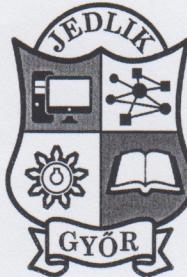




győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Záródolgozat feladatkiírás

Tanuló(k) neve¹: Szombathelyi Levente, Dörnyei Laura, Hegyi Szabolcs

Képzés: nappali

Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

„Lemezbázis” bakelit lemez adás-vétel weboldal

Konzulens: Horváth Norbert

Beadási határidő: 2023. 04. 28.

Győr, 2022. 10. 01

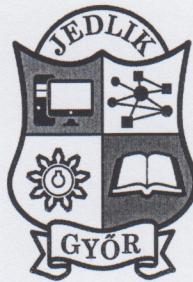
Módos Gábor
igazgató

¹ Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot.



győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Konzultációs lap²

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2022.11.11.	Témaválasztás és specifikáció	
2.	2023.03.10.	Záródolgozat készültségi fokának értékelése	
3.	2023.04.21.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2023. április 28.

tanuló aláírása

tanuló aláírása

tanuló aláírása

Szombathelyi Levente, Dörnyei Laura, Hegyi Szabolcs

„Lemezbázis” bakelit lemez adás-vétel weboldalának dokumentációja



Győri SZC Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium

Szoftverfejlesztő és -tesztelő technikus

2022/2023

1. Bevezetés

1.1. Projektünk ötlete

Az ötletünk elsősorban abból a felismerésből született, hogy hazánkban nincsen olyan online adás-vétel felület, amely kifejezetten lemezgyűjtőknek szól. Sok lemezrajongó nem magyarországi székhellyel rendelkező oldalakat használt eddig, amik nem mindenkor teljesen kiszolgálni az igényeiket, vagy nehézségekbe ütköztek a termék beszerzésével külföldről. Ennek ellenére azonban rengeteg lemezgyűjtő van hazánkban, akik szenvedélyteljesen gyűjti a különféle zenék kiadványait, valamint szeretnének könnyedén hozzájutni azokhoz itt Magyarországon. Erre a problémára szerettünk volna megoldást találni.

Az ötletünk másik motivációja az volt, hogy az online térben egy olyan közösséget hozzunk létre, ahol a zene iránt érdeklődők találkozhatnak egymással, megosztathatják tapasztalataikat, továbbá bővíthetik a zenei ismereteiket.

Első lépésként a weboldalnak karaktert szerettünk volna ölteni, ezért úgy határoztunk, hogy alkotunk egy logót, mely egyediséget sugároz, egyszerű, letisztult, de közben érzékelteti mivel is foglalkozik a piacunk. Erre egyikünk kitalálta, hogy alkalmazzunk egy logó készítő AI-t. Első lépésben megkerestük a megfelelő szolgáltatást, ami igényeinket kielégíti. A legfontosabb szempont az volt, hogy az AI legyen képes kezelní az ékezes betűket. Mikor ezt megtaláltuk bevittük a megfelelő adatokat weboldalunk tulajdonságairól, illetve külleméről, melyek akkorra körvonalazódtak bennünk. A kész, legenerált logókat kiszortíroztuk több szempont alapján, például, hogy fektetett-e, illetve illik-e az oldal profiljához. Amint megtaláltuk a számunkra legideálisabbnak tűnőt, nekiálltunk annak színeinek testreszabásának. Szerettük volna, ha a zöld színek dominálnak, ezért kipróbtuk a zöld különböző árnyalatait és kiválasztottuk a legígéretesebbet. Végső állomásként testre szabtuk a logó hátterét és már kész is volt a szemet gyönyörködtető, ám de nem túl hivalkodó logónk:

1.2. Szoftver célja

Projektünk az idei tanévben indult el, és célunk egy olyan online adás-vétel felület létrehozása volt, amely elsősorban lemezgyűjtőknek szól. Felületünk célkitűzése, hogy lehetővé tegye a felhasználók részére a régi, továbbá újabb zenei kiadványok megvásárlását és azok böngészését, valamint keresését a platform adatbázisában. Az oldal ezzel együtt lehetőséget biztosít a felhasználók részére, hogy eladják saját lemezeiket, ezáltal elősegítse a zenei kultúra és a közösségi élmény bővülését Magyarországon.

Oldalunkon kialakítottunk egy közösségi fórumot, amely lehetővé teszi a felhasználóink részére, hogy különféle témaikban kérdezzék, vitassanak, valamint megoldást keressék. A fórumot különböző témakörök szerint csoportosítottuk, így mindenki megtalálhatja a számára legmegfelelőbb beszélgetést. Lehetőség nyílik még többek között vélemények, tapasztalatok megosztására, ezen kívül a zenei élmények megbeszélésére. A fórum célja az, hogy a felhasználóink közösséget alkossanak, továbbá támogassák egymást az éppen aktuális témaikban. Fórumunk lényeges szerepet tölt be oldalunkon, mivel lehetőséget biztosít kapcsolatteremtésre egymással.

1.3. Funkciókról bővebben

A felhasználói fiók létrehozásáért regisztrációs felület felel. Név, email cím, illetve jelszó megadása kötelező. Mindezek végrehajtása után eltároljuk az újonnan regisztrált felhasználót adatbázisunkban. Ezt követően a bejelentkezés fülre kattintva a megadott adatokkal be is tud lépni az illető és elkezdődhett számára az oldal tényleges használata. Ha esetlegesen nem tud belépni, mivel elfelejtette jelszavát, az sem probléma ugyanis van oldalunkon „elfelejtett jelszó” funkció, melyre kattintva az adott email címére, kérhet egy új jelszót, melyet egyelőre a MailTrap.io weboldal használatával oldottunk meg.

A weboldalunk főmenüje letisztult, felül található egy navigációs sáv, melyekre kattintva az oldal átirányít a kiválasztott útvonalra (Főmenü, Eladó lemezek, Fórum). Az „Eladó lemezek” fülre kattintva az oldal kilistázza az összes eddig feltöltött lemezt, melynek szerepelnek a részletes adatai, valamint, hogy az eladó milyen lemez és borító állapotban hirdeti, és természetesen milyen áron válna meg tőle. Ebbe a blokkba kattintva kerülünk az eladó profiljára és az általa megadott adatok alapján felvehetjük vele a kapcsolatot. A „Fórum”

elemre klikkelve a már éppen felvetett témák jelennek meg, alatta esetleges hozzászólásokkal. De ha maga a felhasználó nem találja a számára megfelelőt, vagy csak szeretne egy újat kezdeni Ő maga, lehetősége nyílik rá a „Kérdés feltétele” gombra kattintva.

Sikeressé bejelentkezést követően a navigációs sáv jobb felső sarkában található „Regisztráció”, valamint „Bejelentkezés” fül átvált „Profil” -ra, melyre kattintva a következő legördülő lista jelenik meg: Feltöltés, Termékek szerkesztése, Kívánságlista, Üzenetek, Kijelentkezés. Ezeket a funkciókat csak a már regisztrált és bejelentkezett felhasználók érhetik el. A „Feltöltés” feliratra kattintva, az oldal elvezérel a lemezek feltöltésére szolgáló felületre, ahol a megjelenő mezőket kitölve és az utolsó gombra kattintva felkerülhet az oldalra az eladásra kínált bakelit.

2. A fejlesztői csapat

2.1. A csapat tagjai és feladataik

Év elején megkaptuk a feladatot, miszerint hármas csoportokban kell dolgozni a projektfeladaton, melynek hallattán szerintem mindenki örült, ugyanis ezzel a módszerrel sokkal közelebb kerültünk az igazi szofverfejlesztéshez, melynek nagy része a csapatmunkára épül, enélkül nem fejezhető be egy hasznosnak és jónak mondható projekt. Belecsöppentünk a programozáshoz szükséges kommunikáció, és egyéb szükséges képességek elsajátításához. A csapatunk tagjai Győri Szakképzési Centrum Jedlik Ányos Gépipari és Informatikai Technikum és Kollégium 2/14A osztályos tanulói: Szombathelyi Levente, Dörnyei Laura, Hegyi Szabolcs. Alapvetően és nyilvánvalóan a feladatot frontend/backend feladatrészre osztottuk fel, még a projekt megkezdése előtt. Dörnyei Laura vállalta és felelt a projekt frontend(előoldal) részéért, tehát ő csinálta a weboldalon megjelenő felületeket, és azok megfelelő működéséhez backend előhívó kódokat. Szombathelyi Levente és Hegyi Szabolcs közösen írta meg az adatbázismodellt, valamint a backendet(hátoldalt), amely az adatokat kezeli, tárolja és feldolgozza azokat a felhasználói interakciók alapján, amelyek a felhasználók által a frontend segítségével történnek. A backend hamarabb kész lett, kevesebb probléma, és akadályba ütközés volt ezen a téren, ezért ahol tudtunk próbáltunk a Laura feladatát könnyíteni egyszerű esztétikai megoldásokkal az oldalon, hogy neki ezekkel már ne kelljen olyan sokat foglalkoznia.

2.2. A csapatmunka megvalósítása

A csapatmunka megvalósításához a kezdetektől fogva a jól ismert GitHubot használtuk, ahol Laura létrehozott egy „lemezbázis” nevű projektet, ahova mindenki kapott egy meghívó linket, melyhez a csapat többi tagja csatlakozott. Eleinte ide mindig feltöltöttük a legújabb verziót, majd később ismét Laura hívta fel a figyelmünket a GitHub Desktop nevű alkalmazás használatára, ahol csak hozzáadtuk projektünket az alkalmazáshoz, megosztottuk egymással és onnantól kezdve bármi változtatást csináltunk a projekten belül azt fel tudtuk „push”-olni az alkalmazásba, melyet többiek könnyedén saját munkaeszközükre le tudták „pull”-olni és mindenkorral frissebb, legújabb kódú változattal tudtuk folytatni a munkát bármilyen messze is voltunk egymástól.

A közös munka érdekében rendszeres megbeszéléseket tartottunk a Discord nevű alkalmazáson, ahol összefoglaltuk, hogy hogyan is haladtunk valójában az elmúlt időben, valamint mik a következő tervezetek, esetlegesen, ha valaki elakadt próbáltuk közösen kiküszöbölni a problémát és megoldani azt. Körülbelül heti rendszerességgel történtek ilyen megbeszélések a munka elkezdésétől számítva.

3. Adatbázis

3.1. Az adatbázis háttér technológiája, adatbáziskezelő program

Az idei évben találkoztunk először a MongoDB adatbázis szoftverrel, amely annyira tetszett nekünk fejlesztői szempontból, hogy ezt választottuk weboldalunk adatbázisának tárolására. A MongoDB nyílt forráskódú dokumentumorientált adatbázis szoftver, amelyekben az adatok struktúrája rugalmasabb, mint a hagyományos relációs adatbázisokban. Emellett a MongoDB nagyon jól skálázható, ami azt jelenti, hogy könnyen lehet bővíteni az adatbázist, ha növekszik a felhasználók száma vagy a tárolt adatok mennyisége. A NoSQL adatbázisszerverek közé tartozik. A dokumentumokat JSON-szerű formátumban tárolja (BSON). Az adatok felvitelét, kollekciók létrehozására a „*MongoDB Compass*” nevű hatékony grafikus felhasználói felületet használtuk, mely a MongoDB-adatok vizuális környezetben történő lekérdezéséhez, összesítéséhez és elemzéséhez is megfelelő.

