# 

**Projektmunka**

**Témavezető:** **Készítette:**

Juhászné Kovács Ildikó Futó Tibor

Szabó Ákos Holovacki Román

NYÍREGYHÁZA

2023

**Tartalom**

[Feladat megadása 3](#_Toc132924298)

[1. Szereplők és igényeik 4](#_Toc132924299)

[2. Use-Case Funkció lista 5](#_Toc132924300)

[2.1. Látogató 5](#_Toc132924301)

[2.2. Bejelentkezett felhasználó 5](#_Toc132924302)

[2.3. Előfizető 6](#_Toc132924303)

[2.4 Adminisztrátor 6](#_Toc132924304)

[3. Magas szintű rendszerterv 7](#_Toc132924305)

[4. Képernyőképek 7](#_Toc132924306)

[5. Modellek 8](#_Toc132924307)

[6. Alkalmazások kiválasztása 11](#_Toc132924308)

[6.1. Front-end 11](#_Toc132924309)

[React 11](#_Toc132924310)

[JavaScript 11](#_Toc132924311)

[HTML5 11](#_Toc132924312)

[CSS3 12](#_Toc132924313)

[axios 12](#_Toc132924314)

[6.2. Backend 13](#_Toc132924315)

[Node.js 13](#_Toc132924316)

[Express 13](#_Toc132924317)

[mysql2 13](#_Toc132924318)

[Sequlieze 13](#_Toc132924319)

[bcrypt 14](#_Toc132924320)

[express-validator 14](#_Toc132924321)

[Nodemon 14](#_Toc132924322)

[concurrently 14](#_Toc132924323)

[6.3. Adatbázis 15](#_Toc132924324)

[XAMPP 15](#_Toc132924325)

[MariaDB 15](#_Toc132924326)

[6.4 API 15](#_Toc132924327)

[TMDB API (v3) 15](#_Toc132924328)

[7. Routing 16](#_Toc132924329)

[7.1. /authorization 16](#_Toc132924330)

[7.2. /authentication 16](#_Toc132924331)

[7.3. /api 17](#_Toc132924332)

[7.4. /admin/users 18](#_Toc132924333)

[7.5. /comments 19](#_Toc132924334)

[7.6 /favorite 20](#_Toc132924335)

[7.7 /rating 21](#_Toc132924336)

[7.8 /subscription 22](#_Toc132924337)

[7.9. /upload/avatar 23](#_Toc132924338)

[7.10. /watchlist 24](#_Toc132924339)

[8. Middlewarek (algoritmusok) 25](#_Toc132924340)

[9. Tesztelés 27](#_Toc132924341)

[10. Bevezetés, éles üzemmód 30](#_Toc132924342)

[Összegzés 31](#_Toc132924343)

# Feladat megadása

Feladatunk egy olyan Single-page application-t létrehozni, amely segít az embereknek keresés útján megállapítani, hogy egy film, vagy sorozat –továbbiakban **média**- elérhető-e valamilyen streaming szolgáltatás keretein belül, és ha igen, melyik az a szolgáltatás. Az ügyfélnek lehetősége van rákeresni médiára, részleteket megismerni róla, felhasználói profil létrehozására.  
Felhasználói profil megléte mellett hozzászólást írhat a médiához, értékelheti azt és kedvencei listájához adhatja. Emellett megismerheti melyek a közeljövőben megjelenő filmek.  
Előfizetés ellenében úgynevezett megnézendő és megnézettek listáját készíthet és elérheti a legnépszerűbb 50 médiát.

Az adminisztrátorok dashboardon felügyelhetik a felhasználók státuszát és jogait.   
Ők képes médiáknál található kommentek moderálása, ill. törlésére.

# Szereplők és igényeik

**Látogató**

* Tudjon böngészni
* Láthassa a médiák részleteit
* Legyen képes regisztrálni
* Legyen képes bejelentkezni

**Bejelentkezett felhasználó**

* Tudjon hozzászólást írni a médiákhoz.
* Láthassa a közeljövőben megjelenő médiákat
* Képes legyen a médiákat a kedvencei listájához adni
* Adhasson meg saját értékelést a médiákhoz

**Előfizető**

* Hozhasson létre megnézendő és megnézettek listáját
* Elérhesse azt 50 legnépszerűbb médiát

**Adminisztrátor**

* Felhasználó felügyelete
* Kommentek moderálása és törlése

# 2. Use-Case Funkció lista

A weboldalon a felhasználókat négy csoportra osztjuk. Ezek a látogatók, bejelentkezett felhasználók, az előfizetéssel rendelkező felhasználók és az adminisztrátorok. A weboldal központi működéséhez minden felhasználó hozzáfér, és használhatja azt, viszont vannak további lehetőségek az eszköztárukban.

## 2.1. Látogató

* Böngészés
* Regisztráció
* Bejelentkezés

## 2.2. Bejelentkezett felhasználó

* Böngészés
* Média értékelése
* Média hozzáadása, törlése „Kedvenceim” listáról
* Saját komment létrehozása, szerkesztése és törlése
* Hamarosan megjelenő filmek megtekinthetősége
* Profilkép feltöltése, kivéve mozgóképet
* Előfizetés
* Kijelentkezés

## 2.3. Előfizető

* Böngészés
* Média értékelése
* Média hozzáadása, törlése „Kedvenceim” listáról
* Média hozzáadása „Megnézendő” vagy „Megnézett” listához
* „Megnézendő” listán lévő média áthelyezésre „Megnézett” listára
* „Megnézett” listán lévő média áthelyezésre „Megnézendő” listára
* Aktuális 50 legnépszerűbb film listájának megtekintése
* Aktuális 50 legnépszerűbb sorozat listájának megtekintése
* Saját komment létrehozása, szerkesztése és törlése
* Hamarosan megjelenő filmek megtekinthetősége
* Profilkép feltöltése, akár mozgóképet is
* Kijelentkezés

Ha a látogató további kiváltságokra tart igényt akkor ez esetben van lehetősége beregisztrálni illetve kívánság szerint előfizetést igényelni. A regisztrációkor meg kell adnia a felhasználónevét, e-mail címét valamint jelszavát. Regisztrálás után szabadon bejelentkezhet bármely általa kívánt időpontban. Ezáltal lehetősége adódik, hozzáférni a bejelentkezett felhasználók kiváltságaihoz valamint előfizethet igy további lehetőségekhez tesz szert.

