

# **Sistema de recomendação de filmes**

para  
SISREC

Preparado por:  
Tiago Nora (1201050)  
João Figueiredo (1230194)  
João Gaspar (1200884)

*Instituto Superior de Engenharia do Porto, Porto  
Sistemas de Informação e Conhecimento  
Sistemas de Recomendação  
António Constantino Lopes Martins (ACM)  
Catarina de Almeida Figueiredo (AZC)  
Joaquim Filipe Peixoto dos Santos (JPE)  
Maria Dulce Fernandes Mota (MDM)*

Porto, 9 de junho de 2024

# Conteúdo

---

<b>Lista de Figuras</b>	<b>v</b>
<b>Lista de Tabelas</b>	<b>vii</b>
<b>Listagens</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Contextualização . . . . .	1
1.2 Objetivos . . . . .	1
1.3 Divisão do documento . . . . .	2
<b>2 Introdução do dataset</b>	<b>3</b>
2.1 Apresentação do dataset . . . . .	3
2.1.1 Apresentação das tabelas . . . . .	3
2.2 Dados estatísticos . . . . .	5
2.3 Preparação do dataset . . . . .	16
2.3.1 Agregação de duas tabelas . . . . .	16
2.3.2 Adição do endereço da imagem do filme . . . . .	17
2.3.3 Separação do título e do ano do filme . . . . .	18
2.3.4 Criação de uma coluna adicional . . . . .	19
<b>3 Apresentação das tecnologias</b>	<b>20</b>
3.1 Tecnologias, plataformas e ferramentas utilizadas . . . . .	20

3.1.1	Angular . . . . .	20
3.1.2	FastAPI . . . . .	21
3.1.3	TypeScript . . . . .	21
3.1.4	Python . . . . .	21
3.1.5	TablePlus . . . . .	21
3.1.6	AWS . . . . .	22
3.1.7	PostegreSQL . . . . .	22
<b>4</b>	<b>Técnicas usadas na recomendação</b>	<b>23</b>
4.1	Processamento de Dados . . . . .	23
4.2	Recomendação não personalizada . . . . .	23
4.2.1	Melhores filmes overall . . . . .	24
4.2.2	Melhores filmes de um determinado género . . . . .	24
4.2.3	Melhores filmes de um ano . . . . .	25
4.2.4	Melhores filmes de uma década . . . . .	25
4.3	Recomendação personalizada . . . . .	26
4.3.1	Filtragem Colaborativa . . . . .	26
4.3.2	Filtragem Baseada em Conteúdo . . . . .	29
4.3.3	Filtragem Baseada em Conhecimento . . . . .	30
4.3.4	Abordagem Hibrida . . . . .	33
4.4	Alternativas para se realizar a recomendação - Trabalho futuro . . . . .	34
<b>5</b>	<b>Solução pensada</b>	<b>36</b>
5.1	Diagrama de componentes . . . . .	36
5.2	Mockup . . . . .	36
5.3	Modelo de domínio . . . . .	49
5.4	Diagrama de casos de uso . . . . .	51
5.5	Solução para os problemas dos sistemas de recomendação . . . . .	53
5.5.1	Cold Start . . . . .	53
5.5.2	Mecanismo de autenticação . . . . .	54

---

<b>6 Frontend</b>	<b>56</b>
6.1 Páginas criadas . . . . .	56
6.1.1 Página inicial . . . . .	56
6.1.2 Autenticação . . . . .	58
6.1.3 Registo . . . . .	58
6.1.4 Últimas Avaliações . . . . .	59
6.1.5 Filmes melhor avaliados . . . . .	60
6.1.6 Avaliação de filmes . . . . .	61
6.1.7 Recomendações personalizadas . . . . .	61
6.1.8 Recomendações de filmes parecidos ao selecionado . . . . .	62
6.1.9 Recomendações de filmes parecidos ao perfil do utilizador . . . . .	63
<b>7 Backend</b>	<b>64</b>
7.1 Endpoints definidos . . . . .	64
7.2 Entidades Criadas . . . . .	83
7.2.1 Movie . . . . .	84
7.2.2 Rating . . . . .	85
7.2.3 User . . . . .	85
7.3 Outras informações . . . . .	86
7.3.1 Inicialização da aplicação . . . . .	86
7.3.2 Operações de segundo plano . . . . .	87
7.3.3 CORS . . . . .	87
7.4 Bibliotecas usadas . . . . .	87
7.4.1 Pandas . . . . .	88
7.4.2 FastAPI . . . . .	89
7.4.3 SQLAlchemy . . . . .	90
7.4.4 Typing . . . . .	90
7.4.5 SQLModel . . . . .	91
7.4.6 Time . . . . .	91

7.4.7	Uvicorn . . . . .	92
7.4.8	Contextlib . . . . .	92
7.4.9	Apscheduler.schedulers.background . . . . .	92
7.4.10	Asyncpg . . . . .	92
7.4.11	Requests . . . . .	93
7.4.12	OS . . . . .	93
7.4.13	Dotenv . . . . .	94
7.4.14	Motivação para Escolha das Bibliotecas . . . . .	94
<b>8</b>	<b>Avaliação do Sistema de Recomendação</b>	<b>95</b>
8.1	Avaliação de sistemas de recomendação - Introdução . . . . .	95
8.2	Formas de avaliação do sistema de recomendação . . . . .	96
<b>9</b>	<b>Conclusão</b>	<b>98</b>
9.1	Resultados obtidos . . . . .	98
9.2	Trabalho futuro . . . . .	99
		<b>100</b>
	<b>Bibliografia</b>	<b>100</b>

---

# Listas de Figuras

2.1	Distribuição dos filmes por género . . . . .	5
2.2	Distribuição das avaliações dos filmes . . . . .	6
2.3	Distribuição das avaliações médias por género . . . . .	7
2.4	Distribuição dos números de filmes por cada ano (2019-1953) . . . . .	8
2.5	Distribuição dos números de filmes por cada ano (1952-1874) . . . . .	9
2.6	Top 20 de filmes — Classificação ponderada Bayesiana . . . . .	10
2.7	Distribuição do número de avaliações realizadas ao longo dos anos . . . . .	10
2.8	Distribuição das avaliações médias realizadas ao longo dos anos . . . . .	11
2.9	Frequência de ocorrência dos géneros . . . . .	11
2.10	Distribuição do número de avaliações realizadas pelos utilizadores . . . . .	12
2.11	Distribuição do número de etiquetas pelos filmes . . . . .	13
2.12	Top 10 das etiquetas mais comuns . . . . .	14
2.13	Top 20 das etiquetas mais comuns . . . . .	15
2.14	Top 20 das etiquetas mais comuns - TF-IDF . . . . .	16
4.1	Não personalizado sem critérios . . . . .	24
4.2	Não personalizado sem critérios . . . . .	25
4.3	Não personalizado conforme o ano . . . . .	25
4.4	Não personalizado conforme a década . . . . .	26
5.1	Diagrama de componentes . . . . .	36
5.2	Realizar o registo do utilizador . . . . .	37

---

5.3	Escolha do géneros . . . . .	38
5.4	Realizar a autenticação do utilizador . . . . .	39
5.5	Recomendação não personalizada . . . . .	40
5.6	Recomendação personalizada . . . . .	41
5.7	Página do filme . . . . .	42
5.8	Página de testes das técnicas . . . . .	43
5.9	Página de testes “Content Based” . . . . .	44
5.10	Página de testes “Collaborative” . . . . .	45
5.11	Página de testes “Knowledge” . . . . .	46
5.12	Escolha das técnicas que se pretende utilizar . . . . .	47
5.13	Página de testes “Hybrid” . . . . .	48
5.14	Avaliação do sistema de recomendação . . . . .	49
5.15	Modelo de domínio . . . . .	50
5.16	Diagrama de casos de uso 1 . . . . .	52
5.17	Diagrama de casos de uso 2 . . . . .	53
5.18	Cold Start - 1 . . . . .	54
5.19	Cold Start - 2 . . . . .	54
5.20	Autenticação Falhada . . . . .	55
6.1	Página Inicial . . . . .	57
6.2	Recomendações não personaliadas de filmes de 2010 . . . . .	57
6.3	Autenticação . . . . .	58
6.4	Registo no sistema . . . . .	59
6.5	Últimas avaliações . . . . .	60
6.6	Filmes Preferidos . . . . .	60
6.7	Avaliação de filmes . . . . .	61
6.8	Recomendações personalizadas . . . . .	62
6.9	Recomendações Content-Based . . . . .	63
6.10	Abordagem Colaborativa . . . . .	63

# Listas de Tabelas

7.1	<i>Endpoints</i> do router MovieRouter . . . . .	64
7.2	<i>Endpoints</i> do router RatingRouter . . . . .	66
7.3	<i>Endpoints</i> do router RecommendationRouter . . . . .	71
7.4	<i>Endpoints</i> do router UserRouter . . . . .	80

# Listagens

2.1	Agregação das tabelas. . . . .	17
2.2	Adição do endereço da imagem do filme. . . . .	18
2.3	Separação do título e do ano do filme. . . . .	18
2.4	Criação da tabela “titleLower”. . . . .	19
4.1	Filtragem baseada em colaborativa. . . . .	28
4.2	Filtragem baseada em conteúdo. . . . .	30
4.3	Filtragem baseada em conteúdo. . . . .	32
7.1	Resposta exemplo do endpoint GET /movies/search. . . . .	65
7.2	Resposta exemplo do endpoint GET /movies/randomMovie. . . . .	66
7.3	Resposta exemplo do endpoint GET /movies/similarMovies. . . . .	66
7.4	Payload exemplo do endpoint POST /ratings/. . . . .	67
7.5	Resposta exemplo do endpoint POST /ratings/. . . . .	67
7.6	Resposta exemplo do endpoint GET /ratings/. . . . .	68
7.7	Resposta exemplo do endpoint GET /ratings/history. . . . .	69
7.8	Resposta exemplo do endpoint GET /ratings/favoriteMovies. . . . .	70
7.9	Resposta exemplo do endpoint GET /recommendation/nonpersonalized. .	72
7.10	Resposta exemplo do endpoint GET /recommendation/nonpersonalized-Genre. . . . .	73
7.11	Resposta exemplo do endpoint GET /recommendation/nonpersonalizedYear. .	74
7.12	Resposta exemplo do endpoint GET /recommendation/nonpersonalized-Decade. . . . .	75
7.13	Resposta exemplo do endpoint GET /recommendation/nonpersonalizedOverall. . . . .	76
7.14	Resposta exemplo do endpoint GET /recommendation/personalizedCollaborative. . . . .	77
7.15	Resposta exemplo do endpoint GET /recommendation/personalizedContent. . . . .	78
7.16	Resposta exemplo do endpoint GET /recommendation/personalizedKnowledge. . . . .	79
7.17	Resposta exemplo do endpoint GET /recommendation/personalizedHybrid. .	80
7.18	Payload exemplo do endpoint POST /users. . . . .	81
7.19	Resposta exemplo do endpoint POST /users. . . . .	81
7.20	Payload exemplo do endpoint POST /users/genres. . . . .	81

7.21 Resposta exemplo do endpoint POST /users/genres. . . . .	82
7.22 Payload exemplo do endpoint POST /users/login. . . . .	82
7.23 Resposta exemplo do endpoint POST /users/login. . . . .	82
7.24 Resposta exemplo do endpoint GET /users. . . . .	83
7.25 Payload exemplo do endpoint PATCH /users. . . . .	83
7.26 Resposta exemplo do endpoint PATCH /users. . . . .	83
7.27 Entidade Movie. . . . .	84
7.28 Entidade Rating. . . . .	85
7.29 Entidade User. . . . .	85
7.30 Inicialização da aplicação. . . . .	86
7.31 Operação de segundo plano. . . . .	87
7.32 CORS. . . . .	87
7.33 Utilização do pandas. . . . .	89
7.34 Utilização do FastAPI. . . . .	90
7.35 Utilização do SQLAlchemy. . . . .	90
7.36 Utilização do Typing e SQLModel. . . . .	91
7.37 Utilização do Time. . . . .	91
7.38 Utilização do Asyncpg. . . . .	92
7.39 Utilização do Requests. . . . .	93

# Capítulo 1

## Introdução

Neste capítulo é feita uma contextualização do trabalho desenvolvido onde são apresentados os objetivos definidos pelo grupo e por último é apresentado a divisão do documento.

### 1.1 Contextualização

Nos últimos anos, a indústria cinematográfica testemunha uma explosão de conteúdo, com uma quantidade crescente de filmes sendo produzidos e disponibilizados em plataformas de streaming e cinemas. Enquanto isso, os espetadores enfrentam o desafio de navegar por essa vasta gama de opções para encontrar os filmes que melhor se adequam aos seus gostos e interesses individuais.

Para resolver esse dilema e melhorar a experiência do utilizador, os sistemas de recomendação de filmes têm se tornado uma ferramenta essencial. Esses sistemas utilizam algoritmos sofisticados que analisam o comportamento do utilizador, preferências passadas e características dos filmes para sugerir conteúdos relevantes e personalizados.

Neste trabalho é explorado a implementação de um sistema de recomendação de filmes, visando proporcionar aos utilizadores uma experiência mais satisfatória na procura de filmes para este ver.

### 1.2 Objetivos

No que toca a objetivos desta parte do trabalho foram identificados os seguintes:

- Identificação do dataset a ser utilizado
- Análise do dataset escolhido

- Preparação do dataset
- Desenvolvimento dos modelos
- Desenvolvimento do backend
- Desenvolvimento do frontend
- Avaliação do sistema desenvolvido

### 1.3 Divisão do documento

O presente documento está divido nos seguintes capítulos:

- Capítulo 1: Introdução
- Capítulo 2: Introdução do dataset
- Capítulo 3: Apresentação das tecnologias
- Capítulo 4: Técnicas usadas na recomendação
- Capítulo 5: Solução pensada
- Capítulo 6: Frontend
- Capítulo 7: Backend
- Capítulo 8: Avaliação do sistema de recomendação
- Capítulo 9: Conclusão

## Capítulo 2

# Introdução do dataset

Neste capítulo é identificado o dataset escolhido tal como as suas características e informações relevantes para o projeto desenvolvido.

### 2.1 Apresentação do database

O dataset escolhido pelo grupo está relacionado com filmes. Estes apresenta várias tabelas tais como:

- Ratings
- Movies
- Genome-tags
- Genome-scores
- Links
- Tags

Este dataset tem um tamanho de aproximadamente 1,1 GB.

#### 2.1.1 Apresentação das tabelas

Na Tabela “Ratings” foram encontradas as seguintes colunas:

- userId

- movieId
- rating
- timestamp

Na Tabela “Movies” foram encontradas as seguintes colunas:

- movieId
- title
- genres

Na Tabela “Genome-tags” foram encontradas as seguintes colunas:

- tagId
- tag

Na Tabela “Genome-scores” foram encontradas as seguintes colunas:

- movieId
- tagId
- relevance

Na Tabela “Links” foram encontradas as seguintes colunas:

- movieId
- imdbId
- tbmdbId

Na Tabela “Tags” foram encontradas as seguintes colunas:

- userId
- movieId
- tag
- timestamp

## 2.2 Dados estatísticos

Na Figura 2.1 é apresentada a distribuição dos filmes por género.

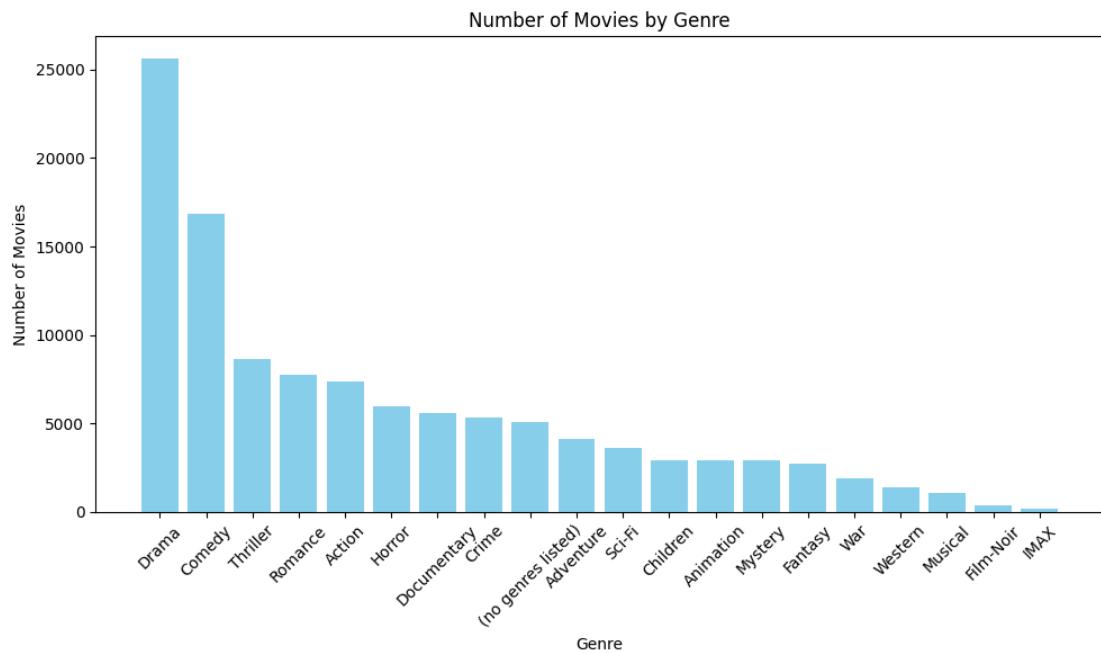


Figura 2.1: Distribuição dos filmes por género

Na Figura 2.2 é apresentada a distribuição das avaliações dos filmes.

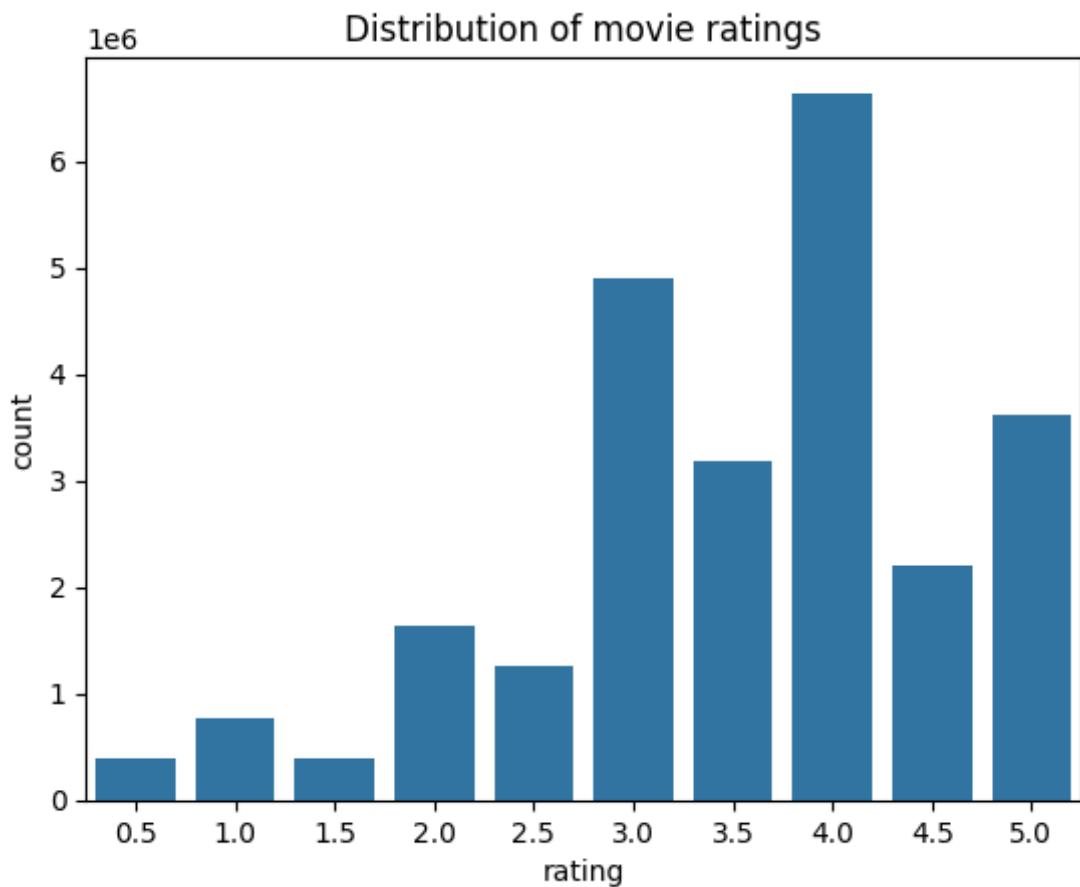


Figura 2.2: Distribuição das avaliações dos filmes

Na Figura 2.3 é apresentada a distribuição das avaliações médias por género.

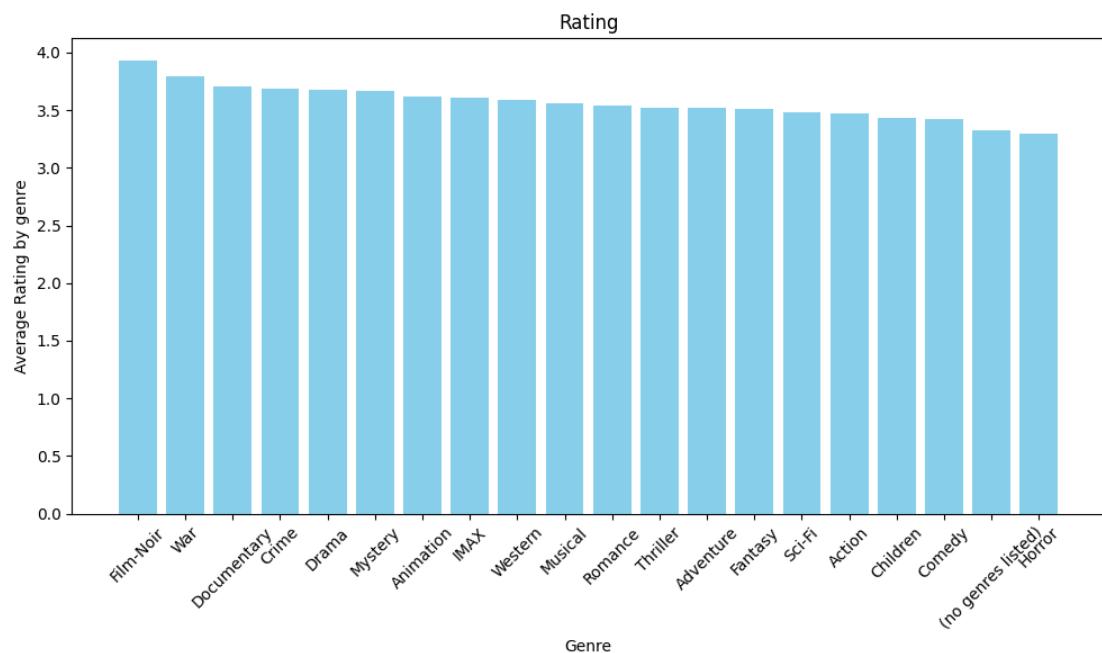


Figura 2.3: Distribuição das avaliações médias por género

Na Figura 2.4 é apresentada a distribuição dos números de filmes por cada ano, neste caso, entre 2019 e 1953.

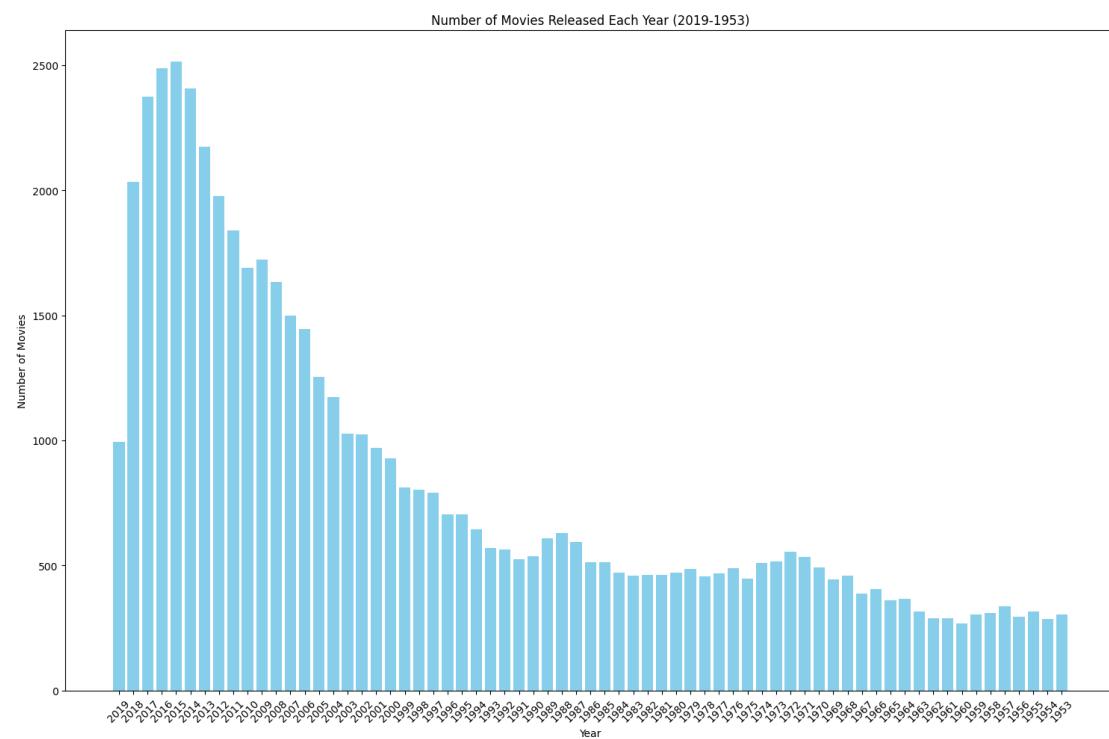


Figura 2.4: Distribuição dos números de filmes por cada ano (2019-1953)

Na Figura 2.5 é apresentada a distribuição dos números de filmes por cada ano, neste caso, entre 1952 e 1874.