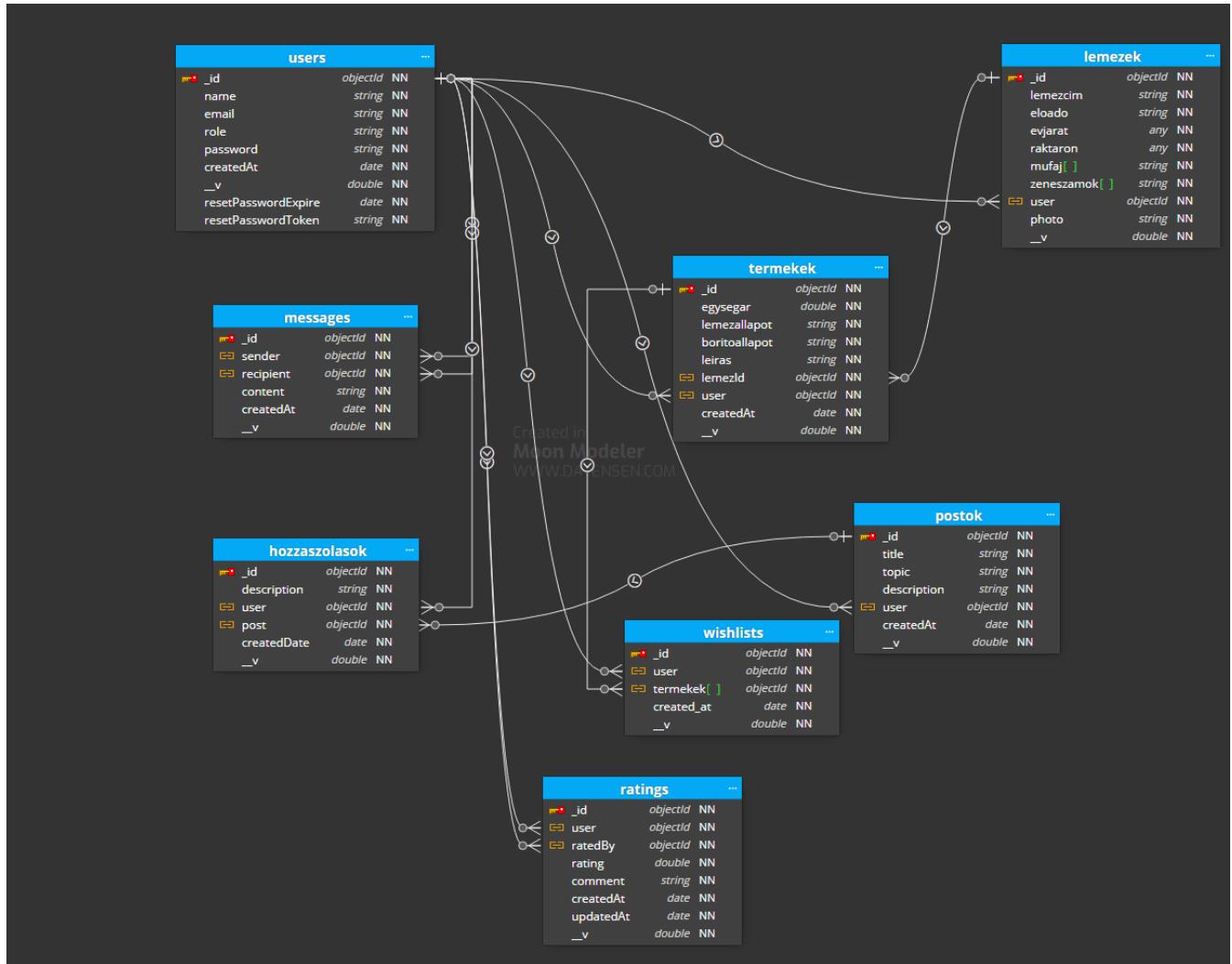
3.2. Az adatbázis leírása, magyarázata

A MongoDB saját fejlesztésű weboldalára való regisztrálás után, létrehoztuk az oldalon egy saját „cluster” -t, melynek nyilvánvalóan az oldalunk nevét adtuk meg. A MongoDB cluster egy olyan elosztott adatbázis rendszer, amely több szerveren tárolja az adatokat, és lehetővé teszi a nagyobb rendelkezésre állást és skálázhatóságot. Itt a létrehozott cluster-nél kértünk egy kapcsolódási címet, melyet a MongoDB Compasson belül a „New Connection” fülre kattintva tudtunk kapcsolódni a létrehozott adatbázishoz. Egyszerű könnyen átláthatóság miatt használtuk a MongoDB Compass applikációt, melyben a létrehozott „lemezbázis” nevű adatbázishoz a következő kollekciókat adtuk hozzá: Hozzaszólások, Lemezek, Messages, Postok, Ratings, Termékek, Users, Wishlist. Ezeken belül vannak tárolva az adatok dokumentum formában. Maga az adatbázis kialakítása látható a következő képen:

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
hozzaszolasok	7	843B	121B	36KB	1	36KB	36KB
lemezek	22	11.06KB	515B	44KB	2	72KB	36KB
messages	3	438B	146B	36KB	1	36KB	36KB
postok	7	1.08KB	158B	36KB	1	36KB	36KB
ratings	3	411B	137B	36KB	1	36KB	36KB
termek	9	1.74KB	198B	36KB	1	36KB	36KB
users	21	4.17KB	204B	36KB	2	72KB	36KB
wishlists	26	2.28KB	90B	36KB	1	36KB	36KB

Adatbázisunk a MongoDB Compass alkalmazáson belül

3.3. Adatbázis diagramja



MongoDB adatbázisunk modellje

4. A backend fejlesztése

4.1. A fejlesztői környezet, adatbázis kommunikáció

A backendet a csapatunk Node.js-ben írta, mely egy JavaScript-alapú szerveroldali programozási nyelv. A kódbázisunk alapját az Express keretrendszer adta, amely egy nagyon népszerű és könnyen használható Node.js-alapú webalkalmazás-keretrendszer. Összességében a Node.js és az Express együtt alkotják a modern webalkalmazások egyik legnépszerűbb szerveroldali stackjét, amely hatékony és könnyen kezelhető megoldást kínál az adatkezelésre és az API-k építésére.

Az adatbázisunk eléréséhez node.js-en belül a mongoose-t használtuk. A mongoose egy ODM (Object Data Modeling) keretrendszer, amely lehetővé teszi a MongoDB adatbázis könnyebb kezelését a Node.js környezetben. Az adatok objektumként való kezelése egyszerűbb és intuitívabb szintaxisat tesz lehetővé, mint a sima MongoDB API. Emellett a mongoose biztosít egy sokoldalú validációs rendszert, amely segít az adatbázis konzisztenciájának és integritásának biztosításában. Ezért választottuk a mongoose-t az adatbázisunk

```
backend > js server.js > ...
1 const path = require('path')
2 const express = require('express')
3 require('dotenv').config() // A .env fájlt olvassa
4 const morgan = require('morgan')
5 const fileUpload = require('express-fileupload')
6 const cookieParser = require('cookie-parser')
7 const errorHandler = require('./middleware/error')
8
9
10 const mongoose = require("mongoose");
11 mongoose.set("strictQuery", true);
12 const mongoString = process.env.DATABASE_URL;
13 mongoose.connect(mongoString);
14 const database = mongoose.connection;
15
16 database.on("error", (error) => {
17   | console.log(error);
18 });
19
20 database.once("connected", () => {
21   | console.log(`Database Connected ${database.host}`);
22 });
23
24
```

Mongoose összeköttetés, dotenv

összekötéséhez:

A fenti parancsok által a .env fájlunkban megadott adatbázis csatlakozási linket megadva már kommunikál is a backend az adatbázissal. A fájl tartalma itt látható:

```
backend > .env
1 PORT=3000
2 DATABASE_URL = mongodb+srv://asd:asd@cluster0.3000-es port, valamint adatbázis URL
3
```

4.2. A fejlesztés menetrendje, mérföldkövek

Első lépésként miután már a backend alkalmazásunk egy szervert indít a 3000-es porton, amely várja a beérkező kéréseket meghatározott végpontokon, elkezdtük a „models” mappán belül felvenni az adatbázisunk kollekcióinak adatainak eltárolásához szükséges modeleket. Ehhez egy példát csatoltan láthatják:

```
backend > models > JS Lemezjs > ...
1  const mongoose = require("mongoose");
2
3
4  const LemezSchema = new mongoose.Schema({
5    lemezcím: {
6      type: String,
7      required: [true, 'Lemezcím mező kitöltése közelező!'],
8      unique: true
9    },
10   eloadó: {
11     type: String,
12     required: [true, 'Előadót meg kell adnia!']
13   },
14   evjárat: {
15     type: Number,
16     required: [true, 'Évjáratot meg kell adnia!']
17   },
18 },
19   mufaj: {
20     type: Array,
21     required: [true, 'Műfajt/műfajokat meg kell adnia!']
22   },
23   zeneszamok: {
24     type: Array,
25     required: [true, 'Zeneszámot/zeneszámokat meg kell adnia!']
26   },
27   photo: {
28     type: String
29   },
30   user: {
31     type: mongoose.Schema.ObjectId,
32     ref: 'User',
33     required: true
34   }
35 },
36 toJSON: { virtuals: true },
37 toObject: { virtuals: true }
38 });
39
40 // Fordított populate virtual segítségével
41 LemezSchema.virtual('termek', {
42   ref: 'Termek',
43   localField: '_id',
44   foreignField: 'lemezId',
45   justOne: false
46 })
47
48 module.exports = mongoose.model("Lemez", LemezSchema, "lemezek");
```

models/Lemez.js példa

Miután felvettünk minden kollekciót a megfelelő „model” -jét, következő lépésként létrehoztuk a controllers mappát, másik nevén a vezérlőket, ahol felvettük a különböző metódusokat. Példaként a következő kód részletben több metódus található a controllers/lemezek.js mappán belül, amelyek különböző funkciókat látnak el az alkalmazásban. Az egyes metódusok a következők:

„getLemezek”: Az összes lemez listázza az adatbázisból. Lehetőség van szűrésre előadó, évjárat vagy műfaj alapján. A lemezekhez tartozó termékekkel is lekérdezi és azokat is megjeleníti.

„getLemez”: Egyetlen lemez adatait kéri le az adatbázisból az azonosítója alapján. Itt is tartalmazza a lemezekhez tartozó termékeket.

„createLemez”: Új lemez hozzáadása az adatbázishoz. A lemez adatait a kérés testében kapja meg a metódus, és a metódus az adatbázisba menti a lemezt.

„updateLemez”: Egy meglévő lemez adatainak módosítása az azonosítója alapján. A módosításokat szintén a kérés testében kapja meg a metódus, és a metódus frissíti az adatbázisban a lemez adatait.

