VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZŠÍŘENÁ MANIPULACE S PLAYLISTY A METADATY HUDEBNÍCH SOUBORŮ V GMPC

EXTENDED MANIPULATION OF MUSIC FILE PLAYLISTS AND METADATA IN GMPC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAKUB ADAMEC**
AUTHOR

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

BRNO 2020

Abstrakt

Cieľom bakalárskej práce je návrh a implementácia klientskej aplikácie pre hudobný server Music Player Daemon (MPD), ktorá umožňuje editáciu lokálnych hudobných súborov a sťahovanie dát z internetových služieb. Súčasne poskytuje špeciálne operácie s playlistmi a získanie tempa zo skladby. Samotnému návrhu aplikácie predchádza teoretický rozbor, ktorý zahrňuje popis metadát, formáty hudobných súborov a rozbor MPD a GMPC. V posledných kapitolách práce je opis riešenia aplikácie zakončený vyhodnotením výsledkov.

Abstract

The aim of the bachelor thesis is to design and supplement a client application for the music server Music Player Daemon (MPD), which allows editing local music files and downloading data from internet services. Along with provide extraordinary operations with playlists and extracts tempos from songs. Before solution of application, we can find theoretical analysis, which includes a description of metadata, music data formats and an analysis of MPD and GMPC. In the last chapters of the work is a description of the application solution ending with the evaluation of the results.

Kľúčové slová

metadáta, hudobné súbory, tagy, python, BPM, playlist, MPD klient

Keywords

metadata, music files, tags, python, BPM, playlist, MPD client

Citácia

ADAMEC, Jakub. *Rozšířená manipulace s playlisty a metadaty hudebních souborů v GMPC.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Vladimír Janoušek, Ph.D.

Rozšířená manipulace s playlisty a metadaty hudebních souborů v GMPC

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pána Doc.Ing. Vladimíra Janouška, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Jakub Adamec
28. mája 2020

Poďakovanie

Touto cestou by som sa rád poďakoval pánovi Doc. Ing. Vladimírovi Janouškovi, Ph.D za pomocné rady, nápady a korekcie pri riešení bakalárskej práce.

Obsah

Kapitola 1

Úvod

Technológia skutočne pomohla zlepšiť životy ľudí. Svoju pôsobnosť rozšírila aj do ďalších oblastí a jednou z oblastí, ktorá získala z technológie veľa výhod, je hudobný priemysel. Tu sa využívajú elektronické zariadenia a počítačový softvér pri prehrávaní, nahrávaní alebo skladaní. Digitálne spracovanie zvuku sa používa v mnohých oblastiach každodenného života, ako sú telefóny, rádio, televízia, videohry... Existuje široké spektrum aplikácií, ktoré riešia manipulovanie s digitálnym zvukom v troch oblastiach: získavanie, zobrazenie a skladovanie.

S digitálnym spracovaním informácií sa rozširuje potreba vyhľadávať informácie o produktoch digitálneho sveta. Fenomén súčasnej doby, internet, sa na tom podieľa nemalou mierou. K veľkej väčšine informácií z internetu získaných sa človek dostáva práve vyhľadávaním. Ak je vyhľadávanie dostatočne efektívne, tak je to najrýchlejší spôsob ako požadovanú informáciu nájsť. V súčasnosti sa však vyhľadávanie nesústredí len na Internete, ale Internet začína byť viac chápaný ako prostredník. K dispozícii sú nespočetné online databázy rôznorodých dát a stále vznikajú ďalšie.

V mojej práci sa venujem problematike metadát hudobných súborov. Túto problematiku riešim v mojej aplikácii, ktorá rozširuje pôvodnu funkcionalitu v prehrávači GMPC, pre ktorý som vytvoril alternatívnu aplikáciu pre MPD klient. Bez metadát by sa hudobný priemysel takmer zastavil. Tieto dôležité informácie sú potrebné na správu hudobných súborov, ktoré umožňujú správne vytváranie, ukladanie, triedenie a použitie údajov v širokej škále aplikácií. Metaúdaje sú ústredným prvkom odvetvia, ktoré spája tvorbu hudby, autorské práva, licenčné poplatky a objavovanie hudby.

Obsah tejto práce je rozdelený do 7 kapitol, ktoré objasňujú princípy a realizáciu pri vytváraní aplikácie. V úvodnej časti, v kapitole 1 je uvedená motivácia výberu témy práce pre spracovanie metadát. Digitalizácia zvuku a jeho kompresia sú spísané v kapitole 2. Postupne si tu čitateľ môže nájsť informácie o spracovaní zvuku a úskaliach jednotlivých hudobných formátov, ktoré sú rozdelené do 3 veľkých kategórií.

V ďalšej kapitole 3 je popísaný úvod do problematiky metadát. Ten obsahuje popis základných pojmov k metadátam hudobných súborov a ich ukladania. Podrobnejšie je tu popísaný prístup k verejným databázam a formáty sťahovaných dát. Kapitola 4 je venovaná vyhodnoteniu funkcionality existujúceho hudobného servera MPD a prehrávača GMPC, vrámci ktorých, bolo mojou úlohou implementovať aplikáciu.

Kapitola 5 sa začína popisom dôležitej veličiny pre dídžejing a vytváranie hudby, tempom. V tejto sekcii je následne uvedený súhrn matematických funkcií, algoritmov pre spracovanie signálu použitých pre výpočet BPM. V následujúcej kapitole 6 som zhrnul návrh aplikácie, implementačné detaily, použité technológie a testovanie. Hlavný dôraz sa kladie

na jednoduchosť a efektívnosť z pohľadu koncového užívateľa. V závere je vyhodnotený prínos práce a možné rozšírenia.

Kapitola 2

Formáty hudobných súborov a ich kompresia

Kompresia zvukových údajov má potenciál znížiť prenosovú šírku pásma a požiadavky na ukladanie zvukových údajov. Algoritmy kompresie zvuku sú v softvéri implementované ako zvukové kodeky. Algoritmy straty zvukovej kompresie poskytujú vyššiu kompresiu za cenu vernosti pôvodného zvuku a používajú sa v mnohých zvukových aplikáciách. Tieto algoritmy sa takmer všetky spoliehajú na psychoakustiku, aby odstránili alebo znížili čistotu menej počuteľných zvukov, čím zmenšujú priestor potrebný na ich uloženie alebo prenos. [14]

Pri stratovej aj bezstratovej kompresii je redundancia informácií znížená pomocou metód, ako je kódovanie, rozpoznávanie vzorov a lineárna predikcia, aby sa znížilo množstvo informácií použitých na znázornenie nekomprimovaných údajov. Základnou myšlienkou akejkoľvek kompresie pre zvuk a obraz je nájsť spôsob, ako reprezentovať dáta, ktoré zaberajú menej miesta. Najdôležitejšou výhodou kompresie údajov je skrátenie času na prenos údajov prostredníctvom komunikačných kanálov.

Keď sa informácie dajú presne získať z bitov, zdrojové kódovanie alebo kompresia sa nazýva bezstratová; v opačnom prípade sa sa jedná o stratovú kompresiu. Aby sa dosiahli vyššie kompresné pomery, stratové algoritmy odstránia informácie, ktoré sa približujú originálu alebo ktoré nie sú vnímateľné.

2.1 Ukladanie zvuku v digitálnej podobe

Najskôr by som chcel objasniť pár spôsobov ako vzorkovať zvukový signál a následne sa pozrieť presnejšie ako vzorkovací proces vplýva na zvuk.

Analógový zvuk tvorí spojitý signál. Digitalizáciou vzniká jeho digitálna podoba a vzniká nespojitý signál. Analógový zvukový signál (signálové napätie) je privedený do takzvaného A/D (analog/digital) prevodníka, ktorý je súčasťou zvukovej karty a ten ho prevedie do digitálnej podoby (na zodpovedajúce čísla). Je samozrejmé, že existuje mnoho spôsobov takéhoto prevodu, a že výsledná kvalita bude závisieť práve na použitom spôsobe, čiže na rozlíšení prevodníka. Charakteristiku rozlíšenie prevodníka sú dôležité dva faktory.

1. Prvým z nich je **vzorkovacia frekvencia**. Toto číslo nám udáva rýchlosť vzorkovania (sampling rate), čiže hovorí, koľkokrát za sekundu počítač zaznamená okamžitú hodnotu analógového signálu.

2. Druhou dôležitou charakteristikou A/D prevodníka je **šírka slova**, čiže vertikálne rozlíšenie. Je to číselné vyjadrenie okamžitej zvukovej hladiny. Presnosť tohto čísla závisí od počtu bitov, ktoré použijeme k jej zapísu. Čím viac bitov, tým viac môžeme zaznamenať zvukových intenzít a zvuk sa potom javí dynamickejší. Pre kvalitný záznam sa používa šestnásť bitová šírka slova, čo dovoľuje rozlíšiť 2^16 čiže 65 536 napäťových úrovní. Táto hodnota dovoľuje zaznamenať zvuk o dynamickom rozsahu 96 dB [2].

Postup digitalizácie

1. **Vzorkovanie** - prevod reálnych vstupných hodnôt na postupnosť diskrétnych reálnych čísel.

2. **Kvantizácia** - prevod postupnosti reálnych čísel na postupnosť celých čísel.

3. **Kódovanie** - spôsob uloženia a kódovania postupnosti celočíselných hodnôt získaných v predošlom kroku.

