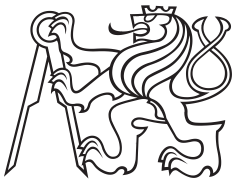
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Integrace audio knihovny

Petr Krýže

Vedoucí: Ing. Daniel Novák, Ph.D.
Studijní program: Kybernetika a robotika
Obor: Robotika
Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Petr K r ý ž e

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Integrace audio knihovny

Pokyny pro vypracování:

1. Seznamte se s problematikou rozhraní audio knihoven jako např. LibriVox.
2. Navrhněte řešení přístupu k vybrané audio knihovně.
3. Řešení naimplementujte v programovacím jazyce Java.

Seznam odborné literatury:

[1] Suzor, Nicolas P., Harpur, Paul D., & Thampapillai, Dilan - Digital copyright and disability discrimination: From braille books to bookshare. (2008)
[2] Vincent Gaudissart, Silvio Ferreira, Celine Thillou, Bernard Gosselin, SYPOLE - Mobile Reading Assistant for Blind People, SPECOM- 9th Conference Speech and Computer. (2004)

Vedoucí bakalářské práce: Ing. Daniel Novák, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 15. 12. 2015

Poděkování

Rád bych poděkoval rodičům za jejich nehynoucí podporu skrze celé mé studium a své přítelkyni za to, že je mou oporou při cestě vesmírem. Velké poděkování patří také všem z vývojového týmu, pracovníkům centra SONS, participantům uživatelského testování a v neposlední řadě vedoucímu práce Danielu Novákovi Ph.D. za optimismus a podporu v průběhu celé práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....
Podpis autora práce

Abstrakt

Náplní této práce je integrace audioknihovny LibriVox do mobilního prostředí pro nevidomé a zrakově postižené uživatele na platformě Android. Aplikace umožňuje uživatelům vyhledávat, stahovat a přehrávat audioknihy zdarma a bez nutnosti registrace. Implementaci aplikace předcházela analýza řešení a konzultace návrhu s nevidomými. Při vývoji se podařilo vyřešit množství problémů s aplikačními rozhraními používaných služeb. Po dokončení práce proběhlo testování aplikace v laboratoři pro uživatelský výzkum.

Klíčová slova: audiokniha, Librivox, knihovna, nevidomí, Android

Vedoucí: Ing. Daniel Novák, Ph.D.

Abstract

This thesis aims to create an application that integrates services of the online audiobook library LibriVox into Android mobile environment for blind and visually impaired users. The application enables users to search, download and play audiobooks for free and without the necessity of registration. Implementation was preceded by an analysis of the solution and consultation of the proposed design with visually handicapped users. During the development, usage of application interfaces created a number of problems that were successfully solved. The work was concluded by final testing of the application in the usability lab.

Keywords: audiobook, Librivox, library, blind, Android

Title translation: Integration of an audio library

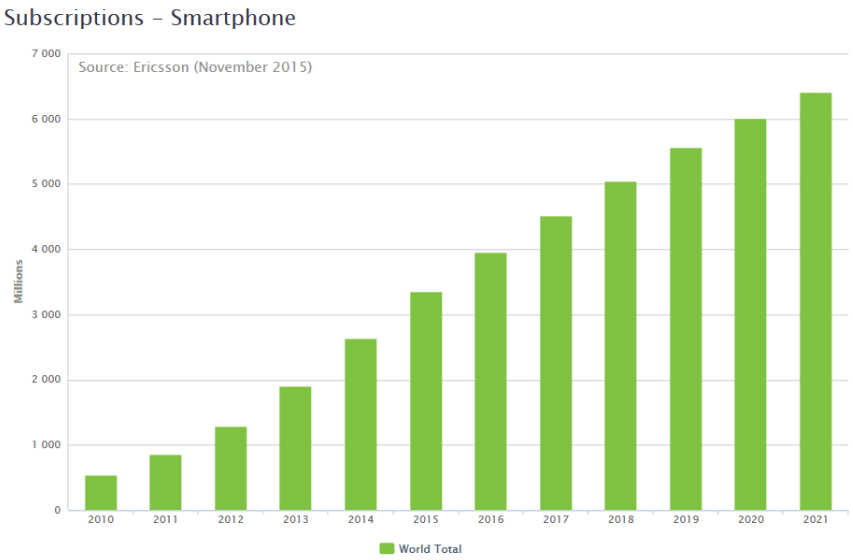
Obsah

Obrázky

Kapitola 1

Úvod

Svět mobilních zařízení v čele s chytrými telefony zažívá v současné době velký rozmach. Podle analýzy *Ericsson Mobility Report* [1] společnosti Ericsson z listopadu roku 2015 se očekává, že v roce 2021 bude celosvětově mezi uživateli více než 6 miliard chytrých telefonů, což je téměř dvojnásobný nárůst za pouhých šest let (viz graf 1.1).



Obrázek 1.1: *Ericsson Mobility Report* - počet přihlášených chytrých telefonů celosvětově mezi roky 2010 až 2015, s výhledem do roku 2021
Zdroj: Vytvořeno pomocí nástroje *Traffic Exploration* - http://www.ericsson.com/TET/trafficView/loadBasicEditor.ericsson (12.5.2016)

Díky neustále se snižujícím výrobním nákladům se chytré telefony dostávají nejen do rukou obyvatel ekonomicky prosperujících zemí, ale i mezi široké řady obyvatelstva, pro které by dříve vlastnictví chytrého telefonu nebylo dostupné. K roku 2011 byly v méně rozvinutých zemích více než tři čtvrtiny světových mobilních zařízení a přes 879 milionů z nich se nacházelo jen v samotné Číně. [2, str. 89]

Chytrý telefon (nebo-li *smartphone*) se pro mnohé stal potřebnou součástí

každodenního života. Možnost mít u sebe neustále k dispozici přístup k internetu, hudebnímu přehrávači, kalendáři, fotoaparátu a dalším, již standardním součástem chytrého telefonu, je nejen velmi užitečná, ale v současném dynamickém světě i brzy nezbytná. Očekává se, že přebráním jejich funkcionality chytré telefony postupem času způsobí vymizení celých odvětví uživatelských technologií. [2, str. 92] Proto by ruku v ruce s průnikem chytrých telefonů mezi široké vrstvy obyvatel mělo přijít i řešení, které umožní využívat komfortu těchto zařízení lidem, kteří jsou zrakově znevýhodněni. Těm často chybí přístup k informacím, jejichž většina je tvořena textovými nebo obrazovými záznamy, což se projevuje i omezením možností použití nových technologií. [3]

Takové řešení nabízí systém vytvořený v roce 2013 jako školní projekt Petra Svobodníka v rámci jeho diplomové práce. [4] V mojí práci se na něj budu odkazovat jako na "stávající"nebo "současný"systém. Ten v mobilním telefonu zcela nahrazuje uživatelské rozhraní systému Android svým vlastním již od chvíle, kdy se telefon zapne. Rozhraní jeho uživatelského prostředí se sestává z jednotlivých obrazovek na kterých je pouze text - komunikace s uživatelem probíhá pomocí hlasové syntézy, (převodu textu na řeč) a vibrační odezvy systému. Mezi jednotlivými obrazovkami (položkami v nabídkách) se pak uživatel pohybuje pomocí jednoduchých gest, přímo za použití dotykové obrazovky telefonu. Hlavní součástí systému jsou pak základní funkce, které smartphone nabízí ve verzích přístupných nevidomým uživatelům - například volání, posílání SMS, budík, poznámky a podobně. Dalšími součástmi jsou nástroje vytvořené speciálně pro potřeby nevidomých (lupa, rozpoznávání bankovek nebo indikátor barev) a aplikace umožňující přístup ke službám, ze kterých mohou mít nevidomí užitek - například internetová rádia nebo databáze audioknih. Projekt je v současné době (2016) stále v aktivním vývoji.

V rámci své práce rozšířím možnosti stávajícího systému o aplikaci LibriVox, která umožní vyhledávání, stahování a přehrávání knih ze stejnojmenné audioknihovny. Nevidomým uživatelům zpřístupním poslech lidmi namluvené literatury, alternativně ke knihám předčítaným hlasovými syntezátory. Strukturu aplikace budu navrhovat s důrazem na jednoduchost, její implementaci pak podřídím rychlosti a robustnosti. Na závěr provedu testování aplikace na nevidomých uživatelích.

Kapitola 2

Příprava řešení

V rámci mojí bakalářské práce jsem se zaměřil na rozšíření dostupných možností poslechu knih ve stávajícím systému. V době vytváření práce byla k dispozici v systému aplikace *Bookshare*, umožňující přístup ke stejnojmenné digitální knihovně[1] zaměřené na zrakově postižené uživatele. Tato aplikace, kterou vytvořil v rámci své bakalářské práce Lukáš Rubeš, umožňuje uživatelům (za roční poplatek pro přístup do své databáze) stahovat knihy v textovém formátu a následně je přehrávat pomocí hlasového syntezátoru.[5]

Mým cílem bylo najít službu, která by uživatelům stávajícího systému umožnila poslech audioknih, tedy knih namluvených lidmi (bez použití hlasové syntézy). Zároveň by tato služba měla být snadno dostupná pro nevidomé a měla by poskytovat možnosti pro vývojáře, tedy použitelné rozhraní API[2], aby bylo možné službu zpřístupnit na mobilním zařízení.

V době vytváření této práce byly hlavními zvažovanými službami následující (ve stručném přehledu):

- **Audible**[3] - ve vlastnictví společnosti *Amazon*. Placené členství (nutnost registrace, základní cena 14.95$ za měsíc), v základním členství 1 kniha na měsíc. K dispozici více než 150 tisíc titulů, vlastní mobilní aplikace.

- **Scribd**[4] - k dispozici textové knihy, audioknihy, komiksy a další díla, nejen od známých autorů. Placené (nutnost registrace, 8.99$ za měsíc) pro členy 1 audiokniha a 3 klasické knihy na měsíc (navíc neomezený přístup k notovým zápisům, dokumentům a dalším vybraným titulům). K dispozici přibližně 150 tisíc titulů (včetně klasických knih, uživatelských děl apod.), API existující, ale nedostupné (ukončili vydávání klíčů pro jeho použití), vlastní mobilní aplikace.

- **Librivox**[5] - knihy namluvené dobrovolníky (jedna kniha namluvena jedním či více lidmi). Zcela zdarma bez nutnosti registrace, k dispozici více než 20 tisíc namluvených knih (většina v angličtině). Nabízí ale

[1] *BookShare* - `https://www.bookshare.org/cms`

[2] "*Application Programming Interface*"- rozhraní tvořené funkcemi a nástroji programu/knihovny pro externí využití. [6]

[3] *Audible* - `http://www.audible.com/`

[4] *Scribd* - `https://www.scribd.com/audiobooks`

[5] *LibriVox* - `https://librivox.org/`

pouze knihy které jsou (v USA) volnými díly (díla, která jsou v "public domain"- viz sekce 2.1.1). Dostupné API s ukončeným vývojem.

- **Audiotéka**[6] - český prodejce audioknih. Nabízí jednotlivé knihy (v cenových relacích klasických knih), je tedy nutná registrace. V nabídce téměř 5 tisíc titulů v českém jazyce. Mají vlastní mobilní aplikaci, ale veřejné API chybí.

Na internetu je zároveň k dispozici mnoho "neoficiální"webů, často spravovaných nadšenci, které dávají na svých stránkách k dispozici namluvená díla (volně) ke stažení, často však bez uvedeného původu a autora nebo bez jakéhokoliv dostupného API k použití pro vývojáře.

Po konzultaci s vedoucím práce byla po předložení nalezených variant vybrána audio knihovna LibriVox, ze dvou důvodů - její otevřenost (použití nevyžaduje registraci a je zdarma) velmi usnadní nevidomým uživatelům ovládání aplikace a přístup ke knihám, a dostupnost veřejného API umožní snadnou integraci do mobilního zařízení.

2.1 LibriVox

LibriVox je nezisková platforma, kterou založil v USA roku 2005 spisovatel Hugh McGuire. Jejím cílem je zpřístupnění volně dostupných děl v podobě audioknih, a to bez reklam a jen za pomoci dobrovolníků. Celý projekt je financovaný pomocí dobrovolných příspěvků, přičemž v letech 2010 a 2013 LibriVox uspořádal a úspěšně dokončil dvě *crowdfundingové*[7] kampaně za účelem zajištění financování nákladů spojených s provozem projektu.

Princip fungování LibriVoxu je jednoduchý: pro uživatele je k dispozici na webových stránkách vyhledávací pole, kde mohou procházet celý katalog audio knihovny, hledat v něm podle jmen autorů, názvů knížek, žánrů, nebo jazyků. V současné době je velká většina nahraných knih v anglickém jazyce (přes 20 tisíc knih), dále jsou zastoupeny knihy v německém (přes 1800 knih), francouzském (přibližně 500 knih) a čínském jazyce (okolo 400 knih). V ostatních jazycích je pak namluvených knih již méně.

Druhou stranou LibriVoxu jsou přispěvatelé - každý může namluvit knihu nebo kapitolu a přispět tak do virtuální knihovny. Koordinace nahrávání nových děl probíhá na fóru Librivox[8]. Kvůli relativní volnosti, kterou uživatelé pro vytváření nahrávek mají, se jejich kvalita může různit.

2.1.1 Volná díla

Základním kamenem, který umožňuje fungování LibriVoxu na otevřené bázi, je skutečnost, že veškeré namluvené knihy jsou v USA tzv. volnými díly[9].

[6] *Audiotéka* - `http://audioteka.com/cz/`

[7] "*Crowdfunding*"- způsob financování projektů pomocí malých příspěvků od velkého množství lidí (typicky za použití internetu a odměn za příspěvky). [7]

[8] *Librivox Forums* - https://forum.librivox.org/

[9] *Public domain* - `https://en.wikipedia.org/wiki/Public_domain`

Díky tomu, že LibriVox funguje pod zákony Spojených států, musí všechny knihy v jeho katalogu splňovat podmínky volného díla. Prakticky to znamená, že namluvena mohou být pouze díla publikovaná (vydaná) před rokem 1923. Platí přitom, že překlad díla je považován jako dílo nové, musí se tedy brát v úvahu rok publikování překladu místo roku publikace původního díla. Dále platí, že dílo, které je volným dílem v USA nemusí být volným dílem v jiných zemích a naopak. Díla publikovaná před rokem 1923 jsou v USA zákonitě volnými díly, díla publikovaná po zmíněném roce pak obvykle volnými díly nejsou, nicméně mohou existovat výjimky. V USA byl rok, po kterém jsou publikovaná díla považována za volná, několikrát zákonně upravován s cílem prodloužení ochrany duševního vlastnictví. [8] V současnosti se pak duševní vlastnictví děl v USA s ohledem na status volného díla řídí několika zákony a ustanoveními [9], nicméně stále platí, že veškerá díla publikovaná před rokem 1923 jsou bez výjimky považovaná za díla volná.

Veškeré namluvené knihy nebo samostatné kapitoly se po publikování v katalogu LibriVox stávají také volnými díly, a kdokoliv s nimi může téměř libovolně nakládat (stahovat je, upravovat je, prodávat je apod.).

To, že se LibriVox řídí zákony Spojených států je dáno zejména tím, že název domény "*LibriVox.org*"je zaregistrován právě v USA a webové stránky společně se všemi audio soubory jsou hostovány tamtéž. Zároveň také spolupracuje s webovými stránkami projektu Gutenberg[10], který se zaměřuje na digitalizaci knižních děl (která jsou právě volnými díly v USA), jejich kontrolu (tedy autorských práv) a volnou distribuci pro uživatele (více než 50 tisíc titulů). Mnoho z knih, které jsou vybrány pro namluvení dobrovolníky z LibriVoxu, pochází právě z katalogu dostupných knih projektu Gutenberg.

2.1.2 LibriVox API

Základním předpokladem pro výběr LibriVoxu byla existence veřejného API (tedy souboru metod, příkazů nebo funkcí volně dostupných pro vývojáře k jednoduchému využití poskytované služby). API LibriVoxu bylo v době vypracovávání práce ve stavu "*released*"[10] (česky vydáno), což prakticky znamená (jak jsem zjistil z příspěvků uživatelů a administrátorů na fóru LibriVox) že vývoj API byl uzavřen a neplánuje se jakékoliv další rozšíření jeho funkcionality. Později se ukázalo, že je to pro vývoj mojí aplikace velká překážka, viz sekce 2.1.2.

API LibriVoxu funguje na principu jednoduchých HTTP[11] dotazů na základní adrese:

```
https://librivox.org/api/feed/audiobooks
```

Tato adresa funguje jako kořen všech dotazů v podobě URI[12] sestávajícího se ze základní adresy a parametrů hledání. To pak probíhá v serverové části

[10] *Project Gutenberg* - `http://www.gutenberg.org/`

[11] *Hypertext Transfer Protocol* - aplikační protokol postavený na systému *dotaz-odpověď*. Používá se zejména pro přenos hypertextových souborů ve formátu HTML. [11]

[12] *Uniform Resource Identifier* - textový řetězec sloužící k identifikaci zdroje, například na internetu v podobě webové adresy. [12]

API a po zpracování dotazu se vrátí zpět odpověď v podobě souboru XML[13] , který obsahuje knihy nalezené na základě odeslaného dotazu.

Povolené parametry hledání jsou zejména:

- `id` - identifikační číslo knihy, odpovědí je jen jediná kniha.
- `author` - odpovědí jsou knihy jejichž autorovo příjmení odpovídá dotazu.
- `title` - navrátí knihy jejichž název odpovídá dotazovanému.

Možné parametry (typy dotazů) jsou ještě `since` (knihy přidané do databáze od zadaného data), a `genre` (knihy s příslušným žánrem). Zároveň je možné specifikovat následující možnosti:

- `extended = 1` - v XML odpovědi jsou u jednotlivých knih doplněny rozšířené informace (např. o překladatelích, kapitolách a žánrech).
- `limit` - určuje maximální počet knih odeslaných v odpovědi. Standardně nastaven na hodnotu 50.
- `offset` - odsadí seznam nalezených knih o specifikovanou hodnotu.

