# Recomendador de Filmes

**Trabalho de Programação em Lógica**

Realizado pelos alunos:
Maximiliano Vítor Phillips e Sá (up202305979),
Orlando Miguel Carvalho-Soares (up202303606),
e Rita Maria Pinho Moreira (up202303885)

# Índice

# 1. Introdução

O seguinte projeto foi desenvolvido em contexto académico, de forma a consolidar competências de programação lógica. Para tal efeito, foi criado um sistema de recomendação de filmes, com interface web e integração com a API TMDb (The Movie Database).

O servidor principal, server.pl, junta todos os módulos auxiliares: gestão de utilizadores avaliações, recomendações, integração TMDb e interface HTML.

# 2. Requisitos/Dependências do Sistema

- OS: Windows/ Linux/ MacOS.
- SWI-Prolog instalado no PATH do sistema (para funcionar no terminal).

# 3. Como configurar

- Descarregue a pasta.
- Aceda ao diretório "data" no terminal.
- Execute o comando: *swipl server.pl*
- Consulta: *server(PORT).* (Substitua o PORT pela porta pretendida, por exemplo, 8080).
- O servidor será executado em: *localhost:PORT*.

# 4. Implementação

Para a criação deste trabalho, foram criados diversos ficheiros, ambos contendo factos e regras respetivas a cada tópico abordado no mesmo. Esses ficheiros foram usados para o ficheiro server.pl, que culmina todo o código para o sistema de recomendação funcionar devidamente.

## Explicação do código no geral:

**Cabeçalho:**
Para este sistema foram usadas variadas librarias built-in, tais como:

- library(http/http_session): gere sessões HTTP;
- library(http/thread_httpd): inicia o servidor HTTP;
- library(http/http_dispatch): define handlers para rotas;
- library(http/http_parameters): lê parâmetros de pedidos GET/POST; library(http(html_write): gere HTML em DCG[1];
- library(http/http_client): fornece as principais ações HTTP, GET, DELETE; POST e PUT;

[1] Definite Clause Grammar, um método para expressar gramáticas em linguagens de programação lógica, como Prolog.

- library(http/http_open): abre os dados num servidor HTTP como uma reprodução do código Prolog;
- library(apply): define meta-predicados que aplicam um predicado em todos os membros de uma lista
- library(http/json): suporta a leitura e escrita em objetos JSON;
- library(date): processa datas, definindo predicados de tempo e datas, tal como estruturas de dados;
- library(uri): fornece primitivas baseadas em C de alta performance, usadas para manipular URIs;
- library(http/http_files): serve ficheiros simples de uma hierarquia.

Devido à utilização de ficheiros de certa forma externos ao ficheiro principal, foram usados os predicados ensure_loaded/1, cujo argumento é um ficheiro dado para importar os seus predicados públicos, e module/1, que torna um módulo default de trabalho num módulo iterativo, visto que estamos a usar diversos ficheiros num só. Assim, foram importados os predicados dos ficheiros users.pl e recommend.pl, e foram importados os módulos knn, tmdb_integration e rating.

O predicado initialization/1, que carrega um ficheiro dado, foi usado para definir a chave de acesso à API do TMDB (The Movie Database), usando um predicado set_tmdb_api_key/1 pertencente ao módulo carregado tmdb_integration, e para inicializar a base de dados das avaliações, com o predicado init_ratings_db/1, pertencente ao módulo rating, que, caso existe o ficheiro ratingsdb.pl, carrega-o.

Foi utilizado o predicado http_handler/3, com a função de processar pedidos de endpoints individuais, não só processaram páginas de site como ficheiros estáticos de CSS e imagens. Cada http_handler(root(X), Predicado, Opções) associa a rota "/X", ou, para root(.), a rota "/", a um predicado que vai gerar resposta HTML ou executar alguma lógica.
Por último, foi definida a base do API, com a chave (tmdb_api_key/1) e url (tmdb_base_url/1).