Shanonov vzorkovací teorém

Shanonov vzorkovací teorém (v literatúre sa vyskytuje pod ďalšími názvami ako Nyquistov teorém) vraví, že analógový signál $s(t)$ možno rekonštruovať z hodnôt vzoriek podľa predpisu 2.1 práve vtedy, keď je hodnota vzorkovacej frekvencie aspoň dvojnásobkom najvyššej frekvencie obsiahnuté vo vstupnom signálu 2.2. Ak je vzorkovacia frekvencia menšia dochádza ku skresleniu zložiek vyšších frekvencií.

$$x(t) = \sum_{n=-\infty}^{\infty} x_n \frac{\sin(\pi(\frac{t}{T} - n)}{\pi(\frac{t}{T} - n))} \tag{2.1}$$

$$f_v = 2f_{max} \tag{2.2}$$

2.2 Nekomprimované hudobné súbory

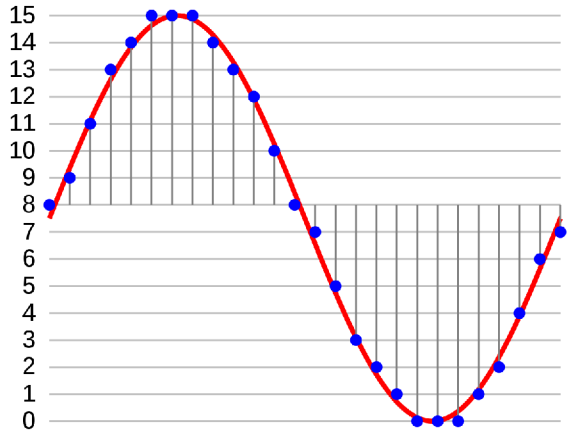
Nekomprimované formáty zvukových súborov dokážu zachytiť a previesť skutočné zvukové vlny do digitálneho formátu. Nie sú potrebné žiadne ďalšie činnosti spracovania. Konečným výsledkom použitia tohto formátu je, že ponúkajú najpresnejšie interpretácie. Nakoniec však zaberajú väčšie množstvo priestoru v porovnaní s inými formátmi. Siahajú okolo 34 MB za minútu pre jednoduchý 24-bitový 96 KHz stereo zvukový súbor.

Medzi najznámejšie formáty z tejto kategórie sa radia:

- PCM
- WAV

PCM

PCM je skratka pre Pulzna kodova modulácia, obsahuje digitálne zobrazenie pôvodných(raw) analógových zvukových signálov. Analógové zvuky existujú ako vlnový priebeh a aby sa priebeh konvertoval na digitálne bity, musí sa zvuk vzorkovať a zaznamenávať v určitých intervaloch (alebo impulzoch) [17].PCM je najbežnejší zvukový formát používaný na CD a DVD. Existuje podtyp PCM nazývaný LPCM lineárna pulzná kódová modulácia, kde sa vzorky odoberajú v lineárnych intervaloch.



Obr. 2.1: Vzorkovanie a kvantovanie signálu pre 4-bitový lineárny PCM

WAV

WAV je skratka pre Waveform Audio File Format. Je to štandard, ktorý vyvinuli spoločnosti Microsoft a IBM v roku 1991. WAV je vlastne kontajner Windows pre rôzne zvukové formáty. To znamená, že súbor WAV môže obsahovať komprimovaný zvuk, ale na tento účel sa veľmi zriedka používa.

WAV je založený na **RIFF** (Resource Interchange File Format), čo je najpoužívanejší štandardizovaný formát uloženia dát v PC. RIFF pracuje s blokmi, takzvanými chunks. Každý blok má svoju hlavičku, ktorá obsahuje informácie o ňom (veľkosť, typ) a potom vlastnú dátovú časť. Bloky možno do seba vnorovať, vytvárať z nich stromovú štruktúru. 2.1.

2.3 Stratové komprimované hudobné súbory

V informačných technológiách je stratová kompresia, trieda metód kódovaných údajov, ktoré na svoju reprezentáciu obsahu používajú nepresné aproximácie a čiastočné vyradenie

Veľkosť	Typ
4 Byty	Identifikátor ASCII pre tento blok
4 Byty	Dĺžka tohoto bloku, little-endian
Pole s nestálou veľkosťou	Samotné dáta o veľkosti z predošlého bloku
Prázdny Byte	Ak by celková veľkosť nebola párna

Tabuľka 2.1: Riff Formát

údajov. Tieto techniky sa používajú na zníženie veľkosti údajov na ukladanie, spracovanie a prenos obsahu. Je to v protiklade s bezstratovou kompresiou údajov, ktorá údaje neznižuje.

Dobre navrhnutá technológia stratovej kompresie často značne zmenšuje veľkosť súborov. Aj keď si to používateľ všimne, môže byť žiaduce ďalšie zníženie údajov (napr. v prípade komunikácie v reálnom čase, skrátenia doby prenosu alebo zníženia náročnosti ukladania) [19]. Najčastejšie používaným stratovým kompresným algoritmom je diskrétna kosínusová transformácia (DCT).

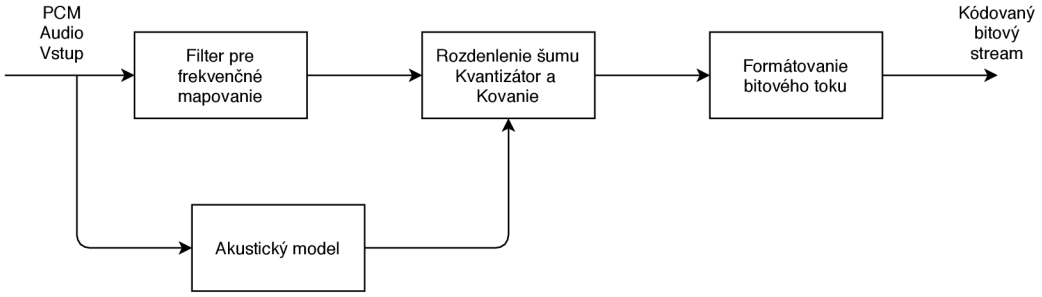
K populárnym stratovým hudobným formátom patria:

- MP3
- AAC
- OGG Vorbis

MP3

MP3 (MPEG-1 Audio Layer III, MPEG-2 Audio Layer III) je kódovací formát pre digitálny zvuk a bol vyvinutý v roku 1993. MP3 súbor, ktorý je vytvorený s konštantnou bitovou rýchlosťou 128 kbit/s zaberá približne len jedenástinu toho, ako súbor rovnakého zvuku uložený v bezstratovom zvukovom formáte WAV. MP3 súbor je možné vytvoriť s vyššou alebo nižšou bitovou rýchlosťou, z čoho vyplýva aj vyššia alebo nižšia kvalita. 128 kbit/s predstavuje stupeň kompresie, pri ktorej laik zvyčajne nerozpozná rozdiel či ide o komprimovaný formát mp3 alebo súbor uložený bez strát [6]. Algoritmus kódovania MP3 je zvyčajne rozdelený do štyroch častí.

1. Rozdelenie zvukového signálu na menšie časti, ktoré sa nazývajú rámce, a na výstupe sa potom vykoná filter s modifikovanou diskrétnou kosínusovou transformáciou (MDCT).
2. Prechádzanie vzorkov do 1024-bodovej rýchlej Fourierovej transformácie (FFT), potom sa použije psychoakustický model a na výstupe sa vykoná filter.
3. Kvantifikuje a kóduje každú vzorku, známu ako rozdelenie šumu, ktorá sa prispôsobuje tak, aby vyhovovala požiadavkám na bitovú rýchlosť a maskovanie zvuku.
4. Formátuje bitový tok nazývaný zvukový rámec, ktorý sa skladá zo 4 častí, hlavičky, kontroly chýb, zvukových údajov a pomocných údajov.



Obr. 2.2: proces kompresie MP3

AAC

AAC je skratka pre Advanced Audio Coding. Bola vyvinutá v roku 1997 ako nástupca MP3. Algoritmus kompresie, ktorý používa AAC, je oveľa vyspelejší a technickejší ako MP3. V súčasnosti ide o štandardnú metódu kompresie zvuku, ktorú používajú služby YouTube, Android, iOS, iTunes.

OGG Vorbis

Ogg je názov formátu pre free open source stream kontajner, teda nie je zaťažený patentom. Ogg je vyvíjaný a udržiavaný nadáciou Xiphl. Vorbis je názov zvukového stratového formátu Ogg. Formát Ogg používa na kódovanie údajov štandardné metódy, ktoré vytvárajú malú veľkosť súboru s dobrou kvalitou zvuku v porovnaní s inými stratovými formátmi. Pre porovnanie pri 110 kb/s, ktoré poskytuje formát ogg nižšej veľkosti znie lepšie než MP3 pri 128 kb/s.

2.4 Bezstratové komprimované hudobné súbory

Bezstratové komprimovanie súborov patria medzi triedu algoritmov kompresie údajov, ktorá umožňuje dokonale rekonštruovať pôvodné údaje z komprimovaných údajov. Medzi najznámejšie bezstratové komprimované formáty patrí FLAC.

FLAC

FLAC je skratka pre Free Lossless Audio Codec, audio formát podobný MP3, s tým rozdielom že zvuk je komprimovaný v FLAC bez straty kvality. Je to podobné tomu, ako zip funguje, s výnimkou systému FLAC získate oveľa lepšiu kompresiu, pretože je navrhnutý špeciálne pre zvuk [8].

Kapitola 3

Problematika metadát

3.1 Metadata

Metadáta sú neoddeliteľnou súčasťou digitálneho uchovávania informácií. Súbory bez vhodných metaúdajov nie sú ľahko zrozumiteľné, interpretovateľné alebo spracovateľné. Efektívne neexistuje uchovávanie ani zmysluplný prístup ku skladbám bez metadát. V systémoch s hudobnými súbormi sa metadáta môžu ukladať pomocou jednej z dvoch primárnych metód [12].

- **Metadáta uložené externe pre digitálny objekt**, napríklad do databázy pomocou súboru XML. Tento externý záznam musí byť nevyhnutne spojený so zdrojovým objektom prostredníctvom jedinečných identifikátorov, ciest pre súbory alebo iných prostriedkov.
- **Metadáta môžu byť vložené do samotného objektu**. Vnorené metaúdaje možno najjednoduchšie definovať ako metaúdaje, ktoré sú uložené v rovnakom súbore alebo kontajneri, pre ktoré sa metadáta vzťahujú.

