

BKSZC Pogány Frigyes Technikum

**Filmnapló**

Szoftverfejlesztő- és tesztelő vizsgaremek

2025. május

Készítette:  
Béres Rajmund  
Szabó Levente  
Kukovics Milán

[Specifikáció 2](#_Toc219009211)

[Telepítési útmutató 3](#_Toc783436386)

[Adatbázis 3](#_Toc1553649690)

[Users tábla 3](#_Toc303840268)

[Film\_diary tábla 4](#_Toc1574314245)

[Film\_recommendation tábla 4](#_Toc37854589)

[Film\_likes 4](#_Toc504825770)

[Series\_recommendation tábla 4](#_Toc545428553)

[Series\_likes 4](#_Toc1211443904)

[Tesztadatok 5](#_Toc82297247)

[Adatbázis jogok 6](#_Toc1878813313)

[Weboldal 6](#_Toc50489451)

[Regisztráció 12](#_Toc362383398)

[Bejelentkezés 13](#_Toc1781578858)

[Backend 13](#_Toc2110132873)

[Server.js: 13](#_Toc1743482021)

[Naplo.js 18](#_Toc1699310574)

[Register.js 20](#_Toc546619642)

[Login.js 21](#_Toc1131485437)

[Recommendation.js 22](#_Toc1609684390)

[Profil.js 25](#_Toc1951084055)

[Trending-movies.js 28](#_Toc1598325065)

[Frontend 30](#_Toc1624068906)

[HTML 30](#_Toc1605196484)

[Felkapott.html 30](#_Toc1744103174)

[Index.html 33](#_Toc257293987)

[Naplo.html 35](#_Toc402942903)

[Registration.html 35](#_Toc1085121035)

[Profil.html 36](#_Toc1642816299)

[Javascript 37](#_Toc1769639424)

[Auth.js 38](#_Toc1831490720)

[Footer.js 39](#_Toc59923228)

[CSS 39](#_Toc517883633)

[auth.css 40](#_Toc1086354374)

[Main.css 43](#_Toc1596268912)

[Naplo.css 45](#_Toc656069369)

[Profil.css 51](#_Toc1748414709)

[styles.css 51](#_Toc18872440)

[Tesztelés 60](#_Toc1534963177)

# Specifikáció

Egy filmnapló lesz, amiben az emberek naplózhatják éppen mit nézhetnek meg

Azért ezt a feladatot választottuk, mert mindenkit összekötnek a filmek. Rengeteg ember néz filmeket

Az adatokat egy online adatbázisban kell tárolni, amelyet egy API-n keresztül lehet elérni. Az elkészített weblapok erről az API-ról töltik le és jelenítik meg az adatokat.

A felhasználókról a következő adatot kell tárolni:

* Felhasználó egyedi azonosítóját
* Felhasználó nevét
* Az email címét
* És a jelszavát

Az adminok által feltett filmről a következő adatokat tároljuk:

* Az egyedi azonosítóját
* A címét
* Rövid leírást a filmről
* A műfaját
* És a film egy borítóképét

Az adminok által feltett sorozatról a következő adatokat tároljuk:

* Az egyedi azonosítóját
* A címét
* Rövid leírást a filmről
* A műfaját
* És a sorozat egy borítóképét

A felhasználók által feltett filmekről a következő adatokat tároljuk:

* az egyedi azonosítóját
* A felhasználó azonosítóját
* A film címét
* A rendezőjét
* A dátumot amikor feltették
* A műfaját
* És az értékelését

A filmek kedveléseiről a következő adatokat tároljuk:

* Egy azonosítót
* A felhasználó azonosítóját
* A film azonosítóját
* A kedvelés idejét

A sorozat kedveléseiről a következő adatokat tároljuk:

* Egy azonosítót
* A felhasználó azonosítóját
* A film azonosítóját
* A kedvelés idejét

Ebben a webalkalmazásban a felhasználók feltehetnek filmet, a filmet értékelhetik.

# Telepítési útmutató

A server mappába lépés után telepítsd az összes szükséges modult az npm i express bcrypt jsonwebtoken mysql2 path cors paranccsal. Az filmnaplo.js importálása automatikusan létrehozza a szükséges táblákat, így az adatbázist előre nem kell elkészíteni. A bejelentkezési és regisztrációs funkciók már működnek, minden felhasználó a saját értékeléseit látja a napo.html oldalon. Javasolt egy profil.html létrehozása kijelentkezési lehetőséggel. Az admin fiókhoz használható belépési adatok: username: admin1, password: admin123. Minden profillal kapcsolatos API-hívásnál szükséges a Bearer ${token} fejlécek használata, és adatbázisba mentéskor a megfelelő JSON struktúrát kell megadni.

Adatbázis

Az adatokat mysql adatbázisban tároljuk.

Az adatbázis hat fő táblát tartalmaz: film\_diary, film\_likes film\_recommendation, series\_recommendation, series\_likes, users

#### Users tábla

|  |  |  |
| --- | --- | --- |
| **Oszlop neve** | **Típusa** | **Leírás** |
| id | int(11) AUTO INCREMENT | A felhasználó azonosítója |
| permission | int(2) | Jogosultságok meghatározása |
| username | varchar(250) | A felhasználó neve |
| email | varchar(250) | A felhasználó emailje |
| password | Varchar(250) | A felhasználó jelszava |

#### Film\_diary tábla

|  |  |  |
| --- | --- | --- |
| **Oszlop neve** | **Típusa** | **Leírás** |
| id | int(11) | A film azonosítója |
| user\_id | int(11) | A usershez tartozó idegen kulcs |
| title | varchar(255) | A film címe |
| release\_year | int(11) | A film megjelenésének éve |
| genre | varchar(255) | A film műfaja |
| director | varchar(255) | A film rendezője |
| review | text | A filmről írt vélemény |
| created\_at | datetime | A vélemény megalkotásának időpontja |

#### Film\_recommendation tábla

|  |  |  |
| --- | --- | --- |
| **Oszlop neve** | **Típusa** | **Leírás** |
| id | int(11) | A film azonosítója |
| title | varchar(255) | A film címe |
| content | text | Leírás a filmről |
| genre | varchar(100) | A film műfaja |
| pic\_url | text | A film boritója |

#### Film\_likes

|  |  |  |
| --- | --- | --- |
| **Oszlop neve** | **Típusa** | **Leírás** |
| id | int(11) | azonosító |
| userId | int(11) | A felhasználó azonosítója |
| filmId | int(11) | A film azonosítója |
| created\_at | timestamp | A kedvelés ideje |

#### Series\_recommendation tábla

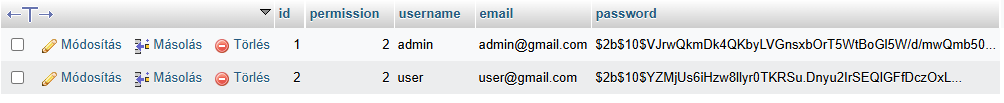
|  |  |  |
| --- | --- | --- |
| **Oszlop neve** | **Típusa** | **Leírás** |
| id | int(11) | A sorozat azonosítója |
| title | varchar( | A sorozat címe |
| content | text | A sorozat leírása |
| genre | varchar | A sorozat műfaja |
| pic\_url | text | A sorozat borítója |

#### Series\_likes

|  |  |  |
| --- | --- | --- |
| **Oszlop neve** | **Típusa** | **leírás** |
| id | int(11) | azonosító |
| userId | int(11) | A felhasználó azonosítója |
| filmId | int(11) | A film azonosítója |
| created\_at | timestamp | A kedvelés ideje |

#### Tesztadatok

A users tábla tesztadatai:



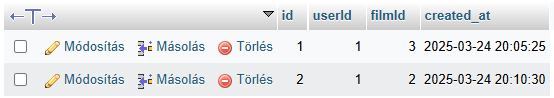
A film\_diary tábla tesztadatai:



A film\_recommendation tábla adatai:



A film\_likes tábla adatai:



A series\_recommendation tábla adatai:  

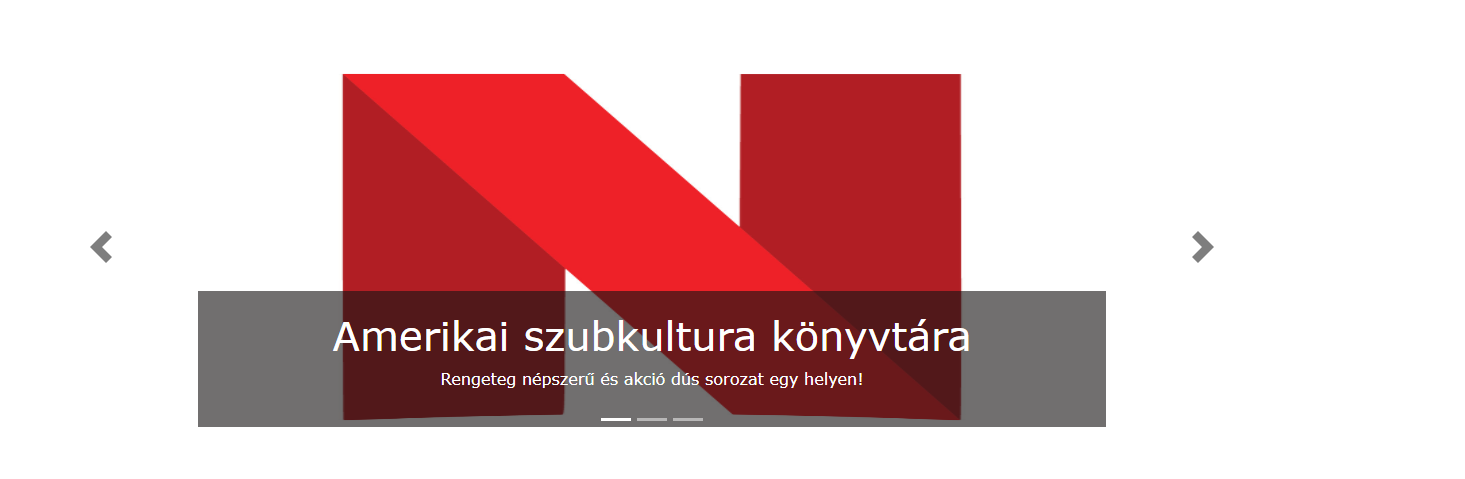

A series\_likes adatai:  


A tanfolyamok adatbázis és a három tábla létrehozásához a, a tesztadatok beviteléhez pedig a filmnaplo.sql fájlt kell importálni, vagy a bennük lévő parancsokat futtatni a PHPMyAdmin alkalmazásban.

Adatbázis jogok

Csináltunk egy admin nevű felhasználót, amit minden joggal felruháztunk. Az ehhez szükséges parancsokat az filmnaplo.sql fájlba mentettük le.

Weboldal

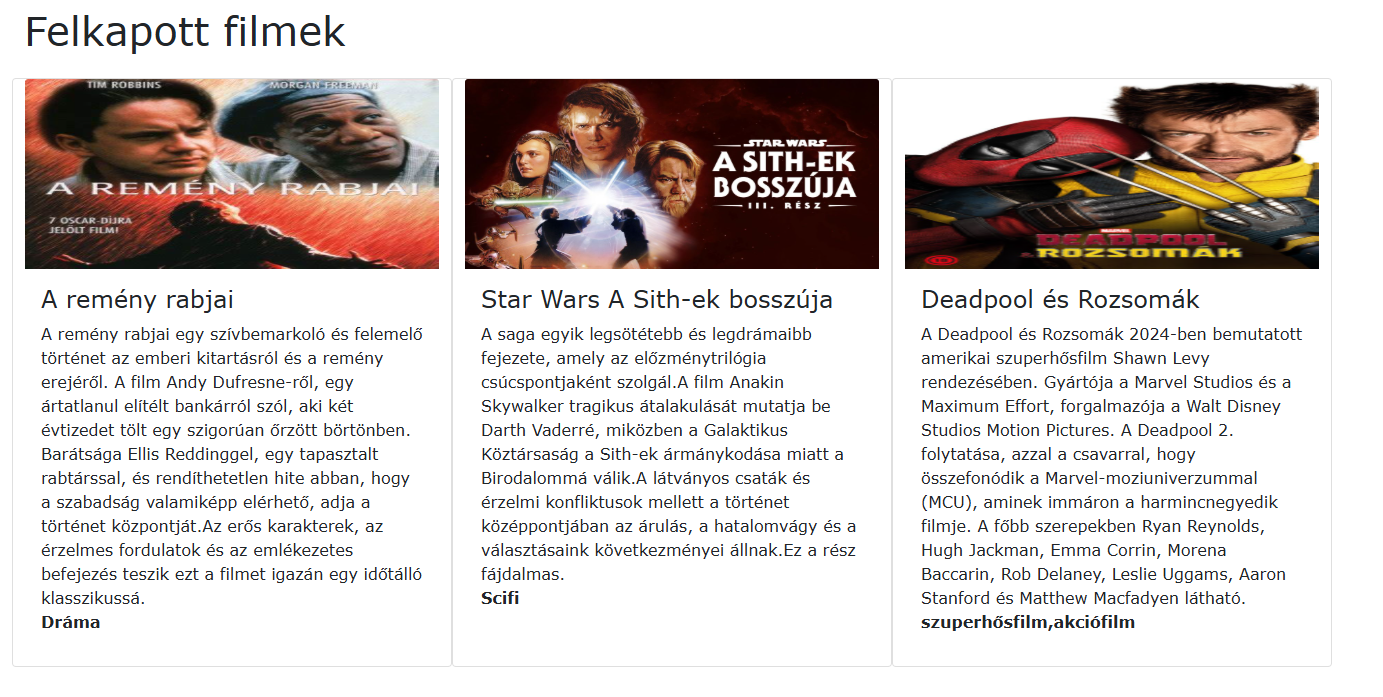

* Streaming szolgáltatások bemutatása.
* A szolgáltatások egyedi jellemzőinek kiemelése.
* Felhasználóbarát dizájn, amely interaktív élményt biztosít.
* Reszponzív elrendezés, amely minden eszközön megfelelő megjelenítést garantál.

