Web technológiák 2.

Movies&Series webalkalmzás

Szkárosi Szilárd

DLWGQZ

Dátum: 2024. 02. 27.

Gyakorlatvezető: Agárdi Anita

Képzés: Mérnökinformatikus BSc nappali

# Tartalomjegyzék

[1. Tartalomjegyzék 1](#_Toc159675773)

[2. Feladat leírása 2](#_Toc159675774)

[3. Frontend 2](#_Toc159675775)

[3.1 Komponensek 3](#_Toc159675776)

[3.1.1 Footer 5](#_Toc159675777)

[3.1.2 Header 7](#_Toc159675778)

[3.2 Oldalak 8](#_Toc159675779)

[3.2.1 Forget-password 8](#_Toc159675780)

[3.2.2 Home 11](#_Toc159675781)

[3.2.3 Login 12](#_Toc159675782)

[3.2.4 Movies 17](#_Toc159675783)

[3.2.5 Series 22](#_Toc159675784)

[3.2.6 Register 28](#_Toc159675785)

[3.2.7 Reset 31](#_Toc159675786)

[4. Backend 35](#_Toc159675787)

[4.1 Modellek 35](#_Toc159675788)

[4.2 Routes (Elérési útvonalak) 37](#_Toc159675789)

[4.3 Kontrollerek 39](#_Toc159675790)

[4.4 Server.js 47](#_Toc159675791)

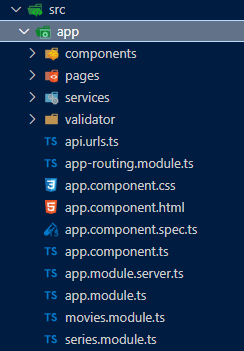
[4.5 Postman 48](#_Toc159675792)

[4.6 MongoDB 49](#_Toc159675793)

# Feladat leírása

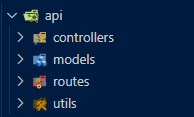
A beadandó feladatban filmeket és sorozatokat tároló webalkalmazást készítettem el. A frontend részét Angular keretrendszerben készítettem el, a backendhez pedig JavaScript-et és MongoDB-t használtam.

Frontend legfontosabb szerkezete:



Az oldalak stílusához a TailWind CSS-t használtam. Az alábbi linken tekinthető meg a telepítéshez szükséges lépések: [https://tailwindcss.com/docs/installation](%20https:/tailwindcss.com/docs/installation)

Backend legfontosabb szerkezete:

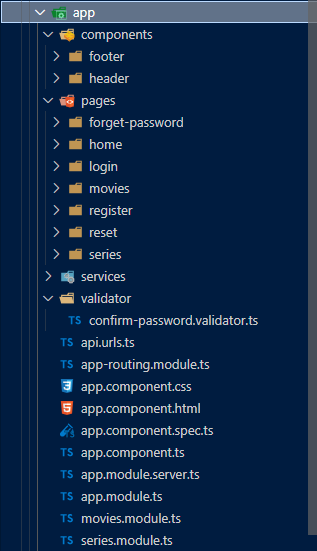


A http kérések helyes műkődésének leellenőrzésére a Postman-t használtam.

Az adatbáziskezeléshez a MongoDB-t használtam.

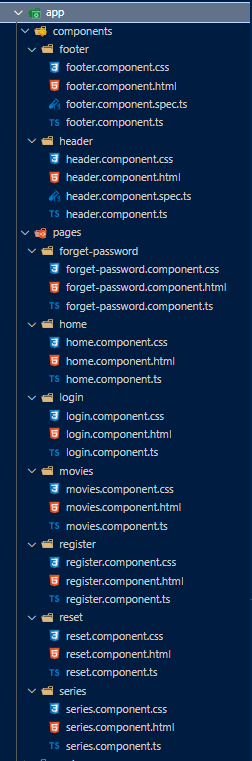
# Frontend

Az előző fejezetben említettem, hogy a frontend elkészítéséhez az Angular keretrendszert használtam. Az oldalak komponensekből, modulokból és szervízből áll.

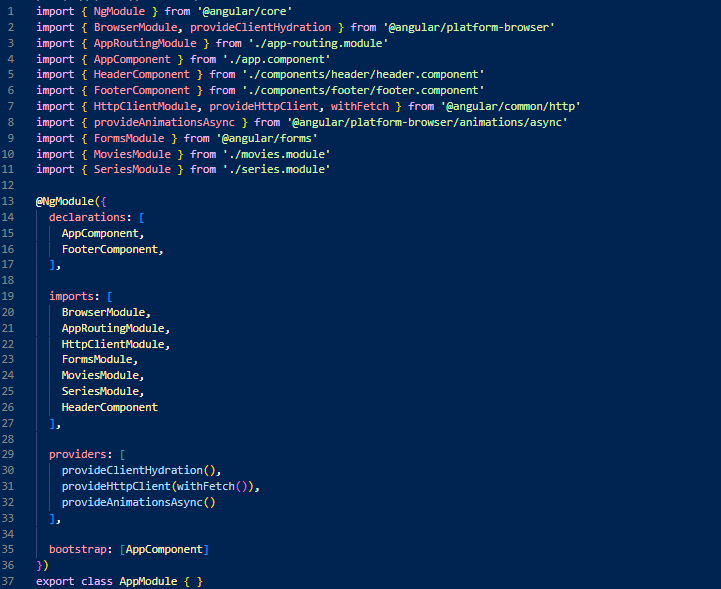


## Komponensek

Az oldalak elkészítéséhez komponenseket alkalmaztam és külön mappákba szerveztem őket, hogy átlátható legyen a mappaszerkezet. A pages mappában is komponensek vannak. A components mappában a HTML tag-ben szereplő kulcsszavak szerepelnek (pl.: footer és header). A komponensekben TypeScript, CSS és HTML kódok szerepelnek. A megfelelő működéshez az app.module.ts fájlban kell deklarálni.

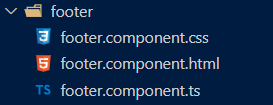


app.module.ts:



### Footer

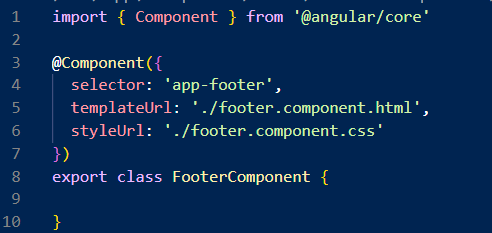
Az oldalak legalján van a footer, ahol az alkotó nevét és további információt tartalmazza.



footer.component.html:

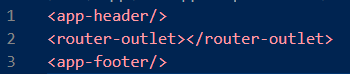


footer.component.ts:



Ahhoz, hogy a footer az oldalon is látható legyen a selector-ban szereplő ’app-footer’ kifejezést kell megadni az app.component.html-ben.

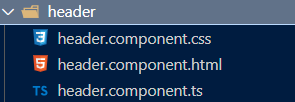
app.component.html:



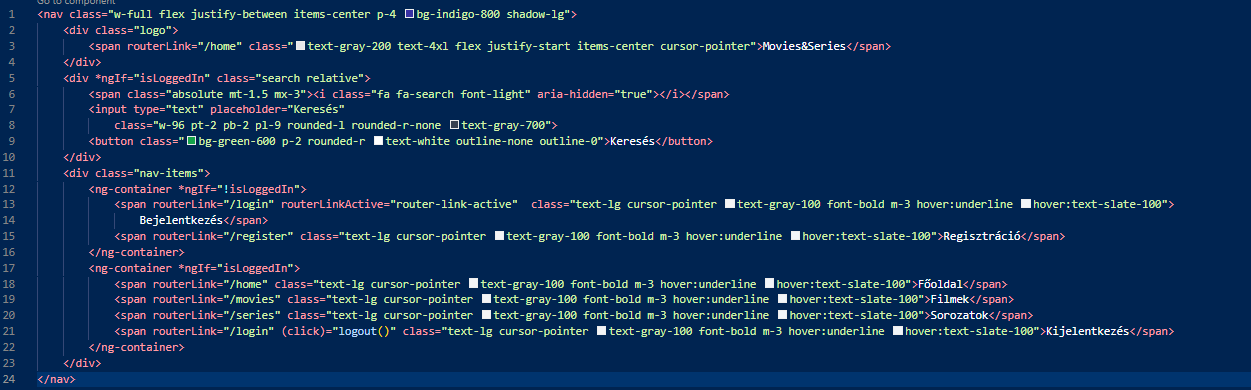


### Header

Az oldalak legtetején található a header, ahol a cím és a menüpontok találhatóak.



header.component.html:

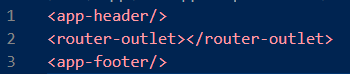


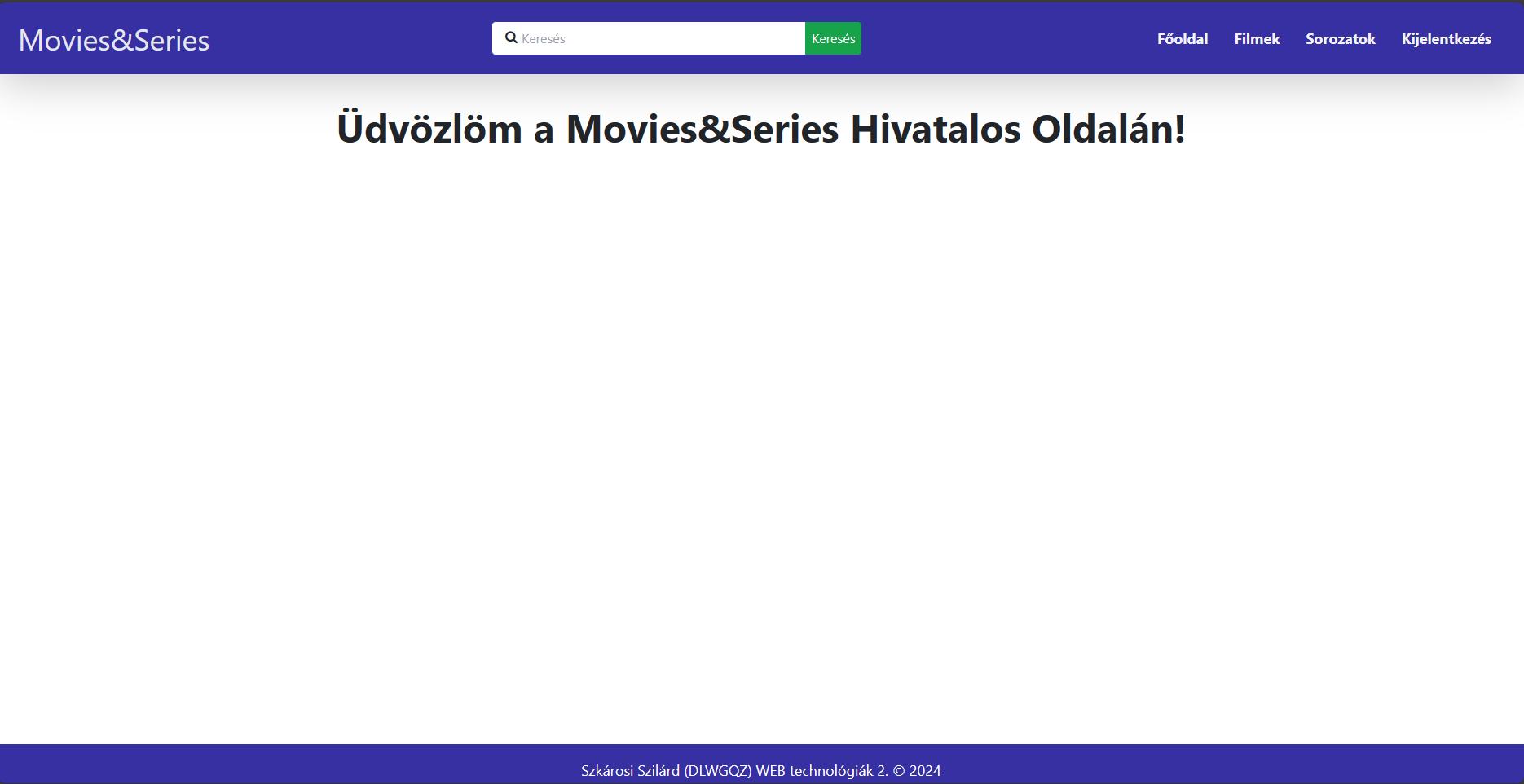
header.component.ts:



Ahhoz, hogy a header az oldalon is látható legyen a selector-ban szereplő ’app-header’ kifejezést kell megadni az app.component.html-ben.

app.component.html:





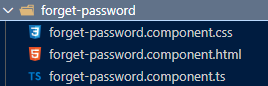
## Oldalak

Külön mappába szerveztem azokat a komponenseket, ahol fontosabb utasításokat lehet elvégezni (pl.: filmek és sorozatok létrehozása, módosítása és törlése, illetve regisztráció, bejelentkezés és jelszó visszaállítása).

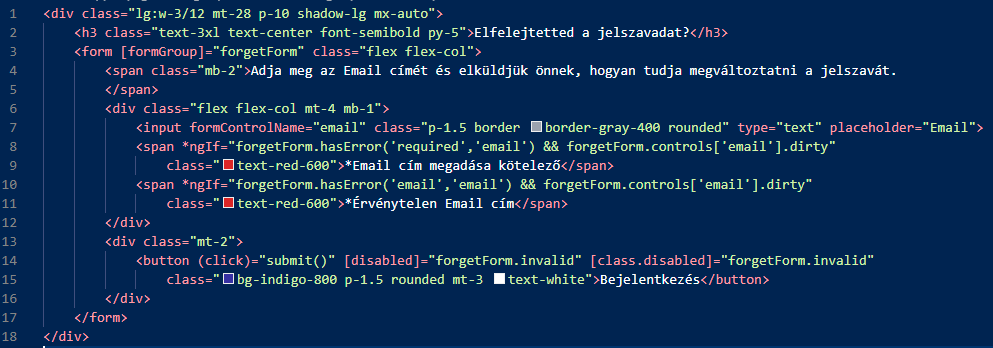


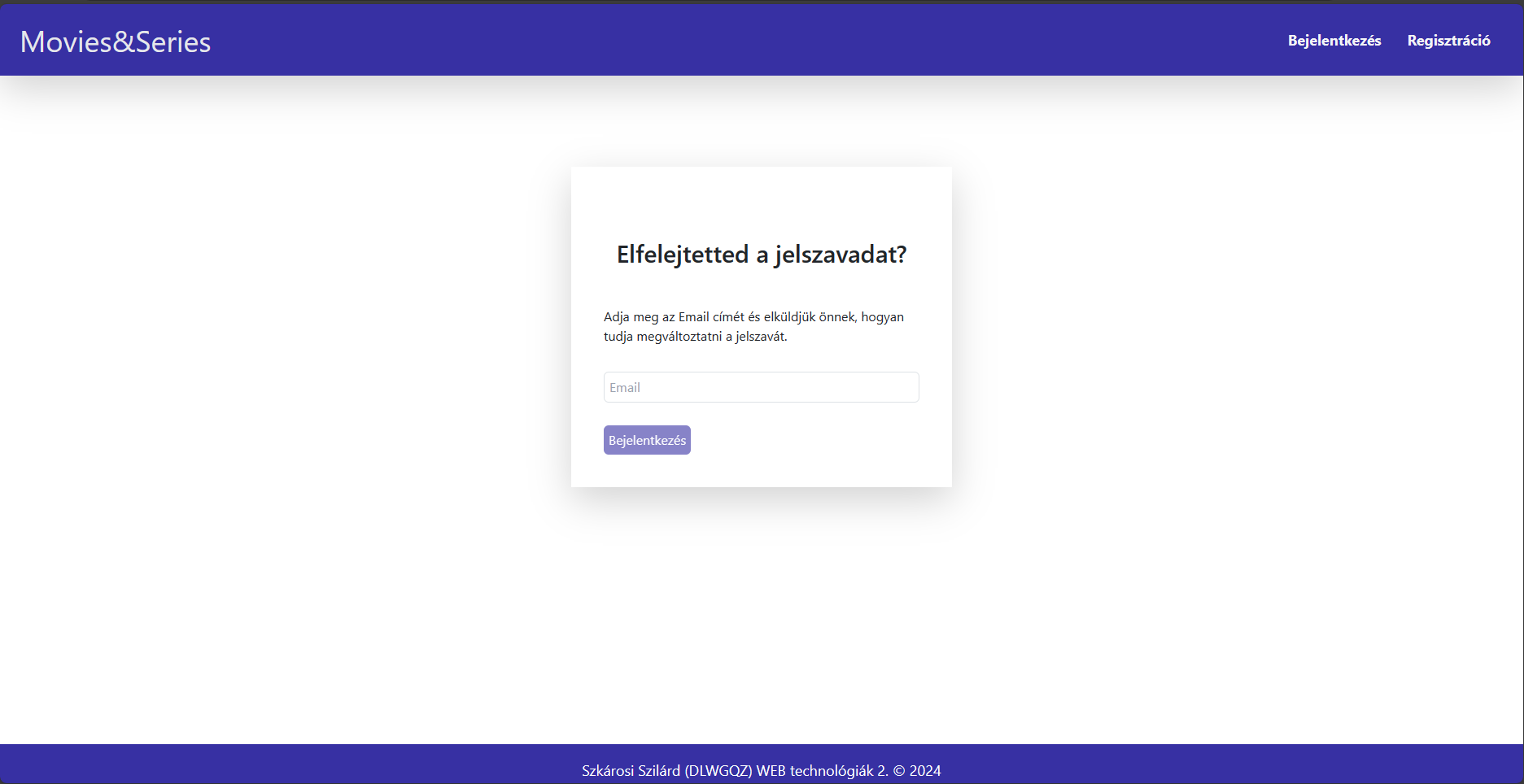
### Forget-password

Az oldal a jelszó visszaállításáért felelős. A végrehajtásához az email címet meg kell megadni.