Az előnye annak, hogy ezeket a metódusokat külön hoztuk létre, hogy könnyen karbantarthatóvá és bővíthetővé teszi az alkalmazást. Egy adott funkcióhoz tartozó kódok elkülönítése megkönnyíti a hibakeresést és az új funkciók hozzáadását is. Ezenkívül, az is segít, hogyha később szükség van valamely metódus módosítására, akkor azt külön lehet végezni anélkül, hogy a többi metódust is érintené.

```
backend > controllers > JS lemezek.js > getLemezek > getLemezek
64 // @route GET /api/lemezek/:id
65 // @access Public
66 exports.getLemezek = async (req, res, next) => {
67   try {
68     const lemez = await Lemez.findById(req.params.id).populate({
69       path: 'termek',
70       populate: [
71         { path: 'user',
72           select: 'name email',
73         }
74       ],
75     });
76     if (!lemez) {
77       return res.status(400).json({ success: false, msg: "Not found" });
78     }
79     res.status(200).json({ success: true, data: lemez });
80   } catch (error) {
81     next(new ErrorResponse(`Lemez id ${req.params.id} helytelen`, 404));
82   }
83 }
84 // @desc Create new lemez
85 // @route POST /api/lemezek/
86 // @access Private
87 exports.createLemez = async (req, res, next) => {
88   try {
89     req.body.user = req.user.id;
90     const lemez = await Lemez.create(req.body);
91     res.status(201).json({ success: true, data: lemez });
92   } catch (error) {
93     next(error);
94   }
95 };
96
97 // @desc Update lemez
98 // @route PUT /api/lemezek/:id
99 // @access Private
100 exports.updateLemez = async (req, res, next) => {
101   try {
102     const lemez = await Lemez.findByIdAndUpdate(req.params.id, req.body, {new: true, runValidators: true});
103     if (!lemez) {
104       return res.status(400).json({success: false, msg: 'Not found'});
105     }
106     res.status(200).json({success: true, data: lemez});
107   } catch (error) {
108     next(error);
109   }
110};
```

controllers/lemezek.js példa

Miután felvettünk minden kollekcionak a megfelelő vezérlőjét, kötött az útválasztás a routes mappán belül. Az útvonalakat Express objektum függvényeivel határozzuk meg. A kérések különböző típusaihoz különböző metódusok tartoznak, amelyek azok fogadására szolgálnak. Az egyes végpontok címeit egy karakterlánc határozza meg, amely után opcionálisan egy paraméter is követhet. Ezenkívül meg kell adni egy visszahívó függvényt, amely fut a végpont hívásakor. Külön fájlokban tároljuk a metódusok funkciójuk és elérhetőségük alapján, mivel így könnyebben átlátható és így a kód sokkal letisztultabb, valamint hiba sokkal egyszerűbb kiigazodni a problémán, majd megoldani azt.

A szöveg alatti példában látható, hogy adott végpontra küldött, GET, POST, PUT, DELETE metódussal hívott kéréseket kötnek össze a különböző „controller” mappából beimportált állományban szereplő visszahívó függvényekkel.

```
const express = require("express");
const {
  getLemez,
  getLemezek,
  createLemez,
  updateLemez,
  deleteLemez,
  lemezPhotoUpload
} = require("../controllers/lemezek");

const advancedResults = require('../middleware/advancedResults')

const termekRouter = require('./termekkek');

const router = express.Router();
```

routes/lemezek.js példa

```
router.use('/:lemezId/termekkek', termekRouter)

router.route('/:id/photo')
  .put(protect, authorize('publisher', 'admin', 'user'), lemezPhotoUpload)

router.route("/")
  .get(getLemezek)
  .post(protect, authorize('publisher', 'admin', 'user'), createLemez);

router
  .route("/:id")
  .get(getLemez)
  .put(protect, authorize('publisher', 'admin', 'user'), updateLemez)
  .delete(protect, authorize('publisher', 'admin', 'user'), deleteLemez);

module.exports = router;
```

routes/lemezek.js példa

Az első mérföldkövünk, annak tekinthető amikor a terveink szerint létrehozott modellek, vezérlők és útválasztások megfelelő összhangban egyaránt, egymásra felépítve a jól működtek a tesztelés fázisban. Utána következtek a finomítások, egy két populattel, esetleges új metódusok létrehozása a frontend szükségletei szerint. Ha valamit az eddig kitalált kóddal, nem tudtunk megjeleníteni írtunk még hozzá, hogy az a tőlünk telhető legjobb felhasználói élményt nyújtsa, mind funkcióiban, mind megjelenítés szempontjából.

```
exports.getRatings = async (req, res, next) => {

  try {
    const userId = req.params.userId;
    const ratings = await Rating.find({user: userId})
      .populate({
        path: 'ratedBy'
      })
      .populate({
        path: 'user'
      });

    res.status(200).json({ success: true, data: ratings });

  } catch (error) {
    res.status(500).json({ success: false, msg: 'Hiba történt az értékelések lekérése során', error });
  }
};
```

Populate példa 1.

```
// @desc Get single post
// @route GET /api/postok/:id
// @access Public
exports.getPost = async (req, res, next) => {
  try {
    const post = await Post.findById(req.params.id).populate({ path: 'user' })
      .populate({ path: 'hozzasolasok', populate: { path: 'user' } });

    if (!post) {
      return next(new ErrorResponse(`Nincs ilyen id-jű post: ${req.params.id}`, 404))
    }
    res.status(200).json({ success: true, data: post });
  } catch (error) {
    next(error);
  }
};
```

Populate példa 2.

4.3. Hibára példa és annak kiküszöbölése, tesztelés

A models/Termek.js fájlban és azon belül a TermekSchema-ban fel lett véve egy „leírás” property, melynek kitöltése kötelező, ám nem muszáj egyedinek lennie, ennek ellenére véletlen felvettünk hozzá egy „unique: true” változót, melynek következtében a leírásnak lett egy id-je, melyre egy duplikált kulcs hibaüzenetet írt ki egy GET kérés tesztelésnél. Erre megoldásként a MongoDB Compasson belül, a terminált használva kitöröltük a leírásnak létrehozott indexet, mellyel megoldódott a probléma és működött a lekérdezés.

```
>_MONGOSH
> use lemezbazis
< switched to db lemezbazis
> db.termek.dropIndex({leiras: 1})
< {
  nIndexesWas: 2,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1680080550, i: 18 }),
    signature: {
      hash: Binary(Buffer.from("b88e9103ceac22919c2ee5781a6f552e8a889cad", "hex")),
      keyId: Long("7182179324668149762")
    }
  },
  operationTime: Timestamp({ t: 1680080550, i: 18 })
}
Atlas atlas-cddr3k-shard-0 [primary] lemezbazis >
```

DuplaIndex hiba

A Thunder Client egy olyan bővítmény a Visual Studio Code-hoz, amely lehetővé teszi az API-k és a webalkalmazások tesztelését. Ezt a bővítményt használtuk a backend teszteléshez, mivel lehetővé teszi HTTP kérések küldését és a válaszok megtekintését anélkül, hogy szükség lenne egy különálló alkalmazásra vagy eszközre. A Thunder Client könnyen kezelhető és konfigurálható, és számos hasznos funkcióval rendelkezik, például az előzmények, a környezet változók, az automatikus kódgenerálás és a tesztek futtatása a kérésre adott válaszok ellenőrzéséhez. Ezek a funkciók lehetővé teszik a hatékonyabb és megbízhatóbb API tesztelést, ami fontos szerepet játszik a backend fejlesztésben és karbantartásában.

5. REST API

5.1. Bevezető

REST API (Representational State Transfer Application Programming Interface) egy programozási interfész, amely lehetővé teszi az alkalmazások számára, hogy kommunikáljanak egymással és adatokat osszanak meg egymás között az interneten keresztül. A REST API-k szabványos formátumban adnak vissza adatokat, általában JSON vagy XML formájában. Nálunk a tesztelésnél JSON formában írtuk, ki, valamint vittük fel az adatokat.

Mint már feljebb is említettem az API kérések tesztelésére és kipróbálására Thunder Client bővítményt alkalmaztuk. Az API kéréseket a fejlesztési fázisban egy a 3000-as porton futó Node.js webszerver fogadta. A végpontok ebből kifolyólag „localhost:3000/api/” kezdetűek és az utána lévő paraméterek befolyásolják a kérés válaszának tartalmát.

5.2. Azonosítás és hitelesítés

Ahhoz, hogy a kéréseink nagy részét el lehessen érni, a belépést meg kellett oldanunk. Hozzáférési azonosításhoz mi JWT tokent használtunk, melyet egy Environmentben felvettünk és tároltunk, majd minden kérésnél az Authentication fül, Bearer mezőjébe a felvett változó nevét raktuk, ami az eltárolt JWT tokent tartalmazza, mellyel megtörténik az azonosítás és kezdődhet a tesztelés.

The screenshot shows two main sections of the Thunder Client application.

Environment Section: This section is titled "Environment" and contains a table for managing environment variables. It has a header row with "Variable Name" and "Value". Below this, there is a single entry for "TOKEN" with the value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0M2Y5ZWfkMjRkNDMzYzkwMGZmOTNhNyIsImIhdCI6MTY4MjQ5Njk0OSwiZ... . There is also a checkbox labeled "Link to .env file".

Auth Section: This section is titled "Authentication/Bearer" and includes tabs for "Query", "Headers", "Auth", "Body", "Tests", and "Pre Run". The "Auth" tab is selected and has a sub-tab "Bearer" which is also selected. Under the "Auth" tab, there is a field labeled "Bearer Token" containing the placeholder "{{TOKEN}}".

5.3. Tesztelések

5.3.1. Felhasználó regisztrálása, bejelentkezése

Tunder Client-be „teszt” Kollekcióból beimportálásával megtekinthető minden tesztelésünk. Elérési útvonala: lemezbazis/backend/thunder-collection_teszt.json Egy felhasználó regisztrálás a következőképpen néz ki: `POST localhost:3000/api/auth/register`

Body/Json Content:

```

1  {
2    "name": "Hegyi Szabolcs",
3    "email": "hegyi.szabolcs2003@gmail.com",
4    "password": "Jedlik1245",
5    "role": "user"
6
7  }

```

JSON Content

Amire a válasz:

Status: 200 OK Size: 0 Bytes Time: 237 ms

Response	Headers 11	Cookies 1	Results	Docs	{}	≡
<pre> 1 { 2 "success": true, 3 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 .eyJpZCI6IjY0NDhmOWMzN2Q2MDlMjB1Yjg3MmRjNCIsImIhdCI6MTY4MjUwNDEzMiwZXhwIjoxNjg1MDk2MTMyfQ .5c5qV7cxzLvLbm48LGU4NVkKox-ulXBReRji6qqk" 5 6 } </pre>					<input type="button" value="Copy"/>	

Response text

Eredmény:

The screenshot shows the MongoDB Compass interface with the database 'lemezbazis' selected. Under the 'lemezbazis' section, the 'users' collection is highlighted. The interface displays two documents. The second document, which has a yellow box around it, represents the registered user with the following data:

```

{
  "_id": ObjectId('64489c37d669b20eb872dc4'),
  "name": "Hegyi Szabolcs",
  "email": "hegyi.szabolcs2003@gmail.com",
  "role": "user",
  "password": "52a5185a5a5qm2iFpFE2upo2d0vBe8C0jUlw73Ifu0Jsgx0Qr4w01MK25Tw",
  "createdAt": 2023-04-24T09:07:03.295+00:00,
  "__v": 0
}

```

Adatbázisban megjelenő regisztrált felhasználó

Egy felhasználó bejelentkezése a következőképpen néz ki: `POST localhost:3000/api/auth/login`

Body/JSON/Content:

```
Json Content
1  {
2    "email": "szombathelyi.levi64@gmail.com",
3    "password": "Nelépjbe123"
4 }
```

Json Content

Amire a válasz:

```
Status: 200 OK  Size: 0 Bytes  Time: 114 ms
Response Headers Cookies Results Docs { } ≡
1  {
2    "success": true,
3    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJpZCI6IjyaM2Y5ZWfkMjRkNDMzYzkwMGZmOTNhNyIsImIhdCI6MTY4MjQ5Njk0OSwiZXhwIjoxNjg1MDg4OTQ5fQ
        .c-ZC03QBK4xdgG7IlvDEhecFPi9mdjkL0PWoA9r-lxY"
4 }
```

Response text

Eredményül, hogy teszteljük sikerült-e a bejelentkezés valójában, a következő végpont használatával kideríthetjük ki is van bejelentkezve: `GET localhost:3000/api/auth/me`

Válasz üzenet:

```
Status: 200 OK  Size: 327 Bytes  Time: 53 ms
Response Headers Cookies Results Docs { } ≡
1  {
2    "success": true,
3    "data": {
4      "_id": "643f9ead2d433c900ff93a7",
5      "name": "Szombathelyi Levente",
6      "email": "szombathelyi.levi64@gmail.com",
7      "role": "admin",
8      "createdAt": "2023-04-19T07:56:29.740Z",
9      "__v": 0,
10     "resetPasswordExpire": "2023-04-22T17:18:41.501Z",
11     "resetPasswordToken": "113313adadb2c622318b1db51d2ea323bbfbaba6d6a87b7fad367a2b80658654"
12   }
13 }
```

Response text

Itt láthatjuk, hogy akivel bejelentkeztünk a GET kérés ōt adja vissza, ez után jöhét a többi funkció tesztelése, mihez szükséges a bejelentkezés.

