Bakalářská práce

MOBILNÍ APLIKACE SPOTIFY PRO NEVIDOMÉ

František Smrž

14. SRPNA 2020

VEDOUCÍ PRÁCE: ING DANIEL NOVÁK, PH.D.
České vysoké učení technické v Praze. Fakulta elektrotechnická, Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Smrž** Jméno: **František** Osobní číslo: **474687**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Zpřístupnění služby Spotify pro nevidomé uživatele

Název bakalářské práce anglicky:

Spotify access implementation for visually impaired users

Pokyny pro vypracování:

Projekt se soustředí na zpřístupnění služby Spotify nevidomým uživatelům. Cílem projektu je vytvoření mobilní aplikace pro Android, která bude komunikovat s API společnosti Spotify. Nesdílnou součástí projektu je i návrh rozhraní, a to společně s uživateli s využitím technologie návrhu zaměřeného na uživatele (tzv. user-centered design).

1. Seznamte se s principy návrhu softwarových řešení pro nevidomé uživatele
2. Zpřístupněte rozhraní aplikace Spotify pro nevidomé uživatele.
3. Rozhraní otestujte na 5 nevidomých uživatelích

Seznam doporučené literatury:

[1] Matt May , Wendy Chisholm , Universal Design for Web Applications, 2008
[2] Universal Design - World Blind Union, 2020

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Daniel Novák, Ph.D., Analýza a interpretace biomedicínských dat FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **30.09.2021**

[redacted]

doc. Ing. Daniel Novák, Ph.D.
podpis vedoucí(ho) práce

[redacted]

podpis vedoucí(ho) ústavu/katedry

[redacted]

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studenta

Poděkování

Velice rád bych poděkoval vedoucímu práce Ing. Danielu Novákovi, Ph.D. za cenné připomínky, doc. Ing. Janovi Hadáčkovi za rady a za pomoc při vytváření aplikace a bych také velice rád poděkoval účastníkům testovaní z organizace SONS.

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 14. srpna 2020

X

Podpis autora

Abstrakt

Tématem práce je vytvoření mobilní aplikace určené pro zrakově nevidomé osoby, který umožní poslouchat hudbu přes aplikaci Spotify, která je nyní nejrozšířenější aplikací pro poslech hudby v mobilním průmyslu. Spotify poskytuje vlastní rozhraní, kde lze získat dostupná data. Aplikace se zabývá jejím zpracováním, přehráváním a ukládáním. Aplikace potřebuje pro provoz mobilní platformu Android.

Klíčová slova

Spotify, nevidomý, hudba, mobilní aplikace, Android

Abstract

The topic of the work is the creation of a mobile application designed for the visually blind, which will allow you to listen to music through the Spotify application, which is now the most widespread application for listening to music in the mobile industry. Spotify provides its own interface where available data can be obtained. The application deals with its processing, playback and storage. The application needs an Android mobile platform to run.

Keywords

Spotify, blind, music, mobile applications, Android

Obsah

Zkratky

API	Application Programming Interface
JSON	JavaScript Object Notation
UX	User Experience
UI	User Interface
OAuth 2.0	industry-standard protocol for authorization

Termíny

Playlist – Soubor písniček

Google play - Aplikace na stahování dalších aplikací od společnosti Google pro Android.

App Store – Aplikace na stahování dalších aplikací od společnosti Apple.

Aplikace Spotify – Mnou vytvořená aplikace Spotify pro nevidomé.

Spotify – Veřejně dostupná aplikace od společnosti Spotify.

Endpoint – Url adresa na které se dá připojit a získat data.

Query - Ve světě databází se tak označuje příkaz, kterému se říká dotaz.

Shared Preferences – Soubor aplikace, který umožňuje ukládání klíč-hodnota dat.

Open Source – Otevřená dostupnost kódu.

WebView – Komponenta, která umožňuje zobrazení webové stránky nebo aplikace na Androi

1. Úvod

Mobilní telefony jsou pro nás v dnešní době nejen něčím velmi přínosným, něčím co nám drasticky usnadňuje život, ale tyto malé přístroje se pro nás staly až nepostradatelnými. Mobilní telefony, které byly na počátku vytvořeny jen pro hovory, nebo posílání zpráv, nám dnes umožňují řadu funkcí, jako je například přístup na internet, fotografování, natáčení videí, navigace, poslouchání hudby a mnoho dalších. V roce 2019 podle statistik používá mobilní telefon naprostá většina obyvatel, konkrétně celých 96% a z toho 81% tvoří uživatelé používající chytré mobilní telefony, které nám díky operačním systémům jako je Android nebo IOS umožňují stahovat do našeho zařízení nejrůznější aplikace, které dobře poslouží například při objednávání jídla, doporučí nám nejbližší restauraci, upozorní nás na stav bankovního účtu, nebo třeba přeloží text do jiných jazyků. Je toho opravdu spousta, co nám mobilní telefony dnešní generace mohou nabídnout. Co na trhu má ale stále velké mezery jsou aplikace pro nevidomé. Až donedávna bylo pro nevidomé téměř nereálné nějak efektivně a samostatně pracovat s mobilním telefonem. Naštěstí tato problematika se pomalu ale jistě začíná více dostávat do povědomí vývojářů a výrobců mobilních telefonů a aplikací, a přinášejí tak pro nevidomé čím dál tím víc usnadnění a nových možností, aby mohli mobilní telefony používat stejně neomezeně a efektivně jako běžní uživatelé. Dnes už mají nevidomí možnost zakoupit si mobilní telefon přímo pro ně, který je přizpůsobený právě pro jejich pohodlné používání bez pomoci kohokoliv jiného. Do takového telefonu si mohou stáhnout pár desítek aplikací, které již byly přímo pro tento účel vytvořeny. Právě pro to jsem se rozhodl podílet na vytváření aplikací pro nevidomé. Vybral jsem si aplikaci Spotify, protože je mi hudba blízká. Nejen že tak budou moci hendikepovaní lidé poslouchat hudbu, ale budou si jí i moct sami najít, stáhnout, vytvářet oblíbený alba, v podstatě vše, na co jsou již běžní uživatelé Spotify už léta zvyklí.

1.1. Cíl práce

Cílem práce je vytvoření mobilní aplikace „Spotify“ pro přehrávání hudby, která běží již na vytvořeném mobilním systému pro nevidomé „Blind Shell“. Celý systém „Blind Shell“ je vytvořen pro nevidomé lidi a již obsahuje pár aplikací pro nevidomé. Aplikace Spotify umožní nevidomým lidem rychle, jednoduše a bez jakýchkoliv obtíží přehrávat hudbu, kterou mají rádi. Aplikace bude umět přehrávat písničky, alba, playlisty. Dané písničky bude moci ukládat do kategorie oblíbených, které budou řazeny do dalších třech kategorií (písničky, alba, interpreti) v kterých půjde procházet.

Spotify bude implementována pod systémem Android v programovacím jazyce Java. Protože aplikace běží na systému „Blind Shell“, který je stavěný pro nevidomé lidi, nebude z pohledu UI a celkově grafického rozhraní tak propracovaný, tudíž se naopak musí více soustředit na implementaci uživatelského rozhraní UX a více dbát na použitelnost. O tuto část se stará „Blind Shell“ aplikace, proto s určitými problémy už počítá a řeší je. Jeden ze základních principů je převedení textu do hlasové podoby, a tímto způsobem aplikace informuje uživatele o průběhu dění. Dále řeší pohybování mezi meny a položkami jako základní stromovou strukturu. Proto se aplikace Spotify bude především zabývat získáváním dat z API Spotify a jejich zpracováním a ukládáním. Jako jedna ze základních potřeb každé hudební aplikace je vyhledávat písničky podle určitých kritérií. Poté také přidávání písniček či alb do oblíbených záložek, přetáčení písniček, pouštění celých alb a dalších potřeb uživatel hudební aplikace.