Ako príklad metadát môžem uviesť textový popis konkrétneho záznamu alebo albumu, ako je meno interpreta alebo skladby, žáner, kľúč, tempo, trvanie, ... Za metaúdaje považujeme aj súvisiace médiá, ako napríklad obaly, hudobné klipy a hudobné udalosti. Webový obsah, sociálne siete, hry, Music API a CloudServices sú všetky faktory, ktoré vedú k rastu v produkcii hudobných metadát a oblasti založenej na jednej základnej technológii, na internete [16].

3.2 ID3 tag

Aj keď formát MP3 dokázal veľmi efektívne komprimovať zvukové súbory bez výrazného zhoršenia kvality, nebolo možné ukladať textové informácie. Na tento účel bola na konci

súboru zavedená 128-bajtová značka(tag) s pevnou veľkosťou. Pôvodný štandard pre označovanie digitálnych súborov vyvinul Eric Kemp v roku 1996 a vytvoril pojem ID3. Názov vznikol spojením slov Identify a MP3 [11].

Existujú hlavné 2 verzie ID3.

- ID3v1
- ID3v2

ID3v1

Značka ID3v1 zaberá 128 bajtov, začínajúc reťazcom TAG. Značka bola umiestnená na koniec súboru, aby sa zachovala kompatibilita so staršími prehrávačmi médií. ID3v1 bohužiaľ nebol šikovný spôsob ukladania textových informácií. Podporoval iba niekoľko polí informácií a tieto boli obmedzené na 30 znakov. Ďalšou nevýhodou bolo, že keďže značka bola umiestnená na koniec, informácie sa nemohli načítať pri streamovaní súboru. Vzhľadom na tieto nevýhody bola vydaná druhá zložitejšia verzia, ID3v2.

Pole	Dĺžka
Hlavička	3
Názov	30
Umelec	30
Album	30
Rok	4
Komentár	30
Žáner	1

Tabuľka 3.1: Štruktúra ID3v1 s veľkosťou 128B

ID3v2

Neskôr sa vytvorila novú špecifikácia s názvom ID3v2. Hoci má názov ID3, jeho štruktúra je veľmi odlišná od ID3v1. Značky ID3v2 majú premenlivú veľkosť a zvyčajne sa vyskytujú na začiatku súboru. Značka ID3 je zameraná hlavne na súbory kódované formátom MP3 ale môžu pracovať s inými typmi kódovaného zvuku alebo ako samostatný formát pre audio metadáta. Značky ID3v2 pozostávajú z niekoľkých rámcov, z ktorých každý obsahuje časť metaúdajov. Rámce môžu mať dĺžku až 16 MB, zatiaľ čo celková veľkosť značky je obmedzená na 256 MB. ID3v2 má ďalšie čiastkové klasifikácie do 3 verzií s malými zmenami v rámcoch medzi nimi [13]:

- ID3v2.2
- ID3v2.3
- ID3v2.4

ID3v2/file identifier	ÏD3"
ID3v2 version	$03 00
ID3v2 flags	%abc00000
ID3v2 size	4 *

Tabuľka 3.2: Hlavička značky ID3v2, ktorá by mala byť prvou informáciou v súbore má dĺžku 10 bajtov

3.3 Prístup k verejnej databáze

Architektúra služby **MusicBrainz** sa riadi zásadami návrhu **REST**. Interakcia s webovou službou sa vykonáva pomocou protokolu **HTTP** a všetok obsah sa poskytuje v jednoduchom, ale flexibilnom formáte **XML**. Rovnako je webová služba k dispozícii aj vo formáte **JSON** [3].

HTTP METÓDY	Operácie, ktoré manipulujú s údajmi
POST	Požiadavka vedie k vytvoreniu nového prostriedku
GET	Načíta reprezentáciu údajov v tele odpovede
PUT	Existujúci prostriedok upraví podľa obsahu teľa odpovede
PATCH	Len čiastočná úprava zdroja, narozdiel od PUT
DELETE	Vymaže zvolené prostriedky

Tabuľka 3.3: Nasledujúca tabuľka ukazuje, ako sú metódy HTTP používané,vrámci REST API

HTTP je protokol definujúci požiadavky a odpovede medzi mojou klientskou aplikáciou a serverom poskytujúcim verejnosti údaje o metadátach. HTTP klient zvyčajne začne požiadavku nadviazaním TCP spojenia na určenom porte vzdialeného stroja (štandardne port 80). HTTP server počúvajúci na danom porte čaká, kým klient pošle reťazec s požiadavkou ako `GET / HTTP/1.1`, nasledovaný sériou hlavičiek podobných formátu MIME opisujúcich detaily požiadavky a nasledovaných ľubovoľnými údajmi. Po prijatí požiadavky server pošle reťazec napríklad s odpoveďou ako `200 OK` nasledovanou hlavičkami spolu so samotnou správou, ktorej telo tvorí obsah požadovaného súboru, textová informácia alebo chybové hlásenie. Medzi najčastejšie HTTP stavové kódy patria [10]:

- 200 OK
- 201 Created
- 202 Accepted
- 400 Bad Request
- 403 Forbidden
- 404 Not Found
- 502 Bad Gateway

XML

Extensible Markup Language(XML) ktorý definuje súbor pravidiel na kódovanie dokumentov v textovom formáte založenom na **SGML - Standard Generalized Markup Language** . Značky XML identifikujú údaje a používajú sa na ukladanie a organizovanie údajov, narozdiel od HTML, ktoré sa používa na zobrazenie údajov [7].

Ako príklad v jazyku XML som uviedol záznam skladby v databaáze. Z tohto príkladu je patrné veľké množstvo úvodných značiek a redundantných informácií v nich obsiahnutých. Preto sa častejšie prechádza na jazyk JSON, ktorý je redundancie zbavený.

```
<metadata xmlns="http://musicbrainz.org/ns/mmd-2.0#">
    <artist-list>
        <artist id="455641ea-fff4-49f6-8fb4-49f961d8f1ad">
            <name>U2</name>
        </artist>
    </artist-list>
    <recording-list>
        <recording id="c410a773-c6eb-4bc0-9df8-042fe6645c63">
            <title>One</title>
            <track>5</track>
        </recording>
    </recording-list>
</metadata>
```

Výpis 3.1: Ukážka XML záznamu v hudobnej databáze

JSON

JavaScript Object Notation (JSON) je textový formát pre serializáciu štruktúrovaných údajov. JSON môže reprezentovať štyri primitívne typy.

- Integer
- Number
- Boolean
- Null

Ďalej repreznetuje 2 štrukturované typy.

- Object
- Array

JSON ukladá všetky svoje údaje v mapovom formáte (kľúč : hodnota), ktorý je ľahšie pochopiteľný. Dá sa povedať, že JSON pomaly nahrádza XML z dôvodu niekoľkých výhod, ako je napríklad jednoduché modelovanie údajov alebo mapovanie priamo na doménové objekty a ľahko pochopiteľná štruktúra. [9].

```
{
    id: "5b11f4ce-a62d-471e-81fc-a69a8278c7da",
    name: "Nirvana",
    sort-name: "Nirvana",
    type-id: "e431f5f6-b5d2-343d-8b36-72607fffb74b",
    type: "Group",
    disambiguation: "90s US grunge band",
    gender: null,
    gender-id: null,
    country: "US"
}
```

Výpis 3.2: Ukážka záznamu vo formáte JSON

3.4 Nadštandardné metadáta

Popri klasických metadátach ako sú Umelec, Názov skladby, Album, Žáner,Dátum vydania, Vydavateľ môžme pristúpiť aj k iným metadátam.

AcoustID

Každá záznamová stránka vo verejných databázach má záložku Fingerprint, v ktorej je uvedený zoznam AcoustID hodnôt priradených k záznamu. AcoustID je webová služba na identifikáciu hudobných záznamov založená na algoritme akustického odtlačku prstov Chromaprint. Ak znejú dva súbory rovnako pre ľudské ucho, ich akustické odtlačky prstov by sa mali zhodovať, aj keď ich binárne zobrazenia sú dosť odlišné. Akustické odtlačky prstov nie sú hašovacie funkcie, ktoré musia byť citlivé na akékoľvek malé zmeny v údajoch. Akustické odtlačky sú viac podobné ľudským odtlačkom prstov, pretože sú tolerované malé variácie, ktoré nie sú významné pre vlastnosti, ktoré používa odtlačok. Je možné si predstaviť prípad rozmazaného ľudského odtlačku prsta, ktorý sa dá presne porovnať s inou vzorkou odtlačkov prstov v referenčnej databáze. Akustické odtlačky prstov fungujú podobným spôsobom.

Popularimeter

Ako z názvu vyplýva, táto entita zobrazuje obľúbenosť skladby. Takmer všade sa môžme stretnúť s hodnoteniami v rozmedzí 1-5 hviezdičiek, ale formát bol prvotne navrhnutý pre hodnoty od 1 po 255.

Nálada

Poskytuje užitočné informácie, ktoré charakterizuje emóciu skladby. Užívateľovi dokáže urýchliť výber skladby.

Komentár Zväčša komentár obsahuje zaujímavosť alebo vystihuje popis objektu, ktorému je komentár určený. Tzv. komentáre disambiguácie sú polia v databáze, ktoré slúžia na rozlíšenie identicky menovaných umelcov, značiek a iných subjektov.

ISRC

Medzinárodný štandardný záznamový kód, skrátene ISRC, je identifikačný systém pre zvukové a hudobné videozáznamy. Používa sa na priradenie jedinečného identifikátora každému samostatnému zvukovému záznamu. ISRC identifikuje konkrétny zvukový záznam, nie samotnú skladbu. Rôznym záznamom, úpravám, remixom a remastérom tej istej skladby bude preto pridelený vlastný ISRC.

Bitová rýchlosť

Bitová rýchlosť (bit rate) udáva, aký objem informácie sa prenesie za jednotku času. Základnou jednotkou bitovej rýchlosti je bit za sekundu (b/s).

