

# TECHNICKÁ UNIVERZITA V KOŠICIACH

Fakulta elektrotechniky a informatiky

## Dokumentácia

Dátové inžinierstvo – zadanie  
2025/26

Bc. Dávid Teplan	33%
Bc. Pavel Páleník	33%
Bc. Bernadeta Gajdošová	33%
<b>Dátum:</b> 12/2025	

# Obsah

<b>Dokumentácia .....</b>	<b>1</b>
<b>Úvod .....</b>	<b>4</b>
<b>1. Poskytovanie našich dát .....</b>	<b>5</b>
1.1. Modul simulovania počtu poslucháčov .....	5
1.2. Modul poskytovania údajov .....	5
<b>2. Zber dát .....</b>	<b>5</b>
2.1. Moduly pre jednotlivé rádiá .....	5
2.2. Modul ukladania dát .....	6
2.3. Modul časovania a paralelného spracovania .....	7
<b>3. ER diagram .....</b>	<b>9</b>
3.1. Entita play_session .....	9
3.2. Entita song_session .....	10
3.3. Entita time .....	10
3.4. Entita song .....	10
3.5. Entita genre .....	10
3.6. Entita radio .....	10
3.7. Vzťahy medzi entitami .....	10
<b>4. Údaje o dátach .....</b>	<b>12</b>
<b>5. Spracovanie dát .....</b>	<b>13</b>
5.1. Modul transformácie dát .....	13
5.2. Modul obohatenia dát .....	13
5.3. Modul prevodu duration na rovnaké jednotky (sekundy) .....	14
5.4. Modul premapovania genre .....	14
5.5. Modul na naplnenie databázy .....	14
5.5.1. Záznamy všetkého okrem listener_count .....	14
5.5.2. Záznamy listener_count .....	14
6.1. Informačný balíček .....	16
6.2. STAR schéma .....	16

<b>6. ETL procesy .....</b>	<b>17</b>
<b>7. Použité služby a technológie .....</b>	<b>18</b>
<b>8. Záver .....</b>	<b>25</b>

# Úvod

Zadanie z predmetu Dátové inžinierstvo sa zaoberá zberom, spracovaním a analýzou dát o skladbách z rôznych rádií. Primárnym cieľom je transformovať tieto dáta do vhodnej štruktúry pre dátové sklady, kde budú pripravené na ďalšie využitie, ako napríklad reportovanie, analytiku alebo podporu rozhodovania.

V tomto konkrétnom prípade boli dáta zozbierané zo šiestich rádií: Melody, Expres, ROCK, Jazz, Vlna, Funradio a Beta. Zber dát prebiehal v rozmedzí od 29.10.2025 až 01.12.2025 a zahŕňa informácie o skladbe (názov, autori, čas kedy začala hrať, dátum prehratia a ďalšie informácie).

Po zozbieraní dát je potrebné ich čistiť, transformovať a integrovať do jednotného formátu, ktorý umožní efektívnu analýzu. Výstupom je dátový sklad, ktorý poskytuje dáta na vizualizovanie.

Projekt poskytuje príležitosť aplikovať teoretické poznatky o spracovaní údajov na reálny scenár a prispieva k pochopeniu využitia moderných technológií v oblasti leteckej dopravy.

# 1. Poskytovanie našich dát

Vytvorili sme službu, ktorá poskytovala cez HTTP GET a WebSockets údaje o skladbe z nášho rádia a simulované hodnoty o počte poslucháčov. Služba bežala online na cloudovej platforme Fly.io.

## 1.1. Modul simulovania počtu poslucháčov

Modul generuje simulované hodnoty počtov poslucháčov podľa vopred zadaných podmienok. Definovali sme nočný útlm poslucháčov – počet poslucháčov je nižší. Počas rána a poobedných časoch sa môže dosahovať špička počúvanosti – hodnoty sú vyššie. Počas víkendu je vyššia počúvanosť poobede.

## 1.2. Modul poskytovania údajov

Táto časť robí scrapovania<sup>1</sup> webu [radio.sk/radio/melody/playlist](http://radio.sk/radio/melody/playlist) a ukladá zoznam skladieb do JSON súboru, pri každom spustení sa pridajú len nové skladby. V pravidelných intervaloch sa sleduje, či sa zmenila aktuálna skladba. Ak sa zmení, tak ju budeme poskytovať cez HTTP GET / WebSockets, zároveň rovnakým spôsobom poskytujeme počet poslucháčov, ktoré sa mení v intervale 5-20sekúnd.

# 2. Zber dát

Zber dát je zabezpečený samostatnou serverovou službou, ktorá bola nasadená a prevádzkovaná na cloudovej platforme DigitalOcean. Vďaka tomuto riešeniu mohla služba bežať nepretržite a nezávisle od klientskych zariadení. Server v pravidelných intervaloch získaval údaje z viacerých externých zdrojov a spracovával ich na jednom centrálnom mieste.

Cieľom zberu dát bolo získavať informácie o aktuálne prehrávanom obsahu v rozhlasových staniciach spolu s doplnkovými údajmi, ako sú napríklad názvy skladieb, interpreti alebo počet poslucháčov. Tieto dáta sa následne upravovali do jednotného formátu a ukladali pre ďalšie spracovanie a analýzu.

## 2.1. Moduly pre jednotlivé rádiá

Každé rádio má v systéme vlastný modul, ktorý rieši získavanie dát z konkrétneho zdroja. Keďže jednotlivé rádiá poskytujú údaje rôznym spôsobom, tieto moduly obsahujú špecifickú logiku potrebnú na ich spracovanie.

