Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologia em Engenharias – FCTE
Engenharia de Software

DevSecOps: um estudo de caso sobre o desenvolvimento do produto MEPA

Autor: Chaydson Ferreira da Aparecida
Orientador: Msc. Hilmer Rodrigues Neri

Brasília, DF
2025

Chaydson Ferreira da Aparecida

DevSecOps: um estudo de caso sobre o desenvolvimento do produto MEPA

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade de Ciências e Tecnologia em Engenharias – FCTE

Orientador: Msc. Hilmer Rodrigues Neri

Brasília, DF

2025

Chaydson Ferreira da Aparecida
DevSecOps: um estudo de caso sobre o desenvolvimento do produto MEPA/ Chaydson Ferreira da Aparecida. – Brasília, DF, 2025-
81 p. : il. (algumas color.) ; 30 cm.

Orientador: Msc. Hilmer Rodrigues Neri

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade de Ciências e Tecnologia em Engenharias – FCTE , 2025.

1. Palavra-chave01. 2. Palavra-chave02. I. Msc. Hilmer Rodrigues Neri. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. DevSecOps: um estudo de caso sobre o desenvolvimento do produto MEPA

CDU 02:141:005.6

XXXXXXXXXXX

Errata

Elemento opcional da **??**, 4.2.1.2). **Caso não deseje uma errata, deixar todo este arquivo em branco**. Exemplo:

FERRIGNO, C. R. A. **Tratamento de neoplasias ósseas apendiculares com reimplantação de enxerto ósseo autólogo autoclavado associado ao plasma rico em plaquetas**: estudo crítico na cirurgia de preservação de membro em cães. 2011. 128 f. Tese (Livre-Docência) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, São Paulo, 2011.

Folha	**Linha**	**Onde se lê**	**Leia-se**
1	10	auto-conclavo	autoconclavo

Chaydson Ferreira da Aparecida

DevSecOps: um estudo de caso sobre o desenvolvimento do produto MEPA

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 28 de julho de 2025:

Msc. Hilmer Rodrigues Neri
Orientador

Dr. Renato Coral Sampaio
Convidado 1

Dr. Tiago Alves da Fonseca
Convidado 2

Brasília, DF
2025

A dedicatória é opcional. Caso não deseje uma, deixar todo este arquivo em branco.

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

A inclusão desta seção de agradecimentos é opcional, portanto, sua inclusão fica a critério do(s) autor(es), que caso deseje(em) fazê-lo deverá(ão) utilizar este espaço, seguindo a formatação de *espaço simples e fonte padrão do texto (sem negritos, aspas ou itálico.*

Caso não deseje utilizar os agradecimentos, deixar toda este arquivo em branco.

A epígrafe é opcional. Caso não deseje uma, deixe todo este arquivo em branco.

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chave: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

Fig.	Area of the i^{th} component
456	Isto é um número
123	Isto é outro número
lauro cesar	este é o meu nome

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

Sumário

1 Introdução

Neste capítulo são apresentados os conceitos fundamentais que norteiam este trabalho: segurança de produtos de software; Modelos de Qualidade; DevOps e; for fim; DevSecOps. Adicionalmente, são apresentados o escopo do problema, a questão de pesquisa que guia essa investigação e a estrutura geral do documento.

1.1 Contexto

Avaliar os fatores de qualidade de um *software* é de suma importância no desenvolvimento de *software* e, para isso, faz-se necessária a utilização de um modelo que guie a avaliação, de forma a sistematizar o processo e reduzir subjetividades (SIAVVAS et al., 2021a). Nesse sentido, o modelo proposto por McCall, Richards e Walters (1977), foi o primeiro modelo proposto para analisar a qualidade de produto de software. Nesse trabalho, foram identicados aspectos e propriedades do produto de software, chamados de fatores e seus respectivos subfatores. Também foi proposto um método de avaliação quantitativa, baseada nos fatores, critérios e métricas. Nesse modelo, aspetos relacionados à segurança, foi percebido como um subfator de Integridade. Tratou-se portanto, de um modelo hierárquico. Subsequentemente, o modelo proposto por Boehm (1978) evoluiu as ideias de McCall, e ambos se tornaram modelos seminais, servindo de referência para os modelos subsequentes. Além de propor novas propriedades e aspectos, renomeou os fatores e subfatores para características e subcaracterísticas. Essa terminologia é utitilzada até hoje, em modelos de referência de qualidade contemporâneos.

A partir da ISO/IEC-9126 (2001), estabeleceu-se um uma norma padrão internacional para analisar a qualidade de produto de software. Essa norma foi baseada nos modelos de McCall, Richards e Walters (1977) e Boehm (1978) além de outros, como por exemplo o modelo de Dromey (1995), que incoporava aspectos e propriedades específicas de código-fonte baseado no paradigma orientado a objetos-OO. Na ISO/IEC-9126 (2001) a segurança foi definida como uma subcaracterística de Funcionalidade. A ISO/IEC-25010 (2010) surgiu como uma evolução da ISO 9126, expandindo seus conceitos e adaptando o modelo para a realidade da qualidade de produto de software modernos. Uma relevante alteração foi transformação da até então, subcaracteristica de qualidade para característica, destacando-a como um dos pilares da qualidade do produto. Além de, reorganizar as características e suas respectivas subcaracteríticas na visão de qualidade interna, também foi incorporada ao modelo a visão da qualidade em uso. Tem-se ainda, a ISO/IEC-27001 (2022), que define requisitos para implementar, manter e melhorar continuamente, a segurança da informação em uma organização; além, da ISO/IEC-27034 (2011), que fornece

diretrizes para as organizações sobre como integrar a segurança em suas aplicações, ao longo de todo o ciclo de vida de desenvolvimento, busacando mitigar as potenciais ameaças cibernéticas. Contudo, nenhuma dessas normas ISO/IEC, fornecem um método para avaliar a segurança de software de maneira quantitativa ou qualitativa. Assim, torna-se necessário recorrer a outros modelos e ferramentas que ofereçam uma abordagem para operacionalizar os conceitos abstratos definidos nessas normas, com suas respectivas fórmulas de cálculo, medidas e métricas, valores de referência, ponderação e agregação numérica, possibilitando o cálculo de indicadores quantitativos e qualitativos de segurança. Entretanto, é importante destacar que existem alguns frameworks e práticas que auxiliam a observação de aspcetos técnicos da segurança, quanto organizaçionais, como por exemplo:

- o modelo de maturidade *OWASP-DSOMM* (OWASP, 2020) e a lista das dez vulnerabilidades de maior ocorrência *OWASP Top 10* (OWASP, 2021);

- o catálogo de vulnerabilidades *Common Weakness Enumeration-CWE*, mantido por uma comunidade que envolve a indústria, academia e governo norte americano (Mitre Corporation, 1999)

Nos dias atuais, a segurança de sistemas e produtos de software é um dos fatores mais críticos. Os ataques cibernéticos explorando vulnerabilidades de produtos de software tem provocado impactos negativos, em ordem mundial. Recente, no Brasil, por exemplo, vivenciamos a tida como maior invasão de dispositivo eletrônico do País, provocando prejuízos financeiros na ordem R$ 800 milhões (UOL, 2025) (G1, 2025) (VALOR-ECONOMICO, 2025). Portanto, torna-se imperativo promover a cultura de desenvolvimento seguro, além de, incorporar práticas e técnicas de segurança, que tornem as análises e decisões mais sistemáticas, ao longo de todo ciclo de desenvolvimento.

Nas últimas três décadas, a Engenharia de Software passou por uma profunda transformação na mameira de como se desenvolver produtos de software. Isso se deu por meio dos métodos ágeis, a filosofia de desenvolvimento Lean e práticas de comunidades de software livre (FITZGERALD; STOL, 2017) (LóPEZ et al., 2022). Atualmente, a cadência do ciclo de desenvolvimento acontece em um pequeno intervalo de tempo (dias, semanas). Isso fez com que entrega e implantação de versões de produtos de software passassem a ser disponibilizas e implantadas, continuamente. Além disso, os antigos sistemas "monolito", transformaram em subsistemas com responsabilidades específicas, especializadas e, que se comunicam entre si. Assim, as versões dos produtos passaram a ser cada vez "menores"e independentes do ponto devista da implantação. Para lidar com essa nova realidade escala no desenvolvimento de produtos de software, é necessário dispormos de tecnologias que nos auxiliem com a automação e sistematização das análises, com efeito na tomada de decisão sobre novas implantações.

Na esteira da transformação ágil, em 2009, Patrick Debois propôs o termo DevOps. Esse termo abarcou um conjunto de nuances que que em essência, buscava remover a barreira existente entre os times de desenvolvimento e operações, quando da implantação. A compreensão sobre esse termo passa por diferentes facetas como: cultura, automação; métodos e procedimentos de desenvolvimento e operações contínuos e integrados; colaboração (FRANçA; JUNIOR; TRAVASSOS, 2016) (DíAZ et al., 2022) (LUZ; PINTO; BONIFÁCIO, 2019) . Um dos principais objetivos é reduzir o ciclo de vida do desenvolvimento de software, permitindo entregas mais frequentes e automatizadas ou semi-automatizadas. As principais práticas de DevOps são a Integração Contínua (CI), que consiste em integrar o código desenvolvido na ramificação principal com validação de build e testes de forma automática para detectar falhas, e a Entrega/Implantação Contínua (CD), que consiste em deixar o software pronto para entrar em produção e realizar o seu lançamento de forma automatizada (RAJAPAKSE et al., 2022).

Já o *DevSecOps* integra os princípios e práticas do *DevOps*, adicionando o time de segurança ao processo. Esse paradigma implementa uma abordagem de segurança chamada *Shift-Left*, na qual os processos de segurança são realizados desde o início do desenvolvimento, com o objetivo de evitar problemas decorrentes de uma avaliação tardia. Além disso, é formado por práticas de segurança como treinamento da equipe, testes de segurança automatizados e feedback contínuo (RAJAPAKSE et al., 2022).

1.2 Problema

Medir a segurança de software representa um grande desafio (RAJAPAKSE et al., 2022). Embora já exista muito conhecimento acumulado na área, ainda carecemos de modelos que apresentem formas sistematizadas de avaliação da segurança. Frequentemente, os métodos são baseados em critérios subjetivos, como a opnião de especialistas e não possuem validação empírica, o que afeta a confiabilidade dos resultados (SIAVVAS et al., 2021a).

No contexto do desenvolvimento e implantação contínua, esse desafio se torna ainda mais difícil. Métodos tradicionais de análise de segurança são impraticáveis devido à velocidade das entregas (RAJAPAKSE et al., 2022). A medição da seguranca se torna ainda mais desafiadora ao lidar ciclos continuos de lançamentos de versões de produtos. A segurança é uma propriedade multifacetada, emergente e dependente do contexto, o que complica sua quantificação (KUDRIAVTSEVA; GADYATSKAYA, 2024). Em essência, a falta de compreensão sobre cultura e práticas de desenvolvimento contínuo e seguro, pode comprometer ou mesmo inviabilizar, a estratégia de negócio de várias organizações mundiais.

