BGSZC Logisztikai és Kereskedelmi Technikum és Szakképző Iskola

Vizsgaremek

*Outclass*

Készítette: Celik Alper Fatih, Chen Jun Bo, Drávicz Gyula Attila

**Tartalom**

1. Bevezetés 5. Oldal
   1. Feladat leírás 5. Oldal
   2. A felhasznált ismeretek 5. Oldal
      1. Frontend fejlesztés 5. Oldal
      2. Backend fejlesztés 5. Oldal
      3. Egyéb ismeretek 5. Oldal
   3. A felhasznált szoftverek 5. Oldal
      1. Fejlesztőeszközök 5. Oldal
      2. Technológiák és könyvtárak 6. Oldal
2. Felhasználói dokumentáció 6. Oldal
   1. A program általános specifikációja 6. Oldal
   2. Rendszerkövetelmények 6. Oldal
      1. Hardver követelmények 6. Oldal
      2. Szoftver követelmények 6. Oldal
   3. A program telepítése 6. Oldal
      1. Szükséges egy helyi webszerver, mint például a XAMPP 6. Oldal
      2. Backend beállítás 8. Oldal
         1. .env fájl beállítása 8. Oldal
      3. Függőségek telepítése, indítása 8. Oldal
         1. Backend 8. Oldal
         2. Frontend 8. Oldal
   4. A program használatának részletes leírása 8. Oldal
      1. Regisztráció és bejelentkezés 8. Oldal
         1. Regisztráció 9. Oldal
         2. Bejelentkezés 9. Oldal
      2. Zene keresés, letöltés, lejátszás 10. Oldal
         1. Zene keresés 10. Oldal
         2. Zene lejátszás 10. Oldal
         3. Zene letöltés 11. Oldal
      3. Zene feltöltése 11. Oldal
      4. Adminisztrátor funkciók 12. Oldal
      5. Kijelentkezés 12. Oldal
3. Fejlesztői dokumentáció 12. Oldal
   1. Az alkalmazott fejlesztői eszközök 12. Oldal
      1. Backend 12. Oldal
      2. Frontend 13. Oldal
   2. Adatmodell leírása 13. Oldal
      1. Users tábla szerkezete 13. Oldal
      2. Aritsts tábla szerkezete 13. Oldal
      3. Albums tábla szerkezete 13. Oldal
      4. Genres tábla szerkezete 14. Oldal
      5. Songs tábla szerkezete 14. Oldal
      6. Playlists tábla szerkezete 14. Oldal
      7. Adatbázis kapcsolatainak összefoglalása 14. Oldal
   3. Részletes feladatspecifikáció, algoritmusok 15. Oldal
      1. Zene letöltés 15. Oldal
         1. Frontend 15. Oldal
         2. Backend 15. Oldal
      2. Felhasználó regisztrálás 16. Oldal
         1. Frontend 16. Oldal
         2. Backend 17. Oldal
   4. Tesztelési dokumentáció 18. Oldal
      1. Tesztesetek 18. Oldal
         1. Új felhasználó regisztrációja 18. Oldal
         2. Felhasználói bejelentkezés 19. Oldal
         3. Új zene feltöltése 20. Oldal
         4. Törölt zenékhez tartozó fájlok fizikai törlése 21. Oldal
4. Asztali alkalmazás (WinForms zenelejátszó) 22. Oldal
   1. Fő funkciók 22. Oldal
   2. Technikai alapok 22. Oldal
   3. Rendszerkövetelmények 22. Oldal
   4. Telepítés / futtatás 22. Oldal
   5. Használat 23. Oldal
5. Összefoglalás 23. Oldal
   1. Fejlesztési kihívások és problémák 23-25. Oldal
   2. Önértékelés 25-26. Oldal
   3. Továbbfejlesztési lehetőségek 26-30. Oldal
6. Felhasznált irodalom 30-31. Oldal
7. Ábrajegyzék 31-32. Oldal

# Bevezetés

Célunk egy Spotify-hoz hasonló webalkalmazás fejlesztése, amely lehetővé teszi a felhasználók számára zenék feltöltését és letöltését. Az alkalmazás biztosítja a felhasználók regisztrálását és bejelentkezését, személyre szabott fiók létrehozásával. A rendszer képes tárolni és kezelni különböző zenei adatbázis-elemeket, mint például előadók, zenei stílusok, dalok és hangszerek. Emellett könnyű keresési lehetőséget kínál előadók, hangszerek és stílusok alapján. A felhasználók személyre szabott lejátszási listákat hozhatnak létre, módosíthatnak és törölhetnek. A zene feltöltése és letöltése is lehetséges, miközben az alkalmazás felhasználóbarát és intuitív felületet biztosít a különböző képességű felhasználók számára. A rendszer kiemelt figyelmet fordít az adatbiztonságra, például a felhasználói jelszavak titkosításával.

## Feladat leírás

A projekt célja egy Spotify-hoz hasonló zenei streaming platform létrehozása, amely a következő funkciókat nyújtja:

* Felhasználókezelés: Regisztráció, bejelentkezés, személyes adatok módosítása.
* Zenei tartalom kezelése: Zenék feltöltése és letöltése (MP3 formátum, max. 5MB/fájl).
* Keresés és szűrés: Előadó, zenei stílus, vagy album alapján.
* Admin felület: Tartalom moderálása, felhasználók kezelése.

Követelmények:

* Reszponzív felület (PC, tablet, mobil).
* Biztonságos jelszó-tárolás (bcrypt titkosítás).
* REST API-alapú kommunikáció.

## A felhasznált ismeretek

### Frontend fejlesztés:

* React.js: Komponensalapú felület, hook-ok, context API.
* Axios: HTTP kérések kezelése.
* CSS-modulok: Reszponzív design és stíluskezelés.

### Backend fejlesztés:

* Node.js + Express.js: REST API végpontok implementálása.
* JWT (JSON Web Token): Felhasználói hitelesítés.
* MySQL/MariaDB: Adatbáziskezelés (felhasználók, zenék, lejátszási listák).
* Bcrypt: Jelszavak biztonságos tárolása.

### Egyéb ismeretek:

* Git: Verziókövetés és csoportmunka.
* API tesztelés: Thunder Client használata.

## A felhasznált szoftverek

### Fejlesztőeszközök:

* Visual Studio Code: Kódszerkesztő.
* MySQL: Adatbázis tervezés és kezelés.
* Thunder Client: API végpontok tesztelése.
* Git: Verziókövetés.

### Technológiák és könyvtárak:

* Frontend: React, React Router, Axios.
* Backend: Node.js 18, Express 4.x, MySQL2.
* Biztonság: Bcrypt.js, jsonwebtoken.
* Fájlfeltöltés: Multer.

# Felhasználói dokumentáció

## A program általános specifikációja

A zenei streaming alkalmazás a következő főbb jellemzőkkel rendelkezik:

* Felhasználóbarát felület: Egyszerű kezelés, intuitív navigáció.
* Többplatformos támogatás: Webböngészőben fut.
* Biztonságos hitelesítés: Jelszavak titkosítva tárolva (bcrypt).
* Keresési lehetőségek: Szűrés előadó, stílus vagy dalcím alapján.
* Lejátszási listák: Személyre szabott zenei gyűjtemények létrehozása.

