UNIVERSIDADE DO SUL DE SANTA CATARINA

DIEGO PRESTES DE SOUSA

SISTEMA DE RECOMENDAÇÃO DE FILMES UTILIZANDO FILTRAGEM COLABORATIVA

Palhoça

2023

DIEGO PRESTES DE SOUSA

SISTEMA DE RECOMENDAÇÃO DE FILMES UTILIZANDO FILTRAGEM COLABORATIVA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Flávio Ceci, Dr.

Palhoça

2023

DIEGO PRESTES DE SOUSA

SISTEMA DE RECOMENDAÇÃO DE FILMES UTILIZANDO FILTRAGEM COLABORATIVA

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel em Sistemas de Informação e aprovado em sua forma final pelo Curso de Graduação em Sistemas de Informação da Universidade do Sul de Santa Catarina.

(Local), (dia) de (mês) de (ano da defesa).

Professor e orientador Flávio Ceci, Dr.
Universidade do Sul de Santa Catarina

Prof. Aran B. T. Morales, Dr.
Universidade do Sul de Santa Catarina

Prof. Saulo P. Zambiasi, Dr.
Universidade do Sul de Santa Catarina

Dedicado a virtude nascida do virtuoso e não do covarde. Dedicado a todos que atribuem o “Eu” as suas próprias ações e hábitos.

AGRADECIMENTOS

As melhores conversas que tive foram com aqueles que já morreram e hoje vivem em seus livros, os quais visito constante e diariamente. Agradeço por me ensinarem a viver:

Zenão de Cítio, o primeiro estoico que transformou a maior tragedia de sua vida em seu maior triunfo. Lucius Annaeus Seneca, Marcus Aurelius Antoninus e Epicteto que me ensinaram a viver com coragem, justiça, autocontrole e sabedoria. À Sócrates e Cato, o jovem, que me ensinaram que filosofia não serve de nada lida ou escrita, ela deve ser vivida.

Sigmund Freud e Carl Gustav Jung que me ensinaram o valor psicológico dos meus sonhos e seus símbolos, como enfrentar a mim mesmo e como me tornar não perfeito, nem mesmo bom, mas completo – compreendendo que: tudo que me incomoda nos outros me leva a uma melhor compreensão de mim mesmo.

Charles Bukowski, que me ensinou que as vezes você fica tão sozinho que até faz sentido, o charme que há em ser extraordinariamente ordinário e ao mesmo tempo, uma peça única – e como encontrar o que eu amo e deixar isso me matar, pois todas as coisas irão me matar, mas é muito melhor ser morto por algo que se ama.

Friedrich Nietzsche, que me ensinou a enxergar a filosofia que há em meu próprio corpo que nenhum livro poderia descrever, o valor na solitude e na boa companhia que sou a mim mesmo. O valor que existe na amoralidade, como resistir aos fardos como um camelo, me libertar de “obrigações morais” impostas como um leão e, como uma criança, criar valores a partir de mim como se tudo fosse novo, e novamente como um recém-nascido ser puro potencial. E que aquilo que se faz por amor está sempre além do bem e do mal.

Filosofia é a cura para o niilismo da atualidade. Drogas e pílulas amortecem os efeitos, mas é a filosofia que cura o tédio, a “vontade do nada” e a construção de um caráter virtuoso baseando-se em seus próprios termos. E a ti, filosofia, eu agradeço.

Aos vivos: agradeço a minha mãe pelo exemplo de resiliência e a me ensinar a ser feliz pelo simples fato de continuarmos vivos. E o professor Flavio Ceci, especialmente pela influência positiva na minha vida como desenvolvedor, acadêmico e humano, tal influência molda o caráter de um jovem – especialmente quando você precisa de um exemplo vívido de como um homem adulto poderia ser se este for tudo o que ele puder.

“Um pensador vê suas próprias ações como experimentos e perguntas – como tentativas de descobrir algo. Sucesso e fracasso são para ele respostas acima de tudo.”

(FRIEDRICH NIETZSCHE, 1882).

RESUMO

A contínua expansão acelerada da internet aumenta significativamente a necessidade de sistemas de recomendação eficazes para filtrar a abundância de informações disponíveis. Este trabalho propõe uma revisão bibliográfica abrangente sobre os principais métodos e abordagens utilizados em sistemas de recomendação. Por meio dessa revisão é identificado as técnicas mais comumente empregadas, com um foco maior nas técnicas ligadas a filtragem colaborativa. A partir da revisão bibliográfica, é desenvolvido um protótipo de sistema de recomendação utilizando uma arquitetura modular e flexível. O protótipo incorpora um modelo de recomendação customizado, juntamente com técnicas de pré-processamento e pós-processamento de dados. Além disso, o protótipo possui uma interface intuitiva que permite aos usuários interagir e fornecer *feedback* sobre as recomendações recebidas. Para avaliar a eficácia do protótipo, é conduzido um estudo qualitativo com um grupo de usuários representativos. Os participantes foram solicitados a utilizar o sistema e fornecer *feedback* sobre a qualidade e a relevância das recomendações recebidas. Os resultados qualitativos do estudo revelaram que o protótipo apresentou um desempenho promissor na geração de recomendações relevantes e úteis para os usuários. Este trabalho contribui para o campo dos sistemas de recomendação, oferecendo uma revisão abrangente e as vezes multidisciplinar da literatura e apresentando um protótipo funcional. Os resultados qualitativos obtidos fornecem *insights* valiosos para futuros desenvolvimentos nessa área, permitindo aperfeiçoar a precisão e a personalização dos sistemas de recomendação.

Palavras-chave: Sistema de recomendação. Filtragem colaborativa. Preferência.

ABSTRACT

The continuous and accelerated expansion of the internet significantly increases the need for effective recommendation systems to filter the abundance of available information. This work proposes a comprehensive literature review on the main methods and approaches used in recommendation systems. Through this review, the most commonly employed techniques are identified, with a greater focus on collaborative filtering techniques. Based on the literature review, a prototype recommendation system is developed using a modular and flexible architecture. The prototype incorporates a customized recommendation model, along with data preprocessing and post-processing techniques. Additionally, the prototype has an intuitive interface that allows users to interact and provide feedback on the received recommendations. To evaluate the effectiveness of the prototype, a qualitative study is conducted with a group of representative users. The participants were asked to use the system and provide feedback on the quality and relevance of the received recommendations. The qualitative results of the study revealed that the prototype showed promising performance in generating relevant and useful recommendations for users. This work contributes to the field of recommendation systems by offering a comprehensive and sometimes multidisciplinary review of the literature and presenting a functional prototype. The qualitative results obtained provide valuable insights for future developments in this area, enabling the improvement of accuracy and personalization of recommendation systems.

Keywords: Recommender Systems. Collaborative filtering. Preference.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A indústria cinematográfica é uma das maiores no ramo do entretenimento e uma das mais relevantes da atualidade. Se a algum tempo atrás a internet era temida por produtores de conteúdo cinematográfico por conta da pirataria, agora se mostra a maior das aliadas: revolucionando a maneira como o conteúdo produzido é distribuído com os serviços de *streaming*. Segundo o artigo de David Curry (2022) a indústria de *streaming* cinematográfica em 2021 faturou 72,2 bilhões de dólares, e tem projeções de faturar 115 bilhões de dólares em 2026. Sem dúvida os serviços de *streaming* têm impulsionado o mercado cinematográfico, principalmente por conta da variedade de conteúdo que podem fornecer ao consumidor final e facilidade de acesso ao material ofertado por esses serviços.

Segundo Amatriain (2013, p. 1, tradução própria):

> Sistemas de recomendação são um excelente exemplo da principal aplicabilidade da mineração de dados em grande escala. Aplicações de *e-commerce*, algoritmos de procura como Google, aplicações de músicas como Spotify e vídeos como Youtube, jogos ou até mesmo aplicativos de namoro online como Tinder, fazem uso de técnicas semelhantes para minerar grandes volumes de dados para melhor atender às necessidades de seus usuários de maneira personalizada.

Filtragem colaborativa refere-se à tarefa de prever as preferências de um determinado "usuário" para alguns "objetos" (como, por exemplo, livros, músicas, produtos, etc.) baseando-se em suas preferências passadas – bem como as preferências de outros usuários. Como bem colocaram Abernethy et al (2009, p. 2, tradução própria):

> [...] em um sistema de recomendação de livros, por exemplo, pode-se sugerir novos livros para um novo usuário com base no que ele e outros usuários compraram e/ou avaliaram recentemente. O objetivo final da filtragem colaborativa é inferir as preferências dos usuários para oferecer-lhes novos objetos.

Quando a filtragem colaborativa é implementada como solução para um sistema de recomendação é necessário assumir que o usuário provavelmente irá gostar de coisas que outros usuários com a mesma inclinação àquele mesmo tópico gostam. Essa ideia foi popularizada por Koren, Bell e Volinsky (2009, p. 3, tradução própria) afirmando na seguinte passagem: "Uma grande vantagem da filtragem colaborativa é que ela é livre de domínio, e ainda pode abordar aspectos de dados que muitas vezes são ilusórios e difícil de aplicar a um perfil de usuário utilizando filtragem de conteúdo. [...]"

1.1 PROBLEMÁTICA

Como explicitado no artigo de Clark da *Business Insider* (2020) sobre a dinamicidade do número de títulos disponíveis na plataforma de *streaming* de filmes, em 2012 a *Netflix* possuía 11 mil títulos no total entre filmes e séries, em 2015 a *Netflix* eram aproximadamente 5.769 títulos e em 2020 esse número é de 5.838, com 1.097 dessas obras sendo originais, produzidas pela própria *Netflix*. E enquanto a biblioteca do serviço de filmes e séries diminuiu, o número de usuários utilizando seus serviços só aumentou, como demonstrado no artigo de Stoll (2022), que demonstra o crescimento do número de usuários de 2013, onde a *Netflix* tinha 34.24 milhões de usuários inscritos, até o primeiro quartil de 2022 – onde esse número pulou para 220.67 milhões. Muitos fatores influenciam nessa tomada de decisão em relação a quantidade de títulos disponíveis na plataforma, entre os principais está o fato de que com o objetivo de oferecer filmes e séries para o usuário que ele possa aproveitar é necessária uma filtragem melhor e mais densa do acervo, e mesmo assim a chance de o usuário não conseguir decidir o que assistir e acabar não assistindo nada é maior. Por muito tempo a *Netflix* foi conhecida por isso: os usuários passam mais tempo procurando algum filme para assistir e rolando a listagem de filmes do que realmente assistindo os títulos disponível na plataforma, a Figura 1 representa bem o humor gerado pela sensação que o usuário tem de “possuir tantas opções, e não saber o que escolher”:

Figura 1 – exemplo de piada com ironia comumente feita em redes sociais sobre a Netflix

Fonte: Autoria própria (2022)

Observando que serviços de *streaming* de filmes e séries costumam ter um acervo gigantesco, e em meio a tantas opções – como recomendar o melhor conteúdo possível para o usuário final? Essa resposta envolve um palpite sobre o futuro, que inevitavelmente precisará considerar a imprevisibilidade do gosto pessoal de cada indivíduo, que está em constante transformação na medida em que o tempo passa.

Segundo Koren, Bell e Volinsky (2009) estratégias para sistemas de recomendação baseadas em conteúdo necessitam de coleta de dados externa, e no caso desta monografia é necessário realizar a coleta das preferências dos usuários em relação aos conteúdos cinematográficos e determinar, probabilisticamente, quais filmes e séries serão atrativos para quais usuários – apesar de que as vezes essas informações não serem tão fáceis de coletar.

De forma geral, existem três estratégias principais para abordar um sistema de recomendação: filtragem de conteúdo, filtragem colaborativa e filtragem híbrida. Este trabalho propõe-se a implementar um sistema de recomendação de filmes baseado em filtragem colaborativa.

1.2 OBJETIVOS

Nesta seção estão os objetivos geral e específico que serviram como guia no desenvolvimento desta monografia.

1.2.1 Objetivo geral

Desenvolver um sistema de recomendação de filmes a partir do uso da filtragem colaborativa.

1.2.2 Objetivo específico

Para obtenção do objetivo geral são necessários os seguintes objetivos específicos:

- Identificar quais as técnicas ligadas a filtragem colaborativa podem ser aplicadas para a construção de um sistema de recomendação;
- Desenhar uma proposta de solução a partir dos requisitos do sistema;
- Construir uma base de dados com a opinião de usuários reais sobre múltiplos filmes;
- Desenvolver um protótipo funcional para materializar a proposta de solução;

- Avaliar o resultado obtido a partir da aplicação de um questionário para algumas pessoas.

1.3 JUSTIFICATIVA

A presente monografia pode ser justificada pela altíssima demanda de sistemas de recomendação eficazes no que se propõe a fazer: recomendar um grupo de itens específicos a um grupo de usuários específicos. Segundo Ricci, Rokach e Shapira (2015) os sistemas de recomendação desempenham um papel importante em sites da internet altamente cotados, como Amazon, YouTube, Netflix, Yahoo, Tripadvisor, Last.fm e IMDb – e esse exemplo citado é de 2010, antes da sociedade estar acostumada a redes sociais como maiores canais de comunicação do ocidente, aplicativos como Instagram e YouTube estão sempre aprimorando seus sistemas de recomendação para atender uma demanda cada vez maior de usuários. Além disso, muitas empresas de mídia estão desenvolvendo e implantando sistemas de recomendação como parte dos serviços que prestam aos seus assinantes. Por exemplo, a Netflix – já quando era conhecida como uma empresa de aluguel de DVD online, e não um serviço de *streaming* – concedeu um prêmio de um milhão de dólares a equipe que primeiro conseguiu melhorar substancialmente o desempenho de seu sistema de recomendação.

Segundo Lü (2012, p. 3, tradução própria):

> Graças aos custos cada vez menores de armazenamento e processamento de dados, os sistemas de recomendação se espalham gradualmente para a maioria das áreas da vida humana. Os vendedores observam atentamente as compras do usuário para recomendar outros produtos e aumentar as vendas, sites sociais analisam os contatos do usuário para ajudá-lo a conectar-se com novos amigos e ficar viciados no site, e estações de rádio online lembram músicas omitidas para servi-las melhor no futuro. Em geral, sempre que há muitos produtos diversos e os clientes não são iguais, a recomendação personalizada pode ajudar a entregar o conteúdo certo para a pessoa certa. [...].

A demanda por sistemas de recomendação e métodos de filtragem de conteúdo tem crescido exponencialmente, impulsionada pelo aumento constante do volume de dados na internet. Nesse contexto, a pesquisa e o desenvolvimento nessa área têm evoluído consideravelmente, com a proposição de diversos algoritmos e técnicas, abrangendo desde a filtragem colaborativa até a análise de conteúdo. No entanto, apesar dos estudos e

implementações existentes, é necessário aprofundar o conhecimento sobre as abordagens disponíveis por meio de uma revisão bibliográfica abrangente. Além disso, é fundamental traduzir esse conhecimento em um uso prático, desenvolvendo um protótipo funcional que permita a validação das técnicas estudadas. Por fim, é essencial realizar testes reais para obter resultados concretos e avaliar o desempenho e a eficácia das abordagens propostas. Dessa forma, justifica-se a presente pesquisa, visando preencher essa lacuna e contribuir para o avanço da área de sistemas de recomendação e filtragem de conteúdo.

1.4 ESTRUTURA DA MONOGRAFIA

Esta monografia é composta pelos seguintes capítulos:

- Capítulo 1: Apresenta a introdução, com a definição do problema abordado, seus objetivos e justificativa.
- Capítulo 2: Apresenta a fundamentação teórica que norteia a presente monografia.
- Capítulo 3: Metodologia aplicada no trabalho.
- Capítulo 4: Apresenta a proposta de solução.
- Capítulo 5: Demonstra o desenvolvimento de *software*.
- Capítulo 6: Conclui o presente trabalho e apresenta possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Se um sistema de recomendação tem como objetivo palpitar a preferência de um usuário sobre determinado tópico – mesmo que muitas vezes o próprio usuário não possua consciência sobre suas preferências, é importante destacar o conceito por trás da geração – ou descobrimento – de novas preferências ao longo do tempo. Este capítulo tem como objetivo introduzir propriamente os conceitos teóricos que regem o desenvolvimento da presente monografia, sendo estes: preferências e como elas se formam e se modificam no indivíduo; sistemas de recomendação; e uma breve revisão sobre mineração de dados, ainda no contexto de sistemas de recomendação.

2.1 PREFERÊNCIAS

A teoria da escolha racional estuda como o indivíduo pode racionalizar seus próprios atos, dito que o indivíduo em questão possui preferências, mas pouco se tem nessa teoria sobre de onde essas preferências vêm. Para Dietrich e List (2012) na teoria da escolha racional as preferências são assumidas como fixas e vistas de maneira exógena. E por mais que essa afirmação não seja necessariamente incorreta, ainda assim não há um esforço na compreensão da "origem da vontade" – por assim dizer – no que se diz nessa teoria. Com questionamentos como estes, e em busca de uma maior compreensão de como ideias, preferências e vontades se formam surgiu o tópico de formação de preferência.

2.1.1 Como as preferências surgem

Por que as pessoas fazem o que fazem? Uma resposta fácil seria que "as pessoas fazem o que elas querem"; mas neste caso, por que as pessoas querem o que querem? A

compreensão da vontade humana tem fascinado intelectuais por séculos, principalmente nas áreas da ciência económica e política.

Como Newell et al. (1990, p. 42, tradução própria) afirma:

> As teorias da cognição humana são, em última análise, teorias de aspectos dos sistemas físico e biológico. A capacidade humana de descrever a própria cognição de tal forma e não de outra forma se baseia sobre tudo na natureza física e biológica de todos os serem humanos. Além disto, o fato de que os seres humanos estão fundamentados no mundo implica restrições adicionais que devem ser levadas em conta na construção de suas próprias teorias.

Com tais afirmações é necessário desmistificar qualquer interpretação da palavra "preferência" nesta monografia por si só. Segundo Druckman e Lupia (2000) não há uma forma global e singular de se decidir qual a definição da palavra preferência, até porque palavras como "gosto" ou até "inclinações" são válidas; mas para o caso de uso dessa monografia em particular a palavra preferência assumirá a definição dada por Druckman e Lupia (2000) onde "preferência" é definida como uma avaliação comparativa de um conjunto de objetos, ou seja, uma classificação sobre tal preferência. A preferência serve como marcador cognitivo que lembra a pessoa como interagir com aspectos de seu ambiente. Portanto, se determinado indivíduo prefere a cor vermelha sobre azul, este indivíduo identifica aspectos no seu ambiente que, em seu próprio raciocínio, faz com que vermelho promova maiores benefícios do que outras cores como azul.

2.1.2 Objetos de preferência

"Com o objetivo de modelar como o indivíduo forma e revisa suas preferências sobre um conjunto não vazio de X objetos de preferência. [...]" (DRIETRICH; LIST, 2012, p. 3, tradução própria). Ainda no mesmo parágrafo os autores continuam: afirmando que esse conjunto pode ser um grupo de objetos físicos como peças de roupa, ou diferentes sofás, por exemplo; ou um amontoado de ideias que se agrupam num contexto específico, que apesar de abstrato e as vezes difícil de visualizar, têm o mesmo impacto preferencial a um nível cognitivo – ou ainda, Para Druckman e Lupia (2000), que colocaram essa compreensão de forma clara dando o exemplo do sapato e tofu: sapatos e tofu são substitutos de vários domínios, onde muitas pessoas preferem vestir sapatos, outras na área da alimentação preferem comer tofu. Se

uma pessoa não se imagina comendo um sapato ou vestindo tofu, então esses objetos de preferência não pertencem ao mesmo domínio preferencial, ou seja, sapatos não pertence ao mesmo conjunto de preferências que o tofu, e nem tofu ao conjunto preferencial dos sapatos.

É importante observar também que as experiências pessoais do indivíduo podem moldar a percepção deste indivíduo sobre um conjunto de preferências, como colocou Druckman e Lupia (2000) "[...] por exemplo, algumas pessoas preferem cachorro ao invés de gato como *pets*. Alguns ainda vão além e distinguem entre diferentes tipos de cachorros e gatos, onde outros indivíduos são indiferentes quanto a 'raça' do cachorro ou gato." E isso se observa nitidamente em alguns tópicos: algumas pessoas preferem margueritas ou tulipas, mas outras simplesmente "gostam de flores"; pessoas mais distantes do cenário político ou com uma visão limitada sobre o assunto costumam afirmar que "político é tudo ladrão", enquanto outros com olhos mais atentos ao cenário político e as mudanças de posição desse meio volátil costumam ter opiniões formadas baseadas no passado e presente dos representantes aos cargos públicos.

Em curta, estudando os trabalhos *Preference theory* (DYER; JIA, 2013), *Preference Formation* (DRUCKMAN; LUPIA, 2000) e *Where do preferences come from?* (DIETRICH; LIST, 2012), os objetos de preferência não estão simplesmente no mundo à espera da percepção humana para serem ranqueados e "preferidos". Estes objetos são, na verdade, objetos que a capacidade humana perceptiva teve a capacidade de se permitir diferencia-los num grupo específico de objetos que são não só substituíveis entre si, mas comparáveis e – mesmo que de forma abstrata e inconsciente – relativamente ranqueáveis, ao menos para a percepção do indivíduo. As experiências do ser humano com esses objetos fornecem incentivo para diferenciar e relembrar e associar tais objetos de preferência com a capacidade cognitiva do indivíduo, para que então este possa identificar e avaliar os objetos sob sua percepção.

2.1.3 De onde vêm as avaliações comparativas de preferência

Como Clark (1997, p. 81, tradução própria) colocou em sua obra *Being There: Putting Brain, and World Together Again*:

> A natureza [...] é fortemente vinculada pelas soluções alcançadas para problemas encontrados anteriormente. Como resultado, as novas vestimentas cognitivas

> raramente são feitas de tecido inteiro; geralmente incluem emendas feitas às pressas a velhas estruturas e estratégias.

Como defendido por Richard B. Brandt (1998, p. 63) em seu trabalho "*The Rational Criticism of Preferences*", contido numa coleção de ensaios chamado *Preferences* (GRUYTER, 1998), onde o autor aborda o criticismo racional das preferências na natureza do indivíduo: "define-se "preferência" como "querendo mais". Psicólogos concordam que o desejo por um tipo de evento aumenta se o tipo de evento é associado a eventos agradáveis no passado, condicionado pela contiguidade". Nesse contexto, o termo "criticismo" normalmente é empregado de forma negativa, mas no caso do texto de Brandt isso não necessariamente implica uma experiência ruim, mas sim implica a racionalização do indivíduo sobre os eventos como uma reflexão – a um nível consciente ou não – dos eventos experienciados no passado. No mesmo parágrafo, Brandt continua: afirmando que eventos como fome e sede não funcionam da mesma forma, pois tais necessidades são ajustadas por desequilíbrios químicos no corpo.

> [...] muitos eventos são tidos como prazerosos por uma questão evolutiva; se estes não eram agradáveis, portanto, o tipo de evento agradável desejado, os indivíduos não sobreviveriam. Essa conexão – evento agradável, sendo desejado do condicionamento clássico, e, portanto, preferência – abre o caminho para a crítica racional. (BRANDT, 1982, p. 63, tradução própria)

2.1.4 Mudança de preferência e formação de preferência

Brandt (1998) afirma que um indivíduo modifica sua preferência sobre determinado objeto diante de um dos seguintes: (1) representação inadequada dos fatos; (2) influência de um estado motivacional temporário; (3) estimulo de generalização de casos anormais; (4) ignorar fatos desagradáveis sobre o objeto; (5) não fazer descriminações; (6) como resultado de sugestões de professores; e (7) resultado de crenças factuais falsas ou injustificadas. Brandt ainda sugere que "[...] a preferência foi racionalmente criticada caso a reflexão de um desses defeitos resultou na modificação de uma preferência.

Segundo Druckman e Lupia (2000) é importante destacar que "mudança de preferência" significa uma de duas coisas – a primeira sendo que o indivíduo pode alterar a própria preferência entre os objetos A e B, pois este obteve uma nova informação sobre, pelo

menos, um desses objetos; ou segundo: este indivíduo decide que os objetos antes conhecidos como um conjunto P são melhor tratados como objetos distintos, tornando-se conjunto P1 e P2. Um exemplo disso são cachorros: o indivíduo pode adorar quaisquer cachorros que sejam de pequeno e médio porte, mas descobriu – ou ainda, decidiu – que cachorros de grande porte "não são a mesma coisa". Então se anteriormente ele adorava o objeto P que define cachorros de modo geral, agora ele os distingue tendo como P1 cachorros de pequeno de pequeno e médio porte, e P2: cachorros de grande porte.

> Em ambos os casos, novas informações sobre o atributo do objeto causam a mudança nas avaliações comparativas realizadas [pelo indivíduo] sobre um conjunto de objetos. Preferências sobre "novos" objetos são exemplos relacionados ao segundo tipo de mudança [na preferência]. Entretanto, quando o indivíduo encontra estímulos que são novos à própria percepção, este indivíduo instantaneamente forma convicções sobre estes estímulos. Esses estímulos – e a preferência que as seguem – não são recém-criadas; ao invés disso, estas [preferências] emergem de preferências e convicções que o próprio indivíduo possui sobre objetos com os quais interagiu no passado. Estes objetos selecionados do passado são aqueles que o indivíduo tem como similar com o novo estímulo que acabou de presenciar. (DRUCKMAN; LUPIA, 2000, p. 6, tradução própria)

Holland, Thagard e Nisbett (1986) esclarecem as condições sob qual tais mudanças são mais suscetíveis a acontecer, e as consequências de tais mudanças. O processo literal que eles descrevem consiste em regras de indução, mas essas regras são equivalentes a condições de mudança de convicção:

"Dois tipos importantes de condições para o desencadeamento são a falha de uma previsão e a ocorrência de algum evento incomum" (HOLLAND; THAGARD; NISBETT, 1986, p. 80, tradução própria)

"Uma nova regra é construída a partir daquela que já existe (a) aumentando a condição da regra existente com propriedades incomuns adicionais do contexto de falha e (b) substituir o resultado inesperado como a ação da nova regra." (HOLLAND; THAGARD; NISBETT, 1986, p. 88, tradução própria)

E ainda segundo Holland, Thagard e Nisbett a concorrência dessas preferências entre si favorecerá aquelas regras que: "(a) fornecem uma descrição da situação atual; (b) tem um histórico de utilidade para o sistema; (c) produzem o maior grau de completude da descrição; (d) tem o maior grau de compatibilidade com outras informações atualmente ativas". (HOLLAND; THAGARD; NISBETT, 1986, p. 49, tradução própria), e vale ressaltar que a palavra "sistema" no contexto de Holland, Thagard e Nisbett é utilizada para se referenciar ao sistema decisório de preferencias do ser humano.

2.2 SISTEMAS DE RECOMENDAÇÃO

Como bem disse Resnick e Varian (1997) as vezes é necessário fazer escolhas sem experiência suficiente e decidir o que se prefere. No dia a dia as pessoas de modo geral dependem de recomendações de outras pessoas: seja de boca em boca, cartas de recomendação, críticas de filmes e livros ou simples pesquisas de satisfação e *scores* que medem a qualidade de determinado produto ou serviço. Ainda nas colocações de Resnick e Varian (1997), sistemas de recomendação auxiliam e fomentam esse processo natural e social de recomendações. Em um sistema de recomendação típico os usuários do sistema podem inserir recomendações como a entrada dos dados, digamos: "o filme é muito bom e intrigante", ou algo mais abstrato como uma carinha feliz após assistir um filme completo num serviço de *streaming* de filmes. O sistema então agrega as recomendações e direciona estas para recipientes apropriados, que no caso seria o usuário que necessita da recomendação em dado momento. As vezes o processo primário de transformação desses dados é a agregação; outras vezes o valor real de tal sistema está na habilidade de formar boas combinações entre os itens disponíveis para recomendação e aqueles que precisam da recomendação – o segundo caso é o ideal para o trabalho proposto, visto que para os padrões dos sistemas que recomendam atuais, agregar alguns dados de forma não dinâmica e sem considerar para quem se está recomendando é, em alguns casos, "arcaico".

O primeiro sistema de recomendação veio muito antes da Amazon ou da Netflix; como descrito por Huttner (2009), a Tapeçaria (nome original: *Tapestry*), desenvolvida pela *Xerox Palo Alto Research Center* e descrita em uma edição de 1992 da "*Communications of the ACM*" tinha como motivação o crescimento do sistema de e-mail eletrônico, no qual resultava em "usuários sendo sobrecarregados de documentos". A primeira tentativa para reduzir a quantidade de e-mails foi "providenciar uma lista de e-mails e permitindo que os usuários se inscrevessem apenas àquelas que estes usuários tinham interesse.", no entanto essa abordagem não teve sucesso prático: as preferências dos usuários dificilmente correspondiam as listagens de e-mail disponíveis.

Huttner (2009) ainda descreve a segunda e mais bem sucedida tentativa desse sistema de filtragem de e-mails: um filtro pessoal para cada usuário, e o usuário só receberia e-mails que se enquadrassem no filtro definido. Mais especificamente, esse sistema iria permitir a filtragem colaborativa, que nesse caso significaria que "as pessoas colaborariam para ajudar outras pessoas a efetuarem suas filtragens gravando as reações das pessoas a esses

documentos". O nome formal dado pelo sistema Tapeçaria a essas reações era "anotações". Uma analogia feita pelo autor Huttner é que essas anotações são o equivalente as *tags* que são comummente implementadas em sites de filmes e comércios virtuais. Então se determinado e-mail faz o usuário dar risadas, este usuário atribui a *tag* de "engraçado", por exemplo.

Supondo que o usuário Giovana colocou a *tag* "engraçado" em um e-mail, e os usuários Diógenes, Marcos e Nicolau possuem em sua caixa de entrada um filtro de e-mails engraçados, a ideia é que estes usuários possuem um gosto similar – e baseado nesse princípio era possível ter uma noção do que os usuários ativos do serviço de filtragem de e-mails preferiam, então se a Giovana gostasse de outras coisas, adicionando outras *tags* em outros e-mails, era provável que Diógenes, Marcos e Nicolau tivessem interesse nesse tópicos também.

Como definido por Adomavicius e Tuzhilin (2005), uma definição mais formal que explica o problema de recomendação pode ser formulada da seguinte maneira:

Seja *C* o conjunto de usuários, *S* seja o conjunto de todos os itens disponíveis para recomendação, onde tanto o espaço de *S* quanto espaço o espaço de *C* pode ser grande – podendo chegar aos milhões –. E seja *u* uma função de utilidade que mede a utilidade do item *s* para o usuário *c*, ou seja, $u: C \times I \rightarrow R$, onde *R* é o conjunto ordenado – por exemplo, um conjunto de números não negativos ou reais dentro de determinado intervalo. Esta ordenação existe para oferecer os itens do mais relevante ao menos relevante para o usuário. Então, para cada usuário $c \in C$, o objetivo é escolher tais itens $s' \in S$ que maximize a utilidade desse item. Ainda nas colocações de Adomavicius e Tuzhilin (2005), podemos representar essa lógica da seguinte forma (Equação 1):

$$\forall c \in C, s'_c = \frac{arg\ max\ u(c,s)}{s \in S} \quad (1)$$

Segundo Ricci, Rokach e Shapira (2015, p. 1, tradução própria):

> "'Item' é o termo específico para denominar o que o sistema recomenda aos usuários. Um sistema de recomendação normalmente foca em tipos específicos de itens (por exemplo, CDs ou notícias) e, consequentemente, seu *design*, sua interface gráfica e a técnica principal de recomendação utilizada para gerar as recomendações [aos usuários] é completamente personalizado para promover sugestões úteis e eficazes para este tipo específico de item."

Ainda nas palavras de Adomavicius e Tuzhilin (2005): cada elemento do espaço de usuários *U* pode ser definido como um perfil que inclua várias características de dado usuário

como idade, gênero, salário, etc. e "[...] essencialmente o modelo que representa o usuário [no sistema] representa suas preferências e necessidades" (RICCI; ROKACH; SHAPIRA, 2015, p. 2, tradução própria). Perfis de usuário são importantíssimos para o processo de recomendação. As informações sobre perfis de usuário podem ser coletadas de maneira implícita: armazenando informações sobre as ações do usuário para tentar perceber padrões de comportamento; ou explícita: quando o próprio usuário informa ao sistema sua preferência em relação aos objetos apresentados a ele, não se restringindo a usar apenas uma dessas abordagens na coleta de dados para perfis de usuário (PRIMO, 2013).

"O problema central de um sistema de recomendação está no fato de que utilidade (u) não costuma ser definido por todo espaço $C \times S$, mas sim por um subconjunto deste [espaço]. Ou seja, u precisa ser *extrapolado* para todo o espaço $C \times S$." (ADOMAVICIUS; TUZHILIN, 2005, p. 2, tradução própria)

Vale também pontuar o fato explicitado por Ricci, Rokach e Shapira em seu livro *Recommender Systems Handbook* (2015): as vezes a utilidade que um item tem para determinado usuário depende de outras variáveis, que é atribuído o nome genérico de "contextual". Por exemplo, o domínio de um usuário em um determinado campo do conhecimento influenciará a escolha deste: um guitarrista iniciante pode comprar qualquer guitarra e estar igualmente satisfeito, já alguém com mais proficiência com o instrumento tenha preferências mais especializadas como uma *Gibson Les Paul*; ou pode depender do momento que a recomendação foi solicitada: se o usuário deseja uma recomendação de refeição à noite provavelmente pedirá um lanche, pizza, sushi, etc... mas se o pedido será feito ao meio dia, provavelmente uma marmita o satisfará melhor; outro fato a se levar em consideração é localização, visto que, por exemplo, um sistema de recomendação de restaurantes não deveria recomendar para um usuário, digamos, que se encontra no sul do Brasil a fazer uma ótima refeição em Paris. Consequentemente, a recomendação deve ser adaptada aos detalhes adicionais específicos – e como resultado, a recomendação acaba se tornando mais difícil de estimar corretamente.

As funções internas de um sistema de recomendação são caracterizadas, segundo Bobadilla et al. (2013), pelo seu algoritmo de filtragem. Bobadilla ainda afirma que existem 4 métodos principais que dividem esse algoritmo: (a) filtragem colaborativa; (b) filtragem demográfica; (c) filtragem baseada em conteúdo; e (d) filtragem hibrida – mas outras literaturas como o livro de Ricci e Shapira *Recommender Systems Handbook* (2015) não chegam a citar a filtragem demográfica entre os principais métodos de filtragem, então serão abordadas as outras

três no decorrer desta sessão, destrinchando em especial e com maior minúcia a temática da presente monografia: filtragem colaborativa.

2.2.1 Filtragem baseada em conteúdo

Segundo Ricci, Rokach e Shapira (2015) sistemas de recomendação baseados em filtragem baseada em conteúdo aprendem a recomendar itens que são similares com aqueles que o usuário gostou no passado. A similaridade dos itens é calculada baseada nos atributos associados com o item comparado. Por exemplo: se um usuário deu uma nota positiva para um filme de comédia (ou melhor dizendo, com o atributo "comedia") então o sistema aprenderá que deve recomendar mais filmes do mesmo gênero cinematográfico, ou em outras palavras, com o atributo comédia.

Segundo Meteren e Someren (2000, p. 3, tradução própria):

> "[...] PRES (acrônimo para *Personalized Recommender System*, ou sistema de recomendação personalizado) é um sistema de recomendação com filtragem baseada em conteúdo. [O sistema] faz recomendações comparando o perfil do usuário com o conteúdo de cada documento na coleção. O conteúdo do documento pode ser representado por um conjunto de termos. Termos são extraídos de documentos executando várias etapas de análise [...]"

Meteren e Someren (2000) continuam: afirmando que a representação mais comum desse método de recomendação é o modelo de vetor no espaço:

No modelo de vetor no espaço um documento *D* se vê representado em um vetor multidimensional, onde cada dimensão corresponde a um termo distinto e *m* é o número total de termos usados em uma coleção de documentos. O vetor de documentos é escrito: onde w_i define o peso do termo t_i que indica a importância. Se documento *D* não contem o termo t_i então o peso w_i é igual a zero. Os pesos para os termos podem ser determinados usando um esquema de *tf-idf*. Nessa abordagem os termos são associados a um peso, e este peso é baseado na quantidade que aquele termo aparece em um documento em particular, e o quão frequente o termo aparece na coleção inteira de modo geral (Equação 2):

$$w_i = tf_i * \log\left(\frac{n}{df_i}\right) \quad (2)$$

Onde tf_i é definido pelo número de ocorrências do termo t_i no documento *D*, *n* é o total de documento na coleção e df_i é o número de documento em que o termo t_i aparece pelo menos uma vez.

É importante ressaltar que o exemplo dado no artigo de Meteren e Someren (2000) menciona análise bayesiana e bastante análise semântica, assuntos que estão fora do escopo deste trabalho científico, o que estamos observando é a metodologia como o exemplo dado pelos autores atribui um peso a um item, e com esse peso podemos calcular a recomendação para determinados perfis de usuários, seguindo o exemplo de Meteren e Someren (2000):

> O perfil do usuário consiste em um vetor que representa um tópico de interesse. Em muitos sistemas de filtragem de informação que usam o *feedback* [do usuário] como relevância o perfil do usuário acaba sendo representado por mais de um vetor. [...] o problema de usar mais de um vetor é que leva algum tempo até que um vetor represente um tópico suficientemente preciso [para o usuário].

Um documento sobre determinado tópico irá, portanto, nem sempre ser atrelado a um vetor de perfil de usuário sobre o mesmo tópico. "[...] quando o usuário passa um determinado tempo lendo um documento *D* o perfil do usuário *P* é atualizado utilizando conforme a seguinte equação (Equação 3):

$$P' = aP + \beta D \tag{3}$$

Os termos de determinado perfil que possuem valores muito baixos são removidos quando o vetor do usuário é atualizado. O peso β determina a importância relativa de um documento para o usuário. β pode assumir diferentes valores em diferentes métodos de filtragem de informações baseada em conteúdo, mas para o caso do exemplo atual esse valor será sempre *1*. O vetor de perfil de usuário é ajustado para uma diminuição dos interesses do usuário por *a*, um peso entre *0* e *1* reduz o valor dos termos no perfil, e este peso é ajustado via experimentação.

Sistemas de recomendação baseados somente na filtragem de conteúdo geralmente sofrem dos problemas de (a) limite de conteúdo para análise e (b) superespecialização (SHARDANAND, 1995). O Problema de limite de conteúdo para análise – como o próprio nome sugere – ocorre quando o sistema não possui informações suficientes sobre o perfil do usuário ativo ou do conteúdo dos itens disponíveis para servir a recomendação. Por exemplo, pode ser que por problemas de segurança de dados os usuários sejam impedidos de fornecer

determinadas informações pessoais, ou haja dificuldades ou custos elevados para extrair o conteúdo de um item disponível para recomendação. Outro problema é o fato do conteúdo de um item seja insuficiente por si só para determinar sua qualidade (RICCI; ROKACH; SHAPIRA, 2015).

> Superespecialização, por outro lado, é um efeito colateral da maneira como sistemas [de recomendação] baseados em filtragem baseada em conteúdo recomendam novos itens, onde a previsão da avaliação do usuário para um item é alta se esse item for semelhante a outros itens que o usuário gostou. (RICCI; ROKACH; SHAPIRA, 2015, p. 55, tradução própria).

Por exemplo, num sistema de recomendação de filmes, o usuário será recomendado a assistir filmes do mesmo gênero que ele assistiu anteriormente, ou que tenha os mesmos atores. Isso faz com que o sistema se limite a outras possibilidades que o usuário não foi recomendado, mas ainda assim iria gostar (RICCI; ROKACH; SHAPIRA, 2015).

O problema apresentado no exemplo dado por Ricci, Rokach e Shapira foi o motivo pelo qual houve um prêmio de um milhão de dólares ofertado pela *Netflix*, como já mencionado anteriormente, para quem criasse um sistema de recomendação que fosse confiável e que de fato fizesse o que se propõe a fazer, com o menor número de limitações possíveis.

2.2.2 Filtragem colaborativa

Já dito que filtragem colaborativa é uma abordagem que assume o fato de que o que um usuário em particular prefere provavelmente se equivale as preferências de outros usuários e vice-versa, contrastando com filtragem baseada em conteúdo que depende do conteúdo da informação.

Como explicitado anteriormente no artigo de Resnick et al. (1994) e Herlocker et al. (1999) grande parte dos algoritmos de filtragem colaborativa utiliza o método dos vizinhos mais próximos como forma de criar a sua função de utilidade. Para isso, um determinado número de usuários é recuperado baseando-se na similaridade com o usuário alvo. O método de “vizinhos mais próximos” também é abordado por Koren, Bell e Volinsky (2009), apesar do *paper* exaltar os modelos de fatoração de matrizes como superiores por permitirem a incorporação de informação adicional como *feedback* implícito, efeitos temporais e níveis de

confiança. Segundo Ricci, Rokach e Shapira (2015), além do método do vizinho mais próximo também existe o método de filtragem baseada em modelos.

2.2.2.1 Filtragem baseada no vizinho mais próximo

Filtragem colaborativa baseada no vizinho mais próximo pode ser divididos de duas abordagens em relação a maneira que realizam o processo de recomendação (RICCI, ROKACH e SHAPIRA, 2015) (PRIMO, 2013) (WANG, VRIES, e REINDERS, 2006):

- Usuário – Usuário;
- Item – Item;

2.2.2.1.1 Usuário – Usuário

Filtragem colaborativa baseada em uma lógica de Usuário – Usuário é o algoritmo mais utilizado para encontrar similaridades entre os usuários de um sistema. A abordagem de Usuário – Usuário baseada no vizinho mais próximo (ou em algumas literaturas, "vizinhança") procura avaliar quais usuários têm gostos mais similares – esta conclusão vem da avaliação desses usuários aos itens propostos a estes, se as avaliações são similares então os usuários tem gostos similares (RICCI; ROKACH; SHAPIRA, 2015) (PRIMO, 2013). O método pode ser descrito em três partes, descrito por Primo (2013, p. 38):

1. Avaliação da similaridade de um usuário em relação aos outros [usuários]. Nesta etapa é comum ser utilizado o coeficiente de correlação *Pearson* (KONSTAN; RIEDL, 1999) (SHARDANAND; MAES, 1995) (RESNICK et al., 1994), comparando os perfis de usuário, que nesse caso é representado por um vetor que possui os itens e as respectivas notas dadas por cada usuário a cada item;

2. Seleção de um conjunto de vizinhos. Neste caso, normalmente é estipulado um limiar ou um número máximo de vizinhos (usuários similares) a serem considerados para o cálculo da predição;

3. O cálculo da predição (função de utilidade) determina a nota que um usuário daria um item que não foi avaliado por este [usuário], mas que foi avaliado pelos seus vizinhos. Esta predição pode ser gerada por meio de uma média ponderada a partir das avaliações dos vizinhos [demonstrado em (RESNICK et al., 1994)].

