INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Sistema de Informação de Televisão: Pesquisa e Recomendações

TIAGO FILIPE MOREIRA DA COSTA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de Informação de Televisão: Pesquisa e Recomendações

Tiago Filipe Moreira Costa

Dissertação para obtenção do grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia do Software

Orientadora: Isabel de Fátima Silva Azevedo

Supervisor: Diogo André Alves Ribeiro Marques

Porto, outubro 2017

Resumo

A InfoPortugal possui dados sobre programação da emissão nacional, que são utilizados em três das quatro principais operadoras nacionais. De forma a melhorar não só, o relacionamento com estes parceiros, mas também para rentabilizar a informação de programação de televisão, optou-se pelo desenvolvimento de uma plataforma que esteja disponível a qualquer tipo de dispositivo, que forneça não só os dados, como também funcionalidade de recomendações personalizadas por utilizador. Trata-se de um sistema único no mercado português e com grande potencial de negócio.

A plataforma REST desenvolvida está disponível para qualquer tipo de dispositivo com acesso a rede. Utilizando um sistema de pesquisa comprovado para o módulo de pesquisa da plataforma, consegue-se obter um motor de pesquisa robusto e rápido. As recomendações personalizadas são geradas usando um algoritmo hibrido que tem em consideração a semelhança entre utilizadores e atributos dos programas.

Para avaliar que as recomendações atingiram a meta de precisão desejada de 15%, foi utilizado para o cálculo uma base de dados externa de interações e também a base de dados de interações no final do projeto. Para avaliar a precisão da funcionalidade de pesquisa são utilizadas expressões comuns em plataformas distintas.

Prevê-se que esta plataforma seja adquirida por uma das grandes operadoras e passe a estar disponível a um grande número de utilizadores brevemente. O alargamento das funcionalidades vai surgir de forma natural, que permitirá a evolução da aplicação através do *feedback* recebido pelo cliente.

Palavras-chave: Programação de televisão, REST, pesquisa, recomendações, híbrido, precisão

Abstract

InfoPortugal has all the data of the Portuguese television programming guide, which is delivered to three of the four main national operators. It was decided to develop a platform that is available in any device, not only improve the relationship with these partners, but also to monetize the television programming data. This platform would also offer a search engine and a personalized recommendation system. This type of application is unique on the national market and has great business potential.

The REST platform is available to any type of device with internet access. Using an off-the-self solution for the search module of the platform, it's possible to use a robust and fast search engine. The personalized recommendations are generated using a hybrid algorithm, that uses the similarity between users and programs features.

An external database with users' reviews and the final interactions database of this project were used to evaluate the recommendations, by calculating if the precision of the algorithm achieves the 15% goal. Same expressions that are common between two different platforms are going to be used to test the accuracy of the search functionality.

It is predicted that this platform is going to be acquired by one of the major operators and it will be available to a huge number of users soon. The extension of new functionalities is going to be a natural process, which will allow the evolution of the application through the feedback sent by the client.

Keywords: television programming, REST, search, recommendations, hybrid, precision

Agradecimentos

Este projeto reflete todo o conhecimento obtido durante os cinco anos de estudo, concluindo a minha etapa académica e uma fase importante da minha vida pessoal. A todas as pessoas que se disponibilizaram a ajudar e que contribuíram, direta ou indiretamente, para este projeto, fico agradecido.

Um obrigado especial para toda a minha família e namorada, por todo o apoio que me têm dado ao longo da vida, e por estarem sempre presentes nos obstáculos que enfrentei até hoje.

Um agradecimento à InfoPortugal, em especial para o Diogo Marques e Nuno Lopes, por todo o apoio, ajuda e conhecimento adquirido ao longo deste projeto. O projeto não é só importante a nível pessoal, mas também profissional, e agradeço o facto de a empresa ter confiado em mim para um trabalho tão ambicioso.

Agradeço à minha orientadora, Isabel de Fátima Silva Azevedo, pela sua ajuda ao longo deste projeto, assim como pela sua contribuição e sugestões, que auxiliaram a evolução e sucesso do mesmo.

À instituição de ensino, um obrigado por todo o conteúdo e técnicas lecionadas, que fizeram com que a transição para o mercado do trabalho fosse suave, e que me preparou para os restantes desafios do futuro.

Por último, um obrigado a todos que contribuíram para este projeto, com as suas sugestões ou até interações com o mesmo.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

Lista de Acrónimos

API	*Application Programming Interface*
CB	*Content Based*
CF	*Collaborative Filtering*
EPG	*Electronic Program Guide*
ES	ElasticSearch
HAL	Hypertext Application Language
JSON	*JavaScript Object Notation*
REST	*Representational State Transfer*

1 Introdução

Neste capítulo de introdução é feito o enquadramento do projeto, explicando de onde surgiu o problema, assim como quais os objetivos a cumprir. É ainda especificada a abordagem metodológica utilizada ao longo do desenvolvimento do projeto, assim como a estrutura deste documento.

1.1 Enquadramento

É importante que as empresas utilizem os seus recursos da melhor e mais eficiente forma possível.

Um recurso ainda bastante inexplorado pela InfoPortugal é o *Electronic Program Guide* (EPG), uma vez que possui uma grande quantidade de dados úteis a vários tipos de utilizadores e está neste momento disponibilizado para diversas operadoras (nacionais e internacionais). A InfoPortugal disponibiliza, neste momento, a informação da programação de televisão nacional para três das quatro principais operadoras do mercado, possuindo mais de metade do mercado português. A acrescer está a boa e longa relação que tem com uma dessas operadoras (Vodafone), ao qual fornece o serviço de programação há cerca de nove anos. Com a evolução do projeto foram sendo incorporados mais dados, como a pontuação disponibilizada pela plataforma Internet Movie Database (IMDb)[1] para vários conteúdos televisivos.

Em conjunto com a grande quantidade de dados que não estão a ser rentabilizados, não existe, neste momento, um serviço onde seja possível pesquisar e ser recomendado sobre possíveis programas do agrado do utilizador, com informação detalhada da programação nacional. Com a identificação dessa oportunidade de negócio e os serviços que podem ser disponibilizados, optou-se pela criação de uma aplicação de pesquisa e disponibilização de todo esse conteúdo,

[1] http://www.imdb.com/

assim como um sistema de recomendações personalizadas para utilizador. Esta aplicação seria depois utilizada pelas operadoras de televisão para utilizarem estas funcionalidades nas *boxes* de televisão.

1.2 Objetivos

Pretende-se realizar um estudo na área de domínio e explorar algoritmos de recomendação já desenvolvidos para problemas similares e avaliá-los, nomeadamente em relação à viabilidade e à adequação dos mesmos para o sistema pretendido; encontrar as melhores ferramentas e tecnologias a utilizar para desenvolver uma aplicação que abranja sistemas como *boxes* de televisão, aplicações *web* e *mobile*; propor e contruir uma solução onde seja possível pesquisar sobre conteúdo emitido em televisão e um sistema de recomendação personalizado; avaliar a precisão do sistema de recomendação e do motor de pesquisa.

Resumindo o objetivo é o desenvolvimento de uma plataforma, em que qualquer tipo de sistema se possa comunicar com a mesma, com o intuito de obter informação acerca da programação de televisão em Portugal, assim como recomendações personalizadas.

Na definição do projeto por parte da InfoPortugal foram delimitadas quais as tecnologias que seriam utilizadas na implementação da solução. Essas tecnologias são as utilizadas atualmente na empresa. Tratam-se de tecnologias que são atuais, com as quais a equipa de desenvolvimento está especializada. As principais tecnologias são enunciadas na Tabela 1.

Tabela 1 - Tecnologias utilizadas

Tecnologia	**Descrição**
ElasticSearch	Motor de Pesquisa
PostgreSQL	Base de Dados
Django	*Framework* de Python para *Web Development*
Python	Linguagem de programação
REST	Protocolo de comunicação

1.3 Abordagem Metodológica

O projeto desenvolvido foi dividido em três grandes fases: estudo do problema, conceitos e tecnologia a utilizar; desenvolvimento da aplicação; e avaliação do trabalho desenvolvido.

Numa primeira fase foram estudados vários algoritmos de recomendação, e como implementar um motor de pesquisa para uma grande quantidade de informação que pode ser filtrada. Tendo em consideração os conceitos adquiridos durante a pesquisa, foi feita uma investigação com o intuito de encontrar as ferramentas ideais para o desenvolvimento do projeto. Numa fase final,

foi feita a avaliação e experimentação do trabalho realizado, para comparar com as metas definidas.

1.4 Convenções de Escrita

Para a elaboração deste documento foi utilizado o tipo de letra "Calibri" com tamanho 11.

Neste documento todas as palavras em inglês são colocadas em itálico, para uma fácil diferenciação de línguas.

Todas as legendas de imagens, assim como as próprias imagens estão alinhadas ao centro, para um fluxo de leitura mais suave. No entanto, as legendas de imagens com mais de uma linha estão justificadas à esquerda. A legenda das imagens também tem um tamanho mais pequeno (10) em relação ao corpo do texto.

1.5 Estrutura

Este documento está estruturado em nove capítulos.

Neste capítulo é feita uma pequena introdução ao problema juntamente com as metas a cumprir, assim como uma pequena introdução teórica aos temas abordados durante este projeto.

No segundo capítulo (*Estado de Arte*) é feita uma introdução ao conceito de EPG, qual o valor e a importância que tem atualmente, o seu potencial, quais os seus concorrentes diretos, e também o enquadramento da área tecnológica com algoritmos de recomendações e os diferentes tipos de bases de dados.

No terceiro capítulo é feita uma análise de valor ao projeto, identificando o que traz valor a este tipo de produto, que faça com que haja interesse no mercado atual.

No quarto capítulo (*Valor, ameaças e oportunidades*) é feita uma análise Forças/Fraquezas e Ameaças/Oportunidades (também conhecida como SWOT) ao EPG, e consequentemente a este projeto, demonstrando como esta solução se encaixaria no mercado atual.

No capítulo *Análise de Negócio*, vão ser apresentados e analisados todos os requisitos funcionais e não-funcionais do projeto, assim como a apresentação do modelo de domínio do projeto.

O sexto capítulo (*Arquitetura da Solução*) apresenta a arquitetura geral do sistema, quais os padrões que foram seguidos para darem origem a essa arquitetura, assim como os pressupostos e decisões tomadas na construção da mesma.

No *Desenho e Implementação da Solução* é explicada detalhadamente a solução encontrada e a desenvolver, convenientemente justificada, e todos os detalhes técnicos específicos e mais importantes para o desenvolvimento da mesma.

Na *Avaliação da solução* são aplicadas várias técnicas de avaliação para perceber quais os objetivos que foram cumpridos.

Por último, no capítulo da *Conclusão* é documentado todo o trabalho desenvolvido, assim como possível trabalho futuro para dar continuidade ao projeto, ou possíveis melhorias a aplicar.

2 Estado de Arte

Este capítulo tem como intuito o levantamento do estado de arte, a explicação dos conceitos de programação de televisão, a avaliação a algumas das soluções já existentes e também uma introdução aos algoritmos de recomendação existentes. São também destacadas as tecnologias utilizadas no sistema, e possíveis alternativas a estas.

2.1 Enquadramento teórico

Neste capítulo vai ser aprofundada toda a matéria teórica que é necessária para a elaboração do projeto, assim como entender partes da área de negócio.

2.1.1 Electronic Program Guide

O EPG é um sistema gráfico que apresenta todos os dados de programação televisiva. Esta aplicação encontra-se na maior parte das vezes nas *boxes* disponibilizadas pela operadora. A sua principal funcionalidade é apresentar ao espectador quando e onde vai ser emitido certo programa televisivo. No entanto também deve ser capaz de apresentar detalhes desse programa, com uma descrição do que de facto vai ser emitido. Um exemplo do EPG é representado na Ilustração 1, onde é possível ver o guia de TV de uma das operadoras nacionais.

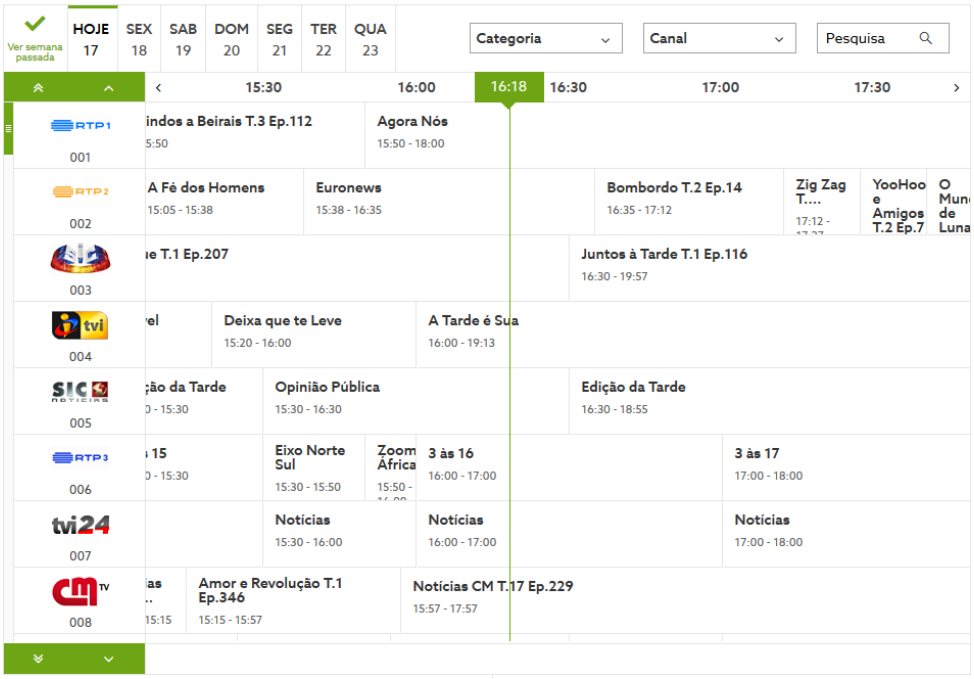

Ilustração 1 - Guia de TV da NOS

Com a evolução da tecnologia, houve um acompanhamento pelo EPG: cada vez mais os programas são mais detalhados em termos de informação, tendo acesso a todo o elenco, ano de produção, local de produção, entre outros; existe apoio visual do programa através da disponibilização de várias imagens e/ou *trailers* que torna a experiência mais apelativa ao utilizador e também simples e intuitiva; as opções de gravação de programas estão cada vez mais avançadas, assim como as suas restrições mais específicas.

O conceito de programa televisivo é um pouco ambíguo uma vez que um filme e um episódio são programas, mas, no entanto, não são o mesmo tipo de *media*. Neste momento um programa televisivo está categorizado como: filme, geralmente conta uma história com início, meio e fim (exemplo: Interstellar, Titanic, Jurassic Park); programa genérico, que pode ser emitido mais do que uma vez, mas não é utilizado para contar uma história nem é subdivido em episódios (exemplo: Noticiários, Galas, programas da tarde, eventos desportivos); episódio, que tem uma sequência e uma história que vai evoluindo ao longo dos vários episódios. Dentro dos episódios ainda existe dois tipos de programas: séries, que contêm temporadas e os episódios estão divididos por temporadas; *TvShow*, que são conteúdos que estão divididos por episódios, mas não têm uma temporada atribuída como por exemplo, telenovelas.

No início deste projeto o EPG feito pela InfoPortugal abrangia 60,4% do mercado nacional, tendo como clientes as operadoras NOS, Vodafone e Cabovisão[2]. De momento a NOS já não

[2] Dados retirados da ANACOM no dia 17 de fev. de 2017 (https://www.anacom.pt/render.jsp?contentId=1400523)

utiliza o serviço de EPG, mas continua a utilizar a meta-informação da base de dados do EPG para detalhes de elencos.

2.1.2 Sistemas de recomendação

Existem várias aplicações que utilizam sistemas de recomendação testados e aprovados. Neste capítulo vai ser explicada a base dos algoritmos de recomendação e as semelhanças entre eles, assim como quais os algoritmos de recomendação mais comuns e em que tipo de casos estes devem ser utilizados.

Os sistemas de recomendação são ferramentas de *software* e técnicas utilizadas para fornecer sugestões de itens a certos utilizadores. As sugestões são dadas de acordo com processos de decisão, como músicas para ouvir, que produtos comprar, ou que artigos ler. "Item" é o termo utilizado pelos sistemas para o que é sugerido para os utilizadores (Ricci, Rokach, Shapira, & Kantor).

Os sistemas informáticos atuais possuem um problema conhecido como *The long tail* (Leskovec, Rajaraman, & Ullman, 2014), onde existe uma grande quantidade de artigos, produtos, programas na aplicação e que para ser rentabilizada deve ser apresentada ao utilizador, para gerar interesse. As recomendações são vistas atualmente em três grandes áreas:

- Venda de produtos, onde são recomendados produtos com características semelhantes, ou compras feitas por utilizadores semelhantes. Um *website* como a Amazon, que possui milhões de produtos, consegue assim rentabilizar grande parte dos seus produtos através do sistema de recomendação, uma vez que existe mais oferta do que procura;

- Recomendação de artigos, que ao contrário de jornais e revistas onde o conteúdo não é atualizado frequentemente, um *website* para além de possuir mais artigos, possui também os artigos mais recentes, por isso é necessário captar os gostos do utilizador e personalizar a listagem de notícias;

- Recomendação de programas, onde uma aplicação como o Netflix, que tem uma biblioteca enorme de filmes e séries e quer cativar o utilizador ao serviço, deve recomendar o conteúdo que possui para rentabilizá-lo.

Existem várias funções que favorecem uma aplicação, caso um sistema de recomendação seja adicionado: (Ricci, Rokach, Shapira, & Kantor):

- Aumentar o número de produtos vendidos, uma vez que são sugeridos produtos que serão do interesse do utilizador;

- Vender produtos mais diversos, em que o sistema de recomendação é capaz de recomendar produtos menos populares a utilizadores que encaixam no perfil desses;

- Aumentar a satisfação do utilizador, ao apresentar sugestões que sejam relevantes e interessantes para o mesmo, que faz com que este aprecie mais o sistema;

- Aumentar a fidelidade do utilizador, em que sendo as recomendações precisas, o utilizador começa a dar mais valor ao sistema, tornando-se cada vez mais leal à aplicação.

Tendo as funções de um sistema de recomendação em conta, deve-se ter em consideração uma outra variável que é o contexto do utilizador. No caso particular de produto é importante contextualizar a sugestão do artigo com, por exemplo, a época do ano. Ou seja, na época do verão não devem ser recomendados aos utilizadores cachecóis. Esta utilização de contexto deve ser tida em conta no cálculo da sugestão de um item para um certo utilizador, oferecendo uma maior precisão e tendo em conta as necessidades do utilizador no contexto em que se insere (Adomavicius & Tuzhilin, 2014).

Para o desenvolvimento de recomendações personalizadas é necessário conhecer o utilizador. Numa aplicação existem várias formas de conhecer os gostos dos utilizadores, através das suas ações. As suas ações dividem-se em dois grupos:

- *Feedback* implícito, em que o sistema capta ações do utilizador através dos cliques, posicionamento do cursor e das suas pesquisas para captar os seus gostos;

- *Feedback* explícito, em que o utilizador dá explicitamente a sua opinião sobre um conteúdo da aplicação, como por exemplo a avaliação de um filme ou partilhar um artigo nas redes sociais.

Normalmente os utilizadores quando estão a utilizar a aplicação não possuem o hábito de avaliar o conteúdo que estão a ver (como, por exemplo, classificar um filme), e por isso torna-se complicado utilizar apenas informação que tem de ser dada de livre vontade pelo utilizador. O *feedback* implícito consegue colmatar essas falhas, no entanto tem algumas desvantagens associadas: de poder influenciar os resultados quando não devia, uma vez que o utilizador pode estar a visualizar a ficha de detalhe de um filme, mas pode não gostar desse filme; tem-se acesso apenas a uma quantidade limitada de informação (Chen, Wang, Wang, & Yu, 2016). É necessário então existir um equilíbrio dessa informação retirada, para consequentemente, ser utilizada no algoritmo de recomendação para gerar sugestões que façam sentido para o utilizador.

Os mais reconhecidos algoritmos de recomendação utilizados nas aplicações são os “Non-personalized recommenders”, “Collaborative filtering” (CF) e o “Content based recommenders” (CB). Cada um tem as suas vantagens e desvantagens que irão ser detalhadas e explicadas nos próximos capítulos.

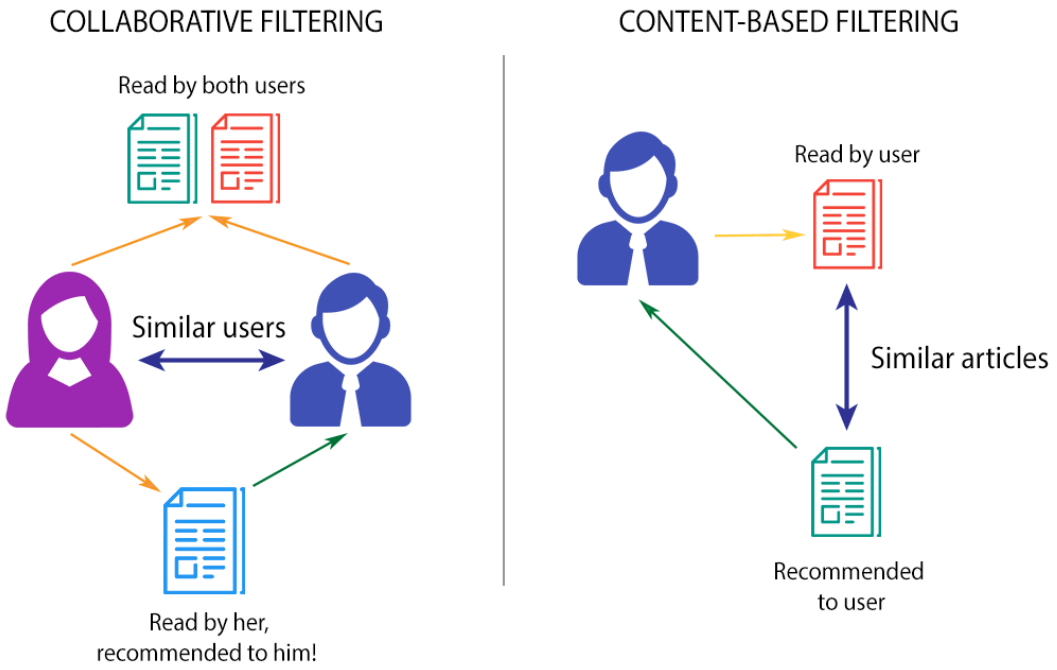

Ilustração 2 - Simples explicação do CF e CB (Fonte: https://www.marutitech.com/wp-content/uploads/2017/05/Collaborative-Filtering-min.png).

2.1.2.1 Non-personalized recommenders

Os algoritmos não personalizados não dependem das preferências dos utilizadores para gerar as recomendações, baseando-se somente no item. Um exemplo deste tipo de algoritmos é a listagem dos produtos mais vendidos de um *website*, ou o filme mais popular neste momento. Este tipo de recomendação é visto, por exemplo, no detalhe do filme no IMDb, onde as recomendações são transversais a todos os utilizadores.

Existe ainda outro tipo de algoritmo não personalizável com recomendações feitas por um grupo restrito de pessoas. Estas recomendações são normalmente feitas por uma equipa especializada nesse tipo de conteúdo, e não dependem em nada do utilizador atual. Este tipo de recomendações é muitas vezes visto em aplicações de artigos gastronómicos, com alguns restaurantes sugeridos pela equipa editorial.

Estas recomendações, embora sejam interessantes, não são o que é pedido para este tipo de projeto, uma vez que o foco deste projeto são as sugestões personalizadas, em que a opinião dada pelo utilizador deve afetar as recomendações apresentadas.

Estes algoritmos têm o mérito de serem a base de muitos algoritmos mais complexos e personalizados.

2.1.2.2 Content based recommenders

Este tipo de algoritmo utiliza os atributos dos conteúdos que vão ser recomendados. Ou seja, um utilizador vai dando *feedback* sobre o conteúdo (por exemplo um filme), onde ele avalia-o, marca-o como visto, visualiza a ficha de detalhe, e utiliza esses dados do utilizador para gerar recomendações de filmes, baseando-se nos atributos dos filmes ao qual o utilizador deu *feedback*. Ou seja, quanto mais *input* for dado pelo utilizador, mais aprimoradas vão ser as recomendações. Este tipo de algoritmo é sugerido para sistemas que tenham uma grande quantidade de informação, mas que possuam poucos utilizadores (Leskovec, Rajaraman, & Ullman, 2014).

Este tipo de sistema de recomendação é uma boa hipótese para a aplicação a desenvolver uma vez que existe neste momento uma base de dados com uma grande quantidade de informação, mas não existe massa crítica que seria necessária para outros sistemas de recomendações como por exemplo o CF. No entanto, por não ter em conta utilizadores semelhantes, as recomendações acabam por perder um dinamismo interessante, e acabam por surgir recomendações sempre em torno do mesmo estilo, uma vez que são só utilizados os atributos dos conteúdos como a categoria e o elenco de um filme. Outro grande problema associado a recomendações utilizando apenas os atributos é que se perde a componente de popularidade do item, uma vez que a popularidade do item não é um atributo associado ao programa.

Para calcular se um utilizador gosta de um certo artigo, produto ou filme é necessário seguir uma sequência de cálculos que permite medir a semelhança entre os gostos do utilizador com o perfil do item.

Numa primeira fase deve-se traçar o perfil de um artigo, mas antes escolhe-se quais serão os atributos desse item que irão ser utilizados para traçar o seu perfil. Por exemplo, quando é escolhido um filme é possível utilizar atributos como o ano de lançamento, elenco do filme, realizador, género, entre outros. Uma vez selecionados todos os atributos que serão utilizados para calcular a semelhança, é traçado o perfil do programa através dos valores desses atributos. Ou seja, considerando dois filmes que têm três atores em comum a sua representação pode ser vista na Tabela 2:

Tabela 2 - Perfil entre dois filmes com três atores em comum

1	0	1	1	0	1	1	0
1	1	1	1	1	0	0	0

Como possuir um ator no elenco trata-se de um elemento *boolean* (ou possui o ator ou não, ou seja, ou é 0 ou 1), consegue-se facilmente traçar semelhanças entre estes dois elementos. No entanto, podem existir atributos numéricos como é o caso do ano do programa ou a sua classificação média. Para estas situações deve ser utilizado o valor real do atributo (e.g. ano

seria 2017) e serão utilizados para calcular a semelhança da mesma forma que os valores booleanos, ou seja, uma única parcela do vetor. É necessário garantir que o seu valor não influencia em demasia o cálculo ou não é considerado o suficiente. Existem várias técnicas a adaptar nestes casos e estes valores devem ser modificados e afinados com o tempo, até se atingir os valores desejáveis (ou seja, boas comparações entre itens) (Leskovec, Rajaraman, & Ullman, 2014).

Tabela 3 - Perfil de dois filmes com atributos numéricos

1	0	1	1	0	1	1	0	7α
1	1	1	1	1	0	0	0	3α

Para calcular a semelhança entre dois filmes, e utilizando os valores indicados na Tabela 3, é apenas efetuado o cálculo de cosseno entre dois vetores. A fórmula para o cálculo entre o cosseno de dois vetores é dada pela divisão entre o produto interno de dois vetores (<v,u>) e multiplicação da norma de dois vetores (Leskovec, Rajaraman, & Ullman, 2014).

$$\cos(\theta) = \frac{< v, u >}{|v| \times |u|}$$

Utilizando o exemplo (Tabela 3) anterior é possível chegar ao cálculo de:

$$\cos(\theta) = \frac{3 + 21\alpha^2}{\sqrt{25 + 290\alpha^2 + 441\alpha^4}}$$

Ajustando agora os valores de α seria possível obter o ângulo entre os dois vetores e descobrir a sua semelhança através disso. Quanto mais pequeno for o ângulo, maior seria a sua semelhança.

Sendo agora possível arranjar uma comparação entre dois programas, falta considerar os gostos do utilizador. Para isso é necessário também traçar um perfil de utilizador, considerando o *feedback* dado por ele. Por exemplo tendo em conta que o utilizador gostou de cinco filmes em 20 em que o ator Bryan Cranston está no elenco, como o atributo de ator no filme é booleano, o peso do vetor do utilizador para a componente do Bryan Cranston iria ser de 25%. No entanto, a maioria do *feedback* dado pelo utilizador em relação a um filme é feito de forma numérica, através da classificação do filme. Para esses casos deve-se ter em conta a média de análises do utilizador e subtrair a cada uma das suas classificações para saber o que são de facto avaliações negativas para o utilizador e as que são positivas (Leskovec, Rajaraman, & Ullman, 2014).