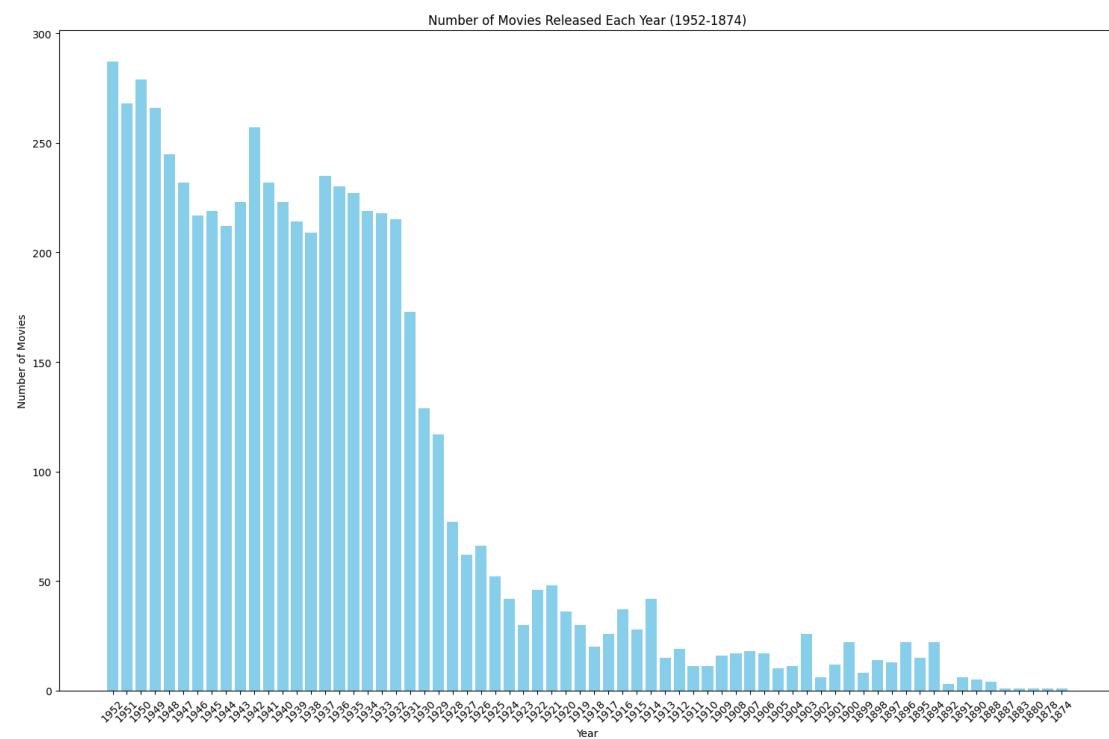


Figura 2.5: Distribuição dos números de filmes por cada ano (1952-1874)

Na Figura 2.6 é apresentada o top 20 de filmes em que é considerada uma classificação ponderada Bayesiana.

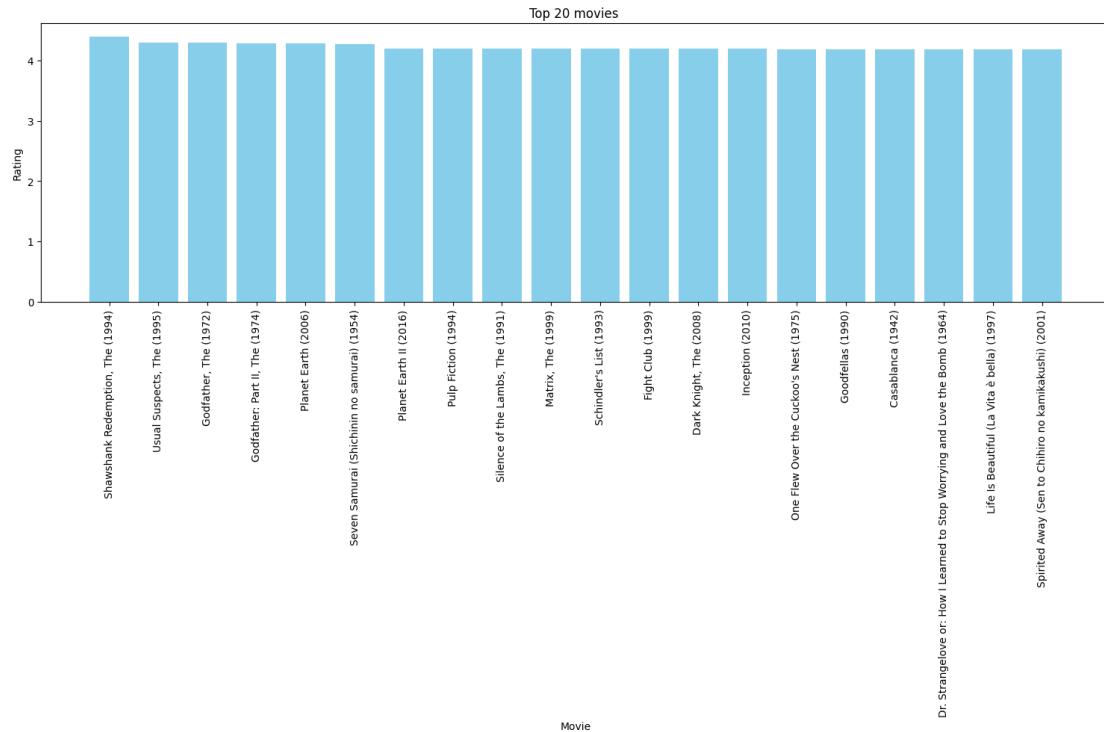


Figura 2.6: Top 20 de filmes — Classificação ponderada Bayesiana

Na Figura 2.7 é apresentada a distribuição do número de avaliações realizadas ao longo dos anos.

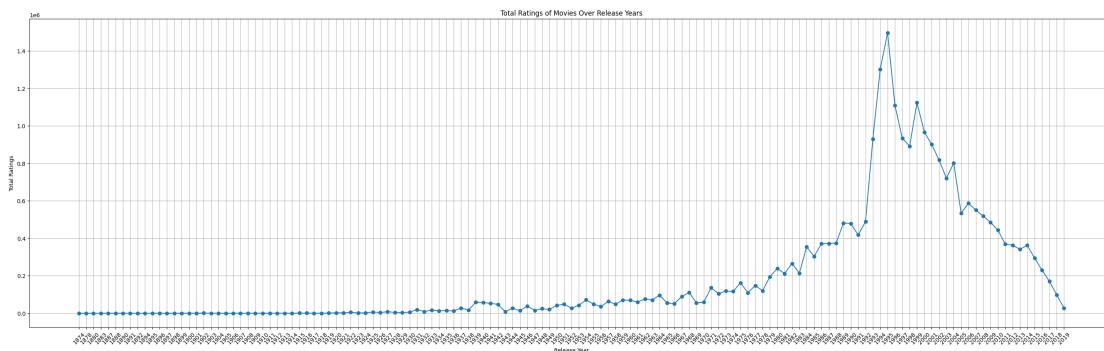


Figura 2.7: Distribuição do número de avaliações realizadas ao longo dos anos

Na Figura 2.8 é apresentada a distribuição das avaliações médias realizadas ao longo dos anos.

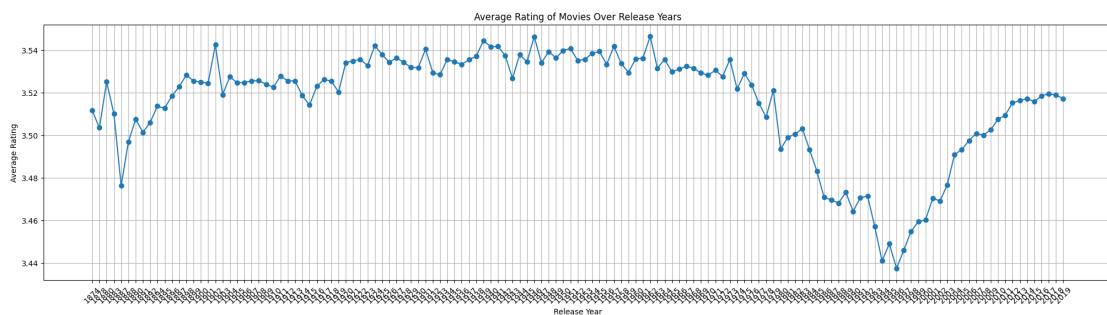


Figura 2.8: Distribuição das avaliações médias realizadas ao longo dos anos

Na Figura 2.9 é apresentada uma imagem na qual o tamanho das palavras depende da frequência de ocorrência.

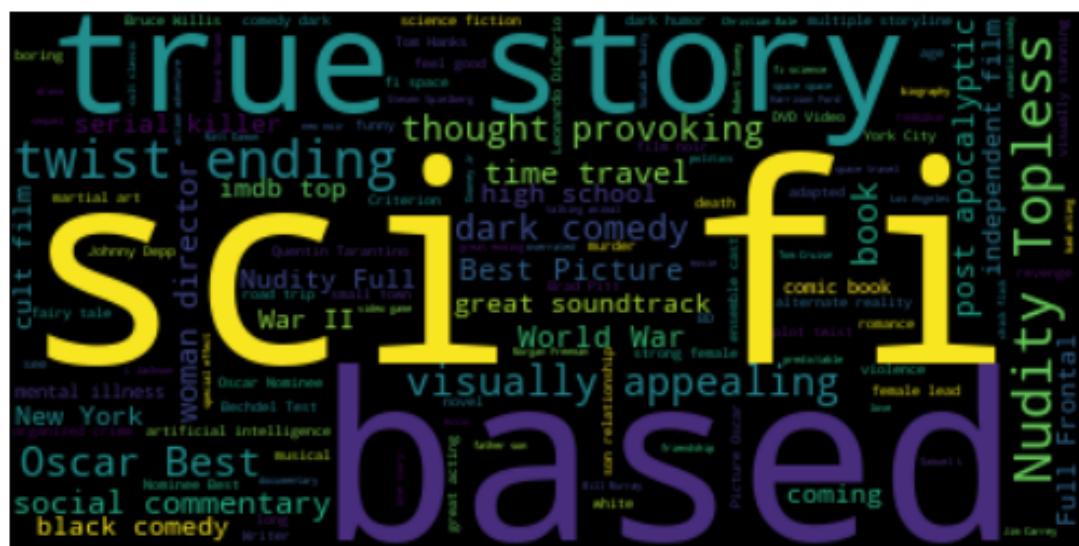


Figura 2.9: Frequência de ocorrência dos géneros

Na Figura 2.10 é apresentada a distribuição do número de avaliações que os utilizadores realizados.

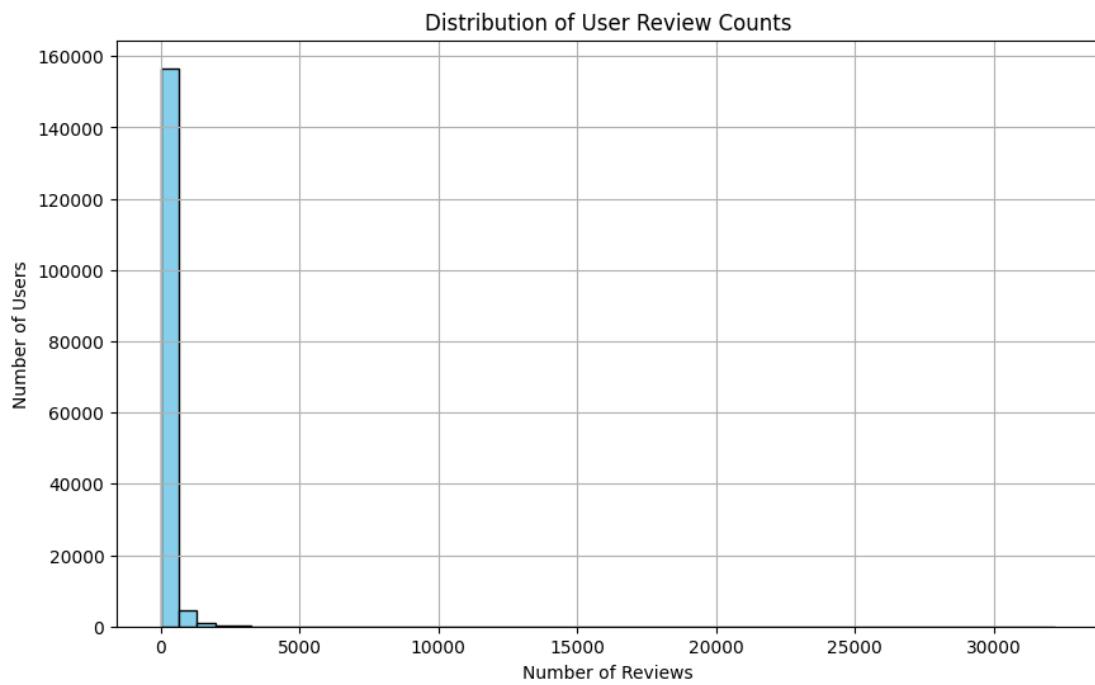


Figura 2.10: Distribuição do número de avaliações realizadas pelos utilizadores

Na Figura 2.11 é apresentada a distribuição do número de etiquetas pelos filmes.

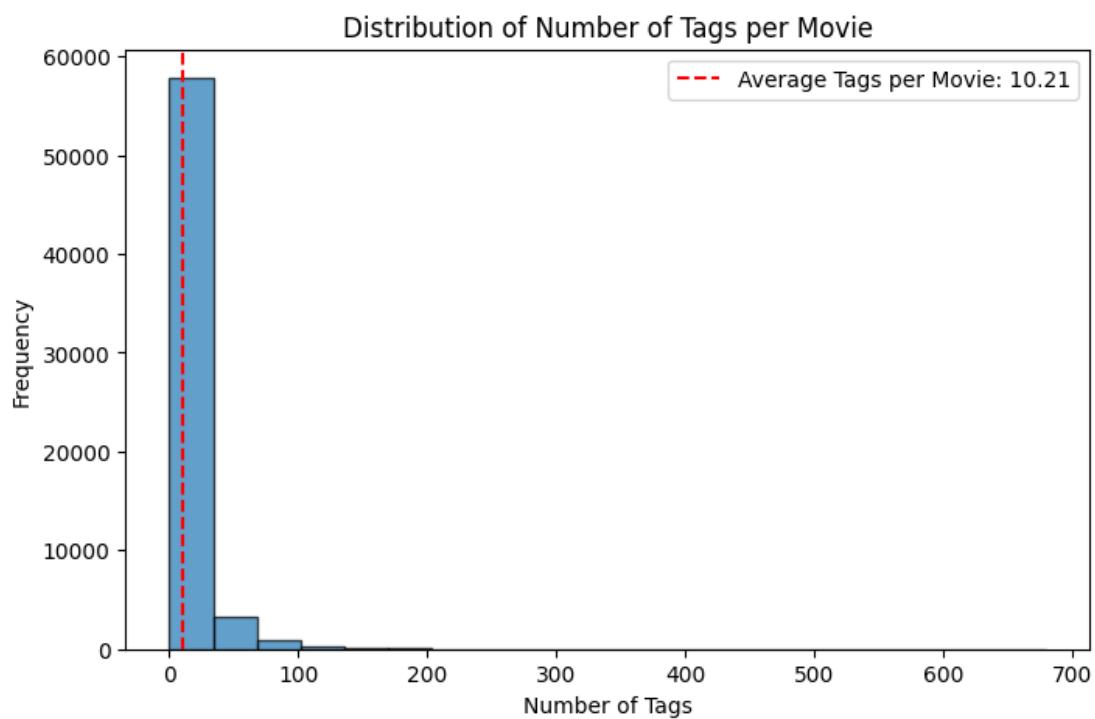


Figura 2.11: Distribuição do número de etiquetas pelos filmes

Na Figura 2.12 é apresentada o top 10 das etiquetas mais comuns.

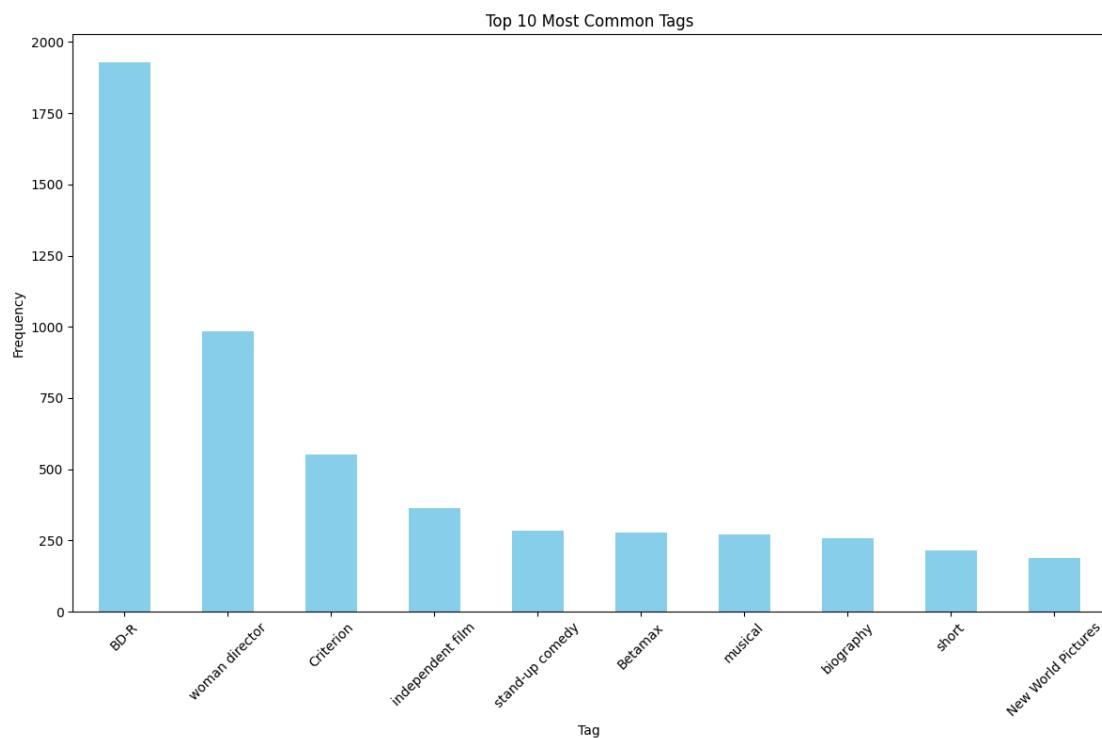


Figura 2.12: Top 10 das etiquetas mais comuns

Na Figura 2.13 é apresentada o top 20 das etiquetas mais comuns.

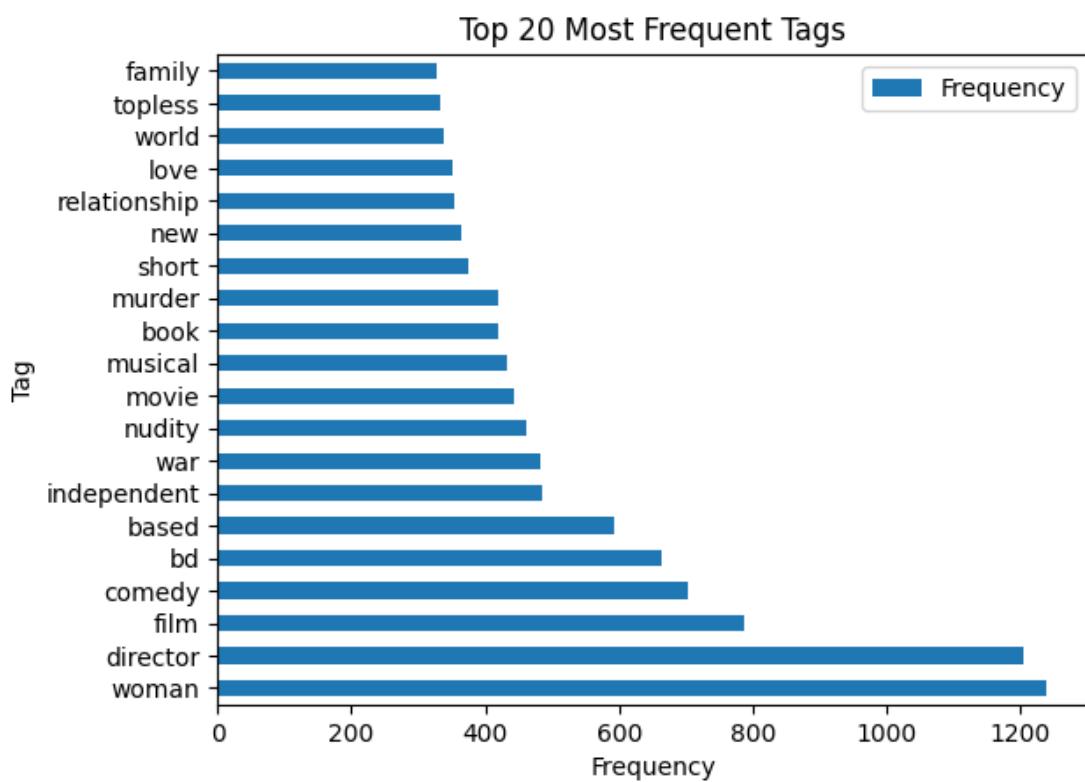


Figura 2.13: Top 20 das etiquetas mais comuns

Na Figura 2.14 é apresentada o top 20 das etiquetas mais comuns através do uso da pontuação da técnica TF-IDF.

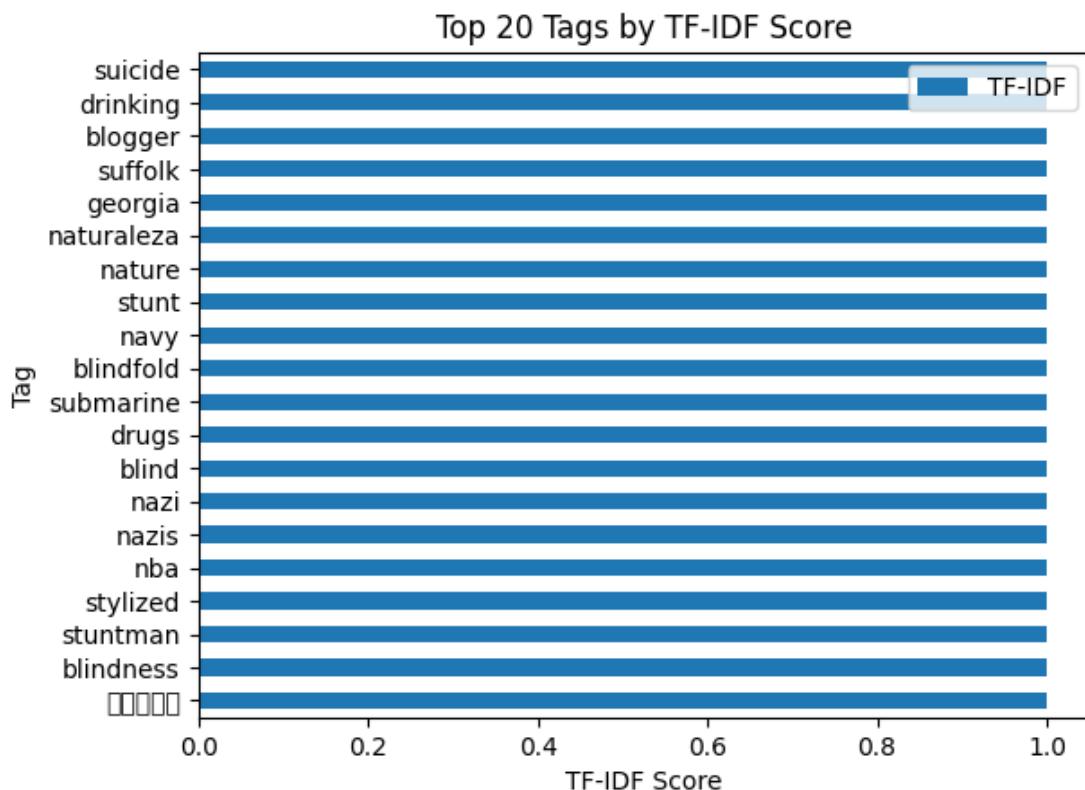


Figura 2.14: Top 20 das etiquetas mais comuns - TF-IDF

## 2.3 Preparação do dataset

Para se conseguir uma melhor apresentação e qualidade dos dados estes passaram por 3 processos, nomeadamente os seguintes:

- Agregação das tabelas “movies” e “links”
- Adição do endereço da imagem do filme na tabela “movies”
- Separação do título e do ano de filme a tabela “movies”

### 2.3.1 Agregação de duas tabelas

Em primeiro foi realizada a agregação das tabelas “movies” e “links”, no qual se obteve uma tabela com as seguintes características:

- movieId

- title
- genres
- imdbId

Na Listagem 2.1 é apresentado o código utilizado para se juntar as duas tabelas mencionadas em cima.