---

<sup>1</sup> Prebratie hodnoty z HTML elementu

Takéto rozdelenie umožňuje jednoduché pridanie nového rádia bez nutnosti meniť zvyšok programu. Stačí vytvoriť nový modul pre daný zdroj. Pre každé rádio pri zozbieraní záznamu pre skladbu ale aj poslucháčov sme pridávali ešte atribúty:

- **raw\_valid** (boolean) – údaj, či prišli očakávané atribúty podľa dokumentácie rádia
- **recorded\_at** (datetime) – niektoré rádio neposielalo dátumový údaj záznamu, tak sme si ho doplnili
- **song\_session\_id** (string) – jedinečný identifikátor, ktorý spája skladbu s počtami zachytávaných poslucháčov

### *Pseudocode*

```
MODULE RadioXAdapter

  CONSTANT SONG_SOURCE_URL
  CONSTANT LISTENERS_SOURCE_URL (optional)

  FUNCTION get_current_song():
    response ← request(SONG_SOURCE_URL)
    raw_data ← parse(response)
    song ← normalize_song(raw_data)           // názov, interpret, čas, id...
    RETURN song

  FUNCTION get_current_listeners():
    response ← request_or_stream(LISTENERS_SOURCE_URL)
    raw_data ← parse(response)
    listeners ← normalize_listeners(raw_data)
    RETURN listeners

  FUNCTION flatten_song(song):
    RETURN {
      "radio": "RadioX",
      "title": song.title,
      "artist": song.artist,
      "timestamp": song.timestamp,
      ...
    }

  FUNCTION flatten_listener(listeners):
    RETURN {
      "radio": "RadioX",
      "listeners": listeners.count,
      "timestamp": listeners.timestamp,
      ...
    }
END MODULE
```

## 2.2. Modul ukladania dát

Táto časť systému zabezpečuje ukladanie dát do úložiska. Je oddelená od samotného zberu dát, čo zjednodušuje údržbu aplikácie a zvyšuje jej prehľadnosť. Počet poslucháčov pre danú skladbu zachytávame v krátkych časových intervaloch, čím získame viac údajov o poslucháčoch počas hrania jednej skladby.

Vďaka tomuto riešeniu je možné údaje neskôr využiť napríklad na analýzu, prehľady alebo vizualizáciu bez nutnosti opakovaného zberu. Naše zozbierané dáta boli skladované v úložisku datalake, konkrétne v Cloudflare R2 Object Storage.

***Pseudocode***

```

MODULE CollectorService

LIST adapters ← [
    RadioMelodyAdapter,
    RadioRockAdapter,
    RadioExpresAdapter,
    ...
]

FUNCTION run():
    FOR EACH adapter IN adapters DO
        START_THREAD collect_loop(adapter)    // paralelne pre každé rádio
    END FOR

    WAIT_FOREVER()

FUNCTION collect_loop(adapter):
    WHILE service_is_running DO
        TRY
            song ← adapter.get_current_song()
            listeners ← adapter.get_current_listeners()

            song_record ← adapter.flatten_song(song)
            listeners_record ← adapter.flatten_listener(listeners)

            save_to_storage(song_record)
            save_to_storage(listeners_record)

        CATCH error
            log_error(adapter.name, error)
        END TRY

        SLEEP(FIXED_INTERVAL_SECONDS)    // časovanie
    END WHILE
END MODULE

```

**2.3. Modul časovania a paralelného spracovania**

Služba pracuje v pravidelných časových intervaloch, ktoré zabezpečuje plánovací mechanizmus. Ten automaticky spúšťa zber dát bez potreby manuálneho zásahu a dohliada na to, aby sa údaje získavali konzistentne a v rovnakých časových rozostupoch. Vďaka tomu je zabezpečená plynulá prevádzka služby aj pri jej dlhodobom nasadení na serveri.

Zber dát je navrhnutý tak, aby bolo možné spracúvať viacero rozhlasových staníc súčasne. Každý zdroj dát je spracovávaný samostatne, čo umožňuje lepšie využitie dostupných systémových prostriedkov a výrazne skracuje čas potrebný na spracovanie jednej zberovej dávky. Tento prístup zároveň zvyšuje celkovú efektivitu služby a umožňuje jej jednoduché rozšírenie o ďalšie zdroje v budúcnosti.

***Pseudocode***

```

PROCEDURE add_new_radio(NewRadioAdapter):

    // 1) vytvor nový modul
    CREATE_FILE "adapters/radio_new.py"
    IMPLEMENT get_current_song()
    IMPLEMENT get_current_listeners()
    IMPLEMENT flatten_song()
    IMPLEMENT flatten_listener()

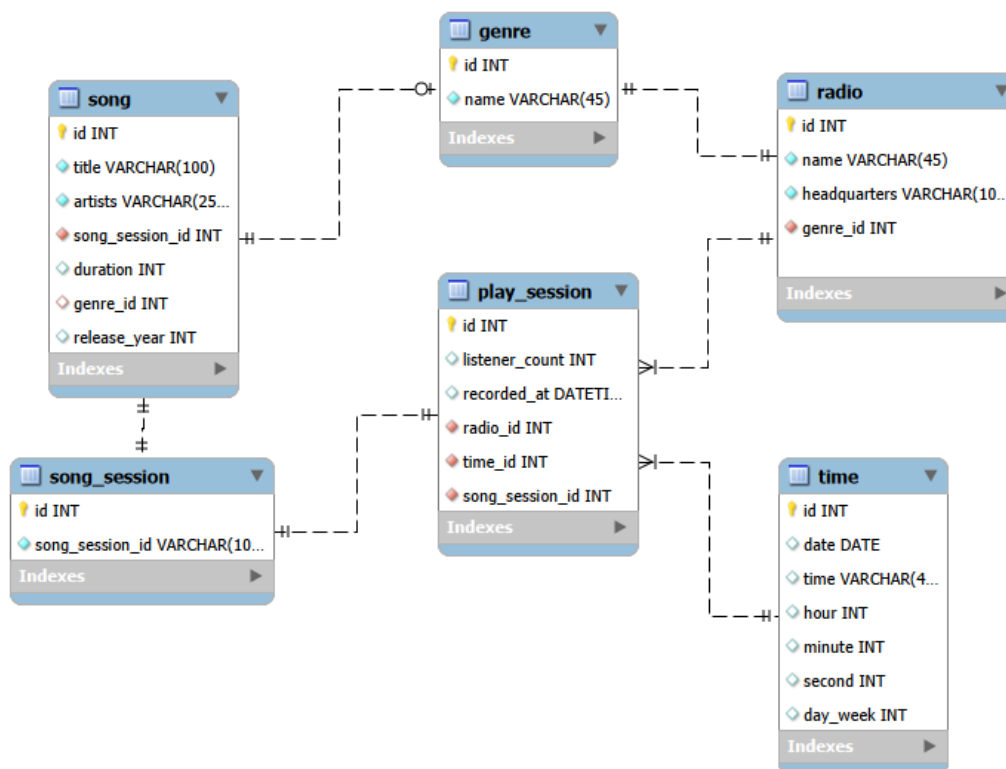
```

```
// 2) zaregistruj adaptér do zoznamu  
ADD NewRadioAdapter TO CollectorService.adapters  
  
// 3) hotovo - služba ho začne automaticky zbierať  
END PROCEDURE
```