5.3.2. Termékek lekérdezése, feltöltése, módosítása, törlése

Termékek össze vannak kapcsolva a Lemezekkel egy populattel. Két féle GET kérésünk van a termékekkel, ami az összes terméket lekéri, valamint ami csak egy adott indexel rendelkező terméket kér csak le. Elsőnek nézzük meg az összes terméket kilistázó kérést. Ezt a következő végponton érhetjük el: `GET localhost:3000/api/termeket/`

Válasz:

Status: 200 OK Size: 2.33 KB Time: 123 ms

Response Headers Cookies Results Docs

```
1  {
2      "success": true,
3      "count": 9,
4      "data": [
5          {
6              "_id": "643fcf7eed4c9d4db5b90cia",
7              "egysgar": 8000,
8              "lemezallapot": "használt",
9              "boritoallapot": "használt",
10             "leiras": "proba",
11             "lemezID": "63bd3f2e6b99d5fd58c1588",
12             "user": "643f9ead24d433c900ff93a7",
13             "createdAt": "2023-04-19T11:24:46.523Z",
14             "__v": 0
15         },
16         {
17             "_id": "643fd35a6d941cfbef8b1828",
18             "egysgar": 8000,
19             "lemezallapot": "új",
20             "boritoallapot": "új",
21             "leiras": "elég cool szntem azért ilyen drága xsd",
22             "lemezID": "63bd38bf6b99d5fd58c1587",
23             "user": "643f9ead24d433c900ff93a7",
24             "createdAt": "2023-04-19T11:41:14.378Z",
25             "__v": 0
26         },
27         {
28             "_id": "64416b719f4291bc65c012d7",
29             "egysgar": 20000,
30             "lemezallapot": "használatlan",
31             "boritoallapot": "újszerű",
32             "leiras": "Eredeti, első kiadás",
33             "lemezID": "63c524281d1e74eac85054",
34             "user": "643f9ead24d433c900ff93a7",
35             "createdAt": "2023-04-20T16:42:25.228Z",
36             "__v": 0
37         }
],
```

Response text

Mint látható a count paraméteren, hogy sikeresen kilistázta mind a 9 jelenleg eladó terméket.

Ha csak egy adott terméket szeretnénk lekérni a következő végpont használatával ez is megoldható: `GET localhost:3000/api/termeket/:id`

Válasz üzenetként:

```
{
  "success": true,
  "data": {
    "_id": "643fcf7eed4c9d4db5b90c1a",
    "egysegar": 8000,
    "lemezallapot": "használt",
    "boritoallapot": "használt",
    "leiras": "proba",
    "lemezId": [
      {
        "_id": "63bd3f2e66b99d5fd58c1588",
        "lemezcim": "Brothers In Arms",
        "eloado": "Dire Straits",
        "evjarat": 2021,
        "raktaron": 3,
        "mufaj": [
          "Rock"
        ],
        "zeneszamok": [
          "So Far Away",
          "Money For Nothing",
          "Walk Of Life",
          "Your Latest Trick",
          "Why Worry",
          "\tRide Across The River",
          "The Man's Too Strong",
          "One World",
          "Brothers In Arms"
        ],
        "user": "6424503e8f138279fd1eb85c",
        "photo": "photo_63bd3f2e66b99d5fd58c1588.jpg",
        "id": "63bd3f2e66b99d5fd58c1588"
      },
      {
        "user": "643f9ead24d433c900ff93a7",
        "createdAt": "2023-04-19T11:24:46.523Z",
        "__v": 0
      }
    ]
  }
}
```

Response text

Látható a lekért termék minden adata, valamint a lemez adatai is megfelelően megjelennek a `populate` parancs pontos használatának köszönhetően. Azonban a termékeket nem csak lekérni és megjeleníteni lehet az oldalon, hanem feltölteni is, amit a következi POST kéréssel lehet megtenni: `POST localhost:3000/api/lemezek/:lemezId/termeket`

Body/JSON/Content: kötelező megadni az alábbi paramétereket:

Json Content

```

1  {
2    "egysegar": 5000,
3    "lemezallapot": "használt",
4    "boritoallapot": "újszerű",
5    "leiras": "Eredeti, első kiadás"
6  }
```

JSON Content

Válasz üzenet:

```

1  {
2      "success": true,
3      "data": {
4          "egysegar": 5000,
5          "lemezallapot": "használt",
6          "boritoallapot": "újszerű",
7          "leiras": "Eredeti, első kiadás",
8          "lemezId": "643ac094da3a2745e9d63070",
9          "user": "643f9ead24d433c900ff93a7",
10         "_id": "644908067d609b20eb872ddf",
11         "createdAt": "2023-04-26T11:16:22.645Z",
12         "__v": 0
13     }
14 }
```

Response text

És végül a következő POST kérés eredménye melyen látszik, hogy megjelenik az adatbázisunkban:

The screenshot shows the MongoDB Compass interface with the following details:

- Collection:** lemezbazis.termek
- Documents:** 9
- Indexes:** 1
- Filter:** Type a query: { field: 'value' }
- Buttons:** ADD DATA, EXPORT COLLECTION, Reset, Find, More Options
- Document Preview:**
 - createdAt: 2023-04-26T11:16:22.645Z
 - __v: 0
 - Object:** (highlighted by a yellow box)
 - _id: ObjectId('644908067d609b20eb872ddf')
 - egysegar: 5000
 - lemezallapot: "használt"
 - boritoallapot: "újszerű"
 - leiras: "Eredeti, első kiadás"
 - lemezId: ObjectId('643ac094da3a2745e9d63070')
 - user: ObjectId('643f9ead24d433c900ff93a7')
 - createdAt: 2023-04-26T11:16:22.645Z
 - __v: 0

Adatbázisban megjelenő termék

A feltöltött termékeket terszmészetesen lehet módosítani, amit a következő végponttal tehetünk meg. `PUT localhost:3000/api/termek/:termekId`. JSON Contentbe ugyanazon paraméterek megadása kötelező a módosításhoz, mint ami a feltöltéshez kellett:

Body/JSON/Content:

```
Json Content
1  {}
2  "egysegar": 2500,
3  "lemezallapot": "új",
4  "boritoallapot": "új",
5  "leiras": "Eredeti, első kiadás"
6  []
```

JSON Content

Válasz:

```
{
  "success": true,
  "data": {
    "_id": "644908067d609b20eb872ddf",
    "egysegar": 2500,
    "lemezallapot": "új",
    "boritoallapot": "új",
    "leiras": "Eredeti, első kiadás",
    "lemezId": "643ac094da3a2745e9d63070",
    "user": "643f9ead24d433c900ff93a7",
    "createdAt": "2023-04-26T11:16:22.645Z",
    "__v": 0
  }
}
```

Response text

Eredményül az adatbázisba való változtatások is megtörténtek:

```
_id: ObjectId('644908067d609b20eb872ddf')
egysegar: 2500
lemezallapot: "új"
boritoallapot: "új"
leiras: "Eredeti, első kiadás"
lemezId: ObjectId('643ac094da3a2745e9d63070')
user: ObjectId('643f9ead24d433c900ff93a7')
createdAt: 2023-04-26T11:16:22.645+00:00
__v: 0
```

Adatbázisban megjelenő változás

Végül, ha a terméket már le szeretné törölni a felhasználója, van rá lehetősége. A következő végpontot hoztuk létre, amivel a lekérdezés végbemegy: **DELETE** <localhost:3000/api/termek/termekId>.

A lekérdezést elküldése után a következő választ üzenetet kapjuk:

```
Status: 200 OK  Size: 26 Bytes  Time: 102 ms

Response Headers 10 Cookies Results Docs

1  {
2    "success": true,
3    "data": {}
4 }
```

Response text

A termékekhez képet is lehet feltölteni a következővel: **PUT** <localhost:3000/api/lemezek/:lemezId/photo>. A kép feltöltéséhez Body/Form/Files elérési útvonalon kell feltöltenünk a kiválasztott képünket.

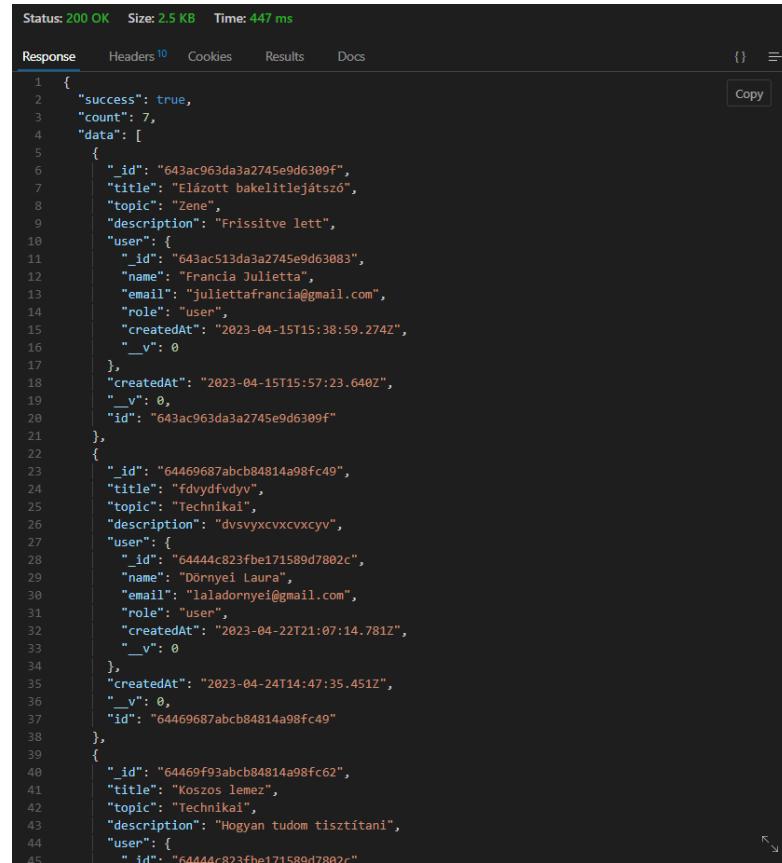
5.3.3. Posztok lekérdezése, létrehozása, frissítése, törlése

Az oldalon megtalálható egy közösségi fórum, ahol a felhasználók tudnak feltölteni posztokat, különböző témaiban, többek között kereshetnek lemezt vagy megvitathatnak bizonyos dolgokat. A többi felhasználó rá tud kattintani a számukra érdekes, vagy hasznos posztra és

tudnak kommentet is írni. Az összes poszt lekérdezéséhez a következőt kell beírnunk: **GET**