## 2.4 Adminisztrátor

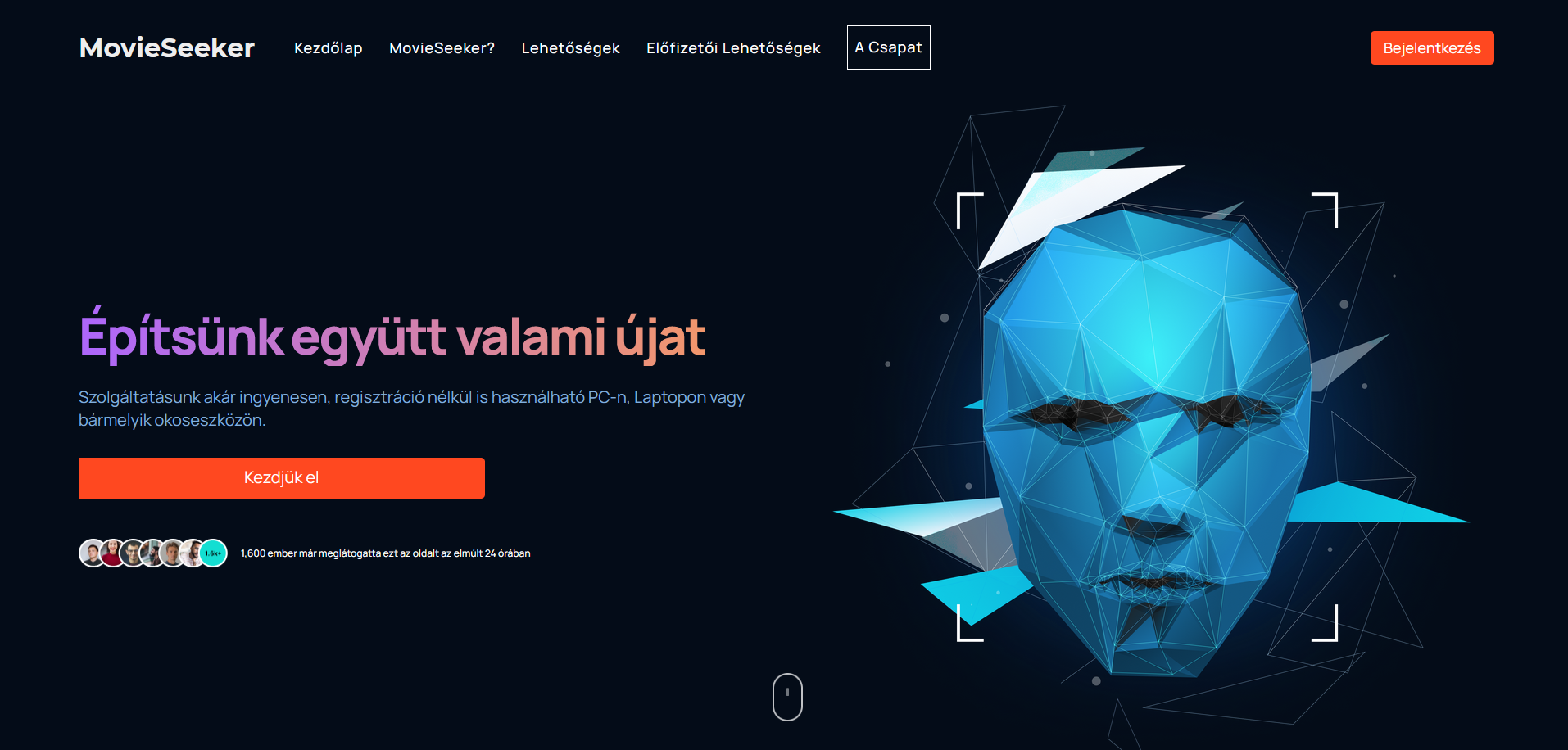
* Böngészés
* Média értékelése
* Média hozzáadása, törlése „Kedvenceim” listáról
* Saját komment létrehozása
* Bármelyik komment módosítása, vagy törlése
* Hamarosan megjelenő filmek megtekinthetősége
* Profilkép feltöltése, kivéve mozgóképet
* Előfizetés
* Kijelentkezés
* Admin Panel elérése, ahol kioszthat, vagy megvonhat adminisztrátori jogot felhasználótól

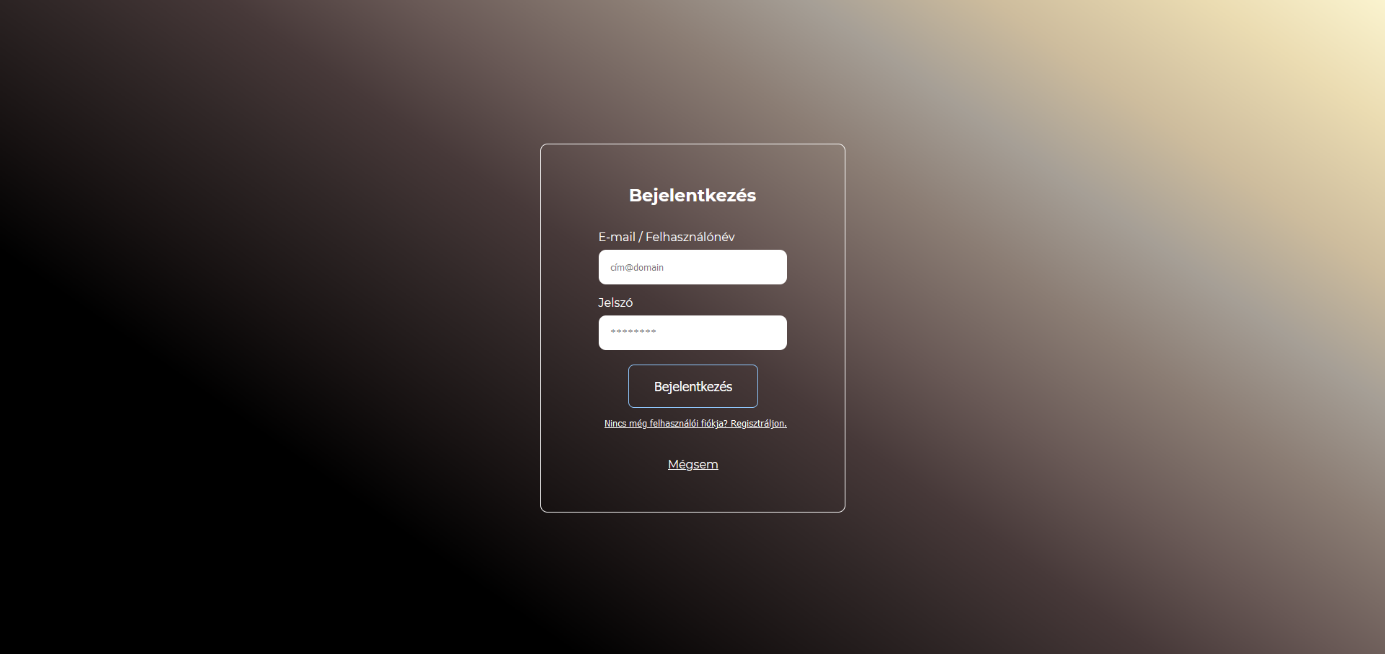
# 3. Magas szintű rendszerterv

A weboldalon minden adatot elkülönítettünk igy az esetleges felhasználói interakciótól azok nem változnak meg, valamint ezáltal könnyebbé tettük a későbbi továbbfejlesztés folyamatát.

# 4. Képernyőképek

A fejlesztés során elsődleges szempontként a felhasználói felületet részesítettük előnyben, ezáltal a felhasználó érthetően, egyszerűen kezelheti a szolgáltatásunkat egy modern, letisztult felületen.



A regisztrációs felület a lehető legminimálisabban terheli a felhasználót a nem kívánatos zavarodottság elkerülése érdekében.

# 5. Modellek

Az adatok tárolására több modellre is szükség van, úgy, mint a felhasználók, kommentek, értékelések, média tartalom.