**Layout:**
Para o layout do server foi usado o predicado html_meta/1, que declara uma regra de renderização HTML, usando o seu conteúdo como argumento.
Foi definido um page_wrapper/2, com argumentos o título da página e o corpo, com acesso ao CSS e ao estado atual do utilizador.

**Server:**
Para gerar a sessão de HTTP, foi definido um predicado server/1, de argumento Port, que utiliza a biblioteca http_server/2, e http_dispatch e Port como respetivos argumentos.

**Movies:** base de dados, ficheiro gendb.pl, tmdb usage (tmdb_integration.pl)

Para se poder aceder a filmes e suas respetivas informações, foi gerada, a partir do dataset imdb movies.csv, uma base de dados em Prolog, armazenada no

ficheiro movie.pl. É usado o predicado db/3 para guardar o ID do filme e os seus dados, tais como o seu nome, duração, ano de estreia, avaliação, país, realizador e género. A base de dados foi gerada usando o ficheiro gendb.pl.

**Users:** DCG, base de dados e ficheiro users.pl

Tendo os filmes listados, é necessário ter usuários para os adicionar e avaliar. Para gerar a base de dados userdb.pl foram usados predicados do ficheiro users.pl, como new_user/2. Para cada utilizador registado, é gerada a sua password encriptada, para maior segurança, e os seus dados (nome de utilizador e password) serão armazenados na base de dados. Se o nome de utilizador já existir, a mensagem será "Username already exists", se não será "New User added and logged in".

**Recommend:** KNN, recommend.pl

O recomendador que tem como base o KNN cria um vetor (User-Rating) para cada filme, calcula a similaridade do cosseno entre todos os vetores, criando um novo vetor (Similarity–Film), remove os filmes já vistos pelo utilizador dessa nova lista, ordena por ordem decrescente a similaridade, e retorna no máximo N (10 no nosso caso) filmes. Se N = 10 fica por aí, se N < 10, o recomendador usa os N filmes que encontrou com o KNN e depois recorre a fallbacks [2]como por exemplo um recomendador que recomenda filmes do género favorito do utilizador (10-N). Logo o recomendador acaba sempre por recomendar 10 filmes. Na página do recomendador temos uma estatística que mostra quantos filmes foi o KNN e/ou os fallbacks.

O outro recomendador mais básico, recebe input do utilizador em forma de perguntas pré escolhidas, e com base nas respostas, devolve os filmes que atendem aos requisitos.

**Ratings:** rating.pl, ratingsdb.pl

Os utilizadores podem dar ratings (0-5 estrelas) e deixar um comentário sobre os filmes que viram. Esta funcionalidade também é uma peça fundamental para o nosso recomendador que usa KNN. Pois é dando match de ratings entre utilizadores que o recomendador decide se filmes devem ser recomendados a certo utilizador ou não. Os ratings são guardados como factos no ficheiro "ratingsdb.pl".

[2] Plano de contingência, opção alternativa ou mecanismo de reserva, usado quando a opção primária não é possível ou não funciona.

## 5. Resultados:

Cada página do site tem o utilizador logged-in no canto superior esquerdo e um logo no canto superior direito. O logo no canto superior direito redireciona o utilizador de volta para a página inicial quando clicado.

### Home Page

A página inicial apresenta o título " Welcome to Prolog, the Movie Recommender" e o subtítulo " All of your movie needs, in one place! :)" e tem os botões seguintes:

- Register
- Login
- Get Recommendations
- Show Your Films
- Show Your Ratings
- Show All Films
- Logout

