Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Hallgató:

Kovács Tamás

2025

Szegedi Tudományegyetem

Informatikai Tanszékcsoport

Webalkalmazás Zene- és Podcast Streaming, Közösségi Funkciók és Személyre Szabott Ajánlások Szolgáltatásával

Szakdolgozat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Készítette: |  | Témavezető: |  |
|  | Kovács Tamás |  | Hegedűs Péter |  |
|  | programtervező informatikus szakos hallgató |  | adjunktus |  |

Szeged

2025

FELADATKIÍRÁS

A szakdolgozat célja egy olyan közösségi webalkalmazás készítése, ami a zenét, illetve podcast hallgatok igényeit elégíti ki. Az alkalmazás funkciói kiterjedjenek a közösségi portálok funkcionalitásaira, úgymint a regisztráció, üzenetküldés, felhasználó keresés. Ezen felül a zenét, illetve podcast hallgatók számára fontos funkciókat is tartalmazza, mint például hangfájl feltöltése, lejátszási listák létrehozása, audió letöltése, fórum, audió kereső. A webalkalmazás az interneten legyen elérhető akár online, illetve offline állapotban, megtekintéséhez csupán egy webböngésző legyen szükséges.

Tartalmi összefoglaló

* ***A téma megnevezése***

Webalkalmazás Zene- és Podcast Streaming, Közösségi Funkciók és Személyre Szabott Ajánlások Szolgáltatásával

* ***A megadott feladat megfogalmazása***:

Egy olyan webalkalmazás készítése, amely közösségi funkciókkal bővítve lehetővé teszi a felhasználók számára a zenehallgatást, lejátszási listák kezelését, tartalmak letöltését, valamint fórumozást, üzenetküldést és kapcsolattartást más felhasználókkal. Az oldal legyen modern, reszponzív, felhasználóbarát, és képes legyen offline működésre is.

* ***A megoldási mód:***

Az alkalmazás fejlesztése Angular keretrendszer segítségével történt. Az adatok tárolását a Firebase Firestore végzi, a fájlkezeléshez Firebase Storage került használatra. A felhasználók hitelesítését szintén a Firebase Authentication szolgáltatása biztosítja. Az offline működés és helyi adattárolás megvalósítása IndexedDB-n keresztül történt. A reszponzív kezelőfelület és a valós idejű adatszinkronizáció együttesen teszik lehetővé az élményszerű felhasználást.

* ***Alkalmazott eszközök, módszerek:***

Angular keretrendszer

TypeScript programozási nyelv

Firebase (Firestore, Authentication, Storage)

IndexedDB helyi adattároláshoz

HTML5, CSS3

Visual Studio Code fejlesztői környezet

* ***Elért eredmények:***

Az elkészült alkalmazás lehetővé teszi a felhasználók számára a zenék valós idejű streamelését és letöltését, saját lejátszási listák kezelését, barátok hozzáadását, fórum- és chat funkciók használatát. A rendszer támogatja a PWA (Progresszív Webalkalmazás) telepítést és offline működést is, így modern, kényelmes és mobilbarát platformot biztosít a zenehallgatást kedvelő közösségek számára.

* ***Kulcsszavak:***

Angular, Firebase, IndexedDB, zenelejátszó, webalkalmazás, PWA, közösségi platform, chat, fórum

Tartalomjegyzék

[FELADATKIÍRÁS 1](#_Toc198759662)

[Tartalmi összefoglaló 2](#_Toc198759663)

[Tartalomjegyzék 4](#_Toc198759664)

[BEVEZETÉS 6](#_Toc198759665)

[1. Technológiák bemutatása 7](#_Toc198759666)

[1.1. Programnyelvek, technológiák 7](#_Toc198759667)

[1.1.1. Angular 7](#_Toc198759668)

[1.1.2. CSS 8](#_Toc198759669)

[1.1.3. Firebase és Firebase Storage 8](#_Toc198759670)

[1.1.4. IndexedDB 9](#_Toc198759671)

[1.2. Fejlesztői környezet 9](#_Toc198759672)

[2. SPECIFIKÁCIÓ 10](#_Toc198759673)

[2.1. Felhasználói felület 11](#_Toc198759674)

[2.1.1. Zenék 11](#_Toc198759675)

[2.1.2. Lejátszó 11](#_Toc198759676)

[2.1.3. Keresés 11](#_Toc198759677)

[2.1.4. Lejátszási listák 11](#_Toc198759678)

[2.1.5. Feltöltés 11](#_Toc198759679)

[2.1.6. Regisztráció és bejelentkezés 12](#_Toc198759680)

[2.1.7. Profil 12](#_Toc198759681)

[2.1.8. Fórum 12](#_Toc198759682)

[2.1.9. Ismerősök kezelése 13](#_Toc198759683)

[2.1.10. Chat 13](#_Toc198759684)

[2.1.11. Navigáció 13](#_Toc198759685)

[2.1.12. Offline felület 13](#_Toc198759686)

[2.1.13. Dizájn 13](#_Toc198759687)

[2.2. Adminisztrációs felület 14](#_Toc198759688)

[2.2.1. Adminisztrációs jogosultságok kezelése 14](#_Toc198759689)

[2.2.2. Fórum moderálás 14](#_Toc198759690)

[2.2.3. Dizájn 14](#_Toc198759691)

[3. tervezés 15](#_Toc198759692)

[3.1. Képernyőtervek 15](#_Toc198759693)

[3.1.1. A felhasználói felület 15](#_Toc198759694)

[3.1.2. Az adminisztrációs felület 22](#_Toc198759695)

[3.2 Az adatbázis megtervezése 23](#_Toc198759696)

[3.2.1. Adatbázisséma és diagramok 23](#_Toc198759697)

[4. Megvalósítás 27](#_Toc198759698)

[4.1. Regisztráció 27](#_Toc198759699)

[4.2. Bejelentkezés 28](#_Toc198759700)

[4.3. Kinézet 28](#_Toc198759701)

[4.4. Zene lejátszása 28](#_Toc198759702)

[4.5. Főoldal 29](#_Toc198759703)

[4.6. Lejátszi listák 30](#_Toc198759704)

[4.7. Fórum 30](#_Toc198759705)

[4.8. Feltöltés 31](#_Toc198759706)

[4.9. Keresés 32](#_Toc198759707)

[4.10. Felhasználó 34](#_Toc198759708)

[4.11. Ismerősök/követések 35](#_Toc198759709)

[4.12. Chat 37](#_Toc198759710)

[4.13. Admin 37](#_Toc198759711)

[4.14. Offline főoldal 38](#_Toc198759712)

[5. Tesztelés 39](#_Toc198759713)

[5.1. A tesztelés menete 39](#_Toc198759714)

[5.2. Tesztesetek 39](#_Toc198759715)

[5.3. Összegzés 43](#_Toc198759716)

[Összefoglalás 44](#_Toc198759717)

[Irodalomjegyzék 45](#_Toc198759718)

[NYILATKOZAT 46](#_Toc198759719)

[KÖSZÖNETNYILVÁNÍTÁS 47](#_Toc198759720)

BEVEZETÉS

Az elmúlt években a digitális zenehallgatás és podcast-fogyasztás mindennapjaink szerves részévé vált, amiben az internet kiemelkedő szerepet játszott. Míg korábban fizikai adathordozókon vagy rádión keresztül hallgattuk kedvenceinket, ma már néhány kattintással elérhetjük.

Az internet nyújtotta lehetőségeket kihasználva egyre több előadó, podcaster, kiadó és független alkotó osztja meg műveit különböző platformokon. Ezek között, azonban könnyű elveszni, ezért jött létre a ZeeSharing, amely célja, hogy a felhasználók egy helyen, kényelmesen böngészhessenek a zenék és podcastek között, felfedezve új kedvenceket és visszatérve a már jól ismert tartalmakhoz.

A ZeeSharing elsődleges célja, hogy a felhasználó könnyen elérhesse az éppen trendi, népszerű vagy frissen megjelent zenéket, podcast epizódokat. A lejátszás mellett lehetőség van lejátszási listák létrehozására, követésére, sőt akár offline letöltésre is. Így a felhasználó bármikor visszatérhetnek a korábban elmentett tartalmakhoz anélkül, hogy internethez lennének csatlakozva.

A böngészés cím, kategória, előadó, podcaster alapján keresi meg a kívánt tartalmat, ami nagyban megkönnyíti a tartalom elérését. A regisztrált felhasználók személyes oldalukon nemcsak kedvenceiket követhetik nyomon, hanem lejátszási listákat hozhatnak létre, amit megoszthatnak más felhasználókkal is. Illetve a felhasználó nyomon követheti az eddig meghallgatott zenéket, illetve podcasteket grafikonok segítségével.

A ZeeSharing másik célja egy aktív közösség kiépítése is. Az előadok, valamint podcasterek a fórumon bejelenthetik új albumjaikat, illetve közelgő eseményeiket is. Lehetőség van egymást követni, illetve ismerősnek jelölni, privát üzeneteket küldeni, tartalmat ajánlani, és egymás lejátszási listáit is megtekinteni, ha azok nyilvánosak. A kapcsolati rendszer átlátható: követések, visszaigazolások, függő kérések külön szekcióban láthatóak.

Az előadók és podcastére számára külön kezelőfelület áll rendelkezésre, ahol új tartalmakat tölthetnek fel, illetve kiválasztják a feltölteni kíván elem kategóriáját is, illetve joguk van a fórum oldalra bejegyzéseket írni. Az adminisztrátorok külön felületen tudnak felhasználókat kezelni, illetve fórum bejegyzéseket törölni.

Összefoglalva, a ZeeSharing egy olyan modern, közösségi alapokra épülő zene- és podcast platform, amely nemcsak a hallgatás élményét biztosítja, hanem lehetőséget nyújt az alkotók és hallgatók közötti aktív, inspiráló kapcsolatteremtésre is.

1. Technológiák bemutatása

1.1. Programnyelvek, technológiák

1.1.1. Angular

Az Angular egy modern, nyílt forráskódú webalkalmazás-fejlesztési keretrendszer, amelyet eredetileg a Google fejlesztett ki, és 2016-ban jelent meg az első stabil változata Angular 2 néven. Bár a fejlesztés kezdete az AngularJS-hez köthető, az új Angular gyökeresen eltér elődjétől: TypeScript nyelven íródott, és a legmodernebb komponensalapó architektúrát követi. A keretrendszert a Google folyamatosan frissíti és karbantartja, így biztosítva a hosszú távú támogatást, valamint a technológiai fejlődéssel való lépéstartást.

Az Angular célja, hogy egységes megközelítést kínáljon az összetett, nagy volumenű frontend alkalmazások fejlesztésére. Már a kezdetektől beépített eszköztárat biztosít a komponensalapú struktúrához, űrlapkezeléshez HTTP-kommunikációhoz, oldalnavigációhoz, teszteléshez és dependency injection-höz – vagyis olyan funkciókhoz, amelyek más keretrendszerekben csak külső osztálytárakkal oldhatók meg.

Az Angular választása számomra egyértelmű volt, mert professzionális szintű, jól dokumentált, és hosszú távon is fenntartható keretrendszert kínál. A hivatalos dokumentáció minden fogalomhoz világos példákat és részletes leírást biztosít. Emellett az aktív fejlesztői közösség folyamatosan bővíti a tudásbázist fórumokon, blogokon és GitHubon keresztűl.

Az Angular teljesen platformfüggetlen, így asztali, mobil és webes alkalmazások készítésére egyaránt alkalmas. Kiválóan együttműködik a Firebase-szel, REST API-kkal, és bármilyen backend technológiával. A szintaxisa egységes, TypeScript alapú, ami segíti az automatikus típusellenőrzést, ezáltal csökkenti a hibák számát és növeli a kód megbízhatóságát. A projekt buildelése és futtatása egy egyszerű böngészővel megoldható, fordítás csak a fejlesztés során szükséges. Támogatja a PWA (Progresszív Webalkalmazás) fejlesztést, offline használatot, és lehetőséget kínál natív asztali alkalmazás készítésére Electronnal. [3, 7, 8, 14, 5]

1.1.2. CSS

A weboldal megjelenítéséhez a CSS (Cascading Style Sheets) stílusleíró nyelvet használtam, amely lehetővé teszi a tartalom és a forma szétválasztását. Különös figyelmet fordítottam a színek összehangjára, az elemek rendezett elhelyezésére, valamint az űrlapok és navigációs elemek esztétikus kialakítására. A stílusokat külső CSS fájlokba szerveztem, külön az egyes felületek számára, így a módosításokat egyszerűen, központilag lehetett elvégezni. A reszponzív megjelenítéssel biztosítottam, hogy a weboldal különböző eszközökön – mobilon, tableten, és asztali gépen – is megfelelően jelenjen meg. A modern CSS lehetőségei, mint a flexbox és az animációk, tovább javították a felhasználói élményt és a vizuális megjelenítést. [6, 9]

1.1.3. Firebase és Firebase Storage

A Firebase a Google által fejlesztett, felhőalapú fejlesztői platform, amely modern webes és mobilalkalmazások háttérrendszerének kialakításához nyújt komplett megoldást. Számos szolgáltatást biztosít, például valós idejű adatbázist, felhasználói hitelesítést, fájltárolást, üzenetküldést és analitikát, ezzel jelentősen megkönnyítve a fejlesztők munkáját. A Firebase erőssége a skálázhatóság, a biztonságos és gyors működés, valamint az, hogy a szolgáltatások szorosan integrálódnak egymással és könnyedén használhatók modern frontend keretrendszerekkel, mint például az Angular.

A projekt során elsősorban két Firebase szolgáltatást alkalmaztam: a Cloud Firestore NoSQL adatbázist és a Firebase Storage fájltárolót.

A Firestore egy dokumentum alapú adatbázis, amely lehetővé teszi strukturált adatok valós idejű tárolását és lekérdezését. Az adatokat gyűjteményekbe és dokumentumokba szervezve tárolja, ahol minden dokumentum kulcs-érték párokat tartalmaz. Előnye, hogy valós idejű szinkronizációt biztosít. Ez különösen hasznos volt a projekt dinamikus részeinél, például a fórum vagy a chat funkciók esetén. Emellett lehetőség van összetett lekérdezésekre, rendezésre, szűrésre, illetve egyéni jogosultsági szabályok beállítására is.

