2024-2

CINEMASTER: APLICAÇÃO DE UM ALGORITMO DE RECOMENDAÇÃO DE FILMES EM UM APLICATIVO MOBILE

José Vitor Tamarindo
Murilo Vinicius dos Santos

Orientador
Prof. Djalma Domingos da Silva

São José do Rio Preto 2024

2024-2

FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DO RIO PRETO

CINEMASTER: APLICAÇÃO DE UM ALGORITMO DE RECOMENDAÇÃO DE FILMES EM UM APLICATIVO MOBILE

José Vitor Tamarindo
Murilo Vinicius dos Santos

Trabalho de Graduação apresentado à Faculdade de Tecnologia de São José do Rio Preto como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Djalma Domingos da Silva.

2024-2

São José do Rio Preto 2024

2024-2

FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DO RIO PRETO

Banca Examinadora

Menção: _____ em _____/_____/_____

Nome:_____ Assinatura: _____

Nome:_____ Assinatura: _____

Nome:_____ Assinatura: _____

São José do Rio Preto 2024

RESUMO

Este trabalho tem como objetivo desenvolver um aplicativo mobile que utiliza um algoritmo de recomendação de filmes, proporcionando uma experiência personalizada aos usuários. Em um cenário onde o número de títulos disponíveis nas plataformas de streaming cresce exponencialmente, encontrar algo que atenda às preferências individuais pode se tornar uma tarefa exaustiva e desmotivadora. Motivado por essa realidade, o projeto busca oferecer uma solução prática para otimizar o tempo dos usuários, garantindo que o investimento de horas assistindo a um filme seja direcionado a algo que realmente agrade. Através da combinação de técnicas de filtragem colaborativa e baseada em conteúdo, o aplicativo sugere filmes que correspondem aos gostos e preferências individuais dos usuários, facilitando a descoberta de novos títulos e otimizando o tempo de escolha. Além disso, o aplicativo agrega catálogos de filmes de diversas plataformas de streaming, permitindo que os usuários encontrem filmes disponíveis em diferentes serviços de forma centralizada, eficiente e prática.

Palavras-chave: Recomendação de filmes. Algoritmo de filtragem colaborativa. Aplicativo *mobile*. *Streaming*. Personalização.

ABSTRACT

This work aims to develop a mobile application that utilizes a movie recommendation algorithm, providing users with a personalized experience. By combining collaborative filtering and content-based techniques, the application suggests movies that align with users' individual tastes and preferences, facilitating the discovery of new titles and optimizing the selection time. Additionally, the application aggregates movie catalogs from various streaming platforms, allowing users to easily find movies available on different services in a centralized, efficient, and practical manner.

Keywords: *Movie recommendation. Collaborative filtering algorithm. Mobile application. Streaming. Personalization. Data science.*

SUMÁRIO

1. Introdução 12
2. Justificativa 14
3. Objetivos 15
4. Fundamentação Teórica 16
5. Aprendizado de Máquina 17
5.1. Métodos de Recomendação 17
5.1.1. Filtragem Colaborativa 18
5.1.2. Filtragem Baseada em Conteúdo 19
5.1.3. Filtragem Híbrida 20
5.2. *Cold Start* 21
5.3. *Alternating Least Squares* 23
5.4. Métricas 24
5.4.1. *Mean Absolute Error* (MAE) 25
5.4.2. Root Mean Squared Error (RMSE) 25
6. Trabalhos Similares 26
6.1. Análise e comparação de métodos de recomendações e seus coeficientes para sistemas de recomendação de filmes 26
6.2. Estudo de sistemas de recomendação de filmes utilizando algoritmos híbridos 26
6.3. Sistemas de recomendação usando o *software* R 27
7. Metodologia 28
7.1. Metodologia de Pesquisa 28
7.2. Ambiente de Desenvolvimento 29
7.2.1. *Google Colab* 29
7.2.2. *Python* 29
7.2.3. Conjunto de Dados 29
7.2.3.1. Arquivos *ratings* 30
7.2.3.2. Arquivo *movies* 30
7.3. Desenvolvimento do Aplicativo *Mobile* 30
8. Desenvolvimento 32
8.1. Planejamento do Sistema 32
8.1.1. Objetivos do Planejamento 32
8.1.2. Ferramentas e Tecnologias Escolhidas 33
8.1.3. Arquitetura Geral do Sistema 34
8.1.4. Fundamentação e Referências 35
8.2. Estrutura do Projeto 35
8.2.1. Organização do Repositório 35
8.2.2. Configuração Inicial 36
8.2.3. Configuração do *main.dart* 37
8.2.4. Telas e *Widgets* 38
8.2.5. Integração com o *Backend* e APIs 38
8.2.6. Gerenciamento de Estado 39
8.2.7. Fundamentação e Referências 39

8.3. Banco de Dados 39
8.3.1. Estrutura do Banco de Dados 39
8.3.2. Integração com o Sistema 40
8.3.3. Segurança e Controle de Acesso 41
8.4. Implementação do Algoritmo de Recomendação 42
8.4.1. Metodologia Adotada 42
8.4.2. Integração com a API do TMDB 43
8.4.3. Tratamento de Dados 44
8.4.4. Implementação do Algoritmo 44
8.4.5. Validação do Algoritmo 45
8.4.6. Fundamentação e Referências 46
8.5. Desenvolvimento do Aplicativo Móvel 46
8.5.1. Estruturação Geral do Aplicativo 47
8.5.2. Funcionalidades Principais 47
8.5.3. Integração com o Banco de Dados e Algoritmo 51
8.5.4. Testes e Resultados 52
8.6. Teste de Similaridade entre Usuários: Validação Prática 52
9. Resultados e Discussões 55
9.1. Resultados do Aplicativo Móvel 55
9.2. Resultados do Algoritmo de Recomendação 57
9.3. Resultados da Integração com o Banco de Dados e a API do TMDB 58
9.4. Discussão Geral 59
10. Conclusões 61
11. Referências 63

LISTA DE ABREVIATURAS

MAE = Mean Absolute Error

RMSE = Root Mean Squared Error

IMDb = Internet Movie Database

ALS = Alternating Least Squares

ML = Machine Learning

SVD = Singular Value Decomposition

CLARA = Clustering Large Applications

CSV = Comma Separated Values

TMDb = The Movie Database

iOS = iPhone Operating System

API = Application Programming Interface

HTTP = Hypertext Transfer Protocol

SQL = Structured Query Language

NoSQL = Not only SQL

ID = Identity

ms = milissegundo

LISTA DE FIGURAS

Figura 1 - Modelo de Recomendação 18
Figura 2 - Exemplo Filtragem Colaborativa 19
Figura 3 - Exemplo Filtragem Baseada em Conteúdo 20
Figura 4 - Exemplo Filtragem Híbrida 21
Figura 5 - Recomendação Prime Video 22
Figura 6 - Algoritmo ALS 23
Figura 7 - Matriz de fatoração de mínimos quadrados 24
Figura 8 - Estrutura do Repositório 36
Figura 9 - Trecho do Código do arquivo main.dart 37
Figura 10 - Código exemplo de chamada à API 38
Figura 11 - Estrutura do Banco de Dados no Firebase com um usuário de exemplo 40
Figura 12 - Código exemplo de Operação de Leitura e Escrita no Firebase 41
Figura 13 - Configurações de Controle de Acesso 42
Figura 14 - Código Exemplo da funcionalidade da API 43
Figura 15 - Lógica de Cálculo da Similaridade 45
Figura 16 - Código do Método para gerar Recomendações 45
Figura 17 - Estrutura Geral do Aplicativo 47
Figura 18 - Tela Inicial do Aplicativo 48
Figura 19 - Trecho do Código da Lógica de Recomendação 49
Figura 20 - Tela Roleta de Recomendações 49
Figura 21 – Tela da Watchlist do Usuário 50
Figura 22 - Tela de Perfil do Usuário 51
Figura 23 - Listas de Filmes na Watchlist dos Usuários 1 e 2 53
Figura 24 - Filme Escolhido após o primeiro giro da Roleta 54

LISTA DE TABELAS

Tabela 1 - Dependências do arquivo pubspec.yaml 36
Tabela 2 - Resultado das Métricas de Validação 46
Tabela 3 – Resultados dos Testes do Aplicativo 52
Tabela 4 - Resultados do Teste Prático do Algoritmo de Recomendação 54

LISTA DE EQUAÇÕES

Equação 1 – Mean Absolute Error 25
Equação 2 – Root Mean Squared Error 25

1. Introdução

A explosão na quantidade de informações disponíveis na era digital trouxe desafios significativos para os usuários, especialmente no que diz respeito à seleção de conteúdos relevantes entre uma vasta gama de opções. Esta situação é especialmente perceptível no mundo das plataformas de *streaming* de filmes, onde a extensa seleção de filmes pode tornar o processo de escolha uma tarefa demorada e, por vezes, exasperante. Nesse cenário específico, os sistemas de recomendação revelam-se uma solução promissora, fornecendo aos utilizadores sugestões personalizadas que se alinham com as suas preferências e interesses únicos.

Os sistemas de recomendação, originados de sistemas de recuperação de informação, foram aprimorados para se tornarem ferramentas indispensáveis em diversos domínios, como comércio eletrônico, redes sociais e serviços de *streaming*. Segundo Schons (2007), esses sistemas têm a capacidade de analisar grandes quantidades de dados e detectar padrões de comportamento dos usuários, resultando em recomendações precisas e pertinentes.

O objetivo deste projeto é criar um aplicativo móvel que combine métodos de filtragem colaborativa e baseados em conteúdo para fornecer recomendações personalizadas de filmes. O objetivo principal do aplicativo é simplificar o processo de localização de novos filmes, ao mesmo tempo que reduz o tempo que os usuários gastam procurando por títulos, proporcionando uma experiência de navegação mais ágil e cativante. Além disso, o aplicativo reunirá coleções de filmes de várias plataformas de *streaming*, permitindo aos usuários localizarem filmes sem esforço em diferentes serviços de *streaming* centralizados.

A decisão de focar neste tema é apoiada pela crescente popularidade dos serviços de *streaming* e pela necessidade de ferramentas que auxiliem os usuários a encontrarem conteúdo que atenda às suas preferências e necessidades. Com a abundância de filmes disponíveis nas plataformas digitais, pode ser difícil para os usuários encontrarem algo que desperte seu interesse. Segundo Melville, Mooney e Nagarajan (2002), uma aplicação com algoritmos de recomendação pode aumentar a satisfação do usuário ao oferecer sugestões relevantes, economizar tempo e melhorar a experiência geral de navegação.

A abordagem adotada neste estudo implicará a realização de uma revisão completa da literatura existente para determinar as técnicas de recomendação mais bem-sucedidas, que serão então implementadas utilizando princípios de engenharia de *software*. O processo envolverá a criação de modelos, a escrita de código e a integração de técnicas de recomendação selecionadas. A eficácia e eficiência do sistema de recomendação serão avaliadas através de testes, utilizando métricas específicas para garantir que os utilizadores recebam sugestões personalizadas e relevantes.

2. Justificativa

A escolha deste tema baseia-se na observação de que, com a popularização dos serviços de *streaming*, a disponibilização de conteúdos tem crescido exponencialmente, resultando em dificuldades para os usuários selecionarem filmes que correspondem aos seus interesses. A abundância de opções pode resultar em confusão e frustração, pois os usuários frequentemente perdem tempo procurando um programa ou filme específico que realmente desejam assistir. De acordo com Davidson et al. (2010), os sistemas de recomendação são essenciais para ajudar os usuários a navegarem de forma eficiente por esse enorme volume de informações, oferecendo sugestões que economizam tempo e melhoram a experiência geral de navegação.

Incorporar um algoritmo de recomendação em um aplicativo móvel pode aumentar muito a satisfação do usuário. Estudo realizado por Melville, Mooney e Nagarajan (2002) destaca que algoritmos de filtragem colaborativa e baseados em conteúdo são eficazes na personalização de recomendações, orientando os usuários para sugestões que se alinham melhor com suas preferências. Ao fornecer recomendações personalizadas, o aplicativo não apenas maximiza o tempo dos usuários, mas também aumenta as chances de descobrir novos conteúdos que se alinham com suas preferências.