**media {**

* id (int)
* imdb\_id (varchar)
* tmdb\_type (enum)
* tmdb\_id (int)
* title (title)
* original\_title (title)
* poster\_path (title)
* created\_at (date)
* updated\_at (date)

}

**users {**

* id (int)
* username (varchar)
* email (varchar)
* hash (char)
* rank (enum)
* created\_at (date)
* updated\_at (date)
* deleted\_at (date)

}

**comments** {

* id (int)
* content (varchar)
* parent\_id (int)
* created\_at (date)
* updated\_at (date)
* deleted\_at (date)
* user\_id (int)
* media\_id (int)

}

**favorites {**

* user\_id (int)
* media\_id (int)
* created\_at (date)

}

**ratings {**

* user\_id (int)
* media\_id (int)
* rating (int)
* created\_at (date)
* updated\_at (date)

}

**subscriptions {**

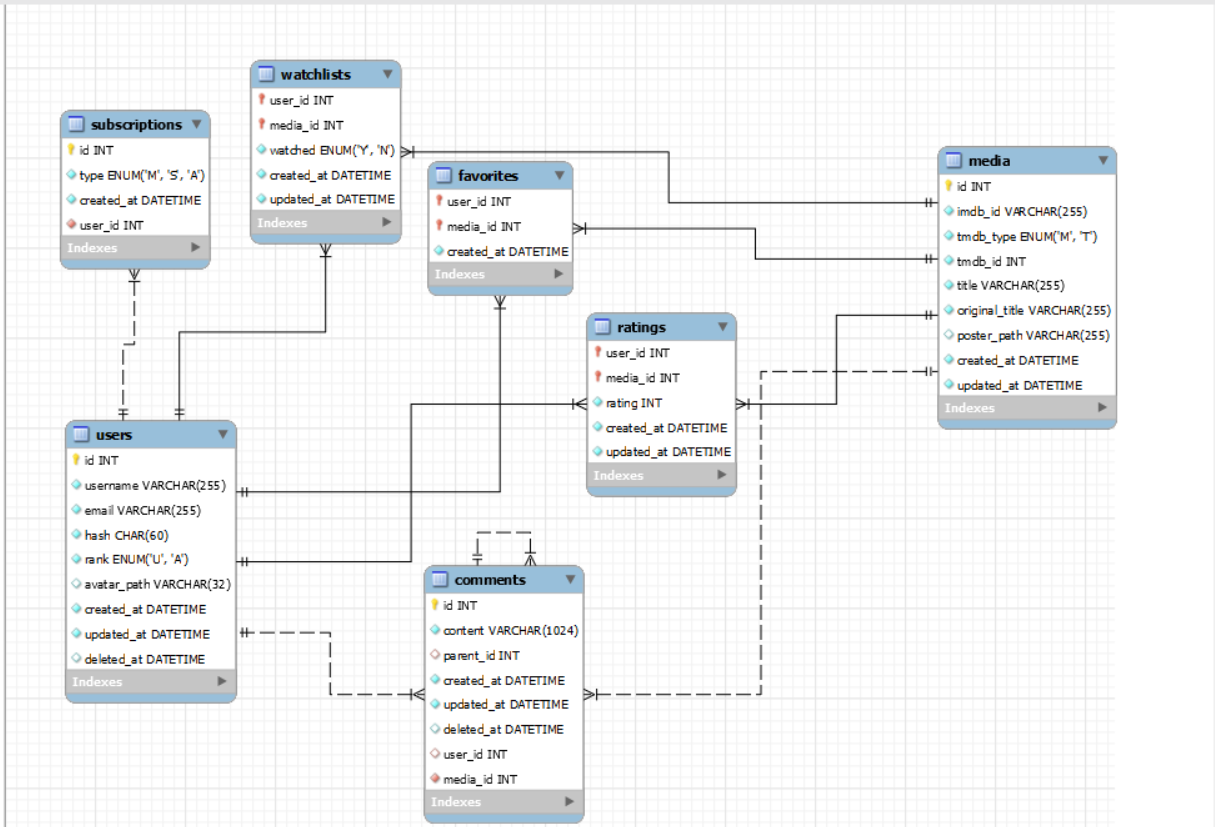
* id (int)
* type (enum)
* created\_at (date)
* end\_at (date)
* user\_id (int)

}

**favorites {**

* user\_id (int)
* media\_id (int)
* watched (enum)
* created\_at (date)

}



Az adatbázis UML diagrammja

# 6. Alkalmazások kiválasztása

## 6.1. Front-end

### React

Egy nyílt forráskódú JavaScript könyvtár felhasználói felületek létrehozására, amely az egyoldalas alkalmazások fejlesztése során felmerülő weboldalak tartalmának részleges frissítési problémáinak megoldására szolgál.

### JavaScript

Dinamikus, objektum-orientált prototípikus programozási nyelv. Az ECMAScript szabvány megvalósítása. Leggyakrabban weboldal szkriptek készítésére szolgál, amely lehetőséget biztosít a kliens oldalon (végfelhasználói eszközön) a felhasználóval való interakcióra, a böngésző vezérlésére, a szerverrel való aszinkron adatcserére, a weboldal szerkezetének és megjelenésének megváltoztatására.

### HTML5

Az HTML (Hypertext Markup Language) egy olyan kódolási nyelv, amelyet a weboldalak létrehozására használnak. Az HTML segítségével strukturált tartalmat és formázást adhatunk a weboldalaknak, beleértve a szöveget, képeket, videókat, hivatkozásokat, űrlapokat és sok más elemet. Az HTML az internet alapvető nyelve, és az összes weboldal felépítésének alapja. Az HTML-t böngészők értelmezik, hogy a tartalom megjelenjen a felhasználóknak a weboldalon. A HTML5 a HTML ötödik verziója, amely az előző verziókhoz képest számos új funkcióval és lehetőséggel rendelkezik, amelyek lehetővé teszik a weboldalak sokkal dinamikusabb és interaktívabb kialakítását.

### CSS3

Az CSS3 (Cascading Style Sheets 3) egy stíluslap nyelv, amelyet weboldalak és webalkalmazások kinézetének és elrendezésének meghatározására használnak. Az első verziója, az CSS1, 1996-ban jelent meg, majd az CSS2 1998-ban. Az CSS3 az előző verziók fejlesztése, amelynek számos új funkciója és lehetősége van, mint például a jobb képkezelés, animációk, árnyékok, átlátszóság, új betűtípusok és hátterek. Az CSS3 hozzájárul a modern webfejlesztéshez, és segít a weboldalak szép és dinamikus megjelenítésében.

### axios

Az Axios egy nyílt forráskódú JavaScript könyvtár, amely lehetővé teszi a fejlesztők számára az aszinkron HTTP kérések végrehajtását böngészőkben és Node.js környezetben. Az Axios egyszerű és könnyen használható interfészt kínál az adatok szerverről való lekérdezéséhez vagy küldéséhez, és támogatja az összes modern böngészőt és Node.js-t is. Az Axios támogatja a Promise API-t, így könnyű hibakezelést és adatmanipulációt biztosít az aszinkron HTTP kérések során.

## 6.2. Backend

### Node.js

A Node.js egy nyíltforrású, többplatformos, szerveroldali JavaScript futtatókörnyezet a Google Chrome V8 JavaScript-motoron alapulva. Leginkább webes applikációk, dinamikus weboldalak fejlesztésére használják, de akár szerveroldali feldolgozószkriptek készítésére is kiválóan alkalmas.

### Express

Az Express egy Node.js alapú keretrendszer, amely segít a webalkalmazások és API-k készítésében. Az Express egyszerű és könnyen használható, és számos olyan funkcióval rendelkezik, amelyek megkönnyítik a fejlesztést, mint például az útvonalak kezelése, a middleware-ek használata, a sablonmotorok integrációja és a HTTP kérés és válasz kezelése. Az Express rugalmas és könnyen testre szabható, így a fejlesztők nagyon sokféleképpen használhatják, attól függően, hogy milyen típusú alkalmazást szeretnének készíteni.