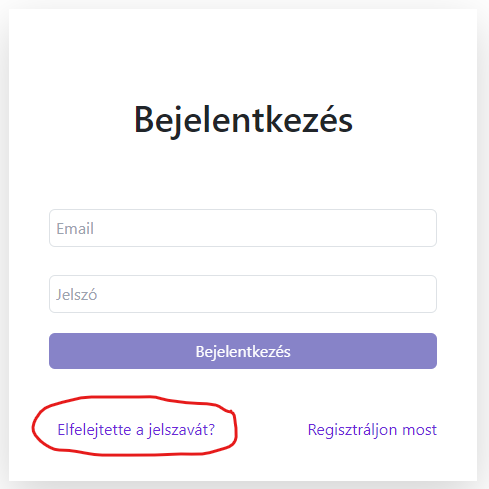


forget-password.component.html:

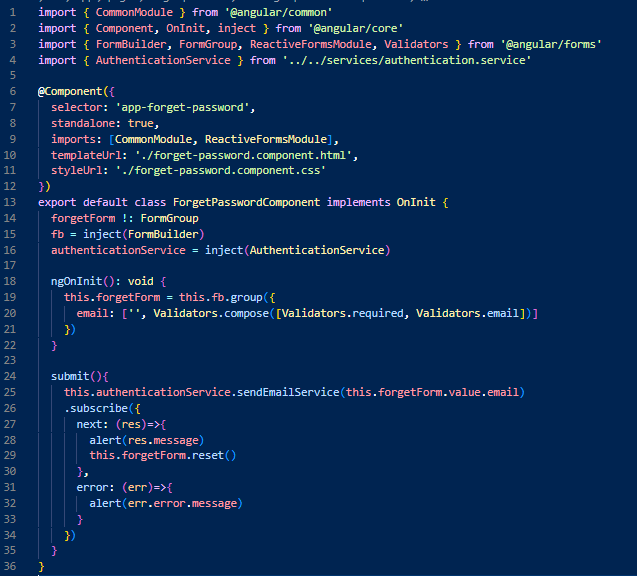




Az oldal eléréséhez a bejelentkezésnél rá kell kattintani az „Elfelejtette a jelszavát” gombra.



forget-password.component.ts:

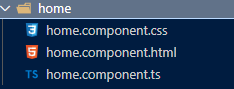


A komponens működéséhez meg kell adni az elérési útját az app-routing.module.ts-ben:



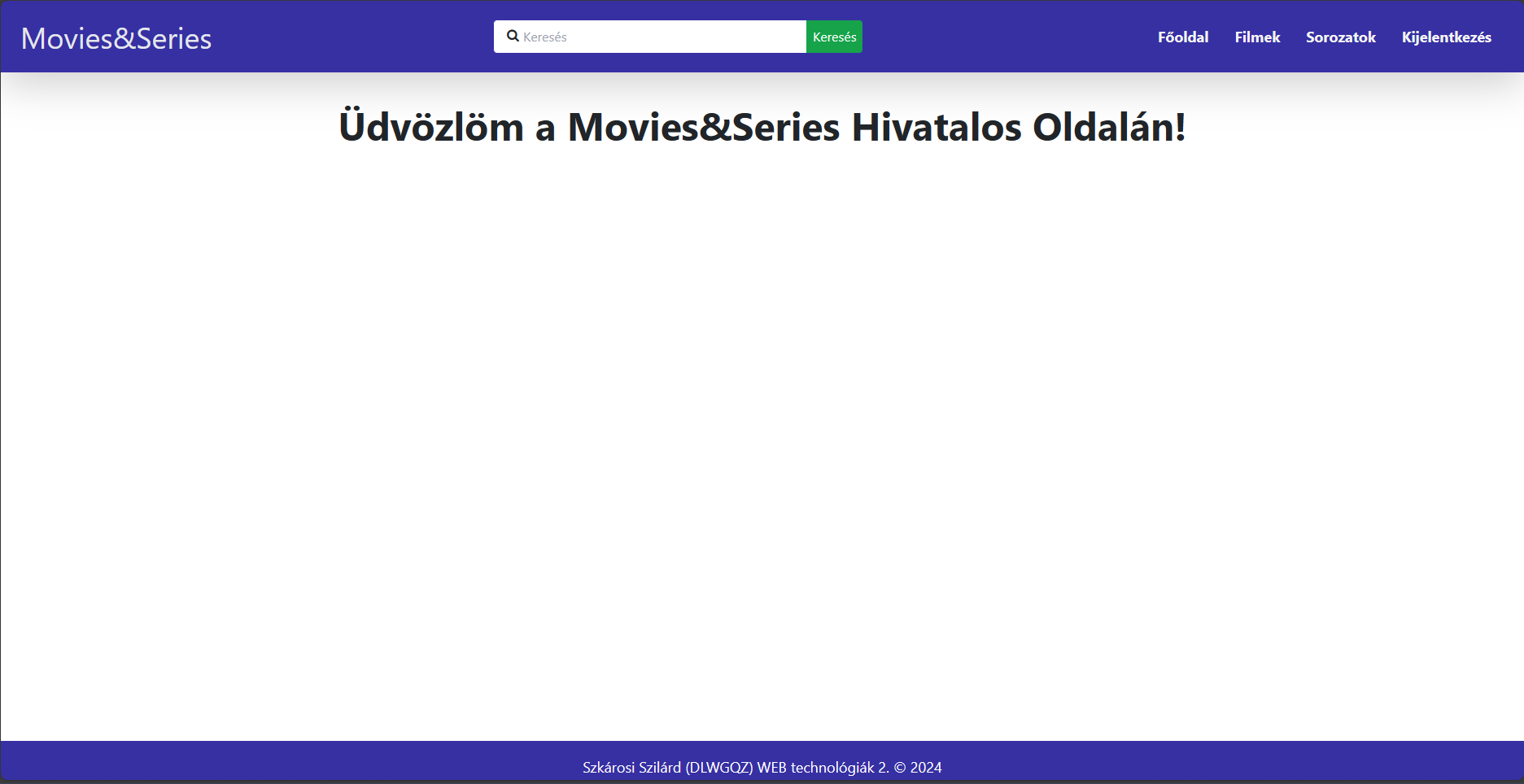
### Home

Ezen az oldalon csak egy egyszerű üdvözlő szöveg található.



home.component.html:





Amikor a felhasználó bejelentkezik az oldalra, akkor egyből a home oldalon találja magát.

home.component.ts:

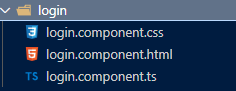


A komponens működéséhez meg kell adni az elérési útját az app-routing.module.ts-ben:

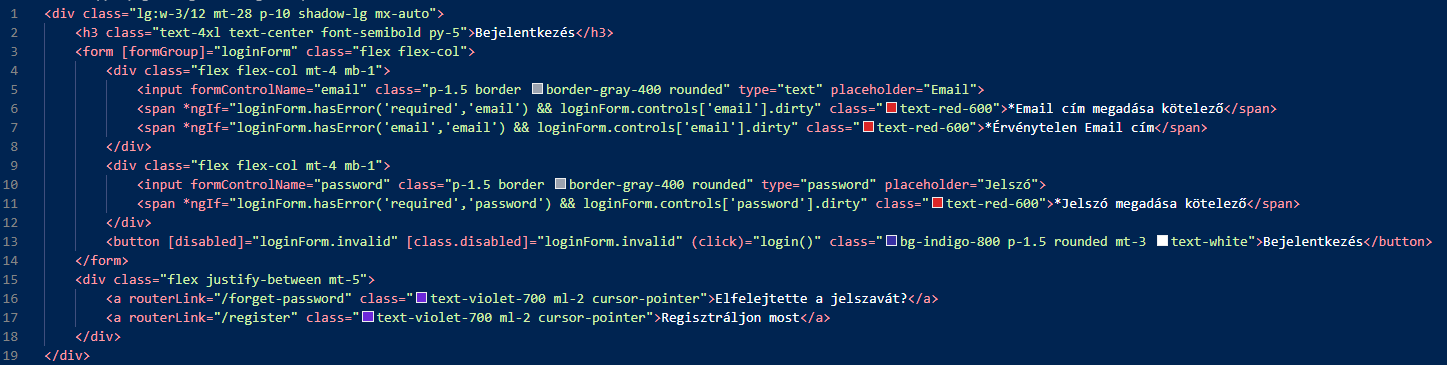


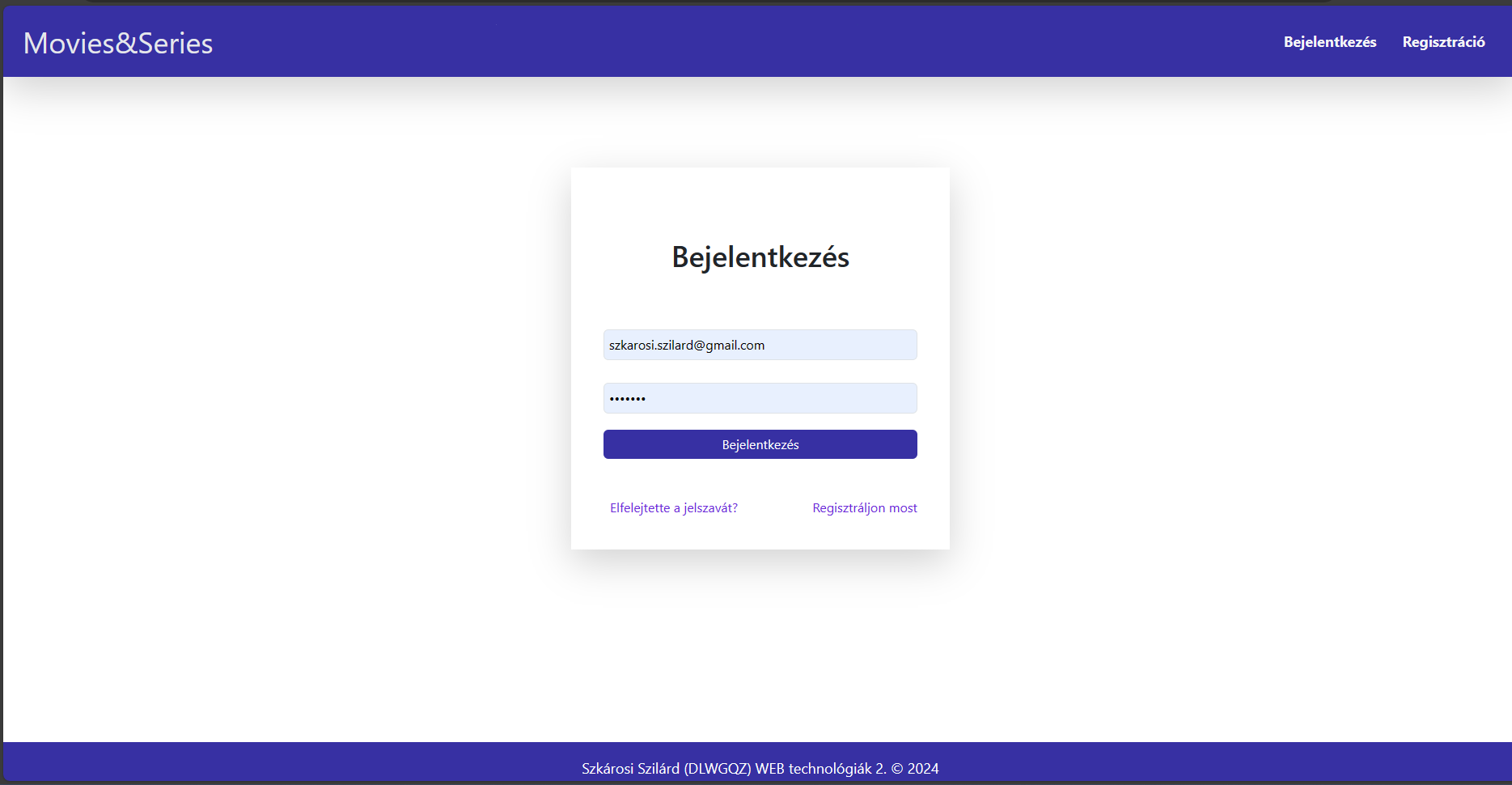
### Login

A felhasználó ezen az oldalon fog tudni bejelentkezni, ha már regisztrált.

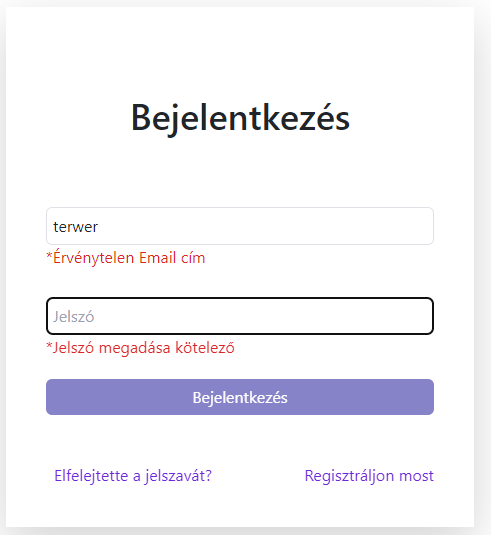


login.component.html:

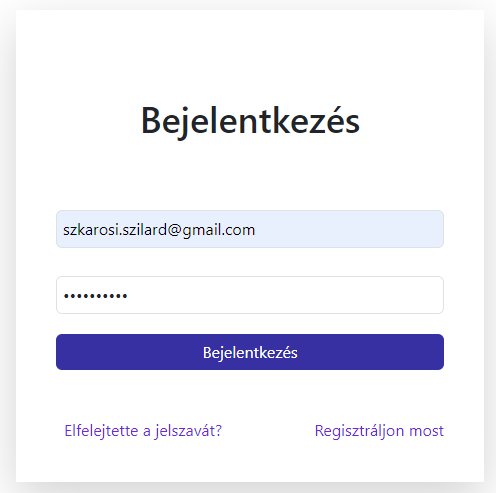




Ha nem adom meg az email cím és a jelszó, akkor megjelennek az üzenetek piros betűkkel:



Ha megadom az email címet és a jelszó, akkor eltűnnek az üzenetek és a Bejelentkezés gomb feloldódik.

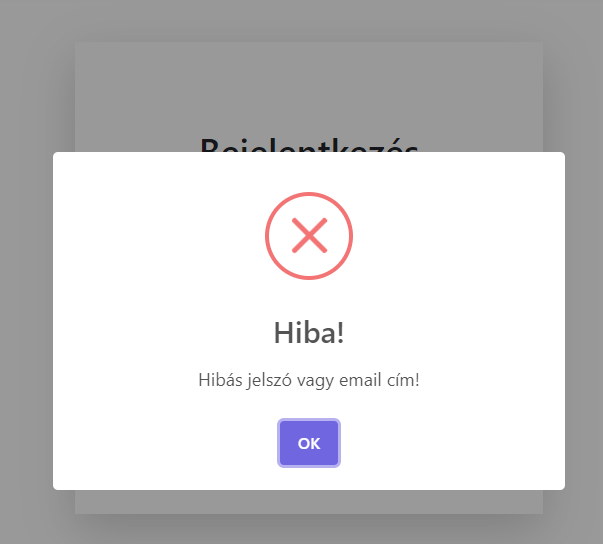


login.component.ts:





Ha nem sikerült a bejelentkezés, akkor az alábbi ablak jelenik meg:



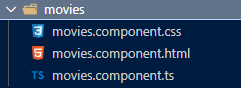
Ebben az esetben a felhasználó vagy az ’Elfelejtette a jelszavát?’ gombra kattint és megváltoztatja, vagy a ’Regisztrálj most’ gombra kattint.

