Eötvös Loránd Tudományegyetem

Informatikai Kar

Adattudományi és Adattechnológiai

Tanszék

Multimédia Vizualizációs Platform

Témavezető:
Tarcsi Ádám
egyetemi tanársegéd

Szerző:
Tarjáni Martin Dominik
programtervező informatikus BSc

Budapest, 2021

Szakdolgozati témabejelentő

Hallgató adatai:

Név: Tarjáni Martin Dominik

Neptun kód: FAZMVB

Képzési adatok:

Szak: programtervező informatikus BSc

Tagozat: Nappali

Témavezető neve: Tarcsi Ádám

Munkahelyének neve, tanszéke:

ELTE Informatikai Kar, Adattudományi és Adatt chnológiai Tanszék

Munkahelyének címe:

117 Budap st, Pázmány Pét r sétány 1/C., 2. m l t, 2.420-as szoba

Beosztás és iskolai végzettsége:

gy t mi tanárs géd

A szakdolgozat címe: Multimédia Vizualizációs Platform

A szakdolgozat témája:

A Multimédia Vizualizációs Platform egy számítógépre, külső HDD-re, SSD-re vagy Pendrive-ra letöltött multimédia - főleg filmek, sorozatok - vizualizációjára szolgál, tehát ezen tartalmak fogyasztására egy áttekinthető és könnyen kezelhető felületet biztosít. Hasonlót, amelyhez már hozzászokhattunk a különböző streaming szolgáltatóktól, csak mindezt nem egy külső adatbázisból betöltve, hanem a saját offline tartalmainkra építve. A felület biztosítja az alapvető funkcionalitásokat amelyeket egy videó lejátszó platformtól el lehet várni: filmek listázása, keresés, lejátszás, megállítás, előre-hátra pörgetés, nyelv és felirat választás, lejátszási listák létrehozása stb. Külön kezeli az egyrészes filmek, a több részes filmsorozatok és a több részes TV sorozatok felületeit. A filmekhez és sorozatokhoz továbbá meta-adatokat (Cím, Szereplők, Leírás, Műfajok stb.) tölt le/be, ezzel is növelve a felhasználói élményt.

Budapest, 2020.12.01.

Köszönetnyilvánítás

Soha nem gondoltam volna, hogy ilyen gyorsan eltelnek az egyetemi évek, sokminden történt ezalatt a 4 év alatt és túl hosszú lenne a lista, hogy kiknek is köszönhetek rengeteg mindent. Ennek ellenére megpróbálkozok vele.

Köszönöm a témavezetőmnek, Tarcsi Ádámnak, hogy mindig ott volt amikor kérdésem volt és segített bármiben amiben csak kellett. Köszönöm Thornak vagyis Gerely Viktor Andrásnak a sok tanácsadást és támogatást amit nyújtott közel se csak a szakdolgozatom elkészültében, de mind a 4 év alatt. Köszönöm barátaimnak, Vida Bálintnak, Tompa Gábornak és Fenyvesi Leventének, hogy mellettem álltak a legelejétől, a Gólyatábortól kezdve egészen a Záróvizsgáig és azon túl. Köszönöm oktatóimnak, gyakorlat vezetőimnek, akik szakértelmükkel hozzájárultak, hogy tudásom ezalatt a 4 év alatt exponenciálisan növekedjen.

Végül, de nagyon nem utolsó sorban köszönöm barátnőmnek, Tamás Laurának és szüleimnek a rengeteg támogatást, akik az elejétől fogva támogatnak és mellettem állnak és akik nélkül egészen biztosan nem állhatnék itt ma.

Tartalomjegyzék

1. fejezet

Bevezetés

A mai világban elég hálás dolog film- és sorozatrajongónak lenni, rengeteg tartalommal bombáznak minket a különböző streaming platformok (Netflix, HBO GO), TV csatornák, Youtube csatornák és még sorolhatnánk. Mindazonáltal természetesen semmi sem tökéletes: biztos a Kedves Olvasó is tapasztalta már, hogy kereste a kedvenc filmjét, sorozatát az éppen akutálisan előfizetett platformon vagy TV-ben. Azoban szinte törvényszerű, hogy amit éppen akarunk nézni az sosincs fent a kínálatban, vagy még rosszabb, amikor fent van, de nem olyan formában, ahogy nekünk az megfelelő (például nincs magyar vagy angol szinkron, magyar felirat stb.), ez tipikus esete a „so close, yet so far" szituációnak.

Mindeközben polcainkon egyre porosodnak - vagy már réges-régen lekerültek onnan - a nem használt CD-k, DVD-k, Blu-Ray lemezek és így 2021-ben élérkeztünk oda is, hogy a nagy becsben tartott kedvenc filmjeinket tartalmazó több terrabájtos külső merevlemezünk is a por és a feledés martalékává válik.

Témaválasztásomat és motivációmat, tehát ezen téma ihlette meg leginkább, hiszen hasonló cipőben járok jómagam is. Temérdek kedvenc filmemnek szerettem volna egy átlátható és szép környezetet biztosítani, amelyen ezen tartalmak fogyasztása mégnagyobb élmény.

Alkalmazásommal ezekre a problémákra szeretnék megoldást nyújtani: egy offline tartalmakra, tehát külső meghajtón, SSD-n vagy pendrive-on tárolt és ezekre a tartalmakra épülő, megjelenésében és funckióiban a már megszokott streaming szolgáltatókra hasonlító, könnyen kezelhető és áttekinthető platform létrehozása.
Ez a Multimédia Vizualizációs Platform vagy röviden MVP.

Funkcióit tekintve biztosít mindent amit egy videó lejátszó platformtól el lehet várni:

- Elemek listázása
- Keresés
- Filmek,sorozatok lejátszása, megállítása
- Előre-hátra pörgetés
- Felirat választás
- Lejátszási listák létrehozása
- Külön kezeli a filmek és a többrészes sorozatok felületeit
- Metaadatok betöltése fájlnévből és NFO fájlból
- Metaadatok letöltése adatbázisból

2. fejezet

Felhasználói dokumentáció

2.1. Ismertetés, célközönség

Az alkalmazás segítségével vizualizálni tudjuk offline mozgóképes tartalmainkat egy könnyen kezelhető és esztétikus megjelenésű platformon, továbbá képes a filmek és sorozatok külön kezelésére, felismerésére - amennyiben a filmek nevezéke megfelelő -, a beolvasott tartalmak keresésére, lejátszására, megállítására, pörgetésére, felirat választására, meta-adatok letöltésére filmes adatbázisból vagy betöltésére fájlnévből, NFO fájlból, mindemellett lejátszási listák létrehozására.

Az alkalmazás első indításakor még nem fogunk találni semmilyen tartalmat a Fő képernyőn, ehhez ugyanis először be kell importálnunk a tartalmakat. A Source Importer eszközt használva, kiválasztjuk a megfelelő mappákat, ezután a szoftver beolvassa az összes támogatott formátumú tartalmat. Ezután ha úgy akarjuk meta-adatokat tudunk letölteni a beolvasott fájlokhoz. Innentől kezdve az alkalmazás minden funkciója készen áll a használatra.

Az alkalmazás célközönségét nem igazán lehet vagy érdemes leszűkítani egy kisseb csoportra, tekintve, mindenkinek szól, aki kicsit is szereti a filmeket, sorozatokat, hozzászokott a különböző streaming szolgáltatók által nyújtott kényelemhez, funckionalitáshoz és renelkezik offline média tartalmakkal, legyenek azok akár filmek, akár sorozatok.

