FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Webová aplikace pro vyhledávání MIDI souborů na základě podobnosti melodií
Student:	Bc. Tadeáš Pála
Vedoucí:	Ing. Jiří Novák, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem práce je návrh, implementace a otestování webové aplikace sloužící pro vyhledávání skladeb v podobě MIDI souborů na základě podobnosti s uživatelem zadanými melodiemi. Aplikace bude obsahovat následující funkce: nahrání vyhledávané melodie pomocí MIDI nástroje nebo počítačové klávesnice, zadání vyhledávané melodie pomocí myši, zobrazení skladeb nejpodobnějších uživatelem zadané melodii, optimalizované zobrazení pro mobilní zařízení. Součástí práce je implementace backend služby poskytující data webové aplikaci pomocí REST API.

Úkoly:

1. Nastudujte a analyzujte problematiku podobnostního vyhledávání MIDI souborů.
2. Navrhněte způsob předzpracování dat pro rychlejší vyhledávání skladeb.
3. Implementujte několik algoritmů pro porovnání skladeb.
4. Algoritmy porovnejte z hlediska přesnosti a rychlosti vyhledávání.
5. Navrhněte rozhraní pro komunikaci mezi webovou aplikací a backendem.
6. Navrhněte a implementujte webovou aplikaci.
7. Aplikaci otestujte.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 24. srpna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Webová aplikace pro vyhledávání MIDI souborů na základě podobnosti melodií

Bc. Tadeáš Pála

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Novák, Ph.D.

4. května 2023

Poděkování

Chtěl bych poděkovat především svému vedoucímu Ing. Jiřímu Novákovi, Ph.D. za veškeré rady, bez kterých by tato práce nemohla vzniknout. Dále bych chtěl poděkovat celé své rodině a všem přátelům za jejich pomoc a podporu.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. května 2023 . . . . . .

Odkaz na tuto práci

Pála, Tadeáš. *Webová aplikace pro vyhledávání MIDI souborů na základě podobnosti melodií*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Tato diplomová práce se zabývá vyhledáváním v hudebních MIDI databázích na základě podobnosti melodií. V práci jsou prezentovány algoritmy pro předzpracování a porovnání skladeb a struktura vyhledávače. Dále je prezentován paralelizační model umožňující zrychlení vyhledávání. U jednotlivých přístupů je porovnána jejich robustnost a efektivita a na základě této analýzy jsou zvoleny ty nejvhodnější. Z vybraných přístupů je vytvořena výsledná aplikace umožňující uživateli vyhledávat skladby pomocí grafického rozhraní.

Klíčová slova hudební databáze, melodická podobnost, hudební vyhledávač, MIDI, vyhledávání melodie

Abstract

This thesis focuses on searching songs in MIDI databases using melodic similarity. Presented are methods used for preprocessing, comparison of musical pieces and an architecture of a search engine leveraging them. Furthermore, a parallelization model allowing search speed up is introduced. The different approaches are compared and based on the results, the most suitable ones are selected for the final application. Said application allows users to search songs based on melodic similarity using graphical interface.

Keywords music database, melodic similarity, music search engine, MIDI, melody search

Obsah

Seznam obrázků

Úvod

Hudba je pro většinu lidí každodenní součástí života. Nemusí vždy hrát hlavní roli. Může sloužit k podkreslení nálady v obchodních centrech, restauracích, čajovnách nebo jenom k vyplnění ticha, když na chvíli ustane konverzace. Slouží jako nástroj k navození atmosféry ve filmech a seriálech. Při řízení pomáhá rádio udržet řidiče vzhůru a v nočních klubech DJ rozpohybovat taneční parket.

Ať už se jedná o skladby z rádia nebo o playlist vytvořený personálem restaurace, všechny tyto situace mají jedno společné. Hudbu si nevybírá posluchač. Zákonitě tedy musí narazit na písničky, jejichž jméno ani autora nezná. V takovém případě může vytáhnout telefon, nahrát krátkou část skladby a použít jednu z mnoha aplikací k jejímu nalezení. Problém však nastává v případě, kdy si člověk vzpomene na skladbu, kterou slyšel minulý týden v rádiu a zapomněl si ji nahrát. Pamatuje si jenom útržek melodie refrénu nebo třeba kytarový riff, který se opakuje ve slokách.

Koncept řešení takového problému by měla nastínit aplikace, která je předmětem této práce. Měla by sloužit lidem, kteří mají „hudebního brouka“ v hlavě. Umožnit jim vyhledat si skladbu, kterou slyšeli minulé úterý cestou do práce v rádiu a ze které si pamatují jeden desetivteřinový kytarový part.

Kapitola 1

Hudební teorie

Tato část je věnována vymezení pojmů z oblasti hudební teorie používaných v této práci. Základním stavebním kamenem tohoto oboru a nejmenší jednotkou je nota. Nota má dva atributy hodnotu a výšku. Hodnota noty určuje její délku, tedy jak dlouho zvuk zní. Délka trvání noty je udávána v dobách. Doby jsou základní jednotkou rytmu hudby. Představují pevně daný časový interval. Například jedno klepnutí metronomu může představovat jednu dobu. Základní hodnoty not jsou celá — dlouhá čtyři doby, půlová — dlouhá dvě doby, čtvrťová — dlouhá jednu dobu, osminová — dlouhá půl doby atd. Kratší noty vznikají zvětšováním exponentu jmenovatele pomyslného zlomku (čtvrtina, osmina, šestnáctina...) a půlením počtu dob trvání noty. I když teoreticky není délka noty zdola nijak omezena, v běžné hudební notaci se vzácně vyskytují noty kratší než šedesátičtvrtinové, které trvají jednu šestnáctinu doby.

Takt je označení části skladby určité délky. V hudební notaci většinou bývá na začátku skladby uvedeno taktové předznamenání v podobě zlomku. Jmenovatel udává, jaká hodnota noty odpovídá jedné době a čitatel udává počet dob v jednom taktu. Například taktové předznamenání $\frac{3}{4}$ znamená, že takty skladby mají délku tří čtvrťových not. Takový takt se nazývá tříčtvrťový a spolu s taktem čtyřčtvrťovým tvoří většinu populární hudby.

Kromě rytmické složky hudby je dále nutné zmínit i tu melodickou. Rovnoměrně temperované ladění je v současnosti jedním z nejvíce používaných ladění. Vzniká rozdělením oktávy, což je interval mezi dvěma tóny s poměrem frekvencí 2:1, na dvanáct stejně velkých intervalů. Krok mezi dvěma sousedními intervaly je v oboru hudební teorie nazýván půltón a dva půltóny tvoří celý tón. Pro přesné ukotvení akustické polohy rovnoměrně temperovaného ladění bývá používáno a1 (komorní a), jemuž je standardně přiřazována frekvence 440 Hz. Od něj jsou dále odvozeny všechny ostatní tóny.

Souzvuk tří a více tónů se označuje jako akord. Příkladem může být stisk tří kláves na klavíru nebo více strun na kytaře. Někdy bývají za akordy označovány i případy, kdy jsou jeho noty zahrané rozloženě jako tzv. arpe-

ggio, ale pro potřeby této práce jsou jako akordy označovány pouze případy, kdy jsou noty zahrány současně. Protějškem vertikálního skládání not v akordech je horizontálně orientované uspořádání nazývané melodie. To je rytmicky organizovaná sekvence na sebe navazujících tónů.

Významným pojmem v kontextu harmonie je tonalita. Ta označuje vztahový řád a uspořádání jednotlivých tónů a akordů skladby. Jejím základem je tónika. Ta je prvním tónem diatonické stupnice, určuje tonální centrum skladby a slouží jako referenční bod pro zbytek not a akordů skladby. Posledním důležitým pojmem, který souvisí s tonalitou a volně navazuje na koncept tóniky je transpozice. Transpozice značí převedení skladby do jiné tóniny, tedy změnu jejího tonálního centra při zachování intervalů mezi jednotlivými tóny beze změny atributů týkajících se tempa, rytmu a přednesu. Zjednodušeně se dá transpozice popsat jako posun každého tónu skladby o určitý počet půltónů nahoru nebo dolů.

1.1 Absolutní a relativní sluch

Protože se tato práce zabývá podobností melodií, je třeba definovat i určité pojmy týkající se vnímání hudby. Následující část krátce popisuje dva základní druhy hudebního sluchu a jejich vliv na schopnost zapamatování a reprodukce melodie.

Absolutní sluch je schopnost identifikovat nebo reprodukovat určitou hudební notu bez potřeby referenčního tónu [1]. Osoba s absolutním sluchem může například uslyšet tón a ihned určit, že se jedná o C nebo F#. Touto schopností disponuje jen velmi malá část populace. Běžně se uvádí, že se vyskytuje pouze u 1 z 10000 lidí, přesné číslo však není známo [2].

Relativní sluch je schopnost identifikovat nebo reprodukovat hudební notu za pomoci referenčního tónu. Jedná se o schopnost rozpoznat interval mezi dvěma tóny. Například když osoba s relativním sluchem uslyší dvě po sobě zahrané noty, dokáže identifikovat druhou notu, pokud zná výšku té první. Lidé s rozvinutým relativním sluchem si dokáží zapamatovat i dlouhé melodické pasáže, nicméně vždy si pamatují pouze relativní vztahy mezi notami a ne jejich absolutní výšky. Na rozdíl od absolutního sluchu, který bývá vrozený, lze schopnost relativního sluchu během života rozvíjet.

Kapitola 2

Analýza

2.1 MIDI

MIDI je technologický standard [3] popisující komunikační protokol, digitální rozhraní a elektronické konektory pro propojení a komunikaci mezi elektronickými hudebními nástroji a dalšími zařízeními jako jsou sekvencery, počítače nebo mixery. MIDI nepřenáší nahrané zvuky. Přenáší pouze informace o notách, rytmu, dynamice, polohách pedálů a dalších hudebních parametrech. MIDI data tedy popisují, jaké tóny se mají hrát, jak dlouho a jak silně mají být hrány, ale nezahrnují žádné zvuky samy o sobě.

Data jsou přenášena ve formě zpráv obsahujících informace o hudebních událostech. Zpráva se skládá z jednoho osmibitového status bytu většinou následovaného jedním nebo dvěma datovými byty. Status byte určuje typ zprávy. Ne všechny hodnoty status bytu jsou specifikací definovány. Některé jsou ponechány jako nedefinované/rezervované pro další rozšíření. Typy zpráv lze rozdělit do dvou kategorií: zprávy kanálu a zprávy systému.

Před bližším popisem kategorií zpráv je nejprve potřeba vysvětlit koncept kanálů. MIDI specifikace definuje 16 kanálů. Kanály jsou číslované virtuální cesty, po kterých se zprávy posílají mezi zařízeními. Každému hudebnímu nástroji nebo zvukovému modulu může být přiřazen kanál, pomocí něhož lze rozlišit, jaké zprávy bude přijímat.

Zprávy kanálu ve status bytu kromě informace o svém typu obsahují ještě čtyři bity, které definují kanál, pro který je zpráva určena. Tato kategorie zpráv tvoří většinu provozu v běžném datovém toku MIDI dat. Nejčastějšími typy zpráv této kategorie jsou Note on a Note off, které značí začátek a konec noty. Dále se vyskytují zprávy typu Program change, které značí, že by měl být změněn nástroj a Control change, které symbolizují změnu nějakého parametru nástroje. Zpráv kanálů existuje více, tyto popsané se však vyskytují nejčastěji.

Zprávy systému neobsahují informace o cílovém kanálu a přijímají je všechna zařízení. Specifikace definuje tři základní druhy systémových zpráv: System

common messages, System real-time messages a System exclusive messages. Do System common messages spadají zprávy jako je výběr písně nebo posun na jinou část písně. System real-time messages jsou zprávy sloužící pro synchronizaci zařízení. Spadá sem spuštění přehrávání, zastavení přehrávání nebo pravidelný synchronizační signál (MIDI clock). System exclusive messages je kategorie zpráv, které umožňují výrobcům definovat vlastní typ zprávy. Běžně se používají k přenosu dat pro aktualizaci firmwaru zařízení nebo nahrání nového zvuku do nástroje.

MIDI specifikace kromě komunikačního protokolu také definuje standard MIDI file (SMF) [4]. To je souborový formát umožňující standardizované kódování a ukládání hudebních dat v binární podobě. Typickou příponou SMF je `.mid` nebo `.midi`. Soubor se skládá z hlavičky a datových bloků. Hlavička obsahuje informace o rozlišení časování, formátu souboru, počtu stop a další metadata. Datové bloky obsahují samotné MIDI zprávy pro každou stopu uspořádány do jedné nebo více stop. Každá stopa může obsahovat zprávy pro jeden nebo více MIDI kanálů, což umožňuje ukládat vícevrstvé aranžmá.

Existují tři formáty SMF souborů. Formát 0 má pouze jednu stopu, která obsahuje všechny MIDI zprávy. Formát 1 ukládá jednu nebo více současně hrajících stop. Formát 2 umožňuje ukládání více stop, které nehrají současně. Nejběžnějšími formáty pro přenos hudebních záznamů jsou 0 a 1.

MIDI zprávy jsou v jedné stopě souboru ukládány v pořadí tak, jak by měly nastat po sobě. Před každou zprávou je uveden časový údaj, který udává dobu která uplynula od předchozí zprávy. MIDI nevyužívá pro měření času běžné jednotky jako jsou vteřiny, ale tzv. ticks. Délka jednoho ticku je definována rozlišením časování v hlavičce souboru. Rozlišení je udáváno v ticks per quarter note (TPQN), to je číslo udávající délku čtvrťové noty v počtu ticků. Pro přesné analýzy MIDI souborů je třeba hodnoty TPQN srovnat a přeškálovat časování zpráv na jednu hodnotu.

2.2 Hudební vyhledávače

Hudební vyhledávače jsou nástroje, které umožňují uživatelům najít hudbu na základě svých preferencí. Lze je rozdělit do dvou hlavních kategorií. Ta první pracuje na základě algoritmů, které umožňují vyhledávat hudbu podle kritérií jako jsou jméno interpreta, název skladby, žánr nebo výsledky podobného hudebního stylu. Často také využívají data o preferencích uživatelů, aby mohly nabízet doporučení a personalizované playlisty. Tyto vyhledávače se dnes vyskytují v různých podobách jako jsou samostatné aplikace, webové stránky, nebo součásti větších digitálních platforem. Mezi nejpopulárnější hudební vyhledávače tohoto typu patří Spotify, Apple Music, Deezer, YouTube Music a Tidal.

Další skupinou jsou vyhledávače umožňující vyhledávání na základě obsahu. Dotazy pro tuto kategorii vyhledávačů mohou mít například podobu

zabroukané melodie, nahrané části písničky nebo ručně zadané série not. Do této kategorie je možné zařadit například Shazam, Hymnary nebo Midomi. Tento typ vyhledávače na rozdíl od předchozí skupiny většinou neslouží k objevování nové hudby nebo k rychlému výběru uživateli již známé skladby. I když k tomu lze některé zmíněné vyhledávače použít, většinou to bývá prezentováno jako jeden ze sekundárních případů užití. Tato kategorie vyhledávačů existuje s primárním cílem nalezení konkrétní skladby, kterou uživatel slyšel například v rádiu nebo v televizi a nezná její název.

2.3 Vyhledávání v hudebních databázích na základě obsahu

Primární předpoklad vyhledávání na základě obsahu je, že metadata jsou pro kýžený typ dotazu buď nevhodná, nedostatečná nebo neexistují. V případě hudebních IR (Information Retrieval) systémů mohou platit všechny tři předpoklady. Metadata mohou být nevhodná, protože jsou buď příliš obecná a nedokážou rozlišit různá hudební díla — existují stovky pomalých balad s anglickým textem zpívaným mužským hlasem — nebo jsou příliš specifická a vyžadují přesné definování informací — jméno zpěváka, jméno alba a datum vydání — nebo vyžadují určitou úroveň hudební znalosti — píseň je ve 3/4 taktu, využívá frygický modus stupnice C dur s kytarou hrající převážně první inverze powerchordů a basou alternující mezi základními tóny akordů a jejich terciemi.

MIR (Music Information Retrieval) systémy pro vyhledávání na základě obsahu lze rozdělit na dvě základní skupiny — systémy pro vyhledávání zvukových dat a systémy pro vyhledávání symbolicky zapsané hudby (např. notový zápis nebo MIDI soubory). Systémy pro vyhledávání zvukových dat mohou být použity například k identifikaci skladby právě hrající v rádiu. Pořízením krátké nahrávky na mobilní telefon a využitím existujícího systému, jako je Shazam, může uživatel nalézt skladbu, aniž by znal její název. Pro tento typ vyhledávání bývá často využíváno techniky zvané audio fingerprinting [5, 6]. Princip audio fingerprintingu spočívá ve vygenerování jednoznačného digitálního otisku zvukové stopy a jeho porovnání s dalšími otisky v databázi. Existují i další techniky porovnávání zvukových stop, nicméně nejsou hlavním předmětem této práce.

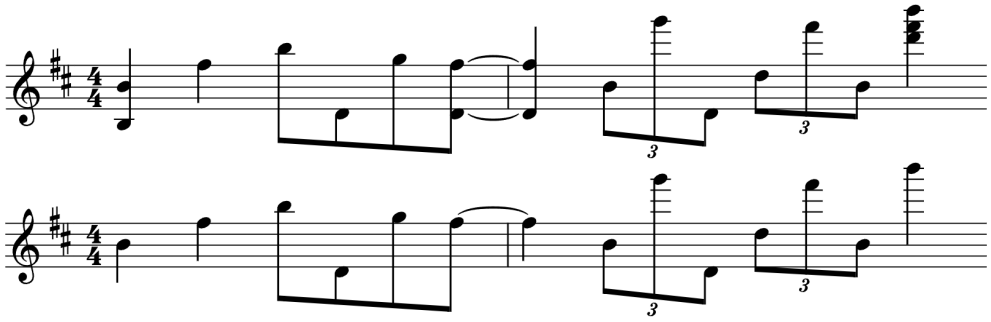
Tato práce se primárně zabývá druhou kategorií — vyhledávači pracujícími nad databázemi skladeb zapsaných symbolickou notací. Vyhledávače spadající do této skupiny se typicky zaměřují na jeden ze dvou přístupů. Buď porovnávají polyfonické reprezentace skladeb nebo porovnávají melodie, tedy monofonické reprezentace. Pro porovnávání polyfonických reprezentací skladeb lze využít mnoho způsobů. Clausen et al. [7] prezentují metodu, která nahlíží na skladby jako množiny not definované počátkem, délkou a výškou. Porovnávání skladeb s dotazem probíhá na základě hledání nadmnožin do-

tazu. Typke et al. [8] také pohlíží na skladby jako na množiny not, ale místo hledání nadmnožin používají k porovnávání transportační vzdálenosti jako Earth mover's distance.