1.3 Questão de Pesquisa

A definição da questão de pesquisa foi elaborada utilizando a abordagem *Goal Question Metric* (GQM). Essa é uma abordagem que tem como objetivo definir, de maneira *top-down* e hierárquica, os objetivos a serem alcançados, as perguntas a serem respondidas para cumprir tais objetivos e as métricas necessárias para responder a cada pergunta de forma quantitativa, como mostra a Figura 1. Essa estrutura foi adaptada para o contexto da pesquisa, conforme a Tabela 1, resultando na seguinte questão:

Tabela 1 – GQM Adaptado

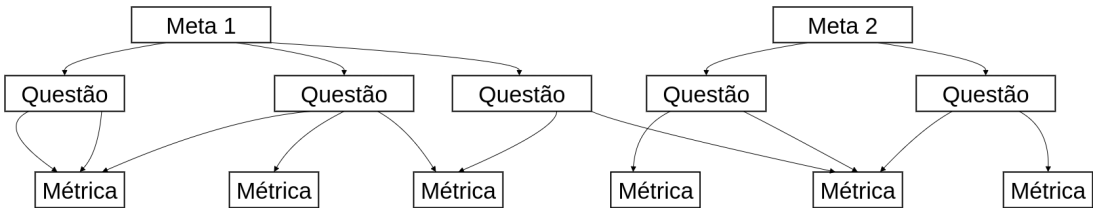
Característica	**Valor**
Analisar	a característica de qualidade de produto de software com foco na segurança e suas respectivas subcaracterísticas, nas visões interna e externa.
Com o propósito de	Caracterizar
Com respeito ao	desenvolvimento contínuo de produtos de software seguros (DevSecOps)
Do ponto de vista de	Pesquisador
No contexto do	desesenvolvimento de aplicações web seguras e de código-fonte aberto

Fonte: Adaptado de Basili, Caldiera e Rombach (1994)

Considerando essa perspectiva definimos a sguinte questão principal de pesquisa deste estudo

Como analisar a característica de segurança no desenvolvimento contínuo de sistemas web, considerando as visões de qualidade interna e externa?

Figura 1 – Abordagem GQM



Fonte: Adaptado de Basili, Caldiera e Rombach (1994)

1.4 Objetivos

O objetivo geral deste trabalho consiste em analisar a adoção de práticas de DevSecOps no ciclo de devida de desenvolvimento do produto de software-livre MEPA[1]. Para alcançar este propósito, foram definidos os seguintes objetivos específicos:

[1] Link para o repositório do projeto

- Fundamentar teoricamente os conceitos de *DevSecOps*, modelos de segurança e metodologias de desenvolvimento seguro.
- Incorporar um conjunto de práticas *DevSecOps* ao ciclo de desenvolvimento do produto de *software* sob análise.
- Planejar um estudo de caso focado na observação das práticas implementadas.
- Conduzir o estudo de caso, realizando a coleta e a análise das medidas e métricas segurança, que apoiem a discussão dos resultados alcançados.
- Apresentar as conclusões e os *insights* resultantes desta investigação.

1.5 Estrutura do Trabalho

A seguir, são apresentados os capítulos que compõem a estrutura deste trabalho.

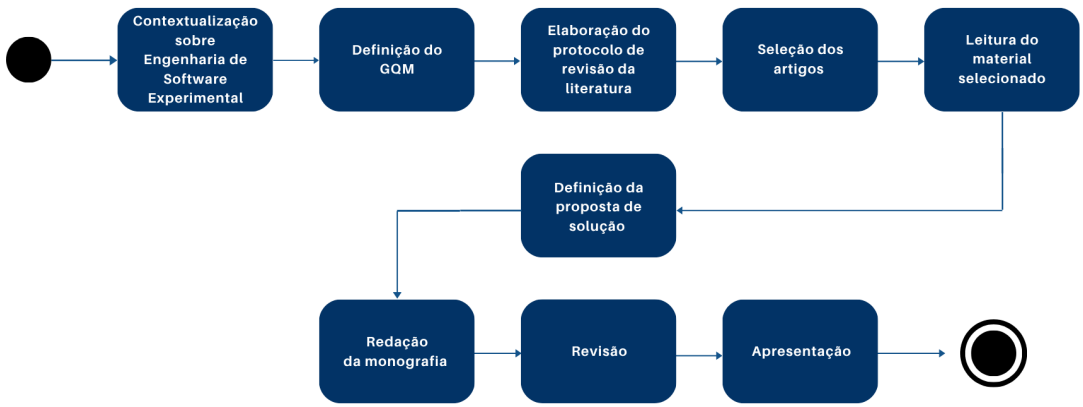
- Introdução: apresenta a contextualização do trabalho, o problema de pesquisa, a definição da questão de pesquisa e dos objetivos do trabalho. Por fim, descreve a estrutura das atividades realizadas.
- Referencial Teórico: estabelece a fundamentação teórica da monografia, abordando os tópicos centrais da pesquisa: DevSecOps, qualidade de software, segurança de software e modelos de avaliação de maturidade.
- Revisão Estruturada da Literatura: explicita o processo empregado para seleção que fundamentam este trabalho incluindo o protocolo de pesquisa e filtragem dos estudos e os resultados obtidos.
- Estudo de caso: descreve o protocolo utilizado para a condução do estudo de caso, detalhando seus objetivos, perguntas de pesquisa, atividades e resultados alcançados.
- Conclusão: consolida os achados obtidos parcial ou integralmente e como esses resultados respondem à questão de pesquisa principal, bem como as limitações do estudo e as possibilidades de aprofundamento de trabalhos futuros.

1.6 Cronograma e Atividades

1.6.1 Primeira Etapa

Esta subseção detalha as atividades desenvolvidas na primeira etapa da monografia. A Figura 2 ilustra o fluxo das atividades, enquanto a Figura 3 apresenta o cronograma correspondente.

Figura 2 – Atividades da Primeira Etapa



Fonte: Autor

- Contextualização sobre Engenharia de Software Experimental: Estudo sobre os métodos de pesquisa empírica em Engenharia de Software, como revisão sistemática da literatura, survey, experimentos e estudo de caso. Conceitos esses, fundamentais para a condução do trabalho proposto. (WOHLIN et al., 2012) (YIN, 2015) (**??**)

- Definição do GQM: Aplicação da abordagem Goal Question Metric (GQM) (BASILI et al., 1994) para a construção da questão principal de pesquisa, suas subquestões e as medidas a serem analisadas.

- Elaboração do Protocolo de Revisão da Literatura: Estruturação e adaptação do protocolo de revisão sistemática da literatura proposto por Kitchenham, Charters et al. (2007). Essa sistematização organizará nosso processo de revivão da literatura pertinente. Este processo inclui a definição da string de busca baseada no protocolo PICO, adapatado da área da medicina (PAI et al., 2004a). Por meio desse protocolo definimos sinônimos de termos, critérios de inclusão e exclusão de estudos, além do método de análise e extração de dados.

- Seleção dos Artigos: Execução da filtragem dos estudos por meio da leitura de títulos, resumos e palavras-chave, com a aplicação dos critérios de inclusão e exclusão definidos.

- Análise do Material Selecionado: Leitura completa dos artigos selecionados para aprofundar o conhecimento sobre o estado da arte em métodos de avaliação de segurança de sistemas web e práticas de desenvolvimento seguro.

Figura 3 – Cronograma da Primeira Etapa

Fonte: Autor

- Definição da Proposta de Solução: Definição os modelos de segurança, as ferramentas e as atividades do estudo de caso.
- Redação da Monografia: Escrita do texto da monografia, conforme a estrutura presente na Seção 1.5.
- Revisão: Realização das correções e dos ajustes solicitados pelo orientador.
- Defesa: Preparação do material e apresentação do trabalho final para a banca examinadora.

2 Referencial Teórico

2.1 Engenharia de Software Experimental

Introdução

Resumo Revisão Sistemática da Literatura

Resumo Estudo de Caso

Resumo Questionário

2.1.1 Estudo de Caso

2.2 Modelos de Qualidade de Software

2.3 Segurança de Software

2.4 DevOps

2.5 DevSecOps

Nesse sentido, o modelo de maturidade de segurança OWASP DSOMM (*DevSecOps Maturity Model*) é uma importante ferramenta na construção e avaliação de projetos *DevSecOps*. Ele define atividades, métricas e tecnologias que devem ser usadas para construir um ambiente *DevSecOps*, além de proporcionar o acompanhamento da maturidade da segurança do projeto em cinco dimensões: *Build and Deployment*, *Culture and Organization*, *Implementation*, *Information Gathering* e *Test and Verification* (LANGE; KUNZ, 2024).

3 Revisão Estruturada da Literatura

Este capítulo destina-se a documentar o processo realizado para selecionar o conjunto de artigos cientícos que compõe a bibliografia desta monografia. Para isso, a base de dados Scopus foi escolhida devido à sua característica de indexar diversos artigos da área da computação, muitos publicados nos principais meios de divulgação científica (ELSEVIER, 2024).

3.1 Protocolo

O protocolo utilizado para realizar a revisão estruturada da literatura foi baseado no modelo proposto por Kitchenham, Charters et al. (2007). Seu objetivo é tornar possível que outros pesquisadores, partindo do mesmo ponto, cheguem aos mesmos resultados, facilitando a replicabilidade em estudos futuros e permitindo a conferência dos resultados obtidos.

3.1.1 String de Busca

A busca em bases de dados acadêmicas requer o uso de um protocolo, pois elas indexam grande quantidade de artigos de várias áreas do conhecimento. Seu uso incorreto pode acarretar em um número excessivo de artigos sem relação com o tema da pesquisa ou, inversamente, retornar um volume insuficiente de estudos para responder à questão de pesquisa.

Por essa razão, o protocolo PICO foi utilizado para guiar a elaboração de uma string de busca adequada às necessidades desta monografia. O protocolo, no entanto, precisou ser adaptado, pois sua origem é na medicina e nem todos os seus elementos se adequam área da Engenharia de Software (PAI et al., 2004a). Uma representação visual que facilita a compreensão do protocolo pode ser vista na Figura 4.

