|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | |  | **Wydział Informatyki i Zarządzania**  kierunek studiów: informatyka  Praca dyplomowa – inżynierska  **Aplikacja do spersonalizowanej selekcji programów telewizyjnych**  Przemysław Zender  słowa kluczowe:  1 linia  2 linia  3 linia  krótkie streszczenie:  1 linia  2 linia  3 linia  4 linia  5 linia  6 linia   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | opiekun pracy  dyplomowej | .................................................. | | ....................... | | ....................... | | | *Tytuł/stopień naukowy/imię i nazwisko* | | *ocena* | | *podpis* | | | Ostateczna ocena za pracę dyplomową | | | | | | | | Przewodniczący Komisji egzaminu dyplomowego | | ..................................................  *Tytuł/stopień naukowy/imię i nazwisko* | | ....................... | | ....................... | | *ocena* | | *podpis* |   *Do celów archiwalnych pracę dyplomową zakwalifikowano do:\**   1. *kategorii A (akta wieczyste)* 2. *kategorii BE 50 (po 50 latach podlegające ekspertyzie)*   *\* niepotrzebne skreślić*   |  | | --- | | pieczątka wydziałowa | |
|  |  |  | Wrocław  2018 |

Spis treści

[1. Wstęp 3](#_Toc15848057)

[1.1. Uzasadnienie tematu 3](#_Toc15848058)

[1.2. Cel pracy 3](#_Toc15848059)

[1.3. Zakres prac 3](#_Toc15848060)

[2. Przegląd istniejących rozwiązań 5](#_Toc15848061)

[2.1. Standardoweprzewodniki programowe 5](#_Toc15848062)

[2.2. Silniki rekomendacji 6](#_Toc15848063)

[2.3. Serwisy streamingowe 7](#_Toc15848064)

[2.4. Filmweb 8](#_Toc15848065)

[2.5. Tabela porównawcza 9](#_Toc15848066)

[2.6. Wnioski 10](#_Toc15848067)

[3. Założenia projektowe 11](#_Toc15848068)

[3.1. Opis problemu 11](#_Toc15848069)

[3.2. Wizja rozwiązania 11](#_Toc15848070)

[3.3. Wymagania funkcjonalne 14](#_Toc15848071)

[4. Projekt aplikacji 15](#_Toc15848072)

[4.1. Baza danych 15](#_Toc15848073)

[4.2. Prototyp interfejsu 16](#_Toc15848074)

[4.3. Architektura aplikacji 22](#_Toc15848075)

[5. Implementacja 23](#_Toc15848076)

[5.1. Baza danych 23](#_Toc15848077)

[5.2. Serwer 24](#_Toc15848078)

[5.3. Klient 31](#_Toc15848079)

[6. Testy 35](#_Toc15848080)

[6.1. Jednostkowe serwera – xUnit 35](#_Toc15848081)

[6.2. Jednostkowe klienta – Jasmine 35](#_Toc15848082)

[6.3. Automatyczne end-to-end – Protractor 35](#_Toc15848083)

[7. Instalacja 36](#_Toc15848084)

[7.1. Założenia 36](#_Toc15848085)

[7.2. Zastosowanie 37](#_Toc15848086)

[7.3. Komunikacja 39](#_Toc15848087)

[8. Podsumowanie 40](#_Toc15848088)

[9. Bibliografia 41](#_Toc15848089)

# Wstęp

## Uzasadnienie tematu

Mimo rosnącej konkurencji ze strony internetu i serwisów streamingowych (jak Netflix lub Hulu), telewizja wciąż jest popularną formą rozrywki. Najpopularniejsze programy oglądają miliony ludzi, a gust każdego z nich jest inny. Ogromna różnorodność dostępnych opcji (ponad 400 kanałów na terenie Polski) sprawia, że znalezienie interesującego nas programu wymaga przejrzenia wielu stron pełnych spisów nadawanych audycji. W efekcie często zamiast zrelaksować się przy ulubionym programie, przełączamy po kolei kanały, licząc że w końcu trafimy na coś ciekawego.

## Cel pracy

Celem pracy jest ułatwienie użytkownikom oglądaniainteresujących ich programów przez opracowanie aplikacji pozwalającej na 3 kluczowe czynności:

* określenie i zapisanie swoich preferencji,
* pobranie informacji o nadawanych w najbliższym czasie programach
* rekomendacja dotycząca pozycji, które mogą uznać za interesujące na podstawie preferencji i aktualnego programu.

## Zakres prac

W przewidzianym zakresie projektu mieści się zbieranie danych dotyczących nadawanych programów oraz preferencji użytkowników, jak również wykorzystanie zebranych danych do przekazania użytkownikom rekomendacji.

Dane dotyczące programów pochodzą z dostępnych w internecie zasobów zebranych z użyciem programu do web scrapingu. Zebrane dane są później poddane analizie w celu wyznaczenia cech charakterystycznych poszczególnych programów i określenia podobieństwa między nimi. Szczegółowy opis wyróżnianych cech znajduje się w dalszej części pracy. Dodatkowo w ramach programu telewizyjnego dostępne są opisy poszczególnych pozycji – na podstawie zawartych w nich słów kluczowych można określić tematyczne podobieństwo między programami.

Pewnym szczególnym przypadkiem są seriale. Różnią się od większości pojedynczych audycji tym, że istotna jest kolejność odcinków, więc nie można rekomendować użytkownikowi oglądania wszystkich odcinków w przypadkowej kolejności. Konieczna jest identyfikacja numeru odcinka i ustalenie który odcinek użytkownik oglądał ostatnio i rekomendowanie tylko jednego, kolejnego, odcinka.

Dane dotyczące swoich preferencji użytkownik wprowadza sam, oznaczając poszczególne seriale, pojedyncze programy, kanały telewizyjne oraz interesujące go cechy charakterystyczne (aktorzy, drużyny, kategorie itd.) jako ulubione. Ponadto, ponieważ nawet najlepsza rekomendacja nie ma sensu, jeżeli użytkownik w danym czasie nie ma dostępu do telewizora, bardzo ważne są godziny w jakich użytkownik może i chce spędzić swój czas oglądając rekomendowane programy. Na podstawie tak zdefiniowanego profilu użytkownika można zaoferować rekomendacje zgodnych z nim programów.

O oferowanych rekomendacjach możemy informować uzytkownika na dwa sposoby. Pierwszym jest standardowy interfejs aplikacji, który jednak wymaga, żeby użytkownik w danej chwili aktywnie z niej korzystał. Drugim sposobem są powiadomienia *push* przekazywane użytkownikowi w czasie gdy aplikacja działa w tle. Najlepiej działa to w przypadku aplikacji mobilnych, ponieważ powiadomienie można przekazać w dowolnym momencie, ale obecnie aplikacje webowe również pozwalają na wyświetlanie powiadomień w przeglądarce użytkownika.

Ze względu na wieloplatformowość i dostępność zbliżonego interfejsu na urządzeniach desktopowych i mobilnych, aplikacja będzie korzystać z rozwiązań webowych. Część serwerowa zostanie wykonana w formie RESTowego API napisanego w języku *C#* z wykorzystaniem frameworka *ASP.NET Core*. Część kliencka będzie wykorzystywać framework *Angular* w połączeniu z językiem *TypeScript*. Na funkcjonalność dotyczącą powiadomień pozwala zastosowanie technologii PWA (*Progressive Web Apps*) opisanej w dalszej części pracy.

Praca zawiera rozdziały skupiające się na szczegółach dotyczących kolejnych etapów procesu powstawania aplikacji, od przeglądu istniejących na rynku rozwiązań, przez przyjęte założenia projektowe i projekt, do implementacji i testów.

# Przegląd istniejących rozwiązań

Istniejące rozwiązania można podzielić na 3 główne grupy - standardoweprzewodniki programowe, silniki rekomendacji i serwisy streamingowe. W dalszej części nastąpi przegląd każdej z tych grup zawierający przykłady, ich funkcjonalność i ograniczenia.

## Standardoweprzewodniki programowe



* [*http://tv.wp.pl*](http://tv.wp.pl)
* [*https://programtv.onet.pl*](https://programtv.onet.pl)
* [*https://programtv.interia.pl*](https://programtv.interia.pl)

Standardowe przewodniki programowe pozwalają na przegląd aktualnie nadawanych programów. Jest to bardzo ważna, podstawowa funkcjonalność, jednakże z wielu powodów niewystarczająca. Brakuje po pierwsze, możliwości zapamiętania ustawień, a po drugie dostosowania wyświetlanych programów do preferencji. W przypadku programu dostępnego przez portal WP.pl, jedynym elementem personalizacyjnym jest zapisywany w pliku *cookie* filtr kanałów wg dostawcy, podobnie w przypadku Onetu i Interii (w 2 ostatnich dochodzi jeszcze możliwość zmiany widoku z pionowego na poziomy).

