DELTA – Střední škola informatiky a ekonomie, Základní škola a
Mateřská škola s.r.o.
Ke Kamenci 151, PARDUBICE

MATURITNÍ PROJEKT

Cloudová streamovací platforma hudby s administrativou.

Autor: Bořek Bartolšic
Třída: 4.A
Studijní obor: Informační technologie 18-20-M/01
Školní rok: 2021/2022

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	*Bořek Bartolšic*
Školní rok:	*2021/2022*
Třída:	*4.A*
Obor:	*Informační technologie 18-20-M/01*
Téma práce:	Cloudová streamovací platforma hudby s administrativou.
Vedoucí práce:	*Bc. Vlaďka Janů*

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem projektu je vytvořit funkční platformu pro streamování hudby, která bude přístupná ve webové aplikaci pro počítače a na zařízeních systému Android a iOS v podobě nativních aplikací. Pro administrativu se bude využívat oddělená jednoduchá webová aplikace. Uživatelem může být jakýkoliv posluchač hudební skupiny nebo komunity, který se buď přes aplikaci, nebo webové rozhraní přihlásí na jimi spravovaný server. Administrativa může spravovat obsah svého serveru (například cloud) a nechat jej streamovat pomocí CMS.

Uživatelská část umožní poslouchat hudbu a podcasty ze serveru a skládat své uživatelské playlisty. Mobilní aplikace si můžou soubory stáhnout a poslouchat je off-line. Administrativní část slouží k přidávání a spravování těchto audio souborů, ale také k sledování základních statistik o přehrávání.

Výstupem bude webová aplikace vytvořená za pomocí Next.js, která bude sloužit k připojení na seznam hudby a podcastů a následně její on-demand poslechu. Dále aplikace pro mobilní

zařízení, které navíc dovolí uživateli uložit na zařízení tyto soubory. A administrativní web ke správě připojených CMS serverů pro účely této aplikace.

Použité základní technologie:

- Pro webovou administrativu a poslech:
  - Next.js (TypeScript)
- Pro nativní aplikaci na iOS a Android:
  - Capacitor (Next.js 11)
- Pro ukázku napojení na systém:
  - Google cloud CMS API
  - MongoDB

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

Konec září: Návrh struktury, začátek prací na administrativní aplikaci a na poslechové aplikaci

Říjen–Listopad: Dokončení webových aplikací, tvorba nativních aplikací.

Prosinec: Napojení na CSM

15. leden: Doladění práce, chyby, testování

Leden–Březen: Dokumentace

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

V Pardubicích dne _____ _____

Bartolšic Bořek

Poděkování

Chtěl bych hlavně poděkovat Bc. Vlaďce Janů za vedení tohoto maturitního projektu, konzultace problémů a návrhy v rámci webových technologií.

Dále bych chtěl poděkovat Ing. Monika Borkovcová, Ph.D. za rady a inspirace při zpracování databázových technologií.

Mé díky také patři společenství Koinonii Jan Křtitel za inspiraci k tématu tohoto projektu a poskytnutí podkladů pro jeho zpracování.

Anotace

Tento projekt má za cíl tvorbu systému pro internetové streamování audio souborů po vzoru služeb jako Spotify, Apple Music nebo SoundCloud. Na rozdíl od těchto služeb, které jsou mířené na masové publikum, tento projekt cílí na menší zájmové skupiny (například jeden umělec a jeho sledující).

Tento systém je složen z 2 hlavních webových aplikací. První je mířena na poskytovatele této služby v rámci skupiny, kdy se jedná o administrátorskou aplikaci, pro rychlou správu seznamů skladeb a jejich položek. Druhá aplikace je pro samotného uživatele, pro poslech těchto zvukových stop. Další funkčnosti projektu jsou API pro administrativu a systém pro správu soborů.

Klíčová slova

Webová aplikace; Audio stream; Hudba; Podcast; HLS; Next.js; MongoDB; Ionic Capacitor

Annotation

This project aims to create a system for online streaming of audio files following the example of services such as Spotify, Apple Music or SoundCloud. Unlike these services, which are aimed at mass audiences, this project targets smaller interest groups (such as one artist and his followers).

This system consists of 2 main web applications. The target of first are the providers of this service within the group, which is an administrator application, for quick management of playlists and their items. The second application is for the user himself, to listen to these audio tracks. Other project functionalities are the API for administration and the file management system.

Keywords

Web application; Audio stream; Music; Podcast; HLS; Next.js; MongoDB; Ionic Capacitor

Obsah

1. Úvod

Během posledních let se streamovací společnosti jako Spotify, Apple Music nebo SoundCloud stali mezi uživateli populární. Posluchači umožňují poslech hudby na základě předplatného, nebo zdarma s patrným omezením [1] (reklamy a omezení funkcí). Tyto služby jsou cíleny na širokou veřejnost, jelikož poskytují univerzální přístup k širokému obsahu hudby, přehratelných seznamů skladeb a podcastů. Tento způsob poskytování zvukových stop online je možný i díky neustále zvyšující se rychlosti internetu [2], která dovoluje přenos množství dat potřebného k streamování hudby (bitrate[a] v 160 kb/s [3]) v decentní kvalitě.

Nevýhodou těchto centralizovaných služeb pro posluchače můžou být podmínky užívání těchto platforem. Tato nevýhoda ovlivní zvláště tvůrce audio obsahu, kteří často musí uzavřít smlouvu s hudebním vydavatelstvím, která mají různé poplatky a právo na část zisku, aby mohli svůj obsah publikovat. Samotná platforma si pak také vyžaduje určitou část zisku. [4] Jiný druh omezení nezávislých autorů může být limitace samotného obsahu (například omezená kvalita zvuku, počet přehrání, velikost úložiště), které lze odstranit za poplatek. [5]

Tyto limitace můžou začínající autory odradit od publikace jejich děl. Tento problém nabízí alternativu decentralizovaného řešení, kde platformu pro sdílení vlastní tvorby, bude spravovat samotný autor.

Proto projekt *Koinoniafy*[b] má za úkol vytvořit uživatelské prostředí pro posluchače této zájmové skupiny, dále administrátorské prostředí pro správu obsahu pro tvůrce samotné a systém pro zpracovávání, ukládání a sdílení dat. Po nasazení této služby na webový server se může stát užitečným nástrojem pro sdílení autorova obsahu s posluchači.

[a] počet bitů, které jsou zprostředkovány nebo zpracovány během časové jednotky
[b] zkrácený název: Koinoniafy, cloudová streamovací platforma hudby s administrativou

1.1 Terminologie

1.1.1 Centralizované řešení x Decentralizované řešení

Centralizované = Řešení, které spoléhá na ústřední bod (server), který rozděluje prostředky a data ostatním bodům v síti.
V kontextu projektu:
Hlavní server, který se stará o distribuci všech dat potřebných k fungování systému a jeho užívaní v rámci sítě.

Decentralizované = Řešení, které nemá konkrétní ústřední bod, kde samostatné uzly rozdělují vlastní prostředky a data pro omezenou síť.
V kontextu projektu:
Několik menších serverů distribuují jen data, která se týkají konkrétních uživatelů v síti. Každý server se stará o svou síť nezávisle na ostatních.

1.1.2 On demand x Realtime

On demand = *Na vyžádání.* Způsob šíření dat, kdy zařízení uživatele (klient) požádá o segment dat, který potřebuje k načtení části audio/video souboru, který bude v blízké době přehrán.

Realtime = *V reálném čase.* Šíření dat bez závislosti na zařízení uživatele. Přenos zprostředkovává stejná data více klientům v relativně stejný čas.

1.1.3 Frontend x Backend

Frontend = Klientská část například webové stránky. To, s čím může uživatel interagovat (tlačítka, formuláře, tabulky, …).

Frontend se stará o zobrazení viditelné části aplikace a o funkčnost základního rozhraní. Na server posílá dotazy s dalšími operacemi, které uživatel prování.

Backend = Kód webové aplikace, který je vykonáván na straně serveru. Uchovává a zpracovává data a stará se o funkčnost klientské části aplikace.

1.1.4 Client-side x Server-side x Pre-rendering

Client-side rendering (vykreslování na straně klienta) = Webová stránka nebo aplikace vykreslovaná prohlížečem. Server po přijetí požadavku na soubor webu pošle na stranu klienta celý soubor. Prohlížeč nezkompiluje web až do přijetí celého souboru.

Tento způsob vykreslování je přízniví pro SEO[c] bez náročnější úpravy kódu.

Při dobrém internetovém připojení je tato metoda funkční a spolehlivá. Nevýhodou je pomalé načítaní velkých webových aplikací při slabém připojení.

Server-side rendering (vykreslování na straně serveru) = Řešení, při kterém server zkompiluje požadované soubory webových stránek při každém volání. Klient pak obdrží plně generovanou stránku, kterou jen zobrazí. Tento starší způsob SSR je náročný na server při velkých webových aplikacích a každé generování může vyžadovat stovky milisekund, až sekundy pro odeslání klientovi.

Dnes díky Reactu a podobným knihovnám lze rozložit SSR mezi server a uživatele pomocí RenderToString metody, kdy se vytvoří izomorfní aplikace. Server předgeneruje stránku dehydratovanou s obsahem pro uživatele a pošle ji klientovi. Prohlížeč se pak postará o hydrataci a dodatečné vykreslování a stránku zobrazí.

Pre-rendering (předvykreslování) = Kompromis mezi CSR a SSR. Server předgeneruje aplikaci s běžným obsahem, přidá kostru rozložení stránky a odešle ji klientovy. Klient obdrží stránku a vykreslí na ní obsah podle dalších požadavků na data. [6]

1.1.5 Ostatní

Webcasting = Přenos mediálních dat přes internet do více zařízení.

Streamování (Streaming) = Kontinuální přenos dat v rámci datového proudu od zprostředkovatele (serveru) k uživateli pomocí internetu (webcasting) v reálném čase, nebo službou on demand.

Open-source = Veřejně přístupný a otevřený zdroj pro software nebo hardware. V případě softwaru se jedná o zdrojový kód. Každý může volně vidět, upravovat a distribuovat produkt, pokud dostojí podmínky licence originálního produktu.

[c] Search Engine Optimization – Oprimalizace pro vyhledávače; pro indexery zvyšuje relevanci stránky pro hledaný obsah

Framework = Softwarová struktura, skládající se z více balíčků softwaru, která je připravená pro rychlé nasazení a pomoc ve vývoji.

Software Package (Balíček) = Modul přidávající nové nebo upravené funkce do programu pro jeho rozšíření. Instalace a správa balíčků je možná za pomocí ***Package managerů*** jako je například npm, nebo yarn.

JSON (JavaScript Object Notation) = Formát zápisu dat v organizovaných objektech a polích. Výstupem je řetězec.

2. Různá řešení

Existují různé platformy pro streamování hudby on demand postavené na různých technologiích. Většina společností pro ochranu obchodního tajemství nezveřejňuje zdrojový kód řešení, ale lze dohledat nejvýznamnější technologie, které využívají. Níže jsou popsány řešení již funkčních produktů a vlastní řešení, které se snaží dosáhnout podobných cílů.

2.1 Spotify

Spotify je služba poskytující digitální hudbu, podcasty a videa s přístupem k široké škále obsahu. Mimo streamování aplikace zahrnuje systém doporučení na základě době poslechu hudby a analýzy jiejího obsahu. Uživatel svůj výběr může sdílet s ostatními pomocí odkazů na obsah, nebo playlisty. [7]

Uživatelé si mohou zakoupit předplatné, které jim zajistí rozšíření funkcí aplikace, neomezené přeskakování, volba konkrétních písní místo náhodného přehrávaní, poslech bez reklam a další výhody poskytované za měsíční poplatek. [8]

Autoři těchto děl mohou díky svému hudebnímu vydavatelství poskytovat svou hudbu na této platformě, nebo podepsáním smlouvy s hudebním distributorem, který se stará o licencování a vyplácení zisků autorům. [9]

Spotify je dostupné na různých zařízeních jako webová aplikace pro počítač dostupná pomocí podporovaného prohlížeče. Dále na mobilních zařízení Android a iOS a jako desktopová aplikace na Mac, Windows a neoficiálně i na Linux. [10]

2.1.1 Webová aplikace

Jako hlavní programovací jazyk pro webovou aplikaci Spotify je JavaScript, který je používán pro serverovou i klientskou část aplikace.

