**Szakképesítés neve:** ……………………………………

**OKJ száma:** …………………

**SZAKDOLGOZAT**

SZAKDOLGOZAT CÍME

Témavezető:  
**Beke Béla**

Készítette:  
**Pataki Dávid Ferenc**

Debrecen  
2023

# Tartalomjegyzék

[Tartalomjegyzék 2](#_Toc130934803)

[1 Bevezetés 4](#_Toc130934804)

[1.1 Miért ezt választottam? 4](#_Toc130934805)

[1.2 Köszönetnyilvánítás 4](#_Toc130934806)

[2 Felhasználói dokumentáció 5](#_Toc130934807)

[2.1 Rendszerkövetelmény 5](#_Toc130934808)

[2.2 Weboldal használata 5](#_Toc130934809)

[3 Fejlesztői dokumentáció 6](#_Toc130934810)

[3.1 Használt npm csomagok 6](#_Toc130934811)

[3.2 Telepítés és indítás 7](#_Toc130934812)

[3.3 Backend 7](#_Toc130934813)

[3.3.1 Tervezési minta (Architektúra) 7](#_Toc130934814)

[3.4 Adatbázis 8](#_Toc130934815)

[3.4.1 user tábla 8](#_Toc130934816)

[3.4.2 auth tábla 9](#_Toc130934817)

[3.4.3 follow tábla 9](#_Toc130934818)

[3.4.4 permission tábla 9](#_Toc130934819)

[3.4.5 review tábla 9](#_Toc130934820)

[3.4.6 movie tábla 10](#_Toc130934821)

[3.4.7 movie\_genre tábla 10](#_Toc130934822)

[3.4.8 genre tábla 10](#_Toc130934823)

[3.5 Algoritmusok 10](#_Toc130934824)

[3.5.1 Regisztráció 10](#_Toc130934825)

[3.5.2 Belépés 11](#_Toc130934826)

[3.5.3 JsonWebToken / JWT 11](#_Toc130934827)

[3.5.4 [TODO: Képfeltötés] 11](#_Toc130934828)

[3.6 Tesztdokumentáció 11](#_Toc130934829)

[3.7 Fejlesztési lehetőségek 11](#_Toc130934830)

[3.8 Fejlesztői környezet 11](#_Toc130934831)

[4 Összefoglalás 12](#_Toc130934832)

[5 Irodalomjegyzék 13](#_Toc130934833)

# Bevezetés

A záródolgozatom témája egy filmeket értékelő és ajánló platform, ahol a felhasználók bejegyzéseket írhatnak megtekintett filmjeikről, hogy megosszák élményüket másokkal is. A felhasználók követhetnek más fiókokat, hogy könnyedén lássák azok értékeléseit.  
A weblap funkciói közt említést érdemel még a filmek keresése név, értékelések és műfajai alapján.

## Miért ezt választottam?

Ezt a témát választottam, mert szeretek filmeket nézni és érdekesnek találtam a filmekről való vélemények összegyűjtését és megosztását. Továbbá úgy véltem, egy ilyen weboldal hasznos lehet mások számára is, hogy segítsen nekik a film választásában.

## Köszönetnyilvánítás

# Felhasználói dokumentáció

## Rendszerkövetelmény

## Weboldal használata

# Fejlesztői dokumentáció

A weboldal fejlesztéséhez [Node.js](https://nodejs.org/en/)-t alkalmaztam. Azért ezt használtam mivel a JavaScript programozási nyelvet jobban elsajátítottam szabad időmben, mint más interaktív weboldal készítésére használt nyelveket, például a PHP-t. Abban is előnyösebb nekem ez a környezet, mert a Frontend - Backend kódot jobban tudom szeparálni.

A JavaScript programozási nyelvet kiegészítem a **TypeScript** nevezetű szintaxis és statikus típus ellenőrző készlettel, a fölösleges hibák elkerülése érdekében.

A Node.js-en belül a [Next.js](https://nextjs.org/) **keretrendszert** használom, ami a React frontend könyvtárat kiegészíti egy Express-hez hasonló backend RESTful API készítő csomaggal.

A [React](https://reactjs.org/) a *Meta* által létrehozott JavaScript könyvtár, mellyel könnyen lehet készíteni interaktív felhasználói felületeket.

Adatbázisnak a MySQL alapú **MariaDB**-t alkalmaztam.

## Használt npm csomagok

A fentieken kívül használtam még másmilyen különböző úgynevezett npm csomagokat kisebb problémák megoldására:

* bcrypt: Egy Kriptografikus könyvtár, amely a jelszók biztonságosabb eltárolása érdekében használok.
* jsonwebtoken: Röviden JWT, egy token alapú autentikációs módszer, amely a session-el ellentétben a kliensen tárolja a bejelentkezett fiók adatait. A szerver csak egy kriptografikus kulcs segítségével validálja a klienstől kapott tokent.
* cookies-next: Next.js-ben nincs alapból cookie-k módosítására lehetőség, de van erre kifejlesztett csomag a készítőktől.
* mysql2: Az adatbázis kapcsolat létrehozására és felhasználására használt könyvtár
* multiparty: Képek feltöltéséhet űrlapot/form-ot kell alkalmazunk, ennek a kódolása JSON helyett multipart/form-data, amelyet Next.js alapból nem képes fordítani, ezért használnom kell egy külön csomagot.