### 3. ER diagram

Návrh databázy je relačný model postavený na jednoznačných primárnych kľúčoch a prepojeniach pomocou cudzích kľúčov. Štruktúra je normalizovaná tak, aby sa minimalizovala redundancia údajov a zachovala konzistencia dát. Vzťahy medzi entitami sú prevažne typu 1:N, čo umožňuje flexibilné a rozšíriteľné ukladanie záznamov. Model oddeľuje popisné údaje od faktických záznamov, čím podporuje efektívne analytické dotazy. Použitie samostatných identifikátorov zabezpečuje referenčnú integritu medzi tabuľkami. Návrh je vhodný na historizáciu údajov a prácu s časovým rozmerom. Databáza je pripravená na ďalšie rozšírenie bez zásadných zásahov do existujúcej schémy. Celková architektúra podporuje prehľadnosť, konzistentnosť a dlhodobú udržateľnosť dátového modelu. Návrh ER diagramu sme vypracovali v MySQL Workbench.



Obr. 1 – Finálna verzia ER diagramu

#### 3.1. Entita **play\_session**

Ukladá záznam o prehrávaní v určitom kontexte. Primárny kľúč je **id** typu INT a má atribút **listener\_count** typu INT (môže byť NULL). Obsahuje cudzie kľúče **song\_session\_id** (INT, NOT NULL) do **song\_session(id)**, **radio\_id** (INT, NOT NULL) do **radio(id)** a **time\_id** (INT, NOT NULL) do **time(id)**.

### 3.2. Entita **song\_session**

Reprezentuje jednu „session“ (identifikovateľný celok) pre skladby a prehrávania. Primárny kľúč je **id** typu INT (AUTO\_INCREMENT) a má atribút **song\_session\_id** typu VARCHAR(45), ktorý je unikátny (UNIQUE).

### 3.3. Entita **time**

Uchováva časovo-dátumové údaje. Primárny kľúč je **id** typu INT (AUTO\_INCREMENT) a ďalšie atribúty sú **hour** (INT, NOT NULL), **minute** (INT, NOT NULL), **second** (INT, môže byť NULL), **date** (DATE, NOT NULL) a **day\_week** (TINYINT, môže byť NULL).

### 3.4. Entita **song**

Eviduje skladby. Primárny kľúč je **id** typu INT (AUTO\_INCREMENT) a ďalšie atribúty sú **title** (VARCHAR(100)), **artists** (VARCHAR(255)) a **duration** (INT, môže byť NULL). Cudzíe kľúče sú **genre\_id** (INT, môže byť NULL) smerujúci do **genre(id)** a **song\_session\_id** (INT, NOT NULL) smerujúci do **song\_session(id)**.

### 3.5. Entita **genre**

Slúži na evidenciu žánrov. Primárny kľúč je **id** typu INT (AUTO\_INCREMENT) a ďalší atribút je **genre** typu VARCHAR(45).

### 3.6. Entita **radio**

Eviduje rádiá. Primárny kľúč je **id** typu INT (AUTO\_INCREMENT) a ďalšie atribúty sú **name** (VARCHAR(45)) a **headquarters** (VARCHAR(45)). Má cudzí kľúč **genre\_id** (INT, NOT NULL), ktorý odkazuje na **genre(id)**.

### 3.7. Vzťahy medzi entitami

ER diagram znázorňuje prepojenia medzi entitami v databáze, ktoré sú realizované prostredníctvom cudzích kľúčov. Entita **song** má priamy vzťah s entitou **genre** cez atribút **genre\_id**, čo umožňuje priradiť skladbe žánr (príp. ho ponechať nešpecifikovaný). Podobne je entita **radio** prepojená s entitou **genre** cez **genre\_id**, čím sa určuje žánrové zameranie konkrétneho rádia. Entita **song** je zároveň naviazaná na entitu **song\_session** cez **song\_session\_id**, vďaka čomu je možné združovať skladby do jednej relácie (napr. playlist/blok). Kľúčovým prepojením je entita **play\_session**, ktorá spája **song\_session\_id**, **radio\_id** a **time\_id**, a tým jednoznačne určuje, čo (relácia skladieb), kde (rádio) a kedy (čas) sa prehrávalo, pričom môže obsahovať aj údaj

**listener\_count.** Tieto väzby poskytujú konzistentný a ucelený pohľad na prehrávanie v rádiu a umožňujú efektívne analyzovať vysielanie naprieč časom, rádiami, reláciami a žánrami. Vďaka referenčnej integrite sa zároveň minimalizujú nekonzistentné dáta a zjednodušuje sa práca s dotazmi pri štatistikách a reportovaní.