Tendo então acesso a ambos os vetores de utilizador e de produtos, torna-se possível fazer o cálculo do cosseno entre estes dois vetores para medir a semelhança entre eles e ordenar os itens de forma a que as melhores recomendações fiquem em primeiro.

Este trabalho de classificação de recomendações pode também ser feito pelo sistema de aprendizagem automática. Aplicando esta técnica faz com que o próprio algoritmo de recomendação vá modificando e aprendendo o que é uma boa recomendação e uma má recomendação, modificando os pesos de cada atributo considerado para os cálculos. Uma das técnicas mais utilizadas neste caso é o da árvore de decisão. No entanto devido à sua complexidade, este tipo de técnica só é recomendado em problemas com uma pequena quantidade de dados uma vez que é necessário gerar uma árvore de decisão para cada utilizador, o que seria bastante dispendioso (Leskovec, Rajaraman, & Ullman, 2014).

2.1.2.3 Collaborative Filtering

Ao invés do algoritmo de recomendação CB, este algoritmo gera recomendações usando utilizadores semelhantes. Ou seja, em vez de captar objetos da base de dados com atributos semelhantes, são captados utilizadores com gostos semelhantes e geradas recomendações a partir desses utilizadores semelhantes, sendo o foco o *feedback* dado por outros utilizadores aos objetos em comparação com o utilizador para quem se está a gerar recomendações. Um exemplo bastante conhecido é o utilizado pela Amazon. Na página de um certo artigo a Amazon lista no final uma lista de artigos com o título "Os utilizadores que compraram produto X também compraram..." (Leskovec, Rajaraman, & Ullman, 2014).

Este tipo de algoritmos geram recomendações bastante viáveis, uma vez que utilizam o gosto pessoal de utilizadores e uma componente humana que é capaz de seguir tendências, ao invés do algoritmo CB que utiliza apenas a meta-data existente. No entanto, este tipo de algoritmo só tem grande utilidade se possuir uma aplicação com uma grande quantidade de utilizadores que dá o seu *feedback*. Outro grande problema deste tipo de algoritmo é que não consegue recomendar itens que ainda não tiveram qualquer tipo de interação, e por isso existe um problema de *cold-start*, onde apenas os artigos com *feedback* têm a oportunidade de serem recomendados.

Tal como no algoritmo CB, este algoritmo também é caracterizado por perfis de utilizadores, numa simples matriz, em que as linhas correspondem aos utilizadores e as colunas aos programas vistos por estes. Cada elemento da matriz representa a interação entre o utilizador e esse *item*. Estas matrizes normalmente possuem um grande número de utilizadores por um grande número de itens, em que grande parte não possui interações. Estas matrizes têm o nome de matriz esparsa, e são caracterizadas pela grande quantidade de zeros que possuem. Utilizando um exemplo simples, o utilizador pode apenas classificar os programas como "gosto" e "não gosto", sendo que o "gosto" é classificado positivamente e o "não-gosto" negativamente. Neste exemplo não é utilizada uma classificação de zero a dez, ou seja, não é necessário normalizar as interações dos utilizadores, de forma a identificar classificações positivas e

negativas.
Tendo acesso a todos as interações, e tendo em conta que estas já estão normalizadas, é necessário aplicar novamente a forma do ângulo do cosseno, para saber a semelhança entre dois utilizadores. Relembrar que quanto mais pequeno for o valor do ângulo, mais semelhantes são estes utilizadores.

No caso do sistema de classificação utilizar uma classificação numérica (por exemplo, de zero a dez), uma possível abordagem é a normalização dos dados. Para um utilizador (que pode ser representado por um vetor) deve-se achar a média das suas classificações. Calculando a média, subtrai-se esse valor a cada uma das suas classificações, tendo assim a normalização de todas as suas interações. Desta forma é fácil de identificar quais as classificações negativas e positivas para aquele utilizador. Com esta abordagem também se normaliza o tipo de utilizador, uma vez que existem utilizadores em que a sua nota média é mais alta do que outros, e através deste cálculo, tem-se apenas classificações negativas, positivas ou indiferentes. Uma vez normalizados os dados de todos os utilizadores, utiliza-se novamente a fórmula do cálculo do cosseno. Esta abordagem tem o nome de coeficiente de correlação de Pearson.

Existe ainda duas formas de utilizar o algoritmo CF: item-item ou user-user.

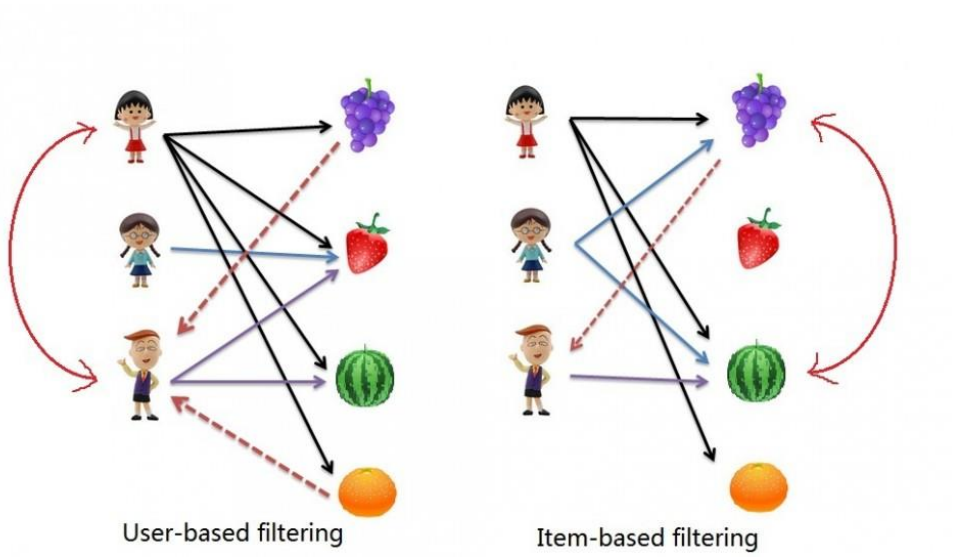

Ilustração 3 - Exemplo de User-User CF e Item-Item CF (Fonte: http://www.salemmarafi.com/code/collaborative-filtering-with-python/)

2.1.2.3.1 Item-item CF

Esta forma de avaliação é usada para recomendar *itens* semelhantes entre si, em vez de usar utilizadores semelhantes. Ou seja, cada item tem a si atribuído um conjunto de classificações dadas por utilizadores do sistema, e são essas classificações usadas para traçar o perfil do item e encontrar itens semelhantes, através dos cálculos já referidos neste documento (Leskovec,

Rajaraman, & Ullman, 2014). Este tipo de classificação traz vantagens em relação ao user-user, uma vez que existe um maior número de classificações sobre itens e torna as recomendações mais precisas para um utilizador que tem menos classificações, que gosta do artigo Y, este que tem um conjunto *n* de artigos semelhantes a este. Este mesmo utilizador no user-user teria recomendações menos precisas, uma vez que o número reduzido de classificações prejudica a qualidade das recomendações porque é um processo complexo encontrar um utilizador semelhante a este. A escalabilidade desta forma também é mais simples comparada ao user-user, uma vez que só se preocupa com a subida do número de itens, enquanto que o user-user preocupa-se com o número de artigos e número de utilizadores (Sarwar, Karypis, Konstan, & Riedl, 2001).

2.1.2.3.2 User-user CF

O User-user CF utiliza técnicas de *nearest-neighbour* para saber quais os utilizadores mais semelhantes ao atual, e gerar recomendações sobre artigos, com base nos gostos destes utilizadores. Neste tipo de algoritmo, numa primeira fase são calculados *n* utilizadores idênticos ao utilizador atual, e a partir da semelhança entre os utilizadores, captar os itens dos utilizadores mais parecidos com o qual o utilizador atual ainda não interagiu.

2.1.2.4 Algoritmos híbridos

Tanto o CB como o CF têm as suas desvantagens: no primeiro as recomendações começam a ficar filtradas pelo tipo de conteúdo que o utilizador gosta. Ou seja, se um utilizador der *feedback* positivo sobre filmes de ação, vai receber apenas recomendações de outros filmes de ação. No entanto, pode haver filmes que não sejam de ação que o utilizador também goste. O CF tem o problema do *cold-start*, em que é necessária uma grande quantidade de utilizadores com uma grande quantidade de interação para encontrar utilizadores semelhantes. Ou seja, os primeiros utilizadores do sistema vão ter recomendações pouco precisas porque a probabilidade de encontrar utilizadores semelhantes a ele é menor. Para além disso, um artigo que não tenha qualquer tipo de interação nunca vai ser sugerido pelo sistema, uma vez que o cálculo utilizado no CF conta com utilizadores semelhantes, em que a recomendação é baseada em artigos que foram bem cotados por outros utilizadores.

No entanto, juntando a técnica dos atributos do CB, com a dos utilizadores semelhantes do CF, consegue-se colmatar quase todas as desvantagens encontradas para recomendar algo útil para o utilizador. Utilizando os atributos do item, juntamente com os gostos de outras pessoas, é possível criar um sistema de recomendação dinâmico, que recomenda qualquer tipo de itens.

2.1.3 Recomendações em soluções existentes

Neste momento existem inúmeros produtos no mercado que oferecem recomendações de conteúdos televisivos com uma grande quantidade de utilizadores.

Um dos mais reconhecidos internacionalmente é o Netflix. O Netflix é um serviço que se destaca pelas recomendações personalizadas a cada utilizador. Através do seu algoritmo de recomendação, e toda a meta-data que consegue obter através do *feedback* dado pelo utilizador (quer seja na avaliação do conteúdo, quer seja na visualização total ou parcial de conteúdo) consegue gerar recomendações de conteúdo semelhante e que seja do agrado do utilizador. Para a elaboração deste projeto este sistema não foi avaliado, uma vez que é necessária uma subscrição para aceder à aplicação. No entanto, foram utilizados dados e uma avaliação feita por terceiros para o capítulo de Avaliação da solução, onde foi indicada a precisão das recomendações geradas pela aplicação.

Outro serviço bastante popular é o IMDb. O IMDb está dentro da lista dos 100 *websites* mais visitados no mundo (57 no Alexa (Alexa, s.d.), e 53 no SimilarWeb (SimilarWeb, s.d.)), sendo a aplicação mais popular para informação sobre filmes e séries. O IMDb é o produto mais semelhante ao desenvolvido neste projeto, uma vez que utiliza apenas as análises dadas pelos utilizadores e o conteúdo que tem sobre os programas para gerar recomendações. No entanto, não se sabe publicamente que tipo de algoritmo de recomendações o IMDb está a utilizar. Possui também a base de dados mais rica e atualizada atualmente, sendo o *website* avaliado em três milhares de milhões de dólares[3].

Existem dois tipos de recomendações no IMDb: uma recomendação na página de detalhe de um programa, onde é apresentada uma lista de programas que os utilizadores que gostarem deste programa também gostaram; e uma recomendação personalizada, baseadas nas análises dadas pelo utilizador e pelos programas que este tem adicionados na sua *Wishlist* (filmes/séries que tem intenção de visualizar).

Para a avaliação do sistema de recomendação do IMDb foi criada uma conta de utilizador e avaliadas as recomendações iterativamente. Este estudo foi efetuado apenas com um utilizador, logo os seus resultados podem não ditar com exatidão a precisão do algoritmo utilizado pelo IMDb.

Avaliar se uma recomendação é "correta" é algo trabalhoso para programas de televisão, uma vez que era necessário haver tempo para provar que de facto a recomendação é correta ao visualizar o programa. Uma vez que se está a avaliar uma grande quantidade de programas, não faria sentido uma pessoa ter de visualizar todo o conteúdo para consequentemente avaliar se a recomendação dada foi correta/errada. Tendo isso em conta, é considerada uma recomendação "correta" um programa televisivo que o utilizador tenha visualizado e que seja do seu agrado e uma recomendação "errada" um programa televisivo que o utilizador tenha visualizado e que não lhe tenha agradado.

O estudo começou por avaliar quatro programas (visíveis na Tabela 4) que o utilizador gosta e ver as recomendações dadas na ficha do programa. Estas recomendações são independentes do *feedback* do utilizador atual e não são pessoais, tal como já foi referido neste documento no

[3] Dados retirados de WorthOfWeb no dia 17 de fev. de 2017 (http://www.worthofweb.com/website-value/imdb.com/)

capítulo 2.1.2.1. Estas recomendações têm em conta todo o *input* dado pelos utilizadores, de uma forma: "os utilizadores que gostaram deste programa também gostaram de...". Através da frase anterior, e tendo em conta a pesquisa efetuada, parte-se do pressuposto que é utilizado o algoritmo de CF para a obtenção destas recomendações, mas que não são focadas para o utilizador atual.

Tabela 4 - Número de recomendações acertadas sobre número total de recomendações na página de detalhe do filme

Breaking Bad	**Interstellar**	**Se7en**	**The Office**
5/12	7/12	3/12	4/12

Após esta análise, fez-se uma avaliação de 20, 40 e por último 75 programas que o utilizador gostou. Para cada iteração de análises foi avaliada o número de recomendações "certas", seguindo a abordagem explicada neste documento. Foi utilizado sempre o mesmo número de total de recomendações (168).

Tabela 5 - Número de recomendações acertadas sobre número de recomendações dadas, tendo por base o número de análises positivas feitas pelo utilizador

Número de análises positivas	**20**	**40**	**75**
Recomendações corretas	45	35	24
Percentagem de recomendações "corretas"	≈ 27 %	≈ 21 %	≈ 14 %
Percentagem de recomendações "erradas"	0%	0%	0%

Tal como se pode visualizar na Tabela 5 acima, não existiu nenhuma recomendação "errada" em nenhuma das iterações. Um fator interessante é que a percentagem de programas que foram vistos e que foram do agrado do utilizador tende a decrescer, uma vez que aumenta o número de programas que o utilizador não viu. Essa situação é perfeitamente normal, uma vez que a oferta de itens é grande. Tendo em consideração estes resultados, e apenas dando como *feedback* análises de programas que o utilizador gostou, pode-se considerar que o sistema de recomendação implementado no IMDb é preciso, uma vez que não gerou uma recomendação "errada" para o utilizador.

Uma última análise foi realizada depois de se dar *feedback* sobre os filmes que o utilizador não gostou, e avaliados novamente os resultados. Foram inseridas 10 análises negativas, ou seja, programas que o utilizador não gostou.

Tabela 6 - Número de recomendações acertadas sobre número de recomendações dadas, tendo por base o número de análises negativas feitas pelo utilizador

Número de análises negativas	**10 (Total de Análises: 85)**
Recomendações certas	28
Percentagem de recomendações "corretas"	≈ 17 %
Percentagem de recomendações "erradas"	0%

Tal como se pode ver na Tabela 6, a análise negativa tem também um impacto no sistema de recomendações pessoais, onde apareceram mais quatro filmes que o utilizador viu e gostou do que anteriormente. Novamente não apareceram filmes que o utilizador viu e que não gostou, tendo novamente o sistema não gerado uma má recomendação.

Concluindo esta análise, o IMDb é neste momento uma grande referência no mundo do cinema e da televisão, com uma vasta equipa de profissionais e recursos que permitem ter acesso a conteúdo editorial exclusivo e rico em informação. No entanto, falha em informação de programas portugueses, onde estes não são tão detalhados como a base de dados da InfoPortugal. Possui também um sistema de recomendação bastante eficaz, que, no entanto, não é um foco do produto pela forma de como o sistema de recomendação personalizado é apresentado (está escondido dentro das opções do utilizador, tornando-se algo complexo para o utilizador vulgar encontrar essa funcionalidade, ao contrário de por exemplo o Netflix, que apresenta a lista de recomendações no seu ecrã principal).

2.1.4 Recomendações com EPG

Neste momento já existem algumas operadoras nacionais a fornecem recomendações aos utilizadores. No entanto, este tipo de recomendações são muito difícil de encontrar para o habitual telespectador, encontrando-se muitas vezes na descrição detalhada do programa.

Fornecer ao telespectador recomendações ligadas ao EPG facilita a utilização da televisão, uma vez que cada vez mais existe o problema de haver uma grande quantidade de programas a serem emitidos, em que grande parte do público visualiza apenas uma pequena parte dos canais. Na Ilustração 4 é possível ver a distribuição dos telespectadores nacionais para o dia 12 de junho de 2017, onde cerca de 57,7% da distribuição foi para quatro canais de televisão.

Ilustração 4 - Dados das audiências de televisão relativas ao dia 12 de junho de 2017 em Portugal

Este tipo de funcionalidade traz uma grande satisfação aos utilizadores. Num projeto semelhante realizado em 2000, foi desenvolvido uma aplicação distribuída por vários *websites*, onde era possível visualizar recomendações sobre o que ver na televisão. Os resultados finais mostraram uma satisfação de 95% nos utilizadores (Cotter & Smyth, 2000).

2.2 Enquadramento tecnológico

Nesta secção apresentam-se as principais tecnologias e ferramentas utilizadas durante o desenvolvimento da solução preconizada, divididas em dois grandes grupos: as utilizadas para a gestão do projeto e as usadas durante a construção.

2.2.1 Gestão do projeto

Para um projeto desta dimensão, é necessário garantir um histórico de todas as versões apresentadas ao cliente, assim como de todas as alterações feitas. Para isso, foi utilizado uma

ferramenta de controlo de versões (git), através da plataforma GitLab. Esta plataforma está hospedada na arquitetura da InfoPortugal, onde todos os seus repositórios estão privados, tendo apenas acesso pessoal autorizado. Na Ilustração 5 é possível ver o repositório do projeto do EPG na plataforma GitLab.

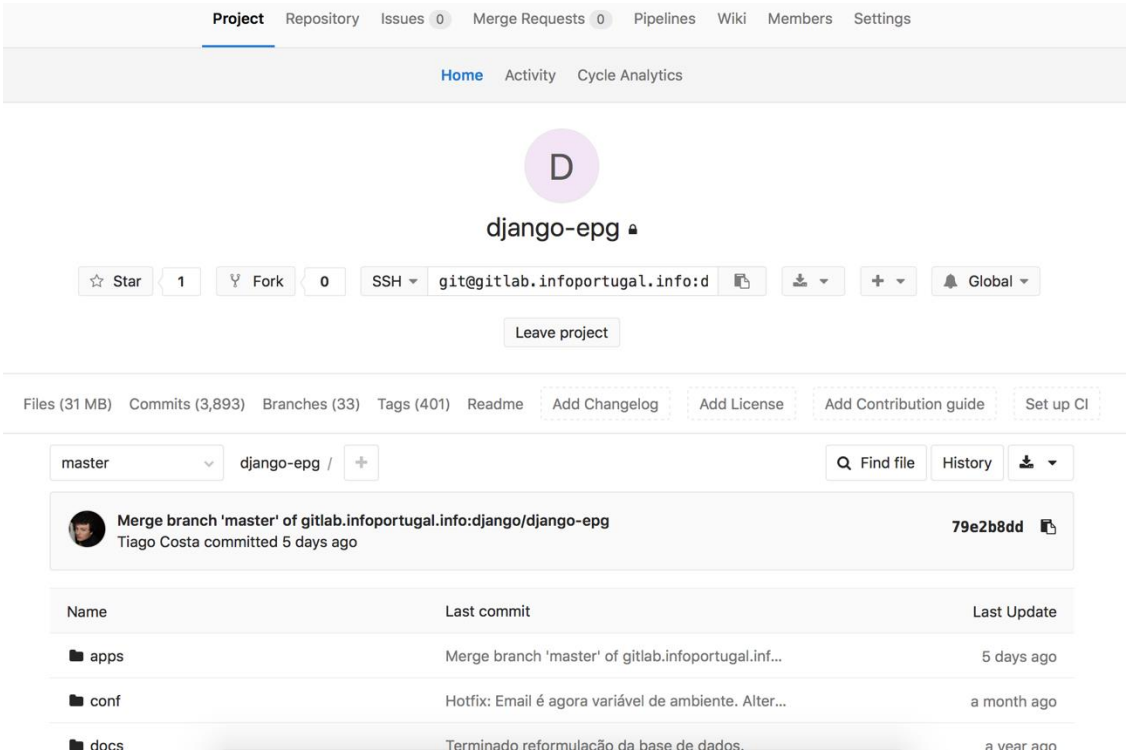

Ilustração 5 - Repositório no GitLab

Para a gestão do projeto em relação à divisão de tarefas, documentação do projeto, comunicação com restantes elementos da equipa e cliente, foi utilizada a ferramenta Redmine. O Redmine é uma plataforma *web* utilizada para a gestão flexível de projetos, que oferece várias funcionalidades como uma AgileBoard, diagramas de Gantt, criação e atribuição de tarefas, entre outras. Na Ilustração 6 é possível ver o diagrama de Gantt gerado automaticamente através da criação de tarefas para este projeto.

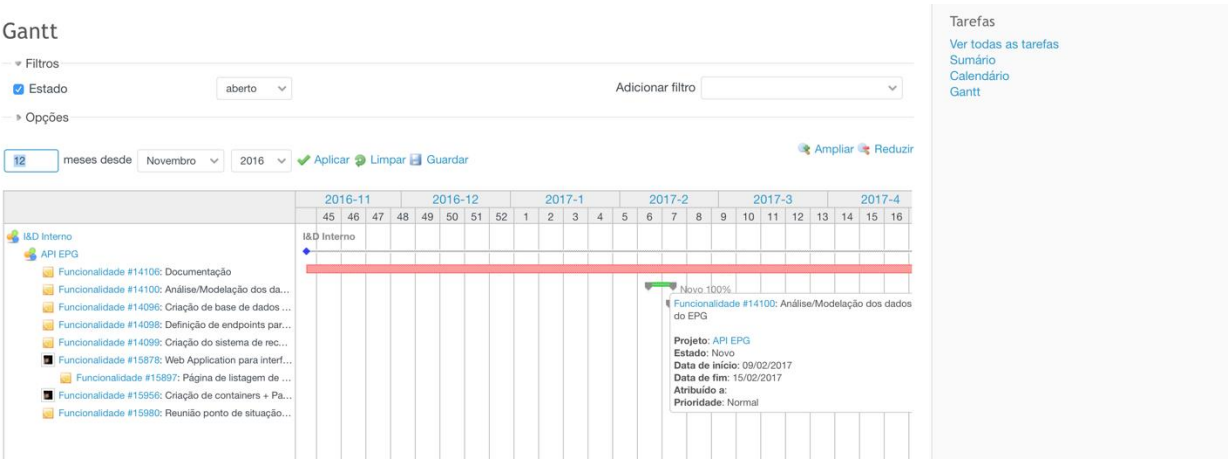

Ilustração 6 - Diagrama de Gantt na plataforma Redmine

2.2.2 Bibliotecas e ferramentas

Neste subcapítulo vão ser apresentadas várias bibliotecas e ferramentas pesquisadas que podem ser solução para o problema encontrado.

Na procura de ferramentas de recomendação foram tomadas algumas decisões prévias em relação à escolha da tecnologia: esta deve ser gratuita; e deve ser facilmente gerida por qualquer elemento da equipa de desenvolvimento da empresa, pelo que deve estar desenvolvida em Python.

2.2.2.1 LightFM

O LightFM é uma biblioteca desenvolvida em Python que possui um número de populares algoritmos de recomendação para *feedback* implícito e explícito (Kula, 2015). Através desta biblioteca torna-se simples a implementação de um sistema de recomendação numa aplicação, uma vez que esta oferece todas as operações necessárias para o cálculo de recomendações, tendo em consideração as interações de utilizadores em relação a um conjunto de itens.

Esta biblioteca oferece um algoritmo do tipo CF de base, onde utiliza os gostos de utilizadores semelhantes para recomendar novos itens ao utilizador atual. No entanto, oferece também uma vertente de CB, onde é possível definir uma matriz de atributos dos itens, que depois também é utilizada para o cálculo das recomendações. Existe ainda uma terceira matriz, onde é possível colocar os atributos do utilizador, como por exemplo o género, idade, local, entre outros. Esta biblioteca utiliza a licença Apache 2.0, que permite o uso do código fonte em projetos *open-source*, ou proprietários, e onde também é possível a edição do código fonte.

Por definição esta biblioteca oferece já alguns exemplos com alguns *datasets*, sendo um deles o *dataset* do Movielens, onde é possível gerar recomendações para 1000 utilizadores, tendo em consideração 100000 interações. Oferece ainda algumas funções para avaliar o algoritmo de recomendação, uma vez que é possível separar as interações em dois conjuntos de dados

(um de teste, e outro para treinar e ser utilizado para as recomendações) e calculando assim a precisão do sistema, entre outros métodos de avaliação de um sistema de recomendação.

Esta biblioteca utiliza técnicas de *machine learning*, uma vez que utiliza as classificações (positivas e negativas) dadas pelo utilizador para gerar novas recomendações. Ou seja, quanto maior for o número de classificações dadas por um utilizador, e quantos mais utilizadores tiver o sistema, o sistema fica mais preciso.

2.2.2.2 Crab

O Crab, também conhecido como "scikits.recommender", é uma *framework* Python para construir motores de recomendação integrando pacotes utilizados na área científica (numpy, scipy e matplotlib).

Este motor fornece um conjunto rico de componentes a partir do qual pode-se contruir um sistema de recomendações através do conjunto de algoritmos fornecidos e pode ser utilizado em vários contextos, como na ciência ou na engenharia (Muriçoca Labs, s.d.).

2.2.2.3 Surprise

Surprise é uma biblioteca "scikit" para construir e analisar sistemas de recomendação. Foi desenvolvida com o intuito de:

- Dar aos utilizadores controlo sobre as experiências a realizar, através de uma forte documentação;
- Facilitar o uso de *datasets*, oferecendo alguns já integrados (Movielens e.g.) ou podendo utilizar os seus próprios *datasets*;
- Oferece vários métodos para avaliar, analisar e comparar os resultados dos vários algoritmos de recomendação.
- Fornece vários algoritmos de previsão, assim como vários tipos de cálculos para semelhança.

A licença do projeto é BSD 3-Clause, que dita que a ferramenta pode ser utilizada para uso comercial (Hug, s.d.).

2.2.3 Motor de Pesquisa

Outra grande componente da aplicação é o motor de pesquisa, na qual seja possível procurar qualquer tipo de conteúdo da informação do EPG. Neste momento existem várias opções no mercado de motores de pesquisa já implementadas que oferecem funcionalidades interessantes que se enquadram com os requisitos desta aplicação. Mas para proceder à escolha, tem de se analisar várias componentes destes sistemas, como por exemplo a utilização de base de dados relacional e não-relacional, e quais as implicações da escolha do tipo da base de dados têm na pesquisa e na *performance* da mesma.

Uma das primeiras decisões a tomar quando se implementa pesquisa na aplicação é escolher como armazenar a informação que vai ser pesquisada. Sendo uma decisão tão importante para

a arquitetura do sistema, os requisitos têm de ser analisados para ponderar qual dos dois tipos de bases de dados devem ser escolhidos.

As bases de dados relacionais são as mais comuns em aplicações de informática. Distinguem-se pelo facto das várias tabelas usadas para identificar um tipo de objeto poderem possuir relações entre elas, de forma a criar a lógica da aplicação tendo em conta os requisitos pedidos.

Uma base de dados não relacional, também referida como NoSQL, como a MongoDB armazena a sua informação em ficheiros do tipo JSON (Homan, 2015). Este tipo de base de dados já não tem qualquer tipo de relação entre as tabelas, e essa é uma das principais razões pela qual estas bases de dados são mais rápidas a obter resultados, uma vez que existe uma "desnormalização" da estrutura de dados. Algumas das principais razões para os programadores optarem por bases de dados não relacionais é quando o modelo de domínio começa a ficar demasiado complexo ou começa-se a investir tempo a desnormalizar o esquema da base de dados (Homan, 2015). Uma grande vantagem das bases de dados não relacionais é o facto de não utilizarem SQL e de não possuírem um *schema* faz com que não seja possível um ataque do tipo "SQL Injection", dos ataques mais comuns a aplicações (Homan, 2015). No entanto, qualquer outro tipo de vulnerabilidade na aplicação pode fazer com que o atacante tenha acesso aos dados de outra forma.

A grande desvantagem de bases de dados não relacional é o facto de estas não possuírem relações. Ou seja, imaginando que para um certo requisito é necessário captar todos os episódios pertencentes a uma série, essa chamada à base de dados é mais complexa que a chamada de uma base de dados relacional. Outra desvantagem está associada ao facto de as bases de dados não relacionais não utilizarem SQL, que traz alguns benefícios como uma sintaxe declarativa e uma base de matemática forte.