Svelte je open-source framework na kterém je postaven frontend aplikace. Stará se o kompilační procedury pro vystavění aplikace. Dynamické změny jsou přímo zapisovány do DOMu aplikace.

Serverová část aplikace funguje na Node.js serveru spolu s aplikačním proxy řešením *envoy* pro transparentnost síťové komunikace v aplikaci.

[11]

2.1.2 Desktopové a mobilní aplikace

Programovací jazyk pro tvorbu aplikace je C++, rozdělený do samostatných modulů funkcí. Spolu tyto moduly do aplikace spojuje proprietární vrstva *Cosmos*, která skládá aplikaci podobně jako webové stránky. Slouží i jako komunikační most mezi moduly.

Pro aplikační UI[d] byla použita technologie *Chrome Embedded Framework,* na kterém lze využít programovací jazyk JavaScript. Jako datový most mezi aplikačním jádrem a uživatelským rozhraním je opět vrstva Cosmos.

Zpracování logiky desktopové aplikace probíhá na klientské části. Serverový backend slouží jen jako datové úložiště a manažer datového provozu.

[12]

2.1.3 Databáze

Spotify používá *Cassandru* jako hlavní databázi pro většinu svých produktů. Spotify uvádí jako důvod, že dobře spravovaná *Cassandra* databáze je vhodná pro proces replikace, v případě síťové poruchy má vhodnou reakci a je tolerantní k obecným poruchám skupiny v databázích. [13]

Cassandra je open-source NoSQL databáze. Řadí se mezi sloupcové databáze, která uchovává data v datových sloupcích místo tradičních řádků. Výhodou je lepší efektivita při dotazování se na menší skupinu sloupců. Používá svůj vlastní jazyk *Cassandra Query Language* (CLQ) pro dotazování se na databázi. [14]

Podle CAP teorému je *Cassandra* kombinace AP (Availability a Partition Tolerance) zajišťující neustálou dostupnost i za cenu neaktuálnosti dat a toleranci na síťové chyby. [15]

2.2 Apple Music

Apple Music je platforma poskytující on-demand streamování hudby na základě předplatného. Díky této službě je možný poslech jakékoli písně z katalogu hudby poskytované v Apple obchodu iTunes. Poskytuje funkce jako off-line poslech stažené hudby, poslech rádia živě, integraci s virtuálním asistentem Siri, kopírování hudby ze zakoupeného CD a další funkce. [16]

[d] User Interface – uživatelské rozhraní

Produkt je primárně určen pro produkty z ekosystému Apple, ale byla vytvořena i aplikace pro Android zařízení a na počítače bez systému OSX. Systém Apple Music se skládá z webové, desktopové a mobilní aplikace. Je podporován i na nositelných zařízeních jako iWatch a na WebOS produktech jako chytré televize a palubní počítače automobilů. [17]

2.2.1 Webová aplikace

Hlavní programovací jazyk pro klientskou a serverovou část je JavaScript. Do nedávné doby byla používána i knihovna *jQuery* zjednodušující práci s JavaScriptem, HTML, stylováním a animací.

Serverová část webové aplikace funguje na Node.js a jako framework pro uživatelské rozhraní je používán React. Všechny své webové produkty Apple hostuje na svých vlastních serverech.

[18]

2.2.2 Desktopové a mobilní aplikace

Většina funkční logiky se zpracovává na Apple serverech, proto jsou aplikace převážně jen uživatelským rozhraním. Přesné technologie použity pro tyto aplikace nejsou podložené, ale nejspíše je aplikace napsaná v jazyku Swift a Objektovém C. [19]

2.2.3 Databáze

Předpokládá se, že Apple Music ukládá svá data na NoSQL open-source databázi *FoundationDB* vytvořenou společností Apple. *FoundationDB* je databáze s daty uloženými ve formátu klíč – hodnota. Tvrdí, že se nedá specifikovat CAP teorémem. [20]

2.3 SoundCloud

SoundCloud je online platforma pro on-demand streamování hudby. Uživatelům umožňuje poslech a sdílení hudby a podcastů. Velká část aplikace je zdarma s možností zakoupení předplatného, poskytující poslech bez reklam a off-line poslech. Uživatel i autor může nahrávat svůj vlastní obsah s limitací 3 hodin obsahu a 4 GB úložiště. [21]

SoundCloud je dostupný jako webová aplikace a jako Android a iOS mobilní aplikace.

2.3.1 Backend a frontend technologie

SoundCloud aplikace je napsána v JRuby (Ruby implementace v jazyku Java), objektově orientovaném jazyku Scala a dynamickém jazyku Clojure.

Jako webový server SoundCloud používá *nginx* a *thin* (Ruby web server) v kombinaci s reverzní proxy *HAProxy*. Pro doručování obsahu je používán systém *EdgeCast* a *Akamai*.

SoundCloud využívá indexového nástroje pro vyhledávání Elasticsearch s REST API.

Pro zvýšení efektivity serveru je využíván *Varnish*, který ponechává ve svém cache úložišti kopie vygenerovaných stránek, které posílá uživatelům místo opakovaného generování stejného obsahu.

Webová aplikace je postavena na Ruby frameworku *Ruby on Rails*. [22] [23] [24]

2.3.2 Databáze a úložiště dat

Jako hlavní databázi SoundCloud využívá sloupcovou NoSQL databázi *Cassandra*. Dále dokumentová NoSQL databázi *MongoDB*, *MySQL* seskupení, cacheovací systém *Memcached* a framework pro nestrukturovaná data *Apache Hadoop*.

Hlavním úložištěm obsahu jsou bucketové úložiště Amazon S3 a AWS Elastic Computing (EC2) pro výpočetní aplikace.

2.4 Plán projektového řešení

Řešení online streamování zvukového obsahu v tomto projektu bude spojení myšlenek již fungujících projektů v kombinaci s různými technologiemi pro dosažení řešení dostupnou pro uživatele a autory.

2.4.1 Cíle

- Webové aplikace pro poslech a správu a API pro správu.
- Řešení úložiště pro multimediální obsah.
- Databázové řešení pro správu dat a řešení pro jejich analýzu.
- Správa formátu souborů pro streamovatelné řešení.
- Využití streamovacího protokolu.
- Uživatelské rozhraní.

2.4.2 Prostředí

Serverové řešení pomocí serveru s operačním systémem Linux.

Používané služby jsou spravovány pomocí process manageru s možností dálkové správy a analytickými nástroji.

Firewall umožní výstup 2 portům obsahující webové aplikace. Ostatní služby jsou poskytovány lokálně.

Dálkový přístup k serveru je možný pomocí SSH přístupu.

2.4.3 Webové aplikace

Jsou využity 2 webové aplikace. Administrátorský přístup pro správu dat, obsahu a zobrazení statistických informací a uživatelský přístup pro poslech hudby. Administrátorský server navíc umožňuje i API přístup pro správu.

Pro propojení webové aplikace a zdrojů dat je použita datová vrstva.

Multimediální obsah je přijat ze stavu formulářů, v backendu aplikace převeden do formátu podporující HLS protokol pro streamování a uložen prozatímním řešením na souborový systém serveru.

2.4.4 Databáze a datová správa

Na serveru je spuštěna databáze jako hlavní úložiště informací o obsahu. Poskytuje základní informace o písních včetně ukazatele na streamovatelný soubor, playlistech, statistických datech a dalších informací.

Vyhledávací nástroj, do kterého jsou indexovány data pro vyhledávání z hlavní databáze.

2.4.5 API

Na administrátorské části je REST API řešení pro tvorbu a správu streamovatelný souborů z tradičních souborových formátů pro zvuk jako je mp3, nahrávání soborů do souborového systému a správa informací o obsahu v databázi.

3. Technologie

Níže je výpis důležitých technologií využitých v projektu. Pro lepší přehlednost jsou rozděleny do 4 sekcí.

V sekci obecných technologií se nachází ty, které jsou jádrem celého projektu a jsou nezbytné pro základní fungování.

V sekci Design jsou technologie využité specificky pro vzhled těchto aplikací.

V sekci API a backend se nachází technologie, které jsou využívány k zpracování dat na serveru a pro komunikaci v rámci HTTP API.

V poslední sekci Databáze jsou popsány databázové technologie, které jsou využity jako úložiště strukturovaných dat a technologie pro jejich propojení.

3.1 Obecné technologie

V této sekci se nachází technologie sloužící pro základní fungování aplikace. Jsou zde i zmíněny programovací jazyky, které byly využity pro napsání projektu, úpravu open-source balíčků a testování operací s databázemi.

3.1.1 Jazyky

3.1.1.1 HTML

Hyper Text Markup Language je značkovací jazyk pro tvorbu kostry elementů webové stránky. Později je prohlížečem zobrazován vizuál takto vytvořené kostry v základním vzhledu. [25]

Pro konkrétní vzhled elementu je možné ho definovat v jazyku CSS. Každému elementu lze přidělit konkrétní identifikaci, nebo jej zařadit do skupiny elementů se stejnou třídou. Díky této identifikaci lze konkretizovat vzhled v jiném souboru se styly, nebo přidělit interakce v souboru obsahující JavaScript. [26]

3.1.1.2 CSS

Cascading Style Sheets je jazyk popisující vizuální formu HTML a XML[e] kostry. Určuje, jakým způsobem bude prohlížeč vykreslovat stylovanou HTML strukturu. Přináší i primitivní interakci na popisovaných elementech. Popisuje i animace a responzivitu webových stránek na různých médiích. [27]

3.1.1.3 JavaScript

JavaScript je objektově orientovaný skriptovací jazyk kompilovaný a spouštěný přímo v prohlížeči, nebo v prostředí Node.js[f] na serveru. V prohlížeči je používán pro tvorbu fungování dynamických webů. Lze ním definovat uživatelskou interakci s webovou stránkou. JavaScript může přistupovat ke kostře (DOM[g]) stránky, k ní přistupovat a upravovat ji. [28]

V prostředí serveru se používá pro zpracování a distribuci dat za pomocí Node.js, *Web API* [29] a zdroji serveru. Lze ním tvořit backend webové aplikace. Za pomocí další balíčků dokáže komunikovat s jinými zařízeními a databázemi, nebo manipulovat se soubory.

Konzole vývojářského prostředí v prohlížečích umožňuje používání *JavaScriptu* ve spojení s konkrétním webovým dokumentem.

3.1.1.4 TypeScript

TypeScript je open-source programovací jazyk, který nabízí všechny funkce JavaScriptu a přidává k nim systém statického typování objektů. Přísné typování může pomoci vývojářům zachytit logické chyby v parametrech dříve, než jsou zkompilovány a přispívá k čitelnosti a porozumění kódu. Do JavaScriptu přidává *interface* po vzoru jiných objektově orientovaných jazycích a další atributy. [30]

Pro vykonání kódu je kompilován prostředím, na kterém je spuštěn, do JavaScriptu a poté spuštěn jako JavaScriptový kód. V IDE[h] funguje i jako základní kontrola syntaxe a typové logiky.

