# Bevezetés

## 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 mindig tudták teljesen kiszolgálni az igényeiket, vagy nehézségekbe ütköztek a termék beszerzésével külföldről. Ennek ellenére azonban rengeteg lemezgyűjtő van hazánkban, akik szenvedéllyel gyűjtik 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 hoz-zunk létre, ahol a zene iránt érdeklődők találkozhatnak egymással, megoszthatják 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 kezelni az ékezetes 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 kipróbáltuk a zöld különböző árnyalatait és kiválasztottuk a legígéretesebbet. Végső állomásként testreszabtuk a logó hátterét és már kész is volt a szemetgyönyörködtető, ám de nem túl hivalkodó logónk:

A képen szöveg, embléma látható

Automatikusan generált leírás

## Szoftver célja

Projektünk az idei tanévben indult el, és célunk egy olyan online adás-vétel felület lét-rehozása 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 témákban kérdezzenek, vitassanak, valamint megoldást keressenek. 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 témákban. Fórumunk lényeges szerepet tölt be oldalunkon, mivel lehetőséget biztosít kapcsolatteremtésre egymással.

## 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 elkezdődhet 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.

Sikeres bejelentkezést követően a navigációs sáv jobb felső sarkában található „Regiszt-ráció”, 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 szolgaló felületre, ahol a megjelenő mezőket kitöltve és az utolsó gombra kattintva felkerülhet az oldalra az eladásra kínált bakelit. „Termékek szerkesztése” fülre klikkelve, ha van a felhasználónak éppen eladó terméke, akkor azt tudja itt módosítani, vagy esetlegesen törölni. Későbbiekben tervezünk az oldalra egy „Üzenetek” funkciót is, melynek a backendje nagyja már kész, de a frontenden még fejlesztésekre van szüksége, ahol a felhasználók, majd tudnak egymásnak direktüzeneteket küldeni, ezzel egyszerűsítve az üzletelés folyamatát az weboldalon. Kívánságlista funkció is megtalálható lesz az oldalon a jövőben, ahol, az egyes termékeket a felhasználó egyszerűen hozzáadhatja a saját Kívánságlistájához, melyen számon tudja majd tartani az áhított termékeket. Végül az utolsó elem a legördülő listán a „Kijelentkezés” gomb, melyre kattintva az oldal egyszerűen kiléptet a bejelentkezett fiókból és a „Profil” fül visszavált „Regisztráció” „Bejelentkezés” fülre a jobb felső sarokban.

# A fejlesztői csapat

## 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.

## 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 alkamazásba, melyet többiek könnyedén saját munkaeszközükre le tudták „pull”-olni és mindig a legfrissebb, 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 tervek, esetlegesen, ha valaki elakadt próbáltuk közösen kiküszöbölni a problémát és megoldani azt. Körülbelül heti rendszerességgel történtek ilyen megbeszélések a munka elkezdésétől számítva.

# Adatbázis

## 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](https://hu.wikipedia.org/wiki/Ny%C3%ADlt_forr%C3%A1sk%C3%B3d) dokumentumorientált [adatbázis](https://hu.wikipedia.org/wiki/Adatb%C3%A1zis) 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](https://hu.wikipedia.org/wiki/NoSQL) adatbázisszerverek közé tartozik. A dokumentumokat [JSON](https://hu.wikipedia.org/wiki/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.

## Az adatbázis leírása, magyarázata

A képen szöveg, számítógép, fedett pályás, computer látható

Automatikusan generált leírásA 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: Hozzászólások, Lemezek, Messages, Postok, Ratings, Termékek, Users, Wishlist. Ezeken belül vannak tárolva az adatok dokumentum formában. Maga az adatbázis kialakítása látható a következő képen:

## Adatbázis diagramja

MongoDB adatbázisunk modellje

# A backend fejlesztése

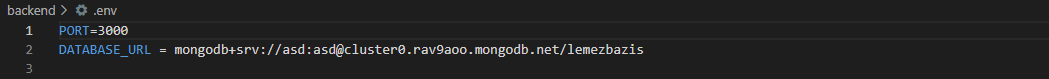
## A fejlesztői környezet, adatbázis kommunikáció

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.

A képen szöveg látható

Automatikusan generált leírásAz 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 szintaxist 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 összekötéséhez:

Mongoose összeköttetés, dotenv

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

3000-es port, valamint adatbázis URL

## 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:

A képen szöveg látható

Automatikusan generált leírás

models/Lemez.js példa

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

„getLemezek": Az összes lemezt listázza az adatbázisból. Lehetőség van szűrésre előadó, évjárat vagy műfaj alapján. A lemezekhez tartozó termékeket 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.

A képen szöveg látható

Automatikusan generált leírás

controllers/lemezek.js példa

Miután felvettünk minden kollekciónak a megfelelő vezérlőjét, jött 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.

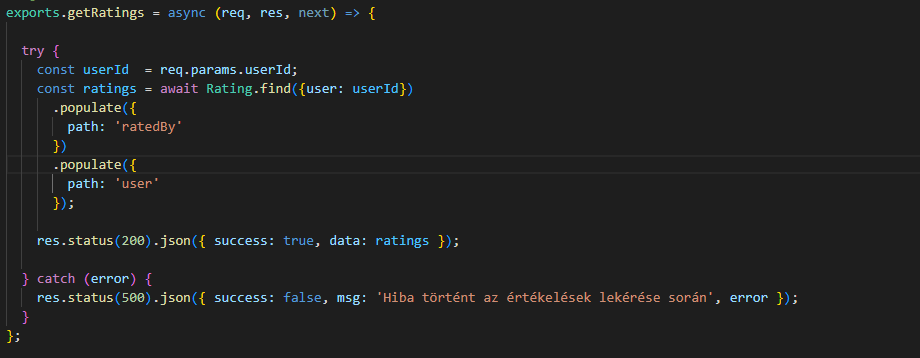
A szöveg alatti példában látható, hogy adott végpontra küldött, GET, POST, PUT, DELETE metódussal hívott kéréseket kötnek össze a különböző „controller” mappából beimportált álA képen szöveg látható

Automatikusan generált leírásA képen szöveg látható

Automatikusan generált leíráslományban szereplő visszahívó függvényekkel.

routes/lemezek.js példa

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.

