



UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”
Câmpus de São José do Rio Preto

Pedrenrique Gonçalves Guimarães

Sistema de Recomendação para Técnicas de Tolerância a Falhas em Ambientes de Nuvem

São José do Rio Preto
2019

Pedrenrique Gonçalves Guimarães

Sistema de Recomendação para Técnicas de Tolerância a Falhas em Ambientes de Nuvem

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Departamento de Ciência da Computação e Estatística, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Universidade Estadual Paulista “Júlio de Mesquita Filho” (UNESP)

Instituto de Biociências, Letras e Ciências Exatas (IBILCE)

Departamento de Ciências de Computação e Estatística

Orientador: Aleardo Manacero Junior

São José do Rio Preto, SP

2019

G963s	<p>Guimarães, Pedrenrique</p> <p>Sistema de recomendação para técnicas de tolerância a falhas em ambientes de nuvem / Pedrenrique Guimarães. -- São José do Rio Preto, 2019</p> <p>59 f. : il., tabs.</p> <p>Trabalho de conclusão de curso (-) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto</p> <p>Orientador: Aleardo Manacero Junior</p> <p>1. Ciência da computação. 2. Sistemas distribuídos. 3. Tolerância a falha. I. Título.</p>
-------	--

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Pedrenrique Gonçalves Guimarães

Sistema de Recomendação para Técnicas de Tolerância a Falhas em Ambientes de Nuvem

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Departamento de Ciência da Computação e Estatística, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Trabalho aprovado. São José do Rio Preto, SP,

Aleardo Manacero Junior

UNESP – Câmpus de São José do Rio Preto
Orientador

Renata Spolon Lobato

UNESP – Câmpus de São José do Rio Preto

Rodrigo Capobianco Guido

UNESP – Câmpus de São José do Rio Preto

São José do Rio Preto, SP

2019

*Dedico este trabalho ao meu avô Lucrécio (in memoriam),
que desde o começo de minha jornada fez questão de declarar seu absoluto apoio,
e que agora descansa na mais perfeita confusão.*

RESUMO

A computação em nuvem tem ganhado grande popularidade nos últimos anos por oferecer uma gama de serviços e recursos diversificados de forma rápida, prática e bastante acessível para o público casual e também empresarial. Tais sistemas têm como principais características a escalabilidade e alta carga de trabalho, na maioria dos casos. Entretanto, as plataformas em nuvem também estão bastante suscetíveis a diversos tipos de falhas que podem comprometer o funcionamento dos serviços e de toda sua infraestrutura, permitindo assim uma perda de dados massiva e possível interrupção de suas atividades por tempo indeterminado. Tais ambientes devem possuir uma boa tolerância a essas falhas, o que significa dizer que a equipe por trás desse ambiente deve ser capaz de identificar e resolver quaisquer falhas de forma a não deixar que estas comprometam o ambiente em questão. No contexto de tratamento de tais falhas, existem diversas técnicas que buscam tratar tais problemas, incluindo uma sistematização em classes com as falhas mais comuns. Para auxiliar administradores desses serviços, este trabalho visa expandir tal sistematização, utilizando um sistema de recomendação inteligente que consiga fazer a sugestão das técnicas corretas para o tratamento de falhas no ambiente de forma automática.

Palavras-chave: ciência da computação, tolerância a falha, sistemas distribuídos

ABSTRACT

Cloud computing is gaining popularity in recent years for offering a variety of different services and resources in a quick, practical and very accessible way for the casual public and corporations. Such systems have the main characteristics the high scalability and workflow, in most cases. However, cloud platforms are also very susceptible to many different types of faults that may compromise the workings of the services and all of its infrastructure, allowing for a massive data loss and also possibly causing an unwanted interruption of its activities for some time. These services should be fault tolerant, which means that any potential failure that may happen during running time should not affect the system, and that the team behind it should be able to identify the problems and solve it without halting the entire operation of the service. There are several techniques available to treat these problems, including a systematic classification of most common faults. To help administrators of these services, this work aims to expand this classification, using an intelligent recommender system that is able to suggest the correct techniques to treat faults automatically.

Keywords: computer science, fault-tolerant computing, distributed systems

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de infraestrutura de computação em nuvem.	14
Figura 2 – Arquitetura da computação em nuvem.	19
Figura 3 – Modelos de sistemas de recomendação	28
Figura 4 – Diagrama do funcionamento da Filtragem Colaborativa	29
Figura 5 – Diagrama do funcionamento de Recomendação baseada em conteúdo	30
Figura 6 – Exemplo de recomendação por conteúdo assistido anteriormente e filmes relevantes	34
Figura 7 – Exemplo de recomendação personalizada	34
Figura 8 – Planejamento inicial de interface da ferramenta	37
Figura 9 – Tela inicial da ferramenta	49
Figura 10 – Informações disponíveis sobre a técnica Tahoe-LAFS	50
Figura 11 – Tela inicial da ferramenta reduzida	51
Figura 12 – Tela que permite visualizar todas as técnicas cadastradas	51
Figura 13 – Sequência de figuras que permitem demonstrar a filtragem por modelos de implementação e tipo de técnica, respectivamente	52
Figura 14 – Sequência de figuras qu demonstram a filtragem por técnica utilizada, nome e gestor, respectivamente	52
Figura 15 – Sequência de figuras mostrando as diferentes recomendações dos usuários com identificador 1, 6 e 10, respectivamente	53

LISTA DE TABELAS

Tabela 1 – Comparação entre Características de Computação em Nuvem e Computação em Grade	16
Tabela 2 – Modelos de Implementação de computação em nuvem	18
Tabela 3 – Modelos de tolerância a falhas	25
Tabela 4 – Descrição dos métodos de hibridação	31
Tabela 5 – Teste com o arquivo userdata_5_10.csv	45
Tabela 6 – Teste com o arquivo userdata_5_100.csv	46
Tabela 7 – Teste com o arquivo userdata_5_1000.csv	46
Tabela 8 – Teste com o arquivo userdata_10_10.csv	46
Tabela 9 – Teste com o arquivo userdata_10_100.csv	46
Tabela 10 – Teste com o arquivo userdata_10_1000.csv	47
Tabela 11 – Teste com o arquivo userdata_5_10.csv usando matriz de fatorização não-negativa	47
Tabela 12 – Teste com o arquivo userdata_5_100.csv usando matriz de fatorização não-negativa	47
Tabela 13 – Teste com o arquivo userdata_5_1000.csv matriz de fatorização não-negativa	47
Tabela 14 – Teste com o arquivo userdata_10_10.csv matriz de fatorização não-negativa	48
Tabela 15 – Teste com o arquivo userdata_10_100.csv matriz de fatorização não-negativa	48
Tabela 16 – Teste com o arquivo userdata_10_1000.csv matriz de fatorização não-negativa	48

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Motivação	12
1.2	Objetivos	13
1.3	Organização do texto	13
2	FUNDAMENTAÇÃO TEÓRICA	14
2.1	Computação em nuvem	14
2.1.1	Características essenciais	15
2.1.2	Modelos de serviço	17
2.1.3	Modelos de implementação	17
2.1.4	Arquitetura de computação em nuvem	17
2.1.5	Aplicações na indústria	18
2.1.5.1	<i>Amazon Web Services (AWS)</i>	18
2.1.5.2	Microsoft Azure	19
2.1.5.3	Dropbox	20
2.1.5.4	Google App Engine	20
2.2	Tolerância a falhas	20
2.2.1	Conceitos básicos	21
2.2.2	Métricas para Tolerância a Falhas	22
2.2.3	Técnicas de tolerância a falhas	23
2.2.3.1	Técnicas reativas	23
2.2.3.2	Técnicas proativas	24
2.2.4	Modelos de tolerância a falhas	24
2.2.5	Injeção de falhas	24
2.3	Sistemas de recomendação	26
2.3.1	Conceitos básicos	26
2.3.2	Modelos de sistemas de recomendação	28
2.3.2.1	Filtragem Colaborativa	28
2.3.2.2	Recomendação Baseada em Conteúdo	30
2.3.2.3	Recomendação Híbrida	30
2.3.3	Aplicações	31
2.3.3.1	Redes sociais	32
2.3.3.2	<i>E-commerce</i>	33
2.3.3.3	Serviços	33
2.4	Trabalhos relacionados	34

2.5	Considerações finais	35
3	PROJETO	36
3.1	Motivação	36
3.2	Metodologia	36
3.2.1	Descrição do Sistema	36
3.2.2	Desafios	37
3.2.3	Implementação	38
3.2.3.1	Coleta de dados	38
3.2.3.2	Geração de dados	40
3.2.3.3	Armazenamento de dados	41
3.2.3.4	Sistema de Recomendação	42
3.2.3.5	Interface gráfica	42
3.3	Considerações finais	43
4	TESTES E RESULTADOS	44
4.1	Validação e Previsões	44
4.1.1	Primeiro teste	44
4.1.2	Segundo teste	46
4.1.3	Terceiro teste	47
4.2	Interface gráfica e recomendações	48
4.3	Considerações Finais	53
5	CONCLUSÃO	54
	REFERÊNCIAS	55

1 INTRODUÇÃO

A computação em nuvem é uma forma de computação que provê diversos serviços distintos por meio da Internet, de forma que o usuário em questão possa acessar os serviços de qualquer plataforma, desde que possua uma conexão com a Internet. Esse tipo de computação tem se tornado muito popular com a expansão da rede mundial de computadores pelo mundo, e com o pesado investimento em infraestrutura proporcionado pelas grandes corporações e operadoras de telecomunicações, que agora tornaram mais acessível ao usuário final o acesso a uma conexão rápida e estável, o que é um requisito fundamental para o correto funcionamento desses serviços.

Gigantes da computação como a Amazon, Google, IBM e Microsoft têm investido pesado em infraestrutura para oferecimento de serviços de computação em nuvem (TSAI; SUN; BALASOORIYA, 2010; LOHR, 2007). Outras empresas ganharam grande notoriedade na indústria por oferecerem exclusivamente serviços de computação em nuvem, como é o caso da Netflix, Hulu e Dropbox, por exemplo (BERMAN et al., 2012).

A alta demanda por tais serviços e modo de operação tem tornado cada vez mais comum o aparecimento de sistemas de computação em nuvem, e cada vez mais profissionais estão sendo treinados para serem capazes de desenvolver e manter tais tecnologias. Entretanto, não diferente de qualquer outro sistema computacional, sistemas de computação em nuvem também são bastante suscetíveis a diversos tipos de falhas e erros que podem comprometer o funcionamento do sistema como um todo. Cada tipo de falha pode ter uma origem diferente, com algum tipo de solução diferente, o que dificulta sua prevenção ou correção, dependendo do tipo de sistema utilizado. Diversos trabalhos na literatura abordam técnicas para lidar com tais falhas e previni-las, e propõem metodologias que visam classificar e identificar as melhores técnicas em casos específicos.

Este trabalho tem o intuito de acrescentar a essas metodologias uma forma de sugestão automática de técnicas de tolerância a falhas por meio de um sistema de recomendação, de forma com que a administração do sistema em questão possa focar na técnica sugerida em vez de buscar diversas soluções distintas que podem resultar em horas infrutíferas de trabalho.

1.1 Motivação

A crescente demanda por serviços baseados em nuvem tem feito o interesse pela área crescer entre os pesquisadores do âmbito da computação para lidar com a tolerância de falhas desses sistemas computacionais. Por consequência, o número de abordagens distintas

para suprir tais falhas tem se multiplicado na literatura, e a escolha de abordagens envolve uma série de variáveis que podem levar a dúvidas com relação a qual destes métodos podem ser escolhidos durante uma situação real de risco. Nem sempre a escolha é a melhor possível para uma determinada situação, e uma abordagem pode ter diversos pontos negativos em sua execução.

Levando em conta que sistemas de computação em nuvem devem permanecer operantes de forma ininterrupta, a escolha dos melhores métodos de tolerância a falhas deve ser feita de forma inteligente e deve também estar sujeito a uma rigorosa avaliação por parte da equipe de administração.

1.2 Objetivos

Com o fato de haver inúmeras técnicas distintas para lidar com a tolerância de falhas nesses sistemas, cada uma com suas vantagens e desvantagens, a escolha dessas técnicas pode ser um processo difícil na contenção de falhas.

Utilizando um sistema de recomendação dessas abordagens, os administradores desses sistemas computacionais conseguem, de forma simples, escolher a melhor abordagem para suprir a situação, eliminando a necessidade de uma consulta mais aprofundada, o que, por sua vez, representa uma grande economia de tempo e esforço da equipe de administração.

1.3 Organização do texto

Este documento é dividido em capítulos, seções e diversas subseções. No capítulo a seguir (Capítulo 2), serão abordados os conceitos básicos necessários para o entendimento do trabalho. Os temas de computação em nuvem, tolerância a falhas e sistemas de recomendação serão analisados e explicados em suas respectivas seções.

Com os conceitos devidamente estabelecidos, o capítulo seguinte (Capítulo 3), expõe as principais propostas para o projeto, com as devidas definições de implementação e detalhes necessários para tal.

No capítulo 4 são executados os testes referentes ao funcionamento do sistema como um todo, buscando validar os recursos que o sistema se propõe a fazer. E, por fim, o capítulo 5 faz a conclusão geral do que foi feito nesse trabalho, e algumas das perspectivas para uso futuro.

2 FUNDAMENTAÇÃO TEÓRICA

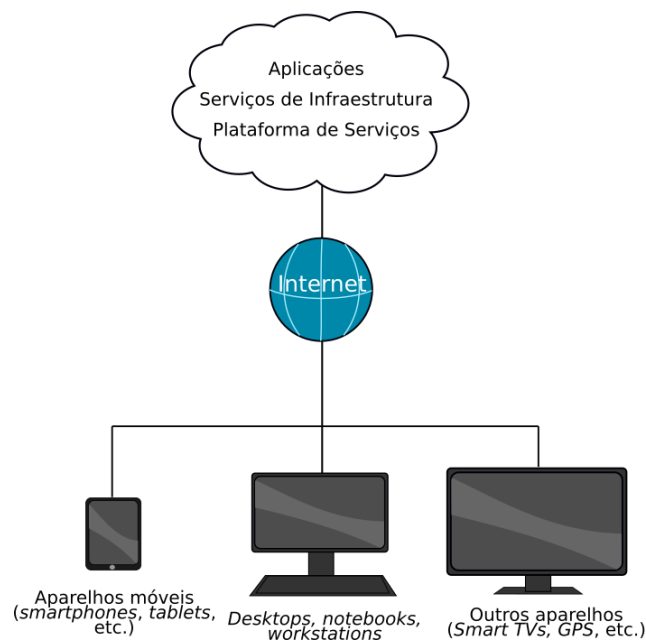
Como parte essencial para o entendimento do trabalho, este capítulo irá tratar dos principais conceitos e fundamentos necessários para a implementação do projeto de sistema de recomendação.

2.1 Computação em nuvem

O termo computação em nuvem tem diferentes significados na literatura, e alguns exemplos podem ser mencionados. Em artigo publicado em 2013, [FERNANDO; LOKE; RAHAYU](#) afirmam que consumidores veem computação em nuvem como um serviço padronizado acessível por meio de um aparelho conectado à Internet. Por outro lado, outro artigo de 2012, [MAZHELIS; TYRVÄINEN](#) alegam que profissionais de Tecnologia da Informação veem computação em nuvem como uma *pool* de recursos compartilhados acessados por ambientes computacionais. ([KHAN, 2018](#), p. 1)

De uma forma geral, pode-se representar um exemplo de infraestrutura básica da computação em nuvem como definida na Figura 1.

Figura 1 – Exemplo de infraestrutura de computação em nuvem.



(Traduzido e adaptado pelo autor ([MARSTON et al., 2011](#)))

Em 2011, o *Information Technology Laboratory* (ITL), que faz parte *National Institute of Standards and Technology* (NIST), uma agência governamental que promove a utilização de padrões em ambientes de tecnologia da informação, lançou um documento que

define a computação em nuvem como “um modelo que permite acesso ubíquo, conveniente e sob demanda para uma gama de recursos computacionais compartilhados que podem ser provisionados e liberados com o mínimo de esforço ou interação do provedor de serviço.” (MELL; GRANCE et al., 2011, p. 2) (Tradução adaptada pelo autor).

O documento também prevê cinco características essenciais, três modelos de serviço e quatro modelos de implementação que definem o ambiente de computação em nuvem, que serão descritas nas subseções seguintes (2.1.1, 2.1.2, 2.1.3).