Nakonec je ještě možné specifikovat formát odpovědi. Standardně (bez specifikace) je nastaven formát XML, je ale možné pomocí identifikátoru *format*= požádat například o odpověď ve formátu JSON, JSONP a dalších.

Jako poslední možnost při formování dotazu je specifikace návratových polí s informacemi pro jednotlivé knihy odpovědi. Jinými slovy, pomocí identifikátoru `fields= { field1, field2, ...}` lze určit jaké informace chceme u jednotlivých knih v odpovědi mít.

Dotazy musí mít pak syntaxi:

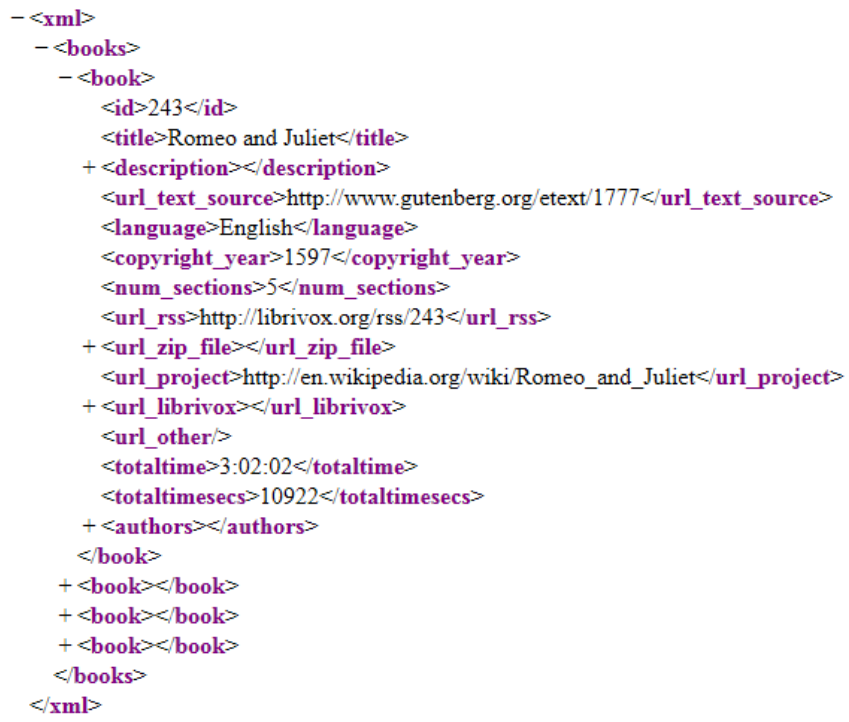
kořenová_URL/?*typ_dotazu*=**dotaz**&*parametr1*&*parametr2*...

Například:

```
https://librivox.org/api/feed/audiobooks/?title=odyssey&
limit=1&extended=1
```

Poslední možností, která pro vyhledávání existuje, je vložení znaku stříšky (^) (při hledání skrze název díla nebo žánru či jméno autora) před zadaný dotaz. To způsobí, že API vrátí v odpovědi i knihy, u nichž například jméno knihy zadaným název pouze začíná. Jinak API vrací pouze knihy se kterými má ve zvoleném poli naprostou shodu. Například tedy dotaz `?title=romeo` hledanou knihu od Williama Shakespeara nenajde, ale `?title=^ romeo` již ano.

[13] *Extensible Markup Language* - (značkovací) jazyk se standardizovanou strukturou pro reprezentaci dokumentů a dat v lidmi a stroji čitelné podobě. [13]



Obrázek 2.1: Typická odpověď z API LibriVox pro dotaz:
`https://librivox.org/api/feed/audiobooks/?title=^romeo`

Problémy

Při implementaci vyhledávání, která je popsaná v dalších částech, jsem narazil na několik problémů s použitím API LibriVox. Byly to zejména:

- Neexistence znaku, který by umožňoval vyhledávání knih, jejichž název (nebo jméno autora) končí zadaným dotazem (opak dostupného znaku stříšky).
- Vyplývající celková kvalita vyhledávání.
- Často chybějící informace v polích XML odpovědi, malá konzistence.

Nejvíce palčivým problémem byla právě kvalita vyhledávání, která způsobovala, že nebylo možné získávat relevantní odpovědi na zadané dotazy. Kupříkladu při hledání knihy *The Canterville Ghost* (*Strašidlo cantervillské*) od Oscara Wildea by mohl uživatel zadat do vyhledávacího pole slovo "*canterville*", a při správné implementaci (za použití znaku stříšky při tvorbě dotazu) by API skutečně tuto knihu v odpovědi vrátila. Pokud by si ale uživatel například nemohl vzpomenout na původ strašidla ve zmíněné knize, a zkoušel by hledat jen podle klíčového slova "*ghost*", správné knihy by se nedočkal - neexistuje totiž (jednoduchý a rychlý) způsob, jak dotaz zformovat programově tak, aby vyhledávací systém API LibriVox našel knihu, jejíž název končí zadaným výrazem. Analogický problém nastává při hledání pomocí

slova, které název dané knihy pouze jen obsahuje (není prvním ani posledním slovem v názvu knihy) - ani zde není API LibriVoxu ku žádné pomoci. Stejná situace je i se jmény autorů.

Pokusil jsem se zjistit, zda není k dispozici nějaká (neoficiální) cesta, například způsob formování dotazů, pomocí kterého by šly tyto komplikace obejít. Podal jsem tedy dotaz do příslušného diskuzního vlákna (založeného pro dotazy ohledně oficiální API) na fóru LibriVox. Po pročtení tohoto vlákna obsahujícího diskuze uživatelů a administrátorů jsem došel k následujícím závěrům:

- API je v nedokončeném stavu, chybí několik základních funkcionalit a mnoho dalších uživatelů má s jeho použitím problémy.
- Vývoj (rozšíření) API je pozastaven a v dohledné době se neplánuje změna.
- Při vývoji současného API došly dobrovolníkům spravujícím LibriVox finance, a veškeré současné zdroje se uplatňují na financování provozu a zajištění stálého nahrávání nových knih.
- Administrátoři LibriVox nemají legální přístup k funkcionalitě současného API a nemohou v něm provádět změny navrhované uživateli.

Situaci shrnuje odpověď jednoho z administrátorů jménem Bart na dotaz ohledně stavu API od jiného uživatele:

> *"... While developing the new system we ran out of money, so we couldn't make the API the way we really wanted it to be... ... That's what we have now, nothing more. And there is no indication on when we can actually work on the system again. So if you want to work with this API, please go ahead. We do not have the capacity to help you any further... "* [14]

Při analýze jsem zjistil, že několik uživatelů si pro své účely vytvořilo vlastní katalogy (databáze) časově náročným stažením informací z katalogu LibriVox přes stávající API. Tyto katalogy pak pravidelně aktualizovali a poskytli k dispozici. Několik uživatelů také do svých katalogů zkombinovalo informace získané s API LibriVox spolu s informacemi získanými ze stránek ***Archive.org***, na kterých má LibriVox uložen veškeré své (audio) soubory.

Vytváření vlastní databáze by pro mě bylo velmi (časově) náročné a nevhodné z hlediska implementace do stávajícího systému (nutnost umístění databáze na server apod.). Nebylo možné ani využít náhradní databáze vytvořené jinými uživateli. Proto jsem se rozhodl vyzkoušet alternativní metodu - pokusit se zcela obejít nespolehlivé API LibriVox, a získávat veškeré informace z existujícího a kvalitního API *Archive.org*.

2.2 The Internet Archive

The Internet Archive[14] (česky "internetový archiv"nebo "archiv internetu", dále jen **TIA**) je nezisková digitální knihovna, založená roku 1996 v San Francisku, USA. Nabízí veřejný přístup k digitalizovaným dokumentům (volným dílům), audio souborům, videím, softwaru a archivovaným verzím webových stránek. Krom toho, že fungují jako úschovna těchto archiválií, také sami organizují digitalizaci dokumentů, sbírání a automatizovanou archivaci webových stránek pomocí web-crawlerů[15] a další aktivity spojené s prezervací digitálních informací a propagací svobodného a otevřeného webu.

Obrázek 2.2: Oficiální logo projektu *The Internet Archive*
Zdroj: `http://www.logoeps.net/internet-archive-logo.html` (14.5.2016)

LibriVox úzce spolupracuje s TIA - veškeré knihy a soubory LibriVoxu jsou uloženy právě na jejich serverech. V databázi TIA jsou pak knihy zařazeny v kolekci "*The LibriVox Free Audiobook Collection*"- každou knihu z databáze LibriVox je možné na stránkách TIA vyhledat, stáhnout či přehrát. Na webové stránce každé knihy lze najít název knihy, autora, stručný popis, seznam kapitol a další informace, včetně přehrávače a seznamu souborů ke stažení.

2.2.1 Identifikátory

Při implementaci testovacího programu (viz sekce 3.2.1) jsem již byl nucen pracovat s prostředím TIA. API LibriVox totiž ve své XML (i jakékoliv jiné) odpovědi nenabízí u jednotlivých knih žádné URL pro stažení jejích kapitol, pouze URL archivu ve formátu *.zip*, který je uložený na stránkách TIA a obsahuje všechny kapitoly dané knihy ve formátu *64kbps MP3*[16]. Toto

[14] *The Internet Archive* - `https://archive.org/index.php`
[15] *Web crawler* - software, který automaticky prochází internetové stránky a sbírá/analyzuje jejich obsah, typicky za účelem indexace. [15]
[16] *kbps* - "kilobit per second", česky "kilobit za sekundu". Jednotka přenosové rychlosti. U audio souborů v kódování MP3 je jedním z hlavních faktorů ovlivňujících kvalitu záznamu.

omezení by znamenalo, že by uživatelé programu neměli možnost stahovat u knih jejich samostatné kapitoly, ale pouze celou knihu v nižší zvukové kvalitě.

Při analýze možností, jak obejít toto omezení, jsem zjistil, že ani není možné vygenerovat URL pro stažení jednotlivých kapitol za využití URL archivu s celou knihou - každá kniha totiž měla jiný tvar identifikátoru pro své jednotlivé kapitoly. Pro představu uvádím příklady identifikátorů pro tři různé knihy (identifikátory jsou extrahované přímo z URL jednotlivých souborů, první část je vždy kořenový adresář knihy, tučně pak samotný identifikátor):

- *The Odyssey*, autor: Homér
  - Identifikátor pro stažení archivu s celou knihou: /odyssey_butler_librivox/**odyssey_butler_librivox**
  - Identifikátor pro stažení kapitol (*xx* je číslo ve formátu *01, 02, ...*): /odyssey_butler_librivox/**odyssey_*xx*_homer_butler**
- *Romeo and Juliet (LibriVox Version 2)*, autor: William Shakespeare
  - Identifikátor pro stažení archivu s celou knihou: /romeo_juliet_1001_librivox/**romeo_juliet_1001_librivox**
  - Identifikátor pro stažení kapitol (*x* je číslo ve formátu *0, 1, ...*): /romeo_juliet_1001_librivox/**romeo_*x*_shakespeare**
- *The Adventures of Sherlock Holmes*, autor: Sir Arthur Conan Doyle
  - Identifikátor pro stažení archivu s celou knihou: /adventures_holmes/**adventures_holmes**
  - Identifikátor pro stažení kapitol (*xx* je číslo ve formátu *01, 02, ...*): /adventures_holmes/**adventureholmes_*xx*_doyle**

Tento problém jsem obešel s využitím souboru *files*.xml (který se vždy nachází v kořenovém adresáři knih). Ten obsahuje metadata[17] o všech souborech uložených v adresáři knihy na serveru TIA. Více o vyřešení problému pak v sekci 3.7.1.

2.2.2 The Internet Archive Search API

Hlavním důvodem pro uvedení TIA v mojí práci je použití jeho vyhledávacího API.[18] Jelikož se veškeré soubory knih z databáze LibriVox nacházejí právě na servech TIA, bylo možné využít jejich API pro vyhledávání knih. Zároveň díky tomu, že TIA u každé knihy ukládá i informace o dílu (nejen o souborech), bylo možné API LibriVox zcela obejít.

API TIA je veřejné, dobře zdokumentované a nabízí vývojářům mnoho možností využití. Jeho dokumentace zároveň obsahuje intuitivní nástroje pro vytváření dotazů a rozšířené možnosti vyhledávání, včetně řazení výsledků a volby formátu odpovědi. Shrnutí vybraných možností vyhledávání:

[17] *metadata* - data obsahující informace o jiných datech.

[18] *The Internet Archive* Search API - `https://archive.org/advancedsearch.php#raw`

- Vyhledávání podle názvu knihy, jména autora, popisu atd. (tak, že zadaný výraz může být kdekoliv v názvu knihy, jménu autora, ...).
- Libovolný výběr polí (informací) k navrácení v odpovědi (omezení navrácených informací jen na potřebné).
- Omezení počtu položek (knih) navrácených v odpovědi.
- Seřazení položek navrácených v odpovědi např. podle názvu, data nebo počtu stažení (sestupně i vzestupně). Možnost až tří navazujících libovolných řazení.
- Volba formátu odpovědi - k dispozici formáty JSON, XML, CSV a další.
- Omezení hledání pouze na položky LibriVox díky specifikaci *subject:(librivox)*.
- *Fuzzy Query* - možnost vyhledávání podle slov podobných slovu zadanému.

Nabídka možností vyhledávání je ještě větší, ale pro stručnost uvádím jen ty, které jsem využil. Je zjevné, že možnosti API TIA vysoce převyšují ty, které nabízí API LibriVox. Jedinou nevýhodou tohoto API je pak absence informace o roku vydání (kterou lze kvalitativně nahradit datem publikace nahrávky daného díla) a jazyku díla (která je mírně problematická z hlediska převodu textu na řeč).

Zvoleným parametrům vyhledávacího řetězce se věnuji v kapitole 3.6.3 o implementaci vyhledávání, zde uvádím (šablonovitý) příklad dotazu. Kořenová adresa pro dotazy na vyhledávací API je:

```
https://archive.org/advancedsearch.php
```

Za touto adresou pak vždy následuje samotný strukturovaný dotaz. Ten se například sestává z následujících částí (rozdělených pro přehlednost popisu do položek seznamu):

- ?q=creator:(*jméno_autora*)+AND+subject:(librivox)
  - **?q=** - identifikace dotazu
  - **creator** - typ dotazu (pole, se kterým se porovnává zadané slovo). Dalšími možnostmi jsou např. *title, description, ...*
  - **subject:(librivox)** - v našem případě fixní specifikace toho, že chceme pouze knihy z projektu LibriVox. Každá kniha nahraná v TIA a pocházející z projektu LibriVox má vždy v poli *subject* klíčové slovo *"librivox"*.
- &fl[]=creator&fl[]=title&sort[]=titleSorter+asc&sort[]=&sort[]=
  - **fl[]=creator** a **fl[]=title** - specifikace toho, jaká pole chceme u každé knihy v odpovědi API. Polí může být z dostupného výběru libovolný počet, zde je například specifikováno, že chceme navrátit název knihy (*title*) a jméno autora (*creator*)

- **sort[]=titleSorter+asc** - řazení výsledků. Identifikátory *sort* jsou vždy tři a každý z nich může být vždy následován identifikátorem typu řazení. V uvedeném případě je to například *titleSorter*, tedy řazení podle názvu díla a dodatek *+asc*, který specifikuje, že řazení bude vzestupné (druhá možnost je *desc*, tedy řazení sestupné).

- &rows=50&page=1&callback=callback&output=xml#raw
  - **rows=50** - max. počet řádků, tedy nalezených položek, které se zobrazí v odpovědi API. Zde je tedy počet omezen na padesát položek (knih).
  - **page=1** - specifikace strany. Výsledek vyhledávání může být rozdělen do několika stran, a tato volba určuje jakou stranu chceme navrátit.
  - **output=xml** - určení formátu odpovědi, v tomto případě je tedy zvolen formát XML.

2.3 Alternativní systémy

Současný systém [4] není jediným řešením pro nevidomé uživatele. Dalším je například americký *Odinmobile*[19], nabízející levné telefony s klasickou fyzickou klávesnicí, které ale nejdou cestou moderního trendu chytrých telefonů.

Tou se naopak vydávají francouzská zařízení *Claria*[20], která používají na svých telefonech (řady *Claria Vox* pro nevidomé) speciální masku s otvory, které odpovídají tlačítkům klávesnice. Pod maskou se pak nachází samotný smartphone.

Chytré telefony používá i další francouzská firma *Kapsys*[21] se přístroji řady *SmartVision* a *Smartvision Lite*. Ty kombinují dotykový displej s hardwarovou klávesnicí a jsou postaveny na platformě Android. Nabízí mnoho funkcí a pokročilých aplikací, vše ale vyvažují vysokou cenou.

Na nevidomé uživatele částečně cílí i americký *Apple* se svými telefony *iPhone*[22], které nabízejí například vestavěnou obrazovkovou čtečku *VoiceOver* a inteligentní pomocnici *Siri*, která pomáhá s ovládáním telefonu za pomocí hlasu. Popularita těchto zařízení je relativně vysoká navzdory své vysoké ceně.

Mezi nevidomými jsou dlouhodobě populární (starší) telefony značky *Nokia* s nativní podporou funkcí a přizpůsobením pro zrakově postižené. S končící životností těchto telefonů však uživatelé často hledají náhradu.

[19] *Odin Mobile* - `http://odinmobile.com/`
[20] *Claria* - `http://www.claria-vision.com/en/`
[21] *Kapsys* - `http://www.kapsys.com/fr/cs/#`
[22] *iOS Accessibility* - `http://www.apple.com/accessibility/ios/#vision`

Kapitola 3
Implementace

3.1 Prostředí a nástroje

Celou práci jsem vytvářel v IDE[1] **Android Studio**[2] od společnosti Google, vývojovém prostředí vytvořeném specificky pro programování a vývoj aplikací na platformu Android. Verze Android Studia na kterých probíhal vývoj aplikace byly zejména *v1.5.0* a *v1.5.1*, při dokončování práce pak i verze *v2.0.0* a *v2.1.0*. Při vývoji jsem také využíval verzovacího systému *Git* skrze webovou službu *Bitbucket*[3] a jejich desktopovou aplikaci *SourceTree*[4], a to jak při vytváření testovacího programu tak i při převádění vytvořené aplikace do prostředí stávajícího systému.