---

```
1 df_movie = pd.read_csv(os.path.join(script_dir,'../recommender/dataset/movies.csv'))
2 dtype_dict = {'imdbId': str}
3 df_links = pd.read_csv(os.path.join(script_dir,'../recommender/dataset/links.csv'),
4                         dtype=dtype_dict)
5 merged_df = pd.merge(df_movie, df_links[['movieId','imdbId']], on='movieId', how='
inner')
5 merged_df['imdbId'] = 'tt' + merged_df['imdbId'].astype(str)
```

---

Listagem 2.1: Agregação das tabelas.

### 2.3.2 Adição do endereço da imagem do filme

Com a nova tabela foram adicionadas duas novas colunas, nomeadamente:

- year
- url

Na Listagem 2.2 é apresentado o código utilizado para se realizar a obtenção do url de cada filme.

---

```

1  for ind in merged_df.index:
2      modified_string, extracted_value = remove_and_extract(merged_df.at[ind,'title'])
3
4      merged_df.at[ind,'title'] = modified_string
5      merged_df.at[ind,'year'] = extracted_value
6
7      url = f"https://moviesdatabase.p.rapidapi.com/titles/{merged_df.at[ind,'imdbId']}"
8          "
9
10     headers = {
11         "X-RapidAPI-Key": os.getenv('X-RapidAPI-Key'),
12         "X-RapidAPI-Host": os.getenv('X-RapidAPI-Host')
13     }
14
15     response = requests.get(url, headers=headers)
16     json = response.json()
17     if json['results'] == None:
18         merged_df.at[ind,'url'] = None
19         merged_df.to_csv(file_path, index=False)
20     elif json['results'][['primaryImage']] == None:
21         merged_df.at[ind,'url'] = None
22         merged_df.to_csv(file_path, index=False)
23     else:
24         merged_df.at[ind,'url'] = json['results'][['primaryImage']]['url']
25         merged_df.to_csv(file_path, index=False)
26     sleep(3.7)

```

---

Listagem 2.2: Adição do endereço da imagem do filme.

### 2.3.3 Separação do título e do ano do filme

Dado que no dataset original o ano e o título do filme vinham juntos, procedeu-se à separação destes. Na Listagem 2.3 é apresentado o código utilizado para se realizar essa operação.

---

```

1  def remove_and_extract(string):
2      parts = string.split('(')
3      if len(parts) > 1:
4          extracted = parts[-1].split(')')[0]
5          modified_string = string.replace(f"({extracted})", "").strip()
6          return modified_string, extracted.strip()
7      else:
8          return string.strip(), None

```

---

Listagem 2.3: Separação do título e do ano do filme.

### 2.3.4 Criação de uma coluna adicional

Por fim, foi adicionada uma nova coluna denominada “titleLower” com o objetivo de se poder realizar a comparação de strings com a strings introduzida pelo utilizador. a Listagem 2.4 é apresentado o código utilizado para se realizar essa operação.

---

```
1 for ind in merged_df.index:  
2     merged_df.at[ind,'titleLower'] = merged_df.at[ind,'title'].lower()  
3 merged_df.to_csv(file_path1, index=False)
```

---

Listagem 2.4: Criação da tabela “titleLower”.

## Capítulo 3

# Apresentação das tecnologias

Neste capítulo são apresentadas as tecnologias, plataformas e ferramentas utilizadas na construção do sistema, onde é realizada para cada uma, uma descrição breve e quais as suas funções no contexto do sistema.

### 3.1 Tecnologias, plataformas e ferramentas utilizadas

Para o projeto desenvolvido foram utilizadas as seguintes tecnologias, plataformas e ferramentas:

- Angular
- FastAPI
- TypeScript
- Python
- TablePlus
- AWS
- PostgreSQL

#### 3.1.1 Angular

Angular é um framework de desenvolvimento web front-end mantido pelo Google. Esta permite a construção de aplicativos web dinâmicos e escaláveis, recorrendo à linguagem TypeScript. A framework oferece uma arquitetura baseada em componentes, facilitando a modularidade e reutilização de código. Além disso, possui

recursos poderosos, como injecção de dependência, roteamento, e binding de dados bidirecionais, que simplificam o desenvolvimento de interfaces que sejam interativas e responsivas.

### 3.1.2 FastAPI

FastAPI é um framework web rápido, fácil de usar e altamente produtivo para a criação de APIs em Python. Ele é construído sobre as funcionalidades assíncronas do Python 3.7+ e utiliza a tipagem de dados do Python através do pydantic para validação de entrada e saída de dados. FastAPI é conhecido por sua alta performance, gerando automaticamente documentação interativa para suas APIs baseada em padrões abertos como OpenAPI e JSON Schema.

### 3.1.3 TypeScript

TypeScript é uma linguagem de programação desenvolvida pela Microsoft que adiciona tipagem estática opcional ao JavaScript. Isso permite aos desenvolvedores detetar erros em tempo de compilação, melhorando a robustez e manutenibilidade do código. TypeScript é compatível com JavaScript existente e oferece recursos avançados, como interfaces, tipos genéricos e suporte a classes, facilitando o desenvolvimento de aplicativos complexos e escaláveis.

### 3.1.4 Python

Python é uma linguagem de programação interpretada, de alto nível e de propósito geral. Conhecida pela sua simplicidade e legibilidade, Python é amplamente utilizado numa variedade de domínios, incluindo desenvolvimento web, ciência de dados, automação de tarefas e inteligência artificial. A sua vasta biblioteca padrão e a abundância de pacotes de terceiros disponíveis facilitam o desenvolvimento rápido de aplicações robustas e escaláveis.

### 3.1.5 TablePlus

TablePlus é uma ferramenta de gestão de bases de dados multiplataforma que oferece uma interface gráfica intuitiva para interagir com diversos sistemas de gestão de bases de dados, como PostgreSQL, MySQL, SQLite, entre outros. Esta oferece recursos avançados de edição de dados, visualização de esquema, consulta SQL, além de suporte para múltiplas conexões e temas personalizáveis, tornando-o uma escolha popular entre desenvolvedores e administradores de bases de dados.

### 3.1.6 AWS

Amazon Web Services é uma plataforma líder de serviços em nuvem que oferece uma ampla gama de serviços de computação, armazenamento, bases de dados, análise, machine learning, entre outros. AWS permite que empresas e desenvolvedores construam, implantem e dimensionem facilmente aplicativos e serviços na nuvem, eliminando a necessidade de investimentos em infraestrutura física. Com recursos como elasticidade, segurança avançada e escalabilidade sob demanda, AWS é amplamente utilizado por empresas de todos os tamanhos em todo o mundo.

### 3.1.7 PostgreSQL

PostgreSQL é um sistema de gestão de bases de dados relacionais de código aberto e altamente confiável. Ele oferece recursos avançados de conformidade com padrões SQL, suporte a transações ACID, replicação, particionamento e extensibilidade, tornando-o uma escolha popular para aplicações que exigem alta disponibilidade, confiabilidade e desempenho. PostgreSQL é conhecido por sua comunidade ativa, atualizações frequentes e forte compromisso com padrões abertos.

## Capítulo 4

# Técnicas usadas na recomendação

Neste capítulo são apresentadas as várias técnicas usadas na recomendação de filmes ao utilizador, tal como outros que podem ser utilizadas para serem formuladas as recomendações.

### 4.1 Processamento de Dados

Antes de prosseguirmos para a aplicação de técnicas de recomendação, é necessário processar os dados e colocá-los nos formatos apropriados. Foram identificados alguns problemas:

- **Problema 1** - Alguns filmes possuem classificações muito altas (5 estrelas); no entanto, o número de classificações é extremamente baixo (em alguns casos, apenas 1). Esta situação nem sempre representa a popularidade de um item e pode ser enganadora.
- **Problema 2** - Reunir os dados e colocá-los no mesmo plano, ou seja, no mesmo dataframe, para serem mais facilmente manipulados e utilizados pelo sistema de recomendação.

### 4.2 Recomendação não personalizada

A recomendação não personalizada é uma abordagem que sugere itens com base em critérios gerais, sem considerar as preferências individuais dos utilizadores. Este tipo de recomendação baseia-se em características intrínsecas dos itens, como popularidade, tendências gerais ou classificações agregadas, sendo aplicada uniformemente a todos os

utilizadores. Esta secção irá explorar os princípios subjacentes à recomendação não personalizada, bem como as metodologias comuns empregadas para a sua implementação.

No caso do nosso conjunto de dados de filmes, as recomendações terão em consideração a classificação dos filmes, o ano de lançamento, década e o género. Conforme mencionado na Secção 4.1, existem filmes com classificações bastante altas; no entanto, o número de vezes que foram avaliados é relativamente baixo. Para resolver este problema, aplicámos a classificação bayesiana [1], que pode ser representada pela seguinte fórmula:

$$\text{BayesianAverage}(\mathbf{WR}) = \left( \frac{\mathbf{v}}{\mathbf{v} + \mathbf{m}} \cdot \mathbf{R} \right) + \left( \frac{\mathbf{m}}{\mathbf{v} + \mathbf{m}} \cdot \mathbf{C} \right) \quad (4.1)$$

De modo a proporcionar ao utilizador uma ampla variedade de conteúdos com base em diferentes critérios, sem a necessidade de conhecimento prévio sobre o utilizador, existem quatro formas de filtrar as recomendações não personalizadas.

- Melhores filmes overall
- Melhores filmes de um determinado género
- Melhores filmes de um ano
- Melhores filmes de uma década

#### 4.2.1 Melhores filmes overall

Deste modo, na ausência de critérios específicos para filtrar os filmes, qualquer utilizador que visite o website verá os seguintes filmes, que apresentam a classificação ponderada mais elevada, demonstrado na Figura 4.1.

						Num_ratings	Bayesian_rating
moviedb	title		genres	year	decade		
318	Shawshank Redemption, The		Crime, Drama	1994.0	1990.0	317	3.849204
356	Forrest Gump	Comedy, Drama, Romance, War	1994.0	1990.0	329	3.777805	
296	Pulp Fiction	Comedy, Crime, Drama, Thriller	1994.0	1990.0	307	3.766295	
593	Silence of the Lambs, The	Crime, Horror, Thriller	1991.0	1990.0	279	3.750706	
2571	Matrix, The	Action, Sci-Fi, Thriller	1999.0	1990.0	278	3.750129	

Figura 4.1: Não personalizado sem critérios

#### 4.2.2 Melhores filmes de um determinado género

Um visitante poderá também filtrar por género, querendo obter os melhores filmes de um determinado género. Eis um exemplo para filmes de crime, demonstrado na Figura 4.2.

Best movie for genre 'Crime':						
	moviedb_id	title	genres	Num_ratings	Bayesian_rating	
0	318	The Shawshank Redemption	[Crime, Drama]	317	3.849204	
2	296	Pulp Fiction	[Comedy, Crime, Drama, Thriller]	307	3.766295	
3	593	The Silence of the Lambs	[Crime, Horror, Thriller]	279	3.750706	
5	2959	Fight Club	[Action, Crime, Drama, Thriller]	218	3.742897	
7	858	The Godfather	[Crime, Drama]	192	3.721965	

Figura 4.2: Não personalizado sem critérios

#### 4.2.3 Melhores filmes de um ano

Um visitante poderá também filtrar por ano de lançamento, querendo obter os melhores filmes de um determinado ano. Eis um exemplo para filmes de 2010, demonstrado na Figura 4.3.

	moviedb_id	title	Num_ratings	Bayesian_rating	year
35	79132	Inception	143	3.633437	2010.0
127	78499	Toy Story 3	55	3.559459	2010.0
129	74458	Shutter Island	67	3.559083	2010.0
156	81845	The King's Speech	58	3.551971	2010.0
199	81834	Harry Potter and the Deathly Hallows: Part 1	47	3.542962	2010.0

Figura 4.3: Não personalizado conforme o ano

#### 4.2.4 Melhores filmes de uma década

Um visitante poderá também filtrar por género, querendo obter os melhores filmes de uma determinada década. Eis um exemplo para filmes da década de 2010, demonstrado na Figura 4.4.

<b>movield</b>		<b>title</b>	<b>Num_ratings</b>	<b>Bayesian_rating</b>	<b>decade</b>
35	79132	Inception	143	3.633437	2010.0
108	91529	Dark Knight Rises, The	76	3.565972	2010.0
113	109487	Interstellar	73	3.563700	2010.0
114	112852	Guardians of the Galaxy	59	3.563327	2010.0
127	78499	Toy Story 3	55	3.559459	2010.0

Figura 4.4: Não personalizado conforme a década

### 4.3 Recomendação personalizada

A recomendação personalizada é uma abordagem que visa sugerir itens com base nas preferências individuais dos utilizadores, utilizando informações detalhadas sobre os seus comportamentos e históricos de interação. Ao contrário da recomendação não personalizada, que se baseia em critérios gerais, a recomendação personalizada procura compreender e antecipar os gostos específicos de cada utilizador, proporcionando sugestões mais relevantes e adaptadas. Nesta secção, serão explorados os métodos e algoritmos utilizados para implementar sistemas de recomendação personalizada, incluindo técnicas de filtragem colaborativa, filtragem baseada em conteúdo, filtragem baseada em conhecimento e modelos híbridos.

Serão apresentadas as várias técnicas utilizadas para recomendar filmes, incluindo, entre outras:

- Filtragem Colaborativa
- Filtragem Baseada em Conteúdo
- Filtragem Baseada em Conhecimento
- Abordagem Híbrida

#### 4.3.1 Filtragem Colaborativa

A recomendação colaborativa é baseada nas preferências e interações do utilizador. As interações que os utilizadores têm com o produto são analisadas em termos da sua semelhança e, com base nas preferências de utilizadores ou itens semelhantes, é feita a recomendação. Por outras palavras, esta técnica requer a existência de informação e que estas interações tenham ocorrido. Este problema é chamado de cold start, que será mencionado mais adiante neste artigo. Um exemplo que pode ser dado é o seguinte: se um

determinado utilizador e outro utilizador tiverem um histórico de compras semelhante, se o primeiro utilizador comprar um determinado produto, o sistema irá recomendar o mesmo produto ao segundo utilizador.

Na Listagem 4.1 é apresentado o código utilizado para ser obtida a recomendação colaborativa. Começa-se por ler os dados de avaliações e informações dos filmes a partir de uma base de dados SQL, calculando o número total de utilizadores e filmes únicos. Em seguida, cria mapeamentos bidirecionais entre os identificadores dos utilizadores, filmes e índices numéricos sequenciais, convertendo os identificadores em índices numéricos. Com esses índices, constrói uma matriz esparsa de avaliações de filmes. Utiliza a similaridade coseno para calcular a similaridade entre o utilizador de interesse e todos os outros utilizadores, identificando os 5 utilizadores mais similares. Para esses utilizadores similares, encontra filmes não avaliados pelo utilizador de interesse, avaliados pelos utilizadores similares, e calcula um score de recomendação com base na similaridade e na avaliação desses filmes. Os filmes recomendados são ordenados pelo score e os top 5 são selecionados. Finalmente, coleta os detalhes dos filmes recomendados a partir do Data-Frame de filmes, gerando uma lista de até 5 filmes recomendados com os seus respetivos detalhes.

---

```

1   df = pd.read_sql_query("""SELECT * from rating""", con=engine)
2   df_movies = pd.read_sql_query("""SELECT * from movies""", con=engine)
3   M = df['userId'].nunique()
4   N = df['movieId'].nunique()
5
6   user_mapper = dict(zip(np.unique(df["userId"]), list(range(M))))
7   movie_mapper = dict(zip(np.unique(df["movieId"]), list(range(N))))
8
9   user_inv_mapper = dict(zip(list(range(M)), np.unique(df["userId"])))
10  movie_inv_mapper = dict(zip(list(range(N)), np.unique(df["movieId"])))
11
12  user_index = [user_mapper[i] for i in df['userId']]
13  item_index = [movie_mapper[i] for i in df['movieId']]
14
15  X = csr_matrix((df["rating"], (user_index,item_index)), shape=(M,N))
16  ser_index = user_mapper[user_id]
17  similarities = cosine_similarity(X[user_index], X)
18
19  similar_users_indices = similarities.argsort() [0][-5:-1][::-1]
20
21  recommended_movies = {}
22
23  for similar_user_index in similar_users_indices:
24      unrated_movies = np.where(np.logical_and(X[user_index].toarray()[0] == 0,
25                                              X[similar_user_index].toarray()[0] != 0))[0]
26
27      for movie_index in unrated_movies:
28          if movie_index not in recommended_movies:
29              recommended_movies[movie_index] = similarities[0,
30                                              similar_user_index] * X[similar_user_index, movie_index]
31          else:
32              recommended_movies[movie_index] += similarities[0,
33                                              similar_user_index] * X[similar_user_index, movie_index]
34
35  recommended_movies = sorted(recommended_movies.items(), key=lambda x: x[1],
36                               reverse=True)
37
38  recommended_movie_ids = [movie_inv_mapper[movie_index] for movie_index, _ in recommended_movies]
39
40  recommendations = recommended_movie_ids[:5]
41
42  movie_details = []
43  for movie_id in recommendations:
44      movie_info = df_movies.loc[movie_id, ['title', 'url', 'genres', 'imdbId', 'year']]
45      movie_details.append({
46          'title': movie_info['title'],
47          'url': movie_info['url'],
48          'genres': movie_info['genres'],
49          'imdbId': movie_info['imdbId'],
50          'year': movie_info['year']
51      })

```

---

Listagem 4.1: Filtragem baseada em colaborativa.

### 4.3.2 Filtragem Baseada em Conteúdo

A filtragem baseada em conteúdo utiliza o perfil do utilizador e as descrições dos itens para recomendar outros itens. Esta técnica utiliza as preferências do utilizador para sugerir itens semelhantes com base nas características do item, ou seja, recomenda elementos com base nas semelhanças das características com os itens que o utilizador em questão prefere. Um exemplo que pode ser dado é o seguinte: um utilizador avalia positivamente um artigo de uma determinada marca e o sistema, utilizando esta informação, pode e provavelmente irá recomendar outros artigos da mesma marca.

Na Listagem 4.2 é apresentado o código utilizado para selecionar cinco filmes parecidos com aquele introduzido. No código apresentado na Listagem referida anterior começa importando os dados das tabelas, “tags” e “movies” da base de dados. Por sua vez, as tags são então transformadas em minúsculas e convertidas para o tipo de dados string para uma melhor manipulação. Dado isso, estas são agrupadas por “movieId” e concatenadas numa única string. Essa informação é então mesclada com o DataFrame dos filmes usando a coluna “movieId”.

Após a junção das duas tabelas, é calculada a matriz de frequência de termos e inversão de documento (TF-IDF) das tags dos filmes recorrendo à classe TfIdfVectorizer do scikit-learn. Com base nisso, a similaridade do cosseno é calculada entre todos os pares de filmes com base nas suas representações TF-IDF.

Posteriormente, é realizada uma busca por filmes semelhantes ao filme escolhido no momento. No caso de não houver correspondência, uma lista vazia é retornada, caso contrário, os cinco filmes mais similares são selecionados e por sua vez retornados ao utilizador.

---

```

1  tags_df = pd.read_sql_query("""SELECT * from tags""", con=engine)
2  movies_df = pd.read_sql_query("""SELECT * from movies""", con=engine)
3
4  tags_df.set_index('movieId', inplace=True)
5  movies_df.set_index('movieId', inplace=True)
6
7  tags_df['tag'] = tags_df['tag'].str.lower()
8  tags_df['tag'] = tags_df['tag'].astype(str)
9
10 movie_tags = tags_df.groupby('movieId')['tag'].apply(lambda x: ' '.join(x)).
    reset_index()
11 movies = pd.merge(movies_df, movie_tags, on='movieId', how='left')
12 movies['tag'] = movies['tag'].fillna(' ')
13
14 tfidf = TfidfVectorizer(stop_words='english')
15 tfidf_matrix = tfidf.fit_transform(movies['tag'])
16
17 cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
18 matching_rows = movies[movies['title'].str.contains(title, case=False, na=False)]
19 if len(matching_rows) == 0:
20     return []
21 idx = matching_rows.index[0]
22
23 sim_scores = list(enumerate(cosine_sim[idx]))
24 sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
25 sim_scores = sim_scores[1:6]
26
27 movie_indices = [i[0] for i in sim_scores]

```

---

Listagem 4.2: Filtragem baseada em conteúdo.

### 4.3.3 Filtragem Baseada em Conhecimento

A filtragem baseada em conhecimento é uma técnica de recomendação que utiliza informações explícitas sobre as preferências e necessidades dos utilizadores para sugerir itens relevantes. Este método é particularmente eficaz em contextos onde é necessário um entendimento profundo das características dos itens e das preferências do utilizador.

No caso da nossa aplicação, a filtragem baseada em conhecimento é realizada com base nos géneros de filmes que o utilizador gosta e não gosta. Esta informação é obtida através de um inquérito diretamente apresentado ao utilizador. Com base nas respostas fornecidas, os filmes que pertencem a géneros indesejados pelo utilizador são automaticamente excluídos das recomendações, assegurando que apenas conteúdos pertinentes são sugeridos.

Na Listagem 4.3 é apresentado o código utilizado para obter-se uma recomendação baseada em conhecimento. O código começa por ler os dados das tabelas ratings e movies

a partir de uma base de dados SQL, carregando-os em DataFrames. Em seguida, define uma função para criar um DataFrame com avaliações ponderadas dos filmes. A função `create_weighted_rating_df` combina os dados de filmes e avaliações, calcula o número de avaliações e a média das avaliações para cada filme, e aplica uma classificação Bayesiana para ponderar essas avaliações. Para isso, calcula-se a média global das avaliações ( $C$ ) e define-se um limiar de 500 avaliações ( $m$ ). A classificação Bayesiana é calculada e utilizada para ordenar os filmes de forma ponderada, sendo a coluna `Average_rating` substituída pela `Bayesian_rating`. Finalmente, as colunas são renomeadas e os géneros dos filmes são separados em listas.

A função `recommend_movies_based_on_genres` recomenda filmes filtrando os géneros preferidos e não preferidos do utilizador. Esta função faz uma cópia do DataFrame e itera sobre cada linha, verificando se os géneros do filme atual coincidem com os géneros preferidos e não incluem os géneros não preferidos do utilizador. Se um filme satisfaz estas condições, é adicionado à lista de recomendações até se atingir um máximo de 5 filmes. A função retorna um DataFrame com os filmes recomendados, incluindo o título, ano, URL, número de avaliações e a classificação ponderada.

Para utilizar estas funções, o código obtém os géneros preferidos e não preferidos do utilizador, cria o DataFrame de avaliações ponderadas e chama a função de recomendação de filmes. Se o utilizador não for encontrado, é levantada uma exceção HTTP com o código 404. O resultado final é um DataFrame contendo até 5 filmes recomendados que satisfazem as preferências do utilizador em termos de géneros.

---

```

1 ratings_df = pd.read_sql_query("""SELECT * from ratings""", con=engine)
2 movies_df = pd.read_sql_query("""SELECT * from movies""", con=engine)
3
4 def create_weighted_rating_df(movies_df, ratings_df):
5     movies_rating_user_df = pd.merge(movies_df, ratings_df, on="movieId", how="inner")
6     movies_rating_df = movies_rating_user_df[['movieId', 'title', 'rating', 'genres', 'year', 'url']].groupby(['movieId', 'title', 'genres', 'year', 'url'])['rating'].agg(['count', 'mean']).round(1)
7     movies_rating_df.sort_values('count', ascending=False, inplace=True)
8     movies_rating_df.rename(columns={'count': 'Num_ratings', 'mean': 'Average_rating'}, inplace=True)
9     C = round(ratings_df['rating'].mean(), 2)
10    m = 500
11    movies_rating_df['Bayesian_rating'] = (movies_rating_df['Num_ratings'] / (
12        movies_rating_df['Num_ratings'] + m)) * movies_rating_df['Average_rating'] +
13        ((m / (movies_rating_df['Num_ratings'] + m)) * C)
14    movies_rating_df.drop(columns='Average_rating', inplace=True)
15    movies_rating_df.sort_values(by='Bayesian_rating', ascending=False, inplace=True)
16    movies_rating_df.rename(columns={'Num_ratings': 'count', 'Bayesian_rating': 'weighted_rating'}, inplace=True)
17    movies_rating_df.reset_index(inplace=True)
18    movies_rating_df['genres'] = movies_rating_df['genres'].str.split('|')
19    return movies_rating_df
20
21 def recommend_movies_based_on_genres(df, preferred_genres=None, disliked_genres=None):
22     df_copy = df.copy()
23     recommended_movies_ids = []
24
25     for idx, row in df_copy.iterrows():
26         movie_genres = set(row['genres'])
27         if disliked_genres:
28             if movie_genres.intersection(set(disliked_genres)):
29                 continue
30         if preferred_genres:
31             if not movie_genres.intersection(set(preferred_genres)):
32                 continue
33         recommended_movies_ids.append(idx)
34     recommended_movies_df = df.loc[recommended_movies_ids]
35     recommended_movies_df = recommended_movies_df[['title', 'year', 'url', 'count', 'weighted_rating']]
36     return recommended_movies_df
37
38 user = await session.get(User, user_id)
39 if not user:
40     raise HTTPException(status_code=404, detail="user not found")
41 preferred_genres = user.genresLike
42 disliked_genres = user.genresDislike
43 df = create_weighted_rating_df(movies_df, ratings_df)
44 recommended_movies_df = recommend_movies_based_on_genres(df,
45     preferred_genres, disliked_genres)

```

---

Listagem 4.3: Filtragem baseada em conteúdo.