A weboldal mobil- és asztali eszközökre optimalizált, biztosítva a zavartalan felhasználói élményt minden képernyőméreten.

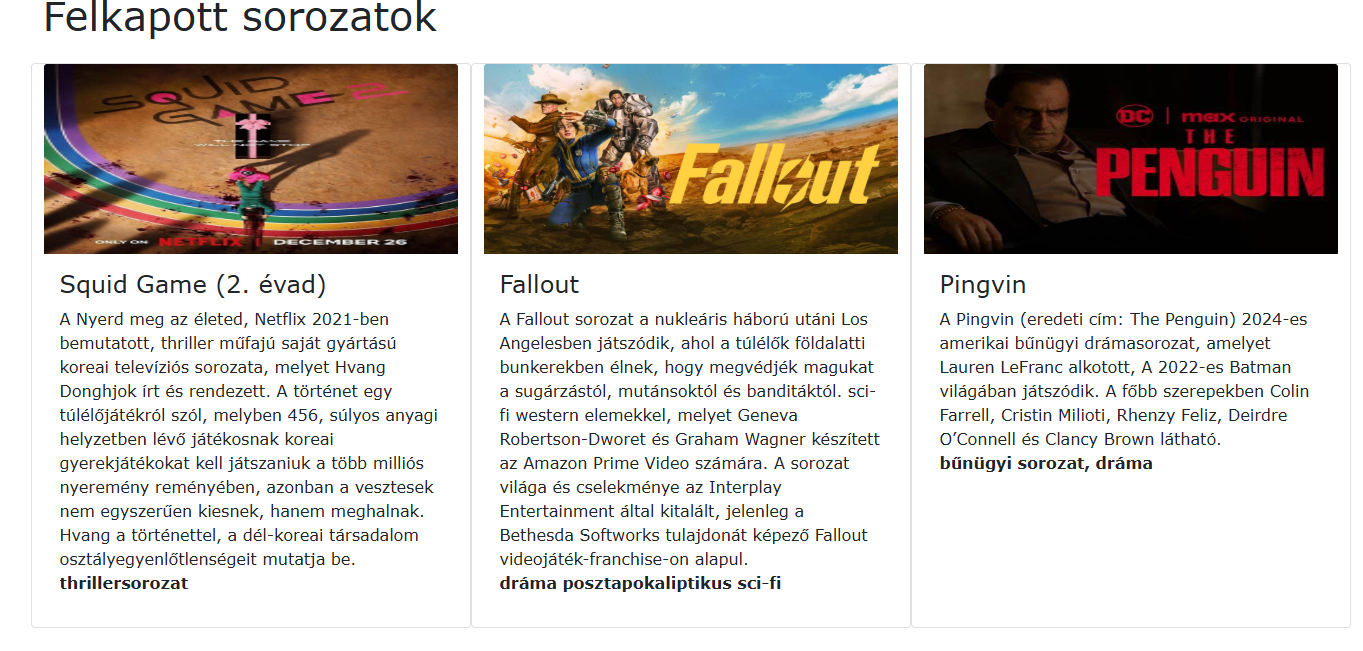
A képgaléria automatikusan lapoz de manuálisan is lapozhat a felhasználó.

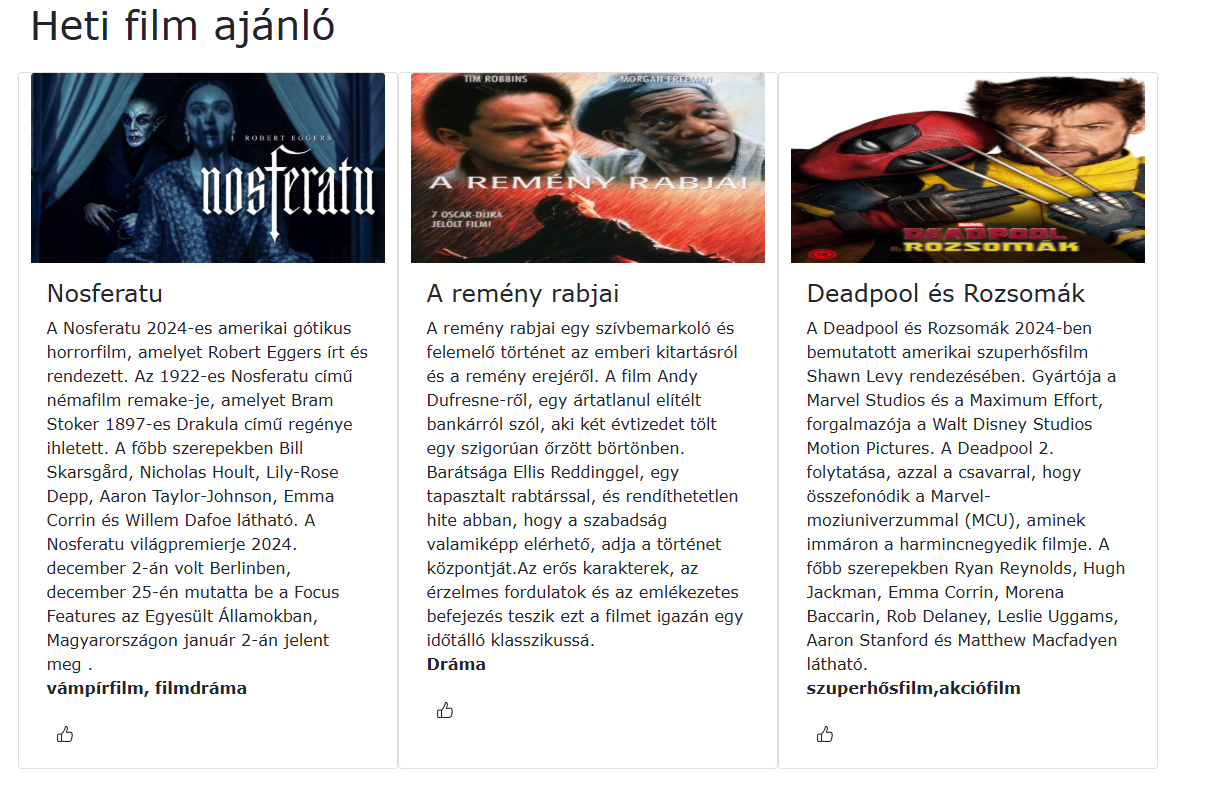
A manuális lapozás nyilak segítségével történik.

Ha az adott képre rákattintnak akkor a képen levő streaming szolgáltató oldalára vezet.



Ez az oldalrész három népszerű filmet mutat be.

Ez az oldalrész három népszerű televíziós sorozatot mutat be.



Főbb funkciók

* Három ajánlott film megjelenítése.
* Filmposzterek betöltése és megjelenítése.
* Filmekhez kapcsolódó adatok (cím, műfaj, rövid leírás, szereplők, premier dátum) dinamikus kezelése.
* Reszponzív megjelenítés különböző eszközökön.

Az oldal struktúrája:

* **Fejléc**: A heti ajánló címe.
* **Filmek listája**: A három film kártyákban jelenik meg.
* **Műfajok és leírások**: Rövid szinopszis minden filmhez.
* **Megtekintési lehetőség**: Linkek külső oldalakra vagy streaming platformokra.

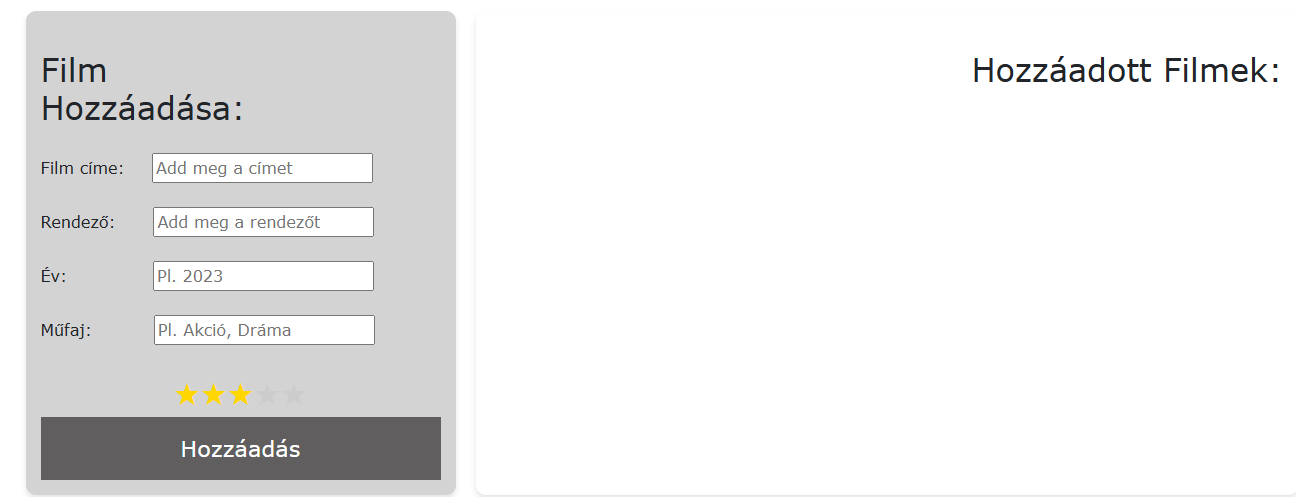


Főbb funkciók

* Három ajánlott sorozat megjelenítése.
* Sorozatposzterek betöltése és megjelenítése.
* Sorozatokhoz kapcsolódó adatok (cím, műfaj, rövid leírás, szereplők) dinamikus kezelése.
* Reszponzív megjelenítés különböző eszközökön.

Az oldal struktúrája:

* **Fejléc**: A heti ajánló címe.
* **Sorozatok listája**: A három sorozat kártyákban jelenik meg.
* **Műfajok és leírások**: Rövid szinopszis minden sorozathoz.
* **Megtekintési lehetőség**: Linkek külső oldalakra vagy streaming platformokra.



Felhasználói felület

Az alkalmazás két fő részből áll:

1. **Film hozzáadása panel** (bal oldali szürke doboz)
2. **Hozzáadott filmek listája** (jobb oldali üres terület)

Film Hozzáadása Panel

Ez a panel lehetőséget biztosít a felhasználók számára, hogy új filmet rögzítsenek. Az alábbi mezőket tartalmazza:

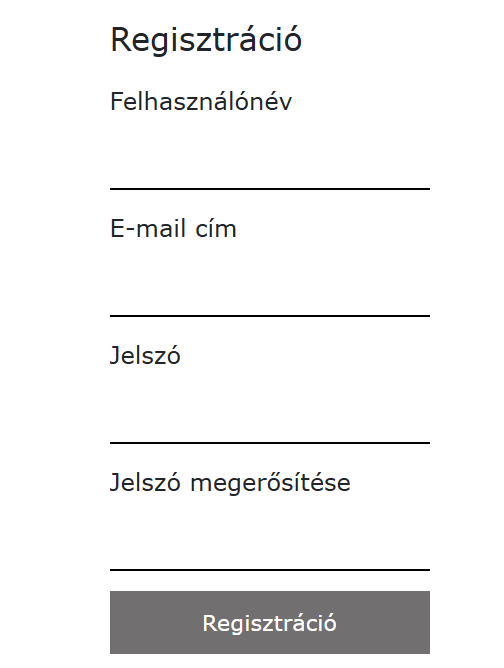
* **Film címe**: Szövegmező, ahol a felhasználó megadhatja a film címét.
* **Rendező**: Szövegmező, ahol a rendező nevét lehet beírni.
* **Év**: Év mező, amelyben az adott film megjelenési évét kell megadni (pl. 2023).
* **Műfaj**: Szövegmező, amelybe a film műfaja(i) írhatók (pl. Akció, Dráma).
* **Értékelés**: Csillag alapú értékelési rendszer (pl. 3 csillag az 5-ből).
* **Hozzáadás gomb**: Egy kattintható gomb, amely a megadott adatokat elmenti és hozzáadja a filmet a listához.

Hozzáadott Filmek Lista

Ez a panel jeleníti meg a felhasználó által hozzáadott filmeket. Kezdetben üres, de a "Hozzáadás" gomb megnyomásával a filmek megjelennek itt.

Működés

1. A felhasználó kitölti a szükséges mezőket a bal oldali panelen.
2. Az értékelés megadására csillagok állnak rendelkezésre.
3. A "Hozzáadás" gombra kattintva a film bekerül a listába.
4. A jobb oldali panelen megjelenik a hozzáadott film az összes megadott adattal.



### Regisztráció

Felhasználói felület

A regisztrációs panel az alábbi mezőket tartalmazza:

* **Felhasználónév**: Szövegmező, amelyben a felhasználó megadhatja egyedi felhasználónevét.
* **E-mail cím**: Szövegmező, amelyben az e-mail cím megadása szükséges.
* **Jelszó**: Jelszómező a bejelentkezési jelszóhoz.
* **Jelszó megerősítése**: Jelszómező a megadott jelszó újbóli beírására a megerősítés érdekében.
* **Regisztráció gomb**: Egy kattintható gomb, amely végrehajtja a regisztrációs folyamatot.

Működés

1. A felhasználó kitölti az összes szükséges mezőt.
2. A rendszer ellenőrzi, hogy a jelszavak megegyeznek-e.
3. A "Regisztráció" gomb megnyomásával a rendszer feldolgozza az adatokat és végrehajtja a regisztrációt.