2.2. Rendszerkövetelmények

2.2.1. Hard eres kö etelmények

Az alkalmazás futtatásához lényegében egy Google Chrome-ot és Node.JS-t futtatni képes konfigurációra van szükség csupán. Ettől függetlenül ajánlott az alábbi technikai kövelmények teljesítése:

- **Kijelző:** legalább HD, azaz 1280×720 pixel
- **Memória:** legalább 4 GB
- **Háttértár:** legalább 200 MB (SSD ajánlott)

2.2.2. Internetkapcsolat

Az alkalmazás futtatására alapvetően nincs szükség internet kapcsolatra, azonban a teljes élmény és funkcionalitás eléréséhez erősen ajánlott. Például a metaadatok letöltése online filmes adatbázisból történik, aminek működéséhez elengedhetetlen az internet kapcsolat.

2.2.3. Softwares kö etelmények

Három előre telepítendő softwarere van szükségünk, hogy működésre bírjuk az applikációt. Ezek pedig:

- Node.JS
- Node Package Manager
- Yarn Package Manager

2.2.4. Operációs rendszer

Az Electron applikációk sajátja, hogy egy kódbázisból - ami jelen esetben HTML, CSS, JavaScript/TypeScript - mindhárom főbb asztali platformra képes teljes funkcionalitású alkalmazást készíteni. Éppen ezért a támogatott operációs rendszerek az alábbiak:

- Microsoft Windows 7, 8, 8.1, 10
- Linux/GNU
- MacOS

2.3. Telepítési útmutató

Több mód is elérhető az alkalmazás telepítésére. Az egyik lehetséges mód a kódból már Production változattá fordított **futtatható állomány** futtatása, a másik pedig a **Yarn** Package Manager-t felhasználva. Az alábbiakban mindkét módot tárgyalni fogjuk:

2.3.1. Telepítés futtatható állománnyal

Az alkalmazásból létezik telepíthető futtatható állomány, ehhez semmi másra nincs szükség csupán a telepítő elindítására. A telepítő ekkor kicsomagolja majd fetelepíti az alkalmazást a számítógépre, parancsikont hoz létre az asztalra és automatikusan el is indítja az alkalmazást.

2.3.2. Telepítés és futtatás Yarn segítségé el

Először is szükség van a megfelelő szoftverek telepítésére, mégpedig a Node.JS-re, Node Package Manager-re (továbbiakban npm) és a Yarn Package Manager-re (továbbiakban yarn). A Node.JS és az npm letölthető a Node.JS hivatalos oldaláról[1], amellyel együtt az npm is letölésre kerül, ajánlott továbbá az LTS verzió letölése, amely „hosszú ideig tartó támogatás”-t jelent. A yarn-t pedig a Yarn hivatalos oldaláról[2] érdemes beszerezni.

Ezen szoftverek telepítése után indítható maga az alkalmazás. A projekt leíró, a *package.json* fájlban definiált script-ek, amellyekkel az alkalmazást elindítani és lefordítani tudjuk az alábbiak:

[1] `https://nodejs.org/en/`

[2] `https://classic.yarnpkg.com/en/docs/install`

Repository klónozása

Először is klónoznunk kell a Repository-t, amely a GitHubon található. Ehhez használhatjuk a git clone parancsot - ha telepítve van a gépünkre a Git[3] - vagy le is tölthetjük azt zip formátumban, az élénk zöld „Code”, majd a „Download ZIP” gombra kattintva.

```
git clone https://github.com/TMD44/mvp
cd mvp
```

2.1. forráskód. Repository klónozása

Függőségek telepítése Yarn segítségével

Ezután a függőségek telepítése következik, ezek olyan csomagok, modulok amelyek a program futásához (dependecies) vagy fejlesztéséhez (devDependecies) elengedhetelenek.

```
yarn
```

2.2. forráskód. Függőségek telepítése Yarn segítségével

Fejlesztői verzió futtatása

Innentől kezdve az alkalmazás indítható, buildelhető. Ezzel a paranccsal az alkalmazás egy fejlesztői verziója indítható el, ebben a módban különböző fejlesztői eszközök is elérhetőek amelyek a produkciós változatba nem kerülnek bele. Például fejlesztői konzol, React és Redux fejlesztői eszköz.

```
yarn start
```

2.3. forráskód. Fejlesztői verzió futtatása

Produkciós verzió futtatása

Az alábbi paranccsal egy produkciós verzió készíthető el.

```
yarn build
```

2.4. forráskód. Produkciós verzió futtatása

[3] https://git-scm.com/downloads

Produkciós verzió létrehozása

Végül de nem utolsó sorban az egyik legfontosabb parancs, amellyel az alkalmazás produkciós verziója hozható létre. Ez az utasítás létrehozza magát a telepítő futtatható állományt és egy hordozható állományt, amely telepítés nélküli futtatást tesz lehetővé. Mindezeket a projekt *release* mappájába fogjuk találni.

Továbbá a parancshoz különböző kapcsolók társíthatók, amellyel specifikálhatjuk, hogy milyen platformra szeretnénk az alkalmazást buildelni. Az *-mwl* kapcsolóval mindhárom platformra tudunk telepítőt létre hozni. (Fontos megjegyezni, hogy MacOS-re csak MacOS-en lehet telepítőt készíteni. Tehát ha például Windowson adjuk ki az -mwl parancsot akkor csak Linuxra és Windowsra tudunk telepítőt készíteni, Macre nem.) A további kapcsolók értelem szerűen specifikusan egy platformra hozzák létre.

```
yarn package

yarn package --[option]
# All:        -mwl
# Windows:    --win, -w
# Linux:      --linux, -l
# MacOS:      --mac, -m
```

2.5. forráskód. Produkciós verzió létrehozása

Teszt fájlok generálása

Az alábbi paranccsal teszt média fájlokat lehet generálni. Az így létrehozott videófájlok, feliratfájlok, NFO fájlok természetesen nem valós vagy sérült fájlok lesznek, tehát például a videófájlokat nem lehet lejátszni, de a Scannelés ellenőrzéséhez tökéletesen elegendőek.

```
yarn generate-media
```

2.6. forráskód. Teszt fájlok generálása

2.4. Program felületek és működésük

2.4.1. Kezdőlap

2.1. ábra. Kezdőlap

A programot elindítva rögtön a Kezdőlapra jutunk, ahol megjelenik az összes beolvasott média tartalom, filmek és sorozatok vegyesen. Ha elsőnek indítjuk el az alkalmazást akkor a Kezdőlapot üresen fogjuk találni egy üzenettel, hogy még nincs beolvasott média tartalom. Innen a kezdőlapról minden más egyéb oldal könnyen megközelíthető, baloldalt találjuk a Menüt, ahonnan a többi oldalt vagy modált érjük el. Felül az Alkalmazás sáv kapott helyet, ahol a Menüt vezérlő gomb, a jelenlegi oldal neve és a Kereső sáv található. A média tartalmak alapértelmezetten tízesével jelennek meg az oldalon, ezt azonban van lehetőségünk megváltoztatni: 10, 25, 50 és 100 film/oldalra is. A beolvasott média tartalmak kis kártyákként jelennek meg, melyekre rákattintva a film vagy sorozat adatlapjára oldalára jutunk. Ahol a fontosabb adatokat - cím, évszám, összefoglalás, borítókép - találjuk, már amennyiben alkalmaztuk az adatbázisból lehívás funkciót, ezen az oldalon találjuk továbbá a film elindításához szükséges gombot vagy sorozatok esetén gombokat. Amelyekre rákattintva megnyílik a videó lejátszó.

2.4.2. Filmek, Sorozatok

2.2. ábra. Filmek, Sorozatok

A további oldalakon az alkalmazás tematikusan szétválogatja a beolvasott filmjeinket és sorozatainkat. A fentebb Kezdőlaphoz leírtak ezekre az oldalakra is érvényesek, tehát az Applikáció struktúrája, baoldalt a Menüsáv, fent az Alkalmazás sáv, Keresés és a többi.