Figura 4 – Abordagem GQM

Fonte: Adaptado de Pai et al. (2004b)

Ao adaptar o modelo PICO para e Engenharia de Software, suas definições adquirem um novo significado no contexto da Engenharia de Software. Por exemplo, o termo **Paciente**, outrora usado para indicar o perfil do paciente, passa a representar a área de aplicação, ou amostra, neste caso, o desenvolvimento de software. A **Intervenção** também sofre adaptação, deixando de significar "tratamento médico"para se referir às diferente metodologia, métodos, técnicas ou procedimentos avaliada. Não obstantemente, Resultados mantém seu sentido original, referindo-se aos efeitos ou consequências observadas. Por fim, a **Comparação** que está relacionada a investigar como a intervenção proposta se relaciona com outras propostas de intervenção, não pode ser adotada, devido a estar muito mais alinhada com os objetivos da medicina que da engenharia de software. Assim, a definição de cada um dos elementos usados do modelo estão definidos na Tabela 2.

Tabela 2 – PICO Adaptado

Elementos	**Termo Central**	**Sinônimos e Termos Relacionados**
População	software development	software systems, online systems, software applications, systems development, application development
Intervenção	DevSecOps	cybersecurity practices, security automation, secure software development, CI/CD, continuous integration, continuous deployment, continuous delivery, DevOps, security development culture
Resultados	security quality	software security, application security, security improvement, security assurance, vulnerability reduction, protection against threats, system security, owasp, cwe, common weakness enumeration, security quality characteristic, injection, cross site request forgery, XSRF, broken authentication, data exposure, external entities, broken access control, XSS, XXE, Out-of-bounds Write, SQL Injection, CSRF, Path Traversal, Directory Traversal, OS Command Injection, UAF, ISO 25010

Fonte: Autor

Desta maneira, a string de busca foi construída usando o operador lógico OR entre os termos de cada elemento, com o objetivo de englobar todos os termos da pesquisa. Já o operador AND foi usado para conectar os diferentes elementos do protocolo PICO, assim restringindo a busca apenas aos estudos que apresentam os termos necessários para responder as perguntas de pesquisa. Ademais, foram adicionados os termos próprios da base de dados SCOPUS para a realização da consulta, resultando na seguinte string de busca:

TITLE-ABS-KEY (("software development"OR "software developments"OR "software

system"OR "software systems"OR "online system"OR "online systems"OR "software application"OR "software applications"OR "system development"OR "systems development"OR "application development"OR "applications development") AND ("DevSecOps"OR "cybersecurity practice"OR "cybersecurity practices"OR "security automation"OR "security automations"OR "secure software development"OR "secure software developments"OR "CI/CD"OR "continuous integration"OR "continuous integrations"OR "continuous deployment"OR "continuous deployments"OR "continuous delivery"OR "continuous deliveries"OR "DevOps"OR "security development culture"OR "security development cultures") AND ("security quality"OR "software security"OR "application security"OR "security improvement"OR "security assurance"OR "vulnerability reduction"OR "protection against threats"OR "system security"OR "owasp"OR "cwe"OR "common weakness enumeration"OR "security quality characteristic"OR "injection"OR "cross site request forgery"OR "XSRF"OR "broken authentication"OR "data exposure"OR "external entities"OR "broken access control"OR "XSS"OR "XXE"OR "Out-of-bounds Write"OR "SQL Injection"OR "CSRF"OR "Path Traversal"OR "Directory Traversal"OR "OS Command Injection"OR "UAF"OR "ISO 25010"))

3.2 Seleção dos Artigos

Com a string de busca criada, foram estabelecidos os critérios de inclusão e exclusão visando selecionar apenas os artigos mais pertinentes ao contexto desta pesquisa. Além disso, em um primeiro momento, foram lidos o título, resumo e palavras-chave de todos os artigos resultantes da execução da string de busca, isso se deu para selecionar com maior rigor aqueles estudos que por ventura não correspondessem ao objetivo do trabalho. Posteriormente, os artigos que aprovados pelos critérios de escolha foram lidos e os dados relevantes para responder as perguntas de pesquisa secundárias, foram extraídos em um formulário. A Tabela 3 contém o protocolo completo.

Ao todo foram analisados 291 artigos resultantes da string de busca, onde 38 desses foram selecionados e lidos de maneira integral. Esses artigos foram obtidos na base de dados Scopus no dia 7 de maio de 2025. Afim de complementar os artigos selecionados por meio da string de busca, foi realizada uma busca manual com o objetivo de responder de maneira mais acertiva as perguntas de pesquisa, resultando em mais dois artigos selecionados, Accelerate State of DevOps 2024 (Google, 2024) e Quantitative DevSecOps Metrics for Cloud-Based Web Microservices (ZHANG; ZHANG, 2024), usados para compor o referencial teórico, totalizando 40 artigos. Os artigos selecionados estão dispostos na Tabelas 4. Para facilitar a compreensão do protocolo de seleção de artigos pode-se observar a Figura 5, que ilustra todas as etapas explanadas anteriormente.

Tabela 3 – Protocolo de Busca

Item	**Descrição**
Questões Secundárias de Pesquisa	1. Como o aspecto de segurança no desenvolvimento contínuo de sistemas web pode ser analisado considerando perspectivas de qualidade interna e externa? 2. Quais práticas de DevSecOps são mais prevalentes nas organizações? 3. Quais modelos de segurança são mais comumente empregados em ambientes DevSecOps? 4. Quais medidas são comumente usadas para avaliação de segurança? 5. Como as práticas de segurança são avaliadas? 6. De que maneiras as medidas de segurança são avaliadas nas organizações? 7. Quais tipos de vulnerabilidades são mitigados pelas práticas de DevSecOps? 8. Quais ferramentas e tecnologias são utilizadas em DevSecOps? 9. Qual é o volume anual de publicações sobre DevSecOps de 2009 a 2025? 10. Quantos artigos foram publicados em revistas acadêmicas? 11. Quantos estudos possuem validação experimental? 12. Se o estudo possui validação experimental, de que tipo?
String de Busca	Vide Tabela 2

Tabela 3 – Continuação

Item	**Descrição**
Critérios de Inclusão	1. Aborda o uso de práticas de desenvolvimento seguro 2. Enfatiza a qualidade de segurança de sistemas web 3. Avalia modelos de qualidade com foco em segurança 4. Concentra-se em produtos de software 5. Inclui validação experimental
Critérios de Exclusão	1. Artigos em idiomas diferentes de inglês ou português 2. Publicações duplicadas 3. Publicações com data de lançamento anterior a 2009 4. Estudos com foco em hardware, dispositivos móveis, segurança de IoT ou outros tópicos não relacionados a sistemas web

Tabela 3 – Continuação

Item	**Descrição**
Formulário de Extração	1. Título 2. Resumo 3. Ano de Publicação 4. Fonte de Publicação 5. Autores 6. Palavras-chave 7. Práticas de DevSecOps Prevalentes 8. Modelos de Segurança Empregados 9. Medidas de Avaliação de Segurança 10. Avaliação de Práticas de Segurança 11. Análise de Modelos de Segurança 12. Avaliação de Segurança Organizacional 13. Vulnerabilidades Mitigadas 14. Ferramentas e Tecnologias 15. Publicado em Revista Acadêmica 16. Validação Experimental 17. Tipo de Validação Experimental 18. Pesquisa Secundária

Fonte: Autor

Tabela 4 – Artigos Selecionados

N°	**Título**	**Publicado em Revista**	**Validação Experimental**
1	Development of Secure Software Based on the New DevSecOps Technology	Sim	Não

Tabela 4 – Continuação

N°	**Título**	**Publicado em Revista**	**Validação Experimental**
2	Automating Security in a Continuous Integration Pipeline	Não	Não
3	Extensive Review of Threat Models for DevSecOps	Sim	Não
4	Implementing and Automating Security Scanning to a DevSecOps CI/CD Pipeline	Sim	Não
5	Automating Static Code Analysis Through CI/CD Pipeline Integration	Sim	Sim
6	Design and Practice of Security Architecture via DevSecOps Technology	Sim	Sim
7	Implementation of DevSecOps by Integrating Static and Dynamic Security Testing in CI/CD Pipelines	Sim	Não
8	Research of Static Application Security Testing Technique Problems and Methods for Solving Them	Sim	Não
9	A Large-scale Fine-grained Empirical Study on Security Concerns in Open-source Software	Sim	Sim
10	Evolution of secure development lifecycles and maturity models in the context of hosted solutions	Não	Não
11	Automation and DevSecOps: Streamlining Security Measures in Financial System	Sim	Não
12	Securing the development and delivery of modern applications	Sim	Não
13	You cannot improve what you do not measure: A triangulation study of software security metrics	Sim	Sim
14	On DevSecOps and Risk Management in Critical Infrastructures: Practitioners' Insights on Needs and Goals	Sim	Sim
15	Container Security in Cloud Environments: A Comprehensive Analysis and Future Directions for DevSecOps	Não	Sim

Tabela 4 – Continuação

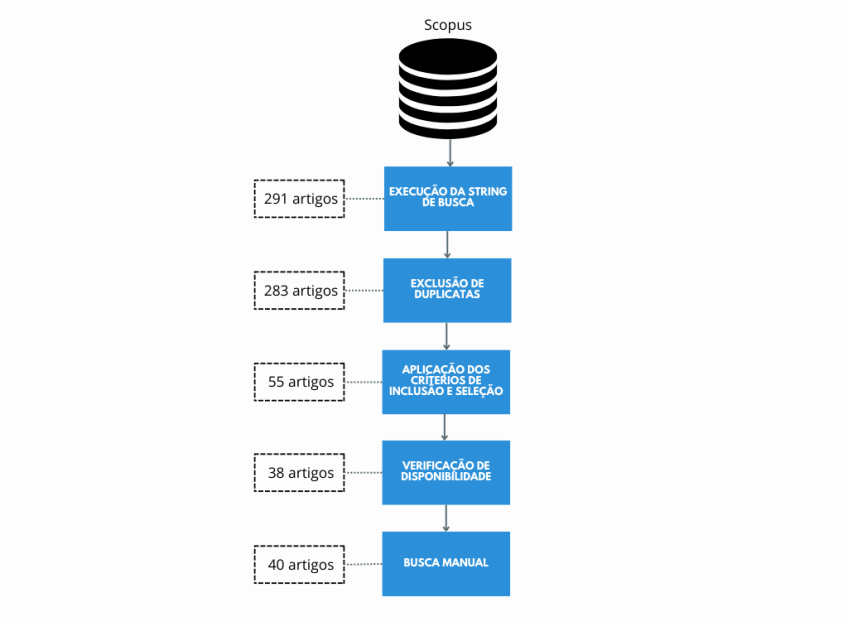
N°	**Título**	**Publicado em Revista**	**Validação Experimental**
16	Microservices-based DevSecOps Platform using Pipeline and Open Source Software	Não	Não
17	Securing the Digital Frontier: A Proactive Approach to Software Development	Sim	Não
18	A Secure Software Development Methodology for Enterprise Business Applications	Sim	Sim
19	Building Resilient CICD Pipelines: A DevOps Security-First Framework	Sim	Não
20	Review of Techniques for Integrating Security in Software Development Lifecycle	Não	Não
21	A hierarchical model for quantifying software security based on static analysis alerts and software metrics	Sim	Sim
22	A preventive secure software development model for a software factory: A case study	Sim	Sim
23	Security impacts of sub-optimal DevSecOps implementations in a highly regulated environment	Sim	Sim
24	A survey and comparison of secure software development standards	Sim	Sim
25	Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines	Sim	Sim
26	Infiltrating Security into Development: Exploring the World's Largest Software Security Study	Não	Sim
27	Challenges and solutions when adopting DevSecOps: A systematic review	Sim	Não
28	Systematic Mapping Study on Security Approaches in Secure Software Engineering	Sim	Não
29	Systematic Literature Review on Security Risks and its Practices in Secure Software Development	Sim	Não

