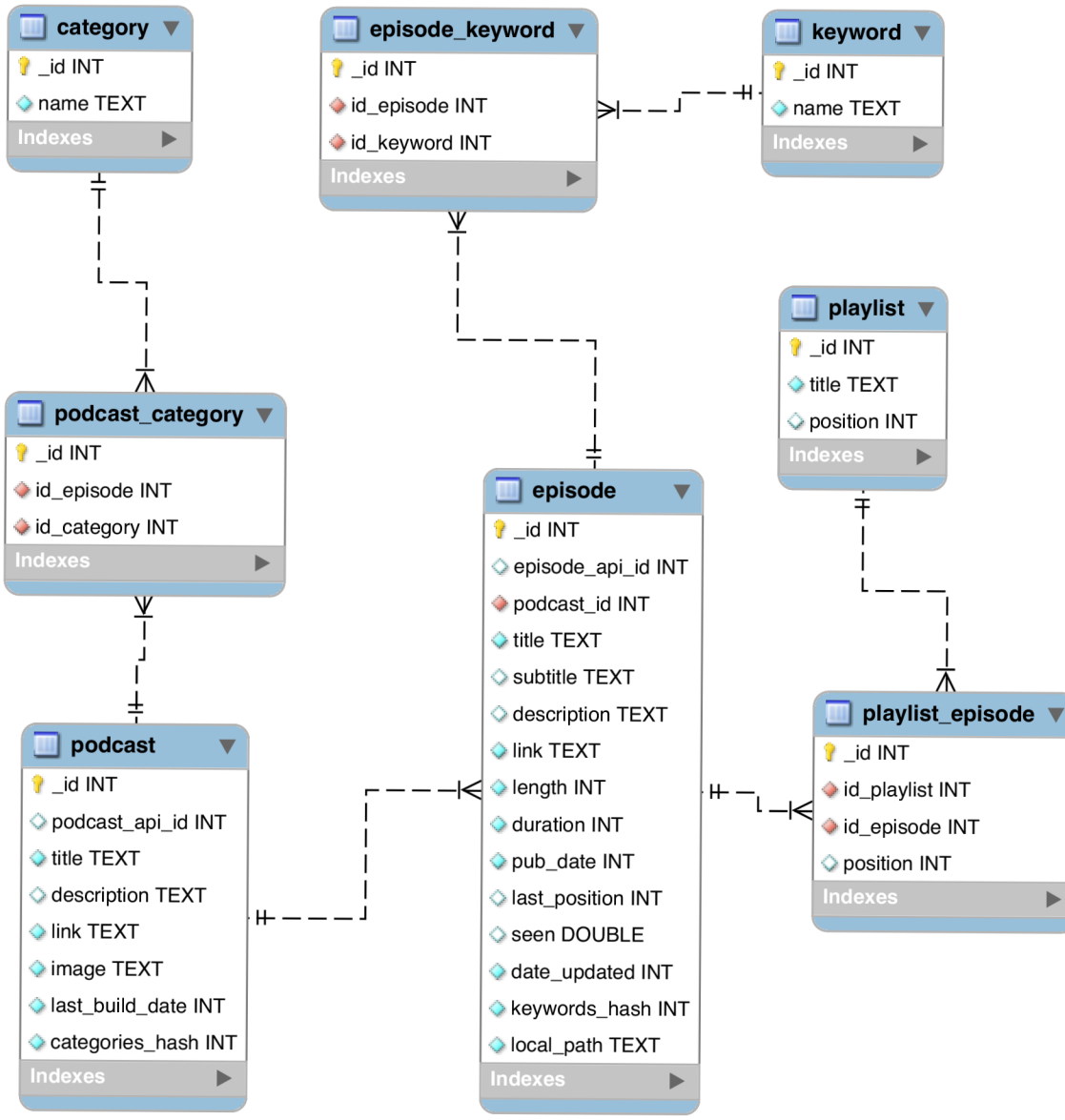
Obr. 3.7: Návrh vyhľadávania na hlavnej obrazovke

3.2 Databáza

Databáza vznikla primárne z dátových štruktúr, ktoré sú popísané v sekcií 2.2. Ako je vidieť na obrázku 3.8, dátovy model má vo výsledku hlbšiu hierarchickú štruktúru, než štruktúra XML feedu 2.2. Zo štruktúry podcastu 2.2 bola vyčlenená entita[2] **category**, spolu s väzobnou tabulkou **podcast_category**. Rovnako sa vyčlenila aj entita **keyword** s väzobnou tabuľkou **episode_keyword**.

Aby bolo možné v aplikácií ukladať, či je epizóda vypočutá alebo nevypočutá, k entite **episode** bol pridaný nový atribút **seen**. Ďalším pridaným atribútom je lokálna cesta k stiahnutému súboru (local_path) a identifikátor epizódy z Audeliver API (episode_api_id). Pri entite **podcast** sa taktiež nachádza unikátny identifikátor podcastu z API (podcast_api_id). Význam atribútov **keywords_hash** a **categories_hash** bude popísaný v kapitole 5.

[2]Entita je prvok reálneho sveta (napr. človek, zviera, vec), ktorá je definovaná svojimi vlastnosťami. Tie sa väčšinou považujú za atribút (napr. meno, stav, velkosť) [4].



Obr. 3.8: Výsledný ER Diagram návrhu databázy

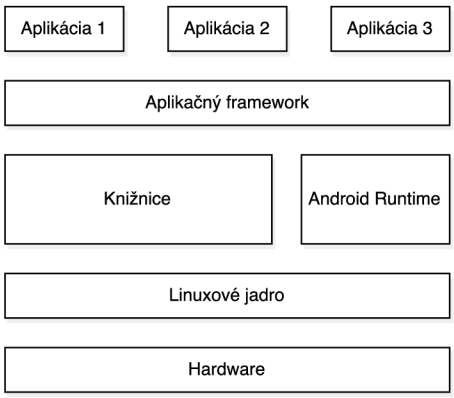
Kapitola 4

Vývoj na platforme Android

Kapitola popisuje základné teoretické princípy vývoja pre platformu Android, architektúru a popis základných funkčných a grafických blokov systému. Tieto časti sú neskôr využité pri návrhu a implementácií. Taktiež popisuje grafické rozhranie. Informácie, ktoré budú popisované, sú čerpané z oficiálnej vývojárskej dokumentácie [1] a z knihy [7].

4.1 Popis architektúry

Dôležitou súčasťou pri vývoji aplikácií pre Android je oboznámenie sa s jeho architektúrou. Architektúra je v členená na viacero blokov, ktoré sú znázornené na obrázku 4.1.



Obr. 4.1: Popis architektúry Android

- **Linuxové jadro (Kernel)** - Zabezpečuje správu procesov, pamäti, základnú sieťovú vrstvu, ovládače, ale aj zabezpečenie systému, správa napájania a vstupno-výstupné operácie. Z dôvodu rozmanitosti harwaru však bola na úrovni jadra vytvorená vstrva **HAL**[1], ktorá reprezentuje rozhranie medzi **Android frameworkom** a špecifickým typom hardwaru.
- **Knižnice** - Zabezpečujú priamy prístup ku komponentom systému, kompozíciu grafického výstupu a výsledné zobrazovanie na obrazovku. Knižnice podporujú aj prácu s multimédiami. Grafický 2D a 3D obsah zabezpečuje knižnica **OpenGL ES**[2].

[1] HAL - https://source.android.com/devices/#Hardware_Abstraction_Layer
[2] OpenGL ES - http://developer.android.com/guide/topics/graphics/opengl.html

- **Android Runtime** - Každá aplikácia sa spúšťa ako samostatný proces inštancie virtuálneho stroja **ART**. ART využíva **ahead-of-time** kompiláciu, čo znamená, že pri inštalácií aplikácie sa skompilovaný **Java bytekód** prevedie na natívne inštrukcie, ktoré po spustení vykonáva **virtuálny stroj**.
- **Aplikačný framework** - Aplikačný framework poskytuje programátorovi znovupoužitelné fragmenty kódu, ktoré obsahujú implementáciu ovládacích prvkov systému. Framework je napísaný v Jave a poskytuje základné služby systému.
- **Aplikácie** - Aplikačná vrstva obsahuje aplikácie, ktoré sú implementované za pomoci aplikačného frameworku. Príkladom môžu byť aplikácie Správy, Telefón, Kontakty, Google Chrome.

4.2 Popis komponentov

Podkapitola obsahuje súhrn funkčných komponentov, ktoré poskytuje **Android Framework**. Tieto komponenty zabezpečujú potrebnú funkčnosť a tvoria základný kameň aplikačného jadra.

Súbor Manifest

Manifest je hlavný konfiguračný súbor aplikácie. Definuje hlavné komponenty aplikácie a taktiež oprávnenia, ktoré sú vyžadované od užívateľa pri inštalácií aplikácie. Manifest definuje taktiež bežiace služby aplikácie.

Poskytovateľ obsahu (Content provider)

Poskytovateľ obsahu sprístupňuje rozhranie pre získavanie, ukladanie alebo zmenu dát a to nie len v rámci jednej aplikácie ale aj medzi viacerými aplikáciami. Primárne sa poskytovatelia obsahu využívajú na prístup k interným databázam, no záleží na implementácií, ako je daný poskytovateľ obsahu implementovaný.

Poskytovateľ obsahu musí byť registrovaný v konfiguračnom manifeste a k vytvoreniu dotazu pre prístup k dátam sa používa **URI**[3] vo formáte: `content://authority/path/id`.