`localhost:3000/api/postok/`

Válaszul a következőt kapjuk:



The screenshot shows a Postman interface with the following details:

- Status: 200 OK
- Size: 2.5 KB
- Time: 447 ms
- Response tab selected
- Headers, Cookies, Results, Docs tabs available
- Copy button available
- JSON response content (lines 1-44) showing 7 posts with their respective details like title, topic, user information, and creation date.

```

1  {
2    "success": true,
3    "count": 7,
4    "data": [
5      {
6        "_id": "643ac963da3a2745e9d6309f",
7        "title": "Eláztott bakelittlejátszó",
8        "topic": "Zene",
9        "description": "Frissítve lett",
10       "user": {
11         "_id": "643ac513da3a2745e9d63083",
12         "name": "Francia Julietta",
13         "email": "juliettafrancia@gmail.com",
14         "role": "user",
15         "createdAt": "2023-04-15T15:38:59.274Z",
16         "__v": 0
17       },
18       "createdAt": "2023-04-15T15:57:23.640Z",
19       "__v": 0,
20       "id": "643ac963da3a2745e9d6309f"
21     },
22     {
23       "_id": "64469687abcb84814a98fc49",
24       "title": "fdvydfvdyv",
25       "topic": "Technikai",
26       "description": "dvsvyxcvxcvxcyy",
27       "user": {
28         "_id": "64444c823fbe171589d7802c",
29         "name": "Dörnyei Laura",
30         "email": "laladormnyei@gmail.com",
31         "role": "user",
32         "createdAt": "2023-04-22T21:07:14.781Z",
33         "__v": 0
34       },
35       "createdAt": "2023-04-24T14:47:35.451Z",
36       "__v": 0,
37       "id": "64469687abcb84814a98fc49"
38     },
39     {
40       "_id": "64469f93abcb84814a98fc62",
41       "title": "Koszos lemez",
42       "topic": "Technikai",
43       "description": "Hogyan tudom tisztítani",
44       "user": {
45         "_id": "64444c823fbe171589d7802c"
46       }
47     }
48   ]
49 }
```

Response text

Mint látható az adatbázisban szereplő minden a hét poszt megjelenik, amit a válaszüzenet ki ír: „count: 7”. Ha egy adott posztot szeretnénk lekérni, annak is megvan a módja, miszerint az indexe alapján keresünk rá. Ez a metódus populate függvényekkel lett kiegészítve, mellyel hozzá kiírtuk pluszba a hozzászólásokat, valamint, hogy kik írták azokat.

GET localhost:3000/api/postok/:postId

```

Response Headers 10 Cookies Results Docs
1 ~ {
2   "success": true,
3   "data": {
4     "_id": "643ac963da3a2745e9d6309f",
5     "title": "Elázott bakelitlejátszó",
6     "topic": "Zene",
7     "description": "Frissítve lett",
8     "user": {
9       "_id": "643ac513da3a2745e9d63083",
10      "name": "Francia Julietta",
11      "email": "juliettafrancia@gmail.com",
12      "role": "user",
13      "createdAt": "2023-04-15T15:38:59.274Z",
14      "__v": 0
15    },
16    "createdAt": "2023-04-15T15:57:23.640Z",
17    "__v": 0,
18    "hosszasolasok": [
19      {
20        "_id": "643acaa9da3a2745e9d630ad",
21        "description": "Még is van ötletem",
22        "user": {
23          "_id": "643ac513da3a2745e9d63083",
24          "name": "Francia Julietta",
25          "email": "juliettafrancia@gmail.com",
26          "role": "user",
27          "createdAt": "2023-04-15T15:38:59.274Z",
28          "__v": 0
29        },
30        "post": "643ac963da3a2745e9d6309f",
31        "createdDate": "2023-04-15T16:02:49.545Z",
32        "__v": 0
33      },
34    ]
}

```

Response text

Posztokat feltölteni is lehet természetesen, melyre a backend ki van alakítva a megfelelő metódussal, így a következő végpont használatával és a JSON Content kitöltésével meg is lehet oldani: `POST localhost:3000/api/postok/`

Body/JSON/Content:

```

JSON Content Format
1 {
2
3   "topic": "Keresek",
4   "title": "Szeretnék egy Azahriah albumot találni nagyon.",
5   "description": "Fizetőképes vevő lennék ,aki egy újonnan megjelent albumnak nagyon örülne!"
6 }

```

JSON Content

Válaszként a következőt kapjuk:

```
Response Headers10 Cookies Results Docs { } └
1   {
2     "success": true,
3     "data": {
4       "title": "Szeretnék egy Azahriah albumot találni nagyon.",
5       "topic": "Keresek",
6       "description": "Fizetőképes vevő lennék ,aki egy újonnan megjelent albumnak nagyon örülne!",
7       "user": "643f9ead24d433c900ff93a7",
8       "_id": "64495e3bd778778abd15768e",
9       "createdAt": "2023-04-26T17:24:11.173Z",
10      "__v": 0,
11      "id": "64495e3bd778778abd15768e"
12    }
13  }
```

Response text

Eredmény az adatbázisunkban látható:

```
_id: ObjectId('64495e3bd778778abd15768e')
title: "Szeretnék egy Azahriah albumot találni nagyon."
topic: "Keresek"
description: "Fizetőképes vevő lennék ,aki egy újonnan megjelent albumnak nagyon örü..."
user: ObjectId('643f9ead24d433c900ff93a7')
createdAt: 2023-04-26T17:24:11.173+00:00
__v: 0
```

Post módosítás a következőképpen lehetséges: `PUT localhost:3000/api/postok/:postId`

JSON Content:

JSON Content

```
1   {
2     "title": "Örülnek egy Azahriah lemeznek"
3   }
```

JSON Content

Válasz:

```
Response Headers10 Cookies Results Docs { } └
1   {
2     "success": true,
3     "data": {
4       "id": "64495e3bd778778abd15768e",
5       "title": "Örülnek egy Azahriah lemeznek",
6       "topic": "Keresek",
7       "description": "Fizetőképes vevő lennék ,aki egy újonnan megjelent albumnak nagyon örülne!",
8       "user": "643f9ead24d433c900ff93a7",
9       "createdAt": "2023-04-26T17:24:11.173Z",
10      "__v": 0,
11      "id": "64495e3bd778778abd15768e"
12    }
13  }
```

Response text

```
_id: ObjectId('64495e3bd778778abd15768e')
title: "Örülök egy Azahriah lemeznek"
topic: "Keresek"
description: "Fizetőképes vevő lennék ,aki egy újonnan megjelent albumnak nagyon örö..."
user: ObjectId('643f9ead24d43c900ff93a7')
createdAt: 2023-04-26T17:24:11.173+00:00
__v: 0
```

Adatbázisban megjelenő változás

És ha jelen példában, melyet most szemléltettem, már megtaláltuk a keresett lemezt, vagy éppen már nem tartunk rá igényt, akkor egyszerűen a poszt törlése is megoldható.

```
DELETE localhost:3000/api/postok/:postId
```

Status: 200 OK Size: 26 Bytes Time: 105 ms

Response	Headers 10	Cookies	Results	Docs
<pre>1 { 2 "success": true, 3 "data": {} 4 }</pre>				

Response text

Mint látható a metódus sikeresen végbement, így a poszt törlésre került.

5.3.4. Hozzájárások feltöltése, szerkesztése, törlése

A fent már megleírt posztok működésinek és funkcióinak bemutatásánál már volt arról szó, hogy lehet hozzájárásokat közzétevni. Ezeket a felhasználó igénye szerint módosíthatja, frissítheti és törölheti is. Feltöltés a következőképpen zajlik:

```
POST localhost:3000/api/postok/:postId/hozzaszolasok
```

JSON Content:

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

```
1 {"description": "Hát nem is tudom, szerintem töröld meg és megfelelő lesz."}
```

JSON Content

Adatbázisban a feltöltés után meg is jelenik az adatbázisban az imént írt hozzászólás:

```
_id: ObjectId('644962e0d778778abd15769a')
description: "Hát nem is tudom, szerintem töröld meg és megfelelő lesz."
user: ObjectId('643f9ead24d433c900ff93a7')
post: ObjectId('64469f93abcb84814a98fc62')
createdAt: 2023-04-26T17:44:00.984+00:00
__v: 0
```

Adatbázisban megjelenő eredmény

A hozzászólás szerkesztéséhez a következő API-s kérést kell beírni: **PUT**
localhost:3000/api/hozzaszolasok/:hozzaszolasId

JSON Content:

```
1 {"description": "Igazából csak vegyél egy újat ebben az esetben"}
```

JSON Content

Adatbázisban megjelenő változás:

```
_id: ObjectId('644962e0d778778abd15769a')
description: "Igazából csak vegyél egy újat ebben az esetben"
user: ObjectId('643f9ead24d433c900ff93a7')
post: ObjectId('64469f93abcb84814a98fc62')
createdAt: 2023-04-26T17:44:00.984+00:00
__v: 0
```

Adatbázisban megjelenő változás

A hozzászólás törléséhez szükséges végpont: **DELETE**
localhost:3000/api/hozzaszolasok/:hozzaszolasId

A kérés elküldésénél a következő válasz üzenetet kapjuk, mely igazolja, hogy az imént feltett hozzászólás törölve lett:

Response	Headers 10	Cookies	Results	Docs
1 ↴ { 2 "success": true, 3 "data": {} 4 }				

Response text

5.3.5. Kívánságlista lekérdezése, hozzáadás, törlés

A felhasználók kívánt termékeit hozzátudják adni saját kívánságlistájukhoz, melyet onnantól kezdve könnyedén számon tudnak tartani.

```
GET localhost:3000/api/wishlists
```

Válasz üzenet:

```
{
  "_id": "643fd35a6d941cfbef8b1828",
  "egysegár": 8000,
  "lemezallapot": "új",
  "boritoallapot": "új",
  "leiras": "elég cool sztem azért ilyen drága xsd",
  "lemezId": {
    "_id": "63bd38bf66b99d5fd58c1587",
    "lemezcím": "Plastic Hearts",
    "eloado": "Miley Cyrus",
    "evjarat": 2021,
    "raktaron": 1,
    "mufaj": [
      "Pop"
    ],
    "zeneszamok": [
      "WTF Do I Know",
      "Plastic Hearts",
      "Angels Like You",
      "Prisoner",
      "Gimme What I Want",
      "Night Crawling",
      "Midnight Sky",
      "High",
      "Hate Me",
      "Bad Karma",
      "Never Be Me",
      "Golden G String",
      "Edge Of Midnight(Midnight Sky Remix)",
      "Heart Of Glass (Live From The iHeart Festival)",
      "Zombie (Live From The NIVA Save Our Stages Festival)"
    ],
    "user": "6424503e8f138279fd1eb85c",
    "photo": "photo_63bd38bf66b99d5fd58c1587.jpg",
    "id": "63bd38bf66b99d5fd58c1587"
  },
  "user": "643f9ead24d433c900ff93a7",
  "createdAt": "2023-04-19T11:41:14.378Z",
  "__v": 0
}
```

Response text

Hozzáadás a Kívánságlistához: `POST localhost:3000/api/wishlists`. JSON Content-be a következő elemek megadásával vihető fel egy termék:

```
{
  "termek": "643fd35a6d941cfbef8b1828"
}
```

JSON Content

Ugyanezen index megadásával törölhető a termék a Kívánságlistákból:

```
DELETE localhost:3000/api/wishlists/:wishlistId
```

```
{
  "message": "Item deleted from wishlist"
}
```

