VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SPRÁVA METADÁT K HUDOBNÝM SÚBOROM

MUSIC FILES METADATA MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK SALOŇ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Saloň Marek**
Program: Informační technologie
Název: **Správa metadat k hudebním souborům**
Music Files Metadata Management
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte online služby, poskytující metadatata k hudebním souborům. Seznamte se s existujícími aplikacemi, která tato metadata využívají.
2. Stanovte požadavky na metadata pro potřeby DJ (např. BPM, styl, původní vydání skladby).
3. Navrhněte poloautomatické prostředky pro uživatelsky přívětivou správu metadat rozsáhlého hudebního archivu a pro přehrávání.
4. Navržené prostředky realizujte, ověřte jejich praktickou použitelnost a vyhodnoťte dosažené výsledky.

Literatura:

- ID3 Tags. URL: http://id3.org/
- MusicBrainz. URL: https://musicbrainz.org/
- 45cat. URL: http://www.45cat.com

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a část návrhu.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Hlavným cieľom tejto bakalárskej práce je navrhnúť a realizovať aplikáciu schopnú spravovať metadáta lokálnej hudobnej knižnice. Za týmto účelom boli implementované metódy, ktoré uľahčujú a automatizujú prácu s väčším množstvom súborov. Jedným z hlavných prostriedkov, ktoré umožňujú automatizáciu, je využitie internetových služieb k sprostredkovaniu relevantných informácií. Táto práca popisuje hudobné súbory a spôsob uchovávania metadát, štruktúru a využitie jednotlivých služieb, diskutuje o možných rozšíreniach a v neposlednom rade, vyhodnocuje výslednú realizáciu.

Abstract

The main goal of this bachelor thesis is to design and implement application that is able to process metadata of local music library. For this purpose, custom methods were implemented to ease and automate working with large number of files. One of the main means of automation is the use of internet services to obtain relevant information. This work describes the music files and methods of storing metadata, the structure and use of individual services, discusses possible extensions and, last but not least, evaluates the resulting implementation.

Kľúčové slová

hudobné súbory, súborový manažment, python, automatizácia, metadáta, tagy, databázové služby, získavanie informácií

Keywords

music files, file management, python, automation, metadata, tags, database services, retrieving information

Citácia

SALOŇ, Marek. *Správa metadát k hudobným súborom.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Vladimír Janoušek, Ph.D.

Správa metadát k hudobným súborom

Prehlásenie

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

.....
Marek Saloň
15. mája 2019

Poďakovanie

Veľmi rád by som sa poďakoval pánovi Doc. Ing. Vladimírovi Janouškovi, Ph.D za poskytnuté rady a nápady, ktoré som mohol využiť pri realizácií tejto práce.

Obsah

1 Úvod **3**

2 Reprezentácia zvuku v počítači **4**
2.1 Transformácia zvukového signálu do digitálnej podoby 4
2.1.1 Vzorkovanie 5
2.1.2 Kvantovanie 6
2.1.3 Kódovanie 7
2.2 Kompresia zvukových dát 8
2.3 Špecifikácia zvukových formátov 11
2.3.1 Multimediálny kontajner 11
2.3.2 Nekomprimované zvukové formáty 11
2.3.3 Bezstratovo komprimované zvukové formáty 12
2.3.4 Stratovo komprimované zvukové formáty 12

3 Metadáta v hudobných súboroch **15**
3.1 ID3 tagy 15
3.1.1 ID3v2.3 16
3.1.2 ID3v2.4 18
3.2 APE tagy 19
3.3 Vorbis comments 20
3.4 Metadáta pre dídžejing 20

4 Zhodnotenie súčasného stavu a návrh riešenia **23**
4.1 Služby poskytujúce prístup k metadátam 23
4.1.1 Editory tagov 23
4.1.2 Internetové hudobné databázy 24
4.2 Rozpoznávanie hudby odtlačkom zvukového signálu 25
4.2.1 Základný koncept 25
4.2.2 Vytváranie digitálneho odtlačku 26
4.2.3 Porovnávanie a hodnotenie 27
4.3 Návrh nástroja pre správu metadát 28

5 Realizácia a vyhodnotenie výsledkov **30**
5.1 Použité technológie 30
5.2 Implementácia 32
5.2.1 Prístup k metadátam 32
5.2.2 Identifikácia skladieb a sťahovanie metadát 34
5.2.3 Grafické užívateľské rozhranie 35

5.3 Vyhodnotenie dosiahnutých výsledkov 36
5.3.1 Predstavenie vytvoreného nástroja 36
5.3.2 Testovanie 38

6 Záver **41**

Literatúra **42**

Kapitola 1

Úvod

Hudobný priemysel prešiel za poslednú dobu veľkou premenou. Príchod výpočtovej techniky zmenil tvorbu, distribúciu a prehrávanie hudby prelomovým spôsobom. Počítače sa tak stali hlavnou doménou pre prácu s hudbou. Vznik hudobnej skladby však stále nie je jednoduchý proces a vyžaduje ohromnú dávku predstavivosti, trpezlivosti a talentu. Technologický pokrok však mnoho činností spojených s tvorbou a úpravou hudby zjednodušuje.

Moderné technológie poskytujú veľa možností ako sa dostať k skladbám svojho obľúbeného interpreta či umelca. V poslednej dobe sa spopularizovali služby, ktoré hudbu streamujú, teda vysielajú vďaka internetovému pripojeniu. Táto možnosť má mnoho výhod a uľahčuje rekreačné počúvanie bez nutnosti skladovania skladieb na konkrétnom zariadení. Problém nastáva v prípade, že internetové pripojenie nie je k dispozícií alebo úroveň kompresie prílišne degraduje výslednú kvalitu záznamu. Ďalšia nevýhoda spočíva v tom, že nie je umožnený prístup k zvukovým dátam, čo znemožňuje prípadnú manipuláciu s hudbou. V praxi každého dídžeja je potreba lokálnych hudobných súborov nevyhnutná.

Hudba je v počítači reprezentovaná vo forme hudobného súboru. Tieto súbory je vhodné klasifikovať a triediť podľa rôznych kritérií. Prevažná väčšina hudobný súborov má množinu vlastností a parametrov, ktorá ich špecifikuje. Pokiaľ užívateľ vlastní veľkú zbierku hudobných súborov, je vhodné tieto súbory vedieť rozumne skladovať. Prevažná väčšina zariadení určených na prehrávanie digitálnej hudby podporuje čítanie tzv. *metadát*, ktoré reprezentujú množinu podrobností o jednotlivých hudobných súboroch a skladbách, ktoré tieto súbory obsahujú. Medzi typické informácie obsiahnuté v metadátach patria údaje o interpretovi, názve skladby, albume, roku vydania a mnoho ďalších. Z pohľadu obyčajného človeka, ale aj profesionálneho dídžeja, je praktické mať hudobné súbory správne identifikované, čo prináša radu výhod pri filtrovaní a samotnom prehrávaní hudby.

Táto práca sa zaoberá spôsobmi, ktoré umožňujú spomínaný cieľ dosiahnuť. Kapitola 2 sa venuje základom spracovania hudby v počítači, kde zároveň popisuje a porovnáva jednotlivé zvukové reprezentácie. V kapitole 3 sú predstavené možnosti, ktoré umožňujú ukladať, spravovať a používať metadáta, napríklad aj v praxi dídžeja. Kapitola 4 popisuje návrh aplikácie a využité koncepty. Implementácia, testovanie a vyhodnotenie výslednej práce sa nachádza v kapitole 5. Hlavný dôraz sa kladie na jednoduchosť a efektívnosť z pohľadu koncového používateľa. Na záver sa diskutuje o možných zlepšeniach a plánovanom budúcom vývoji.

Kapitola 2

Reprezentácia zvuku v počítači

Táto kapitola popisuje základné koncepty reprezentácie zvuku v digitálnej podobe. Jej hlavnou podstatou je oboznámiť čitateľa s procesom digitalizácie za účelom pochopenia širších súvislostí. Sekcia 2.2 sa venuje problematike kompresie zvukových dát a diskutuje o prípadných negatívnych dôsledkoch tohto procesu. V sekcií 2.3 sú predstavené populárne zvukové formáty, ktoré sa používajú na kódovanie zvukových súborov a digitálnej hudby.

2.1 Transformácia zvukového signálu do digitálnej podoby

Hudba je organizovaný systém zvukov, ktorý má rytmický charakter. Z fyzikálneho hľadiska sa však jedná o zvuk, teda mechanické vlnenie. Frekvencie tohto vlnenia sú v určitom pásme počuteľné aj pre ľudské ucho. Hudobné nástroje využívajú rôzne techniky za účelom vytvorenia vlnenia, ktoré sa nazýva tón alebo hluk. V prípade tónu sa jedná o časovo periodické kmitanie, ktoré vytvára vnem zvuku určitej výšky. Pre vyjadrenie zvuku v digitálnom prostredí je nutné previesť toto vlnenie na číselné hodnoty.

Výhody digitálneho záznamu zvuku[13]:

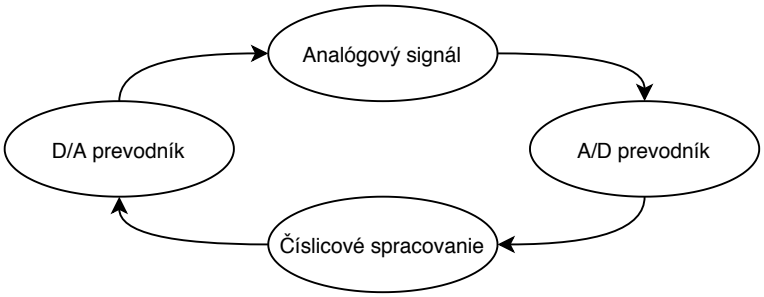
- Kvalita záznamu časom nedegraduje.
- Možnosti úpravy, kopírovania a jednoduchého prenosu.
- Zvuk na digitálnych nosičoch je skladnejší.
- Široká podpora digitálnych formátov.

Záznam zvuku začína nahrávacím zariadením, najčastejšie mikrofónom, ktoré premieňa zvukové vlnenie na elektrický prúd. Tento prúd je potrebné transformovať na digitálny signál. Proces prevodu analógového (spojitého) signálu na digitálny (diskrétny) signál sa nazýva *digitalizácia*. Metóda digitalizácie zvuku sa označuje ako *pulzná kódová modulácia*, skrátene PCM (Pulse Code Modulation). Proces digitalizácie definuje základné parametre výsledného súboru, ktoré vplývajú na jeho kvalitu a veľkosť. Pri reprodukcií zvuku sa z číselných údajov opäť vytvára analógový signál, ktorý je možné opakovane prehrávať pomocou príslušných zariadení. Proces obojsmernej transformácie je vyobrazený na obrázku 2.1.

Zariadenie alebo program, ktorý vykonáva potrebnú transformáciu, sa nazýva *kodek*. Výraz kodek vznikol spojením názvov dvoch primárnych častí, *kodér* a *dekodér*.

V prípade fyzického kodeku sa jedná o elektrické obvody určené na kódovanie a dekódovanie digitálneho signálu. Kodér, často označovaný ako *analógovo-digitálny prevodník (ADC)*, je elektronická komponenta určená na prevod analógového signálu na digitálny signál. Naopak dekodér, teda *digitálno-analógový prevodník (DAC)*, je elektronická komponenta určená na prevod digitálneho signálu na analógový signál. Kvalita samotného kodéra, respektíve dekodéra, výrazne vplýva na výslednú kvalitu digitálneho signálu, respektíve kvalitu rekonštruovaného analógového signálu pri výstupe.

Programový kodek je algoritmus založený na matematickom základe, ktorého úlohou je kódovať alebo dekódovať dátový tok, alebo signál. Rozličné kodeky tvoria podstatnú súčasť všetkých softwarových prehrávačov. Bez ich podpory by prehrávač nebol schopný dekódovať a následne prehrávať multimediálne dáta. Často je nevyhnutné previesť digitálny obsah z jedného kódovania do iného kvôli šetreniu úložného priestoru. Tento proces úzko súvisí s pojmom *kompresia*, ktorý je detailnejšie popísaný v sekcií 2.2.



Obr. 2.1: Proces prevodu analógového signálu na digitálny signál a naopak.

Digitalizáciu je možné rozdeliť na tri fázy, ktoré sa vykonávajú v chronologickom poradí:

1. Vzorkovanie
2. Kvantovanie
3. Kódovanie

2.1.1 Vzorkovanie

Vzorkovanie je prvá fáza digitalizácie. Hlavnou charakteristikou spojitých signálov je ich spojitosť, ktorá znemožňuje číslicovým počítačom s obmedzenou pamäťou uložiť kompletný signál v jeho originálnej podobe. Za týmto účelom sa využíva technika, ktorá umožňuje rekonštruovať spojitý signál do podoby konečného počtu diskrétnych vzoriek.

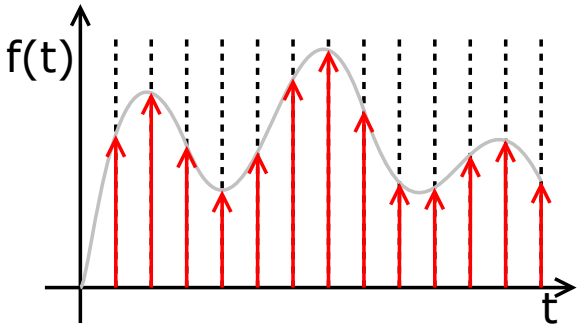
Z princípu je jasné, že dochádza k nevyhnutnej strate informácie, ktorú je možné regulovať pomocou *vzorkovacej frekvencie*. Tento údaj udáva počet zaznamenaných hodnôt za jednu sekundu, čím ovplyvňuje kvalitu a objem výsledného záznamu. Vizualizácia procesu vzorkovania analógového signálu vyjadreného funkciou f v čase t je zobrazená na obrázku 2.2. Jednotlivé vzorky sú tvorené hodnotami v bodoch daných vzorkovacou frekvenciou.

Shannon-Kotelnikov teorém hovorí o tom, že vhodná vzorkovacia frekvencia je dvojnásobne väčšia (vrátane malej rezervy) než je maximálna požadovaná frekvencia. Vzťah je vyjadrený rovnicou

$$f \quad = \quad 2f_{max}, \tag{2.1}$$

kde f je vzorkovacia frekvencia a f_{max} je maximálna frekvencia.

Kvalitný hudobný záznam má preto odporúčanú minimálnu vzorkovaciu frekvenciu na úrovni 44.1 kHz. Táto hodnota sa odvíja od faktu, že priemerný zdravý človek je schopný počuť frekvenčné pásmo v rozmedzí 20 Hz až 20 kHz.



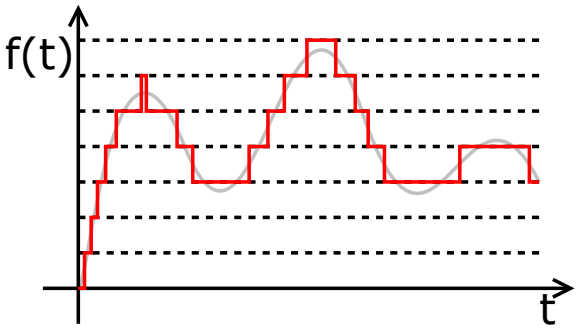
Obr. 2.2: Vzorkovanie vstupného analógového signálu. Prevzaté z [15].

2.1.2 Kvantovanie

Výstupom vzorkovania je konečný počet diskrétnych hodnôt, ktoré však stále obsahujú prebytok informácie. V tomto prípade je nutné získané hodnoty zaokrúhliť podľa predom určených úrovní. Tento proces sa nazýva *kvantovanie*.

Hlavným atribútom kvantovania je *bitová hĺbka*, ktorá vyjadruje počet kvantizačných hladín. Pre dosiahnutie uspokojujúcej kvality záznamu bez skreslenia je nutné túto hodnotu nastaviť dostatočne vysoko, aby ju amplitúda vstupných vzoriek nikdy nepresiahla. V hudobných záznamoch sa najčastejšie používa 2^{16} rôznych hodnôt alebo 16-bitová hĺbka.[11]

Graf na obrázku 2.3 ilustruje spôsob kvantovania. Jedná sa o rovnaký signál ako v predchádzajúcom prípade, avšak tentokrát sa rozdeľuje os funkčných hodnôt na úseky, ktorých počet udáva bitová hĺbka. Pomocou rozhodovacích metód sa určí akej hladine odpovedá signál v konkrétnom čase. Najčastejšie sú lineárne metódy založené na rozhodovacích úrovniach. Úrovne sa nachádzajú v polovičných vzdialenostiach medzi hladinami, čím presne vymedzujú hranice jednotlivých hladín.