Um exemplo comummente usado para ilustrar esse método é utilizar uma matriz para mostrar os usuários e a preferência destes usuários sobre determinados itens. Primo (2013) fornece um ótimo exemplo em sua monografia *Método de representação de conhecimento baseado em Ontologias para apoiar Sistemas de Recomendação Educacionais*:

> [...] Para esse exemplo, será utilizado o Quadro 1 como base de informações. Este quadro contém em suas colunas conjuntos de objetos, em suas linhas usuários e a sua interseção representa a avaliação (valores de 1 à 10) de um objeto por um usuário. Desta forma, a coluna 2 na linha 2 é interpretada como: Usuário 1 avaliou em nota 4 o objeto A. Considerando isto, descreve-se o passo a passo do mecanismo utilizado para estimar a avaliação do Usuário 1 para o objeto G.

Quadro 1 – exemplo da abordagem Usuário – Usuário na filtragem colaborativa

	A	B	C	D	E	F	G	H
Usuário 1	4			2	1	7		
Usuário 2	5	1	2	2	1	7	9	
Usuário 3	6	5	4	1	2		8	5
Usuário 4		2		3		1		2

Fonte: Primo: Método de representação de conhecimento baseado em Ontologias para apoiar Sistemas de Recomendação Educacionais (2013, p. 39).

Quadro 2 – Similaridade dos usuários com o Usuário 1

Usuário 2	+0,98
Usuário 3	+0,87
Usuário 4	-1

Fonte: Primo: Método de representação de conhecimento baseado em Ontologias para apoiar Sistemas de Recomendação Educacionais (2013, p. 39).

Como visto nas etapas do método descrito, o coeficiente de correlação *Pearson* é utilizado para traçar a correlação entre os perfis de usuário (Equação 4):

$$C_p\ (A,B) = \frac{\sum_{i=0}^{n}(A_i\ \times B_i) - \frac{\sum_{i=0}^{n}(A_i)\ \times \sum_{i=0}^{n}(B_i)}{n}}{\sqrt{\left(\sum_{i=0}^{n}(A_i^2) - \frac{\left(\sum_{i=0}^{n} A_i\right)^2}{n}\right) \times \left(\sum_{i=0}^{n}(B_i^2) - \frac{\left(\sum_{i=0}^{n} B_i\right)^2}{n}\right)}} \quad (4)$$

Onde $C_p(A,B)$ se refere ao coeficiente de correlação *Pearson* entre o usuário A e o usuário B, A_i representa uma nota específica de um usuário A, B_i uma nota específica de um usuário B e n representa o total de notas que ambos os usuários tem em comum. O resultado da equação é um valor que varia entre -1 e +1, sendo que -1 indica ausência de correlação e +1 simboliza uma forte correlação.

O exemplo dado por Primo (2013) pretende estimar a avaliação do Usuário 1 para o objeto G, para isto, o Usuário 1 serve como base para o processo de correlação. Vale também ressaltar que são consideradas para correlação apenas notas existentes em ambos os usuários. Desta forma, quando aplicada a equação entre Usuário 1 e o Usuário 2 nós temos (Equação 5):

$$C_p\ (A,B) = \frac{74 - \frac{14\ \times\ 15}{4}}{\sqrt{\left(70 - \frac{(14)^2}{4}\right) \times \left(79 - \frac{(15)^2}{4}\right)}} \approx 0.98 \quad (5)$$

Repetindo a aplicação dessa equação para os Usuários 3 e 4, explica Primo (2013), os resultados são apresentados no Quadro 2, onde a primeira coluna representa os usuários existentes e a coluna 2 representa o valor de correlação com o Usuário 1. Desta forma o Quadro representa: o usuário 1 possui 0,98 de correlação com o Usuário 2. Os cálculos deste coeficiente servirão de base para a função de utilidade expressa através de uma média ponderada (Equação 6):

$$F_u(A,\Theta) = \frac{\sum_{i=1}^{n}\left(C_p(A,B) \times A_i\right)}{\sum_{i=1}^{n} C_p(A,B)} \quad (6)$$

Nessa equação de utilidade, $F_u(A, \Theta)$ representa a possível avaliação de um usuário A para um objeto Θ, n é referente ao número de usuários que avaliaram um objeto Θ e $C_p(A, B)$ representa o coeficiente de similaridade entre o usuário A e usuário B. Levando em consideração os dados do Quadro 2, a estimativa de avaliação do Usuário 1 para o objeto G é, aplicando a Equação 6 (Equação 7):

$$F_u(A, \Theta) = \frac{(0{,}98 \times 9) + (0{,}87 \times 8)}{0{,}98 + 0{,}87} \approx 8{,}52 \quad (7)$$

Isto é, o Usuário 1 tenderá a avaliar o objeto de preferência *G* em 8,52. Lembrando que essa estimativa se dá devido as avaliações prévias deste mesmo objeto por usuários similares ao Usuário 1.

2.2.2.1.2 Item – Item

A abordagem da filtragem colaborativa baseada na lógica Item – Item, diferente da abordagem Usuário – Usuário, prever a nota de um usuário para um item baseado nas notas que o usuário deu para itens similares. Em tal abordagem, dois itens são similares se vários usuários do sistema deram notas similares a estes itens (RICCI; ROKACH; SHAPIRA, 2015).

Ricci, Rockach e Shapira fornecem um ótimo exemplo dessa abordagem em *Recommender Systems Handbook* (2015):

> [...] ao invés de consultar seus pares, Eric – usuário ficcional desse exemplo – determina se quer assistir "Titanic" baseando-se nos filmes que ele já assistiu. Então Eric percebe que as pessoas que assistiram Titanic deram notas similares para os filmes "Forrest Gump" e "Wall-E". Considerando que Eric gostou desses filmes ele então decide que também irá gostar de Titanic.

Formalizando esse exemplo podemos concluir que: dado $N_u(i)$ os itens mais parecidos ao item *i* classificados pelo usuário u. A taxa de previsão de u para i é obtida como uma média pesada das notas dadas pelo usuário u para os itens de $N_u(i)$ (Equação 8):

$$\hat{r}_{ui} = \frac{\sum_{j \epsilon N_u(i)} w_{ij}\, r_{uj}}{\sum_{j \epsilon N_u(i)} | w_{ij} |} \tag{8}$$

Continuando no exemplo de Ricci, Rockach e Shapira (2015): supondo que as previsões são feitas novamente com dois vizinhos próximos, e que os itens mais parecidos com "Titanic" são "Forrest Gump" e "Wall-E", com o peso da similaridade sendo, respectivamente, 0,85 e 0,75. Sendo que Eric deu as notas 5 e 4, respectivamente. A previsão de nota para Titanic seria (Equação 9):

$$\hat{r}_{ui} = \frac{0{,}85 \times 5 + 0{,}75 \times 4}{0{,}85 + 0{,}75} \simeq 4{,}53 \tag{9}$$

As diferenças nas escalas de classificação individuais dos usuários podem ser consideradas, normalizando as classificações com uma função h (Equação 10):

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{j \epsilon N_u(i)} w_{ij}\, r_{uj}}{\sum_{j \epsilon N_u(i)} | w_{ij} |}\right) \tag{10}$$

Além disso, Ricci, Rockach e Shapira (2015) afirmam que também pode-se definir uma abordagem baseada na classificação baseada em itens: nesse caso, os itens *j* avaliados pelo usuário *u* votam na classificação que será dada ao um novo item *i*, e esses votos são pesados pela similaridade que existe entre *i* e *j*. A versão normalizada e formalizada dessa abordagem pode ser observada na seguinte equação (Equação 11):

$$\hat{r}_{ui} = h^{-1}\left(\arg\max r \epsilon S' \sum_{j \epsilon N_u(i)} \delta\big(h(r_{uj}) = r\big) w_{ij}\right) \tag{11}$$

2.2.2.1.3 Usuário – Usuário vs. Item – Item

Há cinco critérios para avaliar qual melhor abordagem para determinada solução, estas são: (a) precisão, (b) eficiência, (c) estabilidade, (d) capacidade de justificação [para dada recomendação] (do inglês *justifiability*, ou "justificabilidade") e (e) acaso (DESROSIERS; KARYPIS, 2015). Na pesquisa de Desrosiers e Karypis (2015), efetuada no capítulo 2 de *Recommender Systems Handbook* esses critérios ficam:

(a) **Precisão:** Sobre precisão, para Desrosiers e Karypis (2015, p. 47, tradução própria):

> A precisão do método de recomendação baseado em vizinhos depende principalmente da relação entre o número de usuários e itens no sistema. A similaridade entre dois usuários em metodologias baseadas em usuário (Usuário – Usuário), o qual determina os vizinhos de tal usuário, costuma ser obtido através da comparação das avaliações deste usuário para os mesmos itens.

Dito que – para os propósitos desta análise – as avaliações estejam distribuídas uniformemente entre os itens (em casos de sistemas de recomendação reais é comum o usuário avaliar uma pequena porção dos itens consumidos). Considerando o Quadro 3, a média de usuários disponíveis como possíveis vizinhos uns dos outros é 650. Entretanto, o número médio de classificações comuns utilizadas para calcular as semelhanças é somente 1. Por outro lado, a abordagem Item – Item computa a similaridade de dois itens comparando as avaliações de determinado usuário sobre estes itens. Assumindo novamente uma distribuição uniforme das avaliações sobre os itens, nós temos um número médio de 99 vizinhos e uma média de avaliações similares computadas de 10 (DESROSIERS; KARYPIS, 2015, p. 47).

Quadro 3 – a média de vizinhos e avaliações utilizadas para computação das similaridades para os métodos de vizinho-mais-próximo baseadas em Usuário – Usuário e Item – Item

	Média vizinhos	Média avaliações
Usuário – Usuário	$(\lvert v\rvert - 1)\left(1 - \left(\frac{\lvert i\rvert - p}{\lvert i\rvert}\right)^{p}\right)$	$\frac{p^2}{\lvert i\rvert}$
Item – Item	$(\lvert i\rvert - 1)\left(1 - \left(\frac{\lvert u\rvert - q}{\lvert u\rvert}\right)^{q}\right)$	$\frac{q^2}{\lvert u\rvert}$

Fonte: Desrosiers e Karypis: *A Comprehensive Survey of Neighborhood-Based Recommendation Methods* (2015, p. 48)

Uma distribuição uniforme é assumida como média do número de avaliações por usuário $p = |R|/|U|$, e a média de avaliações por item se dá formalmente como $q = |R|/|I|$.

Quadro 4 – A complexidade dos métodos Usuário – Usuário e Item – Item, baseado em espaço e tempo, como funções de número máximo de avaliações por usuário p= 〖max〗 _u |I_u |, e máximo de avaliações por item q= 〖max〗 _i |U_i |, e o número máximo de vizinhos utilizado

	Espaço	Tempo	
		Treinamento	Online
Usuário – Usuário	$O(\|U\|^2)$	$O(\|U\|^2 p)$	$O(\|I\|k)$
Item – Item	$(\|I\|^2)$	$O(\|I\|^2 q)$	$O(\|I\|k)$

Fonte: Desrosiers e Karypis: *A Comprehensive Survey of Neighborhood-Based Recommendation Methods* (2015, p. 48)

No geral, um número pequeno de vizinhos de alta confiança é preferível ao invés de muitos vizinhos a qual a confiabilidade nos pesos atribuídos das avaliações não são confiáveis (DESROSIERS; KARYPIS, 2015). No caso de sistemas onde o número de usuários é muito maior que o número de itens, por exemplo Netflix (CLARK, 2020, sobre o número de itens no catálogo da Netflix) (CURRY, 2022, sobre o número de usuários em sites de *streaming*, incluindo Netflix), abordagens baseadas em Item – Item produzem uma recomendação mais precisa. Da mesma forma, sistemas com menos usuários que itens se beneficiará de uma abordagem de vizinho-mais-próximo baseada em Usuário – Usuário. Um bom exemplo da abordagem Usuário – Usuário é o de Herlocker et al. (2003) em seu *paper* que apresenta milhares de usuários, mas centenas de milhares de artigos como itens de recomendação.

(b) **Eficiência:** Como demonstra o Quadro 4 – continuando a dissertação de Desrosiers e Karypis (2015) – a eficiência computacional e de memória de um sistema de recomendação depende muito da relação entre o número de usuários e itens. Portanto, quando o número de usuários exceder o número de itens – o que é muito comum em sistemas de recomendação de grande escala – a abordagem de recomendação Item – Item é mais eficiente, pelo fato de que exige muito menos memória e tempo de computação os pesos similares entre itens (fase de treinamento) comparado a abordagem Usuário – Usuário, fazendo

da abordagem Item – Item mais escalável. Entretanto, a complexidade de tempo em relação a fase de recomendação online – a qual depende somente do número de itens disponíveis e o máximo de vizinhos – é o mesmo para ambas abordagens Usuário – Usuário e Item – Item.

Na prática a computação de pesos similares é muito menos custosa no pior caso possível de complexidade, como demonstra no Quadro 4; isso se dá ao fato de que usuários dão avaliações a somente alguns itens (DESROSIERS e KARYPIS, 2015). Assim, somente os pesos de similaridade diferentes de zero precisam ser armazenados, o que geralmente é muito menor que o número de pares de usuário. Esse número pode ser reduzido ainda mais: armazenando para cada usuário apenas os pesos superiores N, onde N é um parâmetro (RESNICK et al., 1994) que é suficiente para uma cobertura satisfatória dos pares de Usuário – Item. Da mesma forma, pesos diferentes de zero podem ser computados eficientemente sem ter que testar cada par de usuários ou itens – tornando os métodos de vizinho-mais-próximo escaláveis para sistemas de recomendação muito grandes.

(c) **Estabilidade:** Desrosiers e Karypis (2015) sobre a estabilidade entre filtragem baseada em Usuário – Usuário e Item – Item:

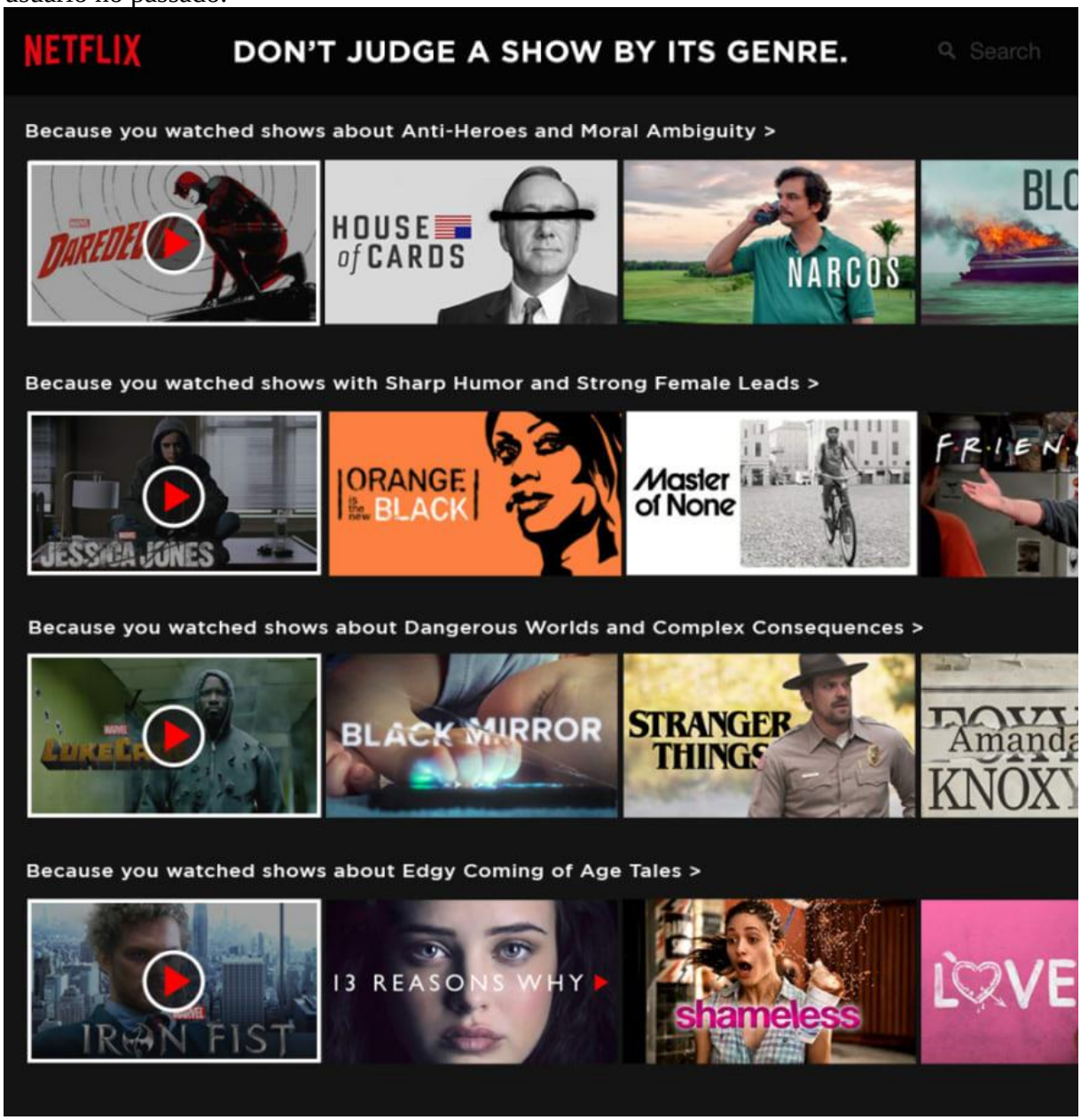
> A escolha entre Usuário – Usuário e Item – Item também depende da frequência e quantidade de mudança nos itens e usuários do sistema. Se a lista de itens disponíveis para o usuário for suficientemente estática em comparação aos usuários do sistema uma abordagem Item – Item será preferível, desde que a similaridade dos pesos pode então ser computada de com um intervalo mais infrequente de tempo e ser capaz de recomendar itens aos novos usuários.

E o contrário também é válido, nas aplicações onde a lista de itens está constantemente mudando, como o exemplo de Herlocker et al. (2003) em seu *paper*, onde haverá constante entrada de artigos, uma abordagem Usuário – Usuário é preferível.

(d) **Capacidade de justificação:** Uma vantagem de utilizar uma abordagem Item – Item é que esta pode ser fácil de justificar. Isto é, a lista de itens vizinhos usada para a recomendação, assim como a similaridade em seus pesos, pode ser apresentada ao usuário como uma explicação de uma recomendação. Sistemas baseados em Usuário – Usuário são menos receptíveis ao processo de justificação da recomendação, pois o usuário não sabe que outros usuários estão

servindo de vizinhos para dada recomendação (DESROSIERS; KARYPIS, 2015). A Figura 2, por exemplo, demonstra as recomendações feitas pela Netflix utilizando o que parece ser a abordagem Item – Item. É possível deduzir isso pela justificativa no topo de cada lista de filmes: acima da primeira lista de filmes, por exemplo, há um texto que pode ser traduzido para "[você está sendo recomendado isso] porque você assistiu séries sobre anti-heróis e ambiguidade moral".

Figura 2 – Netflix recomendando filmes e séries utilizando como base itens assistidos pelo usuário no passado.



Fonte: artigo do site dev.to: "*How did Netflix use ML to become the world's streaming leader?*", publicado por "Mage" em 08 de fev. de 2022.

(e) **Acaso:** em uma abordagem Item – Item, a avaliação prevista para um item é baseada na avaliação do usuário a itens similares. Consequentemente, sistemas de recomendação que utilizam esse método tendem a recomendar ao usuário itens que estão relacionados com aqueles que costumam ser apreciados pelo usuário em questão. Por exemplo, em uma aplicação que recomendará filmes, filmes do mesmo gênero, atores ou diretor serão recomendações prováveis. Por mais que esta abordagem seja segura, pode ser mais difícil para o usuário

descobrir tipos de itens diferentes que talvez ele irá gostar tanto quanto (DESROSIERS; KARYPIS, 2015).

Sistemas de recomendação utilizando uma abordagem Usuário – Usuário, por outro lado, naturalmente terão recomendações mais “ao acaso”. Isso é particularmente verdade para recomendações feitas a um pequeno número de vizinhos-mais-próximos. Por exemplo: um usuário A que só assiste comédia pode ser muito similar a um usuário B que também assiste comédias. Entretanto, se B se gostar de outro gênero de filme, esses filmes de gênero diferente podem ser recomendados ao usuário A através da similaridade que ele compartilha com B (DESROSIERS; KARYPIS, 2015).

2.2.2.2 Filtragem baseada em modelo

Nas palavras de Ricci, Rokach e Shapira (2015, p. 39, tradução própria):

> Ao invés de usar as notas fornecidas pelos usuários diretamente no momento de prever quais itens recomendar ao usuário como a filtragem colaborativa baseada em “vizinho mais próximo”, abordagens que utilizam a filtragem baseada em modelo usam essas notas fornecidas pelo usuário para o aprendizado de um modelo preditivo. Características de usuários e itens que se sobressaírem são capturadas por um conjunto de parâmetros de um modelo.

Para a tarefa de recomendação de itens o método de filtragem baseada em modelo existe várias abordagens como clusterização bayesiana (BREESE; HECKERMAN; KADIE, 1998), análise semântica latente (HOFMANN, 2003), alocação latente de Dirichlet (BLEI; NG; JORDAN, 2003), máxima entropia (ZITNICK; KANADE, 2004), máquinas de Boltzmann (SALAKHUTDINOV; MNIH; HINTON, 2007), máquinas de vetores de suporte (GRČAR et al., 2006), e decomposição em valores singulares (PATEREK, 2007).

2.2.3 Filtragem híbrida

Como o nome sugere, filtragem híbrida é a combinação de duas ou mais técnicas de filtragem (THORAT; GOUDAR; BARVE, 2015) (SÁNCHEZ, 2013). A abordagem híbrida foi introduzida para superar alguns problemas que são comummente associados com as técnicas de filtragem separadamente, por exemplo: filtragem colaborativa costuma ter um problema de "*cold start*" e filtragem baseada em conteúdo tem o problema de superespecialização. Outro motivo para utilizar a filtragem híbrida é a melhora da precisão e eficiência do processo de recomendação (THORAT; GOUDAR; BARVE, 2015); a melhora na precisão e eficiência se dá pelo fato de que com a filtragem híbrida o sistema é capaz de explorar os pontos positivos de cada abordagem (SÁNCHEZ, 2013).

Filtragem híbrida costuma ser utilizada sobre dados biológicos ou probabilísticos. Tais métodos incluem, mas não se limitam a: algoritmos genéticos (SALEHI; POURZAFERANI; RAZAVI, 2013), genética difusa (em inglês, *fuzzy genetic*) (AI-SHAMRI; BHARADWAJ, 2007), redes neurais (REN et al., 2008), redes bayesianas (em inglês, *Bayesian Networks*) (CAMPOS et al., 2010), clusterização (SHINDE; KURKARNI, 2012) e características latestes (em inglês, *43atente features*) (MANEEROJ; TAKASU, 2009), entre outros.

2.2.4 Problemas comuns em sistemas de recomendação

"A pesquisa no campo de sistemas de recomendação deixou de abordar o problema de calcular as classificações previstas de itens para abordar questões que surgiram agora que sistemas de recomendação se tornaram mais difundidos" (BURKE, 2001) (GOOD et al., 1999). Kunaver e Požrl (2017) dissertam bem sobre os problemas relacionados à sistemas de recomendação modernos, trazendo a seguinte lista:

2.2.4.1 Escassez de dados

Independentemente do tipo de abordagem que um sistema de recomendação adotará para fazer as recomendações, todos os sistemas de recomendação sofrem do mesmo problema: escassez de dados (HUANG; CHEN; ZENG, 2004): trabalhando com conjuntos de dados de Usuário – Item praticamente vazios. O problema surge do simples fato de é impossível que cada usuário forneça *feedback* para cada item existente no banco de dados. Ter todas as classificações possíveis também anularia o propósito dos sistemas de recomendação, pois significa que não há mais nada a ser recomendado. E também os números tanto de itens e usuários crescem constantemente à medida que novos itens são adicionados e novos usuários se cadastram no sistema. Sistemas de recomendação precisam ser capazes de trabalhar com as tabelas de dados praticamente vazias (KUNAVER; POŽRL, 2017).

2.2.4.2 Arranque a frio (ou em inglês, *cold start*)

Cada novo item ou usuário apresenta um problema, pois o sistema não pode criar imediatamente o modelo de recomendação necessário. No caso de um novo item, o sistema não consegue decidir se o item é relevante para algum usuário até que o item seja descrito pelos metadados apropriados e/ou for avaliado por pelo menos alguns usuários. No caso de um novo usuário o sistema pode – na maioria dos casos – apresentar apenas recomendações genéricas como itens com as melhores classificações, por exemplo, até que o usuário forneça informações suficientes sobre si mesmo na forma de dados demográficos ou na forma de *feedbacks* sobre alguns dos itens da base de dados. Esse problema é conhecido por “arranque a frio”, ou *cold start* (SCHEIN et al., 2002).

2.2.4.3 Big data

Em poucas palavras, Big Data é um termo que costuma ser utilizado para se referir a um grande conjunto de dados armazenado, esse aglomerado de dados precisa ser armazenado e analisado (MIRANDA, 2017). O problema está no fato de que para uma melhor previsão do que o usuário deseja assistir no momento da recomendação, a coleta de uma grande quantidade de dados deve ser feita e o processamento desses dados complexos e muitas vezes não totalmente relacionados também se faz necessário, ou seja, é preciso um *design* complexo para a coleta dos dados do usuário e a análise eficaz desses dados, tornando Big Data um possível problema, e certamente um desafio em sistemas de recomendação (AMER; ABDALLA; NGUYEN, 2021).

Kunaver e Požrl (2017) fornecem uma visão similar a Amer, Abdalla e Nguyen (2021) em relação ao problema de Big Data no contexto de sistemas de recomendação:

> Embora os sistemas de recomendação enfrentam o problema de escassez de dados, eles também encontram o problema do lado oposto deste espectro – o problema de Big Data. Como dito anteriormente [sobre a escassez de dados], a maioria dos valores está faltando as tabelas de itens do usuário, já que a maioria dos usuários fornece *feedback* sobre apenas alguns itens do conteúdo consumido, mas com o número de usuários e itens na maioria dos sistemas, como o YouTube por exemplo, com a soma desses números na faixa dos milhões, senão bilhões, isso ainda significa que há uma imensa estrutura de dados que normalmente não pode ser processada sem algoritmos especializados.

2.2.4.4 Superespecialização (ou *over-fitting*)

Uma vez que o sistema é capaz de gerar recomendações de forma consistente para cada usuário, surge um novo problema, pois o sistema pode começar a recomendar itens de um espectro muito “estreito” do interesse dos usuários, por exemplo, um serviço de venda de doces online só recomenda rosquinhas de sabores diferentes a um grupo de usuários, mesmo tendo dúzias de variedades que este usuário poderia gostar. Esse problema ocorre quando – ironicamente – um usuário está tentando ser prestativo e fornece *feedback* explícito sobre os itens que ele realmente gosta muito. Isso leva a criação de um modelo de recomendação muito

específico que sabe exatamente as preferências de tal usuário, portanto, não consegue detectar mais nenhum tipo de item que seja do interesse do usuário, pois o usuário não mostrou nenhum interesse nestes itens, por exemplo, o usuário da loja de doces online nunca demonstrou interesse em sonhos, trufas ou panquecas – só rosquinhas. Isso é conhecido como superespecialização ou *over-fitting* (KUNAVER; POŽRL, 2017).

2.2.4.5 Ovelha cinza

Ovelhas cinzas são usuários que possuem um gosto especial e não costumam concordar com a maioria dos usuários. A identificação desses usuários (ovelha cinzas) se torna então um desafio para sistemas de recomendação, pois o sistema poderá ter dificuldades para recomendar itens para este usuário e estas recomendações podem acabar não sendo tão precisas (ZHENG; AGNANI; SINGH, 2017). Fazziki et al. (2019) também define o tema ovelha cinza de forma parecida: "usuários com preferências únicas podem tornar difícil o processo de recomendação e a criação de perfis precisos. Ou seja, a busca por similaridade no momento da recomendação falha em encontrar bons resultados."

2.2.5 Métodos para mineração de dados em sistemas de recomendação

Ricci, Rockach e Shapira (2015) contemplam que tipicamente há 3 passos no processamento de dados, executados nessa sequência: (*a*) pré-processamento de dados, (*b*) modelo de aprendizagem e (*c*) interpretação dos resultados (ou testagem e validação). Aqui é importante uma revisão introdutória sobre as principais etapas de pré-processamento de dados e métodos de aprendizado de máquina mais comummente vistos em implementações de sistemas de recomendação, estes métodos de aprendizado de máquina sendo divididos em (1) aprendizado supervisionado e (2) aprendizado não supervisionado (RICCI; ROKACH; SHAPIRA, 2015) (BISHOP, 2006).

Esse capítulo tem como intensão esclarecer o impacto que a mineração de dados tem no campo dos sistemas de recomendação e revisar métodos e técnicas de mineração de dados que foram utilizadas com sucesso. Dito isso, a Figura 3 a seguir ilustra com clareza as fases do processo de mineração de dados:

Figura 3 – principais passos no processo de mineração de dados

Fonte: autoria própria (2022), imagem baseada na figura 7.1 do livro *Recommender Systems Handbook* (RICCI; ROKACH; SHAPIRA, 2015, p. 228, tradução própria).

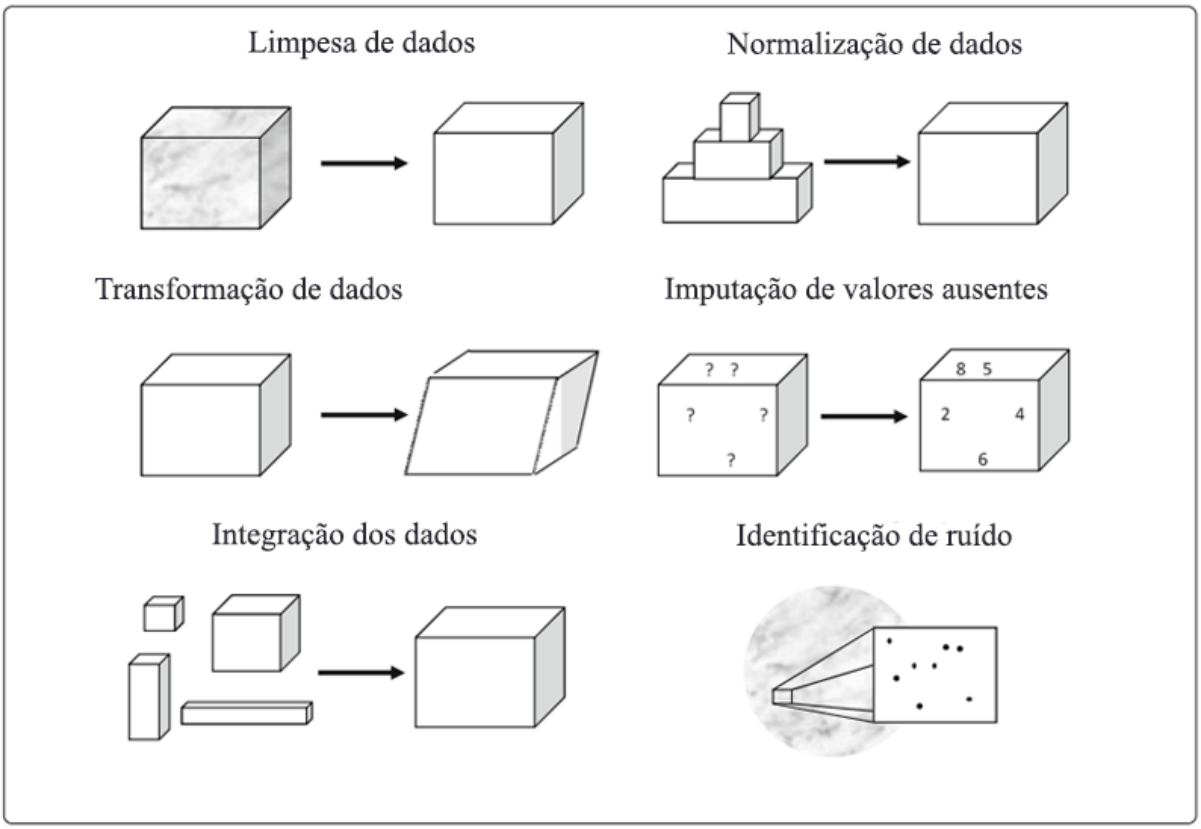
2.2.5.1 Pré-processamento de dados

O conjunto de técnicas utilizadas anteriores a aplicação de métodos de mineração de dados é nomeado de pré-processamento para mineração de dados e é conhecido por ser um dos mais significativos problemas dentro do processo de descobrimento de conhecimento a partir de dados (GARCÍA et al., 2016).

> [...] Visto que os dados provavelmente serão imperfeitos, contendo inconsistências e redundâncias, [o conjunto de dados] não é diretamente aplicável para começar o processo de mineração de dados. É importante também mencionar o rápido crescimento das taxas de geração de dados e seu tamanho em aplicações negociais, industriais, acadêmicas e aplicações científicas. Quanto maior a quantidade de dados coletados, mais sofisticado precisará ser o mecanismo para analisá-los. O pré-processamento de dados é capaz de adaptar os dados aos requisitos impostos por cada algoritmo de mineração de dados, permitindo processar dados que seriam inviáveis [para análise]. (GARCÍA et al., 2016, p. 3, tradução própria)

A Figura 4 que García et al. (2016) nos disponibiliza demonstra com clareza os passos no processo de pré-processamento de dados:

Figura 4 – etapas no pré-processamento de dados



Fonte: *Big data preprocessing: methods and prospects* (GARCÍA et al., 2016, p. 3, o nome das etapas fora traduzido para melhor compreensão)

García et al. (2016) afirma que após a aplicação bem sucedida do estágio de pré-processamento dos dados o conjunto de dados final obtido pode ser considerado confiável e uma fonte de dados adequado para o algoritmo de mineração de dados aplicado posteriormente.

Ricci, Rokach e Shapira (2015) propõem os 4 seguintes passos para o pré-processamento dos dados num sistema de recomendação: metrificações de distância (ou *distance metrics*, em inglês), os autores também se referem a essas metrificações como "medidas de semelhança", amostragem (ou *sampling*, em inglês), redução de dimensionalidade (ou *dimensionality reduction*, em inglês) e eliminação de ruído (ou *denoising*, em inglês). É interessante observar que enquanto García et al. (2016) observa o pré-processamento dos dados para *big data*; Ricci, Rokach e Shapira (2015) em *Recommender Systems Handbook* voltam o olhar do pré-processamento dos dados diretamente para sistemas de recomendação e suas peculiaridades.

2.2.5.1.1 Medidas de semelhança

Uma das abordagens preferíveis para sistemas de recomendação que utilizam filtragem colaborativa é o classificador kNN (*k-nearest neighbor*). Esse método de classificação – assim como a maior parte de classificadores e técnicas de clusterização – é altamente dependente da definição apropriada da similaridade ou a medida de distância entre itens de classificação. Apesar do fato de que as medidas de semelhança serem um processo necessário na maioria dos processos de mineração de dados, este não é um requisito para todos estes processos (RICCI; ROKACH; SHAPIRA, 2015).

O exemplo mais simples e mais comum de medir a distância entre estes itens é, segundo Ricci, Rokach e Shapira (2015), medir a distância Euclidiana ou a "*norma L2*" (Equação 12):

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \qquad (12)$$

Onde n é o número de dimensões (atributos [dos itens disponíveis para recomendação]) e x_k e y_k são os k^o atributos (componentes) dos objetos de dados x e y, respectivamente.

Além de medidas de distância entre itens, podemos também utilizar cálculos de correlação como correlação de *Pearson* (como exemplificado no capítulo 2.2.2.1.1) para determinar métricas de semelhança entre os itens.

2.2.5.1.2 Amostragem

Amostragem é a técnica principal usada em mineração de dados para selecionar um subconjunto de dados relevantes de um grande conjunto de dados. Utilizado tanto no pré-processamento quanto na parte final do processo, onde há a interpretação dos dados. Amostragem é utilizado também porque há um custo computacional muito grande para

processar um conjunto de dados inteiro. Amostragem também pode ser utilizado para criar conjuntos de dados de treinamento e testagem. Nesse caso, o conjunto de treinamento é utilizado para aprender os parâmetros ou configurar os algoritmos utilizados no momento da análise, enquanto um conjunto de dados para testagem é utilizado para avaliar o modelo ou configuração obtido na fase de treinamento, garantindo que a análise performará bem com dados não vistos previamente. Na verdade, na maioria dos casos não só é preciso criar conjuntos para treinamento e testagem, mas um terceiro também para validação. O conjunto de treinamento serve para "modelar o modelo" – por assim dizer – o conjunto de validação serve para aprender hiper parâmetros[1] e por fim, o conjunto de testagem serve para ver como o modelo está generalizando (RICCI; ROKACH; SHAPIRA, 2015). É na amostragem que pode ocorrer a superespecialização, discutida na sessão 2.2.4.4, Ricci, Rokach e Shapira (2015) propõem uma solução para tal problema: dito que os conjuntos de dados de treinamento e testagem são criados a partir do conjunto de dados original, o modelo é treinado utilizando os dados de treinamento e testados com os exemplos nas amostras de testagem. Depois disto, diferentes conjuntos de dados de treinamento e testes são selecionados para o processo de treinar/testar, e esse processo se repete *K* vezes. E por fim, a performance média de *K* aprendida pelos modelos é reportada. Este processo é conhecido como, em tradução literal, "validação cruzada" (*cross-validation* em inglês).

2.2.5.1.3 Redução de dimensionalidade

"Redução de dimensionalidade é o passo no pré-processamento que remove atributos redundantes, ruidosos e dados irrelevantes, esse processo é realizado para melhorar a precisão do aprendizado e reduzir o tempo de treinamento". (VELLIANGIRI; ALAGUMUTHUKRISHNAN; JOSEPH, 2019, p. 1, tradução própria)

Sarwar et al. (2000) em "*Application of Dimensionality Reduction in Recommender System – A Case Study*" apresentam uma análise da redução de dimensionalidade em sistemas de recomendação, nesse estudo de caso os autores fazem dois experimentos onde exploram a

[1] hiper parâmetros, ou *hyperparameters* em inglês, são parâmetros cujos valores controlam o processo de aprendizado e determinam os valores dos parâmetros do modelo que um algoritmo de aprendizado acaba aprendendo. O prefixo "hiper" sugere esses parâmetros como de "nível superior", ou "*top-level*", que controlam o processo de aprendizagem e os parâmetros do modelo que dele resultam (NYUYTIYMBIY, 2020)

tecnologia *Singular Value Decomposition* (SVD) para reduzir a dimensionalidade das bases de dados do sistema de recomendação. O primeiro experimento compara a eficácia de dois sistemas de recomendação em prever as preferências do usuário baseado num banco de dados com *feedback* explícito dos usuários sobre os itens de recomendação. O segundo experimento compara a eficácia de dois sistemas de recomendação na produção de listas "*Top-N*" com base em uma base de dados de compras de clientes da vida real num site de *e-commerce*. Segundo Sarwar et al. (2000) os experimentos feitos com SVD possuem potencial de enfrentar muitos dos desafios no campo dos sistemas de recomendação, sob certas condições.

SVD também é conhecido como fatoração de matrizes (RICCI; ROKACH; SHAPIRA, 2015) (KOREN; BELL; VOLINSKY, 2009). O fato de que a técnica de SVD venceu o prêmio de um milhão de dólares ofertado pela *Netflix* é relembrado no trabalho de Ricci, Rokach e Shapira (2015) e devidamente aprofundado no trabalho de Koren, Bell e Volinsky (2009). Em ambas literaturas a técnica de SVD (ou fatoração de matrizes) é tida como poderosa e simples: "[...] a abordagem de fatoração de matrizes pode ser considerada mais que uma simples técnica de pré-processamento ou redução de dimensionalidade, visto que todo o problema de recomendação pode ser sumarizado como uma conclusão de matriz. [...]". (RICCI; ROKACH; SHAPIRA, 2015)

2.2.5.1.4 Eliminação de ruído (ou em inglês: denoising)

Dados coletados para fins de mineração de dados podem estar sujeitos a diferentes tipos de ruídos, como valores ausentes e pontos fora da curva. Eliminação de ruído é a etapa de pré-processamento que, como o próprio nome diz, remove qualquer efeito indesejado nos dados causado por esses ruídos, enquanto maximiza a informação desses dados. (RICCI; ROKACH; SHAPIRA, 2015)

Ricci, Rokach e Shapira (2015) continuam: definindo ruído como "todo e qualquer artefato indesejado que vieram na etapa de coleta de dados e que podem impactar os resultados

duma análise e interpretação dos dados". Num contexto de sistemas de recomendação, podemos distinguir entre ruídos naturais e maliciosos (O'MAHONY; HURLEY; SILVESTRE, 2006). Os autores definem ruídos naturais como um ruído involuntário, ou seja, introduzido involuntariamente pelo usuário quando forneceu *feedback* sobre suas preferências. Ruídos maliciosos são, segundo O'Mahony, Hurley e Silvestre (2006), "ruídos deliberadamente introduzidos em um sistema com o objetivo de enviesar os resultados.

Fica claro que ruídos maliciosos podem afetar os resultados de saída de um sistema de recomendação. Mas também, como demonstra os estudos feitos por Amatriain, Pujol e Oliver (2009), os efeitos sobre os resultados de saída causados por ruídos naturais não são negligenciáveis. Para corretamente endereçar os ruídos naturais, em 2009 os autores Amatriain, Pujol, Oliver e Tintarev produziram o artigo "*Rate it Again: Increasing Recommendation Accuracy by User re-Rating*", a ideia principal do método é melhorar a precisão das recomendações solicitando *feedback* para o usuário de itens que ele já deu nota.