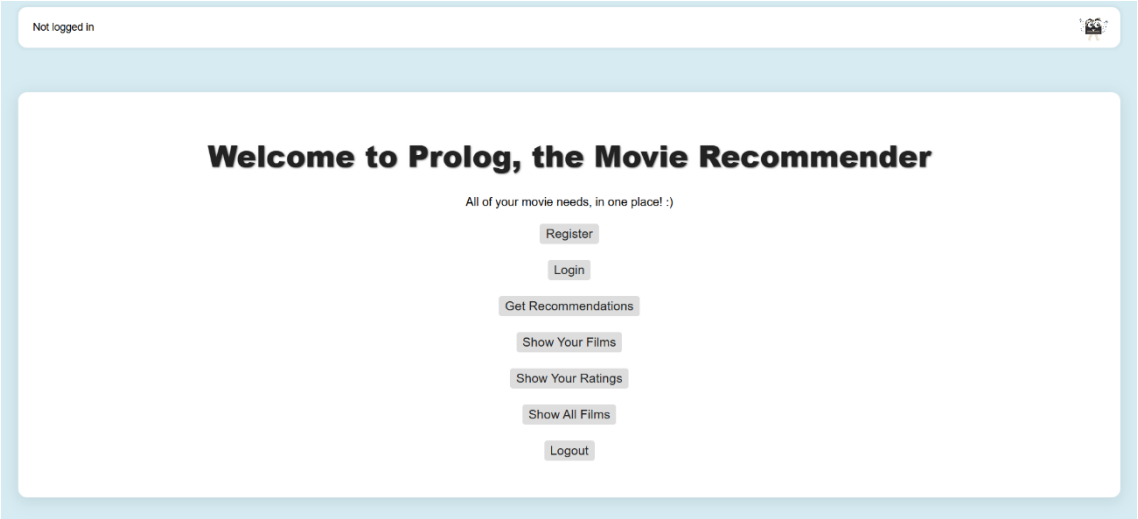


*Figura 1: Home Page (user não logado)*

### Register

A página permite ao utilizador criar uma conta fornecendo um nome de utilizador e uma palavra-passe. O nome de utilizador deve ser um que ainda não exista na base de dados.

### Login

Permite ao utilizador efetuar o login na sua conta com as suas credenciais de nome de utilizador e palavra-passe.

### Get Recommendations

Abre uma página de menu com as seguintes opções de recomendação:

- *Based on Your Films*: O utilizador precisa de estar logged-in para funcionar. Gera uma lista de recomendações com base nas avaliações dos

utilizadores e noutros filmes semelhantes com uma classificação elevada. Nas figuras a baixo está uma demonstração da página, sendo que a primeira corresponde ao user Max e a segunda ao user Orlando.