A *Filmek* oldalon értelemszerűen a filmeket találjuk.

A *Sorozatok* oldalon magától értetődően a sorozatokat találjuk.

2.4.3. Média adatlap

2.3. ábra. Filmek, Sorozatok

TODO

2.4.4. Lejátszási listák

2.4. ábra. Lejátszási listák létrehozása

A *Lejátszási listák* menüponton belül tudunk hozzáadni új lejátszási listákat, vagy a már létezők között böngészni. Új lejátszási lista hozzáadásához írjuk be a lista nevét, majd nyomjunk Entert vagy kattintsunk a + gombra, ezután a listában megjelenik az általunk hozzáadott elem. Erre rákkattintva a főoldalon megjelenik a lejátszási lista tartalma, amennyiben nincs még a listához hozzáadva egy elem sem, azt a felület jelzi számunkra. A teljes lejátszsi listát törölni is tudjuk, a menüpontra kattintva azon belül a jobb oldalon található kuka ikonra kattintva. Ekkor a lista törlődik a rajta lévő filmekkel együtt - fontos megjegyezni, hogy maga a média tartalom nem törlődik a lejátszási lista törlésekor.

Hozzáadni új filmet vagy sorozatot az adott média adatlapján van lehetőségünk. Felül, az ikonsorban kattintsunk a lejátszási lista ikonra, ekkor megnyílik egy kissebb ablak amelyben a már létrehozott listák találhatók. Kattintsunk az választott listára, ekkor a film hozzáadódik a lejátszási listához, az ikon + ikon pipás ikonra változik ezzel visszajelezve a felhasználónak, hogy hozzáadódott. Törölni hasonló módon tudunk, ha hozzá van adva a Lejátszási listához egy film akkor annak adatlapján, a lejátszási listák gombra kattintva, jobb oldalon egy X ikon található, amely megnyomásával a film törlődik a lejátszási listából.

2.5. ábra. Média tartalmak hozzáadása a Lejátszási listához

2.4.5. Source Importer

2.6. ábra. Source Importer 1. fázis: Mappa kiválasztás

Az alkalmazás egyik legfontosabb része a Source Importer vagy „forrás importáló”. A Menüben kattintsunk a *Source Importer* ikonra ekkor megnyílik a modal ablak, itt tudjuk a média tartalmakat a gépünkről beimportálni az alkalmazásba.

Legelső lépésként ki kell választanunk azokat mappákat ahonnan be szeretnénk a filmeket és sorozatokat importálni. A *Mappák megynitása* gombra kattintva operációs rendszertől függően megnyílik egy párbeszéd ablak ahol a mappákat ki tudunk választani, tegyünk így aztán kattintsunk a Mappaválasztás gombra a párbeszéd ablakban. Ekkor visszaérve a Source Importerbe megjelenik a kiválasztott mappák listája, a mappa nevek jobboldalán lévő kis kukára, ha kattintunk egyesével ki tudjuk törölni a kiválasztott mappákat, ha pedig az *Összes törlése* gombra kattintunk akkor értelemszerűen az össze mappa törlésre kerül. Ha mindezekkel megvagyunk és a megfelelő mappák vannak kiválasztva kattintsunk a *Következő* gombra.

2.7. ábra. Source Importer 2. fázis: Offline scan

Az importálási folyamat második állomása az Offline scannelés, kattintsunk *Források importálása* gombra, hogy elinduljon a scannelés. A háttérben a következő dolgok történnek: Az alkalmazás végignézi a kiválasztott mappákat és megkeresi a támogatott formátumú videófájlokat, feliratfájlokat és NFO fájlokat. Sok példa van arra, hogy egy-egy filmfájl mellett minta videófájlok is találhatók, ezeket az alkalmazás szempontjából lényegtelen fájlokat a scannelés során kiszűrjük. Miután megvannak a megfelelő médiafájlok az alkalmazás megpróbál minden használható információt összegyűjteni a fájlnevekből, mappanevekből, NFO fájlokból, majd ezeket összesíti. Például a teljesség igénye nélkül: cím, IMDB ID, felbontás, évszám, nyelv, minőség, évad- és epizódszám, hang és videó kodekek továbbá egyéb hasznos információ.

A scannelés időtartama ha sok tartalmat importálunk be eltarthat néhány percig.

2.8. ábra. Source Importer 3. fázis: Online scan

Az importálási folyamat utolsó, harmadik állomása az úgy nevezett Online scannelés, amely gyakorlatilag a meta-adatok letöltését jelenti. Ehhez mint a nevében is benne van internetkapcsolatra lesz szükség, ha ez nincs meg a scannelés nem végződik el. Kattintsuk a *Metaadatok letöltése* gombra, ekkor elindul a scannelés. Fontos megjegyezni, hogy egyenlőre a meta-adatok letöltése IMDB ID-tól függ - éppen ezért ajánlott ellenőrizni, hogy a média fájlok mellett legyen egy NFO fájl amely tartalmaz egy helyes IMDB linket - , ugyanis ezzel lehet biztosítani azt, hogy ténylegesen a megfelelő adatok kerülnek letöltésre. Bonyodalmakat okozhat ugyanis, ha például több film is létezik az adatbázisban hasonló vagy ugyanolyan névvel, vagy ha a fájl nem megfelelő, értelmes módon van elnevezve akkor a név szerinti keresés szintén hibára vagy eredménytelen válaszra futhat ki. TODO

A scannelés időtartama valamivel több ideig tarthat mint az Offline scannelés, a beolvasott tartalmak és meta-adatok mennyiségétől, az internetkapcsolat gyorsaságától, sávszélesség foglaltságától függően.

2.4.6. Kereső

2.9. ábra. Kereső

A fenti Alkalmazás sávon található a Kereső, amely nevére nem rácáfolandón a beolvasott tartalmak közötti keresésre szolgál. Ha csak rákattintuk a kereső sávra akkor egy legördülő listában azonnal megkapjuk az össze alkalmazás által beolvasott média tartamat. Szöveges mezőbe írással pedig ezen a listán tudunk szűkítani, ha nincs olyan film cím mint amit beírtuk a kereső jelzi számunkra. Amennyiben megtaláltuk keresett filmünket a legördülő menüben rákattintva megnyílik az adott film vagy sorozat adatlapja, ahol is a már fent részletezett módon lehet eljárni.

2.4.7. Beállítások

2.10. ábra. Beállítások

A Menüben a *Beállítások* ikonra kattintva megnyílik a Beállítások ablak TODO

2.5. Hibaelhárítás

TODO

3. fejezet

Fejlesztői dokumentáció

3.1. Használt technológiák

Az alkalmazás fejlesztését JavaScript nyelven kezdtem meg és fejlesztés közben váltottam TypeScript nyelvre. A TypeScript gyakorlatilag egy típusos kiterjesztése a JavaScriptnek, éppen ezért minden kód amit JavaScriptben írtam azok érvényes és valid kódok TypeScriptben is. A TypeScript lényege, hogy típus biztossá teszi az alkalmazást, ami ugyan megnehezíti a fejlesztést, de nagy előnye, hogy sok probléma már a fejlesztés során észlelhető - ezáltal javítható - és nem a kész verzióban történnek katasztrofális dolgok. Váltásom motivációja a TypeScript nyújtotta előnyök mellett az is volt, hogy az ipar jelenlegi trendjei teljesen egészében effelé a nyelv felé mutatnak, minden komolyan vehető program - amely Electront, Angulart vagy más hasonló webes technológiát használ - már TypeScript nyelven íródik. Ez predesztinálta az én hozzáállásomat is a nyelvhez, tekintve a legelejétől fogva célom volt a legfrissebb és legnépszerűbb technológiák használata, ezen piacképes tudások elsajátítása.