### mysql2

MySQL2 csomag egy Node.js-ben írt modul, amely lehetővé teszi a MySQL adatbázis-kezelő rendszer használatát Node.js alkalmazásokban. A csomag a MySQL adatbázis-kezelő rendszerrel való kommunikációhoz kínál egyszerű és hatékony API-t. A MySQL2 csomag magas szintű funkciókat kínál, például támogatja a többutas lekérdezéseket, az előkészített utasításokat, a szövegkezelést és az üzenetküldést. Emellett a csomag lehetővé teszi a MySQL adatbázis-kezelő rendszerrel való összeköttetés felépítését, az adatbázisok és táblák kezelését, a tranzakciók kezelését és az adatok lekérdezését és frissítését.

### Sequlieze

A Sequelize egy Node.js-ben írt ORM (Object-Relational Mapping) keretrendszer, amely lehetővé teszi az adatbázisokkal való kommunikációt. Az ORM egy olyan programozási technika, amely lehetővé teszi az adatbázis-kezelő rendszerek és az alkalmazások közötti átjárhatóság növelését, valamint lehetővé teszi az adatok könnyebb kezelését az alkalmazásban. A Sequelize támogatja a számos adatbázis-kezelő rendszert, például a MySQL-t, PostgreSQL-t, SQLite-ot és a Microsoft SQL Server-t is.

### bcrypt

A bcrypt npm modul egy Node.js környezetben használható csomag, amely lehetővé teszi a bcrypt használatát a Node.js alkalmazásokban. A modul segítségével a Node.js fejlesztők könnyen hashelhetik és összehasonlíthatják a jelszavakat a bcrypt algoritmus használatával.  
A bcrypt használata javasolt, ha a Node.js alkalmazásban jelszavakat kell hashelni és tárolni. A modul biztonságosan kezeli a jelszavakat, és nehézkes folyamatot alkalmaz a hash-elésre, így nehezebbé teszi a jelszavak feltörését és védelmet nyújt a biztonsági támadások ellen.

### express-validator

Az express-validator egy Node.js alapú könyvtár, amelyet az Express keretrendszerrel használnak együtt, hogy ellenőrizzék az HTTP kéréseket és válaszokat. Az express-validator segítségével könnyen és hatékonyan lehet validálni a beérkező adatokat, ellenőrizni azok érvényességét és kezelni a hibákat. A könyvtár különböző validációs szabályokat tartalmaz, amelyeket egyszerűen lehet alkalmazni az adatokra. Az express-validator a fejlesztők számára időt takarít meg és segíti őket abban, hogy biztonságos és megbízható alkalmazásokat hozzanak létre.

### Nodemon

A Nodemon egy olyan szoftver, amelyet a Node.js platformhoz fejlesztettek ki, és az fejlesztési folyamatok során használható. A Nodemon figyeli az alkalmazás forrásfájljait, és amikor változtatást észlel bennük, automatikusan újra indítja az alkalmazást, így nem kell manuálisan újra elindítania minden egyes módosítás után.

### concurrently

A "concurrently" npm csomag egy parancssoros eszköz, amely lehetővé teszi a több parancs egyidejű futtatását egyetlen parancssoros ablakban. Ez különösen hasznos fejlesztés során, amikor például szeretnénk egyszerre futtatni több folyamatot, mint például egy fejlesztői szerver indítása és a forráskód figyelése változásokra. Ez teszi lehetővé projektünk során a frontend és a backend szerver egyetlen parancsban való elindítását.

## 6.3. Adatbázis

### XAMPP

Az XAMPP egy ingyenes és nyílt forráskódú szoftvercsomag, amelyet webalkalmazások fejlesztéséhez és teszteléséhez használnak. A neve az Apache web szerver, a MySQL adatbázis, a PHP programozási nyelv és a Perl programozási nyelv betűiből származik. Az XAMPP telepítése után a felhasználóknak lehetősége van arra, hogy saját számítógépükön futtassák és teszteljék a weboldalakat, mielőtt azokat az interneten közzétennék.

### MariaDB

MariaDB egy nyílt forráskódú relációs adatbázis-kezelő rendszer, amely az eredeti MySQL adatbázis-kezelő rendszer ágazataként jött létre. A MariaDB számos funkcióval rendelkezik, amelyek javítják az adatbázis teljesítményét, skálázhatóságát és megbízhatóságát. Az egyik legjelentősebb különbség a MySQL-hoz képest az, hogy a MariaDB több alapértelmezett tároló motorral rendelkezik, beleértve az XtraDB (InnoDB), Aria, MyISAM és a ColumnStore tároló motort is. Emellett a MariaDB támogatja az SQL szabványok egy sor újabb verzióját, így több funkció érhető el az adatbázis-kezelő rendszerben.

## 6.4 API

### TMDB API (v3)

A TMDB API (The Movie Database API) egy olyan nyilvánosan elérhető API (alkalmazásprogramozási felület), amely lehetővé teszi a fejlesztők számára, hogy hozzáférjenek a The Movie Database adatbázisához és különböző film- és televíziós műsor adatokat kérjenek le. Az adatbázisban megtalálhatók a filmek és televíziós műsorok címei, leírásai, értékelései, szereplői, rendezői, stábtagjai, posztereik és egyéb információik.

# 7. Routing

Végpontjainkat úgynevezett beágyazott útvonalakba (nested routes) szerveztük. A beágyazott útvonalak a webes alkalmazásokban használt útvonalak olyan hierarchikus beágyazása, amely lehetővé teszi, hogy az egyes útvonalakat egymásba ágyazva használjuk.

## 7.1. /authorization

**POST /register**

Tervezett feladat Felhasználó rögzítése

Bemenet Felhasználó adatai

Kimenet A felhasználó kliens oldalon használt adatai, autentikációs JWT

**POST /login**

Tervezett feladat Felhasználó beléptetése

Bemenet Felhasználó adatai

Kimenet A felhasználó kliens oldalon használt adatai, autentikációs JWT

## 7.2. /authentication

**GET /**

Tervezett feladat Felhasználó autentikálása

Bemenet Sütiben tárolt JWT

Kimenet A felhasználó kliens oldalon használt adatai

**DELETE /**

Tervezett feladat Felhasználó kiléptetése

Bemenet Sütiben tárolt JWT

Kimenet Tájékoztatás a művelet sikerességéről

## 7.3. /api

**GET /?s=value**

Tervezett feladat: Keresés médiára

Bemenet: Query sztringben a keresett média címe

Kimenet: Találatok listája

**GET /?m=value&i=value**

Tervezett feladat: Adott médiáról részletek megismerése

Bemenet: Query sztringben a média típusa és TMDB id-ja

Kimenet: A média részletei

**GET /popular/:type**

Tervezett feladat: Legnépszerűbb 50 média elérése, előfizetők számára

Bemenet: A média típusa, azaz, hogy tv vagy movie, sütiben tárolt JWT

Kimenet: Legnépszerűbb 50 média

**GET /upcoming**

Tervezett feladat: Hamarosan megjelenő filmek elérése, felhasználók számára

Bemenet: Sütiben tárolt JWT

Kimenet: Hamarosan megjelenő filmek listája

## 7.4. /admin/users

**GET /**

Tervezett feladat: Az összes regisztrált felhasználó adatainak elérése, adminisztrátorok számára

Bemenet: Sütiben tárolt JWT

Kimenet: Az összes regisztrált felhasználó adatainak listája

**PATCH /**

Tervezett feladat: Felhasználó rangjának módosítása, adminisztrátorok számára

Bemenet: Sütiben tárolt JWT, kiválasztott felhasználó id-ja, és új rangja

Kimenet: Tájékoztatás a művelet sikerességéről

## 7.5. /comments

**GET /:imdbId**

Tervezett feladat: Választott média hozzászólásainak lekérése

Bemenet: A médiához tartozó IMDb id

Kimenet: A médiához tartozó hozzászólások listája

**POST /**

Tervezett feladat: Hozzászólás hozzáadása egy médiához, regisztrált felhasználóknak