2.2.3.1 ElasticSearch

O ElasticSearch (ES) é um motor em tempo-real de pesquisa distribuída. Permite ao utilizador explorar informação a uma velocidade e escala nunca antes possível. Pode ser utilizado com pesquisa do tipo *full-text*, pesquisa estruturada, analisar informação ou os três anteriores em conjunto (Elastic, s.d.). Nenhuma das componentes anteriores é revolucionária, uma vez que já foram desenvolvidas antes do ElasticSearch ser divulgado. No entanto, nunca foram combinadas numa única aplicação de uma forma tão coesa e em tempo-real.

Devido ao potente motor de pesquisa do ElasticSearch, é possível pesquisar com texto inteiro, fazer um tratamento de sinónimos, ordenar os resultados por ordem de relevância, gerar uma pequena análise da pesquisa e agregar informação do mesmo tipo, em tempo-real e sem grande carga para a máquina (Elastic, s.d.).

O ElasticSearch foi desenvolvido sobre o Apache Lucene, motor de pesquisa *full-text*. O Lucene é o mais avançado, com melhor performance e com o melhor motor de pesquisa disponível atualmente. Todavia, a sua integração em projetos é bastante complexa, ao ponto que pode ser necessário um curso para entender como a informação deve ser processada. O ElasticSearch

esconde a complexidade do Lucene atrás de uma simples e coerente API. Destaca-se ainda que o ElasticSearch também oferece outras funcionalidades para além do motor de pesquisa, ao contrario do Lucene (Elastic, s.d.).

O ES é orientado a documentos, ou seja, é armazenado tudo o que seja objeto em documento. Este não só armazena toda a informação, como também indexa o objeto para que este seja pesquisável. Para a serialização de documentos, o serviço utiliza o JSON (JavaScript Object Notation). Os ficheiros do tipo JSON são suportados pela grande maioria das linguagens de programação, porque são simples, coesos e de fácil leitura.

O ES armazena a informação numa base de dados não relacional. De uma forma simples, e em comparação a uma base de dados relacional: o ES suporta a criação de vários índices, equivalentes a base de dados na nomenclatura relacional. Cada índice pode possuir vários tipos (tabelas). Cada tipo possui um número indeterminado de documentos (*rows*), em que cada documento possui os seus atributos (Elastic, s.d.).

3 Análise de Valor

Neste capítulo é apresentada uma análise de valor à plataforma desenvolvida, com o intuito de provar o seu valor no mercado nacional atualmente.

A InfoPortugal faz a distribuição de programação de televisão nacional atualmente para três das quatro principais operadoras, em que estas detêm 60,4% do mercado. Algumas destas operadoras possuem uma relação duradoura, que abrange também outros projetos. Atualmente não existe nenhum sistema que interage com os espectadores portugueses que dê recomendações personalizadas, tendo em conta os seus gostos, tornando-se numa aplicação única.

3.1 Identificação da oportunidade

Neste sentido, surgiu a oportunidade de expandir o serviço de programação de televisão com as operadoras, para não só enviar essa programação, mas também para abranger a recomendações personalizadas e um motor de pesquisa rápido. A identificação desta oportunidade surgiu numa reunião com um dos clientes, onde foi discutida a necessidade de um serviço assim, para aumentar a satisfação dos seus clientes.

3.2 Análise da oportunidade

Tendo em conta outras plataformas semelhantes com a televisão, como por exemplo o serviço Netflix, onde cerca de 75% dos utilizadores vê conteúdo através de algum tipo de recomendação (Amatriain, 2012), isso traria uma grande valia às operadoras, uma vez que iria prender os utilizadores ao seu serviço. Desta forma, é de grande interesse das operadoras fornecem um serviço semelhante aos seus clientes, e uma vez que já existe um negócio com a

InfoPortugal com estes dados, facilita a comunicação e existe já confiança para a contratação destas funcionalidades.

3.3 Geração e enriquecimento de Ideias

A ideia principal seria qualquer aplicação ser capaz de comunicar com este sistema, para obter estes dados de programação, assim como as funcionalidades referidas. Poderiam ser aplicados vários tipos de algoritmos de recomendação, assim como vários tipos de pesquisa sobre estes conteúdos, de diversas formas. De acordo com o cliente, poderia existir mais ou menos conteúdo, assim como o número de funcionalidades mais reduzido ou exclusividade no tipo de conteúdo apresentado.

3.4 Seleção de Ideias

Foi selecionada a ideia de criar uma plataforma que utilize as classificações dos utilizadores, assim como os atributos dos programas para gerar recomendações para os utilizadores. Esta aplicação deve também conter uma componente de pesquisa sobre todo este conteúdo, e em que esta possa ser facilmente filtrada, de acordo com os requisitos do cliente. Esta plataforma deve estar disponível e ser abrangente o suficiente para qualquer tipo de sistema e não se focar apenas num.

3.5 Definição de conceito

Concluindo, irá ser desenvolvida uma plataforma, em que será utilizado a opinião dada pelos utilizadores e atributos de programas para gerar recomendações, e que permita a pesquisa sobre conteúdo que está neste momento a ser emitido na televisão. Esta plataforma responde a pedidos do tipo HTTP, podendo assim ser abrangente o suficiente para qualquer tipo de dispositivo que consiga comunicar através de internet. Para a funcionalidade da pesquisa foi selecionado o ElasticSearch que fornece todos os requisitos necessários para a funcionalidade. Para as recomendações foi escolhida a biblioteca LightFM, uma vez que oferece um algoritmo híbrido de recomendações e cumpre todos os requisitos pedidos.

4 Valor, ameaças e oportunidades

Neste capítulo é documentado todo o processo de análise do ponto de vista de negócio deste projeto, apresentando uma análise das forças, fraquezas, oportunidades e ameaças do projeto, o que o distingue da concorrência e um modelo canvas para explicar o modelo de negócio do mesmo. Esta análise de valor é feita na perspetiva da InfoPortugal, e não pelo ponto de vista de ser a empresa a disponibilizar estas funcionalidades ao utilizador final (telespectador).

4.1 Parceiros e stakeholders

Os principais *Stakeholders* e parceiros associados a estes projetos são as operadoras nacionais com quem a InfoPortugal já possui projetos em comum:

- Vodafone, cliente que a InfoPortugal fornece o EPG, mas também outro tipo de projetos ligados ao EPG, contactos úteis, entre outros. No caso do projeto do EPG, este serviço já é feito há nove anos.

- Cabovisão, cliente que a InfoPortugal fornece o EPG.

- NOS, cliente que a InfoPortugal fornece o EPG.

A Vodafone foi contactada sobre o desenvolvimento deste projeto, de forma a fornecer os dados de telespectadores para estes serem utilizados na geração das recomendações. Não foi possível obter esses dados, mas a Vodafone ficou interessada no projeto, e pretende utilizá-lo nas suas plataformas, estando neste momento em fase de testes para avaliar as funcionalidades desenvolvidas.

4.2 Valor e impacto

Tendo em conta o estado atual de Portugal em relação às operadoras de televisão, onde existem apenas quatro operadoras que abrangem quase a totalidade do mercado (cerca de 99,1%)[4] e que a InfoPortugal fornece a informação a três dessas, o impacto deste projeto iria ser enorme, caso todas as operadoras com quem a InfoPortugal já possui uma ligação quisessem este projeto (aproximadamente 3,6 milhões de lares).

Por várias vezes já foi anunciada a morte da televisão (Mendes, 2017), e uma das formas para as operadoras conseguirem prender os seus espectadores à televisão é a utilização de um sistema de recomendação credível, uma vez que neste momento existe uma enorme oferta de canais, em que grande parte desta oferta não é explorada, como se pode ver pela Ilustração 4.

Neste momento não existe nenhum produto no mercado que ofereça as mesmas funcionalidades que a aplicação, ou seja, um sistema de recomendação baseado em conteúdo nacional e internacional e que possa ser utilizado por qualquer tipo de sistema. É de grande interesse das operadoras terem acesso a um projeto destes, com o intuito de cativar os seus utilizadores.

Este tipo de algoritmos de recomendação são uma tendência nestes últimos anos, e são muito bem avaliados em termos monetários. Um exemplo similar ao desenvolvido neste projeto é o algoritmo de recomendação do Netflix, que é avaliado em mil milhões de dólares por ano (McAlone, 2016). Para além disso, trazem valor para o mercado, uma vez que neste momento é dito que 75% dos conteúdos visualizados no Netflix são a partir de recomendações (Amatriain, 2012).

O desenvolvimento do sistema do EPG faz com que a equipa de programadores tenha conhecimento de todo o modelo de domínio e também dos problemas mais comuns associados a este tipo de informação, por isso existe neste momento uma equipa especializada no desenvolvimento com este tipo de dados.

4.3 Ameaças

A informação da programação de televisão é facilmente encontrada em vários meios: através do *website* do canal, onde é disponibilizada uma grelha com a programação; através de revistas e jornais, que apresentam a programação dos canais com maior audiência e descrições simples; e mesmo pelas próprias operadoras, que apresentam as grelhas nas *boxes* de televisão e no próprio *website*. Tendo isso em consideração, a curva de entrada do mercado é baixa, uma vez que é fácil de obter estes dados e desenvolver um projeto que recomende conteúdo televisão, embora a informação não seja tão detalhada e tratada.

[4] Dados retirados da ANACOM no dia 17 de fev. de 2017 (https://www.anacom.pt/render.jsp?contentId=1400523)

As próprias operadoras podem optar por desenvolver este projeto internamente e tornarem-se concorrentes diretas, uma vez que têm a mesma informação que é disponibilizada pela InfoPortugal e acesso direto ao *feedback* dado pelos utilizadores, que facilita o processo de maturação para as recomendações. Esse exemplo pode ser visto pela Meo, onde é a própria operadora que trata de obter a informação de programação para disponibilizar posteriormente aos seus clientes.

As interações dadas pelo utilizador são também uma ameaça para o nosso projeto, uma vez que o sistema de recomendações vive do *feedback* dado pelo utilizador para aprender os gostos dele, e sem isso as recomendações não evoluem. As operadoras têm acesso ao que o utilizador visualiza, e tem em seu poder essa informação. No entanto, na eventualidade de não haver partilha desses dados para a criação deste projeto, perde-se uma grande quantidade de dados que ajudariam no desenvolvimento e que facilitavam a aprendizagem do algoritmo de recomendação. Por sua vez, e frisando novamente o ponto anterior, é uma vantagem que as operadoras têm em relação à InfoPortugal e que caso esta desenvolve-se um projeto semelhante internamente, teriam facilmente recomendações mais úteis do que com pouca informação acerca do espectador.

Serviços como a Netflix, Hulu e Amazon Video são concorrentes indiretos a este projeto. Estes serviços têm crescido exponencialmente nos últimos tempos. O Netflix, serviço mais popular deste género, possui cerca de 94 milhões de subscritores em todo o mundo no último quadrante do ano de 2016, aproximadamente mais 20 milhões que no ano anterior (Statista, s.d.). O Netflix abrange cada vez mais países e pessoas, e tendo um conteúdo personalizado para os utilizadores e com grande quantidade e disponível numa grande gama de dispositivos faz com que os espectadores não utilizem a televisão para ver filmes e séries.

4.4 Oportunidades

Tal como já foi referido várias vezes neste documento, o produto a ser desenvolvido é o único no mercado português. Possuindo exclusividade neste tipo de produto e conseguindo o monopólio deste nicho de mercado vai ajudar a InfoPortugal a destacar-se ainda mais na programação de televisão e como empresa tecnológica.

A forma de como este projeto é desenvolvido para abranger qualquer tipo de sistema também é uma oportunidade, uma vez que não se prende diretamente às operadoras. Através deste projeto podem ser criados *websites* ou aplicações móveis com este tipo de conteúdo, onde o utilizador é recomendado sobre o que pode ser um programa do seu agrado, ou então encontrar facilmente um programa que queira pesquisar.

A partilha de informação entre a aplicação a desenvolver e o produto final disponibilizado aos utilizadores é uma situação de *win-win* para a InfoPortugal e para a empresa que disponibilizar o conteúdo final:

- A InfoPortugal recebe o *feedback* dado pelo utilizador, sabendo quais os programas que gosta e continua a ensinar o sistema de recomendação, que cada vez fica mais preciso;

- O produto final vai aumentando a precisão das recomendações a fornecer ao utilizador final, uma vez que o sistema de recomendações vai ficando mais maturo, aumentando assim também a satisfação do utilizador final.

4.5 Modelo de Negócio

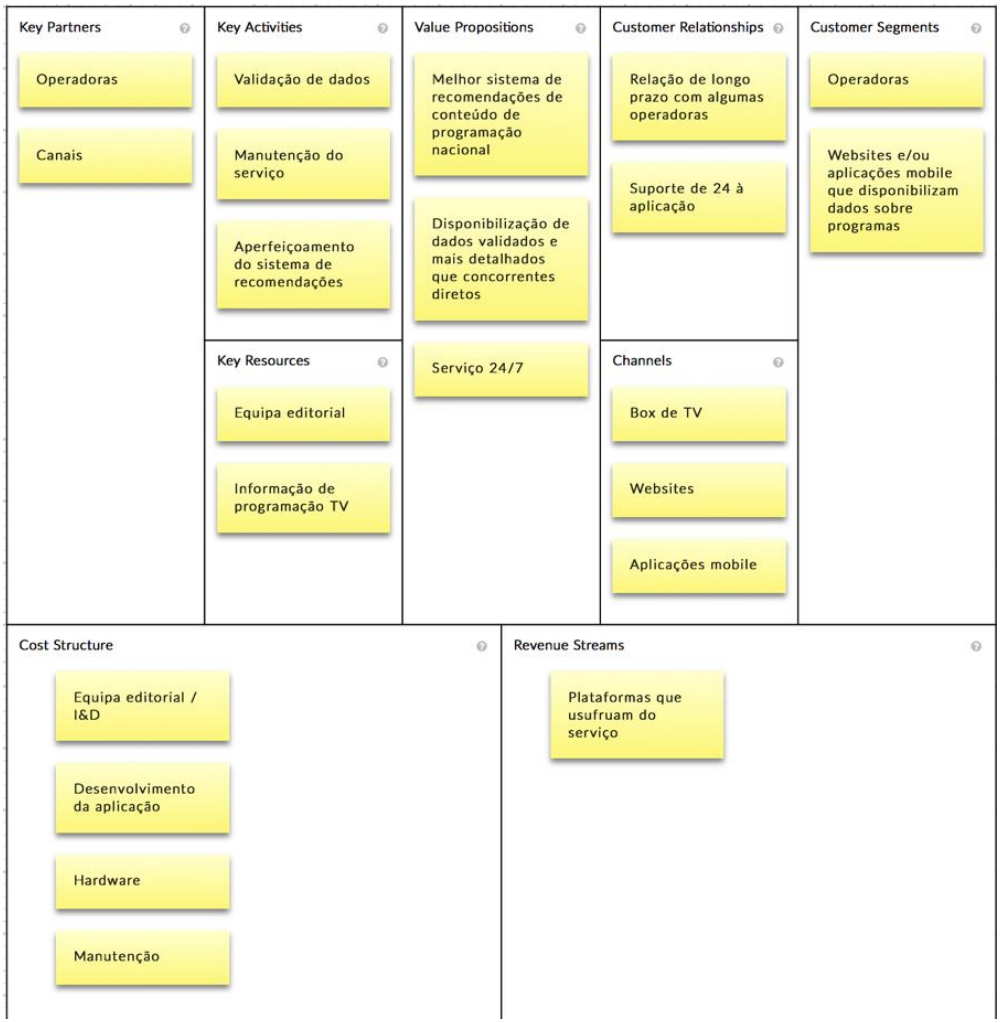

Ilustração 7 - Modelo de Canvas

O modelo de Canvas, proposto por Alexander Osterwalder, tem como objetivo descrever, modelar, e gerir um modelo de negócio (Strategyzer, s.d.). Através deste diagrama é possível numa página descrever de uma forma simples o nosso modelo de negócio para uma empresa/projeto. A Ilustração 7 apresentada ilustra este modelo aplicado neste projeto.

Os principais parceiros no desenvolvimento deste produto são as operadoras e os canais de televisão. As operadoras vão ajudar na aprendizagem do algoritmo de recomendação, ao disponibilizar a informação do que o utilizador visualiza e gosta. Em relação aos canais de televisão, estes enviam a programação de televisão com informação detalhada, em grande parte dos casos, dos programas a serem transmitidos. Uma vez que os canais fornecem essa informação exclusivamente aos EPG *providers*, a informação deste produto torna-se exclusiva (em conjunto com a informação enviada para as operadoras).

Para o desenvolvimento deste produto ser suave e atingir os objetivos é necessário cumprir certas atividades: devem ser validados e tratados todos os dados dos programas que vão ser emitidos e disponibilizados no sistema (esse tratamento já é efetuado atualmente para as operadoras); deve haver uma constante manutenção do serviço, de forma a garantir a disponibilidade do mesmo para os clientes; o algoritmo de recomendação deve ser cada vez mais aperfeiçoado tendo em conta o *feedback* dos clientes e dos utilizadores finais.

Como recursos é fundamental a existência de uma equipa de desenvolvimento e investigação para criar a solução e desenvolvê-la, assim como ter o conhecimento para gerir a manutenção da mesma. A equipa editorial é crucial para a edição e validação dos dados enviados, que faz com que a InfoPortugal tenha acesso à melhor informação da programação televisiva nacional. De forma a garantir que o serviço está operacional é necessário ter *hardware* que resista aos pedidos feitos à aplicação, portanto é necessário um investimento em máquinas físicas (servidores).

Esta aplicação distingue-se das restantes soluções do mercado uma vez que oferece um sistema de recomendação do conteúdo de programação nacional, sendo o único na atualidade que oferece essa funcionalidade. É ainda a forma mais simples para um sistema obter a informação de programação nacional validada por uma equipa para utilizar na sua aplicação. A InfoPortugal garante também que o serviço tem uma disponibilidade alta.

A InfoPortugal já tem estabelecidas algumas relações com alguns dos possíveis clientes, alguns em mais do que um projeto, pelo que a relação com estes clientes iria ser ainda mais fortalecida. De forma a garantir que todos os clientes vão estar sempre satisfeitos com o serviço proposto, existe uma linha telefónica de suporte para onde os clientes podem informar na eventualidade de algum serviço estar indisponível ou esclarecer qualquer dúvida da componente tecnológica.

Os principais canais de acesso ao projeto são as *boxes* de televisão, que terão acesso à pesquisa e ao sistema de recomendações, e *websites* e/ou aplicações móveis com informação da programação televisiva ou sobre programas de televisão. Estes são também os principais clientes que podem utilizar este serviço.

Concluída a análise, chega-se à conclusão que os custos para a produção deste projeto são para as equipas de editorial e desenvolvimento, na compra do *hardware* necessário para disponibilizar a aplicação, e na manutenção da mesma. Como retorno, as operadoras e as empresas que usufruam deste serviço são cobradas pelo uso do mesmo.

5 Análise de negócio e requisitos

Neste capítulo são introduzidos todos os requisitos funcionais e não funcionais do sistema, assim como quais os principais utilizadores do sistema e o que se pretende de uma forma geral com a plataforma desenvolvida.

5.1 Análise de negócio

O diagrama do modelo de domínio do projeto está representado na Ilustração 8.

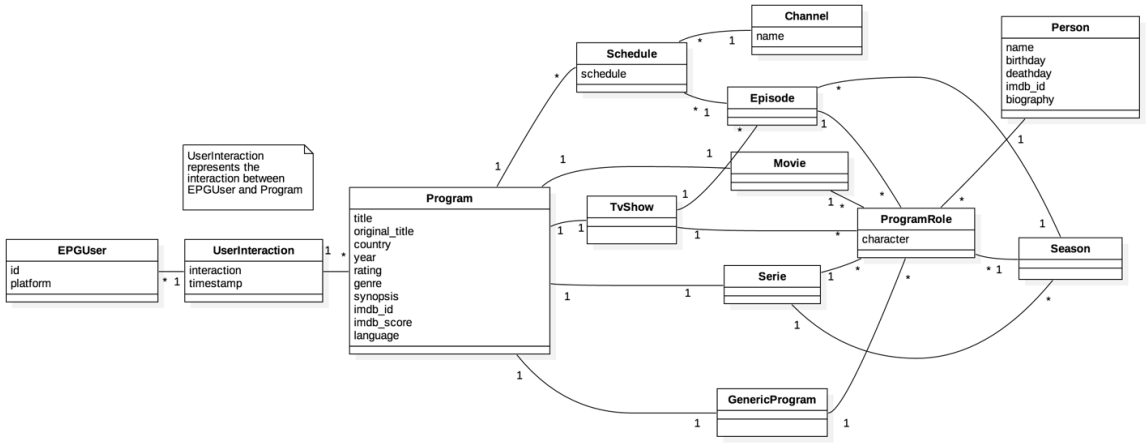

Ilustração 8 - Modelo de domínio

Do lado direito do artefacto é possível verificar os componentes necessários que armazenam a informação da programação de televisão. A tabela “Program” contém toda a informação de um certo programa, e tem a si associado um número indefinido de canais e de papéis associados, uma vez que não existe limite máximo, e pode existir zero em ambos os casos. Cada

"ProgramRole" está associado a uma pessoa, que possui toda a sua informação. A relação entre programa e canal possui uma tabela intermédia de "Schedule", que indica todos os horários de um programa, assim como todos os horários de um certo canal. O episódio pode estar associado a uma temporada ("Season") ou a um TvShow. Uma série possui várias temporadas. Do lado esquerdo é apresentado o utilizador que vai receber recomendações. Este utilizador possui um identificador único que é enviado pela plataforma que esse está a utilizar (tal como se pode conferir através do atributo "platform"). Este utilizador vai interagir com um *n* número de programas, onde essa interação vai ter um valor a si associado, a ser considerado para o sistema de recomendação.

5.2 Análise de requisitos

O objetivo principal deste projeto é desenvolver uma aplicação onde qualquer outro sistema consiga pesquisar sobre conteúdos de programação de televisão portuguesa e que quando um utilizador esteja autenticado nessa plataforma, recomendações desse mesmo conteúdo sejam dadas ao utilizador, tendo em consideração os gostos do utilizador. Nesta secção vão ser especificados os requisitos funcionais e não funcionais do projeto.

5.2.1 Atores do sistema

Os verdadeiros clientes da solução desenvolvida são empresas que tenham interesse em disponibilizar conteúdo da programação e recomendar esse conteúdo numa certa plataforma. Ao longo do documento foi referido que as operadoras poderiam ser um desses clientes, em que as *boxes* de televisão seriam uma boa plataforma para ter esta solução implementada. Outros possíveis utilizadores do sistema são aplicações *Web*/*Mobile* que queiram apresentar estes dados a utilizadores, como por exemplo as grelhas de programação que se encontram *online*. Por último, esta solução pode ser utilizada também para desenvolver alguma solução interna de disponibilização do conteúdo de programação televisiva, na forma de uma aplicação móvel ou de um *website*.

Resumidamente, em todos os casos existe apenas um ator que interage com o sistema que é uma plataforma externa que comunica com a aplicação desenvolvida para obter conteúdo.

5.2.2 Requisitos funcionais

Tendo então em consideração o único ator presente neste sistema, foi modelado o seguinte diagrama de casos de uso:

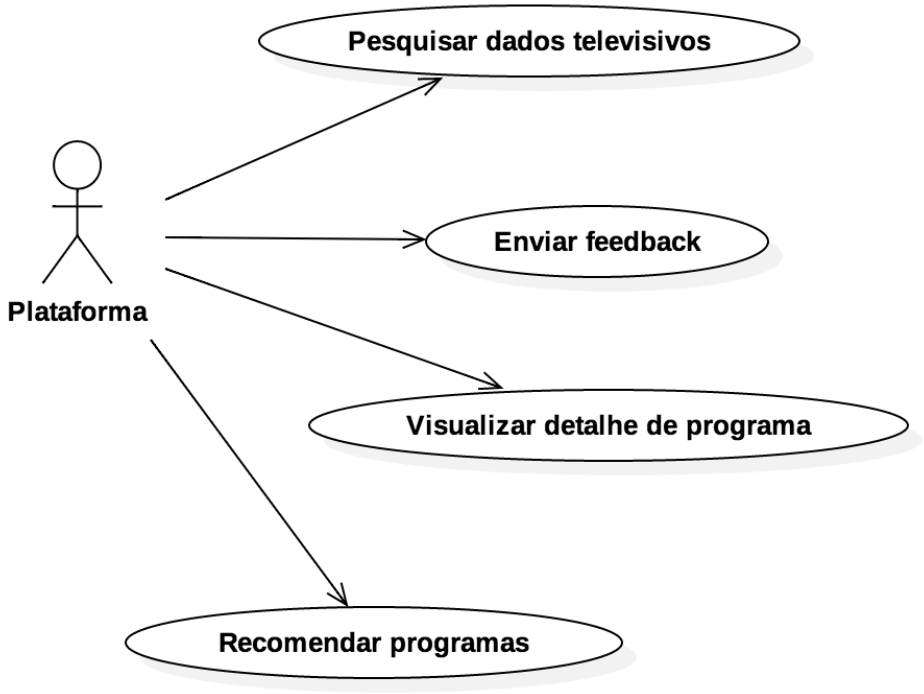

Ilustração 9 - Diagrama de casos de uso

Tal como é possível ver na Ilustração 9 existem no total quatro casos de uso. Estes casos de uso representam todas as funcionalidades que devem estar presentes a qualquer plataforma. Na Tabela 7 são sintetizados todos estes casos de uso, assim como uma breve descrição de todas as funcionalidades.

Tabela 7 - Tabela de casos de uso

UC	**Ator**	**Cenário de Sucesso**	**Cenários Alternativos**
1 – Pesquisar dados televisivos	Plataforma	A plataforma envia o texto, que é processado pelo sistema e devolve os resultados da pesquisa sobre dados televisivos (programas e elencos).	Caso não exista a variável de texto, deve ser retornado um erro de pedido à plataforma. Caso seja enviado um filtro de data, os resultados devem ser filtrados para conter os programas que são transmitidos nesse intervalo.
2 – Enviar *feedback*		A plataforma envia o *feedback* dado por um certo utilizador, que é depois processado pelo sistema e carregado no sistema de recomendação.	Caso o utilizador ainda não exista no sistema, este deve ser criado. Caso os parâmetros enviados sejam incorretos, deve ser indicado à plataforma o erro.

3 – Visualizar detalhe de programa		Apresenta toda a informação associada ao programa.	Caso a plataforma não especifique o identificador do programa, devem ser listados todos os programas. Caso a plataforma especifique que quer os programas em grelha, devem ser apenas apresentados os programas em grelha.
4 – Recomendar programas		Envia *n* recomendações para o utilizador enviado no pedido.	Caso o utilizador ainda não exista no sistema, este deve ser criado. Caso o utilizador não possuir 10 interações com o sistema, são captados 10 programas em cartaz com uma boa classificação no IMDb.

5.2.3 Outros requisitos

É fundamental no desenvolvimento de uma solução deste género seguir certos requisitos em termos de segurança e *performance*, para o produto ter valor e mostrar interesse a possíveis clientes.

Em termos de segurança é importante a atribuição de chaves únicas para cada um dos clientes. Desta forma não é permitido a plataformas anónimas fazerem pedidos à aplicação e é feito um controlo de quem faz os pedidos. Um fator importante a ter em conta é que a aplicação tem uma ligação direta com a atual base de dados com a informação da programação. Deve-se garantir que são atribuídas as permissões corretas, tendo acesso apenas à função de *read-only* dessa base de dados, mantendo a integridade dos dados.

Uma vez que esta aplicação tem como destino plataformas que estão sempre disponíveis a possíveis utilizadores, é necessário garantir uma disponibilidade de 24/7, assim como a elasticidade do sistema deve permitir uma grande carga de pedidos e adaptar-se a essa quantidade de pedidos, sem haver quebras de *performance*. Considerando, por exemplo, a quantidade de pessoas que as operadoras com quem a empresa já tem um acordo, é necessário que o sistema seja facilmente escalável, adaptando-se facilmente a mudanças no número de pedidos.

Um outro requisito bastante importante neste tipo de aplicações é a *performance*. Se um sistema demora a gerar as recomendações para um utilizador, por muito precisas que essas recomendações sejam, o utilizador começa a perder interesse. Um utilizador normal não nota diferenças se o sistema responder em 0.1 segundos, segue a mesma linha de raciocínio se o

sistema demorar 1 segundo, e 10 segundos é o tempo máximo para perder a atenção do utilizador (Nielsen, 1993). É necessário no mínimo garantir que o utilizador não perde o foco da aplicação, e no futuro melhorar o tempo de resposta para ser cada vez mais fluido.