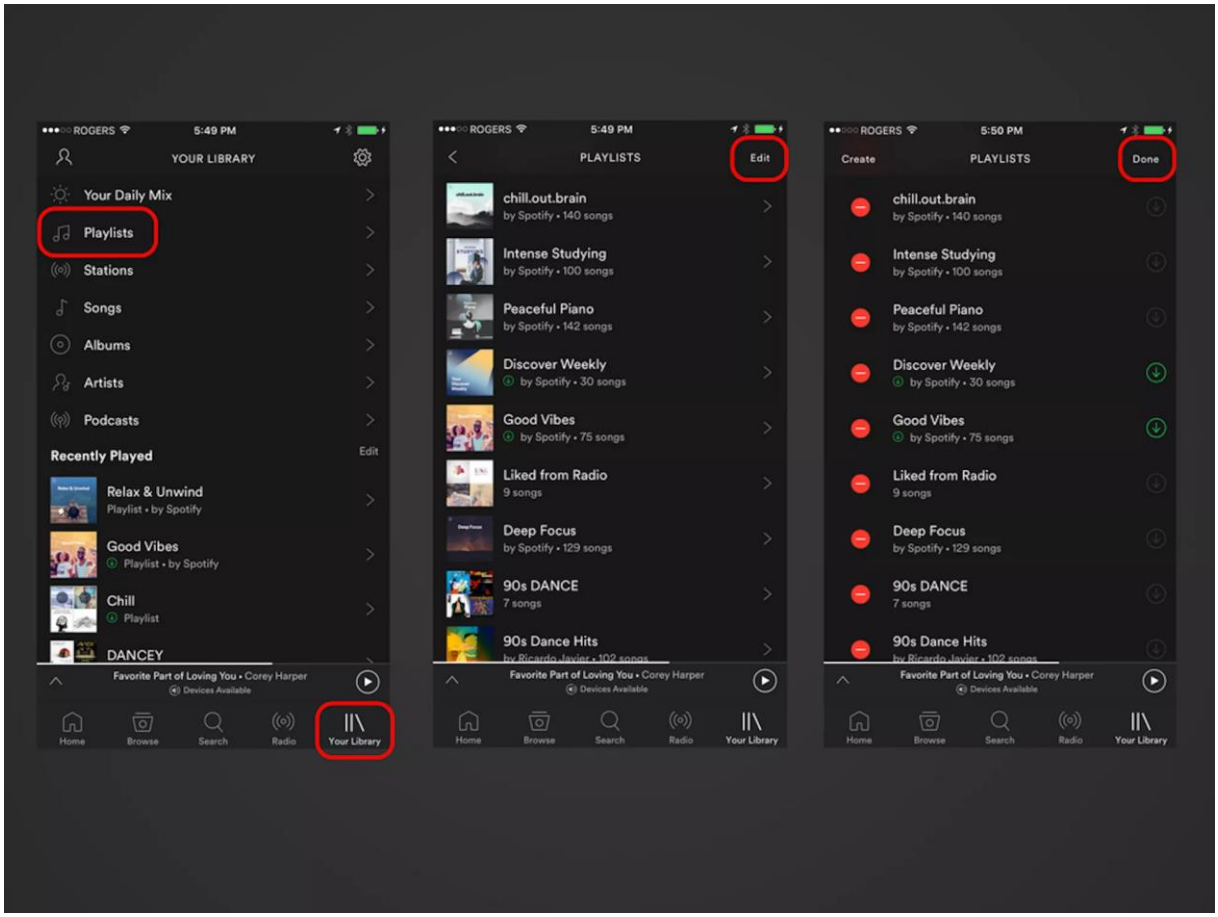
Posledním bodem mé bakalářské práce bude proces, který finálně otestuje funkčnost a použitelnost mojí aplikace. Ten bude spočívat právě v testování mé aplikace nevidomými uživateli, díky čemuž se dozvím, jak aplikace funguje v praxi, a zda se s ní nevidomým pracuje pohodlně. Testy mi také případně pomůžou odhalit, zda má moje aplikace nějaké mezery, či nedostatky, které jsem přehlédnul. A pokud tedy tyto testy proběhnou úspěšně, je moje práce na tomto projektu u konce, a nevidomí ji budou moci oficiálně používat.

2. Analýza existujících aplikací

Tato část se bude zaobírat analýzou již existujících aplikací, které poskytují poslech hudby. Aplikace budou řazeny podle podobnosti funkcionalit a celkové použitelnosti aplikace. Některými z nich se moje aplikace Spotify inspiruje, a doplňuje užitečné a použitelné techniky a principy. Všechny aplikace budou volně dostupné na službě Google Play nebo App Store nebo na internetových stránkách dané aplikace, které budou uvedeny v příloze. [1]

2.1. Spotify

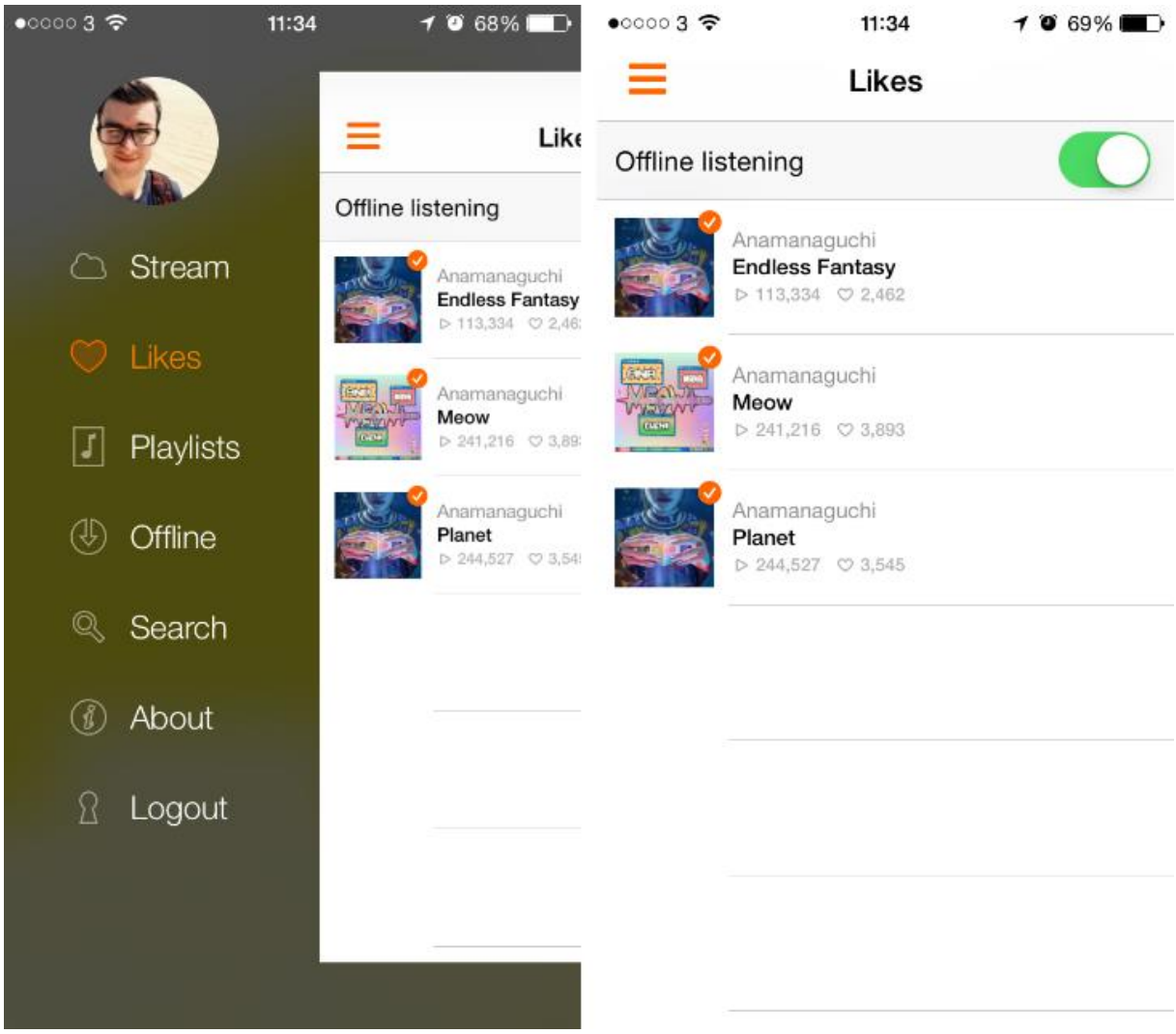
Služba Spotify je původem ze Švédska a vznikla v roce 2006 v hlavním městě Švédska, Stockholmu. Slouží k poslechu a vyhledávání hudby. Službu Spotify lze rozdělit na Spotify Free a Spotify Premium. Spotify Premium se od Spotify Free liší tím, že poskytuje uživatelům přehrávání hudby ve vyšší kvalitě, a zároveň neruší uživatele reklamami mezi poslechem jednotlivých skladeb. Jedinou nevýhodou je, že se za služby Spotify Premium musí měsíčně platit $9.99. Jelikož pracuji na aplikaci Spotify, která vychází ze Spotify a používá její API, tak se musím držet základní funkcionality, která API umožňuje.



Obrázek 1 Náhled do aplikace Spotify

2.2. SoundCloud

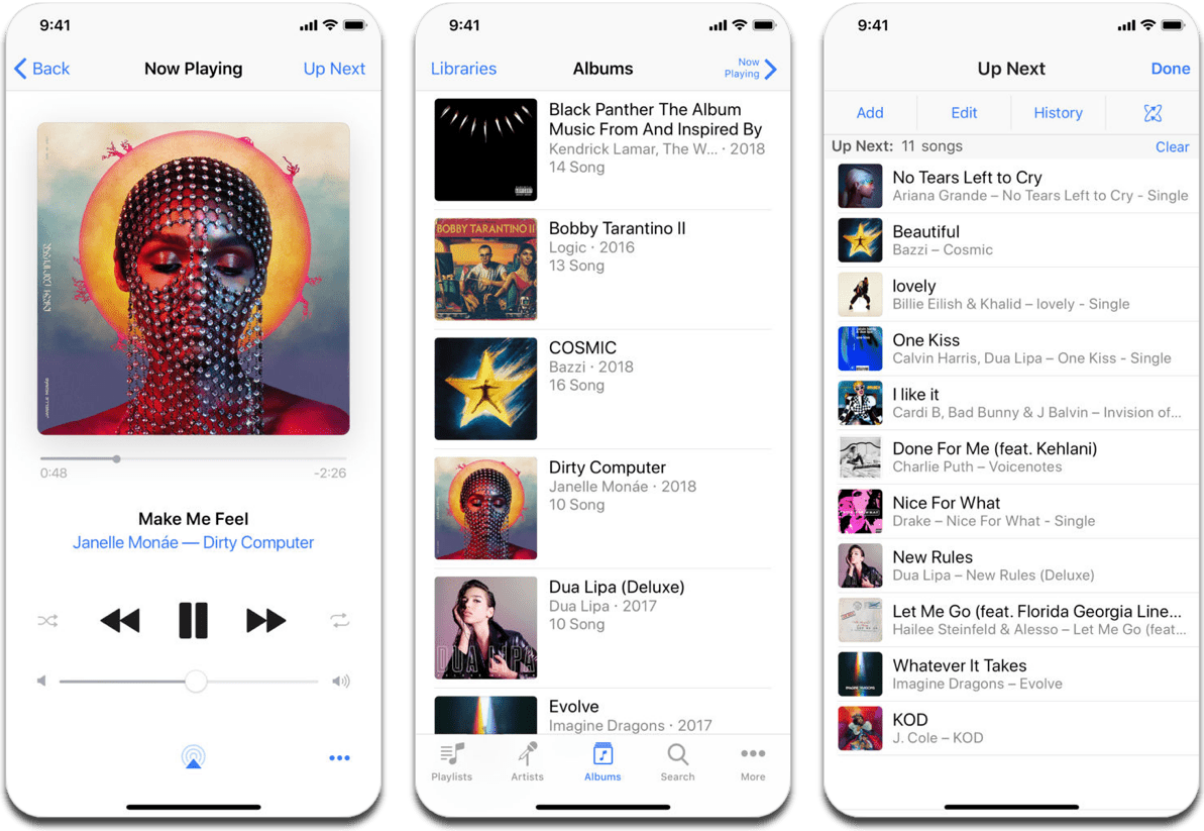
SoundCloud je internetová stránka a také aplikace sloužící k distribuci hudby. Byla založena v roce 2007 v Berlíně. Jeho zakladateli jsou Alex Ljung a hudebník Eric Wahlforss. Původně měl sloužit jen ke sdílení hudby mezi hudebníky samými, a ne s celou širokou veřejností. Na SoundCloud je možné ukládat vlastní hudební nahrávky, ať už je to zpěv, či jen nahrávky instrumentálního podání. Tyto nahrané soubory jsou pak k dispozici komukoliv, kdo využije služeb SoundCloud. Pokud se vám některá skladba bude líbit, můžete si ji stáhnout přímo do svého zařízení. Také pokud si někdo stáhne vaši nahrávku, ukáže se vám, kolik lidí tak již učinilo. Stejně tak uvidíte počet přehrání, či počet “liků”.