Sikeres bejelentkezés eseten az alábbi ablak jelenik meg és elnavigál a főoldalra:

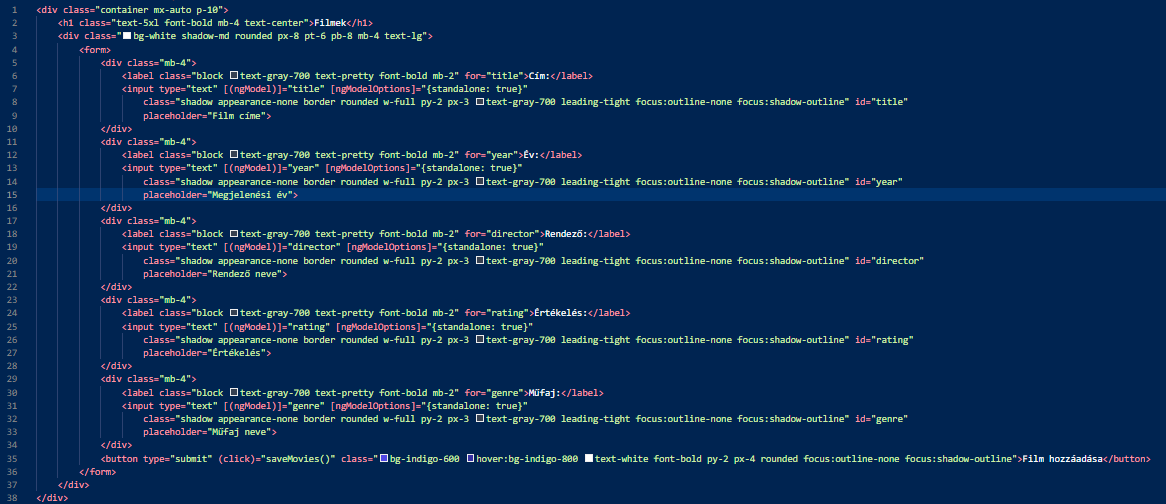


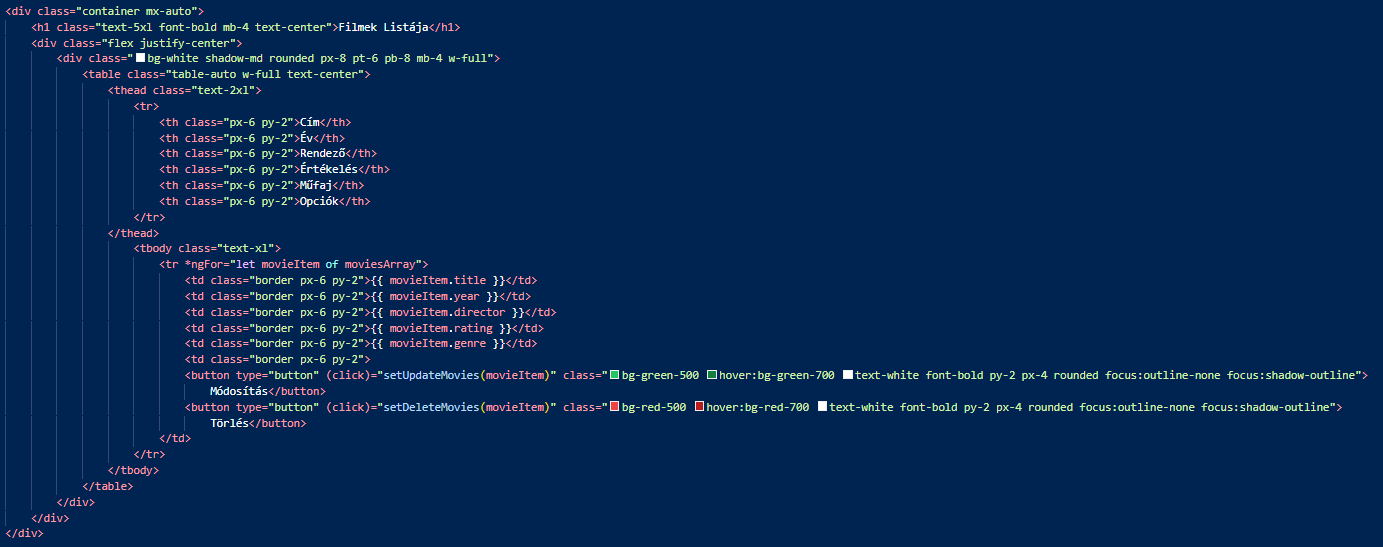
### Movies

Az oldalon a filmek adatai találhatóak táblázatba foglalva.



movies.component.html:





Itt lehet hozzáadni filmeket.



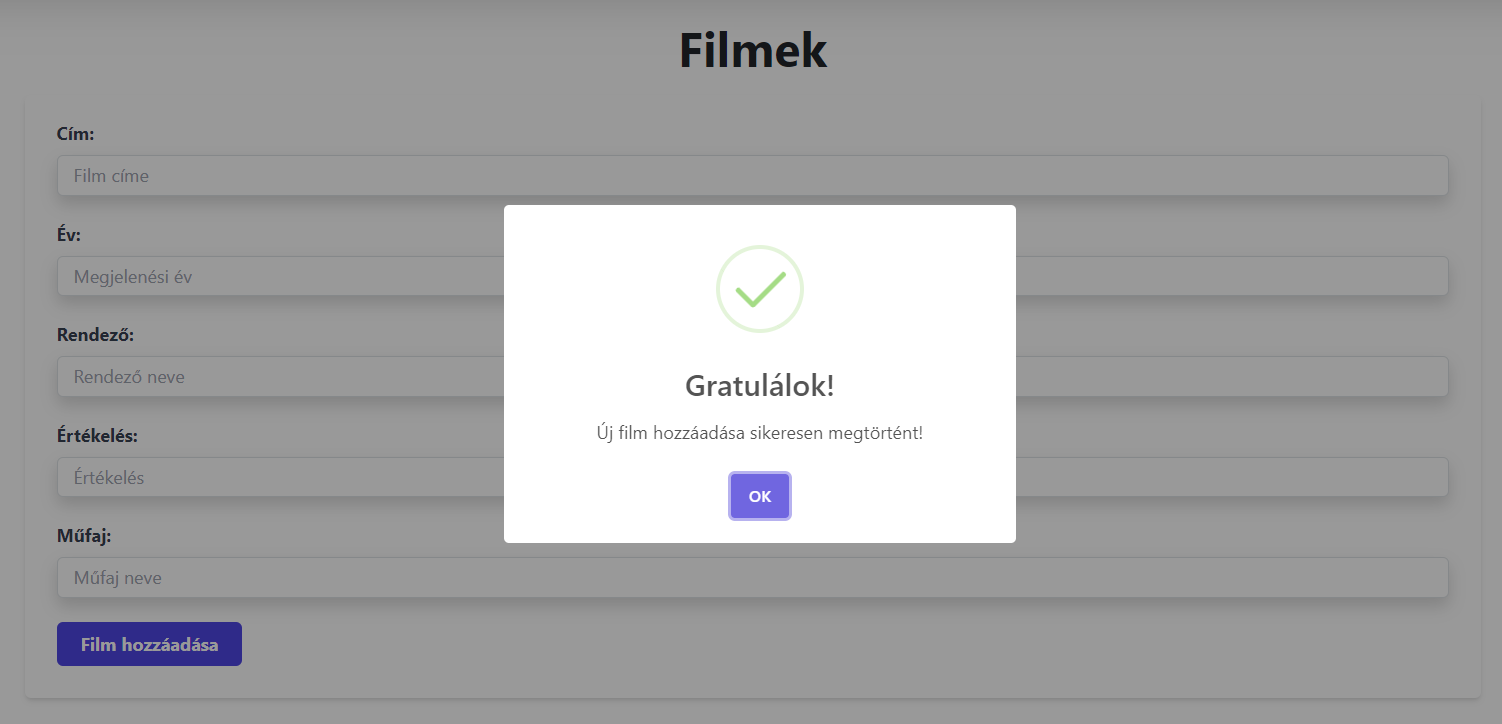
Az oldal legalján tekinthető meg a filmek listája:

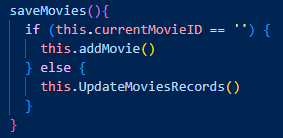


movies.component.ts:



Ha sikerült hozzáadni a filmet, akkor a felhasználót ez az üzenet fogadja:





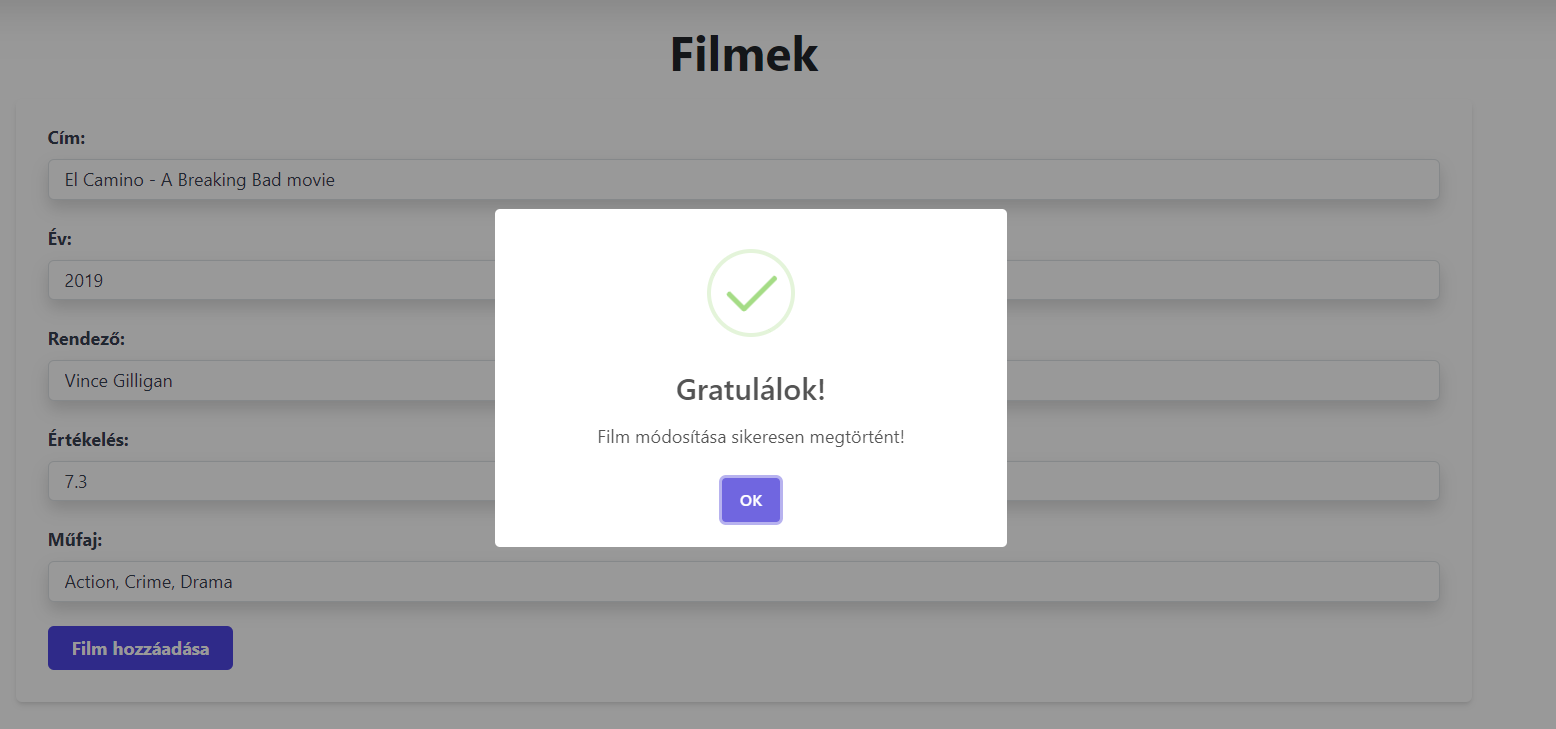
A táblázat legalján lesz mindig látható a hozzáadott film:



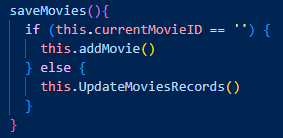
Adott filmet lehet módosítani és törölni.

Ha módosítani szeretnénk egy filmet, akkor a ’Módosítás’ gombra kattintva meg lehet csinálni:



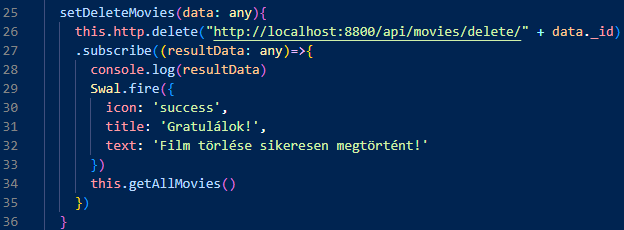


Módosítás után látható is az eredmény:





Ha a táblázat adott sorából szeretnénk törölni, akkor a ’Törlés’ gombra kell kattintani:



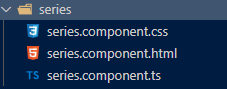


A sikeres törlés után látható, hogy a film már nincs a táblázatban:

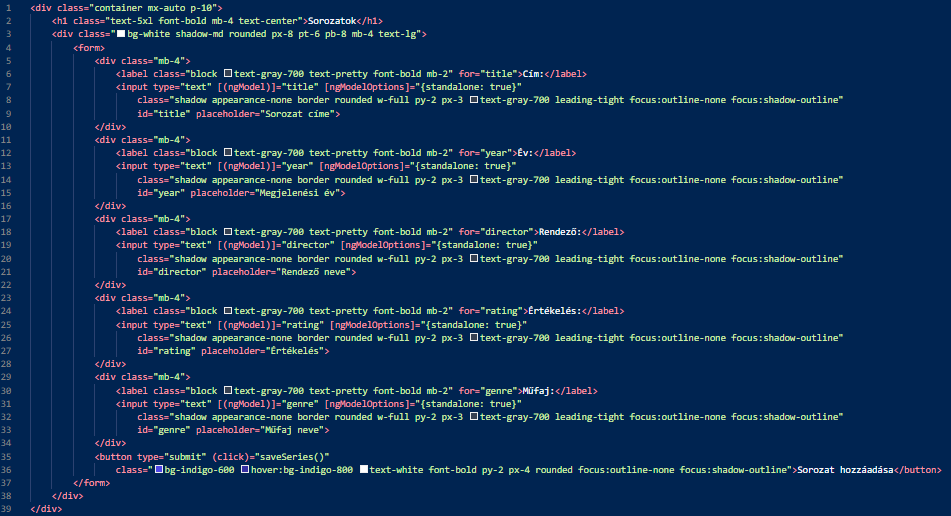


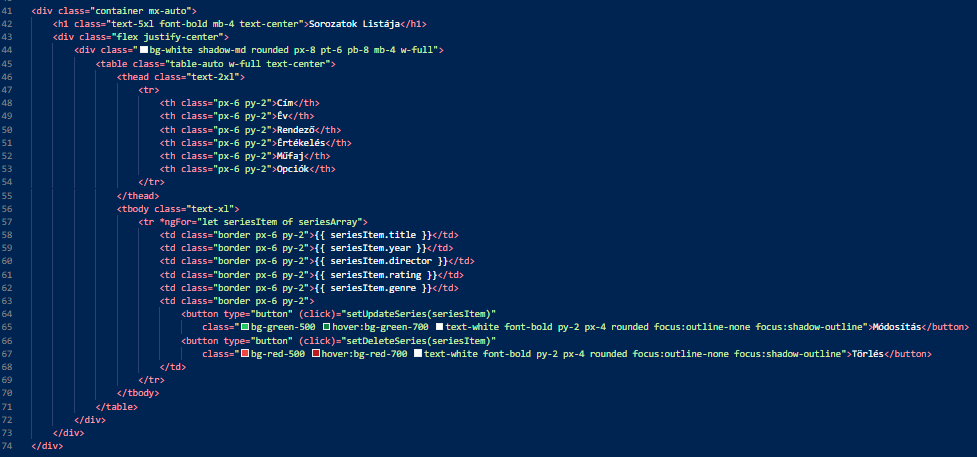
### Series

Az oldalon a sorozatok adatai találhatóak táblázatba foglalva.



series.component.html:





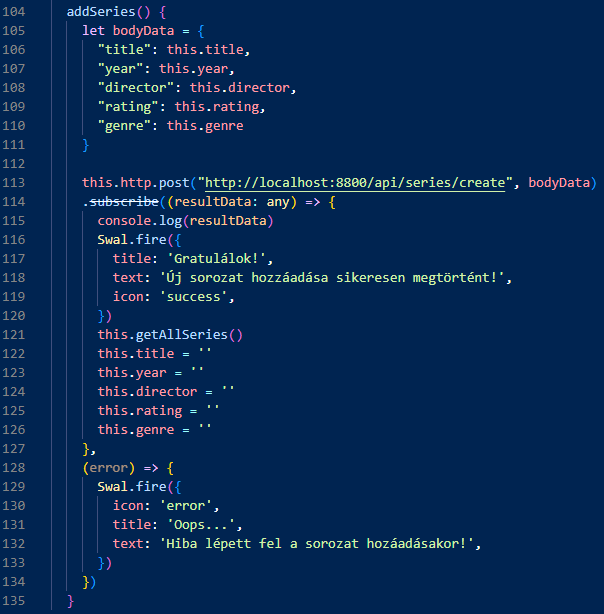
Itt lehet hozzáadni sorozatokat.