Wszystkie 3 główne przykłady oferują możliwość wyszukiwania na podstawie tytułu programu. Szukanie na podstawie cech charakterystycznych umożliwia tylko wp.pl, ale nawet tam jest dość ograniczone. Ponieważ nie zapamiętują żadnych danych poza filtrem kanałów, trzeba za każdym razem określać czego szukamy, co jest uciążliwe. Poza tym brakuje jakichkolwiek sugestii dla użytkownika który nie wie dokładnie czego szuka.

## Silniki rekomendacji

Przykłady:



* [*https://tastedive.com/shows*](https://tastedive.com/shows)
* [*https://www.cabletv.com/what-to-watch*](https://www.cabletv.com/what-to-watch)

Silniki rekomendacji są doskonałym rozwiązaniem dla niezdecydowanego użytkownika. Wymagają wpisania pozycje (w tym przypadku seriale, ale istnieją również takie systemy dla, np. filmów, książek czy muzyki), które użytkownik zna i lubi, a w odpowiedzi przedstawiają inne, podobne. Dla zestawu na przykładowym screenie aplikacja zaproponuje podobną produkcję opartą na komiksach – *The Flash*.

Pokazana aplikacja nie pozwala na żadną dodatkową personalizację, nie można zapamiętać rekomendacji, jedyny sposób na dokładniejsze rekomendacje to 2 przyciski – „not that one” i „start again”. Prostota użytkowania jest niewątpliwym plusem, ale brakuje wielu istotnych funkcjonalności.

Istnieją również silniki rekomendacji korzystające z serwisów społecznościowych do określenia preferencji użytkownika, ich przykładem jest *tastedive.com*. *Tastedive* działa w 2 trybach – pierwszy jest podobny do poprzedniego, ale pozwala tylko na wprowadzenie jednej pozycji (serialu, filmu, książki, artysty muzycznego lub autora). Drugi wymaga zalogowania się i pozwala oznaczać istniejące pozycje jako ulubione (jeżeli logujemy się za pośrednictwem portalu *Facebook*, *Tastedive* zbiera nasze polubienia i wyznacza rekomendacje na tej podstawie).

Problemem, który ogranicza przydatność wszystkich aplikacji z tej grupy jest to, że nie uwzględniają w żaden sposób aktualnego programu telewizyjnego. Znalezienie serialu lub filmu, który okaże się interesujący (o czym użytkownik wie dzięki rekomendacjom) nadal wymaga przeszukiwania tradycyjnych programów telewizyjnych.

## Serwisy streamingowe

Przykłady:



* [*https://pilot.wp.pl/tv*](https://pilot.wp.pl/tv)
* [*https://www.ipla.tv/start*](https://www.ipla.tv/start)
* [*https://www.popcornflix.com*](https://www.popcornflix.com)
* [*https://123vidz.com*](https://123vidz.com)

Serwisy streamingowe są mocno zróżnicowaną grupą, pod względem dostarczanej treści, kosztów użytkowania jak i kwestii prawnych.

Jednym przykładem jest oferowany przez portal *wp.pl* serwis *WP pilot* – pozwala on oglądać dokładnie to, co jest aktualnie nadawane w telewizji. Podstawowa oferta 28 kanałów jest dostępna za darmo, pełna wymaga opłat. Wyszukiwarka pozwala znaleźć kanał lub aktualnie nadawany program na podstawie tytułu lub opisu, dostępny jest też podgląd programów na najbliższe godziny. Nadal brakuje możliwości zapamiętania ulubionych kanałów lub programów, nawet po zalogowaniu, a także rekomendacji.

Przykładem z drugiej strony jest serwis *ipla.tv*. Dostarcza on ofertę filmów, seriali i programów sportowych uboższą od telewizyjnej, ale dostępną na żądanie, w większości za opłatą, choć częśc programów jest darmowa. Rekomendacje sprowadzają się jedynie do wskazania najpopularniejszych, nie są personalizowane. Podobne możliwości, poszerzone o rekomendacje, oferują serwisy takie, jak *Netflix*, *Hulu*, czy *Amazon Video.*

## Filmweb



Polski *Filmweb* jest dość unikalnym przypadkiem, który wydaje się oferować wszystko, czego można by wymagać. Przewodnik programowy, rekomendacje uwzględniające preferencje użytkownika i nadawane programyboth, możliwość wyszukiwania na podstawie różnych cech, filtrowania kanałów wg dostawców i tematycznie.

Brakuje jednak kilku elementów, po pierwsze programów sportowych, popularnonaukowych czy ogólnie niezwiązanych ze światem filmów. Nie powinno to być zaskoczeniem, biorąc pod uwagę to, że Filmweb od początku skupiony był, jak sama nazwa wskazuje, na filmach, częściowo także na serialach. Jest to jednak funkcjonalność, która ma duże znaczenie dla wielu użytkowników.

Poza tym brakuje powiadomień. Dostępne są powiadomienia dotyczące aktywności w serwisie – opublikowanych recenzji, dyskusji na forach itd., ale nic związanego z programem telewizyjnym. Poza tym, nawet te powiadomienia, które są dostępne, wykorzystują e-mail lub komunikator GG zamiast korzystania z nowszych technologii powiadomień push – przez przeglądarkę lub aplikację mobilną.

## Tabela porównawcza

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| funkcjonalność | programtv .onet.pl | tv.wp.pl | pilot.wp.pl | filmweb.pl | tastedive.com | cabletv.com | ipla.pl | popcornflix. com | moja propozycja |
| przegląd nadawanych programów | TAK | TAK | TAK | TAK | NIE | NIE | NIE | NIE | TAK |
| personalizacja kanałów | TAK | TAK | TAK | TAK | NIE | NIE | NIE | NIE | TAK |
| personalizacja treści | NIE | NIE | NIE | TAK | NIE | TAK | TAK | TAK | TAK |
| personalizacja czasu | TAK | TAK | NIE |  | NIE | NIE | TAK | TAK | TAK |
| wyszukiwanie programów | TAK | TAK | TAK | *2 niżej* | TAK | TAK | TAK | TAK | TAK |
| widok chronologiczny | TAK | TAK | NIE | TAK | NIE | NIE | NIE | NIE | TAK |
| rekomendacje filmów | *3 niżej* | *3 niżej* | NIE | TAK | TAK | NIE | *3 niżej* | TAK | TAK |
| rekomendacje seriali | *3 niżej* | *3 niżej* | NIE | TAK | TAK | TAK | *3 niżej* | TAK | TAK |
| rekomendacje programów sportowych | *3 niżej* | *3 niżej* | NIE | NIE | NIE | NIE | *3 niżej* | NIE | TAK |
| śledzenie odcinków seriali | NIE | NIE | NIE | TAK | NIE | NIE | NIE | NIE | TAK |
| VOD | NIE | NIE | NIE | NIE | NIE | NIE | TAK | TAK | NIE |
| streaming treści na żywo | NIE | NIE | TAK | NIE | NIE | NIE | TAK | NIE | NIE |
| powiadomienia | NIE | NIE | NIE | NIE | NIE | NIE | NIE | NIE | TAK |
| darmowy | TAK | TAK |  | TAK | TAK | TAK |  | TAK | TAK |

**Uwagi:**

1. ograniczenia w personalizacji czasu nadawania polegają na możliwości wyboru tylko określonych pór dnia (cały dzień, po południu lub wieczorem)
2. ograniczenia w wyszukiwaniu programów - można znaleźć tylko tytuły
3. ograniczenia w rekomendacjach - wyświetlane są "najpopularniejsze" programy danego typu, bez uwzględnienia preferencji użytkownika
4. *pilot.wp.pl* ma darmowy zestaw kanałów, *ipla.tv* pozwala za darmo obejrzeć tylko pojedyncze programy (głównie informacyjne). *popcornflix.com* jest darmowy, ale dostępne treści są ograniczone do tych mniej popularnych

## Wnioski

Funkcjonalność dotyczącą rekomendacji i programu w pewien sposób łączy serwis Filmweb ([*https://www.filmweb.pl*](https://www.filmweb.pl)), jednak skupia się on tylko na filmach i serialach.W związku z tym nie uwzględnia programów sportowych ani muzycznych, a dane dotyczące np. programów popularnonaukowych są mocno ograniczone.