6 Arquitetura da Solução

Neste capítulo é abordada a arquitetura proposta para o sistema, explicando alguns conceitos básicos, assim como os pressupostos e opções tomadas na elaboração desta arquitetura.

6.1 Alternativas

Neste capítulo são apresentadas algumas alternativas para a arquitetura do projeto, assim com a explicação de alguns conceitos e tecnologias utilizadas na arquitetura do sistema para uma mais fácil compreensão do documento e do projeto.

Uma das tecnologias mais discutidas ao longo do documento e do projeto é Representational State Transfer, também conhecido como REST. O REST é um protocolo de comunicação *stateless* e *cachable* entre cliente-servidor, utilizando o protocolo Hypertext Transfer Protocol (HTTP) (Elkstein, 2008).

O REST é um estilo de arquitetura utilizada para a construção de aplicações *Web*. A ideia é utilizar o HTTP para estabelecer a comunicação entre as diferentes máquinas em vez de tecnologias mais complexas como é o caso do Simple Object Access Protocol (SOAP) (Elkstein, 2008). Através do REST é possível a utilização de vários métodos HTTP:

- POST, pedido geralmente utilizado para a criação de informação;
- PUT, que atualiza o conteúdo de certo objeto;
- GET, utilizado para obter informação;
- DELETE, para apagar conteúdo.

Através destes pedidos o REST consegue cobrir todas as operações de CRUD (Create/Read/Update/Delete).

Uma das grandes vantagens que o REST possui é o facto de poder ser utilizado em qualquer tipo de linguagem e plataforma, tornando-se cada vez mais um *standard* na comunicação entre aplicações *Web*.

Considerados os requisitos funcionais e não funcionais do projeto, duas alternativas foram ponderadas:

- Uma arquitetura de microserviços, utilizando uma *application programming interface* (API) REST para comunicação entre as aplicações e a camada de serviços.
- Uma arquitetura "n-tier architeture", também conhecida como arquitetura por camadas. Com a aplicação deste padrão os serviços estão separados entre si, fazendo com que a aplicação seja mais robusta, mais escalável, e suportar *continuous-delivery* mais facilmente (Richards, 2015).

Em comparação com a arquitetura por camadas, a agilidade de uma arquitetura de microserviços é maior, uma vez que as suas componentes (serviços) estão separadas entre elas e o acoplamento é baixo. Na arquitetura por camadas, embora haja a separação das camadas, estas encontram-se muitas vezes acopladas, perdendo-se muitas vezes tempo a fazer uma alteração simples numa das camadas (Richards, 2015).

Devido ao grande acoplamento que existe entre as camadas, o próprio *deployment* de aplicações pode tornar-se complicado, dado que uma pequena alteração pode gerar erros sequenciais entre as várias camadas, o que faz com que as passagens a produção sejam planeadas com muita antecedência e muitas vezes fora de horas (Richards, 2015). Na arquitetura de microserviços na falha da passagem de um dos serviços, apenas esse serviço fica indisponível. Também se torna possível fazer alterações de apenas uma das componentes do serviço, ao invés de toda a aplicação.

Ambos os padrões são eficazes na componente de testes, não sendo este um ponto diferenciador na escolha:

- Os microserviços estão separados, tornando-se fácil focar nos testes de uma certa componente, assim como é muito mais difícil um serviço quebrar os testes de outro serviço;

- Na arquitetura de camadas, tendo em consideração que cada componente pertence à sua camada, facilmente consegue-se desenvolver testes para uma camada, ou simular outra camada. Ou seja, facilmente consegue-se simular uma interface na camada de negócio, e simular a camada de negócio na camada de interface (Richards, 2015).

Outro ponto em comum entre estes dois padrões é a *performance*. Enquanto que no padrão de arquitetura de camadas a *performance* é afetada porque certas funcionalidades têm de

percorrer todas as camadas, nos microserviços o facto de estes estarem separados faz com que a *performance* não seja a melhor, caso estes estejam ligados entre eles (Richards, 2015).

Em termos de escalabilidade e facilidade de desenvolvimento, a separação dos vários serviços permite escalar cada um deles individualmente, tornando este padrão altamente escalável e o desenvolvimento separado de cada serviço faz com que a gestão de tarefas seja mais simples (Richards, 2015). Na arquitetura de camadas como existe acoplamento torna-se difícil de escalar a aplicação (Richards, 2015). No entanto, em termos de desenvolvimento é o padrão mais conhecido e o primeiro a ser ensinado a muitos programadores, pelo que a sua implementação é fácil e simples.

6.2 Arquitetura Proposta

Ponderadas todas as alternativas apresentadas no capítulo anterior, foi selecionada uma arquitetura de microsserviços, utilizando o REST para a comunicação entre os sistemas.

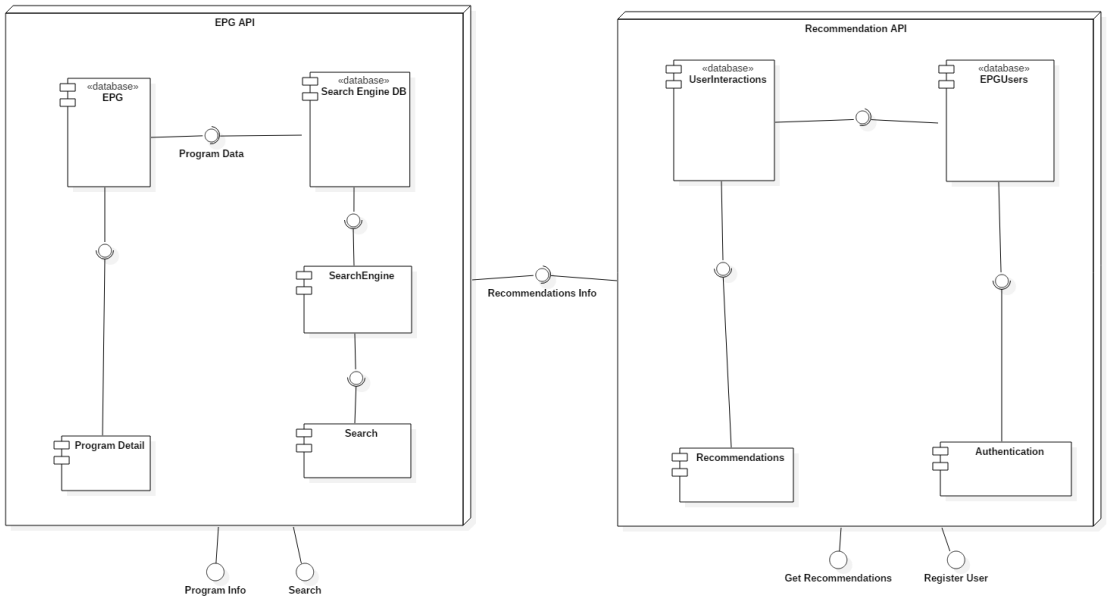

Ilustração 10 - Diagrama de Arquitetura

Tal como é possível ver na Ilustração 10, existem dois módulos principais na aplicação:

- "EPG API", que representa todos os componentes que necessitam da base de dados do EPG, ou estão diretamente ligados à base de dados do EPG. Este subsistema possui os microserviços de pesquisa e de obtenção de informação de programas. A comunicação para o exterior é feita através de REST;

- "Recommendation API", que possui as bases de dados de interações e utilizadores do projeto, assim como os serviços de recomendações e registo de utilizadores.

A plataforma é construída em Django, *framework* desenvolvida em Python. Esta *framework* tem o *motto* de desenvolvimento rápido e limpo com um design pragmático (Django, s.d.). A escolha desta ferramenta para o desenvolvimento reflete exatamente o lema do Django, que é desenvolver algo rapidamente de uma forma muito simples. O facto da empresa também desenvolver outros produtos utilizando esta tecnologia pesa também na decisão, uma vez que o desenvolvimento é ainda mais rápido e o contacto diário com a tecnologia permite ultrapassar eventuais problemas mais rapidamente.

Os vários serviços existentes neste projeto são desenvolvidos em Python, utilizando a ferramenta Django REST Framework. Esta biblioteca possui já algumas funcionalidades intrínsecas e facilita o processo de serialização dos dados em formatos utilizados em APIs, como por exemplo JSON e XML. Novamente esta ferramenta é utilizada na InfoPortugal, não havendo necessidade de um processo de aprendizagem e criação de "falsas expectativas", tendo de se optar por outras opções durante o desenvolvimento. Cada uma das componentes mencionadas possui microsserviços, cada um com o seu *endpoint*, para o qual a interface faz um pedido para aceder aos dados.

O subsistema "EPG API" foi desenvolvido sobre o projeto atual do EPG, onde existe uma expansão deste projeto para conter os *endpoints* de pesquisa e obtenção de dados dos programas. Esta expansão deve ser fácil de desacoplar do EPG, com que uma simples mudança no *pipeline* faça com que todos os componentes sejam desligados e completamente desacoplados do EPG. Esta expansão em nada interfere com a base de dados atual do EPG, assim como na sua estrutura de dados.

No módulo "Recommendations API" existe as bases de dados de interações feitas pelos utilizadores e os utilizadores registados na plataforma, assim como os serviços que vão disponibilizar as recomendações para esses mesmos utilizadores e que permitem o seu registo. Possui interfaces exteriores para que outras aplicações consigam aceder a esta informação, obtendo as recomendações personalizadas. Existe uma ligação com o subsistema "EPG Info" para captar toda a informação de programas, presente na base de dados do EPG.

Tal como já foi referido neste documento e como é possível visualizar no diagrama de arquitetura (Ilustração 10), existem quatro bases de dados neste sistema. Tendo em consideração que esta aplicação não deve estar dependente de uma aplicação em específico, existe uma separação lógica das bases de dados: existe uma base de dados que guarda todos os tipos de utilizadores, independentemente da aplicação pela qual se registaram; uma base de dados de *feedback* dado pelos utilizadores em programas. Cada *feedback* tem a si associado o programa e o utilizador em questão. Existe ainda uma base de dados não relacional utilizada pela ferramenta ElasticSearch, representada pelo componente "Search Engine", que é posteriormente utilizada para pesquisa. O que potencializa este projeto é a informação sobre os programas que passam na televisão portuguesa, logo é importante garantir que essa informação esteja sempre atualizada e validada. Tendo isso em consideração, a base de dados mais importante torna-se o "EPG". Esta base de dados contém toda a informação de programas e programação da televisão e está constantemente a ser atualizada por uma equipa editorial.

Esta base de dados já se encontra construída e desenvolvida desde 2013, ou seja, não é necessário desenvolver nem fazer alterações nesta base de dados para o desenvolvimento deste projeto.

O ElasticSearch possui uma base de dados não relacional que depois é utilizada pelo mesmo para fazer a pesquisa sobre esses conteúdos. Para isso deve existir um sincronismo entre a base de dados "EPG" e o ElasticSearch, em que na gravação de qualquer conteúdo que interesse indexar na base de dados do ElasticSearch deve ser feita uma atualização ou adição desses conteúdos. A mesma situação para conteúdo que é removido, em que o sistema "EPG" tem de despoletar um sinal para apagar o conteúdo do ElasticSearch. Deve-se garantir este sincronismo para que a pesquisa devolva sempre resultados atuais e que possam ser disponibilizados. No entanto, o sistema deve estar preparado para resolver eventuais problemas de sincronismo (como o caso do ElasticSearch devolver um programa que já não existe no EPG). Com esta arquitetura existe uma replicação dos dados do "EPG" no ElasticSearch, onde se poderia eliminar a redundância e eliminar a base de dados da arquitetura do projeto uma vez que não é necessária. Essa não é uma boa solução por dois motivos: o ElasticSearch não persiste a data e apaga a mesma ao final de algum tempo e é sugerido pela Elastic (empresa que desenvolve o ElasticSearch) que o ElasticSearch deve ser utilizado como motor de pesquisa acoplado com uma base de dados.

Um dos requisitos para o desenvolvimento deste projeto é que os utilizadores sejam transversais, independentemente da aplicação com a qual acedem. Para isso é criada uma base de dados de utilizadores, onde é armazenada toda a sua informação. Uma das componentes mais importantes de um sistema de recomendações é o *feedback* dado pelos utilizadores. É fundamental não acoplar os utilizadores com as suas classificações, ao ponto em que se um dos utilizadores é apagado todas as suas avaliações perdem-se. Desta forma cria-se uma base de dados de interações ("UserInteractions"), onde é armazenada toda a informação de *feedback* dada pelos utilizadores. Essa informação é anónima, havendo apenas um identificador único em que não é possível identificar os dados pessoais do utilizador. Assim caso um utilizador seja removido da aplicação, as suas análises não são perdidas e podem continuar a ser utilizadas no sistema de recomendação.

Em relação à infraestrutura do projeto foi criada a solução apresentada na Ilustração 11. Um detalhe que não é possível visualizar neste diagrama é a existência de um servidor, onde se encontram todos os serviços desenvolvidos alojados, denominado de OVH1. Nesse servidor foi instalado o sistema operativo Ubuntu 16.04. Esta instalação foi prévia ao desenvolvimento deste projeto, uma vez que esta máquina aloja outros serviços da empresa, e é a razão pela qual não está representado no diagrama. De forma a que este projeto não criasse qualquer tipo de dependências na máquina, e que o seu *deployment* fosse simples, tanto noutra máquina, como numa máquina pessoal para desenvolvimento, foi utilizado contentores (mais concretamente Docker) para isolar todas as dependências necessárias para a execução do projeto. O Docker cria automaticamente uma rede entre cada uma das suas imagens, de forma a que estas consigam comunicar entre elas. Dentro do contentor encontram-se todos os serviços desenvolvidos neste projeto: "Authentication", "ProgramDetail", "Search" e

“Recommendations”. Todos estes serviços ficam encarregues de garantir que as funcionalidades desenvolvidas estão disponíveis; o “EPGUser” representa a base de dados onde vão ser armazenados todos os utilizadores enviados pelas aplicações que recebem recomendações do sistema; o nó “UserInteractions” representa a base de dados que armazena todas as interações dos utilizadores com os programas do EPG; por último, a biblioteca “SearchEngine” representa a ferramenta escolhida para a funcionalidade de pesquisa.

Dentro do contentor, encontra-se também o “nginx”, que é utilizado como *reverse proxy*, e faz a comunicação com o exterior e indica para qual dos serviços o pedido deve ser encaminhado.

Numa máquina virtual separada encontra-se o projeto do EPG, com a sua base de dados. Este projeto é onde a equipa da InfoPortugal trabalha a inserir e editar conteúdo diariamente. É à base de dados do EPG que o “ProgramDetail” vai captar a informação dos programas.

O nó “Mobile Phone” representa a aplicação móvel desenvolvida em conjunto com este projeto, que serve como uma interface para o projeto desenvolvido. Esta aplicação começou por ser desenvolvida como um projeto de estágio, no entanto foi finalizada para este projeto pela equipa da InfoPortugal, onde o autor deste documento realizou parte das tarefas.

A ligação SQL entre o serviço “ProgramDetail” e a base de dados do “EPG” é feita através da criação de um utilizador na base de dados no EPG, que tem apenas permissão de leitura dos dados. Posteriormente foi acrescentado à lista de endereços de IP permitidos o endereço da máquina onde está alojado o contentor que contém o serviço, permitindo assim acesso aos dados do EPG.

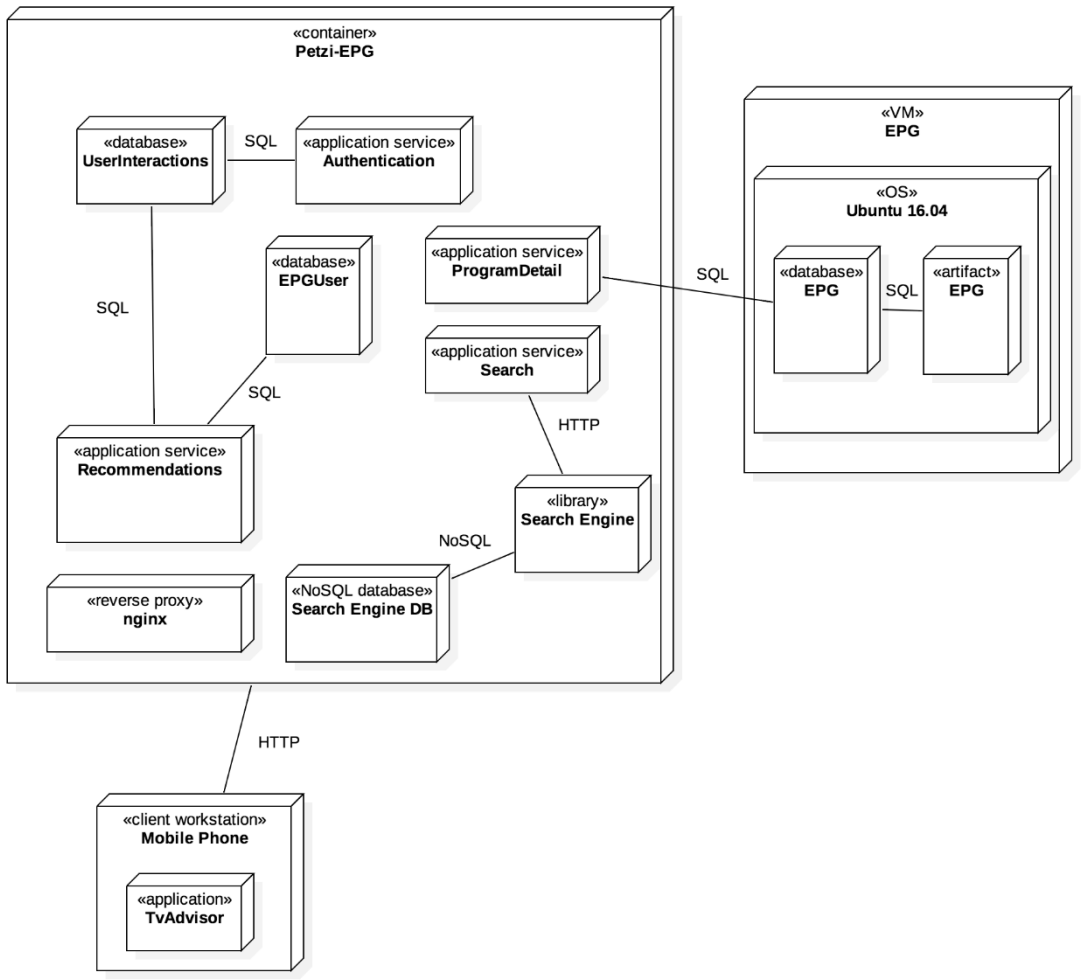

Ilustração 11 - Diagrama de infraestrutura

7 Desenho e implementação da solução

Nesta secção está documenta a solução e a forma de como esta foi implementa, que visa resolver o problema descrito nos primeiros capítulos, contendo vários artefactos para uma fácil compreensão da solução a implementar, assim como comparações com outras possíveis soluções e a justificação pela qual essas não foram escolhidas.

Detalha-se também a forma como as tecnologias adotadas foram integradas no projeto e qual a sua utilidade para o produto final.

7.1 Modelo de dados

O principal modelo de dados deste projeto é o da informação do EPG. Como este projeto só é necessário focar na componente do conteúdo da programação da televisão, é apenas documentada a parte do modelo de dados dessa área, uma vez que o FrontOffice disponibilizado à equipa editorial contém outro tipo de funcionalidades que não são contempladas para este projeto. No Anexo 6 é possível ver o modelo de dados completo do EPG.

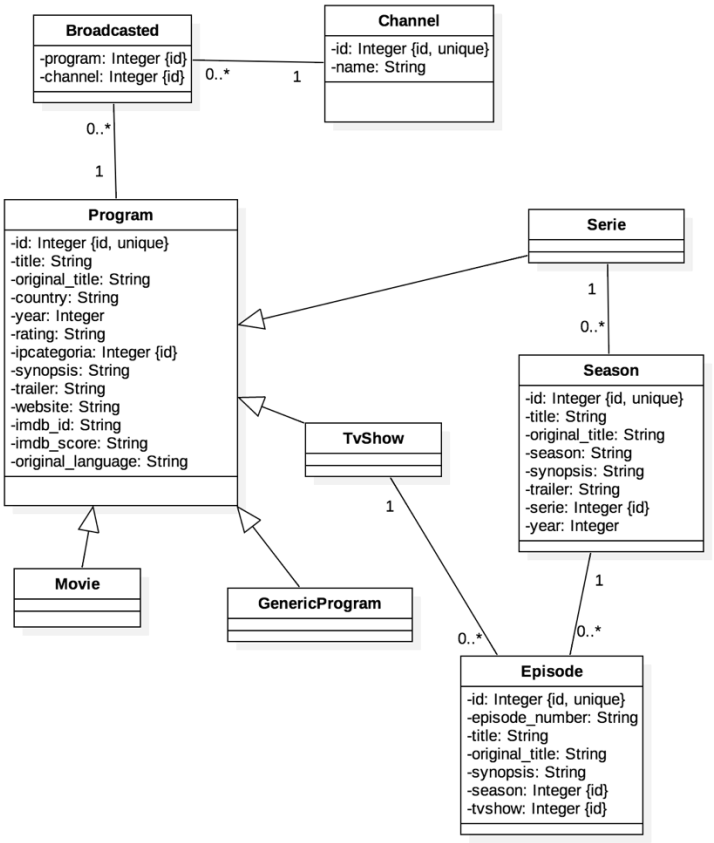

Ilustração 12 - Modelo de dados da área de programação TV

Como é possível ver no diagrama anterior existem quatro tipos de programas: “Movie”, “Serie”, “TvShow” e “GenericProgram”. Cada um destes é generalizado de uma tabela “Program”, que possui todos os atributos de um programa de televisão. Para os episódios existe a tabela “Episode”, que possui uma chave-estrangeira para uma temporada (“Season”) ou “TvShow”, indicando a que programa pertence aquele episódio. A classe temporada (“Season”) possui os atributos da própria temporada e uma associação (por chave-estrangeira) a uma série. Por último, existe uma relação entre canais e programas, com um programa a poder ser emitido por zero ou vários canais, e um canal a poder emitir zero ou mais programas.

Uma possível melhoria para a modelação apresentada na Ilustração 12 era a generalização do episódio como um programa, em que os seus atributos únicos eram depois especializados na sua tabela, e o termo “programa” passava a ser o conteúdo que é de facto transmitido. Outra possível melhoria na tabela de episódio seria o uso de uma chave-estrangeira genérica, onde esta estaria limitada para escolher uma instância de “TvShow” ou “Season”.

Outra componente bastante importante e associada a este tipo de conteúdos é o elenco dos programas. Na Ilustração 13 é exemplificado uma chave-estrangeira genérica, tendo em conta as regras aplicadas pela *framework* Django.

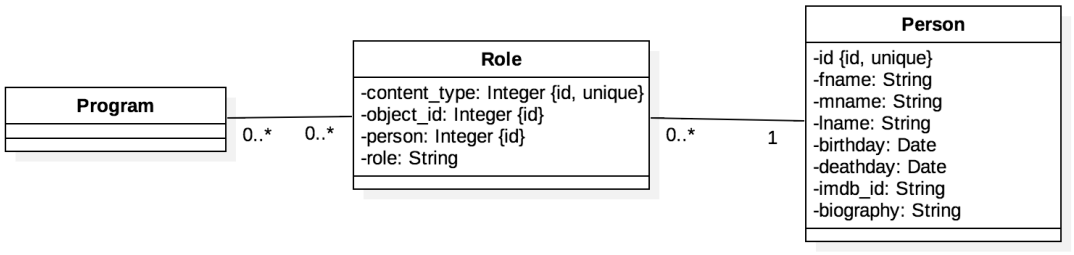

Ilustração 13 - Modelo de dados da área do elenco de um programa

Tal como se pode verificar na tabela "Role", esta serve como uma tabela intermédia entre a tabela de pessoas ("Person") e programas. Nesta tabela existem dois atributos que permitem a associação a qualquer uma das tabelas do esquema atual da base de dados através do "content_type" e "object_id": o primeiro tem um identificador para uma quarta tabela onde todas as tabelas estão registadas e associa-se a uma dessas tabelas por esse identificador; o segundo tem como intuito identificar qual é a instância da tabela escolhida (neste caso em concreto o ID do programa). Por último resta frisar a chave-estrangeira que aponta para a tabela de pessoas. O mesmo esquema é aplicado às imagens para programas.

Uma vez que as aplicações estão constantemente a efetuar pedidos, é necessário garantir que estas plataformas estão autorizadas a aceder ao conteúdo disponibilizado. Desta forma é necessária uma tabela numa base de dados que armazena todas as plataformas que possuem permissão para aceder aos dados, junto com a sua chave que não deve ser partilhada. Esta segurança é reforçada através de uma tabela de "Referer", onde são registados todos os domínios dessa plataforma que podem efetuar os pedidos. Cada plataforma tem a possibilidade de registar os seus utilizadores. O que garante que estes utilizadores são únicos é a restrição aplicada ao conjunto de plataforma com o identificador do utilizador. É da responsabilidade da aplicação que faz o pedido manter a coerência e a integridade do identificador do utilizador, uma vez que é esse identificador que permite distinguir os vários utilizadores dessa mesma plataforma. Na Ilustração 14 é possível ver o modelo de dados discutido.

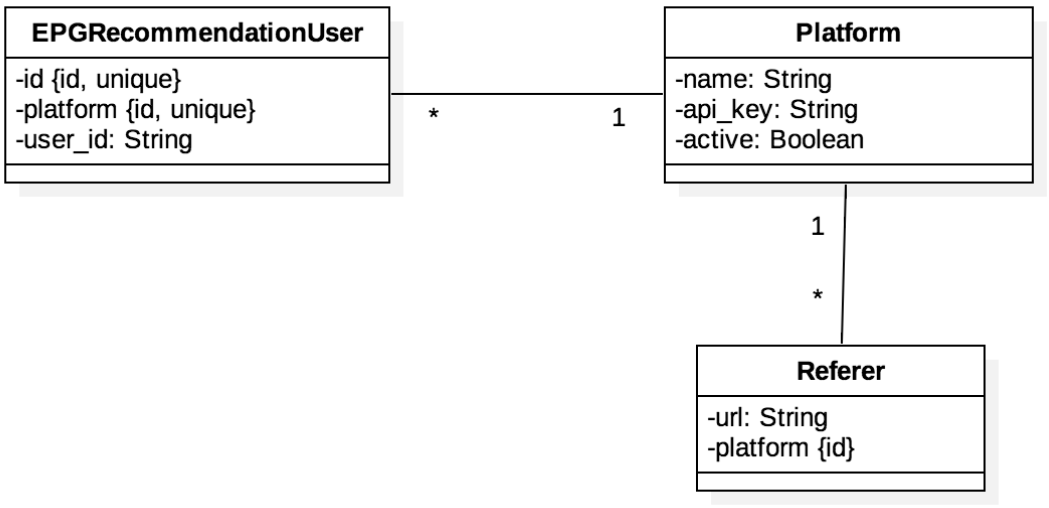

Ilustração 14 - Modelo de Dados dos utilizadores

Assim que um utilizador esteja registado no sistema, é utilizado o seu identificador único automático (gerido pela *framework* Django) para associar a todas as suas interações com programas que sejam necessárias contemplar para o sistema de recomendações. Uma vez que existem várias formas de interagir com um programa (vários tipos de feedback, como visto no capítulo Sistemas de recomendação), deve-se garantir que a tabela criada é abstrata o suficiente para permitir uma expansão de interações no futuro. Desta forma, é definido a seguinte estrutura:

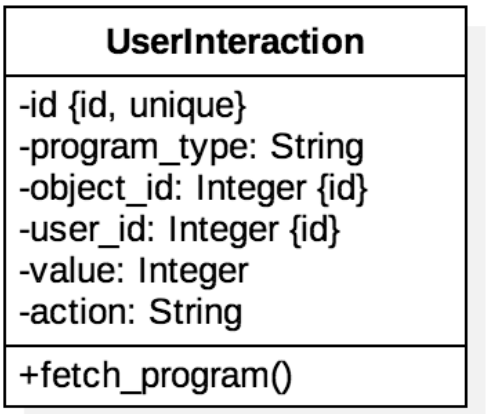

Ilustração 15 - Modelo de dados de interações de utilizadores

O conjunto “program_type” com o “object_id” serve para identificar qual o programa sobre o qual se deu uma interação. O atributo “user_id” é o identificador único da tabela “EPGRecommendationUser”, representado na Ilustração 14. O atributo “value”, representado como um número inteiro, é o valor que vai ser inserido na matriz para o sistema de recomendação. Este valor é controlado apenas por este projeto, e nunca é exposto aos clientes

que utilizam esta plataforma. A *string* "action" representa qual foi a ação desempenhada pelo utilizador "user_id". Neste momento existem apenas três valores possíveis a atribuir a este atributo: "like", "dislike" e "unknown". Com este atributo é possível utilizar a mesma ação para diferentes plataformas e com valores a utilizar no sistema de recomendação diferentes, permitindo que cada cliente tenha recomendações de acordo com o seu agrado. Este último atributo também permite facilmente identificar quais os tipos de ações desempenhadas pelos utilizadores, o que é bastante útil para efeitos estatísticos.