## Rendszerkövetelmények

### Hardver követelmények

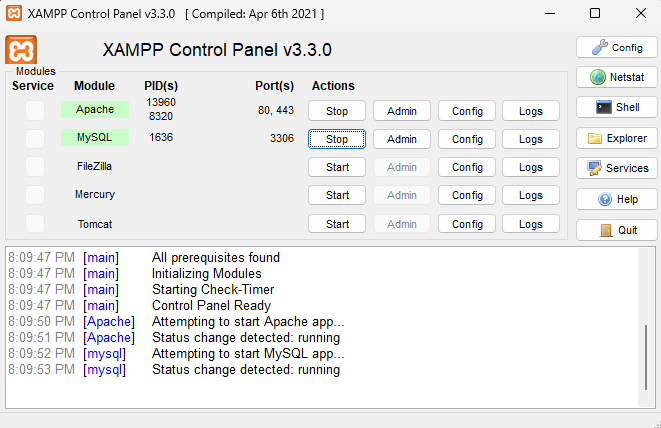
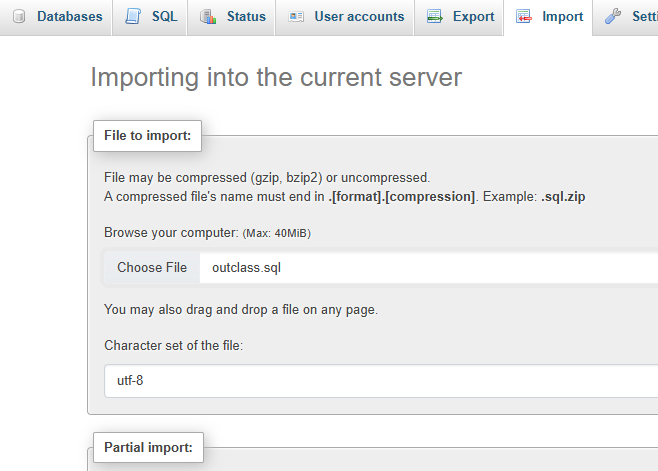
* Processzor: Ajánlott a legalább a 2 magos, 2Ghz.
* Memória: Legalább 4GB.
* Tárhely: Legalább 10GB, a zenefájlok, képek tárolására.
* Hálózat: 10 Mbps letöltési sebesség.

### Szoftver követelmények

* Operációs rendszer: Windows, macOS, Linux (Fedora, Ubuntu stb.).
* Mobil rendszerek: Android, iOS (böngészés).
* Node.js: 18 vagy újabb
* Adatbázis: MySql 8.0+
* Egyéb függőségek: npm 9.x, Git 2.35+ (verziókövetéshez)

## A program telepítése

### Szükséges egy helyi webszerver, mint például a XAMPP.

1. XAMPP telepítése
2. A XAMPP-on elindítjuk az Apache-t és a MySQL-t 
3. Megnyomjuk a MySQL mellett található ‘Admin’ gombot, amely elindítja a phpMyAdmin-t a böngészőnkön.
4. Importáljuk a GitHub repository-ban található ‘outclass.sql’-t

### Backend beállítás

#### .env fájl beállítása

* DB\_HOST=localhost
* DB\_USER=root
* DB\_PASSWORD=
* DB\_NAME=outclass
* DB\_PORT=3306
* JWT\_SECRET=kulcsocska
* NODE\_ENV=development

### Függőségek telepítése, indítása

#### Backend:

* cd backend
* npm install
* npm start / node .\server.js

#### Frontend:

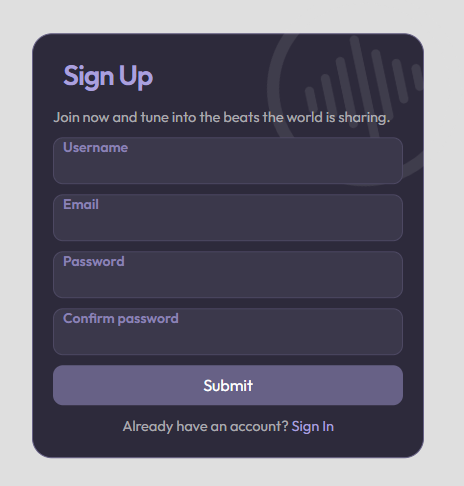
* cd frontend
* npm install
* npm run dev

## A program használatának részletes leírása

### Regisztráció és bejelentkezés

Az alkalmazás használatának első lépése a felhasználói fiók létrehozása, illetve a rendszerbe való bejelentkezés. Ezek a folyamatok biztosítják a felhasználok hozzáférését platform funkcióihoz.

#### Regisztráció



* A felhasználó a bejelentkezés form alján található ‘Sign up’ gombra kattint. Ez átirányítja a regisztrációs űrlapra.
* Az űrlap a következő mezőket tartalmazza:
  + Felhasználónév.
  + Email cím: Egyedi, tartalmaznia kell a ‘@’, illetve a ‘.’ karaktereket.
  + Jelszó: Minimum 6 karakter hosszú.
  + Jelszó megerősítése.
* Az űrlap valós idejű visszajelzést ad. Ha a két jelszó nem egyezik, a rendszer piros színnel jelzett a hibát küld vissza, és a ‘Submit’ gomb inaktív marad, amíg a hiba meg nem javul.
* Az adatok helyes kitöltése után a felhasználó a ‘Submit’ gombra kattint.
* A rendszer elküldi az adatokat a szerverre, amely ellenőrzi, hogy az email cím egyedi-e.
* Sikeres regisztráció esetén a felhasználó átirányításra kerül a bejelentkezési oldalra.

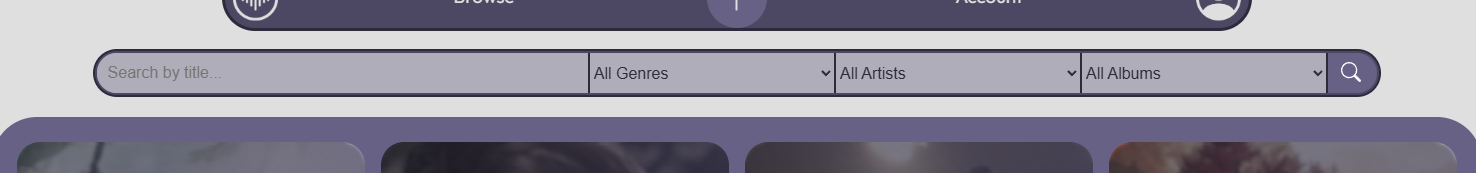
#### Bejelentkezés

* A bejelentkezés oldal igencsak hasonló a regisztrécióéhoz.
* Az űrlap tartalma a következő kettő mező:
  + Email cím.
  + Jelszó.
* A Submit gomb megnyomása után a rendszer ellenőrzi az adatok helyességét.
* Ha az email és jelszó páros érvényes, a szerver egy JSON Web Token-t (JWT) generál, amelyet a böngésző tárol.
* A felhasználó átirányításra kerül a főoldalra, ahol a zeneböngésző felület fogadja.

### Zene keresés, letöltés, lejátszás

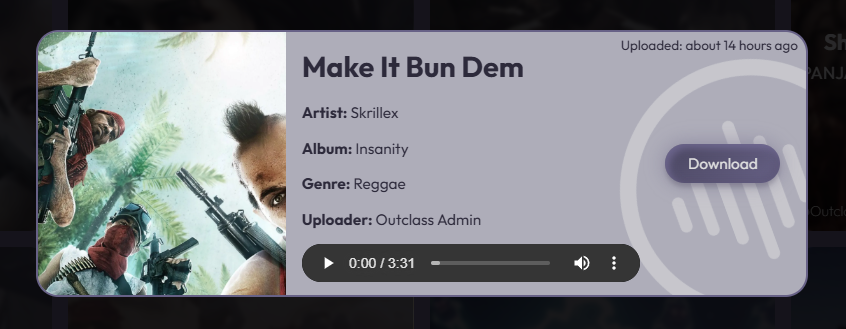
A platform lehetővé teszi a felhasználók számára, hogy különböző szűrők segítségével keressenek zenéket, amelyeket a felhasználók közvetlenül a böngészőről keresztül meg is tudnak hallgatni.