Összegzés

Ez az alkalmazás egyszerű és hatékony módot biztosít a felhasználók számára, hogy filmeket rögzítsenek egy listába és regisztráljanak egy fiókot. Az intuitív felhasználói felület és az alapvető funkciók révén gyorsan és könnyedén használható.



### Bejelentkezés

Felhasználói felület

A bejelentkezési panel az alábbi mezőket tartalmazza:

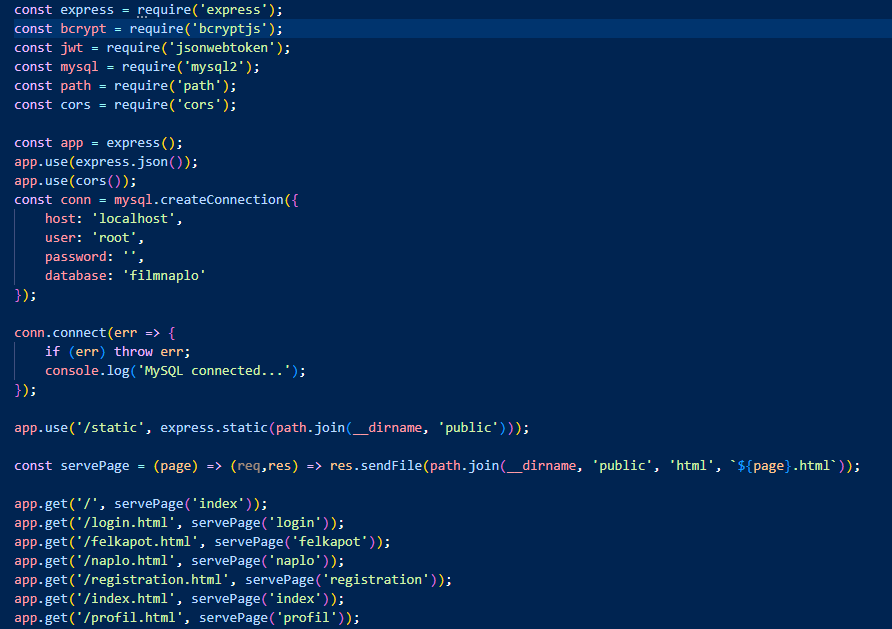
* **Felhasználónév**: A regisztrációkor megadott felhasználónév beírására szolgál.
* **Jelszó**: A felhasználó által megadott jelszó beírására szolgál.
* **Bejelentkezés gomb**: Egy kattintható gomb, amely bejelentkezteti a felhasználót a rendszerbe.
* **Regisztráció link**: Egy kattintható szöveg, amely a regisztrációs oldalra navigál, ha a felhasználónak még nincs fiókja.

Működés

1. A felhasználó megadja a felhasználónevét és jelszavát.
2. A rendszer ellenőrzi az adatok helyességét.
3. Sikeres bejelentkezés esetén a felhasználó hozzáférhet az alkalmazás többi funkciójához

Backend

Server.js:



Működés

1. Adatbázis kapcsolat

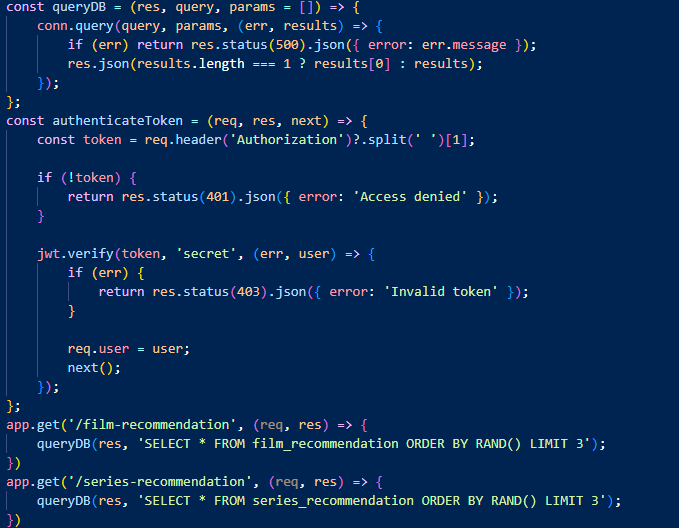
Az mysql2 csomag segítségével a szerver kapcsolódik egy MySQL adatbázishoz.

2. Statikus fájlok szolgáltatása

Az express.static middleware segítségével a public mappa tartalma elérhetővé válik a böngésző számára.

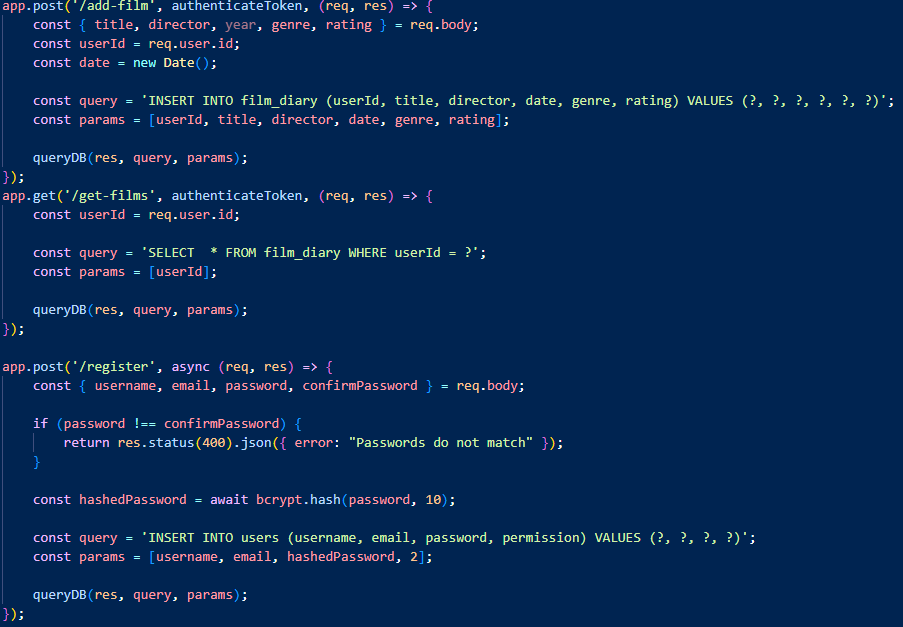
3. HTML oldalak kiszolgálása

A servePage függvény segítségével különböző HTML fájlokat szolgáltatunk, például bejelentkezési vagy profiloldalakat.



Működés

1. **Adatbázis Lekérdezések**
   1. Az adatbázis kapcsolódását és lekérdezését a queryDB segítőfüggvény végzi.
   2. Hibakezelés biztosítja, hogy a lekérdezések eredménye megfelelő formátumban kerüljön visszaadásra.
2. **Token Hitelesítés**
   1. Az API hitelesítési mechanizmussal védi a végpontokat, hogy csak jogosult felhasználók férjenek hozzá az ajánlásokhoz.
   2. A token érvényességét a jwt.verify metódus ellenőrzi.
3. **Film- és Sorozatajánlási Végpontok**
   1. **Film ajánlások**: Véletlenszerűen kiválasztott 3 filmet ad vissza az adatbázisból.
   2. **Sorozat ajánlások**: Véletlenszerűen kiválasztott 3 sorozatot ad vissza az adatbázisból.



Végpontok

1. Film hozzáadása: /add-film

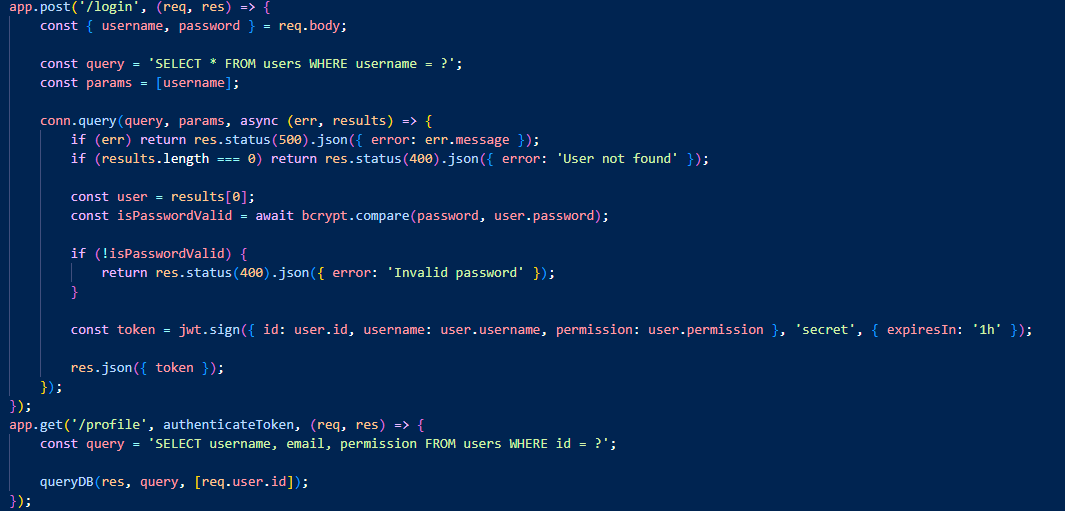
* **HTTP metódus**: POST
* **Leírás**: Hitelesített felhasználók filmeket adhatnak a naplójukhoz.
* **Autentikáció**: Kötelező (Bearer Token).
* **Kérelem törzse**:
  + title (string) - A film címe.
  + director (string) - A film rendezője.
  + year (number) - A film évszáma.
  + genre (string) - A film műfaja.
  + rating (number) - A film értékelése.
* **Válasz**:
  + Siker esetén a film naplóhoz való hozzáadásának megerősítése.
  + Hiba esetén az adott probléma részletei.

2. Filmek lekérése: /get-films

* **HTTP metódus**: GET
* **Leírás**: Hitelesített felhasználók lekérhetik a korábban hozzáadott filmeket.
* **Autentikáció**: Kötelező (Bearer Token).
* **Válasz**:
  + JSON formátumú lista, amely a filmek adatait tartalmazza (cím, rendező, műfaj, értékelés stb.).

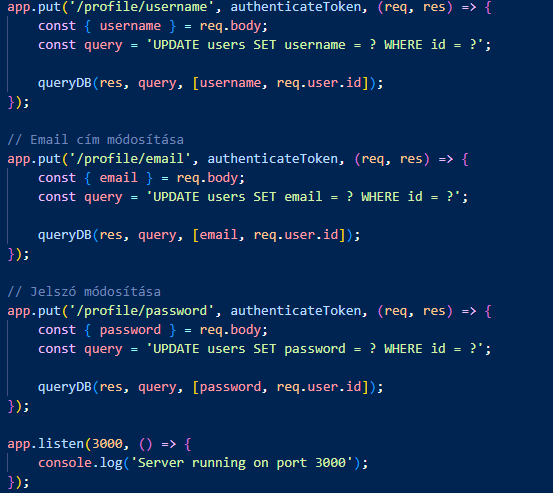
3. Felhasználói regisztráció: /register

* **HTTP metódus**: POST
* **Leírás**: Új felhasználók regisztrációja.
* **Autentikáció**: Nem szükséges.
* **Kérelem törzse**:
  + username (string) - A felhasználó neve.
  + email (string) - A felhasználó e-mail címe.
  + password (string) - A jelszó.
  + confirmPassword (string) - Jelszó megerősítése.
* **Válasz**:
  + Sikeres regisztráció esetén megerősítés.
  + Hiba esetén részletes hibaüzenet, például jelszó nem egyezés.



Működés:

1. A rendszer lekéri az adott felhasználói nevet az adatbázisból.
2. Ha nincs találat, hibaüzenetet küld vissza.
3. Ellenőrzi a jelszót bcrypt segítségével.
4. Érvényes jelszó esetén JWT-t generál és visszaküldi.
5. A middleware (authenticateToken) ellenőrzi a JWT token érvényességét.
6. A lekérdezés végrehajtása után visszaadja a felhasználó adatait.



1. Felhasználónév módosítása (PUT /profile/username)

Ez a végpont lehetővé teszi a hitelesített felhasználók számára a felhasználónév módosítását.

2. Email cím módosítása (PUT /profile/email)

Ez a végpont lehetővé teszi az email cím módosítását.

3. Jelszó módosítása (PUT /profile/password)

A felhasználó módosíthatja a jelszavát ezen a végponton keresztül.

### Naplo.js

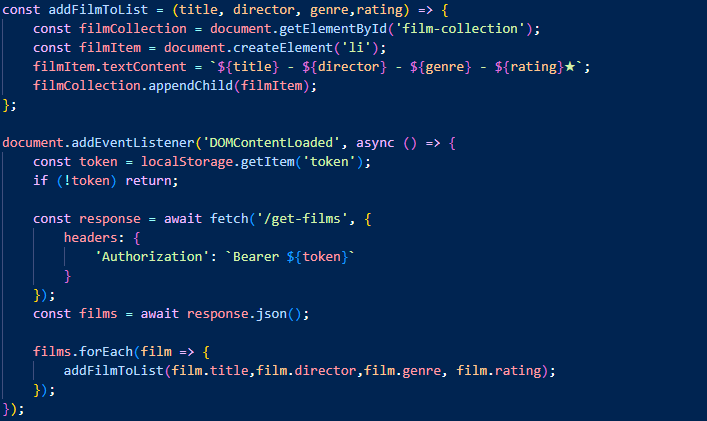


Előfeltételek

* A felhasználó rendelkezik egy érvényes tokennel a localStorage-ban.
* Az alkalmazás tartalmaz egy HTML-űrlapot, amely film címét, rendezőjét, évét, műfaját és értékelését tartalmazza.
* A backend megfelelően kezeli a /add-film végpontot.