Tabela 4 – Continuação

N°	**Título**	**Publicado em Revista**	**Validação Experimental**
30	BP: Security concerns and best practices for automation of software deployment processes: An industrial case study	Sim	Sim
31	Static analysis for web service security - Tools & techniques for a secure development life cycle	Sim	Não
32	Security characterization for evaluation of software architectures using ATAM	Sim	Sim
33	Software security	Sim	Não
34	Using the ISO/IEC 27034 as reference to develop an application security control library	Sim	Sim
35	Hunting for aardvarks: Can software security be measured?	Não	Não
36	Francois Raynaud on DevSecOps	Sim	Não
37	Integrating application security into software development	Sim	Não
38	Busting a myth: Review of agile security engineering methods	Sim	Não

Fonte: Autor

Figura 5 – Seleção dos Artigos



Fonte: Autor

3.3 Resultados

Após a leitura dos artigos, os dados foram extraídos utilizando o formulário de extração da plataforma Parsifal. Essa plataforma foi usada para conduzir a revisão da literatura devido ao seu conjunto de funcionalidades que permitem executar a revisão da literatura de forma eficiente. Depois de realizada a extração dos dados, os mesmos foram consolidados em uma planilha [1] disponível publicamente, sendo que as suas linhas representam cada um dos artigos selecionados e as colunas representam os campos do formulário de extração anteriormente apresentado na Tabela 3.

Nas subseções a seguir são apresentadas as respostas para as questões secundárias de pesquisa, específicas desta revisão.

3.3.1 Quais práticas de DevSecOps são mais prevalentes nas organizações?

- Metodologias e Cultura
  - Shift Security Left: É o princípio fundamental do DevSecOps. Ele consiste em deslocar as práticas de segurança para o início do processo de desenvolvimento do projeto, em vez de deixá-las para o final, como acontece na maioria dos casos. Essa postura eleva a prioridade da segurança no projeto, permitindo que as vulnerabilidades sejam tratadas mais cedo (RAJAPAKSE et al., 2022).
  - Continuos Vulnerability Assessment: Juntamente com Shift Left, a Avaliação Contínua de Vulnerabilidades forma os pilares do DevSecOps. Nessa prática a segurança do software é continuamente verificada, não somente durante o desenvolvimento, mas também após a implantação do software, pois é necessário monitorar a segurança do sistema durante todo o seu ciclo de vida (RAJAPAKSE et al., 2022).
  - CI/CD: Continuous Integration (CI) e Continuous Delivery/Deployment (CD) integram o DevSecOps, que se trata de uma evolução do DevOps. Como esses conceitos são herdados, é importante defini-los para o completo entendimento da metodologia. CI se refere à prática de realizar continuamente integrações de código na branch principal do código. Como essa atividade envolve a alteração da ramificação principal, ela é validada por meio de verificadores de build e testes automatizados. Já o CD visa à implantação automática e contínua das atualizações de código no ambiente de produção. Para isso, o novo código passa por verificações de qualidade e, se tudo estiver de acordo com a polí-

[1] Planilha com o resultado da revisão: <https://docs.google.com/spreadsheets/d/1sGjU2aB8CTqvINyPxsv-z_gWZCGzmErr/edit?usp=sharing&ouid=103637687294157542932&rtpof=true&sd=true>

tica estabelecida, é publicado sem intervenção humana por meio do pipeline (RAJAPAKSE et al., 2022).

  - Continuous Feedback: Obter feedback de maneira contínua e rápida é vital em ambientes de entrega contínua. Nessa abordagem, os problemas são rapidamente identificados, possibilitando que as informações cheguem depressa às equipes para que as medidas necessárias sejam tomadas. Os métodos tradicionais de coleta de dados e feedback são lentos demais para a agilidade requisitada em ambientes DevSecOps, e essa lentidão afeta diretamente a velocidade de localização e resolução dos problemas (RAJAPAKSE et al., 2022).

- Testes de Segurança

  - SAST, DAST, IAST: Ferramentas SAST executam testes estáticos, ou seja, são realizados apenas sobre o código-fonte, podendo indicar potenciais vulnerabilidades e más práticas, conhecidas como code smells. Por outro lado, as ferramentas DAST analisam o software em execução, o que possibilita testar o comportamento do sistema em uso. As ferramentas IAST, por sua vez, fornecem uma abordagem híbrida, incorporando características da análise estática e dinâmica; são modernas e possuem boa integração com ambientes de desenvolvimento contínuo (RAJAPAKSE et al., 2022).
  - Fuzz Testing: O teste de fuzzing, como também é conhecido, destaca-se entre as práticas de teste de segurança. Ele consiste em fornecer à aplicação entradas aleatórias, inválidas e inesperadas, de forma a verificar possíveis casos de borda não testados (MASOOD; JAVA, 2015).
  - BDST: Os testes BDST são baseados no BDD, porém aplicados ao contexto de testes de segurança. Como os testes descrevem seu comportamento em linguagem natural, pessoas de fora da área de desenvolvimento e segurança de software conseguem compreender os testes realizados (RANGNAU et al., 2020).

- Infraestrutura

  - IaC: Essa prática consiste em definir a infraestrutura do projeto como código. É altamente usada em contextos DevSecOps, pois possibilita a rápida configuração da infraestrutura. Como está definida em código, pode-se realizar seu versionamento, teste e implantação de forma ágil e adequada a ambientes complexos que necessitam de segurança robusta (RAJAPAKSE et al., 2022).

- Acompanhamento

  - Monitoring e Logging: Muitas vezes, o registro e a documentação dos eventos relacionados à segurança são negligenciados pelas equipes, o que consiste em um grande erro. Registrar e monitorar fornece um feedback valioso que pode ser

utilizado para tomar decisões estratégicas ou realizar auditorias (RAJAPAKSE et al., 2022).

3.3.2 Quais modelos de segurança são mais comumente aplicados em ambientes DevSecOps?

- OWASP SAMM: O Software Assurance Maturity Model é um modelo prescritivo de maturidade de segurança de software, ou seja, ele informa quais atividades precisam ser realizadas, diferentemente de um modelo descritivo, que apenas descreve as atividades. Sua estrutura é adequada para diferentes tipos e tamanhos de empresas, bem como para distintas metodologias de desenvolvimento, como cascata e ágil (LANGE; KUNZ, 2024).

- OWASP DSOMM: Embora baseado no SAMM, o DevSecOps Maturity Model nasce devido à necessidade de um modelo adequado para ambientes DevOps, onde a segurança é parte essencial do ciclo de vida. Também é prescritivo, mas, diferentemente do SAMM, suas atividades são definidas em um nível mais próximo do programador do que da gestão. Desse modo, ele fornece com detalhes técnicos os requisitos necessários para atingir cada um dos níveis de maturidade em suas diferentes dimensões (LANGE; KUNZ, 2024).

- BSIMM: Diferentemente dos outros, este é um modelo descritivo, ou seja, ele descreve atividades sem exigir que sejam implementadas. Outra diferença fundamental é que se trata de um modelo proprietário, mantido pela Synopsys. A aplicação do modelo, bem como sua metodologia de avaliação, só estão disponíveis mediante contratação dos serviços da empresa (LANGE; KUNZ, 2024).

3.3.3 Quais medidas são comumente usadas para aferição de segurança?

- Métricas de Zhang: Segundo Zhang e Zhang (2024), medir de forma eficaz as características de *softwares web* é fundamental, porém pouco explorado. Para resolver este problema, eles realizaram uma revisão sistemática da literatura e definiram doze métricas voltadas a atender às necessidades dos sistemas *web* que usam *DevSecOps*. Elas permitem quantificar o desempenho do serviço, a segurança e a eficiência da operação, apoiando, assim, as tomadas de decisão e a melhoria contínua das práticas.

  - Non-Comment Lines of Code: Tamanho do código-fonte, excluindo comentários e linhas em branco.

  - Design Defect Ratio: Proporção de defeitos de design em relação às linhas de código não comentadas.

  - Shared or Unknown Library Ratio: Proporção de bibliotecas compartilhadas ou não verificadas em um serviço.
  - Technical Debt Ratio: Compara o custo de resolução de um débito técnico com o custo total do código-fonte.
  - Continuous Deployment Cycles Score: Pontuação dos ciclos de implantação contínua.
  - Mean Change Lead Time: Tempo médio que uma mudança leva desde o commit até chegar à produção.
  - Mean Time to Recover: Tempo médio para recuperação de falhas causadas nos pipelines de CI/CD.
  - Mean Number of Test Cases Per Parameter: Média de casos de teste por parâmetro.
  - Points of Environmental Risk: Total de riscos de segurança não resolvidos em produção.
  - Time for Response: Tempo médio que as equipes de desenvolvimento levam para solucionar incidentes de segurança.
  - Throughput: Mede a capacidade de processamento de um serviço.
  - Errors Per Time Unit: Taxa de erros em determinada unidade de tempo.

- Métricas DORA: O DevOps Research and Assessment (DORA) é um dos principais programas de pesquisa do mundo na área de DevOps. Esse programa, que faz parte do Google, chegou, após anos de estudos, a quatro métricas-chave para medir os aspectos de DevOps de um projeto. Suas métricas passam por uma avaliação estatística rigorosa para possibilitar o entendimento da relação entre as medições e o sucesso das organizações (Google, 2024).
  - Change lead time: Tempo que uma alteração leva para chegar à produção.
  - Deployment frequency: Frequência com que as alterações chegam à produção.
  - Change fail percentage: Percentual de implantações que causam falhas em produção.
  - Failed deployment recovery time: Tempo necessário para se recuperar de uma implantação com falha.