#### 4.3.4 Abordagem Híbrida

A abordagem híbrida é uma técnica avançada de recomendação que combina múltiplos métodos para aproveitar as vantagens de cada um e mitigar as suas limitações. Ao integrar diversas estratégias de recomendação, a abordagem híbrida proporciona sugestões mais precisas e abrangentes.

No contexto do nosso sistema de recomendação, a abordagem híbrida utiliza três técnicas principais: filtragem colaborativa, filtragem baseada em conteúdo e filtragem baseada em conhecimento.

- **Filtragem colaborativa :** Este método analisa as preferências de múltiplos utilizadores para identificar padrões e sugerir itens que utilizadores com gostos semelhantes apreciaram.
- **Filtragem baseada em conteúdo :** Esta técnica recomenda itens com base nas características dos próprios itens, como géneros, diretores ou atores, alinhando-os com os interesses previamente expressos pelo utilizador.
- **Filtragem baseada em conhecimento:** Como mencionado, este método utiliza informações explícitas sobre as preferências do utilizador, recolhidas por meio de um inquérito, para excluir filmes de géneros indesejados e priorizar os géneros preferidos.

Ao combinar estas três abordagens, o sistema de recomendação híbrido pode fornecer sugestões mais robustas e adaptadas às necessidades específicas de cada utilizador, ao mesmo tempo que minimiza as limitações de cada método individual. Na Listagem ?? é apresentado o código utilizado para obter-se uma recomendação híbrida.

---

```

1 def hybrid_based_recommendation(knowledge_recommendations_ids,
2                                 content_recommendations_ids, collaborative_recommendations_ids):
3
4     knowledge_weight = 1
5     content_weight = 0.5
6     collaborative_weight = 0.8
7
8     # Calculate the number of movies to be recommended from each technique
9     num_movies_per_technique = 5
10
11    # Apply weights to each technique's recommendations
12    weighted_knowledge_recommendations = knowledge_recommendations_ids[:num_movies_per_technique] * knowledge_weight
13    weighted_content_recommendations = content_recommendations_ids[:num_movies_per_technique] * content_weight
14    weighted_collaborative_recommendations = collaborative_recommendations_ids[:num_movies_per_technique] * collaborative_weight
15
16    # Combine recommendations from all techniques
17    combined_recommendations = weighted_knowledge_recommendations +
18        weighted_content_recommendations +
19        weighted_collaborative_recommendations
20
21    # # Ensure the total number of recommendations is not more than 10
22    combined_recommendations = combined_recommendations[:10]
23
24    return combined_recommendations

```

---

## 4.4 Alternativas para se realizar a recomendação - Trabalho futuro

Esta secção visa discutir diversas alternativas que podem ser exploradas no futuro para melhorar e diversificar os métodos de recomendação utilizados no nosso sistema. Abordaremos diferentes abordagens e técnicas que poderiam ser implementadas para aprimorar a precisão, relevância e personalização das recomendações oferecidas aos utilizadores.

Algumas das alternativas possíveis incluem:

- **Aprendizagem Federada:** Explorar a aprendizagem federada como uma abordagem para treinar modelos de recomendação distribuídos em vários dispositivos ou servidores, preservando a privacidade dos dados do utilizador, o que é crucial quando se trata de informações sensíveis como preferências de filmes.

- **Incorporação de Contexto Temporal:** Considerar o contexto temporal ao recomendar itens, considerando mudanças sazonais, tendências de popularidade e eventos culturais, o que pode influenciar significativamente as preferências dos utilizadores relativamente aos filmes.
- **Fusão de Dados Multi-fonte:** integrar dados de múltiplas fontes, como redes sociais, histórico de compras e avaliações de produtos, para enriquecer a compreensão do perfil do utilizador e melhorar a precisão das recomendações de filmes, oferecendo uma visão mais abrangente dos interesses e comportamentos dos utilizadores.
- **Exploração de Técnicas de Aprendizagem Semi-supervisionada:** investigar abordagens semi-supervisionadas para recomendação, aproveitando dados rotulados e não rotulados para treinar modelos mais robustos e adaptáveis, o que pode ser especialmente útil quando se dispõe de uma abundância de dados não rotulados sobre preferências de filmes.
- **Utilização de Algoritmos de Reinforcement Learning:** Explorar o uso de algoritmos de reinforcement learning para adaptar as recomendações com base nas interações em tempo real do utilizador com o sistema de recomendação de filmes, melhorando a personalização e a relevância das sugestões ao longo do tempo.
- **Incorporação de Explicações de Recomendação:** Desenvolver métodos para fornecer explicações transparentes sobre como as recomendações são geradas, aumentando a confiança e compreensão do utilizador sobre o sistema de recomendação de filmes, o que pode incentivar uma maior aceitação e adoção das sugestões oferecidas.

Cada uma destas alternativas oferece oportunidades para expandir e aprimorar a capacidade do nosso sistema de recomendação em atender às necessidades e expectativas dos utilizadores de forma mais eficaz e personalizada.

# Capítulo 5

## Solução pensada

Neste capítulo é apresentada a solução pensada onde são apresentados os mockups realizados, o diagrama de componentes, diagrama de casos de uso e modelo de domínio.

### 5.1 Diagrama de componentes

Na Figura 5.1 é apresentado o diagrama de componentes desenvolvido, neste é possível identificar e observar as respetivas dependências e como é feito a ligação entre os vários componentes do sistema.



Figura 5.1: Diagrama de componentes

### 5.2 Mockup

Nesta subsecção é descrito o mockup realizado para a aplicação WEB. Este mockup é uma representação visual estática da interface da aplicação, no qual, é apresentada uma visão tangível do design proposto, permitindo análises detalhadas e iterações antes mesmo do desenvolvimento efetivo começar.

Na Figura 5.2 é apresentado a interface relacionada com a página dedicada para o registo do utilizador, onde é pretendido que o utilizador introduza as seguintes informações:

- Email
- Password
- Username

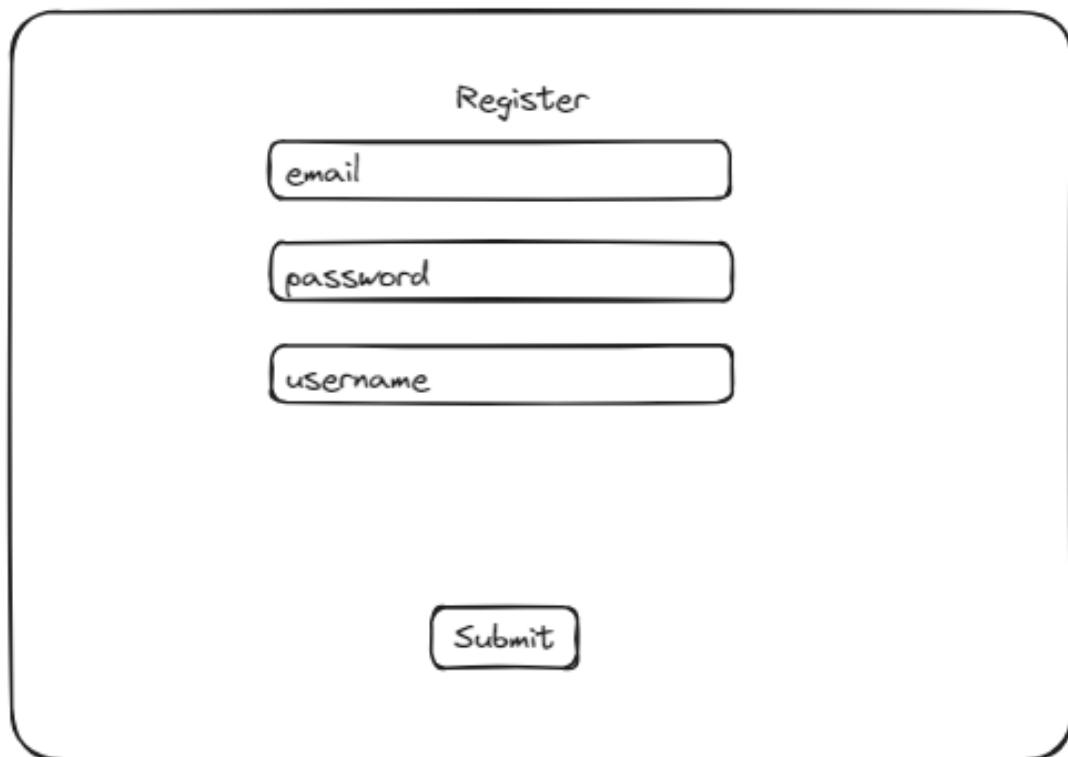


Figura 5.2: Realizar o registo do utilizador

Depois do utilizador ter introduzido os seus dados, de seguida, estes são verificados e este é reencaminhado para a interface disponibilizada na Figura 5.3. Nesta interface o utilizador escolhe os seus géneros de filmes preferidos.

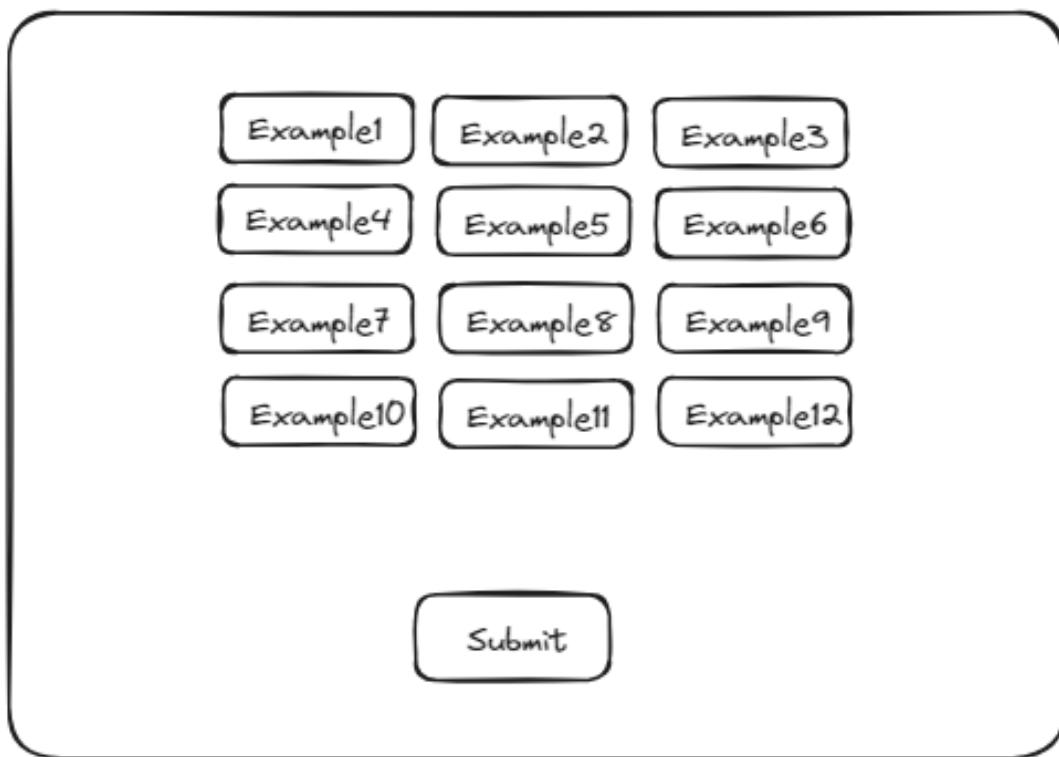


Figura 5.3: Escolha do géneros

Na Figura 5.4 é apresentado a interface relacionada com a página dedicada para o acesso do utilizador, onde é pretendido que o utilizador introduza as seguintes informações:

- Email
- Password

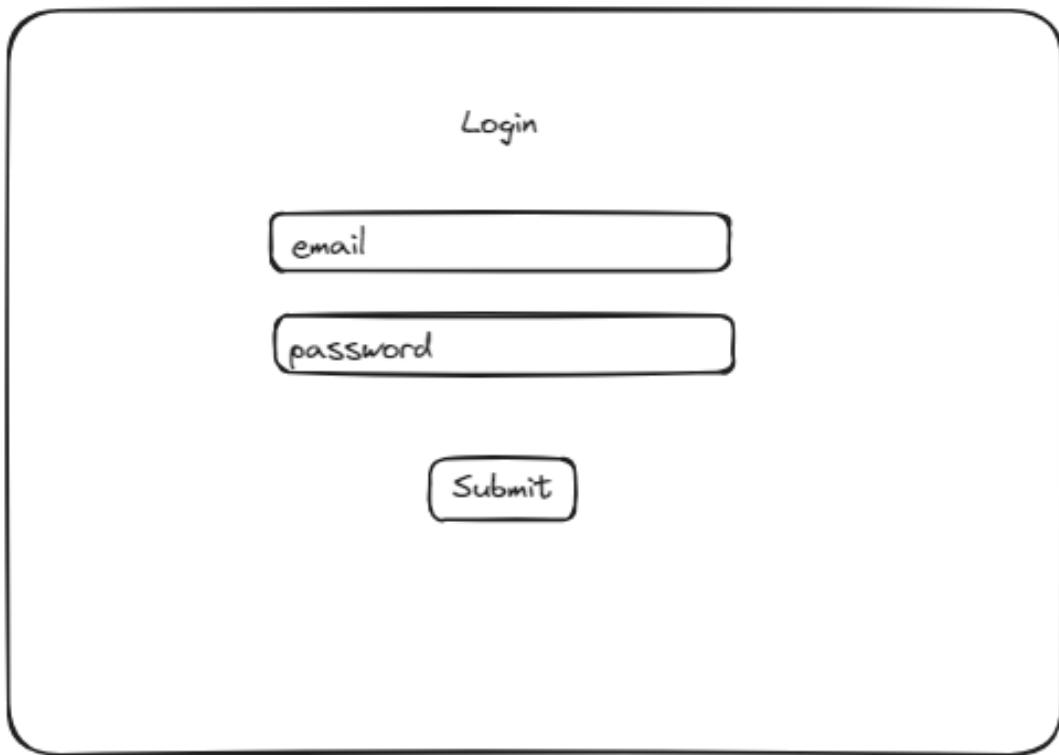


Figura 5.4: Realizar a autenticação do utilizador

Na Figura 5.5 é apresentado a interface relacionada com a página dedicada a apresentar recomendações não personalizadas a utilizadores não registados. Nesta pagina são apresentados cinco resultados para cada um são apresentados as seguintes informações:

- Imagem de capa do filme
- Titulo do filme

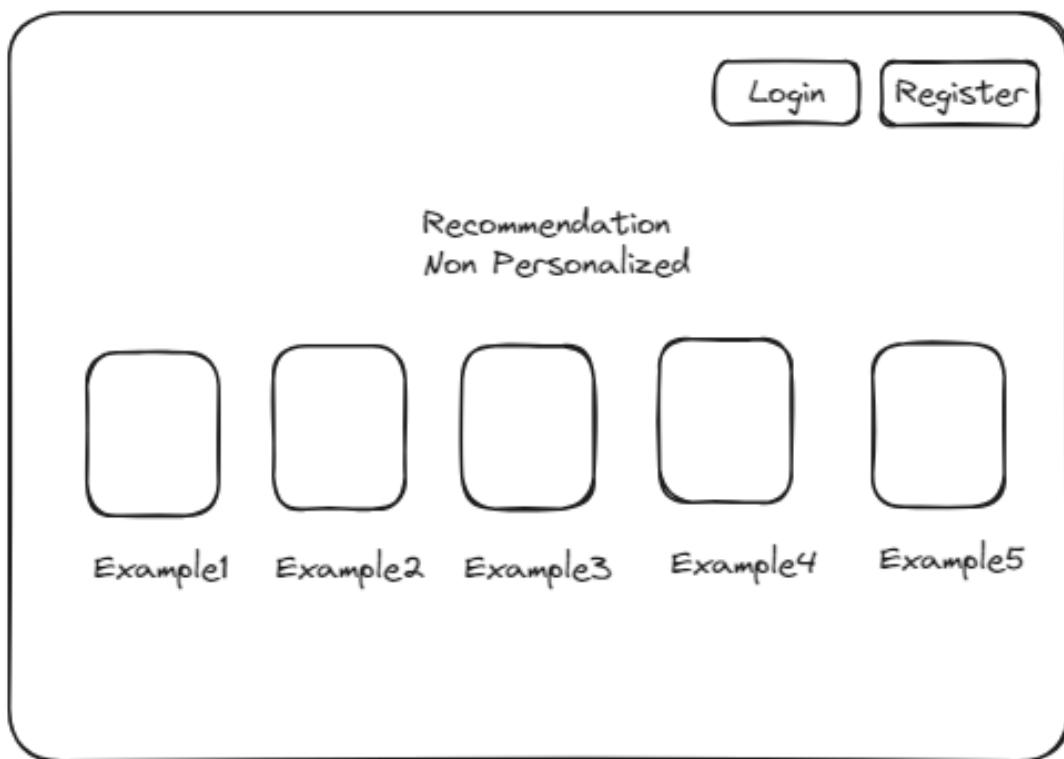


Figura 5.5: Recomendação não personalizada

Na Figura 5.6 é apresentado a interface relacionada com a página dedicada a apresentar recomendações personalizadas a utilizadores registados. Nesta pagina são apresentados cinco resultados para cada um são apresentados as seguintes informações:

- Imagem de capa do filme
- Título do filme

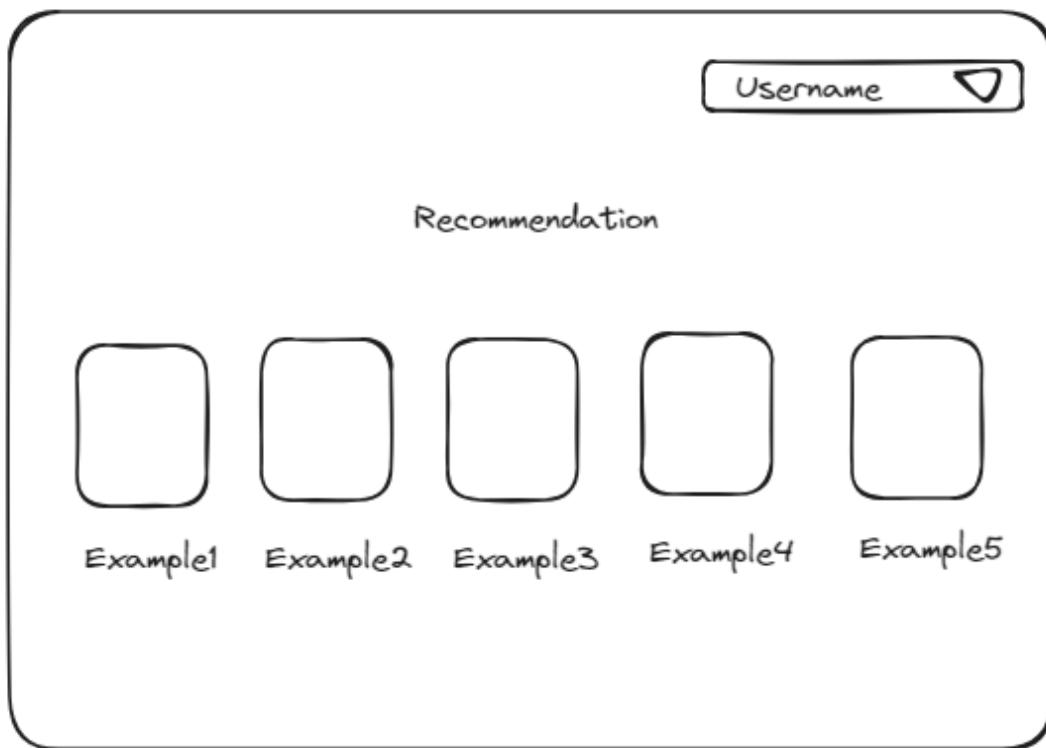


Figura 5.6: Recomendação personalizada

Na Figura 5.7 é apresentado a interface relacionada com a página dedicada a fornecer uma interface onde o utilizador possa realizar a sua avaliação num filme específico. Nesta pagina são apresentadas as seguintes informações:

- Imagem de capa do filme
- Título do filme
- Géneros do filme
- Campo gráfico onde o utilizador escolhe a sua avaliação de 0 a 5 estrelas com intervalos de 0.5 estrelas



Figura 5.7: Página do filme

Na Figura 5.8 é apresentado a interface relacionada com a página dedicada a fornecer uma interface onde é possível para a cada técnica selecionada ou um conjunto de técnicas obter os resultados da recomendação. Esta página deve ser utilizada num contexto de teste da técnica.

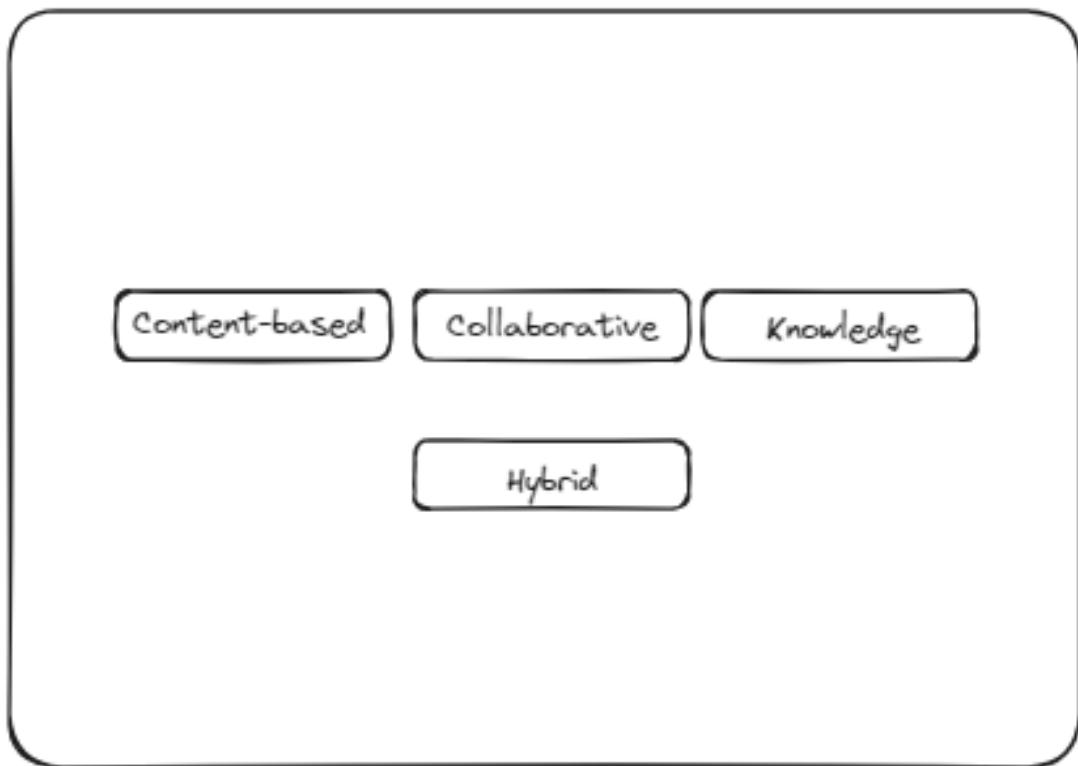


Figura 5.8: Página de testes das técnicas

Na Figura 5.9 é apresentado a interface relacionada com a página dedicada a fornecer uma interface para a técnica “Content Based”. Esta página deve ser utilizada num contexto de teste da técnica.