Prenosová rýchlosť [kbit/s]	Oblas
4	Minimum pre kódovanie reči
8	Telekomunikačné zariadenia
128	Kvalita CD
192	Bežne pre MP3
320	Najvyššia úroveň pre MP3

Tabuľka 3.4: Porovnanie prenosovej rýchlosti v rôznych sférach zvuku

Obrázkové rozšírenia

Z verejných databázy sú u mnohých albumov a interpretov dostupné obrazové dáta. Užívateľ má prístup k obrázkom pre prednú a zadnú stranu obalu albumu, leták, umelca, kapelu, predstavenie alebo obrázok z videozáznamu.

Kapitola 4

Zhodnotenie súčasného stavu MPD a GMPC

4.1 MPD

Music Player Daemon (MPD) umožňuje prehrávať hudbu vo forme súborov MP3, Ogg Vorbis, FLAC, AAC, Mod alebo WAV prostredníctvom sieťového pripojenia sa ich prehrávanie môže riadiť pomocou klientskych aplikácií. Hudba sa spravuje v centrálnej databáze a prehráva sa na zvukovej karte servera. Klientske počítače riadia výstup a môžu spravovať zoznamy skladieb na serveri [4]. Pretože MPD tiež podporuje streamovanie ako zvukový výstup, hudbu je možné vysielať nie len na serveri, ale aj na iných počítačoch (klientoch).

Ako klient môžu byť použité nezávislé programy pre rôzne desktopové prostredia a príkazový riadok, ktoré pripomínajú bežné zvukové prehrávače. K dispozícii sú však aj webové klientske rozhrania a aplikácie pre mobilné telefóny. Taktiež umožní ovládať hudbu, ktorá sa aktuálne prehráva cez MPD zo všetkých počítačov v dome.

4.2 Vyhodnotenie jednotlivých klientov

MPD beží na pozadí a prehráva hudbu zo svojho zoznamu skladieb. Klientske programy komunikujú s MPD za účelom manipulácie s prehrávaním, zoznamom skladieb a databázou [1].
MPD používa databázu na udržiavanie základných informácií o hudobných súboroch, keď nie je spustená. Po spustení démona sa databáza uchová úplne v pamäti a na vyhľadávanie alebo vyhľadávanie miestnych zvukových súborov nie je potrebný prístup na pevný disk. Hudobné súbory sa zvyčajne musia nachádzať pod koreňovým adresárom hudby a do databázy sa pridajú až po odoslaní príkazu na aktualizáciu na server.
Klienta je možné si vybrať z množstva dostupných open source programov, napríklad:

- Konzoloví klienti
  - mpc - stabilný, jednoduchý mpd client v jazyku C
  - ncmpcpp - napísaný v C++, jeho predchodcom je ncmpc
  - pms - praktický hudobný vyhladávač je interaktívny klient napísaný v jazyku Go. Jeho rozhranie je praktické a ľahko konfigurovateľné, velmi podobné ako u Vim.
- Utility klient
  - MPD_sima - Neinteraktívny klient pre automatické fronty. Zaradí do fronty nové skladby, ktoré nasledujú po návrhoch podobných umelcov.
- Weboví klienti
  - netjukebox - flexibilné zdieľanie médií - netjukebox je webový mediálny jukebox
  - MPD.FM - Webový klient pre mobilné telefóny zameraný na prehrávanie internetových rozhlasových staníc
  - ympd - webové GUI napísané v C, využívajúce sockety a Bootstrap/JS
- Grafickí klienti
  - Cantata - jednoduchý Qt klient
  - CoverGrid - zaemriava sa na albumy namiesto jednotlivých skladieb
  - GMPC - gtk klient pre hudobné súbory

4.3 Obsluha MPD

Obsluha serveru je veľmi jednoduchá. Údaje pre konfigúracie sú uložené v súbore `~/etc/mpd.conf`. Táto konfigurácia je vhodnejšia viac pre audio server s viacerými užívateľmi so zdieľanou inštanciou MPD. Po nastavení týchto základných konfiguračných údajov bude možné počúvať hudbu pomocou jedného z klientov.

- `pid_file` - súbor v ktorom MPD načíta ID procesu
- `db_file` - súbor pre konfigurovanie databázy
- `state_file` - štatistika o súčasnom stave MPD
- `playlist_directory` - priečinok pre ukladanie playlistov
- `music_directory` - priečinok pre hudbu z lokálneho adresára
- `sticker_file` - súbor pre označenia

Komunikácia s klientom

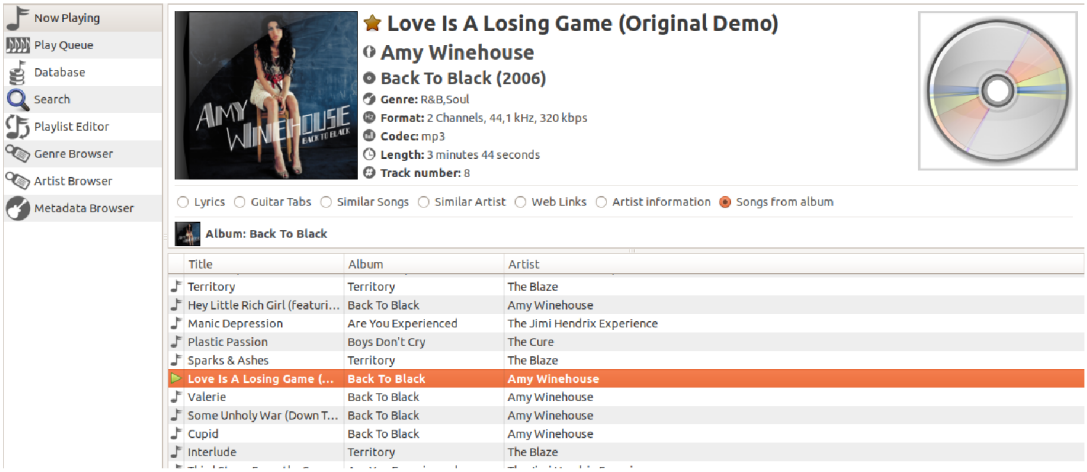
Nastavenie *bind_to_address* špecifikuje ktorým adresám MPD naslúcha pre pripojenia od klientov [5]. Môže sa použiť viackrát na naviazanie viacerých adries, a k tomu uviesť port mimo rozsah globálnych portov.

```
bind_to_address "192.168.1.42:6874"
bind_to_address "127.0.0.1:6602"
```

4.4 GMPC

GMPC je GTK hudobný prehrávač. Vydáva sa pod licenciou GNU General Public License a je to slobodný softvér. Je navrhnutý tak, aby bol jednoduchý, pričom poskytuje plný prístup ku všetkým funkciám MPD. Používateľom je ponúkané niekoľko rôznych spôsobov prehliadania ich hudby. GMPC je najstarší grafický klient pre MPD, ktorý začal v roku 2003 [15].

K dispozícii je tiež prehliadač metadát, v ktorom sa dajú prezerať údaje, ako napríklad celkový čas prehrávania hudby skupiny alebo zoznam všetkých albumov. GMPC je možné rozšíriť pomocou pluginov, ktoré dokážu načítať napríklad obaly albumov a fotografie kapiel z pevného disku, alebo z weboje databáze.



Obr. 4.1: Ukážka rozhrania prehráča GMPC

Dostupné funkcie

GMPC je optimalizovaný pre prácu na zariadeniach nižšej triedy a na pomalších sieťach. Obsahuje veľa skvelých funkcií nad rámec bežných hudobných prehrávačov.

- Prehliadač súborov.
- Prehliadač založený na informáciách ID3.
- Vyhľadávanie.
- Aktuálny prehliadač zoznamov skladieb s vyhľadávaním.
- Informácie ID3.
- Podpora metadát.
- Môže zobrazovať obrázok interpreta, obrázok albumu.

- Môže zobrazovať texty piesní pokiaľ sú dostupné.
- Podpora profilu.
- Podpora pre načítanie / ukladanie zoznamov skladieb.
- Podpora zobrazovania a nastavenia nespracovaných obrazových údajov (raw images).
- Štatistika servera.
- Poskytovanie textov.
- Gitarové tabuľky.

Príklady chýbajúcich funkcií

- Editor hudobných metadát
  - Aj keď dokáže zobrazovať metadáta, tak prehrávač nevie editovať tieto údaje.
- Ekvalizér
- Detailnejšia práca s playlistmi
  - Je ponúknutá len možnosť mať skladby v play-liste. Iná funkcionalita s playlistmi nie je dostupná.

Kapitola 5

Detekcia Tempa

Tempo je užitočným parametrom hudby, ktorá je žiadaná ako od hudobných interpretov, tak od poslucháčov. V tejto kapitole sú zhrnuté poznatky pre popis tempa, funkcie pre spracovanie signálu, ktoré slúžia pre získanie hodnoty tempa a popis implementácie.

5.1 Tempo

Tempo v hudbe označuje rýchlosť striedania jednotlivých dôb za určitú časovú jednotku (obyčajne za minútu), a teda aj rýchlosť samotnej hudobnej skladby. Väčšinou sa uvádza na začiatku skladby pomocou zaužívaných talianskych výrazov. Okrem tohto relatívneho určenia môžeme tempo upresniť číslom, ktoré udáva presný počet dôb za minútu, tzv. **BPM**, čo je skratkou anglického beats per minute. Napríklad pri predpise 60 BPM pripadá na každú sekundu práve jedna doba.

Tempo	**BPM**
Larghissimo	24 a menej
Grave	25-45
Lento	45-60
Larghetto	60-66
Adagio	66-76
Adagietto	70–80
Andante	76-108
Moderato	108–120
Allegro moderato	112–124
Allegro	120–156
Vivace	156–170
Presto	170–200
Prestissimo	200 a viac

Tabuľka 5.1: Opis hodnôt tempa, od najpomalšieho k najrýchlejšiemu

5.2 Diskrétna Fourierova transformácia

Diskrétna Fourierova transformácia je ekvivalentom kontinuálnej Fourierovej transformácie, známej iba v N úsekoch oddelených vzorkovacím časom T.

