Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÁ PRÁCE

2024/2025

Adam Trávníček

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Ročníková práce

Předmět: Programování

Téma: Aplikace s informacemi o umělcích

Autor: Adam Trávníček

Třída: IV. E

Školní rok: 2024/2025

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Prohlášení

Prohlašuji, že jsem na této práci pracoval samostatně, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V .....dne ....

Adam Trávníček

Anotace

Tato dokumentace popisuje ročníkový projekt na téma: aplikace s informacemi o umělcích. Cílem tohoto projektu je vytvořit aplikaci, která bude fungovat jako pomocník při zjišťování si informacích o umělcích, skladbách a žánrech. Přímo v aplikace si bude možné označit své oblíbené umělce, skladby i žánry a aplikace bude doporučovat jiné umělce stejného žánru, aby uživatel mohl poznat nové umělce a rozšiřovat si tak svůj hudební zážitek. Uživatelé si mohou vytvořit kolekci oblíbených autorů. A na základě těchto kolekcí bude ostatním uživatelům doporučovat podobné autory.

Klíčová slova

umělec, uživatel, databáze, API, token

Abstract (English)

This documentation describes a year-long project on the topic: application with information about artists. The goal of this project is to create an application, which will function like an assistant in finding information about artists, songs and genres. It will be possible to mark your favorite artists, songs and genres directly in the application and the application will recommend other artists of the same genre so that the user can get to know new artists and expand their musical experience. Users can create a collection of favorite authors. And based on these collections it will recommend similar authors to other users.

Keywords

artist, user, database, API, token

Obsah

1. Úvod 2
   - 1.1. Zadání 2
   - 1.2. Důvod výběru tématu 2
2. Použité technologie 3
3. Struktura projektu 4
   - 3.1. Backend 4
     - 3.1.1. API routes 4
     - 3.1.2. Komunikace s databází 4
     - 3.1.3. Drizzle ORM 4
   - 3.2. Frontend 5
     - 3.2.1. Struktura souborů a komponentů 5
     - 3.2.2. Správa stavu, interakce a stylování 5
4. Klíčové komponenty 6
   - 4.1. Nejdůležitější části kódu (adresář lib/) 6
     - 4.1.1. db.js 6
     - 4.1.2. spotify.js 7
     - 4.1.3. userSchema.js 8
   - 4.2. Routes a pages 9
     - 4.2.1. Routes 9
     - 4.2.2. Pages 9
   - 4.3. Adresář components/ 10
     - 4.3.1. artist-selection/ 10
     - 4.3.2. ui/ 10
5. Závěr 11
6. Použité zdroje 12

Seznam obrázků 13

1. Úvod

Hudba je jeden z nejčastějších způsobů zábavy, který naše společnost používá. Nezáleží na žádném názoru, postoji, etniku, národnosti či politickému smýšlení, všichni ji poslouchají. Hudba dokáže velice dobře ovlivňovat emoce a tak ji mnoho lidí vyhledává, aby toho využili na potlačení nebo zesílení svých momentálních pocitů. Např. když je člověk naštvaný, tak si pustí hudbu, která evokuje naštvání, aby se tohoto pocitu zbavil. Ovšem není jednoduché hledat novou hudbu, která je podobná hudbě, kterou již posloucháte. Tato aplikace právě tomuto problému pomáhá.

1.1. Zadání

Cílem tohoto projektu je vytvořit aplikaci, která bude fungovat jako pomocník při zjišťování si informacích o umělcích, skladbách a žánrech. Přímo v aplikace si bude možné označit své oblíbené umělce, skladby i žánry a aplikace bude doporučovat jiné umělce stejného žánru, aby uživatel mohl poznat nové umělce a rozšiřovat si tak svůj hudební zážitek. Uživatelé si mohou vytvořit kolekci oblíbených autorů. A bude ostatním na základě těchto kolekcí bude uživatelům doporučovat podobné autory. Při přihlášení se do této aplikace bude dotazník s tím, koho uživatel nejradši poslouchá.

1.2. Důvod výběru tématu

Toto téma jsem si vybral, jelikož mě velice zajímá hudba a sám narážím na problém vyhledávání nových umělců. Také jsem si chtěl rozšířit svou znalost jazyku Javascript a vyzkoušet nový framework Next.js.

2. Použité technologie

K vytvoření této aplikace jsem použil tento programovací jazyk, framework a tyto knihovny.

1. Javascript[1]- programovací jazyk, vytvoření serverové logiky pro práci s databázemi a upravování dat o uživatelích.

2. Next.js[2] - framework, který je postavený na React.js a pomáhá mi integrovat backend a frontend do jednoho projektu. Slouží k vytvoření dynamické stránky a API routes pro komunikaci s databází.