As melhorias na precisão dos resultados da recomendação podem ser maiores na etapa de eliminação de ruído do que as melhorias nos resultados obtidos quando utilizado algoritmos complexos de otimização (RICCI; ROKACH; SHAPIRA, 2015).

2.2.5.2 Aprendizado supervisionado

Manning, Raghavan e Schütze (2009) explicam que aprendizado supervisionado tem esse nome porque um supervisor (humano que define as classes e rotula documentos de treinamento) serve como professor, direcionando o processo de aprendizado. O método de aprendizagem supervisionado é denotado por Γ e escrito $\Gamma(\mathbb{D}) = \gamma$. O método de aprendizado Γ recebe um conjunto de treinamento $\mathbb{D}$ como entrada e retorna uma função de classificação aprendida γ. Essa definição de aprendizado supervisionado é também fornecida por Bishop (2006) em seu trabalho dedicado em reconhecimento de padrão e aprendizado de máquina, no caso de Bishop, o autor do livro afirma que o objetivo da aprendizagem supervisionada é modelar uma distribuição condicional $p(t|x)$ [...].

2.2.5.2.1 Vizinhos mais próximos

O método de classificação de vizinhos mais próximos, abordado no contexto de filtragem colaborativa na sessão 2.2.2.1, é mais antigo que os próprios sistemas de recomendação como *A Tapeçaria* (1992), primeiro sistema de recomendação voltado a filtragem (colaborativa) de e-mails, o artigo *Nearest Neighbor Pattern Classification* de Cover e Hart e publicado em 1967 (p.1, tradução própria) descreve o algoritmo de vizinhos mais próximos como:

> A regra de decisão do vizinho mais próximo atribui a classificação do conjunto mais próximo de um conjunto de pontos previamente classificados a um ponto amostral não classificado. Esta regra é independente da distribuição conjunta subjacente nos pontos amostrais e suas classificações, e, portanto, a probabilidade de um erro R de tal regra deve ser pelo menos tão grande quanto a probabilidade de erro Bayers R*- a probabilidade mínima de erro sobre as regras de decisão considerando a probabilidade estrutural subjacente [...]

Um fato interessante a se citar é que uma das referências para o trabalho de Cover e Hart é o livro "*The Mathematical Foundations of LEARNING MACHINES*", e aborda o tema de descobrir relacionamentos escondidos entre os dados, o livro é de 1965 (NILSSON; SEJNOWSKI; WHITE).

O algoritmo de vizinhos mais próximos foi um dos primeiros algoritmos a resolver o "problema do caixeiro viajante" (KIZILATEŞ; NURIYEVA, 2013).

2.2.5.2.2 Arvores de decisão

Como Quinlan (1986) explica, árvores de decisão são classificadores em um atributo alvo (ou classe) com uma estrutura de árvore. As observações – ou itens – a serem classificados são compostos por seus atributos e valor alvo. Os nós dessa árvore podem ser: (a) nós de decisão: nesses nós um valor de atributo único é testado para determinar para determinar qual ramificação da subárvore se aplica; ou (b) nós de "folha", que indicam o valor do atributo alvo.

Quinlan (1986) propõe um bom exemplo para a visualização do conceito da árvore de decisão. Dado o conjunto de dados:

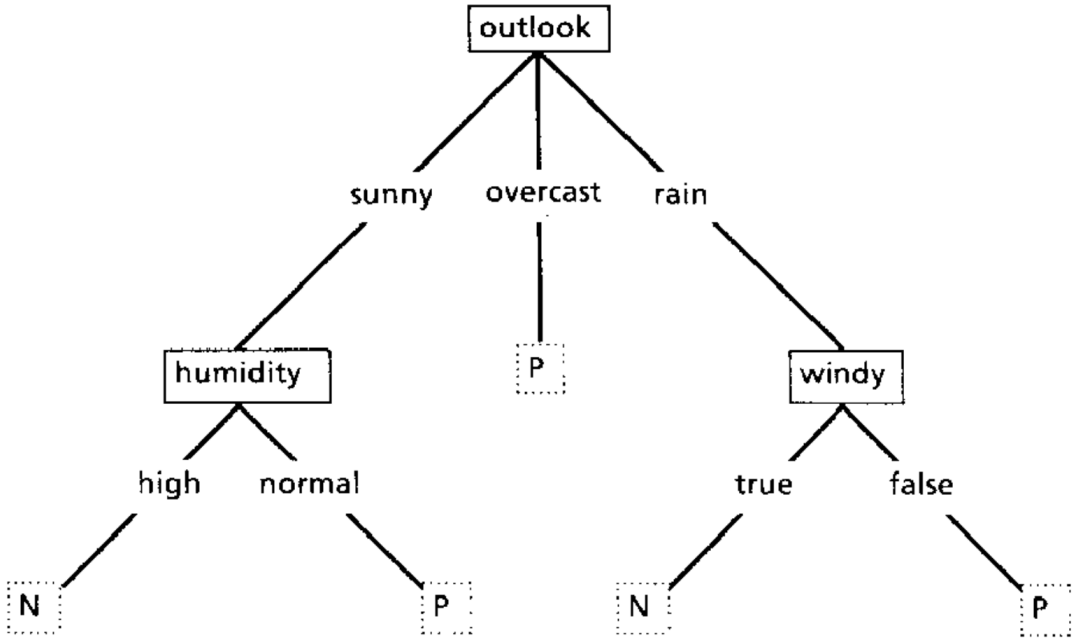
Quadro 5 – exemplo de conjunto de dados para árvore de decisão

Nº	Outlook	Temperature	Humidity	Windy	Class
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Fonte: Induction of Decision Trees (QUINLAN, 1986, p. 6)

Quinlan, utilizando os dados do Quadro 5, ilustra a árvore decisória da seguinte forma (Figura 5):

Figura 5 – Utilizando os dados do Quadro 5 de Quinlan, é ilustrado a lógica da árvore de decisão simples

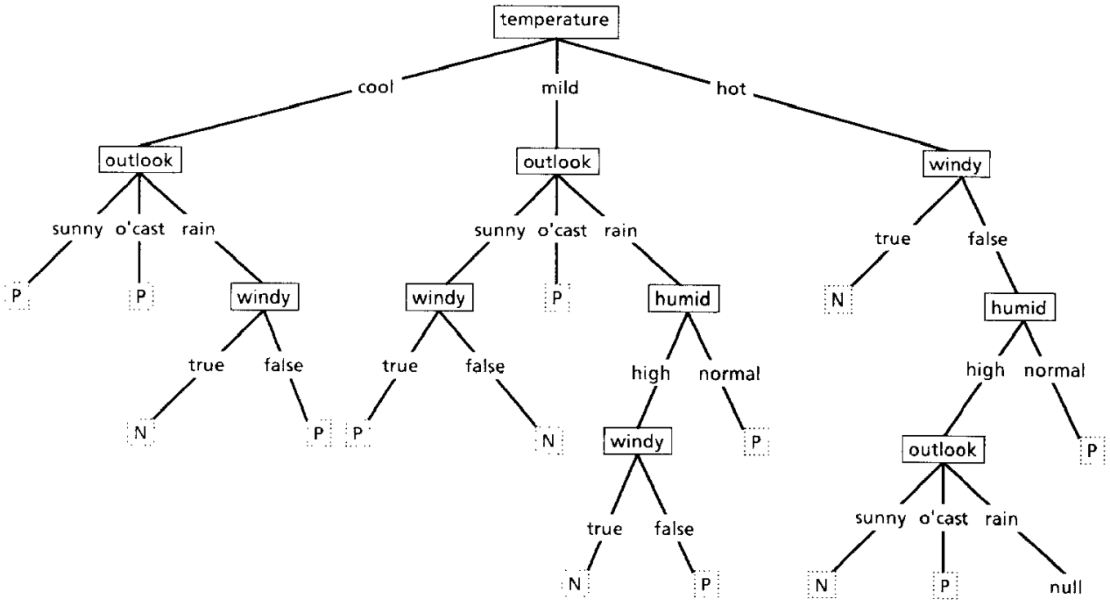


Fonte: Induction of Decision Trees (QUINLAN, 1986, p. 7)

Quinlan esclarece que nesse exemplo não há muitos atributos sendo considerados, portanto essa árvore é uma árvore de decisão simples [...].

Uma árvore de decisão complexa condensa todas as possibilidades em uma só árvore e, apesar de finito, o número dessas possibilidades pode ser bem grande – fazendo duma árvore complexa mais indicada para pequenas tarefas de indução (QUINLAN, 1986)

Figura 6 – Utilizando os dados da Tabela 5 de Quinlan, é ilustrado a lógica da árvore de decisão complexa



Fonte: Induction of Decision Trees (QUINLAN, 1986, p. 8)

Segundo Quinlan, a árvore de decisão complexa (Figura 6) foi criada para outro espectro: onde há muitos atributos e o conjunto de treinamento possui muitos objetos, onde é necessária uma árvore de decisão razoavelmente boa e sem a necessidade de muita computação.

2.2.5.2.3 Classificadores Bayesianos

Um classificador Bayesiano é uma estrutura (ou *framework*, como os autores o chamam em inglês) para resolver problemas de classificação baseado na definição de uma probabilidade condicional e no teorema de Bayers (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997).

> A escola Bayesiana de estatísticas utiliza probabilidade para representar incerteza sobre as relações aprendidas com os dados. Além disso, o conceito ‘anteriores’ é muito importante, pois representa as expectativas ou conhecimento prévio sobre o que o relacionamento verdadeiro pode ser. Em particular, a probabilidade de um modelo de dados (posterior) é proporcional ao produto da probabilidade vezes a probabilidade anterior. O componente de probabilidade inclui o efeito dos dados enquanto a ‘crença’ no modelo antes dos dados serem observados. (RICCI; ROKACH; SHAPIRA, 2015, p. 241, tradução própria).

Na explicação de Ricci, Rokach e Shapira (2015), Classificadores Bayesianos consideram cada atributo e etiqueta da classe (ou como os autores chamaram em inglês: *label*) como variáveis aleatórias (sendo estas discretas ou contínuas). Dando um registro de *N* atributos $(A_1, A_2, \dots, A_N)$, o objetivo é prever a classe C_k encontrando o valor de C_k que maximize a probabilidade posterior da classe: dado os dados $P(C_k | A_1, A_2, \dots, A_N) \propto P(A_1, A_2, \dots, A_N \mid C_k)\, P(C_k)$.

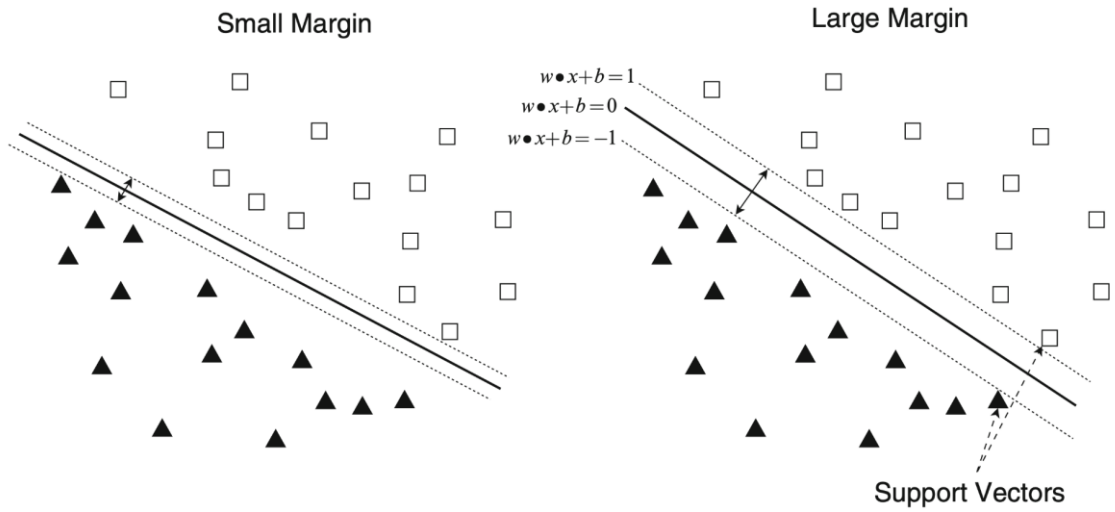
Um classificador utilizado comumente é o *Naive Bayes Classifier*. Para estimar a probabilidade condicional, $(A_1, A_2, \dots, A_N \mid C_n)$, um *Naive Bayes Classifier* assume a independência probabilística dos tributos – por exemplo: a presença ou ausência de um atributo em particular não é relacionado com a presença ou ausência de outro atributo. Essa suposição formalmente é: $P(C_k | A_1, A_2, \dots, A_N) = P(A_1 | C_k) P(A_2 | C_k) \dots P(A_N | C_k)$. Ricci, Rokach e Shapira (2015) citam como benefícios no *Naive Bayes Classifier* o fato de ser robusto em isolar pontos ruidosos nos dados e atributos relevantes, e também lidam com dados faltantes, ignorando-os durante os cálculos de estimativa de probabilidade.

2.2.5.2.4 Support Vector Machines (SVM)

O objetivo de um classificador de SVM é encontrar um hiperplano linear (limite de decisão) que separa os dados de tal forma que a margem é maximizada (CRISTIANINI; SHAWE-TAYLOR, 2000).

Ricci, Rokach e Shapira (2015) fornecem um bom exemplo sobre o tópico de Support Vector Machines:

Figura 7 – Demonstração do exemplo dos autores sobre SVM, demonstrando as fronteiras de um SVM.



Fonte: Recommender Systems Handbook (RICCI, ROKACH e SHAPIRA, 2015, p. 244)

No exemplo em questão os autores utilizam como exemplo o problema de separação de duas classes em duas dimensões – vistas na Figura 7 – evidenciando que existem muitas linhas de fronteira possíveis para separar as duas classes. Cada fronteira tem uma fronteira associada. A lógica por trás dos *SVMs*, segundo Ricci, Rokach e Shapira (2015), é que se escolhermos aquele que maximiza a margem é menos provável que classifiquemos incorretamente itens desconhecidos no futuro. A separação linear entre duas classes é alcançada através da função w ● x + b = 0. A função que pode classificar itens como sendo de classe é definida como +1 ou -1, enquanto esses itens são separados por alguma distância mínima da função de separação de classe.

2.2.5.3 Aprendizado não supervisionado

"Aprendizado não supervisionado não é comumente utilizado na área de sistemas de recomendação", a afirmação pode ser confirmada pelo trabalho de Ricci, Rokach e Sapira (2015), e os autores de *"MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS*" (KOREN; BELL; VOLINSKY, 2009) – observadores do prêmio da *Netflix* como já citado algumas vezes nessa monografia – comentam sobre algoritmos como "Vizinhos mais próximos" e, como o nome do artigo indica: "fatoração de matrizes" (tradução literal de *Matrix*

Factorization), indicando uma inclinação natural dos sistemas de recomendação a utilizarem aprendizado supervisionado para seus sistemas.

Mesmo assim ainda há autores que abordam aprendizado não supervisionado em sistemas de recomendação: "*Conversational Recommendation System with Unsupervised Learning*" (SUN et al., 2016); "*Unsupervised Learning of Paragraph Embeddings for Context-Aware Recommendation*" (XIE et al., 2019). Ainda é possível uma terceira abordagem: utilizar uma combinação de aprendizado supervisionado e não supervisionado: "*Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization*" (BAKSHI et al., 2014).

Ghahramani (2003) descreve aprendizado não supervisionado da seguinte forma: considere uma máquina que recebe uma sequência x_1, x_2, x_3... de *inputs*, esses *inputs* x_t são sensíveis ao tempo *t*. esse *input*, estes que serão chamados de 'dados', podem corresponder a imagem de uma retina, o som de uma voz cantando, ou palavras que correspondem a uma história. Na explicação de Ghahramani o afirma que pode parecer misterioso imaginar o que a máquina poderia aprender dado que não terá *feedback* do seu ambiente. No entanto, é possível desenvolver um *framework* (ou em tradução literal do inglês: "estrutura") formal para o aprendizado não supervisionado com base na noção de que o "objetivo da máquina é construir representações dos *inputs* utilizados na tomada de decisão, prever *inputs* futuros, comunicação eficiente das entradas para outra máquina, etc". Aprendizado não supervisionado pode ser considerado como "encontrar padrões nos dados anteriores e além, do que, antes disso, seria considerado ruído puro não estruturado".

2.2.5.3.1 K-Means

K-means é considerado um dos algoritmos mais populares e poderosos para mineração de dados na comunidade de pesquisa (SERAJ; ISLAM, 2020). Segundo Sinaga e Yang (2020) apesar do k-means ser um algoritmo de aprendizado não supervisionado para clusterização em reconhecimento de padrões e aprendizado de máquina, k-means sempre é influenciado por inicializações com um número necessários de clusters anteriores. Seraj e Islam (2020) em seu artigo sobre a performance do algoritmo k-means cita esse problema de inicialização e propõe alternativas para contorná-lo eficientemente. O artigo de Seraj e Islam:

"*The k-means Algorithm: A Comprehensive Survey and Performance Evaluation*" também realiza *benchmarks* nesses diferentes métodos e brevemente os introduz ao leitor.

Bishop (2006) define k-means utilizando o seguinte exemplo: o primeiro passo é identificar grupos, ou clusters de pontos de dados em um espaço multidimensional. Supondo um conjunto de dados $\{x_1, \ldots, x_N\}$ consistindo de observações *N* de uma variável aleatória D-dimensional euclidiana *x*. Intuitivamente, um cluster pode ser pensado como compreendendo um grupo de pontos de dados cujas distâncias entre pontos são pequenas comparadas com as distâncias para pontos fora do cluster – O objetivo do algoritmo k-means é particionar o conjunto de dados em um número *K* de clusters – Bishop (2006) segue, formalizando sua explicação: um grupo de vetores D-dimensional μ_k, onde k = *1, ..., K*, onde μ_k é um protótipo associado com o k^{th} cluster. O μ_k pode ser considerado como a representação dos centros dos clusters. O objetivo é, então, encontrar uma atribuição de pontos de dados para clusters, bem como um conjunto de vetores $\{\mu_k\}$, tal que a soma dos quadrados das distâncias de cada ponto de dados ao seu vetor mais próximo μ_k, é o mínimo possível.

Há algumas alternativas que não seja o k-means para um sistema de recomendação, algumas dessas são, por exemplo: "*Locality-sensitive hashing (LSH)*", técnica para solucionar o problema da busca do vizinho mais próximo em espaços de alta dimensionalidade. O algoritmo se baseia na utilização de funções de *hash* que preservam a "localidade", ou em outras palavras, agrupa itens que são similares (ANDONI; INDYK, 2008).

Outra alternativa ao k-means seria o "*Latent Dirichlet Allocation (LDA)*", o LDA é modelo de associação mista, em que se considera que cada ponto de dado pode pertencer a mais de um único cluster (BLEI; NG; JORDAN, 2003). Uma aplicação típica de LDA, segundo Ricci, Rokach e Shapira (2015) é identificar tópicos em coleções de documentos. Nesse caso, o LDA também está muito relacionado a análise semântica latente e, portanto, técnicas como *Singular Value Decomposition* (SVD).

3 METODOLOGIA

Para Gil (1999), os métodos definem os procedimentos lógicos que deverão ser realizados no processo de investigação científica dos fatos da natureza e da sociedade. Gil também afirma em seu trabalho "*Como Elaborar Projetos de Pesquisa*" (1996) que uma pesquisa bibliográfica é desenvolvida através de materiais já elaborados, podendo ser livros e artigos científicos.

O presente capítulo tem como objetivo explicitar o tipo de pesquisa, as etapas metodológicas e as devidas delimitações da presente monografia.

3.1 DEFINIÇÃO DO TIPO DE PESQUISA

A pesquisa pode ser dividida, de seu ponto de vista da natureza, como básica ou aplicada (SILVA; MENEZES, 2005). Esta monografia é caracterizada neste sentido como uma pesquisa aplicada. A justificativa para a afirmação é de que o objetivo do presente trabalho científico tem como objetivo gerar conhecimentos para a aplicação prática dirigida a solução de um problema (SILVA; MENEZES, 2005) (PRODANOV; FREITAS, 2013).

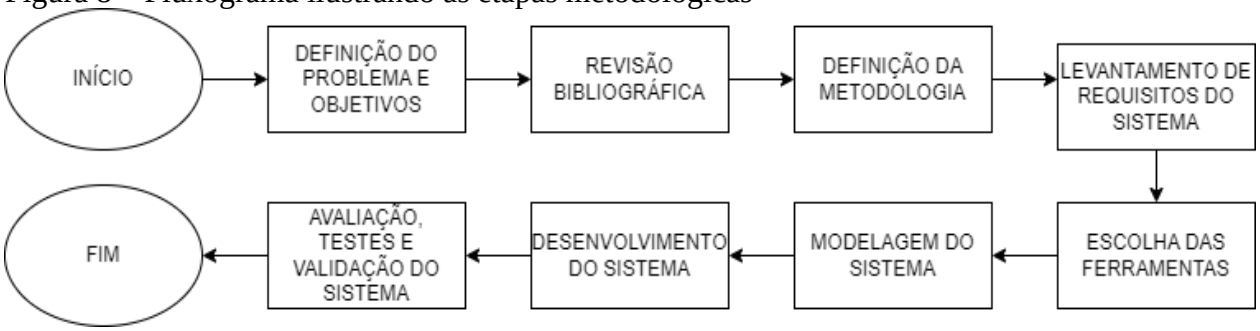
Em relação a procedimentos técnicos, a presente pesquisa se caracteriza como uma pesquisa bibliográfica. Gil afirma em seu trabalho "*Como Elaborar Projetos de Pesquisa*" (1996) que uma pesquisa bibliográfica é desenvolvida através de materiais já elaborados, podendo ser livros e artigos científicos. Numa pesquisa bibliográfica é importante que o pesquisador verifique a veracidade dos dados obtidos, observando as possíveis incoerências ou contradições que as obras possam apresentar. (PRODANOV; FREITAS, 2013, p. 54)

Em relação a abordagem, esta pode ser definida como uma abordagem qualitativa, isto é: "[...] a interpretação dos fenômenos e a atribuição de significados são básicas no processo de pesquisa qualitativa. Não requer o uso de métodos e técnicas estatísticas. O ambiente natural é a fonte direta para coleta de dados e o pesquisador é o instrumento-chave." (SILVA; MENEZES, 2005, p. 20).

3.2 ETAPAS METODOLÓGICAS

Para a visualização das etapas metodológicas, estas podem ser representadas a partir do fluxograma representado pela Figura 8 a seguir:

Figura 8 – Fluxograma ilustrando as etapas metodológicas



Fonte: autoria própria (2022)

3.2.1 Definição do problema e objetivos

A definição do problema e dos objetivos da presente monografia fora definido no capítulo 1: introdução, os objetivos na sessão 1.2 e a definição do problema na sessão 1.1.

3.2.2 Revisão bibliográfica

A revisão bibliográfica está disponível ao decorrer de todo capítulo 2, a revisão inicia com definições psicológicas sobre a habilidade de definir preferências, aborda em maiores detalhes o tópico do presente trabalho: sistemas de recomendação, e também introduz o assunto de mineração de dados, considerando os principais aspectos dessa mineração no âmbito de sistemas de recomendação.

3.2.3 Definição da metodologia

No presente capítulo (capítulo 3) é definida a metodologia, dividida em suas devidas sessões. E na atual sessão, sessão 3.2, é descrito em as etapas metodológicas desta monografia.

3.2.4 Levantamento de requisitos do sistema

Para desenvolver um sistema que atenda às necessidades do usuário é necessário que se defina os requisitos de tal sistema, com o levantamento de requisitos é possível prever as ações possíveis e também impossíveis dentro do nosso sistema e fornecer ao usuário implicitamente uma expectativa do que este usuário poderá performar dentro do sistema desenvolvido.

3.2.5 Escolha de ferramentas

Com os requisitos do sistema estabelecidos, se faz necessário ferramentas para atingir tais requisitos definidos. Essas ferramentas são, por exemplo, a linguagem de programação escolhida e ferramenta de versionamento de código fonte. Na etapa de escolha de ferramentas é descrita a função de tal ferramenta, qual papel dela na solução proposta, e, se se fizer necessário, informações adicionais sobre determinada ferramenta.

3.2.6 Modelagem do sistema

Na etapa de modelagem do sistema é realizado um planejamento de como as diferentes necessidades do sistema – visualizadas no levantamento de requisitos – serão implementadas a nível de software, por exemplo: um sistema de recomendação precisará de um modelo de aprendizagem, este modelo será retroalimentado? Como será feito o retreinamento? Existirá uma API exclusiva para o modelo de aprendizado de máquina ou será tudo disponibilizado em um grande serviço? Essas perguntas, assim como outras deverão ser respondidas nessa etapa para o sucesso do presente trabalho.

3.2.7 Desenvolvimento do sistema

Durante o desenvolvimento do sistema todas as etapas anteriores, e também etapas seguintes, influenciaram a forma que tomará o desenvolvimento do sistema. O desenvolvimento do sistema envolve a prototipação de interfaces gráficas, a implementação e revisão da modelagem feita para o sistema, que pode sofrer alterações ao longo de desenvolvimento se exibir possíveis melhorias a serem feitas. Isto significa dizer que as perguntas feitas e respondidas na modelagem do sistema deverão ser testadas e comprovadas – ou não – no desenvolvimento do sistema.

3.2.8 Avaliação, testes e avaliação dos sistemas

Com o sistema desenvolvido, é necessário expô-lo e ter uma avaliação de um usuário real. Dito que a monografia se propõe a realizar uma análise de forma qualitativa, e como descrito nos objetivos específicos os testes serão realizados através de um questionário aplicado a alguns usuários, o objetivo dos testes e avaliações é medir de forma qualitativa a qualidade do sistema através do *feedback* daqueles que utilizaram o sistema.

3.3 DELIMITAÇÕES

Tendo abordado até então todas as características da presente monografia como ela é; igualmente importante é definirmos como este trabalho *não é*. Sendo assim, as delimitações para a execução deste trabalho são:

- ✓ O protótipo para a visualização do sistema de recomendação será desenvolvido para web, o protótipo é importante para a coleta de dados dos usuários na forma de *feedback* explícito;
- ✓ O objetivo do trabalho é desenvolver um sistema que faça recomendações baseado na filtragem colaborativa utilizando uma abordagem de *Usuário – Usuário* (visto na sessão *2.2.2.1.1* da presente monografia). Dito isso, não será utilizada nenhuma outra forma de filtragem na implementação desse sistema de recomendação.

4 PROPOSTA DE SOLUÇÃO

Considerando que “há mais de um caminho para o topo da montanha” (TOKITSU, 2006), ditado japonês inventado por Miyamoto Musashi em 1645, o presente capítulo não busca explicar técnicas, ferramentas ou paradigmas no quesito linguagens de programação, apesar desses conceitos serem citados brevemente. Não. O objetivo deste capítulo é, portanto: introduzir a metodologia ICONIX: metodologia que, em sua fase inicial, visa explicitar requisitos de sistema, demonstrar protótipos de tela, casos de uso da aplicação, diagrama de robustez, diagrama de sequência e diagrama de domínio.

Também se fará útil a utilização da linguagem UML, destinada a modelagem – comumente utilizada em sistemas construídos em linguagens orientadas a objeto, mas não se limitando a estes – para uma melhor compreensão do protótipo desenvolvido.

A metodologia ICONIX propõe muitas das técnicas de desenvolvimento abordadas no presente capítulo, mas foi o estudo de mais literaturas ao longo do capítulo que diversificou a abordagem para melhor atender as necessidades do projeto.

4.1 ICONIX

A metodologia ICONIX começou a ser desenvolvida em 1993, com o objetivo de mesclar os melhores aspectos das três mais famosas metodologias orientada a objetos vigentes na época, que posteriormente formaram a base da UML (ROSENBERG; STEPHENS, 2007).

Em 2005 a obra “*Agile Development with ICONIX Process: people, process, and pragmatismo*” foi oficialmente publicada por Doug Rosenberg, fundador e presidente da Iconix Software Engineering Inc., juntamente com Matt Stephens e Mark Colling-Cope, ambos especialistas em projetos de desenvolvimento ágil, e com larga experiência em desenvolvimento, arquitetura, iteração e *design* de *softwares* (SBROCCO; MACEDO, 2012).

Nas palavras dos próprios autores, a metodologia ICONIX é minimalista e orientada a casos de uso. A obra de 2005 é repleta de conselhos, estes conselhos têm como objetivo evitar erros comuns no desenvolvimento ágil (ROSENBERG; STEPHENS; COLLINS-COPE, 2005).

Segundo Sbrocco e Macedo (2012) o processo ICONIX possui 8 fases, mas seguindo o próprio manifesto ágil original (BEEDLE et al., 2001), e na mesma mentalidade simplista de Dave Thomas (2015), que afirma que uma metodologia realmente "ágil" é uma que trata de adaptar-se ao longo de sua idealização e implementação, não será feito uso de todas as 8 fases da metodologia ICONIX explicadas no trabalho de Sbrocco e Macedo (2012), mas sim o que é de fato relevante e trará valor ao produto final da presente monografia, dado o contexto inserido[2].

No livro de Rosenberg, Stephens e Collins-Cope (2005) fica evidente que a abordagem que eles escolhem para demonstrar o ICONIX é voltada para sistemas desenvolvidos em linguagens como Java ou .NET/C#, linguagens orientadas a objeto. No capítulo 5 fica claro que a abordagem do protótipo foi diferente, mas o presente capítulo esclarece que a metodologia não se limita ao paradigma de programação que a linguagem de programação do projeto oferta.

4.1.1 Modelagem do domínio

Essa é a fase onde se identifica o domínio do problema e seus objetivos. Se faz necessário identificar e abstrair objetos de domínio de aplicação existente no mundo real, seus relacionamentos, generalizações e agregações, é possível fazer um diagrama de classes de alto nível. O objetivo da fase de modelagem de domínio é construir um modelo razoavelmente correto do domínio, e todos os envolvidos no projeto poderão compartilhar e usar comumente o sistema protótipo (SBROCCO; MACEDO, 2012).

[2] A metodologia ICONIX se direciona a times grandes e multidisciplinares, para projetos enormes e de longa duração – muitas vezes projetos cujo objetivo seja a continuidade e melhoria contínua do produto, enquanto o presente trabalho é elaborado por uma só pessoa e num determinado período de tempo, fazendo com que o escopo do trabalho, assim como a metodologia aplicada para sua realização, seja adaptado para as necessidades que o projeto apresenta. "*Não é a espécie mais intelectual que sobrevive; não é a [espécie] mais forte que sobrevive; mas sim [sobrevive] a espécie que melhor se adapta e se ajusta ao ambiente mutável em que se encontra*" (DARWIN, 1859)

4.1.2 Análise de robustez

A análise de robustez também possui a afinidade de descobrir eventuais ambiguidades, não dos requisitos, mas dos casos de uso representados. Esta análise é feita por um diagrama de robustez que representa um diagrama de classes estereotipado. Essa fase conecta a análise ao projeto, certificando-se do que a descrição dos casos de uso esteja correta. Nessa fase é possível descobrir novos objetos, que não foram vistos no modelo de domínio. (SBROCCO; MACEDO, 2012).

4.1.3 Comportamento dos objetos

Na fase de comportamento dos objetos se inicia o processo de utilização das definições da regra de negócio, previamente definidas utilizando casos de uso. Aqui o objetivo é usar um diagrama de sequência ou atividades (apesar desta monografia optar pelo primeiro). A ideia é utilizar uma representação gráfica da interação e resultados, mostrando como as ações executadas pelo sistema se comportam (SBROCCO; MACEDO, 2012).

4.2 UML

UML, ou Linguagem Unificada de Modelagem, é uma linguagem gráfica para visualização, especificação construção e documentação de artefatos de sistemas complexos de software (BOOCH, 2006). Booch (2006) ainda explica que pelo fato do número de métodos orientados a objeto do fim dos anos 80 e começo dos anos 90, muitos usuários tiveram dificuldades para encontrar uma linguagem de modelagem capaz de atender inteiramente às suas necessidades [...].

Larman (2007) afirma ser necessário uma linguagem para as “plantas de *software*”, tanto como ferramenta de raciocínio quanto ferramenta de comunicação. Larman em sua obra

questiona: “Suponha que é necessário escolher uma única habilidade prática dentre todos os tópicos discutidos aqui – uma habilidade a ser empregada numa ‘ilha deserta’. Qual seria ela?” A resposta que o autor fornece é: “atribuir, habilmente, responsabilidades aos objetos de *software*”. E em seguida explica que essa habilidade influencia drasticamente a robustez, a manutenibilidade e o reuso de componentes de *software*. Essa habilidade citada por Larman é influenciada pela criação do UML, assim como o UML é influenciado pela capacidade do desenvolvedor de atribuir essas responsabilidades.

Segundo Fowler (2005) uma UML válida é o que é definido como bem-formado na especificação. Na prática, no entanto, essa resposta se torna um pouco mais complicada. Mas como via de regra, Fowler afirma que se faz necessário ter em mente de que “um princípio geral na UML é de que qualquer informação pode ser **suprimida** de um diagrama em particular [...]. Em um diagrama, por tanto, nunca se deve inferir algo por sua ausência.”

4.3 PROTOTIPAÇÃO

Segundo Warfel (2009) um protótipo é um modelo representativo ou simulação do sistema final. E, diferentemente de um documento de requisitos ou *wireframe*, protótipos vão além e demonstram como o sistema será experienciado.

No caso de um sistema de recomendação, no trabalho de Ricci, Rokach e Shapira (2015) é proposto que podemos fazer uso do método científico para conduzir experimentos controlados e randomizados, o qual se chama testes *AB*.

Testes *AB*, como explicam Ricci, Rokach e Shapira (2015) utiliza dois grupos de indivíduos, o grupo *A* seria o *grupo de controle* que experienciaria o sistema como ele é atualmente e o grupo *B* teria a experiência da variação proposta numa nova solução. Para fazer tal experimento utilizam-se os seguintes passos: (a) **iniciar com uma hipótese**: isto é, um algoritmo/funcionalidade/*design* X irá crescer o engajamento do usuário com o serviço sendo exposto a teste e consequentemente teremos uma melhor retenção de usuários; (b) **projetar um teste**: aqui é considerado problemas como variáveis dependentes ou independentes, controle e significância; (c) **implementar um teste**: com base na projeção do teste, construir uma solução ou protótipo que execute num ambiente de produção (ou uma simulação de tal) onde o serviço possa servir requisições; (d) **executar teste**: designar diferentes usuários para diferentes

experiências e pedir para estes responder a diferentes experimentos; (e) **analisar o teste**: observar estatisticamente as mudanças significantes nas métricas do negócio, no caso do exemplo dado anteriormente: retenção de usuário, e tentar explicar estas mudanças, através da variação no comportamento das métricas de usuário (e. g. maior seleção das recomendações feitas pelo sistema de recomendação).

4.3.1 Levantamento de requisitos do sistema

No trabalho de Sbrocco e Macedo (2012) sobre a metodologia ICONIX os autores fornecem um exemplo (p. 176, cap. 12.4) que demonstra a metodologia em ação e com um passo a passo e dos oito passos ilustrados pelos autores, se faz uso dos passos descritos na sessão *4.1* da monografia: modelagem do domínio, análise de robustez, comportamento dos requisitos por meio de casos de uso e comportamento dos objetos.

Para Sommerville (2011, p. 57) “os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que [este sistema] oferece e as restrições em relação a seu funcionamento”. Sommerville observa que o termo ‘requisito’ não é utilizado de forma consistente na indústria de *software*: “em alguns casos, o requisito é apenas uma declaração abstrata de alto nível [...]. Em outro extremo, é uma definição detalhada e formal de uma função do sistema.” O detalhamento do que se deve ser feito e especificação de requisitos saem ligeiramente do escopo do ICONIX (SOUSA, 2013).

Quando se está desenvolvendo um sistema utilizando uma metodologia ágil como ICONIX, é importante manter em mente o adjetivo “**agilidade**” que o método se propõe a trazer. Se faz também de interesse do desenvolvedor observar mais de um ponto de vista em relação a dado tópico para que se cultive mais de uma perspectiva sobre o mesmo assunto.

4.3.1.1 Requisitos funcionais

"Na maioria dos processos de desenvolvimento, logo no início do projeto, tem-se um documento textual para expressar o que o sistema deve fazer" (SOUSA, 2013). Para Sousa, esse documento é bem importante, mas não facilita a análise dos requisitos funcionais e a posterior passagem para a etapa do projeto.

"Os requisitos funcionais de um sistema descrevem o que ele deve fazer. Eles dependem do tipo de *software* a ser desenvolvido, de quem são seus possíveis usuários e da abordagem geral adotada pela organização ao escrever os requisitos" (SOMMERVILLE, 2011, p. 57). Sommerville afirma que requisitos funcionais do sistema variam de requisitos gerais, que abrangem o que o sistema deve fazer, até requisitos mais específicos, que refletem os sistemas e as formas de trabalho em uma organização.

O Quadro 6 representa os requisitos funcionais definidos para o protótipo:

Quadro 6 – Requisitos funcionais do protótipo

Requisito	**Descrição**
RF001	O sistema deve permitir o usuário a criar uma conta.
RF002	O sistema deve permitir o usuário a efetuar *login* com e-mail e senha.
RF003	O sistema deve possibilitar o usuário a efetuar *login* utilizando o Google como método autenticador.
RF004	O sistema deve exigir autenticação para acessar os filmes a serem avaliados e recomendados.
RF005	O sistema deve permitir que o usuário avalie vários filmes.
RF006	O sistema deve permitir que o usuário busque um filme utilizando o nome do filme como termo de busca.
RF007	O sistema deve permitir que o usuário peça recomendações para o sistema através de um botão.
RF008	O sistema deve permitir que o usuário saia da aplicação.

Fonte: autoria própria (2023)

4.3.1.2 Requisitos não funcionais

A complexidade de um *software* é determinada em parte por sua funcionalidade, ou seja, o que o sistema faz, e em parte por requisitos gerais que fazem parte do desenvolvimento de *software* como custo, performance, confiabilidade, manutenibilidade, custos operacionais, entre outros. Estes requisitos podem ser chamados de requisitos não funcionais (CYSNEIROS, 2001).

Assim como Cysneiros (2001), Sommerville (2011, p. 60) afirma que os requisitos não funcionais estão relacionados às propriedades emergentes de um sistema, como confiabilidade, tempo de resposta e outros exemplos citados na literatura de Cysneiros.

O Quadro 7 representa os requisitos não funcionais definidos para o protótipo:

Quadro 7 – Requisitos não funcionais do protótipo

Requisito	**Descrição**
RNF001	O modelo de recomendação deve ser escrito em python versão 3.10 ou superior.
RNF002	A API que realizará o cadastro do usuário na base de dados deve ser escrito em TypeScript versão 4.9.5 ou superior.
RNF004	O cliente deve ser desenvolvido utilizando NextJS versão 13 ou superior.
RNF005	O cliente web deve funcionar nos navegadores Google Chrome, Mozilla Firefox e Microsoft Edge.
RNF006	A infraestrutura do protótipo deve seguir o princípio de baixo acoplamento.
RNF007	Toda a infraestrutura do projeto deve estar hospedada no serviço de nuvem pública AWS (Amazon Web Services).
RNF008	Todos os dados a serem persistidos no sistema devem estar numa instância do MongoDB.
RNF009	Toda comunicação entre as aplicações deve seguir o padrão RESTful.
RNF010	A interface deve possuir um caráter simples e objetivo.

Fonte: autoria própria (2023)

4.3.1.3 Regras de negócio

As regras [de negócio] normalmente estão ligadas a leis que regem o negócio e descrevem de forma complementar o funcionamento de seus processos – por isso, muitas vezes são chamadas de regras de negócio [...]. Elas não são especificamente ligadas à solução – e, portanto, ao *software* – mas ao domínio do problema em que estas regras se inserem (VAZQUEZ; SIMÕES, 2016).

Algumas das razões que o negócio precisa de regras inclui, mas não se limita a: minimização de riscos, redução de custos, proteção, etc. (WITT, 2012). Witt (2012) afirma que as regras que governam o negócio precisam ser documentadas e de fácil acesso aos *stakeholders*, a fim de que essas regras sejam revisadas, considerando a relevância e concordância [da regra]. Em seu trabalho: "*Writing Effective Business Rules*" (2012), Witt afirma que os três principais problemas que a maioria das organizações enfrentam são: (1) suas regras [de negócio] não respeitam um formato padrão; (2) suas regras não são acessíveis aos *stakeholders*; e (3) o processo de alteração dessas regras é complexo.

O Quadro 8 representa as regras de negócio que regem o protótipo:

Quadro 8 – Regras de negócio do protótipo

Regra	**Descrição**
RN001	Ao cadastrar-se no sistema (UC001), o usuário deve criar uma senha com pelo menos 6 caracteres.
RN002	Quando o usuário se cadastrar no sistema (UC001), o sistema deve verificar se o e-mail utilizado pelo usuário é único na base de dados.
RN003	Quando o usuário avaliar um filme (UC004), a nota da avaliação deve variar entre 0.5 e 5.
RN004	Quando o usuário solicitar recomendações de filmes (UC005), o sistema deve recomendar filmes que o usuário ainda não avaliou.

Fonte: autoria própria (2023)

Witt (2012) afirma que os negócios, de modo geral, possuem sinônimos: 'passageiro', 'usuário', 'convidado', 'cliente'... no nosso sistema de recomendação é utilizado os termos 'usuário', assim como o 'sistema' que este usuário interage.

4.3.2 Levantamento de requisitos do sistema

"Se o negócio envolve a criação de um *website*, aplicação de *software*, ou sistemas que possuem componentes tanto *software* quanto *hardware*, você não pode deixar de prototipar" (WARFEL, 2009, pag. 3; cap. 4, tradução própria).