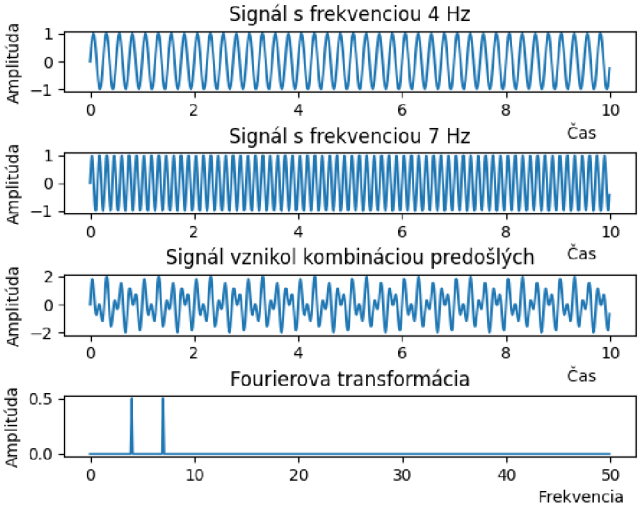
Fourierova transformácia slúži k tomu, aby sme získali spektrum signálu $x(t)$ vo frekvenčnej oblasti $X(\omega)$. Túto transformáciu definujeme vzťahom 5.1

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt, \qquad \omega \in (-\infty, \infty). \tag{5.1}$$

Inverzná Fourierova transformácia je definovaná podľa predpisu 5.2:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega, \qquad t \in (-\infty, \infty). \tag{5.2}$$

Fourierova transformácia pri spracovaní signálov slúži na transformáciu z časovej oblasti, do oblasti frekvenčnej. Je vyjadrením časovo závislého signálu pomocou **harmonických signálov**, t. j. funkcií sínus a kosínus. Signál môže byť buď v spojitom alebo diskrétnom čase.



Obr. 5.1: Výsledkom je Fourierova transformácia signálu, ktorý vznikol skombinovaním 2 jednoduchých signálov

Na druhej strane **diskrétna Fourierova transformácia** nahrádza nekonečný integrál konečnou sumou 5.3.

$$X(\omega_k) \triangleq \sum_{n=0}^{N-1} x(t_n)e^{-j\omega_k t_n}, \qquad k = 0, 1, 2, \ldots, N-1 \tag{5.3}$$

N	počet vzoriek
n	poradové číslo vzoriek
ω_k	jednotlivé body spektra

Vzorkovacie body značíme podľa predpisu:

$$\frac{2\pi}{N}k \qquad k = 0, 1, 2, \ldots, N-1 \tag{5.4}$$

Takto získané spektrum je periodické s periódou $T = 2\pi$, ktorú sme rovnomerne rozložili na N vzoriek podľa vzťahu 5.4 na intervale $(0, 2\pi)$, resp. $(-\pi, \pi)$

5.3 Hannovo okno

Obecne **časové okná** sa využívajú k vylepšeniu výsledkov diskrétnej Fourierovej transformácie. Aplikovanie okna sa robí vynásobením pôvodného signálu s matematickým výrazom okna. Toto násobenie je možné aj v spojitom tvare, ale prakticky sa to nepoužíva.

Funkcia okna je matematická funkcia, ktorá zobrazuje nulu s hodnotou mimo nejakého vybraného intervalu, zvyčajne symetrickú okolo stredu intervalu a blízko maxima v strede, a zvyčajne sa zužuje smerom od stredu.

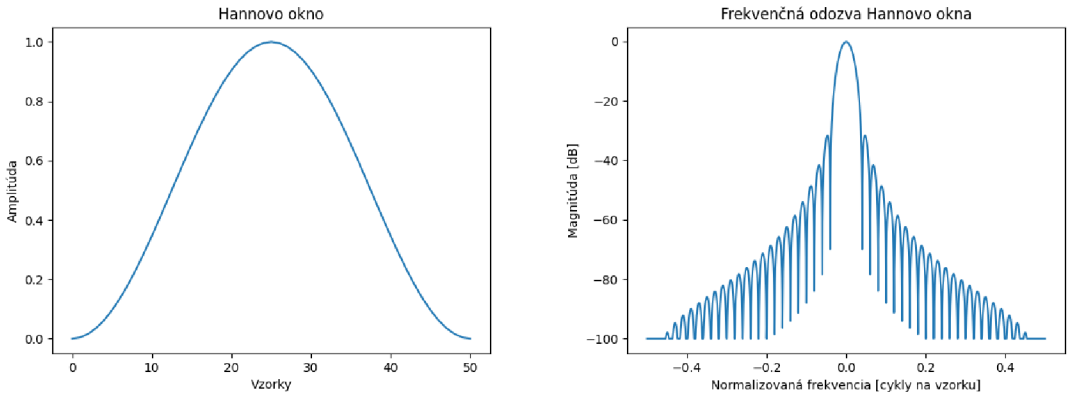
Medzi jednoduchšie okná možme zaradiť

- Dirichletovo okno
- Trojuholníkové okno
- Cosinusové okno
- Hannovo okno
- Hammingovo okno

Hannovo okno je označené po matematikovi Juliusu von Hannovi, je známe aj ako zvýšený kosínus (raised cosine), využíva trigonometrickú funkciu kosínus, ktorá má na krajoch hodnotu blízku k 0. Je jedno z najpoužívanejších časových okien. Predpis funkcie je definovaný nasledovne 5.5:

$$w(n) = \frac{1}{2}(1 - \cos\frac{2\pi n}{N-1}), \qquad 0 \leq n \leq N-1 \tag{5.5}$$

Dĺžka okna je určená vzťahom $L = N + 1$



Obr. 5.2: Hannovo okno a jeho odozva

5.4 Algoritmus získania BPM

Ľudské ucho dokáže určiť hodnotu BPM vnímaním jednotlivých rytmov hudby. Signál zachytený uchom obsahuje určitú energiu, ktorá sa následne premení na elektrický signál, pre mozog ľahko spracovateľný. Je zrejmé, že čím viac energie sa zvukom prenáša, tým hlasnejší bude zvuk. Zvuk však bude počuť ako rytmus iba vtedy, ak jeho súčasna energia je do značnej miery silnejšia než jeho predošlé energetické hodnoty. To znamená, ak mozog zistí zmenu zvukovej energie. Preto, ak ucho zachytí zvuk s veľkými energetickými špičkami, zistí rytmy, ak však budete hrať monotónny hlasný zvuk, nebudete vnímať žiadne rytmy. Základom predpokladom získavania tempa hudby sú **vrcholy zvukovej energie**.

Na začiatku algoritmu je spracovávaný signál rozdelený do **šiestich samostatných signálov**, z ktorých každý pozostáva z frekvenčného obsahu pôvodného signálu z určitého rozsahu zobrazeného v tabuľke 5.2. Rozdelenie sa vykonáva kvôli náchylnosti na chyby v dôsledku konfliktných úderov rôznych nástrojov. Rozdelenie sa uskutoční odobratím Rýchlej Fourierovej transformácie signálu a jeho priradením k ich frekvenčným pásmam. Toto rozdelenie je odvodené od princípov, ktoré vo svojom článku [18] pre detekciu tempa spísal E. Scheirer.

0-200	200-400	400-800	800 - 1600	1600 - 3 200	3 200 - VF

Tabuľka 5.2: Rozdelenie pásiem v Hz, až po hodnotu vzorkovaciu frekvenciu(VF)

- Ďalej je potrebná, aby každá reprezentácia týchto šesť skupín bola vyhladená filtrom. V zásade sa berie každý z našich šiestich frekvenčných pásmových signálov a filtrujeme ich dolnou frekvenciou. Potom je každý zo signálov stočený do pravej polovice Hannova okna s dĺžkou 0,4 sekundy.

- Pre nás je teraz dôležité aby sa zvýrazili jednotlivé zmeny amplitúdy zvuku. Najväčšie zmeny by mali zodpovedať úderom.

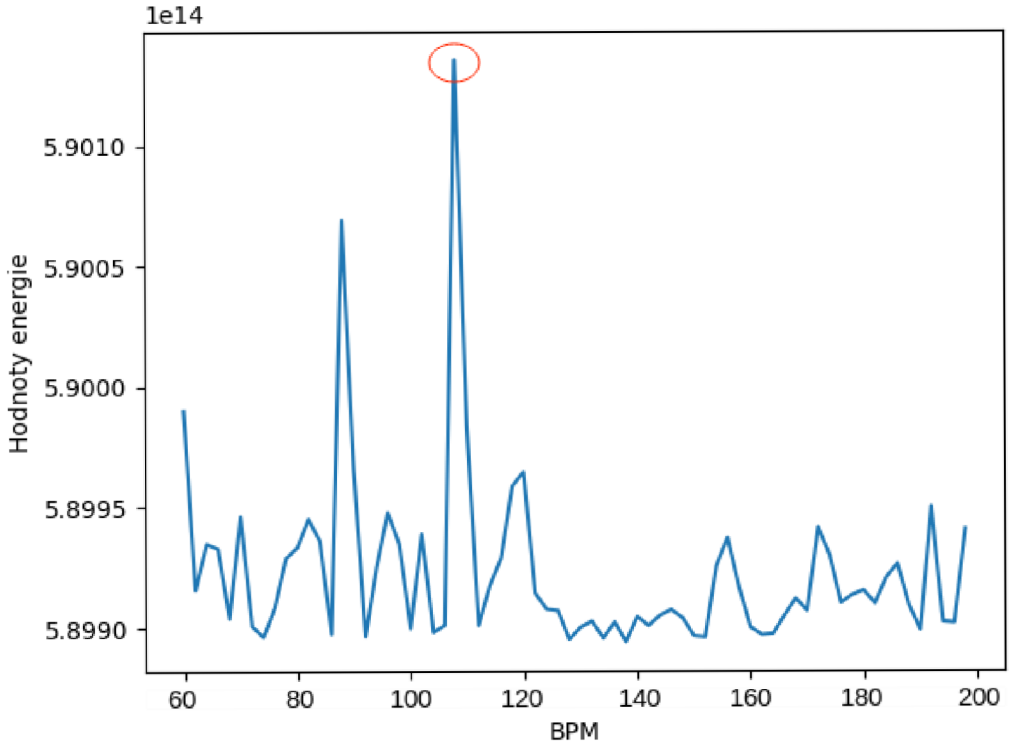
- Nakoniec je po potrebné previesť frekvenčné pásmové signály, aby sme určili, ktor0 poskytuje najvyššiu energiu. Na toto nám slúži **hrebeňový filter**. Hrebeňový filter je v podstate rad impulzov, ktoré sa vyskytujú pravidelne, v tempe, ktoré určíme. Vytvorená sada hrebeňových filtrov sa konvoluje s našim signálom. Rýchla Fourierova