Bemenet: A média IMDb id-ja, és a sütiben tárolt JWT

Kimenet: Tájékoztatás a művelet sikerességéről

**PATCH /**

Tervezett feladat: Saját hozzászólás szerkesztése regisztrált felhasználóval, vagy hozzászólás szerkesztése adminisztrátor által

Bemenet: A hozzászólás id-ja, sütiben tárolt JWT token

Kimenet: Tájékoztatás a művelet sikerességéről

**DELETE /**

Tervezett feladat: Saját hozzászólás törlése regisztrált felhasználóval, vagy hozzászólás törlése adminisztrátor által

Bemenet: A hozzászólás id-ja, sütiben tárolt JWT token

Kimenet: Tájékoztatás a művelet sikerességéről

## 7.6 /favorite

**GET /**

Tervezett feladat: „Kedvenceim” lista elérése, bejelentkezett felhasználónak

Bemenet: Sütiben tárolt JWT

Kimenet: A felhasználó „Kedvenceim” listájának elemei

**POST /**

Tervezett feladat: Média hozzáadása „Kedvenceim” listához, bejelentkezett felhasználóknak

Bemenet: Sütiben tárolt JWT

Kimenet: Tájékoztatás a művelet sikerességéről

**DELETE /**

Tervezett feladat: Média eltávolítása „Kedvenceim” listáról, bejelentkezett felhasználóknak

Bemenet: Sütiben tárolt JWT

Kimenet: Tájékoztatás a művelet sikerességéről

## 7.7 /rating

**GET /:imdbId**

Tervezett feladat: Választott médiára leadott felhasználói értékelés elérése

Bemenet: Sütiben tárolt JWT, médiához tartozó IMDb id

Kimenet: Az értékelés (1-10-ig), ha nincs, akkor null

**POST /**

Tervezett feladat: Értékelés leadása médiára felhasználónak

Bemenet: Az IMDb és az értékelés (1-10), sütiben tárolt JWT

Kimenet: Az új átlag pontszám a médiára leadott összes értékelésből

**PATCH /**

Tervezett feladat: Felhasználó médiára leadott értékelésének módosítása

Bemenet: Az IMDb és az értékelés (1-10), sütiben tárolt JWT

Kimenet: Az új átlag pontszám a médiára leadott összes értékelésből

**DELETE /**

Tervezett feladat: Felhasználó médiára leadott értékelésének törlése

Bemenet: Az IMDb és az értékelés (1-10), sütiben tárolt JWT

Kimenet: Az új átlag pontszám a médiára leadott összes értékelésből

## 7.8 /subscription

**GET /**

Tervezett feladat: Lekérdezni, hogy a jelenleg bejelentkezett felhasználónak van-e aktív előfizetése

Bemenet: Sütiben tárolt JWT

Kimenet: 200-as http kód, ha igen, 403, ha nem

**POST /**

Tervezett feladat: Előfizetés bejelentkezett felhasználónak

Bemenet: Sütiben tárolt JWT, az előfizetés típusa

Kimenet: Tájékoztatás a művelet sikerességéről

## 7.9. /upload/avatar

**GET /:path**

Tervezett feladat: Bejelentkezett felhasználó profilképjének (avatárjának) lekérdezése

Bemenet: Sütiben tárolt JWT, a kép elérési útja

Kimenet: A profilkép, amennyiben létezik az útvonal

**POST /**

Tervezett feladat: Profilkép feltöltése

Bemenet: Sütiben tárolt JWT, a kép fájl

Kimenet: Tájékoztatás a művelet sikerességéről

## 7.10. /watchlist

**GET /**

Tervezett feladat: Az előfizető „Megnézendő” és „Megnézettek” listájának lekérdezése

Bemenet: Sütiben tárolt JWT

Kimenet: A két darab lista

**POST /**

Tervezett feladat: A média hozzáadása a választott listához

Bemenet: Sütiben tárolt JWT, IMDb id, watched paraméter, amely Y és N lehet

Kimenet: Tájékoztatás a művelet sikerességéről

**DELETE /**

Tervezett feladat: A média áthelyezése egyik listáról a másikra

Bemenet: Sütiben tárolt JWT, IMDb id, watched paraméter, amely Y és N lehet

Kimenet: Tájékoztatás a művelet sikerességéről

# 8. Middlewarek (algoritmusok)

A Middleware függvények definíció szerint olyan függvények, amik hozzáférnek a request (kérés - req) és response (válasz - res) objektumokhoz és kezelik azokat.   
(<http://www.inf.u-szeged.hu/~tarib/javascript/webszerverek.html#middleware-fuggvenyek-hibakezeles>)

Mint ahogy korábban említettük, szerver oldalon alkalmazásra került az „express-validator” npm csomag, amely middleware alapon valósítja meg az ellenőrzést, így a backend/src/middlewares/validators útvonalat alatt azok a middlewarek találhatóak, amelyek használja ezt a csomagot. Ugyanakkor ezen kívül is készültek middlewarek, amelyek alább olvashatóak.

* authenticationHandlerMiddleware()
  + Létrehozza a hitelesítéshez szükséges JWT tokent, letárolja sütiben, majd visszatér egy olyan JSON-nel, amely a felhasználó kliens oldalon használt adatait tartalmazza.
* registrationAuthorizationMiddleware()
  + Beregisztrálja a felhasználót.
* loginAuthorizationMiddleware()
  + Belépteti a felhasználót.
* checkErrorsMiddleware()
  + Elkéri az express-validatortól az eddigi feljegyzett, elutasításra került paraméterek –hibák- listáját. Ha ennek van legalább egy eleme, akkor elutasítja a kérést, ha nincs, tovább engedi azt.
* verifyJWTMiddleware()
  + A sütik közül kinyeri a felhasználót azonosító tokent (access-token) és érvényesíti azt a „jsonwebtokn” npm csomag verify metódusával. Ha az érvényesítés sikeres, akkor a payload-ban található felhasználói azonosítót (user id) felhelyezi a kérés törzsére, ellenkező esetben elutasítja a kérést.

Néhány példa a backend/src/middlewares/validators útvonalat alatt található middlewarekből, a teljesség igénye nélkül:

* generalNumberValidatorMiddleware(fieldName)
  + A kapott paraméterről megállapítja, hogy:
    - létezik-e
    - szám-e
* imdbIdValidatorMiddleware()
  + Megállapítja, hogy req-ben található imdbId:
    - létezik-e
    - megfelelő hosszú-e (9-255)
    - megfelelő formátummal rendelkezik (Reguláris kifejezés használatával ellenőrizve)
* isAdminValidatorMiddleware()
  + A kérés törzsében található userId-ből lekérdezi a felhasználót, és megállapítja, hogy adminisztrátor-e, vagy sem. Ha igen, tovább engedi a kérést, ha nem, elutasítja.
* userExistValidatorMiddleware()
  + A kérés törzsében található userId-ből lekérdezi a felhasználót, ha létezik felhasználó a megadott id-val, akkor felhelyezi a kérés törzsére a felhasználói objektumot, és tovább engedi a kérést, ha nem, elutasítja.
* subscriptionExistValidatorMiddleware()
  + A kérés törzsére felhelyezett felhasználó objektum segítségével eldönti, hogy a felhasználónak van-e aktív előfizetése. Ha van, tovább engedi a kérést, ha nem, elutasítja.