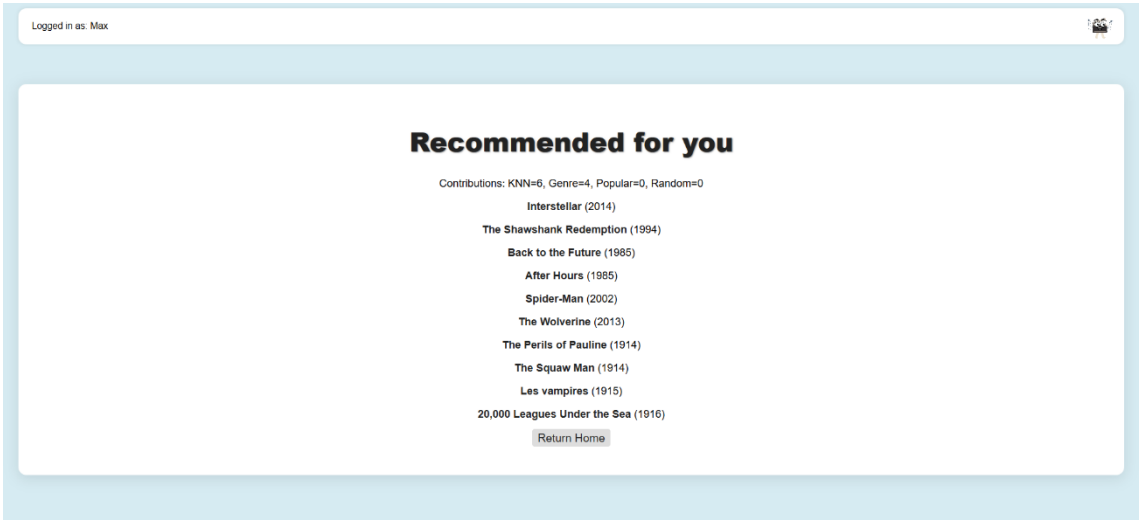


*Figura 2: Recomendações por lista de filmes (Utilizador Max)*

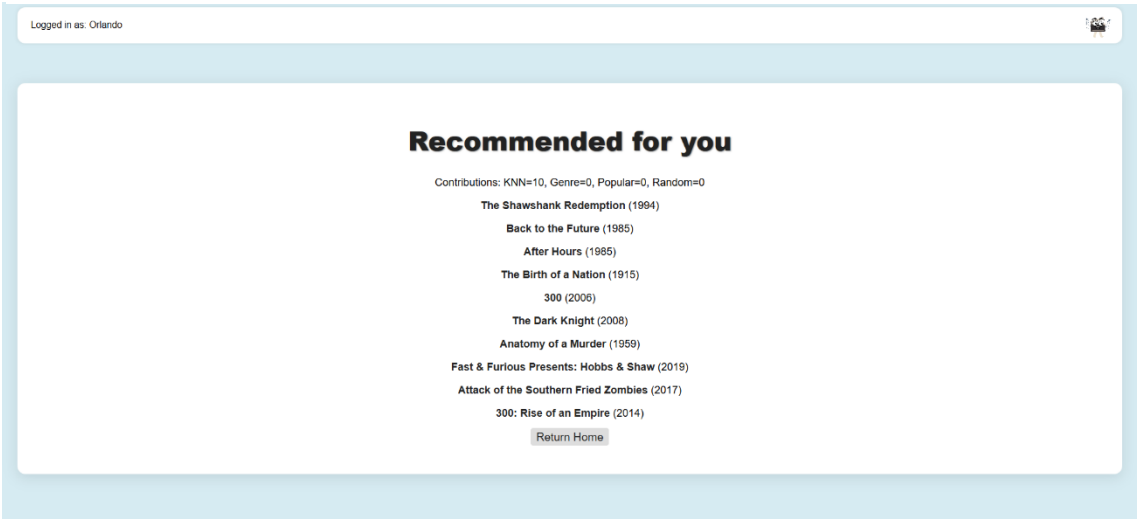


*Figura 3: Recomendações por lista de filmes (Utilizador Orlando)*

- *By Specific Questions*: Abre uma página com as seguintes questões e gera recomendações com base nas respostas (se fornecidas):
    - *Do you prefer older movies (pre-2000)?*
        - *Yes (pre-2000)/ No, modern movies*
    - *What types of movies are you in the mood for?*
        - *Emotional/ Historical/ Cerebral/ Adventurous/ Funny*
    - *Do you have time for longer movies?*
        - *Yes (>119min)/ No (<120min)*
    - *Which country's movie do you prefer?*
        - *US/ UK/ Canada/ Japan/ Korea/ China*
    - *Do you prefer high-scoring movies?*
        - *Yes (>7)*