2.1.1 Características essenciais

O documento do NIST define as seguintes características essenciais para ambientes de computação em nuvem:

- **Atendimento sob demanda:** O consumidor pode requisitar as operações do serviço em tempo real, como armazenamento de dados por exemplo, sem necessidade de interação humana com o provedor de serviços;
- **Amplo acesso à rede:** Os serviços são disponibilizados por meio do acesso à internet, utilizando mecanismos padronizados que permitam o acesso utilizando diferentes plataformas, como celulares, *notebooks*, *tablets*, *desktops* etc;
- **Pool de recursos:** Os recursos computacionais do provedor de serviços deve ser capaz de servir múltiplos consumidores ao mesmo tempo, com diferentes recursos físicos e virtuais sendo destinados dinamicamente de acordo com a demanda do consumidor;
- **Rápida elasticidade:** Os recursos computacionais podem ser fornecidos e liberados de forma elástica, em alguns casos de forma automática, escalando de forma adequada conforme a demanda. Para o consumidor, tais recursos podem parecer ilimitados, ou então podem ser adquiridos em qualquer quantidade a qualquer tempo;
- **Serviços mensuráveis:** Sistemas de nuvem controlam e otimizam automaticamente o uso de recursos utilizando recursos mensuráveis (que geralmente são alcançados utilizando um serviço de “pagar por uso”). O uso de recursos pode ser monitorado, controlado, e reportado, promovendo transparência tanto para o consumidor quanto para o provedor de serviços.

GONG et al. (2010) faz uma relação entre as características de computação em nuvem e computação em grade, um modelo computacional paralela que é baseada em alta performance computacional. Tal comparação está reproduzida na Tabela 1, de forma traduzida e adaptada.

Tabela 1 – Comparação entre Características de Computação em Nuvem e Computação em Grade

Característica	Computação em nuvem	Computação em grade
Orientada a Serviço	Sim	Sim
Acoplamento Fraco	Sim	Metade
Forte Tolerância a Falhas	Sim	Metade
Modelo de negócios	Sim	Não
Facilmente utilizável	Sim	Metade
Baseado em TCP/IP	Sim	Metade
Alta segurança	Metade	Metade
Virtualização	Sim	Metade

Na Tabela 1, os parâmetros “sim” e “não” referem-se à presença dessas características especiais nos sistemas, enquanto o parâmetro “metade” refere-se a uma presença parcial da característica no sistema computacional. Tal comparação é útil para diferenciar o fato de que Computação em Nuvem não necessariamente significa Computação de Alta Performance, embora muitas vezes estejam associados, e mostra claramente que existem diferenças claras entre os dois modelos computacionais.

Dessa forma, pode-se ver que existem características específicas do modelo de computação em nuvem que são bastante importantes para seu funcionamento.

- **Orientado a serviço:** Serviços de computação em nuvem estão disponíveis para uso sob demanda, como descrito nas características essenciais;
- **Acoplamento fraco:** Ambientes de computação em nuvem utilizam-se de virtualização ou outras tecnologias que separam o lógico do físico. Na seção 2.1.4 isso será um pouco melhor explorado.
- **Forte tolerância a falhas:** Aspecto essencial deste trabalho, existem diversas publicações que mostram a importância da tolerância a falhas na computação em nuvem (JHAWAR; PIURI; SANTAMBROGIO, 2012; ATAALLAH; NASSAR; HEMAYED, 2015; AMIN; SINGH; SETHI, 2015; CHERAGHLOU; KHADEM-ZADEH; HAGHPARAST, 2016);
- **Modelo de negócios:** Característica chave que distingue computação em nuvem de computação em grade (GONG et al., 2010, p. 278). A computação em nuvem é fortemente suportada por grandes companhias na área de Tecnologia de Informação, e todas possuem lucro como principal objetivo do investimento nessas plataformas. Grande parte dos sistemas utiliza alguma forma de pagamento por uso de suas plataformas, e isso os torna sistemas bastante rentáveis para suas respectivas companhias.

2.1.2 Modelos de serviço

Os modelos de serviço definidos no documento estão descritos a seguir, com tradução e adaptação pelo autor:

- ***Software as a Service (SaaS)***: O recurso computacional é disponibilizado para o consumidor como uma aplicação executada em infraestrutura na nuvem. Essa aplicação pode ser acessada de diversos dispositivos, podendo-se utilizar um navegador de internet, ou até mesmo uma aplicação própria. O consumidor, neste caso, não tem nenhum controle sobre os servidores, sistemas operacionais, armazenamento ou aplicações que fazem parte da infraestrutura da rede, limitando-se apenas àquilo que lhe foi designado como serviço disponível;
- ***Platform as a Service (PaaS)***: O recurso disponibilizado ao consumidor é para a implementação de um sistema de computação em nuvem, podendo assim criar sua própria aplicação ou serviço utilizando a infraestrutura existente. Neste caso, o consumidor também não possui controle sobre tal infraestrutura, mas pode implementar aplicações e configurações no ambiente hospedeiro;
- ***Infrastructure as a Service (IaaS)***: O recurso computacional disponibilizado é prover processamento, armazenamento, rede e outros recursos fundamentais onde o consumidor pode implementar e executar sistemas operacionais e aplicações a seu gosto, quando disponíveis. O consumidor também não controla a infraestrutura, mas já tem controle sobre os sistemas operacionais, armazenamento e aplicações implementadas.

2.1.3 Modelos de implementação

Os modelos de implementação mais comuns definidos pelo NIST são os modelos de nuvem privada, nuvem comunitária, nuvem pública e nuvem híbrida. As características de cada um desses serviços está descrita na Tabela 2.

2.1.4 Arquitetura de computação em nuvem

De uma forma geral, podemos dividir a arquitetura de um sistema de computação em nuvem em quatro camadas (ZHANG; CHENG; BOUTABA, 2010), como também visto na Figura 2:

- **Camada de *hardware***: Responsável por gerenciar os recursos físicos da nuvem, como servidores, roteadores, *switches*, energia e sistemas de resfriamento. Trata-se dos próprios *data centers*.

Tabela 2 – Modelos de Implementação de computação em nuvem

Modelos de Implementação	
Nuvem privada	A infraestrutura em nuvem provisionada é de uso exclusivo de uma organização, compondo-se de múltiplos consumidores.
Nuvem comunitária	A infraestrutura em nuvem provisionada é de uso exclusivo de uma comunidade específica de consumidores de organizações com interesses em comum.
Nuvem pública	A infraestrutura em nuvem provisionada é aberta para o público em geral.
Nuvem híbrida	A infraestrutura em nuvem disponibilizada é uma composição de duas ou mais infraestruturas de nuvem descritas anteriormente, que continuam operando de forma independente, mas que se interconectam por uma tecnologia padronizada que permite a portabilidade de aplicações e dados.

(Traduzido e adaptado pelo autor)

- **Camada de infraestrutura:** Trata-se da camada de virtualização, que cria a *pool* de armazenamento e recursos computacionais, particionando os recursos físicos usando tecnologias de virtualização.
- **Camada de plataforma:** Consiste no sistema operacional e nos *frameworks* das aplicações. Essa camada facilita a implantação das aplicações executadas.
- **Camada de aplicação:** Consiste nas próprias aplicações executadas em nuvem. Nesses ambientes, as aplicações em nuvem podem ser escaláveis para obter melhor desempenho, acessibilidade e menor custo.

Tal arquitetura torna os serviços de computação em nuvem mais modulares quando comparados a tradicionais *server farms*, por permitir que as camadas funcionem como módulos fracamente conectados entre si, permitindo que esses se expandam e evoluam de forma independente (ZHANG; CHENG; BOUTABA, 2010, p. 9).

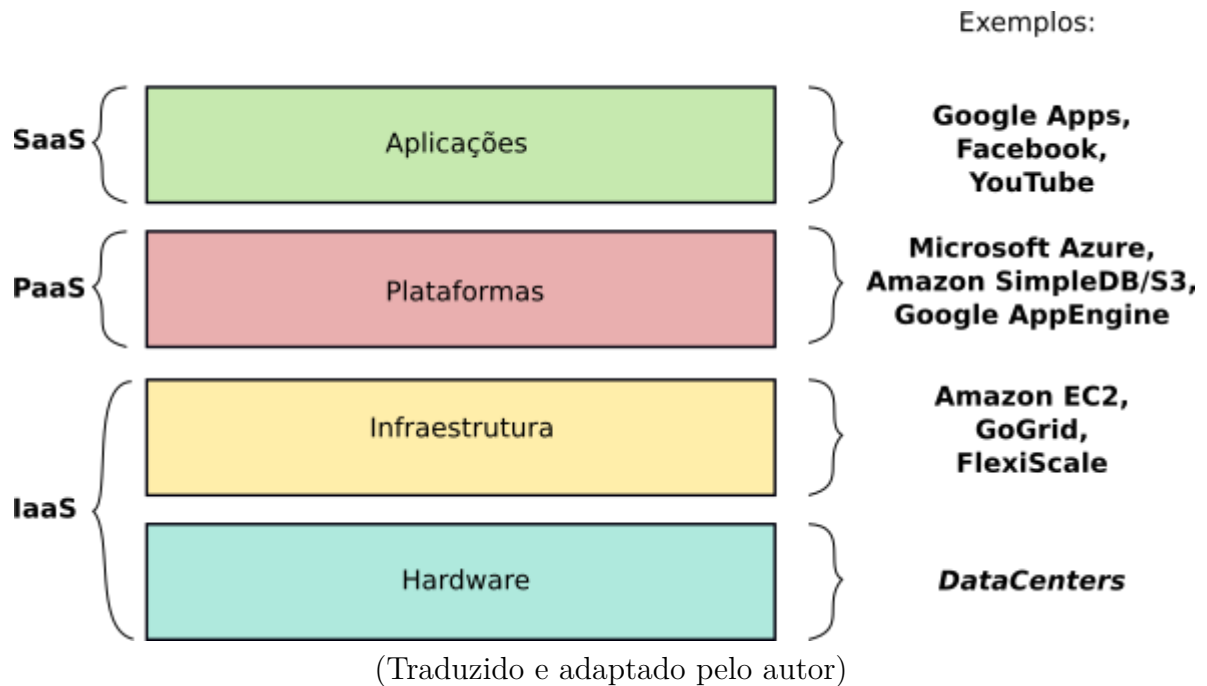
2.1.5 Aplicações na indústria

Existem inúmeras aplicações de computação em nuvem na indústria, que variam de acordo com a natureza da indústria e sua carga de trabalho (KHAN, 2018, p. 10). Nesta seção, serão apresentados alguns serviços mais conhecidos e relevantes de computação em nuvem.

2.1.5.1 Amazon Web Services (AWS)

Amazon Web Services (AWS) (AMAZON.COM, INC., 2019b) é um conjunto de serviços de nuvem, que fornecem computação baseada em nuvem, armazenamento e outras

Figura 2 – Arquitetura da computação em nuvem.



aplicações que atendem tanto o aspecto corporativo quanto a usuários individuais. Nesse ambiente, usuários podem implementar aplicações e serviços sob demanda, pagando de acordo com o uso. (ZHANG; CHENG; BOUTABA, 2010, p. 13)

Neste contexto, a Amazon disponibiliza também o serviço conhecido como **Amazon Elastic Compute Cloud**, também chamado de **Amazon EC2**. Segundo o próprio serviço: “O Amazon Elastic Compute Cloud (Amazon EC2) é um web service que disponibiliza capacidade computacional segura e redimensionável na nuvem. Ele foi projetado para facilitar a computação em nuvem na escala da web para os desenvolvedores.” (AMAZON.COM, INC., 2019a). Tal plataforma permite que o usuário administre servidores e *data centers* utilizando APIs ou ferramentas e utilitários disponíveis. As instâncias da EC2 são máquinas virtuais sendo executadas dentro da engine de virtualização Xen (XENSOURCE, INC., 2019; ZHANG; CHENG; BOUTABA, 2010).

A estrutura da AWS também comporta serviços como o Amazon Virtual Private Cloud (VPC), que se comporta como uma rede privada virtual (VPN), utilizando os recursos da AWS para permitir uma segurança adicional para o usuário final, com serviços de firewall, detecção de intrusão, entre outros serviços.

2.1.5.2 Microsoft Azure

A plataforma Microsoft Azure, antigamente chamada de Microsoft Windows Azure, consiste em uma plataforma da própria Microsoft para executar aplicações e armazenar dados na nuvem (CHAPPELL et al., 2009).

Os serviços oferecidos são diversos: armazenamento de dados em nuvem, execução de aplicações, gerenciamento de contêineres, APIs de detecção facial, ferramentas de testes para desenvolvedores, entre outras aplicações ([MICROSOFT, INC., 2019](#)).

2.1.5.3 Dropbox

Criado em 2007, o serviço conhecido como Dropbox ([DROPBOX, INC., 2019a](#)) ocupa um nicho específico no ramo de computação em nuvem. Seu principal modelo de negócio é o armazenamento de dados em nuvem, oferecendo serviços para corporações e para usuários finais. Ele oferece aos usuários um serviço gratuito de armazenamento de até 2 GB por usuário. Esse montante pode ser expandido em até 3 TB ou mais mediante pagamento de planos específicos, que oferecem recursos adicionais como sincronização automática com dispositivos conectados, busca em texto, recuperação de arquivos desejados e suporte dedicado ([DROPBOX, INC., 2019b](#)).

2.1.5.4 Google App Engine

A Google App Engine é mais um dos serviços de computação em nuvem da Google. Consiste em uma plataforma para criação de aplicativos de forma escalonável, totalmente gerenciada e sem servidor. Na plataforma pode-se utilizar linguagens e frameworks populares como Java, PHP, Node.js, Python, C#, .Net, Ruby e Go, ou o usuário pode também utilizar seu próprio framework sobre a plataforma ([GOOGLE, INC., 2019](#)).

O serviço conta também com ferramentas para contêineres, controle de versão, segurança com certificados gerenciados SSL/TLS, ferramentas para depuração, dentre tantas outras opções para desenvolvedores.

2.2 Tolerância a falhas

O uso de sistemas computacionais é ubíquo no mundo moderno, estendendo-se a aplicações em sistemas de defesa, sistemas aéreo, controle de tráfego aéreo, sistemas bancários, jogos virtuais, dentre tantos outros. A variedade de aplicações e a importância crescente desse setor trouxe à tona a confiabilidade de sistemas computacionais para as mais diversas aplicações ([ANDERSON; LEE, 1981](#)).

Um exemplo de preocupação com o funcionamento adequado desses sistemas ocorreu no fim da década de 1990, quando percebeu-se o modo ineficiente de armazenamento e formatação de datas de calendários em sistemas computacionais da época, e que colocaria em risco a correta execução de milhares de serviços dependentes desses sistemas, podendo também ocasionar uma pane geral de alguns sistemas de alto risco. Tal episódio é conhecido na literatura como “Problema do Ano 2000”, muitas vezes estilizado como “Y2K *problem*”, ou popularmente chamado de “*Bug do Milênio*”. A preocupação com uma pane em massa

de praticamente todos os sistemas computacionais levou a criação de uma série de medidas para garantir a correção desse problema nos sistemas existentes e prevenir a ocorrência de problemas similares em futuros sistemas desenvolvidos (PETERSEN; WHEATLEY; KELLNER-ROGERS, 1998).

Este clássico exemplo de prevenção de falhas mostra uma fraqueza de alguns sistemas computacionais em lidar com falhas previstas anteriormente. Caso tais correções e recomendações não fossem consideradas, especula-se que as consequências poderiam ser catastróficas para os mais diversos setores que dependiam desses sistemas computacionais (SMITH; MÜLLER; TILLEY, 1997). Assim sendo, deve-se levar em consideração que existem problemas aos quais sistemas computacionais estão sujeitos, mas que podem ser tratados adequadamente em tempo de execução, evitando ao máximo comprometer o funcionamento desse sistema.

2.2.1 Conceitos básicos

Assim como os sistemas mencionados anteriormente estavam sujeitos a falhas causadas por uma falta de planejamento dos desenvolvedores, sistemas computacionais modernos também estão sujeitos a inúmeros tipos de falhas, que podem fugir completamente ao controle tanto de administradores do sistema quanto aos desenvolvedores ou fabricantes de tal produto.

Uma falha no sistema ocorre quando o serviço oferecido se desvia do serviço especificado, na qual o serviço especificado significa uma descrição coerente do que é esperado desse serviço. Nesse contexto, pode-se afirmar que ocorreu um erro, decorrente de uma falha. Isto é, o erro é uma manifestação de uma falha no sistema, e a falha é a manifestação do erro no serviço (LAPRIE, 1985).