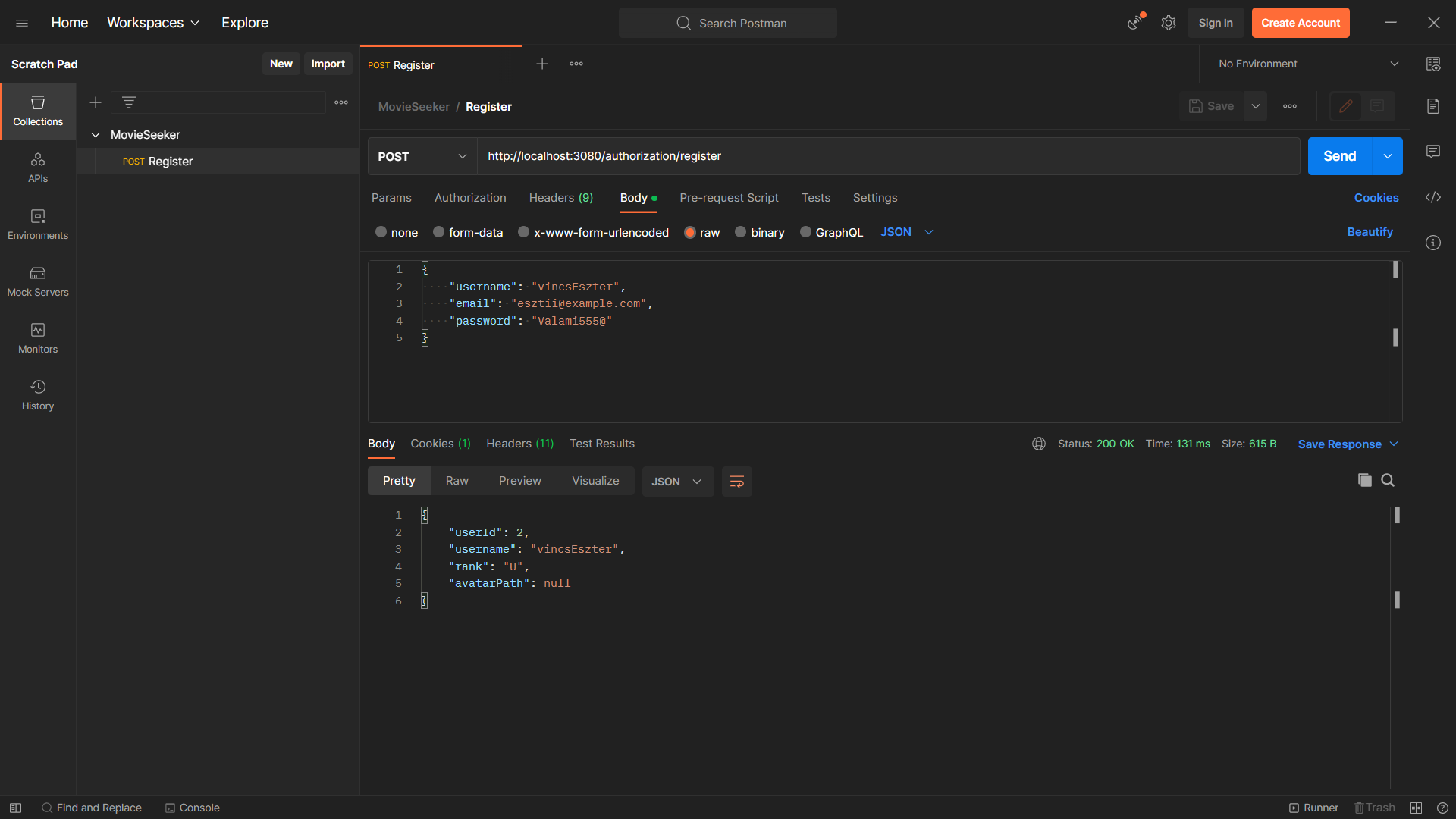
# 9. Tesztelés

A felhasználó szempontjából az alkalmazás használatához olyan eszköz szükséges, amely képes futtatni a mai modern böngészőket.

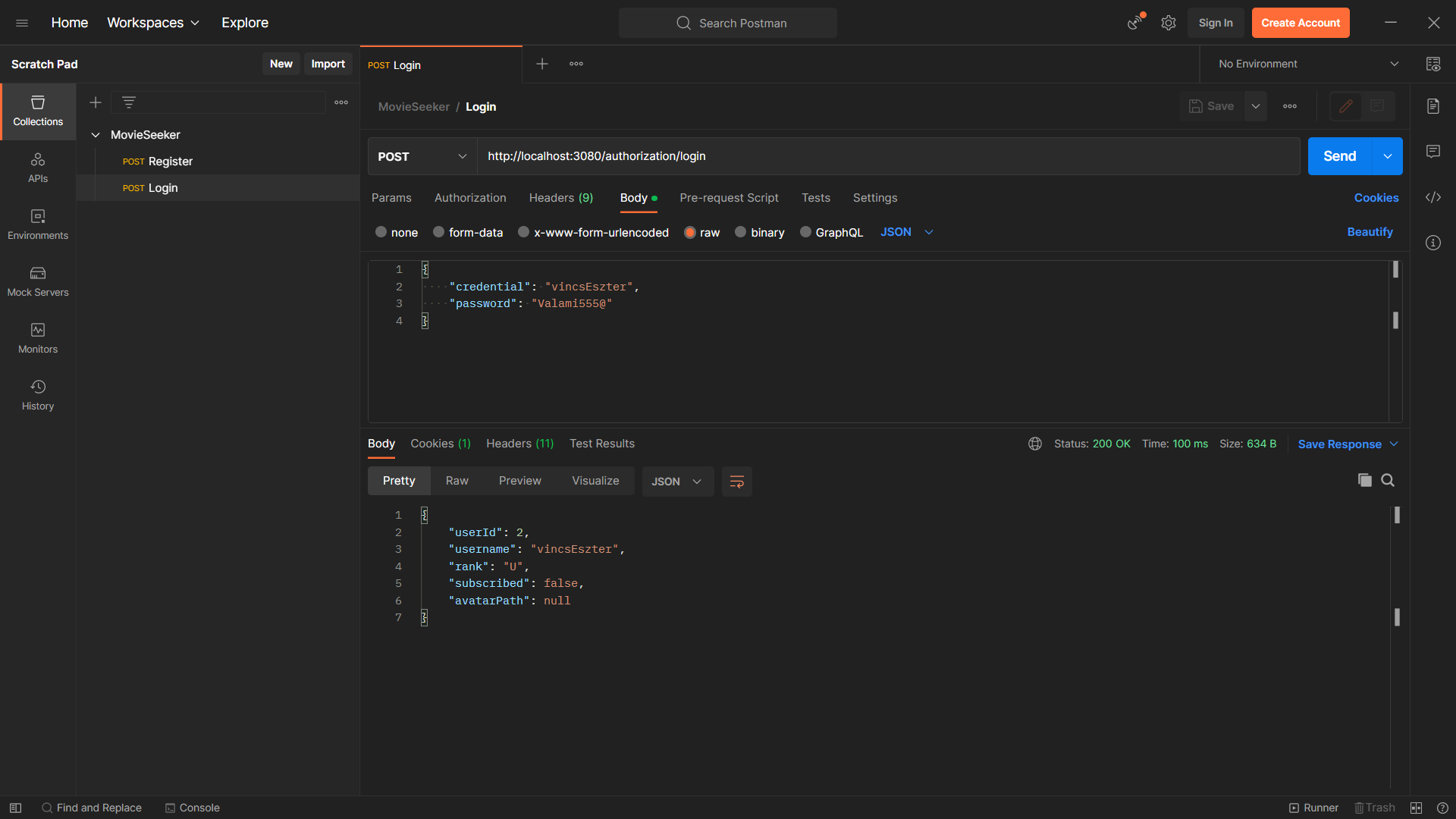
A rendszert több népszerű asztali böngészőn is teszteltünk, ezek pedig a Google Chrome (112.0.5615.138), a Microsoft Edge (112.0.1722.48) és az Opera (98.0.4759.6). Emellett Android operációs rendszerrel rendelkező okostelefonon is teszteltük az applikációt.