[e] *Extensible Markup Language* – značkovací jazyk pro ukládání a serializaci dat čitelných pro člověka i stroj
[f] systém umožňující používání JavaScriptu na serveru přispívající ke konzistentnosti jazyku serveru a klienta
[g] *Document Object Model* – objektové zobrazení XML a HTML dokumentů; umožňuje API přístup k objektům webu
[h] *Integrated Development Enviroment* – vývojové prostředí pro usnadnění práce programátorům

3.1.1.5 MongoDB Shell

MongoDB Shell je přístup k MongoDB databázi z příkazového řádu. Využívá dotazovací jazyk na bázi JavaScriptu. Lze jej použít i pro administrativní operace v databázi. [31]

V projektu byl využíván pro testovací dotazy na databázi. V samotném projektu je používán balíček *mongodb* pro přímé použití s JavaScriptem.

3.1.1.6 Query DLS

Query Domain Specific Language založený na JSON zápisu pro definování dotazů na ElasticSearch databáze pomocí HTTP API volání. *Query DLS* je rozdělen na dva funkční typy. Spojením více dotazů vzniká silný indexovací a vyhledávací nástroj. [32]

Leaf query (Listový dotaz) je zaměřen na hledání hodnoty v rámci jednoho dotazu na jeden atribut v databázi.

Compound query (Sloučený dotaz) spojuje více listových a sloučených dotazů a na základě logických podmínek je vytvořen jeden kombinovaný dotaz. S jeho pomocí je možné měnit parametry jiných dotazů.

3.1.1.7 Python

Python je objektově orientovaný programovací jazyk pro tvorbu webů, softwaru, automatizací a pro analýzu a práci s daty. Není specifikován pro konkrétní užití a je užíván pro obecné účely. Je použitelný jako rozšíření do jazyků C a C++. [33]

Přenosnost aplikací psaných v *Pythonu* je možná na různých zařízeních podporujících standard Unix a Unix-like. [34]

Je rozšiřitelný pomocí vlastního indexu balíčků (*PyPI*) pro python. [35]

3.1.2 Npm

Npm je online repositář open-source balíčků pro Node.js (JavaScript) a CLI[i] nástroj pro instalaci a správu těchto rozšíření. Při instalaci balíčků se npm stará o umístění, díky kterým Node může balíčky najít a používat. [36]

npm příkaz lze konfigurovat pro široký okruh využití jako je publikace, objevování, instalace a vývoj různých balíčků. [37]

[i] *Command Line Interface* (příkazový řádek) – uživatelské prostředí pro komunikaci se zařízením

3.1.3 Node.js

Node.js je open-source serverové prostředí pro různé Unix a Unix-like operační systémy. Umožňuje použití JavaScriptu na serveru pro backendové řešení aplikace. Díky Node.js lze vytvářet dynamické webové stránky, číst a upravovat soubory v souborovém systému serveru a komunikovat s lokální, nebo vzdálenou databází. Node.js podporuje asynchronní operace. [38]

3.1.4 React

React je JavaScriptová knihovna pro tvorbu uživatelského prostředí pomocí komponentů. Komponenty jsou malé izolované kousky kódu, které jsou vykreslovány jako dílčí část uživatelského rozhraní. Tyto komponenty jsou pak navazovány na stránku, z kterých jsou složeny. *React* podporuje vykreslování na požádání, díky tomu vzniká dynamická *single-page* webová aplikace[j]. [39]

React Native přidává podporu uživatelského rozhraní v JavaScriptu pro mobilní aplikace. Lze jej spustit na mobilní verzi Node, nebo přímo ve fragmentu Android aplikace. [40]

3.1.5 Next.js

Next.js je open-source vývojový framework pro Node.js servery rozšiřující knihovnu uživatelského rozhraní React. Přidává podporu kombinace vykreslování pomocí *Server-side rendering* (SSR) a *Pre-rendering* pomocí hydratace a dehydratace kódu. [41]

Přidává dynamické směrování (dynamic routing), které umožňuje plynulý přechod mezi stránkami při zachování single-page funkčnosti i pro unikátní stránky generované přímo s obsahem pro klienta na základě stejného rozložení.

Poskytuje vlastní řešení získávání dat ze serveru, podpora univerzálního rozložení stránek a optimalizace velikosti obrázků pro různé použití.

Přidává i funkce pro vývojáře jako je rychlé kompilování změn prováděných v kódu, analýzu kódu s integrovaným balíčkem *ESLint* s konfigurací pro Next.js a podporu TypeScriptu.

[j] *Single-page application* (SPA) – dynamická webová stránka, která se dynamicky přepisuje místo načítaní nové stránky.

3.1.6 Apollo GraphQL

Apollo server je open-source JavaScriptový server spravující GraphQL operace. Je to služba poskytující datovou vrstvu v rámci GraphQL schématu, která podporuje správu dat mezi serverem a klientem. Spojuje více zdrojů dat jako jsou databáze, API volání a mikroslužby[k] a sjednocuje je do jedné vrstvy ke které může přistupovat webový server, klient a klientské mobilní aplikace.

Dotazy jsou vykonávány na základně požadavků s JSON zápisem pomocí HTTP volání.

3.1.7 Babel a SWC

Babel a *SWC* jsou kompilátory, které převádějí novější verze standardů JavaScriptu a TypeScriptu pro zpětnou kompatibilitu s prohlížeči. V rámci projektu je kompilátor používán hlavně pro překlad TypeScript a React kódu.

Babel je kompilátor, který převádí kód ze standardu *ECMAScript 2015+* do starší prohlížeči podporované verze *ES5*. Přináší i možnost debugeru[l] pro vývoj.

SWC je kompilátor na základě programovacího jazyku Rust, je používán frameworkem Next.js jako hlavní kompilátor. Tvrdí, že je 20x rychlejší než Babel na 1 vlákně a 70x rychlejší na 4 jádrech. Přináší vlastní *webpack* a je optimalizován pro analytického nástroje *Jest*.

Kvůli nekompatibilitě SWC s Apollo serverem je v projektu používán jako hlavní kompilátor Babel.

3.1.8 HLS

HTTP Live Streaming je protokol pro streamování multimediálního obsahu přes internet. Příjemce může upravit svůj bitrate pro stávající podmínky internetového připojení. Poskytuje nepřerušený tok dat v co nejvyšší kvalitě, prioritou je však kontinuálnost. Další vlastností *HLS* je zabezpečení pomocí šifrovacího klíče. Dokáže paralelně poskytovat více variant stejného obsahu, například pro živý překlad do více jazyků.

Tento způsob streamování je možný na základě segmentace multimediálního obsahu do menších celků, které jsou na základě Media Playlistu[m] streamovány klientovi. [42]

[k] *Microservice* – software architektura aplikací, které jsou definovány jako volně provázané služby
[l] Vývojářský asistenční program pro hledání a opravu chyb

Apple produkty podporují ze základu *HLS*, díky čemuž lze přijímat stream bez nutnosti dalšího dekódování. U jiných produktů je dosažena určitá úroveň podpory, ale může vznikat chybovost v přenosech. Proto je v projektu použit balíček *HLS.js*, který dekóduje stream dat z *HLS* protokolu do bitstreamu[n], který dokážou prohlížeče přehrát jako multimediální soubor.

Segmenty dat HLS jsou přenášeny přes TCP[o] rámec na požádání přehrávače on demand. [43]

3.1.9 Formik

Formik je open-source knihovna pro React a React Native přidávající pokročilejší manipulaci s webovým formulářem. Lze určit způsob, jakým jsou zpracovány stavy interakce s formulářem, validaci dat ve formuláři, způsob zobrazování chybových zpráv a zpracování po odeslání formuláře. [44]

Přímý vývoj na React zamezil časté překreslování formuláře po změně jeho stavu, například zadáváním nových hodnot do formuláře. Dále snižuje latenci při zpracování hodnot oproti *Redux* formulářům. [45]

3.1.10 Přehrávače

3.1.10.1 ReactPlayer

ReactPlayer je React komponenta pro přehrávání různých formátů multimediálního obsahu. Mimo standardní formáty přímá a dekóduje streamy HLS, DASH a FVL a dokáže přehrát video a audio ze stránek poskytujících obsah jako YouTube, Twitch, SoundCloud a jiné. [46]

Výhodou je obsáhlá konfigurace a možnost připojení dalších modulů pro přijímání méně konvenčních zdrojů dat. *ReactPlayer* má aktivní podporu vývojářů a díky své popularitě i komunity.

ReactPlayer nemá v základu podporu pro přehrávání playlistu multimediálního obsahu, proto je využíván jen v administrátorské části projektu pro poslechový náhled při správě.

---

[m] *Media Playlist* – formát zápisu přístupu k lokacím a časového rozsahu segmentů dat v listové podobě
[n] *Bitstream* – binární sekvence dat
[o] *Transmission Control Protocol* – protokol transportní vrstvi OSI modelu zaměřený na spolehlivost přenosu dat

3.1.10.2 Video.js

Video.js je rozšiřovatelná knihovna pro HTML5 video element. Umožňuje API rozhraní pro komunikaci s elementem pro vlastní uživatelské prostředí, přidání podpory pro formáty dat jako je HLS stream, sjednocení přehrávačů v rámci webů a možnost přidávání modulů pro rozšířenou funkčnost. [47]

Není poskytována komponenta pro React, proto je práce s Video.js náročnější pro tento projekt. Existují balíčky, které přidávají React funkční komponentu pro *Video.js* za cenu náročnější nebo omezené konfigurace a nižší podporu. Z toho důvodu byl používán jen na začátku tohoto projektu.

3.1.10.3 react-jinke-music-player

Tento balíček je menší řešení přehrávače pro React včetně uživatelského rozhraní. Obsahuje komponentu s vlastní možností přehrávání playlistů, které nepotřebují uživatelskou interakci pro spuštění každé položky v seznamu na Apple produktech (Apple politika [48]). [49]

V uživatelské části projektu je použit tento přehrávač pro vyhovující funkce a dodržení Apple politiky pro kontinuální přehrávání playlistu.

3.2 Design

V této sekci jsou obsaženy technologie pro vzhled uživatelského prostředí. Jsou to knihovny a balíčky softwaru převážně pro frontend webové aplikace.

3.2.1 MUI

MUI je knihovna pro React obsahující komponenty s designem ve stylu *Material Design* s širokou možností užití. Je zahrnuto velké množství prvků pro uživatelskou interakci, vizualizace dat a rozložení stránek. Komponenty lze funkčně vázat a spravovat jejich stavy. Nabízí i více variant u většiny komponentů pro více případů použití. [50]

Je poskytována upravitelná paleta barev a stylů pro různé komponenty pro vizuální sjednocení a jednodušší práci.

Každá komponenta má i vlastní *Component API* pro přístup k prvku přímo z instanci komponenty a nestylovanou verzi pro vlastní design se zachováním funkčnosti.

3.2.2 Nivo

Nivo je React knihovna přinášející komponenty pro vizualizaci dat. Je postavena na základě knihovny pro vizualizaci *D3* s přidanou podporou pro SSR a vysokou přizpůsobitelností.

Je přidána optimalizace D3 prvků pro izomorfní aplikace, jakou je tento projekt. Podporuje i hardwarově zrychlené animace a přechody pomocí *react-motion* balíčku. Umožňuje možnost volby mezi HTML, SVG nebo Canvas zobrazením vizualizací. [51]

3.2.3 Styled components

Styled components je balíček přidávající CSS styly přímo na React komponenty pro lepší přehlednost a variabilnost stylování ve větších projektech. [52]

3.2.4 SortableJS

SortableJS je JavaScriptová knihovna pro správu a užití drag-and-drop listů. *SortableJS* podporuje i React knihovnu s komponentou konfigurovatelnou v kódu. Díky Komponent API lze upravovat a vyvolávat stavy prvku. [53]

V projektu je *SortableJS* využíván pro zobrazení a úpravu fronty audio obsahu k přehrání.