O conceito de tolerância a falhas em computação é bastante antigo e se aplica tanto no âmbito do fabricante de *hardware* quanto na área de desenvolvimento de *software* (RANDELL, 1975). Para ambos, é importante garantir a confiabilidade do sistema para qualquer um que venha a utilizá-lo posteriormente. Isso quer dizer que o sistema deve garantir que, mesmo após a ocorrência de algum tipo de falha, não ocorra interrupções no serviço (BALA; CHANA, 2012). Assim como visto na seção 2.1.1, tolerância a falhas é uma das principais características de um ambiente de computação em nuvem, e é objeto de estudo de diversos trabalhos na área.

A tolerância a falhas é pautada por processamento de erros, que se baseia em duas fases (LAPRIE, 1985):

1. **Processamento de erro efetivo:** essa fase busca trazer o erro de volta ao estado latente, se possível a um estado anterior à ocorrência da falha;

2. **Processamento de erro latente:** tenta garantir que o erro não se torne efetivo novamente.

Entre os benefícios de se implementar tolerância a falhas na computação em nuvem estão inclusos: recuperação em situação de falha, baixo custo e melhora na métrica de desempenho (PATRA; SINGH; SINGH, 2013). No decorrer desta seção serão apresentadas algumas formas de como isso pode ser feito em um sistema de computação em nuvem.

2.2.2 Métricas para Tolerância a Falhas

AMIN; SINGH; SETHI (2015) e PATRA; SINGH; SINGH (2013) definem métricas para tolerância a falhas, que estão devidamente reproduzidas neste documento de forma traduzida e adaptada a seguir:

- **Rendimento:** Define o número de tarefas cuja execução foi concluída. O rendimento de um sistema deve ser alto;
- **Tempo de resposta:** O tempo que leva para um algoritmo responder deve ser minimizado;
- **Escalabilidade:** O número de nós do sistema não deve afetar a capacidade de tolerância a falhas do algoritmo;
- **Desempenho:** Parâmetro que mede a efetividade do sistema. O desempenho do sistema deve ser melhorado lentamente;
- **Disponibilidade:** Sua disponibilidade é diretamente proporcional a sua confiabilidade. Trata-se da possibilidade de que um item está funcional durante um determinado período de tempo;
- **Usabilidade:** Dá a dimensão do quanto um produto pode ser utilizado por um usuário para atingir seus objetivos com efetividade, eficiência e satisfação;
- **Confiabilidade:** Esse aspecto almeja dar o resultado correto ou aceitável durante um determinado período de tempo;
- **Sobrecarga associada:** Trata-se da sobrecarga resultante da implementação de um algoritmo. Sobrecargas podem ser impostas por conta de movimentos de tarefas entre processos ou uma comunicação entre processadores. Para a eficiência da técnica de tolerância a falhas, sobrecargas devem ser minimizadas;
- **Custo-benefício:** O custo neste quesito é apenas definido como custo de monitoramento.

2.2.3 Técnicas de tolerância a falhas

Existem diversas técnicas de tolerância a falhas descritas na literatura (CHERAGHLOU; KHADEM-ZADEH; HAGHPARAST, 2016; ATAALLAH; NASSAR; HEMAYED, 2015; JHAWAR; PIURI; SANTAMBROGIO, 2012). Essas técnicas podem ser classificadas em reativas ou proativas, no ambiente de computação em nuvem (BALA; CHANA, 2012). A seguir será apresentada uma visão geral das técnicas de tolerância a falhas no ambiente de computação em nuvem, sem levar em consideração plataformas específicas de computação em nuvem.

2.2.3.1 Técnicas reativas

Técnicas reativas de tolerância a falhas são utilizadas para reduzir o impacto dessas falhas em uma situação em que elas de fato ocorreram (BALA; CHANA, 2012; PATRA; SINGH; SINGH, 2013). Assim sendo, segue uma descrição dessas técnicas:

- **Check pointing/Restart:** A tarefa em falha é reiniciada a partir do ponto de verificação mais recente. É uma técnica eficiente para grandes aplicações;
- **Replicação:** Executa várias réplicas da tarefa em diferentes recursos até que toda a tarefa replicada seja executada sem interrupções;
- **Migração de trabalho:** Mediante a ocorrência de falha, o trabalho é migrado para uma nova máquina;
- **S-Guard:** É baseado em recuperação de *rollback*, ou seja, desfazer e/ou descartar todas as alterações feitas durante determinada transação.
- **Reexecução:** É a técnica mais simples: reexecuta a tarefa no mesmo recurso de nuvem;
- **Ressubmissão de tarefa:** A tarefa em falha é submetida novamente para a mesma máquina na qual estava operando ou a uma máquina diferente;
- **Manipulação de exceções definidas pelo usuário:** Nesse caso o usuário define ações específicas da falha de tarefas para fluxos de trabalhos;
- **Resgate de fluxo de trabalho:** Permite que o sistema mantenha seu funcionamento após uma falha em qualquer tarefa até não ser mais capaz de prosseguir sem atender a tarefa falhada;
- **Reconfiguração:** Nesse procedimento é eliminado o componente em falha.

2.2.3.2 Técnicas proativas

No caso de técnicas proativas, elas são responsáveis por prever as falhas e substituir componentes suspeitos por outros componentes de trabalho. Nesse caso temos (BALA; CHANA, 2012; PATRA; SINGH; SINGH, 2013):

- **Rejuvenescimento de *software*:** É uma técnica que faz com que o sistema planeje reinícios periódicos. Reinicia o sistema em um estado limpo, começando em um novo estado toda vez que ocorre sua reinicialização.
- **Autocura:** Uma tarefa grande pode ser dividida em partes. Essas multiplicações são feitas para melhorar o desempenho. Quando várias instâncias da aplicação são executadas em diferentes máquinas virtuais, isso automaticamente trata falhas de instâncias específicas.
- **Migração preemptiva:** Nesta técnica a aplicação é constantemente monitorada e analisada. A migração antecipatória de uma tarefa depende do mecanismo de controle de retorno.

2.2.4 Modelos de tolerância a falhas

Baseados nas técnicas descritas anteriormente, vários modelos podem ser implementados (CHERAGHLOU; KHADEM-ZADEH; HAGHPARAST, 2016; PATRA; SINGH; SINGH, 2013). Na Tabela 3 estão descritos tais modelos de forma resumida e adaptada.

2.2.5 Injeção de falhas

A efetividade das técnicas de tolerância falhas implementadas pode ser testada provocando falhas no ambiente de computação em nuvem, dessa forma atestando sua confiabilidade no sistema em questão. A técnica de *Software Fault Injection*, ou Injeção de Falhas em *Software*, é utilizada com o intuito de antecipar possíveis cenários ocasionados por erros, por meio da injeção de falhas no *hardware* e/ou *software* (NATELLA; COTRONEO; MADEIRA, 2016).

Tal metodologia pode ser aplicada de duas formas (HSUEH; TSAI; IYER, 1997):

- *Software*: o mecanismo mais atrativo por não necessitar de *hardware* caro. Permite simular falhas em *hardware* também, e também pode ter como alvo o sistema operacional, o que é muito difícil conseguir com falhas em *hardware*. Nesse caso, o injetor de falhas é inserido na própria aplicação ou colocado numa camada entre a aplicação e o sistema operacional.

Tabela 3 – Modelos de tolerância a falhas

Modelo	Características
AFTRC (Adaptive Fault Tolerance model in Real Time Cloud Computing)	Modelo para computação em nuvem em tempo real. Se aproveita da capacidade computacional e do ambiente virtual escalável de computação em nuvem para melhor implementar aplicações em tempo real. É um ambiente proativo, cujas decisões são tomadas com base na confiabilidade dos nós de processamento.
LLFT (Low Latency Fault Tolerance)	Possui um <i>middleware</i> que fornece tolerância a falhas para aplicações distribuídas executadas no ambiente de tolerância a falhas. O <i>middleware</i> funciona na base da replicação, utilizando uma replicação semi-ativa ou semi-passiva para proteger contra diversos tipos de falhas.
FTWS Fault Tolerant Workflow Scheduling	Algoritmo de escalonamento de fluxo de trabalho fornece um ambiente de tolerância a falhas usando ressubmissão de tarefas baseada na prioridade das tarefas.
FTM Fault Tolerance Manager	Busca dar ao usuário o controle do nível desejado de tolerância a falhas. É um conjunto de componentes de serviços <i>Web</i> , em que cada um possui uma função definida.
Candy	Disponibilidade baseada em componentes. É um modelo que garante alta disponibilidade do serviço em nuvem.
Vega-warden	Sistema de gerenciamento uniforme que fornece um espaço de usuário global para diferentes infraestruturas virtuais e serviços de aplicações no ambiente em nuvem.
FT-Cloud	Baseado na classificação de componentes para estrutura de aplicações em nuvem. Utiliza um algoritmo que determina automaticamente a tolerância a falhas.
Magi-Cube	Altamente confiável e uma arquitetura de baixa redundância para armazenamento em nuvem. Baseado na alta confiabilidade e alto desempenho com baixo custo.

- *Hardware*: Utiliza *hardware* adicional para a introdução de falhas no sistema alvo. São métodos bastante úteis para o estudo de características dependentes de protótipos que requerem alto tempo de resolução de ativação do *hardware* e de monitoramento ou que requerem acesso a localizações que não podem ser alcançadas por outros métodos de injeção de falhas.

Apesar de mais baratas, injeções implementadas por *software* possuem alguns contras (HSUEH; TSAI; IYER, 1997):

- Não pode injetar falhas em localizações inacessíveis ao *software*.
- A instrumentação do *software* pode interferir na carga de trabalho do sistema alvo e até mesmo modificar a estrutura do software original.
- Podem ocorrer problemas de latência, por conta do tempo de resolução do método. Esse problema pode ser resolvido por uma abordagem híbrida, que combina a

versatilidade da injeção por *software* com a precisão do monitoramento de *hardware*.

2.3 Sistemas de recomendação

Parte essencial de vários sistemas computacionais hoje em dia, sistemas de recomendação estão cada vez mais presentes na vida dos usuários de redes sociais e de usuários casuais de internet. Esses sistemas estão muitas vezes presentes de forma imperceptível, e podem ser partes fundamentais do modelo de negócio de várias companhias.

Nesta seção, serão apresentados os conceitos básicos dos sistemas de recomendação, suas abordagens mais conhecidas e aplicadas, bem como exemplos práticos de sua aplicação em situações do cotidiano.

2.3.1 Conceitos básicos

Sistemas de recomendação são um conjunto de técnicas e ferramentas que fornecem sugestões de itens que podem ser úteis para o usuário final (RESNICK; VARIAN, 1997; GOOD et al., 1999; BURKE, 2007). Amplamente utilizados na indústria de computação e no mercado em geral, sistemas de recomendação têm ganhado notoriedade entre grandes empresas de comércio virtual, redes sociais e serviços de computação em nuvem, pela capacidade de entregar ao consumidor ou usuário final uma experiência personalizada e intuitiva com sua aplicação.

Em meio à popularidade e relevância desse tipo de sistema na indústria, um dos casos mais populares de incentivo à pesquisa desse tipo de sistema ocorreu no episódio conhecido como “**Prêmio Netflix**” (em inglês: *Netflix Prize*), ocorrido em outubro de 2006, quando a Netflix (NETFLIX, INC., 2019) lançou um banco de dados contendo 100 milhões de avaliações anônimas de 18.000 filmes, feitas por 480.000 usuários e desafiou a comunidade de cientistas da computação, pesquisadores de aprendizado de máquina e pesquisadores de *data mining* a desenvolver sistemas que conseguiriam superar seu próprio sistema de recomendação, conhecido como *Cinematch* (BENNETT; LANNING et al., 2007). O prêmio dessa competição era 1 milhão de dólares para a equipe vencedora, o que certamente chamou a atenção de muitos pesquisadores ao redor do planeta. Tal episódio gerou uma situação sem precedentes na história de pesquisa na área de filtragem colaborativa (BELL; KOREN, 2007). Em menos de um ano, mais de 20.000 times de 152 países diferentes haviam se registrado, sendo que desses apenas 2.000 haviam submetido seu conjunto de previsões até Junho de 2007 (BENNETT; LANNING et al., 2007). O prêmio final foi dado em 2009 para o time de pesquisa do *AT&T Labs*, chamado de **BellKor**, que havia conseguido resultados cerca de 10,10% melhores que o *Cinematch* (KOREN, 2009).

Existem inúmeras razões que levam empresas e provedores de serviços como a Netflix a explorar e melhorar esse tipo de sistema, as quais estão listadas a seguir (RICCI;

ROKACH; SHAPIRA, 2011):

- *Aumentar o número de itens vendidos:* Uma das funções mais importantes de Sistemas de Recomendação comerciais é ser capaz de vender itens adicionais aos escolhidos por consumidores. Os itens recomendados geralmente se adequam ao perfil de compra do usuário, e por isso são mais prováveis de serem comprados pelo mesmo enquanto pesquisa algo relevante. Aplicações não-comerciais também possuem os mesmos objetivos, mesmo que não haja um custo envolvido para o usuário ao selecionar um item.
- *Vender itens diversificados:* Um sistema de recomendação também deve ser capaz de permitir que o usuário selecione itens que possivelmente sejam mais difíceis de encontrar sem uma recomendação precisa. Por exemplo, ao pesquisar um filme na plataforma da Netflix, um usuário irá se deparar com títulos semelhantes em conteúdo, que possuem temática similar ao procurado, mas que geralmente são menos populares. Essa é uma forma de promoção desses títulos menos populares.
- *Aumentar a satisfação do usuário:* Um sistema de recomendação bem projetado pode melhorar a experiência do usuário com a aplicação. O usuário encontrará recomendações interessantes, relevantes e, com uma boa interação humano-computador, também irá apreciar a utilização do sistema.
- *Aumentar a fidelização do usuário:* Usuários tendem a se tornar mais fiéis a um determinado *Web site* que, ao ser visitado, reconhece o consumidor e o trata como um visitante valioso, personalizando a experiência do usuário com aquele serviço. Consequentemente, quanto mais o usuário utiliza o serviço, mais refinadas são as recomendações do sistema, e melhor é a representação de suas necessidades naquele sistema.
- *Melhor entendimento dos desejos do usuário:* Conectando com o ponto anterior, o entendimento do perfil do consumidor ou usuário é um importante aspecto de um sistema de recomendação. A previsão de desejos do usuário pode ser utilizado pelo provedor de serviço para inúmeros outros objetivos, como por exemplo melhor gerenciar o estoque de determinados itens ou produção, ou personalizar a experiência de outros usuários com gostos similares.

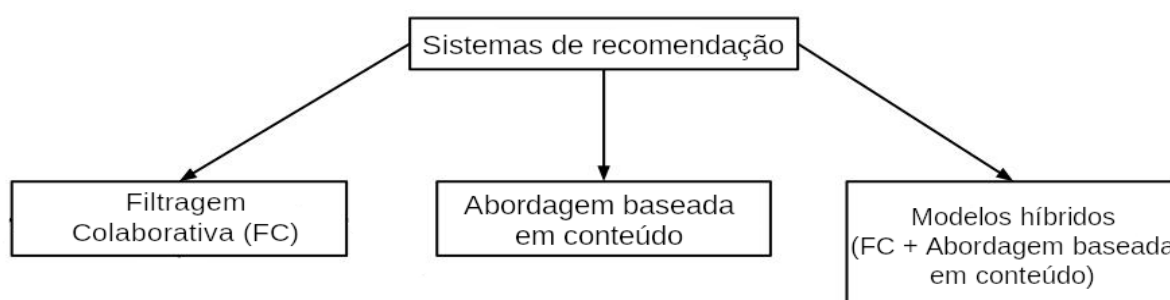
Como visto anteriormente, é importante que sistemas de recomendação colem diversos tipos de dados para construir suas recomendações (RICCI; ROKACH; SHAPIRA, 2011; HERLOCKER et al., 2004), para que dessa forma o sistema consiga fazer uma previsão de algo desejado pelo usuário final. Alguns desses dados podem ser coletados do histórico de outros usuários, ou por um banco de dados pré-estabelecido. Dependendo

da abordagem utilizada, o número de informações contidas no banco de dados pode ser crucial para refinar as recomendações do usuário (RESNICK; VARIAN, 1997).

2.3.2 Modelos de sistemas de recomendação

Existem alguns modelos de sistemas de recomendação trabalhados na literatura. De uma forma simplificada, podemos classificá-los em três abordagens distintas, como visualizado na Figura 3. As abordagens mais comuns são: **Filtragem Colaborativa**, **Baseada em Conteúdo** e **Híbrida** (RICCI; ROKACH; SHAPIRA, 2011).