A Firebase Storage egy biztonságos, Google Cloud Storage alapú fájltároló megoldás, amelyet képek, zenék, videók és egyéb fájlok hatékony kezelésére fejlesztettek ki. A projekt során ezt a szolgáltatást használtam a felhasználók által feltöltött médiafájlok tárolására. A Storage SDK segítségével egyszerűen lehet feltölteni, letölteni vagy törölni a fájlokat közvetlenül a kliensoldali alkalmazásból. A két szolgáltatás közötti integráció lehetővé tette, hogy a feltöltött fájlok elérési útját elmentsem az adatbázisba, így az alkalmazás részein könnyedén hivatkozhattam ezekre. [1,3]

1.1.4. IndexedDB

Az IndexedDB egy beépített, kliensoldali adatbázis, amelyet a modern böngészők biztosítanak, és lehetővé teszi nagy mennyiségű strukturált adat tárolását offline módban is. Az IndexedDB segítségével objektumalapú kulcs-érték párokat menthetünk, ami ideálissá teszi összetettebb webalkalmazások számára, ahol a gyors elérés és az offline működés is elvárás. Az adatok aszinkron módon kerülnek elérésre, így nem blokkolják a fő szál működését, ami különösen fontos felhasználói élmény szempontjából.

A technológiát elsősorban olyan funkciókhoz használtam, ahol fontos volt a gyors helyi adattárolás és elérés, például letöltött médiafájlok, kezelése során. Az IndexedDB használatával a felhasználó akkor is hozzáférhet a korábban letöltött adatokhoz, ha éppen nincs internetkapcsolat. A strukturált tárolás és a tranzakcióalapú működés biztonságos adatkezelést tesz lehetővé. Támogatása széles körű, minden modern böngészőben elérhető, így PWA (Progresszív Webalkalmazás) fejlesztés során is nélkülözhetetlen eszközként szolgál. [4, 5]

1.2. Fejlesztői környezet

Fejlesztés és tesztelés során fontos, hogy mindig a számunkra megfelelő eszközzel dolgozzunk.

Fejlesztői környezet:

* Visual Studio Code 1.100

A projekt fejlesztéséhez a Microsoft által fejlesztett Visual Studio Code nevű, platformfüggetlen, nyílt forráskódú szerkesztőt használtam. A program kiválóan alkalmas webes fejlesztésre, mivel rendkívül rugalmas, bővíthető, és számos hasznos funkciót kínál, mint például az automatikus kódkiegészítés (IntelliSense), beépített terminál, Git integráció és szintaktikai kiemelés különböző nyelvekhez, köztük HTML, CSS, JavaScript, TypeScript, Angular, PHP és JSON formátumhoz. A fejlesztés során nagy előnyt jelentett, hogy könnyen kezelhettem benne a teljes projektstruktúrát, és az Angular alkalmazás moduláris felépítését átlátható módon tudtam kezelni.

Teszteléshez:

* Google Chrome 136
* Mozilla Firefox 138
* Microsoft Edge 136

A fejlesztés során elengedhetetlen volt az alkalmazás több böngészőben történő tesztelése, hogy biztosítsam a kompatibilitást és egységes megjelenést. A Chrome-hoz elérhető Angular DevTools segített az egyes komponensek és adatok valós idejű elemzésében. A különböző böngészők beépített fejlesztői eszközeivel (DevTools) lehetőségem volt JavaScript és TypeScript hibák nyomkövetésére, CSS vizsgálatra, hálózati forgalom elemzésére, valamint a reszponzív nézetek tesztelésére is.

Egyéb eszközök:

* GIMP 3: saját logó szerkesztése
* Firebase console: az adatbázis és tárolók kezelésére
* Git + GitHub: verziókezeléshez és távoli tároláshoz

2. SPECIFIKÁCIÓ

Évek óta foglalkozom webes alkalmazások fejlesztésével hobbiként, és régóta megfogalmazódott bennem az igény egy olyan komplex közösségi platform létrehozására, amely a zenehallgatás élményét ötvözi a közösségi funkciókkal. Tapasztalataim alapján úgy gondolom, hogy a legfontosabb követelmény egy ilyen alkalmazással szemben az, hogy gyors, megbízható és felhasználóbarát legyen, emellett pedig kínáljon olyan egyedi szolgáltatásokat, amelyek miatt a felhasználók rendszeresen visszatérnek, és aktívan használják azt. Ennek megfelelően részletesen megterveztem azokat a funkciókat, amelyeket egy modern zenelejátszó és közösségi platformnak nyújtania kell.

2.1. Felhasználói felület

2.1.1. Zenék

Az oldal központi része, ahol a felhasználók megtekinthetik a legújabb és legnépszerűbb zenéket, illetve podcasteket. Illetve itt még személyre szabott ajánlásokat is kap a felhasználó a meghallgatott tartalmat figyelembe véve.

2.1.2. Lejátszó

A lejátszó minden oldalon megtalálható a regisztráció és bejelentkezésen kívül. A lejátszó biztosítja az alapvető funkciókat a hangfájlokkal kapcsolatban. Köztük a lejátszás, megállítás, következő/előző lejátszása, randomizálás, letöltés, kikapcsolási időzítő, hangerő szabályzás, illetve lejátszási listához adás.

2.1.3. Keresés

A zene és közösségi tartalmak keresését egy keresősáv segíti. A keresés azonnal listázza a találatokat cím, előadó, címke vagy felhasználónév alapján. A találatok rákattintásával a kereső kilistázza a keresett tartalom legtöbb információját. Köztük az előadó és podcaster adatai, a felhasználó adatai és nyilvános lejátszási listái, illetve a címkével megjelölt hangfájlok. A kilistázott adatok segítségével követhetünk be előadókat, jelölhetünk ismerősöket, menthetünk lejátszási listákat, illetve indíthatunk el albumokat.

2.1.4. Lejátszási listák

A felhasználók egyéni lejátszási listákat hozhatnak létre, amelyekbe tetszőlegesen hozzáadhatnak zenéket. A lista láthatósága bármikor állítható nyilvános, csak barátok és privát állapotok között. Illetve lehetőség van más felhasználó lejátszási listáját is elmenteni, ha annak a láthatósága megengedi.

2.1.5. Feltöltés

Regisztrált felhasználók saját zenéket is feltölthetnek. A feltöltés során kötelezően meg kell adni:

* Cím
* Címke(k)
* Hangfájl
* Opcionális borítókép

Amennyiben a felhasználó nem akar képet feltölteni, akkor a feltöltés során automatikusan lementésre kerül a hangfájlba beágyazott kép, amennyiben az elérhető, azonban, ha nincs, akkor egy alapértelmezett kép kerül be borítóképként. A rendszer automatikusan menti a feltöltési idő és a megadott adatokat.

2.1.6. Regisztráció és bejelentkezés

A portál teljes funkcionalitásához regisztráció szükséges. A regisztráció során a felhasználónév, jelszó (kétszeri beírás), e-email cím és a fiók típusának megadása kötelező. Ezeket az adatokat validálni kell: a felhasználónév és az e-email cím egyediségét az adatbázis ellenőrzi. A jelszavak biztonságos, visszafejthetetlen tárolással kerülnek elmentésre. Bejelentkezésnél a felhasználónak megkell adnia az e-mail címét és a jelszavát, majd ezek validálása után szabadon használhatja az oldalt.

2.1.7. Profil

A felhasználók a profiljukon keresztül módosíthatják jelszavukat, profilképüket, valamint áttekinthetik saját listáikat, ismerőseiket, követéseiket és zenei statisztikáikat. A nyilvános profil csak a felhasználónevüket és zenelistáikat mutatja, email-cím vagy bármely privát adat nem jelenik meg. A hallgatási szokásokat a rendszer automatikusan naplózza, és az alábbi kimutatásokban jeleníti meg:

* 7 leggyakrabban hallgatok előadó
* 7 leggyakoribb címke
* Az elmúlt héten a legtöbbet hallgatott előadó/címke

2.1.8. Fórum

A fórum célja, hogy a felhasználók megosszák egymással a zenei élményeiket, kérdéseiket, javaslataikat. A fórum automatikusan cenzúrázza a közzétett szöveget egy adatbázis segítségével, így a tiltott szavak akármikor bővíthetőek. A bejegyzések dátum szerint csökkenő sorrendben listázódnak. A hozzászólások szerkesztése még nem engedélyezett, csak az admin törölhet. A bejegyzésekhez kép is csatolható, ha valaki például egy új album borítóját akarná megosztani.

2.1.9. Ismerősök kezelése

A felhasználók ismerősnek jelölhetik egymást, visszajelölhetnek, vagy elutasíthatják a kérést. A kapcsolatok állapota (kért, visszaigazolt) adatbázisban tárolódik. Az ismerősök külön listában jelennek meg, ahol, ha kiválasztanak egy felhasználót akkor beszélgetést folytathatnak egy külön chat felület segítségével.

2.1.10. Chat

Privát üzenetküldési lehetőség is elérhető. Bármely regisztrált felhasználó írhat bármély másiknak, ismeretségtől függően. A beérkezett és elküldött üzenetek különféle módon tárolódnak az adatbázisban, az új, nem olvasott üzenetek érkezése esetén jelzés jelenik meg a menüfülön. Lehetőség van privát üzenetben zenére is hivatkozni, amit a másik fél bármikor lejátszhat.

2.1.11. Navigáció

Az oldal navigálásában egy navigációs bár segít. Annak segítségével szabadon válthatunk a megjelenítendő felületetek között. A navigációs fülön még megtalálható egy értesítési funkció is, ami akkor jelenik meg, ha a felhasználónak olvasatlan üzenetei vannak.

2.1.12. Offline felület

Amennyiben nincs internet kapcsolat bejelentkezésnél lehetőség van az offline használatra. Ilyenkor az oldal átnavigál minket egy offline felületre, ahol szabadon hallgathatjuk az eddig letöltött zenéket vagy podcasteket. Letöltésnél tárolva van a hangfájl mellett a kép is, hogy azzal is könnyebben tudja azonosítani a felhasználó a lejátszani kívánt zenét vagy podcastet. Ezen a felületen nem érhetőek el az internetet igénylő funkciók.

2.1.13. Dizájn

Az alkalmazás felhasználói felületének átláthatónak, letisztultnak és esztétikusnak kell lennie, hogy kellemes élményt nyújtson a felhasználóknak. Mivel az oldal zenei és közösségi témájú, a színvilág lehet vidám és inspiráló, de kerülnie kell a túl élénk, vibráló színeket, amelyek zavarhatják az olvasást. A betűtípus legyen egyszerű és jól olvasható, a tartalmak pedig logikusan tagoltak, áttekinthetők. A logónak tükröznie kell a platform zenei jellegét, egyben legyen könnyen felismerhető és modern. A stílusokat külső CSS fájlokban célszerű kezelni, hogy egységes és reszponzív megjelenést biztosítsunk. Egy kis méretű, jól felismerhető favicon is szükséges, amely a böngészőfüleken és a telepített alkalmazás ikonjában is jól mutat.

2.2. Adminisztrációs felület

2.2.1. Adminisztrációs jogosultságok kezelése

Az adminisztrációs felület elérhető minden adminisztrátori joggal rendelkező felhasználónak. Az adminok listázhatják az összes regisztrált felhasználót, ahol lehetőség van:

* Admin jogosultság hozzáadására (felhasználónév alapján)
* Felhasználó törlése
* Admin jogosultság törlésére

A rendszer figyelmeztetést küld, ha egy admin a saját jogosultságát próbálja törölni. Az admin státusz tárolása az adatbázisban történik. Amennyiben egy felhasználót töröl az adminisztrátor, akkor annak a felhasználónak minden adata és feltöltött fájlja törlődik, kivéve a hallgatási előzménye, illve az email-cím, hogy megakadályozzuk újbuli regisztrációját.

2.2.2. Fórum moderálás

A fórum összes bejegyzése és hozzászólása listázható admin felületen. Minden hozzászólás mellett elérhető a „Törlés” gomb, amely véglegesen eltávolítja azt a fórumból. A törlésről nem küld automatikus értesítést. A fórum bejegyzés törlése a fórum felületén is elérhető.

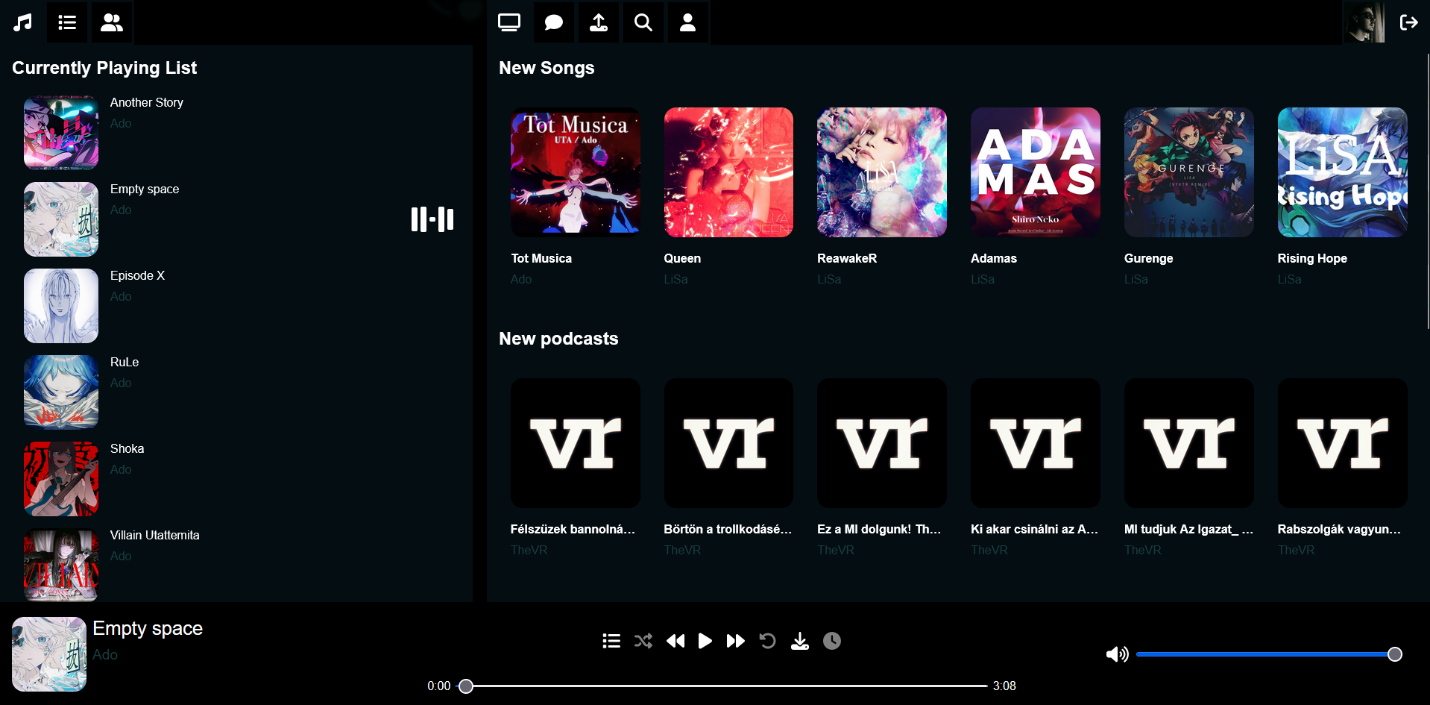
2.2.3. Dizájn

Az admin felület dizájnja egyezzen meg a felhasználói felületével az egységes dizájn érdekében.