Brakuje na rynku rozwiązań oferujących rekomendacje z uwzględnieniem aktualnego programu. Legalnie działające aplikacje dostarczające treści (w formie VOD lub strumienia na żywo) najczęściej są płatne (co nie powinno dziwić ze względu na koszty samej treści). Z tego powodu implementacja tej funkcjonalności nie mieści się w zakresie pracy.

Wszystkie rozwiązania pozwalające na wyszukiwanie mają jeden problem, polegający na tym, że wymagają poprawnie wprowadzonych tytułów, nie są w stanie skorygować błędów użytkownika. Literówka w tytule, oznacza że wyszukiwarka nie zwróci żadnych wyników, mimo że istnieją tytuły podobnie brzmiące, które lepszy algorytm byłby w stanie wychwycić. Jest to szczególnie istotne w przypadku aplikacji skupionych na rekomendacjach.

# Założenia projektowe

## Opis problemu

Obecnie nie ma na rynku żadnego rozwiązania, które pozwalałoby stwierdzić który z aktualnie nadawanych programów zainteresowałby użytkownika. Takie rozwiązanie jest potrzebne, ze względu na dużą liczbę możliwości, która sprawia że ręczne szukanie jest niewygodne, a wielu użytkowników szuka „czegoś do oglądania na wieczór”, bardziej niż konkretnego programu.

## Wizja rozwiązania

Aplikacja ma łączyć w sobie rozwiązania z silników rekomendacji i standardowych programów TV. Dzięki temu użytkownicy będą mogli dowiedzieć się jaki interesujący ich program jest nadawany, w momencie kiedy jest nadawany. To pozwoli im skupić się na oglądanym programie, zamiast na poszukiwaniach i spędzić więcej czasu na oglądaniu.

Po stronie klienta główny widok zawiera chronologiczny wykaz rekomendowanych programów nadawanych w najbliższym tygodniu. Wykaz uwzględnia programy powtarzane – zawiera tylko jedno wystąpienie, wybrane na podstawie godzin w sposób unikający ewentualnych konfliktów z innymi programami, a także uwzględniający godziny w jakich użytkownik ma dostęp do telewizora. Ponadto główny widok zawiera formularz do konfiguracji filtrów, pozwalający użytkownikowi ustalić godziny, kanały i zakres tematyczny. Konflikty, których nie udało się uniknąć są zaznaczone kolorem. Schematy interfejsu zostaną przedstawione w części projektowej.

Do dodatkowych widoków należy widok przewodnika programowego, zawierający wszystkie nadawane programy bez uwzględniania rekomendacji, widok wyszukiwania oraz widok pokazujący szczegóły danego programu. Z poziomu każdego z tych widoków użytkownik może zaznaczyć, że dany program mu się podobał lub nie, może też zapamiętać że chce obejrzeć dany program. Ustawione filtry czasowe i kanałowe oraz oznaczenia programów są zapisywane na serwerze na bieżąco.

Po stronie serwera, rekomendacje generowane są na podstawie cech charakterystycznych programów, do których zaliczamy:

* Osoby pracujące przy produkcji programu (aktorzy, reżyserowie, prezenterzy, scenarzyści, w przypadku transmisji sportowych – grające drużyny, w przypadku programów muzycznych – wykonawcy, itd.)
* Kategorie (dla filmów i seriali są to, np. thriller, fantasy, komedia, dokument; dla sportu dyscyplina i ranga zawodów – liga, puchar)
* Zawarte w słownym opisie programu słowa kluczowe

Obsada i kategoria są znane dzięki danym zebranym z programów telewizyjnych. Słowa kluczowe są dobierane automatycznie.Pozwoli to na wychwycenie tematycznych podobieństw, które nie są widoczne w zdefiniowanych cechach (jak kategorie czy obsada).

Tytuły seriali można ustalić na podstawie wspólnych części początkowych w tytułach odcinków. Łatwo to zauważyć patrząc na tytuły przykładowego serialu krymiminalnego:

* *Castle -ost.*
* *Castle: Kumple z funduszu hedgingowego. s01e03*
* *Castle: W piekle furia nie gości. s01e04*
* *Castle: Der Scharfschütze*
* *Castle. s01e09*
* *Castle . s08e09*

Wszystkie odcinki mają część wspólną na początku – *Castle* – poza tym część z nich zawiera w tytule numer kolejny odcinka (w notacji zawierającej również sezon). Na tej podstawie można stwierdzić do jakiego serialu należy i wykorzystać to do rekomendacji.

**Metoda generowania rekomendacji**

Rekomendacje generowane są w oparciu o ważoną średnią ocen użytkownika wyrażoną wzorem:

Gdzie:

* – program, którego współczynnika rekomendacji poszukujemy
* – programy wcześniej ocenione przez użytkownika
* – ocena wystawiona j-temu programowi przez użytkownika, wynosi 1, jeżeli użytkownik zaznaczył, że program mu się podobał, 0 jeżeli zaznaczył, że program mu się nie podobał. Do tych celów, zapisanie programu jako ten, który użytkownik chce obejrzeć, jest traktowane jako ocena pozytywna.
* – podobieństwo dwóch programów określone wg wzoru:

Gdzie – wagi dla i-tej grupy cech, – podobieństwo i-tej grupy cech danego programu. Wyróżniamy 6 grup cech:

* Gatunki
* Słowa kluczowe
* Aktorzy
* Reżyser
* Kraj produkcji
* Rok produkcji

Dla gatunków, słów kluczowych, reżyserów i aktorów, podobieństwo wyrażamy współczynnikiem Dice’a:

Dla kraju produkcji podobieństwo jest binarne, wynosi 1 gdy wartości są te same, 0 w przeciwnym wypadku.

Dla roku produkcji podobieństwo wyraża się wzorem:

Jest to funkcja homograficzna, przyjmująca wartość 0, gdy argument jest równy 0, wartość gdy i asymptotycznie dążąca do wartości 1. Dla parametru przyjmujemy wartość 20 lat.

Wagi dostosowywane są indywidualnie do każdego z użytkowników. Przy rejestracji nowego użytkownika przyjmuje się wartości średnie dla wszystkich użytkowników, a w przypadku ich braku, wartości następujące:

* Aktorzy: *0,1*
* Kategorie: *0,3*
* Słowa kluczowe: *0,3*
* Reżyser: *0,1*
* Kraj produkcji: *0,1*
* Rok produkcji: *0,1*

Za każdym razem, kiedy użytkownik oceni program, dokonywana jest korekcja wag o wartość:

Gdzie

* – parametr z przedziału ,
* – ocena wystawiona programowi przez użytkownika,
* – wartość funkcji wsparcia rekomendacji dla programu

## Wymagania funkcjonalne

Gość:

* ST-001: Jako gość, chcę przeglądać program TV, żeby sprawdzić co mogę obejrzeć
* ST-002: Jako gość, chcę wybrać tylko te kanały które mnie interesują, żeby uprościć szukanie
* ST-003: Jako gość, chcę zawęzić kanały do oferty mojego dostawcy, żeby uprościć szukanie
* ST-004: Jako gość, chcę sprawdzić o której godzinie nadawany jest interesujący mnie program
* ST-005: Jako gość, chcę zarejestrować się w systemie, żeby zapisać moje preferencje

[Zarejestrowany] użytkownik

* ST-006: Jako użytkownik chcę sprawdzić kiedy nadawane są interesujące mnie programy
* ST-007: Jako użytkownik chcę otrzymywać powiadomienia o interesujących mnie programach, żeby ich nie przegapić
* ST-008: Jako użytkownik chcę zapisać filtr programów, żeby nie ustawiać go za każdym razem
* ST-009: Jako użytkownik chcę otrzymać rekomendacje programów, które mogą mnie zainteresować, żeby ułatwić wyszukiwanie
* ST-010: Jako użytkownik chcę potwierdzić lub odrzucić rekomendację, żeby otrzymywać dokładniejsze wyniki
* ST-011: Jako użytkownik chcę ustawić w jakich godzinach otrzymuję powiadomienia, żeby nie otrzymywać ich wtedy, kiedy nie mam dostępu do telewizora
* ST-012: Jako użytkownik chcę śledzić kolejność odcinków wybranych seriali, żeby nie otrzymywać rekomendacji dotyczących odcinków, które już widziałem

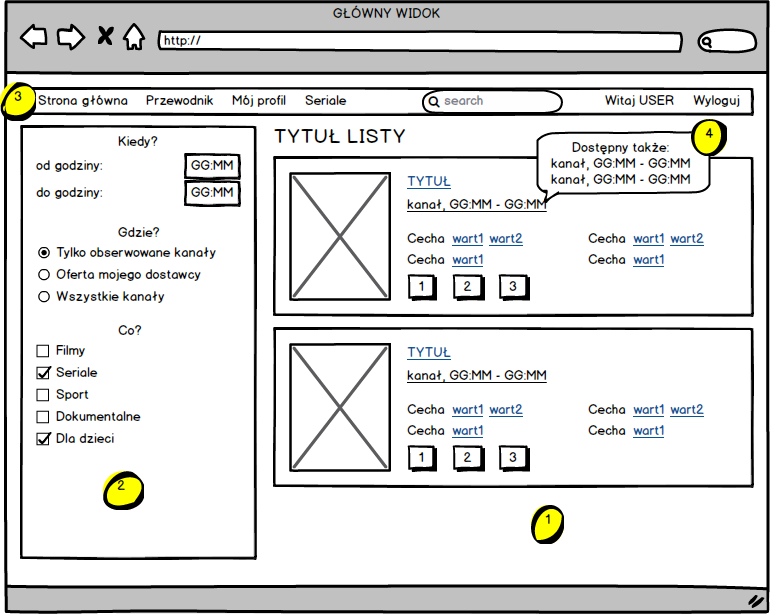
# Projekt aplikacji

## Baza danych



## Prototyp interfejsu

* + 1. Główny widok aplikacji

****

Widok główny składa się z 3 głównych części – oznaczonych odpowiednio na schemacie.