Populate példa 1.

Populate példa 2.

## 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 leirásnak létrehozott indexet, mellyel megoldódott a probléma és működött a lekérdezés.

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 teszteléséhez, 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 fejlesztésében és karbantartásában.

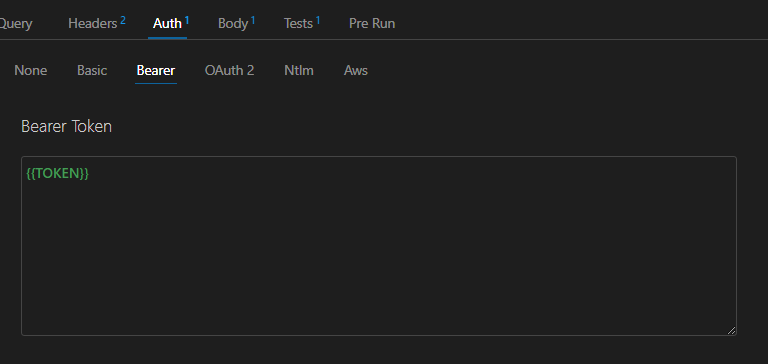
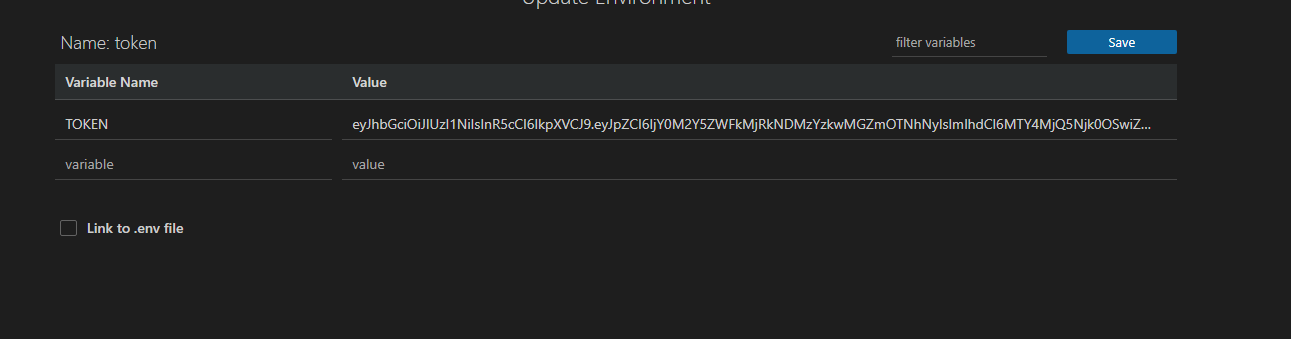
# REST API

## 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 írattuk, 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.

## 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.

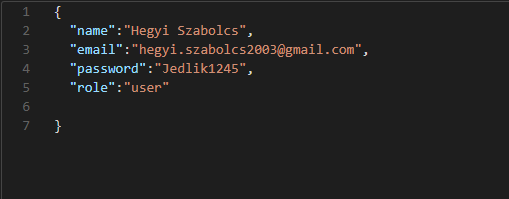
Environment

Authentication/Bearer

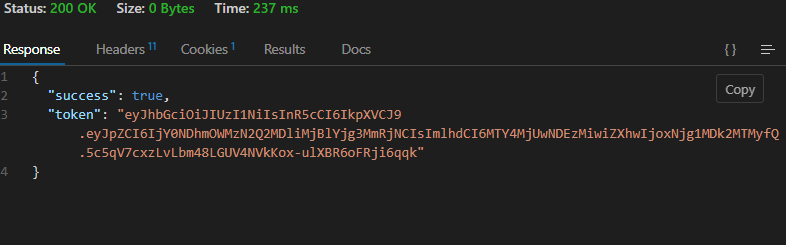
## Tesztelések

### Felhasználó regisztrálása, bejelentkezése

Tunder Client-be „teszt” Kollekciónk 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