## 4. Údaje o dátach

Údaje o dátach v .json súboroch:

- **name** (str) – (v radio) názov rádia
- **headquarters** (str) – miesto, kde má rádio centrálu
- **genre** (str) – názov hudobného žánru
- **title** (str) – názov skladby
- **artists** (str) – meno alebo mená interpretov skladby
- **duration** (int) – dĺžka skladby v milisekundách
- **release\_year** (int) – rok vydania skladby
- **song\_session\_id** (str) – textový identifikátor relácie (napr. externé alebo logické ID session)
- **listener\_count** (int) – počet poslucháčov počas prehrávania (ak je známy)
- **hour** (int) – časový údaj hodiny
- **minute** (int) – časový údaj minúty
- **second** (int) – časový údaj sekundy (voliteľná)
- **date** – dátum prehrávania
- **day\_week** (int) – deň v týždni vyjadrený číselnou hodnotou (napr. 1–7)

## 5. Spracovanie dát

Naše dáta boli získavané v dvojhodinových intervalových zberoch a uložené do JSON súborov. Na konci zberu sme všetky dáta stiahli, spojili do jedného JSON súboru, upravili štruktúru dát a dáta očistili od nepotrebných atribútov a následne obohatili dáta o údaje ako **genre**, **duration** a **release\_year** cez iné API.

### 5.1. Modul transformácie dát

Tento modul prechádza lokálnu adresárovú štruktúru s bronz<sup>2</sup> dátami a zbiera z nej záznamy o prehratých skladbách pre jednotlivé rádiá. Pre každý rádio-priečinok hľadá podadresár song/, potom vnútorné priečinky podľa dátumu a v nich všetky JSON súbory. Každý JSON načíta a spracuje tak, že podporuje formát, kde je buď zoznam záznamov, alebo len jeden objekt. Z každého záznamu sa snaží jednotne vytiahnuť názov skladby, autorov, čas a dátum, pričom toleruje viac rôznych názvov kľúčov (napr. title/song/musicTitle, artists/musicAuthor/artist, start\_time/time atď.). Podobný princíp je aplikovaný pre poslucháčov.

Popritom robí normalizáciu - čas prepisuje do formátu HH:MM:SS a dátum do DD.MM.YYYY. Na konci všetky normalizované záznamy zlúči do jedného zoznamu a uloží ich ako jeden výsledný silver\_merged.json.

### 5.2. Modul obohatenia dát

Modul načíta všetky záznamy cez viacero API (MusicBrainz, Last.fm, iTunes, Spotify a ListenBrainz) doplní metadáta (**genre**, **duration** a **release\_year**) skladieb a uloží výsledok do nového JSON súboru, pričom podporuje prerušovaný beh (checkpointy<sup>3</sup>) a nasadený bol ako batch skript na VPS Hetzner Cloud.

Hierarchia obohatenia je striktné postupná, vždy sa najprv využije skorší zdroj a neskorší sa volá len na doplnenie chýbajúcich polí:

1. MusicBrainz – prvý pokus o získanie **duration**, **genre**, **release\_year**
2. Last.fm – doplní chýbajúce polia, ak nie sú všetky tri po MusicBrainz
3. iTunes – opäť len doplní hodnoty, ktoré sú stále None
4. Spotify – doplní hlavne **duration** a **release\_year**, ak ešte chýbajú
5. ListenBrainz – posledný zdroj, po ňom sa výsledné metadáta zapisujú do záznamu bez ďalšieho fallbacku

---

<sup>2</sup> Nespracované dáta

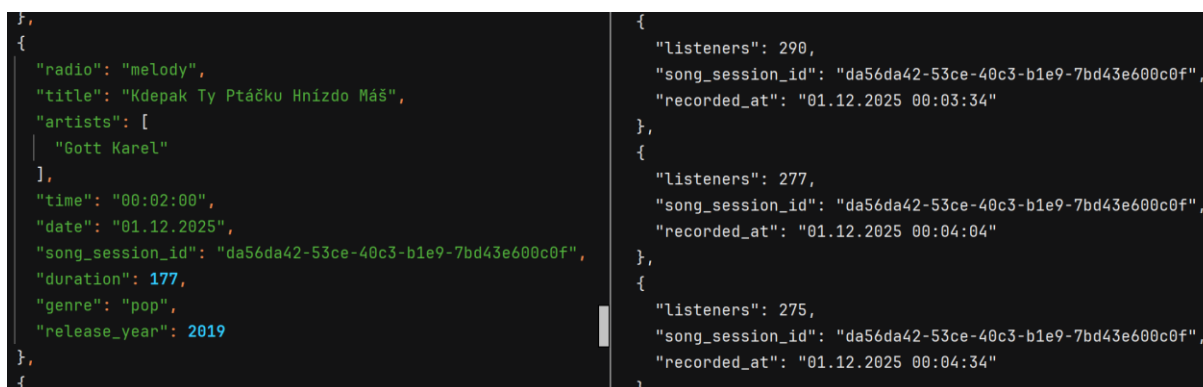
<sup>3</sup> Program priebežne ukladá v dávkach obohatené záznamy do JSON súboru

### 5.3. Modul prevodu duration na rovnaké jednotky (sekundy)

Po skončení obohacovania dát sme načítali obsah posledného výsledného JSONu a previedli hodnoty **duration** na sekundy (niektoré API posielalo hodnoty v ms).