7.2 Alternativas

Existem várias funcionalidades relacionadas com as recomendações, como a inserção de novo *feedback* por parte do utilizador e a obtenção das recomendações para um utilizador. Essas recomendações vão ser geradas utilizando um algoritmo híbrido, que utiliza componentes do Collaborative Filtering em conjunto com o Content Based. O primeiro oferece a semelhança entre utilizadores, o que faz com os resultados sejam mais dinâmicos e que não se prendam apenas a um certo tipo de programas, no entanto, tem a falha de não conseguir gerar recomendações para programas que ainda não tenham tido interações. Essa falha é colmatada pelo CB, que através dos atributos do programa é capaz de gerar recomendações baseando-se apenas nessas propriedades.

Tendo em conta a escolha dos algoritmos de recomendação, existe apenas uma biblioteca que oferece estes dois algoritmos em conjunto: LightFM. Esta escolha recorreu na quantidade de documentação fornecida pela biblioteca (que tem mais do que alternativa Crab) e o algoritmo híbrido que o Surprise não oferece (este último foca-se mais em algoritmos do tipo CF). A disponibilidade do gestor da biblioteca para esclarecer questões também é outro ponto positivo, uma vez que o projeto ainda se encontra ativo e tem ainda alterações recentes. Outro fator importante foi o tempo disponibilizado para a realização do projeto: uma vez que se deve apresentar resultados em pouco menos de um ano, a escolha de uma biblioteca que seja de fácil utilização é um fator importante. Nesse ponto, o LightFM é uma biblioteca com uma curva de aprendizagem bastante reduzida e fácil de se integrar num projeto, mesmo para programadores com pouca experiência na área. Num outro estudo realizado em 2015, onde foram realizados vários testes à biblioteca LightFM, ficou comprovado que este é um sistema a utilizar em cenários de *cold-start* e *warm-start*, em situações em que existe uma grande quantidade de interações de utilizadores ou que exista uma grande quantidade de meta dados a utilizar (Kula, 2015).

Nenhuma destas bibliotecas oferece a possibilidade de contextualizar as sugestões para um dado utilizador. No caso particular do EPG, deve-se ter em conta que os programas recomendados devem ser transmitidos a horas que o utilizador normalmente tem acesso à televisão (por exemplo, programas que comecem às três da madrugada não são do interesse de grande parte dos telespectadores), uma vez que nem todos os utilizadores possuem uma *box* de televisão com possibilidade de gravar programas, apesar da crescente oferta das

diversas operadoras. Devido à restrição de tempo para o desenvolvimento deste projeto, a contextualização das sugestões fica como uma funcionalidade a acrescentar.

7.3 Bases de dados

Tal como foi apresentado no capítulo da arquitetura do sistema, existem quatro base de dados distintas. Uma delas serve apenas para a pesquisa, e trata-se de uma base de dados não relacional, utilizada pela tecnologia ElasticSearch. Outra base de dados deve ter apenas permissão de *read-only*, que é a base de dados do EPG, onde a sua modelação foi apresentada no capítulo anterior. As restantes bases de dados vão ser desenvolvidas neste projeto e estão ligadas entre si. A base de dados de recomendações armazena todas as interações dadas pelos utilizadores em relação a programas. Esse *feedback* é depois utilizado pelo sistema para gerar recomendações personalizadas. De forma a não se perder estes dados, esta base de dados fica separada da base de dados dos utilizadores para manter o anonimato das escolhas e guardando assim todos os dados na íntegra, que é bastante útil para o sistema de recomendações. Numa fase inicial a plataforma está apenas preparada para receber classificações de programas dadas por utilizadores, mas para trabalho futuro seria interessante expandir para receber outro tipo de interações de utilizadores, uma vez que a base de dados está modelada dessa forma.

A última base de dados guarda o identificador único dos utilizadores e a plataforma à qual estes pertencem. Também é nesta base de dados que vai ser armazenada a chave partilhada, que depois é utilizada nos pedidos à aplicação.

A relação entre a base de dados de recomendações e de utilizadores não necessita de sincronismo, ou seja, caso um utilizador seja apagado não é necessário remover todas as classificações que deu a programas, uma vez que quando apagado não há maneira de saber que foi aquela pessoa que deu aquela interação. As recomendações devem, no entanto, ser armazenadas, porque mesmo que um utilizador tenha abandonado a plataforma, as suas opiniões podem ser utilizadas para outros utilizadores que sejam semelhantes e para lhes recomendar outros programas. Ambas estas bases de dados estão desenvolvidas utilizando o PostgreSQL. Esta tecnologia foi escolhida pela fácil integração que tem com ambientes de desenvolvimento Linux e com a *framework* Django. Não oferece grandes vantagens em comparação com MySQL, no entanto, como esta tecnologia é utilizada pela empresa e na base de dados do EPG, de forma a manter a coerência manteve-se a aposta.

Uma vez que as preferências dos utilizadores são uma informação sensível, deve-se garantir que estes dados nunca serão fornecidos sem o conhecimento dos utilizadores. Dessa forma, estas bases de dados vão cumprir o novo plano de proteção de dados, a ser aplicado a partir de maio de 2018 (Comissão Nacional de Proteção de Dados, 2017).

7.4 Serviços web

Pretende-se uma plataforma que seja abrangente a qualquer tipo de dispositivo, para que qualquer tipo de aplicação fosse capaz de comunicar com esta.

Uma das primeiras decisões a tomar e que é fulcral para o desenvolvimento do projeto é a escolha da língua. Uma possível língua que é bastante utilizada na área das recomendações e científica é o R. No entanto, e como já foi referido neste documento, de forma a manter a coerência dentro dos projetos da empresa, a linguagem de programação a utilizar é o Python.

Um estilo de arquitetura bastante popular atualmente é o REST e o escolhido para o desenvolvimento dessa aplicação. Sendo assim, qualquer aplicação que queria comunicar com esta plataforma deve apenas efetuar pedidos HTTP simples que cumpram todos os requisitos e o sistema encarrega-se de devolver a resposta correta. Uma outra possível alternativa seria a utilização de SOAP. Embora sejam diferentes (uma vez que o REST é uma arquitetura e o SOAP é um protocolo) existe um grande debate e comparação entre as duas tecnologias. Neste caso seria possível utilizar o SOAP ao invés de REST, no entanto uma vantagem do último é que suporta ficheiros do tipo JSON, XML, entre outros, enquanto que o SOAP suporta apenas XML. Os ficheiros do tipo XML são maiores em comparação com o JSON, o que faz com que os pedidos e a *performance* sejam mais lentos. O REST também possui uma curva de aprendizagem mais rápida, uma vez que é mais simples do que o SOAP. A possibilidade de o REST utilizar *cache* também faz com que seja mais rápido em termos de *performance* e escalabilidade (Premraj, 2015).

Tendo em consideração a escolha do REST, pretende-se desenvolver vários *endpoints*: obtenção de informação sobre programas e pessoas; adicionar/remover interações por utilizadores; captar recomendações para um certo utilizador; e pesquisar sobre dados televisivos.

Os *endpoints* disponibilizados para a obtenção de informação de programas utilizam apenas a base de dados do EPG, e existe a possibilidade de filtrar estes resultados com alguns dos atributos dos programas.

O *endpoint* de pesquisa valida a expressão enviada pela a aplicação que está a comunicar com a plataforma e utiliza o ElasticSearch para retornar os resultados. Uma vez que o ElasticSearch possui uma comunicação através de HTTP bastante semelhante ao REST, todos os pedidos efetuados à ferramenta vão utilizar esse protocolo. Na componente de pesquisa é possível pesquisar sobre conteúdo que a InfoPortugal possui, ou seja, programas e pessoas. Ao fazer-se uma pesquisa pelo nome de uma pessoa, os resultados devem ter essa pessoa e os programas em que essa pessoa faz parte do elenco. Na pesquisa por título ou título original de um programa, deve apenas aparecer os programas, e não as pessoas que estão associadas a esse programa. De momento a InfoPortugal não possui informação suficiente para ser realizada pesquisa por nome de personagens. O ElasticSearch foi a escolha do motor de pesquisa, uma vez que é uma tecnologia que a empresa utiliza noutro tipo de projetos, e era uma das tecnologias indicadas para a resolução dos requisitos no início do projeto.

7.5 Implementação

Nesta secção vão ser apresentadas todas as tecnologias utilizadas de uma forma mais aprofundada, assim como a forma como estas foram integradas no projeto e qual a sua utilidade para o produto final.

7.5.1 Tecnologias

As três principais tecnologias a explorar neste projeto são o Django (com grande enfase na biblioteca Django REST Framework), ElasticSearch e LightFM. Cada uma desempenha um papel fulcral para a plataforma final.

O Django é a base de todo o projeto, uma vez que é a *framework* que trata de todos os pedidos feitos à plataforma, pedidos à base de dados do EPG e serializa as respostas do sistema de pesquisa e do motor de recomendações. Através da utilização da ferramenta Django REST Framework, é facilmente serializado um objeto em formato JSON, mapeando todos os seus campos de forma automática, simples e intuitiva. Possui ainda funções que permitem a listagem desses mesmo objetos serializados, assim como todas as outras operações do CRUD de forma automática.

Para a implementação do projeto em produção foi utilizada a tecnologia Docker. Através do uso desta tecnologia, é possível manter as dependências encapsuladas num único contentor, o que faz com que seja possível executar esta aplicação em qualquer tipo de ambiente. Existe várias imagens que facilitam a integração de algumas destas tecnologias, como Python, ElasticSearch e nginx.

7.5.2 Metodologia

Para metodologia de trabalho foram seguidos vários princípios ágeis no desenvolvimento. Como a InfoPortugal possui uma pequena equipa de desenvolvimento, não é seguido nenhuma metodologia ágil como o SCRUM, mas sim princípios ágeis, como a gestão de tarefas por um *team-leader*, a integração de testes e comunicação com o cliente ao longo de várias iterações, entre outros. Na Ilustração 16 é possível ver-se um exemplo básico do desenvolvimento utilizando princípios ágeis.

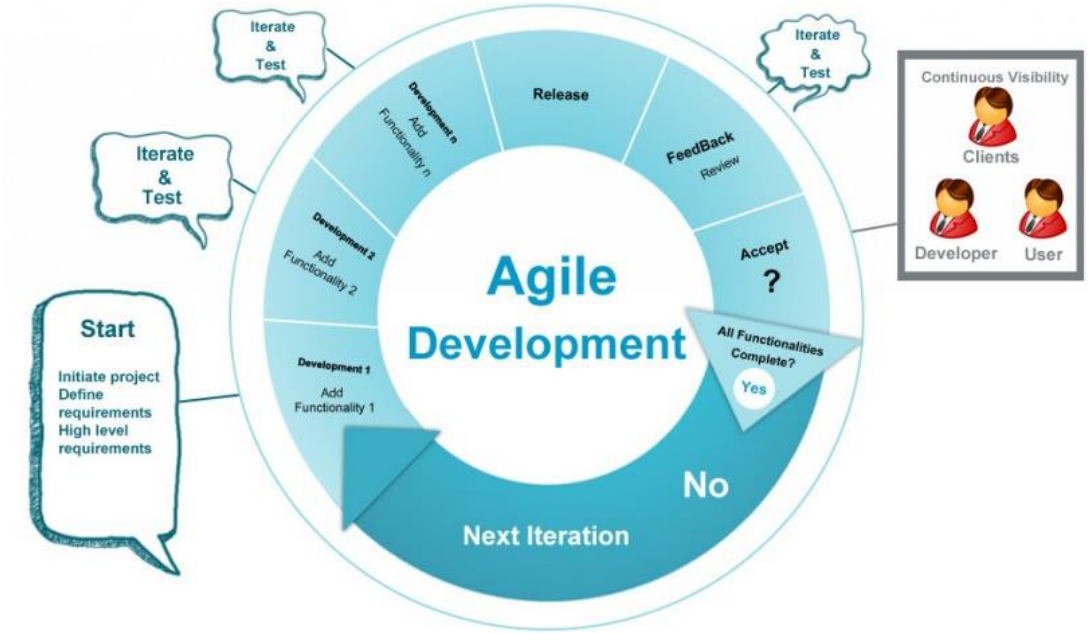

Ilustração 16 - Metodologia ágil (Fonte: https://www.asahitechnologies.com/blog/why-agile-is-the-best-alternate-methodology-to-waterfall/)

Através do uso de uma metodologia ágil, torna-se mais fácil fazer a gestão de tarefas e retornar a um passo anterior (por exemplo, regressar ao desenvolvimento de uma funcionalidade quando se está na fase de testes da mesma), ao contrário de outras metodologias como a metodologia em cascata. Foram também feitas várias apresentações ao longo do projeto aos gestores dos projetos, de forma a obter *feedback* e que outras funcionalidades acrescentar.

Uma vez que foi também desenvolvido um outro projeto que iria utilizar esta plataforma (para apresentar detalhes de programas, recomendações e pesquisa), a gestão deste projeto teve também em conta os *timings* da aplicação móvel a ser desenvolvida, e o *feedback* recebido da equipa que a desenvolveu (Amorim, 2017).

7.5.3 Pesquisar dados televisivos

A integração do Django com o motor de pesquisa ElasticSearch é facilitada através da biblioteca "elasticsearch-dsl". Esta biblioteca desenvolvida para Python facilita a integração do ElasticSearch num projeto Python, tanto na componente de desenvolver pesquisas complexas à plataforma como também na serialização do conteúdo que posteriormente irá ser pesquisado. Através da classe "DocType", é possível criar um novo tipo de objeto com uma estrutura bastante "pythonica", sendo desnecessário a criação de objetos JSON complexos para a comunicação entre estes dois sistemas. A sincronização entre estes dois componentes é bastante importante, uma vez que deve ser garantido que o elemento que está a ser pesquisado existe tanto na base de dados do ElasticSearch, como na base de dados do EPG, de forma a obter mais informação desta última. Através do uso dos sinais do Django, é possível atualizar o conteúdo da base de dados do ElasticSearch sempre que ocorrer alguma alteração nos objetos

a serem observados. Tal como é possível perceber, é utilizado aqui o padrão Observer. De forma a garantir que não existe nenhuma incoerência de dados, existe também um algoritmo que transfere toda a informação do EPG para a base de dados do ElasticSearch, garantindo assim que ambas as bases de dados estão sincronizadas.

No ElasticSearch existe apenas dois tipos de documentos armazenados: programas e pessoas. No caso dos programas existe um terceiro tipo de objeto relacionado com este que é os seus horários. Estes horários estão embutidos dentro do documento "ElasticProgram" (Ilustração 18) e poderão ser utilizados para fazer uma filtragem tendo em conta um dado intervalo de datas. Uma possível filtragem a aplicar seria uma pesquisa de texto, devolvendo apenas os programas que estão atualmente em cartaz. Para além dos horários, existe também um objeto mais simples da pessoa para criar a relação de elenco. Este objeto está associado ao programa para ser possível aparecerem programas na pesquisa onde a expressão utilizada foi o nome de uma pessoa.

Este tipo de pesquisa não deve apenas utilizar a relevância de expressão inserida pelo utilizador para encontrar os resultados certos, uma vez que pode devolver resultados que não possam ser do interesse do utilizador comum. Para isso, são utilizadas duas parcelas que ajudam a calcular a pontuação final do resultado que são a pontuação do IMDb e a popularidade de uma pessoa no The Movie Database (TMDb). Estas duas parcelas são utilizadas numa fórmula para dar relevância a programas e pessoas com um nível de popularidade maior e/ou que sejam de uma melhor qualidade para o público geral. No entanto, a parcela da fórmula utilizada na pesquisa com maior peso continua a ser a relevância da expressão do utilizador. Uma vez que os programas não possuem o valor da popularidade do TMDb, e as pessoas não estão classificadas no IMDb, na eventualidade de estes valores não existirem é utilizada a mediana de cada um destes valores da base de dados do EPG. Utilizando a mediana garante-se que caso um documento que não possua estes dois atributos não é inflacionado demais na sua pontuação, não o apresentando nos primeiros resultados.

A Ilustração 18 representa a estrutura adotada para os programas no ElasticSearch. Através da utilização do "elasticsearch-dsl", a instanciação dos objetos é muito mais simples e enquadra-se com o restante estilo da linguagem de programação, e o seu método "save()" trata da serialização destes objetos e de armazená-los na base de dados. Como nem todos os campos que estão armazenados na base de dados devem ser pesquisáveis, utiliza-se a opção "not_analyzed", associado ao atributo "índex" para utilizar estes atributos apenas como filtros. Alguns exemplos desses atributos são o ano de produção e país de origem.

A estrutura das pessoas (Ilustração 17) segue uma metodologia muito semelhante, no entanto não possui nenhum objeto embutido, como existe no caso dos programas o elenco e os horários de onde o programa vai ser emitido.

```
class ElasticPerson(DocType):
    """
    Person document in ElasticSearch.

    Right now we don't include the names separated, because we wll only search in the full name.
    """
    id_epg = Integer(fields={'raw': Integer(index='not_analyzed')})
    full_name = String()
    # fname = String()
    # mname = String()
    # lname = String()
    country = String(fields={'raw': String(index='not_analyzed')})
    tmdb_popularity = Float(fields={'raw': Float(index='not_analyzed')})
```

Ilustração 17 - Estrutura de uma pessoa na base de dados do ElasticSearch

```
class ElasticProgram(DocType):
    id_epg = Integer(fields={'raw': Integer(index='not_analyzed')})
    title = String()
    original_title = String()
    program_type = String()
    year = Integer(fields={'raw': Integer(index='not_analyzed')})
    country = String(fields={'raw': String(index='not_analyzed')})
    category = String(fields={'raw': String(index='not_analyzed')})
    original_language = String(fields={'raw': String(index='not_analyzed')})
    schedules = Nested(
        doc_class=ProgramSchedule,
        properties={
            'date': Date(fields={'raw': Date(index='not_analyzed')}),
            'channel': String(fields={'raw': String(index='not_analyzed')})
        }
    )
    team = Nested(
        doc_class=ProgramPerson,
        properties={
            'full_name': String(),
            'role': String(fields={'raw': String(index='not_analyzed')})
        }
    )
    imdb_score = Float(fields={'raw': Float(index='not_analyzed')})

    def add_person_to_program(self, person_name, role):
        """Add a new person to a ElasticProgram document."""
        self.team.append(
          {'full_name': person_name, 'role': role})

    def add_schedule_to_program(self, channel, schedule_date):
        """Add a new schedule to a ElasticProgram document."""
        self.schedules.append(
          {'channel': channel, 'date': schedule_date})
```

Ilustração 18 - Estrutura de um programa na base de dados do ElasticSearch

Com as estruturas indicadas acima, pode-se aplicar filtragem pelos programas que estão neste momento em cartaz, ou apresentar apenas programas com um certo ator.

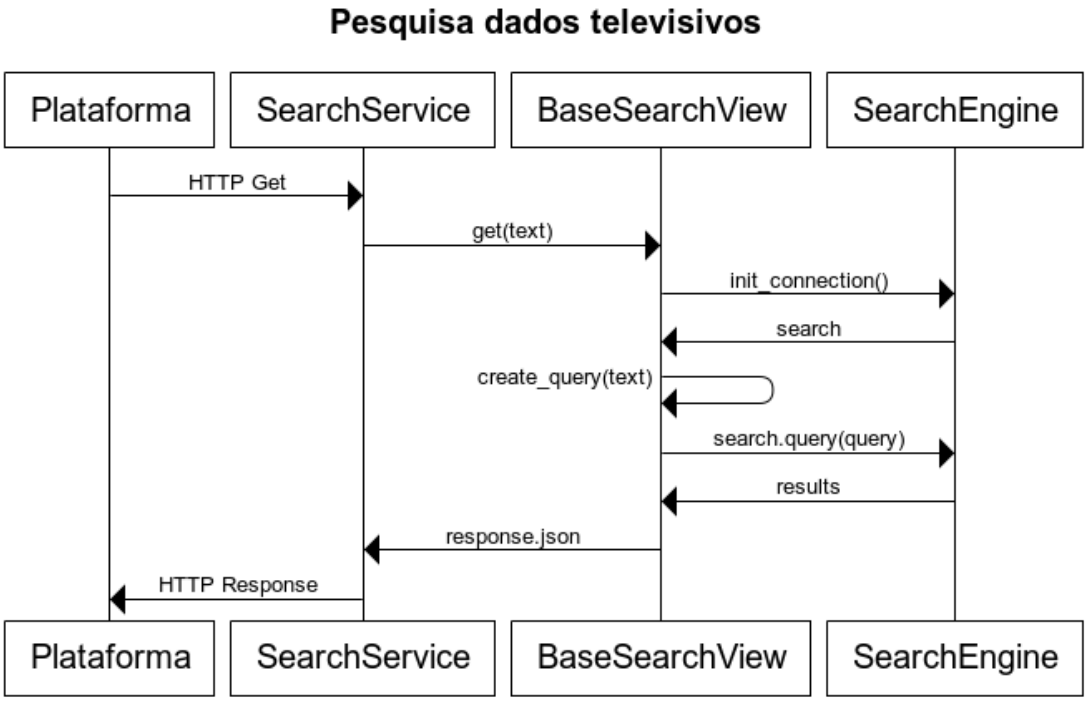

Ilustração 19 - Diagrama de sequência de pesquisa de dados televisivos

Através da Ilustração 19 consegue-se perceber a sequência de ações realizadas pelo sistema para a funcionalidade. É recebido um pedido HTTP no serviço de pesquisa. O controlador do Django reencaminha o pedido para a *view* correta ("BaseSearchView"). Esta conecta-se com o ElasticSearch através do método "init_connection()", criando uma instância do tipo "SearchEngine", representada pelo objeto "search" na figura. Através dos parâmetros recebidos no pedido, é construída uma *query* para o ElasticSearch, que depois é executada pela ferramenta e devolve os seus resultados em formato JSON. De momento os filtros disponíveis permitem filtrar a pesquisa por um intervalo de datas, ou qualquer um dos atributos presentes nos documentos criados.

7.5.4 Visualizar detalhe de programa

Uma vez que uma das grandes funcionalidades a desenvolver é a recomendação de programas, deve haver uma funcionalidade que permita ver toda a informação desse programa, para disponibilizar ao utilizador final.

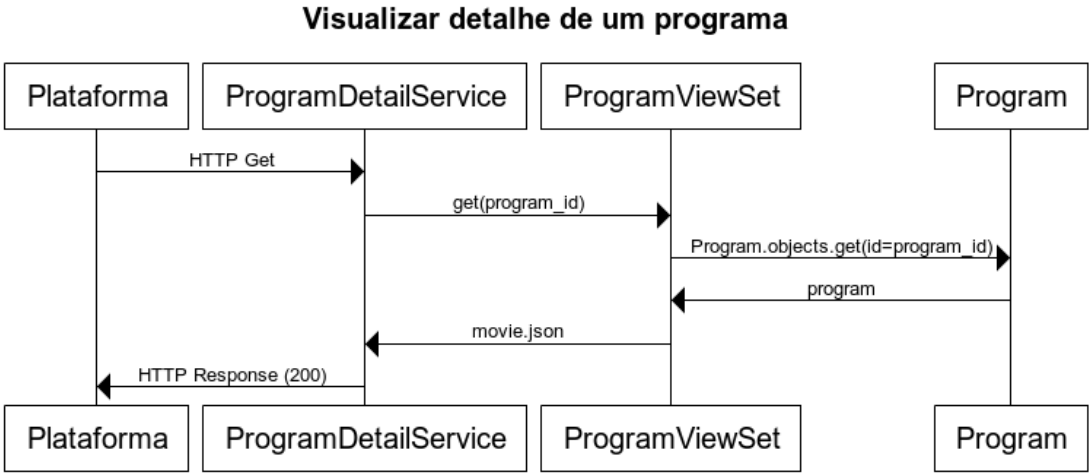

Ilustração 20 - Diagrama de sequência para a visualização de um programa

Esta funcionalidade é facilmente alcançada através dos métodos e classes disponibilizadas pela biblioteca Django REST Framework. Através desta biblioteca pode-se estender a classe "ModelViewSet", que automaticamente cria métodos de listagem e visualização de um único objeto de um certo tipo de classe. Através do "program_id" enviado no pedido pela plataforma, é realizada uma *query* à base de dados com esse mesmo identificador. Esse objeto é depois serializado para ser enviado para a plataforma, num HTTP Response.

Este mesmo método possibilita a listagem de programas, caso não seja especificado o identificador do programa. Também é possível aplicar um filtro, ao adicionar "now-playing" ao final do *endpoint*, onde é apresentada a lista de todos os programas que se encontram neste momento em cartaz.

Existe também a possibilidade de filtrar os resultados apresentados na listagem de programas através de atributos dos programas. É possível verificar os filtros disponíveis na documentação da API, e na Ilustração 27.

7.5.5 Enviar feedback

O envio de *feedback* de um utilizador pela plataforma é crucial para a maturação do algoritmo de recomendação e para a satisfação do utilizador, uma vez que se as suas interações não são consideradas, as recomendações não vão ser personalizadas. Tendo em consideração os vários fatores que já foram descritos ao longo deste documento, foi encontra a solução apresentada na Ilustração 21.

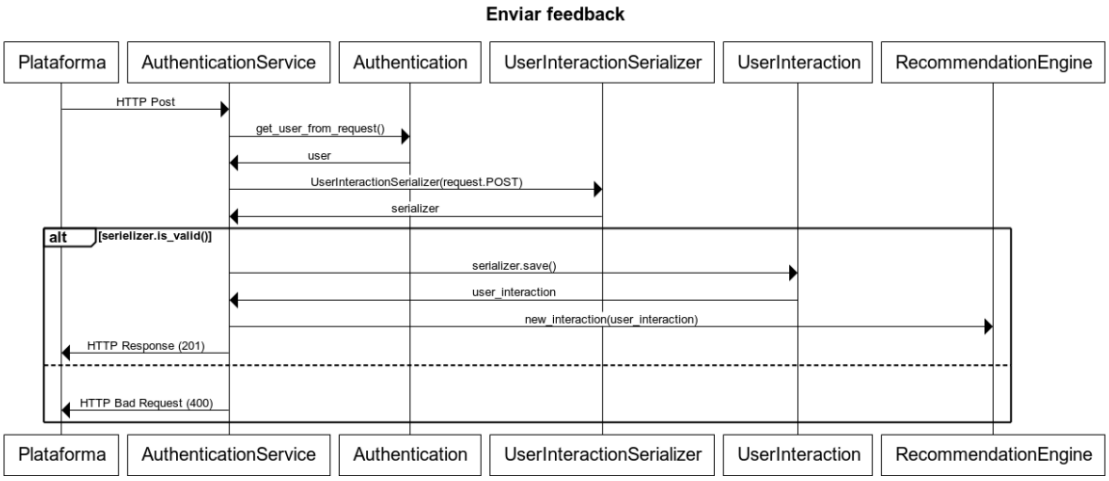

Ilustração 21 - Diagrama de sequência para enviar feedback

Um dos métodos recorrentes ao longo das várias funcionalidades é o de obter o utilizador enviado pela plataforma. Uma vez que não se trata do utilizador que comunica diretamente com API, a utilização de um *token* de autenticação acaba por se tornar mais complexa. Neste momento, a plataforma fica encarregue de enviar um identificador único para o seu utilizador e deve ser da responsabilidade da plataforma manter a unicidade e a integridade do identificador do utilizador. Através desse identificador único, e sabendo qual a plataforma que efetuou o pedido, é possível captar o utilizador a ser usado. Tendo acesso ao utilizador, deve-se agora tratar da informação enviada no pedido.

Tal como já foi referido neste documento na secção 2.1.2, podem existir vários tipos de interações por parte dos utilizadores, pelo qual foi definido que uma interação do utilizador iria ser o mais abstrata possível. Neste momento existem três tipos de ações que estão a ser tratadas pelo projeto: “like”, “dislike” e “unknown”. Dependendo da plataforma e da ação despoletada pelo utilizador, esse valor é convertido num número e depois armazenado na base de dados (através da ação do “serializer.save()”). A serialização é feita para garantir que os dados enviados estão corretos, e são validados antes de serem armazenados. Essa interação é depois adicionada ao sistema de recomendações para ser considerada na próxima previsão de recomendações para qualquer utilizador.