3. Drizzle ORM[5] - knihovna pro JavaScript, kterou jsem využil pro interakci s databází, strukturu databáze a správě oblíbených umělců.

4. SQLite - použitá databázová knihovna

5. NextAuth.js[6] - knihovna pro autentizaci, která podporuje různé metody přihlašování (OAuth, e-mail) ale i vlastní přihlašovací údaje, což jsem k vytvoření této aplikace využil

3. Struktura projektu

Struktura projektu je udělána velice přehledně a dá se jednoduše orientovat mezi částmi projektu.

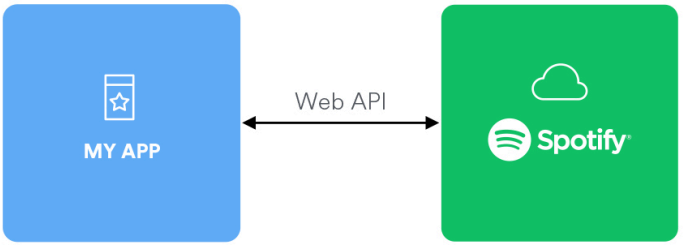
3.1. Backend

3.1.1. API routes

API trasy jsou ve složce *app/api/* a definují serverovou logiku. Každý soubor odpovídá určité trasám, které se týkají určitého subjektu. Např. *app/api/artist/route.js* obsahuje HTTP požadavky týkající se umělců.

3.1.2. Komunikace s databází

Ve složce *lib/* najdeme soubory, které komunikují s databází a Spotify API[4]. Nalezneme zde soubory *db.js*, *spotify.js*, *userSchema.js*, *userSeed.js* a *utils.js*. Podrobně se podíváme na některé z těchto tříd v části Nejdůležitější části kódu (adresář lib/).



Obrázek 1 - Spotify API

3.1.3. Drizzle ORM[5]

Ve složce *drizzle/* se nachází soubory pro správu, konfiguraci a strukturu databáze. Drizzle je ORM (Object-Relational Mapping), což umožňuje efektivní práci s databází a i pro vyhledávání či aktualizace dat.

Obrázek 2 - Drizzle ORM

3.2. Frontend

Frontend aplikace je postaven na frameworku **Next.js**[2] s podporou Reactu.js a používá Tailwind CSS[8].

3.2.1. Struktura souborů a komponenty

Frontend celé aplikace je uspořádán v adresáři *app/*, kde určité soubory odpovídají specifickým stránkám. Příkladem může být soubor login/page.jsx, který obsahuje logiku pro přihlašovací stránku. Komponenty pro uživatelské rozhraní jsou v adresáři *components/* a patří tam několik souborů, kde každý soubor má svoji danou funkci a obsahuje komponenty např. typu *button.jsx*, *card.jsx, dialog.jsx* a další.

3.2.2. Správa stavu, interakce a stylování

Aplikace využívá React hooky[3], ty umožňují připojit logiku nebo stav do komponentů bez potřeby vytváření třídy (například `useState`), pro správu přihlašovacích formulářů a chybových hlášení. Pro vizuály je použit Tailwind CSS a celkový vzhled je responzivní.

4. Klíčové komponenty

4.1. Nejdůležitější části kódu (adresář lib/)

4.1.1. db.js

Databáze aplikace je spravována pomocí **SQLite**, kde binární soubor *database.sqlite* uchovává všechna data. Připojení k databázi probíhá prostřednictvím klienta vytvořeného pomocí metody *createClient*, který běží na pozadí. ORM klient *drizzle*, který umí pracovat s datovými schématy jako jsou *users* a *usersFavouriteArtists*. Funkce *fetchUserFavouriteArtists* slouží k načítání seznamu oblíbených umělců pro konkrétního uživatele a funkce *saveUserFavouriteArtists* umožňuje uživatelům přidávat více umělců do seznamu oblíbených.

```
1  import "dotenv/config";
2  import { drizzle } from "drizzle-orm/libsql";
3  import { createClient } from "@libsql/client";
4  import { users, userFavoriteArtists } from "./userSchema.js";
5  import { eq } from "drizzle-orm";
6
7  if (!process.env.DB_FILE_NAME) {
8    throw new Error("Missing DB_FILE_NAME in environment variables");
9  }
10
11 // Initialize the database client
12 const client = createClient({
13   url: process.env.DB_FILE_NAME,
14 });
15
16 export const db = drizzle(client, { schema: { users, userFavoriteArtists } });
17
18 export async function fetchUserFavoriteArtists(userId) {
19   return await db
20     .select({
21       userId: userFavoriteArtists.userId,
22       artistId: userFavoriteArtists.artistId,
23       createdAt: userFavoriteArtists.createdAt,
24     })
25     .from(userFavoriteArtists)
26     .where(eq(userFavoriteArtists.userId, userId));
27 }
```

Ukázka 1 - db.js, createClient

4.1.2. spotify.js

Tento soubor propojuje aplikaci se **Spotify API**[4], která zajišťuje požadavky týkající se hudby a určitých umělců. K přístupu je požadován "access token", který se generuje a ukládá pro opakované použití.

