FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Letícia Ellen Bernardo

Cine Collection: Um aplicativo para Recomendação de Filmes

Americana, SP

2017

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Letícia Ellen Bernardo

Cine Collection: Um aplicativo para Recomendação de Filmes

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Dr. Kleber de Oliveira Andrade

Área de concentração: Inteligência Artificial e Engenharia de Software.

Americana, SP.

2017

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

B444c BERNARDO, Letícia Ellen

Cine Collection: um aplicativo para recomendação de filmes./ Letícia Ellen Bernardo. – Americana: 2017.

122f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Kleber de Oliveira Andrade

1. Inteligência artificial 2. Android – aplicativos I. ANDRADE, Kleber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 007.52
681.519

Letícia Ellen Bernardo

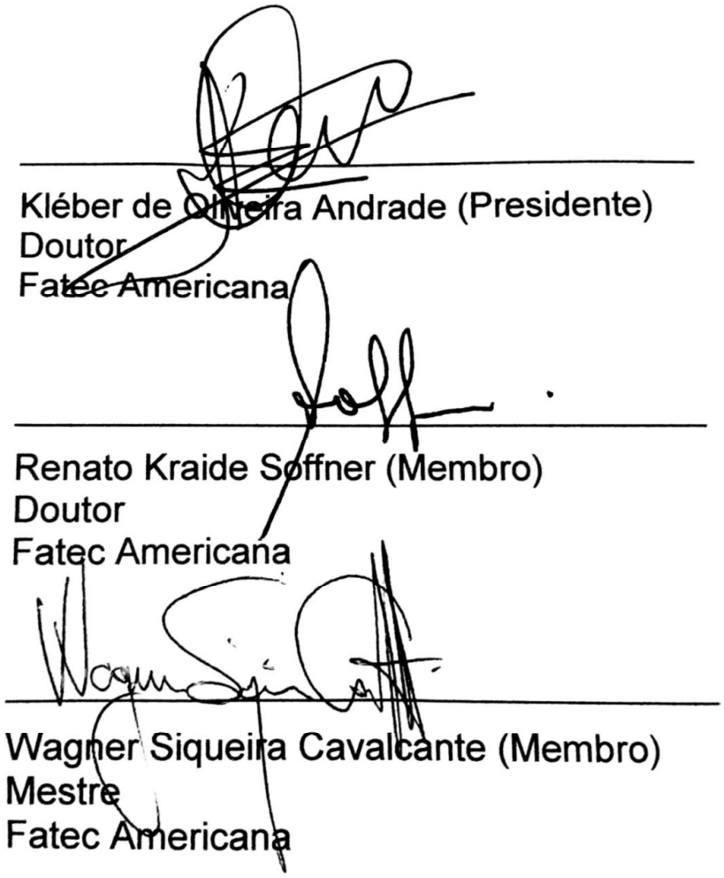
Cine Collection: Um aplicativo para Recomendação de Filmes

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área de concentração: Inteligência Artificial e Engenharia de Software.

Americana, 28 de junho de 2017.

Banca Examinadora:



Kléber de Oliveira Andrade (Presidente)
Doutor
Fatec Americana

Renato Kraide Soffner (Membro)
Doutor
Fatec Americana

Wagner Siqueira Cavalcante (Membro)
Mestre
Fatec Americana

AGRADECIMENTOS

A todas as pessoas que lutam para promover uma educação melhor na Faculdade de Tecnologia de Americana.

Ao meu orientador, Prof. Dr. Kleber de Oliveira Andrade por ter acreditado no meu projeto, aceitando ser meu orientador e por ter me apoiado quando mudei o foco do projeto. A sua participação foi fundamental para minha formação.

Ao Carlos Marcelo Tonisso Jr, por ser meu companheiro e maior incentivador. Por me apoiar com seu carinho e por me ajudar quando precisei. E seus pais, Helena e Marcelo, por cuidarem de mim como uma filha.

À minha avó, Maria Inês, por me acolher e por me incentivar nos estudos. Serei eternamente grata pelos seus ensinamentos e pelo amor.

A toda a minha família, que mesmo distante sempre foi presente em minha vida.

A todos que me ajudaram direta ou indiretamente com o projeto.

"[...] Então, minha queria Amelie, você não tem ossos de vidros. Pode suportar os baques da vida. Se deixar passar essa chance, então, com o tempo, seu coração ficará tão seco e quebradiço quanto meu esqueleto, então, vá em frente, pelo amor de Deus".

O Fabuloso Destino de Amelie Poulain

RESUMO

Com o crescimento do mercado de dispositivos móveis, atualmente, o celular se tornou o meio mais popular dos brasileiros se conectarem a internet. Hoje em dia, os celulares são utilizados para diversas funções, como socializar, buscar informações e suprir passatempos, dentre eles, o mais comum é assistir a vídeos e filmes. Por meio de uma análise de viabilidade foi detectado que muitas pessoas utilizam meios manuais para lembrar-se de filmes e para procurar por novos filmes. Por este fato, o trabalho apresentado documenta o desenvolvimento de um aplicativo de filmes, Cine Collection, que faz uso de um sistema de recomendação de filtragem colaborativa, com a finalidade de apresentar recomendações personalizadas aos usuários que buscam por novos filmes. O trabalho apresenta, primeiramente, um estudo sobre os sistemas de recomendação e, em seguida, o processo de desenvolvimento do aplicativo utilizando diversos recursos, como, a base de dados de filmes gratuita (TMDB), recursos do Firebase, como banco de dados e autenticação do Firebase com login do Facebook, entre outros, como também, apresenta os testes realizados.

Palavras Chave: Inteligência Artificial; Engenharia de Software; Dispositivos móveis – aplicativos.

ABSTRACT

With the growth of the mobile market, today, the smartphone has become the most popular way for Brazilians to connect to the Internet. Nowadays, smartphones are used for various functions, such as socializing, searching for information and supplying hobbies, among which the most common is watching videos and movies. Through a feasibility analysis it has been found that many people use manual means to remember movies and to search for new movies. For this reason, the presented work documents the development of the Cine Collection movie application that uses a collaborative filtering system, with the purpose of presenting personal recommendations to users looking for new movies. The work presents, first, a study on the recommendation systems and then the process of developing the application using several resources, such as The Movie Database (TMDB), Firebase features such as real-time database and Firebase authentication with Facebook login, among others. As well, it presents the tests performed.

Keywords: *Artificial Intelligence; Software Engineering; Mobile - applications.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

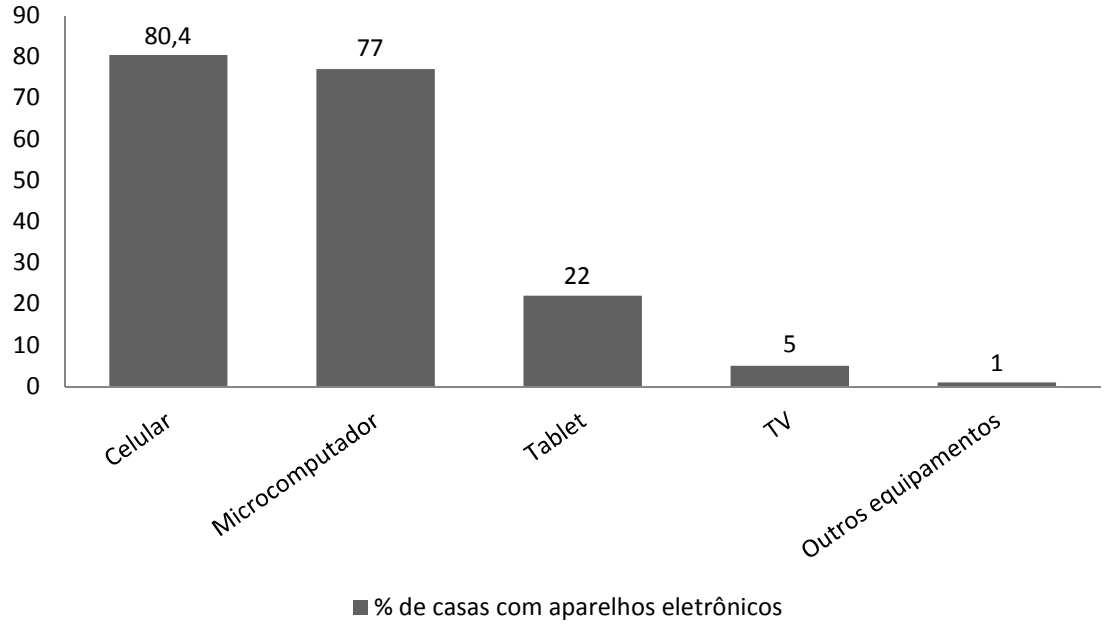
LISTA DE TABELAS

1 INTRODUÇÃO

A origem de novas tecnologias e o aumento da acessibilidade da internet contribuiu para o crescimento exponencial no volume de dados, e seu tratamento em tempo hábil tornou-se inexequível. Um exemplo desse crescimento pode ser encontrado na Pesquisa Nacional Por Amostra de Domicílios (Pnad) divulgada pelo Instituto Brasileiro de Geografia e Estatística (IBGE)[1], que apontou que, entre os anos 2013 e 2014, o índice de pessoas conectadas subiu de 48% para 54,9% do total, isto é, cerca de 36,8 milhões de casas com acesso à internet (PNAD, 2014).

O celular se tornou o meio mais popular dos brasileiros de se conectar a internet, ultrapassando até mesmo os computadores pessoais. Das 36,8 milhões de casas conectadas, 29,6 milhões dispõem de um telefone móvel para se conectar, representando 80,4% do total (PNAD, 2014). A Figura 1 apresenta um comparativo de acesso de dispositivos entre milhões de casas.

Figura 1 – Comparativo de acesso de dispositivos entre milhões de casas.



Fonte: Adaptado de PNAD (2014) [2]

[1] IBGE. Para saber mais acesse: <http://www.ibge.gov.br>

[2] PNAD 2014. Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/livros/liv94935.pdf>. Acesso em: 08/04/2016.

O crescimento do celular foi tão notável que o Android, sistema operacional mais utilizado em smartphones ultrapassou o Windows, sistema operacional mais utilizado em microcomputadores pessoais, em número de dispositivos conectados à internet (STATCOUNTER, 2017).

Os dispositivos móveis estão cada vez mais presentes na vida das pessoas, seja na própria casa, na rua ou até mesmo no trabalho. Suas funções variam entre socializar com outras pessoas, buscar notícias e informações, acessar sites, fazer compras, planejar atividades, entre outros. De acordo com uma pesquisa feita pelo Comitê Gestor da Internet no Brasil (CGI.BR)[3], dentre essas atividades relacionadas a entretenimento, as mais comuns são assistir a filmes e vídeos em sites com 58% do total de usuários de Internet. O número se torna mais expressivo nas faixas de usuários entre 10 a 15 anos (68% e 66%, respectivamente) e entre 16 a 24 anos (67% e 69%) (CGI.BR, 2014).

Uma pesquisa divulgada pela Conectaí apontou que, um em cada três internautas brasileiros (34%) assistem a filmes, pelo menos, uma vez por semana, mostrando que assistir a filmes é um dos passatempos mais comuns entre os brasileiros (CONECTAÍ, 2017).

A fim de suprir as necessidades deste passatempo, existem variados serviços na web de *streaming*[4] de filmes, sites de filmes, organizadores cinéfilos e sistemas de recomendação de filmes. Atualmente, os softwares mais populares que possuem recomendação de filmes são Netflix[5], Google Play Filmes[6], Amazon Prime Video[7] e Youtube[8].

Netflix é o serviço de "*streaming*" de vídeo mais popular do mundo que, mediante ao pagamento de uma mensalidade, oferece acesso ilimitado a filmes, séries de TV, shows e documentários premiados aos milhares de aparelhos

[3] CGI.BR. Para saber mais acesse: <http://www.cgi.br>
[4] *Streaming* é uma tecnologia de transmissão simultânea de dados de áudio e vídeo por meio da rede. Através do *streaming* não há necessidade de fazer o download do conteúdo, pois, na medida que, o software de *streaming* recebe os dados, eles são arquivados temporariamente na máquina e transmitidos ao usuário.
[5] Netflix. Disponível em: <https://www.netflix.com>. Acesso em: 22/04/2017.
[6] Google Play Filmes. Disponível em: <https://play.google.com/store/movies?hl=pt_BR>. Acesso em: 22/04/2017.
[7] Amazon Prime Video. Disponível em: <https://www.primevideo.com>. Acesso em: 22/04/2017.
[8] Youtube. Disponível em: <https://www.youtube.com>. Acesso em: 22/04/2017.

conectados à internet. O Netflix pode ser baixado em smartphones, smart TVs, tablets, entre outros dispositivos e possui plataforma web. Suas recomendações levam em consideração o histórico de filmes do usuário (NETFLIX, 2017).

Google Play Filmes é o serviço que permite assistir filmes comprados ou alugados no Google Play[9]. É possível fazer uma lista de desejos e receber notificações quando os filmes são liberados para compra ou locação. Os usuários podem realizar avaliações e escrever resenhas expressando sua opinião sobre os filmes (GOOGLE PLAY, 2017).

Amazon Prime Video é o serviço de "*streaming*" de vídeo desenvolvido pela Amazon.com[10]. Com a mesma proposta e sendo uma concorrente direta da Netflix, oferece serviço por US$5,99/mês. Os filmes também podem ser baixados no dispositivo e assistidos off-line. As recomendações de filmes são separadas em categorias (AMAZON PRIME VIDEO, 2017).

Youtube é um serviço que permite descobrir, assistir e compartilhar vídeos por usuários. Há também a possibilidade exibir vídeos ao vivo, editar vídeos, transmitir vídeos com direitos autorais feitos pelos canais de televisão e filmes (YOUTUBE, 2017).

Os serviços citados oferecem diversas funcionalidades e valores diferentes, fornecendo recomendações de filmes direta ou indiretamente (quando o usuário não se dá conta de estar recebendo uma recomendação) para seus usuários. No entanto, diversos usuários têm dificuldades para encontrar novos filmes que os agradam nesta abundância de dados.

Levando estes aspectos em consideração, foi elaborada a Tabela 1 mostrando as principais diferenças entre os serviços da Netflix, Google Play Filmes, Amazon Prime Video, Youtube e o Cine Collection, aplicativo desenvolvido neste trabalho:

[9]Google Play é a loja virtual do Google para celulares com o sistema Android. Disponível em: < https://play.google.com/store?hl=pt_BR>. Acesso em: 25/04/2017.
[10]Amazon.com é uma empresa transnacional de comércio eletrônico. Disponível em: <https://www.amazon.com>. Acesso em: 25/04/2017.

Tabela 1 – Comparativo de funcionalidades entre as plataformas de filmes mais populares e o aplicativo desenvolvido neste trabalho.

	Netflix	Google Play Filmes	Amazon Prime Video	Youtube	Cine Collection
Streaming de filmes	X	X	X	X	
Alugar ou comprar filmes		X		X	
Organizar filmes				X	X
“Favoritar” filmes	X		X		X
Exibir informações da equipe técnica e do elenco	X	X			X
Recomendar filmes	X			X	X

Fonte: Elaborado pelo autor

1.1 Objetivos

Este trabalho tem como objetivo geral apresentar o processo de desenvolvimento de um aplicativo utilizando a técnica de filtragem colaborativa como o motor de recomendação de filmes. Também pretende fundamentar que o conceito de sistemas de recomendação pode ser aplicável até mesmo no desenvolvimento de uma aplicação em um dispositivo móvel. Quanto aos objetivos específicos são:

- Fazer um estudo sobre os sistemas de recomendação, suas características e suas técnicas;
- Desenvolver e aplicar a técnica de recomendação baseada em filtragem colaborativa em um aplicativo de recomendação de filmes;
- Utilizar a metodologia cascata e expor os procedimentos realizados;
- Realizar experimentos e discutir os resultados;
- Apresentar as conclusões finais e exibir as possibilidades para trabalhos futuros.

1.2 Organização do Trabalho

O restante do trabalho está organizado em três capítulos conforme descrição a seguir:

- Capítulo 2: define os conceitos sobre sistemas de recomendação e disserta sobre as três principais técnicas. Como também informa exemplos de sistemas de recomendação presentes no mercado e em trabalhos acadêmicos.
- Capítulo 3: descreve o passo a passo sobre o desenvolvimento do aplicativo de recomendação de filmes utilizando a metodologia cascata.
- Capítulo 4: as considerações finais são apresentadas, juntamente com diversas possibilidades para trabalhos futuros.

Alguns apêndices foram criados para facilitar o conhecimento de alguns recursos utilizados, tais como:

- Apêndice A: apresenta um exemplo de cálculo para encontrar os vizinhos mais próximos utilizando a Correlação de Pearson;
- Apêndice B: detalha a pesquisa de viabilidade do projeto;
- Apêndice C: apresenta as telas do aplicativo;
- Apêndice D: apresenta um tutorial de como adicionar o botão de login do Facebook em um aplicativo Android;
- Apêndice E: apresenta um tutorial de como integrar o Firebase ao aplicativo;
- Apêndice F: apresenta um tutorial de como fazer o login do Facebook com autenticação no Firebase;
- Apêndice G: apresenta um tutorial de como conectar o aplicativo na API TMDB.

2 SISTEMAS DE RECOMENDAÇÃO

Os sistemas de recomendação (SR) são uma subárea de aprendizagem de máquina (AM), do inglês, *machine learning*[11], e que tem como objetivo fornecer sugestões para um usuário (IBM, 2017). Essas sugestões relacionam-se com vários processos de decisões humanas como: qual item comprar, qual notícia ler ou até mesmo qual filme assistir. Pode-se dizer que, o sistema de recomendação é um software que antecipa as necessidades do usuário antes mesmo que ele precise (RICCI *et al*., 2011).

Em geral, os sistemas de recomendação seguem uma das linhas mais básicas da comunicação e de escolha dos seres humanos, a escolha por indicação. Quando se quer experimentar algo novo, é comum pedir recomendações de amigos e seguir escolhas de pessoas que possuem certa influência. Os sistemas de recomendação têm como objetivo fornecer as recomendações a níveis computacionais baseando-se nas preferências que coleta do usuário.

Segundo Ricci *et al*. (2011), as recomendações focam normalmente em um tipo específico de itens que são considerados termos genéricos para designar o que o sistema recomenda aos utilizadores, podendo assumir aspectos bem distintos, tais como, livros, filmes, notícias, vídeos, anúncios, links patrocinados ou produtos de uma loja virtual. Desta maneira, os sistemas de recomendação têm a interface totalmente focada em gerar informação útil e eficaz para esse tipo específico de itens.

Os itens recomendados tentam atender as necessidades de um usuário, ou seja, um bom sistema de recomendação tenta se adequar ao utilizador. Para que isso ocorra, foram identificadas algumas das metas e tarefas que o sistema precisa cumprir para que um sistema de recomendação seja avaliado corretamente (HERLOCKER *et al.*, 2004, p. 9-11). São elas:

[11] *Machine Learning* é uma subcampo da ciência da computação que, explora a aprendizagem de máquina de forma automática, no qual, seu objetivo é criar um algoritmo capaz de aprender com seus próprios erros e fazer previsões sobre dados (ENCYCLOPEDIA BRITANNICA, 2017).

- **Encontrar bons itens**. Sugerir itens específicos para seus utilizadores, proporcionando-lhes uma lista ordenada dos itens recomendados, com as previsões de quanto o usuário gostaria de cada item.
- **Encontrar todos os bons itens**. Fazer uma listagem de cobertura de todos os bons itens, apresentando todos os resultados obtidos de recomendações que aquele usuário poderia ter contando com os resultados de probabilidade baixa.
- **Recomendar uma sequência**. Recomendar uma sequência agradável dos itens recomendados já classificados.
- **Função de apenas navegar**. A Amazon.com diz que seus usuários, mesmo quando não tem nenhuma compra iminente, acham agradável navegar em seu site como uma atividade de aprendizagem ou até mesmo de entretenimento, então a precisão desses algoritmos não precisa ser tão assertiva.
- **Ser um sistema recomendador credível**. Os usuários não confiam automaticamente em um sistema recomendador. Muitas vezes os usuários vão “brincar” por um tempo para ver se o sistema corresponde bem aos seus gostos.
- **Formas de melhorar o perfil**. Os usuários oferecem suas informações e classificações, pois acreditam que estão melhorando a qualidade das recomendações que irão receber.
- **Permitir que os usuários se expressem**. Muitos usuários querem uma forma de exprimir suas opiniões e de interagir com outros usuários.
- **Permitir que os usuários se ajudem**. Alguns usuários estão felizes em contribuir com suas classificações para ajudar a comunidade.
- **Influenciar os outros**. Alguns usuários podem influenciar os outros para verem ou comprar um item específico.

Apesar desta lista citada não ser abrangente, a mesma identifica as tarefas mais importantes que o sistema deve desempenhar. Cazella (2010) expõe funções que o sistema de recomendação deve ter para obter algumas metas e tarefas da lista citada anteriormente:

- Enquanto o usuário navega no sistema, toda sua interação deve ser monitorada. Com base nas informações coletadas, mantém-se um modelo de usuário: dados demográficos, itens visualizados, interesses, preferências, entre outros.
- A apresentação do documento pode ser modificada de modo a sugerir ao usuário os próximos passos como itens serem adicionados, modificados, removidos, reorganizados ou comentados.
- O sistema pode esconder ou enfatizar fragmentos de uma página, assegurando que seu conteúdo inclua a informação apropriada, em um nível adequado de dificuldade ou detalhe.

Essas características citadas são denominadas personalização em sistemas de recomendação.

2.1 Personalização em Sistemas de Recomendação

Os Sistemas de Recomendação Personalizados (SRP) têm a capacidade de aprender e identificar as necessidades individuais de cada usuário e de gerar recomendações adequadas para ele.

Este tipo de sistema armazena uma representação interna chamada de perfil do usuário. Tradicionalmente, os SRP empregam técnicas de filtragem de informação e aprendizagem de máquina para gerar recomendações apropriadas aos interesses do usuário a partir da representação de seu perfil (SAMPAIO, 2006).

Para conceber esta representação é necessário detectar os interesses de um usuário que podem ser obtidos por meio implícito ou explícito (SILVA, 2014). Alguns exemplos de captura de informações por meios implícitos são leitura de histórico e aquisição de itens. Já no meio explícito pode ser destacado o preenchimento do questionário de informações requeridas para o uso do software, às avaliações de itens e todas as informações que o usuário registra em seu perfil (REATEGUI, 2005, p.4).

Através da personalização, um SRP é capaz de identificar em tempo real itens de interesse do usuário, apresentando-lhe conteúdo ou produtos relevantes (CAZELLA, 2010, p. 4).

Na forma mais simples, O SRP tenta prever os itens mais adequados com base no perfil do usuário. Sendo assim, este tipo de sistema pode reduzir o tempo que o usuário levaria para encontrar informações relevantes, também como, a probabilidade que um usuário adquira ou acesse um item é bem maior que os sistemas que não são personalizados (CAZELLA, 2010, p.4).

Já os Sistemas de Recomendação Não Personalizados (SRNP), embora se apoiem no processo de recomendação, não são capazes de identificar as preferências e as necessidades do usuário. O papel principal deste tipo de sistema é fazer a coleta e a distribuição das recomendações de forma generalizada para todos os usuários, pois não são capazes de gerar recomendações automaticamente com base no perfil do usuário.

Os SRNP são comumente utilizados em sites de comércio eletrônico, que funcionam como compartilhadores de itens de interesse comum.

2.2 Classificação dos Sistemas de Recomendação

Na literatura, a classificação dos sistemas de recomendação é dividida em três principais técnicas (PAZZANI, 1999; BURKE, 2002; REATEGUI, 2005; SILVA, 2014). São elas:

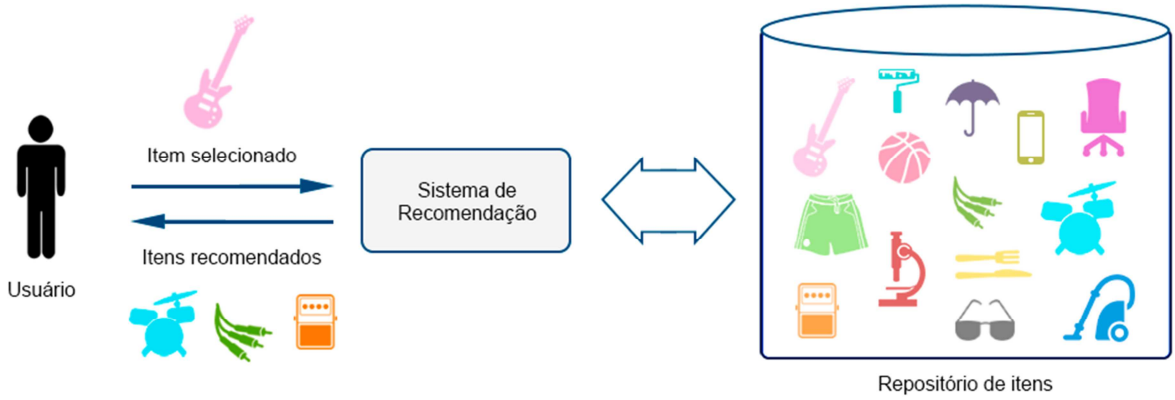
i. **Filtragem baseada em conteúdo**: essa técnica consiste em gerar descrições automáticas dos conteúdos dos itens e comparar estas com as descrições de interesse do usuário (SILVA, 2014).
ii. **Filtragem colaborativa**: utiliza as avaliações de itens feitas pelos “X” usuários e as compara com as avaliações realizadas por um “Y” usuário para gerar a ele, cria-se uma lista de classificados de itens em potencial (SAMPAIO, 2006).

iii. **Filtragem híbrida**: combinam as duas técnicas citadas acima, com a finalidade de eliminar as fraquezas de cada uma (REATEGUI, 2005).

2.2.1 Filtragem Baseada em Conteúdo

A filtragem baseada em conteúdo (FBC) possui como princípio, analisar e categorizar o conteúdo dos itens que foram avaliados pelo usuário e os demais itens do sistema para realizar as recomendações (PAZZANI, 1999). A Figura 2 denota um cenário de recomendação baseada em conteúdo.

Figura 2 – Cenário de recomendação baseada em conteúdo.



Fonte: Adaptado de ARAÚJO (2011)

O conceito desta abordagem leva em consideração que, o usuário tende a se interessar por itens similares aos itens que foram bem avaliados por este no passado. Do modo que, o gênero de um filme seja uma das características extraídas pelo sistema, e sabendo que o usuário fez avaliações positivas sobre os filmes Matrix e Star Trek, o sistema pode entender como uma preferência pelo gênero ficção científica e recomendar o filme *Interstellar*.

Para um sistema de FBC gerar essa recomendação, existem etapas a serem executadas, são elas: criar uma representação do conteúdo dos itens, descobrir interesses dos usuários, aplicar algum algoritmo de classificação ou calcular a similaridade entre dois itens e realizar a recomendação. Cada etapa será descrita detalhadamente a seguir.

2.2.1.1 Representação do Conteúdo

Para Silva (2014), um sistema de FBC necessita construir uma representação do conteúdo do item. Os dados do item podem ser estruturados como, data da notícia, categoria de um produto ou gênero de filme. Entretanto, é comum a presença do formato semiestruturado de dados textuais, tais como, sinopse, descrição, título. No entanto, a forma semiestruturada necessita transformar os dados textuais em uma nova forma.

A representação do conteúdo mais utilizada para dados semiestruturados (a mesma pode ser utilizada em dados estruturados) é feita por meio de termos/palavras (SILVA, 2014). A representação mais popular é conhecida como *Bag-of-words* (BOW), no qual, criam-se vetores de palavras verificando a frequência que aparecem em cada documento. Outros exemplos de abordagens que podem ser utilizadas para gerar representações textuais são: *Bag-of-words with n-gram, Phrases, Bag-of-words and word position, Concept categories* e entre outras (SILVA, 2014 *apud* KOSALA; BLOCKEEL, 2000).

O processo de construção de qualquer representação é composto de três partes, a primeira denominada extração, que decompõe o texto em termos, tendo forte dependência do idioma no qual o texto foi escrito e normalmente são aplicados filtros como *stopword* e *stemming* (SILVA, 2014 *apud* WANG; WANG, 2005).

Na segunda etapa, denominada seleção, os atributos extraídos são mapeados, geralmente em um vetor de características e a cada atributo se estabelece um peso. Para definição dos pesos, pode se realizar por cálculos como *Document Frequency* (DF), *Term Frequency* (TF), *Inverse Document Frequency* (IDF) e *Information Gain* (IG) (SILVA, 2014 *apud* SEBASTIANI, 2002).

E, a última etapa, é a redução de dimensionalidade. Desenvolver uma forma de reduzir a dimensionalidade dos vetores de termos, mantendo a representatividade dos documentos. Segundo Silva (2014), um dos desafios para um recomendador Baseado em Conteúdo é se manter escalável.