Samotná práce je napsaná v programovacím jazyce *Java* společně se soubory textových řetězců a rozvržení (designu) v *XML*. Debugování a vývoj testovací aplikace probíhal na telefonech *Samsung Galaxy* S3 mini (s verzí operačního systému Android 4.1 - Jelly Bean) a *Samsung Galaxy* S5 Neo (s verzí OS Android 5.1 - Lollipop). Integrace aplikace do prostředí stávajícího systému společně s dalším laděním pak už probíhala na telefonu *Samsung Galaxy* Core 2 (s verzí OS Android 4.4.2 - Kitkat).

3.2 Analýza

Před samotnou implementací programu jsem provedl analýzu úkolu. Kvůli tomu, že jsem neměl žádné předešlé zkušenosti s programováním pro Android ani s prací v IDE Android Studio, rozhodl jsem se v první fázi vytvořit testovací aplikaci, která bude implementovat všechny funkce a logiku, ale bude obalena klasickým uživatelským rozhraním, které si vytvořím. Tímto způsobem bylo snazší naučit se základy programování aplikací pro operační systém Android, zejména tedy pochopení struktury takovýchto aplikací, provázanost mezi logikou a uživatelským rozhraním, využití dostupných tříd

[1] *IDE* - (z ang. Integrated Development Environment) vývojové prostředí

[2] Android Studio - *Meet Android Studio* - `https://developer.android.com/studio/intro/index.html`

[3] *Bitbucket* - `https://bitbucket.org/`

[4] *SourceTree* - `https://www.sourcetreeapp.com/`

a metod pro tvorbu aplikace a používání IDE Android Studio včetně ladění pomocí připojeného chytrého telefonu.

V druhé fázi jsem pak tuto testovací aplikaci, po vyladění fundamentální funkcionality, převedl do uživatelského prostředí stávajícího systému a upravil její použitelnost s ohledem na ovládání a celkové chování v daném prostředí.

Před započetím samotného vývoje jsem pak v rámci spolupráce projektu stávajícího systému se Sjednocenou organizací nevidových a slabozrakých (*SONS*[5]) konzultoval plánovanou strukturu a funkcionalitu aplikace. S pracovníky centra jsem prodiskutoval navržené řešení a vyhodnotil následující návrhové koncepty:

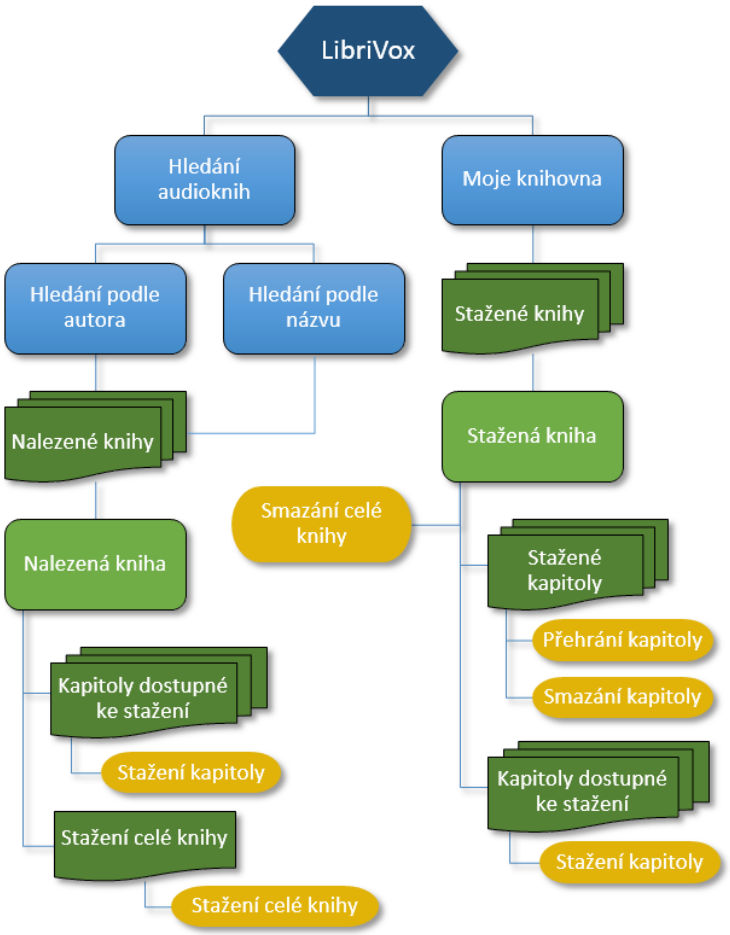
- Důraz na jednoduchost použití nevidomými uživateli.
- Preference minimalistického přístupu při návrhu struktury uživatelského rozhraní.
- Podpora stahování samostatných kapitol a celé knihy.
- Co nejvyšší automatizace aplikace v procesu stahování a vyhledávání.

Poté jsem vytvořil návrh struktury aplikace, vyobrazený na obrázku 3.1.

Tmavě modrý kořenový prvek LibriVox reprezentuje výchozí bod aplikace, světlé modré jsou pak položky v nabídce (menu). Tmavě zelené vrstvené panely reprezentují seznamy (listy) položek, světle zelené jsou pak položky z těchto seznamů. Pro jednoduchost jsem v diagramu neuvedl světle zelené položky v seznamech "stažené kapitoly"a "kapitoly dostupné ke stažení", a rovnou jsem z nich vyvedl okrové panely symbolizující dialogy (volby, akce) na vybraných položkách. Tento diagram v principu reprezentuje jak strukturu uživatelského rozhraní u konečné aplikace tak její funkcionalitu. Tou je tedy:

- Vyhledávání audioknih ve webovém katalogu LibriVox:
  - Podle jména autora knihy.
  - Podle názvu knihy.
- Zobrazení (seřazeného) seznamu nalezených audioknih.
- Zobrazení možností stahování vybrané audioknihy:
  - Výčet samostatných kapitol dostupných ke stažení.
  - Volba stažení celé audioknihy.
- Seznam již stažených audioknih (ať už kompletně či částečně) s možností mazání celých knih z paměti.
- Zobrazení seznamu již stažených kapitol s možností smazaní či přehraní kapitoly.
- Zobrazení seznamu kapitol dostupných ke stažení s možností jejich okamžitého stažení.

[5] *Sjednocená organizace nevidomých a slabozrakých ČR* - `http://www.sons.cz/`



Obrázek 3.1: Návrh struktury aplikace LibriVox

Mimo diagram jsem ještě na konci své práce vytvořil vlastní, jednoduchý přehrávač audio souborů pro přehrávání kapitol stažených knih (viz sekce 3.9). Na diagramu pak také nejsou vyobrazeny žádné principy funkcionality nebo jaké informace se skutečně v jednotlivých sekcích zobrazují - o tom více v dalších částech mojí práce, které se těmto sekcím věnují.

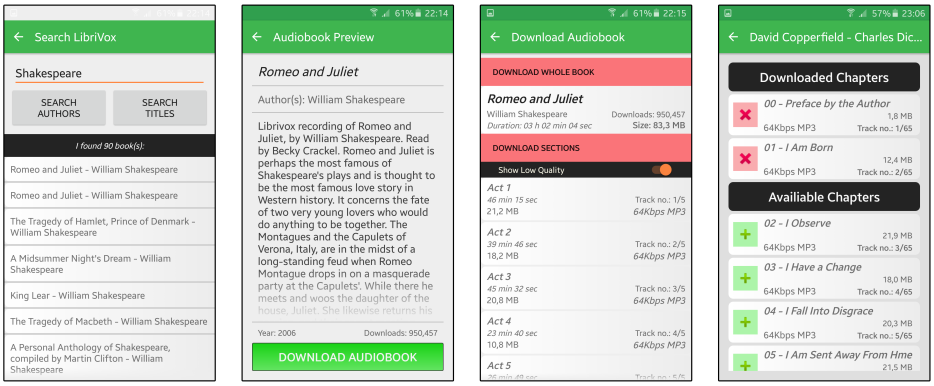
3.2.1 Testovací aplikace

Implementaci konečné aplikace předcházel vývoj testovacího programu ve standardním uživatelském rozhraní pro vidící. V rámci této aplikace jsem naimplementoval všechny funkce a procesy finální práce - ovšem v rozhraní, které mi umožnilo snadné ladění a bylo přitom jednodušší na vyhotovení (s ohledem na mojí nezkušenost s programováním pro platformu Android), než kdybych měl svojí práci přímo vyhotovovat v rozhraní současného systému.

Při vývoji testovací aplikace (kterou jsem pracovně pojmenoval *LibriVox Test*) jsem pak postupoval tak, aby její pozdější převedení (respektive převedení její funkcionality) bylo co nejsnazší - například díky oddělení tříd s

logikou a tříd s uživatelským rozhraním. Po jejím dokončení jsem pak vyladil hlavní nedostatky, a převedl jsem ji společně s veškerou funkcionalitou. Poté co byla aplikace převedena (s množstvím změn souvisejícím zejména s odlišným uživatelským rozhraním) se již základní funkcionalita v jádru neměnila. Proto nebudu odděleně popisovat implementaci testovací aplikace (která ostatně tvoří jen mezikrok mojí práce) a konečné aplikace - je totiž prakticky totožná.

Na následujícím obrázku pak uvádím ukázkové záznamy obrazovek z typických částí mé testovací aplikace. Design uživatelského rozhraní je prostý a navržený jen pro účely ladění.



Obrázek 3.2: Ukázka typických obrazovek z testovací aplikace

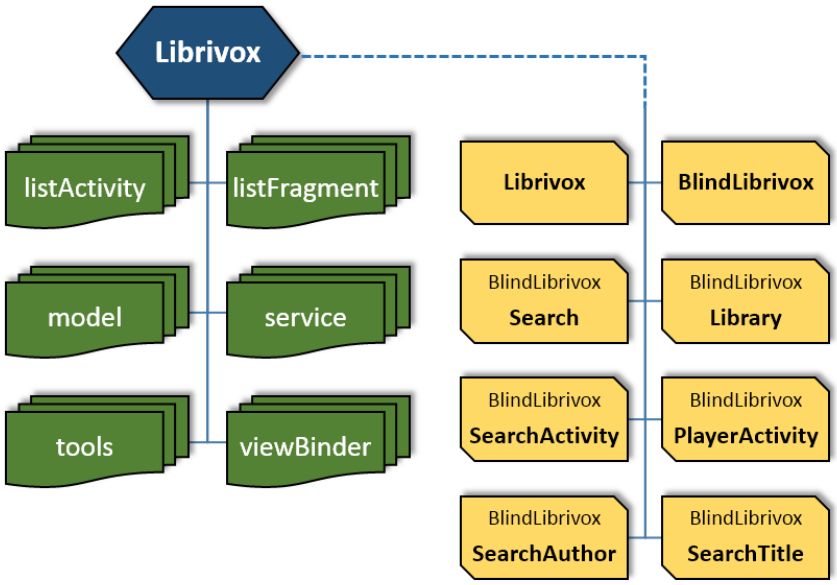
Panely na obrázku jsou pak záznamy obrazovek (vytvořené na mobilním telefonu Samsung Galaxy S5 Neo) které zleva doprava zobrazují následující součásti aplikace:

- Vyhledávání v databázi LibriVox pomocí jména autora s použitím slova "Shakespeare"a zobrazení seznamu nalezených knih.
- Zobrazení informací o nalezené knize (vybrané ze seznamu nalezených knih) - název díla, jméno autora, popis knihy (v anglickém jazyce), rok publikace do audioknihovny LibriVox a počet stažení. V dolní části pak tlačítko pro přechod do nabídky stahování.
- Nabídka stahování - v horní části tlačítko pro stažení archivu s celou knihou v nízké kvalitě, včetně délky celé audioknihy a velikosti archivu. V dolní části se pak nachází seznam jednotlivých kapitol knihy s jejich jmény, čísly a formáty (kvalitou). Také je zobrazena velikost jednotlivých souborů s kapitolami a jejich délka (záznamu). Nad seznamem kapitol je ještě přepínač pro zobrazování kapitol v nízké kvalitě.
- Náhled kapitol knihy v uživatelské knihovně. Kapitoly jsou rozděleny podle toho, zda-li jsou již stažené nebo ne. U kapitol jsou opět uvedeny jejich informace a tlačítka pro smazání či stažení.

Dalšími neuvedenými součástmi aplikace jsou pak hlavní menu (s tlačítky pro přechod k vyhledávání audioknih nebo procházení uživatelské knihovny) a seznam s knihami v uživatelské knihovně.

3.3 Struktura souborů aplikace

Základní struktura uživatelského rozhraní a funkcionality aplikace byla již představena v části 3.2. Struktura kořenového adresáře se složkami (balíky, ang. *packages*) a třídami je pak vyobrazena na diagramu 3.3.



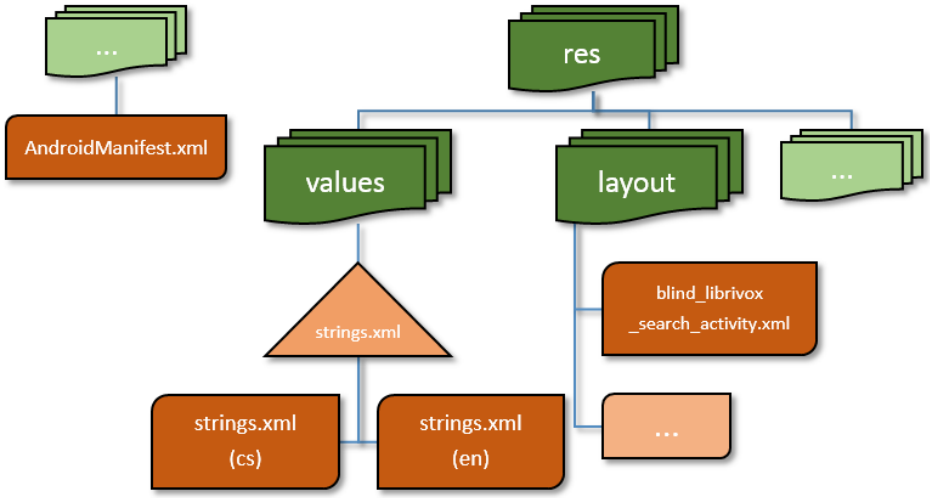
Obrázek 3.3: Struktura kořenového adresáře aplikace s podsložkami a třídami

Modrým pozadím a šestiúhelníkovým tvarem je označen kořenový balík/adresář aplikace, zeleně pak balíky uvnitř kořenového balíku (každý obsahuje vlastní třídy) a žlutě pak samostatné třídy. Tučně je vždy vyznačen podstatný název třídy (celý název s prefixem *BlindLibrivox* - u příslušných tříd - je samozřejmě jednoslovný, rozdělení na dva řádky je pouze z praktických důvodů zobrazení).

Soubory (třídy) aplikace jsou rozděleny do balíku podle svého typu, např. balík ***tools*** obsahuje nástroje - třídy s logikou pro jednotlivé funkce aplikace. Obsahy těch balíků jejichž třídy souvisí přímo s logikou aplikace rozvedu hlouběji v sekcích o implementaci konkrétních funkcí, o balících a třídách tvořících zejména uživatelské rozhraní se zmíním také, ale stručněji.

Součástí aplikace jsou také soubory ve formátu XML, které obsahují zdrojová data pro aplikaci v podobě textových řetězců (*strings*.xml) a souborů s rozvržením uživatelského rozhraní (soubory v adresáři *layout*). Všechny zdroje se nacházejí v příslušných složkách adresáře ***res*** (zkratka z ang. *resources*). Struktura těchto složek (balíků) v aplikaci je pevně daná.

Zjednodušené schéma reprezentuje diagram 3.4.



Obrázek 3.4: Zjednodušená struktura souborů se zdroji

Zeleně jsou vyznačeny složky (balíky), tmavě oranžově soubory ve formátu XML. Trojúhelníkový symbol značí balík se soubory textových zdrojů v jednotlivých jazycích. V adresáři *layout* je množství souborů s popisem rozvržení uživatelského rozhraní, pro představu uvádím jeden a existence dalších je vyjádřena symbolem s třemi tečkami - stejně jako u dalších balíku zdrojových souborů v adresáři *res*.

3.3.1 Android Manifest

Soubor ***AndroidManifest*** obsahuje základní informace o aplikaci (název aplikace, popisky komponent, povolení, min. potřebná úroveň Android API) a je její povinnou součástí. Pro správnou funkci aplikace je potřeba, aby v něm byly zapsány následující položky:

- Deklarace aktivit (které implementují části uživatelského rozhraní):
  - Seznamy (...`ListActivity`)
  - Vyhledávání (...`SearchActivity`)
  - Hlavní menu (...`BlindLibrivox`)
  - Přehrávač (...`PlayerActivity`)
- Deklarace služeb (*services*):
  - Rozbalování archivů (...`UnzipService`)
  - Upozornění po stahování (...`DownloadNotificationService`)
- Povolení (*permissions*):
  - Pro zápis na externí paměťové úložiště (`WRITE_EXTERNAL_STORAGE`)

- Pro čtení z externího paměťového úložiště (`READ_EXTERNAL_STORAGE`)
- Pro stahování bez upozornění (`DOWNLOAD_WITHOUT_NOTIFICATION`)
- Pro přístup k internetu (`INTERNET`)

K některým službám a aktivitám uživatelského rozhraní se pak vracím v dalších částech popisu implementace.