Működés

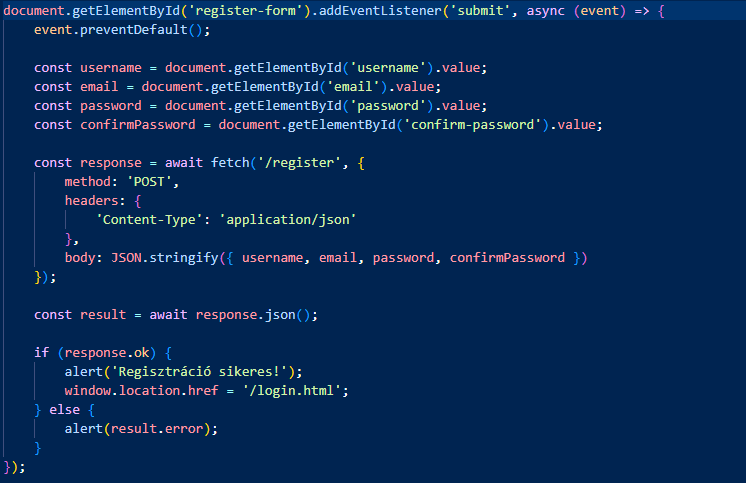
1. **Eseményfigyelő beállítása** A kód egy submit eseményfigyelőt ad az filmcreate azonosítójú HTML-elemhez, amely a film hozzáadását kezeli.
2. **Autentikáció ellenőrzése** Az eseménykezelő ellenőrzi, hogy van-e mentett token a localStorage-ban. Ha nincs, egy figyelmeztető üzenetet jelenít meg a felhasználónak.
3. **Adatok kinyerése az űrlapból** A kód a következő beviteli mezők értékeit olvassa ki:
   1. film-title
   2. director
   3. year
   4. genre
   5. rating (kiválasztott érték)
4. **Adatok elküldése a szervernek** Az adatok egy JSON objektummá alakulnak, amelyet a fetch API segítségével elküld a /add-film végpontra egy HTTP POST kérésben. A kérés tartalmazza a Content-Type: application/json fejlécet és az autentikációs tokent.
5. **Válasz kezelése**
   1. Ha a szerver sikeres választ ad, a felhasználó kap egy visszajelzést a film hozzáadásáról, és a addFilmToList() függvény frissíti a listát.
   2. Hibás válasz esetén egy hibaüzenet jelenik meg a szerver üzenetével.



Működés

1. **Film hozzáadása a listához** A addFilmToList függvény felelős egy új listaelem létrehozásáért és beszúrásáért a film-collection HTML-elembe. A film címe, rendezője, műfaja és értékelése egy formázott szövegként kerül megjelenítésre.
2. **Dokumentum betöltési esemény figyelése** A DOMContentLoaded esemény bekövetkeztekor a kód ellenőrzi, hogy van-e token elmentve, majd lekéri a filmeket a szerverről.
3. **Adatok lekérése a szerverről**
   1. A szervertől érkező választ JSON formátumban feldolgozza a fetch API segítségével.
   2. A szerver visszaküldi a filmek adatait (cím, rendező, műfaj, értékelés).
   3. Az egyes filmeket a addFilmToList függvény segítségével megjeleníti.

### Register.js



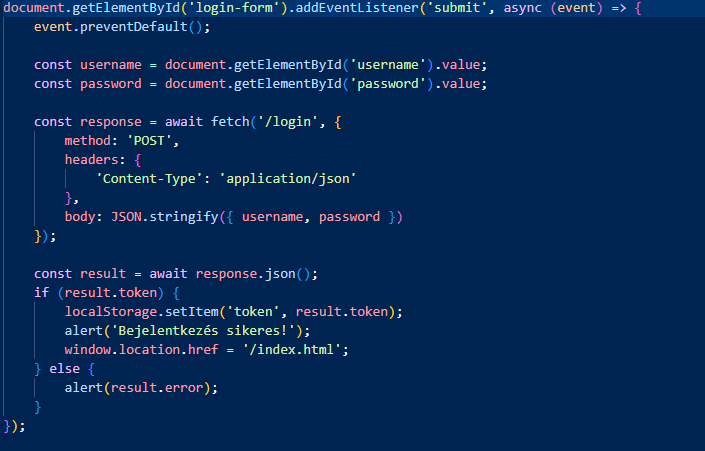
Előfeltételek

* Az alkalmazás tartalmaz egy regisztrációs űrlapot register-form azonosítóval.
* Az űrlapon szerepel a felhasználónév, e-mail cím, jelszó és jelszó megerősítés mezők.
* A backend megfelelően kezeli a /register végpontot és JSON választ küld.

Működés

1. **Űrlap beküldési esemény figyelése** A kód egy submit eseményfigyelőt ad a register-form azonosítójú HTML-elemhez, amely megelőzi az alapértelmezett beküldési viselkedést.
2. **Adatok kinyerése az űrlapból** Az eseménykezelő kiolvassa a következő mezők értékeit:
   1. username (felhasználónév)
   2. email (e-mail cím)
   3. password (jelszó)
   4. confirm-password (jelszó megerősítése)
3. **Adatok elküldése a szervernek** A fetch API segítségével a kód egy HTTP POST kérést küld a /register végpontra, amely JSON formátumban tartalmazza az űrlap adatait.
4. **Válasz kezelése**
   1. Ha a szerver sikeresen válaszol, a felhasználó értesítést kap, és átirányításra kerül a bejelentkezési oldalra.
   2. Hibás válasz esetén egy hibaüzenet jelenik meg, amelyet a szerver visszaküldött.

### Login.js



Működés

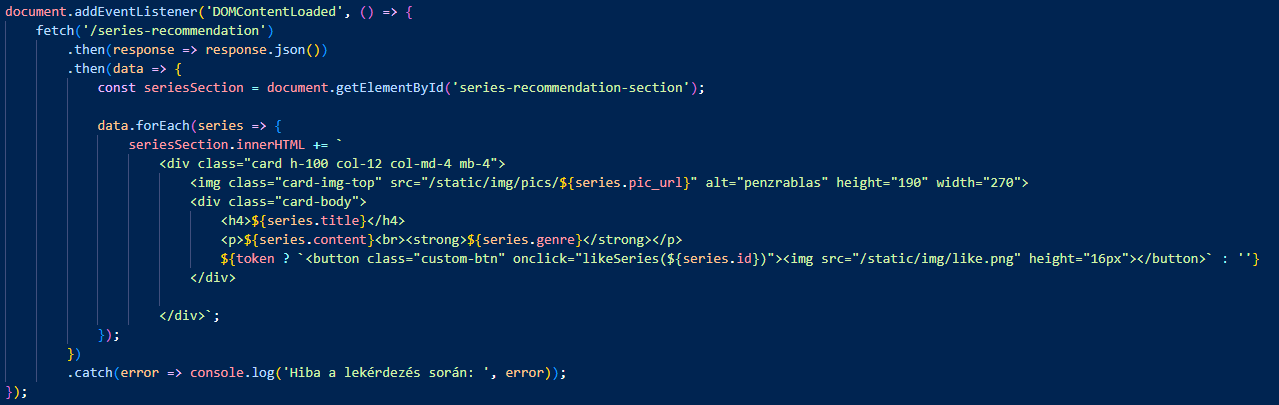
1. **Űrlap Beküldésének Kezelése**
   1. A JavaScript kód figyeli az űrlap beküldési eseményét (submit).
   2. Az alapértelmezett beküldési művelet megakadályozásra kerül (event.preventDefault()).
2. **Adatok Begyűjtése**
   1. Az űrlapból begyűjtjük a felhasználónév és jelszó mezők értékeit.
3. **Adatok Küldése Szerverre**
   1. Egy fetch hívással POST kérés formájában küldjük el az adatokat a szervernek JSON formátumban.
4. **Válasz Kezelése**
   1. Ha a szerver válasza tartalmaz tokent:
      1. A token helyileg tárolódik (localStorage segítségével).
      2. A felhasználó értesítést kap a sikeres bejelentkezésről.
      3. Az oldal átirányításra kerül egy új URL-re.
   2. Ha a szerver hibát küld vissza:
      1. A felhasználó értesítést kap a hibáról.

### Recommendation.js



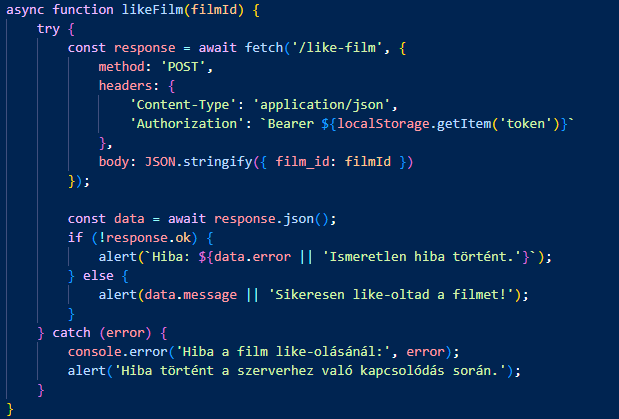
Működés

1. **Eseménykezelés**
   1. Az oldal betöltésekor (DOMContentLoaded esemény) egy JavaScript funkció fut le.
2. **Adatlekérés**
   1. Az adatok lekérése a /film-recommendation végpontról történik fetch segítségével.
   2. A válasz JSON formátumú, amely tartalmazza a megjelenítendő filmek listáját.
3. **HTML Generálás**
   1. A kapott adatokat feldolgozzuk, és minden filmhez egy kártyát hozunk létre.
   2. A kártyák tartalmazzák:
      1. Képet (URL alapján).
      2. Címet.
      3. Tartalmat.
      4. Műfajt.
4. **Hibakezelés**
   1. Hibás kérés esetén egy hibaüzenet jelenik meg a böngésző konzoljában.



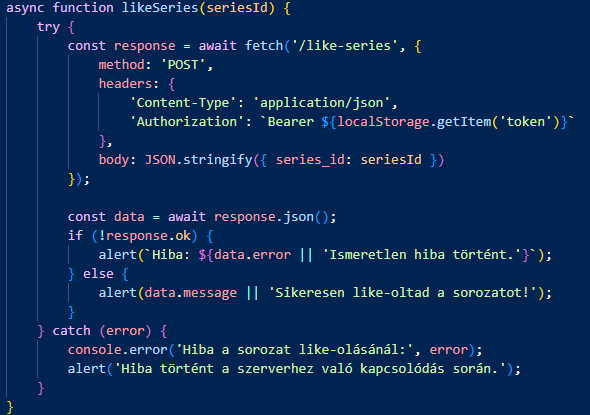
Működés

1. **Adatok Lekérése**
   1. Az adatok lekérése a /series-recommendation API végpontról történik fetch segítségével.
   2. A válasz JSON formátumú, amely tartalmazza a megjelenítendő sorozatok listáját.
2. **Dinamikus HTML Generálás**
   1. Az egyes sorozatokhoz létrehozott kártyák tartalmazzák:
      1. Képet a sorozathoz.
      2. Címet.
      3. Rövid leírást.
      4. Műfajt.
3. **Hibakezelés**
   1. Hibás kérés esetén a konzolon hibaüzenet jelenik meg.
4. **Eseménykezelés**
   1. Az oldal betöltésekor a DOM tartalmának létrehozásáért felelős funkció fut le.



Működés

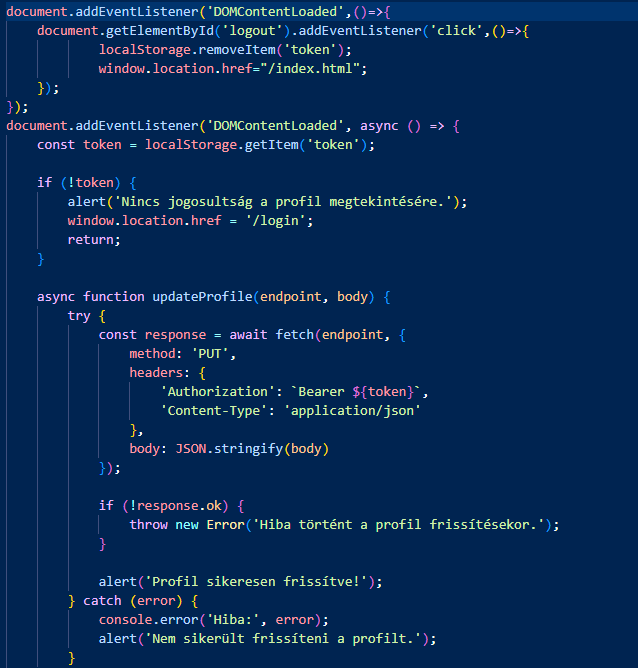
1. **HTTP kérés létrehozása**:
   * A funkció létrehoz egy POST kérést a /like-film végpont felé.
   * Az üzenet fejlécében megadja a Content-Type értéket (application/json), valamint az Authorization fejlécet, amely a felhasználó tokenjét tartalmazza (Bearer ${localStorage.getItem('token')}).
   * A törzsben elküldi a filmId-t JSON formátumban.
2. **Szerver válasz kezelése**:
   * A szerver válaszát JSON-ként dolgozza fel.
   * Ha a válasz nem volt sikeres (response.ok === false), akkor egy hibaüzenetet jelenít meg.
   * Ha sikeres volt, akkor visszajelzést ad a felhasználónak arról, hogy sikeresen like-olta a filmet.
3. **Hibakezelés**:
   * Ha hiba lép fel a folyamat során (pl. hálózati hiba), akkor a funkció naplózza a hibát és egy figyelmeztető üzenetet küld a felhasználónak.