Algumas outras formas de representação de documentos são a representação baseada em grafo (SILVA, 2014 *apud* JIN; SRIHARI, 2007), a fusão

do conteúdo extraído do texto em fusão com enciclopédias de senso comum, como a Wikipédia[12].

2.2.1.2 Descoberta de Interesses dos Usuários

Utilizando o histórico do usuário, é possível modelar conjuntos de interesses ($d+$) e os desinteresses ($d-$) e por meio dessas listas que inicia o processo de "conhecer" o usuário. Os conjuntos $d+$ e $d-$ são atribuídos por termos da representação do item que o usuário demonstrou interesse ou desinteresse. Em seguida, o sistema deve identificar a qual conjunto ($d+$ ou $d-$) pertence os itens ainda não avaliados pelo usuário. Os meios mais tradicionais de descobrir qual conjunto o item desconhecido pertence são: Algoritmos de AM (classificadores) ou utilizando cálculos de similaridade entre os itens (SILVA, 2014).

2.2.1.3 Algoritmos de Classificação

Os algoritmos de classificação são utilizados para associar um documento em classes preestabelecidas. O algoritmo é responsável por dizer se o item é de interesse ou não do usuário, predizendo em qual classe de interesse o item pertence, também definindo o grau de relevância de cada documento em relação aos interesses do usuário.

Alguns exemplos clássicos de algoritmos de classificação são *Naïve Bayes* e *Rocchio's Algorithm* (SILVA *apud* AAS; EIKVIL, 1999; BAHARUDIN *et al.*, 2010).

2.2.1.4 Cálculo de Similaridade

O cálculo de similaridade tem o objetivo de calcular a similaridade entre dois itens, utilizando a representação construída de cada um dos itens, realiza uma comparação entre as duas representações. Ao final do cálculo de similaridade é

[12] Wikipédia é uma enciclopédia de senso comum. Disponível em: <https://pt.wikipedia.org>. Acesso em: 21/05/2017.

possível estabelecer um ranking de quais itens são mais similares às preferências do usuário ($d+$ e $d-$).

Em exemplos de cálculos de similaridade podemos citar Similaridade *Jaccard* e Similaridade Cosseno (SILVA, 2014 *apud* REAL; VARGAS, 1996).

2.2.1.5 Recomendador Baseado em Conteúdo

Após a descoberta de itens de interesse do usuário, é possível executar a recomendação dos itens. Esta etapa consiste em listar os itens de possível interesse ao usuário e manter a lista sempre atualizada. Para a atualização da lista, é feita o cálculo de similaridade entre o novo item e os interesses do usuário já descoberto pelo sistema. Assim quando um novo item nas atualizações é adicionado ou quando o usuário avalia algum novo item, seu perfil de usuário é atualizado, para melhorar ainda mais a precisão das recomendações.

2.2.1.6 Vantagens e Desvantagens

As vantagens e desvantagens citadas por Adomavicius *et al*. (2005) e Silva (2014), são:

- É necessário apenas conhecer o próprio usuário, ou seja, não é preciso conhecê-lo em comunidade. Não é necessário encontrar usuários com preferências semelhantes para realizar a recomendação.
- Normalmente há mais usuários que itens, então seu custo operacional é menor que outros tipos de abordagens.
- Privacidade, pois as recomendações não utilizam os dados de outros usuários.

Em contrapartida, a abordagem baseada em conteúdo tem as seguintes limitações:

- Análise de conteúdo é limitada: o conteúdo de dados pouco estruturados é difícil de ser analisado. Por exemplo, a extração e análise de conteúdo

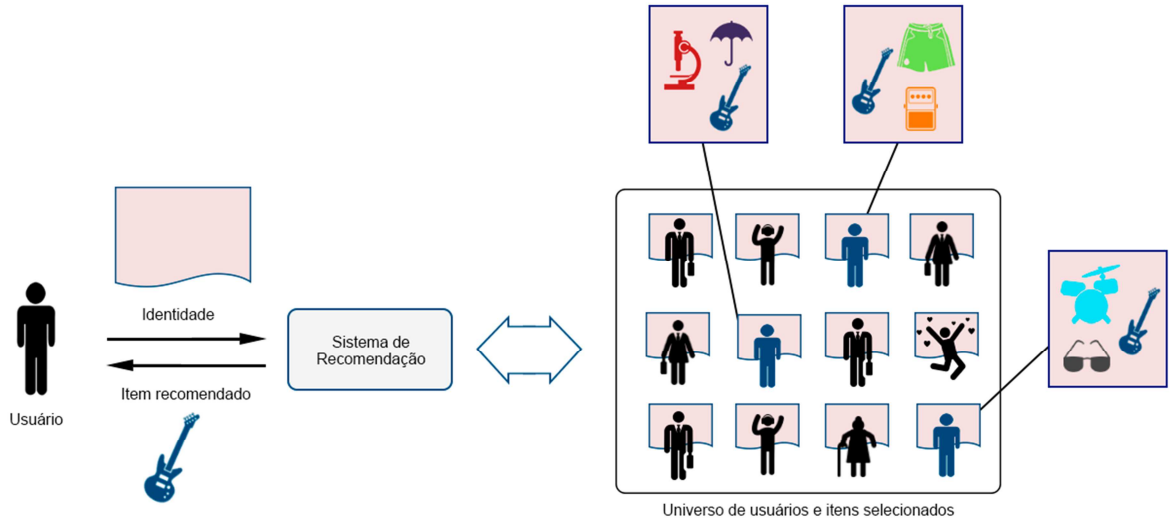
multimídia, como vídeo e som, são muito mais complexas que documentos textuais. Como também, esta abordagem não consegue diferenciar um documento bem escrito de um mal escrito.

- Superespecialização: ocorre quando o sistema recomenda somente itens similares aos que foram avaliados positivamente. Desta forma, itens de que não estão de acordo com o perfil do usuário, não serão apresentados.
- Dificuldades em filtrar com base em qualidade, estilo ou ponto de vista: os algoritmos utilizados para predizer quais itens estão de acordo com as preferências do usuário, não são capazes de “aprender” os aspectos mais sutis que excedem a similaridade textual. Como, por exemplo, os filmes “A culpa é das estrelas” e “Jornada nas estrelas” não são similares.

2.2.2 Filtragem Colaborativa

A filtragem colaborativa (FC) foi criada para solucionar os pontos em aberto da filtragem baseada em conteúdo e se diferencia por não exigir compreensão ou conhecimento do conteúdo dos itens. Sua essência é identificar usuários semelhantes, ou seja, que tenham avaliado uma quantidade relevante de itens de modo semelhante e desta forma apresentar uns aos outros os itens distintos bem avaliados por cada um. A Figura 3 ilustra um cenário de recomendação colaborativa.

Figura 3 – Cenário de recomendação colaborativa.



Fonte: Adaptado de ARAÚJO (2011)

A FC é a técnica de recomendação mais utilizada e que apresenta algoritmos mais consolidados. Segundo Goldberg, o primeiro indício de abordagem FC foi desenvolvido pela Xerox Palo Alto Research Center onde o software experimental Tapestry, um sistema de envio de e-mails, com o objetivo de empregar as recomendações de mensagens eletrônicas em grupos de notícias ou listas de e-mails. As pessoas ajudavam umas as outras, executando uma espécie de filtragem ao lerem um e-mail, votando no seu conteúdo como “bom” ou “ruim” e adicionando comentários expressando sua opinião. Essas avaliações eram gravadas junto com o e-mail e podiam ser acessadas por filtros de busca aplicados por outros usuários do sistema. Assim, um usuário poderia acessar as mensagens não apenas pelo conteúdo, como também, pelas opiniões de outros usuários (GOLDBERG *et al.*, 1992). Por exemplo, poderia se aplicar um filtro “mostrar todas as mensagens que Letícia avaliou como boas”.

Herlocker (2000) e Goldberg *et al.* (1992) esclarecem que o sistema de filtragem Tapestry requeria que os usuários especificassem o relacionamento de predição entre suas opiniões de modo explícito, mas também provia recomendações de modo implícito, possibilitando que um assinante de uma lista criasse um filtro para mostrar-lhe as mensagens que foram respondidas por usuários que respondem as mensagens mais interessantes do fórum (GOLDBERG *et al.*, 1992; HERLOCKER, 2000).

Em um sistema de FC, o perfil do usuário consiste em um vetor de itens avaliados. As avaliações feitas pelo usuário podem ser binárias como, por exemplo, favoritar ou não um filme. Também, é possível que a avaliação seja representada por um número real que indica o grau de preferência do usuário como votar em um filme entre 0 a 5 estrelas.

A técnica de FC pode ser dividida em duas categorias: baseado em memória e baseado em modelo (BOGERS, 2009).

i. Os algoritmos baseados em memória foram os primeiros a serem desenvolvidos e por processarem as informações na memória são chamados de preguiçosos (*lazy*). Este método concentra-se no cálculo da similaridade entre usuários ou entre itens. São mais fáceis de implementar

e apresentam bons resultados. Mas podem apresentar problemas quando há poucos itens ou poucos usuários semelhantes (SILVA, 2014).

ii. Os algoritmos baseados em modelo foram criados com o objetivo de tentar aprimorar o desempenho dos sistemas de recomendação. Incorporam conceitos de aprendizado de máquina e de mineração de dados. Um exemplo são as redes neurais bayesianas. Conforme Silva (2014), estes algoritmos possuem como propriedade “a criação de um modelo preditivo baseado nas avaliações dos usuários como fase preliminar as recomendações”. O uso de algoritmos de Aprendizado de Máquina para a construção do modelo torna a etapa de treinamento mais árdua, por causa disto, estes algoritmos são conhecidos como “impacientes” (*eager*). Porém esse treinamento é compensado durante a tarefa de recomendação.

Neste trabalho, não serão estudados os algoritmos baseados em modelo.

2.2.2.1 Filtragem Colaborativa Usuário-Usuário

A filtragem colaborativa usuário-usuário possui como objetivo recomendar para um usuário alvo, os itens que outros usuários, com gostos similares a ele, gostaram no passado. A similaridade entre dois usuários é calculada com base em seu histórico de avaliações. O princípio desta abordagem é que a nota do usuário alvo a para um item i é provavelmente similar à nota recebida por outro usuário b, se a e b avaliaram itens no passado de forma semelhante.

Para construir um FC usuário-usuário de uma forma genérica o sistema deve executar as seguintes etapas: i) criar representações, ii) encontrar usuários “vizinhos”, iii) selecionar itens recomendáveis e iv) recomendar a lista de itens com as melhores notas.

2.2.2.2 Representação

A representação em um sistema de FC é uma matriz de utilidade m x n, onde m representa os usuários e n representa os itens numa perspectiva histórica. Numa

matriz usuário-item r que, por exemplo, represente os filmes favoritados, o elemento $r_{ij,}$ pode apresentar o valor igual a 1, se o usuário favoritou ou igual a 0, se o usuário não favoritou. A Tabela 2 mostra um exemplo de uma matriz usuário-item.

Tabela 2 – Matriz usuário-item.

	Interstellar	Batman: O Cavaleiro das Trevas	A Origem	Perdido em Marte	Procurando Nemo
João	1		1	1	
Maria		1			1
Letícia	1	1	1		1
José	1		1	1	

Fonte: Elaborado pelo autor

2.2.2.3 Encontrar Usuários Vizinhos

Após a composição da representação, a próxima etapa, e uma das mais importantes em FC, é identificar os usuários “vizinhos” do usuário alvo. O algoritmo para encontrar os possíveis vizinhos pode ser dividido em duas partes: (1) calcular a medida de proximidade (ou similaridade) e (2) selecionar a vizinhança.

Os métodos mais utilizados para calcular a similaridade entre dois usuários e, consequentemente descobrir a vizinhança são o Coeficiente de Correlação de Pearson e a Similaridade Cosseno.

O coeficiente de Correlação de Pearson (REATEGUI, 2005) é normalmente aplicado quando as avaliações dos usuários estão dispostas em uma escala numérica discreta, normalmente entre 1 até 5 ou 1 a 10. Por exemplo, avaliações com estrelas para filmes em uma escala de 1 a 5, que podem representar valores entre “ruim” e “excelente".

A Equação 1 apresenta a Correlação de Pearson (REATEGUI, 2005):

$$corr_{ab} = \frac{\sum_i (r_{ai} - \bar{r}_a)(r_{bi} - \bar{r}_b)}{\sqrt{\sum_i (r_{ai} - \bar{r}_a)^2 \sum_i (r_{bi} - \bar{r}_b)^2}} \quad (1)$$

Sendo o $corr_{ab}$ a correlação do usuário alvo a com um determinado usuário b; r_{ai} é a avaliação que o usuário ativo a atribuiu para o item i; r_{bi} é a avaliação que o usuário ativo b atribuiu para o item i; $\bar{r}_a$ é a média de todas as avaliações do usuário ativo a em comum com o usuário b; $\bar{r}_b$ é a média de todas as avaliações do usuário ativo b em comum com o usuário a. Note que é necessário mais de uma avaliação em comum para que o índice seja útil. Ao final, os resultados são considerados pesos e variam entre 1 para similaridade total e -1 para total dissimilaridade. No apêndice A encontra-se um exemplo da realização deste cálculo.

O segundo método, a Similaridade Cosseno é comumente aplicado quando as avaliações apresentam valores binários, pode representar valores binários, como um item como “bom” ou “ruim”, “favorito” ou “não favorito”.

A Similaridade Cosseno considera os vetores de avaliação entre dois usuários ou itens com quais quer avaliar a similaridade. Para obter o valor da similaridade de um usuário alvo e outro usuário, é necessário comparar as matrizes m x n dos dois usuários.

Na matriz m x n de um usuário-item $r_{a,i}$, a similaridade entre dois itens i e j é definida pelo cosseno entre os vetores de dimensão n correspondentes às i-ésima e j-ésima colunas da matriz R. A similaridade por cosseno entre itens $i = \{i_1, i_2, \dots, i_n\}$ e $j = \{j_1, j_2, \dots, j_n\}$ é dada por (ROZENDO, 2017, p. 31):

$$\cos(a, b) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_{i=j}^{m} r_{a,i} r_{b,i}}{\sqrt{\sum_{i=1}^{m} (r_{a,i})^2} \sqrt{\sum_{i=1}^{m} (r_{b,i})^2}} \quad (2)$$

Na Equação 2, $\vec{a}$ e $\vec{b}$ representam os vetores, $r_{a,i}$ é a avaliação que o usuário alvo deu para o item i, $r_{b,i}$ é a avaliação de outro usuário para o mesmo item. Os resultados do cálculo de similaridade cosseno variam entre 0, para total dissimilaridade e 1, para similaridade total.

Outros métodos utilizados são: a Distância Euclidiana (MELO, 2016, p. 40), *kNN* (CABRÉ, 2011), a Correlação de *Ranking Spearman* (FAZIO, 2013, p. 40), Correlação *Kendall* (LIRA, 2004, p. 106), entre outras que são pouco utilizadas na literatura, uma vez que a grande maioria dos testes com filtros colaborativos utilizam Correlação de Pearson ou Cosseno.

2.2.2.4 Selecionar Itens Recomendáveis

Após achar os vizinhos mais próximos, o próximo passo é recomendar os itens dos usuários vizinhos que o usuário alvo. O requisito principal é que os itens têm quer ser novos, ou seja, por exemplo, em um sistema de recomendação de filmes, encontrar filmes de usuários vizinhos que o usuário alvo, ainda não tenha assistido. Logo, seguindo o exemplo, os itens recomendáveis podem ser (1) todos os filmes não assistidos, (2) selecionar aleatoriamente entre todos os filmes não assistidos até encontrar os *Top*-N filmes, ou (3) pode-se restringir a busca apenas aos filmes assistidos pelos *k* usuários mais semelhantes (*k*-vizinhos) e que ainda não foram assistidos pelo usuário alvo.

Para encontrar recomendações de itens podemos utilizar diversos métodos como médias, *kNN*, métodos híbridos, entre outros. Para exemplificar, será utilizada como método a média ponderada das avaliações dos vizinhos que tiveram um bom percentual de similaridade. A Equação 3 corresponde ao cálculo da predição por média ponderada (REATEGUI, 2005):

$$p_{ai} = \bar{r_a} + \frac{\sum_{b=1}^{n}(r_{bi} - \bar{r_b}) * corr_{ab}}{\sum_{b=1}^{n} |corr_{ab}|} \quad (3)$$

Sendo que $corr_{ab}$ corresponde à similaridade entre o usuário alvo a com um determinado usuário b; p_{ai} corresponde à predição de um item i para um usuário a; $\bar{r_a}$ é a média de todas as avaliações do usuário alvo a aos itens que foram computados por todos os seus usuários similares; r_{bi} é a avaliação que o usuário alvo b atribuiu ao item i; $\bar{r_b}$ é a média de todas as avaliações do usuário b em comum com o usuário a. Este cálculo deve percorrer todos os itens de um

determinado usuário b que não tenham sido adquiridos pelo usuário alvo a. E assim, ao final, terá uma lista de itens com suas notas (frequência).

2.2.2.5 Recomendar Itens

De acordo com o método utilizado na seleção de itens, alguns critérios podem ser levados em consideração na hora de apresentar os itens ao usuário. No caso de uma seleção aleatória, pode-se utilizar como critério “itens que obtiveram nota maior que x serão recomendadas” e também, “itens que obtiveram notas superiores a média do usuário serão recomendadas”.

2.2.2.6 Filtragem Colaborativa Item-Item

A filtragem colaborativa Item-Item veio para solucionar os problemas de escalabilidade da filtragem colaborativa Usuário-Usuário à medida que o número de usuários cresce no sistema. A filtragem colaborativa Item-Item é capaz de atender as grandes demandas de websites de comércio eletrônico, pelo fato de serem escaláveis. A Amazon.com desde 2003 utiliza este tipo de método, sendo comum encontrar a sentença “quem gostou de X se interessou por Y”, ou até mesmo itens relacionados. A Figura 4 apresenta um trecho de uma página da loja virtual da Amazon.com, no qual se destaca a abordagem de suas recomendações.

Figura 4 – Abordagem de recomendação da Amazon.com.

Fonte: AMAZON.COM (2016)

O funcionamento da abordagem Item-Item é que ao invés de usar semelhanças entre os usuários, considera-se a semelhança entre os itens. Se os itens têm avaliações próximas dos mesmos usuários, então eles são semelhantes e os usuários devem ter preferências similares para itens similares. A abordagem Item-Item oferece melhor desempenho quando comparado a Usuário-Usuário nas situações em que o número de usuários é superior ao número de itens (MELO, 2016).

As tarefas de predição Item-Item são parecidas com a baseada em Usuário-Usuário, podem ser utilizados os métodos Correlação de Pearson, Similaridade Cosseno e *kNN* com algumas ressalvas.

2.2.2.7 Vantagens e Desvantagens

A popularização dos algoritmos de recomendação baseados na filtragem colaborativa se deve as suas vantagens, que são:

- Não depender da análise do conteúdo.
- Levar em consideração os conceitos complexos como técnica de processamento de textos com qualidade.
- Apresentar recomendações inesperadas, ou seja, que não possuam características semelhantes aos itens de seu perfil e que não foram pesquisados de forma ativa.
- Possibilidade de formação de comunidades de usuários pela identificação de seus gostos e interesses similares.

Embora, a propagação de filtragem colaborativa ajuda desenvolver melhorias nos algoritmos, ainda existem muitos desafios pertinentes:

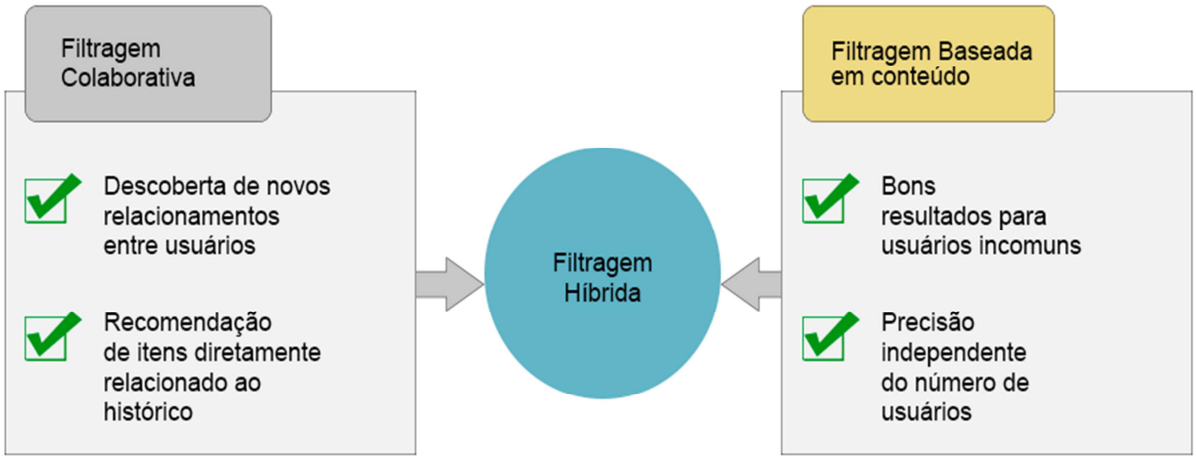
- Problemas de escalabilidade já que muitos sistemas possuem muito mais usuários do que itens.
- Quando o usuário possui preferências muito específicas, sendo pouco semelhantes aos demais usuários (*Black Sheep*). Por conseguinte, este usuário não se beneficiará das recomendações.

2.2.3 Filtragem Híbrida

A Filtragem Híbrida (FH) emprega técnicas vindas tanto da FBC quanto da Filtragem Colaborativa como uma forma unir os pontos fortes destas e limitar os pontos negativos de ambas. Mas também, pode ser utilizada a FC com outras técnicas (REATEGUI, 2005).

A ideia desta filtragem é baseada na busca de itens de acordo com os perfis de interesses dos usuários, mas considerando também a similaridade de conteúdo entre os itens. A Figura 5 apresenta as características da FH herdadas da FC e da FBC.

Figura 5 – Características da Filtragem Híbrida.



Fonte: Adaptado de REATEGUI (2005)

Os sistemas de recomendação na combinação da FBC e a FC podem ter sete tipos (BURKE, 2002). São eles:

1) **Ponderado**: os resultados de várias técnicas de recomendação são agrupados para produzir uma única recomendação.
2) **Alternado**: o sistema de recomendação alterna várias técnicas, dependendo da situação, para produzir a recomendação;
3) **Mista**: utiliza várias técnicas simultaneamente;
4) **Combinação de características**: combina características de dados obtidos na aplicação de diferentes como entrada para uma única técnica;

5) **Cascata**: uma técnica gera a recomendação e outra a classifica;
6) **Aumento de características**: uma técnica que gera a recomendação como entrada para a outra técnica;
7) **Meta-nível**: uma técnica gera um modelo que servirá de entrada para outra técnica.

As vantagens de utilizar esta abordagem são:

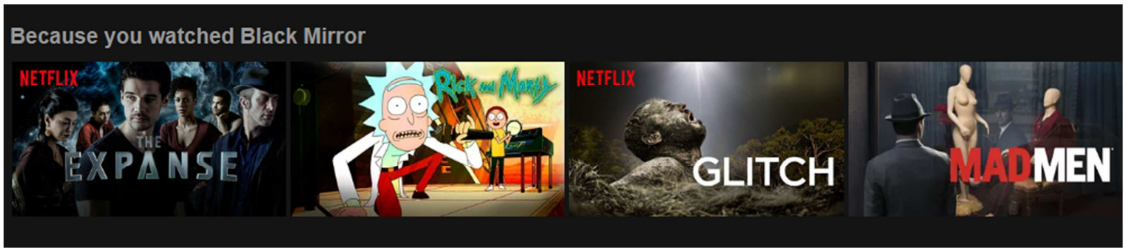
- Recomendação de itens diretamente relacionada ao histórico, suas experiências são levadas em consideração;
- Bons resultados para usuários incomuns, pois não é necessário que haja usuários semelhantes a este;
- Precisão independente do número de usuários;
- Utilizando recomendações baseadas em conteúdo, é possível lidar com itens não vistos por outros usuários.

Em sua desvantagem pode-se assumir que este tipo de recomendação exige um estudo muito aprofundado, na análise da variação na qualidade da recomendação, devido à multidimensionalidade da informação.

2.3 Exemplos de Sistemas de Recomendação

Alguns exemplos reais de sistemas de recomendação postos em prática, pode-se indicar lojas virtuais como Amazon.com, eBay [13] e aplicações como Netflix, Spotify, entre outros. A Figura 6 apresenta a abordagem de recomendação de filmes similares da Netflix.

Figura 6 – Abordagem de recomendação da Netflix.

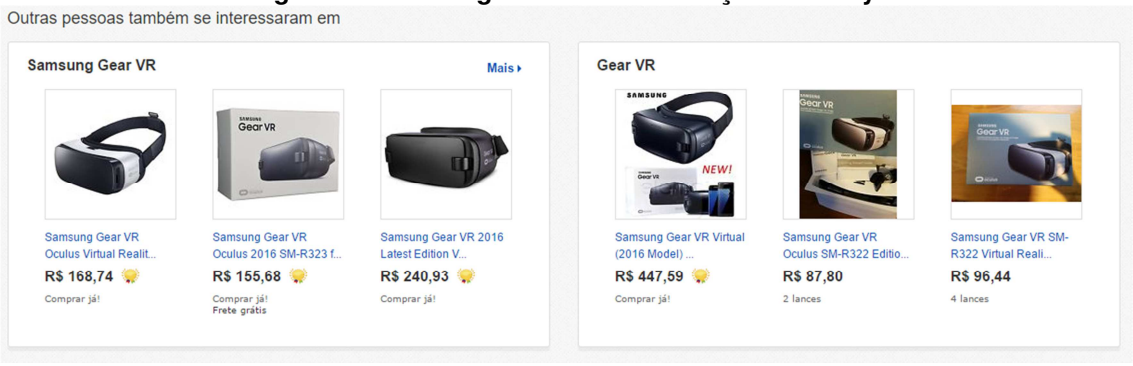


Fonte: NETFLIX (2016)

[13] EBAY. Disponível: <http://ebay.com>. Acesso em: 22/05/2017.

O Netflix apresenta várias listas de recomendações em que cada lista é direcionada por uma razão diferente. Frequentemente, o Netflix usa expressões como “Porque você assistiu...”, enquanto lojas virtuais têm apelos para “Clientes que compraram X também compraram Y”, “Outras pessoas que se interessaram por X também se interessaram em”. A Figura 7 apresenta um exemplo de abordagem de recomendação, é muito comum encontrar expressões como a encontrada no eBay: “Outras pessoas também se interessam em”.

Figura 7 – Abordagem de recomendação do eBay.



Fonte: EBAY (2016)

Os sistemas de recomendação tornaram-se populares em grandes e-commerces a fim de evitar a sobrecarga de informações para os clientes e tornar os produtos desejados mais fáceis de serem encontrados.

2.4 Exemplos de Sistemas de Recomendação Acadêmicos

Os projetos acadêmicos que enfatizam a história dos Sistemas de Recomendação são: *Ringo*[14] (CAZELLA, 2010), *GroupLens*[15] (CAZELLA, 2010) e *Fab*[16] (CAZELLA, 2010).

- *Ringo*: Criado no *Massachusetts Institute of Technology*. Ringo é um sistema desenvolvido para a recomendação de música, no qual seu

[14] Ringo. Disponível em <http://jolomo.net/ringo.html>.
[15] GroupLens. Disponível em <https://grouplens.org>.
[16] Fab, para saber mais informações sobre o desenvolvimento deste software e seu sistema de recomendação, acesse: <https://www.ischool.utexas.edu/~i385q/readings/Balabanovic_Shoham-1997-Fab.pdf>.

sistema de recomendação sonda similaridades entre os gostos dos usuários para fazer recomendações, baseado em que as pessoas apresentam tendências e padrões entre as preferências musicais. No qual, as pessoas descrevem seus gostos através de suas avaliações que compõe o perfil do usuário. Para o funcionamento usuários similares são recomendados, a partir da comparação entre perfis, o sistema pode identificar álbuns/artistas que ainda não foram avaliados pelo mesmo (CAZELLA, 2010).

- *GroupLens*: O *GroupLens* foi um projeto de pesquisa desenvolvido pela Universidade de *Minnesota*, em que se encarregava de mostrar notícias de interesse do usuário baseado na filtragem colaborativa. O sistema coleta as avaliações dos usuários referentes aos artigos lidos e utiliza para encontrar vizinhos próximos com avaliações semelhantes. Prevendo se um usuário gostará de um artigo segundo seu vizinho mais próximo (CAZELLA, 2010).
- *Fab*: Este sistema foi originado em *Stanford*. Baseado em filtragem híbrida (combina filtragem colaborativa e filtragem baseada em conteúdo) recomenda páginas web para usuários. Sendo baseado na filtragem híbrida, os perfis de usuário são gerados a partir da análise de conteúdo e comparados para identificar usuários semelhantes como é feito na recomendação colaborativa. O sistema atua com dois agentes, um na coleta dos documentos na web e outro na seleção de páginas apresentados aos usuários como recomendação (CAZELLA, 2010).