Figura 3 – Modelos de sistemas de recomendação



(Traduzido e adaptado pelo autor (WIKIMEDIA COMMONS., 2019b))

Dentre esses modelos mencionados, o mais popular e mais presente na literatura é o sistema de recomendação de filtragem colaborativa (RICCI; ROKACH; SHAPIRA, 2011; ISINKAYE; FOLAJIMI; OJOKOH, 2015; GOOD et al., 1999), por sua maturidade e robustez de desenvolvimento, e por sua ampla aplicação e métodos de implementação. Como visto na Figura 3, existem diferentes formas de implementação de um sistema de recomendação de Filtragem Colaborativa, que serão exploradas nas subseções seguintes.

Existem também outras abordagens citadas na literatura, algumas até derivadas da filtragem colaborativa, como por exemplo recomendação baseada em contexto (ADOMAVICIUS; TUZHILIN, 2011), recomendação baseada em comunidade, recomendação baseado em conhecimento, recomendação baseada em demografia e recomendação baseada em utilidade (RICCI; ROKACH; SHAPIRA, 2011, p. 13). Algumas dessas abordagens ainda não apresentam um grau de maturidade tão avançado quando o método de Filtragem Colaborativa e, portanto, neste trabalho serão apenas apresentadas as técnicas mais comumente utilizadas e mais bem trabalhadas na literatura: filtragem colaborativa e recomendação baseada em conteúdo.

2.3.2.1 Filtragem Colaborativa

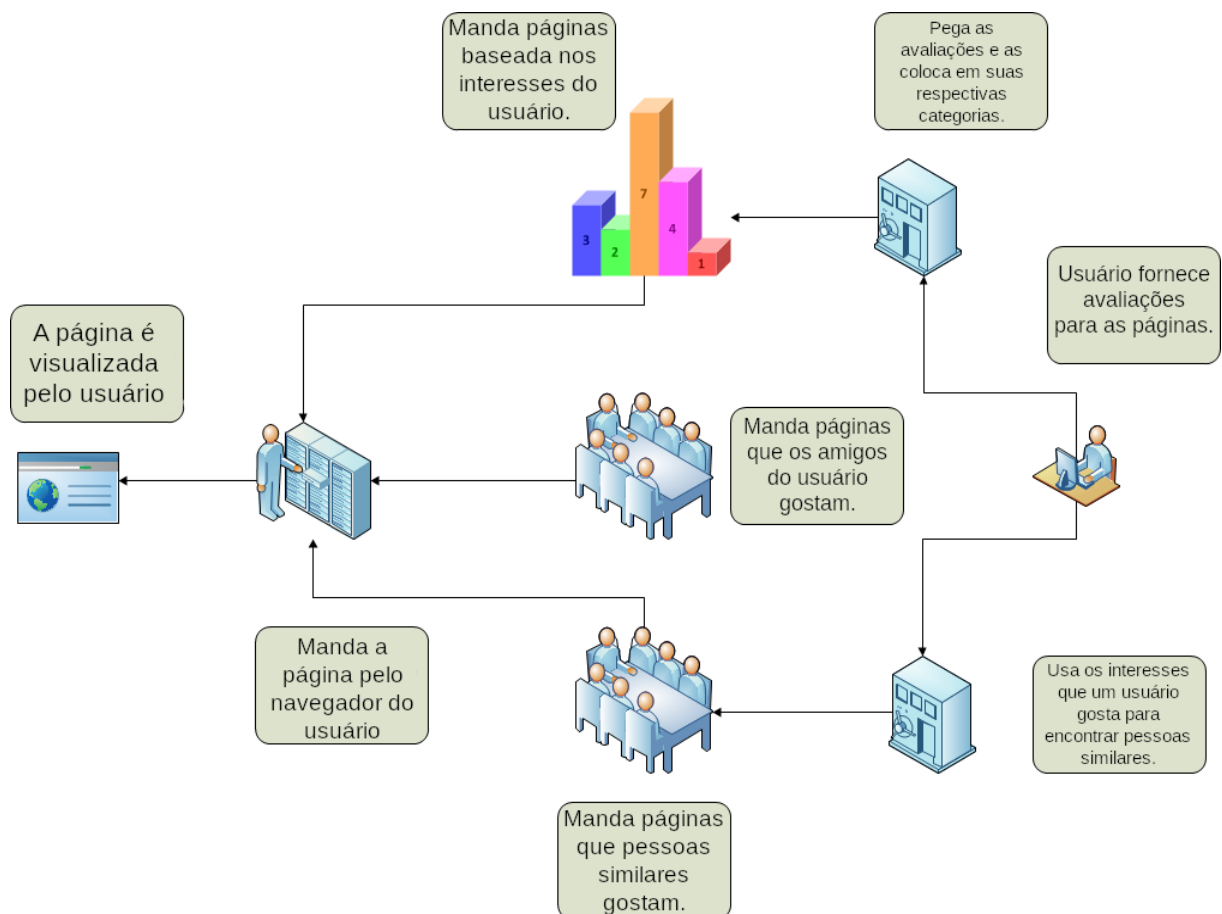
Trata-se do modelo de sistema de recomendação mais trabalhadas na literatura, também reconhecida como uma das mais promissoras tecnologias da área (ISINKAYE;

FOLAJIMI; OJOKOH, 2015; RESNICK; VARIAN, 1997; SCHAFER; KONSTAN; RIEDL, 1999), a filtragem colaborativa funciona por meio da construção de um banco de dados de preferências de itens por usuário (SARWAR et al., 2002). Quando um novo usuário entra no sistema, tal usuário é comparado com outros do banco de dados para descobrir *vizinhos*, que historicamente possuem gostos similares a ele.

A Filtragem Colaborativa é o processo de filtragem de informação ou padrões utilizando técnicas de colaboração entre múltiplos agentes, pontos de vista, fontes de dados, etc (TERVEEN; HILL, 2001). Aplicações que utilizam filtragem colaborativa geralmente envolvem grandes bancos de dados.

Para definir as recomendações, os usuários devem avaliar os itens (HERLOCKER et al., 2004; BURKE, 2007) para que esses sejam posteriormente sugeridos para outros usuários vizinhos no banco de dados. O funcionamento simplificado desse modelo está ilustrado na Figura 4.

Figura 4 – Diagrama do funcionamento da Filtragem Colaborativa



(Traduzido e adaptado pelo autor (WIKIMEDIA COMMONS., 2019a))

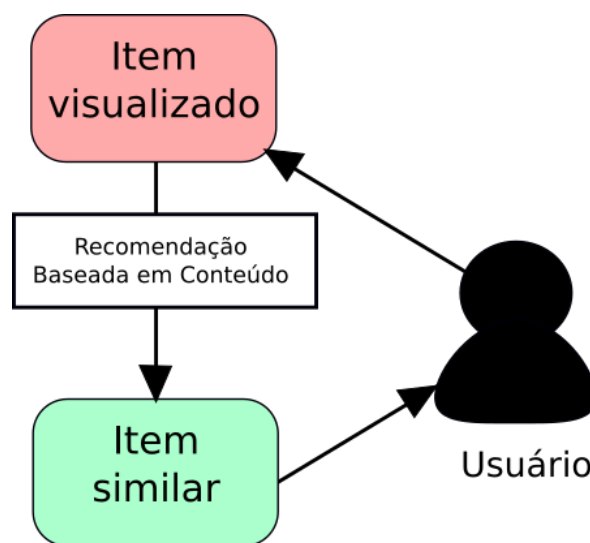
2.3.2.2 Recomendação Baseada em Conteúdo

A recomendação baseada em conteúdo tenta recomendar itens similares aos que o usuário mostrou interesse no passado. O processo básico realizado pelo recomendador consiste em combinar atributos do perfil do usuário nos quais preferências e interesses estão armazenados, com os atributos do item em questão, para assim poder fazer uma recomendação de novos itens interessantes ao usuário (PAZZANI; BILLSUS, 2007).

Dessa forma, em vez de simplesmente utilizar avaliações dos usuários em determinados itens, ele utiliza os aspectos de itens para determinar suas recomendações (BURKE, 2007). Esse tipo de recomendação tem como limitação o fato de servir apenas para recomendar itens relacionados, e não explorar uma gama de itens distintos como faz a filtragem colaborativa (BALABANOVIĆ; SHOHAM, 1997).

De forma simplificada, o diagrama que ilustra o funcionamento de um sistema de recomendação baseado em conteúdo está ilustrado na Figura 5.

Figura 5 – Diagrama do funcionamento de Recomendação baseada em conteúdo



2.3.2.3 Recomendação Híbrida

A utilização de ambas abordagens (recomendação baseada em conteúdo e filtragem colaborativa) para constituição de um sistema de recomendação é uma técnica utilizada por alguns sistemas de recomendação para aproveitar melhor as características de ambas as abordagens (BALABANOVIĆ; SHOHAM, 1997).

Nesse contexto, existem algumas formas distintas de combinar técnicas de recomendação, que estão contidas na Tabela 4 (BURKE, 2007).

Tabela 4 – Descrição dos métodos de hibridação

Método	Descrição
Ponderado	Os resultados dos sistemas de recomendação são combinados para produzir uma única recomendação.
Alternante	O sistema alterna entre os diferentes recomendadores de acordo com a situação.
Combinação de características	Características de diferentes fontes de recomendação são colocadas juntos em um único algoritmo de recomendação.
Mistura	Recomendações de vários recomendadores são apresentadas ao mesmo tempo.
Cascata	Um recomendador refina as recomendações apresentadas pelo outro.
Acréscimo de característica	A saída de uma técnica é utilizada como entrada de característica para outra técnica.
Meta-nível	O modelo aprendido por um recomendador é utilizado como entrada para outro recomendador.

2.3.3 Aplicações

O uso de tecnologias de recomendação é bastante presente em plataformas de computação em nuvem, e mostram-se bastante eficazes na arte de influenciar o usuário final a tomar uma decisão baseada naquilo que lhe é sugerido (SHANI; GUNAWARDANA, 2011). O uso de um sistema de recomendação varia de acordo com o modelo de negócio utilizados e sua plataforma necessária para funcionamento. As pesquisas em sistemas de recomendação são feitas com ênfase principalmente nas aplicações comerciais, já que, sem desmerecer suas contribuições teóricas, há uma preocupação em melhorar sua praticidade comercial (RICCI; ROKACH; SHAPIRA, 2011).

Sistemas como esse também são amplamente utilizados na área do comércio virtual, mais conhecido como *e-commerce* (SARWAR et al., 2002), fazendo a sugestão de produtos e anúncios para usuários que demonstram interesses em determinadas categorias pela internet. De uma forma geral, podemos descrever o uso prático de sistemas de recomendação nas seguintes áreas (RICCI; ROKACH; SHAPIRA, 2011):

- **Entretenimento:** recomendações de filmes, séries, desenhos, músicas, jogos.
- **Conteúdo:** jornais personalizados, recomendação de documentos, livros, páginas *web*, aplicações em aprendizado a distância, filtros de *e-mails*.
- **E-commerce:** recomendação para consumidores para comprar produtos como livros, câmeras, computadores etc.
- **Serviços:** recomendação de serviços de viagens, recomendação de experts para consultas, recomendação de casas para alugar, ou serviços que combinem duas

pessoas, como um aplicativo de encontros, ou até mesmo sugestão de oponentes em algum jogo.

Os sistemas de recomendação também ajudam a melhorar as vendas de três formas (SCHAFER; KONSTAN; RIEDL, 2001):

- **Convertendo visitantes em consumidores:** Visitantes de seu *web site* frequentemente acessam o serviço sem intenção de comprar nada. Os sistemas ajudam os consumidores a encontrar itens que potencialmente eles possam se interessar e comprar.
- **Aumentar a venda cruzada:** Sistemas de recomendação podem ajudar a aumentar a venda cruzada através da sugestão de itens adicionais para que o consumidor possa comprar. Nesse caso, o serviço pode recomendar produtos adicionais durante o processo de pagamento, baseado nos produtos que já estão no carrinho de compras.
- **Construir fidelidade:** A fidelidade de clientes é muito importante na área de comércio. Com um bom sistema de recomendação que atenda às necessidades do seu usuário, o cliente saberá que ao visitar aquele determinado *web site* ele terá uma experiência de compra personalizada, e sempre voltará a comprar no mesmo serviço. Isso não só ajuda o próprio sistema de recomendação a refinar seu conhecimento sobre o cliente, como também traz lucro para a companhia.

Neste contexto, serão abordados neste trabalho, de forma prática, o uso de sistemas de recomendações em redes sociais, *e-commerce*, em serviços diferentes e até mesmo em estudos e pesquisas de áreas como medicina e biologia.

2.3.3.1 Redes sociais

Sistemas de recomendação são onipresentes nas redes sociais, e tornaram-se importantes para formar conexões entre usuários, vender produtos e descobrir novos conteúdos nessas mídias (BURKE, 2007). Eles se tornaram partes importantes de *web sites* como YouTube, Facebook, LinkedIn, Twitter e Instagram, por exemplo.

Um bom exemplo do uso de sistemas de recomendação nesse âmbito é o algoritmo de recomendação de vídeos utilizado pelo YouTube. Um estudo realizado em 2010 (ZHOU; KHEMMARAT; GAO, 2010), mostra uma correlação entre o sistema de recomendação de vídeos do YouTube e as visualizações de cada vídeo afetado, indicando que a maioria das visualizações desses vídeos recomendados pelo *web site* advinham da própria recomendação.

No caso do Facebook, a maior parte dos sistemas de recomendação são baseados em Filtragem Colaborativa (SHAPIRA; ROKACH; FREILIKHMAN, 2013), em que ele

usa dados coletados dos usuários que têm interesses e postagens em comum para sugerir conteúdos que também coincidam com tais interesses. O Facebook, assim como LinkedIn, Twitter e Instagram, também possuem sistemas de recomendações de pessoas que possam fazer parte do círculo social do indivíduo, uma abordagem também baseada na Filtragem Colaborativa, que tem se mostrado bastante poder nessas situações por ser capaz de reunir gostos similares dos usuários e conseguir calcular níveis de confiança em amizades (CHEN; FONG, 2010).

2.3.3.2 *E-commerce*

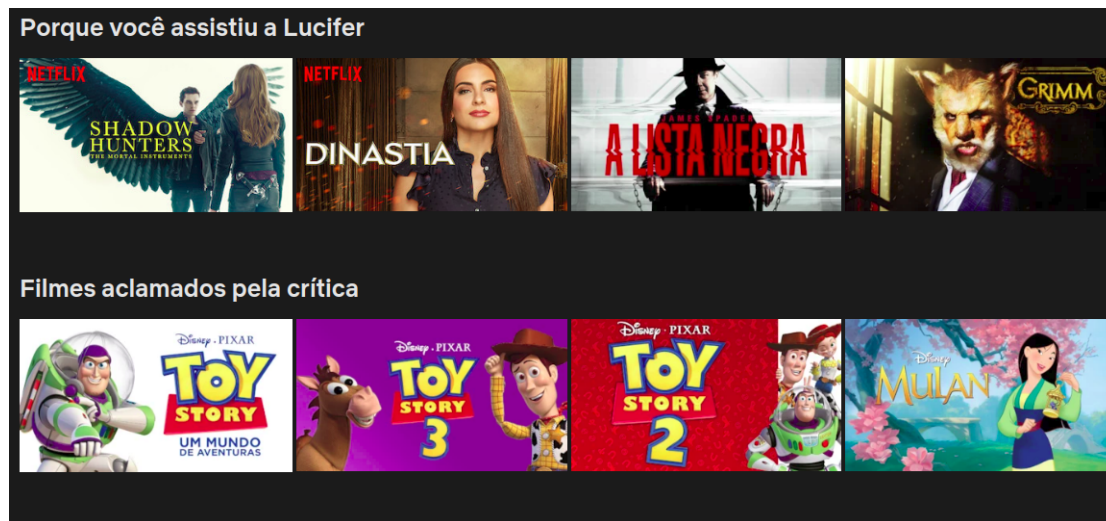
Uma das maiores áreas que utilizam sistemas de recomendação é o âmbito do *e-commerce*, conhecido também como comércio virtual (SCHAFFER; KONSTAN; RIEDL, 1999). Neste, destacam-se no mercado brasileiro grandes nomes do varejo e vendas em geral como Walmart, Carrefour, Saraiva, Casas Bahia, Pontofrio etc. e outras empresas focadas exclusivamente no comércio virtual como Amazon, Mercado Livre, KaBuM, entre outras.