Obrázek 2 Menu a oblíbené v aplikaci Soundcloud

2.3. iTunes

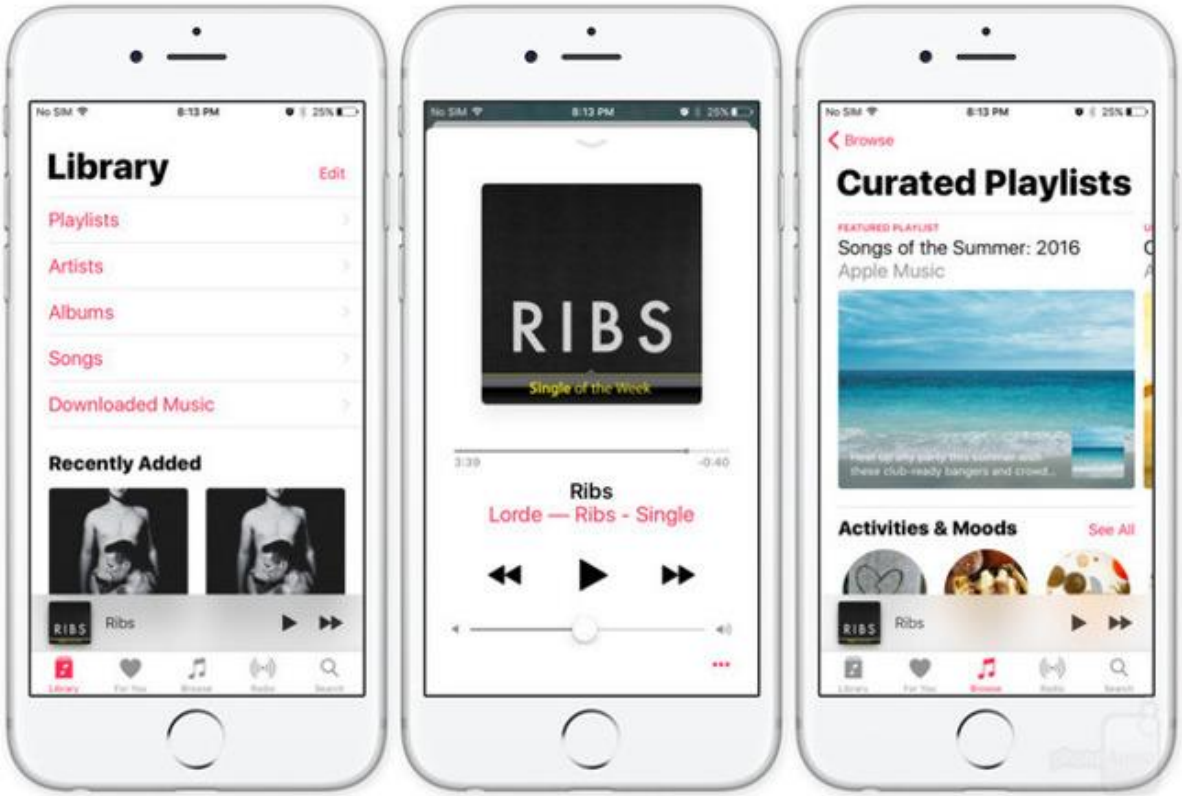
Aplikace iTunes slouží k přehrávání a organizování multimediálních souborů. Dále také slouží jako rozhraní pro správu zařízení společnosti Apple, jako jsou iPhone, iPad, nebo iPod. iTunes byl představen Steavem Jobsem a Steavem Wozniakem v roce 2001. ITunes s sebou přináší možnosti jako je vytváření vlastních hudebních seznamů, úprava informací o nahrávkách, řazení hudebních souborů podle různých kritérií, a nebo třeba i kupování hudby díky vestavěnému hudebnímu e-shopu iTunes Store. Důležitou funkcí, kterou s sebou iTunes přináší je také možnost zálohování hudby na kompaktní disky. Nejnovější verzi aplikace iTunes je dnes možné stáhnout přímo na webu Apple.com pro operační systémy 10.8.5 a vyšší, nebo na Windows 7 a vyšší.



Obrázek 3 Náhled do aplikace iTunes

2.4. Apple Music

Služba Apple music představuje hudební streamovací platformu od americké společnosti Apple. Funguje tak, že si uživatel vyhledá skladbu, kterou si chce pustit, a za současného poslechu se mu hudba stahuje (streamuje). Zároveň má možnost uložit si skladbu do zařízení, aby k ní měl přístup i offline, tzn. bez přístupu k internetu. Když uživatel vstoupí na stránky apple music, a zaregistruje se, získá možnost si v prvním kroku zvolit hudební žánry, či interprety, kteří ho zajímají, a na základě těchto nasbíraných informací mu bude služba Apple music v budoucnu doporučovat nový obsah, který by se mohl uživateli líbit. Ten lze najít v záložce zvané "pro Vás". Jednou z dalších výhod služby Apple music je přítomnost blogovací platformy Connect, na které mohou interpreti sdílet určitý obsah s fanoušky. Apple music je také spjato s hlasovou asistentkou Siri, která na příkaz uživatele přehraje jím vyřčenou skladbu, či přehraje momentální nejlepší skladby v žebříčku.



Obrázek 4 Náhled do aplikace Apple Music

3. Návrh a analýza řešení

V této kapitole se věnuji návrhu řešení a analýze aplikaci Spotify pro nevidomé, která se bude především zabývat získáváním dat z API Spotify a jejich zpracováním a ukládáním. Jako jedna ze základních potřeb každé hudební aplikace je vyhledávat písničky podle určitých kritérií. Pro tento případ se používá API. API je vývojářské webové rozhraní, které umožňuje získávat dat z databáze za pomocí různých požadavků.