3 PROJETO DO APLICATIVO

Este capítulo detalha o processo do desenvolvimento de um aplicativo de recomendação de filmes por meio da engenharia de software, a ciência capaz de representar todos os aspectos de produção de software. Também descreve todas as tecnologias necessárias para o desenvolvimento do aplicativo.

Antes de começar a desenvolver um software é necessário entender seus objetivos, funções e uma série de atividades que devem analisar e projetar para que o software seja desenvolvido de forma eficiente. Para que isso ocorra, é necessário envolver metodologias de desenvolvimento de engenharia de software.

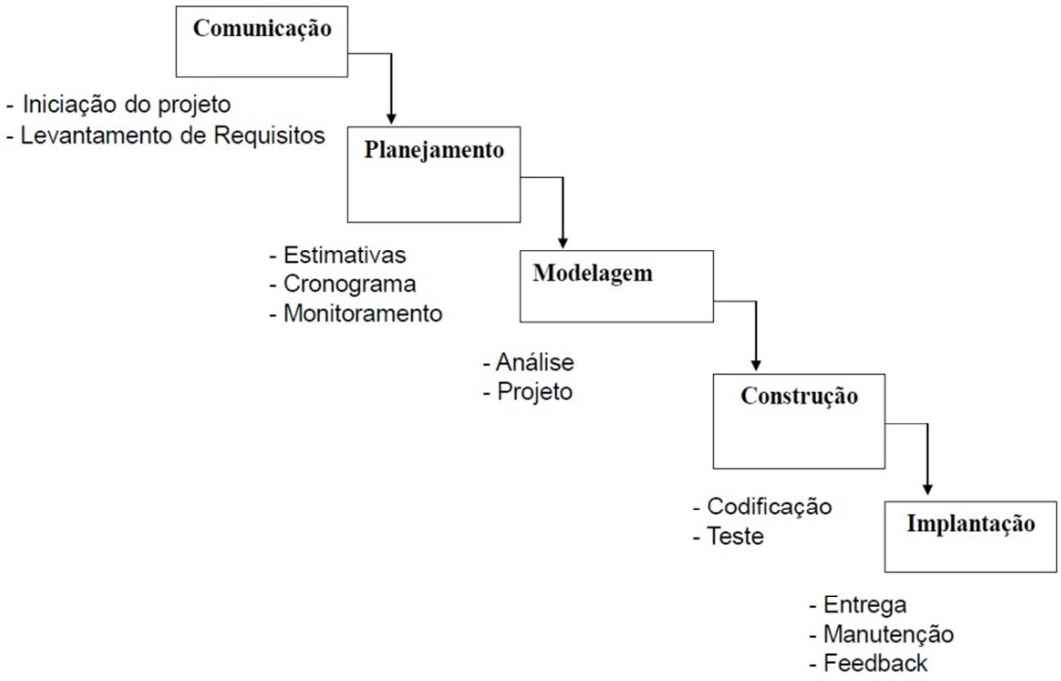
3.1 Metodologias de Desenvolvimento

Segundo Pressman (2007, p. 31) “os métodos de engenharia de software proporcionam os detalhes de “como fazer” para construir o software”. Ao longo dos anos, foram concebidas várias metodologias de desenvolvimento de software, sendo que uma metodologia conhecida como o modelo cascata serviu e ainda serve de base para grande parte das metodologias modernas, como também é muito utilizada até hoje e foi à metodologia escolhida para desenvolver este projeto.

O Modelo Cascata também chamado de ciclo de vida clássico, foi descrito por Winston W. Royce em 1970. Até meados da década de 1980 foi o único modelo com aceitação geral. Este modelo propõe uma abordagem sequencial e sistemática para o desenvolvimento, com o fim de estabelecer ordem no desenvolvimento de grandes produtos de software.

Esta metodologia propõe um conjunto de etapas sistemáticas e sequenciais, sendo que a próxima etapa só pode ser iniciada depois que sua predecessora for finalizada (PRESSMAN, 2007). A Figura 8 decompõe os processos do modelo cascata.

Figura 8 – Processos do Modelo Cascata.



Fonte: Adaptado de PRESSMAN (2011)

O Modelo Cascata possui cinco importantes etapas: comunicação, planejamento, modelagem, construção e implantação. O Quadro 1 descreve cada etapa deste modelo.

Quadro 1 – Principais etapas do Modelo Cascata.

Atividade	Descrição
Comunicação	Antes de iniciar o trabalho técnico, é necessário compreender os objetivos do software, checar sua viabilidade e realizar um levantamento de requisitos para definir funções e características do software.
Planejamento	Determinam-se as tarefas técnicas a serem conduzidos, os riscos, os recursos que serão necessários, os produtos resultantes e um cronograma de trabalho.
Modelagem	Cria-se um esboço de modo que possa entender a ideia como um todo. E como as partes conseguintes se encaixarão e entre outras características. Concebem-se

	modelos para melhor compreender as necessidades do software.
Desenvolvimento	Essa atividade constitui-se em gerar os códigos do software, implementá-los e fazer testes para revelar erros de codificação.
Implantação	O software (como um software completo ou um incremento parcial) é entregue ao cliente, que avalia o produto e fornece *feedback* baseado na avaliação.

Fonte: PRESSMAN (2011)

Nos próximos subcapítulos será detalhada cada atividade de engenharia de software baseando-se nas etapas fundamentais sugeridas por Pressman e aplicadas no contexto deste projeto.

3.2 Comunicação

A comunicação é primeira etapa de um projeto de software, onde são executadas tarefas como: a definição dos objetivos do software, a análise de viabilidade e o levantamento de requisitos conhecida como a engenharia de requisitos.

3.2.1 Análise de Viabilidade

O objetivo de um estudo de viabilidade , como o próprio nome já diz, é avaliar sob o ponto de vista operacional, técnico, econômico e organizacional se o projeto é viável. Para Sommerville (2007) todo sistema novo deve realizar um estudo de viabilidade:

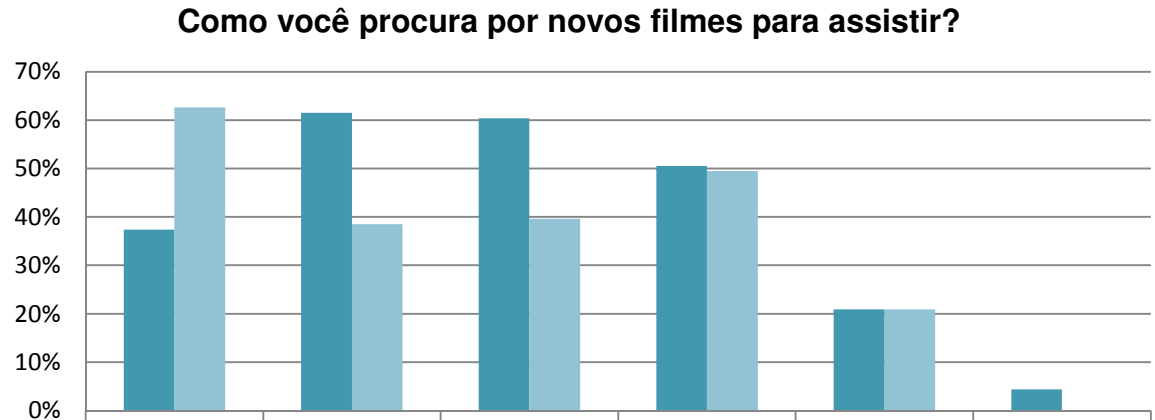
> "Para todos os sistemas novos, o processo de requisitos deve começar com um estudo de viabilidade. A entrada para o estudo de viabilidade é uma descrição geral do sistema e de como ele será utilizado" (SOMMERVILLE, 2007, p. 103).

A pesquisa de viabilidade analisou a proposta do aplicativo de filmes Cine Collection e, por meio de uma pesquisa avaliou sua viabilidade.

A proposta do aplicativo de filmes nasceu pela dificuldade de lembrar-se da lista de filmes assistidos e de achar boas recomendações. Após uma pesquisa, constatou-se que, para celular não há aplicativos eficientes para a realização destas tarefas. Para testar a viabilidade de um aplicativo de filmes, foi executada uma pesquisa no qual as perguntas levariam ao resultado se o aplicativo de filmes Cine Collection é viável ou não.

A pesquisa entrevistou em grande parte do público, jovens entre 19 a 25 anos, como resultado obteve-se que a maioria dos entrevistados não utiliza meios de organizar filmes já assistidos e que a procura por novos filmes para assistir é feita por recomendações pela pesquisa na internet ou de acordo com recomendações de serviços de *streaming*. Porém, os resultados obtidos também apontam que, 48,2% dos entrevistados não acreditam ter controle dos filmes que assistem. A pesquisa completa é encontrada no Apêndice B. A Figura 9 apresenta como os usuários procuram por novos filmes para assistir sendo que o usuário pode escolher mais de uma opção.

Figura 9 – Pergunta da pesquisa elaborada pelo autor.



	Cinema	Recomendações de serviços de streaming (Netflix, Google Filmes)	Pesquisando na internet	Recomendações de amigos	Comerciais de TV (trailer)	Outros
Sim	37%	61,50%	60,40%	50,50%	20,90%	4,40%
Não	62,60%	38,50%	39,60%	49,50%	20,90%	

Fonte: Elaborado pelo autor

De acordo com os resultados obtidos, foi decidido prosseguir com o desenvolvimento do projeto. Seguindo o processo de engenharia de software, depois dos estudos iniciais de viabilidade, segundo Sommerville (2007), “[...] o próximo estágio do processo de engenharia de software é o levantamento e análise de requisitos”, denominada Engenharia de Requisitos.

3.2.2 Engenharia de Requisitos

A engenharia de requisitos (RE – *Requirements Engineering*) é o processo de descobrir, analisar, documentar e verificar requisitos de um sistema. Um requisito pode ser definido como uma descrição dos serviços fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2007). Tradicionalmente, os requisitos são divididos em dois tipos: requisitos funcionais e requisitos não funcionais.

3.2.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema deve fazer, isto é, definem a funcionalidade desejada do software (SOMMERVILLE, 2007). O Quadro 2 apresenta os requisitos funcionais deste projeto.

Quadro 2 – Requisitos funcionais do projeto.

Identificação	Requisito Funcional	Categoria	Prioridade
RF001	O sistema deve ter uma tela de carregamento.	Design e Programação	Baixa
RF002	O sistema deve ter login com Facebook	Design e Programação	Alta
RF003	O sistema deve exibir uma lista de filmes recomendados personalizado para cada usuário.	Programação	Alta
RF004	O sistema deve ter um motor de busca de filmes que exibirá a foto e o nome do filme procurado.	Programação	Alta
RF005	Cada filme deverá conter ao menos uma foto, o título, a sinopse do filme, o elenco e a equipe técnica.	Design e Programação	Alta

RF006	O sistema deve exibir um botão de “Favoritar”	Design e Programação	Média
RF007	O sistema deve ter a possibilidade de o usuário avaliar um filme	Programação	Alta
RF008	O usuário deverá ter um perfil com uma lista dos filmes avaliados.	Design e Programação	Baixa

Fonte: Elaborado pelo autor

3.2.2.2 Requisitos Não Funcionais

“Os requisitos não funcionais são aqueles não diretamente relacionados às funções específicas fornecidas pelo sistema” (SOMMERVILLE, 2007). O Quadro 3 apresenta os requisitos não funcionais deste projeto.

Quadro 3 – Requisitos não funcionais do projeto.

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	Portabilidade: o aplicativo deve funcionar na maioria dos celulares.	Programação	Alta
RNF002	Requisitos Técnicos: o aplicativo não funcionará sem rede Wifi ou 3G.	Programação	Muito Alta
RNF003	Usabilidade: oferecer botões e links de tamanhos razoáveis facilitando a navegação.	Design	Alta
RNF004	Design: o aplicativo deve conter um design criativo e autêntico, mas que seja fácil de utilizar na visão dos usuários.	Design	Média
RNF005	Desempenho: Tempo limite de processamento de todas as recomendações não deve ultrapassar de 2000 milissegundos.	Programação	Alta
RNF006	Segurança: O servidor deve realizar um backup semanal do banco de dados.	Programação	Alta

Fonte: Elaborado pelo autor

3.3 Planejamento

O planejamento do aplicativo de filmes, baseando-se no modelo cascata, propõe uma análise dos principais riscos, os recursos que serão necessários para desenvolver o projeto e o cronograma.

3.3.1 Análise de Riscos

Os riscos podem afetar o cronograma de projeto ou a qualidade do software. Os riscos podem ser relacionados: ao projeto: afetando a programação ou os recursos do projeto; ao produto: afetando a qualidade ou o desempenho do software que está em desenvolvimento e ao negócio: afetando a quem está desenvolvendo ou adquirindo o software (SOMMERVILLE, 2007, p. 69). O Quadro 4 apresenta os riscos encontrados que podem afetar este projeto.

Quadro 4 – Riscos encontrados que podem afetar este projeto.

Risco	Tipo de Risco	Descrição
Indisponibilidade da API de filmes	Produto	A API de filmes que será utilizada para ler uma biblioteca de filmes pode ficar indisponível.
Alteração de requisitos	Projeto	Podem haver mais requisitos ou mudança nos requisitos do que o previsto.
Mudança de tecnologia	Negócio	A tecnologia básica a qual o software opera pode ser superada por uma nova tecnologia.
Tamanho subestimado	Projeto e Produto	O tamanho ou a complexidade do sistema pode ser subestimado.
Prazo subestimado	Projeto e Negócio	Não entregar o projeto no prazo estipulado.

Fonte: Elaborado pelo autor

3.3.2 Recursos e Ferramentas

Esta seção contempla as ferramentas de programação e os conceitos necessários para o desenvolvimento do aplicativo.

O aplicativo foi desenvolvido para celulares Android, utilizando o ambiente de desenvolvimento Android Studio. Pela facilidade de implementação foi empregado uma base de dados de filmes gratuita denominada The Movie Database (TMDB) [17] e, os dados dos usuários e outros dados do aplicativo serão armazenados em um banco de dados Firebase. O sistema possui um login com o Facebook, portanto também comportará a biblioteca do Facebook.

3.3.2.1 A Plataforma Android e suas Ferramentas

A plataforma Android é o sistema operacional mais utilizado no mundo (STAT COUNTER, 2017), baseado no kernel do Linux. Apesar de ter sido desenvolvido para smartphones, hoje está presente em outras aplicações como tablets ou relógios. O Android é uma máquina virtual Java rodando sobre o kernel do Linux e que dá suporte para o desenvolvimento de aplicações Java através de um conjunto de bibliotecas e serviços (SCHULTE, 2016).

O Android surgiu em 2003, na cidade Palo Alto na Califórnia e foi desenvolvido por empresários no ramo da tecnologia, que fundaram a Android Inc. Os criadores desenvolveram um sistema móvel com uma interface simples e funcional baseada no kernel do Linux. Sendo que a ideia inicial era oferecer um sistema operacional gratuito para todas as pessoas (SCHULTE, 2016).

O Google adquiriu a empresa Android Inc. em 17 de agosto de 2005 e comercializou o software Android para as empresas fabricantes de celulares e tornou o Android, o sistema operacional mais utilizado do mundo.

Com o desenvolvimento do código do Android são lançadas novas versões, e cada nova versão contém melhorias e novidades que, não são aplicadas a versões anteriores fazendo com que as versões mais antigas deixem de obter suporte para alguns aplicativos.

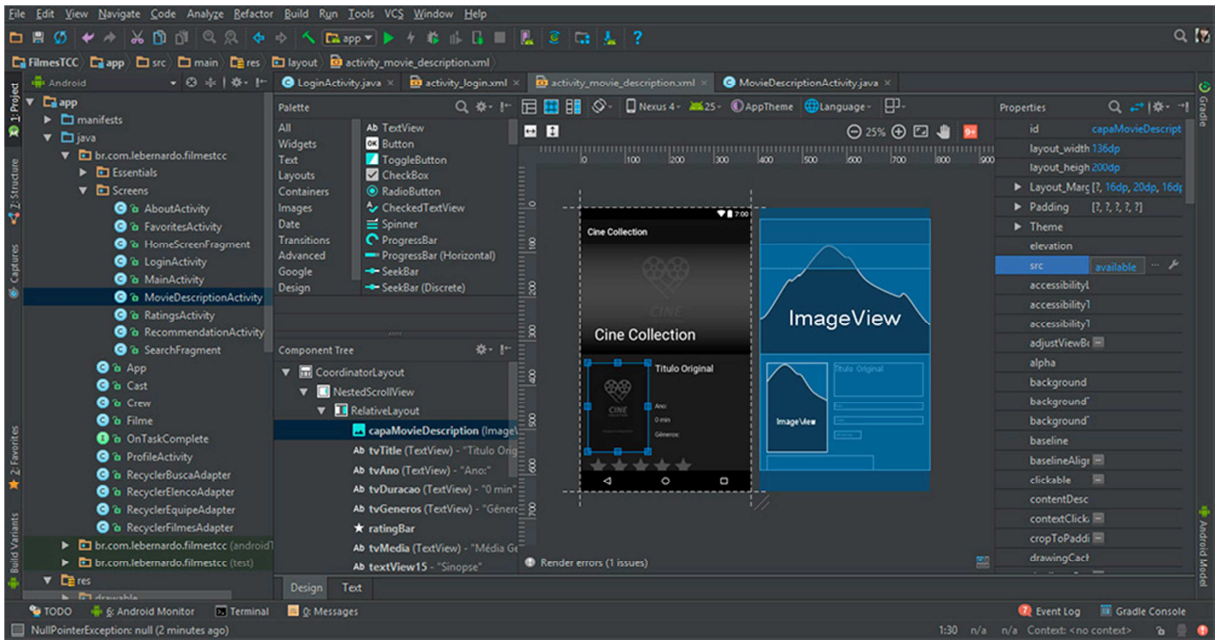
[17] The Movie Database. Disponível em: <https://www.themoviedb.org/?language=pt-BR>.

Sua primeira versão alpha foi lançada em 2007. As versões são desenvolvidas sob um codinome e lançadas em ordem alfabética: *Alpha, Beta, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow e Nougat* (ANDROID, 2017).

O aplicativo do projeto foi desenvolvido na versão *Jelly Bean* do Android 4.3, atuando em aproximadamente 76,9% do total de celulares ativos na Google Play Store.

O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android, baseado no InteliJ IDEA[18], que oferece um ambiente unificado para o desenvolvimento de aplicativos. No qual é possível desenvolver, fazer debugs, testes e interfaces para smartphones e tablets Android e dispositivos Android Wear, Android TV e Android Auto (ANDROID, 2017). A Figura 10 apresenta o ambiente de desenvolvimento Android Studio.

Figura 10 – Ambiente de desenvolvimento do Android Studio.



Fonte: Elaborado pelo autor

[18] Intellig IDEA é um JAVA IDE da empresa JetBrains. Disponível em: <https://www.jetbrains.com/idea>.

3.3.2.2 Linguagem Java

O Java é uma linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems, capaz de criar tanto aplicativos para desktop, aplicações comerciais, softwares robustos e aplicativos para web (JAVA, 2017).

Apesar de a sintaxe utilizada ser derivada do C++, ela é um modelo mais simples. Sua principal característica é ter seu código escrito todo dentro de classes e tudo é um objeto, com exceção de variáveis. O objetivo desta linguagem é que deveria ser simples e de fácil aprendizagem, não apenas para programadores experientes.

As aplicações em Java são executadas em qualquer plataforma que possua a Java Virtual Machine (JVM) instalada.

3.3.2.3 The Movie Database

The Movie Database (TMDB) é um banco de dados de filmes e séries gratuitos e de código aberto. Criado por Travis Bell em 2008, o TMDB é atualizado constantemente pela comunidade (TMDB, 2017).

Para utilizar a API TMDB é necessário fazer uma requisição de uma chave de API[19]. A comunidade do TMDB disponibiliza bibliotecas prontas em várias linguagens para o uso de desenvolvedores. No caso do aplicativo desenvolvido neste projeto não utilizaremos estas bibliotecas.

A requisição pelo TMDB é feita por meio de uma *Representational State Transfer* (REST), em português, Transferência de Estado Representacional, frequentemente aplicado à *web services* fornecendo APIs para acesso a um serviço qualquer na web. Um REST se comunica através do protocolo HTTP [20] para fazer as Requisições e, comumente, o retorno destas APIs utilizam notações como XML[21] ou JSON[22]. No caso do TMDB o retorno é no formato de JSON.

---

[19] API, do inglês, *Application Programming Interface* é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web.

[20] HTTP, do inglês, *Hypertext Transfer Protocol*, em português Protocolo de Transferência de Hipertexto, é um protocolo base para a comunicação de dados da web.

[21] XML, do inglês, *Extensible Markup Language*, É um formato de texto simples, muito flexível derivado de SGML (ISO 8879). Para saber mais acesse: <https://www.w3.org/XML/>

3.3.2.4 Firebase

O Firebase[23] é o serviço na nuvem do Google para desenvolvimento móvel e de aplicações web. Permitindo alguns serviços como a implementação do Google Analytics[24], autenticações no Facebook[25], Twitter[26], Github[27], autenticações personalizadas, banco de dados que permite a sincronização de dados em tempo real, entre outros.

3.3.2.5 SDK do Facebook

O SDK [28] do Facebook para Android permite integrar os serviços do Facebook no aplicativo de forma mais rápida. O SDK do Facebook contém:

- Login com Facebook: permite o usuário fazer a autenticação com as credenciais do Facebook;
- *Account Kit*: conecta pessoas usando o número de telefone ou e-mail;
- Diálogos de compartilhamento de envio: habilita o compartilhamento de conteúdo do aplicativo no Facebook;
- Eventos do Aplicativo: registra os eventos no aplicativo;
- API de Gráfico[29]: Lê e grava na API de Gráfico.

Para utilizar a SDK do Facebook é necessário se registrar como um desenvolvedor do Facebook pelo Facebook *Developers*[30]. O Facebook *Developers* é

---

[22] JSON, do inglês, *JavaScript Object Notation*, em português, Notação de Objetos JavaScript, é uma formatação para troca de dados. Para saber mais acesse: <http://www.json.org/json-pt.html>

[23] Firebase. Disponível em: <https://firebase.google.com>. Acesso em: 22/05/2017.

[24] Google Analytics é uma ferramenta gratuita do Google que permite analisar as estatísticas de visitação de um website. Disponível em: <https://analytics.google.com>. Acesso em: 22/05/2017.

[25] Facebook. Disponível em: <https://facebook.com>. Acesso em: 22/05/2017.

[26] Twitter é uma rede social que permite aos usuários enviar e receber atualizações em textos de até 140 caracteres. Disponível em: <https://twitter.com>. Acesso em: 22/05/2017.

[27] Github é uma plataforma de hospedagem de código para controle de versão e colaboração. Disponível em: <https://github.com>. Acesso em: 22/05/2017.

[28] SDK é conhecido também como "*devkit*", é um conjunto de ferramentas de desenvolvimento e códigos pré-gravados que podem ser usados pelos desenvolvedores para criar softwares. Os SDKs geralmente ajudam a reduzir a quantidade de esforço e tempo que seria necessário para os profissionais escreverem seus próprios códigos.

[29] API de Gráfico do Facebook é uma plataforma do Facebook para interagir com gráfico social do Facebook. Disponível em: <https://developers.facebook.com/docs/graph-api?locale=pt_BR>. Acesso em: 22/05/2017.

[30] Facebook Developers é o sistema do Facebook desenvolvido para desenvolvedores de aplicativos na plataforma do Facebook. Disponível em <https://developers.facebook.com>.

o site que ajuda os desenvolvedores a implementar, monitorar e criar soluções com os produtos oferecidos.

3.3.3 Cronograma

O cronograma do projeto documenta as atividades durante todo o desenvolvimento do projeto. As tarefas são dispostas sequencialmente por datas, cada tarefa possui uma data inicial e a data de término, além dos recursos usados. A Figura 11 apresenta o cronograma do projeto.

Figura 11 – Cronograma do projeto.

Id	Modo da Tarefa	Nome da Tarefa	% concluída	Início	Término	Predecessoras
1		**Desenvolvimento do Aplicativo**	**91%**	**Sex 10/03/17**	**Qua 31/05/17**	
2		**Escopo**	**100%**	**Sex 10/03/17**	**Ter 28/03/17**	
3		Definir objetivos	100%	Sex 10/03/17	Sáb 18/03/17	
4		Plano de requisitos	100%	Seg 20/03/17	Ter 28/03/17	3
5		**Planejamento e Modelagem**	**100%**	**Ter 28/03/17**	**Sex 14/04/17**	
6		Selecionar recursos	100%	Ter 28/03/17	Qui 30/03/17	
7		Criar caso de uso	100%	Ter 11/04/17	Qua 12/04/17	
8		Criar diagrama de classe	100%	Qua 12/04/17	Sex 14/04/17	
9		Definir riscos	100%	Qua 12/04/17	Sex 14/04/17	
10		Realizar um cronograma	100%	Qui 13/04/17	Sex 14/04/17	
11		**Design**	**100%**	**Seg 15/05/17**	**Sex 19/05/17**	
12		Desenvolver o logotipo	100%	Seg 15/05/17	Ter 16/05/17	10
13		Criar protótipo	100%	Ter 16/05/17	Qui 18/05/17	12
14		**Desenvolvimento**	**95%**	**Qua 05/04/17**	**Qua 31/05/17**	
15		Rever especificações funcionais	100%	Qua 05/04/17	Sex 07/04/17	
16		Criar telas do aplicativo	100%	Sex 07/04/17	Ter 11/04/17	15
17		Inserir Login pelo facebook	100%	Ter 11/04/17	Qui 13/04/17	16
18		Adicionar API do firebase ao projeto	100%	Qui 13/04/17	Ter 18/04/17	17
19		Guardar autenticação do usuário no firebase	100%	Ter 18/04/17	Sáb 22/04/17	18
20		Conectar API de filmes	100%	Qui 27/04/17	Ter 02/05/17	21
21		Listar filmes populares	100%	Seg 24/04/17	Qui 27/04/17	19
22		Apresentar informações do filme	100%	Qui 27/04/17	Sáb 29/04/17	21
23		Adicionar o elenco às informações do filme	100%	Seg 01/05/17	Qua 03/05/17	22
24		Adicionar a equipe técnica às informações do filme	100%	Qua 03/05/17	Sex 05/05/17	23
25		Favoritar filme	100%	Sex 05/05/17	Sex 12/05/17	24
26		Avaliar filme	100%	Sex 12/05/17	Sáb 20/05/17	25
27		Busca de filmes	100%	Seg 22/05/17	Ter 23/05/17	26
28		Página Sobre	80%	Ter 23/05/17	Qua 24/05/17	27
29		Página de Conteúdo	100%	Qua 10/05/17	Sex 12/05/17	
30		Adicionar Perfil de usuário	100%	Qui 11/05/17	Sáb 13/05/17	
31		Adicionar Banner	100%	Seg 22/05/17	Ter 23/05/17	
32		Recomendar Filmes	56%	Sex 19/05/17	Sex 26/05/17	
33		**Testes**	**0%**	**Sáb 20/05/17**	**Sex 26/05/17**	
34		Testes de Uso	0%	Seg 22/05/17	Sex 26/05/17	

Fonte: Elaborado pelo autor

3.4 Modelagem

Na fase da modelagem é feita a documentação do aplicativo, se tratam de diagramas que facilitam na compreensão do projeto de forma padronizada.

A documentação deste trabalho utilizará a linguagem de modelagem *Unified Modeling Language* [31] (UML) para modelar os casos de uso e o diagrama de classe.

3.4.1 Casos De Uso

Os diagramas de caso de uso descrevem um cenário de funcionalidades do ponto de vista do usuário, catalogando os requisitos funcionais do sistema. Dentro do diagrama são retratados os atores (representado pelos bonecos), as funcionalidades (representadas pelos balões com a ação escrita por dentro) e as relações (representadas pelas linhas).