Páginas *web* como **Amazon.com** apresentam sistemas de recomendação que mostram produtos similares a itens visualizados recentemente, sistemas com recomendações personalizadas para usuários, ofertas mais baratas, e também um sistema que mostra os itens mais comprados pelos usuários (ofertas em alta ou mais vendidos) (SCHAFFER; KONSTAN; RIEDL, 2001). Tal prática também é seguida por diversos outros serviços de *e-commerce* disponíveis.

2.3.3.3 Serviços

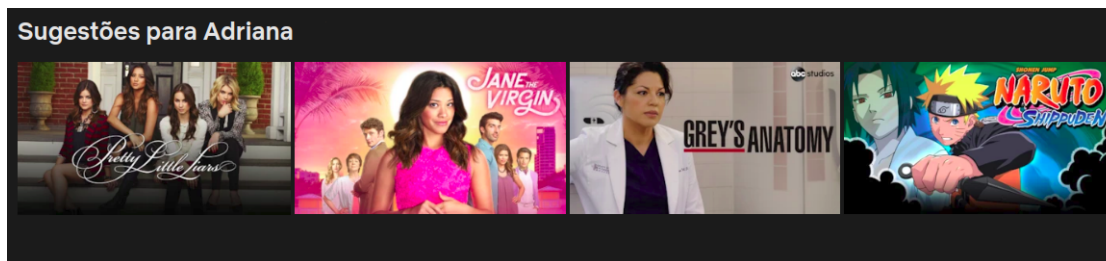
Como visto anteriormente, serviços como Netflix focam bastante em sistemas de recomendação para melhorar a experiência do usuário com seus serviços. Utilizando tal serviço como exemplo, pode-se identificar diversos exemplos do uso de sistemas de recomendação na sua página inicial (GOMEZ-URIBE; HUNT, 2016), como ilustram as Figuras 6 e 7.

Figura 6 – Exemplo de recomendação por conteúdo assistido anteriormente e filmes relevantes



Como pode-se observar na Figura 6, os conteúdos sugeridos baseiam-se no conteúdo de itens previamente assistidos e na coluna seguinte mostra filmes bem avaliados de modo geral, que se encaixam no perfil do usuário em questão.

Figura 7 – Exemplo de recomendação personalizada



Já na Figura 7, os conteúdos sugeridos baseiam-se no perfil do usuário construído pelo sistema de recomendação, e também associa o conteúdo a perfis similares, o que leva a uma série de sugestões personalizadas ao usuário final. Tal prática exemplificada pelo serviço Netflix também é replicada por muitos outros serviços da categoria, como Amazon Prime Video, Hulu, entre outros.

2.4 Trabalhos relacionados

Tratando-se de computação em nuvem, há diversos trabalhos e aplicações publicados que podem ser mencionados. Os serviços do Microsoft Azure, que fazem parte dessa área, têm sido utilizados em diversas pesquisas distintas, como pesquisa em geoprocessamento (GONG; YUE; ZHOU, 2010), predição de estruturas de proteínas em 3D (MROZEK;

GOSK; MAŁYSIAK-MROZEK, 2015) e utilização de aprendizado de máquina em pesquisa de radiologia (KOHLI et al., 2017).

Também pode-se mencionar trabalhos utilizando serviços da Amazon, como os trabalhos na área da Biomedicina utilizando computação em nuvem (FUSARO et al., 2011), experimento com sequenciamento genômico chamado **Globus Genomics** (MADDURI et al., 2014) e análise de performance de computação de alto desempenho (JACKSON et al., 2010). Tais artigos demonstram o potencial crescente de plataformas de computação em nuvem, e demonstram que tais ambientes podem ser utilizados com sucesso em pesquisas importantes, nas quais a obtenção de ferramentas computacionais poderosas próprias é bastante dificultada ou quase impossível.

No âmbito de tolerância a falhas em computação em nuvem, este trabalho está associado ao projeto realizado por ANDRADE (2019), que realizou uma revisão sistemática da literatura e implementou um sistema de classificação de técnicas de tolerância a falhas.

Na área de sistemas de recomendação, ADOMAVICIUS; TUZHILIN (2005) realizou um *survey* sobre o estado da arte desse tipo de tecnologia, levando em consideração trabalhos de RESNICK et al. (1994), HILL et al. (1995) e BALABANOVIĆ; SHOHAM, que implementaram seus sistemas de recomendação e introduziram diferentes abordagens para execução.

2.5 Considerações finais

Serviços de computação em nuvem são praticamente onipresentes na vida de usuários de sistemas computacionais nos dias modernos. Pesquisadores se esforçam diariamente para manter os sistemas operantes e livre de quaisquer tipos de falhas que possam vir a comprometer seu funcionamento em tempo de execução. Como pôde ser visto, o número de técnicas de tolerância a falhas é bastante elevado e pode causar confusão em administradores de sistemas.

Nesse âmbito pode-se observar uma lacuna em que se encaixa um sistema de recomendação para as técnicas de tolerância a falhas mais comuns, já que não foi possível encontrar na literatura quaisquer tentativas que caminhassem nessa direção.

3 PROJETO

O projeto em questão consiste na implementação de um sistema de recomendação para técnicas de tolerância a falhas utilizando os dados coletados em um trabalho realizado por [ANDRADE \(2019\)](#). Nesse trabalho foi desenvolvido um aplicativo com as informações sobre as técnicas de tolerância dispostas em forma de tabelas estáticas.

Levando isso em consideração, este projeto visa aprimorar tal conceito, implementando uma forma de armazenar os dados contidos nessas tabelas em um arquivo de troca de dados, para que os usuários consigam interagir com esses elementos, criando um sistema de recomendação em cima desses dados e permitindo a busca por caracteres relevantes. Dessa forma, a ferramenta desenvolvida se torna ideal para a busca de técnicas de tolerância a falhas, de uma forma intuitiva e facilitada para o usuário final.

3.1 Motivação

O intuito deste trabalho é facilitar a pesquisa de técnicas de tolerância a falhas por meio de uma ferramenta gráfica intuitiva, com recomendação de técnicas e com a possibilidade de pesquisa. Para isso, utilizamos os dados coletados no trabalho citado anteriormente, alimentando este projeto com informações sobre as diferentes 52 técnicas de tolerância a falhas, permitindo sua pesquisa e recomendação na ferramenta.

Assim, o projeto final deve apresentar uma interface gráfica básica e um sistema de recomendação, que poderá filtrar por tipo de falha, podendo ser acessado e aprimorado pelo usuário, por meio da avaliação das técnicas apresentadas.

3.2 Metodologia

Neste projeto será utilizado o *framework* Ionic para implementação do projeto e sua interface gráfica, um sistema de recomendação básico escrito em Python, utilizando biblioteca oferecidas pela linguagem de programação e o *framework* Flask para fazer a comunicação entre os sistemas.

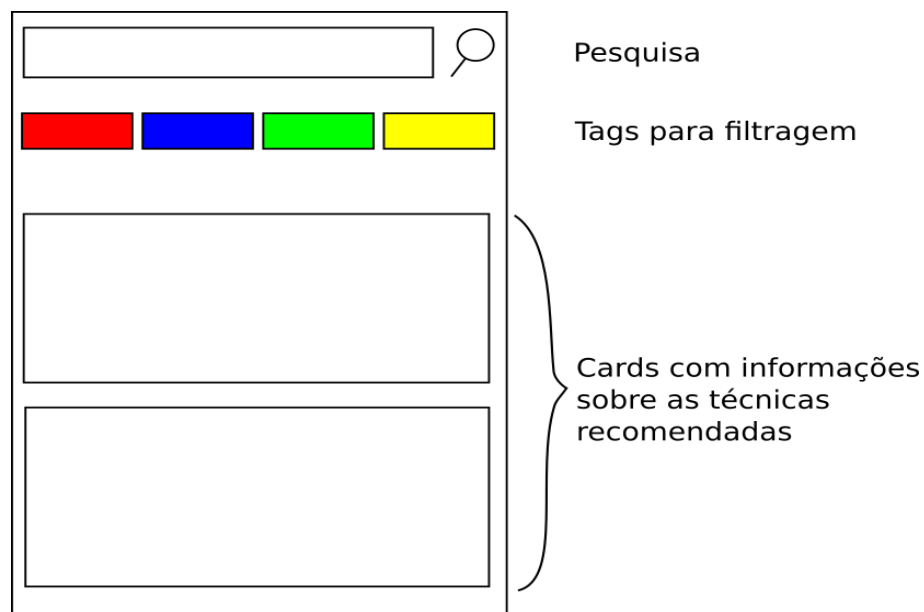
3.2.1 Descrição do Sistema

O sistema proposto neste trabalho trata-se de um protótipo de uma ferramenta com sistema de recomendação que pode ser utilizado e expandido para uso posterior ou para quaisquer outros propósitos. Nesta ferramenta, os itens a serem recomendados são as técnicas de tolerância a falhas, e o usuário poderá filtrar as técnicas por tipos de falhas,

ou então poderá pesquisá-las por meio de uma barra de busca. O intuito dessa ferramenta é auxiliar o trabalho de administradores de sistemas que demandam tolerância a falhas, tornando mais simples o trabalho de pesquisa e comparação de técnicas de tolerância a falhas, permitindo também que o usuário avalie a técnica disponível e consiga aperfeiçoar o sistema para outros usuários.

Na figura 8 está representado um projeto da tela inicial do sistema e seus principais componentes: a pesquisa, a filtragem pelas falhas (*tags*) e os *cards* com as informações das técnicas de tolerância recomendadas.

Figura 8 – Planejamento inicial de interface da ferramenta



3.2.2 Desafios

Uma grande parte do conteúdo disponível sobre sistemas de recomendação está voltado para itens mais comuns no cotidiano, como serviços de *e-commerce*, recomendação de filmes e itens, recomendação de amizades ou conexões prováveis, dentre tantas outras aplicações que estão distantes das formas com que o sistema é aplicado neste trabalho. Além disso, existem inúmeras limitações para sistemas de recomendação com poucos dados, como é o caso deste trabalho, em que os dados coletados e disponíveis não estão calibrados suficientemente por uma enorme gama de usuários que utilizam o serviço.

Por conta desses detalhes o trabalho teve de ser adaptado e cuidadosamente trabalhado para evitar que essas limitações viessem a atrapalhar as recomendações resultantes dos itens registrados. As adaptações feitas estão descritas na seção 3.2.3, bem como alguns detalhes sobre os algoritmos empregados.

3.2.3 Implementação

O início da implementação do projeto consiste em transformar os dados disponíveis sobre as técnicas de tolerância a falhas em dados que possam ser trabalhados pela aplicação, utilizando tags relevantes e conteúdos que possam ser armazenados em um banco de dados ou em um arquivo de troca de dados, como por exemplo arquivos no formato JSON (JavaScript Object Notation) ou CSV (Comma Separated Values).

Neste projeto, foi escolhido o armazenamento de dados utilizando arquivos JSON e CSV, pela simplicidade oferecida pelos arquivos e pela forma rápida de responsividade desses arquivos utilizando as linguagens Python e JavaScript.

3.2.3.1 Coleta de dados

Os dados referentes às técnicas de tolerância a falhas foram coletados manualmente em um arquivo de extensão “.txt”, utilizando as tabelas desenvolvidas no trabalho de [ANDRADE \(2019\)](#), e mais tarde foi utilizado um script desenvolvido em Python pelo autor para adequá-los ao formato JSON.

Cada uma das tabelas descritas no documento foi cuidadosamente analisada, adicionando os dados relevantes para tornar tais tabelas mais compatíveis com um modelo de sistema de recomendação, colocando itens como classificação de usuários. No total, foram identificadas 52 técnicas de tolerância a falhas, divididas em 4 tipos de falhas principais: falhas relacionadas a dados, a processos, a virtualização e a comunicação.

As técnicas selecionadas estão listadas a seguir ([ANDRADE, 2019](#)):

- | | |
|---|---|
| – InterGrid Gateway (IGG) | – Algoritmo híbrido baseado em MapReduce |
| – HySARC | – Morpho |
| – SkyCDS | – Dynamic Cost-Aware Re-Replication and Re-balancing Strategy (DCR2S) |
| – DARGOS | – Fuzzy Inference based Reliable Replica Replacement (FIR3) |
| – Protótipo para tratamento automático de anomalias | – Medical Image File Accessing System (MIFAS) |
| – Tahoe-LAFS | – Sistema baseado no princípio 80/20 |
| – SwiftER | – Middleware de nuvem para garantir desempenho e alta disponibilidade de aplicativos soft em tempo real |
| – Sistema de armazenamento privado multicamadas | |
| – Gluster File System (GFS) | |

-
- mOSAIC
 - Pilot-Data
 - ONHelp
 - Sistema de resgate de spot inesperados usando instâncias de spot heterogêneos e excesso de provisão
 - Symbiotic Virtualization (SymVirt)
 - Mantis
 - Fault-Tolerant Stateful Firewall (FT-FW)
 - Sistema de controle de acesso avançado
 - Protótipo para Extração sistemática dos estados de rede
 - pFTree-Ext e pFTree-Wt
 - Auto healing service framework
 - Mayflies
 - TOPSIS
 - Group-U
 - Dering Over Shared Memory (DO-RADO)
 - COSCAnet-FT
 - Cloud Inter-operation Toolkit(CIT)
 - OpenADN
 - SprintNet
 - Future Automation System Architecture (FASA)
 - Fault Tolerance Manager (FTM)
 - Fault-tolerance in federated Clouds (FT-FC)
 - Promethee Scheduler
 - Proposta de arquitetura de cluster virtual tolerante a falhas
 - Cloud-Agnostic Checkpointing Service (CACS)
 - Unibus
 - Availability Knob
 - Protótipo para Sistema de Migração de VM
 - Fault-Tolerant Elastic Scheduling Algorithm for Workflow (FTESW)
 - Sistema de detecção e mitigação de falhas
 - Algoritmo de tolerância a falhas energéticas para computação de alto desempenho
 - Zurmo Native
 - Proactive Fault Tolerance Mechanism for Data Center Network
 - Replicação de VM para melhorar a capacidade de sobrevivência de aplicativos de missão crítica
 - HAShield
 - FTStack

Cada uma das características de cada técnica foi extraída e compilada de forma adequada no arquivo de dados JSON correspondente. Dessa forma, cada uma das técnicas

cadastradas no sistema possui um conjunto único de características e permite que o usuário possa visualizá-las para poder fazer a comparação e descobrir suas vantagens e desvantagens. As características essenciais são: Nome, Técnicas utilizadas, Gestor de Nuvem, Modelo de implementação, Problema que visa solucionar, Tipo de falha tratada, Vantagens e, por fim, Desvantagens.

Um aspecto importante para um sistema de recomendação é a avaliação de usuários sobre as técnicas em questão. É importante que haja uma relação direta entre usuários e as técnicas para que o sistema de recomendação produza resultados precisos e refinados. Entretanto, tais sistemas geralmente trabalham com grandes quantidades de dados, e quando não há uma grande coleção de informações para trabalhar, as recomendações feitas são bastante imprecisas.

Um dos maiores problemas de sistemas de filtragem colaborativa é o problema *cold-start*, que descreve justamente o que foi mencionado anteriormente: novos itens ou novos usuários não possuem dados suficientes para que se possa fazer uma recomendação baseada em seu perfil ou em características descritas (LIKA; KOLOMVATSOS; HADJIEFTHYMIADES, 2014; LAM et al., 2008).

Para que se possa validar os testes do sistema, deve-se então possuir uma ampla base de usuários e dados que validem as recomendações feitas. No entanto, como este trabalho possui um número bastante limitado de usuários e sequer se trata de um produto de larga escala, realizar essa validação pode parecer impossível.

Dessa forma, para contornar este problema inicial foi feita a geração de dados aleatórios seguindo uma distribuição normal para diferentes quantidades de usuários, e também para servir de base para o sistema inicial. Assim, é possível produzir recomendações básicas para usuários do sistema e evitar o problema inicial para as propostas deste trabalho. Na seção 3.2.3.2 será discutido como tal método foi implementado.

3.2.3.2 Geração de dados

Os dados utilizados para os testes de validação do sistema de recomendação foram gerados de forma automática, utilizando um *script* em Python escrito pelo próprio autor que gera um arquivo em CSV com os seguintes componentes:

- Identificador do Usuário (UID)
- Identificador da Técnica (IID)
- Avaliação Consolidada (Rating)

O *script* em questão gera um identificador aleatório para o usuário e também gera um valor no intervalo de 0 a 51, correspondendo às 52 técnicas propostas, e assinala uma

avaliação que segue uma distribuição gaussiana com os parâmetros $\mu = 3$ e $\sigma = 1$, obtendo resultados mais próximos de uma avaliação real em termos estatísticos. Para as finalidades deste trabalho, os valores de avaliação gerados foram arredondados utilizando uma função piso, evitando assim qualquer tipo de conflito durante os cálculos de avaliações.