transformácia každého frekvenčne pásmového signálu sa vynásobila rýchlou Fourierovou transformáciou každého hrebeňového filtra a odobrala sa energia každého z nich. Následne sa energie frekvenčných pásiem spočítali. Výsledne vyberieme maximálnu hodnotu týchto energií ako aproximáciu základné tempa našej skladby.

```
for i in range(0, nbands-1):
            val = (abs(dftfil[:,0]*dft[:,i]))
            x = power(val, 2)
            e = e + sum(x)

        if e > maxe:
            sbpm = bpm
            maxe = e
```

Výpis 5.1: Ukážka časti kódu získavania hodnoty BPM pri spočítaní energie v jednotlívých pásmach



Obr. 5.3: Vyhodnocovanie BPM podla amplitúdy energie, pre hodnotu BPM = 108

Kapitola 6

Návrh aplikácie a implementácia

V tejto kapitole chcem objasniť návrh jednotlivých prvkov vytovrenej aplikácie pre klienta MPD. V predošlej kapitole 5 už bola zhrnutá časť implementácie, ktorá sa týka algoritmu pre výpočet BPM.

6.1 Požiadavky na aplikáciu

Cieľom práce bolo vytvoriť aplikáciu pre špeciálne operácie a manipuláciu s metadami. Po preskúmaní existujúceho hudobného prehrávača, som vytvorili funkcionalitu, ktorú by mal tento prehrávač obsahovať. Je zameraná pre časť poslucháčov hudby, ktorí ocenia efektívnejšiu prácu so skladbami.

6.2 Grafické užívateľské rozhranie

Cieľom navrhovania aplikácie prehrávača, je ponúknuť užívateľovi interaktívne prostredie, ktoré umožňuje obohatenú funkcionalitu oproti pôvodnému prehrávaču GMPC, ale zároveň má byť jednoduché a elegantné.

Dôležité je aby užívateľ mal možnosť výberu svojho adresára s obsahom zoznamu svojich skladieb. Ja osobne som sa rozhodol pre písanie aplikácie v príkazovom riadku terminálu, a nepoužívať žiaden z dostupných platených nástrojov pre editovanie grafických prvkov. Grafické užívateľské rozhranie je podobné rozhraniam iným hudobných prehrávačov. Ako väčšina PyQt5 aplikácií, musí byť vytvorený objekt aplikácie **QApplication**

Hlavné okno obsahuje viacero jednoduchých tlačidiel triedy **QPushButton**. Slúžia pre prehrávanie hudby a CRUD operácie nad playlistmi, ako aj operácie nad skladbami.

Pre zobrazenie metadát získaných z lokálnych hudobných súborov slúži **Qlabel**. Ten obsahuje **QHBoxLayout**, rozdelený na dve rovnaké časti tvorené **QVBoxLayout**, prvá zobrazuje prípadne obrázok stiahnutý z lokálneho adresára a druhá časť zobrazuje základne meta údaje.

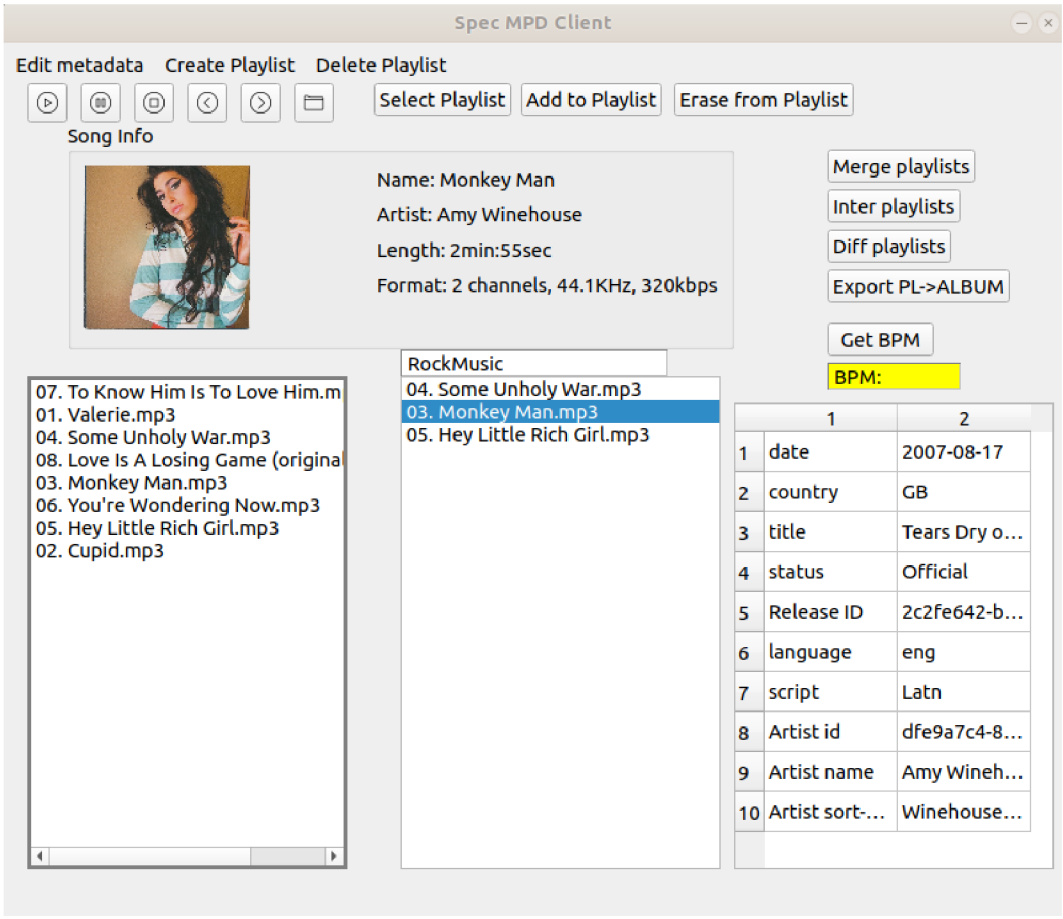
Zobrazenie zoznamu skladieb je tvorené prvkom **QListWidget**. Od tohto zoznamu sa odvíjajú akcie nad ostatnými časťami aplikácie. Pri kliknutí na jednu skladbu z tohoto

zoznamu, užívateľ môže zistiť potrebné metadáta, siahnuť informácie z verejnej databázy, vypočítať BPM skladby, alebo začleniť skladbu do playlistov.

Na zobrazenie zoznamu playlistu vytvoreného užívateľom slúži ďalší QListWidget. Obsah tohto zoznamu je ovládateľný tlačidlami pre playlisty `Select Playlist`, `Add to Playlist`, `Create Playlist`, `Delete from Playlist` v hornej časti programu.

Funkcionalita zabezpečujúca operácie nad playlistami tvoria tlačídlá **QPushButton** `Merge playlists`, `Inter Playlists`, `Diff Playlists`. Tlačidlo `Export PL -> ABUM` vygeneruje playlist ako album.

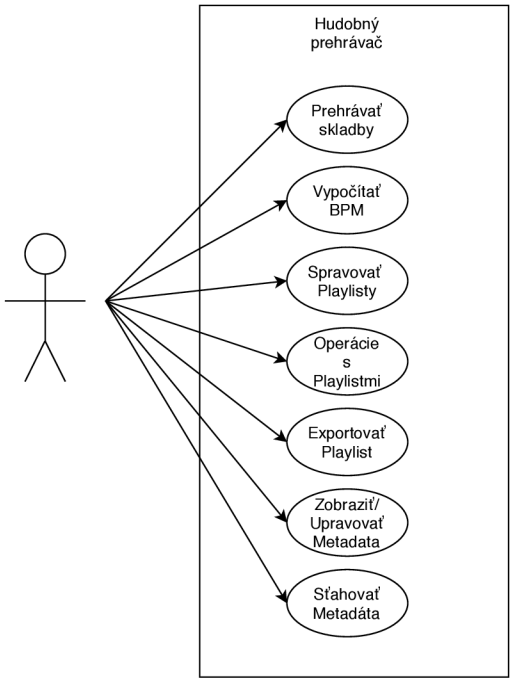
Stiahnuté metadáta z verejnej databáze sa zobrazujú v **QTableWidget** pomocou tabuľky s dvomi stlpčekmi a premenlivým počtom riadkom podľa počtu získaných dát.