Pro porovnávání monofonických reprezentací skladeb bývají často využívány techniky běžně se vyskytující v oblasti porovnávání textových řetězců [9, 10]. Využívány bývají algoritmy jako je lokální sekvenční alignment nebo nejdelší společná podposloupnost, které umožňují nalezení přibližné shody. Použití transportačních metrik jako Earth mover's distance a podstatu segmentace prezentují Typke et al. [11]. Suyoto a Uitdenbogerd [12] ukazují, že jako efektivní způsob vyhledávání lze použít také porovnávání n-gramů. Tato práce se zabývá tvorbou a porovnáváním monofonických reprezentací skladeb pomocí algoritmů pro porovnání textových řetězců. Architektura vyhledávače a jeho části popsané níže vychází ze struktury popsané Uitdenbogerd a Justin [9].

2.4 Kroky vyhledávání

Tato sekce je věnována podrobnější analýze jednotlivých fází vyhledávání. Blíže se zaměřuje a vysvětluje podstatu jednotlivých kroků a popisuje relevantní algoritmy.



Obrázek 2.1: Extrakce melodie

2.4.1 Extrakce melodie

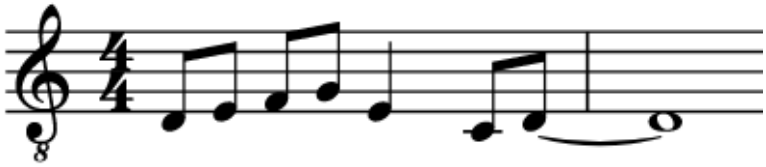
Prvním krokem vyhledávání je extrakce melodie. Ta se vyskytuje z důvodu, že vyhledávač se zaměřuje na monofonické reprezentace skladeb. Je tedy před dalším zpracovaním nutné převést polyfonické stopy (v jeden okamžik mohou obsahovat více jak jeden tón zahraný současně) na monofonické. Polyfonické záznamy se objevují, když jsou zaznamenány nějaké vícehlasé nástroje, jako je například kytara nebo klavír.

Za melodii je vždy považována nejvyšší nota. Ostatní noty nejsou pro potřeby porovnávání brány v potaz. Příklad extrakce melodie je vidět na

obrázku 2.1. Horní notový zápis ukazuje zjednodušeně zaznamenanou polyfonickou část skladby G.O.A.T. od skupiny Polyphia. Spodní zápis znázorňuje extrahovanou melodii. Uitdenbogerd a Zobel [9] ukazují, že tento způsob extrakce je v porovnání s jinými komplexnějšími způsoby sice jednoduchý, ale velmi efektivní.

2.4.2 Standardizace melodie

Standardizací melodie je v kontextu této práce nazýván proces převedení melodií na jednotnou zjednodušenou reprezentaci s cílem vyzdvihnout některé žádoucí vlastnosti, odstranit ty nežádoucí a převést některé z nich na obecnější úroveň. Následující sekce se zabývají jednotlivými metodami standardizace melodie.



Obrázek 2.2: Příklad standardizace

2.4.2.1 Absolutní výška

Tato metoda je ze všech implementovaných nejjednodušší. Melodie je převedena na sérii čísel reprezentujících výšky not dle specifikace MIDI. Například melodie na obrázku 2.2 by byla reprezentována jako série čísel 50, 52, 53, 55, 52, 48, 50.

Pokud je melodie standardizována tímto způsobem, ztrácí se většina informace o rytmu. Nevýhodou tohoto přístupu je potřeba přesného určení tonálního centra melodie, což v některých případech bez absolutního sluchu, kterým disponuje velmi malá část lidstva, může být velmi obtížné. V mnoha případech se může stát, že jsou melodie totožné, pouze je jedna z nich transponována do jiné tóniny. V takovém případě budou řetězce absolutních výšek tónů naprosto odlišné, i když jsou melodie kromě změny tóniny stejné. Tuto nevýhodu dokáží eliminovat dále popsané metody. Standardizace pomocí absolutních výšek not slouží převážně pro získání referenčních výsledků, se kterými mohou být porovnávány výsledky dalších standardizačních metod.

2.4.2.2 Relativní intervaly

Dalším ze způsobů standardizace melodie je převedení na sérii relativních intervalů [9]. Tato metoda přikládá vysokou váhu vztahům mezi výškami jednotlivých not vůči notám ostatním. Relativní vzdálenost výšek not je podle

Deutsch [13] jedním z nejvýznamnějších faktorů pro identifikaci melodie posluchačem.

První notě je přidělena hodnota 0. Každé následující notě je přidělena hodnota rovna počtu půltónů, o který klesla nebo vzrostla oproti notě předchozí. Například notový zápis na obrázku 2.2 by byl převeden na sérii relativních intervalů $0, 2, 1, 2, -3, -4, 2$ (nota D je od noty E vzdálena 2 půltóny, F je od E 1 půltón, G je od F 1 půltón atd.).

Tento zápis umožňuje porovnávat různé části bez ohledu na jejich tonální centrum, neboť skladba transponovaná do různých tónin bude vždy reprezentována stejnou sérií relativních intervalů. Nevýhodou této reprezentace je, že se ztratí část informace o rytmu, neboť jsou všechny noty převedeny na stejnou délku. Tento fakt však může při vyhledávání i pomoci, neboť díky němu může uživatel zadat dotaz, který není úplně rytmicky totožný s originálem (např. jedna nota je o půl doby delší, než by měla být) a vyhledávač by ho stále měl být schopen rozpoznat.

2.4.2.3 Parsonsův kód

Parsonsův kód byl poprvé představen Denysem Parsonsem v *The directory of tunes and musical themes* [14]. Tento způsob standardizace umožňuje převést melodii na její melodickou konturu bez informace o konkrétních intervalech. Melodie je zapsána jako série znaků popisujících, zda je nota oproti předchozí nižší nebo vyšší. Denys Parsons pro tento zápis používá znaky:

***** začátek melodie

D nota je nižší

R nota je stejné výšky

U nota je vyšší

Notový zápis na obrázku 2.2 by byl Parsonsovým kódem zapsán následovně: * U U U D D U.

Tento zápis je možné považovat za zobecnění výše zmíněných relativních intervalů. Z toho vyplývají i podobné výhody a nevýhody. Jednou z jeho výhod pro použití při porovnávání skladeb na základě melodie je stejně jako u relativních intervalů eliminace potřeby shodného tonálního centra (melodie bude reprezentována stejně v jakékoli transpozici). Nevýhodou je, že se opět ztrácí informace o rytmu melodie. Větší obecnost zápisu oproti relativním intervalům může mít za následek, že větší množství melodií bude zapsáno stejným způsobem. Toto zobecnění však nemusí mít vždy pouze negativní následky. Umožňuje například větší toleranci chyb při zadávání dotazů.

2.4.2.4 Baseline intervaly

Metoda standardizace melodie pomocí baseline intervalů využívá podobných principů, jako metoda relativních intervalů, snaží se však výsledku dosáhnout jiným způsobem. Oproti metodě relativních intervalů nejsou výšky not porovnávány s předchozí notou, ale s první notou melodie. První notě je přiřazena hodnota 0. Každé další notě je přiřazena hodnota odpovídající počtu půltónů, o kolik se liší od noty první. Například notový zápis na obrázku 2.2 by byl převeden na následující sérii čísel: $0, 2, 3, 5, 2, -2, 0$. Série baseline intervalů je vlastně série prefixových součtů relativních intervalů.

Takto standardizovaná melodie, podobně jako ostatní výše popsané metody standardizace, udržuje pouze malou část informace o rytmu. Výhodou je, že při metodě baseline intervalů — stejně jako při metodě relativních intervalů — jsou melodie lišící se pouze transpozicí reprezentovány stejnou sérií číslic.

2.4.3 Podobnostní metriky

Po předzpracování melodie výše zmíněnými metodami nastává krok jejich vzájemného porovnání. Podobnost předzpracované skladby a uživatelského dotazu je určována na základě metrik popsaných v této části.

2.4.3.1 Nejdelší společná podposloupnost

Nejdelší společná podposloupnost je posloupnost prvků, která se vyskytuje ve všech vstupních posloupnostech (prvky podposloupnosti nemusí ve vstupních posloupnostech následovat přímo za sebou).

Pro řešení problému hledání nejdelší společné podposloupnosti je možné využít dynamického programování. Pro vstupní řetězce X a Y délky m a n lze algoritmus popsat následovně:

1. Nechť $A \in \mathbb{R}^{m,n}$
2. Matice A je vyplněna podle následujících pravidel:

$$A_{i,j} = \begin{cases} \emptyset & \text{pokud } i = 0 \text{ nebo } j = 0 \\ A_{i-1,j-1} + 1 & \text{pokud } i, j > 0 \text{ a } X_i = Y_j \\ \max(A_{i-1,j}, A_{i,j-1}) & \text{pokud } i, j > 0 \text{ a } X_i \neq Y_j \end{cases}$$

3. Délka nejdelší společné podposloupnosti se nachází v (i, j)-tém prvku (poslední řádek a poslední sloupec) matice A.

Prvky matice jsou v popisu algoritmu indexovány od 0 (levý horní prvek má index $(0, 0)$). Časová i prostorová složitost tohoto algoritmu je $O(mn)$.

2.4.3.2 Dynamic time warping

Algoritmus dynamic time warping se nejčastěji používá k porovnávání časových řad. Poprvé byl použit k analýze mluveného slova [15]. Algoritmus DTW je možné implementovat pomocí dynamického programování. Pro dva řetězce X a Y délky m a n jej lze popsat následovně:

1. Nechť $A \in \mathbb{R}^{m,n}$
2. Matice A je vyplněna podle následujících pravidel:

$$A_{i,j} = \begin{cases} d(X_i, Y_j) + \min(A_{i-1,j}, A_{i,j-1}, A_{i-1,j-1}) & \text{pokud } i > 0 \\ & \text{nebo } j > 0 \\ 0 & \text{pokud } i = j = 0 \end{cases}$$

kde d značí cenovou funkci. V případě standardizace pomocí výše zmíněných metod je cenová funkce definována následovně:

$$d(a, b) = |a - b|$$

Časová a paměťová složitost tohoto algoritmu je $O(mn)$.

2.4.3.3 Lokální sekvenční alignment

Primární využití sekvenčního alignmentu je v bioinformatice. Spočívá v seřazení dvou či více řetězců tak, aby odpovídající znaky ležely pod sebou. Bývá užíván pro seřazení sekvencí DNA, RNA nebo proteinu.

Pro lokální sekvenční alignment lze použít Smithův-Watermanův algoritmus [16], který využívá dynamického programování. Vstupem algoritmu jsou dvě sekvence X a Y (v tomto případě se jedná o dva řetězce standardizované melodie) délky m a n, bodové ohodnocení/penalizace vložení mezery d, bodové ohodnocení za shodu znaků x a bodová penalizace za neshodu znaků y. Pro zmíněný vstup lze algoritmus popsat následovně:

1. Nechť $A \in \mathbb{R}^{m+1,n+1}$ (matice reálných čísel o velikosti $m + 1 \times n + 1$)
2. Matice A je vyplněna podle následujících pravidel:

$$A_{i,j} = \begin{cases} 0 & \text{pokud } i = 0 \\ & \text{nebo } j = 0 \\ \max(A_{i-1,j} + d, A_{i,j-1} + d, A_{i-1,j-1} + x) & \text{pokud } i, j > 0 \\ & \text{a } X_i = Y_j \\ \max(A_{i-1,j} + d, A_{i,j-1} + d, A_{i-1,j-1} + y) & \text{pokud } i, j > 0 \\ & \text{a } X_i \neq Y_j \end{cases}$$

3. Výsledným skórem je hodnota nejvyššího prvku matice A.

Prvky matice jsou v popisu algoritmu indexovány od 0 (levý horní prvek má index $(0, 0)$). Časová a prostorová složitost tohoto algoritmu je $O(mn)$.

2.4.3.4 Earth mover's distance

Earth mover's distance je metrika primárně využívaná ve statistice pro měření vzdálenosti mezi dvěma pravděpodobnostními rozděleními. V případě, že si rozdělení představíme jako dvě hromady hlíny, lze EMD popsat jako minimální množství práce nutné k přesunutí hlíny z jedné hromady na druhou, aby byl jejich výsledný tvar stejný. Jedna jednotka práce je v tomto případě rovna práci nutné k přesunutí jedné jednotky váhy o jednu jednotku vzdálenosti.

Pro formálnější definici EMD je třeba nejprve definovat prováděcí plán. Nechť P a Q jsou histogramy s m a n prvky, kde P_i a Q_i značí hodnoty i-té skupiny histogramu P a Q. Prováděcí plán je matice F velikosti $m \times n$ s prvky $f_{i,j}$, která splňuje následující:

- $f_{i,j} \geq 0, 1 \leq i \leq n, 1 \leq j \leq n$
- $\sum_{j=1}^{n} f_{i,j} \leq P_i, 1 \leq i \leq m$
- $\sum_{i=1}^{m} f_{i,j} \leq Q_j, 1 \leq j \leq n$
- $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{i,j} = \min(\sum_{i=1}^{m} P_i, \sum_{i=1}^{n} Q_i)$

Dále je nutné definovat matici D o velikosti $m \times n$, jejíž prvky d_{ij} označují vzdálenost mezi i-tou skupinou histogramu P a j-tou skupinou histogramu Q (přesná definice matice vzdálenosti se odvíjí od řešeného problému). Práce nutná k vykonání prováděcího plánu je definována následovně:

$$\mathrm{WORK}(F, P, Q) = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}$$

Nechť $\mathcal{F}(P, Q)$ je množina všech prováděcích plánů mezi histogramy P a Q. EMD je rovno práci nutné k vykonání optimálního prováděcího plánu normalizovaného součtem hodnot nižšího z histogramů:

$$\mathrm{EMD}(P, Q) = \frac{\min_{F \in \mathcal{F}(P,Q)} \mathrm{WORK}(F, P, Q)}{\min(\sum_{i=1}^{m} P_i, \sum_{i=1}^{n} Q_i)}$$

V rámci práce bylo EMD pro porovnání standardizovaných melodií implementováno dvěma odlišnými způsoby. První z nich nejprve převede standardizované melodie na histogramy četností jednotlivých hodnot (výšek not) a následně vypočítá EMD mezi nimi.

Druhý způsob využívá mapování standardizované melodie do dvourozměrného prostoru. Každému z prvků standardizovaných melodií je přiřazeno číslo odpovídající jeho pořadí. Tím vzniknou uspořádané dvojice reprezentující souřadnice jednotlivých prvků ve dvourozměrném prostoru. Funkci pro převod standardizované melodie $M = n_1, \ldots, n_m$ na množinu bodů dvourozměrného prostoru lze definovat následovně:

$$f(M) = \{\forall i \in \{1, \ldots, m\} : (i, n_i)\}$$

Následně jsou pro melodie vytvořeny histogramy na základě četností jednotlivých bodů. Posledním krokem je výpočet EMD mezi histogramy. Prvky matice vzdáleností D mají v tomto případě podobu euklidovských vzdáleností mezi jednotlivými body.

Kapitola 3

Návrh a realizace

Tato kapitola je věnována návrhu a implementaci webové aplikace pro vyhledávání MIDI souborů na základě podobnosti melodií. Popsány jsou požadavky na aplikaci, použité technologie, rozdělení kódu do funkčních celků, použité algoritmy, architektura aplikace a relevantní části kódu.

3.1 Funkční požadavky

F1 Uživatel bude mít možnost zadat dotaz pomocí myši nebo dotykové obrazovky

F2 Uživatel bude moct nahrát dotaz pomocí počítačové klávesnice simulující klaviaturu

F3 Uživatel bude moct nahrát dotaz pomocí MIDI nástroje

F4 Uživatel bude mít možnost svůj dotaz upravit přidáním, smazáním nebo změnou noty

F5 Aplikace zobrazí nejpodobnější výsledky dotazu

F6 Aplikace umožní přehrávání dotazu uživatele

F7 Aplikace umožní přehrávání výsledků vyhledávání

F8 Aplikace umožní zobrazit a upravit noty výsledků vyhledávání

F9 Aplikace umožní spustit nové vyhledávání na základě upraveného výsledku předchozího vyhledávání

F10 Do aplikace bude možné nahrát nové MIDI skladby

F11 Aplikace automaticky zpracuje nově nahrané MIDI skladby

3.2 Nefunkční požadavky

NF1 Systém bude rozdělen na klientskou a serverovou aplikaci

NF2 Uživatelské rozhraní bude mít podobu webové aplikace

NF3 Zpracování dotazů a vyhledávání bude probíhat na serveru

NF4 Klientská a serverová aplikace spolu budou komunikovat přes HTTP

NF5 Uživatelské rozhraní bude responsivní a bude podporovat počítače, tablety a mobilní telefony

NF6 Uživatelské rozhraní bude uzpůsobeno pro práci s myší i pro dotykové obrazovky

NF7 Všechny části systému budou kontejnerizovány

NF8 Aplikace bude umožňovat nezávisle škálovat jednotlivé části

3.3 Použité technologie

Pro implementaci vyhledávače MIDI souborů bylo využito mnoho různých technologií, knihoven a frameworků. Následující část se bude zabývat popisem nejvýznamnějších z nich.

3.3.1 Python, Quart a Hypercorn

Python je vysokoúrovňový skriptovací jazyk. Vyznačuje se jednoduchostí syntaxe a podporou různých programovacích paradigmat, včetně objektově orientovaného, procedurálního a v omezené míře funkcionálního. Je interpretovaným programovacím jazykem, což má za následek jeho nižší výkonnost oproti jiným kompilovaným jazykům.

Quart [17] je mikroframework pro tvorbu asynchronních webových aplikací v jazyce Python. Jedná se o reimplementaci frameworku Flask [18] pomocí asynchronní knihovny asyncio. Tento framework je navržen tak, aby byl snadno použitelný a rozšiřitelný a nekladl příliš mnoho požadavků na architekturu aplikace. Quart podporuje několik různých způsobů zpracování HTTP požadavků a odpovědí, včetně šablonovacího systému Jinja2 a knihoven pro zpracování dat ve formátu JSON. Poskytuje také jednoduchý způsob, jak vytvořit RESTful API. Jeho asynchronní povaha umožňuje využít neblokujícího kódu pro některé I/O operace a tím zefektivnět využití zdrojů serveru. Příkladem takové operace je komunikace s datovou vrstvou nebo paralelní odesílání HTTP požadavků externímu serveru.

Hypercorn [19] je asynchronní webový server. Vyznačuje se vysokým výkonem a efektivním využitím zdrojů serveru. Díky asynchronnímu I/O modelu dokáže obsluhovat více požadavků současně a snižovat tak latenci.

3.3.2 Dramatiq

Dramatiq [20] je nástroj pro asynchronní zpracování úloh v programovacím jazyce Python. Umožňuje vytvářet, distribuovat a řídit úlohy, které lze spouštět asynchronně v pozadí. Podporuje různé technologie pro ukládání úloh a výsledků jako jsou Redis, RabbitMQ nebo PostgreSQL. Umožňuje řídit a konfigurovat jednotlivé workery zpracovávající úlohy. V aplikaci je použit k rozdělování úloh zpracování uživatelských dotazů.