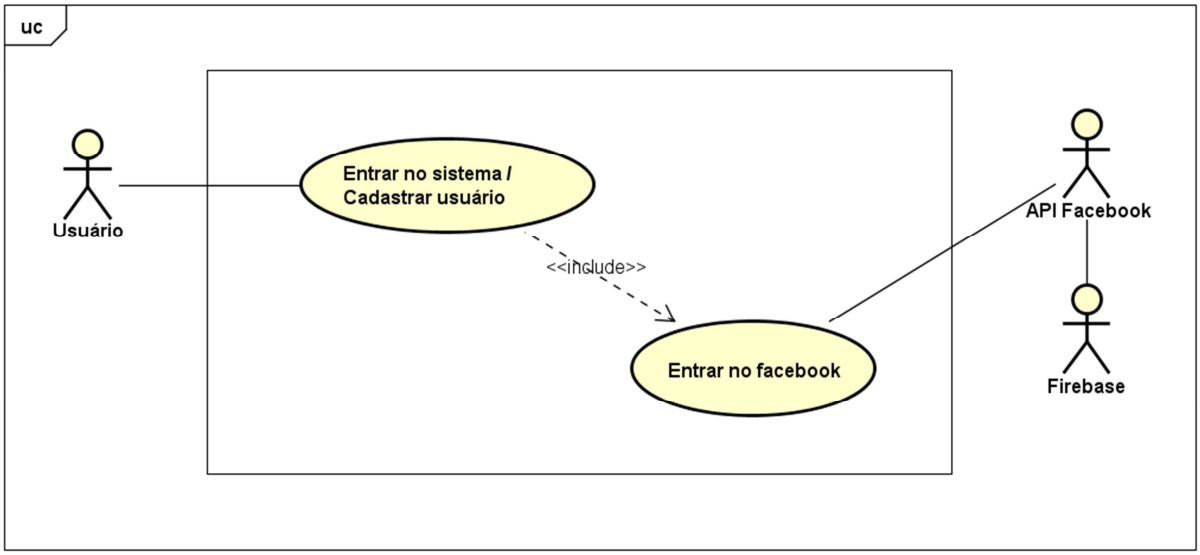
Os atores que interagem com o sistema são: o Usuário, API Facebook, Firebase, API TMDB e o Sistema de recomendação. O sistema é um caso de uso explícito e se trata do sistema em si em que os casos de uso acontecem.

- **Usuário** é o ator que representa os utilizadores deste aplicativo. Um ator pode, por exemplo, buscar filmes, avaliar, favoritar, entre outros.
- **API Facebook** representa o ator da API que permite a interação entre o aplicativo e o Facebook, por exemplo, o login com as credenciais da rede social.
- **Firebase** representa o banco de dados em tempo real, onde o sistema armazena as informações de usuários e de filmes.
- **API TMDB** representa a API de filmes que faz requisições e recebe os dados utilizando uma REST.
- **Sistema de Recomendação** representa a parte do sistema responsável por gerar recomendações ao usuário de modo automático.

A Figura 12 apresenta o caso de uso para a entrada do usuário no sistema.

[31] *Unified Modeling Language* ou Linguagem Unificada de Modelagem (UML) é uma linguagem padrão para modelagem e documentar os sistemas orientados a objetos.

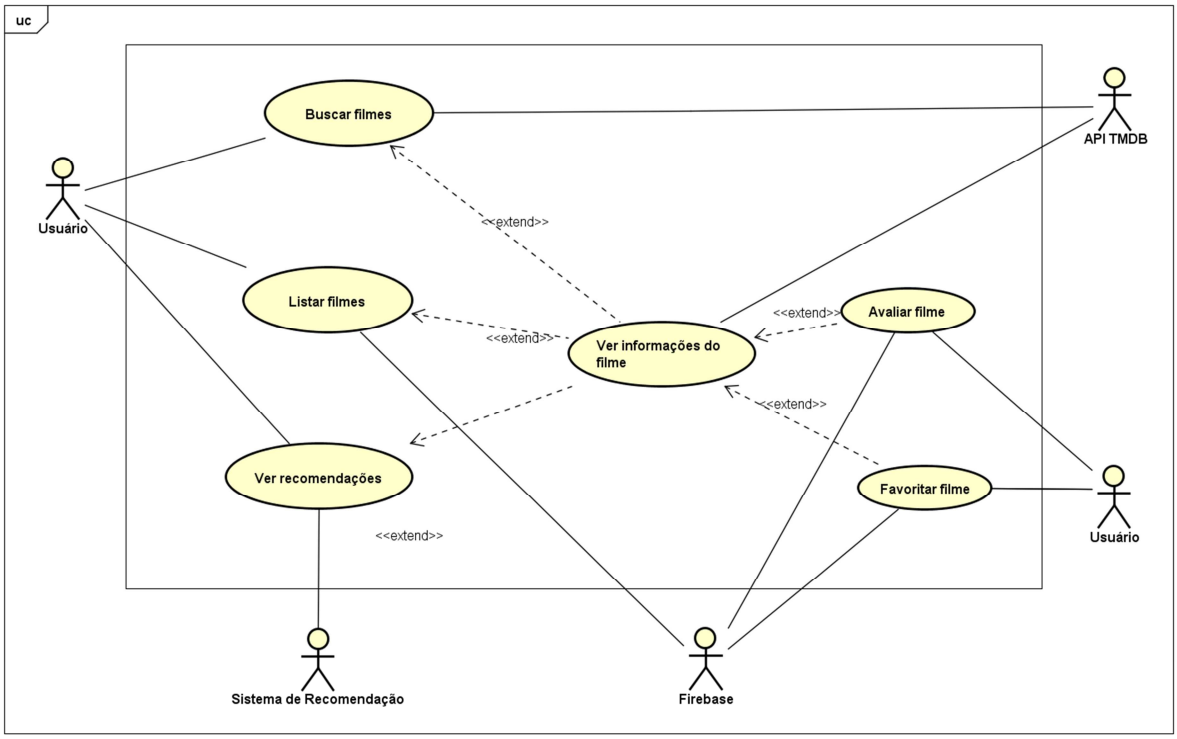
Figura 12 – Diagrama de caso de uso do login de usuário.



Fonte: Elaborado pelo autor

A Figura 13 apresenta o caso de uso que representa as funcionalidades relacionadas ao usuário após a entrada no sistema.

Figura 13 – Caso de uso de funcionalidades do aplicativo.



Fonte: Elaborado pelo autor

No subcapítulo 3.4.2 será apresentado à documentação dos casos de uso do projeto deste trabalho.

3.4.2 Documentação dos Casos de Uso

Cada funcionalidade dos diagramas de casos de uso será descrito do Quadro 5 ao Quadro 11. O caso de uso “Entrar no Sistema / Cadastrar Usuário” utiliza a API do Facebook para credenciar o login do usuário e seu retorno, o Firebase faz a autenticação do usuário por meio do e-mail. Caso o usuário exista, a base confirma o login e atualiza a última vez que o usuário se conectou. Se o usuário não existir, o Firebase o cria em sua base, registrando o e-mail, o tipo de provedor utilizado (neste caso o Facebook), a data em que foi criada a conta, a última vez o usuário que foi conectado e uma *User ID* (UID) gerada pelo Firebase. O Quadro 5 detalha o passo a passo da execução deste caso de uso.

Quadro 5 – Caso de uso “Entrar no Sistema / Cadastrar Usuário”.

Nome do caso de uso	Entrar no Sistema / Cadastrar Usuário
Atores envolvidos	Usuário, API Facebook, Firebase, Sistema.
Objetivo	Este caso de uso descreve os passos do login e / ou cadastro de um usuário no sistema
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O usuário clica em login com o Facebook	
	2. O sistema leva para a API do Facebook que se encarrega do processo de verificação de conta e retorna com as informações.
	3. Ao retornar com as informações do usuário, o Firebase autentica o usuário em sua base de dados.
	4. Após a autenticação, o sistema redireciona para a página inicial do aplicativo.

Validações	Para o login seja efetuado, o usuário deve entrar com seu usuário e senha do Facebook.

Fonte: Elaborado pelo autor

O caso de uso “Buscar Filmes” utiliza a API TMDB para encontrar filmes condizentes às informações que o usuário deseja. O sistema faz uma requisição a API TMDB com a informação desejada pelo usuário, a chave da API e opcionalmente, a língua que os filmes retornarão. A API checa a chave e retorna, em caso de sucesso, os resultados obtidos. O Quadro 6 apresenta o passo a passo da execução deste caso de uso.

Quadro 6 – Caso de uso “Buscar Filmes”.

Nome do caso de uso	Buscar Filmes
Atores envolvidos	Usuário, API TMDB, Sistema.
Objetivo	Este caso de uso descreve os passos da realizar uma busca de filmes
Prioridade de desenvolvimento	Alta
Ações do ator	**Ações do Sistema**
1. O usuário provem informações sobre o filme desejado	
	2. O sistema solicita a API TMDB, uma busca indicando as informações procuradas, a chave e a língua no qual a busca deve retornar.
	3. A API TMDB verifica a chave e retorna os resultados (filmes).
	4. O sistema recebe os dados e os manipula para serem visualizados pelo usuário em forma de lista.
Validações	Para que a busca seja realizada o cliente precisa informar algum valor.

Fonte: Elaborado pelo autor

O caso de uso “Ver Informações do Filme” é o caso de uso no qual o sistema faz o processamento para exibir as informações de um determinado filme. Uma requisição à API do TMDB é feita utilizando a chave da API, a ID do filme e a língua que o filme retornará. A API valida a chave e procura o filme pelo seu ID e, em caso

de sucesso, retorna as informações. O sistema executa outras duas requisições solicitando o elenco e a equipe técnica, por meio de seu ID. O sistema também faz requisições ao Firebase utilizando o UID e o ID do filme para checar se o usuário favoritou ou não e, se avaliou ou não o filme. O Quadro 7 apresenta o passo a passo do caso de uso “Ver Informações do Filme”.

Quadro 7 – Caso de uso “Ver Informações do Filme”.

Nome do caso de uso	Ver Informações do Filme
Atores envolvidos	Usuário, API TMDB, Sistema.
Objetivo	Este caso de uso descreve os passos para um usuário ver as informações detalhadas de um filme.
Prioridade de desenvolvimento	Alta
Ações do ator	**Ações do Sistema**
1. O usuário clica em uma miniatura do filme para ver informações detalhadas de um filme.	
	2. O sistema se conecta a API do TMDB e busca as informações do filme por meio do ID do filme solicitado, a chave e a API retorna as informações do filme.
	3. O sistema faz duas novas requisições pelo ID do filme para elenco e a equipe.
	4. O sistema faz uma requisição ao Firebase utilizando o id de autenticação do usuário do Firebase e o ID do filme e retorna as informações.
	5. O sistema exibe todas as informações do filme com o retorno da API.
	6. O sistema checa se o filme foi favoritado ou não e exibe para o usuário.
	7. O sistema checa se o filme foi avaliado e exibe para o usuário.

Fonte: Elaborado pelo autor

O caso de uso “Favoritar Filme” se trata na ação do usuário favoritar ou não favoritar um determinado filme. No momento em que o usuário faz esta requisição o

sistema checa se o filme já havia sido favoritado e caso sim, envia ao Firebase o UID, o ID e o valor do campo “favorito” como *false*, caso contrário enviaria o valor *true*. O Quadro 8 apresenta o passo a passo para caso de uso “Favoritar Filme”.

Quadro 8 – Caso de uso “Favoritar Filme”.

Nome do caso de uso	Favoritar Filme
Atores envolvidos	Usuário, Firebase, Sistema.
Objetivo	Este caso de uso descreve os passos de que um usuário favoritar um determinado filme
Prioridade de desenvolvimento	Média
Ações do ator	**Ações do Sistema**
1. O usuário entra na página do filme desejado.	
2. O usuário clica em “Favoritar”.	
	3. O sistema checa se o filme já tinha sido favoritado ou não. Se sim, guarda o valor *false*, senão o valor *true*.
	4. O sistema envia ao Firebase as informações do UID, ID do filme e o valor do favorito.

Fonte: Elaborado pelo autor

O caso de uso “Avaliar Filme” se trata na ação do usuário avaliar um determinado filme. O sistema envia ao Firebase o UID, o ID e um valor de 1 a 5 em função da avaliação do usuário. O Quadro 9 apresenta o passo a passo para caso de uso “Avaliar Filme”.

Quadro 9 – Caso de uso “Avaliar Filme”.

Nome do caso de uso	Avaliar Filme
Atores envolvidos	Usuário, Sistema, Firebase.
Objetivo	Este caso de uso descreve os passos de que um usuário favoritar um determinado filme
Prioridade de desenvolvimento	Média

Ações do ator	**Ações do Sistema**
1. O usuário entra na página do filme desejado.	
2. O usuário avalia o filme com estrelas de 1 a 5.	
	3. O sistema se conecta ao Firebase e envia as informações para o banco de dados.

Fonte: Elaborado pelo autor

O caso de uso “Listar Filmes” apresenta uma listagem de filmes com filtro de filmes avaliados ou favoritados pelo usuário. O Firebase após receber uma requisição do sistema, procura em sua base de dados, todos os filmes com o filtro requisitado e retorna a lista. O Quadro 10 apresenta o passo a passo o caso de uso “Listar Filmes”:

Quadro 10 – Caso de uso “Listar Filmes”.

Nome do caso de uso	Listar Filmes
Atores envolvidos	Usuário, Sistema, Firebase.
Objetivo	Este caso de uso descreve os passos de listagem de filmes
Prioridade de desenvolvimento	Alta
Ações do ator	**Ações do Sistema**
1. O usuário entra no seu perfil	
2. O usuário clica em “Ver filmes”	
	3. O sistema checa se existe conexão com a internet
	4. O sistema checa se o usuário está logado
	5. O sistema solicita os filmes avaliados do usuário no Firebase e o retorno é uma lista de filmes.

Fonte: Elaborado pelo autor

O caso de uso “Ver Recomendações” se trata do sistema mostrar uma listagem de filmes em que o usuário possa se interessar. O sistema busca os filmes que o usuário avaliou com maiores notas e retorna uma lista. O sistema faz uma busca no Firebase em todos os usuários que avaliaram cada filme da lista do usuário com boas notas e retorna uma lista de usuários. O SR por meio de cálculos descobre usuários similares. E, a partir de um usuário semelhante, busca os filmes que o usuário não tenha avaliado ainda. O quadro 11 apresenta o passo a passo deste caso de uso.

Quadro 11 – Caso de uso “Ver Recomendações”.

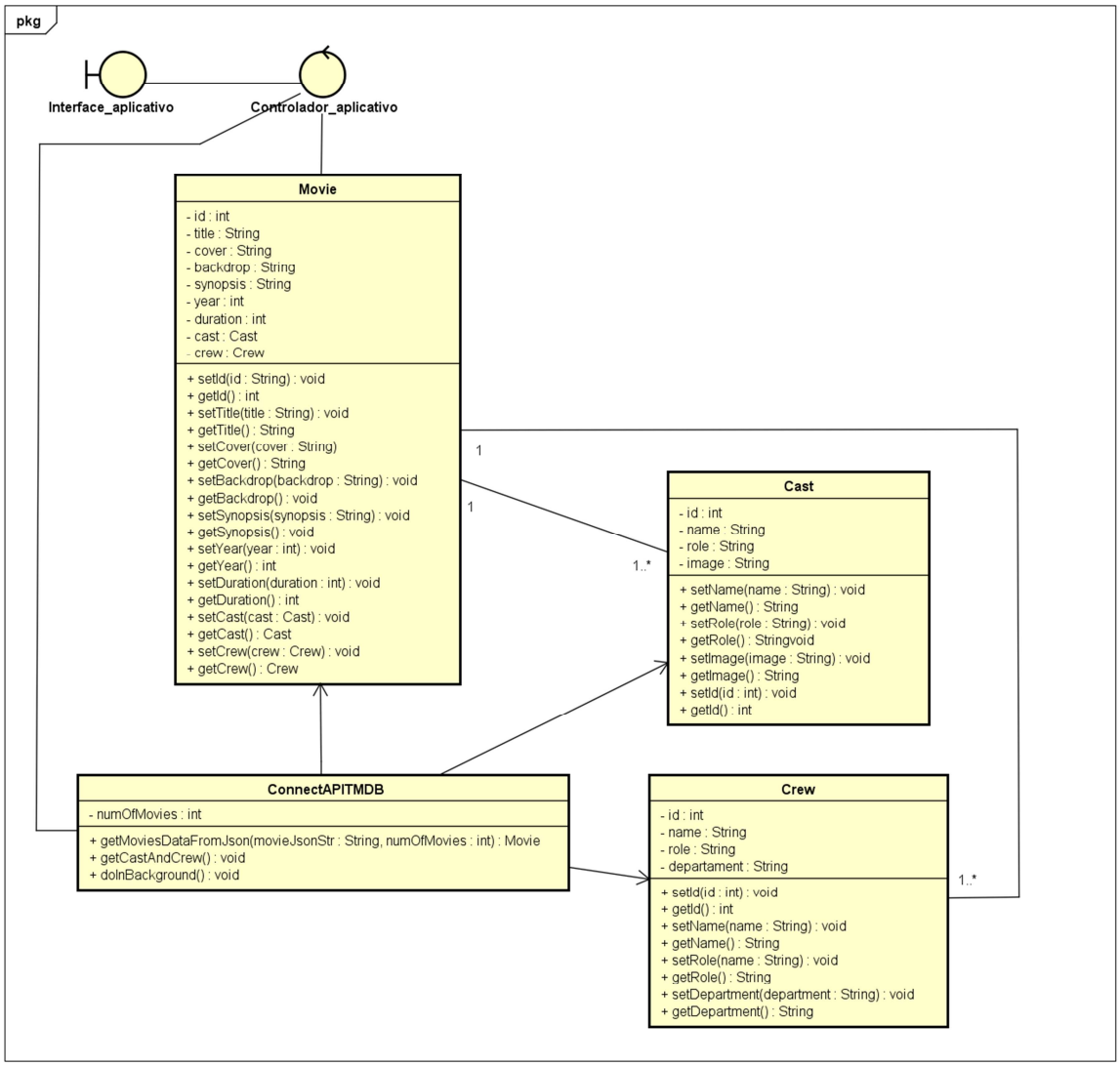
Nome do caso de uso	Ver Recomendações
Atores envolvidos	Usuário, Sistema, Firebase, Sistema de Recomendação.
Objetivo	Este caso de uso descreve os passos de o sistema fará para fornecer recomendação para um determinado usuário.
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O usuário entra na página de listagem de recomendações.	
	2. O sistema busca no Firebase todos os filmes que o usuário.
	3. O sistema procura usuários que possuem filmes avaliados.
	4. O SR faz cálculos para detectar o usuário mais parecido (chamado também de vizinho mais próximo), ou seja, que tem a lista de filmes mais parecida com determinado usuário.
	5. O SR busca os filmes que o usuário não tenha avaliado ainda na lista de filmes de seu vizinho mais próximo.
	6. O SR gera uma lista de recomendação ordenada por filmes com maior índice de similaridade e exibe os resultados.

Fonte: Elaborado pelo autor

3.4.3 Diagrama de Classe

O diagrama de classe é responsável por oferecer uma representação da estrutura e relações das classes como também as operações solicitadas pelos atores que servem de modelo para os objetos. Para entender melhor a estrutura funcional do projeto, a Figura 14 apresenta as principais classes do aplicativo **Cine Collection**.

Figura 14 – Diagrama de classe de filmes.



Fonte: Elaborado pelo autor

A classe *ConnectAPITMDB* é responsável por criar e gerenciar a conexão com a API. Possuindo dois métodos que são:

- ***doInBackground***: este método inicia a *Thread* [32] que acessa a API TMDB, retorna os dados como um objeto JSON e utiliza este objeto no método *getMovieDataFromJson*.
- ***getMovieDataFromJson***: retorna os dados de um filme específico ou de um grupo de filmes, de acordo com o requerimento feito pela interface.
- ***getCastAndCrew***: obtém os dados do Elenco e da Equipe técnica de um filme em específico.

O método *getCastAndCrew* só é acessado pelo método *getMovieDataFromJson* e quando a interface necessita de dados específicos de um filme.

A classe *Movie* é responsável por armazenar todos os dados de um filme. Além disso, fornece acesso aos seus atributos por meio de métodos *getters* e *setters*. Dentro os dados do filme estão as classes *Cast e Crew*.

A classe *Cast* é encarregada de armazenar os dados do elenco de um filme, já a classe *Crew* guarda os dados de equipe técnica de um filme. Ambas possuem os métodos assessores *getters* e *setters*.

Outras classes que compõem o aplicativo são as classes controladoras, ou seja, classes denominadas como *Activity* ou *Fragments*. A Figura 15 apresenta um conjunto de classes de visualização e controle utilizados no projeto.

As classes controladoras têm como objetivo fazer a ligação entre o modelo (Firebase e APIs) e a View (Layouts em XML), como também controla o que está sendo visualizado pelo usuário.

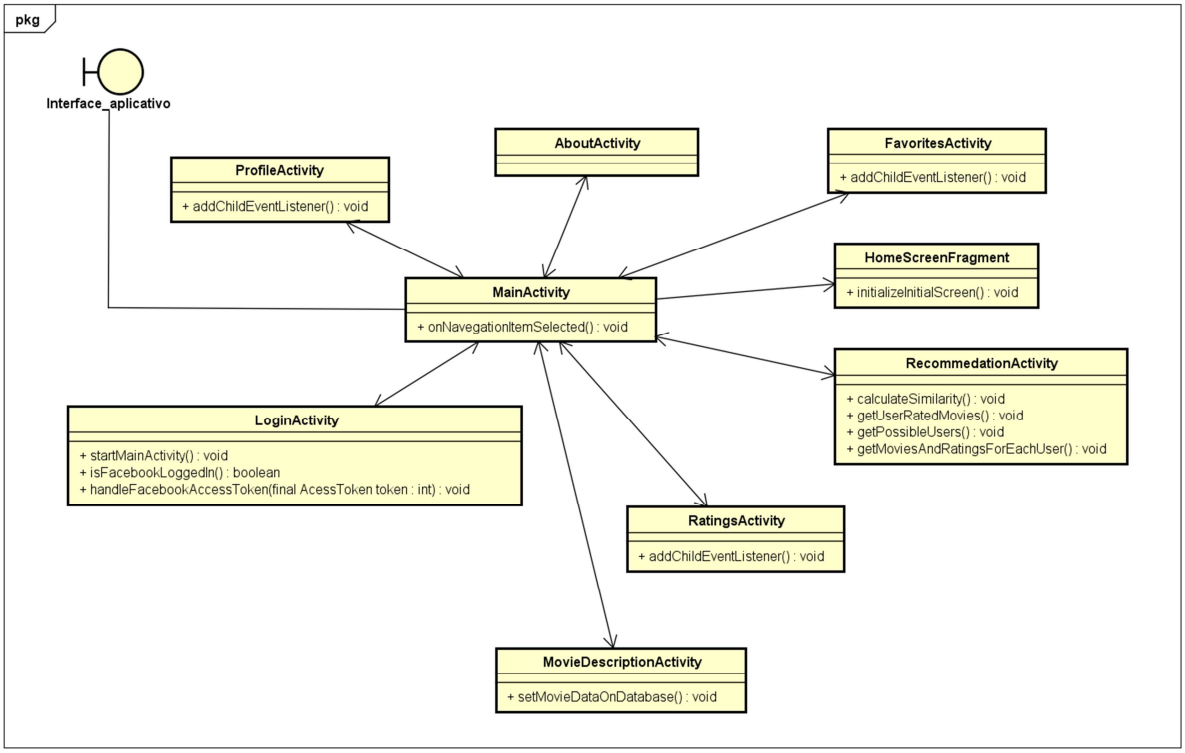
A classe *LoginActivity* possui os seguintes métodos:

- ***isFacebookLoggedIn***: este método checa se objeto *AccessToken* do Facebook não é um valor nulo, caso não veja nulo, retorna *true*.
- ***handleFacebookAccessToken***: este método é responsável por pegar o *token* da pessoa que fez o login do Facebook pelo aplicativo e checar se já existe ou não sua conta no banco de dados do Firebase, se não existir, ele adiciona o usuário a base e faz o login, caso já exista o usuário, o banco loga com a conta do usuário.

[32] *Thread* ou Encadeamento de execução é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas em paralelo.

- ***startMainActivity***: este método inicia a MainActivity que leva para a página principal do aplicativo.

Figura 15 – Diagrama de classe de visualização e controle.



Fonte: Elaborado pelo autor

A classe *MainActivity* possui como métodos:

- ***onNavigationItemSelected***: implementado através de um *NavigationView,* este método tem como objetivo fornecer a navegação entre as páginas do aplicativo.

A classe *AboutActivity* retorna para a view as informações sobre o desenvolvedor.

A classe *MovieDescriptionActivity* é responsável por carregar todos os dados de um determinado filme e possibilitar o usuário Favoritar ou Avaliar um filme. Possui como método:

- ***setMovieDataOnDatabase***: este método persiste o filme no banco de dados Firebase para que possa utilizado em outras partes do aplicativo com o intuito de agilizar a busca dos filmes.

- ***OnTaskCompleted**:* é executada assim que a classe *ConnectAPITMDB* termina sua execução.

A classe *ProfileActivity* é responsável por apresentar uma pequena mostra de filmes que o usuário já avaliou e já favoritou.

As classes *FavoritesActivity* e *RatingActivity* retornam uma listagem de filmes a partir de uma requisição ao Firebase.

A classe *RecommendationActivity* apresenta ao usuário as recomendações geradas pelo Sistema de Recomendação. O método *getUserRatedMovies()* é responsável por buscar todos os filmes bem avaliados do usuário e a partir disto, é chamado o método *getPossibleUsers(),* que busca os usuários que também avaliaram os filmes que o usuário alvo avaliou e chama o método *getMoviesAndRatingsForEachUser()* que monta a matriz de $usuários$ x $itens$ com suas respectivas avaliações e, após isto, o método *calculateSimilarity()* é chamado no qual é calculado a similaridade entre o usuário alvo e os outros usuários, encontrado os usuários vizinhos, é encontrado os filmes que os usuários vizinhos avaliaram com boas notas e que o usuário alvo ainda não tenha avaliado e é recomendado estes itens.

3.5 Desenvolvimento

Neste capítulo será descrito o processo de desenvolvimento do projeto, onde serão apresentadas detalhadamente as atividades de cada fase.

O desenvolvimento inicial teve como principal objetivo codificar as funcionalidades mais básicas do aplicativo, como: a criação das interfaces de usuário, a navegação do aplicativo, o login do Facebook, a autenticação com Firebase e a conexão com a API responsável por popular o aplicativo com dados de filmes.

3.5.1 Interfaces de Usuário

A Interface de Usuário (UI), do inglês *User Interface*, é o espaço onde a interação entre humanos e máquinas ocorre. No Android, as UIs são criadas no formato XML e são controladas por classes Java. Na Figura 16 está um exemplo de um botão chamando o clique de outro botão.

Na Figura 16 analisa-se que o código na classe Java faz referência ao botão criado no arquivo de formato XML, que manipula em seu evento de clique, o evento de clique em outro botão. Todas as telas do aplicativo são encontradas no Apêndice C.

Figura 16 – Exemplo de criação de elemento de interface XML e acesso via classe.

```
<Button
    android:layout_marginTop="38dp"
    android:id="@+id/login_button"
    android:background="@drawable/btn_login_facebook"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true"
    android:layout_height="46dp"
    android:layout_width="290dp" />
```

```
Button btn;
LoginButton loginButton;
btn = (Button) findViewById(R.id.login_button);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        loginButton.performClick();
    }
});
```

Fonte: Elaborado pelo autor

3.5.2 Navegação do aplicativo

A navegação do aplicativo foi implementada a partir do componente *Navigation Drawer* (menu lateral). O *Navigation Drawer* é uma espécie de menu no qual exibe as principais opções da navegação do aplicativo na borda esquerda da tela. Durante a maior parte do tempo, fica oculta, mas é revelada quando o usuário

desliza o dedo a partir da borda esquerda da tela, ou, na barra de ações, quando o usuário toca no ícone do componente.

A utilização do componente necessita primeiramente de um arquivo XML que inclua o componente *DrawerLayout*, com isto pronto, o layout é utilizado na *Activity* que irá implementá-lo. A Figura 17 ilustra o momento que ocorre a criação do *NavigationDrawer* em conjunto com seus componentes de controle.

Figura 17 – Exemplo da criação do *NavigationDrawer*.

```
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
drawer.setDrawerListener(toggle);
toggle.syncState();

final NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);
```

Fonte: Elaborado pelo autor

O objeto *ActionBarDrawerToggle* controla o estado do componente como “aberto” ou “fechado” e define o *Listener* a ser utilizado para o controle dos cliques em seus itens.

3.5.3 Login do Facebook com autenticação do Firebase

A partir de tela de abertura do aplicativo é possível acessar a funcionalidade de login com o Facebook e, com isso, entrar na tela inicial do aplicativo. Os passos para a configuração do login com o Facebook estão descritos no apêndice D.

O Firebase permite a autenticação de usuários em diversas plataformas, incluindo o Facebook. Para a utilização da mesma, foi habilitada a autenticação do Facebook no Firebase. No projeto do aplicativo, a autenticação é controlada por um objeto *FirebaseAuth*, com este objeto foi utilizado o método estático *signInWithCredential()* com a credencial fornecida pela API do Facebook. A Figura 18 apresenta o método utilizado para adicionar e fazer a autenticação do usuário no banco.

Os passos para integração do aplicativo com Firebase estão descritos no Apêndice E. Os passos para desenvolver a autenticação do Firebase utilizando o provedor do Facebook se encontram no Apêndice F.

