Професионална гимназия по компютърно моделиране и компютърни системи „Академик Благовест Сендов“Картина, която съдържа Графика, екранна снимка, изкуство, графичен дизайн

Описанието е генерирано автоматично

**ДИПЛОМНА РАБОТА**

**Тема:**

**Разработка на видеотека**

**Изготвил: Петър Станимиров Янков**

**СПЕЦИАЛНОСТ ОТ ПРОФЕСИЯ: „Системно програмиране”**

**ПРОФЕСИЯ: „Системен програмист”**

**ПРОФЕСИОНАЛНО НАПРАВЛЕНИЕ: „Компютърни науки”**

**Ръководител: Невена Иванова**

**Дата: 09.02.2024**

Варна 2024 г.

**Съдържание**

[1 Увод 5](#_Toc166487410)

[2 Цел 8](#_Toc166487411)

[2.1 Гледане на филми и сериали 8](#_Toc166487412)

[2.2 Уебсайт на български и английски език 8](#_Toc166487413)

[2.3 Интелигентно търсене 9](#_Toc166487414)

[2.4 Търсене по различни критерии 9](#_Toc166487415)

[2.5 Автоматизирано добавяне на филми и сериали 10](#_Toc166487416)

[2.6 Създаване на потребителски профили 10](#_Toc166487417)

[2.7 Верификация на профила 10](#_Toc166487418)

[2.8 Добавяне на опции към потребителските профили 11](#_Toc166487419)

[2.9 Промяна на информацията в профилите 11](#_Toc166487420)

[2.10 Харесване на филми и сериали 12](#_Toc166487421)

[2.11 Запазване в „Гледани“ 12](#_Toc166487422)

[2.12 Рейтинг на филми и сериали 12](#_Toc166487423)

[2.13 Добавяне на коментари 13](#_Toc166487424)

[2.14 Администраторски права 13](#_Toc166487425)

[2.15 Супер Администратор и неговите права 14](#_Toc166487426)

[2.16 Система за точки и нива 14](#_Toc166487427)

[2.17 Светла и тъмна тема 14](#_Toc166487428)

[3 Задачи 15](#_Toc166487429)

[3.1 Намиране на API за информация за филми и сериали 15](#_Toc166487430)

[3.2 Намиране на API за видео източници 15](#_Toc166487431)

[3.3 Разработване на функционалност за интелигентно търсене 15](#_Toc166487432)

[3.4 Разработване на система за създаване на потребителски профили 15](#_Toc166487433)

[3.5 Разработване на система за харесване на филми и сериали 16](#_Toc166487434)

[3.6 Разработване на система за запазване в "Гледани" 16](#_Toc166487435)

[3.7 Добавяне на секции за филми и сериали 16](#_Toc166487436)

[3.7.1 Популярни 16](#_Toc166487437)

[3.7.2 Трендинг 16](#_Toc166487438)

[3.7.3 Сега пуснати 17](#_Toc166487439)

[3.8 Създаване на страница за гледане на филм или сериал 17](#_Toc166487440)

[3.9 Добавяне на коментари под всеки филм или сериал 17](#_Toc166487441)

[3.10 Администраторски панел 17](#_Toc166487442)

[4 Структура на уебсайта 18](#_Toc166487443)

[4.1 Използвани технологии 18](#_Toc166487444)

[4.1.1 Cloudflare 18](#_Toc166487445)

[4.1.2 Nginx 18](#_Toc166487446)

[4.1.3 PM2 19](#_Toc166487447)

[4.2 Използвани API 19](#_Toc166487448)

[4.2.1 TheMovieDB API 19](#_Toc166487449)

[4.2.2 vidsrc.me API, vidsrc.to API и 2embed.cc 19](#_Toc166487450)

[4.2.3 mailersend API 19](#_Toc166487451)

[4.3 Фронтенд 20](#_Toc166487452)

[4.3.1 Езици за програмиране 20](#_Toc166487453)

[4.3.2 Библиотека Vite + ReactJS 21](#_Toc166487454)

[4.3.3 Пакети 21](#_Toc166487455)

[4.3.4 Файлове и технологии 23](#_Toc166487456)

[4.4 Бакенд 65](#_Toc166487457)

[4.4.1 Език за програмиране 65](#_Toc166487458)

[4.4.2 Пакети 66](#_Toc166487459)

[4.4.3 Файлове и технологии 67](#_Toc166487460)

[4.5 Use Case диаграма 92](#_Toc166487461)

[4.6 База данни 93](#_Toc166487462)

[База данни схеми 95](#_Toc166487463)

[5. Използвана литература 98](#_Toc166487464)

[5.1 ReactJS 98](#_Toc166487465)

[5.1 Vite 98](#_Toc166487466)

[5.2 ExpressJS 98](#_Toc166487467)

[5.3 TypeScript 98](#_Toc166487468)

[5.4 Mongoose 98](#_Toc166487469)

# Увод

С развитието на интернет технологиите и нарастването на броя на потребителите на онлайн видео съдържание, се появява остра нужда от платформи, които предлагат достъп до разнообразни филми и сериали безплатно. Това е отговор на нарастващата конкуренция от страна на платените абонаментни услуги за стрийминг на видео и изискванията на потребителите за по-голяма гъвкавост и избор. Такъв уебсайт би представлявал важен ресурс за потребителите, които не могат или не желаят да си позволят плащането на абонаменти особено в контекста на икономическите предизвикателства, които са налице в много общества по света.

Освен това, развитието на такъв уебсайт би отговаряло на нуждите на потребителите за по-широк достъп до културно съдържание и разнообразния в развлекателната сфера, като по този начин би спомогнало за повишаване на културната осведоменост и обогатяване на културния диалог. В крайна сметка, изборът на тази тема за дипломна работа има потенциала да допринесе за подобряване на достъпа до културно съдържание за много хора и за насърчаване на равен достъп до информация и развлечение в онлайн пространството.

Темата за разработка на уебсайт видеотека за безплатно гледане на филми и сериали съществува от ранните дни на развитието на интернет и онлайн видео съдържание. Въпреки че точният момент на започване не може да бъде определен с точност, първите уебсайтове за споделяне на видео материали без заплащане са се появили още през началото на 2000-те години. Тези уебсайтове често са били предназначение за споделяне на самостоятелно създадени видеоклипове или за обмен на телевизионни програми и филми между потребители.

С развитието на технологиите за видео стрийминг и увеличаването на интернет скоростите, уебсайтовете за безплатно гледане на филми и сериали са се разраствали и разраствали в по-голям мащаб. Те стават по-достъпни и популярни сред потребителите, които търсят алтернативни начини за гледане на любимите си филми и сериали без да плащат за тях. Така тази тема се развива и съществува вече няколко десетилетия, като продължава да привлича вниманието на потребители и разработчици по цял свят.

Развитието на темата за уебсайтове за безплатно гледане на филми и сериали е белязано от непрекъснатия технологичен напредък и изменения в потребителските предпочитания. От първите дни на споделянето на видео онлайн, когато уебсайтовете са качвали предимно потребителски генерирани съдържания, към по-късните години, когато се развиват платформи за стрийминг на филми и сериали, уебсайтовете за безплатно гледане започват да предлагат по-широка гама от съдържание.

С разрастването на интернет и увеличаването на броя на потребителите, уебсайтовете за безплатно гледане стават по-популярни, но също така и подлежат на по-строги правни ограничения и мерки за защита на авторското право. В резултат, някои от тях са принудени да прекратят дейността си, докато други преминават към по-иновативни модели за монетизация или сътрудничество с официални доставчици на съдържание.

Днес, уебсайтовете за безплатно гледане на филми и сериали продължават да привличат голям интерес, особено сред потребителите, които не могат или не желаят да плащат за абонаментни услуги. Въпреки трудностите, които срещат, тези уебсайтове продължават да бъдат важен източник на развлекателно съдържание за милиони потребители по света.

Темата за уебсайтовете за безплатно гледане на филми и сериали със сигурност събужда редица проблемни области и въпроси, с които трябва да се справят както разработчиците, така и потребителите.

Авторското право и законовата защита: Най-големият проблем за тези уебсайтове е нарушаването на авторското право. Голяма част от съдържанието, което се предлага безплатно е обект на законови ограничения и може да доведе до съдебни производства и затваряне на уебсайтове.

Качеството и надеждата на съдържанието: Много от уебсайтовете за безплатно гледане предлагат съдържание с ниско качество или са подложени на проблеми като загуба на видео кадри и забавяне на поточното възпроизвеждане.

Безопасността на потребителските данни: Някои уебсайтове за безплатно гледане могат да бъдат заплашени от проблеми със защитата на данните на потребителите, като например зловреден софтуер или събиране на лична информация без съгласието на потребителите.

Финансова устойчивост: За да се поддържа оперативността и качеството на услугата уебсайтовете за безплатно гледане често се нуждаят от реклами или други форми на финансиране, което може да доведе до прекомерно нахлуване на реклами или излагане на потребителите на потенциални опасности.

Легитимност и етика: Понякога съдържанието, което се предлага на тези сайтове може да бъде със съмнителен произход или да нарушава етични стандарти, като например пиратство, насилие, сексуално съдържание и други.

Тези проблемни области изискват сериозно внимание и усилия за разрешаване от страна на разработчиците, регулаторите и общността като цяло.

# Цел

Целта на уебсайта видеотека е да предостави на потребителите удобна и достъпна платформа за гледане на безплатни филми и сериали. Сайта се стреми да създаде пространство, където любителите на киното и телевизионните сериали могат да намерят широка гама от разнообразно съдържание, което да удовлетвори техните различни вкусове и предпочитания. Освен това, сайтът предлага удобен за потребителите начин за организиране на техните гледани и предпочитани филми и сериали, както и възможност за персонализиране на преживяването им в сайта. Със съчетаването на богат избор от съдържание, интуитивен интерфейс и функционалности за потребителско управление, сайтът се стреми да предоставя на потребителите пълноценно и приятно изживяване в света на киното и сериалите, безплатното и на всяко време.

## Гледане на филми и сериали

Потребителите на сайта ще имат възможност да гледат филми и сериали директно от уебсайта, без да е необходимо да изтеглят или инсталират допълнителен софтуер. Гледането на съдържание ще бъде лесно и интуитивно, като потребителите ще могат да навигират в уебсайта, като търсят конкретни заглавия или да разглеждат по категории, жанрове, и други характеристики. Потребителите могат да намират и да споделят любимите си филми и сериали с приятели и семейство, чрез споделяне на линк към сайта. Освен това, сайтът осигурява опция за регулиране на качеството на видеото и в зависимост за добавяне на субтитри, за да се удовлетворят предпочитанията на потребителите. Тази платформа е създадена за цел да предостави на потребителите удобство и възможност за откриване на разнообразни филмови и телевизионни произведения, като същевременно насърчава обмена на културни интереси и преживявания.

## Уебсайт на български и английски език

Изборът за предоставяне на уебсайта на два езика е направен с цел осигуряването на максимално удобство и достъпност за широк кръг от потребители. Независимо от техните езикови предпочитания, потребителите ще могат да се възползват от платформата и да имат достъп до желаните от тях съдържания на предпочитания от тях език. Тази функционалност не само улеснява потребителите в тяхната навигация по сайта, но и помага за по-широко разпространение на платформата в различни културни среди с различни езикови предпочитания.

## Интелигентно търсене

Сайтът ще предоставя възможност за търсене по част от името на филм или сериал. Това означава, че потребителите ще могат да въвеждат част от заглавието на желания от тях филм или сериал в полето за търсене, а не е обходимо да въвеждат цялото му точно заглавие. Системата ще предложи съответващи резултати, които съдържат въведената от потребителя част от името, което прави процеса на намиране на желаното съдържание по-лесен и по-бърз. Тази функционалност е създадена с цел улеснена навигация на потребителите в уебсайта и да им позволи бързо да откриват желаното съдържание, дори ако не помнят точното му заглавие.

## Търсене по различни критерии

Сайтът ще предоставя възможност за търсене на филми и сериали по различни критерии, като например име на филма или сериала, жанр, държава на производство и година на издаване. Потребителите ще могат да използват тази функционалност, за да филтрират и намерят конкретни съдържания, които отговарят на техните предпочитания и интереси. Търсенето по различни критерии позволява на потребителите да намерят желаното съдържание по-бързо и по-лесно. Тази функционалност е създадена с цел да улесни и подобри потребителския опит на сайта, като предоставя широка гама от опции за навигация и откриване на съдържание.

## Автоматизирано добавяне на филми и сериали

Когато потребителят използва уебсайта и търси определен филм или сериал, ако този филм или сериал все още не се намира в базата данни на уебсайта, системата автоматично извършва търсене във външни източници на информация като например бази данни за филми и сериали. Ако информация за търсения филм или сериал бъде открита, тя се добавя автоматично към базата данни на уебсайта, така че потребителите да могат да го намерят и да го гледат. Тази функционалност осигурява на потребителите достъп до широка гама от филми и сериали, дори ако те не са предварително добавени в системата от администраторите или други потребители. Тя подобрява потребителския опит, като предоставя възможност за намиране на желаното съдържание по-лесно и по-бързо.

## Създаване на потребителски профили

Уебсайтът ще предлага възможност за създаване на персонализирани потребителски профили. Чрез създаването на профил, потребителите ще могат да получат персонализирано преживяване на платформата, като получават достъп до различни функционалности и възможности за персонализация. Потребителските профили позволяват на потребителите да запазят своите предпочитания и настройки, както и да управляват списъците си с харесани или гледани съдържания. Създаването на потребителски профили е ключова функционалност, която улеснява персонализацията и социалната интеракция на потребителите на уебсайта, като им предоставя по-богат и ангажиращ потребителски опит.

## Верификация на профила

Уебсайтът ще предлага възможност за верификация на потребителските профили. Този процес на верификация има за цел да потвърди идентичността на потребителите и да укрепи сигурността на платформата. Потребителите могат да верифицират профила си чрез отиването до линка предоставен от изпратения при регистрация имейл. Верифицираните профили обикновено получават допълнителни привилегии или функционалности, които не са достъпни за не верифицираните потребители. Тази функционалност е създадена с цел да осигури по-голяма сигурност и доверие в общността на потребители на уебсайта, като същевременно защити от нежелани активности и злоупотреби.

## Добавяне на опции към потребителските профили

Сайтът ще предоставя възможност за персонализация на потребителските профили чрез добавяне на различни опции. Потребителите ще могат да избират от разнообразие от настройки и функционалности, които да съответстват на техните предпочитания и нужди. Тези опции включват настройки за език и регион, предпочитания за уведомления и други. Освен това, потребителите ще могат да изберат дали желаят автоматично запазване на гледаните съдържания в техния профил. Тази опция им дава контрол върху това дали да запазват автоматично информация за гледаните филми и сериали, което може да им помогне да се организират и да следят своите гледани съдържания по-ефективно. Освен това, потребителите ще имат възможност да изберат между публичен и личен акаунт. Публичният акаунт позволява на другите потребители да виждат информацията в профила, докато личният акаунт запазва информацията само за потребителя или ограничена група хора. Тези опции за персонализация на профилите целят да предоставят на потребителите по-голяма гъвкавост и контрол върху техния потребителски опит в уебсайта.

## Промяна на информацията в профилите

Сайтът ще предоставя възможност за потребителите да променят информацията в своите потребителски профили. Това включва различни опции за актуализиране на личните данни, настройките за сигурност и предпочитанията за уведомления. Потребителите ще могат да променят своето потребителско име и парола за вход, да обновят контактната си информация, като електронна поща. Тази функционалност е създадена с цел да предостави на потребителите контрол върху техния потребителски профил и да ги улесни в актуализирането на информацията си, за да съответства на техните текущи нужди и предпочитания.

## Харесване на филми и сериали

Сайтът ще предоставя възможност на потребителите да отбелязват филми и сериали, които им харесват, за да създават персонализиран списък с предпочитани съдържания. Когато потребител хареса определено филмово или серийно заглавие, то се добавя към техния списък с „Харесани“, което им позволява по-лесно да го открият и достъпят в бъдеще. Това предоставя на потребителите удобен начин за организиране на техните предпочитания и интереси в света на киното и сериалите.

## Запазване в „Гледани“

Когато потребителят избере да гледа определен филм или сериал, платформата ще му предостави възможността да запази това съдържание в раздела „Гледани“. При сериалите системата ще запомни на кой последен сезон и епизод е потребителят спрял гледането си. Това ще позволи на потребителите по-лесно да продължат гледането от точката, на която са спрели без да е необходимо търсене или преглеждане на списъци с епизоди. В секция „Гледани“ потребителите ще могат да видят списък със запазените от тях филми и сериали, като това ще ги улесни в организацията и управлението на техните предпочитания. Тази функционалност е създадена с цел да осигури на потребителите по-голямо удобство и персонализация на техния престой в уебсайта, като същевременно подобри тяхното потребителско преживяване.

## Рейтинг на филми и сериали

Потребителите ще имат възможност да добавят свой рейтинг към филми и сериали, което ще допринесе за по-добра видимост и справедливост при преглеждането на съдържанието. Този персонализиран рейтинг може да бъде базиран на личните им преференции и впечатления от гледането на определен филм или сериал. Потребителите могат да дадат оценка от определен брой звезди или да използват други методи за оценка на качеството на съдържанието. Техният рейтинг може да бъде визуализиран на уебсайта и да бъде използван за препоръки и класации, които да помогнат на други потребители при избора на филми и сериали за гледане. Тази функционалност дава на потребителите възможността да споделят своите мнения и впечатления с общността и да се възползват от техните препоръки при избора на съдържание за гледане.