Odberateľ broadcastu (Broadcast receiver)

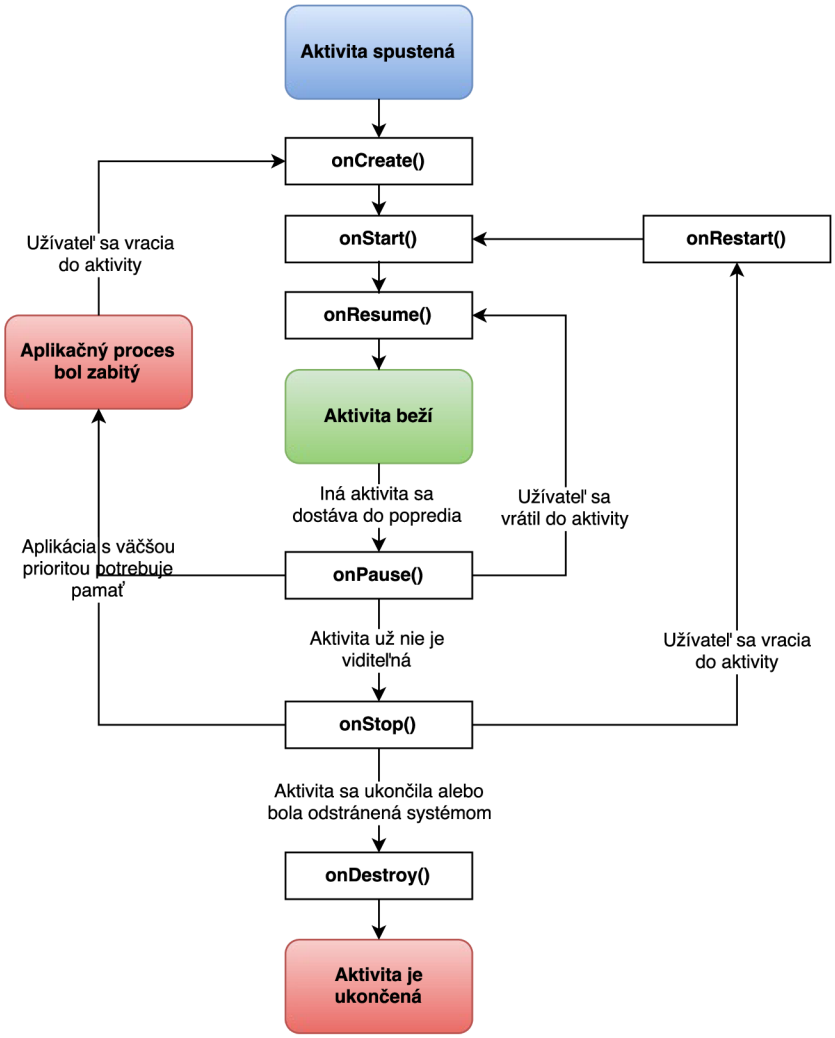
Aplikácie sú schopné generovať udalosti, ktoré sú smerované cez **Broadcast** do vysielania. Ostatné aplikácie majú **prijímače (receivers)**, ktorými sú schopné tieto udalosti odchytiť a vyvolať vlastnú akciu, buď spustením zvolenej aplikácie alebo notifikačným centrom.

Príkladom môže byť služba prehrávača na pozadí, ktorá po dohraní skladby vytvorí udalosť, ktorá oznamuje, že skladba skončila. Aplikácia, ktorá načúva prostredníctvom odberateľa broadcastu, zachytí túto udalosť a aktualizuje uživateľské rozhranie.

Aktivity

Aktivita je hlavná trieda, ktorá sa užívatelovi zobrazí po spustení aplikácie. Aplikácia sa môže skladať z viacerých aktivít. Usporiadanie aktivít je **hierarchické**.

[3] URI - http://www.faqs.org/rfcs/rfc2396.html



Obr. 4.2: Životný cyklus aktivity

Ak chceme realizovať určitú akciu v jednom zo stavov, je potrebné preťažiť jednu z metód na obrázku 4.2. V prípade, že otočíme displej zariadenia, aktivita prejde do stavu `onPause()` a následne do `onStop()`, kedy sa dealokuje pozícia v zozname alebo práve editované užívateľské dáta. Tomu môžme predísť, ak preťažíme napríklad metódu `onStop()` a uložíme uživateľské dáta, ktoré následne v stave `onCreate()` obnovíme.

Fragmenty

Fragmenty sa vkladajú do aktivít a majú vlastný rozšírený cyklus aktivity. Príkladom použitia môže byť zobrazenie dvoch fragmentov pri tablete *(menu + obsah)* alebo integrácia prehrávača do ostatných aktivít a zabránenie zbytočnej redundancie.

Služby

Služby poskytujú možnosť behu dlhotrvajúcej operácie na pozadí. Nezaťažuje sa hlavné vlákno aplikácie, a tak zostáva responzívnosť aplikácie plynulá. Služby nepotrebujú užívateľskú interakciu a dokážu pristupovať k aplikačným dátam alebo komunikovať medzi

sebou.

Príkladom služby môže byť prehrávanie hudby. Služba sa môže naplánovať a spustiť alebo spúšťať periodicky podla naplánovania alebo môžu byť spustené na vyžiadanie v rámci aktívnej aplikácie.

AIDL

Niektoré služby pracujú nezávysle, uživateľ ich spustí, služba sa vykoná a sama skončí. Iné služby však môžu bežať samostatne a čakať na požiadavky od užívateľa. Jedna z možností ako môže služba príjmať požiadavky od užívateľa je vzdialené rozhranie **AIDL**[4].

Samotné rozhranie neobsahuje implementáciu, služba teda musí toto rozhranie implementovať a aplikácia následne využíva iba dané rozhranie služby.

Príkladom využitia môže byť rozhranie služby pre ovládanie prehrávania hudby. Služba beží na pozadí a aplikácia si vyžiada prostredníctvom AIDL rozhrania zastavenie skladby. Služba beží ďalej na pozadí aj po tom, čo sa aplikácia ukončí.

Možnosti uloženia dát

Pre uloženie perzistentných dát poskytuje Android viacero možností:

- **Zdielané nastavenia** - privátne dáta aplikácie uložené vo forme **kľúč-hodnota**; vhodné pre uloženie uživateľských nastavení aplikácie.
- **Interné a externé úložisko** - privátne dáta aplikácie alebo voľne dostupná zdielaná pamäť pre užívateľa.
- **SQLite databáza** - ukladanie privátnych štruktúrovaných dát; vhodné napríklad pre uchovanie informácií o skladbách.
- **Sieťové úložisko** - vlastný alebo firemný server; vhodné pre ukladanie uživateľských dát, ktoré sú následne zdielané medzi zariadeniami.

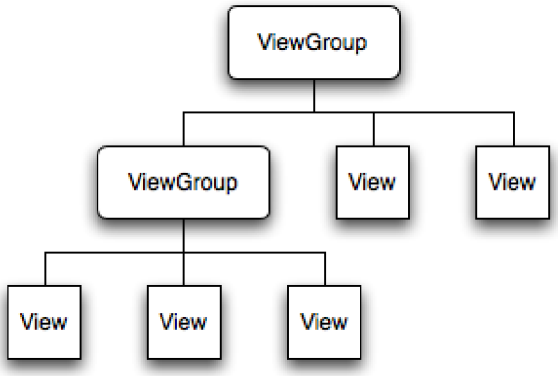
4.3 Grafické rozhranie

Sekcia popisuje základné, ale aj pokročilé grafické prvky systému Android, ktoré sú potrebné pri návrhu užívateľského rozhrania, ale aj samotnej funkčnosti. Tieto elementy sa sádzajú pomocout značkovacieho jazyka **XML**[5].

Pohľady (Views)

Pohľady predstavujú základné triedy pre tvorbu užív. rozhrania. Pohľady môžu byť členené do tzv. **View Groups** a môžu by ť ľubovolne usporiadané v hierarchií tak, ako ukazuje napr. obrázok 4.3. Od pohľadov sú odvodené aj základné grafické prvky akými sú napríklad *TextView*, *ImageButton*, *ProgressBar* a ďalšie...

[4] AIDL (Android Interface Definition Language) - http://developer.android.com/guide/components/aidl.html
[5] XML - http://www.w3.org/TR/xml/



Obr. 4.3: ViewGroups

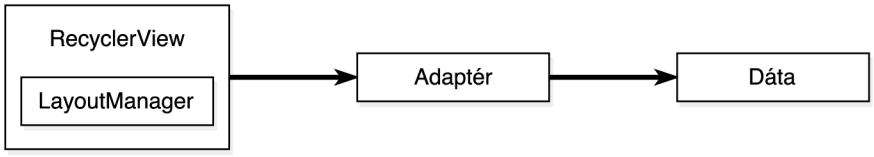
Kontajnery (Layouts)

Kontajnery slúžia k rozmiestneniu grafických prvkov a sú odvodené od triedy **ViewGroup**. Pre jednoduchosť spomeniem len zopár použitých.

- **LinearLayout** - umožňujú rozmiestnovať vnorené prvky vo vybranej orientácií *(horizontálne, vertikálne)*. Umožňujú taktiež vnoreným prvkom definovať váhu rozlošenia a tak určiť ich relatívnu šírku v rámci layoutu alebo určiť ich zarovnanie.
- **RelativeLayout** - umožňujú relatívne usporiadanie medzi vnorenými prvkami. Prvky, voči ktorým sa chcú pozicovať prvky ostatné, musia mať unikátny identifikátor.
- **FrameLayout** - je principiálne jednoduchý layout, ktorý zvyčajne obsahuje len jeden vnorený prvok. Pri väčšom počte sa prvky prekrývajú, čo môže byť využité napríklad pri stále viditeľnom prehrávači audia, ktorý sa môže animáciou roztiahnuť na celú obrazovku a zakryť tak informácie za ním.

Pokročilejšie grafické prvky

Medzi pokročilejšie grafické prvky patrí **RecyclerView**, ktorého diagram je znázornený na obrázku 4.4. Poskytuje možnosti, ako efektívnym spôsobom zobrazovať veľké množstvo dát. RecyclerView obmedzuje počet pohľadov, ktoré sa načítavajú a pri posúvaní zoznamu recykluje už neviditeľné položky. Tie položky sa využívajú na načítanie nových dát.



Obr. 4.4: Diagram použitia RecyclerView

Obrázok 4.4 popisuje použitie RecyclerView. **LayoutManager** zaisťuje rozmiestnenie prvkov v pohľade. Na zobrazenie samotných dát je potrebné implementovať **Adaptér**, podľa rovnakomenného návrhového vzoru[5], ktorý prepojí položky zoznamu dátami.

4.4 Využité knižnice

Aplikácia využíva natívny Android framework, avšak boli využité aj knižnice, ktoré uľahčujú implementáciu rutinných operácií alebo prácu s určitými komponentami. Tieto knižnice sú stabilné a väčšina z nich je kompletne pokrytá testami a sú teda vhodné do produkcie.

Glide[6]

Glide je veľmi rýchla a efektívna open source knižnica, ktorá poskytuje možnosti manažovania a sťahovania obrázkov. Glide zahŕňa a implementuje akcie akými sú dekódovanie obrázkových dát, dočasné a efektívne ukladanie na disk, do cache alebo škálovanie.