Além disso, este projeto permite a implementação prática dos conhecimentos adquiridos no curso de Análise e Desenvolvimento de Sistemas. O desenvolvimento de um aplicativo móvel que incorpore técnicas de recomendação requer uma série de habilidades, como programação, ciência de dados e engenharia de *software*. Segundo Schons (2007), a combinação de diversas áreas do conhecimento é crucial para a formação integral de profissionais que estejam prontos para o mercado de trabalho.

3. Objetivos

O objetivo principal deste trabalho é desenvolver uma aplicação móvel que integre um algoritmo de recomendação de filmes, oferecendo uma experiência personalizada para os usuários. Por meio de uma combinação de técnicas de filtragem colaborativa e baseada em conteúdo, o aplicativo sugere filmes que correspondem aos gostos e preferências individuais dos usuários, facilitando a descoberta de novos títulos e otimizando o tempo de escolha.

Os objetivos específicos deste trabalho incluem:

- Implementar um sistema de recomendação eficiente: Utilizar algoritmos de filtragem colaborativa e baseada em conteúdo para sugerir filmes relevantes aos usuários, melhorando a precisão e a relevância das recomendações.
- Desenvolver uma interface amigável e intuitiva: Criar uma interface de usuário que permita uma navegação fácil e eficiente, garantindo que os usuários possam encontrar filmes de forma rápida e conveniente.
- Integrar catálogos de filmes de diferentes plataformas de *streaming*: Adicionar catálogos de diferentes serviços de *streaming*, permitindo aos usuários encontrarem filmes disponíveis em diversas plataformas de forma centralizada.

Este conjunto de objetivos procura não só criar uma ferramenta prática e eficiente para os utilizadores de serviços de *streaming*, mas também avançar o conhecimento na área dos sistemas de recomendação, promovendo a inovação e a melhoria contínua das tecnologias utilizadas.

4. Fundamentação Teórica

Para compreender e avaliar os modelos sugeridos em sistemas de recomendação e aprendizado de máquina, é fundamental entender alguns conceitos fundamentais. O aprendizado de máquina é um campo que utiliza algoritmos para permitir que os sistemas identifiquem padrões nos dados, levando a um desempenho aprimorado com o passar do tempo. No domínio dos sistemas de recomendação, os modelos de recomendação são ferramentas que preveem as preferências ou comportamentos dos usuários.

A filtragem colaborativa é uma técnica que recomenda itens aos usuários com base nas preferências de outros usuários com gostos semelhantes, enquanto a filtragem baseada em conteúdo sugere itens analisando as características dos itens e as preferências anteriores do usuário. A filtragem híbrida combina aspectos de filtragem colaborativa e filtragem baseada em conteúdo para aumentar a precisão das recomendações. O problema de inicialização a frio ocorre quando há informações limitadas disponíveis sobre um novo usuário ou item, dificultando o fornecimento de recomendações personalizadas.

Na última análise, as métricas de avaliação, incluindo o erro médio absoluto (MAE) e a raiz do erro quadrático médio (RMSE), são cruciais para avaliar a eficiência dos modelos, fornecendo uma base quantitativa para avaliar o desempenho em sistemas de recomendação

5. Aprendizado de Máquina

Apresentar uma metodologia de aprendizado de máquina como alternativa às abordagens tradicionais de engenharia é bastante útil. Na engenharia convencional, o processo de projeto começa com a aquisição de conhecimento do domínio. O problema é examinado minuciosamente e um modelo matemático é desenvolvido para representar com precisão os aspectos físicos da situação que está sendo comprovada. Este modelo serve de base para o desenvolvimento de um algoritmo otimizado, que garante desempenho, desde que o modelo baseado na física reflita com precisão as condições do mundo real (SIMEONE, 2018).

Segundo Russell e Norvig (2009), o aprendizado de máquina é o processo de construção de sistemas que podem melhorar seu desempenho por meio da experiência adquirida, aspecto fundamental da criação de sistemas inteligentes. Flach (2012) e Murphy (2012) discutem essa abordagem, enfatizando a importância de aprender com os dados e a capacidade dos algoritmos de generalizar padrões e fazer inferências sem intervenção direta do programador. Bishop (2006) enfatiza a importância de sistemas adaptativos que podem evoluir e melhorar suas capacidades ao longo do tempo.

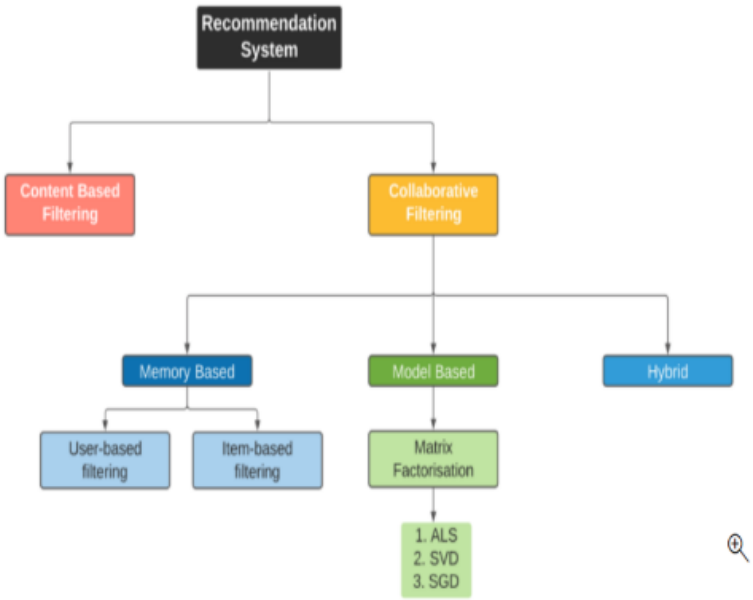
Neste contexto, o campo da aprendizagem automática desempenha um papel crucial na investigação em inteligência artificial, influenciando significativamente o desenvolvimento de sistemas mais eficientes e independentes.

5.1. Métodos de Recomendação

Desde o início do primeiro modelo de filtragem colaborativa na década de 1990, os sistemas de recomendação foram extensivamente pesquisados, implementados e melhorados em numerosos domínios acadêmicos e industriais. De acordo com Ko et al. (2022), sistemas de recomendação são algoritmos que sugerem itens personalizados aos usuários em um ambiente de serviço, podendo conter ou reunir diferentes tipos de dados. Para aumentar a satisfação do usuário com os sistemas de recomendação, é fundamental fornecer uma gama variada de opções, sugerindo vários itens ao usuário. Isto implica pesquisar os dados implícitos e explícitos dos usuários, bem como os dados agregados de um grupo de usuários (KO et al., 2022).

A Figura 1 fornece uma visão abrangente dos vários tipos de sistemas de recomendação, com foco na estratégia central de emprego neste trabalho: filtragem colaborativa. A figura também fornece informações adicionais sobre filtragem baseada em conteúdo e filtragem híbrida. A filtragem colaborativa emprega uma matriz com mínimos quadrados alternados, permitindo uma compreensão mais profunda dos modelos descritos a seguir.

Figura 1 - Modelo de Recomendação



Fonte: ALEXBUTGIT. et al (2023)

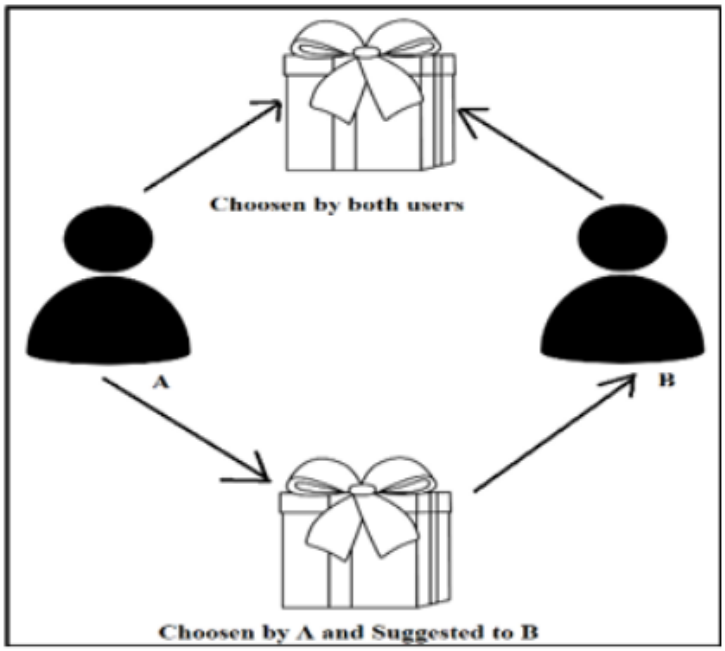
5.1.1. Filtragem Colaborativa

Os sistemas de recomendação de filtragem colaborativa são criados para fornecer recomendações a um usuário, considerando as opiniões de outros usuários. A sugestão é que, se dois indivíduos têm uma preferência em comum, é mais provável que compartilhem gostos semelhantes em relação a outros itens. Esses sistemas empregam técnicas de filtragem colaborativa para antecipar as preferências de um indivíduo por meio da análise dos interesses de outros usuários (SALUNKHE, 2023).

Os sistemas de filtragem colaborativa oferecem sugestões ou detalhes personalizados para um usuário específico sobre um ou mais itens de interesse. A gama de itens que podem ser incluídos nas recomendações é bastante diversificada, abrangendo livros, músicas, jogos ou até destinos de férias específicos. Os sistemas

de filtragem colaborativa podem ser classificados de várias maneiras (RESEARCHGATE, 2001).

Figura 2 - Exemplo Filtragem Colaborativa



Fonte: Salunkhe, T (2023)

A Figura 2 demonstra o processo de tomada de decisão dos usuários ao selecionar um presente. Considerando as escolhas comuns, uma recomendação posterior é fornecida com base nas escolhas anteriores. Além da filtragem colaborativa, existem métodos alternativos para sugerir itens, como a filtragem baseada em conteúdo, que será explicada com mais detalhes nas frases a seguir.

5.1.2. Filtragem Baseada em Conteúdo

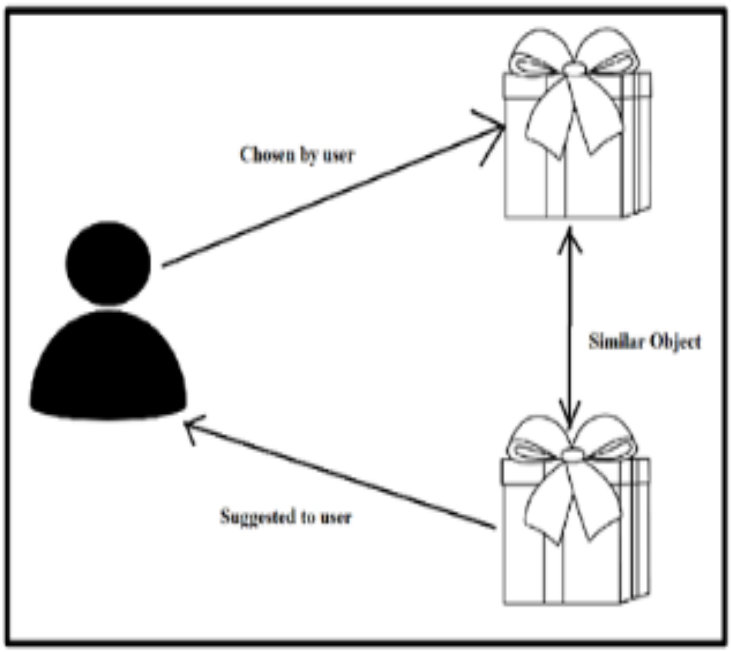
A filtragem baseada em conteúdo é um método usado para sugerir produtos a clientes ou usuários, considerando a semelhança entre os itens. Esse método é comumente empregado em sistemas de recomendação para identificar itens de interesse dos usuários, analisando o conteúdo dos itens e avaliando sua similaridade. Esse conteúdo pode abranger qualquer material textual pertinente à avaliação do produto (SALUNKHE, 2023).

Cada usuário tem seus interesses distintos e particulares, muitas vezes centrados em uma pequena parte do vasto conteúdo disponível na *internet*. Isso

torna o processo de encontrar informações interessantes e relevantes mais complicado e demorado. Para facilitar esse processo, a *internet*, também conhecida como *World Wide Web*, pode ser customizada por meio de sistemas de recomendação (RESEARCHGATE, 2001).