Figura 18 – Autenticação do usuário pelo Firebase.

```
private void handleFacebookAccessToken(AccessToken token) {
    Log.d("aqui está o token", "handleFacebookAccessToken:" + token);

    AuthCredential credential = FacebookAuthProvider.getCredential(token.getToken());
    mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    Log.d("signincomplete", "signInWithCredential:onComplete:" + task.isSuccessful());

                    if (!task.isSuccessful()) {
                        Log.w("sign in", "signInWithCredential", task.getException());
                        Toast.makeText(MainActivity.this, "Authentication failed.",
                                Toast.LENGTH_SHORT).show();
                    }
                    else {
                        String uid = mAuth.getCurrentUser().getUid();

                        Toast.makeText(MainActivity.this, "Logado!!!!!!!!!!!!" , Toast.LENGTH_SHORT).show();
                    }

                }
            });
}
```

Fonte: Elaborado pelo autor

3.5.3 Trabalhando com a API TMDB

Antes de dar início ao sistema de recomendação é fundamental ter uma base de dados de filmes para que os usuários possam avaliar os filmes e futuramente, receber recomendações. Para tornar isso possível, foi escolhido a API TMDB, que consiste em um banco de dados de filmes e séries. Esta base de dados pode ser acessada por meio da API, o passo a passo para a conexão da mesma se encontra no Apêndice G.

Quando um usuário novo entra no aplicativo, este não possui recomendações, pois, não há como o SR entender quais são as suas preferências, se o usuário é novo, o sistema apenas exibe uma lista de filmes populares. Esta lista é populada pela API TMDB.

Após o retorno dos filmes dado pela classe *ConnectApiTMDB*, o método *onTaskCompleted()* é executado usando um *array* de filmes montado pela classe. Com este *array* é possível popular uma *RecyclerView* para exibir os filmes na tela. A

Figura 19 mostra o *RecyclerAdapter* sendo criado com o *array* de filmes e o mesmo sendo inserido na *RecyclerView*.

Figura 19 – Criação da *RecycleView*, do *LayoutManager* e do *Adapter*.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_home_screen, container, false);
    // Inflate the layout for this fragment

    progressDialog = new ProgressDialog(getActivity());
    recyclerView = (RecyclerView) view.findViewById(R.id.recyclerViewFilmes);

    RecyclerView.LayoutManager layoutManager = new GridLayoutManager(getContext(), 3);
    recyclerView.setLayoutManager(layoutManager);
    recyclerView.setHasFixedSize(true);
```

```
@Override
public void OnTaskCompleted(ArrayList<Filme> result)
{
    filmes = result;

    RecyclerFilmesAdapter recyclerFilmesAdapter = new RecyclerFilmesAdapter(getContext(), filmes);

    recyclerView.setAdapter(recyclerFilmesAdapter);
}
```

Fonte: Elaborado pelo autor

O *Adapter* utilizado é uma customização do *Adapter* padrão da *RecyclerView*. Suas principais diferenças se encontram no Layout do item utilizado e na construção do mesmo, utilizando o *array* de filmes para definir o título a ser mostrado, a capa e qual ID passar no momento do clique da miniatura para a montagem da *Activity* de descrição do filme.

O *LayoutManager* define o modo em que os itens serão exibidos na *RecyclerView* e sua orientação (horizontal e vertical).

Após a implementação das funcionalidades fundamentais do aplicativo, foi codificada a tela de informações detalhadas do filme, como também as telas de listagem de avaliações do usuário, listagem de filmes favoritos, tela de perfil e tela de Sobre.

3.5.4 Codificação da tela de informações detalhadas do filme

A tela de informações detalhadas do filme recebe um ID do filme a ser exibido, este ID é usado para executar a busca na API utilizando a classe *ConnectApiTMDB*.

A *Activity* por sua vez faz três buscas na API: a primeira retorna os dados específicos do filme, a segunda retorna os dados sobre a equipe técnica e a terceira retorna os dados sobre o elenco do filme.

No retorno da API são criados *Listeners* que, utilizando uma referencia do ID do filme e do UID no banco, retornam qual o valor das variáveis favorito e nota no banco.

Estas variáveis são definidas de acordo com a interação do usuário com a *Activity*, caso ele dê uma nota ao filme, o banco é chamado, e é criada uma referência a essa variável. A Figura 20 expõe o código utilizado para checar as informações de filmes favoritos e avaliações.

Figura 20 – Checagem dos dados pelo Firebase.

```
ratingBar.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener() {
    @Override
    public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {
        if(rating != 0){
            moviesRatingRef.setValue(rating);
            userMovieRef.child("nota").setValue(rating);
        }else{
            moviesRatingRef.removeValue();
            userMovieRef.child("nota").removeValue();
        }
    }
});
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    if(item.getItemId() == R.id.movie_description_favoritar) {
        if (isFavoritado) {
            item.setIcon(R.drawable.ic_action_favorite);
            isFavoritado = false;
            userMovieRef.child("favorito").setValue(isFavoritado);
            moviesFavRef.removeValue();
        } else {
            item.setIcon(R.drawable.ic_action_unfavorite);
            isFavoritado = true;
            userMovieRef.child("favorito").setValue(isFavoritado);
            moviesFavRef.setValue(isFavoritado);
        }
    }

    return super.onOptionsItemSelected(item);
}
```

Fonte: Elaborado pelo autor

As telas de listagem de filmes favoritos e filmes avaliados são feitas utilizando os dados do Firebase, a partir das avaliações da tela de informações detalhadas do filme. Já na tela de perfil do usuário encontra-se um resumo das últimas avaliações que o usuário executou.

A última etapa do desenvolvimento e a mais importante é o desenvolvimento das recomendações personalizadas para cada usuário.

3.5.5 Codificação do Sistema de Recomendação

Para desempenhar a tarefa do Sistema de Recomendação de filmes foi escolhida a abordagem de Filtragem Colaborativa Usuário-Usuário, utilizando o cálculo de Correlação de Pearson para encontrar os usuários vizinhos por ser a abordagem mais utilizada e mais simples de ser aplicada. O Sistema de Recomendação do aplicativo foi construído seguindo as etapas de criação da abordagem colaborativa do Subcapítulo 2.2.2.1.

3.5.5.1 Representação

A primeira fase de um Sistema de Recomendação é a representação do conteúdo, ou seja, é necessário montar a matriz de filmes de usuários x filmes.

Para isto, primeiramente, foi realizada uma busca aos filmes avaliados do usuário logado, ou seja, o usuário alvo (a) e foi adicionado em um *array* de notas por filme utilizando um *SparseArray*[33]. Após isto, foi efetuada uma chamada ao Firebase que retorna todos os outros usuários e, a cada usuário (b) é feita uma busca no *array* de filmes do usuário a para checar se o usuário b avaliou algum filme em comum com o usuário a, caso tenha avaliado, este é adicionado ao *SparseArray*. Caso o usuário b não tenha avaliado um determinado filme é adicionado uma nota 0 à sua avaliação e, caso o usuário b tenha avaliado um filme que o usuário a não tenha avaliado, é adicionado uma nota 0 ao usuário a.

[33] *SparseArray* é uma espécie de *array* de objetos com chaves utilizando um tipo primitivo (*integer,* no caso). Este tipo de *array* mantêm os dados em uma matriz e utiliza uma pesquisa binária para encontrar as chaves. Além disso, é mais eficiente em termos de memória do que usar um *Hashmap.* Para saber mais acesse: <https://developer.android.com/reference/android/util/SparseArray.html>.

Os usuários também são adicionados em um *ArrayList* de *availableUsers*, no qual, há a ordem dos usuários atribuídos na mesma ordem de suas notas dos filmes no *SparseArray*.

A Figura 21 apresenta a codificação da representação dos usuários do Sistema de Recomendação.

Figura 21 – Codificação da representação da matriz usuário x filme.

```
SharedPreferences sharedPrefs;
DatabaseReference mRootRef;
DatabaseReference userRef;
ArrayList<String> movies = new ArrayList<>();
String uid;
HashMap<String, Integer> userMovies;
SparseArray<ArrayList<Float>> ratingsPerUser = new SparseArray<>();
ArrayList<String> availableUsers = new ArrayList<>();

boolean lastMovie = false;

int numOfMovies = 0;
```

```
if (postSnapshot.hasChild("stars")) {
    DataSnapshot child = postSnapshot.child("stars");
    if(ratingsPerUser.get(id) != null) {
        ArrayList<Float> currentRating = ratingsPerUser.get(id);
        if(currentRating.size() == userUniqueIdentifier) {
            currentRating.add(child.getValue(Float.class));
        }else{
            int zeroOffset = userUniqueIdentifier - currentRating.size();
            for(int i = 0; i < zeroOffset; i++){
                currentRating.add((float) 0);
            }
            currentRating.add(child.getValue(Float.class));
        }
        ratingsPerUser.put(id,currentRating);
    }else{
        ArrayList<Float> movieArray = new ArrayList<>();
        for(int i = 0; i < userUniqueIdentifier; i++){
            movieArray.add((float) 0);
        }
        movieArray.add(child.getValue(Float.class));
        ratingsPerUser.put(id, movieArray);
    }
}
```

Fonte: Elaborado pelo autor

3.5.5.2 Descobrir vizinhança

A segunda etapa são calculados os usuários vizinhos. Tendo em vista a representação de usuários x filmes e nota, é executado o cálculo de similaridade utilizando a Correlação de Pearson, na qual se calcula a similaridade do usuário a para cada usuário b. Para complementar o entendimento, a explicação do cálculo da

Correlação de Pearson está presente no Subcapítulo 2.2.2.3 e, um exemplo do cálculo está presente no Apêndice A.

Para cada usuário disponível no *ArrayList availableUsers*, é feito o cálculo de similaridade. Primeiramente, é calculada a média das notas do usuário a e do usuário b. Após este cálculo, é iniciado uma iteração que passa por todos os filmes do *SparseArray* para montar o divisor e o dividendo do cálculo. Ao término do cálculo o resultado é adicionado a um *ArrayList similarityResults* que será utilizado para determinar quais usuários são os vizinhos mais próximos.

A Figura 22 mostra a codificação do cálculo de Correlação de Pearson do Sistema de Recomendação.

Após os cálculos se tem um *ArrayList* de todos os usuários similares com o valor de sua similaridade e, a partir de um algoritmo de *quickSort* foi ordenado por ordem decrescente os usuários similares, já que, 1 indica similaridade total e -1 dissimilaridade total.

Figura 22 – Cálculo de Correlação de Pearson.

```
private void calculateSimilarity(){
    float loggedUserTotal = 0;
    float currentUserTotal = 0;
    int numberOfMovies = 0;
    ArrayList<Float> similarityResults = new ArrayList<>();

    for(int currentUser = 1; currentUser <= availableUsers.size(); currentUser++){
        for(int currentMovieIndex = 0; currentMovieIndex < ratingsPerUser.size(); currentMovieIndex++){
            numberOfMovies++;
            ArrayList<Float> currentRatings = ratingsPerUser.get(ratingsPerUser.keyAt(currentMovieIndex));
            loggedUserTotal += currentRatings.get(0);

            if(currentUser < currentRatings.size()) {
                currentUserTotal += currentRatings.get(currentUser);
            }
        }
        float loggedUserAverage =  loggedUserTotal / numberOfMovies;
        float currentUserAverage = currentUserTotal / numberOfMovies;
        float dividendResult = 0;
        float loggedUserDivisorResult = 0;
        float currentUserDivisorResult = 0;

        for(int currentMovieIndex = 0; currentMovieIndex < ratingsPerUser.size(); currentMovieIndex++){
            ArrayList<Float> currentRatings = ratingsPerUser.get(ratingsPerUser.keyAt(currentMovieIndex));

            float notaUser1 = currentRatings.get(0);;
            float notaUser2 = 0;
            if(currentUser < currentRatings.size()) {
                notaUser2 = currentRatings.get(currentUser);
            }

            dividendResult += ((notaUser1 - loggedUserAverage) * (notaUser2 - currentUserAverage));

            loggedUserDivisorResult += Math.pow((notaUser1 - loggedUserAverage), 2);
            currentUserDivisorResult += Math.pow((notaUser2 - loggedUserAverage), 2);
        }

        similarityResults.add((float) (dividendResult / (Math.sqrt(loggedUserDivisorResult) * Math.sqrt(currentUserDivisorResult)