Működés

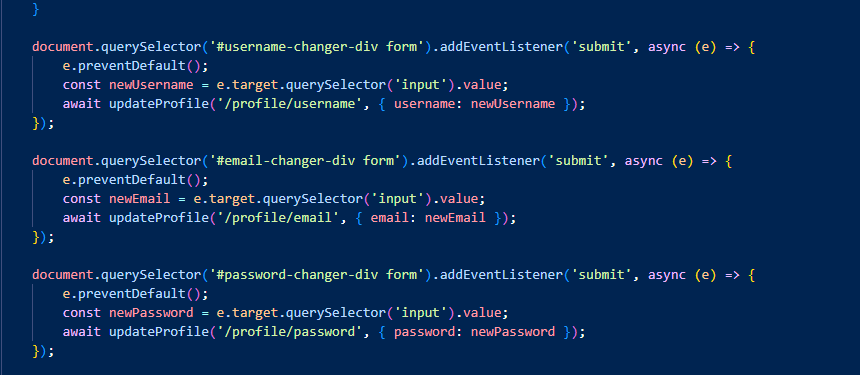
1. **HTTP kérés létrehozása**:
   * A funkció létrehoz egy POST kérést a /like-series végpont felé.
   * Az üzenet fejlécében megadja a Content-Type értéket (application/json), valamint az Authorization fejlécet, amely a felhasználó tokenjét tartalmazza (Bearer ${localStorage.getItem('token')}).
   * A törzsben elküldi a seriesId-t JSON formátumban.
2. **Szerver válasz kezelése**:
   * A szerver válaszát JSON-ként dolgozza fel.
   * Ha a válasz nem volt sikeres (response.ok === false), akkor egy hibaüzenetet jelenít meg.
   * Ha sikeres volt, akkor visszajelzést ad a felhasználónak arról, hogy sikeresen like-olta a sorozatot.
3. **Hibakezelés**:
   * Ha hiba lép fel a folyamat során (pl. hálózati hiba), akkor a funkció naplózza a hibát és egy figyelmeztető üzenetet küld a felhasználónak.

### Profil.js



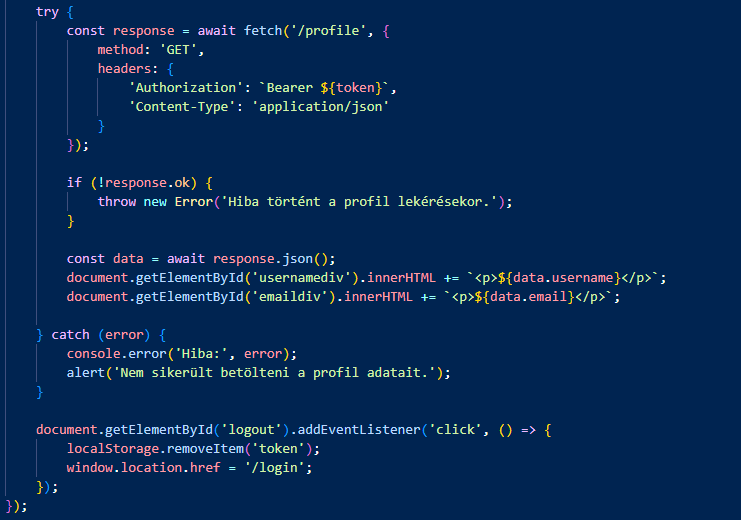
Működés

1. **Kijelentkezés Implementálása**
   1. Egy gomb (logout azonosítóval) figyeli a kattintási eseményeket.
   2. A kattintás hatására a helyileg tárolt token törlődik, és az alkalmazás átirányítja a felhasználót a kezdőoldalra.
2. **Profilkezelés és Hitelesítés**
   1. A token jelenlétét ellenőrzi az alkalmazás oldalbetöltéskor.
   2. Ha a token hiányzik, a rendszer értesíti a felhasználót, majd átirányítja a bejelentkezési oldalra.
3. **Profilfrissítés**
   1. Az updateProfile függvény lehetővé teszi a felhasználói adatok szerveroldali frissítését.
   2. A szerver válaszának sikeressége alapján visszajelzés jelenik meg a felhasználónak.
4. **Hibakezelés**
   1. Sikertelen token-ellenőrzés vagy szerverhibák esetén értesítések és naplózott hibaüzenetek segítenek a problémák azonosításában.



Működés

1. **Űrlap Beküldésének Kezelése**
   1. Minden egyes formhoz külön addEventListener eseményfigyelő van hozzáadva.
   2. Az események kezelésére az submit típus van használva.
   3. Az alapértelmezett beküldési művelet megakadályozására az event.preventDefault() parancs van alkalmazva.
2. **Adatok Begyűjtése**
   1. A felhasználó által megadott értékeket az e.target.querySelector('input').value segítségével olvassuk ki az adott űrlap mezőjéből.
3. **Adatok Küldése a Szerverre**
   1. A fetch API hívás a megfelelő végpontra (/profile/username, /profile/email, /profile/password) továbbítja az új adatokat PUT metódussal.
   2. Az adatok JSON formátumban kerülnek elküldésre a szervernek.
4. **Aszinkron Frissítés**
   1. Az async/await szintaxis biztosítja, hogy az adatküldés és válaszkezelés egyszerű és átlátható legyen.



Működés

1. **Profil Adatok Lekérése**
   1. Az alkalmazás a /profile végpontról kéri le a felhasználói adatokat HTTP GET kérés segítségével.
   2. Az adatok JSON formátumban érkeznek, és a felhasználónév, illetve email megjelenítéséhez használatosak.
2. **Adatok Megjelenítése**
   1. A kapott adatokat dinamikusan hozzáadjuk az oldalon található usernamediv és emaildiv elemekhez.
3. **Hibakezelés**
   1. Ha a profil adatok lekérése sikertelen, egy hibaüzenet kerül naplózásra a konzolban, és értesítés jelenik meg a felhasználónak.
4. **Kijelentkezés Funkció**
   1. A logout gomb eseménykezelője törli a token-t a localStorage-ból, majd átirányítja a felhasználót a bejelentkezési oldalra.

### Trending-movies.js



Működés

1. **Oldal Betöltés Esemény (**DOMContentLoaded**)**
   * Az eseményfigyelő (document.addEventListener) biztosítja, hogy a script csak akkor fusson, amikor a DOM teljesen betöltődött.
2. **Adatok Lekérése a Szervertől**
   * A fetch('/ranked-films') parancs küld egy GET kérést az API-nak.
   * A kapott JSON adatokat feldolgozza és egy tömbként visszaadja.
3. **Dinamizált Tartalom Létrehozása**
   * A kapott adatok alapján létrehoz egy filmkártya elemet (<div class="card">).
   * Az egyes filmek adatait hozzáfűzi a DOM megfelelő szekciójához.
4. **Hibakezelés**
   * Ha a lekérdezés hibába ütközik, a funkció naplózza a hibát és megjelenít egy hibaüzenetet a konzolon.



Működés

1. **Oldal Betöltés Esemény (**DOMContentLoaded**)**
   1. Az eseményfigyelő (document.addEventListener) biztosítja, hogy a script csak akkor fusson, amikor a DOM teljesen betöltődött.
2. **Adatok Lekérése a Szervertől**
   1. A fetch('/ranked-series') parancs küld egy GET kérést az API-nak.
   2. A kapott JSON adatokat feldolgozza és egy tömbként visszaadja.
3. **Dinamizált Tartalom Létrehozása**
   1. A kapott adatok alapján létrehoz egy sorozatkártya elemet (<div class="card">).
   2. Az egyes sorozatok adatait hozzáfűzi a DOM megfelelő szekciójához.
4. **Hibakezelés**
   1. Ha a lekérdezés hibába ütközik, a funkció naplózza a hibát és megjelenít egy hibaüzenetet a konzolon.

Frontend

## HTML

### Felkapott.html

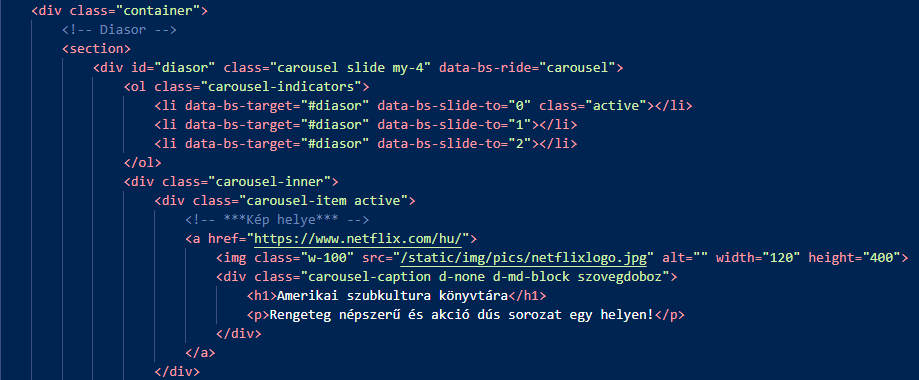


A dokumentum fejlécében megtalálható:

* Karakterkódolás beállítása (utf-8).
* Reszponzivitás biztosítása (meta viewport).
* Külső CSS fájlok csatolása (Bootstrap, main.css).
* Favicon (favicon.ico).
* Weboldal címe: **Film Napló**.

A weboldal egy Bootstrap alapú navigációs sávval rendelkezik, amely:

* Tartalmazza a weboldal nevét és főoldali hivatkozását.
* Beépített **hamburger-menü** funkcióval rendelkezik mobilnézethez.
* Tartalmaz egy üres <ul> listát (#nav-items), amely dinamikusan generált menüelemeknek biztosít helyet.



Működés

A Carousel folyamatosan váltogatja a tartalmát, az data-bs-ride="carousel" attribútumnak köszönhetően. Az indikátorok lehetővé teszik a manuális lapozást, míg az előre- és visszagombok biztosítják a navigációt. A Carousel rész mindegyik html-ben megtalálható.



A következő elemek biztosítják a navigációt a diavetítésben:

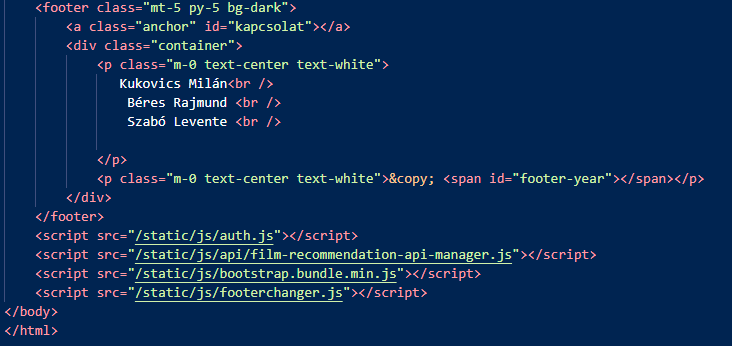
* **Előző gomb:** <a class="carousel-control-prev" href="#diasor" data-bs-slide="prev">
* **Következő gomb:** <a class="carousel-control-next" href="#diasor" data-bs-slide="next">
* **Ikonok:** carousel-control-prev-icon és carousel-control-next-icon az előző és következő gombok megjelenítésére.

A **felkapott filmek** számára dedikált tartalmi blokk:

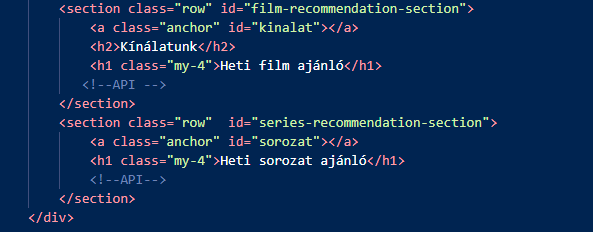
* **Fejléc:** <h2>Kínálatunk</h2> és <h1 class="my-4">Felkapott filmek</h1>
* **API integráció:** A szekció dinamizálható külső API-val, amely frissíti a filmajánlatokat.

A **felkapott sorozatok** számára fenntartott tartalmi blokk:

* **Fejléc:** <h1 class="my-4">Felkapott sorozatok</h1>
* **API támogatás:** A szekcióba külső API segítségével tölthetők be az aktuális legnépszerűbb sorozatok.



### Index.html



A **film-ajánlások** számára dedikált HTML blokk:

* **Azonosító:** <section class="row" id="film-recommendation-section">
* **Horgony (Anchor):** <a class="anchor" id="kinalat"></a> biztosítja a szakasz könnyű elérését.
* **Fejléc:** <h2>Kínálatunk</h2> és <h1 class="my-4">Heti film ajánló</h1>.
* **API integráció:** Egy megjegyzés (<!--API-->) jelzi, hogy a filmek listája dinamikusan tölthető be.

A **sorozatok** számára fenntartott tartalmi blokk:

* **Azonosító:** <section class="row" id="series-recommendation-section">
* **Horgony (Anchor):** <a class="anchor" id="sorozat"></a> segít az egyszerű navigációban.
* **Fejléc:** <h1 class="my-4">Heti sorozat ajánló</h1>.
* **API integráció:** A megjegyzés (<!--API-->) mutatja, hogy a sorozatok listája dinamikusan beilleszthető.



* A <section> elem tartalmazza az egész bejelentkezési felületet.
* A **container** osztály biztosítja a Bootstrap-alapú elrendezést.