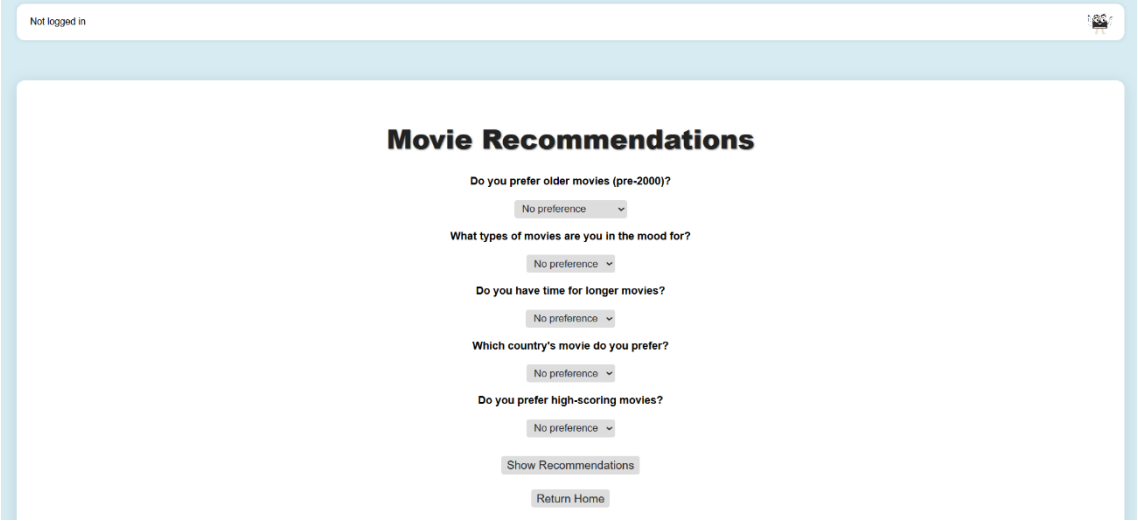


*Figura 4: Recomendações por perguntas específicas*

## Show Your Films

É necessário estar logged-in para ter acesso. Apresenta uma lista dos filmes que o utilizador adicionou à sua conta, com as opções de remover ou fazer um rating ao lado de cada "título (ano)" da lista. O utilizador pode clicar no nome do filme para abrir a página respectiva.

## Show Your Ratings

É necessário estar logged-in para ter acesso. Apresenta uma lista de todas as ratings feitas pelo utilizador atual e uma opção para ver os seus filmes ou regressar à página incial.

## Show All Films

Apresenta uma lista de todos os filmes na base de dados com opções de filtro (ano, país, género), um campo para pesquisar pelo nome do filme e botões para aplicar os filtros fornecidos pelo utilizador e repor os filtros para os valores predefinidos. Na parte inferior da página existe uma opção para voltar à página inicial.