## Добавяне на коментари

Потребителите ще имат възможност да добавят коментари към филми и сериали. Това им дава възможността да споделят своите мнения, впечатления и препоръки с другите потребители на уебсайта. Коментарите могат да бъдат използвани за обсъждане на различни аспекти на филмите и сериалите, като например сюжет, персонажи, режисура и други. Те също така могат да служат за създаване на интерактивна общност от любители на кино и телевизия, които да споделят общ интерес и страст към различните видове съдържание. Коментарите могат да бъдат видими за другите потребители на уебсайта и да се използват за управление на дискусии и обмен на идеи и мнения. Тази функционалност допринася за по-богато и ангажиращо потребителско преживяване на уебсайта и насърчава активното участие на потребителите в общността.

## Администраторски права

Администраторите ще имат специални права и достъп до административен панел, който им позволява да управляват различни аспекти на уебсайта. Те ще бъдат отговорни за изтриване на нежелани или неподходящи коментари, както и за управление на акаунти, които не спазват правилата на общността. Администраторите ще имат възможността да изтриват коментари и да деактивират акаунти, които не отговарят на стандартите на уебсайта, като по този начин поддържат ред и безопасност в общността. Важно е да се отбележи, че само супер администраторите ще имат права да управляват другите администратори и да извършват по-значителни промени в системата.

## Супер Администратор и неговите права

Супер администраторът ще бъде единственият администратор с пълни права на управление на уебсайта. Той ще притежава всички права и функционалности на обикновените администратори, като същевременно ще има и възможността да назначава, управлява и премахва други администратори. Това включва контрол върху техните права и достъп до административната панела, както и възможността да управлява техните действия и активности. Супер администраторът ще бъде отговорен за поддържането на ред и безопасност в административния аспект на уебсайта, както и за назначаването на подходящи администратори за изпълнение на различни административни задачи.

## Система за точки и нива

Потребителите ще имат възможност да събират точки, като гледат филми и сериали на уебсайта. За всяко гледане на съдържание потребителите ще получават определен брой точки, които ще се добавят към техния акаунт. Тези точки могат да бъдат използвани за различни цели и привилегии в уебсайта. Освен това, потребителите ще имат възможност да покачват нива в системата чрез натрупване на точки. Системата за точки и нива ще насърчи активно участие на потребителите в уебсайта и ще им предостави по-голяма мотивация да използват различните функционалности за гледане на съдържание.

## Светла и тъмна тема

Потребителите ще имат възможност да изберат между светла и тъмна тема за дизайна на уебсайта. Тази функционалност ще позволи на потребителите да персонализират визуалния си опит в зависимост от техните предпочитания и настройки. Светлата тема може да бъде предпочитана за по-ярко и леко преживяване, докато тъмната тема е подходяща за по-контрастно и комфортно гледане, особено при ниска осветеност на екрана или при продължително използване през нощта. Потребителите ще могат лесно да превключват между различните теми според техните предпочитания и условия, което ще им предостави по-удобен и приятен интерфейс за използване на уебсайта.

# Задачи

## Намиране на API за информация за филми и сериали

Трябва да се направи проучване и да се открие подходящ API, който предоставя информация за филми и сериали, включително данни като заглавия, описания, години на издаване, държави на издаване и т.н.

## Намиране на API за видео източници

Трябва да се намери API, който предоставя видео източници за филми и сериали, които могат да бъдат възпроизведени на уебсайта. Това включва намиране на платформи или услуги, които предлагат легално поточно предаване на филми и сериали.

## Разработване на функционалност за интелигентно търсене

Трябва да се разработи алгоритъм или модул, който позволява на потребителите да извършват интелигентно търсене на филми и сериали по различни критерии като заглавие, жанр и други.

## Разработване на система за създаване на потребителски профили

Трябва да се разработи функционалност, която позволява на потребителите да създават и управляват свои потребителски профили, като в тях могат да запазят предпочитанията си за гледане, да следят своите гледани филми и сериали и други.

## Разработване на система за харесване на филми и сериали

Трябва да се разработи функционалност, която позволява на потребителите да харесват филми и сериали, които са им харесали, което може да бъде използвано за персонализиране на препоръките за съдържание.

## Разработване на система за запазване в "Гледани"

Трябва да се разработи функционалност, която позволява на потребителите да запазват филми и сериали в списък с "Гледани", което им позволява по-лесно да следят какво вече са гледали и какво все още искат да гледат.

## Добавяне на секции за филми и сериали

### Популярни

Тази секция ще включва списък с най-популярните филми и сериали, които са в момента горещи сред потребителите. Филмите и сериалите в тази секция ще бъдат избрани въз основа на техния брой гледания или рейтинг. Секцията ще разполага с множество страници, за да покрие широк спектър от популярно съдържание.

### Трендинг

В тази секция потребителите ще могат да открият най-актуалните и популярни филми и сериали, които в момента се разглеждат широко от аудиторията. Филмите и сериалите в тази секция ще бъдат избрани въз основа на техния растящ интерес и популярност. Секцията ще бъде организирана в множество страници, за да предостави обхватна селекция от трендове.

### Сега пуснати

Тази секция ще представя списък с най-новите филми и сериали, които са били скоро пуснати и са на разположение за гледане. Това ще даде възможност на потребителите да бъдат в течение на последните премиери и да откриват нови заглавия за гледане. Секцията ще включва множество страници, за да покрие широк спектър от свежо съдържание.

## Създаване на страница за гледане на филм или сериал

Трябва да се разработи страница, която да позволява на потребителите да гледат избрания от тях филм или сериал. Страницата трябва да включва опции за възпроизвеждане, пауза, превъртане напред и назад, както и други функционалности за управление на видеото.

## Добавяне на коментари под всеки филм или сериал

Трябва да се разработи функционалност, която позволява на потребителите да добавят коментари и отзиви под всяко филмово или сериално заглавие. Това ще даде възможност за обмен на мнения и препоръки между потребителите и ще увеличи ангажираността на аудиторията.

## Администраторски панел

Трябва да се разработи администраторски панел, който да позволява на администраторите да управляват съдържанието на уебсайта. Това включва добавяне, редактиране и изтриване на филми и сериали, управление на коментарите и потребителските профили, както и други административни задачи.

# Структура на уебсайта

## Използвани технологии

### Cloudflare

HTTPS сигурност: Cloudflare предоставя SSL сертификати, които позволяват на уебсайта ти да се зарежда през HTTPS протокола. Това осигурява шифрована връзка между потребителя и сървъра, което е от съществено значение за поверителността на данните и сигурността на информацията, която се предава през уебсайта ти.

Ускоряване на достъпа до уебсайта: Cloudflare предлага различни оптимизации, които увеличават скоростта на зареждане на уебсайта. Това включва кеширане на статично съдържание, оптимизация на изображенията и минификация на CSS и JavaScript файлове. Като резултат, потребителите ще имат по-бърз и по-приятен опит при използване на уебсайта ти.

Защита от DDoS атаки: Cloudflare предоставя мощни инструменти за защита срещу различни видове атаки, включително DDoS атаки. Технологиите на Cloudflare могат да филтрират и блокират зловреден трафик, преди да стигне до твоя сървър, което осигурява непрекъснато функциониране на уебсайта ти, дори при атаки.

### Nginx

Nginx е уеб сървър, който играе ключова роля в създаването на сигурна и надеждна инфраструктура за нашия уебсайт. Той действа като връзка между интернет и нашия уебсайт, управлявайки входящите и изходящите заявки, осигурявайки сигурност, бързина и надеждност. В нашия случай, Nginx е конфигуриран да осигурява няколко ключови функции. Първо, той пренасочва заявките от публичната мрежа към нашия частен сървър, което допълнително засилва защитата на нашия уебсайт. Също така, Nginx осигурява SSL/TLS защита, която криптира комуникацията между потребителите и нашия сървър, гарантирайки поверителността на данните и сигурността на нашите потребители.

### PM2

PM2 е процесен управляващ уебсървър, който се използва за стабилен и ефективен контрол на Node.js приложения и техните процеси. Той предлага функции като автоматичен рестарт на приложенията при грешки, мониторинг на процесите и балансиране на товара, което го прави подходящ инструмент за поддръжка на уебсървъри. В нашия случай, използваме PM2 за поддръжка на както фронтенд, така и бакенд частите на уебсайта, осигурявайки стабилна и надеждна работа на целия проект.

## Използвани API

### TheMovieDB API

themoviedb API е услуга, която предоставя достъп до база данни с информация за филми, сериали, актьори и други филмови данни. Тя предоставя различни ендпойнти, които позволяват на разработчиците да търсят, филтрират и извличат информация за филми и сериали. Използва се широко от уеб и мобилни приложения, които изискват информация за филми или сериали, като например уебсайтове за стрийминг на видео или мобилни приложения за каталоги на филми.

### vidsrc.me API, vidsrc.to API и 2embed.cc

vidsrc.me API, vidsrc.to API и 2embed.cc представляват API-и, които предоставят достъп до различни видео съдържания, като филми и сериали. Те могат да предоставят възможности за търсене, поточно възпроизвеждане, филтриране и други операции, свързани с видео съдържанието. Тези API-и се използват от уебсайтове и приложения за гледане на филми и сериали онлайн. Разработчиците могат да ги използват, за да интегрират видео съдържание в своите проекти.

### mailersend API

mailersend API е услуга за изпращане на имейли, която предоставя различни ендпойнти за създаване, изпращане и проследяване на имейли. Тя предоставя възможности за персонализиране на имейлите, управление на списъци с получатели, анализ на доставките и други функции, свързани с електронната поща. Използва се от разработчици и компании, които изпращат масови или персонализирани имейли, като например имейл маркетинг кампании, известия за потребители или системни известия.

## Фронтенд

Фронтенд е частта от уебсайта, с която потребителят взаимодейства директно. Това включва всичко, което потребителят вижда и използва в браузъра, като например интерфейса, бутоните, формите и други елементи за навигация и взаимодействие. Фронтенд разработчиците работят по създаването на удобен и атрактивен интерфейс, който да предоставя на потребителите приятен и лесен за използване опит.

### Езици за програмиране

#### HTML (HyperText Markup Language)

HTML е стандартен език за маркиране, използван за създаване на структура и съдържание на уеб страниците. Той дефинира различни елементи и техните атрибути, които се използват за представяне на текст, изображения, форми и други елементи на уеб страницата.

#### CSS (Cascading Style Sheets)

CSS е език за стилизиране, който се използва за определяне на визуалния вид и представяне на уеб страниците. Той позволява задаване на различни стилове като цветове, шрифтове, размери и позициониране на елементите на страницата.

#### JavaScript

JavaScript е високо нивов език за програмиране, който се използва широко за разработка на динамични уеб приложения. Той позволява добавяне на интерактивност към уеб страниците, манипулиране на DOM елементи и взаимодействие с потребителите на страницата.

#### TypeScript

TypeScript е супермножество на JavaScript, което добавя статично типизиране и други напредни функции към езика. Той предоставя по-голяма сигурност и устойчивост на грешки по време на разработката, като същевременно запазва синтаксиса и функционалността на JavaScript. TypeScript е широко използван в уеб разработката за създаване на по-големи и по-сложни уеб приложения.

### Библиотека Vite + ReactJS

Vite е модерен инструмент за създаване на уеб приложения, който осигурява изключителна скорост на разработка и сборка. Съчетавайки го с библиотеката за създаване на потребителски интерфейси React JS, можем да създадем бързи, мощни и интерактивни уеб приложения. Vite предоставя предварително компилиране на кода и динамично зареждане на модули по време на разработка, което осигурява бърза и ефективна работа с React JS компонентите. Този комбиниран подход улеснява създаването на модерни уеб приложения с висока производителност и качество.

### Пакети

#### nextUI

Библиотека, която предоставя готови компоненти за създаване на потребителски интерфейси в уеб приложенията, използвайки ReactJS. Тя включва разнообразие от компоненти като бутони, форми, карти и други, които са стилово и функционално оптимизирани за употреба в приложенията.

#### framer-motion

Библиотека за анимации в React приложения, която предоставя прост и мощен начин за добавяне на интерактивни и динамични анимации към компонентите. Тя предлага гъвкави API-та за дефиниране на различни видове анимации като промяна на позицията, мащабиране, завъртане и други.

#### js-cookie

JavaScript библиотека, която предоставя лесен начин за управление на бисквитките в уеб приложенията. Тя позволява на разработчиците да създават, четат и изтриват бисквитки, което е полезно за работа със съхранение на данни на клиентската страна.

#### lucide-react (icons)

Библиотека с икони за употреба в уеб приложенията, представени като компоненти в React. Lucide предлага широк набор от векторни икони с модерен дизайн, които могат да бъдат лесно вграждани в приложенията и стилизирани по желание.

#### next-themes

Пакет, който предоставя възможност за темна и светла тема в Next.js приложенията. Той предоставя готова интеграция с Next.js и React, позволявайки на потребителите да превключват между различни теми на уебсайта.

#### react-router-dom

Библиотека за маршрутизация в React приложенията, която предоставя инструменти за дефиниране на маршрути и навигация между различните страници и компоненти в приложението. Тя позволява на разработчиците да създават динамични и интерактивни приложения с множество страници.

#### tailwindcss

CSS библиотека, която предоставя набор от готови класове за стилизиране на уеб елементите. TailwindCSS е базиран на концепцията на utility-first CSS, където вместо да пишете CSS правила, използвате готови класове за стилизиране на елементите. Това прави разработката по-бърза и ефективна, като същевременно предлага голяма гъвкавост при стилизирането на уеб приложенията.

### Файлове и технологии

#### “src” папка

В папката "src" в React.js приложението се съдържат основните файлове и директории, които са част от изходната структура на проекта. Включените файлове и директории обикновено включват:

**main.tsx**

Файлът "main.tsx" е основният входен файл за вашия React приложение. Той използва "ReactDOM.createRoot" за рендиране на основния компонент на приложението - компонента "App" върху кореновия елемент в HTML документа, който е намерен с помощта на "document.getElementById('root')!".

Това е стандартният начин за стартиране на React приложение с React 18 или по-нова версия, където използваме "createRoot" вместо "render". Файлът също така включва:

* Импортиране на основния компонент на приложението от "App.tsx".
* Импортиране на стиловете на приложението от "index.css".
* Вграждане на компонента за темиране на приложението от "next-themes".
* Вграждане на компонента за предоставяне на UI компоненти от "NextUIProvider".
* Вграждане на компонента за маршрутизация от "react-router-dom".
* Вграждане на компонента за предоставяне на контекст от "AppProvider".

Основната цел на "main.tsx" е да конфигурира и стартира React приложението, като го свърже с външни библиотеки и компоненти, необходими за функционирането му.

**App.tsx**  
Файлът "App.tsx" е основният компонент на вашия React приложение. Той дефинира основната структура на приложението и определя какви компоненти да се показват на различни маршрути. Ето какво прави:

* Импортира необходимите компоненти и библиотеки, включително компоненти за навигация, контекст и линк към страниците на приложението.
* Дефинира основни маршрути за различните страници на приложението, като използва компонентите, които се зареждат лениво (lazy loading) с помощта на "Suspense".
* Показва компонента за зареждане, докато приложението се инициализира и зарежда началните данни.
* Връща основния компонент на приложението, включващ навигационната лента, маршрутите и подвалния елемент.

Компонентът "ErrorComp" се използва за обработка на ненамерени страници (404 грешка) и предоставя възможност за връщане към началната страница. Той се показва при грешка в маршрута или при опит за достъп до невалиден URL адрес.

#### “assets” папка

Тук се съдържат статичните ресурси на приложението като изображения, шрифтове и други медийни файлове. Те могат да бъдат използвани в компонентите на приложението за стилизиране и визуализация.

#### “components” папка

Това е директорията, където се съдържат всички компоненти на приложението. Всеки компонент обикновено е разделен в отделен файл или директория, която съдържа своите собствени файлове за компоненти.

Файлът „**CardComponent.tsx**“ е компонент, който представлява карта за показване на информация за филм или сериал във вашето React приложение. Ето какво прави този файл:

* Импортира необходимите компоненти от библиотеката @nextui-org/react, lucide-react и други. Също така, импортира интерфейса mediaType и контекста AppContext.
* Дефинира типовете на пропсите, които приема компонента - data, onOpen, setMediaId и isMovie.
* Дефинира функции за обработка на данни като handleReleaseDate и handleVoteAverage.
* Използва контекста AppContext, за да получи информация за езика по подразбиране на приложението.
* Връща JSX структура, която представлява картата. Картата включва изображение, заглавие, дата на излизане, оценка, тип на медията (филм или сериал) и друга информация.
* Използва се мапиране на данните от пропсите на компонента, за да се покажат съответните данни.
* Този компонент се използва във вашето приложение, за да покаже информация за филмове или сериали на различни места, като например на началната страница, в резултатите от търсене или в други секции.

Файлът „**Footer.tsx“** е компонент, който представлява долния колонтитул (footer) на уеб страницата във вашия React приложение. Ето какво прави този файл:

* Импортира необходимите компоненти и контекст от други файлове във вашия проект, като **AppContext** от **../../utils/AppContext** и други компоненти от **@nextui-org/react**.
* Използва контекста **AppContext**, за да получи информация за темата на приложението.
* Определя цветовата схема на footer компонента в зависимост от избраната или системната тема на приложението.
* Връща JSX структура, която представлява долния колонтитул на страницата. Включва линк към началната страница, година на копирайта, линк към DMCA защитата и друга информация за копирайт.
* Включва линк към началната страница, линк към DMCA защитата и друга информация за копирайт.

Този компонент се използва за добавяне на долния колонтитул на страницата във сайтът, който съдържа важна информация като копирайт, линкове към защита на авторски права и други важни ресурси.