ButterKnife[7]

Knižnica ButterKnife uľahčuje získavanie referencií na pohľady grafického uživateľského rozhrania. Tieto prvky grafického rozhrania mapuje na inštancie tried za použitia **anotácií**[8]. Knižnica teda odstieňuje vývojára od písania zbytočného kódu.

Gson[9]

Gson je knižnica, ktorá pochádza priamo od Google. Slúži na serializáciu a deserializáciu Java objektov do formátu Json a naspäť. Knižnica je využívaná pri vlastnej implementácií triedy na ukladanie slovníkových uživateľských dát **MyPreferenceManager**. Metódy, ktoré využívajú Gson, serializujú dáta predstavujúce napríklad zoznam aktuálne prehrávaných epizód. Tento zoznam sa pri ukončení aplikácie serializuje a uloží vo formáte kľúč-hodnota. Tieto dáta je možno pri obnovení sedenia znovu deserializovať.

Retrofit[10]

Retrofit je REST klient vhodný na použitie pre Android a Javu. Uľahčuje jednoduché prijímanie a odosielanie dát vo formáte Json (alebo iných štrutúrovaných dát) k službám, ktoré využívajú REST API. Retrofit môže byť nakonfigurovaný, aby používal vlastný konvertor dát na serializáciu a deserializáciu dát. Pre spracovanie HTTP požiadavkov používa Retrofit typicky knižnicu OkHttp.

Crashlytics[11]

Crashalitics je knižnica používaná na získavanie štatistických dát v reálnom čase. V prípade chyby dokáže vývojárovi okamžite odoslať spätnú väzbu, ktorá obsahuje detailný popis. Ten zahŕňa taktiež výpis všetkých aktívnych vlákien a riadky kódu, pri ktorých bola chyba spôsobená. Crashalitics oznamuje počet aktívnych užívateľov, počet inštalácií, počet chýb a štatistiky generuje v prehľadných grafoch.

[6] Glide - https://github.com/bumptech/glide
[7] ButterKnife - https://github.com/JakeWharton/butterknife
[8] Anotácia - https://docs.oracle.com/javase/tutorial/java/annotations/basics.html
[9] Gson - https://github.com/google/gson
[10] Retrofit - https://square.github.io/retrofit/
[11] Crashlytics - https://try.crashlytics.com

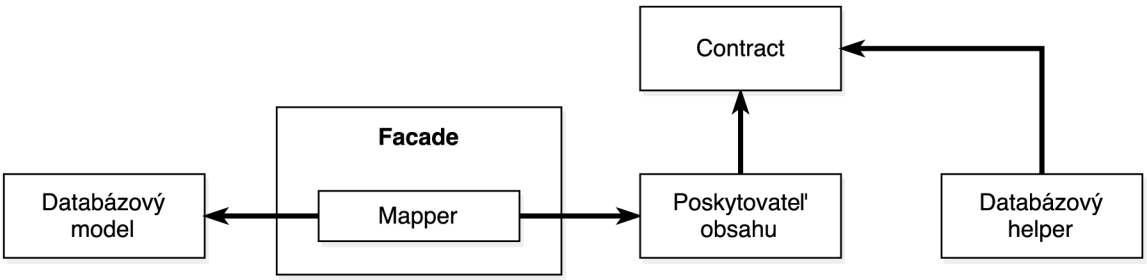
Kapitola 5

Implementácia

Implementácia aplikácie prebiehala vo vývojovom prostredí **Android Studio**[1], ktorý je primárne určený na vývoj pre mobilnú platformu Android. Implementácia vychádza z návrhu riešenia, ktoré bolo popísané v kapitole 3. Samotná kapitola je venovaná návrhu databázy, ktorá bude slúžiť pre uchovanie dát zo serveru služby Audeliver. Jej integrácia a synchronizácia bude taktiež popísaná. V ďalších sekciách bude vysvetlená implementácia prehrávania, streamovania a sťahovania epizód podcastu a implementácia časovača, spolu s princípom označovania epizód a vytvárania mediálnych notifikácií. V neposledom rade sa na konci kapitoly popíšu použité knižnice a základné metriky kódu.

5.1 Databáza

Ako databázový relačný systém bol zvolený **SQLite**, ktorý je natívne integrovaný v Android frameworku. Umožňuje najväčšiu flexibilitu, no vyžaduje implementáciu obslužného kódu, ktorý je obsiahly. Implementáciu databáze znázorňuje obrázok 5.1. Proces návrhu začal nadefinovaním **databázového modelu**. Tento model bol popísaný v **contracte**. Contract definuje konštanty, ktoré pomáhajú pracovať aplikáciám alebo častiam aplikácií s menami položiek databáze, systémovými akciami alebo inými funkciami **content provideru** 4.2.



Obr. 5.1: Konceptuálny diagram implementácie databázy

Po vytvorení databázového modelu bolo potrebné implementovať tzv. **databázový helper**. Tento helper inicializuje stav databáze alebo ho pri zmene verzií aktualizuje. Pre efektívny prístup k dátam aplikácie bola použitá vlastná implementácia **poskytovateľa obsahu**, ktorá tvorí rozhranie pre efektívne dotazovanie nad dátami, nie len v rámci ap-

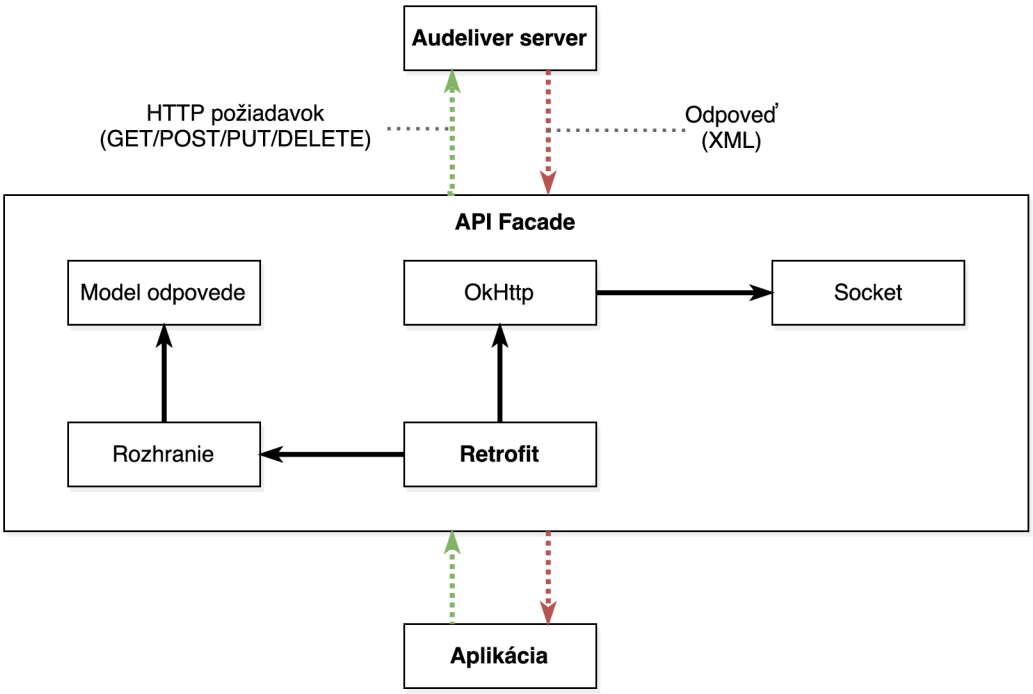
[1] Android Studio - https://developer.android.com/studio/index.html

likácie, ale aj pre **widgety**[2] a služby na pozadí. V prípade označenia poskytovateľa za externý, môže vývojár poskytnúť aplikačné dáta aj externým aplikáciám. Implementácia pokrýva metódy na dotazovanie sa nad dátami, mazanie, vytváranie a editáciu.

Pre uľahčenie práce s databázou bola vytvorená tzv. **Facade**[3] trieda. Výsledkom je trieda, ktorá obaľuje komplexnejšie metódy s SQL dotazmi a poskytuje jednoduché rozhranie. Výsledky dotazov sa mapujú na triedy dátových modelov pomocou triedy **Mapper**. Metódy vracajú inštancie jednotlivých databázových modelov alebo ich kolekcie. Vďaka tejto triede nebolo potrebné používať dotazovací jazyk SQL, ale len jednoduché rozhranie.

5.2 Integrácia služby Audeliver

Pre integráciu služby Audeliver bola použitá knižnica **Retrofit**, popísaná v sekcií 4.4. Tá odstieňuje vývojára od vlastnej implementácie sieťovej komunikácie. Ako adaptér pre sieťovú komunikáciu pracujúcu na nižšej aplikačnej vrstve bola zvolená knižnica **OkHttp**[4]. Zjednodušený konceptuálny diagram implementácie tried sa nachádza na obrázku 5.2.



Obr. 5.2: Konceptuálny diagram implementácie komunikácie aplikácie so službou Audeliver. Plné čiary predstavujú závislosti medzi časťami aplikáčného rozhrania. Prerušované čiary znázorňujú komunikáciu a jej smer.