3.3.4 Como as práticas de segurança são avaliadas?

É fundamental que as práticas de segurança sejam avaliadas para que seja possível analisar a eficácia das abordagens adotadas, evoluir as existentes ou adotar novas que, se

adequem melhor às necessidades da organização. Sendo assim, a avaliação de métricas representa uma forma eficiente e quantitativa de avaliar práticas de segurança, pois permite acompanhar a evolução das atividades e verificar os resultados de cada uma. Outra forma de analisar as práticas de segurança é por meio de auditorias, que verificam a adequação da organização aos padrões de segurança e conformidade (RAJAPAKSE et al., 2022).

Por fim, podem-se usar modelos de avaliação de maturidade, pois eles introduzem uma linha de base para comparação com a organização avaliada. Eles permitem analisar o estado atual da aplicação, além de identificar áreas de melhoria e traçar um plano para alcançar um nível de segurança mais elevado (LANGE; KUNZ, 2024).

3.3.5 De que maneiras as medidas de segurança são avaliadas nas organizações?

As medidas de segurança são avaliadas de diversas formas dentro das organizações. Uma das principais é a análise da repercussão das métricas de segurança nos KPIs da organização. KPIs (Indicadores-Chave de Desempenho) são as métricas centrais que medem a saúde dos projetos. É crucial criar alertas e dashboards para acompanhar o desenvolvimento das métricas e KPIs, pois, assim, pode-se obter insights de como as métricas de segurança impactam nos indicadores da empresa, além de possibilitar o rastreio das métricas durante todo o período em que foram monitoradas (JOSHI, 2024).

Outra maneira de avaliar as medidas de segurança está relacionada ao quanto elas se adequam ao *compliance* da organização ou a aspectos regulatórios. Muitas vezes, as empresas precisam seguir rígidos padrões de conformidade relacionados às métricas — como a cobertura de testes, que, dependendo da área, precisa ser extremamente alta —, bem como em setores financeiros, onde uma falha de segurança pode gerar um prejuízo bilionário (KUDRIAVTSEVA; GADYATSKAYA, 2024).

3.3.6 Que tipos de vulnerabilidades são mitigadas pelas práticas de DevSecOps?

Existem diversos tipos de falhas de segurança que podem ocorrer durante o processo de desenvolvimento de software, entre elas, falhas que podem extrapolar o escopo do trabalho, como as relacionadas ao hardware utilizado. Por esse motivo, é necessário entender quais problemas de segurança são afetados pelas práticas DevSecOps, pois, desse modo, será possível analisar de forma mais assertiva a repercussão da adoção dessa metodologia na qualidade da segurança. Na Tabela 5, foram listadas as principais vulnerabilidades impactadas por esse paradigma.

Tabela 5 – Vulnerabilidades Reduzidas

Vulnerabilidade	**Referência**
SQL Injection	Saeed et al. (2025)
Command Injection	Ramirez, Aiello e Lincke (2020)
XSS	Saeed et al. (2025)
XXE	Nocera et al. (2023)
Buffer Overflow	Ramirez, Aiello e Lincke (2020)
CSRF	Kushwaha, David e Suseela (2024)
DDoS	Saeed et al. (2025)
MITM	Nocera et al. (2023)
Broken Authentication	Saeed et al. (2025)
Broken Access Control	Saeed et al. (2025)
Security Misconfiguration	Saeed et al. (2025)
Session Hijacking	Kushwaha, David e Suseela (2024)
SSRF	Nocera et al. (2023)

Fonte: Autor

3.3.7 Quais ferramentas e tecnologias são usadas no DevSecOps?

- Monitoramento: Prometheus, Grafana, Loki
- Infraestrutura: Terraform, Kubernetes, Docker
- CI/CD: Jenkins, GitLab CI/CD, GitHub Actions, Tekton, ArgoCD
- Testes: SonarQube, FindBugs, Snyk, OWASP Dependency-Check, OWASP ZAP, Trivy, Detect Secrets, Asylo, StackHawk, JMeter, Selenium

3.3.8 Qual é o volume anual de publicações sobre DevSecOps de 2009 a 2025?

O volume anual de artigos está representado na Figura 6. Observa-se o crescimento das pesquisas sobre o tema, principalmente após o ano de 2020, chegando a seu pico em 2024. Em abril de 2025, mês em que a string de busca foi executada, a quantidade de estudos já se igualava ao volume de todo o ano de 2023, o que evidencia o crescimento e a importância da área nos meios acadêmico e profissional.

Figura 6 – Volume Anual de Artigos entre 2009 e 2025

Fonte: Autor

3.3.9 Quantos artigos foram publicados em periódicos acadêmicos?

Como indicado na Figura 7, a grande maioria dos artigos foi publicada em revistas científicas. Desse modo, sabe-se que a maior parte dos artigos passou por um critério alto de revisão e análise de qualidade, elevando o nível de confiabilidade dos resultados da pesquisa.

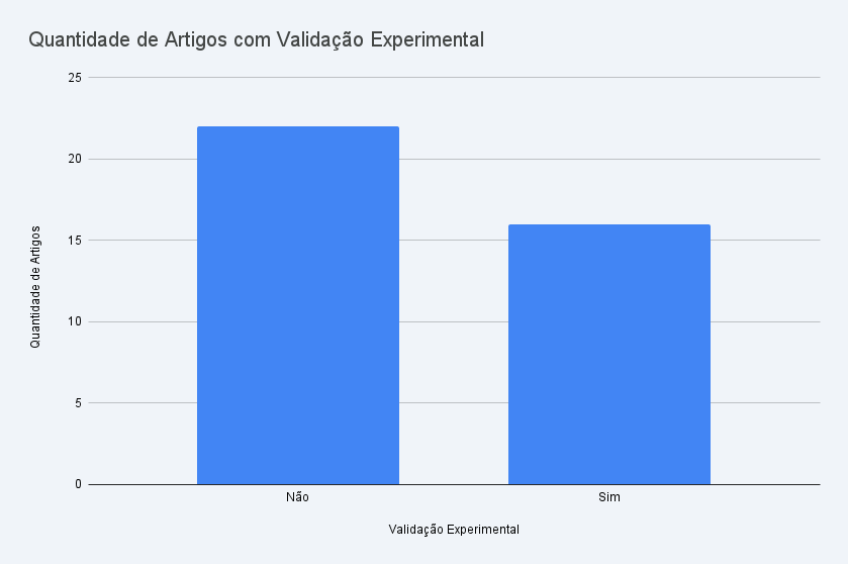
Figura 7 – Artigos Publicados em Revistas

Fonte: Autor

3.3.10 Quantos estudos têm validação experimental?

Um total de dezesseis estudos, dos quarenta selecionados, ou seja, quarenta porcento, conforme pode ser observado na Figura 8, que apresenta a quantidade de estudos que possuem ou não essa validação. O baixo índice de validação experimental pode estar relacionado à característica emergente da área; por ser muito recente, ainda falta estabelecer e consolidar os métodos de pesquisa comumente usados em outras áreas para a validação dos estudos.

Figura 8 – Quantidade de Artigos com Validação Experimental

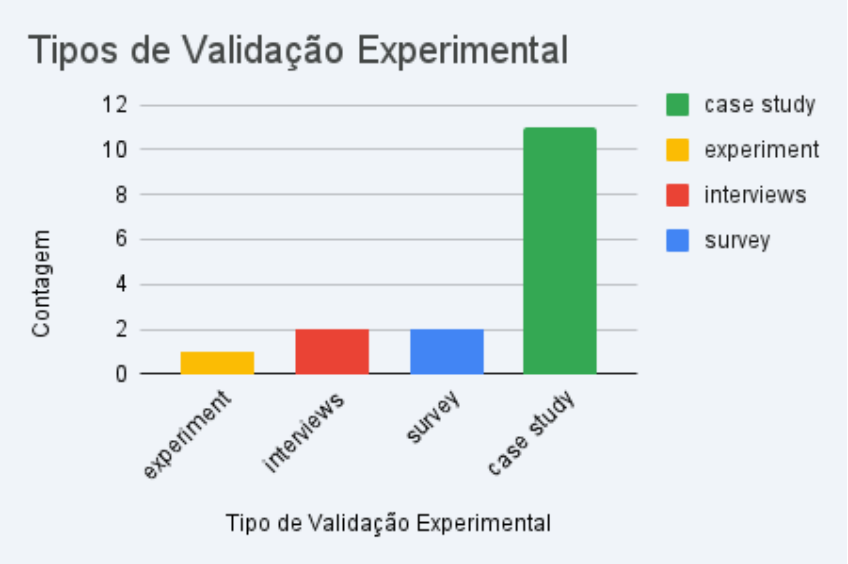


Fonte: Autor

3.3.11 Caso o estudo tenha validação experimental, qual foi o tipo?

Existem diferentes tipos de validação experimental, entre eles o estudo de caso, o experimento, as entrevistas e as pesquisas. Esses foram os métodos empíricos utilizados na vaidação de alguns dos estudos. A quantidade de cada tipo está representada no gráfico da Figura 9, do qual se depreende que a validação por estudo de caso é, com grande margem, o método mais utilizado. Isso se deve, principalmente, à característica do DevSecOps de estar ligado a muitos projetos reais, tanto na academia quanto na indústria, o que cria um ambiente propício para a aplicação de estudos de caso, pois eles possibilitam analisar e obter insights ao estudar o desenvolvimento realizado durante a construção de um produto.

Figura 9 – Tipos de Validação Experimental dos Artigos



Fonte: Autor

4 Planejamento do Estudo de Caso

Neste capítulo é apresentado o planejamento do estudo de caso conduzido na segunda etapa do trabalho, aqui são estabelecidos os conceitos e definições relacionados a esse tipo de estudo, além de conter as estratégias adotadas para efetivamente responder as perguntas de pesquisa.

4.1 Definição

Yin (2015) define o estudo de caso como uma investigação empírica que analisa um determinado fenômeno da atualidade em seu contexto real, ou seja, o pesquisador se insere no ambiente cotidiano onde o objeto de estudo está sendo executado. Essa abordagem é particularmente relevante para a Engenharia de Software, pois frequentemente os fenômenos analisados nessa área são complexos e interligados, o que dificulta analisá-los isoladamente do seu ambiente de execução (RUNESON; HöST, 2009).

Adicionalmente, Yin (2015) destaca que o estudo de caso é flexível e iterativo, isso significa que a estrutura do estudo pode se adaptar no decorrer da pesquisa, pois o pesquisador ao realizar as iterações de coleta e análise dos dados, pode vir a perceber características do caso que não foram possíveis serem identificar a priori.

Ademais, visando assegurar o rigor e a confiabilidade da investigação, os estudos de caso devem coletar dados de múltiplas fontes, dessa forma, ao verificar que conclusões obtidas através de dados obtidos em diferentes locais apontam para o mesmo resultado, aumenta-se robustez e confiabilidade dos resultados, pois esse processo de triangulação diminui a probabilidade de erro ou viés é, além de fornecer uma visão mais ampla sobre o caso (YIN, 2015).