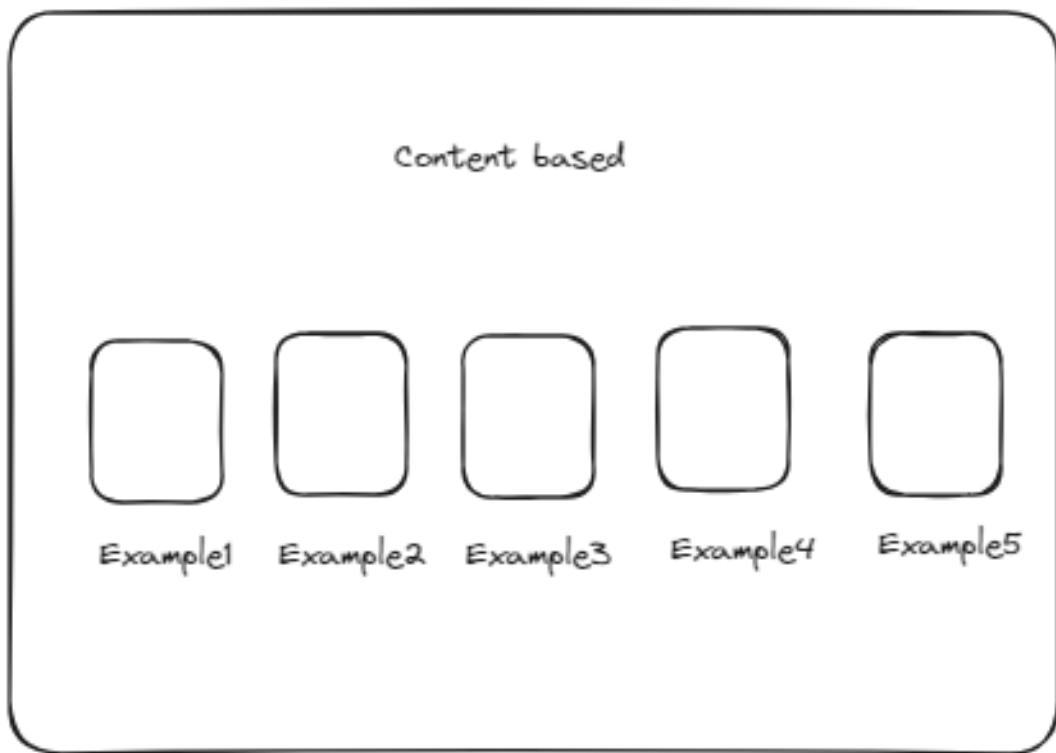


Figura 5.9: Página de testes “Content Based”

Na Figura 5.10 é apresentado a interface relacionada com a página dedicada a fornecer uma interface para a técnica “Collaborative”. Esta página deve ser utilizada num contexto de teste da técnica.

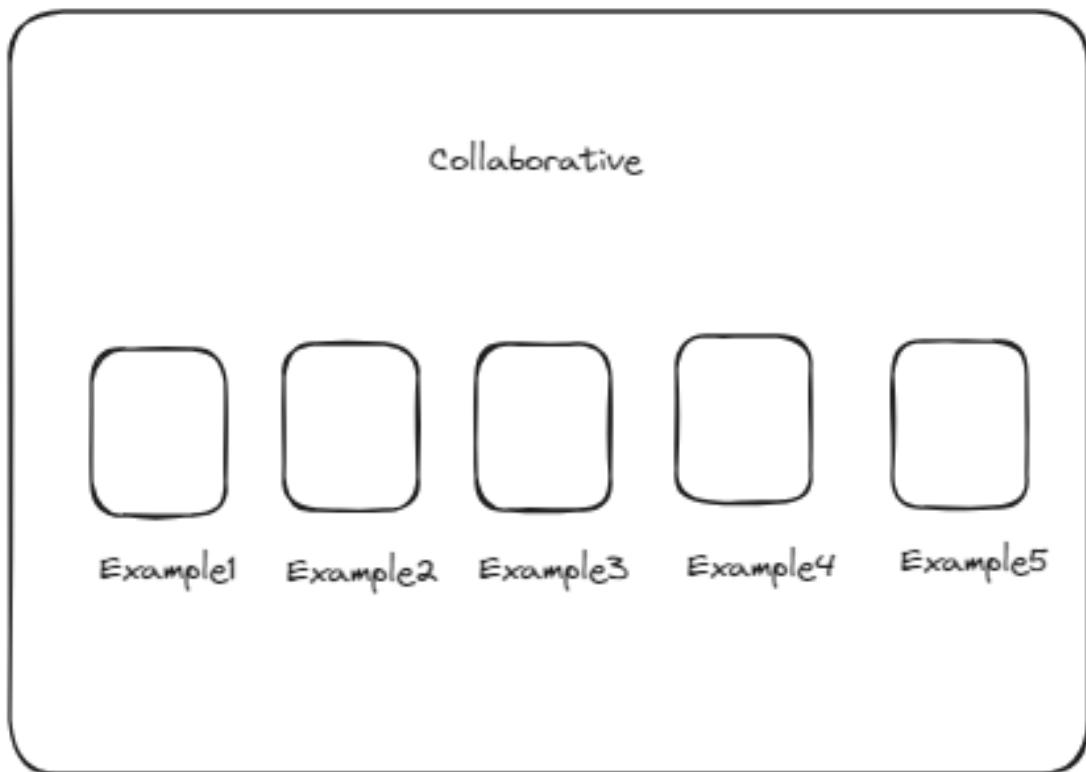


Figura 5.10: Página de testes “Collaborative”

Na Figura 5.11 é apresentado a interface relacionada com a página dedicada a fornecer uma interface para a técnica “Knowledge”. Esta página deve ser utilizada num contexto de teste da técnica.

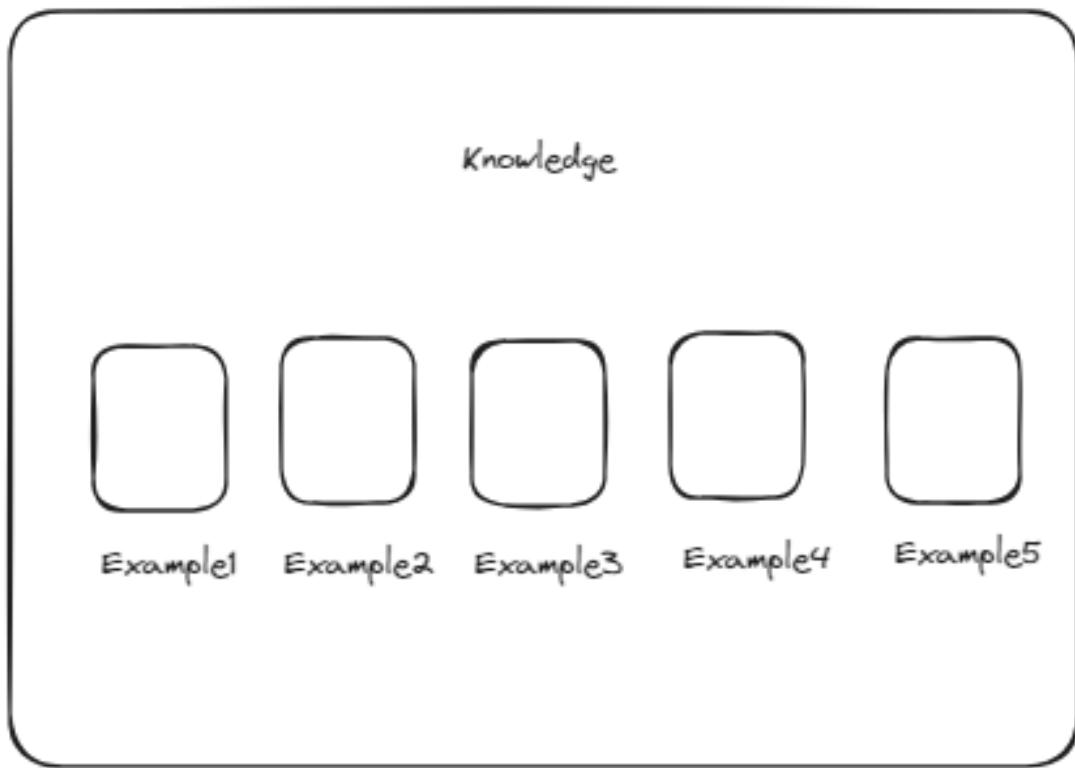


Figura 5.11: Página de testes “Knowledge”

Na Figura 5.12 é apresentado a interface relacionada com a página dedicada a fornecer uma interface para serem escolhidas as técnicas a serem utilizadas. Esta página deve ser utilizada num contexto de teste da técnica.

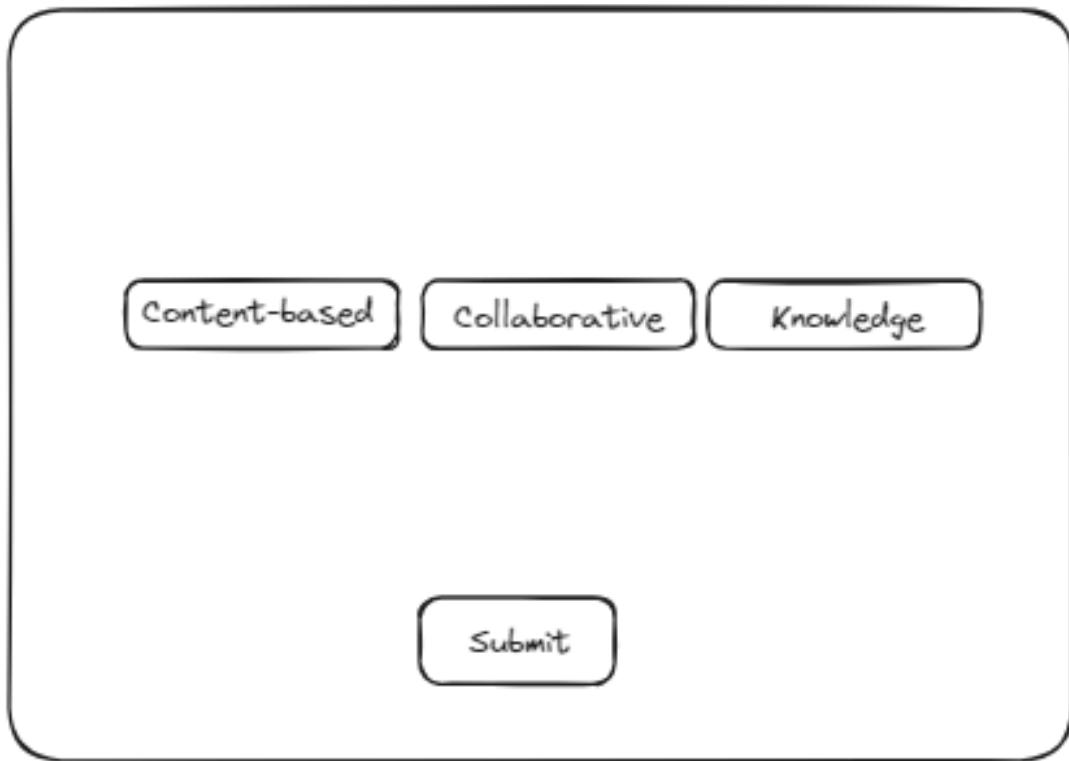


Figura 5.12: Escolha das técnicas que se pretende utilizar

Na Figura 5.13 é apresentado a interface relacionada com a página dedicada a fornecer uma interface para a técnica “Hybrid”. Esta página deve ser utilizada num contexto de teste da técnica.

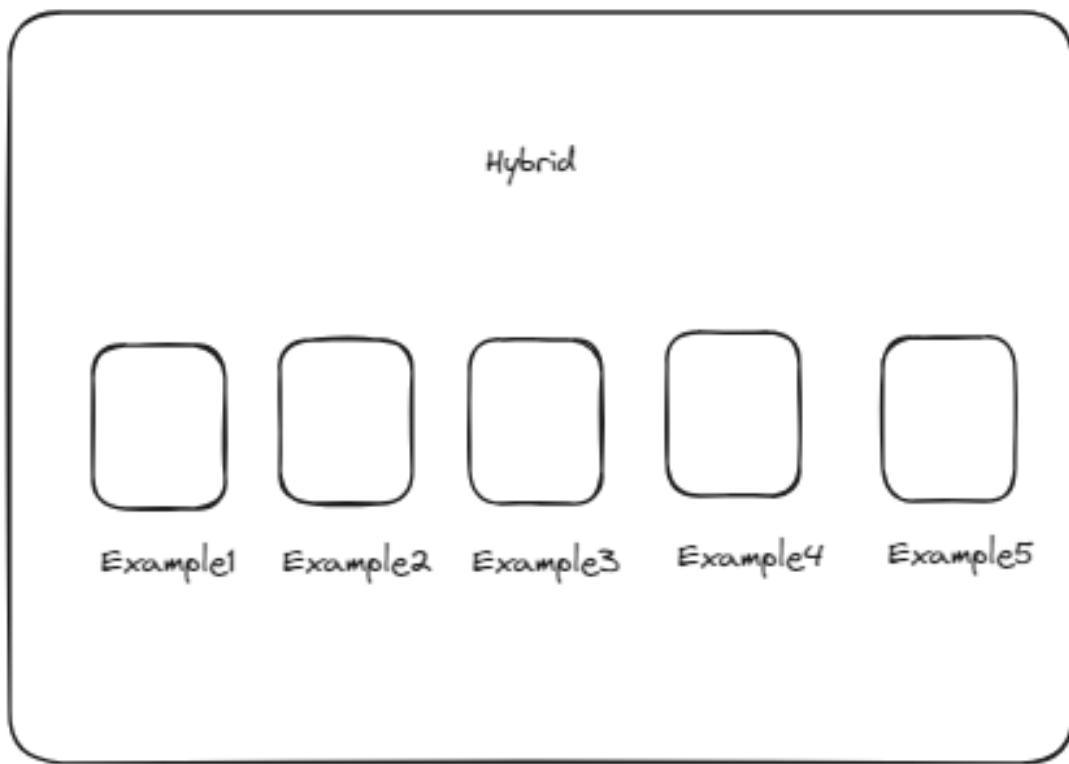


Figura 5.13: Página de testes “Hybrid”

Na Figura 5.14 é apresentado a interface relacionada com a página dedicada a fornecer uma interface para ser realizado a avaliação do sistema de recomendação implementado. Esta página deve ser utilizada num contexto de teste da técnica. Por sua vez, nesta pagina são apresentadas as seguintes informações:

- Afirmações com dados
- Gráficos
- Outras visualizações

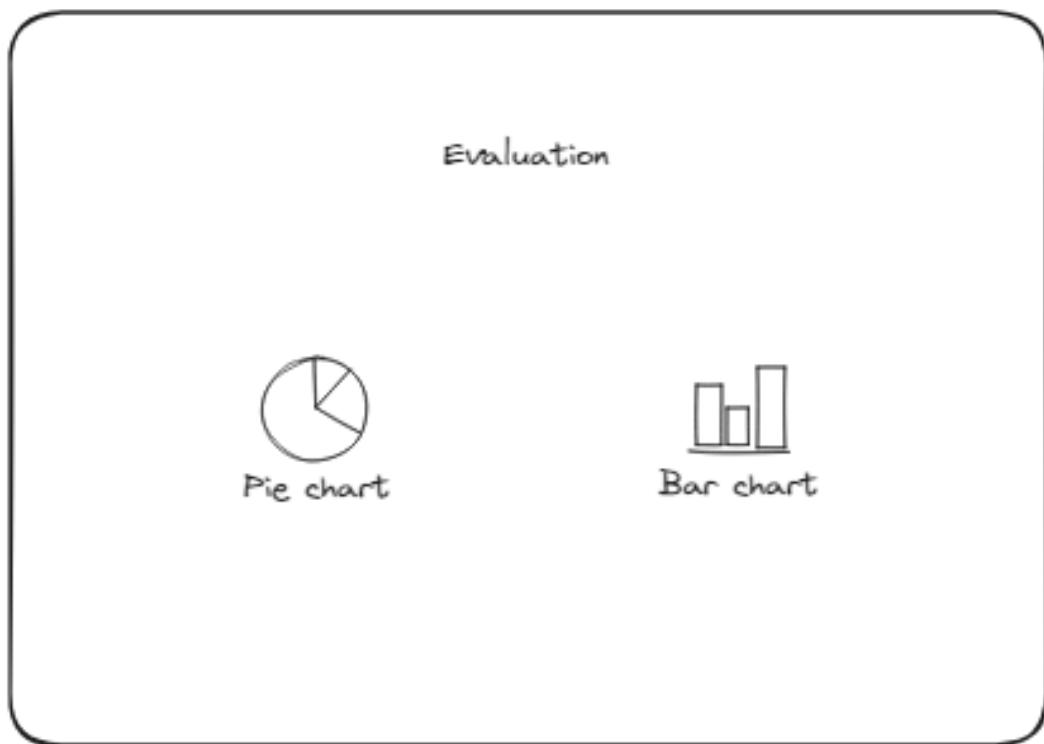


Figura 5.14: Avaliação do sistema de recomendação

### 5.3 Modelo de domínio

Um modelo de domínio é usado para representar conceitos, objetos e relações com outros objetos que sejam importantes num determinado domínio de negócios. O modelo correspondente ao presente sistema pode ser visto na Figura 5.15.

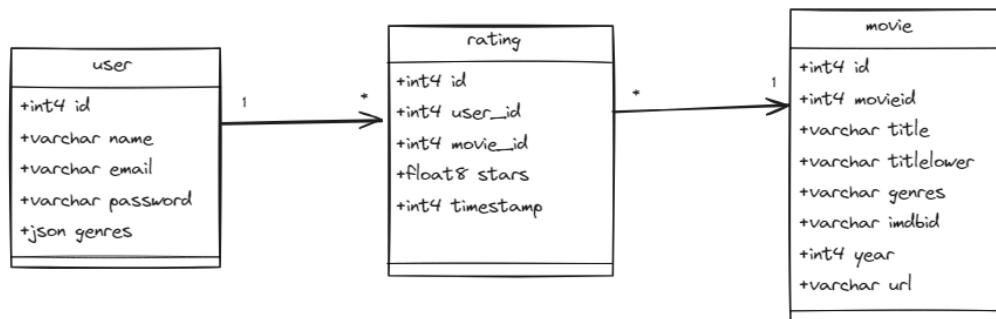


Figura 5.15: Modelo de domínio

A Figura 5.15 apresenta os vários conceitos:

- User: Representa um utilizador
- Rating: Representa uma avaliação feita por um utilizador num certo filme
- Movie: Representa um filme

A entidade User é constituída pelos seguintes parâmetros:

- Id
- Name
- Email
- Password
- Genres

A entidade Rating é constituída pelos seguintes parâmetros:

- Id
- UserId
- MovieId
- Stars
- Timestamp

A entidade Movie é constituída pelos seguintes parâmetros:

- Id
- MovieId
- Title
- TitleLower
- Genres
- ImdbId
- Year
- Url

## 5.4 Diagrama de casos de uso

Os casos de uso são uma ferramenta valiosa na engenharia de *software* para documentar e compreender os requisitos funcionais de um sistema. Por outro lado, os diagramas de caso de uso são as representações gráficas dos casos de uso, estes detalham as interações do sistema com todos os atores do mesmo.

- US01 - Criar o utilizador
- US02 - Atualizar os dados do utilizador
- US03 - Adicionar os géneros favoritos do utilizador
- US04 - Realizar o login do utilizador
- US05 - Procurar o utilizador através do seu id
- US06 - Realizar uma recomendação não personalizada
- US07 - Realizar uma recomendação personalizada
- US08 - Procurar um certo filme através do seu título
- US09 - Obter um filme de uma forma randomizada
- US010 - Criar uma avaliação
- US011 - Procurar uma avaliação que um utilizador fez num certo filme
- US012 - Obter as últimas 5 avaliações que um utilizador fez

- US013 - Obter os 5 filmes preferidos do utilizador
- US014 - Realizar recomendação não personalizada recorrendo a um género
- US015 - Realizar recomendação não personalizada recorrendo a um ano
- US016 - Realizar recomendação não personalizada recorrendo a uma década
- US017 - Realizar recomendação não personalizada sem critérios
- US018 - Realizar recomendação personalizada recorrendo à filtragem colaborativa
- US019 - Realizar recomendação personalizada recorrendo à filtragem baseada em conteúdo
- US020 - Realizar recomendação personalizada recorrendo à filtragem baseada em conhecimento
- US021 - Realizar recomendação personalizada recorrendo a uma abordagem híbrida
- US022 - Obter filmes similares

Os diagramas de casos de uso desenvolvido para este sistema podem ser observados nas Figuras 5.16 e 5.17.

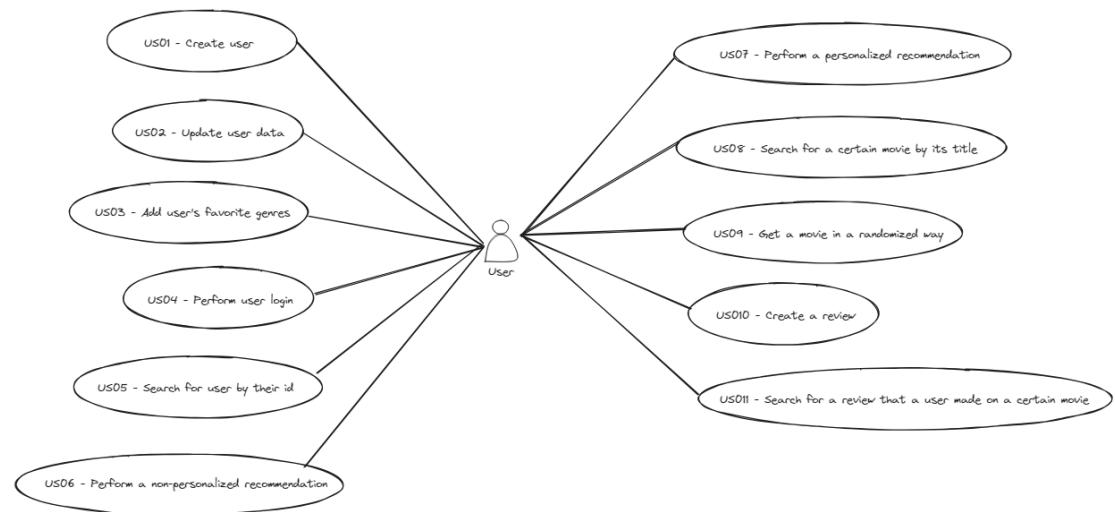


Figura 5.16: Diagrama de casos de uso 1

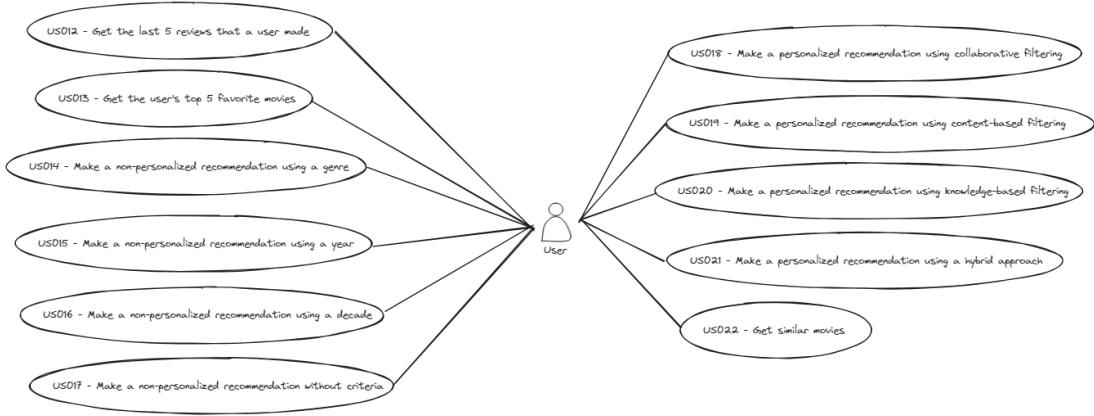


Figura 5.17: Diagrama de casos de uso 2

## 5.5 Solução para os problemas dos sistemas de recomendação

Na construção de sistemas de recomendação surgem alguns problemas que têm que ser resolvidos, tais como:

- Cold Start
- Mecanismos de autenticação
- Atualizar o perfil do utilizador com os ratings deste

### 5.5.1 Cold Start

O Cold Start é um problema que se prende com a ausência de informação quando um utilizador faz o registo da sua conta, já que não tem avaliações até então. Assim, as recomendações não serão “personalizadas” tendo em conta os gostos deste, já que estes ainda não foram descobertos. Para mitigar este problema, quando o utilizador faz o seu registo depois de a sua informação ser verificadas é pedido que este escolha dos vários géneros de filmes os seus preferidos, fazendo assim com que este manifeste as suas preferências. Esta implementação está demonstrada nas Figuras 5.18 e 5.19.

Movie App

Login  
Sign Up

Choose your favourite movie genres

<input type="checkbox"/> Action	<input type="checkbox"/> Adventure	<input type="checkbox"/> Animation	<input type="checkbox"/> Children	<input type="checkbox"/> Comedy
<input type="checkbox"/> Fantasy	<input type="checkbox"/> Romance	<input type="checkbox"/> Drama	<input type="checkbox"/> Action	<input type="checkbox"/> Crime
<input type="checkbox"/> Thriller	<input type="checkbox"/> Horror	<input type="checkbox"/> Mystery	<input type="checkbox"/> Sci-Fi	<input type="checkbox"/> IMAX
<input type="checkbox"/> Documentary	<input type="checkbox"/> War	<input type="checkbox"/> Musical	<input type="checkbox"/> Western	<input type="checkbox"/> Film-Noir

At least 3 genres should be selected

Figura 5.18: Cold Start - 1

Movie App

Login  
Sign Up

Choose your favourite movie genres

<input type="checkbox"/> Action	<input type="checkbox"/> Adventure	<input type="checkbox"/> Animation	<input type="checkbox"/> Children	<input checked="" type="checkbox"/> Comedy
<input type="checkbox"/> Fantasy	<input type="checkbox"/> Romance	<input type="checkbox"/> Drama	<input type="checkbox"/> Action	<input type="checkbox"/> Crime
<input checked="" type="checkbox"/> Thriller	<input checked="" type="checkbox"/> Horror	<input type="checkbox"/> Mystery	<input type="checkbox"/> Sci-Fi	<input type="checkbox"/> IMAX
<input type="checkbox"/> Documentary	<input type="checkbox"/> War	<input type="checkbox"/> Musical	<input type="checkbox"/> Western	<input type="checkbox"/> Film-Noir

Submit

Figura 5.19: Cold Start - 2

Depois da escolha dos géneros preferidos do utilizador, o sistema já tem informação afeta ao utilizador com que possa realizar recomendações baseadas exclusivamente nestas opções e, que, após realizar avaliações no sistema, serão aprimoradas e atualizadas.