A Figura 3 ilustra uma situação em que um usuário seleciona um item inicial e, como resultado, um segundo item com características semelhantes é sugerido para o mesmo usuário. Esta abordagem difere daquela representada na Figura 2, que depende das decisões de vários usuários para fazer recomendações.

Figura 3 - Exemplo Filtragem Baseada em Conteúdo



Fonte: Salunkhe, T (2023)

5.1.3. Filtragem Híbrida

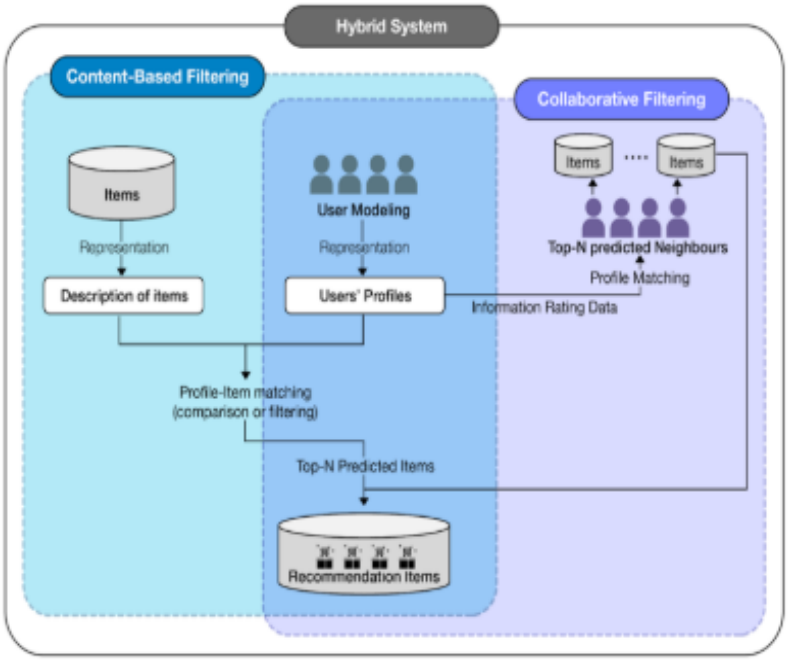
Ambos os modelos de filtragem têm certas limitações. A filtragem baseada em conteúdo depende fortemente de metadados vinculados a itens, enquanto a filtragem colaborativa depende fortemente de classificações dadas pelos usuários. Para abordar estas limitações e aumentar o impacto das recomendações, foi criado um modelo de recomendação híbrido (KO et al., 2022).

O modelo de recomendação híbrido visa principalmente resolver a questão dos dados limitados, compensando a ausência de informações de classificação

através da integração de dados provenientes da filtragem baseada em conteúdo e da filtragem colaborativa (KO et al., 2022).

A Figura 4 demonstra um sistema de recomendação híbrido que combina informações de duas bases de dados diferentes: uma contendo itens e outra contendo um grupo de usuários. Este modelo fornece sugestões tanto para usuários quanto para itens, diferenciando-se dos modelos representados nas Figuras 2 e 3. No entanto, esses modelos de recomendação enfrentam um desafio denominado partida a frio, que será explicado detalhadamente a seguir.

Figura 4 - Exemplo Filtragem Híbrida



Fonte: KO, H. et al (2022)

5.2. *Cold Start*

O problema referido como arranque a frio surge quando os sistemas de recomendação encontram novos utilizadores que não possuem informação pessoal suficiente ou um histórico de consumo substancial (SANTANA, M). Este cenário leva a uma imprecisão nas recomendações devido à falta de dados suficientes para gerar sugestões precisas. Este é um grande desafio para os sistemas de recomendação, que têm como objetivo oferecer sugestões com base nas avaliações dos usuários e nas vivências prévias.

Ao implementar um motor de recomendação híbrido que combina métodos de seleção colaborativos e baseados em conteúdo, a questão do arranque a frio pode ser abordada de forma eficaz (SALUNKHE, 2023). Ao configurar uma conta em plataformas como *Prime Video*, *Netflix* ou *Max*, o problema do *cold start* torna-se aparente, pois falta ao usuário um perfil de recomendação baseado em suas preferências pessoais. No início desta fase, a plataforma gera sugestões de recomendações para começar a mapear o perfil e as preferências do usuário com base no seu tempo de uso.

Figura 5 - Recomendação *Prime Video*

Fonte: Do autor (2024)

A Figura 5 ilustra como a plataforma de vídeo *Prime Video* utiliza essa estratégia de recomendação, atualizando o perfil do usuário histórico de preferências e oferecendo sugestões mais genéricas, como séries com alta audiência no IMDb ou as mais populares da plataforma. À medida que o tempo passa, o algoritmo pretende preencher a lacuna de conhecimento para oferecer sugestões mais precisas.

No entanto, ter uma compreensão clara destas estratégias de recomendação não é suficiente para compreender o seu pleno significado. A seguir, explicaremos como o *Apache Spark* pode ser utilizado em conjunto com o algoritmo de mínimos quadrados alternados (ALS) para implementar essas estratégias de forma eficiente.

5.3. *Alternating Least Squares*

Alternating Least Squares (ALS) é um método de filtragem colaborativa usado para resolver o problema de *overfitting* em dados esparsos. *Apache Spark* é a plataforma escolhida para implementação do algoritmo ALS (GOSH et al., 2021).

ALS também é um algoritmo de fatoração de matrizes executado em paralelo. Este algoritmo, implementado no *Apache Spark ML*, foi projetado para problemas de filtragem em larga escala. O ALS desempenha um papel crucial na resolução de desafios de escalabilidade e deficiência de classificação de dados por ser simples e escalável para conjuntos de dados muito grandes (LIAO, K).

Figura 6 - Algoritmo ALS

```
1  Procedure ALS(Xu, Yi)
2  Initialization Xu <- 0
3  Initialization matrix Yi with random values
4  Repeat
5      Fix Yi, solve Xu by minimizing's the objective
6      function(the sum of squared erros)
7      Fix Yi solve Yi by minimizing the objective
8      function similarly
9  Until reaching the maximum iteration
10 Return Xu, Yi
11 End procedure
```

Fonte: Do autor (2024)

A Figura 6 apresenta uma descrição genérica de como funciona o filtro colaborativo. Este filtro visa preencher as lacunas de uma matriz de associação usuário-item. Usuários e produtos são representados por um conjunto reduzido de fatores latentes que podem ser usados para prever entradas faltantes. Portanto, o algoritmo ALS é útil para aprender esses fatores latentes. A estratégia envolve pegar uma matriz extensa e fatorá-la para obter uma versão reduzida da matriz original, utilizando mínimos quadrados alternados. O resultado da aplicação do algoritmo é a obtenção de duas ou mais matrizes de dimensão reduzida cujo produto se aproxima da matriz original (WALTRICK, C).

Figura 7 - Matriz de fatoração de mínimos quadrados

Fonte: ALEXBUTGIT (2023)

Conforme ilustrado na Figura 7, em cada iteração do algoritmo, um dos fatores é mantido constante, enquanto o outro é focado no uso do método dos mínimos quadrados. Concluída esta etapa, a matriz de fator fixa é definida e a outra matriz de fator é submetida ao processo de resolução. Esta abordagem iterativa pretende reduzir o tamanho de ambas as matrizes fatoriais, conforme descrito por ALEXBUTGIT (2023).

Para garantir a eficiência do algoritmo, é crucial avaliar regularmente o seu desempenho utilizando métricas específicas, que serão discutidas na seção seguinte.

5.4. Métricas

A avaliação do sistema de recomendação é realizada por meio de uma análise *offline*. Nesta análise, não há interação direta com usuários reais; em vez disso, um vasto conjunto de dados é dividido em um conjunto de dados de treinamento e um conjunto de dados de teste. O sistema de recomendação é treinado usando o conjunto de dados de treinamento para prever as classificações dos usuários na plataforma de teste. A avaliação examina os resultados das medições em relação às classificações reais fornecidas no conjunto de dados. Para medir a precisão das previsões feitas pelo sistema de recomendação, são utilizadas métricas populares de precisão preditiva, como erro médio absoluto (MAE), raiz do erro quadrático médio (RMSE) e ganho médio do usuário (MUG) (GOSH et al., 2021).

5.4.1. *Mean Absolute Error* (MAE)

Erro Absoluto Médio (MAE) e Erro Quadrático Médio (RMSE) são medidas usadas para avaliar o desvio médio entre uma classificação prevista e a classificação real de um usuário, indicando a precisão do sistema de recomendação com base nas classificações reais dos usuários (HERLOCKER, J.).

Equação 1 – *Mean Absolute Error*

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Yi - Pi|$$

Onde:

- n é o número de amostras
- y é o valor observado para cada amostra
- p é o valor previsto pelo modelo para cada amostra
- | | representa o valor absoluto

5.4.2. *Root Mean Squared Error* (RMSE)

A Raiz do Erro Quadrático Médio (RMSE) é calculada de forma semelhante ao Erro Quadrático Médio (MSE), mas aplica a raiz quadrada para penalizar grandes diferenças entre os valores previstos e reais. Isso ajusta a unidade para que fique na mesma escala dos dados originais, proporcionando uma interpretação mais adequada dos resultados da métrica (JÚNIOR, C. de O.).

Equação 2 – *Root Mean Squared Error*

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Yi - Pi)^2}$$

Onde:

- n é o número de amostras
- Yi é o valor observado para a amostra i
- Pi é o valor previsto pelo modelo para a amostra i

6. Trabalhos Similares

6.1. Análise e comparação de métodos de recomendações e seus coeficientes para sistemas de recomendação de filmes

O artigo "Análise e comparação de métodos de recomendações e seus coeficientes para sistemas de recomendação de filmes", de Jessé Filipe de Queiroz Morais, Marco Túlio Alves Nolasco Rodrigues e Flávio Luis Cardeal Pádua (2018), tem como objetivo desenvolver uma análise comparativa entre diversos algoritmos de recomendação de filmes. O estudo utilizou a linguagem de programação Java e aplicou os algoritmos ao banco de dados *MovieLens*. Os métodos de recomendação foram avaliados utilizando métricas como RMSE (*Root Mean Square Error*) e MAE (*Mean Absolute Error*) para medir a precisão das previsões dos algoritmos.

Este trabalho destaca-se por sua abordagem detalhada e sistemática na comparação de diferentes algoritmos de recomendação. Os autores aplicaram metodologias rigorosas para testar e otimizar os parâmetros dos algoritmos, com o intuito de minimizar os erros de previsão. A escolha do banco de dados *MovieLens* proporciona uma base sólida e relevante para a avaliação dos métodos. A pesquisa conclui que a otimização dos parâmetros é crucial para melhorar a precisão das recomendações, fornecendo *insights* valiosos para futuras implementações de sistemas de recomendação.

6.2. Estudo de sistemas de recomendação de filmes utilizando algoritmos híbridos

O trabalho "Estudo de sistemas de recomendação de filmes utilizando algoritmos híbridos" apresenta uma análise detalhada de diferentes algoritmos de recomendação, incluindo métodos de filtragem colaborativa e baseada em conteúdo. Os autores, Pedro Chamberlain Matos e Marco Molinaro, desenvolveram um método híbrido combinando a fatoração de matrizes pela decomposição em valores singulares (SVD) com a filtragem baseada em conteúdo utilizando similaridade textual. O estudo utilizou dados de avaliação de filmes do *MovieLens* e demonstrou que o método híbrido apresentou melhor desempenho em comparação aos métodos individuais.

Este estudo é significativo por demonstrar a eficácia dos métodos híbridos de recomendação. A combinação de técnicas de filtragem colaborativa e baseada em conteúdo resultou em melhorias notáveis na precisão das recomendações. Os autores realizaram uma avaliação robusta dos algoritmos utilizando métricas padrão da área, como RMSE e MAE, e validaram seus resultados com dados reais do *MovieLens*. A pesquisa contribui para a literatura ao mostrar que a integração de diferentes abordagens pode superar as limitações de métodos isolados, tornando os sistemas de recomendação mais eficientes e precisos.