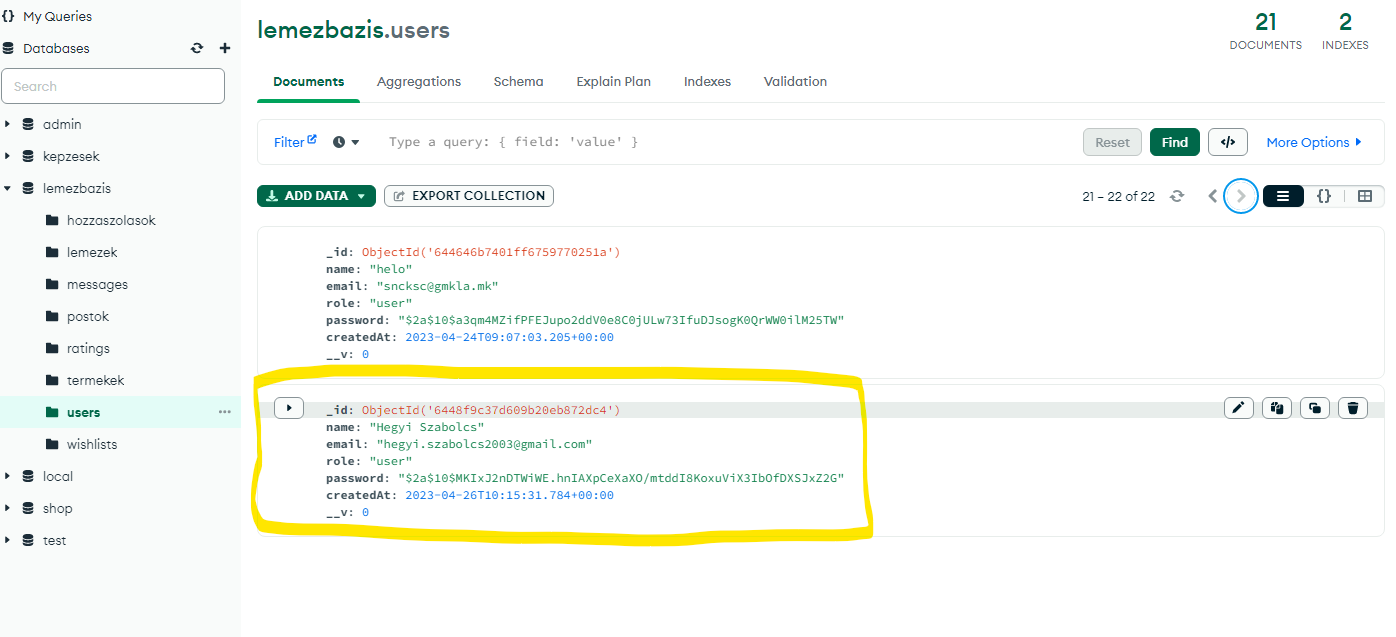
Body/Json/Json Content:

JSON Content

Amire a válasz:

Response text

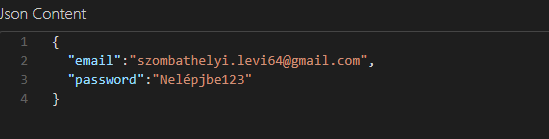
Eredmény:



Adatbázisban megjelenő regisztrált felhasználó

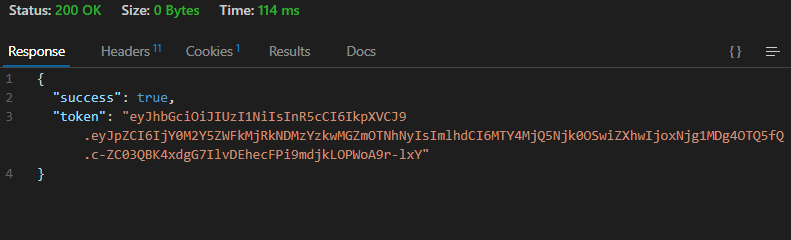
Egy felhasználó bejelentkezése a következőképpen néz ki: POST localhost:3000/api/auth/login

Body/JSON/Content:



Json Content

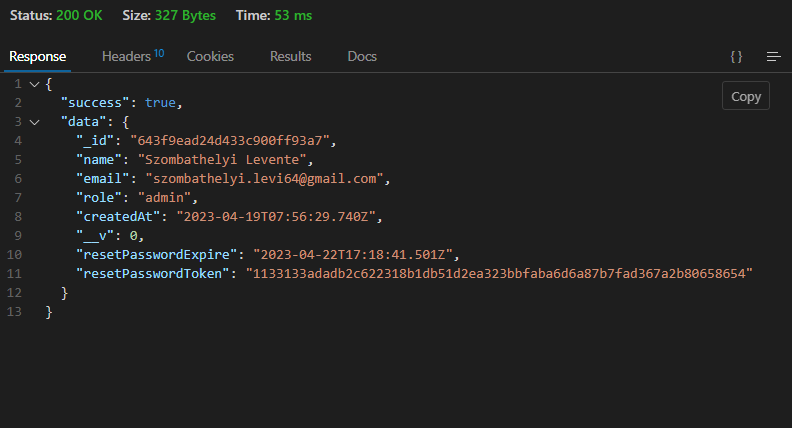
Amire a válasz:



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:



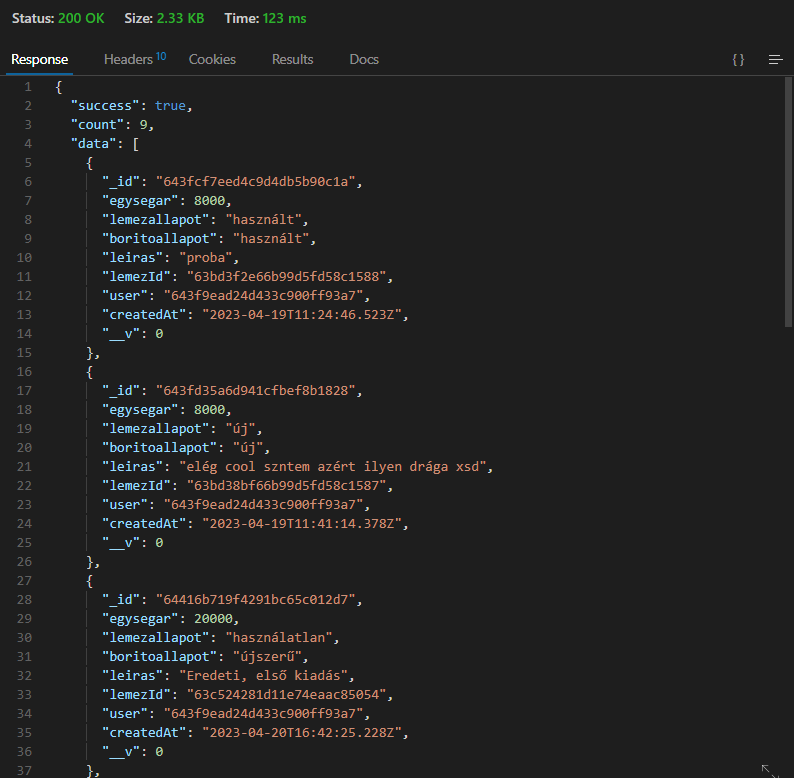
Response text

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

### 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/termekek/

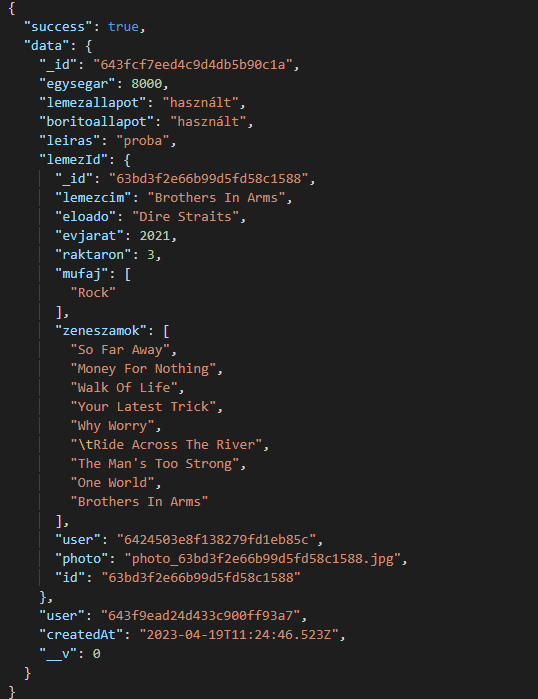
Válasz:



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/termekek/:id

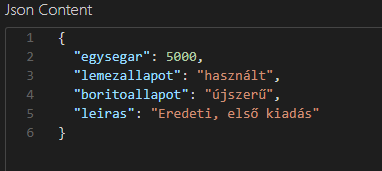
Válasz üzenetként:



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/termekek

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



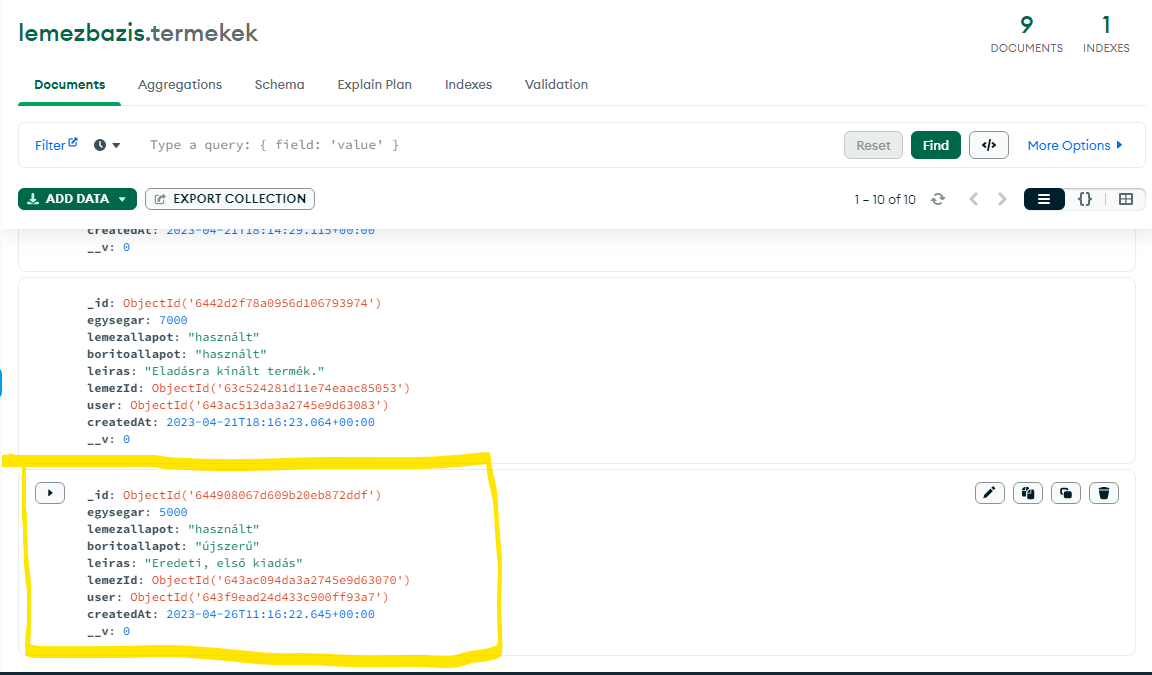
JSON Content

Válasz üzenet:



Response text

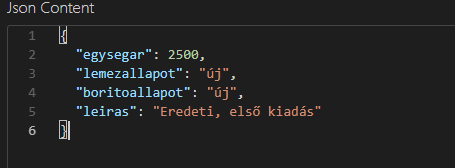
És végül a következő POST kérés eredménye melyen látszik, hogy megjelenik az adatbázisunkban:



Adatbázisban megjelenő termék

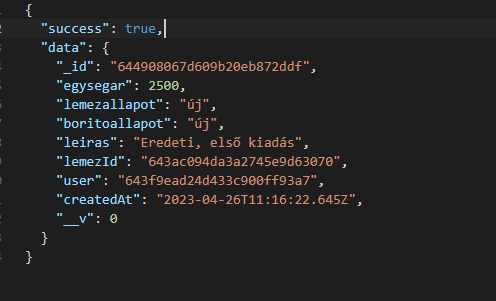
A feltöltött termékeket terszmészetesen lehet módosítani, amit a következő végponttal tehetünk meg. PUT localhost:3000/api/termekek/:termekId. JSON Contentbe ugyanazon paraméterek megadása kötelező a módosításhoz, mint ami a feltöltéshez kellett:

Body/JSON/Content:



JSON Content

Válasz:



Response text

Eredményül az adatbázisba való változtatások is megtörténtek:



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/termekek/termekId.