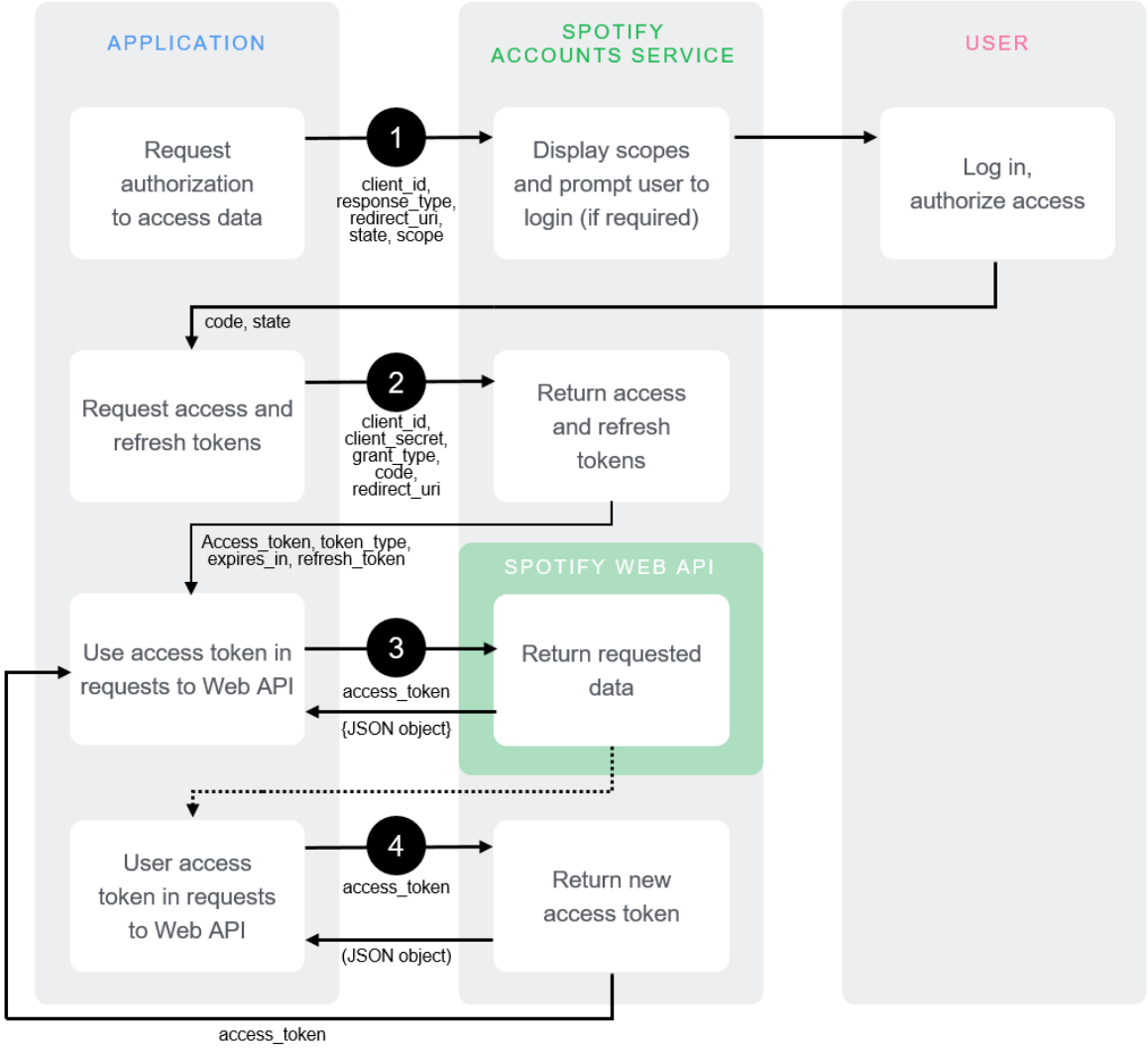
3.1. Získávání dat

Jak už bylo zmíněno, hlavní potřebou aplikace Spotify, je získávání dat, aby aplikace vůbec něco dokázala přehrát. Spotify má své vývojářské API volně dostupné na internetu, které přistupuje do databáze Spotify, kde můžeme naleznout veškerou databázi písniček, alb a interpretů. Na začátek je potřeba říct, jak se k API vůbec dostat. Připojení API ne nejčastěji chráněno určitým zabezpečením, aby ho nikdo nemohl jen tak zneužít.

Nejprve musíte zařídit, aby vaše aplikace mohla komunikovat s API a poté se bude moci autorizovat. Aplikace Spotify se musí zaregistrovat mezi aplikace, které budou využívat Spotify API, to lze přes webové rozhraní na stránkách Spotify, kde při zaregistrování obdržíte url, kde bude probíhat komunikace, Client ID, který je vaše hlavní identifikace aplikace a Secret ID, je klíč, který posíláte při zabezpečovacích dotazech na Spotify API.

3.1.1. Autentizace

Spotify API zabezpečení autorizuje aplikaci Spotify pro přístup k jejím datům a funkcím Spotify. Zabezpečení Spotify API je definováno OAuth 2.0 specifikací. Autorizace Spotify API lze různými možnosti: uživatelem obnovující autorizace, dočasná autorizace a aplikací obnovující autorizace. Dočasná autorizace je určená pro ty aplikace, které jsou implementováni pomocí JavaScriptu a běží v prohlížeči. Podávání dotazů a tím pádem získávání dat je rychlejší, ale neumožňuje získat žádný obnovovací token, který je potřeba, když nechcete, aby se uživatel opětovně přihlašovat, a proto se pro aplikaci Spotify nehodí. Aplikací obnovující autorizace se používá při ověřování mezi servery. Přístup pro aplikace, které nemají přístup k informacím o uživateli. Výhodou je že se nemusí používat autorizačního přístupového tokenu, a proto se také moc nehodí pro aplikaci Spotify. Uživatelem obnovující autorizace se nejvíce hodí pro dlouhodobé aplikace, ve kterých uživatel uděluje oprávnění pouze jednou. Poskytuje přístupový token, který lze obnovovat a tím zajistit, že se uživatel nemusí opakovaně přihlašovat. Proto se nejlépe hodí pro aplikaci Spotify.



Obrázek 5 Uživatelem obnovující autorizace - (Refreshable user authorization)

Pro celkovou komunikaci mezi Spotify API a aplikací Spotify řeší metody protokolu HTTP jako GET, POST, PUT atd., které zajišťují získávání a dat. Tyto metody požadují definici odpovídajících struktur (např. XML, HTML, JSON), se kterou se poté pracuje. Spotify Api definuje jako odpovídající strukturu JSON. Tato data mohou být organizována v polích nebo v objektech.

Prvním krokem k autentizaci je inicializace aplikace, a to pomocí HTTP dotazu GET, který se pošle na stanovenou url. Tento dotaz vás dostane na klasickou přihlašovací url Spotify.

Endpoint: `GET https://accounts.spotify.com/authorize`

Příkladem takového dotazu je např.

GET https://accounts.spotify.com/authorize?client_id=5fe01282e44241328a84e7c5cc169165&response_type=code&redirect_uri=https%3A%2F%2Fexample.com%2Fcallback&scope=user-read-private%20user-read-email&state=34fFs29kd09

Pokud získáte autorizační kód tak se může přejít k druhému kroku vyplnění údajů na přihlašovacím formuláři a odeslání dalšího dotazu. Další dotaz je typu POST, který ve svém tělíčku musí obsahovat zakódovaný parametr v *application/x-www-form-urlencoded*.

Endpoint: `POST` https://accounts.spotify.com/api/token

Odpověď na tento dotaz přijde ve formátu JSON, ve kterém přijde přístupový token a obnovovací token, přístupový token slouží k autentizaci aplikace pro přístup do Spotify API, díky pak kterému může aplikace Spotify získávat data pro pouštění písniček, alb atd.

Příklad JSON, který přijde jako odpověď na dotaz.


```
{
    "access_token": "NgCXRK...MzYjw",
    "token_type": "Bearer",
    "scope": "user-read-private user-read-email",
    "expires_in": 3600,
    "refresh_token": "NgAagA...Um_SHo"
}
```


Přístupový token má ale pouze určitou životnost, která je také uložena v získaném JSONu a je stanovena na 3600 vteřin. Když tento token vyprší, pomocí obnovovacího tokenu ho lze obnovit. Obnovit ho lze zase před dotaz na Spotify API, který je typu POST a v tělíčku dotazu se pošle obnovovací token a přijde nový přístupový token.

Endpoint: `POST` https://accounts.spotify.com/api/token

Příklad JSON, který přijde jako odpověď na dotaz.


```
{
    "access_token": "NgA6ZcYI...ixn8bUQ",
    "token_type": "Bearer",
    "scope": "user-read-private user-read-email",
    "expires_in": 3600
}
```

3.1.2. Data

Po autorizovaní aplikace se zpřístupní druhá část API, která umožňuje získávání dat o autorech, albách a písničkách. Data se jednoduše získávají pomocí HTTP GET requestu na určité endpointy. Každý endpint má svojí vlastní adresu, na kterou se může poslat query pro získání určitých dat. Query obsahuje různé parametry. Podle těchto parametrů např. „type“ se určuje, jestli hledáme písničku nebo celé album. Dalším parametrem je „market“, který rozhoduje pro kterou zemi chceme vyhledávat, protože některé písničky nejsou povoleny ve všech zemích. Dále můžeme nastavit limit výsledků. Query také musí obsahovat hlavičku, kde se specifikuje v jakém formátu se data budou vracet a dále se v ní posílá autorizační token, který byl získán při autorizaci.

Příklad pro query vyhledání písničky:

```
"GET"
"https://api.spotify.com/v1/search?q=tania%20bowra&type=artist" -H "Accept: application/json" -H "Content-Type: application/json" -H "Authorization: Bearer BQBAc4JdEKTPqaDBfyJw3jAXL0bxZ5s88OeanG04ExuaKAovspNNTthJ38k05ZlMIyI7XklN_mxy2d6vYI9jXzsK3s4FsvkgF2rWIp9AiKeeUIanFoOu-bxl_DZNV7_uCmJTQxzApR8k3qwFTeJv_MS3CI2DZ-NZUVkEzAlB-gk0APZC"
```

Odpověď API:

```
{
  "artists": {
    "href": "https://api.spotify.com/v1/search?query=tania+bowra&type=artist&offset=0&limit=20",
    "items": [
      {
        "external_urls": {
          "spotify": "https://open.spotify.com/artist/08td7MxkoHQkXnWAYD8d6Q"
        },
        "followers": {
          "href": null,
          "total": 201
        },
        "genres": [],
        "href": "https://api.spotify.com/v1/artists/08td7MxkoHQkXnWAYD8d6Q",
        "id": "08td7MxkoHQkXnWAYD8d6Q",
        "images": [
          …
        ],
        "name": "Tania Bowra",
        "popularity": 1,
```

```
        "type": "artist",
        "uri": "spotify:artist:08td7MxkoHQkXnWAYD8d6Q"
      }
    ],
    "limit": 20,
    "next": null,
    "offset": 0,
    "previous": null,
    "total": 2
  }
}
```


Obrázek 6 - Odpověď API ve formátu JSON na GET HTTP request (Vyhledání písničky).

3.2. Ukládání dat

Další otázkou je, jak ukládat již získaná data. Pro určitá data bude jistě potřeba databáze, jako například ukládání písniček, alb a artistu, jelikož vyžadují mezi sebou určitou strukturu, artista má více alb a album má mnoho písniček. Na druhou stranu databáze na uložení přihlašovacích údajů uživatele nebo například různé druhy autorizačních tokenu asi nebude nejvhodnější varianta, pro tyto účely má Android vlastní řešení a tím jsou Shared Preferences.

Shared Preferences fungují jako jednoduché úložiště dat s reprezentací klíč-hodnota. Pro použití Shared Preferences se nejprve musí vytvořit úložiště kam se pak budou moc data ukládat. Pro vkládání dat se otevře úložiště a pomocí Shrared Preferences editoru se přidávají pomocí klíč hodnota data.