3.3.2 Soubory s textovými zdroji

V souborech ***strings***.xml se nacházejí veškeré texty (řetězce znaků, ang. *string*), které aplikace používá v rámci uživatelského rozhraní (jinými slovy všechny nápisy a texty viditelné pro uživatele). Ty jsou uloženy separátně od funkčního kódu aplikace pro snadný přístup a následnou úpravu, a zejména pro možnost jejich snadného překladu do dalších jazyků. Existují tak jednotlivé soubory textových zdrojů pro různá jazyková prostředí. V rámci své aplikace jsem vytvořil dva soubory *strings*, a to s českými a anglickými textovými zdroji. Jejich součástí jsou veškeré popisy pro uživatelské rozhraní aplikace, včetně textů nápovědy.

Ukázka části textových dat v souboru *strings*.xml (v anglickém jazyce):

```
<!-- Librivox app -->
<string name="app_librivox">Librivox</string>
<string name="app_librivox_author">Author</string>
<string name="app_librivox_author_null">Unknown</string>
<string name="app_librivox_year">Year</string>
<string name="app_librivox_book">Book</string>
<string name="app_librivox_description">Description</string>
<string name="app_librivox_chapter">Chapter</string>
<string name="app_librivox_duration_hours">Hours</string>
<string name="app_librivox_duration_minutes">Minutes</string>
<string name="app_librivox_duration_seconds">Seconds</string>
<string name="app_librivox_duration_hours_abbr">h</string>
<string name="app_librivox_duration_minutes_abbr">min</string>
<string name="app_librivox_duration_seconds_abbr">sec</string>
<string name="app_librivox_size_bytes">Bytes</string>
<string name="app_librivox_size_kilobytes">Kilobytes</string>
<string name="app_librivox_size_megabytes">Megabytes</string>
<string name="app_librivox_size_gigabytes">Gigabytes</string>
<string name="app_librivox_no_sdcard">There is no SD card inserted in the device.</string>
<string name="app_librivox_chapter_number_from">of</string>
```

Obrázek 3.5: Ukázka textových dat v souboru *strings*.xml

3.3.3 Soubory s rozvržením uživatelského rozhraní

V balíku ***layout*** se nachází XML soubory, ve kterých jsou definovány rozvržení jednotlivých částí uživatelského rozhraní aplikace. K jejich popisu se pak vrátím v příslušných sekcích. Jsou zde následující soubory rozvržení (*layout*):

- 4 soubory fungující jako propojení seznamu (*list_fragment*) s aktivitou (*fragment_activity*)
- 3 soubory rozvržení jednotlivých položek seznamů (*list_fragment_item*)

- 2 soubory rozvržení uživ. rozhraní aktivit (přehrávače a vyhledávání)

Díky tomu, že aplikace je vyvíjena specificky na jeden daný typ mobilního telefonu (s danou verzí systému Android), není nutné diferencovat několik souborů rozvržení pro různé orientace zařízení nebo velikosti obrazovky.

3.4 Ovládání a vnější nástroje

Ovládání a pohyb mezi nabídkami v aplikaci Librivox je (až na výjimky) stejný jako v celém systému (dotyky se mohou realizovat kdekoliv na obrazovce):

- Dlouhý dotyk jedním prstem pro potvrzení (ve smyslu "ano") dialogu a vyvolání nebo vstup do nabídky.
- Dlouhý dotyk dvěma prsty pro zrušení (ve smyslu "ne") dialogu a návrat zpět z aktuální nabídky.
- Krátký dotyk dvěma prsty pro zopakování (přečtení) aktuálního textu (informace) na obrazovce.
- Krátký dotyk jedním prstem na levé či pravé straně obrazovky pro pohyb v seznamu položek či mezi panely nabídky vlevo a vpravo (vpřed a vzad).
- Dlouhé stisknutí tlačítka napájení (ang. *power button*) pro vyvolání nápovědy.
- Krátký stisk tlačítka napájení pro zamknutí obrazovky.

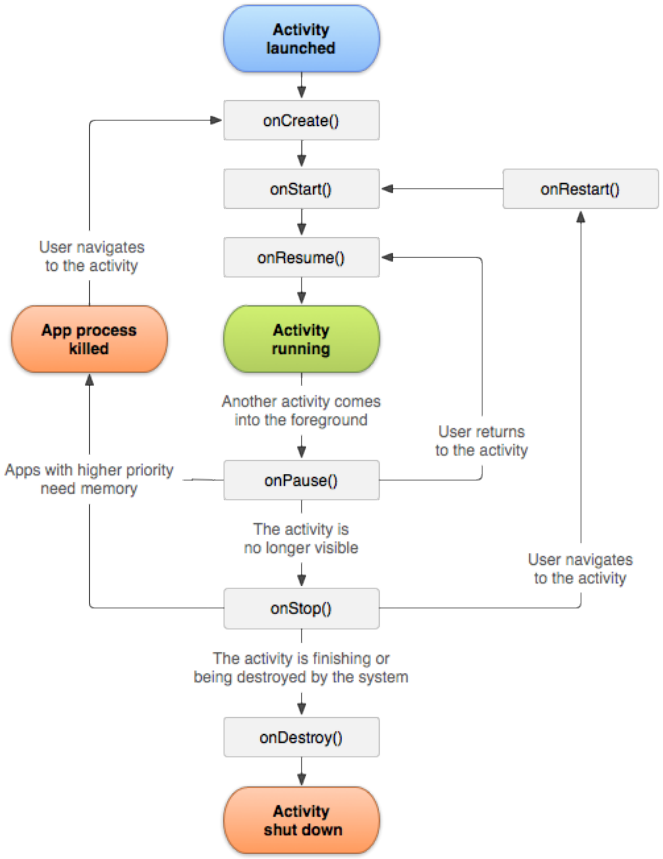
Dotyky na obrazovce (a stisky tlačítek) nadřazený systém vyhodnocuje a zaznamenává zejména pomocí třídy `GestureDetector`, dalším třídám pak dává k dispozici skrze její rozhraní metody jako `confirmSelection`, `tapTwoFingers` atd. pro implementaci akcí, které se mají provést při příslušných gestech.

Některé z těchto gest mají již v základních třídách stávajícího systému (od kterých pak dědí třídy aplikace LibriVox) výchozí implementaci. V příslušných kapitolách nebudu hlouběji rozebírat implementace všech gest v různých částech aplikace, ale pouze ty, u nichž se naprogramovaná akce nějakým způsobem výrazně liší od té výchozí, nebo je jinak důležitá pro běh aplikace. U ostatních gest se pak tedy předpokládá výchozí funkčnost.

Základní třídy (pro tvorbu uživatelského rozhraní) stávajícího systému pak také dávají k dispozici metody životního cyklu aktivit[6] (a fragmentů[7]) v prostředí Android, protože dědí například od třídy `FragmentActivity` která sama dědí od třídy `Activity` apod.

[6] *Android Developers* - Activity - `https://developer.android.com/reference/android/app/Activity.html`

[7] *Android Developers* - Fragment - `https://developer.android.com/reference/android/app/Fragment.html`



Obrázek 3.6: Životní cyklus aktivity (autor: Google Inc.)
Zdroj: `https://developer.android.com/reference/android/app/Activity.html` (21.5.2016)

Metody životního cyklu mají v třídách, od kterých dědí třídy aplikace LibriVox, vlastní implementaci která zajišťuje správný chod v rámci prostředí nadřazeného systému. V některých případech však tuto implementaci v třídách rozšiřuji o potřebnou funkcionalitu pro můj program (a implementaci metody v rodičovské třídě zachovávám pomocí `super`).

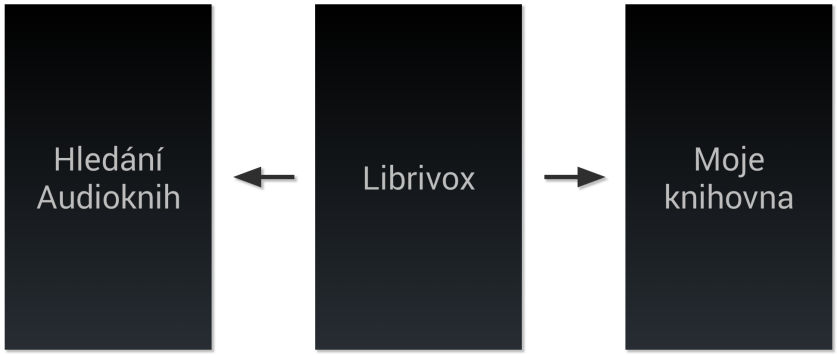
3.4.1 Hlasová syntéza

V programu také hojně využívám služeb hlasové syntézy (ang. *Text-to-speech*, zkr. TTS) pro čtení textů na obrazovce a dalších informací (pro interakci s nevidomými uživateli). Tu kompletně zajišťuje třída `BlindTts` se svými statickými metodami pro hlasovou syntézu, zejména s metodou `speak`, která bere na vstupu textový řetězec a zajišťuje jeho přečtení hlasovým syntezá-

torem. Pokud se v dalším textu budu zmiňovat ve smyslu toho, že aplikace nějaký text "přečte", je tím myšleno právě použití hlasové syntézy.

3.5 Hlavní nabídka

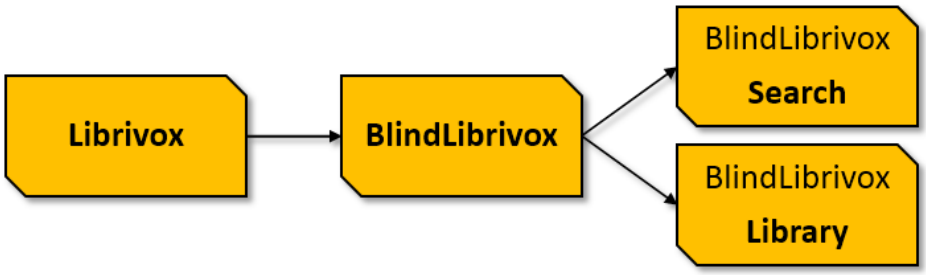
Popis implementace začíná v hlavní nabídce aplikace. Do té se vstupuje přes panel s fixním nápisem *Librivox*.



Obrázek 3.7: Panely hlavní nabídky aplikace LibriVox

Vstup do hlavního menu zajišťují třídy `Librivox` a `BlindLibrivox`. První zmiňovaná třída implementuje rozhraní `App` a funguje jen jako panel jehož obsahem je jediné textové pole (komponenta `TextView`), které zobrazuje název aplikace. Jediná důležitá metoda je `getAppClass` s návratovou hodnotou typu `Class`. Třída kterou navrací je ta třída (aktivita), která se spustí po dlouhém dotyku jedním prstem na obrazovce s panelem Librivox - v našem případě je to právě `BlindLibrivox`.

Třída `BlindLibrivox` funguje jako kontejner pro hlavní nabídku aplikace, jejíž obsahem jsou dva panely - *Hledání Audioknih* (třída `BlindLibrivox-Search`) a *Moje knihovna* (třída `BlindLibrivoxLibrary`). `BlindLibrivox` dědí od třídy `BlindAppItem`, která zajišťuje propojení s panely tvořícími obsah hlavní nabídky. Při načtení třídy se v metodě `onResume` inicializuje text nápovědy pro hlavní nabídku (uložený v souborech *strings*.xml).



Obrázek 3.8: Hierarchie tříd hlavní nabídky

Rodičovské třídě se při inicializaci v metodě `onCreate` přiřadí do proměnné `mSectionsPagerAdapter` nová instance vnitřní třídy `SectionsPagerAdapter`, která dědí od `FragmentStatePagerAdapter`[8]. V konstruktoru vnitřní třídy se deklaruje mapa (`LinkedHashMap`) do níž jsou pak vloženy nové instance tříd fragmentů. Tyto fragmenty jsou právě položkami v hlavní nabídce (tedy třídy `BlindLibrivoxSearch` a `BlindLibrivoxLibrary`). Celá konstrukce zajišťuje zobrazení panelů hlavní nabídky jako seznamu, který lze procházet. Metoda `getItem` vrací fragment (panel) pro odpovídající pozici v seznamu (v mapě a nabídce).

Poslední činností, která se provede při inicializaci třídy `BlindLibrivox` v metodě `onCreate`, je vyčištění dat aplikace uložených v preferencích. O ukládání informací do sdílených preferencí (*SharedPreferences*) bude řeč později. Čištění se provádí v metodě `preferencesInit`, která v novém vlákně získá instanci systémové služby `DownloadManager` a zjistí pomocí metody `query`, jestli právě neprobíhá nějaké stahování (ať už aktivní, čekající nebo pozastavené). Pokud ne, data uložená v preferencích, která se používají při rozbalování stažených archivů třídou `LibrivoxUnzipService` (viz sekce 3.7.2), se vymažou.

V případě, že by se data smazala (například při odchodu a návratu do aplikace) při probíhajícím stahování archivu s celou knihou, nebylo by pak možné tento archiv správně rozbalit do příslušného adresáře. Data se tak pročišťují pravidelně při každém vstupu do aplikace, když není aktivní stahování. Pročišťování dat je zajištěno zejména kvůli tomu, aby po delším používání aplikace nebylo v preferencích příliš datových položek, což by působilo zpomalování jejich procházení.

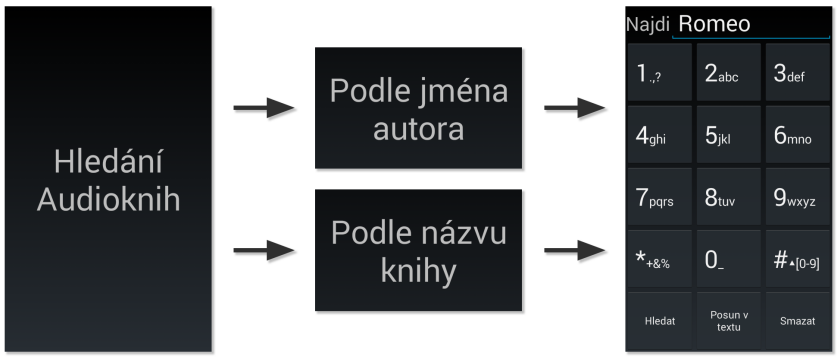
3.6 Vyhledávání

Do nabídky vyhledávání se vstupuje přes panel "Hledání audioknih"(třídu `BlindLibrivoxSearch`). Třída `BlindLibrivoxSearch` funguje (analogicky k třídě `BlindLibrivox`) jako kontejner, který přes vnořenou třídu (která dědí od `FragmentStatePagerAdapter`) nastavuje nabídku pro sekci vyhledávání. Tu tvoří dvě položky - (vyhledávání) podle názvu knihy (třída `BlindLibrivoxSearchTitle`) a podle jména autora (třída `BlindLibrivoxSearchAuthor`). Obě třídy jen tvoří panely v uživatelského rozhraní, které obsahují jediné textové pole a nemají žádnou další funkcionalitu. Liší se zobrazovaným textem a hodnotou, jež při výběru z nabídky předávají třídě `BlindLibrivoxSearchActivity`, kterou spouští.

Ve třídě `BlindLibrivoxSearchActivity` je deklarovaný `enum` s hodnotami `TITLE` a `AUTHOR`. Před spuštěním vyhledávací aktivity se do příslušného `Intent` přidá `extra` v podobě typu vyhledávání (`TITLE` nebo `AUTHOR`). Po spuštění vyhledávací aktivity (`BlindLibrivoxSearchActivity`) se `extra` s typem vyhledávání získá zpět a uloží se do globální proměnné třídy. Celé

[8] *Android Developers* - FragmentStatePagerAdapter - `https://developer.android.com/reference/android/support/v4/app/FragmentStatePagerAdapter.html`

vyhledávání (včetně textu výzvy k zadání slova k vyhledání) se nastaví podle zvoleného typu.



Obrázek 3.9: Panely nabídky vyhledávání

Struktura vyhledávání je velmi jednoduchá - uživatel si zvolí, že chce vyhledávat audioknihy a následně vybere typ vyhledávání. Po tomto výběru se objeví obrazovka s klávesnicí a uživatel je vyzván k zadání názvu knihy nebo jména autora.

Implementace dalších typů vyhledávání byla možná (dle možností daných API - viz sekce 2.2.2), omezil jsem se však pouze na hledání podle jména autora a názvu knihy zejména pro výslednou štíhlost nabídek aplikace. Pokud by se ukázalo, že uživatelé vyžadují více možností vyhledávání, jejich dodatečná implementace je velmi snadná.

3.6.1 Model audioknihy

Třída `LibrivoxAudioBook` jednoduše reprezentuje audioknihy získané z odpovědi API TIA. Objekty typu `LibrivoxAudioBook` mají pak následující vlastnosti:

- **id** - textový identifikátor, stejný jako používá TIA ve své databázi pro rozlišování subjektů
- **title** - název díla (audioknihy)
- **author** - autor (autoři) díla
- **description** - popis audioknihy
- **year** - rok publikace do systému LibriVox
- **download** - URL adresa, na které se nachází archiv s celou knihou (získává se pomocí kombinace kořenové adresy, textového identifikátoru a postfixu)
- **dlCount** - počet stažení audioknihy ze stránek TIA

- **zipSizeBytes** - velikost archivu s celou knihou v bajtech (používá se pro kontrolu zda-li je v zařízení dostatek místa ke stažení)
- **zipSizeHuman** - velikost archivu s celou knihou v lidmi čitelné podobě
- **duration** - délka celé knihy (délka přehrávání audio souborů)

Ke všem vlastnostem pak existují příslušné *gettery* a *settery*. V některých *setterech* se pak vstupní informace ještě upravují do výhodnější podoby (např. u počtu stažení se doplní čárky oddělující tisíce apod.).

3.6.2 Zadání a zpracování dotazu

Úkolem třídy `BlindLibrivoxSearchActivity` je následující:

- Zobrazení klávesnice.
- Kontrola a zpracování zadaného textu.
- Kontrola internetového připojení.
- Odeslání zadaného textu k vyhledání.
- Zpracování výsledků vyhledávání a spuštění aktivity k zobrazení seznamu výsledků.

Při inicializaci se spustí metoda `onCreate`, ve které se nastaví typ vyhledávání (jak bylo již popsáno výše) a aktivitě se metodou `setContentView` přiřadí příslušné rozvržení rozhraní (definované v souboru XML). Rozhraní tvoří jedno textové pole (`TextView`) s nadpisem vyhledávání (na obrázku 3.9 slovo "Najdi"na panelu vpravo) a jedno pole pro zadávání textu k vyhledání.