3.3.3 TypeScript a React

TypeScript je programovací jazyk rozšiřující syntaxi jazyka JavaScript o statické typování. To umožňuje eliminovat některé problémy spojené s dynamicky typovanými jazyky jako například obtížnost refaktorování nebo běhové chyby způsobené předáním nesprávného typu. Běžně bývá nazýván nadmnožinou JavaScriptu, neboť jakýkoli validní JavaScript kód je i validní TypeScript kód. TypeScript je kompilován do JavaScriptu, takže je možné ho po kompilaci používat ve většině moderních prohlížečů. Nevýhodou je zvýšení komplexity, kvůli nutnosti kompilačního kroku. Tato problematika je však mitigována moderními systémy pro sestavování jako jsou například Vite nebo Webpack.

React je knihovnou pro tvorbu uživatelských rozhraní pomocí JavaScriptu nebo TypeScriptu. Jeho hlavním cílem je usnadnit tvorbu dynamických a interaktivních rozhraní pro webové aplikace a stránky. Základními stavebními kameny Reactu jsou komponenty reprezentující jednotlivé části aplikace. Samotný React slouží primárně k tvorbě single page aplikací (SPA), které se načítají jako jediná stránka a interakce s nimi probíhají bez nutnosti načítání dalších stránek ze serveru. SPA jsou vhodné pro tvorbu komplexních rozhraní, která vyžadují vysokou úroveň interaktivity. Mezi nevýhody SPA patří delší čas pro inicializaci aplikace, obtížnější optimalizace pro vyhledávače a větší nároky na výkon pro starší a méně výkonná zařízení. Některé z těchto nevýhod dokáží eliminovat tzv. meta frameworky, které využívají Reactu a přidávají funkcionalitu pro podporu vícestránkových aplikací a renderování na straně serveru. Tato aplikace využívá pouze Reactu, neboť pro účely práce je jeho funkcionalita dostačující.

3.3.4 Redis

Redis [21] je in-memory databáze, umožňující ukládání dat do paměti a získávání rychlých výsledků bez nutnosti přístupu k disku. Podporuje mnoho datových struktur včetně řetězců, seznamů, množin, map a binárních objektů. Jedná se o key-value databázi. K datům lze přistupovat pomocí unikátních klíčů. Běžně se používá pro ukládání dat jako jsou uživatelské relace, cachování nebo fronty zpráv. V této aplikaci slouží právě jako fronta zpráv.

3.3.5 MongoDB

MongoDB [22] je dokumentově orientovaná NoSQL databáze, která umožňuje ukládání dat ve formátu BSON (Binary JSON). Hlavním prvkem MongoDB je dokument, což je záznam, který ukládá data v JSON podobě. Dokumenty jsou ukládány v kolekcích. To jsou skupiny dokumentů s podobnými vlastnostmi. V kolekcích mohou být dokumenty s různými strukturami, což umožňuje velkou flexibilitu v ukládání dat. Výhodou je schopnost škálování. Umožňuje horizontální škálování. Může být nasazena na více serverů a data mohou být rovnoměrně rozdělena mezi ně. Tento přístup umožňuje dosažení vysoké dostupnosti a odolnosti proti výpadkům. Datový model MongoDB je vhodný pro data, která mezi sebou nemají příliš mnoho relací. V této aplikaci je použita na místě datové vrstvy. Udržuje předzpracované skladby a záznamy o zpracování úloh.

3.3.6 Google Cloud Storage

Google Cloud Storage (GCS) je objektové úložiště poskytované společností Google, které umožňuje ukládání a správu dat v cloudu. Umožňuje uživatelům ukládat a přistupovat k datům v různých formátech. V aplikaci slouží jako vstupní bod pro nahrávání nových písniček do databáze (písničky jsou staženy z GCP a uloženy do databáze). Je také možné aplikaci nasadit ve variantě, kdy GCS funguje jako databáze skladeb a tím nahrazuje MongoDB.

3.3.7 Nginx

Nginx [23] je webový server a reverzní proxy server, který je navržen tak, aby byl rychlý, spolehlivý a škálovatelný. Nabízí funkce jako je podpora SSL a TLS, výkonnostní optimalizace, řízení toku a vyvažování zátěže. V aplikaci slouží jako reverzní proxy a webový server poskytující statický obsah vygenerovaný sestavením React aplikace.

3.3.8 Docker a Docker Compose

Docker je nástroj pro kontejnerizaci aplikací, který umožňuje spouštět je v izolovaném prostředí v tzv. kontejnerech. Kontejnery jsou balíčky, obsahující vše potřebné pro běh aplikace včetně kódu, knihoven, závislostí a konfigurace. Tato izolace umožňuje spouštět aplikace v různých prostředích bez nutnosti instalovat závislosti na hostitelském systému. Docker umožňuje snadno distribuovat a spouštět aplikace na různých platformách bez nutnosti řešení závislostí a konfigurace pro každou platformu zvlášť. To zjednodušuje nasazení aplikací. Všechny části vytvořené aplikace jsou kontejnerizované pomocí Dockeru, díky čemuž je možné je nasadit mnoha způsoby od jediného centrálního stroje až po distribuovaný Kubernetes cluster.

Pro orchestraci kontejnerů během vývoje a nasazení je použito technologie Docker Compose. Docker Compose umožňuje definovat kontejnery, sítě a další konfigurace pomocí YAML souborů a následně je spouštět a spravovat pomocí příkazů v příkazové řádce. Jednou z funkcí je možnost definovat více konfigurací pro různá prostředí. Toho je v aplikaci využito pro oddělení konfigurací pro lokální vývoj, testovací prostředí v CI/CD a nasazení.

3.4 Předzpracování MIDI souborů

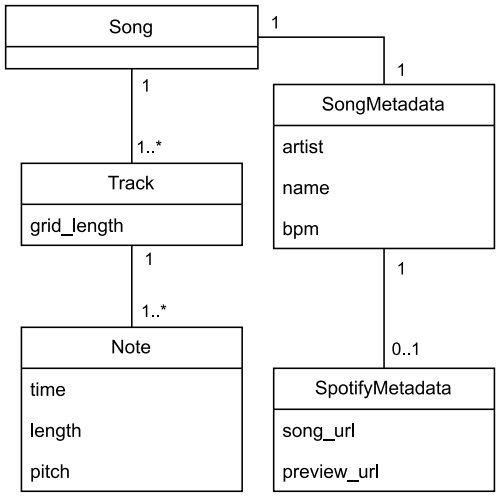
MIDI soubory je nutné před použitím předzpracovat, neboť tento formát není pro zvolený způsob vyhledávání vhodný. Pro potřeby vyhledávače jsou některé v něm obsažené informace redundantní a jiné jsou nevhodně formátované.

K načítání MIDI souborů je použito knihovny Miditoolkit [24], která dekóduje binární data a převede je na datové struktury jazyka Python reprezentující MIDI události. Dekódované události jsou rozděleny podle stop. Dalším krokem je převedení dekódovaných událostí do formy, která je vhodná k porovnání.

Nejprve je určeno tempo skladby podle první Set Tempo události, která se v MIDI souboru vyskytne. Další Set Tempo události nejsou brány v potaz. Pokud je TPQN skladby jiné než 480, jsou všechny časové údaje přeškálovány. To je z důvodu, aby bylo TPQN napříč celým datasetem jednotné a bylo možné se všemi skladbami pracovat stejně. Škálování časového údaje na 480 TPQN je definováno následovně:

$$s(t) = \left\lfloor \frac{480}{p} \cdot t \right\rfloor$$

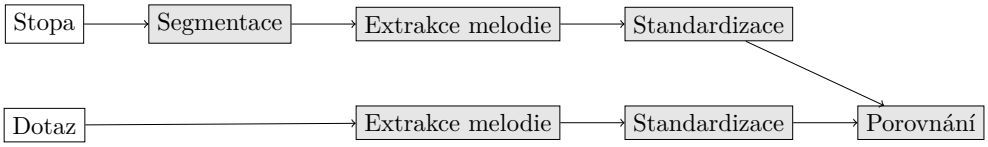
Kde t je časový údaj a p je původní TPQN škálované skladby. Časové údaje událostí jsou v MIDI souborech zaznamenány relativně k předchozí události. Relativní údaje jsou převedeny na absolutní výpočtem posloupnosti částečných součtů. Posledním krokem je extrakce názvu skladby a interpreta. Ta probíhá na základě jména souboru. Názvy souborů datasetu mají podobu `Umělec - Skladba.mid`, jejich zpracování je tedy triviální. Z předzpracovaných dat jsou vytvořeny instance tříd popsaných diagramem 3.1. Instance jsou následně serializovány a ukládány buď do databáze MongoDB nebo jako binárně serializované objekty jazyka Python.



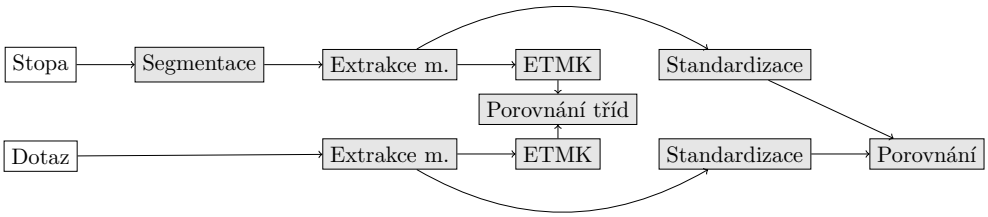
Obrázek 3.1: Diagram tříd předzpracované skladby

3.5 Struktura vyhledávače

Vyhledávač se skládá z několika částí (modulů), kde každá zastává jiný úkol. Jednotlivé moduly lze vyměnit za jiné. V některých případech bylo v rámci práce implementováno několik alternativ. Zpracování dotazu a porovnání se stopou v databázi může mít jednu ze dvou podob popsanou na obrázcích 3.2 a 3.3.



Obrázek 3.2: Porovnání dotazu se stopou v databázi



Obrázek 3.3: Porovnání dotazu při použití tříd melodických kontur — Standardizace a porovnání proběhne pouze pro segmenty, které sdílí alespoň třetinu tříd melodických kontur s dotazem (podrobněji popsáno v části 3.5.5). ETMK — Extrakce tříd melodických kontur

Struktury vyhledávače i jednotlivé moduly lze za běhu aplikace vyměnit. Každý přináší jisté výhody a nevýhody. V následující části budou popsány jednotlivé části vyhledávače a jejich implementace.

3.5.1 Segmentace

V rámci práce byly implementovány dva segmentační moduly. První z nich je velmi jednoduchý. Skladbu nerozděluje, jednotlivé stopy pouze „ořeže“, tedy ze začátku a konce stopy ubere části, které neobsahují žádné noty. Slouží převážně pro poskytování referenčních hodnot pro porovnání s dalšími segmentačními přístupy.

Druhý přístup rozděluje stopu na základě délky dotazu. Všechny segmenty jsou stejně dlouhé jako dotaz a navzájem se nepřekrývají. Tento způsob segmentace není příliš složitý, v kombinaci s ostatními použitými metodami však poskytuje dobré výsledky. Pro použití v aplikaci je vhodný, neboť uživatelské rozhraní umožňuje zadávat dotazy různých délek.

3.5.2 Extrakce melodie

Modul pro extrakci melodie je relevantní v případech, kdy je stopa polyfonická. Extrakce je implementována velmi jednoduše. Postupně je iterováno přes všechny noty a v případě, že se některé z nich překrývají (vyskytují se ve stejném čase) je ta nižší z nich odstraněna. Implementace modulu vychází z analýzy problematiky popsané v části 2.4.1.

3.5.3 Standardizace melodie

Standardizace melodie je důležitým krokem vyhledávání, neboť umožňuje převést všechny melodie na podobnou reprezentaci. Kromě toho také umožňuje odstranit některé nežádoucí vlastnosti a převést jiné na obecnější úroveň. Standardizační modul využívá metody popsané v části 2.4.2. V rámci práce bylo implementováno několik verzí modulu umožňující standardizaci pomocí absolutní výšky, relativních intervalů, Parsonsova kódu a baseline intervalů. Pro standardizaci je použit vždy právě jeden ze zmíněných přístupů. Výchozí volbou je metoda standardizace pomocí relativních intervalů.

3.5.4 Porovnání

Modul pro porovnávání skladeb pracuje nad melodiemi předzpracovanými výše popsanými komponenty vyhledávače. K porovnávání skladeb s dotazem využívá podobnostních metrik popsaných v části 2.4.3. Celkem bylo v rámci práce implementováno 5 různých verzí tohoto modulu: nejdelší společná podposloupnost, DTW, lokální sekvenční alignment, EMD pro četnost výšek not melodií a EMD s mapováním melodie do 2D prostoru.

3.5.5 Třídy melodické kontury

Tato metoda vychází z metody popsané Weydem a Datzkem [25], kteří se zabývají tvorbou indexu nad hudební databází obsahující hudbu zapsanou symbolickou notací. Weyd a Datzke oproti této implementaci využívají jiný způsob segmentace a kromě zjednodušené reprezentace výšek not melodie berou při tvorbě indexu v potaz i rytmus. V této práci je použita pouze k reprezentaci výšek not a neslouží k tvorbě indexu, nýbrž jako jeden z kroků vyhledávání.

Metoda tříd melodických kontur využívá Parsonsův kód k reprezentaci melodie. Nejprve je melodie rozdělena na překrývající se n-gramy o délce pět. Melodie na obrázku 2.2 — v Parsonsově kódu reprezentována R U U U D D U — by byla rozdělena na následující tři n-gramy: R U U U D, R U U D D, R U D D U. Všechny řetězce Parsonsova kódu reprezentující melodii jsou zakódovány do binární podoby, která je převedena na číselnou reprezentaci v desítkové soustavě. Výsledná množina čísel v desítkové soustavě reprezentuje jednotlivé třídy melodické kontury. Funkci transformace melodie reprezentované řetězcem $M = n_1, \ldots, n_m$, kde n_i označuje výšku noty melodie, na třídu melodické kontury lze popsat následovně:

$$e(M) = \begin{cases} e(n_1, \ldots, n_{m-1}) \cdot 2^2 + b(n_{m-1}, n_m) & \text{pokud } m \geq 2 \\ 10b & \text{pokud } m = 1 \\ 0 & \text{pokud } m = 0 \end{cases}$$

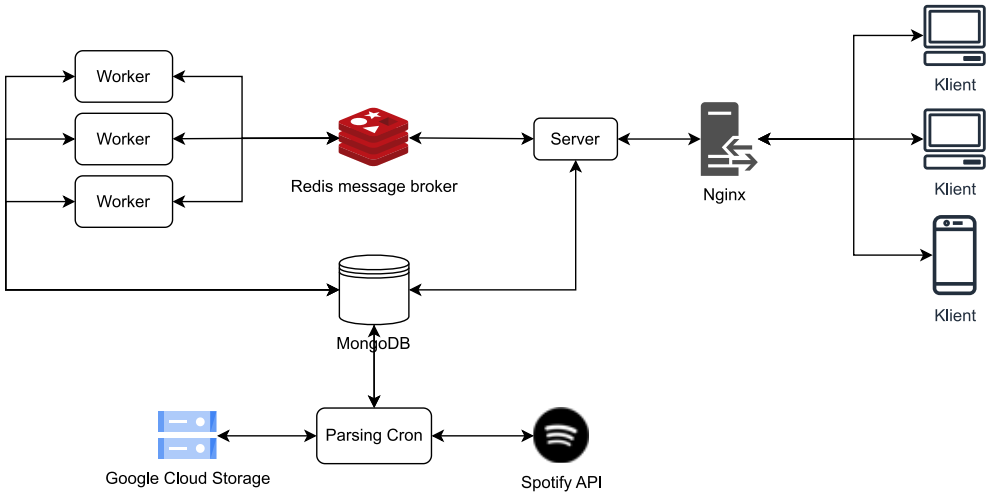
$$b(n_1, n_2) = \begin{cases} 01b & \text{pokud } n_1 < n_2 \\ 10b & \text{pokud } n_1 = n_2 \\ 11b & \text{pokud } n_1 > n_2 \end{cases}$$

Melodie na obrázku 2.2 by byla reprezentována množinou tříd melodických kontur $\{599, 607, 637\}$.

Tato metoda je využita pro zrychlení vyhledávání. Před samotným porovnáváním pomocí výše zmíněných podobnostních metrik jsou eliminovány ty segmenty, které s uživatelským dotazem nesdílí alespoň třetinu tříd melodických kontur. Tento přístup je výhodný, neboť porovnání společných tříd melodických kontur je rychlejší operace, než porovnávání pomocí podobnostních metrik.

3.6 Architektura aplikace

Aplikace se skládá z několika propojených komponent. V následující části budou popsány jednotlivé komponenty a způsob, jakým mezi sebou komunikují.



Obrázek 3.4: Komunikace jednotlivých komponent aplikace

3.6.1 Nginx

Nginx plní dvě úlohy. Tou první je poskytování statického obsahu (HTML, CSS, JavaScript a obrázky) uživatelům. Druhou je reverzní proxy. Všechny požadavky z webové aplikace jsou směřovány na Nginx, který je předá serveru k dalšímu zpracování. Tento komunikační model mezi klientskou aplikací a serverem zjednodušuje problematiku zajištění šifrovaného spojení, cachování dotazů a vyvažování zátěže. Šifrované spojení je navázáno pouze mezi webovou aplikací a Nginx.

3.6.2 Server

Server přijímá HTTP požadavky z webové aplikace. Na základě požadavků vytváří nové úlohy a vkládá je do fronty. Kromě vytváření úloh se také dotazuje databáze na stav běžících úloh a případně vrací výsledky již hotových úloh. Komunikace mezi serverem a klientskou webovou aplikací prochází přes Nginx. S workery komunikuje přes frontu zprostředkovávanou Redis.

3.6.3 Redis message broker

Redis slouží jako fronta pro ukládání úloh určených ke zpracování. Skrze tuto frontu je zprostředkovávána komunikace mezi serverem a workery.

3.6.4 Worker

Úkolem workerů je zpracování vytvořených úloh ve frontě (každá úloha odpovídá jednomu uživatelskému dotazu). Každý worker zpracovává v jeden moment vždy maximálně jednu úlohu. Workeři kromě fronty úloh komunikují

také s databází. Z té získávají písničky pro porovnání s uživatelským dotazem a zapisují do ní výsledky vyhledávání.