Příklad vložení dat:

```
sharedpreferences = getAplicationContext().getSharedPreferences("Preferences", 0);
Editor editor = sharedpreferences.edit();
editor.putString("key", "value");
editor.commit();
```

Poté lze zase data získat:

```
sharedpreferences = getAplicationContext ().getSharedPreferences("Preferences", 0);
String value = preferences.getString("key", null);
```

Pro strukturované data existuje spousta druhů databází, proto jsem vytvořil tabulku podle které jsem se rozhodoval jakou databázi použít.

Databaze	Simplicity	Encryption	Speed	Size	Data Migration	Sum
SQLite	6	8	7	9	8	**38**
Realm	9	5	8	5	6	33

SQLite je jedna z nejčastějších variant offline databází u android aplikací. Základním cílem SQLite je zbavit se server-client architektury. Ukládá všechny data přímo na mobilním zařízení. Realm je non-relation databáze, která vám umožní vytvářet relace mezi objekty stejně, jako u většiny programovacích jazyků. Realm je o mnoho novější jak SQLite.

Z tabulky vyšlo SQLite jako lepší varianta, proto jsem se jí rozhodl použít.

3.3. Přehrávání

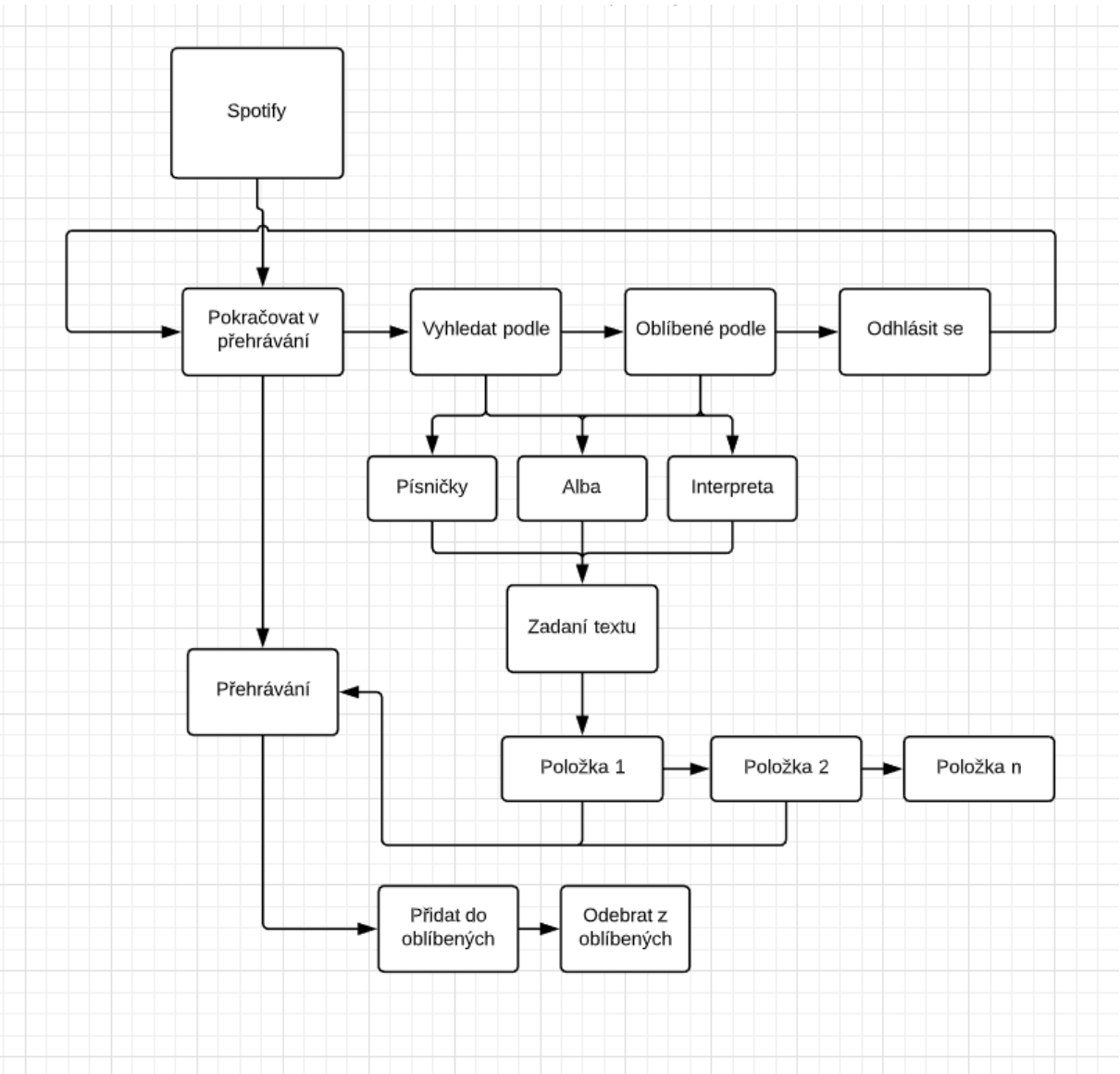
Samotné přehrávaní písniček v androidu je zprostředkováno pomocí MediaPlayeru, který dokáže zpracovat proud data a přehrát je, ale jelikož Spotify API poskytuje pouze URL daných písniček, alb a autorů, tak je pro toto využití příliš složité. Existuje knihovna SpotifyPlayer, která dokáže ze získané URL ze Spotify API dostat proud dat, které následně přehraje. Proto je pro moje využití perfektní. Po získání URL písniček lze jednoduše do SpotifyPlayeru tuto URL dodat a knihovna zajistí přehrání určité písničky. Avšak tato knihovna má nějaké nepříjemné drobnosti, bohužel nepodporuje přetáčení v písničkách ani nepodporuje nějaký druh časovače. Proto tyto vlastnosti budou muset být implementovány zvlášť.

3.4. Klíčové vlastnosti

V této části bych rád shrnul co přesně by moje aplikace Spotify měla dodat uživatelům. Podle analýzy existujících aplikací mají všechny aplikace jeden zásadní problém a tím je, že obsahují příliš mnoho funkcí a celkově se v nich hůře orientuje. Protože aplikace je primárně vyvíjena pro nevidomé nebo slabozraké, je orientace stěžejní bod aplikace, proto by aplikace měla být co nejjednodušší a dobře přehledná. Ale i přes tyto požadavky by aplikace měla být schopná přinést uživatelům plný užitek aplikace Spotify.

3.5. Struktura aplikace

Jedna z nejdůležitějších částí aplikace je, aby se nevidomý uživatelé dobře orientovali. Měli by vždy vědět, kde se ve struktuře aplikace nachází a v jakém stavu se nachází aplikace. Struktura by měla zachovávat jednoduchost a nebýt příliš složitá. Tato problematika bude konzultována s experty ze společnosti SONS a návrh rozložení aplikace je vidět na Obrázku 2.



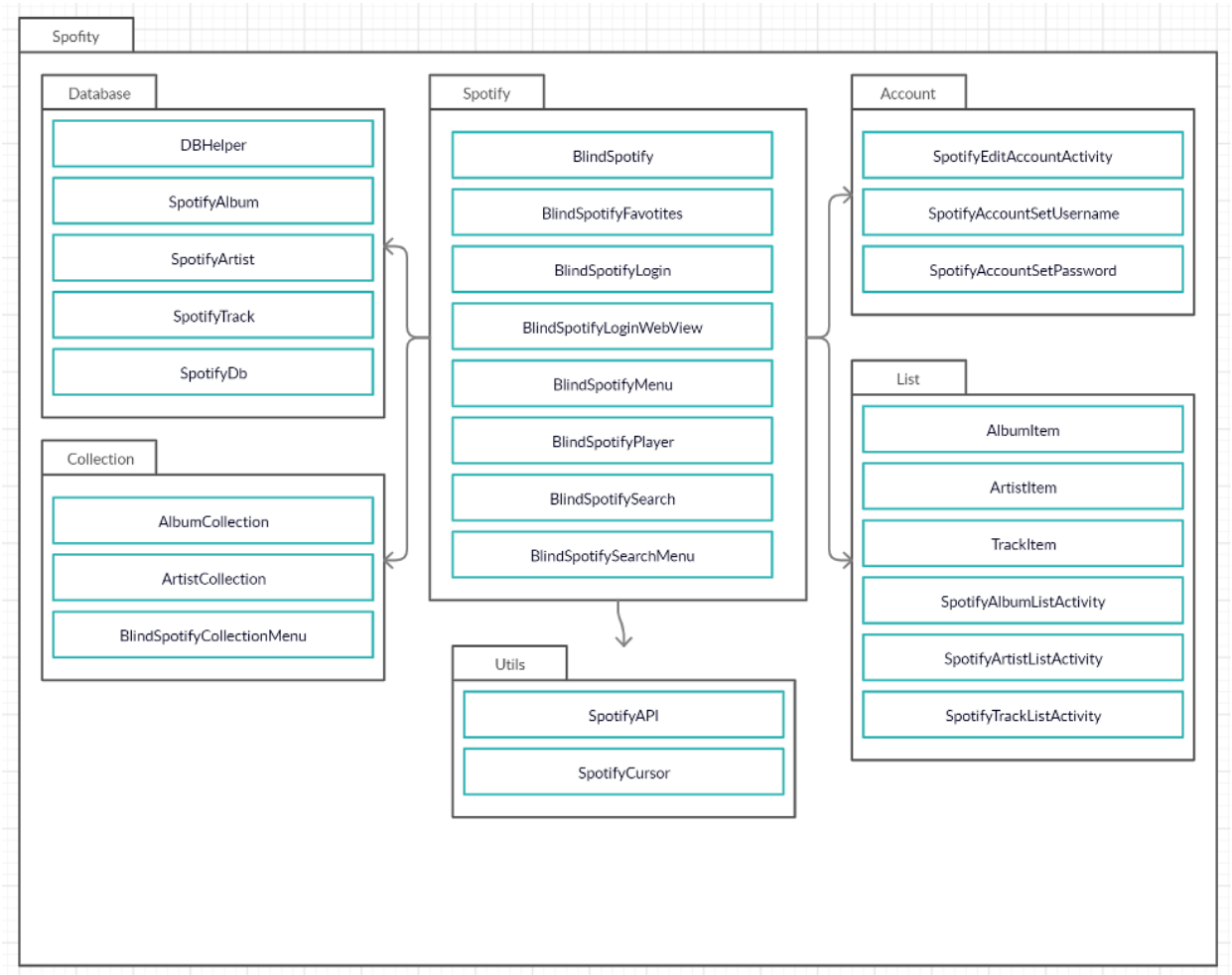
Obrázek 7 Navrhovaná struktura aplikace Spotify (Wireframes)