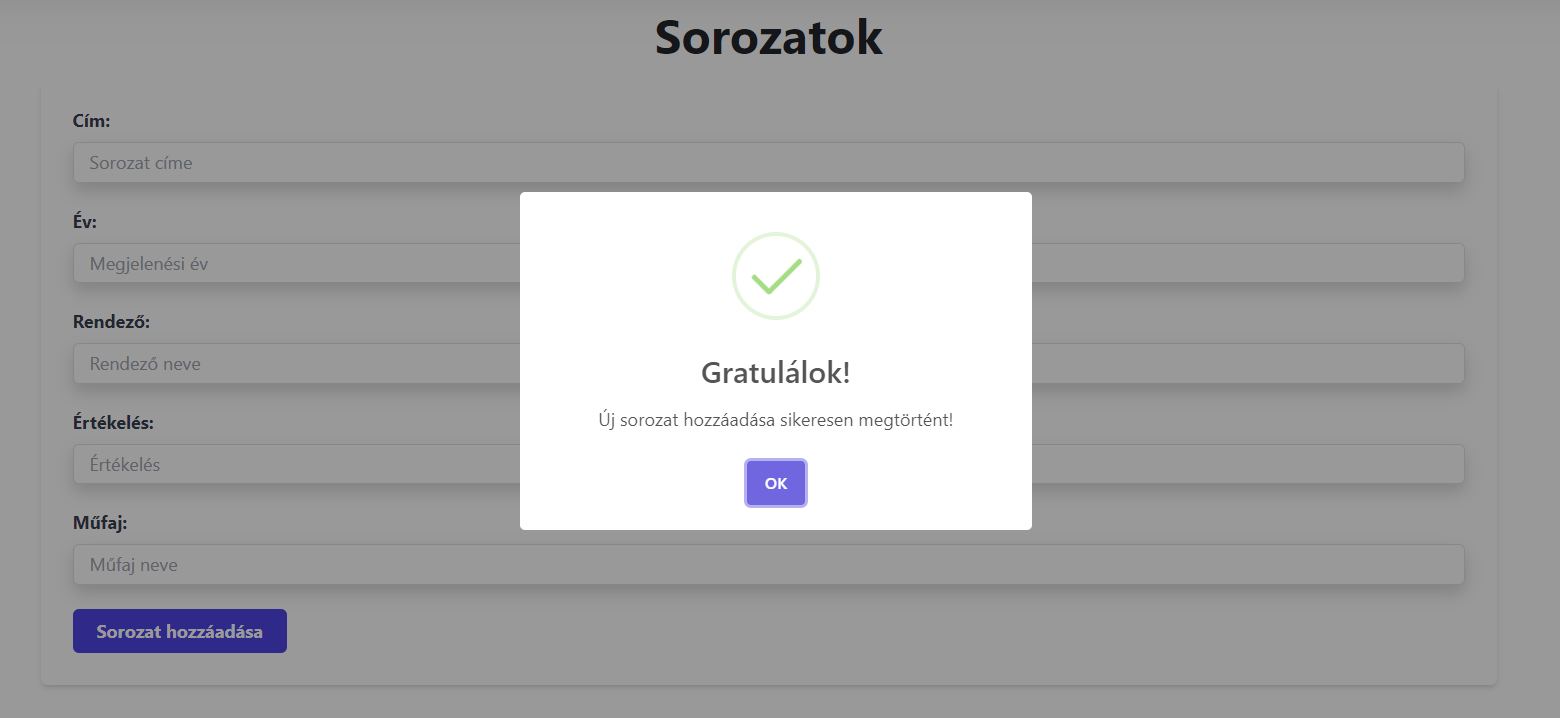
Az oldal legalján tekinthető meg a sorozatok listája:

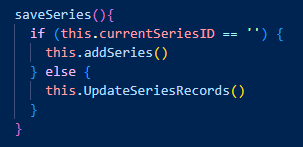


series.component.ts:



Ha sikerült hozzáadni a sorozatot, akkor a felhasználót ez az üzenet fogadja:





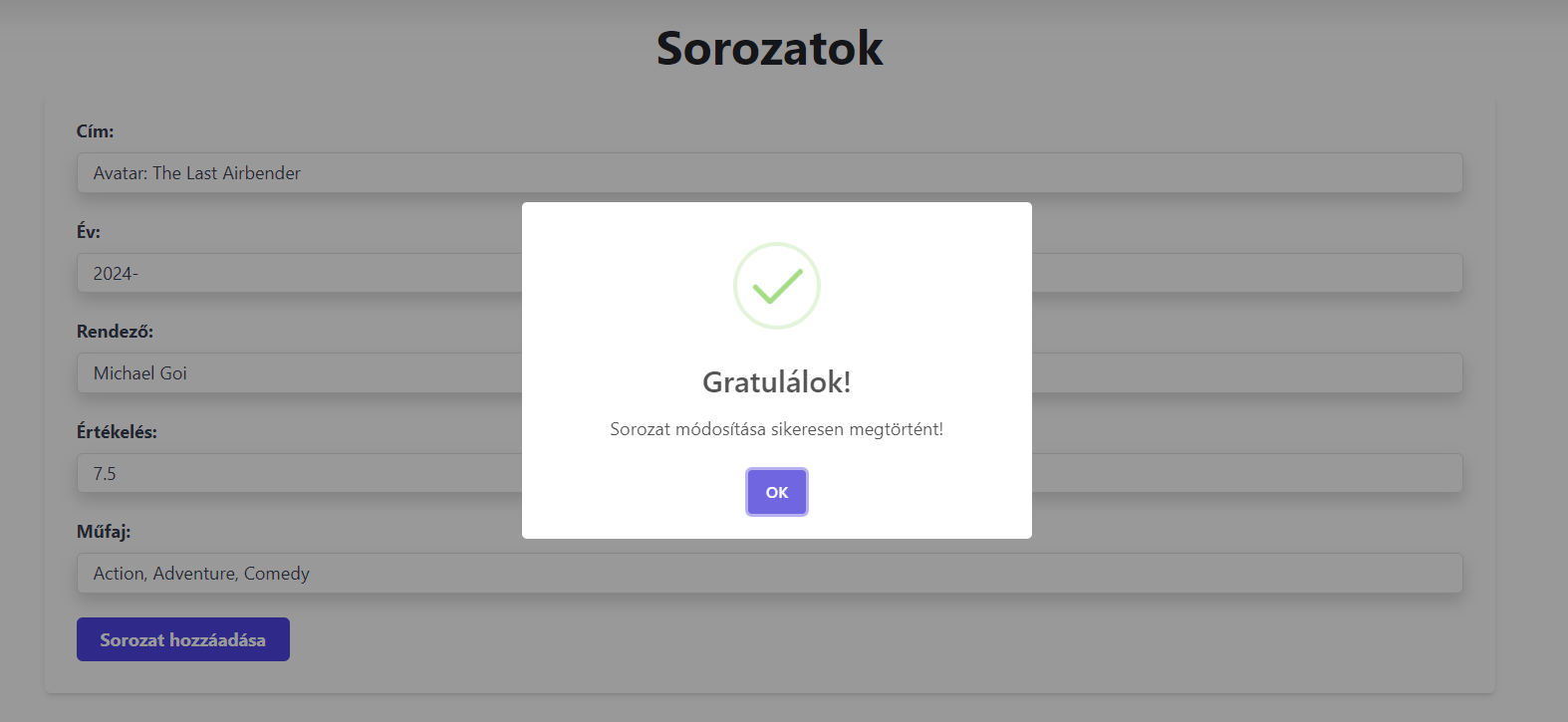
A táblázat legalján lesz mindig látható a hozzáadott sorozat:



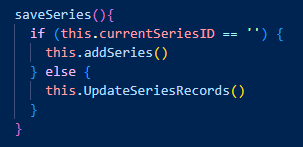
Adott sorozatot lehet módosítani és törölni.

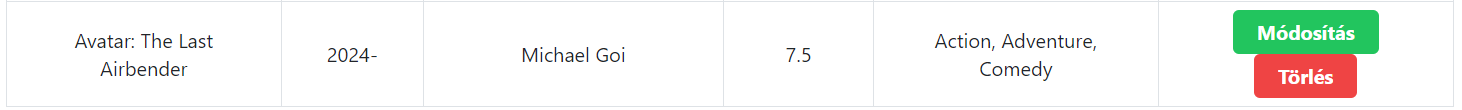
Ha módosítani szeretnénk egy sorozatot, akkor a ’Módosítás’ gombra kattintva meg lehet csinálni:



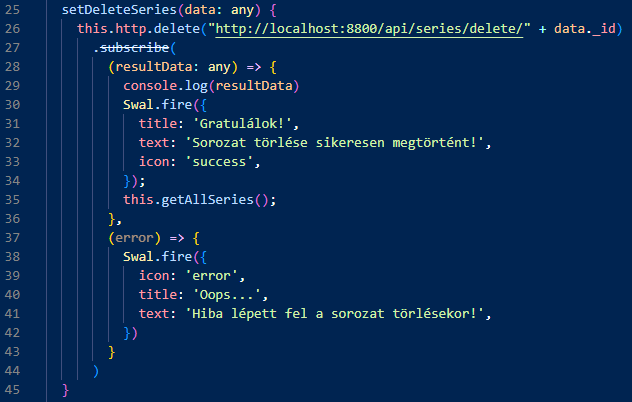


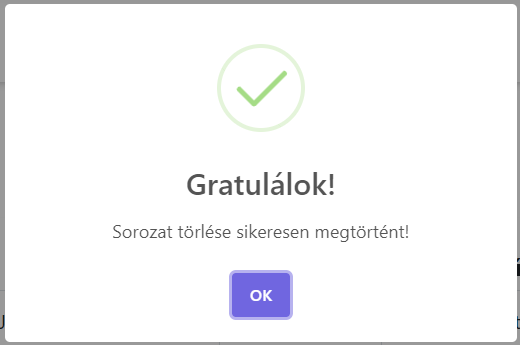
Módosítás után látható is az eredmény:





Ha a táblázat adott sorából szeretnénk törölni, akkor a ’Törlés’ gombra kell kattintani:



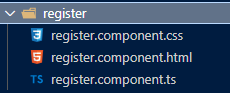


A sikeres törlés után látható, hogy a sorozat már nincs a táblázatban:

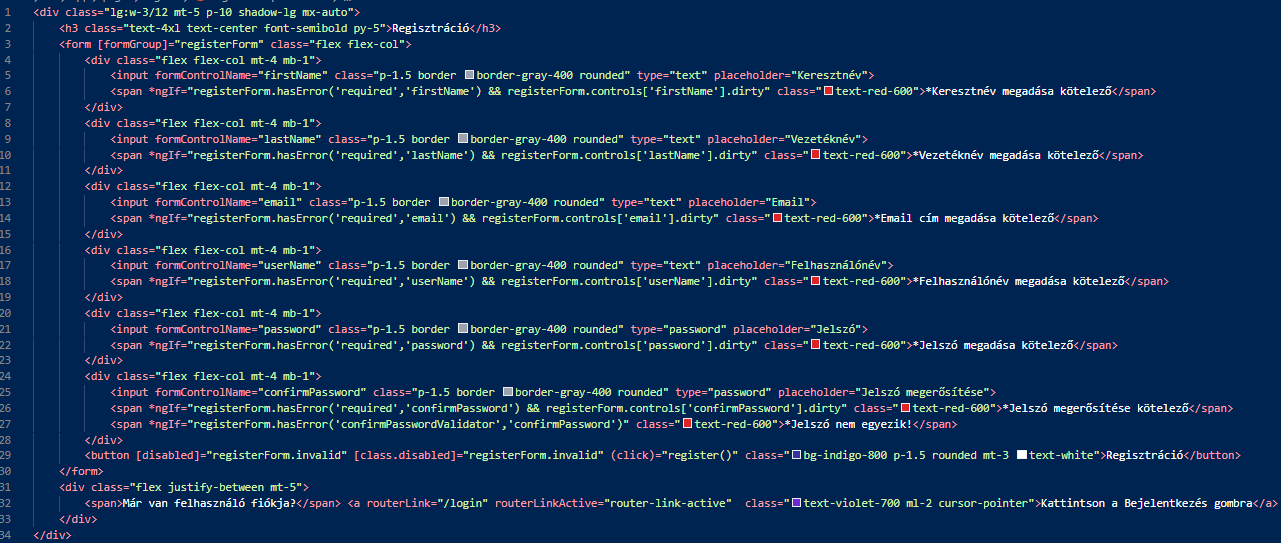


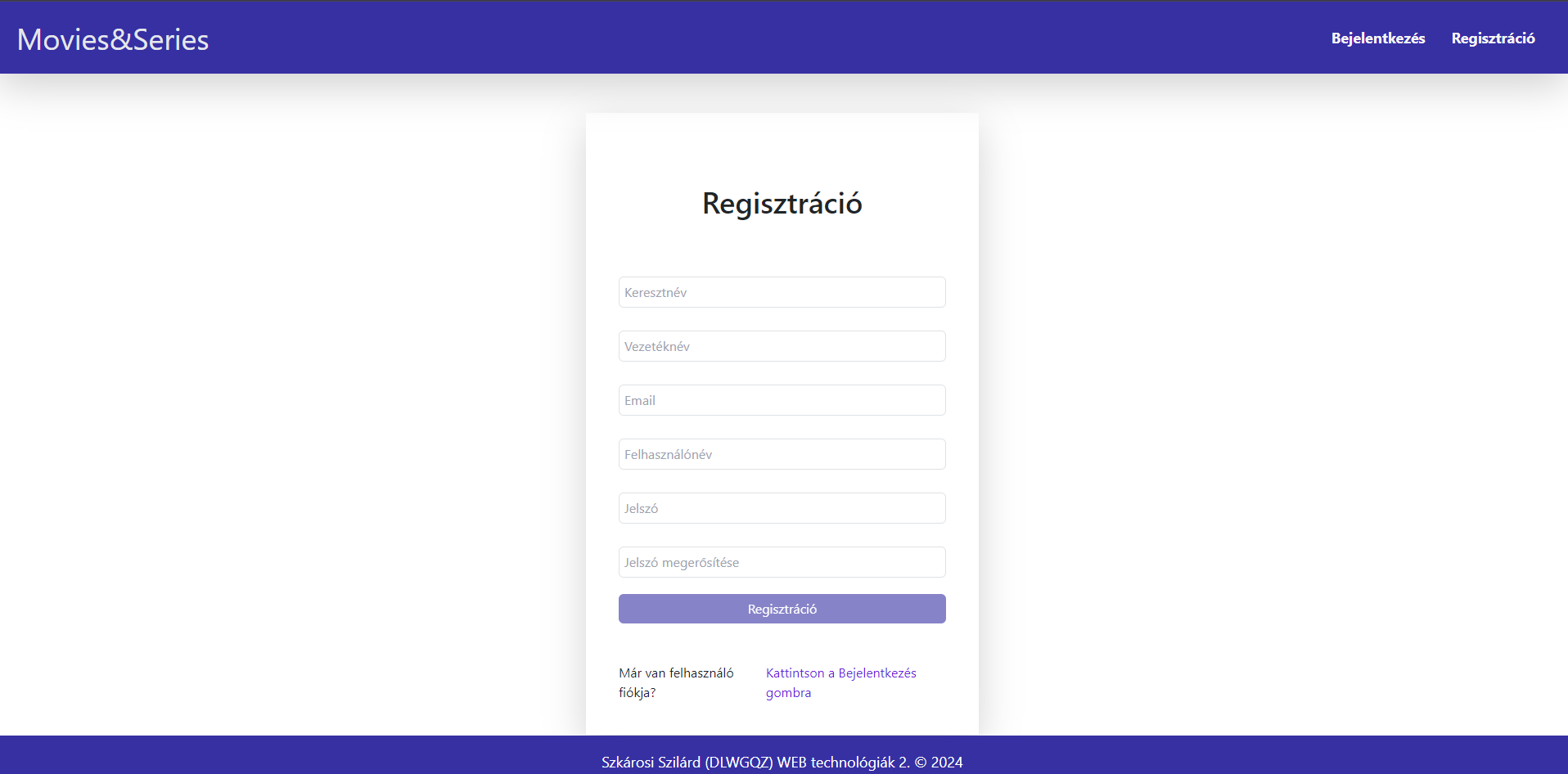
### Register

A felhasználó ezen az oldalon fog tudni regisztrálni.

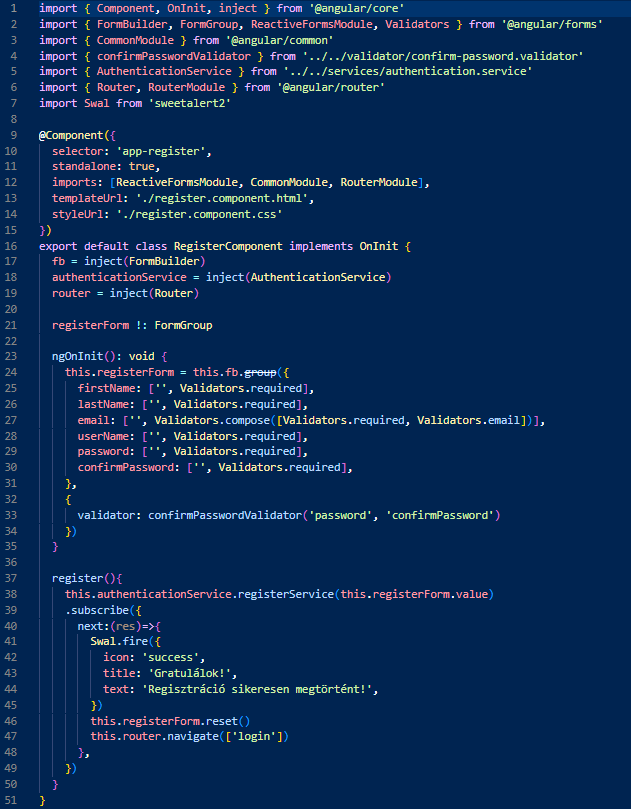


register.component.html:



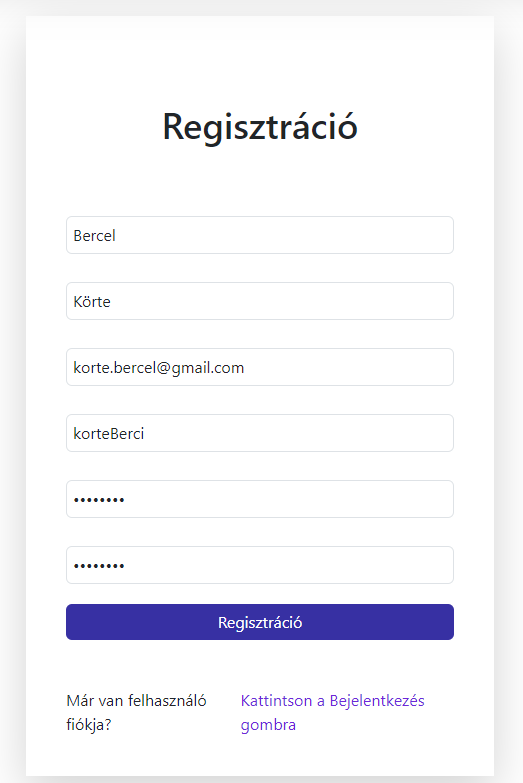


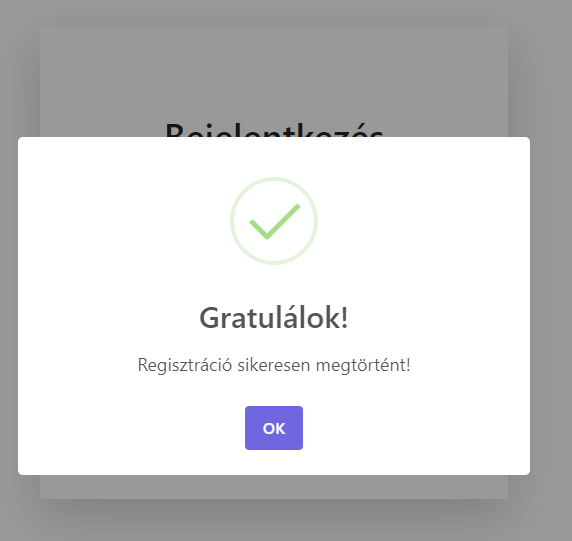
register.component.ts:



Regsztrációkor egyértelműen mindent meg kell adni.