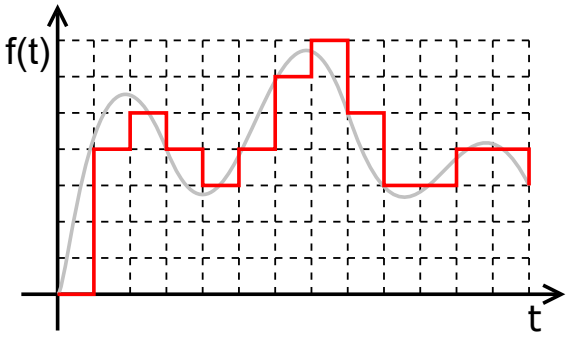


Obr. 2.3: Kvantovanie vstupného analógového signálu (bez vzorkovania). Prevzaté z [15].

2.1.3 Kódovanie

Cieľom poslednej fázy digitalizácie je jednotlivým hladinám kvantovaného signálu priradiť konkrétne binárne číslo. Tento proces nám umožní reprezentovať analógový signál pomocou binárneho kódu. Hladiny kvantovaného signálu, ktoré sú definované v desiatkovej sústave, stačí previesť do binárnej sústavy. Každá vzorka vstupného signálu je zakódovaná b-bitovým binárnym číslom, kde b je počet kvantizačných hladín. Toto číslo sa potom označuje ako *kódové slovo*. Tento „surový“ kód však spravidla nie je vhodný, preto ho je nutné prekódovať. Za týmto účelom sa využíva programový kodek.

Obrázok 2.4 znázorňuje podobu výsledného digitálneho signálu. Hlavnou myšlienkou je vyjadriť presnú hodnotu vzorky v diskrétnom čase pomocou binárnej hodnoty z rozsahu bitovej hĺbky. Z grafu je zrejmé, že výsledný signál je pozmenený. Táto strata informácie je však nevyhnutná a z časti akceptovateľná. Preto platí, že čím vyššia je vzorkovacia frekvencia a bitová hĺbka, tým viac sa výsledný digitálny signál podobá tomu pôvodnému.



Obr. 2.4: Výsledný tvar signálu po procese vzorkovania a kvantovania. Prevzaté z [15].

Medzi podstatné pojmy opisujúce výsledný signál patrí aj *bitový tok* (bitová rýchlosť). Bitový tok udáva počet prenesených bitov za sekundu a je definovaný ako súčin vzorkovacej frekvencie a dĺžky bitového slova. Pri zvukových súboroch je najčastejšie vyjadrená v *kilobitoch za sekundu (kbps)*. Bitový tok je možné vypočítať vynásobením vzorkovacej frekvencie, bitovej hĺbky a počtom zvukových kanálov.

Zvukový kanál je jeden dátový tok zvukového záznamu. Človek vníma zvuk dvoma receptormi, ktoré mu umožňujú určiť pozíciu zdroja zvuku v priestore. Z tohto dôvodu sa využíva *viackanálový zvuk*, ktorý je pre človeka prirodzenejší. Pojem *stereo* zvuku popisuje dva rozličné kanály, ktoré sú najčastejšie rozdelené na *pravý (R)* a *ľavý (L)* kanál.

Jednotlivé zvukové formáty sú definované hlavne z pohľadu miery kompresie a bitového toku. *Konštantný bitový tok (CBR)* znamená, že zvuk je zakódovaný stálym počtom bitov. Bitový tok však nemusí byť vždy konštantný. Niektoré formáty podporujú taktiež *variabilný bitový tok (VBR)*, ktorý umožňuje flexibilne meniť veľkosť kódovaných dát za jednotku času. Tento prístup je postavený na jednoduchom princípe. Čím je informácia v danom momente komplexnejšia, tým viac bitov sa použije na jej zakódovanie. Hlavnou výhodou variabilného dátového toku je lepší pomer veľkosti a kvality zvukového záznamu. Medzi hlavné nevýhody patrí vyššia časová zložitosť procesu kódovania a nižšia podpora hlavne u starších zariadení. Na základe týchto skutočností vznikol pojem *priemerný bitový tok (ABR)*, ktorý je vyjadrený vzťahom:

$$x_{ABR} = \frac{\sum_{i=0}^{t} m_i}{t}, \tag{2.2}$$

kde x_{ABR} je priemerný bitový tok, t je dĺžka zvukového záznamu v sekundách a m je veľkosť kódovaných dát v bitoch. Celkovú veľkosť súboru je možné vypočítať vynásobením priemerného bitového toku a dĺžky trvania záznamu.

Na prvý pohľad sa môže zdať, že bitový tok určuje kvalitu zvukového záznamu. To však vždy neplatí. V prípade nekvalitného prevodníka, algoritmu alebo zdrojového signálu môže dochádzať k javom, ktoré môžu výsledný signál ovplyvniť (skresliť) aj pri vysokom bitovom toku. Odpovede na túto tému budú popísané v nasledujúcej sekcií.

2.2 Kompresia zvukových dát

Hlavnou úlohou *kompresie* je minimalizovať množstvo dát, ktoré slúžia na reprezentáciu určitého digitálneho obsahu. Za týmto účelom sa využívajú rôzne algoritmy, ktoré tento cieľ dosahujú znižovaním redundancie dát alebo odstránením určitého množstva nepotrebnej informácie. Pri tejto činnosti sa kladie dôraz aj na časovú náročnosť. Kompresia môže byť *stratová* alebo *bezstratová*.

Podstatným parametrom je *kompresný pomer*, ktorý je vyjadrený ako pomer veľkosti pôvodných dát a skomprimovaných dát. Na základe tohto parametru vieme radiť kompresné algoritmy podľa ich efektívnosti.

Bezstratová kompresia sa riadi triedou algoritmov[3], ktoré dokážu spätne rekonštruovať komprimované dáta do pôvodnej podoby. Hlavným aspektom tejto metódy je, že žiadne dáta sa pri procese kompresie nestratia. To má za následok nižšiu efektivitu z pohľadu kompresného pomeru.

Algoritmy *stratovej kompresie* naopak počítajú s určitým stupňom degradácie. Vo väčšine prípadov sa jedná o zanedbateľnú informáciu, ktorá minimálne ovplyvní výslednú kvalitu komprimovaných dát. Tieto algoritmy sú z hľadiska kompresného pomeru veľmi efektívne, niekedy dosahujú aj hranice 90%.

Všeobecný princíp stratovej kompresie je jednoduchý. Na úvod je nutné dáta transformovať do formy, z ktorej je následne jednoduché vybrať to najdôležitejšie. Nepodstatné informácie sú potláčané oveľa výraznejšie ako tie dôležité. Nakoniec sa výsledok skomprimuje niektorým z kompresných algoritmov. Algoritmus stratovej kompresie má teda dve podstatné časti – *transformácia pôvodných dát* a *potlačenie menej dôležitých dát*.

- K transformácii pôvodných dát sa zvyčajne používajú metódy, medzi ktoré patrí aj algoritmus *Rýchlej Fourierovej transformácie (FFT)*. Tieto metódy prevedú pôvodné dáta do iných domén, napríklad z časovej do frekvenčnej, čo má za následok zníženie priestorových nárokov dôležitých informácií. Ak zvyšok dát nahradíme nejakými vopred známymi alebo vypočítateľnými dátami, výsledný signál bude veľmi podobný originálu. Pri tejto transformácii ešte nemusí dochádzať k degradácii pôvodných dát.

- Potláčanie menej dôležitých informácií je založené na psychoakustickom modely, ktorý určuje, aké informácie môžu byť potláčané alebo dokonca úplne odstránené. Napríklad pri kompresií zvuku sa hľadajú frekvencie, ktoré človek nemôže vnímať.

Na obrázkoch 2.5 a 2.6 je možné vidieť priame porovnanie a dôsledky kompresie. Vizualizácia je tvorená *spektogramom*, ktorý na vertikálnej osi vyjadruje frekvenciu (Hz) a na horizontálnej osi čas. Sfarbenie jednotlivých segmentov (pixelov) obrázka vyjadruje to, aký hlasitý (dB) je tón daného segmentu, teda amplitúdu tónu, ktorý reprezentuje. Táto reprezentácia je častá v prípadoch, kedy sa vstupný zvukový signál skúma detailnejším spôsobom. Obrázky boli vytvorené voľne dostupným programom Spek[1].

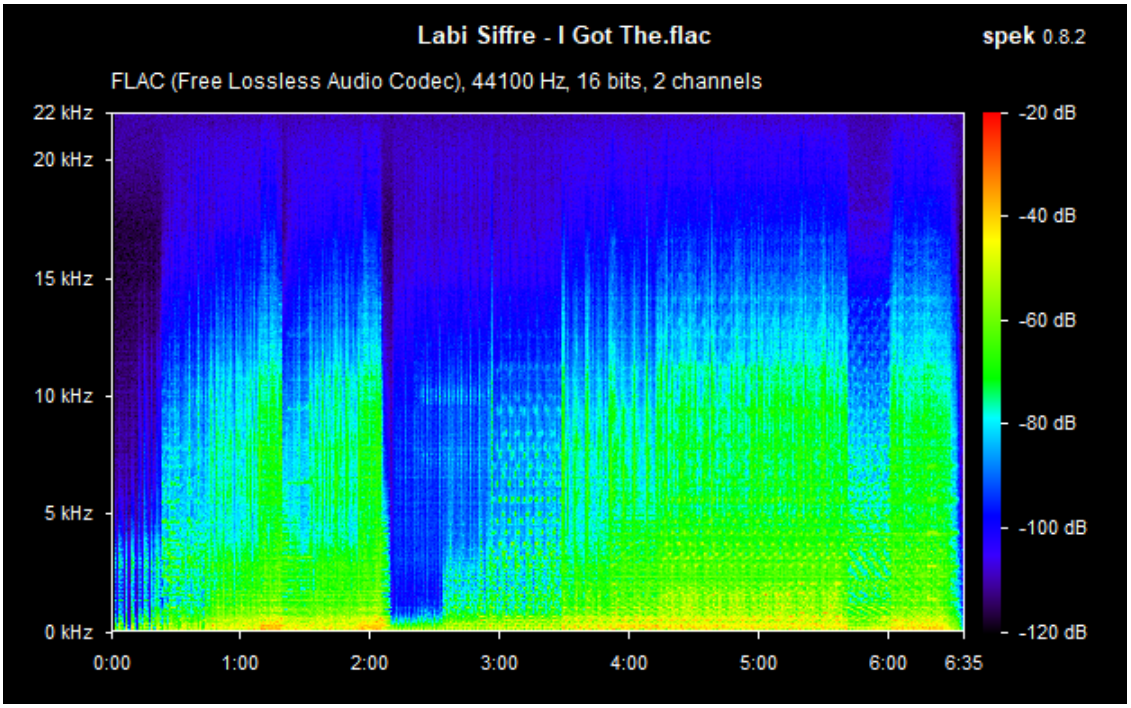
Obrázok 2.5 znázorňuje spektrum skladby, ktorá je v bezstratovom formáte FLAC. Z údajov, ktoré je možné jednoduchým pohľadom vyčítať je zrejmé, že zvukový signál pokrýva celé frekvenčné spektrum, teda frekvencie až do 22 kHz. Na druhej strane, prevažná väčšina tónov nad 15 kHz je veľmi nízkej hlasitosti, ktorá je pri tak vysokých frekvenciách takmer nepočuteľná. Ľudské ucho vekom stráca svoju pružnosť, čo znamená, že jeho schopnosť kmitať vysokým tempom sa časom minimalizuje. Tejto skutočnosti využíva stratová kompresia, ktorej výsledok je zobrazený na obrázku 2.6. V tomto prípade sa jedná o formát MP3, ktorý využíva viacero techník pre minimalizáciu finálnej veľkosti.

Prvý podstatný rozdiel je orezanie frekvencií na úrovni zhruba 16 kHz v každom časovom okamihu. To, od ktorej frekvencie sa informácia odstráni určuje algoritmus tak, aby sa pri kódovaní zmestil do vyžadovaného bitového toku. Táto hranica sa často označuje výrazom *prah*.

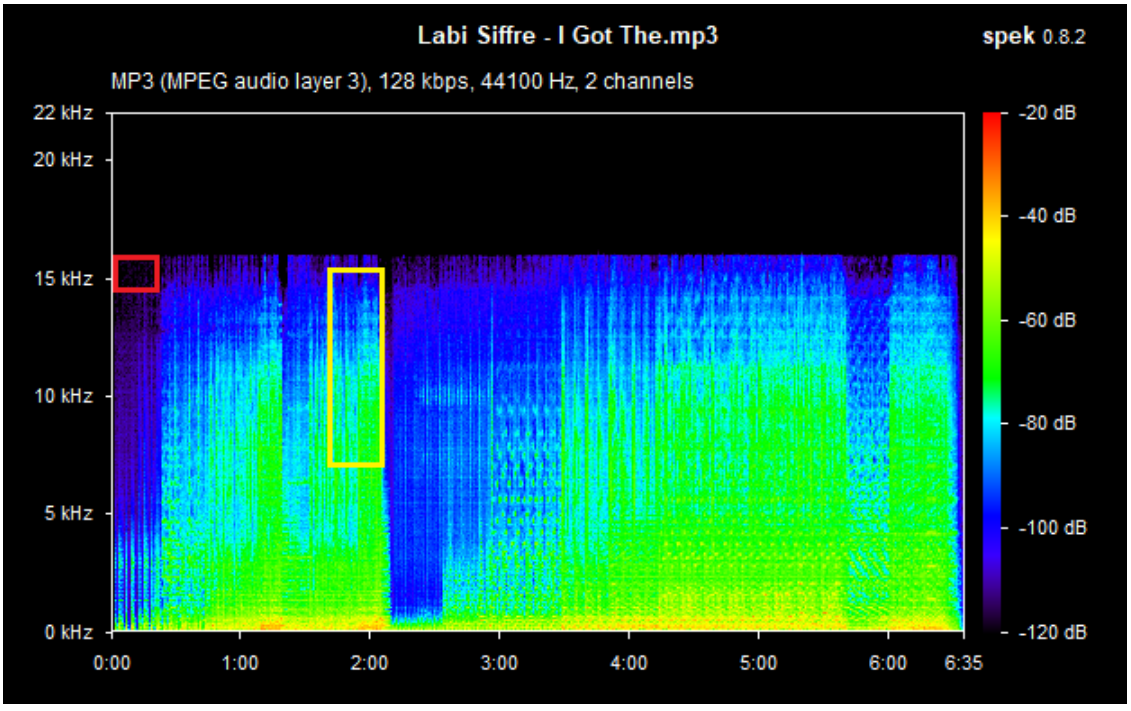
Nasledujúcim podstatným rozdielom je odstránenie niektorých frekvencií aj pod 16 kHz hranicou. Tento jav je na obrázku 2.6 vyznačený červeným štvorcom. Väčšinou sa jedná o frekvencie, ktoré majú v danom čase tak nízku hlasitosť, že po ich odstránení výsledný signál nestratí podstatnú informáciu. Podstatnou informáciou sa rozumejú tie najvýraznejšie tóny v danom okamihu, ktoré spôsobujú to, že vypustená informácia je pre drvivú väčšinu populácie nepočuteľná.

Poslednou viditeľnou zmenou je degradácia kvality signálu a vznik tzv. artefaktov, ako je možné vidieť napríklad vo vyznačenej časti obrázka 2.6. Signál je už na pohľad o niečo „hranatejší“ a o niečo menej podobný svojmu originálu. Počuteľnosť týchto deformácií je však v tejto situácií ešte prípustná a neskúsenému poslucháčovi môže úplne uniknúť. Tento jav je často spojený s *konvertovaním*, čo je proces, ktorý transformuje dáta jedného formátu na iný. Pri zvukových formátoch sa k tomu využívajú potrebné kodeky. V prípade výberu príliš nízkeho bitového toku je riziko výskytu artefaktov väčšie. Artefakty môžu byť spôsobené aj zlým výpočtom z pohľadu kódujúceho algoritmu.[9]

[1] www.spek.cc



Obr. 2.5: Spektogram na obrázku vyjadruje rozloženie a intenzitu frekvencií v závislosti na čase. V tomto prípade sa jedná o bezstratový signál, ktorý bol vytvorení kodekom FLAC.



Obr. 2.6: Spektogram rovnakej skladby vo formáte MP3, ktorý bol kódovaný kodekom LAME.

Tradičným využitím konvertovania je tzv. *downscaling*, ktorý transformuje priestorovo náročné dáta vysokého rozlíšenia na dáta skladnejšieho charakteru v nižšom výslednom rozlíšení. Význam tohto výrazu sa úzko prelína s výrazom *kompresia*.