## Telepítés és indítás

A project optimális futása érdekében erősen ajánlott a [Fejlesztői környezet](#_Fejlesztői_környezet)-ben megjelölt Node.js verzió használata. Régebbi, ritka esetben akár új verziók is, képesek előre nem látható problémákat okozni, amelyek akadályozhatják a program futását.

A /next.config.js állományban érdemes átírni az adatbázis hitelesítő adatait.

databaseHost: elérés cím  
databaseUser: felhasználói fiók  
databasePassword: felhasználó jelszava  
databaseDatabase: adatbázis neve

A /scripts mappában található telepítési és indítási scriptek Linux-ra és Windows-ra, ha a parancssor túl rémisztő.

* ./install: telepíti a npm csomagokat, létrehozza az adatbázist és dinamikusan generál titkos szerver oldali tokeneket a kriptografikus műveleteknek.
* ./start-dev: Elindítja a fejlesztői környezetet, ebben a módban a projectben történő változtatások egyből megváltoznak a weboldalon is, viszont ez fölösleges rendszer erőforrásokat és optimalizálatlan kódot futtat ezért a weboldal lassabbnak tűnhet.
* ./build & ./start-prod: Ezzel a két scripttel lehet egy optimalizáltabb környezetet létrehozni a projectnek, viszont minden változtatás után újra le kell futtatni a építés parancsot.

## Backend

### Tervezési minta (Architektúra)

[TODO: Kép]

A backend kód struktúrája a Model-Service-Controller (Modell-Szolgáltatás-Vezérlő) architektúrán (röviden MSC architektúra) alapul. Minden API kérés ezen a „vezetéken” megy át. Három fontos részből áll:

1. Controller: A beérkező kéréseket irányítja a megfelelő irányba, ezen a rétegen történik a http kérésének szintaktikai ellenőrzése, és a http válaszok visszaküldése.
2. Service: Ezen a rétegen történik a komplexebb ellenőrzések (pl. felhasználó név létezik e már az adatbázisban), ez a réteg konkrétan nem foglalkozik az adatbázissal, sem a http kapcsolatokkal, de viszont egy fontos átmeneti réteg a következő rétegnek.
3. Model: Feladata az adatok lekérdezése az adatbázistól és annak értelmezése.

Ahogy lehet látni minden rétegnek meg van a saját felelőssége. Kisebb projektekben túlzás ilyen architektúrákat alkalmazni, lassítja a fejlesztési sebességét, viszont ezzel a tervezési mintával javítható a szoftver skálázhatósága és karbantarthatósága.

## Adatbázis



### user tábla

Ez a tábla a felhasználók alap adatait tárolja.

* **id**: Egyedi elsődleges kulcs
* **name**: Egyedi szöveg típus, a felhasználónevet tárolja
* **created\_at**: Dátum típus, a felhasználó regisztálásának dátumát tárolja
* **description**: Szöveg típus, nem kötelező, felhasználó által beálított leírás vagy szöveg
* **picture\_path**: Szöveg típus, nem kötelező, felhasználó profilképének a szerveren található file neve
* **permission\_id**: Idegen kulcs a permisson táblába, a felhasználó jogát határozza meg

### auth tábla

Ez a tábla tartalmazza a felhasználók authentikációs adatait.

* **id**: Egyedi elsődleges kulcs
* **user\_id**: Egyedi idegen kulcs a user táblába, mely felhasználó belépési adatai az egyed
* **email**: Egyedi szöveg típus, a felhasználó email címe
* **password\_hash**: Fix hosszúságú szöveg, a jelszavakat soha nem tároljuk egy per egy az adatbázisban, hanem a jelszót odaadjuk egy egy-irányú enkriptációs algoritmusnak és csak az eredményt tároljuk.

A jelszó tárolásáról a [Algoritmusok](#_Algoritmusok)(TODO) fejlécben többet megtudhatunk.

### follow tábla

Követések feljegyzésére szolgáló tábla.

* **whoUserID**: Idegen kulcs a user táblába, melyik felhasználó követ
* **whomUserID**: Idegen kulcs a user táblába, melyik felhasználót követi

### permission tábla

A különböző féle jogi szintek tárolására szolgáló tábla. Jelenleg 3 különböző szintet különböztünk meg, Felhasználó, Moderátor, Admin.