### 5.4. Modul premapovania genre

Po skončení zmeny jednotiek **duration** sme premapovali **genre**, aby sme sa vyhli väčšine podtypom ako napríklad pop soul, pop in Spanish na pop; heavy metal, death metal na metal...



```

{
  "radio": "melody",
  "title": "Kdepak Ty Ptáčku Hnízdo Máš",
  "artists": [
    "Gott Karel"
  ],
  "time": "00:02:00",
  "date": "01.12.2025",
  "song_session_id": "da56da42-53ce-40c3-b1e9-7bd43e600c0f",
  "duration": 177,
  "genre": "pop",
  "release_year": 2019
},
{
  "listeners": 290,
  "song_session_id": "da56da42-53ce-40c3-b1e9-7bd43e600c0f",
  "recorded_at": "01.12.2025 00:03:34"
},
{
  "listeners": 277,
  "song_session_id": "da56da42-53ce-40c3-b1e9-7bd43e600c0f",
  "recorded_at": "01.12.2025 00:04:04"
},
{
  "listeners": 275,
  "song_session_id": "da56da42-53ce-40c3-b1e9-7bd43e600c0f",
  "recorded_at": "01.12.2025 00:04:34"
}

```

Obr. 2 - Výsledný JSON po transformácii dát (song / listeners)

### 5.5. Modul na naplnenie databázy

Databáza sa naplnila cez použitie PHP skriptov. Nahrávanie bolo rozdelené na 2 časti: na naplnenie záznamov pre všetko okrem počúvanosti a napokon naplnenie faktovej tabuľky s počúvanosťou.

#### 5.5.1. Záznamy všetkého okrem listener\_count

Zobrali sme náš 1 JSON súbor, ktorý obsahuje všetky zozbierané záznamy zo všetkých 6 rádií a následne sme ten použili ako argument pre PHP skript **import\_rest.php**, ktorý sa prihlási na localhost MySQL na databázu “radioDB”, nad ktorou vykoná SELECT a INSERT príkazy na doplnenie záznamov do daných tabuliek v databáze. Skript bol spustený v prehliadači cez link “http://localhost/import\_rest.php”.

#### 5.5.2. Záznamy listener\_count

Rovnaký postup sme použili aj pre doplnenie počúvanosti. Všetky záznamy počúvanosti sme “merge-li” do jedného súboru, ktorý sme následne nahrali do

databázy prostr. PHP skriptu **import\_listeners.php**. Skript bol spustený v prehliadači cez link “[http://localhost/import\\_listeners.php](http://localhost/import_listeners.php)”.

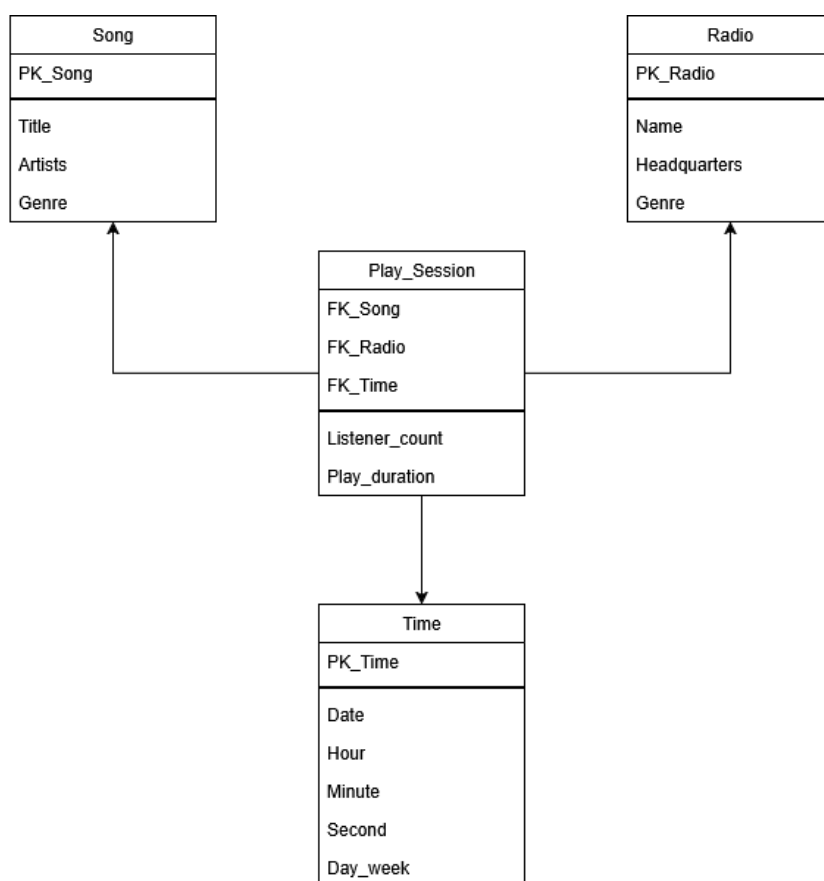
# Dimenzionálne modelovanie

## 6.1. Informačný balíček

Song	Radio	Time
Title	Name	Date
List of artists	Headquarters	Hour
Genre	Genre	Minute
		Second
		Day of Week
<b>Facts:</b> listener count, play duration		
<b>What happened?</b> A song was played on a radio station		

Obr. 3 – Informačný balíček

## 6.2. STAR schéma



Obr. 4 – Definovaná STAR schéma



## 6. ETL procesy

V tejto fáze projektu bolo potrebné návrhy STAR schémy a informačného balíka pretaviť do reality a to s využitím dát, ktoré získal náš tím.

Z nášho datalake sme stiahli všetky záznamy. Modul transformácie dát všetky zaznamenané dáta z rádií prejde a zjednotí formáty a štruktúru atribútov.

Modul obohatenia dát pomocou API pridáva atributy **genre**, **duration** a **release\_year**.

Modul na prevod jednotiek **duration** z milisekúnd na sekundy.