„**handleFunctions.tsx**” представлява модул, който съдържа много различни функции, свързани с извличането и обработката на данни от уеб сървър чрез HTTP заявки. Ето какво правят различните функции в този файл:

* **GetTheme**: Връща текущата тема на приложението, която е запазена в локалното съхранение (localStorage). Ако няма такава тема, връща "system".
* **Login**: Изпраща POST заявка към сървъра за аутентикация на потребител. Проверява се отговорът от сървъра и ако е успешен, запазва токен в бисквитките на браузъра.
* **VerifyUser**: Изпраща GET заявка към сървъра за потвърждение на потребителския токен.
* **UpdateProfile**: Изпраща POST заявка към сървъра за обновяване на потребителския профил.
* **Register**: Изпраща POST заявка към сървъра за регистрация на нов потребител.
* **Language**: Запазва избран от потребителя език в локалното съхранение.
* **Logout**: Изпраща GET заявка към сървъра за излизане от профила.
* **ToggleWatchList**: Променя състоянието на списъка с гледани медии за даден потребител.
* **QuickSearch**: Изпраща GET заявка към сървъра за бързо търсене на медии по ключова дума.
* **Discover**: Изпраща GET заявка към сървъра за откриване на медии по различни критерии като жанр, година и други.
* **ToggleLikedFunc**: Променя състоянието на харесване за дадена медия от страна на потребителя.
* **ToggleIsWatched**: Променя състоянието на медията, дали е гледана или не, от страна на потребителя.
* **AddComment**: Изпраща GET заявка към сървъра за добавяне на коментар към медия.
* **DeleteComment**: Изпраща GET заявка към сървъра за изтриване на коментар.
* **ReplyComment**: Изпраща GET заявка към сървъра за отговор на даден коментар.
* **ViewMoreComments**: Изпраща GET заявка към сървъра за зареждане на още коментари.
* **ToggleIsPublic**: Променя състоянието на профила на потребителя дали е публичен или не.
* **PopularTV**, **TrendingTV**, **NowPlayingTV**: Изпращат GET заявки към сървъра за получаване на списъци с популярни, тенденции и в момента излъчвани телевизионни предавания.
* **TVChange**: Променя текущия списък с телевизионни предавания, базиран на избрания език.
* **ResendVerifyEmail**: Изпраща GET заявка към сървъра за преизпращане на имейл за потвърждение.
* **GetScore**: Изпраща GET заявка към сървъра за получаване на резултат на потребителя.
* **AddRated**: Изпраща GET заявка към сървъра за добавяне на рейтинг за медия.
* **GetLiked**, **GetWatched**: Изпращат GET заявки към сървъра за получаване на списъци с харесани и гледани медии от страна на потребителя.
* **DeleteOne**, **DeleteLiked**, **DeleteWatched**: Изпращат GET заявки към сървъра за изтриване на профил, харесани и гледани медии от страна на потребителя.
* **AccountFinder**: Изпраща GET заявка към сървъра за намиране на профил по потребителско име.
* **DiscordToken**: Изпраща GET заявка към сървъра за синхронизиране на потребителски профил с Discord.
* **MediaDetails**: Изпраща GET заявка към сървъра за получаване на детайли за медия (филм или сериал).
* **MediaPreview**: Изпраща GET заявка към сървъра за получаване на преглед на медия (филм или сериал).
* **MediaTrailer**: Изпраща GET заявка към сървъра за получаване на трейлър на медия (филм или сериал).
* **FindAll**: Изпраща GET заявка към сървъра за намиране на всички потребители.
* **ChangeRoleAdmin**, **DeleteOneAdmin**: Изпращат GET заявки към сървъра за промяна на ролята на администратор и изтриване на администраторски профил.

Тези функции са организирани във файл, за да бъдат лесно достъпни и използвани от други компоненти и модули във вашия проект.

“**Loader.tsx“,** използва се за показване на индикатор за зареждане за потребителите по време на извършване на асинхронни операции. Ето обяснение на кода:

* **LoaderProps**: Интерфейсът описва пропсите, които компонентът Loader приема. В случая има само един пропс - **lang**, който определя езика на зареждащия текст.
* **Loader**: Компонентът **Loader** е функционален компонент, който приема **LoaderProps** като аргумент и връща JSX елемент. В зависимост от езика (**lang**), компонентът показва зареждащ текст и индикатор за зареждане.
* **CircularProgress**: Компонентът **CircularProgress** е взет от пакета **@nextui-org/react** и се използва за създаване на кръгов индикатор за зареждане.
* **LoaderText**: Обектът **LoaderText** съдържа текстовите стойности за зареждащия текст и атрибута **aria-label** за индикатора за зареждане за различните езици. Този обект се използва за определяне на текста в зависимост от езика.

Компонентът **Loader** е добре структуриран и предлага гъвкавост при превключване на езици. Използвайки този компонент, вашият проект може лесно да предостави потребителска преживяване на зареждане, което е по-приятно за потребителите.

Файлът **MediaList.tsx** представлява компонент, който показва списък с медийни елементи (например филми или сериали). Ето обяснение на кода:

* **MediaListTypes**: Типът **MediaListTypes** описва формата на пропсите, които компонентът **MediaList** приема. Тези пропси включват заглавие (**title**), данни (**data**), функция (**onOpen**) и флаг (**isMovie**), указващ дали медийните елементи са филми.
* **MediaList**: Компонентът **MediaList** е функционален компонент, който приема пропсите, описани в **MediaListTypes**. Той показва заглавие (**title**), следвано от картички за всеки медиен елемент от списъка, предоставен в **data**. За всяка картичка се извиква компонентът **CardComponent**, който приема данните за съответния медиен елемент, както и други пропси.
* **CardComponent**: Компонентът **CardComponent** се извиква за всяка картичка, която се показва в списъка. Той е отговорен за представянето на информацията за медията и съдържа логика за отваряне на медията при клик.
* **AdsComponent**: Компонентът **AdsComponent** се показва след списъка с медийни елементи и служи за показване на реклами.

Този компонент е добре структуриран и предоставя мощност за показване на списъци с медийни елементи във вашето приложение. Включването на рекламен компонент след списъка добавя стойност към потребителското преживяване, като предоставя разнообразие и интерактивност.

Файлът **MediaModal.tsx** съдържа компонент, който представлява модално прозорец, показващ детайли за определен медиен елемент, като филм или сериал. Ето обяснение на кода:

* **MediaModalProps**: Този тип определя пропсите, които компонентът **MediaModal** приема. Те включват **mediaId** (идентификатор на медията), **isMoviesMedia** (флаг, указващ дали медията е филм или сериал), **isOpen** (флаг, указващ дали модалният прозорец е отворен) и **onOpenChange** (функция за промяна на статуса на отваряне на модалния прозорец).
* **MediaModal**: Това е функционален компонент, който показва модалния прозорец. В този компонент се извличат данни за медията, като заглавие, описание, рейтинг, трейлър и други. Също така, в този компонент се извикват различни функции за манипулиране на данните за медията, като форматиране на дати и обработка на рейтинг.
* **Loading State**: Когато се зареждат данни за медията, се показва индикатор за зареждане.
* **ToggleLikedFunc**: Функцията **handleToggleLiked** се извиква при клик върху бутона за добавяне в любими. Тя превключва състоянието на иконата за любимо и прави съответния HTTP заявка към API.
* **Rendering Media Details**: Детайлите за медията се показват в модалния прозорец, включително заглавие, рейтинг, година на издаване, продължителност, държава на производство, жанрове и описание.
* **YouTube Trailer**: Втората част на модалния прозорец съдържа видео плейър с трейлър от YouTube за съответната медия.
* **Modal Actions**: В долната част на модалния прозорец се показват бутони за затваряне на прозореца и възпроизвеждане на медията.

Този компонент е добре организиран и предоставя подробна информация за медията, като същевременно осигурява възможност за взаимодействие с потребителите чрез добавяне в любими и възпроизвеждане на трейлър.

Файлът **MediaModalText.tsx** дефинира обект **MediaModalText**, който съдържа преводите на различни текстови елементи, използвани в модалния прозорец за медия. Тези текстови елементи се различават в зависимост от езика, избран от потребителя. Ето какво прави всяка част от този файл:

* **MediaModalText**: Този обект съдържа два ключа: **bg** и **en**, които представляват преводите на различните текстови елементи за български (**bg**) и английски (**en**) език. За всеки елемент, като например продължителност (**min**), държава (**country**), жанр (**genre**), бутони за затваряне (**closeBtn**) и възпроизвеждане (**playBtn**), се предоставят преводи на съответния език.
* **Props Interface Import**: Файлът импортира интерфейса **mediaModalProps** от **props.interface**, който вероятно съдържа типовете на тези текстови елементи.
* **Export**: Обектът **MediaModalText** се експортира като част от интерфейса **mediaModalProps**, за да може да бъде използван от други компоненти, които изискват тези преводи за текстови елементи на модалния прозорец за медия.

Този файл е полезен, когато искате лесно да преведете текстови елементи на различни езици във вашия приложение, като използвате единен начин за управление на преводите.

Файлът **onloadFuncs.tsx** съдържа няколко асинхронни функции, които извършват заявки към API и обработват отговорите от тях. Ето какво прави всяка функция:

* **PopularMovies**: Изпраща заявка към API за популярните филми, като подава параметри като език (**lang**) и страница (**page**). Връща обект с данни за популярните филми.
* **TrendingMovies**: Изпраща заявка към API за актуалните филми, като подава параметри като език (**lang**), времеви период (**time\_window**) и страница (**page**). Връща обект с данни за актуалните филми.
* **NowPlayingMovies**: Изпраща заявка към API за филмите, които в момента се възпроизвеждат, като подава параметри като език (**lang**) и страница (**page**). Връща обект с данни за филмите, които в момента се възпроизвеждат.
* **PopularTV**: Изпраща заявка към API за популярните сериали, като подава параметри като език (**lang**) и страница (**page**). Връща обект с данни за популярните сериали.
* **TrendingTV**: Изпраща заявка към API за актуалните сериали, като подава параметри като език (**lang**), времеви период (**time\_window**) и страница (**page**). Връща обект с данни за актуалните сериали.
* **NowPlayingTV**: Изпраща заявка към API за сериалите, които в момента се възпроизвеждат, като подава параметри като език (**lang**) и страница (**page**). Връща обект с данни за сериалите, които в момента се възпроизвеждат.
* **Load**: Изпраща заявка към API за зареждане на потребителски данни, като подава токен за достъп. Връща обект със заредените данни.
* **WatchListStatus**: Изпраща заявка към API за статуса на списъка със запазени елементи на потребителя, като подава идентификатор на медия (**mediaId**), тип на медията (**mediaType**) и токен за достъп. Връща обект с информация за статуса на списъка със запазени елементи.
* **addSiteApiKeyHeader**: Добавя заглавие за API ключ към опциите за заявка, ако API ключът е наличен в средата на изпълнение. Този ключ се използва за удостоверяване на потребителите.
* **Fetch**: Използва функцията **addSiteApiKeyHeader**, за да добави заглавие с API ключ и след това прави заявка с помощта на глобалната функция **fetch**.

Всички тези функции са групирани и експортирани за да бъдат използвани в други части на приложението за заявки към API и обработка на данните отговори.

Файлът **props.interface.ts** съдържа интерфейси и типове за различни пропъртита, които се използват в приложението. Ето какво прави всяка интерфейс/тип:

* **AuthModalProps**: Определя пропъртитата, които се използват за модални прозорци за удостоверяване на потребителите, като например дали е отворен и функция за промяна на този статус.
* **SignInTextProps**: Съдържа текстовите пропъртита, свързани с формата за влизане, като заглавие, етикети за полетата за въвеждане, текстове за описание, бутони за влизане и препратки към регистрация.
* **SignUpProps**: Аналогично на **SignInTextProps**, но за формата за регистрация.
* **SignInProps**: Определя пропъртитата, използвани в компонент за влизане, като функция за затваряне на модалния прозорец и функция за промяна на статуса на влизане.
* **NavbarTextTypes**: Съдържа текстовите пропъртита, използвани в навигационната лента на приложението, като текстове за вход/изход, настройки и език.
* **MenuItemsList**: Списък с елементите на менюто на приложението на различни езици.
* **PopularTextInterface**: Определя текстовите пропъртита за популярните медии на различни езици.
* **TrendingTextInterface**: Аналогично на **PopularTextInterface**, но за актуалните медии.
* **NowPlayingTextInterface**: Аналогично на **PopularTextInterface**, но за медиите, които в момента се възпроизвеждат.
* **mediaProps**: Съдържа пропъртита, свързани с медийните елементи като филми и сериали.
* **mediaModalProps**: Определя текстовите пропъртита, използвани в модалния прозорец за медии, като текст за минути, държава, жанр и други.
* **CardProps**: Съдържа пропъртита за компонента на карта, който се използва за представяне на медийни елементи.
* **mediaImportProps**: Определя пропъртита, свързани със зареждането на медийни елементи, като данни за филми и сериали, текстове на бутони и други.
* **movieType**: Определя пропъртита за филми, като заглавие, описание, популярност и други.
* **mediaType**: Определя пропъртита за медийни елементи като филми и сериали, като заглавие, описание, рейтинг, продължителност и други.
* **ResponseType**: Съдържа пропъртита за отговора от API заявки, като статус код, препоръчани медийни елементи, коментари и други.
* **CommentI**: Определя пропъртита за коментари, като потребителско име, съдържание, медия и други.

Тези интерфейси и типове помагат за структурирането на данните и определянето на очакваните пропъртита в различните компоненти и функции на приложението.

Във файла **Search.tsx** се дефинира компонентът **SearchComp**, който представлява търсачката в приложението. Ето какво прави компонентът:

* Използва React hooks като **useState**, **useEffect** и **useContext** за управление на състоянието и контекста на приложението.
* Използва хукът **useNavigate** от **react-router-dom** за навигация в приложението.
* Дефинира състояния за търсачката, избрания елемент и данните за резултатите от търсенето.
* Извършва асинхронно търсене, като използва функцията **QuickSearch** и показва резултатите в **SearchSuggestions**.
* Има функция за debounce на функцията **toggleLiked**, която се извиква при кликане върху бутона за харесване на медийни елементи.
* Използва **toast** от библиотеката **react-hot-toast** за показване на съобщения след успешни или неуспешни действия.
* Връща JSX, който съдържа форма за търсене, компонент за показване на резултатите и **Toaster** за показване на съобщения.

Компонентът е създаден, за да предостави потребителско интерфейсно изживяване при търсене в приложението, като осигурява възможност за интерактивност и бърза навигация.

Файлът **SearchForm.tsx** съдържа компонента **SearchForm**, който представлява формата за търсене в приложението. Ето какво прави компонента:

* Приема пропсовете **isLoading**, **isMobile**, **placeholderTxt**, **filterTxt**, **search**, **setSearch** и **setSelected**.
* Показва два елемента - поле за въвеждане на текст и бутон за изпращане на заявката за търсене.
* Полето за въвеждане на текст има следните възможности:
  + Цвят на текста - бял
  + Режим на изпълнение - **flat**
  + Форма - кръгла
  + Иконка за филтриране (ако е в мобилна версия, в противен случай показва текст)
  + При въвеждане на текст, ако дължината му е по-голяма от 3 символа, се активира опцията за избор.
  + При изчистване на полето, текстът се нулира.
* Бутона за търсене има следните възможности:
  + Цвят на текста - бял
  + Режим на изпълнение - **flat**
  + Форма - кръгла
  + Иконка за търсене, която се показва при неактивно състояние или се заменя с анимация за зареждане при активно състояние.
  + При кликане върху бутона се изпраща заявка за търсене.

Този компонент е важна част от интерфейса на потребителя, която позволява на потребителите да извършват търсене на медийни елементи в приложението.

Файлът **SearchSuggestions.tsx** съдържа компонента **SearchSuggestions**, който представлява списък със съответстващи резултати на търсенето. Ето какво прави компонентата:

* Приема пропсовете **data**, **selected**, **search**, **isLoading** и **handleClicked**.
* Визуализира списък с резултатите от търсенето, ако има такива и потребителят е избрал да види резултати.
* Показва снимка, заглавие и дата на излизане за всеки резултат от търсенето.
* Показва бутон за добавяне към любими или за премахване от тях за всеки резултат.
* При кликване върху резултата се навигира към подробно описание на медията.
* Ако потребителят не е влезнал в системата, показва бутона за отваряне на модално прозорец за влизане в системата.
* Показва бутона за виждане на всички резултати, който пренасочва към страницата за филтриране на резултатите.

Тази компонента е важна част от интерфейса на потребителя, която предоставя навигация и възможност за добавяне към любими на резултатите от търсенето.

Файлът **useMediaData.tsx** съдържа хука **useMediaData**, който се използва за извличане на данни за медията (филми или сериали) от различни източници на основата на типа на заявката (популярни, в тренд, в момента) и типа на медията (филм или сериал).

Ето какво прави хукът:

* Приема параметри **{ type, isMovie }**, където **type** е типът на заявката (популярни, в тренд, в момента), а **isMovie** указва дали заявката е за филми или за сериали.
* Използва контекста на приложението за езика по подразбиране и текущия път на маршрута с помощта на **useLocation**.
* Използва **useState** за съхранение на текущата страница, общия брой на страниците, данните за медията, както и за грешките и състоянието на зареждане.
* Използва **useEffect**, за да извика функция за извличане на данни за медията, която е подходяща за съответния тип и тип на заявката, на база на текущия език и страница. Това може да включва функции като **PopularMovies**, **TrendingMovies**, **NowPlayingMovies** за филми и съответните функции за сериали.
* Връща обект със състоянията и функциите, необходими за управление на данните за медията, като например текущата страница, грешките, състоянието на зареждане и т.н.