Część oznaczona numerem 1 zawiera listę programów, której zawartość zmienia się w zależności od kontekstu. Widok listy używany jest w następujących przypadkach:

* Lista rekomendowanych programów
* Lista programów podobnych do danego
* Lista programów charakteryzowanych przez daną cechę
* Lista programów nadawanych na danym kanale
* Lista odcinków śledzonych przez użytkownika seriali

Kliknięcie na tytuł programu w liście przekierowuje do widoku szczegółów danego programu, kliknięcie na jeden z linków dotyczących cech przekierowuje do listy programów zawierających tą cechę (widok główny wypełniony listą programów o danej cesze). Popdowiedź oznaczona numerem 4 pojawia się po najechaniu myszą na informację o kanale i godzinach. Przyciski 1,2,3 oznaczają odpowiednio „podoba mi się”, „chcę obejrzeć” i „nie podoba mi się” – kliknięcie w jeden z nich zapisuje ocenę programu na serwerze.

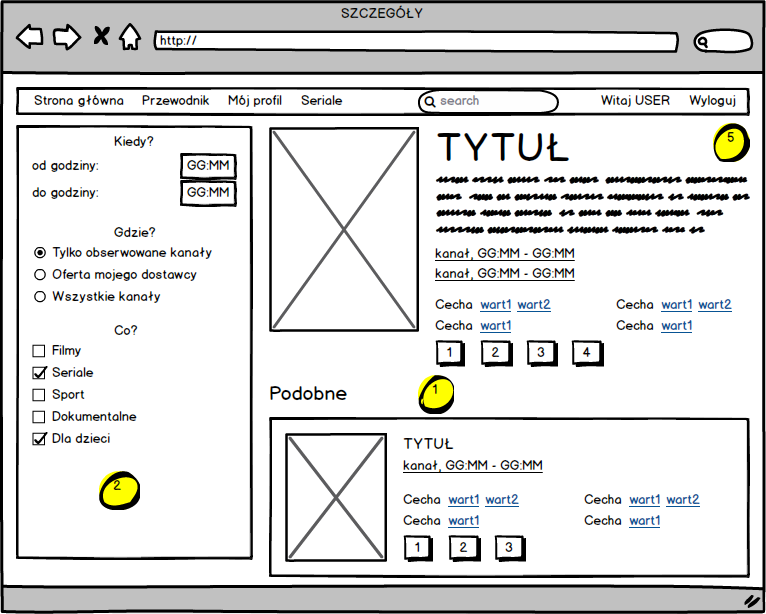
Część oznaczona numerem 2 pozwala na dostosowanie zawartości listy do preferencji użytkownika w zakresie godzin, kanałów i tematyki. Dokładniejszy wybór kanałów jest dostępny przez ustawienia konta (opisane w dalszej części). Ustawienia są zapisywane na bieżąco na serwerze i wykorzystywane do filtrowania aktualnej zawartości listy (opisanej wcześniej pod numerem 1)

Część oznaczona numerem 3 jest wspólna dla wszystkich widoków i pełni funkcję menu nawigacyjnego. Przyciski po lewej stronie przekierowują do odpowiednich podstron systemu, kolejno:

* Strona główna – strona zawierająca listę rekomendacji dla użytkownika (widok główny wypełniony listą rekomendacji)
* Przewodnik – widok przewodnika programowego, omówiony szczegółowo w dalszym rozdziale
* Mój profil – ustawienia profilu użytkownika, omówione w dalszym rozdziale
* Seriale – strona zawierająca listę odcinków śledzonych przez użytkownika seriali (widok główny wypełniony listą odcinków seriali)

Użycie paska wyszukiwania powoduje wyświetlenie listy wyników – programów powiązanych z wpisanym terminem poprzez tytuł lub jedną z cech (widok główny wypełniony wynikami wyszukiwania posortowanymi wg stopnia dopasowania liczonego na podstawie współczynnika Dice’a pomiędzy słowami wpisanymi przez użytkownika, a cechami i tytułem programu).

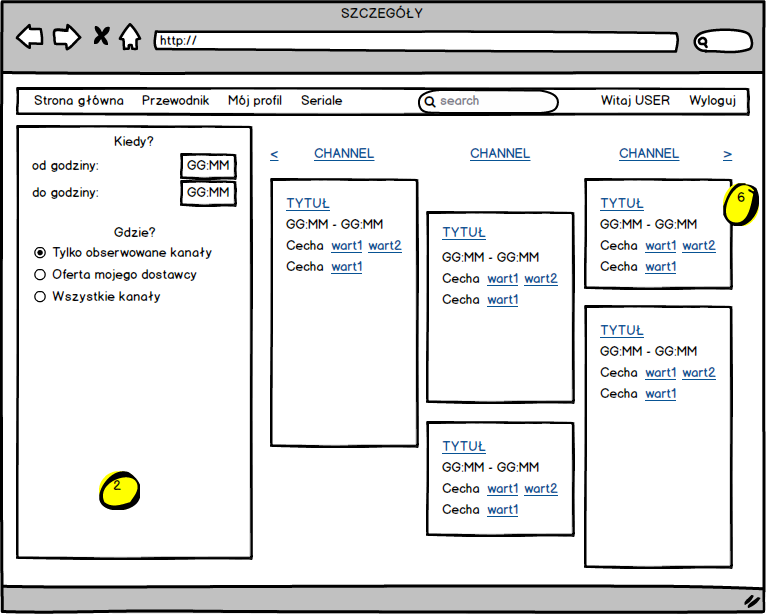
* + 1. Widok szczegółów programu



Szczegóły programu są przedstawione w części oznaczonej numerem 5. Zawiera ona, poza elementami widocznymi na pozycjach listy, dokładny opis programu, listę wszystkich emisji ułożoną chronologicznie oraz jeden dodatkowy przycisk, oznaczony numerem 4. Jest on widoczny tylko jeżeli program jest odcinkiem serialu, zapisuje on informację o śledzeniu tego serialu.

Poniżej szczegółów znajduje się lista podobnych do niego programów oznaczona numerem 1 (jest to ten sam element, co w widoku głównym, wypełniony listą podobnych programów). Listę można filtrować wykorzystując znajdujące się po lewej stronie ustawienia (oznaczone numerem 2).

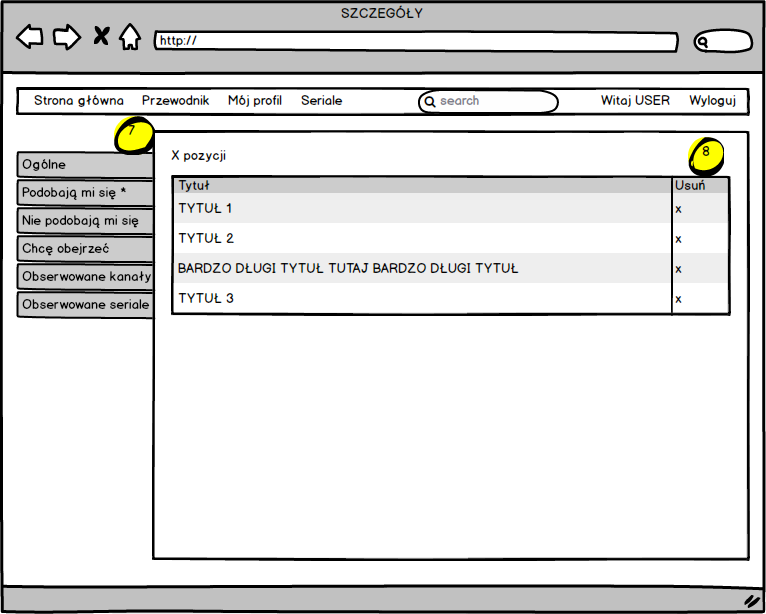
* + 1. Widok przewodnika programowego



Widok przewodnika programowego zawiera informację o wszystkich programach nadawanych na wybranym kanale (oznaczoną numerem 6). Każdy program jest reprezentowany przez jeden prostokąt zawierający podstawowe informacje na jego temat. Umiejscowienie prostokąta na osi pionowej zależy od godziny rozpoczęcia programu, a wysokość prostokąta od czasu jego trwania. Kliknięcie w tytuł programu przekierowuje użytkownika do widoku szczegółów omówionego wcześniej, linki odpowiadające cechom do stron zawierających wszystkie programy charakteryzowane przez daną cechę, a linki opisane nazwą kanału – do listy wszystkich programów nadawanych na danym kanale.