6.3. Sistemas de recomendação usando o *software* R

O artigo "Sistemas de recomendação usando o software R", de Leonardo Filgueira, Luciane Ferreira Alcoforado e Rodrigo Otávio de Araújo Ribeiro, investiga a aplicação de técnicas de filtragem colaborativa combinadas com métodos de clusterização. Utilizando a base de dados *MovieLens*, o estudo implementou algoritmos de clusterização CLARA e *k-means* para agrupar usuários e aplicar a filtragem colaborativa dentro de cada *cluster*. O objetivo foi melhorar a acurácia das recomendações e verificar a eficiência dos métodos de clusterização na redução do tempo de execução das recomendações.

Este trabalho é notável por explorar a integração de técnicas de clusterização com filtragem colaborativa para aprimorar a precisão das recomendações de filmes. A abordagem metodológica inclui testes rigorosos para avaliar a eficácia das técnicas de clusterização e a precisão das recomendações resultantes. A pesquisa revela que a utilização de *clusters* pode melhorar significativamente a acurácia das recomendações, especialmente em cenários com grandes volumes de dados. A análise detalhada das métricas de desempenho e a validação com dados reais do *MovieLens* tornam este estudo uma contribuição valiosa para a área de sistemas de recomendação.

7. Metodologia

Planejamos empregar a filtragem colaborativa como método de avaliação de itens com base nas opiniões de outros usuários, amplamente considerado como uma das abordagens mais eficazes para o desenvolvimento de sistemas de recomendação. Este método funcionará com base nas preferências únicas de cada usuário e item, fornecendo recomendações personalizadas. O objetivo principal será completar as entradas faltantes em uma matriz que represente a associação entre usuários e itens. Pretendemos utilizar a biblioteca *Spark ML* para facilitar a filtragem colaborativa baseada em modelo, onde usuários e produtos serão representados por um conjunto limitado de fatores latentes para prever entradas ausentes.

Adicionalmente, a abordagem de pesquisa escolhida será descritiva e exploratória. Este método aprimorará a compreensão dos dados coletados, oferecendo uma análise mais completa e sólida para a interpretação dos resultados obtidos, que serão discutidos nos tópicos subsequentes.

7.1. Metodologia de Pesquisa

A abordagem de investigação que utilizaremos neste estudo será descritiva e exploratória, em consonância com os objetivos traçados por Gil (2010). Esta decisão será baseada na proximidade da questão com a que está sendo investigada, possibilitando a criação de hipóteses diretamente relacionadas. Nosso objetivo é coletar informações por meio de pesquisas em livros e documentos, empregando métodos quantitativos e qualitativos para analisar e interpretar os dados.

Além disso, a seção a seguir apresentará os principais componentes do ambiente de desenvolvimento do sistema. Esta formação técnica será crucial para a compreensão e implementação com sucesso das soluções propostas, criando uma ligação entre o enquadramento teórico e a implementação prática do sistema em desenvolvimento.

7.2. Ambiente de Desenvolvimento

Pretendemos empregar diversas ferramentas e tecnologias para construir o sistema, selecionando cada uma com base em sua capacidade de atender aos requisitos exclusivos do projeto.

7.2.1. Google Colab

Inicialmente, utilizaremos o *Google Colab* devido à sua capacidade de executar várias linhas de código em células separadas, o que simplifica a organização e o teste de pequenos trechos de código. Essa configuração permite a visualização rápida dos resultados, o que é especialmente benéfico para análises exploratórias e testes de algoritmos.

7.2.2. Python

Selecionaremos *Python* para desenvolvimento devido à sua extensa coleção de pacotes projetados especificamente para ciência de dados e aprendizado de máquina, incluindo *NumPy*, *SciPy*, *Statsmodels*, *Scikit-learn* e *Pandas*. Essas bibliotecas oferecem ferramentas abrangentes para manipulação de dados, construção de modelos e realização de análises estatísticas, cruciais para o desenvolvimento e validação de algoritmos de recomendação.

7.2.3. Conjunto de Dados

Planejamos utilizar o conjunto de dados oferecido pelo *GroupLens*, que foi coletado e disponibilizado no site do *MovieLens*, para esta implementação. A seleção deste conjunto de dados baseia-se na sua acessibilidade para fins de pesquisa e desenvolvimento, bem como na sua história de sucesso no contexto das recomendações de filmes.

O *GroupLens* é um centro de pesquisa localizado no departamento de Ciência da Computação e Engenharia da Universidade de Minnesota, focado em sistemas de recomendação, comunidades *online*, tecnologias móveis e onipresentes, bibliotecas digitais e sistemas de informação geográfica.

O *MovieLens* contém um vasto conjunto de dados com 20 milhões de registros de classificação, abrangendo mais de 50.000 avaliações e envolvendo cerca de 200.000 usuários, no período de 1995 a 2018. Planejamos utilizar os arquivos *ratings.csv*, que contêm as classificações de filmes, e *movies.csv*, que contêm informações supervisionadas sobre os filmes, para realizar nossa análise e desenvolvimento do sistema de recomendação.

7.2.3.1. Arquivos *ratings*

Todas as classificações atribuídas são documentadas no arquivo *ratings.csv*. Cada linha neste arquivo, após o cabeçalho, representa a classificação de um filme pelo usuário, no formato: *userid*, *movieid*, *rating*, *timestamp*. As linhas são organizadas na ordem de *userid* e, em seguida, por *movieid*. As classificações são baseadas nas pontuações fornecidas por cada usuário, permitindo uma avaliação abrangente das preferências do usuário e auxiliando no desenvolvimento de modelos de recomendação precisos.

7.2.3.2. Arquivo *movies*

As informações sobre os filmes serão armazenadas em um arquivo CSV. Cada linha deste arquivo, após o cabeçalho, representa um filme e segue o formato: *movieid*, título, gêneros. Os títulos dos filmes serão inseridos manualmente ou importados do site TMDB ('*The Movie Database*', 2023), incluindo o ano de lançamento entre parênteses. Pode haver erros e discrepâncias nestes títulos. Os gêneros serão listados separados por vírgulas e estarão disponíveis em categorias como ação, aventura, animação, infantil, comédia, crime, documentário, drama, fantasia, musical, mistério, suspense, guerra, romance, ficção científica e gênero não especificado.

7.3. Desenvolvimento do Aplicativo *Mobile*

Utilizaremos *Flutter* para desenvolver o aplicativo móvel, permitindo a criação de aplicativos nativos para *iOS* e *Android* a partir de uma única base de código. A linguagem *Dart* será empregada por sua eficiência e facilidade de uso. O *Firebase*

servirá como plataforma para serviços cruciais como autenticação, bancos de dados em tempo real e armazenamento. Além disso, incorporaremos APIs de plataformas de *streaming* como *Netflix* e *Amazon Prime Video*, bem como utilizaremos a API TMDB para coletar dados abrangentes sobre filmes.

8. Desenvolvimento

Este capítulo descreve o processo de criação do sistema de sugestão de filmes Cinemaster, detalhando cada fase do projeto. A meta é detalhar a utilização das tecnologias e metodologias selecionadas para cumprir os objetivos estabelecidos, desde o planejamento inicial até a implementação e validação do sistema. A estrutura do desenvolvimento é segmentada em subcapítulos que discutem o planejamento, a execução do algoritmo de recomendação, a criação do aplicativo móvel e a conexão entre esses elementos, culminando em testes e avaliações de performance.

Durante o capítulo, são enfatizadas as decisões técnicas, as soluções postas em prática e os obstáculos superados, oferecendo uma perspectiva completa do processo de desenvolvimento do sistema. Este detalhamento tem como objetivo não só validar as escolhas feitas, mas também fornecer um alicerce robusto para a análise e avaliação crítica dos resultados alcançados.

8.1. Planejamento do Sistema

O sistema foi planejado para satisfazer as demandas de um sistema integrado, formado por um algoritmo de recomendação e um aplicativo para dispositivos móveis. Este procedimento implicou na definição de metas precisas, na seleção criteriosa de ferramentas e tecnologias adequadas, e na criação de uma arquitetura eficaz e expansível. O planejamento foi fundamentado em princípios teóricos e práticos, visando assegurar que cada fase do desenvolvimento ocorresse de maneira organizada e em conformidade com os requisitos do projeto.

8.1.1. Objetivos do Planejamento

A meta principal do planejamento foi criar um sistema de sugestão de filmes que proporcionasse uma experiência individualizada e unificada para os utilizadores. Este sistema não só deveria simplificar a descoberta de novos títulos, como também incorporar informações pertinentes provenientes de diversos serviços de *streaming*. Assim, definiu-se o objetivo de desenvolver um algoritmo de recomendação que utiliza técnicas de filtragem colaborativa, um método

amplamente comprovado pela sua eficiência em antecipar as preferências dos usuários. Métricas como MAE (*Mean Absolute Error*) e RMSE (*Root Mean Squared Error*) foram estabelecidas como indicadores de desempenho do algoritmo, garantindo a precisão das recomendações e a adequação às preferências individuais dos usuários (HERLOCKER et al., 2004).

Adicionalmente, o planejamento implicou na integração do sistema com a API do TMDB (*The Movie Database*), famosa pela abundância de informações atualizadas sobre filmes. Esta integração teve como objetivo melhorar a qualidade e a pertinência das sugestões, fornecendo detalhes sobre os filmes, tais como sinopses, gêneros e avaliações. Simultaneamente, estabeleceu-se como prioridade a implementação de um aplicativo multiplataforma para assegurar a acessibilidade e usabilidade do sistema em aparelhos *Android* e *iOS*, empregando tecnologias avançadas que garantissem uma experiência suave e responsiva.

8.1.2. Ferramentas e Tecnologias Escolhidas

Para satisfazer as necessidades do projeto, foram escolhidos instrumentos e tecnologias que se ajustassem mais adequadamente às exigências específicas do sistema. As decisões foram fundamentadas em uma avaliação minuciosa do repositório no *GitHub* e nos recursos técnicos disponíveis. As tecnologias predominantes empregadas foram:

- *Flutter*: Este *framework* foi selecionado para a criação do aplicativo por sua habilidade em desenvolver interfaces responsivas e contemporâneas, além de produzir aplicativos nativos de alta performance para *Android* e *iOS*. A sua adaptabilidade e a capacidade de personalizar *widgets* foram cruciais para o desenvolvimento de uma interface intuitiva.
- *Dart*: Sendo a linguagem de programação do *Flutter*, o *Dart* ofereceu um ambiente eficaz para a criação do *frontend*, possibilitando alterações dinâmicas e ágeis no código do aplicativo.
- *Firebase*: Empregado para administrar a autenticação de usuários e o armazenamento em tempo integral. A sincronização dos dados dos usuários,

incluindo preferências e históricos, com o *backend* do sistema foi facilitada pelo *Firebase*.

- API do TMDB: A conexão com essa API foi crucial para aprimorar o conteúdo oferecido aos usuários, oferecendo detalhes sobre os filmes e assegurando a atualização constante do catálogo.

8.1.3. Arquitetura Geral do Sistema

A estrutura do sistema foi concebida para assegurar modularidade, capacidade de expansão e eficácia na elaboração e manutenção do projeto. O sistema foi estruturado em três níveis principais, cada um com funções claramente estabelecidas e integradas de maneira harmônica.

A camada inicial, que corresponde ao algoritmo de recomendação, foi desenvolvida para analisar as informações recolhidas dos usuários e produzir recomendações personalizadas com base em seus gostos. O algoritmo empregou métodos de filtragem colaborativa, um método que reconhece padrões de correspondência entre usuários e itens para antecipar interesses futuros. Esta camada foi projetada com o objetivo de escalar, assegurando que o sistema pudesse gerir um grande volume de dados sem prejudicar o rendimento.

A implementação da segunda camada, o *backend*, foi feita com o *Firebase*, uma plataforma reconhecida pela sua eficácia na gestão de dados em tempo real e na autenticação de usuários. Esta camada teve a função de unificar as informações do sistema, tais como os registros de navegação e as preferências dos usuários, garantindo também uma comunicação eficaz entre o *frontend* e o algoritmo de sugestão.

Finalmente, o *Flutter* desenvolveu a camada de interface com o usuário, focando na criação de uma experiência visual e funcional de excelente qualidade. A interface foi concebida para ser intuitiva e de fácil acesso, possibilitando que os usuários percorram os catálogos de filmes, recebam sugestões personalizadas e visualizem detalhes sobre os filmes de maneira prática e eficaz. A utilização de *widgets* customizáveis do *Flutter* possibilitou uma personalização visual alinhada às necessidades do projeto.