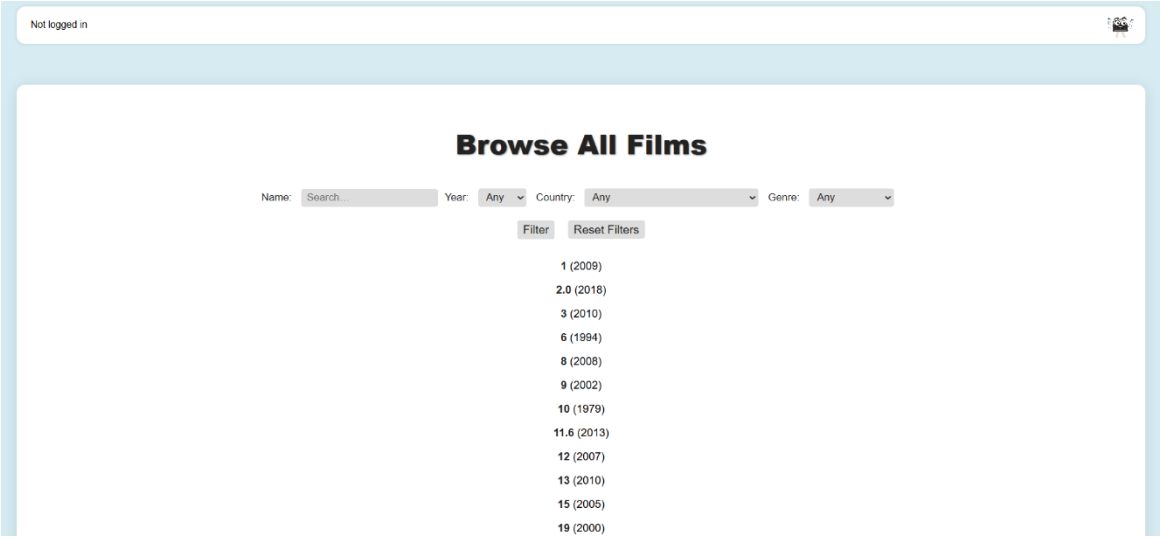


*Figura 5: Lista de todos os filmes disponíveis*

## Film Pages

Página que exibe um poster e detalhes de qualquer filme, bem como botões para adicionar aos seus filmes, fazer um rating ou aceder à página oficial do IMDB. É necessário estar logged-in para ter acesso aos botões para adicionar aos seus filmes ou fazer um rating. Os ratings feitos pelos utilizadores para o filme também são listados na página do filme, juntamente com a opção de voltar à página inicial.

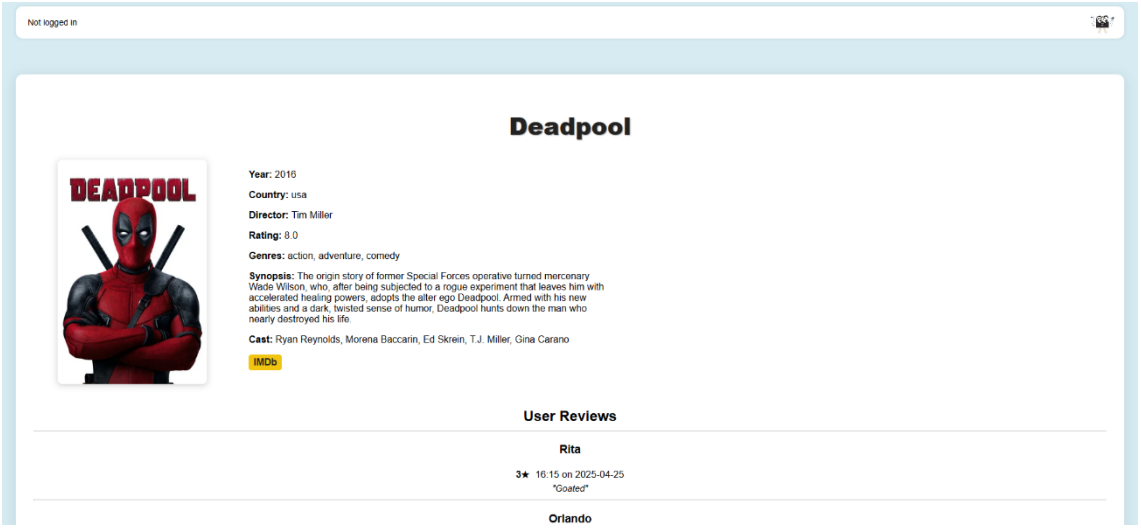


*Figura 6: Informações sobre filme Deadpool*

### Logout
Encerra a sessão do utilizador atual e permanece na página inicial.

## 6. Escalabilidade

O sistema ainda está em fase inicial, mas possui diversos aspetos técnicos e organizacionais que podem permitir um crescimento sustentável.

### Crescimento do Volume de Dados

À medida que o sistema for adotado por mais utilizadores e passar de milhares para centenas de milhares de filmes e milhões de avaliações, torna-se impraticável manter todos os factos em memória num único ficheiro Prolog. Uma mitigação seria a migração para uma base de dados, isto é, serializar factos Prolog para uma base de dados relacional, onde consultas indexadas garantem leituras/escritas eficientes. Outra solução seria *sharding* de dados, ou seja, particionar coleções por género, ano, ID de utilizador, etc., para distribuir o armazenamento e reduzir tempos de resposta em consultas muito grandes.

### Otimização do Motor Prolog

O SWI-Prolog carrega factos em memória e usa índices para acelerar chamadas, mas pode ser usada uma indexação avançada, que identifica predicados usados com maior frequência e garante índices compostos para consultas multi-campo, e/ou, em vez de carregar todo o ficheiro .pl de uma só vez, usar load_files/2 com opções de lazy loading[3], ou até mesmo modularizar ainda mais o código e carregar módulos específicos quando necessários.

