UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

OTÁVIO SILVA GOES

SISTEMA DE RECOMENDAÇÃO DE ALERTAS CIBERNÉTICOS: PROJETO E DESENVOLVIMENTO

CAMPO MOURÃO

2021

OTÁVIO SILVA GOES

SISTEMA DE RECOMENDAÇÃO DE ALERTAS CIBERNÉTICOS: PROJETO E DESENVOLVIMENTO

Recommendation System for Cyber Security Alerts: Design and Development

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Rodrigo Campiolo

CAMPO MOURÃO

2021

OTÁVIO SILVA GOES

SISTEMA DE RECOMENDAÇÃO DE ALERTAS CIBERNÉTICOS: PROJETO E DESENVOLVIMENTO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 18/agosto/2021

Luiz Arthur Feitosa Dos Santos
Doutorado
UTFPR

Marcos Silvano Almeida
Doutorado
UTFPR

Rodrigo Campiolo
Doutorado
UTFPR

CAMPO MOURÃO

2021

RESUMO

O crescente número de ameaças e vulnerabilidades sendo reportadas na Internet dificulta para administradores de redes identifiquem alertas relevantes, considerando a infraestrutura tecnológica de suas organizações. Logo, evidenciar tais alertas possibilitaria prover respostas antecipadas ou mais rápidas a essas ameaças e vulnerabilidades. Desta forma, neste trabalho foi projetado e desenvolvido um sistema de recomendação para alertas cibernéticos que facilite o acesso a alertas que evidenciam vulnerabilidades ou ameaças à segurança de sistemas, que considera as preferências de alertas especificados pelos usuários do sistema. Neste contexto, foram desenvolvidos o modelo de recomendação e uma arquitetura escalável visando a manutenção e desempenho do sistema. O sistema foi avaliado em testes de estresse e o modelo de recomendação por meio de uma base de dados sintética. Como resultados, o sistema suportou uma alta carga de requisições e a implementação do modelo se mostrou viável para a recomendação de alertas, apesar de necessitar ainda de ajustes e mais testes. Verificou-se que o sistema e o modelo de recomendação conseguem lidar com alta carga de alertas e recomendar alertas de interesse para o administrador.

Palavras-chaves: cibersegurança. sistemas distribuídos. sistemas recomendação.

ABSTRACT

The growing number of threats and vulnerabilities being reported on the Internet makes it difficult for network administrators to identify relevant alerts, considering their organizations' technology infrastructure. Therefore, highlighting such alerts would make it possible to provide early or faster responses to these threats. Thus, this work aims to develop a recommendation system for cyber alerts that facilitates access to alerts that evidence vulnerabilities or threats to the security of systems used by each user and/or organization. In this context, the recommendation model and a scalable architecture were developed for the maintenance and performance of the system. The system was evaluated in stress tests and the recommendation model through a synthetic database. As results, the system supported a high load of requests and the implementation of the model proved feasible for the recommendation of alerts, although it still needed adjustments and further tests. It was found that the system and recommendation model can handle high load of alerts and recommend alerts of interest to the administrator.

Keywords: cybersecurity. distributed systems. recommendation systems.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CISA: *Certified Information Systems Auditor*. 14
CVE: *Common Vulnerabilities and Exposures*. 9, 15, 20

DoS: *Denial of Service*. 13

HTTP: *Hypertext Transfer Protocol*. 23

IDS: *Intrusion Detection System*. 13
IODEF: *The Incident Object Description Exchange Format*. 14

NVD: *National Vulnerability Database*. 15

OSINT: *Open Source Intelligence*. 14, 15
OVAL: *Open Vulnerability and Assessment Language*. 14

REST: *Representational State Transfer*. 23, 29
RPC: *Remote Procedure Call*. 23, 29, 34

SGBD: Sistema de Gerenciamento de Banco de Dados. 23, 28, 33, 34
SQL: *Structured Query Language*. 12
SR: Sistemas de Recomendação. 11, 15–18, 20
STIX: *Structured Threat Information eXpression*. 14

TIC: Tecnologia da Informação e Comunicação. 9

SUMÁRIO

1 INTRODUÇÃO

A evolução dos meios de comunicação e informação possibilita ganho de produtividade em todas as esferas e, em particular, a Internet tem papel essencial neste contexto. O número de usuários e serviços na Internet cresce todos os dias. O aumento da disponibilidade de serviços e aplicações na Internet é acompanhado de riscos às infraestruturas e sistemas, que podem emergir de comportamentos internos, vulnerabilidades, códigos maliciosos ou de entidades externas aos sistemas e organizações.

A cibersegurança é a área da computação que visa manter a segurança da informação nos meios computacionais, como na Internet. Aplicações devem prover a confidencialidade, integridade e disponibilidade dos sistemas e das informações. Para tal, é necessário implantar processos e mecanismos para proteger usuários e organizações de ameaças como ataques, acessos não autorizados, vulnerabilidades de serviços, entre outros.

Contudo, com a alta demanda de informações empregada aos sistemas é possível observar um aumento nos ataques relacionados a sistemas na Internet, também conhecidos como ciberataques. Muitos desses ataques ocorrem em razão de alguma vulnerabilidade explorada em determinado sistema, pondo em risco a segurança do mesmo. Essas vulnerabilidades se originam por falhas na implementação ou implantação de sistemas computacionais.

Neste contexto, administradores de sistemas devem estar atualizados a respeito do surgimento de novas ameaças e vulnerabilidades. Também seria interessante que compartilhassem informações de ameaças e vulnerabilidades que identificam em suas organizações. O acesso a informações relevantes considerando as preferências do administrador, isto é, infraestrutura de software e hardware da organização, em conjunto com a colaboração com outros administradores, pode facilitar e agilizar a identificação das ameaças potenciais.

1.1 Problema de Pesquisa

A quantidade de sistemas e serviços nas organizações propiciam um aumento nas ameaças que podem surgir para estes mesmos aparatos, o que pode dificultar o trabalho de administradores de rede a atuar preventivamente para combater possíveis danos aos sistemas e serviços gerenciados por eles. Por isso, é exigido que o administrador tenha atualização constante, além de conhecimento a respeito de novas ameaças e vulnerabilidades da infraestrutura computacional que administra. Essas informações podem ser obtidas em alertas de fabricantes de *software*/*hardware*, notícias em *sites* especializados ou publicações em redes sociais.

Se por um lado há muitas fontes de informação na Internet provendo alertas, por outro, isso acaba por gerar uma sobrecarga ao administrador com dados não desejáveis e que demandariam muito tempo e esforço para serem filtrados. Além disso, nem sempre um alerta apresenta as soluções para lidar com o problema. Assim, o desenvolvimento de mecanismos que auxiliassem o administrador a

atentar-se somente a alertas importantes e de interesse para sua infraestrutura proveriam aumento na produtividade e na segurança dos sistemas supervisionados por ele.

Apesar da existência de bases de alertas, como o *Common Vulnerabilities and Exposures* (CVE) (CVE, 2020), que contém dados de vulnerabilidades conhecidas de cibersegurança, nem sempre são suficientes para informar de todos os ataques. Essas fontes não cobrem todos os sistemas e muitas vezes demoram alguns dias para serem atualizadas. Neste intervalo, notícias de vulnerabilidades e ameaças já foram publicadas em outras mídias, além disso, nem todos os alertas gerados evidenciam ameaças reais a sistemas computacionais.

1.2 Objetivos

Objetiva-se neste trabalho o desenvolvimento de um sistema de recomendação para filtrar e evidenciar alertas de cibersegurança considerando a infraestrutura de software e hardware específica das organizações. Dessa forma, alertas relevantes podem ser identificados mais rapidamente propiciando aos administradores aplicarem medidas proativas para a proteção de suas organizações.

Como objetivos específicos, têm-se:

- Aprimorar o modelo de recomendação híbrido para alertas de cibersegurança proposto por Esposte et al. (2016).
- Projetar uma arquitetura escalável para suportar o modelo de recomendação.
- Implementar um sistema de recomendação voltado à cibersegurança.
- Avaliar o modelo de recomendação.

1.3 Justificativa

Os sistemas de recomendação auxiliam na filtragem de conteúdo, tornando a experiência do usuário mais dinâmica e personalizada (RICCI et al., 2011). Com a grande quantidade de conteúdo gerado a todo momento, empresas como Netflix[1] e Amazon[2] têm grandes sistemas de recomendação que tornam a experiência do usuário ainda mais personalizada, auxiliando na escolha de itens de interesse.

Portanto, quando há uma grande quantidade de dados sendo gerada a todo momento, examina-los manualmente pode ser muito difícil. Por isso, justifica-se o desenvolvimento de um sistema capaz de filtrar informações sobre vulnerabilidades e ataques contra infraestruturas de Tecnologia da Informação e Comunicação (TIC), para que administradores possam receber informações acerca de conteúdos ou tecnologias de acordo com o perfil da sua organização. Assim, facilitando a resolução ou proporcionando um tempo essencial para que alguma medida preventiva seja aplicada e, por consequência, protegendo os sistemas e infraestrutura de ameaças conhecidas.

[1] https://www.netflix.com/br Acessado em 30/08/2021
[2] https://www.amazon.com.br/ Acessado em 30/08/2021

1.4 Contribuições

Com o desenvolvimento do sistema proposto pretende-se:

- Melhorar a atividade de identificação e correção para problemas de segurança por administradores de redes/sistemas.
- Prover colaboração entre administradores de diferentes organizações por meio de avaliações, comentários de soluções e recomendações de alertas.
- Uma implementação funcional de um sistema de recomendação para cibersegurança.

1.5 Organização do Texto

O Capítulo 2 apresenta o referencial teórico do trabalho, abordando os conceitos-chave em cibersegurança e sistemas de recomendação, e os trabalhos relacionados. O Capítulo 3 apresenta o projeto e o desenvolvimento do sistema de recomendação para alertas cibernéticos. O Capítulo 4 descreve os resultados e discussões a respeito da avaliação do sistema criado. O Capítulo 5 apresenta as considerações finais e trabalhos futuros.

