|  |  |
| --- | --- |
| Картина, която съдържа мотив, символ, Симетрия, дизайн  Описанието е генерирано автоматично | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**    **ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ**  **СПЕЦИАЛНОСТ КОМПЮТЪРНО И СОФТУЕРНО ИНЖЕНЕРСТВО** |

**Студент:** Валери Ивайлов Райков

**Фак. №:** 121222139

**Група:** 42Б

**Дисциплина:**

Програмиране за мобилни устройства

**Тема на проекта:**

Мобилно приложение за търсене и преглед на филми и телевизионни предавания

**Съдържание на документацията**

**[1. Увод - 3 -](#_Toc699)**

**[2. Анализ на съществуващи разработки - 3 -](#_Toc31390)**

[ Netflix - 3 -](#_Toc14830)

[ IMDb - 4 -](#_Toc25732)

[ JustWatch - 4 -](#_Toc21217)

[ TMDb (The Movie Database) - 4 -](#_Toc1634)

[Заключение: - 5 -](#_Toc21832)

**[3. Проектиране - 5 -](#_Toc15562)**

[ Целева аудитория: - 5 -](#_Toc6231)

[ Използвани данни: - 5 -](#_Toc25445)

[ Съхранение на данните: - 6 -](#_Toc27984)

[ Бизнес процеси в приложението: - 6 -](#_Toc16827)

[ Как ще бъдат достъпени функционалностите от потребителя? - 7 -](#_Toc5025)

[ Динамични маршрути - 7 -](#_Toc32620)

[ Компоненти за подобряване на UX: - 7 -](#_Toc27336)

[ Примерен user flow: - 7 -](#_Toc25319)

**[4. Реализация и приложение - 8 -](#_Toc14497)**

[ Services - 9 -](#_Toc30499)

[ Hooks - 13 -](#_Toc20161)

[ Context - 16 -](#_Toc22508)

[ App: - 18 -](#_Toc28683)

[ Components - 21 -](#_Toc8636)

[ Appwrite backend: - 22 -](#_Toc8110)

**[5. Потребителско ръководство - 25 -](#_Toc10555)**

**[6. Заключение - 37 -](#_Toc32729)**

**[7. Литература - 38 -](#_Toc11452)**

# Увод

С нарастващия интерес към стрийминг платформите и онлайн съдържанието, потребителите все по-често търсят централно място, където да откриват нови филми и сериали, да следят рейтинги и да получават препоръки. Настоящият проект цели създаване на мобилно приложение, което предоставя на потребителя интуитивен и бърз начин за намиране на филми и телевизионни сериали чрез свързване с външни бази данни. Приложението е универсално и е подходящо за потребители на всякаква възраст.

# Анализ на съществуващи разработки

В тази секция се разглеждат вече съществуващи приложения със сходна идея и цел, като се анализират техните функционалности, предимства и недостатъци.

## Netflix

**Описание:** Netflix е една от най-популярните платформи за стрийминг на филми и телевизионни сериали в света. Осигурява достъп до богата колекция от съдържание – както лицензирано, така и оригинално, включително световноизвестни продукции. Приложението е налично за мобилни устройства, уеб и телевизори, и предлага персонализирано потребителско изживяване.

**Предимства:**

* Огромна библиотека от съдържание – разнообразие от жанрове, държави и езици.
* Оригинално съдържание – Netflix създава собствени сериали и филми, често с високо качество.
* Интелигентни препоръки – базирани на историята на гледане и оценки.
* Мултиплатформена поддръжка – можеш да гледаш на всяко устройство с интернет.
* Възможност за изтегляне – съдържание за офлайн гледане.

**Недостатъци:**

* Платено съдържание – липса на безплатна версия, което го прави недостъпно за някои потребители.
* Ограничения по държави – не всички филми/сериали са налични навсякъде.
* Понякога твърде много съдържание – трудно е да избереш какво да гледаш.
* Няма потребителски рецензии – липсва възможност за писане на коментари и обсъждане между потребителите.

## IMDb

**Описание:** Официално мобилно приложение за достъп до огромна база данни с филми, актьори и оценки.

**Предимства:**

* Богата база с информация
* Много потребителски рецензии и рейтинги
* Надежден източник на данни

**Недостатъци:**

* Интерфейсът е претрупан
* Има реклами
* Няма персонализирани препоръки

## JustWatch

Описание: Приложение за търсене на филми и сериали по това къде са налични (Netflix, HBO, Disney+ и др.)

**Предимства:**

* Отличен филтър по платформи
* Удобен за потребители, които имат много абонаменти

**Недостатъци:**

* Малко информация за самите филми
* Няма потребителска регистрация или история на гледаното

## TMDb (The Movie Database)

**Описание:** Отворена платформа с богата информация, често използвана от други приложения (**включително и настоящото**)

**Предимства:**

* API за разработчици
* Добро визуално представяне на заглавията

**Недостатъци:**

* Понякога непълни данни
* По-слаба поддръжка на нови сериали

Заключение:

Настоящето приложение е изработено с цел да съчетае възможно най-много от предимствата на горепосочените разработки и същевременно да минимизира техните недостатъци. То предлага собствен потребителски интерфейс и уникална функционалност, които ще бъдат разгледани в следващите секции от документацията.

# Проектиране

## Целева аудитория:

Приложението е насочено към младежи и възрастни потребители, които обичат киното и търсят бърз достъп до информация за филми и сериали (както български така и чуждестранни).

## Използвани данни:

**Входни данни от потребителя:** Потребителят въвежда или взаимодейства със следните типове данни:

**Функционалност | Данни, които се въвеждат/генерират**

Регистрация/Вход | Email, Парола, Потребителско име

Запазване на филм | ID на филм, Заглавие, Плакат, Описание

Търсене на филм | Търсене по заглавие

Преглед на детайли|

за филм | Взема се ID от URL параметъра и се използва за fetch

Изход | Сесията се изтрива

**Структури на данни (класoве и обекти):**

**Потребител (User):**

type User = {

$id: string;

email: string;

name: string;

}

**Филм (Movie):**

type Movie = {

id: number;

title: string;

overview: string;

genres: string[];

poster\_path: string;

release\_date: string;

runtime: number;

vote\_average: number;

vote\_count: number;

production\_companies: string[];

budget: number;

revenue: number;

}

**Запазен филм (Saved movie):**

type SavedMovie = {

userId: string;

movieId: number;

title: string;

poster\_path: string;

savedAt: string;

}

## Съхранение на данните:

**Appwrite Database** – използва се за съхранение на запазените филми като документи, свързани с конкретен потребител.

**Appwrite Account** – управлява се сесията и идентичността на потребителя.

**TMDB API** – използва се за извличане на детайли за филми (няма нужда от локално съхранение).

Състоянието на UI се държи в useState, useEffect и custom hooks като useFetch.

## Бизнес процеси в приложението:

**Процес** | **Описание**

Регистрация | Създаване на потребител чрез Appwrite, логване на сесия

Вход/Изход | Създаване и изтриване на сесии с Appwrite

Запазване на филм | Изпращане на POST заявка към Appwrite DB

Преглед на филм | Взимане на ID (URL), fetch (TMDB), показване на детайли

Списък със запазени| Взимане от база с филмите на потребителя

Навигация | Чрез табове и динамични маршрути в Expo Router

## Как ще бъдат достъпени функционалностите от потребителя?

Интерфейс на приложението (UI)

Навигацията и структурата на потребителския интерфейс са реализирани с Expo Router (React Navigation) и използват:

**Раздел (Tab) | Екран**

Home (/) | Начален екран с популярни филми

Search (/search) | Позволява търсене на филми по ключова дума

Saved (/saved) | Показва списък със запазените от потребителя филми

Profile (/profile) | Показва информация за профила и бутон за изход

## Динамични маршрути

**tabs/movies/[id].tsx**- Това е страница с детайли за филм.

ID се взима от useLocalSearchParams(), после се fetch-ват данните и се визуализират.

## Компоненти за подобряване на UX:

**TabIcon** – показва икона и име на всеки таб.

**MovieCard, MovieInfo** – визуализират данни за филм по красив и четим начин.

**AlertModal** – за показване на съобщения (успех, грешка и т.н.).

**GoBack** – бутон за връщане към предишната страница.

**useModal hook** – управлява показването на модалните прозорци.

## Примерен user flow:

1. **Потребителят избира език и влиза** → отива на "Home"
2. **Разглежда филми и TV предавания** → цъка на един → отваря се movies/[id]
3. **Натиска "Save"** → филмът се запазва
4. В раздел "Saved" вижда запазените си филми
5. Може да излезе от профила чрез "Profile" tab

# Реализация и приложение

**Използвани технологии:**

* **React Native** - framework за изграждане на крос-платформени мобилни приложения с JavaScript / TypeScript и React. Той позволява използването на една кодова база за iOS и Android, като използва нативни компоненти.
* **Expo** - инструмент и набор от библиотеки, които улесняват разработката на React Native приложения, като предоставят готови решения за достъп до хардуерни функции (камера, GPS и др.) и бързо тестване.
* **Tailwind CSS** - е utility-first CSS framework, който позволява бързо стилизиране на уеб приложения чрез предварително дефинирани класове, вместо да се пише ръчен CSS.
* **TypeScript - строго типизиран superset на JavaScript, който добавя опционални типове и подобрена поддръжка за големи проекти, намалявайки грешките по време на разработка.**

**Източници на данни:**

* **TMDb API** - предоставя безплатен достъп до огромна база данни с филми, сериали и актьори, което го прави идеален за създаване на кино-приложения. Той предлага разнообразни endpoints за търсене, популярни филми, рейтинги и други данни. Може да се използва с REST заявки и има добра документация за интеграция.
* **Appwrite -** open-source backend as a service (BAAS) платформа, която предлага готови решения за потребителска аутентикация, бази данни, файлово хранилище и други cloud функции. Той улеснява разработката на уеб и мобилни приложения, като намалява нуждата от ръчно писане на сложен backend код. Поддържа множество SDK-та за различни езици и frameworks, включително JavaScript, Flutter и React Native.

**Контекст:**

Контекстът в React/React Native е механизъм за споделяне на данни между компоненти без необходимост от пропърти-дрилване (props drilling). Той е полезен за глобални стойности като тема, потребителски данни, език и др. useContext е React Hook, който позволява лесен достъп до стойностите на контекста във функционални компоненти.

* **AuthContext.tsx / useAuth() -** използван за управление на състоянието на потребителя
* **SavedMoviesContext.tsx / useSavedMovies()** - използван за управление на запазените от потребител филми

**Динамично зареждане:**

* Използване на персонализиран React Hook (useFetch), който опростява извличането на данни от API или асинхронни операции.

**По-интересни примери от кода:**

## Services

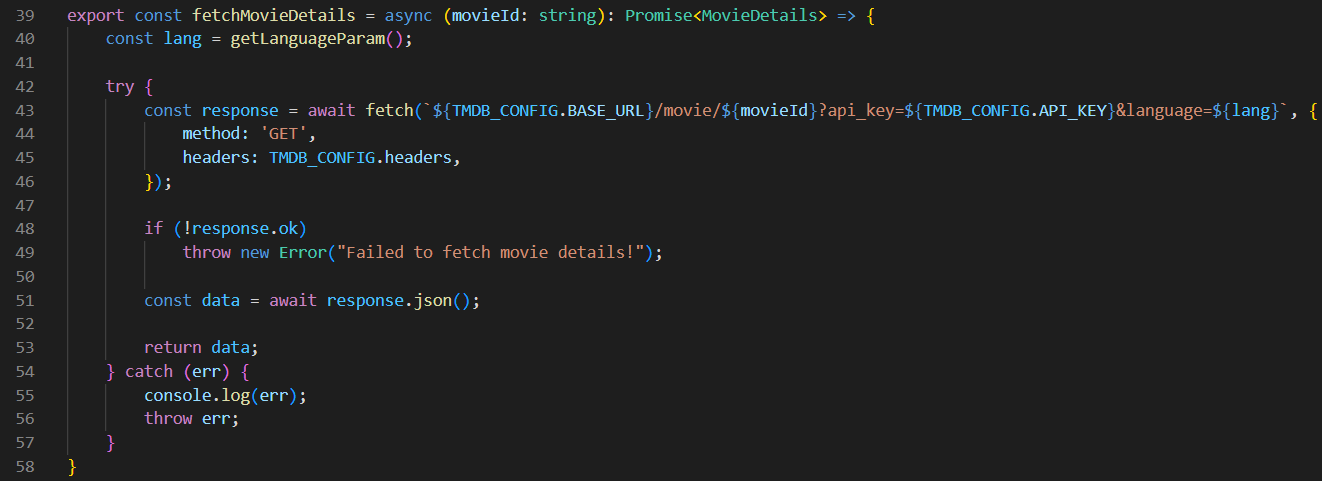
**api.ts:** Този код е част от услуга за достъп до външен API – The Movie Database (TMDB). Основната му функция е да извлича филми по зададен текст за търсене или да връща популярни филми по подразбиране.