    }
}
```

Fonte: Elaborado pelo autor

3.5.5.3 Selecionar itens recomendáveis

O terceiro passo é selecionar os itens recomendáveis. Para isto, no *ArrayList de availableUsers* são utilizados os cinco usuários mais similares. Dentro destes usuários é feita uma nova comparação de filmes em relação ao usuário a para determinar os filmes que serão recomendados. Sendo que, os filmes ideais para recomendação são os filmes com média maior que 3.5 e, que, o usuário a ainda não tenha avaliado. Em base a estes critérios, é montado uma lista de filmes recomendáveis.

3.5.5.4 Recomendar

O último passo é recomendar filmes, ou seja, escolher uma abordagem atrativa para o usuário e exibir a lista de filmes. A abordagem para chamar a atenção do usuário pode ser feita com textos, como por exemplo: “Usuários que assistiram X filmes também assistiram Y filmes”, “Recomendações para você”, “Já que você assistiu a X filme”, entre outras.

3.6 Avaliação

A avaliação se consistiu em verificar se o aplicativo é funcional em todos os aspectos, apresentando assim “qualidade”. Existem diversas definições sobre a qualidade de software propostas na literatura, pois qualidade é um termo que pode ter diferentes interpretações. Segundo Pressman (2007):

> “Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais”.

Definição de Peters (2001):

> “Qualidade de software é avaliada em termos de atributos de alto nível chamados fatores, que são medidos em relação a atributos de baixo nível chamados de critérios”.

De acordo com a ISO 9126 (1994):

> “A qualidade é a totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer às necessidades declaradas ou envolvidas”.

As definições enfatizam três aspectos importantes:

1. Os requisitos de software são a base a partir da qual a qualidade é medida.
2. Padrões especificados definem um conjunto de critérios de desenvolvimento que orientam a maneira pelo trabalho de engenharia.
3. Existe um conjunto de requisitos implícitos que frequentemente não são mencionados na especificação.

A ISO 9126 representa a atual padronização mundial para a qualidade de software e do produto, a norma propõe que um software para ter qualidade deve possuir as seguintes características: Funcionalidade (i), Confiabilidade (ii), Usabilidade (iii), Eficiência (iv), Manutenibilidade (v) e Portabilidade (vi).

i. São as funções que justificam a existência do produto satisfazendo as necessidades explícitas e implícitas.
ii. Determina a capacidade de o sistema lidar com eventos inesperados.
iii. Capacidade do software de manter seu nível de desempenho sob condições estabelecidas durante um período de tempo estabelecido. Ser fácil de usar.
iv. Analisa o nível de desempenho do software e a quantidade de recursos utilizados.
v. Capacidade de manutenção no software.
vi. Analisa a execução do software em várias plataformas.

O software desenvolvido neste projeto foi validado sob a norma ISO 9126, possuindo as seguintes características:

- Funcionalidade: o software possui um sistema recomendador de filmes baseado em filtragem colaborativa, de acordo com a proposta do aplicativo.
- Confiabilidade: toda vez que ocorre um erro inesperado, o software mostra o erro que ocorreu informando o seu motivo.
- Usabilidade: o software possui um sistema intuitivo e fácil de ser utilizado, pois este aplicativo segue os padrões estabelecidos pelo Material Design do Google.
- Eficiência: o software se mantém estável durante toda sua utilização, quando há um processamento demorado a ser carregado, o software informa que está carregando o conteúdo.
- Manutenibilidade: o aplicativo pode ser facilmente atualizado por atualizações enviadas a loja da Google Play Store. Quando o aplicativo está em uma nova versão na loja, o usuário é notificado de atualizações do software.
- Portabilidade: foi priorizado a renderização do software em smartphones, sendo compatível com 76,9% dos smartphones Android utilizados no mundo (ANDROID STUDIO, 2017).

O aplicativo móvel desenvolvido atingiu seus objetivos em fornecer recomendações de filmes com base nos filmes que o usuário forneceu boas avaliações, baseado em filtragem colaborativa, o sistema, através de seus vizinhos mais próximos, estudados no capítulo 2.2.2, busca fornecer recomendações.

4 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo final, aplicar um estudo sobre sistemas de recomendação e suas técnicas mais conhecidas. Para consolidar esse estudo foi realizado o desenvolvimento do aplicativo Cine Collection utilizando a linguagem Java para Android no qual foi implementado um Sistema de Recomendação baseada em filtragem colaborativa e garantiu os resultados propostos.

O trabalho inicialmente explorou o que é um Sistema de Recomendação e trouxe o conceito das técnicas de filtragem baseada em conteúdo, filtragem colaborativa e filtragem híbrida. Foi escolhido à filtragem colaborativa como sistema recomendador para a recomendação de filmes, pela facilidade de gerar as informações necessárias, pelo fato de, poder existir uma comunidade colaborativa.

Na sequência, apresentou-se todo o processo de desenvolvimento do aplicativo, ministrando todo o planejamento desde a escolha da metodologia de desenvolvimento cascara, a escolha das tecnologias e das ferramentas utilizadas, plano de requisitos, diagramas e por fim o desenvolvimento, no qual foi desenvolvido as partes fundamentais do software.

No desenvolvimento do aplicativo, houveram algumas dificuldades dado, principalmente, pela curva de aprendizado de algumas ferramentas utilizadas. Os principais pontos foram à inexperiência com bancos de dados não relacionais e, também, ao modelo de desenvolvimento de aplicativos Android.

Contudo, o desenvolvimento do aplicativo ainda foi facilmente implementado, graças a documentação do Android e a outros pequenos projetos feitos no passado, que serviram de base para implementação de várias partes do código como, as requisições da API do Facebook, as requisições utilizando uma REST API, entre outras.

A implementação ainda pode ser melhorada em alguns pontos, como, a conexão com o banco de dados do Firebase, o cálculo do sistema de recomendação e a organização geral do projeto.

A interface do usuário foi construída a fim de manter a experiência do mesmo concisa, seguindo os padrões da plataforma Android para facilitar a navegação e manter a familiaridade do usuário em relação a outros aplicativos do gênero. O aplicativo é simples de ser utilizado e pode ser utilizado por usuários de qualquer nível de experiência, mesmo sendo focado para o público jovem.

O aplicativo cumpre o que promete, gerando recomendações de novos filmes aos usuários, porém, para ser funcional necessita de uma base de usuários maior para ser mais efetivo.

Como possíveis trabalhos futuros, pode-se apontar: um sistema recomendador de filmes com base no humor do usuário ou em bases comportamentais; o desenvolvimento do aplicativo também para iPhone, oferecendo a mesma estrutura que da plataforma Android, até mesmo, um sistema com recomendação híbrida para suprir as fraquezas de ambas às técnicas de recomendação.

REFERÊNCIAS

AAS, K.; EIKVIL, L. **Text categorisation**: A survey. Technical report, Norwegian Computing Center, 1999.

ADOMAVICIUS, G. *et al*. **Incorporating contextual information in recommender systems using a multidimensional approach**. ACM Transactions on Information Systems, 2005. p. 103-145.

AMAZON PRIME VIDEO. Disponível em: <https://www.primevideo.com>. Acesso em: 22 maio 2017.

ANDROID. Disponível em: <https://www.android.com/intl/pt-BR_br>. Acesso em: 07 de mar. 2017.

ANDROID STUDIO. Disponível em: <https://developer.android.com/studio>. Acesso em: 09 de mar. 2017.

ARAÚJO, T.C. **AppRecommender: um recomendados de aplicativo GNU/Linux**. 2011. 104 p. Tese (Mestrado em Ciências) - Instituto de Matemática e Estatística, Universidade de São Paulo, 2011. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-28092012-150655/publico/thesis_tassia_corrigida.pdf>. Acesso em: 30 abr. 2017.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 9126: Engenharia de software – Qualidade de Produto**, 1994.

BAHARUDIN, B.; LEE, L.; KHAN, K. **A review of machine learning algorithms for textdocuments classification**. Journal of Advances in Information Technology, 2010. p. 4-20.

BOGERS, T. **Recommender Systems for Social Bookmarking**. Tese (PhD), Tilburg University, 2009.

BURKE, R. **Hybrid recommender systems: Survey and experiments**. User modeling and user-adapted interaction, v. 12, n. 4, p. 331-370, 2002.

CABRÉ, G.J. **Filtragem Colaborativa Aplicada à Recomendação Musical**. 2011. 56 p. Dissertação (Mestrado em Informática Aplicada) - Fundação Edson Queiroz, Universidade de Fortaleza, Fortaleza, 2011. Disponível em: <https://upcommons.upc.edu/bitstream/handle/2099.1/13071/Monografia_FINAL.pdf>. Acesso em: 19 de mar. 2017.

CAZELLA, S.C.; NUNES, M.A.S.; REATEGUI, E.B. **A Ciência da opinião: Estado da arte em Sistemas de Recomendação**. CSBC – XXX Congresso da SBC – Jornada de Atualização de Informática-JAI, p. 161-216, 2010.

CGI.BR – Comitê Gestor da Internet no Brasil. **TIC Domicílios 2014**. Disponível em: <http://www.cgi.br/media/docs/publicacoes/2/TIC_Domicilios_2014_livro_eletronico.pdf>. Acesso em: 07 de mar. 2017.

CONECTAÍ. **34% dos internautas brasileiros assistem conteúdo on demand uma vez por semana**. Disponível em: <http://conecta-i.com/?q=pt-br/34-dos-internautas-brasileiros-assistem-conte%C3%BAdo-demand-uma-vez-por-semana>. Acesso em: 09 de mar. 2017.

EBAY. Disponível em: <http://www.ebay.com>. Acesso em: 13 de nov. 2016.

ENCYCLOPEDIA BRITANNICA. **Machine Learning**. Disponível em: <https://global.britannica.com/technology/machine-learning>. AcessO em: 08 de maio 2017.

FACEBOOK DEVELOPERS. Disponível em: <https://developers.facebook.com>. Acesso em: 09 de mar. 2017.

FAZIO, M.R. **Previsão de Avaliações em Sistemas de Recomendação para Nichos de Mercado**. 2013. 114 p. Dissertação (Pós-Graduação em Engenharia de Sistemas) - Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013. Disponível em: <http://www.cos.ufrj.br/uploadfile/1365598708.pdf>. Acesso em: 05 de maio 2017.

FIREBASE. Disponível em: <https://firebase.google.com>. Acesso em: 09 de mar. 2017.

GOLDBERG, D; NICHOLS, D; OKI, B.M; TERRY, D. **Using collaborative filtering to weave an information Tapestry**. InCommun. ACM, New York, vol .35, n.12, p. 61-70, 1992.

GOOGLE PLAY. Disponível em: <https://play.google.com>. Acesso em: 22 maio 2017.

HERLOCKER, J. *et al*. **Evaluating Collaborative Filtering Recommender Systems**. ACM Transactions on Information Systems, 2004, p. 5-53.

IBGE. **Instituto Brasileiro de Geografia e Estatística**. Disponível em: <http://www.ibge.gov.br>. Acesso em: 07 de mar. 2017.

PNAD. **Pesquisa Nacional por Amostra de Domicílios (PNAD) 2014**. Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/livros/liv94935.pdf>. Acesso em: 22 de maio 2017.

IBM. **Sistemas de Recomendação**. Disponível em: < https://www.ibm.com/developerworks/br/local/data/sistemas_recomendacao>. Acesso em: 11 de mar. 2017.

JAVA. Disponível em: <https://www.java.com>. Acesso em: 07 de mar. 2017.

JIN, W.; SRIHARI, R. **Graph-based text representation and knowledge discovery**. In Proceedings of the 2007 ACM symposium on Applied computing. ACM. New York, 2007. p. 807–811.

KOSALA, BLOCKEEL R. **Web mining research: a survey.** SIGKDD Explor, 2000.

LIRA, S.A. **Análise de Correlação**: Abordagem Teórica e de Construção dos Coeficientes com Aplicações. 2004. 209 p. Dissertação (Pós-Graduação em Métodos Numéricos) - Engenharia, Universidade de Federal do Paraná, 2004. Disponível em: <http://www.ipardes.gov.br/biblioteca/docs/dissertacao_sachiko.pdf>. Acesso em: 23 maio 2017.

MELO, E.V. **Sistema de Recomendação de Imagens Baseado em Atenção Visual**. 2016. 142 p. Tese (Doutorado em Ciência da Computação) - Faculdade de Computação, Universidade Federal de Uberlândia, Minas Gerais, 2016. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/17824/1/SistemaRecomendacaoImagens.pdf>. Acesso em: 11 de maio 2017.

NETFLIX. Disponível em: <https://www.netflix.com>. Acesso em: 13 nov. 2016.

PAZZANI, M.J. **A Framework for Collaborative, Content-Based and Demographic Filtering**. Artificial intelligence review, 1999, p. 393-408.

PETERS, J.F. **Engenharia de Software**: Teoria e Prática. Rio de Janeiro. Editora Campus, 2001.

PRESSMAN, R.S. **Engenharia de Software**. 7ª Edição, São Paulo, Editora: Makron Books, 2007.

PRESSMAN, R.S. **Engenharia de Software:** Uma abordagem Profissional. 7ª Edição, São Paulo, Editora: Makron Books, 2011.

REAL, R.; VARGAS, J. **The probabilistic basis of jaccard's index of similarity**. Systematic biology. 1996. p. 380–385.

REATEGUI, E.B.; CAZELLA, S.C. **Sistemas de recomendação**. XXV Congresso da Sociedade Brasileira de Computação, 2005. p. 306-348. Disponível em: <https://pdfs.semanticscholar.org/ff37/56d0e7d480dd098b334df5006a740d11ce06.pdf>. Acesso em: 20 de set. 2016.

RICCI, F., ROKACH, L., SHAPIRA, B. **Introduction to recommender systems handbook**. Recommender Systems Handbook. Boston, MA: Springer, 2011
.
ROZENDO, R.G. **Avaliação de Sistemas de Recomendação com uma Proposta de um Algoritmo Híbrido**. 2017. 90 p. Dissertação (Graduação em Engenharia de Computação e Informação) - Escola Politécnica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017. Disponível em: <http://monografias.poli.ufrj.br/monografias/monopoli10019604.pdf>. Acesso em 13 abr. 2017.

SAMPAIO, I.A. **Aprendizagem Ativa em Sistemas de Filtragem Colaborativa**. 2006. 86 p. Tese (Mestrado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2006. Disponível em: < http://repositorio.ufpe.br/bitstream/handle/123456789/2608/arquivo5346_1.pdf>. Acesso em: 11 de mar. 2017.

SCHULTE, R.G. **Escalonamento no Sistema Operacional Android**. 2016. Congresso de Sistemas Operacionais do CPoli da UCPEL, VOL. 1, NO. 1, 2016. Disponível em: <http://www.olaria.ucpel.tche.br/soii/lib/exe/fetch.php?media=rafael_gouvea_schulte.pdf>. Acesso em: 18 de maio 2017.

SILVA, R.G.N. **Sistema de Recomendação baseado em conteúdo textual: avaliação e comparação**. Salvador, 2014. Dissertação (Mestrado Multiinstitucional em Ciência da Computação) – Universidade Federal da Bahia, Universidade Estadual de Feira de Santana, 2014. Disponível em: <https://repositorio.ufba.br/ri/bitstream/ri/19281/1/dissertacao_mestrado_ciencia_computacao_rafael_glauber.pdf>. Acesso em: 22 de Maio 2017.

SEBASTIANI, F. **Machine learning in automated text categorization**. ACM. 2002. p 1-47.

SOMMERVILLE, I. **Engenharia de Software**. 8ª Edição. Editora: Pearson Addison-Wesley. São Paulo, 2007.

STATCOUNTER. **Android overtakes Windows for first time**. Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/livros/liv94935.pdf>. Acesso em: 08 de abr. 2017.

SU, X.; KHOSHGOFTAAR, T.M. **A survey of collaborative filtering techniques**. Adv. in Artificial Intell. 2009.

THE MOVIE DATABASE. Disponível em: <https://www.themoviedb.org>. Acesso em: 09 de mar. 2017.

WANG, Y;WANG, X.J. **A new approach to feature selection in text classification**. In Machine Learning and Cybernetics. Proceedings of 2005 International Conference on, vol. 6. 2005. p. 3814–3819.

YOUTUBE. Disponível em: <https://www.youtube.com>. Acesso em: 13 nov. 2016.

APÊNDICE A – Exemplo do cálculo do vizinho mais próximo utilizando a Correlação de Pearson

Dada a seguinte matriz de avaliações que os usuários deram para os seguintes filmes presentes na Tabela 3:

Tabela 3 – Matriz de notas.

		Filmes			
Usuários		Matrix	Clube da Luta	O Jogo da Imitação	A Viagem de Chihiro
1	João	4	0	5	5
2	Maria	4	1	3	5
3	José	3	0	2	4
4	Amelie	4	4	0	0
5	Pedro	2	1	3	5

Elaborado pelo autor

Dado dois usuários João e Pedro, o cálculo de correlação de Pearson explicado no Subcapítulo 2.2.2.3, corresponde a Equação 4:

$$corr_{1,5} = \frac{\sum_i (r_{ai} - \bar{r}_a)(r_{bi} - \bar{r}_b)}{\sqrt{\sum_i (r_{ai} - \bar{r}_a)^2}\sqrt{\sum_i (r_{bi} - \bar{r}_b)^2}} \quad (4)$$

Sendo que, $\bar{r}_a$ é a média de todas as avaliações de filmes do usuário A (João):

$$\frac{4 + 0 + 5 + 5}{4} = 3{,}5 \quad (5)$$

Sendo que, $\bar{r}_b$ é a média de avaliações de filmes do usuário B (Pedro):

$$\frac{2+1+3+5}{4} = 2{,}75 \tag{6}$$

r_{ai} é a avaliação que o usuário A deu para um item, neste caso, um filme e r_{bi} é a avaliação que o usuário B deu para o mesmo item. Sendo assim:

$$corr_{1,5} = \frac{\sum_i (4-3.5)*(2-2.75)+(0-3.5)*(1-2.75)+(5-3.5)*(3-2.75)+(5-3.5)*(5-2.75)}{\sqrt{(4-3.5)^2+(0-3.5)^2+(5-3.5)^2+(5-3.5)^2}\sqrt{(2-2.75)^2+(1-2.75)^2+(3-2.75)^2+(5-2.75)^2}}$$

$$corr_{1,5} = \frac{\sum_i (0.5)*(-0.75)+(-3.5)*(-1.75)+(1.5)*(1.25)+(1.5)*(2.25)}{\sqrt{(0.5)^2+(-3.5)^2+(1.5)^2+(1.5)^2}\sqrt{(-0.75)^2+(-1.75)^2+(0.25)^2+(-2.25)^2}} \tag{7}$$

$$corr_{1,5} = \frac{-0.375+6.125+0.375\_3.375}{\sqrt{17}\sqrt{8{,}1875}} = \frac{-0.375+6.125+0.375\_3.375}{4{,}123*2{,}86}$$

$$corr_{1,5} = \frac{9.5}{4{,}123*2{,}86} = 0{,}8$$

Os resultados dos cálculos do usuário João com os outros usuários: João x Maria: 0,3; João x José: -0,8; João x Amelie: 0,7; João x Pedro: 0.8.

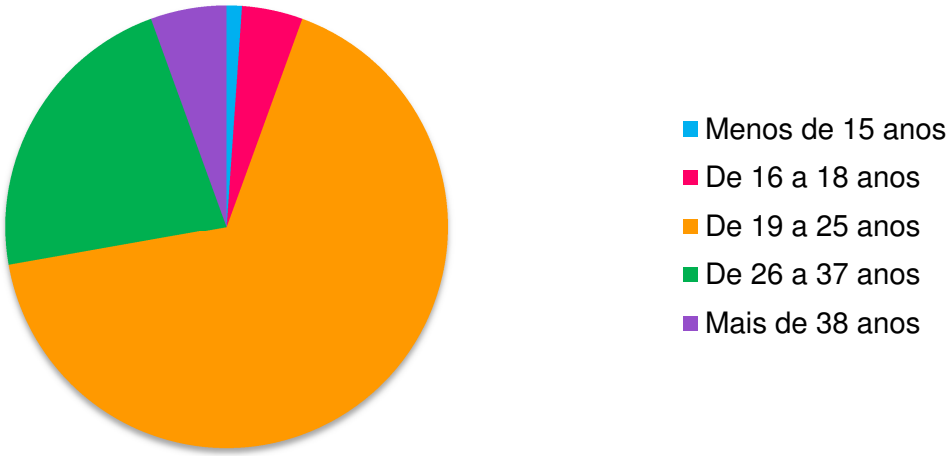
Os resultados variam de 1 para total similaridade e -1 para dissimilaridade total, então, pode-se confirmar que dentro desta lista de usuários, Pedro é o vizinho mais próximo de João.

APÊNDICE B – Pesquisa de Viabilidade do Projeto

A pesquisa de viabilidade buscou por respostas sobre a carência de aplicativos de filmes e sobre qual seria o público alvo, caso o aplicativo fosse lançado.

A maioria dos entrevistados foram pessoas entre 19 a 25 anos e 26 a 37 anos. A Figura 23 apresenta a faixa etária dos entrevistados.

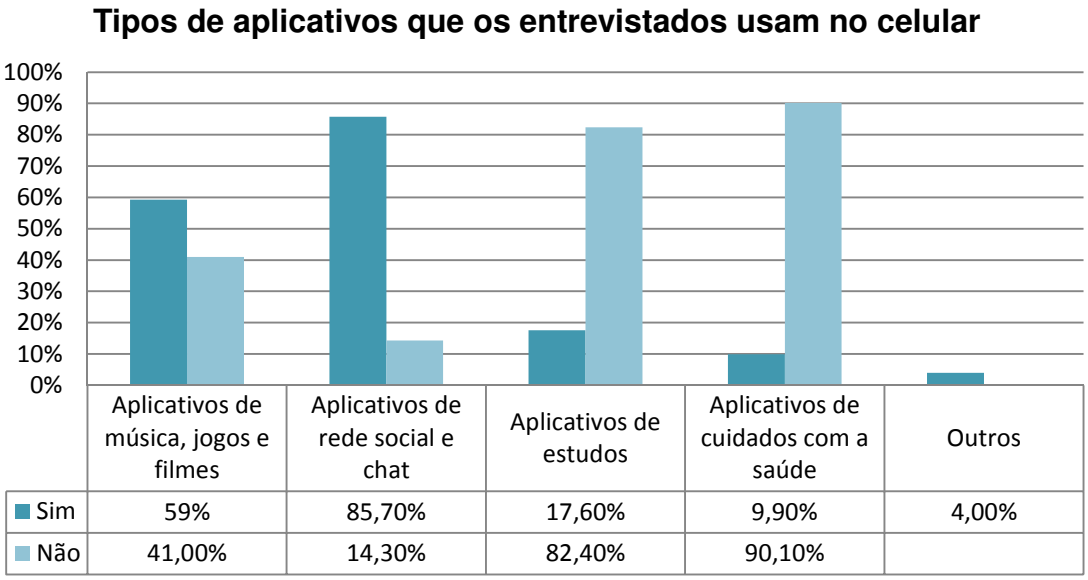
Figura 23 – Faixa etária dos entrevistados da pesquisa.



Fonte: Elaborado pelo autor.

Diante da pergunta “Quais tipos de aplicativos, você mais usa no celular?”, sendo que o entrevistado poderia responder mais de uma resposta. Mais de 85% responderam “Aplicativos de rede social e chat” e 59,3% para “Aplicativos de música, jogos e filmes”. Pode-se entender que os entrevistados utilizam os aplicativos de celular como forma de comunicação com outras pessoas e para o entretenimento. A Figura 24 apresenta os aplicativos mais utilizados dentre os entrevistados.

Figura 24 – Aplicativos mais utilizados pelos entrevistados da pesquisa.

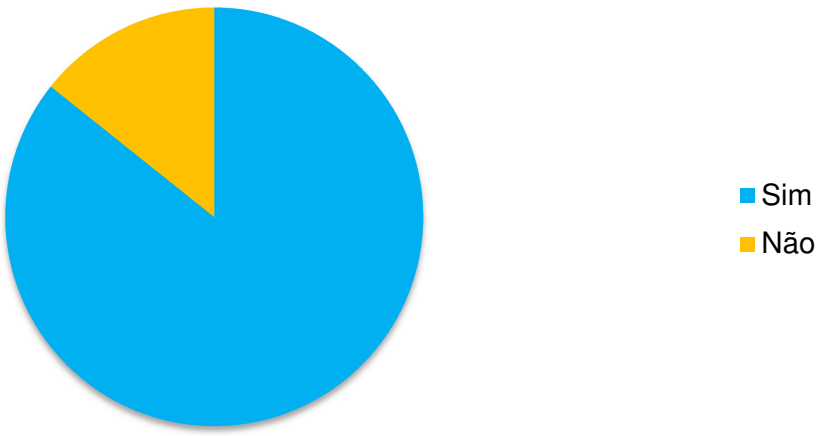


	Aplicativos de música, jogos e filmes	Aplicativos de rede social e chat	Aplicativos de estudos	Aplicativos de cuidados com a saúde	Outros
Sim	59%	85,70%	17,60%	9,90%	4,00%
Não	41,00%	14,30%	82,40%	90,10%	

Fonte: Elaborado pelo autor

O resultado da pergunta “Onde você procura por novos filmes para assistir?” trouxe a rotina dos entrevistados ao procurar indicações para assistir novos filmes. Os entrevistados têm buscado por novos filmes através de recomendações de serviços de *streaming* (Netflix, Google Filmes, Amazon Prime Video, entre outros) com 61,5% e pesquisando na internet com 60,4%. Outras formas votadas foram: recomendações de amigos (50,5%), cinema (37,4%), comerciais de TV (20,9%) e outros (4,4%). De acordo com a Figura 25, o total de entrevistados, 85,7% costumam receber recomendações de amigos ou conhecidos.

Figura 25 – Porcentagem de entrevistados que recebem recomendações de amigos.



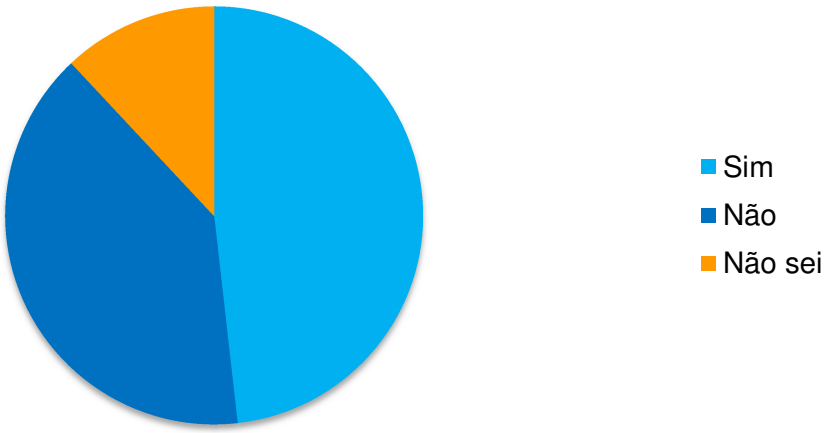
Fonte: Elaborado pelo autor.

Um dos resultados mais importantes da pesquisa foi o da pergunta “Você utiliza outras maneiras para organizar os filmes que já assistiu? Se sim, quais?”. De acordo com os resultados obtidos muitos entrevistados não têm meios de organizar os filmes já assistidos ou guardam tudo na memória, outros guardam sua lista de filmes de forma manual. As respostas obtidas foram:

1. Por algum tempo, mantive uma lista de melhores filmes já assistidos no Evernote. Não eram todos, mas sim os que eu julgava que eram realmente bons;
2. Utilizo o bloco de notas;
3. Sim, uso a listagem do Facebook já faz vários anos (porém acho meio precária hoje em dia), preciso migrar para uma ferramenta mais recente e com melhor controle;
4. Planilha;
5. Filmow.

Foi questionado também o uso do Netflix sobre o controle de filmes. 76,9% dos entrevistados fazem uso do Netflix, destes responderam a pergunta “Caso você utilize o Netflix, você acredita ter controle dos filmes que já assistiu?”. 48,2% responderam, não, 39,8% responderam sim e 12% não sabiam. A Figura 26 figura os resultados sobre o controle de filmes do Netflix.

Figura 26 – Resposta dos entrevistados sobre o controle de filmes do Netflix.



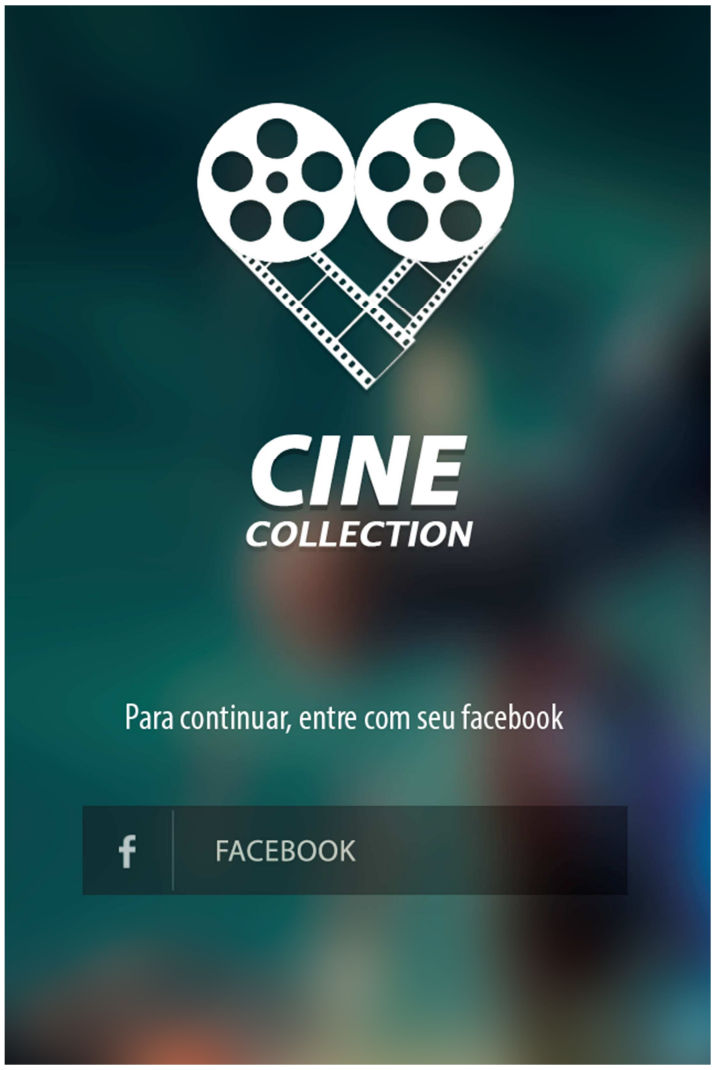
Fonte: Elaborado pelo autor

APÊNDICE C – Telas do Aplicativo

A.1 Tela de login do aplicativo

Ao abrir o aplicativo será exibido ao usuário uma espécie de tela de abertura, a qual exibe uma imagem do logotipo do Cine Collection. Observa-se essa tela na Figura 27.

Figura 27 – Tela de abertura do aplicativo.



Fonte: Elaborado pelo autor

A tela de abertura, além de exibir uma imagem ao usuário, também tem a função de carregar o botão do login e fazer com que redirecione para a tela inicial do aplicativo. A Figura 28 apresenta a tela de login com o Facebook no aplicativo.

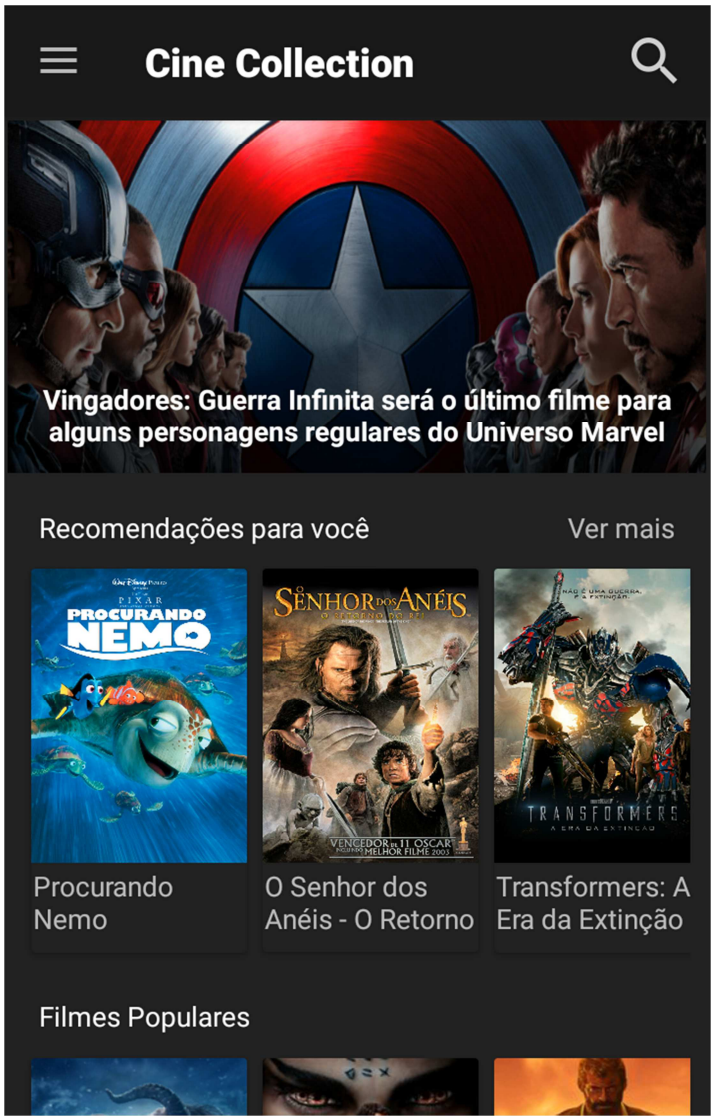
Figura 28 – Tela de login com o Facebook.

Fonte: Elaborado pelo autor

A.2 Tela inicial

Caso o retorno da tela inicial seja bem sucedido, ele redireciona para tela inicial do aplicativo. A tela inicial do aplicativo exibe os filmes populares e as recomendações em caso que login da tela inicial seja bem sucedido, além de incluir o botão de menu e de busca. A Figura 29 apresenta a tela inicial do aplicativo quando o usuário está logado.

Figura 29 – Tela inicial de recomendação de filmes.

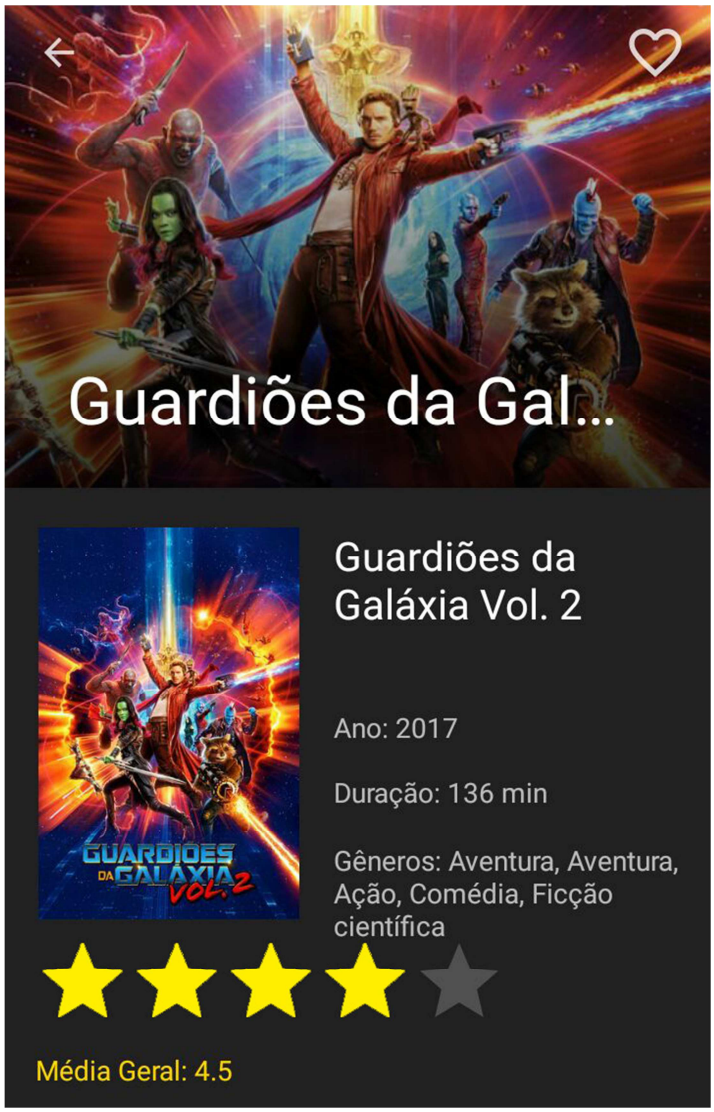


Fonte: Elaborado pelo autor

A.3 Tela de informações do filme

A tela de informações do filme mostra todas as informações do filme, como título, ano, sinopse, duração. Note que esta tela também traz as informações sobre os votos e os botões de Favoritar ou não, como também, há um elemento chamado "*Rating Bar*" que se trata do campo de avaliação utilizando as "estrelas". A Figura 30 apresenta a tela de informações detalhadas sobre o filme.

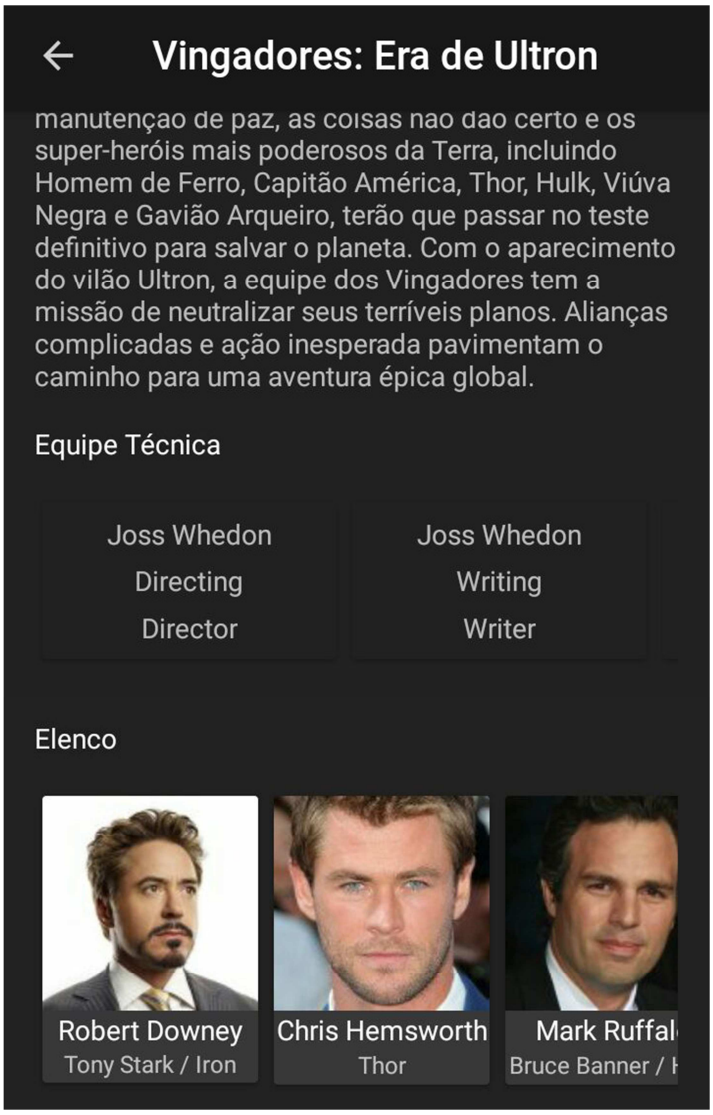
Figura 30 – Tela de informações detalhadas sobre um filme.



Fonte: Elaborado pelo autor

Ainda na tela de informações do filme é possível ver as informações sobre a equipe técnica e sobre o elenco. A Figura 31 apresenta como as informações adicionais são apresentadas.

Figura 31 – Informações do filme sobre o elenco e a equipe técnica.

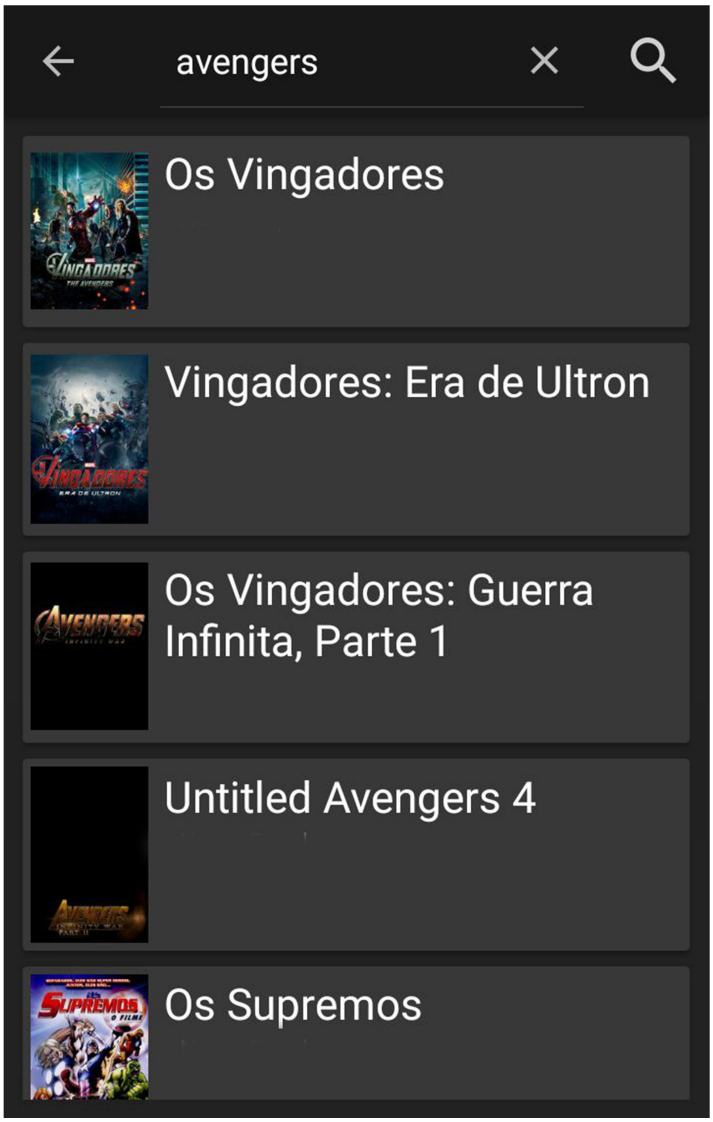


Fonte: Elaborado pelo autor

A.4 Tela de buscar filmes

A tela de busca de filmes mostra os filmes de acordo com o resultado de uma requisição através do campo de texto solicitando a pesquisa por alguma informação do filme. A Figura 32 apresenta a tela de busca por filmes.

Figura 32 – Tela de busca por filmes.

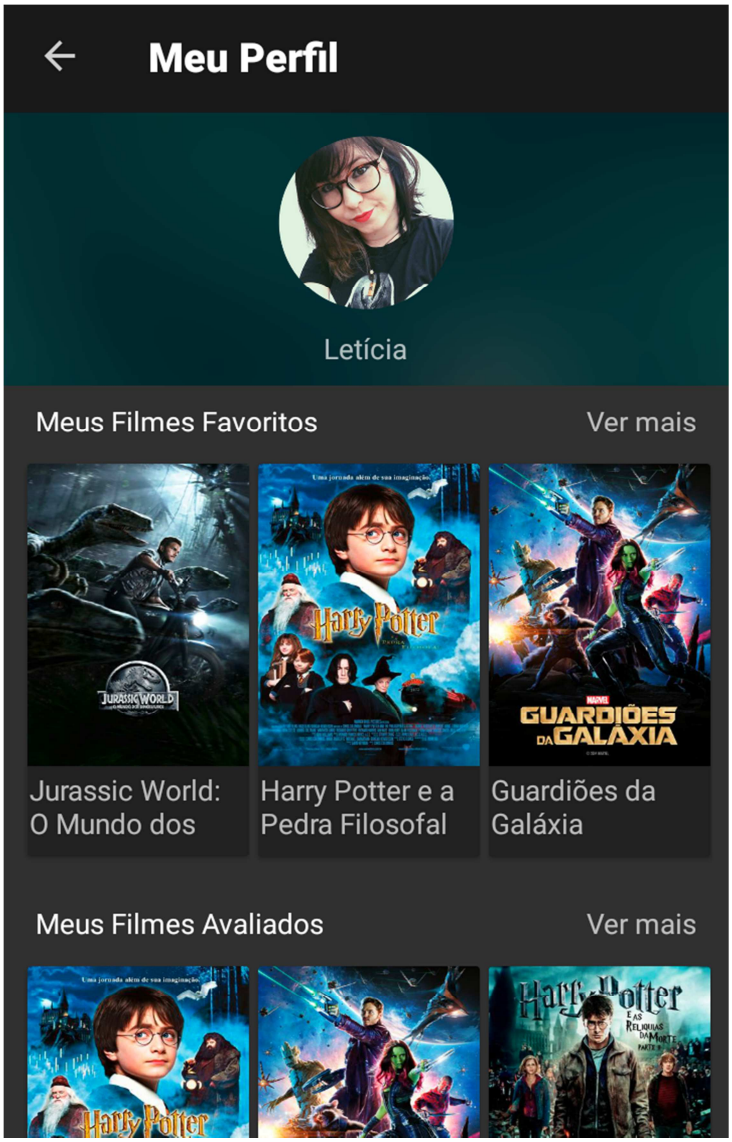


Fonte: Elaborado pelo autor

A.5 Tela de Perfil

A tela de perfil mostra um resumo dos filmes favoritados e dos avaliados bem como a quantidade de filmes de cada uma dessas categorias. Após clicar no botão “Ver mais” o aplicativo irá para a listagem completa da categoria selecionada. A Figura 33 apresenta a tela de perfil do usuário.

Figura 33 – Tela de perfil do usuário.

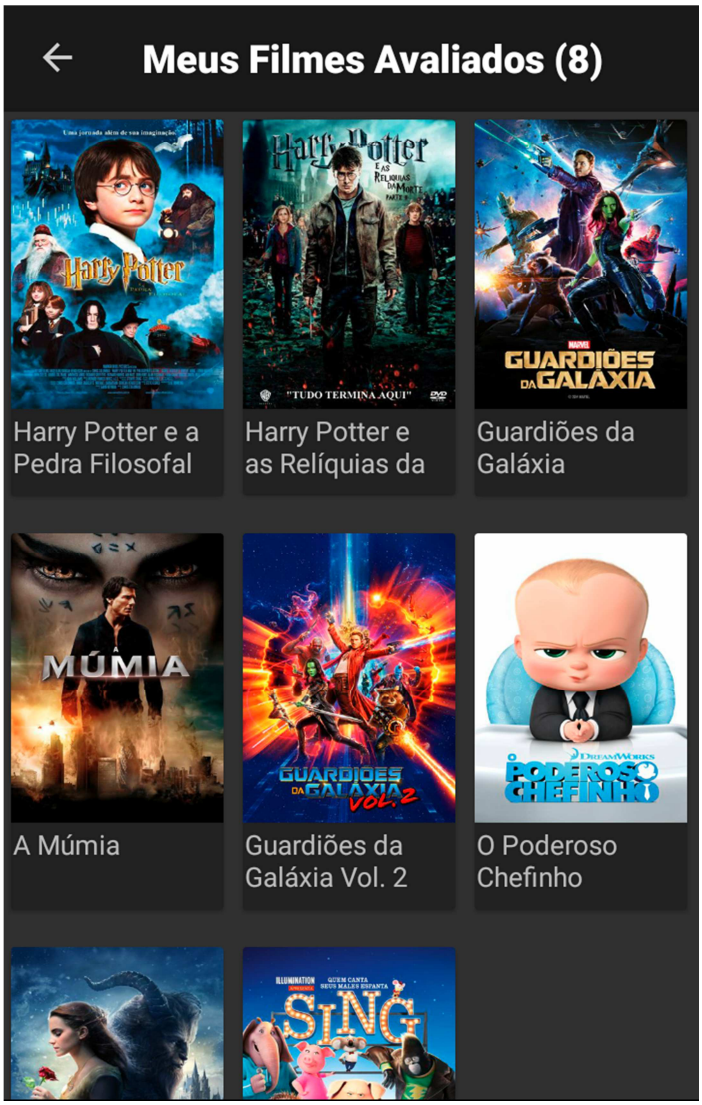


Fonte: Elaborado pelo autor

A.6 Tela de listagem de filmes

A tela de filmes favoritados e avaliados mostra uma listagem de todos os filmes que o usuário favoritou ou avaliou de acordo com a que o usuário solicitou, no qual ao clicar em uma miniatura do filme, o sistema leva para a página de informações do filme. A Figura 34 apresenta a tela de listagem de filmes favoritados ou avaliados pelo usuário.

Figura 34 – Tela de listagem de filmes avaliados e favoritados pelo usuário.

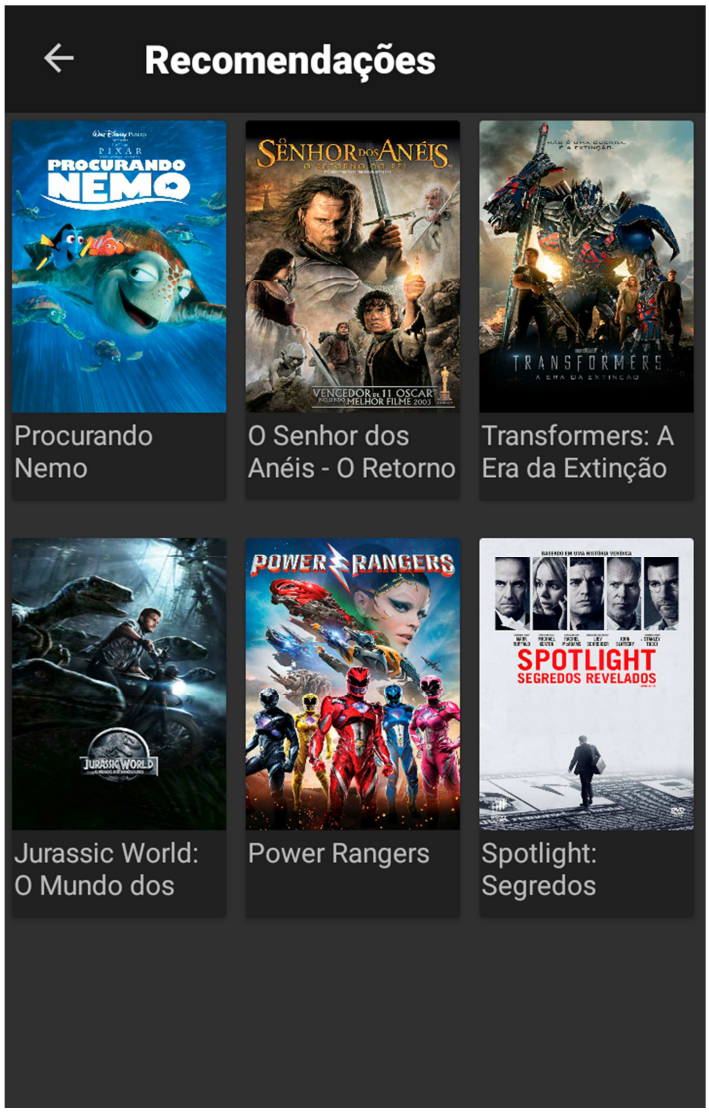


Fonte: Elaborado pelo autor

A.7 Tela de Recomendações de Filmes

A tela de recomendações de filmes se trata das recomendações personalizadas que o sistema de recomendação do aplicativo o processou. As recomendações são carregadas de forma automática quando a tela de recomendação é solicitada. A Figura 35 apresenta a tela de recomendação de filmes.

Figura 35 – Tela de Recomendação de Filmes.

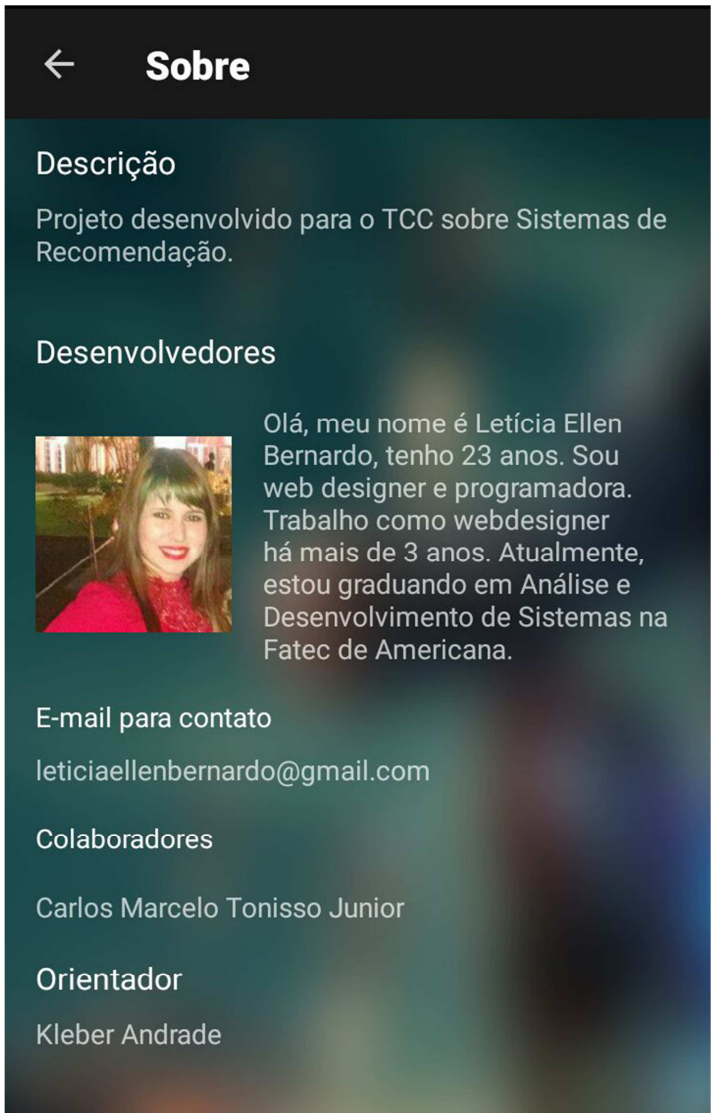


Fonte: Elaborado pelo autor

A.8 Tela de Sobre

A tela de Sobre trata-se da tela com informações sobre o aplicativo e sobre informações sobre o desenvolvedor e sobre a Fatec. Além de, uma descrição sobre o projeto. A Figura 36 apresenta a tela “Sobre”.

Figura 36 – Tela de informações sobre o aplicativo.



Fonte: Elaborado pelo autor

APÊNDICE D – Como adicionar login pelo Facebook no Android

Para utilizar os produtos do Facebook, primeiramente, deve ser um desenvolvedor do Facebook[34]. É necessário criar a conta de desenvolvedor, para isso, faça o login com o Facebook e cadastrar-se. Após o cadastro, o desenvolvedor já pode criar os seus próprios aplicativos do Facebook.

Para criar um aplicativo do Facebook, clique “Meus aplicativos” na barra lateral à direita e em “Adicionar um novo aplicativo”. A Figura 37 mostra como adicionar um aplicativo no Facebook.

Figura 37 – Como adicionar um novo aplicativo no Facebook.

Fonte: Elaborado pelo autor

Crie um nome para o seu aplicativo e clique para prosseguir. Já direcionado para a página de configuração do aplicativo, clique na navegação “adicionar produto” e na caixa de Login do Facebook, clique em “Começar”. A Figura 38 mostra como adicionar o produto Login do Facebook.

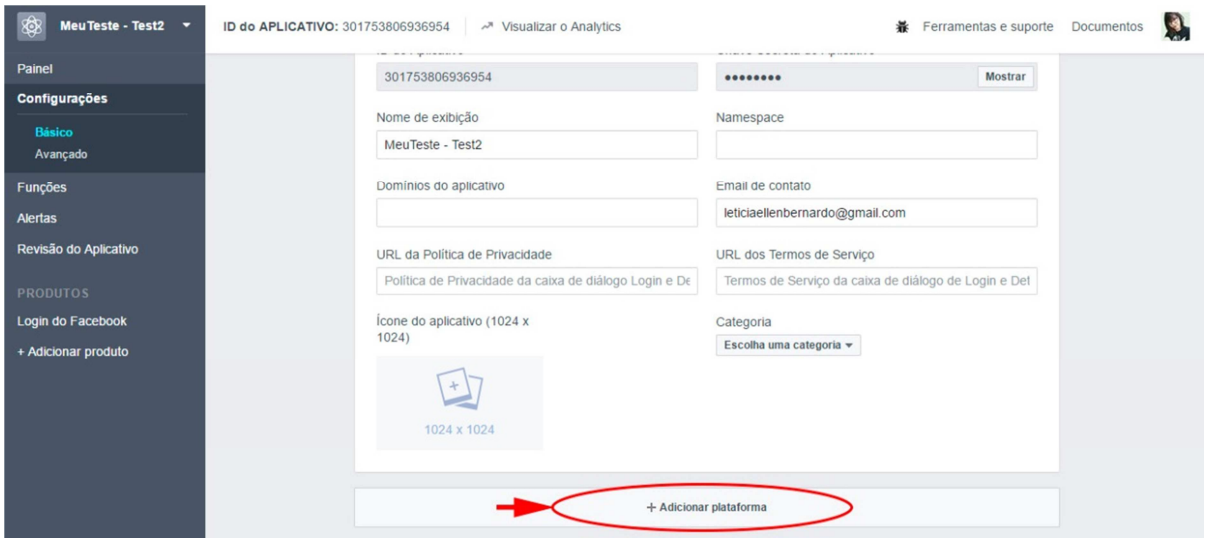
[34] Facebook Developers. Disponível em: <https://developers.facebook.com>.

Figura 38 – Como adicionar o produto “Login do Facebook”.

Fonte: Elaborado pelo autor

Para configurar o login do Facebook no aplicativo é necessário o Facebook liberar o acesso do seu aplicativo e da plataforma do Android. Na navegação Configurações > Básico, clique em “Adicionar plataforma”. A Figura 39 mostra como Adicionar a plataforma Android ao seu aplicativo do Facebook.

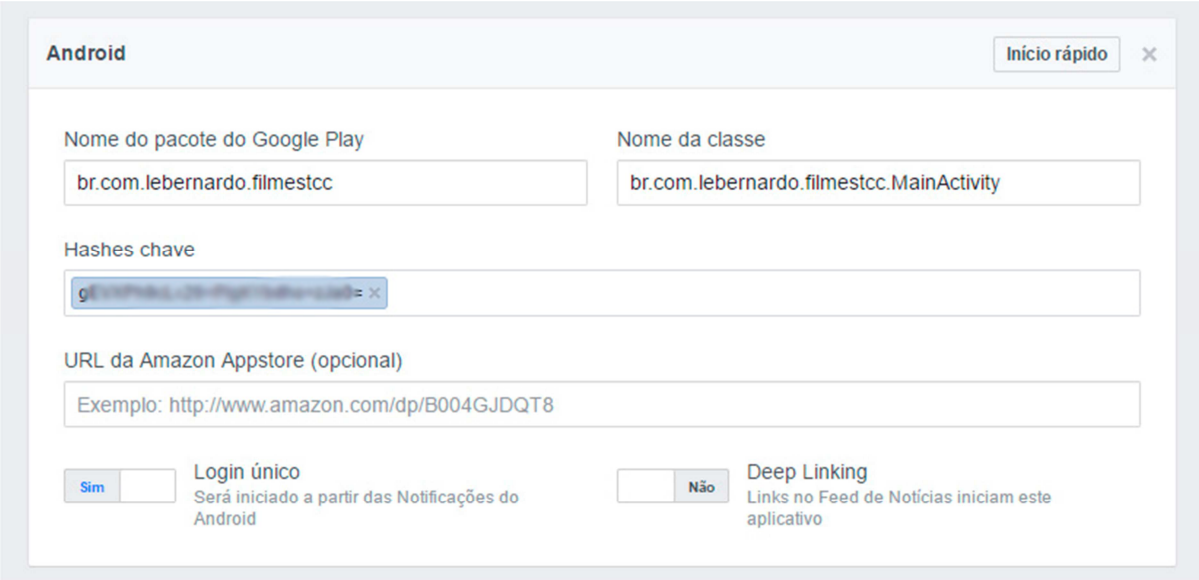
Figura 39 – Adicionar plataforma do Android do aplicativo no Facebook.



Fonte: Elaborado pelo autor

Abrirá um modal com várias opções de plataformas, selecione plataforma Android. Na próxima caixa, insira as informações: o nome do pacote do seu projeto, o nome da classe que terá o Login e a chave de *Hash*. A Figura 40 demonstra um exemplo de como preencher as informações ao adicionar a plataforma Android no aplicativo do Facebook.

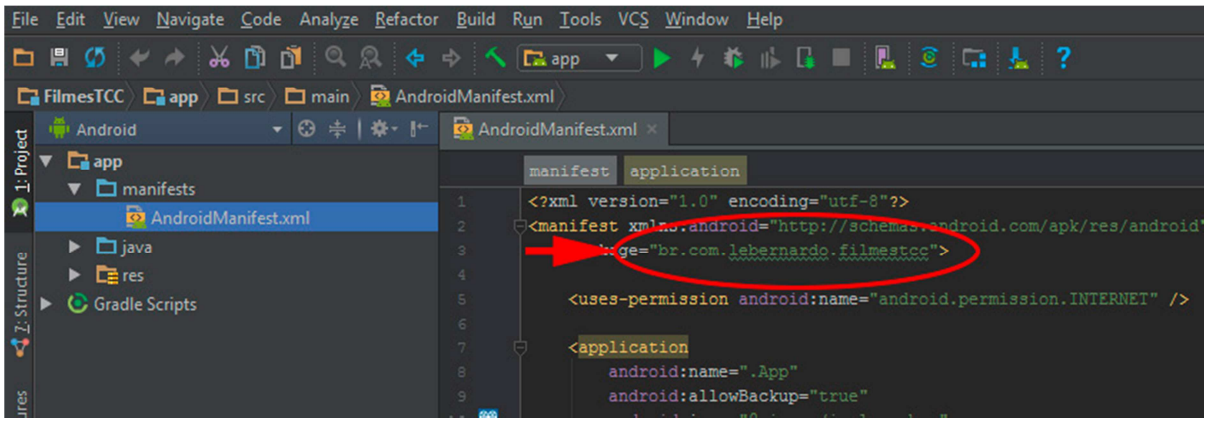
Figura 40 – Preenchimento das informações ao adicionar a plataforma Android no Facebook.



Fonte: Elaborado pelo autor

O nome do pacote do projeto é encontrado no arquivo AndroidManifest.xml. A Figura 41 aponta o local no qual contem o nome do pacote do aplicativo.

Figura 41 – Aonde encontrar o nome do pacote do aplicativo Android.



Fonte: Elaborado pelo autor

É necessário gerar um *Hash* de chave de lançamento, pois sem ele, a integração pode não funcionar corretamente quando o aplicativo for lançado. Para gerar um *Hash* de chave de lançamento e adicioná-lo as configurações do Android, entre pelo terminal na pasta do Java, utilizando o comando:

cd C:\Program Files\Java\jdk1.8.0_60\bin

Verifique a localização da instalação do seu Java e certifique-se que está na pasta *bin* e que nesta pasta contém o arquivo *keytool.exe*. E execute este comando alterando os valores entre parênteses.

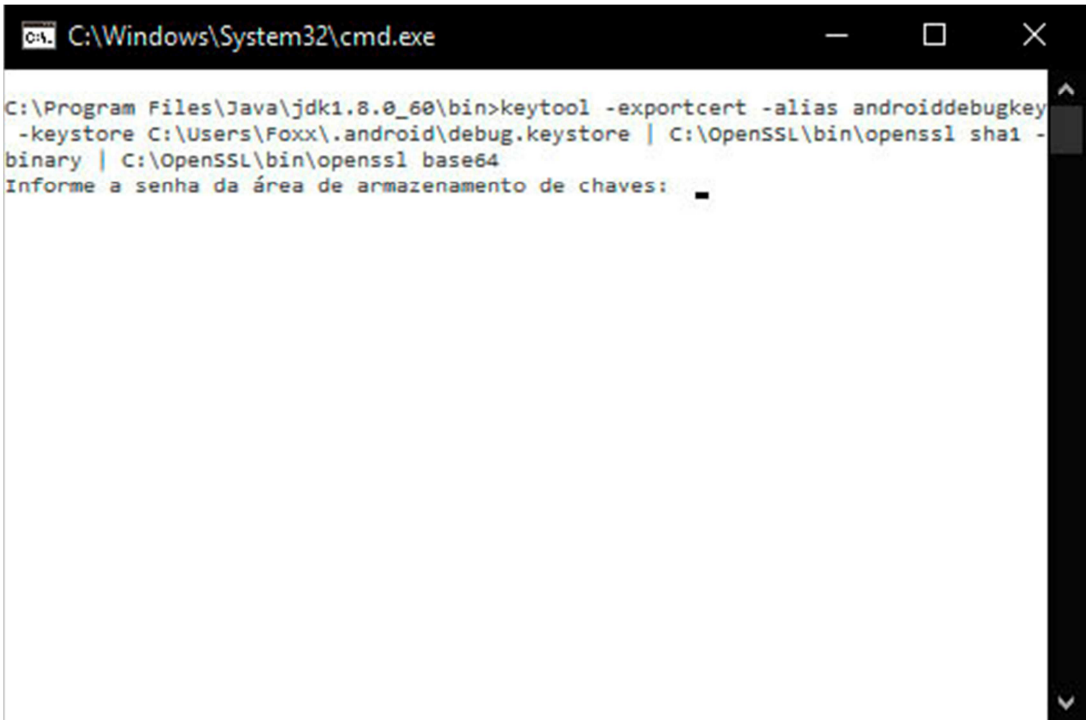
keytool -exportcert -alias androiddebugkey -keystore (local da keystore do android)\debug.keystore | (Seu local do OpenSSL)\openssl sha1 -binary | (Seu local do OpenSSL)\openssl base64

Um exemplo de utilização:

keytool -exportcert -alias androiddebugkey -keystore C:\Users\Foxx\.android\debug.keystore | C:\OpenSSL\bin\openssl sha1 -binary | C:\OpenSSL\bin\openssl base64

A Figura 42 apresenta um exemplo da requisição de senha ao executar o comando acima:

Figura 42 – Exemplificação do comando keytool.



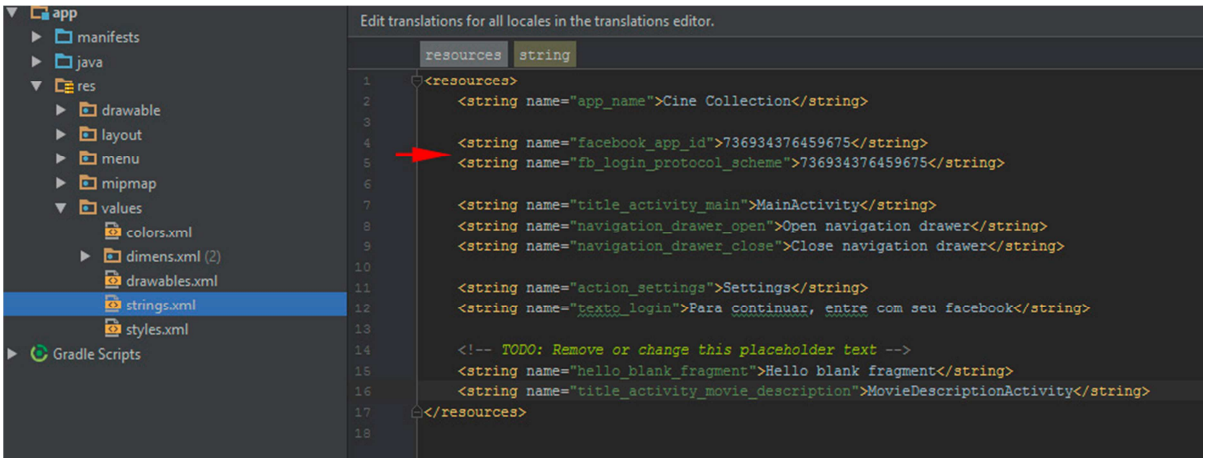
Fonte: Elaborado pelo autor

Será solicitada uma senha para o armazenamento de chaves, digite uma senha e aperte <*enter*>. Copie e cole a chave de *Hash* do Prompt de Comando no campo de "*Hash* chave" do Facebook, libere o Login único e clique em "Salvar alterações".

Para adicionar o botão de login ao seu projeto, inicialmente, copie e cole a id do aplicativo no arquivo strings.xml.

Um exemplo de caminho: app > res > values > strings.xml. A Figura 43 mostra como adicionar a chave da API do Facebook.

Figura 43 – Local onde adiciona-se a chave da API do Facebook.



Fonte: Elaborado pelo autor

Em seguida, abra o AndroidManifest.xml. Adicione um elemento de *uses-permission* ao manifesto:

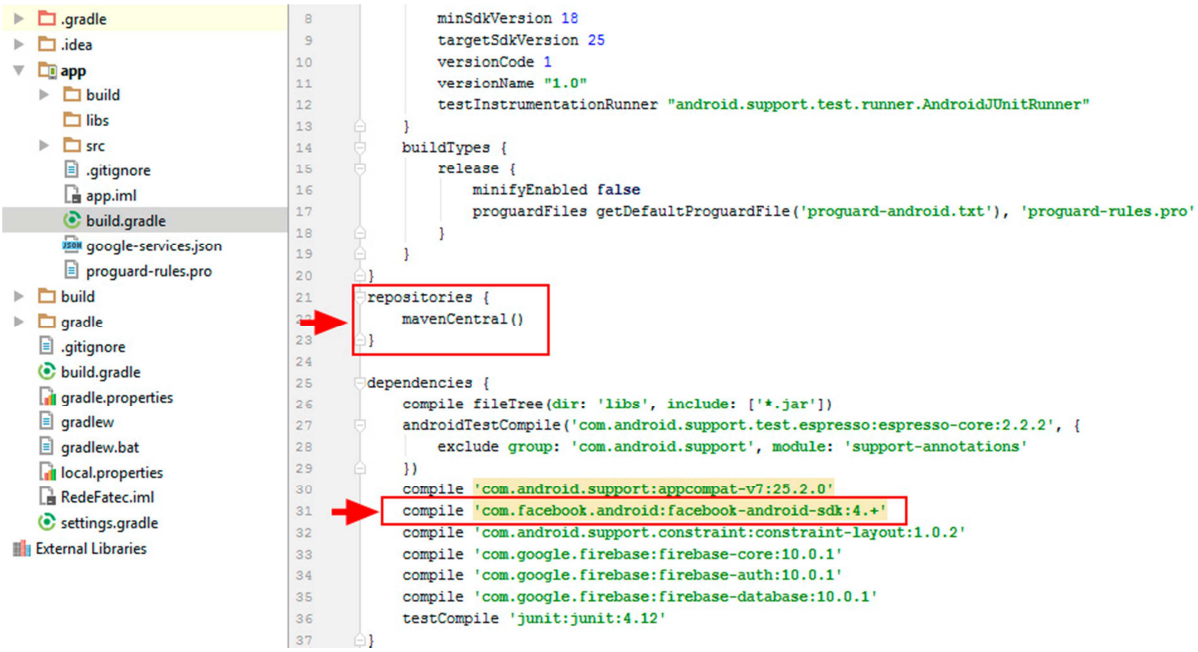
<uses-permission android:name="android.permission.INTERNET"/>

Adicione um elemento *meta-description* ao elemento *application* e o preencha com o ID do Facebook:

<meta-data android:name="com.facebook.sdk.ApplicationId"
android:value="@string/facebook_app_id"/>

Abra o build.gradle. Um exemplo de caminho: app > build.gradle. Adicione a biblioteca *mavenCentral()* aos repositórios e dependência do SDK do Facebook. A Figura 44 aponta como adicionar as dependências do SDK do Facebook.

Figura 44 – Local no qual adiciona-se as dependências do Facebook.



Fonte: Elaborado pelo autor

Para mais informações sobre a implementação do Sdk do Facebook, acesse a documentação do SDK do Facebook[35].

Adicione o botão de login do Facebook no arquivo.xml. A Figura 45 demonstra como adicionar o botão do Facebook.

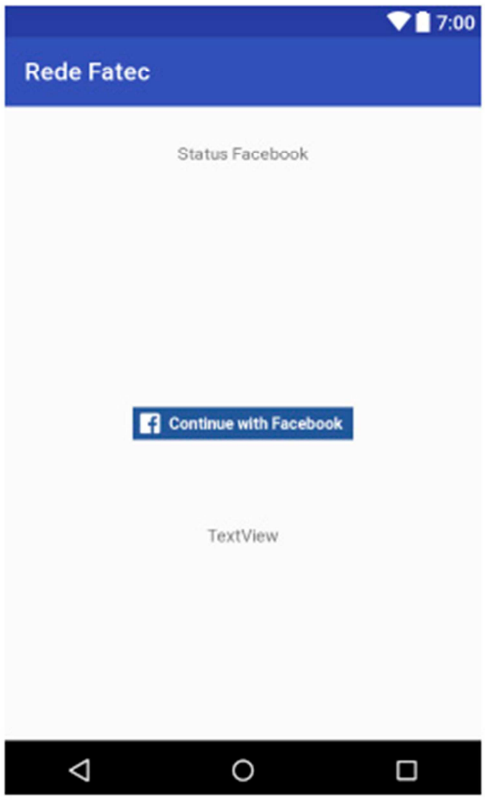
[35] Documentação do SDK do Facebok. Disponível em <https://developers.facebook.com/docs/android/getting-started>.

Figura 45 – Botão do Facebook.