Widok przewodnika programowego można filtrować z wykorzystaniem panelu oznaczonego numerem 2, zawiera on te same ustawienia co w poprzednich widokach z wyjątkiem wyboru tematyki. Ustawienia są zapamiętywane na serwerze, więc ich wartość jest przechowywana pomiędzy podstronami.

* + 1. Widok ustawień preferencji



Widok ustawień preferencji pozwala na przegląd zapisanych dotąd ocen i usunięcie ich. Po lewej stronie znajdują się zakładki służące do wyboru aktualnie zarządzanej listy (oznaczone numerem 7). Tabela oznaczona numerem 8 pojawia się w przypadku wyboru zakładek:

* Podobają mi się
* Nie podobają mi się
* Chcę obejrzeć
* Obserwowane kanały
* Obserwowane seriale

Kliknięcie w x po prawej stronie tabeli usuwa wpis dotyczący danej oceny z abazy danych.

W zakładce „ogólne” znajdują się opcje dotyczące zarządzania kontem (jak zmiana hasła) oraz wybór dostawcy telewizji.

* + 1. Dostępne listy programów

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nr | Nazwa | Zalogowany | Filtr | Sortowanie | Dodatkowa funkcjonalność |
| V-01 | Rekomendacje | TAK | Funkcja rekomendacji  > próg | Funkcja rekomendacji | Podobne spośród ocenionych |
| V-02 | Ocenione | TAK | Isnieje ocena | Wartość oceny, potem funkcja rekomendacji | Wartość oceny |
| V-03 | Kanał | NIE | Emisja na danym kanale | Godzina emisji | Możliwość śledzenia kanału |
| V-04 | Kanał | TAK | Emisja na danym kanale | Godzina emisji | Możliwość śledzenia kanału |
| V-05 | Cecha | NIE | Występowanie danej cechy | Godzina emisji | - |
| V-06 | Cecha | TAK | Występowanie danej cechy | Funkcja rekomendacji,  potem godzina emisji | - |
| V-07 | Podobne | NIE | Podobieństwo do wskazanego programu  > próg | Podobieństwo do wskazanego programu | Szczegóły wskazanego programu |
| V-08 | Podobne | TAK | Podobieństwo do wskazanego programu  > próg | Podobieństwo do wskazanego programu | Szczegóły wskazanego programu |
| V-09 | Szukaj | NIE | Tytuł lub cechy zawierają szukane słowa | Ilość dopasowanych słów | - |
| V-10 | Szukaj | TAK | Tytuł lub cechy zawierają szukane słowa | Ilość dopasowanych słów | - |

|  |  |
| --- | --- |
| Nr widoku | Adres kontrolera API |
| V-01 | /api/Users/*login*/Recommended |
| V-02 | /api /Users/*login*/Ratings |
| V-03 | /api /Channel/*id*/Programmes |
| V-04 | /api /Channel/*id*/Programmes |
| V-05 | /api/Feature/*id*/Programmes |
| V-06 | /api/Feature/*id*/Programmes?username=*login* |
| V-07 | /api/Programmes/*id*/Similar |
| V-08 | /api/Programmes/*id*/Similar?username=*login* |
| V-09 | /api/Programmes?search=*search terms* |
| V-10 | /api/Programmes?search=*search terms* |

## Architektura aplikacji

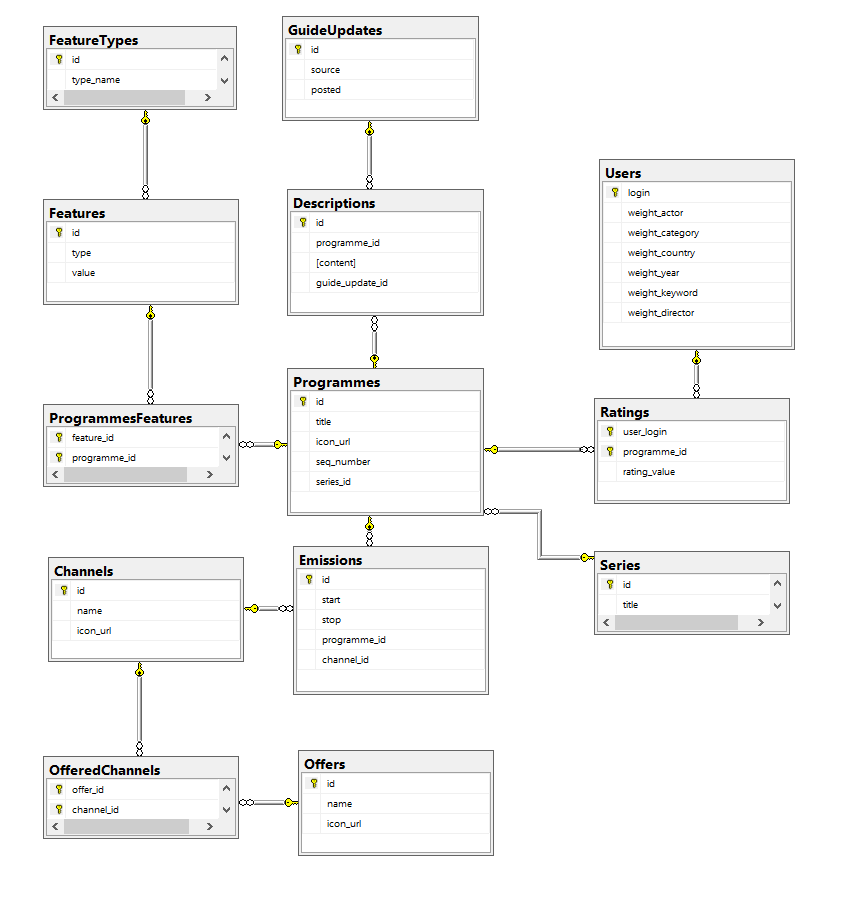
# Implementacja

## Baza danych

* + 1. Technologia:

Ze względu na najlepsze wsparcie dla Entity Framework Core, aplikacja korzysta z bazy danych Microsoft SQL

* + 1. Przechowywane dane



1. Wygenerowany przez SQL Server Management Studio schemat bazy danych

Baza danych aplikacji przechowuje informacje dotyczące poszczególnych programów telewizyjnych w tabelach *Programmes, Emissions, Descriptions* i *ProgrammesFeatures*. Tabela *Features* zawiera poszczególne cechy – nazwiska aktorów, reżyserów i scenarzystów. Tabela *Ratings* zawiera oceny wystawione przez użytkowników programom. Dostępne wartości ocen:

* 0 – „*nie interesuje mnie*” – aplikacja nie będzie rekomendowała podobnych programów
* 1 – „*podobał mi się*” *­*– aplikacja będzie rekomendowała podobne programy, ale pomijała ten konkretny (użytkownik już go oglądał, więc spodziewamy się że nie chce oglądać go po raz kolejny)
* 2 – „*chcę obejrzeć*” – aplikacja będzie rekomendowała podobne programy i przypomni użytkownikowi przed planowaną emisją

## Serwer

* + 1. **Obiekty domenowe i komunikacja z bazą danych**

Obiekty domenowe składają się z publicznych właściwości i anotacji, na podstawie których *EntityFramework* określa typy poszczególnych kolumn w bazie danych.



Przykładowa implementacja klasy reprezentującej kanał telewizyjny.