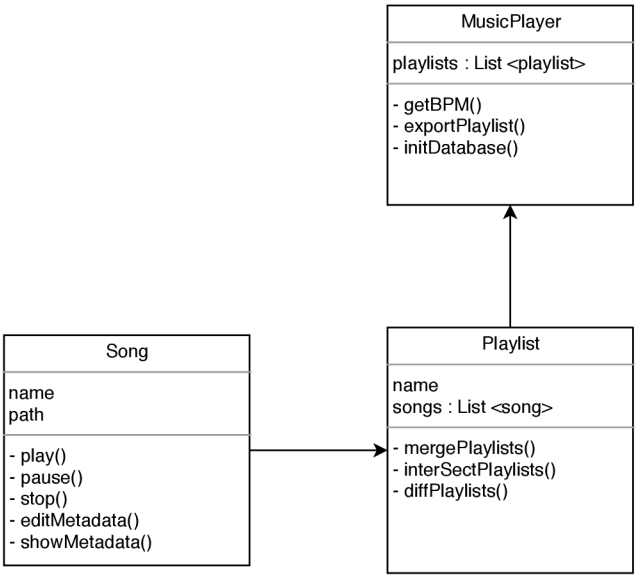
Obr. 6.1: Ukážka rozhrania

6.3 Návrh

Aplikácia bola rozdelená do jednotlivých častí, ako sú napríklad sťahovanie metadát, algoritmus pre výpočet BPM, spravovanie prehrávača a databáza playlistov. V tej sa uchovávajú dynamicky vytvorené dáta, ktoré obsahujú playlisty a skladby. Pre návrh štruktúry aplikácie bol vytvorený diagram tried 6.3 a pre znázornenie funkcií, diagram prípadov použitia 6.2.



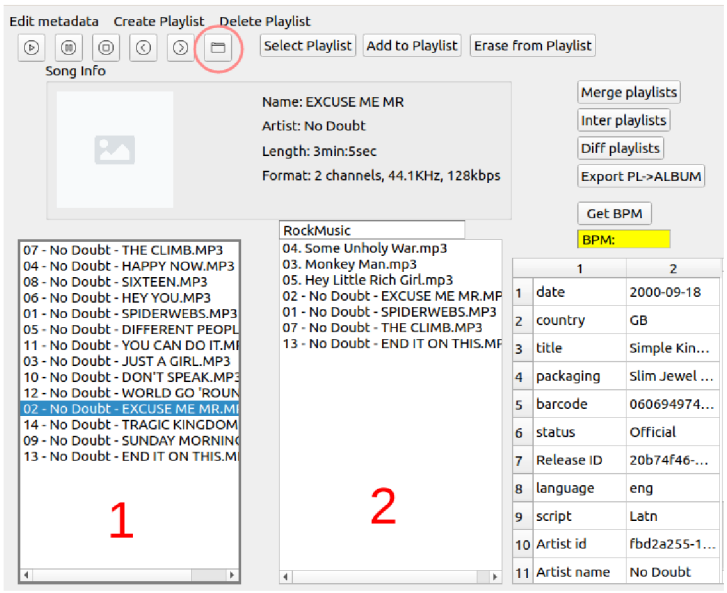
Obr. 6.2: Diagram prípadov použitia pre aplikáciu



Obr. 6.3: Diagram tried pre aplikáciu

Operácie s playlistmi

Viaceré hudobné služby a aplikácie umožňujú používateľom kategorizovať, upravovať a počúvať zoznamy skladieb online. Iné sa zameriavajú na vytváranie zoznamov skladieb pomocou hodnotení a recenzií. Na niektorých weboch používatelia vytvárajú a zdieľajú zoznamy skladieb s poznámkami, čo návštevníkom umožňuje počas čítania načítať kontextové informácie alebo komentáre recenzentov ku každej skladbe. Ja som sa snažil vytvoriť funkcionalitu pre jednoduchú manipuláciu so skladbami v playlistoch. Za týmto účelom bol vytvorený objekt `Playlist`.



Obr. 6.4: Diagram tried pre aplikáciu

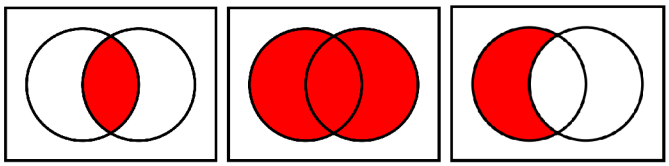
Užívateľ si kliknutím na ikonku zložky vyznačnej na obrázku vyberie zložku s mp3 súbormi. Tá sa prevedie do hlavného okna označeného číslom 1 na obrázku 6.4. Pomocou tlačidla `Create Playlist` si môže vytvoriť svoj playlist. Jeho výber vykoná kliknutím na `Select Playlist`. Keď je zvolený playlist, užívateľ si doňho môže pridávať skladby pomocou tlačidla `Add to Playlist`, alebo skladby odoberať pomocou `Erase from Playlist`. Jeho odstránenie sa dá vykonať pomocou `Delete Playlist`.

Algebraické operácie

Algebraické operácie nad playlistmi sú pre lepšie pochopenie vyobrazené na obrázku 6.5. Vždy sa jedná o dve množiny, resp. v našom prípade playlisty, medzi ktorými môžu prebehnúť operácie prienik, zjednotenie alebo rozdiel. Novo vzniknutý playlist je možné **exportovať** v podobe albumu vo formáte `zip` súboru s editovanými údajmi skadieb pre album.

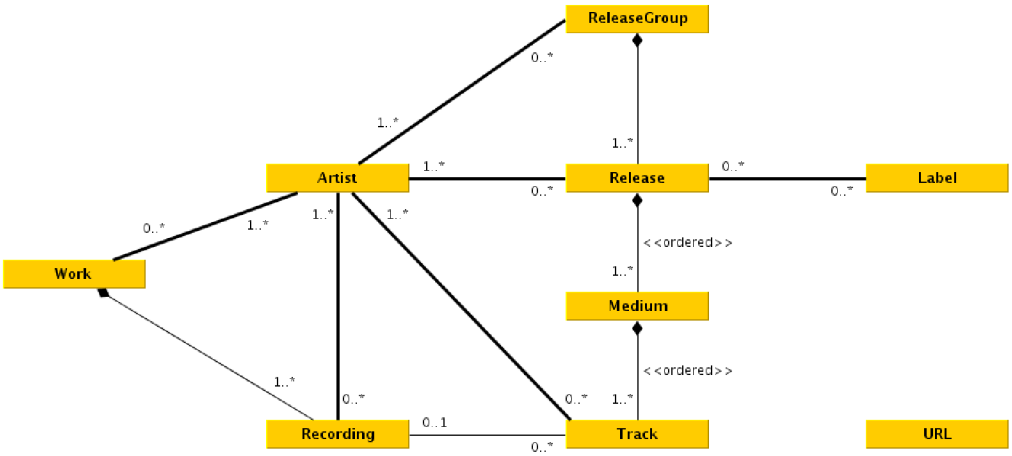
Automatizované sťahovanie metadát

V module `data` môžeme nájsť zdrojový kód pre automatizované sťahovanie metadát. Výsledok z databázy sa uloží do `dictionary` (dict), ktoré obsahuje páry kľúč, hodnota pre stiahnuté metadáta. Vyhľadávanie dát sa vykonáva podľa názvu skladby a mena hudob-



Obr. 6.5: Prienik, zjednotenie, rozdiel

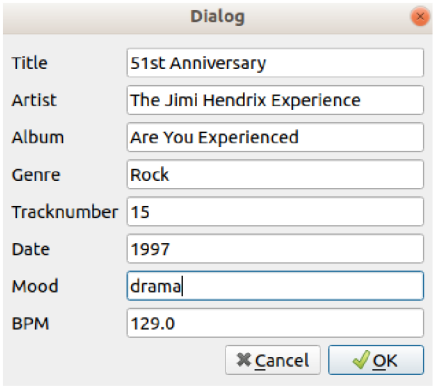
ného interpreta. Subjekty zoznamu majú typ `list`. Dáta sťahované z verejnej databázy **MusicBrainz** sú vo forme JSON. Pre ich stiahnutie nie je potrebná registrácia. API z modulu `musicbrainzngs` ponúka možnosť stiahnuť si údaje o skladbe, albume, umelcovi alebo konkrétnej nahrávke. 6.6. Dáta sú následne zobrazené v tabuľke.



Obr. 6.6: UML model dát v databáze

Prístup k metádatam

Údaje, ktoré sú už v súboroch je možné editovať lokálne pomocou dialóga na obrázku 6.7.



Obr. 6.7: Editovanie metadát

6.4 Použité technológie

Na tvorbu tejto aplikácie bol použitý multi-paradigmový programovací jazyk python, ktorý mi umožnil prácu s mnohými užitočnými knižnicami, prosté vytváranie objektov a garbage collector.

Pre grafické rozhranie bol použitý framework **PyQt5** od Qt Company. Efektívne poslúžil ako set vývojových nástrojov platforiem pre grafické užívateľské prostredie. Qt je všeobecne ale najviac používaný vrámci vývoja aplikácií v C++. Komunita PyQt ponúka rozsiahle množstva príkladov a cenných rád, čo mi dopomohlo k výberu tomuto najpoužívanejšiemu grafickému frameworku vrámci pythonu.

Python-vlc

`Python-vlc` patrí medzi moduly pre prehrávanie hudby. Zahŕňa počítačovú grafiku a zvukové knižnice.

Mutagen

`Mutagen` je modul Pythonu na spracovanie zvukových metadát. Podporuje zvukové formáty ako sú `ASF, FLAC, MP4, MP3, Ogg a AIFF`. Podporované sú všetky verzie ID3v2 a analyzované sú všetky štandardné rámce ID3v2.4. Dokáže prečítať hlavičky aby presne vypočítal dátový tok a dĺžku MP3. Značky ID3 a APEv2 je možné upravovať bez ohľadu na formát zvuku.

Musicbrainzngs

`Musicbrainzngs` implementuje väzby(bindings) pre python z webovej služby MusicBrainz. Pomocou tejto knižnice je možné načítať všetky druhy hudobných metaúdajov z databázy MusicBrainz. Architektúra služby sa riadi zásadami návrhu REST. Interakcia s webovou službou sa vykonáva pomocou protokolu HTTP a všetok obsah sa poskytuje vo formáte JSON. Rovnaká webová služba je k dispozícii aj vo formáte XML.