2 REFERENCIAL TEÓRICO

Esse capítulo aborda os conceitos para a compreensão e desenvolvimento de um sistema de recomendação de alertas cibernéticos. Também evidencia os trabalhos bases e relacionados que fundamentaram a presente pesquisa. A Seção 2.1 aborda os conceitos associados à cibersegurança. São apresentadas definições de cibersegurança, ameaças, alertas, *open source intelligence* e sistemas de compartilhamento de alertas. A Seção 2.2 aborda definições e características de Sistemas de Recomendação (SR), como aplicações, tipos de recomendação e avaliação. A Seção 2.3 apresenta os trabalhos relacionados. A Seção 2.4 apresenta o modelo de recomendação base para este trabalho.

2.1 Cibersegurança

Cibersegurança é definida como a prevenção contra danos, uso não-autorizado, exploração e a restauração, se necessário, de informações eletrônicas, além de sistemas computacionais e suas informações, a fim de fortalecer a confidencialidade, integridade e disponibilidade desses sistemas (HOGAN; NEWTON, 2015).

A confidencialidade consiste em manter os recursos acessíveis somente para os indivíduos, processos ou entidades que possuem as credenciais necessárias, evitando obtenção não autorizada de dados/recursos.

A integridade é um conceito que define como os recursos devem se manter durante o ciclo de vida da informação. Um recurso íntegro é aquele que se mantém preciso e consistente sem mudanças inesperadas ou involuntárias.

A disponibilidade de dados especifica a garantia de acesso de determinada informação. O projeto, modelo e arquitetura de um sistema que provê este princípio deve garantir que os recursos sempre estarão disponíveis no momento em que forem requisitados.

Segundo a norma 27002 (ISO, 2005), implementar um conjunto de controles, incluindo políticas, processos, procedimentos, estruturas organizacionais e funções de *hardware* e *software*, é necessário para garantir a segurança da informação e dos sistemas em organizações. Todo o controle deve ser estabelecido, implementado, monitorado, revisado e melhorado quando necessário.

Caso um sistema não tenha sido preparado ou sua segurança tenha sido negligenciada, o sistema pode ser prejudicado por ameaças que se beneficiam de vulnerabilidades existentes no conjunto. A cibersegurança é a área da computação responsável pela prevenção de ameaças à segurança, mitigação de ataques e recuperação de sistemas afetados (STALLINGS; BROWN, 2013).

2.1.1 Vulnerabilidades

Uma vulnerabilidade é definida como uma falha em um sistema de informação, implementação, controles internos ou em procedimentos na segurança do sistema que podem ser exploradas ou

desencadeada por uma fonte/agente de ameaça (FIPS-USA, 2006). As situações em que há recursos do sistemas que podem ser acessados sem a autorização necessária ou serviços potencialmente frágeis ou desatualizados, são exemplos de possíveis vulnerabilidades. O gerenciamento de vulnerabilidades é um processo interminável de identificação, classificação, mitigação e recuperação de vulnerabilidades.

A probabilidade de exploração define o risco de uma vulnerabilidade para um sistema. Uma vulnerabilidade com risco baixo para um sistema tem menos chances de ser usada em um ataque, bem como, as vulnerabilidades que podem ser exploradas mais facilmente possuem um risco elevado e podem prejudicar mais o sistema.

Dentre as diversas vulnerabilidades que podem existir em sistemas computacionais, destacam-se:

- Senhas fracas: senhas formadas por nomes, apelidos ou palavras comuns (como as encontradas em dicionários) são frequentemente usadas por serem de fácil memorização.
- Acesso irrestrito: a falta de restrições a conteúdos sensíveis em sistemas pode acarretar no uso/acesso indevido a estes recursos.
- Bibliotecas externas: o uso de códigos externos ao sistema pode acarretar na adição de códigos maliciosos (*malwares*) ocultos em bibliotecas de terceiros.
- Má configuração: um sistema mal configurado permite que recursos sensíveis ou detalhes de implementações sejam expostos a terceiros.
- Falta de validação: a falta de verificação das entradas pode gerar complicações, como injeção de *Structured Query Language* (SQL) - ato de alterar o comportamento espero da requisição em sistema de gerenciamento de banco de dados - ou dados inconsistentes.
- Estouro de *buffer*: Consiste na não verificação do limite de capacidade de determinada região de memória, que quando ultrapassado pode acarretar em perdas ou corrupções de dados legítimos temporariamente alocados em outros *buffers* pelo sistema ou ainda prover acesso indevido ao sistema.

Em geral, as vulnerabilidades são originadas por falhas no projeto, uso de tecnologias desatualizadas ou métodos de restrições brandos. Estas falhas de segurança de um sistema podem nunca serem notadas até o momento que forem exploradas por algum atacante.

As vulnerabilidades de sistemas de comunicação que operam na Internet, por exemplo, não se limitam somente à infraestrutura de software. Os ambientes físicos podem conter vulnerabilidades que podem comprometer todo o sistema, tais como o livre acesso a uma sala de servidores ou sistemas de que gerenciam o banco de dados, ambientes onde estão sujeitos a incêndios ou alagamentos podendo comprometer a disponibilidade de informações de todo o sistema ou um usuário que armazena suas senhas escritas em "*post-its*"na sua mesa para que não sejam esquecidas.

Independente da vulnerabilidade, abre-se brechas para que um sistema possa sofrer ataques, além da possibilidade de perda, roubo ou corrupção de recursos importantes.

2.1.2 Ameaças Cibernéticas

Quando um sistema pode sofrer algum dano por conta de uma vulnerabilidade ou ataque, isso é chamado de ameaça à segurança ou ameaça cibernética (ciberameaça). O roubo/sequestro de informações, *Denial of Service* (DoS) ou *malwares* são exemplos de ameaças que podem acometer um sistema de informações. Também se refere a possibilidade de ataque bem sucedido que possui o intuito de ganhar acesso não-autorizado, danificar, corromper ou roubar informações sensíveis. As ameaças à segurança podem ser intencionais ou ocasionais.

As ameaças intencionais são criadas por indivíduos ou sistemas que operam sobre uma determinada vulnerabilidade, tendo em vista prejudicar de alguma forma o sistema atacado. Esse tipo de ameaça é o mais comum de ser noticiado. Geralmente são criados software que auxiliam o intruso no seu objetivo, visto que há situações em que o acesso aos recursos de *hardware* e *software* deste tipo são chamados de *malwares* ou *software* maliciosos. As principais motivações dos atacantes estão relacionadas a vantagens financeiras, coleta de informações confidenciais, ativismos sociais (hacktivismo) ou vandalismo (CERT, 2020).

Software malicioso é aquele que causa danos a sistemas computacionais, os tipos de aplicações deste tipo incluem:

- Spywares: programas que coletam informações sensíveis a respeito do sistema hospedeiro.
- Worms: programas que podem se multiplicar e causar danos a computadores sem a ação humana.
- Vírus: programas similares aos *worms*, no entanto precisam ser explicitamente executados.
- Rootkits: possibilitam ao atacante executar operações restritas em sistemas operacionais, podem se acobertar por outras ferramentas para que sejam despercebidos.

Os softwares maliciosos muitas vezes são instalados em um sistema ao usar versões extraoficiais de aplicações, *cracks* de licenças ou periféricos infectados.

As ameaças ocasionais são causadas por situações involuntárias que podem acometer sistemas, como quedas de energia ou mal funcionamento não detectado de um ou mais dispositivos. Problemas físicos também podem prejudicar os sistemas de informação, como o acesso não autorizado a salas importantes ou pane de *hardware*.

2.1.3 Alertas

As ameaças cibernéticas podem causar comprometimento de sistemas e sérios problemas para os administradores de sistemas. Por sua vez, os alertas ou notificações de segurança são as notícias ou avisos sobre estas adversidades. Segundo Campiolo (2016), um alerta de segurança constitui uma forma de antecipar possíveis ameaças que podem comprometer a segurança da infraestrutura de rede.

Os alertas podem ser gerados por sensores de redes tradicionais, como *Intrusion Detection System* (IDS), *firewalls*, *honeypots*, entre outros, ou de fontes externas, como *sites* especializados, mídias sociais e correspondência eletrônica. A ideia de compartilhar alertas possibilita reações antecipadas

ou mais rápidas a ameaças. No entanto, o compartilhamento esbarra em algumas dificuldades, como padronizações, relevância e contexto dos alertas para uma organização.

Plataformas como o MITRE (MITRE, 2020) reúnem diversos tipos de especificações e formatos colaborativos em esforços comunitários para o gerenciamento de vulnerabilidades, garantias de *software*, gestão de ativos, relatórios empresariais, proteção contra *malwares* e compartilhamento de informações sobre ameaças.

Sites Web tais como o *HackerNews* (YCOMBINATOR, 2020) geram informações a respeito de vulnerabilidades ou novos ataques que possam ocorrer em sistemas na Internet. Essas fontes possibilitam gerar bases em tempo real de novos alertas. Outro exemplo é o *CyberTwitter* (MITTAL et al., 2016), um software que coleta informações do Twitter relacionadas a vulnerabilidades e exibe para os usuários. Tanto alertas obtidos de *sites* Web, mídias sociais ou outras fontes, relacionam-se com a ideia de *Open Source Intelligence* (OSINT).

A principal dificuldade em lidar com múltiplas fontes de dados é a estruturação empregada por cada ferramenta/serviço. Os dados gerados por *firewall* se limitam às informações do tráfego de rede e/ou informações a respeito dos serviços com base nas portas requisitadas. Existem esforços, por exemplo do CERT (USCERT, 2020), que especificam parâmetros bem definidos para descrever e ampliar o entendimento de um alerta.

Também há outras padronizações que descrevem alertas de cibersegurança, como *The Incident Object Description Exchange Format* (IODEF) (DANYLIW et al., 2007). O uso de padronizações facilita o compartilhamento de ameaças à segurança entre organizações, indivíduos ou software. Padronizações como *Open Vulnerability and Assessment Language* (OVAL) (OVAL, 2020), uma linguagem que define vulnerabilidades e problemas de configuração, *Structured Threat Information eXpression* (STIX) (STIXPROJECT, 2020), uma maneira de representar informações de ameaças com uma estrutura definida, são formas protocoladas de compartilhamento de alertas computacionais.

As principais características encontradas nas padronizações incluem dados como: atributo identificador, empresa/organização afetada, vulnerabilidades exploradas, infraestrutura e padrões de ataque usados para relacionar múltiplas ameaças.

Para que haja conhecimento das ameaças que atingem as ferramentas de softwares, foram criados sistemas que possibilitam o compartilhamento de informações relacionadas a vulnerabilidades e/ou ameaças, denominados de sistemas de compartilhamento de alertas.