3.3 API a backend

V této sekci jsou popsány technologie použité pro zpracování dat na serveru a knihovny spravující API volání na server.

3.3.1 FFmpeg

FFmpeg je open-source framework složený ze sady knihoven pro operace s multimediálním obsahem. Umožňuje zakódovat a dekódovat obsah do různých kodeků, streamovat, filtrovat, nahrávat a spouštět audio a video a další funkce. Je schopen i konverze mezi formáty. [54]

V projektu je využívána JavaScript knihovna pro konverzi audio souborů do formátu použitelný pro HLS. Vytváří M3U8 soubor s URI na segmenty HLS streamu.

3.3.2 Axios

Axios je HTTP klient pro node.js a prohlížeče. Vytváří HTTP požadavky na straně prohlížeče a umožňuje posílat požadavky i ze serveru na jiné servery. Spolu s *Axios* lze využít *Promise* Web API. [55]

3.3.3 Next-connect

Next-connect je odlehčený middleware a router pro Next.js framework. Tento balíček je postaven na middlewaru *trouter*, který doplňuje / nahrazuje směrovací část *Express.js*. V projekt je využit na příjem a přesměrování API volání na server. [56]

3.3.4 Multer

Multer je middleware pro upload souborů na server na základě žádosti klienta. Je zodpovědný za správu *multipart / form-data*, které jsou právě bitovým tokem nahrávaného souboru / skupiny souborů z odeslaného formuláře. Je postaven na základě modulu *busboy* starající se o čtení příchozích dat z formuláře.

3.4 Databáze

V této sekci jsou popisovány databázové technologie, které jsou projektem užívány pro ukládání relativně strukturovaných dat a technologie pro indexování databáze v reálném čase.

3.4.1 MongoDB

MongoDB je NoSQL[p] dokumentová databáze fungující na Unix a Unix-like systémech jako úložiště dat flexibilně strukturovaných do JSON-like dokumentů s dynamickým schématem. Poskytuje rychlé a jednoduché ukládání dat, jejichž struktura se v průběhu může měnit.

Databáze dynamicky mapuje data do objektového modelu pro snazší přístup v kódu pro vývojáře. Zpracovává dotazy v momentě, kdy jsou žádány, indexuje data pro snazší hledání a načítání a agreguje dotazy s následným výpočtem. [57]

V CAP teorému představuje CP variantu, kde databáze poskytuje vždy konzistentní nejaktuálnější data a odolnost k přerušení zajišťující fungování i v případě zdržení, nebo ztráty části zprávy v síti. [15]

[p] databázový koncept nevyužívající tradiční tabulkové schéma s relační strukturou.

V projektu je tato databáze použita pro její jednoduché a rychlé použití, čitelnost dat a snadné napojení na projekt. V této databázi jsou uloženy data o písních, playlistech, vrcholech hlasitosti a základní statistické data.

3.4.2 Elasticsearch

Elasticsearch je otevřený nástroj pro analýzu a hledání v datech. Data mohou být ve formě textu, čísel, souřadnic, objektů i nestrukturovaných dat. *Elasticsearch* přijímá data z různých zdrojů, které jsou dále čteny, normalizovány a poté indexovány.

Po procesu indexování se může uživatel dotazovat pomocí *Query DLS* jednoduchými i komplexními JSON-like strukturami. Uživatel tak může provádět fulltextové vyhledávání[q] s podporou hledání přesné shody (exact match), částečné shody (partial) i *fuzzy* hledání, které využívá *Levenshteinovy vzdálenosti* [58] ke shodě (přehazování, doplňování a mazání znaků). Jako odpověď na REST API volání vrací JSON objekt s požadovaným výsledkem. [59]

Společně s nástroji *Kibana* a *Logstash* tvoří *Elasticsearch* silný analytický nástroj pro jednotlivé aplikace i komplexní business modely.

V tomto projektu je Elasticsearch používán pro indexaci databáze MongoDB, a nástroj pro speciální vyhledávání písní nebo playlistů s podporou *fuzzy* s prioritizací přesné shody a numerické shody.

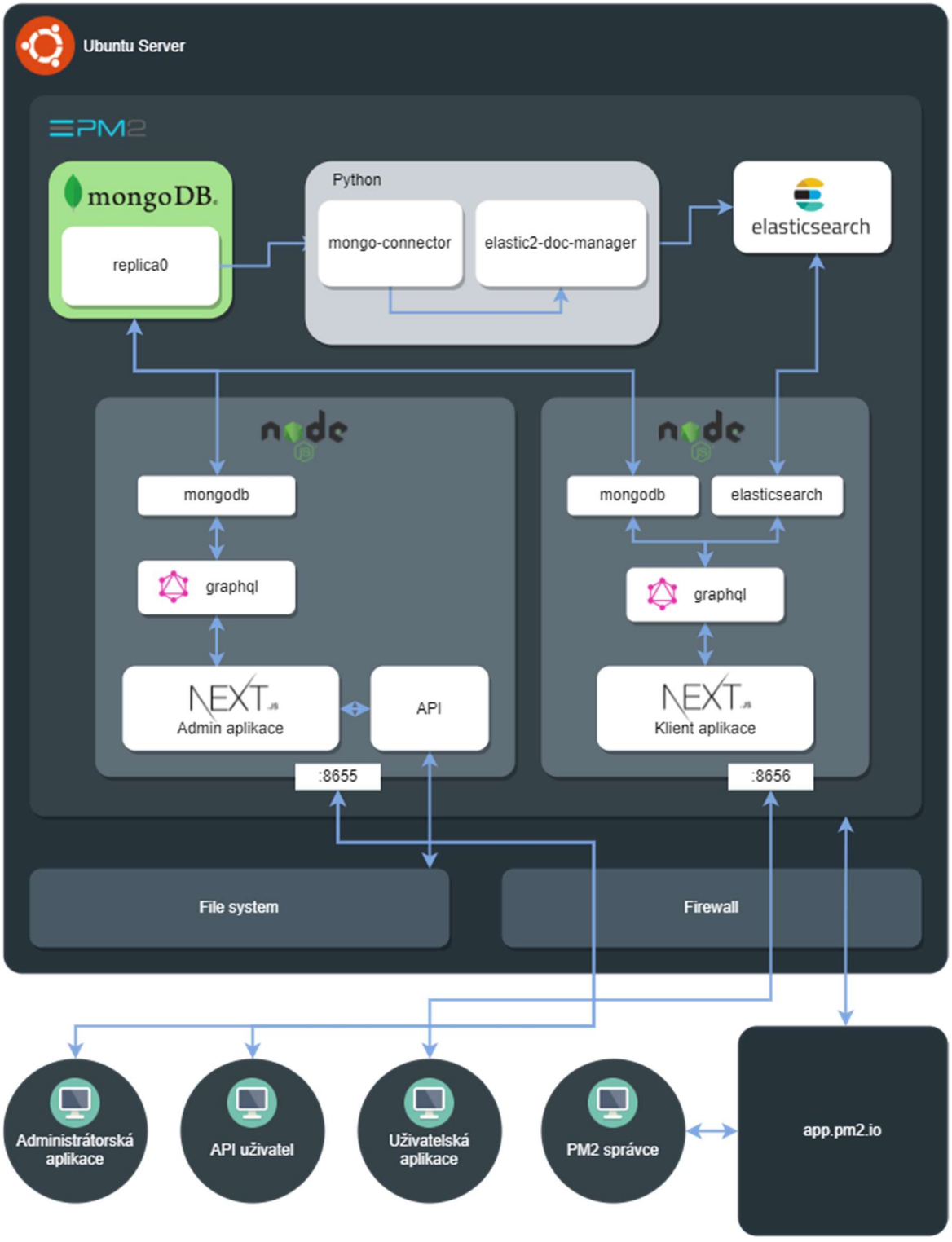
3.4.3 Mongo-connector

Mongo-connector je nástroj napsaný v jazyku Python, fungující jako služba, která v reálném čase kopíruje změny v dokumentech a posílá je požadovanému cíli. Dokument manažer se stará o překlad změn do základních CRUD operací, které jsou následně prováděny na cíli. V konfiguraci lze nastavovat jaké dokumenty a pole se mají indexovat, frekvenci zápisu změn, identifikace a jiné. [60]

V případě tohoto projektu se změny v databázi MongoDB předkládají pomocí *elastic2-doc-manager* a indexují do Elasticsearch. Indexují se všechny textové atributy z dokumentu písní i playlistů s okamžitým zápisem.

[q] hledání kritérií v řetězcích celého textového dokumentu

4. Architektura systému



Obr. 1 Diagram architektury systému

4.1 Popis

4.1.1 Server

Celý systém funguje na Ubuntu serveru verze 20.04.3 který je hostovaný na cloudové infrastruktuře DigitalOcean. Tento virtuální server funguje na konfiguraci se 4 jádry procesoru AMD EPYC, 8 GB RAM a 128 SSD. Všechny procesy jsou spravovány pomocí služby PM2, díky které je možné spojit službu se vzdálenou správou.

4.1.2 Node.js

Na host serveru funguje Node.js server pro administrátorskou část a zvlášť Node.js server pro uživatelskou část projektu. Na těchto serverech fungují webové single-page aplikace na frameworku Next.js, na kterých je kód převážně v TypeScriptu následně kompilován pomocí Babel. Next.js aplikace jsou vykreslovány SSR (Server-Site Rendering) v kombinaci s Pre-renderingem pro společné komponenty rozložení aplikace a jako dehydratovaný řetězec jsou posílány na prohlížeč uživatele, kde jsou zpět hydratovány a připraveny k použití.

4.1.3 Datová vrstva (Apollo GraphQL)

Aplikace je spojená s Apollo serverem s GraphQL datovou vrstvou pro komunikaci serveru a klienta s datovými zdroj. Datová vrstva se dotazuje na balíčky *mongodb* a *elasticsearch*, které se napojují na lokální služby databáze a indexového nástroje.

4.1.4 Admin API

Na administrátorském serveru je možné využít API pro správu databáze a nahrávání souborů na souborovém systému host serveru. K těm je možné přistupovat přes cestu /api/.

4.1.5 MongoDB

Dále na host serveru funguje instance MongoDB databáze, kde jsou uloženy data o jednotlivých položkách a statistická data. Je vytvořena i replika rs0 pro fungování indexování této databáze do Elasticsearch. MongoDB je i živě zálohována na cloudovou službu MongoDB Atlas.