**Съдържа:**

* **Конфигурация на API (ред 3 -10) -** конфигурирано веднъж с възможност за преизползване!
* **Функция за език (ред 12–14) -** взима текущия език от i18n библиотеката
* **Функция за извличане на филми (ред 17–36) - Параметър:** обект със свойство query, което е текстът за търсене. Ако query е зададен – се търсят филми с това име. Ако няма query, се извличат популярни филми. Избира подходящия краен URL за заявката: Ако има търсене (query) – използва **/search/movie**. Ако няма – използва **/discover/movie**, сортирани по популярност **(ред 20-22)**. След това се изпраща GET заявка към избрания endpoint **(ред 24-27)**. Следва обработка на грешки и провекра за статус кода на получения response **(ред 29-32)**. При проблем се хвърля грешка с тектс за грешка, а при успех се връщат извлечените от API филми.



Следващата функция е за извличане на детайлна информация за конкретен филм от TMDB API, използвайки ID на филма. Иползва се обработка за грешки (try - catch). Отново се изгражда и изпраща заявка към API, проверява се дали response е валиден и при успех се преобразува в JSON формат и се връща:



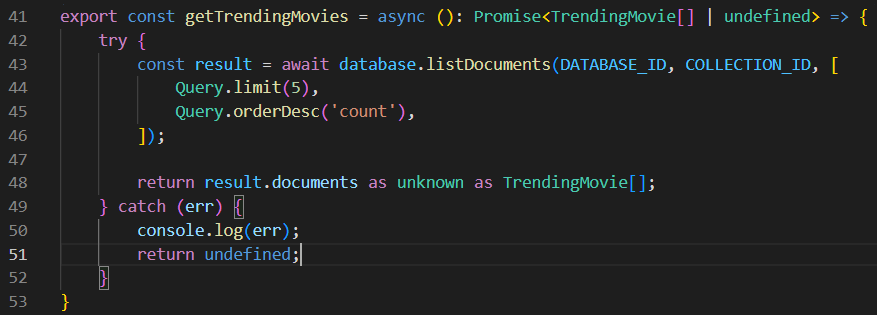
Останалите функции от този файл са със сходна идея - извлизане на български филми и след това функции за извличане на TV предавания. Идеята и кодът са сходни и затова няма да ги разглеждам по отделно.

* **appwrite.ts:** този файл съдържа функции за работа с Appwrite (backend платформа) в React Native приложение. Той използва Appwrite SDK за взаимодействие с база данни и система за потребителска аутентикация. Тук се инициализира appwrite client към текущия проект **(ред 7-9)** и се създават инстанции за достъп до базата и акаунтите **(ред 11-12)**. Извършват се следните функции:
* Запазва филми в "любими".
* Показва тенденции в търсенето.
* Управлява потребителски сесии.

**Функция: updateSearchCount:** записва или актуализира информация в база данни за това колко пъти даден филм е бил търсен по зададен низ (query). Проверява дали вече съществува запис за търсенето (query) в базата данни, като използва **Query.equal**. Ако записът съществува, увеличава броя търсения с едно, иначе създава нов запис



**Функция: getTrendingMovies:** Тази функция извлича списък с "трендови" (най-популярни) филми от базата данни на Appwrite. Използва се за показване на "Най-търсените филми днес" в началния екран.



**Функция saveMovie:** Запазва филм в колекцията "запазени филми", ако вече не е запазен. Проверява дали филмът вече е запазен чрез **checkIfMovieSaved(movie.id).** Ако не е запазен, създава нов документ в колекцията SAVED\_COLLECTION\_ID

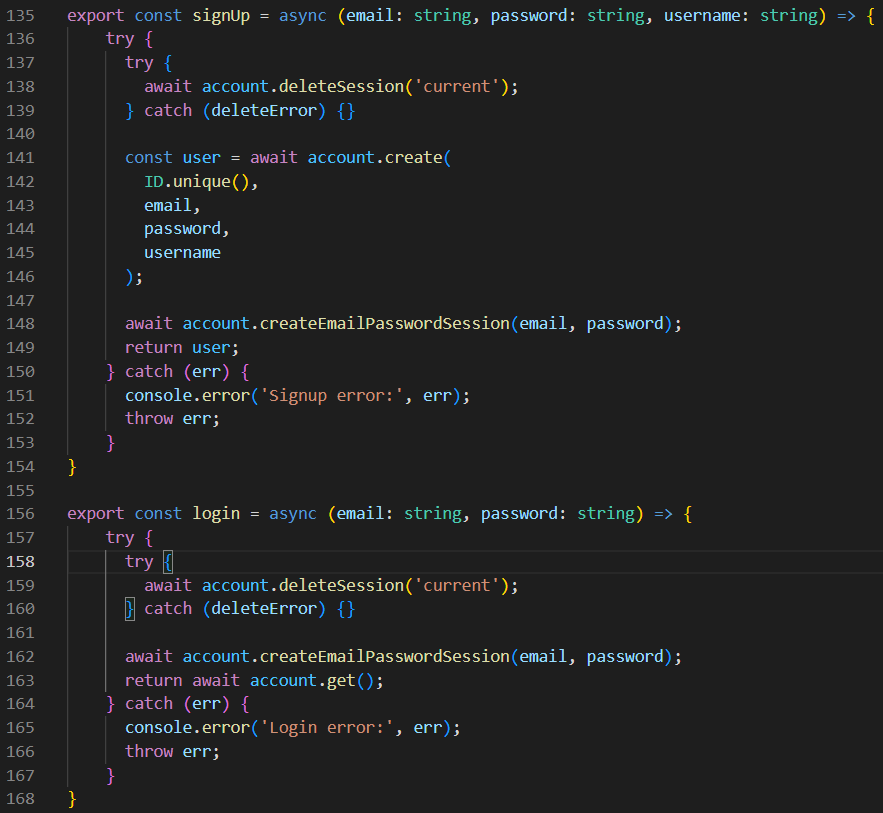
**Функция getSavedMovies:** Връща всички запазени филми от базата данни.



Следващите 2 функции са насочени към аутентикацията в приложението:

**Функция: signUp:** Създава нов потребител и го вписва автоматично (log in) след регистрация. В началото изтрива текущата сесия, ако има такава, за да се избегнат конфликти с вече активен потребител. След това създава нов потребител и автоматично го вписва:

**Функция: login:** Вписва съществуващ потребител в приложението. Отново изтрива текущата сесия, ако има такава, след това създава нова сесия за вписване, ако email и паролата са коректни. Накрая връща информация за текущия потребител:



и т.н.

## Hooks

* **useFetch.ts:** потребителски (custom) React hook, наречен useFetch. Служи за централизирано и многократно използване на логика за извличане на данни (fetch). Подобна логика често се повтаря, затова създаването на такъв хук прави кода по-четим и лесен за поддръжка.

**useFetch<T>** приема функция, която прави заявка към API (или друга асинхронна операция) и връща:

**data:** резултат от заявката

**loading:** дали се зарежда в момента

**error:** евентуална грешка при извличане

**refetch:** функция за повторно извикване

**reset:** функция за нулиране на състоянието

**Аргументи:**

**useState hooks:** Съхраняват данните, състоянието на зареждане и евентуална грешка.

**fetchData функцията:** Изпълнява fetchFunction, задава loading = true, записва резултатите или грешка.

**reset функцията:** Изчиства всичко – удобно при повторни заявки или смяна на страници.

**useEffect hook:** Автоматично стартира заявката при първоначално зареждане, ако autoFetch = true.

**Връща:**

return {

data, // резултат от заявката

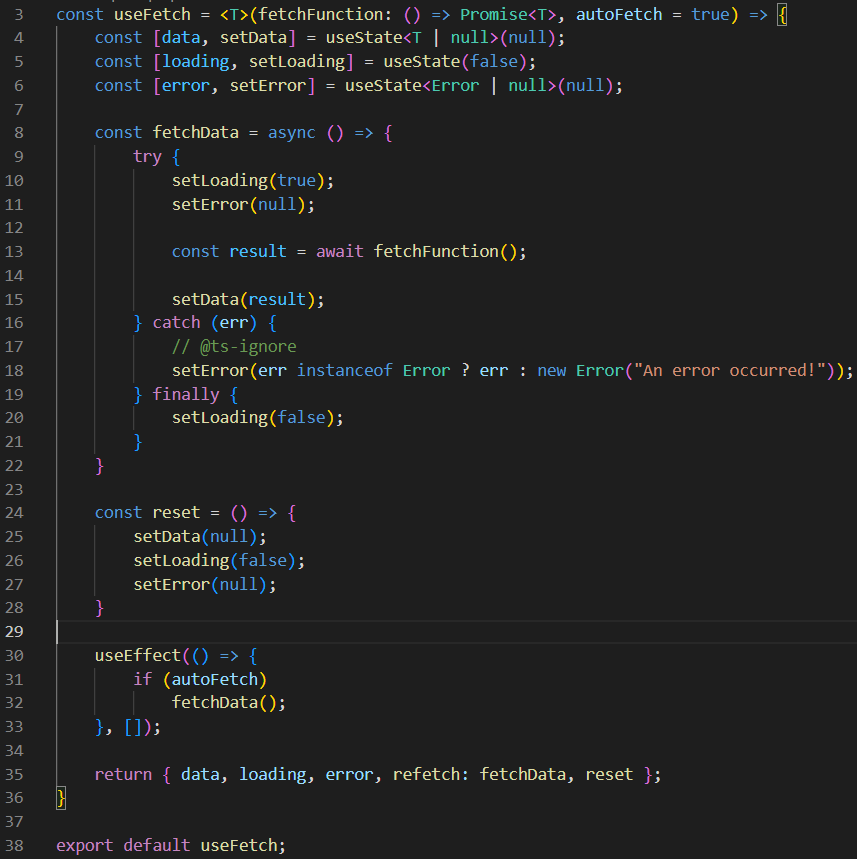
loading, // дали е в процес на зареждане

error, // ако има грешка

refetch: fetchData, // функция за презареждане

reset // функция за нулиране

};



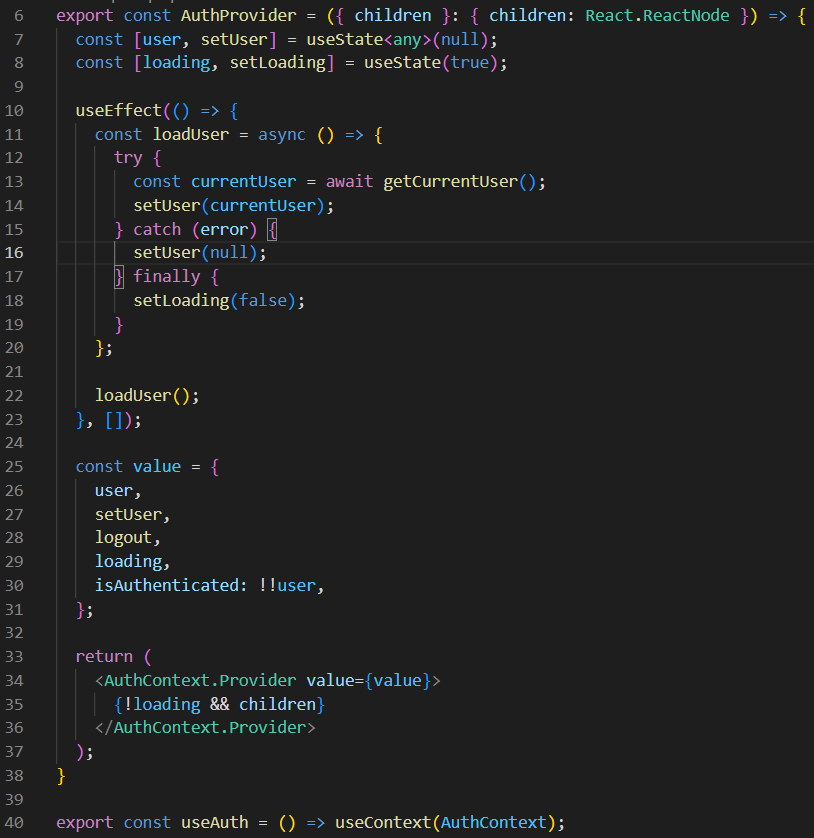
* **useModal.ts:** Този код дефинира потребителски React hook (custom hook), наречен useModal, който предоставя централизирана логика за управление на модални прозорци (modal windows) в потребителския интерфейс. Той управлява:
* дали модалът е видим
* заглавие на модала
* съобщение
* тип (икона/цвят) на модала – например info, success, warning, error