Problém z pohľadu kvality nastáva pri tzv. *upscalingu*, ktorý slúži na presný opak. Algoritmy, spojené s týmto procesom, majú za úlohu dopočítať chýbajúcu informáciu na základe rozličných heuristík. Primárny cieľ je zvýšiť celkovú kvalitu. Avšak pri hudobných súboroch nastáva situácia, kedy tento proces dokáže kvalitu dokonca zhoršiť. Ako príklad je možné uviesť pokus konvertovať zvukový súbor stratového formátu pri bitovom toku 128 kbit/s na výsledných 320 kbit/s. Tento prípad vedie k pravdepodobnej degradácií, ktorá vznikne pri kódovaní, pričom výsledné rozlíšenie sa nezmení. Výsledný zvukový súbor bude mať väčšiu veľkosť, pričom dáta sa zduplikujú, čo vo výsledku zreplikuje signál spolu s artefaktmi. Tento príklad poukazuje na fakt, že zvukový formát a bitový tok nie sú jediné parametre pri posudzovaní kvality digitálnej hudby. Vykreslením spektogramu je možné prípadné nekvalitné či klamlivé zvukové súbory jednoducho identifikovať.

2.3 Špecifikácia zvukových formátov

2.3.1 Multimediálny kontajner

Reprezentácia multimediálneho obsahu v digitálnej podobe viedla k aktívnemu využívaniu špeciálneho typu súboru, ktorý sa nazýva *kontajner*. Špecifickou vlastnosťou kontajnera je možnosť uloženia rôznych typov dát do jedného počítačového súboru. Táto vlastnosť značne zjednodušuje manipuláciu s príslušnými dátami a ich popisom. Hlavnými zložkami multimediálneho kontajnera sú samotné *dáta* (zvuk, video) a *metadáta* (informačný popis dát).

Formáty kontajnerov sa líšia vo viacerých vlastnostiach, napríklad v schopnosti uchovania rôzneho typu multimediálneho obsahu. Niektoré kontajnery slúžia na uloženie zvukovej stopy, iné zas výhradne na video stopy, avšak existujú aj univerzálne kontajnery.

Je však nevyhnutné podotknúť, že kontajner nedefinuje kompresiu dát ani ich konkrétne kódovanie. Tieto vlastnosti sú dané použitým kodekom pričom kontajner tvorí len akúsi virtuálnu obálku.

2.3.2 Nekomprimované zvukové formáty

WAV

Waveform audio file format (WAV) je zvukový formát vyvinutý spoločnosťou IBM a Microsoft. Jeho podstatu tvorí natívny súborový formát, ktorý ukladá nekomprimované hudobné dáta. Z hľadiska objemu je to najnáročnejší zvukový formát vôbec. Tento formát našiel využitie v štúdiovom prostredí, alebo ako základný formát pred konverziou do ďalších formátov. WAV je varianta všeobecného multimediálneho kontajnerového formátu *RIFF*. Tento formát pozostáva z viacerých *blokov*, pričom metadáta je možné uložiť v bloku s názvom INFO. Súbor WAV navyše môže obsahovať aj metadáta typu XMP alebo ID3.

AIFF

Audio Interchange File Format (AIFF) je zvukový formát vyvinutý firmou Apple, pričom jeho základ tvorí kontajnerový formát *IFF*. Štandardne AIFF obsahuje hudobné dáta v nekomprimovanej PCM podobe, ale existuje i varianta *AIFF-C* podporujúca kompresiu.

AIFF má v štruktúre vyhradené bloky na metadáta základného typu, pričom je možné pridať bloky XMP alebo ID3v2 tagov.

2.3.3 Bezstratovo komprimované zvukové formáty

FLAC

Free Lossless Audio Codec (FLAC) je v dnešnej dobe asi najobľúbenejší bezstratový zvukový formát, ktorý spadá pod licenciu GNU GPL a BSD. Tento bezstratový formát zaisťuje kvalitu na úrovni WAV, pri redukcií svojej veľkosti až okolo 60%. V porovnaní s ostatnými bezstratovými formátmi, FLAC ako taký, umožňuje rýchle dekódovanie a streamovanie nezávislé na úrovni kompresie. Ďalšia výhoda je podpora tagov, ktoré využívajú systém podobný *Vorbis comments*. Kodek a kontajner nesú rovnaký názov ako formát samotný. FLAC dokáže kódovať signál vo vysokom rozlíšení (vzorkovacia frekvencia vyššia než 48 kHz a bitová hĺbka vyššia než 16 bitov) a kóduje i viackanálový zvuk (viac než 2 kanály). Typickým použitím je aj záloha CD Audio disku vďaka možnosti generovania *CUE súboru*, ktorý obsahuje CD text a rozloženie stôp.

ALAC

Apple Lossless Audio Codec (ALAC) je bezstratový formát spoločnosti Apple. Používa kontajner typu MP4. Využívaný kodek sa nazýva *Apple Lossless Encoder (ALE)*, ktorý je schopný znížiť veľkosť záznamu až na polovicu. Výsledný záznam podporuje až 8 zvukových kanálov a vysoké rozlíšenie (do 384 kHz vzorkovacej frekvencie a 32-bitovej hĺbky).

APE

Monkey's Audio (APE) je ďalší z rady bezstratových kompresných formátov. Výhodou tohto formátu je lepší kompresný pomer oproti FLAC a ALAC. Na druhú stranu využíva neefektívny symetrický algoritmus, ktorý je náročnejší na dekódovanie. Metadáta sú uložené v podobe APE tagov, ktoré boli pôvodne vyvinuté konkrétne pre tento formát.

2.3.4 Stratovo komprimované zvukové formáty

MP3

MPEG-1 Audio Layer III (MP3) je patentovaný, digitálny audio formát, využívajúci stratovú kompresiu. Pri kompresií vypúšťa nepodstatné dáta pomocou technológie *frekvenčného a časového maskovania*.

Frekvenční maskovanie využíva nedokonalosti ľudského ucha. Pokiaľ zaznejú dva tóny zvuku súčasne, môže jeden z nich potlačiť počuteľnosť toho druhého. Ak napríklad zaznie tón o frekvencii 5000 Hz a súčasne s ním slabší tón s frekvenciou 6000 Hz, nebude ľudské ucho schopné druhý tón rozpoznať.

Časové maskovanie sa uplatňuje pred a po silnom zaznení zvuku. Využíva sa toho, že ucho nepočuje niekoľko milisekúnd po zaznení výrazného tónu. Také maskovanie sa nazýva *postmasking*. Zaujímavé je, že vďaka zložitému vnímaniu ľudského mozgu, ucho nie je schopné počuť ani pár milisekúnd pred zaznením výrazného tónu. Tomuto javu sa hovorí *premasking*[10].

Pri vlastnom kódovaní je možné stanoviť hodnotu bitového toku, ktorou bude vstupný signál kódovaný. Medzi podporované dátové toky patria hodnoty od 32 kbit/s do 320 kbit/s

a vzorkovacie frekvencie 32, 44,1 a 48 kHz. Najlepších výsledkov dosahuje kodek pri použití bitového toku okolo 160 kbit/s a vzorkovacej frekvencií 44,1 kHz. Pomer veľkosti originálnej nahrávky ku veľkosti vzniknutej MP3 je v tomto prípade 9:1. Formát MP3 podporuje CBR, VBR a ABR bitový tok. Pod skratkou ABR sa rozumie variabilný bitový tok, ktorý sa pohybuje v okolí priemerného bitového toku.

Veľkosť výslednej kompresie je možné ovplyvniť počtom kanálov a ich kvalitou:

- *Mono* – Jednokanálový úsporný zvuk, pri ktorom hudobné súbory dosahujú nízku kvalitu.
- *Stereo* – Dvojkanálový zvuk, ktorý rozdeluje šírku bitového toku medzi dva kanály.
- *Joint stereo* – Najpoužívanejší mód. Tento mód kódovania využíva faktu, že v oboch kanáloch sa v danom momente vyskytuje rovnaká informácia. Ľavý a pravý kanál sa prepočíta na stredový a stranový. Viac bitového toku je pridelené stredovému kanálu, ktorý obsahuje dôležitejšiu informáciu. Pri prehrávaní sú obsahy ľavého a pravého kanála spätne dopočítavané.

Medzi najpopulárnejšie MP3 kodeky patrí open-source kodek *LAME*, ktorý má hlavné prednosti v kvalite a rýchlosti kódovania. Jeho implementácia je vo všetkých komerčných i nekomerčných hudobných programoch. Popularita formátu MP3 stála za vytvorením ID3 tagov, ktoré využíva aj v súčasnosti. Alternatívne umožňuje použitie APEv2 tagov.

AAC

Advanced Audio Coding (AAC) je stratový formát vyvíjaný ako nástupca formátu MP3. Podporuje až 48 zvukových kanálov s vzorkovacou frekvenciou do 96 kHz a použiteľnou bitovou hĺbkou v rozmedzí 8 až 24 bitov. Využíva pokročilé kompresné metódy, ktoré zaručujú vynikajúce výsledky hlavne pri nízkych dátových tokoch. Súbory uložené štandardom MPEG-4 majú prípony `.mp4`, `.m4a` a alebo `.m4p`. Súbory uložené podľa špecifikácie MPEG-2 používajú príponu `.aac`. Medzi najpoužívanejšie kodeky tohto formátu patrí *FFmpeg* a *FAAC*. Formát má definovanú vlastnú štruktúru metadát, ku ktorým je možné navyše pridať metadáta formátu XMP.

OGG

Vorbis audio (OGG), nazývaný rovnomerne podľa samotného kodeku, je open-source projekt nadácie Xiph.org. Jeho komprimačné algoritmy a kvalita zvuku je lepšia než u MP3. Nevýhodou je, že Vorbis nemá žiadnu ochranu proti chybám pri samotnom kódovaní. V dnešnej dobe si tento formát našiel využitie hlavne v počítačových hrách a online rádiách. Pôvodne používal kontajner typu *OGG*, ktorý bol neskôr nahradený multifunkčnejším kontajnerom *Matroska*. Metadáta sú prístupné vo formáte *Vorbis comments*, ktorý bol pôvodne vytvorený priamo pre tento zvukový formát.

WMA

Windows Media Audio (WMA) je súbor zvukových formátov a kodekov vytvorených spoločnosťou Microsoft. Technológia je postavená na frameworku *Windows Media* a pozostáva z štyroch rôznych kodekov.

Kodek WMA, ktorý je najčastejšie využívaný, stratovo komprimuje dáta na základe psychoakustickým štúdií.

Kodek WMA Pro je vylepšenie pôvodného kodeku, ktorý je schopný kódovať zvuk vo vyššom rozlíšený, avšak stále v spojení so stratovou kompresiou.

Kodek WMA Losseless je verzia kodeku určená na kódovanie bezstratovou kompresiou podporujúca viackanálové audio.

WMA Voice je stratový kodek primárne určený na hlasové záznamy.

WMA je postavený na kontajnerovom formáte *ASF*. Tento formát definuje štruktúru a kódovanie metadát, ktoré sú blízke ID3 tagom. Medzi voliteľné metadáta patri napríklad hodnota zvukovej normalizácie alebo technológia *DRM*, slúžiaca na kontrolu autorských práv.

MPC

Mousepack (MPC) je open-source stratový formát primárne určený na kódovanie zvukových dát v rozmedzí 160-180 kbit/s. MPC využíva MPEG-2 kodek, ktorého psychoakustický model je rozšírený a podporu *CVD (Clear Voice Detection)*. Jeho prednosťou je vysoká kvalita kompresie pri nižších bitových tokoch v porovnaní s MP3 alebo AAC. Metadáta sú sprostredkované pomocou tagov APEv2.

Kapitola 3

Metadáta v hudobných súboroch

Metadáta reprezentujú informácie, ktoré sú zviazané v jednom súbore spolu s multimediálnym obsahom. Existujú mnohé implementácie metadát, mnoho z nich je aj štandardizovaných. V tejto kapitole budú tie najznámejšie a najpoužívanejšie implementácie detailne popísané. Záver kapitoly je venovaný konkrétnym metadátam, ktoré sú podstatné z pohľadu dídžeja.

3.1 ID3 tagy

ID3 tag je najznámejšou a najpoužívanejšou formou metadát hudobných súborov. Za ich rozsiahli rozmach môže samotný úspech MP3, pre ktorý bol tento formát metadát pôvodne navrhnutý. Celý princíp je postavený na súborovom type *kontajner*, ktorý je stavebným kameňom ID3 tagov. Projekt začal v roku 1996 a za ten čas prešiel viacerými zmenami.

ID3v1 bol prvým pokusom o združenie zvukového súboru s jeho textovým popisom. Táto metóda sa následne stala zaužívaným štandardom pre definíciu metadát k hudobným súborom. Telo ID3v1 tagov tvorilo 128 bajtov, ktoré mali presne definovaný význam. Metadáta sa k cieľovému súboru pripájali od konca, čo bolo dané tým, že vtedajšie prehrávače ešte neboli plne kompatibilné s touto metódou. Hudobné žánre boli definované číslom, ktoré im bolo v rámci štandardu pevne priradené. Bolo viac než jasné, že vývojom hudby budú vznikať nové hudobné žánre, čo viedlo k rozšíreniu repertoáru žánrov vývojármi, ktorý stáli za prehrávačom WinAmp[1].

Vylepšenie v podobe *ID3v1.1* prinieslo podporu informácie ohľadom poradia danej skladby v albume. Dosiahlo sa to tým, že pole pre komentár sa zmenšilo o 2 bajty z pôvodnej veľkosti, ktoré mohli byť následne využiteľné inak.

Táto implementácia žiaľ nebola vhodná pre ďalšie rozšírenia práve kvôli fixnej dĺžke obmedzenej na 128 bajtov. Z tohto dôvodu sa v roku 1998 začalo pracovať na *ID3v2* tagoch. Koncepcia sa rázne zmenila, čo bolo spôsobené potrebou streamovať hudbu spolu s metadátami. Tentokrát boli metadáta pripojené k súboru na jeho začiatku, aby tieto informácie boli prístupné ešte pred prehratím samotnej skladby. Podobu ID3v2 tagu ilustruje obrázok 3.1.

Metadáta sa skladajú z rámcov, ktoré môžu obsahovať ľubovoľnú informáciu. Veľkosť jedného rámca je maximálne 16 MB, pričom celý tag nesmie presiahnuť 256 MB. Vďaka prepracovanej štruktúre a použitiu rámcov je prípadné rozšírenie veľmi jednoducho realizovateľné vytvorením nových typov rámcov.

[1] https://www.winamp.com/

Pri tvorbe ID3v2 sa kládol dôraz na viacero kritérií, ktoré musel vynovený štandard spĺňať. Jedno z nich bola vlastnosť tagu neobsahovať žiadny *synchronizačný signál*, ktorý sa využíva pri dekódovaní a prehrávaní zvukových dát. Druhým kritériom bola rozšíriteľnosť a zavedenie podpory aj pre rámce, ktoré nie sú štandardizované. Z tohto dôvodu rámce okrem identifikátora povinne obsahujú aj svoju veľkosť, aby mohli byť bezpečne preskočené programom, ktorý ich nepozná.

Obr. 3.1: Štruktúra tagu ID3v2. Prevzaté z [5].

Prvá verejná verzia bola označovaná ako *ID3v2.2* a bola špecifická tým, že na identifikáciu rámcov využívala iba 3 bajty. To sa však náhle zmenilo rozšírením na 4 bajty vo verzii *ID3v2.3*, ktorá je dodnes najpoužívanejšou verziou ID3 tagov. Poslednou a najnovšou verziou je verzia *ID3v2.4*, ktorá prináša podporu kódovania v UTF-8 a možnosť pripojiť ďalšie tagy ku koncu súboru podobne ako v prípade ID3v1.

3.1.1 ID3v2.3

Hlavná štruktúra tagu sa skladá z hlavičky a jednotlivých rámcov, ktoré obsahujú metadáta. V nasledujúcich tabuľkách budú hexadecimálne hodnoty začínať znakom `$`, z čoho vyplýva, že spojenie `$xx` reprezentuje jeden bajt, kde `x` je hexadecimálne číslo. Bitové hodnoty začínajú znakom `%`, čo znamená, že spojenie `%xxxxxxxx` reprezentuje jeden bajt, kde `x` je binárne číslo. Štruktúra hlavičky je vyobrazená v tabuľke 3.1.[5]

ID3v2 identifikátor	`'ID3'`
ID3v2 verzia	`$03 00`
ID3v2 príznaky	`%abc00000`
ID3v2 veľkosť	`4 * %0xxxxxxx`

Tabuľka 3.1: Tabuľka znázorňuje konštrukciu hlavičky, ktorá pozostáva z prvých 10 bajtov súboru.