Response text

5.3.6. Vélemények lekérése, hozzáadása, módosítása és törlése

Vélemények hozzáadásához is van lehetőség az oldalon. Ezt a felhasználók egymásról tudják alkotni 1-5-ös skálán. Egy adott vélemény lekéréséhez szükség van a vélemény indexére és a következő végpont elküldésével meg is kapjuk válaszként: `GET localhost:3000/api/ratings/:ratingId`

Válasz üzenet:

```
{
  "success": true,
  "data": [
    {
      "_id": "643fd1e56d941cfbef8b181c",
      "user": null,
      "ratedBy": {
        "_id": "643f9ead24d433c900ff93a7",
        "name": "Szombathelyi Levente",
        "email": "szombathelyi.levi64@gmail.com",
        "role": "admin",
        "createdAt": "2023-04-19T07:56:29.740Z",
        "__v": 0,
        "resetPasswordExpire": "2023-04-22T17:18:41.501Z",
        "resetPasswordToken": "1133133adadb2c622318b1db51d2ea323bbfaba6d6a87b7fad367a2b80658654"
      },
      "rating": 1,
      "comment": "szar.",
      "createdAt": "2023-04-19T11:35:01.443Z",
      "updatedAt": "2023-04-19T11:35:01.443Z",
      "__v": 0
    }
  ]
}
```

Response text

Mint látható magán a hozzászóláson és az értékelésen felül a lekérdezés kiírja, hogy a vélemény kiről is lett alkotva, valamint, hogy ki adta hozzá azt. Úgy, mint az eddigiekben is véleményt lehet hozzáadni is. `POST localhost:3000/api/ratings/:userId`

JSON Content:

```
JSON Content
1  {
2    "rating": 5,
3    "comment": "Great service! Highly recommended."
4  }
5
```

Válasz üzenet sikeres küldéskor:

```
{  
  "success": true,  
  "msg": "Értékelés sikeresen hozzáadva"  
}
```

Vélemény frissítéséhez a következő végponton kell a változtatásokat megejteni: **PUT**
localhost:3000/api/ratings/:ratingId

JSON Content

```
1  {  
2    "rating": 1,  
3    "comment": "Poor service!"  
4  }  
5
```

Válasz üzenet sikeres küldés esetén:

```
1  {  
2    "success": true,  
3    "msg": "Értékelés sikeresen frissítve",  
4    "data": {  
5      "_id": "644978ce7eae9882d8ce3670",  
6      "user": "644250e9235582947adb38b8",  
7      "ratedBy": "643f9ead24d433c900ff93a7",  
8      "rating": 1,  
9      "comment": "Poor service!",  
10     "createdAt": "2023-04-26T19:17:34.749Z",  
11     "updatedAt": "2023-04-26T19:21:18.613Z",  
12     "__v": 0  
13   }  
14 }
```

Eredmény az adatbázisban is látható:

```
_id: ObjectId('644978ce7eae9882d8ce3670')  
user: ObjectId('644250e9235582947adb38b8')  
ratedBy: ObjectId('643f9ead24d433c900ff93a7')  
rating: 1  
comment: "Poor service!"  
createdAt: 2023-04-26T19:17:34.749+00:00  
updatedAt: 2023-04-26T19:21:18.613+00:00  
__v: 0
```

A törlés ezen a végponton megvalósítható: `DELETE localhost:3000/api/ratings/:ratingId`

Válasz:

```
{
  "success": true,
  "msg": "Értékelés sikeresen törölve",
  "data": {
    "_id": "644978ce7ae9882d8ce3670",
    "user": "644250e9235582947adb38b8",
    "ratedBy": "643f9ead24d433c900ff93a7",
    "rating": 1,
    "comment": "Poor service!",
    "createdAt": "2023-04-26T19:17:34.749Z",
    "updatedAt": "2023-04-26T19:21:18.613Z",
    "__v": 0
  }
}
```

Response text

5.3.7. Szűrések

Az weboldalunkon a felhasználók használhatnak különböző szűrésmódokat, amivel leegyszerűsíthetik a keresést az oldalunkon. Tudnak böngészni az eladó lemezek között megjelenés, lemezcím, műfaj és előadó alapján is. A fórumon, pedig a téma szerint tudják a felhasználók a szűrést elvégezni. Természetesen ehhez ki kellett egészítenünk a meglévő metódusunkat „if” elágazásokkal.

```
try {
  const { eloado } = req.query;
  const { evjarat } = req.query;
  const { mufaj } = req.query;
  const { lemezcim } = req.query
  let query;

  if (eloado) {
    query = Lemez.find({ eloado: eloado }).populate({
      path: 'termek',
      populate: {
        path: 'user',
        select: 'name email',
      }
    });
  }
  else if (evjarat) {
    query = Lemez.find({ evjarat: evjarat }).populate({
      path: 'termek',
      populate: {
        path: 'user',
        select: 'name email',
      }
    });
  }
  else if (mufaj) {
    query = Lemez.find({ mufaj: mufaj }).populate({
      path: 'termek',
      populate: {
        path: 'user',
        select: 'name email',
      }
    });
  }
}
```

If elágazások

Megjelenés alapján a szűrést következő végpont használatával lehet: **GET**

```
localhost:3000/api/lemezek?evjarat=""
```

Válasz üzenetben a 2020-as megjelenésű eladó lemezeket dobja ki:

```
{
  "success": true,
  "count": 3,
  "data": [
    {
      "_id": "63bd3f2e66b99d5fd58c158b",
      "lemezcim": "The New Abnormal",
      "eloado": "The Strokes",
      "evjarat": 2020,
      "raktaron": 11,
      "mufaj": [
        "Rock"
      ],
      "zeneszamok": [
        "The Adults Are Talking",
        "Selfless",
        "Brooklyn Bridge To Chorus",
        "Bad Decisions",
        "Eternal Summer",
        "At The Door",
        "Why Are Sundays So Depressing",
        "Not The Same Anymore",
        "Ode To The Mets"
      ],
      "user": "6424503e8f138279fd1eb85c",
      "photo": "photo_63bd3f2e66b99d5fd58c158b.jpg",
      "termeket": [],
      "id": "63bd3f2e66b99d5fd58c158b"
    }
  ]
}
```

Response text

Műfaj alapján való szűrés: **GET localhost:3000/api/lemezek?mufaj=""**

Válasz üzenet:

```
{
  "success": true,
  "count": 3,
  "data": [
    {
      "_id": "63bd38bf66b99d5fd58c1583",
      "lemezcim": "Ószönlény",
      "eloado": "Krábí",
      "evjarat": 2020,
      "raktaron": 10,
      "mufaj": [
        "Hip Hop"
      ],
      "zeneszamok": [
        "Jéghideg",
        "Lejtő",
        "Felejtő",
        "Csekk",
        "Dimamit",
        "Petőfi",
        "Puszl",
        "Sapiens",
        "Kutya",
        "Copofcska"
      ],
      "user": "6424503e8f138279fd1eb85c",
      "photo": "photo_63bd38bf66b99d5fd58c1583.jpg",
      "termeket": [
        {
          "_id": "6442d22b8a0956d10679395f",
          "egysgar": 2000,
          "lemezallapot": "használt",
          "boritoallapot": "használt"
        }
      ]
    }
  ]
}
```

A válaszban a Hip-Hop műfajban felrakott lemezeket listázta ki.

Response text

Előadó alapján való szűréshez a következő végponton lehet a lekérdezést elküldeni: GET localhost:3000/api/lemezek?eloado=""

Válasz üzenet:

```
{
  "success": true,
  "count": 1,
  "data": [
    {
      "_id": "63bd38bf66b99d5fd58c1584",
      "lemezcim": "Sour",
      "eloado": "Olivia Rodrigo",
      "evjarat": 2021,
      "rakarton": "2",
      "mufaj": [
        "Pop"
      ],
      "zeneszamok": [
        "Brutal",
        "Traitor",
        "Drivers License",
        "1 Step Forward, 3 Steps Back",
        "Deja Vu",
        "Good 4 U",
        "Enough For You",
        "Happier",
        "Jealousy, Jealousy",
        "Favorite Crime",
        "Hope Ur Ok"
      ],
      "user": "6424503e8f138279fd1eb85c",
      "photo": "photo_63bd38bf66b99d5fd58c1584.jpg",
      "termekek": [
        {
          "_id": "6442d25b8a0956d106793966",
          "egyezaglapot": "viseltes",
          "boritoallapot": "hasznalt",
          "leiras": "Eladásra kínált termék.",
          "lemezid": "63bd38bf66b99d5fd58c1584",
          "user": {
            "_id": "643ac513da3a2745e9d63083",
            "name": "Francia Julietta",
            "email": "juliettafrancia@gmail.com"
          },
          "createdAt": "2023-04-21T18:13:47.884Z",
          "__v": 0
        }
      ]
    }
  ]
}
```

A lekérdezésben az előadónak Olivia Rodrigo-t adtam meg és sikeresen ki is listázta azt az egyet, ami fenn van az oldalon.

És az utolsó szűrésfajta, ami megtalálható oldalunkon az a fórumon helyezkedik el, ahol a téma között lehet válogatni: GET localhost:3000/api/postok?topic=""

Válasz sikeres küldés esetén:

```
{
  "success": true,
  "count": 1,
  "data": [
    {
      "_id": "6446c22babcb84814a98fd5a",
      "title": "gzgjvgjvhgvhgv",
      "topic": "Zene",
      "description": "ftchtgcch",
      "user": {
        "_id": "64444c823fbe171589d7802c",
        "name": "Dörnyei Laura",
        "email": "laladornyei@gmail.com",
        "role": "user",
        "createdAt": "2023-04-22T21:07:14.781Z",
        "__v": 0
      },
      "createdAt": "2023-04-24T17:53:47.084Z",
      "__v": 0,
      "id": "6446c22babcb84814a98fd5a"
    }
  ]
}
```

Response text

A „Zene” témaban létrehozott bejegyzéseket listázta ki a fórumról a kérés.

5.3.8. Engedélyek, hibaüzenetek

Természetesen kialakításra került minden módosításhoz és törléshez, olyan funkció, ahol csak az adott terméket/posztot/hozzászólást/véleményt író vagy közzé tevő felhasználó tudja azokat elvégezni, másképpen a válasz egy hibaüzenetet ír ki. Az API hibák esetén JSON formátumban adja vissza az alábbi válaszokat:

{ "success": false }	Általános hibaüzenet, amely akkor jelenik meg, ha valami nem stimmel az adatbázissal vagy a szerverrel.
{ "success": false, "msg": "Not found" }	Amikor a kérés nem találja az adott erőforrást.
{ "success": false, "msg": "Nincs engedélyed ehhez az útvonalhoz!" }	Amikor az adott művelethez nincs megfelelő jogosultság.

Fontos megjegyezni, hogy az egyes végpontok különböző hibaüzeneteket is visszaadhatnak a specifikus hibák jelzésére. Ezek az üzenetek az adott végpont dokumentációjában találhatók.