Além disso, o *script* também checa para ver se a tupla usuário e técnica já foi registrada anteriormente, evitando registrar duas avaliações distintas para o mesmo usuário, o que também apresentaria um problema para o sistema de recomendação.

Com esses dados é possível fazer com que o sistema de recomendação faça sugestões com melhor precisão, evitando o problema descrito anteriormente. Assim, recebendo um parâmetro x e um determinado número de usuários (**nuser**), o *script* é capaz de gerar o produto desses dois parâmetros como o total de avaliações possíveis.

Por exemplo, se for estabelecido: $x = 10$ e $nuser = 5$, então teremos um total de 50 avaliações nos arquivos de dados.

Com isso, foram gerados seis diferentes arquivos de extensão CSV para os testes dos dados, com o seguinte nome: **userdata_x_nuser.csv**. Escolhendo o parâmetro $x = 5$ e $x = 10$ de maneira arbitrária, com um número de usuários crescente, partindo de 10 usuários até 1000, os seguintes arquivos foram gerados:

- userdata_5_10.csv
- userdata_5_100.csv
- userdata_5_1000.csv
- userdata_10_10.csv
- userdata_10_100.csv
- userdata_10_1000.csv

Tais parâmetros foram escolhidos para simplificar a disposição dos testes neste documento.

Todos os arquivos mencionados, incluindo o código fonte do *script* mencionado estão disponíveis na página do projeto no GitHub.

3.2.3.3 Armazenamento de dados

Os dados coletados e gerados são armazenados em formatos de arquivos simples JSON e CSV, como mencionado anteriormente. A simplicidade e rapidez que o trabalho com essas extensões de arquivos oferece torna desnecessário o uso de algum sistema de gerenciamento de banco de dados mais robusto, pelo menos para os propósitos deste trabalho.

3.2.3.4 Sistema de Recomendação

O sistema de recomendação implementado segue o modelo de filtragem colaborativa, por ser o modelo mais amplamente utilizado na indústria, o mais robusto e confiável nesse quesito.

A implementação do sistema de recomendação referente teve de ser bastante cautelosa e profundamente estudada. Existem inúmeros algoritmos que compõem sistemas de recomendação de diversos tipos, e, neste projeto, construir um sistema de recomendação que seja capaz de validar resultados e que seja flexível o suficiente era o maior objetivo. A implementação de cada um desses algoritmos de forma eficiente e otimizada tornou-se inviável para a proposta deste trabalho, e por esse motivo optou-se por uma biblioteca especializada para esse tipo de aplicação.

A linguagem Python possui grandes aplicações na área de inteligência artificial em geral, e as formas de lidar com ciência de dados são bastante numerosas. Não surpreendentemente, existem diversas bibliotecas que possibilitam a implementação de sistemas de recomendação usando a linguagem.

Dessa forma, para tornar mais prático o trabalho de implementação de diversos tipos de sistemas de recomendação, e ser capaz de realizar testes de validação das previsões desses sistemas, será utilizada a ferramenta **Surprise** (HUG, 2017), uma biblioteca em Python que facilita a implementação e validação de sistemas de recomendação de filtragem colaborativa. Tal ferramenta é composta por algoritmos matemáticos como a decomposição em valores singulares, máxima verossimilhança, fatorização de matrizes, k-NN, entre outras abordagens matemáticas e estatísticas que são aplicadas nesse âmbito.

Além disso, a ferramenta tem funções para lidar com conjuntos de dados (*datasets*) e possibilita o treinamento dos algoritmos mencionados, para obtenção de melhores resultados.

3.2.3.5 Interface gráfica

A interface gráfica da ferramenta foi implementada usando o *framework* Ionic com a plataforma Angular. Tal interface conecta-se ao sistema desenvolvido em Python por meio de um intermediário, o Flask, um *framework web* escrito em Python.

Os *cards* de recomendação com as informações das técnicas utilizam os dados processados pelo sistema de recomendação e são mostrados na forma de lista pelo Ionic, com as informações básicas dispostas. O usuário, a partir da interface, pode clicar nas respectivas técnicas para obter informações detalhadas sobre elas. Além disso, a interface gráfica possui uma barra de pesquisa que permite filtragem por *tags* diversas, que serão descritas na seção 4.2.

3.3 Considerações finais

Neste capítulo foram apresentados todos os procedimentos relevantes para a implementação do projeto, bem como os desafios e funcionalidades planejados. No capítulo a seguir, serão apresentados os testes realizados utilizando os sistemas implementados, bem como apresentações das funcionalidades da ferramenta em si, mostrando seu potencial para os fins desejados.

4 TESTES E RESULTADOS

Neste capítulo serão apresentados os testes e resultados obtidos utilizando o sistema implementado. De forma geral, serão apresentadas algumas análises dos dados processados, formas de validação dessa análise, e por fim uma visão geral do funcionamento do sistema como um todo.

4.1 Validação e Previsões

Os dados gerados são bastante extensos para uma análise bastante detalhada, pois existem muitas informações a serem consideradas e analisadas. Sabendo disso, podemos utilizar os métodos de validação cruzada disponíveis na biblioteca Surprise para avaliar a precisão das previsões feitas pelo algoritmo de recomendação utilizado.

Nos métodos de validação cruzada serão utilizadas duas formas de medida para determinar tal precisão: o Erro Médio Absoluto, ou *Mean Absolute Error* (MAE), e a Raiz do Erro Quadrático Médio, ou *Root Mean Squared Error* (RMSE).

De forma simplificada, ambas as medidas podem ser utilizadas para a validação de modelos preditivos, sendo bastante utilizadas em aplicações de aprendizado de máquina, modelos de regressão e simulações computacionais. Elas são utilizadas nas chamadas validações cruzadas, que consistem em dividir o conjunto de dados em K conjuntos menores de treino e testes, de forma independente, para que seus resultados sejam comparados e que assim se possa determinar a proximidade dos resultados obtidos na previsão desses conjuntos menores.

As medidas de erro são aplicadas em cada um dos conjuntos de validação cruzada, e quanto mais próximo de zero, mais precisa é a previsão obtida.

4.1.1 Primeiro teste

Neste primeiro teste serão aplicados os algoritmos de validação cruzada para os arquivos que receberam parâmetros $x = 5$, ou seja, os arquivos com um número relativamente baixo de dados disponíveis. Assim pode-se estabelecer uma comparação entre os resultados obtidos com poucos dados e os resultados obtidos com muitos dados.

Primeiramente é necessário escolher um algoritmo para o treinamento dos dados e assim poder realizar as previsões. Neste primeiro teste será utilizado o algoritmo de Decomposição em valores singulares, e o código para executar todo esse treinamento e validação é o seguinte:

```
def get_predictions(dataset):
    reader = Reader(line_format='user item rating', sep=',', rating_scale=(1, 5))
    data = Dataset.load_from_file(dataset, reader=reader)
    trainset = data.build_full_trainset()
    algo = SVD()
    algo.fit(trainset)

    testset = trainset.build_anti_testset()
    predictions = algo.test(testset)

    cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

get_predictions('userdata_5_10.csv')
get_predictions('userdata_5_100.csv')
get_predictions('userdata_5_1000.csv')
```

Deve-se notar que a cada execução, os dados obtidos dos algoritmos variam, assim como suas previsões realizadas, e isso pode prejudicar a reprodutibilidade dos resultados. O motivo desta mudança é que a cada execução a biblioteca Surprise gera um número aleatório para a validação cruzada e predição de algoritmos.

Para resolver este problema, o *script* utilizado utiliza um valor de semente *seed* = 0 para a biblioteca *random* do Python. Todos os testes executados neste documento utilizarão o mesmo valor de semente, para que esses resultados sejam reprodutíveis posteriormente. O seguinte código deve ser incluído no código fonte a ser executado:

```
import random
import numpy as np

my_seed = 0
random.seed(my_seed)
np.random.seed(my_seed)
```

Com os parâmetros estabelecidos, os resultados obtidos na validação cruzada de cada um dos arquivos gerados estão descritos nas tabelas 5, 6 e 7.

Tabela 5 – Teste com o arquivo userdata_5_10.csv

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.0986	0.8522	1.0072	1.0630	1.0661	1.0174
MAE	0.9509	0.6995	0.8610	0.8912	0.9178	0.8641

Tabela 6 – Teste com o arquivo userdata_5_100.csv

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.1350	1.0167	0.9913	1.0644	0.9738	1.0362
MAE	0.9513	0.8564	0.8235	0.8928	0.8213	0.8690

Tabela 7 – Teste com o arquivo userdata_5_1000.csv

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.0064	1.0083	0.9964	0.9933	1.0135	1.0036
MAE	0.8403	0.8444	0.8216	0.8251	0.8393	0.8341

Pode-se observar das tabelas que os valores obtidos estão bastante próximos de 1, o que nos diz que a margem de erro das previsões feitas é baixa, mas que ainda assim não é precisa o suficiente. Isso se deve talvez pelo baixo número de avaliações disponíveis nos arquivos.

4.1.2 Segundo teste

De maneira similar ao teste anterior, os testes desta seção serão executados nos arquivos gerados que receberam parâmetro $x = 10$, tratando-se então de um conjunto de dados relativamente maior que o anterior. As alterações no código descritas anteriormente apenas limitaram-se apenas à mudança dos arquivos de dados gerados.

Os resultados obtidos desses novos testes encontram-se nas tabelas 8, 9 e 10.

Tabela 8 – Teste com o arquivo userdata_10_10.csv

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.2226	1.2449	1.3538	1.0392	0.9741	1.1669
MAE	1.0095	1.0007	1.1697	0.8224	0.7771	0.9559

Tabela 9 – Teste com o arquivo userdata_10_100.csv

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	0.9977	1.0369	1.0416	1.0479	1.0037	1.0256
MAE	0.8316	0.8675	0.8781	0.8501	0.8234	0.8502

Neste novo caso, os valores ainda estão mais próximos de 1, o que indica uma margem de erro próxima àquela vista nos testes anteriores, mesmo com um volume maior de dados. De forma geral, pode-se notar por esses testes que a distribuição das avaliações tem grande impacto no valor obtido no erro de previsões, e não somente pela quantidade de dados disponíveis.

Ainda assim, os dados gerados não estão tão longe de uma aplicação real, em que a margem de erro também gira em torno de 0,7 a 1,1 na maioria dos casos. Isso significa

Tabela 10 – Teste com o arquivo userdata_10_1000.csv

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.0190	1.0149	1.0354	0.9931	1.0313	1.0188
MAE	0.8459	0.8404	0.8538	0.8208	0.8527	0.8427

que o sistema é sim capaz de gerar boas recomendações utilizando o algoritmo descrito, mas também há margem para melhores recomendações.

4.1.3 Terceiro teste

Dessa vez, de forma sucinta, serão apresentados os dados obtidos de todos os arquivos utilizando um algoritmo de previsão diferente, conhecido como Matriz de Fatorização Não-Negativa, ou NMF. Neste caso a única alteração no código foi relacionada ao algoritmo usado foi a mudança do valor de $algo = SVD()$ para $algo = NMF()$, que é a função já implementada do algoritmo na biblioteca.

Tal algoritmo é bastante utilizado em diversas outras áreas computacionais, como bioinformática, processamento de sinais, aprendizado de máquina e visão computacional, também tendo um papel bastante importante em sistemas de recomendação por filtragem colaborativa (LEE; SEUNG, 2001).

As tabelas 11, 12, 13, 14, 15 e 16 representam os resultados obtidos segundo os parâmetros estabelecidos previamente.

Tabela 11 – Teste com o arquivo userdata_5_10.csv usando matriz de fatorização não-negativa

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	0.7532	0.9825	0.9904	1.0747	0.9760	0.9553
MAE	0.6006	0.8421	0.7783	0.8695	0.7790	0.7739

Tabela 12 – Teste com o arquivo userdata_5_100.csv usando matriz de fatorização não-negativa

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.2298	1.1795	1.2699	1.3074	1.2851	1.2544
MAE	1.0099	0.9216	1.0362	1.0335	1.0565	1.0115

Tabela 13 – Teste com o arquivo userdata_5_1000.csv matriz de fatorização não-negativa

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.2246	1.1870	1.2114	1.2279	1.1568	1.2015
MAE	0.9810	0.9523	0.9764	0.9881	0.9328	0.9661

Tabela 14 – Teste com o arquivo userdata_10_10.csv matriz de fatorização não-negativa

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.4075	1.4194	1.3718	1.2201	1.4750	1.3788
MAE	1.0163	1.1976	1.1508	0.9800	1.0849	1.0859

Tabela 15 – Teste com o arquivo userdata_10_100.csv matriz de fatorização não-negativa

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.1635	1.1435	1.1476	1.1072	1.2127	1.1549
MAE	0.9502	0.9151	0.9485	0.8900	0.9915	0.9391

Tabela 16 – Teste com o arquivo userdata_10_1000.csv matriz de fatorização não-negativa

Medida	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Média
RMSE	1.1523	1.1766	1.1920	1.1729	1.1573	1.1702
MAE	0.9329	0.9496	0.9654	0.9473	0.9321	0.9455

Os resultados com esse algoritmo exibiram uma margem de erro mais elevada do que o algoritmo anterior, mas ainda assim muito próxima do valor 1. Isso mostra o quanto os dados gerados podem produzir resultados relevantes para um sistema de recomendação.

4.2 Interface gráfica e recomendações

A implementação do sistema de recomendação separa os dados das técnicas de tolerância a falhas de suas avaliações dos usuários, utilizando uma lista de números para oferecer recomendações, sem associar esses números com os dados específicos.

Como exemplo, a seguir estão algumas recomendações feitas pelo sistema para os usuários com identificadores 327, 266 e 481, escolhidos de forma arbitrária, ao utilizar um dos algoritmos descritos anteriormente sobre as condições propostas nas seções anteriores:

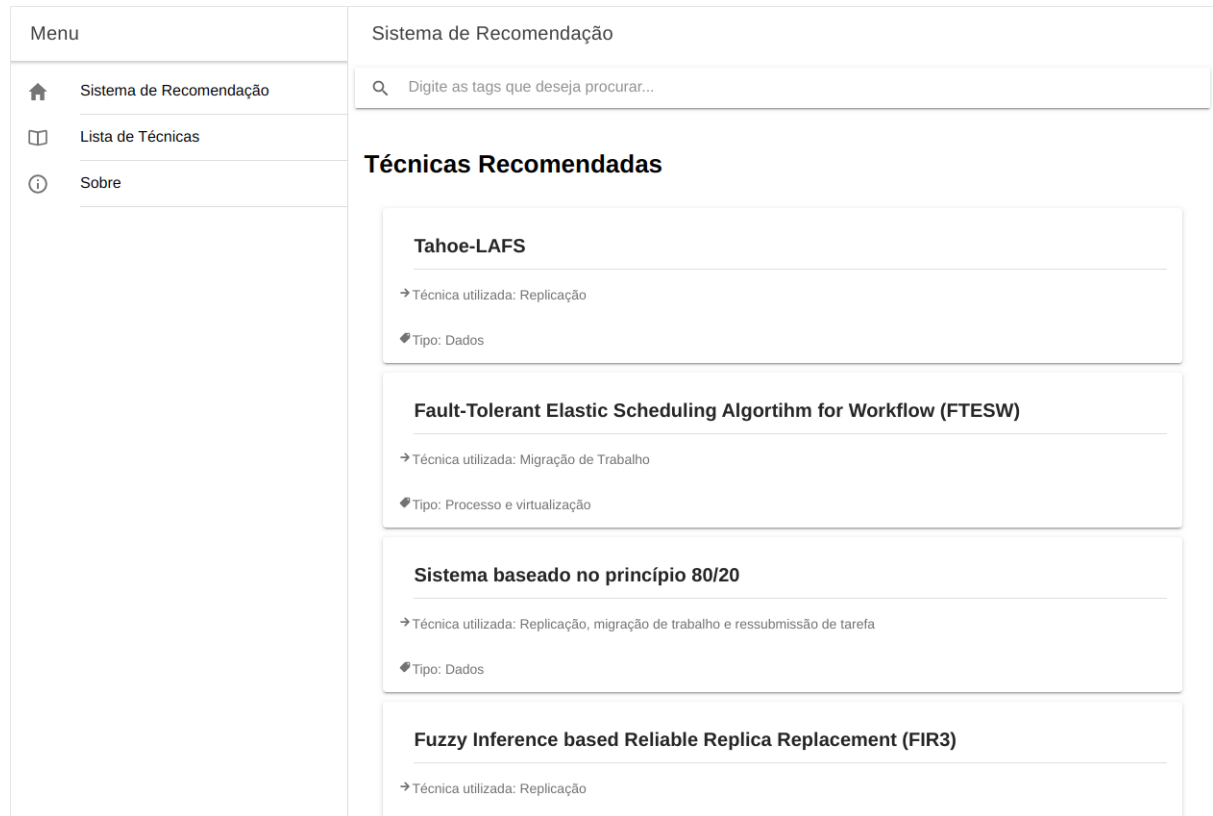
```
327 ['38', '51', '14', '32', '34', '33', '44', '36', '0', '21']
266 ['5', '17', '21', '39', '15', '34', '3', '47', '8', '19']
481 ['39', '7', '22', '43', '14', '28', '15', '3', '1', '21']
```

Neste caso, cada usuário recebe uma recomendação diferente de 10 técnicas, cada uma com seu identificador próprio (IID), sem estar associada a uma técnica em um banco de dados. Dessa forma torna-se complicado conectar os dados de maneira eficiente, e como o sistema é capaz de calcular recomendações para muitos usuários diferentes, torna-se totalmente inviável apresentar cada caso individualmente neste trabalho.