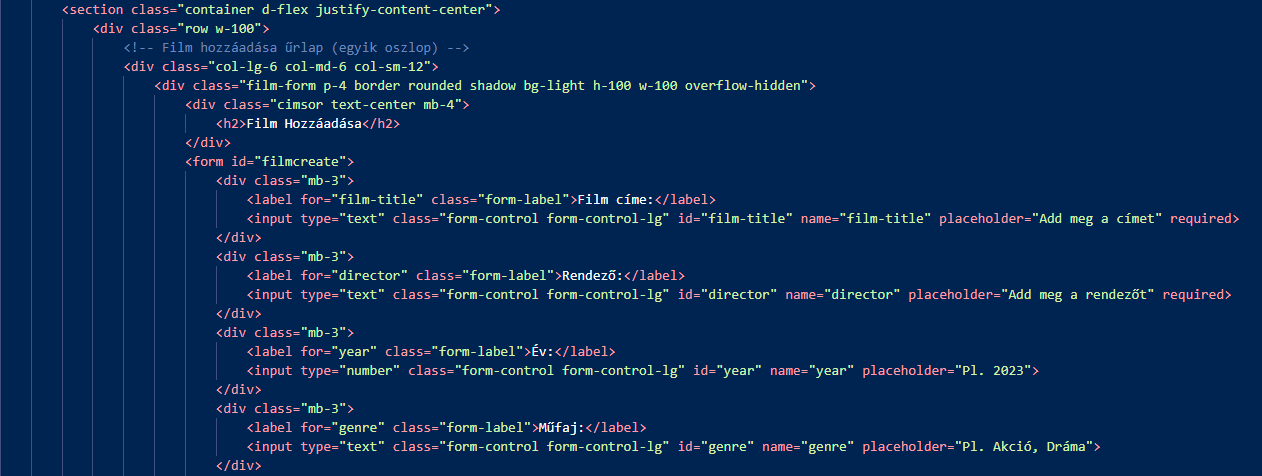
A **bejelentkezési űrlap** fő elemei:

* **Fejléc:** <h2>Bejelentkezés</h2> kijelzi a szakasz célját.
* **Űrlap (**<form id="login-form">**)**:
  + **Felhasználónév mező (**<input type="text">**)**
    - id="username"
    - name="username"
    - required attribútummal kötelezővé téve.
  + **Jelszó mező (**<input type="password">**)**
    - id="password"
    - name="password"
    - required attribútummal kötelezővé téve.
  + **Bejelentkezés gomb (**<button type="submit">**)**
    - id="auth-btn"
    - Szöveges tartalma: **Bejelentkezés**.

A bejelentkezési űrlap alatt található egy **regisztrációs link**, amely lehetővé teszi az új felhasználók számára a fiók létrehozását.

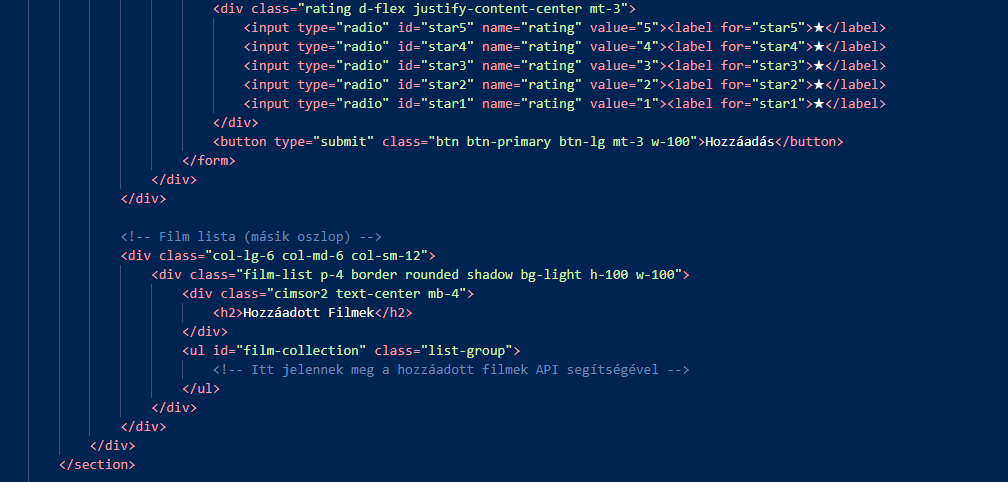
* id="reg" azonosítóval rendelkezik.
* A hivatkozás célja: registration.html.

### Naplo.html



Működés

* Az űrlap a filmcreate azonosítót használja, amely később JavaScript segítségével **feldolgozható**.
* A mezők **Bootstrap form-control osztályokkal** formázottak, hogy modern és reszponzív kinézetet kapjanak.
* Az adatok elküldéséhez egy JavaScript függvény szükséges, amely feldolgozza az űrlapot.



Működés

* Az értékelési mezők rádiógombként működnek, lehetővé téve az egyszerű **egycsillagos-ötcsillagos** értékelést.
* A filmgyűjteménybe kerülő filmek **API-hívásokkal** töltődnek be és jelennek meg.
* A film-collection lista dinamikusan frissíthető JavaScript segítségével.

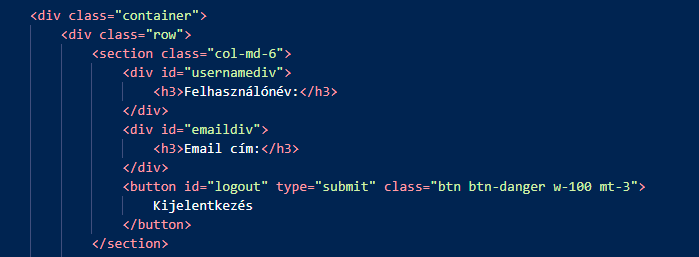
### Registration.html



Működés

* Az id="register-form" segítségével az űrlap **JavaScript alapú validálásra és adatkezelésre** is használható.
* A required attribútum megakadályozza az üres mezők beküldését.
* Az adatok elküldéséhez egy **backend API** szükséges az adatok mentésére és hitelesítésére.

### Profil.html



A profiladatokat egy <section> elem tartalmazza, amelyben két <div> található:

* usernamediv – A felhasználónév megjelenítésére.
* emaildiv – Az e-mail cím megjelenítésére. Ezen kívül egy <button> elem biztosítja a kijelentkezés lehetőségét.

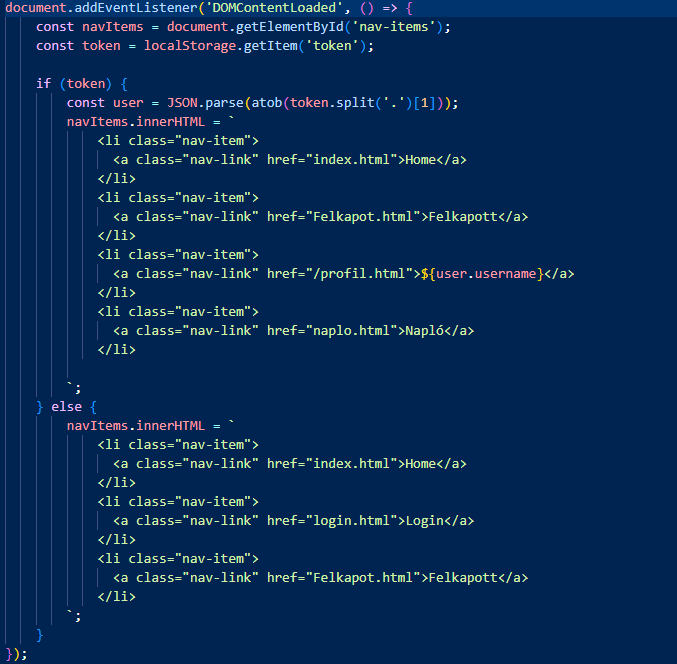


A profilbeállítások egy <section> elemben helyezkednek el, amely három különálló <div> elemet tartalmaz. Mindegyik div egy saját formot (<form>) tartalmaz, amely egy beviteli mezőt (<input>) és egy mentés gombot (<button>) foglal magába.

* **Beviteli mezők (**<input>**)**: A felhasználók itt adhatják meg az új adatokat (felhasználónév, email, jelszó).
* **Mentés gomb (**<button>**)**: Az új értékek mentéséhez szolgál, amikor a felhasználó rákattint.
* **Bootstrap osztályok** (btn btn-primary, form-control, mb-2, mt-3): Segítenek az esztétikai és strukturális kialakításban.

## Javascript

### Auth.js

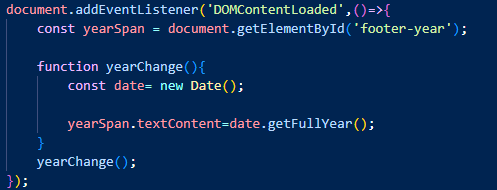


Működési elv

A kód a DOMContentLoaded eseményre figyel, ami biztosítja, hogy a DOM teljesen betöltődjön, mielőtt a navigáció módosításra kerül.

1. **Token ellenőrzése:**
   1. A localStorage.getItem('token') segítségével ellenőrzi, hogy van-e mentett token.
   2. Ha van token, akkor a felhasználót hitelesítettnek tekinti.
2. **Felhasználói adatok kinyerése:**
   1. A token egy Base64-kódolású JSON objektumot tartalmaz, amelyet atob() függvénnyel dekódol.
   2. Az objektum tartalmát JSON-ként értelmezi, hogy kinyerje a felhasználó nevét.
3. **Navigáció dinamikus beállítása:**
   1. Ha a felhasználó be van jelentkezve:
      1. Megjelenik a **profiloldal** elérhetősége, a felhasználó neével (pl. Profilom (user.username)).
      2. A "Bejelentkezés" menüpont nem jelenik meg.
   2. Ha a felhasználó nincs bejelentkezve:
      1. Megjelenik a **bejelentkezés** lehetősége.
      2. Nem látszik a profiloldal.

### Footer.js



Működési elv

1. **Betöltési esemény figyelése**
   1. A DOMContentLoaded eseményre figyel, ami biztosítja, hogy a DOM teljesen betöltődjön, mielőtt a kód futna.
2. **Elem kiválasztása**
   1. A document.getElementById('footer-year') segítségével megkeresi a lábjegyzet év megjelenítésére szánt HTML-elemet.
3. **Dátum kezelése**
   1. A new Date().getFullYear() függvény segítségével lekéri az aktuális évet.
4. **Dátum beállítása**
   1. A kiválasztott elem textContent tulajdonságát az aktuális évre állítja.

## CSS

### auth.css

  
Áttekintés

Ez a CSS-kód egy autentikációs űrlap (auth-form) stílusát határozza meg. Tartalmaz formázási szabályokat a címkékre (label), beviteli mezőkre (input), gombokra (button), valamint a beviteli mezők aktív állapotára is.

Stíluselemek

1. Címkék (.auth-form label)

A címkék megjelenítését szabályozza:

* **display: block** - Blokkszintű elemként jelenik meg.
* **font-size: 1.5rem** - A betűméret 1.5-szöröse az alap betűméretnek.
* **margin-bottom: 10px** - 10 pixel margó alul, hogy elkülönüljön a beviteli mezőktől.

2. Beviteli mezők (.auth-form input)

A beviteli mezők alapértelmezett stílusa:

* **width: 80%** - Az elem a szülője 80%-át foglalja el.
* **padding: 15px** - 15 pixeles belső tér.
* **margin-bottom: 20px** - 20 pixeles alsó margó.
* **border: none** - Alapértelmezett szegély eltávolítása.
* **border-bottom: 2px solid #000** - Csak az alsó szegély marad meg, 2 pixel vastagon, fekete színnel.
* **font-size: 1.2rem** - A betűméret 1.2-szöröse az alap betűméretnek.
* **background: transparent** - Átlátszó háttér.

3. Beviteli mezők (fókuszállapot) (.auth-form input:focus)

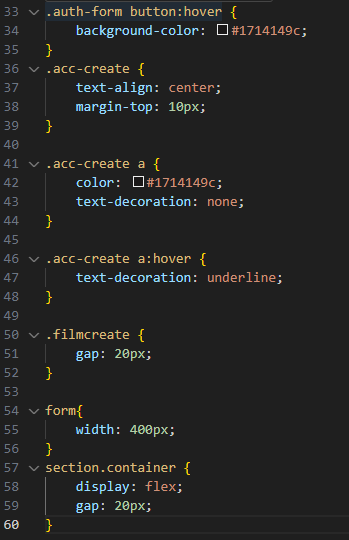
Ha a felhasználó a mezőre kattint vagy belekattint, a következő változások lépnek életbe:

* **outline: none** - Az alapértelmezett keret kikapcsolása.
* **border-bottom: 2px solid gray** - Az alsó szegély színe szürkére változik.

4. Gombok (.auth-form button)

A gombok stílusát határozza meg:

* **width: 80%** - A gomb a szülő elem 80%-át foglalja el.
* **padding: 15px** - 15 pixeles belső tér.
* **border: none** - Alapértelmezett szegély eltávolítása.
* **background-color: #1714149c** - Sötét színű háttér, átlátszósággal.
* **color: #fff** - Fehér betűszín.
* **font-size: 1.4rem** - A betűméret 1.4-szerese az alap betűméretnek.
* **cursor: pointer** - Az egérmutató kattintásra alkalmas ikonra változik.
* **transition: background-color 0.3s ease** - A háttérszín 0.3 másodperc alatt változik meg, lágy átmenettel.



1. auth-form button:hover

Ez a szakasz meghatározza az auth-form osztályú formon található gombok lebegő állapotának stílusát. Ha az egér a gomb fölé kerül, a háttérszín átvált a megadott #1714149c értékre, amely átlátszó hatású.

2. acc-create

Ez az osztály a center-be helyezést és a 10px-es margót alkalmazza a körülötte lévő tartalomhoz.

* **Specifikus elem:** A belső linkek színe szintén #1714149c, és alapértelmezésben nincs aláhúzva, azonban lebegéskor aláhúzás jelenik meg.

3. filmcreate

Ez az osztály 20px-es rést állít be az elemek között, a layout rendszerezését segítve.