Този хук е полезен за управление на данни за медията в компонентите, които имат нужда от такива данни, като осигурява централизирана логика за извличане на данни и управление на състоянието.

##### “Filter” папка

Това е директорията, където се съдържат всички компоненти на използвани във Filter.

Файлът **ChipBtn.tsx** дефинира компонент **ChipBtn**, който представлява чип бутон с определени стилове и функционалности. Ето какво прави компонентът:

* Приема пропъртитата **value**, **onClick** и **is**. **value** е стойността на чипа, която се показва върху бутона. **onClick** е функция, която ще се изпълни при натискане на бутона. **is** е флаг, който указва дали чипът трябва да бъде активен или не.
* Визуализира бутона със стойността на **value** и връзка към функцията **onClick**.
* Визуализира икона **Check**, ако **is** е true, което показва, че чипът е активен.
* Управлява цвета на бутона в зависимост от стойността на **is**. Ако **is** е true, бутона ще бъде с цвят на акцент (примерно, синьо), в противен случай ще бъде с цвят по подразбиране.
* Използва различни варианти на стилове за бутона в зависимост от стойността на **is**. Ако **is** е true, бутона ще бъде с издигащ сянка, в противен случай ще бъде със затъмнена прозрачност.
* Прилага клас "cursor-pointer", който променя курсора, когато се ховърва над бутона, показвайки, че той е интерактивен.

Този компонент е полезен за създаване на чипове с различни стилове и функционалности, които могат да бъдат използвани в различни части на приложението.

**FilterCategory.tsx** представлява компонент, който показва филтри за категории на съдържанието. Ето основните функции на компонента:

* Приема контекста на приложението чрез **useContext** от **AppContext**, за да използва езика по подразбиране и информация за мобилното устройство.
* Дефинира локални състояния за различни параметри на филтъра като тип на съдържанието, начин на сортиране, ключова дума за търсене, жанрове, държави и период от време.
* Показва селектори за избор на тип на съдържанието, начин на сортиране и период от време.
* Използва компонента **DateRangePicker** за избор на времеви период.
* Показва жанровете и държавите в акордеони, като предоставя възможност за избор и филтриране.
* Предоставя бутон "Филтър", който при натискане генерира URL адрес, в който се включват всички избрани филтри. После пренасочва потребителя към този URL адрес.

Този компонент предоставя гъвкав и интерактивен начин за филтриране на съдържание в приложението, като дава възможност за избор на различни параметри за филтриране.

Top of Form

**Genres.tsx** е компонент, който показва жанровете като чипове и предоставя възможност за избор на жанрове чрез тях. Ето основните функции на компонента:

* Приема пропс параметри, включително функция **handleChipClick** за обработка на щраквания върху чиповете, масив от избрани жанрове **selGenres** и език по подразбиране **defaultLanguage**.
* Използва масива от жанрове **selGenres**, за да генерира чип за всеки жанр.
* За всеки чип се използва компонента **ChipBtn**, която показва името на жанра и предоставя възможност за избор чрез кликване.

Този компонент е част от системата за филтриране на съдържанието и предоставя интуитивен начин за избор на жанрове от потребителите.

**MediaList.tsx** е компонент, който показва списък с медийни елементи, като филми, сериали и други. Ето основните функции на компонента:

* Приема пропс параметри като заглавие (**title**), данни (**data**), функция за отваряне на медия (**onOpen**), функция за задаване на идентификатор на медия (**setMediaId**) и функция за определяне на типа на медията (**setMovieMedia**).
* Генерира списък от карти (**CardComponent**) за всеки елемент от данните (**data**).
* Компонентът предоставя възможност за кликване върху всяка карта, което активира функцията **onOpen**, като предава идентификатора на медията.
* Компонентът може да бъде използван за показване на различни категории от медийни елементи, като например популярни филми, нови сериали и други.
* Поддържа респонсивен дизайн за адаптивност към различни размери на екрана.

Компонентът също така включва **AdsComponent**, който се показва в края на списъка с медийни елементи. Това може да бъде използвано за включване на рекламни съобщения или други видове съдържание между медийните елементи.

##### “HomePage” папка

Това е директорията, където се съдържат всички компоненти на използвани във HomePage.

**AdsComponent.tsx** е компонент, който генерира и показва рекламни карти.

Ето основните характеристики на този компонент:

* Приема няколко празни реда от елементи във формат на карти (**Card**), като всяка карта съдържа тяло (**CardBody**) и подвал (**CardFooter**).
* Генерира карти със съдържание, съответстващо на рекламно съобщение, което се показва в тялото на картата и в подвала.
* Използва цикли за създаване на определен брой карти въз основа на дължината на списъка от елементи, което се използва за визуализиране на рекламата.
* Компонентът е визуализиран като вертикална колона със статична височина и разстояние между елементите.
* Всяка карта може да се натисне, тъй като има **isPressable** пропс.

Този компонент може да се включи в други компоненти като част от макета на уебсайта, за да се покажат рекламни съобщения или други видове съдържание между основните елементи.

**Header.tsx** е компонент, който представлява заглавната част на уеб страницата. Ето основните характеристики на този компонент:

* Приема и използва контекст от **AppContext** и **HomePageContext**, за да получи информация за езика по подразбиране (**defaultLanguage**), състоянието на отвореност на модалното прозорец (**isOpen**), идентификатора на медията (**mediaId**) и други състояния и методи.
* Показва заглавен текст и няколко параграфа със съдържание, което може да варира в зависимост от избрания език.
* Съдържа компонент за търсене (**Search**), който позволява на потребителя да търси медийно съдържание.
* Използва компонент за модално прозорец (**MediaModal**), който се използва за показване на допълнителна информация за медията при кликване върху нея.
* Обработва състоянието на грешка и, ако има грешка, не показва нищо.

Този компонент представлява основната част от заглавната секция на уеб страницата и включва визуализация на текст и функционалност за търсене на медийно съдържание.

**MediaList.tsx** е компонент, който отговаря за показването на списък от медийно съдържание, като филми или сериали. Ето някои ключови точки за този компонент:

* Приема пропсове като **title**, **buttonText1**, **buttonText2**, **moviesData**, **tvData**, **handleTVChange**, **isMovies**, **setIsMovies**, **type**, които са необходими за изграждането на списъка от медийно съдържание и за превключване между филми и сериали.
* Използва контекст от **HomePageContext** и **AppContext**, за да получи необходимите данни и настройки, като състоянието на медията (**isMovies**), функцията за промяна на медията (**handleTVChange**), езика по подразбиране и други.
* Рендерира заглавие, бутони за превключване между филми и сериали, и бутон за виж повече, който пренасочва потребителя към страницата с всички филми или сериали.
* Използва **CardComponent** за визуализиране на медийното съдържание, като филмите или сериалите, като пренасочва потребителя към подробности при кликване върху картичката.
* Включва рекламен компонент (**AdsComponent**), който се показва в края на списъка от медийно съдържание.

Този компонент предоставя динамично и интерактивно преживяване за потребителите, като им позволява да превключват между филми и сериали и да разглеждат списъка с медийно съдържание със задоволително изживяване.

Top of Form

**NowPlaying.tsx, Popular.tsx, Trending.tsx** са компонени със едни и същи характеристики, които отговарят за показването на списък с медийно съдържание. Ето някои ключови точки за този компонент:

1. Компонент използва контекст от **AppContext** и **HomePageContext**, за да получи необходимите данни и настройки, като текущия език по подразбиране (**defaultLanguage**) и медийното съдържание, което в момента се излъчва (**nowPlayingMovies**).
2. Използва функцията **TVChange**, която се грижи за промяната между филми и сериали в списъка с медийно съдържание, когато потребителят превключва типа медия.
3. Рендерира списък от медийно съдържание чрез компонента **MediaList**. Този списък включва заглавието "Now Playing", бутони за превключване между филми и сериали, и съответния списък с медийно съдържание, което в момента се излъчва.
4. Показва зареждащ компонент (**Loader**), ако има забавяне при зареждането на данните или ако списъкът с медийно съдържание е празен.

Този компонент предоставя потребителско преживяване, което позволява на потребителите да разглеждат медийно съдържание, което в момента се излъчва, и да превключват между филми и сериали с лекота.

##### “Movie” папка

Това е директорията, където се съдържат всички компоненти на използвани във Movie.

В **Comment.tsx Comments** и **Comment** са компоненти, които се използват за показване на коментари и възможност за добавяне на нови коментари. Ето основните технологии в тези компоненти:

* **Comments**:
  + Показва списък от коментари, включително възможност за добавяне на нов коментар.
  + Приема списък с коментари (**comments**), уникално идентификатор на медията (**mediaId**) и тип на медията (**mediaType**).
  + Използва контекст от **AppContext** за достъп до информация като потребителско име, език и тема.
  + Рендерира всеки коментар чрез компонента **Comment**.
  + Включва форма за добавяне на нов коментар.
* **Comment**:
  + Показва отделен коментар и възможност за добавяне на отговори към него.
  + Приема конкретен коментар (**comment**), уникално идентификатор на медията (**mediaId**) и тип на медията (**mediaType**).
  + Използва контекст от **AppContext** за достъп до информация като потребителско име, език и тема.
  + Рендерира информация за потребителя, съдържанието на коментара и възможност за добавяне на отговори.
  + Включва логика за изтриване на коментар, показване на допълнителни коментари (ако има) и добавяне на нови отговори.

Тези компоненти са част от функционалността за коментари на приложението, която позволява на потребителите да общуват и споделят мнения за съдържанието. Взаимодействието с базата данни и съобщенията към потребителите се извършва чрез извикване на различни функции за обработка на коментарите, дефинирани във **handleFunctions**.

**ErrorComponent.tsx** е прост компонент за показване на грешки, който се използва, когато възникне грешка или проблем в приложението. Ето основните характеристики на този компонент:

* Приема пропс **errorText**, **smallText** и **backBtn**, които съдържат съответно основния текст за грешката, пояснителен малък текст и текст за бутон за връщане.
* Използва компонента **Button** от библиотеката **@nextui-org/react** за създаване на бутон за връщане към началната страница.
* Използва **useNavigate** хук от **react-router-dom** за навигация при натискане на бутона за връщане.
* Рендерира текстове за грешката в голям шрифт (8xl) и пояснителен малък текст в по-малък шрифт (2xl).
* Компонентът е центриран вертикално и хоризонтално в секция с височина от 92vh, за да се осигури видимост дори при големи екрани.
* Бутонът за връщане е стилизиран с цвят, радиус и ефект на сянка, за да привлече вниманието на потребителя и да го насочи към възможността за връщане към началната страница.

Този компонент е полезен за управление на сценарии, в които потребителят се нуждае от информация за възникнала грешка и възможност за връщане назад към началната точка на приложението.

**FirstSection.tsx (идентична логика като и в папката TV)** е част от уеб страницата за гледане на филми онлайн и представлява първата секция от страницата, която включва видео плейър и бутони за превключване на различни видове сървъри за видеото. Ето някои характеристики на този компонент:

* Приема пропс **movieId**, **server**, **possibleServers** и **handleServerChange**. **movieId** е идентификаторът на филма, който се възпроизвежда, **server** е текущият използван сървър за видеото, **possibleServers** са наличните сървъри за видеото, а **handleServerChange** е функция за промяна на избрания сървър.
* В компонента се използва **useState** за съхранение на линка към видеото, което ще се покаже.
* Използва се **useContext** за достъп до контекста на приложението, за да се вземат настройките като език и информация за мобилност.
* **useEffect** хук се използва, за да се актуализира линка към видеото при промяна на сървъра или идентификатора на филма.
* Компонентът се състои от две части - видео плейър и панел с бутони за превключване на сървъри.
* Видео плейърът е представен чрез **ResponsiveEmbed** компонента от библиотеката **react-responsive-embed**, която позволява вграждането на реактивни видео плейъри с поддръжка на различни опции като пълноекранен режим и други.
* Панелът с бутони за сървъри позволява на потребителя да превключва между различните сървъри, а при натискане на бутона се извиква функцията **handleServerChange**, която променя текущия сървър.
* Бутоните за сървъри се представят чрез компонента **Button** от библиотеката **@nextui-org/react**, а при навлизане с мишката върху тях се показва подсказка с името на сървъра.
* Текстовете, които се показват в заглавието и информационния текст, се вземат от обекта **FirstSectionText** в зависимост от текущия език на приложението.

Този компонент предоставя интерактивно преживяване за потребителите, като им дава възможност да избират от различни сървъри за гледане на видеото и предоставя информация за сървърите, ако текущият сървър не работи.

**Recommended.tsx** представлява секция от уеб страницата, където се показват препоръчани медийни елементи като филми или телевизионни предавания. Ето основните характеристики на този компонент:

* Приема пропс **recommended**, който е масив от медийни елементи, които трябва да бъдат показани в секцията с препоръчани.
* Използва **useState** за съхранение на текущото състояние на модалното прозорец и за идентификация дали се избира медия от тип филм или телевизионно предаване.
* Използва **useDisclosure** хук от библиотеката **@nextui-org/react** за управление на видимостта на модалния прозорец.
* Компонентът се състои от две подкомпоненти: **MediaList** и **MediaModal**.
* **MediaList** компонентът се използва за показване на списък с препоръчани медийни елементи. Той приема данните за препоръчаните медии, заглавие и функции за отваряне на модалния прозорец и избор на медия. При избор на медия, се активира функцията **onOpen**, която отваря модалния прозорец.
* **MediaModal** компонентът се използва за показване на детайлна информация за избраната медия. Той приема текущото състояние на избора на медия (филм или телевизионно предаване), идентификатор на медията, статус на видимостта на модалния прозорец и функция за промяна на статуса на видимостта на модалния прозорец.
* Когато потребителят избере медия от списъка с препоръчани, модалният прозорец се отваря и показва подробности за избраната медия.

Този компонент предоставя лесен начин за преглед на препоръчани медийни елементи и дава възможност за преглед на подробности за избраните медии.

**SecondSection.tsx (идентична логика като и в папката TV)**  компонентът представлява секция на уеб страницата, която показва подробна информация за определен медиен елемент, например филм или телевизионно предаване. Ето основните характеристики на този компонент:

* Приема различни пропсове, които са необходими за показване на информацията за медията, като заглавие, оценка, рейтинг, дължина, жанрове, страна на производство и други.
* Използва различни хука като **useState**, за да управлява състоянието на компонента.
* Визуализира подробна информация за медията, включително заглавие, оценка, рейтинг, дължина, описание и други характеристики.
* Позволява на потребителя да добавя медията в своите харесани или да я оцени, като използва бутоните за това.
* Използва различни компоненти като **Button**, **Image**, **Tooltip**, **Divider** и други, за да стилизира и представи информацията по подходящ начин.

Този компонент е създаден, за да предостави подробна и информативна прегледна страница за медийните елементи и дава възможност на потребителите да взаимодействат с тях, като ги оценяват или добавят към своите харесани.

##### “Nav” папка

Това е директорията, където се съдържат всички компоненти на използвани във NavigationBar.

**AuthModal.tsx** представлява модален прозорец, който се използва за откриване на форми за вход или регистрация на потребителите. Ето основните характеристики на този компонент:

* Приема пропсове за управление на видимостта на модалния прозорец (**isOpen**) и функция за промяна на тази видимост (**onOpenChange**).
* Използва вътрешно състояние чрез хук **useState**, за да проследи дали потребителят е в процес на влизане или регистриране (**isLogin**).
* Показва модален прозорец с помощта на компонента **Modal** от NextUI, като настройва различни параметри като фонов бекграунд, размер, преходни ефекти и други.
* Използва компонентите **SignIn** и **SignUp** за представяне на формите за вход и регистрация, съответно.
* Позволява на потребителите да превключват между формите за вход и регистрация, като използва бутона за смяна на режима.

Този компонент предоставя удобен начин за вход и регистрация на потребителите чрез модален прозорец със стилизиран и интерактивен интерфейс.

**LoggedInView.tsx** представлява панел за навигация, който се показва, когато потребителят е влязъл в системата. Ето някои ключови характеристики на този компонент:

* Приема пропсове, които съдържат информация за потребителя, като потребителско име, имейл, тема и език, както и функции за промяна на темата и езика, както и за управление на сесията на потребителя.
* Използва контекст от приложението, за да получи информация за цветовата тема, достъпа до системата и точките на потребителя.
* Показва панел за потребителя с различни опции като профил, настройки, харесани, гледани, промяна на темата и езика, както и излизане от профила.
* Показва прогрес бар за точките на потребителя и предоставя възможност за преглед на точките чрез бутона.
* Позволява на потребителя да променя темата и езика на приложението чрез падащи менюта.
* Предоставя бутон за излизане от сесията на потребителя, който премахва маркера за достъп и изчиства бисквитките.

Този компонент предоставя удобен начин за навигация и управление на профила на потребителя, като същевременно осигурява възможности за персонализация и настройка на приложението.

**LoggedOutView.tsx** представлява панел за навигация, който се показва, когато потребителят не е влязъл в системата. Ето някои ключови характеристики на този компонент:

* Приема пропсове, които съдържат информация за темата, езика и текстовете на панела, както и функции за промяна на темата и езика.
* Показва панел за навигация с опции за избор на тема и език, както и вход в системата.
* Позволява на потребителя да променя темата и езика на приложението чрез падащи менюта.
* Предоставя опция за вход в системата, като извиква функцията **onOpen**, която отваря модален прозорец за вход в системата.

Този компонент осигурява начин за потребителите да променят настройките на приложението и да влязат в системата, като същевременно предоставя възможности за персонализация и управление на профила на потребителя.