Existe ainda um outro *endpoint* (“/get_user_interactions/”), onde, a partir de um pedido HTTP GET, são apresentadas todas as interações que um utilizador teve com o sistema. É possível filtrar essas interações pelo tipo de ação que foi enviada no pedido. Este *endpoint* foi desenvolvido meramente para a aplicação móvel a ser desenvolvida em conjunto com este projeto, de forma apresentar os gostos e não gostos de um certo utilizador.

7.5.6 Recomendar programa

Para o sistema de recomendação era necessário garantir que a biblioteca a ser escolhida garantia certos requisitos, tendo em conta o objetivo deste projeto. Para isso foram avaliadas

várias bibliotecas que ofereciam um sistema de recomendações até ter sido escolhido o LightFM.

O LightFM utiliza matrizes como o seu principal parâmetro para treinar um modelo, de forma a sugerir recomendações baseadas neste modelo. Deve então ser construída uma matriz que representa todas as interações dos utilizadores com todos os programas. Esta matriz deve ser construída assim que o sistema esteja disponível e sempre que houver novas interações de utilizadores ou novos programas, esta deve atualizar o valor na matriz, para que as próximas recomendações tenham em conta essa interação. Existe uma segunda matriz que retrata o perfil dos programas. Esta matriz está dividida pelos atributos dos programas da base de dados do EPG. No desenvolvimento do projeto foram considerados os seguintes atributos:

- Categoria e Subcategoria (p. ex.: Categoria – Filmes; Subcategoria: Ação/Aventura);
- Elenco, dividido por realizador e os dois atores principais do programa (cada programa tem o seu elenco ordenado pela relevância);
- Época do programa, onde é utilizado o ano do programa para o enquadrar numa certa época. Neste caso são utilizados intervalos de cinco anos, onde cada programa se enquadra num deles;
- País em que o programa foi criado.

Tendo acesso às duas matrizes, é possível utilizar a biblioteca LightFM com um algoritmo hibrido de recomendações, que utiliza o CF para a semelhança entre utilizadores e o CB para a semelhança entre os atributos dos programas.

Uma vez que se está a trabalhar numa base de dados que está constantemente a ser utilizada, é importante garantir que existe um identificador único de programa na matriz. Um possível atributo a utilizar para garantir a integridade seria a utilização do ID incremental, que é único e utilizado como chave primária nos programas. No entanto, esse identificador possui o problema de criar "espaços" que nunca serão preenchidos na matriz, uma vez que o último ID de programa na base de dados encontra-se perto dos 300 mil, e existem, neste momento, cerca de 110 mil programas. Isso deve-se ao facto da equipa editorial apagar programas que já não são necessários, ou que surgiram como erro de editorial, e o Django incrementa automaticamente através do último número e não reutiliza anteriores. Dessa forma, deve ser feito um mapeamento dos IDs dos programas, onde é criado um *map* que indica a posição do programa da matriz, e onde o seu valor é o ID desse mesmo programa. Esse mesmo mapeamento é feito de forma semelhante para a matriz dos atributos, uma vez que cada atributo tem a sua posição na outra matriz. No caso dos atributos, foram utilizadas várias formas para garantir que nenhum atributo se sobrepõe a outro. Dessa forma, são utilizados os nomes das categorias e subcategorias; o ISO 3166-1 alpha-2 (International Organization for Standardization, s.d.) para a identificação dos países; e no caso do elenco, para identificar o ator é concatenado "a" no início da chave primária, e no realizador o "d". O ano em que programa foi realizado é assim o único valor que é totalmente numérico e que não se sobrepõe a qualquer

outro. Outro fator importante é garantir que é utilizada a mesma chamada à base de dados na construção da matriz e na declaração do mapeamento, uma vez que a base de dados com a qual se está a trabalhar está em constante mudança e podem ser apagados programas entre as duas ações. Existem então duas alternativas para resolver este problema: a utilização da mesma *query* para ambas as ações; utilizar uma única transação para garantir que o estado da base de dados não altera. Foi escolhida a primeira alternativa uma vez que torna o código mais legível e é a solução mais simples de implementar.

As duas matrizes criadas devem ter um tamanho definido na sua declaração. A matriz que possui a interação dos utilizadores é representada por $M_{u,p}$, onde "u" representa os utilizadores, e "p" os programas do EPG. O tamanho definido na sua declaração é do número de utilizadores existentes no sistema, com uma margem de 1000. Utilizando esta margem na declaração da matriz possibilita que não seja necessário estar a construir a matriz sempre que entra um novo utilizador no sistema, e assim o utilizador vai apenas ocupar uma linha que já estava definida. Sempre que o número de linhas e/ou colunas é ultrapassado, é feita novamente a construção da matriz e o treino do algoritmo. Como este processo é custoso, e tem efeitos na geração de recomendações, é utilizada a margem para evitar precocemente esse problema. Para o número de programas, é utilizado o número total de programas, utilizando novamente uma margem, para salvaguardar a criação de novos programas no EPG. A matriz dos atributos de programas tem uma declaração semelhante, mas os programas representam as linhas da matriz e os atributos dos programas as colunas. Cada atributo tem um peso a si atribuído, que deve ser regulado de forma a aumentar a precisão das recomendações. No desenvolvimento deste projeto, ficou definido dar-se um maior peso aos atributos de realizador (cinco) e atores principais (dois). No entanto, deve haver um maior estudo na precisão do algoritmo, para encontrar a melhor distribuição dos pesos sobre os atributos dos programas. Na Ilustração 22 é possível ver a sequência de ações na criação de ambas as matrizes.

```
def __init__(self, *args, **kwargs):
    """Construct the default Matrix for EPG or a new one with UserInteractions queryset."""
    # Get size of the features matrix
    programs_count = get_first_value('epg', db_queries.COUNT_PROGRAM)
    role_programs_count = get_first_value('epg', db_queries.COUNT_ROLES_PROGRAM)

    self.matrix_features = lil_matrix(
        (
            programs_count + self.number_of_additional_programs,
            role_programs_count + self.number_of_additional_features
        ),
        dtype=np.int32)

    # Get the number of users that uses the EPG Recommendation API
    users_count = get_first_value('epg_users', db_queries.COUNT_EPG_USERS)

    self.matrix = lil_matrix(
        (
            users_count + self.number_of_additional_users,
            programs_count + self.number_of_additional_programs
        ),
        dtype=np.int32)

    # This creates the mapping for the matrix between users and programs interactions
    self.program_mapping = Mapping()
    self.program_mapping.create(programs_count + self.number_of_additional_programs)
    # This creates the mapping for the program attributes matrix
    self.features_mapping = Mapping()
    self.features_mapping.create(role_programs_count + self.number_of_additional_features)
    # Fill Feature Program Matrix
    self.fill_features_matrix()
    # Fill User Interaction matrix
    self.fill_user_interactions_matrix()
    print 'Done making Matrix'
```

Ilustração 22 - Construção da matriz de interações e atributos de programas

De forma a otimizar o preenchimento da matriz foram utilizadas *queries* de SQL diretamente à base de dados, uma vez que utilizando o ORM do Django causa algum *overhead* aos pedidos, o que prejudica o tempo de criação da matriz. Na Ilustração 23 é possível ver o exemplo de uma *query* que capta todos os programas e os atributos necessários para a criação da matriz dos atributos.

```
ALL_PROGRAMS_ATTR = (
    """
    SELECT
        "epguide_program"."id",
        "epguide_program"."country",
        "epguide_program"."year",
        "ipgenre"."gname",
        "ipgenre"."sgname"
    FROM "epguide_program"
    LEFT OUTER JOIN
        "ipgenre"
        ON
        ("epguide_program"."ipcategoria_id" = "ipgenre"."id")
    WHERE (NOT ("epguide_program"."rating" = 'Adultos') AND NOT ("ipgenre"."gname" = 'Adultos' AND "ipgenre"."gname" IS NOT NULL))
    """
)
```

Ilustração 23 - Query para captar todos os programas e os seus atributos para preencher a matriz de atributos

Uma vez instanciado o sistema de recomendações, a plataforma está disponível para receber pedidos para gerar recomendações para um certo utilizador. A primeira validação é obter o número de interações que esse utilizador possui no sistema. Caso o número de interações seja inferior a 10, é utilizada uma lista de programas dinâmica, que retorna *n* programas em cartaz, com categorias e tipos de canais distintos, ordenados pela sua pontuação de IMDb. Esta lista serve para traçar um primeiro perfil ao utilizador, usando uma diversidade alta de programas, de forma a dar a conhecer ao sistema as preferências do mesmo.

Para o caso do utilizador possuir 10 ou mais interações com o sistema, é efetuado um pedido ao LightFM (representado no diagrama como "RecommendationEngine") para prever para esse mesmo utilizador quais os programas que esse pode gostar. O LightFM retorna uma lista com a pontuação de cada programa para aquele utilizador, sendo feita a filtragem e a ordenação para apresentar os programas que mais se enquadram. De seguida, essa lista de programas é filtrada para apresentar apenas os programas que se encontram neste momento em cartaz. Essa lista de programas é depois serializada pela biblioteca Django REST framework no formato JSON para ser enviado num HTTP Response. Este serializador envia todos os detalhes fundamentais do programa que sejam necessários numa plataforma, como imagens, titulo, titulo original, ano, elenco e quando/onde é transmitido em cartaz. É possível ver toda esta sequência de ações no diagrama de sequência ilustrado na página 79.

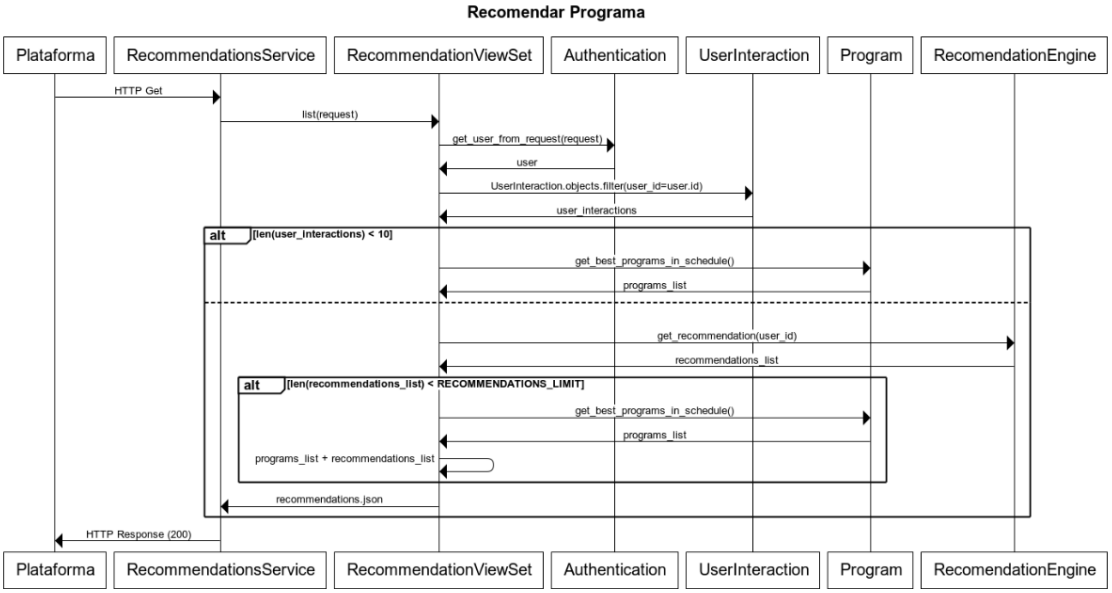

Ilustração 24 - Diagrama de sequência para recomendar programas

Este *endpoint* permite ainda filtrar por um intervalo de datas a definir pela aplicação que efetua o pedido, abrangendo assim recomendações que não se enquadram com o que está em cartaz.

7.5.7 Considerações gerais

Uma vez que esta aplicação não deve ser disponibilizada ao mundo exterior, deve-se garantir o mínimo de permissões de acesso a este conteúdo. Desta forma, foi desenvolvida uma permissão específica para esta aplicação, através dos métodos e classes disponíveis no Django REST Framework. Sempre que é realizado um pedido a este sistema, antes de executar a funcionalidade do pedido, é verificada a autenticidade de quem fez o pedido, através do seu endereço e da chave enviada. Este processo pode ser compreendido através da Ilustração 25.

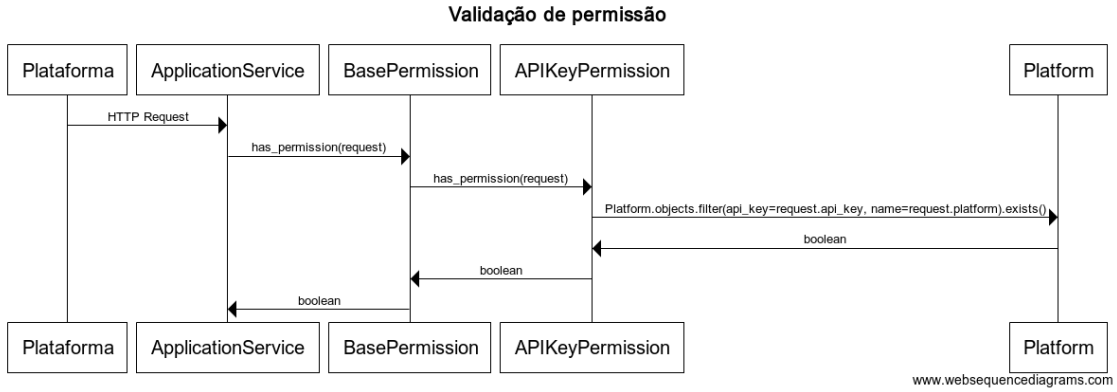

Ilustração 25 - Diagrama de sequência de validação de permissão

A documentação de uma API é um requisito fundamental, uma vez que esta deve ser sempre consultada por qualquer entidade que a utilize, de forma a esclarecer questões sobre o funcionamento da mesma e que tipo de informação vai obter.

Para a documentação da API foi então criada uma interface, onde é possível consultar todos os pedidos existentes nesta API, assim como o que cada método realiza. Para a criação desta interface foi utilizada a biblioteca "django-rest-swagger", que através dos diversos comentários no código, é capaz de gerar uma interface com todos os *endpoints* disponibilizados pelo Django REST Framework e a sua documentação. Na Ilustração 26 é possível ver quais os pedidos disponíveis para a componente de recomendação na plataforma.

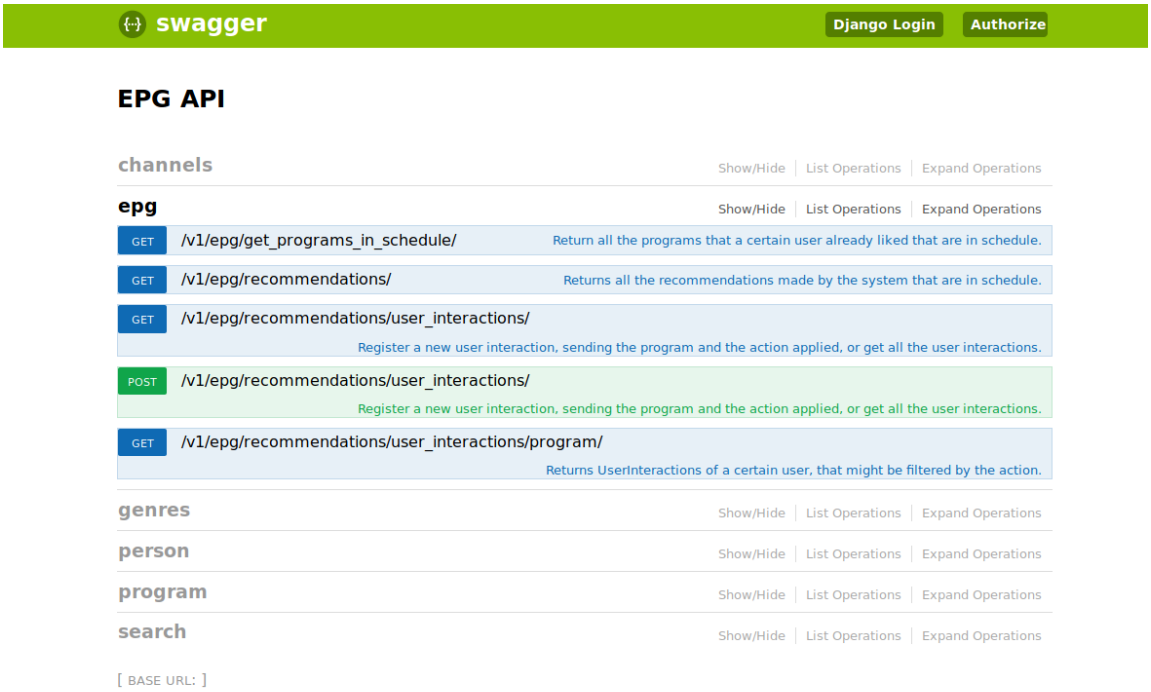

Ilustração 26 - Interface com a listagem dos endpoints do sistema de recomendação

Existe também a possibilidade de ver com mais detalhe o que um pedido faz, e que tipo de parâmetros são necessários. No exemplo da Ilustração 27 é possível ver os diversos filtros a poder aplicar numa listagem de programas.

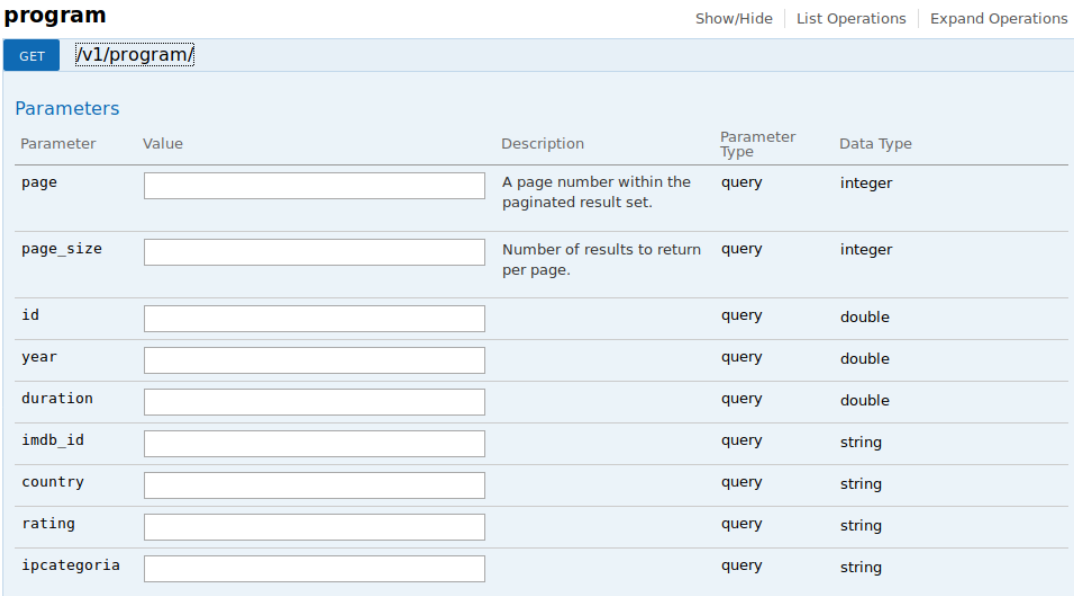

Ilustração 27 - Detalhe do pedido de listagem de programas

Para além da documentação é necessário definir um *standard* no envio da informação proveniente da API. De forma a melhorar a navegação sobre os dados provenientes dos serviços e definir uma estrutura base para essa informação foi utilizado o Hypertext Application

Language (HAL), que permite a ligação entre os vários recursos existentes na API. Utilizando o HAL é possível tornar as respostas da API mais *user-friendly*, uma vez que um dos principais objetivos do HAL é fazer com que os utilizadores consigam navegar pelos recursos utilizando *links*. Estes URLs podem ser a página anterior ou seguinte de uma listagem de objetos, como também a página de detalhe de um certo recurso. Tendo em consideração que muitos dos *endpoints* desenvolvidos neste projeto devolvem conjuntos de objetos, é importante garantir que é feita uma paginação simples de utilizar e que a aplicação que efetuou o pedido consiga aceder à página de detalhe destes objetos (Kelly, 2013). Na Ilustração 28 é possível ver a representação de um recurso segundo o HAL. Os *embedded resources* representam objetos que estão ligados com o objeto serializado. Um exemplo, utilizando o projeto atual, seria a categoria de um programa, uma vez que a categoria é uma chave estrangeira de um programa.

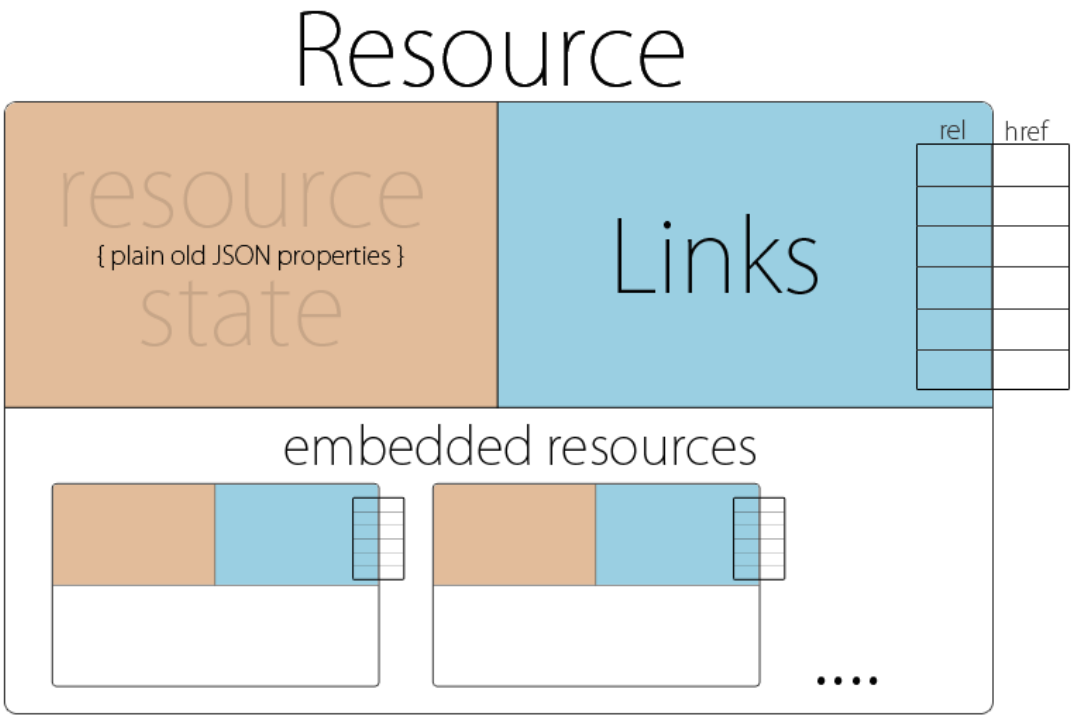

Ilustração 28 - Representação de um recurso de acordo o HAL

Para a utilização do HAL no projeto atual foi utilizada a biblioteca "django-rest-hal", que utilizando as classes oferecidas pela biblioteca Django REST Framework como base, oferece as funcionalidades do HAL numa nova classe denominada "HalModelSerializer". Esta biblioteca acrescenta então um método diferente de serialização, ao que já é oferecido no Django REST Framework.

Na imagem seguinte pode-se ver um exemplo de uma resposta da plataforma, com exemplos de ligações entre os vários recursos disponíveis na aplicação.

```
{
    "count": 448,
    "next": "http://37.59.20.11/v1/epg/recommendations/?api_key=1e5288fc2df38db555009c1ba51bb35412a7107f&page=2&user_id=tiagofmc94%40gmail.com"
    ,
    "previous": null,
    "results": [
        {
            "_links": {
                "url": "http://37.59.20.11/v1/program/876/"
            },
            "id": 876,
            "title": "A Vila",
            "original_title": "The Village",
            "year": 2004,
            "program_images": {
                "image_url": "http://epg.infoportugal.info/media/files/images/68/d3/b69153c4_8mgFYga.jpg"
            },
            "score": 39.407325744628906,
            "interaction": "Unknown",
            "_embedded": {
                "ipcategoria": {
                    "gname": "Filmes",
                    "sgname": "Drama"
                }
            }
        },
        {
            "_links": {
                "url": "http://37.59.20.11/v1/program/10438/"
            },
            "id": 10438,
            "title": "Sinais",
            "original_title": "Signs",
            "year": 2002,
            "program_images": {
                "image_url": "http://epg.infoportugal.info/media/files/images/79/34/MV5BNDUwMDUyMDAyNF5BMl5BanBnXkFtZTYwMDQ3NzM3
                    ._V1_SX300_us4LcyF.jpg"
            },
            "score": 36.32673645019531,
```


Ilustração 29 - Exemplo de uma resposta do serviço de recomendações

7.6 Prova de conceito

Como prova de conceito, e como já foi referido ao longo deste documento, foi utilizada uma aplicação *mobile* para os sistemas iOS e Android, que usufrui das funcionalidades deste projeto. Esta aplicação é de grande importação para este projeto, uma vez que serve como uma interface para o utilizador comum poder interagir com este sistema. É também esta aplicação que será apresentada a futuros clientes, uma vez que a apresentação de uma plataforma REST não é algo visualmente apelativo.

Esta aplicação foi desenvolvida como um projeto de estágio, e depois concluída em conjunto com este projeto. Era importante garantir que ambos os projetos crescessem em conjunto e estivessem sincronizados.

Foi utilizada a tecnologia React-Native, que permite a utilização do mesmo código fonte para ambas as plataformas, fazendo com o desenvolvimento seja mais rápido e a curva de aprendizagem menor.

A aplicação apresenta ao utilizador quais as recomendações (Ilustração 30) que possui para o mesmo e que se encontram neste momento em cartaz, e permite ao utilizador classificar os programas através da interação de gosto/não gosto. É possível ainda ver o detalhe destes mesmos programas (Ilustração 31), de forma a mostrar qual o tipo de informação que é disponibilizada por esta plataforma.

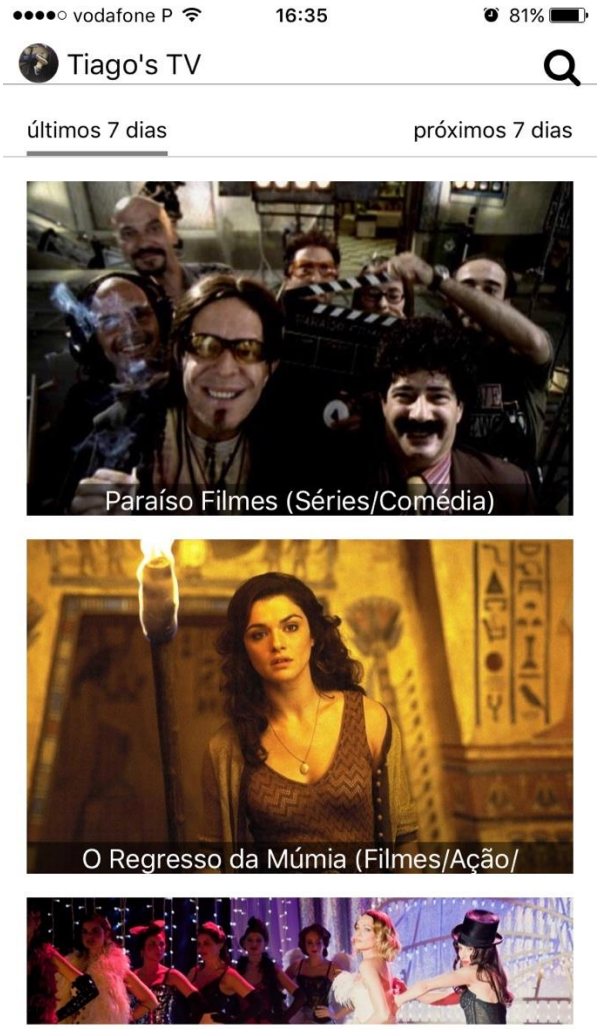

Ilustração 30 - Recomendações para um utilizador que se encontram em cartaz

Ilustração 31 - Detalhe de um programa

7.7 Testes

Um dos princípios básicos na aprendizagem de programação é que o código desenvolvido deve ser testado, para garantir que numa passagem a produção nenhuma componente foi afetada pelo novo código desenvolvido e que todas as situações foram testadas e avaliadas no caso de algum tipo de falha. O próprio criador do Django admite que código sem testes está mal modelado (Kaplan-Moss, 2009).

Existem dois tipos de testes que foram desenvolvidos ao longo deste projeto:

- Teste unitários, que servem para validar se componentes/unidades individuais comportam-se da forma como foram modelados (Software Testing Fundamentals, s.d.);

- Testes funcionais, utilizados para verificar que as funcionalidades do sistema funcionam como é pressuposto (Software Testing Fundamentals, s.d.).