Prvé 3 bajty, inými slovami *identifikátor*, obsahuje reťazec, ktorý jednoznačne identifikuje, že sa na začiatku súboru nachádza ID3v2 tag. Nasledujúce 2 bajty vyjadrujú o ktorú

verziu sa jedná, pričom v tomto prípade ide o IDv2.3.0. V treťom poli sa nachádzajú pomocné *príznaky*, ktoré majú rozličné funkcie, kde

- `a` indikuje využitie mechanizmu pre odstránenie synchronizačného signálu z tagu,
- `b` indikuje, či po hlavičke nasleduje rozšírená hlavička,
- `c` slúži na experimentálne účely.

Posledné pole hlavičky, *veľkosť tagu*, je kódované 4 bajtmi, kde každý z nich začína nulovým bitom. Toto pole vyjadruje celkovú veľkosť tagu vrátane všetkých rámcov, výplne a rozšírenej hlavičky, ak sa tam nachádza. Ako príklad je možné uviesť postupnosť bajtov, ktoré reprezentujú prítomnosť ID3 tagu:

```
$49 44 33 yy yy xx zz zz zz zz
```

- `yy` – bajt verzie (`y < $FF`)
- `xx` – bajt príznakov
- `zz` – bajt veľkosti (`zz < $80`)

Rozšírená hlavička neobsahuje nevyhnutné informácie pre spracovanie tagu, preto je jej použitie voliteľné. Obsah rozšírenej hlavičky pozostáva z jej veľkosti, veľkosti výplne a rozšírenia poľa príznakov pre podporu voliteľných funkcií ako je napríklad *CRC kontrola*.

Podobne ako štruktúra celého tagu, tak aj jednotlivé rámce obsahujú svoju hlavičku. Validný tag musí obsahovať aspoň jeden rámec o veľkosti minimálne 1 bajt, pričom veľkosť hlavičky sa do celkovej veľkosti nepočíta. Konštrukcia hlavičky rámca je znázornená tabuľkou 3.2.

Identifikátor	`$xx xx xx xx` (štyri znaky)
Veľkosť	`$xx xx xx xx`
Príznaky	`%abc00000 %ijk00000`

Tabuľka 3.2: Tabuľka znázorňuje konštrukciu hlavičky jedného rámca, ktorá pozostáva z identifikátora, veľkosti a príznakov.

Identifikátor sa skladá z veľkých písmen (A–Z) a čísla (0–9), avšak identifikátory začínajúce na X, Y alebo Z, slúžia na experimentálne účely. Prevažná väčšina rámcov má textový charakter, až na pár špeciálnych výnimiek. Príklad typického identifikátora je napríklad `TIT2`, ktorý naznačuje, že rámec by mal obsahovať názov skladby. Pri štandardných textových rámcoch začína identifikátor písmenom `T`, pri rámcoch obsahujúcich webové stránky je to písmeno `W` a špeciálne rámce majú nezávislé označenie. Popis všetkých podporovaných rámcov je dostupný v dokumentácií.[5]

Veľkosť rámca bez hlavičky tvorí pole 4 bajtov. Príznaky sú tvorené 2 bajtmi, ktoré majú nasledovný význam:

- Príznak `a` popisuje, či sa má rámec zahodiť, ak je neznámy a tag je pozmenený.
- Príznak `b` popisuje, či sa má rámec zahodiť, ak je neznámy a súbor je pozmenený.

- Príznak `c` označuje rámec, ktorý je určený iba na čítanie.
- Príznak `i` indikuje, či je daný rámec komprimovaný.
- Príznak `j` indikuje, či daný rámec šifrovaný.
- Príznak `k` označuje rámec, ktorý patrí do skupiny rámcov a nesie informácie o tejto skupine.

Zvukové dáta vo formáte MPEG majú taktiež podobu rámcov, ktoré tvoria spolu s metadátami dátový tok. Hlavným cieľom je nájsť také rámce v toku, ktoré je možné dekódovať. Tomuto procesu sa hovorí *synchronizácia* a spočíva v používaní synchronizačného slova na začiatku každého rámca.[12]

Synchronizačné slovo je vyjadrené 11 alebo 12 binárnymi jednotkami za sebou v jednom rámci. Tento vzor sa však v tagu nachádzať nesmie, inak by mohol byť samotný tag reprezentovaný ako zvukové dáta. Za účelom odstránenia prípadných synchronizačných slov sa používa mechanizmus, ktorý má podporu aj v ID3 tagoch. Princíp spočíva v doplnení jedného nulového bajtu, ktorý rozdelí postupnosť nasledovne:

`%11111111 111xxxxx` → `%11111111 00000000 111xxxxx`

Vedľajším efektom je nutnosť transformovať všetky `$FF 00` na `$FF 00 00`, inak by ich proces dekódovania mohol negatívne ovplyvniť. Pri kódovaní treba brať v úvahu, že odstránenie synchronizácie z tagov musí byť vykonané až po kompresií. V prípade dekódovania musí byť poradie presne opačné.

3.1.2 ID3v2.4

Hlavička má rovnakú štruktúru ako predchádzajúca verzia, pribudol len jeden nový príznak. Nová verzia podporuje pätičku na konci súboru, ktorá je kópiou pôvodného tagu len s iným identifikátorom. Jej hlavnou úlohou je urýchliť vyhľadávanie od konca súboru. Táto implementácia však vyžaduje v prvom tagu `SEEK` rámec, ktorý obsahuje informácie o pätičke. Rozdielnosť novej hlavičky je možné vidieť v tabuľke 3.3.

ID3v2 identifikátor	`'ID3'` (v pätičke `'3DI'`)
ID3v2 verzia	`$04 00`
ID3v2 príznaky	`%abcd0000`
ID3v2 veľkosť	`4 * %0xxxxxxx`

Tabuľka 3.3: Tabuľka znázorňuje konštrukciu hlavičky tagu ID3v2.4, kde význam jednotlivých príznakov je obdobný s verziou ID3v2.3, pričom nový príznak `d` indikuje, či sa na konci súboru nachádza pätička.

Štruktúra rámcov pozostáva z hlavičky a tela. Hlavička obsahuje identifikátor, veľkosť a príznaky, podobne ako v predchádzajúcej verzií. Hlavnými zmenami prešli príznaky, ktoré majú nasledovný tvar:

`%0abc0000 %0h00kmnp`

Na prvý pohľad si je možné všimnúť, že príznaky stále pozostávajú z dvoch bajtov. Prvý bajt ostal významom rovnaký, posunula sa však pozícia príznakových bitov o jeden bit doprava. Druhý bajt prešiel viacerými zmenami, pričom aktuálna interpretácia príznakov je nasledovná:

- Príznak `h` označuje rámec, ktorý patrí do skupiny rámcov a nesie informácie o tejto skupine.
- Príznak `k` indikuje, či je daný rámec komprimovaný.
- Príznak `m` indikuje, či daný rámec šifrovaný.
- Príznak `n` indikuje, využitie mechanizmu pre odstránenie synchronizačného signálu z rámca.
- Príznak `p` indikuje, či bol do rámca pridaný indikátor veľkosti rámca.

3.2 APE tagy

APE tagy sú alternatívou k populárnym tagom ID3. Pôvodne vznikli ako kontajner pre zvukové súbory vo formáte APE, neskôr však dostali podporu aj pre iné formáty. Štruktúra metadát pozostáva z *položiek*, ktoré obsahujú *kľúč* a *hodnotu*. Celý tag je situovaný na konci zvukového súboru za účelom bezproblémovej rozšíriteľnosti. V prípade novšej verzie môže byť aj na začiatku, čo sa ale z dôvodu horšej kompatibility neodporúča.[4]

APEv1 je prvá verzia, ktorá používa kódovanie ASCII a neobsahuje hlavičku. Telo tagu, zobrazené tabuľkou 3.4, sa skladá z *položiek (items)*. Položky pozostávajú z množiny predom definovaných *kľúčov*, *hodnoty* určitého typu alebo kódovania a *veľkosti tejto hodnoty*. Každý kľúč sa môže v tagu vyskytovať maximálne jedenkrát a plní podobnú funkciu ako identifikátor rámca v ID3 tagoch. Koniec tagu je tvorený *pätičkou (footer)*, ktorá obsahuje číslo *verzie tagu*, *celkovú veľkosť tagu* a *počet položiek*.

Hlavička	32 bajtov
Položka č. 1	$10 - m$ bajtov
Položka č. 2	$10 - m$ bajtov
...	...
Položka č. n	$10 - m$ bajtov
Pätička	32 bajtov

Tabuľka 3.4: Štruktúra APE tagu, kde hlavička bola pridaná až vo verzií APEv2. Hodnota m vyjadruje maximálnu možnú veľkosť jednej položky.

Novšia verzia označovaná ako *APEv2*, kódovaná pomocou UTF-8, priniesla podporu príznakov a hlavičky. Jej tvorba bola bola podmienená formátom MPC, ktorý sa mal stať hlavnou doménou APEv2 tagov. *Hlavička* tvorí presnú kópiu *pätičky*, pričom pri oboch pribudlo pole pre *príznaky*. Práve prvý príznakový bit rozlišuje medzi hlavičkou a pätičkou tak, ako je naznačené v tabuľke 3.5.

Telo položky je znázornené tabuľkou 3.6, kde *len* vyjadruje veľkosť hodnoty obsiahnutú na začiatku každej položky. Hodnota k určuje veľkosť kľúča, ktorá je v rozmedzí 2 až 255 bajtov. Definícia príznakov jednotlivých položiek je dostupná v dokumentácií[2].

[2] http://wiki.hydrogenaud.io/index.php?title=Ape_Tags_Flags

Preambula	8 bajtov	`'APETAGEX'`
Verzia tagu	8 bajtov	`1000` (APEv1) `2000` (APEv2)
Veľkosť tagu	8 bajtov	Veľkosť tagu bez hlavičky
Počet položiek	8 bajtov	Číslo počtu položiek
Príznaky tagu	8 bajtov	Globálne príznaky
Rezervované	8 bajtov	Vynulované

Tabuľka 3.5: Tabuľka vyobrazuje formu hlavičky a pätičky v APEv2 tagoch.

Veľkosť hodnoty	4 bajty	Dĺžka hodnoty v bajtoch
Príznaky	4 bajty	Lokálne príznaky
Kľúč	*k* bajtov	`'Title'`, `'Artist'` ...
0x00	1 bajt	Ukončovací bajt kľúča
Hodnota	*len* bajtov	UTF-8 alebo binárne dáta

Tabuľka 3.6: Tabuľka zobrazuje štruktúru jednej položky v APEv2 tagoch.

3.3 Vorbis comments

Vorbis comments tvorí metadátový kontajner, ktorý slúži na uchovanie dodatočných informácií textového charakteru. Jeho štruktúra je veľmi jednoduchá a z časti pripomína štruktúru APE tagov.

Vorbis tag obsahuje *položky*, ktoré pozostávajú z *názvu* a *hodnoty*. Názov je kódovaný v ASCII a obsahuje iba tlačiteľné znaky. Hodnota podporuje kódovanie UTF-8 a jej obsah je ľubovoľný. Výsledný formát tagu je zoznam položiek v tvare `Názov = Hodnota`. Položky s rovnakým názvom sa môžu v jednom tagu nachádzať viac krát, čo sa neskôr interpretuje ako jedna položka, spojením všetkých hodnôt rovnakého názvu. Špecifikácia Vorbis tagov obsahuje niekoľko preddefinovaných názvov, ktoré je možné použiť.[16]

3.4 Metadáta pre dídžejing

Medzi hlavné predispozície profesionálneho díďžeja patrí bohatá hudobná knižnica, ktorú je nutné spravovať takým spôsobom, aby bola čo najefektívnejšie využiteľná v praxi. To znamená, že je vhodné využívať plný potenciál metadát hudobných súborov pre ich jednoduchšiu klasifikáciu a manipuláciu. V tejto kapitole špecifikujem, ktoré technické a informačné detaily hudobných súborov potrebuje díďžej k úspešnému plneniu svojej práce.

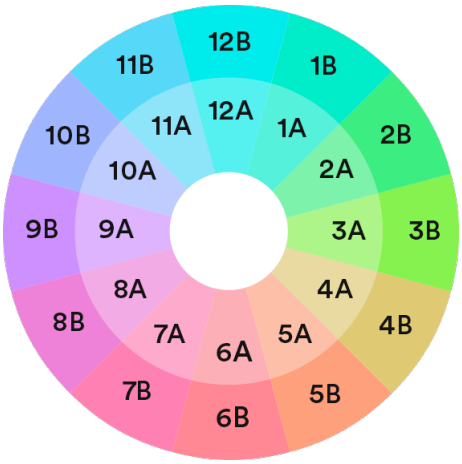
- **Názvy skladieb, interpretov a albumov** patria všeobecne medzi hlavné metadáta. Takmer každé zariadenie určené na prehrávanie digitálnej hudby podporuje čítanie týchto metadát, najmä za účelom ich štrukturovaného zobrazenia. Keďže identifikácia obsahu zvukového súboru je najpodstatnejším významom existencie tagov, platí to aj v tomto prípade.
- **Hudobný žáner** je ďalšou podstatnou informáciou v tele tagu. Žánre klasifikujú hudbu na rozsiahlej množine vlastností, ktoré daný žáner definujú. Z tohto dôvodu je využitie tejto vlastnosti vhodné najmä z pohľadu triedenia a vyhľadávania. Keďže

prax dídžeja vyžaduje prípravu vhodných skladieb ešte pred vystúpením, jeden z hlavných podporných informácií pri výbere skladieb je ich žáner.

- **Rok (dátum) vydania** definuje hudbu na základe roku jej vzniku. Táto informácia radí skladby do určitého obdobia, ktoré neskôr umožňuje jednoduchší výber a zvýši priehľadnosť v hudobnej knižnici. Táto položka však môže obsahovať aj zavádzajúce informácie, ktoré nemusia úplne zodpovedať ich pôvodnému významu. Napríklad v prípade, že vydanie skladby predchádzalo jej zahrnutiu v nadchádzajúcom albume, môže táto skladba obsahovať dátum vydania albumu, čo však nie je úplne pravdivá informácia. Kvôli tomu má mnoho tagov položky ako pôvodný rok vydania, pôvodný interpreti a podobne.

- **Tempo** je vlastnosť skladby udávaná v úderoch za minútu (BPM). Táto vlastnosť je jednou z najhlavnejších informácií o skladbe z pohľadu dídžeja, pretože umožňuje vykonávanie prechodov a mixovanie skladieb. Trik spočíva v synchronizácií tempa všetkých mixovaných skladieb a následnej úpravy hlasitosti, tak aby celý tento proces znel prirodzene. Software i hardware, určený na mixovanie, poskytuje natívnu podporu pre počítanie tempa skladby hneď pri jej načítaní. V prípade, že táto možnosť nie je k dispozícií alebo potrebujeme mať informácie o tempe v predstihu, je možné uložiť hodnotu tempa aj v podobe tagu.

- **Kľúč** udáva v akej tónine (frekvenčnej výške) sa skladba nachádza. Tento atribút poskytuje kritérium, ktoré je vhodné dodržiavať pri mixovaní skladieb, tak aby výsledný mix znel harmonicky. Na obrázku 3.2 je zobrazený tzv. *Camelot Wheel*, ktorý určitému kľúču priradí množinu vhodných kľúčov z jeho okolia. Podobne ako pri tempe, tak aj kľúč je možné dopočítať v príslušnom software.



Obr. 3.2: Obrázok znázorňuje tzv. Camelot Wheel, ktorý poukazuje na harmonickú kompatibilitu kľúčov pri mixovaní. Prevzaté z [8].

- **Hlasitosť** je vyjadrením intenzity zvuku a udáva sa v jednotke decibel (dB). Pri nahrávaní skladieb môže dôjsť k určitým odlišnostiam zo strany celkovej hlasitosti. Jednoducho povedané, hlasitosť jednej skladby je vyššia ako hlasitosť skladby druhej, a to aj v prípade, že obe skladby pochádzajú z rovnakého zdroja alebo albumu. Tento

problém je možné riešiť v podobe položky v tagu, ktorá sa nazýva *Replay Gain*. Špeciálne pri mixovaní skladieb je nutné, aby všetky skladby s ktorými sa narába, boli v neutrálnom stave na rovnakej úrovni hlasitosti.

- **Nastavenie ekvalizéra (EQ preset)** je preddefinované nastavenie hlasitosti v jednotlivých frekvenčných pásmach. Zápis tejto hodnoty je často relatívny k priemernej hlasitosti zvukového súboru. Z hľadiska kompatibility je však využiteľnosť tejto položky nízka, navyše drvivá väčšina zariadení či aplikácií obsahuje vstavaný ekvalizér, ktorý je možné upravovať počas prehrávania. Na druhej strane, pokiaľ prehrávač nedisponuje vlastným ekvalizérom, využitie tejto položky môže prísť vhod.