3.6. Shrnutí analýzy

Pro aplikaci Spotify je nejdůležitější částí získávání dat a poté přehrávání, pro získávání dat bude sloužit Spotify API, kdy data budou poskytnuta před HTTP dotazy na server. Data se budou vracet ve formátu JSON. Přehrávání je realizováno přes knihovnu SpotifyPlayer. Aplikace Spotify bude obsahovat vyhledávání písniček, alb a autorů. Následující přehrávání písniček, také možnost uživatele si písničky uložit. Ukládány budou do struktury autor-album-písnička implementované pomocí databáze. V písničkách se bude moci posouvat v čase.

4. Implementace

Tato část se bude zabývat implementací aplikace Spotify pro přehrávání písniček. Aplikace Spotify je implementována již do existujícího systému BlindShell, který je přímo upravený pro nevidomé. Proto je důležité, aby aplikace dodržovala již vytvořenou strukturu a syntaxi. Určité používání tříd a metod, které třídy aplikace dědí, jsou převzaté ze systému BlindShell. Proto se některé třídy označují řetězcem „Blind“.



Obrázek 8 Package diagram

4.1. Menu

Menu aplikace Spotify je vytvořena pomocí tříd *BlindMenuActivity,* která zajišťuje pohybování se po aplikaci pomocí mobilních inputů. Jedno z hlavních menu aplikace je BlindSpotifyMenu, do toho menu se uživatel dostane, pokud spustí aplikaci Spotify. Menu obsahuje položky „Pokračovat v přehrávání“, „Kolekce“, „Oblíbené“, „Vyhledávaní“, „Odhlášení“ většina těchto položek vede na další menu z kterých se pak volají určité aktivity. Samotné menu je pak

tvořeno z tříd *BlindMenuFragment.* Při *onCreate* funkci se vytvoří tato třída a vloží se do ní *BasicMenuItem,* nebo samotné třídy. Tyto parametry reprezentují položky v menu.

```
@Override
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);

	setSectionsPagerAdapter(new BlindStaticMenuAdapter(getSupportFragmentManager(),
	new BlindMenuFragment[]{
		new ContinuePlayingMenuItem(),
		new BasicMenuItem(BlindSpotifyFavorites.class,
		getString(R.string.app_spotify_favorites)),
		new BasicMenuItem(BlindSpotifyCollectionMenu.class,
		getString(R.string.app_spotify_collection)),
		new BasicMenuItem(BlindSpotifySearchMenu.class,
		getString(R.string.app_spotify_search)),
		new SpotifyLogout(),
	}));
	super.finishCreate();
}
```

Obrázek 9 - Ukázka vytvoření hlavního menu

4.2. API

Získávání dat pro přehrávání zajišťuje třída SpotifyApi. Pro posílání HTTP požadavků je využívána knihovna Volley.[X] Data je potřeba získat všechny naráz, až se kompletně navrátí z požadavku. Proto jsou tu použity callbacky. Zavoláte třídu SpotifyApi s callbackem a čekáte až se vrátí callback s daty, abyste s nimi mohli dál pracovat.

4.2.1. Knihovna Volley

Knihovna Volley je Open Source a je v Android dokumentaci doporučována pro práci s http dotazy. Umožňuje jednoduché používání http dotazů, automatické plánování dotazů, které pak může různě prioritizovat. Knihovna využívá frontu dotazů (RequestQueue) dále jen jako fronta, která je realizována jako singleton. Funguje jako obyčejná fronta, do fronty přidáváte HTTP dotazy, které jsou

potom postupně vykonávány podle pořadí, jak přišly. Vytváření a zpracování dat je vykonáváno ve třídě SpotifyApi. Z této třídy jsou poté vráceny pomocí callbacku, zpět z volající třídy.

4.2.2. SpotifyApi

SpotifyApi je třída, která zajišťuje veškerou komunikaci s API Spotify. Třída pracuje s Callbackem, který je realizován pomocí abstraktní statické třídy vnořené v třídě SpotifyApi. Ukázka třídy Callback dále jen Spotify Callback.

```
public static abstract class Callback<T> {
    public abstract void invoke(String searchUrl, List<T> searchResults, String nextPageToken);
    public abstract void error(String message);
}
```

Pro vytváření HTTP dotazů slouží přípravné funkce, které nejprve připraví dotaz a vloží se do něho všechny potřebné parametry, jako například ve funkci *searchByTrack.* Tato funkce má vyhledat písničku podle vloženého textu. Funkce má jako parametr hledaný název písničky a Spotify Callback. Ve funkci se vezme URL na které je Endpoint určený pro search a přidají se mu parametry dotazu: *Type, Limit, Market, Offset* a další. V tomto případě to bude *Type,* který je roven hodnotě „track“ písnička. Poté se zavolá funkce *searchTrack.*

```
public static void searchByTrack(String track, SpotifyApi.Callback callback){
    String queryUrl = QUERY + track.replaceAll("\\s+","+");
    String searchUrl = SEARCH_URL + queryUrl + TYPE + "track";
    searchTrack(searchUrl, searchUrl, callback);
 }
```

Funkce *searchTrack* vezme připravenou URL s parametry a pomocí knihovni Volley, vytvoří HTTP dotaz a pošle ho do fronty, kde se pak dále pošle na určitý Endpoint. K tomuto HTTP dotazu je potřeba připojit hlavičku. Parametry hlavičky jsou: v jakém formátu se mají vrátit data *JSON,* Autorizační token, který funkce získá ze SharedPreferences. Funkce *searchTrack* také obstarává zpracování vrácených dat z Endpointu. Jelikož data přicházejí v JSON formátu. Tak se pak pomocí JSONObject postupně získávají data a vytváří se z nich objekty aplikace, v tomto případě hledané písničky.

```
private static void searchTrack
(String newSearchUrl, String nextPageSearchUrl, SpotifyApi.Callback callback){
preferences = getStaticContext().getSharedPreferences("SpotifyPreferences", 0);
String accessToken = preferences.getString("accessToken", null);
JsonRequest jsArrRequest = new JsonObjectRequest(Request.Method.GET, newSearchUrl ,null,
```

```
    response -> {
        String nextPageToken = null;
        try {
            nextPageToken = response.getString("nextPageToken");
        } catch (JSONException e) {
            e.printStackTrace();
            callback.error(BlindResources.getString(R.string.app_spotify_network_not_available));
        }
        try {
            JSONObject tracks = response.getJSONObject("tracks");
            JSONArray items = tracks.getJSONArray("items");
            List<SpotifyTrack> searchResults = new ArrayList<>();
            for (int i = 0; i < items.length(); i++) {
                JSONObject id = items.getJSONObject(i);
                String trackName = id.getString("name");
                String trackUri = id.getString("uri");
                ....
                SpotifyTrack(trackName, trackUri, trackDuration, artist, album);
                searchResults.add(track);
            }
        callback.invoke(nextPageSearchUrl, searchResults, nextPageToken);
        } catch (JSONException e) {
            e.printStackTrace();
            callback.error(BlindResources.getString(R.string.app_spotify_network_not_available));
        }
        }, error -> {
            Log.e(TAG, "search: " + error.getMessage());
            callback.error(BlindResources.getString(R.string.app_spotify_network_not_available));
        }
        ){
            @Override
            public Map<String, String> getHeaders() throws AuthFailureError {
                Map<String, String> headers = new HashMap<>();
                headers.put("Accept", "application/json");
                headers.put("Content-Type", "application/json");
                headers.put("Authorization", "Bearer " + accessToken);
                return headers; }
        };
    RequestQueue rq = Volley.newRequestQueue(getStaticContext());
    rq.add(jsArrRequest);
}
```

Obrázek 10 Ukázka získaní písniček z API

Tímto způsobem se zpracovávají všechna data, která se z API získávají. Autorizační token který posílám v hlavičkách HTTP dotazů se nejprve musí získat autorizací. Autorizaci také částečně spravuje třída SpotifyApi.