Verifica-se, também, a necessidade de determinar se o estudo será holístico ou incorporado (YIN, 2015). Os estudos de caso holísticos analisam o caso como um todo, ou seja, o fenômeno é visto como um sistema único e integrado, portanto, o pesquisador obtém uma visão ampla e geral do caso. Já os estudos incorporados, tem como principal característica o aprofundamento em múltiplas unidades de análise dentro de um mesmo caso, permitindo uma análise mais aprofundada.

Para além disso, (RUNESON; HöST, 2009) explicita a necessidade de considerar as ameaças a validade do estudo desde o início da pesquisa, pois ignorar tais fatores interfere na confiabilidade dos resultados. A primeira ameaça definida pelo autor está relacionada à validade do constructo, que representa em qual extensão as medidas operacionais realmente representam o que o pesquisador tem em mente e o que está sendo investigado

de acordo com as perguntas de pesquisa. Já a ameaça a validade interna, refere-se ao risco de existir um fator não mapeado pelo pesquisador que cause interferência na relação causal entre os fatores selecionados. As ameaças externas, por outro lado, estão relacionadas a capacidade de generalização do estudo, ou seja, deseja-se que os achados do estudo tenham relevância para casos com características semelhantes. Por fim, a confiabilidade refere-se ao grau em que os dados e a análise dependem de pesquisadores específicos, isto é, se outro pesquisador conduzisse o mesmo estudo, o resultado deveria ser o mesmo.

4.2 Objetivo

Segundo Siavvas et al. (2021b), o desenvolvimento de software seguro é pautado na medição da qualidade da segurança de software, pois assim, é possível avaliar o nível da segurança do produto e conseguir traçar metas para guiar os processos de melhoria contínua do sistema. Porém, por diversas vezes são utilizados critérios de avaliação subjetivos ou que não possuem a devida validação, o que pode acarretar catastofrés relacionadas a segurança do produto. Situação essa que se agrava ao se tratar de práticas emergentes na indústria, como DevSecOps, que apesar de seu destaque no desenvolvimento ágil por muitas vezes carece de avaliação por metodologias apropriadas.

Assim, o objetivo desse estudo consiste em descorbrir como as práticas DevSecOps afetam os aspectos relacionados a segurança de um projeto de software, usando métodos propostos por pesquisadores que compõe o estado da arte da engenharia de software para análise da segurança.

4.3 Caso

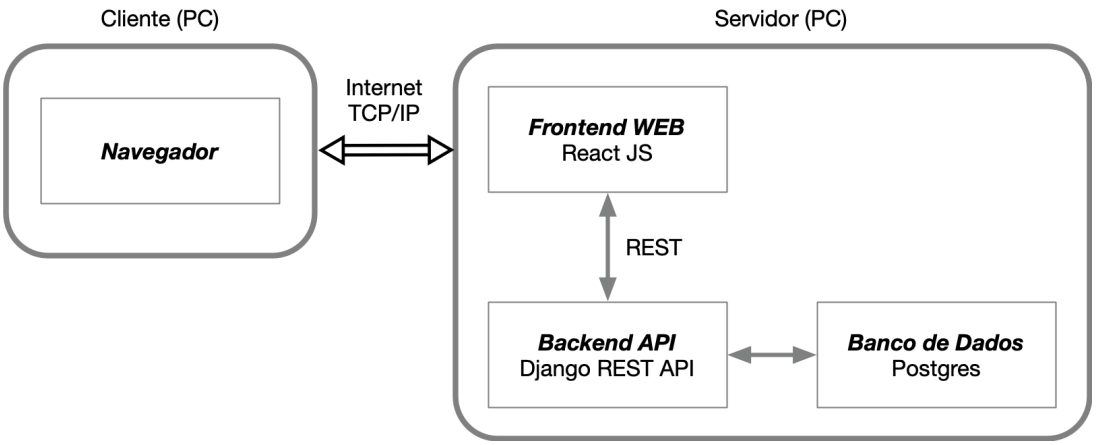
O MEPA[1] é uma plataforma de software livre focada na recomendação, gestão e otimização de contratos de energia, desenvolvida pela Universidade de Brasília (UnB) com financiamento do Governo Federal. A solução foi criada para aprimorar os contratos de fornecimento de energia em instituições públicas, sendo reconhecida pelo Tribunal de Contas da União (TCU) como referência na gestão de energia elétrica para as Instituições Federais de Ensino Superior (IFES).

Empregando inteligência artificial, o sistema realiza uma análise pormenorizada de dados de consumo, como o histórico de faturas das unidades consumidoras, para identificar as opções contratuais mais vantajosas e economicamente eficientes para cada instituição. Adicionalmente, a plataforma também gera relatórios estratégicos com gráficos detalhados sobre o consumo, possibilitando que os gestores a identifiquem padrões e permitindo a tomada de decisões assertivas relaciondas a otimização de custos.

[1] Página do institucional da plataforma

A arquitetura consiste em um frontend construído com o framework Next.js do React, que forma a interface com o usuário. Essa interface se comunica com um servidor Python construído com o framework Django. Os dados são persistidos em um banco de dados PostgreSQL que se conecta ao sistema pelo Django. A comunicação entre a interface e o servidor é feita por uma API REST; dessa forma, a interface gráfica se comunica apenas com o servidor, e este faz a comunicação com o banco de dados, fornecendo os dados necessários para a exibição na interface. Essa arquitetura pode ser observada na Figura 10 e o seu código-fonte está disponibilizado no GitLab[2].

Figura 10 – Arquitera Geral



Fonte: LAPPIS

A API (servidor) é composta por quatro módulos. O primeiro, chamado de Models, contém as definições dos objetos que serão armazenados no banco de dados, incluindo seus atributos e os comportamentos básicos de criar, remover, atualizar e deletar.

O módulo Serializer é responsável por serializar e desserializar os objetos definidos nos Models. Ele traduz e valida os dados entre o formato de comunicação da API (JSON) e o formato mais complexo usado internamente pelo framework.

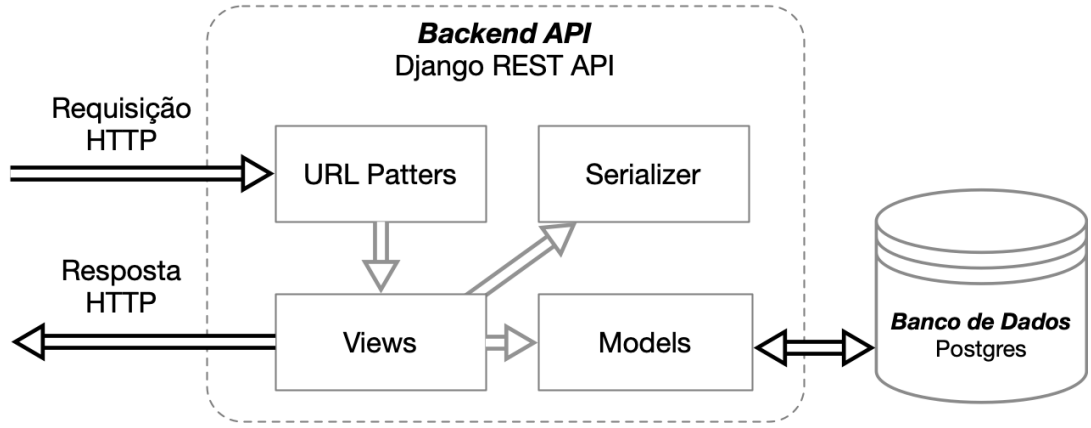
Após a tradução dos objetos, o módulo de visualização (Views) realiza o processamento das requisições e respostas HTTP, que são capturadas pelo módulo de roteamento de requisições. Essa arquitetura está ilustrada na Figura 11.

A Figura 12 representa a arquitetura da interface gráfica (frontend), construída com o framework Next.js e a biblioteca React para a criação de componentes. Utiliza-se também a biblioteca Redux para gerenciar os estados e manter a consistência do fluxo de dados da aplicação. Os componentes são a parte principal da interface, pois é através deles que o usuário interage com o sistema. Eles representam todos os elementos visuais e são utilizados para evitar o acoplamento do código e facilitar a resolução de problemas.

Quando o usuário executa uma ação em um componente, uma função criadora

[2] Repositórios do MEPA

Figura 11 – Arquitera do Backend

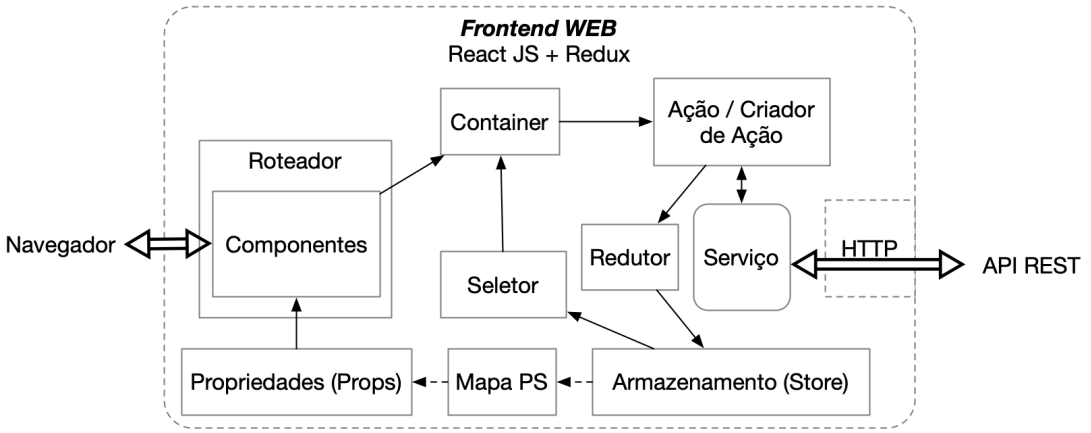


Fonte: LAPPIS

de ação é chamada. Essa função gera um objeto de ação que descreve o que aconteceu. Em seguida, essa ação é enviada para um redutor. O redutor, por sua vez, avalia a ação e determina como o estado da aplicação deve mudar, retornando um novo estado atualizado.

Esse novo estado é salvo na Store, que centraliza e armazena todas as informações de estado da aplicação. Quando a interface detecta uma mudança na Store, os novos dados são passados para os componentes relevantes por meio das Propriedades (Props), garantindo que a interface do usuário reflita o estado atual da aplicação.

Figura 12 – Arquitera do Backend



Fonte: LAPPIS

4.4 Trabalhos Relacionados

Alguns trabalhos, obtidos através do protocolo de revisão da literatura explanado no Capítulo 3, foram utilizado como referência para a realização deste trabalho. Esses artigos foram escolhidos, pois abordam os temas pertinentes para o planejamento e execução da pesquisa, colaborando com o ampliamento das percepções em relação ao tema.