Az alkalmazás fejlesztése teljes egészében a Visual Studio Code nyílt forráskódú kódszerkesztővel zajlott, Electron-t, React-et és Redux-ot továbbá Git verziókezelő rendszert felhasználva és egy publikus GitHub repositoryban tárolva a kódot, amely elérhető az alábbi címen: `https://github.com/TMD44/mvp`

3.1.1. Electron

Az Electron[4] egy nyílt forráskódú keretrendszer, amelyet asztali alkalmazások fejlesztéséhez lett kifejlesztve. Különlegessége, hogy a már ismert és bejáratott webes technolgiákat - tehát HTML, CSS, JavaScript - lehet használni asztali alkalmazások fejlesztéséhez. Az Electron két külön, egymástól hermetikusan elzárt része a *main process* és a *renderer process*, melyek között alá-fölé rendelt viszony található.

Az Electron „fő szála" - ezt nem véletlenül tettem idézőjelbe, tekintve a JavaScript egy egyszálú programozási nyelv, tehát JavaScriptben nem tudunk konkurrens programokat írni -, vagy hívhatjuk angolul *main process*nek is, ami gyakorlatiag egy Node.JS alkalmazás, amely a böngésző ablakok létrehozásáért, az operációs rendszerrel való kommunikációért és az alkalmazás életciklusárt felel. Ez gyakorlatilag egy fájl, amely az alkalmazásunk belépési pontját fogja képezni.

Az előbb említett böngésző ablak pedig az Electron másik elszeparált és alárendelt része, a *renderer process*, amelyet a *main process* hoz létre és menedzseli is azt. Ez tulajdonképpen egy Chromium böngésző ablakot fog jelenteni, amely egy HTML fájlt fog megjeleníteni, ahogy azt a webes világban már megszokhattuk. Lehetőgésünk van továbbá Node intergrációhoz is, amit a *main process*en belül tudunk engedélyezni. Az integráció lényege, hogy a böngésző ablakból használhatjuk a Node által nyújtott funckiókat is, például fájlrendszer elérése, modulok importálása és exportálása.

3.1.2. React

A React[5] egy JavaScript könyvtár, amely főleg felhasználó felületek készítésére nyújt egy fölöttébb elegáns megoldást. A segítségével rendkívül egyszerűen tudunk, úgy nevezett Single Page Application-ket készíteni. Az innováció ebben a megközelításben az, hogy nincs szükség az oldal újbóli betöltésére, hanem a felhasználói interakció során az oldal tartalmai dinamikusan újratölthetőek és nem is az egész oldal, hanem annak kisseb részei, úgy nevezett komponensei.

[4] `https://www.electronjs.org/`

[5] `https://reactjs.org/`

3.1.3. Redux

A Redux[6] egy state management tool, vagyis magyarul egy állapot kezelő rendszer. A Redux lényege és legnagyobb előnye, hogy az alkalmazásunk állapotát képes külön, egy külön egységként tárolni és menedzselni. Az állapotot memóriában tárolja, gyakorlatilag egy nagy JSON objektumban, ezen adatok módosítása pedig *dispatch* útján valósítható meg.

3.2. Megvalósítási terv

3.2.1. Kö etelmény specifikáció

Az alkalmazás működéshez és a feladatleírás teljesítéséhez az alábbi funkciókat kell implementálni:

- **Film felsoroló platform:** Az alkalmazásnak felületet kell biztosítania a beolvasott média tartalmak megjelenítéséhez, listázásához.
- **Média tartalmak beolvasása:** Mappák kiválasztásával az azokban fellelhető média tartalmak beolvasása.
- **Keresés a média tartalmak között:** Az alkalmazásnak lehetőséget kell bizotsítania a beolvasott média tartlamak közötti kereséshez.
- **Lejátszás, Megállítás, Pörgetés:** Bizotsítania kell a videók lejátszását.
- **Felirat választás:** Feliratok megjelenítése lejátszáskor, amennyiben elérhető.
- **Sorozatok és filmek kezelése:** A felület kezelje külön a sorozatokat és filmeket, tehát például a sorozat epizódjai ne külön-külön jelenjenek meg a listában, hanem a sorozatra rákkatintva, annak részletező ablakjában legyenek listázva az epizódok.
- **Metaadatok betöltése és letölése:** Használható és értelmes adatok beolvasása fájlnévből, mappanévből, NFO fájlból és külső adatbázisból metadatokat kell tudnia letölteni.

[6] `https://redux.js.org/`

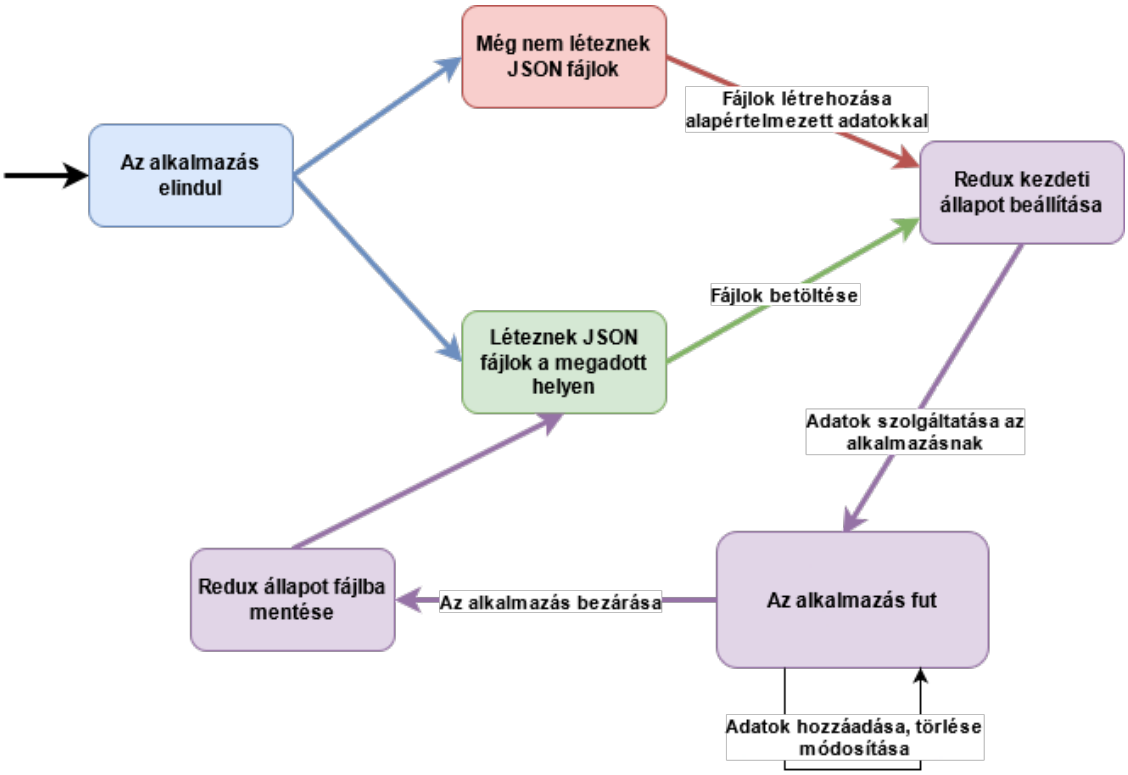
- **Lejátszási listák:** Lehessen lejátszási listákat létrehozni, ezen listákat filmekkel feltölteni.