Az adatok mentéséhez szükségünk volt egy adatbázisra, ehhez pedig egy adatbázist szervert kellett futtatnunk. A fejlesztés során ennek megvalósításához a XAMPP beépített MariaDB szerverét használtuk.

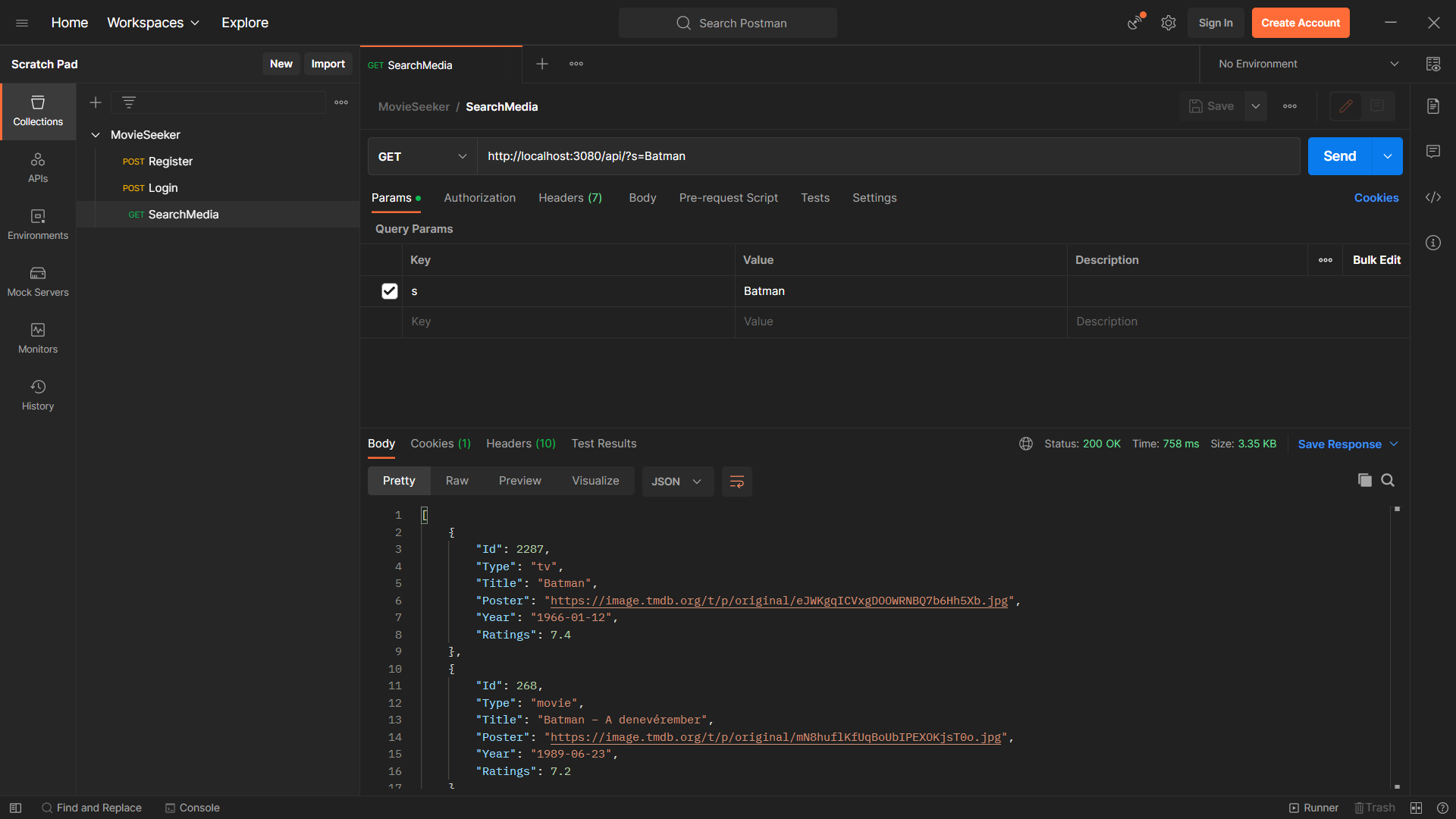
A routeok tesztelése legsokrétűbben a Postman szoftverrel valósítható meg, amelyet kifejezetten fejlesztőknek találtak ki API-jaik tervezésére, elkészítésére és tesztelésére.