Dále se vytvoří nová instance vyhledávacích nástrojů `LibrivoxSearchTools` a nastaví se k nim příslušný posluchač (*listener*). Nakonec se klávesnice inicializuje - nastaví se nápis na tlačítku pro spuštění vyhledávání (na obrázku 3.9 tlačítko klávesnice vlevo dole) a akce po jeho stisknutí. Ta má následující strukturu:

1. Kontrola přístupu k internetu pomocí příkazu *ping*.
2. Získání zadaného textu a jeho kontrola.
3. Spuštění vyhledávání.

Kontrola připojení k internetu

Kontrola přístupu k internetu probíhá pomocí příkazu *ping* na IP adresu `8.8.8.8`, což je adresa veřejných *DNS*[9] serverů společnosti Google. Tato adresa byla vybrána jako etalon přístupu k internetu - servery Google DNS jsou stabilní, přístupné a mají rychlou odezvu. Jinými slovy, pokud není

[9] *DNS* - z ang. *Domain Name System*. Systém, který k sobě přiřazuje názvy internetových domén a IP adresy.

úspěšný *ping* na servery Google DNS, je více než pravděpodobné, že není k dispozici přístup k internetovému připojení. Tento způsob kontroly jsem vybral, protože je jednoduchý a spolehlivý. Při zjišťování přístupu k internetu pomocí kontroly zapnutí WiFi nebo mobilních dat je například nevýhodou, vyjma složitější implementace, že daná funkce může být zapnuta, ale připojení k internetu stejně nemusí být k dipozici.

Programově pak kontrola probíhá nejprve získáním instance třídy `Runtime`[10]. Na té je spuštěn metodou `exec` příkaz ping:

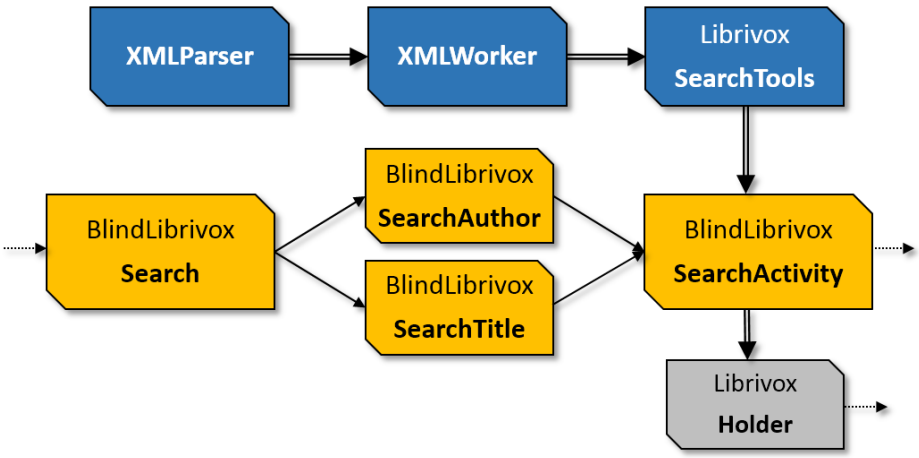
```
/system/bin/ping -c 1 8.8.8.8
```

Parametr *-c* s hodnotou 1 pak určuje, že se *ping* má provést jen jednou. Návratová hodnota (objekt *Process*) metody `exec` se pak uloží do proměnné. Vyčká se na ukončení procesu - při návratové hodnotě 0 byl příkaz *ping* úspěšný a připojení je dostupné. V tom případě program může pokračovat dál - pokud ale připojení k internetu dostupné není, přečte se příslušné oznámení a vyhledávání není spuštěno.

3.6.3 Proces vyhledávání

Při dostupném připojení se ze vstupního pole extrahuje zadaný text. Nejprve se z něj odstraní veškeré nadbytečné řádky (*newline characters*). Poté se zkontroluje, jestli není text zcela prázdný nebo tvořený pouze mezerami - v každém případě se ale přečte oznámení, buď o startu vyhledávání nebo o chybně zadaném výrazu. V novém vlákně (*Thread*) se pak metodou `search` na instanci třídy `LibrivoxSearchTools` spustí vyhledávání.

Na následujícím diagramu je vyobrazena hierarchie tříd vyhledávání:



Obrázek 3.10: Hierarchie tříd vyhledávání

Tmavě žlutou barvou jsou označeny třídy, které pracují s uživatelským rozhraním. Modré jsou třídy, ve kterých je skryta logika vyhledávání. Jedno-

[10] *Android Developers* - Runtime - `https://developer.android.com/reference/java/lang/Runtime.html`

duché šipky označují, že jedna třída spouští druhou, dvojitá šipka pak značí přenos dat. Tečkovaná šipka označuje pokračování. Konečně šedě označená třída je singleton, používaný pro přenos dat mezi částmi aplikace.

Vyhledávání pracuje asynchronně - po stisknutí tlačítka vyhledávání se proces spustí na samostatném vlákně, třídy provedou veškeré potřebné úkony a po jejich dokončení vyvolají ve třídě `BlindLibrivoxSearchActivity` předem nadefinovanou metodu, která zpracuje výsledky vyhledávání. Celý systém tak pracuje na principu několika řetězově propojených posluchačů.

Každá z tříd `XMLParser`, `XMLWorker` a `LibrivoxSearchTools` má v sobě deklarované rozhraní posluchače, obsahující metody, které se mají provést poté, co daná třída dokončí svůj úkol. Akce, která se provede po dokončení úkolu je pak deklarovaná ve třídě o "stupeň výš"v hierarchii.

- `LibrivoxSearchTools`
  - Přijímá práci od tříd uživatelského rozhraní.
  - Obsahuje rozhraní posluchače *SearchToolsListener*.
  - V konstruktoru vytváří novou instanci třídy `XMLWorker`.
  - Vytváří nový *XMLWorkerListener* (posluchač pro dokončené akce třídy `XMLWorker`) který je deklarovaný rozhraním ve třídě `XMLWorker` a implementuje předdefinované metody rozhraní - akce, které se mají provést po dokončení úkolu třídy `XMLWorker`.
  - Zadává práci třídě `XMLWorker`.
- `XMLWorker`
  - Přijímá práci od `LibrivoxSearchTools`.
  - Obsahuje rozhraní posluchače *XMLWorkerListener*.
  - V konstruktoru vytváří novou instanci třídy `XMLParser`.
  - Vytváří nový *XMLParserListener* - deklarovaný rozhraním ve třídě `XMLParser` a implementuje jeho metody.
  - Zadává práci třídě `XMLParser`.
  - Po dokončení úkolu vyvolává metody svého posluchače.
- `XMLParser`
  - Přijímá práci od `XMLWorker`.
  - Obsahuje rozhraní posluchače *XMLParserListener*.
  - Po dokončení úkolu vyvolává metody svého posluchače.

Inicializace vyhledávání

Vstupem metody `search` je textový řetězec se zadaným textem k vyhledání a typ tohoto vyhledávání (podle jména autora nebo názvu knihy, hodnota z dříve definovaného *enumu*). Pomocí globální proměnné typu *boolean* se

zkontroluje, zda-li již neprobíhá vyhledávání (v případě že ano, metoda se ukončí). Po dokončení vyhledávání se hodnota této proměnné "překlopí".

Další globální proměnná typu *ArrayList*, která uchovává v podobě seznamu veškeré nalezené knihy, se při každém novém vyhledávání vyprázdní. Poté se pomocí metody `makeQuery` vytvoří textový řetězec obsahující URI dotazu k odeslání do API TIA (viz sekce 2.2.2). Je složen z následujících částí:

- Kořenová URL API
- Druh vyhledávání - **title** nebo **creator**
- Text k vyhledání
- Omezení vyhledávání pouze na položky LibriVox (**subject:(librivox)**)
- Seznam polí (informací o jednotlivých knihách), které má API navrátit:
  - Jméno autora
  - Popis knihy
  - Počet stažení
  - Identifikační řetězec
  - Datum publikace
  - Název díla
- Způsob seřazení výsledků - získaný třídy `LibrivoxSettings`, kde je staticky deklarovaný pro snadnou změnu
- Maximální počet navrácených knih (řádků) - také získaný z třídy `LibrivoxSettings`
- Formát návratových dat - XML

Zvolený způsob řazení výsledků je sestupně podle počtu stažení - je to způsob, kterým lze zajistit určité řazení dle relevance. Na prvních místech v seznamu nalezených knih tak budou knihy s největším počtem stažení, což teoreticky koreluje s oblíbeností nalezených knih. Maximální počet navrácených knih je nastaven na 250 děl, což je (zatím) dostatečná rezerva a návratový formát je nastaven na XML.

Po vytvoření textového řetězce s URI se tento odešle v argumentu metody `getXMLResponse` třídy `XMLWorker`. V podřízených třídách se úkoly zpracují, a při dokončení zavolají metody svých posluchačů. Výsledky jejich úkolů tak "probublají"až do třídy `LibrivoxSearchTools`. Výstupem z třídy `XMLWorker` je seznam typu *ArrayList*, obsahující objekty typu `LibrivoxAudioBook` - představené v části 3.6.1. Obsah seznamu (je-li nějaký) se vloží do globálního seznamu nalezených knih.

Pokud se po takto provedeném vyhledávání nepodařilo nalézt více než pět knih (toto číslo je definované jako statická proměnná v třídě `LibrivoxSettings`), provede se vyhledávání pomocí tzv. *Fuzzy Query*. URI pro API TIA zůstává stejné, pouze se před text k vyhledání vloží znak vlnovky.

Takovéto vyhledávání zkouší hledat podle slov podobných zadání, například se změněnými znaky na určitých pozicích. Dle dokumentace k API TIA ale tento způsob dotazu vyžaduje větší množství výpočtů na serverové straně a je tak pomalejší, proto je používán jen pokud klasické hledání nenavrátí dostatek výsledků.

Mezi první a druhou úrovní vyhledávání (pokud je potřeba) metoda `search` čeká na odpověď podřízených tříd - to je implementováno pomocí objektů typu *Lock*[11] a *Condition*[12], které se používají pro řízení přístupu ke sdílenému zdroji. V mém případě se postup metody uzamkne a čeká na signál, tedy na splnění podmínky - tou je úspěšné dodání knih z podřízených metod. Poté se metoda opět odemkne a vyhodnocuje se dál.

Při přidávání knih nalezených pomocí druhé úrovně vyhledávání je ještě ošetřen výskyt duplicit. Po ukončení (obou úrovní) vyhledávání se zavolá metoda `onSearchFinished` posluchače třídy `LibrivoxSearchTools`, v jejímž argumentu se zpět do uživatelského rozhraní odešle seznam nalezených knih (objektů *LibrivoxAudioBook*).

XMLWorker

Tato třída má za úkol odeslat síťový dotaz (do API TIA) a získat odpověď ve formátu XML. Pro vykonávání síťových úloh využívá knihovnu ***Volley***[13] vyvinutou společností Google specificky pro jednoduchou a rychlou síťovou komunikaci a přenos dat v aplikacích na vyvíjených pro Android. Tuto knihovnu jsem zvolil zejména díky její snadné a přehledné použitelnosti.

V konstruktoru této třídy je vytvořen nový objekt typu `RequestQueue` - fronta požadavků (*requests*), které knihovna automaticky zpracovává. V našem případě jsou do fronty zařazovány požadavky typu `StringRequest`, tedy HTTP dotazy, u nichž se očekává jako odpověď serveru řetězec znaků (soubor XML v podobě řetězce znaků).

Z důvodů špatně rozeznávaného kódování u odpovědí ve formátu XML, obsahujících metadata o souborech v adresářích knih databáze TIA, jsem musel vytvořit svůj vlastní objekt typu `Request`. Ten jsem deklaroval jako vnitřní třídu v třídě `XMLWorker` a využívám ho při zařazování požadavků do fronty. Má stejný tvar jako základní požadavek typu `StringRequest`, pouze má pevně předepsané dekódování odpovědí ve formátu *UTF-8*.

Dále jsou v konstruktoru implementovány dva objekty, které zpracovávají obdržené odpovědi ze serverů. Jeden objekt typu `Response.Listener<String>`, odpovědný za zpracování standardních odpovědí a druhý typu `Response.ErrorListener`, který zpracovává případy chybové odpovědi. Oba tyto objekty při vytvoření jen implementují příslušné posluchačské metody jejich rozhraní (`onResponse` a `onErrorResponse`), které se spustí při daných událostech.

[11] *Android Developers* - Lock - `https://developer.android.com/reference/java/util/concurrent/locks/Lock.html`

[12] *Android Developers* - Condition - `https://developer.android.com/reference/java/util/concurrent/locks/Condition.html`

[13] *Android Developers* - Volley (Transmitting Network Data Using Volley) - `https://developer.android.com/training/volley/index.html`

```
− <response>
  + <lst name="responseHeader"></lst>
  − <result name="response" numFound="4" start="0">
    − <doc>
        <str name="creator">William Shakespeare</str>
      − <str name="description">
          Librivox recording of Romeo and Juliet, by William Shakespeare. Read by Becky Crackel. Romeo and Juliet
          is perhaps the most famous of Shakespeare's plays and is thought to be the most famous love story in Western
          history. It concerns the fate of two very young lovers who would do anything to be together. The Montagues
          and the Capulets of Verona, Italy, are in the midst of a long-standing feud when Romeo Montague drops in on
          a masquerade party at the Capulets'. While there he meets and woos the daughter of the house, Juliet. She
          likewise returns his passion, and their secret meeting later that night on her bedroom balcony begins a series
          of tragic events that no one could have foretold. (Summary by Becky Crackel) For more free audiobooks, or
          to become a volunteer reader, please visit librivox.org. For further information, including links to online text,
          reader information, RSS feeds, CD cover or other formats (if available), please go to the LibriVox catalog
          page for this recording. M4B audio book (84mb)
        </str>
        <str name="downloads">951209</str>
        <str name="identifier">romeo_and_juliet_librivox</str>
        <str name="publicdate">2006-08-06T02:58:06Z</str>
        <str name="title">Romeo and Juliet</str>
      </doc>
```


Obrázek 3.11: Odpověď ve formátu XML z API TIA (pro dotaz na knihy jejichž název obsahuje slovo "romeo")

Požadavky se pak jednoduše přidávají do fronty pomocí metody `add`, jejímž argumentem je vždy nová instance mého vlastního požadavku s nadefinovanými posluchači pro případ správně odpovědi a chybové odpovědi.

Při přijetí správné odpovědi (XML souboru v podobě textového řetězce), která není prázdná, se v novém vlákně spustí na třídě `XMLParser` parsovací metoda pro získání dat z odpovědi. V opačné případě (chybová odpověď serveru nebo jiný problém) třída zavolá metodu svého posluchače ve které navrátí třídě `LibrivoxSearchTools` hodnotu *null*.

Celá třída je (stejně jako třída `XMLParser`) napsaná tak, aby byla použitelná nejen pro potřeby vyhledávání (kde získává odpověď serveru s nalezenými knihami ve formátu XML), ale i pro stahování (kde získává XML metadata o souborech v adresáři knihy na serveru TIA). Funkce třídy je určena již při vytvoření dané instance, protože argumentem jejího konstruktoru je jedna z hodnot (`DOWNLOAD` nebo `SEARCH`) v ní deklarovaného *enumu*. Získávání síťové odpovědi funguje totožně, jen se liší tím, jaká metoda (parsování knih do objektů `LibrivoxAudioBook` nebo `LibrivoxArchiveFile`) se zavolá na třídě `XMLParser` při obdržení správných dat.

XMLParser

Účelem třídy `XMLParser` je parsování[14] souborů XML ve formátu textových řetězců. Výstupem tohoto procesu je v závislosti na typu parsování buď seznam knih (objektů typu `LibrivoxAudioBook`), nebo souborů (objektů typu `LibrivoxArchiveFile`).

Parsování probíhá buď v metodě `getBooks` (pro získání audioknih), nebo v

[14] *Parsování* (také syntaktická analýza) - proces analýzy řetězce znaků (na základě daných pravidel) a tvorby stromu (vztahů) získaných dat. [16]

metodě `getFiles` (pro získání souborů). Je prováděno pomocí objektu *XMLPullParser*[15] - jednoduchého nástroje pro parsování XML souborů. Na vstupu tohoto nástroje (nastaveného metodou `setInput`) je textový řetězec obsahující soubor XML se získanými daty. Parsovací nástroj si vytvoří na základě analýzy textového řetězce pracovní datovou strukturu (definovanou značkami XML) a tu pomocí cyklů a podmínek následně procházím a získávám potřebná data, která doplňuji do vytvořené instance objektu `LibrivoxAudioBook` nebo `LibrivoxArchiveFile`.

Celý proces probíhá na základě předešlé znalosti struktury XML souboru s informacemi o audioknihách (viz obrázek 3.11), nebo audio souborech knihy.

- V prvním případě parsování probíhá od startovní značky dokumentu v cyklu `while` až do té chvíle, než parser narazí na značku konečnou.
- Mezitím se hledají značky (tagy) **doc**, které ohraničují jednotlivé nalezené knihy.
- Pokud parser takovou značku najde, vytvoří nový objekt typu `LibrivoxAudioBook` reprezentující nalezenou audioknihu.
- Pak prohledává hierarchicky podřazené značky (jenž mají stejný název **str** značící, že jejich obsah je textový řetězec, ale liší se hodnotami atributů **name**) až do konečného tagu **doc**.
- Pokud hodnota atributu **name** odpovídá hledané (např. pole *creator* obsahující název autora), tak obsah mezi tagy **str** extrahuji a přiřadím skrze *setter* k příslušné vlastnosti objektu audioknihy.
- Poté, co narazí parser na konečnou značku typu **doc**, přidá vytvořenou audioknihy do seznamu nalezených knih.

Poté, co je parsování dokončeno, vyvolá se metoda posluchače `onXMLParsed` v jejímž argumentu se předá seznam nalezených audioknih třídě `XMLWorker`. Ta pak získaný seznam jen předá výš, do třídy `LibrivoxSearchTools`, opět v argumentu metody svého posluchače `onListLoaded`.