#### Zene keresés



* A főoldal tetején található Filter Bar a következő szűrési lehetőségeket tartalmazza:
  + Zene, illetve előadó neve alapján való szűrés.
  + Előadó alapján való szűrés.
  + Műfaj alapján történő szűrés.
  + Album alapján való szűrés.
* Miután kiválasztotta a megfelelő szűrőket, az Apply Filter gombra kattint.
* A rendszer valós időben frissíti az eredményeket, kártyák formájában megjelenítve a találatokat.
* A kártyák reszponzívak, így mobilon és asztali gépen is jól láthatóak.
* Minden zenei kártya tartalmazza a zene címét, az előadó nevét, az album nevét, egy borítóképet és a feltöltő felhasználónevét.
* Ha a keresésre nincsen találat, a következő üzenet jelenik meg: “There are no results matching the applied filters.”.

#### Zene lejátszás



* A keresési eredmények között a felhasználó kiválaszt egy zenét, amelynek rákattint a kártyájára.
* Ez egy felugró ablakot nyit meg.
* A felugró ablak a következő információkat:
  + A zene címe.
  + Az előadó neve.
  + Az album neve.
  + A zene műfaja.
  + A feltöltő felhasználóneve.
  + Hogy mikor töltötték fel.
  + A zene borítóképét.
  + A zene mp3 fájlt.
* A felugró ablakban a lejátszó nyilacskára kattintva a zene elindul. A böngésző HTML5 audio lejátszóját használja, amely gyors és stabil.
* A lejátszó lehetővé teszi a szüneteltetést, a hangerő szabályozását, valamint a zene adott pontjára való ugrást a csúszka segítségével.

#### Zene letöltés

* A letöltési funkció lehetővé teszi a felhasználók számára, hogy mp3 formátumban mentsék le a zenéket saját eszközeikre.
* A felugró ablakon található ‘Download’ gomb egy HTTP kérést küld a szervernek, amely a zene mp3 fájlját szolgálja ki.
* A böngésző automatikusan elindítja a letöltést a következő fájlnév formátumban: ‘Dal címe’ - ‘Előadó neve’ ('Make it Bun Dem – Skrillex').
* Letöltés előtt a felhasználónak ajánlott ellenőrizni, hogy van-e elég tárhelye az eszközén.

### Zene feltöltése

A zene feltöltése lehetővé teszi a felhasználók számára, hogy saját muzsikájukat és borítóképeiket megosszák a platformon, hogy mások is hozzáférhessenek azokhoz.



* A navigációs sávban található ‘+’ gombra kattintva a felhasználó a zene feltöltési űrlapra kerül.
* Az ‘upload’ oldalon található űrlap a következő mezőket tartalmazza:
  + Zene címe.
  + Zene műfaja, a felhasználó választhat 50 előre megadott műfaj közül.
  + Előadó neve.
  + Ha már egy meglévő előadótól szeretnének zenét feltölteni, akkor az ‘Existing Artist’ mezőben kiválaszthatja azt.
  + Album neve.
  + Ha már egy meglévő albumba kíván zenét feltölteni, akkor azt az ‘Existing Album’ legördülőmezőben tudja megtenni.
  + A borítókép jpg fájlja, ami maximum 5MB lehet, amennyiben a feltölteni kívánt fájl mérete ezt meghaladja, hibaüzenetet ad vissza.
  + A zene mp3 fájlja, amelynek mérete szintén maximum 5MB lehet, szintén hibaüzenetet ad vissza az ezt meghaladó fájloknak.
* Az űrlap kitöltése és a fájlok kiválasztása után a felhasználó az ‘Upload Song’ gombra kattinthat.
* A rendszer ellenőrzi a fájlformátumokat és a méretkorlátozásokat, majd feltölti a fájlokat a szerverre.
* Sikeres feltöltés esetén a felhasználó egy üzenetet kap:”Song successfully uploaded.”.
* A pontos adatok megadása elősegíti a többi felhasználót a zene megtalálásában. Ajánljuk a jó minőségű borítóképet, hogy a dal vizuálisan is vonzó legyen.

### Adminisztrátor funkciók

Az adminisztrátori jogosultságokkal rendelkező felhasználók további funkciókhoz férnek hozzá, amelyek a platform tartalmának és felhasználóinak kezelésére szolgálnak.

* Az adminisztrátorok törölhetik az összes feltöltött zenét, kiszűrve azokat, amelyek nem felelnek meg a platform szabályainak.
* A törléshez a ‘Download’ gomb alatt található ‘Delete’ gombra kell kattintani, majd megerősíteni a műveletet.
* Az admin jogosultsággal nem rendelkező felhasználók csak a saját maguk által feltöltött zenéket képesek letörölni.
* Az adminisztrátorok megtekinthetik a felhasználók listáját, és szükség esetén törölhetik a fiókokat.
* Az adminisztrátorok mindig alaposan ellenőrizzék a törlési műveleteket, mivel azok visszafordíthatatlanok.

### Kijelentkezés

A kijelentkezés funkció lehetővé teszi a felhasználók számára, hogy biztonságosan kilépjenek a platformról. Ez különösen fontos, ha a felhasználó nyilvános eszközön használja azt.

* A folyamat egyszerű, gyors, és a platform intuitív felületén könnyen elérhető.
* A bejelentkezett felhasználók a ‘browse’ oldal jobb felső sarkában találják a ‘Sign out’ gombot.
* A Logout gombra kattintva a rendszer azonnal törli a böngészőben tárolt JSON Web Token-t (JWT).
* A kijelentkezés során a rendszer biztosítja, hogy a JWT token érvénytelenítve legyen, így az többé nem használható a platform elérésére.
* Ez különösen fontos a fiók biztonságának fenntartása érdekében, főleg, ha a felhasználó az egy admin.
* A művelet nem igényel további megerősítést, hogy a felhasználói élmény gyors és zökkenőmentes maradjon.
* A kijelentkezés után a felhasználó automatikusan átirányításra kerül a bejelentkezés oldalra.

# Fejlesztői dokumentáció

## Az alkalmazott fejlesztői eszközök

### Backend

* **Nyelv:** JavaScript
* **Keretrendszer:** Express.js
* **Adatbázis:** MySQL
* **Hitelesítés:** JsonWebToken (JWT) + bcrypt jelszó-titkosítás
* **Fájlfeltöltés:** Multer middleware
* **Fejlesztői eszközök:** 
  + **IDE:** Visual Studio Code
  + **API tesztelés:** Thunder Client
  + **Verziókövetés:** GitHub

### Frontend

* **Környezet:** Vite
* **Keretrendszer:** React
* **HTTP Kliens:** Axios
* **Stíluskezelés:** CSS Modules

## Adatmodell leírása

Az adatmodell 6 fő táblából áll, melyek a működéséhez szükséges adatokat tárolják. Az alábbiakban részletezzük a táblákat és kapcsolataikat:

### Users tábla szerkezete

* Célja a felhasználók adatainak tárolása.
* Oszlopai:
  + userId: INT, elsődleges kulcs, egyedi azonosító.
  + userName: VARCHAR (255), felhasználónév.
  + userEmail: VARCHAR (255), email cím.
  + userPassword: VARCHAR (255), titkosított jelszó.
  + isAdmin: TINYINT (1), admin jogosultság.
  + userCreated: TIMESTAMP, regisztráció időpontja.

Példaadat:

* userId: 43, userName: “Sándor”, userEmail: “sandor@outclass.com”, userPassword: “$2y$10$...”, isAdmin: 0, userCreated: '2025-01-01 00:00:00'

### Aritsts tábla szerkezete

* Célja az előadók számontartása.
* Oszlopai:
  + artistId: INT, elsődleges kulcs, egyedi azonosító.
  + artistName: VARCHAR (255), előadó neve.

Példaadat:

* ArtistId: 21, artistName: “BigLajosX”

### Albums tábla szerkezete

* Cálja az albumok számontartása
* Oszlopai:
  + albumId: INT, elsődleges kulcs, egyedi azonosító.
  + albumName: VARCHAR (255), album neve.
  + artistId: INT, idegen kulcs, kapcsolódó előadó.