3.6.5 MongoDB

Databáze slouží k ukládání předzpracovaných písniček a k udržování informací o stavu jednotlivých úloh. Obsahuje pouze dvě entity. Podobu jednotlivých dokumentů, které je reprezentují ilustrují výpisy 3.1 a 3.2.

```
{
    "_id": "640de4fe59ecd96ce1ace4ef",
    "tracks": [
        {
            "notes": [...]
            "grid_length": ...
        },
        ...
    ],
    "metadata": {
        "artist": "Good Tiger",
        "name": "Sunthrower Flower",
        "bpm": 140,
        "spotify": {
            "preview_url": ...,
            "song_url": ...
        }
    }
}
```

Výpis kódu 3.1: Dokument písničky

```
{
    "_id": "640de4f06d5523d2c881733e",
    "status": "completed",
    "results": [
        "metadata": {
            "artist": "LCD Soundsystem",
            "name": "Dance Yrself Clean",
            "bpm": 98,
            "spotify": {
                "preview_url": ...,
                "song_url": ...
            }
        },
        "track": {
```

```
            "notes": [...],
            "grid_length": ...
        }
        ...
    ]
}
```

Výpis kódu 3.2: Dokument úlohy

3.6.6 Parsing cron

Úkolem této komponenty je předzpracování písniček a jejich ukládání do databáze. V hodinových intervalech je spouštěna úloha skládající se ze dvou částí. V první části jsou z úložiště staženy MIDI soubory, které jsou následně předzpracovány do formy vhodné pro další použití v rámci aplikace a uloženy do databáze. Ve druhé části jsou jednotlivým písničkám přiřazeny URL adresy, pomocí kterých je lze nalézt na Spotify a přehrát jejich náhled.

3.7 Zpracování dotazu

Uživatelem zadaný dotaz do uživatelského rozhraní je nejprve v těle HTTP metody POST předán serveru. Server vloží do databáze dokument obsahující informace o stavu úlohy a jejím výsledku. Úlohy mohou být ve stavu „pending“ — úloha ještě nebyla zpracována nebo „completed“ — úloha již byla zpracována a její výsledek byl uložen do databáze. Při vytvoření je stav úlohy nastaven na „pending“ a její výsledek je `null`. Server vloží do fronty novou úlohu obsahující uživatelem zadaný dotaz. Jako odpověď na výše zmíněný HTTP požadavek jsou odeslány informace o vytvořené úloze a identifikátor, pomocí kterého je možné se dotazovat na její výsledek. Klient se následně periodicky dotazuje serveru pomocí metody GET a získaného identifikátoru, zda již byla úloha splněna. Server se na stav a případné výsledky úlohy dotazuje databáze. V případě, že úloha nebyla splněna, má odpověď serveru následující podobu:

```
{
    "id": "640de51177b857a46c062d27",
    "status": "pending",
    "results": null
}
```

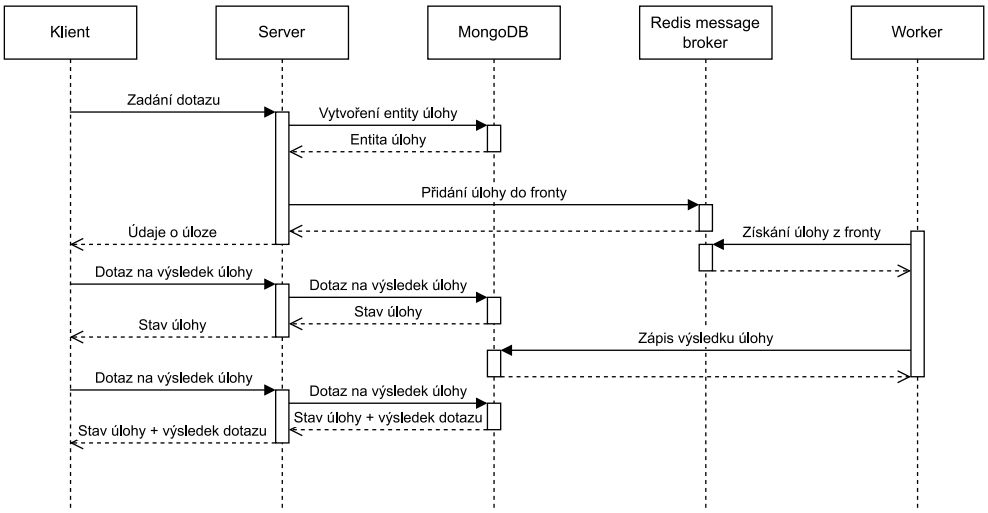
Výpis kódu 3.3: Odpověď serveru — nevyřízený dotaz

V případě, že splněna byla (úloha je ve stavu „completed“), bude odpověď serveru kromě informací o úloze obsahovat i výsledky vyhledávání:

```
{
    "id": "640de51b1f1ba76a0478e223",
    "status": "completed",
    "results": [
        {
            "artist": "Hella",
            "name": "1-800-GHOST-DANCE",
            "notes": [...]
            ...
        }
        ...
    ]
}
```

Výpis kódu 3.4: Odpověď serveru — vyřízený dotaz

Úlohy (dotazy uživatelů) zařazené do fronty jsou postupně odebírány a zpracovávány workery. Worker úlohu z fronty odebere, zpracuje ji (vyhledá podobné skladby) a následně aktualizuje informace v databázi (nastaví stav úlohy na „completed“ a uloží její výsledky). Celý proces zpracování jednoho požadavku je znázorněn sekvenčním diagramem 3.5.



Obrázek 3.5: Zpracování dotazu

3.8 Popis kódu serverové části aplikace

Kód je členěn do několika logických celků (balíčků). Každý logický celek zastává část funkcionality aplikace. V následující části bude popsáno rozložení kódu a budou vyzdvihnuty jeho důležité části.

3.8.1 common

Balíček `common` obsahuje společné části kódu, které používají ostatní balíčky. Obsahuje třídy poskytující rozhraní pro komunikaci s databází, třídy implementující algoritmy vyhledávání písniček, definuje používané datové typy a pomocné metody. Důležitou součástí balíčku jsou také metody pro konfiguraci jednotlivých částí aplikace, jako je například logování.

3.8.2 worker

Balíček `worker` obsahuje funkcionalitu nutnou pro nastavení a spuštění worker procesů. V tomto balíčku je definována úloha, kterou procesy vykonávají při vyhledávání.

3.8.3 midi_scraper

Balíček `midi_scraper` obsahuje funkce používané pro scraping datasetu a jeho obohacení o URL získané ze Spotify API.

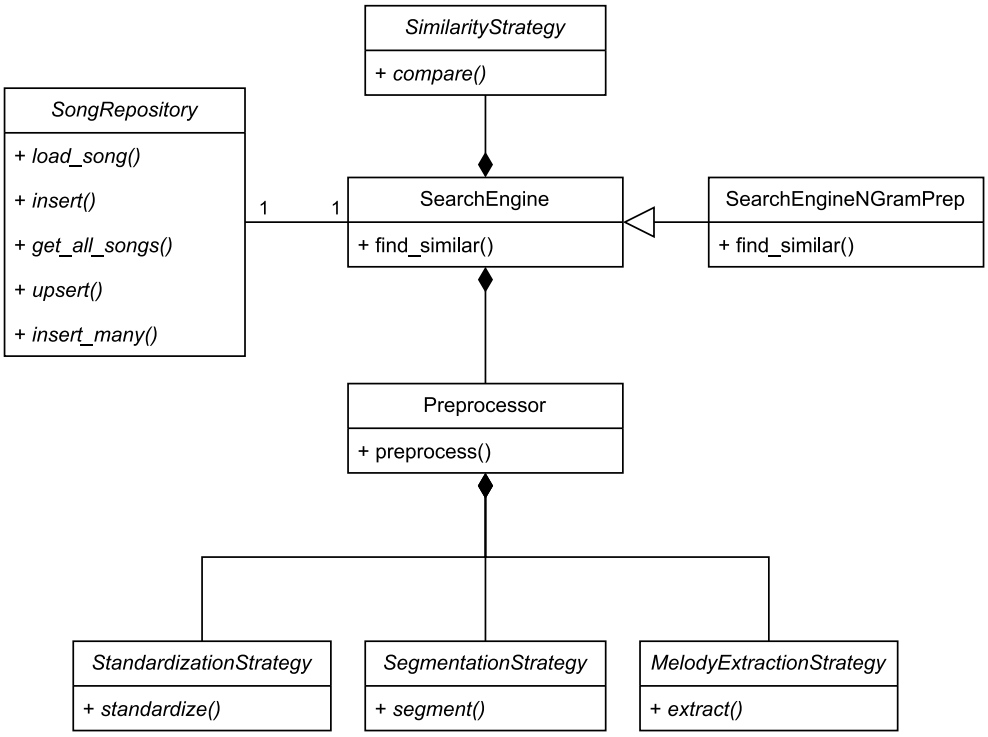
3.8.4 lab

Tento balíček obsahuje kód související s analýzou chování použitých algoritmů. Jeho součástí jsou funkce umožňující měření času a kvality jednotlivých částí aplikace. Kromě kódu pro samotné měření také obsahuje kód sloužící ke generování testovacích dat.

3.8.5 Důležité třídy

Kód serverové části aplikace je napsán s důrazem na objektově orientované programovací paradigma. Využívá principů kompozice, dědičnosti, polymorfismu a zapouzdření. Kromě balíčků popsaných výše je kód členěn ještě na třídy.

Jednou z nejdůležitějších tříd je `SearchEngine`, která zapouzdřuje funkcionalitu pro vyhledávání skladeb a poskytuje nad ní jednoduché rozhraní. Skládá se z několika komponent. Vztahy mezi nimi popisuje diagram 3.6. Třída s názvem `Preprocessor` má za úkol předzpracování skladeb. Je komponována ze tří dalších tříd — `StandardizationStrategy`, `SegmentationStrategy` a `MelodyExtractionStrategy`, které pomocí návrhového vzoru Strategy zapouzdřují algoritmy pro předzpracování melodie popsané výše v částech 3.5.1, 3.5.2 a 3.5.3. Tyto třídy jsou abstraktní a popsané algoritmy jsou implementovány v konkrétních třídách, které od nich dědí. Stejného návrhového vzoru je využito pro `SimilarityStrategy`, jejíž úlohou je porovnávání již předzpracovaných skladeb. Podtřídy `SimilarityStrategy` implementují algoritmy popsané v části 3.5.4.



Obrázek 3.6: Diagram tříd `SearchEngine`

Další důležitou částí `SearchEngine` je abstraktní třída `SongRepository`. Ta poskytuje jednotné rozhraní pro přístup k databázi skladeb. Abstrakce databázových operací umožňuje implementaci několika různých přístupů k ukládání dat. Metody třídy `SongRepository` jsou implementovány třemi způsoby. První a nejjednodušší z nich využívá lokálního souborového systému. Skladby jsou ukládány na disk ve formě binárních souborů, které obsahují serializované objekty jazyka Python. Tato implementace je využívána při měření a testování jednotlivých algoritmů viz. kapitola 4. Druhá implementace funguje velmi podobně, avšak namísto lokálního souborového systému využívá úložiště Google Cloud Storage. Tato konkrétní třída umožňuje nasazení aplikace bez použití databáze. Třetí konkrétní třída implementující metody `SongRepository` zprostředkovává komunikaci s databází MongoDB a je výchozí volbou při nasazení aplikace.

Třída `SearchEngineNGramPrep` je podtřídou `SearchEngine`. Implementuje metody pro extrakci tříd melodických kontur popsané v části 3.5.5 a přidává do vyhledávání krok porovnání extrahovaných tříd (viz. diagram 3.2).

3.9 Testování

Testování serverové aplikace probíhalo formou unit testů napsaných za pomoci knihovny Pytest. Jejich hlavním předmětem byly výše popsané třídy `SearchEngine` a `SearchEngineNGramPrep`. Jelikož byly implementovány kompozicí několika komponent, bylo možné jednoduše otestovat každou jejich část zvlášť. Pro všechny implementované algoritmy extrakce melodie, standardizace, segmentace a porovnání byly vytvořeny testy ověřující jejich korektnost. Pro samotné třídy `SearchEngine` a `SearchEngineNGramPrep` byly vytvořeny testy ověřující, že kompozice komponent funguje dle očekávání. Kromě testování tříd zapouzdřujících funkcionalitu vyhledávače obsahuje aplikace i unit testy ověřující správnost podpůrných tříd, které se starají o načítání MIDI souborů, serializaci a deserializaci entit, komunikaci s databází a zpracování uživatelských dotazů. Databázová připojení a souborový systém jsou v unit testech simulovány pomocí mock objektů.

Testování klientské aplikace probíhalo manuálně. Po implementaci nové funkcionality byla vždy ověřena její správnost. Na konci práce proběhlo manuální testování klientské aplikace ověřující, že jsou splněny všechny stanovené funkční požadavky.

3.10 Kontinuální integrace a nasazení

Následující část je věnována implementaci kontinuální integrace a nasazení (CI/CD) aplikace. CI/CD je souhrnné označení pro postupy a nástroje, sloužící k automatizaci vývoje a nasazování softwaru. Hlavním cílem je urychlení a zefektivnění opakovaných procesů. Tato část je zaměřena na popis jednotlivých fází CI/CD.

CI/CD probíhá pro klientskou a serverovou aplikaci odlišně. Kontinuální integrace v případě klientské React aplikace se skládá ze dvou kroků. Prvním je statická analýza kódu pomocí nástroje ESLint. Ta napomáhá odhalit chybnou syntax, nekonzistenci kódu, nepoužití deklarovaných proměnných, nesprávné použití operátorů a další. Ve druhém kroku je zkompilován TypeScript kód a vytvořena optimalizovaná verze aplikace pomocí nástroje Vite. Vygenerovaný JavaScript, HTML a CSS jsou poté použity při sestavování kontejneru s webovým serverem Nginx. Sestavený kontejner je nahrán do registru kontejnerů.

Kontinuální integrace serverové aplikace se skládá ze tří kroků. Prvním z nich je sestavení kontejneru. V kontejneru jsou nainstalovány všechny závislosti a je nahrán do registru kontejnerů. Všechny následující kroky probíhají uvnitř vytvořeného kontejneru, aby testovací prostředí co nejvíce odpovídalo tomu produkčnímu. Nejprve je provedena statická analýza kódu pomocí nástrojů Flake8 a Mypy. Flake8 identifikuje chyby jako např. nesprávná syntaxe, nesprávné odsazení, použití nesprávného stylu nebo nevyužité proměnné.

Protože je Python dynamicky typovaný jazyk, mohou být některé typové chyby odhaleny až za běhu programu. Tomu je možné předejít za pomoci typových anotací a nástroje Mypy. Mypy kontroluje, zda jsou proměnné, argumenty funkcí a návratové hodnoty správného typu a zda jsou používány správným způsobem. Třetí fází kontinuální integrace serverové aplikace je spuštení unit testů. Ty slouží k ověření správnosti funkcionality jednotlivých částí aplikace. Pro testování je využito knihovny Pytest.

Kontinuální nasazení probíhá pomocí Docker Compose. Docker umožňuje spouštění příkazů na vzdáleném stroji pomocí SSH. Při kroku nasazení dojde k připojení na vzdálený virtuální stroj a spuštění kontejnerů, které obsahují aktualizovaný kód, konfiguraci a závislosti.

3.11 Škálovatelnost

Architektura aplikace byla navržena s důrazem na škálovatelnost. Rozdělení aplikace na několik komponent a omezení zodpovědnosti každé z nich umožňuje škálovat jednotlivé části nezávisle na sobě. Výhodou zvolených technologií je také, že každá část systému může bez problémů běžet na úplně jiném stroji a vytvořit tak distribuovaný systém odolný vůči selhání některé z komponent.

Výpočetně nejnáročnější částí aplikace jsou workeři, jimž jsou delegovány úkoly nutné pro zpracování uživatelských dotazů. Díky zvolenému komunikačnímu modelu mezi workery a serverem je možné je škálovat jak vertikálně, tak horizontálně a to i napříč několika stroji. Protože komunikace je zprostředkována přes frontu úloh a výsledky jsou ukládány do databáze, je přidání dalšího workera triviální a vyžaduje pouze minimální konfiguraci. Vertikální škálovatelnost workerů spočívá v možnosti nastavení počtu procesů využívaných k paralelnímu zpracování dotazů (viz. část 3.12).

V případě potřeby je horizontální škálovatelnost serveru zajištěna funkcionalitou pro vyvažování zátěže Nginx. Díky tomu, že všechny požadavky prochází přes Nginx fungující jako reverzní proxy, je možné nakonfigurovat seznam serverů, kam by měly být požadavky rozesílány a Nginx se postará o jejich rovnoměrné rozložení.

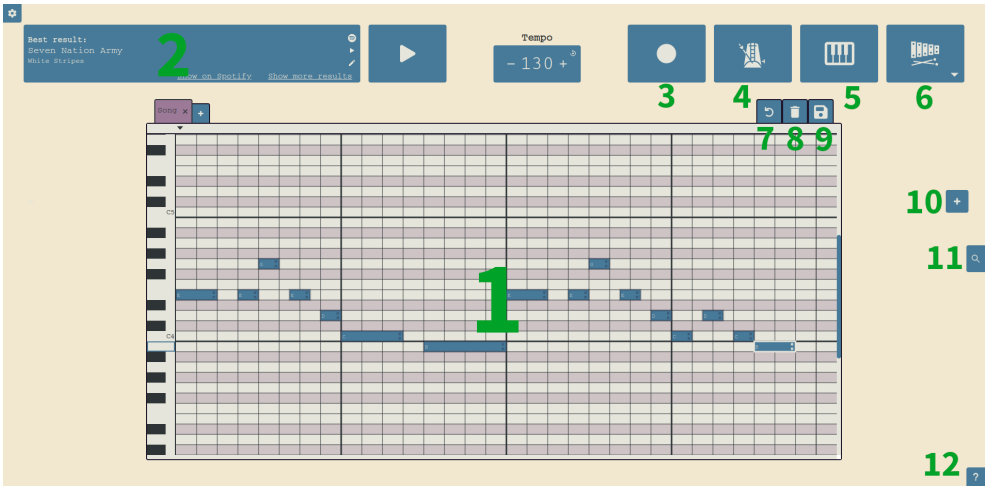
3.12 Paralelizace vyhledávání

Výše popsaný model vyhledávání je velmi snadno paralelizovatelný. Toho je dosaženo díky tomu, že předzpracování a porovnávání dvou skladeb může probíhat plně v izolaci a nejsou k němu potřeba výsledky žádných jiných výpočtů.