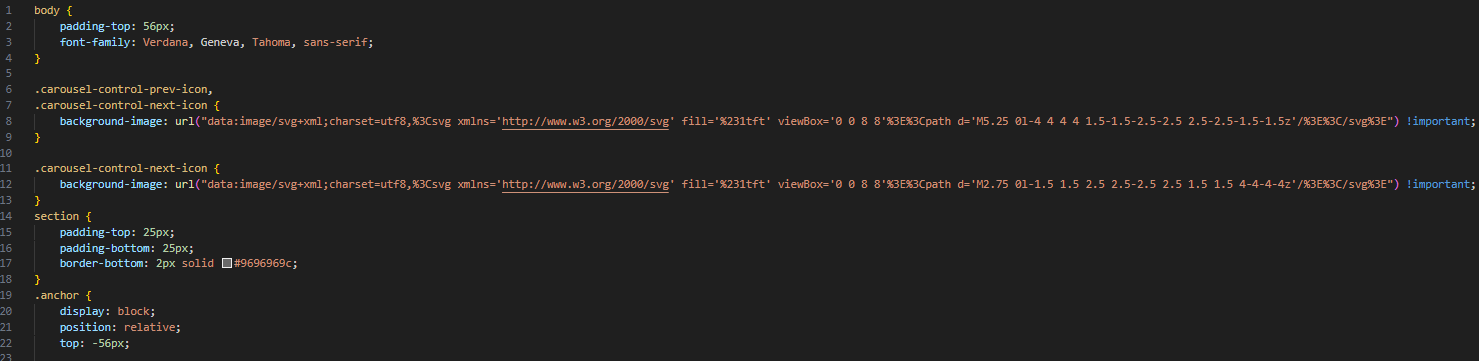
4. form

Ez az osztály minden form elem szélességét 400px-re korlátozza.

5. section.container

A container osztályú section elem flexbox alapú layoutot használ, és 20px-es rést állít be az alatta lévő elemek között.

### Main.css

  
1. body

* **Stílusok:**
  + 56px felső margót ad hozzá, valószínűleg egy fix fejléc helyének biztosítására.
  + A betűtípus Verdana, tartalékként pedig Geneva, Tahoma és sans-serif.

2. carousel-control-prev-icon és carousel-control-next-icon

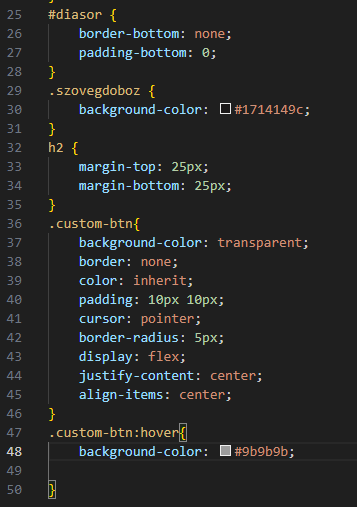
* **Stílusok:**
  + Ezek az elemek egyedi navigációs ikonokat jelenítenek meg a carouselben, SVG adat URI-k használatával háttérképként.
  + Az !important használata biztosítja, hogy ezek a stílusok felülírják a többit.

3. section

* **Stílusok:**
  + 25px-es párnázást ad hozzá felül és alul a tartalom távolságának biztosítására.
  + 2px-es #969696 színű alsó szegélyt használ a szekciók vizuális elválasztására.

4. anchor

* **Stílusok:**
  + Blokk szintűvé teszi az elemeket a rugalmasság érdekében.
  + -56px pozíciót állít be felül, így biztosítja, hogy sima görgetéskor az elem egy kicsit feljebb kerüljön.



1. #diasor

* **Stílusok:**
  + Eltávolítja az alsó keretet (border-bottom: none).
  + 0 pixel alsó párnázást állít be (padding-bottom: 0).

2. .szovegdoboz

* **Stílusok:**
  + Háttérszínt állít be: #171449c.

3. h2

* **Stílusok:**
  + 25 pixel margót állít be felül és alul (margin-top: 25px, margin-bottom: 25px).

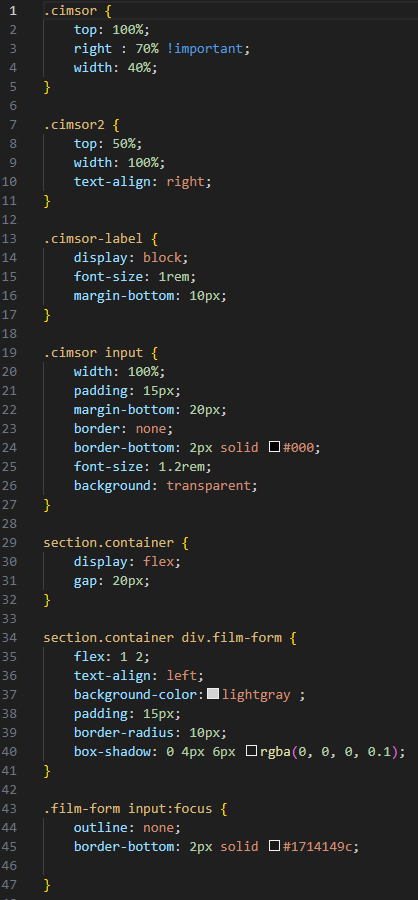
4. .custom-btn

* **Stílusok:**
  + Áttetsző háttérszínt és körvonalat alkalmaz (background-color: transparent, border: none).
  + Örökölhető színt biztosít (color: inherit).
  + Beállítja a párnázást (padding: 10px 10px), valamint hozzáad egy kurzor mutatót (cursor: pointer).
  + Lekerekíti a gombok sarkait (border-radius: 5px) és flexboxot használ a tartalom középre helyezésére (display: flex, justify-content: center, align-items: center).

5. .custom-btn:hover

* **Stílusok:**
  + Módosítja a gomb háttérszínét lebegéskor (background-color: #9b9b9b).

### Naplo.css



1. .cimsor

* A fejlécet a képernyő felső 100%-ra helyezi.
* Az elem jobb oldalát 70%-kal tolja ki !important használatával.
* 40%-os szélességet ad az elemnek.

2. .cimsor2

* Az elemet a képernyő közepére helyezi.
* 100%-os szélességet alkalmaz.
* Jobbra igazítja a szöveget.

3. .cimsor-label

* Blokk szintű megjelenítést használ (display: block).
* A betűméretet 1 rem-re állítja.
* 10px margót biztosít az alján.

4. .cimsor input

* Teljes szélességet biztosít a bemeneti mezőnek (width: 100%).
* Párnázást ad hozzá (padding: 15px), és margót az aljához (margin-bottom: 20px).
* Eltávolítja a keretet, de alsó szegélyt ad hozzá (border-bottom: 2px solid #000).
* 1.2 rem betűméretet alkalmaz.
* Átlátszó hátteret használ (background: transparent).

5. section.container

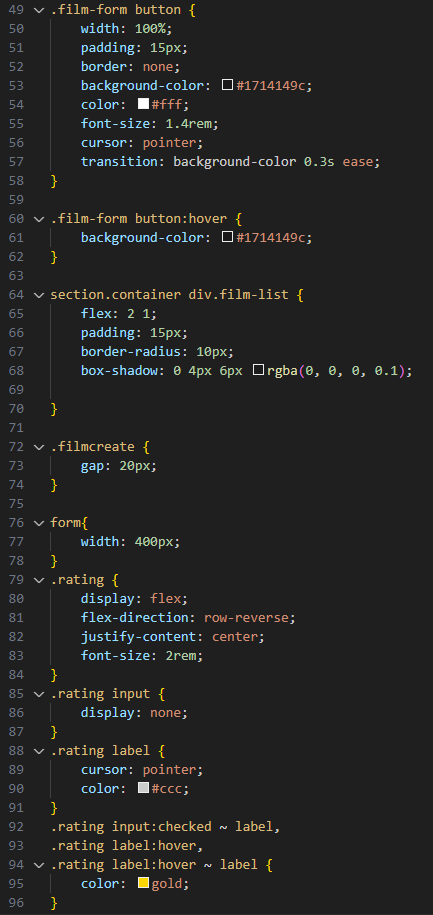
* Rugalmas elrendezést alkalmaz flexbox segítségével (display: flex).
* 20px-es rést ad az elemek között (gap: 20px).

6. section.container div.film-form

* Az elemek közötti egyensúlyért felel (flex: 1 2).
* Jobbra igazítja a szöveget, világosszürke háttérrel.
* Párnázást (padding: 15px), lekerekített sarkokat (border-radius: 10px) és árnyékot (box-shadow) ad hozzá.

7. .film-form input:focus

* Eltávolítja az outline állapotot, és módosítja az alsó szegély színét fókuszban (border-bottom: 2px solid #171449c).



1. .film-form button

* Teljes szélességet foglal el (width: 100%).
* 15 pixel párnázás (padding: 15px).
* Eltávolított keret (border: none).
* Áttetsző háttérszín használata (background-color: #1714149c).
* Fehér betűszín (color: #fff).
* Betűméret: 1.4 rem (font-size: 1.4rem).
* Kézkurzor megjelenítése (cursor: pointer).
* Hátérszín átmenetes változása (transition: background-color 0.3s ease).

2. .film-form button:hover

* Lebegéskor a háttérszín: #1714149c.

3. section.container div.film-list

* A rugalmas méretezés két egy az egyhez arányt használ (flex: 2 1).
* 15 pixel párnázás (padding: 15px).
* Lekerekített sarkok (border-radius: 10px).
* Árnyékolás: box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1).

4. .filmcreate

* 20 pixel rést állít be az elemek között (gap: 20px).

5. form

* Szélesség: 400px (width: 400px).

6. .rating

* Flexbox alapú elrendezés fordított sorrendben (flex-direction: row-reverse).
* Középre igazított elemek (justify-content: center).
* Betűméret: 2 rem (font-size: 2rem).

7. .rating input

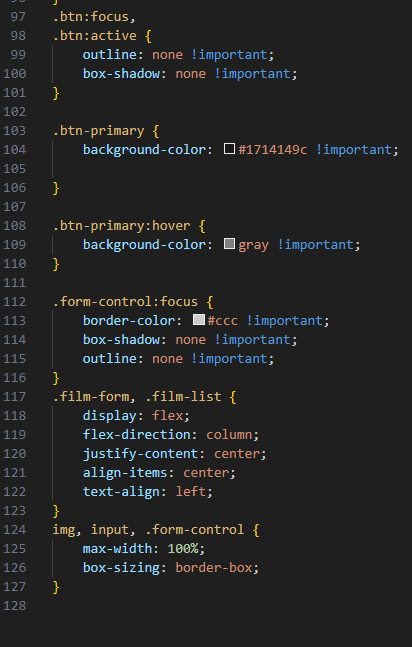
* Nem jeleníti meg az input mezőket (display: none).

8. .rating label

* Kézkurzor alkalmazása (cursor: pointer).
* Alapértelmezett szín: szürke (color: #ccc).

9. .rating input:checked ~ label, .rating label:hover, .rating label:hover ~ label

* Arany szín a kijelölt vagy lebegő elemekhez (color: gold).



1. .btn:focus, .btn:active

* A fókuszált és aktív gombok körvonala eltávolítva (outline: none !important) és árnyékmentes megjelenés biztosított (box-shadow: none !important).

2. .btn-primary

* A gomb háttérszíne #1714149c.

3. .btn-primary:hover

* Lebegéskor a háttérszín szürkévé változik (background-color: gray !important).

4. .form-control:focus

* Fókuszáláskor a keretszín #ccc lesz (border-color: #ccc !important), eltávolított árnyék és körvonal.

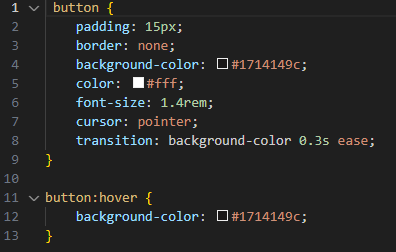
5. .film-form, .film-list

* Flexbox alapú elrendezés. A tartalom középre igazítva (justify-content: center; align-items: center), balra igazított szöveg (text-align: left).

6. img, input, .form-control

* Maximális szélesség biztosítva a tartalomhoz igazítva (max-width: 100%), és box-model meghatározás (box-sizing: border-box).

### Profil.css



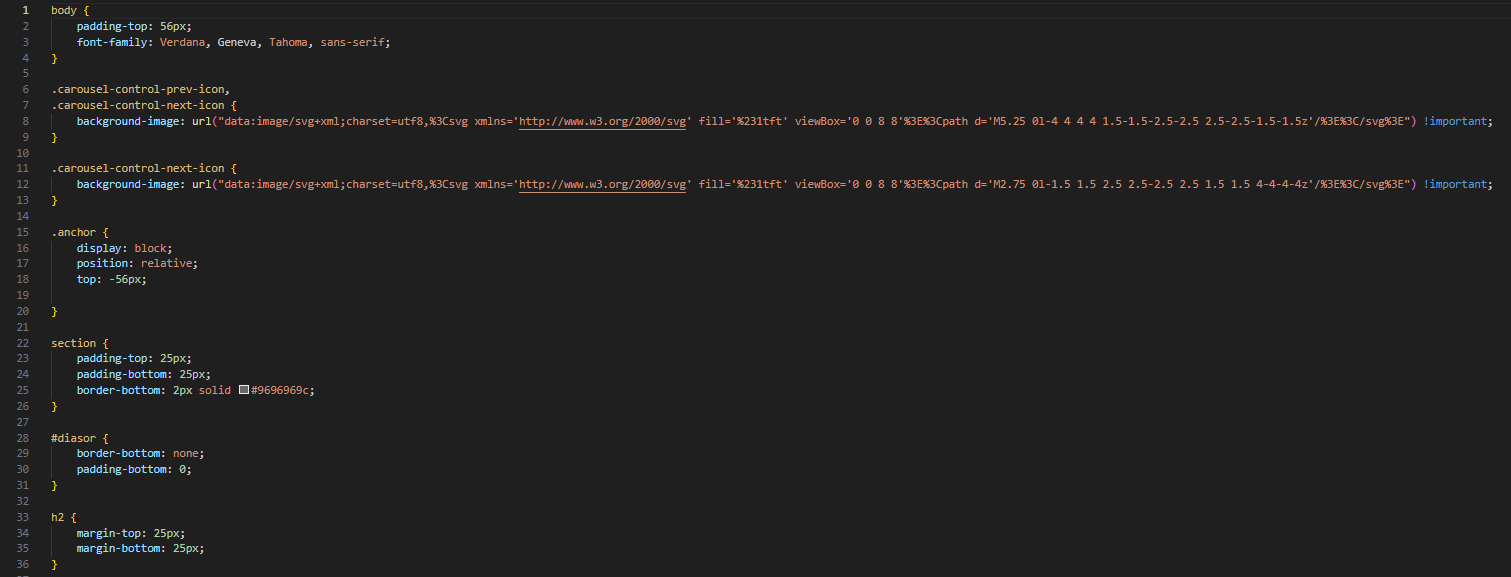
1. button

* 15 pixel párnázást alkalmaz (padding: 15px).
* A gomb kerete eltávolítva (border: none).
* Átlátszó háttérszínt használ (background-color: #1714149c).
* Betűszíne fehér (color: #fff).
* Betűmérete 1.4 rem (font-size: 1.4rem).
* Mutató kurzor beállítása (cursor: pointer).
* Háttérszín sima átmenetes változása (transition: background-color 0.3s ease).