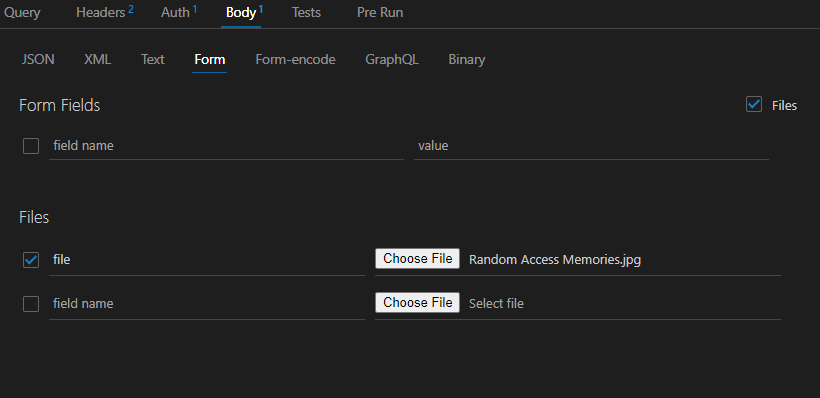
A lekérdezést elküldése után a következő választ üzenetet kapjuk:

A képen szöveg látható

Automatikusan generált leírás

Response text

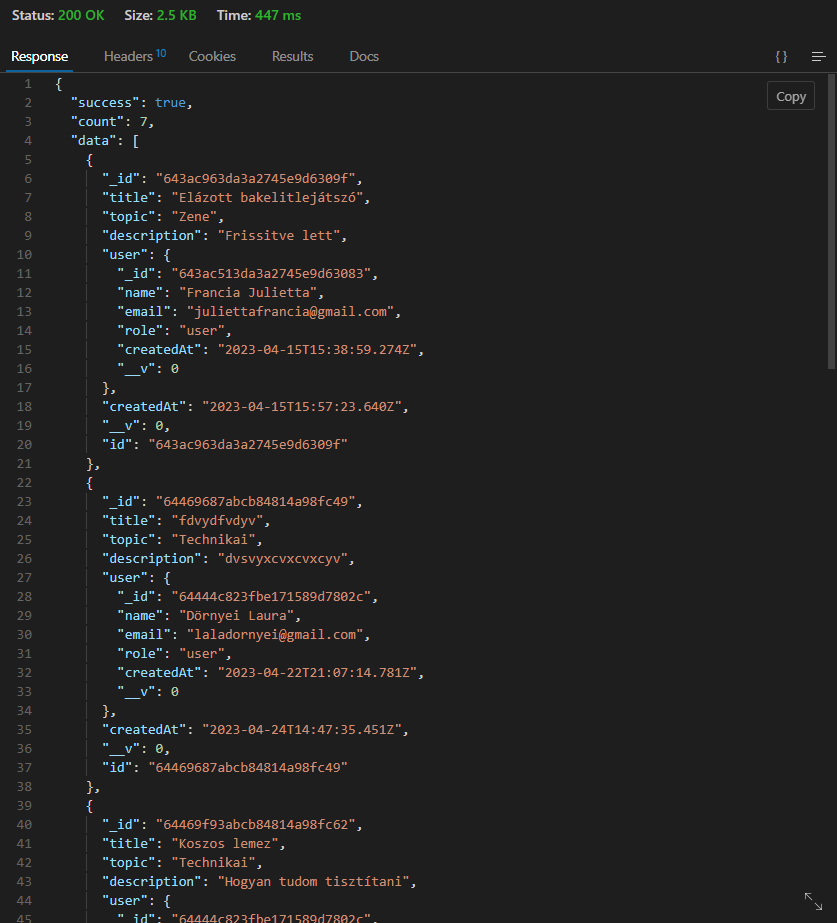
A termékekhez képet is lehet feltölteni a következővel: PUT localhost:3000/api/lemezek/:lemezId/photo. A kép feltöltéséhez Body/Form/Files elérési útvonalon kell feltöltenünk a kiválasztott képünket.



### Posztok lekérdezése, létrehozása, frissítése, törlése

Az oldalon megtalálható egy közösségi fórum, ahol a felhasználók tudnak feltölteni posztokat, különböző témákban, 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/