Celý proces zpracování uživatelského dotazu lze rozdělit na tři části. První část zahrnuje předzpracování uživatelského dotazu (extrakce melodie, standardizace a případná extrakce tříd melodických kontur) a probíhá synchronně. Druhá část se skládá z načítání, předzpracování a porovnávání jednotlivých

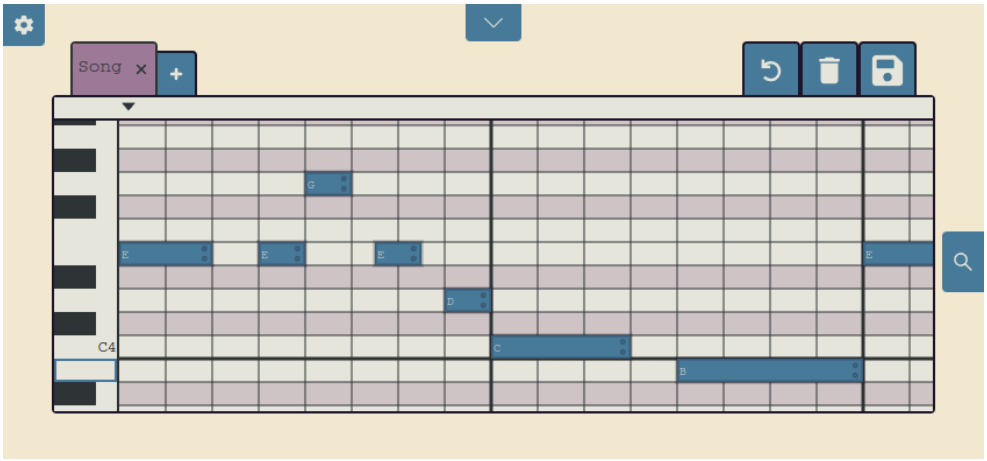
skladeb v databázi s uživatelským dotazem. Tato část probíhá paralelně. Databáze skladeb je rozdělena na několik částí a každá z nich je přidělena jednomu procesu. Procesy provedou nutné výpočty a naleznou ve své přidělené části databáze ty nejpodobnější skladby. Poslední část výpočtu probíhá synchronně. Jejím cílem je agregovat výsledky jednotlivých procesů a vybrat z nich celkově nejpodobnější skladby.

3.13 Uživatelské rozhraní



Obrázek 3.7: Rozhraní vyhledávače

1. Pianoroll pro zadávání melodie dotazu
2. Melodie nejpodobnější dotazu — lze přehrát náhled z databáze a náhled na spotify nebo otevřít melodii k editaci
3. Spuštění/zastavení nahrávání — nahrávat lze pomocí MIDI kontroléru, počítačové klávesnice nebo klaviatury na obrazovce otevřené pomocí tlačítka 5.
4. Spuštění/vypnutí metronomu
5. Otevření/zavření klaviatury na obrazovce
6. Změna zvuku (nemá vliv na vyhledávání, pouze na to, jak zní přehrávané noty)
7. Zpět
8. Smazání not v aktuální záložce
9. Export melodie ve formátu Ogg
10. Prodloužení dotazu
11. Zobrazení výsledků dotazu
12. Spuštění tutoriálu



Obrázek 3.8: Rozhraní vyhledávače — mobilní telefon

Uživatelské rozhraní vyhledávače má podobu webové aplikace. Umožňuje zadávat a nahrávat melodie a zobrazovat podobné skladby. Je optimalizované pro velké obrazovky i pro mobilní zařízení viz. obrázek 3.8. Jednotlivé funkce jsou popsány na obrázku 3.7.

Nejdůležitější částí uživatelského rozhraní je pianoroll (na obrázku 3.7 označen číslem 1), který umožňuje pomocí myši nebo dotykové obrazovky přidávat a odebírat noty dotazu. Tato funkcionalita je implementována pomocí stavového automatu popsaného diagramem 3.9. `LeftClick`, `RightClick`, `LeftRelease`, `RightRelease` a `Move` značí události myši, při kterých dochází k přechodu do jiného stavu:

`LeftClick` — stisknutí levého tlačítka myši

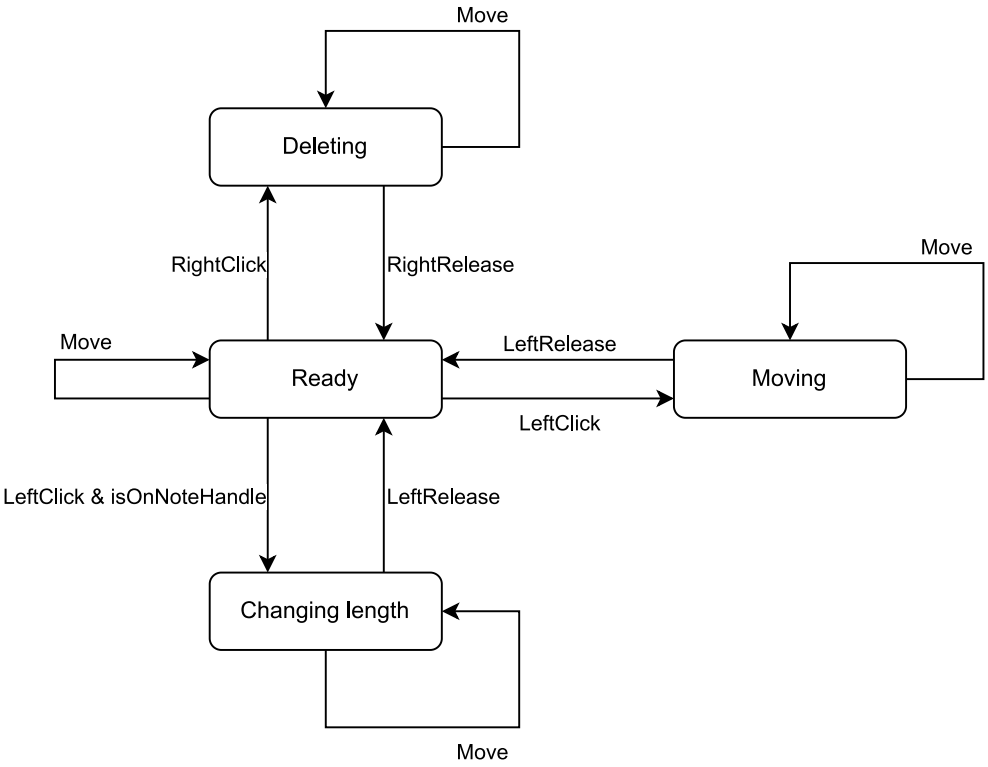
`RightClick` — stisknutí pravého tlačítka myši

`LeftRelease` — uvolnění levého tlačítka myši

`RightRelease` — uvolnění pravého tlačítka myši

`Move` — pohyb myši

V případě, že dojde k události, pro kterou není označen ze současného stavu přechod, je událost ignorována (např. automat se nachází ve stavu `Moving` a dojde k události `RightClick`). Stav `Ready` je počáteční. Ve stavu `Ready` nejsou vykonávány žádné akce. Přechod ze stavu `Ready` do stavu `Moving` může nastat dvěma způsoby. V případě, že došlo k levému kliknutí na notu, přejde se do stavu `Moving` a každý další pohyb myši „nakliknutou“ notu přesune na jiné místo. Pokud došlo ke zmáčknutí levého tlačítka mimo notu, je před přechodem do stavu `Moving` na místě kliknutí nota vytvořena. K přechodu do stavu `Changing length` dochází, pokud došlo k levému kliknutí na „handle“ noty (nejpravější část označená dvěma tečkami). Ve stavu `Changing length`



Obrázek 3.9: Stavový automat pro ovládání pianorollu

pohyb myši mění délku noty. K přechodu do stavu `Deleting` dochází po stisknutí pravého tlačítka myši. V tomto stavu jsou smazány všechny noty, na které je najeto myší.

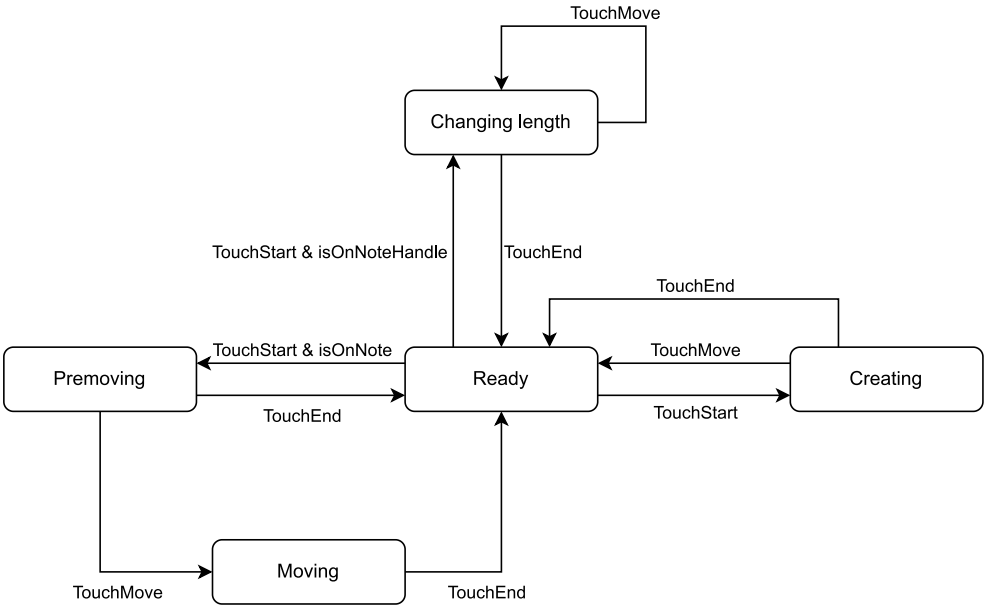
Dotykové obrazovky nedisponují pravým tlačítkem, ani kolečkem myši pro posouvání pianorollu. Je tedy nutné upravit ovládání. To popisuje diagram stavového automatu 3.10. Události pro přechod mezi stavy mají následující význam:

`TouchStart` — dotyk obrazovky

`TouchEnd` — ukončení dotyku obrazovky

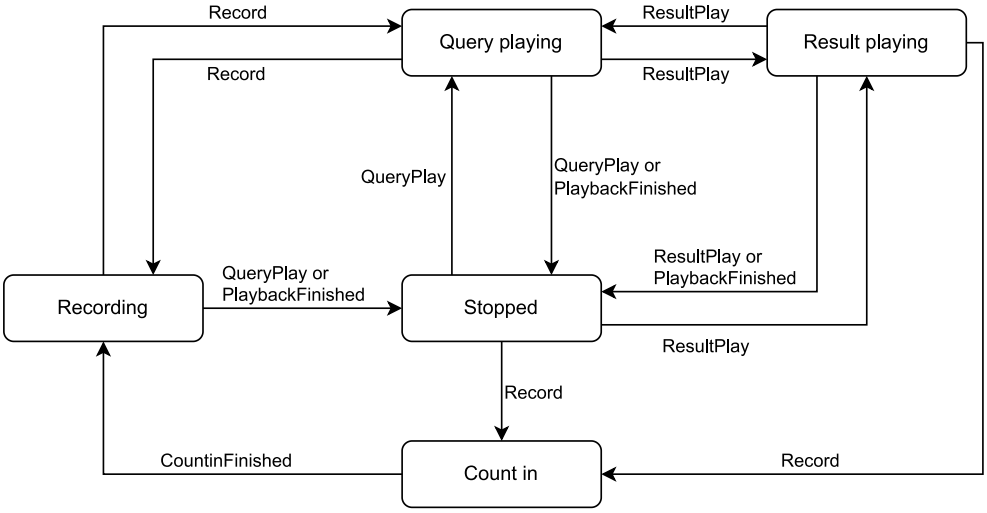
`TouchMove` — pohyb dotyku po obrazovce

Stejně jako u předchozího diagramu je počátečním stavem `Ready`. Při dotyku obrazovky mimo notu se ze stavu `Ready` přejde do stavu `Creating`. Pokud ve stavu `Creating` dojde k události `TouchEnd`, je vytvořena nová nota a automat se navrátí do stavu `Ready`. Pokud ve stavu `Creating` dojde k události `TouchMove`, nota vytvořena není. Tímto způsobem jsou odlišena gesta pro vytváření not a skrolování. Stav `Premoving` slouží k odlišení gest



Obrázek 3.10: Stavový automat pro ovládání pianorollu — dotyková obrazovka

pro smazání a přesunutí noty. Do tohoto stavu se přechází, pokud dojde k dotyku obrazovky v místě některé z not. V případě, že ve stavu `Premoving` dojde k události `TouchEnd` je nota, na které došlo k dotyku smazána. Pokud v tomto stavu dojde k události `TouchMove`, přejde se do stavu `Moving` a zvolená nota je přesouvána podle pohybu dotyku po obrazovce. Posledním stavem je `Changing length`. Do něj se přechází při dotyku „handle“ noty a pohybem dotyku dochází k prodlužování nebo zkracování noty.



Obrázek 3.11: Stavový automat pro ovládání přehrávání a nahrávání

Kromě zpracování uživatelského vstupu v rámci pianorollu, musí rozhraní správně reagovat i na požadavky na nahrávání nových melodií a přehrávání dotazu a výsledků vyhledávání. Tato funkcionalita je implementována pomocí stavového automatu znázorněného diagramem 3.11. K přechodu mezi stavy dochází při následujících událostech:

`Record` — stisknutí tlačítka nahrávání (na obrázku 3.7 označeno číslem 3)

`QueryPlay` — stisknutí tlačítka spustit/zastavit přehrávání dotazu

`ResultPlay` — stisknutím tlačítka spustit/zastavit přehrávání výsledku (na obrázku 3.7 se nachází v části označené číslem 2)

`CountinFinished` — skončení odpočtu před nahráváním

`PlaybackFinished` — přehrávání výsledku nebo dotazu dokončeno

Počátečním stavem je `Stopped`. V tomto stavu nedochází ani k přehrávání, ani k nahrávání. Pokud se automat nachází ve stavu `Query playing` nebo `Result playing` dochází k přehrávání uživatelského dotazu nebo výsledku vyhledávání. Stav `Count in` označuje odpočet před začátkem nahrávání. Ve stavu `Recording` dochází k zaznamenávání not zahraných uživatelem na počítačovou klávesnici nebo připojený MIDI nástroj. Tento stavový automat zajišťuje, že nebude docházet k nechtěnému přehrávání několika stop najednou a nahrávání nebude spouštěno v situacích, kdy to není žádoucí. Implementace funkcionality pomocí automatu zaručuje, že se aplikace nebude nacházet ve stavu, který by z hlediska použití nedával smysl (například zároveň spuštěné nahrávání dotazu a přehrávání výsledku vyhledávání).

Kapitola 4

Experimentální část

V této části jsou popsány metody a data, kterých bylo využito při evaluaci různých přístupů vyhledávání. Hodnocení algoritmů probíhalo s důrazem na jejich užitečnost pro výslednou aplikaci. Dále jsou prezentována a interpretována naměřená data, na jejichž základě byly voleny algoritmy pro výslednou aplikaci.

4.1 Dataset

Použitý dataset vznikl sloučením dvou zdrojů. Prvním je Rob's midi file library [26], ze kterého pochází 837 skladeb. Druhým je 150 nejpopulárnějších písniček ze stránky Free Midi [27]. Dataset obsahuje celkem 987 skladeb. Celkový počet stop činí 10636 (9170 bez rytmických nástrojů a perkusů). Obsaženy jsou skladby mnoha žánrů od popu až po metal. Nepředzpracovaný dataset má podobu MIDI souborů.

4.1.1 Testovací data

Před samotným měřením je nejprve potřeba vytvořit sadu dat, která bude sloužit k ověření výsledků. V tomto případě má podobu oanotovaných dotazů. Jsou to dotazy, pro které je předem známé, jaký mají mít výsledek. Testovací data jsou vytvořena s cílem se co nejvíce přiblížit dotazům uživatelů.

Tvorba testovacích dotazů probíhala v několika fázích. Nejprve bylo pro každou požadovanou délku (2 takty, 4 takty, 6 taktů a 8 taktů) z datasetu vybráno 150 náhodných segmentů. Další fází bylo vytvoření dotazů simulujících chybně zadané dotazy. Tato data mají otestovat odolnost algoritmů vůči různým druhům chyb. Prvním simulovaným druhem chyby bylo posunutí jedné až pěti náhodných not dotazu o náhodný počet tónů nahoru nebo dolů. Druhým simulovaným druhem chyby je transpozice celého dotazu o náhodný počet tónů nahoru nebo dolů. Třetím je kombinace obou předchozích způsobů. Pro každý druh chyby a délku obsahují testovací data 150 dotazů.

4.2 Metodika měření

Měření bylo zaměřeno na účinnost algoritmů, jejich rychlost, odolnost vůči chybám a efektivitu paralelizace. Pro měření bylo místo databáze MongoDB použito lokálního souborového systému s předzpracovanými skladbami uloženými jako binárně serializované objekty jazyka Python. V rámci práce byly otestovány všechny možné kombinace přístupů a částí vyhledávače popsaných v části 3. Pro každou kombinaci modulů byla měřena její úspěšnost a rychlost běhu.

Za správný výsledek vyhledávání je považována jakákoli část pocházející ze stejné skladby jako dotaz. Nemusí se jednat přímo o konkrétní segment. To je z důvodu, že se některé části mohou ve skladbách opakovat na několika různých místech (např. stejná melodie v každém refrénu nebo basová kytara hrající stejný riff jako kytara sólová).

Vyhledávač do výsledků z každé skladby vždy zahrne pouze jeden nejpodobnější segment. Protože dataset neobsahuje žádné duplicitní skladby, znamená to, že výsledky vyhledávání dotazovanou skladbu budou obsahovat vždy maximálně jednou. Výskyt pouze jednoho relevantního výsledku každého dotazu má za následek, že tradiční metriky pro evaluaci information retrieval systémů jako je precision, recall nebo f-skóre o úspěšnosti implementovaného vyhledávače neposkytují příliš dobré informace. Metriky vhodné k evaluaci tohoto typu vyhledávače jsou MRR a recall@k.

MRR je statistická metrika sloužící k evaluaci procesu, který vyprodukuje řadu odpovědí na dotaz seřazených podle pravděpodobnosti správnosti. V tomto případě jsou odpověďmi na uživatelský dotaz skladby. MRR je definováno následovně:

$$\mathrm{MRR} = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{\mathrm{rank}_i}$$

Kde Q je počtem dotazů a $\frac{1}{\mathrm{rank}_i}$ je reciproční hodnota pořadí, kde rank_i značí pozici prvního relevantního výsledku. Pokud se v řadě odpovědí relevantní výsledek nevyskytuje, je reciproční hodnota pořadí nahrazena 0.

Recall@k je metrika, která udává poměr relevantních výsledků nalezených v prvních k výsledcích. Matematicky jej lze definovat následovně:

$$\texttt{R@}k = \frac{p\hat{p}}{p\hat{p} + p\hat{n}}$$

Kde $p\hat{p}$ je počet relevantních výsledků v prvních k výsledcích navrácených vyhledávačem a $p\hat{n}$ je počet relevantních výsledků mimo prvních k výsledků. V případě, že vždy existuje pouze jeden relevantní výsledek, bude recall@k dosahovat pouze binárních hodnot 0 — relevantní výsledek se nevyskytuje v prvních k nebo 1 — relevantní výsledek se vyskytuje v prvních k. Tato metrika pro jeden izolovaný dotaz nenese pro evaluaci implementovaného vyhledávače příliš vysokou informační hodnotu, nicméně výpočtem průměrného

recall@k napříč několika dotazy lze získat informace relevantní pro vyhodnocení úspěšnosti.