## Context

Context API позволява споделяне на стойности между компоненти без нужда от пропсинг надолу по дървото. Използва се за:

* удостоверяване (auth)
* настройки (theme, език)
* глобални данни (напр. филми, потребител)
* **AuthContext.tsx** – Контекст за удостоверяване -> Държи информация за текущия потребител, статуса на зареждане и дали е автентикиран. При зареждане на приложението, проверява дали има влязъл потребител, като използва getCurrentUser() от Appwrite API. Ако има такъв – записва го в user, иначе – null. Подава контекста и децата на този провайдър имат достъп до всички стойности чрез **useAuth() hook.**



* **SavedMoviesContext.tsx** – Контекст за запазени филми -> Управлява списък със запазени филми на потребителя. Използва useFetch хук, за да зареди филмите от getSavedMovies. Съхранява:
* savedMovies – масив с филми
* loading - зареждане
* error – грешка
* refreshSavedMovies – функция за повторно зареждане



## App:

Това е мястото, където се структурира логиката на маршрутизацията (routing), страниците (pages) и сървърни/клиентски компоненти.

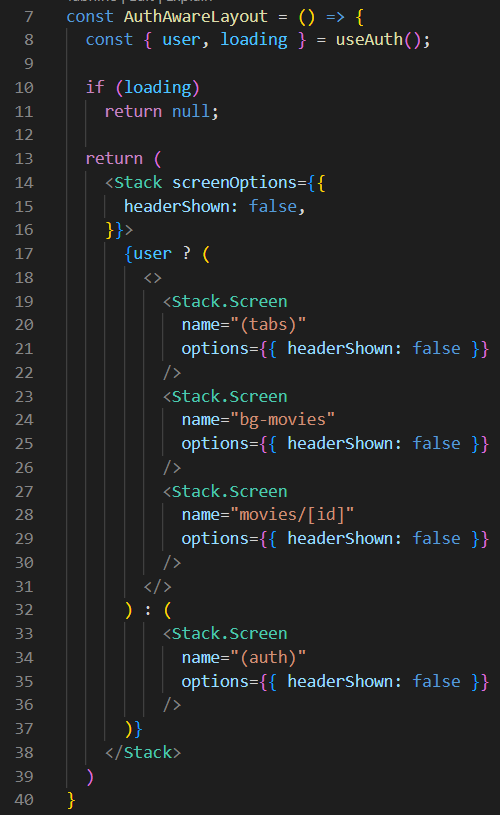
* **\_layout.tsx**: Това е основен шаблон за всички страници. Използва се като "рамка", в която се вграждат различните page.tsx.
* **(auth)/**: сегмент от маршрути, свързани с потребителската автентикация. Скобите () в Next.js и Expo Router означават "групиране без URL сегмент", т.е. те:
* групират файлове и папки логически, но
* не влияят на пътя (route), който се вижда в браузъра или приложението.

Тук се намират **index.tsx, login.tsx, signup.tsx -> От тук се управляват:**

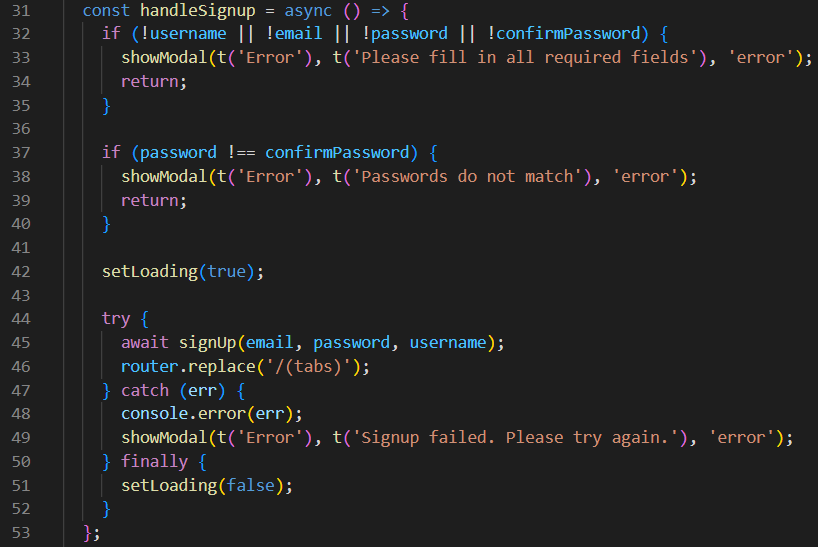
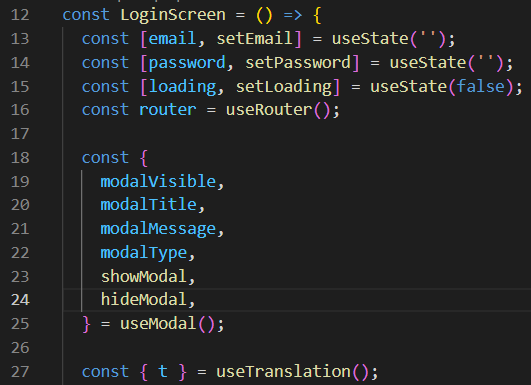
* **Регистрация** – потребител въвежда email, парола и име.
* **Вход** – създаване на сесия и извличане на текущ потребител
* Преминаване към основното приложение след логин
* **(tabs)/:** съдържа всички екрани, които се показват като част от таб-базирана навигация (табове долу в приложението). Това е част от т.нар. layout-based routing. Позволява на потребителя да преминава между различни екрани като „Home“, „Search“, „Saved“ и „Profile“.

Тук се намират **\_layout.tsx, index.tsx, profile.tsx, saved.tsx, search.tsx** (Описани са малко по-надолу):

\_layout.tsx:



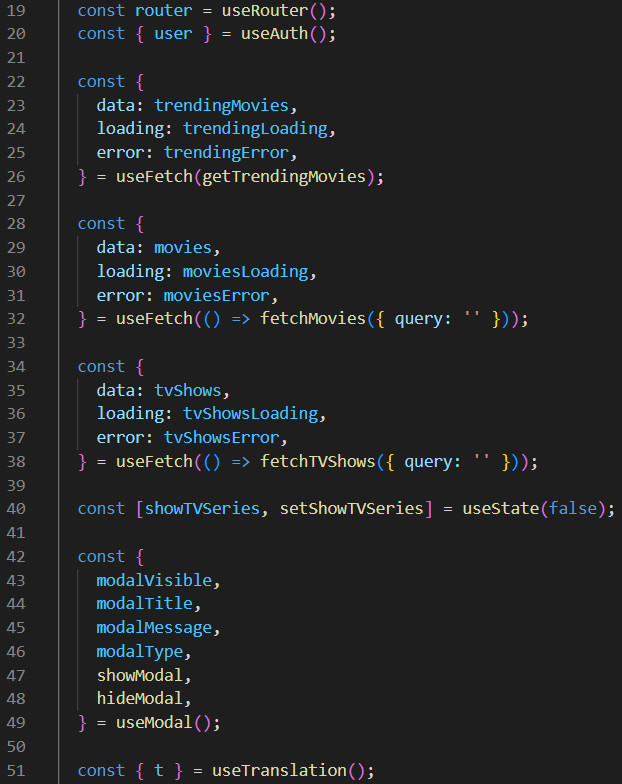
(auth)/login.tsx: (auth)/signup.tsx:



(tabs)/\_layout.tsx:

Index.tsx -> по-интересни чсти от кода:



Останалите tab страници са с идентична логика и функционалност спрямо конкретната нужда, затова няма да ги разглеждам по отделно.

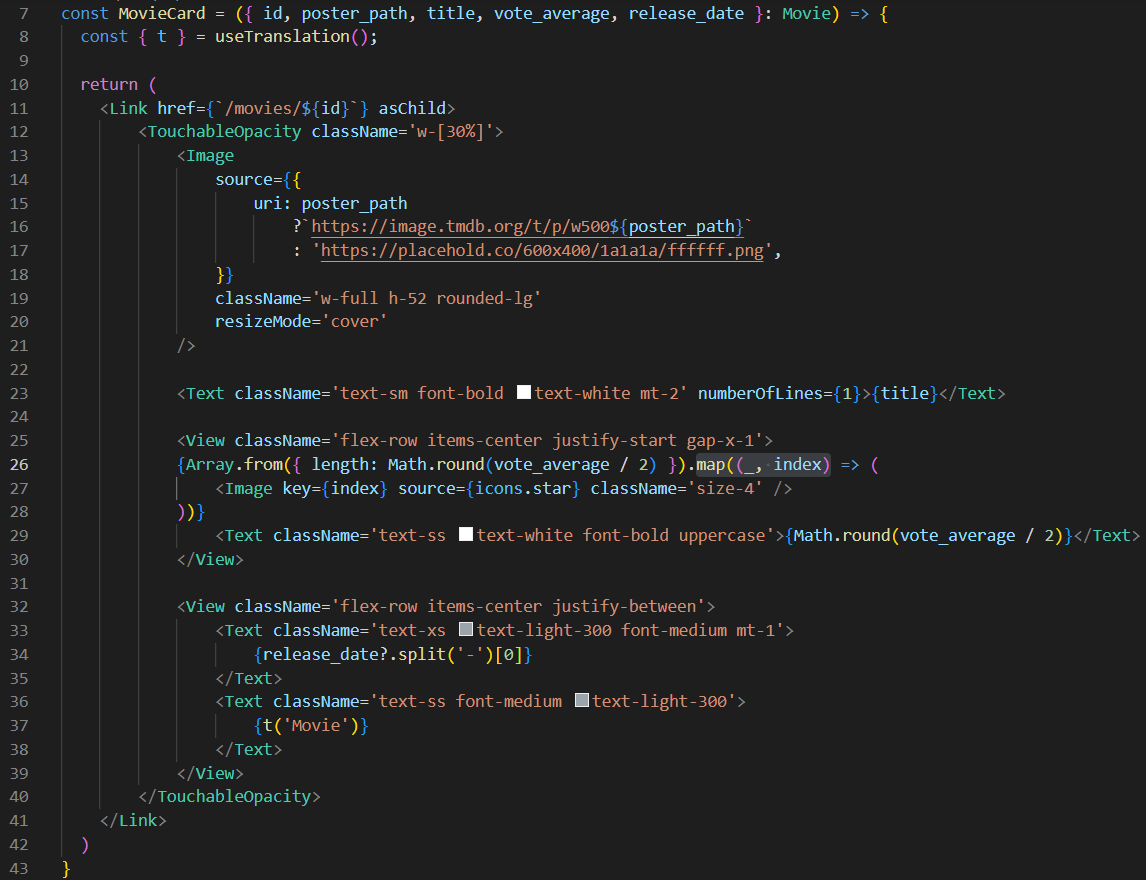
* **movies/[id].tsx:** представлява детайлната страница за даден филм. Тя се показва, когато потребителят избере конкретен филм от списъка.
* **useLocalSearchParams()** извлича id от URL-а (пример: /movies/1234 → id = 1234).
* **useFetch()** извиква API-то за филма.
* **useAuth()** проверява дали потребителят е логнат.
* **useModal()** управлява състоянието на модал прозорци.