Файлът **NavigationBar.tsx** съдържа компонент, който представлява горния панел на навигация във вашия уеб сайт. Ето основните характеристики на този компонент:

* **Структура на навигационния панел**:
  + Панелът се състои от лого, меню с категории, бутон за вход/профил и възможност за смяна на езика и темата на приложението.
  + В зависимост от състоянието на потребителя (влязъл ли е в системата или не), се показва различно съдържание в профилния бутон.
* **Лого и бутон за вход/профил**:
  + Логото води до началната страница на сайта.
  + Профилният бутон включва аватар, който показва потребителския профил или икона за вход, ако потребителят не е в системата.
  + При натискане на аватара се отваря падащо меню с опции за вход в системата или управление на профила.
* **Меню с категории**:
  + Показва списък с категории като "Популярни", "Тенденции" и "В момента по кината", като всяка категория има свое подменю.
  + Подменютата позволяват на потребителите да избират между различни видове медийно съдържание като филми и сериали.
* **Възможност за промяна на темата и езика**
  + Потребителите могат да променят темата на уеб сайта (светла, тъмна или системна) и езика на интерфейса.
  + Промените се отразяват в реално време и се запазват в локалното съхранение на браузъра.

Този компонент предоставя удобен начин за навигация и персонализация на потребителския опит във вашия уеб сайт.

Файлът **SignIn.tsx** представлява компонент за вход в системата, който се използва за аутентикация на потребителите във вашето уеб приложение. Ето основните характеристики и функционалности на този компонент:

* **Интерфейс за вход**:
  + Потребителите могат да въведат потребителско име и парола в отделните полета за вход.
  + Полетата се валидират за валидност на въведените данни и показват съобщения за грешка при невалидни данни.
* **Визуализация на грешки и успех**:
  + Показват се съобщения за грешка при невалидни данни за вход или грешки, възникнали при сървъра.
  + При успешен вход се показва прогрес бар и съобщение за успех, които потвърждават успешното аутентикиране на потребителя.
* **Функционалност за вход и обработка на данни**:
  + При натискане на бутона за вход се изпращат въведените потребителско име и парола към сървъра за валидация.
  + След получаване на отговор от сървъра се обработват данните и се вземат необходимите действия в зависимост от резултата от входа.
* **Променлив език и текстове**:
  + Интерфейсът за вход поддържа различни езици, като текстовете се превеждат според настройките за езика на потребителя.
  + Текстовете за вход се извличат от външен източник (например, файла **SignInText**), което улеснява превода и локализацията на приложението.
* **Интерактивност и визуализация**:
  + Компонентът предлага възможности за интерактивност, като скриване/показване на паролата и визуализация на прогреса при влизане в системата.

Този компонент е важна част от аутентикационния процес във вашия уеб сайт и предоставя лесен и удобен начин за влизане в системата за потребителите.

Файлът **SignUp.tsx** съдържа компонента за регистрация, който позволява на потребителите да се регистрират във вашия уеб сайт. Ето основните характеристики и функционалности на този компонент:

* **Интерфейс за регистрация**:
  + Потребителите могат да въведат потребителско име, имейл адрес, парола и потвърдена парола в отделните полета за регистрация.
  + Полетата се валидират за валидност на въведените данни и показват съобщения за грешка при невалидни данни.
* **Визуализация на грешки и успех**:
  + Показват се съобщения за грешка при невалидни данни за регистрация или грешки, възникнали при сървъра.
  + При успешна регистрация се показва прогрес бар и съобщение за успех, които потвърждават успешната регистрация на потребителя.
* **Функционалност за регистрация и обработка на данни**:
  + При натискане на бутона за регистрация се изпращат въведените данни към сървъра за валидация и регистрация.
  + След получаване на отговор от сървъра се обработват данните и се вземат необходимите действия в зависимост от резултата от регистрацията.
* **Променлив език и текстове**:
  + Интерфейсът за регистрация поддържа различни езици, като текстовете се превеждат според настройките за езика на потребителя.
  + Текстовете за регистрация се извличат от външен източник (например, файла **SignUpText**), което улеснява превода и локализацията на приложението.
* **Интерактивност и визуализация**:
  + Компонентът предлага възможности за интерактивност, като скриване/показване на паролата и визуализация на прогреса при регистрация.

Този компонент е важна част от регистрационния процес във вашия уеб сайт и предоставя лесен и удобен начин за регистрация на потребителите.

##### “User” папка

Това е директорията, където се съдържат всички компоненти на използвани във User.

Файлът **Admin.tsx** съдържа компонента за управление на администратори. Ето някои от ключовите характеристики и функционалности на този компонент:

* **Извличане на данни и визуализация**:
  + Компонентът извлича данни за администраторите от сървъра, използвайки функцията **FindAll**.
  + Получените данни се визуализират в таблица с помощта на компонентите от **@nextui-org/react**.
* **Управление на администраторски права**:
  + За супер администраторите се показва допълнителен интерфейс за управление на администраторите.
  + Супер администраторите имат възможност да редактират ролята на другите администратори чрез бутони за редакция.
  + След промяната на ролята се извежда съобщение за успешно завършена операция.
* **Изтриване на администраторски профили**:
  + Администраторите могат да изтриват администраторски профили чрез бутоните за изтриване.
  + При успешно изтриване се извежда съобщение за успех.
* **Визуализация на съобщения**:
  + При успешни или неуспешни операции се показват съобщения с помощта на библиотеката **react-hot-toast**.
* **Локализация и настройки на темата**:
  + Компонентът поддържа различни езици и визуализации в зависимост от избрания език.
  + Темата на компонента се настройва в съответствие с избраната тема или системната тема на потребителя.

Този компонент предоставя мощен интерфейс за управление на администраторските права във вашия уеб сайт и прави операциите с администраторите лесни и удобни.

Top of Form

Файлът **Liked.tsx** съдържа компонент, който показва списък със запазени медийни елементи (например, филми или сериали), които потребителят е отбелязал като харесани. Ето някои от ключовите характеристики и функционалности на този компонент:

* **Извличане на харесани медийни елементи**:
  + Компонентът извлича списък с харесани медийни елементи от сървъра чрез функцията **GetLiked**.
  + Получените данни се визуализират в списък с картички за всеки елемент.
* **Изтриване на харесани елементи**:
  + Потребителят може да изтрие всички харесани елементи едновременно чрез бутона "Delete All".
  + Преди да бъдат изтрити всички елементи, се появява потвърждаващ диалог.
* **Динамично зареждане на медийни елементи**:
  + Ако няма харесани елементи, се извежда съобщение, че няма резултати.
  + При наличие на харесани елементи, те се визуализират като картички с информация за всеки елемент.
* **Лениво зареждане на медийни модали**:
  + Когато потребителят щракне върху карта на медийния елемент, се отваря модално прозорец с подробности за този елемент.
  + За забавяне на зареждането на модалното прозореце се използва **lazy** функцията за лениво зареждане на компонента **MediaModal**.
* **Визуализация на съобщения за потвърждение**:
  + Преди изтриването на всички харесани елементи се показва потвърждаващ диалог с информация за операцията.

Този компонент предоставя потребителски интерфейс за преглед и управление на харесаните медийни елементи и прави операциите с тях лесни и удобни за потребителя.

Файлът **Profile.tsx** представлява компонент, който показва профилната страница на потребителя. Ето някои от ключовите характеристики и функционалности на този компонент:

* **Профилни данни и валидация**:
  + Потребителите могат да редактират потребителското име, имейл и парола.
  + Всички полета подлежат на валидация, която включва проверка за валиден формат на имейл и минимална дължина на паролата.
* **Визуализация на грешки**:
  + При невалидни въведени данни се показват подходящи съобщения за грешка.
* **Смяна на парола**:
  + Потребителите имат възможност да променят паролата си, като въвеждат нова парола и я потвърждават.
* **Потвърждаване на имейл**:
  + Потребителите, чиито профили не са потвърдени, виждат съобщение за потвърждаване на имейл.
  + Предоставя се връзка за потвърждаване, която изисква повторно изпращане на имейл за потвърждение.
* **Обработка на събития**:
  + Потребителите могат да изпратят заявка за потвърждаване на имейл.
  + Потребителите могат да запазят направените промени в профила си.

Този компонент предоставя възможност за лесно управление на профилните данни на потребителя и осигурява сигурността и удобството при работата с акаунта.

Файлът **Settings.tsx** съдържа компонент, който представлява страницата за настройки на потребителския профил. Ето някои от ключовите функционалности на този компонент:

* **Превключватели за настройки**:
  + Потребителите могат да променят настройките на профила си, като например активиране на таен профил или включване на история на гледане.
* **Изтриване на профил**:
  + Потребителите имат възможност да изтрият своя профил.
  + Преди потвърждаване на изтриването се показва подсказка със съобщение за потвърждение.
* **Потвърждение на профил**:
  + При потребители, чиито профили не са потвърдени, се показва съобщение за потвърждаване на профила.
* **Визуализация на съобщения**:
  + Показват се съобщения за успешни операции, като активиране на таен профил или изтриване на профил.
* **Извличане и обработка на данни**:
  + Компонентът извлича информация от контекста на приложението, като език по подразбиране, потвърждение на профила, настройки за таен профил и други.

Този компонент предоставя възможност за персонализация на потребителския профил и управление на настройките на профила.

Файлът **Tabs.tsx** съдържа компонент, който представлява табове за навигация на потребителския интерфейс. Ето някои от ключовите функционалности на този компонент:

* **Навигационни табове**:
  + Потребителите могат да избират между различни табове за навигация, като например "Харесвани", "Гледани", "Профил", "Настройки" и "Админ".
* **Икони и текст**:
  + Всеки таб включва икона и текстов етикет, които показват на потребителя къде ще бъде пренасочен след щракване върху таба.
* **Активен таб**:
  + Активният таб се подчертава, за да бъде ясно за потребителя коя страница е отворена в момента.
* **Динамична промяна на табовете**:
  + Табовете могат да се променят динамично, в зависимост от действията на потребителя или от контекста на приложението.
* **Управление на навигацията**:
  + Потребителите могат да навигират към различни страници, като кликнат върху съответния таб.
* **Контрол на достъпа**:
  + Някои табове могат да се показват или скриват в зависимост от ролята на потребителя, като например администраторският таб, който се показва само за администратори.

Този компонент предоставя удобен начин за навигация в приложението и дава възможност на потребителите да достъпват различни функции и страници на потребителския интерфейс.

Файлът **Verify.tsx** съдържа компонент, който се използва за потвърждаване на профил в приложението. Ето някои от ключовите му функционалности:

* **Потвърждаване на профил**:
  + Когато компонентът се зареди, той изпраща заявка за потвърждаване на профила, използвайки предоставения токен от параметрите на маршрута.
* **Използване на контекст**:
  + Компонентът използва контекста на приложението, за да достъпи езика по подразбиране и функционалността за задаване на маркер за достъп.
* **Динамично показване на съобщения**:
  + В зависимост от отговора от заявката за потвърждаване на профила, компонентът динамично показва съобщения за успешно потвърждение или грешка.
* **Навигация към началната страница**:
  + Потребителят може да се върне към началната страница на приложението, като натисне бутона "Back".
* **Използване на параметри от маршрута**:
  + Компонентът използва параметъра **token** от маршрута, за да извлече токена за потвърждаване на профила.

Този компонент осигурява удобен начин за потвърждаване на потребителски профили в приложението и дава възможност на потребителите да активират своите профили след успешна верификация.

Файлът **Watched.tsx** съдържа компонент, който представя историята на гледане на потребителя в приложението. Ето някои от ключовите му характеристики:

* **Извличане на данни за гледане**:
  + Компонентът използва функцията **GetWatched** за извличане на данни за гледането на потребителя от базата данни.
* **Изтриване на елементи от историята на гледане**:
  + Потребителите могат да изтриват единични елементи от историята на гледане, както и цялата история, чрез бутоните "Delete" и "Delete All".
* **Динамично показване на съдържание**:
  + Ако има записани елементи в историята на гледане, те се показват в вид на картички с информация за медията.
  + Ако няма записи в историята на гледане, се показва съобщение, което информира потребителя за това.
* **Използване на контекст и езикови настройки**:
  + Компонентът използва контекста на приложението, за да достъпи езиковите настройки и информацията за темата на приложението.
* **Динамично зареждане на медийни данни**:
  + Медийните данни за избран елемент се зареждат динамично, като се използва модално прозорец при отварянето на картичката на медията.

Този компонент предоставя удобен начин за преглед на историята на гледане на потребителите и им дава възможност да управляват и изтриват своите гледани медийни елементи.

#### “pages” папка

Това е директорията, където се съдържат всички Страници.

Файлът **User.tsx** представлява основния компонент за рутинг и навигация в потребителската част на приложението. Ето основните му характеристики:

* **Проверка на достъпа до потребителския профил**:
  + Компонентът използва **accessToken**, за да определи дали потребителят има достъп до потребителския профил.
  + Ако няма валиден **accessToken**, потребителят се пренасочва към началната страница на приложението.
* **Управление на пътища и рутиране**:
  + Компонентът се грижи за управлението на рутите за различните части на потребителския профил, като профил, настройки, администраторски панел и други.
  + В случай на грешка при навигацията, се показва страница за грешка с код 404.
* **Търсене на медийни елементи**:
  + Компонентът включва търсачка, която позволява на потребителя да търси медийни елементи.
* **Използване на контекст и езикови настройки**:
  + Използват се контекстът на приложението и езиковите настройки за правилното показване на съдържанието на потребителския профил.
* **Поддръжка на мултиезичност**:
  + Потребителският профил е поддържан на няколко езика, като се използва езиковата конфигурация, зададена от потребителя.
* **Динамично зареждане на съдържание**:
  + Съдържанието на потребителския профил се зарежда динамично чрез използване на компонента **Suspense**.
* **Интернационализация и локализация**:
  + Текстовете се показват на различни езици в зависимост от предпочитанията на потребителя.

Този компонент осигурява централизирана точка за управление на навигацията и предоставя лесен и интуитивен начин за достъп до различните части на потребителския профил в приложението.

Файлът **HomePage.tsx** съдържа компонент, който представлява началната страница на приложението. Ето основните характеристики:

* **Хедър с лого и навигационни елементи**:
  + Включва компонент за хедър (**Header**), който съдържа лого на приложението и навигационни елементи.
* **Раздел "Популярни"**:
  + Визуализира популярните медийни елементи, които потребителите гледат най-много.
* **Раздел "В момента"**:
  + Показва медийни елементи, които се излъчват в момента или са на върха на популярността.
* **Раздел "Тенденции"**:
  + Показва медийни елементи, които са в тренда в момента.
* **Контекст за началната страница**:
  + Използва контекст (**HomePageProvider**), който предоставя информация и функционалност, свързани с началната страница.
* **Динамично променящ се заглавен текст**:
  + Заглавието на страницата се променя в зависимост от избрания език, като се прилагат подходящи преводи.
* **Поддръжка на теми и стилове**:
  + Фоновите и цветовите теми на страницата се променят в зависимост от избраната тема на приложението.

Този компонент е отговорен за визуализирането на основните секции и функционалности на началната страница на приложението, като осигурява добро потребителско изживяване и лесна навигация.

Файловете **: Movie/Popular.tsx**, **Movie/NowPlaying.tsx, Movie/Trending.tsx TV/Popular.tsx, TV/NowPlaying.tsx, TV/Trending.tsx** имат една и съща логика, затова ще се опише само един.

Файлът**Trending.tsx** съдържа компонент, който показва списък с популярни филми. Ето основните характеристики на този компонент:

1. **Получаване на контекст от приложението**:
   * Използва хука **useContext** за получаване на контекста на приложението, който съдържа информация за текущия език, тема и др.
2. **Навигация и търсене**:
   * Използва хук **useNavigate** от **react-router-dom** за навигация в приложението.
   * Включва компонент за търсене (**SearchComp**), който позволява потребителят да търси сред популярните филми.
3. **Показване на списък с медийни елементи**:
   * Използва хук **useMediaData**, за да зареди данните за популярните филми.
   * Визуализира списък с медийни елементи, които са популярни филми.
   * Включва **MediaList** компонент, който показва списъка с медийни елементи.
4. **Пагинация**:
   * Включва пагинационен контрол, който позволява на потребителя да преминава между различните страници на списъка с популярни филми.
   * Използва компонента **Pagination** от **@nextui-org/react**.
5. **Модално прозорец за медия**:
   * Включва модално прозорец за показване на подробности за определен филм при натискане на съответния елемент от списъка.
6. **Поддръжка на различни езици**:
   * Поддържа различни езици чрез включване на подходящи преводи на текстовите съобщения.
7. **Обработка на грешки**:
   * При наличие на грешка показва подходящо съобщение за грешка.

Този компонент е отговорен за визуализирането на списък с популярни филми и предоставянето на функционалност за търсене, навигация и взаимодействие с медийните елементи.

Файлът **Movie.tsx** съдържа компонент, който показва подробности за определен филм. Ето основните характеристики на този компонент:

* **Получаване на параметри от URL-а**:
  + Използва хука **useParams** от **react-router-dom** за получаване на параметъра **movieId** от URL-а.
* **Зареждане на данни за филма**:
  + Използва функцията **MediaDetails** от **handleFunctions** за извличане на данни за филма от сървъра.
  + Извлича и запазва информация за филма, препоръчани филми и коментари към филма.
  + Показва зареждащ индикатор докато данните се зареждат.
* **Поддръжка на различни езици**:
  + Показва текстови съобщения на подходящия език в зависимост от текущия език на приложението.
* **Показване на секции за информация за филма**:
  + Включва първа секция, която показва основна информация за филма, като заглавие, жанр, оценка и т.н.
  + Включва втора секция, която показва по-подробна информация за филма, като описание, рейтинг, режисьори и актьори.
* **Избор на сървър за гледане на филма**:
  + Позволява на потребителя да избере сървър за гледане на филма.
* **Показване на препоръчани филми и коментари**:
  + Показва списък с препоръчани филми, които са подобни на текущия.
  + Показва коментари към филма, ако има такива.