4.1.5.1 Datový model dokumentů MongoDB

```
/* playlists */
{
    "_id": {
        "$oid": string
    },
    "name": string,
    "description": string,
    "image_path": string,
    "songs": [{
        "$oid": string
    }],
    "isPublic": true,
    "modifiedDate": {
        "$date": datetime
    },
    "createdDate": {
        "$date": datetime
    }
}
```

Obr. 4 JSON model prvku playlist

```
/* song */
{
    "_id": {
        "$oid": string
    },
    "name": string,
    "file_path": string,
    "image_path": string,
    "isPublic": true,
    "createdDate": {
        "$date": datetime
    },
    "modifiedDate": {
        "$date": datetime
    }
}
```

Obr. 5 JSON model prvku song

```
/* waveform */
{
    "_id": {
        "$oid": string
    },
    "songID": string,
    "waveform": [number]
}
```

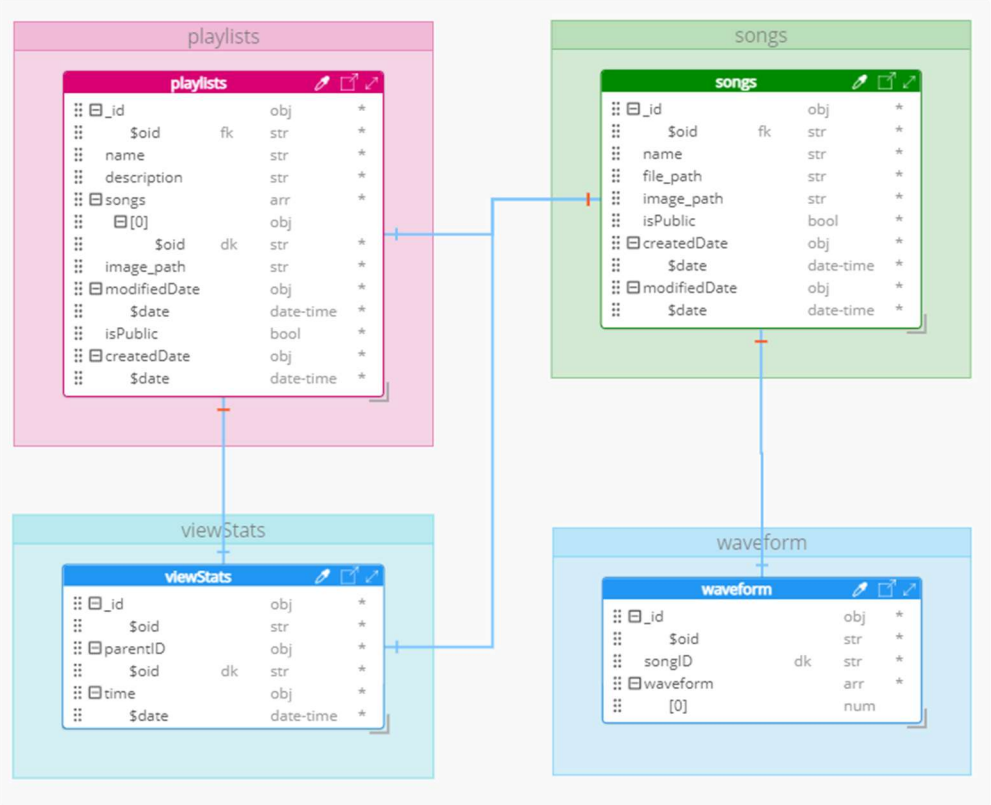
Obr. 3 JSON model prvku waveform

```
/* viewStats */
{
    "_id": {
        "$oid": string
    },
    "parentID": {
        "$oid": string
    },
    "time": [{
        "$date": datetime
    }]
}
```

Obr. 2 JSON model prvku viewStats

Dokumenty v kolekcích MongoDB databáze jsou JSON objekty. ID těchto objektů je speciální typ *ObjectID*, který v sobě drží kernelový čas vytvoření objektu. Díky tomuto identifikátorům tak lze filtrovat datum a čas přidání objektu do dokumentu.

4.1.5.2 Model volných relací mezi dokumenty



Obr. 6 Model relací mezi kolekcemi databáze

4.1.6 Elasticsearch

Vyhledávácí a indexovací služba Elasticsearch je také instancována na host serveru. V projektu je využívána pro vyhledávací funkce s možností fuzzy hledání. Data z MongoDB databáze jsou živě indexována podle šablony v nástroji Kibana. V této šabloně se nachází předpis pro filtrování a analýzu dat pro vykonávání dotazů.

Výběr filtrů používaných v analyzérech.

no_stop filtr zamezuje automatické odstraňování často vyskytujících se slov v užitém jazyce

extract_numbers filtruje pouze numerické výrazy z toku textu

ngram_filter určuje maximální délku ngram segmentů

4.1.6.1 Modely indexovaných objektů v Elasticsearch

```
/* song */

{
    "_index" : "koinoniamusicdbsongs",
    "_type" : "_doc",
    "_id" : string,
    "_score" : number,
    "_source" : {
        "name" : string
    }
}
```

Obr. 8 JSON model indexovaného prvku song

```
/* playlist */
{
    "_index" : "koinoniamusicdbplaylists",
    "_type" : "_doc",
    "_id" : string,
    "_score" : number,
    "_source" : {
        "name" : string,
        "description" : string
    }
}
```

Obr. 7 JSON model indexovaného prvku playlist

4.1.6.2 Analyzátory použité s atributy

ngram_token_analyzer – pro indexování převede všechny znaky na malá písmena a normalizuje je do ascii řetězce, dále zakáže odstraňování stopslov a index rozdělí na segmenty s délkou 1 až 10 znaků.

search_term_analyzer – pro vyhledávání indexovaného řetězce převede dotaz na malá písmena a normalizuje je do ascii řetězce bez odstraňování stopslov

exact_number_analyzer – z indexu vyfiltruje pouze čísla z desítkové soustavy

folding – převede řetězce do malých písmen a normalizuje je do ascii řetězce

```
/* binding analyzers */
{
  "songs": {
    "name": {
      "index": "ngram_token_analyzer",
      "search": "search_term_analyzer",
      "multi-fileds": {
        "extract_number": "exact_number_analyzer"
      }
    }
  },
  "playlists": {
    "name": {
      "index": "folding",
      "search": "folding",
      "multi-fileds": {
        "extract_number": "exact_number_analyzer"
      }
    },
    "description": {
      "index": "folding",
      "search": "folding"
    }
  }
}
```

Obr. 9 JSON vizualizace napojení analyzátorů

4.1.7 Mongo-connector

Je nástroj pro živé kopírování změn v MongoDB databázi s okamžitým překladem pro indexování do Elasticsearch pomocí *elastic2-doc-manager*. Python balíček je spuštěn jako služba spravovaná managerem PM2.

Mongo-connector kopíruje z MongoDB spolu s unikátním identifikátorem pole *name* z kolekce *songs* a soubor polí *name* a *description* z kolekce *playlist*, ve které lze vyhledat.

4.1.8 Převod zvuku do waveform

V rámci administrátorské webové aplikaci se při každém API volání přidávající nebo upravující záznam o písni v databázi vygeneruje nová množina hlasitostních vrcholů (waveform) které jsou následně zobrazovány při přehrávání písně.

Vrcholy mají hodnotu od 0 (ticho) do 1 (nejhlasitější část). Jsou vytvořeny převodem zvukového souboru do vyrovnávací paměti (*buffer*) pole, které je dále segmentováno podle počtu požadovaných vzorků. Hlasitost je následně zprůměrována ze všech vzorků v segmentu a přidána do výsledné množiny. Data jsou následně normalizována, aby vznikla škála mezi hlasitostí 0 a 1 u každé písně.

5. Funkce systému

5.1 Administrátorské funkce

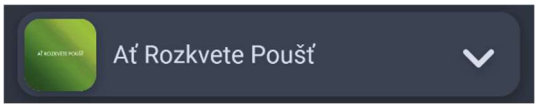
Webová aplikace je responzivní jak pro mobilní zařízení, tak i pro stolní počítače a laptopy. Je určena pro administrátora systému, který má možnosti přidávání, upravování a odstranění zvukových stop nebo jejich listů a pro prohlédnutí základní statistiky.

Obr. 10 Mobilní zobrazení aplikace

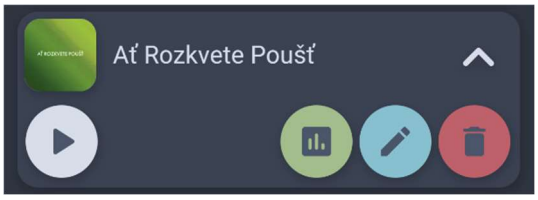
5.1.1 Seznam písní/playlistů

V základním přehledu je seznam všech uložených prvků databáze písní a playlistů. Jsou zobrazeny jako prvky listu.

Každá komponenta prvku má 2 fáze zobrazení. Základní fáze obsahuje pouze minimální informace, jako je titul a obrázek. Po kliknutí na tlačítko rozbalení se otevře 2. fáze zobrazení s ovládacími prvky pro administraci (statistika, úprava a smazání) a tlačítko pro přehrání. Po kliknutí na obrázek se položka přidá do výběru pro hromadné akce.



Obr. 12 Komponenta listu



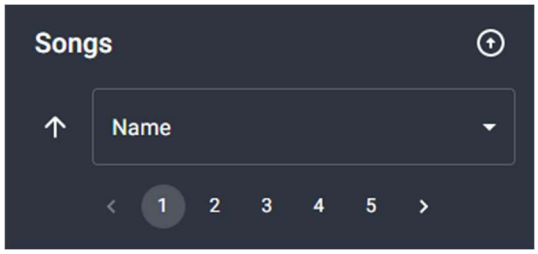
Obr. 11 Komponenta listu – rozšířená

První prvek listu je tlačítko pro přidání písně nebo playlistu do databáze.

Ve vrchní části listu se nachází lišta rychlých akcí, která obsahuje menu pro přepínání zobrazení a tlačítko pro zobrazení dalších nástrojů. Menu přepíná zobrazení mezi playlisty a písněmi. Po kliknutí na tlačítko dalších nástrojů se rozbalí možnosti seřazení a stránkování. Administrátor může seřadit seznam podle různých vlastností prvků uložených v databázi. Pokud je vybrán ze seznamu jeden nebo více prvků, lišta rychlých akcí se změní na hromadné akce, kde je možné prvky hromadně spravovat.

Obr. 15 Komponenta vrchní lišty – hromadné akce

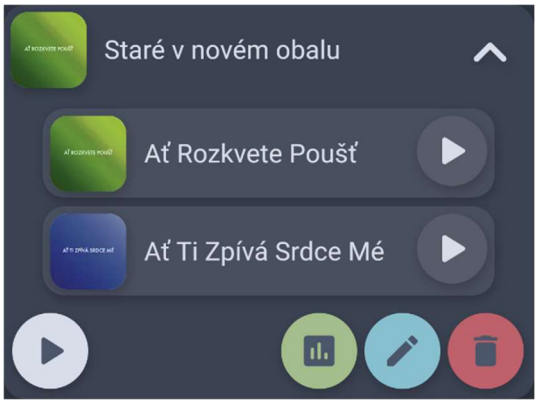
Obr. 13 Komponenta vrchní lišty – rychlé akce



Obr. 14 Komponenta vrchní lišty – seřazení a stránkování

Pokud je zvolen list playlistů, po rozkliknutí komponenty prvku se zobrazí i seznam písní, které jsou přiřazeny do tohoto seznamu.

Na konci každého listu se nachází ovládací prvek stránkování pro rychlý přechod mezi stránkami listu.



Obr. 16 Komponenta listu – rozšířená (playlist)

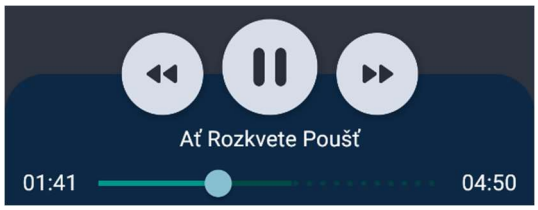
5.1.2 Přehrávač

Ve spodní části obrazovky se nachází přehrávač se základními ovládacími prvky. Po kliknutí na možnost přehrání na jakékoli položce ze seznamu se začne na přehrávači načítání a následně se spustí stream audio souboru ze serveru.

Přehrávač obsahuje prvky pozastavit/spustit, rychlý posun dopředu / zpět, který posune aktuálně poslouchanou část písně o 5 sekund dříve nebo později. V prostřední části

přehrávače se zobrazuje titul souboru, který je momentálně přehráván. Ve spodní části přehrávače je posuvník, který umožní administrátorovy rychle prohlížet části písničky. Tento posuvník zároveň ukazuje i aktuální čas převedený do vizuální podoby poslouchané písně. Vlevo od posuvníku se nachází vypsaný aktuální čas přehrávaného souboru a vpravo se nachází celková délka souboru.