- **Kvalita zvuku a jeho technická špecifikácia** sú ďalšie faktory, ktoré je možné brať v úvahu pri práci dídžeja. Technické vlastnosti zvukového súboru typicky nemajú formu tagu, ale zariadenia a aplikácie sú schopné tieto údaje zistiť zo samotného dátového toku. V digitálnej forme je kvalita zvuku typicky určovaná jej formátom a bitovým tokom. Ako už bolo spomenuté v kapitole 2.2, niektoré formáty v sebe zahŕňajú určitú úroveň kompresie, ktorá negatívne vplýva na výslednú kvalitu zvuku. V praxi však nie sú vyžadované zvukové súbory v štúdiovej (bezstratovej) kvalite, pretože mnohé zvukové systémy využívané na verejných akciách, tzv. *PA systémy*, nie sú schopné dôkladne vykresliť všetky hudobné detaily. Je preto vhodné zvoliť takú úroveň kvality, ktorú je možné plne vykresliť, ale zároveň má rozumné pamäťové nároky.

- **Hodnotenie skladby** je často používaný atribút, ktorý umožňuje ohodnotiť skladbu na úrovni 1 až 5 (hviezdičiek). Toto hodnotenie podporujú mnohé formáty tagov podobne ako samotné prehrávače. Hodnotenie poskytuje užívateľsky definovateľný parameter, ktorý dopomáha k triedeniu skladieb v knižnici.

- **Komentár** je položka, v ktorej je možné uložiť dodatočné informácie, ktoré inak nemajú presne stanovené miesto v tagu. Výhodou použitia komentárov je široká podpora a kompatibilita zo strany softwaru, ktorý nemusí rozpoznať užívateľsky definované položky metadát.

Kapitola 4

Zhodnotenie súčasného stavu a návrh riešenia

V tejto kapitole sa nachádza vyhodnotenie súčasných riešení a služieb, ktoré úzko súvisia s hudbou a informáciami okolo nej. V sekcií 4.1 sú predstavené populárne editory tagov, ale aj služby, poskytujúce prístup k metadátam v internetovom prostredí. Nasledujúca sekcia 4.2 popisuje spôsob rozoznávania hudobných skladieb na základe odberu vzorky signálu. Kapitolu uzatvára návrh aplikácie, ktorá je schopná popísané služby využiť v podobe praktického nástroja na správu metadát hudobných súborov.

4.1 Služby poskytujúce prístup k metadátam

4.1.1 Editory tagov

Editor tagov je software určený pre správu metadát hudobných súborov. Jeho hlavnou úlohou je ponúknuť užívateľsky prívetivé prostredie pre úpravu tagov s čo najširšou podporou formátov a štandardov. Mnohé z editorov sa líšia zameraním a funkcionalitou, pričom väčšina sa snaží o čo najflexibilnejšie riešenie.

Medzi základné funkcie patrí *schopnosť manuálnej úpravy* tagov jednotlivých zvukových súborov. Ďalšou užitočnou funkciou je schopnosť *komunikovať s internetovou databázou*, ktorá obsahuje plno informácií o skladbách a ich interpretoch. Na základe tejto komunikácie je daný software schopný stiahnuť a doplniť metadáta k príslušným súborom. V prípade, že súbor neobsahuje žiadne metadáta a jeho názov neidentifikuje skladbu, je možné využiť služby *identifikácie zvukovej stopy (fingerprinting)*. Služba umožňuje využiť vzorku signálu zvuku v podobe odtlačku, ktorý sa následne porovná s odtlačkami v externej databáze. Na základe kladného výsledku je potom možné určiť všetky potrebné metadáta a uložiť ich.

- *EasyTag*[1] je bezplatný open-source software určený pre Linux a Windows. Implementácia je založená na jazyku C a GTK+. Grafické rozhranie spolieha na jednoduchosť a intuitívnosť. Podporuje takmer všetky známe zvukové formáty a kontajnery. Medzi ďalšie funkcie patrí automatizované dopĺňanie tagov podľa predpisu, čítanie hlavičiek zvukových súborov, generovanie playlistov a podpora CDDB.

[1] https://wiki.gnome.org/Apps/EasyTAG

- *MusicBrainz Picard*[2] je bezplatný multiplatformový open-source software určený na správu hudobnej knižnice napísaný v jazyku Python. Odlišuje sa hlavne tým, že poskytuje vlastnú databázu skladieb, ktorú využíva k automatizácií procesu spracovania hudobných súborov. Databáza je verejne dostupná na stránkach služby a v podobe programového API.

4.1.2 Internetové hudobné databázy

Internetové databázy poskytujú širokú ponuku metadát a dodatočných informácií nielen o hudobných skladbách. Ich primárnym zameraním je slúžiť ako verejná encyklopédia, ktorej obsah je dotváraný širokou verejnosťou. Databázy obsahujú štruktúrované informácie o umelcoch, skladbách, albumoch, skladateľoch, producentoch a rôznych väzbách medzi nimi. Mnohé databázy sú schopné sprostredkovať okrem textových dát aj obrázky albumov alebo dokonca zvukové odtlačky skladieb. Časté využitie majú v rámci klientskych aplikácií, ktoré sprostredkúvajú ich služby koncovým používateľom. V tejto podsekcií sú popísané hlavné črty a vlastnosti databázových služieb, ktoré boli využité aj pri návrhu finálnej aplikácie.

Discogs

Discogs patrí medzi najväčšie hudobné databázy, ktoré fungujú na princípe verejného pridávania, kde informačný obsah tvoria príspevky od bežných používateľov. Po prihlasení je možné pridávať rôznorodé informácie v podobe *príspevku (submission)*, ktoré musia byť následne schválené *volebným procesom*. Tento proces taktiež vykonáva „široká verejnosť“, teda bežný používatelia tejto služby.

Pri pridávaní je nutné rozlišovať *vydania (releases)* podľa média, ktorým boli distribuované. Discogs rozlišuje medzi *digitálnymi médiami (CD)*, *platňami (vinylové platne)* a *kazetami (CC)*. Vydanie tvorí hlavný stavebný kameň databázy, od ktorého sa neskôr odvodzujú ostatné informácie. Pod jedným vydaním sa rozumie jedno vydanie albumu, kompilácie skladieb alebo jednej skladby (singel) v danej krajine, príslušným vydavateľstvom a na konkrétnom médiu. To znamená, že jeden album môže a zvyčajne má viac vydaní, ktoré sa môžu líšiť. Databáza obsahuje mnohé väzby medzi umelcami, ich tvorbou a vydavateľstvom (label). Medzi podstatné informácie, ktoré Discogs plne podporuje, patrí *katalógové číslo (catalogue number)* konkrétneho vydania. Toto číslo jednoznačne identifikuje dané vydanie. Vydania majú obvykle jednu alebo viacero skladieb, ktoré sú očíslovane podľa poradia v danom vydaní. Vydania môžu obsahovať aj informácie o skladateloch, producentoch, dirigentoch a iných ľudí zo „zákulisia“.

MusicBrainz

MusicBrainz je projekt, ktorý vznikol za účelom vytvorenia hudobnej encyklopédie. Architektúra databázy má niekoľko podobných čŕt s Discogs. Dáta sú tvorené užívateľmi, ktorý si ich navzájom schvaľujú.

Proces pridávania je veľmi podobný tomu v Discogs, preto ho nebudem detailne opisovať. Za zmienku však stojí volebný systém nových príspevkov. Každý nový príspevok má 7 dní na to, aby získal dostatok kladných hlasov. Pokiaľ v tejto dobe príspevok nedostane dostatok kladných hlasov alebo žiadny hlas, aktuálny počet sa vyhodnotí. V prípade, že kladných hlasov je viac než negatívnych alebo nie sú žiadne hlasy, príspevok je schválený. Príspevok je naopak zamietnutý vtedy, keď je záporných hlasov viac ako kladných.

[2] https://picard.musicbrainz.org/

To že databáza stojí na podobnom princípe ako Discogs značí, že hlavnou entitou je opäť jedno *vydanie*. Podrobnú schému databáze je možné nájsť v oficiálnej dokumentácií[3]. Z technickej stránky je založená na technológií *PostgreSQL*, ktorá tvorí primárnu relačnú databázu. Vyhľadávanie je založené na indexovaní, ktoré vylepšuje technológia *SOLR* projektu Apache Lucene. Táto implementácia sprístupňuje koncovým užívateľom tie najrelevantnejšie informácie, ktoré systém vyhodnotí ako najpravdepodobnejšie. Služba taktiež podporuje pokročilú syntax dotazov, ktorej detailný popis je dostupný na stránkach projektu Apache Lucene[4].

Spotify

Spotify je služba, ktorá je aktuálne veľmi obľúbená v oblasti *streamovania* hudby. Vysoká popularita zabezpečila veľkú hudobnú databázu, ktorá obsahuje nemalé množstvo hudobných metadát. Na rozdiel od predchádzajúcich služieb, Spotify tvorí oficiálny obsah, ktorý je získaný priamo od zdroja, teda priamo od umelcov a vydavateľov. Služba ma komerčný charakter, teda v prípade, že sa chce užívateľ vyhnúť obmedzeniam a reklamám, musí si za službu platiť. Z týchto poplatkov putuje až 70 % priamo umelcom a vydavateľom na základe ich popularity a počte prehratí.

Služba umožňuje aj technickú analýzu skladieb, čo znamená, že okrem informačných metadát poskytuje aj informácie o tempe skladby, kľúča, nálady, akustiky, energickosti a mnoho ďalších. Tieto informácie umožňujú tvoriť *zoznamy skladieb (playlist)* podľa preferencií poslucháča. Algoritmy sprostredkovala spoločnosť The Echo Nest[5], ktorá bola v roku 2014 odkúpená práve spoločnosťou Spotify. Táto platforma sa dlhodobo zaoberá inteligentným skúmaním hudby pre účely identifikácie a predikcie.

4.2 Rozpoznávanie hudby odtlačkom zvukového signálu

Rozpoznávanie hudby pomocou odberu vzorky zvukového signálu (acoustic fingerprinting) je jedna z metód identifikácie hudobných súborov. Princíp spočíva v extrahovaní potrebných informácií zo vstupného zvukového signálu, ktorý je následne porovnávaný s databázou. Medzi najpopulárnejšie komerčné služby, ktoré túto metódu využívajú patrí multi-platformová aplikácia Shazam[6]. V tejto sekcií budú predstavené základné koncepty tohto procesu, ktorý si našiel miesto aj v koncovom návrhu.

4.2.1 Základný koncept

Diskrétna Fourierova transformácia

Zvukový signál je možné bežne zobraziť pomocou priebehu vo tvare *sínusoidy*. Táto reprezentácia však nie je vhodná pre účely hlbšej analýzy, preto sa využívajú techniky na jej transformáciu. Bitový tok digitálneho signálu pozostáva z jednotiek a núl, ktoré vyjadrujú frekvencie. Odtlačky, ktoré sa pri identifikácií skladieb používajú, pozostávajú práve z frekvencií. Z tohto dôvodu je nutné previesť zvukový signál z časovej do frekvenčnej domény. Pri diskrétnych signáloch sa využíva *Diskrétna Fourierova transformácia (DFT)*, ktorej

[3] https://musicbrainz.org/doc/MusicBrainz_Database/Schema
[4] https://lucene.apache.org/solr/guide/6_6/the-standard-query-parser.html
[5] http://the.echonest.com/
[6] https://www.shazam.com/

definícia je vyjadrená vzťahom

$$X_n = \sum_{k=0}^{N-1} x_k e^{-j(2\pi kn/N)} \tag{4.1}$$

kde X_n vyjadruje frekvenčný rozsah n-tej vzorky, N je *veľkosť časového okna* vo vzorkách a x_k je k-ta vzorka signálu.

Výstupom tejto transformácie je vyjadrenie signálu frekvenciou a jej intenzitou. Každá vzorka reprezentuje určitý frekvenčný rozsah (frequency bin), ktorý závisí od veľkosti časového okna a štandardnej vzorkovacej frekvencie vstupného signálu. Napríklad, ak vstupný diskrétny signál je vzorkovaný na obvyklej frekvencií 44,1 kHz a veľkosť okna je na úrovni 4096 vzoriek počas 0,1 sekundy signálu, výsledný frekvenčný rozsah (rozlíšenie) je 10,77 Hz. Tento údaj je možné získať jednoduchým delením vzorkovacej frekvencie veľkosťou časového okna. Zvýšenie frekvenčného rozsahu vzorky je možné dosiahnuť zväčšením okna, čo má za následok jeho časové predĺženie a stratu schopnosti detekovať rýchle frekvenčné zmeny.[2]

Z praktických a hlavne časových dôvodov sa na počítanie Diskrétnej Fourierovej transformácie využívajú algoritmy *Rýchlej Fourierovej transformácie* (FFT). Najznámejším zástupcom týchto algoritmov je *Cooley-Tukey algoritmus*.[1]

Časové okno frekvenčnej analýzy

Ak je pri analýze cieľom získať frekvencie v pravidelných okamihoch, napríklad v každej sekunde signálu o dĺžke 10 sekúnd, potom je nutné vykonať Diskrétnu Fourierovu transformáciu až 10krát. Tento proces je implicitne spojený s násobením pôvodného signálu funkciou časového okna. Najprimitívnejším príkladom takého okna je *obĺžnikové okno*. Pri aplikácií tohto typu okna je vstupný vzorkovaný signál násobený 1 (v prípade počítanej vzorky) alebo 0 všade inde. Pri aplikácií tohto okna na všetky vzorky signálu vzniká vedľajší efekt nazývaný ako *preskakovanie spektra (spectral leakage)*. Tento efekt negatívne vplýva na výslednú transformáciu, pretože intenzívne prvky s vysokou amplitúdou ovplyvňujú okolité vzorky. Existuje viac druhov okien, ktoré vďaka ich tvaru tento negatívny efekt čiastočne potláčajú.[7]

4.2.2 Vytváranie digitálneho odtlačku

Downsampling

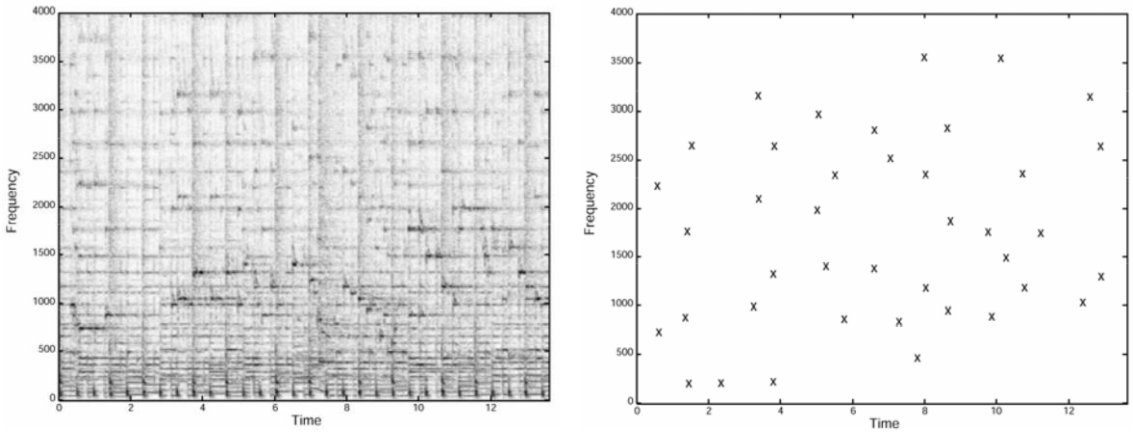
Dokonca aj pri Rýchlej Fourierovej transformácií je čas potrebný na výpočet prudko závislí na veľkosti časového okna. Za účelom minimalizácie časovej náročnosti sa využíva metóda nazývaná *downsampling*. Táto metóda umožňuje znížiť veľkosť okna pri zachovaní pôvodného frekvenčného rozlíšenia tým, že zníži vzorkovaciu frekvenciu vstupného zvukového signálu. Za týchto okolností sa maximálne frekvenčné pásmo oreže, pričom bude obsahovať len tie najpodstatnejšie informácie zostalého pásma. Napríklad pri vzorkovacej frekvencií 11 kHz, je amplitúda frekvencie na úrovni zhruba 5 Hz. Služby, ktoré slúžia na detekciu skladieb pomocou odtlačku, túto techniku praktizujú z toho dôvodu, že najpodstatnejšia informácia sa nachádza presne v tomto zostalom pásme.[2]

Filtering

Jedným zo spôsobov vytvorenia odtlačku zvukového súboru je využitie *spektogramu*. Porovnávanie na bitovej úrovni nie je zrovna praktické, kvôli použitiu rôznych zvukových

formátov, ktoré kódujú dáta rozličným spôsobom. Spektogram je vytvorený až po vykonaní downsamplingu, aby neobsahoval nepodstatné informácie a jeho veľkosť bola priateľná. Z vytvoreného spektogramu sa následne vyberú tie najvýraznejšie frekvencie, pretože vybrané algoritmy počítajú s určitou mierou šumu. Zjednodušený algoritmus pozostáva z týchto krokov[2]:

1. Rozdelenie každého výsledku FFT logaritmicky na frekvenčné pásma.
2. Pre každé pásmo sa vyberú najvýraznejšie frekvencie.
3. Z týchto vybraných frekvencií sa určí arimetický priemer.
4. Zachovajú sa iba frekvencie, ktoré sú nad týmto priemerom.