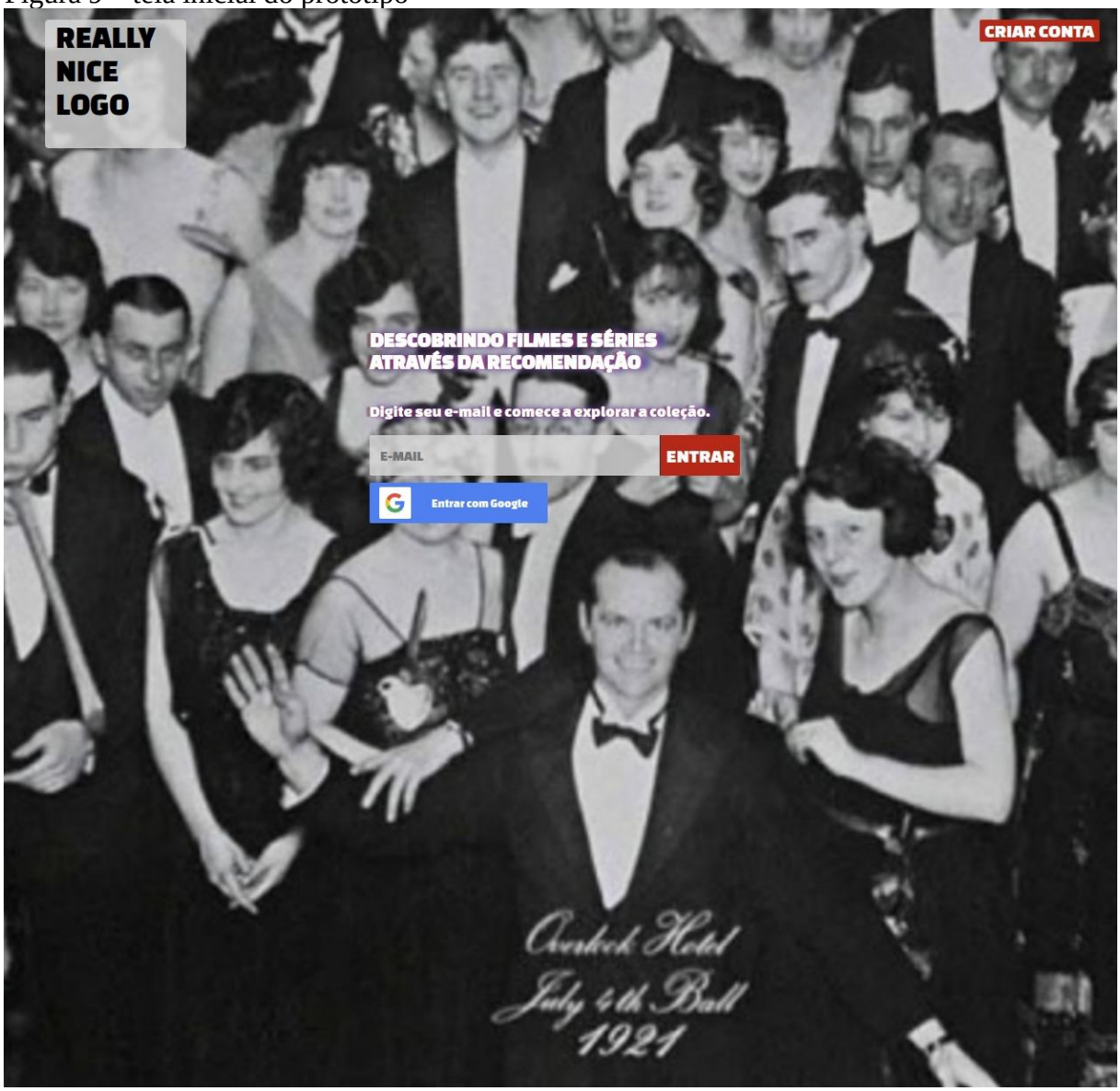
> O cliente frequentemente define um conjunto de objetivos gerais para o *software*, mas não identifica detalhadamente requisitos de entrada, processamento ou saída. Em outros casos, o desenvolvedor pode estar inseguro da eficiência de um algoritmo, da adaptabilidade de um sistema operacional ou da forma que a interação homem/máquina deve assumir. Nessas e em muitas outras situações, um paradigma de prototipagem pode oferecer a melhor abordagem. [...] O projeto rápido parte de um protótipo. (SBROCCO; MACEDO, 2012, p. 62)

Apesar do foco da monografia estar longe de ser voltado à experiência de usuário no que diz interface de tela, o protótipo precisar ser – no mínimo – apresentável. As telas do sistema seguem os conceitos básicos de UI/UX fornecidos por Werfel (2009) e mais ainda em especial o livro "*UX for the Web: Build websites for user experience and usability*" (WINTERBOTTOM; RITTER, 2017), onde o capítulo 1: "*The Fundamentals of UX*" serviu de linha guia para o desenvolvimento do protótipo como um todo.

Os protótipos de tela a seguir demonstram as telas que o usuário acessará em sua jornada pela aplicação web.

A Figura 9 mostra a tela inicial do protótipo:

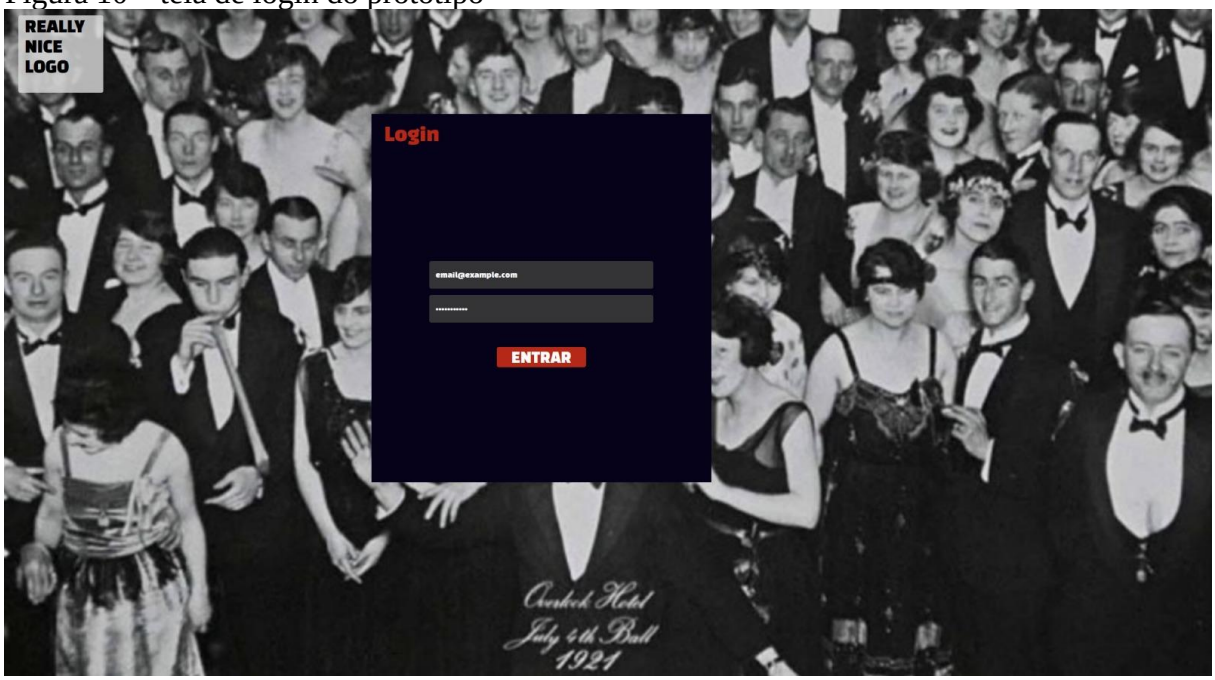
Figura 9 – tela inicial do protótipo



Fonte: autoria própria (2023)

Na Figura 9 é possível ver que há um botão de ENTRAR junto ao *input* de e-mail, e caso o usuário preencha um e-mail não cadastrado no banco de dados ele será redirecionado para tela de cadastro (Figura 11), caso contrário, o usuário é redirecionado a tela de *login*, onde preenche a senha – esse fluxo de *login* e cadastro é inspirado no site oficial da *Netflix* (Figura 10):

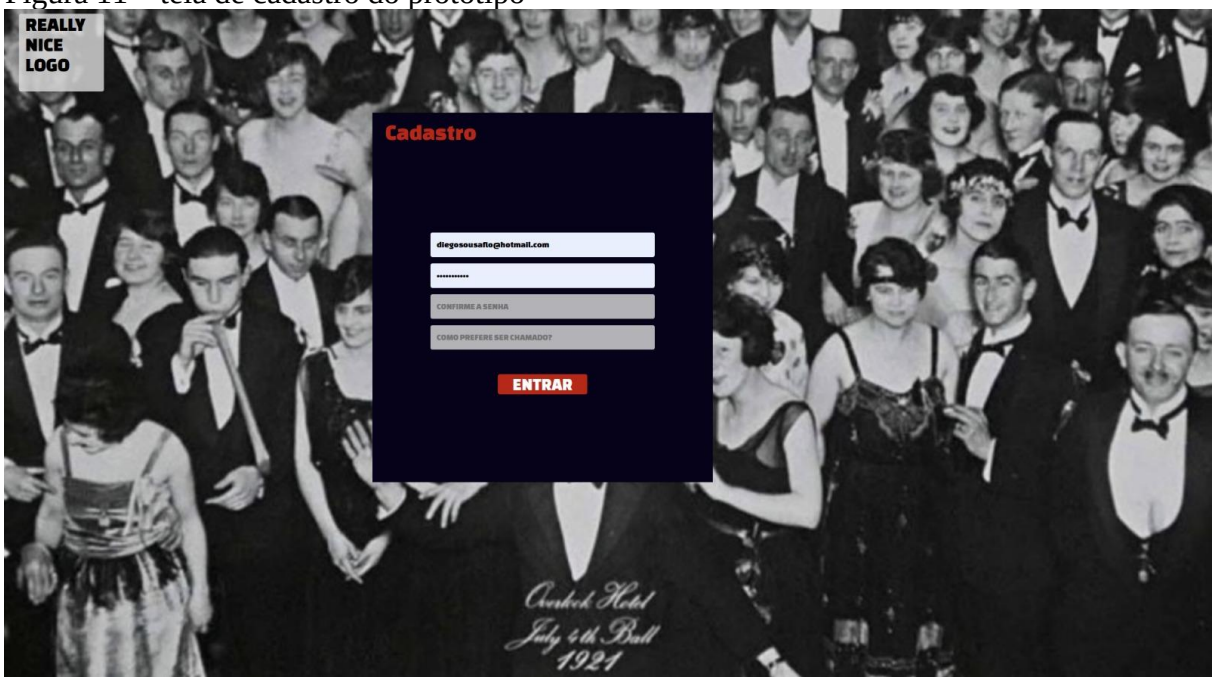
Figura 10 – tela de login do protótipo



Fonte: autoria própria (2023)

A tela de login é autoexplicativa, o e-mail do usuário já vem preenchido com o e-mail utilizado na tela inicial (Figura 9), e o usuário precisa colocar a senha correta.

Figura 11 – tela de cadastro do protótipo

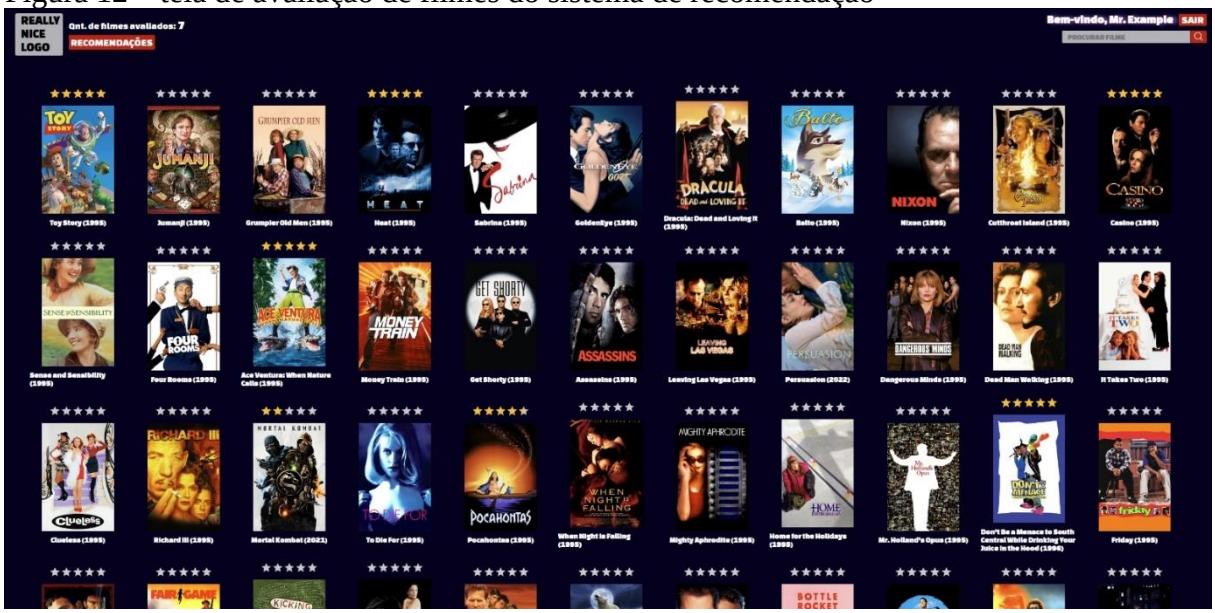


Fonte: autoria própria (2023)

Na tela de cadastro (Figura 11) é realizada a coleta de todos os dados necessários para criação da conta do usuário: e-mail, senha e como este usuário prefere que seu nome seja

mostrado. Considerando a filtragem colaborativa baseada em Usuário – Usuário não precisaremos de outros dados, pois os dados utilizados para recomendação serão as avaliações que este usuário atribuirá na próxima tela (Figura 12):

Figura 12 – tela de avaliação de filmes do sistema de recomendação



Fonte: autoria própria (2023)

No canto superior direito está a barra de buscas por filmes, e logo acima o botão “SAIR”, que direciona o usuário para a tela inicial do protótipo (Figura 12). No lado esquerdo superior está a quantidade de filmes avaliados por aquele usuário e um botão de “RECOMENDAÇÕES”, ao clicar nesse botão o modelo de recomendação irá enviar os Ids dos filmes recomendados ao cliente e o cliente irá listar esses filmes para o usuário. No restante da tela estão os filmes: no topo de cada filme estão 5 estrelas para a avaliação, a capa do filme e a baixo o nome do filme com o ano de lançamento entre parênteses.

O *design* do site em si não segue à risca um padrão, visto que um padrão de *design* no quesito UI/UX não é pré-requisito para o protótipo, mas definitivamente é algo a ser melhorado no futuro – mas sem deixar a estética *retro* de lado, definitivamente.

4.3.3 Casos de uso

Os casos de uso são uma técnica de descoberta de requisitos introduzida inicialmente no método *Objectory* (JACOBSON *et al.*, 1993 *apud* SOMMERVILLE, 2011). Sommerville (2011, p. 74) afirma que em sua forma mais simples, um caso de uso identifica os atores envolvidos em uma interação e dá o nome ao tipo de interação. Essa [interação] é, então, suplementada por informações adicionais que descrevem a interação com o sistema.

Segundo Sousa (2013) o ICONIX minimiza a paralisia da análise, ignorando a semântica de estereótipos do padrão UML, tais como << *extend* >> e << *include* >>, etc. que faz o desenvolvedor perder tempo, retardando a passagem da análise para o projeto.

Para que fique claro os casos de uso serão apresentados como diagramas e quadros, no trabalho de Cockburn (2005): "*Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software*", o autor demonstra detalhadamente o processo de ideação de casos de uso para um sistema:

Figura 13 – Quadro de casos de uso, como demonstrado por Cockburn (2005)

CASO DE USO 1 Comprar Ações na Internet

Ator Primário: Comprador
Escopo: Conselheiro Particular / Pacote Financeiro (PAF)
Nível: Objetivo do usuário
***Stakeholders* e Interesses:**
Comprador – deseja comprar ações e tê-las adicionadas ao portfólio PAF automaticamente.
Agência de ações – deseja toda informação da compra.
Pré-condição: Usuário já está com o PAF aberto.
Garantia Mínima: Existirá informação de registro suficiente para que o PAF possa detectar que algo errado aconteceu e pedir para o usuário fornecer detalhes.
Garantia de Sucesso: O *Website* remoto confirmou a compra; o registro e o portfólio do usuário são atualizados.
Cenário de Sucesso Principal:
1. Comprador seleciona comprar ações na Internet.
2. PAF obtém o nome do *Website* para usar (E*Trade, Schwab, etc.) do usuário.
3. PAF abre uma conexão Web com o *site*, mantendo controle.
4. Comprador navega e compra ações no *site*.
5. PAF intercepta respostas do *site* na Internet e atualiza o portfólio do comprador.
6. PAF mostra ao usuário a nova posição do portfólio.

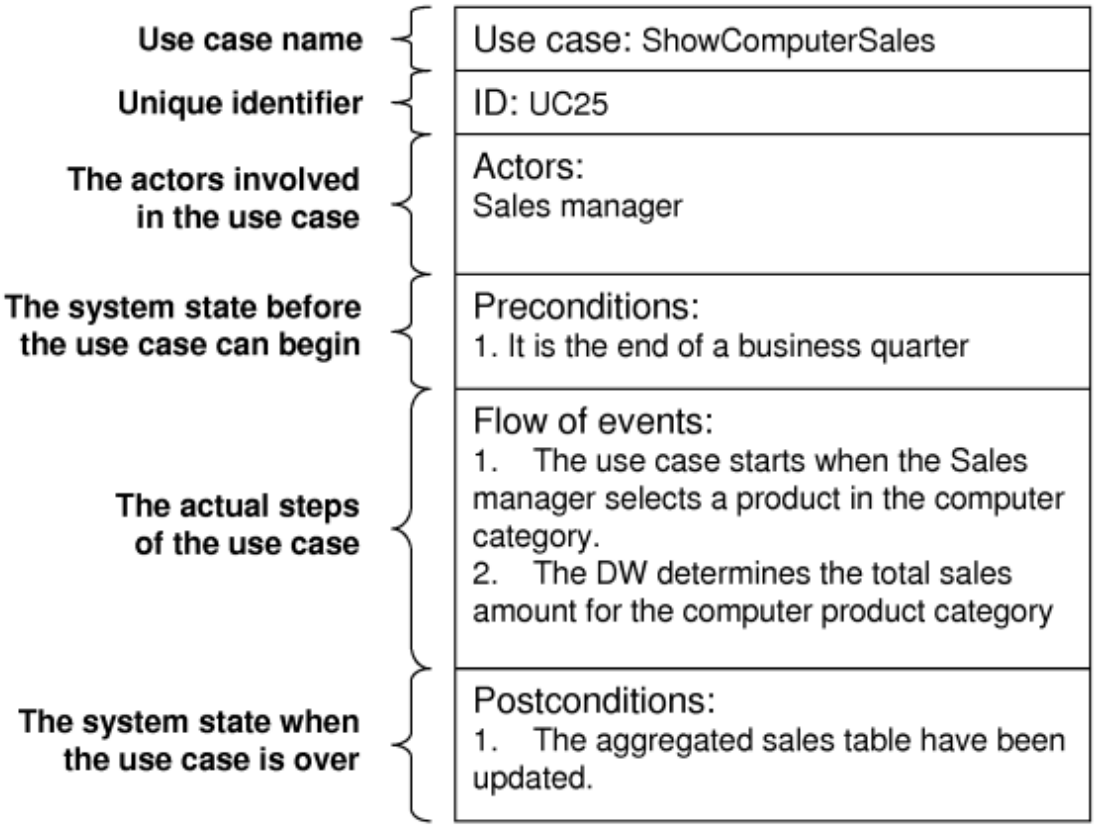
Extensões:
2a. Comprador deseja um *Website* que o PAF não suporta:
2a1. Sistema obtém nova sugestão do comprador, com a opção de cancelar o caso de uso.
3a. Algum tipo de falha na Internet durante a configuração:
3a1. Sistema informa a falha ao comprador com aviso, retorna ao passo anterior.
3a2. Comprador ou sai do caso de uso, ou tenta novamente.
4a. Computador estraga ou é desligado durante a transação de compra:
4a1. (O que fazemos aqui?)
4b. *Website* não confirma a compra, mas a põe em espera:
4b1. PAF registra a espera, ajusta um cronômetro para perguntar ao comprador sobre o resultado.
5a. *Website* não retorna a informação necessária para a compra:
5a1. PAF registra a falta de informação, no log, o comprador recebe atualize compra questionada.

Fonte: *Escrevendo Casos de Usos Eficazes: Um guia prático para desenvolvedores de software* (COCKBURN, 2005, p. 23)

Como afirma Cockburn (2005): “cada situação requer um estilo de escrita levemente diferente [...]”, e com base no contexto da presente monografia, se faz uso do modelo de escrita de casos de uso de Cockburn de forma objetiva, simplificada e mais alinhada com a metodologia ICONIX que não utiliza uma forma padrão de escrita em seus casos de uso (SOUSA, 2013, p. 38).

Um excelente modelo que ilustra com maestria os casos de uso de forma limpa e organizada é o modelo de casos de uso fornecido por Luján-Mora (2005) em sua tese de doutorado, apresentado na Figura 14:

Figura 14 – modelo de casos de uso proposto por Luján-Mora



Fonte: *Data Warehouse Design with UML* (LUJÁN-MORA, 2005, p.39)

Seguindo o modelo de Luján-Mora, estes são os casos de uso do sistema de recomendação (Quadros 9 – 13):

Quadro 9 – Caso de uso UC001

UC001:	Criação de conta de usuário no sistema de recomendação.
Ator:	Usuário.
Pré-condições:	Nenhuma.
Objetivo:	O ator cria uma conta para entrar no sistema.
Fluxo principal:	**Passo 1:** O ator acessa a página inicial do sistema. **Passo 2:** O ator insere e-mail, senha, confirmação de senha e como prefere ser chamado em seus respectivos campos. **Passo 3:** O ator aperta no botão “ENTRAR”.
Fluxo alternativo:	**Passo 1:** O ator acessa a página inicial do sistema. **Passo 2:** O ator clica no botão “*login* com Google”. **Passo 3:** O ator é redirecionado para a tela de *login* com Google. **Passo 4:** O ator completa o fluxo de *login* providenciado pela Google.
Pós condições:	O ator é redirecionado para a tela de avaliação de filmes.

Fonte: autoria própria (2023)

Quadro 10 – Caso de uso UC002

UC002:	*Login* no sistema de recomendação via e-mail e senha.
Ator:	Usuário.
Pré-condições:	O ator possui uma conta cadastrada no sistema.
Objetivo:	O ator utilizou suas credenciais para entrar no sistema.
Fluxo principal:	**Passo 1:** O ator acessa a página inicial do sistema. **Passo 2:** O ator insere e-mail e senha em seus respectivos campos. **Passo 3:** O ator aperta no botão “ENTRAR” e é redirecionado para a tela de avaliação de filmes.
Fluxo alternativo:	**Passo 1:** O ator acessa a página inicial do sistema. **Passo 2:** O ator clica no botão “*login* com Google”. **Passo 3:** O ator é redirecionado para a tela de *login* com Google. **Passo 4:** O ator completa o fluxo de *login* providenciado pela Google.
Pós condições:	O ator é redirecionado para a tela de avaliação de filmes.

Fonte: autoria própria (2023)

Quadro 11 – Caso de uso UC003

UC004:	Avaliar filme.
Ator:	Usuário.
Pré-condições:	Estar *logado* no sistema.
Objetivo:	O ator avalia um filme de 0.5 à 5 estrelas.
Fluxo principal:	**Passo 1:** O usuário acessa a página de avaliação de filmes. **Passo 2:** O usuário clica na nota que deseja atribuir ao filme utilizando as estrelas que estão acima do respectivo filme.
Pós condições:	A nota do ator persiste num banco de dados.

Fonte: autoria própria (2023)

Quadro 12 – Caso de uso UC004

UC005:	Solicitar recomendações ao sistema.
Ator:	Usuário.
Pré-condições:	Estar *logado* no sistema.
Objetivo:	O ator solicita recomendações ao sistema.
Fluxo principal:	**Passo 1:** O ator acessa a página de avaliação de filmes. **Passo 2:** O ator clica no botão “RECOMENDAÇÕES”.
Pós condições:	O ator visualiza os filmes recomendado pelo sistema.

Fonte: autoria própria (2023)

Quadro 13 – Caso de uso UC005

UC006:	Sair do sistema.
Ator:	Usuário.
Pré-condições:	Estar *logado* no sistema.
Objetivo:	O ator sai do sistema de recomendação.
Fluxo principal:	**Passo 1:** O ator acessa a página de avaliação de filmes. **Passo 2:** O ator clica no botão “SAIR”.
Pós condições:	O ator é redirecionado para tela inicial e precisar efetuar *login* novamente se quiser acessar a tela de avaliação e recomendação.

Fonte: autoria própria (2023)

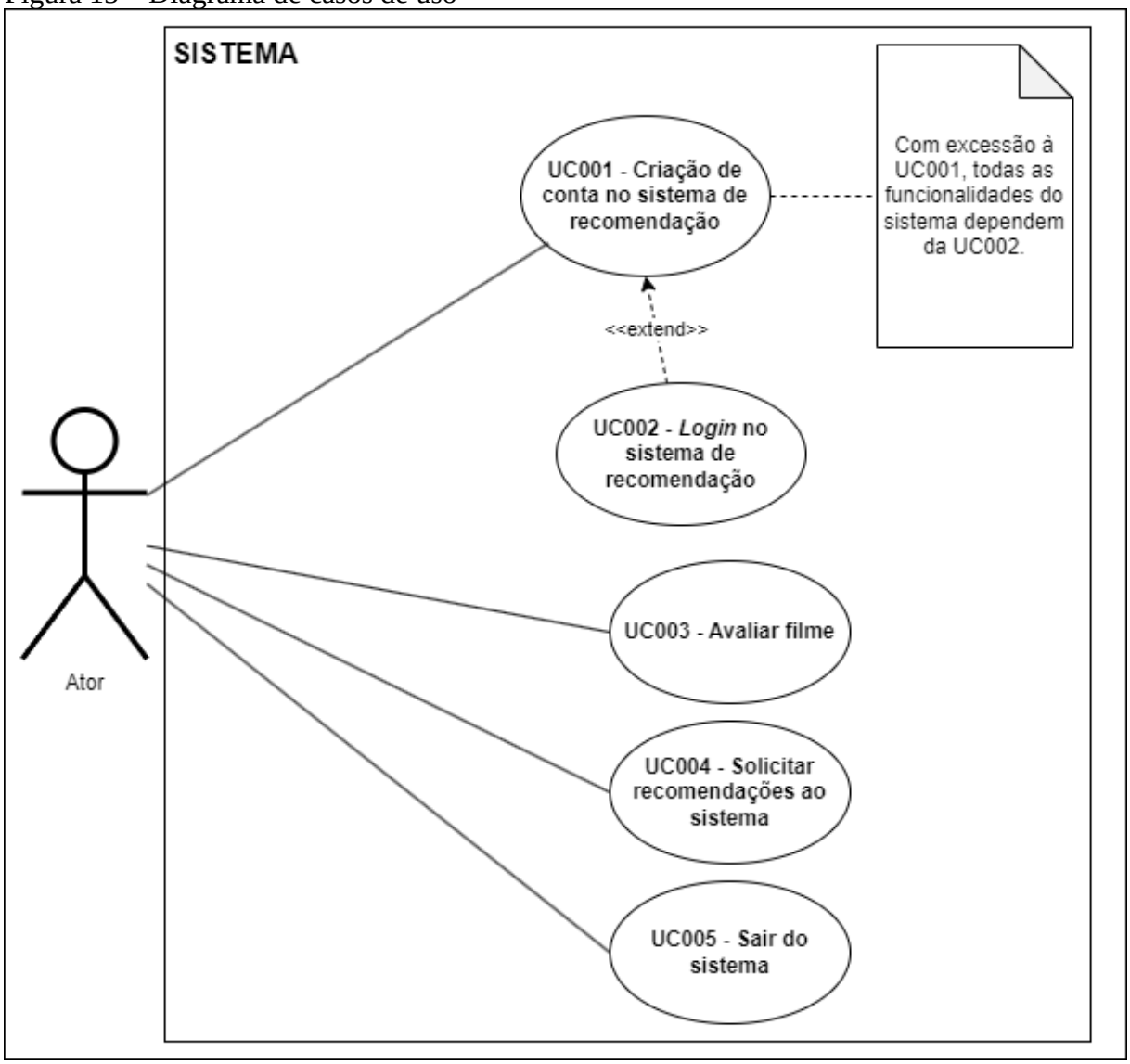
Os quadros acima (Quadros 9 – 13) demonstram os casos de uso da aplicação proposta para demonstração dos conhecimentos obtidos na construção da presente monografia.

Observando os casos de uso, os Quadros 9 e 10 possuem o mesmo fluxo alternativo, isso se dá ao fato de que ao realizar *login* com o Google o usuário é automaticamente cadastrado, o sistema de recomendação torna transparente esse cadastro na base de dados, mas "por baixo dos panos" o primeiro *login* do usuário utilizando Google é, na verdade, um cadastro de usuário na base de dados do sistema de recomendação.

Segundo Sommerville (2011) algumas pessoas consideram cada caso de uso um cenário único; outros encapsulam um conjunto de cenários em um único caso de uso. Mas como afirma Sommerville (2011) "você pode, na prática, usá-los de qualquer forma", o que deixa claro que os quadros e diagramas de casos de uso estão abertos para misturar e combinar diferentes maneiras de organização de tais informações, desde que o objetivo seja sempre a fácil compreensão dos requisitos do usuário.

Sousa (2013) afirma que no ICONIX os casos de uso devem ser organizados através de um diagrama de casos de uso, no caso do protótipo sendo desenvolvido tal diagrama é desenvolvido de maneira simplista, como demonstra a Figura 15:

Figura 15 – Diagrama de casos de uso



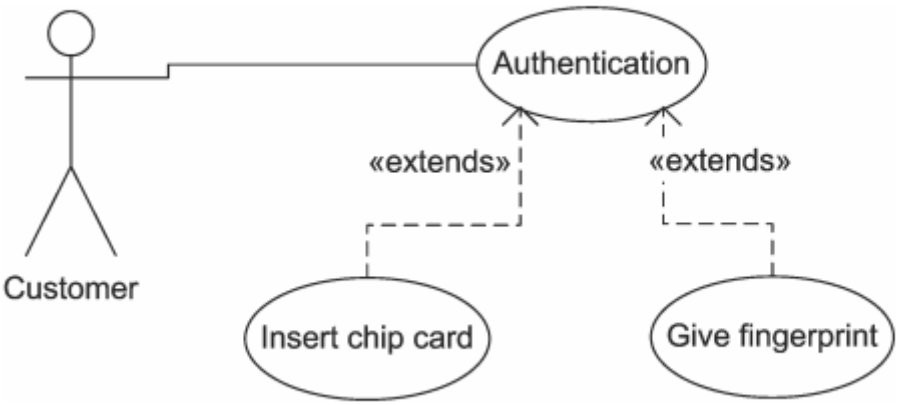
Fonte: autoria própria (2023)

Assim como os quadros que representam os casos de uso, o diagrama de casos de uso pode ser feito de qualquer forma, desde que atenda o requisito de informar de forma clara e objetiva os casos de uso de um sistema. Apesar de Sousa (2013) afirmar não ser necessário a utilização de notações como <<include>> ou <<extend>>, Rosenberg e Scott (2001) recomendam em seu trabalho "*Use Case Driven Object Modeling with UML*" a utilização desses estereótipos do UML para uma melhor leitura do diagrama. Considerando que o objetivo do diagrama de casos de uso é a melhor compreensão dos casos de uso em si, foi feito uso desses estereótipos recomendado por Rosenberg e Scott.

Casos de uso podem estar relacionados a outros casos de uso utilizando os relacionamentos *extend*, *include* ou *generalization*, exemplificado na Figura 16. O <<extend>> implica que o relacionamento que aquele caso de uso pode estender um comportamento descrito

em outro caso de uso; O <<include>> significa que o caso de uso inclui o comportamento descrito em outro caso de uso; O <<generalization>> entre casos de uso significa que a criança é uma forma específica do caso de uso pai (SCHMID, 2002).

Figura 16 – exemplo de diagrama de caso de uso

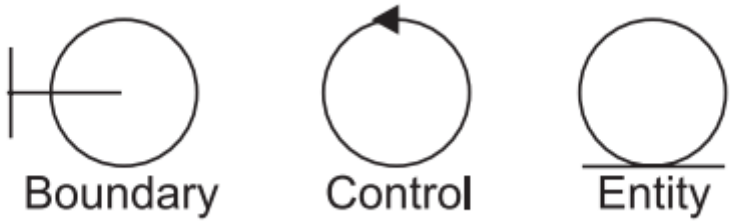


Fonte: *International Workshop on Requirements Engineering for Product Lines* (SCHMID, 2002, p. 28)

4.3.4 Diagrama de robustez

Trata-se de um diagrama aderente às arquiteturas atuais do MVC – Modelo, Visão e Controle, que auxilia o desenvolvedor, gerenciando, visualizando e persistindo dados de uma aplicação. A palavra ‘robustez’ indica a utilização de um método capaz de traçar passo a passo toda a interação do usuário com o sistema. Envolve um conjunto de símbolos representativos que indicam a interface utilizada, os menus e possíveis resultados obtidos, facilitando assim o entendimento e a identificação dos objetos e ações propostas pelo sistema. O diagrama de robustez é composto por três símbolos (SBROCCO; MACEDO, 2012):

Figura 17 – Símbolos do diagrama de robustez

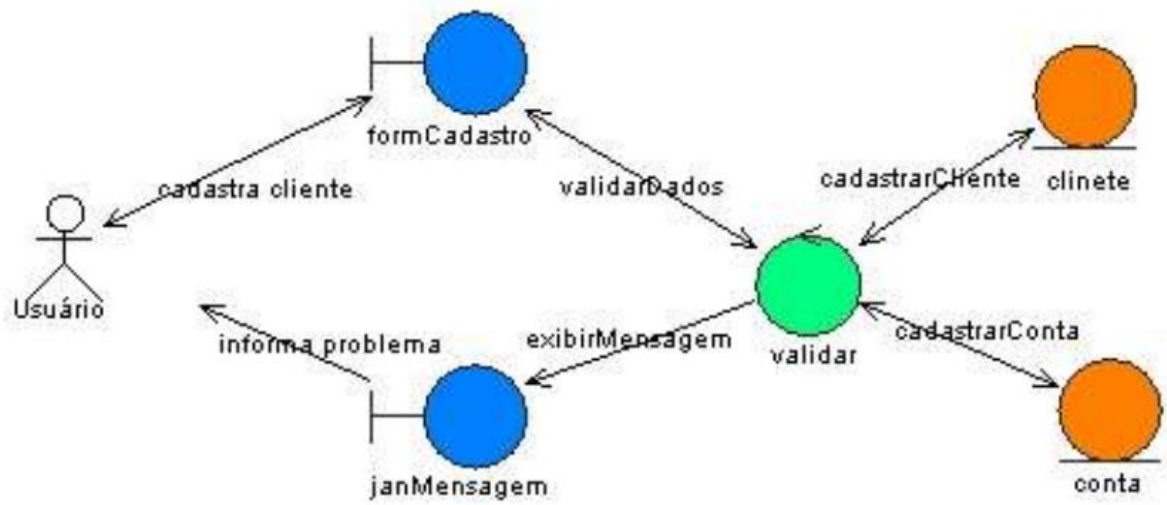


Fonte: *Metodologias Ágeis: Engenharia de Software sob Medida* (SBROCCO; MACEDO, 2012, p. 175)

Os símbolos da Figura 17 se traduzem como: *Boundary* (ou Limite), refere-se à interface que será oferecida ao usuário do sistema; *Control* (ou Controle), refere-se ao controlador com as regras de negócio do sistema; e *Entity* (ou Entidade), refere-se ao armazenamento dos dados provenientes do sistema (SBROCCO; MACEDO, 2012).

Maia (2016) afirma que o diagrama de robustez possui quatro regras: (1) Atores (que pode ser lido também como o ‘usuário’ do sistema de recomendação) somente podem se comunicar com objetos de Limite; (2) Limites podem se comunicar somente com Atores e Controles; (3) Entidades só podem se comunicar com Controles; (4) Controles pode se comunicar somente com Entidade e Limite, e também com outros controladores, mas não com Atores. A Figura 18 servirá de exemplo para o diagrama de robustez do protótipo do sistema de recomendação.

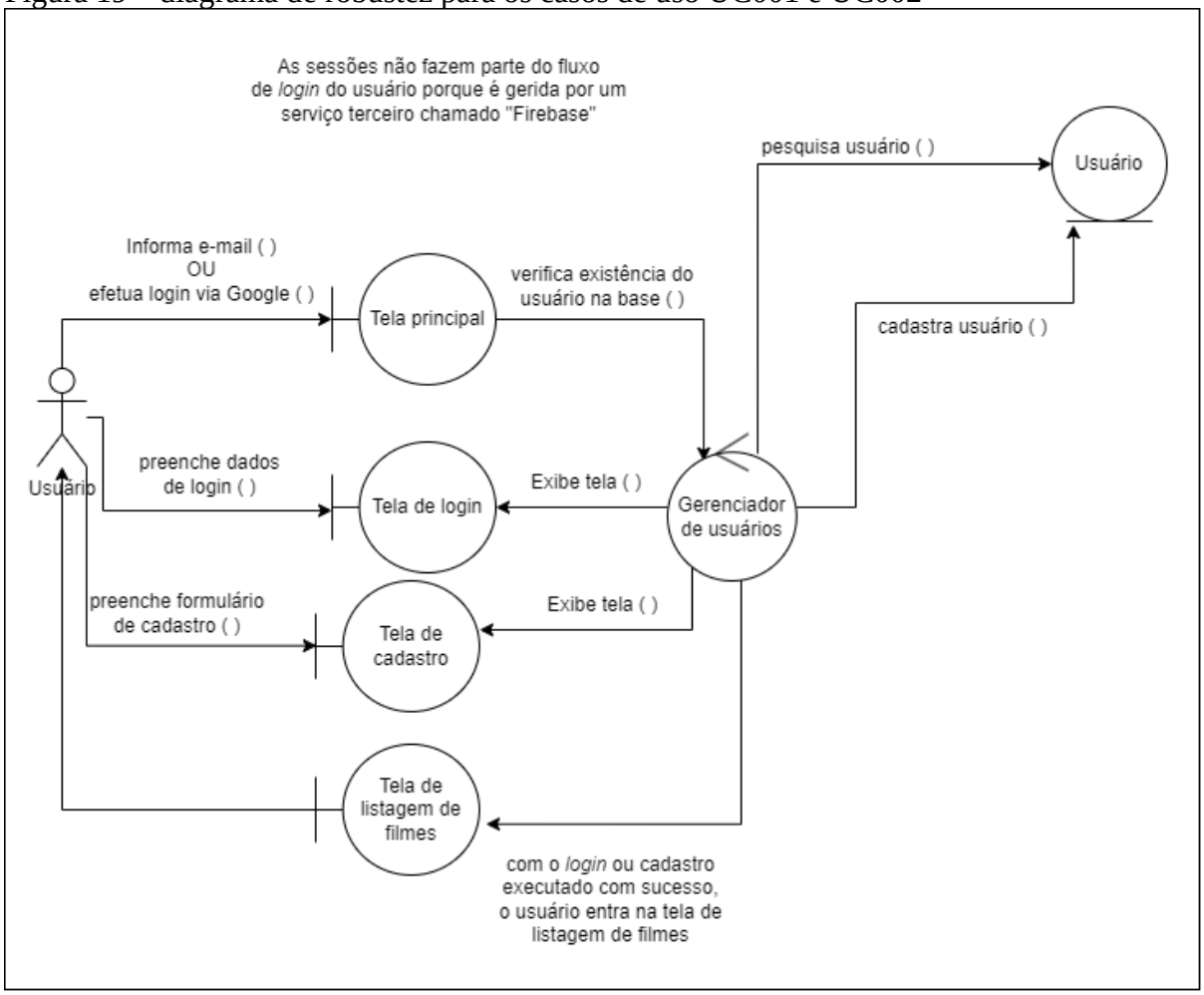
Figura 18 – exemplo de diagrama de robustez



Fonte: *Construindo Softwares com Qualidade e Rapidez Usando ICONIX* (MAIA, 2016, p. 10)

Para os diagramas de robustez do sistema de recomendação ficarem sucintos, estes estão separados em partes:

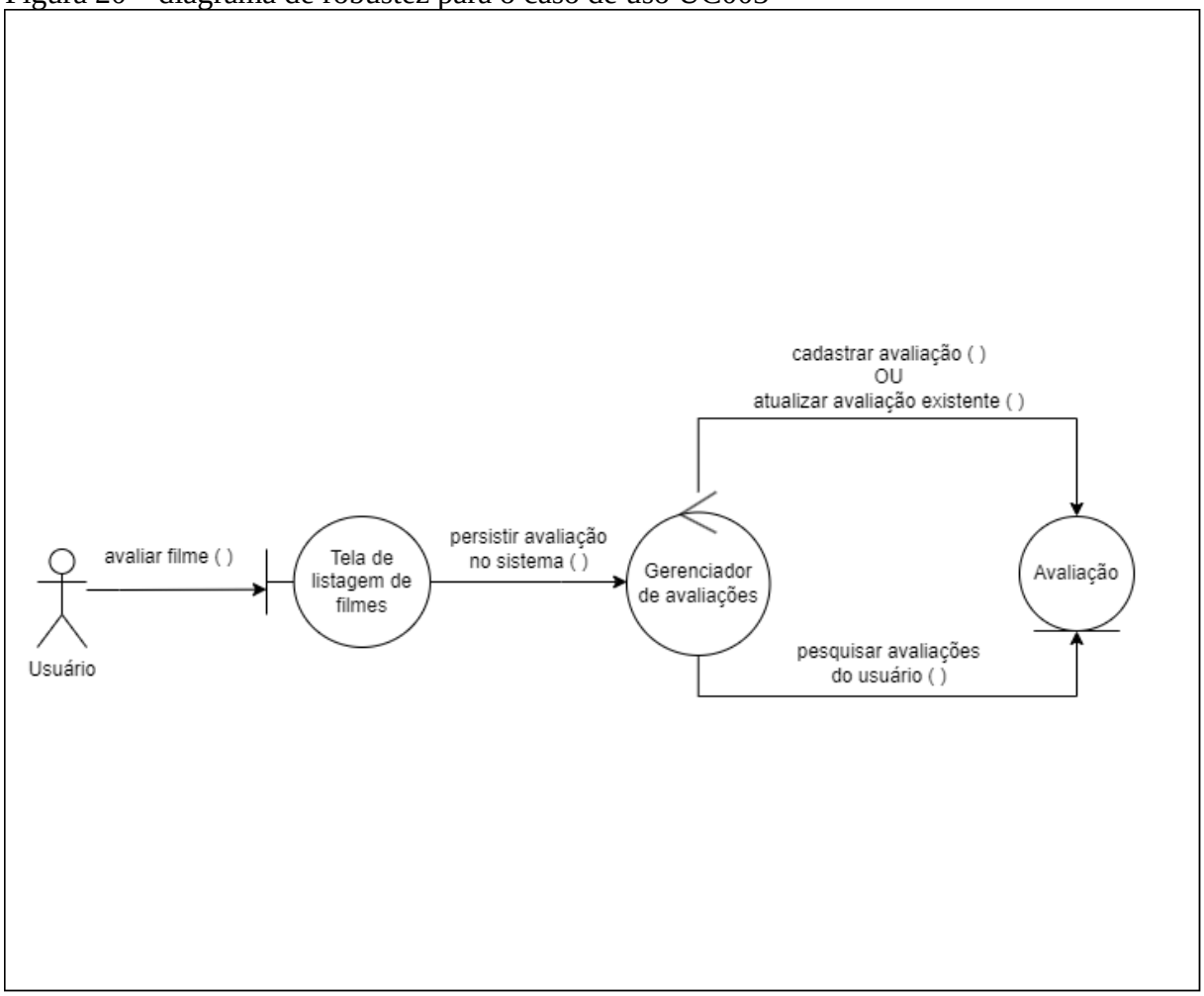
Figura 19 – diagrama de robustez para os casos de uso UC001 e UC002



Fonte: autoria própria (2023)

A Figura 19 demonstra o diagrama de robustez para os casos de uso UC001 e UC002, ambos possuem o objetivo de gerenciar a conta do usuário, portanto faz sentido eles serem exibidos no mesmo diagrama. Além dos itens presentes nesse diagrama, é importante observar o que não se faz presente neste e nos demais diagramas de robustez: um sistema de gerenciamento de sessão adequado. Essa ausência se dá por conta do escopo do trabalho, o qual se propõe a fazer recomendações de filmes e não garantir a devida integridade dos serviços envolvidos nessa recomendação.

