**EÖTVÖS LORÁND TUDOMÁNYEGYETEM**



Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

**Közösségi zeneajánló platform**

Témavezető: Szerző:  
**Pataki Norbert Lakosi Bence**Docens, PhD Programtervező informatikus BSc.

Budapest, 2023

# Tartalomjegyzék

Később generálva 2 oldal hagyva neki

tartalomjegyzék folyt…

# 1. Bevezetés

## 1.1 Témaválasztás

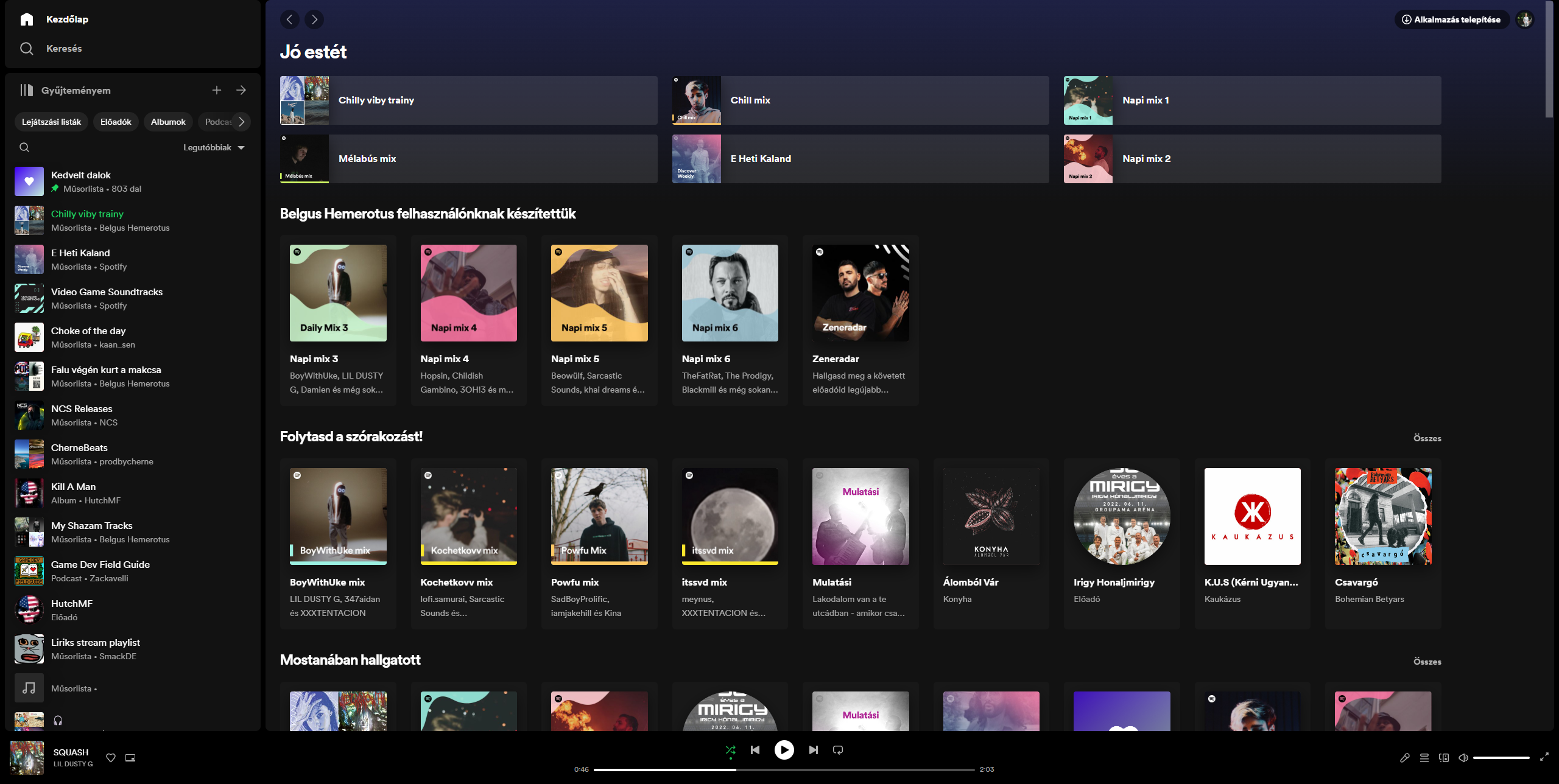
Az egyetemi éveim alatt mindig is nagy szerepet játszott a zene. Legyen az örömhírre vagy rossz hírre, tanulás alatt, közlekedés közben, sporthoz, és még sok más alkalom miatt. Több streaming szolgáltatót is kipróbálva, - többek között Tidal, YouTube Music, Amazon Music Unlimited, Apple Music, Spotify -, a Spotify-t találtam a leghasznosabbnak, mivel nem csak hivatalos zenéket találhatunk meg rajta és jobban elterjedt, mint konkurenciái, legalábbis Magyarországon.