Typy správ posielané pri komunikácií so službou Audeliver boli popísané v sekcii 2.3. Pri implementácií konkrétneho riešenia bolo treba nadefinovať rozhranie pre komunikáciu. Toto rozhranie predstavuje metódy, ktoré využívajú špeciálne anotácie z knižnice Retrofit. Tie zabezpečujú, že sa pri komunikácií vyberie vhodná metóda HTTP dotazu (GET, POST, PUT, DELETE). Zároveň je treba vytvoriť model odpovedí, ktoré predstavujú štruktúru,

[2]Widget - malá aplikácia, ktorú si užívateľ môže umiestniť na domovskú obrazovku.
[3]Facade pattern - Návrhový vzor: https://sourcemaking.com/design_patterns/facade
[4]OkHttp - http://square.github.io/okhttp/

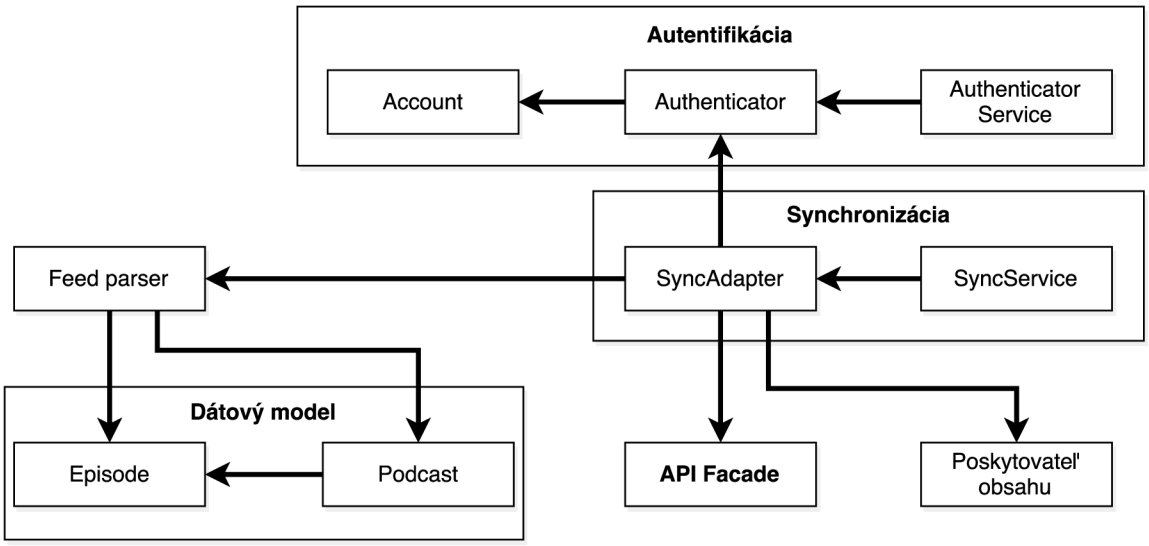
posielanú ako odpoveď vo forme XML. Tieto modely využívajú taktiež anotácie z knižnice Retrofit a definujú hierarchickú štruktúru odpovede.

Ako neúspešná odpoveď zo serveru sa vracala odpoveď s kladným **respond** kódom (HTTP 200), no štruktúra odpovede bola zmenená. Preto pri definícií modelov museli byť všetky anotácie atribútov označené ako nepovinné a ku očakávaným kladným atribútom pridané atribúty **status** a **message**. Tieto atribúty obsahovali kód a správu chyby.

5.3 Synchronizácia

Po spustení synchronizácie sa kontroluje, či je uživateľ prihlásený. Slúži na to služba **AuthenticatorService**, ktorá kontroluje, či je v systémových účtoch vytvorený účet pre službu Audeliver. Ak uživateľ prihlásený nie je, alebo sú jeho prihlasovacie údaje nesprávne, aplikácia požiada o znovuprihlásenie. Táto služba je navrhnutá pomocou vzoru **SyncAdapter**[5]. Ten zabezpečí bezpečné uchovanie uživateľských prihlasovacích údajov.

Ak je užívateľ prihlásený, trieda **FeedParser** znázornená na obrázku 5.3 požiada o odkazy k XML feedom. Na získanie odkazov využíva triedu **API Facade** popísanú v sekcií 5.2. Ak sa získanie odkazov nepodarí, uživateľ bude notifikovaný prostredníctvom notifikačnej komponenty **Toast**[6].



Obr. 5.3: Konceptuálny diagram implementácie synchronizácie znázorňujúci závislosti medzi triedami.

Na spracovanie (parsovanie) XML feedu sa používa trieda **Feed parser**. Táto trieda najskôr spracuje základné údaje o podcaste zo štruktúry, popísanej v tabuľke 2.2. Ak už daný podcast existuje, porovná sa kontrolný súčet (**hash**), ktorý je vytvorený z textovej reprezentácie podcast atribútu **category**. Ušetria sa tak veľmi drahé databázové operácie, ktoré prebiehajú s internou SQLite databázou. Služba nezaťažuje hlavné vlákno, v dôsledku čoho je responzívnosť aplikácie zachovaná. Pre samotné ukladanie dát do databázy sa využíva poskytovateľ obsahu.

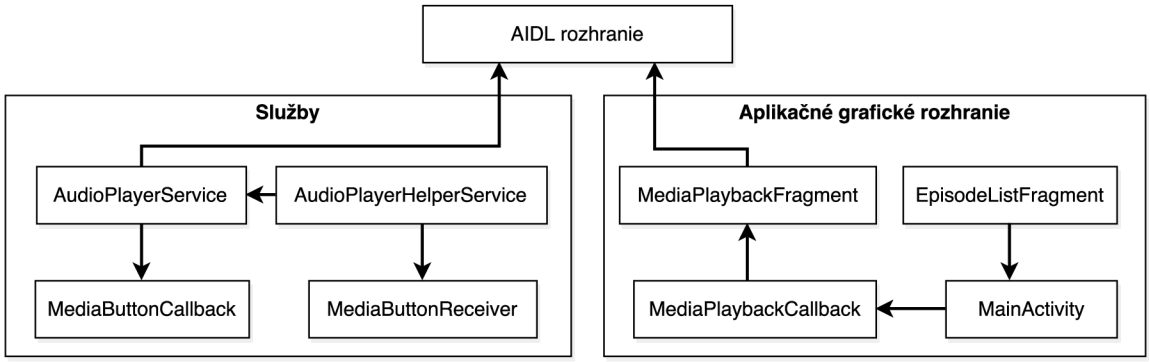
[5]SyncAdapter - http://developer.android.com/training/sync-adapters/creating-authenticator.html
[6]Toast - http://developer.android.com/guide/topics/ui/notifiers/toasts.html

Periodická synchronizácia je vykonávaná prostredníctvom služby **SyncService**, ktorá sa spúšťa každé 3 hodiny. Táto hodnota je napevno nastavená, avšak zvažuje sa jej nastaviteľnosť prostredníctvom uživateľského nastavenia.

5.4 Prehrávanie a streaming

Prehrávanie podcastov je implementované ako služba **AudioPlayerService** bežiaca na pozadí. Tá je naviazaná na službu **AudioPlayerHelperService**, ktorá obsluhuje správy odosielané zo zamknutej obrazovky, notifikácií, náhlavnej súpravy, smart-hodiniek alebo bluetooth zariadení prostredníctvom triedy **MediaButtonReceiver**.

Služba AudioPlayerHelperService sa spustí v momente, kedy je spustená hlavná aktivita aplikácie. V prípade, že je aplikácia ukončená a aktuálne sa neprehráva žiadne audio, služba sa sama ukončí. Konceptuálny diagram tejto architektúry je popísaný na obrázku 5.4.



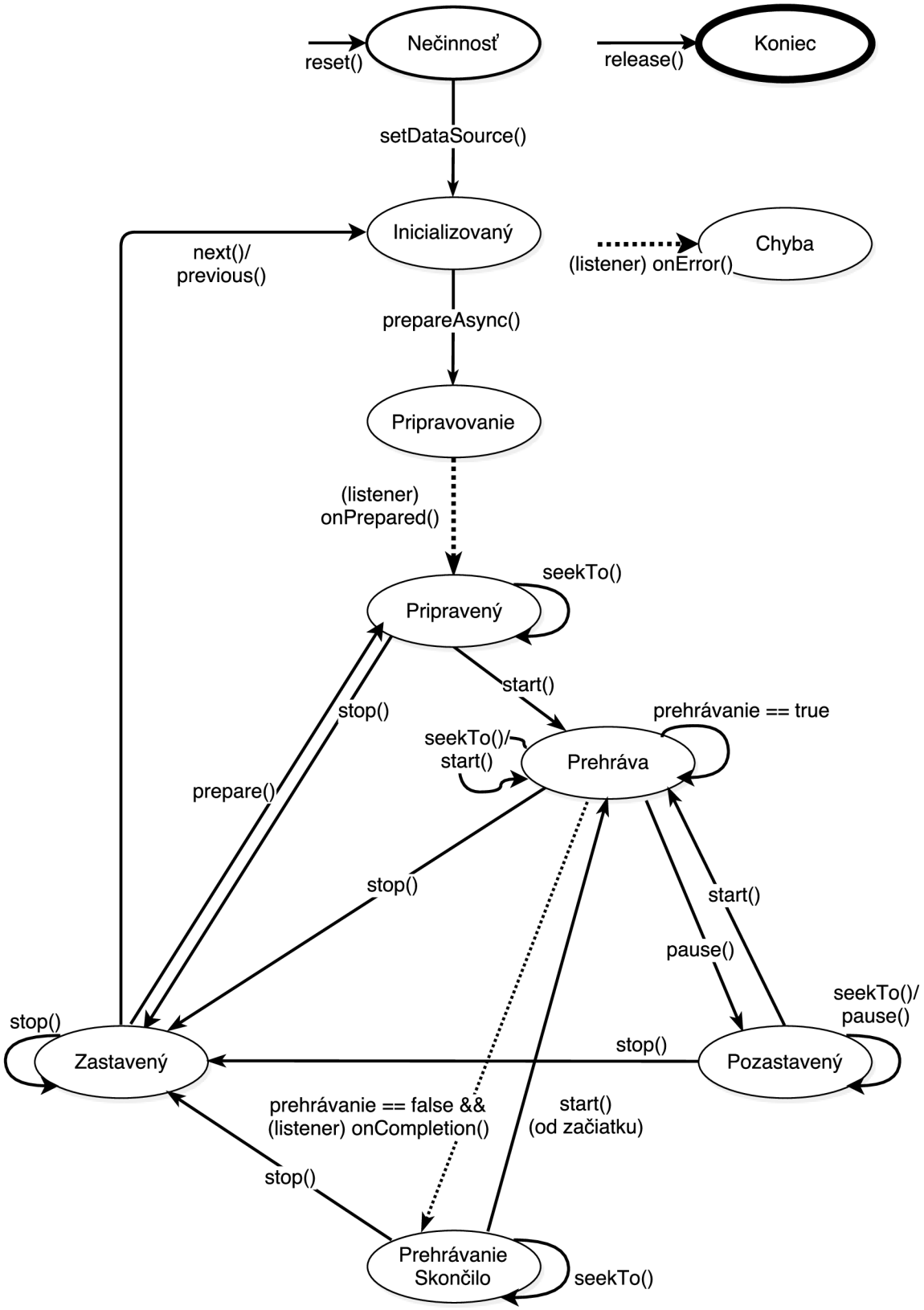
Obr. 5.4: Konceptuálny diagram implementácie prehrávania a streamovania audio súborov znázorňujúci závislosti medzi triedami.

Služba AudioPlayerService implementuje vzdialené rozhranie AIDL popísané v sekcií 4.2. Rozhranie definuje základné metódy pre nastavenie skladby, prepnutie skladby, spustenie, zastavenie, posúvanie sa v audio nahrávke, nastavenie a získanie aktuálne prehrávaného playlistu.