Példaadat:

* albumId: 2, albumName: “lajos emléke”, artistId: “21”

### Genres tábla szerkezete

* Célja a műfajok számontartása.
* Oszlopai:
  + genreId: INT, elsődleges kulcs, egyedi azonosító.
  + genreName: VARCHAR (255), műfaj neve.

Példaadat:

* genreId: 3, genreName: “Rock”

### Songs tábla szerkezete

* Célja a zenék adatainak való eltárolása.
* Oszlopai:
  + SongId: INT, elsődleges kulcs, egyedi azonosító.
  + songName: VARCHAR (255), zene címe.
  + artistId: INT, idegen kulcs, kapcsolódó előadó.
  + albumId: INT, idegen kulcs, kapcsolódó album.
  + genreId: INT, idegen kulcs, kapcsolódó műfaj.
  + songUploaderId: INT, idegen kulcs, feltöltő felhasználó.
  + songPath: VARCHAR (255), zene fájl elérési útja.
  + songImage: VARCHAR (255), borítókép elérési útja.
  + songUploadedAt: TIMESTAMP, feltöltés időpontja.

Példaadat:

* songId: 820, songName: “Hatalmas Cipők”, artistId: 21, albumId: 2, genreId: 25, songUploaderId: 10, songPath: “/uploads/songs/song1.mp3”, songImage: “/uploades/covers/cover3.png”, songUploadedAt: ‘2025-04-19 17:45:00’

### Playlists tábla szerkezete

* Célja a felhasználók által létrehozott lejátszásilisták számontartása.
* Oszlopai:
  + plId: INT, elsődleges kulcs, egyedi azonosító.
  + plName: VARCHAR (255), lejátszásilista neve.
  + plOwnerId: INT, idegen kulcs, létrehozó felhasználó.
  + plCreated: TIMESTAMP, létrehozás időpontja.
  + songIds: INT, idegen kulcs, kapcsolódó zenék.

Pédaadat:

* plId: 18, plName: “Edzésre való muzsikák”, plOwner: 1, plCreated: ‘2025-04-19 17:45:00’, so

### Adatbázis kapcsolatainak összefoglalása

* Users -> Songs
* Users -> Playlists
* Artists -> Albums
* Artists -> Songs
* Albums -> Songs
* Genres -> Songs
* Songs -> Playlists

## Részletes feladatspecifikáció, algoritmusok

### Zene letöltés

A felugró ablakon található ‘Download’ gomb a következő módon műkszik:

#### Frontend

* A gomb megnyomását követően előkészítünk egy letöltési címet, például:   
   http://localhost:3000/songs/download/91, ahol a 91 a zeneszám azonosítója.
* Majd kérést küldünk a szervernek, amelyre, ha nem kapunk hibát, akkor az mp3 file-t átalakítjuk olyan formátumba, amit a böngésző kezelni tud, ilyen a Blob.
* Ezek után elindítjuk a letöltést a felhasználó számítógépére, mielőtt töröljük az ideiglenes webcímet, hogy ne foglaljon felesleges helyet.
* Végül pedig a hibakezelés.
* Így néz ki a letöltést elindító függvény:

const handleDownload = async () => {

try {

const downloadUrl = `http://localhost:3000/songs/download/${*song*.songId}`;

// Fetch hívás a letöltéshez

const response = await fetch(downloadUrl);

if (!response.ok) {

throw **new** *Error*(`HTTP error! status: ${response.status}`);

}

// Letöltés

const blob = await response.blob();

const blobUrl = window.*URL*.createObjectURL(blob);

const link = document.createElement('a');

link.href = blobUrl;

link.download = `${*song*.songName} - ${getArtistName(*song*.artistId)}.mp3`;

link.click();

// Hely felszabadítás

window.*URL*.revokeObjectURL(blobUrl);

} catch (error) {

console.error("Letöltési hiba:", error);

alert("A letöltés sikertelen. Próbáld újra!");

}

}

* A következő módon hívjunk meg a függvényt:

<button *onClick*={handleDownload} *className*='downloadButton'>Download</button>

#### Backend

* A letöltés végpont a következő:

router.get('/download/:songId', async (*req*, *res*) => {

try {

// Zene adatainak lekérése az adatbázisból

const [song] = await pool.query('SELECT songPath, songName FROM songs WHERE songId = ?', [*req*.params.songId]);

if (!song[0]) return *res*.status(404).send('Zene nem található');

// Elérési út összeállítása

const filePath = path.join(\_\_dirname, '../', song[0].songPath);

// Fájlnév tisztítása (ékezeteket, szóközök eltávolítása)

const cleanedFileName = song[0].songName

.normalize('NFD').replace(/[\u0300-\u036f]/g, '')

.replace(/\s+/g, '\_') + '.mp3';

// Letöltés

*res*.download(filePath, cleanedFileName, {

headers: {

'Content-Disposition': `attachment; filename="${cleanedFileName}"`

}

});

} catch (error) {

console.error('Letöltési hiba:', error);

*res*.status(500).send('Szerverhiba');

console.log(filePath);

}

});

* Az adatbázisból kikeressük a megadott azonosítóval rendelkező zenét.
* Ha megtaláltuk, összerakjuk a teljes elérési utat, ha nem, hibaüzenetet kapunk vissza.
* A kapott fájlnevet megformázzuk (ékezeteket, szóközöket kivesszük).
* Elküldjük a file-t a kliensnek.

### Felhasználó regisztrálás

A felhasználók regisztrálása a következő módon műkszik:

#### Frontend

* Megakadályozzuk, hogy a felhasználók alapértelmezett módon ne küldhessék el az űrlapot.
* Megnézzük, hogy a megadott jelszavak megegyeznek-e egymással, illetve, hogy minden mező helyesen van-e kitöltve.
* Elküldjük a kitöltött űrlap adatait a szervernek a ‘http://localhost:3000/users/register’ végpontnak.
* Sikeres regisztráció esetén átirányítjuk a felhasználót a bejelentkezés oldalra.

const handleSubmit = async (*e*) => {

*e*.preventDefault();

setError('');

if (formData.password !== formData.confirmPassword) {

setError("A jelszavak nem egyeznek!");

return;

}

if (!validateForm()) return;

try {

const response = await axios.post('http://localhost:3000/users/register', {

userName: formData.username,

userEmail: formData.email,

userPassword: formData.password

});

if (response.data.success) {

navigate('/signin');

}

} catch (err) {

setError(err.response?.data?.message || 'Regisztráció sikertelen');

}

};

* A következő módon hívjuk meg a függvényt:

<form *className*="form" *onSubmit*={handleSubmit}>

#### Backend

* A végpont kinézete a következő:

router.post('/register', async (*req*, *res*) => {

const { userName, userEmail, userPassword } = *req*.body;

try {

const [existingUser] = await pool.query('SELECT \* FROM users WHERE userEmail = ?', [userEmail]);

if (existingUser.length > 0) {

return *res*.status(400).json({ message: 'Email already exists' });

}

const salt = await bcrypt.genSalt(10);

const hashedPassword = await bcrypt.hash(userPassword, salt);

const [result] = await pool.query(

'INSERT INTO users (userName, userEmail, userPassword) VALUES (?, ?, ?)',

[userName, userEmail, hashedPassword]

);

*res*.status(201).json({ success: true, userId: result.insertId });

} catch (error) {

console.error('Registration error:', error);

*res*.status(500).json({

message: 'Szerverhiba',

...(process.env.NODE\_ENV === 'development' && { error: error.message })

});

}

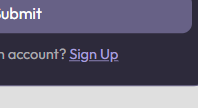
});

* A kérésből kiolvassuk a felhasználónevet, emailt, illetve a jelszót.
* Megnézzük, hogy van-e már az adott email-el rendelkező felhasználó, ha van visszautasítjuk a kérést.
* Bcrypt-el titkosítjuk a jelszót ‘hash’-el, növeljük a biztonságot ‘salt’-tal.
* Beszúrjuk az új felhasználót az adatbázisba.
* Ha nem sikerül, hibaüzenetet küldünk.