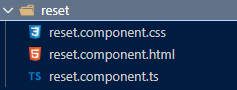
Ha minden adatot megadtunk, akkor feloldódik a ’Regisztráció’, majd megjelenik az üzenet:





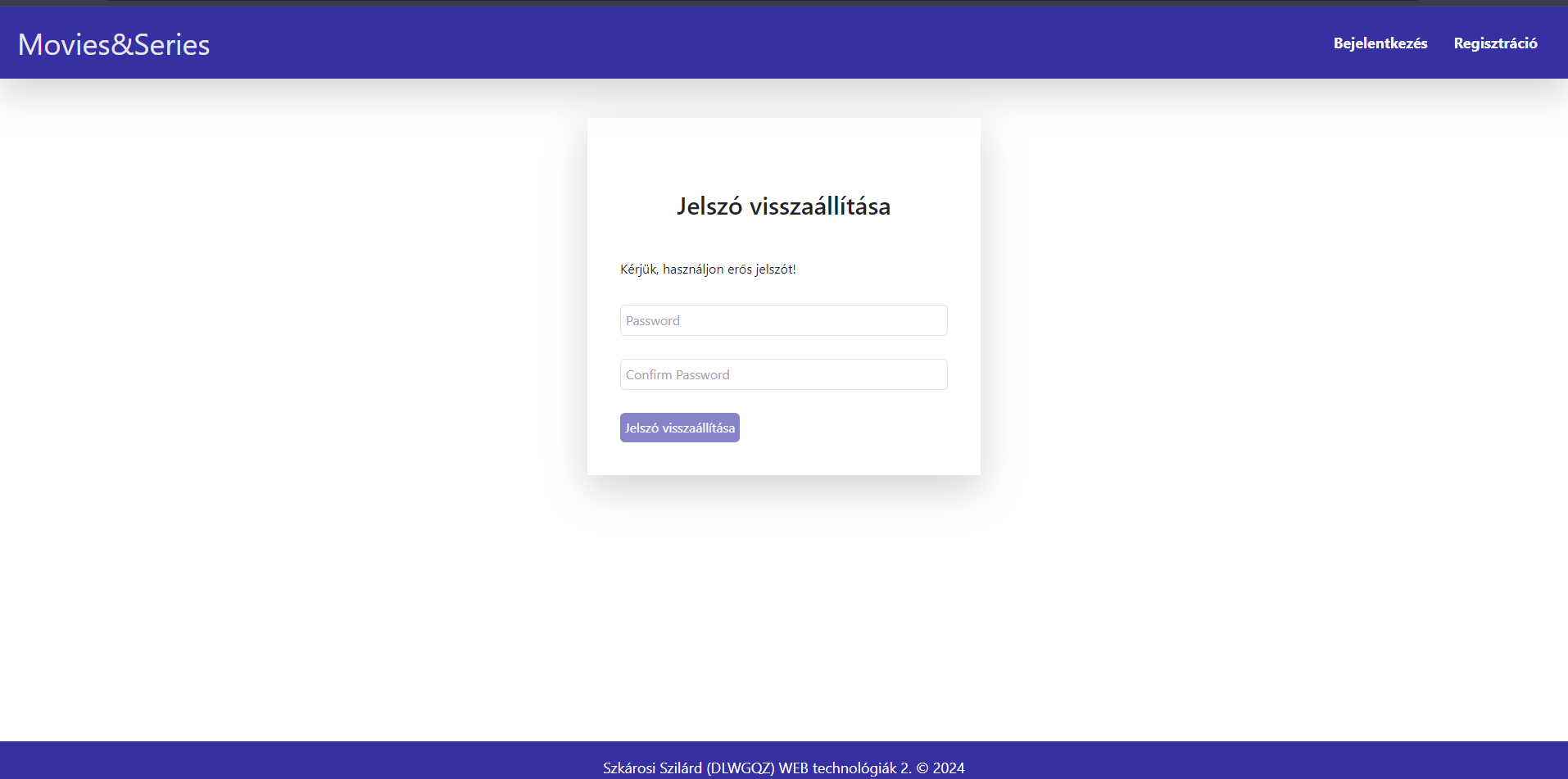
### Reset

A felhasználó vissza tudja állítani a jelszavát.

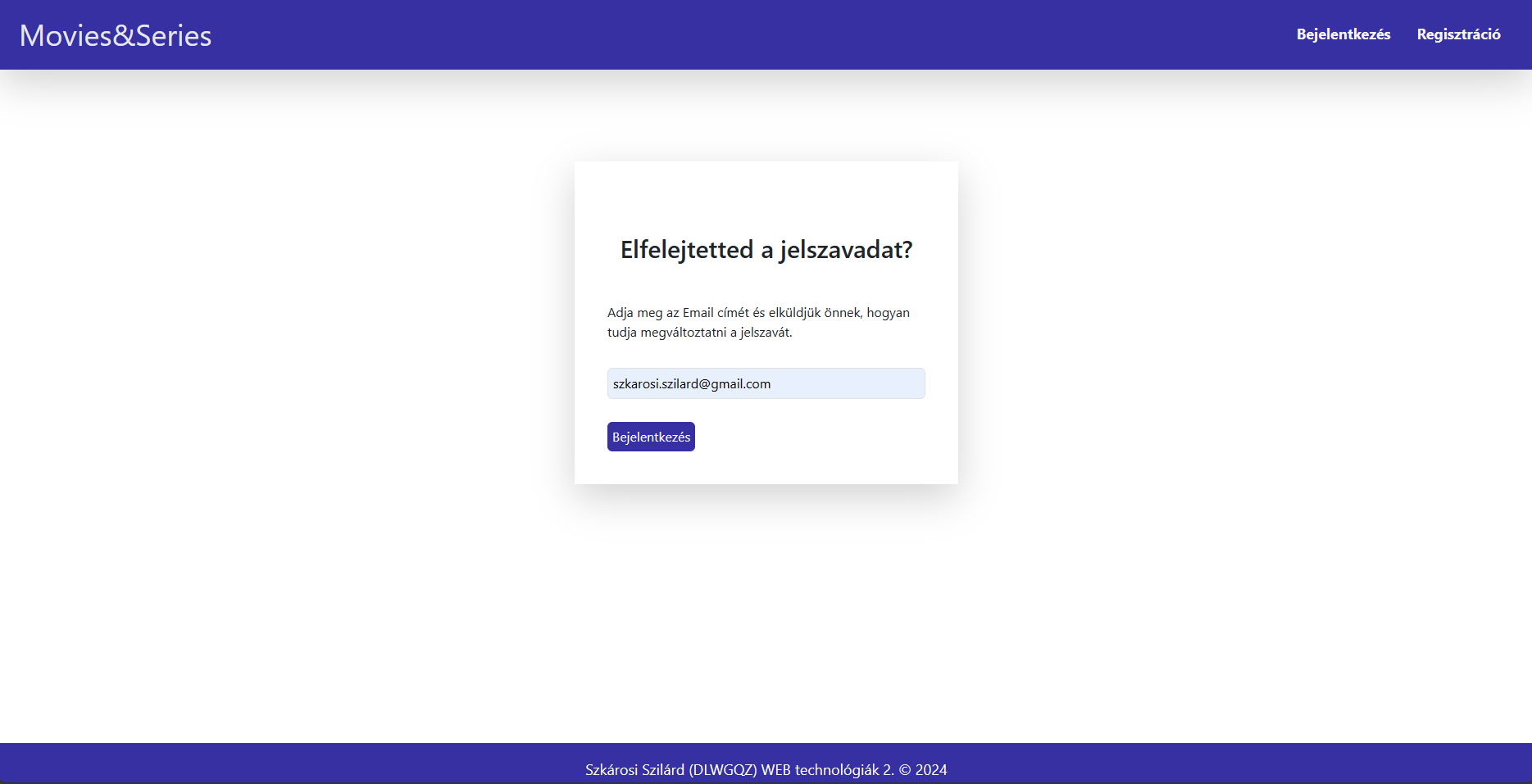


reset.component.html:

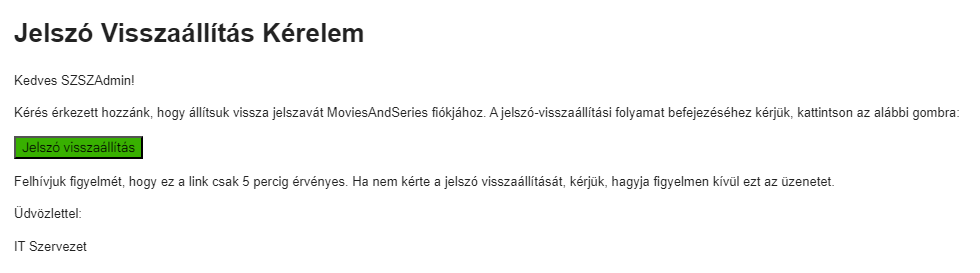




Az oldal akkor jelenik meg, hogyha a felhasználó a ’Bejelentkezés’ gombra kattint a forget-password oldalon.



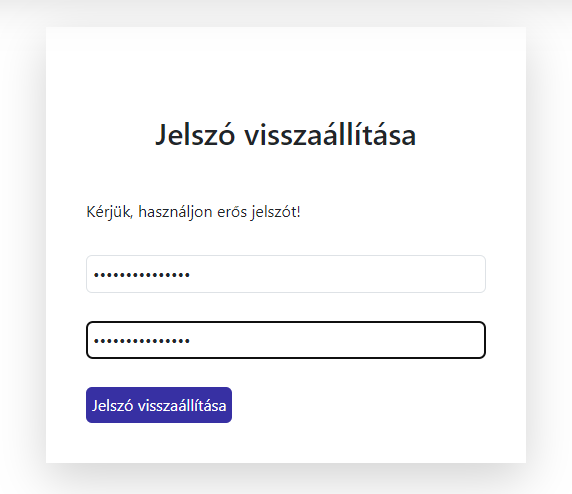
Utána a felhasználó kap egy emailt a kérelemről. Ebben a példában a saját email címemet adtam meg.



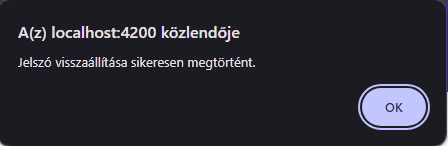
reset.component.ts:



A felhasználónak itt lesz lehetősége megváltoztatni a jelszavát.



Ha sikerült visszaállítani a jelszót, akkor az alábbi üzenet fogadja:

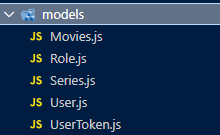


# Backend

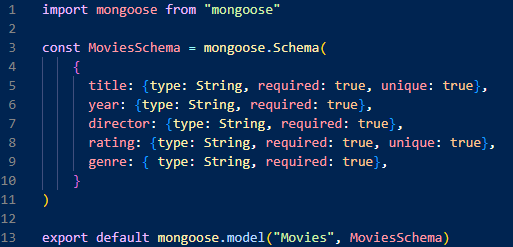
A webalkalmazás elkészítéséhez nagyon fontos a backend, mivel az biztosítja az adatok tárolását, CRUD (Creat, Read, Update, Delete) elv használatát az adatok feldolgozásához, illetve az autentikációt.

## Modellek

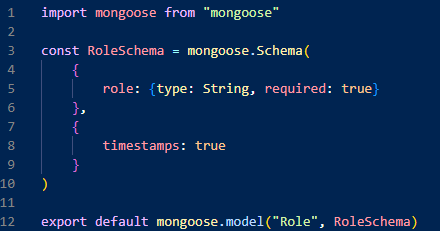
A modellek az adatok tárolására, kezelésére és feldolgozására szolgálnak.



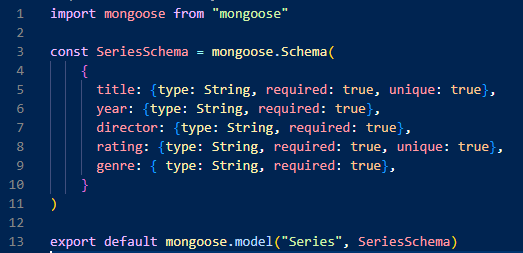
Movies.js:



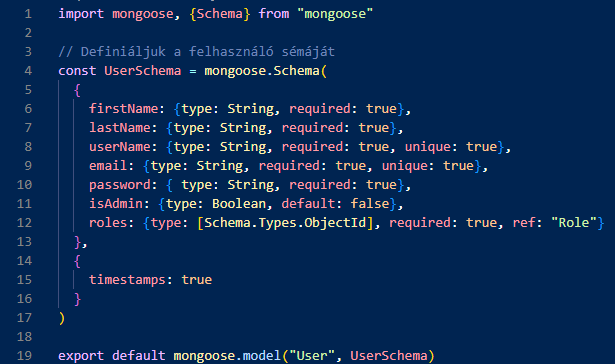
Role.js:



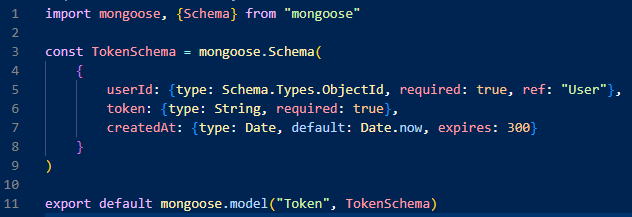
Series.js:



User.js:

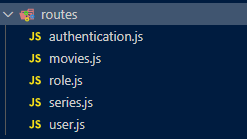


UserToken.js:

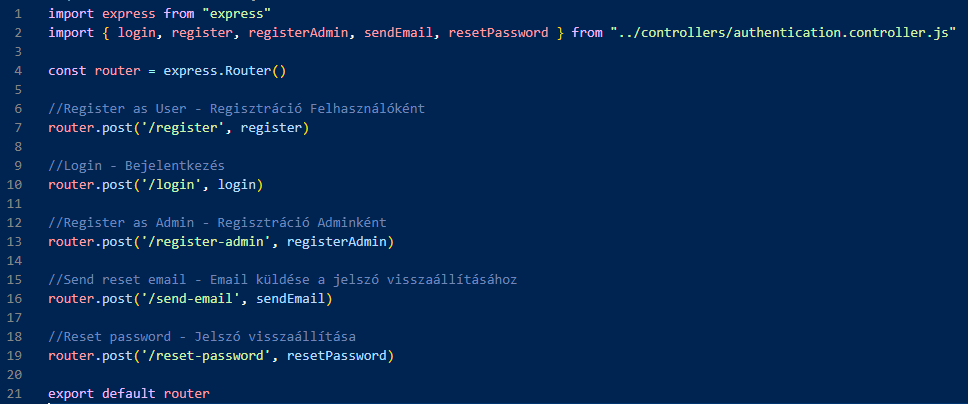


## Routes (Elérési útvonalak)

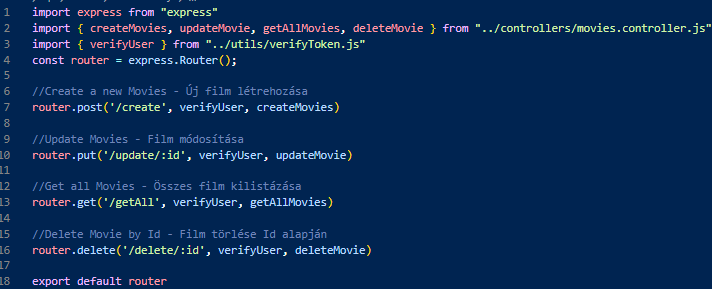
Az elérési útvonalak biztosítják a http kérés kezelését (pl.: GET, POST, DELETE, PUT) a szerveroldalon.