V tomto souboru jsou také důležité funkce, které jsou rozděleny v několika metodách, na jejichž začátku se zjistí pokaždé, zda-li je validní token:

- *fetchArtistToken* - získává a spravuje přístupový token (access token)
- *fetchRandomArtistsByArtistGenre* - vytváří doporučení umělců na základě jejich daného žánru
- *fetchArtistByGenre* - vyhledává umělce podle žánru
- *fetchArtistTopTracks* - získává nejpopulárnější skladby umělce
- *searchArtists* - vyhledává umělce na základě uživatelského dotazu ("query")

```
export async function fetchRandomArtistsByArtistGenres(artistIds) {
  try {
    // Default genres if none are found
    const fallbackGenres = ["pop", "edm", "techno"];

    let allGenres = [];

    // Fetch genres sequentially for each artist
    for (const artistId of artistIds) {
      try {
        const artistGenres = await fetchArtistGenres(artistId);
        if (artistGenres && artistGenres.length > 0) {
          allGenres = [...allGenres, ...artistGenres];
        }
      } catch (error) {
        console.error(`Error fetching genres for artist ${artistId}:`, error);
        // Continue with next artist if one fails
      }
    }

    // If no genres were found for any artists, use fallback
    const genresToUse = allGenres.length > 0 ? allGenres : fallbackGenres;

    // Select a random genre from collected genres
    const randomGenre =
      genresToUse[Math.floor(Math.random() * genresToUse.length)];

    // Fetch artists by the selected genre
    const artists = await fetchArtistsByGenre(randomGenre);

    return artists;
  } catch (error) {
    console.error("Error in fetchRandomArtistsByArtistGenres:", error);
    throw error;
  }
}
```

Ukázka 2 - fetchRandomArtistsByRandomGenre

4.1.3. userSchema.js

Soubor *userSchema.js* se používá k definování databázové struktury uživatelů a jejich umělců pomocí **Drizzle ORM**[5]. Obsahuje schéma tabulek "users" a "userFavouriteArtists", která určují, jak budou data k databázi zorganizována.

Tabulka "users" obsahuje určitá data, mezi která patří: *id*, které slouží jako primární klíč a díky *{ autoIncrement: true }* se zajišťuje jedinečnost id. Dále *name, email* a *password*, kde ani jedna nesmí být prázdná (null) a email musí být unikátní. A pak už jen *createdAt*, což zapisuje čas vytvoření záznamu.

A tabulka "userFavouriteArtists" slouží k přiřazení oblíbených umělců konkrétním uživatelům přes *user_id*, což je cizí klíč odkazující na tabulku "users".

```
1  import { sqliteTable, text, integer } from "drizzle-orm/sqlite-core";
2  import { sql } from "drizzle-orm";
3
4  export const users = sqliteTable("users", {
5    id: integer("id").primaryKey({ autoIncrement: true }),
6    name: text("name").notNull(),
7    email: text("email").unique().notNull(),
8    password: text("password").notNull(),
9    createdAt: text("created_at").default(sql`CURRENT_TIMESTAMP`),
10 });
11
12 export const userFavoriteArtists = sqliteTable("user_favorite_artists", {
13   userId: integer("user_id")
14     .references(() => users.id)
15     .notNull(),
16   artistId: integer("artist_id").notNull(),
17   createdAt: text("created_at").default(sql`CURRENT_TIMESTAMP`),
18 });
19
```

Ukázka 3 - userSchema.js, users, userFavouriteArtists

4.2. Routes a pages

4.2.1. Routes

"Routes" neboli trasy jsou esenciální pro tento projekt, jelikož se používají na zpracovávání **Spotify API**[4] a uživatelských požadavků. Příkladem může být *route.js* v adresáři *app/api/auth/[..nextauth]/*, který je k autentizaci pomocí **NextAuth.js**[6] s poskytovatelem umožňuje přihlášení uživatelů pomocí e-mailu a hesla. Tento princip používá JWT[7] tokeny, které se rychle přenášejí mezi databází a serverem a, co je nejdůležitější, obsahují všechny informace o subjektu, aby se zabránilo opakovanému dotazování databáze.

4.2.2. Pages

Každý soubor v rámci *app/* představuje samostatnou stránku, která odpovídá určité trase. Těchto souborů je opravdu mnoho, ale zmíním zde např. *page.js* v adresáři *register/, který* slouží uživatelům na vytvoření nového účtu. Hlavní funkcí je *handleRegister*, která nejdřív zjistí, zda-li jsou splněny požadavky na heslo (jestli má nad šest znaků a shodné hesla v obou polích) a pak pošle POST požadavek (data jsou odesílána skrytě, nejsou viditelná v URL) na "API Endpoint", což zde je */api/auth/register*. "API Endpoint" je adresa, která zpřístupňuje aplikaci zdroje a funkce v API.

4.3. Adresář components/

V tomto adresáři jsou různé React komponenty, které jsou rozděleny do různých podsouborů. Každý komponent zodpovídá za určitou část aplikace.

4.3.1. artist-selection/

V podadresáři *artist-selection/* jsou tři komponenta: *ArtistCard.jsx*, který obsahuje informace o jednotlivých umělcích ve formě karet. Umožňuje zobrazit si detaily o umělci a vybrat či odebrat umělce; *ArtistSelectionClient.jsx*, který umožňuje vybrat až tři umělce a obsahuje funkci pro vyhledání informací umělců; a *SearchResults.jsx*, který zobrazuje nalezené výsledky vyhledávání a když nenajde žádný výsledek, tak zobrazí skeletonové obrazce či zprávu o nenalezení umělce.

4.3.2. ui/

V podadresáři *ui/* je několik komponent, které slouží jako prvky UI(User Interface) neboli uživatelského rozhraní. Komponenty v souborech *alert.jsx, avatar.jsx, button.jsx* a *card.jsx* obsahují prvky, jako jsou upozornění, tlačítka a karty. Soubory *input.jsx* a *label.jsx* tvoří pole pro zadávání textu a popisku ve formulářích. Soubory *dialog.jsx* a *toast.jsx*, které ukazují malé okna informující důležitých změnách. Soubor *progress.jsx* je ukazatel pokroku, zatímco *skeleton.jsx* je náhrada, když se nic nenajde. A nakonec soubor *separator.jsx* vytváří oddělovače a soubor *connect.jsx*, což je komponent pro vyhledávání a zadávání příkazů.

5. Závěr

Tato aplikace je určena každému, kdo rád objevuje novou hudbu a nové umělce. Aplikaci se mi povedlo zprovoznit do podoby popsané v zadání a naučil jsem se mnoho nových dovedností, mezi které patří práce s API, časová efektivita a poznání nových zajímavých knihoven. V aplikaci by se v budoucnu mohl zlepšit celkový vzhled a přehledu v aplikaci samotné. Tato práce je velice pravděpodobně mou poslední prací z programování, kterou kdy udělám, jelikož se programování nadále věnovat nechci. Ale musím říct, že mi dělání této práce dalo zase o něco větší přehled, který se vždy hodí mít.

6. Použité zdroje

[1] - Javascript Documentation - https://developer.mozilla.org/en-US/docs/Web/JavaScript

[2] - Next.js Documentation - https://nextjs.org/

[3] - React Hooks Fundamentals for Beginners, URL: https://www.freecodecamp.org/news/react-hooks-fundamentals/

[4] - Spotify Web API, URL: https://developer.spotify.com/documentation/web-api/reference/search

[5] - Drizzle ORM Documentation, URL:https://orm.drizzle.team/docs/overview

[6] - NextAuth.js Documentation, URL: https://next-auth.js.org/getting-started/introduction

[7] - JSON Web Tokens, URL:https://auth0.com/docs/secure/tokens/json-web-tokens

[8] - Tailwind CSS Documentation, URL:https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://tailwindcss.com/docs&ved=2ahUKEwjgzvW19bKMAxW3BxAIHZ0EMYMQmuEJegQIDBAB&usg=AOvVaw2BxsHvnrh02q_24JJZ3a00

Seznam obrázků

[1] - *Obrázek 1 - Spotify API*,
https://miro.medium.com/v2/resize:fit:1400/1*xt_TlsuG3FtSy22WPrqmNg.png
[2] - *Obrázek 2 - Drizzle ORM*,
https://media.licdn.com/dms/image/v2/D5612AQHHUzkz9Ie1Zg/article-cover_image-shrink_600_2000/article-cover_image-shrink_600_2000/0/1695866608071?e=2147483647&v=beta&t=_APvBuOfjBsgZpuvPP5M91g8qKptqAZGW4qzR2zsYfM
[3] - *Ukázka 1 - db.js, createClient*
[4] - *Ukázka 2 - fetchRandomArtistsByRandomGenre*
[5] - *Ukázka 3 - userSchema.js, users, userFavouriteArtists*