```
<TextView
    android:id="@+id/tvStatus"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:gravity="center"
    android:text="Status Facebook" />

<com.facebook.login.widget.LoginButton
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"/>

<TextView
    android:id="@+id/tvNome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="69dp"
    android:text="TextView"
    android:layout_below="@+id/login_button"
    android:layout_centerHorizontal="true" />
```



Fonte: Elaborado pelo autor

Na classe Java que manipula o aquivo.xml, inicie o *CallbackManager*. O *CallbackManager* gerencia os retornos de chamada do Sdk do Android a partir do método *onActivityResult*.

Inicie um *LoginButton* dentro do método *onCreate*, o *LoginButton* é a referência do botão de login com o Facebook. Para detectar as ações ou eventos desse botão é utilizado a classe *CallbackManager*. Através do retorno do *CallbackManager* no método on *ActivityResult*, pode-se passar o retorno para o botão de login através do método *registerCallback*, O retorno pode ser de 3 possíveis resultados:

Se o login foi bem sucedido, executará o método *onSuccess*, que pode retornar as informações básicas do usuário, como o *token* de sessão. Quando a ação é cancelada, o método *onCancel* é executado. E se retornar erro, o método *onError* é executado. A Figura 46 apresenta o trecho do código da chamada do botão do Facebook para mostrar a interface de login.

Figura 46 – Trecho do código do botão de Login com o Facebook.