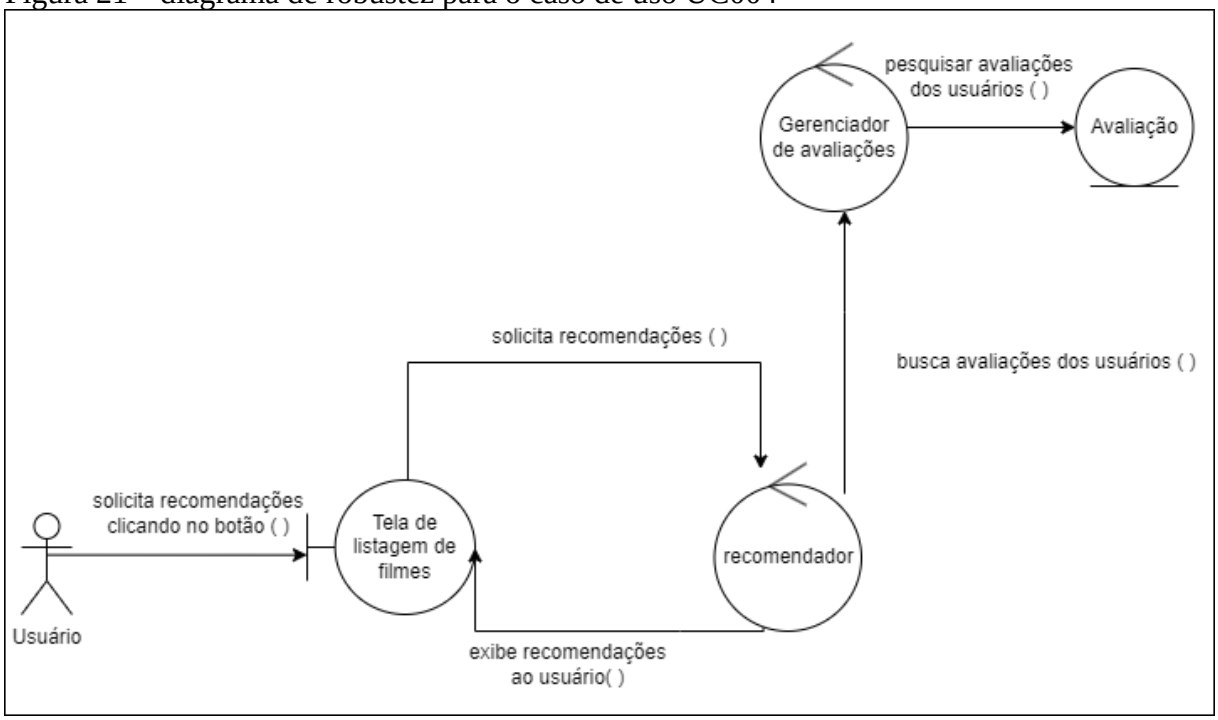
Figura 20 – diagrama de robustez para o caso de uso UC003



Fonte: autoria própria (2023)

O diagrama na Figura 20 que ilustra o caso de uso UC003 é bem direto ao ponto: o usuário avalia o filme e o sistema cadastra ou atualiza a avaliação, para saber qual das duas ações performar o sistema deve checar a existência da avaliação antes de cria-la.

Figura 21 – diagrama de robustez para o caso de uso UC004

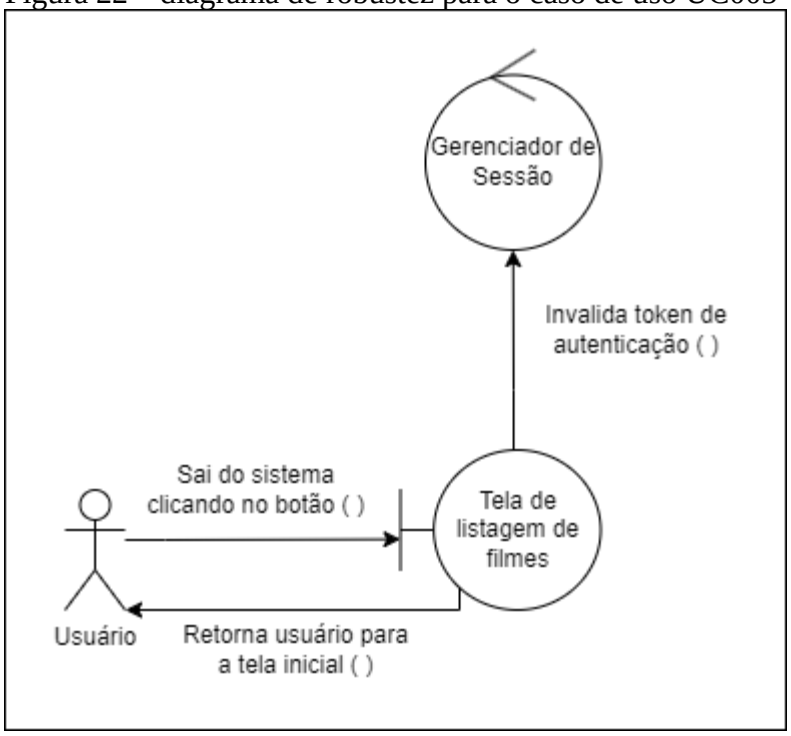


Fonte: autoria própria (2023)

A Figura 21 mostra o diagrama do caso de uso mais importante do sistema: a recomendação de filmes ao usuário (caso de uso UC004). No capítulo 5: desenvolvimento fica explícito como os dados do conjunto de dados pré-definido e os dados armazenados no banco de dados da aplicação se interagem, essa interação entre dois conjuntos de dados serve para mitigar uma das dificuldades enfrentadas nos sistemas de recomendação, vista no capítulo 2.2.4.2, *cold start* (ou arranque a frio).

Outra coisa a se observar é o fato de que o serviço de recomendação não interage diretamente com a entidade Avaliação, e sim solicita os dados ao Gerenciador de avaliações, isso se dá por conta da RNF006 que exige o baixo acoplamento dos serviços.

Figura 22 – diagrama de robustez para o caso de uso UC005



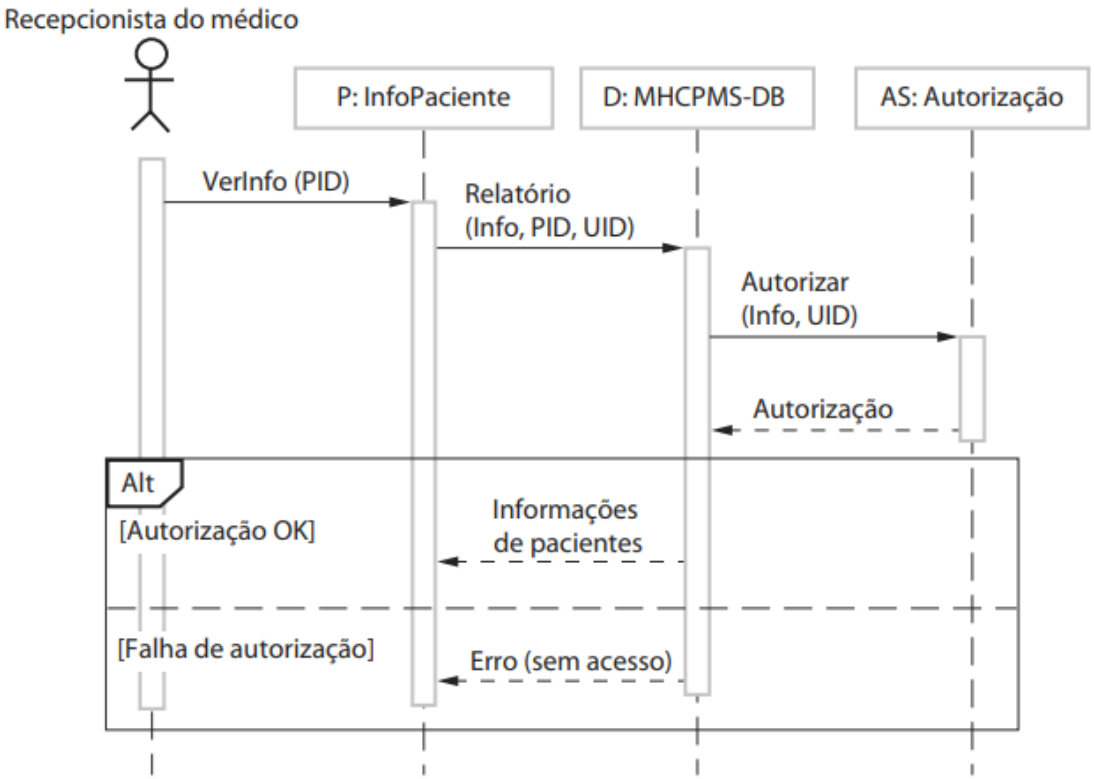
Fonte: autoria própria (2023)

Como citado anteriormente: a sessão não é validada de maneira adequada, mas ainda há uma validação feita do lado do cliente (ou em outras palavras: internamente no navegador), como demonstrado na Figura 22. Essa sessão não expira por si só, mas quando o usuário solicita a saída do sistema de recomendação este não mais acessa a tela de listagem de filmes sem os UC001 ou UC002, referentes a cadastro e *login* no sistema.

4.3.5 Diagrama de sequência

Para Sommerville (2011) os modelos de caso de uso e diagrama de sequência podem ser usados juntos, pois apresentam diferentes níveis de detalhamento do sistema. O autor Sommerville (2011) afirma que “diagrama de sequência, como o nome indica, mostra a sequência de interações que ocorrem durante um caso de uso em particular ou em uma instância de caso de uso. Na literatura de Sommerville há um exemplo claro de como um diagrama de sequência pode ser escrito, dado o caso de uso, conforme mostra a Figura 23:

Figura 23 – modelo de diagrama de sequência do caso de uso fictício “Ver informações de pacientes”.



Fonte: *Engenharia de Software* (SOMMERVILLE, 2011, p. 88)

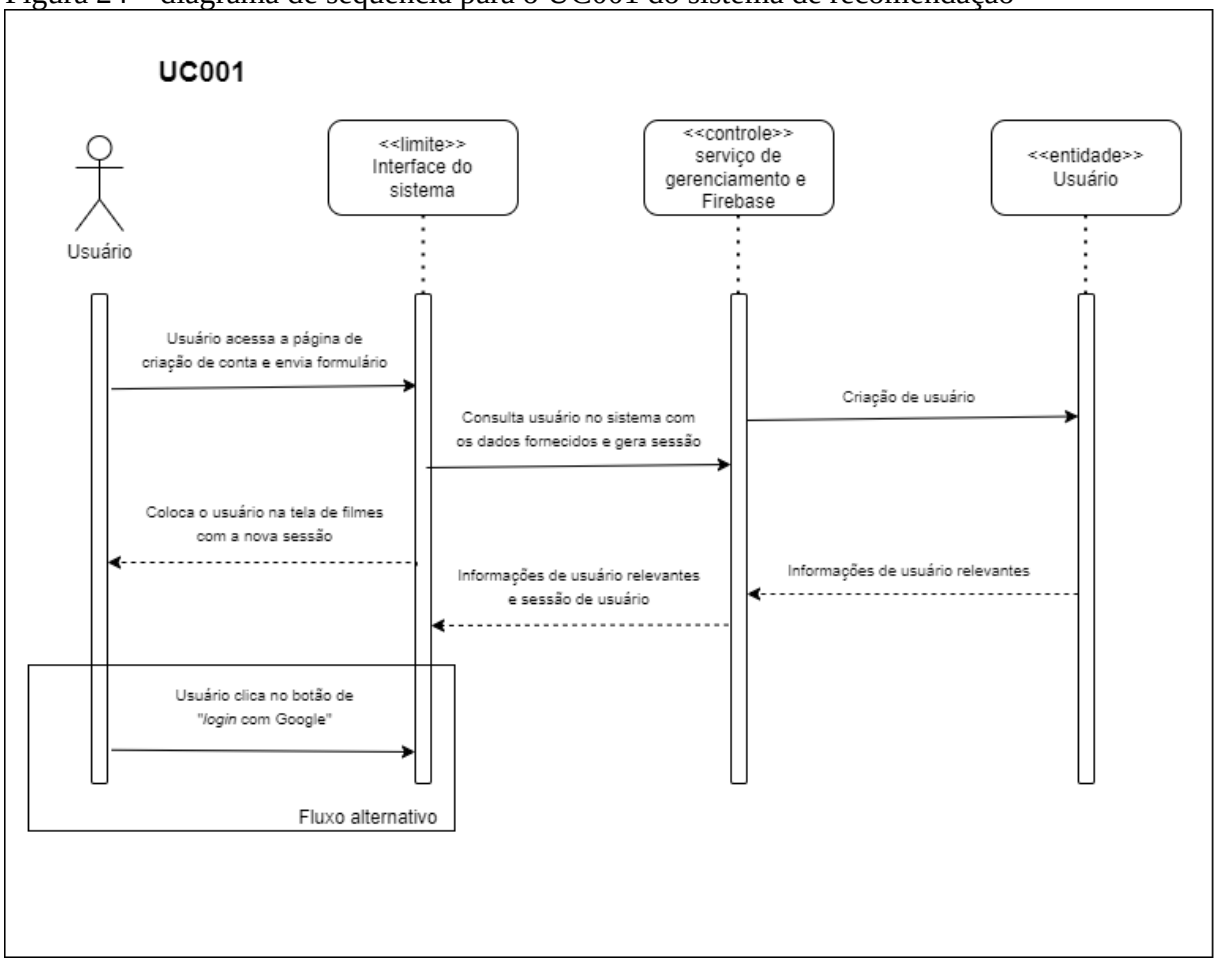
Aqui vale destacar que diferente de Sousa (2013) e Sbrocco e Macedo (2012, p. 179), Sommerville (2011) não disponibiliza o diagrama de sequência como uma ferramenta da metodologia ICONIX, mas sim como um método de descrever casos de uso num nível mais alto de detalhamento.

O diagrama de sequência tem como ponto forte as mensagens de retorno, e é mais utilizado que um diagrama de atividades (SBROCCO e MACEDO; 2012).

> O diagrama de sequência tem como objetivo mostrar a colaboração dinâmica (troca de mensagens) entre os vários objetos do *software*. No ICONIX, assim como na maioria dos processos que utilizam a UML como linguagem de modelagem, o comportamento de um caso de uso é detalhado por meio do diagrama de sequência. O principal objetivo dessa atividade é designar comportamento, proveniente das classes de controle, para as classes de entidade e interface. (SOUSA, 2013, p. 48)

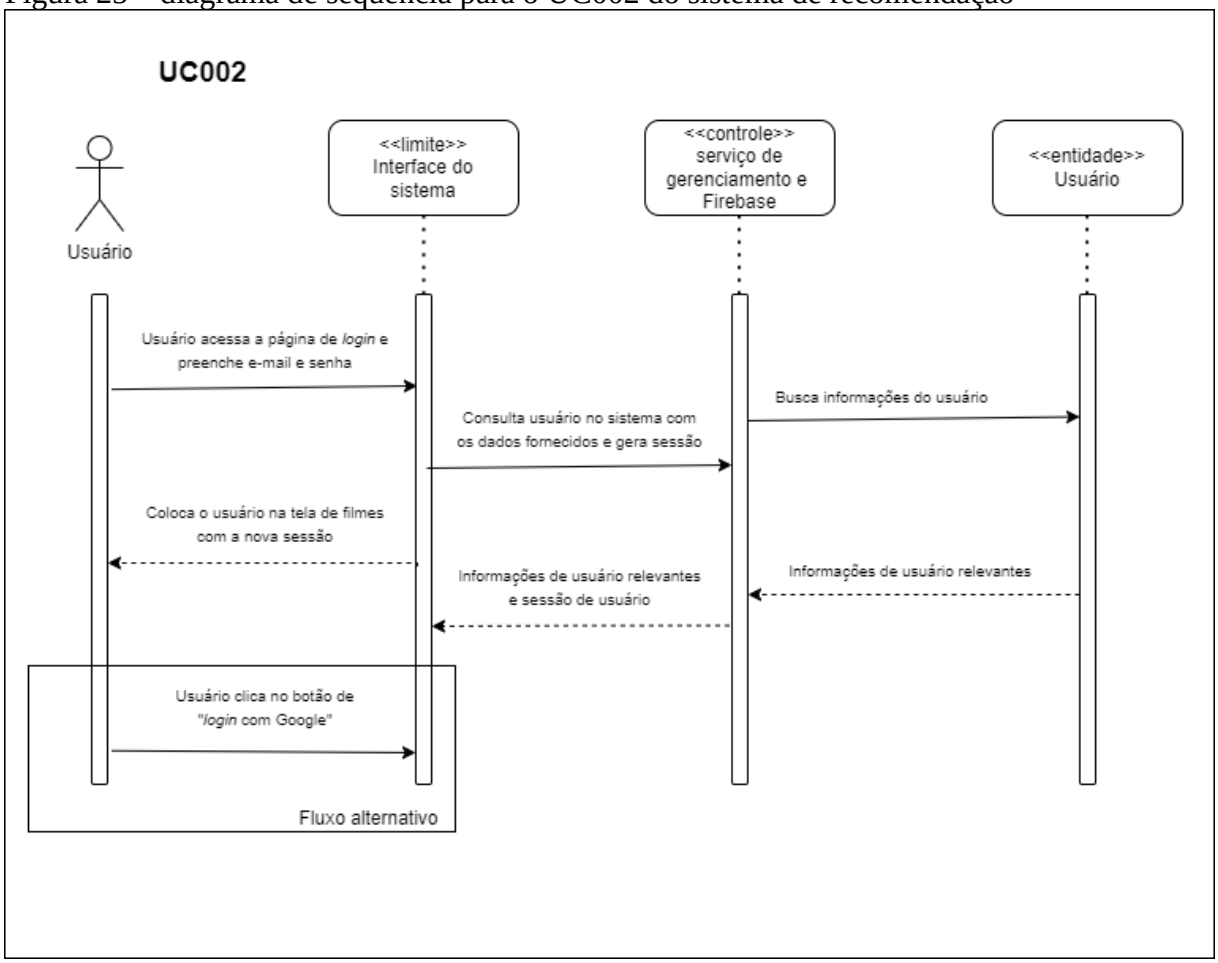
Dita as vantagens descobertas na literatura sobre o diagrama de sequência utilizado na metodologia ICONIX – mas não se limita a esta metodologia –, a Figura 24 demonstra o diagrama de sequência para o sistema de recomendação:

Figura 24 – diagrama de sequência para o UC001 do sistema de recomendação



Fonte: autoria própria (2023)

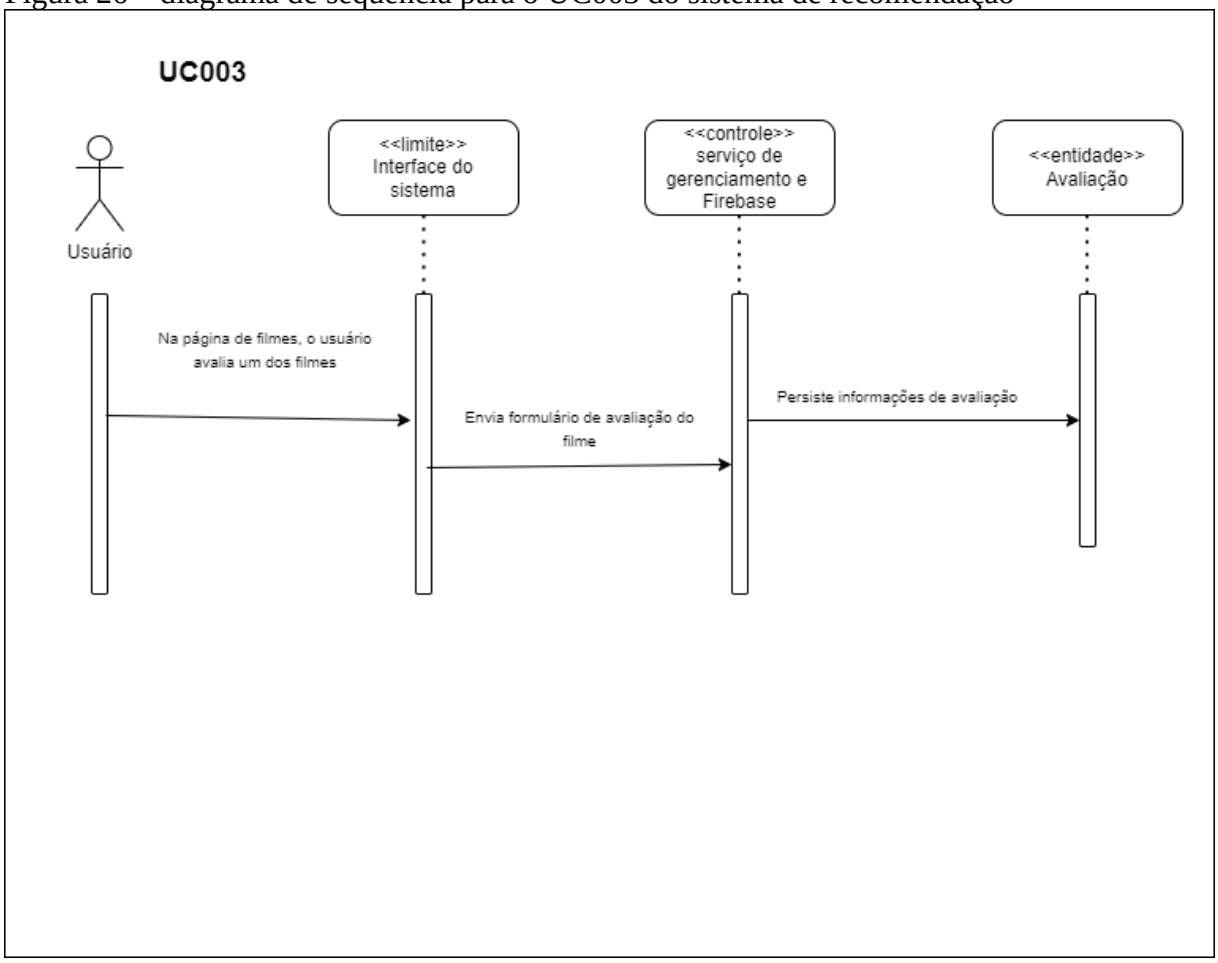
Figura 25 – diagrama de sequência para o UC002 do sistema de recomendação



Fonte: autoria própria (2023)

Os casos de uso UC001 e UC002 (Figuras 24 e 25, respectivamente) possuem suas similaridades, para o usuário por exemplo o *login* ou criação de conta utilizando Google fica transparente, e ao utilizar e-mail e senha há 1 passo extra, que é o preenchimento dos campos necessários tanto para o cadastramento da conta quanto para o *login*, onde o usuário utiliza o e-mail e a senha.

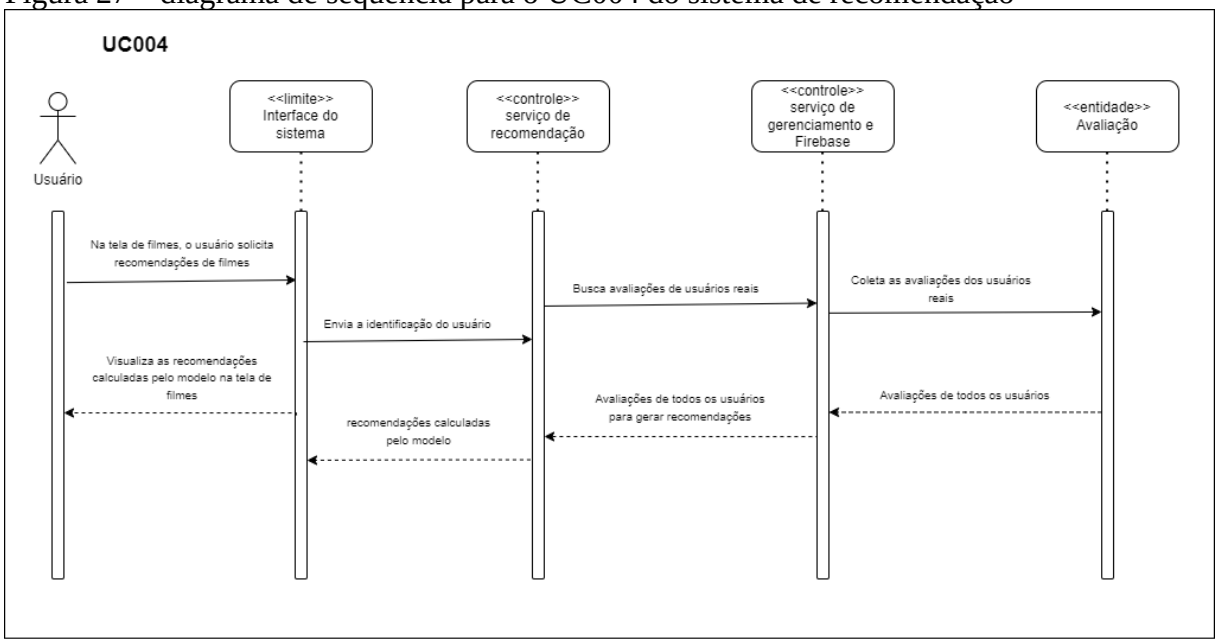
Figura 26 – diagrama de sequência para o UC003 do sistema de recomendação



Fonte: autoria própria (2023)

Simples e direto, a resposta ao usuário para o UC003 (Figura 26) foi a própria ação de avaliar o filme e a informação salva é utilizada tanto para recomendação quanto para as próximas visitas do usuário no site.

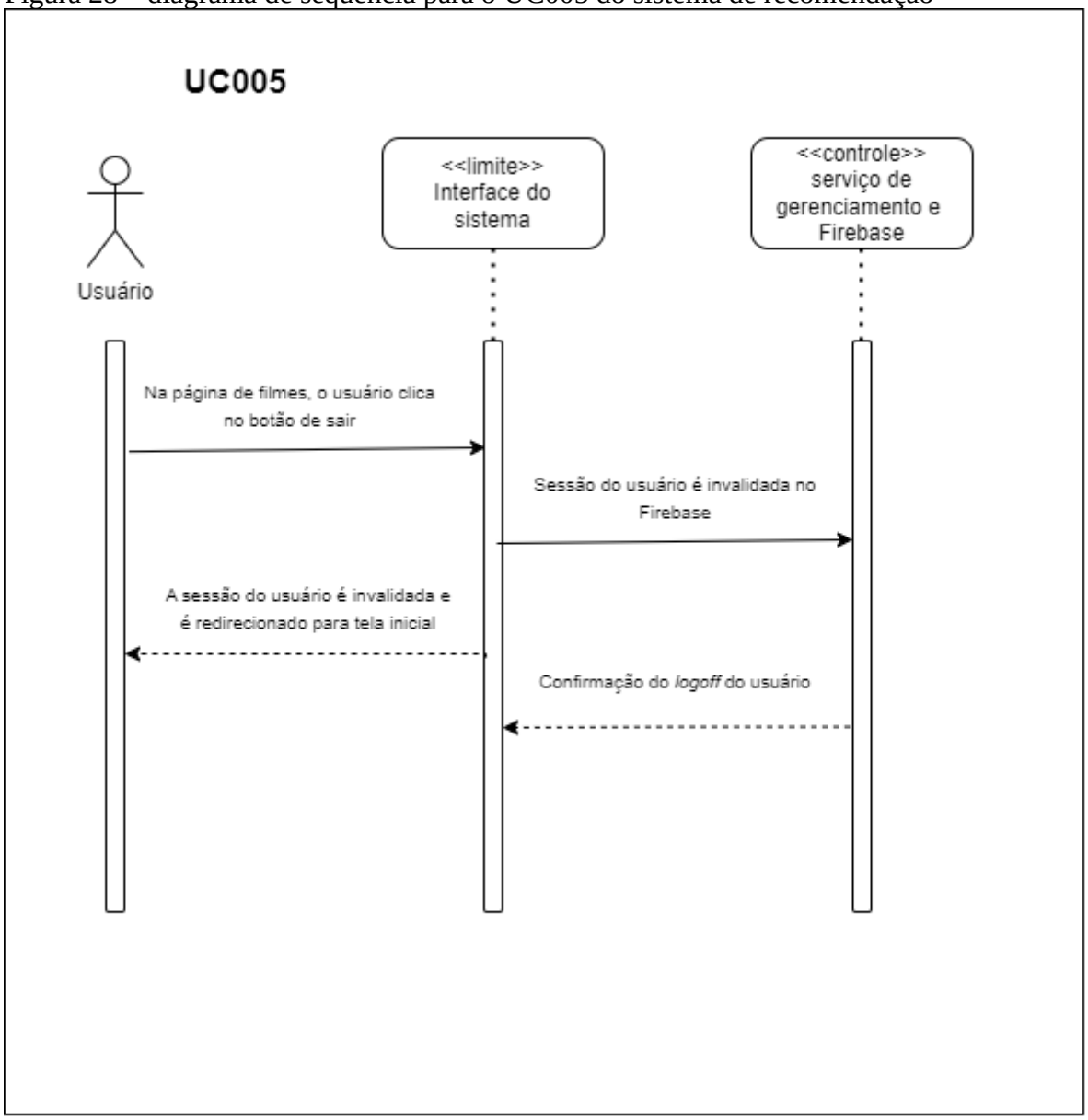
Figura 27 – diagrama de sequência para o UC004 do sistema de recomendação



Fonte: autoria própria (2023)

O caso de uso UC004 (Figura 27) é o mais complexo de todos. No capítulo 5 em que o desenvolvimento é abordado fica claro todo o fluxo entre os serviços.

Figura 28 – diagrama de sequência para o UC005 do sistema de recomendação



Fonte: autoria própria (2023)

O caso de uso UC005 (Figura 28) simplesmente efetua o *logoff* do usuário no sistema, invalidando sua sessão.

Toda a literatura do presente capítulo guiará o processo de desenvolvimento.

5 DESENVOLVIMENTO

O capítulo 5 é destinado a demonstrar a utilização da filtragem colaborativa baseada em Usuário – Usuário em um protótipo funcional. O desenvolvimento do protótipo inclui: o desenvolvimento de um serviço para o gerenciamento de cadastros de usuários e suas avaliações, o cliente para que seja possível que o usuário interaja com o sistema de recomendação, e o modelo que fará os cálculos de recomendação. Com os serviços do sistema em mãos, o protótipo também precisará de uma infraestrutura para ser hospedado e acessado num DNS público. O processo de desenvolvimento e a configuração infraestrutura são destrinchados nas sessões a seguir num formato em que fica claro o processo de desenvolvimento, com todos os erros e tentativas ao longo do caminho, e por fim, a avaliação do protótipo funcional com usuários reais.

É também importante que para não gerar dúvidas fique claro que a palavra "serviço" no contexto do presente capítulo se destina a um repositório de código, que possui **uma** responsabilidade e irá executar **independente** dos outros serviços, assim o sistema mantem o baixo acoplamento. Nesse protótipo temos 3 serviços, já mencionados anteriormente: o modelo de recomendação, o gerenciador de usuários e avaliações e o cliente, o último sendo uma aplicação web. A palavra "sistema", portanto, refere-se ao conjunto desses serviços trabalhando juntamente com serviços externos, que, como o serviço da Google, o Firebase, é responsável pela validação da autenticação do usuário no sistema; e também ao ambiente onde este sistema está hospedado, então "sistema" também engloba infraestrutura. A palavra "infraestrutura" é autoexplicativa, e não faz parte do desenvolvimento do *software* no quesito funcionalidades, mas tem a responsabilidade de manter o *software* disponível para o usuário, com a menor latência possível.

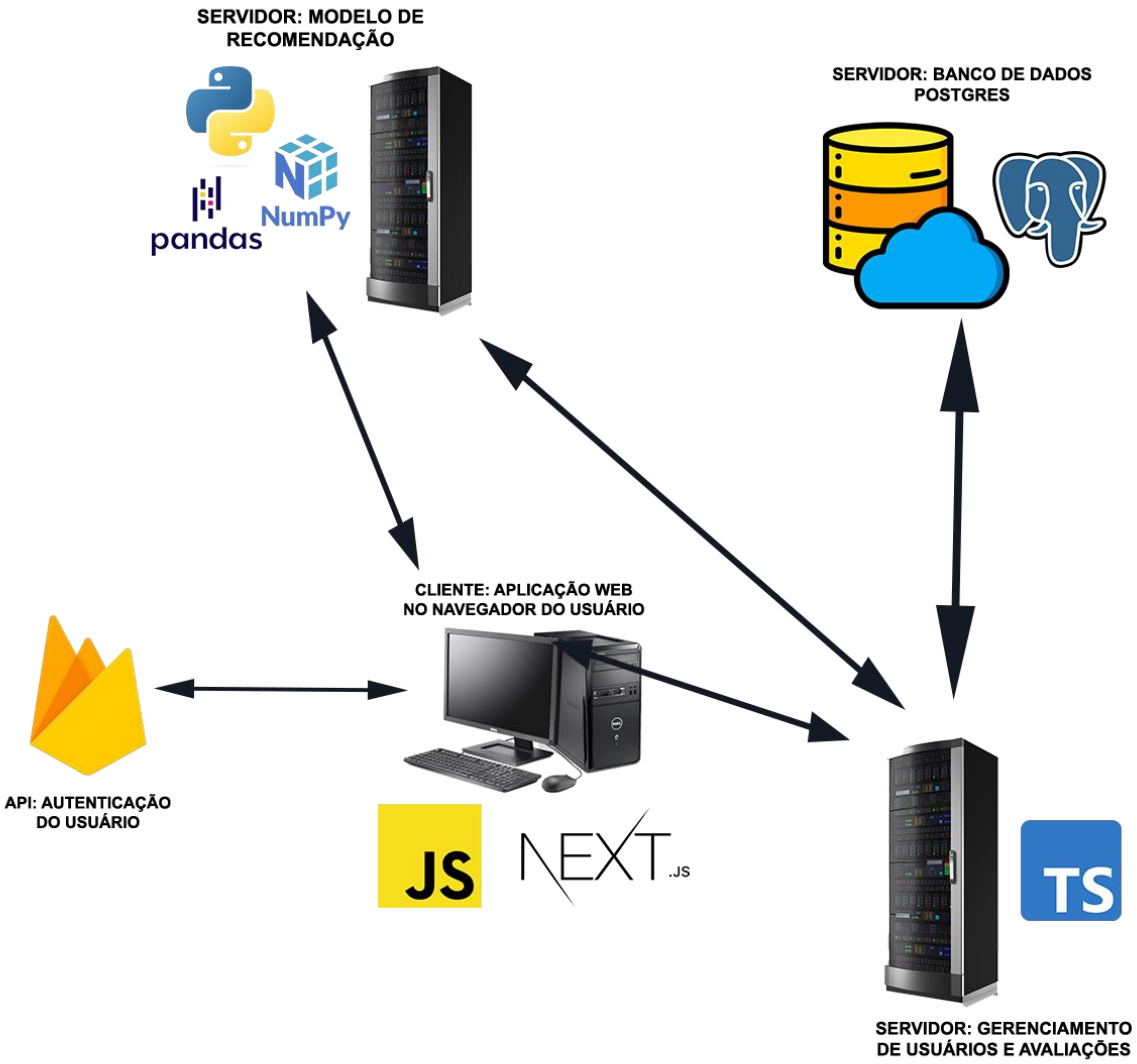
As figuras a seguir, Figura 29 e Figura 30, são respectivamente: as ferramentas utilizadas no desenvolvimento do protótipo (mesmo que estas não tenham sido incluídas no protótipo final), e a arquitetura do sistema desenvolvido.

Figura 29 – Ferramentas utilizadas

Fonte: autoria própria (2023)

As ferramentas na Figura 29 estão dispostas de forma a agrupa-las pela sua função, e ao ler a sessão 5.1 e 5.2 da monografia fica claro que, por exemplo, TypeScript foi utilizado juntamente com PostgreSQL; Firebase, JavaScript e NextJS foram utilizados em conjunto e assim por diante.

Figura 30 – Arquitetura do sistema



Fonte: autoria própria (2023)

5.1 PROCESSO DE DESENVOLVIMENTO DOS SERVIÇOS

Essa arquitetura modular, ou micro serviços, foi escolhida pela sua capacidade de dividir responsabilidades ao mesmo tempo em que remove “ponto único de falha”. O mesmo resultado pode ser obtido em arquiteturas monolíticas, mas visto que a presente monografia foi fortemente influenciada pela excelência de *software* que empresas como *Netflix* demonstram

(EVANS, 2016) e o comprometimento dos times de desenvolvimento por trás dos produtos de empresas como *Netflix*, a abordagem de micro serviços se faz adequada.

5.1.1 Gerenciador de usuários e avaliações

O serviço de gerenciamento de usuários e avaliações é um simples cadastro parcial de usuários e avaliações. A palavra parcial na descrição desse serviço é porque que nem todas as operações básicas de cadastro – essas sendo: criar, ler, atualizar e deletar – estão presentes para os usuários ou avaliações. As responsabilidades desse serviço são:

- Cadastrar um usuário;
- Encontrar um usuário pelo seu e-mail;
- Encontrar um usuário pelo seu ID;
- Cadastrar uma avaliação de filme nova para o usuário;
- Atualizar a avaliação que o usuário forneceu sobre o filme;
- Encontrar todas as avaliações de um usuário;
- Retornar todas as avaliações [para o modelo de recomendação];

5.1.1.1 Tecnologias e ferramentas

Existiram duas versões desse serviço: a primeira foi criada utilizando Go (ou Golang), que é uma linguagem de fácil leitura, uma excelente abordagem em relação a programação concorrente e performance sólida e persistia os cadastros de usuários e avaliações no banco de dados não relacional MongoDB, mas por motivos que discutiremos no histórico de desenvolvimento, essa versão foi substituída por um serviço que possuía as mesmas funcionalidades, mas foi escrito em TypeScript e persistia suas informações de usuários e avaliações no banco de dados relacional PostgreSQL. Todas essas mudanças se deram por conta de serviços ofertados pela AWS, e é melhor abordado na sessão de infraestrutura.

5.1.1.1.1 Go

Go é uma linguagem de programação de código aberto que, segundo a documentação (2023) do próprio Go, permite que os desenvolvedores sejam mais produtivos.

Ainda segundo a documentação do Go (2023), a linguagem é expressiva, concisa, limpa e eficiente. Seus mecanismos de concorrência tornam fácil a escrita de programas que fazem proveito de processadores multicores, ao mesmo tempo que seu sistema de tipagem permite a construção de um sistema flexível e modular.

Go foi escolhido como primeira opção por conta de sua performance e facilidade na hora de trabalhar, e foi descartado por não ser tão popular, mas essa decisão se baseia mais na infraestrutura e na dinâmica desejada para a abordagem deste protótipo.

5.1.1.1.2 MongoDB

MongoDB é um banco de dados de código aberto baseado em documentos com escalabilidade e flexibilidade (MONGODB, 2023). Segundo a documentação do MongoDB o banco de dados é simples e eficiente.

MongoDB foi escolhido pela sua simplicidade e facilidade de integração, mas assim como Go, foi descartado por falta de integração e facilidade nas ferramentas de *deploy* da AWS, outra decisão baseada na infraestrutura disponível somada com a dinâmica desejada para a abordagem deste protótipo.

5.1.1.1.3 TypeScript

Segundo a documentação: mais de 20 anos depois de sua criação, JavaScript é uma das linguagens mais utilizadas, sendo adaptada para múltiplas plataformas. TypeScript

adiciona, entre outras coisas, checagem e definição de tipos estáticos para JavaScript, e possui seu próprio compilador (TypeScript, 2023).

Node.js, é o *runtime* responsável por permitir que a linguagem seja executada como linguagem de servidor (NODEJS, 2023), e esse *runtime* é o motivo pela troca: ele é popular, isto é, é possível integrá-lo com facilidade em serviços de nuvem pública como AWS sem muitos esforços.

5.1.1.1.4 PostgreSQL

PostgreSQL é um banco de dados relacional orientado a objetos de código aberto com mais de 35 anos de desenvolvimento, possui excelente robustez e performance (POSTGRES, 2023).