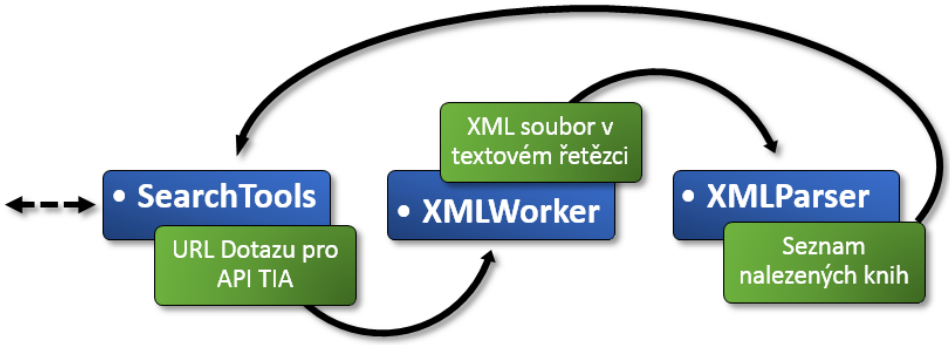
Přenos informací mezi třídami při vyhledávání ilustruje schéma 3.12.

Modré panely reprezentují třídy, zelené panely přenášené informace. Šipky pak směr toku dat, přerušovaná šipka v levé části naznačuje spojení v dalšími třídami (uživatelským rozhraním).

Dokončení vyhledávání

Poté, co je seznam knih vytvořený ve třídě `XMLParser` odeslán výše, doputuje až do třídy `LibrivoxSearchTools`, která zavolá metodu `onSearchFinished`. Ta je konečně implementovaná ve třídě `BlindLibrivoxSearchActivity`, která celé vyhledávání spustila. V argumentu metody je předán zmíněný seznam audioknih (objektů `LibrivoxAudioBook`). Pokud je seznam prázdný,

[15] *Android Developers* - XmlPullParser - `https://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html`



Obrázek 3.12: Přenos informací mezi třídami při vyhledávání

uživatel je informován o tom, že nebyly nalezeny žádné knihy. V opačném případě se oznámí počet nalezených knih, seznam je předán do singletonu `LibrivoxHolder` a spustí se třída `BlindLibrivoxSearchListActivity`, která má na starosti zobrazení nalezených knih v podobě seznamu.

3.6.4 Zobrazení výsledků vyhledávání

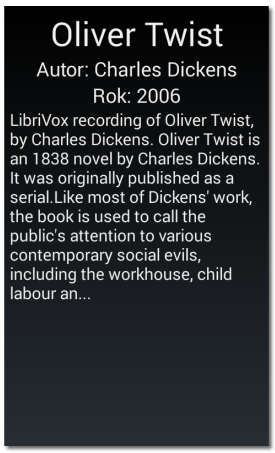
Výsledky vyhledávání, tedy seznam nalezených knih, zobrazuje třída `BlindLibrivoxSearchListFragment`. Ta se inicializuje oklikou, nejprve přes třídu `...SearchListActivity`, kde je vyvolána metoda `setContentView`, v jejímž argumentu se nachází XML soubor `..._search_list_fragment_activity.xml`. Ten obsahuje komponentu *fragment*, která odkazuje elementem *class* na třídu `...SearchListFragment`. Třída `...SearchListFragment` dědí od třídy `BlindListFragment` a zajišťuje zobrazení seznamu knih a zpracování gest. Při její inicializaci se seznam nalezených knih získá se singletonu `LibrivoxHolder`, kde je uložen.

Obrázek 3.13: Proces spuštění třídy *BlindLibrivoxSearchListFragment*

Seznam knih je zobrazen jako řada panelů, na každém z nich jsou informace o právě jedné knize. Zobrazené informace (uložené v objektech `LibrivoxAudioBook`) jsou:

- Název knihy
- Autor díla
- Rok publikace v databázi LibriVox
- Popis knihy (omezený pro úspornost zobrazení na 300 znaků - hodnota definovaná ve třídě `LibrivoxSettings`)

Informace jsou předávány do uživatelského rozhraní přes třídu `LibrivoxSearchListViewBinder`, která dědí od třídy `ViewBinder`, je propojená s vnitřním kurzorem třídy `...SearchListFragment` a odesílá textové řetězce s daty do jednotlivých textových polí, definovaných v XML souboru, který definuje rozvržení zobrazení jednotlivých panelů seznamu - `blind_librivox_search_list_fragment.xml`.



Obrázek 3.14: Ukázka jednoho z panelů seznamu nalezených knih

Při procházení seznamu je u každé knihy automaticky přečten název díla, jméno autora a rok publikace - pro přečtení popisu knihy je nutné se dotknout obrazovky krátce dvěma prsty. Poté se zopakují informace na obrazovce a k tomu se přečte celý popis knihy.

Dlouhým dotykem jednoho prstu se aplikace pokusí přesunout do menu pro stahování aktuální knihy na obrazovce. Předtím se ale ještě provede kontrola, zda-li je v zařízení přítomna paměťová SD karta, která je pro stahování knih vyžadována. Kontrola probíhá vyčtením seznamu externích úložišť, ve kterých se pak hledá úložiště s cestou (*path*), která odpovídá empiricky známé cestě k adresáři paměťové karty v zařízení (*/storage/extSdCard*). Tato kontrola je nutná zejména protože cílové zařízení (Samsung *Galaxy Core 2)* vede svou interní paměť telefonu jako externí úložiště (*/storage/emulated/0*). Pokud v zařízení není vložena paměťová karta, přečte se příslušné oznámení. V opačném případě se spustí nabídka pro stahování vybrané knihy.

3.7 Stahování

3.7.1 Uživatelské rozhraní

Do nabídky stahování se vstupuje dlouhým dotykem na vybrané knize ze seznamu nalezených knih. Nabídka stahování má podobu seznamu, který má na prvním místě panel s nabídkou stažení celé knihy (zabalené v archivu), a na dalších pozicích jednotlivé kapitoly vybrané knihy. Ty jsou seřazeny podle formátu a čísla kapitoly.

Seznam nabídky stahování funguje na stejném principu jako seznam nalezených knih v sekci vyhledávání. Širší popis jsem provedl již v sekci 3.6.4. Proces ilustruje schéma 3.15.

Obrázek 3.15: Proces spuštění třídy *BlindLibrivoxDownloadListFragment*

Třída `BlindLibrivoxDownloadListActivity` je spuštěna dlouhým dotykem při výběru knihy ze seznamu nalezených knih. Žádnou další funkcionalitu třída nemá. Jednotlivé panely nabídky stahování pak mají strukturu popsanou v XML souboru rozvržení `..._librivox_download_list_fragment_item.xml`. Ta obsahuje pět viditelných textových polí (*TextView*), do kterých se dodávají informace (textové řetězce) přes třídu `LibrivoxDownloadListViewBinder`.

Na panelech s kapitolami jsou zobrazeny následující informace:

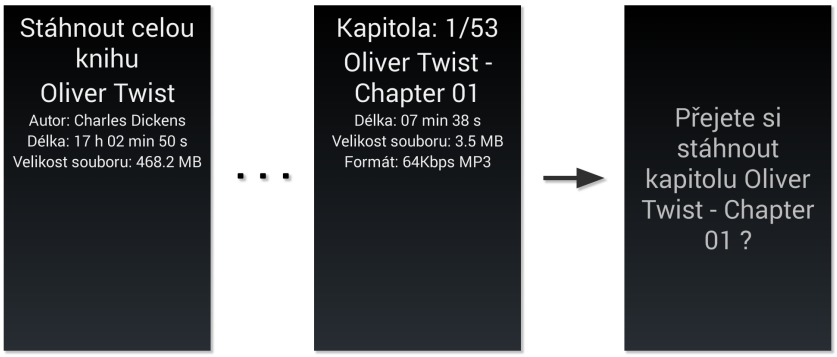
- Číslo kapitoly
- Název kapitoly
- Délka kapitoly (délka přehrávání audio souboru)
- Velikost souboru s kapitolou v čitelné formě
- Formát souboru s kapitolou (nízká, vyšší či jiná kvalita)

U prvního panelu, který nabízí stažení celé knihy v archivu jsou uvedeny tyto informace:

- Nadpis panelu ("Stáhnout celou knihu")
- Název knihy
- Jméno autora díla
- Délka celé knihy (délka přehrávání všech audio souborů)
- Velikost archivu s celou knihou (v čitelné formě)

Seznam cykluje - pokud na posledním prvku klepnete jedním prstem v pravé části obrazovky, ocitnete se zpět na začátku a naopak.

Při procházení seznamu jsou u kapitol automaticky přečtena jejich čísla a názvy, u archivu s celou knihou se přečte nadpis panelu ("Stáhnout celou knihu"), název knihy a jméno autora. Při krátkém dotyku dvěma prsty se podobně jako u seznamu nalezených knih zopakují informace na obrazovce a poté se přečte i délka kapitoly, velikost souboru a u kapitol i jejich formát. Všechny předčítané informace jsou formátovány tak, aby byly pro uživatele



Obrázek 3.16: Panely nabídky stahování

co nejsrozumitelnější - například u textového řetězce s délkou souboru se nahradí zkratky 'h', 'min' a 's' celými slovy - 'hodin', 'minut' a 'sekund'. Stejný princip je uplatněn i u řetězce s velikostí souboru.

Při výběru stažení celé knihy nebo kapitoly se objeví dialog (objekt typu `BlindAppDialog`), požadující potvrzení úkonu (ukázka dialogu - panel vpravo na obrázku 3.16). U požadavku na stažení celé knihy se krom dotazu na potvrzení také přečte velikost archivu (jako dodatečné upozornění při stahování velkých souborů). Při požadavku na stažení kapitoly se přečte jen její název. Potvrzení dlouhým dotykem jednoho prstu spustí příslušnou metodu na třídě `LibrivoxDownloadTools`, která zajišťuje stahování a uživatel je informován o spuštění stahování. Poté se dialog uzavře.

Načítání informací

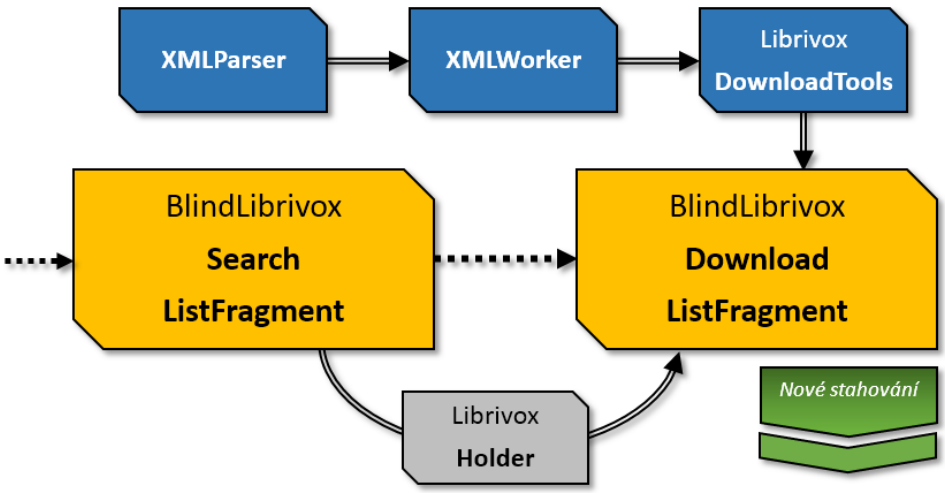
Při inicializaci uživatelského rozhraní nabídky stahování se automaticky spustí proces získávání informací o kapitolách ze souboru s metadaty, který je uložen v depozitáři knihy na serveru TIA. Stejně tak se načítají informace o velikosti a délce celé knihy v archivu. Než jsou informace úspěšně načteny a mohou být zobrazeny v uživatelském rozhraní, funguje seznam nabídky stahování v omezeném režimu:

- U panelu s nabídkou stažení celé knihy je pouze nadpis, název díla a jméno autora (ostatní informace se načítají).
- Při výběru stažení celé knihy (dlouhým dotykem jednoho prstu) se neobjeví potvrzovací dialog - uživatel je místo toho upozorněn, že se informace ještě načítají (nelze stáhnout archiv bez dat o jeho velikosti z důvodů kontroly volného místa).
- Při pokusu o pohyb vpřed nebo vzad v seznamu (k prohlížení jednotlivých kapitol) je uživatel informován, že se kapitoly ještě načítají.
- Po úspěšném načtení kapitol se ozve zvukový signál, a uživatel již může procházet seznam.

3.7.2 Logika stahování

Třída `LibrivoxDownloadTools` obsahuje veškeré potřebné metody pro kontrolu, přípravu adresářů, inicializaci stahování apod. Instanci třídy `...DownloadTools` si vytvoří `...DownloadListFragment` již při své inicializaci. Argumenty konstruktoru `...DownloadTools` jsou dva - objekt typu *Activity* (získaný pomocí `getActivity` a používaný pro spouštění služeb, přístup ke sdíleným preferencím apod.) a kniha (objekt `LibrivoxAudioBook`), vybraná ze seznamu nalezených knih, v jejíž nabídce stahování se nacházíme.

Hierarchii tříd stahování ilustruje schéma 3.17.



Obrázek 3.17: Hierarchie tříd stahování

Modře jsou označeny třídy logiky stahování, žlutě třídy uživatelského rozhraní. Šedý panel označuje singleton `LibrivoxHolder`. Dvojitá šipka ilustruje přenos dat mezi třídami. Zelený panel ve tvaru šipky pak ilustruje akci, tedy spuštění nového stahování.

Příprava kapitol

Jelikož vyhledávací API TIA neposkytuje žádné informace o kapitolách knihy (URL pro jejich stáhnutí, názvy kapitol atd.), je nutné tyto informace získat jinak. Každá kniha (položka) v databázi TIA má svůj vlastní unikátní identifikátor v podobě textového řetězce (ten je získán z vyhledávacího API pro každou knihu a uložen v objektu `LibrivoxAudioBook`). Pro každou knihu pak na webu TIA existuje stránka s informacemi o knize s URL v následujícím formátu [17]:

```
http://archive.org/details/[identifikátor]
```

Analogicky také existuje pro každou knihu webový depozitář, který obsahuje veškeré soubory příslušící ke knize. Jeho adresa má vždy tvar:

```
http://archive.org/download/[identifikátor]
```

Jedním ze souborů, který se vždy nachází v depozitáři knihy, je XML soubor obsahující informace o všech souborech, které se nacházejí v daném adresáři. Tento soubor má vždy stejnou adresu ve tvaru:

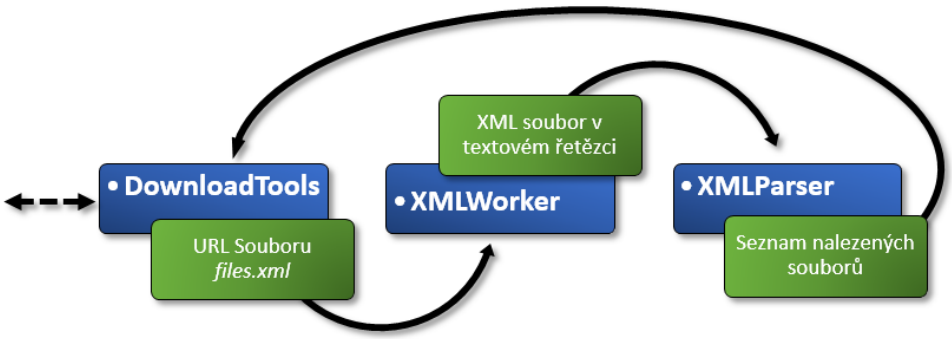
```
http://archive.org/download/[identifikátor]/[identifikátor]
_files.xml
```

Náhled toho, jak vypadá vnitřní struktura souboru *files*.xml ukazuje obrázek 3.18.


```
− <files>
  + <file name="romeo_juliet_1001_librivox_128kb.m3u" source="derivative"></file>
  + <file name="romeo_juliet_1001_librivox_files_all.torrent" source="derivative"></file>
  + <file name="romeo_juliet_1001_librivox.json" source="original"></file>
  − <file name="romeo_4_shakespeare.mp3" source="original">
      <format>128Kbps MP3</format>
      <creator>William Shakespeare</creator>
      <album>Romeo and Juliet</album>
      <title>4 - Act 4</title>
      <md5>3962c42031b7df15b0dd37acd3f74816</md5>
      <track>5/6</track>
      <mtime>1263786865</mtime>
      <size>23636096</size>
      <crc32>f7b52fb6</crc32>
      <sha1>b0e90e8e879964d0edad0fee164c11ecc8644db4</sha1>
      <length>1477.12</length>
      <height>0</height>
      <width>0</width>
      <artist>William Shakespeare</artist>
      <genre>Speech</genre>
    </file>
```


Obrázek 3.18: Ukázka obsahu souboru *files*.xml
Zdroj: `https://archive.org/download/romeo_juliet_1001_librivox/romeo_juliet_1001_librivox_files.xml`

Jak bylo popsáno v sekci 3.6.3 a částech věnujících se třídám pro získávání XML souborů a jejich parsování, funkcionalita těchto tříd je totožná i pro potřeby stahování (pouze se při vytváření instance `XMLWorker` předá v konstruktoru jiný argument s typem použití třídy). Celý proces pak ilustruje diagram 3.19.



Obrázek 3.19: Proces získaní informací o kapitolách

1. Třída uživatelského rozhraní (`...DownloadListFragment`) při své inicializaci vytvoří instanci nástrojů `...DownloadTools` (v konstruktoru je předán objekt s knihou).

2. Třídě `...DownloadTools` je přiřazen posluchač, ve kterém je definováno, jaké akce se mají provést po načtení informací o kapitolách a archivu.

3. Zavolá se metoda `getChapterFilesList` ve třídě `...DownloadTools`. Ta pomocí identifikátoru z poskytnutého objektu knihy vytvoří URL příslušného souboru *files*.xml a odešle ho do třídy `XMLWorker`.