4.2.3. Autorizace

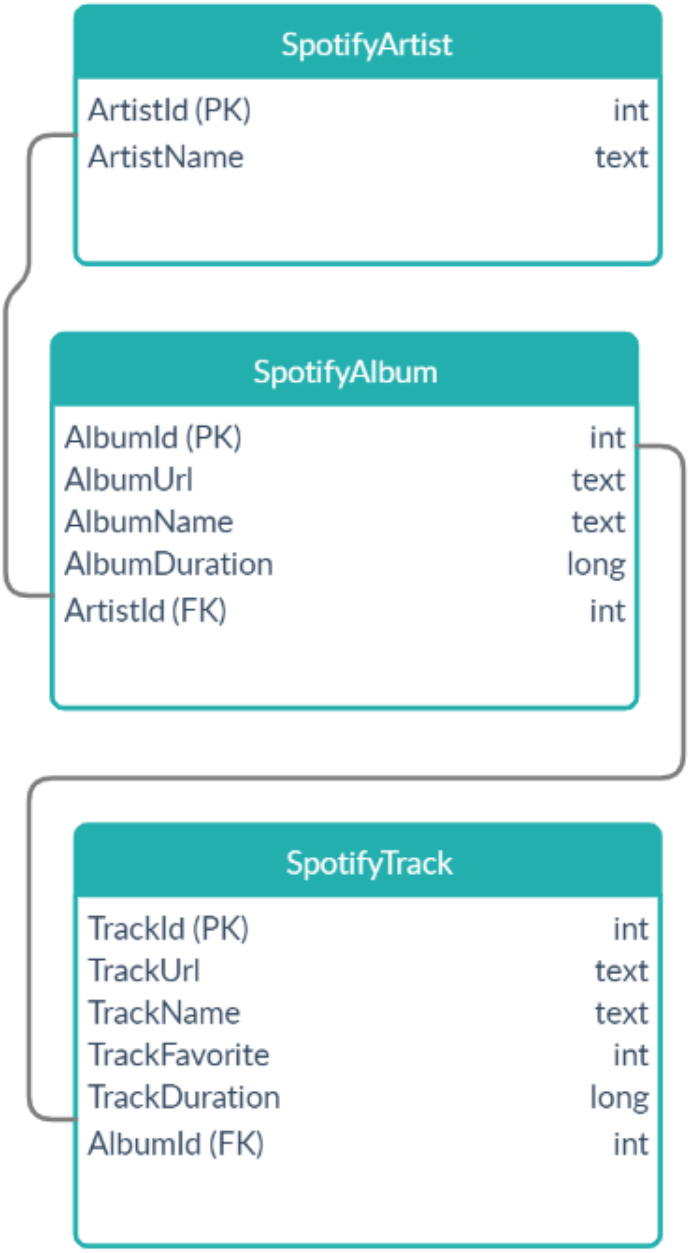
Autorizace u Spotify API je definováno OAuth 2.0 specifikací. Nejprve se musí uživatel přihlásit pomocí dalšího HTTP dotazu na API. Přihlašování je realizováno pomocí třídy *BlindSpotifyLogin* uživatel zadá přihlašovací údaje, heslo a email. Jelikož nevidomý lidé by se nemohli přihlašovat na webovém rozhraní, tak jsem vytvořit WebView z přihlašovací stránky a zmenšil, aby nebylo vidět. A poté jsem vytvořil vlastní UI v Spotify aplikaci, které se pak následně vloží do WebView a pošle. Aplikace si poté údaje uloží do SharedPreferences, poté se přes třídu SpotifyApi pošle HTTP dotaz na API. Dotaz je zpracován a zpět přijde autorizační a refresh token. Tyto token se také uloží do SharedPreferences. A uživatele to přesměruje na menu aplikace Spotify. Pokud je uživatel přihlásí, přihlášen zůstane až něž se sám odhlásí. A proto je tu refresh token. Pokud je uživatel již přihlášen mohla by nastat situace kdy autorizační token vyprší a Spotify API by už pak s aplikací nekomunikovalo. Refresh token se pošle jako HTTP dotaz, tento dotaz ještě s pár dalšími nemusí být autorizován, který pak navrátí nový autorizační token.

```
final String js = "javascript:" +
    "var auth = document.getElementById('auth-accept');" +
    "if(auth){ auth.click(); } else {" +
    "var username = document.getElementById('login-username');" +
    "var password = document.getElementById('login-password');" +
    "var event = new Event('input');" + "username.value='" +
    preferences.getString("spotifyUsername", null) + "';" +
    "username.dispatchEvent(event);" + "password.value='" +
    preferences.getString("spotifyPassword", null) + "';" +
    "password.dispatchEvent(event);" + "var login = document.getElementById('login-button');" +
    "login.click();" + "var auth = document.getElementById('auth-button');" +
    "}";
```

Obrázek 11 Ukázka předaní přihlašovacích údajů do WevView

4.3. Ukládaní dat

V této kapitole bych rád rozebral, jak jsem postupoval při implementaci ukládaní dat. Jak jsem již zmiňoval v analýze použil jsem databázi SQLite. Postupoval jsem podle databázového schématu.



Obrázek 12 - Databázové schéma

V první řade se databáze musí vytvořit k tomu slouží třída DBHelper. Tato třída dědí po *SQLiteOpenHelper*, která poté zajišťuje vytváření celkové databáze. Databáze je také obsahuje verze, které se postupně podle úprav navyšují. Obsahuje funkce na vytváření tabulek. Funkce *SQL_CREATE_ENTRIES_TRACK* vytvaří tabulku track podle předem definovaných parametrů. S tím souvisejí třídy *SpotifyTrack, SpotifyAblum, SpotifyArtist.* Tyto třídy reprezentují tabulky z databáze v Aplikaci Spotify. Všechny tyto třídy obsahují vnitřní třídu *Contract*, která obsahuje předem vytvořené parametry pro vytvoření tabulky.

```
private static final String SQL_CREATE_ENTRIES_TRACK = "CREATE TABLE " +
    SpotifyTrack.Contract.TABLE + " (" + SpotifyTrack.Contract.COL_TRACKID + " INTEGER primary
key
    AUTOINCREMENT," +
    SpotifyTrack.Contract.COL_TRACKURI + " TEXT UNIQUE," +
    SpotifyTrack.Contract.COL_TRACKNAME + " TEXT," +
    SpotifyTrack.Contract.COL_TRACKFAVORITE + " INTEGER DEFAULT 0," +
    SpotifyTrack.Contract.COL_ALBUMID + " INTEGER," +
    SpotifyTrack.Contract.COL_TRACKDURATION + " LONG," +
    "FOREIGN KEY ("+COL_ALBUMID+")
    REFERENCES "+SpotifyAlbum.Contract.TABLE+"("+SpotifyAlbum.Contract.COL_ALBUMID+"))";
private static final String SQL_DELETE_ENTRIES_TRACK = "DROP TABLE IF EXISTS " +
SpotifyTrack.Contract.TABLE;
```

Obrázek 13 - Ukázka vytvoření tabulky *track* v databázi

S databází dále pracuje třída SpotifyDb, která představuje komunikaci mezi databází a aplikací Spotify. Obsahuje funkce pro získání všech písniček, alb, artistu, ale také například *Favorites,* oblíbené písničky nebo pouze písničky z určitého alba a mnoho dalších. Všechny tyto funkce obsahují *Cursor,* pomocí kterého získávají data z databáze. *Cursor* je v androidu vestavěná třída, která je přímo vytvořená pro získávání dat z databáze. Funguje podobně jako array, je to struktura položek s ukazatelem, kdy ukazatel ukazuje před první položku. Proto se používá funkce *moveToFirst* pro získání prvního prvku.

```
public SpotifyTrack getFavoriteTrackFromTableBySpotifyUri(String table, String uri) {
	SpotifyTrack track = null;
	Cursor c = db.query(table,
		null,
		SpotifyTrack.Contract.COL_TRACKURI + "=? AND " +
		SpotifyTrack.Contract.COL_TRACKFAVORITE + "=?",
		new String[]{uri, "1"},
		null,
		null,
		null);
	if(c!= null && c.moveToFirst()) {
		track = new SpotifyTrack(c);
	}
	c.close();
	return track;
}
```

Obrázek 14 - Ukázka funkce pro získání určité oblíbené písničky

Pomocí těchto funkcí se získávají veškerá data pro vybudování struktury aplikace při vyhledávání písniček ve své kolekci a dále také pro přehrávaní dat. Pro zpětné vytvoření objektu s kterými muže aplikace Spotify pracovat, je potřeba z *Cursoru* získat data. K tomu slouží konstruktory ve třídách *SpotifyTrack, SpotifyAlbum, SpotifyArtist.* Tento konstruktor bere jako parametr právě daný *Cursor,* z kterého vybere data.

```
protected SpotifyTrack(Cursor c) {
	name = c.getString(c.getColumnIndex(Contract.COL_TRACKNAME));
	uri = c.getString(c.getColumnIndex(Contract.COL_TRACKURI));
	duration = c.getLong(c.getColumnIndex(Contract.COL_TRACKDURATION));
	album = SpotifyDb.getInstance().getAlbumFromTableBySpotifyId(
		SpotifyAlbum.Contract.TABLE,
		c.getString(c.getColumnIndex(Contract.COL_ALBUMID))
		);
	artist = album.getArtist();
}
```

Obrázek 15 - Ukázka funkce pro vytvoření objektu písničky z cursoru

4.4. Přehrávání

Přehrávání v aplikaci Spotify uskutečňuje třída BlindSpotifyPlayer, která využívá třídu *SpotifyPlayer*. *SpotifyPlayer* je třída, která dokáže přehrávat Spotify URI, využívá k tomu vestavěný androidí *Player. Player* slouží jako přehrávací medium v android zařízeních.

Poté co je písnička získána z databáze nebo z API, je dale poslána pomocí Intentu do třídy BlindSpotifyPlayer. Tato třída obstarává veškeré ovládání od, přehrávání, zobrazování času, přetáčení, pozastavování až po přidavání do databáze a do oblíbených. Přehrávaní je tu řešeno přímo třídou *SpotifyPlayer* kdy se jednoduše zavolá funkce *playUri* s určitou URI a třída spustí android *player* a začne přehrávání písničky.