Ako mediálny prehrávač bol zvolený vstavaný systémový prehrávač **MediaPlayer**[7]. Tento prehrávač dokáže prehrávať a streamovať mediálne audio a video súbory do zariadenia. Neobsahuje však implementáciu životného cyklu prehrávania, a tak musel byť navrhnutý stavový automat, znázornený na obrázku 5.5.

Pri zmene stavov sa vykonáva viacero akcií, popísané budú len tie najdôležitejšie. Tabuľka činností, ktoré sa vykonávajú v určitých stavoch je znázornená v tabuľke 5.1. Z tabuľky je jasné, aké akcie má na starosti trieda MediaPlayer, ktorá je súčasťou služby AudioPlayerService. Čo je však potrebné kontrolovať, je stav tzv. **Audio Focusu**. Ten má na starosti to, aby sa zvuky z ostatných aplikácií medzi sebou neprekrývali. Ak aplikácia získa **AudioFocus** a súbor na prehrávanie je pripravený, prehrávanie sa začne. Ostatné aplikácie, žiadajúce o AudioFocus tento Focus nedostanú, až pokiaľ sa ho aktuálna aplikácia nevzdá.

[7]MediaPlayer - https://developer.android.com/reference/android/media/MediaPlayer.html



Obr. 5.5: Stavový automat znázorňujúci cyklus prehrávania audio súborov. Plné čiary znázorňujú synchrónné prechody medzi stavmi. Prerušované čiary znázorňujú asynchrónne prechody.

Ďalším opatrením, na ktoré je treba dávať pozor je tzv. stav **NOISY**. Tento stav sa mení napríklad pri vytiahnutí slúchadiel zo zariadenia počas prehrávania. V tom prípade je treba zabezpečiť, aby sa prehrávanie pozastavilo. Stav NOISY registruje aj prípad, kedy napríklad prichádza hovor alebo sa spustí alarm. V tom prípade je buď potrebné znížiť hlasitosť prehrávania alebo ho pozastaviť a po skončení alarmu prehrávanie znovu spustiť.

	Nečinnosť	**Prehráva**	**Pozastavený**	**Zastavený**	**Koniec**
MediaPlayer	initializácia	prepare ->play	pause	stop	uvolniť prostriedky
Audio Focus		request focus		abandon focus	
NOISY		register	unregister		
MediaSession	inicializácia obnova session nastavenie callbackov	setActive(true) nastavenie meta dát aktualizácia	aktualizácia	setActive(false)	uvolniť prostriedky uložiť session
Notifikácie		start FG	stopFG(false)	stopFG(true)	

Tabuľka 5.1: Operácie, ktoré sa vykonávajú pri zmene stavu prehrávania.

Pri zmene stavu sa prostredníctvom mediálnych notifikácií popísaných v nasledujúcej sekcií notifikuje uživateľ. S použitím mediálnych notifikácií má uživateľ možnosť priamo ovládať prehrávanie a prepínať si skladby alebo pozastavovať prehrávanie. Prehrávanie sa dá ovládať aj pomocou bluetooth zariadení a to správnou implementáciou **MediaSession**. MediaSession sa aktualizuje pri každej zmene stavu. Notifikácia je pri prehrávaní perzistentná. To znamená, že ju užívateľ nemôže odstrániť. Po pozastavení prehrávania sa táto perzistentnosť odstráni a uživateľ môže notifikáciu odstrániť. Pri pozastavení sa uloží aktuálne sedenie (session) do uživateľských nastavení.

5.5 Mediálne notifikácie

Mediálne notifikácie sú notifikácie, ktoré obsahujú informácie o skladbe a o danom podcaste spolu s obrázkom podcastu. Tieto notifikácie sa zobrazujú pri každej zmene stavu prehrávania alebo aj v prípade, že sa obnoví **session** (sedenie) po znovuspustení aplikácie.

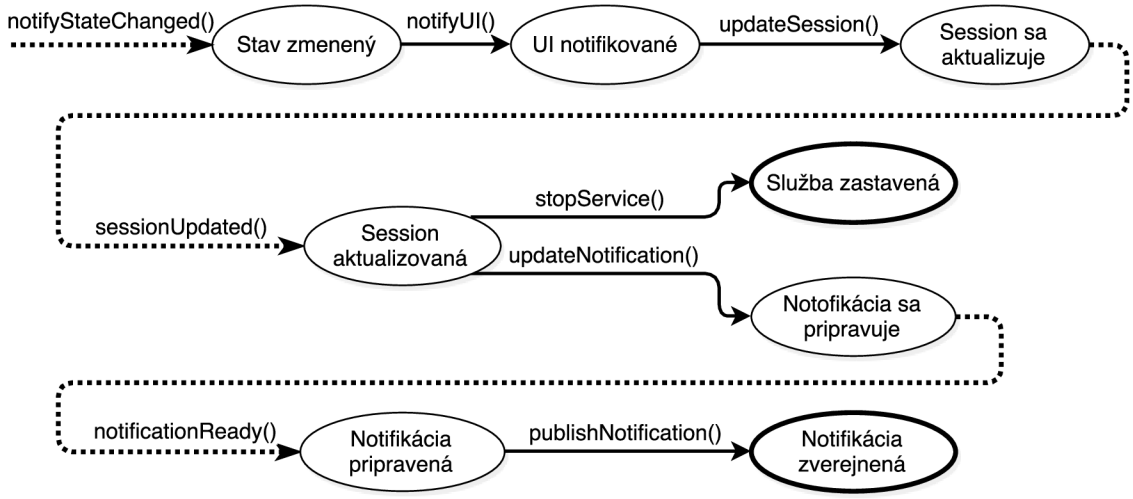
Po úspešnej zmene stavu prehrávania sa asynchrónne spustí udalosť, ktorej priebeh je znázornený na obrázku 5.6. Táto udalosť je implementovaná ako trieda zdedená z triedy **AsyncTask**[8]. AsyncTask je vhodná pre operácie, ktoré sa vykonávajú mimo hlavného **UI vlákna**[9] aplikácie a výsledok tejto operácie sa má zverejniť na UI vlákne.

V momente, kedy sa notifikuje zmena stavu, aktualizuje sa session. Pri aktualizácií sa z internetu alebo z **LRU cache**[10] vyberie a dekóduje bitmapa. Táto operácia je časovo náročná a mohla by spôsobiť výpadok snímkov alebo potencionálne zamrznutie aplikácie. Z tohto dôvodu je táto operácia znovu asynchrónna. V momente, kedy sa zmení stav príliš rýchlo, predchádzajúca operácia sa zruší a spustí sa nová. Ak sa mení stav na koniec prehrávania, v momente aktualizácie session sa ukončí služba na prehrávanie epizód. Po aktualizácií session sa vykoná posledná asynchrónna operácia prípravy notifikácie. Táto príprava taktiež zahŕňa dekódovanie bitmapy v inej veľkosti. Po zostavení sa notifikácia zverejní na UI vlákne.

[8] AsyncTask - http://developer.android.com/reference/android/os/AsyncTask.html

[9] UI vlákno - hlavné vlákno aplikácie, ktoré zabezpečuje zobrazenie uživateľského rozhrania. Na zobrazenie snímku má aplikácia 16ms. Po prekročení tejto hodnoty vplyvom veľkého množstva operácií môže uživateľ pocítiť mrznutie aplikácie - https://www.youtube.com/watch?v=CaMTIgxCSqU

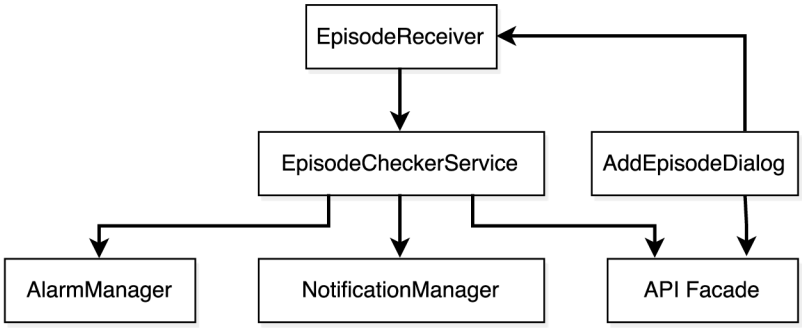
[10] Cachovanie obrázkov - http://developer.android.com/training/displaying-bitmaps/cache-bitmap.html



Obr. 5.6: Proces aktualizácie mediálnej notifikácie. Celé čiary znázorňujú synchrónne operácie a prerušované čiary operácie asynchrónne.

5.6 Pridávanie epizód

Z obrazovky zoznamu všetkých epizód alebo detailu podcastu je možnosť pridávania epizód z internetového zdroja. Tento zdroj musí byť podporovaný službou Audeliver. Ak sa pri zobrazení dialógu deteguje, že v systémovej schránke sa nachádza validný URL odkaz, tento odkaz bude automaticky vložený do vytvoreného dialógu **AddEpisodeDialog**. Taktiež je možnosť pridať epizódu priamo z aplikácií, akou je napríklad aplikácia YouTube. Diagram tried je názornený na obrázku 3.6b.



Obr. 5.7: Konceptuálny diagram implementácie pridávania epizód znázorňujúci závislosti medzi triedami.

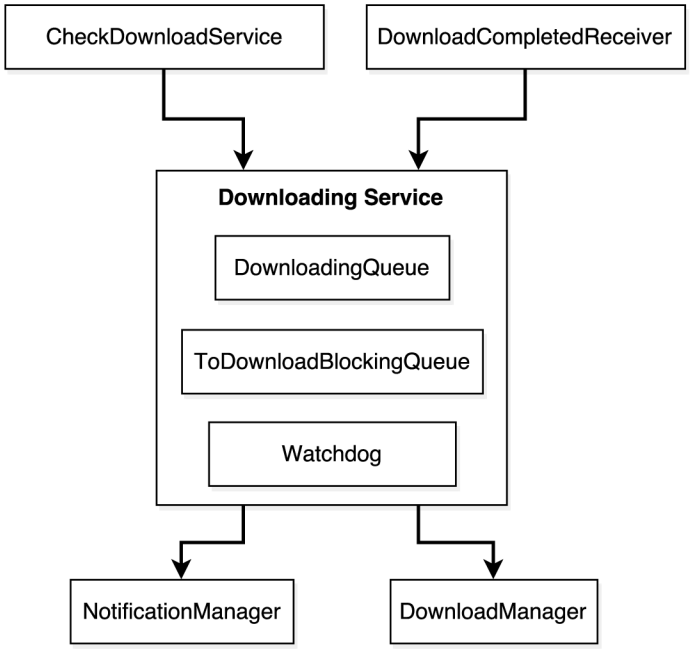
Po potvrdení dialógového okna sa prostredníctvom triedy **API Facade** odošle požiadavok na server o pridaní novej epizódy do vybraného podcastu. Ak príde odpoveď, v ktorej je uvedené, že nová epizóda sa spracováva, užívateľ je o tejto udalosti upozornený prostredníctvom **Toast**[11] správy. Následne sa nastaví prostredníctvom triedy **AlarmManager**[12] alarm, ktorý naplánuje opakované dotazovanie na server a vytvorí sa dočasná položka v zozname epizód.