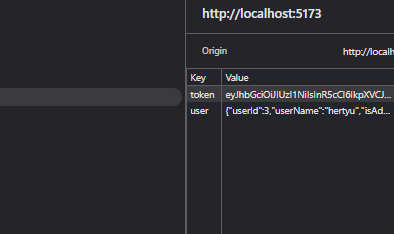
## Tesztelési dokumentáció

### Tesztesetek

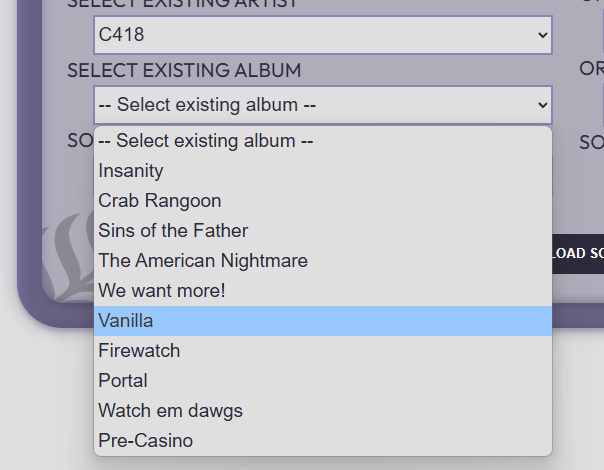
#### Új felhasználó regisztrációja

* Annak ellenőrzése, hogy az új felhasználók sikeresen regisztrálhatnak az alkalmazásba, és az adataik megfelelően mentésre kerülnek az adatbázisba.
* Előfeltételek:
  + Az alkalmazás működőképes.
  + Az adatbázis aktív.
  + A regisztrációs űrlap elérhető a nyilvános felületen.
  + A validációk működnek, pl: jelszóhossz, jelszók megegyezése stb.
* Tesztlépések:
  + Indítsa el az alkalmazást.
  + Nyomja meg a Sign in form alján található ‘Sign up’ gombot.
  + Töltse ki a mezőket arra ügyelve, hogy eleget tegyen a megkötéseknek.
  + Próbáljon meg nem megegyező jelszavakat megadni.
  + Próbáljon meg duplikált email címet megadni.
  + Nyomja meg a ‘Submit’ gombot.
  + A Sign in formon próbáljon meg bejelentkezni az imént megadott adatokkal.
  + Siker esetén ellenőrizze le az adatbázist, hogy megjelent-e az új felhasználó.
* Várt eremény:
  + A rendszer sikeres regisztrációs üzenetet jelenít meg.
  + A felhasználó bekerül az adatbázisba a megadott adatokkal.
  + A regisztrált e-mail-címmel történő későbbi bejelentkezés működik.
  + Nem történik hiba vagy adatvesztés.
* Teszteredmény:
  + A felhasználó regisztrációja rendben lezajlott, az adatok elmentésre kerültek, a bejelentkezés működik.

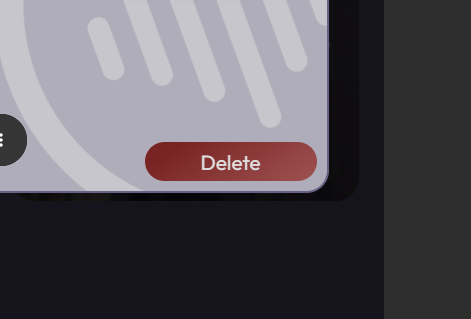
#### Felhasználói bejelentkezés

* Annak ellenőrzése, hogy egy korábban regisztrált felhasználó sikeresen be tud-e jelentkezni az alkalmazásba érvényes hitelesítő adatokkal.
* Előfeltételek:
  + Az alkalmazás működőképes.
  + Az adatbázis aktív.
  + A bejelentkezési űrlap elérhető a nyilvános felületen.
  + Legalább egy felhasználó már regisztrálva van az adatbázisban.
  + A jelszó titkosítva van tárolva, de a rendszer képes azt helyesen ellenőrizni.
* Tesztlépések
  + Indítsa el az alkalmazást.
  + A felugrő oldal a bejelentkező, ‘Sign in’ oldal fogja fogadni.
  + Adja meg a már az adatbázisba felvett felhasználó adatát.
  + Kattintson a ‘Submit’ gombra.
  + Amennyiben sikeres volt a bejelentkezés, át fog irányulni az alkalmazás fő, zene keresés oldalra.
  + Amennyiben viszont helytelen adatokat adott meg, egy hibaüzenetet fog kapni.
  + Nézze meg hogy létrejött-e a token.
* Várt eredmény:
  + A bejelentkezés sikeres, a felhasználót átirányítja a rendszer a főoldalra.
  + Nem jelenik meg hibaüzenet.
  + A bejelentkezés után elérhetőek a csak hitelesített felhasználók számára szánt funkciók.
  + A token tárolás sikeresen megtörténik.
* Teszteredmény:
  + A rendszer bejelentkezett a felhasználóval, helyes adatokat kapott, elérhetőek az authentikált funkciók, a token sikeresen tárolva lett.

#### Új zene feltöltése

* Annak ellenőrzése, hogy a felhasználó sikeresen feltölthet új zenét mp3 fájl formátumában és borítóképet JPG formátumban, és hogy az adatok megfelelően elmentődnek.
* Előfeltételek
  + A felhasználó be van jelentkezve az alkalmazásba.
  + A „Zene feltöltése” funkció elérhető és aktív.
  + A fájlformátumok ellenőrzése (pl. csak MP3 és JPG engedélyezett) be van állítva.
  + A fájlkezelés működőképes.
  + Az adatbázis működőképes.
* Tesztlépések
  + Jelentkezzen be.
  + Navigáljon az ‘/upload’ oldalra.
  + Töltse ki az űrlapot.
    - Ügyeljen arra, hogyha már létező előadót vagy albumot akar kiválasztani, akkor a legördülőmenüből tegye meg azt.
    - Figyeljen arra, hogy az Ön által feltöltött mp3 file nem haladja meg az 5MB-ot.
    - Figyeljen arra, hogy az Ön által feltöltött jpg file nem haladja meg az 5MB-ot.
    - Csak mp3 file-okat fogad el azt űrlap zenefájlként.
    - Csak jpg file-oket fogad el az űrlap borítóképként.
  + Kattintson az „UPLOAD SONG” gombra.
  + Várja meg a rendszer visszajelzését, és először ellenőrizze, hogy az új zeneszám megjelenik-e az adatbázisban, majd azt, hogy megjelenik-e a zene listában.
  + Ellenőrizze, hogy a fájlok fizikailag megjelentek-e a fájlredszerben.
* Várt Eredmény
  + A rendszer sikeresen menti az új zenei bejegyzést az adatbázisba.
  + Az MP3 és JPG fájlok megfelelő helyre kerülnek a fájlrendszerben.
  + A feltöltött zeneszám azonnal megjelenik a zene listában
  + A felhasználó pozitív visszajelzést kap (“Song successfully uploaded.”).
* Teszteredmény
  + A zene megjelenik az adatbázisban és a felületen, a fájlok a helyükön vannak.