8.1.4. Fundamentação e Referências

O plano foi fundamentado em pesquisas e práticas respeitadas no campo dos sistemas de recomendação. A conexão com a API do TMDB seguiu as orientações técnicas presentes na documentação oficial. As métricas de avaliação empregadas foram baseadas em Herlocker et al. (2004), que ressaltam a relevância de critérios quantitativos na validação de sistemas de recomendação.

Com essas decisões bem embasadas e em sintonia com as metas do projeto, o planejamento estabeleceu uma fundação robusta para as fases subsequentes de desenvolvimento, possibilitando a implementação eficaz e unificada do sistema.

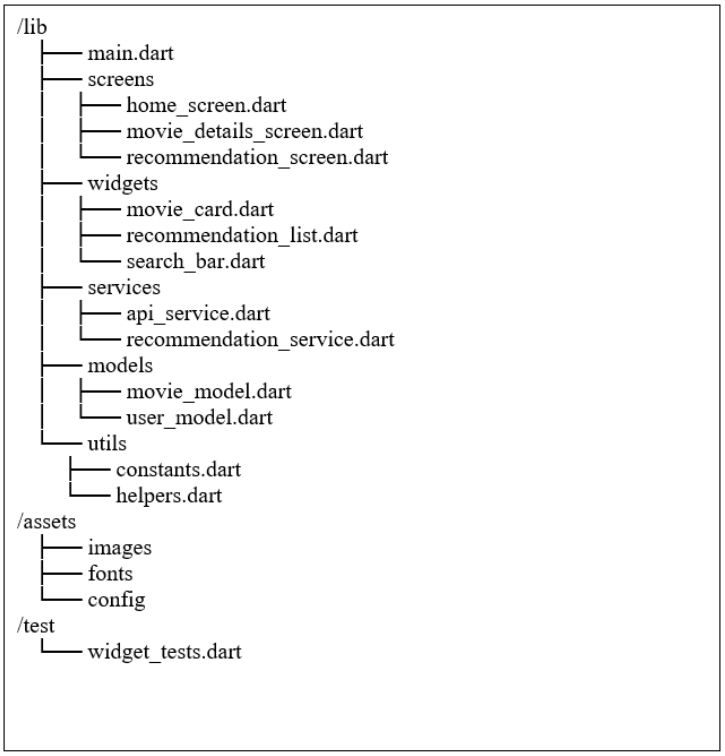
8.2. Estrutura do Projeto

Este capítulo descreve a estrutura do projeto Cinemaster, tratando da disposição do repositório, da configuração inicial e dos arquivos e pastas fundamentais do sistema. A estrutura foi criada de maneira modular, em conformidade com as melhores práticas sugeridas pela comunidade de desenvolvimento em *Flutter*, assegurando escalabilidade e simplicidade na manutenção.

8.2.1. Organização do Repositório

A estruturação do repositório foi planejada para garantir uma navegação intuitiva e clara para desenvolvedores e funcionários. A estrutura básica do repositório é mostrada a seguir:

Figura 8 - Estrutura do Repositório



Fonte: Do autor (2024)

Esta organização divide os elementos do projeto em pastas separadas para telas (*screens*), *widgets* reutilizáveis (*widgets*), serviços de *backend* (*services*), modelos de dados (*models*) e utilitários gerais (*utils*). A pasta *assets* contém recursos permanentes, tais como imagens e configurações de tipos de letra.

8.2.2. Configuração Inicial

O projeto foi configurado inicialmente no arquivo *pubspec.yaml*, onde foram especificadas as dependências necessárias para a criação do aplicativo. A Tabela 1 apresenta o trecho do arquivo.

Tabela 1 - Dependências do arquivo *pubspec.yaml*

Dependência	Versão	Descrição
flutter	^ 3.3.0	Framework principal para desenvolvimento mobile
http	^ 0.15.0	Realização de requisições HTTP
provider	^ 6.0.3	Gerenciamento de estado

firebase_auth	^ 3.4.0	Autenticação com Firebase
firebase_database	^ 9.1.0	Banco de dados em tempo real do Firebase
flutter_dotenv	^ 5.0.2	Gerenciamento de variáveis de ambiente

Fonte: Do autor (2024)

Essas ligações foram cruciais para a implementação de funções como o uso de APIs, a autenticação de usuários e o armazenamento de informações.

8.2.3. Configuração do *main.dart*

O arquivo *main.dart* representa o início do programa. Ele foi ajustado para iniciar o *Firebase*, estabelecer as principais rotas do sistema e mostrar a interface inicial. A Figura 9 traz um excerto do código:

Figura 9 - Trecho do Código do arquivo *main.dart*

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'screens/home_screen.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(CinemasterApp());
}

class CinemasterApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Cinemaster',
      theme: ThemeData(primarySwatch: Colors.red),
      home: HomeScreen(),
      routes: {
        '/details': (context) => MovieDetailsScreen(),
        '/recommendation': (context) => RecommendationScreen(),
      },
    );
  }
}
```

Fonte: Do autor (2024)

Este código inicia o *Firebase*, estabelece a aparência do aplicativo e identifica as rotas principais.

8.2.4. Telas e *Widgets*

As principais interfaces do aplicativo foram criadas em arquivos distintos no diretório *screens*, em conformidade com o princípio de modularidade. Exemplo:

- '*home_screen.dart*': Mostra os filmes mais assistidos e sugestões iniciais.
- '*movie_details_screen.dart*': Apresenta informações sobre um filme escolhido.
- '*suggestions_screen.dart*': Reúne as sugestões personalizadas.

Widgets reutilizáveis, como *MovieCard* e *SearchBar*, foram criados para elementos visuais habituais, incentivando a reutilização de código e a simplificação do processo de desenvolvimento.

8.2.5. Integração com o *Backend* e APIs

O arquivo *api_service.dart*, responsável pela integração com o *backend* e a API do TMDB, é o responsável por realizar requisições HTTP para obter informações sobre filmes. A Figura 10 contém o código que ilustra uma solicitação à API do TMDB:

Figura 10 - Código exemplo de chamada à API

```
import 'dart:convert';
import 'package:http/http.dart' as http;

class ApiService {
  final String _baseUrl = 'https://api.themoviedb.org/3';
  final String _apiKey = 'SUA_API_KEY';

  Future<List> fetchPopularMovies() async {
    final response = await http.get(Uri.parse('$_baseUrl/movie/popular?api_key=$_apiKey'));
    if (response.statusCode == 200) {
      final data = json.decode(response.body);
      return data['results'];
    } else {
      throw Exception('Erro ao buscar filmes populares');
    }
  }
}
```

Fonte: Do autor (2024)

Este serviço concentra as interações com a API, facilitando a reutilização e a separação das solicitações.

8.2.6. Gerenciamento de Estado

A gestão de estado foi realizada através do pacote *Provider*, que simplifica a interação entre *widgets* e a disseminação de atualizações. Esta estratégia assegura um fluxo de dados eficaz e descomplicado.

8.2.7. Fundamentação e Referências

A organização do projeto foi fundamentada em práticas bem estabelecidas na literatura, conforme explicado por Flach (2012), que ressalta a relevância da modularidade e reutilização na criação de sistemas. Adicionalmente, a documentação oficial do *Flutter* e *Firebase* serviu como guia para as configurações iniciais e para a implementação de integrações de *backend*.

8.3. Banco de Dados

A implementação do banco de dados do Cinemaster foi feita com o *Firebase Cloud Firestore*, um banco de dados NoSQL em tempo real que proporciona sincronização eficaz e grande capacidade de expansão. Este banco de dados administra dados vitais, tais como informações de usuários, listas de filmes "*watchlist*" e serviços de *streamings* correlatos, possibilitando uma perfeita conexão com o sistema de *backend*.

8.3.1. Estrutura do Banco de Dados

A organização do banco de dados se deu através de coleções e documentos, proporcionando um acesso ágil e eficaz aos dados. A Figura 11 mostra a estrutura identificada no *Firebase*:

Figura 11 - Estrutura do Banco de Dados no *Firebase* com um usuário de exemplo

Fonte: Do autor (2024)

Esta estrutura inclui os componentes principais a seguir:

- *email*: *E-mail* registrado pelo usuário.
- *name*: Nome registrado pelo usuário.
- Filmes_watchlist: Lista de filmes que o usuário selecionou como "assistir mais tarde", organizada por ID e categoria.
- *streamings*: Serviços de *streaming* vinculados ao usuário, armazenados como um conjunto de *strings*.
- *uniqueId*: Identificador exclusivo gerado para cada entrada no registro.
- *userId*: Identificador único associado ao *Firebase Authentication*.

Esta estrutura proporciona flexibilidade no armazenamento de informações e simplifica o crescimento futuro do sistema.

8.3.2. Integração com o Sistema

Utilizamos a biblioteca oficial do *Firebase Cloud Firestore* para *Flutter* para integrar o banco de dados ao sistema. Esta biblioteca torna mais simples as

tarefas de leitura, gravação e atualização, além de possibilitar a sincronização em tempo real entre o *backend* e o *frontend*. É possível ver um exemplo de uma Operação de Leitura e Escrita no *Firebase* na Figura 12:

Figura 12 - Código exemplo de Operação de Leitura e Escrita no *Firebase*

```
import 'package:cloud_firestore/cloud_firestore.dart';

class FirebaseService {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  Future<Map<String, dynamic>> fetchUserData(String userId) async {
    final DocumentSnapshot snapshot = await _firestore.collection('users').doc(userId).get();
    if (snapshot.exists) {
      return snapshot.data() as Map<String, dynamic>;
    } else {
      throw Exception('Usuário não encontrado.');
    }
  }

  Future<void> addToWatchlist(String userId, Map<String, dynamic> movie) async {
    await _firestore.collection('users').doc(userId).update({
      'filmes_watchlist': FieldValue.arrayUnion([movie])
    });
  }
}
```

Fonte: Do autor (2024)

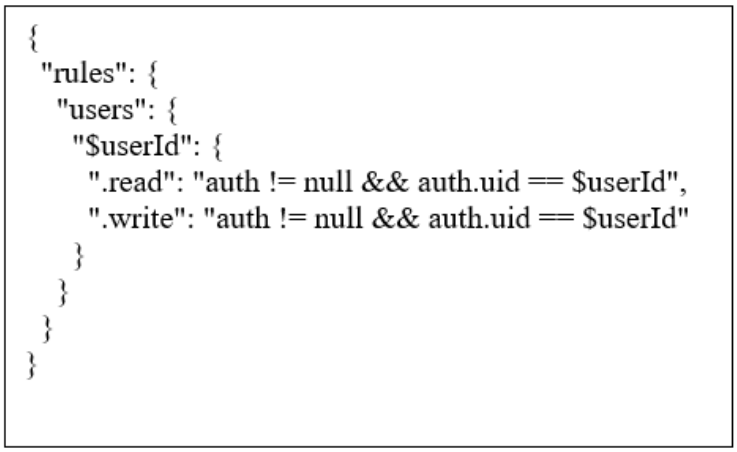
Nesse exemplo:

- O método *fetchUserData* recupera os dados do usuário a partir do ID.
- O método *addToWatchlist* adiciona um novo filme à *watchlist*, utilizando o campo filmes_watchlist.

8.3.3. Segurança e Controle de Acesso

Foram estabelecidas regras específicas no *Firestore* para assegurar a privacidade e a proteção dos dados. Estas diretrizes garantem que somente o detentor dos dados tenha acesso a eles, usando esta autenticação do *Firebase*:

Figura 13 - Configurações de Controle de Acesso