3. tervezés

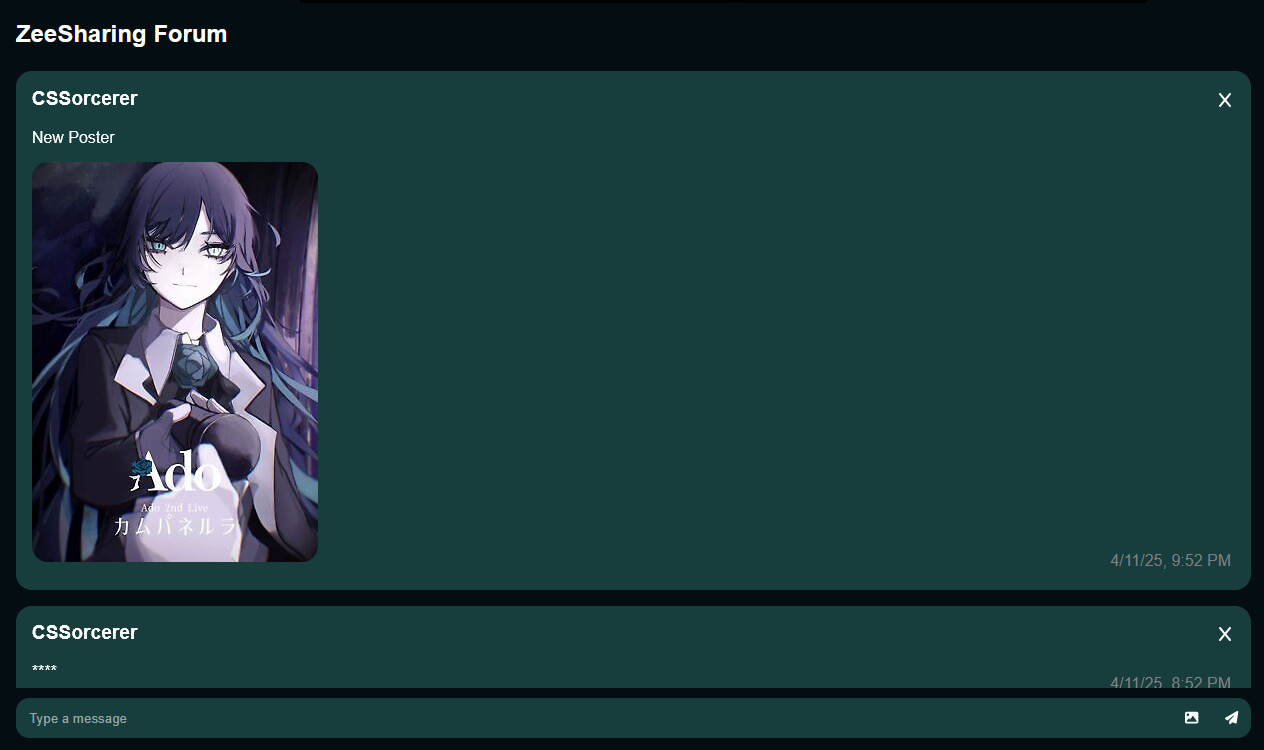
3.1. Képernyőtervek

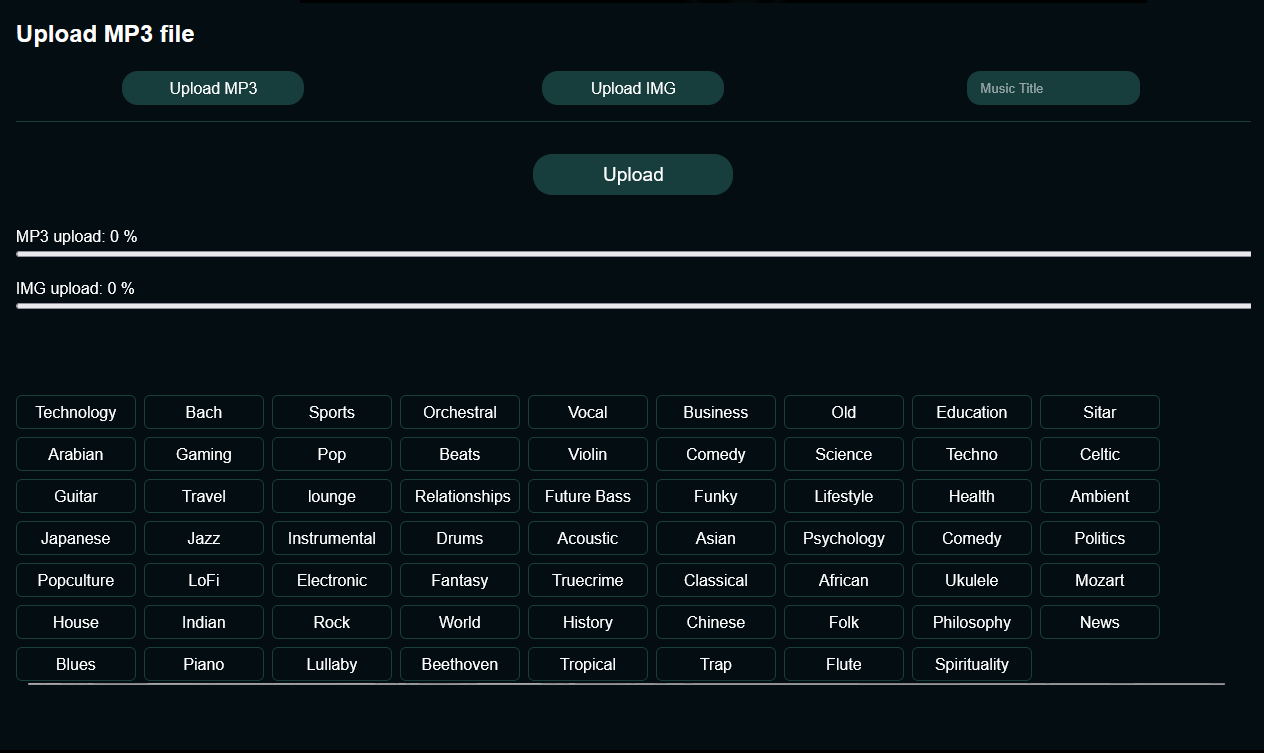
3.1.1. A felhasználói felület

A regisztrált felhasználók számára belépés után elérhetővé válik az egész oldal, a főoldallal (3.1 ábra) kezdve. A főoldalon megjelenik a navigációs fül, ahol az oldal egyes elemi között tudnak lépkedni. Illetve alaphelyzetben megjelenik a ,,NEWS” felület, ahol meglehet tekinteni a legújabb feltöltéseket, illetve a személyes ajánlásokat. Az oldalon látható még az abban a pillanatban lejátszott lejátszási lista is.

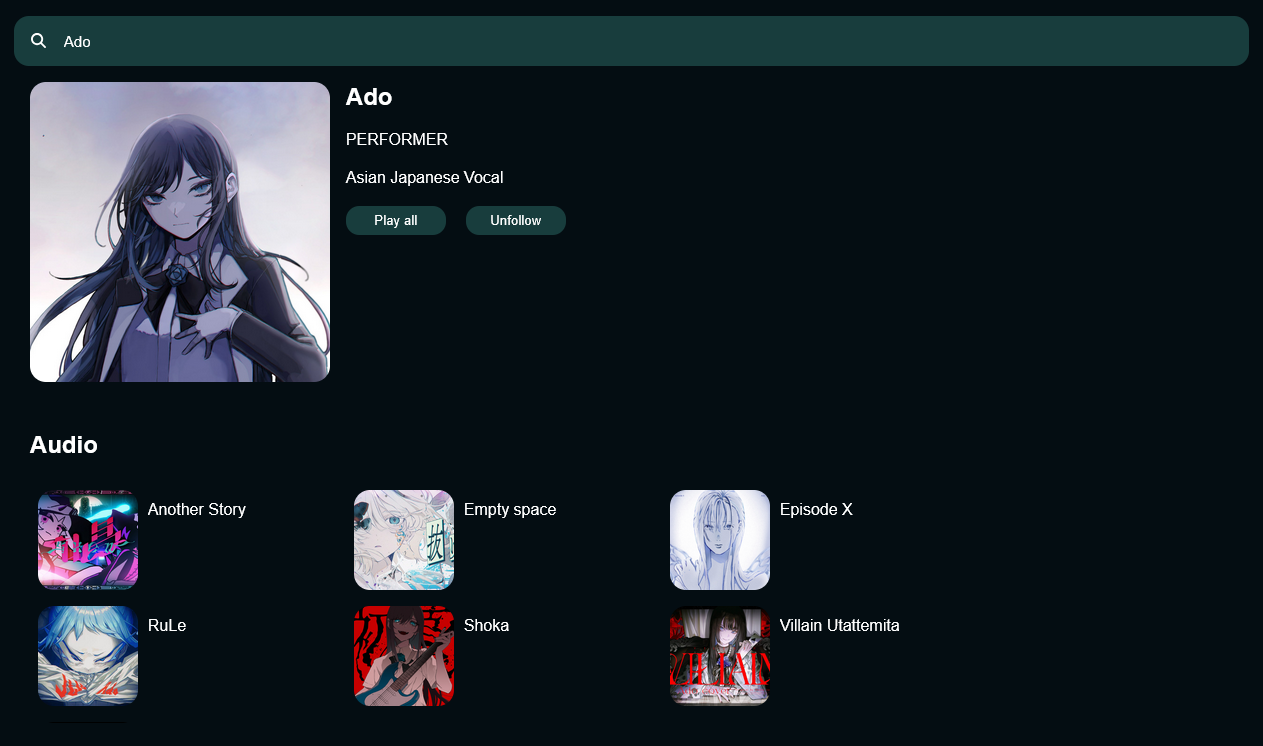
3.1 ábra – Kezdőlap

A ,,Forum” menüpont megjeleníti a fórum felületet. Itt látható a bejegyzések létrehozójának a felhasználóneve, a bejegyzés létrehozásának a dátuma, illetve maga a bejegyzés. Bejegyzésként kép is megadható, amit a bejegyzés alján találunk meg. Illetve még admin felhasználók számára egy törlés gomb is megjelenik. (3.2 ábra)

3.2 ábra – Fórum

 A ,,Feltöltés” felület segítségével a felhasználók feltölthetik zenéiket, illetve podcastjeiket. A felületen megtalálható a hangfájl kiválasztása, a kép kiválasztási, a fájl címe, illetve maga a feltöltés gomb. Ezek alatt látható egy sáv, ami a fájlok feltöltésének állapotát mutatja. Az oldal alján a feltöltő kiválaszthatja a megfelelő címkéket, a kiválasztott címkék más háttérrel jelennek meg. (3.3 ábra)

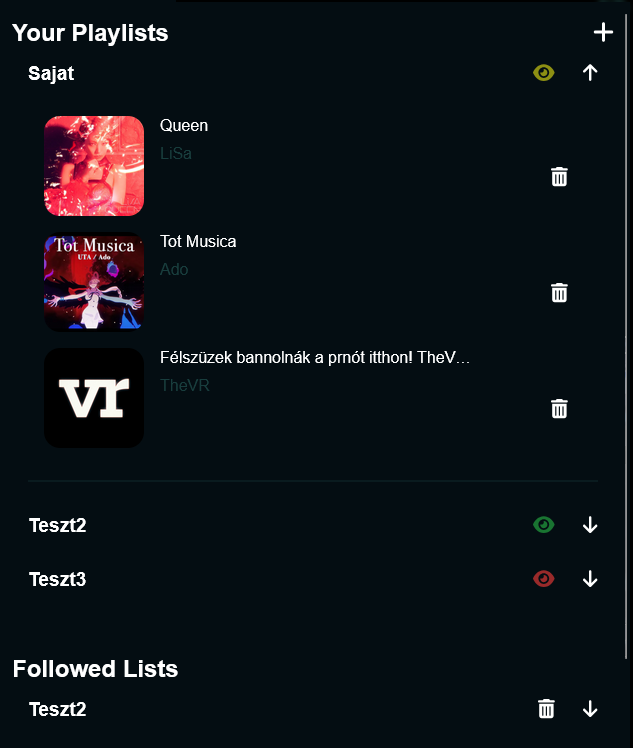
3.3 ábra – Feltöltés

 A „Keresés” felületen a felhasználó könnyedén kereshet rá a kívánt információkra. Az oldal tetején megtalálható egy kereső sáv, amibe, ha minimum 2 karaktert begépel, akkor megjelenik alatta egy keresési lehetőségek felület, ahol az adott szövegre kapott találatok találhatók csoportosítva. Miután kiválasztotta a keresett elemet, megjelenik annak az elemnek az adatai. Például az előadó profilképe, neve, leggyakrabban használt címkéi, illetve a feltöltött zenéi. Illetve 2 gomb is megjelenik, amik segítségével bekövethetjük az előadót, illetve lejátszhatjuk az összes feltöltését. (3.4 ábra)