* + 1. **Funkcjonalność**
       1. Wczytywanie danych

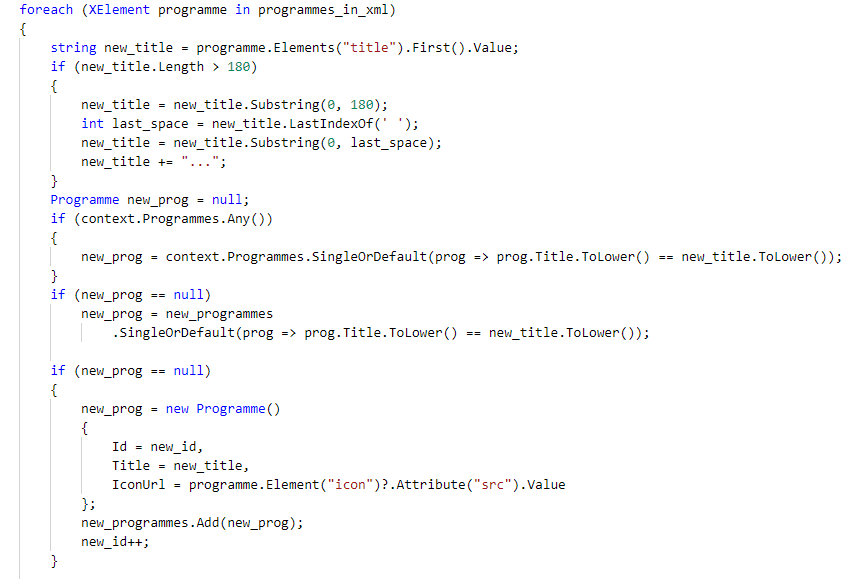
Wczytywanie danych jest procesem dwuetapowym:

* Etap 1 – pobranie programu z internetu
* Etap 2 – wczytanie programu do bazy

Etap 1 jest realizowany przez zewnętrzny program Webgrabplus, działający w osobnym kontenerze dockerowym i uruchamiany cyklicznie, codziennie o 1. w nocy. Efektem działania jest plik XML zawierający listę kanałów i nadawanych programów na ustaloną liczbę dni (w tym przypadku 4).

Etap 2 jest realizowany przez właściwy serwer aplikacji, do tego celu jest wyznaczony osobny endpoint API – pod adresem */api/GuideUpdate*. Zaciągnięty w etapie 1 program jest wysyłany do serwera aplikacji pod tym adresem.

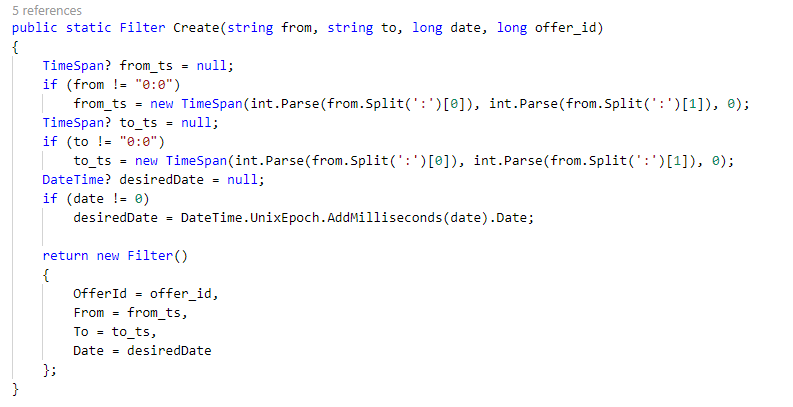
Serwer po odebraniu pliku przekształca zapis XML na odpowiednie obiekty domenowe (sprawdzając, czy nie istnieją one wcześniej w bazie), po czym zapisuje przekształcone obiekty w bazie danych. Za samo przetwarzanie pobranych danych odpowiada klasa *GuideUpdateService*. Korzysta ona z wbudowanego w język C# narzędzia   
LINQ-to-XML, które znacznie ułatwia przetwarzanie plików w tym formacie.



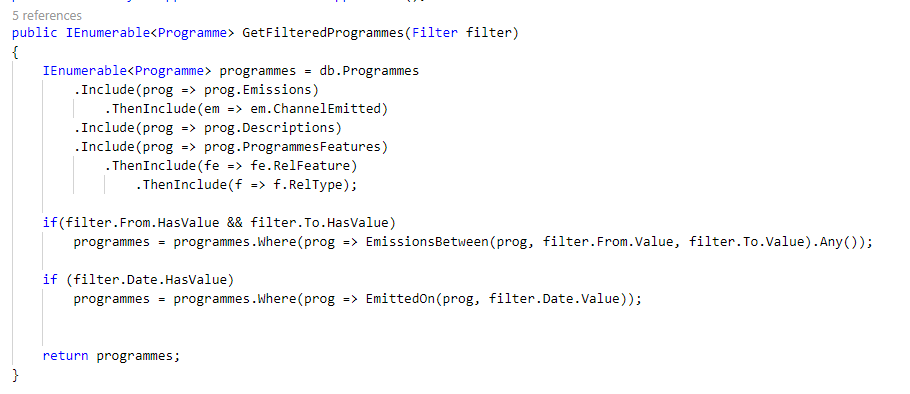
W podobny sposób przetwarzane są dane dotyczące godzin emisji i cech

* + - 1. Filtrowanie

Filtrowanie na podstawie daty, godziny i oferty kanałów jest realizowane w klasie *ProgrammeService*. Za utworzenie filtra z przekazanych parametrów odpowiada metoda *Create*

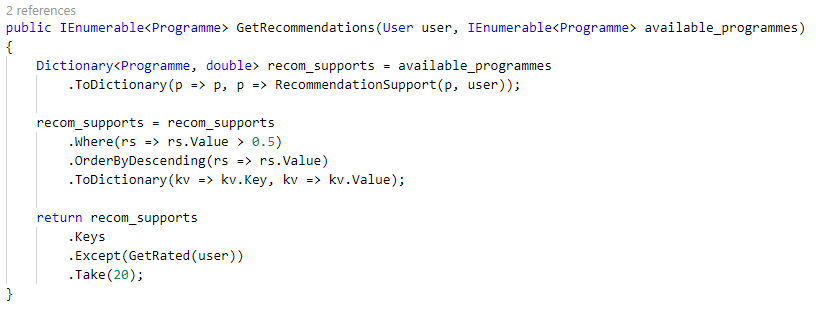


Przygotowany filtr wykorzystuje metoda *GetFilteredProgrammes* z klasy odpowiedzialna za pobieranie listy programów z bazy danych

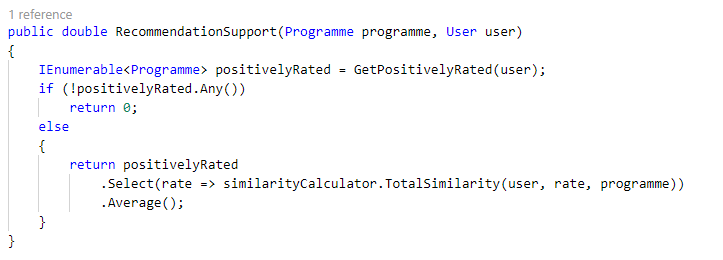


* + - 1. Wyznaczanie rekomendacji

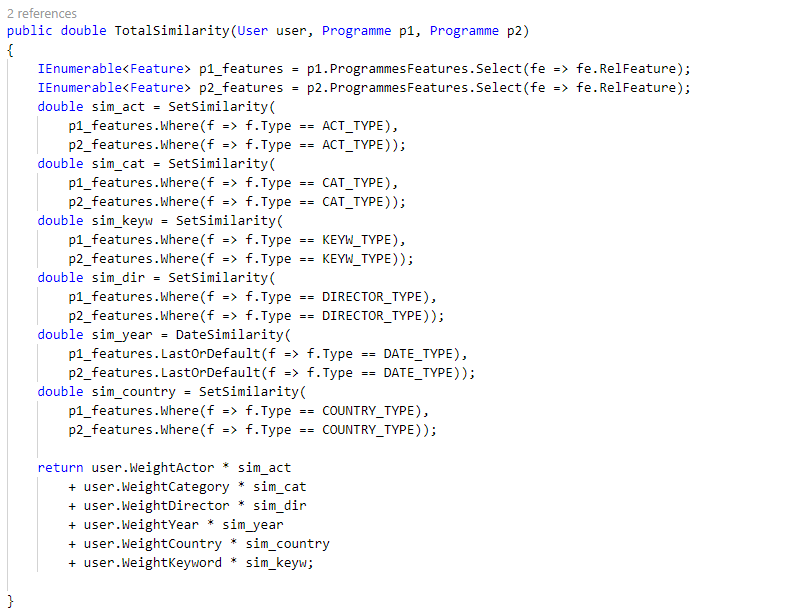
Za wyznaczanie rekomendacji użytkownikom odpowiada klasa *RecommendationService* oraz pomocnicza klasa wyznaczająca podobieństwa ­– *SimilarityCalculator*. Bezpośrednio wywoływaną metodą jest *GetRecommendations*, przyjmująca jako parametr dane użytkownika i listę dostępnych programów