Apesar da eficácia do sistema gerenciador de banco de dados, não foi a primeira escolha. A flexibilidade de um banco não relacional como MongoDB e a velocidade que permite o desenvolvedor a corrigir erros sem precisar alterar tabelas que ainda estão em fase de prototipação são as grandes vantagens de um banco de dados nesse formato, no entanto, a ferramenta a qual se pretende usar, AWS Lightsail, não fornece suporte nativo ao MongoDB, mas mais sobre isso na sessão de infraestrutura (5.2).

5.1.2 Cliente

O cliente (ou *frontend)* é a parte que o usuário pode interagir, com base nos protótipos da sessão 4.3.2 desta monografia, o cliente permite que o usuário:

- Crie uma conta utilizando seu e-mail;
- Crie uma conta utilizando sua conta Google;
- Efetue login com seu e-mail, caso já possua uma conta;
- Listar os filmes disponíveis para avaliação;

- Efetuar recomendações com o clique de um botão;

5.1.2.1 Tecnologias e ferramentas

O desenvolvimento do cliente foi simples e direto: utilizar um *framework* popular, desenvolver todas as telas e integrar todas as funcionalidades dos demais serviços, incluindo o serviço externo Firebase, responsável pela autenticação dentro do cliente.

5.1.2.1.1 Javascript

Javascript (ou JS) é uma linguagem de programação leve e interpretada (ou "compilado na hora"). Mais conhecida por ser a linguagem de script para páginas web, muitas aplicações utilizam o *runtime* Node.js para executar Javascript e suas variações em outros ambientes fora do navegador (DEVELOPER MOZILLA, 2023).

5.1.2.1.2 Next.js

Next.js é um *framework* que, a partir do React e suas contradições na parte de configuração, expandiu para se tornar uma ferramenta completa, fornecendo configuração própria com um compilador próprio, criado pela equipe por trás do Next.js, a equipe também disponibiliza uma plataforma de *deploy* própria para aplicativos web criados com Next.js chamada Vercel (NEXT.JS, 2023).

A escolha do *framework* foi baseada na sua dinâmica de lidar com roteamento, o que está se tornando comum entre outros *frameworks* web: Next.js faz o roteamento de toda aplicação baseando-se na estrutura dos diretórios e arquivos na raiz do projeto. Outras

funcionalidades que vale ressaltar, apesar de minimamente utilizadas no protótipo é o SSR (*Server Side Rendering*) que permite o desenvolvimento de uma aplicação web que carrega itens estáticos do servidor antes de renderizar a página, tornando a experiência do usuário melhor, e aprimorando o SEO, assunto que está fora do escopo do presente trabalho (NEXT.JS, 2023).

5.1.2.1.3 Firebase

Firebase é uma plataforma de desenvolvimento que ajuda a construir aplicativos e jogos mantida pela Google (FIREBASE, 2023).

Firebase é uma solução popular entre a comunidade de desenvolvedores para funcionalidades comuns que, dentro da plataforma são fáceis de usar e escalar, mas para implementar manualmente costumam dar mais trabalho, no caso do protótipo a única função utilizada foi a de autenticação de usuários.

5.1.3 Modelo de recomendação e coleta de dados

Responsável por calcular as recomendações de cada usuário. Existem algumas abordagens ao se criar um sistema de recomendação do zero, sem nenhum dado ou ponto de referência, no livro de Ricci, Rokach e Shapira: *Recommender Systems Handbook* (2015), uma delas é a recomendação baseada no conteúdo, tendo em vista um sistema zerado de informações do usuário, não se sabe o que nenhum usuário quer em nenhum momento, então a recomendação baseada em conteúdo faz sentido. No entanto, conforme já afirmado anteriormente, o intuito é a recomendação baseada na técnica de filtragem colaborativa baseada em Usuário – Usuário. Então é necessário encontrar dados iniciais para tentar mitigar ao máximo o problema de “arranque a frio” de sistemas de recomendação, e no caso do presente trabalho os dados inferidos inicialmente estão disponíveis em *datasets* de sites como Kaggle.

5.1.3.1 Tecnologias, ferramentas

Para resolver o problema da recomendação foi reinventada a roda alguma vezes, a estrutura dos dados precisou ser constantemente adaptada sempre que um novo modelo era posto para teste.

5.1.3.1.1 Python 3

Python é uma linguagem de programação interpretada, orientada a objeto e de alto nível com uma semântica dinâmica. Pelo fato de o Python ter estruturas de dados de alto nível, e com tipagem e semântica dinâmica, acaba sendo uma excelente opção para desenvolver aplicações de forma rápida (PYTHON, 2023).

Python foi escolhido por possuir uma variedade de bibliotecas relacionadas a aprendizado de máquina. Visto que Python foi a única linguagem utilizada, as ferramentas e tecnologias a seguir são bibliotecas disponíveis na linguagem que facilitaram muito o desenvolvimento do protótipo funcional.

5.1.3.1.2 Pandas

Pandas é uma escolha fácil para tratamento de dados, considerando que foi utilizado arquivos JSON e CSV e muitas iterações no processo de manipulação desses dados para melhor servir o intuito do protótipo. Pandas, segundo a sua documentação é: uma ferramenta de código aberto para manipulação e análise de dados rápida, poderosa, flexível e fácil de usar, construída em Python (PANDAS, 2023).

5.1.3.1.3 Numpy

Numpy é uma biblioteca do Python para computação científica, esta biblioteca fornece vetores multidimensionais, rotinas de ordenamento para rápidas operações com vetores, Numpy fornece rotinas matemáticas, lógicas, manipulação de formatos, ordenação, seleção, álgebra linear básica, estatística básica e muitas outras funcionalidades (NUMPY, 2023).

Uma pequena parcela do potencial do Numpy foi utilizada no protótipo para auxiliar nos cálculos durante a construção dos modelos de recomendação, mas muito importante para a construção dos modelos.

5.1.3.2 Pesquisa e tratamento dos dados

Os dados utilizados para o protótipo, a partir desse capítulo chamados de *datasets*, são uma união de duas fontes diferentes, os *datasets* tiveram a fase da coleta de dados, preparo, processo de pré-processamento, este que ocorre toda vez que o modelo treina, e enfim os dados estão prontos para serem utilizados pelo modelo. No presente trabalho o termo *dataset* se refere a um conjunto de dados que não necessariamente estão limpos ou corretos, mas se têm em volume e os dados estão agregados de forma a serem usados como um *Dataframe* do Pandas, por exemplo. Um conjunto de dados pode ser preparado para se tornar um *dataset*, este processo foi realizado num dos *datasets*.

5.1.3.2.1 Sobre os datasets

Os dados iniciais do modelo de recomendação consistem de quase 20 mil filmes da base de dados do TMDB, uma base de recursos de filmes *open source* e que ajudou imensamente no desenvolvimento do protótipo. A necessidade que estes dados atendem é o recurso que ela disponibiliza para a listagem dos filmes, a foto do filme, como o protótipo exige

uma imagem do filme, demonstrado na Figura 12, faz sentido coletarmos filmes de uma fonte de dados que nos fornece a imagem dos filmes disponíveis para avaliação e recomendação.

O segundo *dataset* utilizado no protótipo é o "*MovieLens 20M Dataset*", comumente utilizado para o propósito de treinamento de modelos para recomendação de filmes. Como o nome indica, o *dataset* possui pouco mais de 20 milhões de linhas de avaliações de usuários, e um *dataset* de filmes com algumas poucas informações sobre os filmes avaliados, como nome, por exemplo. Para referência, o *dataset* com filmes coletados e agrupados do site TMDB será chamado de *dataset* TMDB, e ambos *datasets* de filme e avaliações desses filmes que o *dataset* fornecido pelo *MovieLens*, disponível no Kaggle serão chamados de *dataset MovieLens*.

5.1.3.2.2 Coleta e preparo dos datasets

A fase de coleta se aplica somente para o *dataset* TMDB, pois os *datasets MovieLens* disponibilizados no Kaggle já estão agrupados da forma certa. O TMDB possui uma API pública que permite a coleta de filmes, e um dos *endpoints* dessa API é de filmes populares, este tem uma paginação de 20 itens por vez e um máximo de 1000 páginas. Um script simples em python efetuando essas requests em loop, agregando os filmes buscados em cada página e salvando-os em um arquivo JSON ao final foi suficiente para finalizar a coleta de dados dos filmes com os dados necessários para a listagem na aplicação cliente do sistema.

A parte de preparo *datasets* foi necessária porque (a) dos quase 20 mil filmes coletados para a formação do *dataset* TMDB, nem todos estes filmes eram presentes no *dataset MovieLens*; e (b) nem todas as avaliações do *dataset MovieLens* eram de filmes disponíveis no *dataset* TMDB. Considerando tais informações, é possível visualizar um problema de intersecção: TMDB ∩ *MovieLens*. Para realizar essa intersecção é necessário encontrar um campo em comum entre essas duas fontes de dados, o único campo que essas duas fontes de dados compartilham em comum é o nome e ano do filme, e estes precisam ser formatados, a intersecção dos *datasets* se deu com os seguintes passos:

- Utilizando o exemplo do filme Toy Story (1995): um filme disponibilizado no *dataset* TMDB possui o nome por extenso e somente o nome por extenso,

então Toy Story estava descrito com o campo "*title*": "Toy Story", e um campo de data de lançamento no mesmo JSON; no *dataset MovieLens* o nome do filme era descrito como "Toy Story (1995)", então o tratamento desse campo tipo *string* foi necessário para remover o ano do nome do filme para outra coluna no *dataset* efetuar a união dos dois *datasets* de forma correta, utilizando nome e ano de lançamento do filme como referência;

- Após isso, a lógica é simples: inserir o ID presente nos filmes do *dataset MovieLens* dentro do objeto JSON do *dataset* TMDB, os filmes cujos nomes não foram encontrados em ambos *datasets* foram descartados. Foram encontrados quase 5 mil filmes na intersecção dos dois *datasets* intersecção.

Após a intersecção dos filmes, foi necessário remover as avaliações do restante dos filmes que não foram encontrados na intersecção do *dataset* de avaliações do *MovieLens*. O processo é simples: criar uma lista de Ids dos quase 5 mil filmes e verificar quais não estão nessa lista, resolvendo o problema de diferença e garantindo que todas as avaliações de filmes disponíveis no *dataset MovieLens* sejam de filmes que o usuário poderá ter como recomendação.

5.1.3.2.3 Pré-processamento de dados

Diferente do processo de preparo, o pré-processamento de dados ocorrerá toda vez que o modelo de recomendação for treinado, isso porque sempre que o modelo inicia o treino o "*dataset* final" de avaliações é atualizado – este "*dataset* final" sendo a soma das avaliações dos usuários que estão no banco de dados, que no caso do protótipo é o PostgreSQL, e o *dataset* de avaliações do *MovieLens* que preparamos na fase de preparo dos dados. Os passos para o pré-processamento são:

- A diminuição do *dataset* considerando a quantidade de avaliações realizadas por usuários do sistema de recomendação ao invés de dados do *dataset* inicial – esse número costuma flutuar conforme a memória RAM disponível na máquina rodando o algoritmo, a máquina de desenvolvimento local possui 16GB de RAM e houve casos de erros de segmentação, tanto no pré-

processamento quanto no treinamento de alguns modelos de recomendação, por tanto o tamanho dos dados a serem utilizados é dinâmico;

- Para finalizar o tratamento dos dados, os dados do *dataset* de avaliações são convertidos para melhor atenderem as necessidades do modelo final, tal modelo precisará responder, com velocidade, perguntas como: (a) dado usuário U, quais filmes M este usuário avaliou? (b) dado filme M, quais usuários U avaliaram este filme? E (c) dado usuário U e filme M, qual a nota está atribuída na avaliação? Ao final, existem 3 dicionários. Dicionário em Python é uma estrutura de dados com base em chave e valor, e no caso da pergunta c, por exemplo, pode-se ter uma chave que combina dois valores que corresponde ao mesmo valor, efetuando uma busca mais rápida durante o processo de treinamento e recomendação.

5.1.3.3 Modelos de recomendação

Inicialmente, os modelos de recomendação estavam organizados no trabalho junto com as ferramentas e tecnologias. O motivo dessa primeira abordagem em relação a organização textual é simples: muitas vezes os modelos de recomendação estão fortemente baseados na tecnologia por trás, por exemplo: a biblioteca TensorFlow possibilita a criação de modelos e possui modelos prontos para fazer recomendação, então o modelo de recomendação criado a partir do TensorFlow estaria fortemente associado à biblioteca em si, por isso faz sentido apresentar o modelo e a lógica por trás deste a partir da ferramenta. Mas por outro lado, essas ferramentas: Surprise (SURPRISE, 2023) e TensorFlow (TENSORFLOW, 2023) cumprem propósitos diferentes e específicos, por este motivo, a sessão atual: “Modelos de recomendação” trata especificamente do treinamento desses modelos e experiência de desenvolvimento obtida com estes.

5.1.3.3.1 Como medir a performance de um sistema de recomendação?

Como saber se um sistema de recomendação está recomendando de maneira correta? Quais são as métricas? Como identificar se o usuário está satisfeito com as recomendações? Há bastante literatura sobre a metrificação da satisfação do usuário final com as recomendações, e é isso que esta sessão se dedica a explorar.

Em sistemas de recomendação e na filtragem colaborativa, as medidas de similaridade[3] têm sido o componente operacional principal para avaliar a performance que a filtragem colaborativa está desempenhando. Segundo os autores – que questionam essa estratégia e fazem um estudo de caso para propor uma nova – essas medidas de similaridade são utilizadas aos montes para evitar o problema do espaçamento de dados na matriz da filtragem colaborativa (problema esse mais conhecido como "arranque a frio"), mas mesmo com essas várias métricas, estas por si só também sofrem do problema do arranque a frio, e ainda costumam possuir um *design* complexo [para a captura e utilização das informações] (AMER; ABDALLA; NGUYEN, 2021). O artigo de Amer, Abdalla e Nguyen (2021) propõe que num sistema de recomendação deve-se utilizar medidas de similaridade altamente eficazes para determinar a semelhança entre os vizinhos, assim, melhora-se a performance.

Mas como a performance do sistema de recomendação é medida? Uma resposta objetiva a pergunta se dá no trabalho de Ricci, Rokach e Shapira (2015), que afirma que "a forma mais comumente aceita de avaliação das medidas para um sistema de recomendação é calculando o *"Mean Average Error*" (MAE) ou "*Root Mean Squared Error*" (RMSE) da classificação prevista e a real classificação fornecida. Essas métricas computam a assertividade sem nenhuma suposição ou sem levar em consideração o propósito do sistema de recomendação. No entanto, como afirmam os autores, há muito mais a se levar em consideração do que a assertividade do modelo de recomendação para saber se um item deverá ou não ser recomendado.

Considerando o fato de que "existe um motivo para um item ser recomendado," e "levando em consideração o propósito de um sistema de recomendação", o time da *Netflix* (TINGLEY et al., 2021) publicou em seu blog uma série de artigos sobre a tomada de decisão

[3] Como fica claro no livro de Ricci, Rokach e Shapira: *Recommender Systems Handbook* (2015) as medidas de similaridade são, como o nome indica, as medidas utilizadas para identificar quais usuários (ou itens) são similares a quais usuários (ou itens).

e como é o processo de teste AB, processo esse já abordado na sessão 4.2 sobre prototipação, e de acordo com os dois artigos o processo é como descrito anteriormente nessa monografia: dividir os usuários em amostra A e amostra B, o grupo A o grupo de controle, o grupo B o grupo de teste e comparar as métricas decididas no processo de testagem. Essas métricas são de fato o que decide o quão eficaz, ou no caso de um teste AB o quão mais eficaz o modelo de recomendação é. No artigo de Tingley et al. (2021) sobre a tomada de decisão da *Netflix* é claro que as decisões são tomadas por aqueles que detêm o maior poder de voz: os usuários. E o método de teste AB é o método de escolha para a coleta desses *feedbacks* implícitos, por exemplo: tempo de engajamento do usuário com item recomendado; quantidades de cliques do usuário; se o usuário assistiu o item recomendado; se o usuário voltou para a plataforma para assistir o item recomendado; etc.

Num outro artigo da *Netflix* (AMATRIAIN; BASILICO, 2012) chamado "*Netflix Recommendations: Beyond the 5 stars (Part 2)*" os autores afirmam que se você procura uma função de ranqueamento que otimiza consumo, uma solução óbvia é ordenar os itens por popularidade. O motivo é simples: no geral, um usuário vai estar mais propenso a assistir o que os outros usuários mais estão assistindo. No entanto, a lógica de ordenar os itens por popularidade é o contrário de "personalização": a ordem dos itens ofertados será a mesma para todos os usuários. Portanto, o objetivo é encontrar uma função de ranqueamento de itens – no caso da Netflix: filmes e séries de TV – que seja melhor que a popularidade dos itens, para assim satisfazer os usuários do sistema com os gostos mais variados.

Para o caso do protótipo funcional do presente trabalho o objetivo é: utilizando uma base de dados terceira, treinar um modelo de recomendação para usuários reais e, através de um questionário (*feedback* explicito) coletar a opinião destes usuários. A parte curiosa do experimento é utilizar uma base de dados externa, ou em outras palavras uma base de dados que não reflete os usuários reais do experimento, para se observar o quanto se consegue mitigar (ou quem sabe até piorar) na questão do "arranque a frio", que todo sistema de recomendação – e em especial sistemas de recomendação baseados em filtragem colaborativa – sofrem.

5.1.3.3.2 Modelo 1: Surprise

Surprise é uma biblioteca de código aberto que utiliza “por baixo dos panos” a biblioteca Scikit Toolkits e disponibiliza modelos para sistemas de recomendação (SURPRISE, 2023), em outras palavras, Surprise é a biblioteca Scikit Toolkits adaptada para sistemas de recomendação.

O motivo dessa biblioteca ser a primeira opção é a facilidade, a documentação clara e objetiva e facilmente ser possível adaptar o modelo de recomendação com mais de um algoritmo de recomendação, exemplo: com pequenas adaptações em parâmetros é possível utilizar o modelo de recomendação com kNN, já observado na monografia, na sessão 2.2.5.1.1; outro algoritmo que foi adaptado junto a biblioteca Surprise é o SVD, observado na sessão 2.2.5.1.3 desta monografia. Ambos tiveram resultados relativamente satisfatórios, mas a um custo computacional muito grande, e por isso acabou sendo deixado de lado.

O modelo criado utilizando Surprise sofria de um problema comum chamado “*segmentation fault*”, que basicamente encerra a execução do programa por ter utilizado toda memória RAM disponível. No caso do modelo construído baseado na biblioteca Surprise o arquivo CSV utilizado para executar o treino tinha pouco menos de 600 mil de linhas, e 16GB de memória RAM não eram suficientes – com certeza algumas configurações poderiam ser feitas para mitigar a execução, mas depois de um tempo excessivo de tentativas com configurações diferentes, a abordagem acabou ficando de lado.

5.1.3.3.3 Modelo 2: TensorFlow

O modelo de recomendação utilizando TensorFlow e sua extensa biblioteca de recomendação foi rápido. Diferente da biblioteca Surprise, TensorFlow é uma biblioteca extensa e com muito potencial, um bom exemplo do uso da tecnologia em ação é o de Lin e Chi (2019): um sistema de recomendação de filmes que mescla filtragem colaborativa com redes neurais.

Mas considerando a natureza da filtragem colaborativa baseada em usuário, eliminar a complexidade dos ajustes necessários para a biblioteca funcionar de forma otimizada

e adicionar a complexidade de escrever um modelo do zero e especializado no algoritmo proposto se fez mais útil e didático – e otimizado, tanto de um ponto de vista de performance computacional quanto do ponto de vista exploratório. Mas novamente, com certeza a falta de *expertise* na ferramenta fez com que esta não fosse a melhor opção no desenvolvimento do protótipo.

5.1.3.3.4 Modelo 3

Ao longo do trabalho de Ricci, Rokach e Shapira (2015), os autores fornecem muitas abordagens para a construção de um modelo simples, tanto de regressão quanto de classificação para a filtragem colaborativa baseada em usuário, no exemplo de Herlocker et al. (1999) os autores levantam a questão de que a significância das notas que um usuário atribui é diferente de usuário para usuário, isto é, considerando um usuário propenso a dar notas altas, este fornece muitas notas altas e poucas notas baixas; já um usuário que costuma dar notas mais baixas e dificilmente fornece avaliações acima de 4 [numa escala de 0.5 à 5], por exemplo, precisam ter suas "tendências" ajustadas para que a recomendação seja feita de forma assertiva.

O modelo de recomendação utilizado no protótipo não é baseado em um modelo de bibliotecas como os modelos disponibilizados pela biblioteca TensorFlow, mas sim seguindo o mesmo princípio abordado num curso de sistemas de recomendação disponível na Udemy (PROGRAMMER, 2018).

Para explicar o modelo criado, se faz necessário entender qual o modelo abordado no curso em questão: segundo o instrutor do curso (PROGRAMMER, 2018), um algoritmo básico para calcular a média das notas dos itens é formalizado da seguinte forma (Equação 13):

$$s(j) = \frac{\sum_{i \in \Omega_j} r_{ij}}{|\Omega_j|} \tag{13}$$

Sendo *s(j)* a soma de todas as notas fornecidas para o item *j*, dividido pela quantidade de notas fornecidas para o item *j*. o símbolo ômega representado por Ω sendo o conjunto de todos os usuários que avaliaram o item *j*, e por fim, r_{ij} é a nota que o usuário *i*

forneceu ao item *j*. A limitação da abordagem demonstrada na equação acima é a falta de personalização.

Outra limitação é o não ajuste das tendências de cada usuário, como apontado no exemplo de Herlocker et al. (1999), para contornar essa limitação se faz necessário calcular essa tendência, mais formalmente conhecida como desvio (Equação 14):

$$dev(i,j) = r(i,j) - \bar{r}_i \tag{14}$$

Sendo $r(i,j)$ a avaliação que o usuário *i* forneceu ao item *j*, o símbolo $\bar{r}_i$ expressa o desvio padrão para o usuário i, e $dev(i,j)$ é o desvio da avaliação do usuário para o item avaliado, o resultado simboliza o quanto o usuário *i* gostou dos itens *j* avaliados, um em comparação a outro. Observando a equação é possível deduzir que caso o usuário seja alguém que em média fornece avaliações em torno de 2,5 será mais fácil de acertar as recomendações: tudo avaliado abaixo de 2,5 de 5 não é recomendável, tudo avaliado acima de 2,5 é recomendável e uma avaliação próxima de 4,5 ou 5 com certeza precisa ser recomendado. Mas o contrário também é válido, se o usuário avalia a maioria dos filmes com 5, então a recomendação será mista e incerta, este problema é conhecido como problema da ovelha cinza, abordado na sessão 2.2.4.5.

$$d\hat{e}v(i,j) = \frac{1}{|\Omega_j|} \sum_{i' \in \Omega_j} r(i',j) - \bar{r}_{i'} \tag{15}$$

A Equação 15 é a formalização da ideia principal do modelo: calcular "o quão *mais* – ou *menos* – o usuário *i* irá gostar da recomendação", sendo i' a iteração com os outros usuários. O desvio $d\hat{e}v(i,j)$ é a média de outros desvios, desvios estes vindo dos outros usuários que avaliaram o item *j*, dividido pela quantidade de pessoas que avaliaram o mesmo item. Sendo assim, uma forma simples de expressar as equações observadas até então se dá na Equação 16:

$$s(i,j) = \bar{r}_{i'} + d\hat{e}v(i,j) \tag{16}$$

Considerando que o objetivo é uma recomendação gerada a partir da relação entre usuários, ou seja, as avaliações dos outros usuários i' terão peso nas recomendações que o

usuário i receberá, o peso de cada usuário, um em relação a outro, deverá ser considerado na equação, para calcular o peso *w* será utilizado a equação de *Pearson*, abordado na sessão 2.2.2.1.1 da presente monografia, e a equação completa para o cálculo das recomendações ao usuário com os pesos fica (Equação 17):

$$s(i,j) = \bar{r}_{i'} + \frac{\sum_{i' \in \Omega_j} w_{ii'} \{r_{i'\,j} - \bar{r}_{i'}\}}{\sum_{i' \in \Omega_j} |w_{ii'}|} \tag{17}$$

Sendo $w_{ii'}$ o peso de cada usuário i' em relação ao usuário i, esse peso pode ser negativo, por isso o valor é tido como absoluto.

No momento de efetuar os cálculos é preciso ter algumas constantes, como o mínimo de filmes em comum que o usuário i e i' possuem em comum, no caso deste algoritmo essa constante é 5, ou seja, com menos de 5 filmes em comum os cálculos não chegam a ser executados, pois a relação seria fraquíssima e a probabilidade resultados contribuírem negativamente para a recomendação é alta. Outra coisa a se considerar é que para o algoritmo não tomar muito tempo no momento do treinamento é não calcular a similaridade entre todos os usuários de todos os filmes e suas relações entre si. Ao invés disso, uma abordagem mais dinâmica é, com os vizinhos mais próximos já encontrados para dado usuário, considerar somente tais vizinhos para calcular as recomendações, o número de vizinhos é guardado na variável *K*. Esta abordagem é conhecida como *kNN*, já abordada na monografia, a constante K costuma variar entre 25 e 50, e no caso do protótipo é utilizado 25 vizinhos para cada usuário.

Com o modelo criado pelo autor do curso "*Recommender Systems and Deep Learning in Python*" (PROGRAMMER, 2018) formalmente descrito, vale dizer que a implementação precisou de adaptações por conta dos dados e da dinâmica de trabalhar não só com um *dataset* fixo, mas com usuários novos sendo criados e anexados ao *dataset* de avaliações, mas toda a teoria se manteve, e as mudanças foram realizadas no tratamento e na forma de acessar os dados em questão.

5.1.4 Outras ferramentas e tecnologias

Além de todas as ferramentas utilizadas para desenvolver os serviços, algumas ferramentas são comuns entre si, pois são partes fundamentais no processo de desenvolvimento de *software* que se tem atualmente, estas ferramentas sendo: um repositório para versionamento de código e uma plataforma para a hospedagem desse código que foi desenvolvido, essas ferramentas são, respectivamente: Git e GitHub; além disso, é necessário um ambiente de desenvolvimento (ou um editor de texto, se preferir), comumente chamado de IDE, que significa "ambiente de desenvolvimento integrado" (em inglês: *Integrated Development Environment*). A IDE utilizada para toda edição dos textos foi o Visual Studio Code. Essas ferramentas são introduzidas e explicadas a seguir.

5.1.4.1 Git e GitHub

Ambas as ferramentas estão fortemente conectadas, então faz sentido introduzi-las juntas, pois a segunda não existiria sem a primeira. Git é um sistema de controle de versão grátis e de código aberto criado por Linus Torvalds em 2005, e desde 2005 mantido principalmente por Junio Hamano (GIT, 2023).

GitHub é uma plataforma de hospedagem de código gratuita que usa o Git como base para o controle de versão dos códigos hospedados na plataforma (GITHUB, 2023).

A motivação para usar tais ferramentas é sua praticidade; e segundo, mas não menos importante, sua popularidade.

5.1.4.2 Visual Studio Code

Visual Studio Code combina a simplicidade de um editor de texto comum com as poderosas ferramentas de desenvolvimento, como Intelli Sense (segundo a própria

documentação do Visual Studio Code: Intelli Sense é quando a ferramenta de texto preenche código de forma inteligente e acelera o processo de codificação, facilitando o processo de desenvolvimento de *software*) (VISUAL STUDIO CODE, 2023).

Itens como a IDE que se utiliza para o desenvolvimento costumam ser escolhidas a partir de dois critérios: o gosto pessoal do desenvolvedor e a necessidade da linguagem ou projeto desenvolvido. Por exemplo: há desenvolvedores que utilizam os produtos da JetBrains, que fornece para muitas linguagens uma IDE pesada e cheia de botões e funcionalidades – que quando utilizadas da forma correta podem economizar muitas horas de desenvolvimento; outros preferem a simplicidade de uma IDE como VIM, mas as comparações entre esses produtos estão além do escopo da presente monografia. O importante é deixar claro que o gosto pessoal do desenvolvedor influencia fortemente tal decisão; outro fator é o projeto: projetos escritos em Java ou C#, por exemplo, costumam ser desenvolvidos em um ambiente de desenvolvimento mais robusto, essas linguagens foram desenvolvidas para projetos de larga escala, e costumam ser mais produtivas quando utilizadas juntas com um ambiente de desenvolvimento – incluindo uma IDE, mais robusta.

5.1.4.3 Ubuntu 22.04.2 LTS

Ubuntu é um sistema operacional gratuito e de código aberto baseado no Kernel Linux e criado em 2004 (UBUNTU, 2023).

Ubuntu é uma ótima escolha para sistema operacional de forma geral: gratuito, fácil de usar e com todos os benefícios de um ambiente de Kernel Linux, e sua popularidade faz com que seja fácil achar as respostas quando necessário, além de consumir muito menos memória RAM que um sistema como Windows.

Vale uma observação sobre o sistema operacional Windows 10: este possui uma ferramenta chamada WSL – em especial WSL2 – que permite o uso de uma máquina virtual Ubuntu num *host* Windows, mas a experiência pode não ser das melhores – pelo menos até a data do desenvolvimento do presente trabalho –, como fica evidente no repositório do WSL no GitHub (WSL, 2023).

5.2 PROCESSO DE CONFIGURAÇÃO DA INFRAESTRUTURA

O processo de configuração da infraestrutura, comumente conhecido como *deploy*, foi efetuado inteiramente dentro da plataforma de nuvem pública AWS, mais especificamente no serviço *Lightsail*. Outras alternativas foram cogitadas como a plataforma *Vercel* (NEXT.JS, 2023) para *deploy*, mas não pareceu tão dinâmica e customizável quanto configurar uma máquina virtual, como é o caso do *Amazon Lightsail*, que utiliza as imagens da *Bitnami* nas máquinas virtuais que hospedam as aplicações. Outra observação interessante é o fato de não se ter utilizado Docker ou nenhum tipo de ambiente baseado em container no desenvolvimento do projeto, essa ausência não é acidental: a integração entre as máquinas virtuais e outros serviços dentro do ambiente AWS, a geração de certificados SSL para domínios e subdomínios, configuração de CORS e outras dores que o ambiente de produção – em especial quando o ambiente envolve máquinas virtuais e não containers – foi utilizado pelo autor como ponto forte no quesito exploratório, dada a falta de conhecimento prático do autor sobre *deploy* do zero com máquinas virtuais num DNS público.

5.2.1 AWS: Amazon Lightsail

AWS, ou Amazon Web Services é o serviço de nuvem publica mais utilizado no mundo (AWS, 2023). A AWS oferece serviços para atender as necessidades mais variadas e o valor inicial oferecido gratuitamente pela plataforma é bem generoso e permite os usuários a explorar muitas possibilidades, uma dessas sendo o Amazon Lightsail.

O Amazon Lightsail, como diz na própria documentação do serviço: é um serviço que permite aos usuários o *deploy* das aplicações de forma rápida e com baixo custo – apesar de ser mais caro que serviços como EC2 (EC2, 2023) para a alocação de máquinas virtuais. O mesmo vale também para outros serviços paralelos que exigem mais configuração dentro da própria nuvem AWS, o valor pago por recurso computacional é maior dentro do Amazon Lightsail, mas a configuração poupa muito trabalho, ou seja, o Amazon Lightsail é ideal para produtos no seu estágio inicial: fácil de lançar e manter, e o desenvolvedor pode migrar facilmente para outros serviços quando houver necessidade, ou encerrar os recursos caso o

produto não continue em desenvolvimento, facilitando a fase de configuração. Amazon Lightsail não é recomendado para produtos escalarem (LIGHTSAIL, 2023).

Para mais comparações entre serviços, em especial serviços computacionais que têm seus valores altamente variáveis, a AWS proporciona uma excelente calculadora que permite a visualização desses gastos (CALCULATOR, 2023).

O propósito do serviço Amazon Lightsail é claro: uma plataforma para efetuar *deploy* de forma simplificada, mas sem sacrificar a customização que cada sistema necessita, então foi capaz configurar domínio, SSL, CORS, comunicação entre serviços – garantindo que as outras portas estejam fechadas para evitar falhas obvias de segurança, e outras configurações que a maioria dos serviços web necessitam.

5.2.2 Preparo dos serviços para o ambiente de produção

O preparo para o *deploy* seguiu os seguintes passos:

- Configurar as máquinas virtuais que foram alocadas no serviço Amazon Lightsail;
- Configurar o banco de dados PostgreSQL no Amazon Lightsail;
- Utilizando a imagem da máquina virtual fornida pela Bitnami, configurar o SSL para disponibilizar o cliente e as APIs do serviço de recomendação e o gerenciador de usuários e avaliações;
- Por fim, criar a configuração do CORS em ambas APIs para comunicação entre si, e restringir comunicação desnecessária – aumentando a segurança.

De fato, o serviço Amazon Lightsail somado a imagem da Bitnami para as máquinas virtuais simplificou muito todo o processo: dentro do Lightsail o banco de dados PostgreSQL é criado com extrema facilidade e sem nenhuma configuração, escalando automaticamente caso necessário; as máquinas virtuais já são disponibilizadas com uma versão suficientemente nova do NodeJS para rodar ambos o cliente e API de gerenciamento de usuários e avaliações, escritas em JavaScript e TypeScript, respectivamente. O mesmo vale para o Python 3: pelo fato de não ter sido feito uso de funcionalidades mais recentes da linguagem e

das bibliotecas que a linguagem oferece (por exemplo: Pandas e Numpy) instalar uma versão um pouco menos atualizada das bibliotecas bastou para executar o serviço de recomendação sem problemas. Nessa breve descrição da configuração já foi descrita a configuração das máquinas virtuais para executar as aplicações e o banco de dados para integrar com o gerenciador.

Um destaque importante a ser feito é a imagem disponibilizada pela Bitnami (BITNAMI, 2023), que está disponível na AWS Marketplace e já vem integrada no Amazon Lightsail, essa imagem vem com uma ferramenta já integrada na imagem para máquina virtual que permite o host da máquina virtual gerar certificados SSL com facilidade, disponibilizando a URL com protocolo HTTPS, e não HTTP. O domínio obtido para o sistema desenvolvido foi comprado através dos domínios da Google, com um preço de R$50 ao ano, o domínio se chama "themoviebakery.com", que é uma referência a música "The Bakery", do Arctic Monkeys. Com o domínio adquirido, a configuração do DNS é autoexplicativa através das configurações de DNS e Network da plataforma Amazon Lightsail e, para utilizar o domínio em mais de um serviço foi necessário a criação de subdomínios, estes sendo: "backend.themoviebakery.com" e "recommender.themoviebakery.com", ambos com nomes autoexplicativos: o primeiro sendo para o gerenciador em TypeScript, e o segundo sendo para o serviço de recomendação. Outra observação a se fazer é o fato de que foi necessário alocar um IP estático para as aplicações para atribuir os domínios aos IPs, e consequentemente, às máquinas virtuais hospedando os serviços.

E por fim, a configuração do CORS: CORS é um mecanismo baseado nos "Headers" HTTP que permitem que um servidor indique quais chamadas de quais "origens" são permitidas a acessar a aplicação (MOZILLA, 2023). Então mesmo que um domínio esteja executando chamadas a um subdomínio que pertence ao mesmo domínio, exemplo: themoviebakery.com executando uma chamada ao serviço hospedado na URL backend.themoviebakery.com, o serviço que serve essas chamadas precisa de uma configuração concedendo permissão a essa origem específica. A título de curiosidade, uma API pública possui uma configuração de CORS que permite explicitamente que qualquer origem a acesse, e tal API necessita de mais mecanismos de segurança para garantir que não haja abuso no número das requisições executadas ou ataques em seus servidores.

5.2.3 Falhas no design, arquitetura e algumas alternativas

Por se tratar de um protótipo, existem muitas falhas na arquitetura e *design* da solução, no livro de Sommervile (2011) e nas metodologias estudadas no capítulo 4 da presente monografia, durante o estudo sobre abordagens no desenvolvimento de *software* é possível observar o protótipo funcional como a primeira entrega da primeira iteração sobre o produto, somente com o essencial para o sistema fazer o que se propõe: recomendar filmes a um usuário baseando-se nos outros usuários e suas preferências. Dito isso, o sistema está longe de um estado ideal, tanto de um ponto de vista negocial quanto de desenvolvimento e experiência de usuário. Tais observações não tiram o mérito alcançado com o protótipo funcional, mas servem não só para a melhoria do sistema de modo geral em próxima iterações sobre o sistema, iterações estas que vão além do escopo do presente trabalho, mas também enriquece o contexto em que um sistema de recomendação pode atuar, esta sessão é um interlúdio para a sessão final da monografia: trabalhos futuros.

5.2.3.1 Falha de arquitetura: não utilizar o Next.JS como servidor

Next.JS (NEXT.JS, 2023) é um "*framework Full-Stack*", ou seja, a ferramenta permite chamadas a um banco de dados e as devidas validações, o motivo de não utilizar é simplesmente o interesse em hospedar mais serviços no serviço de nuvem da AWS, ou seja, a solução foi projetada para ser mais trabalhosa e exigir mais configuração, por motivos exploratórios antes de mais nada. Vale também observar que a mais sábia decisão em relação a arquitetura seria remover o serviço de gerenciamento e utilizar o Next.JS como gerenciador também, e, pensando num produto que de fato busca crescimento, quando o produto crescesse a ponto de fazer uso de um *backend* próprio, uma reestruturação dessa parte seria bem-vinda, provavelmente utilizando uma linguagem como Java ou Go.

5.2.3.2 Falha de experiência do usuário: botão de “treinamento”

O processo de treinamento do serviço de recomendação é uma questão quase que multidisciplinar: quantas vezes executar o treinamento? Qual métrica ativa a função de treinamento? Se a periodicidade do treinamento for alta o custo computacional será desnecessário, pois chances é de que não houve novos *inputs* o suficiente para justificar o processo de aprendizado – a não ser que seja feita a utilização de “transferência de conhecimento” – que significa dizer que a inteligência artificial irá partir de um conhecimento já obtido e buscar novos conhecimentos a partir dos novos *inputs* (TORREY, 2010) –, de qualquer forma, não foi feito uso da “transferência de conhecimento” no serviço de recomendação – apesar de ser uma funcionalidade relativamente fácil de implementar, dadas especificações atuais do protótipo, outra potencial falha de *design* na arquitetura –. O treinamento do protótipo é feito através de um botão que o usuário pode apertar, o motivo é simples: esse protótipo serve para que um número ínfimo de usuários possa testar o sistema de recomendação, e o treinamento é parte dele, então para acelerar o processo de recomendação e facilitar os testes de usuário e coletar *feedback* explícito através de um formulário, um botão de “treinamento” foi adicionado, e no caso de uso específico proposto, atendeu as necessidades:

Figura 31 – protótipo com botão de “treinamento”

Fonte: autoria própria (2023)

Para uma aplicação em produção e não dedicada a usuários seletos essa opção de treinamento simplesmente não é uma opção. Os autores Ricci, Rokach e Shapira (2015) abordam o tema no decorrer do livro e o curso consumido para o desenvolvimento da monografia e do protótipo funcional sobre sistemas de recomendação em Python (PROGRAMMER, 2023) também aborda a temática de periodicidade.

5.2.3.3 Falha de experiência do usuário: somente filtragem colaborativa baseada em usuário

O sistema de recomendação proposto melhoraria consideravelmente se, ao invés de utilizar filtragem colaborativa baseada em usuário como a principal forma de recomendação, utilizar a recomendação baseada em conteúdo, esta que não depende de *feedback* do usuário de antemão, e então, utilizando um sistema simplificado de recomendação os usuários forneceriam *feedback* com o tempo – nessa parte, mecanismos de coleta de *feedback* implícito fazem toda

diferença para utilizar como indicadores do que os usuários estão gostando e não gostando além das preferências do usuário em particular (RICCI; ROKACH; SHAPIRA, 2015).

O motivo da abordagem do trabalho ter sido a filtragem colaborativa baseada em usuário é puramente experimental: dada uma base de dados preexistente e somando as avaliações dos usuários com esta base de dados, o quanto se mitiga do problema de "arranque a frio", comumente reportado nos sistemas de recomendação.

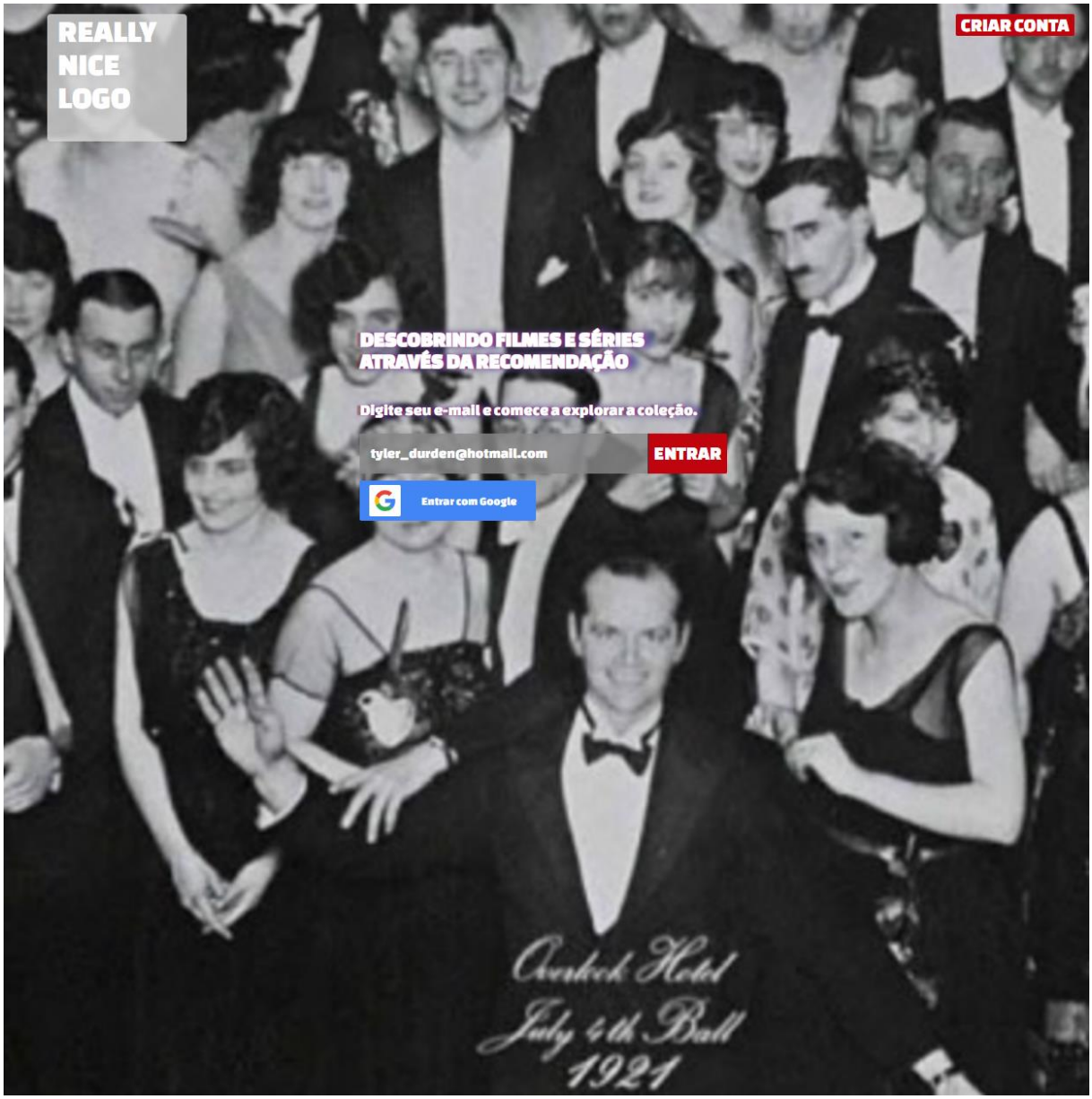
5.3 APRESENTAÇÃO DO PROTÓTIPO

Realizado o desenvolvimento do protótipo funcional nas sessões 5.1 e 5.2, a atual sessão é destinada a apresentação do resultado final.