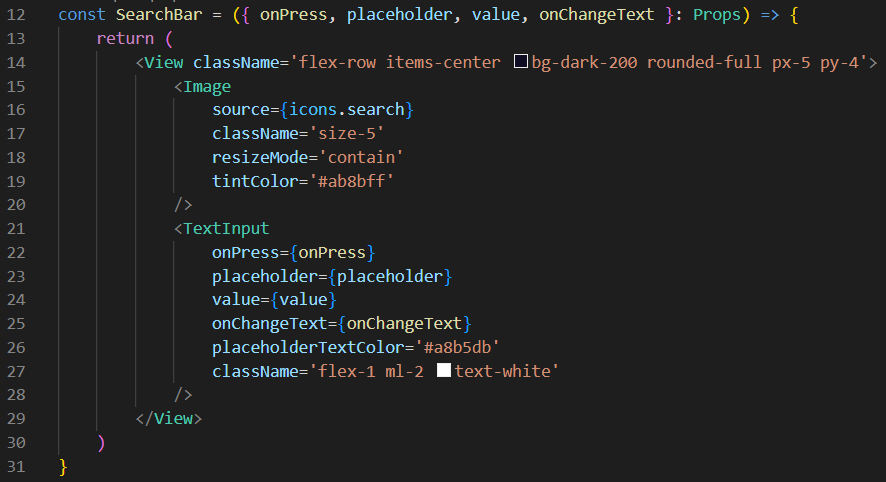
## Components

Това са модули от код, които представляват части от потребителския интерфейс (UI). Всеки компонент може да бъде независимо дефиниран и използван многократно. Позволяват:

* Повторно използване
* По-добра структура и четимост на кода
* По-лесна поддръжка
* Изолация на логика (например: модал, таб, карта, форма и т.н.)
* **MovieCard.tsx: Отделен компонент, котйо представлява елемента, визуализиращ картичката с филма на различни места от приложението.**



* **SearchBar.tsx: Търсачката в приложението.**



Това са по-интересните части от кода на приложението. Останалото ще се обясни на място при защитата в случай на възникнали въпроси.

* Appwrite backend:

В конкретното приложение се използва за следните функционалности:

* **Аутентикация (Authentication)**: Позволява на потребителите да:

- Се регистрират (signUp функцията)

- Влизат в профила си (login)

- Излизат от профила си (logout)

- Получават текущия си профил (getCurrentUser)

Използва се account обект от Appwrite SDK, за да се създават и управляват сесии на потребителите.

* **Запазване на филми (Database):** Потребителите могат да запазват филми в свой списък чрез:

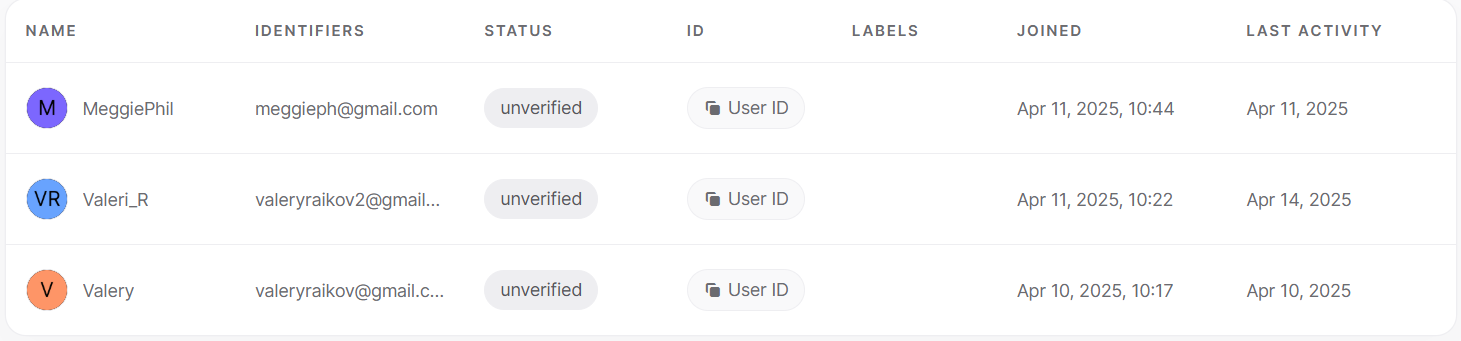
- saveMovie – записва филм в Appwrite база данни

- getSavedMovies – взема списък със запазени филми

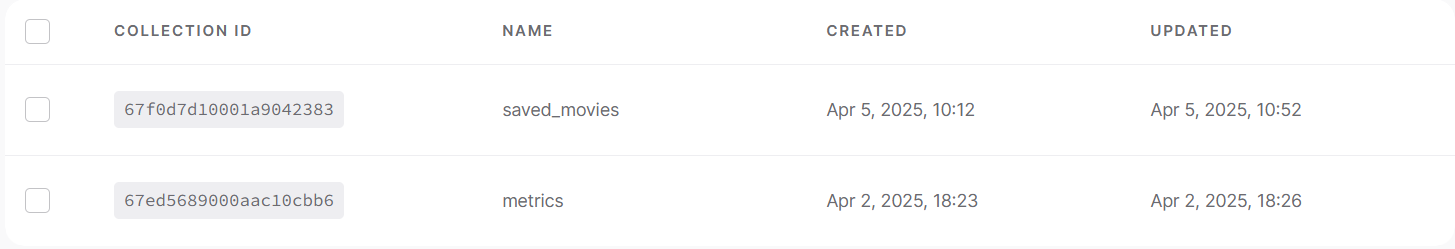
- checkIfMovieSaved – проверява дали даден филм вече е запазен

Това става чрез използване на базата данни (Database service) на Appwrite – ти създаваш записи (документи) в колекция, където съхраняваш информация за всеки филм и кой потребител го е запазил.

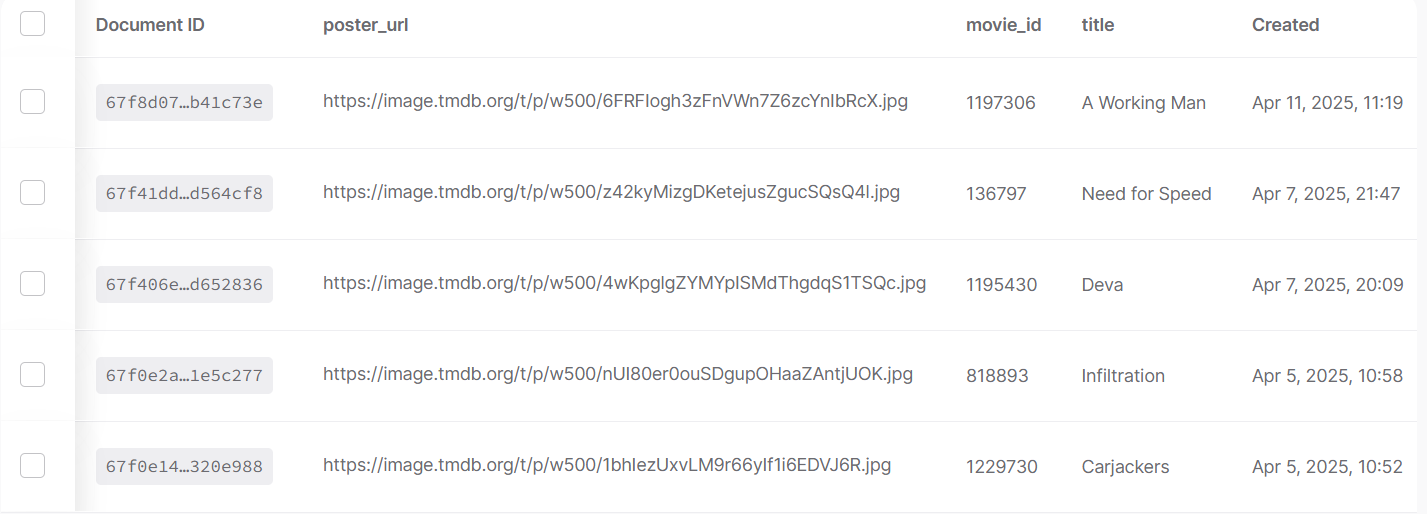
* **Appwrite/auth** - потребителите, регистрирани (влезли) в приложението:



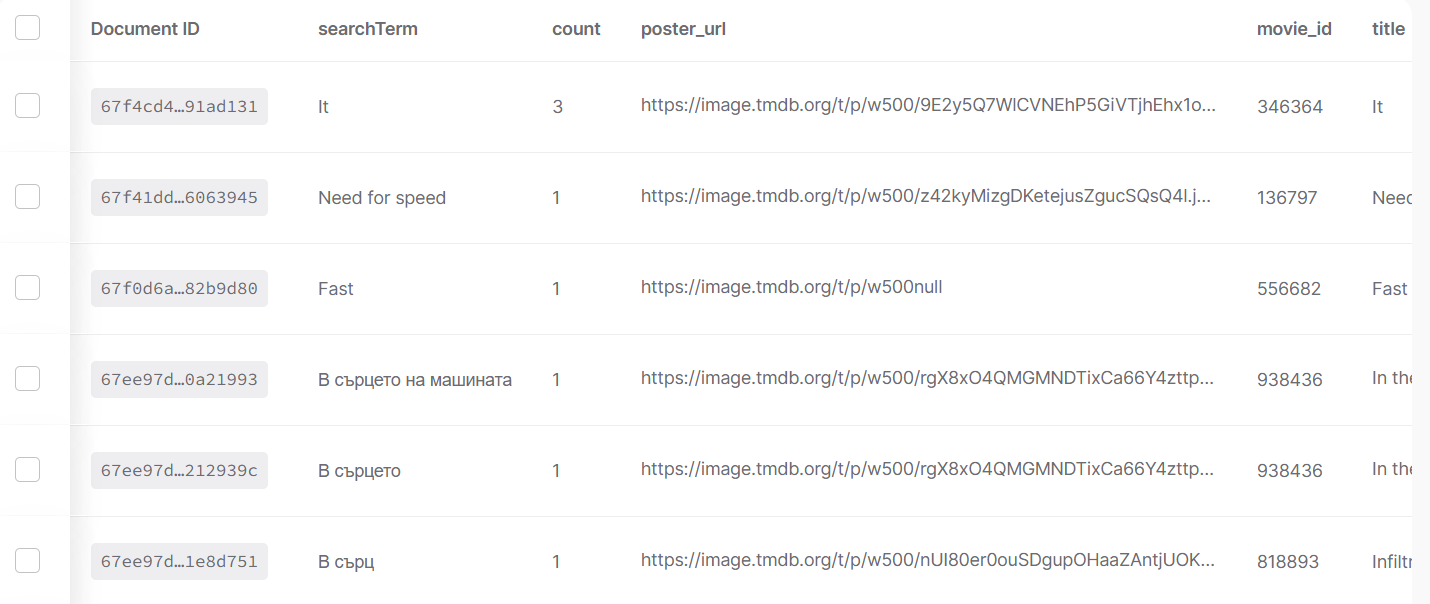
* Създадените колекции:



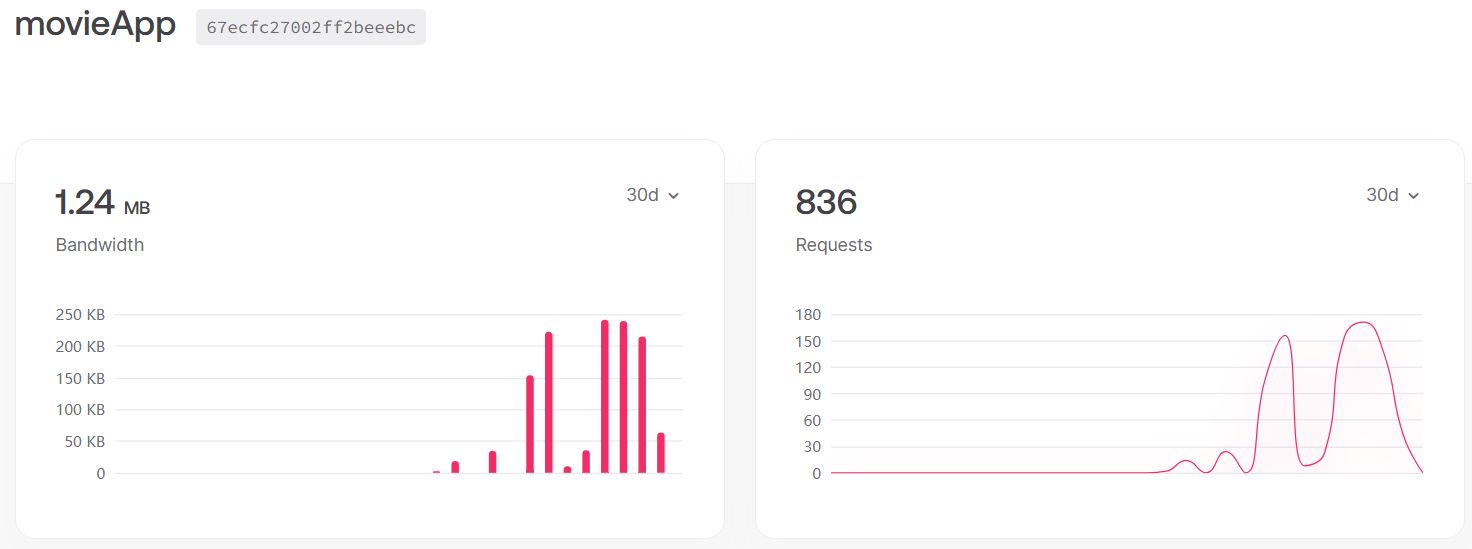
* saved\_movies колекция:

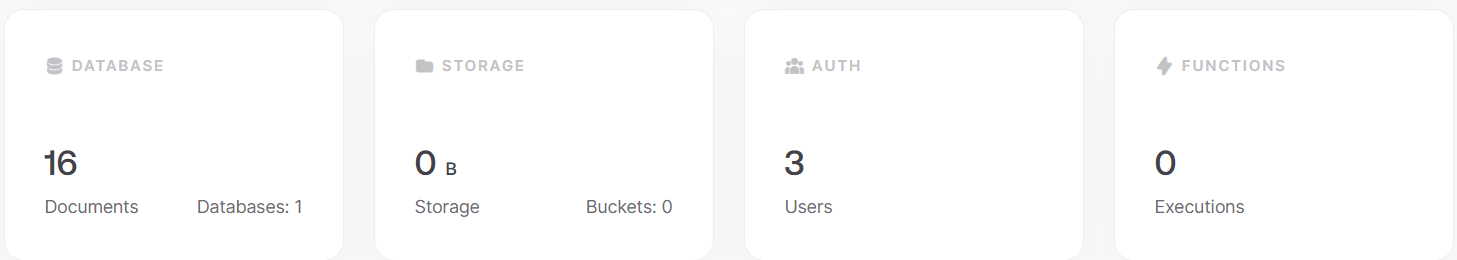


* metrics колекция - показва колко пъти даден филм е търсен от потребителите и се използва за извличане на набиращите популярност филми (те се определят именно от броя търсения за всеки филм!)



* Workflow диаграми и анализ на данните за зявки към сървиса:



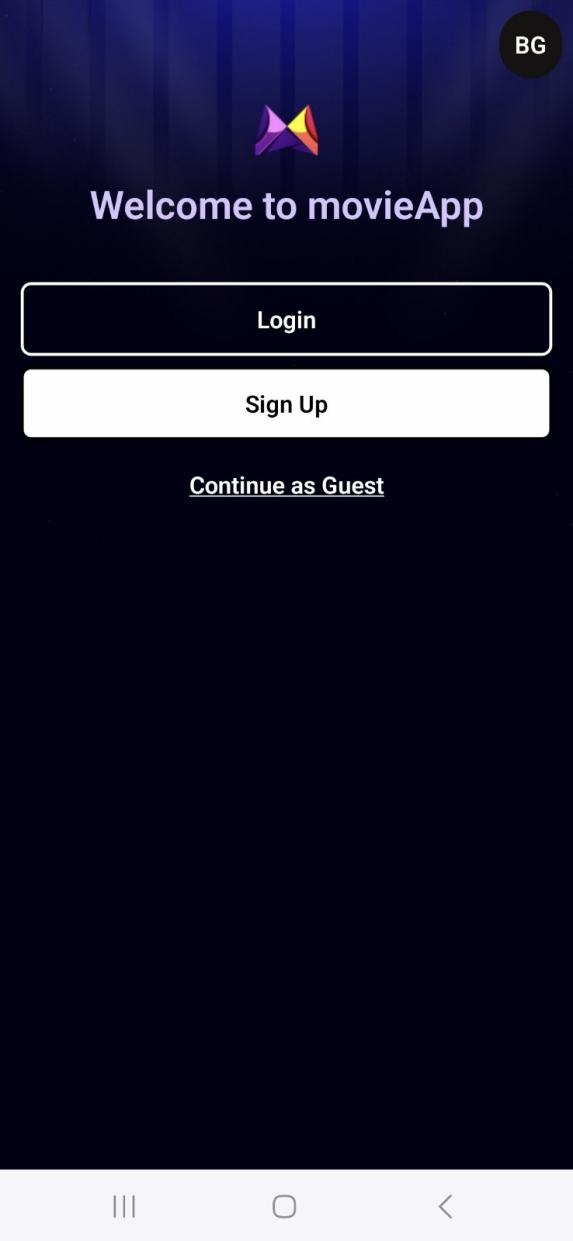
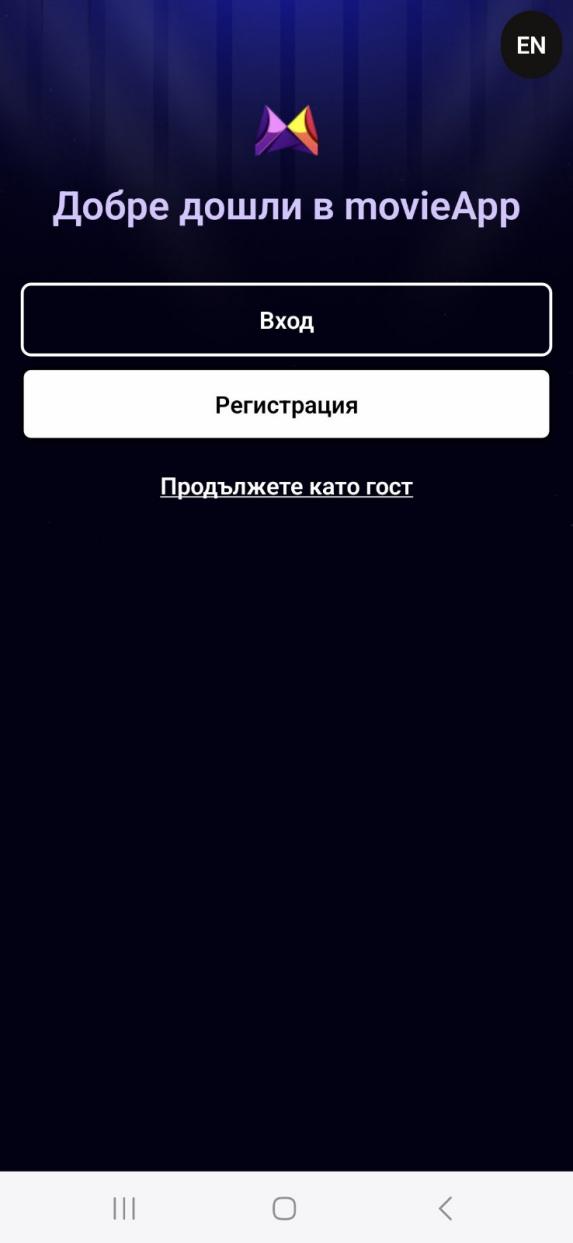


# Потребителско ръководство

В тази секция се разглежда самото приложение и как Вие потребителите да го ползвате правилно.

## Потребителят отваря приложението:

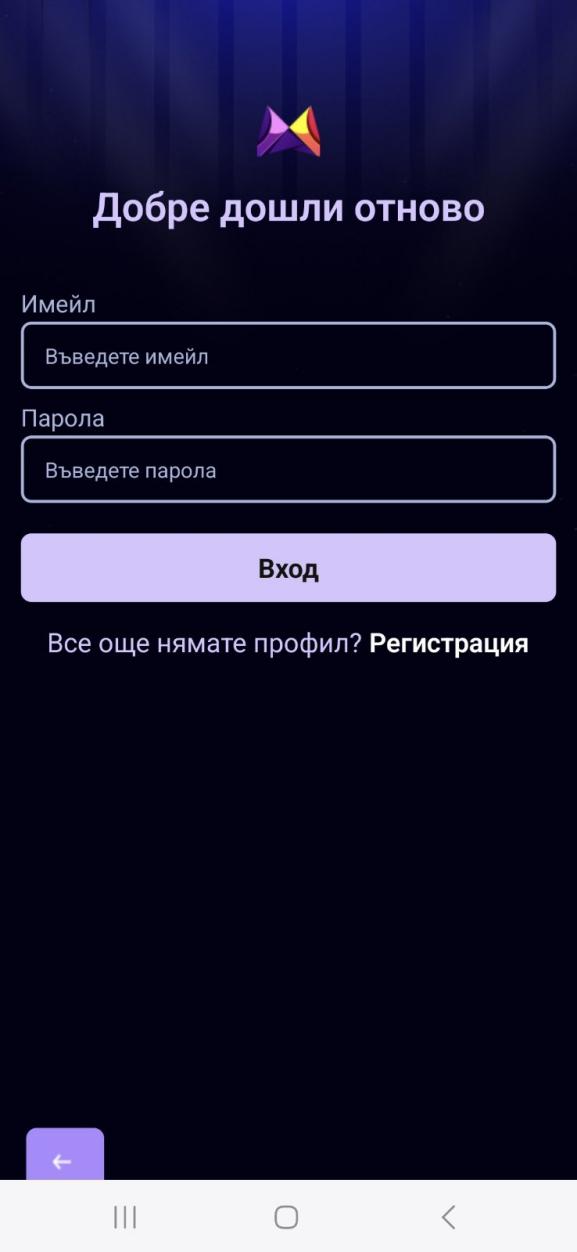
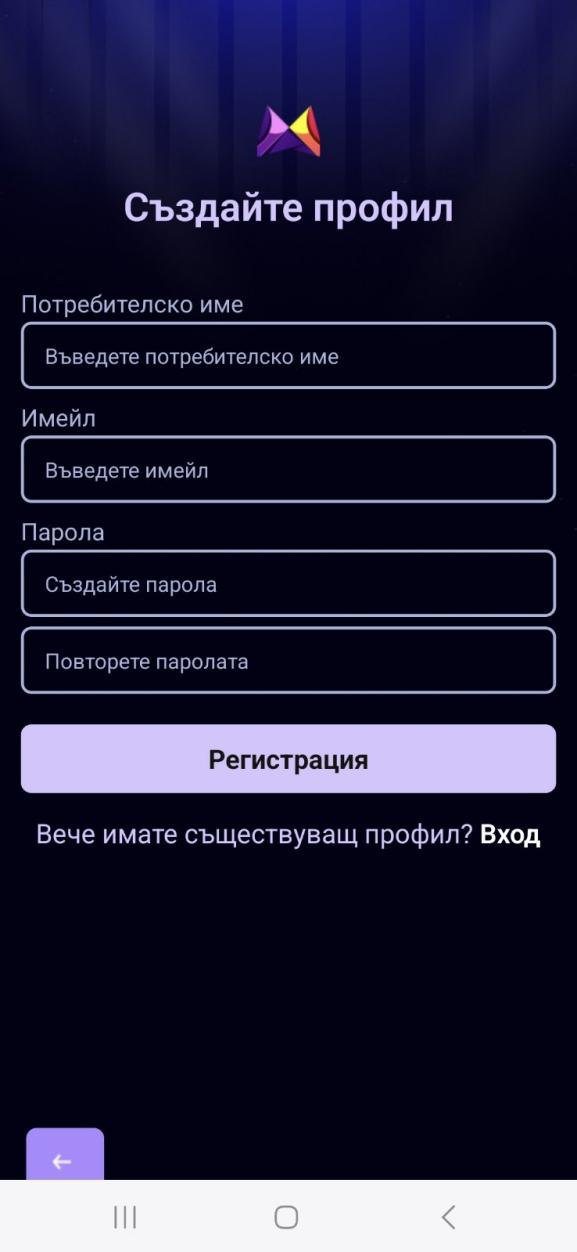
При първоначално стартиране на приложението потребителят вижда следния екран: По подразбиране езикът на приложението е настроен на англисйски, но за наше улеснение ще го сменим на български (след като веднъж изберем езика, цялото приложение се превежда автоматично):