3.4 ábra – Keresés

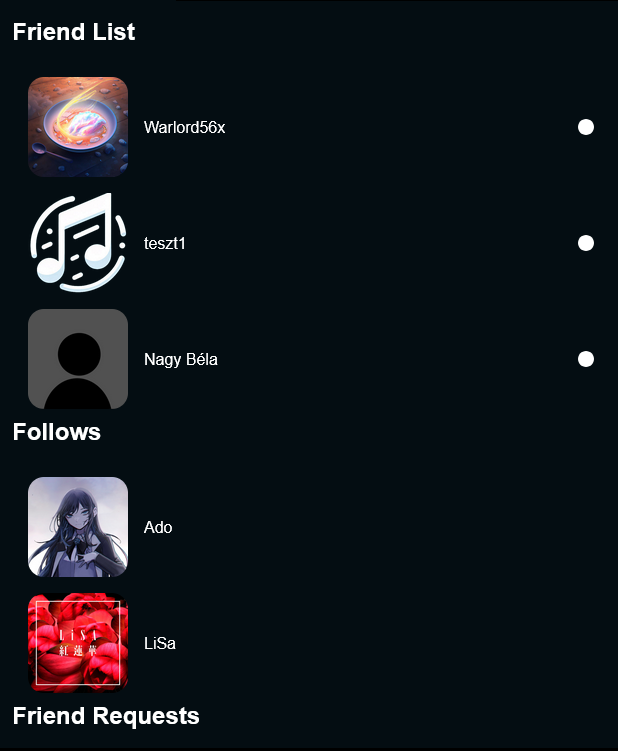
A „Felhasználó” felület segítségével a felhasználó könnyen kezelheti fiókját, illetve láthatja hallgatási adatait. A tetején 2 gomb található, amivel meglehet változtatni a jelszót, illetve új profilképet lehet beállítani. Azok alatt 3 lenyíló lista van, amik segítségével a felhasználó törölhet ismerősöket, jelöléseket, lejátszási listákat, követéseket. A felület alján 4 grafikon található, amik az mutatják, hogy az adott előadókat/címkéket milyen gyakran hallgatták. (3.5 ábra)

3.5 ábra – Felhasználó

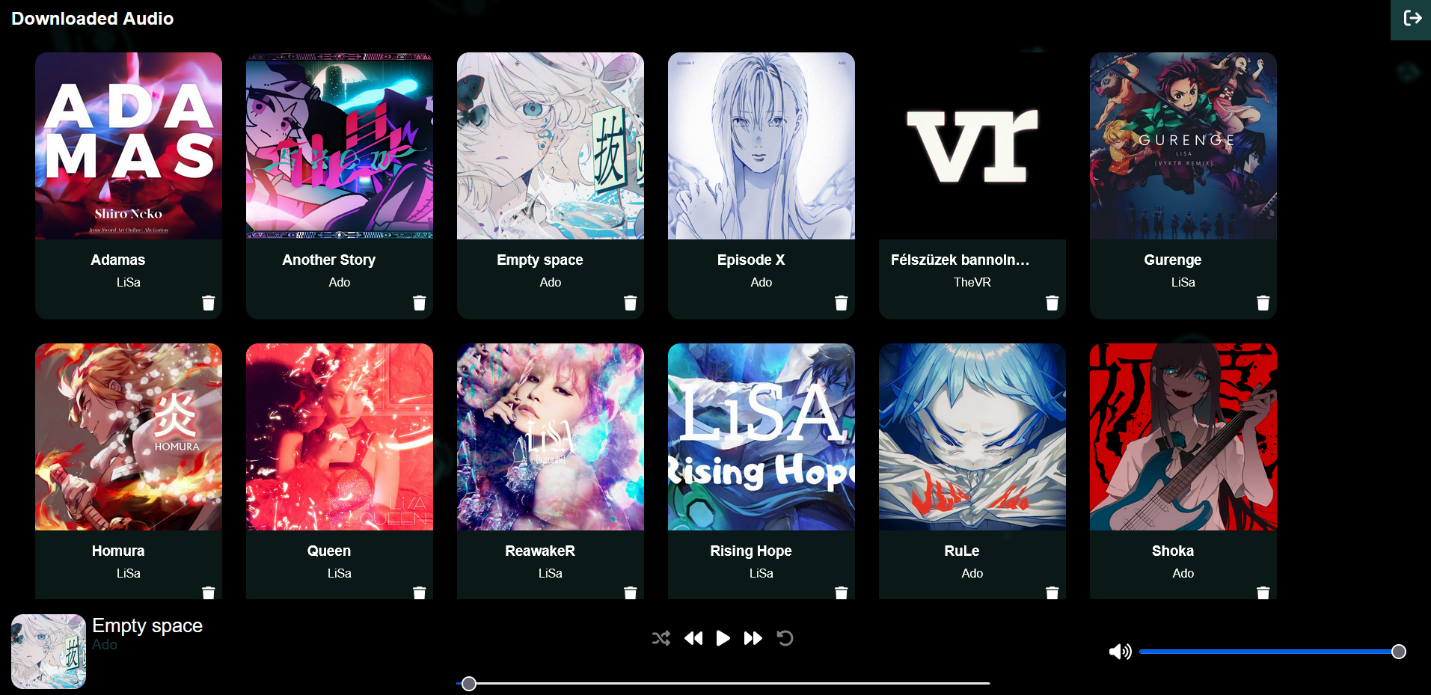
 A „Lejátszási listák” felület segít a felhasználónak lejátszási listáit kezelni, mint például új létrehozása, láthatóság megváltoztatása, zene törlése a listából, követett listák törlése. A listák lenyithatóak, és a lenyílt zenéket bármikor törölheti, vagy lejátszhatja. Még látható egy szem ikon is, ami a lista láthatóságát mutatja, rákattintva azt megváltoztatja. (3.6 ábra)

3.6 ábra – Lejátszási listák

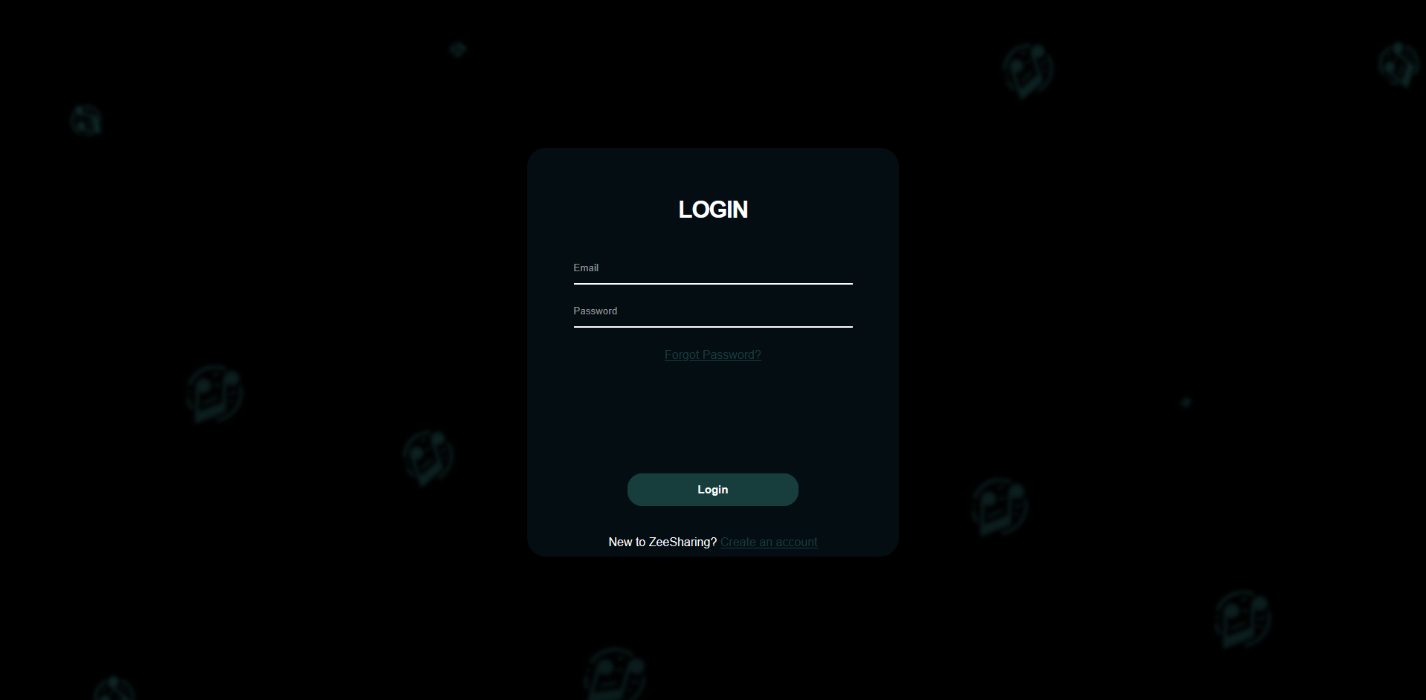
Az „Ismerősök” felület felsorolja a felhasználó ismerőseit, követéseit, illetve ismerősnek jelölésit. Az ismerősöknél van egy kis karika, ami, ha zölden jelenik meg, akkor azt mutatja, hogy a felhasználónak olvasatlan üzenete van attól az ismerőstől. Az ismerősre kattintva megjelenik a chat felület. Itt található még a követések is, ami segítségével könnyen tudunk navigálni az abban a pillanatban követett előadók között. Egy fontos elem, hogy az ismerősnek jelöléseket eltudjuk fogadni, így azok is itt vannak felsorolva. Egy zöld pipa vagy piros x segítségével tudjuk elfogadni, illetve elutasítani a kérelmet. (3.7 ábra)

3.7 ábra – Ismerősök

Az „Offline főoldal” oldal csak internet nélkül érhető el. Az oldalon csak a letöltött zenék és podcastek vannak felsorolva. Minden elemet lehet törölni egy kuka ikon segítségével, ekkor a lista automatikusan frissül. Az alap lejátszási funkciók itt is működnek (lejátszás, előző, következő, újra, random). (3.8 ábra)

 3.8 ábra – Offline főoldal

A „Bejelentkezés”, illetve „Regisztráció” oldalak egy dizájn alapon osztoznak. Középen található az összes input mező, és ezek kitöltése kötelező. Fontos, hogy a jelszó típusú mezőt titkosítva lehessen kitölteni, illetve, hogy az oldal nem megy tovább, ha egy adat nem megfelelő. A bejelentkező és regisztráló oldal között link segítségével lehet közlekedni. Az üres dizájnú háttér lebegő ZeeSharing logókkal lett feltöltve. (3.9 ábra)



3.9 ábra – Bejelentkezés

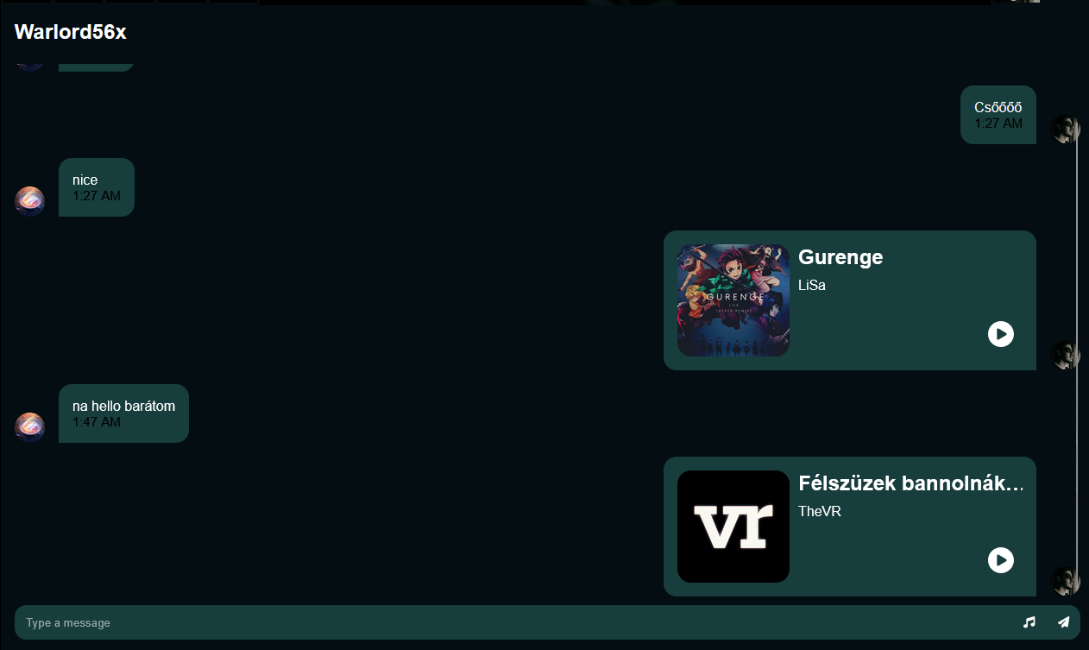
A „Lejátszó” felület biztosítja az alapvető funkciókat a hangfájlok lejátszásához. A felület bal oldalán láthatóak az éppen lejátszott fájl adatai (kép, cím, előadó). A felület jobb oldalán a hangerő sáv található, amivel a lejátszás hangerejét lehet állítani, illetve egy gomb, amivel teljesen lelehet némítani. A közepén egy sáv látható, ami jelöli, hogy jelenleg hol is tart a lejátszás. Illetve még középen találhatóak az alapvető funkciók is, ezek balról jobbra haladva (3.10 ábra):

* Lejátszási listához adás: megjelenik egy lista, ahol kiválasztható, hogy melyik listához adjuk hozzá a zenét
* Véletlen lejátszás: megnyomásával véletlenül fog választani a lejátszási listából kikapcsolásig
* Előző lejátszása: az listából elindítja az előző zenét
* Indítás, megállítás: megállítja vagy elindítja a zenét az ikonnak megfelelően
* Következő lejátszása: az listából elindítja az következő zenét
* Újra lejátszás: újra lejátssza az jelenlegi zenét. Egyszer megnyomva egyszer, kétszer lenyomva végtelenszer
* Letöltés: letölti a jelenleg lejátszott zenét
* Időzítő: aktiválása után 30 perc múlva kiléptet az oldalról