2.1.4 Sistemas de Compartilhamento de Alertas

Os sistemas de compartilhamento de alertas relacionados à cibersegurança são ferramentas responsáveis por agrupar informações que podem ser úteis a determinados usuários, informando-os sobre possíveis ameaças ou vulnerabilidades ligadas aos seus sistemas ou infraestruturas.

Sistemas como *Certified Information Systems Auditor* (CISA)(USCERT, 2020) fornecem vasta quantidade de informações sobre vulnerabilidades e alertas cibernéticos, além de ferramentas como

o *framework* de compartilhamento colaborativo de informações (JASPER, 2017), que pode auxiliar ainda mais a comunidade de cibersegurança.

A plataforma CVE(CVE, 2020) possui listas públicas de vulnerabilidades e falhas de segurança em sistemas computacionais. Para cada item listado na plataforma, é atribuído um número de identificação *ID*. Além do CVE, O *National Vulnerability Database* (NVD) é uma base de dados dos Estados Unidos que armazena dados a respeito de vulnerabilidades, referências de *checklists* de segurança, falhas de *software* relacionadas à segurança, configurações incorretas e métricas de impacto.

O grande problema desses sistemas, do ponto de vista dos usuários, é justamente o grande volume de informações geradas pelos mesmos, problema esse que pode ser solucionado com uma ferramenta que possa filtrar alertas e suas informações, para que seja entregue ao usuário somente o que seja útil a ele.

Há alguns padrões para compartilhamentos de alertas, tais como os apresentados na Seção 2.1.3 e os especificados por Danyliw et al. (2007), no qual se trata de uma definição que propicia a obtenção de parâmetros bem definidos para cada alerta cibernético. No entanto, os alertas de fontes de dados não estruturados são difíceis de automatizar a extração e classificação das informações relevantes para compartilhamento segundo as necessidades de um usuário ou uma organização.

2.1.5 OSINT

A obtenção e compartilhamento de informações a partir de fontes públicas ou abertas é denominado de Inteligência de Fontes Abertas, do inglês, OSINT. Livros, revistas, jornais, publicações, mapas, mídias de áudio/vídeo, a Internet, rádio, televisão e bancos de dados (bibliotecas, universidades, governo) são exemplos da ampla gama de fontes públicas ou abertas existentes (HRIBAR et al., 2014).

Como citado por Glassman e Kang (2012), OSINT possui o propósito de reunir, codificar seletivamente e combinar de forma transparente informações para o desenvolvimento de ferramentas que possibilitem a resolução de problemas.

No contexto de cibersegurança, as informações obtidas de fontes abertas possibilitam o conhecimento de vulnerabilidades que possam afetar os sistemas computacionais que possam afligir os administradores e organizações.

Contudo, nem todas as informações que podem ser encontradas em fontes públicas são de origem pública, como cita Hribar et al. (2014). Exemplos incluídos nesse trabalho: informações de empregados de empresas, regras de conduta não classificada de forças militares informados por um soldado, entre outros.

2.2 Sistemas de Recomendação

Os SR são ferramentas de *software* que auxiliam usuários no uso de alguma plataforma ou sistema, almejando melhorar a experiência e tornando o processo de decisão mais simplificado. Os SR's são

amplamente usados em aplicações de entretenimento em que há uma grande quantidade de itens, bem como em sistemas de compra/venda *on-line* chamados *e-commerces*.

Um SR necessita de informações do usuário para que seja possível operar e recomendar conteúdos. Essas informações podem ser passadas diretamente, o usuário informa os dados ao sistema, ou indiretamente, o sistema verifica as ações do usuário e define características, como uma página visitada ou um *post* curtido. Estas informações indicam as preferências do usuário e permite que o sistema possa recomendar itens relacionados.

O conteúdo recomendado por SR é chamado de **item** e as interações entre um usuário e um SR são chamadas de **transações**, as quais geram mais dados a respeito do usuário e auxiliam no processo de recomendação pelos algoritmos empregados no sistema.

2.2.1 Aplicações

Geralmente os esforços em SR têm ênfase na prática e aplicações comerciais, logo é comum o uso dos sistemas de recomendação nas seguintes áreas:

- Entretenimento: recomendação de filmes, músicas e jogos
- E-commerce: recomendação de produtos para compra, como *smartphones*, computadores ou livros.
- Conteúdo: recomendação de páginas web, jornais ou documentos.

Exemplos tradicionais de aplicações/empresas que usam sistemas de recomendação são:

- Amazon, no qual os produtos são recomendados aos usuários com base nos itens já adquiridos, avaliados e também vistos.
- Netflix, que gera muitos dados sobre filmes e séries, o sistema de recomendação é usado para que os usuários tenham acessos aos itens mais especificamente relacionados aos seus gostos pessoais e conteúdos exclusivos.
- Spotify, o conteúdo fornecido pela plataforma é amplo e usa diversos tipos de recomendação para músicas e *podcasts*.

Com o uso de SR por grandes organizações como as supracitadas, o uso e estudo acerca de sistemas que proporcionem uma experiência melhor ao usuário por meio de melhores recomendações tem aumentado proporcionalmente ao crescimento de conteúdo disponível na rede.

2.2.2 Tipos de recomendação

Existem diversas abordagens que são empregadas para se obter os melhores resultados quando o assunto é recomendação. Estas abordagens definem como o sistema opera e quais dados podem ser usados no processo de indicação de determinado item.

Os sistemas de recomendação podem ser elencados em dois paradigmas, sendo eles: **recomendação personalizada**, em que os itens indicados possuem alguma relação com as interações

do usuário no sistema e as preferências do mesmo, o outro paradigma é o de **recomendação não personalizada**, que pode sugerir itens com base em atributos não relacionados ao usuário, mas que pode servir para a maioria deles, por exemplo, uma lista das músicas mais tocadas.

Quanto aos métodos usados em sistemas de recomendação, destacam-se o baseado em conteúdo, filtragem colaborativa e a abordagem híbrida, que são definidas a seguir.

Baseado em Conteúdo

Os SR baseados em conteúdo usam os itens que o usuário já interagiu para distinguir quais itens devem ser recomendados. Cada alerta terá os seus atributos avaliados, de forma que o grau de relevância de cada item para os usuários seja definido. Por exemplo, um usuário pode avaliar bem um filme que o gênero seja ação, assim o sistema de recomendação pode indicar filmes com gênero ação.

Esta abordagem é muito usada, porém pode ser problemática em situações em que um item não tenha sido avaliado ainda, logo o sistema não possui os dados necessários para fazer uma boa recomendação.

Filtragem Colaborativa

Este tipo de recomendação usa os dados de outros usuários com o mesmo perfil de consumo para predizer quais itens têm mais relevância e poderão ser recomendados. A filtragem de usuários com os mesmos gostos é feita com base na similaridade das avaliações dos usuários. Esta abordagem é considerada a mais popular e também a mais usada em sistemas de recomendação (RICCI et al., 2011).

Um dos principais problemas deste método se dá quando um usuário recente ainda não possui avaliações suficientes para que o sistema o relacione com outros usuários, logo a recomendação acaba obtendo uma baixa taxa de acerto.

Sistemas de Recomendação Híbridos

Neste tipo de recomendação o sistema pode usar mais de uma abordagem para que seja possível fazer melhores indicações, resolvendo problemas específicos de algum método com características de um outro. Além disso, outros atributos podem ser usados para personalizar ainda mais os resultados obtidos pelo recomendador.

Apesar das vantagens de se combinar dois ou mais métodos, o desempenho do sistema pode ser prejudicado, devido ao aumento do volume de operações para a obtenção das recomendações, visto que é necessário fazer mais avaliações e com técnicas diferentes.

2.2.3 Avaliação de SRs

Como os sistemas de SR serão avaliados é um assunto que requer muita atenção, pois pode tornar a experiência do usuário ruim. Um sistema que prediz bem os resultados mas que demora muito para fazer isso não é bom e certamente o usuário sofrerá por conta dessa característica. É necessário

que os sistemas possam ser ajustados nos mínimos detalhes, para que o consumo de recursos sejam mínimo do mesmo modo que o sistema faça boas recomendações.

Algumas métricas usadas na avaliação são desempenho na execução e o grau de acertos alcançada pelo sistema. Estes estudos podem ser demorados, pois exigem uso extenso e prolongado do sistema e deve contar com a participação dos usuários nas avaliações.

2.3 Trabalhos Relacionados

Este trabalho relaciona-se diretamente com o desenvolvido por Esposte et al. (2016) e Campiolo (2016), no qual um modelo de SR para recomendação de alertas cibernéticos foi desenvolvido e avaliado. Esposte et al. (2016) desenvolveram uma pesquisa a respeito dos padrões de usuários do modelo ou sistema, além de especificar a fundamentação matemática importante para o sistema. O modelo citado faz parte da tese desenvolvida por Campiolo (2016), na qual alertas antecipados a respeito de ameaças de segurança em sistemas computacionais foram colhidos a partir de base de dados não estruturados. Campiolo (2016) avaliou o modelo em uma base de dados sintética com recomendações de alertas geradas a partir de alertas do CVE. Ambos trabalhos são bases para o desenvolvimento desta monografia, são explicados em detalhes na Seção 2.4.

A área de sistemas de compartilhamento de alertas cibernéticos está relacionada a este projeto, porém como alguns dos sistemas deste tipo não produzem itens que são específicos para cada tipo de usuário como quando usado algum método de recomendação, assim como este projeto. Alguns dos sistemas de compartilhamento de alertas podem servir de foonte de dados relacionados a ameaças cibernéticas, tais como (JASPER, 2017) e (ZHAO; WHITE, 2012).

Alguns trabalhos seguem o mesmo princípio de informar ao usuário de alertas que possam acometê-lo, como alertas extraídos de *tweets* (MITTAL et al., 2016), em que o usuário será informado sobre *tweets* que servem como alertas para a infraestrutura usada pelo usuário.

2.4 Modelos de Recomendação de Alertas Cibernéticos

Esta seção descreve brevemente o sistema de recomendações base para este projeto, desenvolvido por Esposte et al. (2016). O trabalho base desta monografia tem o objetivo de criar um modelo de recomendação para alertas cibernéticos, alertas estes coletados a partir de fontes de dados externas. O desenvolvimento de uma ferramenta de recomendação tem em foco os usuários com o perfil de administradores de rede, cujo o interesse tecnológico é voltado à área de cibersegurança.