6. A frontend fejlesztése

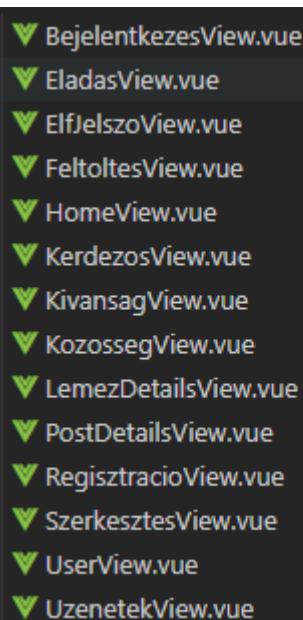
6.1. A fejlesztői környezet és kialakítás

A frontendünk Vue 3 keretrendszerrel készült. A Vue egy progresszív, inkrementális JavaScript keretrendszer, amely lehetővé teszi az egyszerű és hatékony felhasználói felületek készítését. A Vue 3 az előző verzióhoz képest számos új funkcióval és fejlesztéssel rendelkezik, többek között az új telepítési lehetőségekkel, a fejlettebb reaktivitással és a komponens szintű dinamikus importálással. A Vue 3 továbbá optimalizálva van a teljesítményre és a memória használatra, ami javítja a felhasználói élményt és a fejlesztői produktivitást. Mi ezzel a keretrendszerrel szintén az idén találkoztunk először, de egyből elnyerte tetszésünket, ezért is választottuk ezt a záródolgozat frontend részéhez.

Router: Az alkalmazás elérési útvonalait kezeli. A router/index.js fájlban található, és definiálja az összes útvonalat, amelyet az alkalmazás használ. Fontos, hogy az útvonalak és a hozzájuk kapcsolódó komponensek összhangban legyenek egymással:

```
import { createRouter, createWebHistory } from "vue-router";
import HomeView from '../views/HomeView.vue';
import EladasView from '../views/EladasView.vue';
import KozossegView from '../views/KozossegView.vue';
import FeltoltesView from '../views/FeltoltesView.vue';
import BejelentkezesView from '../views/BejelentkezesView.vue';
import SzerkeszesView from '../views/SzerkeszesView.vue';
import KivansagView from '../views/KivansagView.vue';
import UzenetekView from '../views/UzenetekView.vue';
import ElfJelszoView from '../views/ElfJelszoView.vue';
import LemezDetailsView from '../views/LemezDetailsView.vue';
import RegisztracioView from '../views/RegisztracioView.vue';
import UserView from '../views/UserView.vue';
import KerdezView from '../views/KerdezosView.vue';
import PostDetailsView from '../views/PostDetailsView.vue';
```

Routes/index.js view importálások



View fájlaink

Views: Views mappa az összes oldalt tartalmazza, amelyek megjelennek az alkalmazásban. Ezek az oldalak a routeron keresztül érhetők el, és általában az alkalmazás állapotát:

Stores: Az alkalmazás állapotát kezeli, és az összes adatot tárolja, amelyet az alkalmazás használ.

Main.js: Az alkalmazás fő belépési pontja, amely a Vue alkalmazást inicializálja. Az index.html fájlban a main.js fájl hivatkozása található meg. A main.js fájl a Vue alkalmazás fő beállításait végzi, például a Vue instance létrehozását, a router és store beállításait, az alkalmazás alapértelmezett stílusainak beállítását, valamint a globális komponensek és pluginok beállításait is végzi.

```
import { createApp } from 'vue'
import App from './App.vue'
import {createPinia} from 'pinia';
import router from './router';
import VueCookies from 'vue-cookies';

import 'bootstrap/dist/css/bootstrap.css';
import 'bootstrap/dist/js/bootstrap.bundle.js';
import './assets/css/style.css';

const app = createApp(App);

app.use(createPinia());
app.use(router);
app.use(VueCookies);
VueCookies.config('7d');

export const piniaStore = createPinia();

app.mount('#app');
```

Main.js tartalma

Components: Components mappában találhatók az alkalmazás felhasználói felületének különböző komponensei. Ezek a komponensek újra felhasználható elemek, amelyeket külön-külön lehet használni, és össze lehet állítani belőlük az alkalmazás felhasználói felületének különböző részeit:

Header.vue	A webalkalmazás fejlécének komponense.
Footer.vue	A webalkalmazás láblécének komponense.
AppNav.vue	A webalkalmazás navigációs sávjának komponense.
FeltoltesForm.vue	A webalkalmazás Feltöltés űrlapjának komponense.
PostCard.vue	Posztok kártyájának komponense.
PostForm.vue	Posztok űrlapjának komponense.
TermekCard.vue	Termékkártyák komponense.
KepFeltoltesForm.vue	Termékek képfeltöltésének komponense.

A weboldal kinézetének személyre szabásához a style.css módosítását használtuk, mellyel többek között a footer-ünk lett a legnagyobb mértékben módosítva. A footer alján megtalálható egy social icon, melynek megjelenítéséhez a következő parancsot írtuk be a terminal-ba: npm install vue-socials. A megfelelő icon kiválasztása után, már csak a megfelelő helyre kellett beilleszteni annak egyedi svg kódját.

6.2. A fejlesztés menetrendje, első mérföldkő

A fejlesztés első lépéseként megterveztük magának az oldalunknak a kinézetét nagyvonalakban, hogy mik és hol jelenjenek meg. Miután összeállt a kép a fejünkben elkezdtük a valódi programozást, és lefektettük az alapokat. App Vue-ba beilleszthető komponensek következtek: header, footer és navbar. Törekedtünk a letisztult megjelenésre, ezért a header tartalma mindenkor egy kép, melyet magunk szerkesztünk, hogy megfelelő legyen és illeszkedjen a weboldalunk témájához. Az eredeti képre még rá lett szerkesztve az oldalunk logója. A footer szintén az egyszerű megjelenést képviseli az oldalunkon, itt helyezkedik a social icon, amit már említettem feljebb. Navigációs sáv létrejöttén is sokat agyaltunk, hogy hogyan is működjön és milyen legördülő lista jelenjen meg, de az alapok után elkezdtünk a tervezünk szerint sorrendbe haladni és megvalósítani azokat. Ennek következtében létrehoztuk a kiterelt oldalak kinézetét tartalmazó vue fájlokat a views mappán belül. minden egyes ilyen kész oldalnak kialakítottuk a megfelelő útvonal választást a router/index.js mappán belül. Mikor az oldal kinézete készen volt következhetett a funkciók beépítése az oldalunkba, melyre a backend már készen állt és a

frontendet kellett ráhangolni, valamint összekötni a kettőt, hogy kommunikáljanak egymással és működjenek a lekérdezések, mely célunk eléréshez Axios-t használtunk. Az Axios egy HTTP kliens könyvtár, amely lehetővé teszi a böngésző és a Node.js közötti kommunikációt. Ennek segítségével képesek vagyunk HTTP kéréseket küldeni aszinkron módon a backend felé, majd a válaszokat feldolgozzuk. Az Axios-t választottuk, mert nagyon könnyű használni, nagyon jól dokumentált, és széles körben használják a fejlesztők. Támogatja az aszinkron kéréseket és lehetőséget biztosít az egyszerű hibakezelésre.

Továbbá a Vue.js keretrendszer támogatja az Axios használatát, így könnyen integrálható a Vue alkalmazásokkal. A services/dataservices.js fájlba bekerült a szükséges kód, mely tartalmazza az Axios importálást, valamint a szükséges bázisURL-t a kommunikáció megvalósításához.

```
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/",
      name: "home",
      component: HomeView,
    },
    {
      path: "/eladas",
      component: EladasView,
    },
    {
      path: "/kozosseg",
      component: KozossegView,
    },
    {
      path: "/feltoltes",
      component: FeltoltesView,
    },
    {
      path: "/bejelentkezes",
      component: BejelentkezesView,
    },
    {
      path: "/szerkeszes",
      component: SzerkeszesView,
    },
    {
      path: "/kivansag",
      component: KivansagView,
    },
  ],
})
```

Routes/index.js útvonal választás

```
import Axios from 'axios';
let token = sessionStorage.getItem("token");

const instance = Axios.create({
  baseURL : 'http://localhost:3000/api',
  headers:{ 
    'Content-type': 'application/json',
    'teszt':'teszt',
    'Authorization': 'Bearer ' + token
  }
});

export default instance;
```

Axios és URL importálása

A stores/index.js fájlba bekerültek a metódusok, melyekre szükség van az oldal megfelelő működéséhez, majd ezek a metódusok megfelelő importálásával és feltételek megadásával sikerült elérni a kívánt funkciókat az oldalon. Első mérföldkőnek azt mondánám, mikor a bejelentkezés teljes mértékben működött és ennek ellenőrzése is sikeres volt. Nagyon sokat bajlódtunk vele, ugyanis hibából hibába ütközünk és bejelentkezés nélkül a legtöbb funkciónk nem működik.

6.3. A fejlesztés fontosabb megoldandó problémái

6.3.1. Bejelentkezés fixálása

A bejelentkezéshez szükséges token el volt tárolva cookie-ba, de nem gondoltunk sessionStorage használatára először, ezáltal nem tudtuk ellenőrizni, hogy biztosan be van- e lépve a felhasználó. A sessionStorage használata megoldást nyújtott erre problémára.

6.3.2. Eladó lemezek megjelenítése

A probléma nem volt más, minthogy amikor az eladó lemezeinket akartuk kilistázni, azokat a lemezeket is kiírtattuk, melyekhez nem volt feltöltve termék, ezért üres div-ek formájában jelentek meg az oldalon, ezzel hézagos lett az elrendezés, mely esztétikailag nagyon nem volt megfelelő. Erre a problémára végül megoldást jelentett, hogy nem a kártya megjelenítésénél szűrtünk, hiszen így hiába nem jelent meg a kártya, a kártya helye le lett generálva. Ezért stores-ban létrehoztunk egy getEladoLemezek() függvényt, amely már alapból kiválogatva adja át az adatokat az oldalnak.

```
getEladoLemezek() {
  return Axios.get('/lemezek')
    .then(resp => {
      //this.lemezek = resp.data.data;
      this.lemezek = []
      resp.data.data.forEach(item => {
        if (item.termek != '') { this.lemezek.push(item) }
      })
    })
    .catch(err => {
      return Promise.reject(err);
    })
}
```

stores/index.js-ben a problémát megoldó függvény

6.4. Telefonos nézet, reszponzivitás

Az oldalunkon való megfelelő mobilos megjelenítést és reszponzivitást a Bootstrap keretrendszer segítségével értük el. A Bootstrap alapvetően egy reszponzív, mobilbarát CSS keretrendszer, amely lehetővé teszi, hogy egy weboldalt könnyedén alkalmazkodtassunk különböző méretű képernyőkhöz, így a felhasználók bármilyen eszközön kényelmesen és hatékonyan tudják használni az oldalt. A Bootstrap class-okat használva a különböző elemek, min címsorok, paragrafusok és gombok dinamikusan változnak a különböző képernyőméretekhez és az azokon megjelenő elemekhez igazodva.

7. Tesztelés

7.1. Tesztelés formája

A tesztelés formája a kézi tesztelés: A kézi tesztelés során manuálisan ellenőriztük az webalkalmazásunk egyes funkcióit. Ezek a tesztek elsősorban a felhasználói élményre, a megjelenésre, valamint az alkalmazás általános működésére fókuszáltak.

7.2. Funkciók működésének tesztje

Ezt úgy dokumentáltuk, hogy a weboldalon megtalálható funkciókat elvégeztük, majd, ha az ott megfelelően megjelent, valamint a backendünk terminálja 200-as státusz kódot, vagy a kereső fejlesztőeszközén belül a konzolban success üzenetet kaptunk, azaz sikeres küldést jelzett, akkor a funkciót működőnek dokumentáltuk.

7.2.1. Példa tesztelés: Belépés/kilépés és regisztráció

A megfelelő email és jelszó használata után a

success: true, jelenik meg a böngésző

fejlesztőeszközének konzolában.