Utilizando a interface gráfica criada, pode-se testar de forma mais eficiente as recomendações obtidas por diferentes usuários. Esta também é uma forma de validação do

uso da interface para os propósitos deste projeto, como uma ferramenta fácil e intuitiva para o usuário final.

Figura 9 – Tela inicial da ferramenta



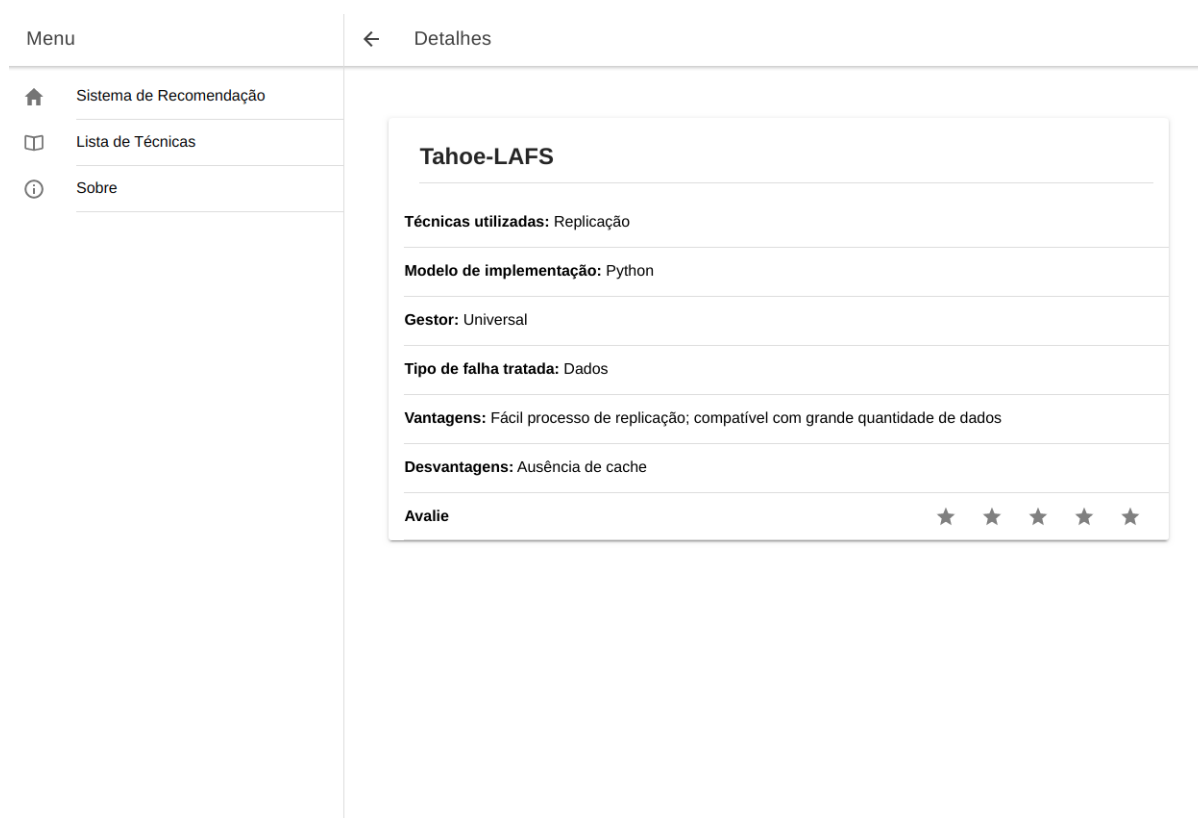
A Figura 9 mostra a interface inicial da ferramenta desenvolvida neste trabalho, mostrando seus aspectos principais de navegação, bem como o sistema de recomendação plenamente funcional. Para intuito de demonstração, foi selecionado o usuário de identificador 8 e as respectivas recomendações calculadas para tal. Pode-se observar que a aplicação apresenta os dados básicos das técnicas

O sistema de recomendação é acessado pelo *framework* Ionic através de um *request* HTTP com o identificador do usuário à aplicação Flask, que foi mantida em execução para permitir as recomendações. A aplicação Flask executa a cada requisição um novo cálculo de recomendação para exibição na tela inicial. Para manter a consistência dos dados exibidos e permitir reprodutibilidade dos resultados, também foi escolhida uma semente $seed = 0$ neste caso. Para os efeitos de demonstração também utilizaremos o arquivo gerado *userdata_5_10.csv* como base para mostrar os resultados.

Para evitar conflitos de compartilhamento de recursos entre páginas *web*, também foi utilizado um pacote extra para o *framework* Flask chamado de *flask-cors*, que implementa o CORS (*Cross Origin Resource Sharing*), uma medida que permite que um website acesse outros websites mesmo em domínios distintos.

Como se pôde observar pela Figura 9, as informações dispostas na tela inicial são informações simplificadas das técnicas recomendadas, com o conteúdo completo de cada técnica podendo ser visualizado clicando no card em questão como mostra a Figura 10.

Figura 10 – Informações disponíveis sobre a técnica Tahoe-LAFS



As informações estão dispostas de forma simples e legível para o usuário final, permitindo também que ele volte para a página anterior, e também permitindo a avaliação da técnica em questão para aprimorar as recomendações feitas pelo sistema.

A interface responsiva permite redimensionamento sem perda relevante de dados, como mostrado na figura 11, o que a torna apropriada também para dispositivos móveis.

São dispostas 10 recomendações na tela inicial, um número que pode ser alterado dentro do código fonte livremente. O sistema também pode mostrar todas as 52 técnicas cadastrada, como mostrado na figura 12.

Nas figuras 13 e 14 estão demonstradas as funcionalidades de filtragem por escrita da ferramenta. As filtrações podem ser feitas utilizando as principais características das técnicas: o nome, tipo de técnica, modelo de implementação e gestor.

Figura 11 – Tela inicial da ferramenta reduzida

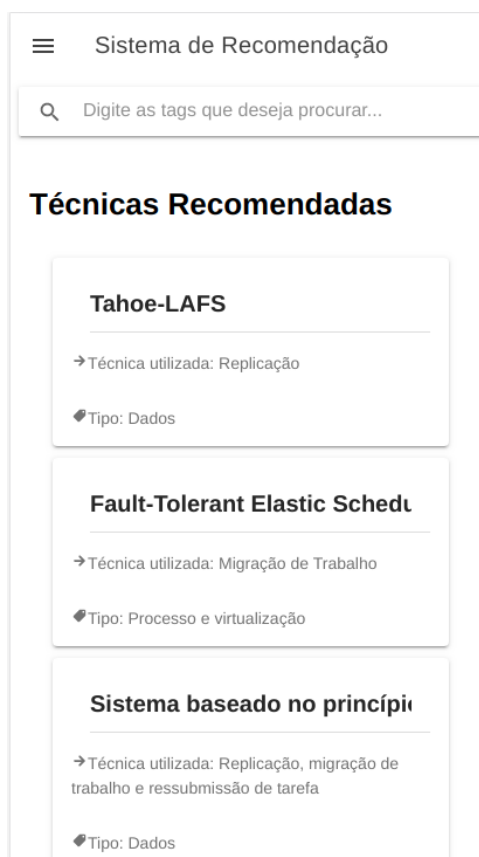


Figura 12 – Tela que permite visualizar todas as técnicas cadastradas

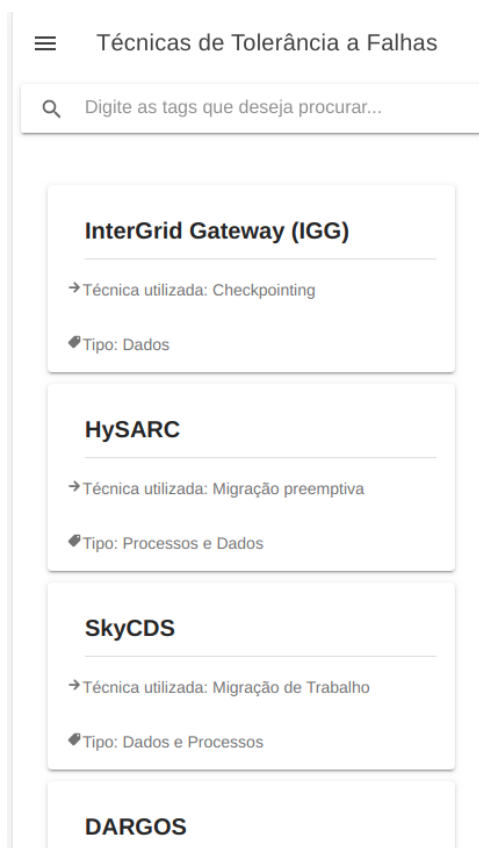


Figura 13 – Sequência de figuras que permitem demonstrar a filtragem por modelos de implementação e tipo de técnica, respectivamente

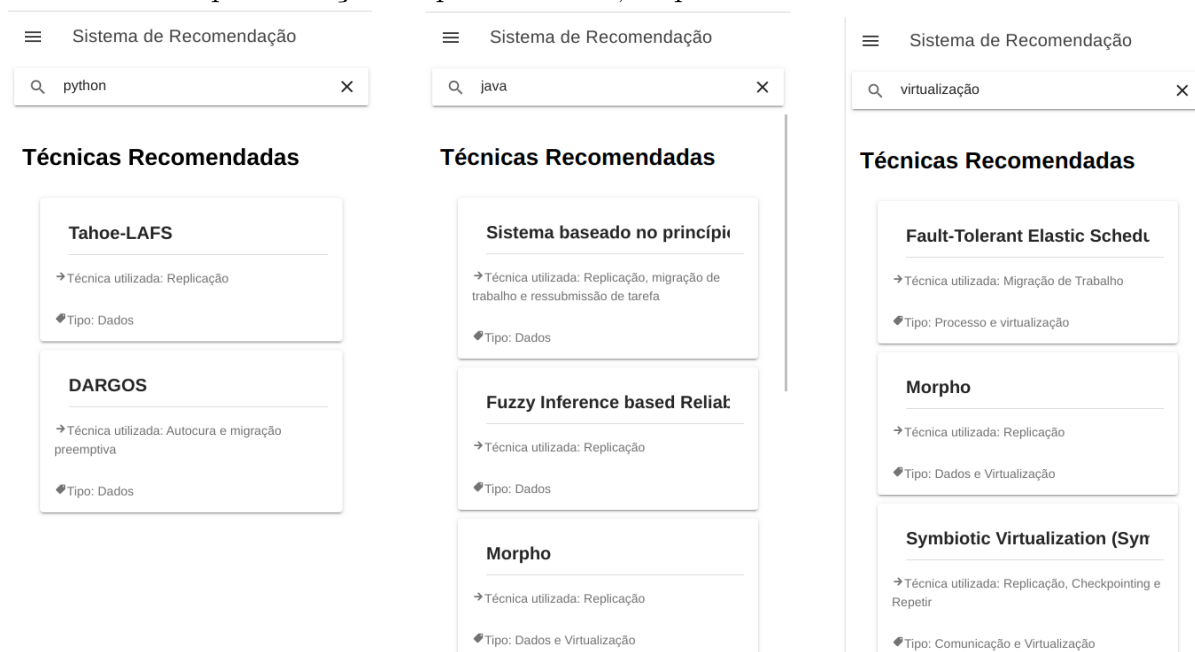
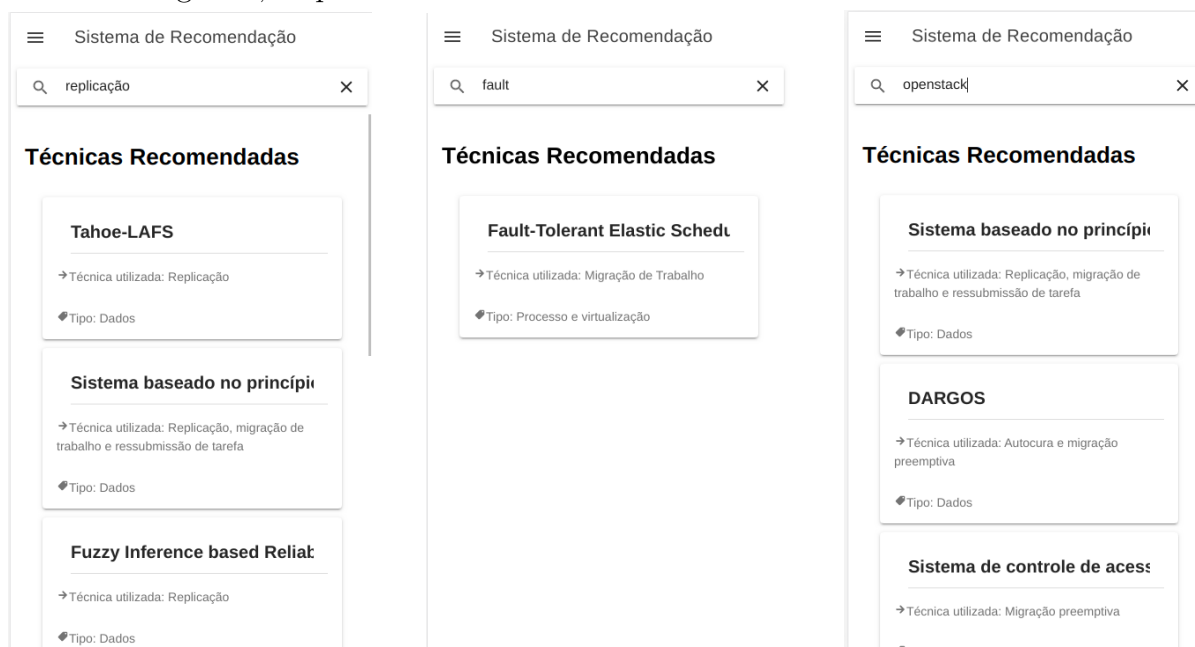


Figura 14 – Sequência de figuras que demonstram a filtragem por técnica utilizada, nome e gestor, respectivamente

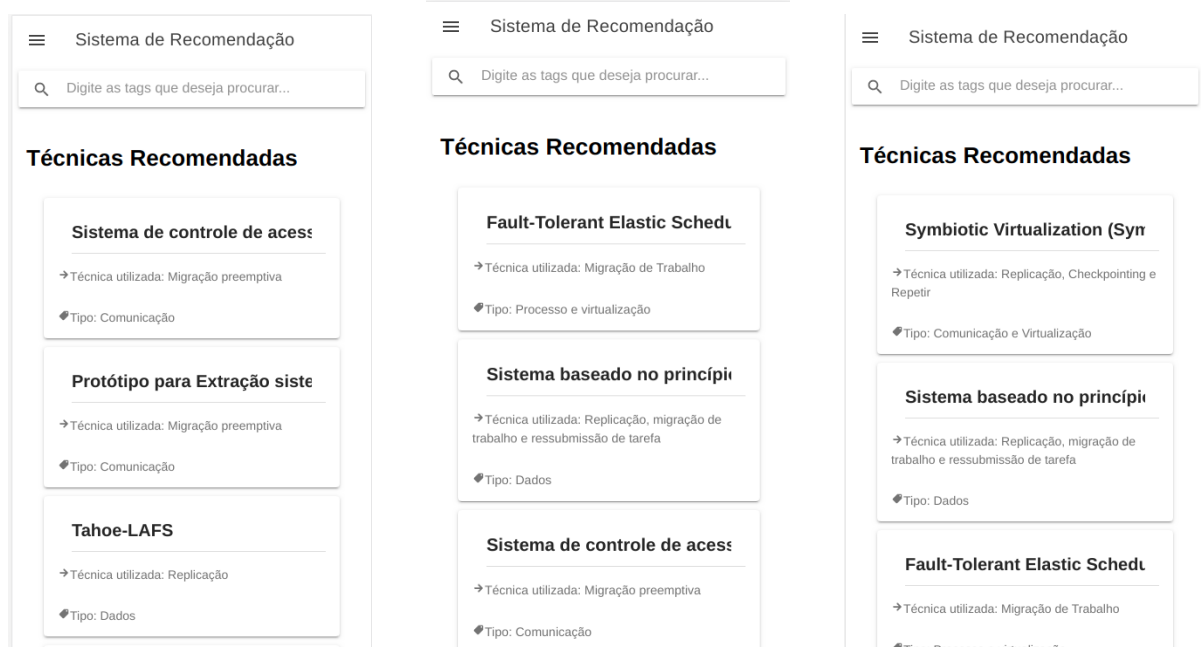


A técnica de filtragem também pode ser utilizada na lista completa de técnicas de tolerância a falhas, que possui funcionamento idêntico ao descrito anteriormente.