Amit szerettem volna elérni a szakdolgozatommal, hogy egy kicsit átvariálva, kevesebb funkcióval, az én preferenciám szerint létrehozni egy úgymond ’’Spotify 2.0’’ – át. Amit átvettem nagyban tőle az a dizájn és magát a zenét is a Spotify-ból kérem le. Amit máshogy csinálok, hogy kizárólag a lejátszási listákon van a hangsúly. Ezeket tölti be az oldal, lehet kiválasztani őket és megnézni, milyen számok vannak rajta. Először a mi saját listáinkat tölti be az oldal, ezekkel nagyon interaktivitás nincs, csak eltudjuk indítani a lejátszást egy aktív eszközre, majd mutat ajánlott lejátszási listákat is. Ezek az ajánlások a Spotify által ajánlott listák, illetve más felhasználók által készített és ezen applikáció által beolvasott listák lehetnek. Ezeket tudjuk követni, így későbbre elmenteni, és szintén számokat lejátszani belőle egy aktív eszköz segítségével.

Dolgozatomat webes környezetben szerettem volna megvalósítani, mivel szakmai gyakorlatom alatt is webes technológiákkal foglalkoztam így, ezzel biztosítva a platform függetlenséget, és így a legkényelmesebb a felhasználóknak az elérés.

## 1.2 Spotify

A Spotify egy Svédországban kifejlesztett zenei streaming platform, ami ma már a világ sok más országában, területein elérhető. Tartalmai másolásvédettek, nagyobb zenei kiadóknak, illetve kisebb szabadúszó zenei alkotóknak is lehetőséget ad műveik feltöltésére és terjesztésére. A platform ingyen használható, ugyanakkor van egy prémium csomagja is. Ez a csomag többfajta módon is elérhető, más-más ár címkézéssel pl. Egyéni Prémium, Családi Prémium, Hallgatói Prémium. Többek között, amit átvettem zenék lejátszása funkciót, az ehhez a prémium-hoz kötött, így ezt csak olyan felhasználók tudják használni, akiknek van aktív prémium előfizetésük Spotify-ra.

  
**0. ábra** Spotify új webes dizájnja 2023.május

# 2. Alapvető technológiák

## 2.1 React

React[1], React.js, vagy ReactJS napjaink legelterjedtebb, ingyenes és nyílt forráskódú front-end JavaScript könyvtára. Fejlesztve a Facebook (most Meta) által, először 2013-ban került kiadásra. React-el komponens alapú felhasználói felületeket lehet létrehozni, deklaratív és egységbe zárt módon. Ennek előnye, hogy komponenseinket könnyedén újra felhasználhatjuk. React azonban csak az állapotkezeléssel, és DOM (Document Object Model) manipulánsával foglalkozik, így minden más tipikus front-end funkciók más csomagokon keresztül érhető el (pl. react router az útválasztáshoz).

React használható ’’egy oldalas’’, mobil, vagy szerver oldali betöltésű alkalmazások fejlesztésére más keretrendszerek által amik rá épülnek, mint például a Next.js. Tipikusan JSX-et használva (vagy TSX, ha typescript-et használunk) fejlesztjük React alkalmazásunkat, ami a háttérben átkonvertál szimpla JavaScriptre, így, ha akarjuk JS fájlokkal is dolgozhatunk, de JSX használata sokkal egyszerűbb, és nagyon könnyedén tanulható, szintaxisa szinte megegyező. A JSX egy ’’superset”-e a JS-nek.