Pro potřeby měření chování algoritmů a přístupů byla zvolena platforma Linode. Jedná se o cloudovou platformu, která umožňuje používání virtuálních strojů. Měření probíhalo na virtuálním stroji Dedicated 32 GB se 16 virtuálními procesory, pamětí 32 GB a operačním systémem Debian 11.

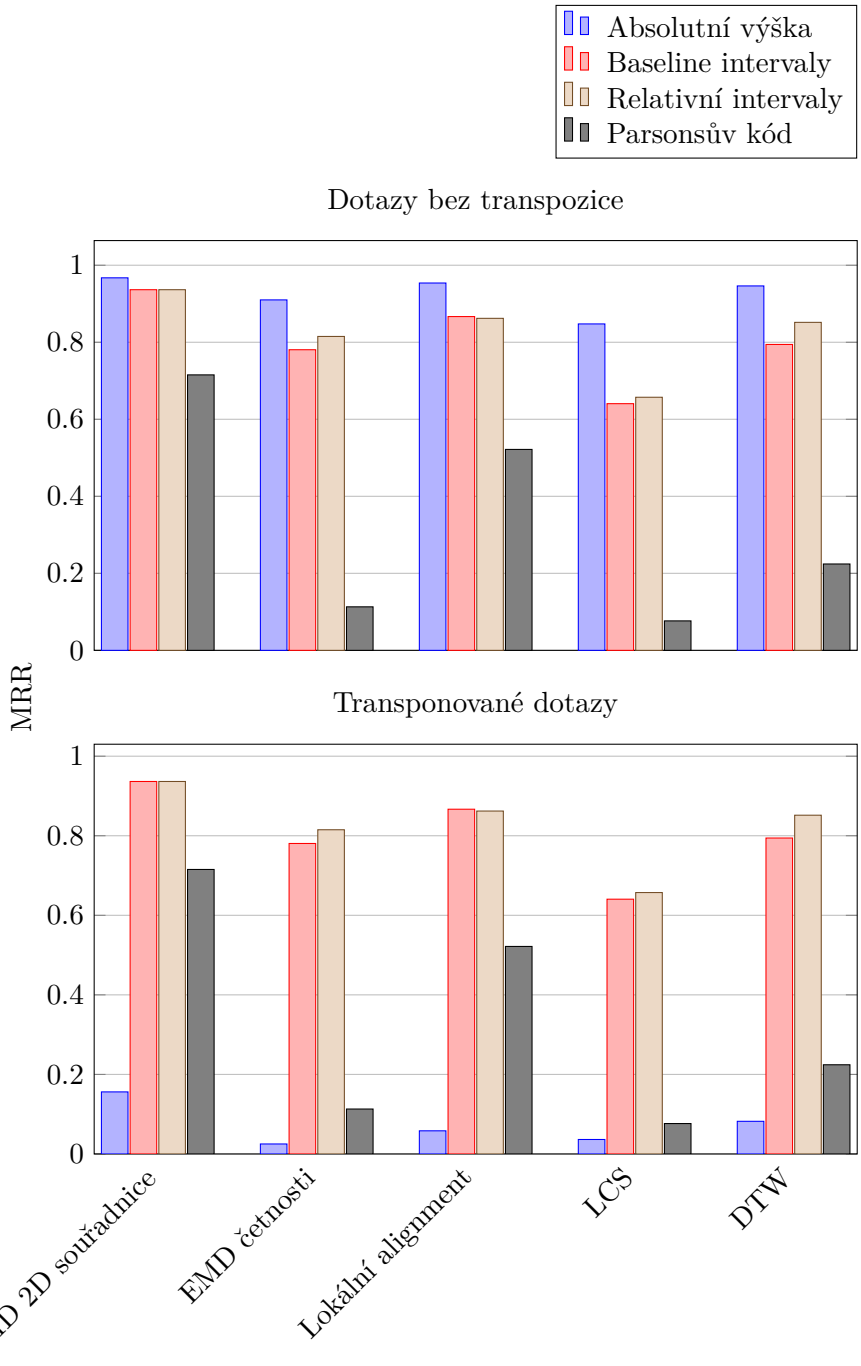
4.3 Výsledky

Tato část je věnována výsledkům naměřeným pomocí metod popsaných výše. Obsahuje interpretaci naměřených hodnot a odůvodnění volby algoritmů pro výslednou aplikaci. Zaměřuje se na vliv záměny jednotlivých modulů vyhledávače a na hledání nejefektivnější možné kombinace částí. Přesné hodnoty zde prezentovaných dat jsou vypsány v příloze B.

4.3.1 Standardizace

Standardizace melodie je velmi důležitým krokem vyhledávání. Její vliv není patrný, pokud je dotaz ve stejné tónině jako výsledek. Pokud je však dotaz oproti výsledku transponován, vyhledávač bez standardizace nedosahuje příliš dobrých výsledků.

Horní graf na obrázku 4.1 zobrazuje vypočítané MRR pro dotazy o délce 2 taktů, které nebyly transponovány (nachází se tedy ve stejné tónině jako relevantní výsledek). Přesné naměřené hodnoty se nachází v tabulce B.1. Sdružené sloupce ilustrují úspěšnost podobnostních metrik a vliv různých standardizačních metod. V případě dotazů v originální tónině dosahují všechny podobnostní metriky nejlepších výsledků při standardizaci pomocí absolutní výšky. Výsledky se výrazně změní, když vstoupí do hry transpozice. To ilustruje spodní graf na obrázku 4.1 zobrazující úspěšnost vyhledávače pro dotazy o délce 2 taktů, které byly transponovány o náhodný počet půltónů nahoru nebo dolů. Hodnoty grafu jsou také v tabulce B.2. Pro transponované dotazy dosahuje standardizace pomocí absolutní výšky not naprosto nejhorších výsledků. Na úspěšnost ostatních standardizačních metod transpozice vliv nemá, neboť eliminují potřebu shodného tonálního centra dotazu a vyhledávané skladby. Nejlepších výsledků pro transponované dotazy je dosaženo při použití standardizace pomocí relativních nebo baseline intervalů.

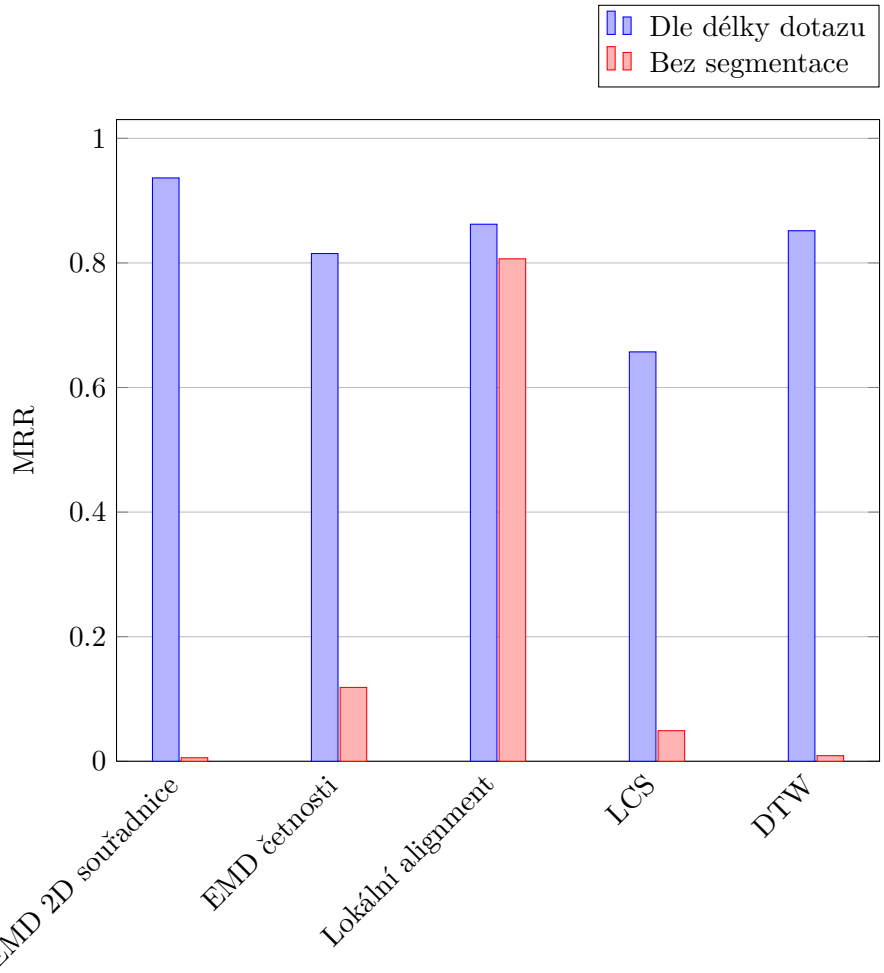


Obrázek 4.1: Vliv standardizace na efektivitu podobnostních metrik (dotazy o délce 2 takty)

4.3.2 Segmentace

Velký vliv na úspěšnost vyhledávače má segmentace skladeb. Její důležitost je ilustrována grafem 4.2 a tabulkou B.3, ve kterých je porovnávána úspěšnost vyhledávače bez segmentace a se segmentací na nepřekrývající se části stejné délky jako dotaz. Výsledky jsou pro dotazy délky 2 takty. Melodie pro toto měření byly standardizovány pomocí relativních intervalů.

U všech podobnostních metrik došlo při použití segmentace k výraznému zlepšení. Nejmenší vliv měla segmentace při použití lokálního sekvenčního alignmentu, nicméně i zde se výsledky zlepšily. Největší zlepšení při použití segmentace lze pozorovat u podobnostní metriky EMD s mapováním melodie na 2D souřadnice, kde hodnota MRR s použitím segmentace vzrostla z 0.0057 na 0.94.



Obrázek 4.2: Vliv segmentace na efektivitu podobnostních metrik (transponované dotazy o délce 2 takty)

4.3.3 Odolnost vůči chybám

Uživatelské dotazy nemusí vždy přesně odpovídat melodii cílové skladby. Je tedy nutné změřit, jak se algoritmy v takové situaci chovají. Pro potřeby měření byla chyba dotazu simulována posunutím jedné až pěti not o náhodný počet půltónů. V grafech je tato proměnná označena jako „počet změněných not“. Prezentované hodnoty byly naměřeny při použití segmentace a standardizace melodie pomocí relativních intervalů, neboť se tato kombinace ukázala jako nejefektivnější (viz. části 4.3.1 a 4.3.2).

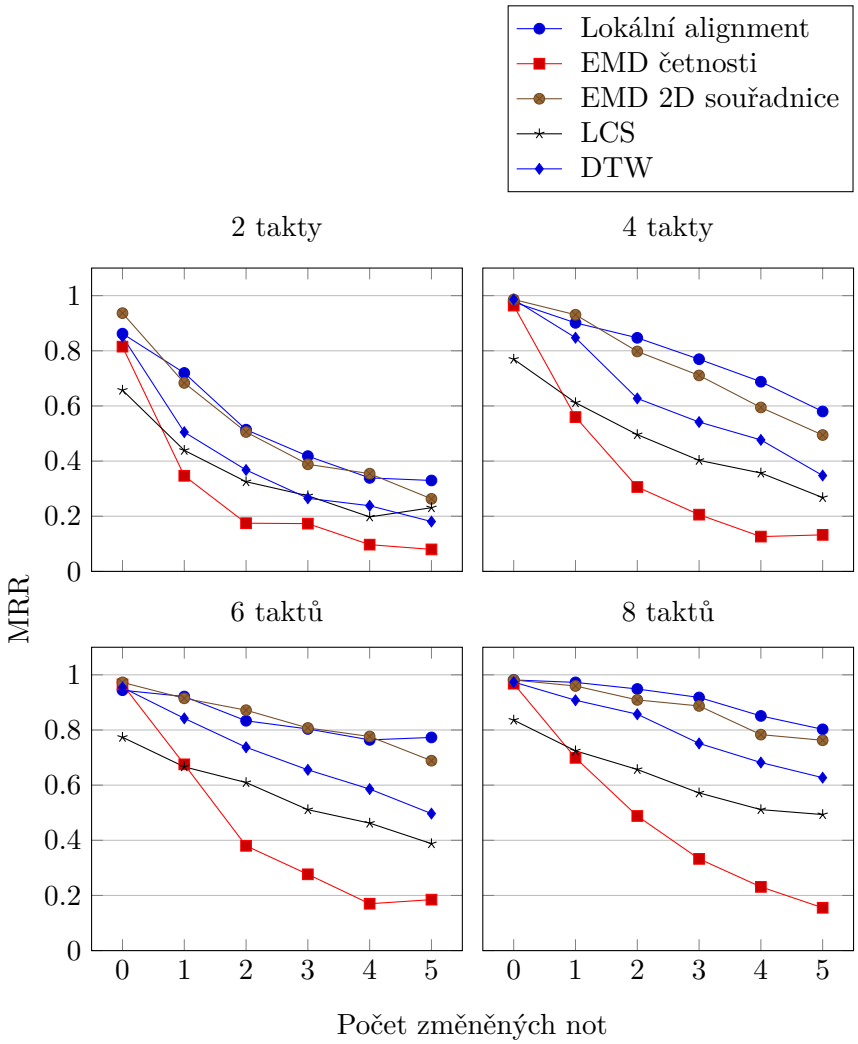
Odolnost algoritmů vůči chybám s délkou dotazu roste. Největší pokles úspěšnosti s růstem počtu chyb je možné pozorovat u všech podobnostních metrik při vyhodnocování dotazů délky 2 takty a nejmenší pokles u dotazů délky 8 taktů.

Za podobnostní metriky nejodolnější proti chybám lze označit EMD s mapováním melodie na 2D souřadnice a lokální sekvenční alignment. To je patrné z grafů 4.3 a 4.4. Přesné hodnoty se nachází v tabulkách B.4 až B.15. U lokálního sekvenčního alignmentu je MRR pro dotazy o délce 4 takty bez změny not rovno 0.98. Pro dotazy stejné délky se třemi změněnými notami klesne na 0.77 a pro 5 změněných not je MRR rovno 0.58. Odolnost lokálního sekvenčního alignmentu vůči chybám ukazuje také metrika recall@10, která je pro čtyřtaktové dotazy bez změny not rovna 1. To znamená, že ve sto procentech měřených případů se cílová skladba vyskytla v prvních 10 výsledcích. Recall@10 lokálního sekvenčního alignmentu při změně tří not čtyřtaktového dotazu klesne na 0.81 a při změně pěti na 0.65. I při takto výrazných změnách melodie se správné výsledky stále ve většině případů vyskytují na prvních pozicích.

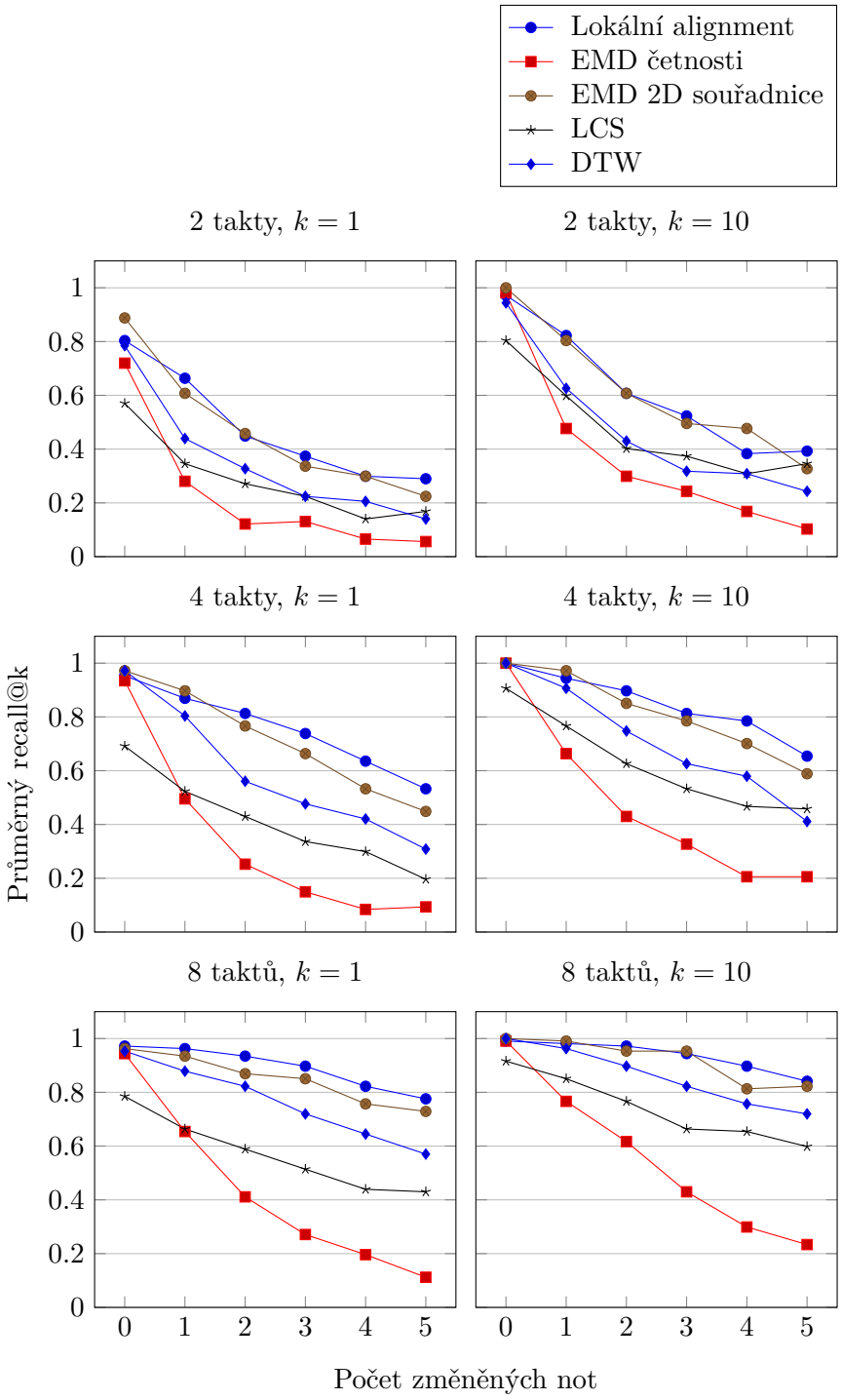
Úspěšnost EMD s mapováním melodie na 2D souřadnice je srovnatelná s úspěšností lokálního sekvenčního alignmentu. EMD dosahuje lehce lepších výsledků pro kratší dotazy a dotazy bez chyb, nicméně lze pozorovat výraznější zhoršení při zavedení umělé chyby (změně not dotazu). Pro dotazy délky 4 takty bez chyby je MRR rovno 0.99. U čtyřtaktových dotazů se třemi změněnými notami klesne MRR na 0.71 a pro stejně dlouhé dotazy s pěti chybami na 0.49. I když je zde odolnost proti chybám o něco menší, než u lokálního sekvenčního alignmentu, stále vyhledávač při použití této podobnostní metriky dosahuje dobrého recall@10. Podobně jako u lokálního sekvenčního alignmentu je recall@10 pro bezchybné čtyřtaktové dotazy roven 1. Se změnou tří not klesne na 0.79 a pro pět změněných not se rovná 0.59. I když není úspěšnost tak vysoká, jako u lokálního sekvenčního alignmentu, stále se i s výraznou změnou melodie relevantní výsledky ve velké části dotazů pohybují na horních příčkách.