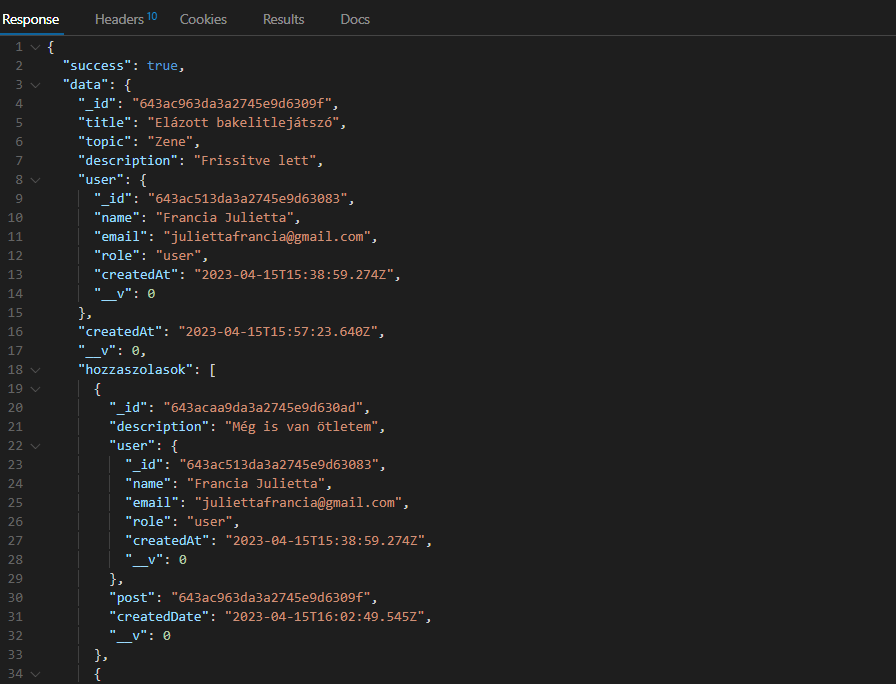
Válaszul a következőt kapjuk:



Response text

Mint látható az adatbázisban szereplő mind a hét poszt megjelenik, amit a válaszüzenet ki is 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.

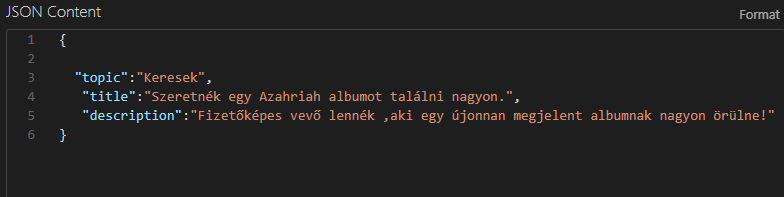
GET localhost:3000/api/postok/:postId



Response 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

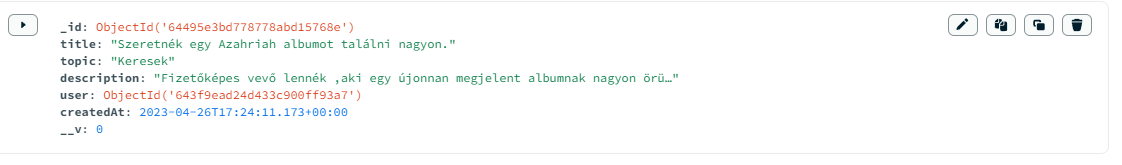
Válaszként a következőt kapjuk:

A képen szöveg látható

Automatikusan generált leírás

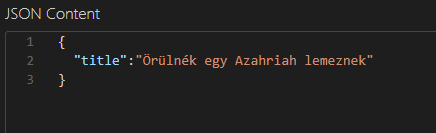
Response text

Eredmény az adatbázisunkban látható:



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

JSON Content:

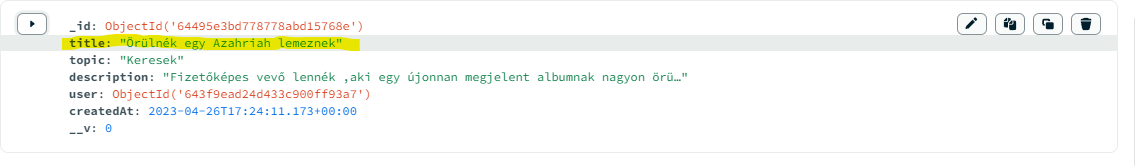


JSON Content

A képen szöveg látható

Automatikusan generált leírásVálasz:

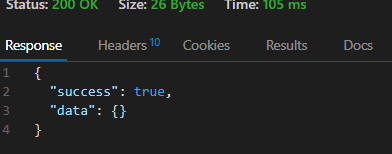
Response text



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



Response text

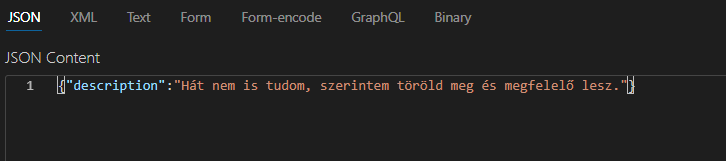
Mint látható a metódus sikeresen végbement, így a poszt törlésre került.

### Hozzászólások feltöltése, szerkesztése, törlése

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

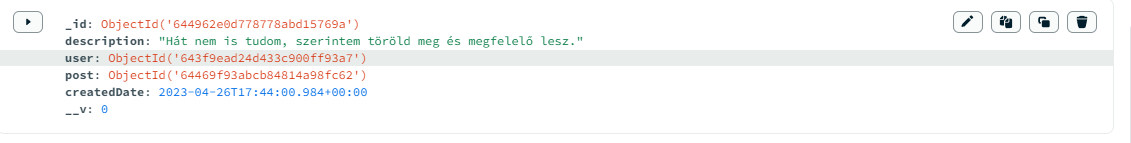
POST localhost:3000/api/postok/:postId/hozzaszolasok

JSON Content:



JSON Content

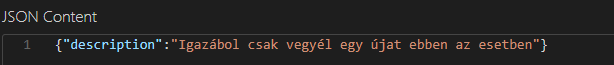
Adatbázisban a feltöltés után meg is jelenik az adatbázisban az imént írt hozzászólás:



Adatbázisban megjelenő eredmény

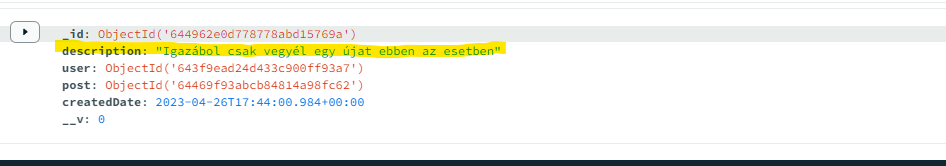
A hozzászólás szerkesztéséhez a következő API-s kérést kell beírni: PUT localhost:3000/api/hozzaszolasok/:hozzaszolasId

JSON Content:



JSON Content

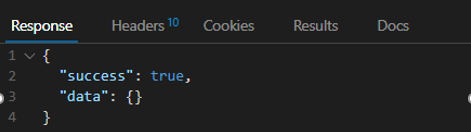
Adatbázisban megjelenő változás:



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:



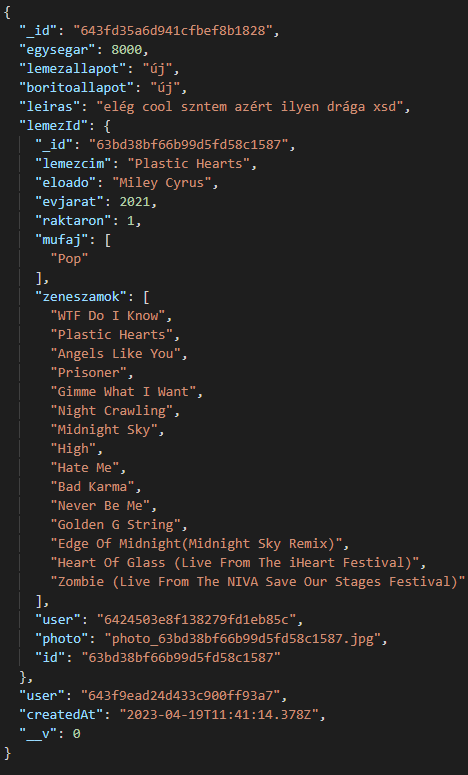
Response text

### Kívánságlista lekérdezése, hozzáadás, törlés

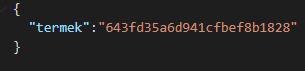
A felhasználók kívánt termékeit hozzátudják adni saját kívánságlistájukhoz, melyet onnantól kezdve könnyedén számon tudnak tartani.

GET localhost:3000/api/wishlists

Válasz üzenet:

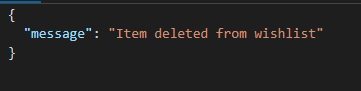


Hozzáadás a Kívánságlistához: POST localhost:3000/api/wishlists. JSON Content-be a következő elemek megadásával vihető fel egy termék:



Ugyanezen index megadásával törölhető a termék a Kívánságlistánkból:

DELETE localhost:3000/api/wishlists/:wishlistId



### Vélemények lekérése, hozzáadása, módosítása és törlése

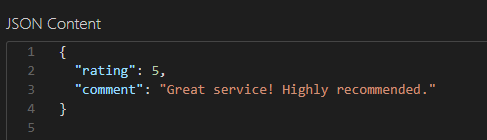
Vélemények hozzáadásához is van lehetőség az oldalon. Ezt a felhasználók egymásról tudják alkotni 1-5-ös skálán. Egy adott vélemény lekéréséhez szükség van a vélemény indexére és a következő végpont elküldésével meg is kapjuk válaszként: GET localhost:3000/api/ratings/:ratingId

Válasz üzenet:

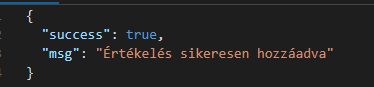


Mint látható magán a hozzászóláson és az értékelésen felül a lekérdezés kiírja, hogy a vélemény kiről is lett alkotva, valamint, hogy ki adta hozzá azt. Úgy, mint az eddigieknél is véleményt lehet hozzáadni is. POST localhost:3000/api/ratings/:userId

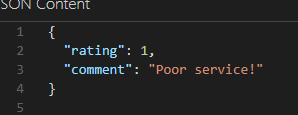
JSON Content:



Válasz üzenet sikeres küldéskor:



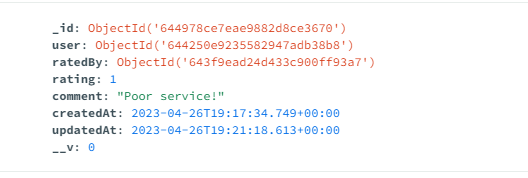
Vélemény frissítéséhez a következő végponton kell a változtatásokat megejteni: PUT localhost:3000/api/ratings/:ratingId



Válasz üzenet sikeres küldés esetén:

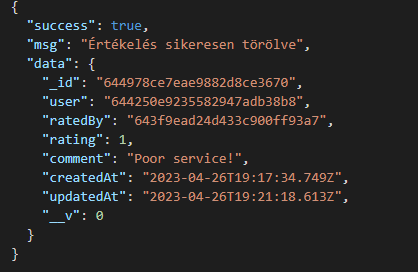


Eredmény az adatbázisban is látható:



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

Válasz:



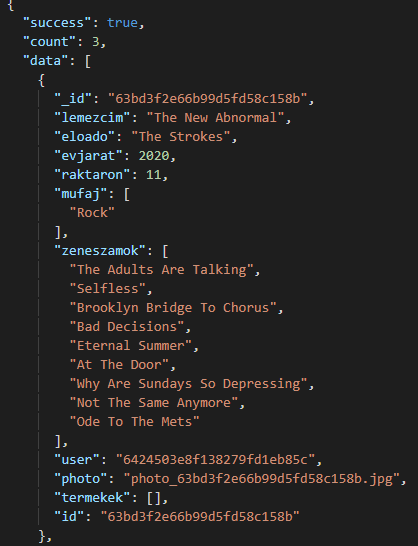
### Szűrések

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



Megjelenés alapján a szűrést következő végpont használatával lehet: GET localhost:3000/api/lemezek?evjarat=””