**Бутон за смяна на езика**

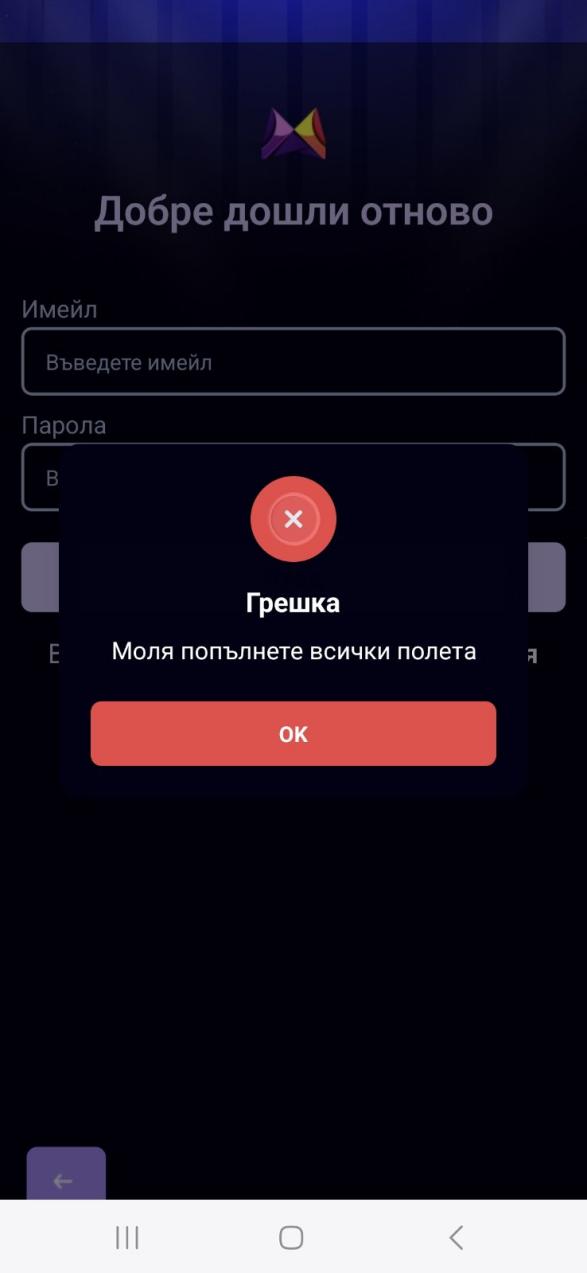
Потребителят има следните 3 възможности, за да продължи напред :

* Вход (за вече съществуващ профил)
* Регистрация (създаване на нов профил)
* Продължете като гост (използване на приложението без регистрация, но с ограничени възможности):

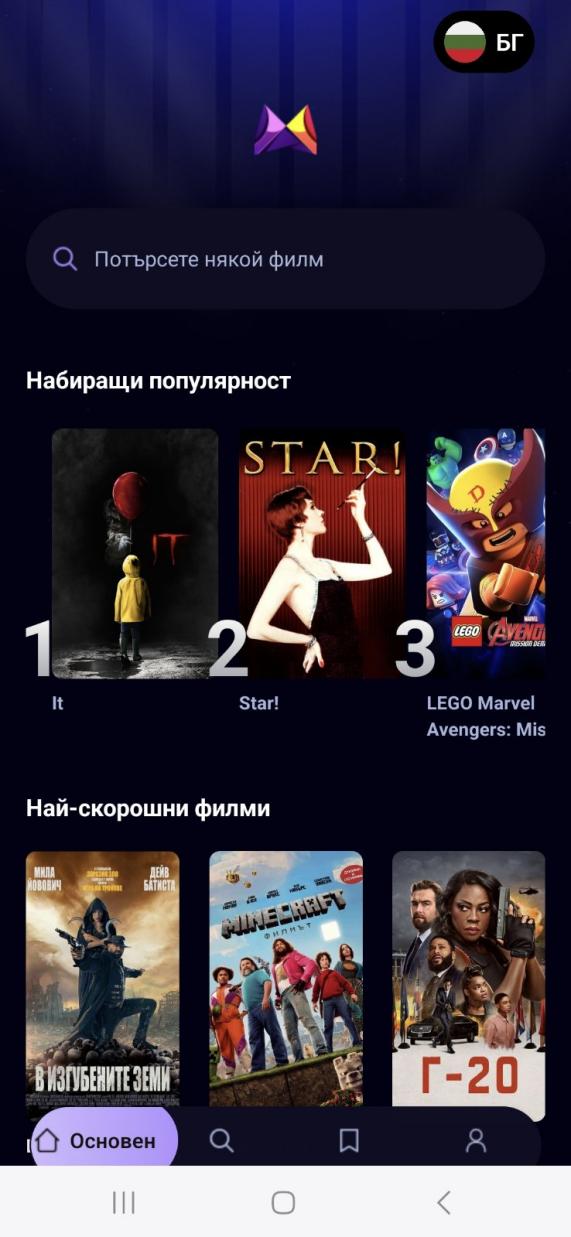
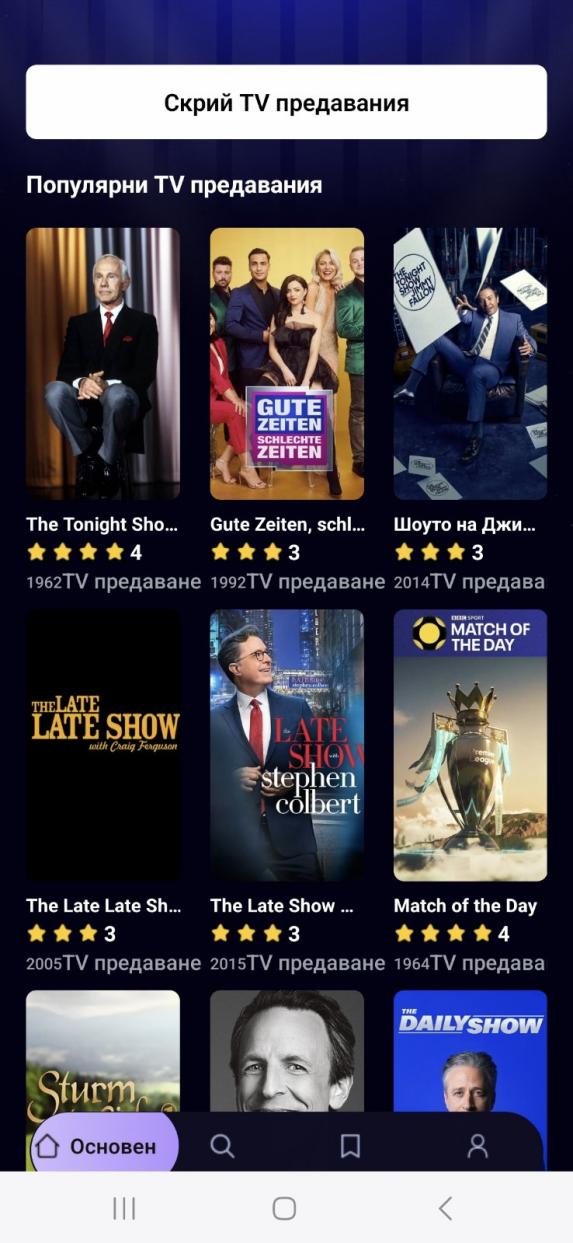
**Бутон за връщане назад**

При въвеждане на грешни данни (невалиден имейл или парола, несъответсвие с данните, различие в паролите, непопълнени полета и т.н) се показва специално предназначения за грешки модал. Пример:

-

Приемаме, че потребителят е решил да достъпи приложението като гост (без вход в профил):

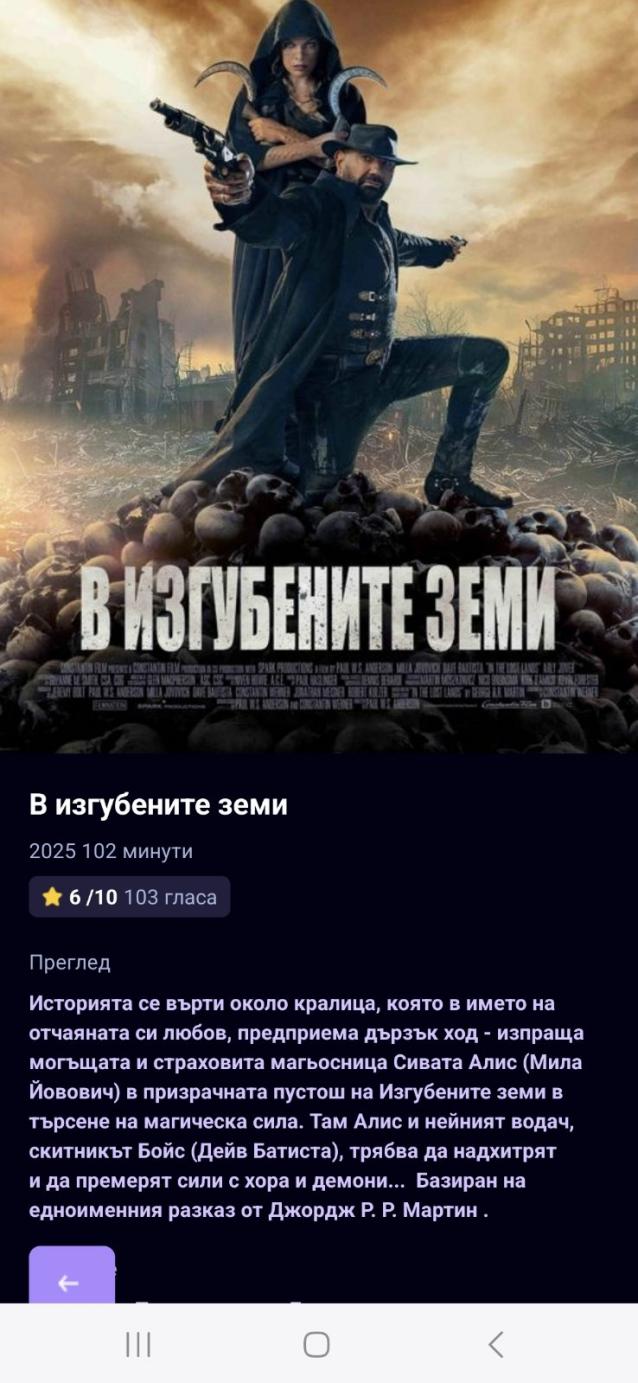
* Основен екран - потребителят разполага с бутон за достъп до страницата с български филми и предавания, търсачка, набиращи популярност филми, последно излезли филми и възможност за показване / скриване на TV предавания