#### Törölt zenékhez tartozó fájlok fizikai törlése

* Az adatbázisból törölt zenékhez tartozó zenefájlok (MP3) és képek (.JPG) törlése a tárhely megtakarítása érdekében.
* Előfeltételek:
  + Az alkalmazás működőképes.
  + Az adatbázis aktív.
  + Létezik legalább egy zenei tétel az adatbázisban.
  + Fizikailag elérhetőek a fájlok.
  + A fájlok elérési útvonalait az adatbázis tárolja.
* Tesztlépések:
  + Bejelentkezés az adminisztrációs felületen.
  + Egy a bejelentkezett felhasználó által feltöltött zene kiválasztása, kivéve, ha a felhasználó admin.
  + Törlés gomb megnyomása.
  + A felugró ablakok elfogadása.
  + Amennyiben minden jól sikerült, egy másik felugró ablak értesíti erről.
  + Ellenőrzés, hogy a zenei tétel valóban eltűnt-e az adatbázisból.
  + Navigáljon a fájlrendszeren belül az ‘uploads/songs’, illetve az ‘uploads/images’ mappába.
  + Ellenőrzés, hogy fizikailag elűntek-e a mappából.
* Várt eredmény:
  + Az adatbázisból törölt zenei tételhez tartozó fájlok, mind az mp3, mind a jpg, eltűnnek a fájlrendszerből.
  + A fájlrdenszer mérete csökken, kevesebb tárhelyet foglalva.
* Teszteredmény:
  + A fájlok törlődtek a fájlrendszerből.

# Asztali alkalmazás (WinForms zenelejátszó)

A webes Outclass‑szal párhuzamosan készült egy Windows‑os, offline zenelejátszó is, hogy internet nélkül is demonstrálható legyen a legalapvetőbb funkció­kör.

## Fő funkciók

* MP3/WAV fájlok feltöltése vagy drag‑and‑drop importálása.
* Lista nézet: Cím, előadó, hossz automatikus kitöltése
* Lejátszás / szünet, előző / következő szám, törlés
* Állapot‑csúszka: aktuális idő, oda‑ugrás
* Hangerő‑szabályzás
* Billentyűparancsok: Space, ← →, Delete

## Technikai alapok

|  |  |
| --- | --- |
| **Technológia** | **Leírás** |
| C# / .NET 8 | Keretrendszer |
| WinForms | UI |
| NAudio | Hangdekódolás |
| TagLibSharp | ID3-tag olvasás |
| Visual Studio 2022 | Fejlesztőkörnyezet |

## Rendszerkövetelmények

* Windows 10/11 (x64)
* .NET 6 vagy újabb runtime
* 2 magos CPU, 2GB RAM, 50 mb tárhely

## Telepítés / futtatás

1. Setup.exe / OutclassAdmin.msi telepítése.
2. ‘MusicPlayerWinForms.exe’ futtatása

## Használat

* Fájlok behúzása (drag and drop), az ‘Upload’ gombra való kattintás után kiválasztása.
* Duplaklikk a megjelent zenére, vagy pedig a lejátszás gomb megnyomása.
* Amennyiben több zene is be van tárazva, az ‘előző’, illetve a ‘következő’ gombra kattintva lehet váltani.
* Az alul látható csúszka segítségével ugrálhatunk a zene különböző pontjai között.
* A jobboldalt található csúszka pedig a hangerőt szabályozza.

# Összefoglalás

## Fejlesztési kihívások és problémák

* Drávicz Gyula Attila nehézségei

A projekt során az adatbázis megtervezése közben több kihívás is adódott, főleg amiatt, hogy hogyan lehet a különböző táblákat (felhasználók, zenék, előadók, műfajok stb.) logikusan és átláthatóan összekapcsolni. Többször is előfordult, hogy újra kellett gondolni egy-egy kapcsolatot vagy szerkezetet, hogy a rendszer ne csak működjön, hanem hosszú távon is jól kezelhető és bővíthető maradjon. Különösen nagy figyelmet kellett fordítanom az adatkapcsolatok normalizálására, hogy elkerüljük az adatredundanciát és biztosítsuk az adatok integritását. Ez gyakran aprólékos munkát jelentett, hiszen egy-egy módosítás hatással volt más részekre is, így az egész struktúra újragondolásához vezetett.

Emellett nehézséget okozott az is, hogy a dokumentációt úgy kellett megírni, hogy technikailag pontos legyen, de mégis érthető azok számára is, akik nem ismerik mélyen a háttérfolyamatokat. Külön figyelmet kellett fordítani arra, hogy ne csak felsoroljam a részeket, hanem logikusan mutassam be az összefüggéseket is. Ezt főként az tette nehézzé, hogy egyszerre kellett szakmai szemmel és laikus fejjel gondolkodnom. Több részletet is újra kellett fogalmazni vagy átrendezni, mire kialakult az a szerkezet, ami mindkét oldalnak megfelel. Ezek a feladatok időigényesek voltak, de közben rengeteget tanultam arról, hogyan lehet komplex technikai tartalmat világosan és rendszerezetten bemutatni, valamint arról is, hogy mennyire fontos az átlátható adatstruktúra egy nagyobb rendszer működésében.

* Celik Alper Fatih nehézségei

A projekt során számomra az egyik legnagyobb kihívást a frontend és a backend integrációja jelentette. Mivel mindkét területért én feleltem, folyamatosan koordinálnom kellett a két oldal közötti kommunikációt, hogy az adatáramlás és a funkciók összhangban működjenek. Ez különösen a tokenes hitelesítésnél és a zenei fájlok kezelésénél volt nehézkes, mivel mindkét funkció bonyolult logikát igényelt, és már kisebb eltérés is hibát okozhatott a működés során. A hitelesítési folyamatban pontosan kellett kezelni a tokenek generálását, érvényesítését és lejáratát, miközben biztosítani kellett a felhasználói élményt is. A zenei fájlok feltöltése, streamelése és törlése során pedig arra kellett figyelni, hogy minden művelet biztonságos legyen, és megfelelő visszajelzést adjon a felhasználónak.

A frontend fejlesztés sem volt egyszerű: itt az egyik fő nehézséget az jelentette, hogy egyszerre kellett optimalizálnom az alkalmazást mobilra és asztali gépre. A különböző képernyőméretek és eszközök eltérő viselkedése miatt gyakran előfordultak kisebb-nagyobb eltérések, amelyeket egyenként kellett javítani és újra tesztelni. Ez nemcsak technikai, hanem dizájn szempontból is kihívást jelentett, mert minden elemnek reszponzívan kellett viselkednie, miközben vizuálisan is egységes maradt. A felhasználói felület dizájnját folyamatosan finomítottam: figyeltem a színekre, gombok elhelyezésére, betűméretekre, animációkra és interakciókra is, hogy az egész alkalmazás letisztult, modern és jól használható legyen. Ez a fajta részletekbe menő munka sok időt igényelt, de úgy érzem, megérte, mert a végeredmény egy stabil és felhasználóbarát rendszer lett.

* Chen Jun Bo nehézségei

A projekt során több kihívással is szembesültem, különösen a WinForms alapú zenelejátszó fejlesztése közben. Az eseményvezérelt grafikus felület és az aszinkron zenelejátszás működésének összehangolása technikailag összetett feladatnak bizonyult. Könnyen kialakulhattak deadlock-helyzetek, főként akkor, amikor a felhasználó gyors egymásutánban több vezérlőelemet – például a lejátszás, szünet, előző/következő gombokat – használta. Emellett az állapot- és hangerő-csúszkák valós idejű frissítése is folyamatos figyelmet igényelt, mivel az események gyors egymásutánban történő kezelése könnyen instabilitáshoz vezethetett. Sok kísérletezésre, hibakeresésre és részleges újratervezésre volt szükség, mire sikerült olyan architektúrát kialakítanom, amely minden funkciót gördülékenyen és megbízhatóan működtet, akár folyamatos felhasználói interakció mellett is. Ezzel a folyamattal sokat fejlődtem a párhuzamos végrehajtás, eseménykezelés és GUI-stabilitás területén.