### 5.5.2 Mecanismo de autenticação

De forma a poder trabalhar com os utilizadores é necessário criar uma conta referente a cada um destes, onde cada utilizador irá realizar as suas avaliações. Assim, é necessário, no registo, que o utilizador introduza o seu e-mail (que não pode ser um email já existente no sistema), a sua password e o seu username. Assim, no momento da autenticação, o utilizador apenas poderá entrar na sua conta aquando de uma combinação de email e password associada, correta. Na Figura 5.20 está representado o ecrã, quando a combinação destes dois fatores é incorreta.

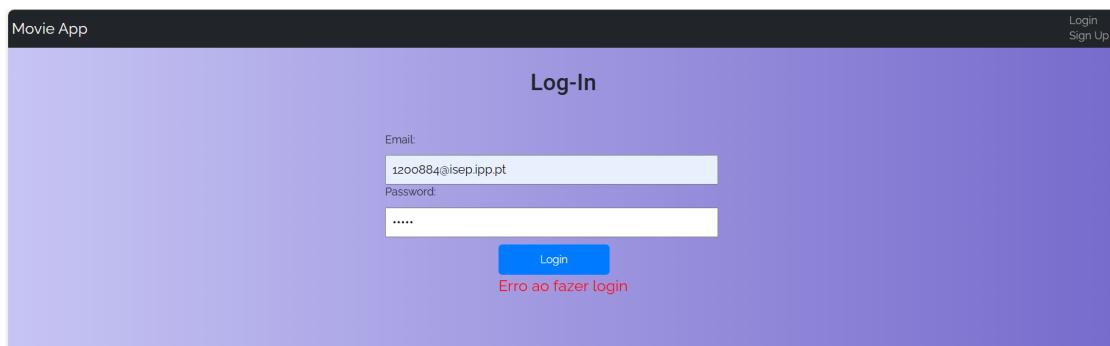


Figura 5.20: Autenticação Falhada

Já uma autenticação bem-sucedida, levará o utilizador às recomendações personalizadas.

# Capítulo 6

## Frontend

Neste Capítulo é apresentado o frontend, onde são mencionadas as páginas criadas a partir dos mockups construídos.

### 6.1 Páginas criadas

#### 6.1.1 Página inicial

Na página inicial da aplicação estão contidas as recomendações não personalizadas que são iguais para todos os utilizadores, não se baseando no perfil do utilizador (até porque, neste ponto, o utilizador ainda não se autenticou), baseando-se nos ratings de todos os utilizadores e da quantidade de ratings que o filme teve. Ponderando estes dois fatores através da fórmula do IMDB que foi utilizada anteriormente, obtém-se 5 filmes que são apresentados nesta página, como demonstra a Figura 6.1.

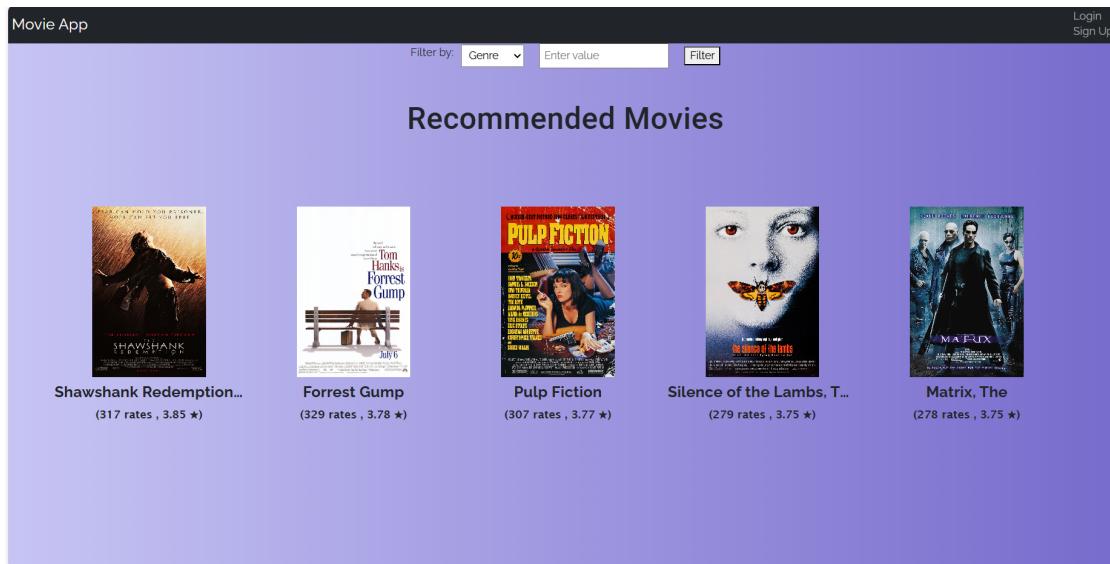


Figura 6.1: Página Inicial

Por outro lado, é permitido também ao utilizador filtrar estas recomendações não personalizadas, por ano, década, ou género. Este filtro, permite que o utilizador consiga encontrar as recomendações de filmes que prefere, apesar de não serem personalizadas ao gosto dele, utilizando para este fim a mesma fórmula ponderada do IMDB. Um exemplo de um filtro que pode ser o ano é o ano 2010, em que os filmes recomendados deste ano, assim como a utilização do filtro estão representados na Figura 6.2.

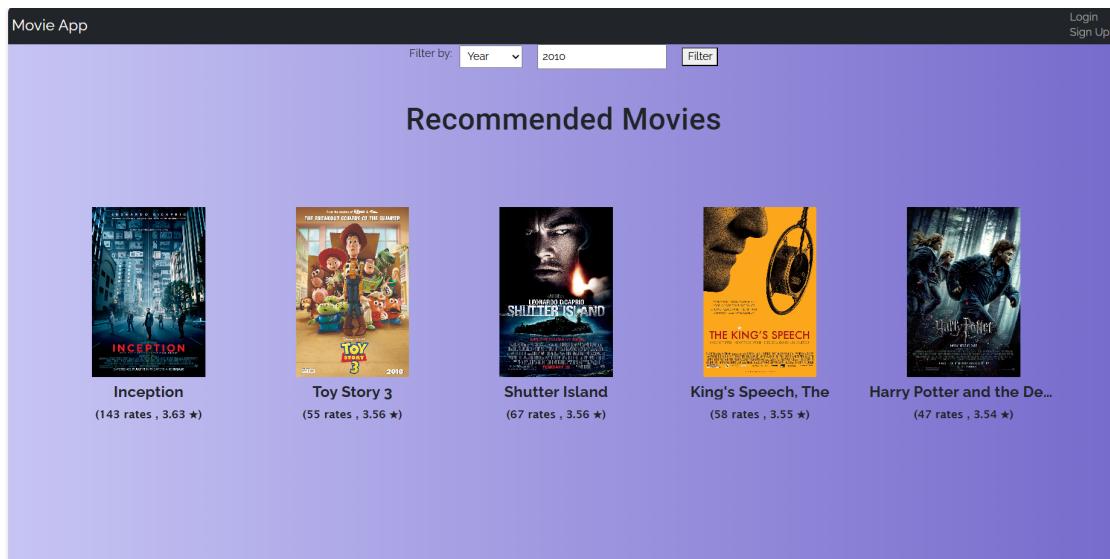


Figura 6.2: Recomendações não personaliadas de filmes de 2010

Note-se que como o utilizador ainda não se encontra autenticado, os botões no canto permitem que ele execute esta ação se já tiver uma conta no sistema ou registar-se, se não é o caso.

### 6.1.2 Autenticação

Se o utilizador, já tiver uma conta no sistema, para aceder a esta, precisa de se autenticar utilizando o botão referido anteriormente que o redireciona para esta página, demonstrada na Figura 6.3.

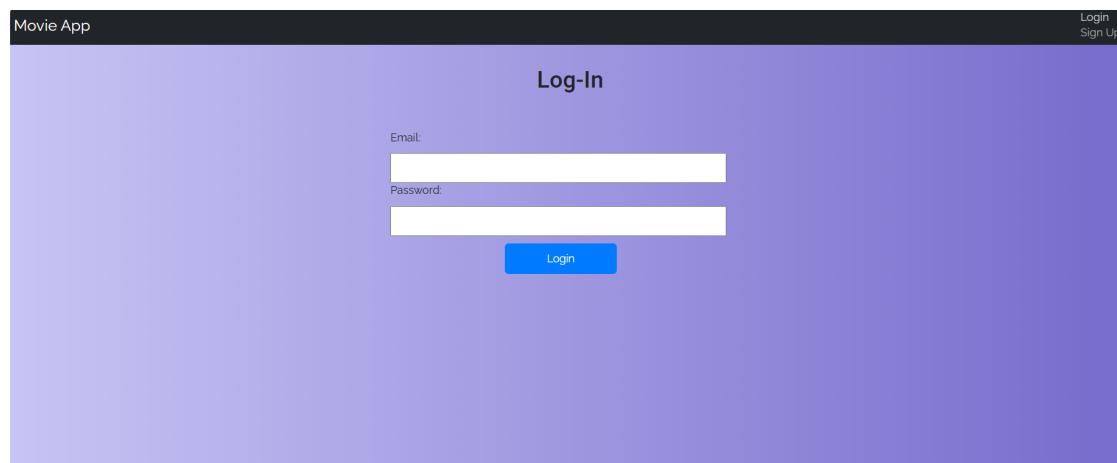


Figura 6.3: Autenticação

O utilizador necessita então de introduzir o e-mail e a palavra-passe que usou para se registar no sistema, e, se a combinação for a correta, este é autenticado com sucesso e redirecionado para as recomendações personalizadas.

### 6.1.3 Registo

Se o utilizador não tiver um perfil no sistema, carregando no botão de registo presente na 6.1, este pode fazê-lo, obtendo a página, presente na Figura 6.4.

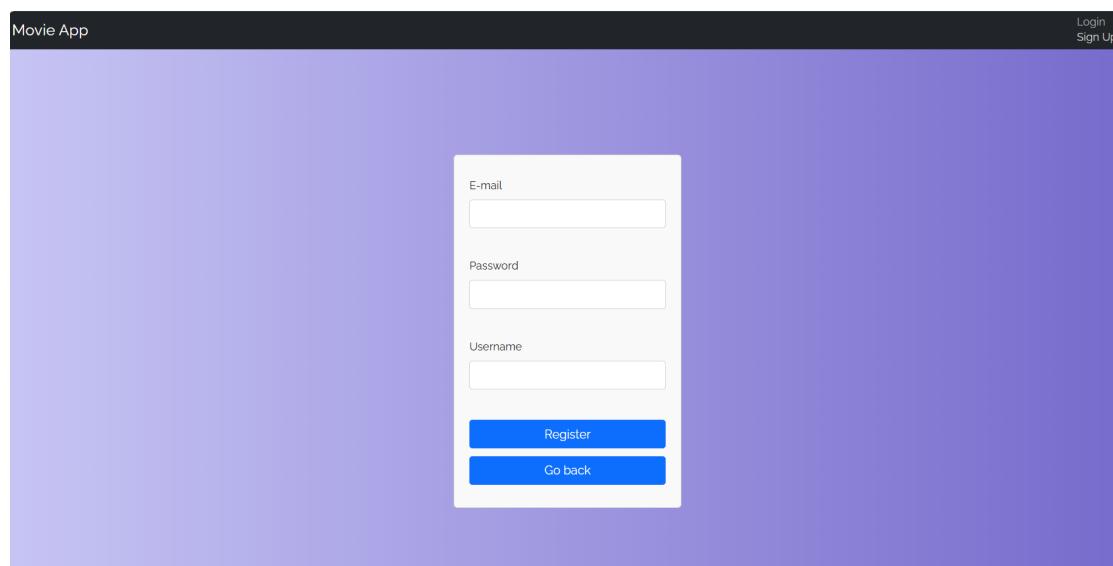


Figura 6.4: Registo no sistema

Após introduzir o seu nome, e-mail e palavra passe, que irão ser usados para identificar a atividade deste utilizador, assim como para que este possa entrar no seu perfil, aparecer-lhe-á os géneros de filmes que este tem preferência, resolvendo assim o problema do cold-start, já que atribuindo preferências, o sistema já pode recomendar alguns filmes que se enquadrem nestas. Esta página está presente na Figura .

#### 6.1.4 Últimas Avaliações

Para que o utilizador tenha dados estatísticos referentes à sua atividade no sistema, foi também implementada uma página que permita a este visualizar quais foram as 5 últimas avaliações de filmes realizadas, demonstrando de que filme se trata e qual foi a avaliação atribuída. Esta página está presente na Figura 6.5.

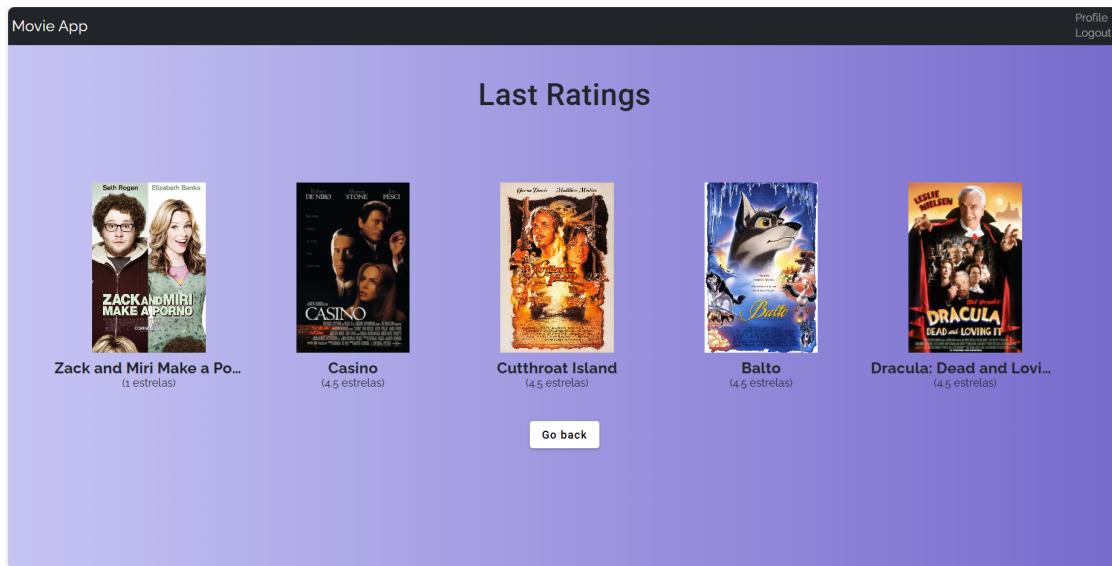


Figura 6.5: Últimas avaliações

### 6.1.5 Filmes melhor avaliados

Outro dado estatístico que pode interessar ao utilizador, referindo-se à sua atividade no sistema, será a demonstração dos filmes melhor avaliados, permitindo que este saiba quais foram os filmes que gostou mais, se este quiser revê-los, por exemplo. Esta página está presente na Figura 6.6.

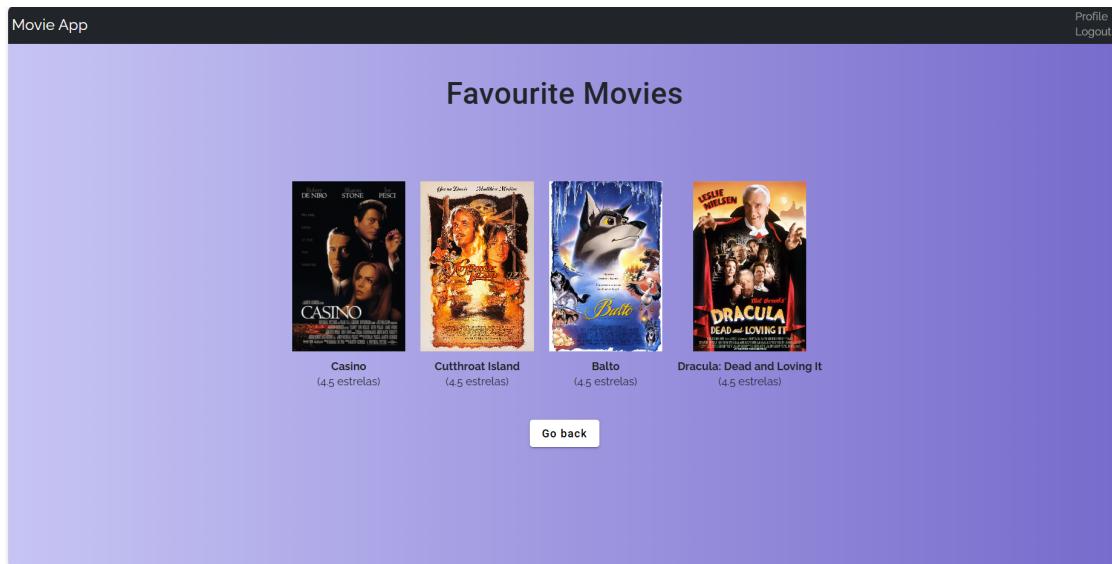


Figura 6.6: Filmes Preferidos

### 6.1.6 Avaliação de filmes

Na página de avaliar filmes, o utilizador pode procurar qual é o filme que pretende avaliar através da inserção do seu nome e, carregando no filme pretendido, aparece o filme com os detalhes, como a capa e título, seguido de algumas estrelas que servem para este realizar a sua avaliação. Este processo pode ser repetido as vezes que o utilizador quiser, para filmes diferentes, permitindo apenas uma avaliação por filme. A página está representada na Figura 6.7.

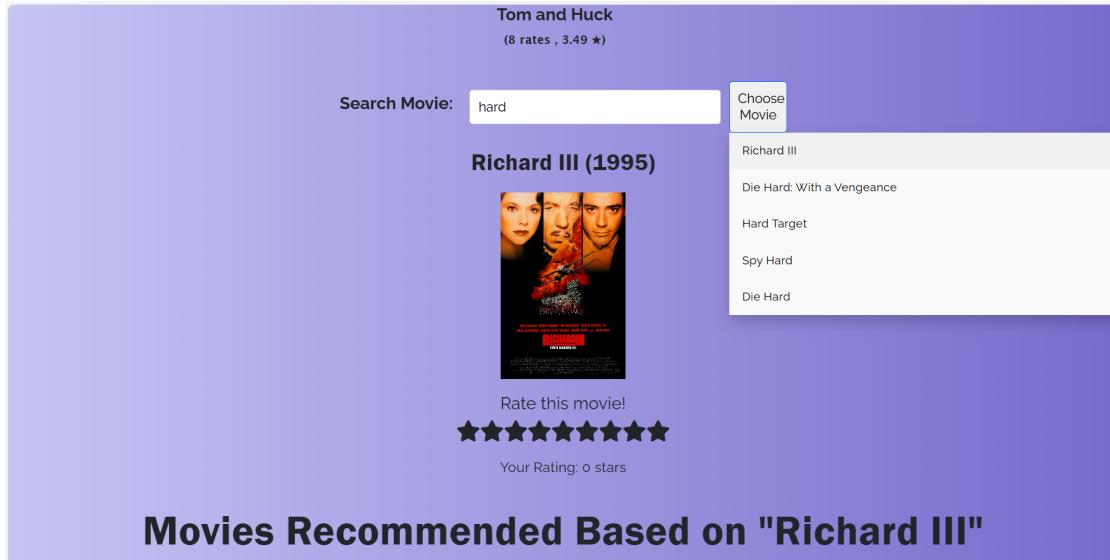


Figura 6.7: Avaliação de filmes

### 6.1.7 Recomendações personalizadas

Após o utilizador ter uma conta assim como alguma atividade associada no que toca a avaliar filmes, o sistema consegue recomendar-lhe os filmes que mais se adequam à sua atividade. Desta forma, estas recomendações apresentam-se na Figura 6.8.



Figura 6.8: Recomendações personalizadas

Existe também o detalhe de identificar os géneros para que o utilizador também esteja enquadrado no conteúdo do filme. Convém ressalvar que não são recomendados filmes com géneros que este referiu previamente que não gostava.

### 6.1.8 Recomendações de filmes parecidos ao selecionado

Quando o utilizador carrega num filme ou o avalia, existe uma funcionalidade que permite ao utilizador receber recomendações de filmes parecidos ao que avaliou ou selecionou. Assim, esta abordagem "content-based", está representada na Figura 6.9.

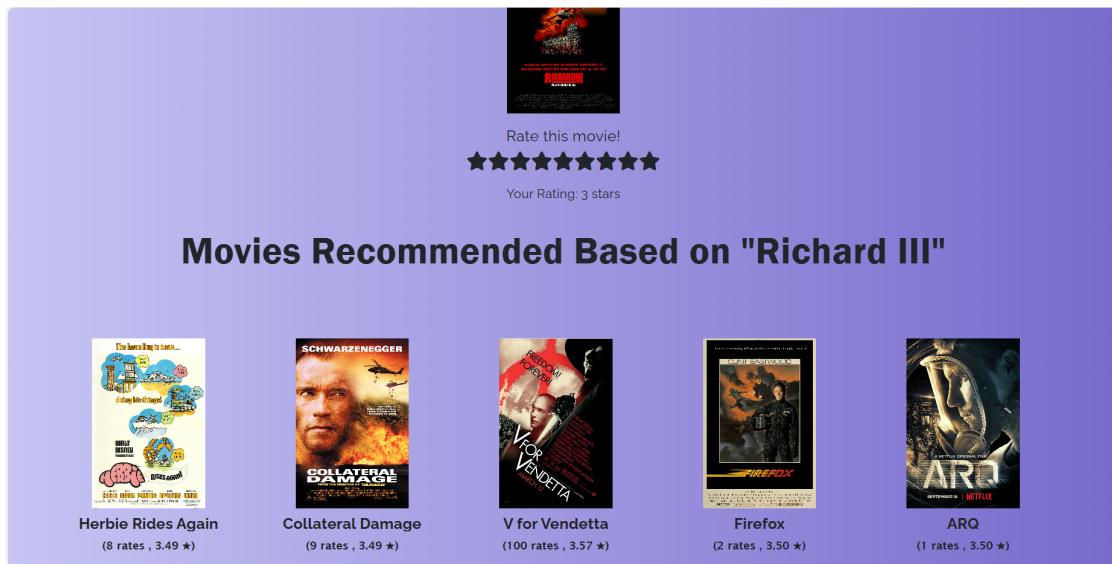


Figura 6.9: Recomendações Content-Based

### 6.1.9 Recomendações de filmes parecidos ao perfil do utilizador

Já na secção do perfil do utilizador, este pode utilizar uma funcionalidade que lhe permite a recomendação de filmes com base em utilizador com um perfil semelhante ao seu. Esta abordagem colaborativa, está presente na Figura 6.10.

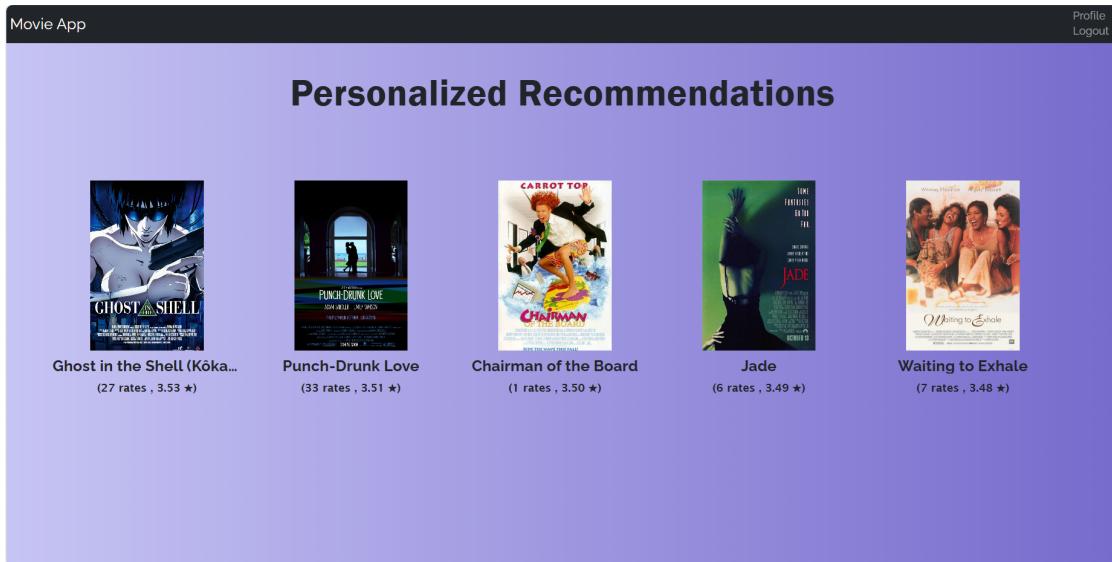


Figura 6.10: Abordagem Colaborativa

# Capítulo 7

## Backend

Neste Capítulo é apresentado a construção da parte backend da aplicação, onde são apresentados os endpoints definidos, as bibliotecas usadas e outras informações relevantes.