2. button:hover

* Lebegéskor a háttérszín: #1714149c.

### styles.css



1. body

* Felső párnázás: 56px.
* Betűtípus: Verdana, alternatívákkal Geneva, Tahoma, sans-serif.

2. carousel-control-prev-icon és carousel-control-next-icon

* Egyedi SVG ikonokat használ háttérképként.
* Az !important deklaráció biztosítja, hogy ezek a stílusok felülírják az esetlegesen meglévő egyéb szabályokat.

3. anchor

* Blokk szintű elem (display: block).
* Relatív pozícióval rendelkezik, -56px felső eltolással.

4. section

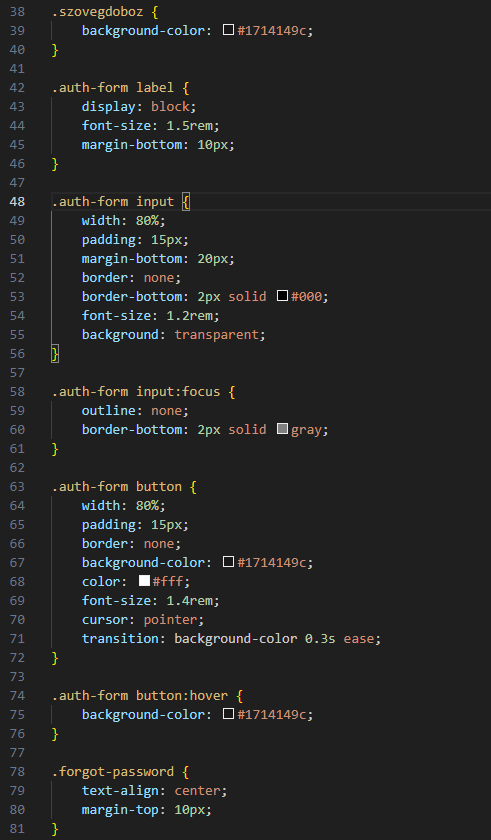
* Felső és alsó párnázás: 25px.
* Alsó szegély: 2px solid #969696.

5. #diasor

* Az alsó szegély eltávolítva (border-bottom: none).
* Alsó párnázás: 0.

6. h2

* Felső és alsó margó: 25px.



1. .szovegdoboz

* Háttérszín: #1714149c.

2. .auth-form label

* Blokk szintű megjelenítés (display: block).
* Betűméret: 1.5rem.
* Alsó margó: 10px.

3. .auth-form input

* Szélesség: 80%.
* Párnázás: 15px.
* Alsó margó: 20px.
* Keret: Nincs (border: none).
* Alsó szegély: 2px solid #000.
* Betűméret: 1.2rem.
* Háttér: Átlátszó (background: transparent).

4. .auth-form input:focus

* Outline: Nincs.
* Alsó szegély színe: 2px solid gray.

5. .auth-form button

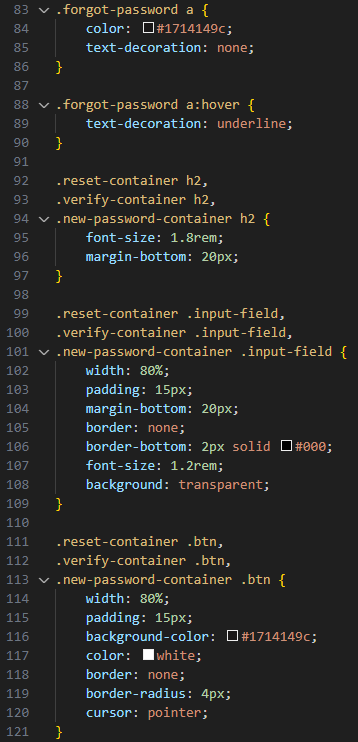
* Szélesség: 80%.
* Párnázás: 15px.
* Keret: Nincs (border: none).
* Háttérszín: #1714149c.
* Betűszín: #fff.
* Betűméret: 1.4rem.
* Kurzort állít: Mutató (cursor: pointer).
* Átmeneti effekt: Háttérszín változása 0.3s ease.

6. .auth-form button:hover

* Háttérszín: #1714149c.

7. .forgot-password

* Szöveg igazítás: Középre (text-align: center).
* Felső margó: 10px.



1. .forgot-password a

* Szövegszín: #171449c.
* Szövegdekoráció: Nincs (text-decoration: none).

2. .forgot-password a:hover

* Szövegdekoráció: Aláhúzás (text-decoration: underline).

3. h2 (reset-container, verify-container, new-password-container)

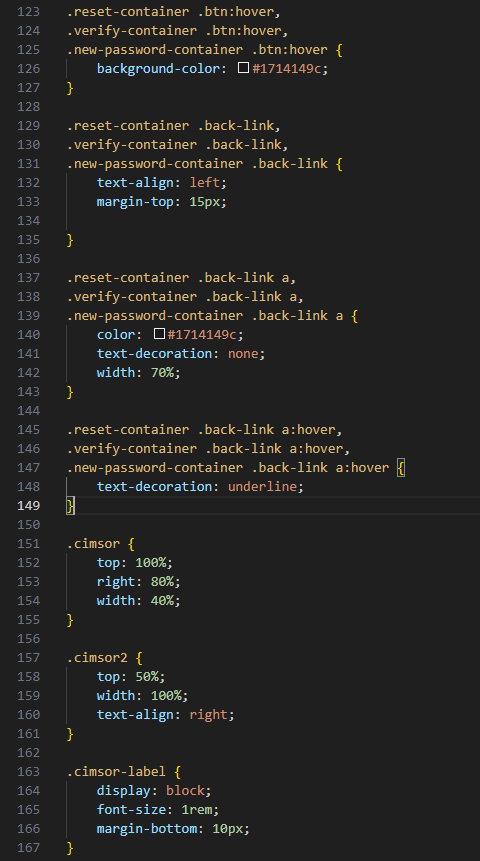
* Betűméret: 1.8rem.
* Alsó margó: 20px.

4. .input-field (reset-container, verify-container, new-password-container)

* Szélesség: 80%.
* Párnázás: 15px.
* Alsó margó: 20px.
* Keret: Nincs (border: none).
* Alsó szegély: 2px solid #000.
* Betűméret: 1.2rem.
* Háttér: Átlátszó (background: transparent).

5. .btn (reset-container, verify-container, new-password-container)

* Szélesség: 80%.
* Párnázás: 15px.
* Háttérszín: #171449c.
* Szövegszín: Fehér (color: white).
* Keret: Nincs (border: none).
* Lekerekített sarkok: 4px.
* Kézkurzor megjelenítése (cursor: pointer).



1. .reset-container .btn:hover, .verify-container .btn:hover, .new-password-container .btn:hover

* Háttérszín: #1714149c.

2. .reset-container .back-link, .verify-container .back-link, .new-password-container .back-link

* Szöveg igazítása: Balra (text-align: left).
* Felső margó: 15px.

3. .reset-container .back-link a, .verify-container .back-link a, .new-password-container .back-link a

* Szövegszín: #1714149c.
* Szövegdekoráció: Nincs (text-decoration: none).
* Szélesség: 70%.

4. .reset-container .back-link a:hover, .verify-container .back-link a:hover, .new-password-container .back-link a:hover

* Szövegdekoráció: Aláhúzás (text-decoration: underline).

5. .cimsor

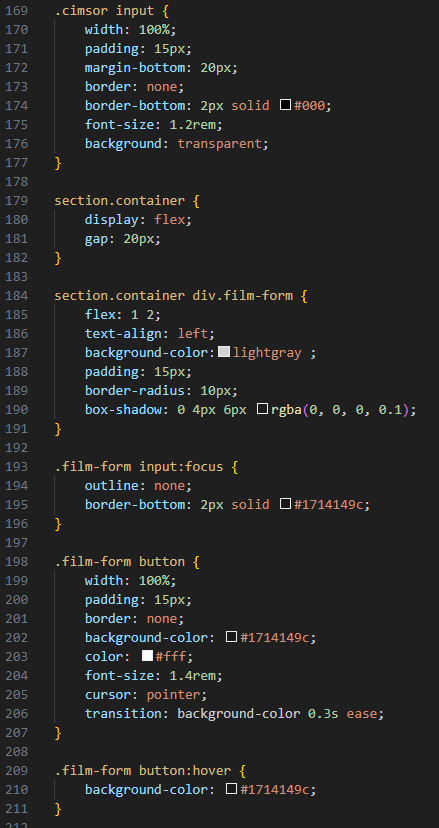
* Felső pozíció: 100% (top: 100%).
* Jobb pozíció: 80% (right: 80%).
* Szélesség: 40%.

6. .cimsor2

* Felső pozíció: 50% (top: 50%).
* Szélesség: 100%.
* Szöveg igazítása: Jobbra (text-align: right).

7. .cimsor-label

* Blokk szintű elem (display: block).
* Betűméret: 1rem.
* Alsó margó: 10px.



1. .film-form input

* Szélesség: 100%.
* Párnázás: 15px.
* Alsó margó: 20px.
* Keret: Nincs (border: none).
* Alsó szegély: 2px solid #000.
* Betűméret: 1.2rem.
* Háttér: Átlátszó (background: transparent).

2. .film-form input:focus

* Körvonal: Nincs (outline: none).
* Alsó szegély színe: 2px solid #1714149c.

3. .film-form button

* Szélesség: 100%.
* Párnázás: 15px.
* Keret: Nincs (border: none).
* Háttérszín: #1714149c.
* Betűszín: Fehér (color: #fff).
* Betűméret: 1.4rem.
* Kurzormutató: Kéz (cursor: pointer).
* Áttűnés a háttérszínre: 0.3s ease (transition: background-color 0.3s ease).

4. .film-form button:hover

* Háttérszín: #1714149c.

5. section.container

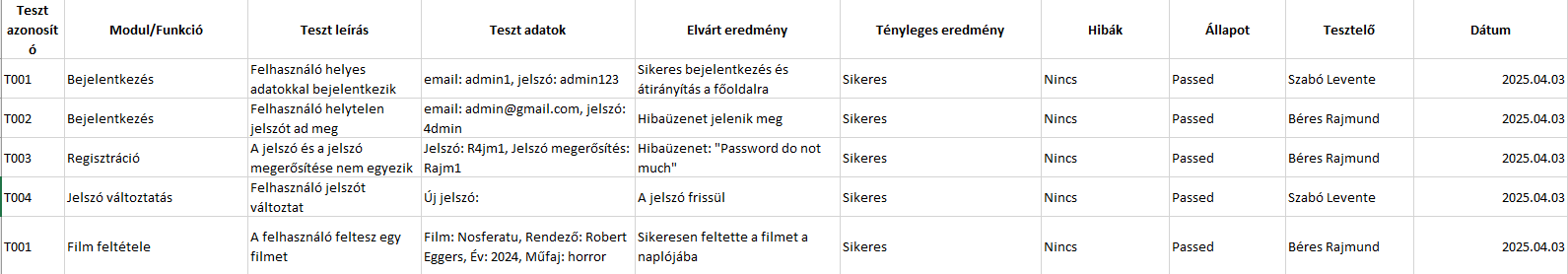
* Flexbox-alapú elrendezés (display: flex).
* Elemek közötti rés: 20px (gap: 20px).

6. section.container div.film-form

* Rugalmas arány: flex: 1 2.
* Szöveg igazítása: Balra (text-align: left).
* Háttérszín: Világos szürke (background-color: lightgray).
* Párnázás: 15px.
* Lekerekített sarkok: 10px (border-radius: 10px).
* Árnyék: 0 4px 6px rgba(0, 0, 0, 0.1).



# Tesztelés



Tesztelési Struktúra

A tesztelési táblázat a következő oszlopokat tartalmazza:

* **Teszt azonosító**: Egyedi azonosító minden teszthez.
* **Modul/Funkció**: A tesztelt rendszerkomponens neve.
* **Teszt leírás**: Rövid leírás arról, hogy mit tesztelünk.
* **Teszt adatok**: Az input adatok, amelyeket a tesztelés során használunk.
* **Elvárt eredmény**: A teszt sikerességének kritériuma.
* **Tényleges eredmény**: A teszt futtatásakor kapott eredmény.
* **Hibák**: Az esetleges eltérések és problémák.
* **Állapot**: A teszt sikere (Passed) vagy sikertelensége (Failed).
* **Tesztelő**: Az a személy, aki végrehajtotta a tesztet.
* **Dátum**: A teszt végrehajtásának időpontja.