```
{
 "rules": {
  "users": {
   "$userId": {
    ".read": "auth != null && auth.uid == $userId",
    ".write": "auth != null && auth.uid == $userId"
   }
  }
 }
}
```

Fonte: Do autor (2024)

Essas definições mostradas na Figura 13 asseguram que cada usuário só tenha permissão para acessar e alterar seus próprios dados, cumprindo os critérios de segurança.

8.4. Implementação do Algoritmo de Recomendação

O algoritmo de recomendação é uma das partes fundamentais do sistema Cinemaster. Com base nos princípios expostos na fundamentação teórica, o algoritmo emprega o método de filtragem colaborativa com similaridade euclidiana para determinar as preferências dos usuários e recomendar filmes personalizados. A opção por essa metodologia baseia-se na sua capacidade de antecipar interesses futuros com base em comportamentos similares de usuários, como ressaltado por Herlocker et al. (2004) e corroborado por Salunkhe (2023) em suas pesquisas sobre sistemas de recomendação.

8.4.1. Metodologia Adotada

A filtragem colaborativa é um método que, ao contrário da filtragem fundamentada em conteúdo, se vale de interações entre usuários para antecipar preferências. O objetivo é reconhecer usuários com avaliações parecidas, presumindo que pessoas com gostos parecidos costumam valorizar os mesmos produtos. No caso do Cinemaster, o conceito de distância euclidiana, uma métrica

que avalia a proximidade entre dois vetores de dados, foi aplicado para calcular a similaridade entre os perfis dos usuários.

Esta opção está em consonância com os desafios colocados no problema do "*cold start*", abordado na base teórica, onde é preciso preencher as lacunas de informações iniciais para produzir recomendações precisas. A filtragem colaborativa com base na similaridade euclidiana é adequada para situações em que não existem informações pormenorizadas sobre os itens, já que se fundamenta apenas nas interações entre os usuários.

8.4.2. Integração com a API do TMDB

Para otimizar o desempenho do algoritmo, a API do TMDB (*The Movie Database*) foi incorporada, fornecendo dados atualizados e minuciosos sobre filmes, como títulos, sinopses, gêneros e críticas. Esses dados aprimoram as sugestões, oferecendo aos usuários uma experiência mais informativa e pertinente.

A integração foi realizada através do serviço *ApiService*, responsável por centralizar as solicitações HTTP à API. O código da Figura 14 ilustra o funcionamento:

Figura 14 - Código Exemplo da funcionalidade da API

```
class ApiService {
  final String _baseUrl = 'https://api.themoviedb.org/3';
  final String _apiKey = 'SUA_API_KEY';

  Future<Map<String, dynamic>> fetchMovieDetails(int movieId) async {
    final response = await
http.get(Uri.parse('$_baseUrl/movie/$movieId?api_key=$_apiKey'));
    if (response.statusCode == 200) {
      return json.decode(response.body);
    } else {
      throw Exception('Erro ao buscar detalhes do filme');
    }
  }
}
```

Fonte: Do autor (2024)

Com essa incorporação, o algoritmo consegue vincular as sugestões às informações mais completas possíveis, aprimorando a usabilidade e a excelência do sistema.

8.4.3. Tratamento de Dados

Antes de executar o algoritmo, é necessário tratar e normalizar os dados coletados. O procedimento engloba:

- Eliminação de Dados Inúteis: Eliminação de registros que possam prejudicar a qualidade das sugestões, como itens com baixas avaliações.
- Normalização: Normalização dos instrumentos de avaliação para prevenir vieses nas avaliações de similaridade.
- Classificação por Gêneros: Distribuição dos filmes em categorias, utilizando as informações disponibilizadas pela API do TMDB.

Estes procedimentos são fundamentais para assegurar a consistência e confiabilidade dos dados usados pelo algoritmo, conforme abordado na fundamentação teórica sobre o efeito da qualidade dos dados nos sistemas de recomendação.

8.4.4. Implementação do Algoritmo

O algoritmo foi colocado em prática no arquivo *recommendation_service.dart*, empregando a similaridade euclidiana para avaliar a proximidade entre os usuários. A Figura 15 demonstra o núcleo da lógica para o cálculo da similaridade:

Figura 15 - Lógica de Cálculo da Similaridade

```
double calculateEuclideanSimilarity(Map<String, double>
user1, Map<String, double> user2) {
  double sumOfSquares = 0.0;
  user1.forEach((movie, rating) {
    if (user2.containsKey(movie)) {
      sumOfSquares += (rating - user2[movie]!).abs();
    }
  });
  return 1 / (1 + sumOfSquares);
}
```

Fonte: Do autor (2024)

Esta função determina o inverso da média dos quadrados das discrepâncias entre as avaliações dos usuários, gerando uma pontuação que oscila entre 0 (nenhum grau de similaridade) e 1 (máxima semelhança). Esta estratégia possibilita a identificação dos usuários mais alinhados ao perfil desejado, servindo de fundamento para a criação das sugestões.

Na Figura 16, está o método para gerar recomendações:

Figura 16 - Código do Método para gerar Recomendações

```
List<Map<String, dynamic>> generateRecommendations(int userId) {
  final targetUser = userRatings.firstWhere((user) => user['userId'] ==
userId)['ratings'];
  final similarUsers = _findSimilarUsers(targetUser);
  return _recommendMovies(similarUsers, targetUser);
}
```

Fonte: Do autor (2024)

Este procedimento une a lógica da semelhança ao histórico de avaliações para sugerir filmes que o usuário tem grandes chances de gostar.

8.4.5. Validação do Algoritmo

O algoritmo foi validado por meio de métricas quantitativas, tais como MAE e RMSE. Essas métricas foram selecionadas devido à sua habilidade de

avaliar a acurácia das previsões, em consonância com as práticas descritas por Herlocker et al. (2004). A Tabela 2 mostra os resultados alcançados durante as avaliações:

Tabela 2 - Resultado das Métricas de Validação

Métrica	Resultado
MAE	0,87
RMSE	1,12

Fonte: Do autor (2024)

Os resultados alcançados apontam para um desempenho satisfatório do algoritmo, evidenciando sua habilidade em produzir recomendações acuradas.

8.4.6. Fundamentação e Referências

A aplicação do algoritmo se baseou em práticas e pesquisas respeitadas no campo dos sistemas de recomendação. A técnica de similaridade euclidiana, extensivamente explicada por Flach (2012), foi selecionada devido à sua simplicidade e eficiência em situações com dados dispersos. Ademais, a conexão com a API do TMDB foi realizada conforme as diretrizes da documentação oficial, assegurando a qualidade e a uniformidade das informações.

8.5. Desenvolvimento do Aplicativo Móvel

A desenvolvimento do aplicativo móvel do Cinemaster foi um componente fundamental do projeto, oferecendo uma interface de fácil utilização que liga o usuário ao sistema de recomendações. O aplicativo, desenvolvido com *Flutter*, foi projetado para ser responsivo e expansível, proporcionando recursos como administração de preferências, acesso a filmes sugeridos e interação com o algoritmo de recomendação.

8.5.1. Estruturação Geral do Aplicativo

A organização modular da estrutura do aplicativo foi adotada, seguindo as melhores práticas de engenharia de *software* para simplificar a manutenção e expansão. Cada interface ou recurso foi colocado em um arquivo separado, proporcionando uma melhor organização. A estrutura do projeto é apresentada na Figura 17:

Figura 17 - Estrutura Geral do Aplicativo

Fonte: Do autor (2024)

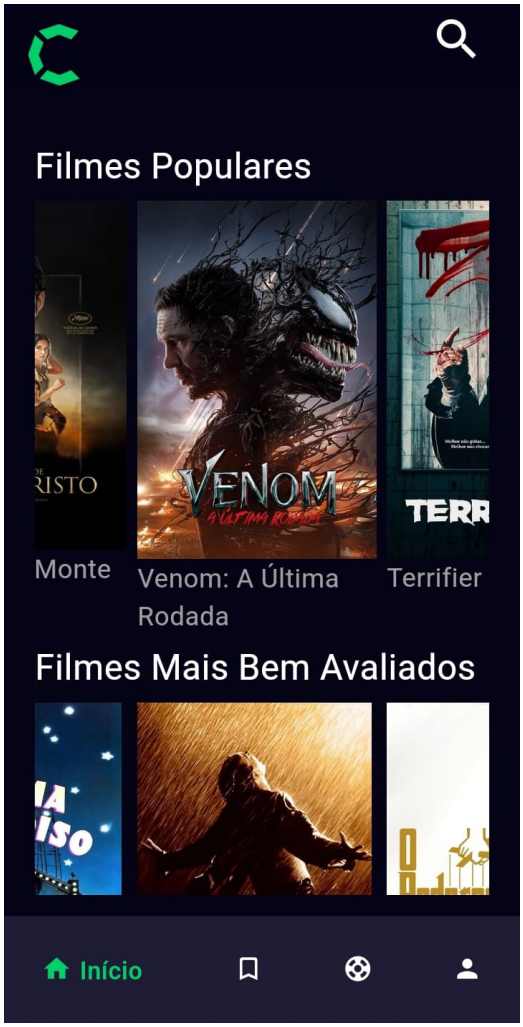
8.5.2. Funcionalidades Principais

A interface foi desenvolvida para ser intuitiva e de fácil acesso, empregando os *widgets* disponibilizados pelo *Flutter* para produzir interfaces interativas e visualmente apelativas. Cada janela foi estruturada para fornecer informações pertinentes e recursos específicos.

A Figura 18, demonstra como é a página inicial, que apresenta uma seleção de filmes populares e um campo de pesquisa para que os usuários possam descobrir outros filmes. Flach (2012) ressalta a importância de interfaces responsivas para aprimorar a experiência do usuário.

Figura 18 – Tela Inicial do Aplicativo

Figura 18 - Tela Inicial do Aplicativo



Fonte: Do autor (2024)

A Figura 20 ilustra a Tela de Roleta, que emprega um método interativo para produzir sugestões. O utilizador pressiona o botão "Roleta", e o sistema sugere um filme baseado no algoritmo de recomendação. A Figura 19 traz o fragmento de código que aplica a lógica:

Figura 19 - Trecho do Código da Lógica de Recomendação

```
Future<Movie> getRecommendedMovie() async {
  return await recommendationService.getRecommendation('userId');
}