Efektem działania jest lista co najwyżej 20 programów posortowanych wg średniego podobieństwa do programów, które użytkownik wcześniej oznaczył jako będące w jego guście. Średnie podobieństwo określa metoda *RecommendationSupport*:



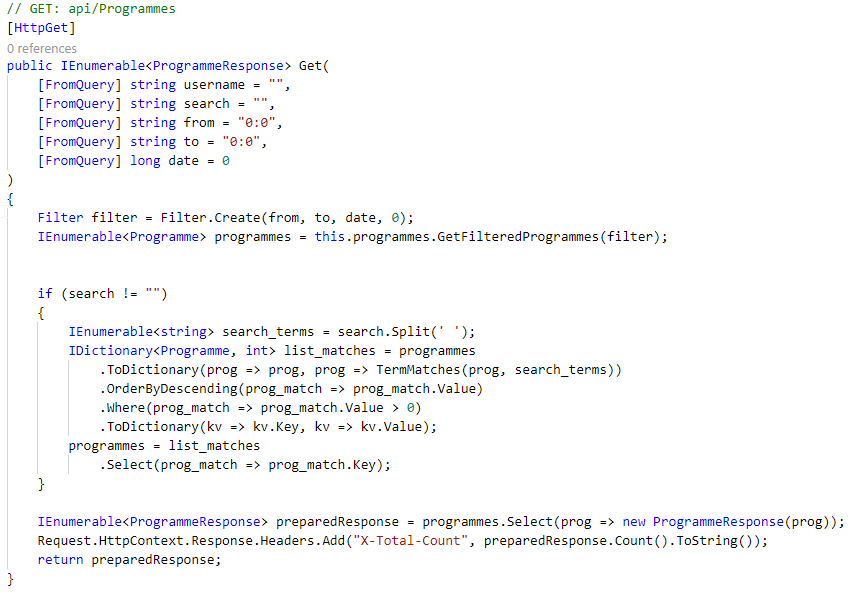
*RecommendationSupport* z kolei korzysta z metod określających podobieństwo programów zdefiniowanych w *SimilarityCalculator*, pierwszą z nich jest *TotalSimilarity*, zwracająca średnią ważoną wg wag charakterystycznych dla użytkownika.



Podstawy matematyczne przedstawionej tu logiki zostały omówione w rozdziale 3.

* + 1. **Technologia**
       1. ASP.NET Core Web API

Kluczową funkcjonalnością dostarczaną przez framework ASP.NET Core jest powiązanie kontrolerów i ich metod z odpowiednimi żądaniami HTTP przekazywanymi pod poszczególne adresy. Jest to zrealizowane za pomocą odpowiednich anotacji



Ponieważ każda klasa kontrolera dziedziczy po należącej do frameworka klasie *BaseController,* do programisty należy tylko implementacja odpowiednich metod i oznaczenie ich anotacjami, powiązanie z adresem URL i metodą HTTP następuje automatycznie.

* + - 1. Dostęp do bazy danych – EF Core

Do komunikacji z bazą danych aplikacja wykorzystuje udostępnione przez Microsoft narzędzie ORM (*Object – Relational Mapping*) w podejściu *code – first*. W tym podejściu programista definiuje klasy obiektów domenowych (typu POCO, nie wykonujące żadnej logiki, ani nie posiadające żadnych zależności) oraz klasę kontekstu (w tym przypadku *TvAppContext*) przechowującą parametry połączenia z bazą (i dodatkowe informacje dotyczące schematu nieokreślone przez klasy domenowe – np. indeksy). Na podstawie tych klas EntityFramework generuje kod migracji, a potem polecenia DDL wywoływane na bazie danych, w tym przypadku SQL–owej.

Główną zaletą takiego podejścia jest znaczne uproszczenie komunikacji z bazą danych przez dodatkową warstwę abstrakcji. Zapis i odczyt danych z bazy sprowadza się do interakcji z kolekcją (dostarczona w EF Core klasa bazowa *DbSet* implementuje interfejs *IEnumerable*, ten sam który wykorzystują wszystkie dostępne w C# kolekcje). W podejściu *code – first* całkowicie wyeliminowana została konieczność pisania zapytań do bazy za pośrednictwem SQL.Dodatkowo pozwala to korzystać z biblioteki LINQ, która udostępnia metody pozwalające na filtrowanie, sortowanie i przekształcanie danych w kolekcji.

Dodatkowo wykorzystanie narzędzia ORM zwiększa poziom bezpieczeństwa aplikacji blokując ataki typu *SQL injection* – polegające na „wstrzyknięciu” kodu SQL do składanych dynamicznie zapytań.

* + - 1. Powiadomienia Push – SignalR

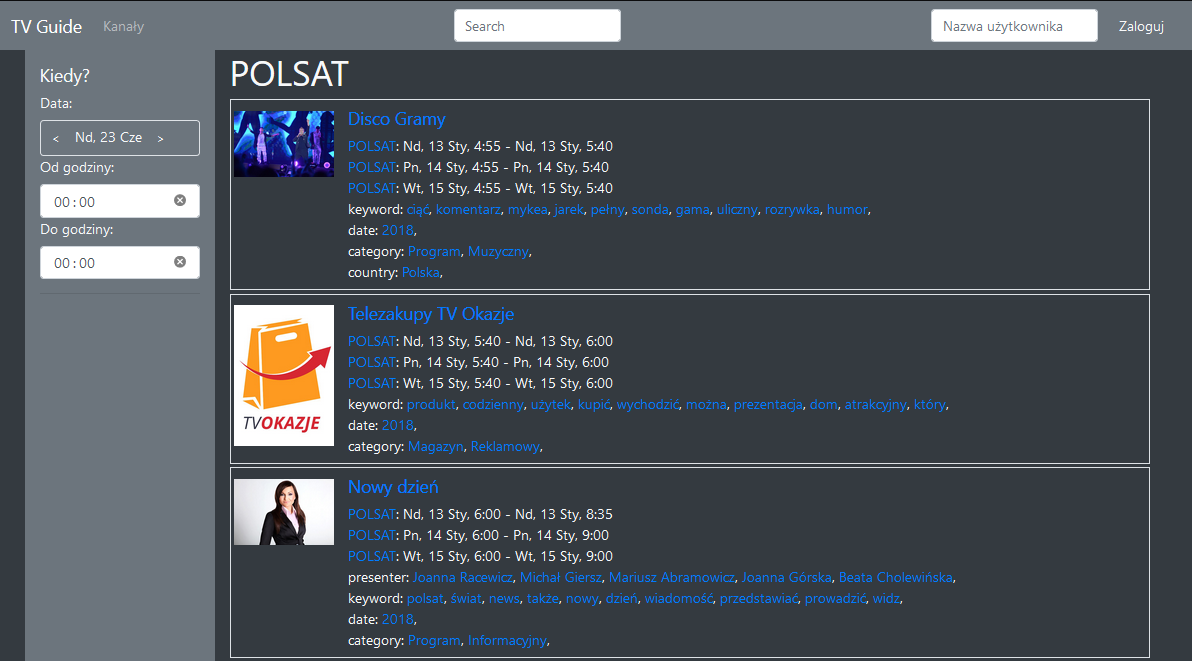
## Klient

Kod aplikacji klienckiej jest podzielony na 2 główne foldery: *containers* i *shared*. Folder *containers* zawiera główne ekrany aplikacji, dla każdego z nich pliki określające odpowienio:

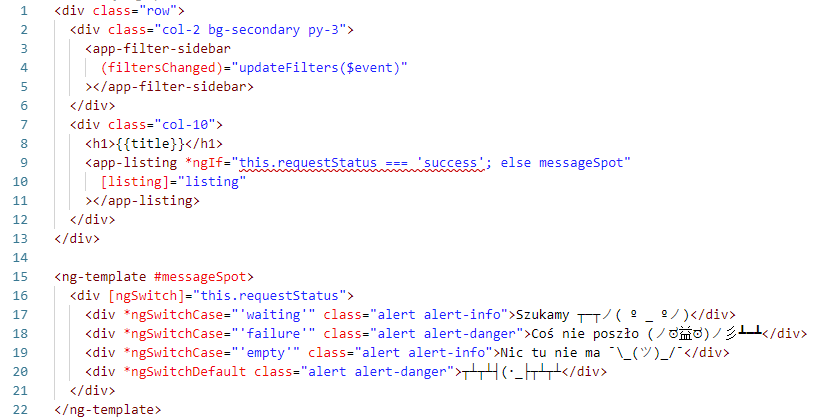
* .component.html – układ elementów na ekranie
* .component.scss – specyficzne dla danego widoku style
* .component.ts – używane przez widok dane i wewnętrzna logikę
* .component.spec.ts – testy jednostkowe dla widoku
* .service.ts – logika odpowiadająca za komunikację z API
* .service.spec.ts – testy jednostkowe dla komunikacji z API