Čas zobrazujícím polohu ve přehravaní písničce je tvořen pomocí *Runnable timer. Runnable* je blok kódu, který se opakuje na pozadí aplikace. Tento *Runnable* je pak spravován třídou *Handler*: jeho základní funkcionalitou je funkce *post,* která spustí prvotní *Runnable* block.

```
private Runnable timer = new Runnable() {
	public void run() {
		long currentDuration;
		if (player != null && player.getPlaybackState().isPlaying) {
			currentDuration = player.getPlaybackState().positionMs;
			updateTimer(currentDuration);
			timeView.postDelayed(this, 0);
		} else {
			timeView.removeCallbacks(this);
		}
	}
};
…
timeView.post(timer);
```

Obrázek 16 - Ukázka třídy *Runnable timer*

Uživatel si může přehrát naposledy přehrávanou písničku přes položku „Pokračovat v přehrávání“, která je umístěna v hlavním menu. Přehrát naposledy přehrávanou položku lze jen tehdy, když uživatel již přehrával nějakou písničku. Poslední přehrávaná stanice se nastavuje pokud člověk spustí určitou písničku a poté se přepisuje, jak se písničky posouvají nebo pokud uživatel nespustí přehrávání jiné písničky. Data o písničce se ukládají do metadat třídy *SpotifyPlayer* z kterých se pak jednoduše získávají pomocí funkce *getMetadata.*

5. Testování

Testování má 2 části, první část měla probíhat pomocí testerů z organizace SONS, které mělo přesně doupravit otestovat, jestli je ovládaní vhodné a jestli jsou zahrnuté veškeré potřebné požadavky. Tato část se bohužel musela zrušit s nenadálým příchodem nového druhu koronaviru COVID-19, která bohužel tuto část zcela zpomalila a tím pádem se musela odložit. Tento test je rozdělen na čtyři části: informace o participantovi, dotazník před testem, samotný test a dotazník po testu. Některé z těch částí byly upraveny, kvůli aktuální situaci. Při testování s nevidomými participanty je v místnosti také osoba, co pomáhá průběhu testu. Tato osoba pomáhá participantovi předčítat otázky a přitom si zaznamenává poznámky ohledně testu.

5.1. Informace o participantovi

Tato část má pomoci s výběrem vhodných participantu pro testování a také zasazení testovacích dat do kontextu participanta. Informace jsou získávány jednoduchým dotazníkem. Dotazník obsahuje např. věk, jak často využívá mobilní zařízení a další. Dotazník je součástí přílohy A.

5.2. Dotazník před testem

Dotazník před testem je už přímo spojený s tématem aplikace Spotify a mobilního telefonu a zjišťuje, co by participant očekával a jaké jsou jeho představy. Otázky v dotazníku jsou většinou otevřeného typu, aby se participant mohl lépe vyjádřit. Tento dotazník je také součástí přílohy A.

5.3. Test použitelnosti

Po „Dotazníku před testem“ se přejde samotný na test. Test má před připravené úkoly ohledně aplikace Spotify, které by měl participant podstoupit, pokud některou nezvládne pokračuje na další. Tato část je nejdůležitější pro osobu, co test podává, musí sledovat každý pohyb uživatele po aplikaci a zaznamenávat ho. Dále se požádá participanta, aby všechny kroky, co chce a bude dělat předem nahlas vyslovoval, i z těchto informací lze zjistit, jestli je vše na správném místě. Úkoly jsou následovné:

1) Najděte a spusťte aplikaci Spotify.
2) Přihlaste se pomocí svých přihlašovacích údajů do Spotify.
3) Vyhledejte svojí oblíbenou písničku a přehrajte jí.
4) Zkuste písničku přidat do kolekce.

5) Vraťte se do měnu a najděte vaší uloženou písničku v kolekci a přehrajte ji.
6) Přidejte si písničku do oblíbených.
7) Vraťte se do menu a najděte vaší písničku v oblíbených a přehrajte ji.
8) Při přehrávání zkuste přetáčení písničky.
9) Odstraňte vaší písničku z oblíbených.
10) Vraťte se do menu a odhlaste se.

5.4. Dotazník po testu

Po dokončení testu je participantovi položeno pár otázek v „Dotazníku po testu“, tyto otázky se zaměřují na průběh testu, a jestli měl participant s něčím problém nebo mu bylo nejasné. Složí tak k vhodnému doplnění poznámek vytvořených při provádějí úkolů. Tento dotazník je také součástí přílohy A.

6. Závěr

Práce popisuje projekt vytvoření mobilní aplikace Spotify pro nevidomé. V analýze jsem porovnával určité aplikace, které se také zaobírají přehráváním hudby, což vedlo k získání užitečných informací, které pak dále vedly k lepšímu specifikovaní požadavků a potřebám nevidomých uživatelů. V části implementace vysvětloval, jak je program vytvořený, z jakých částí se skládá a jak a proč jsem naprogramoval jaké části aplikace. Bakalářskou práci jsem stihl vyhotovit a otestovat, po prodloužení lhůty, kvůli novému výskytu nemoci COVID-19. Myslím si že moje práce (program) bude užitečný nevidomým lidem, kteří rádi poslouchají hudbu. A do teď si písničky museli do mobilních telefonu stahovat nebo poslouchat pouze přes aplikaci rádio, která již v mobilu je vytvořena. Také bych moc rád poděkoval mému vedoucímu panu Ing. Danielovi Novákovi Ph.D. a oponentu Ing. Janu Hadáčkovi. Kteří mi dávali zpětnou vazbu a doprovázeli mě při vytváření této bakalářské práce.

Příloha A

Dotazník

Dotazník, který byl použit při testování.

1.B Informace

Jméno:
Email:
Telefon:

1) Jaký je váš Věk:
2) Testoval jste již nějakou jinou aplikaci?
3) Pro jaké funkce využíváte svůj telefon?
4) Jak často využíváte zařízení rádio, které již v mobilním telefonu je?
5) Jak často posloucháte hudbu?
6) Víte, co je to Spotify?
7) Máte Spotify účet?

2.B Dotazník před testem

1) Používáte mobilní telefon rád/a?
2) Jaké na něm vnímáte plusy?
3) Jaké na něm vidíte mínusy?
4) Chtěl/a byste nějakou hudební aplikaci ve svém telefonu?
5) Co by taková aplikace měla umět?

3.B Dotazník po testu

1) Byly pro vás úkoly náročné?
2) Jaký úkol byl pro vás nejtěžší?
3) Je něco, co vám přišlo v aplikaci navíc?
4) Je něco, co vám na aplikaci chybělo?
5) Používal byste aplikaci, kdybyste chtěl poslouchat hudbu?

Příloha C

Data získaná z dotazníků

Tyto data byly získány od participantu (členů vlastní rodiny) z dotazníku a testu.

1.C Participant 1

1.1.C Informace

1. 41
2. Ano
3. Ano
4. 2krát týdně
5. Pořád
6. Ano
7. Ano

1.2.C Dotazník před testem

1. Ano
2. Jednoduché ovládání, pohotovost
3. Výdrž baterie, zadávání textu
4. Ano
5. Vyhledávání podle žánru, přidávat položky do oblíbených, off-line hudbu.

1.3.C Dotazník po testu

1. Ne
2. Vyhledávání, orientace
3. Collection
4. Žánr nebo předvolba podle doporučených
5. Ano youtube music

2.C Participant 2

2.1.C Informace

8. 50
9. Ano
10. I jiné
11. Denně
12. Denně
13. Ano
14. Ano

2.2.C Dotazník před testem

6. Ano
7. Jak je všechno pohromadě, převod textu na hlas
8. Malá baterie, psaní, práce s textem
9. Ano
10. Vyhledávání, třídění, nabídka již poslouchaných, playlisty

2.3.C Dotazník po testu

6. Ne
7. Collection
8. Ne
9. Žánr, seřazení podle posloupnosti (filter)
10. Ano

3.C Participant 3

3.1.C Informace

15. 33
16. Ano
17. I jiné
18. 3krát týdně
19. Denně
20. Ano
21. Ne

3.2.C Dotazník před testem

11. Ano
12. Spousta funkcí na jednom místě
13. Skladnější
14. Ano
15. Vyhledávání, ukládání do oblíbených

3.3.C Dotazník po testu

11. Ne
12. matoucí oblíbené a collection
13. Ne
14. Žánr, vyhledávání podle nějaké inspirace
15. Spíše youtube

Příloha D

Obsah přiloženého CD

Na CD nosiči se nachází kód aplikace a tento dokument.

Literatura

[1] Android. Android Interfaces and Architecture. [online]. [28. 04. 2016]. URL: https://source.android.com/devices/.

[2] Spotify.com. Spotify. [online]. URL: https://www.spotify.com/cz/.

[3] Developer spotify. Web API. [online]. URL: https://developer.spotify.com/documentation/web-api/.

[4] Google Play. Application information. [online]. [05. 05. 2016]. url: https://play.google.com/store.

[5] Developer Android. Android developers. [online]. [14. 05. 2016]. url: https://developer.android.com/.

[6] SONS (Sjednocená organizace nevidomých a slabozrakých ČR). SONS. [online]. [10. 05. 2016]. url: https://www.sons.cz/.

[7] Android. Author: Jackson, Wallace.Android Apps for Absolute Beginners. [Apress]. [2017]. Počet stran: 484.

Zdroje

Obrázek 1 Náhled do aplikace Spotify

https://www.lifewire.com/make-a-playlist-on-spotify-4138575

Obrázek 2 Menu a oblíbené v aplikaci Soundcloud, [05.01.2020]

https://thenextweb.com/wp-content/blogs.dir/1/files/2014/05/Soundcloud2.jpg

Obrázek 3 Náhled do aplikace iTunes

https://www.macobserver.com/cool-stuff-found/itunes-remote-app-update/

Obrázek 4 Náhled do aplikace Apple Music

https://appleinsider.com/articles/16/12/30/apple-music-ranks-ninth-in-2016-nielsen-charts-of-most-used-mobile-apps

Obrázek 5 **Uživatelem obnovující autorizace - (Refreshable user authorization)**

https://developer.spotify.com/documentation/general/guides/authorization-guide/