Modul premapovania **genre** – prepísanie podžánrov na všeobecnejšie žánre.

Modul na naplnenie databázy v dvoch častiach naplní databázu údajmi. Najprv cez PHP skript sa naplní všetkými údajmi o skladbách a v druhej časti sa naplní aj listeners.

## 7. Použité služby a technológie

- Python – jazyk použitý v našich kódach
- PHP – jazyk použitý pri plnení databázy údajmi
- Fly.io – cloudová platforma využitá na poskytovanie údajov nášho rádia
- DigitalOcean – cloudová platforma využitá na zbieranie údajov z rádií
- Hetzner Cloud – cloudová platforma, na ktorej bežal program na obohacovanie našich dát (cca 12h)
- API MusicBrainz, Last.fm, iTunes, Spotify a ListenBrainz – pridávali k našim dátam metadáta
- MySQL Workbench – v ňom sme vytvorili ER diagram
- Cloudflare - objektové úložisko navrhnuté pre ukladanie veľkého objemu neštruktúrovaných dát (Cloudflare R2 Object Storage)
- XAMPP – lokálny server, na ktorom sme mali našu databázu

## 8. Ukážky SELECTov

### 8.1. Krok 1

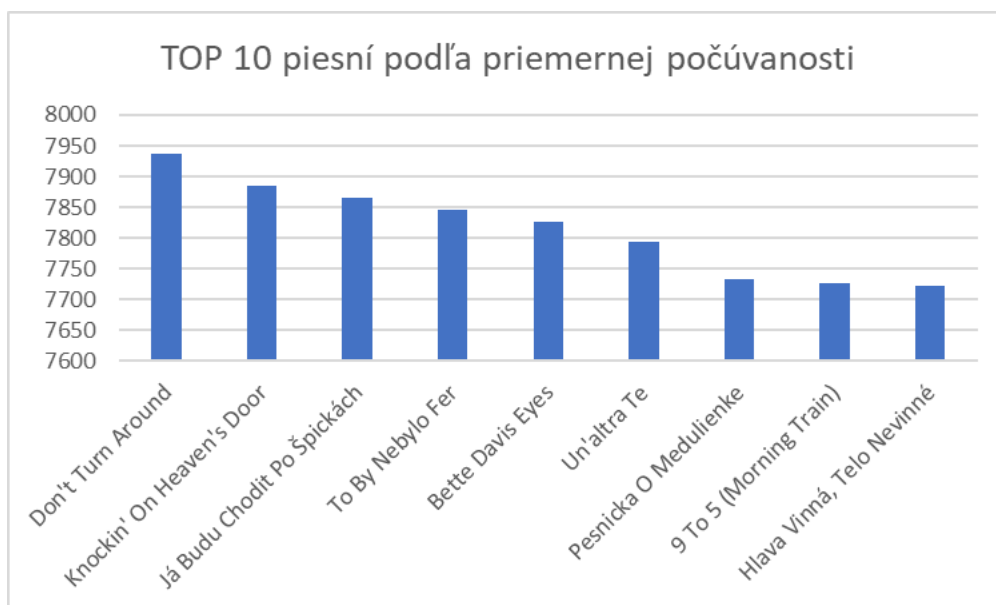
TOP 10 piesní podľa priemernej počúvanosti

```
SELECT s.title,
       s.artists,
       Round(Avg(p.listener_count), 1) AS avg_listeners
FROM   play_session p
JOIN   song s
ON     p.song_session_id = s.song_session_id
GROUP BY s.title,
         s.artists
HAVING Count(*) >= 5
ORDER BY avg_listeners DESC limit 10;
```

#### 8.1.1. Výsledok SQL query

title	artists	avg_listeners ▾ 1
Don't Turn Around	Ace Of Base	7935.8
Knockin' On Heaven's Door	Dylan Bob	7885.8
Já Budu Chodit Po Špičkách	Flamengo Novák Petr	7864.7
To By Nebylo Fer	Hana Zagorova	7846.0
Bette Davis Eyes	Kim Carnes	7826.0
Un'altra Te	Ramazotti Eros	7794.4
Pesnička O Medulienke	Gladiator, Pavol Hammel & Petr Rajchert	7733.0
9 To 5 (Morning Train)	Sheena Easton	7725.8
Hlava Vinná, Telo Nevinné	Peha	7722.2
Didn't We Almost Have It All	Houston Whitney	7718.5

#### 8.1.2. Graf



## 8.2. Krok 2

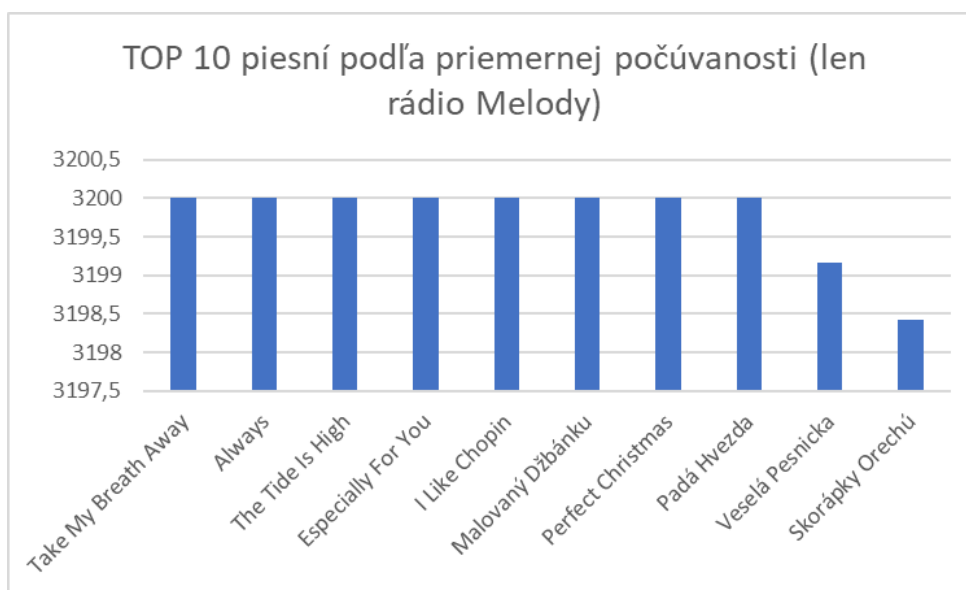
TOP 10 piesní podľa priemernej počúvanosti (rádio Melody)

```
SELECT    s.title                AS song_title,
          Avg(ps.listener_count) AS avg_listeners,
          Count(ps.id)           AS measurements
FROM      play_session ps
JOIN      radio r
ON        ps.radio_id = r.id
JOIN      song s
ON        ps.song_session_id = s.song_session_id
WHERE     r.NAME = 'melody'
GROUP BY  s.song_session_id,
          s.title
ORDER BY  avg_listeners DESC limit 10;
```