Przykładowo, ekran zawierający listę programów nadawanych na określonym kanale – zawarty w folderze *channel-listing*:



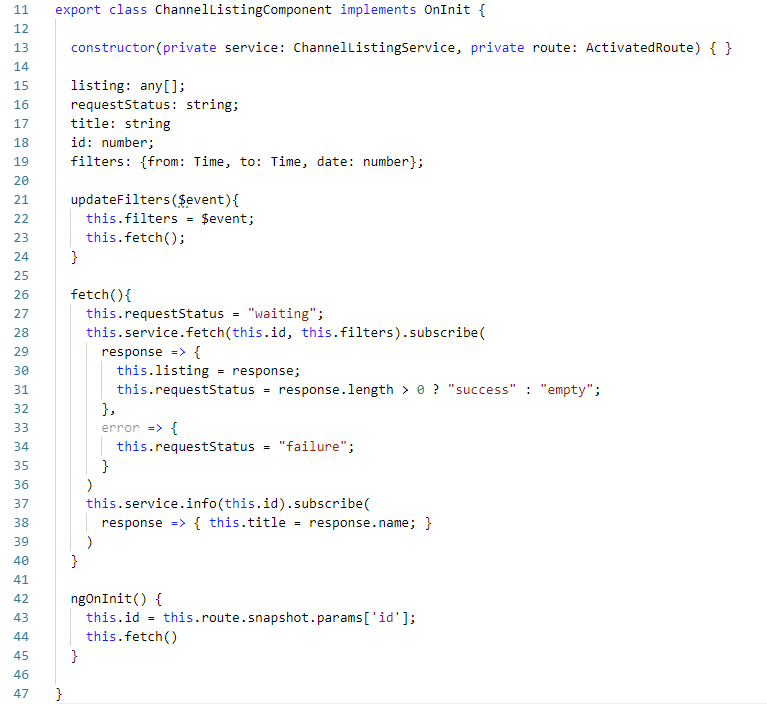
Układ elementów zdefiniowano w pliku channel-listing.component.html



W kolumnie po lewej stronie umieszczono filtry, po prawej widać listing programów, jeżeli są dostępne lub odpowiednią informację o błędzie w przeciwnym wypadku

Ponieważ ten widok nie korzysta z żadnych właściwych sobie styli, wszystkie są dostarczone przez zewnętrzną paczkę *bootstrap*, plik .scss jest pusty.

Plik *channel-listing.component.ts* zawiera logikę związaną z wyświetlaniem informacji na ekranie.



W tym przypadku jest to wywołanie żądania do API w momencie otwarcia i po każdej zmianie filtrów (za samo żądanie odpowiada serwis *channel-listing.service.ts*), oraz ustawienie odpowiedniej flagi w zależności od statusu żądania (czy się powiodło, czy oczekujemy na informację etc.).

Implementacja serwisu korzysta z angularowego klienta HTTP. Żądanie zawiera 3 główne elementy, w tym przypadku:

* Numer kanału
* Ustawione przez użytkownika filtry
* Nazwa użytkownika, jeżeli jest dostępna – pochodzi ze współdzielonego między widokami serwisu przechowującego dane użytkownika



Elementy współdzielone są zdefiniowane w folderze *shared*, należą do nich serwisy odpowiedzialne za funkcjonalność taką jak:

* Wyświetlanie powiadomień push
* Przechowywanie obecnego użytkownika

Poza tym, do współdzielonych elementów zaliczają się szablony elementów widoku, czyli:

* Szablon pozycji na liście programów
* Boczne menu pozwalające na określenie filtrów
* Pasek nawigacyjny

# Testy

## Jednostkowe serwera – xUnit

## Jednostkowe klienta – Jasmine

## Automatyczne end-to-end – Protractor

# Instalacja

## Założenia

Aplikacja korzysta z platformy Docker ([*https://docs.docker.com/*](https://docs.docker.com/)), co pozwala na zautomatyzowanie procesu instalacji i gwarantuje, że aplikacja będzie działała na dowolnym serwerze tak samo jak na deweloperskim. Docker pozwala stworzyć obraz środowiska, w którym działa aplikacja, środowisko zawiera system operacyjny i wszystkie potrzebne zależności (biblioteki, pakiety etc.), nie wymagając jednocześnie emulacji sprzętu (jak w przypadku maszyn wirtualnych). Zawartość każdego kontenera i kolejne czynności podczas jego budowy są określone w pliku Dockerfile.

Gotowe obrazy są przechowywane w rejestrze obrazów dockerowych, w tym przypadku publicznie dostępnym [*https://hub.docker.com/*](https://hub.docker.com/) (istnieje możliwość uruchamiania rejestru na prywatnym serwerze, ale w tym przypadku nie widzę takiej potrzeby). Kompletna procedura wdrożenia budowanej w ten sposób aplikacji składa się z czterech kroków:

1. Budowa obrazów – wywołanie na stacji deweloperskiej komendy   
   *docker-compose build*
2. Publikacja obrazów do rejestru – również ze stacji deweloperskiej, komendą   
   *docker-compose push*
3. Pobranie pliku konfiguracyjnego *docker-compose.yml* na serwer produkcyjny (przechowywany razem z kodem źródłowym aplikacji w repozytorium Git)
4. Pobranie obrazów i uruchomienie na serwerze produkcyjnym – wywołaniem komendy  
   *docker-compose up*

## Zastosowanie

* + 1. Serwer

Aplikacja została stworzona w technologii .NET Core, ponieważ pozwala to, w przeciwieństwie do poprzedniej platformy Microsoftu, .NET Framework, na uruchamianie jej w systemie Linux, a zatem również wewnątrz obrazów platformy Docker.

Kontener zawierający część serwerową jest opisany w następujący sposób:



Dockerfile opisujący część serwerową jest dwuetapowy

* pierwszy etap korzysta z obrazu *mcr.microsoft.com/dotnet/core/sdk:3.0-bionic* - opartego na Ubuntu i zawierającego wszystkie zależności, potrzebne do zbudowania aplikacji .NET Core
* drugi etap korzysta z innego obrazu, *mcr.microsoft.com/dotnet/core/aspnet:3.0-bionic* - również opartego na Ubuntu, ale zawierającego tylko te zależności, które potrzebne są do uruchomienia gotowej aplikacji
  + 1. Klient

Artefaktem powstałym w wyniku kompilacji kodu Angulara (podobnie jak Reacta, Vue czy innych front-endowych frameworków) jest zestaw plików html, css i js (oraz dodatkowych zasobów - ikon, dźwięków lub plików konfiguracyjnych). Te pliki muszą być obsłużone po stronie serwera i przesłane przeglądarce użytkownika na żądanie. Obecnie najpopularniejszym serwerem do obsługi statycznych plików jest NGINX, z którego korzysta kontener obsługujący część kliencką.

Kontener zawierający część kliencką jest opisany w następujący sposób:



Dockerfile opisujący aplikację kliencką również jest dwuetapowy

* pierwsza część zawiera środowisko node.js i służy do pobrania wszystkich zależności i zbudowania docelowych artefaktów
* druga część to sam serwer NGINX z odpowiednią konfiguracją i plikami do obsługi

## Komunikacja

Zgodnie z dobrymi praktykami, kontener z aplikacją powinien zawierać tylko i wyłącznie tą jedną aplikację, jeżeli wymagane są inne procesy, powinny one działać w osobnych kontenerach.

Za budowę i uruchamianie odpowiednich kontenerów odpowiada usługa docker-compose będąca częścią całej platformy Docker. Pozwala ona zdefiniować plik konfiguracyjny YAML, określający w jaki sposób mają być budowane poszczególne kontenery, w jaki sposób mają być uruchamiane i jak powinny się ze sobą komunikować. Na opisywaną aplikację składają się 4 kontenery, wszystkie zdefiniowane w pliku   
*docker-compose.yml*

Poza dwoma opisywanymi wcześniej obrazami server i client, aplikacja korzysta z obrazu zawierającego serwer bazy danych MS-SQL oraz opracowany przez grupę linuxserver.io, obraz zawierający program pobierający programy telewizyjne, WebGrab+.

# Podsumowanie

# Bibliografia

**Brak źródeł w bieżącym dokumencie.**