Atualmente várias ferramentas e linguagens de programação oferecem inúmeros componentes que facilitam na criação de testes. Neste caso, foram utilizadas as componentes já integradas no Python denominadas de PyUnit. Estes testes podem ser utilizados para avaliar se certos *endpoints* se comportam da forma correta tendo em consideração os parâmetros enviados, ou para testar componentes individuais e garantir que a sua modelação está correta e abrange todas as alternativas possíveis.

O PyUnit incorpora alguns conceitos fundamentais na criação de testes (Python, s.d.):

- *test fixture*, que representa a preparação que deve haver para um ou mais testes. A criação ou seleção de uma base de dados enquadra-se neste ponto;

- *test case*, correspondente à unidade mais pequena de teste. É utilizada para verificar uma certa resposta para um dado *input*;

- *test suite*, que representa um conjunto de *test cases*, *test suites*, ou ambos;

- *test runner,* componente responsável pela execução de todos os testes e apresentar o resultado ao utilizador.

O próprio Django possui componentes de testes que: cria automaticamente base de dados de teste; cria um cliente onde podem ser executados pedidos; entre outros processos automatizados. No entanto, utilizando este tipo de testes, e ao gerar uma base de dados de testes através desta forma, o processo de execução de testes torna-se demasiado lento. Numa segunda tentativa foi criada uma base de dados a ser utilizada pelos testes antes da execução, mas devido às validações feitas em tempo de execução o processo continuou a ser bastante moroso.

Como o projeto desenvolvido é dinâmico no tipo de conteúdo que é apresentado e depende de uma base de dados para apresentar recomendações e resultados de pesquisa, optou-se por espelhar a base de dados do EPG. Com este processo é possível executar qualquer tipo de testes sobre esta, sem afetar a base de dados principal. Através de uma configuração do Django é possível que todas as transações da base de dados de produção sejam replicadas na base de dados espelhada, pelo que esta está sempre atualizada.

Nos testes funcionais, existiu um grande foco para garantir que caso o cliente esteja a enviar os argumentos errados para obter qualquer tipo de informação, a API deve responder de forma a dar informação suficiente ao sistema sobre o erro ocorrido, e que esta não fique indisponível.

Logo todos os parâmetros recebidos em todos os *endpoints* foram devidamente testados, de forma a garantir que não exista qualquer *downtime*, e que o erro de reportado é o correto.

```
def test_program_formatting(self):
    # Check list of program list formatting
    data = {'programs': '1,2+3,5', 'user_id': 'tiagofmc94@gmail.com'}
    response = requests.get(
        self.api_url + '/v1/epg/recommendations/user_interactions/?api_key=' + self.api_key, data)
    self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST,
                     'GetUserInteractionsTests: Formatação do programa')
    data = {'programs': '1,a,5', 'user_id': 'tiagofmc94@gmail.com'}
    response = requests.get(
        self.api_url + '/v1/epg/recommendations/user_interactions/?api_key=' + self.api_key, data)
    self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST,
                     'GetUserInteractionsTests: Formatação do programa')
    data = {'programs': '1,,,,', 'user_id': 'tiagofmc94@gmail.com'}
    response = requests.get(
        self.api_url + '/v1/epg/recommendations/user_interactions/?api_key=' + self.api_key, data)
    self.assertEqual(response.status_code, status.HTTP_400_BAD_REQUEST,
                     'GetUserInteractionsTests: Formatação do programa')
    data = {'programs': '1', 'user_id': 'tiagofmc94@gmail.com'}
    response = requests.get(
        self.api_url + '/v1/epg/recommendations/user_interactions/?api_key=' + self.api_key, data)
    self.assertEqual(response.status_code, status.HTTP_200_OK, 'GetUserInteractionsTests: Formatação do programa')
```

Ilustração 32 - Teste funcional para validar o parâmetro de lista de programas

Na obtenção de recomendações não existe um grande número de componentes individuais que possam ser testadas, uma vez que grande parte das operações que ocorrem são chamadas à base de dados, ou processos de serialização que são tratados automaticamente através da biblioteca Django REST Framework.

Existe, no entanto, inúmeras funções que devem ser validadas na criação da matriz e do mapeamento do sistema de recomendações. É importante garantir que o mapeamento criado não ultrapassa os limites da matriz, que este é bem executado e armazena a informação da forma correta. A criação da matriz é a base para uma das principais funcionalidades do sistema, logo a cobertura destas funções deve ser alta.

```
class MappingTests(unittest.TestCase):
    def setUp(self):
        self.mapping = Mapping()

    def test_mapping(self):
        # This creates the mapping for the matrix between users and programs interactions
        self.mapping.create(10)
        self.mapping.add('a')
        self.assertEqual(self.mapping.get('a'), 0)
        self.assertEqual(self.mapping.get_reverse(0), 'a')
        for i in range(10):
            self.mapping.add(i)

    def test_out_of_bounds(self):
        # This creates the mapping for the matrix between users and programs interactions
        self.mapping.create(1)
        for i in range(2):
            self.mapping.add(i)
        with self.assertRaises(MappingOutOfBounds):
            self.mapping.add('out_of_bounds')

    def test_get_unexistent_elem(self):
        # Get element that doesn't exist on the mapping
        self.mapping.create(1)
        self.mapping.add('a')
        self.assertEqual(self.mapping.get('b'), None)
        self.assertEqual(self.mapping.get(0), None)

    def test_get_unexistent_position(self):
        # Get element that doesn't exist on the mapping
        self.mapping.create(1)
        self.mapping.add('a')
        self.assertEqual(self.mapping.get_reverse(2), None)
        self.assertEqual(self.mapping.get_reverse(1), None)
```

Ilustração 33 - Testes aplicados à classe de mapeamento usada na construção de matrizes

Para além do foco do mapeamento e criação é importante garantir que a obtenção dos atributos para a matriz de atributos de programa decorre da forma prevista, uma vez que se trata de uma grande quantidade de atributos a utilizar, que tende a crescer cada vez mais com o aumento de conteúdo. Como exemplo neste documento é apresentado os testes realizados à obtenção da lista de anos, separada de cinco em cinco anos, na Ilustração 34.

```
class ListOfYearsTest(unittest.TestCase):
    def test_list_of_years_step_5(self):
        # Test the limit of the 'first_year'
        last_year = 2016
        self.assertEqual(get_list_of_years(1999, last_year), [1995, 2000, 2005, 2010, 2015])
        self.assertEqual(get_list_of_years(2000, last_year), [2000, 2005, 2010, 2015])
        self.assertEqual(get_list_of_years(2001, last_year), [2000, 2005, 2010, 2015])
        last_year = 2015
        self.assertEqual(get_list_of_years(2000, last_year), [2000, 2005, 2010, 2015])
        last_year = 2014
        self.assertEqual(get_list_of_years(2000, last_year), [2000, 2005, 2010])
```

Ilustração 34 - Testes aplicados à obtenção de uma lista de cinco em cinco anos.

Concluindo, vários testes foram realizados para garantir que todos os *endpoints* que estão a ser servidos pela plataforma gerem a informação enviada da forma correta e que não existe forma do cliente não receber qualquer tipo de resposta. Foram também realizados vários testes a métodos auxiliares para obtenção de informação e criação do sistema de recomendações, vitais para o funcionamento deste sistema.

7.8 Análise à performance

Um dos requisitos não-funcionais fundamental para este projeto é a *performance*. Como o sistema de recomendações tem de gerar recomendações em tempo útil para um utilizador, é necessário que os resultados sejam apresentados sem que o mesmo note lentidão. Neste caso, foi definida a meta de 800 milissegundos para uma lista de 10 recomendações para um utilizador.

Para medir a performance, e gerar gráficos que permitam analisar em que métodos e funções se gasta mais tempo (*profiling*), foi utilizado a ferramenta Silk. O Silk é uma biblioteca para *profiling* e inspeção na *framework* Django. Todos os pedidos são analisados, e é possível verificar todas as *queries* feitas à base de dados, assim como um gráfico de acumulação de tempo.

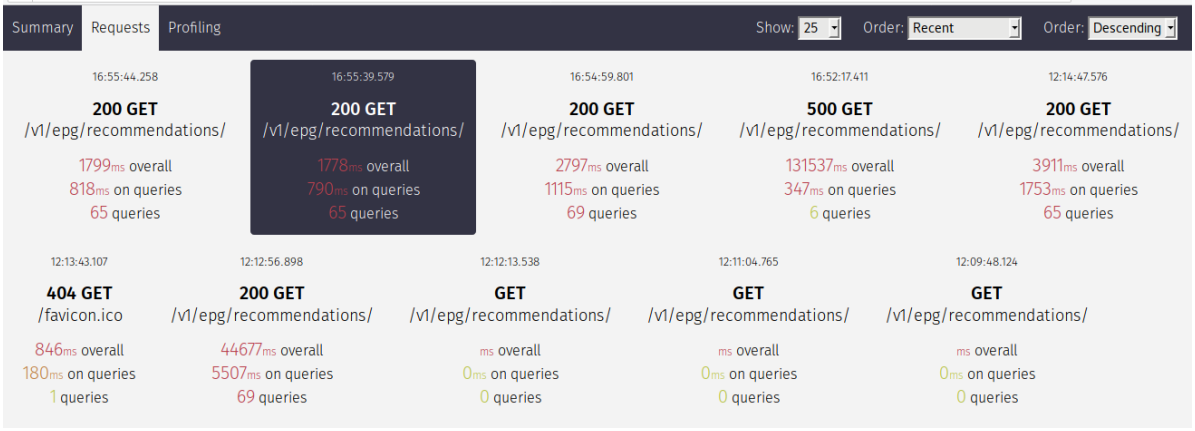

Ilustração 35 - Exemplo da página de pedidos do Silk

No início do desenvolvimento foi realizado um teste para 15 mil utilizadores com 98 mil programas, em que cada utilizador deu a opinião sobre 98. Num computador de trabalho o sistema conseguia gerar recomendações para o utilizador em aproximadamente um segundo. Este teste garantiu que a escolha do LightFM para gerar recomendações foi acertada, e que um eventual problema de *performance* não seria devido ao LightFM, uma vez que este aguenta com uma quantidade de dados substancial.

Consequentemente, e agora utilizando o processo de obter recomendações utilizando a base de dados do EPG e interações reais com o sistema, foi feito o *profilling* e medidos os tempos de forma a reduzir o número de queries feito à base de dados. Este *profilling* pode ser visualizado na Ilustração 36.

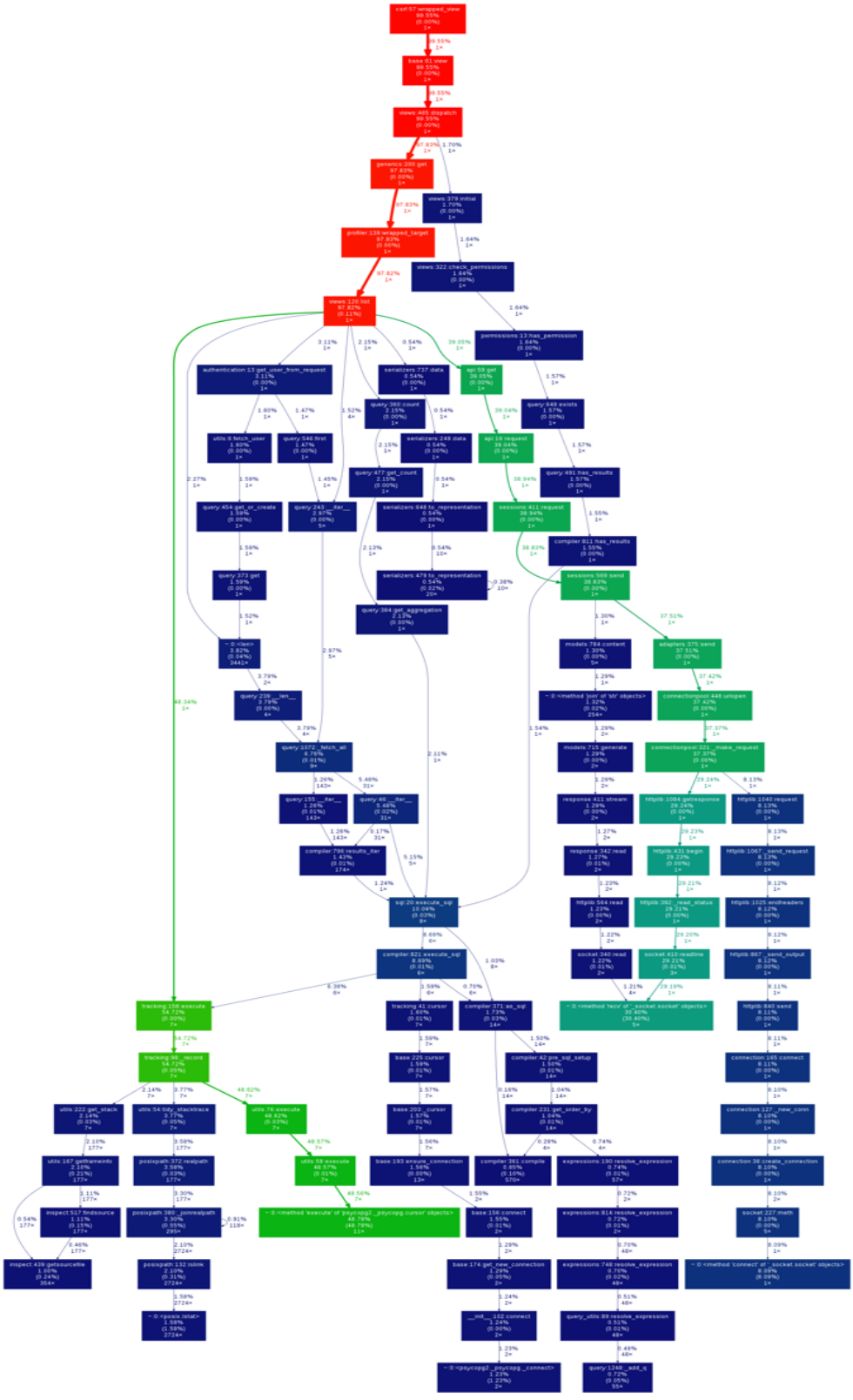

Ilustração 36 - Profilling do método que gera recomendações para um utilizador

Foram feitas várias otimizações ao longo das iterações de desenvolvimento. Um dos primeiros problemas encontrados foi obter todos os programas em cartaz numa única *query*, e que esta não fosse lenta. Um horário pode estar associado a um filme, programa genérico ou episódio. Como um episódio não se trata de um programa, mas sim a série e um TvShow, existe então a possibilidade de existir dois tipos de programas que estão indiretamente associados ao horário, que não se conseguem obter de forma imediata, a não ser através do "JOIN" de três tabelas ("Season", "Serie" e "TvShow"). Existe a possibilidade de executar essa *query* sempre que for necessária, ou então utilizar uma *cache* de forma a não estar sempre a chamar a base de dados e melhorar o tempo de resposta. A utilização de um mecanismo de *cache* como o Redis esteve em consideração, uma vez que se poderia armazenar todos os programas em cartaz e não seria necessário submeter carga à base de dados com o mesmo pedido. No entanto, traria mais uma dependência ao projeto. Foi então utilizada uma "Materialized View" (PostgreSQL, 2017), que armazena a resposta dessa mesma *query*, e é atualizada de 15 em 15 minutos, fazendo com que esta esteja relativamente atualizada e com que o sistema de recomendações não esteja constantemente a requisitar a base de dados e que responda em tempo útil. Esta *view* (Ilustração 37) efetua pedidos a uma outra, criada antecipadamente a este projeto, que é utilizada na exportação de informação de horários para os clientes, que por si só já é uma otimização na obtenção dessa informação.

```
-- Materialized View: public.mv_schedule

-- DROP MATERIALIZED VIEW public.mv_schedule;

CREATE MATERIALIZED VIEW public.mv_schedule AS
 SELECT DISTINCT ON (schedules_utc.movie_id, serie_tbl.id, tvshow_tbl.id, schedules_utc.generic_program_id, schedules_utc.day) COALESCE(schedules_utc.movie_id, serie_tb
    schedules_utc.day
   FROM schedules_utc
     LEFT JOIN episode ON episode.id = schedules_utc.episode_id
     LEFT JOIN season ON season.id = episode.season_id
     LEFT JOIN epguide_program serie_tbl ON serie_tbl.id = season.serie_id
     LEFT JOIN epguide_program tvshow_tbl ON tvshow_tbl.id = episode.tvshow_id
WITH DATA;

ALTER TABLE public.mv_schedule
  OWNER TO epg;

-- Index: public.mv_schedule_id_day

-- DROP INDEX public.mv_schedule_id_day;

CREATE INDEX mv_schedule_id_day
  ON public.mv_schedule
  USING btree
  (id, day);
```

Ilustração 37 - Materialized view que apresenta os programas em cartaz

Com a utilização de uma "Materialized View", e através da junção de tabelas feitas com o ORM do Django, tornou-se possível responder em tempo útil ao *endpoint* das recomendações de um utilizador. No entanto, o facto do projeto em produção estar numa máquina diferente da base de dados do EPG, faz com que a comunicação demore mais tempo e que este não atinja ainda a meta estabelecida de 800ms para a obtenção de recomendações de um utilizador. Existe ainda outras formas para aumentar a *performance* do sistema de recomendação, através do uso de técnicas de *clustering* para juntar a vizinhança de um certo utilizador.

8 Avaliação da solução

Neste capítulo é apresentado os métodos de avaliação utilizados para testar a solução desenvolvida, e a aplicação dos mesmos para estabelecer se as metas definidas foram atingidas.

8.1 Sistema de recomendação

No desenvolvimento desta plataforma um dos focos na avaliação da solução foi no sistema de recomendações e se este sistema gera de facto resultados positivos para os utilizadores. Tal como já foi documentado no capítulo "2.1.2", será considerada uma boa recomendação um programa que o utilizador em questão já viu e indicou ter apreciado. A razão pela qual é utilizada esta designação para definir uma boa recomendação é que o processo se torna bastante moroso se cada programa que foi recomendado for visto, pelo tempo envolvido. Por exemplo, alguns filmes possuem mais de duas horas de duração. Assim, e da mesma forma, uma má recomendação para o utilizador é um programa que o utilizador viu e não gostou.

É extremamente difícil desenhar uma experiência de avaliação com todos os seus aspetos totalmente controlados. A secção 8.1.1 descreve as possibilidades que foram ponderadas e na secção 8.1.2 as experiências realizadas para avaliar a funcionalidade.

8.1.1 Alternativas

Uma das possíveis forma de avaliação de solução seria a utilização da aplicação móvel desenvolvida e questionar os utilizadores, através do inquérito, sobre a sua satisfação em relação ao algoritmo de recomendação. No entanto, devido ao pouco tempo de desenvolvimento dado para este projeto, não foi possível chegar a uma fase em que o projeto estivesse maduro suficientemente para se proceder à recolha de opinião dos utilizadores sobre a qualidade das sugestões. No entanto, note-se que esta possibilidade, barata e relativamente

rápida, à semelhança das que se obtém com a utilização de plataformas como Survey Monkey (SurveyMonkey, s.d.), não garantiria a obtenção de uma amostra representativa, tendo em conta a grande quantidade de telespectadores nacionais.

Uma outra possível forma de avaliação seria a utilização de cálculos matemáticos para calcular a precisão do algoritmo de recomendação. Esta forma usa apenas dados de classificações dos utilizadores para efetuar cálculos matemáticos, que avaliam diversos fatores do algoritmo e que permitem quantificar, de certa forma, a satisfação dos utilizadores. Utilizando este método de avaliação, considera-se que os utilizadores estão a dar classificações honestas sobre os itens. Tendo em consideração todos os fatores, esta foi a opção adotada.

8.1.2 Experiência de avaliação

Foi utilizado um *dataset* do MovieLens que já possui uma quantidade significativa de recomendações por utilizadores de todo o mundo. O MovieLens é uma plataforma focada em recomendações desenvolvida pelo GroupLens, grupo de investigação da faculdade de Minnesota (GroupLens, s.d.). Neste momento existem vários *datasets* disponíveis no *website* da empresa (Harper & Konstan, 2016), onde o escolhido foi um com cerca de 100 mil de classificações, aplicadas a 1700 filmes por 1000 utilizadores. Cada utilizador tem pelo menos 20 classificações. Todos os *datasets* possuem uma estrutura semelhante, separada por vários ficheiros:

- "tags.csv", que inclui todas as palavras que os utilizadores usaram para associar a filmes. Para esta avaliação não foi considerado este ficheiro;

- "links.csv", que inclui o identificador do filme do MovieLens, e dois identificadores externos do IMDb e do TheMovieDatabase. Através destes identificadores externos, torna-se fácil a associação do conteúdo da InfoPortugal com este *dataset*;

- "movies.csv", que incluí a informação essencial de todos os filmes (nome e ano de lançamento), assim como as categorias do filme;

- "ratings.csv", que possui todas as classificações dadas pelos utilizadores. Esta informação está separada em quatro colunas: identificador do utilizador, identificador do filme, classificação (de zero a cinco) e o *timestamp* da classificação.

Através destes dados e adaptando essa informação para ser consumida pelo algoritmo de recomendação, foi possível testar a precisão do algoritmo. Foi escolhido um grupo de utilizadores, e um conjunto de filmes classificados por cada um desses utilizadores (em que esse conjunto de filmes não pode ser a totalidade de classificações dadas pelo utilizador). Através disso verificou-se as recomendações geradas pelo sistema e se estas coincidem com outras avaliações positivas dadas pelo mesmo utilizador. Com este método de avaliação aplicou-se o mesmo conceito referido neste documento, uma vez que se o utilizador deu uma classificação bastante positiva significa que o utilizador gostou do filme e que se trata de facto de uma boa

recomendação. Neste método é considerada uma má recomendação caso o utilizador tenha classificado negativamente o filme (inferior ou igual a 2,5). Todas as restantes recomendações não poderão ser classificadas como erradas ou corretas, uma vez que dependeria de o utilizador ver ou não o filme para classificar a recomendação.

Tendo em consideração todas as ações efetuadas para avaliar o algoritmo de recomendação, a metodologia de avaliação aplicada foi de "Cross Validation", utilizando o método *holdout*. Este método divide um conjunto de dados que consequentemente é utilizado para avaliar o algoritmo de recomendação numa certa grandeza (neste caso a precisão). Optou-se por este método porque é o mais simples do "Cross Validation" e avalia o algoritmo de recomendação de uma forma lógica. No entanto, foi necessário estar atento aos grupos escolhidos para aprendizagem/teste, uma vez que podem existir resultados amplos dependendo da amostra escolhida.

Considera-se então que, para testar a precisão algoritmo de recomendação, é utilizada a seguinte fórmula:

$$precisão = \frac{nº\ de\ boas\ recomendações}{nº\ total\ de\ recomendações}$$

Para testar a eficácia do algoritmo da plataforma foi necessário definir uma meta que o sistema deve atingir numa primeira iteração. Utilizando a mesma fórmula em dois produtos semelhantes (Netflix e IMDb), foi possível verificar qual a precisão de ambos para definir uma meta para este projeto. No caso do IMDb esta métrica foi estimada no início do trabalho e está descrita no capítulo de "Recomendações em soluções existentes", onde um utilizador avaliou a precisão do sistema. Esta avaliação deveria de usar um maior número de utilizadores, no entanto, não foi encontrado nenhum estudo ao sistema de recomendação do IMDb e não foi possível obter a disponibilidade de outros utilizadores para testar eficazmente a precisão desta aplicação.

Para o Netflix foi utilizado o estudo realizado por Saaket Unadkat e Shruti Shetty. Neste estudo foram realizados vários testes à plataforma escolhendo quatro utilizadores, que consequentemente à realização das tarefas atribuídas, tiveram de responder a um inquérito. Neste estudo a precisão teve um valor de 0.525 (Unadkat & Shetty, 2013).

Uma vez que o resultado final de precisão do IMDb foi 0.17, em conjunto com os dados do Netflix (0.525), foi definida a meta de precisão para a primeira iteração deste sistema de recomendação de 0.15, equivalente a 15%.

Uma das razões para a escolha do LightFM como biblioteca para o algoritmo de recomendação recaiu sobre o facto de este já possuir métodos de avaliação, para calcular e a avaliar as sugestões de um utilizador.

Como é possível consultar na documentação *online*, existe um método que apresenta qual a precisão do algoritmo escolhido, para o modelo instanciado. Esse método é o "precision_at_k",

que através das matrizes e utilizando todas as interações de um utilizador e comparando-as com o *k*, que representa o número de recomendações sugeridas pelo sistema para esse mesmo utilizador, revela a fração de resultados positivos (sendo um resultado positivo uma boa recomendação). Na terminologia utilizada no documento, o *k* corresponde à amostra de teste no método *holdout*. O resultado é uma lista que apresenta a precisão de cada um dos utilizadores. Caso o utilizador não tenha qualquer interação é retornado 0 (Kula, 2015).

Na própria documentação da ferramenta é possível ver a precisão do algoritmo para o *dataset* do MovieLens. O *dataset* escolhido contém cerca de 100 mil classificações, de 1000 utilizadores em 1700 programas. São considerados todos os utilizadores e filmes, uma vez que quanto maior a quantidade de dados, mais preciso seria o sistema de recomendação, teoricamente. Utilizando as primeiras cinco sugestões para cada utilizador, e tendo em conta todas as interações dos utilizadores, obteve-se uma precisão de 42% (Kula, 2015).

Aplicando o mesmo método aos dados obtidos ao longo do desenvolvimento deste projeto, a precisão obtida para o sistema de recomendação é de 34.6%. Este valor foi obtido tendo em consideração a base de dados do dia quatro de outubro de 2017, que possui cerca de 4571 interações em 48 utilizadores e 119734 programas disponíveis.

Este valor comprova que o algoritmo de recomendações está neste momento num estado superior àquele que era espectável no início do projeto, e que cumpre as metas estabelecidas. No entanto, é necessário ter em consideração que o valor de interações é muito reduzido comparando a outro tipo de sistemas, como o IMDb e Netflix, logo não pode ser efetuada uma comparação direta com estes, uma vez que os valores obtidos representam uma amostra muito pequena, comparando à escala das outras plataformas. Deve ser mantida uma fase de avaliação dos resultados no final de cada iteração do projeto, para garantir que o mesmo vai evoluindo e ficando mais preciso.

8.2 Pesquisa

Sendo a pesquisa outra das principais funcionalidades do sistema, esta deve ter também a sua forma de avaliação. Uma vez que para o módulo de pesquisa foi utilizada a ferramenta ElasticSearch, uma tecnologia já comprovada no mercado de desenvolvimento com cerca de seis milhões de downloads no início de 2014 e utilizado por empresas reconhecidas como Netflix, Facebook, Wikipedia, Atlassian e Github (Harris, 2014), pode-se comparar a componente de pesquisa a produtos mais maduros que já se encontram no mercado, como por exemplo o IMDb.

Para comparar as pesquisas dos dois produtos é necessário definir uma comparação comum entre estes dois sistemas. É então considerada uma pesquisa precisa se o primeiro resultado corresponder à expressão solicitada. Como população vai ser utilizada a programação relativa a um dia de um canal que possua bastante programas em comum, tanto para o sistema da InfoPortugal, como para o IMDb. Tendo isso em consideração, foi escolhida a programação do

canal FOX, uma vez que este transmite conteúdo internacional havendo assim os programas em ambas as bases de dados.

Existem então duas hipóteses para testar a componente de pesquisa:

- H_0 - Os resultados de pesquisa da plataforma desenvolvida são igualmente precisos aos resultados do IMDb;

- H_1 - Os resultados de pesquisa da plataforma desenvolvida são mais precisos que os resultados do IMDb.

Como termos de pesquisa, utilizados em ambas as plataformas, foram utilizados títulos (em português) de programas, títulos originais de programas e nomes de pessoas do elenco/equipa. Como resultados para títulos de programas foram esperados apenas programas. Para nomes de pessoas foram esperadas pessoas, e programas onde essas pessoas fazem parte do elenco/equipa. Foram também utilizados termos de pesquisa que não sejam exatamente iguais aos armazenados em base de dados. Um exemplo seria pesquisar por "Brad", e verificar se nos primeiros resultados estaria "Brad Pitt". Através dessa pesquisa é avaliada a popularidade dos itens, uma vez que é feito o teste para saber se o motor de pesquisa é dinâmico o suficiente para seguir tendências.

Não foi realizada nenhuma avaliação ao IMDb no capítulo de Recomendações em soluções existentes, uma vez que a amostra iria ser diferente à que foi realizada no final deste projeto.

Como teste estatístico foi utilizado o "paired t-test", uma vez que existe uma distribuição normal, em que os valores são emparelhados (é usado o mesmo grupo de valores para ambos os casos, ou seja, o mesmo termo de pesquisa) e existe apenas dois grupos para comparar.