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



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

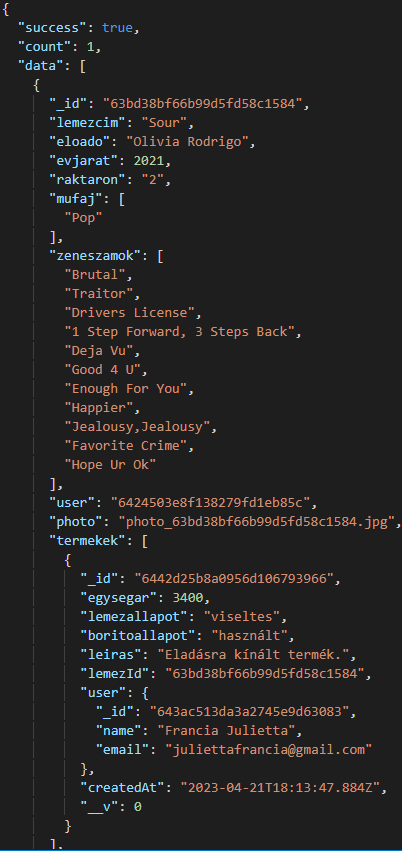
Válasz üzenet



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

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

Válasz üzenet:



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 témák között lehet válogatni: GET localhost:3000/api/postok?topic=””

Válasz sikeres küldés esetén:



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

### 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. |

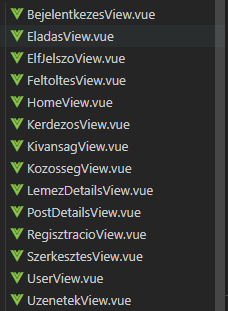
Fontos megjegyezni, hogy az egyes végpontok különböző hibaüzeneteket is visszaadhatnak a specifikus hibák jelzésére. Ezek az üzenetek az adott végpont dokumentációjában találhatók.

# A frontend fejlesztése

## 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 memóriahasználatra, 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:

**Views**: A views mappa az összes oldalt tartalmazza, amelyek megjelennek az alkalmazásban. Ezek az oldalak a routeren 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.

**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. |

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 az annak egyedi svg kódjával.

## A fejlesztés menetrendje, mérföldkövek

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 a programozást, és lefektettük az alapokat. App Vue-ba beilleszthető komponensek következtek: header, footer és navbar. Törekedtünk a letisztult megjelenésre, ezért a header tartalma mindössze egy kép, melyet magunk szerkesztettünk, hogy megfelelő legyen és illeszkedjen a weboldalunk témájához. Az eredeti képre még rá lett szerkesztve az oldalunk logója.

## A fejlesztés fontosabb megoldandó problémái

## Telefonos nézet, reszponzivitás

# Felhasználói útmutató:

A mobil nézetről leírás, képekkel, technológiája, lehet forráskód is

# Tesztelés

Kötelező elem a tesztelés. Ha készült automata teszt, akkor arról, ha nem, akkor minél részletesebben a szisztematikus manuális programtesztelésről. Lehet felhozni a tesztelés során felderített hibákat (lehetőleg megoldással).

# Felhasználói útmutató

Röviden a weboldal(program) használatáról. Nem kell alapvető részletesség, a magától értetődő dolgokat nem kell részletezni. Jöhetnek képek, de ne túl sok, pl. egy 2 beviteli mezős bejelentkezésről nem kell teljes képernyőkép.

# Összegzés

Foglalják össze az egész folyamatot így utólag visszatekintve.

Jó volt-e a feladat

Jó volt-e a csapat, a kommunikáció, a tudásuk

Mit sikerült, mit nem

Tartalom

[1. Bevezetés 1](#_Toc133259517)

[2. A csapat 3](#_Toc133259518)

[2.1. A csapat tagjai és feladataik 3](#_Toc133259519)

[2.2. A csapatmunka megvalósítása 3](#_Toc133259520)

[3. Adatbázis 4](#_Toc133259521)

[3.1. Az adatbázis diagramja 4](#_Toc133259522)

[3.2. Az adatbázis leírása, magyarázata 4](#_Toc133259523)

[3.3. Az adatbázis háttér technológiája, adatbáziskezelő program 4](#_Toc133259524)

[4. REST API 5](#_Toc133259525)

[5. A backend fejlesztése 6](#_Toc133259526)

[5.1. A fejlesztés eszközei, szoftverek, fejlesztői környezetek 6](#_Toc133259527)

[5.2. A fejlesztés menetrendje, mérföldkövek 6](#_Toc133259528)

[5.3. A fejlesztés fontosabb megoldandó problémái 6](#_Toc133259529)

[6. A frontend fejlesztése 7](#_Toc133259530)

[6.1. A fejlesztés eszközei, szoftverek, fejlesztői környezetek 7](#_Toc133259531)

[6.2. A fejlesztés menetrendje, mérföldkövek 7](#_Toc133259532)

[6.3. A fejlesztés fontosabb megoldandó problémái 7](#_Toc133259533)

[7. Mobil alkalmazás, mobil nézet 8](#_Toc133259534)

[8. Tesztelés 9](#_Toc133259535)

[9. Felhasználói útmutató 10](#_Toc133259536)

[10. Összegzés 11](#_Toc133259537)