Como já citado durante o trabalho, o desenvolvimento do sistema de recomendação utilizou uma métrica e somente uma métrica como parâmetro para as recomendações ao usuário: o *feedback* explícito do usuário sobre os filmes apresentados no sistema de recomendação. Para essa apresentação é utilizado o usuário (ou *persona*) "Tyler" para demonstrar todo o fluxo que o usuário do sistema irá percorrer: da criação de conta à solicitação das recomendações. A apresentação demonstra o botão de treinamento, utilizado pelos usuários do teste qualitativo que gerou a sessão 5.4: avaliação.

O primeiro passo é a criação de conta, o usuário desse teste – a partir de agora chamado de Tyler – precisa criar sua conta para utilizar o serviço de recomendação (Figura 32):

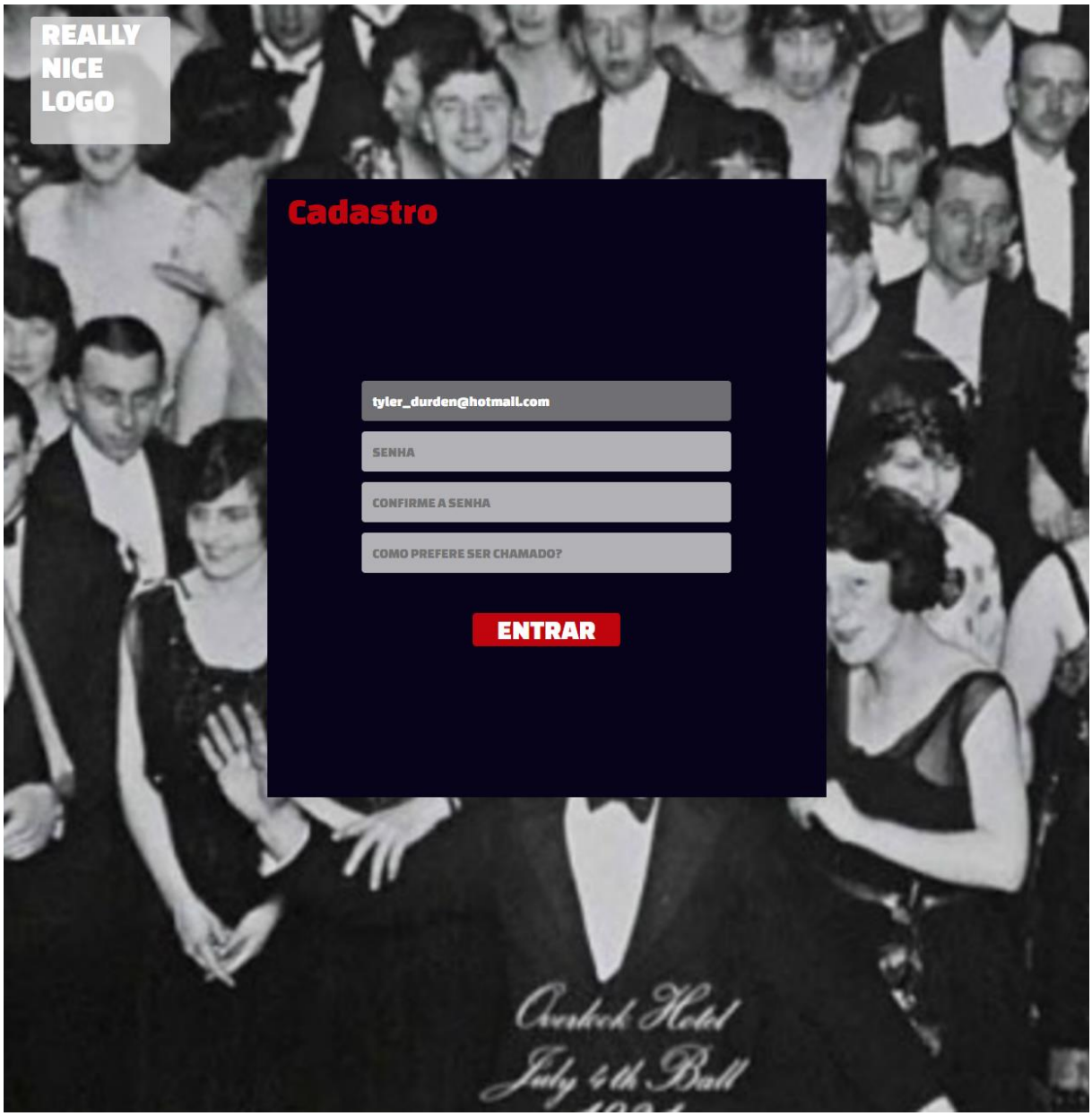
Figura 32 - tela inicial, onde o usuário insere o e-mail



Fonte: autoria própria (2023)

Tyler não possui um e-mail já cadastrado, então ele é redirecionado para a tela de cadastro (Figura 33):

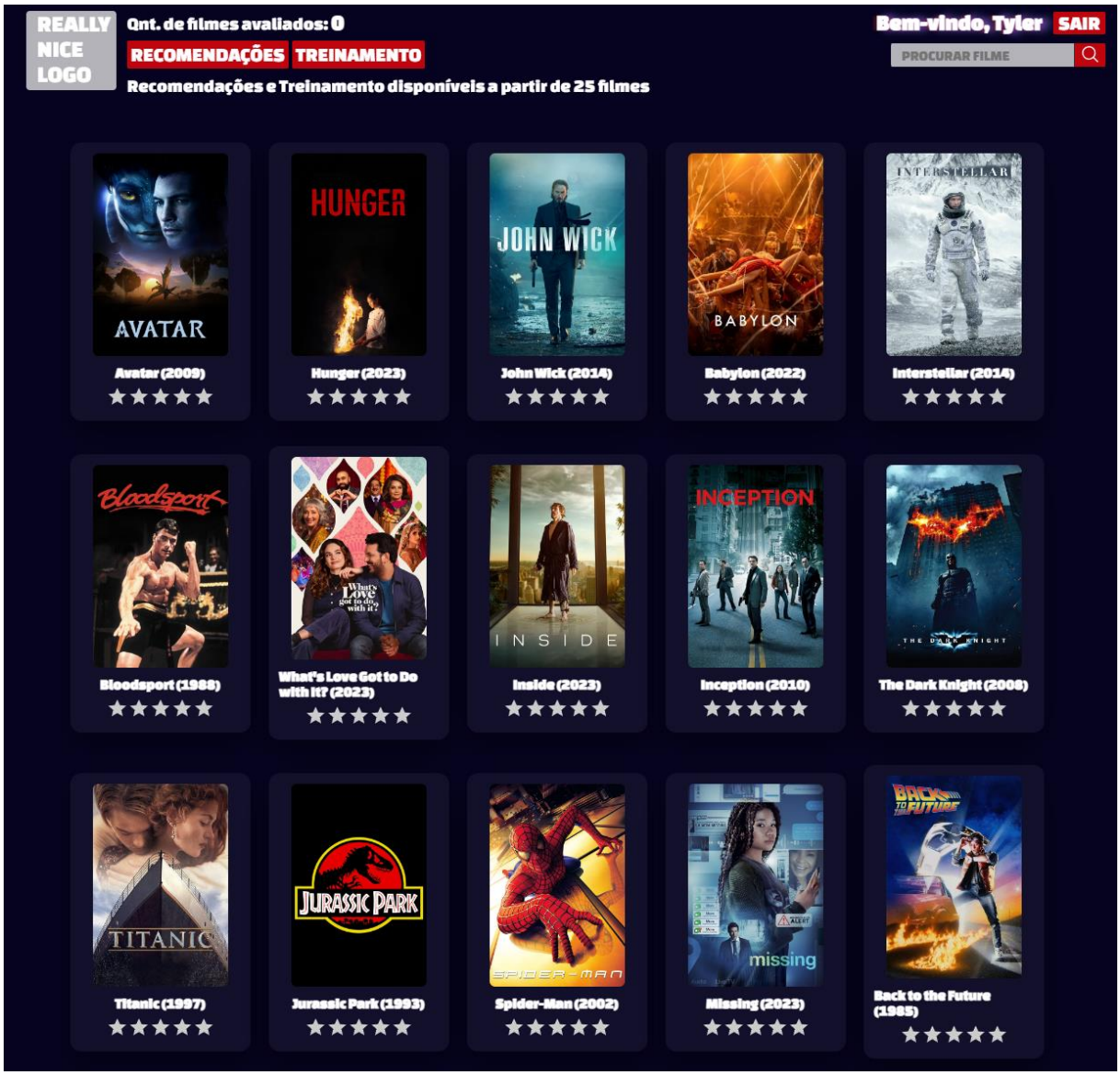
Figura 33 - tela de cadastro de usuário, onde o usuário coloca os dados necessários para seu cadastro no sistema de recomendação



Fonte: autoria própria (2023)

Tendo preenchido os campos que o sistema pediu, Tyler é redirecionado para a tela onde pode avaliar os filmes disponíveis, note que a tela (Figura 34) ficou diferente do protótipo apresentado no capítulo 4 da presente monografia, o motivo disso é a soma dos *feedbacks* dos usuários que realizaram alguns testes durante o desenvolvimento e as necessidades que foram se apresentando ao longo do desenvolvimento para guiar melhor o usuário durante sua experiência:

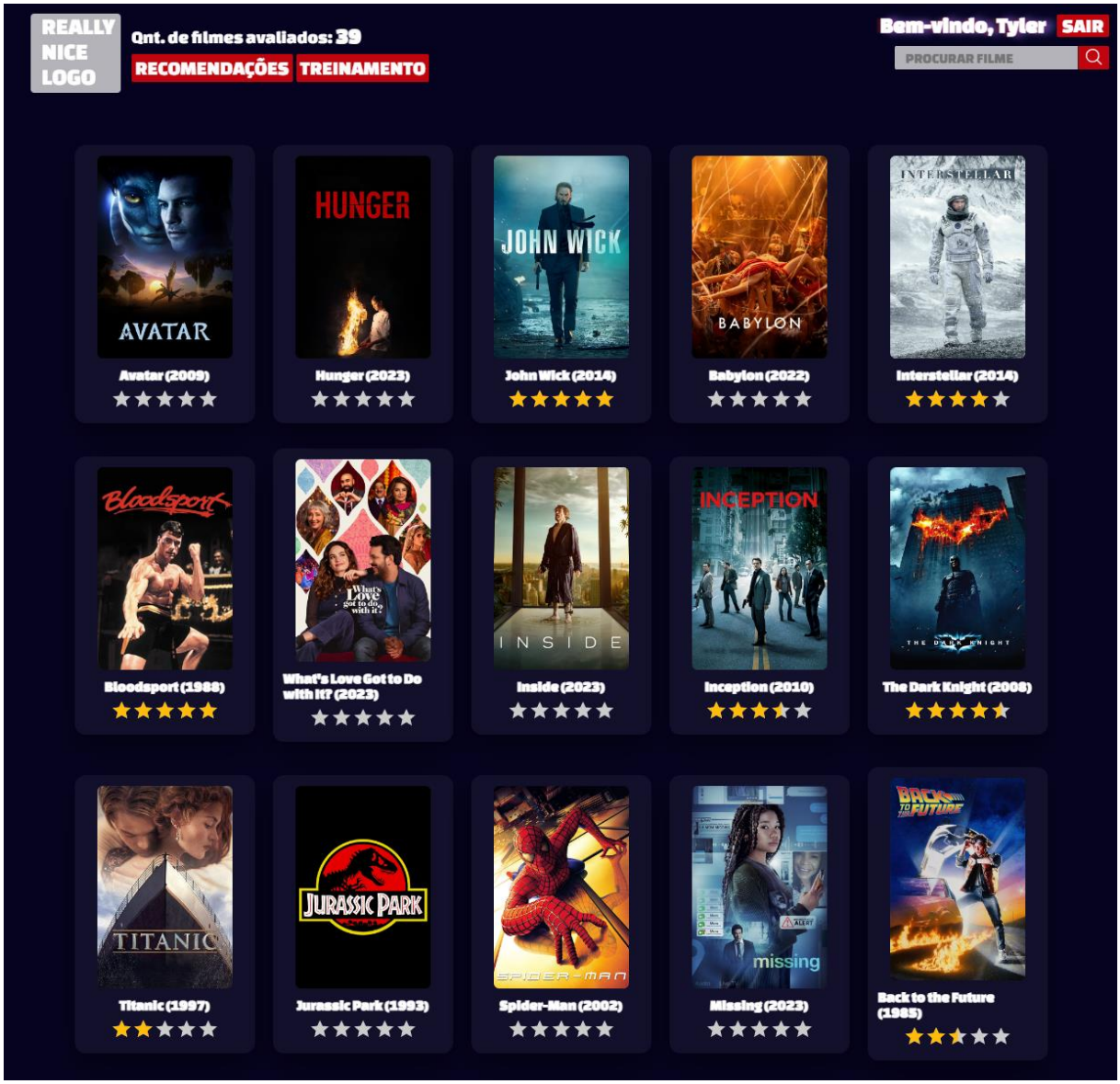
Figura 34 – tela de avaliação de filmes, onde o usuário precisa avaliar ao menos 25 filmes até os botões “treinamento” e “recomendações” fiquem disponíveis



Fonte: autoria própria (2023)

Após realizar a avaliação de 39 filmes, Tyler já pode pedir para treinar o sistema para gerar suas recomendações (Figura 35):

Figura 35 - tela de avaliação de filmes, agora com um número maior de 25 avaliações



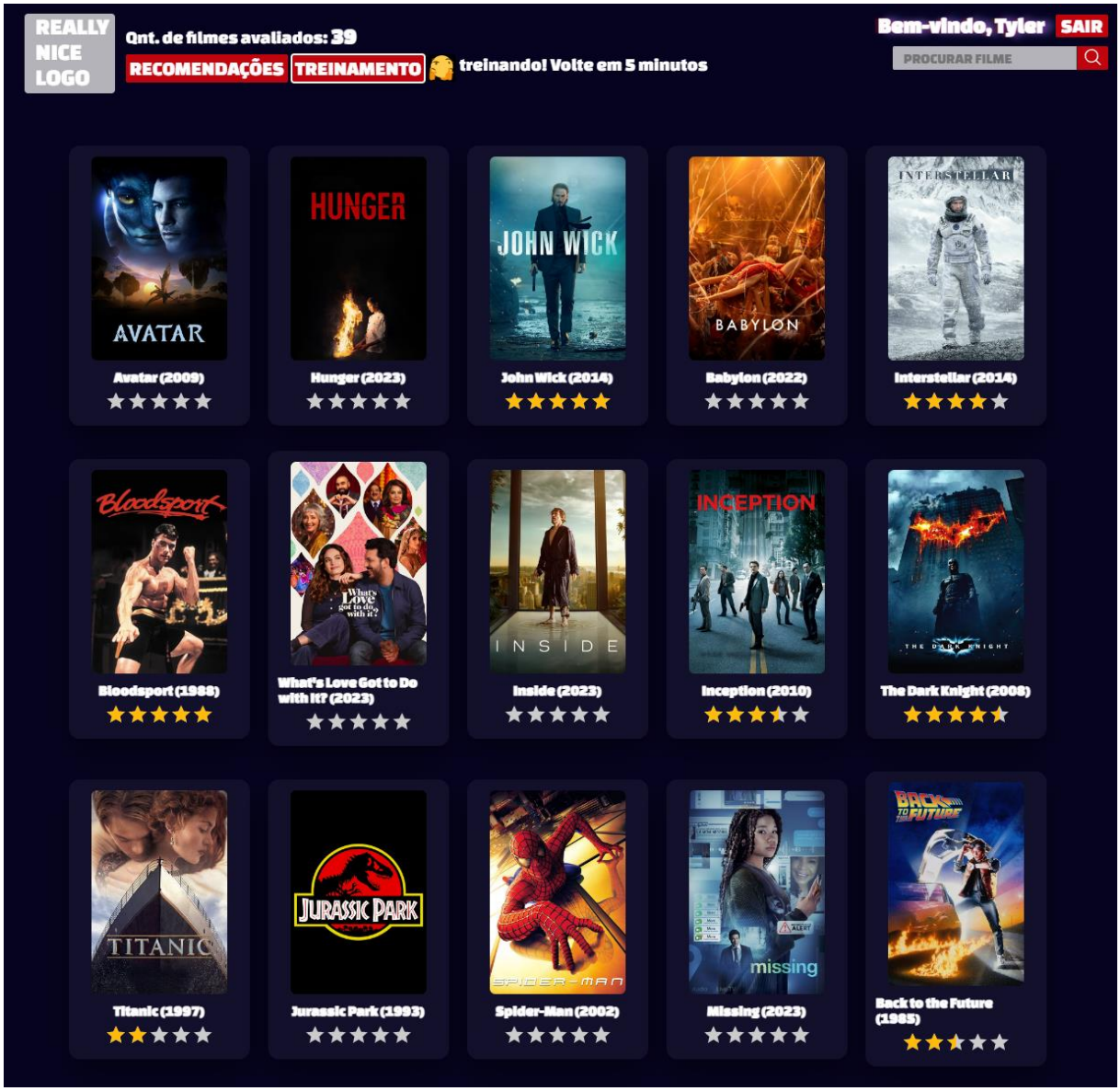
Fonte: autoria própria (2023)

Ao clicar no botão de treinamento, um *gif* aparece ao lado de um texto pedindo para esperar uns 5 minutos antes de solicitar a recomendação. Isso se dá porque (1) essa requisição realizada do serviço cliente ao serviço de recomendação retorna um erro, isso porque a maneira como a solicitação de treino é lidada internamente no serviço não está adequada (e como já mencionado, o botão de “treinamento” não é a forma correta de se treinar um modelo de recomendação), mas mesmo retornando um erro, o serviço consegue executar o treinamento; (2) o algoritmo de treinamento ainda está mal otimizado, portanto treinar do zero acaba sendo custoso e levando 5 minutos toda vez que realiza o treino. Tendo em vista os pontos (1) e (2) que acabam de ser levantados, não é possível mostrar o avanço do treinamento para o usuário ou retornar o *feedback* quando o treinamento foi de fato concluído, mas é possível avisá-lo o

tempo de duração da fase de treinamento, o tempo foi deduzido após inúmeras iterações do algoritmo na parte de desenvolvimento da monografia.

A Figura 36 demonstra o *gif* na tela de avaliação quando Tyler clica no botão “treinamento”:

Figura 36 - *gif* ao clicar no botão de treinamento

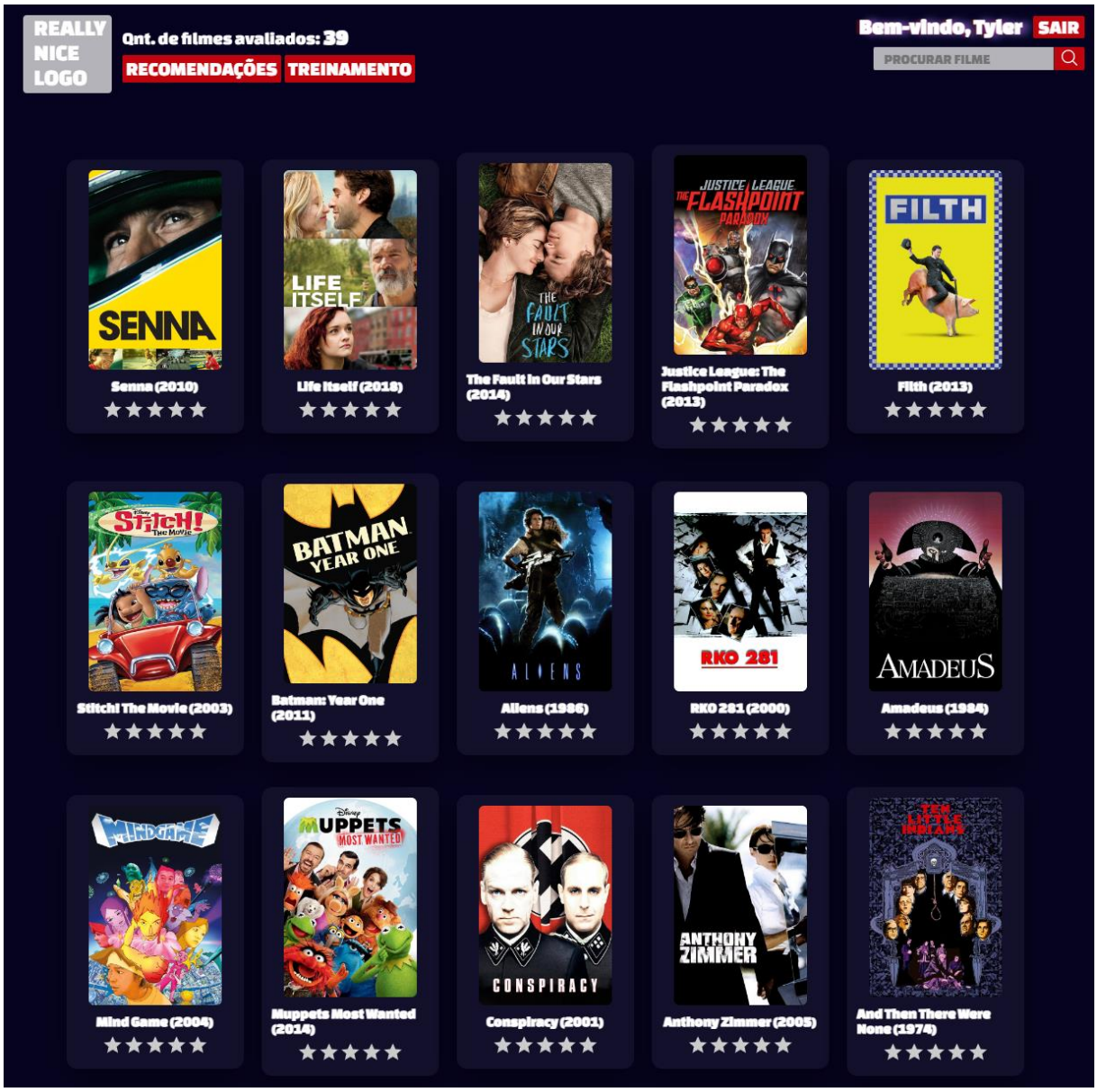


Fonte: autoria própria (2023)

Agora Tyler precisa esperar os 5 minutos. Ao idealizar um sistema de recomendação – ou qualquer sistema no geral –, ter um mecanismo que faz o usuário esperar é uma falha grande de *design*. Ter mais de um método de recomendação num sistema de recomendação mitiga essa falha e permite flexibilidade, fazendo com que o sistema possa trocar

o modelo de recomendação e testar inúmeras abordagens sem que o usuário perceba, tornando a experiência muito mais fluída e natural (RICCI; ROKACH; SHAPIRA, 2015).

Figura 37 - recomendações fornecidas ao usuário Tyler



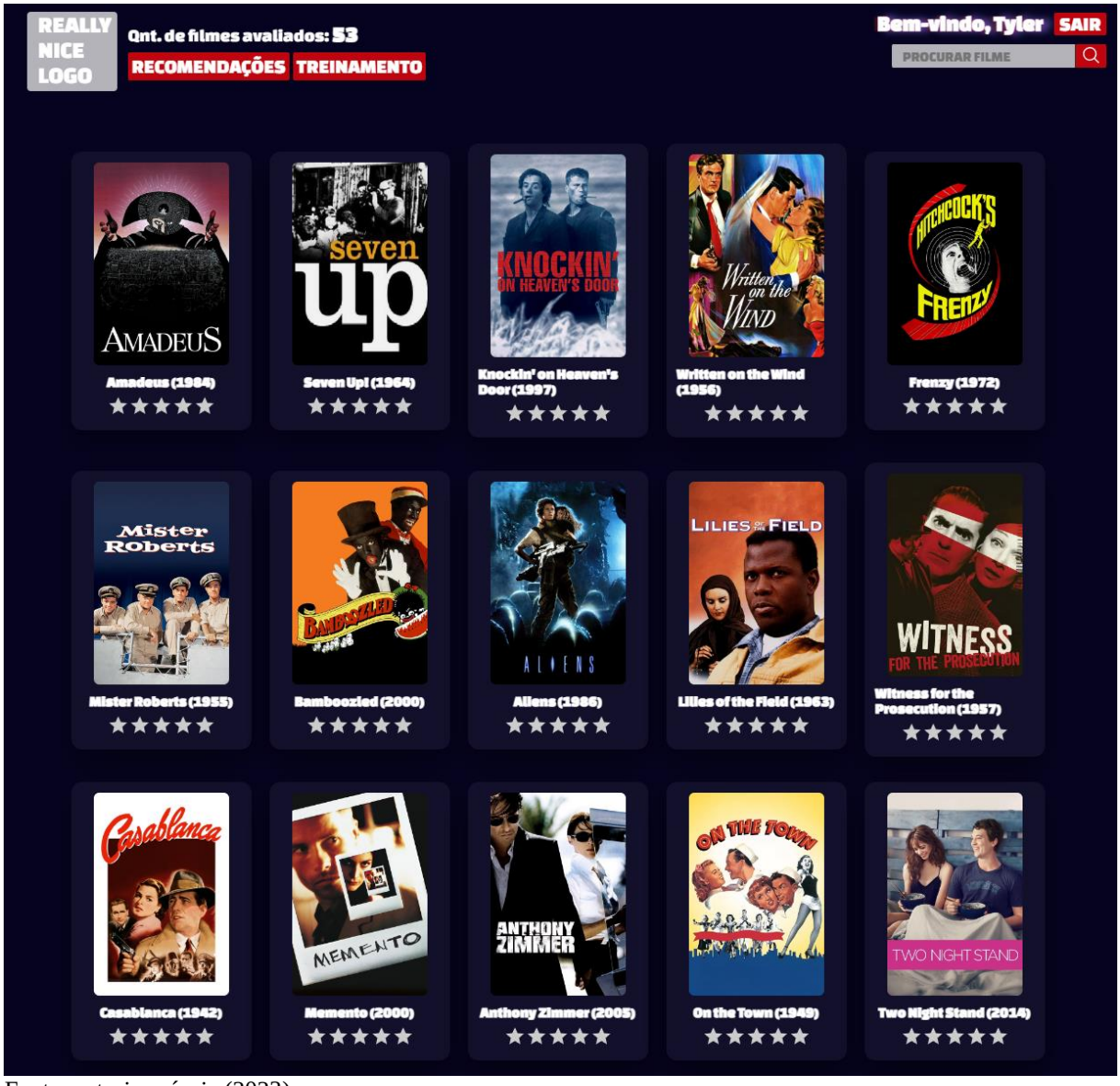
Fonte: autoria própria (2023)

A Figura 37 demonstra as recomendações de Tyler. Dado que muitos dos filmes que Tyler avaliou não tinham nada em comum com os filmes recomendados, as recomendações feitas para Tyler não foram muito boas.

Talvez se Tyler avaliar mais filmes o sistema pode adiciona os vizinhos certos no cálculo, e assim, realizar as recomendações de forma mais assertiva. Novamente Tyler passa

pelo processo de avaliação, dessa vez dos filmes que fora recomendado ao solicitar as recomendações, e agora o sistema precisa novamente passar por uma fase de treinamento.

Figura 38 - resultado da segunda solicitação de Tyler por recomendações de filmes[



Fonte: autoria própria (2023)

A Figura 38 demonstra o resultado obtido na segunda recomendação. Na primeira vez em que Tyler solicitou recomendações de filmes o usuário tinha feito 39 avaliações de diferentes filmes; para a segunda solicitação de recomendações Tyler avaliou um total de 53 filmes.

As recomendações estão melhores. Muitos dos filmes avaliados por Tyler antes de solicitar recomendações pela segunda vez foram com avaliação negativa, ou seja, Tyler

informou o que **não** gosta. Nas primeiras 39 avaliações fornecidas ao sistema, Tyler forneceu muito mais notas altas que baixas ao sistema, fazendo com que o sistema não conseguisse calcular o que Tyler **não** gosta e não queria que fosse recomendado. Mesmo assim é importante observar que as recomendações não mudaram tanto, isso se dá porque não há tantos vizinhos cadastrados no treinamento, então a probabilidade dos vizinhos se repetirem é alta, mesmo depois de adicionar avaliações ao sistema, outros fatores já observados no decorrer da monografia também podem fazer parte da resposta por trás das recomendações de Tyler, mas dado o contexto do teste controlado no cenário apresentado, é possível afirmar que este é o primeiro fator a se levar em consideração.

5.4 AVALIAÇÃO

Baseado no protótipo desenvolvido até a sessão atual, esta sessão propõe uma avaliação deste protótipo funcional baseando-se no fluxo do usuário durante do teste de usabilidade de todo o sistema, fluxo esse demonstrado na sessão 4, em especial na sessão 4.3.2 onde é possível ter o ponto de vista da jornada do usuário durante a navegação do sistema, e aplicando um questionário para obter o *feedback* explícito dos usuários selecionados na amostra podemos estabelecer algumas métricas para efetuarmos de forma concisa a análise dos resultados.

O tópico de medição de um sistema de recomendação foi abordado também na sessão 5.1.3.3.1, mas de um ponto de vista um pouco mais isolado, isto é, os autores e o conteúdo da sessão citada se preocupam um pouco mais com a medição da recomendação em si, mas como fica claro, a recomendação realizada por essa parte isolada de todo um sistema tem influência direta do resto do sistema, se preferir.

Dito que a avaliação é do sistema de recomendação, sistema esse que tem como objetivo recomendar filmes ao usuário, o objetivo final portanto se torna aumentar o nível de satisfação do usuário. A satisfação é definida por Moraes et al. *apud* Lovelock e Wright (2003, p. 106) uma “reação emocional de curto prazo ao desempenho específico de um serviço”. Moraes et al. (2003) continua afirmando que “a avaliação da satisfação é passageira, uma vez que cada experiência com o serviço pode causar a satisfação ou insatisfação do usuário”. No trabalho de Ricci, Rokach e Shapira (2015) todas as facetas do sistema de recomendação são

abordadas – e avaliação desse sistema também é analisada ao longo dos capítulos. Os autores afirmam que um desenvolvedor de um sistema de recomendação de dado domínio de aplicação (seja este domínio uma rede social que recomenda pessoas, ou um domínio de entretenimento onde as recomendações são de filmes e séries) deve entender as facetas específicas do domínio, seus requisitos, desafios e limitações.

5.4.1 Formulário

O formulário é composto de 6 perguntas e a pesquisa tem como objetivo obter informações sobre a satisfação do usuário em relação as recomendações e o tempo que o sistema demora para responder ao pedido de recomendação, além de observar o comportamento do usuário: se costuma assistir filmes e, segundo suas próprias observações sobre comportamento, questionar sobre a abordagem que este usuário teve em relação à média de pontos que este forneceu aos filmes avaliados. O formulário foi criado dentro da ferramenta Google Forms e permite a organização dos dados coletados através das respostas gratuitamente (GOOGLE, 2023). Cada pergunta possui sua própria motivação e para que estas motivações sejam compreendidas, vale uma introdução as perguntas do formulário:

(a) “De 0 a 10, sendo 0 péssimo e 10 sendo ótimo, qual nota você daria para as recomendações que o sistema lhe ofereceu?”. Pergunta de múltipla escolha numa escala linear.
(b) “Sobre a velocidade das recomendações (botão de recomendações), o quão rápidas você diria que elas foram calculadas?”. Pergunta de múltipla escolha com as escolhas: (1) muito rápido; (2) rápido; (3) mais ou menos; (4) devagar; e (5) muito devagar.
(c) “No geral, é possível afirmar que você distribuiu notas positivas e negativas sobre os filmes que avaliou?”. Pergunta de múltipla escolha com as escolhas: (1) sim, avaliei filmes tanto com notas positivas quanto negativas; (2) não, avaliei a maioria de forma negativa; (3) não, avaliei a maioria de forma positiva.

(d) “Com que frequência você costuma assistir filmes?”. Pergunta de múltipla escolha com as escolhas: (1) toda semana; (2) quase toda semana; (3) todo mês; (4) quase nunca; (5) nunca.

(e) “Você achou que as recomendações mudaram para melhor ou pior após adicionar ou alterar avaliações?”. Pergunta de múltipla escolha com as escolhas: (1) mudou, para pior; (2) mudou, para melhor; (3) nem percebi que mudou.

(f) “Você utilizaria um sistema para obter recomendações de filmes e séries e também pegar as recomendações de outras pessoas?”. Pergunta de múltipla escolha com as escolhas: (1) sim; (2) não.

As perguntas (a) e (b) têm como objetivo coletar a satisfação do usuário, sendo a (b) mais preocupada com a performance do sistema. As perguntas (c) e (d) têm como objetivo obter o comportamento do usuário, sendo a (c) o comportamento do usuário em relação a sua tendencia de avaliar filmes mais para negativo ou positivo e a (d) observar com que frequência o usuário assiste filmes – é valido comentar que pessoas mais envolvidas no cinema costumam ser mais críticas, então recomendar filmes aclamados pela crítica e com alta reputação costuma ser uma boa estratégia, no entanto, usuários com um perfil contrário, ou seja, que costumam ver filmes que estão na moda e esquecerão destes assim que os créditos aparecerem na tela, não irão apreciar com tanta facilidade filmes que definem o gênero que pertencem, um exemplo de filme que define o gênero que habita é o filme *Se7en*. A pergunta (e) diz a respeito do retreinamento: ao realizar o retreino a recomendação ficou melhor? Pior? Ou ainda, mal houve efeito? Os usuários foram solicitados a avaliar ao menos 25 filmes, e então, depois das recomendações geradas, solicitar mais um treino e recomendação após avaliar mais 25 filmes, essa pergunta também diz a respeito da satisfação do usuário em relação ao retreinamento. A sexta e última pergunta (f) diz a respeito do comportamento do usuário, dado que redes sociais como Filmow (FILMOW, 2023) estão online e ativas, com usuário de todo mundo utilizando seus serviços para encontrar filmes e criar listas com filmes vistos e filmes que desejam ver, o usuário se descreveria como alguém que utilizaria uma “rede social para cinéfilos”? Ou no caso do presente trabalho, um “sistema de recomendação de filmes”.

5.5 RESULTADOS

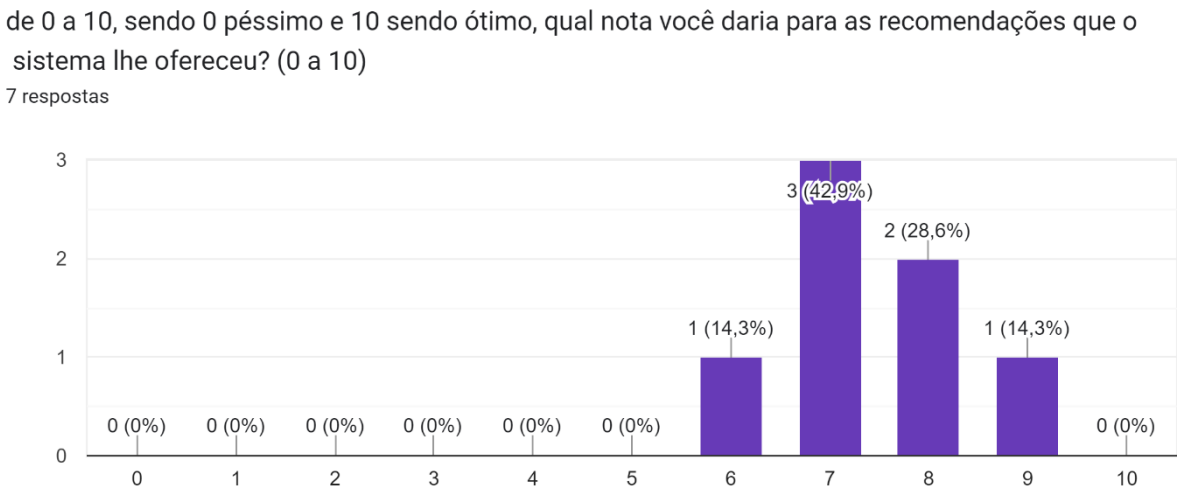
Na sessão 8.2.4 do livro de Ricci, Rokach e Shapira (2015), nomeado de “tirando conclusões confiáveis” os autores argumentam sobre o fato de que em qualquer tipo de experimento é importante que os desenvolvedores se deem confiantes em relação ao sistema de recomendação que escolheram para lidar com os dados ainda não vistos no futuro. Ainda assim, é possível que determinado algoritmo performe melhor num ambiente controlado e obtenha resultados inferiores ao esperado, considerando os dados imputados pelos dados utilizando os sistemas recomendação escolhidos. Para reduzir a possibilidade de tais erros estatísticos os autores recomendam performar testes de significância sobre os resultados.

Considerando o fator tempo, a realização dessa “análise de longa distância” sobre o presente trabalho não se faz possível ou necessária, mas a soma de técnicas de recomendação sobre a técnica já implantada, tal técnica de recomendação sendo a filtragem colaborativa baseada em usuário, se faz útil num sistema em que o objetivo não é explicitamente a experimentação, mas sim a satisfação dos usuários deste sistema.

Idealmente se traria uma massa de usuários considerável e demograficamente diversa, porém, considerando que os serviços estão hospedados de forma pública e não possui os devidos mecanismos de defesa e segurança para impedir ataques simples – mas severos – que facilmente derrubariam o sistema desenvolvido para o protótipo funcional, a amostra de usuários para o teste foi efetuada de forma controlada, afim de não convidar usuários maliciosos a inundar os servidores de requisições ou explorar outras falhas no *software* desenvolvido (MEDEIROS, 2016) (HOQUE; BHATTACHARYYA; KALITA, 2015).

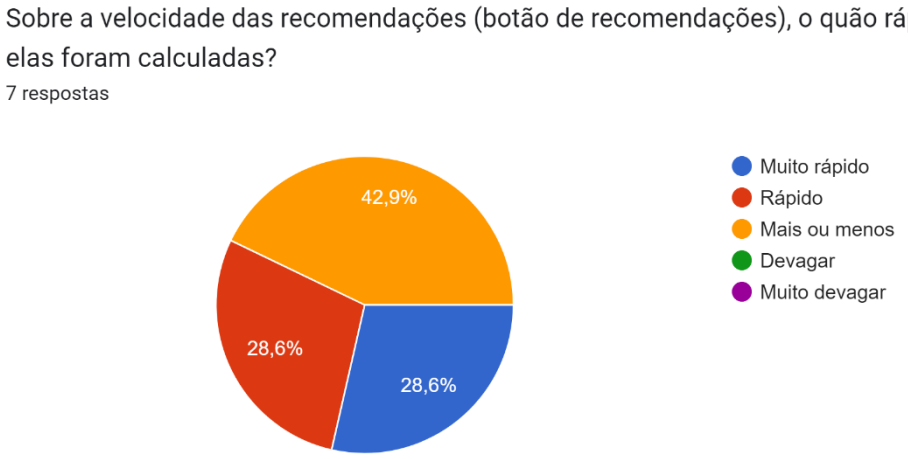
Tendo revisado o formulário aplicado e o contexto em que o teste foi conduzido, as Figuras 39–44 correspondem aos resultados obtidos a partir da aplicação do formulário:

Figura 39 – Respostas fornecidas para pergunta (a), representadas num gráfico de colunas



Fonte: autoria própria (2023)

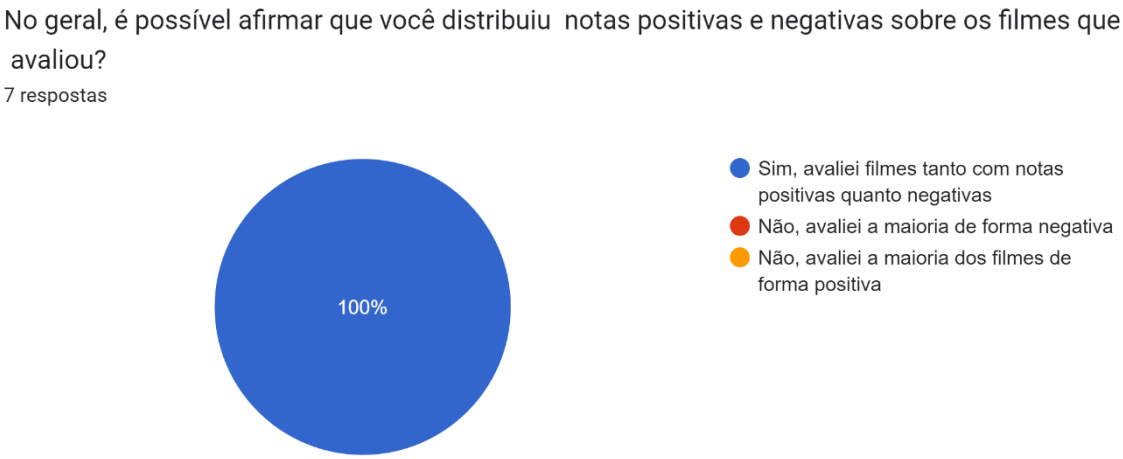
Figura 40 – Respostas fornecidas para pergunta (b), representadas num gráfico de setores



Fonte: autoria própria (2023)

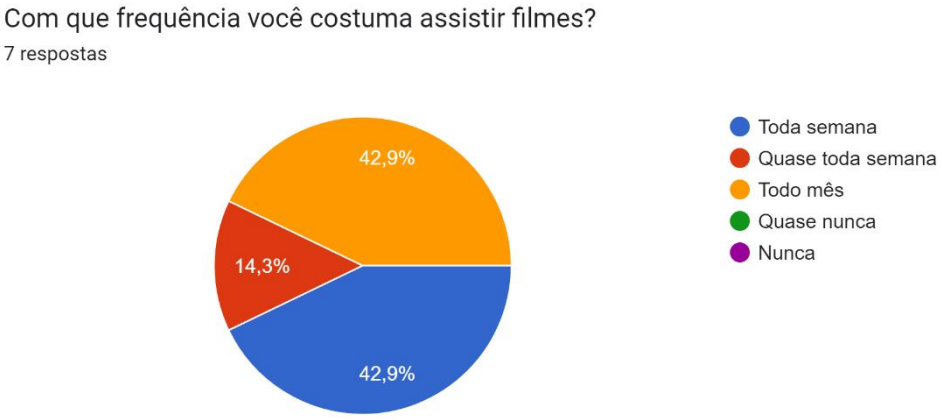
As respostas para pergunta (b), apresentadas no gráfico da Figura 40 demonstram um ponto positivo do sistema de recomendação apresentado pelo protótipo: o sistema calcula com antecedência as recomendações para o usuário e assim pode entregar tais recomendações assim que o usuário pedir. O motivo da haver 42,9% de respostas como “Mais ou menos” poder ser o fato de o serviço cliente ter que buscar as imagens de fora sempre que executa, ao invés de utilizar cache, possuir os arquivos de imagem localmente ou qualquer outra técnica que aumente a velocidade da request.