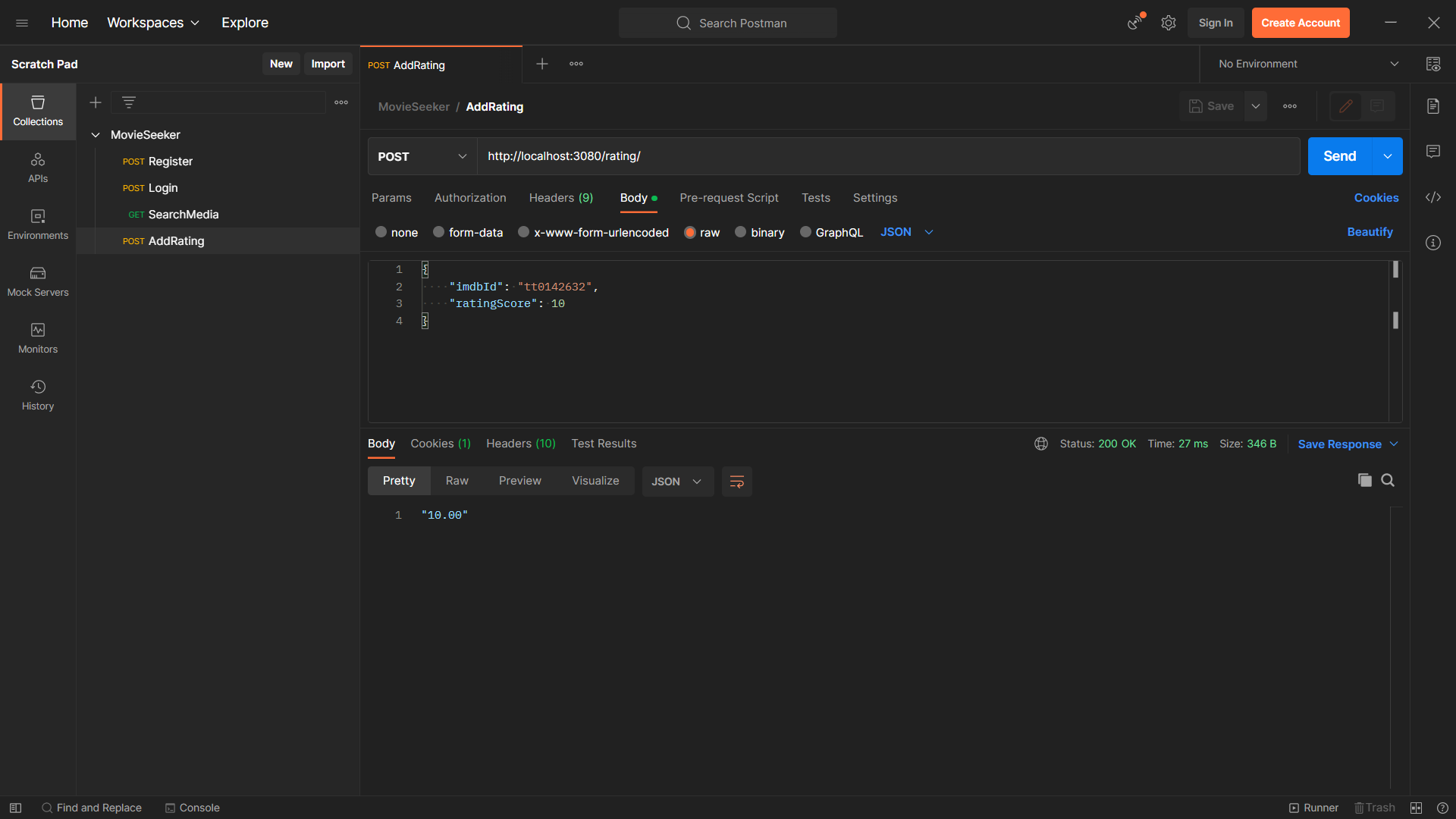
Regisztráció tesztelése Postman-ben



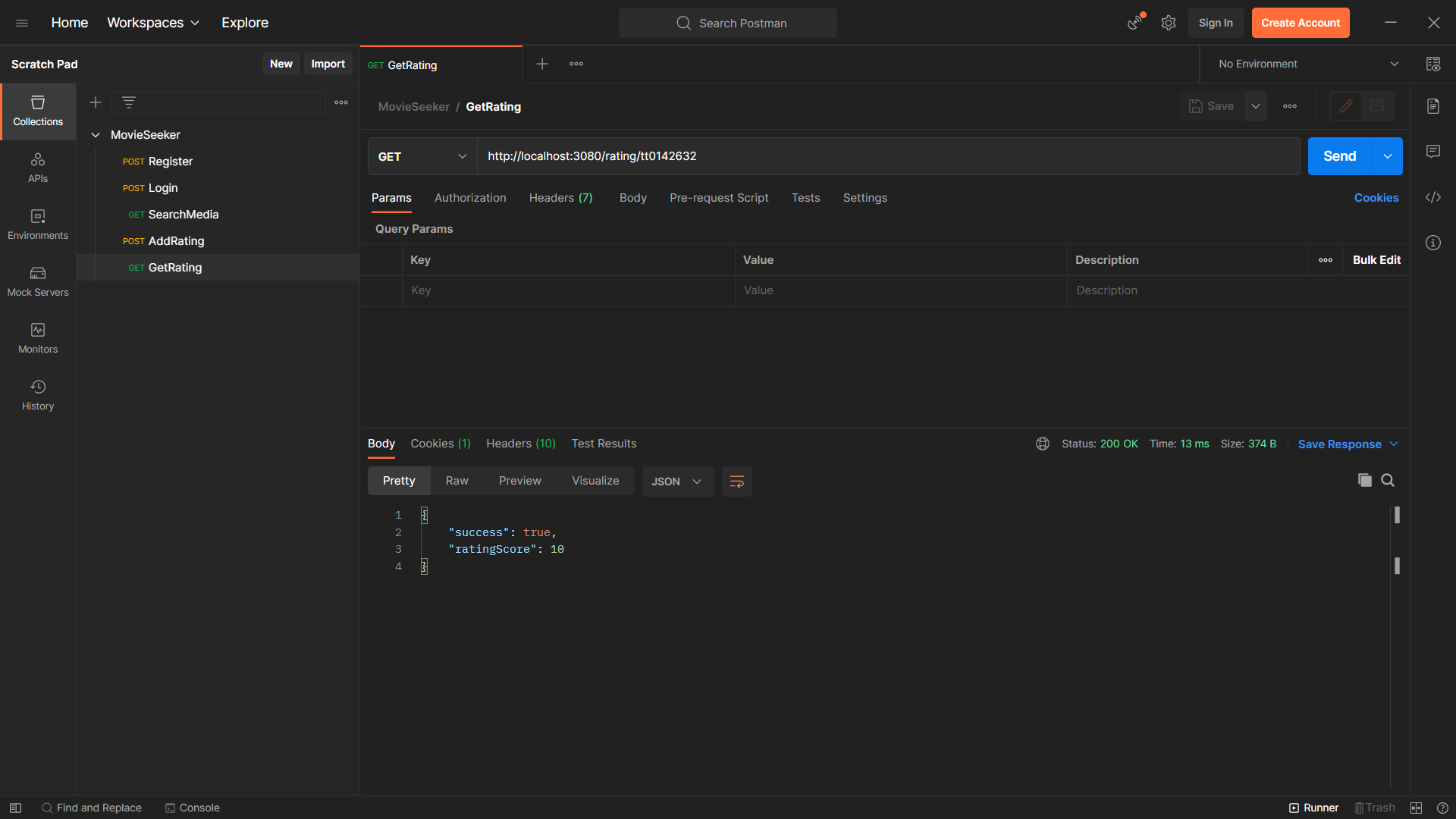
Beléptetés tesztelése Postman-ben



Médiára keresés tesztelése Postman-ben



Médiához való értékelés leadás tesztelése Postman-ben



Felhasználó által médiára leadott értékelés lekérdezésének tesztelése Postman-ben

# 10. Bevezetés, éles üzemmód

* Az alábbi weboldalról: <https://www.apachefriends.org/hu/index.html> töltsük le és telepítsük fel az operációs rendszerünknek megfelelő XAMPP webszerver-szoftvercsomag aktuális verziószámát.
* Sikeres telepítés után indítsuk el a XAMPP-ot, azon belül az Apache-t és a MySQL-t is indítsuk el. Ezután a MySQL Admin gombját válasszuk.
* A phpMyAdmin-ba importáljuk be az alábbi elérési útvonalon található adatbázist - backend/ src/ database/ movie\_seeker.sql
* Ezután töltsük le, és telepítsük fel a Node.js JavaScript futtató környezetet a <https://nodejs.org/weboldalról>.
* Utoljára a Visual Studio Code-ot kell letöltenünk és telepítenünk, mégpedig erről a weboldalról: <https://code.visualstudio.com/>
* Visual Studio Code-ban a Terminal megnyitásával és az " npm –i " paranccsal telepítsük fel az összes szükséges modult. Ezután az " npm start " paranccsal indítsuk el az alkalmazást, amit a <http://localhost:3000> címen tudunk elérni. Első indításnál az alkalmazás indítása pár percet vehet igénybe.

# Összegzés

Ez az általunk létrehozott Restful alkalmazás mind a négy CRUD műveletet kihasználva mutatja be kliens és a szerver oldal között fellépő kommunikációt.

A fejlesztés során fő szempont volt, hogy a különböző jogkörök jól láthatóan elkülönüljünk egymástól, ezért ennek megfelelően úgy alakítottuk ki a 4 felhasználói csoport lehetőségeit, hogy azok tükrözzék e szándékunkat.

Bár az alkalmazás már elkészült, további finomításokra és fejlesztésekre van szüksége annak érdekében, hogy jobban működjön. Ilyen fejlesztések lehetnék például:

* Regisztráció megerősítő e-mail küldése
* Új jelszó igénylése
* Felhasználók profilképének megjelenítése kommentszekcióban
* Előfizetők kiemelése kommentszekcióban
* Jelvényrendszer bevezetése az aktivitás jutalmazásért
* Bankkártyás fizetés megvalósítása

Az alkalmazás Git repository-ja:

<https://github.com/Blade1201/MovieSeeker>