[11] Toast - http://developer.android.com/guide/topics/ui/notifiers/toasts.html
[12] AlarmManager - https://developer.android.com/reference/android/app/AlarmManager.html

Po uplynutí nastavenej doby (5s) zachytí tento alarm trieda **EpisodeReceiver**. EpisodeReceiver spustí službu **EpisodeCheckerService**, ktorá sa informuje o stave spracovávanej epizódy. Pri stálom spracovávaní sa naplánuje ďalšia kontrola. Pri chybe alebo úspešnom spracovaní sa uživateľovi zobrazí notifikácia o úspešnom spracovaní a záznam sa pridá do internej databáze.

5.7 Sťahovanie

Ak uživateľ klikne na stiahnutie epizódy, spustí sa služba **DownloadService**. Konceptuálny diagram tried pre sťahovanie je znázornený na obrázku 5.8. Po spustení služby sa jej asynchrónne predá argument s identifikáciou epizódy, ktorú chce uživateľ stiahnuť. Tento identifikátor sa dostáva do fronty, ktorú predstavuje trieda **ToDownloadBlockingQueue**. Táto fronta má implementovaný výlučný prístup. Znamená to, že do fronty môže vkladať alebo z nej vyberať iba jedno vlákno. Výlučný prístup k tejto fronte je implementovaný z dôvodu, že pri asynchrónnom predaní argumentu s identifikátorom epizódy sa následne vyberá prvá položka z tohto zásobníku pre začatie sťahovania a mohlo by dôjsť k výberu už vybraného identifikátoru.



Obr. 5.8: Konceptuálny diagram implementácie sťahovania epizód znázorňujúci závislosti medzi triedami.

Po vložení identifikátoru do fronty na stiahnutie sa vyberie povolený počet identifikátorov, zistí sa URL adresa a spustí sa sťahovanie, ktoré má na starosti trieda **DownloadManager**[13]. Sťahované epizódy sa dostanú do fronty sťahovaných epizód **DownloadingQueue**.

Úspešné stiahnutie zachytí trieda **DownloadCompletedReceiver**. Následne sa zistí cesta k stiahnutému súboru a zapíše sa k sťahovanej epizóde. Identifikátor stiahnutej epizódy

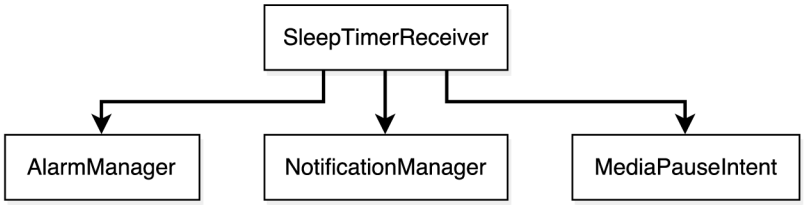
[13]DownloadManager - https://developer.android.com/reference/android/app/DownloadManager.html

sa vymaže z fronty DownloadingQueue a uživateľ bude notifikovaný o úspešnom dokončení sťahovania prostredníctvom triedy **NotificationManager**.

Trieda **Watchdog** periodicky kontroluje stav sťahovania a stav front. Ak Watchdog zistí, že sú obe fronty prázdne, služba na sťahovanie sa ukončí. Pri zlihaní služby alebo reštarte zariadenia sa po spustení aplikácie skontrolujú posledné sťahované epizódy. Túto funkcionalitu obsluhuje trieda **CheckDownloadService**.

5.8 Časovač

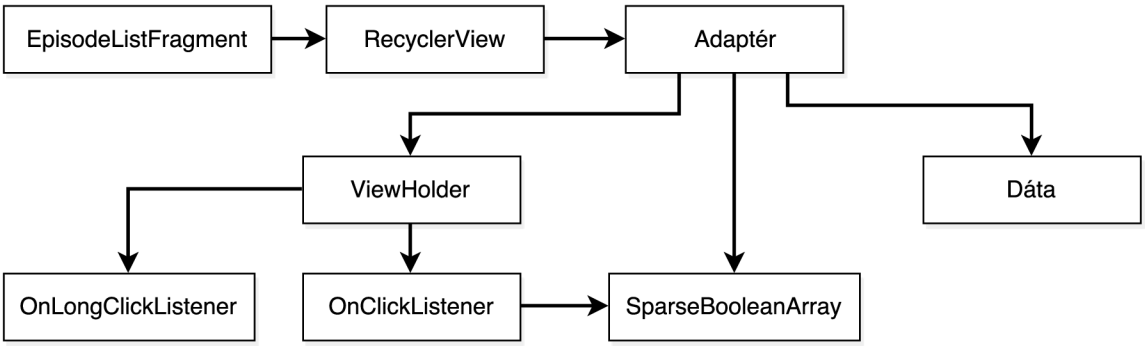
Časovačom je myslená komponenta, ktorá po uplynutí nastaveného času pozastaví prehrávanie. Po aktivácií časovača sa vytvorí notifikácia, oznamujúca zostávajúci počet minút do pozastavenia prehrávania. Následne sa použitím triedy **AlarmManager** naplánuje ďalšia aktualizácia notifikácie so zostávajúcim časom. Aktiváciu alarmu zachytí trieda **SleepTimerReceiver**, ktorá znovu naplánuje ďalší alarm a aktualizuje notifikáciu. Ak už uplynul nastavený čas, po **broadcaste** sa vyšle **MediaPauseIntent**, ktorý je podmnožinou triedy **PendingIntent**[14]. Ak beží služba na prehrávanie podcastov, príjmač na strane služby pozastaví prehrávanie. Diagram tried je znázornený na obrázku 5.9.



Obr. 5.9: Konceptuálny diagram implementácie časovača na pozastavenie prehrávania podcastov. Diagram znázorňuje závislosti medzi triedami.

5.9 Označovanie položiek

Framework Androidu neposkytuje jednoduchú možnosť ako označovať položky zoznamov a zároveň získať referencie na označené položky. Z tohto dôvodu musela byť navrhnutá vlastná implementácia tohto problému. Diagram tried a ich závyslostí sa nachádza na obrázku 5.10.



Obr. 5.10: Konceptuálny diagram implementácie označovania položiek v zoznamoch znázorňujúci závislosti medzi triedami.

[14]PendingIntent - http://developer.android.com/reference/android/app/PendingIntent.html

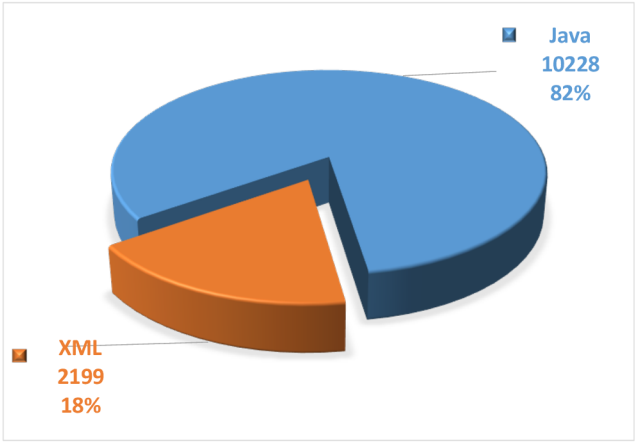
Princípy implementácie zobrazenia zoznamu boli popísané v sekcií s obrázkom 4.4. Avšak v tomto prípade je potrebné získať zoznam označených položiek z komponenty **EpisodeListFragment**. Samotné dáta, ktorých referencie chceme získať, sa nachádzajú až za triedou **Adaptér**. Na triedu **ViewHolder**, ktorá reprezentuje grafické znázornenie položky zoznamu, boli nastavené obslužné funkcie (**callbacky**), ktoré reagujú na klik alebo dlhý klik.

Po dlhom kliknutí sa aktivuje možnosť označovania a zároveň sa položka označí. Ostatné položky je možné následne označiť jedným kliknutím. Pri kliknutí na položku sa v riedkom poli, implementovanom v triede **SparseBooleanArray**, označí poriadie položky v zozname. Ak chceme získať identifikátory označených položiek, z triedy **Adaptér** získame zoznam označených pozícií (SparseBooleanArray). Tento zoznam už vieme namapovať na dáta a vytvoriť tak zoznam identifikátorov, ktorý sa zostaví na vyžiadanie a bude dostupný aj pre triedu **EpisodeListFragment**.

5.10 Metriky

V tejto kapitole boli popísané len najdôležitejšie časti riešenia implementácie. Veľké množstvo kódu nie je algoritmicky náročná, no architektonicky náročná, kedy je potrebné riešiť komunikáciu medzi triedami, ktoré sú prepojené veľkým počtom API vrstiev. Veľa zdanlivo jednoduchých problémov je potrebné riešiť komplexne a implementovať tak veľké množstvo kódu. Z dôvodu obmedzeného rozsahu nie je teda možné popísať detailnejšie časti implementácie. Ako názorná ukážka obsiahlosti projektu však môžu napovedať metriky kódu a to konkrétne počet riadkov kódu v jednotlivých programovacích jazykoch.

Na obrázku 5.11 je znázornené podielové zastúpenie dvoch hlavných jazykov. Ide konkrétne o značkovací jazyk **XML** využívaný pri návrhu uživateľského rozhrania. Hlavným implementačným jazykom je **Java**, ktorá tvorí 82% celého projektu. Na výpočet metrík bol použité rozšírenie **MetricsReloaded**[15] pre **Android Studio**, v ktorom bolo projekt implementovaný. Graf taktiež zobrazuje počet riadkov čistého kódu, bez zarátania externých a interných knižníc. Do výsledku nie sú zarátané ani komentáre a prázdne riadky.