A coleta de informações relacionadas aos atributos e informações que os administradores estariam dispostos a oferecer ao sistema foi baseada em uma pesquisa com algumas dezenas de possíveis usuários. A pesquisa foi dividida em 4 (quatro) fases, em que cada uma delas serviu para identificar características específicas dos usuários que poderiam vir a usar o sistema:

- Perfil: identificar características gerais dos usuários
- Cibersegurança: quantificar o grau de proficiência do usuário na área.

- Interesses: verificar considerações com relação ao modelo criado, identificando os principais elementos que o modelo deveria conter.
- Casos de uso: simular potenciais interações dos usuários com um sistema de recomendação hipotético.

As questões e resultados da pesquisa podem ser acessados no repositório do Gitlab[1]

2.4.1 Especificação do Modelo

Com base na pesquisa realizada, foi definida as estruturas que representam os usuários do sistema, bem como os itens a serem recomendados, que neste caso se trata de alertas cibernéticos. As preferências dos usuários podem ser definidas segundo os sistemas operacionais e os tipos de ataques relacionados aos alertas. Assim como os potenciais dados dos usuários do sistema, a estrutura dos alertas foi definida visando os atributos mais relevantes. Os atributos definidos como importantes foram: (1) título, (2) fonte da informação, (3) nível de criticidade do alerta, (4) alvos relacionados (*hardware*/*software*) e (5) palavras-chave.

Os itens serão recomendados com base nos índices atribuídos a eles, os índices definem a qual ferramenta ou software referenciado pelo alerta, sendo estes os valores que representam algumas das transações do sistema (definir como crítico, curtir, rejeitar). Assim, a similaridade entre itens e entre usuários é definida com base nos índices e atributos recomendados e que o usuário interagiu. Os valores obtidos são usados nas abordagens aplicadas pelo trabalho, as quais forma de recomendação baseado em conteúdo, filtragem colaborativa e o modelo que faz a junção das abordagens citadas.

2.4.2 Transações

Cada interação do usuário com o sistema é chamada de **transação**, foram definidos no modelo 5 (cinco) tipos de transações possíveis ao usuário em relação aos itens do modelo.

- Visualizar: indica que o usuário teve acesso ao item, para que sirva de recurso para definir itens mais acessados. Além disso, possibilita distinguir quais alertas não podem ser recomendados a este mesmo usuário.
- Categorizar: permite o usuário definir *chaves* atributos que definem aos itens. Assim é possível identificar qual tecnologia o item está referenciando.
- Definir nível crítico: registra a opinião do usuário com relação a urgência do item. O nível de criticidade sobrepõe os demais níveis de avaliação dos alertas.
- Curtir: define uma relação positiva entre o usuário e o item, gerando ainda mais dados que poderão ser usados no processo de recomendação dos itens.
- Rejeitar: o contrário de **curtir**, ou seja, diminui as chances de um item ruim ser recomendado para mais usuários.

[1] https://gitlab.com/konsilo/survey/blob/master/results.pdf - Acessado em 5 de Junho de 2020

2.4.3 Avaliação

O método de avaliação *offline* foi usado para analisar o comportamento do SR. A base de dados experimental usada no experimento foi obtida da plataforma MovieLens (GROUPLENS, 2020). Essa base de dados contém informações a respeito de filmes e as respectivas avaliações de usuários nos mesmos. Campiolo (2016) usou uma base sintética de alertas gerada a partir de alertas do CVE.

As avaliações feitas nesse trabalho são separadas para cada tipo de recomendação usada no modelo. Foram definidos parâmetros de avaliação dos métodos que se baseiam nos falsos positivos obtidos nas avaliações.

- Precisão: define a habilidade de o sistema recomendar só o que é relevante para o usuário.
- Revocação: propensão do sistema recomendar tudo que é relevante para o usuário.

Ao final da avaliação desenvolvida foi observado um bom resultado de precisão, onde a variância dos dados coletados foi menor com o método de filtragem colaborativa. Já os dados de revocação o método de filtragem obteve melhores resultados com a média de propensão de 25,5% em comparação com o método de filtragem baseado em conteúdo que obteve uma média de 7%.

2.5 Considerações

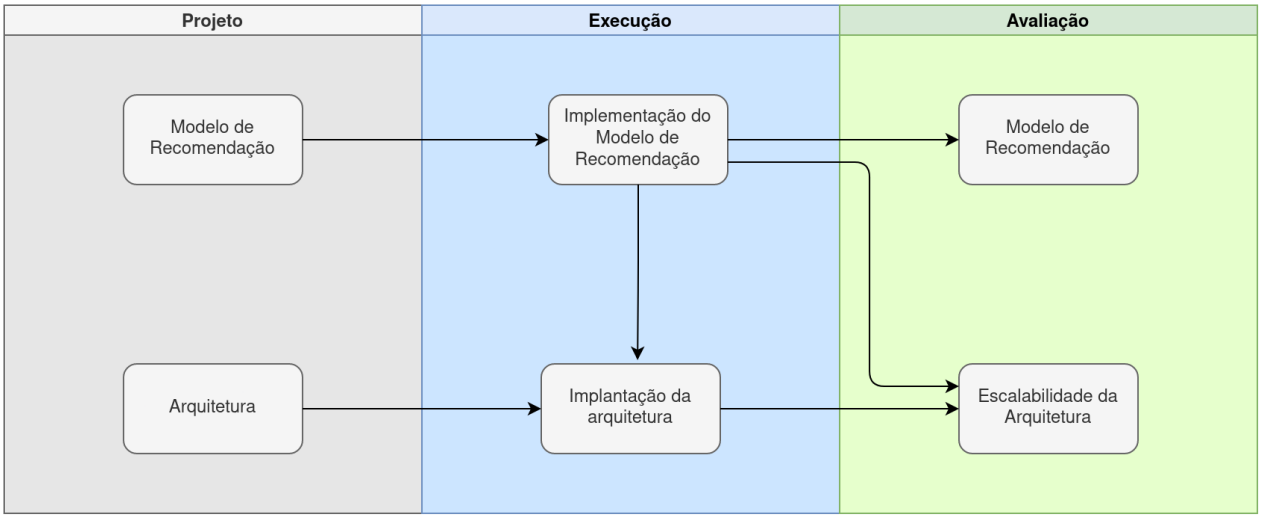
Neste capítulo foram abordados os temas que fazem parte da base para o desenvolvimento deste trabalho, em especial, os conceitos e características dos alertas cibernéticos. Também foram introduzidos conceitos-chave sobre e o modelo base para a construção do sistema proposto nesta monografia. O próximo capítulo apresenta o projeto e o desenvolvimento do sistema de recomendação para a filtragem de alertas cibernéticos.

3 PROJETO E DESENVOLVIMENTO DO SISTEMA

Este capítulo descreve as etapas para o desenvolvimento do sistema de recomendação que visa propiciar aos administradores os alertas cibernéticos mais relevantes, facilitando a filtragem de conteúdo em conjuntos de dados volumosos de alertas cibernéticos. Como principal contribuição deste trabalho, espera-se prover um sistema que auxilie os administradores de sistemas computacionais reagir antecipadamente ou mais rápido a ameaças cibernéticas fazendo uso dos alertas recomendados pelo sistema.

O desenvolvimento do sistema foi dividido em três principais etapas (Figura 3.1): Projeto, Execução e Avaliação. A etapa de Projeto aborda a modelagem e especificação da arquitetura do sistema, e a extensão e aperfeiçoamento do modelo de recomendação. A etapa de Execução corresponde à implementação do sistema de recomendação e da implantação na arquitetura proposta. A etapa de Avaliação corresponde à avaliação do modelo de recomendação e da escalabilidade arquitetura.

Figura 3.1. Fluxograma de desenvolvimento do projeto



Fonte: Autoria própria

As seções seguintes detalham cada uma dessas etapas.

3.1 Arquitetura

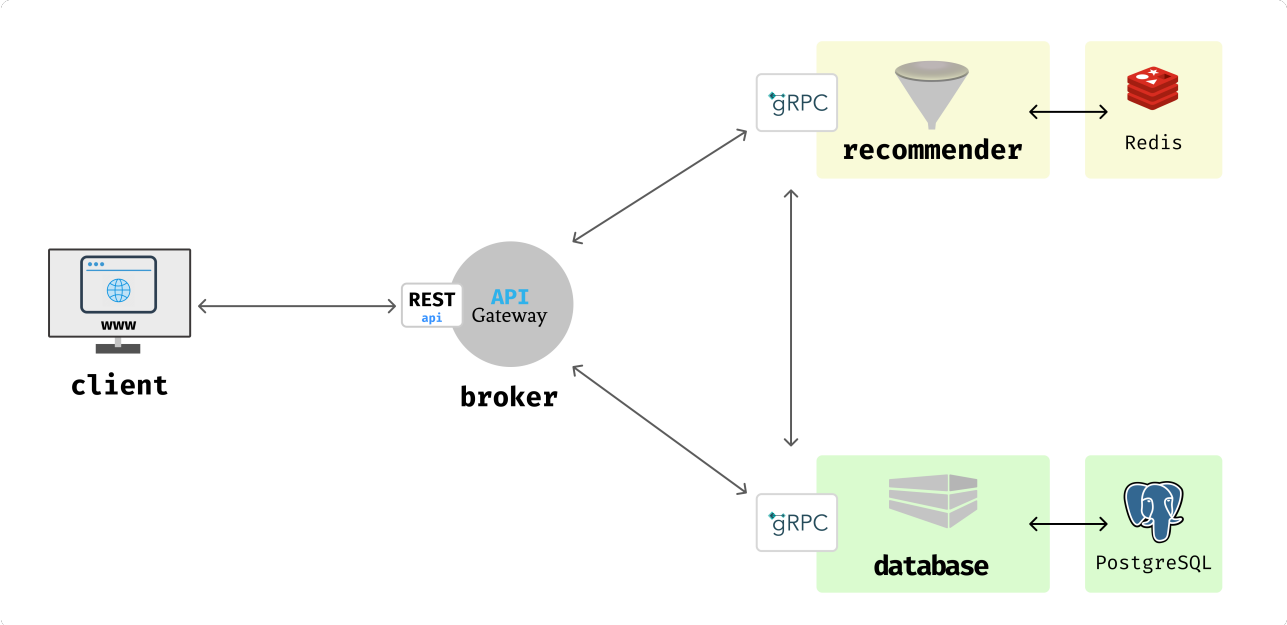
No projeto e desenvolvimento de sistemas distribuídos há dois principais tipos de decisões arquiteturais: arquitetura de sistema e arquitetura de software. A arquitetura de sistemas descreve como os elementos do sistema são dispostos em um sistema distribuído e a topologia das conexões entre os

componentes. A arquitetura de software especifica a organização lógica dos elementos de software, que define como os componentes interagem entre si e suas estruturas (código e/ou dados).