Mnohem drastičtější pokles MRR a recall@k v závislosti na počtu změněných not lze pozorovat při použití EMD pro četnosti výšek not. Už při změně jedné noty čtyřtaktových dotazů klesne MRR z 0.96 na 0.56 a pro dotazy se třemi změněnými notami je rovno pouze 0.21. Ve srovnání s EMD

s mapováním melodie na 2D souřadnice a lokálním sekvenčním alignmentem neposkytují příliš dobré výsledky ani DTW a LCS. I když DTW pro čtyřtaktové dotazy bez chyby dosahuje MRR 0.99, se změnou tří not dotazu klesne MRR na 0.54 a pro pět not na 0.35. LCS dosahuje ze všech měřených podobnostních metrik nejhorších výsledků pro bezchybné dotazy jakékoli délky, je však odolnější proti chybám než EMD pro četnosti výšek not. MRR pro dotazy délky čtyři takty je pro LCS rovno 0.77. Se změnou tří not klesne na 0.40, tedy stále více než EMD pro četnosti výšek not, ale výrazně méně, než lokální sekvenční alignment nebo EMD s mapováním melodie na 2D souřadnice.



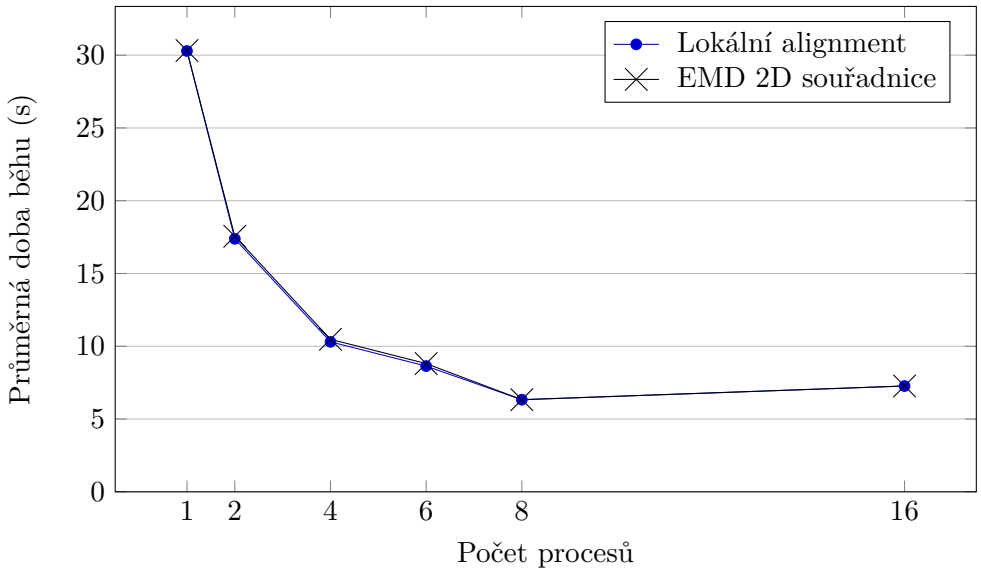
Obrázek 4.3: MRR pro různé délky dotazů a počty chyb



Obrázek 4.4: Recall@k pro různé délky dotazů a počty chyb

4.3.4 Paralelizace

Efektivitu paralelizace ukazuje graf 4.5. Zobrazuje průměrnou dobu běhu celého procesu vyhledávání od předzpracování dotazu až po jeho porovnávání s ostatními skladbami. Prezentované hodnoty jsou průměrnou dobou běhu napříč všemi dotazy délky 4 takty. Jejich přesná hodnota se nachází v tabulce B.16. Chování bylo měřeno při použití standardizace pomocí relativních intervalů a segmentace skladeb na základě délky dotazu.



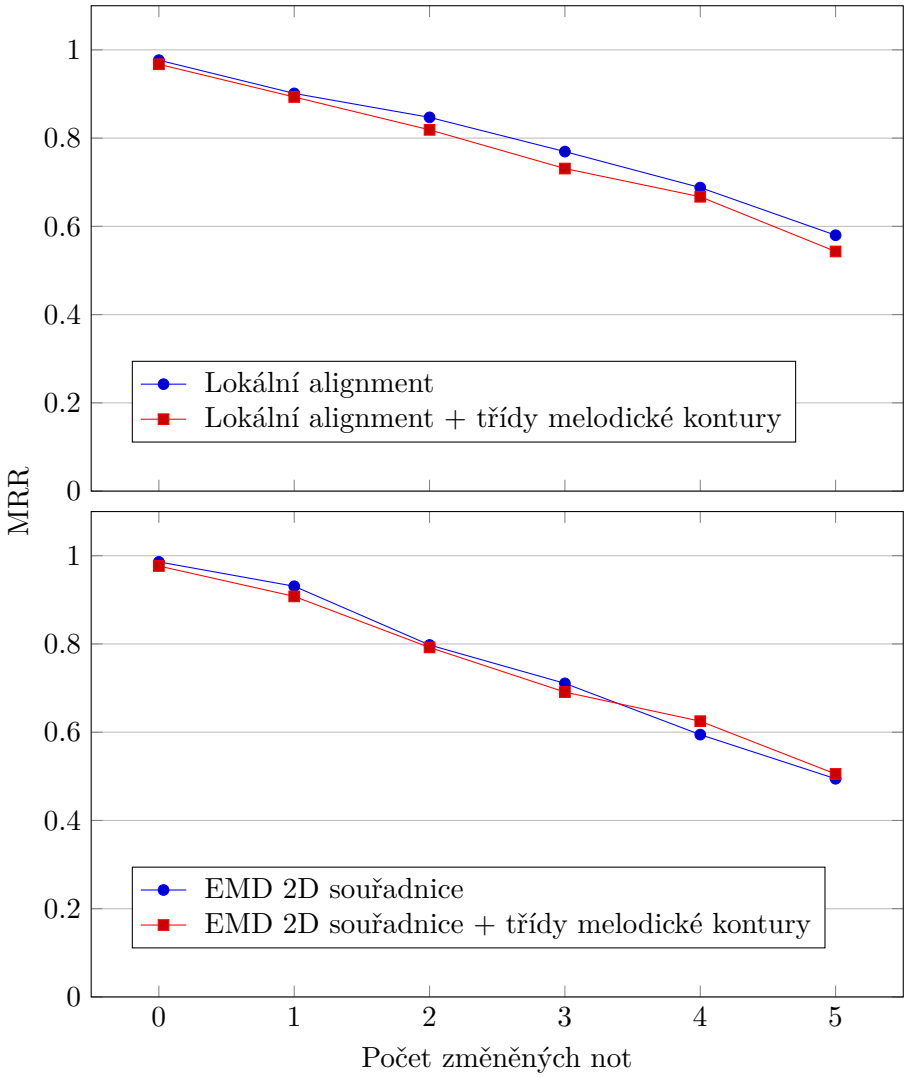
Obrázek 4.5: Efektivita paralelizace (dotazy o délce 4 takty)

Graf zobrazuje doby běhů při použití podobnostních metrik lokální sekvenční alignment a EMD s mapováním melodie na 2D souřadnice. Rozdíl doby běhů při použití obou metod porovnání je zanedbatelný. Největší zrychlení lze pozorovat při přechodu z jednoho procesu na dva a ze dvou procesů na čtyři. Pro lokální sekvenční alignment se při použití čtyř procesu místo jednoho sníží průměrná doba běhu z 30.3 s na 10.3 s. Vyšší počty procesů než čtyři nepřináší tak markantní zrychlení. To je následkem nutnosti většího množství meziprocesové komunikace. Při 16 procesech došlo ke zvýšení průměrné doby běhu. Použití většího množství procesů by bylo efektivní při větším množství skladeb.

4.3.5 Třídy melodické kontury

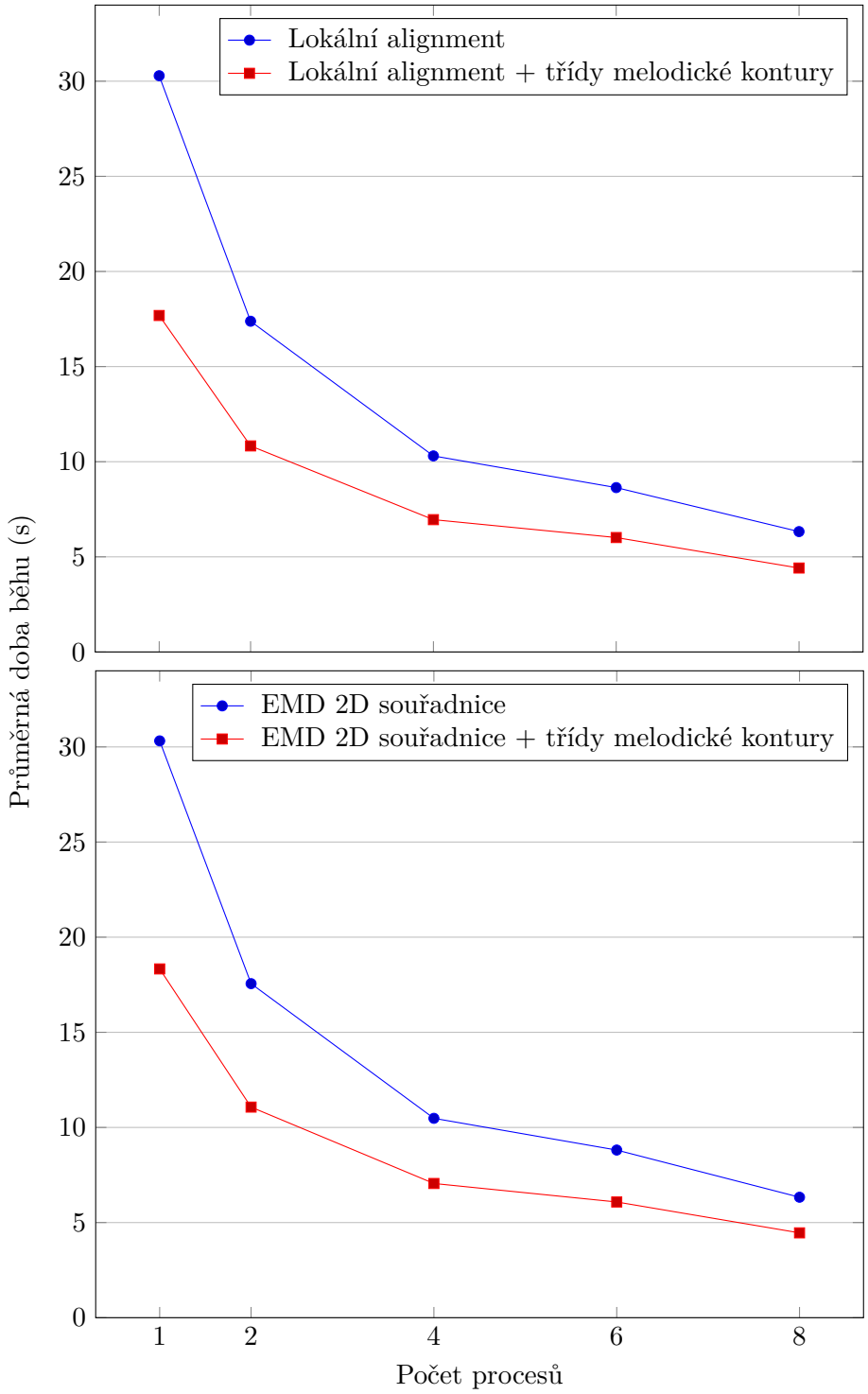
Extrakce tříd melodické kontury a jejich porovnání před použitím podobnostních metrik probíhá s cílem zrychlení vyhledávání. Při použití této metody dochází v některých případech k lehkému zhoršení výsledků vyhledávače. To je patrné z grafů na obrázku 4.6 a z tabulky B.17. V případě lokálního sek-

venčního alignmentu došlo k největšímu poklesu MRR pro dotazy o délce čtyři takty se třemi změněnými notami. MRR v tomto případě kleslo o pouhých 0.038. Pro EMD s mapováním melodie na 2D souřadnice byl největší pokles MRR 0.031 a to pro čtyřtaktové dotazy se čtyřmi změněnými notami. Pro melodie se čtyřmi a pěti změněnými notami naopak došlo u EMD ke zlepšení a MRR nepatrně vzrostlo.



Obrázek 4.6: Vliv porovnání pomocí tříd melodických kontur na úspěšnost algoritmů

Zrychlení vyhledávání ukazují grafy na obrázku 4.7 a tabulka B.16. Pro jeden proces dochází k průměrnému zrychlení o 41%. Pro dva, čtyři, šest a osm procesů je zrychlení při použití tříd melodických kontur více než 30%.



Obrázek 4.7: Zrychlení výpočtu — porovnání tříd melodických kontur (dotazy o délce 4 takty)

4.3.6 Volba algoritmů pro výslednou aplikaci

Pro použití v aplikaci byla na základě výsledků zvolena kombinace segmentace dle délky dotazu, standardizace pomocí relativních výsledků a porovnávání pomocí lokálního sekvenčního alignmentu. Tato varianta se ukázala jako nejúspěšnější a nejodolnější vůči chybám a transpozicím dotazu. Aplikace nabízí v uživatelském rozhraní možnost přepnutí mezi použitím nebo nepoužitím extrakce a porovnání tříd melodických kontur. Výchozím nastavením je použití tříd melodických kontur z důvodu výrazného zrychlení vyhledávání. Možnost vypnutí použití tříd melodických kontur je dostupná, protože jak bylo ukázáno výše, v některých případech dosahuje vyhledávač v takové konfiguraci lehce horších výsledků.

Kapitola 5

Diskuze

V rámci práce bylo implementováno, otestováno a evaluováno několik různých přístupů extrakce a porovnání melodií pomocí algoritmů pro porovnávání textových řetězců. Byla prezentována modulární architektura vyhledávače MIDI souborů umožňující jeho snadné rozšíření a záměnu jednotlivých částí. Nakonec byla vytvořena aplikace poskytující uživatelské rozhraní ve formě webové aplikace a využívající implementované metody k vyhledávání skladeb na základě uživatelských dotazů.

Některé otestované metody porovnávání skladeb se ukázaly úspěšnější než jiné. Zejména segmentace melodie na základě délky uživatelského dotazu a její následná standardizace pomocí relativních intervalů měly na výsledky vyhledávače velmi pozitivní vliv. V kombinaci s lokálním sekvenčním alignmentem jako podobnostní metrikou se ukázaly i jako velmi odolné vůči některým chybám v dotazu. Tato kombinace pro čtyřtaktové transponované dotazy bez chyby dosáhla MRR rovnému 0.98. MRR pro tuto konfiguraci vyhledávače pro dotazy o stejné délce se třemi chybami klesne na 0.77. Na dobré výsledky vyhledávače poukazuje i recall@10, který pro stejné dotazy se třemi chybami dosahuje hodnoty 0.81. Takto nakonfigurovaný vyhledávač je tedy schopen poskytnout užitečné výsledky, i když uživatel nezadá melodii přesně tak, jak se nalézá v databázi.

Velkou roli v úspěšnosti jednotlivých přístupů hrála délka dotazu. Vyhledávač ve výše jmenované konfiguraci byl s relativně vysokou přesností (MRR = 0.86) schopen vyhledat dotazy dlouhé pouze dva takty, nicméně byl zaznamenán významný pokles odolnosti vůči chybám.

V rámci dalšího vývoje vyhledávače by bylo možné vydat se mnoha směry. Pozitivní vliv na jeho výsledky by mohla mít kombinace více podobnostních metrik najednou nebo složitější segmentační metody. V této práci bylo experimentováno pouze se segmentací skladeb v databázi, nikoli se segmentací samotného dotazu. Přidání segmentačního kroku do předzpracování dotazu by mohlo přinést určité zvýšení úspěšnosti. Všechny představené standardizační metody kladly hlavní důraz na výšku tónů melodie. Přidání repre-

zentace rytmu do některé ze standardizačních metod by také mohlo příznivě ovlivnit úspěšnost vyhledávače.

V práci byl představen paralelizační model umožňující zrychlení za použití více než jednoho procesu a byly prezentovány výsledky prokazující jeho efektivitu. Zrychlení vyhledávání bylo kromě paralelizace dosaženo také metodou extrakce a porovnání tříd melodických kontur. Naměřené hodnoty ukazují, že vliv této metody na úspěšnost vyhledávače je zanedbatelný a průměrné zrychlení vyhledávání je 30–41%.

Dataset použitý v této práci obsahoval 987 skladeb. Jeho rozšířením by bylo možné dále ověřit chování algoritmů a efektivitu paralelního zpracování. V této práci byla paralelizace evaluována maximálně pro 16 procesů, neboť v takovém počtu přestalo docházet ke snižování doby běhu kvůli potřebě velkého množství meziprocesové komunikace.

V rámci práce byla také navržena architektura pro nasazení výsledné aplikace. Prezentovaná architektura je škálovatelná a pro potřeby práce dostačující, nicméně některé zvolené technologie se v průběhu práce ukázaly jako slabší volba oproti alternativám. Prvním případem je jazyk Python. Jelikož je interpretovaným jazykem, nedosahuje stejné rychlosti, jako některé kompilované jazyky. Protože některé použité algoritmy jsou relativně výpočetně náročné, aplikace zákonitě dosahuje vyšší doby odezvy. Dalším problémem je jeho dynamické typování. Kvůli rozsahu výsledné aplikace by bylo vhodnější použití staticky typovaného jazyka, který by napomohl předejít některým chybám. Ty byly sice z velké části eliminovány kontrolou typových anotací pomocí Mypy během kontinuální integrace, nicméně s použitím staticky typovaného jazyka by tento krok nebyl nutný.

Databáze Redis se ukázala jako velmi efektivní pro tento případ užití. Jelikož však jejím primárním účelem není sloužit jako fronta zpráv, postrádá část funkcionality, kterou nabízí alternativy. Pro potřeby této práce nebyl tento nedostatek patrný, ale při dalším rozšiřování aplikace by bylo vhodnější zvolit technologii, pro níž je distribuované zpracování zpráv primárním případem užití.