### Caching e Pré-computação

Para reduzir a latência e carga de CPU pode-se recorrer a uma cache de sessão que, após gerar recomendações para um utilizador, guarda-as num cache durante X horas, invalidando-as sempre que o utilizador submeter novas avaliações. Outra ideia é o pré-cálculo offline, que consiste em, num processo batch diário, calcular matrizes de similaridade (KNN) e guardá-las num formato binário para acesso imediato no runtime.

### Arquitetura Distribuída

[3] Técnica onde recursos não são carregados imediatamente quando a página web é solicitada, mas sim sob demanda, normalmente quando o utilizador desce até aos mesmos ou interage com a página.

Separar responsabilidades evita gargalos num único processo. Para tal, podem ser usados microsserviços [4]como: um serviço de recomendação, que expõe apenas a lógica Prolog/KNN; um serviço de autenticação, em Node.js ou Python, que gere hashes de password; e como frontend o uso de React ou outra stack, que consome as APIs REST.

### Escalabilidade Horizontal

Para atingir este aspeto, cada microserviço pode ter múltiplas réplicas atrás de um balanceador de carga, ou seja, uma replicação de instâncias, como também a base de dados deve ser única ou um cluster de BD, e as sessões e cache devem ser centralizadas.

### Gerência de Rate Limits da TMDb

A TMDb API impõe limites de chamadas por segundo/dia. Para resolver esse possível problema pode ser implementada uma lógica que deteta erros 429 e faz back.off exponencial antes de tentar novamente, tal como armazenar respostas de endpoints da TMDb, como descrições, posters e ratings agregados, numa base rápida para evitar pedidos redundantes.

### Testes de Carga e Benchmarking[5]

É necessário também validar empiricamente a escalabilidade do sistema antes de entrar em produção, usando cenários de stress, isto é, simular centenas ou milhares de utilizadores concorrentes a pedir recomendações, determinar se a adição de mais réplicas reduz linearmente a latência, ou se surgem novos gargalos, ou utilizar testes de resiliência, que desligam instâncias a quente (chaos testing[6]) para observar recuperação automática pelo orquestrador.

## 7. Conclusão

Em conclusão, a criação de um sistema de recomendação de filmes em Prolog permitiu-nos aprimorar competências de programação lógica e estimular criatividade, tal como aprender técnicas como o KNN, a integração de APIs externas e a gestão de factos Prolog. Permitiu-nos também desenvolver uma interface em DCG reforçar aspetos de segurança (hashing[7] e armazenamento seguro de passwords). Constatou-se que o sistema apresenta elevada precisão nas recomendações, mas também algumas limitações ao nível da escalabilidade.

[4] Abordagem de desenvolvimento de software onde uma aplicação grande é dividida em pequenos serviços independentes e autónomos.
[5] Processo de comparar e analisar o desempenho de uma organização com o de outras.
[6] Método de teste de resiliência do sistema, que insere intencionalmente falhas e interrupções num sistema para observar como se comporta sob stress.
[7] Técnica que transforma dados num valor fixo, chamado de hash ou código de hash, usando uma função matemática chamada função hash.

No futuro, este projeto poderá evoluir não só com as estratégias descritas no capítulo de escalabilidade, mas também através de integrações com aplicações móveis ou outras plataformas, caminhando assim para um produto real.

## 8. Referências

kaggle.com. (n.d.). *IMDb movies extensive dataset*. [online] Available at: https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset.

The Movie Database (2018). *The Movie Database*. [online] Themoviedb.org. Available at: https://www.themoviedb.org/.

Wilk, J. (2023). *How to Build a Movie Recommendation System Based on Collaborative Filtering*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/how-to-build-a-movie-recommendation-system-based-on-collaborative-filtering/.