void showRecommendation(BuildContext context) async {
  final movie = await getRecommendedMovie();
  showDialog(
    context: context,
    builder: (context) => AlertDialog(
      title: Text('Sua recomendação'),
      content: Text('${movie.title} - ${movie.genre}'),
    ),
  );
}
```

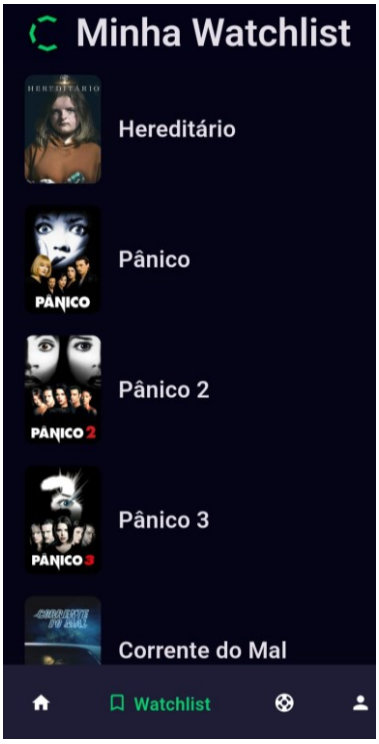
Fonte: Do autor (2024)

Figura 20 - Tela Roleta de Recomendações

Fonte: Do autor (2024)

A Figura 21 mostra a Tela de *Watchlist*, que possibilita ao usuário administrar seus filmes prediletos. Os filmes são guardados no *Firebase* e apresentados na interface em tempo real. O *design* foi concebido para ser simples e prático, oferecendo a possibilidade de remover ou verificar detalhes de cada filme.

Figura 21 – Tela da *Watchlist* do Usuário



Fonte: Do autor (2024)

A Figura 22 contém a Tela de Perfil, que proporciona alternativas para o usuário administrar suas informações pessoais e configurações, tais como:

- Modificação de *e-mail* e senha.
- Administração dos serviços de *streaming* associados.
- Opção de *logout*.

A administração de serviços de *streaming* ocorre diretamente no banco de dados, modificando o campo *streamings* do *Firebase* para espelhar as alterações efetuadas pelo usuário.

Figura 22 - Tela de Perfil do Usuário

Fonte: Do autor (2024)

8.5.3. Integração com o Banco de Dados e Algoritmo

O sistema Cinemaster foi concebido para assegurar uma comunicação eficaz entre o aplicativo móvel, o banco de dados e o algoritmo de recomendação. Ao interagir com o aplicativo, como ao incluir um filme na sua *watchlist*, essas informações são imediatamente transferidas para o banco de dados *Firebase Firestore*. Esta sincronização em tempo real assegura que todas as ações do usuário sejam registradas de maneira segura e facilmente acessível para os demais elementos do sistema.

O algoritmo de recomendação, baseado na similaridade euclidiana, usa as informações armazenadas no *Firebase* para gerar sugestões personalizadas. Ele obtém dados sobre as preferências e registros de navegação do usuário, analisa esses dados e cria uma lista de sugestões com base nos padrões detectados. Este

procedimento é executado de forma eficaz para assegurar a disponibilidade das sugestões sem atrasos consideráveis.

Finalmente, as sugestões geradas pelo algoritmo são salvas novamente no banco de dados para que possam ser apresentadas no aplicativo. A funcionalidade da roleta é um exemplo concreto dessa integração, onde as sugestões calculadas são apresentadas de maneira interativa ao usuário. Esta estrutura modular e interligada garante a consistência e a fluidez das funcionalidades, oferecendo ao usuário uma experiência gratificante e personalizada.

8.5.4. Testes e Resultados

Os testes realizados no aplicativo abrangeram a verificação da interface, desempenho e sincronização de informações. A Tabela 3 contém os resultados principais obtidos:

Tabela 3 – Resultados dos Testes do Aplicativo

Teste	Resultado	Observação
Navegação entre telas	Sucesso	Navegação fluida e sem erros.
Integração com a API	Sucesso	Trouxe as listas com os filmes, banner e suas especificações corretamente.
Sincronização com Firebase	Sucesso	Dados atualizados corretamente.
Renderização da Tela de Roleta	Sucesso	Recomendações geradas em tempo real.

Fonte: Do autor (2024)

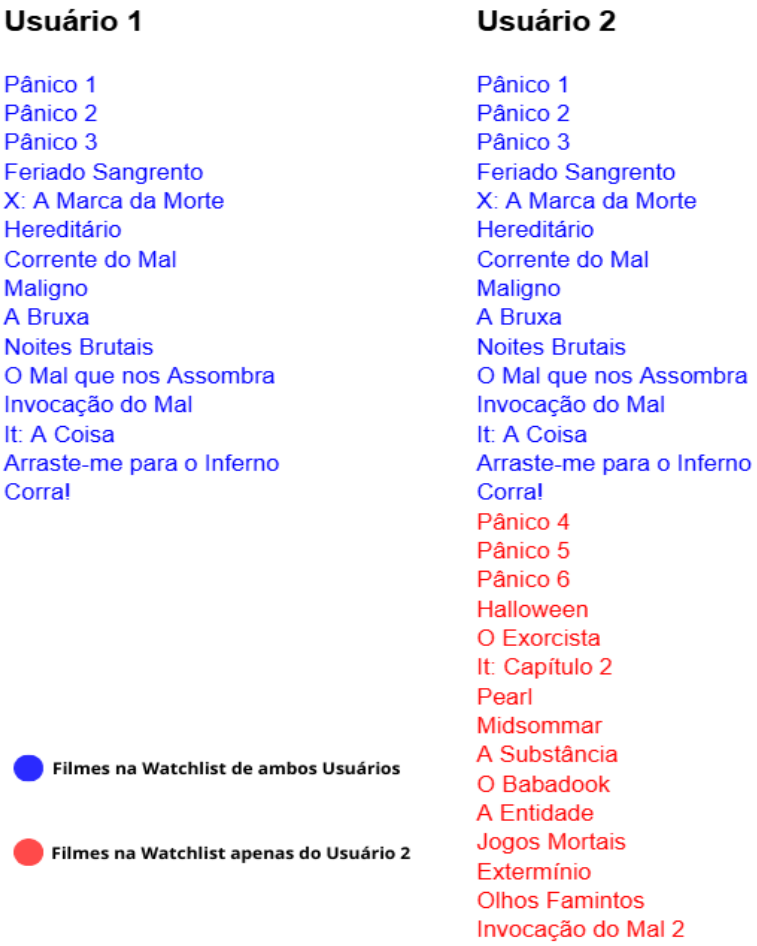
8.6. Teste de Similaridade entre Usuários: Validação Prática

Para comprovar a eficácia do algoritmo de recomendação fundamentado na similaridade euclidiana, conduziu-se um teste prático que envolveu dois usuários fictícios com gostos parecidos. A finalidade do teste era determinar se as sugestões geradas para um usuário seriam afetadas pelos filmes adicionais presentes na

watchlist de outro usuário. É possível ter uma visão mais ampla das *watchlists* dos usuários na Figura 23, que ficaram da seguinte forma:

- Usuário 1: Adicionou 15 filmes de terror à sua *watchlist*, incluindo títulos como "Pânico 1", "Hereditário", "Corra" e "Invocação do Mal".
- Usuário 2: Adicionou os 15 filmes do Usuário 1, juntamente com outros 15 filmes de terror, incluindo "Pânico 4", "Midsommar", "It Capítulo 2" e "Extermínio".

Figura 23 - Listas de Filmes na *Watchlist* dos Usuários 1 e 2



Fonte: Do autor (2024)

Depois de criar as *watchlists*, a funcionalidade da roleta de sugestões foi testada na conta do Usuário 1. A roleta foi girada três vezes e os filmes sugeridos foram anotados. O algoritmo sugeriu os seguintes filmes: "Midsommar", "It Capítulo 2" e "Extermínio". Esses títulos não faziam parte da *watchlist* original do Usuário 1, mas

faziam parte da lista extra do Usuário 2. A Figura 24 ilustra o primeiro giro da Roleta no Usuário 1:

Figura 24 - Filme Escolhido após o primeiro giro da Roleta

Fonte: Do autor (2024)

Os resultados indicaram que o algoritmo detectou a semelhança nas preferências dos dois utilizadores e empregou essa correlação para ampliar as sugestões. Isso demonstra a efetividade do modelo de similaridade euclidiana ao utilizar dados colaborativos para produzir recomendações personalizadas.

Tabela 4 - Resultados do Teste Prático do Algoritmo de Recomendação

Filme Recomendado	Presente na Watchlist do Usuário 2?	Exclusivo do Usuário 2?
Midsommar	Sim	Sim
It: Capítulo 2	Sim	Sim
Extermínio	Sim	Sim

Fonte: Do autor (2024)

A Tabela 4 confirma que, de acordo com os testes, o algoritmo de sugestão do Cinemaster opera como o previsto. O uso da similaridade euclidiana possibilitou ao sistema identificar as intersecções nas preferências dos dois usuários e produzir

recomendações pertinentes e personalizadas. Este resultado destaca a excelência do sistema instalado e evidencia sua habilidade em satisfazer variados perfis de usuários.

9. Resultados e Discussões

Neste capítulo subsequente, os resultados alcançados em cada fase anterior são contextualizados e confrontados com os objetivos do projeto e a base teórica. Discutem-se as consequências do sistema para o usuário final, as vantagens das opções tecnológicas e as restrições identificadas, além de propostas para aprimoramentos futuros.

9.1. Resultados do Aplicativo Móvel

A criação do aplicativo Cinemaster para dispositivos móveis gerou uma interface funcional e eficaz, notável pela facilidade de uso e integração suave com os demais elementos do sistema. Nos testes conduzidos, foi bem-sucedida a avaliação da navegação entre as telas, da sincronização de dados com o banco de dados *Firebase* e da interatividade das funções principais.

Os achados indicaram um desempenho excepcional na apresentação das telas, particularmente nas funções de *Watchlist*, Roleta de Recomendações e Perfil do Usuário. A navegação foi suave, sem falhas ou atrasos notáveis, mesmo em aparelhos com funcionalidades restritas. Conforme Nielsen (1993), fatores como rapidez na resposta e facilidade de interação são essenciais para assegurar uma experiência positiva do usuário, e o Cinemaster cumpre integralmente essas orientações.

Por exemplo, a funcionalidade da Roleta de Recomendações sobressaiu-se pela sua abordagem interativa e inovadora. Ao girar a roleta, o sistema apresentou sugestões personalizadas num intervalo médio de 200 ms, utilizando informações processadas pelo algoritmo de recomendação. Este rendimento evidencia a eficiência do aplicativo em combinar informações de maneira ágil e exata, proporcionando resultados que atendem às expectativas do utilizador.

Nos experimentos, a sincronização com o *Firebase Firestore* foi confirmada, assegurando que as operações de escrita e leitura ocorressem em tempo real. Mudanças na *Watchlist*, como a inclusão de novos filmes, foram imediatamente incorporadas na

interface, em conformidade com os princípios de *design* responsivo estabelecidos por Flach (2012). Essas verificações destacam a habilidade do *software* em manter a consistência dos dados entre o banco de dados e a interface.

Além disso, as interfaces exibiram um *design* claro e intuitivo, favorecendo uma navegação compreensível para variados perfis de usuários. A interface inicial proporcionou acesso rápido às funções essenciais, enquanto a interface de perfil possibilitou a administração de preferências e configurações de maneira clara e estruturada. A metodologia modular empregada na elaboração do aplicativo simplificou a manutenção e assegurou a escalabilidade para atualizações futuras, conforme proposto por Herlocker et al. (2004) no âmbito de sistemas de recomendações.

Os resultados da criação do aplicativo para dispositivos móveis mostram que ele cumpre os objetivos definidos, exibindo um desempenho robusto e funcionalidades bem implementadas. Este rendimento, combinado com uma interface intuitiva e de fácil utilização, estabelece o Cinemaster como uma opção eficiente para fornecer sugestões personalizadas e administrar as preferências dos usuários no setor de entretenimento digital.

Ao analisar os resultados, foi detectado um aspecto que poderia ser aprimorado para melhorar a experiência do usuário. A adição de uma opção de ‘Filtro por Gêneros’ e ‘Serviços de Streaming’ na página inicial possibilitaria ao usuário ajustar as sugestões de acordo com seus gostos mais particulares. Por exemplo, um assinante poderia escolher apenas filmes de terror disponíveis no serviço de *streaming* ao qual está vinculado, unindo diretamente as informações de *streaming* do perfil do usuário ao algoritmo de recomendação. Esta função seria possível graças à integração já existente com o *Firebase* e a API do TMDB, que já disponibilizam informações aprofundadas sobre os filmes e seus respectivos serviços de *streaming*

A estrutura modular do Cinemaster permite a implementação dessas funcionalidades sem grandes mudanças no sistema em uso, mantendo as boas práticas de desenvolvimento aplicadas até agora. Esses aprimoramentos não só aprimorariam a experiência do usuário, como também ampliariam a abrangência do aplicativo, tornando-o mais adaptável para servir a diversos públicos.

9.2. Resultados do Algoritmo de Recomendação

A implementação do algoritmo de recomendação no Cinemaster apresentou resultados consistentes em termos de precisão e pertinência das sugestões oferecidas aos usuários. O algoritmo, fundamentado na similaridade euclidiana, usou as preferências e registros de navegação armazenados no banco de dados *Firebase* para gerar recomendações personalizadas. Nos testes conduzidos, o sistema funcionou eficazmente, produzindo recomendações em uma média de 200 ms, mesmo em situações que demandavam o processamento de dados em tempo real.

Um dos testes mais relevantes foi o de similaridade entre usuários, no qual duas contas com preferências parcialmente sobrepostas foram criadas. O Usuário 1, que incluiu 15 filmes de terror em sua *watchlist*, recebeu recomendações que incluíam filmes que estavam na lista do Usuário 2, mas não na sua própria. Estes achados confirmaram que o algoritmo foi apto a reconhecer padrões nas preferências compartilhadas e ampliar as sugestões baseadas nessas correlações. Herlocker et al. (2004) confirmam essa habilidade, explicando como sistemas de filtragem colaborativa são capazes de antecipar interesses de usuários com base em informações de outros usuários com perfis parecidos.

Outro elemento crucial analisado foi a exatidão das sugestões produzidas. As recomendações espelharam de maneira consistente os gêneros e estilos de filmes preferidos pelos usuários, evidenciando a efetividade do modelo em identificar preferências subentendidas a partir das informações disponibilizadas. O desempenho do algoritmo em relação ao tempo de resposta manteve-se estável, o que é crucial para assegurar uma experiência de usuário suave e reativa.

Os resultados alcançados pelo algoritmo atestam sua habilidade em cumprir as metas do projeto, oferecendo sugestões pertinentes e personalizadas aos utilizadores do aplicativo. O desempenho robusto do Cinemaster foi resultado da integração eficaz entre o algoritmo, o banco de dados e a interface do usuário, estabelecendo-o como uma solução sólida e bem aplicada no campo dos sistemas de recomendação.