O estudo realizado por Siavvas et al. (2021b) foi o estudo central para a elaboração da monografia. Sua relevância se dá, pois ele apresenta um modelo hierárquico de avaliação de segurança que quantifica a segurança interna do software com base em alertas de análise estática (SAST) e métricas de software. O autor demonstrou que é possível avaliar a segurança de forma confiável e quantiativa usando modelos de qualidade e análise estática.

O trabalho de (ZHANG; ZHANG, 2024), apresenta doze métricas quantitativas de DevSecOps especificamente projetadas para microsserviços web baseados em nuvem. O foco é avaliar a qualidade, segurança e eficiência operacional, com o intuito de auxiliar na tomada de decisões informadas e na melhoria contínua.

A revisão sistemática feita por Rajapakse et al. (2022), identifica os desafios e soluções na adoção do DevSecOps, incluindo seleção de ferramentas, avaliação contínua de segurança e o equilíbrio entre velocidade e segurança

Saeed et al. (2025) aborda técnicas para integrar a segurança no ciclo de desenvolvimento de software, enfatizando a necessidade de uma abordagem colaborativa e ferramentas automatizadas para análise de ameaças e testes de segurança.

4.5 Questão de Pesquisa

Similar ao processo realizado no planejamento da revisão da literatura, a metodologia GQM Basili, Caldiera e Rombach (1994) foi usada para definição das perguntas espécificas derivadas da pergunta principal e suas métricas para conduzir o estudo, de modo a não desviar do objetivo principal e estabelecer a avaliação quantitativa de cada uma das perguntas derivadas da pergunta principal, que agora norteaiam o estudo de caso.

A ISO/IEC 25010 (2023) define como confidencialidade, a capacidade do sistema de impedir o acesso não autorizado às informações, assim, impedindo que os dados privados sejam visíveis para quem não possui as permissões necessárias. Ela será uma a sub-característica de segurança analisadas neste estudo de caso, por meio da análise de vulnerabilidades detectadas no pipeline de CI/CD. Para corroborar com essa análise e fazer a triangularização da coleta de dados, uma avaliação qualitativa com os membros do time é necessária para observar os impactos dessas novas práticas no processo de desenvolvimento.

À vista disso, as seguintes perguntas específicas foram elaboradas:

- Questão Específica 1: A aplicação de práticas DevSecOps permitiu identificar vulnerabilidades de segurança sob as perspectivas da qualidade interna e externa do produto?

  - Métrica 1.1: Média de vulnerabilidades encontradas por ferramentas SAST, por nível de severidade.
  - Métrica 1.2: Média de vulnerabilidades encontradas por ferramentas DAST, por nível de severidade.
  - Métrica 1.3: Proporção de vulnerabilidades em bibliotecas encontradas por ferramentas SCA, por nível de severidade.

- Questão Específica 2: A análise automática da segurança do pipeline e as métricas coletadas ajudaram na tomada de decisões relacionadas ao projeto?
  - Métrica 2.1: Taxa de builds/deploys bloqueados devido à descoberta de vulnerabilidades de criticidade média ou superior.
  - Métrica 2.2: Feedback do time obtido por um questionário sobre os impactos das novas práticas.

4.6 Fonte de Dados

Para coletar os dados necessários para a posterior avaliação das métricas são necessárias diferentes fontes de dados. Primeiramente, as ferramentas SAST e SCA são executadas diretamente no código-fonte. Outra fonte de dados é o sistema em uso, que será usado para a obtençao dos dados referentes às ferramentas DAST que analisam o software em execução. Adiconalmente, o orquestrador de CI/CD atuará como fonte de dados para coletar os tentativas falhas de integração do código, evidênciando a identificação de falhas pelas ferramentas.

Por fim, a equipe técnica será a fonte de dados dos formulários de avaliação ao final do estudo de caso, permitindo a análise qualitativa e triangularização dos resultados.

4.7 Procedimentos

Para atender à complexidade das questões de pesquisa, este trabalho aptou pelo método de estudo de caso incorporado, fundamentado por Yin (2015). Esta abordagem foi escolhida, pois permite que o caso principal seja investigado por meio da análise aprofundada de múltiplas subunidades de análise. Especificamente, o estudo se debruça sobre diferentes faces do projeto, como o impacto na qualidade da segurança interna e externa do produto, e a percepção da equipe sobre as mudanças implementadas. Ao examinar cada um desses componentes seguindo o protocolo de estudo de caso, é possível construir uma visão detalhada e triangulada que fortalece a validade das conclusões e oferece uma resposta mais completa ao problema investigado.

Desse modo, é necessário realizar a integração e configuração das ferramentas de segurança ao pipeline de CI/CD, nesta etapa serão definidos os críterios para o bloqueio ou merge das novas versões do código para que o processo de desenvolvimento não se torne oneroso devido as restrições de segurança.

Após a configuração das ferramentas, é preciso estabeler a linha de base de segurança do projeto, ou seja, avaliar o estado atual do aplicação e gerar o primeiro conjunto de dados que serão usados para comparação na conclusão da monografia.

Então, a execução contínua da análise de segurança será iniciada, durante o segundo semestre letivo de 2025 os desenvolvedores utilizarão a nova pipeline em seu cotidiano, enquanto as métricas são coletads para análise futura.

Ao final da observação, os dados quantitativos serão centralizados e analisados para produzir as métricas obtidas ao final do estudo, além de ser realizada a aplicação do questionário de coleta e da percepção da equipe para obter a opinião dos participantes do estudo.

4.8 Análise de Dados

A análise dos dados quantitativos será fundamentada em estatísticas descritivas e em análise de tendência, visando avaliar a evolução da segurança da aplicação ao longo do tempo. Para as métricas de detecção de vulnerabilidades por ferramentas SAST e DAST, Métrica 1.1 e Métrica 1.2, adaptadas da métrica B.30 Vulnerability landscape da norma ISO/IEC 27004 (2016), será calculado um indicador de tendência para permitir avaliar o volume de vulnerabilidades, por nível de severidade, a cada sprint.

O cálculo do indicador é obtido através da razão entre a média de vulnerabilidades das últimas duas sprints concluídas e a média das últimas seis sprints concluídas, sendo que, um valor inferior a 1.0 indica uma tendência de melhoria, entre 1.0 e 1.3 sinaliza uma tendência de estabilidade e superior a 1.3 indica uma piora significativa.

Para a Métrica 1.3, adaptada da métrica Shared or Unknown Library Ratio proposta por Zhang e Zhang (2024), será calculada a proporção de bibliotecas com vulnerabilidades em relação ao total. Por sua vez, a Métrica 2.1, uma adaptação da métrica Change Fail Percentage elaborado pelo Google (2024), será calculada pela taxa percentual de builds/deploys bloqueados por vulnerabilidades de criticidade média ou superior. O acompanhamento destas métricas será realizado continuamente, visando observar tendência ao longo do tempo.

Para a análise de frequência de respostas do questionário, Métrica 2.2, a frequência de cada das respostas de cada pergunta será registrada, de modo a possibilitar o cálculo da moda, pois ao ter a opinião da maioria dos participantes sobre o tópico solicitado será

possível obter a percepção geral do impacto das atividades realizadas.

4.9 Instrumentação

A instrumentação se refere às ferramentas que serão utilizadas para a realização do estudo de caso, aqui estão definidas as ferramentas usadas para a análise da segurança do sistema, controle de implementação do código, versionamento do código da pipeline feita e coleta de informações da equipe.

O SonarQube é uma ferramenta de open-source de avalição da qualidade e segurança do código-fonte. Ele realiza análise estática para detectar bugs, vulnerabilidades e code smells em várias linguagens. Ele fará parte das ferramentas white-box integradas ao pipeline realizando a análise estática de segurança de aplicação (SAST).

O Trivy é um scanner de segurança de código aberto. Ele é utilizado na análise de composição de software, verificando as bibliotecas de terceiros do projeto, garantindo que componentes externos ao projeto não insiram vulnerabilidades no sistemas. Além disso, ele é capaz de buscar vulnerabilidades em containeres e configurações de infraestrutura como código. Ele complementará o SonarQube como ferramenta white-box.

O OWASP ZAP foi desenvolvido para encontrar problemas de segurança em aplicações web em execução. Ele será empregado para realizar a Análise Dinâmica de Segurança de Aplicação (DAST) em um ambiente de testes.

O GitLab CI/CD é uma ferramenta integrada ao GitLab que permite a automação das etapas do ciclo de vida do software, através dele que as ferramentas de segurança serão executadas automaticamente e ele servirá para bloquear o build/deploy caso vulnerabilidades sejam encontradas.

O Git é um sistema de controle de versão e o GitHub é uma plataforma de hospedagem de código-fonte para controle de versão com Git. Eles serão utilizados para o versionamento e armazenamento dos artefatos por este estudo de caso.

O Google Forms permite a criação rápida e fácil de formulários online, além de permitir a gestão e análise dos resultados. Ele será utilizado para a aplicação do questionário que coleta os dados qualitativos da equipe.

4.10 Ameaças à Validade do Estudo

Conforme destacado por Runeson e Höst (2009), é essencial considerar as ameaças à validade desde o início da pesquisa para garantir a confiabilidade dos resultados. Nesta seção, são identificadas as principais ameaças à validade deste estudo de caso e as estratégias adotadas para sua mitigação.

4.10.1 Ameaças à Validade do Constructo

As métricas selecionadas podem não capturar completamente os aspectos de segurança que se pretende avaliar. Para mitigar essa ameaça, foram selecionadas métricas baseadas em padrões internacionais como a ISO/IEC 27004 (2016) e em trabalhos do estado-da-arte da literatura, como o trabalho de Zhang e Zhang (2024).

4.10.2 Ameaças à Validade Interna

Mudanças na equipe e alterações no escopo do projeto podem influenciar os resultados. Para mitigar essa ameaça, será mantido um registro detalhado de quaisquer mudanças significativas durante o período do estudo.

4.10.3 Ameaças à Validade Externa

O estudo está limitado a um único projeto e o MEPA possui características específicas que podem não ser adequadas para outros sistemas de software. Para mitigar essa ameaça, é fornecida uma descrição detalhada do contexto do estudo para que outros pesquisadores possam avaliar a aplicabilidade dos resultados em seus próprios contextos.

4.10.4 Ameaças à Confiabilidade

Variações na forma como os dados são coletados e analisados podem afetar os resultados. Para mitigar essa ameaça, serão estabelecidos procedimentos claros e, quando possível, automatizados, para a coleta de dados, minimizando a intervenção manual e sistematizando a análise.