### 8.2.1. Výsledok SQL query

song_title	avg_listeners	measurements
Malovaný Džbáňku	3200.0000	6
Perfect Christmas	3200.0000	5
Padá Hvězda	3200.0000	5
Take My Breath Away	3200.0000	11
Always	3200.0000	8
The Tide Is High	3200.0000	9
I Like Chopin	3200.0000	9
Especially For You	3200.0000	10
Veselá Pesnička	3199.1667	7
Skoráčky Orechů	3198.4286	8

### 8.2.2. Graf



## 8.3. Krok 3

Priemerná počúvanosť za každú hodinu v dni

```
SELECT hour(p.recorded_at) AS hour, round(avg(p.listener_count), 1) AS
avg_listeners FROM play_session p GROUP BY hour ORDER BY hour;
```

### 8.3.1. Výsledok SQL query

hour	avg_listeners
0	383.1
1	384.2
2	384.5
3	380.3
4	370.3
5	422.5
6	609.5
7	869.1
8	928.4
9	921.2
10	754.8
11	666.7
12	711.1
13	760.2
14	760.2
15	698.9
16	773.1
17	901.1
18	1048.1
19	948.1
20	851.1
21	773.1
22	666.7
23	511.1
24	433.3

### 8.3.2. Graf



## 8.4. Krok 4

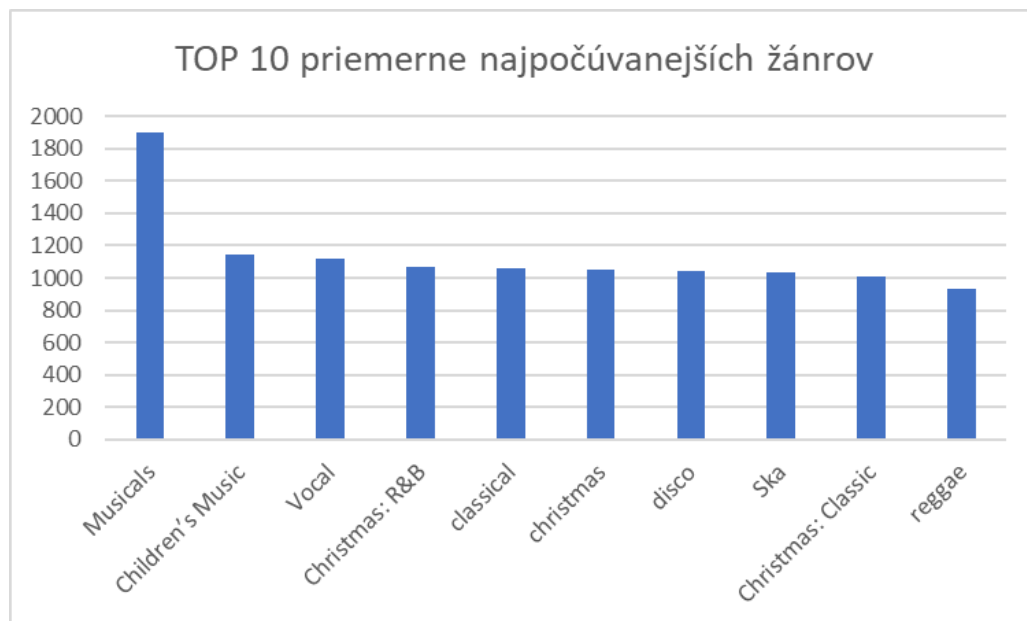
TOP 10 priemerne najpočúvanejších žánrov

```
SELECT g.genre,
       Round(Avg(p.listener_count), 1) AS avg_listeners
FROM   play_session p
JOIN   song s
ON     p.song_session_id = s.song_session_id
JOIN   genre g
ON     s.genre_id = g.id
GROUP BY g.genre
ORDER BY avg_listeners DESC limit 10;
```

### 8.4.1. Výsledok SQL query

genre	avg_listeners ▼ 1
Musicals	1902.5
Children's Music	1145.1
Vocal	1118.3
Christmas: R&B	1064.2
classical	1055.6
christmas	1052.0
disco	1044.8
Ska	1034.2
Christmas: Classic	1010.8
reggae	930.4