Para efeito de demonstração da mudança das técnicas recomendadas, na Figura 15 estão mostradas diferentes recomendações para diferentes usuários, sob as mesmas

condições apresentadas anteriormente.

Figura 15 – Sequência de figuras mostrando as diferentes recomendações dos usuários com identificador 1, 6 e 10, respectivamente



4.3 Considerações Finais

Os resultados obtidos dos testes foram bastante satisfatórios, considerando as limitações impostas pelas tecnologias escolhidas e todo o esforço empregado na realização deste projeto. De maneira geral, o sistema atende às expectativas traçadas no projeto inicial e permite uma expansão dos seus recursos a partir do que foi implementado, além de possuir uma flexibilidade para ser utilizado em outras áreas.

5 CONCLUSÃO

Neste trabalho foi desenvolvida uma ferramenta que tem como maior atrativo um sistema de recomendação, e que também permite a busca facilitada por meio de caracteres de itens relacionados no arquivo de troca de dados. Pode-se observar que um sistema de recomendação facilita a busca por itens diversos, e possibilita que o usuário se concentre apenas nos dados relevantes para sua pesquisa, sem que a ferramenta o atrapalhe a fazer seu trabalho.

Os resultados obtidos mostram a precisão do modelo do sistema de recomendação desenvolvido neste trabalho, e também comprovam a praticidade e fácil usabilidade do sistema em questão. Esse sistema pode ser livremente utilizado por administradores de sistemas sensíveis a falhas, e deve ser de grande ajuda para os usuários interessados em entender melhor cada tipo de técnica de tolerância a falhas disponível.

Devido à natureza do projeto, ele pode ser posteriormente expandido e também pode ser reaproveitado para outros fins, levando em consideração todas as adaptações feitas para o funcionamento deste projeto. O sistema está disponível para *download* gratuitamente, incluindo o código fonte, instruções para implantação e uma demonstração na página no GitHub (<https://github.com/sturdy-robot/sistema-final>), sob licença permissiva MIT.

REFERÊNCIAS

- ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, IEEE, n. 6, p. 734–749, 2005. Citado na página 35.
- ADOMAVICIUS, G.; TUZHILIN, A. Context-aware recommender systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 217–253. Citado na página 28.
- AMAZON.COM, INC. *Amazon EC2*. 2019. <<https://aws.amazon.com/pt/ec2/>>. Acessado em: 10 de Junho de 2019. Citado na página 19.
- AMAZON.COM, INC. *Amazon Web Services*. 2019. <<http://aws.amazon.com/>>. Citado na página 18.
- AMIN, Z.; SINGH, H.; SETHI, N. Review on fault tolerance techniques in cloud computing. *International Journal of Computer Applications*, Foundation of Computer Science, v. 116, n. 18, 2015. Citado 2 vezes nas páginas 16 e 22.
- ANDERSON, T.; LEE, P. *Fault tolerance: Principles and practice, 1981*. [S.l.]: Prentice Hall, 1981. Citado na página 20.
- ANDRADE, V. S. Uma taxonomia de sistemas de tolerância a falhas em ambientes de computação em nuvem open source. Universidade Estadual Paulista (UNESP), 2019. Citado 3 vezes nas páginas 35, 36 e 38.
- ATAALLAH, S. M.; NASSAR, S. M.; HEMAYED, E. E. Fault tolerance in cloud computing-survey. In: IEEE. *2015 11th International Computer Engineering Conference (ICENCO)*. [S.l.], 2015. p. 241–245. Citado 2 vezes nas páginas 16 e 23.
- BALA, A.; CHANA, I. Fault tolerance-challenges, techniques and implementation in cloud computing. *International Journal of Computer Science Issues (IJCSI)*, International Journal of Computer Science Issues (IJCSI), v. 9, n. 1, p. 288, 2012. Citado 3 vezes nas páginas 21, 23 e 24.
- BALABANOVIĆ, M.; SHOHAM, Y. Fab: content-based, collaborative recommendation. *Communications of the ACM*, ACM, v. 40, n. 3, p. 66–72, 1997. Citado 2 vezes nas páginas 30 e 35.
- BELL, R. M.; KOREN, Y. Lessons from the netflix prize challenge. *SiGKDD Explorations*, Citeseer, v. 9, n. 2, p. 75–79, 2007. Citado na página 26.
- BENNETT, J.; LANNING, S. et al. The netflix prize. In: NEW YORK, NY, USA. *Proceedings of KDD cup and workshop*. [S.l.], 2007. v. 2007, p. 35. Citado na página 26.
- BERMAN, S. J. et al. How cloud computing enables process and business model innovation. *Strategy & Leadership*, Emerald Group Publishing Limited, v. 40, n. 4, p. 27–35, 2012. Citado na página 12.
- BURKE, R. Hybrid web recommender systems. In: *The adaptive web*. [S.l.]: Springer, 2007. p. 377–408. Citado 4 vezes nas páginas 26, 29, 30 e 32.

- CHAPPELL, D. et al. Introducing windows azure. *Microsoft*, Dec, 2009. Citado na página 19.
- CHEN, W.; FONG, S. Social network collaborative filtering framework and online trust factors: A case study on facebook. In: IEEE. *2010 Fifth international conference on digital information management (ICDIM)*. [S.l.], 2010. p. 266–273. Citado na página 33.
- CHERAGHLOU, M. N.; KHADEM-ZADEH, A.; HAGHPARAST, M. A survey of fault tolerance architecture in cloud computing. *Journal of Network and Computer Applications*, Elsevier, v. 61, p. 81–92, 2016. Citado 3 vezes nas páginas 16, 23 e 24.
- DROPBOX, INC. *Dropbox*. 2019. <<https://www.dropbox.com/>>. Citado na página 20.
- DROPBOX, INC. *Dropbox Plans*. 2019. <https://www.dropbox.com/pt_BR/individual/plans-comparison>. Citado na página 20.
- FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: A survey. *Future generation computer systems*, Elsevier, v. 29, n. 1, p. 84–106, 2013. Citado na página 14.
- FUSARO, V. A. et al. Biomedical cloud computing with amazon web services. *PLoS computational biology*, Public Library of Science, v. 7, n. 8, p. e1002147, 2011. Citado na página 35.
- GOMEZ-URIBE, C. A.; HUNT, N. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, ACM, v. 6, n. 4, p. 13, 2016. Citado na página 33.
- GONG, C. et al. The characteristics of cloud computing. In: IEEE. *2010 39th International Conference on Parallel Processing Workshops*. [S.l.], 2010. p. 275–279. Citado 2 vezes nas páginas 15 e 16.
- GONG, J.; YUE, P.; ZHOU, H. Geoprocessing in the microsoft cloud computing platform-azure. In: CITESEER. *Proceedings the Joint Symposium of ISPRS Technical Commission IV & AutoCarto*. [S.l.], 2010. p. 6. Citado na página 34.
- GOOD, N. et al. Combining collaborative filtering with personal agents for better recommendations. *AAAI/IAAI*, v. 439, 1999. Citado 2 vezes nas páginas 26 e 28.
- GOOGLE, INC. *Google App Engine*. 2019. <<https://cloud.google.com/appengine/?hl=pt-br>>. Citado na página 20.
- HERLOCKER, J. L. et al. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 22, n. 1, p. 5–53, 2004. Citado 2 vezes nas páginas 27 e 29.
- HILL, W. et al. Recommending and evaluating choices in a virtual community of use. In: ACM PRESS/ADDISON-WESLEY PUBLISHING CO. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 1995. p. 194–201. Citado na página 35.
- HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. *Computer*, IEEE, v. 30, n. 4, p. 75–82, 1997. Citado 2 vezes nas páginas 24 e 25.

- HUG, N. *Surprise, a Python library for recommender systems*. 2017. <<http://surpriselib.com>>. Citado na página 42.
- ISINKAYE, F.; FOLAJIMI, Y.; OJOKOH, B. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, Elsevier, v. 16, n. 3, p. 261–273, 2015. Citado 2 vezes nas páginas 28 e 29.
- JACKSON, K. R. et al. Performance analysis of high performance computing applications on the amazon web services cloud. In: IEEE. *2010 IEEE second international conference on cloud computing technology and science*. [S.l.], 2010. p. 159–168. Citado na página 35.
- JHAWAR, R.; PIURI, V.; SANTAMBROGIO, M. Fault tolerance management in cloud computing: A system-level perspective. *IEEE Systems Journal*, IEEE, v. 7, n. 2, p. 288–297, 2012. Citado 2 vezes nas páginas 16 e 23.
- KHAN, Z. H. *Cloud Computing: A Guide for IT Leaders*. [S.l.]: ZAZAD Solutions Inc, 2018. Citado 2 vezes nas páginas 14 e 18.
- KOHLI, M. et al. Implementing machine learning in radiology practice and research. *American journal of roentgenology*, Am Roentgen Ray Soc, v. 208, n. 4, p. 754–760, 2017. Citado na página 35.
- KOREN, Y. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, v. 81, n. 2009, p. 1–10, 2009. Citado na página 26.
- LAM, X. N. et al. Addressing cold-start problem in recommendation systems. In: ACM. *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. [S.l.], 2008. p. 208–211. Citado na página 40.
- LAPRIE, J.-C. Dependable computing and fault-tolerance. *Digest of Papers FTCS-15*, p. 2–11, 1985. Citado na página 21.
- LEE, D. D.; SEUNG, H. S. Algorithms for non-negative matrix factorization. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2001. p. 556–562. Citado na página 47.
- LIKA, B.; KOLOMVATSOS, K.; HADJIEFTHYMIADES, S. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, Elsevier, v. 41, n. 4, p. 2065–2073, 2014. Citado na página 40.
- LOHR, S. Google and ibm join in ‘cloud computing’research. *New York Times*, v. 8, 2007. Citado na página 12.
- MADDURI, R. K. et al. Experiences building globus genomics: a next-generation sequencing analysis service using galaxy, globus, and amazon web services. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 26, n. 13, p. 2266–2279, 2014. Citado na página 35.
- MARSTON, S. et al. Cloud computing—the business perspective. *Decision support systems*, Elsevier, v. 51, n. 1, p. 176–189, 2011. Citado na página 14.
- MAZHELIS, O.; TYRVÄINEN, P. Economic aspects of hybrid cloud infrastructure: User organization perspective. *Information Systems Frontiers*, Springer, v. 14, n. 4, p. 845–869, 2012. Citado na página 14.

- MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National ... , 2011. Citado na página 15.
- MICROSOFT, INC. *Microsoft Azure*. 2019. <<https://azure.microsoft.com/>>. Citado na página 20.
- MROZEK, D.; GOSK, P.; MAŁYSIAK-MROZEK, B. Scaling ab initio predictions of 3d protein structures in microsoft azure cloud. *Journal of Grid Computing*, Springer, v. 13, n. 4, p. 561–585, 2015. Citado na página 35.
- NATELLA, R.; COTRONEO, D.; MADEIRA, H. S. Assessing dependability with software fault injection: A survey. *ACM Computing Surveys (CSUR)*, ACM, v. 48, n. 3, p. 44, 2016. Citado na página 24.
- NETFLIX, INC. *Netflix*. 2019. <<https://www.netflix.com/>>. Citado na página 26.
- PATRA, P. K.; SINGH, H.; SINGH, G. Fault tolerance techniques and comparative implementation in cloud computing. *International Journal of Computer Applications*, Foundation of Computer Science, v. 64, n. 14, 2013. Citado 3 vezes nas páginas 22, 23 e 24.
- PAZZANI, M. J.; BILLSUS, D. Content-based recommendation systems. In: *The adaptive web*. [S.l.]: Springer, 2007. p. 325–341. Citado na página 30.
- PETERSEN, J. L.; WHEATLEY, M.; KELLNER-ROGERS, M. The y2k problem: Social chaos or social transformation? *The Futurist*, World Future Society, v. 32, n. 7, p. 21, 1998. Citado na página 21.
- RANDELL, B. System structure for software fault tolerance. *Ieee transactions on software engineering*, IEEE, n. 2, p. 220–232, 1975. Citado na página 21.
- RESNICK, P. et al. Grouplens: an open architecture for collaborative filtering of netnews. In: ACM. *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. [S.l.], 1994. p. 175–186. Citado na página 35.
- RESNICK, P.; VARIAN, H. R. Recommender systems. *Communications of the ACM*, Association for Computing Machinery, Inc., v. 40, n. 3, p. 56–59, 1997. Citado 3 vezes nas páginas 26, 28 e 29.
- RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 1–35. Citado 3 vezes nas páginas 27, 28 e 31.
- SARWAR, B. M. et al. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In: *Proceedings of the fifth international conference on computer and information technology*. [S.l.: s.n.], 2002. v. 1, p. 291–324. Citado 2 vezes nas páginas 29 e 31.
- SCHAFER, J. B.; KONSTAN, J.; RIEDL, J. Recommender systems in e-commerce. In: ACM. *Proceedings of the 1st ACM conference on Electronic commerce*. [S.l.], 1999. p. 158–166. Citado 2 vezes nas páginas 29 e 33.

- SCHAFER, J. B.; KONSTAN, J. A.; RIEDL, J. E-commerce recommendation applications. *Data mining and knowledge discovery*, Springer, v. 5, n. 1-2, p. 115–153, 2001. Citado 2 vezes nas páginas 32 e 33.
- SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. In: *Recommender systems handbook*. [S.l.]: Springer, 2011. p. 257–297. Citado na página 31.
- SHAPIRA, B.; ROKACH, L.; FREILIKHMAN, S. Facebook single and cross domain data for recommendation systems. *User Modeling and User-Adapted Interaction*, Springer, v. 23, n. 2-3, p. 211–247, 2013. Citado na página 32.
- SMITH, D. B.; MÜLLER, H. A.; TILLEY, S. R. *The Year 2000 Problem: Issues and Implications*. [S.l.], 1997. Citado na página 21.
- TERVEEN, L.; HILL, W. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium*, Addison-Wesley Glen View, IL, v. 1, n. 2001, p. 487–509, 2001. Citado na página 29.
- TSAI, W.-T.; SUN, X.; BALASOORIYA, J. Service-oriented cloud computing architecture. In: IEEE. *2010 seventh international conference on information technology: new generations*. [S.l.], 2010. p. 684–689. Citado na página 12.
- WIKIMEDIA COMMONS. *Diagrama do funcionamento da Filtragem Colaborativa*. 2019. <https://upload.wikimedia.org/wikipedia/commons/2/27/Stumble_Upon_Collaborative_Filtering_Diagram.png>. Citado na página 29.
- WIKIMEDIA COMMONS. *Modelos de sistemas de recomendação*. 2019. <https://upload.wikimedia.org/wikipedia/commons/2/2c/Collaborative_Filtering_in_Recommender_Systems.jpg>. Citado na página 28.
- XENSOURCE, INC. *Xen*. 2019. <www.xensource.com>. Citado na página 19.
- ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, Springer, v. 1, n. 1, p. 7–18, 2010. Citado 3 vezes nas páginas 17, 18 e 19.
- ZHOU, R.; KHEMMARAT, S.; GAO, L. The impact of youtube recommendation system on video views. In: ACM. *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. [S.l.], 2010. p. 404–410. Citado na página 32.