3.10 ábra – Lejátszó

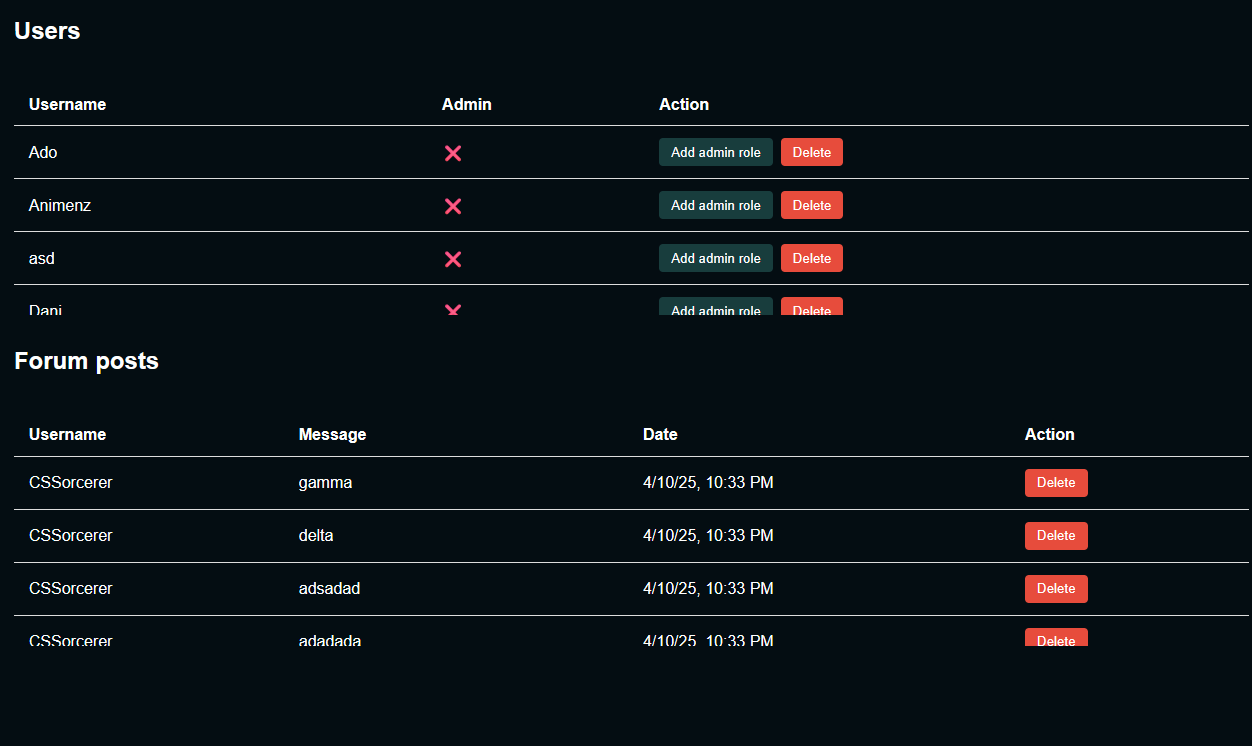
A „Chat” felület egy ismerős megnyitása után lesz látható. Legalul mindig a legújabb üzenet látható, bal oldalt az ismerősé míg jobb oldalt a saját üzenetek. Sima üzeneteken kívül még lehet zenéket is küldeni és az elküldött zenéket le is lehet játszani. A chat része egy input mező az üzenetek írására, illetve egy a zene keresésére. Az üzenetet a küldés gombbal lehet elküldeni. (3.11 ábra)



3.11 ábra – Chat

3.1.2. Az adminisztrációs felület

Az admin jogú felhasználok számára elérhető oldalak dizájnja nem tér el nagyban. A felületen az admin adminisztrációs jogot adhat más felhasználónak, illetve azokat törölheti is. Ez egy listaként jelenik meg a felület tetején. Amennyiben törölni szeretnénk egy felhasználót, azt egy felugró ablakkal jóvá kell hagyni. Az oldal alján még található egy fórumbejegyzés törlése rész is, ahol szintén egy listából tudja az admin törölni a bejegyzést. (3.12 ábra)



3.12 ábra – Admin

3.2 Az adatbázis megtervezése

Az alkalmazás működéséhez szükséges különféle adatokat és beállításokat tartósan kell tárolni. Erre a célra Firebase adatbázist és IndexedDB-t használunk, amelyek lehetővé teszik az adatok strukturált, jól karbantartható és átlátható kezelését. A Firebase valós idejű adatbázisa ideális a felhasználói fiókokhoz, lejátszási listákhoz, feltöltött zenékhez és közösségi funkciókhoz, míg az IndexedDB a kliensoldali tárolást (pl. letöltött zenék offline elérését) biztosítja. Az adatok lekérdezése és módosítása az Angular szolgáltatásokon keresztül történik, amely biztosítja az adatok szinkronizálását és kezelését.

Külön táblákban kell tárolni:

* A regisztrált felhasználó adatait
* A zenék adatait
* A podcast adatait
* A lejátszási listákat
* A lejátszási listák követését
* Az ismerősök kapcsolatát
* A követéseket
* A chat üzeneteket
* A fórum bejegyzéseket
* A címkéket
* Az eddig lejátszott tartalmakat
* A letöltött tartalmakat

3.2.1. Adatbázisséma és diagramok

A ZeeSharing adatbázissémája:

**Users** (email, username, type, picture)

**Musics** (name, performer, img, audio, tags, uploadDate)

**Podcasts** (name, performer, img, audio, tags, uploadDate)

**PlayLists** (title, user, songs, type, createdAt)

**FollowedLists** (Lista, User1, User2)

**Friends** (User1, User2, type)

**Follows** (user, performer, followedAt)

**Messages** (sender, receiver, message, type, time, seen)

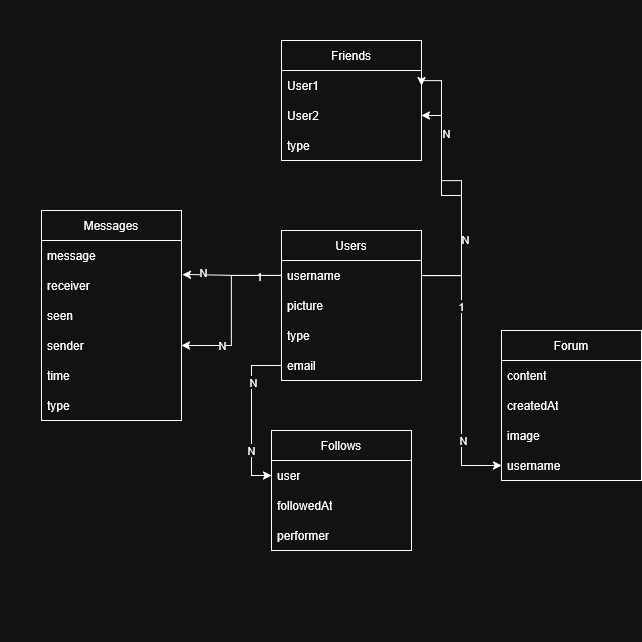
**Forum** (username, content, image, createdAt)

**Tags** (tag)

**PlayHistory** (name, performer, img, playedAt, tags, user)

**Downloaded** (key, name, performer, audio, img, imgURL)

A táblák tartalma bővebben 2 részre bontva:

* Közösségi és kommunikációs táblák (3.13 ábra)

3.13 ábra – Adatbázis séma diagram 1

- **Users**: a felhasználók adatait tartalmazza. Azok közül az e-mail cím és a felhasználónév egyedi, mivel ezekkel tudjuk azonosítani a felhasználót. Illetve eltárol egy kép elérési linket is, ami regisztrációnál egy alapértelmezett képre mutat, de ez megváltoztatható. Illetve még tárolva van a típusa, hogy a felhasználó előadó, podcaster vagy átlag felhasználó-e. A jelszó azért nincs ebben a táblában tárolva, mivel azt a Firebase Authentication funkciójába van mentve titkosítva.

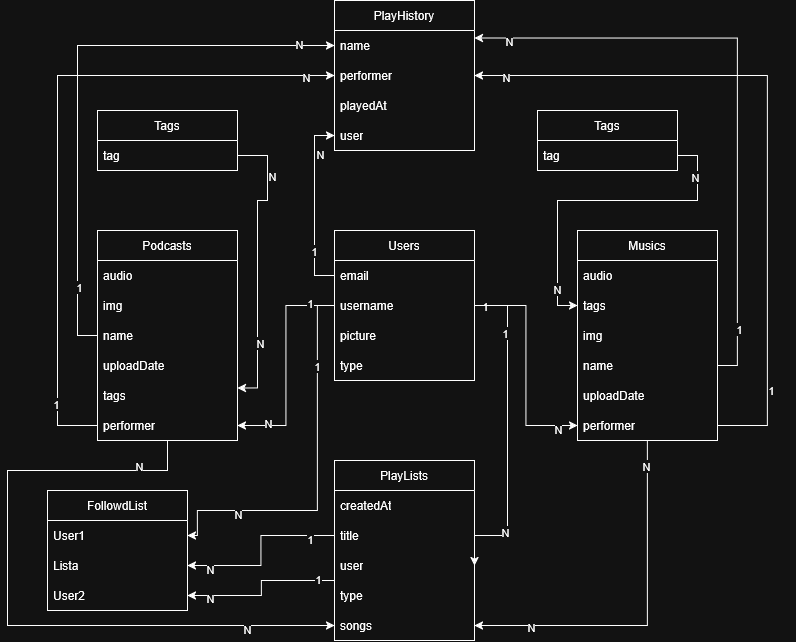
- **Messages**: a felhasználók közötti privát beszélgetéseket tárolja. Tartalmazza a küldő és címzett fél nevét, ami miatt könnyebb a megjelenítés menete, illetve au üzenet típusát, mivel az lehet sima szöveg, illetve zene hivatkozás is (msg – sima üzenet, music – zene hivatkozás). A sorrendbe rendezésben segít, hogy az elküldési idő is le van mentve. Az oldal az is figyelembe veszi, hogy az üzenetet látta-e már a címzett (true – látta, false – nem látta), ezért az is tárolva lett. Itt az obszcén kifejezéseket is tároljuk, illetve a megjelenítés során nem cenzúrázzuk azokat.

- **Friends**: ez a tábla tartalmazza az ismerősi kapcsolatokat. Tartalmazza a két felhasználó nevét, illetve mivel 2 típusa van egy kapcsolatnak (ismerős, jelölés), így azt is külön lementjük. Azt nem tároljuk, hogy egy kérés ellett-e utasítva, hanem csak simán töröljük a ismerős igénylést.

- **Forum**: a fórum bejegyzéseket tárolja. Menti a bejegyzés létrehozóját, a feltöltés dátumát, magát a bejegyzés szövegét, illetve egy képnek a linkjét. Ha nincs kép feltöltve a bejegyzéshez, akkor „NoImage” érétket vesz fel az a mező. Obszcén kifejezések is eltárolódnak az adatbázisban, de azokat megjelenítéskor cenzúrázzuk.

- **Follows**: a követési kapcsolatokat raktározza. Menti, hogy melyik felhasználó melyik előadót vagy podcastert követi, és hogy mikortól követi.

- **FollowedLists**: tárolja, hogy egy felhasználó mely más felhasználók által létrehozott listákat követi. A listákat a lista nevével és a lista létrehozójával lehet azonosítani, mivel egy felhasználó nem hozhat létre ugyanolyan nevű listákat.

* Médiával kapcsolatos táblák (3.14 ábra)

3.14 ábra - Adatbázis séma diagram 1

- **Tags**: ez a tábla tartalmazza azokat a címkéket, amikkel a zenéket, illetve podcasteket megjelölheti a feltöltő.

- **PlayLists**: a felhasználó által létrehozott lejátszási listákat tárolja. Menti a lista nevét, a létrehozó nevét, illetve a létrehozás dátumát. Mivel a lista lehet privát, publikus, vagy csak az ismerősök által látható (private, public, friend), így azt típusként is eltároljuk. A listába mentett zenék egy tömbben vannak tárolva.

- **PlayHistroy**: az összes eddigi lejátszást tárolja. Menti, hogy ki hallgatta a fájl, illetve, hogy mikor lett lejátszva. A zene/podcast azonosítása érdekében még eltárolva a feltöltő nevét, illetve a fájl címét is.

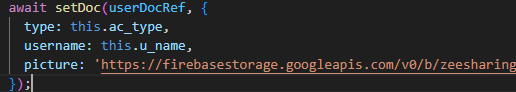
-**Musics/Podcasts**: ezekben a táblákban tároljuk a feltöltött tartalmat adatait. Két elérési út is elvan mentve, amik a hangfájlokra és a képekre hivatkoznak. Egy tömbben vannak tárolva a feltöltéskor megadott címkék, ami mellé maga a dátum is mentve van.

4. Megvalósítás

Az adatbázis séma kialakítását követően a fejlesztést lépésről lépésre, logikus sorrendben végeztem. Kiemelt figyelmet fordítottam arra, hogy a megírt kód moduláris és újrahasznosítható legyen, ezáltal elősegítve a könnyebb karbantarthatóságot és a jövőbeni bővíthetőséget. A komponensek és szolgáltatások kialakításánál törekedtem a jól elkülönülő felelősségi körökre (Separation of Concerns), hogy a rendszer áttekinthető és skálázható maradjon. A fejlesztés során Angular keretrendszert használtam a frontend logika és komponensstruktúra kialakításához, Firebase-t [1] az autentikációhoz és felhőalapú adatkezeléshez, valamint IndexedDB-t [4] az offline elérhetőség és a lokális tárolás biztosítására. A refaktorálási lehetőségeket már a tervezéskor figyelembe vettem, hogy a későbbi módosítások a lehető legkisebb erőfeszítéssel végrehajthatók legyenek.