Numpy

Programovací jazyk Python nebol pôvodne navrhnutý pre numerické výpočty, preto vznikol modul `Numpy`, ktorý je v podstate základná knižnica pre vedecké práce v Pythone. Poskytuje nástroje pre prácu s výkonným multi-dimenzionálným poľom.

Scipy

`SciPy` poskytuje veľké množstvo funkcií, ktoré fungujú na množinových poliach a sú užitočné pre rôzne typy vedeckých a technických aplikácií. V mojej práci bola použitá aj na výpočet rýchlej Fourier Transformácie, či inverznej rýchlej Fourierovej transformácie.

Matplotlib

V mojej práci bol modul `Matplotlib` použitý pre vykresľovanie grafov, ktoré znázorňujú spracovanie zvuku. Vykresľovanie funkcií je porovnateľné s vykresľovaním v `MATLAB`

Json a Objectpath

Tieto moduly slúžia k extrakcii dát z formátu JSON, ktorý je použitý pre sterilizáciu a deserializáciu dát prenášaných cez web.

6.5 Vyhodnotenie výsledkov

Vyhodnotenie výsledkov bolo zamerané na presnosť získaných hodnôt BPM. Za účelom testovania boli použité mp3 súbory s rôznymi žánrami. Spustenie aplikácie a testy boli prevedené na notebooku Lenovo M5400 so 4-jadrovým procesorom i5-4200M CPU 2.50GHz, ktorý beží na operačnom systéme Ubuntu 18.04.3 LTS, 64-bit.Použité hudobné súbory musia byť v čo najlepšej kvalite a vo formáte mp3.

Názov skladby	Žáner	BPM ALG	BPM WEB	Čas výpočtu [s]
Darude - Sandstorm	Elektro	129	136	1.673
The Blaze Territory	Elektro, Dance	126	120	1.664
J. Hendrix - Remeber	Rock	120	100	1.474
A. Winehouse - Cupid	R&B	78	87	1.471
Aerosmith - I dont wanna...	Rock	129	123	1.638

Tabuľka 6.1: Výsledky testovania, BPM ALG znázorňuje výsledky algoritmu z aplikácie, BPM WEB zobrazuje hodnoty z internetovej databázy Spotify

Vyhodnotenie rýchlosti

Pre meranie času pre jazyk Python bol použitý modul `time`. Čas vykonania algoritmu bol vždy uvedený v sekundách

Vyhodnotenie merania

Namerané hodnoty ukázali, že algoritmus pri meraniach niektorých hudobných štýlov, ukazuje odchýlky merania. Ako referenčné hodnoty boli použité hodnoty z webovej databázy pre vyhľadávanie BPM množstva skladieb, ktoré poskytuje služba Spotify. Nepresnosť získaných hodnôt k referenčným mala hodnotu 8%. Hlavnou devízou tohto algoritmu je jeho rýchlosť.

Tempo nie je vždy rovnaké v rámci jednej skladby. V skutočnosti sa môže počas skladby často meniť, aby vyvolalo záujem alebo určité pocity pre poslucháča. Keď sa hudba pohybuje z jedného tempa do druhého, buď spomaľuje alebo zrýchľuje.

Možné rozšírenia

Práca kladie dôraz na vytvorenie desktopovej aplikácie pre operačný systém Linux. Z tohto hľadiska by bolo vhodné doplniť nasledovné rozšírenia:

- Podpora viac audio formátov.
- Funkcia pre generovanie hudby, ktorá by užívateľovi pomohla pri výbere, podľa štatistík prehrávania jednotlivých skladieb a žánrov.
- Vytvorenie jednoduchého ekvalizéra, pre rozdelenie zvuku do rôznych frekvenčných pásiem s prednastavenými profilmi napríklad pre zníženie hlasitosti na niektorých frekvenciách.

Kapitola 7

Záver

Cieľom tejto práce bolo preštudovať problematiku metadát hudobných súborov. Následne analyzovať aktuálne schopnosti hudobného prehrávača GMPC ohľadom manipulácie s playlistmi a s metadátami hudobných súborov.

Získané informácie boli následne použité na vytvorenie doprovodnej aplikácie, ktorá má za úlohu predviesť moje riešenia pre danú problematiku. Jasne a zreteľne sú zdokumentované teoretické a praktické časti tejto problematiky. Prvým krokom bolo štúdium metadát. Nasledoval spôsob ich ukladania a z čoho sú zložené, pre lepšie pochopenie v čom spočíva moja úloha. Ďalej nasledovala kategorizácia hudobných súborov, ktorú je potrebné pochopiť, pretože v mojej aplikácii pracujem s niektorými z uvedených formátov a dôvody prečo pri algoritme so spracovaním zvuku narábam s komprimovaným formátom. V nasledovnej kapitola pre zhrnutie funkcií prehrávača bolí schopnosti prehrávača.

S porozumením týchto informácií bola vytvorená a implementovaná aplikácia pre klient MPD, ktorý pracuje s metadatami z lokálnych súborov, automatizuje sťahovanie dát z verejných databáz. Táto aplikácia umožňuje užívateľovi manipulovať s playlistmi. Ak by som mal navrhnúť ďalšie funkcie prehrávača, pracoval by som už s navrhnutým klonom a na už implementovaných základoch pridával možné rozšírenia. V aplikácii je implementovaný algoritmus pre výpočet BPM, ktorý spracováva nahrávky hudobných súborov MP3.

Program bol implementovaný v interpretovanom jazyku Python v spojení s framewrokom PyQt5, ktorý umožňoval tvorbu grafického užívateľského rozhrania. Súčasťou implementácie bolo použitie viacerých modulov ako Mutagen a iných. Algoritmus pre získanie BPM vykazoval odchýlky pri rôznych žánroch hudby, ale záviselo to čiastočne aj na kvalite danej skladby.

Literatúra

[1] *Music Player Daemon* [https://www.musicpd.org/]. Online; 15 Januar 2020.

[2] *Digitization - Digital Sound & Music* [online]. May 2014 [cit. 2020-05-20]. Dostupné z: http://digitalsoundandmusic.com/5-1-2-digitization.

[3] *Development / XML Web Service / Version 2 - MusicBrainz*. May 2020. [Online; accessed 18. May 2020]. Dostupné z: https://musicbrainz.org/doc/Development/XML_Web_Service/Version_2.

[4] *MPD Wiki ubuntuusers.de* [online]. May 2020 [cit. 2020-05-20]. Dostupné z: https://wiki.ubuntuusers.de/MPD.

[5] *User's Manual Music Player Daemon 0.22 documentation* [online]. Apr 2020 [cit. 2020-05-20]. Dostupné z: https://www.musicpd.org/doc/html/user.html.

[6] Brandenburg, K. MP3 and AAC Explained. In: *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Sep 1999. Dostupné z: http://www.aes.org/e-lib/browse.cfm?elib=8079.

[7] Bray, T., Yergeau, F., Paoli, J., Maler, E. a Sperberg McQueen, M. *Extensible Markup Language (XML) 1.0 (Third Edition)*. W3C Recommendation. W3C, február 2004. Dostupné z: http://www.w3.org/TR/2004/REC-xml-20040204/.

[8] Coalson, J. *FLAC - Free Lossless Audio Codec* [online]. May 2020 [cit. 2020-05-20]. Dostupné z: https://xiph.org/flac.

[9] Crockford, D. *The application/json Media Type for JavaScript Object Notation (JSON)* [Internet Requests for Comments]. RFC 4627. July 2006. Dostupné z: http://www.rfc-editor.org/rfc/rfc4627.txt.

[10] Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L. et al. *Hypertext Transfer Protocol – HTTP/1.1* [Internet Requests for Comments]. RFC 2616. RFC Editor, June 1999. Dostupné z: http://www.rfc-editor.org/rfc/rfc2616.txt.

[11] Kemp, E. *ID3v1* [online]. 1997 [cit. 2020-05-18]. Dostupné z: https://id3.org/ID3v1.

[12] Lacinak, C. *Embedded Metadata in WAVE Files* [online]. [cit. 2010-05-18]. Dostupné z: https://www.weareavp.com/wp-content/uploads/2017/07/EmbeddedMetadata.pdf.

[13] Nilsson, M. *ID3 tag version 2.4.0* [online]. 2000 [cit. 2020-05-18]. Dostupné z: https://id3.org/id3v2.4.0-structure.

[14] Omar Adil Mahdi, A. J. M. *Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique* [online]. [cit. 2020-05-18]. Dostupné z: http://ijcsi.org/papers/IJCSI-9-6-3-53-59.pdf.

[15] Qball. *Gnome Music Player Client* [online]. [cit. 2020-5-25]. Dostupné z: https://gmpclient.org/.

[16] Robert Macrae. *Linking Music Metadata*. 2002. Dostupné z: www.eecs.qmul.ac.uk/~simond/phd/RobertMacrae-PhD-Thesis.pdf.

[17] Russ, M. *Sound Synthesis and Sampling*. CRC Press, 2012 [cit. 2020-05-20]. ISBN 9781136122149.

[18] Scheirer, E. D. *Tempo and beat analysis of acoustic musical signals* [online]. 1998 [cit. 2020-05-20]. Dostupné z: http://www-labs.iro.umontreal.ca/~pift6080/H09/documents/papers/scheirer_jasa.pdf.

[19] Steinebach, M., Lang, A. a Dittmann, J. StirMark Benchmark: audio watermarking attacks based on lossy compression. In: III, E. J. D. a Wong, P. W., ed. *Security and Watermarking of Multimedia Contents IV*. SPIE, 2002, s. 79 – 90. Dostupné z: https://doi.org/10.1117/12.465333.

Príloha A

Manuál

Pokyny pre inštaláciu pre Linux:

1. Rozbaľte súbor `Package_xadame41.zip`
2. V tomto súbore spustite inštaláciu v termináli:
   `python3 setup.py install --user`
   `pip3 install -r requirements.txt`
3. V zložke so súborom `main.py` spustite aplikáciu:
   `python3 main.py`