A arquitetura de software para o desenvolvimento deste projeto foi a arquitetura baseada em microsserviços, que consiste na divisão de responsabilidades e funções entre múltiplos componentes de software no sistema. Os motivos para a escolha deste tipo de organização estrutural do sistema foi o isolamento de responsabilidades/falhas, a possibilidade da combinação de tecnologias e escalabilidade.

A arquitetura de sistema desenvolvida segue o padrão API-Gateway, estilo arquitetural propicia a abstração de responsabilidades não funcionais do sistema como autenticação do usuário e validação das informações. Este estilo amplia a capacidade de monitoramento, balanceamento de carga e o reuso de funcionalidades comuns aos microsserviços. Como evidenciado por Venugopal (2017), os serviços abstraídos por um único componente possuem bases de código menores, facilitando a manutenção e extensão dos componentes. A Figura 3.2 apresenta a arquitetura do sistema de recomendação.

Figura 3.2. Arquitetura dos componentes do sistema



Fonte: Autoria própria

3.1.1 Serviços

O sistema foi divido em 6 (seis) partes como visto na Figura 3.2, dos quais dois são ferramentas para o armazenamento de dados dos alertas e as interações dos usuários com os mesmos (Seção 3.2.2). Para que os padrões de desenvolvimento de uma arquitetura baseada em microsserviços fosse seguidos, foi necessário fazer a divisão de responsabilidades entre todos os serviços no sistema.

Cliente (*client*)

Este serviço é responsável pela interface de interação do usuário com o sistema. Todo o gerenciamento de informações do usuário e as interações com os alertas recomendados é feito por componentes visuais, facilitando assim o uso da ferramenta.

Com apenas um ponto de conexão com o sistema, este serviço está limitado a comunicar-se com o intermediador (*broker*) por meio de uma interface *Representational State Transfer* (REST), que define um padrão de comunicação entre componentes de software usando o protocolo *Hypertext Transfer Protocol* (HTTP).

Intermediador (*broker*)

O serviço que intercepta todas as requisições feitas pelo (*cliente*) e as redireciona para os serviços específicos. Esse componente implementa as funcionalidades comuns aos outros serviços, proporcionando assim uma diminuição da base de código nos outros componentes. Suas principais funcionalidade são a autenticação dos usuários do sistema e validação das requisições.

Além de receber requisições do (*cliente*), este serviço se comunica e redireciona requisições para os serviços de (*banco de dados*) e o (*recomendador*). Com o acesso às informações das requisições é possível verificar a autenticidade dos usuários, estruturar os conteúdos e validar as requisições.

Para a comunicação com este serviço foi escolhido uma interface de comunicação REST, por manter simples a integração com as ferramentas mais comuns para o desenvolvimento de interfaces de interação com usuários, como aplicações Web e móveis.

O intermediador mantém contato com o demais serviços do sistema por meio do protocolo *Remote Procedure Call* (RPC) provido pelo *framework* gRPC[1]. Este método de comunicação necessita de interfaces de comunicação bem definidas e conhecidas por todos os *peers*.

Gerenciador do Banco de Dados (*database*)

Este serviço é responsável pela interação com o Sistema de Gerenciamento de Banco de Dados (SGBD) relacional *PostgreSQL*[2]. O principal motivo para a criação deste serviço foi a estruturação de operações de acesso ao SGBD de forma modularizada e centralizada.

A comunicação com os outros elementos da arquitetura é feita por RPC. Além disso é o único dos elementos que possui a conexão direta com o SGBD. Devido a falta de um estado próprio, a clusterização horizontal pode ser implantada para que haja o aumento da escalabilidade em situações de um número elevado de requisições.

[1] https://grpc.io/ Acessado em 11/07/2020
[2] https://www.postgresql.org/ Acessado em 11/07/2020

Recomendador (*recommender*)

Este serviço é responsável pela avaliação e filtragem de informações para as recomendações de alertas. O modelo de recomendação dos alertas é implementado somente neste serviço e recebe requisições somente do **intermediador**.

Além do **intermediador**, este serviço mantém a comunicação com o gerenciador de dados estruturados em memória *Redis*[3], o qual armazena os resultados primários do modelo de recomendação, permitindo o rápido acesso e reduzindo o consumo de recursos para as requisições de recomendação.

3.1.2 Escalabilidade

Para alcançar um alto grau de escalabilidade, a arquitetura do sistema foi projetada para que os serviços sejam disponibilizados (*deploy*), testados e mantidos separadamente. Além da estruturação de todo o sistema, os componentes que necessitarem de aumento de desempenho ou memória podem ser atualizados individualmente.

Por conta da maior necessidade computacional, o serviço de **recomendação** pode ser alterado individualmente para que seja escalado horizontalmente com a divisão de tarefas entre múltiplos computadores, também chamado de *cluster*.

3.2 Modelo de Recomendação

O modelo de recomendação deste trabalho se trata de uma atualização do modelo criado por Esposte et al. (2016) e Campiolo (2016). Além da adição de novos métodos de recomendação, este novo modelo traz modificações nos métodos já existentes, tais alterações levam em consideração as possíveis avaliações de cada usuário com relação aos alertas, tornando possível recomendar alertas já vistos pelos usuários, ao contrário de não incluir os alertas já avaliados, como era feito anteriormente. Tais modificações são evidenciados na Seção 3.2.3.

3.2.1 Itens

Os itens recomendados pelo sistema são representações de alertas cibernéticos e possuem os principais atributos:

- *id*: identificador único do alerta;
- *produto*: define o possível alvo cibernético descrito no alerta;
- *provedor*: empresa/organização responsável pelo produto alvo do alerta;
- *descrição*: define mais detalhes a respeito do alerta gerado.

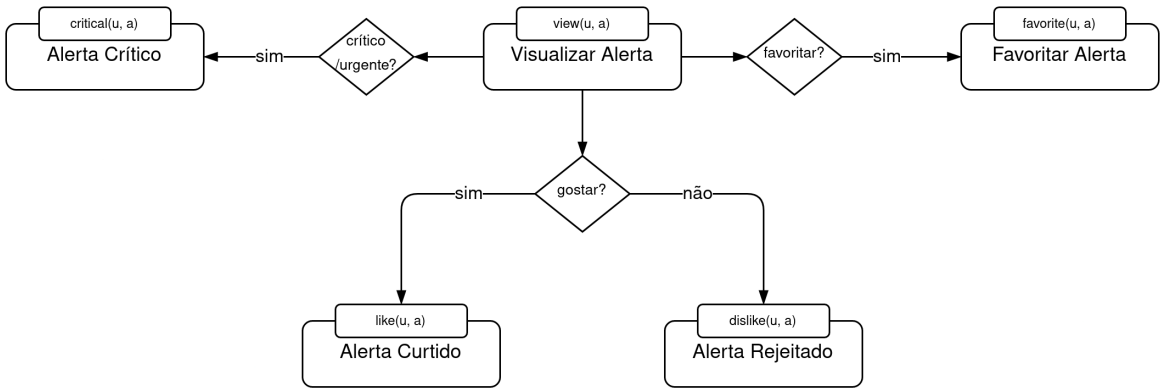
Dentre os atributos de cada item, os mais relevantes para a recomendação são o ***product*** e ***provider***. Estes atributos foram usados para definir o quão relevante o alerta é para o usuário, definido pelo peso de cada atributos no conjunto de preferências especificados pelo usuário.

[3] https://redis.io/ Acessado em 11/07/2020

3.2.2 Transações

As interações básicas com os itens do sistema estão mapeadas na Figura 3.3 e descritas a seguir.

Figura 3.3. Transações do usuário no sistema



Fonte: Autoria própria

Visualização

Transação executada no instante em que o usuário obtém mais informações a respeito de um item. Pode ser usada para estimar os alertas mais populares ou que obtiveram mais interações. Além disso, os itens já visualizados por um usuário têm uma chance menor de serem recomendados.

Criticar

Transação que assinala a urgência daquele item no ponto de vista do usuário do sistema. Atribui um peso maior quando comparado às outras formas de avaliação de um item pelo usuário.

Aprovar

Transação realizada quando o usuário interage de forma positiva com um item, de forma que o conteúdo deste é relevante para o usuário.

Desaprovar

Transação que registra uma interação negativa entre o usuário e um item, de forma que destaca uma insatisfação por parte do usuário com o conteúdo de um alerta cibernético.

Favoritar

Transação que define que o item passará a ser mostrado entre os primeiros em uma lista de todos os itens visualizados/avaliados pelo usuário. Essa transação não é considerada na recomendação de novos itens.

3.2.3 Métodos de Recomendação

Definidos os itens do sistema e as interações do usuário com os mesmos, estão dispostos a seguir os métodos e equações implementados no sistema para a recomendação dos itens. Para classificar os itens no sistema, foram especificadas equações que definirão o grau de importância de um determinado item no sistema ou para um usuário do mesmo.

Algumas das equações foram retiradas do trabalho base (ESPOSTE et al., 2016), além de algumas modificações e adição de novas equações para a classificação dos itens. A Equação 3.1 define a importância de um item.

$$score(i) = \frac{(\bar{v} \times \bar{r}) + (v_i \times r_i)}{\bar{v} + |r_i|} \tag{3.1}$$

onde $\bar{v}$ é a média do número de votos para todos os alertas, sendo esse referentes a *likes*, *dislikes* e críticos; $\bar{r}$ é a média das avaliações de todos os alertas; v_i é o total de votos para o alerta i; e r_i é a classificação de todas a avaliações daquele alerta, definida pela Equação 3.2.

$$r_i = 4c_i + l_i - 2d_i \tag{3.2}$$

A Equação 3.2 define o valor de classificação de um item, baseado nas avaliações do mesmo, onde c_i é a total de votos que categorizam o alerta como crítico; l_i é o número total de *likes* do alerta e d_i é a quantidade de *dislikes*.