3.2.2. Adatok tárolása, kezelése

Az alkalmazás futása közben a Redux segítségével kezeljük a felhasznlói felülethez és a beolvasott média tartalmakhoz kapcsolódó adatokat. Reduxot főleg webalkalmazásokhoz használnak, de nekünk itt asztali környezetben is alkalmazható lesz egy kis trükkel, mégpedig amikor megnyitjuk az alkalmazást, a Redux beállítja a kezdeti állapotot - ez a webes környezetben annyit tenne, hogy egy előre meghatározott alapértelmezett állapot sémát alakalmazna - ami a mi esetünkben két fájl lesz, a *config.json* és a *mediaDB.json*. A *config.json* fájl az alkalmazás futásához, a beolvasáshoz és a felhasználó személyes preferenciáihoz szükséges adatokat, míg a *mediaDB.json* a beolvasott média tartalmakhoz kapcsolodó adatokat, metaadatokat tartalmazza. Ez a két JSON fájl két féleképpen jöhet létre:

- Létrejöhet az alkalmazás bezárásakor, ekkor az alkalmazás jelenlegi állapota kerül kiírásra a fájlba.

- Valamint létrejöhet az alkalmazás indításakor is, ha a program úgy érzékeli, hogy a megadott helyen még nem létezik ez a két fájl. Ilyenkor egy előre megadott alapértelmezett állapot kerül kiírásra, amely alapértelmezett beállításokat (felhasználói felület, beolvasásssal kapcsolatos preferenciák stb.) és egy üres vázat tartalmazza, amelybe bekerülnek majd a scannelés során gyűjtött adatok.



3.1. ábra. A Redux kezdeti állapot beállítása

Tehát az alkalmazás futása közben az adatok a Redux store-ban tárolódnak, amikor az alkalmazás bezárásra kerül, akkor az állapot két fájlba kerül kírásra - ebből fakadóan két futás között az adatok fájlban átrolódnak - amit aztán újranyitás esetén egyszerűen betöltünk a kezdeti állapotba.

Példa config.json fájl

```
{
"creation_time": "2021.05.10. 12:00:00",
"modification_time": "2021.05.10. 12:00:00",
"user_info": { "name": "Anonymous" },
"app_info": {
    "app_name": "Multimedia Visualization Platform",
    "app_version": "0.1.0",
    "app_current_directory": "G:\\mvp\\src\\main",
    "app_locale": "hu",
    "app_locale_country_code": "HU",
    "app_paths": {
        "home": "C:\\Users\\tmd-pc",
```

```
        "user_data": "C:\\Users\\tmd-pc\\AppData\\Roaming\\
            Multimedia Visualization Platform",
        "desktop": "C:\\Users\\tmd-pc\\Desktop",
        "downloads": "C:\\Users\\tmd-pc\\Downloads",
    }
},
"scan_preferences": {
    "scan_language": "en-US",
    "scan_paths": [ "G:\\SOROZATOK" ],
    "scan_file_types": {
        "media": [ ".mkv",".mp4",".avi" ],
        "sub": [ ".srt",".ass",".vtt",".ssa",".sub",".stl",".scc"
            ],
        "nfo": [ ".nfo" ]
    },
    "scan_results": {
        "media": [{
            "fn": "Game.of.Thrones.S01E01.BDRIP.x264.Hun.Eng",
            "ext": ".mkv",
            "path": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.
                x264.Hun.Eng",
            "full": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.
                x264.Hun.Eng\\Game.of.Thrones.S01E01.BDRIP.x264.Hun.
                Eng.mkv"
            }],
        "sub": [{
            "fn": "Game.of.Thrones.S01E01.BDRIP.x264.Hun.Eng",
            "ext": ".srt",
            "path": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.
                x264.Hun.Eng",
            "full": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.
                x264.Hun.Eng\\Game.of.Thrones.S01E01.BDRIP.x264.Hun.
                Eng.srt"
            }],
        "nfo": [{
            "fn": "Game.of.Thrones.COMPLETE.BDRIP.x264.Hun.Eng",
            "ext": ".nfo",
            "path": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.
                x264.Hun.Eng",
```

```
43              "full": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.
                    x264.Hun.Eng\\Game.of.Thrones.COMPLETE.BDRIP.x264.
                    Hun.Eng.nfo"
44              }],
45      }
46  }}
```

3.1. forráskód. Példa config.json fájl

Példa mediaDB.json fájl

```
1  {
2  "creation_time": "2021.05.09. 14:44:49",
3  "modification_time": "2021.05.09. 14:44:49",
4  "movies": {
5      "23b5b6d765ac7a28badbd2f6e6d31ea3": {
6          "id": [ "23b5b6d765ac7a28badbd2f6e6d31ea3" ],
7          "media_type": "movie"
8      },
9      "6209ef3cb9451c0def78013521b84d5f": {
10         "id": [ "6209ef3cb9451c0def78013521b84d5f" ],
11         "media_type": "movie"
12     },
13 },
14 "tv_series": {
15     "Game of Thrones": {
16         "id": [
17             "98cddc3eb8b19c7ebb2a84049c5fc4f0",
18             "6a216938ea15d2a1e5282ec47d9a0649",
19             "6044b7976f57147dd35da9c3719fd921",
20         ],
21         "media_type": "series"
22     }
23 },
24 "playlists": {
25     "Game of Thrones Best of": {
26         "contents": {
27             "Game of Thrones": {
28                 "id": [
```

```
                    "98cddc3eb8b19c7ebb2a84049c5fc4f0",
                    "6a216938ea15d2a1e5282ec47d9a0649",
                    "6044b7976f57147dd35da9c3719fd921",
                ],
                "media_type": "series"
            }
        },
        "media_type": "playlist"
    }
},
"all_media": {
    "98cddc3eb8b19c7ebb2a84049c5fc4f0": {
        "id": "98cddc3eb8b19c7ebb2a84049c5fc4f0",
        "media_name": "Game.of.Thrones.S01E01.BDRIP.x264.Hun.Eng",
        "extension": ".mkv",
        "path": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.x264
            .Hun.Eng",
        "full_path": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP
            .x264.Hun.Eng\\Game.of.Thrones.S01E01.BDRIP.x264.Hun.Eng
            .mkv",
        "subtitles": [ "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.
            BDRIP.x264.Hun.Eng\\Game.of.Thrones.S01E01.BDRIP.x264.
            Hun.Eng.vtt" ],
        "nfo": "G:\\SOROZATOK\\Game.of.Thrones.COMPLETE.BDRIP.x264.
            Hun.Eng\\Game.of.Thrones.COMPLETE.BDRIP.x264.Hun.Eng.nfo
            ",
        "metadata": {
            "title": "Game of Thrones",
            "season": 1,
            "episode": 1,
            "languages": ["HUN", "ENG"],
            "codec": "X264",
            "quality": "BDRIP",
            "media_type": "series",
            "imdb_id": "tt0944947",
            "imdb_url": "https://www.imdb.com/title/tt0944947"
        }
    },
```

```
}}
```

3.2. forráskód. Példa mediaDB.json fájl

3.2.3. Média tartalmak beol asása

Az alkalmazás egyértelműen legfontosabb része a média tartalmak beolvasása. Ezt három részre lehet lebontani TODO

Mappa kiválasztás

A mappák megnyitásához az Electron egy beépített modulját, a „dialog”ot fogjuk használni. A probléma csak az, hogy ezt a modult nem leszhet a renderer processben használni csak a main processben, ilyenkor jön jól az IPC renderer és IPC main modulok amelyek kommunikációt tesznek lehetővé a két réteg között. Az „Open Dir” gombra kattintva tehát elküldjük az IPC-n keresztül a main process-nek, ami aztán a „dialog” modult felhasználva megnyitnya az operációs rendszer natív párbeszédablakját, ebben az esetben kifejezetten mappák megnyitását engedélyezzük csak, azokból is többet. Az ablakot bezárva az eredményt string tömb formájában visszaküljük a renderer processnek, amely elvégzi a Redux store-ba dispatch-elését. A config objecten belül „scan paths” kulcs néven találjuk a beolvasott mappák tömbjét.

```
ipcMainCommunication.on('open-dir-sync', (event) => {
    dialog
        .showOpenDialog(window, {
            properties: ['openDirectory', 'multiSelections'],
        })
        .then((result) => {
            if (result.canceled === false) {
                event.returnValue = result.filePaths;
            } else {
                event.returnValue = [];
            }
        })
        .catch((err) => {
            console.log(err);
        });
```

```
16 });
```