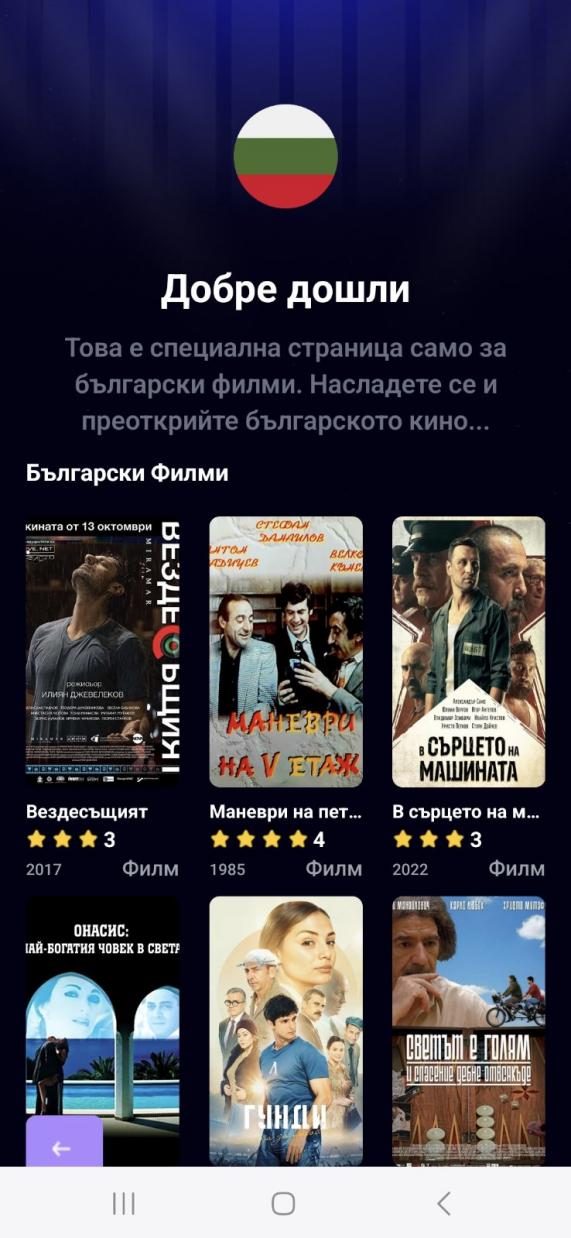
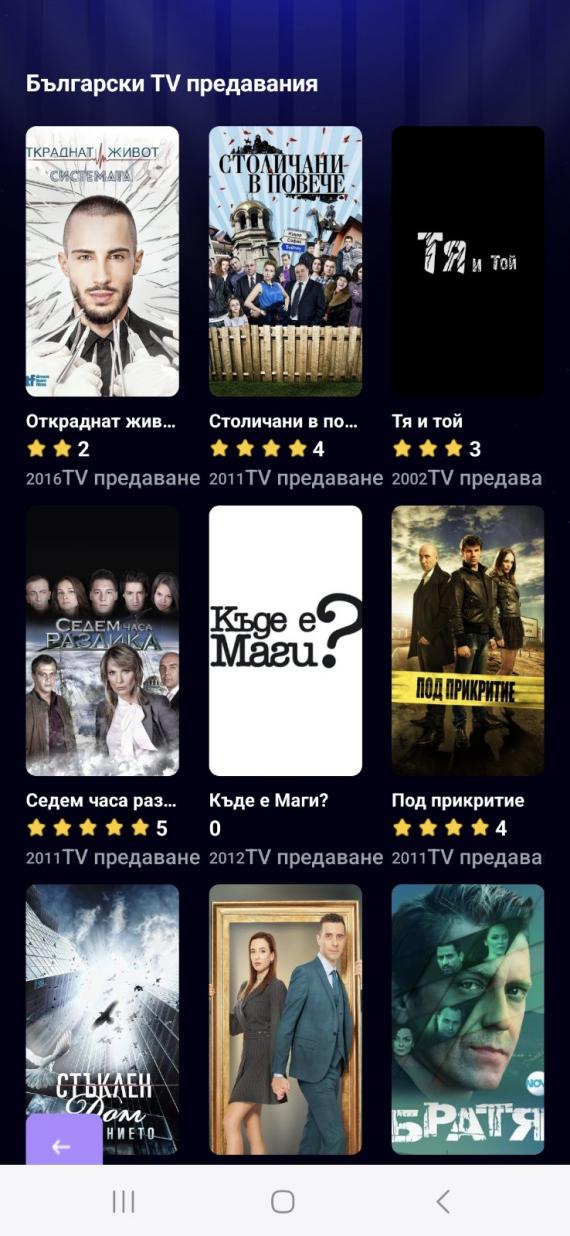
**Бутон за достъп до БГ кино**

**Бутон за показване / скриване на TV предавания**

При натискане на даден филм / предаване, гост потребителят вижда специална страница само за конкретния филм / конкретно предаване, на която се показва цялата необходима информация достъпна от външния API сървис (title, release date, runtime, vote\_average, vote\_count, overview, genres, budget, revenue, production companies и т.н):



Страницата, която е отделена само и единствено за българско кино, изглежда по следния начин:

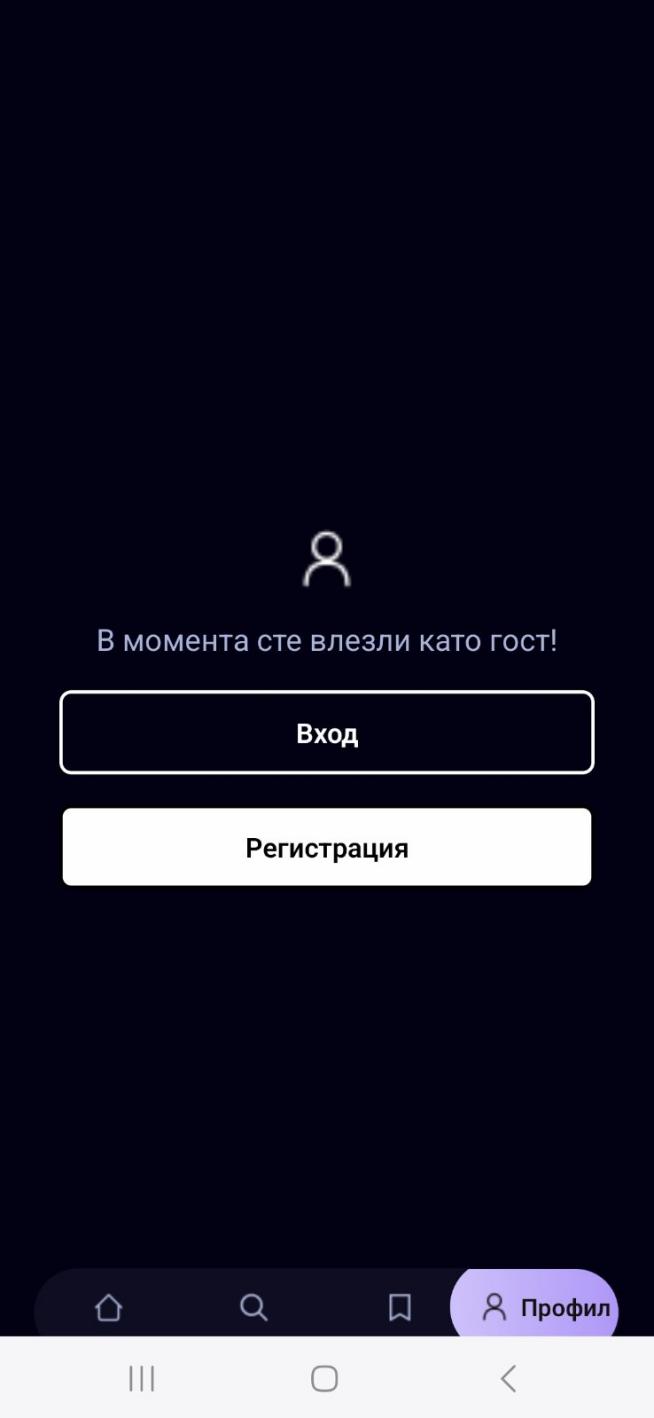
Следващия екран, който е достъпен чрез навигацията в дъното на екрана, е екрана с търсачката, където потребителите имат да възможност да търсят филми по техен избор:



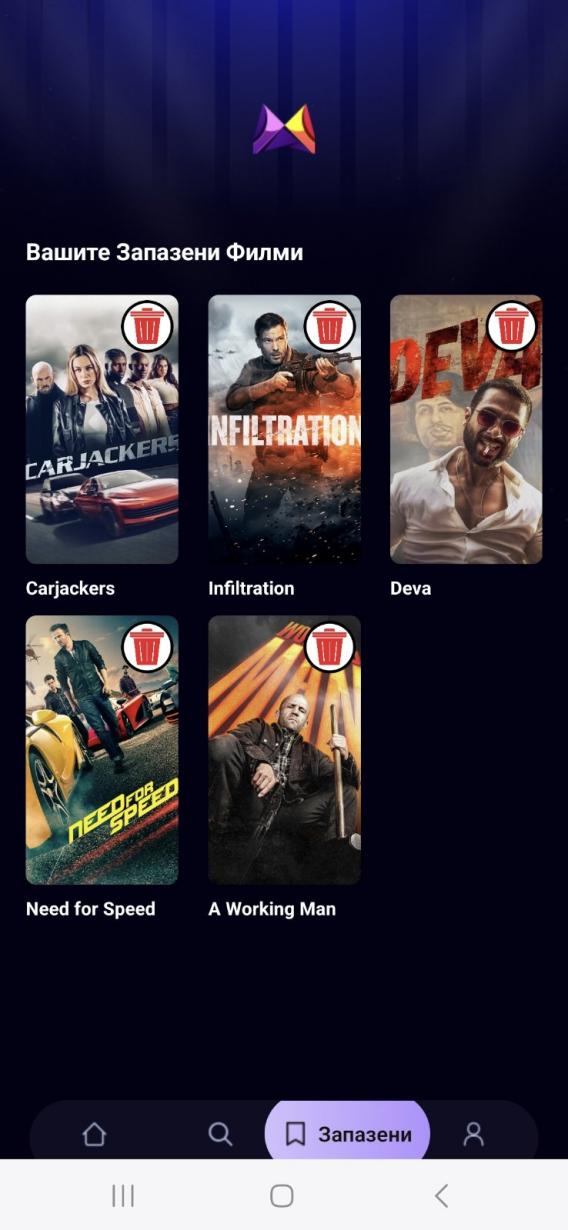
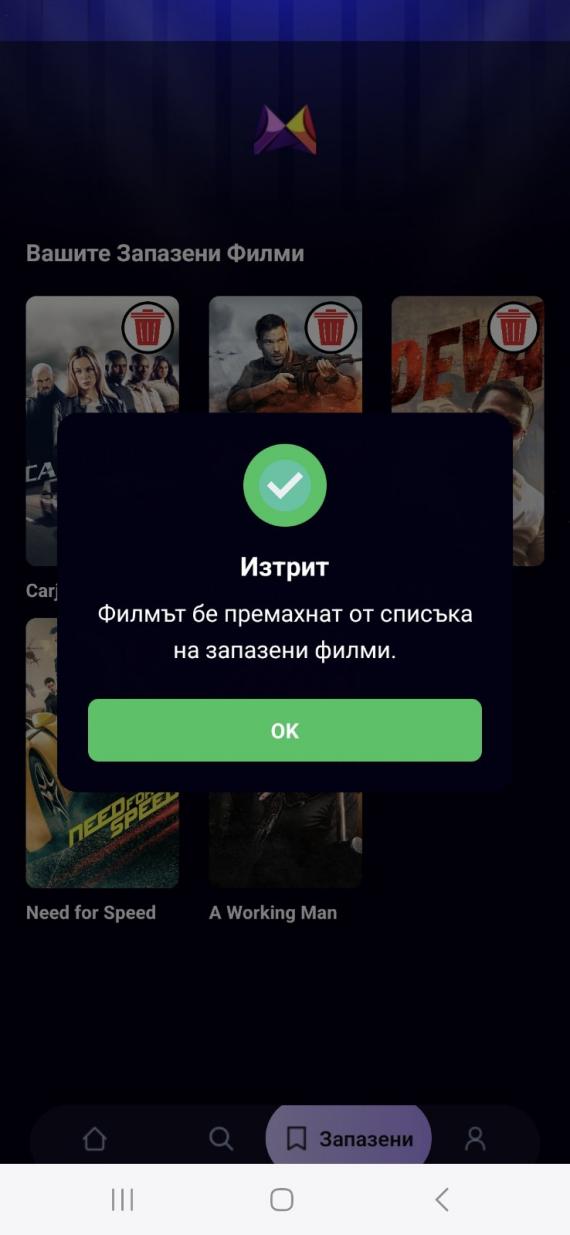
Следващия екран е този със запазените от потребителя филми (В момента той е неактивен / нефункционален, тъй като сме влезли като гости):