Além das equações retiradas do trabalho base, neste projeto foi adicionado uma forma de classificar os itens que já foram vistos/avaliados pelo usuário, tais itens não eram considerados nas recomendações do trabalho anterior. A Equação 3.3 define a relevância de um item para usuário levando em consideração uma possível interação.

$$relevance(u, i) = \begin{cases} 2, & r_i \notin R_u \\ relevance_{r_i} \end{cases} \tag{3.3}$$

onde R_u é o conjunto de avaliações de um usuário; r_i é uma possível avaliação do item pelo usuário e $relevance_{r_i}$ é o grau de relevância da avaliação do usuário para aquele item, considerando a importância de um voto para aquele usuário. Dentre as avaliações de um usuário, a quantidade de votos (*likes* e *dislikes*) são levados em consideração, assim caso um usuário possua mais avaliações com *like* em relação *dislike*, as avaliações negativas deste usuário terão um peso maior, e vice-versa.

Recomendação Baseada em Conteúdo

A Equação 3.4 é usada para recomendar itens usando recomendação baseada em conteúdo.

$$content\_based(u,i) = score(i) \times (tag\_weight(u,i) + 1) \times relevance(u,i) \tag{3.4}$$

onde o $score(i)$ é definido pela Equação 3.1; $tag\_weight(u,i)$ define o quão importante um determinado alerta é para um usuário, de forma que os atributos que caracterizam um alerta estejam relacionados às preferências do usuário; e $relevance(u,i)$ é o resultado da Equação 3.3 do usuário e o item.

Filtragem Colaborativa

A filtragem colaborativa considera as semelhanças entre as preferências dos usuários, tal similaridade é defina pela Equação 3.5. Para classificar os alertas neste método de recomendação foi usada a Equação 3.6.

$$J(u,v) = \frac{|U \cap V|}{|U \cup V|} = \frac{|U \cap V|}{|U| + |V| - |U \cap V|} \tag{3.5}$$

onde U é o conjunto de preferências do usuário u; e V é o conjunto de preferencias do usuário v.

$$collaborative(u,i) = \frac{\sum_{v \in V} tag\_weight(v,i) \times J(u,v)}{|\sum_{v \in V} J(u,v)|} \times relevance(u,i) \tag{3.6}$$

onde V é o conjunto de todos os usuários do sistema; ; e $tag\_weight(v,i)$ define o grau de importância do alerta para tal usuário.

Recomendação Não-Personalizada (*Top-N*)

Por se tratar de um tipo de recomendação que não depende das informações de usuário, este método usa somente a Equação 3.1, ordenando de forma que o itens com um *score* mais alto têm mais relevância, selecionando assim os *N* primeiros.

3.3 Implementação do Sistema

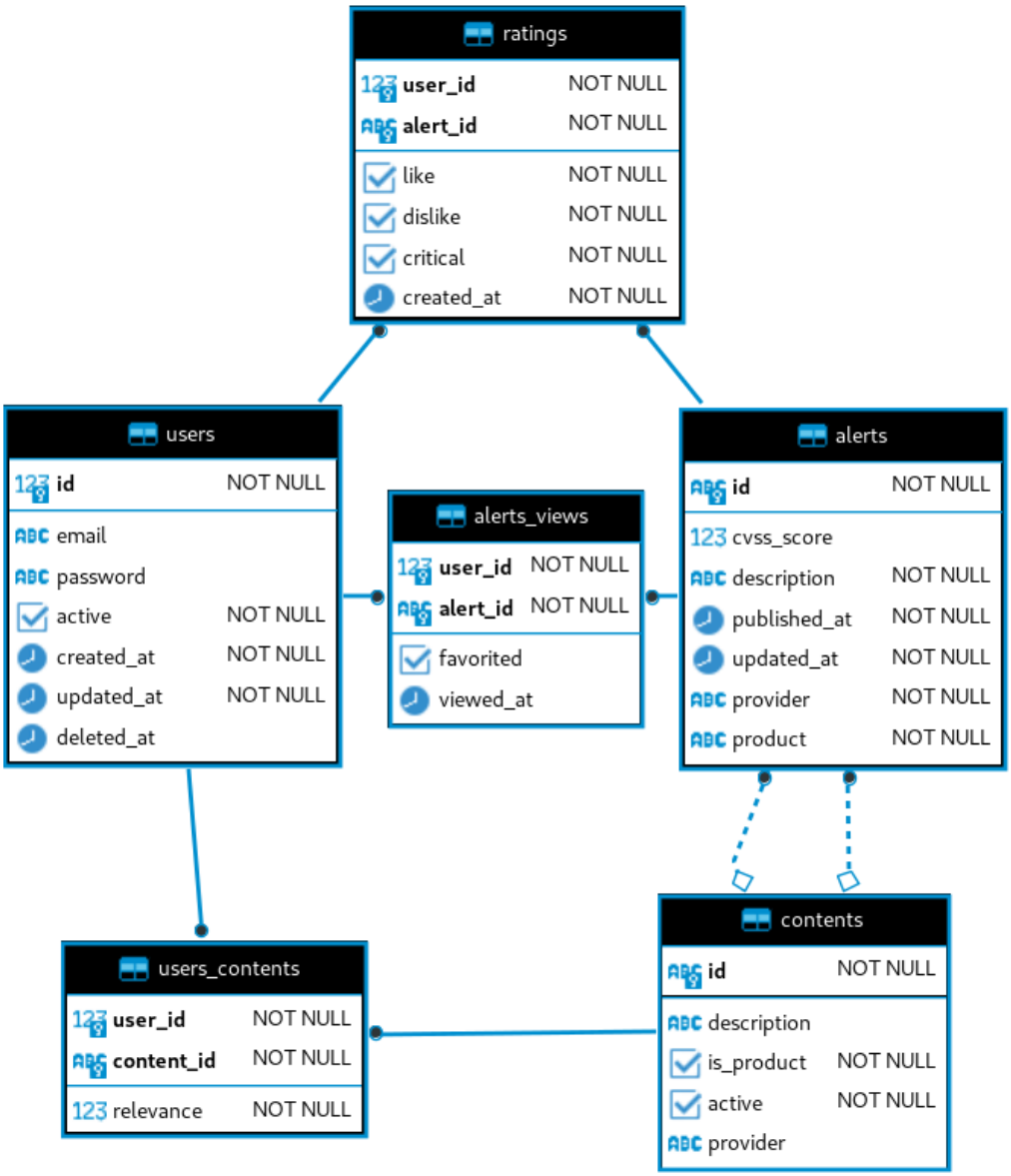
Nesta seção serão descritos os principais componentes e estruturas desenvolvidos, além da descrição das ferramentas e métodos de comunicação utilizadas para a criação do sistema.

3.3.1 Persistência de Dados

O SGBD escolhido foi o *PostgreSQL*, o uso de um sistema de gerenciamento de banco de dados estruturados proporciona maior segurança e confiabilidade com relação aos dados armazenados, visto que não é possível inserir/obter dados que não estão de acordo com a estrutura definida na criação das entidades.

Outro critério de escolha foi a licença de uso e distribuição da ferramenta, o *PostgreSQL* tem a sua própria licença que se assemelha às licenças MIT e BSD. Os dados armazenados seguem o modelo descrito na Figura 3.4.

Figura 3.4. Diagrama de entidade relacionamento do sistema



Fonte: Autoria própria

3.3.2 Dados em Memória

Para os dados calculados pelo recomendador, foi usado o sistema gerenciador de banco de dados Redis. Uma alternativa aos sistemas tradicionais que gerencia dados em memória, que se traduz em um aumento no desempenho e redução do custo causado por operações em disco convencionais. Tal ferramenta permite uma maior flexibilidade na inserção/obtenção de dados, pois trabalha com o conceito de chave/valor.

Um dos usos de sistemas de gerenciamento de banco de dados em memória é o *cache* de informações muito requisitadas ou de valores pré-calculados, contribuindo para o desempenho do sistema em geral.

Para aumentar o desempenho do serviço de recomendação, algumas informações relevantes foram salvas no Redis. Os dados de preferência de usuários, as similaridades entre os usuários do sistema e as avaliações de cada usuário. Para os alertas, após aplicados os cálculos do modelo de recomendação, os resultados são salvos de forma ordenada pelo *score* de cada alerta no sistema.

3.3.3 Linguagens de Programação

Com o intuito aumentar o desempenho do sistema, devido a grande carga de processamento exigida principalmente pelo serviço de recomendação, foi escolhida uma linguagem de programação compilada e que não possui sistema de gerenciamento de memória embutido (*Garbage Collector*), assim evitando o uso de recursos computacionais para o alocação/desalocação de memória.

Dentre as opções mais viáveis estavam as linguagens C++ e Rust, as duas alternativas poderiam ter sido usadas. A escolha entre estas duas opções foi influenciada pelo gerenciamento de pacotes e ambiente de desenvolvimento de cada uma delas. Observando os pontos destacados a linguagem Rust foi escolhida. Esta linguagem foi usada para os principais serviços do sistema, que inclui o ***broker***, ***recommender*** e ***database***.

Para a implementação do serviço cliente do sistema, foi usada a linguagem de programação TypeScript, pois além de manter uma coesão com um sistema de tipagem é de fácil integração com o principal *framework* de desenvolvimento de aplicações Web chamado React.

3.3.4 Comunicação

Para a comunicação entre os serviços foi usado dois protocolos de comunicação entre componentes de software, o *framework gRPC*[4], uma implementação universal (múltiplas linguagens) de alto desempenho e de código aberto do protocolo RPC e o padrão REST de comunicação via HTTP.

O protocolo RPC foi usado entre os serviços cruciais do sistema (*broker*, *recommender* e *database*) pois possui uma interface comunicação estruturada e fixa, interface essa definida por um conjunto de especificações dadas pelo mecanismo de serialização de dados estruturados *Protocol Buffers*.

[4] https://grpc.io/ Acessado em 30/08/2020

Para tornar mais fácil o desenvolvimento do cliente do sistema, cujo a linguagem tem um suporte nativo para a serialização *JSON*, foi criado uma interface HTTP que segue os padrões que caracterizam um aplicação como *RESTful*, ou seja, segue os padrões de comunicações REST entre os componentes.

3.4 Implantação da Arquitetura

Todo o sistema foi compartimentalizado em diferentes serviços e processos para que a implantação pudesse ocorrer separadamente. Por conta das características da arquitetura implementada, os serviços de recomendação e gerenciamento do banco de dados podem ser escalados sem que a estrutura e interface de comunicação entre o serviço cliente e intermediador sejam alterados.