Този компонент предоставя подробности за определен филм, включително основна информация, по-подробна информация, препоръки и коментари.

Файлът **TVShows.tsx** съдържа компонент, който показва подробности за определен телевизионен сериал. Ето основните характеристики на този компонент:

* **Получаване на параметри от URL-а**:
  + Използва хука **useParams** от **react-router-dom** за получаване на параметрите **tvId**, **season** и **episode** от URL-а.
* **Зареждане на данни за телевизионния сериал**:
  + Използва функцията **MediaDetails** от **handleFunctions** за извличане на данни за телевизионния сериал от сървъра.
  + Извлича и запазва информация за сериала, препоръчани сериали и коментари към сериала.
  + Показва зареждащ индикатор докато данните се зареждат.
* **Поддръжка на различни езици**:
  + Показва текстови съобщения на подходящия език в зависимост от текущия език на приложението.
* **Показване на секции за информация за сериала**:
  + Включва първа секция, която показва основна информация за сериала, като заглавие, жанр, оценка и т.н.
  + Включва втора секция, която показва по-подробна информация за сериала, като описание, рейтинг, сезони и епизоди.
* **Избор на сървър за гледане на сериала**:
  + Позволява на потребителя да избере сървър за гледане на сериала.
* **Показване на препоръчани сериали и коментари**:
  + Показва списък с препоръчани сериали, които са подобни на текущия.
  + Показва коментари към сериала, ако има такива.

Този компонент предоставя подробности за определен телевизионен сериал, включително основна информация, по-подробна информация, препоръки и коментари.

Top of Form

Файлът **Filter.tsx** представлява компонент, който се използва за филтриране на медийни съдържания като филми и сериали. Ето основните характеристики на този компонент:

* **Използване на параметри от URL-а**:
  + Използва **URLSearchParams** за извличане на параметрите от текущия URL, като ключови думи за търсене, жанрове, тип медия (филм или сериал), година на издаване и други.
* **Извличане на медийни данни от сървъра**:
  + Използва функцията **Discover** от **handleFunctions** за извличане на медийни данни въз основа на филтърните критерии.
  + Показва списък с медийни елементи, като включва информация за заглавие, снимка и други релевантни детайли.
* **Използване на модално прозорец за показване на подробности за медията**:
  + Използва компонента **MediaModal** за показване на подробности за медията в модално прозорец при кликване върху определен елемент от списъка с медийни данни.
* **Поддръжка на различни езици и теми**:
  + Използва текущия език и тема от контекста на приложението за подходящо показване на текстови съобщения и стилове на компонентите.
* **Показване на зареждащ индикатор**:
  + Показва зареждащ индикатор, докато се извличат медийни данни от сървъра.

Компонентът **Filter** е основен инструмент за търсене и филтриране на медийно съдържание в приложението, който осигурява удобен начин за навигация и избор на желаното съдържание за гледане.

Файлът **Acc.tsx** представлява компонент, който се използва за показване на информация за потребителски акаунт в приложението. Ето основните характеристики на този компонент:

* **Използване на параметри от URL-а**:
  + Използва хука **useParams** от **react-router-dom** за извличане на параметъра **username** от текущия URL, който определя потребителския акаунт, за който ще се показва информацията.
* **Извличане на информация за акаунта от сървъра**:
  + Използва функцията **AccountFinder** от **handleFunctions** за извличане на информация за потребителския акаунт въз основа на предоставеното потребителско име (**username**).
  + Показва информация като име на потребителя, имейл адрес, роля, точки и др.
* **Поддръжка на различни езици и теми**:
  + Използва текущия език и тема от контекста на приложението за подходящо показване на текстови съобщения и стилове на компонентите.
* **Използване на бутони и прогрес бар**:
  + Използва компоненти като **Button** и **Progress** за показване на различни елементи на потребителския интерфейс, като бутон за навигация към началната страница и прогрес бар за показване на нивото на потребителските точки.
* **Поддръжка на стилизация в зависимост от темата**:
  + Използва стилове, които се променят в зависимост от избраната тема на приложението (светла или тъмна).

Компонентът **Acc** предоставя полезна информация за потребителския акаунт и осигурява възможност за навигация към началната страница на приложението.

#### “utils” папка:

В тази директория се намират помощни функции, константи или други файлове, които се използват в различни части на приложението.

Файлът **AppContext.tsx** съдържа дефиницията на контекста на приложението (**AppContext**) и компонента **AppProvider**, който го предоставя на дъщерните компоненти. Ето основните характеристики на този файл:

* **Контекст на приложението (AppContext)**:
  + Контекстът представлява обект, който съдържа стойности на различни глобални променливи и функции, достъпни в рамките на цялото приложение.
  + Тези стойности включват език по подразбиране, настройки за темата, информация за потребителския акаунт и други.
* **Компонент AppProvider**:
  + Компонентът **AppProvider** е отговорен за предоставянето на контекста на приложението на всички дъщерни компоненти.
  + Използва се **createContext** функцията от React за създаване на контекст.
  + Стойността на контекста се задава чрез **useState** и **useRef** хукове, които съхраняват и управляват различни стойности и референции, необходими за приложението.
  + Използва се **useEffect** за извършване на различни действия при монтирането и ъпдейтването на компонента, като например проверка на системната тема, зареждане на потребителски данни и др.
  + Контекстът се предоставя чрез **AppContext.Provider** компонента и се задава като стойност на свойството **value**.

Файлът **AppProvider** е важен елемент от архитектурата на приложението, тъй като предоставя възможност за споделяне на информация между различните компоненти и осигурява консистентност и достъпност на глобални данни и функционалности.

Файлът **HomepageContext.tsx** представлява дефиниция на контекста за началната страница (**HomePageContext**) и компонента **HomePageProvider**, който предоставя този контекст на дъщерните компоненти на началната страница. Ето ключовите аспекти на този файл:

* **Контекст за началната страница (HomePageContext)**:
  + Контекстът **HomePageContext** е създаден с помощта на **createContext** функцията от React и предоставя информация и функционалност, свързани с началната страница на приложението.
  + Стойностите на контекста включват списъци с популярни, в тренда и в момента възпроизвеждани филми, състояния за показване на модални прозорци и грешки, както и функции за управление на тези състояния.
* **Компонент HomePageProvider**:
  + Компонентът **HomePageProvider** е отговорен за предоставянето на контекста за началната страница на дъщерните компоненти.
  + Използва се **useContext** хук за достъп до стойностите на контекста на приложението (**AppContext**), за да се получи езикът по подразбиране.
  + Стойностите за популярни, в тренда и в момента възпроизвеждани филми се зареждат чрез изпълнението на асинхронни функции като **PopularMovies**, **TrendingMovies** и **NowPlayingMovies**, където езикът се подава като параметър.
  + Връщат се списъци с филми, като се взима предвид езикът на потребителя. В случай на грешка се показва съобщение за грешка.
  + Компонентът се отчита за монтирането и разглеждането на асинхронните операции с помощта на хуковете **useEffect** и **useRef**.
  + Контекстът за началната страница се предоставя на дъщерните компоненти чрез **HomePageContext.Provider** компонента и се задава като стойност на свойството **value**.

Този файл играе ключова роля в осигуряването на началната страница с достъп до нужната информация и функционалност, като предоставя начин за споделяне на тази информация между различните компоненти, които съставят началната страница.

## Бакенд

"Бакенд" се отнася до частта от софтуерната архитектура, която се грижи за обработката на данни, бизнес логиката и връзката с базите данни. Това е тази част от приложението, която работи на сървъра и осигурява функционалността, която клиентското приложение използва.

### Език за програмиране

#### JavaScript:

JavaScript е високо нивов език за програмиране, който се използва широко за разработка на динамични уеб приложения. Той позволява добавяне на интерактивност към уеб страниците, манипулиране на DOM елементи и взаимодействие с потребителите на страницата.

#### TypeScript :

TypeScript е супермножество на JavaScript, което добавя статично типизиране и други напредни функции към езика. Той предоставя по-голяма сигурност и устойчивост на грешки по време на разработката, като същевременно запазва синтаксиса и функционалността на JavaScript. TypeScript е широко използван в уеб разработката за създаване на по-големи и по-сложни уеб приложения.

### Пакети

#### bcrypt:

bcrypt е библиотека за хеширане на пароли. Тя предоставя сигурен начин за съхранение на пароли в бази данни чрез хеширане със сол (salt), което усложнява атаките. Обикновено се използва в приложенията, които изискват аутентикация на потребители.

#### body-parser

body-parser е middleware за Express.js, който обработва тялото на HTTP заявките, например данни от формуляри или JSON данни, и ги прави достъпни в **req.body**. Често се използва за обработка на данни, изпратени от клиентската страна, например при обработка на формуляри в уеб приложенията.

#### cors

cors (Cross-Origin Resource Sharing) е middleware за Express.js, който позволява заявките към вашето приложение от други домейни. Без този middleware, браузърът ще блокира такива заявки поради политиката на съществуващите сигурност. Полезен е, когато вашето приложение се нуждае от достъп от други домейни, като например когато API-то се използва от уеб страници на различни домейни.

#### dotenv

dotenv е библиотека, която зарежда променливи от **.env** файл в **process.env**. Това позволява на разработчиците да управляват конфигурационни променливи за приложението си в един файл. Често се използва за управление на чувствителна информация като ключове за достъп до бази данни или API.

#### express

express е минималистичен уеб сървър за Node.js, който улеснява създаването на API и уеб приложения. express се използва широко за създаване на уеб приложения и API, тъй като е лесен за използване и разширяване.

#### fs

fs е модул в Node.js, който предоставя функции за работа с файлова система. Той позволява четене, запис и манипулиране на файлове. Често се използва за работа с файлове, като например четене на конфигурационни файлове или запис на логове.

### Файлове и технологии

Файлът **server.ts** е основният файл за стартиране на бакенда. Той използва Express за създаване на уеб сървър и предлага API за комуникация с клиентската страна на приложението. Ето някои от най-важните компоненти и функции в този файл:

1. **Импортиране на необходимите пакети**:
   * **express**: Използва се за създаване на уеб сървър.
   * **cors**: Позволява Cross-Origin Resource Sharing (CORS) между различни домейни.
   * **mongoose**: Библиотека за свързване с MongoDB база данни.
   * **https**: Позволява създаване на HTTPS сървър.
   * **fs**: Предоставя функционалност за работа със файлове.
   * **crypto**: Използва се за генериране на случайни символи.
   * **dotenv**: Помага за зареждането на променливи от **.env** файл.
   * **fetch**: Използва се за извличане на ресурси от мрежата.
2. **Импортиране на рутери за API пътища**:
   * **movieRouter**, **tvRouter**, **userRouter**, **filterRouter**, **countryRouter**, **rateRouter**, **commentRouter**: Рутери, които се използват за обработка на заявки за различните ендпойнти на API.
3. **Създаване на уеб сървър с Express**:
   * Създава се нов екземпляр на Express и се запазва в променливата **app**.
4. **Настройка на CORS**:
   * Използва се **cors** middleware, за да се позволи достъп от определени източници до API на приложението.
5. **Свързване с базата данни MongoDB**:
   * Използва се **mongoose.connect** за свързване с MongoDB базата данни чрез предоставените конфигурационни данни.
6. **Използване на маршрути**:
   * Поддържат се различни маршрути чрез използването на различни рутери (**app.use('/route', router)**).
7. **Настройка на HTTPS сървър**:
   * Създава се HTTPS сървър с помощта на **https.createServer**.
   * Използва се ключ и сертификат, предоставени във файловете **server.key** и **server.cert**.
8. **Запуск на сървъра**:
   * Сървърът се стартира да слуша определен порт (**server.listen(PORT)**).
9. **Дефиниране на маршрут /asd**:
   * В този маршрут се изпраща заявка към външен API и се връща отговор, който съдържа случаен низ и данни от API.

Този файл е сърцето на твоето уеб приложение, където се настройват всички основни компоненти за стартиране на сървъра и обработка на заявките от клиентите.

#### “config” папка:

Файлът **env.interface.ts** дефинира интерфейс за конфигурационните променливи на приложението. Този интерфейс е използван за типизиране на обект, който съдържа конфигурационни данни, които се използват в различни части на приложението. Ето някои от ключовите полета и техните обяснения:

* **PORT**: Номер на порта, на който сървърът трябва да слуша за заявки.
* **URL**: URL адрес на сървъра или домейн.
* **JWT\_SECRET\_KEY**: Тайният ключ, използван за подписване и верифициране на JSON Web Tokens.
* **connString**: Стринг за връзка с базата данни MongoDB.
* **movieURL**: URL адрес за заявки свързани с филми.
* **tvURL**: URL адрес за заявки свързани с телевизионни предавания.
* **searchURL**: URL адрес за заявки свързани с търсене.
* **basicURL**: Базов URL адрес на приложението.
* **discoverURL**: URL адрес за заявки за откриване на съдържание.
* **genreMovieURL**: URL адрес за заявки за жанрове на филми.
* **genreTVURL**: URL адрес за заявки за жанрове на телевизионни предавания.
* **countryURL**: URL адрес за заявки за държави.
* **accessToken**: Токен за достъп до външни API, като например TheMovieDB API.
* **bg**: Стойност за език на интерфейса "bg-BG".
* **en**: Стойност за език на интерфейса "en-US".
* **tv**: Стойност за телевизионни предавания.
* **movie**: Стойност за филми.
* **authUser**: Потребителско име за аутентикация в базата данни.
* **authPass**: Парола за аутентикация в базата данни.
* **db**: Име на базата данни.
* **mailerAPI**: API ключ за изпращане на имейли.
* **siteAPIKey**: Ключ за аутентикация към сайтовете на външните API.

Този интерфейс помага за организиране на конфигурационните данни и гарантира, че те са правилно типизирани при използване в различни части на приложението.

Файлът **env.ts** служи за зареждане на конфигурационните променливи от системната среда (environment variables) и тяхната организация в обект, който съответства на интерфейса **Env**. Ето какво прави този файл:

1. Импортира интерфейса **Env** от файла **env.interface.ts**, който съдържа дефиницията на конфигурационните променливи.
2. Използва **process.env** за достъп до системната среда и зарежда стойностите на конфигурационните променливи.
3. Преобразува и парсира стойностите към подходящ тип, където е необходимо (например, числови стойности се парсират към **number**).
4. Създава обект **env**, който съдържа конфигурационните променливи, подредени в съответствие с интерфейса **Env**.
5. Изнася обекта **env** като **default** експорт, за да може да бъде използван в други части на приложението.

Този файл е отговорен за зареждане на конфигурационните променливи, които се използват в приложението, и ги предоставя в удобен за използване формат. Това прави настройката и конфигурирането на приложението по-лесно и по-гъвкаво, като позволява на разработчиците да променят настройките без промяна на кода на приложението.

Top of Form

#### “controllers” папка

Този файл се нарича **controller.comment.ts** и съдържа няколко контролера (controller) за управление на коментарите в приложението. Контролерите са функции, които се използват за обработка на заявките, получени от клиентската страна чрез рутерите в Express приложението.

Ето кратко обяснение за всяка от контролерните функции:

1. **addComment**: Тази функция се извиква, когато клиентът иска да добави нов коментар. Получава информация за коментара от заявката и извиква функцията **createComment** от **../utils/comment**, която създава нов коментар в базата данни. След успешното създаване на коментара, функцията връща отговор към клиента със статус 200 и съобщение за успешно добавен коментар.
2. **deleteComment**: Тази функция се извиква, когато клиентът иска да изтрие съществуващ коментар. Получава идентификатора на коментара от заявката и извиква функцията **deleteCommentWithNested** от **../utils/comment**, която изтрива коментара и всички вложени коментари към него от базата данни. След успешното изтриване на коментара, функцията връща отговор към клиента със статус 200 и съобщение за успешно изтрит коментар.
3. **replyComment**: Тази функция се извиква, когато клиентът иска да добави отговор на съществуващ коментар. Получава информация за отговора от заявката и извиква функцията **createReplyComment** от **../utils/comment**, която създава нов коментар като отговор на вече съществуващ. След успешното създаване на отговора, функцията връща отговор към клиента със статус 200 и съобщение за успешно добавен отговор.
4. **viewMore**: Тази функция се извиква, когато клиентът иска да зареди още коментари за определен медиен обект (филм или сериал). Получава информация за медията и родителския коментар от заявката и извиква функцията **nestedComments** от **../utils/comment**, която връща всички вложени коментари към определен медиен обект. След успешното зареждане на коментарите, функцията връща отговор към клиента със статус 200 и съобщение за успешно заредени коментари, както и самите коментари.

Файлът **controller.country.ts** съдържа две контролерни функции за управление на операции свързани с държави в приложението. Ето какво правят тези функции:

1. **findCountry**: Тази функция се извиква, когато клиентът търси държави по ключова дума (например, име на държава). Получава ключовата дума от заявката и използва модела **Country** и метода **.aggregate()** за да търси държави, които съдържат ключовата дума в английското си име. След успешното намиране на държавите, функцията връща отговор към клиента със статус 200 и списък с намерените държави.
2. **countryPagination**: Тази функция се извиква, когато клиентът иска да извлече определен брой държави, като използва пагинация. Получава параметри за езика, отместване (offset) и лимит (limit) от заявката. Използва модела **Country** и метода **.aggregate()** за да извлече държавите от базата данни, като прескача определен брой записи (offset) и извлича определен брой записи (limit). След успешното извличане на държавите, функцията връща отговор към клиента със статус 200 и списък с извлечените държави.

В допълнение, във файла се използва модулът **handle**, който вероятно представлява обект или функции за обработка на различни видове отговори към клиента, като например успешен отговор, отговор за ненамерени данни и други.