### 8.4.2. Graf



## 9. DRILL DOWN piesne pre rádio Melody

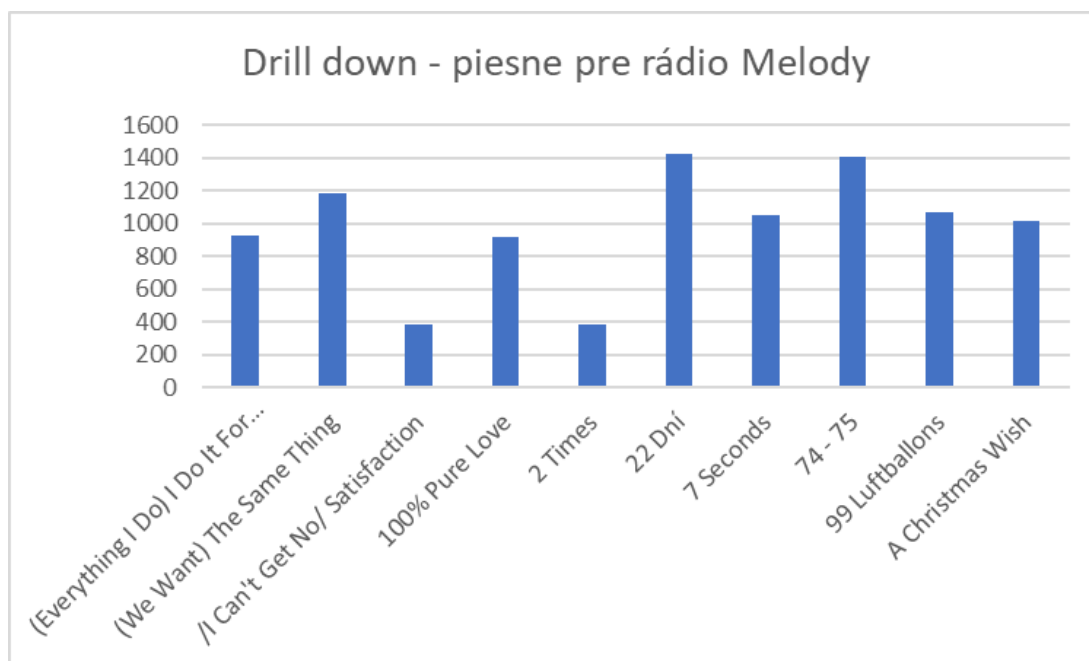
### 9.1. Krok 1

```
SELECT s.title AS category,
       Avg(ps.listener_count) AS value
FROM play_session ps
JOIN song s
ON ps.song_session_id = s.song_session_id
JOIN radio r
ON ps.radio_id = r.id
WHERE r.NAME = 'melody'
GROUP BY s.title limit 10;
```

#### 9.1.1. Výsledok SQL query

category	value
(Everything I Do) I Do It For You	926.2024
(We Want) The Same Thing	1182.4559
/I Can't Get No/ Satisfaction	384.8333
100% Pure Love	916.0909
2 Times	383.8571
22 Dní	1423.5610
7 Seconds	1049.6739
74 - 75	1408.4746
99 Luftballons	1066.0000
A Christmas Wish	1016.2875

#### 9.1.2. Graf



## 10. ROLL-UP priemerná počúvanosť podľa rádia

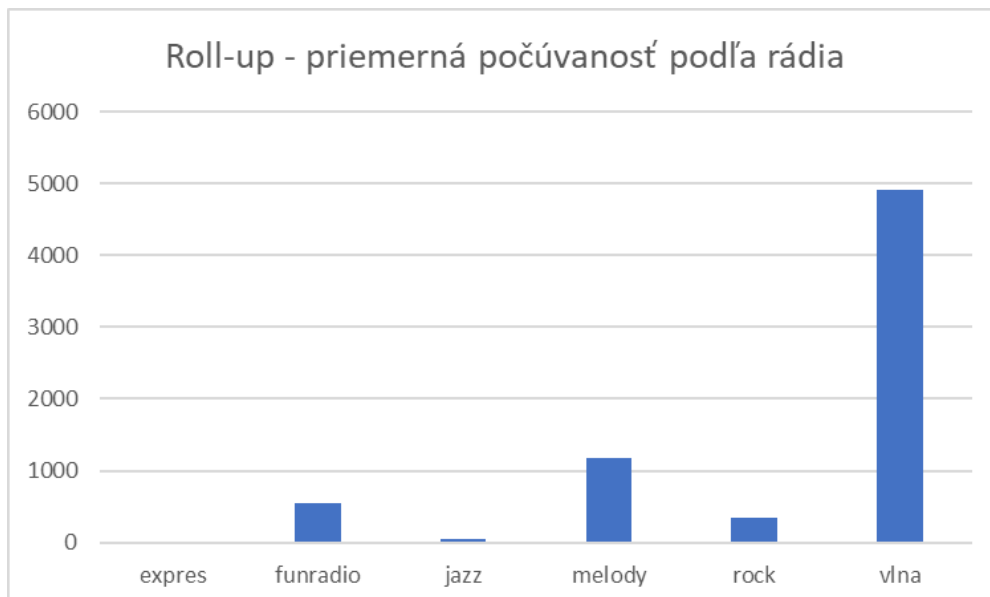
### 10.1. Krok 1

```
SELECT r.NAME AS category, avg(ps.listener_count) AS value FROM  
play_session ps JOIN radio r ON ps.radio_id = r.id GROUP BY r.NAME;
```

#### 10.1.1. Výsledok SQL query

category	value
expres	NULL
funradio	555.5588
jazz	47.0198
melody	1172.8570
rock	338.2157
vlna	4917.7883

#### 10.1.2. Graf





## 11. Záver

Sme si istý, že naše riešenie nie je optimálne, no vieme tiež skonštatovať, že sme úspešne implementovali dátové jazero spolu s databázou podľa potrieb FM rádio staníc. Podarilo sa nám ukladať dáta rozličných rádio staníc v daných intervaloch, následné obohatenie a uloženie záznamov do nami navrhnutej štruktúry.

Z hľadiska toho, čo by sme zlepšili, by sme určite sa pokúsili vyhodit' všetky NULL záznamy, alebo záznamy, ktoré boli skoro úplne prázdne (2 z troch polí mali NULL).