4.1. Regisztráció

A fejlesztés első lépése a felhasználói regisztráció megvalósítása volt, mivel ez alapfeltétele a későbbi, személyre szabott funkciók használatának. A regisztrációs űrlap mezői közé bekerült a felhasználónév, e-mail cím, jelszó, jelszó megerősítés, valamint egy legördülő lista a fióktípus (előadó, podcast, alapértelmezett felhasználó) kiválasztására. Az űrlap elküldése az *(ngSubmit)="onRegister()"* eseményen keresztül történik, amely a *RegisterComponent* komponens *onRegister()* metódusát hívja meg. A regisztráció során elsőként ellenőrzésre kerül, hogy minden kötelező mező ki van-e töltve, valamint, hogy a két jelszó megegyezik-e. Amennyiben ezek a feltételek teljesülnek, a *createUserWithEmailAndPassword()* Firebase Auth metódus segítségével létrejön az új felhasználói fiók. Sikeres regisztráció után az alkalmazás automatikusan létrehoz egy Firestore dokumentumot a *Users* gyűjteményben, ahol eltároljuk a felhasználónév, fióktípus és egy alapértelmezett profilkép értékét is. Végül automatikusan bejelentkezik a frissen regisztrált felhasználó, és átirányításra kerül a bejelentkezett felületre. A regisztrációhoz kapcsolódó hibaüzenetek (*registerError*) szintén a komponens állapotában kerülnek tárolásra és jelennek meg a sablonban, így felhasználóbarát visszajelzést adnak. Ez a részlet mutatja, hogyan történik meg a felhasználó adatainak mentése a Firestore adatbázisba. (4.1 ábra)



4.1 ábra – Regisztráció feltöltése

4.2. Bejelentkezés

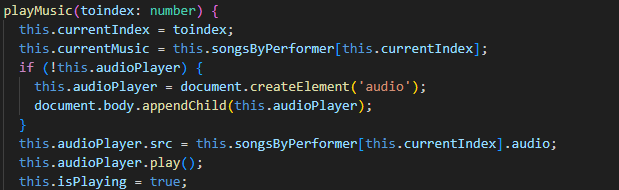
A bejelentkezési funkció Firebase Authentication segítségével történik. A felhasználó megadja email-címét és jelszavát, amelyet a *signInWithEmailAndPassword()* metódus ellenőriz. Sikeres bejelentkezés után a rendszer lekéri a felhasználóhoz tartozó adatokat a Firestore „*Users*” gyűjteményéből (felhasználónév, profilkép, fióktípus), majd a *UserService* szolgáltatáson keresztül elmenti ezeket a sessionhöz. Hibás adatok esetén figyelmeztető üzenet jelenik meg, illetve lehetőség van jelszó-visszaállítás kérésére a *sendPasswordResetEmail()* metódussal. A komponens figyeli az internetkapcsolat állapotát, és offline módban egy külön gombbal az „offline főoldalra” navigál.

4.3. Kinézet

A CSS-t minden felület kialakításakor annak megfelelően, lépésenként írtam meg – először kizárólag az asztali nézetre fókuszálva. Miután az egész projekt funkcionálisan elkészült, utólag készítettem el a teljes reszponzivitást, hogy a felület mobilon és tableten is megfelelően jelenjen meg. [2, 6, 9, 10]

4.4. Zene lejátszása

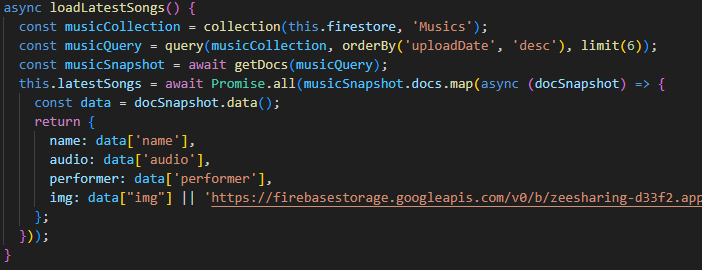
A projekt következő lépése a zenelejátszó rendszer megvalósítása volt. Kezdetben manuálisan töltöttem fel az adatokat az adatbázisba (Firebase Firestore), azaz egyes zenék címét, előadóját, borítóképét, címkéit és az audio fájl URL-jét. Ezek után a fő célom egy működő, alapvető zenelejátszó komponens létrehozása volt. A lejátszó a *playMusic()* függvényen keresztül indítja el a zenét, amely egy audio HTML-elemet hoz létre (ha még nem létezik), és beállítja a kiválasztott szám forrását. (4.2 ábra)

4.2 ábra – Zene lejátszása

A lejátszó vezérlői (lejátszás/szünet, következő, előző, ismétlés, véletlenszerű lejátszás, hangerő, időcsúszka stb.) az Angular komponens HTML sablonjában kaptak helyet. A zene lejátszása mellett naplóztam is a hallgatási előzményeket Firestore-ban, hogy később statisztikát lehessen készíteni a lejátszott zenékről.

4.5. Főoldal

A főoldalon több hasonló felépítésű tartalmi blokkot valósítottam meg, amelyek mind azonos komponensstruktúrát és adatbetöltési mintát követnek. Például a legújabb zenék komponensében (*LatestMusicComponent*) a Firestore adatbázisból lekérdeztem (4.3 ábra) a legfrissebb feltöltéseket, rendezetten az *uploadDate* alapján, limitálva a lekérdezést a legutóbbi 6 elemre. Az adatok aszinkron betöltését az *ngOnInit* életciklusban kezeltem.



4.3 ábra – Legújabb zenék betöltése

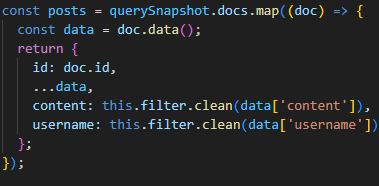
Az ajánlások elkészítéséhez először összegyűjtöttem a felhasználó eddigi hallgatási adatait, amelyeket egy külön Firestore gyűjteményben tárolok. Ezekből lekérdeztem az elmúlt hét nap legtöbbet hallgatott zenéit az adott felhasználóra szűrve, majd azok címkéje alapján alakítottam ki a személyre szabott ajánlásokat. Illetve lekértem az elmúlt 7 nap legtöbbet hallgatott zenéit is, és a legújabb zenékhez hasonlóan megjelenítettem. Ezek megjelenítettem a lejátszott listának az elemeit, hogy a felhasználó szabadon tudjon váltani az elemek között.

4.6. Lejátszi listák

A lejátszási lista funkciók fejlesztése során egy teljeskörű rendszert építettem ki, amely lehetővé teszi a felhasználók számára saját listák létrehozását, azok nyilvánossági szintjének szabályozását (publikus, barát, privát), zenék hozzáadását és eltávolítását, valamint más felhasználók listáinak követését is. A listákhoz tartozó adatokat Firestore-ban tárolom és frissítem. A saját listák betöltését és kezelését a *PlayListsComponent* komponens végzi, a háttérlogikát pedig egy különálló *PlaylistService* szolgáltatásba szerveztem. A lejátszási lista létrehozásánál figyeltem, hogy a felhasználónak van-e már olyan nevű listája, és ha van, akkor arról értesítést kap. Mikor egy felhasználó a *toggleSongInPlaylist()* metódus segítségével új zenét akar a listához, akkor megvizsgálom, hogy az a zene szerepel-e már a listában és, ha igen, akkor eltávolítom azt.

4.7. Fórum

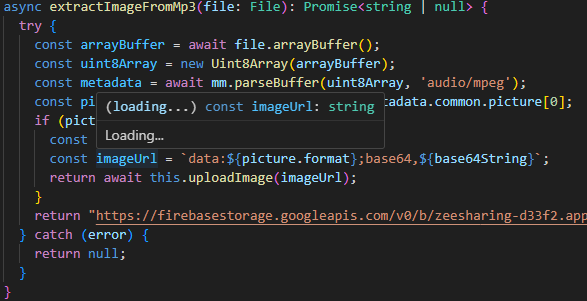
A ZeeSharing Forum modul lehetővé teszi a felhasználók számára, hogy szöveges és képes bejegyzéseket osszanak meg egymással, valamint megtekintsék a korábbi hozzászólásokat. A bejegyzések Firebase Firestore-ban tárolódnak, képek esetén a feltöltés Firebase Storage-on keresztül történik. A komponens induláskor lekéri a legújabb 20 bejegyzést, amelyek folyamatosan frissülnek a *ForumService* szolgáltatás *posts$ observable*-ének segítségével. A felhasználó lefelé görgetve automatikusan betöltheti a régebbi posztokat is. Üzenetküldés csak nem alapértelmezett (*def\_user*) típusú felhasználók számára engedélyezett, és lehetőség van kép csatolására is. A bejegyzések szöveges tartalma és felhasználóneve szűrésen megy keresztül egy bővített szótárat használó sértőszó-szűrő segítségével. (4.4 ábra) Adminisztrátori jogosultsággal a posztok törölhetők, a hozzájuk tartozó képek pedig automatikusan eltávolításra kerülnek a tárhelyről is. A modul két fő szolgáltatásra épül: a *UserService* biztosítja a felhasználói adatokat, a *ForumService* kezeli a posztokkal kapcsolatos műveleteket. A komponens egyszerű, jól áttekinthető felületet biztosít, ikonokkal ellátott gombokkal, dinamikus tartalombetöltéssel és képfeltöltési lehetőséggel.

4.4 ábra – Üzenet szűrése

4.8. Feltöltés

A feltöltés funkció megvalósításánál arra törekedtem, hogy a felhasználók egyszerűen és kényelmesen tudjanak MP3 fájlokat feltölteni az alkalmazásba, opcionálisan borítóképet is csatolva, illetve címkéket hozzárendelve a zenéhez. Az interfész két különálló fájlbemenetet tartalmaz, egyet az MP3 fájl kiválasztására, egyet pedig az opcionális kép feltöltésére. A kiválasztott fájlok nevét egy egyedi stílusú felirat jeleníti meg, amely helyettesíti a hagyományos fájlválasztó mezőt. A zene címét egy szöveges mezőben kell megadni, amely kötelező, így a feltöltés csak akkor engedélyezett, ha van érvényes MP3 fájl és megadott cím. Az elérhető címkéket a komponens indításakor töltöm be a Firestore „*Tags*” gyűjteményéből, és ezeket gombokként jelenítem meg a felhasználó számára, akik ezek közül kiválaszthatják a zenéhez illő címkéket. A címkék kiválasztása vagy törlése egyszerű kattintással történik.

A feltöltési folyamat két fő részre oszlik. Először, ha a felhasználó megadott borítóképet, azt külön töltöm fel a Firebase Storage megfelelő helyére, miközben valós időben követem a feltöltés állapotát, hogy a felhasználó láthassa a haladást. Ha nincs kép kiválasztva, megpróbálom az MP3 fájl metaadataiból kinyerni a beágyazott borítóképet a music-metadata könyvtár segítségével. (4.5 ábra) A folyamat úgy zajlik, hogy a kiválasztott MP3 fájlt először ArrayBuffer formátumba olvasom be, majd ebből Uint8Array-t készítek, amit átadok a music-metadata-nak a *parseBuffer* metóduson keresztül. Ez elemzi a fájl belső adatait, és a *common.picture* mezőben visszaadja a borítóképet, amennyiben az megtalálható. A kép valójában egy bináris adat, amit base64 formátumba kódolok az encode függvény segítségével, majd az így kapott base64 stringet beágyazom egy adat-URI formátumba (data:<formátum>;base64,<kódolt\_adat>), amit már a böngésző is közvetlenül tud kezelni.

4.5 ábra – Kép kinyerése a hangfájlból

Siker esetén ezt is feltöltöm, ellenkező esetben egy alapértelmezett képet használok. Ezután következik az MP3 fájl feltöltése, mely szintén valós időben jelzi a haladást. A feltöltések befejezése után az audiofájl és a kép URL-jét, a megadott címet, a kiválasztott címkéket, a feltöltés időpontját és a feltöltő felhasználó nevét mentem a Firestore megfelelő gyűjteményébe. Ez a gyűjtemény a felhasználó típusától függően „*Musics*” vagy „*Podcasts*” lehet. A dokumentumazonosító a fájl neve, így egyértelműen azonosítható minden feltöltés. [1, 3]

4.9. Keresés

A *SearchComponent* egy Angular komponens, amely egy komplex keresőfunkciót valósít meg a zenék, podcastok, előadók, felhasználók és címkék között. A felhasználó szöveges bevitelen keresztül indít keresést, melyet a komponens a *searchText* változóban köt le, és az *onSearchChange()* metódus indít el. A keresési kérések feldolgozását a komponens egy RxJS Subject-tel, debounceTime-mal és distinctUntilChanged operátorokkal kezeli, hogy minimalizálja a felesleges lekérdezéseket (4.6 ábra), így csak akkor hajt végre keresést, ha a felhasználó legalább 2 karaktert beírt, és nem gépel túl gyorsan.

4.6 ábra – Kereső lekérdezések

A tényleges keresés a Firestore adatbázison belül több gyűjteményre vonatkozik: *Musics, Podcasts, Users és Tags*. A komponens a Firestore *query*, *where* és *getDocs* metódusaival végzi a szűrést, ahol az adott beviteli szöveg alapján prefix keresést alkalmaz, azaz a keresett szöveghez illeszkedő dokumentumokat keresi, például zenék vagy előadók nevét. A keresési eredmények egy strukturált objektumba (*searchResults*) kerülnek, amely kategorizálva tartalmazza az eredményeket: zenék, podcastok, előadók, felhasználók és címkék. A keresési eredmények megjelenítése az HTML-ben szekciókra bontva történik, ahol minden kategóriához külön blokk van definiálva, és az eredménylista elemei egyenként kattinthatók, hogy részletes nézetet nyissanak meg. A részletes nézet megjelenítésekor a komponens eltávolítja a korábbi keresési eredményeket, és a kiválasztott elem alapján (zene, előadó vagy felhasználó) megjeleníti az adott entitás adatait és kapcsolódó tartalmait. Például előadó esetén lekéri az összes hozzá tartozó zenét vagy podcastot, a címkék gyűjteményét, és az adott felhasználó követési állapotát is.

