Desenvolvimento de um Sistema de Recomendação para Músicas, Filmes e Jogos

Alexsander Brum Cristofari, Alessandro André Mainardi de Oliveira
Curso de Ciência da Computação
UFN - Universidade Franciscana
Santa Maria - RS
alexcristofari2@gmail.com, alessandroandre@ufn.edu.br

***Abstract*—O crescimento exponencial de conteúdo digital tem resultado em uma sobrecarga de informações, dificultando a descoberta de recomendações relevantes para os usuários. Este trabalho apresenta o desenvolvimento de uma plataforma digital para fornecer recomendações personalizadas de músicas, jogos e filmes. A arquitetura do sistema é baseada no modelo cliente-servidor, com um *frontend* desenvolvido em React e um *backend* em Python exposto por uma API HTTP. Foi desenvolvido um protótipo de banco de dados utilizando SQL Server, preparado para implementação futura de filtragem colaborativa. Na versão atual, o sistema emprega filtragem baseada em conteúdo, utilizando cálculo de similaridade de cosseno para gerar recomendações personalizadas. Os dados têm origem em arquivos CSV, que são pré-processados em matrizes numéricas otimizadas para consulta. O resultado é uma plataforma funcional que facilita a descoberta de conteúdo digital de forma personalizada.**

***Index Terms*—Sistemas de Recomendação, Filtragem Baseada em Conteúdo, Similaridade de Cosseno, React, Python**

I. Introdução

O acesso a um grande volume de conteúdo digital, como filmes, músicas e jogos, tornou-se uma característica central da era digital [1]. Essa abundância, contudo, gera o desafio da sobrecarga de informação, dificultando a capacidade dos usuários de encontrar conteúdos alinhados aos seus interesses. Como resposta a esse problema, os sistemas de recomendação foram desenvolvidos e integrados como componentes essenciais em diversas plataformas online, atuando como mecanismos de filtragem para auxiliar na descoberta de novos itens [2].

A implementação desses sistemas é, em geral, focada na personalização da experiência do usuário, utilizando algoritmos para analisar comportamento e prever preferências. No entanto, essa abordagem levanta questões sobre autonomia, transparência e influência algorítmica [3]. Frequentemente, o objetivo principal de plataformas comerciais é a retenção da atenção do usuário, o que pode levar ao desenvolvimento de mecanismos de consumo contínuo de conteúdo, associados a padrões de uso prolongado [4]. Essa dinâmica pode resultar na formação de "bolhas de filtro" [5], limitando a exposição do usuário a novas perspectivas e reduzindo diversidade de descoberta.

Neste contexto, o presente trabalho propõe uma plataforma de recomendação que explora principalmente características internas das mídias consumidas, sem depender de grandes volumes de interações de múltiplos usuários. O foco está na clareza do funcionamento do algoritmo e na promoção de uma experiência de descoberta mais ativa e consciente.

A. Justificativa

O problema central abordado por este projeto é a passividade do usuário frente a algoritmos otimizados para engajamento contínuo, o que pode limitar a descoberta orgânica de novo conteúdo cultural. A justificativa para este trabalho reside na criação de uma alternativa a esse modelo. Propõe-se uma plataforma que utiliza técnicas de filtragem baseada em conteúdo para oferecer recomendações personalizadas a partir de características das mídias, resultando em uma experiência de descoberta mais significativa e alinhada aos interesses genuínos do usuário.

B. Objetivo Geral

Desenvolver uma plataforma digital para a recomendação de músicas, filmes e jogos, utilizando técnicas de filtragem baseada em conteúdo e cálculo de similaridade.

C. Objetivos Específicos

- Implementar uma arquitetura de sistema cliente-servidor, utilizando React no *frontend* e serviços em Python no *backend*;
- Projetar um protótipo de banco de dados em SQL Server para suporte futuro a funcionalidades avançadas;
- Implementar um sistema de recomendação baseado em filtragem de conteúdo, utilizando cálculo de similaridade de cosseno;
- Processar e vetorizar *datasets* de músicas, filmes e jogos para possibilitar o cálculo de similaridade;
- Desenvolver interfaces de usuário intuitivas para seleção de preferências e visualização de recomendações.

II. Referencial Teórico

Esta seção apresenta os conceitos fundamentais que embasam sistemas de recomendação e técnicas utilizadas em cenários de descoberta de conteúdo digital.

A. Sistemas de Recomendação

Sistemas de recomendação são sistemas de software que fornecem sugestões de itens que possam ser do interesse de um determinado usuário [1]. Esses sistemas auxiliam na navegação em catálogos extensos, reduzindo a sobrecarga de informação

e aumentando a probabilidade de descoberta de itens relevantes [2].

As abordagens mais comuns são [2]:

- filtragem baseada em conteúdo (*Content-Based Filtering*);
- filtragem colaborativa (*Collaborative Filtering*);
- sistemas híbridos, que combinam múltiplas técnicas [9].

