Szakképesítés neve: ……

OKJ száma: ……

SZAKDOLGOZAT

SZAKDOLGOZAT CÍME

Témavezető:

Beke Béla

Készítette:

Pataki Dávid Ferenc

Debrecen

2023

Tartalomjegyzék

1 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.

1.1 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.

1.2 Köszönetnyilvánítás

2 Felhasználói dokumentáció

2.1 Rendszerkövetelmény

2.2 Weboldal használata

3 Fejlesztői dokumentáció

A weboldal fejlesztéséhez *Node.js*-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* **keretrendszert** használom, ami a React frontend könyvtárat kiegészíti egy Express-hez hasonló backend RESTful API készítő csomaggal.

A *React* a *Meta* által létrehozott JavaScript könyvtár, mellyel könnyen lehet készíteni interaktív felhasználói felületeket.

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

3.1 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.

3.2 Telepítés és indítás

A project optimális futása érdekében erősen ajánlott a *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.

3.3 Backend

3.3.1 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.

3.4 Adatbázis

3.4.1 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 beállított 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

3.4.2 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*(TODO) fejlécben többet megtudhatunk.

3.4.3 follow tábla

Követések feljegyzésére szolgáló tábla.

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

3.4.4 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

3.4.5 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 beállított leírás vagy szöveg
- **create_date**: Dátum + idő típus, értékelés létrehozásának idejét tárolja

3.4.6 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

3.4.7 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

3.4.8 genre tábla

Film műfajok tárolására szolgáló tábla

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

3.5 Algoritmusok

[TODO: Kép]

3.5.1 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.

3.5.2 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.

3.5.3 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 enkriptációs algoritmussal bizonyítjuk be, hogy az a token tényleg valódi.

3.5.4 [TODO: Képfeltötés]

3.6 Tesztdokumentáció

3.7 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.

3.8 Fejlesztői környezet

4 Összefoglalás

5 Irodalomjegyzék

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