Az autentikációs rendszer tervezésekor és implementálásakor szintén több nehézséggel kellett szembenéznem. A JWT-tokenek lejáratának pontos és biztonságos kezelése mellett külön figyelmet kellett fordítanom arra is, hogy a middleware-ek megfelelően ellenőrizzék a jogosultságokat minden egyes végponton. Ez nemcsak technikai, hanem logikai kihívást is jelentett: a rendszernek egyszerre kellett biztonságosnak, hatékonynak és fenntarthatónak lennie. Ezen kívül törekedtem arra is, hogy a kód olvasható és könnyen bővíthető maradjon, mivel egy többtagú csapatban ez kulcsfontosságú szempont. A problémák leküzdése közben rengeteget tanultam a biztonságos backend-fejlesztésről, a .NET környezet haladó használatáról, és arról is, hogyan lehet jól strukturált, biztonságos és stabil rendszereket építeni a gyakorlatban.

## Önértékelés

* Drávicz Gyula Attila önértékelése

A projekt során főként az adatbázis tervezése és megvalósítása volt az én feladatom. Ez elég komoly kihívást jelentett, mivel több különböző entitást kellett összekapcsolni (felhasználók, zenék, előadók, műfajok, stb.), és fontos volt, hogy a rendszer hosszú távon is átlátható és jól kezelhető maradjon. Az elején sokat kellett gondolkozni a logikus felépítésen, de végül sikerült egy stabil struktúrát kialakítani, amit később könnyen tudtunk használni a backend oldalon is.

Emellett az én főfeladatom a dokumentáció írása volt, amit úgy kellett összeállítani, hogy világos legyen azok számára is, akik nem látnak bele teljesen a technikai részletekbe. Próbáltam átláthatóan és logikusan felépíteni a részeket, hogy tükrözze a rendszer működését, céljait és a megvalósítás folyamatát is.

A munkám során sokat fejlődtem MySQL-ben és a dokumentációs gyakorlatban is, valamint megtapasztaltam, milyen egy csapatban, konkrét határidőre dolgozni egy komplex rendszeren.

* Celik Alper Fatih önértékelése

Az alkalmazás frontend és backendjéért egyaránt én feleltem. A szerveroldali fejlesztésnél én írtam meg az API-végpontokat, ezek felelnek többek között a regisztrációért, bejelentkezésért, tokenes hitelesítésért, zenei fájlok kezeléséért, keresésekért és a playlist funkciókért. Sokat dolgoztam azon, hogy a backend biztonságos és megbízható legyen, a jelszavakat bcrypttel titkosítottuk, a hozzáféréseket pedig tokennel oldottuk meg Bo-val.

A frontendet React segítségével készítettem el. Igyekeztem egy modern, könnyen kezelhető felületet kialakítani, ahol minden funkció gyorsan elérhető: keresés, zenék kezelése, lejátszási listák, profilbeállítások stb. Külön figyelmet fordítottam arra, hogy mobilon és asztali gépen is jól működjön az alkalmazás. Mindezek mellett a dokumentáció írásából is kivettem a részem.

Rengeteget tanultam a projekt alatt, főleg React-ben fejlődtem sokat. A teljes rendszer összekapcsolása elég összetett feladat volt, néha elklételkedtem rajta, hogy nem-e túl nagy falat ez a feladat számunkra, de ettől függetlenül nagyon élveztem, hogy egy ilyen átfogó és működőképes alkalmazást csinálhattam.

* Chen Jun Bo önértékelése

A vizsgaremekben két fő területet vállaltam:

Biztonságos autentikáció: megterveztem és implementáltam a regisztráció‑, bejelentkezés‑, session‑ és JWT‑kezelést. Külön figyeltem a jelszavak bcrypt‑tel való titkosítására és a tokenek lejáratának megfelelő kezelésére. A middleware‑ek révén jogosultság‑ellenőrzést valósítottam meg, miközben sokat tanultam a modern webes hitelesítés gyakorlati kérdéseiről. Alperrel szoros együttműködésben dolgoztam, hogy a frontend–backend interfész egyértelmű legyen.

WinForms zenelejátszó: offline asztali kliensként egy Windows‑os lejátszót építettem fel a nulláról. A fő funkciók: lokális zenék feltöltése / drag‑and‑drop, lejátszás‑vezérlés (play/pause, előző/következő), állapot‑ és hangerő‑csúszka, sor törlése. A háttérben NAudio kezeli a lejátszást, a metaadatokat TagLib# olvassa ki. A legnagyobb kihívás az eseményvezérelt GUI és az aszinkron lejátszás dead‑lock‑mentes összehangolása volt, amit sikerült stabilan megoldanom.

Összességében egy megbízható autentikációs rendszert és egy felhasználóbarát asztali lejátszót adtam a csapatnak, ami jól illeszkedik a projekt teljes ökoszisztémájába, és megerősítette az érdeklődésemet a backend‑biztonság és a .NET‑alapú fejlesztés iránt.

## Továbbfejlesztési lehetőségek

Az Outclass jelenlegi formájában egy stabil, működőképes és használható zenelejátszó és -kezelő rendszer, viszont rengeteg olyan ötlet, lehetőség és irány van, amelyeket a jövőben érdemes lenne megvalósítani vagy legalább megfontolni. Az alábbiakban összegyűjtöttük azokat a területeket, amelyeken mi is látunk fejlődési potenciált – akár technikai, akár felhasználói élmény szempontjából.

#### **1. Natív mobilalkalmazás fejlesztése**

Jelenleg az alkalmazás nem rendelkezik külön mobilalkalmazással, csak böngészőn keresztül használható mobil eszközökön. A jövőben érdemes lenne kifejezetten Androidra és iOS-re fejleszteni egy natív mobilalkalmazást, amely jobban igazodik az adott platformhoz és a mobilhasználat igényeihez. Ez lehetővé tenné a gyorsabb működést, stabilabb élményt és számos olyan funkció beépítését, amely egy webes felületen nehezen megvalósítható. Például:

* **Offline mód** bevezetése, ahol a felhasználók letölthetik a zenéiket és internetkapcsolat nélkül is hallgathatják őket.
* **Push értesítések** integrálása (pl. új zene feltöltéséről, követett előadó új daláról).
* **Testreszabás lehetősége** a felhasználók számára (témák, színsémák, profilbeállítások).
* **Zeneajánló rendszer** beépítése, amely figyelembe veszi a hallgatási szokásokat és kedvenceket.
* **Képfeltöltés** profilhoz, lejátszási listákhoz.

#### **2. Webes felület bővítése**

A jelenlegi webes felület már reszponzív és könnyen használható, de még több interaktivitás és modern funkció bevezetése lehetséges:

* **Dark mode / világos-sötét téma váltás**, amit a felhasználók választhatnak.
* **Drag & drop funkciók** a lejátszási listák rendezéséhez.
* **Közösségi funkciók**, mint például kommentelés dalokhoz, értékelések, kedvenc lista megosztása más felhasználókkal.
* **Többnyelvűség** támogatása (például angol, magyar, német nyelv közötti váltás).

#### **3. Technikai háttér fejlesztése**

A technikai oldal is folyamatos fejlesztést kíván, főleg ha a rendszer később nagyobb felhasználószámot szolgál ki:

* **Jobban skálázható adatbázis** (pl. NoSQL megoldások vizsgálata vagy MySQL optimalizálása).
* **Streaming funkciók optimalizálása** a zenelejátszás gyorsabbá tételéhez.
* **Felhasználói tevékenységek naplózása** (pl. mikor mit hallgatott, statisztikák).
* **AI alapú ajánlórendszer** beépítése a hallgatási előzmények alapján.
* **Részletes adminfelület** kialakítása a tartalmak, felhasználók és jogosultságok kezelésére.

#### **4. Biztonság és adatvédelem továbbfejlesztése**

Noha már most is figyelünk az adatbiztonságra (pl. jelszavak titkosítása, HTTPS), a jövőben még komolyabb védelem kialakítása lenne indokolt:

* **Kétfaktoros hitelesítés (2FA)** lehetősége a bejelentkezésnél.
* **Adatvédelmi audit funkciók** (pl. mit tárolunk a felhasználóról, kérheti az adatai törlését).
* **Jogosultsági rendszer továbbfejlesztése**, külön adminisztrátor és moderátor szerepkörök bevezetésével.