## 2.2 Next.js

A Next.js[2] egy nyílt forráskódú webfejlesztéshez használt keretrendszer. React-re épül és ad hozzá több funkcionalitást. Legnagyobb előnye, hogy egyben front-end és back-end-ként is használható, nem kell külön egy back-end könyvtárat keresni és használni. Node.js-et használ, mint fontosabb technológia, ami a REST-en alapul, mint kommunikáció a front-end és back-end részek között.

Fontosabb funkciói, amivel kiegészíti és megkönnyíti a React használatát:

* Oldal útválasztás – ’’Out-of-the-Box'’ működik fájlrendszer alapján
* Statikus oldali betöltés
* Szerver oldali betöltés
* Hibrid oldali betöltés
* Könnyű adatlekérés back-end-ről

## 2.3 Prisma

A Prisma[3] egy Node.js és TypesScript-en alapuló ORM (Object-relation mapping / Objektum-relációs leképzés). Ennek segítségével nem kell ’’nyers’’ SQL utasításokkal manipulálni adatbázisunkat (ennek egyik előnye az sql-injection támadások elleni védelem), így gyorsan és olvashatóan készíthetünk modelleket a sémánkban, könnyedén migrálhatunk adatokat, és GraphQL-hez hasonló módon végezhetünk ’’runtime’’ CRUD operációkat adatbázisunkban. Mindehhez társul egy ’’logger’’, amivel megtekinthetjük a háttérben lefutó SQL parancsokat. Jelenleg támogatott adatbázis fajtái a PostgreSQL, MySQL, SQLite, és MongoDB.

## 2.4 tRPC

A tRPC[4] egy End-to-end típus biztonságos API könyvtár. Legnagyobb előnye, hogy a fejlesztés során könnyebben el lehet kapni kisebb bug-okat, amik nagy fejtörést okoznának, ha egy elő production környezetben kerülnének elő. Más megemlítésre méltó előnye, hogy pl. VSCode-ban nagyban segít az autocomplete, és az egész könyvtár lightweight és nincs build lépés benne, így nem lesz generálva miatta új kód, tehát nem lesz miatta ’’runtime loat’’.

## 2.5 NextAuth.js

A NextAuth.js[5] egy nyílt forráskódú autentikációs megoldás Next.js alkalmazásokhoz. Egyenesen a szerver nélküli elvet használó Next.js-hez volt fejlesztve. Ugyanakkor sok szerver oldali alkalmazáshoz is használható, tehát nagyon rugalmas. Lehet vele többek között OAuth 1.0, 1.0A, 2.0, és email/jelszó nélküli autentikációt végezni. Adapter támogatása is van, mi esetünkben ez a Prisma és ezt az opciót bekapcsolva egyszerűen az adatbázisunkba mentődnek el a felhasználók. Sok ’’Provider”-t támogat többek között, Google, Apple, Facebook, Spotify, stb.

## 2.6 TailwindCSS

A TailwindCSS[6] egy nyílt forráskódú CSS keretrendszer. A legnagyobb különbség más keretrendszerekkel ellentétben, mint például a Bootstrap, hogy nincsenek benne előre meghatározott klasszok, például gombok és táblák, hanem egyfajta ’’utility” klasszokat definiál. Ennek eredményeképp minden HTML elemre külön vagy csoportosítva lehet használni ezeket a kalsszokat akárhogyan keverve őket. Ugyanakkor akár saját új klasszt is létrelehet vele hozni. Plugin-okat is lehet használni, amik lényegében más fejlesztők által megírt új klasszok.

## 2.7 Spotify Web Api

A Spotify Web Api[7] REST-en alapuló, a Spotify adat katalógushoz kapcsolódó API. Használatához felhasználói szinten külön jogosultság kell. JSON meta adatokat ad vissza különböző kérésünkre. Ezek a kérések lehetnek zene előadókról, számokról, lejátszási listákról, vagy épp a lejátszás távoli vezérlése.

# 3. Felhasználói dokumentáció

## 3.1 Rövid ismertetés

Az alkalmazás lehetőséget ad a felhasználó Spotify regisztrációja/bejelentkezése után a lejátszási listái betöltésére, azok zenéinek megnézésére, ajánlott lejátszási listák megnézésére, azok elmentésére, és értékelésére {+ funkció még nem biztos, hogy belefér}. Ha a felhasználónak van Spotify Prémium csomagja, akkor távirányító szerűen egy aktív Spotify eszközön letudja játszani a látott zenéket.

## 3.2 Célközönség