B. Filtragem Baseada em Conteúdo

Na filtragem baseada em conteúdo, os itens são representados por vetores de características, e o sistema recomenda novos itens semelhantes àqueles que o usuário já demonstrou interesse [6]. O perfil do usuário é construído a partir dos itens selecionados ou avaliados positivamente, e a recomendação é formulada com base na similaridade entre esse perfil e os itens restantes do catálogo.

Um componente fundamental dessa abordagem é a representação dos itens. Para atributos textuais, é comum o uso de TF-IDF (*Term Frequency–Inverse Document Frequency*), que atribui pesos maiores a termos frequentes em um documento, mas raros na coleção como um todo [6]. Para atributos categóricos, técnicas como *One-Hot Encoding* e *Multi-Label Binarization* produzem vetores binários indicando a presença ou ausência de gêneros, categorias ou desenvolvedores. A combinação dessas representações resulta em vetores numéricos capazes de capturar múltiplas dimensões de cada item.

Apesar de simples e interpretável, a filtragem baseada em conteúdo pode apresentar limitações como *overspecialization*, quando o sistema passa a recomendar itens muito semelhantes entre si, reduzindo diversidade de descoberta [6].

C. Similaridade de Cosseno

A similaridade de cosseno é amplamente utilizada em filtragem baseada em conteúdo por ser robusta em espaços vetoriais de alta dimensionalidade, especialmente em representações esparsas derivadas de TF-IDF [6]. Dado dois vetores de características $\vec{A}$ e $\vec{B}$, a similaridade de cosseno é definida como:

$$\text{similaridade}(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|}$$

O valor resultante está entre 0 e 1, onde 1 indica máxima similaridade e 0 indica ausência de alinhamento direcional entre os vetores.

D. Filtragem Colaborativa

A filtragem colaborativa (*Collaborative Filtering*) utiliza o histórico de interações de um conjunto de usuários para identificar padrões de preferência compartilhados [8]. Em abordagens *user-based*, buscam-se usuários com gostos similares; em abordagens *item-based*, identificam-se itens frequentemente consumidos em conjunto.

Embora apresente bom desempenho em plataformas com grande volume de usuários e avaliações, a filtragem colaborativa enfrenta desafios típicos, como esparsidade de avaliações e *cold start* (falta de dados para novos usuários/itens) [8].

E. Sistemas Híbridos

Sistemas híbridos combinam filtragem baseada em conteúdo e filtragem colaborativa para mitigar limitações de cada abordagem [9]. Essa combinação pode aumentar robustez em cenários de poucos dados e, simultaneamente, aproveitar padrões coletivos quando interações de múltiplos usuários estão disponíveis.

F. Objetivos e Implicações de Sistemas de Recomendação

Além de precisão, sistemas de recomendação podem ser projetados para otimizar objetivos como diversidade e novidade, evitando recomendações repetitivas e reduzindo o risco de reforçar preferências de forma estreita [1], [2]. Em plataformas digitais, também se discutem implicações sociais e comportamentais relacionadas à influência algorítmica e à formação de "bolhas de filtro" [3], [5]. Esses efeitos são relevantes quando o desenho do sistema prioriza retenção de atenção em detrimento de autonomia e exploração, podendo contribuir para padrões de consumo prolongado [4].

G. Representação de Itens e Matriz de Características

Em sistemas de recomendação baseados em conteúdo, a qualidade da recomendação depende diretamente de como cada item é representado numericamente. Como músicas, filmes e jogos possuem descrições heterogêneas, é comum adotar uma representação multiatributo, combinando campos textuais, categóricos e numéricos em um mesmo vetor de características [2], [6]. Em termos práticos, essa representação forma uma matriz $\mathbf{X} \in \mathbb{R}^{n \times d}$, em que cada linha corresponde a um item do catálogo e cada coluna representa uma *feature* (atributo) utilizada para comparação.

Campos textuais (por exemplo, sinopses, descrições curtas, palavras-chave) normalmente são convertidos em vetores usando TF-IDF, resultando em matrizes esparsas e de alta dimensionalidade [6]. Já campos categóricos (gêneros, categorias, desenvolvedores, artistas) podem ser representados por codificações binárias como *One-Hot Encoding* ou *Multi-Label Binarization*. Por fim, atributos numéricos (popularidade, duração, notas, *BPM* e atributos de áudio) devem ser normalizados para evitar que escalas maiores dominem o cálculo de similaridade [2]. A concatenação dessas partes gera um vetor final que busca capturar diferentes aspectos do item e permitir comparação consistente entre itens do mesmo domínio.

H. Ponderação por Domínio e Importância Relativa de Atributos

Embora a concatenação de múltiplas fontes de informação aumente a riqueza descritiva dos itens, nem todos os atributos contribuem da mesma forma para a percepção de similaridade. Por exemplo, em músicas, gênero e atributos de áudio tendem a ser muito informativos; em filmes, elenco, direção e palavras-chave podem ser determinantes; em jogos, categorias e *tags* frequentemente representam melhor o tipo de experiência oferecida [1], [2]. Assim, é comum atribuir pesos diferentes

para grupos de *features*, refletindo sua importância relativa no domínio.