4. Instance `XMLWorker`, založená v konstruktoru `...DownloadTools`, získá pro danou URL síťovou odezvu v podobě textového řetězce obsahujícího soubor *files*.xml. Tento řetězec pak odešle jako argument metody `getFiles` do třídy `XMLParser`.

5. Instance `XMLParser`, založená v konstruktoru `XMLWorker`, vyparsuje ze souboru XML všechny informace o kapitolách knihy. Ty uloží do instancí objektu `LibrivoxArchiveFile`. Jejich seznam poté přes posluchače navrátí až do třídy uživatelského rozhraní.

Model souboru kapitoly. Třída `LibrivoxArchiveFile` reprezentuje kapitolu knihy zobrazenou v nabídce stahování. Každý objekt kapitoly má následující vlastnosti:

- Číslo kapitoly
- Název kapitoly
- Délku kapitoly - čas přehrávání audio souboru (ze souboru *files*.xml je získán v sekundách, při zpracování *setterem* se zformátuje do standardní podoby)
- URL adresa pro stažení kapitoly
- Velikost souboru s kapitolou v lidmi čitelné podobě
- Velikost souboru v bajtech (takto je získána ze souboru *files*.xml, do lidmi čitelné podoby se opět zformátuje v příslušném `setteru`)
- Formát kapitoly (64kbps MP3, 128kbps MP3 nebo jiný) určující kvalitu nahrávky

Parsování kapitol probíhá analogicky k parsování knih popsanému v sekci 3.6.3. Důležité je, že parser vybírá ze souboru *files*.xml pouze položky se soubory, jejichž název končí "*.mp3*". Takto vyseparuje z metadat pouze informace o audio souborech. Pro každý nalezený audio soubor formátu MP3 pak založí novou instanci objektu `LibrivoxArchiveFile`, kterou po naplnění informacemi vloží do seznamu nalezených kapitol. Navíc se při parsování ještě lokalizují metadata souboru s celou knihou zabalenou v archivu. Z těch se

získá délka knihy (přehrávání) a velikost archivu. Pro přesun informací do uživatelského rozhraní se poté zavolají příslušné metody posluchačů.

Po načtení kapitol do třídy `LibrivoxDownloadTools` se jejich seznam zkontroluje a seřadí podle čísla kapitoly a jejího formátu. Poté se odešle do třídy uživatelského rozhraní, společně s jedním objektem `ArchiveFile` který obsahuje informace o archivu s celou knihou a uživatelské rozhraní se následně aktualizuje. Nastavením ve třídě `LibrivoxSettings` lze určit, zda-li se v seznamu kapitol budou zobrazovat i kapitoly v nízké kvalitě. Seznam kapitol se uloží do sdílených preferencí aplikace pod svým unikátním identifikátorem.

Při načtení objektu nesoucího informace o archivu se tato data předají k zobrazení do uživatelského rozhraní. Pokud by objekt určitá data neobsahoval (nebyly zapsány v souboru *files*.xml), získají se alternativní cestou. V případě chybějícího údaje o velikosti archivu se k jeho získání použije metoda `getFileSize`, která se připojí přímo na adresu archivu a zjistí jeho velikost. Pokud chybí informace o délce (přehrávání) celé knihy, vypočítá se součtem délek jednotlivých kapitol.

Systém stahování

Proces systému stahování je ilustrován na diagramu 3.20.

Obrázek 3.20: Proces stahování

- Kořenový adresář LibriVox se nachází v systémem přidělené složce aplikace na paměťové kartě.
- Název adresáře knihy tvoří vždy název knihy, jméno autora a unikátní identifikátor.

- Pokud není dostatek místa v úložišti, nebo je soubor již stažen, proces se přeruší a situace je oznámena uživateli.
- Ke stahování souborů je použita systémová služba *DownloadManager*[16], která celý systém stahování řídí na pozadí.
- Při stažení celé knihy se do jejího adresáře uloží textový soubor s informacemi o knize pro pozdější referenci.

Požadavek (*request*) na stažení, který se zařazuje do fronty, má titulek s názvem stahované knihy a popis, ve kterém je zaznamenán typ stahování (celá kniha nebo kapitola).

Služby

Aplikace používá dvě služby:

- `LibrivoxDownloadNotificationService` - služba hlasového upozornění při dokončení stahování.
- `LibrivoxUnzipService` - služba rozbalování stažených archivů s celou knihou.

Obě služby běží v pozadí systému i po ukončení aplikace LibriVox a hlídají dokončená stahování.

LibrivoxDownloadNotificationService. Tato služba slouží jako náhrada za notifikace ve stavové liště systému Android. Dědí od třídy *Service*[17] a používá objekt *BroadcastReceiver*[18] k zaznamenávání událostí dokončení stahování.

- *BroadcastReceiver* zaznamená událost dokončeného stahování a aktivuje metodu `onReceive`.
- Z objektu *Intent* zaznamenané události se vyčte identifikační číslo dokončeného stahování.
- Pomocí identifikačního čísla se vyčte příslušné stahování ze služby *DownloadManager*.
- Proběhne kontrola zdali se jedná o stahování kapitoly z aplikace LibriVox (porovnáním popisku stahování).
- Zjistí se status dokončeného stahování:
  - Pokud bylo stahování úspěšné (`STATUS_SUCCESSFUL`), ozve se tón upozornění a hlasové oznámení s názvem knihy.

[16] *Android Developers* - DownloadManager - `https://developer.android.com/reference/android/app/DownloadManager.html`
[17] *Android Developers* - Service - `https://developer.android.com/reference/android/app/Service.html`
[18] *Android Developers* - BroadcastReceiver - `https://developer.android.com/reference/android/content/BroadcastReceiver.html`

- Pokud stahování neuspělo (`STATUS_FAILED`), ozve se tón ohlašující chybu a hlasové oznámení s důvodem neúspěchu stahování.

Tato služba funguje jen pro stažené kapitoly z aplikace LibriVox. Upozornění na stažení celé knihy včetně jejího rozbalení pak řeší druhá služba.

LibrivoxUnzipService. Zaznamenávání událostí dokončeného stahování funguje ve službě `LibrivoxUnzipService` stejně jako v první zmíněné službě. Při spuštění služby `LibrivoxUnzipService` se ale navíc ze sdílených preferencí vyčtou následující informace:

- Uložená identifikační čísla přidělená požadavkům na stažení celé knihy službou *DownloadManager*.
- Uložené cesty ke stahovaným archivům na paměťové kartě (páry cesta-klíč, kde klíč je ID stahování).

První tři body procesu jsou pak identické jako u předešlé služby. Dále se provede následující:

- Proběhne kontrola zdali se jedná o stahování celé knihy z aplikace LibriVox porovnáním popisku stahování a identifikačního čísla dokončeného stahování s uloženými.
- Zjistí se status dokončeného stahování:
  - Pokud bylo stahování úspěšné (`STATUS_SUCCESSFUL`), spustí se v novém vlákně proces rozbalování archivu. Cesta k archivu se získá z uložených cest zadáním identifikačního čísla dokončeného stahování. Poté se ozve tón upozornění a hlasové oznámení o dokončení stahování a rozbalování knihy.
  - Pokud stahování neuspělo (`STATUS_FAILED`), ozve se tón ohlašující chybu a hlasové oznámení s důvodem neúspěchu stahování.

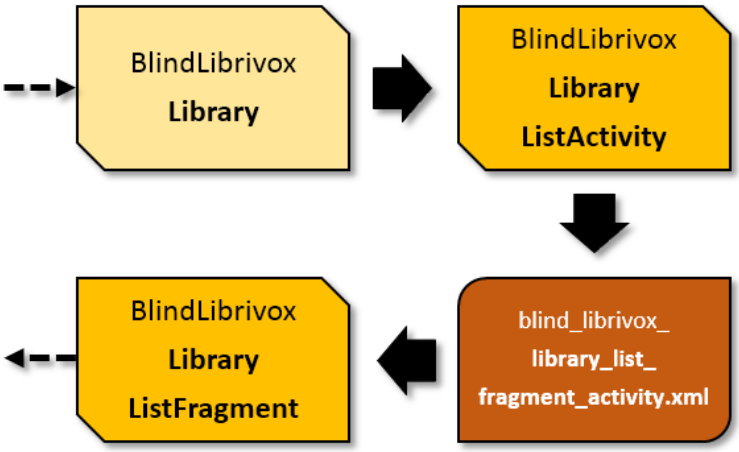
Samotné rozbalování archivu (odzipování) probíhá pomocí objektu *ZipInputStream*[19]. Jeho vstupem je právě soubor s archivem. Prochází se po jednotlivých položkách archivu (objekty *ZipEntry*) a soubory v archivu se extrahují na objektu `FileOutputStream` metodou `write`. Po dokončení rozbalování (dojetí na konec objektu *ZipInputStream*) je archiv smazán z paměťového úložiště.

3.8 Knihovna

3.8.1 Seznam uložených knih

Popis knihovny bude velice jednoduchý, neboť používá většinu nástrojů a zobrazení z předešlých částí programu. Hierarchii tříd základní nabídky knihovny ilustruje diagram 3.21.

[19] *Android Developers* - ZipInputStream - `https://developer.android.com/reference/java/util/zip/ZipInputStream.html`



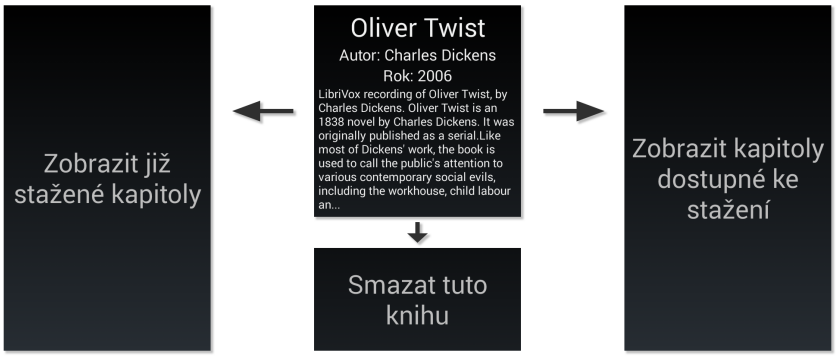
Obrázek 3.21: Hierarchie tříd hlavní nabídky knihovny

Do knihovny se vstupuje skrz hlavní menu aplikace. Pokud v zařízení není vložena paměťová karta, do knihovny není možné vstoupit (není odkud načítat uložené knihy). Hierarchie tříd má stejnou strukturu jako u zobrazení seznamu knih nalezených při vyhledávání v databázi. Způsob ovládání, množství informací a jejich zobrazení je také identické. Na vyčtení knih k zobrazení knihovna používá nástrojovou třídu `LibrivoxLibraryTools`.

- V konstruktoru třídy se načtou sdílené preference obsahující:
  - Objekty `LibrivoxAudioBook` reprezentující uložené knihy.
  - Pole ke kterém jsou uschovány seznamy (objektů) kapitol uložených knih.
  - Cestu k paměťovému úložišti.
- Vyčtou se všechny adresáře, které obsahuje kořenový adresář aplikace.
- Ze jmen nalezených neprázdných adresářů se extrahují identifikátory knih.
- Za použití identifikátorů se vyberou z načtených seznamů ze sdílených preferencí příslušné objekty knih a seznamy objektů kapitol. Ty se pak vloží do připravených seznamů určených k výstupu.
- Výstupní seznamy včetně seznamu cest k adresářům knih se předají metodou posluchače do uživatelského rozhraní.

V uživatelském rozhraní se tyto seznamy uloží do proměnných a při výběru knihy ze seznamu se příslušné informace v nich uložené předají dále.

Při výběru knihy ze seznamu dá aplikace k dispozici dialog s nabídkou vyobrazenou na schématickém obrázku 3.22. Smazání knihy probíhá v samostatném vlákně pomocí rekurzivní metody `deleteFolderWithContents` která smaže celý adresář s knihou včetně jeho obsahu. Předtím je nutné potvrdit



Obrázek 3.22: Panely nabídky knihovny

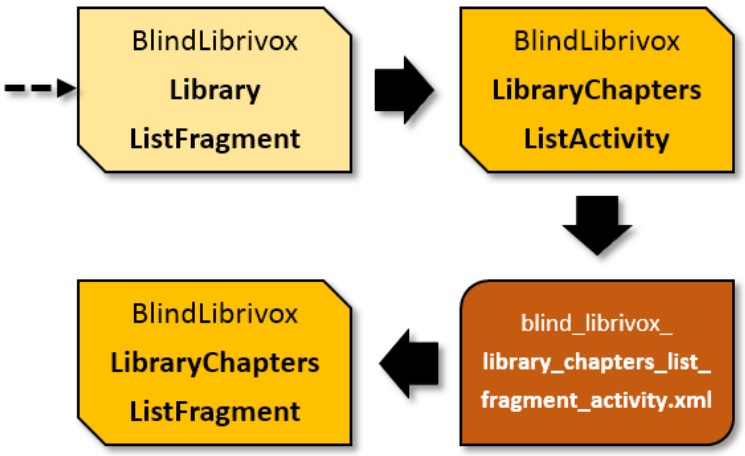
kontrolní dialog. Zobrazení kapitol (stažených anebo dostupných ke stažení) probíhá spuštěním aktivity `BlindLibrivoxLibraryChaptersListActivity`. Předtím se do singletonu `LibrivoxHolder` předá:

- Objekt `LibriVoxAudioBook` s aktuální knihou.
- Textový řetězec s cestou k adresáři knihy.
- Seznam všech (objektů) kapitol vybrané knihy.
- Druh zobrazení (stažené-nestažené kapitoly) podle zvolené nabídky.

Po spuštění třídy `BlindLibrivoxLibraryChaptersListFragment` se všechny informace opět ze singletonu načtou zpět.

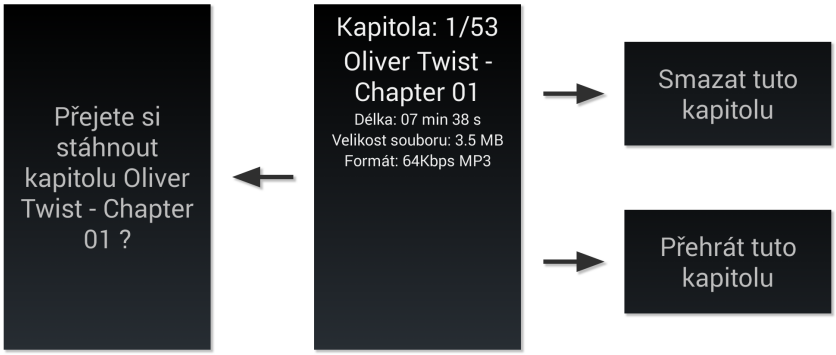
3.8.2 Výčet kapitol

Hierarchii tříd ke spuštění zobrazení kapitol vybrané knihy z knihovny ilustruje diagram 3.23.



Obrázek 3.23: Hierarchie tříd k zobrazení kapitol vybrané knihy v knihovně

Princip zobrazení je totožný k zobrazení seznamu kapitol v nabídce stahování. Při inicializaci třídy se ze singletonu `LibrivoxHolder` vyčtou informace předané předešlou nabídkou (se seznamem knih). Z adresáře knihy se nahrají názvy přítomných audio souborů. Poté se seznam souborů přítomných v úložišti porovná se seznamem všech kapitol - při zvoleném režimu zobrazení stažených kapitol se do uživatelského rozhraní načtou kapitoly, které jsou přítomny, při režimu zobrazení kapitol ke stažení se vyčte seznam nepřítomných kapitol.



Obrázek 3.24: Nabídka pro kapitoly vybrané knihy z knihovny

Na obrázku 3.24 jsou vidět uživatelské možnosti při výběru kapitoly ze seznamu. V případě, že je to seznam stažených kapitol, má uživatel možnost kapitolu smazat nebo přehrát.

- Mazání kapitoly probíhá pomocí stejné funkce jako mazání celé knihy. Je nutné kontrolní potvrzení a po vymazání se uživ. rozhraní aktualizuje.
- Přehrávání souborů obstarává třída `BlindLibrivoxPlayerActivity` (viz sekce 3.9).

V seznamu kapitol dostupných ke stažení je možné kapitolu stáhnout (proces probíhá identicky ke stahování kapitoly popsanému v kapitole 3.7). Před započtením stahování se provede kontrola připojení k internetu.

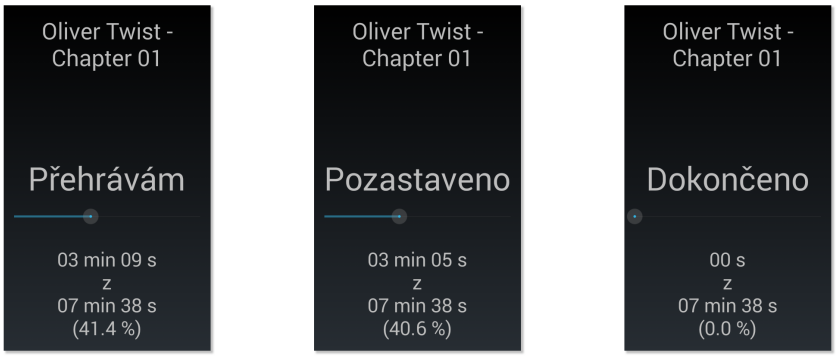
3.9 Přehrávač

Součástí mojí aplikace je i jednoduchý přehrávač audio souborů implementovaný ve třídě `BlindLibrivoxPlayerActivity`. Ten využívá statické metody třídy `BlindMediaPlayer`, která sama používá k přehrávání metody objektu *MediaPlayer*[20] a doplňuje je o dodatečnou funkcionalitu.

Na obrázku 3.25 je ukázka vzhledu uživatelského rozhraní přehrávače. To tvoří odshora dolů:

- Textové pole s názvem přehrávané kapitoly.