Foi selecionado a programação do dia dois de outubro de 2017 do canal FOX. Foram pesquisadas na totalidade cerca de 62 expressões em ambos os motores de pesquisa e documentados os resultados de ambas. O número total de programas é de oito, e o número total de pessoas utilizadas é de 13. Na escolha das pessoas, foi escolhido o elenco de uma série e de um filme, uma vez que muitas vezes a popularidade entre as séries e filmes varia, e a popularidade é tida em consideração na pesquisa. De forma a garantir que os resultados do IMDb não são influenciados pelo utilizador a efetuar os testes, estes foram realizados em navegação anónima.

Tabela 8 - Total de pesquisas em que o primeiro resultado corresponde ao esperado

	API EPG	IMDB
Título de programa em português	8	7
Título original de programa	8	8
Nome de pessoa	12	13
Título incompleto de programa em português	3	1
Título original incompleto de programa	3	0
Nome incompleto de pessoa	6	2

A Tabela 8 representa o total de resultados obtidos para expressões pesquisadas, em que o primeiro resultado é o esperado. Aplicando agora o teste estatístico para a amostra retirada vão ser efetuados os seguintes cálculos: calcular a diferença entre cada um dos pares, de forma distinguir as diferenças positivas e negativas; calcular a diferença média; calcular o desvio padrão, e consequentemente erro padrão; calcular "t-statistic", dado pela divisão da diferença média pelo desvio padrão; finalmente, resta calcular o *p*. (Shier, 2004).

T Test: Two Paired Samples

SUMMARY			Alpha	0,05		Hyp Mean D	0	
Groups	*Count*	*Mean*	*Std Dev*	*Std Err*	*t*	*df*	*Cohen d*	*Effect r*
Before	6	6,666667	3,444803					
After	6	5,166667	5,036533					
Difference	6	1,5	1,352247	0,552052	2,717133	5	1,109265	0,772149

T TEST					
	p-value	*t-crit*	*lower*	*upper*	*sig*
One Tail	0,020959	2,015048			yes
Two Tail	0,041918	2,570582	0,080904	2,919096	yes

Ilustração 38 - Resultados aplicando o teste estatístico "paired t-test"

Numa primeira análise à tabela e aos cálculos, o projeto desenvolvido aparenta ter resultados semelhantes em comparação ao IMDb. No entanto, o IMDb possui uma base de dados muito maior e mais detalhada que a do projeto, o que faz com que apareçam resultados que não se encontram ainda no EPG, simplesmente porque não são emitidos na televisão portuguesa. O

IMDb também possui programas que ainda não foram transmitidos em televisão ou estrearam em cinema, como é o caso do “Hellboy”, utilizado neste teste. O “Hellboy” aparece como primeiro resultado na aplicação desenvolvida, e no IMDb aparece na segunda posição, uma vez que existe um novo “Hellboy” que ainda não foi lançado.

Para as expressões incompletas, o IMDb retornava sempre o resultado, mesmo que este não tivesse na primeira posição, o que é totalmente espectável. No caso do projeto desenvolvido existem expressões que não retornaram qualquer tipo de resultado (por exemplo, “Infil” do filme “Infiltrado”).

Concluindo, embora os resultados obtidos para os primeiros resultados do projeto sejam semelhantes em comparação ao IMDb, deve-se ter em consideração a escala do IMDb, o que faz com que este muitas vezes não seja tão preciso como o atual, simplesmente pela maior quantidade de informação. A aplicação de um teste estatístico nesta circunstância pode tender para o sistema de pesquisa desenvolvido, mas através de uma análise independente ao teste estatístico é possível verificar que não é verdade. No anexo 7 do documento é possível verificar quais os termos pesquisados, nas respetivas tabelas. O número corresponde à posição em que a expressão pesquisada foi encontrado na plataforma. Caso esta esteja na primeira posição é assinalada com “v”.

8.3 Limitações

Uma das limitações já mencionadas nesta secção foi o tempo disponibilizado para o desenvolvimento. Não foi possível obter uma versão madura do algoritmo em tempo útil, e que desse para criar um inquérito para questionar aos utilizadores o seu nível de satisfação com as recomendações dadas. Poderiam ter sido feitas também outro tipo de avaliações, como avaliar a precisão do algoritmo de recomendação tendo em conta apenas o CF, em deterimento ao algoritmo híbrido que foi utilizado como solução. No entanto, esta seria apenas uma comparação entre diferentes versões de um algoritmo com os mesmos dados, em que um destes não seria a solução desejada para o problema encontrado.

Cada vez mais os algoritmos de recomendação e motores de pesquisa têm em conta a cultura do país na apresentação dos resultados, assim como o contexto em que o utilizador se insere. Essa informação da cultura demográfica pode até ajudar o sistema a gerar melhores resultados para utilizadores que tenham dado pouco *feedback* à plataforma (Ferwarda, Vall, Schedl, & Tkalcic, 2011).

Neste projeto não é contemplada a diversidade cultural. Os atributos de um utilizador como a sua localização, idade, entre outros não estão a ser utilizados para o cálculo do sistema de recomendações. Um sistema como o Netflix tem acesso a esse tipo de informação, e poderá utilizá-la para sugerir recomendações, tendo em conta o contexto em que se insere o utilizador. O facto de não se contextualizar culturalmente pode afetar a precisão das recomendações e também os resultados de pesquisa.

9 Conclusões

Neste capítulo é apresentado todo o trabalho realizado e quais os objetivos que foram cumpridos, assim como quais os contributos deste projeto e algum trabalho futuro a desenvolver.

9.1 Objetivos alcançados

No início deste projeto foram destacados vários objetivos: o desenvolvimento de um sistema de recomendação, tendo como dados a programação das emissões em Portugal; um motor de pesquisa onde fosse possível pesquisar sobre este mesmo conteúdo; a criação de uma plataforma que possa ser acedida por outros sistemas, para usufruir destas duas funcionalidades. Todos estes objetivos foram cumpridos havendo, no entanto, algumas melhorias a efetuar uma vez que esta é a primeira versão desta plataforma.

Numa primeira fase foi efetuada uma investigação de forma a aferir qual o melhor sistema de recomendação a aplicar tendo em conta o contexto em que se encontrava o projeto. Foram assim estudados vários algoritmos de recomendação, assim como a área das recomendações, bibliotecas já existentes que oferecem essa funcionalidade, e avaliadas outras soluções já existentes. Foram ainda frequentados alguns cursos *online* sobre *data-mining* e sistemas de recomendação, para perceber melhor em que contexto são utilizados os sistemas de recomendação, e a área abrangente a este assunto. Através desse estudo, foi possível escolher as tecnologias utilizadas para a solução final, assim como desenhar a solução e a arquitetura do sistema.

Foi realizada uma análise de valor, onde foi estudado todo o mercado e possíveis clientes para esta plataforma. Tendo em consideração que a InfoPortugal já estabelecia uma relação com alguns destes, sempre se considerou que este projeto iria fortalecer mais estas relações e iria

facilitar a apresentação deste produto com as operadoras. Durante o desenvolvimento deste projeto, ficou provado que esta análise foi de facto precisa, uma vez que uma das operadoras ficou interessada no produto, e que se encontra neste momento em fase de avaliação do produto.

Na fase do desenho da solução foram avaliadas várias alternativas à solução escolhida, justificando o porquê dessas escolhas. Foi desenvolvido um sistema com uma arquitetura de microsserviços, que é facilmente expansível e testável, de forma a que este projeto possa abranger no futuro outro tipo de conteúdos e funcionalidades. A utilização de uma tecnologia já existente para a funcionalidade de pesquisa, através da sua fácil utilização e instalação, assim como a precisão dos seus resultados, revelou-se uma escolha acertada. A escolha de uma biblioteca para o sistema de recomendações que colmata o facto de não existir ainda uma grande quantidade de interações, faz com que as recomendações sejam precisas o suficiente para garantir a satisfação dos utilizadores.

Na fase final do projeto foram disponibilizadas aplicações móveis à Vodafone, com o intuito de vender a plataforma desenvolvida neste projeto e receber *feedback* da mesma para evoluir as suas características e expandir para novas funcionalidades.

Concluindo, através deste projeto a InfoPortugal vai conseguir rentabilizar melhor o conteúdo de programação de televisão nacional, oferecendo uma plataforma que dispõe de recomendações com um grau de precisão satisfatório e um sistema de pesquisa preciso, que é de grande utilidade para os telespectadores nacionais, e de grande interesse para as operadoras fornecerem estas funcionalidades aos seus clientes.

9.2 Limitações e trabalho futuro

O desenvolvimento e avaliação de um sistema de recomendações é um processo de longa duração. Estes tipos de algoritmos demoram não só muito tempo a implementar, mas principalmente na avaliação da precisão das recomendações do mesmo. Outros tipos de requisitos, como a *performance*, devem também ser tidas em consideração. Existe então melhorias a fazer na *performance* do sistema, e garantir que este responde cada vez mais rápido a obter este tipo de dados. Existe ainda o aumento da precisão das recomendações, de forma a aumentar a satisfação e a confiança dos clientes.

Tal como foi referido neste documento, existe ainda a possibilidade de contextualizar estas recomendações. Esta abordagem não foi utilizada nesta primeira fase de desenvolvimento, mas deve ser considerada no futuro. A opção de um utilizador poder filtrar os resultados das suas recomendações, podendo escolher quais os canais que possui ou as horas disponíveis que tem para poder ver televisão podem também ser consideradas.

Uma outra forma de aumentar a precisão e tornar o algoritmo mais diversificado é ter em conta os atributos de um utilizador, como a sua idade, localização, género, entre outros. Através

destes atributos seria possível contextualizar um utilizador numa certa cultura e sugerir programas baseando-se na cultura desse mesmo utilizador.

Outra funcionalidade a melhorar é a utilização de mais tipos de interação para o sistema, onde seja possível a avaliação através de uma classificação numérica ou para ter em consideração *feedback* implícito. A utilização de um mecanismo simples de gosto/não gosto, facilita a interação do utilizador com a interface. No entanto, ficou comprovado pelo *feedback* recebido pelos utilizadores da aplicação móvel que não é suficiente para dar a sua opinião acerca de um programa. Muitos utilizadores gostariam de ter a opção de destacar mais um programa, em detrimento de outros. Existe ainda a classificação de um programa nem ser bom ou mau para um utilizador, que não pode ser quantificada no sistema de classificação atual.

Existe ainda a possibilidade de ser enviado outro tipo de conteúdo mais enriquecedor de programas, como a disponibilização de *trailers* no detalhe do programa.

Na componente de pesquisa devem ser melhoradas as pesquisas que não utilizam termos completos. Essa situação foi verificada quando foi efetuada a avaliação/experimentação com diferentes termos, em que nenhum resultado é retornado.

No entanto, e tendo em consideração que esta se trata da primeira fase deste projeto, o objetivo principal foi cumprido, mas existem fatores que ainda podem ser melhorados.

Referências

Adomavicius, G., & Tuzhilin, A. (Fevereiro de 2014). Context-aware recommender systems. *User Modeling and User-Adapted Interaction, 24*.

Alexa. (s.d.). *imdb.com Traffic Statistics*. Obtido em 30 de 12 de 2016, de Alexa: http://www.alexa.com/siteinfo/imdb.com

Amatriain, X. (6 de Abril de 2012). *Netflix Recommendations: Beyond the 5 stars (Part 1)*. Obtido em 25 de Fevereiro de 2017, de The Netflix Tech Blog: http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html

Amorim, R. (2017). *TVadvisor - Recomendações de Televisão - Relatório de projeto/estágio.* Instituto Superior de Engenharia do Porto, Engenharia Informática.

Chalmers Innovationskontor. (27 de Outubro de 2014). *Inventory of Intellectual Assets.* Obtido em 30 de Setembro de 2017, de Chalmers Innovationskontor: https://innovationskontor.chalmers.se/sites/innovationskontoret.cms.chalmers.se/files/IKV-IAI-Guidelines-English-1.4.pdf

Chen, J., Wang, C., Wang, J., & Yu, P. S. (21 de Julho de 2016). Recommendation for Repeat Consumption from User Implicit Feedback. *IEEE Transactions on Knowledge and Data Engineering, 28*, 14.

Comissão Nacional de Proteção de Dados. (janeiro de 2017). *10 medidas para preparar a aplicação do regulamento europeu de proteção de dados.* Obtido em 21 de agosto de 2017, de https://www.cnpd.pt/bin/rgpd/10_Medidas_para_preparar_RGPD_CNPD.pdf

Cotter, P., & Smyth, B. (2000). Personalisation Technologies for the Digital TV World. *14th European Conference on Artificial Intelligence*.

Django. (s.d.). *The Web framework for perfectionists with deadlines*. Obtido em 3 de Fevereiro de 2017, de Django: https://www.djangoproject.com/

Elastic. (s.d.). *Getting Started | Elastic Search: The Definitive Guide*. Obtido em 3 de Fevereiro de 2017, de Elastic: https://www.elastic.co/guide/en/elasticsearch/guide/current/getting-started.html

Elastic. (s.d.). *Indexing Employee Documents | Elasticsearch: The Definitive Guide*. Obtido em 6 de Fevereiro de 2017, de Elastic: https://www.elastic.co/guide/en/elasticsearch/guide/current/_indexing_employee_documents.html

Elastic. (s.d.). *You Know, For Search... | ElasticSearch: The Definitive Guide*. Obtido em 3 de Fevereiro de 2017, de Elastic: https://www.elastic.co/guide/en/elasticsearch/guide/current/intro.html

Elkstein, M. (Fevereiro de 2008). *Learn REST: A Tutorial: 1. What is REST?* Obtido em 3 de Fevereiro de 2017, de Learn REST: A Tutorial: http://rest.elkstein.org/2008/02/what-is-rest.html

Ferwarda, B., Vall, A., Schedl, M., & Tkalcic, M. (Julho de 2011). Exploring Music Diversity Needs Across Countries. *ACM Transactions on Information Systems, 29*.

GroupLens. (s.d.). *MovieLens*. Obtido em 18 de Fevereiro de 2017, de MovieLens: https://movielens.org/

Harper, F. M., & Konstan, J. A. (7 de Janeiro de 2016). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS) - Regular Articles and Special issue on New Directions in Eye Gaze for Interactive Intelligent Systems (Part 1 of 2), 5*.

Harris, D. (28 de Janeiro de 2014). *6 million downloads later, Elasticsearch launches a commercial product*. Obtido em 3 de Março de 2017, de Gigaom: https://gigaom.com/2014/01/28/6-million-downloads-later-elasticsearch-launches-a-commercial-product/

Homan, J. (15 de Dezembro de 2015). *Relational vs. non-relational databases: Which one is right for you?* Obtido em 27 de Janeiro de 2017, de Pluralsight: https://www.pluralsight.com/blog/software-development/relational-non-relational-databases

Hug, N. (s.d.). *Surprise - A Python scikit for recommender system*. Obtido em 20 de agosto de 2017, de Surprise - A Python scikit for recommender system: http://surpriselib.com/

International Organization for Standardization. (s.d.). *Glossary for ISO 3166 - Codes for countries and their subdivisions*. Obtido em 30 de Setembro de 2017, de International Organization for Standardization: https://www.iso.org/glossary-for-iso-3166.html

Kaplan-Moss, J. (15 de Abril de 2009). *Developing Django Apps with ZC.Buildout*. Obtido em 10 de outubro de 2017, de Jacobian: https://jacobian.org/writing/django-apps-with-buildout/#s-create-a-test-wrapper

Kelly, M. (18 de setembro de 2013). *HAL - Hypertext Application Language*. Obtido em 5 de outubro de 2017, de The Hypertext Application Language: http://stateless.co/hal_specification.html

Koren, Y., Bell, R., & Volinsky, C. (Agosto de 2009). *Matrix Factorization Techniques for Recommender Systems.* Obtido em Novembro de 2016, de DataJobs: https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

Kula, M. (2015). Metadata Embeddings for User and Item Cold-start Recommendations. *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th, 1448*, 14-21.

Leskovec, J., Rajaraman, A., & Ullman, J. (2014). Mining of Massive Datasets. Em J. Leskovec, A. Rajaraman, & J. Ullman, *Mining of Massive Datasets* (Segunda Edição ed., pp. 307-341). Cambridge University Press.

McAlone, N. (14 de Junho de 2016). *Why Netflix thinks its personalized recommendation engine is worth $1 billion per year*. Obtido em 16 de Dezembro de 2016, de Business Insider: http://www.businessinsider.com/netflix-recommendation-engine-worth-1-billion-per-year-2016-6

Mendes, P. B. (8 de janeiro de 2017). *Na idade da pós-televisão.* Obtido em 17 de fevereiro de 2017, de Expresso: http://expresso.sapo.pt/sociedade/2017-01-08-Na-idade-da-pos-televisao

Muriçoca Labs. (s.d.). *Recommender Systems Framework in Python*. Obtido em 20 de agosto de 2017, de Crab - A Recommender Framework in Python: http://muricoca.github.io/crab/index.html

Nicola, S. (5 de Dezembro de 2016). *Análise de Valor*. Obtido de Moodle ISEP: https://moodle.isep.ipp.pt/pluginfile.php/142680/mod_resource/content/1/An%C3%A1lise_Valor_Aula1.pdf

Nielsen, J. (1993). Response Times: The 3 Important Limits. Em J. Nielsen, *Usability Engineering.*

PostgreSQL. (31 de Agosto de 2017). *CREATE MATERIALIZED VIEW*. Obtido em 3 de Setembro de 2017, de PostgreSQL: https://www.postgresql.org/docs/9.5/static/sql-creatematerializedview.html

Premraj. (8 de Dezembro de 2015). *web services - SOAP vs REST (differences)*. Obtido em 6 de Fevereiro de 2017, de StackOverflow: http://stackoverflow.com/questions/19884295/soap-vs-rest-differences

Python. (s.d.). *25.3. unittest — Unit testing framework*. Obtido em 10 de outubro de 2017, de Python: https://docs.python.org/2/library/unittest.html

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (s.d.). *Recommender Systems Handbook.* Springer Science+Business Media.

Richards, M. (2015). *Software Architecture Patterns.* Sebastopol, California, Estados Unidos da América: Heather Scherer.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). *Item-Based Collaborative Filtering Recommendation Algorithms.* University of Minnesota, Department of Computer Science and Engineering, Minneapolis.

Shier, R. (2004). Statistics: Paired t-tests. Em R. Shier, *Mathematics Learning Support Centre.*

SimilarWeb. (s.d.). *imdb.com Analytics - Market Share Stats & Traffic Rating*. Obtido em 30 de 12 de 2016, de SimilarWeb: https://www.similarweb.com/website/imdb.com

Software Testing Fundamentals. (s.d.). *Functional Testing*. Obtido em 10 de Outubro de 2017, de Software Testing Fundamentals: http://softwaretestingfundamentals.com/functional-testing/

Software Testing Fundamentals. (s.d.). *UNIT TESTING Fundamentals*. Obtido em 10 de outubro de 2017, de Software Testing Fundamentals: http://softwaretestingfundamentals.com/unit-testing/

Statista. (s.d.). *Netflix Subscribers count 2016*. Obtido em 27 de 1 de 2017, de Statista: The Statistics Portal: https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/

Strategyzer. (s.d.). *Business Model Canvas*. Obtido em 16 de Dezembro de 2016, de Strategyzer: https://strategyzer.com/canvas/business-model-canvas

SurveyMonkey. (s.d.). *SurveyMonkey*. Obtido de SurveyMonkey: https://pt.surveymonkey.com/?

Unadkat, S., & Shetty, S. (2013). *Netflix: A proposal to evaluate its recommendation system.* University of Michigan School of Information, Information Retrieval, Michigan.

Anexos

Anexo 1: Tabela de descrição do modelo de domínio do projeto

Tabela 9 - Descrição dos atributos do modelo de domínio do projeto

Atributo	Tabela	Descrição
id	EPGUser	Identificador único utilizado na construção da matriz de interação
plataform	EPGUser	Plataforma à qual o utilizador tem
interaction	UserInteraction	Representa a interação feita pelo o utilizador num determinado programa.
timestamp	UserInteraction	Data e hora na qual foi criada a interação
title	Program	Título em português do programa
original_title	Program	Título na língua original do programa
country	Program	País onde o programa foi realizado
year	Program	Ano em que o programa foi realizado
rating	Program	Classificação etária do programa
genre	Program	Categoria do programa
synopsis	Program	Descrição do programa
imdb_id	Program	Identificador único do IMDb para identificar o programa
imdb_score	Program	Classificação dos utilizadores do IMDb para o programa
language	Program	Língua utilizado no áudio
character	ProgramRole	Personagem interpretada no programa
name	Person	Nome da pessoa
birthday	Person	Data de aniversário da pessoa
deathday	Person	Data da morte da pessoa
imdb_id	Person	Identificador único do IMDb para identificar a pessoa
biography	Person	Biografia da pessoa
schedule	Schedule	Horário em que um programa vai ser emitido num certo canal
name	Channel	Nome do canal

Anexo 2: Tabela de descrição do modelo de dados de programas

Tabela 10 - Descrição do modelo de dados de programas

Atributo	Tabela	Descrição
id	Program	Identificador único incremental
title	Program	Título em português do programa
original_title	Program	Título na língua original do programa
country	Program	País onde o programa foi realizado

year	Program	Ano em que o programa foi realizado
rating	Program	Classificação etária do programa
ipcategoria	Program	Categoria do programa. Trata-se de uma chave estrangeira para uma tabela de categorias (IPGenre)
synopsis	Program	Descrição do programa
trailer	Program	URL para trailer de um programa
website	Program	Website oficial do programa
imdb_id	Program	Identificador único do IMDb para identificar o programa
imdb_score	Program	Classificação dos utilizadores do IMDb para o programa
language	Program	Língua utilizado no áudio
program	Broadcasted	Chave estrangeira que indica o programa
channel	Broadcasted	Chave estrangeira que indica o canal
schedule	Broadcasted	Horário do programa no canal
id	Channel	Identificador único incremental
name	Channel	Nome do canal
id	Season	Identificador único incremental
title	Season	Título da temporada
original_title	Season	Título na língua original da temporada
season	Season	Número da temporada
synopsis	Season	Descrição da temporada
trailer	Season	Trailer da temporada
serie	Season	Chave estrangeira para a série
year	Season	Ano da temporada
id	Episode	Identificador único incremental
episode_number	Episode	Número do episódio
title	Episode	Título do episódio
original_title	Episode	Título na língua original do episódio
synopsis	Episode	Descrição do episódio
season	Episode	Chave estrangeira para a temporada
tvshow	Episode	Chave estrangeira para o TvShow

Anexo 3: Tabela de descrição do modelo de dados de pessoas

Tabela 11 - Descrição do modelo de dados de pessoas

Atributo	Tabela	Descrição
role	Role	Indica que tipo de papel a pessoa interpreta no programa
content_type	Role	Indica o tipo de objeto ao qual a pessoa está associada (pode ser qualquer tipo de programa, temporada ou episódio)
object_id	Role	Identificador único do objeto ao qual a pessoa está associada

person	Role	Chave estrangeira para pessoa
id	Person	Identificador único incremental
fname	Person	Primeiro nome da pessoa
mname	Person	Nomes do meio da pessoa
lname	Person	Apelido da pessoa
birthday	Person	Data de aniversário da pessoa
deathday	Person	Data da morte da pessoa
imdb_id	Person	Identificador único do IMDb para identificar a pessoa
biography	Person	Biografia da pessoa

Anexo 4: Tabela de descrição do modelo de dados dos utilizadores da plataforma

Tabela 12 - Descrição do modelo de dados dos utilizadores da plataforma

Atributo	Tabela	Descrição
id	EPGRecommendationUser	Identificador único incremental
platform	EPGRecommendationUser	Chave estrangeira que indica a plataforma do utilizador
user_id	EPGRecommendationUser	Identificador único fornecido pela plataforma como forma de identificar um certo utilizador
name	Platform	Nome da plataforma
api_key	Platform	Chave utilizada nas comunicações com a aplicação para garantir a autenticidade da plataforma
active	Platform	Booleano que indica se a plataforma deve estar ativa
url	Referer	URL da plataforma, de forma a limitar os pedidos e reforçar a segurança
platform	Referer	Chave estrangeira de plataforma

Anexo 5: Tabela de descrição do modelo de dados das interações dos utilizadores

Tabela 13 - Descrição do modelo de dados das interações dos utilizadores

Atributo	Tabela	Descrição
id	UserInteraction	Identificador único incremental
program_type	UserInteraction	Indica o tipo de objeto ao qual a interação está associada (pode ser qualquer tipo de programa, temporada ou episódio)
object_id	UserInteraction	Identificador único do objeto ao qual a interação está associada

name	UserInteraction	Nome da plataforma
user_id	UserInteraction	Chave estrangeira que indica o EPGRecommendationUser da interação
value	UserInteraction	Valor a colocar na matriz das interações
action	UserInteraction	Ação que foi despoletada na interação (“Like”, “Dislike” ou “Unknown”

Anexo 6: Modelo de dados completo do EPG

Anexo 7: Tabelas de recolha para os testes no sistema de pesquisa

Tabela 14 - Pesquisa de títulos completos de programas em português

Termo	EPG	IMDb
Bad Ass	v	v
Hawai: Força Especial	v	v
Nunca Chove em Filadélfia	v	v
Investigação Criminal: Los Angeles	v	v
Puro Aço	v	v
Infiltrado	v	2º
Percy Jackson e os Ladrões do Olimpo	v	v
Hellboy	v	2º (novo filme com mesmo nome)
Marcado Para Matar	2	v

Tabela 15 - Pesquisa de títulos originais completos

Termo	EPG	IMDb
Bad Ass	v	v
Hawaii Five-0	v	v
It's Always Sunny in Philadelphia	v	v
NCIS: Los Angeles	v	v
Real Steel	v	v
Inside Man	v	v
Percy Jackson & the Olympians: The Lightning Thief	v	v
Hellboy	v	2º (novo filme com mesmo nome)
Marked for Death	2	v

Tabela 16 - Pesquisa de pessoas pelo nome completo

Nome da pessoa	Nome do filme	EPG	IMDb
Danny Trejo	Bad Ass	v	v
Charles S. Dutton	Bad Ass	v	v
Ron Perlman	Bad Ass	v	v
Patrick Fabian	Bad Ass	v	v
Joyful Drake	Bad Ass	v	v
Craig Moss	Bad Ass	v	v
Chris O' Donnell	NCIS: Los Angeles	3	v

Daniela Ruah	NCIS: Los Angeles	v	v
LL Cool J	NCIS: Los Angeles	v	v
Eric Christian Olsen	NCIS: Los Angeles	v	v
Linda Hunt	NCIS: Los Angeles	v	v
Barret Foa	NCIS: Los Angeles	v	v
Shane Brennan	NCIS: Los Angeles	v	v

Tabela 17 - Pesquisa de títulos de programas incompletos em português

Program	Termo	EPG	IMDb
Bad Ass	Bad	6	34
Hawai: Força Especial	Hawai	v	4
Nunca Chove em Filadélfia	Nunca Chove	v	v
Investigação Criminal: Los Angeles	Investigação Criminal	2	4
Puro Aço	Puro	v	4
Infiltrado	Infil	0	5
Percy Jackson e os Ladrões do Olimpo	Percy Jackson	2	2
Hellboy	Hell	0	8
Marcado Para Matar	Marcado	2	7

Tabela 18 - Pesquisa de títulos originais incompletos

Program	Termo	EPG	IMDb
Bad Ass	Bad	6	34
Hawaii Five-0	Hawaii	v	4
It's Always Sunny in Philadelphia	It's Always	v	2 (Filme com o mesmo titulo)
NCIS: Los Angeles	NCIS	3	4
Real Steel	Real	8	6
Inside Man	Inside	14	5
Percy Jackson & the Olympians: The Lightning Thief	Percy Jackson	2	2
Hellboy	Hell	0	8
Marked for Death	Marked	v	4

Tabela 19 - Pesquisa de nomes incompletos de pessoas

Nome da pessoa	Termo pesquisado	EPG	IMDb
Danny Trejo	Danny	2	5

Charles S. Dutton	Charles S.	v	v
Ron Perlman	Ron	v	5
Patrick Fabian	Fabian	v	10
Joyful Drake	Joyful	v	2
Craig Moss	Moss	6	50
Chris O' Donnell	Chris	39	46
Daniela Ruah	Daniela	9	4
LL Cool J	LL Cool	v	v
Eric Christian Olsen	Eric	v	7
Linda Hunt	Linda	4	12
Barrett Foa	Barrett	3	4
Shane Brennan	Brennan	7	30

Anexo 8: Diagrama de arquitetura