Formalmente, a ponderação pode ser vista como uma transformação em que cada dimensão do vetor recebe um fator w_j, resultando em um vetor ponderado $\vec{x}' = (w_1x_1, w_2x_2, \ldots, w_dx_d)$. Essa estratégia permite calibrar o impacto de certos atributos no cálculo de similaridade, reduzindo ruído e aumentando a aderência semântica das recomendações. No entanto, a definição desses pesos é um ponto sensível: pesos mal ajustados podem induzir recomendações enviesadas, enfatizando excessivamente um único tipo de atributo e reduzindo diversidade [6].

I. Limitações Clássicas em Recomendação Baseada em Conteúdo

Apesar de interpretável e aplicável mesmo com poucos usuários, a filtragem baseada em conteúdo possui limitações recorrentes na literatura. A primeira é o *cold start* do item: quando um item novo possui metadados incompletos ou pouco informativos, sua representação vetorial fica pobre e o sistema tende a sub-recomendá-lo [2]. Diferentemente da filtragem colaborativa, que pode se beneficiar de sinais coletivos, a abordagem baseada em conteúdo depende fortemente da qualidade e consistência dos atributos disponíveis.

Outro problema frequente é a *overspecialization*, em que o sistema recomenda itens excessivamente semelhantes aos já selecionados, restringindo a exploração do catálogo e reduzindo novidade [6]. Esse efeito pode ser agravado quando há forte peso em categorias dominantes ou quando o perfil do usuário é representado por uma agregação que privilegia preferências mais recorrentes.

Por fim, destaca-se o viés de popularidade versus nicho. Atributos como "popularidade" ou "nota média" podem introduzir uma tendência de recomendar itens amplamente consumidos, mesmo quando o objetivo do usuário é descobrir conteúdos alternativos [1], [2]. Por isso, sistemas práticos frequentemente equilibram relevância e diversidade por meio de heurísticas e ajustes de ponderação, buscando recomendações que sejam semelhantes, mas não repetitivas.

III. Trabalhos Correlatos

Para contextualizar a presente proposta, foram analisados projetos que exploram diferentes aspectos da recomendação de entretenimento. Cada trabalho contribuiu de forma específica para o desenho da plataforma.

O trabalho de D. L. S. Junior com o Smartbox [16] explora a criação de *playlists* musicais adaptadas a ambientes compartilhados, conciliando preferências de múltiplos ouvintes. Esse sistema motivou o uso de algoritmos de recomendação para apoiar descobertas musicais em contextos coletivos. Diferentemente do Smartbox, que se restringe ao domínio musical, a plataforma aqui proposta abrange três tipos de mídia (músicas, filmes e jogos).

O sistema MOVE! [17] foca em recomendação de músicas para atividades físicas, utilizando atributos como andamento (*BPM*) e intensidade. Esse trabalho reforçou a importância de explorar metadados específicos na modelagem das recomendações. Na plataforma atual, a ideia é generalizada para outros domínios: atributos de áudio são combinados com gênero em músicas, enquanto jogos e filmes utilizam tags, palavras-chave e informações de elenco.

Por fim, o trabalho de Furlan [18] implementa e descreve em detalhes abordagens de filtragem colaborativa e baseada em conteúdo aplicadas a filmes e séries. Sua monografia serviu como referência técnica direta para definição de *pipeline* de vetorização, uso de TF-IDF e aplicação de similaridade de cosseno. O presente projeto retoma essas técnicas, mas as aplica em um cenário multi-domínio, com arquiteturas de serviços independentes para cada tipo de mídia.

IV. Metodologia

A. Abordagem de Desenvolvimento

A metodologia de gerenciamento adotada foi o Kanban [7]. Esse método ágil organiza o trabalho em um quadro de colunas (por exemplo, *Backlog*, *Em Progresso*, *Concluído*), permitindo acompanhar visualmente o fluxo das tarefas. Para um projeto de pesquisa aplicada, que passa por ciclos de experimentação e refinamento, a flexibilidade do Kanban mostrou-se adequada [19].

As principais frentes de trabalho foram divididas em: preparação de dados, implementação dos serviços de recomendação, desenvolvimento do *frontend*, modelagem do banco de dados e documentação. Essa divisão facilitou a priorização gradual das funcionalidades essenciais, garantindo que um protótipo funcional de ponta a ponta estivesse disponível antes da inclusão de refinamentos.

B. Arquitetura do Sistema

O sistema proposto adota arquitetura cliente-servidor. O *frontend* foi desenvolvido em React [13] e consome uma API HTTP unificada. No *backend*, serviços em Python [12] executam a recomendação por domínio (músicas, filmes e jogos) e expõem rotas específicas para consulta.

A comunicação entre o *frontend* e os serviços é intermediada por um *API Gateway*, responsável por expor uma interface consistente para o cliente e encaminhar requisições ao serviço correspondente. Essa decisão simplifica o *frontend*, centraliza configurações e permite, futuramente, adicionar autenticação e *logging* de forma transversal.