#### **5. Új funkciók bevezetése**

Vannak olyan ötletek, amelyek a jelenlegi rendszer működését bővítenék ki új típusú tartalmakkal vagy lehetőségekkel:

* **Podcast-funkciók**: nem csak zenéket, hanem beszélgetéseket, epizódokat is lehetne feltölteni és hallgatni.
* **Élő stream lehetőség** előadóknak vagy DJ-knek.
* **Közös lejátszási listák** létrehozása barátokkal, amelyekbe többen is hozzáadhatnak zenéket.
* **Támogatási rendszer**, például lehetőség a kedvenc előadók "támogatására" kis összegű utalásokkal.

#### **6. Közösségi funkciók hozzáadása**

A közösségi élmény növelése érdekében bevezethetőek lennének különböző funkciók, amelyek segítik a felhasználókat abban, hogy kapcsolatba lépjenek egymással és megosszák zenei ízlésüket:

* Lehetőség lenne más felhasználók követésére, így mindenki nyomon követhetné, milyen zenéket hallgatnak barátaik, vagy milyen lejátszási listákat hoznak létre.
* Lejátszási listákat, dalokat vagy előadókat lehetne nyilvánosan megosztani, így egyes tartalmak akár "trending" kategóriába is kerülhetnének.
* Kommentelési lehetőséget is be lehetne vezetni, amivel a felhasználók visszajelzést adhatnának egymás tartalmaira, ezzel egy közösségibb élményt biztosítva.
* Kedvelések és értékelések alapján toplisták is generálhatóak lennének, ami ösztönözné a felhasználókat az aktívabb részvételre.
* Beépíthető lenne egy privát üzenetküldő vagy fórum jellegű rendszer is, ahol a felhasználók zenéről beszélgethetnének, vagy akár együttműködhetnének.

#### **7. Metaadatok automatikus felismerése**

A zenei fájlok feltöltésekor a felhasználók sokszor nem töltenek ki minden adatot (pl. cím, előadó, album, műfaj), vagy hibásan töltik ki azokat. Ezért érdemes lenne egy intelligens metaadat-felismerő rendszert beépíteni, amely:

* Automatikusan elemzi a feltöltött fájlokba ágyazott információkat (pl. ID3 tag-ek), és kitölti a megfelelő adatmezőket.
* Ha a feltöltött fájl nem tartalmaz elegendő információt, a rendszer online zenei adatbázisokat (pl. MusicBrainz, Last.fm, Spotify API) is lekérdezhetne, hogy kiegészítse vagy javítsa az adatokat.
* Az automatikus kitöltés mellett lehetőség lenne kézi módosításra is, hogy a felhasználók végső szinten kontrollálhassák a feltöltött tartalmat.
* Ez a funkció jelentősen javítaná a keresési élményt és a rendszer adatminőségét, valamint segítené a lejátszási listák és ajánlások pontosabb működését.

#### **8. Hangalapú keresés**

A felhasználói élmény további fejlesztése érdekében érdemes lenne bevezetni egy hangalapú keresési lehetőséget, amely lehetővé tenné a dalok, előadók vagy albumok azonosítását egy rövid hangrészlet alapján:

* A rendszer képes lenne felismerni egy rövid ének- vagy zenerészletet (akár dúdolva is), és ennek alapján azonosítani a dalt.
* Ez különösen hasznos lehet olyan esetekben, amikor a felhasználó nem emlékszik pontosan a címre vagy előadóra.
* Az algoritmus egy háttérrendszeren (például az ACRCloud vagy Shazam API-hoz hasonló megoldással) keresztül összevetné a hangmintát az adatbázisban szereplő dalokkal.
* A hangalapú keresés támogatná a gyorsabb és élvezetesebb interakciót, különösen mobilon vagy útközben.

#### **9. Okoseszköz-támogatás**

A rendszer jövőbeli bővítése kiterjedhetne különféle okoseszközök integrálására, így a felhasználók a megszokott zenehallgatási élményt más platformokon is élvezhetnék:

* Támogatás olyan eszközökhöz, mint a Google Home, Amazon Alexa vagy Apple HomePod, így hangutasításokkal lehetne zenét lejátszani, szüneteltetni vagy váltani.
* A webalkalmazás vagy a mobilalkalmazás kiterjeszthető lenne okostévék vagy médialejátszók számára, lehetővé téve a nappaliban történő zenehallgatást is.
* Az alkalmazás Bluetooth-on vagy Wi-Fi-n keresztül képes lenne automatikusan felismerni hordozható hangszórókat és okoseszközöket, és azokra irányítani a lejátszást.
* Az eszközök közötti zökkenőmentes váltás (pl. telefonról hangszóróra, majd vissza) szintén kényelmesebbé tenné a zenehallgatást.

# Felhasznált irodalom

* React hivatalos dokumentáció: <https://reactjs.org/docs/getting-started.html>
* Node.js API dokumentáció: <https://nodejs.org/en/docs>
* MySQL kézikönyv: <https://dev.mysql.com/doc/>
* bcrypt dokumentáció: <https://www.npmjs.com/package/bcrypt>
* Stack Overflow – JWT tokenkezelés példák és hibák kezelése: <https://stackoverflow.com/questions/39008071/how-to-store-jwt-token-in-http-only-cookie>
* Git dokumentáció: <https://git-scm.com/doc>
* Dev.to – REST API best practices: <https://dev.to/galymzhan/rest-api-best-practices-256>
* ChatGPT – dokumentációs javaslatok, technológiai magyarázatok
* MDN Web Docs – HTML/CSS/JavaScript referenciák  
   [**https://developer.mozilla.org/en-US/**](https://developer.mozilla.org/en-US/)
* Bcrypt hashelés – npm dokumentáció  
   [**https://www.npmjs.com/package/bcrypt**](https://www.npmjs.com/package/bcrypt)

# Ábrajegyzék

1. Ábra: A program telepítse
   1. A XAMPP-on elindítjuk az Apache-t és a MySQL-t 3. oldal
   2. Importáljuk a GitHub repository-ban található ‘outclass.sql’-t 3. Oldal
2. Ábra: Regisztráció 5. Oldal
3. Ábra: Bejelentkezés
   1. Zene keresés 6. oldal
   2. Zene lejátszás 6. Oldal
4. Ábra: Zene feltöltése 7. Oldal
5. Ábra: Részletes feladatspecifikáció, algoritmusok

* 1. Frontend 11. Oldal
  2. A következő módon hívjunk meg a függvényt: 11. Oldal
  3. Backend: A letöltés végpont a következő: 11.oldal

1. Ábra: Részletes feladatspecifikáció, algoritmusok
   1. Backend: A letöltés végpont a következő: 12. Oldal
   2. Felhasználói regisztrálás: frotend 12. Oldal
2. Ábra: Részletes feladatspecifikáció, algoritmusok
   1. Felhasználói regisztrálás: frotend 13. Oldal
   2. Felhasználói regisztrálás: backend 13. Oldal
3. Ábra: Részletes feladatspecifikáció, algoritmusok
   1. Felhasználói regisztrálás: backend 14. Oldal
   2. Tesztelési dokumentáció: Új felhasználó regisztrációja 14. Oldal
4. Ábra: Tesztelési dokumentáció
   1. Felhasználói bejelentkezés 15. Oldal
5. Ábra: Tesztelési dokumentáció
   1. Új zene feltöltése 16. Oldal
6. Ábra: Tesztelési dokumentáció
   1. Törölt zenékhez tartozó fájlok fizikai törlése 17. Oldal
7. Ábra: Asztali alkalmazás (WinForms zenelejátszó)
   1. Használat 19. Oldal