Obr. 4.1: Názorná ukážka výberu výrazných frekvencií z orezaného spektogramu skúmaného zvukového súboru. Prevzaté z [14].

4.2.3 Porovnávanie a hodnotenie

Ak je zvukový odtlačok vytvorený, nasleduje proces *porovnávania a vyhodnocovania*. Aplikácie, ktoré poskytujú túto službu, využívajú architektúru *klient-server*. Hneď po skonštruovaní odtlačku posielajú tento odtlačok do databázy na porovnanie. Ak databáza narazí na zhodu, ktorá vyhovuje určitej akceptovatelnej podobnosti, posiela informácie o náleze späť klientovi. To, ako jednotlivé porovnávanie a hodnotenie funguje, závisí od použitého algoritmu a implementácie.

Chromaprint

Chromaprint je hlavný komponent projektu AcoustID[7]. Je to knižnica, určená na klientske použitie, ktorá obsahuje algoritmy vykonávajúce proces extrakcie zvukového odtlačku zo vstupného súboru.

Výsledná implementácia využíva taktiež spektogram k reprezentácií výsledného odtlačku. Na rozdiel od predchádzajúceho príkladu, všetky výrazné frekvencie premieňa na

[7] https://acoustid.org/

hudobné *noty*. Výsledok je možné reprezentovať vo forme obrázka, ktorý je následne porovnávaný s odtlačkami príslušnej databázy. Obrázky 4.2 a 4.3 ilustrujú podobu takto vytvorených odtlačkov.

Obr. 4.2: Reprezentácia zvukového signálu pomocou algoritmu knižnice Chromaprint.

Obr. 4.3: Finálna podoba odtlačku, ktorá sa využíva pri porovnávaní.

Porovnávanie prebieha spracovaním finálnej reprezentácie signálu a rozdelením na menšie obrázky. Na každý z týchto obrázkov sa aplikuje jeden z *16 filtrov*, ktoré zachytávajú rozdiely intenzity jednotlivých nôt v čase. Filtre spočítavajú malé časti obrázku, ktorý je prevedený na čierno-biely, podľa rozloženia čiernych a bielych pixelov. Po ich odpočítaní vznikne jedno reálne číslo, ktoré je podľa filtra prevedené na celé *dvojbitové číslo (Gray code)*. Po aplikácií všetkých filtrov, každý z obrázkov je definovaný 32-bitovou hodnotou, ktorá určuje výsledný odtlačok. Jednotlivé filtre boli zvolené na základe strojového učenia a dlhodobého testovania. Najjednoduchší spôsob porovnávania takýchto odtlačkou je pomocou *bitovej chybovosti (BER)*.[6]

4.3 Návrh nástroja pre správu metadát

V tejto sekcií sa nachádza popis návrhu aplikácie, ktorej cieľom je vytvoriť univerzálne rozhranie pre správu metadát hudobných súborov. V návrhu a následnej implementácií sú využité mnohé služby a nástroje, ktoré boli v tejto práci už predstavené.

Hlavná myšlienka

Hlavnou motiváciou je vytvoriť nástroj, ktorý bude dostatočne intuitívny a použiteľný aj v rukách bežného človeka. Z tohto dôvodu návrh čerpá zo zaužívaných nápadov, ktoré využíva prevažná väčšina súčasných desktopových aplikácií.

Aplikácia má za úlohu riešiť viacero problémov a ponúknuť rozličné prostriedky k ich riešeniu. Metadáta hudobných súborov sú väčšinou textového charakteru, čo znamená, že výsledný program musí vedieť s takými informáciami efektívne pracovať. To platí aj z pohľadu zvyšných typov dát, ktoré sa v tagoch môžu nachádzať.

Medzi hlavné črty a odlišnosti môjho riešenia patrí automatizácia získavania metadát z internetových databáz. Konvenčné aplikácie, ktoré som mal možnosť testovať, z tohto hľadiska nenapĺňali moje očakávania. Dôvodom je hlavne nepresnosť vyhľadávania alebo nízka kapacita použitej databázy. Cieľom môjho návrhu je vytvoriť logiku, ktorá je na základe dostupných informácií, schopná vyhľadať a získať ďalšie dodatočné dáta. Tieto dáta zároveň čerpá z viacerých zdrojov, podľa preferencií koncového používateľa. Zdrojom metadát sú

dostupné služby, ktoré poskytujú voľný prístup k ich databázam. V prípade nedostupnosti informácií, ktoré definujú nevyhnutné identifikačné údaje, sa využíva identifikácia na základe zvukového odtlačku. Toto všetko by malo byť možné jednoduchým stlačením tlačidla, bez žiadnych ďalších zásahov používateľa.

Predpokladané cieľe

Už pri návrhu je z princípu vhodné poznať všetky ciele a parametre vyvíjaného softwaru. Ak sú ciele stanovené vhodne, tak sú aj v očakávanom čase realizovateľné. Z tohto pohľadu som si stanovil nasledovné ciele:

- **Jednoduchosť používania** je spojená s vytvorením intuitívneho používateľského rozhrania, ktoré bude jednoduché a priehľadné. Hlavná koncepcia spočíva v zobrazení hudobnej knižnice, podľa zvoleného priečinku či umiestnenia v zariadení. Týmto spôsobom môže užívateľ jednoducho vykonávať akcie, pre ktoré bola aplikácia stvorená. Všetky nástroje sú zreteľne označené vo vyhradenom panely, aby boli vždy k dispozícií, keď je ich potreba. V prípade, že ich funkcia nie je známa, je možné využiť pomocníka.

- **Lokálna editácia tagov** je jedna z primárnych funkcií, ktoré musí editor tagov podporovať. Táto funkcionalita umožňuje upravovať všetky dostupné polia daného formátu tagu. Výber skladby je realizovaný pomocou grafického rozhrania, ktoré zobrazuje aj prípadné modifikovateľné metadáta.

- **Správa obrázkov pripojených k metadátam** je ďalšia funkcia, ktorá úzko súvisí s klasickou editáciou tagov. Jediný rozdiel je v tom, že sa nejedná o textové dáta ale o bitmapové obrázky. Obrázky je možné pridávať buď z lokálneho disku, alebo umiestnenia na internete.

- **Nástroje na manipuláciu s lokálnymi súbormi** sú metódy, poskytnuté skrz užívateľské rozhranie, ktoré užívateľovi poskytujú schopnosť manipulovať s vyhradenými súbormi na úrovni operačného systému. To znamená, že je možné súbory premenovať, premiestniť, kopírovať či vymazať.

- **Nástroje na radenie hudobných súborov podľa metadát** tvoria súbor funkcií, ktoré sú na základe užívateľskej konfigurácie schopné vykonávať mnohé pokročilé operácie. Použitie spočíva v preddefinovaní správania jednotlivých funkcií, ktoré sú schopné premenovať súbor alebo ho zaradiť do priečinku podľa hodnoty vybraného tagu.

- **Automatizácia sťahovania tagov** je základným stavebným kameňom aplikácie. Táto funkcia umožňuje vyhľadať a následne stiahnuť metadáta o skladbe či albume s použitím úplného minima informácií. Celý tento proces je automatizovaný, vďaka čomu je používateľ odtienený od celého postupu vyhľadávania. Mnohé existujúce aplikácie neponúkajú dostatočnú úroveň automatizácie, kedy v prípade častých konfliktov konfrontujú užívateľa, aby ich vyriešil ručne. Toto však netvorí súčasť môjho návrhu. Priorita je kladená výhradne na zabezpečenie čo najvyššej presnosti bez nutnosti užívateľského zásahu.

Kapitola 5

Realizácia a vyhodnotenie výsledkov

Táto kapitola sa venuje nástroju, ktorý bol spolu s touto prácou vyvíjaný. V sekcií 5.1 sú predstavené všetky využité technológie, ktoré umožnili implementáciu a vývoj výslednej aplikácie. Sekcia 5.2 popisuje spôsob implementácie a riešenie jednotlivých problémov. V sekcii 5.3 sa nachádza predstavenie výslednej aplikácie, výsledky testovania a porovnanie rozličných prístupov riešenia.

5.1 Použité technológie

V tejto sekcií sú uvedené hlavné prostriedky, ktoré umožnili vypracovanie programovej časti tejto bakalárskej práce. Znalosť použitých technológií uľahčuje pochopenie implementačných detailov a zdrojového kódu. U každej technológie sú spomenuté základné vlastnosti, dôvody využitia a osobné dojmy z používania.

Výsledná aplikácia je vyvíjaná na platforme Windows v spojení s vývojovým prostredím Microsoft Visual Studio 2017 Enterprise. Ako verzovací systém som použil populárny Git, spolu s internetovým repozitárom GitHub. Vďaka využitým technológiám by nemal byť problém s kompatibilitou ani na unixovým desktopových systémoch.

Python

Python je interpretovaný programovací jazyk, ktorého hlavné výhody sú vysoká kompatibilita a možnosť multi-paradigmového prístupu. Tento jazyk je *dynamický typovaný* a na správu pamäte využíva *garbage collection*. Napriek tomu, že Python sa považuje za skriptovací jazyk, stále je vhodný na menšie až stredne veľké projekty. Jazyk plne podporuje objektovo orientované programovanie, ktoré bolo z určitej časti využité aj v prípade implementovanej aplikácie.

Jedným z hlavných dôvodov výberu tohto jazyka je jeho jednoduchá rozšíriteľnosť v podobe modulov. Vďaka tomu som mohol vo výslednej aplikácií vytvoriť samostatne fungujúce moduly, ktorých služby sú následne využívané v GUI. Ďalšou nespornou výhodou je možnosť využiť knižnice jazyka C/C++. Medzi fundamentálne vlastnosti jazyka Python patria kolekcie, ktoré mi umožnili jednoduchý prístup k veľkému množstvu dát, ich spracovaniu a uchovaniu. Za ďalšie veľké plus vnímam vysokú natívnu podporu vstavaných

knižníc a charakter open-source projektu. V implementácií je použitý jazyk Python verzie 3.6[1].

PyQt5

Užívateľské prostredie aplikácie bolo vytvorené knižnicou PyQt5, ktorá sprístupňuje funkcionalitu Qt pre jazyk Python. Táto knižnica spadá pod licenciu GNU GPLv3, ktorá umožňuje voľné použitie v oblasti nekomerčných open-source projektov. Vytváranie a následné generovanie grafického rozhrania uľahčuje nástroj Qt Designer, ktorý tvorí súčasť balíčka PyQt5. K výberu tejto knižnice viedla široká komunita vývojárov, podpora externých nástrojov a pomerne jednoduchý princíp komunikácie medzi objektmi grafického rozhrania.

Mutagen

Mutagen je modul pre jazyk Python, ktorý sprístupňuje rozhranie k manipulácií s metadátami v hudobných súboroch. Ponúka širokú ponuku podporovaných zvukových formátov a kvalitnú dokumentáciu. Tento modul je taktiež zastrešovaný licenciou GNU GPLv2 alebo novšou. V súčasnosti existuje mnoho nástrojov, ktoré Mutagen aktívne využívajú. Práve toto bol dôvod, prečo som sa rozhodol pre tento modul.

Discogs API

Služba Discogs bola spomínaná v kapitole 4.1. K realizácií aplikácie som využil prostriedky, ktoré táto služba ponúka cez svoje verejné *API (Application Programming Interface)*. Základom Discogs API je webové rozhranie na získavanie dát vo formáte JSON. Na využitie tohto API v klientskej aplikácií, som použil oficiálny modul `discogs_client` pre Python. Tento modul poskytuje prijemné rozhranie, pričom odpovede vracia podobe objektu typu `dict`. Miernou nevýhodou je, že prístup k pokročilým funkciám umožňuje až po autentifikácií používateľa. Medzi tieto funkcie radí aj vyhľadávanie reťazcom, čo je v porovnaní s konkurentami hlavná slabina inak obrovskej (ak nie aktuálne najväčšej) hudobnej databázy.

MusicBrainz API

Služba MusicBrainz taktiež ponúka otvorené API a zároveň modul jazyka Python, ktorý toto API využíva. Hlavné vlastnosti a charakteristiky služby sú popísané v kapitole 4.1. Modul sa nazýva `musicbrainzngs` a ponúka rozhranie vo forme objektu typu `dict`. Rozdiel oproti Discogs spočíva v mierne odlišnej architektúre databázy a používanej terminológií. Medzi výhody, ktoré súvisia s potrebami vytvorenej aplikácie, patrí jednoduchá identifikácia reťazcom bez nutnosti potreby klientskych kľúčov (API keys).

Spotify API

Táto služba je výnimočná tým, že jej zameranie je mierne odlišné od tých predchádzajúcich. Viac detailov o službe ako takej je možné nájsť v kapitole 4.1. Spotify ponúka kvalitné API, ktoré umožňuje čerpať informácie z ich databáz.. V tomto prípade však poskytuje aj dodatočné informácie o technických detailoch jednotlivých skladieb. Vysoká spoľahlivosť a nízka redundancia výsledkov umožňuje jednoduché vyhľadávanie skladieb a interpretov.

[1] https://www.python.org/downloads/release/python-360/

K využitiu tohto API v prostredí jazyka Python som použil oficiálny modul s názvom `spotipy`. Medzi hlavné výhody patrí jednoduchá kompozícia odpovedí zo strany servera, čo značne uľahčuje spracovanie informácií pre potreby vytvorenej aplikácie. Toto API využíva autorizačnú techniku na princípe API kľúčov, ktoré identifikujú klientsku aplikáciu.

AcoustID

Je projekt a služba, ktorá umožňuje identifikovať skladby na základe ich zvukového odtlačku. K vytvoreniu odtlačku využíva nástroj Chromaprint, ktorého princíp bol popísaný v kapitole 4.2. Táto služba využíva prepojenie s MusicBrainz databázou, vďaka čomu môže využívať jedinečné identifikátory entít tejto databázy. K autorizácií sa využívajú klientske API kľúče a odpoveď zo strany servera je typicky vo formáte JSON. Pôvodná implementácia knižnice je v jazyku C++, pričom ja som využil dostupný modul `pyacoustid` pre jazyk Python. Pre optimálnu funkčnosť je odporúčané nainštalovať knižnicu `FFmpeg`, ktorá vykonáva potrebné transformácie medzi rôznymi zvukovými formátmi.

5.2 Implementácia

Táto sekcia popisuje postup implementácie a prístup k riešeniu rozličných problémov. Pre dosiahnutie cieľov popísaných v kapitole 4.3, bolo nutné rozdeliť celkový návrh na menšie časti, ktoré riešia konkrétne problémy. Jednotlivé časti sú rozdelené do osobitných modulov, ktoré ponúkajú samostatnú funkčnosť.

Vytvorená aplikácia nesie názov *TikTag*, podľa čoho sú pomenované aj ostatné moduly aplikácie. Umiestnenie modulov je organizované v osobitných priečinkoch spolu s triedou, ktorá tvorí rozhranie medzi použitými nástrojmi a hlavným programom. Ďalšie detaily a popis modulov je vysvetlený nižšie.

5.2.1 Prístup k metadátam

Prístup k metadátam zvukových súborov bol prvý z problémov, ktorý bolo nutné vyriešiť. V prípade, že by aplikácie nebola schopná pristupovať a modifikovať metadáta v súborových kontajneroch, jej funkcionalita by bola značne obmedzená.

Pri riešení tejto časti som sa zameral na dva najpopulárnejšie zvukové formáty súčasnosti. Medzi tieto formáty patrí stratový formát MP3 a bezstratový formát FLAC, ktoré sú predstavené v kapitole 2.3. Oba formáty sú špecifické svojimi vlastnosťami a stavbou kontajnera, v ktorom sa nachádzajú. Zatiaľ čo formát MP3 využíva tagy ID3, formát FLAC využíva mierne modifikovanú verziu tagov Vorbis comments. Viac informácií o štruktúre jednotlivých tagov je možné nájsť v kapitole 3.

Tieto menšie odlišnosti viedli k vytvoreniu statickej triedy, ktorá je schopná narábať s rôznymi formátmi uniformným spôsobom. Táto trieda slúži ako rozhranie modulu `TikTagCtrl` a nachádza sa v súbore `Tagger.py`.