[20] *Android Developers* - MediaPlayer - `https://developer.android.com/reference/android/media/MediaPlayer.html`



Obrázek 3.25: Ukázka vzhledu přehrávače LibriVox

- Textové pole s aktuálním stavem přehrávání (přehrávám, pozastaveno, dokončeno).
- Posuvník (ang. *Seek bar*) - komponenta uživatelského rozhraní, která zobrazuje postup v přehrávání.
- Textové pole s informacemi o aktuální pozici přehrávání (klasicky a procentuálně).

Po načtení přehrávače se přečte název přehrávané kapitoly a přehrávání se automaticky spustí. Přehrávání lze pozastavit dlouhým dotykem jednoho prstu, stejným gestem je možné přehrávání opět spustit. Krátkým dotykem jedním prstem vpravo či vlevo se přehrávání přetočí o 5 % délky vpřed, nebo vzad. Pokud se uživatel dotkne obrazovky krátce dvěma prsty, přehrávání se (případně) pozastaví a přečte se jeho aktuální stav a pozice. Po dokončení přehrávání přehrávač čeká na další akci - dlouhým dotykem jednoho prstu lze přehrávání spustit znovu. Při zamknutí obrazovky přehrávání pokračuje dál, ale odchodem z přehrávače se automaticky ukončí.

Kapitola 4
Testování

Testování aplikace probíhalo 19. 5. 2016 v laboratoři pro uživatelský výzkum *ulab*[1]. Ta se nachází na Katedře počítačové grafiky a interakce v budově ČVUT na Karlově náměstí. Laboratoř tvoří dvě oddělené místnosti - v jedné z nich se nachází participant a moderátor testování, v druhé je pozorovatel, který zaznamenává průběh testování. Tento průběh sleduje na obrazovkách propojených s kamerami v druhé místnosti. Zvuk je přenášen k pozorovateli pomocí několika mikrofonů.

Při testování aplikace jsem byl přítomen v místnosti s participantem a moderoval jsem testování pokládáním otázek z připraveného dotazníku a zadáváním uživatelských úkolů. V místnosti pozorovatele pak probíhal záznam probíhajícího testování v podobě zápisu participantových odpovědí na připravené otázky a zaznamenávání průběhu řešení zadaných úkolů. Navíc byl po celou dobu testování nahráván audiozáznam zvuku z místnosti s participantem.

K testování aplikace byli pozváni dva nevidomí participanti. Prvním úkolem bylo vyplnění krátkého dotazníku, sestávajícího se z pěti otázek, před samotným testování. Poté participant dostal připravený mobilní telefon a seznámil se s ním. Následovalo postupné zadání devíti úkolů - po každém zadání participant úkol sám řešil až do úspěšného dokončení. V případě, že se během řešení úkolu dostal do situace, ve které si nevěděl rady, moderátor mu pomohl k dokončení úkolu. Po splnění všech úkolů byl s participantem vyplněn konečný dotazník, který se sestával ze tří otázek.

4.1 Průběh testování

Obě účastnice měly předchozí zkušenosti se nadřazeným systémem a znaly principy jeho ovládání. Všechny zadané úkoly dokázaly zdárně dokončit.

První účastnice testování (žena, 46 let) měla již zkušenosti s namluvenými audioknihami a možnost jejich poslechu v mobilním zařízení by využila. Při plnění úkolů měla problém pouze s orientací v nabídce stahování, kdy zprvu nepochopila strukturu seznamu položek. Při vyhledávání úspěšně dokázala na druhý pokus najít požadovanou knihu. Mírné potíže působilo, že se po

[1] *Usability Lab* at CTU in Prague - `http://ulab.cz/`

dokončení každého úkolu vrátila do hlavní nabídky telefonu. Kvůli návaznosti úkolů se tak musela stále vracet.

Orientaci a ovládání aplikace zhodnotila jako přiměřeně obtížné. Vyjádřila přesvědčení, že aplikace bude užitečnou součástí mobilního telefonu zejména pro anglicky mluvící nevidomé. Kladně hodnotila skutečnost, že je aplikace zdarma a nevyžaduje registraci.

Druhá účastnice testování (žena, 23 let) měla zkušenost pouze s audioknihami ve formě textu předčítaného hlasovým syntezátorem. Při testování měla největší problémy s ovládáním klávesnice při zadávání dotazů k vyhledání - potíže jí činila pomalá odezva klávesnice. S řešením úkolů neměla problémy, správně využívala k jejich řešení nápovědu. Sama zhodnotila ovládání aplikace a orientaci v ní jako snadnou. Nebyla si jistá, zda-li bude aplikace užitečnou součástí mobilního telefonu pro nevidomé, protože sama audioknihy poslouchá pouze na počítači.

4.2 Shrnutí

Testování aplikace na cílových uživatelích ze dvou věkových kategorií ukázalo její relativně bezproblémovou použitelnost. Při znalosti ovládání prostředí by uživatelé neměli mít větší problémy s jejím používáním. Největší prostor k vylepšení se pak ukázal v nabídce stahování, kde by mohla být srozumitelnost struktury menu vylepšena tak, aby se uživatelé mohli zorientovat i bez použití nápovědy. Kvůli načítání kapitol má nabídka zprvu pouze jedinou položku a to je pro uživatele matoucí. Přidáním hlasového upozornění o načítání nebo načtení kapitol by se mohla nabídka více zpřístupnit.

Kapitola 5

Závěr

Aplikaci LibriVox považuji za zajímavé rozšíření mobilního prostředí pro nevidomé. Tvoří dostupnou alternativu k existující aplikaci BookShare, a to především díky tomu, že je zdarma a nevyžaduje registraci. Implementaci práce předcházela důkladná analýza a příprava řešení, která zajistila, že výsledná aplikace stojí na stabilních základech a je uživatelsky přístupná.

Cíl vytvořit jednoduchou, ale účinnou aplikaci pro vyhledávání, stahování a přehrávání audioknih, byl dle mého mínění splněn. Aplikace je připravena se přizpůsobit v řadě aspektů, na základě případného budoucího ohlasu uživatelů. Existují i možnosti rozšíření aplikace, například vytvoření nabídky uživatelského nastavení nebo implementace dalších možností vyhledávání. Testování aplikace také ukázalo, že by se u vybraných nabídek dala zvýšit jejich přehlednost.

Při vývoji aplikace se objevily nedostatky, jejichž náprava byla buď nad rámec mé práce (detekce jazyka z textu pro jejich korektní čtení hlasovým syntezátorem) nebo si žádala systémové řešení v rámci nadřazeného prostředí (pokročilý notifikační systém a jednotný přehrávač). Řešení těchto problémů bude s velkou pravděpodobností realizováno v blízké budoucnosti.

S ohledem na jazykové poměry knih v databázi LibriVox je jasné, že cílovou skupinu aplikace tvoří především anglicky mluvící uživatelé. Je však možné, že se časem databáze LibriVox stane populárnější a jazyková diverzita namluvených knih se obohatí. V každém případě mě těší vědomí, že jsem mohl pomoci pootevřít bránu světa literatury nevidomým uživatelům.

Bibliografie

[1] Ericsson. *Ericsson Mobility Report*. Ed. Patrik Cerwall, Stephen Carson a Monika Byléhn. URL: http://www.ericsson.com/res/docs/2016/mobility-report/ericsson-mobility-report-feb-2016-interim.pdf (cit. 11. 05. 2016).

[2] Lee Rainie a Barry Wellman. *Networked. The new social operating system*. The MIT Press, 2012. ISBN: 978-0-262-01719-0.

[3] Vincent Gaudissart et al. "SYPOLE - Mobile Reading Assistant for Blind People". In: *SPECOM- 9th Conference Speech and Computer* (2004).

[4] Bc. Petr Svobodník. "Zpřístupnění mobilních telefonů se systémem Android pro nevidomé uživatele". Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická - Katedra počítačové grafiky a interakce, 2013.

[5] ČVUT v Praze (autor práce: Lukáš Rubeš. *ČVUT DSpace - Digitální knihovna ČVUT. Implementace audio-knihovny pro zrakově postižené uživatele*. URL: http://hdl.handle.net/10467/62020 (cit. 12. 05. 2016).

[6] Wikipedia contributors. *Application programming interface*. Ver. 718858917. Wikipedia, The Free Encyclopedia. 6. květ. 2016. URL: https://en.wikipedia.org/w/index.php?title=Application_programming_interface&oldid=718858917 (cit. 12. 05. 2016).

[7] Oxford University Press. *Crowdfunding. Definition of Crowdfunding - Oxford Dictionaries*. 2016. URL: http://www.oxforddictionaries.com/us/definition/american_english/crowdfunding (cit. 12. 05. 2016).

[8] Wikipedia contributors. *Copyright Term Extension Act*. Ver. 719872458. Wikipedia, The Free Encyclopedia. 12. květ. 2016. URL: https://en.wikipedia.org/w/index.php?title=Copyright_Term_Extension_Act&oldid=719872458 (cit. 12. 05. 2016).

[9] Sunstein Kann Murphy & Timbers LLP. *Copyright Flowchart. Flowchart for determining when U.S. copyrights in fixed works expire*. 2016. URL: http://sunsteinlaw.com/practices/copyright-portfolio-development/copyright-pointers/copyright-flowchart/ (cit. 12. 05. 2016).

[10] Librivox. *Librivox API Info*. URL: https://librivox.org/api/info (cit. 12. 05. 2016).

[11] Wikipedia contributors. *Hypertext Transfer Protocol*. Ver. 719567397. Wikipedia, The Free Encyclopedia. 10. květ. 2016. URL: https://en.wikipedia.org/w/index.php?title=Hypertext_Transfer_Protocol&oldid=719567397 (cit. 14. 05. 2016).

[12] Wikipedia contributors. *Uniform Resource Identifier*. Ver. 719814054. Wikipedia, The Free Encyclopedia. 11. květ. 2016. URL: https://en.wikipedia.org/w/index.php?title=Uniform_Resource_Identifier&oldid=719814054 (cit. 14. 05. 2016).

[13] Wikipedia contributors. *XML*. Ver. 717949842. Wikipedia, The Free Encyclopedia. 30. dub. 2016. URL: https://en.wikipedia.org/w/index.php?title=XML&oldid=717949842 (cit. 14. 05. 2016).

[14] Bart. *LibriVox Forums. LibriVox API Discussion Thread*. 23. led. 2015. URL: https://forum.librivox.org/viewtopic.php?p=1085516#p1085516 (cit. 23. 05. 2016).

[15] Wikipedia contributors. *Web crawler*. Ver. 719923098. Wikipedia, The Free Encyclopedia. 12. květ. 2016. URL: https://en.wikipedia.org/w/index.php?title=Web_crawler&oldid=719923098 (cit. 23. 05. 2016).

[16] Wikipedia contributors. *Parsing*. Ver. 718049885. Wikipedia, The Free Encyclopedia. 1. květ. 2016. URL: https://en.wikipedia.org/w/index.php?title=Parsing&oldid=718049885 (cit. 21. 05. 2016).

[17] *Internet Archive Blogs - How Archive.org items are structured*. 2011. URL: http://blog.archive.org/2011/03/31/how-archive-org-items-are-structured/ (cit. 22. 05. 2016).

Příloha A

Dotazník prvního participanta

Dotazník

Před-testové otázky:

1. Pohlaví a věk
   - Žena, 46 let
2. Používáte mobilní zařízení, a pokud ano, jaká? (mobil, tablet?, jiná)
   - Používá menší mobilní telefon se starou verzí systému BlindShell.
3. Co byste na zařízeních, která používáte nejraději změnil/a?
   - Ve svém telefonu (se systémem BlindShell) by uvítala možnost vytáčet kontakty přímo z obrazovky upozornění na zmeškaný hovor.
4. Máte jakékoliv předešlé zkušenosti s audioknihami, pokud ano, jaké?
   - Má zkušenosti s namluvenými knihami pro nevidomé.
   - Dříve poslouchala audioknihy na audiokazetách, dnes je poslouchá zejména na počítači v podobě CD nebo souborů MP3. Také poslouchá namluvené knihy na stránkách YouTube. Na mobilních zařízeních ale ne.
5. Pokud byste pomocí aplikace měla možnost zdarma využívat služby audio knihovny, jaké by bylo vaše očekávání?
   - Ráda by poslouchala audioknihy na mobilu. Očekávala by různé žánry, i encyklopedie.

Uživatelské úkoly:

1. V nabídce telefonu najděte aplikaci LibriVox a spusťte ji.
2. Dlouhým stiskem zamykacího tlačítka po straně telefonu aktivujte nápovědu a poslechněte si její obsah. Ve všech částech aplikace pak stejným stiskem opět vyvoláte nápovědu, vždy pro právě aktivní sekci.
3. V hlavní nabídce aplikace zvolte vyhledávání audioknih a pokuste se libovolným způsobem najít knihu „Romeo and Juliet“ od Williama Shakespeara.
4. V seznamu nalezených knih si u libovolné knihy vyvolejte přečtení informací o knize včetně popisu. Ignorujte prosím nesprávné čtení anglického textu českým hlasem.
5. Nalezněte v seznamu knihu „Romeo and Juliet“ namluvenou roku 2009 a vyvolejte nabídku pro její stáhnutí.
6. V seznamu souborů ke stažení nalezněte pátou kapitolu knihy ve vyšší kvalitě (128Kbps MP3) a dejte ji stáhnout.
7. Vraťte se do hlavní nabídky aplikace LibriVox a vstupte do vaší knihovny.
8. V seznamu knih v knihovně nalezněte knihu „Oliver Twist“ od Charlese Dickense, zobrazte její stažené kapitoly a první kapitolu knihy přehrajte.
9. Vraťte se do seznamu stažených kapitol stejné knihy, nalezněte třetí kapitolu knihy a smažte ji.

Otázky po testování:

1. Bylo pro vás těžké orientovat se v nabídkách a ovládání aplikace?
   - Přiměřeně, angličtina jí dělá problémy.
2. Jsou podle vás možnosti vyhledávání dostatečné, nebo byste jich uvítal/a více?
   - Takto to stačí, možná podle nakladatele nebo roku. Ale spíše jí přijdou dostačující.
3. Myslíte si, že je aplikace LibriVox užitečná jako součást mobilního telefonu pro nevidomé?
   - Myslí, že ano (když nebere v ohledu jazyky). Pro anglicky mluvící uživatele je to výborné, žádné heslo, přihlašování, ani kredit pro poslech.

Příloha B

Dotazník druhého participanta

Dotazník

Před-testové otázky:

1. Pohlaví a věk
   - Žena, 23 let
2. Používáte mobilní zařízení, a pokud ano, jaká? (mobil, tablet?, jiná)
   - Má služební telefon se systémem BlindShell (3-4 měsíce) a telefon Nokia C5 (tlačítkový telefon s hlasovou odezvou).
3. Co byste na zařízeních, která používáte nejraději změnil/a?
   - Je spokojená s telefonem Nokia (rychlost při psaní SMS a procházení). U telefonu se systémem BlindShell jí vadí pomalé psaní.
4. Máte jakékoliv předešlé zkušenosti s audioknihami, pokud ano, jaké?
   - Pouze na počítači, kde přehrává knihy z knihovny KDD.
5. Pokud byste pomocí aplikace měla možnost zdarma využívat služby audio knihovny, jaké by bylo vaše očekávání?
   - Upřednostňuje text čtený hlasovým syntezátorem. V aplikaci, která by četla pomocí hlasového syntezátoru, by uvítala procházení textu, možnost pauzy či záložky.

Uživatelské úkoly:

1. V nabídce telefonu najděte aplikaci LibriVox a spusťte ji.
2. Dlouhým stiskem zamykacího tlačítka po straně telefonu aktivujte nápovědu a poslechněte si její obsah. Ve všech částech aplikace pak stejným stiskem opět vyvoláte nápovědu, vždy pro právě aktivní sekci.
3. V hlavní nabídce aplikace zvolte vyhledávání audioknih a pokuste se libovolným způsobem najít knihu „Romeo and Juliet" od Williama Shakespeara.
4. V seznamu nalezených knih si u libovolné knihy vyvolejte přečtení informací o knize včetně popisu. Ignorujte prosím nesprávné čtení anglického textu českým hlasem.
5. Nalezněte v seznamu knihu „Romeo and Juliet" namluvenou roku 2009 a vyvolejte nabídku pro její stáhnutí.
6. V seznamu souborů ke stažení nalezněte pátou kapitolu knihy ve vyšší kvalitě (128Kbps MP3) a dejte ji stáhnout.
7. Vraťte se do hlavní nabídky aplikace LibriVox a vstupte do vaší knihovny.
8. V seznamu knih v knihovně nalezněte knihu „Oliver Twist" od Charlese Dickense, zobrazte její stažené kapitoly a první kapitolu knihy přehrajte.
9. Vraťte se do seznamu stažených kapitol stejné knihy, nalezněte třetí kapitolu knihy a smažte ji.

Otázky po testování:

1. Bylo pro vás těžké orientovat se v nabídkách a ovládání aplikace?
   - Ne, hodnotí to jako snadné.
2. Jsou podle vás možnosti vyhledávání dostatečné, nebo byste jich uvítal/a více?
   - Možnosti vyhledávání hodnotí jako dostatečné.
3. Myslíte si, že je aplikace LibriVox užitečná jako součást mobilního telefonu pro nevidomé?
   - Protože poslouchá knihy spíše na počítači, myslí, že to nemůže úplně posoudit. Každopádně je přesvědčena, že hodně lidí aplikaci uvítá.

Příloha C

Obsah přiloženého CD

- `Testování`
  - `Audiozáznam testování - Participant 1.mp3`
  - `Audiozáznam testování - Participant 2.mp3`
  - `Dotazník - Participant 1.pdf`
  - `Dotazník - Participant 2.pdf`
- **`Zdrojové kódy aplikace`**

  (*Více o struktuře v kapitole* 3.3)
  - **`java`**

    (*Složka se zdrojovými kódy programu ve formátu* .java)
  - **`layout`**

    (*Složka se zdrojovými soubory rozvržení uživatelského rozhraní ve formátu* .xml)
  - **`strings`**

    (*Složka se zdrojovými soubory textových řetězců v českém a anglickém jazyce ve formátu* .xml)
- `BP_Kryze_2016.pdf`

  (*Tato práce v elektronické podobě*)