Top of Form

Файлът **controller.filter.ts** съдържа две контролерни функции, които се използват за обработка на заявки, свързани с филтриране и търсене в приложението. Ето какво правят тези функции:

1. **quickSearch**: Тази функция се извиква, когато клиентът изпрати заявка за бързо търсене на ключова дума в мултимедийни ресурси като филми и телевизионни предавания. Получава ключовата дума и езика от заявката. След това прави заявка към външно API, използвайки ключовата дума и други параметри като език и страница. По получаване на отговор от API, функцията филтрира резултатите, извличайки само филми и телевизионни предавания, след което ги обработва и връща към клиента със статус 200 и списък с намерените мултимедийни ресурси.
2. **discover**: Тази функция се извиква, когато клиентът изпрати заявка за откриване на мултимедийни ресурси (филми, телевизионни предавания) чрез филтриране по различни критерии като жанр, година на издаване, държава и други. Получава различни параметри от заявката, включително ключова дума, тип на мултимедийния ресурс, жанрове, години, сортиране, страница и език. В зависимост от дали е получена ключова дума или не, функцията прави заявка към външно API, използвайки параметрите от заявката, и след това обработва резултатите и ги връща към клиента със статус 200 и списък с намерените мултимедийни ресурси.

Тези функции също така използват модула **handle**, който съдържа функции за обработка на различни видове отговори към клиента, като успешен отговор, отговор за ненамерени данни и други.

Файлът **controller.rated.ts** съдържа контролерна функция, която се използва за добавяне на оценка към определен мултимедиен ресурс (филм или телевизионно предаване). Ето какво прави тази функция:

1. **addRate**: Тази функция получава заявка за добавяне на оценка и извлича необходимите данни като идентификатор на мултимедийния ресурс (**mediaId**), тип на мултимедийния ресурс (**media**), оценката (**rating**) и езикът на отговора (**lang**). След това функцията извиква функцията **addRated** от модула **rated**, която се използва за добавяне на оценка в базата данни. След успешното добавяне на оценката функцията връща към клиента отговор със статус 200 и подходящо съобщение за успешно оценяване, в зависимост от избрания език.

Тази контролерна функция е част от модула за управление на оценки (**rated**) и се използва за обработка на заявки за оценяване на мултимедийни ресурси в приложението.

Файлът **controller.tv.ts** съдържа редица контролерни функции, които управляват заявки, свързани с телевизионните предавания. Ето какво правят тези функции:

1. **getById**: Тази функция се използва за извличане на информация за конкретно телевизионно предаване по зададено идентификатор.
2. **search**: Функцията извършва търсене на телевизионни предавания по даден текстов низ или ключови думи.
3. **getSimilar**: Тази функция връща списък с подобни телевизионни предавания на базата на предоставено телевизионно предаване.
4. **getByGenre**: Функцията се използва за извличане на телевизионни предавания по зададен жанр.
5. **getByKeyword**: Тази функция връща телевизионни предавания, свързани с определени ключови думи.
6. **getLatest**: Функцията връща най-новите телевизионни предавания.
7. **getPopular**: Тази функция връща най-популярните телевизионни предавания.
8. **getTopRated**: Функцията връща най-високо оценените телевизионни предавания.
9. **toggleLike**: Тази функция се използва за превключване на харесването на дадено телевизионно предаване.
10. **toggleWatched**: Функцията се използва за превключване на статуса на гледане на дадено телевизионно предаване.
11. **toggleFavorite**: Тази функция се използва за превключване на статуса на отметката "любимо" на дадено телевизионно предаване.
12. **getEpisodes**: Функцията връща епизоди на дадено телевизионно предаване.

Тези функции обработват различни видове заявки и осигуряват информация за телевизионни предавания, като се използва връзка с базата данни и други източници на данни.

Този файл “controller.movie.ts” е част от приложение, написано с Express.js и TypeScript, и съдържа контролери, които обработват заявки свързани с филми. Вотът им е да управляват логиката на приложението, като извличане на данни от базата данни, обработка на тези данни и връщане на резултати към клиента.

Ето кратко обяснение за всяка от функциите в този файл:

1. **details**: Извлича подробна информация за даден филм, включително коментари, препоръки и взаимодействия на потребителите като оценяване и отбелязване като гледан.
2. **preview**: Връща основна информация за филм, предимно за предварителен преглед. Също така се грижи за взаимодействията на потребителите като проверяване дали потребителят е харесал филма.
3. **movieTrailer**: Извлича трейлъри за даден филм, използвайки външно API.
4. **PopularTrendingFetchMovies**: Помощна функция, която извлича популярни или трендиращи филми в зависимост от зададени критерии като език, номер на страница и други.
5. **popular**: Извлича списък от популярни филми в зависимост от зададените параметри като език и номер на страница.
6. **trending**: Извлича списък от трендиращи филми в зависимост от времеви период (например, днес, тази седмица).
7. **nowPlaying**: Извлича списък от филми, които в момента се излъчват в кината.
8. **findAll**: Извлича всички филми от базата данни.
9. **deleteAll**: Изтрива всички филми от базата данни.

Тези функции работят асинхронно, използвайки ключовата дума **async** и **await** за обработка на асинхронни операции като извличане на данни от базата данни или изпращане на заявки към външни API-и. Те също така се грижат за коректното обработване на грешки и връщане на адекватни отговори към клиента.

Файлът **controller.user.ts** съдържа редица контролерни функции, които управляват заявки, свързани с управление на потребителски профили и данни. Ето какво правят тези функции:

1. **Импортиране на модули и дефиниране на интерфейси**: На първите редове има импортиране на различни модули и дефиниране на интерфейси за използване по-късно.
2. **Функции за автентикация и регистрация**: **login** функцията обработва заявки за вход, като проверява дали потребител със съответното потребителско име или имейл и парола съществува в базата данни. Ако входът е успешен, генерира JWT токен и го връща в отговор. Функцията **register** се използва за създаване на нов потребител в системата. Проверява дали потребителското име и имейл адрес са уникални и ако са, създава нов потребител в базата данни и изпраща имейл за потвърждение на регистрацията.
3. **Функции за управление на профила**: В този блок са реализирани различни функции за управление на потребителския профил, като промяна на ролята, промяна на парола, промяна на имейл адреса, промяна на настройките за обществен достъп, изпращане на имейл за потвърждение на регистрация и други.
4. **Функции за управление на харесвания и гледания**: Тук са функции за добавяне, премахване и извличане на списъци с харесани и гледани медии.
5. **Други помощни функции**: В този раздел са помощни функции като изтриване на потребителски профил, извличане на списъци с потребители и др.
6. **Изход и изтриване на сесия**: **logout** функцията се използва за изчистване на сесията на потребителя, като премахва JWT токена.

#### “middleware” папка

Middleware в контекста на уеб приложенията, особено в рамките на Express.js, представлява функция, която има достъп до заявката (request), отговора (response) и следващата функция в цикъла на заявката-отговора (next). Middleware функциите се изпълняват последователно и могат да извършват различни действия като обработка на заявката, манипулиране на отговора, аутентикация, авторизация, обработка на грешки и други.

В контекста на Express.js, когато създавате уеб приложение, можете да добавяте middleware функции към пътевете или към целия приложението, като използвате **app.use()** метода на инстанцията на Express приложението. Тези middleware функции се изпълняват при всяка заявка към съответния път или приложение, което позволява гъвкавост при обработката на заявките.

Middleware функциите са съществено средство за създаване на по-сложна логика в уеб приложенията и могат да се използват за изпълнение на различни операции преди и след обработката на заявките от контролерите или другите части на приложението.

„**adminAuth.ts”** дефинира middleware функцията **adminAuth**, която се използва за аутентикация на администраторски потребители в уеб приложение, базирано на Express.

1. **Импортиране на модули и дефиниране на интерфейси**: На първите редове има импортиране на модули и дефиниране на интерфейс **RequestType**, който наследява интерфейса **Request** и добавя два допълнителни полета: **user** и **token**.
2. **Дефиниране на тип за достъп до потребителски данни**: Типът **userAccessType** описва форматът на данните за потребителя, който ще бъде използван от middleware функцията за аутентикация.
3. **Middleware функцията за аутентикация на администратори**: Функцията **adminAuth** приема три параметъра: **req** (заявка), **res** (отговор) и **next** (следваща функция). Тя се използва за проверка на валидността на JWT токена в HTTP заглавката "access-token" и за проверка на ролята на потребителя.
   * Първо се извлича JWT токенът от HTTP заглавката "access-token".
   * След това токенът се верифицира с помощта на JWT секретния ключ **JWT\_SECRET\_KEY**.
   * Ако токенът е валиден и успешно верифициран, се извличат данните за потребителя от декодирания токен.
   * Потребителските данни се проверяват, за да се уверим, че потребителят има роля "admin".
   * Ако потребителят е администратор, тогава данните за потребителя се добавят към обекта **req**, за да бъдат достъпни в последващите middleware функции.
   * Ако потребителят не е администратор, се връща отговор със статус 401 (неоторизиран достъп).
   * Ако има грешка при аутентикацията (например, невалиден токен), се връща отговор със статус 401 и съобщение за грешка.
4. **Експорт на middleware функцията**: Функцията **adminAuth** се експортира, за да може да бъде използвана в други части на приложението като middleware за аутентикация на администраторски потребители.

Това е общ преглед на файла. Ако имате конкретни въпроси или нуждаете от допълнителни обяснения, не се колебайте да попитате!

Файлът **auth.ts** представлява middleware функция, която се използва за аутентикация на потребителите в уеб приложението. В конкретния случай, тази middleware функция е създадена за Express.js приложения.

Ето какво прави тази middleware функция:

1. Проверява дали в заглавките на заявката се предава токен за достъп (access token). Този токен се очаква да бъде предаден в заглавките на заявката под идентификатор "access-token".
2. Ако токенът за достъп липсва или е **undefined**, функцията връща отговор със статус код 401 (неоторизиран достъп) и съобщение, че липсва токен за достъп.
3. Ако токенът за достъп е наличен, той се верифицира използвайки **jwt.verify()**, като се използва секретният ключ за подписване на токените (JWT\_SECRET\_KEY). Ако токенът е валиден, резултатът от декодирането му съдържа информация за потребителя.
4. Ако токенът не е валиден или няма потребителска информация, функцията връща отговор със статус код 401 и съобщение, че няма потребител.
5. Ако токенът за достъп е валиден и съдържа информация за потребителя, тази информация се добавя към обекта **req**, така че контролерите или другите middleware функции да могат да я използват в последствие. След това функцията извиква **next()**, за да продължи към следващата middleware функция или контролер в цикъла на заявката-отговора.

Тази middleware функция е от съществено значение за асигурирането на защита на определени ресурси или рутери в уеб приложението, като изисква аутентикация на потребителите, преди те да получат достъп до тези ресурси.

Файлът **authNext.ts** също представлява middleware функция за аутентикация, като работи по сходен начин като предишния файл, но с някои разлики в имплементацията. Ето какво прави тази middleware функция:

1. Проверява дали в заглавките на заявката се предава токен за достъп (access token). Токенът за достъп се очаква да бъде предаден в заглавките на заявката под идентификатор "access-token".
2. Ако токенът за достъп липсва или е **'undefined'**, той се смята за невалиден и свойството **isAuth** на обекта **req** се установява на **false**. След това се извиква **next()**, за да продължи към следващата middleware функция или контролер в цикъла на заявката-отговора.
3. Ако токенът за достъп е наличен, той се верифицира използвайки **jwt.verify()**, като се използва секретният ключ за подписване на токените (JWT\_SECRET\_KEY). Ако токенът е валиден, резултатът от декодирането му съдържа информация за потребителя.
4. Ако токенът не е валиден или няма потребителска информация, свойството **isAuth** на обекта **req** се установява на **false**. След това се извиква **next()**.
5. Ако токенът за достъп е валиден и съдържа информация за потребителя, тази информация се добавя към свойството **user** на обекта **req**, а свойството **isAuth** се установява на **true**. След това се извиква **next()**.
6. Ако възникне грешка при верифицирането на токена или при неговата обработка, се извежда съобщение за грешка и се връща отговор със статус код 401 (неоторизиран достъп).

Общо взето, тази middleware функция играе ролята на аутентикационен филтър, който проверява валидността на токена за достъп и управлява достъпа на потребителите до защитени ресурси в уеб приложението.

Файлът **siteAuth.ts** представлява middleware функция, която се използва за удостоверяване на заявки към сайта чрез API ключ. Ето какво прави тази middleware функция:

1. Извлича секретният API ключ за сайта от конфигурационния файл (**env**).
2. Проверява дали API ключът, предаден в заглавките на заявката под идентификатор "site-api-key", съвпада с очакваният API ключ за сайта.
3. Ако предаденият API ключ не съвпада с очакваният, връща се отговор със статус код 403 (забранен достъп) и съобщение "Forbidden". Това означава, че заявката е отхвърлена поради липса на правилни удостоверителни данни.
4. Ако предаденият API ключ съвпада с очакваният, middleware функцията извиква **next()**, което позволява заявката да продължи към следващата middleware функция или контролер в цикъла на заявката-отговора.

Тази middleware функция осигурява основна защита за сайта, като изисква всяка заявка към API-то да предоставя валиден API ключ. Това помага да се предотвратят нежелани или недоволни заявки към сайта от непозволени източници.

#### “middleware” папка

Файлът **comment.route.ts** дефинира пътищата (routes) за коментарите в приложението, като свързва всяка от тях със съответните контролери и middleware функции. Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортира пакетът **express**, контролерът за коментари (**CommentController**) и middleware функцията за удостоверяване (**auth**).
2. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
3. Дефинира пътищата за различните операции с коментари, като използва методите на HTTP заявките (**GET**, **POST**, **DELETE** и т.н.). Всяка от тези пътища е свързана със съответния контролер за коментари.
4. Всеки път, преди изпълнението на даден контролер, middleware функцията за удостоверяване (**auth**) се извиква. Това осигурява, че всички заявки към тези пътища са удостоверени преди да се изпълни съответната операция с коментарите.
5. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операциите с коментари и гарантира, че всички заявки към тях са удостоверени преди да бъдат изпълнени.

Top of Form

Файлът **country.route.ts** дефинира пътищата (routes) за операции свързани с държавите в приложението. Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортира пакетът **express** и контролерът за държавите (**countryController**).
2. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
3. Дефинира два пътища: **/find** и **/pagination**, които са свързани със съответните операции в контролера за държавите.
4. Всяко от тези пътища е асоциирано с определена функция от контролера за държавите. Например, пътят **/find** е свързан с функцията **findCountry**, която търси държава, докато пътят **/pagination** е свързан с функцията **countryPagination**, която осъществява пагинация на списъка с държави.
5. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операции свързани с държавите в приложението и асоциира всяка операция със съответната функция от контролера.

Файлът **country.route.ts** дефинира пътищата (routes) за операции свързани с държавите в приложението. Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортира пакетът **express** и контролерът за държавите (**countryController**).
2. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
3. Дефинира два пътища: **/find** и **/pagination**, които са свързани със съответните операции в контролера за държавите.
4. Всяко от тези пътища е асоциирано с определена функция от контролера за държавите. Например, пътят **/find** е свързан с функцията **findCountry**, която търси държава, докато пътят **/pagination** е свързан с функцията **countryPagination**, която осъществява пагинация на списъка с държави.
5. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операции свързани с държавите в приложението и асоциира всяка операция със съответната функция от контролера.

Файлът **movie.route.ts** дефинира пътищата (routes) за операции, свързани с филмите в приложението. Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортира пакетът **express** и контролерът за филмове (**movieController**).
2. Импортира middleware функцията **authNext** от пътя **../middleware/authNext**, която осъществява автентикацията на потребителя преди достъп до определени ресурси.
3. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
4. Дефинира редица пътища, свързани с различни операции в контролера за филмове (**movieController**):
   * **/details/:id**: Показва детайлите за определен филм, като изисква аутентикация чрез middleware функцията **authNext**.
   * **/preview/:id**: Показва преглед на определен филм, изисквайки аутентикация.
   * **/trailer/:id**: Показва трейлър на определен филм, без изискване за аутентикация.
   * **/popular**: Показва списък с популярни филми, без изискване за аутентикация.
   * **/trending**: Показва списък с трендиращи филми, без изискване за аутентикация.
   * **/nowPlaying**: Показва списък с филми, които в момента се възпроизвеждат, без изискване за аутентикация.
   * **/findAll**: Намира всички филми, без изискване за аутентикация.
   * **/deleteAll**: Изтрива всички филми, без изискване за аутентикация.
5. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операции свързани с филмите в приложението и асоциира всяка операция със съответната функция от контролера. Някои от операциите изискват аутентикация преди достъп до съдържанието, докато други не изискват.

Файлът **rated.route.ts** дефинира път за операцията за добавяне на рейтинг към даден обект. Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортира пакетът **express** и контролерът за рейтинг (**RatedController**).
2. Импортира middleware функцията **auth** от пътя **../middleware/auth**, която осъществява автентикацията на потребителя преди достъп до определени ресурси.
3. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
4. Дефинира пътя **/add**, който приема GET заявка и извиква функцията **addRate** от контролера за рейтинг (**RatedController**). Пътят изисква аутентикация чрез middleware функцията **auth**.
5. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операции, свързани с рейтинг на обекти и асоциира пътя **/add** с функцията за добавяне на рейтинг, като изисква аутентикация преди достъп до ресурса.