3.3. forráskód. Mappák megnyitása

Offline scan

Az Offline scannelés megkezdéséhez két dologra lesz szükségünk, az egyik a mappák amelyekben keresni fogunk, a másik hogy milyen fájl típusokat fogunk keresni, igencsak hatékonytalan lenne ha minden fájlt beolvasnánk amelyek a program működését tekintve nem lényeges információkat tartalmaznak, éppen ezért szűrni fogjuk a fájlokat. A Redux store-ból tehát behívjuk a „scan file types" kulcshoz tartozó objektumot amely egy tematikusan tömböket tartalmazó objektum, ezek előre meghatározott fájltípusok, a felhasználónak lehetősége van megváltoztatni őket. Behívjuk továbbá az előző pontban hozzáadott „scan paths"-eket.

A megadott mappa elérési utvonalakon végig iterálva minden útvonalra meghívjuk a „scanDir" függvényt amely rekurzívan a mappa tartalmán - amennyiben a mappában még több mappa található, ezekben az estekben is hasonlóképpen jár el - és a megadott fájltípusokra szűrve egy tömbbe pusholja a talált fájlokat (videófájlok, például „.mkv"„.mp4" vagy felirat fájlok, például „.srt", „.vtt", illetve NFO fájlok „.nfo"). Történik még egy filterezé ezt követően, tekintve nagyon sok példa van rá, hogy az offline videó tartalmaink mellet úgy nevezett „sample" vagy minta videó fájlok is helyt kapnak, ezek a fájlok szűrésre kerülnek, ugyanis a program működésének szempontjából lényegtelen fájlokról beszélünk. Eztuán minden talált elemet átalakít egy sokkal kezelhetőbb formátumra, a fentebb már config.json példában bemutatott „scan_results" kulcsban. A lényeg itt annyi, hogy a megfelelő formátumú fájlokat a megfelelelő objektumbra szortírozzuk, tehát média fájlokat a media objektumba, a felirat fájlokat a sub objektumba, az NFO fájlokat pedig az nfo objektumba. Mindezek elvégzése után elmentjük a Redux store-ba megintcsak dispatch útján.

```
1 const { media, sub, nfo } = getScanFileTypes();
2 const paths = getScanPaths();
3
4 for (const i in paths) {
5     const files = await scanDir(paths[i], [media, sub, nfo].flat())
          ;
```

```
6       for (const j in files) {
7           if (!excludedFromScan(files[j])) {
8               fileSorting(files[j], media, sub, nfo, scanResults);
9           }
10      }
11  }
12  store.dispatch(addScanResults(scanResults));
13  const results = getScanResults();
14
15  for (const file in results.media) {
16      const result = await mediaJSONGenerator( results.media[file],
            results );
17      scan.mediaInJSON[result.id] = result;
18  }
19  store.dispatch(addMediaAtOnce(scan.mediaInJSON));
```

3.4. forráskód. Mappák megnyitása

A következő for ciklusban már ezeket az átalakított objektumokat fogjuk felhasználni és tovább alakítjuk „mediaJsonGenerator” függvény segítségével. A függvény elször összepárosítja a felirat fájlokat a hozzá kapcsolódó videófájlhoz, tekintve a feliratfájloknál eddig is az volt a konvenció, hogy a felirat fájl neve legyen ugyanaz mint a videó fájl neve, ezt felhasználva könnyű dolgunk van. Végeredményben egy tömböt fogunk kapni, amely tartalmazza a videó fájlhoz köthető feliratokat. A felirat fájlokkal történik a program ezen pontján még egy fontos dolog, mégpedig egy „.vtt” formátumra konvertálás, ennek oka az, hogy habár asztali alkalmazást készítünk az eszköztár amelyet használunk webes technológiákra épülnek éppen ezért igazodnunk kell a web által támogatott formátumokra, amely jelen esetünkben felirat fájlok esetén a „.vtt”.

Eztuán az NFO fájl - általában egy NFO fájl köthető egy videó fájlhoz - videófájlhoz párosítása következik. Ez már kicsit bonyolultabb eset, itt is fenn áll ugyan a „legyen ugyanaz a neve”, de a fájlstruktúrán belül a fájl elhelyezkedése változhat. Sorozatok esetén például általában nincs mindegyik epizódhoz külön NFO fájl, hanem magához az évadhoz vagy az egész sorozathoz, tehát a fájl helye nem feltétlenül a videó fájl mellet található. A scannelés folyamata lekezelei ezeket a különböző eseteket, tehát végeredményben megkapjuk a videófájlhoz köthető NFO fájlt. Innentől

kezdve rendelkezünk tehát NFO fájllal amely rengeteg hasznos offline információt tartalmaz a filmünkről, azonban itt olyan problémába ütközünk, hogy az NFO fájl tartalmát tekintve semmilyen konvenció vagy szokás nincs, éppen ezért algoritmikus úton kifejezetten nehéz biztosan helyen információkat kinyerni. Ennek következtében az NFO fájlban kifejezetten csak a legfontosabb információt fogjuk keresni, mégpedig egy IMDB linket. Az IMDB a legelterjedtebb filmes adatbázisok egyike jelenleg és az url címében a film egy egyedi azonosítója található, amelyet fel fogunk tudni használni majd a metaadatok letöltésekor. Az NFO fájl tartalmának beolvasását követően, egy reguláris kifejezés segítségével (*/tt[0-9]7,8/gi*) IMDB azonosítót keresünk, ha találtunk akkor nincs is más dolgunk mint lementeni egy objektumba, amit aztán visszaadunk a „mediaJsonGenerator" függvénynek. Abban az esetben ha nem találtunk explicit azonosítót, megpróbálkozunk megképp. Számos esetben megfigyelhető, hogy az NFO fájlokban URL rövidítő szolgáltasokat használnak, például TinyURL, Bit.ly stb. Scannelésünk folyamatában rendkívül fontos információ az IMDB ID, éppen ezért az algoritumus az egyik legnépszerűbb a TinyURL visszafejtésével is megpróbálkozik. Abban az esetben ha sikerrel jár és a rezolválás eredménye egy IMDB azonosító akkor a már fentebb ismeretett módon járunk el, ha nem járunk sikerrel akkor az információ nem kerül bele az eredmény objektumba.