Komunikace mezi klientskou a serverovou aplikací probíhá pomocí opakovaného dotazování (pollingu). Toto řešení je pro potřeby aplikace dostačující a z hlediska použití funguje bez problémů, nicméně existují i jiné, možná vhodnější alternativy, které by při dalším vývoji stály za zvážení. Polling by mohl být nahrazen buď pomocí komunikačního protokolu WebSocket nebo pomocí Server-Sent events.

Jako vhodně zvolené technologie lze hodnotit React a MongoDB. React umožnil vytvořit komplexní uživatelské rozhraní s rozsáhlou funkcionalitou. MongoDB se ukázala jako vhodná volba databázové technologie. Jelikož ukládaná data nejsou relačního rázu a nemají přesně danou strukturu, poskytla MongoDB veškerou funkcionalitu, kterou aplikace vyžaduje.

Vytvořená aplikace umožňuje vyhledávání skladeb pomocí webového uživatelského rozhraní. Díky tomu je přístupná z jakéhokoli webového prohlížeče.

Rozhraní je responsivní a podporuje různé velikosti obrazovek od mobilních telefonů až po počítačové monitory. I přes drobné technologické nedostatky, které však nemají na koncového uživatele téměř žádný vliv, je aplikace použitelná a s rozšířením databáze o další skladby její užitečnost poroste.

Závěr

V rámci této práce vznikla aplikace umožňující uživateli vyhledat v hudební MIDI databázi skladbu na základě zadané melodie. Aplikace vznikala s cílem umožnit uživateli zbavit se „hudebního brouka“ v hlavě. Měla by pomoci s řešením situací, kdy člověku v hlavě hraje melodie, ale nemůže si vzpomenout, kde ji slyšel. Od této myšlenky se také odvíjel návrh všech jejích částí od uživatelského rozhraní až po volbu algoritmů. V současné chvíli databáze obsahuje 987 skladeb a lze ji snadno rozšířit.

V práci byly prezentovány metody pro porovnávání skladeb. Všechny metody byly evaluovány s ohledem na jejich užitečnost pro finální software. Byla ukázána modulární struktura hudebního vyhledávače rozdělující vyhledávání do několika kroků. Představen byl i paralelizační model umožňující hledání skladby zrychlit.

Na základě naměřených výsledků byly zvoleny algoritmy vhodné pro použití v aplikaci. Ta umožňuje uživateli zadat melodii pomocí myši nebo ji nahrát pomocí počítačové klávesnice či MIDI nástroje a vyhledat k ní nejpodobnější skladbu. Nakonec byla navržena a implementována architektura umožňující nasazení aplikace do produkčního prostředí s důrazem na škálovatelnost.

Literatura

[1] Ward, W.: Absolute pitch. In *The Psychology of Music (Second Edition)*, editace D. Deutsch, Cognition and Perception, Academic Press, second edition vydání, 1998, ISBN 978-0-12-213564-4, str. 265–298.

[2] Bachem, A.: Absolute Pitch. *The Journal of the Acoustical Society of America*, ročník 27, č. 6, 06 2005: s. 1180–1185, ISSN 0001-4966, doi:10.1121/1.1908155, `https://pubs.aip.org/asa/jasa/article-pdf/27/6/1180/11664526/1180_1_online.pdf`. Dostupné z: `https://doi.org/10.1121/1.1908155`

[3] Association, M. M.: MIDI 1.0 Detailed Specification. Technická Zpráva 4.2.1, 02 1996.

[4] Association, M. M.: Standard MIDI Files 1.0. Technická zpráva, 02 1996.

[5] Haitsma, J.; Kalker, T.: A highly robust audio fingerprinting system. In *ISMIR*, 2002, s. 107–115.

[6] Wang, A.; aj.: An industrial strength audio search algorithm. In *ISMIR*, Washington, DC, 2003, s. 7–13.

[7] Clausen, M.; Engelbrecht, R.; Meyer, D.; aj.: PROMS: A Web-based Tool for Searching in Polyphonic Music. In *ISMIR*, Plymouth, MA, 2000.

[8] Typke, R.; Giannopoulos, P.; Veltkamp, R. C.; aj.: Using transportation distances for measuring melodic similarity. In *ISMIR*, 2003, s. 107–114.

[9] Uitdenbogerd, A.; Zobel, J.: Melodic Matching Techniques for Large Music Databases. In *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, MULTIMEDIA '99, New York, NY, USA: Association for Computing Machinery, 1999, ISBN 1581131518, str. 57–66, doi:10.1145/319463.319470. Dostupné z: `https://doi.org/10.1145/319463.319470`

[10] Liu, C.-C.; Hsu, J.-L.; Chen, A.: An approximate string matching algorithm for content-based music data retrieval. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, ročník 1, 1999, s. 451–456 vol.1, doi:10.1109/MMCS.1999.779244. Dostupné z: https://doi.org/10.1109/MMCS.1999.779244

[11] Typke, R.; Wiering, F.; Veltkamp, R. C.: Evaluating the earth mover's distance for measuring symbolic melodic similarity. In *MIREX-ISMIR 2005: 6th International Conference on Music Information Retrieval*, Citeseer, 2005.

[12] Suyoto, I. S.; Uitdenbogerd, A. L.: Simple efficient n-gram indexing for effective melody retrieval. *Proceedings of the Annual Music Information Retrieval Evaluation exchange*, 2005.

[13] Deutsch, D.: Grouping Mechanisms in Music. In *The Psychology of Music (Second Edition)*, editace D. Deutsch, Cognition and Perception, Academic Press, second edition vydání, 1998, ISBN 978-0-12-213564-4, s. 299–348.

[14] Parsons, D.: *The directory of tunes and musical themes*. Cambridge, Eng.: S. Brown, 1975.

[15] Rabiner, L. R.; Juang, B.-H.: *Fundamentals of speech recognition*. Pearson Education, 2005.

[16] Smith, T.; Waterman, M.: Identification of common molecular subsequences. *Journal of Molecular Biology*, ročník 147, č. 1, 1981: s. 195–197, ISSN 0022-2836, doi:https://doi.org/10.1016/0022-2836(81)90087-5. Dostupné z: https://www.sciencedirect.com/science/article/pii/0022283681900875

[17] Jones, P.: Quart. 2017. Dostupné z: https://quart.palletsprojects.com/

[18] Ronacher, A.: Flask. 2010. Dostupné z: https://flask.palletsprojects.com/en/2.2.x/

[19] Jones, P.: Hypercorn. 2018. Dostupné z: https://pgjones.gitlab.io/hypercorn/

[20] Popa, B.: Dramatiq. 2017. Dostupné z: https://dramatiq.io/

[21] Sanfilippo, S.: Redis. 2009. Dostupné z: https://redis.io/

[22] MongoDB Inc.: MongoDB. 2009. Dostupné z: https://www.mongodb.com/

[23] Sysoev, I.: NGINX. 2004. Dostupné z: https://nginx.org/

[24] Hsiao, W.-Y.; Ju, Z.; Zhang, Y.; aj.: MidiToolkit. 2019. Dostupné z: https://github.com/YatingMusic/miditoolkit

[25] Weyde, T.; Datzko, C.: Efficient Melody Retrieval with Motif Contour Classes. In *ISMIR*, 2005, s. 686–689.

[26] Harvey, R.: Rob's Midi Library. 2001. Dostupné z: http://www.storth.com/music2006.htm

[27] Free Midi. Dostupné z: https://freemidi.org/

[28] Spotify AB: Spotify for Developers. Dostupné z: https://developer.spotify.com/

[29] Spotify AB: Web API. Dostupné z: https://developer.spotify.com/documentation/web-api

Příloha **A**

Seznam použitých zkratek

BSON Binary JSON

CI/CD Continuous Integration and Continuous Delivery

DTW Dynamic Time Warping

EMD Earth Mover's Distance

GCS Google Cloud Storage

IR Information Retrieval

MIDI Musical Instrument Digital Interface

MIR Music Information Retrieval

JSON JavaScript Object Notation

LCS Longest Common Subsequence

MRR Mean Reciprocal Rank

SMF Standard MIDI File

SPA Single Page Application

TPQN Ticks Per Quarter Note

Příloha B

Naměřené výsledky

	Absolutní v.	Baseline i.	Relativní i.	Parsonsův k.
EMD 2D souřadnice	0.967	0.936	0.936	0.715
EMD četnosti	0.909	0.781	0.815	0.113
Lokální alignment	0.954	0.867	0.862	0.522
LCS	0.848	0.641	0.657	0.077
DTW	0.946	0.794	0.852	0.224

Tabulka B.1: MRR pro různé metody standardizace (netransponované dotazy o délce 2 takty)

	Absolutní v.	Baseline i.	Relativní i.	Parsonsův k.
EMD 2D souřadnice	0.156	0.936	0.936	0.715
EMD četnosti	0.025	0.781	0.815	0.113
Lokální alignment	0.058	0.867	0.862	0.522
LCS	0.036	0.641	0.657	0.077
DTW	0.082	0.794	0.852	0.224

Tabulka B.2: MRR pro různé metody standardizace (transponované dotazy o délce 2 takty)

	Dle délky dotazu	Bez segmentace
EMD 2D souřadnice	0.936	0.006
EMD četnosti	0.815	0.119
Lokální alignment	0.862	0.807
LCS	0.657	0.049
DTW	0.852	0.009

Tabulka B.3: MRR pro různé metody segmentace (transponované dotazy o délce 2 takty)

Změněné n. / Podobnostní m.	0	1	2	3	4	5
EMD 2D souřadnice	0.936	0.683	0.505	0.388	0.355	0.263
EMD četnosti	0.815	0.347	0.175	0.173	0.097	0.08
Lokální alignment	0.862	0.72	0.514	0.418	0.339	0.33
LCS	0.657	0.439	0.325	0.274	0.198	0.231
DTW	0.852	0.505	0.368	0.265	0.238	0.181

Tabulka B.4: MRR pro různé počty změněných not (transponované dotazy o délce 2 takty)

Změněné n. / Podobnostní m.	0	1	2	3	4	5
EMD 2D souřadnice	0.986	0.931	0.798	0.711	0.594	0.494
EMD četnosti	0.964	0.56	0.306	0.206	0.126	0.132
Lokální alignment	0.977	0.901	0.847	0.769	0.688	0.58
LCS	0.77	0.612	0.496	0.403	0.357	0.268
DTW	0.986	0.847	0.627	0.542	0.477	0.348

Tabulka B.5: MRR pro různé počty změněných not (transponované dotazy o délce 4 takty)

Změněné n. / Podobnostní m.	0	1	2	3	4	5
EMD 2D souřadnice	0.973	0.915	0.872	0.807	0.776	0.689
EMD četnosti	0.965	0.675	0.38	0.277	0.17	0.185
Lokální alignment	0.944	0.921	0.833	0.804	0.764	0.773
LCS	0.774	0.666	0.609	0.511	0.462	0.388
DTW	0.954	0.842	0.737	0.656	0.586	0.497

Tabulka B.6: MRR pro různé počty změněných not (transponované dotazy o délce 6 taktů)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.981	0.96	0.909	0.887	0.783	0.763
EMD četnosti	0.968	0.699	0.488	0.333	0.231	0.155
Lokální alignment	0.981	0.973	0.949	0.918	0.851	0.802
LCS	0.836	0.724	0.657	0.572	0.511	0.493
DTW	0.974	0.908	0.857	0.751	0.682	0.627

Tabulka B.7: MRR pro různé počty změněných not (transponované dotazy o délce 8 taktů)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.888	0.607	0.458	0.336	0.299	0.224
EMD četnosti	0.72	0.28	0.121	0.131	0.065	0.056
Lokální alignment	0.804	0.664	0.449	0.374	0.299	0.29
LCS	0.57	0.346	0.271	0.224	0.14	0.168
DTW	0.785	0.439	0.327	0.224	0.206	0.14

Tabulka B.8: Průměrný recall@1 pro různé počty změněných not (transponované dotazy o délce 2 takty)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.972	0.897	0.766	0.664	0.533	0.449
EMD četnosti	0.935	0.495	0.252	0.15	0.084	0.093
Lokální alignment	0.953	0.869	0.813	0.738	0.636	0.533
LCS	0.692	0.523	0.43	0.336	0.299	0.196
DTW	0.972	0.804	0.561	0.477	0.421	0.308

Tabulka B.9: Průměrný recall@1 pro různé počty změněných not (transponované dotazy o délce 4 takty)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.963	0.897	0.841	0.785	0.757	0.654
EMD četnosti	0.953	0.626	0.327	0.224	0.121	0.168
Lokální alignment	0.916	0.888	0.794	0.757	0.72	0.748
LCS	0.701	0.579	0.533	0.421	0.402	0.327
DTW	0.935	0.822	0.71	0.626	0.551	0.439

Tabulka B.10: Průměrný recall@1 pro různé počty změněných not (transponované dotazy o délce 6 taktů)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.963	0.935	0.869	0.85	0.757	0.729
EMD četnosti	0.944	0.654	0.411	0.271	0.196	0.112
Lokální alignment	0.972	0.963	0.935	0.897	0.822	0.776
LCS	0.785	0.664	0.589	0.514	0.439	0.43
DTW	0.953	0.879	0.822	0.72	0.645	0.57

Tabulka B.11: Průměrný recall@1 pro různé počty změněných not (transponované dotazy o délce 8 taktů)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	1.0	0.804	0.607	0.495	0.477	0.327
EMD četnosti	0.981	0.477	0.299	0.243	0.168	0.103
Lokální alignment	0.972	0.822	0.607	0.523	0.383	0.393
LCS	0.804	0.598	0.402	0.374	0.308	0.346
DTW	0.944	0.626	0.43	0.318	0.308	0.243

Tabulka B.12: Průměrný recall@10 pro různé počty změněných not (transponované dotazy o délce 2 takty)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	1.0	0.972	0.85	0.785	0.701	0.589
EMD četnosti	1.0	0.664	0.43	0.327	0.206	0.206
Lokální alignment	1.0	0.944	0.897	0.813	0.785	0.654
LCS	0.907	0.766	0.626	0.533	0.467	0.458
DTW	1.0	0.907	0.748	0.626	0.579	0.411

Tabulka B.13: Průměrný recall@10 pro různé počty změněných not (transponované dotazy o délce 4 takty)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.981	0.953	0.907	0.832	0.804	0.776
EMD četnosti	0.981	0.776	0.477	0.364	0.262	0.234
Lokální alignment	0.972	0.953	0.897	0.888	0.841	0.822
LCS	0.869	0.804	0.757	0.664	0.589	0.505
DTW	0.981	0.897	0.804	0.72	0.654	0.589

Tabulka B.14: Průměrný recall@10 pro různé počty změněných not (transponované dotazy o délce 6 taktů)

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	1.0	0.991	0.953	0.953	0.813	0.822
EMD četnosti	0.991	0.766	0.617	0.43	0.299	0.234
Lokální alignment	0.991	0.981	0.972	0.944	0.897	0.841
LCS	0.916	0.85	0.766	0.664	0.654	0.598
DTW	1.0	0.963	0.897	0.822	0.757	0.72

Tabulka B.15: Průměrný recall@10 pro různé počty změněných not (transponované dotazy o délce 8 taktů)

Podobnostní m. \ Procesy	1	2	4	6	8	16
EMD 2D souřadnice	30.317	17.557	10.479	8.811	6.332	7.273
Lokální alignment	30.287	17.383	10.305	8.639	6.327	7.267
EMD 2D souřadnice + ETMK	18.331	11.069	7.054	6.082	4.459	5.76
Lokální alignment + ETMK	17.686	10.831	6.954	6.014	4.412	5.718

Tabulka B.16: Průměrná doba běhu ve vteřinách pro různé počty procesů (transponované dotazy o délce 4 takty), ETMK — S extrakcí a porovnáním tříd melodických kontur

Podobnostní m. \ Změněné n.	0	1	2	3	4	5
EMD 2D souřadnice	0.986	0.931	0.798	0.711	0.594	0.494
Lokální alignment	0.977	0.901	0.847	0.769	0.688	0.58
EMD 2D souřadnice + ETMK	0.977	0.908	0.792	0.691	0.625	0.505
Lokální alignment + ETMK	0.967	0.893	0.819	0.731	0.667	0.543

Tabulka B.17: MRR při použití porovnání tříd melodických kontur (transponované dotazy o délce 4 takty), ETMK — S extrakcí a porovnáním tříd melodických kontur

Příloha C

Spuštění aplikace

U všech příkazů popsaných v této sekci je předpokládáno, že jsou spouštěny z kořenového adresáře přílohy, pokud není řečeno jinak.

Pro propojení aplikace se Spotify API je před spuštěním nutné v souboru `.env` změnit proměnné `SPOTIFY_CLIENT_ID` a `SPOTIFY_CLIENT_SECRET`. Aplikace bude bez problémů fungovat i beze změny `.env` souboru, ale nebude poskytovat prokliknutí a náhledy skladeb ze Spotify. Tyto údaje je možné získat z portálu Spotify for Developers [28]. Celý proces je popsán v dokumentaci [29]. Aplikaci lze spustit tímto příkazem:

```
docker-compose up --build -d
```

Tím se sestaví a spustí všechny potřebné kontejnery. Takto spuštěná aplikace poběží na portu 3000 a bude obsahovat pouze pět skladeb. Více skladeb lze do běžící aplikace nahrát tímto příkazem:

```
docker exec -it midi-search-backend python3
   src/scrape.py download --parse --spotify
```

Tím se uvnitř jednoho z běžících kontejnerů spustí skript, který předzpracuje MIDI soubory a nahraje je do spuštěné databáze MongoDB. V případě, že byly skrze `.env` soubor poskytnuty validní údaje pro komunikaci se Spotify API, jsou skladby obohaceny o URL pro zvukový náhled a zobrazení na Spotify.

Analýzu chování algoritmů a vyhledávače lze spustit pomocí skriptu nazvaného `run_lab.py` ve složce `backend`. V případě, že je skript spouštěn lokálně (mimo kontejner), je nutné nejprve nainstalovat potřebné závislosti. Nainstalovat je lze příkazem:

```
pip install -r backend/requirements.txt
```

Po instalaci závislostí je možné skript spustit navigací do adresáře `backend` a spuštěním příkazu:

```
python3 src/run_lab.py
```

Příloha D

Obsah elektronické přílohy

- `.env` ..... proměnné prostředí
- `.github` ..... definice CI/CD
- `frontend` ..... zdrojový kód klientské aplikace
- `backend` ..... zdrojový kód serverové aplikace
- `dataset`
  - `example`
    - `raw` ..... pět MIDI skladeb načtených při každém spuštění aplikace
  - `full` ..... celý dataset včetně oanotovaných dotazů
- `docker-compose.yml` ..... konfigurace kontejnerů
- `docker-compose.override.yml` ..... konfigurace kontejnerů
- `docker-compose.ci.yml` ..... konfigurace kontejnerů pro CI/CD
- `docker-compose.prod.yml` ..... konfigurace pro produkční prostředí