Файлът **tv.route.ts** дефинира пътищата за операции, свързани с телевизионните предавания (TV). Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортира пакетът **express** и контролерът за телевизионни предавания (**TVController**).
2. Импортира middleware функцията **authNext** от пътя **../middleware/authNext**, която осъществява автентикацията на потребителя преди достъп до определени ресурси.
3. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
4. Дефинира редица пътища, които приемат GET заявки и извикват съответните функции от контролера за телевизионни предавания (**TVController**). Някои от тези пътища изискват аутентикация чрез middleware функцията **authNext**.
5. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операции като показване на детайли, преглед на предавания, трейлъри, списъци с популярни предавания, предавания в момента на излъчване и други дейности свързани с телевизионни предавания.

Top of Form

Файлът **users.route.ts** управлява пътищата, свързани с потребителите в системата. Ето какво прави този файл:

1. Импортира съответните пакети и контролери. В този случай се импортират пакетът **express** и контролерът за потребителите (**UserController**), както и различни middleware функции за аутентикация (**auth**, **adminAuth**, **authNext**).
2. Създава нов router обект от **express.Router()**, който се използва за дефиниране на пътищата.
3. Дефинира редица пътища, които приемат различни видове HTTP заявки (POST, GET) и извикват съответните функции от контролера за потребителите (**UserController**). Някои от тези пътища изискват аутентикация чрез различните middleware функции (**auth**, **adminAuth**, **authNext**), като например пътят за изтриване на всички потребители, ролята на потребителя и други операции, които може да извършва само администратор.
4. Изнася router обекта, за да може да бъде използван от други части на приложението.

Този файл управлява пътищата за операции като регистрация, вход, изход, потвърждение на електронната поща, управление на профила, промяна на ролята и други операции, свързани с потребителите в системата.

#### “utils” папка

**comment.ts** управлява създаването, извличането и изтриването на коментари, включително вложени коментари, за медийни обекти като филми или телевизионни предавания. Това е описание на :

1. **createComment**:
   * Тази функция създава нов коментар в базата данни. Приема параметри като **id** (идентификатор на коментара), **username** (потребителско име на автора на коментара), **content** (съдържание на коментара), **media** (вид на медията, като "movie" или "tv"), **mediaId** (идентификатор на медията) и опционален параметър **parentId** (идентификатор на родителски коментар, ако е отговор на друг коментар).
2. **createReplyComment**:
   * Тази функция създава отговорен коментар в базата данни. Подобно на **createComment**, тя приема параметри за идентификатор, потребителско име, съдържание, вид на медията, идентификатор на медията и идентификатор на родителския коментар.
3. **initComments**
   * Тази функция инициализира списък с коментари за определена медия и идентификатор на медията. Тя използва агрегация на данни за да извлече коментарите от базата данни, сортирани по време на създаване, и добавя информация за вложените коментари.
4. **deleteCommentWithNested**:
   * Тази функция изтрива коментар от базата данни и всички свързани с него вложени коментари. Приема параметър за идентификатор на коментара.
5. **nestedComments**:
   * Тази функция извлича вложените коментари за дадена медия, идентификатор на медията и родителски коментар. Подобно на **initComments**, тя използва агрегация на данни за да добави информация за вложените коментари към основните коментари.

Файлът **filter.ts** съдържа функции, които се използват за филтриране и обработка на медийни съдържания. Ето какво прави този файл:

1. Дефинира интерфейс **MediaI**, който обединява свойствата на интерфейсите **MovieI** и **TVI**. Този интерфейс добавя допълнителни свойства към медийните обекти, като **media\_type** (тип на медията - филм или сериал) и **check** (флаг, указващ дали потребителят е харесал този медиен обект).
2. Дефинира функция **searchProcess**, която обработва резултатите от търсене на медийни съдържания. Тази функция добавя информация за това дали потребителят е харесал медийните обекти в резултатите на търсенето, ако потребителят е аутентикиран.
3. Дефинира функция **processDiscover**, която извършва процеса на откриване на медийни съдържания според различни критерии като тип (филм, сериал или комбинирано), странициране, език, период на излизане, сортиране, жанрове и др.
4. Дефинира функция **processKeywordDiscover**, която извършва процеса на откриване на медийни съдържания по ключова дума. Тази функция работи подобно на **processDiscover**, но приема допълнителни параметри като ключова дума и филтрира резултатите според нея.
5. Дефинира помощна функция **sortFn**, която сортира медийните обекти според определен критерий като популярност, оценка, заглавие или дата на излизане.
6. Дефинира помощна функция **befDateCalc**, която сравнява дати и връща дали едната дата е преди другата.

Този файл предоставя функции за обработка и филтриране на медийни съдържания в приложението.

Файлът **movie.ts** съдържа функции, свързани с манипулиране на информация за филми:

1. **getMovie**: Тази функция изпраща заявка към API за извличане на информация за определен филм по зададен идентификатор (**movieId**) и език (**lang**). Връща информацията за филма в Promise.
2. **saveMovie**: Тази функция приема информация за филм (**movieDetails**) и език (**lang**) и създава нов обект **Movie**, който се съхранява в базата данни. Функцията връща създадения обект **Movie**.
3. **findMovie**: Тази функция търси в базата данни филм по даден идентификатор (**id**) и език (**lang**). Връща информацията за филма, ако е намерен, в противен случай връща **null**.
4. **saveMovieTwoLanguages**: Тази функция съхранява информация за филм на два езика. Приема идентификатор на филма (**id**), първия и втория език (**lang** и **sec**). За всяка версия на филма се извлича информацията и се съхранява като отделен обект **Movie**.
5. **processMovies**: Тази функция обработва списък от филми (**movies**) и ги съхранява в базата данни. Проверява съществуващите записи и добавя само новите филми. Филмите трябва да отговарят на определени критерии, като наличие на дата на излизане, постер, жанрове, продукционни компании и др.
6. **recMovies**: Тази функция връща препоръчителни филми за даден филм по неговия идентификатор (**id**) и език (**lang**). Извлича се списък с препоръчани филми от API и се обработват по същия начин като в **processMovies**.

Тези функции се използват за извличане, съхранение и обработка на информация за филми в приложението.

Файлът **rated.ts** съдържа функции за работа с рейтинги на потребителите:

1. **addRated**: Тази функция приема идентификатора на потребителя (**user**), идентификатора на медията (**mediaId**), типа на медията (**media**) и рейтинга (**rating**). Първоначално търси за запис в базата данни, който съответства на зададените критерии. Ако няма такъв запис, създава нов обект с подадените данни и го съхранява в базата данни. В противен случай обновява съществуващия запис с новия рейтинг.
2. **loadRated**: Тази функция приема потребителския обект (**user**), идентификатора на медията (**mediaId**) и типа на медията (**media**). Търси запис в базата данни, който съответства на зададените критерии. Връща намерения запис или **null** ако няма намерен такъв запис.

Тези функции се използват за добавяне и зареждане на рейтинги на медии от потребителите в приложението.

Файлът **tv.ts** съдържа функции за работа с телевизионни сериали:

1. **getTV**: Тази функция извлича информация за даден телевизионен сериал по зададен идентификатор (**tvId**) и език (**lang**). Използва се за извличане на данни за определен телевизионен сериал от външен източник, като например API за филми и сериали.
2. **saveTV**: Функцията приема данни за телевизионен сериал (**tvDetails**) и език (**lang**). Създава нов обект, който представлява телевизионния сериал, и го съхранява в базата данни.
3. **findTV**: Тази функция търси телевизионен сериал в базата данни по зададен идентификатор (**id**) и език (**lang**). Връща намерения телевизионен сериал или **null**, ако не е намерен.
4. **saveTVTwoLanguages**: Функцията извлича информация за телевизионен сериал на два различни езика и ги съхранява в базата данни.
5. **processTVs**: Тази функция обработва списък от телевизионни сериали и ги съхранява в базата данни след филтриране на валидните записи.
6. **recTVs**: Функцията извлича препоръки за телевизионни сериали, свързани с даден телевизионен сериал, по зададен идентификатор (**id**) и език (**lang**).

Файлът **user.ts** съдържа функции, свързани с потребителите:

1. **verifyEmail**: Тази функция изпраща имейл за потвърждение на регистрацията на потребител, който включва линк за потвърждение на имейл адреса.
2. **changeEmail**: Функцията изпраща имейл с линк за потвърждение за смяна на имейл адреса на потребителя.
3. **resendVerificationEmail**: Функцията изпраща повторно имейл за потвърждение на регистрацията на потребител, който включва линк за потвърждение на имейл адреса.
4. **togglePublicStatus**: Функцията променя публичния статус на потребителя между публичен и частен.
5. **toggleIsWatchedStatus**: Функцията превключва статуса "Гледан" на потребителя между активен и неактивен.
6. **addScore**: Функцията увеличава резултата на потребителя, който е гледал или оценил медия (филм или сериал).
7. **addWatched**: Функцията добавя филм или сериал към списъка с гледани от потребителя.
8. **statusLiked**: Функцията проверява дали потребителят е харесал даден филм или сериал.
9. **toggleLikedProcess**: Функцията превключва статуса "Харесан" на филм или сериал относно даден потребител.
10. **checkEmail**: Функцията проверява дали въведеният имейл адрес е уникален за регистрация.
11. **checkUsername**: Функцията проверява дали въведеното потребителско име е уникално за регистрация.
12. **RequestUserI**: Интерфейс, който описва структурата на обекта на потребителя, който се използва в приложението.

Top of Form

#### “view” папка

Файлът **filter.ts** съдържа няколко функции за управление на отговорите на сървъра:

1. **internalError**: Тази функция връща отговор със статус 404 и съобщение за грешка в случай на вътрешна грешка на сървъра. В зависимост от избрания език се връщат съобщения на български или английски.
2. **notFound**: Тази функция връща отговор със статус 404 и съобщение за липса на резултати. В зависимост от избрания език се връщат съобщения на български или английски.
3. **success**: Тази функция връща отговор със статус 200 и съобщение за успешно намерени резултати, като включва и медията, която е намерена.
4. **successKeywordFilter**: Тази функция връща отговор със статус 200 и съобщение за успешно намерени резултати след ключово филтриране, като включва и медията, която е намерена, както и информация за следващата страница, ако има такава.

Всички функции приемат обект на отговора на Express (**Response**) и избрания от потребителя език ("bg-BG" или "en-US").

Top of Form

Файлът **movie.ts** съдържа няколко функции за управление на отговорите на сървъра, свързани с обработката на филми:

1. **createdMovie**: Тази функция връща отговор със статус 200 и съобщение за успешно създаден филм, включващ детайлите за филма, коментарите към него (които са null в този случай) и препоръчани филми.
2. **internalError**: Тази функция връща отговор със статус 500 и съобщение за вътрешна грешка на сървъра.
3. **movieExists**: Тази функция връща отговор със статус 200 и съобщение, че филмът вече съществува, като включва детайлите за филма, коментарите към него и препоръчани филми.
4. **moviesFetched**: Тази функция връща отговор със статус 200 и съобщение за успешно извлечени филми, включващ списък с детайли за филмите.
5. **previewSuccess**: Тази функция връща отговор със статус 200 и съобщение за успешно извлечени детайли за филма.
6. **moviesDeleted**: Тази функция връща отговор със статус 200 и съобщение за успешно изтрити филми.

Всички функции приемат обект на отговора на Express (**Response**) и допълнителни данни, свързани със съответната операция.

Top of Form

Файлът **tv.ts** съдържа функции за управление на отговорите на сървъра, свързани с обработката на телевизионни предавания:

1. **createdTV**: Тази функция връща отговор със статус 201 и съобщение за успешно създадено телевизионно предаване, включващо детайлите за предаването, коментарите към него (които са null в този случай) и препоръчани предавания.
2. **internalError**: Тази функция връща отговор със статус 500 и съобщение за вътрешна грешка на сървъра.
3. **tvExists**: Тази функция връща отговор със статус 200 и съобщение, че телевизионното предаване вече съществува, като включва детайлите за предаването, коментарите към него и препоръчани предавания.
4. **tvShowsFetched**: Тази функция връща отговор със статус 200 и съобщение за успешно извлечени телевизионни предавания, включващ списък с детайли за предаванията.
5. **previewSuccess**: Тази функция връща отговор със статус 200 и съобщение за успешно извлечени детайли за телевизионно предаване.
6. **tvShowsDeleted**: Тази функция връща отговор със статус 200 и съобщение за успешно изтрити телевизионни предавания.

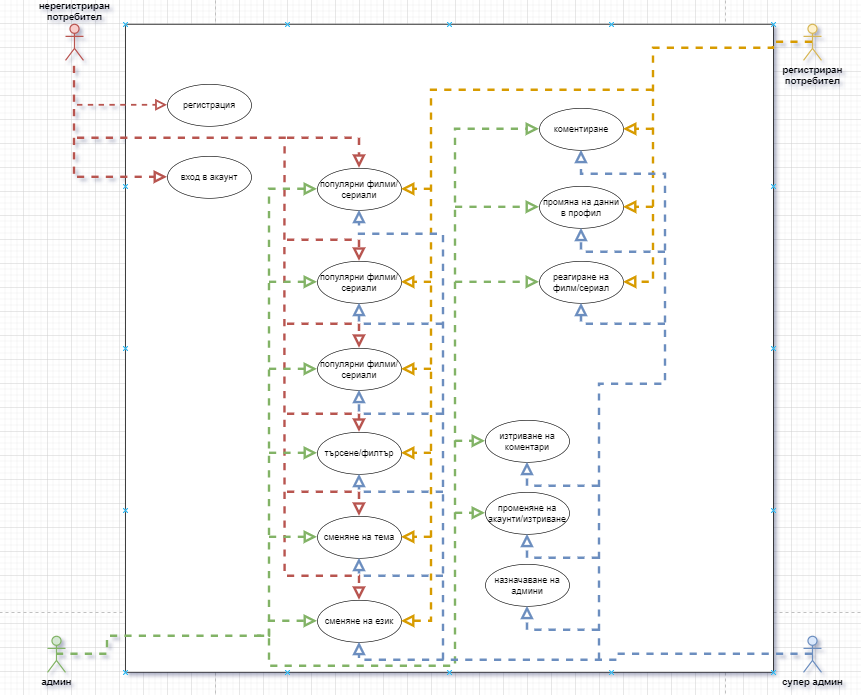
Всички функции приемат обект на отговора на Express (**Response**) и допълнителни данни, свързани със съответната операция.

Top of Form

Файлът **users.ts** съдържа функции, които управляват отговорите на сървъра, свързани с потребителските операции:

1. **serverError**: Тази функция връща отговор със статус 500 и съобщение за вътрешна грешка на сървъра. В зависимост от езика (**lang**), съобщението е на английски или български.
2. **sendUnauthorized**: Тази функция връща отговор със статус 401 и съобщение за невалидно потребителско име или парола. Съобщението може да бъде на английски или български, в зависимост от **lang**.
3. **sendRegisterConflict**: Тази функция връща отговор със статус 409 и съобщение за конфликт при регистрация (потребителско име или имейл вече съществуват). Съобщението може да бъде на английски или български, в зависимост от **lang**.
4. **loginSuccess**: Тази функция връща отговор със статус 200 и съобщение за успешно влязла в системата потребител. Връща и информация за потребителя, като съобщението може да бъде на английски или български, в зависимост от **lang**.
5. **registerSuccess**: Тази функция връща отговор със статус 200 и съобщение за успешно създаден потребител. Съобщението може да бъде на английски или български, в зависимост от **lang**.
6. **usersDeleted**: Тази функция връща отговор със статус 200 и съобщение за успешно изтрити потребители.
7. **sendCreated**: Тази функция връща отговор със статус 200 и съобщение за успешно създаден ресурс. Поддържа и добавяне на допълнителни HTTP заглавки, ако са предоставени.

## Use Case диаграма



Фигура 1: use case диаграма

## База данни

MongoDB е документориентирана, NoSQL база данни, която се използва за съхранение на структурирани и неструктурирани данни. Вместо таблици, както в релационните бази данни, MongoDB съхранява данни в документи, които са в JSON-подобен формат, наречен BSON (Binary JSON). Тази база данни е разработена да бъде гъвкава, мащабируема и лесна за използване.

MongoDB се ползва за различни цели, включително:

1. **Уеб приложения**: MongoDB е популярен избор за съхранение на данни в уеб приложения, особено в приложенията, които изискват гъвкава схема на данните и мащабируемост.
2. **Анализ на данни**: Поради своята гъвкавост и способността да се мащабира лесно, MongoDB се използва за анализ на големи обеми от данни.
3. **Системи за управление на съдържанието**: MongoDB е използван за съхранение на съдържание в системи за управление на съдържанието (CMS) и системи за управление на бази данни с отворен код.
4. **Интернет на нещата (IoT)**: MongoDB се използва в IoT приложения, където данните могат да бъдат неструктурирани и голям обем.
5. **Микросервизи и архитектури, базирани на услуги**: MongoDB е чест избор за съхранение на данни в микросервизни архитектури, тъй като ефективно справя със задачата да се мащабира и да работи в разпределена среда.

MongoDB се отличава с гъвкавостта си, лесната за използване JSON-подобна схема и способността да се мащабира хоризонтално за да се справи с големи обеми данни.

### База данни схеми

A screenshot of a computer

Description automatically generated

Фигура 2: модел на схема за филми

A screenshot of a computer

Description automatically generated

Фигура 3: модели на схеми на коментар, потребител, и реагирано

A screenshot of a computer

Description automatically generated

Фигура 4: модел на схема за сериал

# Използвана литература

## ReactJS

Link : <https://react.dev/>

## Vite

Link: <https://vitejs.dev/guide/>

## ExpressJS

Link: <https://expressjs.com/>

## TypeScript

Link: <https://www.w3schools.com/typescript/>

## Mongoose

Link: <https://mongoosejs.com/>

Линк към уебсайта : [**https://crackflix.site**](https://crackflix.site)