Az előadók részletes nézetében megjelennek az előadó profiladatai (felhasználónév, típus, kép), a címkék halmaza, és az összes zene, amelyhez az előadó tartozik. Innen elérhető a teljes zenei lista lejátszása az *onSongSelected()* metódussal, vagy egy adott dal kiválasztása az *onOneClicked()* eseménykibocsátón keresztül. A komponens kezeli a követés funkciót is, amely Firestore-ban tárolt „*Follows*” gyűjtemény alapján mutatja, hogy az aktuális felhasználó követi-e az adott előadót, és ennek megfelelően kínálja fel a „Follow” vagy „Unfollow” gombokat.

Felhasználó esetén a részletes nézet megjeleníti a profilképet, felhasználónevet és egy státuszt, amely a barátsági kapcsolatot mutatja: nincs kapcsolat, kérés elküldve vagy barát. Ezen státusz alapján megváltoznak a gombok is, például barátkérés küldése, törlése vagy barát törlése. Ha barátként van regisztrálva a kapcsolat, a komponens lekéri és megjeleníti az adott felhasználó lejátszási listáit, beleértve a nyilvános és baráti típusokat is, valamint a listák zenéit. A zenék kattintásra lejátszhatók, és a listákban a zenék megjelenítésének megnyitása vagy elrejtése is kezelhető.

A komponens a felhasználói állapotokat és adatokat (pl. követési állapot, baráti státusz, lejátszási listák) a Firestore lekérdezések mellett a *UserService*-ből szerzi meg, amely az aktuális bejelentkezett felhasználó adatait biztosítja. Minden lekérdezés és állapotváltoztatás aszinkron módon történik, hogy a felhasználói élmény zökkenőmentes legyen, és az adatfrissítés valós időben történjen.

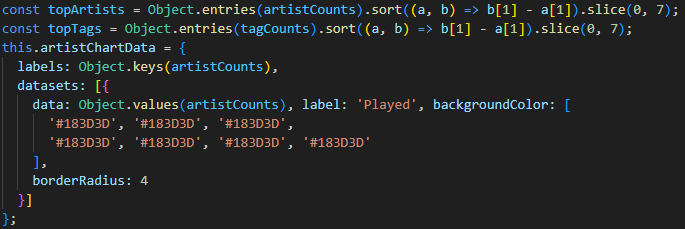
Az eseménykezelés fontos része a komponensnek, amely az Angular EventEmitter-en keresztül továbbítja a kiválasztott zeneszám vagy zenei lista eseményeket a szülő komponensek felé, így például a lejátszó komponens is reagálhat a keresőből érkező eseményekre. A komponens egyaránt használ Angular formok és kétirányú adatkövetést (ngModel), valamint dinamikus DOM elemeket az \*ngIf és \*ngFor direktívákkal a keresési eredmények és részletek megjelenítéséhez.

4.10. Felhasználó

A *UserComponent* az Angular alapú alkalmazás felhasználói profil kezelő komponense, amely a felhasználó adatainak megjelenítését, jelszóváltoztatást, profilkép cserét, barátok és követett előadók kezelését, valamint lejátszási listák kezelését végzi. Az összes adat a Firestore adatbázisból töltődik be és kerül frissítésre, a Firebase Storage-ban tárolódnak a profilképek.

Az *ngOnInit* metódus betölti a felhasználó adatait a *userService*-ből, majd lekéri az elmúlt 7 nap hallgatási adatokat a *PlayHistory* gyűjteményből, ahol az adott felhasználó email címére szűr. Ezek alapján statisztikákat generál különböző grafikonokra, melyeket a ng2-charts csomag Chart.js alapú komponenseivel jelenít meg. A hallgatási adatokból az előadók és címkék gyakorisága számítódik ki, majd az első 7 leggyakoribb elemet jeleníti meg külön diagramokon. (4.7 ábra) A barátok, követett előadók és lejátszási listák szintén a Firestore megfelelő gyűjteményeiből töltődnek be, a lekérdezések az aktuális felhasználó nevére vagy email címére szűrnek.

A jelszóváltoztatás az Firebase Authentication reset jelszó email küldő funkcióját használja, amely aszinkron módon elküldi a visszaállító linket az aktuális email címre, erről egy alert értesíti a felhasználót. A profilkép cseréje egy input mezővel történik, ahol a kiválasztott képfájlt a Firebase Storage-ba töltjük fel az adott felhasználónévhez tartozó útvonalra. A feltöltés előtt töröljük az előző képet, majd a letöltési URL-t elmentjük a *Users* gyűjtemény megfelelő dokumentumában és a *userService*-ben is frissítjük a lokális adatot, így az UI azonnal frissül.



4.7 ábra – Grafikon létrehozása

Barát törléskor a *Friends* gyűjteményben keresünk olyan dokumentumot, ahol a két felhasználó szerepel egymás mellett, és ha megtaláljuk, töröljük az adott rekordot, majd újratöltjük a barátokat. Hasonlóan, a követett előadók törlése a *Follows* gyűjteményből töröl egy bejegyzést a jelenlegi felhasználó és az előadó alapján. Lejátszási lista törlés is a *PlayLists* gyűjteményben szűri a felhasználóhoz tartozó listákat és törli a kiválasztottat.

4.11. Ismerősök/követések

A *FriendsComponent* az Angular alkalmazás egyik közösségi központú része, amely a felhasználók baráti és követési kapcsolatait jeleníti meg és kezeli. A komponens célja, hogy a felhasználók könnyedén elérhessék barátaikat, figyelemmel kísérjék a követett előadók tevékenységeit, valamint gyorsan reagálhassanak a beérkező barátkérelmekre.

A komponens betölti azokat a felhasználókat, akikkel az aktuális felhasználó kölcsönös kapcsolatban van – vagyis barátként szerepelnek a Firestore *Friends* kollekciójában. A kapcsolatok irányától függetlenül (azaz akár a felhasználó kezdeményezte, akár ő kapta a kérelmet), minden olyan kapcsolat megjelenik, amely „friend” típusúra lett állítva. A lekérdezett barátokról további adatokat is megjelenít – például profilképet –, amit a *Users* kollekcióból kér le. Ezek alapján épül fel a vizuális barátlista. (4.8 ábra)

4.8 ábra – Barátok betöltése

A komponens megjeleníti, hogy melyik baráttól érkezett új, még olvasatlan üzenet. Ezt úgy valósítja meg, hogy a *ChatService*-en keresztül lekéri az üzenetek olvasottsági állapotát, és az adott felhasználónévhez tartozó státusz alapján jelenít meg egy kis zöld pöttyöt. Ez lehetővé teszi, hogy a felhasználó könnyedén felismerje, kitől érkezett új üzenet anélkül, hogy minden beszélgetést külön-külön meg kellene nyitnia.

A barátlistától külön jelenik meg a követett előadók listája, akiket a felhasználó követési szándékkal jelölt meg. Ezeket a kapcsolatok a *Follows* kollekció tartalmazza. A komponens itt is lekérdezi a releváns felhasználói adatokat, és egy külön listában jeleníti meg őket. Ha a felhasználó egy követett előadóra kattint, a komponens automatikusan lekéri az adott előadóhoz tartozó összes zenét a *Musics* kollekcióból, és azokat lejátszásra előkészíti. Így közvetlen kapcsolat van a társas interakció és a zenei tartalomfogyasztás között.

A komponens aktív szerepet vállal a közösségi kapcsolatok kialakításában is. Betölti azokat a felhasználóktól érkező kapcsolatfelvételi kéréseket, amelyek még függőben vannak. A kérelmek vizuálisan elkülönítve, egy másik listában jelennek meg, ahol a felhasználó egy gombnyomással elfogadhatja vagy elutasíthatja azokat. Elfogadás esetén a kérelmek típusát "friend"-re állítja, míg elutasítás esetén törli őket az adatbázisból. Minden ilyen művelet után a komponens újratölti a kérelmek listáját, hogy a felület mindig naprakész maradjon.

A komponens több eseményt is képes kibocsátani, amikor egy barátra vagy követett előadóra kattint a felhasználó. Ez lehetővé teszi a szülő komponensek számára, hogy a felhasználó interakcióira reagáljanak – például betöltsék a kiválasztott személy profilját, elindítsanak egy beszélgetést, vagy lejátszanak egy zenét. Emellett a FriendService is használatban van, amely lehetővé teszi a felhasználónév állapotának megosztását más komponensekkel, például egy chat vagy profilkomponenssel.

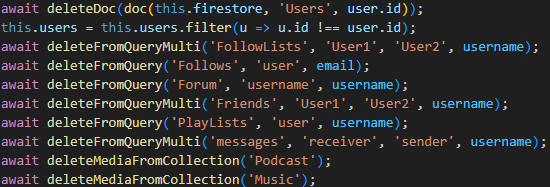
4.12. Chat

A felhasználók privát üzenetküldés keretében cseveghetnek egymással, ahol a *ChatComponent* gondoskodik a felület megjelenítéséről, a küldésről, fogadásról és zeneajánlásról is. A címzett felhasználó neve felül jelenik meg, az üzenetek időrendben listázódnak. A küldő és fogadó képe megjelenik az üzenet mellett. A szöveges üzenetek mellett zeneszámokat vagy podcasteket is lehet küldeni, amelyek miniatűr borítóképpel, címmel és előadóval jelennek meg, lejátszhatóak a felületen.

A zenék keresése debounceTime-al történik, és két Firestore gyűjteményben (*Musics*, *Podcasts*) keres a *performSearch* metódus. Az ajánlott találatok cím szerint szűrhetők, és egy kattintással küldhetők. A *ChatService* gondoskodik az üzenetek küldéséről, mentéséről, olvasatlanság követéséről, valamint a valós idejű fogadásról. A *listenForMessages* metódus kétirányú lekérdezéssel figyeli a küldött és kapott üzeneteket, és összeolvasztva időrendbe állítja őket. A chat automatikusan legörget az új üzenetekhez, és külön figyeli, ha másik felhasználó kerül kiválasztásra a barátlistából. A zeneajánló rész rejtve van, és külön gombbal előhívható, így nem zavarja a normál beszélgetést. A chat támogatja a Zene típusú objektumokat, melyek később más komponens felé is továbbküldhetők (@Output()).

4.13. Admin

Az *AdminComponent* egy belső adminisztrációs felületet biztosít a felhasználók és fórumposztok kezelésére. Betöltéskor automatikusan lekéri a *Users* és *Forum* Firestore gyűjtemények adatait, és táblázatos formában megjeleníti azokat. A felületen látható, hogy ki rendelkezik admin jogosultsággal, és lehetőség van az admin szerepkör ki-be kapcsolására (*toggleAdmin*).

A felhasználók törlése során több kapcsolódó adatszerkezetből is megtörténik a takarítás (4.9 ábra). A *deleteUser* metódus megerősítés után eltávolítja a felhasználót a *Users* gyűjteményből, és az alábbi gyűjteményekből is törli az érintett bejegyzéseket: *FollowLists*, *Follows, Forum, Friends, PlayLists, messages*. Ezen kívül a *Podcast* és *Music* gyűjteményekből is törlésre kerülnek a felhasználó által feltöltött médiaelemek, valamint a kapcsolódó Firebase Storage fájlok, ha nem az alapértelmezett kép URL-jét használják. A médiafájlok útvonala a fájl URL-jéből kerül visszafejtésre, és külön-külön kerülnek törlésre (*deleteObject*).

4.9 ábra – Adatok törlése

A fórum bejegyzések szintén törölhetők a felületen. A *deleteForumPost* metódus a Forum gyűjteményből azon bejegyzéseket keresi meg, melyek *username* és *createdAt* mezői megegyeznek a kijelölt poszttal, és törli azokat. Az eltávolítás után az UI listából is eltűnik az adott bejegyzés.

4.14. Offline főoldal

Az offline zenehallgatás funkció egy dedikált *OfflineMainComponent* komponensben valósul meg, amely az IndexedDB segítségével biztosítja a letöltött zenék böngészését és kezelését internetkapcsolat nélkül is. A letöltött dalokat és borítóképeiket bináris formában tároljuk, majd betöltéskor Blob objektumokká alakítjuk és Object URL-ként jelenítjük meg. A felület listázza a letöltött fájlokat, lehetőséget biztosít azok törlésére. A megoldás Progressive Web App formában készült, hogy a felhasználók natív alkalmazásként is telepíthessék. A PWA-t azért választottam, mert lehetővé teszi az offline működést, a lokális adattárolást, valamint a telepíthetőséget. Ez különösen fontos volt a zenehallgatás offline elérhetőségének biztosításához. A háttérben működő IndexedDB tökéletesen együttműködik a PWA modellel, lehetővé téve a teljes zenei élmény elérhetőségét internet nélkül is. [5]

5. Tesztelés

A tesztelés célja a hibák felderítése és megelőzése, valamint a rendszer megbízhatóságának növelése. A vizsgálat a teljes rendszerre kiterjedt, beleértve az adatbázist is, és a specifikációban megfogalmazott követelmények teljesülését ellenőriztem.

5.1. A tesztelés menete

A munkám során figyelembe vettem a „tesztelési aranyszabályokat”. Minden elkészült funkciót tesztadatok segítségével ellenőriztem, és ha hibát találtam, javítottam azt, így a későbbi tesztelés során ezek a problémák már nem jelentkeztek, ezzel is megkönnyítve a végső ellenőrzést. A rendszert alapos, szisztematikus teszteléssel vizsgáltam, ami időigényes volt, hiszen minden tesztesetet át kellett néznem. Különös figyelmet fordítottam a szélsőséges helyzetekre is, vagyis extrém eseteket is kipróbáltam. A funkciókat a specifikációban szereplő elvárások alapján teszteltem, mind helyes, mind hibás bemenetekkel, utóbbiakkal elsősorban a hibák felderítése volt a cél.