Funkcionalita tohto modulu je postavená na knižnici `mutagen`, ktorá bola opísaná v predchádzajúcej sekcií. Napriek tomu, že knižnica poskytuje všeobecný prístup k metadátam rôznych formátov, bolo nutné vytvoriť osobitnú triedu pre každý použitý formát. To bolo zapríčinené miernymi odlišnosťami z pohľadu podpory jednotlivých typov tagov a informácií, ktoré tieto tagy obsahujú.

Na rozdiel od Vorbis comments, tagy ID3 sú striktne štruktúrované. To v praxi znamená, že hlbší prístup k týmto tagom je o niečo zložitejší. To platí aj v prípade získavania

technických vlastností zvukového súboru, ktoré sú špecifické pre každý použitý zvukový formát a kodek. Tieto vlastnosti definujú hlavné parametre zvukového súboru, ktoré boli popísané v kapitole 2.

Obrázky

Medzi ďalšiu odlišnosť je možné zaradiť prístup k obrázkom, ktoré sú pripojené k tagu. ID3 tagy sa skladajú z rámcov, ktoré sú určené svojim identifikátorom podľa špecifikácie a kľúčom, ktorý zaistuje jedinečnosť rámca. Pretože rámce a ich štruktúra sa na základe typu odlišujú, knižnica `mutagen` obsahuje pre každý rámec osobitnú triedu. Rámec obrázka má identifikátor `APIC` a okrem dát obsahuje tieto informácie:

- *MIME typ* značí typ kódovania dát obrázka, teda jeho formát.
- *Všeobecný typ* sa využíva na určenie významu obrázka
- *Popis* tvorí ľubovoľná textová informácia, ktorá najčastejšie popisuje daný obrázok.

Jedninečnosť obrázka v ID3 tagoch je tvorený kombináciou identifikátora a popisu oddeleného dvojbodkou. To znamená, že vloženie viacerých obrázkov je možné len v prípade rozdielneho popisu. Tým pádom bolo nutné zaistiť, aby bol užívateľ patrične informovaný o tejto skutočnosti.

Užívateľ je schopný obrázky pridávať, meniť a odstraňovať. Kontrola kompatibility pridávaných obrázkov je v réžii modulu PIL, ktorý je určený na prácu s obrazovými dátami a dokáže ich analyzovať. Vytvorená trieda `AlbumArt`, ktorá je použitá v aktuálne každom podporovanom formáte, ponúka zjednodušené rozhranie pre použitie v GUI.

Obrázky v tagoch typu Vorbis comments majú o niečo jednoduchšiu štruktúru. Ich jedinečnosť nie je potreba riešiť a ich počet je teoreticky neobmedzený. Mierne zložitejšie môže pôsobiť ich pridávanie, ktoré zahrňuje vytváranie nového objektu `Picture`. Obrázky tohto tagu obsahujú väčšie množstvo dodatočných informácií, napríklad údaj o bitovej hĺbke pridávaného obrázka. Z dôvodu využitia rovnakého prístupu, bolo nutné vytvoriť kľúče, ktoré budú aj v tomto prípade jednoznačne identifikovať každý obrázok. Preto sú jednotlivé obrázky automaticky očíslované.

Textové metadáta

Prístup a modifikácia textových metadát je v prípade oboch formátov principiálne rovnaká. Malé odlišnosti spočívajú v rozdielnosti množiny informačných polí, ktoré podporujú. *Informačné polia* tvoria abstrakciu nad výrazom rámec u ID3 tagov a položiek u Vorbis comments.

Zoznam informačných polí je definovaný v triede každého podporovaného formátu. Tento zoznam má tvar objektu `dict`, ktorý tvorí slovník názvov jednotlivých polí. Kľúče tohto slovníka sú názvy tagov reprezentované v GUI, pričom hodnoty predstavujú kľúče využívané knižnicou `mutagen` k prístupu k metadátam. Je nutné podotknúť, že všetky hodnoty sú vo tvare objektu `list`, preto ich treba vedieť pri manipulácií spracovávať osobitne a vracať v preddefinovanom tvare.

5.2.2 Identifikácia skladieb a sťahovanie metadát

Sťahovanie metadát z internetu a identifikácia skladieb je v rámci zdrojového kódu umiestnená v module `TikTagServices`. Tento modul obsahuje triedy, ktoré sú pomocou API schopné komunikovať so službami popísanými na začiatku tejto kapitoly.

Keďže každá služba komunikuje rozličným spôsobom, bolo nutné vytvoriť rozhranie, ktoré bude tento proces dostatočnou mierou abstrahovať. Na tento účel slúži trieda s názvom `OnlineServices`. Táto trieda umožňuje hlavnému programu zavolať všetky služby v jednom momente podľa nastavení používateľa. Odpoveď vo forme získaných metadát je prevedená na objekt typu `dict`. Trieda inicializuje potrebné služby a v prípade potreby rieši autentizáciu.

Spotify

Každá trieda reprezentujúca službu obsahuje zoznam tagov, ktoré je schopná v jednom požiadavku získať zo servera. Služba Spotify ponúka priaznivé prostredie pre získavanie metadát práve kvôli štruktúre ich databázy. Na rozdiel od zvyšku implementovaných služieb, dáta sú centralizované okolo hlavnej entity, ktorú tvorí hudobná skladba. Tento model je vhodný z toho dôvodu, že aj výsledná aplikácia sa sústredí na jednotkové spracovanie skladieb. To vo výsledku znamená, že nie je potrebné poznať celý kontext skladby, ale stačí názov diela a meno umelca. Z tohto dôvodu má trieda `SpotifyDB` veľmi jednoduchú štruktúru. Pri spracovaní odpovede vyhodnotí obmedzený počet záznamov a vyberie najrelevantnejší z nich.

Discogs

Komunikáciu zo serverom Discogs je možné nájsť v triede `DiscogsDB`. Ako jediná služba vyžaduje registrovanie za účelom využitia vyhľadávacích funkcií API. Na tento účel sú vytvorené funkcie, ktoré pri zvolení služby presmerujú užívateľa na stránky Discogs, kde sa musí prihlásiť a vybrať overovací kód. Aplikácia si následne vyžiada kód a pošle ho späť serveru. Tento proces je založený na protokole OAuth**??**, ktorý v jazyku Python využíva rovnomennú knižnicu.

Práca s databázou pre potreby výslednej aplikácie je o niečo zložitejšia. Koncepcia databázy bola spomenutá v kapitole 4.1. Databáza Discogs radí medzi hlavne entity jednotlivé vydania, čo v širšom význame zahrňuje všetky formáty podobné albumu. To však značne sťažuje prístup k informáciám o konkrétnych skladbách. Viacero vydaní, ktoré predstavujú rovnaké nahrávky, tvorí tzv. *hlavné vydanie (master release)*. Hlavné vydanie sumarizuje všetky konkrétne vydania z rozličných štátov a prevedení do jednotného celku. Tento princíp umožňuje získať všeobecné informácie o danom diele bez závislosti na konkrétnom vydaní. Vďaka tomu je možné získať informácie o pôvodnom vídaní albumu či skladby. Nie všetky vydania majú svoje hlavné vydanie, preto s týmto faktom musí aplikácia v každom prípade počítať.

Z programovej stránky je vyhľadávanie vyriešené nasledovne. Primárne sa prehľadávajú databázy hlavných vydaní. Potenciálny neúspech vedie k opätovnému prehľadávaniu, tentokrát však klasických vydaní. Problém môže nastať vtedy, keď metadáta obsahujú názov albumu, ktorý však zo skladbou vôbec nesúvisí. V takom prípade je prehľadávanie vykonané ešte raz, ale bez údaju o albume.

Vyhodnotenie zhody prebieha algoritmom založeným na Levensteinovej vzdialenosti[2], ktorý je dostupný v knižnici `fuzzywuzzy`. Odpoveď zo servera je štruktúrovaná, čo umožňuje porovnať názvy skladieb a umelcov týmto algoritmom. Výsledok zhody závisí na percentuálnej podobnosti zadaného a získaného reťazca. Ak je výsledná zhoda akceptovateľná, metadáta sú spracované a poslané hlavnému programu.

MusicBrainz

MusicBrainz je posledná implementovaná služba. Princíp získavania a spracovania metadát je veľmi podobný ako pri službe Discogs. Služba MusicBrainz však nevyžaduje registráciu na účely vyhľadávania a prístupu k databáze. Trieda `MusicBrainzDB` obsahuje všetky potrebné metódy na prehľadávanie a spracovávanie metadát touto službou. K získaniu potrebných informácií sa využíva trojitý prechod, pričom každý prechod využíva rozdielne informácie k identifikácii skladby.

Treba spomenúť fakt, že táto databáza neobsahuje záznamy o hlavným vydaniach. Napriek tomu však obsahuje informácie o jednotlivých *nahrávkach (recordings)*, ktoré reprezentujú skladby z trochu širšieho uhlu pohľadu. To znamená, že každá skladba musí pochádzať z jednej nahrávky, ale nahrávka môže byť zdrojom viacerých skladieb. Ak je nahrávka upravená alebo je vydaná v nejakom albume, automaticky sa z nej stáva skladba.

V prvom prechode sa skladba identifikuje na základe názvu a mena interpreta. Ako prvé sú prehľadávané záznamy o nahrávkach. Z množiny možných výsledkov sa vyberú také, ktoré sú najstaršie a vyhovujú funkcií, ktorá hodnotí podobnosť reťazcov. V prípade dostatočnej zhody zapíše metadáta do pripraveného slovníka. Následne vyhľadá príslušné vydanie, na ktorej sa skladba nachádza a doplní zvyšné informácie. V prípadnom neúspechu vyhľadá vydanie na základe jeho katalógového čísla, ak sa taká informácia v tagu nachádza. Ak ani táto možnosť neobstojí, skúsi nájsť vydanie podľa názvu albumu a interpreta. Nevýhodou odpovedí serveru MusicBrainz je to, že neobsahujú všetky dostupné informácie o vyhľadávanom objekte v jednej odpovedi. Z tohto dôvodu sú k dispozícií jedinečné identifikátory, ktoré umožňujú v prípade potreby zvyšné informácie dohľadať.

5.2.3 Grafické užívateľské rozhranie

Grafické rozhranie je vyvinuté balíkom Qt, hoci v tomto konkrétnom prípade sa jedná o PyQt5. Využité sú štandardné prostriedky tohto balíka k realizácií čo najprívetivejšieho prostredia. To viedlo k zaužívanému rozloženiu ovládacích prvkov, kde je panel nástrojov situovaný v hornej časti rozhrania. Tento panel obsahuje zoznam hlavných akcií, ktoré je možné kliknutím spustiť.

Akcie sú rozdelené oddeľovačom do skupín podľa zamerania. Akcie môžu byť aktivované iba v prípade, že je to v súčasnom kontexte realizovateľné. V opačnom prípade sú deaktivované, čo dáva používateľovi tušiť, čo všetko môže v danej situácií vykonať. Väčšina akcií pracuje s vyhradeným priestorom, kde sa zobrazuje súborový systém. Tento priestor tvorí trieda `QTreeView`, ktorá slúži na zobrazenie dát v podobe stromovej štruktúry, čím vytvára vhodného súborového prieskumníka.

Pre prácu zo súborovým systémom bola využitá abstrakcia v podobe `QFileSystemModel` modelu, ktorý ponúka priamo knižnica Qt. Pre účely rozšírenia funkcionality, je nutné tento model mierne modifikovať. Vytvorením triedy `MyFileSystemModel` s použitím dedičnosti,

[2] https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance

je možné pridávať ľubovoľné stĺpce k jednotlivým položkám. To je užitočné pri zobrazovaní vlastností a stavu súborov.

Na pravej strane od súborového prieskumníka sa nachádza rozhranie slúžiace na zobrazenie a modifikáciu metadát. Zvolením hudobného súboru z prostredia prieskumníka sa všetky potrebné informácie zobrazia v pravej časti. Táto časť je od prieskumníka oddelená prispôsobiteľným oddeľovačom, ktorého stav je možné ľubovoľne meniť. Metadáta je možné editovať v jednoduchom tabuľkovom prostredí.

Práca s obrázkami je možná po kliknutí na zmenšenú verziu obrázka v pravom hornom rohu. Po vykonaní tohto kroku sa objaví rozhranie, ktoré umožňuje pomocou kontextového menu spravovať všetky obrázky vybranej skladby. Toto rozhranie má podobu zoznamu, kde ku každému obrázku je zobrazený popis vyjadrujúci jeho vlastnosti.

Hlavná logika spravujúca grafické rozhranie a zároveň beh celej aplikácie sa nachádza v súbore `MainWindow.py`. Ten tvorí trieda hlavného okna a definícia funkcií, ktoré obsluhujú chod celého GUI. Základný princíp komunikácie medzi objektami GUI z pohľadu Qt je pomocou tzv. *signálov* a *slotov*. Pri spustení aplikácie je nutné inicializovať potrebné prepojenia, aby bola zaručená bezproblémová komunikácia. Aplikácia implementuje aj spôsob automatického ukladania konfigurácie.

5.3 Vyhodnotenie dosiahnutých výsledkov

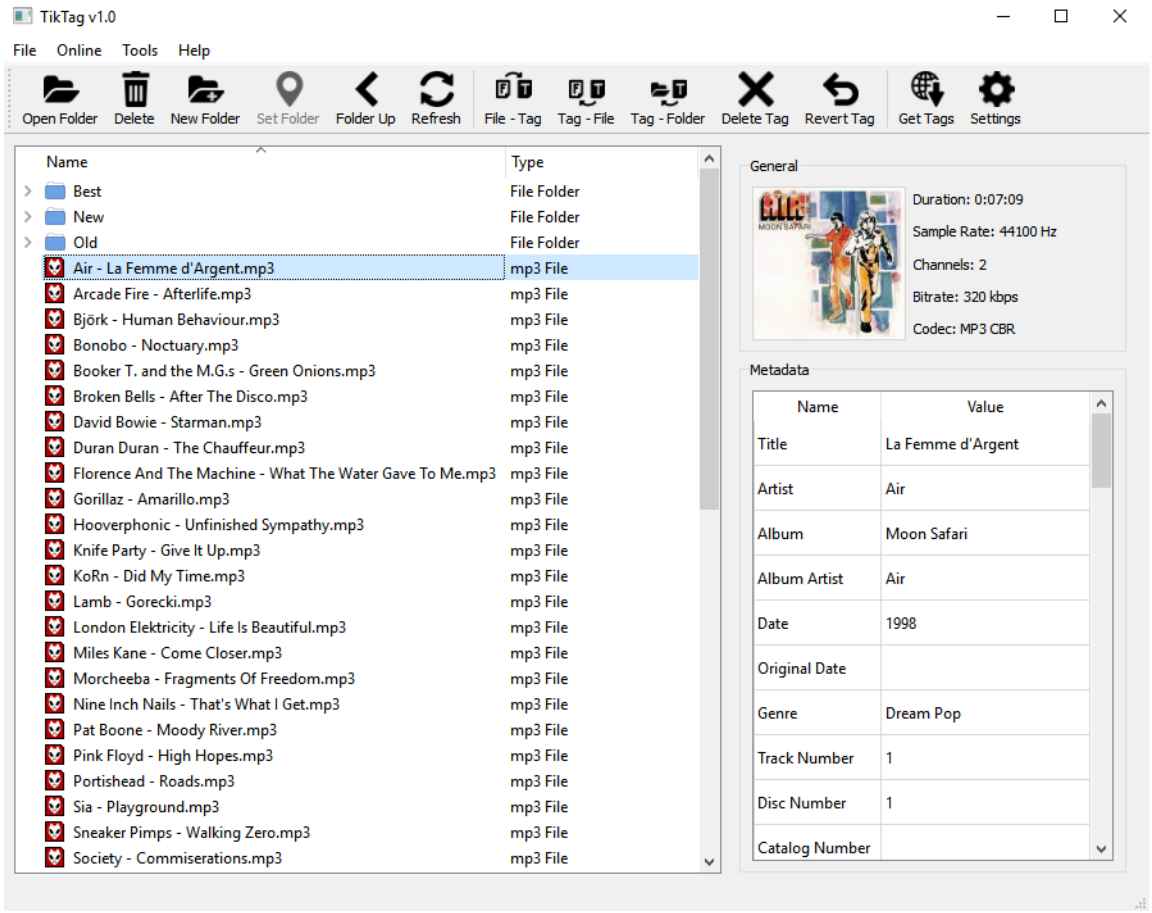
Táto sekcia vyhodnocuje dosiahnutie stanovených cieľov a diskutuje o praktickosti koncového riešenia. Na aplikácií boli vykonávané viaceré testy a experimenty, ktorých výsledky je možné nájsť v sekcií 5.3.2.