### 7.1 Endpoints definidos

Para suprir as necessidades do projeto foram criados routers em que cada um disponibiliza vários endpoints. Foram criados os seguintes routers:

- MovieRouter
- RatingRouter
- RecommendationRouter
- UserRouter

Na Tabela 7.1 são apresentados todos os *endpoints* ao router MovieRouter.

Tabela 7.1: *Endpoints* do router MovieRouter

<i>Endpoints</i>	
<b>Tipo de pedido</b>	<b>Endpoint</b>
GET	/movies/search
GET	/movies/randomMovie
GET	/movies/similarMovies

- /movies/search: Este endpoint é acedido através do método GET e é utilizado para realizar uma pesquisa de filmes através da introdução do seu título. Neste endpoint necessita a introdução de um parâmetro nomeadamente o title e não é necessário a introdução de um payload. Na Listagem 7.1 é apresentado um exemplo de uma resposta obtida.

---

```
1   [
2     {
3       "id": 29,
4       "movied": 29,
5       "titlelower": "city of lost children, the (cité des enfants perdus, la)",
6       "imdbid": "tt0112682",
7       "url": "https://m.media-amazon.com/images/M/
8         MV5BZGQxZDMwYzYtYmFjNi00NWYyLThjZjAtMDJhODZhYTkyZD
9         NhXkEyXkFqcGdeQXVyNTAyODkwOQ@@_.V1_.jpg",
10      "title": "City of Lost Children, The (Cité des enfants perdus, La)",
11      "genres": "Adventure|Drama|Fantasy|Mystery|Sci-Fi",
12      "year": 1995
13    },
14    {
15      "id": 37,
16      "movied": 37,
17      "titlelower": "across the sea of time",
18      "imdbid": "tt0112286",
19      "url": "https://m.media-amazon.com/images/M/
20        MV5BOTIwMzk1MDc1MF5BMl5BanBnXkFtZTewMTEzNDkyMQ@@.
21        _V1_.jpg",
22      "title": "Across the Sea of Time",
23      "genres": "Documentary|IMAX",
24      "year": 1995
25    }
]
```

---

Listagem 7.1: Resposta exemplo do endpoint GET /movies/search.

- /movies/randomMovie: Este endpoint também é acedido através do método GET e é utilizado para obter um filme aleatório. Neste endpoint não é necessário a introdução de parâmetros nem de payload. Na Listagem 7.2 é apresentado um exemplo de uma resposta obtida.

---

```

1   {
2     "id": 27347,
3     "movieid": 128912,
4     "titlelower": "mike birbiglia: my girlfriend's boyfriend",
5     "imdbid": "tt2937390",
6     "url": "https://m.media-amazon.com/images/M/
    MV5BYTk2NzdlNTktNmRlNy00NGUxLTg4N2YtOWVmOGYzZD
7   FjYjMwXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_.jpg",
8     "title": "Mike Birbiglia: My Girlfriend's Boyfriend",
9     "genres": "Comedy",
10    "year": 2013
11  }

```

---

Listagem 7.2: Resposta exemplo do endpoint GET /movies/randomMovie.

- /movies/similarMovies: Este endpoint é acedido através do método GET e é utilizado para se obter os cinco filmes mais semelhantes ao filme introduzido. Neste endpoint necessita a introdução de um parâmetro, nomeadamente o title e não é necessário a introdução de um payload. Na Listagem 7.3 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     "Toy Story",
3     "Toy Story 2",
4     "Bug's Life, A",
5     "Toy Story 4",
6     "Finding Dory"
7   ]

```

---

Listagem 7.3: Resposta exemplo do endpoint GET /movies/similarMovies.

Na Tabela 7.2 são apresentados todos os *endpoints* ao router RatingRouter.

Tabela 7.2: *Endpoints* do router RatingRouter

<i>Endpoints</i>	
<b>Tipo de pedido</b>	<b>Endpoint</b>
POST	/ratings/
GET	/ratings
GET	/ratings/history
GET	/ratings/favoriteMovies

- POST /ratings/: Este endpoint é acedido através do método POST e é utilizado

para submeter uma avaliação de um filme. Neste endpoint não é necessário a introdução de parâmetros, mas é necessário a introdução de um payload, com a estrutura apresentada na Listagem 7.4. Na Listagem 7.5 é apresentado um exemplo de uma resposta obtida.

---

```
1  {
2    "user_id": 2,
3    "movie_id": 5,
4    "stars": 5
5 }
```

---

Listagem 7.4: Payload exemplo do endpoint POST /ratings/.

---

```
1  {
2    "timestamp": 1715385105,
3    "id": 25000129,
4    "movie_id": 5,
5    "user_id": 2,
6    "stars": 5
7 }
```

---

Listagem 7.5: Resposta exemplo do endpoint POST /ratings/.

- GET /ratings: Este endpoint é acessado através do método GET e é utilizado para obter todas as avaliações de filmes através. Neste endpoint é necessário a introdução de parâmetros nomeadamente o userId e o movieId, mas não é necessário a introdução de um payload. Na Listagem 7.6 é apresentado um exemplo de uma resposta obtida.

```
1   {
2     "id": 25000097,
3     "movieId": 23,
4     "userId": 200000,
5     "stars": 5,
6     "timestamp": 1714836082,
7     "title": "Assassins",
8     "titleLower": "assassins",
9     "genres": "Action|Crime|Thriller",
10    "imdbId": "tt0112401",
11    "year": 1995,
12    "url": "https://m.media-amazon.com/images/M/
13      MV5BZGI1NDA4ZDItNTRjMi00YTU3LTkwZDEtYjdlNTI1ZjQxZD
14      M1XkEyXkFqcGdeQXVyNDc2NjEyMw@@._V1_.jpg"
}
```

Listagem 7.6: Resposta exemplo do endpoint GET /ratings/.

- GET /ratings/history: Este endpoint também é acedido através do método GET e é utilizado para obter o histórico de avaliações de filmes realizadas pelo o utilizador. Neste endpoint é necessário a introdução de parâmetros nomeadamente o userId, mas não é necessário a introdução de um payload. Na Listagem 7.7 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     {
3       "id": 25000116,
4       "movieId": 139,
5       "userId": 200000,
6       "stars": 3,
7       "timestamp": 1715102007,
8       "title": "Target",
9       "titleLower": "target",
10      "genres": "Action|Drama",
11      "imdbId": "tt0114618",
12      "year": 1995,
13      "url": "https://m.media-amazon.com/images/M/
14        MV5BMjM3NjkxYjgtMzQ2Ni00ZjMxLTlkYzUtYTQ2ZWM2YTgy
15        OWU5XkEyXkFqcGdeQXVyMjM5NDM1MTE@._V1_.jpg"
16    },
17    {
18      "id": 25000115,
19      "movieId": 341,
20      "userId": 200000,
21      "stars": 1,
22      "timestamp": 1714933287,
23      "title": "Double Happiness",
24      "titleLower": "double happiness",
25      "genres": "Drama",
26      "imdbId": "tt0109655",
27      "year": 1994,
28      "url": "https://m.media-amazon.com/images/M/
29        MV5BNDNhZDBkZmYtYTM0Yy00ZGViLWExOGItMWQxYjI1OG
30        M4ODZjXkEyXkFqcGdeQXVyMTQ3Njg3MQ@@._V1_.jpg"
31    }
32  ]

```

---

Listagem 7.7: Resposta exemplo do endpoint GET /ratings/history.

- GET /ratings/favoriteMovies: Este endpoint é acedido através do método GET e é utilizado para obter os filmes favoritos do utilizador. Neste endpoint é necessário a introdução de parâmetros nomeadamente o userId, mas não é necessário a introdução de um payload. Na Listagem 7.8 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2   {
3     "id": 25000105,
4     "movieId": 23,
5     "userId": 200000,
6     "stars": 5,
7     "timestamp": 1714836641,
8     "title": "Assassins",
9     "titleLower": "assassins",
10    "genres": "Action|Crime|Thriller",
11    "imdbId": "tt0112401",
12    "year": 1995,
13    "url": "https://m.media-amazon.com/images/M/
14      MV5BZGI1NDA4ZDItNTRjMi00YTU3LTkwZDEtYjdlNTI1ZjQx
15      ZDM1XkEyXkFqcGdeQXVyNDc2NjEyMw@@._V1_.jpg"
16  },
17  {
18    "id": 25000104,
19    "movieId": 23,
20    "userId": 200000,
21    "stars": 5,
22    "timestamp": 1714836631,
23    "title": "Assassins",
24    "titleLower": "assassins",
25    "genres": "Action|Crime|Thriller",
26    "imdbId": "tt0112401",
27    "year": 1995,
28    "url": "https://m.media-amazon.com/images/M/
29      MV5BZGI1NDA4ZDItNTRjMi00YTU3LTkwZDEtYjdlNTI1ZjQx
30      ZDM1XkEyXkFqcGdeQXVyNDc2NjEyMw@@._V1_.jpg"
31  }]

```

---

Listagem 7.8: Resposta exemplo do endpoint GET /ratings/favoriteMovies.

Na Tabela 7.3 são apresentados todos os *endpoints* ao router RecommendationRouter.

Tabela 7.3: *Endpoints* do router RecommendationRouter

<i>Endpoints</i>	
<b>Tipo de pedido</b>	<b>Endpoint</b>
GET	/recommendation/nonpersonalized
GET	/recommendation/nonpersonalizedGenre
GET	/recommendation/nonpersonalizedYear
GET	/recommendation/nonpersonalizedDecade
GET	/recommendation/nonpersonalizedOverall
GET	/recommendation/personalizedColaborative
GET	/recommendation/personalizedContent
GET	/recommendation/personalizedKnowledge
GET	/recommendation/personalizedHybrid

- GET /recommendation/nonpersonalized: Este endpoint é acedido através do método GET e é utilizado para obter recomendações não personalizadas, ou seja, recomendações que não são baseadas no histórico individual do utilizador. Neste endpoint não é necessário a introdução de parâmetros nem de payload. Na Listagem 7.9 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     {
3       "index": 314,
4       "movieid": 318,
5       "title": "Shawshank Redemption, The",
6       "genres": "Crime|Drama",
7       "imdbid": "tt0111161",
8       "year": 1994,
9       "url": "https://m.media-amazon.com/images/M/
10      MV5BMDFkYTc0MGEtZmNhMC00ZDlzLWFmNTEtODM1Zm
11      RIYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_.jpg",
12      "titlelower": "shawshank redemption, the",
13      "count": 81482,
14      "mean": 4.413576004516335,
15      "weighted_rating": 4.39730333795193
16    },
17    {
18      "index": 840,
19      "movieid": 858,
20      "title": "Godfather, The",
21      "genres": "Crime|Drama",
22      "imdbid": "tt0068646",
23      "year": 1972,
24      "url": "https://m.media-amazon.com/images/M/
25      MV5BM2MyNjYxNmUtYTAwNi00MTYxLWJmNWYtYzZlODY
26      3ZTk3OTFlXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_.jpg",
27      "titlelower": "godfather, the",
28      "count": 52498,
29      "mean": 4.324336165187245,
30      "weighted_rating": 4.300915434613464
31    }
32  ]

```

---

Listagem 7.9: Resposta exemplo do endpoint GET /recommendation/nonpersonalized.

- GET /recommendation/nonpersonalizedGenre: Este endpoint é acedido através do método GET e é utilizado para obter recomendações não personalizadas através do género introduzido. Neste endpoint não é necessário a introdução de parâmetros nem de payload. Na Listagem 7.10 é apresentado um exemplo de uma resposta obtida.

---

```
1   [
2     {
3       "movieId": 318,
4       "title": "Shawshank Redemption, The",
5       "url": "https://m.media-amazon.com/images/M/
6         MV5BMDFkYTc0MGEtZmNhMC00ZDlzMWFmNTEtODM1ZmRl
7         YWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_.jpg",
8       "count": 317,
9       "weighted_rating": 3.849204406364749
10      },
11      {
12        "movieId": 296,
13        "title": "Pulp Fiction",
14        "url": "https://m.media-amazon.com/images/M/
15          MV5BNGNhMDIzZTUtNTBlZi00MTRlWFljM2ItYzViMjE3YzI5MjljX
16          kEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_.jpg",
17        "count": 307,
18        "weighted_rating": 3.7662949194547695
19      }
20  ]
```

Listagem 7.10: Resposta exemplo do endpoint GET /recommendation/nonpersonalizedGenre.

- GET /recommendation/nonpersonalizedYear: Este endpoint é acedido através do método GET e é utilizado para obter recomendações não personalizadas através do ano introduzido. Neste endpoint não é necessário a introdução de parâmetros nem de payload. Na Listagem 7.11 é apresentado um exemplo de uma resposta obtida.

```
1   [
2     {
3       "movieId": 79132,
4       "title": "Inception",
5       "url": "https://m.media-amazon.com/images/M/
6         MV5BMjAxMzY3NjcxNF5BML5Ban
7         BnXkFtZTcwNTI5OTM0Mw@@._V1_.jpg",
8       "count": 143,
9       "weighted_rating": 3.633437013996889
10      },
11      {
12        "movieId": 78499,
13        "title": "Toy Story 3",
14        "url": "https://m.media-amazon.com/images/M/
15         MV5BMTgxOTY4Mjc0MF5BML5Ban
16         BnXkFtZTcwNTA4MDQyMw@@._V1_.jpg",
17       "count": 55,
18       "weighted_rating": 3.55945945945946
19     }
]
```

Listagem 7.11: Resposta exemplo do endpoint GET /recommendation/nonpersonalizedYear.

- GET /recommendation/nonpersonalizedDecade: Este endpoint é acedido através do método GET e é utilizado para obter recomendações não personalizadas através da decada introduzido. Neste endpoint não é necessário a introdução de parâmetros nem de payload. Na Listagem 7.12 é apresentado um exemplo de uma resposta obtida.

```
1   [
2     {
3       "movieId": 79132,
4       "title": "Inception",
5       "url": "https://m.media-amazon.com/images/M/
6         MV5BMjAxMzY3NjcxNF5BM
7         l5BanBnXkFtZTcwNTI5OTM0Mw@@._V1_.jpg",
8       "count": 143,
9       "weighted_rating": 3.633437013996889
10      },
11      {
12        "movieId": 91529,
13        "title": "Dark Knight Rises, The",
14        "url": "https://m.media-amazon.com/images/M/
15         MV5BMTk4ODQzMjY3MjI5BM
16         l5BanBnXkFtZTcwODA0NTM4Nw@@._V1_.jpg",
17        "count": 76,
18        "weighted_rating": 3.5659722222222223
19      }
20   ]
```

Listagem 7.12: Resposta exemplo do endpoint GET /recommendation/nonpersonalizedDecade.

- GET /recommendation/nonpersonalizedOverall: Este endpoint é acedido através do método GET e é utilizado para obter recomendações não personalizadas. Neste endpoint não é necessário a introdução de parâmetros nem de payload. Na Listagem 7.13 é apresentado um exemplo de uma resposta obtida.

---

```
1   [
2     {
3       "movieId": 318,
4       "title": "Shawshank Redemption, The",
5       "url": "https://m.media-amazon.com/images/M/
6         MV5BMDFkYTc0MGEtZmNhMC00ZDlzMWFmNTEtODM1Zm
7         RIYWMwMWFMXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_.jpg",
8       "count": 317,
9       "weighted_rating": 3.849204406364749
10      },
11      {
12        "movieId": 356,
13        "title": "Forrest Gump",
14        "url": "https://m.media-amazon.com/images/M/
15         MV5BNWIwODRIZTUtY2U3ZS00Yzg1LWJhNzYt
16         MmZiYmEyNmU1NjMzXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_.jpg",
17        "count": 329,
18        "weighted_rating": 3.777804583835947
19      }
20   ]
```

Listagem 7.13: Resposta exemplo do endpoint GET /recommendation/nonpersonalizedOverall.

- GET /recommendation/personalizedColaborative: Este endpoint é acedido através do método GET e é utilizado para obter recomendações personalizadas, no meadamento com recurso à filtragem colaborativa. Neste endpoint é necessário a introdução do user\_id mas não é preciso a introdução de payload. Na Listagem 7.14 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     {
3       "title": "Hoodlum",
4       "url": "https://m.media-amazon.com/images/M/
      MV5BMmExY2QyMDYtZjc1NC00YzRmLWJlYzUtODY1MzA2OTliN
6       mE1XkEyXkFqcGdeQXVvMTMxMTY0OTQ@._V1_.jpg",
5       "genres": "Crime|Drama|Film-Noir",
6       "imdbId": "tt0119311",
7       "year": 1997
8     },
9     {
10       "title": "Best Man, The (Testimone dello sposo, Il)",
11       "url": "https://m.media-amazon.com/images/M/
      MV5BYmY4MDBkNzQtYTk1Yi00MTMyLThiMDYtODA4NmM3MjN
12       jYWlyXkEyXkFqcGdeQXVvMTQ3Njg3MQ@@._V1_.jpg",
13       "genres": "Comedy|Drama|Romance",
14       "imdbId": "tt0133413",
15       "year": 1998
16     }

```

---

Listagem 7.14: Resposta exemplo do endpoint GET /recommendation/personalizedColaborative.

- GET /recommendation/personalizedContent: Este endpoint é acedido através do método GET e é utilizado para obter recomendações personalizadas, nomeadamente com recurso à filtragem baseada em conteúdo. Neste endpoint é necessário a introdução do user\_id e do título, mas não é preciso a introdução de payload. Na Listagem 7.15 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     {
3       "title": "Monsters, Inc.",
4       "year": 2001,
5       "url": "https://m.media-amazon.com/images/M/
6         MV5BMTY1NTI0ODUyOF5
7         BMI5BanBnXkFtZTgwNTEyNjQ0MDE@._V1_.jpg",
8       "count": 132,
9       "weighted_rating": 3.5835443037974684
10      },
11      {
12        "title": "Toy Story 2",
13        "year": 1999,
14        "url": "https://m.media-amazon.com/images/M/
15         MV5BMWM5ZDcxMTYtNTEyNS00MDRkLWI3YTItNThmMGE
16         xMWY4NDIwXkEyXkFqcGdeQXVyNjUwNzk3NDc@._V1_.jpg",
17       "count": 97,
18       "weighted_rating": 3.56499162479062
19     }
20   ]

```

---

Listagem 7.15: Resposta exemplo do endpoint GET /recommendation/personalizedContent.

- GET /recommendation/personalizedKnowledge: Este endpoint é acedido através do método GET e é utilizado para obter recomendações personalizadas, nomeadamente com recurso à filtragem baseada em conhecimento. Neste endpoint é necessário a introdução do user\_id mas não é preciso a introdução de payload. Na Listagem 7.16 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     {
3       "title": "Shawshank Redemption, The",
4       "year": 1994,
5       "url": "https://m.media-amazon.com/images/M/
6         MV5BMDFkYTc0MGEtZmNhMC00ZDlzMWFmNTE
7         tODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxODk2OTU@._V1_
8         .jpg",
9       "count": 317,
10      "weighted_rating": 3.8492044063647493
11    },
12    {
13      "title": "Forrest Gump",
14      "year": 1994,
15      "url": "https://m.media-amazon.com/images/M/
16        MV5BNWIwODRlZTUtY2U3ZS00Yzg1LWJhNzY
tMmZiYmEyNmU1NjMzXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_.jpg"
17      ,
18      "count": 329,
19      "weighted_rating": 3.777804583835947
20    }
21 ]

```

---

Listagem 7.16: Resposta exemplo do endpoint GET /recommendation/personalizedKnowledge.

- GET /recommendation/personalizedHybrid: Este endpoint é acedido através do método GET e é utilizado para obter recomendações personalizadas, nomeadamente com recurso à abordagem híbrida. Neste endpoint é necessário a introdução do user\_id, mas não é preciso a introdução de payload. Na Listagem 7.17 é apresentado um exemplo de uma resposta obtida.

---

```

1   [
2     {
3       "movieId": 737,
4       "title": "Barb Wire",
5       "genres": [
6         "Action",
7         "Sci-Fi"
8       ],
9       "year": 1996,
10      "url": "https://m.media-amazon.com/images/M/
11        MV5BYWQ3NTc1YzQtNDJhYS00NDIzMWFmM2EtZj
12        gxNzQwNDQ4YjNmXkEyXkFqcGdeQXVyNjQ2MjQ5NzM@._V1_.jpg",
13      "count": 20,
14      "weighted_rating": 3.4653846153846155,
15      "features": "Action Sci-Fi"
16    },
17    {
18      "movieId": 3450,
19      "title": "Grumpy Old Men",
20      "genres": [
21        "Comedy"
22      ],
23      "year": 1993,
24      "url": "https://m.media-amazon.com/images/M/
25        MV5BMzNiYzQyNGEtYjFiOS00OTcyLT
26        g5YzItMDQ2ZGRmZjE1N2Y4XkEyXkFqcGdeQXVyMTQxNzMzNDI@.
27        _V1_.jpg",
28      "count": 29,
29      "weighted_rating": 3.4890359168241964,
30      "features": "Comedy"
31    }
32  ]

```

---

Listagem 7.17: Resposta exemplo do endpoint GET /recommendation/personalizedHybrid.

Na Tabela 7.4 são apresentados todos os *endpoints* ao router UserRouter.

Tabela 7.4: *Endpoints* do router UserRouter

<i>Endpoints</i>	
<b>Tipo de pedido</b>	<b>Endpoint</b>
POST	/users
POST	/users/genres
POST	/users/login
GET	/users
PATCH	/users

- POST /users: Este endpoint é acedido através do método POST e é utilizado para criar um novo utilizador. Neste endpoint não é necessário a introdução de parâmetros, mas é necessário a introdução de um payload, com a estrutura apresentada na Listagem 7.18. Na Listagem 7.19 é apresentado um exemplo de uma resposta obtida.

---

```
1  {
2    "name": "EXAMPLE",
3    "email": "EXAMPLE",
4    "password": "EXAMPLE"
5 }
```

---

Listagem 7.18: Payload exemplo do endpoint POST /users.

---

```
1  {
2    "id": 200003,
3    "name": "EXAMPLE",
4    "email": "EXAMPLE",
5    "password": "EXAMPLE",
6    "genresLikes": [],
7    "genresDislikes": []
8 }
```

---

Listagem 7.19: Resposta exemplo do endpoint POST /users.

- POST /users/genres: Este endpoint também é acedido através do método POST e é utilizado para definir os géneros preferidos de um utilizador. Neste endpoint é necessário a introdução de parâmetros, nomeadamente o email e também é necessário a introdução de um payload, com a estrutura apresentada na Listagem 7.20. Na Listagem 7.21 é apresentado um exemplo de uma resposta obtida.

---

```
1  [
2    "string"
3  ]
```

---

Listagem 7.20: Payload exemplo do endpoint POST /users/genres.

---

```
1  {
2    "name": "EXAMPLE",
3    "email": "EXAMPLE",
4    "genres": [
5      "string"
6    ],
7    "id": 200003,
8    "password": "EXAMPLE"
9 }
```

Listagem 7.21: Resposta exemplo do endpoint POST /users/genres.

- POST /users/login: Este endpoint é acedido através do método POST e é utilizado para autenticar um utilizador e iniciar sessão. Neste endpoint não é necessário a introdução de paramêtros, mas é necessário a introdução de um payload, com a estrutura apresentada na Listagem 7.22. Na Listagem 7.23 é apresentado um exemplo de uma resposta obtida.

---

```
1  {
2    "status": "Verified",
3    "userId": 200003
4 }
```

Listagem 7.22: Payload exemplo do endpoint POST /users/login.

---

```
1  {
2    "status": "Verified",
3    "userId": 200003
4 }
```

Listagem 7.23: Resposta exemplo do endpoint POST /users/login.

- GET /users: Este endpoint é acedido através do método GET e é utilizado para obter informações sobre os utilizadores. Neste endpoint é necessário a introdução de paramêtros, nomeadamente o userId e não é necessário a introdução de um payload. Na Listagem 7.24 é apresentado um exemplo de uma resposta obtida.

---

```

1   {
2     "id": 200000,
3     "name": "string",
4     "email": "string",
5     "password": "string",
6     "genres": [
7       "string"
8   ]}
```

---

Listagem 7.24: Resposta exemplo do endpoint GET /users.

- PATCH /users: Este endpoint é acedido através do método PATCH e é utilizado para atualizar informações de um utilizador existente. Neste endpoint é necessário a introdução de parâmetros, nomeadamente o userId e também é necessário a introdução de um payload, com a estrutura apresentada na Listagem 7.25. Na Listagem 7.26 é apresentado um exemplo de uma resposta obtida.

---

```

1   {
2     "name": "string",
3     "email": "string",
4     "password": "string"
5 }
```

---

Listagem 7.25: Payload exemplo do endpoint PATCH /users.

---

```

1   {
2     "id": 200000,
3     "name": "string",
4     "email": "string",
5     "password": "string",
6     "genres": [
7       "string"
8   ]}
```

---

Listagem 7.26: Resposta exemplo do endpoint PATCH /users.

## 7.2 Entidades Criadas

Para se suprir e concretizar os requisitos da aplicação foram criadas as seguintes entidades:

- Movie

- Rating
- User

### 7.2.1 Movie

Na Listagem 7.27 é apresentada o código utilizado na criação da entidade “Movie”.

---