Regisztráció esetén a következő történik. Az imént regisztrált személy meg is jelenik az adatbázisba.

```
index.js:262
  {success: true, token: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
  > C9.eyJpZCI6IjY0M...jU1fQ.ED3eja7Kwd6PHTQwLbzEhyvW0ssBIrwuc
  gXytxMH9k'}
```

Konzol üzenet belépésnél

Fejlécek	Előnézet	Válasz	Kezdeményező	Időzítés	Cookie-k
		1 {"success":true,"msg":"Felhasználó kijelentkezett."}			

Kijelentkezés üzenet

Regisztráció

Név:	Példa Tamás
Email:	peldafelhasznalo@students.jedlik.eu
Jelszó:	*****
<input type="button" value="Regisztráció"/>	

Regisztrációs adatok

Fejlécek	Adatok	Előnézet	Válasz	Kezdeményező	Időzítés	Cookie-k
			1 {"success":true,"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6...			

Konzol/Hálózat/register üzenet regisztrálásnál

```
_id: ObjectId('644aaadd3c1bf37f59aa51db')
name: "Példa Tamás"
email: "peldafelhasznalo@students.jedlik.eu"
role: "user"
password: "$2a$10$0Y3lXx9/TGup86l.tohPr.iuL5BWovLHofv2Ypjzzf.sLiHVc36tC"
createdAt: 2023-04-27T17:03:25.259+00:00
__v: 0
```

Adatbázisban megjelenő regisztráció

8. Felhasználói útmutató

Lemezbázis weboldalunk egy olyan online adás-vétel felület, amely elsősorban lemezgyűjtőknek készült. A weboldal látogatható asztali számítógépről, laptopról, tabletről és telefonról is a reszponzivitása miatt. A következő alcímek alatt elolvasható, hogy mégis hogyan lehet használni weboldalunk.

8.1. Regisztráció és Belépés

Az oldalunk ugyan látogatható anélkül is, hogy legyen profilja a felhasználónak, de így az elérhető funkciók korlátozottak és az eladó termékeket vagy fórumot csak megtekinthetik az odatévedők, de terméket feltölteni vagy posztot kirakni esetleg hozzájárulást írni csak egy regisztrált felhasználónak tud. Ezért érdemes az oldal használónak regisztrálni, majd belépní. A regisztrációnál meg kell adni a felhasználónak saját teljes nevét, egy létező email címet, valamint egy minimum hat karakterből álló jelszót

8.2. Főmenü

A sikeres regisztrációt követően a bevitt adatokkal be is lehet jelentkezni. A sikeres belépést követően a weboldal a főmenüre irányítja a felhasználót és megjelenik a navigációs sávon a profil fül, ahol különböző funkciók közül lehet választani. A főmenün olvasható egy pár bekezdés oldalunkról, valamint megjelenik az éppen eladó termékek egy része.

Alatta látható egy gomb, melyre kattintva átirányít az eladó lemezek oldalra.



8.3. Eladó lemezek

Megjelenik az összes jelenleg eladó lemez a weboldalunkon, melyet felhasználók töltöttek fel. Ezen az oldalon felül található egy szűrő, ahol tudnak böngészni az eladó lemezek között megjelenés, lemezcím, műfaj és előadó alapján is. Ha kiválasztanak egy adott lemezt, mely felkeltette érdeklődésüket az oldal átirányítja, ahol már csak az adott termék jelenik meg a hozzá feltöltött információkkal. Odacsatolva megjelenik még az eladó is a lemeznek állapotával, borítójának állapotával és árával, amire kattintva az oldal tovább irányít az eladó saját fiókjára, ahol fel tudjuk venni vele a kapcsolatot és elkezdődhet az üzletkötés.

8.4. Vélemények

Az eladóra rákattintva az „Eladó lemez” oldalról átirányít az felhasználó profiljára. Itt tudjuk őt értékelni 1-5-ös skálán értékelhetőek.

8.5. Fórum

A Fórumra kattintva a navigációs sávon, vagy a footeren átkerülünk a fórum oldalra. Itt a felhasználók szűrőnk segítségével böngészhetnek a különböző téma között, melyet az oldal kilistáz. Az itt megjelenő posztokat más felhasználók töltötték fel, rájuk kattintva megnyílik a poszt, és az, hogy milyen témaban íródott az, valamint mire keres megoldást. Az oldal felhasználói képesek a poszt alá kommentelni is. Ha esetleg ő maga szeretne posztot létrehozni akkor a „kérdés feltétele” gombra kattintva megteheti azt.

8.6. Profil fül

8.6.1. Feltöltés

Ez a profil fül első eleme a listában, erre kattintva elirányít az feltöltés oldalra, ahol terméket tudunk feltölteni, melyet eladásra szánunk. Először egy szűrést kell elvégeznünk, mely az adatbázisunkba szereplő összes lemezet átnézi, és ha egyezést talál kidobja azt. Ekkora kiválasztjuk a lemezt és csak a termék adatait kell kitöltenünk (Ár, lemezállapot, borítólállapot, leírás). Ha nem szerepel adatbázisunkban a „Lemez feltöltése” gombra kattintva ezeken felül még ki kell tölteni a következőket: Lemezcím, előadó, megjelenés, műfaj és zeneszámok. „Adatok feltöltése” gombra kattintva véglegesítjük a feltöltést és felkerül az az eladó lemezekhez.

8.6.2. Kívánságlista

Az oldalon a jövőben lesz egy ilyen funkció, melyet a „Tervek” címsor alatt kifejtettünk

8.6.3. Üzenetek

Az oldalon a jövőben lesz egy ilyen funkció, melyet a „Tervek” címsor alatt kifejtettünk

8.6.4. Kijelentkezés

Ha a felhasználó rákattint a kijelentkezés gombra, akkor a webalkalmazás törli az összes aktuális felhasználói adatot a böngészőben tárolt munkamenetből, a profil fül visszavált Regisztráció és Belépés opcióra.

9. Tervek

Későbbiekben tervezünk az oldalra egy „Üzenetek” funkciót is, aminek a helye már kialakításra került a Profil fül legördülő listánkba, melynek a backendje már kész, de a frontenden még fejlesztésekre van szüksége, ahol a felhasználók, majd tudnak egymásnak direktüzeneteket küldeni, ezzel egyszerűsítve az üzletelés folyamatát a weboldalon. Kívánságlista funkció is megtalálható lesz az oldalon a jövőben, melynek helye szintén megtalálható és a backendben és már működik, ahol az egyes termékeket a felhasználó egyszerűen hozzáadhatja a saját Kívánságlistájához, melyen számon tudja majd tartani az áhitott termékeket.

10. Összegzés

Visszatekintve a projektre, sok kihívással kellett szembenéznünk, melyek megoldása sokszor majdnem egy teljes délutánt vett igénybe. Ilyen élményben még egyikünknek sem volt része. Nyilván nagy volt rajtunk a felelősség saját munkánk iránt ezáltal a teher is, ugyanis ekkora feladatot még nem kaptunk a középiskolás éveink alatt, de örömmel tekintünk vissza a folyamatra, ugyanis végül sikerült megoldani a hibákat és folytatni a munkát, melyből úgy gondoljuk a tőlünk telhető legjobbat sikerült kihozni. Az elképzelt és magunk elé kitűzött feladatok nagy része sikeresen megvalósult, és azok, amelyek még fejlesztés alatt állnak, remélhetőleg hamarosan befejeződnek, ezzel tovább javítva a felhasználói élményt és funkcionálitást.

Az egész projekt izgalmas és érdekes volt számunkra, hiszen sok tapasztalatot szereztünk a fejlesztési folyamatokkal kapcsolatban. A fejlesztői csapat kiválóan működött együtt, és a csapatmunka megvalósítása is nagyon hatékony volt.

Ez a projekt új kihívásokat jelentett számunkra, és fontos tapasztalatokkal gazdagodtunk, amelyek hasznosak lesznek a jövőbeni munkahelyi vagy akár egyetemi projektek során is. Összességében nagyon elégedettek vagyunk a végeredménnyel, és büszkék vagyunk arra, amit elértünk.

11. Források

Bootstrap osztályok, gombok, űrlap, ikonok:

<https://getbootstrap.com/>

<https://www.npmjs.com/package/vue-socials?activeTab=readme>

<https://mdbbootstrap.com/docs/standard/extended/social-media/>

Logó és képek szerkesztése:

<https://imageresizer.com/>

<https://www.remove.bg/>

<https://pixlr.com/x/>

<https://logo.com/>

Axios:

<https://axios-http.com/docs/intro>

Node .js:

Soós Gábor osztályfőnökünk óráin tanult kódok és órai munkák

<https://nodejs.org/en/about>

Vue3:

Bólya Gábor Tanár Úr óráin tanult kódok és órai munkák

Footer:

<https://codepen.io/scancode/pen/MEZPNd>

MongoDB:

<https://www.mongodb.com/docs/atlas/>

<https://www.datensen.com/data-modeling/moon-modeler-for-databases.html>

Bizonyos hibák megoldására:

<https://stackoverflow.com/>

Tartalom

1.	Bevezetés.....	1
1.1.	Projektünk ötlete.....	1
1.2.	Szoftver célja	2
1.3.	Funkciókról bővebben	2
2.	A fejlesztői csapat	4
2.1.	A csapat tagjai és feladataik.....	4
2.2.	A csapatmunka megvalósítása	4
3.	Adatbázis	5
3.1.	Az adatbázis háttér technológiája, adatbáziskezelő program.....	5
3.2.	Az adatbázis leírása, magyarázata	6
3.3.	Adatbázis diagramja	7
4.	A backend fejlesztése.....	8
4.1.	A fejlesztői környezet, adatbázis kommunikáció.....	8
4.2.	A fejlesztés menetrendje, mérföldkövek	9
4.3.	Hibára példa és annak kiküszöbölése, tesztelés	13
5.	REST API.....	14
5.1.	Bevezető.....	14
5.2.	Azonosítás és hitelesítés	14
5.3.	Tesztelések	15
5.3.1.	Felhasználó regisztrálása, bejelentkezése	15
5.3.2.	Termékek lekérdezése, feltöltése, módosítása, törlése.....	17
5.3.3.	Posztok lekérdezése, létrehozása, frissítése, törlése	21
5.3.4.	Hozzászólások feltöltése, szerkesztése, törlése	25
5.3.5.	Kívánságlista lekérdezése, hozzáadás, törlés	27
5.3.6.	Vélemények lekérése, hozzáadása, módosítása és törlése	28
5.3.7.	Szűrések	30
5.3.8.	Engedélyek, hibaüzenetek	33
6.	A frontend fejlesztése	34
6.1.	A fejlesztői környezet és kialakítás	34
6.2.	A fejlesztés menetrendje, első mérföldkő	36

6.3.	A fejlesztés fontosabb megoldandó problémái	37
6.3.1.	Bejelentkezés fixálása	37
6.3.2.	Eladó lemezek megjelenítése.....	37
6.4.	Telefonos nézet, reszponzivitás	37
7.	Tesztelés	38
7.1.	Tesztelés formája.....	38
7.2.	Funkciók működésének tesztje	38
7.2.1.	Példa tesztelés: Belépés/kilépés és regisztráció	38
8.	Felhasználói útmutató.....	39
8.1.	Regisztráció és Belépés.....	39
8.2.	Főmenü	39
8.3.	Eladó lemezek	39
8.4.	Vélemények.....	39
8.5.	Fórum.....	40
8.6.	Profil fül.....	40
8.6.1.	Feltöltés	40
8.6.2.	Kívánságlista.....	40
8.6.3.	Üzenetek	40
8.6.4.	Kijelentkezés.....	40
9.	Tervek.....	41
10.	Összegzés.....	41