Referências

BASILI, V. et al. Goal question metric (gqm) approach. In: _____. [S.l.: s.n.], 1994. ISBN 9780471028956. Citado na página 32.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The Goal Question Metric Approach. In: MARCINIAK, J. J. (Ed.). Encyclopedia of Software Engineering. [S.l.]: Wiley, 1994. v. 1, p. 528–532. Citado 2 vezes nas páginas 30 e 61.

BOEHM, B. W. Characteristics of Software Quality. North-Holland, 1978. Disponível em: <https://books.google.com.br/books?id=jLFXswEACAAJ>. Citado na página 27.

DROMEY, R. G. A model for software product quality. IEEE Trans. Softw. Eng., IEEE Press, v. 21, p. 146–162, 2 1995. ISSN 0098-5589. Disponível em: <http://dx.doi.org/10.1109/32.345830>. Citado na página 27.

DíAZ, J. et al. Harmonizing devops taxonomies - a grounded theory study. 08 2022. Citado na página 29.

ELSEVIER. Banco de dados de resumos e citações organizado por especialistas. 2024. Disponível em: <https://www.periodicos.capes.gov.br/>. Citado na página 37.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering: A roadmap and agenda. The Journal of Systems & Software, v. 123, p. 176–189, 2017. Citado na página 28.

FRANçA, B. B.; JUNIOR, H. J.; TRAVASSOS, G. Characterizing devops by hearing multiple voices. In: . [S.l.: s.n.], 2016. Citado na página 29.

G1, P. 2025. Https://g1.globo.com/sp/sao-paulo/noticia/2025/07/04/policia-ataque-hacker-ao-sistema-que-liga-bancos-ao-pix.ghtml. Acessado: 25-07-2025. Citado na página 28.

Google. DORA. [S.l.], 2024. Acessado em: 20 de julho de 2025. Disponível em: <https://dora.dev>. Citado 3 vezes nas páginas 40, 50 e 63.

ISO/IEC-25010. ISO/IEC 25010 System and software quality models. [S.l.], 2010. Citado na página 27.

ISO/IEC 25010. ISO/IEC 25010:2023 Systems and software engineering — Systems and software Quality Requireme Geneva, CH, 2023. Citado na página 61.

ISO/IEC-27001. Information technology – Security techniques – Information security management systems – Requirements. 2022. Available at: https://www.iso.org/standard/70032.html. Citado na página 27.

ISO/IEC 27004. ISO/IEC 27004:2016 Information technology — Security techniques — Information security manage Geneva, CH, 2016. Citado 2 vezes nas páginas 63 e 65.

ISO/IEC-27034. Information technology – Security techniques – Application security – Part 1: Overview and concepts. 2011. ISO/IEC 27034-1:2011. Citado na página 27.

ISO/IEC-9126. ISO/IEC 9126. Software engineering – Product quality. [S.l.]: ISO/IEC, 2001. Citado na página 27.

JOSHI, H. A secure software development methodology for enterprise business applications. In: . [s.n.], 2024. p. 7 – 12. Cited by: 1. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85217554758&doi=10.1109%2fICODSE63307.2024.10829891&partnerID=40&md5=aa77a0827de37592c910cb0f32cbb0fc>. Citado na página 51.

KITCHENHAM, B.; CHARTERS, S. et al. Guidelines for performing systematic literature reviews in software engineering. Keele, UK, 2007. Citado 2 vezes nas páginas 32 e 37.

KUDRIAVTSEVA, A.; GADYATSKAYA, O. You cannot improve what you do not measure: A triangulation study of software security metrics. In: . [s.n.], 2024. p. 1223 – 1232. Cited by: 6; All Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85194836591&doi=10.1145%2f3605098.3635892&partnerID=40&md5=0beecf2d8290ab7b673bdda3b6300a57>. Citado 2 vezes nas páginas 29 e 51.

KUSHWAHA, M. K.; DAVID, P.; SUSEELA, G. Automation and devsecops: Streamlining security measures in financial system. In: . [s.n.], 2024. Cited by: 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85205770194&doi=10.1109%2fCONECCT62155.2024.10677271&partnerID=40&md5=6dd76e3d104de49d5559093b3f58c303>. Citado na página 52.

LANGE, F.; KUNZ, I. Evolution of secure development lifecycles and maturity models in the context of hosted solutions. Journal of Software: Evolution and Process, v. 36, n. 12, 2024. Cited by: 0; All Open Access, Hybrid Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85200057398&doi=10.1002%2fsmr.2711&partnerID=40&md5=9b0595cd71d28171da66b90852ed7301>. Citado 3 vezes nas páginas 35, 49 e 51.

LUZ, W. P.; PINTO, G.; BONIFÁCIO, R. Adopting devops in the real world: A theory, a model, and a case study. J. Syst. Softw., v. 157, 2019. Disponível em: <https://doi.org/10.1016/j.jss.2019.07.083>. Citado na página 29.

LóPEZ, L. et al. Quality measurement in agile and rapid software development: A systematic mapping. Journal of Systems and Software, v. 186, p. 111187, 2022. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121221002661>. Citado na página 28.

MASOOD, A.; JAVA, J. Static analysis for web service security - tools & techniques for a secure development life cycle. In: . [S.l.: s.n.], 2015. Citado na página 48.

MCCALL, J.; RICHARDS, P. K.; WALTERS, G. F. Factors in software quality. volume i, ii and iii. concepts and definitions of software quality. US Rome Air Development Center Reports, US Department of Commerce, USA, p. 168, 1977. Citado na página 27.

Mitre Corporation. Common Weakness Enumeration (CWE™). 1999. Acessado em: 24-07-2025. Disponível em: <http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm>. Citado na página 28.

NOCERA, S. et al. A large-scale fine-grained empirical study on security concerns in open-source software. In: . [s.n.], 2023. p. 418 – 425. Cited by: 3. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85183330517&doi=10.1109%2fSEAA60479.2023.00069&partnerID=40&md5=cedd775ef204cc60b75b12f44bdda858>. Citado na página 52.

OWASP. OWASP Software Assurance Maturity Model (SAMM). 2020. Https://owaspsamm.org/. Acessado: 24-07-2025. Citado na página 28.

OWASP. OWASP Top Ten 2021. 2021. <https://owasp.org/TopTen/>. OWASP Top Ten 2021 Report. Citado na página 28.

PAI, M. et al. Clinical research methods. THE NATIONAL MEDICAL JOURNAL OF INDIA, v. 17, n. 2, 2004. Citado 2 vezes nas páginas 32 e 37.

PAI, M. et al. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. National Medical Journal of India, v. 17, n. 2, p. 86–95, mar-apr 2004. PMID: 15141602. Citado na página 38.

RAJAPAKSE, R. N. et al. Challenges and solutions when adopting devsecops: A systematic review. Information and Software Technology, v. 141, 2022. Cited by: 91; All Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85114377924&doi=10.1016%2fj.infsof.2021.106700&partnerID=40&md5=8d37a7b4dea325db0f7ecd9a9ed17a0e>. Citado 6 vezes nas páginas 29, 47, 48, 49, 51 e 61.

RAMIREZ, A.; AIELLO, A.; LINCKE, S. J. A survey and comparison of secure software development standards. In: . [s.n.], 2020. Cited by: 17. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85100614059&doi=10.1109%2fCMI51275.2020.9322704&partnerID=40&md5=8b4d0a69a6b627d3d34f5be03aa2c6ba>. Citado na página 52.

RANGNAU, T. et al. Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In: . [s.n.], 2020. p. 145 – 154. Cited by: 57; All Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85096513818&doi=10.1109%2fEDOC49727.2020.00026&partnerID=40&md5=c40a98ef4d8eff4fb8b08e234290f023>. Citado na página 48.

RUNESON, P.; HöST, M. Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Engg., Kluwer Academic Publishers, USA, v. 14, n. 2, p. 131–164, abr. 2009. ISSN 1382-3256. Disponível em: <https://doi.org/10.1007/s10664-008-9102-8>. Citado 2 vezes nas páginas 57 e 64.

SAEED, H. et al. Review of techniques for integrating security in software development lifecycle. Computers, Materials and Continua, v. 82, n. 1, p. 139 – 172, 2025. Cited by: 2; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85214507365&doi=10.32604%

2fcmc.2024.057587&partnerID=40&md5=16c958450978504c24f727dbb66e06d5>. Citado 2 vezes nas páginas 52 e 61.

SIAVVAS, M. et al. A hierarchical model for quantifying software security based on static analysis alerts and software metrics. Software Quality Journal, Kluwer Academic Publishers, USA, v. 29, n. 2, p. 431–507, jun. 2021. ISSN 0963-9314. Disponível em: <https://doi.org/10.1007/s11219-021-09555-0>. Citado 2 vezes nas páginas 27 e 29.

SIAVVAS, M. et al. A hierarchical model for quantifying software security based on static analysis alerts and software metrics. Software Quality Journal, v. 29, n. 2, p. 431 – 507, 2021. Cited by: 21; All Open Access, Green Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85106207514&doi=10.1007%2fs11219-021-09555-0&partnerID=40&md5=b0ecf71985d4ef4c10438eabe73da349>. Citado 2 vezes nas páginas 58 e 61.

UOL, P. 2025. Https://economia.uol.com.br/noticias/estadao-conteudo/2025/07/05/entenda-golpe-que-desviou-r-800-milhoes-da-cm.htm. Acessado: 25-07-2025. Citado na página 28.

VALOR-ECONOMICO, P. 2025. Https://valor.globo.com/financas/noticia/2025/07/02/hackers-teriam-desviado-milhoes-de-reais-apos-invadir-empresa-que-conecta-instituicoes-financeiras-ao-pix.ghtml. Acessado: 25-07-2025. Citado na página 28.

WOHLIN, C. et al. Experimentation in Software Engineering. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435. Citado na página 32.

YIN, R. Estudo de Caso - 5.Ed.: Planejamento e Métodos. Bookman Editora, 2015. ISBN 9788582602324. Disponível em: <https://books.google.com.br/books?id=EtOyBQAAQBAJ>. Citado 3 vezes nas páginas 32, 57 e 62.

ZHANG, J. Y.; ZHANG, Y. Quantitative devsecops metrics for cloud-based web microservices. IEEE Access, v. 12, p. 160317 – 160342, 2024. Cited by: 2; All Open Access, Gold Open Access. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85208091278&doi=10.1109%2fACCESS.2024.3486314&partnerID=40&md5=cc74abbeeef3802bac7d726302f6c5e2>. Citado 5 vezes nas páginas 40, 49, 61, 63 e 65.

Apêndices

APÊNDICE A – Primeiro Apêndice

Texto do primeiro apêndice.

APÊNDICE B – Segundo Apêndice

Texto do segundo apêndice.

Anexos

ANEXO A – Primeiro Anexo

Texto do primeiro anexo.

ANEXO B – Segundo Anexo

Texto do segundo anexo.