Figura 41 – Respostas fornecidas para pergunta (c), representadas num gráfico de setores



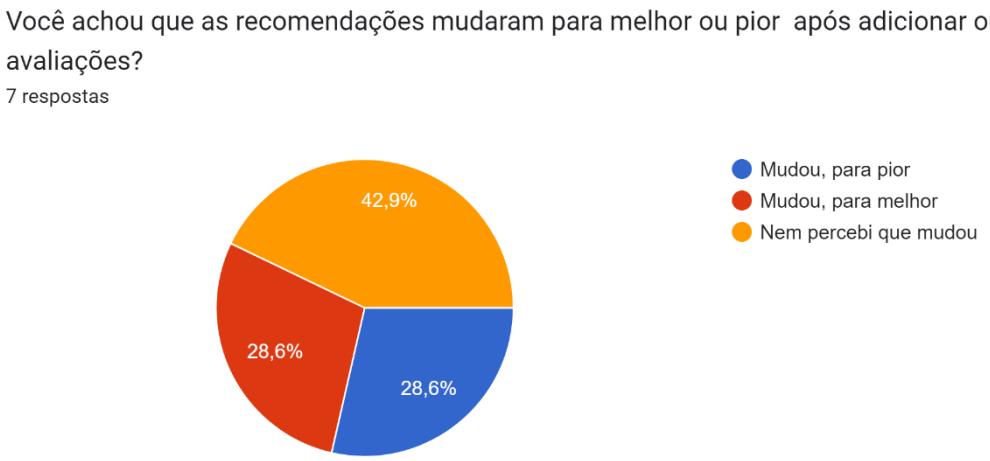
Fonte: autoria própria (2023)

Figura 42 – Respostas fornecidas para pergunta (d), representadas num gráfico de setores



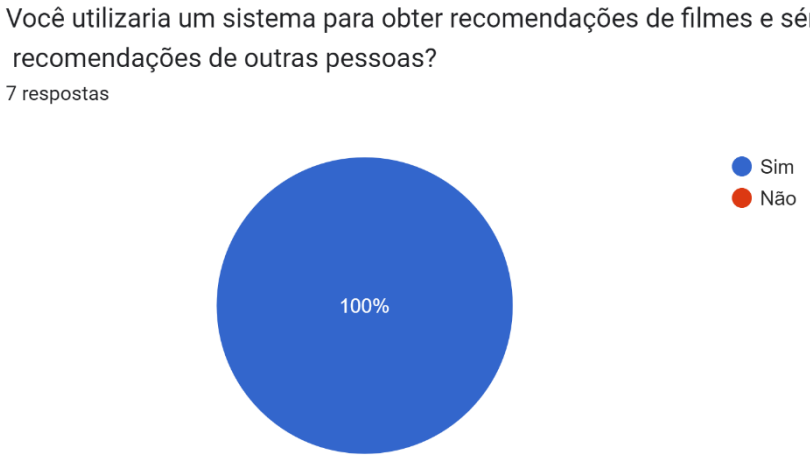
Fonte: autoria própria (2023)

Figura 43 – Respostas fornecidas para pergunta (e), representadas num gráfico de setores



Fonte: autoria própria (2023)

Figura 44 – Respostas fornecidas para pergunta (f), representadas num gráfico de setores



Fonte: autoria própria (2023)

Apesar da pequena amostra de usuários, é possível traçar algumas hipóteses. Com base nas respostas para as perguntas (a) (Figura 39) e (e) (Figura 43) em especial, onde fica evidente o nível de satisfação do usuário de forma explícita é possível observar que, para um modelo simplório de recomendação que utiliza filtragem colaborativa baseada em usuários, além desta filtragem ainda utilizando a abordagem baseada em vizinhos (ou *kNN*, se preferir) o *feedback* foi surpreendentemente positivo para tal algoritmo. Se for levado em consideração somente a técnica de recomendação e a avaliação dos filmes pelos usuários, e construir uma avaliação do modelo somente levando essas duas variáveis, estas sendo: (1) o modelo de

recomendação; e (2) as avaliações dos usuários em si no formulário; é possível inferir que há outros fatores puxando as avaliações para cima. A partir dos resultados obtidos o objetivo então é realizar essa inferência de possíveis hipóteses, e então concluir o capítulo de resultados.

O resultado positivo também se deve ao fato de que, conforme demonstra as respostas para pergunta (c) (Figura 41), os usuários foram explicitamente incentivados a avaliarem filmes tanto de forma positiva quanto de forma negativa, para não ocorrer o problema da ovelha cinza.

5.5.1 Sobre os usuários escolhidos para o teste

Os usuários que participaram do teste e preencheram o formulário eram, em sua maioria, pessoas que gostavam de cinema ou que ao menos gostavam de parar para ver filmes, o que faz sentido para o contexto do objeto sendo testado. O fato é que quando se assiste vários filmes e se têm conhecimentos da sétima arte, e a lista é composta de filmes famosos (filmes esses que serão abordados logo em seguida nas deduções vindas dos testes), não é uma surpresa que a chance de a recomendação ser boa é alta. É uma tendência humana. Na sessão 2.1 da atual monografia, onde se inicia a dissertação científica sobre preferências, é definida a palavra "preferência" seguindo a definição de Druckman e Lupia (2000) onde "preferência é definida como uma avaliação comparativa de um conjunto de objetos, ou seja, uma classificação sobre [tal preferência]. A preferência serve como marcador cognitivo que lembra a pessoa como interagir com aspectos de seu ambiente. [...]" e, de fato, a faixa etária das pessoas fazendo os testes era de 20 a 30 anos e estas, ao navegarem pela listagem de filmes acabaram se identificando com os resultados obtidos.

Isso faz com que o conjunto de usuários seja a (3) variável a ser considerada ao observar os resultados o formulário. Para entender uma opinião, é preciso entender a pessoa por trás daquela opinião (SOLOMON; HIGGINS apud NIETZSCHE, 1999).

5.5.2 Sobre os filmes disponíveis no dataset

O *dataset* foi montado com os filmes mais populares no momento da coleta, isto é, começo de 2023. Dito isso, filmes como *Cinderella* (1950), *Toy Story* (1995) e *Spider-Man* (2002) estão entre os filmes listados. Estes são alguns dos filmes que marcaram a infância de muitas pessoas da faixa etária exposta ao teste, ou seja, boas memórias de bons filmes que a pessoa avaliou – provavelmente com uma pontuação alta – ou não chegou a avaliar, e então teve esta recomendação feita para si. O importante é ter em mente que os pouco mais de 5 mil filmes dispostos para avaliação nos testes realizados são, antes de mais nada, filmes populares.

Se faz necessário observar a *Netflix* para entender melhor as colocações da sessão atual, o serviço de *streaming* diminuiu drasticamente seu catálogo de filmes e séries, mas investiu pesado em produções próprias de filmes e séries (CLARK, 2020), ou seja, a empresa não visa ter todos os títulos que o usuário procura, mas sim ter títulos originais – e produzidos por ela mesma – que atenda seu público alvo. É importante ressaltar que o artigo de Clark (2020) para o *Business Insider* só aponta os dados, e aqui as hipóteses são traçadas a partir destes dados.

Ou seja, o conjunto de filmes é a variável (4) do conjunto de itens que influencia nos resultados do sistema de recomendação, e esse é o tipo de fato que soa obvio depois de afirmado, mas pode passar batido quando a busca é, por exemplo: maior acerto nas recomendações. Não é possível recomendar “a melhor espécie de alho para um vampiro”, por assim dizer. Se determinado *software* se propõe a atender determinado público alvo, é melhor o público alvo ser bem estabelecido, compreendido, e então atendido.

6 CONCLUSÕES E TRABALHOS FUTUROS

O presente capítulo se dedica a concluir a monografia, apresentando os objetivos definidos e como estes objetivos foram alcançados, além de possíveis extensões do atual trabalho desenvolvido e os possíveis trabalhos futuros que poderão ser realizados, partindo da atual monografia.

6.1 CONCLUSÕES

Tendo em vista a complexidade da implementação de um sistema de recomendação eficaz e o quão simples este mesmo sistema pode ser, os resultados mostram que um sistema de recomendação que possui em seu catálogo de itens com excelentes opções para aquilo que se está recomendando, no caso do presente trabalho: filmes, o sistema recomenda-se sozinho.

Para casos específicos como o sistema abordado no protótipo funcional foi possível atingir bons resultados utilizando uma base de dados inicial e um algoritmo limitado para calcular as recomendações. Mas, serviços como a Amazon que possuem uma variedade de itens e clientes tão diferentes entre si, a solução para o problema de recomendação pode se tornar um emaranhado de diferentes técnicas e abordagens, somando diferentes aspectos dos usuários e dos itens disponíveis para realizar uma recomendação apropriada, solução esta que a Amazon parece estar dominando até então (PALMER, 2022).

É possível observar que o arquiteto de um sistema de recomendação precisa entender, em especial, as restrições que envolvem o sistema de recomendação que se pretende construir, e partindo dessas restrições é possível mitiga-las com as devidas ferramentas e técnicas. No caso do protótipo funcional desenvolvido, o modo como foi contornado o que segundo a literatura é o maior empecilho para sistemas de recomendação: o “arranque a frio”, foi adotando uma base de dados preexistente e somando os dados já conhecidos da base com os dados obtidos pelos usuários do sistema, mas é possível ver que a falta de *feedback* dos usuários e a volatilidade de suas opiniões é de fato o maior dos desafios no desenvolvimento de um sistema de recomendação.

Mesmo com resultados satisfatórios, um cenário de testes mais realista seria a divulgação do sistema inteiro para o grande público, permitindo com que grupos de milhares de pessoas avaliassem filmes, solicitassem recomendações, e preenchessem o formulário disponível, mas como já mencionado anteriormente esta abordagem não foi realizada por uma limitação de tempo somada a insegurança de divulgar uma URL pública a muitos estranhos ao mesmo tempo, sem adicionar as devidas camadas de segurança.

Por fim, a conclusão desse protótipo demonstra uma das diversas possibilidades de se implementar um sistema de recomendação. O fato de se ter desenvolvido um sistema de recomendação sem a utilização de bibliotecas especializadas e sim fazendo uso da matemática e de bibliotecas como *Numpy* para auxiliar em operações mais complexas com maior performance trouxe uma desmistificação ao processo de recomendação, e também ao processo de construção de modelos de aprendizagem de máquina. Tendo atingido os objetivos definidos no começo da monografia, conclui-se o trabalho apresentando possíveis trabalhos futuros.

6.2 TRABALHOS FUTUROS

Ao fim do protótipo funcional no capítulo 5 foram definidas melhorias ao protótipo atual e expansões deste protótipo. A diferença de um para outro não é clara de forma semântica, mas ao observar os itens descritos é possível visualizar a diferença entre uma melhoria e uma expansão, caso não tenha ficado claro até então:

Como melhoria, faz-se útil pensar em:

(a) Se o *software* almeja atender melhor o público, as estratégias de recomendação precisam ser atualizadas e definidas de forma dinâmica, para usuários novos os conteúdos precisam se organizar entre si, para usuários já conhecidos, a filtragem colaborativa precisa se mostrar melhor que uma simples filtragem de conteúdo;

(b) As melhorias e falhas citadas na sessão 5.2.3 se fazem útil nesta sessão;

(c) A inclusão de filmes de bases de dados como TMDB farão com que os usuários busquem o sistema desenvolvido como referência, então um *script* diário de atualização baseado num site de crítica de cinema pode fazer com que o sistema atinja esse objetivo.

Como expansão, faz-se útil considerar:

(a) Uma expansão interessante seria o usuário clicar no filme desejado e este filme possui suas próprias propriedades e atributos;

(b) Pessoas gostam de criar listas e listar coisas. Fazer com que usuários sejam capazes de criar listas para si com um nome e talvez *tags* para estas listas seria interessante;

(c) Além de criar listas, permitir usuários listarem filmes que determinado diretor dirigiu, ou ator atuou, por exemplo, é uma excelente expansão.

(d) Além do *script* citado na lista anterior, permitir o usuário a adicionar os próprios filmes – no caso de filmes menores que são produzidos com baixo orçamento e não costumam ser listados por bases de dados como TMDB – seria uma boa forma de incentivar os usuários a contribuir com o site.

REFERÊNCIAS

CURRY, David. **Video Streaming App Revenue and Usage Statistics (2022)**. Disponível em: <https://www.businessofapps.com/data/video-streaming-app-market/#:~:text=The%20video%20streaming%20industry%20reached,reach%20%24115%20billion%20by%202026>. Acesso em 24 ago. de 2022

AMATRIAIN, Xavier. **Big & Personal: data and models behind Netflix recommendations**. Disponível em: <https://amatriain.net/pubs/BigAndPersonal.pdf>. Acesso em: 21 ago. de 2022

ABERNETHY, Jacob; BACH, Francis; EVGENIOU, Theodoros; VERT, Jean-Philippe. **A New Approach to Collaborative Filtering: Operator Estimation with Spectral Regularization**. Disponível em: <https://www.jmlr.org/papers/volume10/abernethy09a/abernethy09a.pdf>. Acesso em 21 ago. De 2022

KOREN, Yehuda; BELL, Robert; VOLINSKY, Chris. **MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS**. Disponível em: <https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf>. Acesso em 23 ago. de 2022

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. **Recommender Systems Handbook**. Disponível em: < https://www.researchgate.net/publication/227268858_Recommender_Systems_Handbook>. Acesso em 23 ago. De 2022

GOLDBERG, David; NICHOLS, David; OKI, Brian M.; TERRY, Douglas. **Using COLLABORATIVE FILTERING to Weave na Information TAPESTRY**. Disponível em: <https://dl.acm.org/doi/abs/10.1145/138859.138867>. Acesso em 28 ago. de 2022

DIETRICH, Franz; LIST, Christian. **Where do preferences come from?** Disponível em: <http://eprints.lse.ac.uk/46864/1/Where%20do%20preferences%20come%20from(lsero).pdf> . Acesso em 03 set. de 2022

DRUCKMAN, James N.; LUPIA, Arthur. **Preference Formation**. Disponível em: <http://www.u.arizona.edu/~zshipley/pol431/PreferenceFormation.pdf>. Acesso em 10 set. de 2022

NEWELL, Allen; LEWIS, Richard; HUFFMAN, Scott; JOHN, Bonnie; LAIRD, John; LEHMAN, Jill; ROSENBLOOM, Paul; SIMON, Tony; TESSLER, Shirley. **SOAR AS A UNIFIED THEORY OF COGNITION**. DYER, James; JIA, Jianmin. **PREFERENCE THEORY**. Disponível em: <https://www.researchgate.net/publication/304156019_Preference_theory> Acesso em 11 set. de 2022

CLARK, Andy. **Being There: Putting Brain, and World Together Again**. Londres, Inglaterra. 1997

CLARK, Travis. **Netflix has lost 3,000 movies in the last decade, but its originals catalog has soared to 1,100**. Disponível em: < https://www.businessinsider.com/how-netflix-movie-and-tv-show-catalog-changed-over-time-2020-2>. Acesso em 12 set. de 2022

STOLL, Julia. **Number of Netflix paid subscribers worldwide from 1st quarter 2013 to 2nd quarter 2022**. Disponível em: < https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide>. Acesso em 12 set. de 2022

GRUYTER, Walter de; **Preferences**. Berlin – New York. 1998

HOLLAND, John Henry; THAGARD, Paul; NISBETT, Richard. **Introduction: Processes of Inference, Learning and Discovery.** Cambridge, Massachusetts. 1986

RESNICK, Paul; VARIAN, Hal. R. Recommender Systems. disponível em: < https://dl.acm.org/doi/pdf/10.1145/245108.245121>. Acesso em 17 set. de 2022

HUTTNER, Joseph. **From Tapestry to SVD A Survey of the Algorithms That Power Recommender Systems**. Disponível em: < https://scholarship.tricolib.brynmawr.edu/handle/10066/3706>. Acesso em 18 set. de 2022

LÜ, Linyuan; MEDO, Matúš; YEUNG, Chi Ho; ZHANG, Yi-Cheng; ZHANG, Zi-Ke. **Physics Report: Recommender systems**. Acesso em: <https://www.researchgate.net/publication/221662122_Recommender_Systems>. Acesso em 19 set. de 2022

ADOMAVICIUS, Gedeminas; TUZHILIN, Alexander. **Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions**. Disponível em: < https://ieeexplore.ieee.org/abstract/document/1423975>. Acesso em 21 set. de 2022

BRANDT, Richard B. **The Rational Criticism of Preferences**. Disponível em: <https://philpapers.org/rec/BRATRC>. Acesso em 02 nov. de 2022

PRIMO, Tiago Thompsen. **Método de representação de conhecimento baseado em Ontologias para apoiar Sistemas de Recomendação Educacionais**. Disponível em: < https://www.lume.ufrgs.br/handle/10183/83654>. Acesso em 23 set. de 2022

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A; GUTIÉRREZ, A. **Recommender systems survey**. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0950705113001044>. Acesso em 25 set. de 2022

METEREN, Robin van; SOMEREN, Maarten van. **Using Content-Based Filtering for Recommendation**. Disponível em: <http://users.ics.forth.gr/~potamias/mlnia/paper_6.pdf>. Acessado em 29 set. de 2022

BALABANOVIĆ, Marko; SHOHAM, YOAV. **Fab: Content-Based, Collaborative Recommendation**. Disponível em: <https://dl.acm.org/doi/pdf/10.1145/245108.245124>. Acesso em 30 set. de 2022

BILLSUS, Daniel; PAZZANI, Michael J. **User Modeling for Adaptative New Access**. Disponível em: <https://link.springer.com/article/10.1023/A:1026501525781>. Acesso em 30 set. de 2022

SHARDANAND, Upendra; MAES, Pattie. **Social Information Filtering: Algorithms for Automating "Word of Mouth"**. Disponível em: <https://dl.acm.org/doi/fullHtml/10.1145/223904.223931>. Acesso em 30 set. de 2022

RESNICK, Paul; IACOVOU, Neophytos; SUCKAK, Mitesh; BERSTROM, Peter; REIDL, John. **GroupLens: An Open Architecture for Collaborative Filtering of Netnews**. Disponível em: <http://sistemas-humano-computacionais.wdfiles.com/local--files/capitulo%3Aredes-sociais/GroupLens.pdf>. Acesso em 01 out. de 2022

HERLOCKER, Jonathan L.; KONSTAR, Joseph A.; BORCHERS, Al; RIEDL, John. An **Algorithmic Framework for Performing Collaborative Filtering**. Disponível em: < http://files.grouplens.org/papers/algs.pdf >. Acesso em 01 out. de 2022

SCHAFER, J. Ben; KONSTAN, Joseph; RIEDL, John. **Recommender Systems in E-Commerce**. Disponível em: <https://www.researchgate.net/publication/2507550_Recommender_Systems_in_E-Commerce>. Acesso em 01 out. de 2022

PRIMO, Tiago Thompsen. **Método de representação de conhecimento baseado em Ontologias para apoiar Sistemas de Recomendação Educacionais**. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/83654/000905491.pdf?sequence=1>. Acesso em 02 out. de 2022

BREESE, John S.; HECKERMAN, David; KADIE, Carl. **Empirical Analysis of Predictive Algorithms for Collaborative Filtering**. Disponível em: <https://arxiv.org/pdf/1301.7363>. Acesso em 06 out. de 2022

HOFMANN, Thomas. **Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis**. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.2476&rep=rep1&type=pdf>. Acesso em 06 out. de 2022

BLEI, David M.; NG, Andrew Y.; JORDAN, Michael I**. Journal of Machine Learning Research 3 – Latent Dirichlet Allocation**. Disponível em: <https://dl.acm.org/toc/jmlr/2003/3/null>. Acesso em 06 out. de 2022

ZITNICK, C. Laurence; KANADE, Takeo. **Maximum Entropy for Collaborative Filtering**. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.1729&rep=rep1&type=pdf>. Acesso em 07 out. de 2022

SALAKHUTDINOV, Ruslan; MNIH, Andriy; HINTON, Geoffrey. **Restricted Boltzmann Machines for Collaborative Filtering**. Disponível em: <https://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf>. Acesso em 07 out. de 2022

GRČAR, Miha; FORTUNA, Blaž; MLADENIČ, Dunja; GROBELNIK, Marko. kNN versus **SVM in the Collaborative Filtering Framework**. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.370.5678&rep=rep1&type=pdf>. Acesso em 07 out. de 2022

PATEREK, Arkadiusz. **Improving regularized singular value decomposition for collaborative filtering**. Disponível em: <https://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings/Regular-Paterek.pdf>. Acesso em 07 out. de 2022

DESROSIERS, Christian; KARYPIS, George. **A Comprehensive Survey of Neighborhood-based Recommendation Methods**. Disponível em: <https://www.inf.unibz.it/~ricci/ISR/papers/handbook-neighbor.pdf>. Acesso em 08 out. de 2022

HERLOCKER, Jonathan; KONSTAN, Joseph A.; TERVEEN, Loren G.; RIEDL, John T. **Evaluating Collaborative Filtering Recommender Systems**. Disponível em: <https://grouplens.org/site-content/uploads/evaluating-TOIS-20041.pdf>. Acesso em 09 out. de 2022

WANG, Jun; VRIES, Arjen P.; REINDERS, Marcel J. T. **Unifying user-based and item-based collaborative filtering approaches by similarity fusion**. Disponível em: <https://dl.acm.org/doi/10.1145/1148170.1148257>. Acesso em 09 out. de 2022.

GOUDAR, R. M.; BARVE, Sunita. **Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System**. Disponível em: <https://www.academia.edu/download/59762468/10.1.1.695.642820190617-91457-z4s1rf.pdf>. Acesso em 16 out. de 2022

SÁNCHEZ, José Luis. **Improving Collaborative Filtering Based Recommender Systems Using Pareto Dominance**. Disponível em: <https://oa.upm.es/23087/1/JOSE_LUIS_SANCHEZ_SANCHEZ.pdf>. Acesso em 16 out. de 2022

SALEHI, Mojtaba; POURZAFERANI, Mohammad; RAZAVI, Amir Hossein. **Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model**. Disponível em: <https://www.researchgate.net/publication/257498496_Hybrid_attribute-based_recommender_system_for_learning_material_using_genetic_algorithm_and_a_multidimensional_information_model>. Acesso em 16 out. de 2022

AI-SHAMRI, Mohammad Yahya; BHARADWAJ, Kamal K. **Fuzzy-genetic approach to recommender systems based on a novel hybrid user model**. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S095741740700351X>. Acesso em 16 out. de 2022

REN, Lei; HE, Liang; Gu, Junzhong; XIA, Weiwei; WU, Faqing. **A Hybrid Recommender Approach Based on Widrow-Hoff Learning**. Disponível em: <https://ieeexplore.ieee.org/document/4734054>. Acesso em 16 out. de 2022

CAMPOS, Luis M.; FERNÁNDEZ-LUNA, Juan M.; HUETE, Juan F.; RUEDA-MORALES, Miguel A. **Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks**. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888613X10000460>. Acesso em 16 out. de 2022

SHINDE, Subhash K.; KURKARNI, Uday. **Hybrid personalized recommender system using centering-bunching based clustering algorithm**. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0957417411011316>. Acesso em 16 out. de 2022

MANEEROJ, Saranya; TAKASU, Atsuhiro. **Hybrid Recommender System Using Latent Features**. Disponível em: <https://ieeexplore.ieee.org/document/5136724>. Acesso em 16 out. de 2022

KUNAVER, Matevž; POŽRL, Tomaž. **Diversity in recommender systems – A survey**. Disponível em: <https://papers-gamma.link/static/memory/pdfs/153-Kunaver_Diversity_in_Recommender_Systems_2017.pdf>. Acesso em 16 out. de 2022

BURKE, Robin. **Hybrid Recommender Systems: Survey and Experiments**. Disponível em: <https://link.springer.com/article/10.1023/A:1021240730564>. Acesso em 16 out. de 2022

GOOD, Nathaniel; SCHAFER, J. Bem; KONSTAN, Joseph A.; BORCHERS, Al; SARWAN, Badrul Sarwan; HERLOCKER, Jon; RIEDL, John. **Combining collaborative filtering with personal agents for better recommendations**. Disponível em: <https://aaai.org/Papers/AAAI/1999/AAAI99-063.pdf>. Acesso em 16 out. de 2022

HUANG, Zan; CHEN, Hsinchun; ZENG, Daniel. **Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering**. Disponível em: <https://dl.acm.org/doi/abs/10.1145/963770.963775>. Acesso em 16 out. de 2022

SCHEIN, Andrew I.; POPESCUL, Alexandrin; UNGAR, Lyle H.; PENNOCK, David M. **Methods and metrics for cold-start recommendations**. Disponível em: <https://dl.acm.org/doi/abs/10.1145/564376.564421>. Acesso em 16 out. de 2022

ZHENG, Yong; AGNANI, Mayur; SINGH, Mili. **Identifying Grey Sheep Users By The Distribution of User Similarities In Collaborative Filtering**. Disponível em: <https://dl.acm.org/doi/abs/10.1145/3125649.3125651>. Acesso em 22 out. de 2022

FAZZIKI, Abdellah EL; AISSAOUI, Ouafae El; ALAMI, Yasser El Madani El; ALLIOUI, Yaussouf EL; BENBRAHIM, Mohammed. **A new collaborative approach to solve the gray-sheep users problem in recommender systems**. Disponível em: <https://ieeexplore.ieee.org/document/8942256>. Acesso em 22 out. de 2022

GARCÍA, Salvador; RAMÍREZ-GALLEGO; LUENGO, Julián; BENÍTEZ, José Manuel; HERRERA, Francisco. **Big data preprocessing: methods and prospects**. Disponível em: <https://bdataanalytics.biomedcentral.com/articles/10.1186/s41044-016-0014-0>. Acesso em 02 nov. de 2022

NYUYTIYMBIY, Kizito. **Parameters and Hyperparameters in Machine Learning and Deep Learning**. Disponível em: <https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac>. Acesso em 02 nov. de 2022

SARWAR, Badrul; KARYPIS, George; KONSTAN, Joseph; RIEDL, John. **Application of Dimensionality Reduction in Recommender System - A Case Study**. Disponível em: <https://apps.dtic.mil/sti/citations/ADA439541>. Acesso em 02 nov. de 2022

VELLIANGIRI, S.; ALAGUMUTHUKRISHNAN, S.; JOSEPH, S. Iwin Thankumar. **A Review of Dimensionality Reduction Techniques for Efficient Computation**. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050920300879#:~:text=Dimensionality%20Reduction%20(DR)%20is%20the,feature%20selection%20and%20extraction%20method.>. Acesso em 03 nov. de 2022

O'MAHONY, Michael P.; HURLEY, Neil J.; SILVESTRE, Guénolé C.M. **Detecting noise in recommender system databases**. Disponível em: <https://dl.acm.org/doi/10.1145/1111449.1111477>. Acesso em 03 nov. de 2022

AMATRIAIN, Xavier; PUJOL, Josep M.; OLIVER, Nuria. **I like it... I like it not: Evaluating User Ratings Noise in Recommender Systems**. Disponível em: <https://www.nuriaoliver.com/recsys/LikeIt_umap09.pdf>. Acesso em 03 nov. de 2022

AMATRIAIN, Xavier; PUJOL, Josep M.; OLIVER, Nuria; TINTAREV, Nava. **Rate it Again: Increasing Recommendation Accuracy by User re-Rating**. Disponível em: <https://amatriain.net/pubs/xamatriain_Recsys09.pdf>. Acesso em 03 nov. de 2022

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. Disponível em: <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>. Acesso em 04 nov. de 2022

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **An Introduction to Information Retrieval**. Disponível em: <https://nlp.stanford.edu/IR-book/information-retrieval-book.html>. Acesso em 05 nov. de 2022

COVER, T. M.; HART, P. E. **Nearest Neighbor Pattern Classification**. Disponível em: <https://isl.stanford.edu/~cover/papers/transIT/0021cove.pdf>. Acesso em 05 nov. de 2022

KIZILATEŞ, Gözde; NURIYEVA, Fidan. **On the Nearest Neighbor Algorithms for the Traveling Salesman Problem**. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-319-00951-3_11>. Acesso em 05 nov. de 2022

NILSSON, Nils J.; SEJNOWSKI, Terrence J.; WHITE, Halbert. **The Mathematical Foundations of LEARNING MACHINES**. Disponível em: <https://papers.cnl.salk.edu/PDFs/Introduction%201990-3222.pdf>. Acesso em 05 nov. de 2022

QUINLAN, J.R. **Induction of Decision Trees**. Disponível em: <https://link.springer.com/article/10.1007/BF00116251>. Acesso em 05 nov. de 2022

FRIEDMAN, Nir; GEIGER, Dan; GOLDSZMIDT, Moises. Bayesian Network Classifiers. Disponível em: <https://link.springer.com/article/10.1023/A:1007465528199>. Acesso em 06 nov. de 2022

CRISTIANINI, Nello; SHAWE-TAYLOR, John. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Disponível em: <https://www.cambridge.org/core/books/an-introduction-to-support-vector-machines-and-other-kernelbased-learning-methods/A6A6F4084056A4B23F88648DDBFDD6FC>. Acesso em 06 nov. de 2022

SUN, Yueming; ZHANG, Yi; CHEN, Yunfei; JIN, Roger. **Conversational Recommendation System with Unsupervised Learning**. Disponível em: <https://dl.acm.org/doi/abs/10.1145/2959100.2959114>. Acesso em 10 nov. de 2022

XIE, Jin; ZHU, Fuxi; HUANG, Minxue; XIONG, Naixue; HUANG, Sheng; XIONG, Wei. **Unsupervised Learning of Paragraph Embeddings for Context-Aware Recommendation**. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8676110>. Acesso em 10 nov. de 2022

BAKSHI, Sagarika; JAGADEV, Alok Kumar; DEHURI, Satchidananda; WANG, Gi-Nam. **Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization**. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S1568494613003529>. Acesso em 10 de nov. de 2022

GHAHRAMANI, Zoubin. **Unsupervised Learning**. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-540-28650-9_5>. Acesso em 10 nov. de 2022

SERAJ, Raihan; ISLAM, Syed Mohammed Shamsul. **The k-means Algorithm: A Comprehensive Survey and Performance Evaluation**. Disponível em: <https://www.mdpi.com/2079-9292/9/8/1295>. Acesso em 10 nov. de 2022

SINAGA, Kristina P.; YANG, Miin-Shen. **Unsupervised K-Means Clustering Algorithm**. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9072123>. Acesso em 10 nov. de 2022

ANDONI, Alexandr; INDYK, Piotr. **Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions**. Disponível em: <https://web.mit.edu/andoni/www/papers/cSquared.pdf>. Acesso em 10 nov. de 2022

GIL, Antonio Carlos. Métodos e técnicas de pesquisa social. São Paulo: Atlas, 1999

GIL, Antonio Carlos. Como Elaborar Projetos de Pesquisa. São Paulo: Atlas, 1996.

SILVA, E. L; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. Disponível em: <https://tccbiblio.paginas.ufsc.br/files/2010/09/024_Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes1.pdf>. Acesso em 12 nov. de 2022

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar. **Metodologia do trabalho científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. Disponível em: <https://aedmoodle.ufpa.br/pluginfile.php/291348/mod_resource/content/3/2.1-E-book-Metodologia-do-Trabalho-Cientifico-2.pdf>. Acesso em 12 nov. de 2022

WITT, Graham. **Writing Effective Business Rules**. Disponível: <https://books.google.com.br/books?hl=pt-BR&lr=&id=jWTeHxQJJ8YC&oi=fnd&pg=PP1&dq=write+business+rules&ots=XbnMWTTuOO&sig=kO6Zy5E5lsNq3tww9yzqi5CvWxY#v=onepage&q=write%20business%20rules&f=false>. Acesso em 10 de fev. de 2023

ROSENBERG, Doug; SCOTT, Kendall. **Applying Use Case Driven Object Modeling with UML: An Annotated E-Commerce Example**. Disponível em: <https://pja.mykhi.org/0sem/MAS/books/Addison.Wesley.Applying.Use.Case.Driven.Object.Modeling.pdf>. Acesso em 10 de fev. de 2023.

SCHMID, Klaus. **International Workshop on Requirements Engineering for Product Lines**. Disponível em: <https://www.researchgate.net/publication/239541304_International_Workshop_on_Requirements_Engineering_for_Product_Lines>. Acesso em 10 de fevereiro de 2023.

MAIA, José Anízio Pantoja. **Construindo Softwares com Qualidade e Rapidez Usando ICONIX**. Disponível em: <https://docplayer.com.br/1630968-Construindo-softwares-com-qualidade-e-rapidez-usando-iconix.html>. Acesso em 13 de fev. de 2023.

EVANS, Josh. **Mastering Chaos – A Netflix Guide to Microservices, QCon San Francisco International Software Conference, Nov. 2016**. Disponível em: <https://www.youtube.com/watch?v=CZ3wIuvmHeM&ab_channel=InfoQ>. Acesso em 20 de março de 2023.

GO. **Documentation**. Disponível em: <https://go.dev/doc>. Acesso em 20 de março de 2023

MONGODB. “What is MongoDB”. Disponível: <https://www.mongodb.com/what-is-mongodb>. Acesso em 20 de março de 2023.

TYPESCRIPT. **The TypeScript Handbook**. Disponíve em: <https://www.typescriptlang.org/docs/handbook/intro.html>. Acesso em 20 de março de 2023.

NODEJS. About Node.js. Disponível em: <https://nodejs.org/en/about>. Acesso em 20 de março de 2023.

POSTGRES. “New to PostgreSQL?”. Disponível em: <https://www.postgresql.org/>. Acesso em 20 de março de 2023.

DEVELOPER MOZILLA. **JavaScript**. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript#reference>. Acesso em 21 de março de 2023.

NEXT.JS. **Docs**. Disponível em: <https://nextjs.org/docs>. Acesso em 21 de março de 2023.

PYTHON. **What is Python? Executive Summary**. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em 21 de março de 2023.

PANDAS. **Pandas**. Disponível em: <https://pandas.pydata.org/>. Acesso em 21 de março de 2023.

NUMPY. “What is Numpy?”. Disponível em: <https://numpy.org/doc/stable/user/whatisnumpy.html>. Acesso em 21 de março de 2023.

SURPRISE. Surprise documentation. Disponível em: <https://surprise.readthedocs.io/en/stable/index.html>. Acesso em 21 de março de 2023.
GIT. **About Git**. Disponível em: <https://git-scm.com/>. Acesso em 22 de março de 2023.

GITHUB. **Hello World**. Disponível em: <https://docs.github.com/en/get-started/quickstart/hello-world>. Acesso em 22 de março de 2023.

VISUAL STUDIO CODE. **“Why did we build Visual Studio Code?”**. Disponível em: <https://code.visualstudio.com/docs/editor/whyvscode>. Acesso em 22 de março de 2023.

UBUNTU. **The story of Ubuntu**. Disponível em: <https://ubuntu.com/about>. Acesso em 22 de março de 2023.

WSL. Issues no GitHub. Disponível em: <https://github.com/Microsoft/WSL/issues>. Acesso em 22 de março de 2023.

AMER, Ali A.; ABDALLA, Hassan I.; NGUYEN, Loc. **Enhancing recommendation systems performance using highly-effective similarity measures**. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0950705121001052>. Acesso em 23 de março de 2023.

TINGLEY, Martin; ZHENG, Wenjing; EJDEMYR, Simon; LANE, Stephanie; MCFARLAND, Colin. **Decision Making at Netflix**. Disponível em: <https://netflixtechblog.com/decision-making-at-netflix-33065fa06481>. Acesso em 23 de março de 2023.

TINGLEY, Martin; ZHENG, Wenjing; EJDEMYR, Simon; LANE, Stephanie; MCFARLAND, Colin. **What is A/B Test?**. Disponível em: <https://netflixtechblog.com/what-is-an-a-b-test-b08cc1b57962>. Acesso em 23 de março de 2023.

AMATRIAIN, Xavier; BASILICO, Justin. **Netflix Recommendations: Beyond the 5 stars (Part 2).** Disponível em: <https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5>. Acesso em 23 de março de 2023.

LIN, Chu-Hsing; CHI, Hsuan. **A Novel Movie Recommendation System Based on Collaborative Filtering and Neural Networks**. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-15032-7_75>. Acesso em 25 de março de 2023.

PROGRAMMER, Lazy. **RECOMMENDER SYSTEMS AND DEEP LEARNING IN PYTHON**. Disponível em: <https://www.udemy.com/course/recommender-systems/?utm_source=aff-campaign&utm_medium=udemyads&LSNPUBID=a1LgFw09t88&ranMID=47907&ranEAID=a1LgFw09t88&ranSiteID=a1LgFw09t88-fhsaTOzyifePvSohem2i7A>. Acesso em 31 de março de 2023.

FIREBASE. **Make your app the best it can be**. Disponível em: <https://firebase.google.com/>. Acesso em 03 de abril de 2023.

AWS. **Cloud computing with AWS**. Disponível em: <https://aws.amazon.com/what-is-aws>. Acesso em 24 de abril de 2023.
LIGHTSAIL, Amazon. **Build applications and websites fast with low-cost, pre-configured cloud resources**. Disponível em: <https://aws.amazon.com/pt/lightsail>. Acesso em 25 de abril de 2023.

EC2, Amazon. **Secure and resizable compute capacity for virtually any workload**. Disponível em: <https://aws.amazon.com/ec2>. Acesso em 25 de abril de 2023.

CALCULATOR, AWS Pricing. Disponível em: <https://calculator.aws>. Acesso em 25 de abril de 2023.

BITNAMI. **Loved by Developers. Trusted by Ops**. Disponível em: <https://bitnami.com>. Acesso em 25 de abril de 2023.

MOZILLA, Developer. **Cross-Origin Resource Sharing (CORS)**. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. Acesso em 25 de abril de 2023.

TORREY, Lisa. **Machine Learning Applications and Trends**. Disponível em: <https://www.igi-global.com/chapter/transfer-learning/36988>. Acesso em 26 de abril de 2023.

MORAES, Gustavo Hermínio Saliti Marcondes; SILVA, Danilo Soares; BOLDRIN, Juliana; FUJIRUMA, Ana Paula Akemi Araki; ROCHA, Anna Kathleen Lopes. **Como é estar nas nuvens? Satisfação, lealdade e intenção de uso dos usuários de serviços de computação em nuvem da Netflix**. Disponível em: <https://periodicos.utfpr.edu.br/rts/article/view/9355/7419>. Acesso em 02 de maio de 2023.

FILMOW. Filmow. Disponível em: <https://filmow.com>. Acesso em 02 de maio de 2023.

GOOGLE. **Gere insights facilmente com o Google Forms**. 2021b. Disponível em: <https://www.google.com/intl/pt-BR/forms/about/>. Acesso em: 02 de maio de 2023.

MEDEIROS, Otavio Lopes. **SEGURANÇA EM APLICAÇÕES WEB**. Disponível em: <http://ric-cps.eastus2.cloudapp.azure.com/handle/123456789/271>. Acesso em 06 de maio de 2023.

HOQUE, Nazrul; BHATTACHARYYA, Dhruba K.; KALITA, Jugal K. **Botnet in DDoS Attacks: Trends and Challenges**. Disponível em:

<https://ieeexplore.ieee.org/abstract/document/7160662/authors#authors>. Acesso em 06 de maio de 2023.

SOLOMON, Robert C.; HIGGINS, Kathleen. **Lectures on Nietzsche – The Great Courses on Tape**. Disponível em: <https://www.youtube.com/playlist?list=PLdnXkNG3FrNMr_rUXKRsg2sNXiFBNASuP>. Acesso em 06 de maio de 2023.

PALMER, Annie. **Amazon says more than 300 million items sold during 'biggest' Prime Day event**. Disponível em: <https://www.cnbc.com/2022/07/14/amazon-prime-day-results-more-than-300-million-items-sold.html>. Acesso em 07 de maio de 2023.

MIRANDA, Wagner Rodrigues. **Netflix: Big Data e os algoritmos de recomendação**. Disponível em: <https://www.portalintercom.org.br/anais/sudeste2017/resumos/R58-0517-1.pdf>. Acesso em 27 de maio de 2023.