Obr. 18 Přehrávač – zmenšený



Obr. 17 Přehrávač – plná velikost

Přehrávač má 2 fáze zobrazení. Plná velikost při prohlížení celého listu, kde obsahuje ovládací prvky, titul a časové informace. Zmenšená velikost se aktivuje v moment, kdy se otevře jakákoli dialogová karta. V tomto druhém zobrazení se nachází ovládací prvky, ale ve zmenšené velikosti a časové informace s posuvníkem. Nezobrazuje se však už titulek přehrávaného souboru.

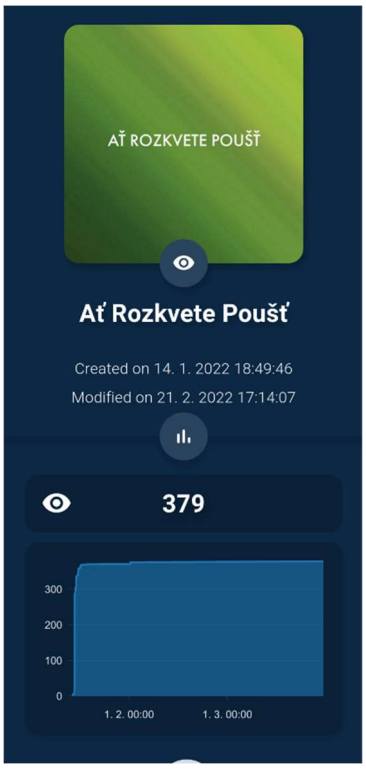
5.1.3 Možnosti správy

Po rozbalení komponenty prvku listu se nachází v pravé spodní části 3 tlačítka pro správu prvku. Mají za úkol otevřít různé dialogové karty, které slouží jako prostředí pro zobrazení statistiky, úpravu nebo smazání prvku.

5.1.3.1 Statistika

V této dialogové kartě se nachází informace o písni nebo playlistu. Zobrazují se zde vlastnosti jako jsou titul písně, podcastu nebo playlistu, datum vytvoření a datum poslední změny a stav zveřejnění. Ve hlavní části se nachází i obrázek tohoto prvku.

V části statistiky se nachází číselná informace o celkovém počtu přehrání a grafová statistika počtu přehrání v časovém rozmezí od začátku, až do přítomnosti.



Obr. 19 Karta statistik

5.1.3.2 Úprava

Tato dialogová karta nabízí úpravu již existujícího prvku databáze. U písní je možné upravovat její titul pomocí textového

vstupu, zviditelnění a zveřejnění pomocí označení checkboxu. Dále ve samodopňovacím poli lze přidávat a odebírat prvek z playlistů.

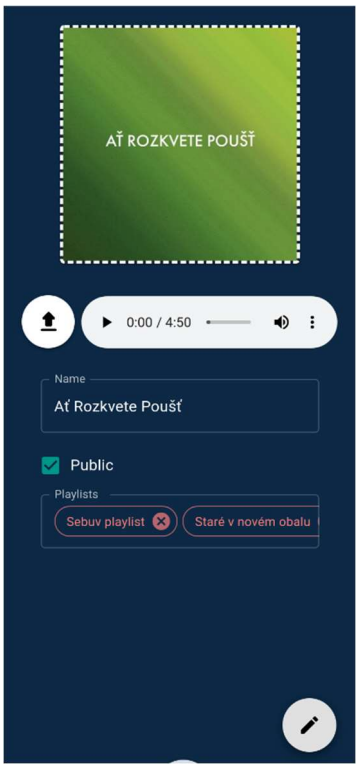
Úprava obrázku probíhá klepnutím na obrázek, kdy se otevře dialogové okno operačního systému pro výběr souboru. Druhý způsob je využití možnosti drag & drop funkce, kdy po přetažení souboru nad komponentu obrázku. Automaticky se tak nahraje nový soubor na server, ale zatím se nepropíše do databáze.

Podobným způsobem funguje i nahrávaní audio souboru, ale s rozdílem, že vlevo je tlačítko pro nahrání. Drag & drop funkce zde funguje při přetažení nad komponentu.

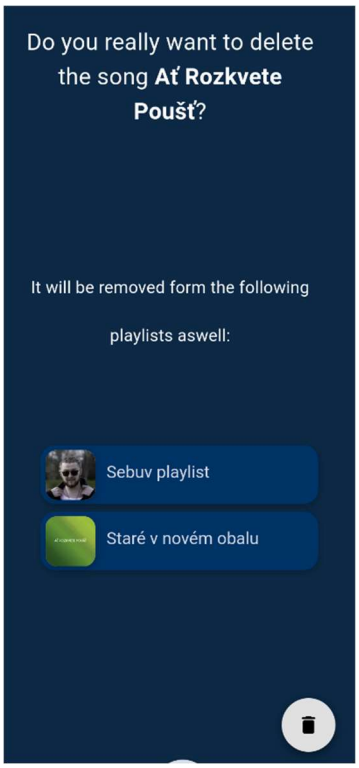
Při úpravě playlistu se přidává textový vstup pro popisek. Mizí zde vstup pro přidávání/odebírání playlistů a je nahrazen listem aktuálních písní, které jsou součástí tohoto playlistu. Komponentu tohoto listu lze přehrát a odstranit.

5.1.3.3 Odstranění

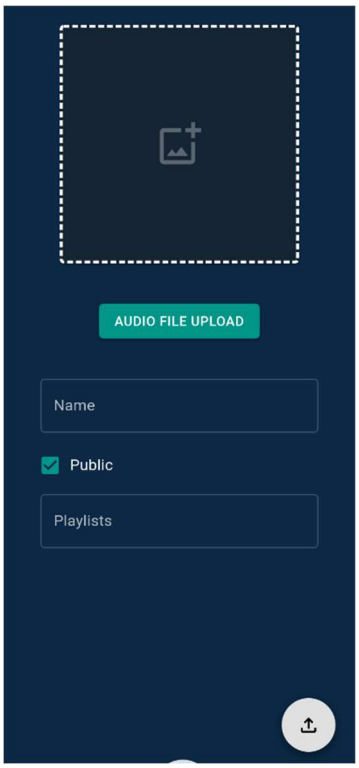
Tato dialogová karta slouží jako upozornění před smazáním playlistu nebo písně. Obsahuje textovou rekapitulaci titulu a list všech prvků databáze, které obsahují referenci na tento prvek. V případě mazání playlistů se objeví v listu všechny písně, které jsou součástí playlistů. V případě mazání písně se naopak objeví všechny playlisty, které obsahují v sobě tuto konkrétní píseň.



Obr. 20 Karta úpravy



Obr. 21 Karta odstranění



Obr. 22 Karta přidání

5.1.3.4 Přidat novou píseň/playlist

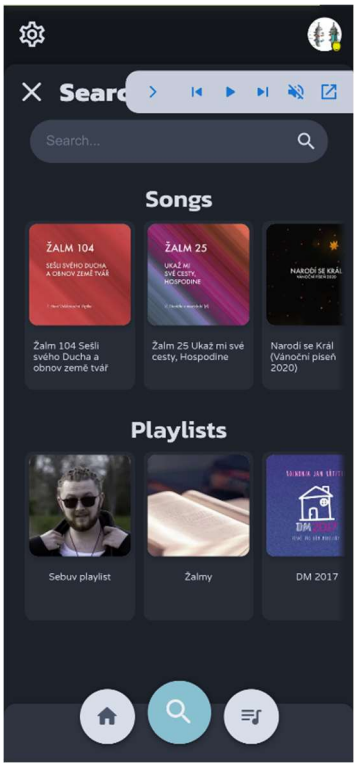
Tato možnost se nachází jako první prvek v hlavním listu. Slouží jako možnost pro přidání nového playlistu nebo písně do databáze a pro vytvoření reference na streamovatelný soubor. Po kliknutí na tuto možnost se otevře dialogová karta, ve které lze uvádět veškeré potřebné vlastnosti, nahrávat soubory a přidávat písně do playlistů podobně jako v kartě úpravy.

5.2 Uživatelské funkce

Druhá webová aplikace hlavně pro mobilní zařízení. Je určena pro uživatele na vyhledávání a poslech hudby. Uživatel může konkrétní skladby a playlisty vyhledat, nebo si je nechat doporučit. Uživatel může přidávat hudbu do fronty poslechu, která se bude přehrávat za sebou. Díky přihlášení má uživatel možnost vytvářet vlastní playlisty a konkrétní písně označovat za oblíbené.

Stránka pro mobilní zobrazení je rozložená na hlavní stránku podle obsahu a navigační funkce. Ve spodní části jsou při tlačítka pro orientaci mezi 3 základními stránkami (feed, search, library). V horní části obrazovky se nachází výsuvné menu, kde se může uživatel přihlásit a spravovat svůj účet a otevřít základní nastavení.

Všechny hlavní stránky obsahují i malé ovládání přehrávače pro rychlé akce jako spuštění / zastavení, přeskočení a ztišení přehrávané písně. Zmenšenou verzi, kde lze ovládat jen spuštění a zastavení lze zvětšit pro více ostatní funkce a otevření karty přehrávače.



Obr. 23 Mobilní zobrazení aplikace – rozbalené vrchní menu a ovládání přehrávače

5.2.1 Domovská stránka (feed)

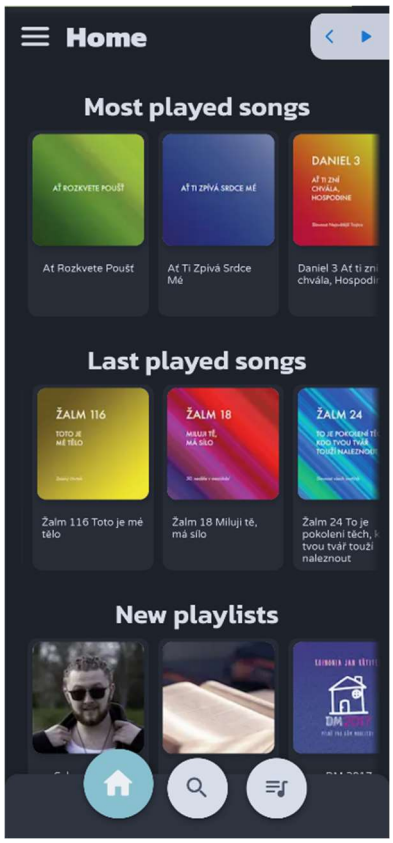
Na domovské stránce se nachází výběr doporučených písní a playlistů na základě různých kritérií. Uživatel zde najde například kategorie globálně nejpopulárnější, nejčastěji poslouchané uživatelem, naposledy poslouchané a jiné. Položky jsou zobrazovány jako náhledový obrázek a titul. Uživatel může rozkliknout položku, která otevře kartu detail pro více informací a další funkce.

5.2.2 Stránka vyhledávání (search)

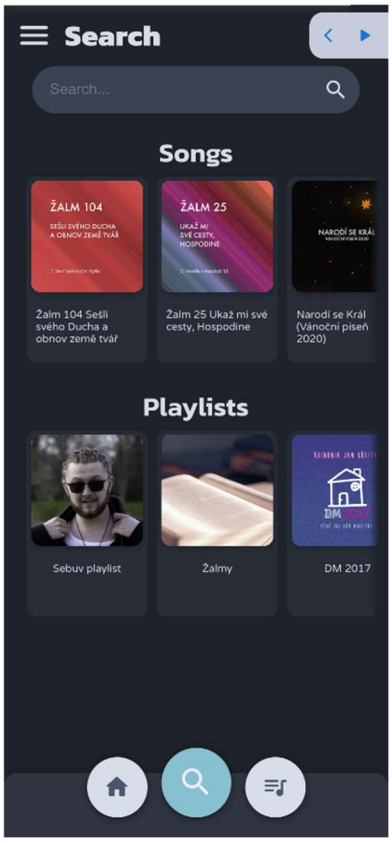
Stránka vyhledávání obsahuje pole hledání, které zobrazí pod sebou výsledky písní a playlistů, shodující se v kritériích. Vyhledávání probíhá s funkcí fuzzy, které zobrazují i podobné výsledky na základě přehazování písmen. Vyhledávání může v různých případech využití preferovat i číselné shody nad částečnou shodou nebo fuzzy shodou. Zobrazované výsledky se nachází pod vyhledávacím polem v samostatných skupinách písní a playlistů.