Последния таб от навигацията е потребителския профил, с който потребителя е влезнал. Тъй като в момента сме влезли като гости, този екранизглежда така:



Сега да разгледаме функциоалността на приложението при логнат потребител. След успешен вход, потребителят има достъп до страницата със запазените от него филми:

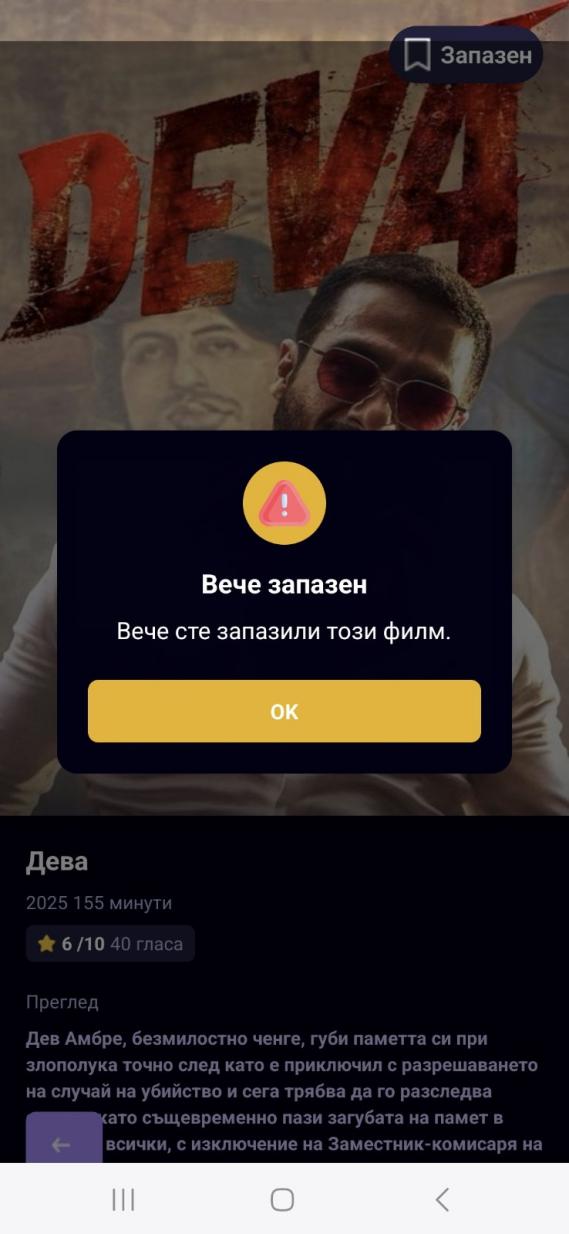
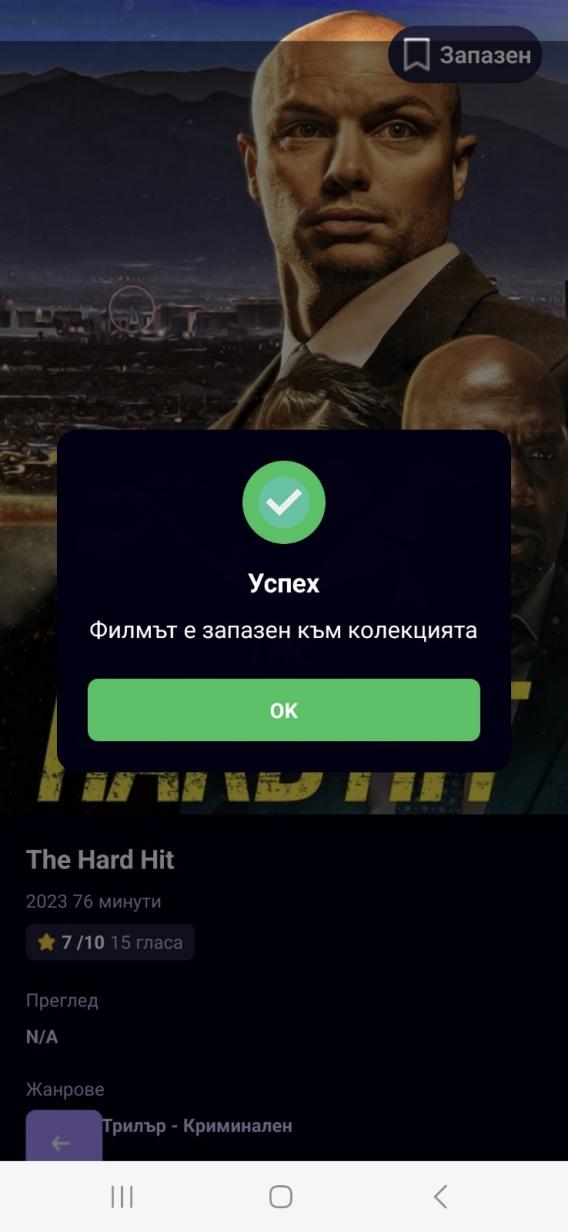
**Бутон за премахване на запазен филм**

Освен това, когато логнат потребител досъпи страницата с детайлите за конкретен филм, има възможност да го запази в своя списък на запазени:

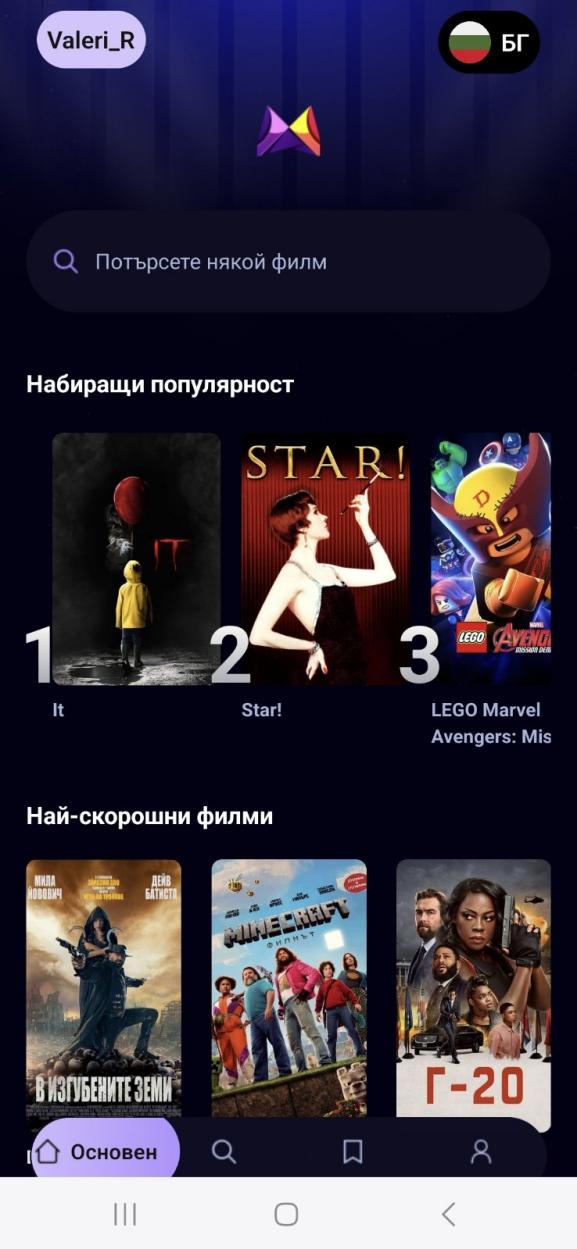
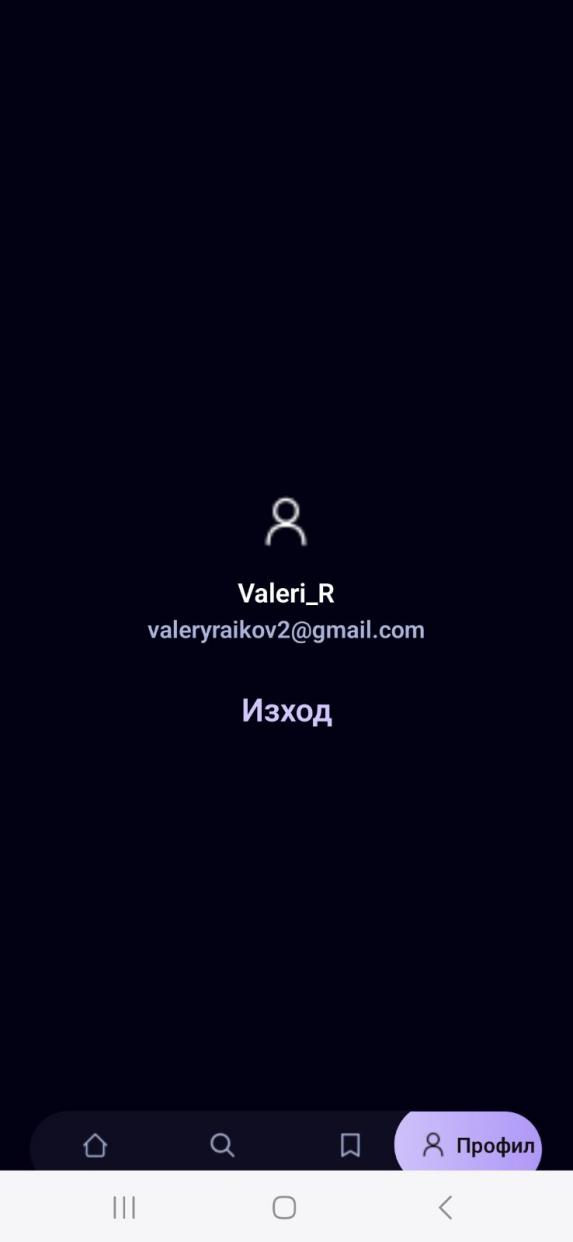


**Бутон за запазване на филм**

Ако филмът е вече запазен (както е в случая) получаваме съобщение, че филмът е запазен, а ако не е запаен, съобщение за успех и филмът се добавя към списъка съз запазени:

Останалите разлики в приложението след успешен вход или регистрация на поребител са: Бутон, индикиращ успешен вход, който при натискане препраща към “Профил” страницата, която показва детайлите на влезлия потребител:

**Бутон за изход от профил**

**Бутон за влезнал потребител**

1. Заключение

Разработеното приложение изпълнява всички основни функционалности, които са заложени в проекта, като предоставя надеждно и стабилно потребителско изживяване. Интерфейсът е достатъчно атрактивен, но същевременно удобен и лесен за използване от потребителя. За допълнително улеснение и по-добро потребителско преживяване, приложението поддържа 2 езика - български и английски. Връзката с външните API услуги е стабилна и всички потенциални грешки и проблеми, които биха могли да се появят, са прихванати и съответните грешки се показват в персонализиран модал за грешки. Продуктът е подходящ за реална употреба и може да бъде разширен с допълнителни функции като любими, рейтинги, списъци за гледане и други.

# Литература

* **React Native Official Docs** – <https://reactnative.dev>
* **Expo Documentation** – <https://docs.expo.dev>
* **TypeScript Documentation** - <https://www.typescriptlang.org/docs/>
* **Tailwind CSS** – <https://tailwindcss.com>
* **TMDb API Documentation -** <https://developer.themoviedb.org/reference/intro/getting-started>
* **Appwrite Docs** – <https://appwrite.io/docs>
* **i18next Documentation -** <https://www.i18next.com/>
* **YouTube tutorials** - <https://www.youtube.com/watch?v=f8Z9JyB2EIE,> <https://www.youtube.com/watch?v=sm5Y7Vtuihg,> <https://www.youtube.com/watch?v=YSUmzHH_OMg,> <https://www.youtube.com/watch?v=_JDeJgsU-bI,> <https://www.youtube.com/watch?v=JUU_XzdbzN0>, <https://www.youtube.com/watch?v=r88z8nrk8Ww,> <https://www.youtube.com/watch?v=DTQ7EceGACM>
* **Stack Overflow**
* **AI tools**

**Можете да достъпите проекта в моя GitHub профил:**

<https://github.com/ValeryRaikov/react-native-projects/tree/main/movieApp>