Obr. 5.11: Graf znázorňujúci zastúpenie programovacích jazykov a počet riadkov kódu.

[15]MetricsReloaded - https://github.com/BasLeijdekkers/MetricsReloaded

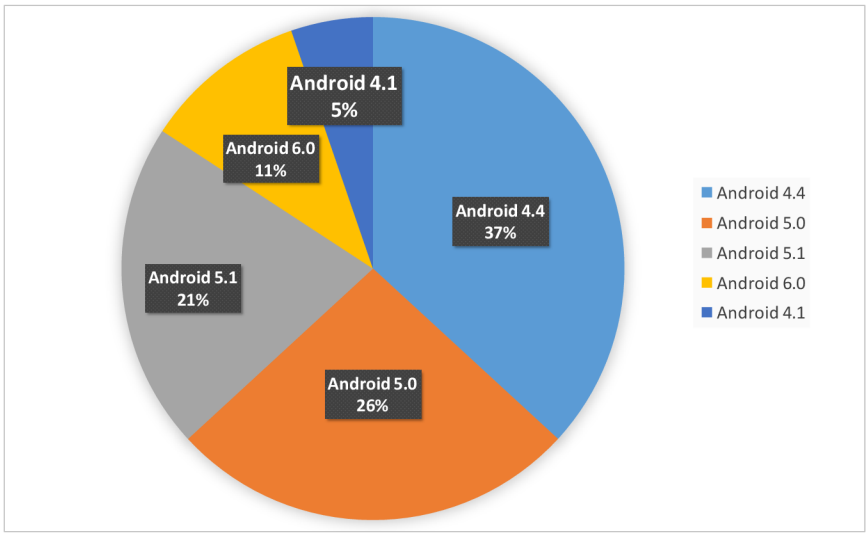
Kapitola 6

Testovanie

Súčasťou zadania je zverejnenie výslednej aplikácie. K zverejneniu bolo treba vytvoriť vývojársky účet a zaplatiť jednorázový členský poplatok. Pri zverejňovaní sa aplikácia digitálne podpisuje privátnym kľúčom. Aplikácia bola v neskorších fázach úspešne zverejnená v obchode **Google Play**, pod názvom **Audeliver**[1]. Pred verejnou publikáciou prebiehala fáza beta testovania. Beta testovanie bolo dostupné len pre prihlásených testerov. Táto kapitola rozoberá testovanie a postupnú iteráciu vývoja funkčnosti a vzhľadu aplikácie. Grafy štatistík sú z obdobia mesiacov apríl a máj 2016, kedy bola aplikácia zverejnená ako finálna verejná verzia. Tieto štatistiky pochádzajú zo služby Crashalitics, popísanej v sekcií 4.4.

6.1 Testovanie stability a funkčnosti

V beta fáze uživatelia testovali aplikáciu podľa prípadov užitia, popísaných na obrázku 3.1. Testovanie odhalilo množstvo chýb, ktoré boli spôsobené fragmentáciou systému. Problémy nastávali aj pri rovnakej verzií operačného systému. Je to z dôvodu, že mnohí výrobcovia k zariadeniam dodávajú upravené verzie systémov a modifikujú tak aj základné systémové komponenty. Fragmentácia uživateľskej základne je znáznornená na obrázku 6.2.

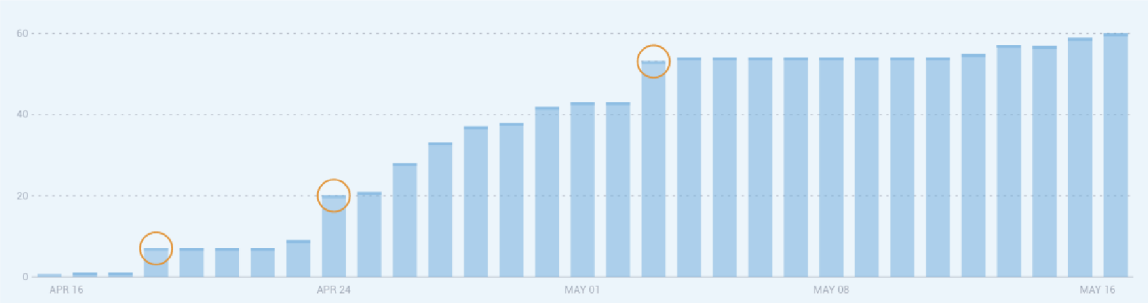


Obr. 6.1: Graf znázorňujúci fragmentáciu systému uživateľskej základne aplikácie Audeliver.

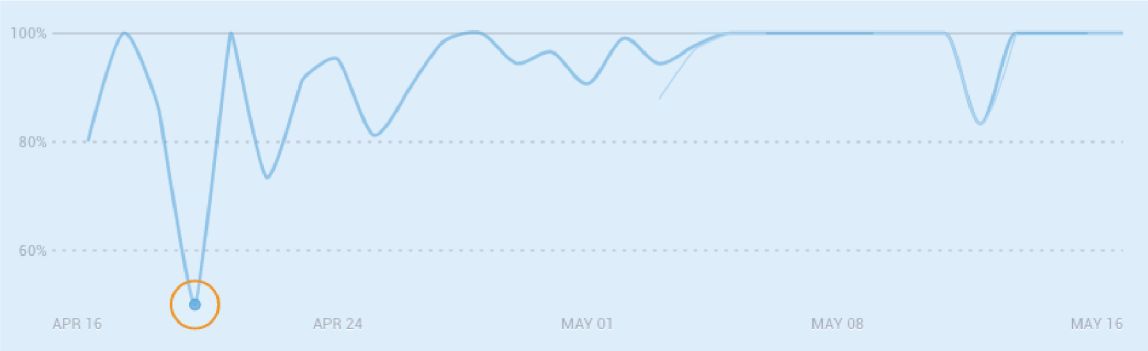
[1] Audeliver - https://play.google.com/store/apps/details?id=com.bajanik.podcasts

Chybovosť aplikácie

Celkový počet chýb, spôsobujúcich pád aplikácie v produkcií bol 141, avšak počet zasiahnutých užívateľov bolo iba 9 a to z celkového počtu 60. Chybami teda trpelo len pár zariadení. Ako vidieť na obrázku 6.2 a obrázku 6.3, najväčšia chybovosť bola zaznamenaná 16. apríla, kedy si aplikáciu nainštalovali prvý užívatelia. V nasledujúcich dňoch boli chyby vďaka reportom modulu Crashalitics odstránené a nahlásené chyby sa už neopakovali.



Obr. 6.2: Graf znázorňujúci počet aktívnych užívateľov v rozpätí dvoch mesiacov. Oranžové krúžky znázorňujú nezvyčajný nárast počtu uživateľov voči predchádzajúcim dňom.

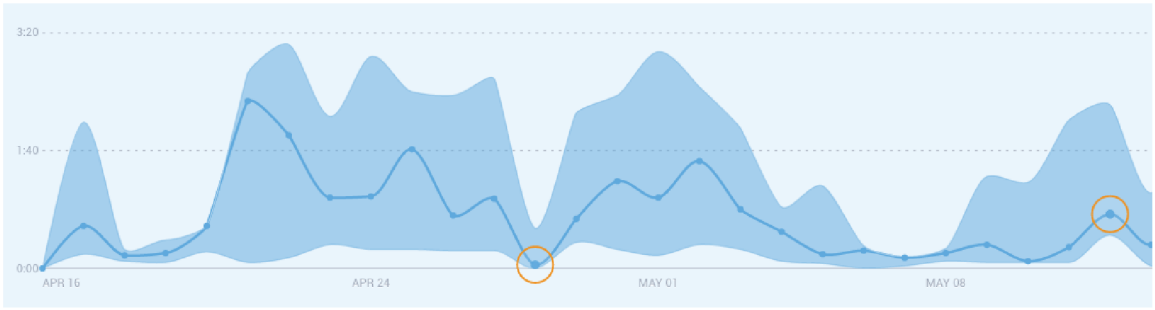


Obr. 6.3: Graf znázorňujúci percentuálny pomer bezpádových behov aplikácie v priebehu jej verejného zverejnenia. Oranžový krúžok znázorňuje stav, kedy bola chybovosť takmer 50%.

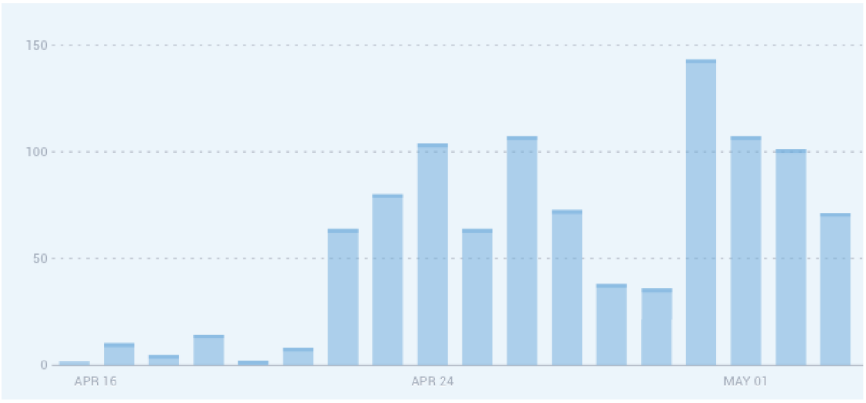
Používanie aplikácie

Graf na obrázku 6.4 znázorňuje čas, strávený uživateľmi v aplikácií počas jedného sedenia (session). Jedno sedenie predstavuje čas, kedy je aplikácia aktívna na obrazovke a užívateľ ju používa. Tento čas predstavuje medián zo všetkých jednotlivých sedení v danom dni. Do štatistík sa nezapočítava čas, ktorý je strávený prehrávaním audio nahrávok alebo ovládaním aplikácie prostredníctvom iného bluetooth zariadenia alebo z mediálnych notifikácií.

Ako je možné vidieť na grafe 6.5, uživatelia využívajú aplikáciu aktívne. Maximálny počet sedení v jednom dni dosiahol počtu 143. Za relatívne krátku dobu si tak aplikácia získala svojich uživateľov, ktorým budem aj nadalej poskytovať podporu a aplikáciu podľa ďalších odoziev vylepšovať.



Obr. 6.4: Graf znázorňujúci medián stráveného času v aplikácií, počas jedného sedenia.