5.3.1 Predstavenie vytvoreného nástroja

Nástroj *TikTag* aktuálne podporuje zvukové formáty MP3 a FLAC. Prípadná rozšíriteľnosť je pomerne jednoduchá a spočíva v implementovaní jednotlivých metód v module `TikTagCtrl`. Ako príklad implementácie s použitím knižnice `mutagen`, môžu poslúžiť aktuálne triedy `MP3tag` a `FLACtag`.

Pri prvom zapnutí aplikácie je nutné zadať cestu k adresáru, v ktorom sa nachádzajú hudobné súbory na spracovanie. V tom momente si aplikácia danú polohu zaznamená, aby pri ďalšom spustení pokračovala na tom istom mieste. Podoba hlavného okna je zobrazená na obrázku 5.1.

Prvá skupina ovládacích prvkov, ktorá sa nachádza v panely nástrojov, slúži k manipulácií a navigácií v súborovom systéme. Prieskumník zobrazuje len podporované priečinky a súbory. Vyhradené časti aplikácie podporujú kontextové menu ponúkajúce ďalšiu funkcionalitu.



Obr. 5.1: Grafické rozhranie vytvoreného nástroja TikTag.

Nasledujúca skupina ovládacích prvkov reprezentuje sadu funkcií na manipuláciu s metadátami. Funkcia `File-Tag` umožňuje automatické doplňanie informácií do tagu na základe názvu súboru. Po kliknutí sa zobrazí dialógové okno s konfiguráciou v tvare formátovacieho reťazca. Užívateľ si tak môže nadefinovať predpis, podľa ktorého budú nasledujúce zmeny vykonávané. Z ponuky metadát je možné vybrať práve tie, ktoré sa v názve nachádzajú a automaticky ich presunúť do formy tagu. Náhľad výslednej podoby tagu je znázornený v okne pod týmto výberom. Funkcia `Tag-File` funguje na rovnakom princípe, tentokrát však pomocou metadát mení názvy jednotlivých súborov. Funkcia `Tag-Folder` organizuje súbory podľa zvoleného predpisu do priečinkov. Funkcia `Delete-Tag` odstráni celý tag z vybraných súborov. Funkcia `Revert-Tag` slúži na vrátenie pôvodných informácií po modifikácií tagu.

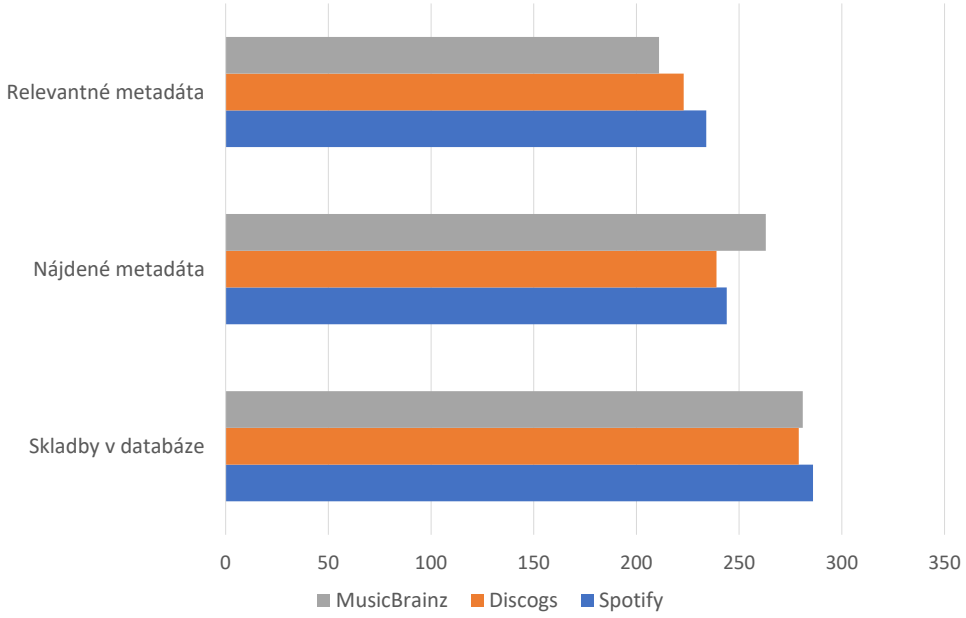
V poslednej skupine ovládačov je funkcia pre získavanie metadát z internetu a nastavenia tohto procesu. Aplikácia umožňuje zvoliť jednotlivé služby alebo využívať informácie z viacerých služieb naraz. Prioritu jednotlivých služieb je možné upraviť v okne *nastavenia*. Toto okno obsahuje aj možnosť aktivácie identifikácie skladieb na základe ich odtlačku alebo zvoliť *režim zapisovania stiahnutých metadát*.

5.3.2 Testovanie

Testovanie je rozdelené na dve časti a zaoberá sa hlavne presnosťou získaných informácií pomocou implementovaných metód. Na účely testovania bola použitá zbierka hudobných súborov rozličného žánru a obdobia. Túto zbierku tvorí 300 skladieb, ktoré boli vybrané s dôrazom na vzájomnú odlišnosť. Testy prebiehali na prenosnom počítači s procesorom Intel i5-7300HQ podporeným 8 GB operačnej pamäte.

Testovanie úspešnosti vyhľadávania jednotlivých služieb

Prvé testovanie spočívalo v použití jednotlivých služieb k vyhľadávaniu informácií o testovaných skladbách. Výsledky testu je možné vidieť v grafe 5.2. Tento graf porovnáva úspešnosť vyhľadávania a zároveň relevantnosť získaných informácií. Pri nedostatku identifikačných prostriedkov, je zo strany aplikácie snaha o získanie metadát, ktoré sú čo „najpôvodnejšie“. Pod týmto výrazom si je možné predstaviť tag, ktorý pozostáva len z názvu skladby a interpreta. V takom prípade však nie je jasné, z ktorého albumu alebo kompilácie skladba naozaj pochádza. Tieto situácie sú riešené hľadaním originálnych prameňov, z ktorých hľadaná skladba pôvodne vychádza. Úspech tohto hľadania je započítaný v rámci výslednej relevantnosti výsledkov. *Skladby v databáze* sú všetky skladby, ktoré boli nájdené na serveroch pomocou webového klienta.



Obr. 5.2: Výsledky testu sťahovania metadát z jednotlivých služieb.

Implementácia služby Spotify bola z pohľadu relevantnosti a presnosti najvydarenejšia. Vďaka koncepcií databáze a inteligentným algoritmom na strane servera, bolo možné dosiahnuť celkom dobrých výsledkov.

Služba Discogs sa v tomto ohľade drží tesne na druhom mieste. Ponúka širokú databázu, pričom chýbalo pár nových titulov. Relevantnosť je na dobrej úrovni vzhľadom na celkový počet nájdených zhôd.

MusicBrainz v tomto teste dosiahlo najviac nájdených zhôd. To je zapríčinené menej striktnou implementáciou, ktorá z dôvodu architektúry databázy, nedokáže pristupovať priamo k jednotlivým skladbám. Princíp je založený na tom, že ak nenájde konkrétnu nahrávku hľadanej skladby, pokúsi sa aspoň dohľadať informácie o albume. Tento prístup zapríčinil pomerne nízku relevantnosť metadát.

	Úspešnosť	Dĺžka trvania
Spotify	78 %	2:31
Discogs	74 %	16:22
MusicBrainz	70 %	5:18

Tabuľka 5.1: Vyhodnotenie testu z hľadiska celkovej úspešnosti a času potrebného na spracovanie.

V tabuľke 5.1 je sumarizácia výsledkov tohto testu. *Úspešnosť* je vyjadrená ako pomer relevantných nálezov ku vyhľadávaným skladbám. Z tejto hodnoty je možné odvodiť mieru automatizácie celého procesu vyhľadávania a sťahovania metadát. Vďaka jednoduchému prístupu k potrebným metadátam je Spotify najúspešnejšie implementovaná služba.

Enormný čas hromadného spracovania je zo strany Discogs spôsobený hlavne obmedzeným počtom HTTP požiadaviek za jednotku času. Tradične sa k vyhľadávaniu jednej skladby môžu použiť až štyri dotazy, čo má za následok nutnosť krátku dobu čakať pred odoslaním ďalšieho požiadavku. Túto nevýhodu vyvažuje rozumná úspešnosť.

MusicBrainz je z pohľadu úspešnosti najnižšie, stále sa však jedná o použiteľné číslo. Trvanie je takmer na úrovni jedného súboru za sekundu. V tomto ohľade je použitie tejto implementácie vhodné najmä vtedy, keď nezáleží na relevantnosti metadát a užívateľ si vystačí aj so všeobecnými informáciami.

Testovanie úspešnosti vyhľadávania po optimalizácií porovnávania

V tomto teste som sa zameral na optimalizáciu metód určených na porovnávanie. Taktiež som využil možnosť použitia viacerých služieb v rámci jedného spracovania súborov. Na tieto účely bola použitá rovnaká hudobná zbierka ako v predchádzajúcom teste.

Výsledok úspešnosti po optimalizácií logiky porovnávania je možné vidieť v tabuľke 5.2. Optimalizácia spočíva v prevedení vyhľadávaných reťazcov do podoby, v ktorej obsahujú len tie najdôležitejšie informácie. Princíp je založený na odstránení nepodstatných slov z názvu skladby a interpreta. Typickým príkladom je anglické slovo *featuring*, ktoré z hľadiska porovnávania tvorí prebytočnú informáciu. Zoznam slov, ktoré sú vo väčšine situácií zbytočné, je určený na základe predchádzajúceho testovania.

	Úspešnosť	Počet nových nálezov
Spotify	83 %	16
Discogs	77 %	13
MusicBrainz	71 %	6

Tabuľka 5.2: Vyhodnotenie testu z hľadiska celkovej úspešnosti a počtu nových nálezov po optimalizácií.

Testovanie úspešnosti pri použití všetkých služieb

Tento test využíva všetky dostupné služby, ktoré sú aktuálne implementované. Spracovanie prebieha v užívateľsky definovateľnom poradí, pričom toto poradie určuje aj prioritu získaných informácií. Test využíva odosielanie odtlačkov aj vtedy, keď s dostupnými informáciami nenájde žiadnu zhodu.

Výsledkom tohto testu je 89% úspešnosť na rovnakej množine zvukových súborov. Tento výsledok považujem za veľmi dobrý a plne dostačujúci z pohľadu praktického použitia. Čas spracovania bol necelých 20 minút, čo však značne predĺžilo využívanie služby Discogs. Nastavenie *režimu sťahovania (download mode)* bolo v stave prepisovania pôvodných metadát. Poradie služieb v teste bolo nasledovné:

1. Spotify
2. Discogs
3. MusicBrainz

Rád by som podotkol, že výsledné namerané hodnoty neodrážajú skutočný potenciál jednotlivých služieb. S týmito hodnotami úzko súvisí cesta, ktorou boli jednotlivé služby implementované v rámci tejto aplikácie. Výsledky sú taktiež značne ovplyvnené logikou vyhodnocovania zhody pri prehľadávaní databáz.

Možnosti rozšírenia

Žiadny software nie je dokonalý. To platí aj v prípade vytvorenej aplikácie. V tejto záverečnej časti by som rád spomenul prípadné rozšírenia a vízie, ktoré sú v rámci budúceho vývoja realizovateľné. Jedná sa hlavne o nové funkcie, ktoré je možné implementovať do aktuálneho riešenia.

- Rozšírenie podpory zvukových formátov.
- Detekcia zvukových súborov s klamlivým bitovým tokom.
- Odosielanie metadát do databáz využitých služieb.
- Zefektívnenie prehľadávania databáz a presnoti výsledkov.
- Lokálna implementácia algoritmu na merianie tempa (BPM).

Kapitola 6

Záver

Táto práca sa zaoberala správou digitálnej hudobnej knižnice a vytvorením nástroja, ktorý tento proces zjednoduší a poskytne istú úroveň automatizácie. Na začiatku som uviedol čítateľa do problematiky reprezentácie zvuku v digitálnej podobe. Tieto fundamentálne základy boli nutné pre pochopenie širších súvislostí. Nasledujúcim logickým krokom bolo predstaviť techniky uchovávania dodatočných informácií v podobe metadát.

Vytvorený nástroj by sa mal odlišovať od existujúcich riešení, preto som pár takých riešení v stručnosti predstavil. Pred samotným návrhom som stručne popísal problematiku rozpoznávania hudobných skladieb na základe odltačku zvukového signálu. Samotný návrh pozostával z určenia cieľov, ktoré má vysledná aplikácia spĺňať. V nasledujúcej časti tejto práce som popísal použité nástroje a služby, ktoré mi umožnili realizáciu a poskytli potrebné informácie.

Postup riešenia bol rozdelený na podproblémy, ktoré som následne popísal v poradí ich riešenia. Vo výsledku sa mi podarilo vytvoriť nástroj, ktorý umožňuje pristupovať k metadátam a lokálne ich upravovať. Nástroj taktiež podporuje automatické sťahovanie metadát a obsahuje logiku, ktorá umožňuje komunikovať s internetovými databázami. Podarilo sa mi vytvoriť jednoduché a intuitívne grafické rozhranie, ktoré uľahčuje prácu s hudobnými súbormi. Dosiahnuté výsledky som prakticky otestoval a vyhodnotil.

Celkový výstup práce hodnotím kladne. Aplikácia dosahuje uspokojivé výsledky z pohľadu presnosti a funkčnosti. Malé nedostatky vidím v odladenosti a podpore formátov. Použitie aplikácie umožňuje prácu so súbormi a metadátami, čo vo výsledku môže uľahčiť prácu nielen dídžejovi.

Literatúra

[1] Bosi, M.; Goldberg, R. E.: *Introduction to Digital Audio Coding and Standards*. Kluwer Academic Publishers, 2003, ISBN 1-4020-7357-7.

[2] Code-greek.com: *How Shazam works*. [Online; navštívené 20.03.2019].
URL http://coding-geek.com/how-shazam-works/

[3] Hans, M.; Schafer, R. W.: *History of Lossless Data Compression Algorithms*. [Online; navštívené 9.01.2019].
URL https://ieeexplore.ieee.org/abstract/document/939834

[4] HydrogenAudio: *APEv2 specification*. [Online; navštívené 13.01.2019].
URL http://wiki.hydrogenaud.io/index.php?title=APEv2_specification

[5] ID3: *ID3v1 and ID3v2 documentation*. [Online; navštívené 17.01.2019].
URL http://id3.org

[6] Lalinský, L.: *How does Chromaprint work?* [Online; navštívené 28.03.2019].
URL https://oxygene.sk/2011/01/how-does-chromaprint-work/

[7] Matus, G.: *Časové okno pro frekvenční analýzu signálů využívající hladkých funkcí*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008, vedoucí bakalářské práce prof. Ing. Pavel Jura, CSc.
URL
https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=55853

[8] Mixedinkey.com: *Official Harmonic Mixing guide*. [Online; navštívené 20.01.2019].
URL https://mixedinkey.com/harmonic-mixing-guide/

[9] Pras, A.; Zimmerman, R.; Levitin, D.; aj.: *Subjective evaluation of mp3 compression for different musical genres*. Presented at the 127th Convention, 2009, October 9–12, New York, NY, USA, [Online; navštívené 20.1.2019].
URL https://www.researchgate.net/publication/
257068576_Subjective_Evaluation_of_MP3_Compression_for_Different_Musical_Genres

[10] Raissi, R.: *The Theory Behind Mp3*. [Online; navštívené 17.01.2019].
URL https://www.mp3-tech.org/programmer/docs/mp3_theory.pdf

[11] Reichl, J.; Všetička, M.: *Kvantování signálu*. [Online; navštívené 5.1.2019].
URL http://fyzika.jreichl.com/main.article/view/1357-kvantovani-signalu

[12] Ruckert, M.: *Understanding MP3*. Friedr. Vieweg and Sohn Verlag, 2005, ISBN 3-528-05905-2.

[13] Sehnal, I. J.: *Digitální záznam zvuku*. [Online; navštívené 5.1.2019].
URL https://coptkm.cz/portal/reposit.php?action=0&id=7459&revision=-1&instance=1

[14] Wang, A. L.-C.: *An Industrial-Strength Audio Search Algorithm*. [Online; navštívené 20.03.2019].
URL https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

[15] Wikipedia.org: *Diskrétní signál*. [Online; navštívené 30.12.2018].
URL https://cs.wikipedia.org/wiki/Diskrétní_signál

[16] Xiph.org: *Ogg Vorbis I format specification*. [Online; navštívené 17.01.2019].
URL https://xiph.org/vorbis/doc/v-comment.html