A másik forrás amelyből offline használható információkat tudunk kinyerni, a fájlnév és a mappa neve. Ehhez a fájl név felismerő függvényünket fogjuk használni, amely az NFO-s azonosító felismeréshez hasonlóan regulásris kifejezések szövegre illesztésével fogjuk kivitelezni. Az „fnr" függvény a következő képpen működik, megkap egy stringet, ami az egyik esteben a fájl neve, a másik esetben a mappa neve. Ezután ezután sorban, reguláris kifejezéseket illeszt a szövegre és ha illeszkedik rá azt a szövegből kivesszük és a visszaadandó eredmény objektumba rakjuk, az eredeti stringben pedig aláhúzás (_) karakterre cseréljük a talált eredményt. Ezt minden egyes Regex típusra eljátszuk, ennek következtében a végére kapunk egy eredmény objektumot amelyben tematikusan szét van válogatva a talált stringek, illetve a visszamaradó stringet amelyben már csak a cím maradt az egyetlen hasznos információ. Ekkor az algoritmus a string elejétől elkezd „értelmes karaktereket" keresni tehát kisbetűs, nagybetűs karakterek illetve számok, és addig megy a szövegben amíg el nem kezd találni minimum kettő vagy annál több „értelmetlen kataktereket", ezek

az aláhúzás (_), a pont (.) és a szóköz ( ). Ha 2 vagy több értelm „értelmetlen kataktert" találtunk, akkor az addig „értelmes" karaktereket kimentjük, ez lesz a cím. Továbbá kicsit szépítünk is rajta például ha a szavak között pont van azt szóközre cseréljük és az első karaktert nagybetűsre cseréljük. Abban az esetben ha valami hiba történik a cím keresésekor a string már üres, vagy rossz formátumú tehát üres stringet adna vissza, az algoritmus az eredeti, még a függvény paraméterül megkapott stringet teszi meg címnek.

```
const fnrPatterns = {
    resolution: /[0-9]{3,4}[PI]{1}|[0-9]{3,4}[.\-_ ]?[X][.\-_
       ]?[0-9]{3,4}/gi,
    year: /((?:19[3-9]|20[0123])[0-9])/g,
    languages: /[.\-_ ](ENG|HUN|GER|JAP)[^a-zA-z]/gi,
    bluray: /BD[0-9]{2}|BD100/gi,
    season: /[.,\-_ ]S([0-9]{1,2})-(S)?([0-9]{1,2})|[.,\-_ ]S
       ([0-9]{1,2})|[^0-9]([0-9]{1,2})X/gi,
    episode: /[.,\-_ ]E([0-9]{1,3})-(E)?([0-9]{1,3})|E([0-9]{1,3})
       (?:[^0-9]|$)|[Xx]([0-9]{1,3})(?:[^0-9]|$)|(EP|EPISODE)
       ([0-9]{1,3})(?:[^0-9]|$)/gi,
    codec: /XVID|DIVX|AVC|HEVC|X[.\-_ ]?26[0-9]|H[.\-_ ]?26[0-9]/gi
       ,
    audio: /(DD|DDP|MA|AC3|AAC|PCM|LPCM|FLAC|DTS[._\- ]?(HD)?|
       TRUEHD[+._\- ]?ATMOS|TRUEHD|ATMOS)[+._\- ]?[0-9]\.?[0-9]|DTS
       [._\- ]?(HD|ES)?|DUAL[._\- ]?AUDIO|DOLBY[+._\- ]?(DIGITAL[+.
       _\- ]?(PLUS)?|VISION|ATMOS)|HALF-OU/gi,
};
```

3.5. forráskód. Példa reguláris kifejezésekre

Végeredményben tehát két objektumot fogunk visszakapni a függvénytől, egyet a fájlnévből kinyert adatokkal és egy másikat a mappa nevéből kinyert adatokkal, ezek egyesítése a következő függvény a „dataSum" feladata. Az alap logikánk egyszerű, éppen ezért kivitelezésünk is: az adatok egyesítése során azaz objektum lesz a domináns amelyik több kulcsot tartalmaz, a több kulcs több kinyert információt is jelent. Az eredmény objektumunkba először a kevesebb kulcsú objektumot rakjuk bele aztán ráengedjük a több kulcsú objektumot, ilyenkor kulcs egyezés esetén a másodjára jövő objektum felülírja a már meglévő kulcsot, ezáltal az eredmény objektumban benne lesz mind a két objektum egyedi kulcsai (amik csak a fájlnév

objektumban és mappanév objektumban voltak), hasonló kulcsok esetén pedig a domináns, tehát több kulcsú objektum adatai maradnak meg. Ezáltal megkapjuk a metaadatok objektumot.

```
1 FNR
```

3.6. forráskód. Példa az fnr függvény eredményére és egyesítésére

Legvégül minden fent kiszűrt adatot összesítünk egy nagy objektumban, amely tehát tartalmazni fogja külön kulcsokra szétszedve a videó fájl nevét, kiterjesztését, elérési útvonalát, a felirat fájlok útvonalait (több is lehet), az NFO fájl útvonalát (egy lehet), a metaadatok objektumát, illetve egy egyedi ID azonosítót amelyet MD5-ös hasheléssel hozunk létre a videó fájl útvonalából, ez az azonosító az operációs rendszer által bizosítottan egyedi lesz, hiszen nem lehet 2 ugyanolyan nevű és kiterjesztésű fájl ugyanazon az útvonalon. Az eredményt pedig dispatcheljük a Redux storeba.

Az Offline scannelésből egy dolog maradt még hátra. Érdemes meggondolni ezen média tartalmak megjelenítését, az egy részet filmek megjelenítésével nincsen probléma, tekintve ezek egy videó fájlból állnak tehát egy ilyen média objektummal le tudjuk írni az összes hozzá köthető adatot. A probléma a többrészes sorozatoknál kezdődik ahol is több videó fájl található tehát több média objektum is. Éppen ezért megjelenítésnél nem az „all_media” kulcson tárolt nagy objektumokat fogjuk megjeleníteni, hanem készítünk egy „movies” és egy „tv_series” objektumot amelybe értelem szerűen a filmek és sorozatok kerülnek majd külön. Mindkét objektumba a média tartalmak ID azonosítóit fogjuk tenni, értelemszerűen filmek esetén egy egy azonosítót fog jelenteni, sorozatok esetében pedig midegyik epizód azonosítója bekerül. Az objektumok struktúrája mindkét esetben hasonló, hogy a megjelenítési feldoglozásnál ne kelljen külön lekezelni a két esetet. A 3.2-es forráskódban példát is hozunk minden fentebb leírtakra.

Online scan

Az Online scannelés keretein belül a The Movie Database[7] (továbbiakban TMDB) API-ját fogjuk használni, amely egy szabadon, ingyenesen felhasználható

[7] https://www.themoviedb.org/

interfészt biztosít a számunkra, továbbá az API egy Node-os impelentációját fogjuk még használni a „node-themoviedb" modult[8].

Az algoritmus végig iterál a összes beolvasott tartalmon és amennyiben talál IMDB azonosítót - fentebb már részleteztük, hogy mennyire fontos ez, hiszen így biztosan helyes információkat fogunk visszakapni - akkor az azonosító alapján megejti a kérési a TMDB API felé, amely helyes válaszküldés esetén, némi kissebb adatátalakítás - például az API-ból a műfajok szám azonosító formában jönnek vissza ezeket szövegeket tartalmazó tömbökké alakítjuk át - és kulcs átnevezés után bedispatcheljük a Redux storeban a metaadatok objektumba. A dispatchelés során ha egyező kulcsokat talál akkor a régi érték felülíródik az új értékkel, amennyiben pedig nincs egyezés úgy egyszerűen belekerülnek az objektumba.