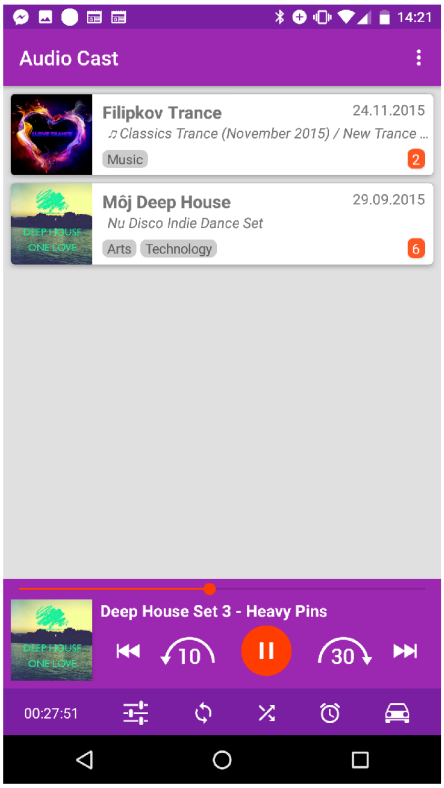
Obr. 6.5: Graf znázorňujúci celkový počet sedení v danom dni.

6.2 Testovanie uživateľského rozhrania

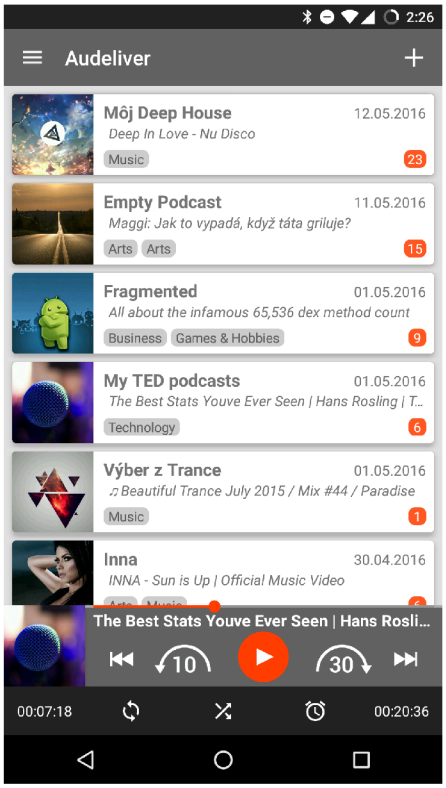
Užívateľské rozhranie prešlo počas fáze beta testovania rôznymi zmenami. Niektoré prvky grafického rozhrania neboli na menších displejoch viditeľné alebo naopak, zaberali príliš veľkú časť obrazovky. Príkladom je obrázok 6.6a, kde prehrávač zaberal značnú časť obrazovky. Vhodnejšie využitie voľného miesta redizajnom prehrávača, je znázornené na obrázku 6.6d. Prehrávač je možné gestom zmenšiť a uvolniť tak ďalšie miesto na obrazovke. Atribút dĺžky epizód sa premiestnila do pravej časti položky zoznamu a nadpis sa tak mohol roztiahnuť na dva riadky. Pri menších uhlopriečkach bol výsledok značný.

Po spracovaní uživateľských odoziev sa v aplikácií zmenila aj pôvodná farebná schéma aplikácie a to na farby viac neutrálne. Výsledok je vidieť medzi obrázkami 6.6a a 6.6b. Čo sa týka farieb, zmenami prešiel aj detail podcastu. Rozdiel je znázornený na obrázkoch 6.6c a 6.6d. Z obrázku podcastu (artworku) sa extrahovali dve dominantné farby, ktoré sú najvýraznejšie a aplikovali sa na farebnú schému otvoreného detailu podcastu. Ak sa na obrázku nenájde vhodná farba, použije sa pôvodná farebná schéma aplikácie.

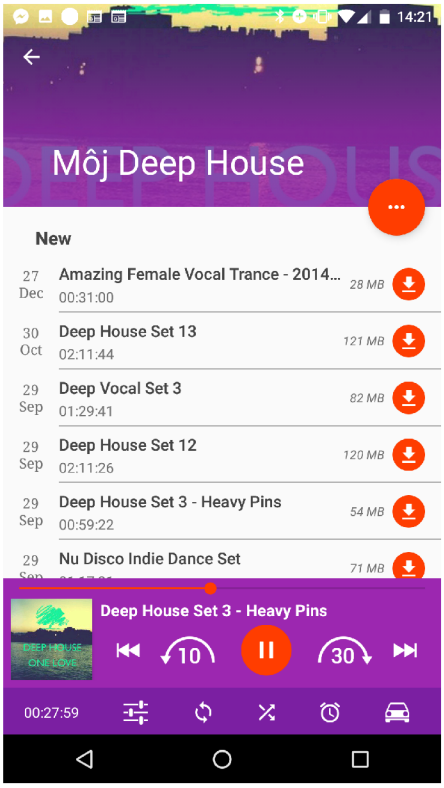
Veľký obrázok podcastu v hornej časti obrazovky umožňuje pri väčších uhlopriečkach ovládanie jednou rukou. Jednou rukou je teda možné bez prehmatu vybrať najvyššiu položku zoznamu. Pri vertikálnom posúvaní zoznamu smerom nadol sa obrázok skryje.



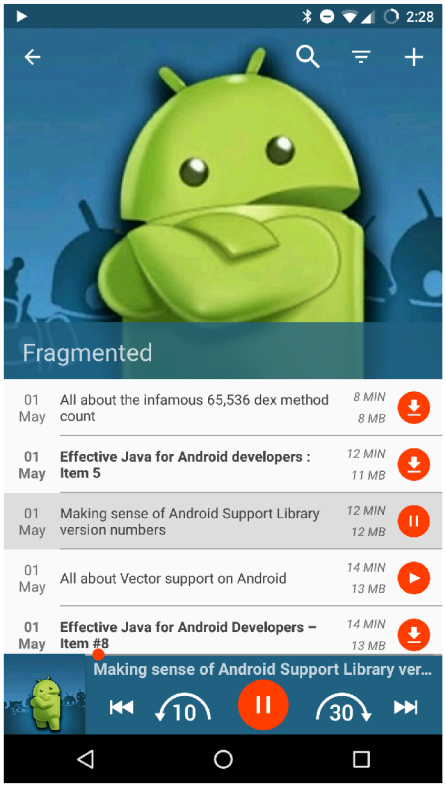
(a) Zoznam podcastov pred úpravou



(b) Zoznam podcastov po úprave



(c) Detailu podcastu pred úpravou



(d) Detail podcastu po úprave

Obr. 6.6: Obrázky znázorňujú časť zmien, ktoré boli vykonané počas testovania.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať Android aplikáciu pre sťahovanie a posluch audio podcastov. V druhej kapitole boli analyzované podcasty, ich štruktúra a konkurenčné riešenia. Z nich bol následne vyvodený diagram prípadu užitia. Tento diagram slúžil na vytvorenie špecifikácií požiadavkov, ktoré boli ďalej detailne rozpísané. Ďalšia kapitola sa zaoberala návrhom grafického uživateľského rozhrania, spolu s návrhom databázy.

Grafické rozhranie, ale aj funkčné jadro aplikácie bolo úspešne implementované a otestované. Výsledkom implementácie je aplikácia, ktorá dokáže synchronizovať audio podcasty spolu s obrázkami podcastov a má jednoduché a účelné uživateľské rozhranie. Aplikácia dokáže streamovať a sťahovať ľubovolnú epizódu zo zvoleného podcastu, posúvať sa v čase prehrávania na ľubovolnú pozíciu, zastavovať prehrávanie a prehrávať audio súbory na pozadí.

Ďalšími funkciami prehrávača je možnosť rýchlej tvorby playlistov, označením viacerých položiek alebo možnosť náhodného prehrávania. Veľkou výhodou aplikácie je aj integrácia služby Audeliver, kedy je možné z aplikácií akou je napríklad YouTube, importovať audio záznam z videa, priamo do vlastného podcastu. Podcasty je možné aj vytvárať. Jednou z implementovaných častí bola komunikácia s bluetooth zariadeniami, umožňujúca vzdialené ovládanie.

V budúcnosti by som rád pokračoval vo vývoji a v rozširovaní funkčnosti aplikácie. Jednou z možností by mohla byť integrácia generických podcastov a začlenenie vyhľadávania podcastov z rôznych verejných databáz.

Aplikácia bola spočiatku uverejnená v beta programe Google Play a jej dostupnosť bola na pozvánku. Po prechode z fáze beta testovania bola aplikácia zverejnená ako finálna verejná verzia. Aplikáciu je možné nájsť v obchode Google Play pod názvom **Audeliver**.

Literatúra

[1] Android Developers: Develop Apps. [Online; navštívené 15.2.2016].
URL https://developer.android.com/

[2] RSS tags for Podcasts Connect. [Online; navštívené 23.4.2016].
URL https://help.apple.com/itc/podcasts_connect/#/itcb54353390

[3] Berners-Lee, T.; Fielding, R. T.; Nielsen, H. F.: Hypertext Transfer Protocol – HTTP/1.0. RFC 1945, RFC Editor, May 1996, section 11.
URL http://www.rfc-editor.org/rfc/rfc1945.txt

[4] Chen, P. P.-S.: The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, ročník 1, 1976: s. 9–36.

[5] Freeman, E.; Bates, B.; Sierra, K.; aj.: *Head First Design Patterns*. OReilly Media, 2004, ISBN ISBN 978-0-596-00712-6.

[6] Ergonomics of human-system interaction. 2010-03-15.
URL http://www.iso.org/iso/catalogue_detail.htm?csnumber=52075

[7] Ľuboslav Lacko: *Vývoj aplikací pro Android*. Brno: Computer Press, první vydání, 2015, ISBN 9788025143476.

[8] Rodriguez, A.: RESTful Web services: The basics. 2016-4-19, [Online; navštívené 22.4.2016].
URL https://www.ibm.com/developerworks/webservices/library/ws-restful/

Prílohy

Príloha A

Obsah CD

- Prezentačný plagát (**poster.pdf**)
- Prezentačné video (**video.mp4**)
- Popis k videu (**video_description.xml**)
- Zdrojové súbory aplikácie (**application.zip**)
- Spustitelná verzia aplikácie (**app-release.apk**)
- Technická správa (**documentation.pdf**)