C. Modelagem do Banco de Dados

Foi desenvolvido um protótipo de banco de dados relacional utilizando SQL Server Management Studio [10], com o objetivo de preparar a plataforma para funcionalidades futuras de persistência de usuários e histórico de interações, necessárias para evolução em direção à filtragem colaborativa.

A Figura 1 apresenta o modelo relacional proposto. O esquema inclui entidades para usuários, listas de mídias, itens de lista, avaliações e tabelas de detalhes específicos de músicas, filmes e jogos. A tabela *midias* concentra atributos comuns e se relaciona com tabelas de detalhes (por exemplo, *filme_detalhes*, *musica_detalhes* e *jogo_detalhes*), enquanto

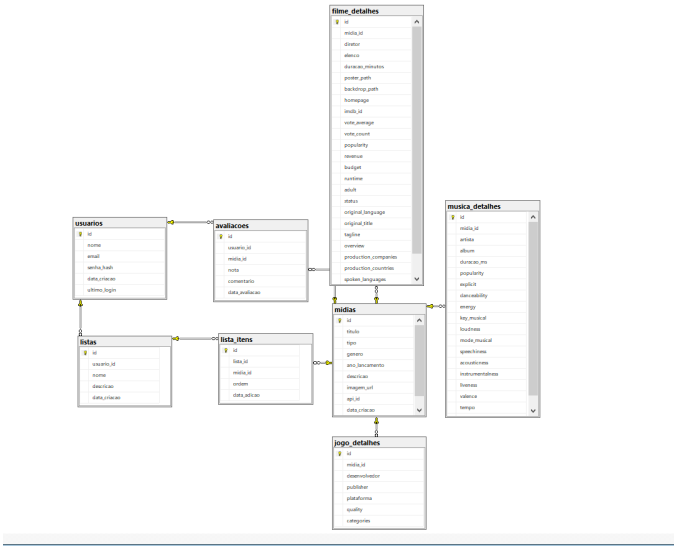

Fig. 1. Protótipo do modelo relacional do sistema.

tabelas como *avaliacoes* e *listas* suportam histórico de preferências.

Na versão atual do protótipo funcional, a recomendação opera a partir de estruturas pré-processadas em disco, mantendo o banco de dados como componente planejado para evoluções futuras.

D. Segurança e Autenticação

Como preparação para persistência de usuários e rotas protegidas, foi prototipado um mecanismo de autenticação baseado em JSON Web Tokens (JWT) [11]. O modelo contempla cadastro, verificação de credenciais e emissão de tokens assinados para autenticação *stateless* em requisições HTTP. No protótipo final avaliado neste trabalho, esse mecanismo permanece desativado para manter o foco na recomendação baseada em conteúdo.

E. Processamento de Dados e Pipeline de Recomendação

Os dados de entrada do sistema são *datasets* públicos em formato CSV [20]–[22]. Para reduzir latência durante execução, foi adotado pré-processamento offline: scripts em Python carregam os CSVs, executam limpeza, seleção de atributos, vetorização e normalização, e gravam os resultados em arquivos otimizados (por exemplo, tabelas *Parquet* e matrizes serializadas).

Durante execução, os serviços carregam diretamente essas estruturas e calculam recomendações a partir de vetores de características. Essa estratégia evita reprocessamento a cada requisição e favorece tempos de resposta adequados.

F. Levantamento de Requisitos

O levantamento de requisitos resultou nos itens funcionais e não funcionais sintetizados nas Tabelas I e II. Os requisitos foram usados como guia para priorização das atividades, com foco em garantir que o usuário pudesse selecionar mídias, receber recomendações em tempo adequado e navegar em uma interface responsiva.

TABLE I
REQUISITOS FUNCIONAIS

ID	Descrição
RF01	Permitir que o usuário selecione músicas, filmes ou jogos de seu interesse.
RF02	Exibir recomendações de mídias baseadas nas seleções do usuário.
RF03	Calcular a similaridade entre itens usando vetores de características.
RF04	Apresentar categorias de recomendações (principais, joias escondidas, clássicos).
RF05	Exibir *scores* de similaridade normalizados para cada recomendação.

TABLE II
REQUISITOS NÃO FUNCIONAIS

ID	Descrição
RNF01	Interface responsiva (desktop e dispositivos móveis).
RNF02	Tempo de resposta para gerar recomendações inferior a 3 segundos.
RNF03	Comunicação cliente-servidor por meio de protocolo HTTPS.
RNF04	Processamento eficiente de *datasets* em formato CSV pré-processados.

V. RESULTADOS

A. Tela de Login

Embora a autenticação JWT não esteja ativa no protótipo final, foi construída uma tela de login que ilustra o fluxo planejado de acesso por usuário. A Figura 2 apresenta essa interface, que poderá ser conectada ao sistema de autenticação quando a persistência de usuários for ativada.

B. Serviços de Recomendação e Pré-processamento

Os três serviços de recomendação (músicas, filmes e jogos) compartilham a mesma filosofia de implementação: cada serviço possui um script de pré-processamento executado de forma offline e um aplicativo que expõe recomendações via HTTP.