```
private CallbackManager callbackManager;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    FacebookSdk.sdkInitialize(getApplicationContext());
    setContentView(R.layout.activity_main);

    LoginButton loginButton = (LoginButton) findViewById(R.id.login_button);
    callbackManager = CallbackManager.Factory.create();

    loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {...}

        @Override
        public void onCancel() { Log.i("TAG", "onCancel"); }

        @Override
        public void onError(FacebookException error) { Log.i("TAG", "onError"); }
    });

}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    callbackManager.onActivityResult(requestCode, resultCode, data);
}
```

Fonte: Elaborado pelo autor

Outro ponto importante ao configurar o login com o Facebook é liberar e tornar o aplicativo disponível ao público. Caso contrário, apenas o criador e os usuários liberados por meio do painel funcionarão o login.

Na navegação “Revisão do Aplicativo”, torne seu aplicativo público. A Figura 47 apresenta como tornar o aplicativo público à todos os usuários.

Figura 47 – Como tornar o aplicativo público.

Fonte: Elaborado pelo autor

APÊNDICE E – Integração do aplicativo Android com o Firebase

Para fazer a integração do aplicativo com o Firebase é necessário estar logado em uma conta do Google e entrar na página do Firebase[36]. Para criar um projeto no Firebase, clique em “Primeiros Passos”. A Figura 48 mostra como começar um projeto no Firebase.

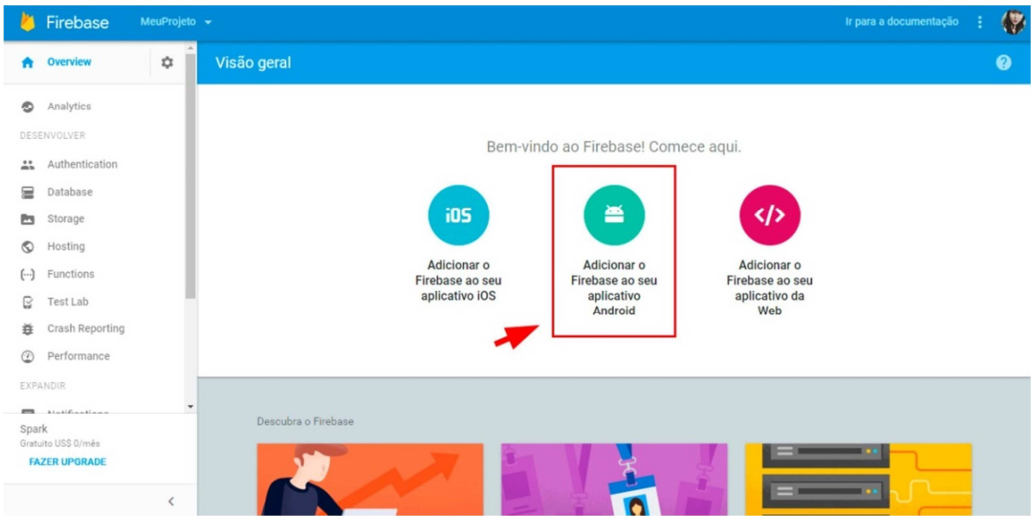
Figura 48 – Começando um projeto no Firebase.

Fonte: Elaborado pelo autor

Adicione um novo projeto, escolhendo um nome e a localidade, clique em “Criar projeto”. Na página inicial do projeto no Firebase, clique em “Adicionar o Firebase ao seu aplicativo Android”. Figura 49 apresenta como iniciar a integração do aplicativo Android com o Firebase.

[36] Firebase. Disponível em: <https://firebase.google.com>.

Figura 49 – Iniciando a integração do aplicativo com o Firebase.



Fonte: Elaborado pelo autor

Insira as seguintes informações: o nome do pacote do aplicativo Android, um apelido (opcional) e o certificado de assinatura de depuração SHA-1 (opcional).

Para obter um certificado de assinatura de depuração SHA-1, vá ao Android Studio com seu projeto aberto, acesse o terminal e digite o comando:

keytool -exportcert -list -v -alias androiddebugkey -keystore c:\Users\Foxx\.android\debug.keystore

Coloque a senha e aperte <*enter*>. A assinatura retornará em vários formatos. Copie a assinatura da chave em SHA-1 e cole no campo de certificado de assinatura e clique em “Registrar App”.

Faça o download do arquivo *google-services.json*. Altere a visualização do Android Studio para Projeto. Mova o arquivo *google-services.json* para o diretório raiz do módulo *app*.

Modifique os arquivos build.gradle para usar o plugin. Adicione as dependências no build.gradle do nível do usuário. Um exemplo de caminho: raiz > build.gradle.

```
classpath 'com.google.gms:google-services:3.0.0'
```

E as dependências do nível do aplicativo:

```
apply plugin: 'com.google.gms.google-services'
```

Por fim, pressione “Sincronizar agora” na barra lateral que aparece no IDE. A Figura 50 apresenta o link “Sync now” do Android Studio.

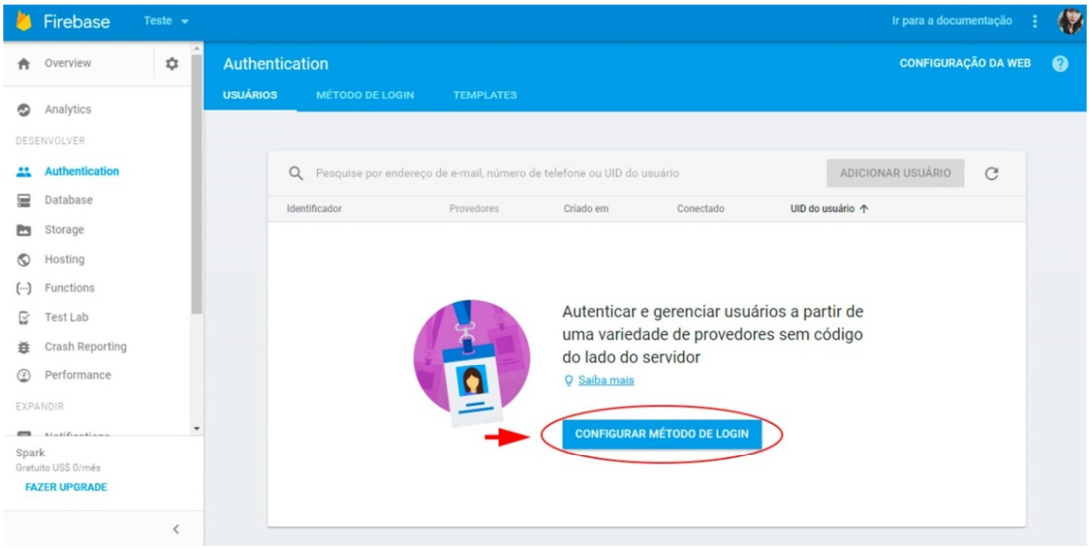
Figura 50 – Como sincronizar o projeto do Android Studio.

Fonte: Elaborado pelo autor

APÊNDICE F – Login do Facebook com autenticação no Firebase

Na página inicial do projeto no Firebase, entre no item “*Authentication*” do menu ao lado esquerdo e clique “Configurar método de login”. A Figura 51 apresenta a tela inicial de autenticação de usuários e como encontrar o botão “Configurar método de login”.

Figura 51 – Tela de autenticação de usuários do Firebase.



Fonte: Elaborado pelo autor

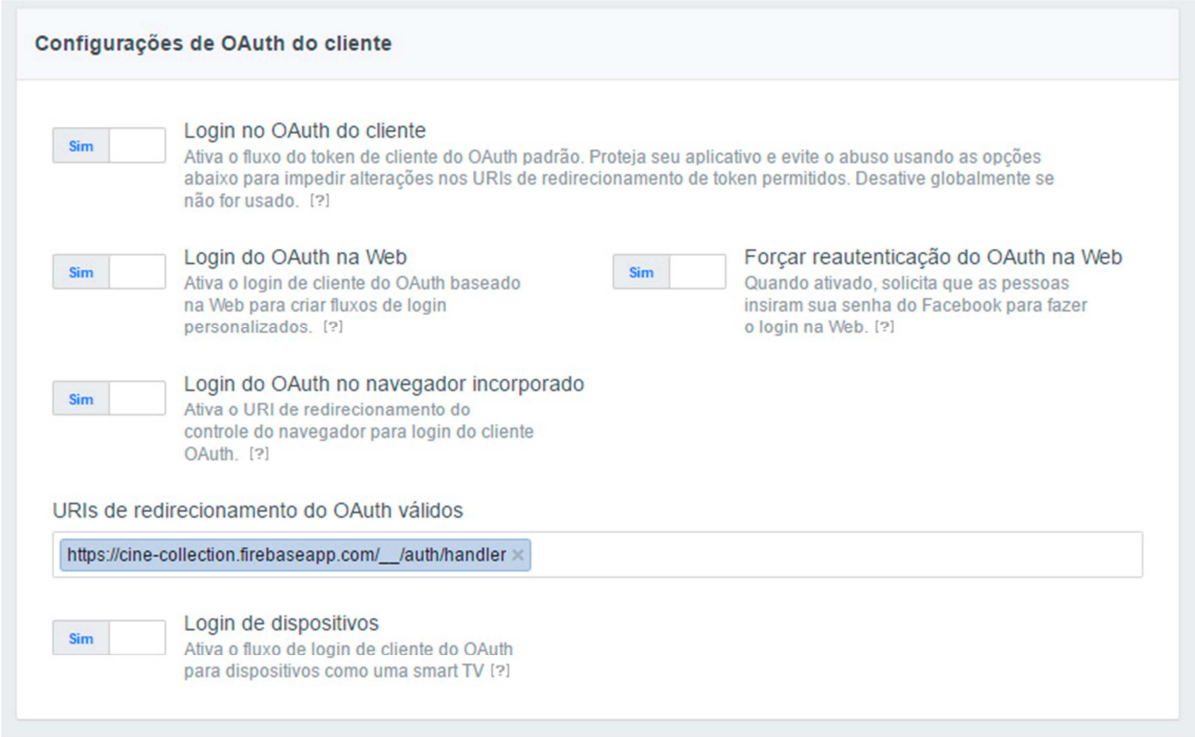
Escolha o provedor do Facebook, clique em “ativar”. Preencha as informações de App Id e App Secret pelas chaves de API fornecidas pelo Facebook acessadas pelo painel de administração do aplicativo. A Figura 52 apresenta como encontrar a as chaves da API do Facebook.

Figura 52 – Chaves da API do Facebook.

Fonte: Elaborado pelo autor

Copie a URI de redirecionamento de OAuth do Firebase e, no painel do aplicativo do Facebook, clique no menu “Login do Facebook” e, em seguida, na caixa “Configurações de OAuth do cliente”. Cole a URI de redirecionamento do OAuth no campo fornecido, habilite as configurações e salve as alterações. A Figura 53 mostra as configurações de OAuth do cliente.

Figura 53 – Configurações de OAuth do Facebook.



Fonte: Elaborado pelo autor

Adicione as dependências que são necessárias para fazer a autenticação:

```
compile 'com.google.firebase:firebase-core:10.0.1'
compile 'com.google.firebase:firebase-auth:10.0.1'
```

Na classe Java que manipula o login, adicione a classe:

```
private FirebaseAuth mAuth;
private FirebaseAuth.AuthStateListener mAuthListener;
```

No método *onCreate*(), adicione a instância do *FirebaseAuth*:

```
mAuth = FirebaseAuth.getInstance();
```

Adicione o método *signInWithCredential* que ao receber o *token* de acesso do Facebook faz a autenticação pelo Firebase.

```
private void handleFacebookAccessToken(AccessToken token) {
    Log.d("aqui está o token", "handleFacebookAccessToken:" + token);

    AuthCredential credential =
  FacebookAuthProvider.getCredential(token.getToken());

    mAuth.signInWithCredential(credential)
.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
      Log.d("signincomplete", "onComplete:" + task.isSuccessful());

    if (!task.isSuccessful()) {
      Log.w("sign in", "signInWithCredential", task.getException());
    }
    else {
      String uid = mAuth.getCurrentUser().getUid();

      Toast.makeText(MainActivity.this, "Logado!!!!!!!!!!!!" ,
Toast.LENGTH_SHORT).show();
    }

   }
 });
}
```

No método *registerCallback* do botão de login do Facebook, adicione ao método *onSuccess* a chamada para o método *handleFacebookAccessToken* passando o *token* de acesso como parâmetro.

```
loginButton.registerCallback(callbackManager, new
FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) {

        tvStatus.setText("Login feito com sucesso!\n"
                + loginResult.getAccessToken().getUserId()
                + "\n\n"
                + loginResult.getAccessToken().getToken());

        handleFacebookAccessToken(loginResult.getAccessToken());

        Log.d("", "facebook:onSuccess:" + loginResult);

    }

    @Override
    public void onCancel() {
        tvStatus.setText("Login Cancelado");

    }

    @Override
```

```
    public void onError(FacebookException error) {

    }
});
```

Crie o método *AuthStateListener* do *FirebaseAuth*, este método checa se o usuário foi logado. Se o retorno do objeto *FirebaseUser* for diferente de nulo. Indica que a autenticação foi validada com sucesso.

```
mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {
        FirebaseUser user = firebaseAuth.getCurrentUser();

        if (user != null) {
            Log.d("logando", "onAuthStateChanged:signed_in:" +
user.getUid());
        } else {
            Log.d("saindo", "onAuthStateChanged:signed_out");
        }
    }
};
```

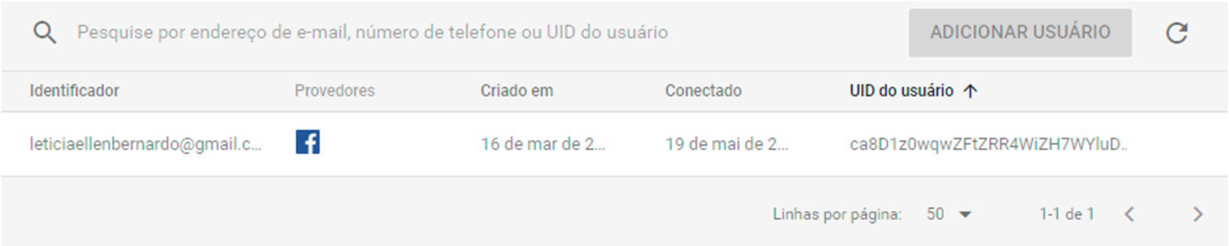
E por fim, sobrescreva os métodos *onStart()* e *onStop()*, colocando as verificações de estado de autenticação do Firebase. O método *onStart()* é executado quando a atividade está disponível para o usuário. O método *onStop()* é chamado quando a atividade é encerrada.

```
@Override
public void onStart() {
    super.onStart();
    mAuth.addAuthStateListener(mAuthListener);
}

@Override
public void onStop() {
    super.onStop();
    if (mAuthListener != null) {
        mAuth.removeAuthStateListener(mAuthListener);
    }
}
```

As autenticações do usuário podem ser vistas pelo painel do Firebase, no item do menu “*Authentication*”. A Figura 54 mostra uma lista de usuários que fizeram o login pelo Facebook com autenticação no Firebase.

Figura 54 – Listagem de usuários autenticados pelo Firebase.

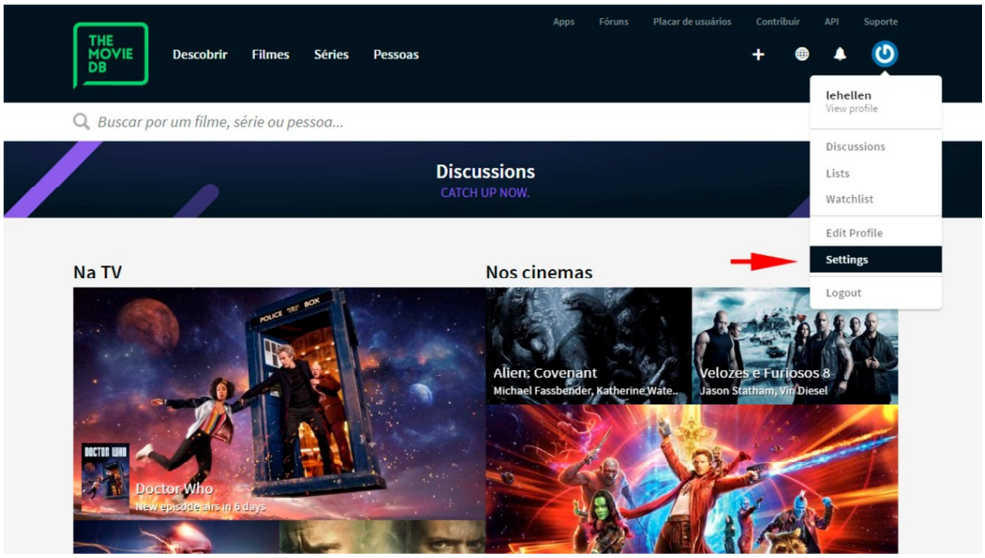


Fonte: Elaborado pelo autor

APÊNDICE G – Como utilizar a API TMDB

Para utilizar a API é necessário se registrar como um usuário na página do TMDB[37]. Efetue e o login como um usuário e busque a API de acesso ao TMDB seguindos os passos: Clique no menu "*Settings*" de sua conta. A Figura 55 apresenta como encontrar o link *Settings*.

Figura 55 – Como encontrar a chave da API do The Movie DB.



Fonte: Elaborado pelo autor

Clique no menu "API" do lado esquerdo e, logo depois, clique em "*Create*" para receber a chave da API. Neste tutorial, será utilizada a chave da API (v3 auth).

Seguidamente, o aplicativo será configurado para receber as requisições da API, utilizando JSON para retornar os filmes populares. Acessando a página de desenvolvedores do TMDB[38] é possível ver todos os tipos de requisições que a API oferece.

Para fazer uma requisição de filmes populares, por exemplo. Crie uma classe Java denominada *Movie* para utilizar as informações recebidas dos filmes.

Na classe *Movie* serão criados os atributos, o construtor, os *getters* e *setters*:

```
public class Movie {
    private int id;
    private String titulo;
```

[37] TMDB. Disponível em: <https://www.themoviedb.org>.
[38] TMDB Developers. Disponível em: <https://developers.themoviedb.org>.

```
    private Bitmap capa;

    public Movie(int id, String titulo, Bitmap capa) {
        this.id = id;
        this.titulo = titulo;
        this.capa = capa;
    }
    //adicione os getter e setters
}
```

Crie outra classe Java com o nome de *ConnectAPITMDB*, esta classe será responsável por todo o processo de requisição da API. Para receber o retorno da API TMDB será necessário trabalhar com *AsyncTask* para rodar o processamento de requisição da API em paralelo ao carregamento dos outros elementos da aplicação.

Estenda a classe *AsyncTask* na classe *ConnectApiTMDB* e sobrescreva o método *doInBackground()* que fará com que a *AsyncTask* rode em paralelo:

```
public class ConnectAPITMDB extends AsyncTask<String, Void,
ArrayList<Movie>> {

    @Override
    public ArrayList<Movie> doInBackground(String... params) {

        return null;

    }
}
```

Os parâmetros da *AsyncTask* correspondem a:

```
AsyncTask<Parâmetros,Progresso,Resultado>
```

- Os parâmetros são o tipo de dados de qualquer valor que serão passados para os métodos da classe;

- Progresso é o tipo de dado que mostrará o progresso na tela, por exemplo, a porcentagem de download;

- O Resultado é o retorno da *Thread* de processamento para a *Thread* principal que será trabalhado para ser exibido na interface do usuário.

O método *doInBackground* é obrigatório e é o responsável por todo o processamento pesado, pois ele é executado em uma *Thread* separada. Seu retorno é passado por parâmetro para o método *onPostExecute*;

O método *doInBackground* se encarregará do processamento do retorno da REST API TMDB.

A fim de evitar erros inesperados para o usuário, faça as operações dentro de um *try/catch.* Defina uma String com a URL no qual fará uma requisição para o TMDB, será necessária a conversão para URI e para construir a URI utilizando .*buildUpon()* e finalmente *build()* e retornar uma nova URL.

Em seguida, será utilizada a classe *HttpURLConnection*. Esta classe dá suporte para recursos específicos do protocolo HTTP.

O método *openConnection()* envia uma requisição da URL por meio do método GET e se conecta a página.

Através do *InputStream* retorna os dados em formato de bytes. O *BufferedReader* é uma classe que concatena os diversos chars para formar uma String através do método *readLine()*. O *StringBuffer* é uma String sincronizada que recebe as Strings e concatena através do método *append().*

Após o processamento o buffer é passado para a String *movieJsonStr* e fecha a conexão.

O próximo passo é converter a String para um objeto JSON, para obter os resultados dos filmes.

Um exemplo de código do método *doInBackground()*:

```
@Override
public ArrayList<Movie> doInBackground(String... params) {

    String apiKey = "colque_sua_chave_da_api";
    HttpURLConnection urlConnection = null;
    BufferedReader reader = null;
    String movieJsonStr = null;

    try {
        final String BASE_URL =
"https://api.themoviedb.org/3/movie/popular?api_key=" + apiKey +
"&language=pt-BR&page=1";

        URL url = new URL(BASE_URL);

        urlConnection = (HttpURLConnection) url.openConnection();
        urlConnection.setRequestMethod("GET");
        urlConnection.connect();

        InputStream inputStream = urlConnection.getInputStream();
        StringBuffer buffer = new StringBuffer();
        if (inputStream == null) {
            // Nothing to do.
            return null;
        }
        reader = new BufferedReader(new
InputStreamReader(inputStream));
```

```
            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line + "\n");
            }

            if (buffer.length() == 0) {
                return null;
            }

            movieJsonStr = buffer.toString();


        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("PlaceholderFragment", "Error closing stream",
e);
                }
            }
        }

        try {
            return getMoviesDataFromJson(movieJsonStr);
        }catch(JSONException e){
            e.printStackTrace();
            return null;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }
```

Crie um método *getMoviesDataFromJson()* para manipular os resultados e coloque como retorno um *ArrayList* de Filmes, já adicione também as exceções do JSON e IOException e como parâmetro passe os resultados obtidos da String movieJsonStr por parâmetro.

```
private ArrayList<Movie> getMoviesDataFromJson(String movieJsonStr)
throws JSONException, IOException {
}
```

Dentro do método, inicie as variáveis locais que armazenaram os conteúdos.

```
int id;
String title;
String posterPath;
```

```
Bitmap posterBitmap;
ArrayList<Movie> resultFilmes = new ArrayList<>();

final String OWN_RESULTS = "results";
final String OWN_ID = "id";
final String OWN_TITLE = "title";
final String OWN_POSTER = "poster_path";
```

Agora, converta a String de filmes para um objeto em JSON para conseguir capturar os resultados. E a partir do objeto JSON, converta-o em um *JsonArray* guardando os resultados na variável final *OWN_RESULTS*.

```
JSONObject movieJson = new JSONObject(movieJsonStr);
JSONArray movieArray = movieJson.getJSONArray(OWN_RESULTS);
```

Em seguida, será lido o objeto JSON a fim de carregar as informações de cada filme para a classe *Movie*. Para que o sistema possa ler a imagem fornecida pela API, ela será transformada em Bitmap. A API retorna no JSON apenas o ID da imagem, necessitando a construção do link completo para que possamos acessa-la. O link padrão das fotos do TMDB é “https://image.tmdb.org/t/p/w160/idDaImagem”. Por fim, faça uma condição para checar se a imagem existe, a fim de evitar erros ao buscar a imagem, então, adicione o filme ao *ArrayList* de filmes e ao final retorne os resultados.

```
for (int i = 0; i < 20; i++) {

    JSONObject movieObject = movieArray.getJSONObject(i);

    id = movieObject.getInt(OWN_ID);
    title = movieObject.getString(OWN_TITLE);
    posterPath = movieObject.getString(OWN_POSTER);

    Log.d("Filme: ", "ID: " + id + " | Título: " + title + " |
Poster: " + posterPath);

    if (!posterPath.contains("null")) {
        posterBitmap = getBitmapFromUrl(new
URL("https://image.tmdb.org/t/p/w160" + posterPath));
        resultFilmes.add(new Movie(id, title, posterBitmap));
    } else {
        resultFilmes.add(new Movie(id, title, null));
    }
}
return resultFilmes;
```

O código completo do método *getMoviesDataFromJson():*

```
private ArrayList<Movie> getMoviesDataFromJson(String movieJsonStr)
throws JSONException, IOException {

    int id;
    String title;
    String posterPath;
    Bitmap posterBitmap;
    ArrayList<Movie> resultFilmes = new ArrayList<>();

    final String OWN_RESULTS = "results";
    final String OWN_ID = "id";
    final String OWN_TITLE = "title";
    final String OWN_POSTER = "poster_path";

    JSONObject movieJson = new JSONObject(movieJsonStr);
    JSONArray movieArray = movieJson.getJSONArray(OWN_RESULTS);

    for (int i = 0; i < 20; i++) {

        JSONObject movieObject = movieArray.getJSONObject(i);

        id = movieObject.getInt(OWN_ID);
        title = movieObject.getString(OWN_TITLE);
        posterPath = movieObject.getString(OWN_POSTER);

        Log.d("Filme: ", "ID: " + id + " | Título: " + title + " |
Poster: " + posterPath);

        if (!posterPath.contains("null")) {
            posterBitmap = getBitmapFromUrl(new
URL("https://image.tmdb.org/t/p/w160" + posterPath));
            resultFilmes.add(new Movie(id, title, posterBitmap));
        } else {
            resultFilmes.add(new Movie(id, title, null));
        }
    }
    return resultFilmes;
}
```

Crie o método *getBitmapFromURL()* passando como parâmetro a URL da imagem. Este método tem como objetivo se conectar em uma URL de imagem, baixá-la utilizando um buffer e a decodificar como um Bitmap. A seguir, o código deste método:

```
private Bitmap getBitmapFromUrl(URL url){
    Bitmap bitmap = null;
    try {
        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
        connection.setDoInput(true);
        connection.connect();
        InputStream input = connection.getInputStream();
        bitmap = BitmapFactory.decodeStream(input);
    }catch (Exception e){
        e.printStackTrace();
    }
```

```
        return bitmap;
    }
```

Agora esta classe receberá os dados do TMDB.

Para utilizar os dados em uma *Activity*. Crie uma *Activity* ou abra a *Activity* na classe Java desejada, neste exemplo criaremos uma *Activity* denominada *PopularMoviesActivity*. Crie também uma interface e dê o nome *onTaskComplete*.

Para criar uma interface, clique com o botão direito no pacote, vá em: New > Java Class e na caixa, mude o tipo para Interface. E adicione a interface *OnTaskComplete* para fazer o retorno.

```
public interface OnTaskComplete {
    void OnTaskCompleted(ArrayList<Movie> result);
}
```

Implemente a interface OnTaskComplete a sua classe Java, crie um *ArrayList* de filmes e o método *onTaskCompleted*.

```
public class PopularMoviesActivity extends AppCompatActivity
implements OnTaskComplete {

    ArrayList<Movie> filmes = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_popular_movies);

        ConnectAPITMDB connectApiTMDB = new ConnectAPITMDB(this);
        connectApiTMDB.execute();
    }
}
```

Na classe *ConnectAPITMDB* adicione como atributo a classe *OnTaskComplete* e adicione um construtor passando como parâmetro a *OnTaskComplete*:

```
private OnTaskComplete callback;

public ConnectAPITMDB(OnTaskComplete onTaskComplete){
    callback = onTaskComplete;
}
```

Adicione também o método *onPostExecute* para retornar os resultados a interface *OnTaskComplete*:

```
@Override
protected void onPostExecute(ArrayList<Movie> result){
    callback.OnTaskCompleted(result);
    Log.d("AsyncTask", "Entrou no PostExecute");
    super.onPostExecute(result);
}
```

Na Activity *PopularMoviesActivity*, dentro do método *OnTaskCompleted*, percorra o *ArrayList* de filmes.

```
@Override
public void OnTaskCompleted(ArrayList<Movie> result) {
    filmes = result;

    for(int i=0;i<filmes.size();i++){
        System.out.println(filmes.get(i).getTitulo());
    }

}
```

Teste o aplicativo. Note pelo console do Android Studio que os filmes são listados. Para mostrar os filmes na *View*, o desenvolvedor pode carregar os filmes com uma *ListView* personalizada, *CardView*, entre outros.