5.2.3 Stránka knihovny (library)

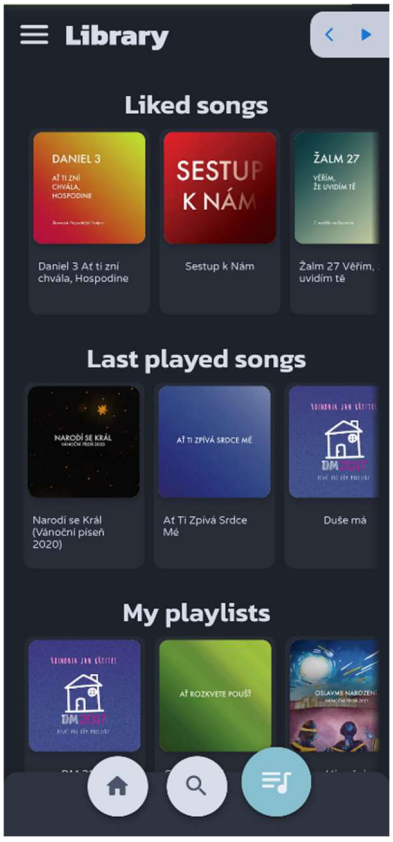
Na této stránce se nachází obsah uživatele. Ve složkách a skupinách se zobrazují uživatelem oblíbené písně, vlastní a převzaté playlisty a jiný zvolený obsah.



Obr. 26 Domovská stránka (feed)



Obr. 24 Stránka vyhledávání



Obr. 25 Stránka knihovny – uživatelský obsah

5.2.4 Detail písně / playlistu

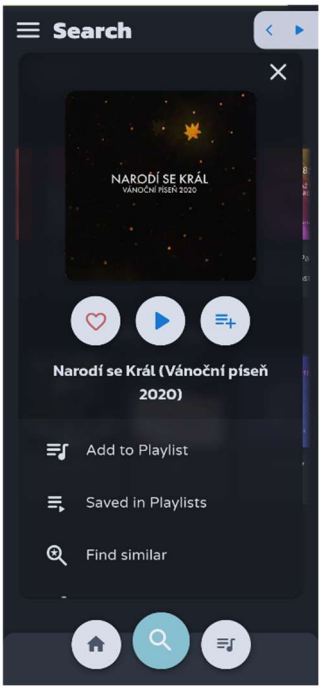
Karta zobrazená po rozkliknutí náhledu písně nebo playlistu. Nachází se zde základní informace jako titulek a úvodní obrázek. Uživatel na této kartě může položku přidat do oblíbených, spustit nebo přidat do fronty. Pokud je položka playlist, zobrazí se i seznam písní, které se nachází v seznamu. V spodní části se nachází operace jako přidání do playlistu.

5.2.5 Přehrávač

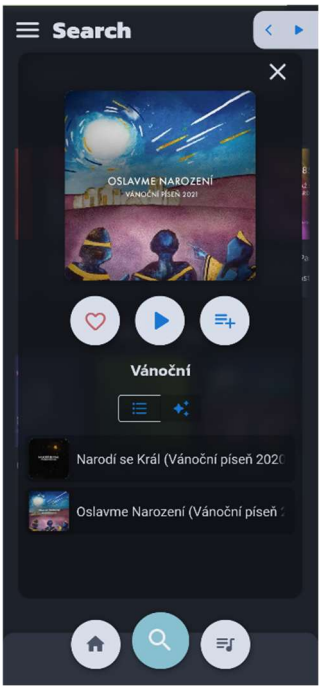
Po spuštění písně z karty detailu nebo po zvětšení malého ovladače se otevře karta přehrávače, kde se zobrazuje právě přehrávaná píseň. Ve vrchní části karty se zobrazuje úvodní obrázek a titul písně.

Ve spodní části se nachází tlačítka ovládání zastavit / spustit přehrávání a přeskočit na předchozí a další píseň. Nad ovládacími prvky se nachází i vizualizace hlasitostních vrcholů, které zároveň fungují jako zobrazení aktuálního času a posun přehrávání písně.

Ve druhém zobrazení přehrávače se nachází fronta, kde jsou za sebou zobrazeny následující písně. Po skončení jedné se automaticky pustí první z fronty. Pořadí lze upravovat posouváním položek ve frontě dotykem.



Obr. 27 Karta detailu (píseň)



Obr. 29 Karta detailu (playlist)

Obr. 30 Karta přehrávače – přehrávač

Obr. 28 Karta přehrávače – řada písní

5.3 API

Na serveru administrátorské aplikace se nachází API pro další možné projekty pro administrátorskou správu. Jejím účelem je spravovat položky písní a playlistu na databázi a nahrávání nových souborů pro písně a převod do formátu HLS.

5.3.1 Upload souboru

Pro nahrávání nových souborů do veřejné složky na serveru z formuláře. Přímá soubory nahrané do formuláře a destinaci pro uložení souboru.

```
/* POST */
/* Destination: /api/upload */
{
    headers: {
      "content-type": "multipart/form-data",
      "destination": string,
    },
    data: formData
}
```

Obr. 31 Hlavička POST žádosti – nahrávání souboru

Soubor přijatý z objektu formData je přečten spolu s hlavičkou, kde se nachází destinace souboru. Multer následně uloží soubor na požadované místo pod normalizovaným názvem souboru.

5.3.2 Správa HLS a písní

5.3.2.1 hlsCreate

Pro vytvoření HLS z nahraného souboru. Přímá v hlavičce lokaci zdroje, lokaci složky, do které se uloží výsledek, samotnou budoucí lokaci souboru, lokaci budoucího m3u8 souboru, lokaci nahraného obrázku, název písně a případně viditelnost písně v listech. Po provedení API vrátí hlášku „success“, kód 200 a odpověď databáze na novou položku.

```
/* POST */
/* Destination: /api/hlsCreate */
{
    headers: {
      "name": string,
      "source": string,
      "destination": string,
      "destinationFolder": string,
      "destinationFile": string,
      "destinationImage": string,
      "isPublic": boolean,
      "audioLevels": [number]
    }
}
```

Obr. 32 Hlavička POST žádosti – hlsCreate

Po zpracování a uložení HLS souboru se se všemi potřebnými atributy nahraje jako nová píseň na databázi MongoDB.

5.3.2.2 hlsUpdate

HLS soubor i s databází lze aktualizovat pomocí této API. Systém v hlavičce příjme všechny potřebné informace. Informace, které nemají být zmíněny budou stejné, jako je v záznamu databáze. Nejdříve se vytvoří a uloží nový HLS ze souboru na disku, následně jsou nové informace zapsány do databáze. Pokud vše proběhne správně, API vrátí jako data „success“ spolu s kódem 200 a odpověď aktualizované položky v databázi.

```
/* POST */
/* Destination: /api/hlsUpdate */
{
    headers: {
      "_id": string,
      "name": string,
      "source": string,
      "destination": string,
      "destinationFolder": string,
      "destinationFile": string,
      "destinationImage": string,
      "isPublic": boolean,
      "imageIsInitial": boolean,
      "fileIsInitial": boolean,
      "possibleInitialImage": string,
      "possibleInitialFile": string,
      "audioLevels": [number]
    }
}
```

Obr. 33 Hlavička POST žádosti – hlsUpdate

V hlavičce POST požadavku se posílá id existujícího prvku v databázi, zdrojový soubor pro HLS, složku cíle, cesta pro obrázek, cesta pro HLS soubor, veřejná cesta pro zvukový soubor, název, zveřejnění, název obrázku, logická hodnota, zda je soubor již na disku, název souboru a hlasitostí vrcholy

5.3.2.3 hlsDelete

Pro smazání souboru písně a jeho záznamu v databázi lze použít tuto API. V hlavičce POST žádosti je požadováno id záznamu v databázi, který má být smazán a cesta k HLS souboru. Pokud celý proces proběhne v pořádku, API zpět pošle kód 200, data „success“, odpovědí databáze a stav mazání.

```
/* POST */
/* Destination: /api/hlsDelete */
{
    headers: {
      "id": string,
      "destinationImage": string,
      "pathtofile": string
    }
}
```

Obr. 34 Hlavička POST žádosti – hlsDelete

5.3.3 Správa playlistů

5.3.3.1 playlistCreate

Tento API přístup je pro přidávání záznamu o playlistu do databáze. V hlavičce volání jsou vyžadovány informace o názvu playlistu, jeho popis, cesta k obrázku obálky na disku a logická hodnota o jeho veřejnosti. API zpět vrátí při úspěšném provedení kód 200, data se stavem „success“ a odpověď z databáze.

```
/* POST */
/* Destination: /api/playlistCreate */
{
    headers: {
      "name": string,
      "description": string,
      "destinationImage": string,
      "isPublic": boolean
    }
}
```

Obr. 35 Hlavička POST žádosti – playlistCreate

5.3.3.2 playlistUpdate

Pro úpravu záznamu o playlistu v databází je poskytnut tento API přístup. Pro úpravu jsou vyžadována i původní data pro neupravované pole a upravené atributy. Do hlavičky tohoto volání patří id objektu v databázi, titul playlistu, popisek, cestu k souboru obrázku pro obálku, logickou hodnotu o veřejnosti objektu, logickou hodnotu o využití starého souboru na disku a název souboru obrázku. API při úspěšném splnění požadavku vrátí kód 200, data s hláškou „success“ a id upravené položky v databázi.

```
/* POST */
/* Destination: /api/playlistUpdate */
{
    headers: {
      "_id": string,
      "name": string,
      "description": string,
      "destinationImage": string,
      "isPublic": boolean,
      "imageisinitial": boolean,
      "possibleinitialimage": string
    }
}
```

Obr. 36 Hlavička POST žádosti – playlistUpdate

5.3.3.3 playlistAddSong

Do záznamech o playlistech jde přidávat i reference na písně do množiny referencí. V hlavičce tohoto API přístupu se vepisuje identifikátor záznamu playlistu a identifikátor existujícího záznamu o písni.

```
/* POST */
/* Destination: /api/playlistAddSong */
{
    headers: {
      "id": string,
      "song": string,
    }
}
```

Obr. 37 Hlavička POST žádosti – playlistAddSong

5.3.3.4 playlistRemoveSong

Referenci na píseň v playlistu lze odebrat pomocí tohoto API volání. V hlavičce se nachází id objektu playlistu a id písně. Odpověď při úspěchu je kód 200, hláška „success“ a odpověď databáze.

```
/* POST */
/* Destination: /api/playlistRemoveSong */
{
    headers: {
      "id": string,
      "song": string,
    }
}
```

Obr. 38 Hlavička POST žádosti – playlistRemoveSong

5.3.3.5 playlistDelete

Záznam o playlistu lze smazat z databáze pomocí této API funkce. V hlavičce se nachází jen id objektu a odpověď je kód 200, „success“ a odpověď databáze.

```
/* POST */
/* Destination: /api/playlistDelete */
{
    headers: {
      "id": string
    }
}
```

Obr. 39 Hlavička POST žádosti – playlistDelete

6. Závěr

Tato práce má za výsledek prvotní verzi systému složeného ze dvou webových aplikací pro administrativu a uživatelský poslech. Aplikace jsou živě spuštěny na cloudovém hostingu DigitalOcean spolu s databázemi a s nimi spojenými nástroji. K administrátorské aplikaci je k dispozici i API správa. Tyto aplikace lze nasadit na libovolný linuxový server. Administrátor však musí změnit konfiguraci databáze a kód webových aplikací a API, aby byl systém vyhovující vlastnímu případu užívaní.