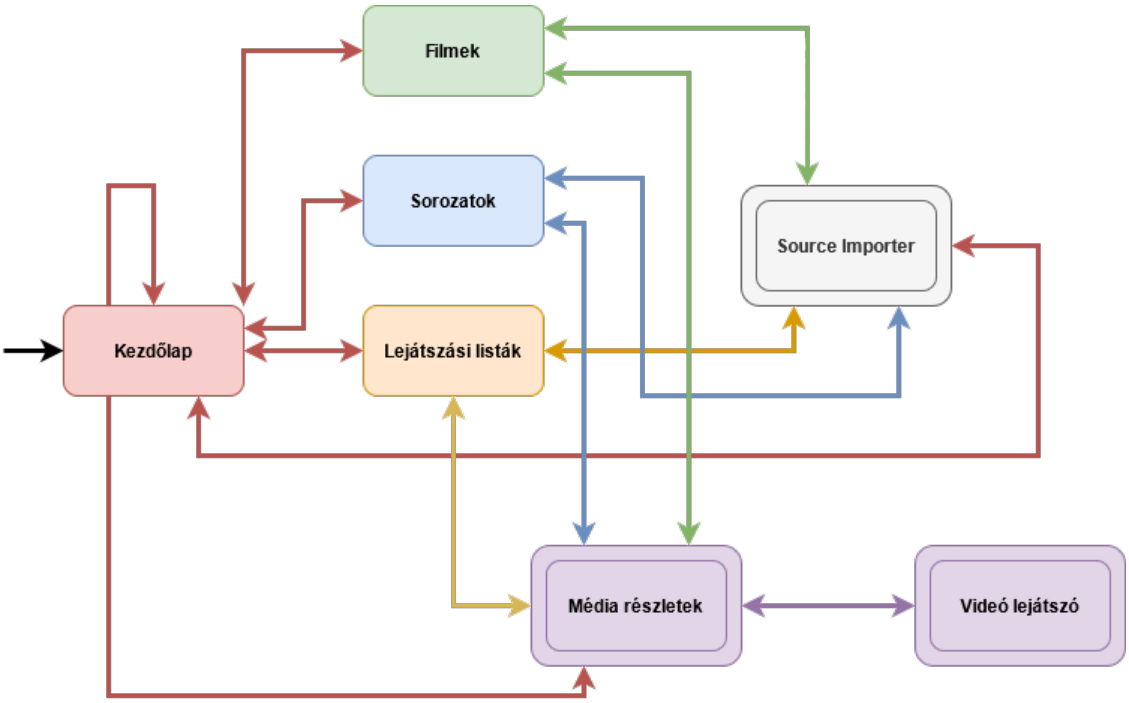
Amennyiben nem talál IMDB azonosítót, de talált évszámot és címet, akkor ezek elapján intéz kérést az API felé, helyes viszontválasz esetén pedig a már fentebb leírtak hajtódnak véghez ebben az esetben is, tehát kissebb adatátalakítás után a Redux storba illesztés történik.

Miután ezzel ekészült szintén lefut média típus szétválogatás mint az Offline scannelés végén, tehát a filmek azonosítói a „movies" objektumba másoldonak bele, a sorozatok epizódjainak azonosítói pedid egy a „tv_series" objektumon belüli tömmbe másolódnak bele.

3.2.4. Felhasználói felület ter e

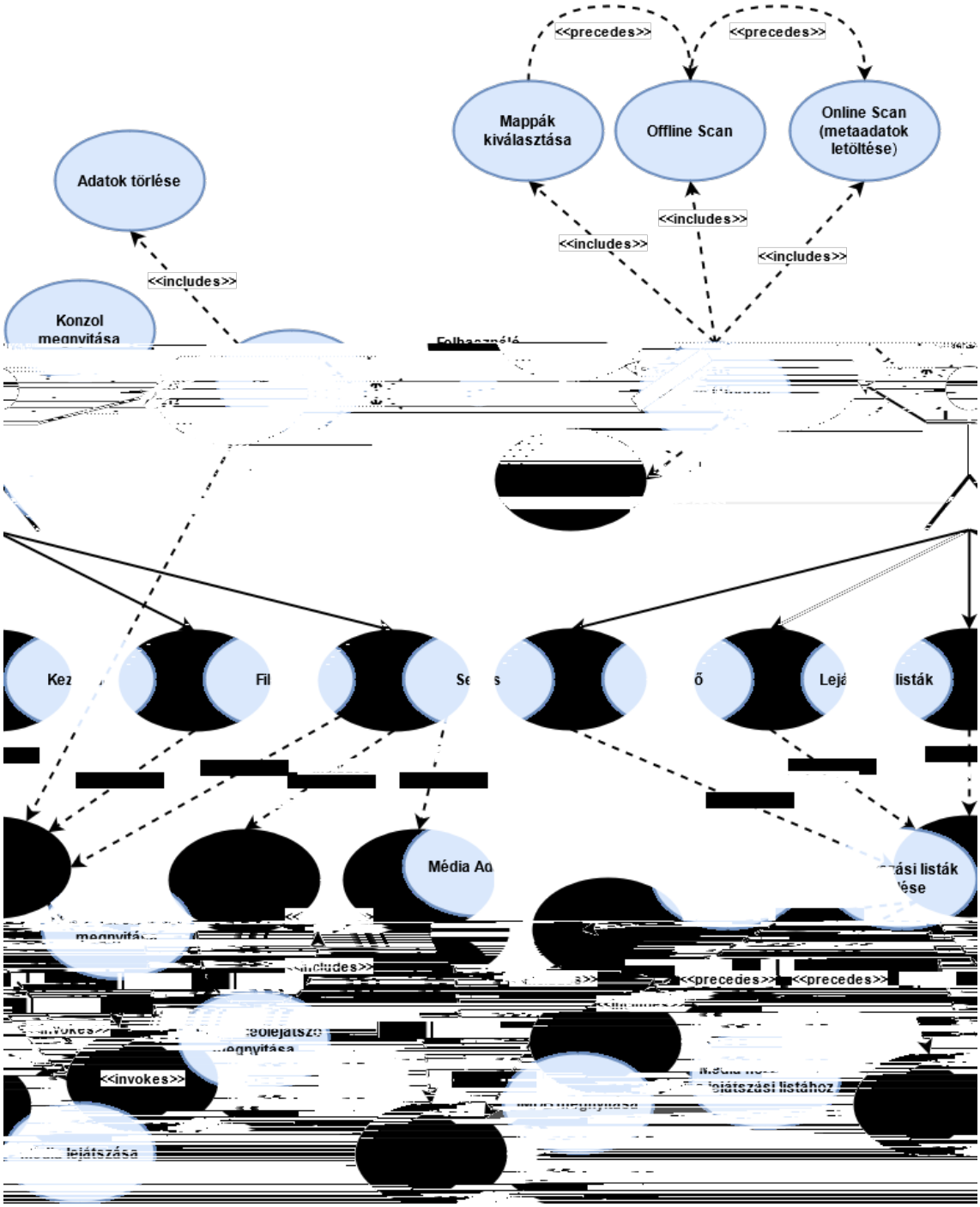
A felhasználói felület teljes mértékben React segítségével készült. Az alkalmazás felhasználói felületének belépési pontja az *index.html* és *index.tsx* fájl, amelyben a React render függvénye kerül meghívsára az App komponensen. Az App komponensen belül pedig a Layout komponensre megyünk tovább ahol a fő UI logika található. Az felhasználói felületnek két fő megjelenítési felülete lehet, az egyik az úgy nevezett Main, ami a fő ablak tartalmát hivatott leírni és változtatni, a másik pedig a Modal, amely a fő ablakon belüli, alárendelt, belső ablakot nyit meg. TODO

[8] `https://www.npmjs.com/package/node-themoviedb`



3.2. ábra. Navigáció az oldalak között

3.2.5. Használati esetek diagram



3.3. ábra. Használati esetek diagram

3.3. Implementáció

TODO

3.3.1. Funkcionális megközelítés

TODO

3.4. Tesztelés

A fejlesztés a legelejétől kezdve gyakorlatilag folyamatos kézi tesztelés mellett zajlott.

3.4.1. Hibás agy eredménytelen megközelítések

4. fejezet

Összegzés

Az alkalmazás tehát filmek és sorozatok vizualizációját tűzte ki célul

A. függelék

Továbbfejlesztési lehetőségek

Az alkalmazás természetesen még közel se tökéletes, rengeteg lehetőség áll még előtte.

Ábrák jegyzéke

Forráskódjegyzék