1) Sistema de Músicas: No domínio musical, o *dataset* "Spotify Tracks DB" [20] foi utilizado como base. Um script de preparação carrega o CSV, remove duplicatas e seleciona atributos de áudio como `danceability`, `energy`, `valence`, `tempo` e `loudness`. Esses atributos são normalizados e combinados com uma representação *one-hot* de gêneros.

Após o pré-processamento, são gerados um arquivo *Parquet* com os dados limpos e uma matriz de *features* serializada em disco. Durante a execução, o serviço carrega essas estruturas e atende requisições de recomendação sem precisar reprocessar o CSV.

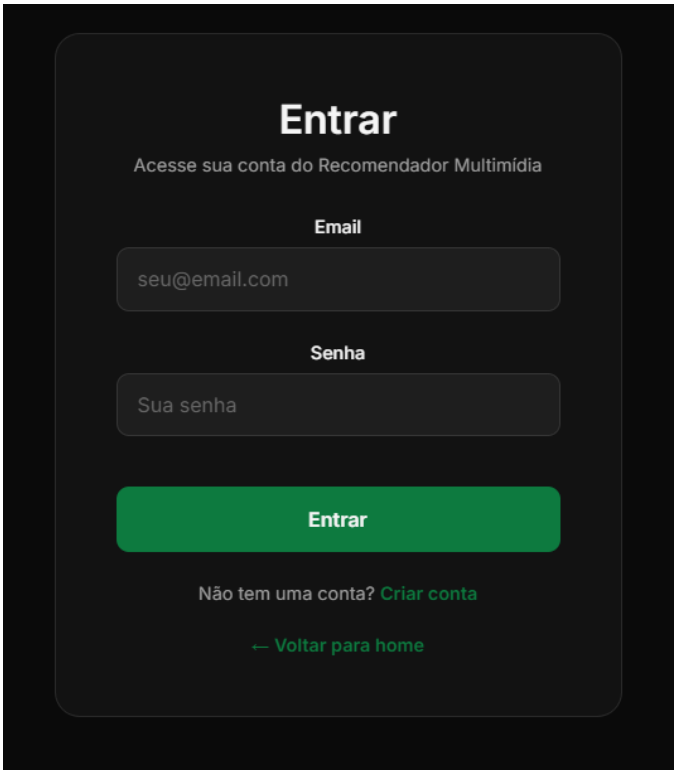

Fig. 2. Tela de login planejada para o sistema.

2) Sistema de Jogos: O sistema de jogos utiliza dados coletados a partir da API SteamSpy e filtrados por número mínimo de avaliações [22]. Para cada jogo, são consideradas *features* como gêneros, categorias, desenvolvedores, qualidade e uma descrição curta. Matrizes de características são construídas para cada grupo e combinadas com pesos, de modo que gêneros e categorias tenham maior impacto no *score* final.

3) Sistema de Filmes: Para filmes, foi adotado o "TMDB 5000 Movie Dataset" [21], que é resultado da fusão de dois arquivos CSV (*movies* e *credits*). O pré-processamento unifica os arquivos, extrai gêneros, palavras-chave e informações de elenco e direção, e aplica TF-IDF às colunas textuais. Um vetor médio é utilizado para representar o "perfil" das seleções do usuário.

C. Evolução do Algoritmo de Recomendação

A primeira versão do algoritmo utilizava a média dos vetores das mídias selecionadas para compor um único vetor de perfil do usuário. Durante os testes, observou-se que essa estratégia diluía gostos minoritários: itens de nicho acabavam ofuscados por preferências mais frequentes.

Para contornar esse problema, o algoritmo foi reformulado para uma abordagem "item-a-item". Em vez de um único vetor médio, cada item selecionado contribui com sua própria similaridade em relação aos candidatos. Para cada candidato C e conjunto de itens selecionados S, o *score* é calculado por:

$$\text{Score}(C) = \sum_{s \in S} \text{similaridade\_cosseno}(C, s)$$

Essa estratégia preserva a influência de itens de nicho, pois candidatos muito semelhantes a um único item selecionado ainda podem alcançar uma pontuação elevada.

D. Implementação em Código

Para fins de documentação, trechos ilustrativos da implementação do algoritmo são representados nas Figuras 3, 4 e 5.

```
    # Combinar todas as recomendações
    if all_recommendations:
        combined_recs = pd.concat(all_recommendations, ignore_index=True)
        combined_recs = combined_recs.sort_values('similarity', ascending=False)
        combined_recs = combined_recs.drop_duplicates(subset=['id'], keep='first')
    else:
        combined_recs = pd.DataFrame()

    # Potencializar similaridade (amplifica diferenças)
    if not combined_recs.empty:
        combined_recs['similarity'] = combined_recs['similarity'] ** 4

    # Mesclar com dados completos
    result = combined_recs.merge(self.df, on='id', how='left')

    # Aplicar penalização de repetição de artistas
    result = self._apply_artist_penalty(result)

    # Selecionar top n_recommendations
    result = result.head(n_recommendations)

    # ===== CORREÇÃO CRÍTICA: ADICIONAR SIMILARITY_SCORE =====
    result = self._calculate_display_scores(result)
    # ========================================================

    print(f"Geradas {len(result)} recomendações")

    return result
```