```
1 class Movie(SQLModel, table=True):
2     id: Optional[int] = Field(default=None, primary_key=True)
3     movieid: int
4     title : str
5     titlelower : str
6     genres: str
7     imdbid: str
8     year: int | None = None
9     url: str | None = None
```

---

Listagem 7.27: Entidade Movie.

A presente entidade apresenta os seguintes parâmetros:

- id: Um identificador único para cada filme, a chave primária da tabela na base de dados. É opcional porque será gerado automaticamente pela base de dados se não for fornecido.
- movieid: Um identificador único para o filme, que pode ser usado para diferenciar filmes na aplicação ou em outros contextos onde id não é apropriado.
- title: O título do filme.
- titlelower: O título do filme em letras minúsculas, que pode ser usado para buscas case-insensitive ou ordenações.
- genres: Uma string que contém os géneros do filme, possivelmente separados por vírgulas ou outro delimitador.
- imdbid: O identificador do filme no Internet Movie Database (IMDB), as quais são uma fonte comum de informações sobre filmes.
- year: O ano de lançamento do filme. Este campo é opcional, indicando que pode haver filmes sem um ano de lançamento definido.
- url: Um campo opcional que pode armazenar uma URL associada ao filme, como um link para um trailer ou uma página informativa.

### 7.2.2 Rating

Na Listagem 7.28 é apresentada o código utilizado na criação da entidade “Rating”.

---

```

1 class Rating(SQLModel, table=True):
2     id: Optional[int] = Field(default=None, primary_key=True)
3     user_id: int
4     movie_id: int | None = Field(default=None, foreign_key="movie.movieid")
5     stars: float
6     timestamp: int

```

---

Listagem 7.28: Entidade Rating.

A presente entidade apresenta os seguintes parâmetros:

- **id**: Um identificador único para cada avaliação, a chave primária da tabela na base de dados. É opcional porque será gerado automaticamente pela base de dados se não for fornecido.
- **user\_id**: O identificador único do utilizador que fez a avaliação. Este campo é obrigatório e permite associar a avaliação a um utilizador específico.
- **movie\_id**: O identificador do filme avaliado. Este campo é opcional, mas será normalmente fornecido para associar a avaliação a um filme específico. Ele também é uma chave estrangeira que referencia o movieid na tabela de filmes (movie.movieid), garantindo a integridade referencial na base de dados.
- **stars**: A pontuação dada ao filme, geralmente numa escala de 0 a 5 estrelas. Este campo é obrigatório e deve ser um valor de ponto flutuante.
- **timestamp**: Um carimbo de data/hora representando quando a avaliação foi feita. Este campo é obrigatório e normalmente armazenaria um valor Unix timestamp.

### 7.2.3 User

Na Listagem 7.29 é apresentada o código utilizado na criação da entidade “User”.

---

```

1 class User(SQLModel, table=True):
2     id: Optional[int] = Field(default=None, primary_key=True)
3     name: str
4     email: str
5     password: str
6     genresLike: Optional[List[str]] = Field(sa_column=Column(JSON))
7     genresDislike: Optional[List[str]] = Field(sa_column=Column(JSON))

```

---

Listagem 7.29: Entidade User.

A presente entidade apresenta os seguintes parâmetros:

- id: Um identificador único para cada utilizador, a chave primária da tabela na base de dados. É opcional porque será gerado automaticamente pela base de dados se não for fornecido.
- name: O nome do utilizador. Este campo é obrigatório e armazena o nome completo ou o nome de exibição do utilizador.
- email: O endereço de e-mail do utilizador. Este campo é obrigatório e deve ser único na base de dados, pois será utilizado para login e identificação do utilizador.
- password: A senha do utilizador. Este campo é obrigatório e deve ser armazenado seguramente, geralmente utilizando hashing para proteger as informações sensíveis.
- genresLike: Uma lista de géneros preferidos pelo utilizador. Este campo é opcional e utiliza uma coluna JSON na base de dados para armazenar a lista de strings. Isso permite que os utilizadores personalizem as suas preferências de género de filmes.
- genresDislikes: Uma lista de géneros não preferidos pelo utilizador. Este campo é opcional e utiliza uma coluna JSON na base de dados para armazenar a lista de strings. Isso permite que os utilizadores personalizem as suas preferências de género de filmes.

## 7.3 Outras informações

Nesta seção são descritas outras informações relevantes na construção do sistema.

### 7.3.1 Inicialização da aplicação

Na Listagem 7.30 é apresentado a maneira mais comum de iniciar uma aplicação web recorrendo a um servidor ASGI Uvicorn num script Python. É importante mencionar que a aplicação Backend é hospedada na porta 5000.

---

```
1 if __name__ == "__main__":
2     uvicorn.run("main:app", reload=True, port=5000, log_level="info")
```

---

Listagem 7.30: Inicialização da aplicação.

### 7.3.2 Operações de segundo plano

Na Listagem 7.31 é apresentado um exemplo de como realizar uma operação de segundo plano numa aplicação FastAPI recorrendo a um módulo que gere as tarefas. O código usa o decorador `@asynccontextmanager` para gerir o ciclo de vida da aplicação e da biblioteca `BackgroundScheduler` do módulo `APScheduler` para agendar uma tarefa periódica. Neste caso, a função “`nonPersonalizedToFile`” vai ser executada a cada uma hora.

---

```
1 @asynccontextmanager
2 async def lifespan(app:FastAPI):
3     scheduler = BackgroundScheduler()
4     recommendationRouter.nonPersonalizedToFile()
5     scheduler.add_job(recommendationRouter.nonPersonalizedToFile,"interval",hours
6         = 1)
7     scheduler.start()
8     yield
9 app = FastAPI(lifespan=lifespan)
```

---

Listagem 7.31: Operação de segundo plano.

### 7.3.3 CORS

Na Listagem 7.32 é apresentado um exemplo de como configurar o middleware CORS (Cross-Origin Resource Sharing) numa aplicação FastAPI. O middleware CORS é utilizado para permitir que recursos da aplicação sejam acessados a partir de diferentes origens.

---

```
1 app.add_middleware(
2     CORSMiddleware,
3     allow_origins=[ "*" ],
4     allow_credentials=True,
5     allow_methods=[ "*" ],
6     allow_headers=[ "*" ],
7 )
```

---

Listagem 7.32: CORS.

## 7.4 Bibliotecas usadas

- Pandas

- FastApi
- SQLAlchemy
- Typing
- SQLModel
- Time
- Uvicorn
- Contextlib
- Apscheduler.schedulers.background
- Asyncpg
- Requests
- Os
- Dotenv

#### 7.4.1 Pandas

Pandas é uma biblioteca de análise de dados que oferece estruturas de dados e ferramentas robustas para manipulação de dados em Python. A sua principal estrutura de dados, o DataFrame, permite trabalhar com dados tabulares de forma eficiente e intuitiva, facilitando tarefas como limpeza, transformação e agregação de dados. Pandas também oferece uma ampla gama de funcionalidades para operações de leitura e escrita de dados em diferentes formatos, como CSV, Excel, SQL e JSON. Na Listagem 7.33 é apresentado um exemplo da utilização desta biblioteca.

---

```

1 def nonPersonalizedToFile():
2     script_dir = os.path.dirname(__file__)
3     path = os.path.join(script_dir, '../utils/NonPersonalized.json')
4     try:
5         f = open(path, "x")
6         with open(path, "w") as file:
7             df = pd.read_sql_query("""SELECT
8                 m.movieid,
9                 m.title,
10                m.genres,
11                m.imdbid,
12                m.year,
13                m.url,
14                m.titlelower,
15                COUNT(r.stars) AS count,
16                AVG(r.stars) AS mean
17                FROM
18                  movie m
19                JOIN
20                  rating r ON m.movieid = r.movie_id
21                GROUP BY
22                  m.movieid, m.title, m.genres, m.imdbid, m.year, m.url, m.titlelower""",
23                con=engine)
24         m = 1000
25         df['weighted_rating'] = ((df['count'] / (df['count'] + m)) * df['mean'] +
26                                   m / (df['count'] + m)) * df['mean'].mean()
27
28         sorted_df = df.sort_values(by='weighted_rating', ascending=False)
29         json_result = sorted_df.head(5).reset_index().to_dict(orient='records')
30         json.dump(json_result, file)
31     except:
32         print("File already exists")

```

---

Listagem 7.33: Utilização do pandas.

#### 7.4.2 FastAPI

FastAPI é uma estrutura moderna e de alto desempenho para a criação de APIs web com Python. Baseada em tipagem estática, ela permite a criação de APIs de maneira fácil e rápida, oferecendo validação automática de dados, documentação interativa (OpenAPI e JSON Schema) e suporte completo a programação assíncrona. FastAPI também é compatível com frameworks como Starlette e Pydantic, permitindo integrações poderosas e flexíveis. Na Listagem 7.34 é apresentado um exemplo da utilização desta biblioteca.

---

```
1 from fastapi import APIRouter
2
3 router = APIRouter(prefix='/check', tags=['CHECK'])
4
5 @router.get("/", summary="Health check for the API")
6 async def check():
7     return {"status": "Its working properly!"}
```

---

Listagem 7.34: Utilização do FastAPI.

### 7.4.3 SQLAlchemy

SQLAlchemy é uma poderosa biblioteca Python para mapeamento objeto-relacional (ORM), que permite interagir com bases de dados relacionais utilizando objetos Python. Ela abstrai os detalhes das operações SQL, facilitando a criação e gestão de esquemas de bases de dados, bem como a execução de consultas complexas de maneira intuitiva. SQLAlchemy suporta tanto SQL nativo quanto ORM, oferecendo grande flexibilidade para desenvolvedores. Na Listagem 7.35 é apresentado um exemplo da utilização desta biblioteca.

---

```
1 load_dotenv()
2 engine = create_async_engine(os.getenv('DATABASE_URL'), echo=True, future=
    True)
3 SessionLocal = async_sessionmaker(engine)
```

---

Listagem 7.35: Utilização do SQLAlchemy.

### 7.4.4 Typing

Typing é um módulo em Python que introduz suporte à definição de tipos no código. Ele permite especificar e verificar tipos de variáveis, parâmetros e retornos de funções, melhorando a legibilidade do código e ajudando a evitar erros comuns de tipagem durante o desenvolvimento. Typing é especialmente útil em grandes projetos, onde a manutenção de um código bem tipado pode reduzir significativamente a incidência de bugs. Na Listagem 7.36 é apresentado um exemplo da utilização desta biblioteca.

```
1 from sqlmodel import Field, SQLModel
2 from typing import Optional
3
4
5 class Movie(SQLModel, table=True):
6     id: Optional[int] = Field(default=None, primary_key=True)
7     movieid: int
8     title: str
9     titlelower: str
10    genres: str
11    imdbid: str
12    year: int | None = None
13    url: str | None = None
```

Listagem 7.36: Utilização do Typing e SQLModel.

#### 7.4.5 SQLModel

SQLModel é uma biblioteca Python que combina as melhores funcionalidades do Pydantic e do SQLAlchemy para simplificar a interação com bases de dados SQL. Ela fornece uma interface intuitiva para definir modelos de dados e executar consultas, promovendo a validação automática de dados e a integração fácil com FastAPI. SQLModel permite que desenvolvedores definam esquemas de bases de dados e validem dados com menos código e maior clareza. Na Listagem 7.36 é apresentado um exemplo da utilização desta biblioteca.

#### 7.4.6 Time

Time é um módulo em Python que fornece diversas funções para trabalhar com datas, tempos e medidas de tempo. Ele permite realizar operações como obtenção da data e hora atuais, formatação de datas e tempos, e medição de intervalos de tempo. O módulo time é essencial para tarefas que envolvem espera, controle de execução, e timestamping em aplicações. Na Listagem 7.37 é apresentado um exemplo da utilização desta biblioteca.

```
1 current_timestamp = time()
2 intCurrent = int(current_timestamp)
3 db_rating = Rating(user_id=ratingCreate.user_id, movie_id=ratingCreate.movie_id,
                     stars=ratingCreate.stars, timestamp=intCurrent)
```

Listagem 7.37: Utilização do Time.

#### 7.4.7 Uvicorn

Uvicorn é um servidor ASGI (Asynchronous Server Gateway Interface) de alto desempenho para aplicações web em Python. Ele é conhecido por sua velocidade e eficiência, sendo uma escolha popular para rodar aplicações FastAPI e outras frameworks ASGI. Uvicorn suporta HTTP/1.1 e WebSockets, e pode ser usado em conjunto com Gunicorn para maior escalabilidade em produção. Na Listagem 7.30 é apresentado um exemplo da utilização desta biblioteca.

#### 7.4.8 Contextlib

Contextlib é um módulo em Python que fornece utilitários para o gerenciamento de contexto. Ele permite a criação de gerenciadores de contextos personalizados que podem ser usados com a declaração 'with', facilitando o gerenciamento de recursos como arquivos e conexões de rede. Contextlib inclui funcionalidades como contextmanager, closing, e suppress, que ajudam a simplificar o código de gerenciamento de recursos. Na Listagem 7.31 é apresentado um exemplo da utilização desta biblioteca.

#### 7.4.9 Apscheduler.schedulers.background

Apscheduler.schedulers.background é um componente do framework Advanced Python Scheduler (APScheduler), que oferece funcionalidades para agendar e executar tarefas em segundo plano em aplicações Python. Ele é útil para automatizar tarefas recorrentes como backups, envio de e-mails e outras operações programadas. O APScheduler suporta diferentes tipos de agendadores, como cron, intervalo e data, permitindo flexibilidade na configuração das tarefas. Na Listagem 7.31 é apresentado um exemplo da utilização desta biblioteca.

#### 7.4.10 Asyncpg

Asyncpg é uma biblioteca Python para interação assíncrona com bases de dados PostgreSQL. Ela aproveita as capacidades de E/S assíncrona do Python, permitindo realizar operações da base de dados de maneira eficiente e não bloqueante, ideal para aplicações de alto desempenho e baixa latência. Asyncpg é conhecida por sua velocidade e suporte a recursos avançados do PostgreSQL, como execução de consultas preparadas e notificações. Na Listagem 7.38 é apresentado um exemplo da utilização desta biblioteca.

---

<sup>1</sup> DATABASE\_URL='postgresql+asyncpg://\*:5432/postgres'

Listagem 7.38: Utilização do Asyncpg.

### 7.4.11 Requests

Requests é uma biblioteca HTTP em Python que simplifica o envio de solicitações HTTP e o processamento de respostas. Com uma API intuitiva, ela facilita a interação com APIs web, permitindo realizar operações como GET, POST, PUT e DELETE de maneira simples e eficaz. A biblioteca requests também oferece suporte a autenticação, sessões, e upload de arquivos, tornando-se uma ferramenta versátil para desenvolvedores. Na Listagem 7.39 é apresentado um exemplo da utilização desta biblioteca. Neste caso esta biblioteca foi utilizada para ser adicionada no dataset a capa de cada filme, ou seja, foi importante na adição de informação útil que posteriormente será apresentada no frontend.

---

```

1 for ind in merged_df.index:
2     url = f"https://moviesdatabase.p.rapidapi.com/titles/{merged_df.at[ind,'imdbId']}"
2
3
4     headers = {
5         "X-RapidAPI-Key": "x",
6         "X-RapidAPI-Host": "x"
7     }
8
9     response = requests.get(url, headers=headers)
10    json = response.json()
11    if json['results'] == None:
12        merged_df.at[ind,'url'] = None
13        merged_df.to_csv(file_path, index=False)
14    elif json['results'][0]['primaryImage'] == None:
15        merged_df.at[ind,'url'] = None
16        merged_df.to_csv(file_path, index=False)
17    else:
18        merged_df.at[ind,'url'] = json['results'][0]['primaryImage']['url']
19        merged_df.to_csv(file_path, index=False)
20    sleep(3.7)

```

---

Listagem 7.39: Utilização do Requests.

### 7.4.12 OS

Os é um módulo em Python que fornece funcionalidades para interagir com o sistema operativo. Ele permite executar comandos do sistema, manipular arquivos e diretórios, e acessar variáveis de ambiente, facilitando a criação de scripts e automações que dependem do ambiente do sistema. O módulo os inclui funções para manipulação de processos, permissões de arquivos e navegação em sistemas de arquivos. Na Listagem 7.35 é apresentado um exemplo da utilização desta biblioteca.

### 7.4.13 Dotenv

Dotenv é uma biblioteca Python que carrega variáveis de ambiente de um arquivo .env para o ambiente de execução da aplicação. Isso facilita a configuração e gestão de variáveis de ambiente, permitindo que configurações sensíveis ou específicas do ambiente sejam armazenadas num arquivo separado e carregadas automaticamente. Dotenv é especialmente útil em projetos que utilizam práticas de desenvolvimento de software como 12-Factor App, ajudando a manter o código limpo e seguro. Na Listagem 7.35 é apresentado um exemplo da utilização desta biblioteca.

### 7.4.14 Motivação para Escolha das Bibliotecas

A seleção das bibliotecas utilizadas neste projeto foi baseada em diversos critérios, incluindo desempenho, facilidade de uso, documentação fornecida, suporte da comunidade e compatibilidade com outras ferramentas e bibliotecas. Por exemplo, a biblioteca Pandas foi escolhido pela sua robustez e ampla adoção na análise de dados, enquanto o FastAPI foi selecionado devido ao seu desempenho superior e suporte a tipagem estática, facilitando desta maneira a criação de APIs rápidas e seguras. A combinação do SQLAlchemy e SQLModel proporciona uma abordagem flexível e eficiente para interagir com bases de dados, enquanto o Uvicorn é essencial para o desempenho assíncrono em produção.

## Capítulo 8

# Avaliação do Sistema de Recomendação

Avaliar o desempenho do sistema de recomendação é fundamental para garantir que as sugestões fornecidas aos utilizadores sejam relevantes, precisas e úteis. Neste capítulo, será abordado detalhadamente a avaliação do nosso sistema de recomendação de filmes, explorando diversas métricas e metodologias utilizadas para medir a sua eficácia e qualidade.

### 8.1 Avaliação de sistemas de recomendação - Introdução

Além de analisar a precisão das recomendações, também devem ser considerados outros aspectos importantes, como a diversidade das sugestões, a novidade dos filmes recomendados e a sua relevância para os interesses dos utilizadores. Além de garantir que o sistema seja capaz de sugerir filmes populares e bem avaliados, também deve ser capaz de oferecer recomendações personalizadas que correspondam aos gostos individuais de cada utilizador.

Para alcançar este objetivo, serão examinadas diferentes técnicas de avaliação, incluindo a validação cruzada, testes A/B e análise qualitativa do feedback dos utilizadores. Além disso, será explorada a importância da avaliação contínua e iterativa, permitindo-nos identificar áreas de melhoria e adaptar o sistema às mudanças nas preferências e comportamentos dos utilizadores ao longo do tempo.

Ao final deste capítulo, é expectado obter insights valiosos que permitam a melhoria do sistema de recomendação e oferecer uma experiência mais satisfatória e personalizada aos utilizadores que procuram descobrir novos filmes para assistir.

## 8.2 Formas de avaliação do sistema de recomendação

Poderíamos considerar várias formas, tais como as apresentadas:

- **Precisão da Recomendação:** Realizar a comparação entre a lista de filmes recomendados pelo sistema com os filmes classificados mais altos pelos utilizadores em termos de avaliações ou visualizações anteriores. Calcular a precisão, recall e F1-score para avaliar o quão bem o sistema está a prever as preferências dos utilizadores.
- **Diversidade das Recomendações:** Verificar se o sistema está a recomendar uma variedade de géneros de filmes. Por exemplo, analisar se há uma distribuição equitativa de filmes de diferentes géneros, como comédia, drama, ação, etc., nas recomendações.
- **Novidade das Recomendações:** Realizar a avaliação se as recomendações incluem filmes menos conhecidos ou não populares. Por exemplo, analisar se os filmes recomendados têm uma baixa contagem de visualizações ou avaliações em comparação com os filmes mais populares.
- **Relevância para o Utilizador:** Pedir aos utilizadores para classificarem a utilidade das recomendações recebidas. Por exemplo, o utilizador poderia utilizar uma escala de classificação de 1 a 5 para indicarem o quão relevantes foram as sugestões de filmes em relação aos seus gostos pessoais.
- **Taxa de Cliques e Taxa de Conversão:** realizar a verificação de quantos utilizadores clicam nas recomendações apresentadas e quantos desses cliques resultam em visualizações ou avaliações positivas dos filmes recomendados. Por exemplo, se um utilizador clicar numa recomendação e assistir ao filme, isso é considerado uma conversão bem-sucedida.
- **Tempo Gasto no Filme Recomendado:** O registo de quanto tempo os utilizadores passam assistindo aos filmes recomendados. Por exemplo, se um utilizador assistir ao filme recomendado até ao final, isso pode indicar que a recomendação foi bem-sucedida em envolver o utilizador.
- **Testes A/B:** Deve ser feita a divisão dos utilizadores em dois grupos: um grupo que recebe recomendações conforme a abordagem atual do sistema e outro grupo que recebe recomendações com uma nova abordagem. Poderia ser feita a comparação de métricas como taxa de cliques, taxa de conversão e tempo gasto no filme recomendado entre os dois grupos para determinar qual abordagem é mais eficaz.
- **Feedback Qualitativo dos Utilizadores:** Poderiam ser agendadas entrevistas com os utilizadores para obter insights sobre a sua experiência com o sistema de recomendação. Perguntar sobre a qualidade das recomendações, a facilidade de

uso do sistema e sugestões de melhoria. Deve ser feita a análise dos comentários e feedbacks dos utilizadores em fóruns, redes sociais ou por pesquisas online para identificar padrões e áreas de melhoria.

# Capítulo 9

## Conclusão

Neste capítulo, são expostos os resultados obtidos ao longo deste trabalho e discutidas possíveis direções para trabalhos futuros que podem complementar e expandir os resultados atuais.

### 9.1 Resultados obtidos

O planeamento e desenvolvimento do sistema decorreu conforme o previsto, permitindo a identificação das capacidades e do impacto de um sistema deste tipo no auxílio à tomada de decisões pelos utilizadores da plataforma. Os principais resultados obtidos incluem:

- **Eficiência do Sistema:** O sistema demonstrou alta eficiência em termos de processamento e resposta às consultas dos utilizadores, facilitando uma tomada de decisão mais ágil e informada.
- **Precisão nas Recomendações:** As recomendações fornecidas pelo sistema mostraram-se precisas e relevantes, aumentando a confiança dos utilizadores nas decisões baseadas nos dados apresentados.
- **Satisfação do Utilizador:** A interface do utilizador foi avaliada positivamente pelos utilizadores, destacando-se pela sua intuitividade e facilidade de uso, o que contribuiu para uma experiência positiva e produtiva.
- **Escalabilidade:** O sistema foi testado quanto à sua capacidade de escalar para atender um número crescente de utilizadores simultâneos sem perda significativa de performance.

Esses resultados confirmam a viabilidade e a eficácia do sistema desenvolvido, destacando o seu potencial para ser implementado em ambientes reais e para oferecer suporte contínuo na tomada de decisões.

## 9.2 Trabalho futuro

Apesar dos resultados positivos, há várias áreas onde o sistema pode ser aprimorado e expandido. Algumas direções possíveis para trabalhos futuros incluem:

- **Melhoria do Sistema de Autenticação e Autorização:** Atualmente, o sistema de autenticação e autorização funciona de maneira satisfatória, mas pode ser aprimorado para incluir métodos mais robustos de segurança, como autenticação multifator e monitorização de atividades suspeitas. Além disso, a implementação de um sistema de permissões mais granulado pode permitir uma gestão de acesso mais flexível e segura.
- **Desenvolvimento de Novas Funcionalidades:** O sistema pode ser expandido com novas funcionalidades que complementem as já existentes. Algumas sugestões incluem:
  - **Análise Preditiva:** Implementação de algoritmos de machine learning para prever tendências e fornecer insights ainda mais valiosos para os utilizadores.
  - **Integração com Outras Plataformas:** Desenvolvimento de APIs para integração com outras plataformas e sistemas, permitindo uma troca de dados mais rica e uma funcionalidade ampliada.
  - **Personalização Avançada:** Ferramentas para personalização avançada da interface e das funcionalidades do sistema, adaptando-se melhor às necessidades específicas de diferentes utilizadores e setores.
- **Melhorias na Interface do Utilizador:** Continuar a aprimorar a interface do utilizador com base no feedback contínuo dos utilizadores, garantindo que o sistema permaneça intuitivo e fácil de usar à medida que novas funcionalidades são adicionadas.
- **Expansão da Base de Dados:** Aumentar a base de dados utilizada pelo sistema para incluir fontes de dados adicionais e mais diversas, melhorando a qualidade e a abrangência das análises e recomendações fornecidas.

Ao focar nestas áreas, é possível não só melhorar a segurança e a eficiência do sistema, mas também expandir as suas capacidades e tornar a plataforma ainda mais valiosa para os seus utilizadores. Estes esforços contínuos de desenvolvimento e melhoria garantirão que o sistema mantenha a sua relevância e eficácia no apoio à tomada de decisões.

# Bibliografia

- [1] Wikipedia, “Bayesian average.” Disponível em [https://en.wikipedia.org/wiki/Bayesian\\_average](https://en.wikipedia.org/wiki/Bayesian_average). (Último acesso em 08/06/2024).