Com o aumento das operações do sistema, a criação de *clusters* para os banco de dados (PostgreSQL e Redis) é possível sem grandes problemas, desde que mantenham a mesma interface de comunicação e operabilidade.

3.5 Avaliação

Visando a avaliação do sistema desenvolvido e da arquitetura projetada, foi executado uma série de testes de carga. Os testes de carga têm um papel fundamental na avaliação de sistemas, pois nos mostra resultados importantes com relação à disponibilidade e eficiência por meio de uma grande quantidade de interações com o sistema. Para este experimento foram executados uma grande quantidade de requisições, visando observar a escalabilidade do sistema e sua arquitetura. A avaliação de escalabilidade proposta foi dividida em 3 (três) baterias de testes, um para cada tipo de recomendação no sistema. Cada bateria de testes contou com 250.000 (duzentas e cinquenta mil) requisições.

Para executar os testes foi usada a ferramenta ab (*Apache HTTP server benchmarking tool*) disponível em ambientes Unix para a análise de servidores HTTP. Dentre os dados gerados por essa ferramenta, os usados nesta análise foram a quantidade média de requisições por segundo, tempo médio de resposta e o percentual de requisições por tempo de resposta.

Para a avaliação do sistema foi usada uma base de dados sintética contendo **6.022** (seis mil e vinte dois) alertas, **100** (cem) usuários e **8.266** (oito mil duzentas e sessenta e seis) avaliações. A avaliação do modelo de recomendação contou com a recomendação de 10 (dez) alertas para 10 (dez) usuário com preferências distintas, com os dados gerados foi possível averiguar a porcentagem de acurácia na recomendação de itens de acordo as preferências dos usuários.

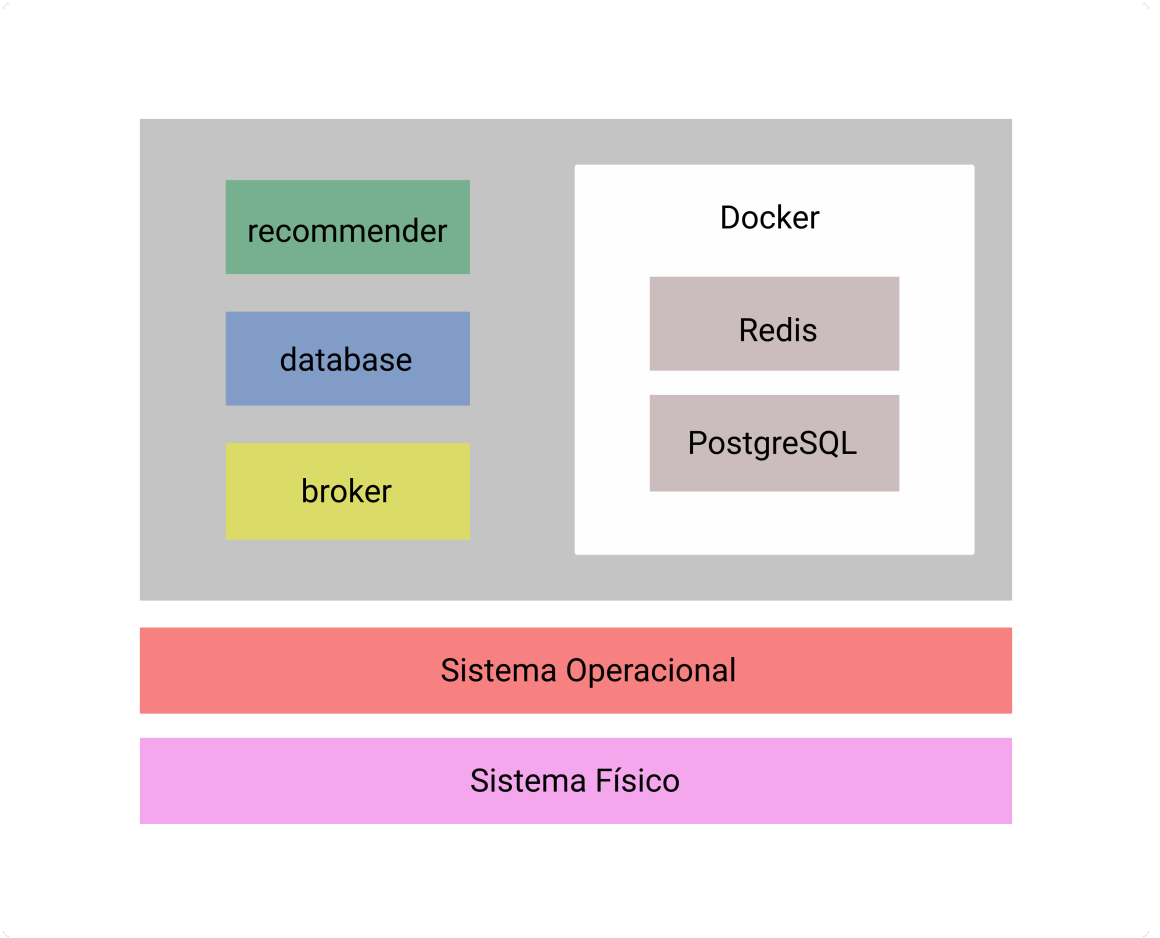
Os testes foram executados em uma máquina com as seguintes configurações: **CPU:** Intel i5-6300U (4) @ 3GHz, **Memória**: 8GB, **Tipo de armazenamento**: SSD.

Para a execução de algumas ferramentas foram utilizados *containers*, facilitando assim a implantação. Os sistemas gerenciadores de bancos de dados (PostgreSQL e Redis) foram instalados em *containers* pois já possuíam configurações padrões que satisfazem os requisitos dos testes para

o sistema. O uso de uma camada pode aumentar o tempo de operação do sistema, contudo não foi notado grandes alterações dos comportamentos esperados do sistema por completo.

A avaliação do sistema foi executada de forma que os componentes do sistema estivessem dispostos no sistema como na Figura 3.5, dois dos componentes do sistema foram executados em *containers* utilizando o Docker[5] e o restante dos serviços implementados foram executados diretamente no sistema hospedeiro.

Figura 3.5. Diagrama de componentes do sistema usado na avaliação



Fonte: Autoria própria

A avaliação do modelo de recomendação já foi realizada em (CAMPIOLO, 2016) com dados sintéticos. No contexto deste trabalho, tinha-se inicialmente o objetivo de analisar em um cenário real. No entanto, resolveu-se avaliar somente a implementação do modelo, medindo a acurácia em uma avaliação empírica simples. A avaliação consistiu na requisição e recomendação de 10 (dez) alertas para 10 (dez) usuários com preferências distintas. Em seguida, foi verificado manualmente se os alertas recomendados atendiam os perfis dos usuários.

[5] https://www.docker.com/ acessado em 11/07/2020

3.6 Considerações

Este capítulo abordou o projeto e o desenvolvimento da arquitetura de software e o modelo de recomendação. Foram apresentadas as etapas e características do desenvolvimento do projeto além dos motivos e escolha de uso para cada tecnologia ou abordagem utilizadas no desenvolvimento. No próximo capítulo são apresentados alguns detalhes da implementação, a avaliação da arquitetura e do modelo, e a discussão dos resultados.

4 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados e discussões a respeito do desenvolvimento deste trabalho. A Seção 4.1 aborda detalhes de implementação do sistema, a Seção 4.2 apresenta os resultados da avaliação de escalabilidade da arquitetura, a Seção 4.3 apresenta os resultados da avaliação empírica do modelo de recomendação. Por fim, a Seção 4.4 discorre sobre questões da avaliação do sistema.

4.1 Implementação

Para a implementação foram criados repositórios de código separados no Github[1]. Além dos repositórios de cada serviço implementado, foi criada uma base de código para os arquivos de *ProtoBuffers*, os quais especificam as interfaces de comunicação entre os componentes que utilizam o método de comunicação *gRPC*. Todo o código desenvolvido neste trabalho usa a licença MIT, sendo assim livre para a utilização, modificação e distribuição.

Os módulos implementados foram projetados para que possuam um alto desempenho. Procurou-se aplicar boas práticas de desenvolvimento, assim como o uso de ferramentas e linguagens de referência no desenvolvimento de aplicações.

Dentre os módulos implementados, o de maior relevância é o ***recommender***[2]. Esse módulo é responsável pela filtragem de alertas e implementação das funções/equações descritas neste trabalho. Os outros módulos implementados, ***broker***[3] e ***database***[4], são responsáveis pela validação das requisições e autenticação dos usuários no sistema e estruturação de consultas no SGBD, respectivamente.

Os serviços são particionados em módulos, cada módulo sendo representado por um diretório na estrutura de código. A seguir estão descritos os principais módulos implementados para cada serviço no sistema.

broker

- *config*: Consiste da definição de variáveis de ambiente do serviço;
- *controllers*: Provê a implementação dos funções que definirão o comportamento de cada *endpoint* no serviço;
- *grpc*: Define os atributos e funções para a comunicação com ambos os microsserviços: *recommender* e *database*;

[1] https://github.com/ogoestcc
[2] https://github.com/ogoestcc/recommender
[3] https://github.com/ogoestcc/broker
[4] https://github.com/ogoestcc/database

- *middlewares*: Estruturas de insterceptação de requisições, tais como a autenticação de usuário e validação do conteúdo de requisições;
- *models*: Define os tipos de dados usados no serviço;
- *route*: Configura o roteamento do serviço HTTP;
- *main.rs*: Ponto inicial do serviço, no qual é configurado o *logger* e o servidor HTTP.

database

- *database*: Prove a implementação dos métodos de consulta no SGBD;
- *models*: Define os tipos de dados usados no serviço;
- *services*: Contém a implementação RPC para o serviço.

recommender

- *config*: Consiste da definição de variáveis de ambiente do serviço;
- *models*: Define os tipos de dados usados no serviço;
- *recommender*: Contém a implementação dos métodos de filtragem e recomendação dos alertas no sistema;
- *redis*: Provê as funções essenciais para a comunicação com o *Redis*;
- *services*: Define os métodos de comunicação com o serviço *database* e a implementação RPC para o *recommender*;

A implementação do serviço ***client*** do sistema não foi completada durante o tempo disponível, sendo esse uma das potenciais adições de trabalhos futuros.