Při dalším vývoji má tento systém potenciál fungovat v aktivním nasazení pro užívání v menších skupinách jako je společenství *Koinonia* pro sdílení vlastní tvorby.

Pro další verze projetu je plánováno ustálení grafického designu aplikace a přidání animací při interakci. Dále pokročilá nastavení pro uživatele jako je ekvalizace, vlastní kvalita zvuku, konfigurace fuzzy vyhledávání a jiné.

Jednou z dalších funkcí může být propojení více serverů pro dosažení pseudo-centralizovaného zážitku se sdíleným obsahem a využití algoritmů pro doporučení obsahu na základě podobnosti žánrů.

Projekt nakonec nebyl zpracován jako mobilní aplikace, ale tato možnost je pořád v plánu příštího vývoje. Aplikace by však musela vzniknout samostatně za pomocí knihovny React Native pro mobilní zařízení. Místo toho jsou webové aplikace určeny svým rozložením převážně pro mobilní zařízení.

Projekt má také přínos pro mě, jako pro vývojáře. Díky řešení logických problémů s kompatibilitou jsem se přiučil novým způsobům, jak pracovat s technologiemi pro webový vývoj, omezeními různých výrobců softwaru a hardwaru a databázovými nástroji pro indexování a vyhledávání.

7. Reference

[1] H. C. Yulianto a A. N. H. Oroh, „The Effects of Social Value, Value for Money, App Rating, and Enjoyment on the Intention to Purchase the Premium Service of the Spotify App,“ 22 Březen 2021. [Online]. Available: http://3.65.204.3/index.php/KnE-Social/article/view/8815.

[2] TRADING ECONOMICS, „Internet Speed by Country,“ TRADING ECONOMICS, 31 Březen 2017. [Online]. Available: https://tradingeconomics.com/country-list/internet-speed. [Přístup získán 25 Leden 2022].

[3] S. Carbone, „Free Spotify vs Spotify Premium,“ Sound Guys, 12 Červenec 2021. [Online]. Available: https://soundguys.com/free-spotify-vs-spotify-premium-36632/#tablepress-9. [Přístup získán 25 January 2022].

[4] M. O'Dair a A. Fry, „Beyond the black box in music streaming: the impact of recommendation systems upon artists,“ 10 Jun 2019. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/15405702.2019.1627548. [Přístup získán 25 Leden 2022].

[5] D. Hesmondhalgh, E. Jones a A. Rauh, „SoundCloud and Bandcamp as Alternative Music Platforms,“ 26 Listopad 2019. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/2056305119883429.

[6] G. Breux, „Client-side vs. Server-side vs. Pre-rendering for Web Apps,“ Toptal Developers, [Online]. Available: https://www.toptal.com/front-end/client-side-vs-server-side-pre-rendering.

[7] Spotify, „What is Spotify?,“ Spotify AB, 2 Únor 2022. [Online]. Available: https://support.spotify.com/us/article/what-is-spotify/.

[8] Spotify, „Premium plans,“ Spotify AB, 16 Únor 2022. [Online]. Available: https://support.spotify.com/us/article/premium-plans/.

[9] Spotify, „Getting music on Spotify,“ Spotify AB, [Online]. Available:

https://artists.spotify.com/help/article/getting-music-on-spotify?ref=claimflow.

[10] Spotify, „Supported devices for Spotify,“ Spotify AB, 11 Březen 2022. [Online]. Available: https://support.spotify.com/us/article/supported-devices-for-spotify/.

[11] W3Techs, „Site Info - Spotify.com,“ Q-Success, [Online]. Available: https://w3techs.com/sites/info/spotify.com.

[12] A. Blixt, „How did Spotify make a multiplatform, lightweight, well-designed desktop application? What is the technology behind it? Answer by Andreas Blixt,“ Quora, Inc., [Online]. Available: https://www.quora.com/How-did-Spotify-make-a-multiplatform-lightweight-well-designed-desktop-application-What-is-the-technology-behind-it.

[13] B. Edström, „In praise of “boring” technology,“ Spotify AB, 25 Únor 2013. [Online]. Available: https://engineering.atspotify.com/2013/02/in-praise-of-boring-technology/.

[14] Apache Cassandra, „Apache Cassandra’s documentation,“ The Apache Software Foundation, [Online]. Available: https://cassandra.apache.org/doc/latest/.

[15] L. DiFelice, „The CAP Theorem With Apache Cassandra® and MongoDB,“ Instaclustr, [Online]. Available: https://www.instaclustr.com/blog/cassandra-vs-mongodb/.

[16] M. Tillman, „What is Apple Music and how does it work?,“ Pocket-lint Limited, [Online]. Available: https://www.pocket-lint.com/apps/news/apple/136725-what-is-apple-music-and-how-does-it-work.

[17] J. Clover, „Apple Music: Our Complete Guide,“ MacRumors.com, LLC, 14 Prosinec 2021. [Online]. Available: https://www.macrumors.com/guide/apple-music/.

[18] W3Techs, „Site Info - Apple.com,“ Q-Success, [Online]. Available: https://w3techs.com/sites/info/apple.com.

[19] C. Silva, „Is Apple Music for Android written in Java or Kotlin?,“ Quora, Inc., 18 Říjen 2018. [Online]. Available: https://www.quora.com/Is-Apple-Music-for-Android-written-in-Java-or-Kotlin.

[20] FoundationDB, „FoundationDB 6.3 Documentation,“ Apple, Inc, [Online]. Available: https://apple.github.io/foundationdb/.

[21] A. A. Demarest, „What is SoundCloud? Everything you need to know about the music and podcast platform,“ Insider Inc., 26 Únor 2021. [Online]. Available: https://www.businessinsider.com/what-is-soundcloud.

[22] D. Noël, „What technologies is SoundCloud built on? This includes their backend, front end, database, language, etc. Answered by David Noël,“ Quora, Inc., 18 Srpen 2012. [Online]. Available: https://www.quora.com/What-technologies-is-SoundCloud-built-on-This-includes-their-backend-front-end-database-language-etc.

[23] tim-thimmaiah, „Dev stack of SoundCloud,“ StackShare, Inc., [Online]. Available: https://stackshare.io/soundcloud/soundcloud.

[24] S. Treadway, „Evolution of SoundCloud’s Architecture,“ SoundCloud Global Limited & Co. KG, 30 Srpen 2012. [Online]. Available: https://developers.soundcloud.com/blog/evolution-of-soundclouds-architecture.

[25] WHATWG, „HTML Living Standard,“ 17 Březen 2022. [Online]. Available: https://html.spec.whatwg.org/multipage/.

[26] MDN Web Docs, „HTML: HyperText Markup Language,“ Mozilla Corporation, 1998-2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Přístup získán 18 Březen 2022].

[27] MDN Web Docs, „CSS: Cascading Style Sheets,“ Mozilla, 21 Leden 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS.

[28] MDN Web Docs, „JavaScript,“ Mozilla, 28 Červenec 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[29] MDN Web Docs, „Web APIs,“ Mozilla, 14 Září 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API.

[30] Microsoft, „TypeScript Documentation,“ Microsoft, [Online]. Available: https://www.typescriptlang.org/docs/.

[31] MongoDB, „MongoDB Shell (mongosh),“ MongoDB, Inc., [Online]. Available: https://docs.mongodb.com/mongodb-shell/.

[32] Elasticsearch B.V., „Query DLS,“ Elasticsearch B.V., [Online]. Available:

] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html.

[33 PyPA, „Python 3.10.3 documentation,“ Python Software Foundation, [Online]. Available:
] https://docs.python.org/3/.

[34 PyPA, „Python Packaging User Guide,“ Python Software Foundation, [Online].
] Available: https://packaging.python.org/en/latest/.

[35 PyPA, „Installing Packages¶,“ Python Software Foundation, [Online]. Available:
] https://packaging.python.org/en/latest/tutorials/installing-packages/.

[36 npm, „About npm,“ Npm, Inc, [Online]. Available: https://www.npmjs.com/about.
]

[37 npm open-source documentation, „npm, javascript package manager,“ Npm, Inc.,
] [Online]. Available: https://docs.npmjs.com/cli/v6/commands/npm.

[38 W3School, „Node.js Introduction,“ Refsnes Data, [Online]. Available:
] https://www.w3schools.com/nodejs/nodejs_intro.asp.

[39 Facebook React, „React,“ Meta Platforms, Inc., [Online]. Available: https://reactjs.org/.
]

[40 Facebook React Native, „React Native,“ Meta Platforms, Inc., [Online]. Available:
] https://reactnative.dev/.

[41 Next.js, „Next.js documentation,“ Vercel Inc., [Online]. Available:
] https://nextjs.org/docs/getting-started.

[42 Apple, „Using Apple's HTTP Live Streaming (HLS) Tools,“ Apple Inc., [Online].
] Available:
https://developer.apple.com/documentation/http_live_streaming/using_apple_s_http_live_streaming_hls_tools.

[43 Apple, „HTTP Live Streaming,“ Apple, Inc., [Online]. Available:
] https://datatracker.ietf.org/doc/html/rfc8216#section-1.

[44 Formik, „Formik documentation,“ Formium, Inc., [Online]. Available:
] https://formik.org/docs/overview.

[45] R. Caldwell, „What is Formik?,“ Liquid Web, LLC, [Online]. Available: https://www.liquidweb.com/kb/formik-react/.

[46] C. (. Pete, „ReactPlayer,“ GitHub, Inc., [Online]. Available: https://github.com/CookPete/react-player.

[47] Video.js, „Video.js Documentation,“ Brightcove, Inc., [Online]. Available: https://docs.videojs.com/.

[48] Apple, „Delivering Video Content for Safari,“ Apple Inc., [Online]. Available: https://developer.apple.com/documentation/webkit/delivering_video_content_for_safari.

[49] L. (. Jinke, „react-jinke-music-player,“ GitHub, Inc., [Online]. Available: https://github.com/lijinke666/react-music-player.

[50] MUI, „Supported components,“ Material-UI SAS, [Online]. Available: https://mui.com/getting-started/supported-components/.

[51] R. Benitte, „About Nivo,“ Nivo, [Online]. Available: https://nivo.rocks/about/.

[52] styled-components, „Styled components documentation,“ styled-components, [Online]. Available: https://styled-components.com/docs.

[53] SortableJS, „Sortable repository,“ GitHub, Inc., [Online]. Available: https://github.com/SortableJS/Sortable#readme.

[54] FFmpeg , „FFmpeg Documentation,“ FFmpeg , [Online]. Available: https://ffmpeg.org/documentation.html.

[55] M. Zabriskie, „Axios - Promise based HTTP client for the browser and node.js,“ Axios, [Online]. Available: https://axios-http.com/docs/intro.

[56] V. (. Hoang, „next-connect repository read me,“ GitHub, Inc., [Online]. Available: https://github.com/hoangvvo/next-connect#readme.

[57] MongoDB, „What Is MongoDB?,“ MongoDB, Inc., [Online]. Available: https://www.mongodb.com/what-is-mongodb.

[58] R. Haldar a D. Mukhopadhyay, „Levenshtein Distance Technique in Dictionary Lookup,“

] 6 Leden 2011. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1101/1101.1232.pdf.

[59 Elasticsearch, „Elasticsearch Guide," Elasticsearch B.V., [Online]. Available:] https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.

[60 YouGov (yougov), „mongo-connector wiki," YouGov PLC, [Online]. Available:] https://github.com/yougov/mongo-connector/wiki.

8. Seznam obrázků