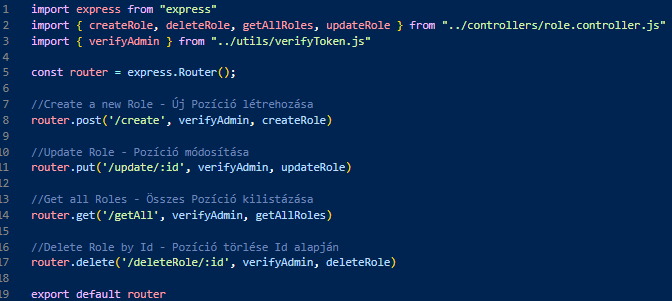
authentication.js:



movies.js:



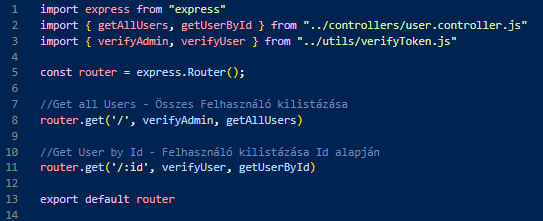
role.js:



series.js:



user.js:

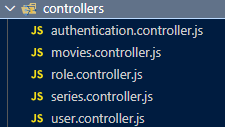


## Kontrollerek

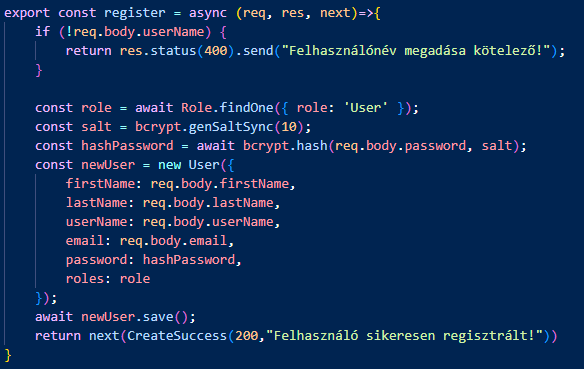
A kontrollerek olyan komponensek, amelyek az adatok feldolgozását és az alkalmazás logikáját kezelik.

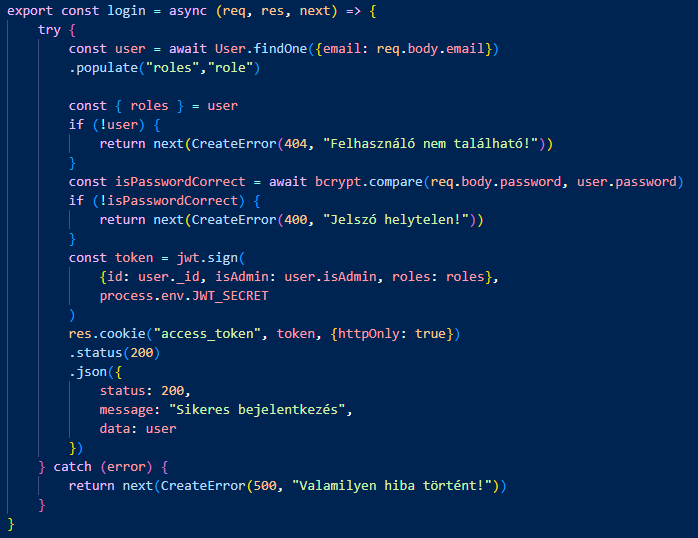
A kontroller több feladatot is ellát:

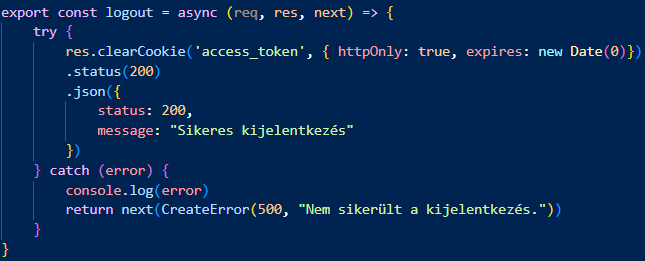
* http kérések fogadása és válaszok feldolgozása
* adatok validálása
* válaszok kialakítása és visszaküldése
  + kontroller dönti el, hogy milyen formátumban adja vissza a választ (pl. JSON, HTML, XML)

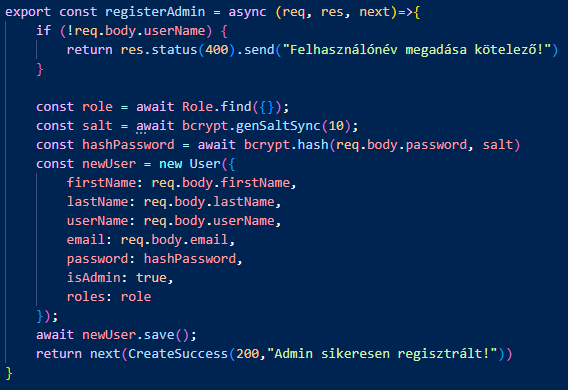


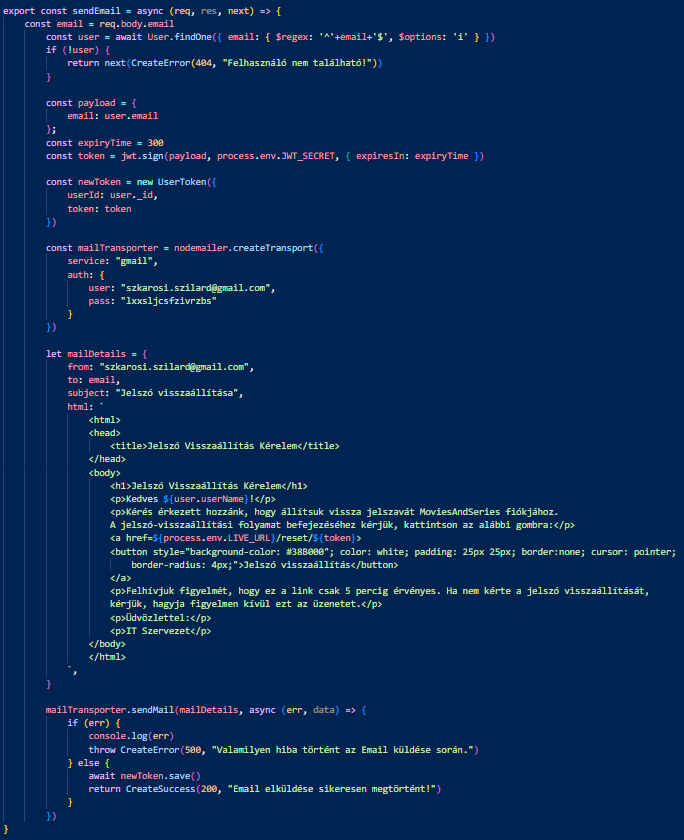
authentication.controller.js:

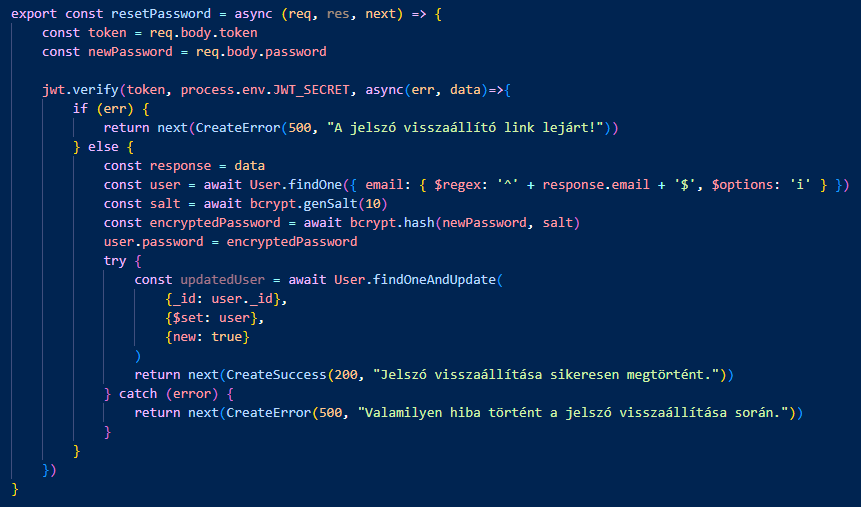




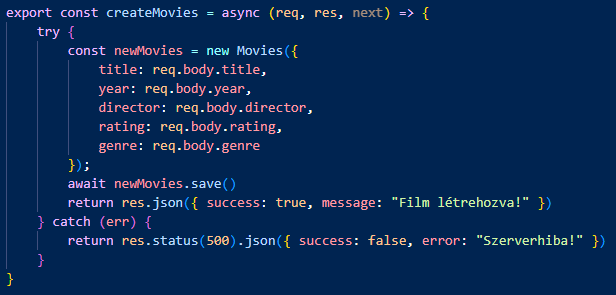


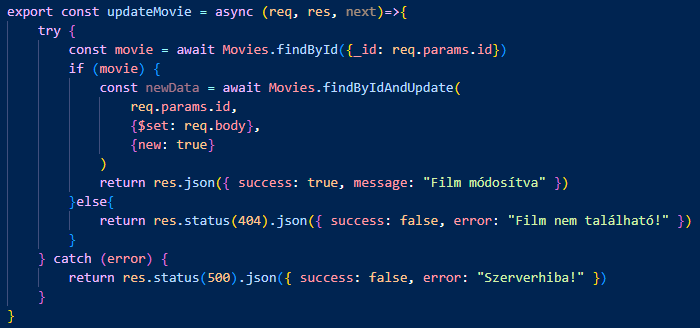


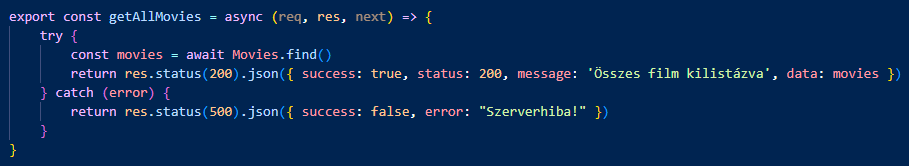




movies.controller.js:



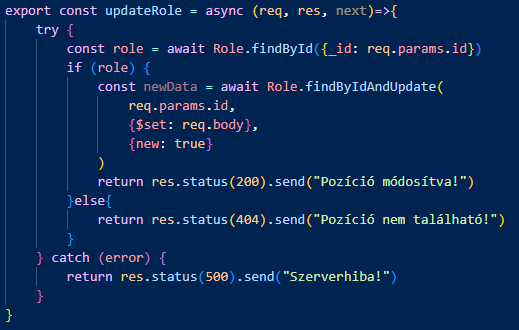


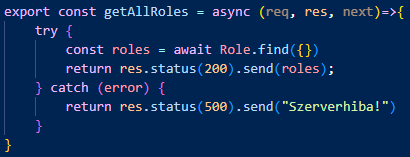


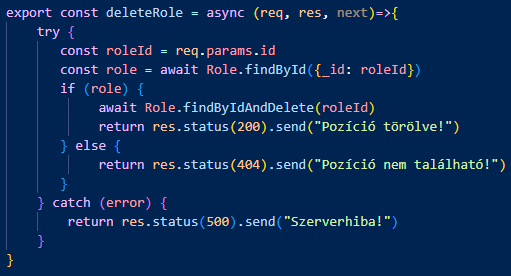


role.controller.js:

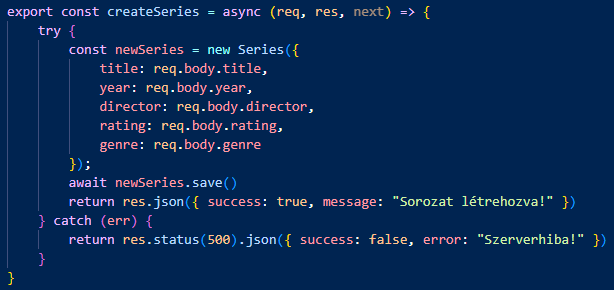


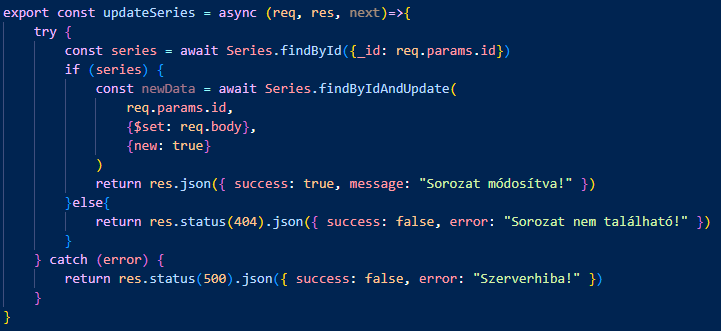


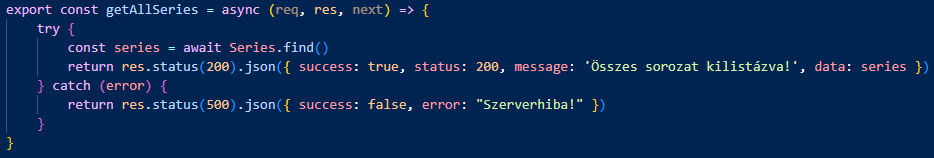


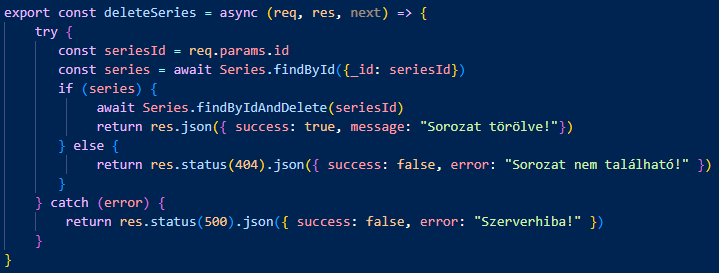


series.controller.js:

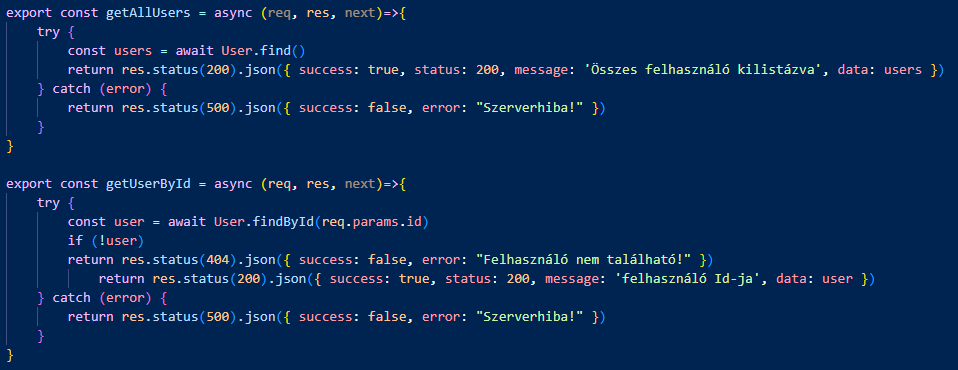








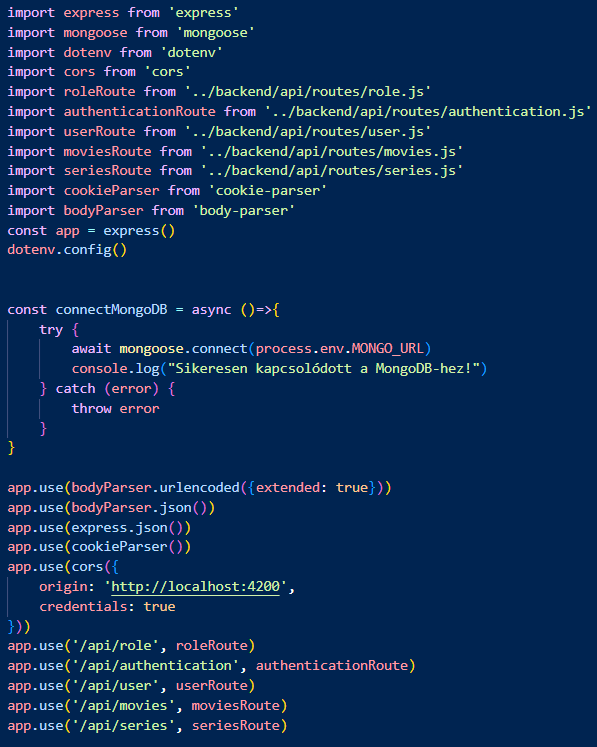
user.controller.js:

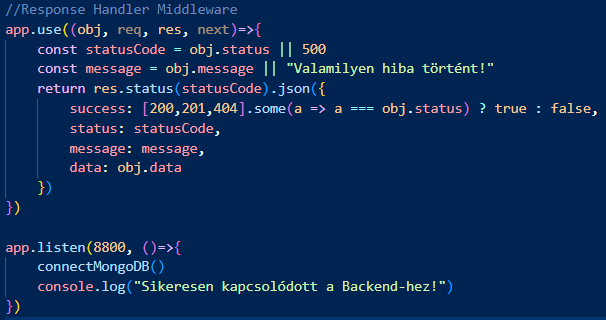


## Server.js

A server.js fájl a backend alkalmazás belépési pontja. Ez a fájl felelős a szerver konfigurálásáért, kérések feldolgozásáért, adatbázishoz való kapcsolódásért és a válaszok elküldéséért.

server.js:

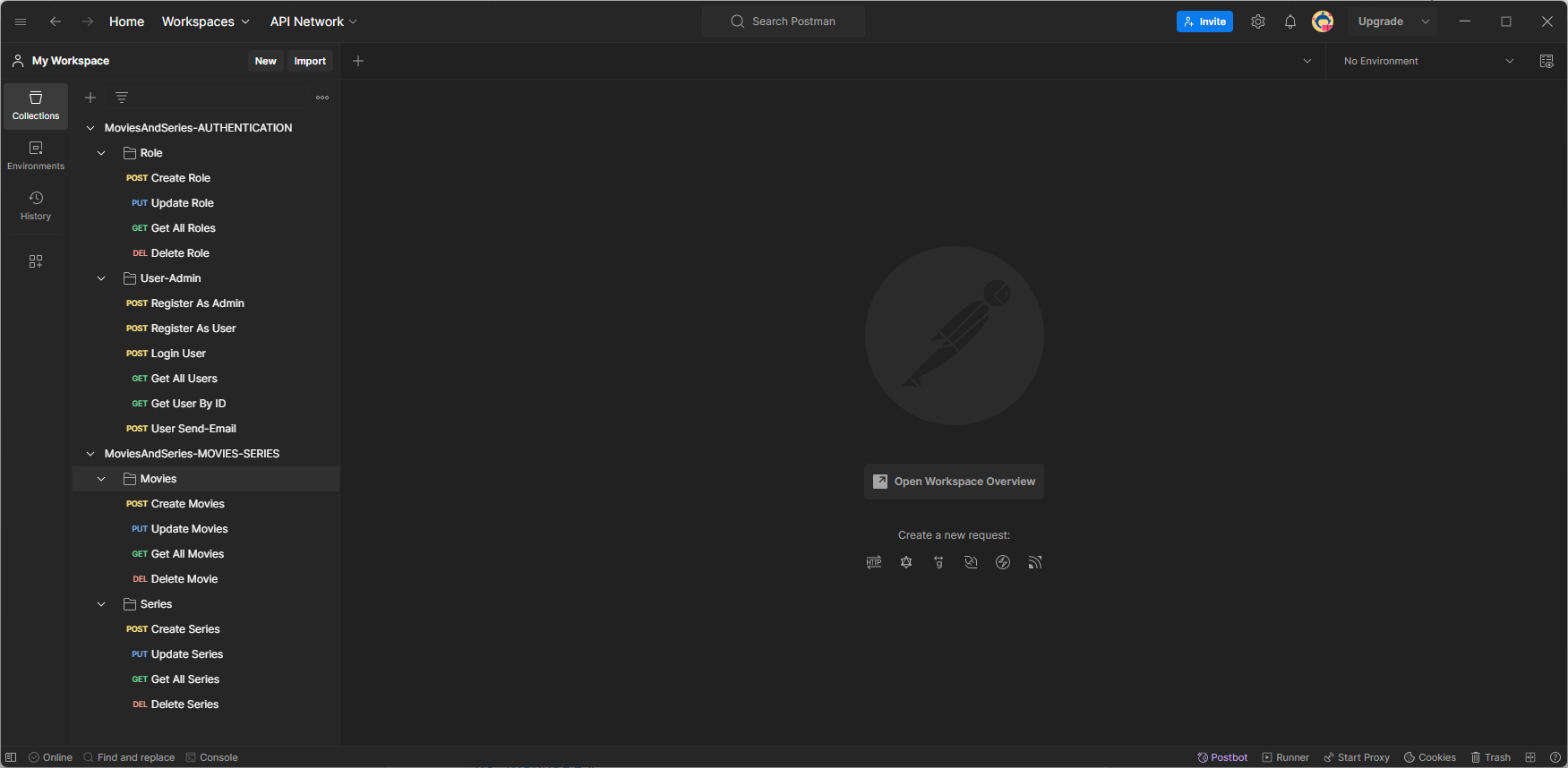




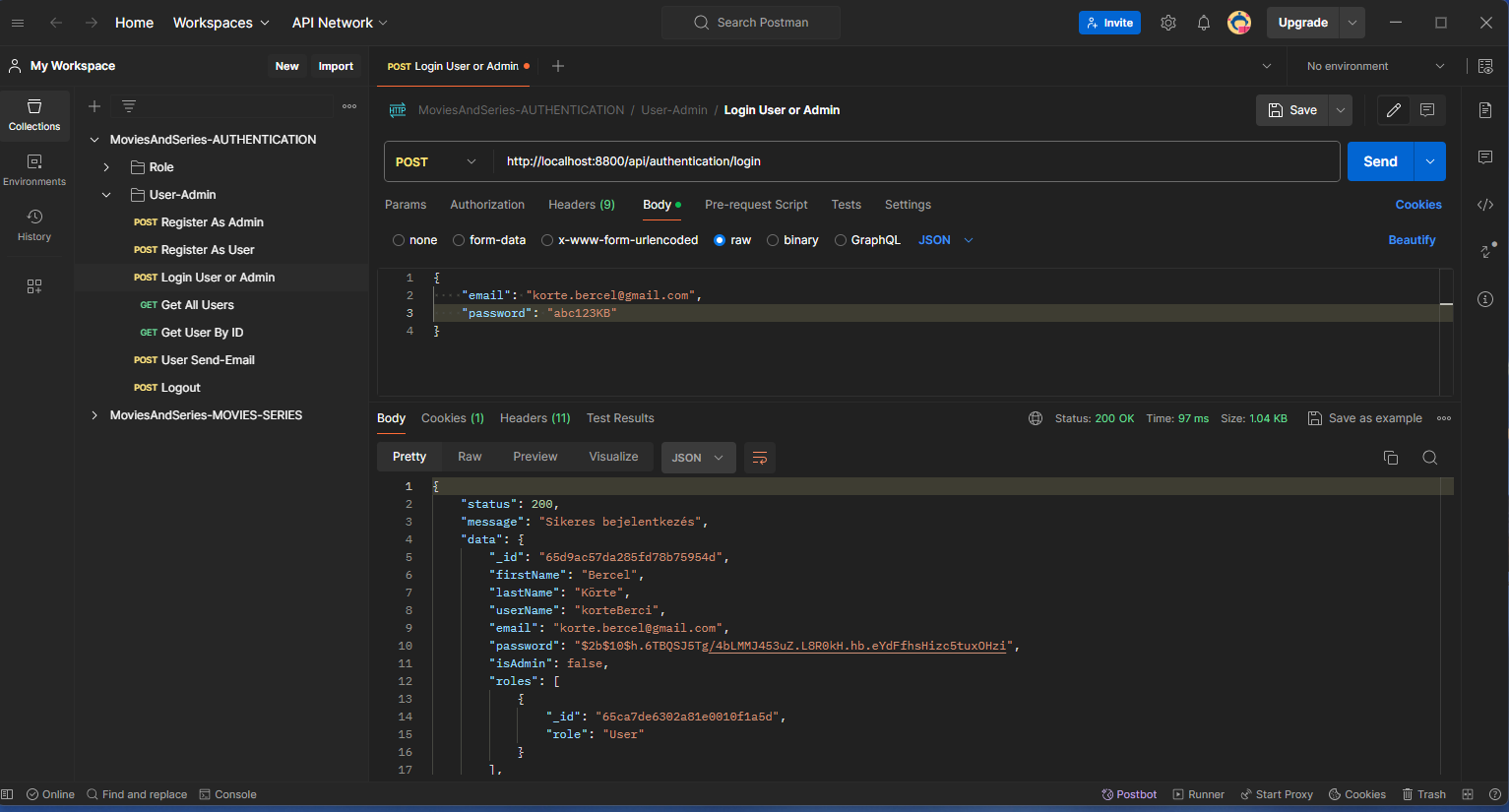
## Postman

A http kérések megfelelő működésének ellenőrzésére a Postmant használtam. A Postman egy olyan alkalmazás, amelynek segítségével az API-k különböző műveleteit (pl.: kérések küldése, válaszok fogadása, hibakeresés) tudjuk hatékonyan tesztelni és dokumentálni.

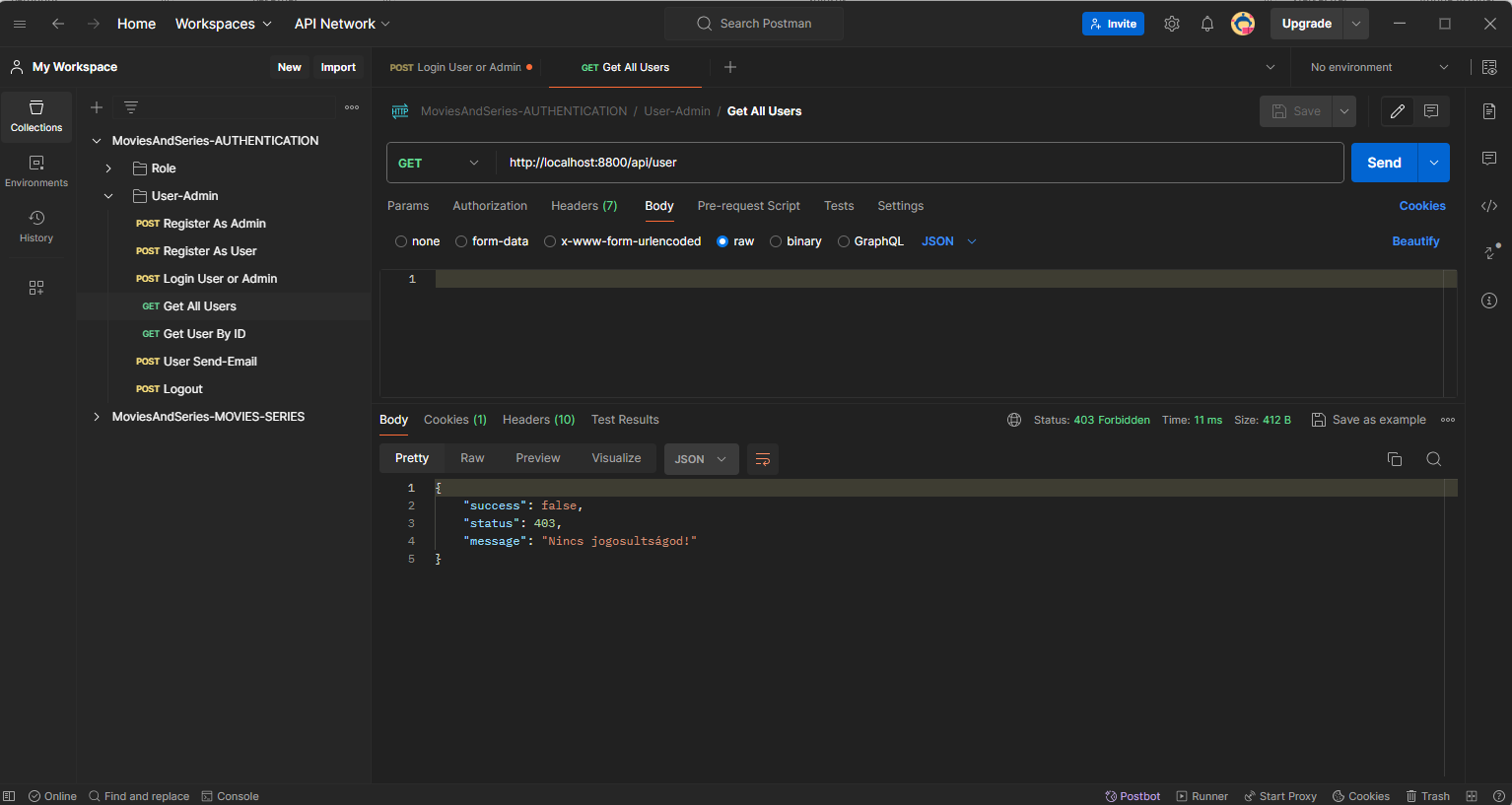
Postman:

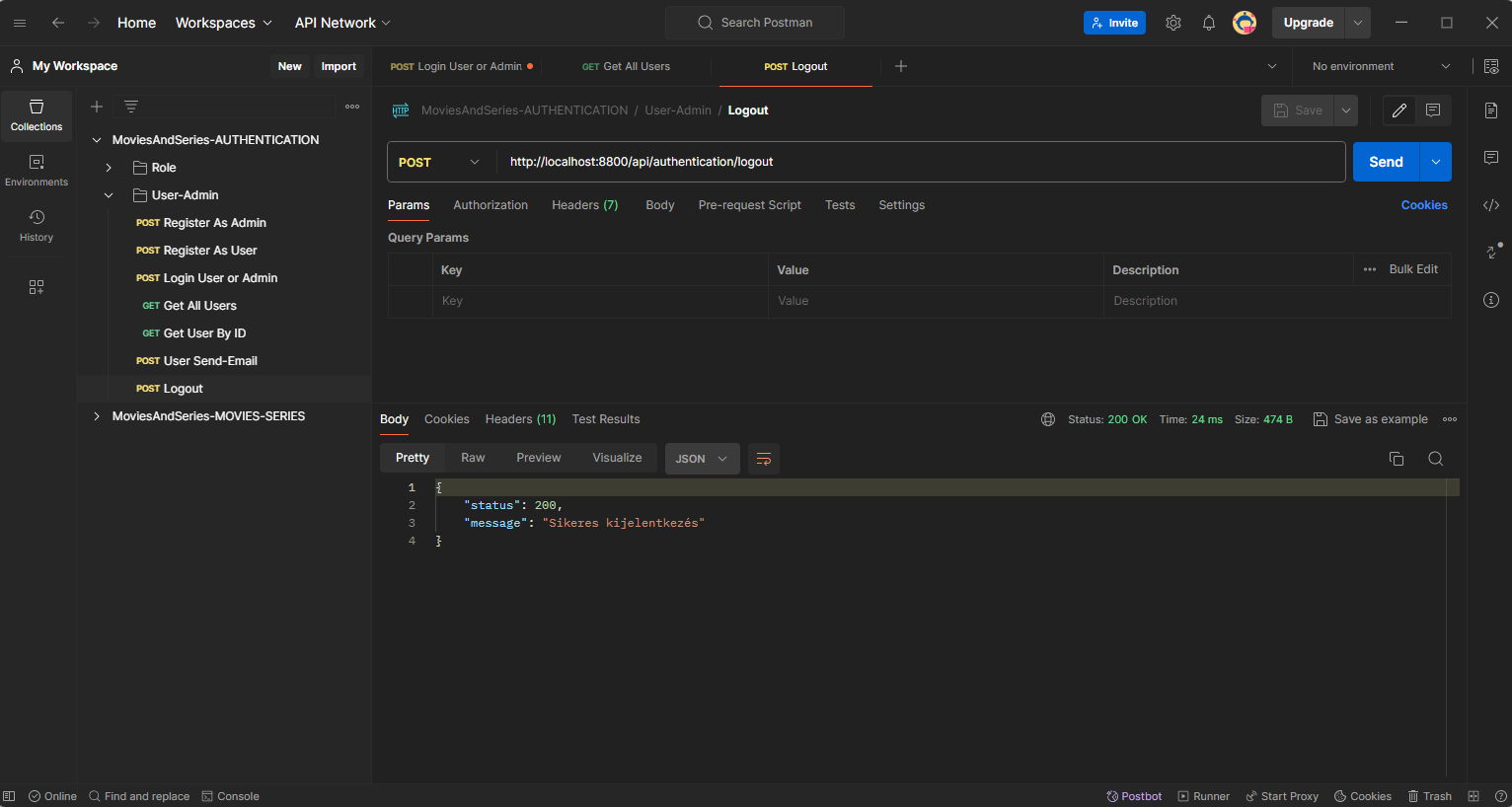


Bejelentkezés user-ként:

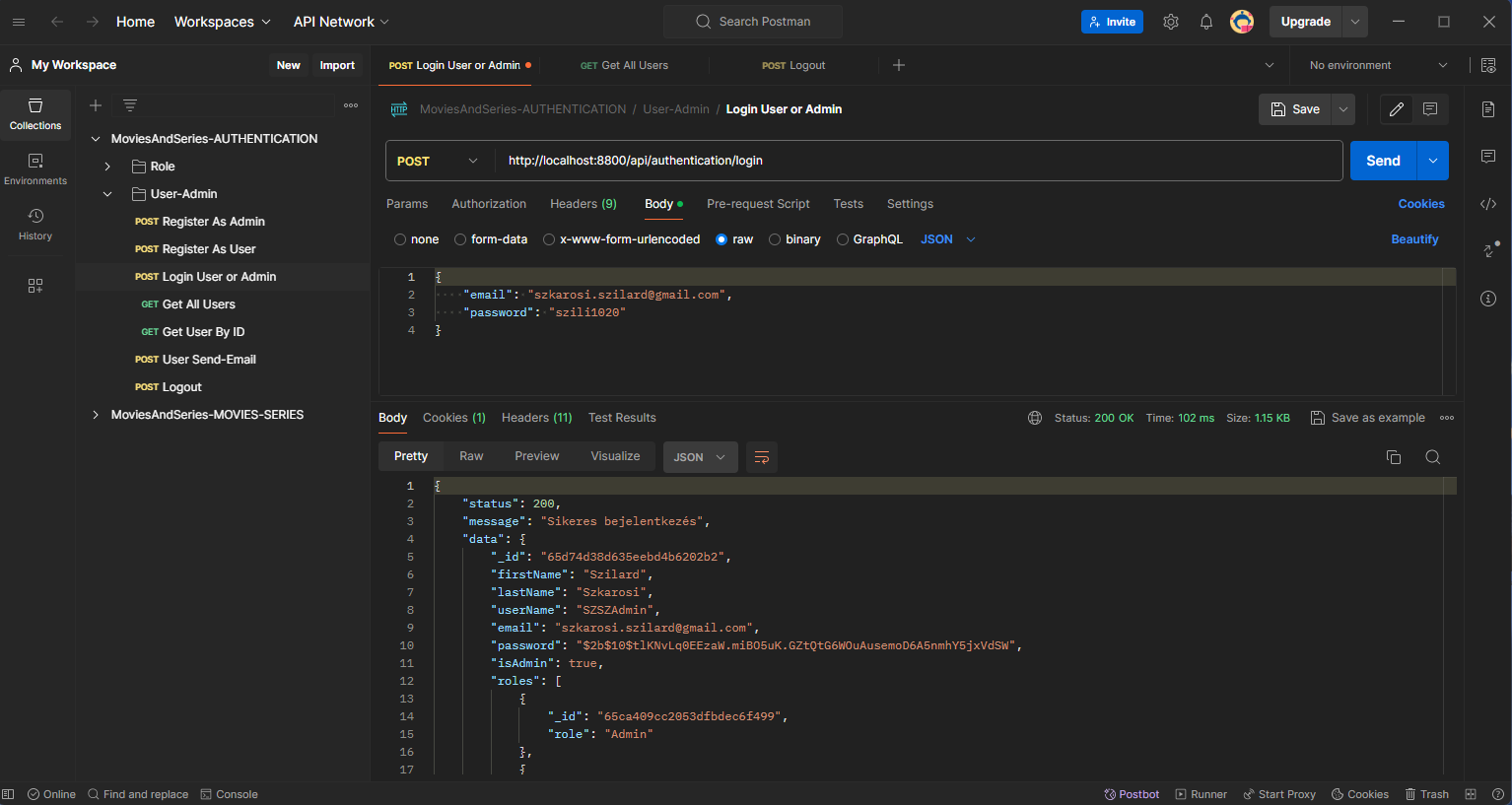


Mivel a felhasználó user-ként jelentkezett be, ezért nincs jogosultsága lekérni az összes felhasználót. Csak az admin teheti meg.