Fig. 3. Montagem dos vetores de características a partir dos dados pré-processados.

```
def get_recommendations(self, track_ids, genre_to_explore=None, n_recommendations=100):
    """
    Gera recomendações baseadas nas faixas selecionadas
    """
    # Encontrar índices das faixas selecionadas
    selected_indices = []
    for track_id in track_ids:
        idx = self.df[self.df['id'] == track_id].index
        if len(idx) > 0:
            selected_indices.append(idx[0])

    if not selected_indices:
        print("Nenhuma faixa selecionada encontrada no dataset")
        return pd.DataFrame()

    print(f"Processando {len(selected_indices)} faixas selecionadas...")

    # Para cada faixa selecionada, encontrar similares
    all_recommendations = []

    for idx in selected_indices:
        # Obter vetor de features da faixa
        track_vector = self.feature_matrix[idx:idx+1]

        # Converter matriz esparsa para array denso
        if hasattr(track_vector, 'toarray'):
            track_vector = track_vector.toarray()
        else:
            track_vector = np.asarray(track_vector)

        # Converter feature_matrix também se necessário
        if hasattr(self.feature_matrix, 'toarray'):
            feature_matrix_dense = self.feature_matrix.toarray()
        else:
            feature_matrix_dense = self.feature_matrix
```

Fig. 4. Cálculo da similaridade de cosseno entre itens selecionados e catálogo.

Nos serviços implementados, após o cálculo de similaridade são aplicadas heurísticas adicionais: potenciação da similaridade para enfatizar diferenças, penalização de artistas ou desenvolvedores repetidos e normalização dos *scores* em uma escala percentual entre 0 e 100, facilitando a interpretação pelo usuário.

```python
# Calcular similaridade com todas as outras faixas
similarities = cosine_similarity(track_vector, feature_matrix_dense).flatten()

# Criar DataFrame com resultados
similar_df = pd.DataFrame({
    'id': self.df['id'].values,
    'similarity': similarities
})

# Remover a própria faixa e faixas já selecionadas
similar_df = similar_df[~similar_df['id'].isin(track_ids)]

# Selecionar top 20 mais similares
top_similar = similar_df.nlargest(20, 'similarity')

all_recommendations.append(top_similar)
```

Fig. 5. *Ranking* de recomendações, penalização e normalização de *scores*.

E. Interfaces de Usuário

A navegação do sistema inicia pela tela de login (Figura 2). Após a autenticação ou entrada no protótipo, o usuário é direcionado para a página inicial, ilustrada na Figura 6, que destaca o domínio de músicas como ponto de partida para exploração das mídias.

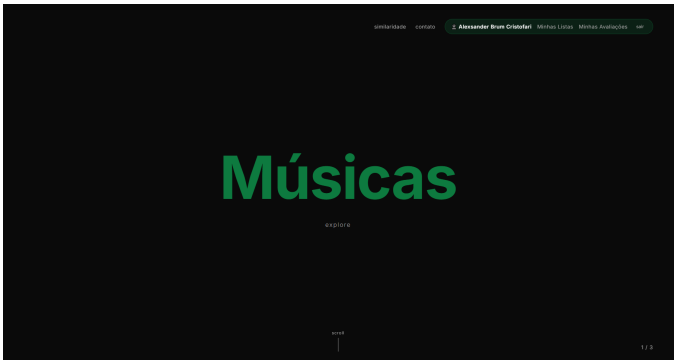

Fig. 6. Página inicial da plataforma com destaque para o domínio de músicas.

Em seguida, o *frontend* é organizado em páginas independentes para músicas, filmes e jogos, com componentes compartilhados para cabeçalho, cartões de mídia e listagem de recomendações.

1) Sistema de Recomendação de Músicas: O subsistema de músicas possui uma tela de seleção (Figura 7) e uma tela de resultados (Figura 8). Na tela de seleção, o usuário pode buscar faixas e adicioná-las a uma lista de preferências. Na tela de resultados, as recomendações são exibidas em listas ordenadas com seus respectivos *scores*.

2) Sistema de Recomendação de Jogos: De forma semelhante, o subsistema de jogos apresenta telas de seleção (Figura 9) e resultados (Figura 10), nas quais são exibidos títulos, gêneros, categorias e qualidade dos jogos recomendados.

3) Sistema de Recomendação de Filmes: O subsistema de filmes enfatiza informações como pôster, sinopse e nota média. As Figuras 11 e 12 ilustram as telas de seleção e resultados, respectivamente.

F. Validação e Testes

Foram realizados testes de unidade para verificar a consistência das transformações de dados e dos cálculos de

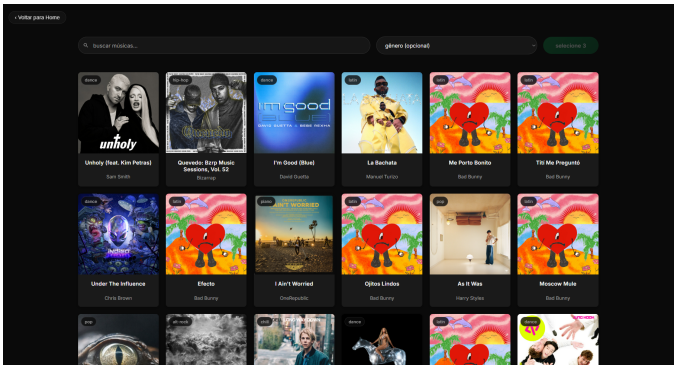

Fig. 7. Interface de seleção do sistema de músicas.

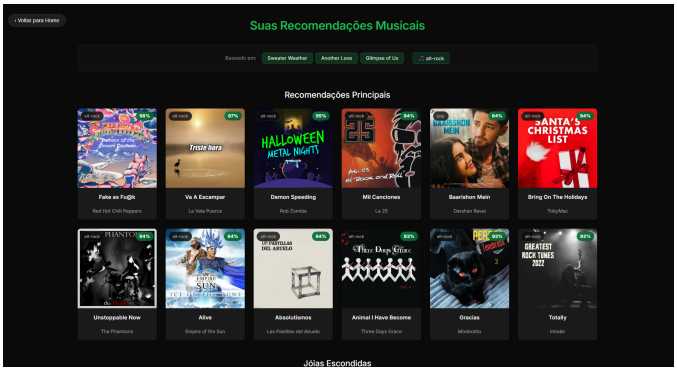

Fig. 8. Interface de resultados do sistema de músicas.

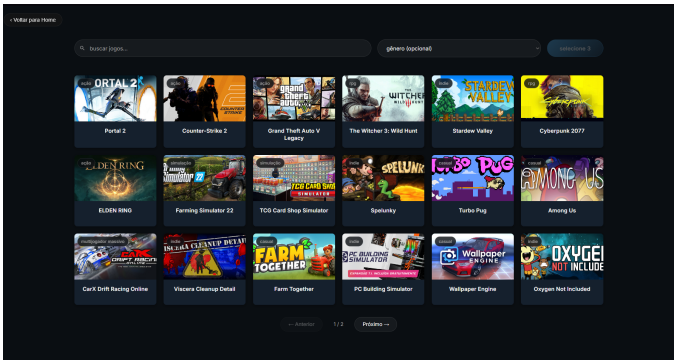

Fig. 9. Interface de seleção do sistema de jogos.

similaridade, bem como testes de integração envolvendo o *API Gateway*, os serviços de recomendação e o *frontend*. Em testes práticos, o sistema foi capaz de responder com listas de recomendações em menos de 2 segundos para *datasets* contendo até 170 000 músicas, 5 000 filmes e 800 jogos, atendendo ao requisito RNF02.

VI. Conclusão

Este trabalho apresentou o desenvolvimento de uma plataforma digital para recomendação de músicas, filmes e jogos, integrando conceitos de sistemas de recomendação, processamento de dados e desenvolvimento web. A solução adotou filtragem baseada em conteúdo, representando itens por

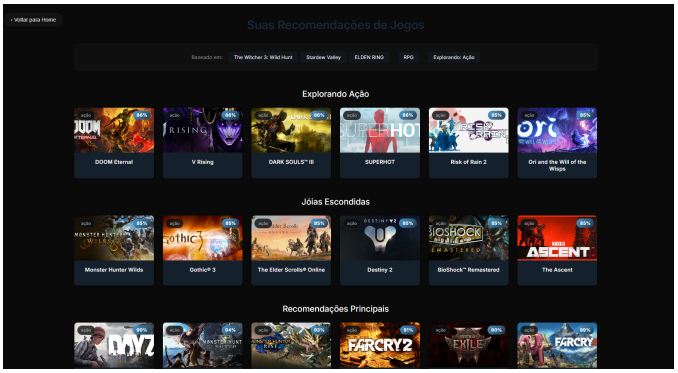

Fig. 10. Interface de resultados do sistema de jogos.

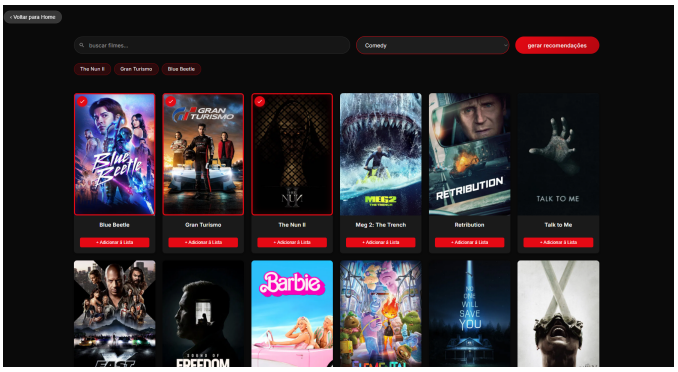

Fig. 11. Interface de seleção do sistema de filmes.

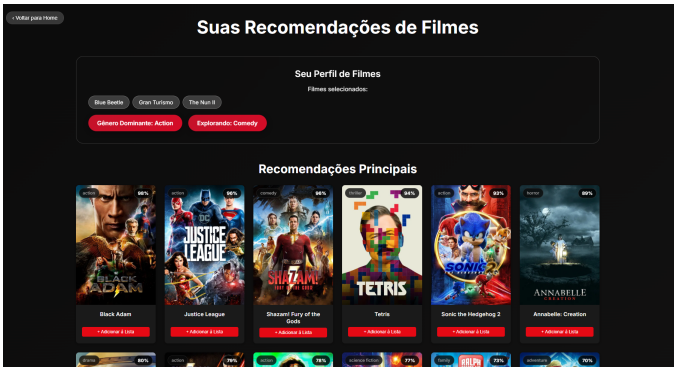

Fig. 12. Interface de resultados do sistema de filmes.

vetores de características e utilizando similaridade de cosseno para medir proximidade entre mídias.

Do ponto de vista acadêmico, o projeto possibilitou a aplicação prática de técnicas de vetorização (TF-IDF, *One-Hot Encoding*), normalização e cálculo de similaridade em *datasets* reais. Também proporcionou experiência na construção de uma arquitetura modular com serviços por domínio, no uso de um *API Gateway* e na criação de uma interface reativa em React.

Algumas limitações foram identificadas. A ausência de filtragem colaborativa impede que o sistema explore correlações entre múltiplos usuários, e a carga completa das matrizes em memória pode se tornar um gargalo em cenários de escala maior. Além disso, o protótipo de banco de dados e o mecanismo de autenticação JWT permanecem desativados na versão atual.

Como trabalhos futuros, destacam-se: (i) ativar persistência de usuários e implementar filtragem colaborativa, combinando-a com a abordagem baseada em conteúdo; (ii) empregar estruturas de índice vetorial para acelerar consultas em catálogos maiores; (iii) integrar o sistema a APIs oficiais de provedores de mídia, permitindo atualização automática de catálogos; e (iv) adicionar mecanismos de *feedback* explícito dos usuários, para que o algoritmo possa ajustar seus parâmetros ao longo do tempo.

Apesar dessas limitações, o protótipo cumpre o objetivo de demonstrar, de forma clara e funcional, como técnicas de filtragem baseada em conteúdo podem ser utilizadas para construir um sistema de recomendação multi-domínio.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*, 3rd ed. New York, NY, USA: Springer, 2021.

[2] C. C. Aggarwal, *Recommender Systems: The Textbook*. Cham: Springer International Publishing, 2016.

[3] D. Boyd and K. Crawford, "Critical questions for big data," *Information, Communication & Society*, vol. 15, no. 5, pp. 662–679, 2012.

[4] G. T. Zewude et al., "Digital Addiction and Its Impact on the Mental Wellbeing of Adolescents," *ERIC*, Tech. Rep. EJ1455986, 2024. [Online]. Available: https://files.eric.ed.gov/fulltext/EJ1455986.pdf

[5] E. Pariser, *The Filter Bubble: What the Internet Is Hiding from You*. London, UK: Penguin, 2011.

[6] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and new challenges," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 4, pp. 1–41, 2011.

[7] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[8] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer, 2007, pp. 291–324.

[9] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[10] Microsoft Corporation, *SQL Server Documentation*, 2025. [Online]. Available: https://docs.microsoft.com/en-us/sql/

[11] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, May 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

[12] Python Software Foundation, "Python Language Reference," 2025. [Online]. Available: https://www.python.org/

[13] Meta and community, *React Documentation*, 2025. [Online]. Available: https://react.dev/

[14] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Apress, 2014.

[15] GitHub, Inc., "GitHub," 2025. [Online]. Available: https://github.com/

[16] D. L. S. Junior, "Smartbox - Sistema de Recomendação Musical para Ambientes Compartilhados," Monografia, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2018.

[17] D. L. da Silva Junior, F. L. C. de Araújo, and G. M. B. Barroso, "MOVE! Um Sistema de Recomendação de Músicas para uma Atividade Física," in *Anais do XIX Encontro Regional de Software e Sistemas do Rio de Janeiro (ERSI-RJ 2021)*, 2021.

[18] L. Furlan, "Desenvolvimento de um Sistema de Recomendação Híbrido para Filmes e Séries," Trabalho de Conclusão de Curso, Universidade Franciscana, Santa Maria, 2018. [Online]. Available: https://tfgonline.lapinf.ufn.edu.br/media/midias/Leonardo_Furlan.pdf

[19] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. McGraw-Hill Education, 2019.

[20] Y. M. Çano, "Spotify Tracks DB," Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/ymcanb/spotify-tracks-db

[21] R. Thota, “TMDB 5000 Movie Dataset,” Kaggle, 2017. [Online]. Available: https://www.kaggle.com/datasets/tmdb/tmdb-5000-movie-dataset
[22] N. Paiva, “Steam Store Games,” Kaggle, 2019. [Online]. Available: https://www.kaggle.com/datasets/nikdavis/steam-store-games