5.2. Tesztesetek

1. Teszteset

* *Teszt tárgya*: Regisztráció (felhasználónév, jelszó és e-mail cím) és elfelejtett jelszó tesztelése
* *Leírás*: Megfelelő információval megtöltjük az űrlapot, majd teszteljük, hogy felkerül-e az adatbázisba
* *Feladat*:   
  1. LÉPÉS: Mezők kitöltése szintaktikailag helyes  
  *Tesztadatok*: Felhasználónév: ASD, Email: asd@gmail.com Jelszó/Jelszó újra: Pass1234, Típus: előadó  
  *Elvárt eredmény*: A felhasználó létrejön, adatai eltárolódnak  
  *Kapott eredmény*: Sikeres a regisztráció. – Sikeres lépés  
  2. LÉPÉS: Mezők kitöltése ugyanazzal az adatokkal  
  *Elvárt eredmény*: A felhasználó nem jön létre létező e-mail miatt  
  *Kapott eredmény*: Sikertelen a regisztráció. – Sikeres lépés

2. Teszteset

* *Teszt tárgya*: Belépés tesztelése
* *Leírás*: Hibás adatokkal próbálunk bejelentkezni
* *Feladat*:   
  1. LÉPÉS: Mezők kitöltése hibás adatokkal  
  *Tesztadatok*: Felhasználónév: ASD1, Email: asd@gmail.com Jelszó/Jelszó újra: Pass1234, Típus: előadó  
  *Elvárt eredmény*: A felhasználó nem tud belépni  
  *Kapott eredmény*: Sikertelen belépés. – Sikeres lépés  
  2. LÉPÉS: Mezők kitöltése jó adatokkal  
  *Elvárt eredmény*: Sikeres belépés, továbblépés a főoldalra  
  *Kapott eredmény*: Sikeres belépés Nem lép tovább a főoldalra. – Sikertelen lépés  
  *Megoldás*: továbblépési funkció meghívása

3. Teszteset

* *Teszt tárgya*: Lejátszás tesztelése
* *Leírás*: A lejátszó összes funkcióját tesztelni kell, hogy a későbbiekben ne legyen ezzel probléma
* *Feladat*:   
  1. LÉPÉS: Lista betöltése és elindítása, majd a gombok tesztelése sorba  
  *Tesztadatok*: Ado előadó lejátszási listája  
  *Elvárt eredmény*: Összes funkció működése  
  *Kapott eredmény*: Az előre és visszalépésnél megakad a lépés a listák elején és végén. – Sikertelen lépés  
  *Megoldás*: lista elejéről/végéről az ugrás kezelése  
  2. LÉPÉS: Gombok tesztelése sorba újra  
  *Elvárt eredmény*: Összes funkció működése  
  *Kapott eredmény* Összes funkció működése. – Sikeres lépés

4. Teszteset

* *Teszt tárgya*: Zene feltöltése
* *Leírás*: Teszteljük a hangfájlok feltöltését, hogy minden adat felkerül-e az adatbázisba, tárgelyre
* *Feladat*:   
  1. LÉPÉS: Fálj kiválasztása (most saját képpel)  
  *Tesztadatok*: Ado előadó egy zenéje képpel  
  *Elvárt eredmény*: Adatok felkerülése   
  *Kapott eredmény*: Minden adat sikeresen felkerült. – Sikeres lépés  
  2. LÉPÉS: Fálj kiválasztása (most saját kép nélkül)  
  *Tesztadatok*: Ado előadó egy zenéje kép nélkül  
  *Elvárt eredmény*: Adatok felkerülése  
  *Kapott eredmény*: Kép feltöltés sikertelen – Sikertelen lépés  
  *Megoldás*: a kinyerés lassabban történt, mint a zene feltöltése, és lezárta a funkciót. Addig nem zár le, míg fel nem tölti a 2 fájlt

5. Teszteset

* *Teszt tárgya*: Forum üzeneteinek a betöltése
* *Leírás*: az üzeneteket darabonként töltjük be, hogy gyorsabb legyen az alkalmazása
* *Feladat*:   
  1. LÉPÉS: Fórum betöltése és görgetés   
  *Tesztadatok*: Fórum adatbázis  
  *Elvárt eredmény*: Az elemek tízesével töltenek be   
  *Kapott eredmény*: Üres fórum felület. – Sikertelen lépés  
  *Megoldás*: a hibát egy rossz adatbázislekérés okozta, kijavítása után jól működött

6. Teszteset

* *Teszt tárgya*: Chat üzenet küldése ismerősnek
* *Leírás*: leteszteljük, hogy az üzenetek átmennek-e a felhasználónak
* *Feladat*:   
  1. LÉPÉS: Ismerős kiválasztása és üzenetküldés   
  *Tesztadatok*: ASDASDADS üzenet  
  *Elvárt eredmény*: Az üzenet felkerül az adatbázisba, megjelenik a felületen   
  *Kapott eredmény*: Az üzenet felkerült/megjelent. – Sikeres lépés  
  2. LÉPÉS: Ismerős kiválasztása és zene küldése   
  *Tesztadatok*: Ado előadó egyik zenéje  
  *Elvárt eredmény*: Az üzenet felkerül az adatbázisba, megjelenik a felületen   
  *Kapott eredmény*: Az üzenet felkerült, de nem jelent meg. – Sikertelen lépés  
  *Megoldás*: nem volt vizsgálva, hogy milyen típus az üzenet, így sima üzenetet akart megjeleníteni.

7. Teszteset

* *Teszt tárgya*: User felület tesztelése
* *Leírás*: leteszteljük, hogy az felhasználó tudja-e használnia neki szánt funkciókat
* *Feladat*:   
  1. LÉPÉS: Sorban leteszteljük a funkciókat   
  *Tesztadatok*: új kép, ASD nevű barát, Teszt1 nevű lista, Ado előadó követése  
  *Elvárt eredmény:* Minden a megfelelő módon működik   
  *Kapott eredmény*: Minden funkció lefut. – Sikeres lépés

8. Teszteset

* *Teszt tárgya*: Admin felület tesztelése
* *Leírás*: leteszteljük, hogy az admin tudja-e használnia neki szánt funkciókat
* *Feladat*:   
  1. LÉPÉS: Sorban leteszteljük a funkciókat   
  *Tesztadatok*: ASD nevű felhasználó (törlés), random fórum poszt (törlés)  
  *Elvárt eredmény:* Minden a megfelelő módon működik   
  *Kapott eredmény*: Minden funkció lefut. – Sikeres lépés

9. Teszteset

* *Teszt tárgya*: Letöltött zenék lejátszása offline felületen
* *Leírás*: az offline oldal éri el a letöltött zenéket, ennek a működését feltétlen tesztelni kellett
* *Feladat*:   
  1. LÉPÉS: Internet nélküli elérés, majd zene lejátszása   
  *Tesztadatok*: Ado előadónak a zenéi letöltve  
  *Elvárt eredmény:* A zenéket szabadon lejátszhatjuk   
  *Kapott eredmény*: Nem töltenek be a zenék – Sikertelen lépés  
  2. LÉPÉS: Internet nélküli elérés, majd zene lejátszása a javított verzión  
  *Tesztadatok*: Ado előadónak a zenéi letöltve  
  *Elvárt eredmény:* A zenéket szabadon lejátszhatjuk   
  *Kapott eredmény*: a zenék betöltenek, lejátszhatóak, de a képek nem elérhetőek – Sikertelen lépés  
  *Megoldás*: a képet rossz címke alapján volt elérve, annak átírása megoldotta a problémát

5.3. Összegzés

A tesztelési folyamat eredményesnek bizonyult. Minden funkciót két alkalommal ellenőriztem: először a fejlesztésük befejezése után, majd a teljes rendszer elkészültekor ismét. A tesztelés során több hiba is napvilágra került. Az hibákat a felismerésüket követően azonnal kijavítottam, és a tesztelést ezután zavartalanul folytattam.

Összefoglalás

Szakdolgozatom keretében egy olyan közösségi webalkalmazást készítettem, amely a zene- és podcasthallgatók igényeit hivatott kielégíteni, és a dolgozat leadását követően is használható marad. A tervezés során figyelembe vettem a célcsoport – zenekedvelők és podcast hallgatók – visszajelzéseit, és az implementáció alatt is folyamatosan kikértem véleményüket az egyes funkciókról.

Tapasztalataim szerint akkor tekintenek egy alkalmazást igazán hasznosnak, ha az kimondottan az ő érdeklődési körükre épül, és minden olyan funkciót tartalmaz, ami megkönnyíti és élvezetesebbé teszi a zenehallgatást. Ezt figyelembe véve építettem be az olyan lehetőségeket, mint a hangfájlok feltöltése, lejátszási listák létrehozása, letöltési lehetőség, fórum, üzenetküldés, valamint az audió keresés. Az alkalmazás célja, hogy közösségi élményt is nyújtson, így elérhető benne a regisztráció, felhasználókeresés és privát üzenetküldés funkció is.

Fontos szempont volt, hogy az oldal ne csak online, hanem offline módban is működjön, és használatához elegendő legyen egy egyszerű webböngésző – így bárki számára könnyen hozzáférhető marad.

Kiemelt figyelmet fordítottam a rendszer alapos tesztelésére is, hiszen megfelelő tesztelés hiányában rejtett hibák vagy akár súlyos biztonsági problémák maradhatnak a rendszerben. A saját teszteseteim elvégzése után felkértem a célcsoport tagjait, hogy próbálják ki az alkalmazást valós használat közben, és jelezzék, ha hibát vagy hiányosságot tapasztalnak.

Az implementálás során szem előtt tartottam a kód újrafelhasználhatóságát is. Azokat a részeket, amelyek más webalkalmazásokban is hasznosak lehetnek, külön modulba szerveztem és elláttam részletes megjegyzésekkel, hogy a jövőben is könnyen újra lehessen használni őket.

Számomra fontos a zenét szerető közösség támogatása, így a jövőben – amennyiben igény mutatkozik rá – szívesen fejlesztem tovább az alkalmazást, hogy még teljesebb élményt nyújtson a felhasználók számára.

Irodalomjegyzék

[1] <https://www.tutorialspoint.com/firebase/index.htm>

[2] <http://www.color-hex.com/>

[3] Dr. Bilicki Vilmos - Webfejlesztési keretrendszerek előadás, 2025

[4] <https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB>

[5] <https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps>

[6] <https://www.w3schools.com/css/>

[7] <https://angular.dev/>

[8] <https://devdocs.io/angular/>

[9] <https://css-tip.com/>

[10] <https://fontawesome.com/v5/search>

[11] <https://firebase.google.com/docs/hosting/quickstart>

[12] <https://developer.mozilla.org/en-US/docs/Web/HTML>

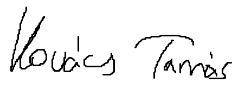
[13] <https://www.w3schools.com/html/>

[14] <https://www.w3schools.com/angular/>

NYILATKOZAT

Alulírott, Kovács Tamás, programtervező informatikus BSc szakos hallgató, kijelentem, hogy a szakdolgozatban ismertetettek saját munkám eredményei, és minden felhasznált, nem saját munkából származó eredmény esetén hivatkozással jelöltem annak forrását.

Szeged, 2025. május 19.



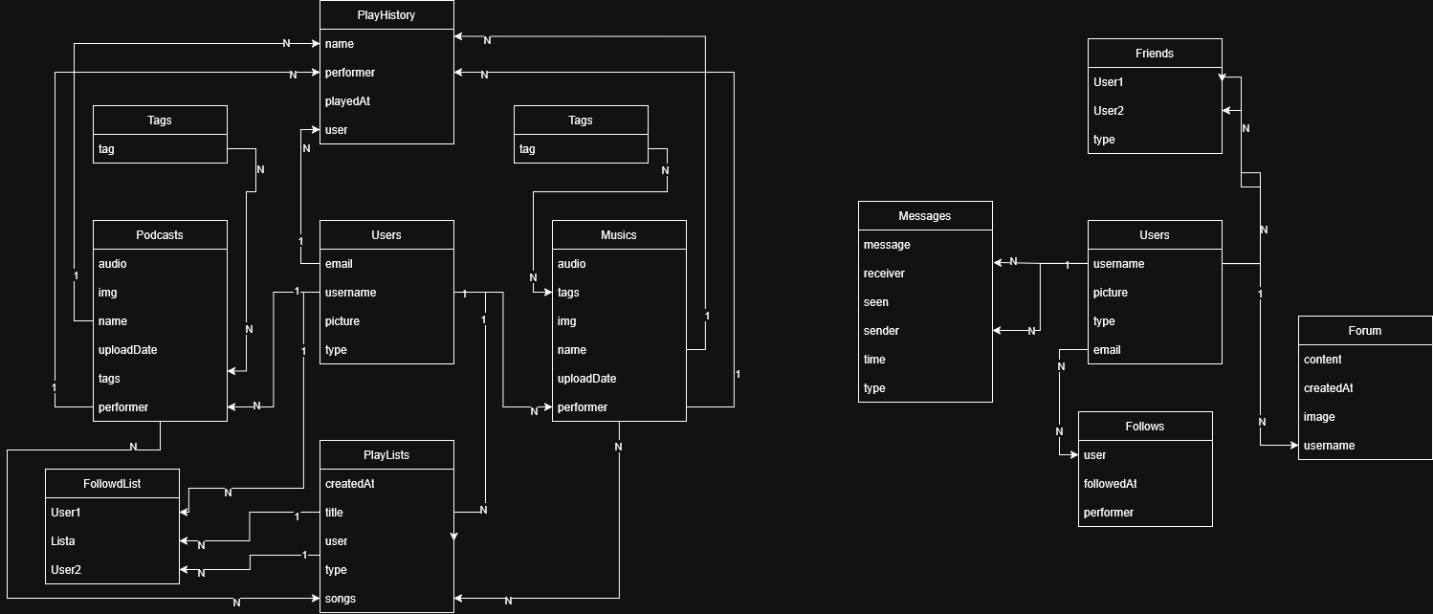
Aláírás

KÖSZÖNETNYILVÁNÍTÁS

Ezúton szeretném kifejezni hálámat mindazoknak, akik hozzájárultak szakdolgozatom elkészítéséhez. Külön köszönettel tartozom témavezetőmnek, Hegedűs Péter adjunktusnak az értékes szakmai útmutatásért és támogatásért. Hálás vagyok továbbá mindazoknak a zenekedvelő és podcasthallgató barátaimnak, akik véleményükkel, javaslataikkal és visszajelzéseikkel segítették a rendszer kialakítását és fejlesztését.

MELLÉKLETEK

1. Adatbázis teljes diagramja



2. Projekt publikus GitHub linkje

<https://github.com/4n1m3Hun/ZeeSharing>