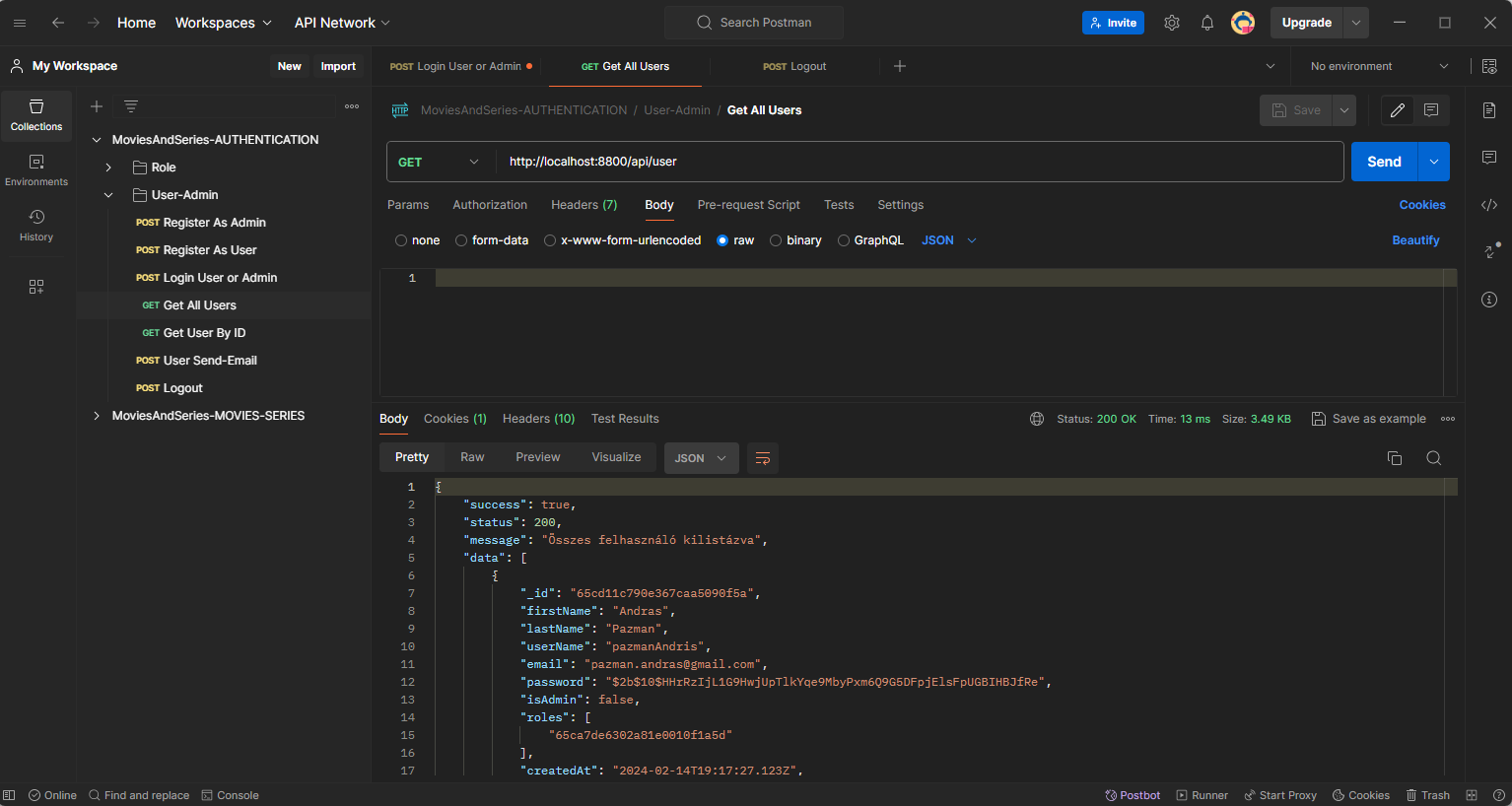




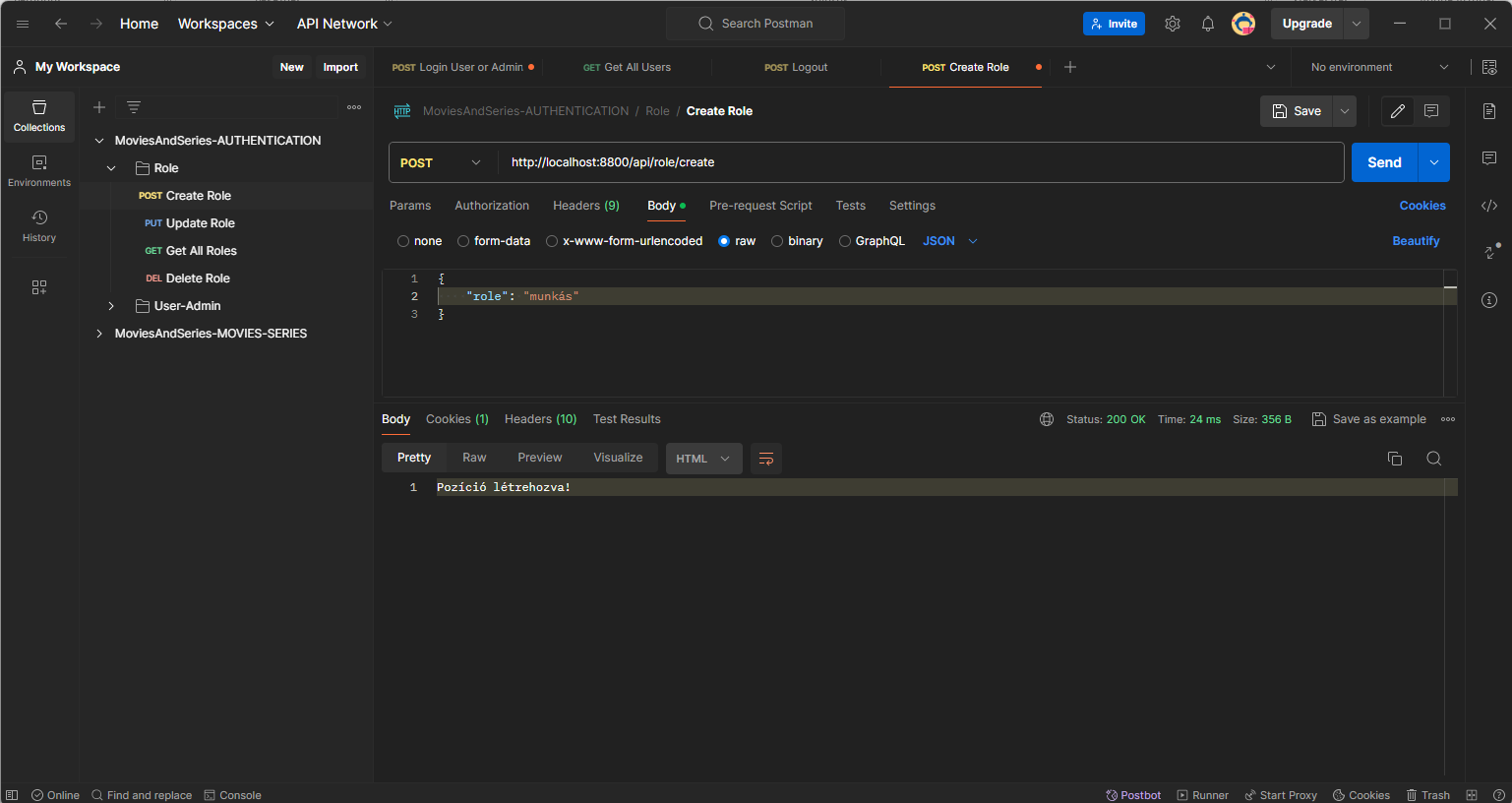
Bejelentkezés adminként:



Az admin képes az összes felhasználót és adott ID-val rendelkező felhasználót lekérni.



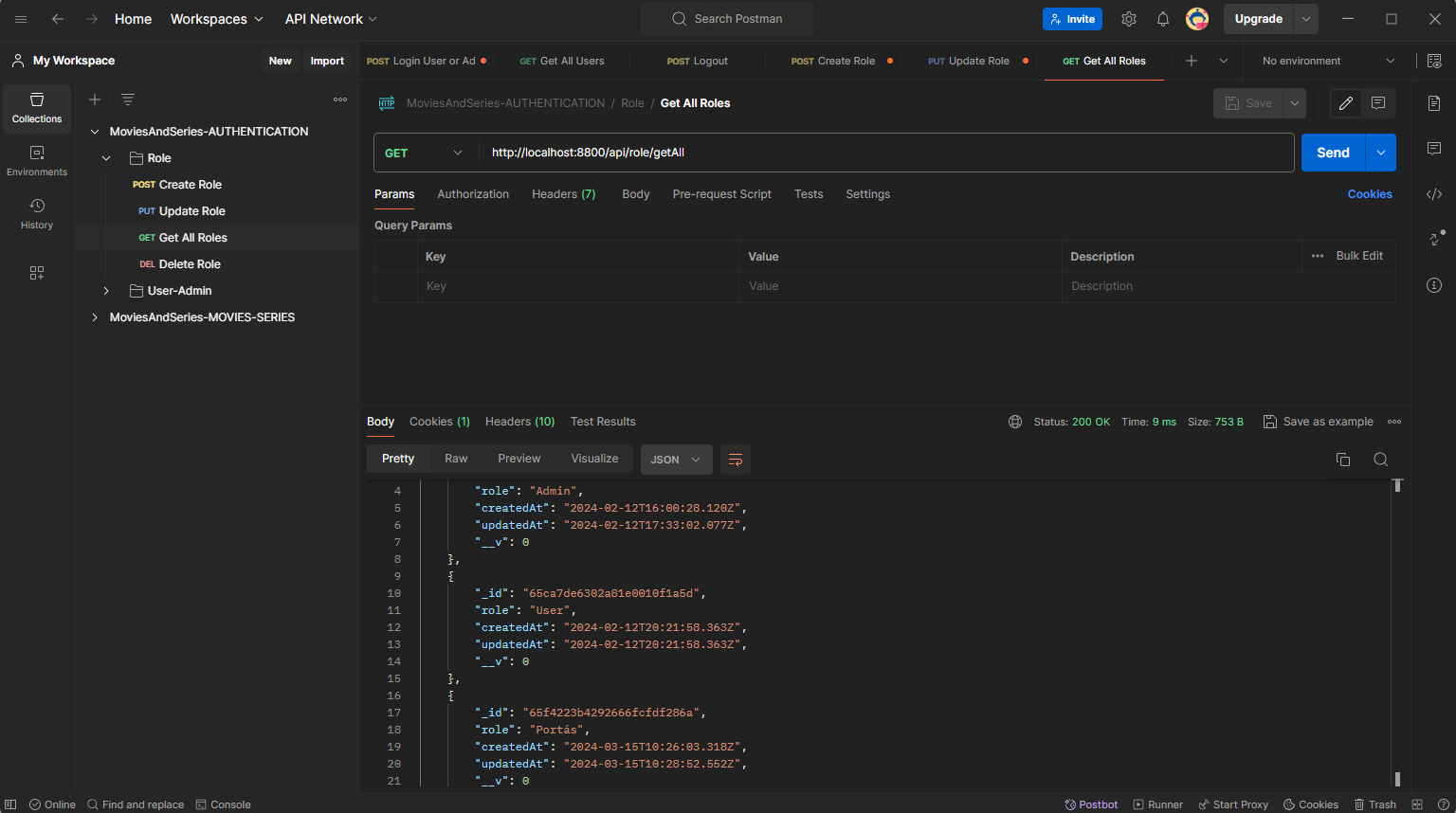
Az admin bejelentkezés után tud új pozíciót létrehozni.



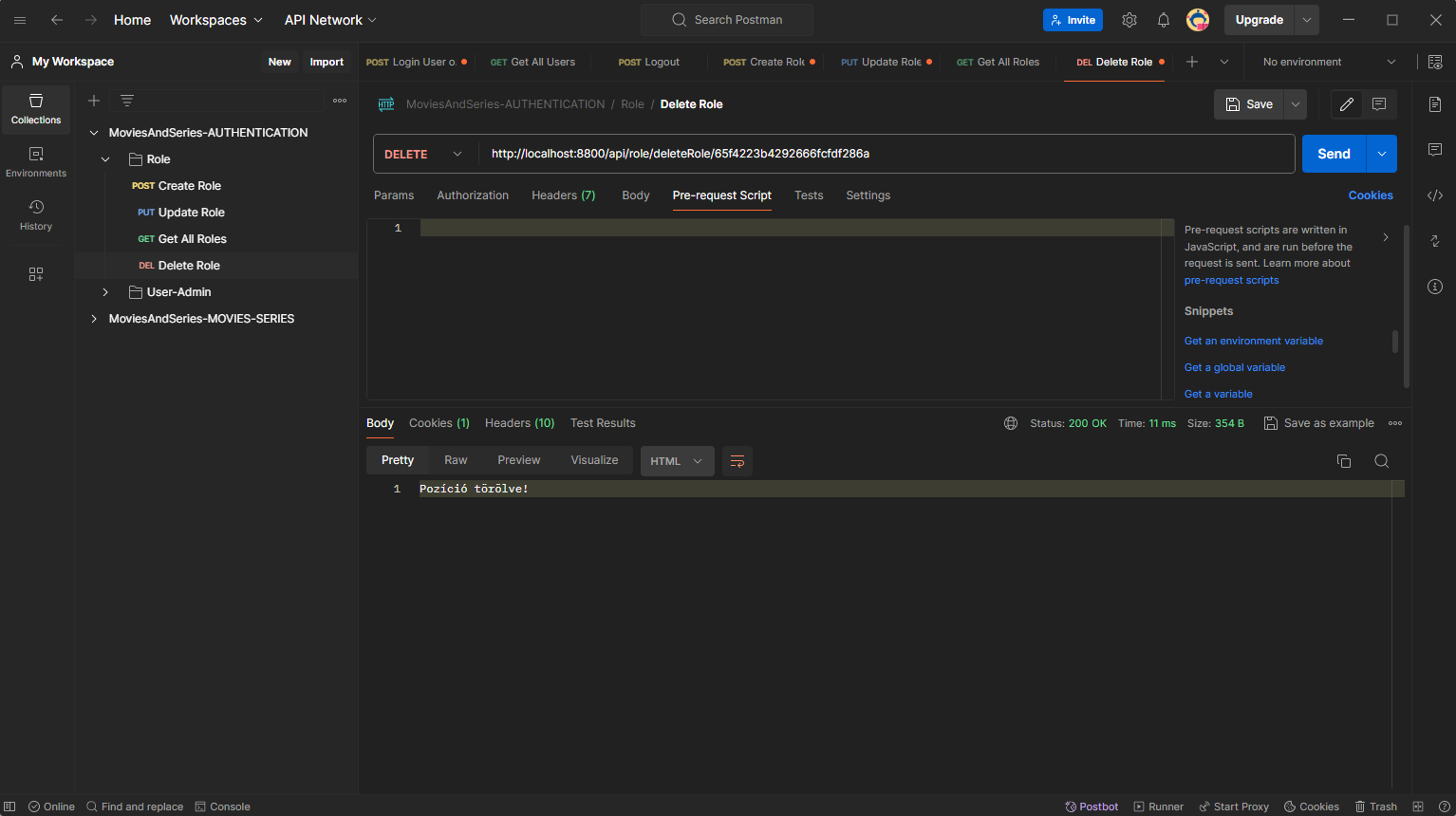
Tud pozíciót módosítani ID alapján.



Le tudja kérni az összes pozíciót.



Tud pozíciót törölni ID alapján.



## MongoDB

A MongoDB egy dokumentum-orientált adatbázisrendszer, amely az adatokat dokumentum formájában tárolják.

MongoDB Compass:

