UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

DANIEL SOUZA AVELLAR

Uma proposta de um Sistema de Recomendação como Serviço

Prof. Filipe Braida do Carmo, D.Sc.
Orientador

Nova Iguaçu, Abril de 2022

Uma proposta de um Sistema de Recomendação como Serviço

Daniel Souza Avellar

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Daniel Souza Avellar

Aprovado por:

Prof. Filipe Braida do Carmo, D.Sc.

Prof. Leandro Guimarães Marques Alvim, D.Sc.

Prof. Marcel William Rocha da Silva, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Abril de 2022

Agradecimentos

Primeiramente, quero agradecer a minha família, principalmente meus pais, Aureo e Patricia, por todo o incentivo e apoio incondicional. Creio que só estou aqui por consequência disso. Sou grato a minha irmã, Raquel, que foi o meu pontapé inicial nos estudos: por ser mais velha e já frequentar a escola, eu comecei a querer aprender me inspirando nela, tanto que já entrei na escola sabendo ler. Agradeço também aos meus tios, Marcos e Vanderleia, que sempre estiveram acompanhando essa caminhada bem de perto.

Gostaria de agradecer a todos os professores que fizeram parte dessa jornada até aqui, compartilhando não só o conteúdo das matérias ministradas, mas contribuindo para a minha formação pessoal. Creio que no Daniel de hoje tem um pouquinho de cada um. No entanto, quero fazer um agradecimento especial ao meu orientador Filipe Braida, que admiro, respeito e que não desistiu de mim, mesmo eu tendo enrolado essa monografia por muito tempo.

Também sou muito grato aos amigos que a Rural me deu, que pretendo levar para toda a vida. Angelito, que não esqueço das nossas voltas pra casa escutando aquele CD ao vivo do Forfun. Gabi, minha primeira dupla de trabalho na Cesgranrio, com quem aprendi muito. Gabigol, que já me ajudou muito na faculdade pois é a única pessoa que achou ela fácil. Léo, meu companheiro de maracanã e conterrâneo da Faetec, assim como o Vaqueiro, já que nosso péssimo humor e gosto musical é o que nos une. Vini, meu vizinho e companheiro de degustação de hambúrgueres e pizza e o Thales, que foi minha primeira amizade na faculdade, já fomos até sócios, mas nunca vou esquecer do nosso primeiro de muitos sofrimentos juntos, fazendo o jogo de COMP1. Enfim, cada um de vocês é corresponsável por eu estar onde estou.

Obrigado!

Agradeço, especialmente, a minha companheira dessa vida, Clarissa. Que acompanhou toda essa jornada de perto, sofrendo junto comigo, me incentivando e me motivando a nunca desistir. Seu apoio foi fundamental para a conclusão deste trabalho e eu sou eternamente grato por isso.

A todos que direta ou indiretamente contribuíram para este momento, deixo o meu muito obrigado.

RESUMO

Uma proposta de um Sistema de
Recomendação como Serviço

Daniel Souza Avellar

Abril/2022

Orientador: Filipe Braida do Carmo, D.Sc.

Com o grande volume de dados disponível atualmente, simples atos como o de escolher algum produto para comprar, um filme para assistir, uma música para ouvir, entre outros, se tornam mais complicados. Os Sistemas de Recomendação surgiram para auxiliar nessa tomada de decisão, ao ofertarem itens de maior relevância para o usuário. Com isso, muitas empresas utilizam a recomendação como uma forma de gerar valor, seja vendendo mais produtos ou mantendo o usuário mais ativo, por exemplo. No entanto, o custo envolvido na entrega de uma recomendação é muito alto e vai muito além de um algoritmo, o que gera um cenário em que apenas grandes empresas conseguem investir nesse tipo de tecnologia. Já o modelo de Software como Serviço, tem como objetivo distribuir software, com um consumo sob demanda, no qual tira o peso de implementação da solução do lado consumidor, deixando ele focar em seu negócio. Tendo isso em vista, este trabalho apresenta uma proposta de Sistema de Recomendação como Serviço, que tem o objetivo de entregar recomendações nas quais a integração funcione da maneira mais simples possível, e atenda aos mais diversos clientes. Com isso, será expandida a gama de plataformas que conseguirão se aproveitar das vantagens de um Sistema de Recomendação, sem que elas tenham a necessidade de arcar com todos os custos da implementação.

ABSTRACT

Uma proposta de um Sistema de

Recomendação como Serviço

Daniel Souza Avellar

Abril/2022

Advisor: Filipe Braida do Carmo, D.Sc.

With the large volume of data currently available, simple acts such as choosing a product to buy, a movie to watch, a song to listen to, among others, become more complicated. Recommender Systems emerged to assist in this decision making, by offering items of greater relevance to the user. With this, many companies use the recommendation as a way to generate value, either by selling more products or keeping the user more active, for example. However, the cost involved in delivering a recommendation is very high and goes far beyond an algorithm, which creates a scenario in which only large companies are able to invest in this type of technology. The Software as a Service model, on the other hand, aims to distribute software, with on-demand consumption, in which it takes the burden of implementing the solution from the consumer side, leaving it to focus on its business. With this in mind, this work presents a proposal for a Recommendation System as a Service, which aims to deliver recommendations in which the integration works in the simplest possible way, and serves the most diverse customers. With this, the range of platforms that will be able to take advantage of a Recommender System will be expanded, without having to bear all the costs of implementation.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

SR	Sistemas de Recomendação
SaaS	*Software as a Service*
PaaS	*Plataform as a Service*
IaaS	*Infrastructure as a Service*
IAaaS	*Artificial intelligence as a Service*
ORM	*Object Relational Mapper*
vscode	*Visual Studio Code*
IDE	Ambiente de Desenvolvimento Integrado
API	*Application Programming Interface*

Sumário

Capítulo 1

Introdução

Em nosso cotidiano, nos deparamos com recomendações de diversas maneiras e fontes: desde um amigo recomendando um novo restaurante de comida japonesa, até um colega de trabalho te indicando um atalho para fugir do engarrafamento na volta para casa. Com o surgimento da internet, o ato de recomendar, que já era tão natural, passou a existir também no mundo digital. Em um cenário em que as pessoas têm acesso a uma imensa quantidade de produtos com um simples clique, o consumo e a comparação de preços ficou mais fácil, ao mesmo tempo em que o despertar de interesse ficou mais difícil, já que há muitas ofertas e possibilidades. Na perspectiva das empresas, cresceu muito o acesso e o armazenamento de dados dos clientes, contudo, a concorrência para chegar primeiro a esse consumidor também é muito maior. Assim, para suprir as necessidades de ambas as partes, surgiram os Sistemas de Recomendação.(MELVILLE; SINDHWANI, 2017)

Nesse sentido, os Sistemas de Recomendação são ferramentas de software e técnicas que fornecem sugestões de itens que podem ser usados pelo usuário e que estão relacionadas a vários processos de tomada de decisão (RICCI; ROKACH; SHAPIRA, 2011). Os Sistemas de Recomendação obtiveram uma grande relevância no mundo digital, sendo o *e-commerce* sua principal área de atuação, em que a recomendação é utilizada para aumentar as vendas. Contudo, também é possível encontrar recomendações em redes sociais, catálogos de filmes, aplicativos de *streaming*, entre outros, sempre com o objetivo de facilitar o acesso aos produtos ou serviços de interesse,

aumentando, assim, o consumo. (KOWALCZYK; SZLÁVIK; SCHUT, 2012)

Um outro aspecto desse cenário é a existência de uma quantidade massiva de dados. Já em 2003, havia uma estimativa de que a humanidade havia acumulado, aproximadamente, 12 exabytes de dados (equivalentes a 600.000 anos corridos de vídeo em qualidade DVD). Além disso, entre 2006 e 2010 esse número subiu de 166 exabytes para 988 exabytes (FLORIDI, 2010). Esse crescimento exponencial fez com que, em 2018, chegássemos na casa dos 33 zettabytes, com projeções para alcançar 175 zettabytes em 2025 (COUGHLIN, 2018).

Nesse contexto, os Sistemas de Recomendação se aproveitam dessa massiva quantidade de dados, pois quanto mais dados, maiores são as possibilidades e a assertividade em suas recomendações, aumentando cada vez mais sua relevância, pois quanto maior for esse volume, mais completa e profunda poderá ser a análise (KOWALCZYK; SZLÁVIK; SCHUT, 2012). Vale ressaltar que o simples aumento da quantidade de dados, por si só, não é garantia de melhores resultados, pois é preciso ainda uma modelagem e um tratamento adequados para eles, além da escolha do melhor método de ranqueamento ou previsão que atenda suas necessidades. Só assim, será possível tirar o melhor proveito desse grande volume de dados disponível atualmente. (ZHOU; KHEMMARAT; GAO, 2010)

Logo, a implementação de um Sistema de Recomendação se torna muito custosa, porque grande parte das empresas não tem um negócio compatível para desenvolvê-lo. Para isso, seria necessário obter profissionais da área, o que custa caro, aguardar o tempo necessário para o desenvolvimento, que pode ser longo, e, com isso, o tempo para o retorno também seria maior, entre outras dificuldades.

A partir dessa problemática, implementar um serviço de recomendação é uma solução interessante e vem se tornando uma ferramenta cada vez mais popular. Nesse formato, um provedor entrega um aplicativo em um único conjunto de dados de código comum, em um modelo “um para muitos”, que é utilizado por todos os consumidores contratados, a qualquer momento. Normalmente, esses softwares são pagos por demanda de uso. O crescimento da popularidade se deve às facilidades que os SaaS geram, sendo a principal delas o foco no negócio, uma vez que o contratante

não precisa se preocupar com diversos fatores relacionados à solução, já que os softwares como serviço abstraem essa parte. A redução de custos de desenvolvimento e infraestrutura, a segurança, a garantia de disponibilidade e a escalabilidade são algumas outras vantagens oferecidas. (SEETHAMRAJU, 2013)

Tendo isso em vista, um Sistema de Recomendação como Serviço tem o objetivo de facilitar o acesso à recomendação, uma vez que une todas as vantagens de um Sistema de Recomendação em si, com os benefícios dos softwares como serviço (SaaS) — aumentando o acesso a uma ferramenta tão importante como a recomendação. Por isto, este trabalho traz uma proposta de Sistema de Recomendação como serviço que sirva como uma solução plausível, e que possa ser utilizada e aprimorada, trazendo a facilidade para que qualquer um possa utilizá-lo e tirar proveito de todos os benefícios de um Sistema de Recomendação.

1.1 Objetivo

Os objetivos desse trabalho são:

- Fazer uma proposta de arquitetura por serviço que simplifique a integração com um Sistema de Recomendação.
- Implementar essa arquitetura de forma que seja uma solução viável e de simples utilização.
- Analisar os pontos fracos, fortes e as melhorias da solução.

1.2 Organização do Trabalho

Essa monografia está dividia da seguinte maneira:

O capítulo 2 contém a base teórica, na qual abordamos os Sistemas de Recomendação e Software como Serviço (SaaS), que são a base deste trabalho.

No capítulo 3, encontra-se a proposta de solução para a criação de um Sistema

de Recomendação como Serviço, em que serão apresentados os requisitos e a nossa modelagem.

O capítulo 4 apresentará a solução desenvolvida como resultado de nossa proposta.

Por fim, no capítulo 5 serão apresentadas a conclusão e as referências bibliográficas deste trabalho.

Capítulo 2

Fundamentação Teórica

Para melhor entendimento do trabalho realizado, esse capítulo tem como objetivo se aprofundar em assuntos chaves para a proposta de Sistemas de Recomendação como serviço. A primeira é o próprio Sistema de Recomendação, com a sua motivação, seus métodos e vantagens. Em seguida, falaremos sobre Software como Serviço (SaaS), suas características, principais vantagens e desvantagens, sendo esses os principais temas abordados neste trabalho.

2.1 Sistemas de Recomendação

Novas tecnologias e o surgimento da Web transformaram os modos de consumo conhecidos até então. Na perspectiva dos consumidores, essas transformações vieram acompanhadas de uma facilitação dos pagamentos que, por sua vez, gerou um crescimento na quantidade, na variedade e no consumo dos produtos disponíveis para compra. Por outro lado, isso aumentou a concorrência entre os vendedores, que enfrentam os desafios de pensar em estratégias para chegar primeiro ao cliente. Em paralelo a isso, estão cada vez mais frequentes as transações a partir de dados dos clientes, que permitem uma análise cada vez mais profunda das interações destes com as ofertas de produtos. Nesse sentido, os Sistemas de Recomendação evoluíram para atender as necessidades tanto dos clientes, oferecendo acesso a recomendações especializadas, quanto dos vendedores, chegar de forma mais fácil a seu público alvo

com mais assertividade. (MELVILLE; SINDHWANI, 2017)

Os Sistemas de Recomendação são ferramentas de software que fornecem sugestões significativas de itens, *i.e.* produtos ou serviços, que podem ser usados ou consumidos pelos usuários. Essas sugestões estão relacionadas a vários tipos de tomadas de decisão, desde escolher o que comprar até a próxima música que irá ouvir (RICCI; ROKACH; SHAPIRA, 2011). Empresas, como a Netflix[1] - um Sistema de Recomendação de filmes e séries, usam a forma mais comum de avaliação, que é o "gostei/não gostei", mas também buscam informações mais específicas de cada usuário, como a localização e as características dos itens consumidos (MELVILLE; SINDHWANI, 2017).

O problema de recomendação foi definido mais formalmente por Adomavicius e Tuzhilin (2005). Seja U o conjunto de todos os usuários e I o conjunto de todos os itens que podem ser recomendados, como um livros, filmes e etc. Seja f uma função que mede a utilidade do item I para o usuário U, ou seja, $f\colon U \times I \rightarrow P$ onde $P \in R$. Então, para cada usuário $u \in U$ queremos escolher o item $i' \in I$ que maximiza a função de utilidade do usuário. Mais formalmente:

$$\forall u \in U,\ i'_u = \arg\max_{i \in I} f(u, i). \tag{2.1}$$

Cada elemento no espaço U pode ser definido como um perfil que inclua características do usuário, como data de nascimento, gênero, nacionalidade, entre outras. No caso mais simples, o usuário pode conter apenas seu identificador. Da mesma forma, cada elemento do conjunto de I é definido por um conjunto de características. Em um aplicativo de recomendação de jogos, como por exemplo, onde I é uma coleção de jogos, cada jogo pode ser representado, além de seu identificador único, por seu ano de lançamento, gênero, plataforma, etc.

A função utilidade de um item em Sistemas de Recomendação normalmente é representada por uma avaliação, que indica o gosto de um usuário em relação a um determinado item, por exemplo: Daniel deu a nota 9 de 10 ao jogo Hades. No entanto, a função utilidade pode ser definida por uma heurística, como um exemplo

[1] https://www.netflix.com/br/

simples, podemos fazer recomendações a partir de perfis semelhantes, ou ela pode ser estimada e otimizada através de um critério de desempenho, como por exemplo o erro médio quadrático.

O principal problema dos Sistemas de Recomendação é o fato de a utilidade f, normalmente, não é definida para todo o espaço $U \times I$, mas apenas para um subconjunto dele. Sendo assim, precisa ser realizada a extrapolação para todo o espaço $U \times I$.

Por exemplo, em um aplicativo de recomendação de jogos, os usuários inicialmente avaliam alguns subconjuntos de jogos. Um exemplo de uma matriz de classificação de item de usuário para um aplicativo de recomendação de jogos é apresentado na tabela 2.1, onde as classificações são especificadas na escala de 1 a 5. O símbolo $\emptyset$ representa que o usuário não explicitou o seu gosto sobre o item correspondente. Portanto, o Sistema de Recomendação deve ser capaz de prever os gostos dos usuários sobre os jogos que não foram explicitados por estes com o objetivo de gerar recomendações apropriadas com base nessas previsões.

	God of War	The Last of Us: Part 2	Bloodborne
Daniel	5	5	2
João	4	$\emptyset$	5
Felipe	$\emptyset$	5	4
Thiago	1	3	$\emptyset$

Tabela 2.1: Fragmento de uma matriz de notas de um Sistema de Recomendação de jogos.

Com base em como as recomendações são feitas. Em Ricci, Rokach e Shapira (2011), essas abordagens foram divididas em seis tipos:

- **Baseada em conteúdo:** O sistema aprende a recomendar com base nas similaridades dos itens que o usuário gostava no passado.
- **Filtragem colaborativa:** A forma mais simples dessa abordagem recomenda ao usuário ativo com base no histórico de gostos em comum dele com os de outros usuários.

- **Demográfico:** Esse sistema recomenda itens com base no perfil demográfico do usuário.
- **Baseada no conhecimento:** Esses sistemas recomendam com base em um conhecimento de um domínio específico sobre os recursos do item.
- **Baseada na comunidade:** Esse tipo de sistema recomenda com base nos amigos do usuário ativo.
- **Abordagens híbridas:** Esse sistema combina as abordagens citadas acima.

Embora a maioria das técnicas utilizadas nos Sistemas de Recomendação atuais tenham sido desenvolvidas há mais de uma década (ADOMAVICIUS; TUZHILIN, 2005), existem muitos estudos e pesquisas nessa área. Isso se deve a um mundo que é cada vez mais tecnológico, o que propicia que os Sistemas de Recomendação estejam cada vez mais presentes no dia a dia da sociedade (JANNACH et al., 2010). Com isso, ainda hoje, essa área possui diversos desafios e alguns deles foram descritos por Ricci, Rokach e Shapira (2011).

A escalabilidade dos algoritmos com grandes conjuntos de dados do mundo real é um dos desafios pontuados pelos autores. Aplicar as técnicas de recomendação nessa grande quantidade de dados, a partir das dinâmicas interações dos usuários com os itens, e ainda entregar o resultado em tempo satisfatório é muito desafiador. Por isso, a solução atual mais utilizada é fazer a previsão de forma *offline* e a partir de um conjunto de dados relativamente pequeno.

Uma outra preocupação que sempre esta presente em um contexto onde há manipulação de dados de usuários é a de preservação da privacidade, visto que os Sistemas de Recomendação exploram esses dados para gerar recomendações personalizadas. Quanto mais dados forem explorados, maior será a taxa de acerto na previsão e é justamente isso que motiva a coleta da maior quantidade de dados possível. No entanto, essa prática gera um impacto negativo no que diz respeito à privacidade dos usuários e aumenta a responsabilidade sobre a segurança desses dados, o que inclui protegê-los de usuários mal-intencionados.

No processo de construção de uma lista de recomendações, também é desafiador pensar em uma integração das preferências dos usuários de longo e curto prazo. Nesse sentido, os autores explicam que os Sistemas de Recomendação podem ser divididos em duas classes: aqueles que utilizam todos os dados disponíveis para recomendação e traçam um perfil de longo prazo; e os que estão interessados nas preferências dos usuários apenas naquela época específica. No entanto, novas pesquisas sobre abordagens híbridas têm sido realizadas.

Por fim, é cada vez maior a demanda por recomendadores projetados para operar em dispositivos móveis e diferentes contextos de uso. Com o celular sendo a principal forma de acesso à Web, é preciso adaptar os Sistemas de Recomendação a esse meio. Quando se trata de lojas e hotéis, por exemplo, surgiram mais parâmetros para a recomendação, como a localização. No entanto, essa nova demanda também impõe limitações à recomendação, como a conexão, o tamanho de tela e o poder computacional dos dispositivos móveis.

2.2 Software como Serviço

Por muito tempo, os softwares das empresas eram hospedados em sua própria infraestrutura. Esse tipo de solução trazia uma série de problemas e era uma grande barreira para a adoção de novas tecnologias por parte dessas corporações. Para melhorar essa realidade, o modelo de Software como Serviço (SaaS) vem se solidificando como uma ótima opção, na qual a empresa deixa de comprar um software e arcar com todo o custo de implementação e manutenção, e passa a pagar por um serviço pronto para uso. (FRYER, 2019)

O modelo SaaS faz parte de uma tecnologia chamada Computação em Nuvem, que é um ambiente que permite a utilização do software, dos dados e dos recursos de qualquer lugar na Internet (MOHAMMED; ZEEBAREE, 2021). Ele é dividido em três tipos de modelos básicos: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). Segundo Rashid et al. (2018) um IaaS é o fornecimento de computação, armazenamento, rede e outros serviços

primários de computador, de modo que o consumidor possa instalar e gerenciar máquinas virtuais, incluindo sistemas operacionais e aplicações. Um bom exemplo de IaaS seria a Amazon AWS[2]. Já um PaaS fornece um *framework* ou um meio para o desenvolvimento de aplicações, sem a necessidade de instalar ou gerenciar o ambiente de produção. Todos os servidores, o armazenamento e a rede podem ser gerenciados pela empresa ou por um provedor terceirizado, enquanto os desenvolvedores podem manter o gerenciamento dos aplicativos. Um exemplo de PaaS é a Microsoft Azure[3].

O Software como Serviço (SaaS) pode ser definido como um modelo de distribuição de software que funciona, basicamente, como um fornecedor de uma peça de software, aplicação ou serviço. Ele se aproveita da centralização oferecida pela internet e utiliza uma arquitetura de instância única, para vários clientes (SUN et al., 2007). Assim, o fornecedor é responsável pela distribuição, operação e manutenção da infraestrutura. Essas características de um SaaS fazem com que ele seja destinado exclusivamente ao usuário final, a figura 2.1 mostra a comparação entre os modelos citados a cima. (GUANG et al., 2014).

No passado, o modelo *on-premise* era bastante utilizado. Nele, o cliente hospedava tudo em seu próprio servidor e era responsável pela manutenção dessa infraestrutura, implicando em um investimento inicial mais alto. Já no mercado atual, há um domínio das soluções de SaaS: o modelo é responsável por 75% do lucro relativo das empresas que o utilizam, o que significa cerca de 380 bilhões de dólares (ROCHE, 2020). Logo, empresas sem oferta de SaaS agora representam apenas um quarto da receita total do setor.

Não só o SaaS, mas todos os modelos de Computação em Nuvem compartilham de diversas vantagens. Segundo Leavitt (2009) há algumas características importantes, como o consumo sob demanda, no qual o consumidor pode requisitar um serviço ou um recurso conforme precisar, e a elasticidade, que permite ao cliente aumentar ou diminuir seus recursos, se adptando ao seu consumo. Essas funcionalidades são bem diferentes de um modelo *on-premise*, no qual o consumidor precisa alocar recursos durante o ano inteiro, mesmo que seu pico de demanda seja em apenas dois meses

[2]https://aws.amazon.com/pt/
[3]https://azure.microsoft.com/pt-br/

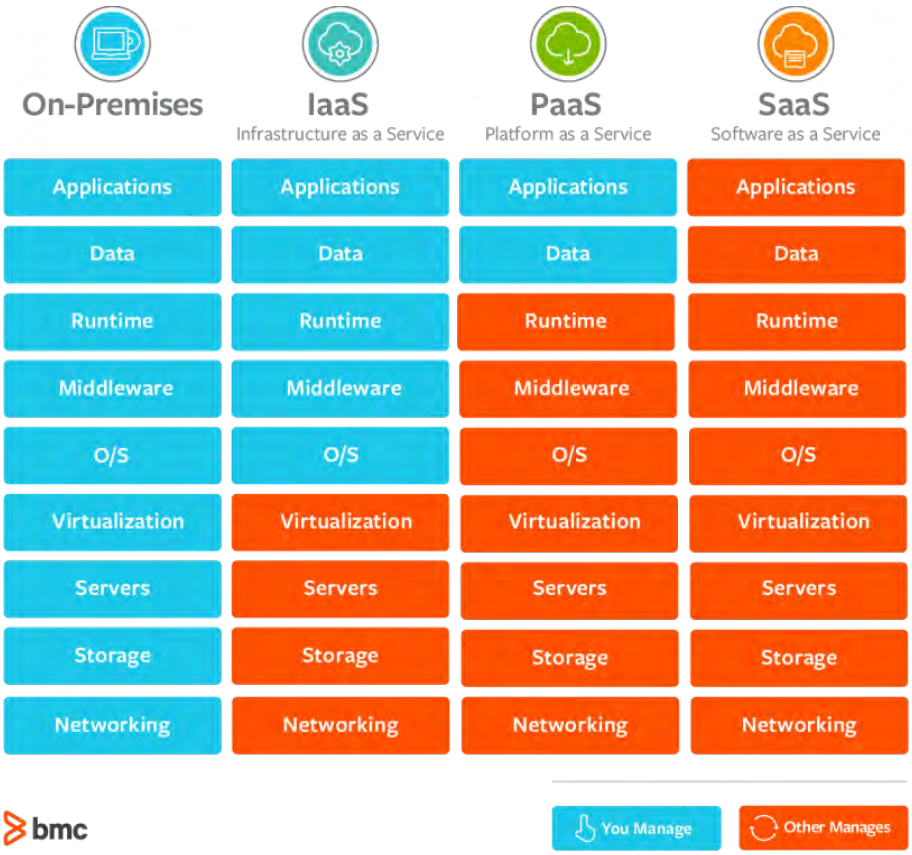

Figura 2.1: Comparação de abrangência dos modelos de computação na nuvem[4].

no ano, o que gera muitos gastos desnecessários. Além disso, as facilidades não são apenas no consumo, já que são oferecidas garantias de serviço. As plataformas garantem a disponibilidade de 99,9% do serviço e, para isso, oferecem diversas ferramentas de monitoramento de recursos, por exemplo. (BUILDER, 2018)

Além da facilidade de implementação do SaaS, suas vantagens também estão atreladas ao trabalho de manter a solução, o que no modelo *on-premise* conta com muitos custos implícitos. No que diz respeito ao hardware, por exemplo, há todo o empenho em manter as condições necessárias e, quando houver algum problema, realizar a manutenção. Quanto ao software, é muito importante evoluir a aplicação e mantê-la atualizada, para que não se torne obsoleta, além da necessidade de pessoas responsáveis pela garantia de disponibilidade e segurança do serviço, em caso de eventuais problemas. Todos esses detalhes são os custos para ter uma aplicação que muitas vezes não são levados em conta.(WATERS, 2005)

[4]Imagem retirada do projeto Worldwideweb no endereço: <https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose>

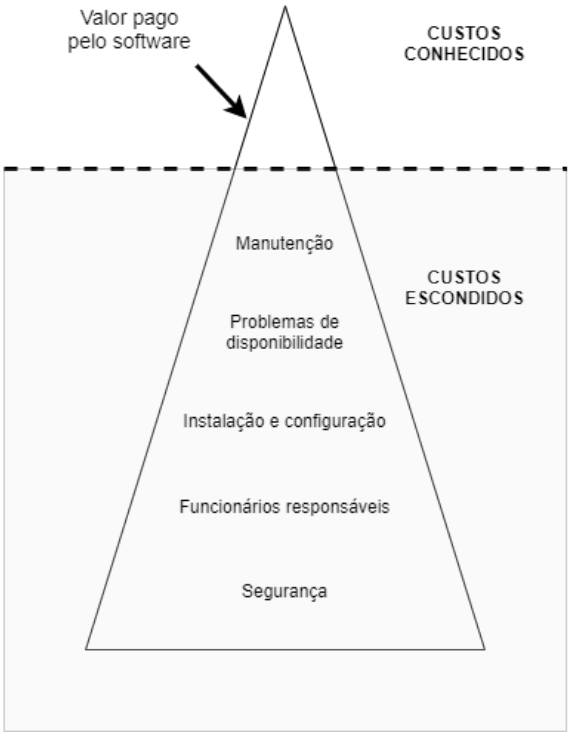

Figura 2.2: Custos escondidos de um software *on-premise*

Além de acabar com o modelo de uso de software *on-premise*, o SaaS é uma ótima alternativa para empresas que até poderiam decidir por elas mesmas desenvolverem sua solução, mas teriam todos os problemas citados anteriormente. Além disso, há também a dificuldade de desenvolver um software de qualidade, ainda mais em empresas que não são especificamente de tecnologia. Com isso, ao adotar esse modelo, muitas corporações conseguem diminuir seus custo iniciais para investirem em tecnologia. (CHOUDHARY, 2007)

Um subgênero do SaaS que vem se popularizando e nossa solução pode se encaixar como, é o IAaaS (Inteligência Artificial como Serviço), que se refere a ferramentas de IA prontas para uso e que permitem que as empresas implementem técnicas de IA por uma fração do custo de uma IA.(KIDD, 2021)

Capítulo 3

Sistema de Recomendação como Serviço

Este capítulo irá apresentar o sistema proposto, sua motivação e a modelagem da solução. A seção 3.1 discutirá a problemática em torno dos Sistemas de Recomendação, na seção 3.2, serão citados trabalhos que tenham algum ponto de semelhança com este, enquanto na seção 3.3 será apresentada a modelagem da solução proposta, com base nos problemas já apresentados.

3.1 Motivação

Grandes empresas como Netflix[1], Amazon[2], Google[3], e outras têm se beneficiado dos Sistemas de Recomendação, seja para aumentar suas vendas, ou para manter seus usuários ativos, ao disponibilizarem, com frequência, produtos passíveis de seu interesse. Essa é uma área que tem recebido muito destaque e motivado investimentos, por aproveitar a grande quantidade de informações disponibilizadas no atual cenário tecnológico e por estar em constante evolução (ADOMAVICIUS; TUZHILIN, 2005).

A Amazon, que é referência no uso de Sistemas de Recomendação, utiliza diferentes

[1]https://www.netflix.com/
[2]https://www.amazon.com/
[3]https://www.google.com/

tipos de dados, além dos mais comuns, como avaliação ou compra de um item, pois também são utilizados dados demográficos ou de itens visualizados, por exemplo. Essa modalidade se chama *feedback* implícito, que é a conversão de comportamento do usuário em opinião, através de observação do seu comportamento (LI, 2019).Vale ressaltar que essa inteligência gerada a partir dos dados dos usuários não é utilizada apenas na recomendação de itens similares, mas há uma personalização da experiência em todo o site. A página inicial, por exemplo, é toda customizada a partir dos dados daquele usuário, a fim de guiá-lo a itens de seu interesse, antes mesmo de ele buscar por algum (LINDEN; SMITH; YORK, 2003).

Contudo, um Sistema de Recomendação vai muito além do algoritmo e, para que ele seja eficiente, são necessários vários itens e etapas importantes. Em um primeiro momento, a análise, o tratamento dos dados e a escolha do melhor algoritmo para recomendar dependem de conhecimento na área e, consequentemente, de profissionais especializados. Além disso, a infraestrutura também é complexa, uma vez que é preciso garantir uma recomendação em um tempo aceitável, de forma escalável e com a segurança garantida (RICCI; ROKACH; SHAPIRA, 2011).

Como o investimento em Sistemas de Recomendação costuma ser alto, acaba sendo inviável para muitas empresas (TEAMR, 2022). Isso porque, muitas vezes, o Sistema de Recomendação não tem nenhuma ligação com o negócio da empresa, ou seja, ela não tem o conhecimento necessário para construí-lo. Assim, seria necessária a criação de uma nova área, com funcionários especializados, e também uma superação da falta de maturidade da empresa no tema. Nesse cenário, a implementação acaba sendo mais lenta, o custo é alto e a margem para erro é muito baixa, fazendo com que o risco do investimento seja grande e, muitas vezes, uma barreira para a utilização de um Sistema de Recomendação.

Pensando nesse viés da acessibilidade da tecnologia, o Software por Serviço (SaaS) surge como uma ótima alternativa, uma vez que ele abstrai toda a lógica e a operação necessárias para a solução. Com isso, ela é adquirida pronta para uso, o que reduz os custos de implementação. Esse modelo de negócio tem se popularizado muito e tem avançado em conjunto com o crescimento da computação em nuvem. Um bom

exemplo disso é o Paypal[4], um serviço de pagamentos online, que tem a praticidade e a segurança como seus principais benefícios. Nesse sentido, levando em consideração os desafios relacionados aos Sistemas de Recomendação e as facilidades propostas pelo SaaS, percebe-se que a ideia de um Sistema de Recomendação como serviço é bem plausível (SUSHIL; LEENA; SANDEEP, 2010).

Assim como no SaaS, em um Sistema de Recomendação como serviço, toda a infraestrutura e o ambiente já estarão prontos para o uso. Além disso, há diversas vantagens, que incluem a segurança dos dados e um sistema em constante evolução. Caso a solução não funcione, a empresa pode optar simplesmente por não continuar com ela, diferentemente de quando há um investimento muito alto em um sistema que foi construído do zero pela empresa. O custo inicial é muito baixo pois você só paga pelo uso e não precisa pagar por todo o tempo de implementação que existiria sem o SaaS. É basicamente contratar e já utilizar.

Com isso, o Sistema de Recomendação como serviço oferece um investimento bem menor, democratizando o uso de Sistemas de Recomendação e oferecendo um bom custo-benefício.

3.2 Trabalhos Relacionados

Em Afify et al. (2017), temos a proposta de um Software como Serviço (SaaS) de recomendação de Qualidade de Serviço (QoS) baseado em reputação. Nele, são processadas, semanticamente, as solicitações dos usuários para encontrar serviços que tenham correspondência com o negócio. Posteriormente, a filtragem híbrida é utilizada para validar os serviços com base em metadados de serviços, reputação e interesses do usuário. E por fim, um conjunto de serviços recomendado é classificado usando algumas características. Além de propor um novo método para calcular a reputação do serviço com base em feedbacks de usuários. Alguns desafios discutidos nesse artigo são os mesmos enfrentados por Sistemas de Recomendação genéricos como os problemas de *cold-start*, análise de conteúdo limitada e baixo desempenho.

[4]https://www.paypal.com/

Além de fazer a proposta de uma abordagem híbrida para o serviço, a fim de aumentar a aceitação dos usuários.

Já em Raigoza e Karande (2017), é feito um estudo de um Sistema de Recomendação em um ambiente em nuvem responsável por recomendar filmes. Com base no modelo de Software como Serviço (SaaS) proposto, foi feita uma análise de dois modelos de recomendação, a filtragem colaborativa e a filtragem baseada em conteúdo, onde foi analisado também o ganho a partir de uma base de dados com mais avaliações, até o problema de *cold-start*.

No caso de Abbas et al. (2015), é proposta uma estrutura baseada em nuvem que oferece recomendações personalizadas sobre os planos de saúde. São analisados problemas de um Sistema de Recomendação genérico que lida com dados heterogêneos e diferentes representações de um mesmo dado, em uma estrutura de Software como Serviço (SaaS).

Diferente de artigos, a SmartHint[5] é uma empresa, e já é um dos maiores fornecedores de serviços de sistemas de busca e recomendação para *e-commerce* da América Latina e faz parte do Grupo Magazine Luiza. Eles contam com diversas ferramentas que melhoram a experiência do usuário com um *e-commerce* e uma delas é uma vitrine inteligente, na qual recomendam os produtos de forma inteligente dentro da loja on-line.

3.3 Proposta

Conforme discutido anteriormente, os Sistemas de Recomendação têm bastante relevância atualmente, com uma popularidade cada vez maior, devido aos benefícios que eles oferecem. No entanto, com os altos custos de um investimento nessa área e com uma proposta mais acessível oferecida pelo SaaS, entende-se que um Sistema de Recomendação como serviço é uma ótima alternativa. Com isso, este trabalho irá trazer uma proposta de um Sistema de Recomendação como serviço, a fim de democratizar o uso desse tipo de tecnologia em qualquer lugar. Para isso, o foco será

[5]https://www.smarthint.co/

na arquitetura do sistema e na análise da mesma, para atingir os objetivos deste trabalho.

O Sistema de Recomendação como Serviço aqui proposto terá como objetivo a entrega de uma recomendação utilizando a técnica de Filtragem Colaborativa. Também, para facilitar o entendimento e a abstração do problema, a modelagem proposta por este trabalho levará em conta, em um primeiro momento, apenas um modelo de serviço, para um cliente, contendo apenas um Sistema de Recomendação. Nesse modelo, o primeiro desafio com relação à proposta é a avaliação e suas diferentes formas de ser realizada. No iFood[6], por exemplo, as avaliações são por meio de estrelas, em uma escala que varia de um a cinco; já no Instagram[7], é utilizado o modelo de gostar.

Com isso, a modelagem deste trabalho prevê que cada Sistema de Recomendação utilizará um Conjunto de Preferências, no qual será abstraído qualquer formato de avaliação. Por isso, nomeamos a classe como Feedback, mas vale ressaltar que um mesmo Conjunto de Preferências pode ser utilizado por diversos Sistemas de Recomendação. Além disso, um Conjunto de Preferências é composto de suas Preferências: o modelo de avaliação de gostar ou não gostar da Netflix[8], por exemplo, tem duas Preferências, o "gostar" e o "não gostar". No entanto, não podemos utilizar apenas feedbacks explícitos, logo, nossa modelagem permite também feedbacks implícitos, como um vídeo assistido, um artigo que foi lido, a compra de um produto, entre outros.

Essa modelagem inicial para o problema está representada pela figura 3.1, na qual um Sistema de Recomendação é composto, primeiramente, por UsuárioSR (Usuário de um Sistema de Recomendação) e ItemSR (Item de um Sistema de Recomendação), que têm a principal interação do sistema. A partir dessa relação, tem-se o Feedback, em que um Usuário pode avaliar vários Itens e um Item pode ser avaliado por diferentes Usuários. Além disso, ao dar o feedback, o Usuário determina sua Preferência e, com isso, cada Feedback possui uma Preferência, mas

[6]https://www.ifood.com/
[7]https://www.instagram.com/
[8]https://www.netflix.com/

essa Preferência pode estar em diversos Feedbacks. Todas essas informações estão dentro de um Sistema de Recomendação que, por sua vez, tem ligação com quase todas as classes.

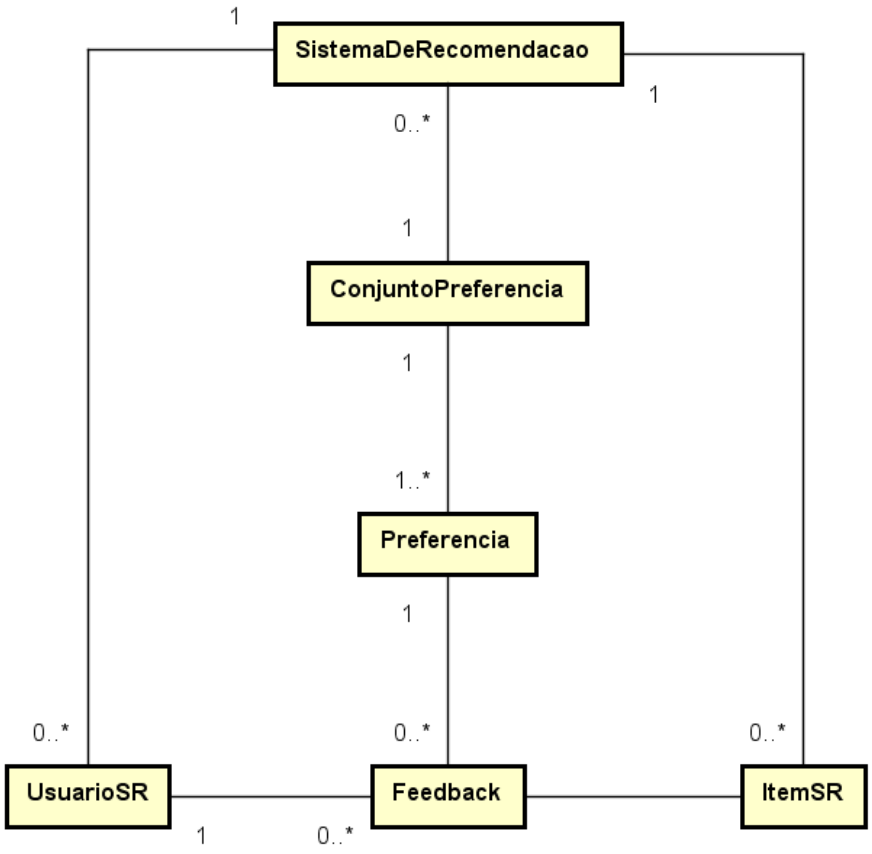

Figura 3.1: Diagrama de classes de análise de um Sistema de Recomendação

Nessa proposta de Sistema de Recomendação como SaaS, temos como característica o consumo de recomendação por vários usuários, com seus próprios contextos, fazendo requisições no mesmo serviço. Para melhor atender a essas características, o modelo utilizado será o de multi-tenant (KOZIOLEK, 2010). Conforme mostra a figura 3.2, uma aplicação multi-tenant conta com vários usuários, com suas próprias configurações e contextos, mas que compartilham recursos como memória, banco de dados e *assets*. Tal formato permite que os usuários tenham muitas características customizáveis, ao contrário da proposta de um serviço genérico que atenda uma grande variedade de casos.

Sendo assim, é necessário atualizar a modelagem para atender às características de

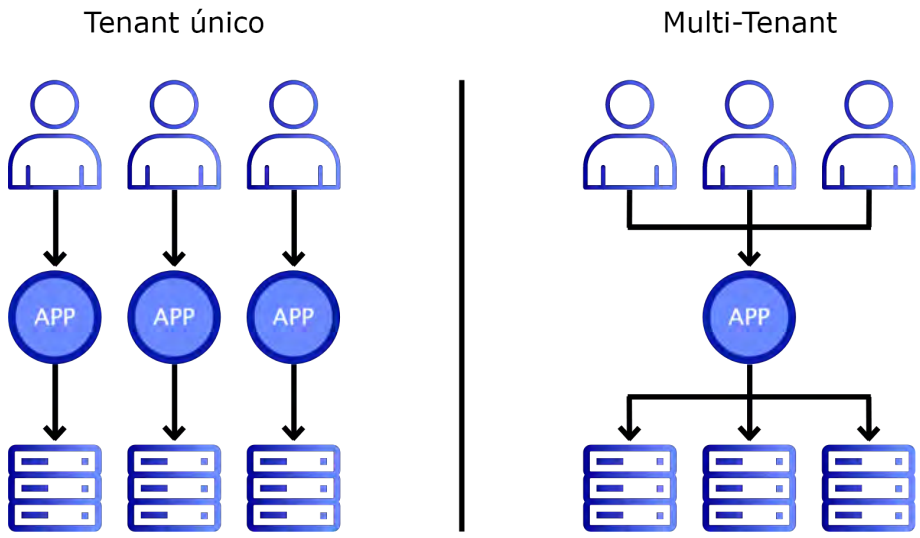

Figura 3.2: Comparação entre um modelo de Tenant único com um modelo Multi-Tenant

um serviço multi-tenant. Sua nova versão, presente na figura 3.3, tem como principal mudança a adição da classe Usuário. Outra característica importante do serviço é que ele não é restrito a apenas um algoritmo, já que um Sistema de Recomendação pode escolher dentre uma gama de algoritmos que foram implementados pelo serviço. Conforme citado anteriormente, a recomendação é muito mais que um algoritmo, pois saber qual é a melhor abordagem para cada modelo de dados é fundamental, tornando esse requisito extremamente importante. Por esse motivo, é necessária uma modelagem que preveja a abstração do algoritmo a ser utilizado, facilitando a adição de vários algoritmos no serviço.

Com isso, cada Usuário terá seu contexto de recomendação, ou seja, além de terem recomendações baseadas nas avaliações de seus clientes e nos seus Itens, também será possível utilizar uma variedade de algoritmos. Assim, foi adicionada a classe Modelo à modelagem, conforme mostra a figura 3.3. Nessa versão atualizada, um Sistema de Recomendação possui um Modelo, e um Modelo pode pertencer a vários Sistemas de Recomendação e também pode ter vários algoritmos de recomendação herdando desse modelo genérico, o que permite que um mesmo Usuário tenha um ou vários Modelos a sua disposição. Um mesmo Usuário pode solicitar um Sistema de Recomendação para recomendar os itens mais parecidos com determinado item de sua loja, e também uma recomendação de produtos que o usuário tem mais chances

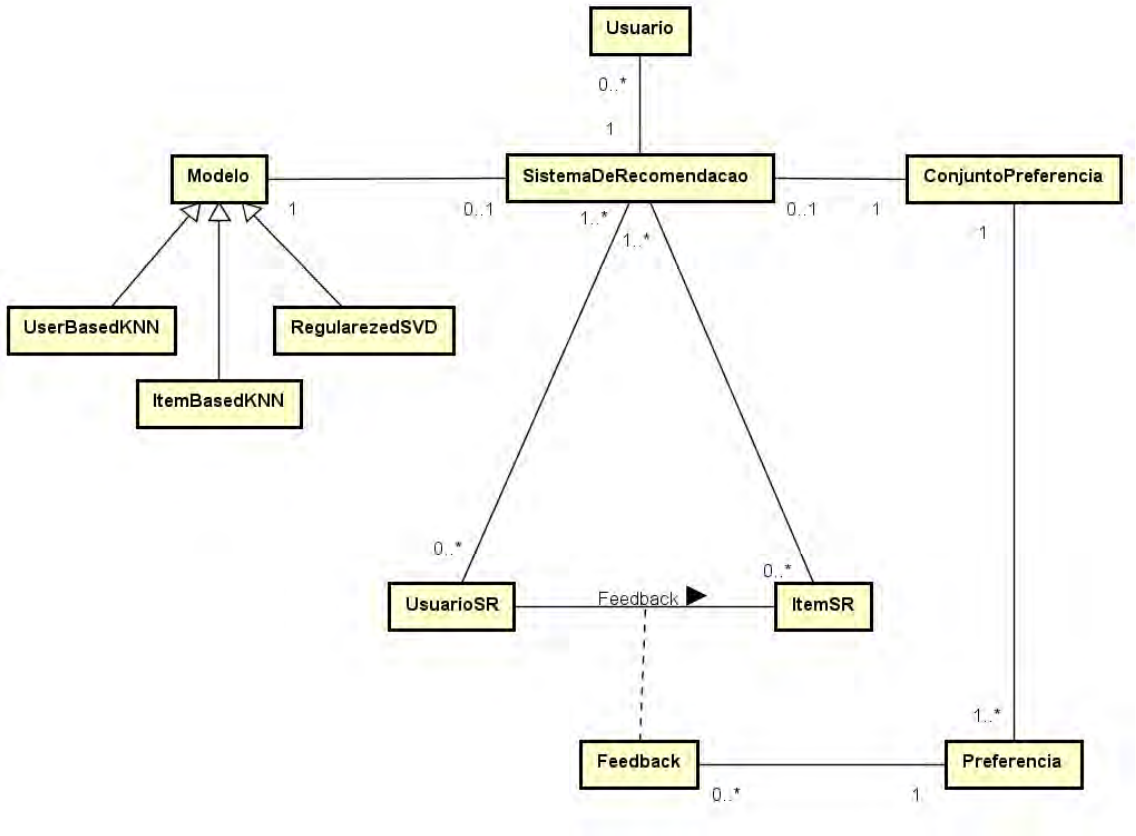

Figura 3.3: Diagrama de classe de análise do Sistema de Recomendação para arquitetura via serviço *Multi-Tenant*

de comprar, a partir de seu histórico de compras, por exemplo. Por isso, é importante ter várias formas de recomendar para os mesmos clientes, uma vez que esse é um requisito importante para chegar a um serviço versátil, que atenda a diversos tipos de usuários.

Na modelagem, o Modelo foi abstraído em uma única classe, contudo, ao analisar o problema, um requisito importante é a possibilidade de ter vários métodos de recomendação, para termos a possibilidade de escolher qual modelo utilizar, a qualquer momento. Com isso, um Modelo possui classes filhas, que são os métodos de recomendação que estarão disponíveis. Assim, cada algoritmo utiliza as funções básicas do modelo e também implementa suas funções específicas.

Com o diagrama de Classes do nosso Sistema de Recomendação como Serviço completo, podemos detalhar mais nosso modelo e discutir suas principais finalidades.

Com isso, temos a figura 3.4, que mostra o diagrama de classes de projeto, criado a partir de todos os requisitos definidos neste capítulo, e que será base para o desenvolvimento de nossa proposta.

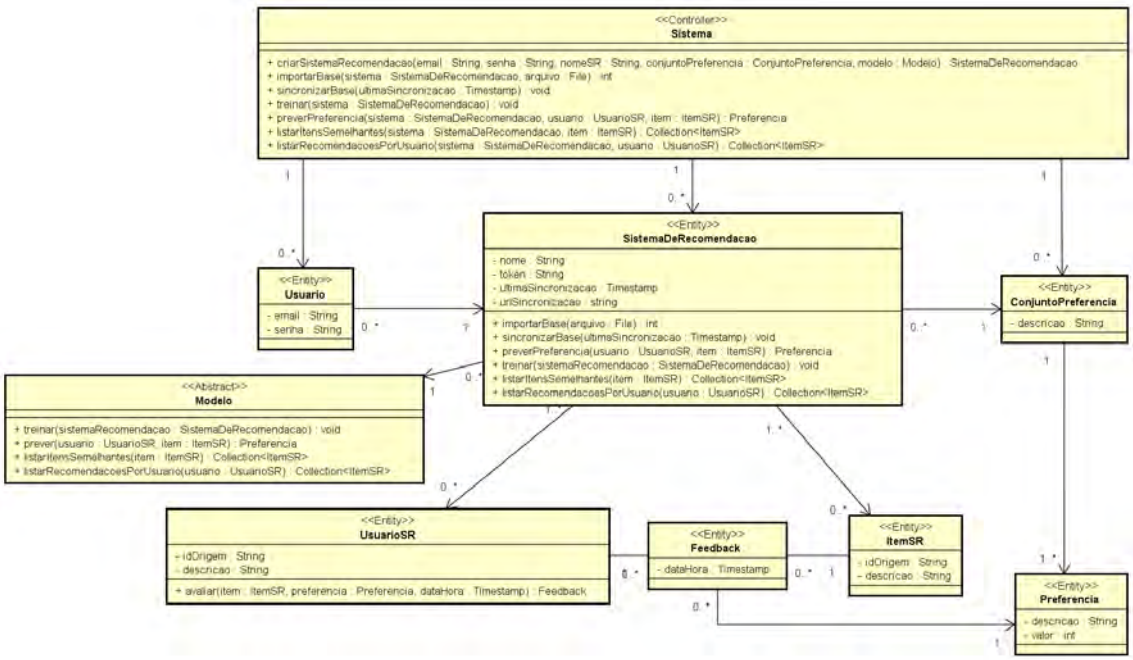

Figura 3.4: Diagrama de classe de projeto do Sistema de Recomendação para arquitetura via serviço *Multi-Tenant*

Como primeiro desafio, temos a importação da base de dados do Usuário, na qual a principal dificuldade é a padronização dos dados, já que cada cliente tem o dado em um formato. Sendo assim, devemos definir um padrão para a importação, garantindo que seja simples a geração desses dados por parte do cliente, e tenhamos o mínimo de erros ao realizá-la.

Com a importação realizada, logo após a conclusão, é bastante provável que a nossa base de dados já esteja desatualizada. Então, é preciso ter uma forma de sincronizar essas bases. A partir disso, e considerando que o objetivo é entregar a recomendação em um tempo aceitável, foi escolhida a recomendação *offline*, uma vez que nela, o modelo é atualizado de forma assíncrona, mas, dependendo da base e do algoritmo utilizado, esse processo pode demorar horas. Já a recomendação online tem a vantagem de estar sempre atualizada, pois está em um constante processo de aprendizado.

Logo, a sincronização dos dados deve ser feita com o cliente, de forma assíncrona, em um intervalo de tempo determinado, que pode variar a depender de cada cliente. Para isso, será feita uma requisição ao cliente e será enviado um *timestamp*, que consiste em um cadeia de caracteres que representa a data e a hora decorrida a partir de uma data e hora fixa, e é um recurso bastante utilizado para a comparação de dois tempos distintos. Neste caso, o *timestamp* terá a data e a hora da última sincronização realizada para obter apenas os novos feedbacks, usuários e/ou itens. Com isso, o cliente nos retornará um arquivo no mesmo padrão da importação para ser realizada a sincronização dos dados e alteração da data e hora da última.

Conforme a figura 3.5, o processo para obter os dados atualizados para o treinamento, funcionará da seguinte forma: o serviço irá realizar uma requisição ao Usuário informando o *timestamp* da última sincronização efetuada com esse Usuário, e receberá a resposta com uma lista de feedbacks, usuários e/ou itens novos a partir do tempo informado até o momento da requisição. Com o sucesso da requisição, ocorre o treinamento do modelo do Usuário, caso necessário.

A escolha de fazer a atualização dos dados por meio de um serviço ao cliente se deve à facilidade de obter apenas os itens novos. Com isso, não será preciso gerenciar os dados que foram alterados, nem obter toda a base sempre que o algoritmo de recomendação precisar ser treinado. Por outro lado, esse modelo também tem problemas, uma vez que essa solução implica na criação de um serviço por parte do Usuário para disponibilizar os dados. Um outro caminho possível seria a disponibilização de uma *View* da base de dados do Usuário, para que os dados fossem consumidos a partir dessa fonte, ou até mesmo em forma de arquivo, mas isso aumentaria a complexidade da solução do serviço. Ou a disponibilização de um método para sincronização onde a responsabilidade de agendar o envio de arquivos seria do cliente.

A partir dos dados sincronizados, o próximo passo é o treinamento do algoritmo, para realizar a recomendação dos Itens aos Usuários. Como o modelo de algoritmo é genérico, então o treinamento dependerá de cada algoritmo implementado. No entanto, o treinamento utilizará, basicamente, os dados de UsuárioSR, Feedback, ItemSR e Preferência. Uma vez que os dados estejam sincronizados e o algoritmo

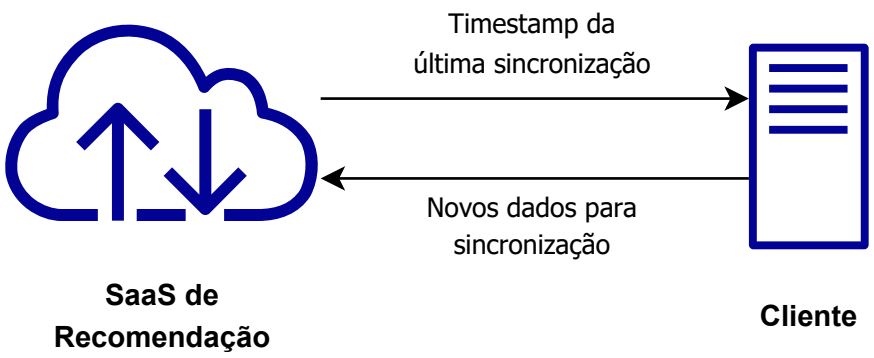

Figura 3.5: Arquitetura do processo de Sincronização da base de dados

treinado, já será possível começar a recomendar os Itens.

Para que o usuário final obtenha uma recomendação a partir de nosso serviço, o servidor do cliente precisará informar o UsuárioSR e/ou o ItemSR para receber uma lista com os identificadores dos itens recomendados, assim como mostra a figura 3.6. Dessa forma, nosso serviço retorna a recomendação para o serviço do cliente e a aplicação web recebe uma resposta com a recomendação. Além disso, a depender do tipo de recomendação, o serviço poderá precisar apenas do UsuárioSR ou apenas do ItemSR. Com essa lista de recomendações recebida, bastará que o Cliente exiba os itens recomendados para o UsuárioSR, sem precisar se preocupar com o processo de construção desse Sistema de Recomendação.

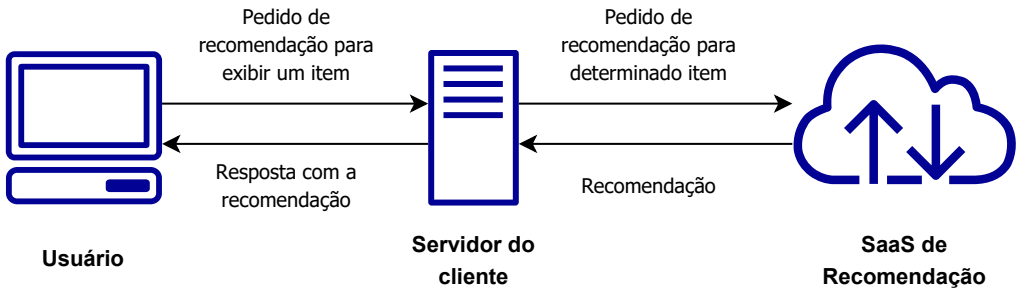

Figura 3.6: Ilustração do processo de obter uma recomendação a partir do sistema de um cliente qualquer

Ainda que o o serviço se utilize de vários métodos de recomendação, tratamos as exceções de forma única, para facilitar a implementação dos algoritmos. Nesse

sentido, um problema difícil, porém bastante comum é o *cold-start*, que consiste na necessidade de recomendar a partir de um ItemSR que ainda não teve feedback. Nesses casos, a filtragem colaborativa pura não consegue realizar a recomendação (SCHEIN et al., 2002), visto que ela é totalmente baseada nas preferências. Como alternativa para esses casos, serão utilizados os dados do conteúdo dos itens para encontrar a relação deles com outros itens, sem que haja necessidade de algum feedback.

Outro problema parecido com o *cold-start*, causado pela recomendação offline, é a sua sincronização. Nesse cenário, podem surgir três problemas: um UsuárioSR pode se cadastrar depois da última sincronização, ou seja, ele existe na base de dados do Cliente, mas não no serviço; isso também pode acontecer com o ItemSR e com os dois aos mesmo tempo. Como já mencionado, essas exceções serão tratadas da mesma forma, independentemente do algoritmo implementado.

Com isso, caso o UsuárioSR não exista, será utilizada a filtragem colaborativa para obter a recomendação desse UsuárioSR. Quando não houver os dados do ItemSR, será utilizada a recomendação baseada no perfil do UsuárioSR. E, nos casos em que não existem dados nem do UsuárioSR, nem do ItemSR, o que é raro, serão retornados para o Cliente os itens mais bem avaliados.

Capítulo 4

Projeto

Mesmo com todo o crescimento e os avanços dos Sistemas de Recomendação, uma grande barreira que ainda existe é a sua implementação. Com isso, esse projeto propõe um Sistema de Recomendação como Serviço, no qual nos aproveitamos das vantagens de um Software como Serviço (SaaS), facilitando o acesso a um Sistema de Recomendação, que deve atender aos mais diversos consumidores, com negócios dos mais variados tipos, e que utilizam diferentes métodos de avaliação e recomendação.

Neste capítulo, mostraremos o desenvolvimento da solução, as tecnologias utilizadas, as decisões, os principais desafios e suas soluções, assim como as vantagens e desvantagens de cada decisão tomada. Assim, a seção 4.1 se refere às tecnologias utilizadas no desenvolvimento, enquanto o serviço proposto é detalhado na seção 4.2. Além disso, foi desenvolvido um cliente para o consumo desse serviço, a fim de ter uma perspectiva completa do processo e obter uma recomendação proveniente do nosso Sistema de Recomendação como Serviço, que será especificado na seção 4.3.

4.1 Tecnologias utilizadas

Para o desenvolvimento de nossa proposta, foi utilizada a linguagem de programação JavaScript, especificamente o AdonisJS, um *framework* de Node.js, que oferece uma solução *backend* para criação de aplicações web do lado do servidor. O banco

de dados escolhido foi o PostgreSQL, por ser de licença gratuita, de código aberto, que atende a todos os requisitos da nossa proposta e ainda possui integração nativa com o ORM (*Object Relational Mapper*) do AdonisJS, o Lucid.

Nas próximas subseções serão apresentadas as tecnologias utilizadas no presente trabalho.

4.1.1 Ambiente de desenvolvimento

O editor de texto gratuito *Visual Studio Code* (vscode) foi utilizado para o desenvolvimento, pela sua alta popularidade, o que justifica o fato de que em 2019, ele era responsável por mais da metade do uso de Ambiente de Desenvolvimento Integrado (IDE) (OVERFLOW, 2019). Para o gerenciamento do banco de dados PostgreSQL, utilizamos o pgAdmin[1], que é gratuito e multiplataforma. Já para o versionamento, utilizamos o Git[2] que é o sistema de controle de versão mais usado no mundo (ATLASSIAN, 2021).

4.1.2 JavaScript

JavaScript é a linguagem de programação interpretada estruturada, de script em alto nível, com tipagem dinâmica fraca e multi-paradigma (protótipos, orientado a objeto, imperativo e funcional). Segundo Overflow (2020), o JavaScript era a linguagem mais usada pelos desenvolvedores no mundo e o grande motivo dessa popularidade é a grande compatibilidade dos navegadores web e sua facilidade de aprendizado. Flanagan (1998) cita que JavaScript é uma linguagem onipresente, possuindo uma enorme comunidade, cursos e suporte e até hoje vem sendo atualizada. Por esses motivos, e também, pelo conhecimento já adquirido na linguagem, escolhemos o JavaScript para o desenvolvimento de nosso trabalho, tanto no cliente quanto no servidor.

[1] https://www.pgadmin.org/
[2] https://git-scm.com/

4.1.3 Node.js

O NodeJs é um ambiente de execução JavaScript que executa do lado do servidor. Com isso é possível criar aplicações JavaScript que não rodam só no *browser*. O NodeJs é utilizado por grandes empresas atualmente, como Netflix[3], Uber[4] e Linkedin[5].(LENON, 2018)

Como principais características do Node.js, temos o fácil aprendizado, por ser baseado em JavaScript, a linguagem mais utilizada em 2020, como dito anteriormente, ser multiplataforma, e a flexibilidade com o NPM (*Node Package Manager*), que é um gerenciador de pacotes. Ele é o maior repositório de software atualmente, com uma comunidade bastante ativa, o que aumenta a produtividade, já que temos muitos pacotes que implementam diversas facilidades (TILKOV; VINOSKI, 2010). Outras vantagens ao utilizar o NodeJs, são que seu modelo com apenas uma thread e I/O não bloqueante permite um grande volume de requisições ao mesmo tempo, além de não consumir tanto recurso do hardware utilizado.

4.1.4 AdonisJS

Existe uma grande quantidade de frameworks para o desenvolvimento web para o NodeJS. De modo geral, eles são considerados microframeworks, em que eles são especializados em resolver um problema, como por exemplo o Express[6] para o roteamento ou Sequelize[7] que faz ORM. Desta maneira, o desenvolvedor deve juntar todos os microframeworks para facilitar a criação de um sistema web, criando um custo de manutenção dessa integração.

Por outro lado, em 2004 surgiu o framework Rails[8] que revolucionou o desenvolvimento Web, quando implementou fortemente conceitos que visam aumentar a produtividade do desenvolvimento como o *DRY* e *convenction over configuration*. Desta maneira, o Rails possui uma forte opinião em como fazer e organizar o

[3]https://www.netflix.com/br/
[4]https://www.uber.com/br/pt-br/
[5]https://www.linkedin.com/
[6]https://expressjs.com/pt-br/
[7]https://sequelize.org/
[8]https://rubyonrails.org/

desenvolvimento de software, facilitando o desenvolvimento.

Nesse contexto, surge o AdonisJS, que é um framework opinado de backend que ajuda a criar aplicações web. Ele define padrões de organização do projeto e tem algumas ferramentas muito interessantes, como o Lucid ORM, que é um mapeador de objeto relacional, em que o modelo do banco de dados é baseado no modelo de objetos de domínio da aplicação. Também podemos autenticar usuários, fazer *upload* de arquivos, enviar e-mails de forma simples, já que o framework implementa muitas tarefas básicas presentes em uma aplicação web.

4.1.5 React

React[9] é uma biblioteca JavaScript para criação de interfaces de usuário de forma declarativa, eficiente e flexível. Tem uma estrutura baseada em componentes encapsulados que gerenciam seu próprio estado, em que sua combinação resulta em interfaces complexas. Com o React Native[10] temos a possibilidade de gerar aplicativos nativos Android e IOS usando React.

O React surgiu em 2011 desenvolvido pelo Facebook[11] e vem sendo uma biblioteca muito usada. Neuhaus (2017) mostra que já em 2017, o React ultrapassava *frameworks* web bastante famosos como Angular e o VueJS.

4.2 Desenvolvimento

A partir de toda a modelagem e dos requisitos presentes em nossa proposta, foi realizado o desenvolvimento de nosso Sistema de Recomendação como Serviço. Suas principais características são a simplicidade no processo, desde a criação de um novo Sistema de Recomendação até a recomendação ao cliente, e também uma grande aderência à maioria das modelagens para recomendação, por conta de sua modelagem mais genérica. Serão apresentados os detalhes de como foi desenvolvimento do presente trabalho e será detalhada cada decisão do desenvolvimento, junto com suas

[9] https://pt-br.reactjs.org/
[10] https://reactnative.dev/
[11] https://pt-br.facebook.com/

vantagens e desvantagens, além de outras opções que poderão ser implementadas em trabalhos futuros.

Para a demonstração do desenvolvimento da proposta, utilizaremos um caso prático, mostrando em cada etapa a chamada do serviço e seu retorno, simulando o consumo de nossa solução por uma aplicação. Para isso, utilizaremos a ferramenta Insomnia [12] que é um *API Client* de código aberto, no qual podemos visualizar os parâmetros enviados a *API* e seus retornos. Também foi desenvolvida para fins de demonstração, uma aplicação web que utilizará nosso serviço na prática. Nela, são realizadas avaliações de filmes e, a partir delas, são recomendados novos filmes. Para essa demonstração, utilizaremos a base de dados para teste do MovieLens[13], que contém avaliações de filmes e é disponibilizada de forma gratuita.

Logo, para a demonstração de tudo que foi desenvolvido, esta seção foi dividida em cinco subseções, nos quais passaremos por toda a solução. A subseção 4.2.1 falará a respeito da primeira etapa, a de cadastros básicos, ou seja, o cadastro do mínimo necessário para ter acesso a uma recomendação, além das partes de importação das avaliações que estão em 4.2.2. A subseção 4.2.3 abordará uma parte importante do processo de recomendar, que não necessariamente é obrigatória. Já a subsecção 4.2.4 falará da recomendação em si, ou seja, a partir de tudo que for cadastrado, importado e, se necessário, treinado, o serviço irá fornecer a recomendação solicitada pelo cliente. E por último, na subsecção 4.2.5, será abordada a sincronização dos dados dos clientes, uma etapa importante, que ocorrerá durante toda a vida útil de um Sistema de Recomendação. Com isso, teremos mostrado o processo completo e passado por todos os detalhes do desenvolvimento de nossa proposta.

4.2.1 Cadastros

Como citado anteriormente, o primeiro passo para obter a recomendação desejada é o cadastro. Primeiramente, para simplificar o processo de criação de um Sistema de Recomendação, foi criada uma rota de *setup* inicial, em que serão enviados dados

[12] https://insomnia.rest/
[13] https://movielens.org/

para o cadastro e configuração inicial do sistema que deseja receber recomendação, além da criação do usuário administrador. Na rota, como mostra na figura 4.1, deve ser informado o e-mail e senha do usuário administrador, nome do sistema que irá obter as recomendações, e o conjunto de preferências que será utilizado nesse Sistema de Recomendação.

A princípio, foram adicionados ao serviço dois conjuntos de preferências: o conjunto de notas que vão de um a cinco, que também é conhecido por meio das estrelas, e o conjunto binário, que pode ser o tão utilizado "gostei" e "não gostei". O usuário criador será associado a esse novo sistema e precisará se autenticar quando for realizar a importação. Além disso, será criado um token referente ao novo sistema, que será utilizado para autenticar o sistema no consumo da recomendação.

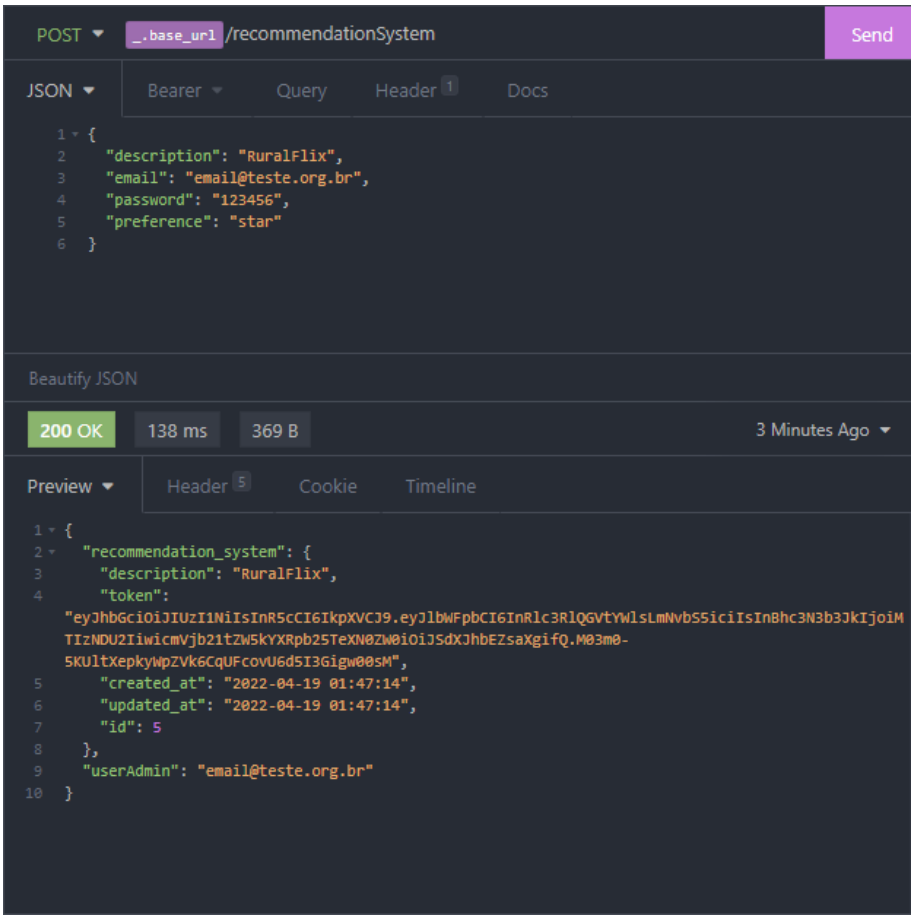

Figura 4.1: Exemplo da chamada a *API* e seu retorno para a criação de um Sistema de Recomendação

Também é realizado um cadastro básico do usuário, em que são definidos seu e-

mail de login e senha e qual Sistema de Recomendação irá utilizar. Em nosso serviço, todas as requisições necessitam ser autenticadas, exceto a de cadastro de usuário e o setup inicial do Sistema de Recomendação. Nas demais requisições feitas pelo cliente, no caso de alterações em configurações de um Sistema de Recomendação, utilizamos uma autenticação no formato JWT, na qual ao logar, o usuário recebe um token e utiliza o mesmo para autenticar suas chamadas ao nosso serviço, como mostra a figura 4.2. No caso de chamadas para obter recomendação, importação ou sincronização utilizamos o token estático, obtido na criação do Sistema de Recomendação, conforme mostra a figura 4.1.

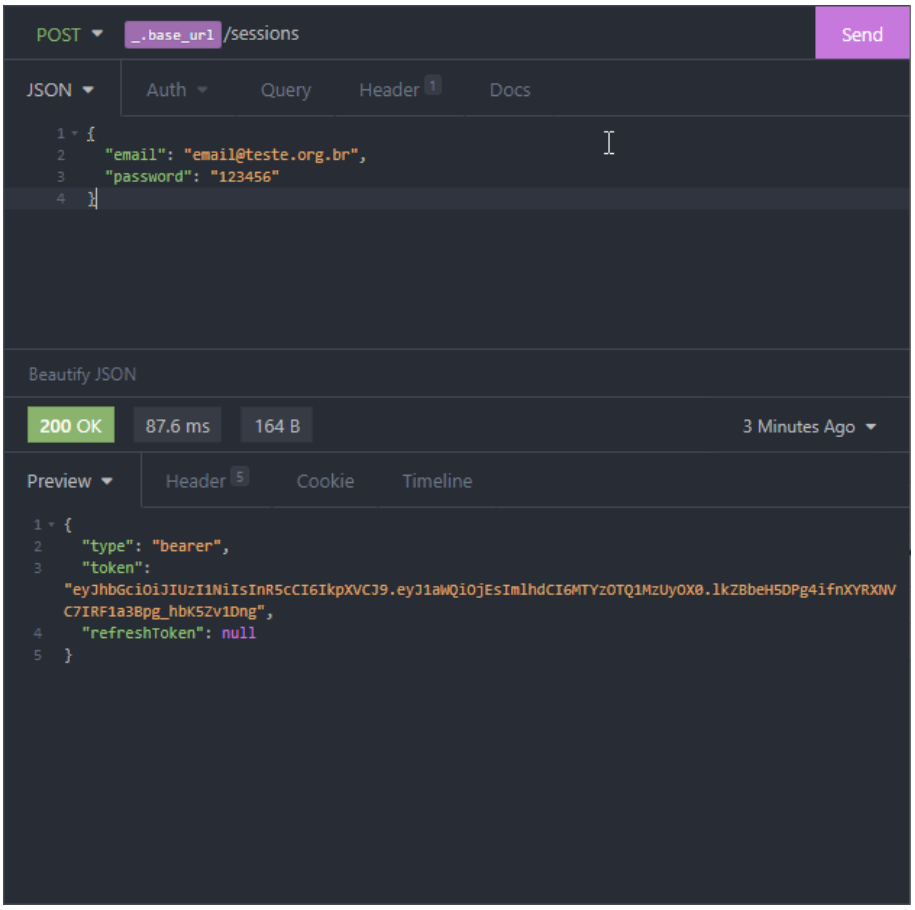

Figura 4.2: Exemplo da chamada a API e seu retorno para a autenticação no Sistema de Recomendação

4.2.2 Importação

Depois de ter todos os dados básicos necessários no cadastro, podemos realizar a importação dos dados de *feedback* do cliente. A princípio será utilizado um arquivo com os dados básicos, que são o usuário, *feedback*, item e o *timestamp* da data do *feedback* e estes dados formam um arquivo CSV (*Comma-separated values*). Na figura 4.3 temos um exemplo de importação, onde também é informado o token que identifica o respectivo Sistema de Recomendação e temos como resposta o número total de *feedbacks* importados.

Esse é o caso mais simples, pois como utilizaremos um modelo de recomendação baseado em Filtragem Colaborativa, não há necessidade de termos características a mais do usuário nem do item. No entanto, temos a possibilidade de estender nossa modelagem, de modo que seja possível adicionar qualquer característica, seja do item, usuário e até da própria avaliação, permitindo o uso da recomendação baseada em conteúdo ou até mesmo híbrida.

Nesses casos de modelos de recomendação que necessitem de mais informações dos itens e/ou usuário, deverá ser feita uma importação de três arquivos, ou seja, além do arquivo de avaliações, teremos o de itens e o de usuários com os seus respectivos identificadores e todos os seus atributos. Esses identificadores ficam salvos no banco como um identificador de importação e não são caracterizados como chave primária, já que podemos ter outros Sistemas de Recomendação que utilizam identificadores iguais. No entanto, ainda é armazenado separadamente porque esse é o identificador conhecido pelo cliente que irá consumir esse serviço. Com tudo certo, o serviço irá retornar um código de sucesso e caso ocorra algum erro, teremos a identificação do mesmo para o cliente.

4.2.3 Treinamento do Modelo

A partir de toda as etapas iniciais de cadastro e importação concluídas, podemos tratar da recomendação em si, mas pra isso, primeiramente precisamos definir quais modelos de algoritmo iremos prever. Com isso, definimos que cada Sistema de

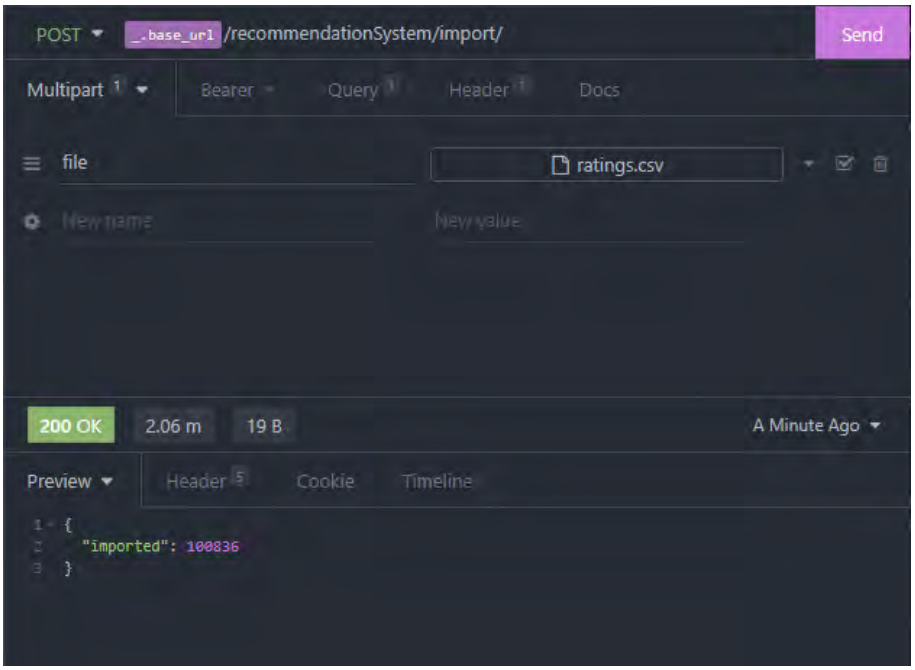

Figura 4.3: Exemplo da chamada a API e seu retorno na importação dos *feedbacks* para o Sistema de Recomendação

Recomendação poderá utilizar dois grupos de algoritmos, a princípio: os algoritmos de ranqueamento e os algoritmos de previsão. Esses são alguns dos mais utilizados na maioria dos casos e, dentro desses modelos, existe uma grande quantidade de algoritmos.

O modelo de previsão é um método que tenta prever uma avaliação de um usuário sobre determinado item, seja uma nota, ou um *like*, etc, a partir de um treinamento prévio. Esse treinamento consiste, basicamente, em ter uma base com os dados históricos das avaliações, para que o algoritmo escolhido possa definir um padrão e consiga prever as próximas avaliações com o mínimo de erro possível. Logo, nosso algoritmo de previsão é dividido em dois métodos obrigatórios (BREESE; HECKERMAN; KADIE, 2013): o de treinamento e o de recomendação, levando em conta essa obrigatoriedade do treino, garantindo que o algoritmo esteja sempre treinado para realizar a recomendação. Para atender esse requisito, disponibilizamos uma rota para que o cliente sempre tenha a possibilidade de realizar o treinamento. Inclusive, uma característica desejada é que o cliente possa programar a forma que ocorrerá esse treinamento, onde pode ocorrer num intervalo de tempo, com base

numa porcentagem de avaliações novas e etc. Mas, a princípio, temos só o *endpoint* de treinamento.

O modelo de ranqueamento, em sua forma básica, é bem comum e bastante simples, no qual temos um algoritmo que pode trazer, por exemplo, os itens mais bem avaliados no mês, os principais lançamentos da semana e etc (DESHPANDE; KARYPIS, 2004). Esse modelo é importante até para resolver um problema que discutimos na documentação que é o *cold-start*. Estes exemplos, diferentemente do modelo de previsão, não precisam de um treinamento obrigatório. Assim, existem algoritmos de ranqueamento que utilizam de treinamento, enquanto outros não exigem essa etapa.

4.2.4 Recomendação

Esta subseção trata do núcleo central do presente trabalho, na qual a aplicação irá de fato recomendar um ou vários itens, podendo ser a partir dos gostos de um usuário ou dos principais itens relacionados a um determinado item, entre outros tipos de recomendação. Além disso, há a previsão da relação de um item com um usuário. Logo, a partir da escolha do modelo que será utilizado, na criação do Sistema de Recomendação, temos os métodos que todos esses algoritmos implementam por padrão, como foi definido na proposta. São eles: a previsão, a listagem de itens semelhantes e a lista de recomendação.

Na previsão, temos a hipótese do *feedback* do usuário para com um item. Como mostra a figura 4.4, é passado o identificador do usuário e o identificador do item, além do token que identifica o Sistema de Recomendação, tendo como reposta a previsão da preferência do usuário com o item.

Já na listagem de itens semelhantes, obtemos uma lista de itens que apresentam certa relevância com base em um item previamente informado. A figura 4.5 mostra um exemplo prático da utilização desse método, no qual é passado o código de um item, a quantidade de itens a serem retornados e o token que identifica o Sistema de Recomendação, tendo como retorno uma lista de itens como maior taxa de relevância com base no item informado.

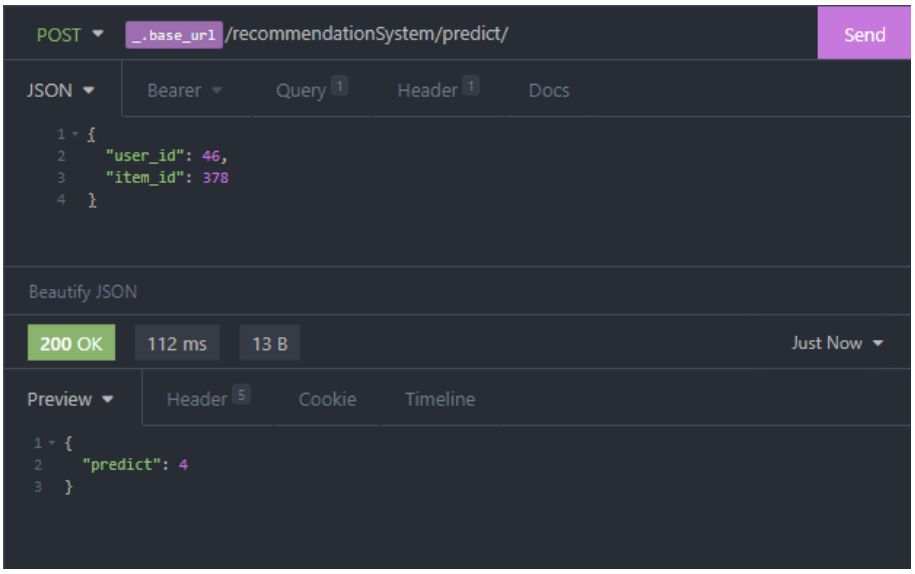

Figura 4.4: Exemplo da chamada a API e seu retorno na previsão do *feedback* do usuário para com um item

Por fim, temos a lista de recomendação para um usuário específico. A figura 4.6 mostra o exemplo prático da chamada ao serviço, no qual é passado o usuário alvo da recomendação, o limite de itens a serem retornados, e o token que identifica o Sistema de Recomendação, tendo como retorno uma lista de itens como maior taxa de relevância com base no usuário informado.

Como o foco deste trabalho é a validação de um modelo de Sistema de Recomendação como Serviço, não foi implementado nenhum modelo de recomendação. Os exemplos mostrados acima utilizam uma função que retornam itens aleatórios de um sistema de recomendação cadastrado.

4.2.5 Sincronização

Apesar de o processo principal estar concluído, ou seja, a criação, a configuração e a recomendação, uma parte importante para que a nossa lista de recomendação sempre se mantenha relevante é a sincronização com a base do cliente. Como já foi dito, um sistema de grande porte pode ter milhares de dados gerados em questão de minutos, ou seja, quanto mais atualizamos nossos dados e treinamos, mais assertividade temos em nossa recomendação. No entanto, esse não é um processo

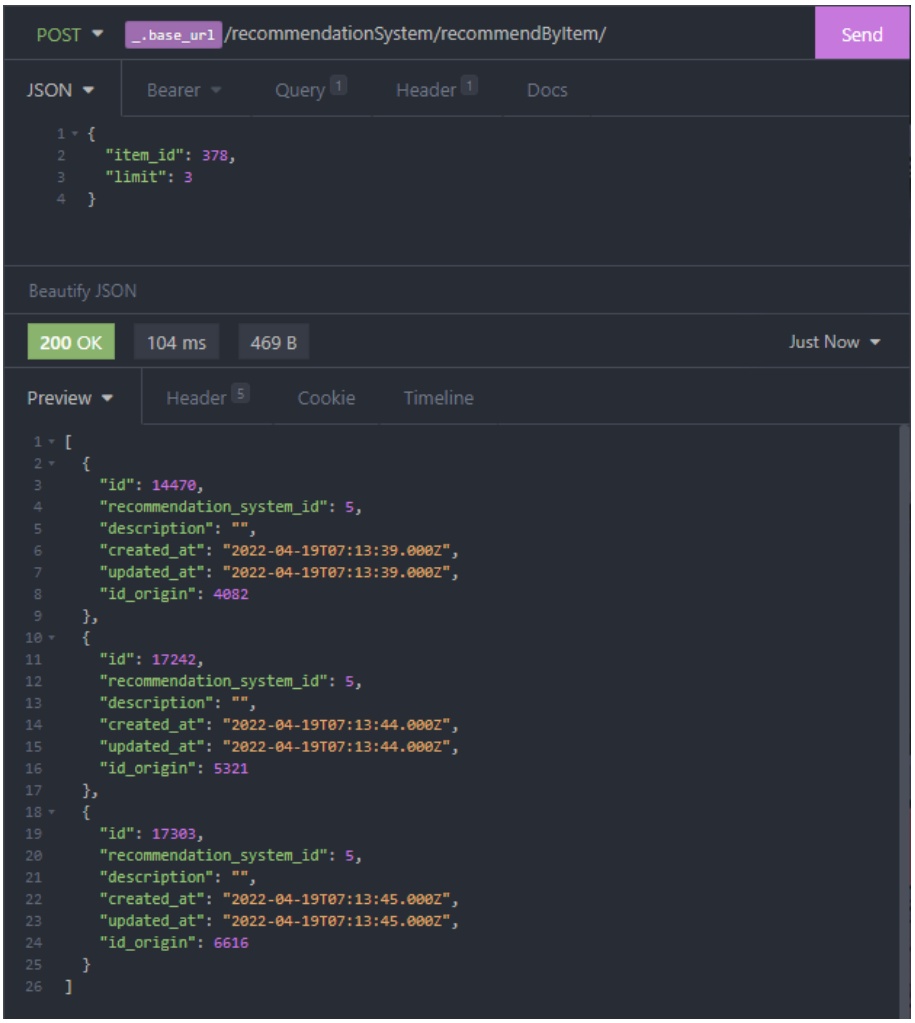

Figura 4.5: Exemplo da chamada a API e seu retorno da lista de itens semelhantes a item específico

rápido e varia de cliente para cliente, então nosso serviço tem uma rota disponível para a sincronização dos dados. Assim, será passada uma nova lista de dados, no modelo da importação, mas com os dados complementares aos que já existem. Em nossa plataforma, além de passar os dados de sincronização, deverá ser informado o token do sistema e um parâmetro opcional que é o que se deseja após a sincronização realizar o treino, caso necessário, para a nova base.

Além da rota de sincronização, disponibilizamos uma para data e hora da última realizada, e o status, a fim de ajudar na consulta do status, pois pode ser um processo demorado, e facilitar na seleção da diferença dos dados do cliente cadastrados em relação ao nosso serviço. Outra opção desejada é o próprio cliente disponibilizar

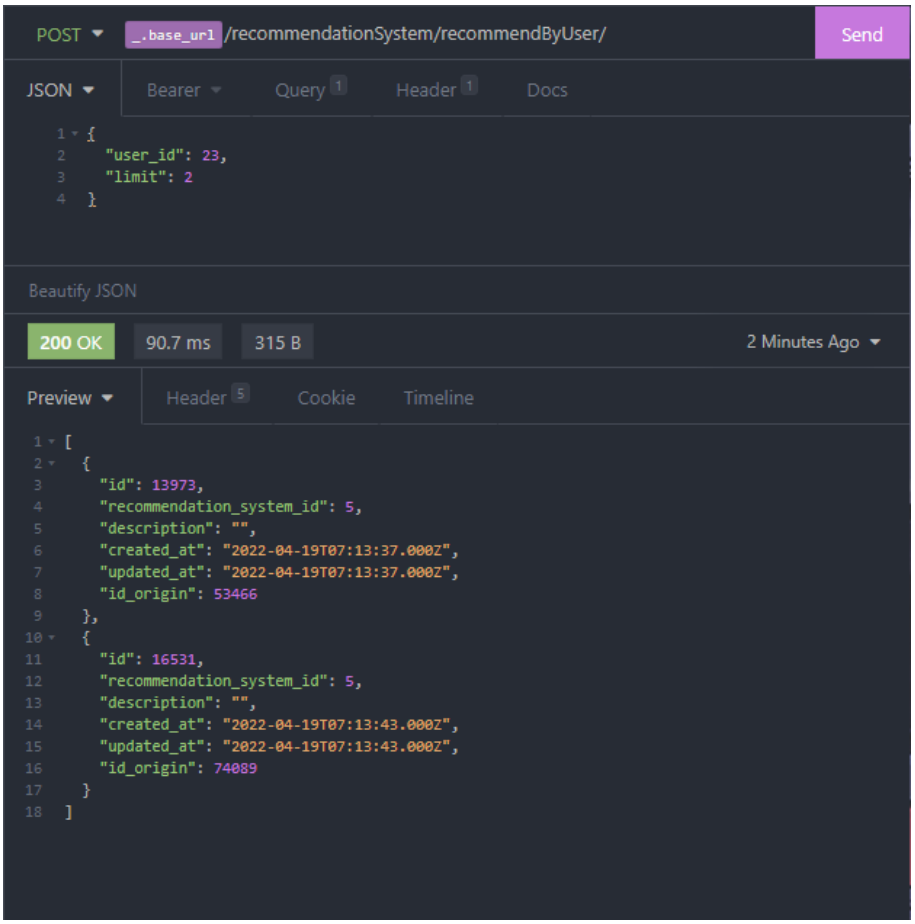

Figura 4.6: Exemplo da chamada a API e seu retorno da lista de itens relevantes para um usuário específico

uma rota e agendar a sincronização, de forma que tudo ocorra de modo automático. No entanto, para ter uma solução mais genérica, e que tenha menos chances de ocasionar erros, pelo fato de o contrato e o formato dos dados para sincronização serem definidos do lado do serviço, optamos pela solução citada acima.

4.3 Cliente e caso de teste

Como nossa solução proposta é um SaaS de recomendação, em que o principal objetivo é atender a diversos clientes, seja um site de receitas ou uma loja de roupas, tivemos a necessidade de mostrar, na prática, como seria o resultado em um cenário próximo do real. O objetivo era que toda a complexidade de uma recomendação fique do lado do nosso SaaS.

A partir disso, também foi desenvolvida uma aplicação web de recomendação de filmes que foi chamada de RuralFlix. Basicamente, a partir de um usuário, ela recomenda uma lista de filmes e, ao selecionar o filme, somos apresentados aos detalhes dele, seguidos por uma lista de filmes recomendados a partir do filme escolhido. Para isso, utilizamos dados das avaliações de filmes da base do MovieLens[14].

Além disso, essa lista de filmes mostrada nas figuras 4.7 e 4.8, funciona como um *plugin* desacoplado do resto do site, desenvolvido em React[15]. No projeto do RuralFlix, foi apenas importada e passada uma lista dos itens que foram retornados de nossa API de recomendação. Além disso, são possibilitados os parâmetros de customização, ou seja, é possível definir o número de itens a ser mostrado, a orientação da lista (horizontal e vertical), podendo passar também um link de referência para clicar no item e redirecionar para os detalhes do item desejado.

Na figura 4.7 temos a tela inicial da nossa aplicação web, com duas listas de filmes. Elas poderiam utilizar dois métodos de recomendação diferentes, ainda que chamem o mesmo serviço de recomendação para o mesmo cliente, mas seriam diferenciadas pelo algoritmo que irá recomendar esses itens. A primeira é uma lista de filmes mais bem avaliados no mês, que utiliza o método de ranqueamento, no qual aparecem os filmes com maior nota de avaliação dentre os usuários, em um intervalo de tempo. Já a segunda lista apresenta uma previsão de filmes que o usuário provavelmente gostará, com base em seu histórico, utilizando um algoritmo de previsão.

Já na figura 4.8, que apresenta os detalhes de um filme específico, aparece uma lista de recomendações a partir desse filme que está sendo detalhado, podendo utilizar uma recomendação que busque itens similares ao filme selecionado. Essa lista também está utilizando o mesmo componente de lista de recomendação utilizado na tela inicial, figura 4.7, mesmo tendo um número diferente de itens, pois isto é um parâmetro de configuração.

Este componente, que é basicamente uma lista de itens, foi criado com o objetivo de se adequar a qualquer tipo de tela, de forma que foi criado sem guardar estado

[14]https://movielens.org/
[15]https://pt-br.reactjs.org/

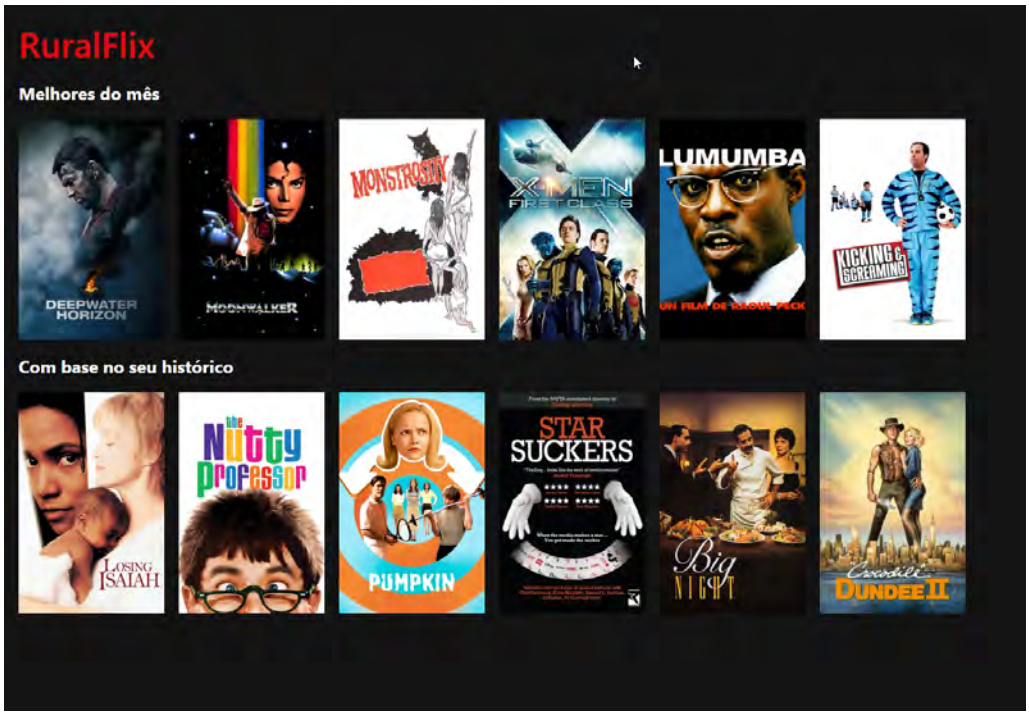

Figura 4.7: Tela inicial da aplicação onde são apresentadas dois tipos de lista de recomendação de filmes

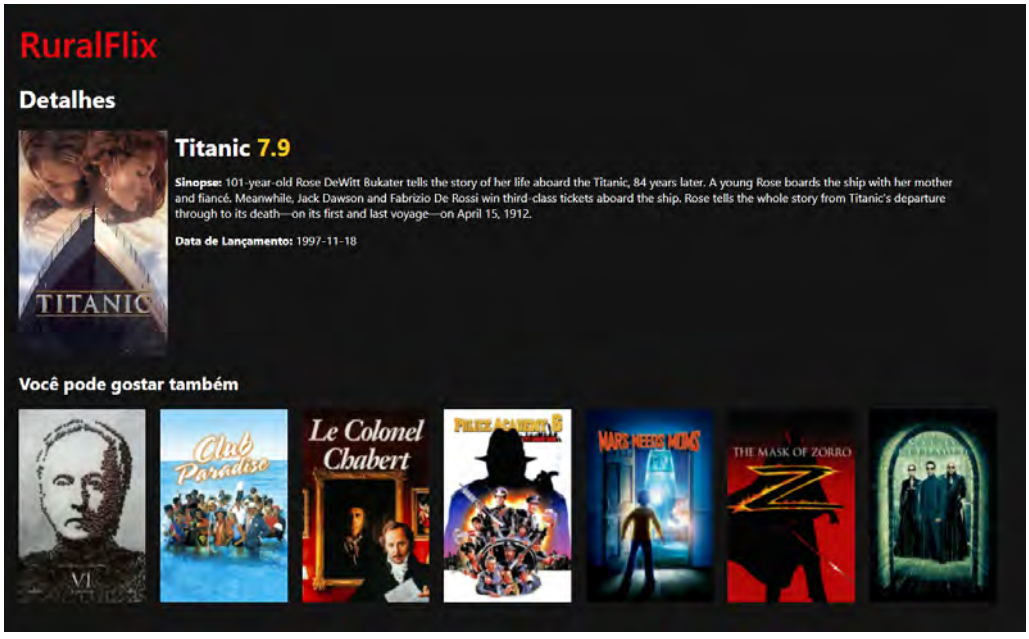

Figura 4.8: Tela de detalhes de um filme com uma lista de recomendação de filmes que o usuário também pode gostar como sugestão

e recebendo parâmetros para sua configuração. O código 4.9 mostra a estrutura da tela de detalhes do filme, figura 4.7, na qual o componente de recomendação é

denominado *ListRecommendation* e passamos os parâmetros que são variáveis para o componente.

```
export default function Details(props) {
    const { details } = props.location.state;
    return (
        <>
            <Container>
                <Header></Header>
                <TitlePage>Detalhes</TitlePage>
                <DetailsContainer>
                    <FolderMovieContainer>
                        <FolderMovie src={urlBaseImg + details.poster_path} />
                    </FolderMovieContainer>
                    <DetailsTextContainer>
                        <TitleMovie>
                            {details.original_title} <ReviewMovie>{details.vote_average}</ReviewMovie>
                        </TitleMovie>
                        <DescriptionMovie>
                            <b>Sinopse: </b>
                            {details.overview}
                        </DescriptionMovie>
                        <DateMovie>
                            <b>Data de Lançamento: </b> {details.release_date}
                        </DateMovie>
                    </DetailsTextContainer>
                </DetailsContainer>
                <ListRecommendation listTitle="Você pode gostar também" limit="7"></ListRecommendation>
            </Container>
        </>
    );
}
```

Figura 4.9: Trecho do código da pagina de detalhes do filme

O código da figura 4.10 mostra a estrutura do componente *ListRecommendation*, no qual ele recebe os parâmetros, obtém a lista de filmes recomendados e passa para um subcomponente, que é o Item dos seus respectivos dados, montando assim o componente completo. Já a figura 4.11 mostra o código do componente Item, responsável por mostrar a imagem do filme e ter a referência para caso o usuário clique no filme, levar para tela de detalhes do mesmo.

Com isso, temos um cenário bem completo do objetivo final deste trabalho, em que desenvolvemos um SaaS de recomendação de forma genérica que possa atender a diversas necessidades. Além disso, também foi desenvolvida uma aplicação para consumo desse SaaS, que mostra o quão simples e necessária essa solução se mostra.

```
const ListRecommendation = (props) => {
    const [details, setDetails] = useState([]);

    const { listTitle, limit } = props;

    useEffect(() => {···
    }, [props]);

    return (
        <>
            <ListContainer>
                <ListTitle> {listTitle} </ListTitle>
                <ListFlex>
                    {details.map((item) => {
                        return <Item key={item.id} itemDetails={item}></Item>;
                    })}
                </ListFlex>
            </ListContainer>
        </>
    );
};
```

Figura 4.10: Trecho do código do componente *ListRecommendation*, que é a lista de filmes para recomendados

```
const Item = (props) => {
    const [movie, setMovie] = useState({});
    const { itemDetails } = props;
    let history = useHistory();

    useEffect(() => {
        getMovieDetails(itemDetails.id_origin);
    }, [itemDetails.id_origin]);

    const getMovieDetails = function (itemId) {
        api.get(urlMovie + "?id=" + itemId).then(function (response) {
            theMovieDb.movies.getById(
                { id: response.data[0].tmdbId },
                (movie_return) => {
                    setMovie(JSON.parse(movie_return));
                },
                (error) => {···
                }
            );
        });
    };

    function handleOnClick() {
        history.push("/movie", {
            details: movie,
        });
    }

    return (
        <>
            <ItemContainer onClick={handleOnClick}>
                <ItemImage src={urlBaseImg + movie.poster_path}></ItemImage>
            </ItemContainer>
        </>
    );
};

export default Item;
```

Figura 4.11: Trecho do código do componente que Item, que é o filme na lista de filmes recomendados

Capítulo 5

Conclusão

Neste capítulo, será feita a conclusão do trabalho. A seção 5.1 apresenta as considerações finais a respeito do trabalho, enquanto a Seção 5.2 descreve as limitações provenientes da solução assim como possíveis trabalhos futuros.

5.1 Considerações finais

Os Sistemas de Recomendação têm sido uma tecnologia bastante presente em nosso dia a dia, e cada vez mais empresas vêm investindo nessa tecnologia, porém, seu custo é bastante alto, limitando sua utilização. Com o objetivo de suprir esse grande obstáculo, foi proposto e desenvolvido um formato de Sistema de Recomendação como Serviço, em que aproveitamos os benefícios da computação em nuvem, utilizando o modelo de SaaS, que fornece um serviço de recomendação que abstrai toda a dificuldade da implementação de um Sistema de Recomendação.

Com isso, foi desenvolvida uma *API*, com uma modelagem que tem como principais características a utilização de várias formas de avaliação, seja explícita como um *like* ou implícita como assistir um video até o fim. Também é possível fazer a seleção do algoritmo que será utilizado na recomendação e, no caso deste trabalho, utilizamos apenas algoritmos de filtragem colaborativa. Contudo, a modelagem pode ser estendida para que itens, usuários e também a própria avaliação tenham

mais atributos, viabilizando a recomendação baseada em conteúdo e até a junção das duas, que é a forma híbrida. Além disso, há funções muito importantes para o funcionamento de um Sistema de Recomendação e, por isso, foram criadas rotas para importação dos dados, enquanto o treinamento do modelo pode ser acionado a partir de uma chamada ao serviço. Assim, o cliente consegue uma recomendação sem muitas dificuldades, que é a grande proposta deste trabalho.

Ainda vale ressaltar que, para o auxiliar nos testes e como uma prova de conceito, foi desenvolvido um sistema web de recomendação de filmes, em que as recomendações apresentadas são provenientes do Sistema de Recomendação como Serviço criado nesse trabalho.

5.2 Limitações e trabalhos futuros

Durante o desenvolvimento deste trabalho, foi observado que, a concretização de uma solução completa, no ponto de vista de atender a maior gama de sistemas web possíveis e principalmente, explorar as vantagens derivadas de um Sistema de Recomendação como Serviço, seria uma tarefa bastante difícil para apenas este trabalho.

Com isso, ao fim desta monografia, podemos obter diversas sugestões trabalhos futuros vindos de algumas limitações observadas neste trabalho e também a exploração de novos desenvolvimentos a partir das vantagens de ter um Sistema de Recomendação como Serviço.

Uma limitação deste trabalho é a necessidade de uma interação direta com a *API*, pois não há um sistema administrativo para realizarmos configurações, conferir status do serviço e gerenciar permissões. Isso é algo bastante interessante pensando no usuário que irá utilizar a plataforma, o que torna relevante a criação dessa possibilidade em um trabalho futuro. Também temos o componente criado para listar as recomendações, que poderia ser convertido em um plugin para ser usado em diversos sistemas web diferentes, tendo mais opções de customização.

Outro item importante é a implementação e a extensão da gama de algoritmos

disponíveis para o uso, não só em filtragem colaborativa, mas também baseado em conteúdo e as versões híbridas. Com isso, para obter análises e melhorias com relação ao desempenho, uma possibilidade bastante interessante são os testes A/B em que podemos colocar, "lado a lado", dois métodos de recomendação para grupos de usuários distintos, e avaliar qual se sai melhor em determinado cenário. Para isso, seria necessário adequar a modelagem para que um método utilize mais de um algoritmo.

Com relação ao desempenho na entrega da recomendação, a princípio a solução foi modelada de forma a ser consumida pelo *back-end* do cliente, fazendo com que não sejamos tão performáticos com relação ao cliente final. Uma solução ideal que ajudaria nesse problema, seria que o componente de recomendação, apresentado para o cliente final, consultasse diretamente a recomendação de nosso serviço, e carregasse de forma assíncrona e desacoplada do resto da tela. No entanto, teríamos dificuldades relacionadas à responsabilidade em torno da segurança e da regra de negócio.

Outra possibilidade que esse modelo apresenta é a utilização do *transfer learning*, que é basicamente utilizar um modelo treinado em outro outro conjunto de dados, já que teremos diversos modelos. Contudo, para isso colocar em prática, surgiriam questões de segurança em relação o compartilhamento dos dados entre os clientes. Por isso tudo, percebe-se que o presente trabalho se encerra com portas abertas para diversas possibilidades e soluções que podem surgir a partir do que foi proposto aqui.

Referências

ABBAS, A. et al. A cloud based health insurance plan recommendation system: A user centered approach. *Future Generation Computer Systems*, v. 43-44, p. 99–109, 2015. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X14001587>.

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, v. 17, n. 6, p. 734–749, 2005.

AFIFY, Y. M. et al. A personalized recommender system for saas services. *Concurrency and Computation: Practice and Experience*, v. 29, n. 4, 2017. E3877 CPE-15-0484.R1. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3877>.

ATLASSIAN. *O que é Git*. 2021. Disponível em: <https://www.atlassian.com/br/git/tutorials/what-is-git>.

BREESE, J.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithm for collaborative filtering. *UAI*, 01 2013.

BUILDER, P. *SaaS ou On-Premises? Compreenda o que é melhor para você*. 2018. Disponível em: <https://www.projectbuilder.com.br/blog/saas-ou-on-premises/>.

CHOUDHARY, V. Software as a service: Implications for investment in software development. *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 2007.

COUGHLIN, T. *175 Zettabytes By 2025*. 2018. Disponível em: <https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/?sh=7d57a2705459>.

DESHPANDE, M.; KARYPIS, G. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 1, 2004. ISSN 1046-8188. Disponível em: <https://doi.org/10.1145/963770.963776>.

FLANAGAN, D. *Javascript: The Definitive Guide*. Third. Sebastopol, California: O'Reilly, 1998.

FLORIDI, L. *Information: A Very Short Introduction*. OUP Oxford, 2010. (Information: A Very Short Introduction). ISBN 9780199551378. Disponível em: <https://books.google.com.br/books?id=AiXR3qoXd-EC>.

FRYER, V. *The History of SaaS: From Emerging Technology to Ubiquity*. 2019. Disponível em: <https://www.bigcommerce.com/blog/history-of-saas/#the-history-of-saas>.

GUANG, L. et al. Positioning antifragility for clouds on public infrastructures. *Procedia Computer Science*, v. 32, p. 856–861, 12 2014.

JANNACH, D. et al. *Recommender Systems: An Introduction*. [S.l.]: Cambridge University Press, 2010.

KIDD, A. M. C. *What Is AIaaS? AI as a Service Explained*. 2021. Disponível em: <https://www.bmc.com/blogs/ai-as-a-service-aiaas/>.

KOWALCZYK, W.; SZLÁVIK, Z.; SCHUT, M. C. The impact of recommender systems on item-, user-, and rating-diversity. In: CAO, L. et al. (Ed.). *Agents and Data Mining Interaction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN 978-3-642-27609-5.

KOZIOLEK, H. Towards an architectural style for multi-tenant software applications. In: ENGELS, G.; LUCKEY, M.; SCHäFER, W. (Ed.). *Software Engineering 2010*. Bonn: Gesellschaft für Informatik e.V., 2010. p. 81–92.

LEAVITT, N. Is cloud computing really ready for prime time? *Computer*, v. 42, n. 1, 2009.

LENON. *Node.js – O que é, como funciona e quais as vantagens*. 2018. Disponível em: <https://www.opus-software.com.br/node-js/>.

LI, S. *Building a Collaborative Filtering Recommender System with ClickStream Data*. 2019. Disponível em: <https://towardsdatascience.com/building-a-collaborative-filtering-recommender-system-with-clickstream-data-dffc86c8c65>.

LINDEN, G.; SMITH, B.; YORK, J. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, v. 7, n. 1, p. 76–80, 2003.

MELVILLE, P.; SINDHWANI, V. Recommender systems. In: _____. *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer US, 2017. p. 1056–1066. ISBN 978-1-4899-7687-1. Disponível em: <https://doi.org/10.1007/978-1-4899-7687-1_964>.

MOHAMMED, C. M.; ZEEBAREE, S. R. Sufficient Comparison Among Cloud Computing Services: IaaS, PaaS, and SaaS: A Review. *International Journal of Science and Business*, v. 5, n. 2, 2021. Disponível em: <https://ideas.repec.org/a/aif/journl/v5y2021i2p17-30.html>.

NEUHAUS, J. *Angular vs. React vs. Vue: A 2017 comparison*. 2017. Disponível em: <https://medium.com/pixelpassion/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176>.

OVERFLOW, S. *Developer Survey Results 2019*. 2019. Disponível em: <https://insights.stackoverflow.com/survey/2019>.

OVERFLOW, S. *2020 developer survey*. 2020. Disponível em: <https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers>.

RAIGOZA, J.; KARANDE, V. A study and implementation of a movie recommendation system in a cloud-based environment. *Int. J. Grid High Perform. Comput.*, IGI Global, USA, v. 9, n. 1, jan 2017. ISSN 1938-0259. Disponível em: <https://doi.org/10.4018/IJGHPC.2017010103>.

RASHID, Z. N. et al. Distributed cloud computing and distributed parallel computing: A review. In: *2018 International Conference on Advanced Science and Engineering (ICOASE)*. [S.l.: s.n.], 2018.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: _____. *Recommender Systems Handbook*. Boston, MA: Springer US, 2011. p. 1–35. ISBN 978-0-387-85820-3.

ROCHE, P. *The next software disruption: How vendors must adapt to a new era*. 2020. Disponível em: <https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-next-software-disruption-how-vendors-must-adapt-to-a-new-era.html>.

SCHEIN, A. et al. Methods and metrics for cold-start recommendations. In: . [S.l.: s.n.], 2002. p. 253–260.

SEETHAMRAJU, R. Determinants of saas erp systems adoption. *Proceedings - Pacific Asia Conference on Information Systems, PACIS 2013*, 01 2013.

SUN, W. et al. Software as a service: An integration perspective. In: . [S.l.: s.n.], 2007. p. 558–569. ISBN 978-3-540-74973-8.

SUSHIL, B.; LEENA, J.; SANDEEP, J. An approach for investigating perspective of cloud software-as-a-service (saas). *International Journal of Computer Applications*, v. 10, 11 2010.

TEAMR, A. *How Much Does It Cost To Build A Recommendation System*. 2022. Disponível em: <https://azati.ai/recommendation-system-development-costs/>.

TILKOV, S.; VINOSKI, S. Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, v. 14, n. 6, 2010.

WATERS, B. Software as a service: A look at the customer benefits. *Journal of Digital Asset Management*, v. 1, 2005.

ZHOU, R.; KHEMMARAT, S.; GAO, L. The impact of youtube recommendation system on video views. In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: Association for Computing Machinery, 2010. (IMC '10). ISBN 9781450304832. Disponível em: <https://doi.org/10.1145/1879141.1879193>.