* **id**: Egyedi elsődleges kulcs
* **name**: Egyedi szöveg típus, a jog neve
* **level**: Szám típus, amely 0-255 között meghatározza a jog „szintjét”, minél magasabb annál több képessége van annak a jognak

### review tábla

A felhasználók által generált értékelések tárolására szolgáló tábla

* **id**: Egyedi elsődleges kulcs
* **author\_id**: Idegen kulcs a user táblába, az értékelés szerzőjét határozza meg
* **movie\_id**: Idegen kulcs a movie táblába, mely filmnek az értékelése
* **rating**: Szám típus, 1-10 közötti érték, amely meghatározza hogyan értékelte a filmet egy tízes skálán
* **description**: Szöveg típus, nem kötelező, felhasználó által beálított leírás vagy szöveg
* **create\_date**: Dátum + idő típus, értékelés létrehozásának idejét tárolja

### movie tábla

A filmek tárolására szolgál

* **id**: Egyedi elsődleges kulcs
* **name**: Egyedi szöveg típus, a film nevét tárolja
* **release\_date**: Év típus, a film kijövetelének évét tárolja
* **image\_path**: Szöveg típus, nem kötelező, a film borítóképének a szerveren található file neve

### movie\_genre tábla

Mivel egy filmnek lehet több műfaja és egy műfajnak lehet több filmje, kell egy összekapcsolási tábla, ez az

* **movie\_id**: Idegen kulcs a movie táblába, melyik filmnek a támáját állítjuk most
* **genre\_id**: Idegen kulcs a genre táblába, milyen műfajt állítunk állítunk be

### genre tábla

Film műfajok tárolására szolgáló tábla

* **id:** Egyedi elsődleges kulcs
* **name**: Egyedi szöveg típus, műfaj nevét határozza meg

## Algoritmusok

[TODO: Kép]

### Regisztráció

Az új felhasználó adatait a **/auth** oldalon adjuk meg. Amikor a **Regisztráció** gombra kattintunk, ellenőrizzük a szövegmezőket annak érdekében, hogy helyesen legyenek formázva: pl. a felhasználónév hossza 5 és 32 karakter között legyen, az e-mail cím pedig kövesse az *„username@server.domain”* formátumot stb. Ha bármilyen hiba merül fel, értesítjük a felhasználót. Ellenkező esetben HTTP kérést küldünk a **/api/user** címre.

A backend-en a Controller újra ellenőrzi a kérés testének formázását. Ezután a Service réteg ellenőrzi, hogy van-e már hasonló felhasználónév vagy e-mail cím az adatbázisban. Ha igen, akkor értesítjük a felhasználót, ha nem, akkor folytatjuk a Model réteggel. Itt a megadott jelszót **bcrypt** segítségével kódoljuk. Ezután tároljuk a **user** és az **auth** adatbázis táblában a felhasználói és az azonosítási adatokat. Amennyiben minden sikeres a felhasználót átirányítjuk a belépési oldalra.

### Belépés

Belépési adatokat a **/auth** oldalon adhatjuk meg. Itt is ellenőrizzük a felhasználónév és jelszó hosszát, hiba esetén szólunk a felhasználónak. Ezután egy HTTP kérést küldünk a **/api/auth** címre.

A backend-en ellenőrzés után, megnézzük, hogy létezik-e ilyen nevű felhasználó. Ha igen akkor **bcrypt** segítségével összehasonlítjuk az adatbázisban tárolt hash-et a felhasználó által megadott jelszóval. Hibás adatok esetén értesítjük a felhasználót. Amennyiben megfelel az adat, visszaküldünk egy **JWT** token-t, amivel a böngésző a későbbi kéréseknél be tudja bizonyítani, hogy melyik felhasználó küldte a kérelmet. Ezt a sütik-ben tároljuk.

### JsonWebToken / JWT

Ahogyan az előző paragrafusban említettem, az authentikációhoz JSON Web Token-eket alkalmazunk. A belépés-t követően a böngésző sütik-ben tároljuk, hogy későbbi kéréseknél automatikusan elküldjük. A Session-el elletétben a Tokeneket nem kell adatbázisban tárolni, helyette a felhasználó tárolja és egy enkriptácíós algoritmussal bizonyítjuk be, hogy az a token tényleg valódi.

### [TODO: Képfeltötés]

## Tesztdokumentáció

## Fejlesztési lehetőségek

* Bejegyzésekre válaszolás, nem csak a filmekre lehet válasz írni, hanem a bejegyzésekre is.
* Egyes bejegyzések értékelése, amivel a fölösleges/értelmetlen bejegyzéseket valamilyen szinten moderálni lehet.
* Moderációs rendszer bővítése: felhasználók jelentése, kitiltása.

## Fejlesztői környezet

# Összefoglalás

# Irodalomjegyzék

* <https://nextjs.org/docs>
* <https://reactjs.org/>
* <https://stackoverflow.com/>
* <https://www.svgrepo.com/>
* <https://imdb-api.com/>