4.2 Avaliação de Escalabilidade

A avaliação de escalabilidade foi executada de acordo com os testes descritos na Seção 3.5. Os resultados estão sintetizados nas Tabelas 4.1 e 4.2.

Tabela 4.1. Média de requisições por segundo e tempo de resposta para cada tipo de recomendação

	Requisições por segundo	Tempo de resposta (ms)
Top-N	305	13,1
Baseada em Conteúdo	177	22,6
Filtragem Colaborativa	35	113,5

Fonte: Autoria própria

A Tabela 4.1 mostra os valores médios para a quantidade de requisições servidas por segundo e o tempo de resposta médio (em milissegundos) para cada um tipo de recomendação projetados e implementados neste trabalho.

A Tabela 4.2 evidencia a porcentagem de requisições servidas e seus respectivos tempo de resposta, de modo que no método de recomendação *Top-N*, 50% (cinquenta por cento) das requisições

Tabela 4.2. Porcentagem de requisições servidas em determinada quantidade de tempo (ms) por tipo de recomendação

	50%	66%	75%	80%	90%	95%	98%	99%	100%
Top-N	9	11	17	21	23	24	28	32	**243**
Baseada em Conteúdo	17	29	-	30	32	34	40	43	**260**
Filtragem Colaborativa	97	141	144	145	150	154	164	191	**1255**

Fonte: Autoria própria

foram servidas em um tempo menor que 9 (nove) milissegundos e 90% (noventa por cento) da requisições foram servidas em menos de 23 (vinte e três) milissegundos.

4.3 Avaliação Empírica do Modelo de Recomendação

Para a avaliação do modelo de recomendação, foi executado um teste empírico do modelo para se obter resultados preliminares com relação ao desempenho e acurácia do modelo projetado. Foram coletados os resultados, descritos na Tabela 4.3, da acurácia da recomendação para cada tipo de recomendação. Para os alertas recomendados foram avaliados a presença das preferências do usuário.

Tabela 4.3. Porcentagem de acurácia obtidas na avaliação de cada tipo de recomendação personalizada

	Min (%)	Média (%)	Max (%)
Recomendação Baseada em Conteúdo	50	66	80
Filtragem Colaborativa	60	70	90

Fonte: Autoria própria

Observa-se na Tabela 4.3 que o modelo recomenda com acerto médio de 66 e 70% segundo as preferências dos usuários usados nesta avaliação.

4.4 Discussões

Observando a Tabela 4.2, verifica-se que os valores destacados em negrito dizem respeito ao tempo de resposta de uma requisição mais longo durante todo a bateria de testes para cada tipo de recomendação. Esses valores, por serem maiores que os anteriores, correspondem apenas a 1% das das 2500 (duas mil e quinhentas) requisições. Em geral, verifica-se que a maior parte das requisições não possui valores tão altos de tempo de resposta, tornando assim a requisição mais longa um caso isolado.

Devido a execução dos testes terem ocorridos em uma única máquina, não foi possível avaliar o impacto gerado pela transferência de dados em rede e suas possíveis implicações nos tempos de respostas observados.

Apesar de conter dados importantes a respeito do quão bom um servidor pode se comportar perante a uma grande quantidade de requisições em série, a ferramenta usada (**ab**) não informa dados precisos de porcentagem de tempos de resposta das requisições. A quantidade de requisições com um tempo de resposta maiores que 99% de todas precisam ser analisadas para evidenciar possíveis quedas de desempenho.

Com os resultados obtidos a partir dos teste de carga do sistema, podemos tirar conclusões satisfatórias a respeito da implementação e implantação de um sistema como o desenvolvido, assim trabalhos futuros podem adicionar e aperfeiçoar ainda mais o modelo e arquitetura desenvolvidos neste trabalho.

A avaliação do modelo, apesar de ter sido desenvolvida de forma empírica e com uma base de dados reduzida e sintética, nos proporcionou uma visão das capacidades de recomendação. Apesar da carência de testes mais completos e mais abrangentes, os resultados obtidos demonstram um potencial satisfatório que poderá ser explorado em trabalhos futuros.

4.5 Considerações

Este capítulo apresentou os resultados da implementação do modelo de recomendação e como foi desenvolvido usando microsserviços, contêineres e armazenamento de dados em memória para prover escalabilidade. A arquitetura foi avaliada com testes de carga e o modelo de recomendação com testes usando base sintética. A arquitetura suportou as requisições considerando um cenário de testes simples. O modelo de recomendação mostrou-se capaz de recomendar alertas de interesse para administradores, apesar de necessitar de melhoramentos. No próximo capítulo são apresentadas as conclusões e os trabalhos futuros.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho alcançou o objetivo principal de criação de um sistema de recomendação para alertas cibernéticos e da maioria dos objetivos específicos propostos, dentre eles a extensão do modelo de recomendação base Esposte et al. (2016) e a implementação e avaliação de uma arquitetura escalável para tal sistema.

A arquitetura e o modelo de recomendação atenderam aos requisitos de escalabilidade e precisão nas recomendações. A arquitetura consegue lidar com uma alta carga de requisições e, caso necessário, suporta a execução de mais instâncias dos serviços. O modelo de recomendação implementado retornou inúmeros alertas verdadeiros positivos, indicando que foi implementado e aperfeiçoado corretamente.

Devido à limitação de tempo e às restrições de distanciamento social ocasionadas pela pandemia COVID-19, não foi possível realizar uma avaliação com usuários e ambientes reais. No entanto, na avaliação com dados da base de recomendações sintética, o sistema mostrou-se funcional e alcançou precisão satisfatória, mesmo sem ajustes finos.

Sendo assim, verificou-se que a ferramenta para a recomendação de alertas cibernéticos atende as especifidades de recomendações na área de Cibersegurança, provendo ao usuário informações relevantes e contextualizadas de um conjunto volumoso de alertas.

Como trabalhos futuros, sugere-se a implantação e avaliação da arquitetura e sistema em ambiente real, aperfeiçoamento do modelo de recomendação, construção de mecanismos de coleta de alertas para alimentar o sistema e/ou padronização das estruturas de representação de alertas segundo os modelos já existentes.

REFERÊNCIAS

CAMPIOLO, Rodrigo. ***Análise e extração de alertas antecipados sobre ameaças e incidentes de segurança em sistemas computacionais usando fontes de dados não estruturados***. Tese (Doutorado) — Instituto de Matemática e Estatística da Universidade de São Paulo, 2016.

CERT. ***Códigos Maliciosos***. 2020. Disponível em: https://cartilha.cert.br/malware, Acessado em: 21/11/2020.

CVE. ***CVE***. 2020. Disponível em:https://cve.mitre.org/about/index.html, Acessado em: 07/08/2020.

DANYLIW, R. (CERT); MEIJER, J. (UNINETT); DEMCHENKO, Y. (University of Amsterdam). ***The Incident Object Description Exchange Format***. [S.l.], 2007. 23 p.

ESPOSTE, Arthur de Moura Del; CAMPIOLO, Rodrigo; KON, Fabio; BATISTA, Daniel. **A Collaboration Model to Recommend Network Security Alerts Based on the Mixed Hybrid Approach**. In: *Anais do SBRC 2016*. Salvador, Bahia: [s.n.], 2016. p. 586 599.

FIPS-USA. ***Minimum security requirements for federal information and information systems***. [S.l.], 2006.

GLASSMAN, Michael; KANG, Min Ju. **Intelligence in the internet age: The emergence and evolution of Open Source Intelligence (OSINT)**. *Computers in Human Behavior*, v. 28, p. 673–682, 2012.

GROUPLENS. ***MovieLens***. 2020. Disponível em: http://grouplens.org/datasets/movielens, Acessado em: 09/06/2020.

HOGAN, Michael; NEWTON, Elaine. ***Supplemental Information for the Interagency Report on Strategic U.S. Government Engagement in International Standardization to Achieve U.S. Objectives for Cybersecurity***. [S.l.], 2015.

HRIBAR, Gašper; PODBREGAR, Iztok; IVANUŠA, Teodora. **OSINT: A "Grey Zone"?** *International Journal of Intelligence and CounterIntelligence*, v. 27, n. 3, p. 529–549, 9 2014. ISSN 0885-0607.

ISO. ***Information technology — Security techniques — Code of practice for information security management***. [S.l.], 2005.

JASPER, Scott E. **U.S. Cyber Threat Intelligence Sharing Frameworks**. *International Journal of Intelligence and CounterIntelligence*, v. 30, n. 1, p. 53–65, 1 2017. ISSN 0885-0607.

MITRE. ***MITRE***. 2020. Disponível em: https://www.mitre.org/, Acessado em: 03/08/2020.

MITTAL, Sudip; DAS, Prajit Kumar; MULWAD, Varish; JOSHI, Anupam; FININ, Tim. **CyberTwitter: Using Twitter to generate alerts for cybersecurity threats and vulnerabilities**. In: *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.: s.n.], 2016. p. 860–867. ISBN 978-1-5090-2846-7.

OVAL. ***OVAL***. 2020. Disponível em: https://oval.mitre.org/about/, Acessado em: 03/08/2020.

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha; KANTOR, Paul B. ***Recommender Systems Handbook***. 1. ed. [S.l.]: Springer, Boston, MA, 2011. ISBN 9780387858203.

STALLINGS, William; BROWN, Lawrie. ***Segurança de Computadores***. 2. ed. [S.l.: s.n.], 2013. ISBN 978-85-352-6450-0.

STIXPROJECT. ***STIX***. 2020. Disponível em: https://stixproject.github.io/about/, Acessado em: 03/08/2020.

USCERT. ***Cybersecurity & Infrastructure Security Agency***. 2020. Disponível em: https://us-cert.cisa.gov, Acessado em: 07/08/2020.

VENUGOPAL, M.V.L.N. **Containerized Microservices architecture**. *International Journal of Engineering and Computer Science*, v. 6, n. 11, 11 2017. ISSN 23197242.

YCOMBINATOR. ***Hacker News***. 2020. Disponível em: https://news.ycombinator.com/, Acessado em: 03/08/2020.

ZHAO, Wanying; WHITE, Gregory. **A collaborative information sharing framework for Community Cyber Security**. In: *2012 IEEE Conference on Technologies for Homeland Security (HST)*. [S.l.: s.n.], 2012. p. 457–462. ISBN 978-1-4673-2709-1.