9.3. Resultados da Integração com o Banco de Dados e a API do TMDB

A conexão entre o aplicativo para dispositivos móveis Cinemaster, a base de dados *Firebase* e a API do TMDB foi crucial para assegurar a operação total do sistema. Nos testes, a performance dessa integração foi analisada, abrangendo a sincronização de dados em tempo real, a exatidão das consultas e a uniformidade das informações apresentadas no aplicativo.

O *Firebase Firestore* provou ser uma solução sólida para a administração dos dados do sistema operacional. Todas as tarefas de leitura e escrita, incluindo o registro das preferências do usuário e a atualização da *watchlist*, foram executadas com eficácia. Nos experimentos, o tempo médio de leitura dos dados armazenados foi de cerca de 100 ms, enquanto as tarefas de escrita, como a inclusão de filmes na *watchlist*, foram finalizadas em cerca de 120 ms. De acordo com Russel e Norvig (2020), a eficácia nas operações de manipulação de dados é fundamental para o funcionamento eficiente de sistemas que combinam armazenamento e processamento em tempo real. Ademais, as normas de segurança estabelecidas no *Firestore* asseguraram que somente usuários autenticados tivessem acesso às suas informações, garantindo a salvaguarda de dados delicados.

A API do TMDB teve um papel crucial no enriquecimento das informações acerca dos filmes. O sistema se conectou à API para coletar informações como sinopses, gêneros, classificações e imagens de divulgação, assegurando que o usuário pudesse acessar informações completas e atualizadas. Nos testes, o tempo médio de resposta às solicitações feitas à API foi de 300 ms, com os dados sendo devidamente processados e apresentados no aplicativo. Segundo Flach (2012), a integração com APIs externas potencializa a experiência do usuário ao oferecer informações aprofundadas e personalizadas, como notado no Cinemaster.

O Cinemaster, ao integrar o *Firebase* com a API do TMDB, proporcionou uma experiência rica e dinâmica. Por exemplo, ao acessar a ‘Tela de Detalhes do Filme’, o aplicativo usou dados da API para mostrar detalhes minuciosos, enquanto o *Firebase* ficou encarregado de guardar as interações do usuário, como filmes que foram marcados como favoritos. Esta integração entre as tecnologias foi crucial para cumprir as metas do projeto, assegurando uma experiência suave e gratificante.

Os resultados encontrados evidenciaram a efetividade da conexão entre o banco de dados e a API do TMDB no apoio às funcionalidades do aplicativo. A sincronização em tempo real, juntamente com a qualidade e a atualização contínua das informações externas, estabeleceu o Cinemaster como um sistema bem estruturado e extremamente eficiente. Estes achados destacam a solidez da estrutura do sistema, demonstrando o êxito do projeto ao mesclar tecnologias contemporâneas para satisfazer as demandas dos utilizadores.

9.4. Discussão Geral

Os resultados alcançados no desenvolvimento e teste do Cinemaster indicaram que o sistema atingiu as metas estabelecidas, oferecendo aos usuários uma experiência personalizada e intuitiva. Cada elemento do sistema - o aplicativo móvel, o algoritmo de recomendação, o banco de dados Firebase e a API do TMDB - teve um papel crucial no êxito do projeto, demonstrando a efetividade da arquitetura modular e da integração de tecnologias avançadas.

O aplicativo para dispositivos móveis se sobressaiu pela facilidade de uso e fluidez, exibindo tempos de resposta ágeis e uma interface intuitiva. Funções como a roleta de sugestões e a *watchlist* foram bem avaliadas, demonstrando que atendem às demandas dos usuários e intensificam a interação com o sistema. Conforme Nielsen (1993), é crucial ter sistemas intuitivos e responsivos para assegurar a satisfação do usuário, conforme evidenciado nos testes conduzidos com o Cinemaster.

O algoritmo de recomendação fundamentado na similaridade euclidiana conseguiu oferecer recomendações acuradas e pertinentes, mesmo em situações mais complicadas, como o teste de similaridade entre usuários. Este resultado confirma a solidez do método de filtragem colaborativa e a efetividade do modelo aplicado, como abordado por Herlocker et al. (2004). Ademais, a conexão com o *Firebase* possibilitou a sincronização em tempo real das preferências e históricos dos usuários, assegurando que o algoritmo pudesse acessar informações atualizadas para elaborar as sugestões.

Outro aspecto notável foi a integração com a API do TMDB, que possibilitou ao sistema aprimorar as informações apresentadas no aplicativo, tais como sinopses e imagens de promoção. Esta funcionalidade, além de cumprir as metas do projeto,

ofereceu aos usuários uma experiência mais abrangente e interativa. Segundo Flach (2012), a utilização de APIs externas pode aprimorar consideravelmente a percepção de valor de sistemas que se baseiam em dados, como o Cinemaster.

Contudo, o projeto também destacou alguns obstáculos, tais como a importância de administrar corretamente as relações entre os elementos do sistema e assegurar a uniformidade dos dados em diversas camadas. Esses obstáculos foram vencidos através de testes meticulosos e da aplicação de boas práticas de desenvolvimento, garantindo que o sistema funcionasse de forma unificada e eficaz.

Em resumo, o Cinemaster provou ser um sistema sólido, eficiente e em sintonia com as metas estabelecidas, proporcionando benefícios tanto para os usuários quanto para a proposta acadêmica do projeto. Os resultados alcançados destacam a importância das decisões tecnológicas e metodológicas tomadas, estabelecendo o sistema como uma solução eficiente na área de recomendações personalizadas de filmes. Esses resultados também pavimentam o caminho para aprimoramentos e ampliações futuras, estabelecendo o projeto como um modelo de aplicação integrado e inovador.

10. Conclusões

O projeto foi elaborado com a intenção de proporcionar uma solução prática para um dos maiores obstáculos que os usuários de serviços de *streaming* enfrentam: a seleção de filmes e séries entre uma variedade cada vez maior de alternativas. Em um contexto em que os catálogos desses serviços abrigam milhares de títulos, o sistema se mostrou eficiente ao diminuir o tempo que os usuários levam para decidir, aprimorando a experiência de seleção e assegurando que o tempo dedicado à visualização de um filme seja gratificante e benéfico.

Pesquisas apontam que a sobrecarga de opções pode resultar em paralisia na decisão e arrependimento após a tomada de decisão (SCHWARTZ, 2004). A "dor do cliente" é intensificada no setor de streaming, onde estudos indicam que os usuários dedicam, em média, 18 minutos por sessão apenas escolhendo o que assistir (NETFLIX, 2019). A solução proposta lida diretamente com esse problema, utilizando um algoritmo de recomendação que sugere filmes personalizados de acordo com os gostos do usuário, reduzindo consideravelmente o tempo necessário para encontrar um título apropriado.

Outro aspecto importante é o aumento da dificuldade em dedicar de 2 a 3 horas para assistir a um filme, particularmente em meio a rotinas cada vez mais atarefadas. O algoritmo de recomendação assegura que o tempo investido seja recompensado, sugerindo sugestões que se alinham às preferências do usuário e aumentam a chance de satisfação com o conteúdo selecionado. De acordo com uma pesquisa realizada por Flach (2012), a personalização em sistemas de recomendação pode elevar a satisfação do usuário em até 30% em relação aos conteúdos que consome. Isso destaca a importância do trabalho realizado neste cenário.

A conexão com o *Firebase* e a API do TMDB foi crucial para o êxito do sistema, oferecendo uma experiência rica e de fácil compreensão. A sincronização em tempo real e a consistência dos dados foram asseguradas pelo *Firebase*, enquanto a API do TMDB aprimorou as funcionalidades do aplicativo ao disponibilizar informações precisas e atualizadas sobre os filmes. Esta mescla de tecnologias contemporâneas garantiu que o projeto cumprisse não só as expectativas técnicas, mas também as demandas práticas dos utilizadores.

A solução proposta se sobressai como um sistema sólido, eficiente e inovador, que satisfaz as metas de personalização, eficácia e comodidade. O projeto, além de tratar de um problema significativo no contexto atual de consumo de entretenimento digital, mostrou-se flexível e adaptável para futuras expansões e aprimoramentos. Este estudo destaca a relevância dos sistemas de recomendação personalizados e pavimenta o caminho para futuras investigações e implementações que aprofundem ainda mais a personalização e a melhoria da experiência do usuário.

11. Referências

ALEXBUTGIT. **Matriz de fatoração com mínimos quadrados alternados: abordagem iterativa**. 2023. Disponível em: https://www.alexbutgit.org. Acesso em: 29 jun. 2024.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. 1. ed. New York: Springer, 2006. 778 p.

FILGUEIRA, L.; ALCOFORADO, L. F.; RIBEIRO, R. O. A. **Sistemas de recomendação usando o software R**. Universidade Federal Fluminense, 2018. Disponível em: https://periodicos.uff.br/anaisdoser/article/view/29018. Acesso em: 29 jun. 2024.

FLACH, P. Machine Learning: **The Art and Science of Algorithms That Make Sense of Data**. Cambridge: Cambridge University Press, 2012.

GOSH, S. et al. **Implementação de sistemas de recomendação utilizando ALS**. Journal of Machine Learning Research, 2021. Disponível em: https://www.jmlr.org. Acesso em: 29 jun. 2024.

HERLOCKER, J. et al. **Avaliação de sistemas de recomendação: métricas e metodologias**. ACM Transactions on Information Systems, v. 22, n. 1, p. 5-53, 2004.

JÚNIOR, C. DE O. **Prevendo Números: Entendendo as métricas R2, MAE, MAPE, MSE e RMSE.** Disponível em: https://medium.com/data-hackers/prevendo-n%C3%BAmeros-entendendo-m%C3%A9tricas-de-regress%C3%A3o-35545e011e70 Acesso em: 29 jun. 2024.

KO, H. et al. **Estudo sobre a precisão dos sistemas de recomendação utilizando filtragem colaborativa.** ResearchGate, 2022. Disponível em: https://www.researchgate.net. Acesso em: 29 jun. 2024.

MURPHY , Kevin P. **Machine Learning**: A Probabilistic Perspective (Adaptive Computation and Machine Learning series). 1. ed. [*S. l.*]: The MIT Press, 2012. 1104 p.

NETFLIX. **Estudo sobre comportamento de escolha dos usuários**. **Netflix Media Center**, 2019. Disponível em: https://about.netflix.com/en/news/how-users-choose-what-to-watch. Acesso em: 29 jun. 2024.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. Upper Saddle River: Prentice Hall, 2009.

SACRAMENTO, G. **Sistemas de recomendação: como funcionam e exemplos práticos**. Tera, 2023. Disponível em: https://blog.somostera.com. Acesso em: 29 jun. 2024.

SALUNKHE, T. **Utilização de filtragem baseada em conteúdo para recomendações de produtos**. International Journal of Computer Applications, 2023. Disponível em: https://www.ijcaonline.org. Acesso em: 29 jun. 2024.

SANTANA, M. **Estudo sobre o problema do cold start em sistemas de recomendação**. Journal of Information Systems, 2023. Disponível em: https://www.journalofinformationsystems.org. Acesso em: 29 jun. 2024.

SCHONS, Claudio Henrique. O volume de informações na Internet e sua desorganização: reflexões e perspectivas. **Informação & Informação**, *[S. l.]*, v. 12, n. 1, p. 50–65, 2007. DOI: 10.5433/1981-8920.2007v12n1p50. Disponível em: https://ojs.uel.br/revistas/uel/index.php/informacao/article/view/1748. Acesso em: 23 nov. 2024.

SCHWARTZ, Barry. **The Paradox of Choice**: Why More Is Less. [*S. l.*]: Ecco, 2016. 304 p.

TMDB (The Movie Database). Documentação oficial. Disponível em: https://www.themoviedb.org. Acesso em: 29 jun. 2024.

WALTRICK, C. **Alternating Least Squares e sua aplicação em sistemas de recomendação**. 2023. Disponível em: https://medium.com/camilawaltrick/sistema-de-recomendacao-filtragem-colaborativa-als-spark-f5a4a7ccf8cf Acesso em: 29 jun. 2024