UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO ACADÊMICO DE RECIFE

Bruno Martins Santos

Segurança em Pipelines de CI/CD: Análise de Riscos, Detecção de Anomalias e Notificações através de um Chatbot Inteligente

Recife

2023

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

SISTEMAS DE INFORMAÇÃO

Bruno Martins Santos

Segurança em Pipelines de CI/CD: Análise de Riscos, Detecção de Anomalias e Notificações através de um Chatbot Inteligente

Monografia apresentada ao curso Sistemas de Informação da Universidade Federal de Pernambuco, como requisito para a obtenção do Título de Bacharel em Sistemas de Informação.

Orientador(a): Vinicius Cardoso Garcia

RECIFE

2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Santos, Bruno Martins Santos.
Segurança em Pipelines de CI/CD: Análise de Riscos, Detecção de Anomalias e Notificações através de um Chatbot Inteligente / Bruno Martins Santos Santos. - Recife, 2023.
79 p : il.

Orientador(a): Vinicius Cardoso Garcia Garcia
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2023.
Inclui referências, anexos.

1. Segurança. 2. Pipelines. 3. Chatbot. 4. CI/CD. I. Garcia, Vinicius Cardoso Garcia. (Orientação). II. Título.

000 CDD (22.ed.)

Bruno Martins Santos

Segurança em Pipelines de CI/CD: Análise de Riscos, Detecção de Anomalias e Notificações através de um Chatbot Inteligente

Monografia apresentada ao curso Sistemas de Informação da Universidade Federal de Pernambuco, como requisito para a obtenção do Título de Bacharel em Sistemas de Informação

Aprovado em: 25/09/2023.

BANCA EXAMINADORA

_____

Profº. Dr. Vinicius Cardoso Garcia (Orientador)
Universidade Federal de Pernambuco

_____

Profº. Dra. Carla Taciana Lima Lourenco Silva (Examinador Interno)
Universidade Federal de Pernambuco

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que desempenharam um papel fundamental na realização deste trabalho e na minha jornada acadêmica.

Primeiramente, minha gratidão vai para minha querida família, cujo amor, apoio incondicional e crença em mim foram o alicerce que me permitiu chegar até aqui. Vocês são minha fonte constante de inspiração.

Aos meus amigos, que compartilharam risadas, desafios e momentos inesquecíveis ao longo dos anos, obrigado por enriquecerem minha vida de maneiras que palavras não podem expressar completamente. Um agradecimento especial a Giovani Albuquerque Villarim De Siqueira, que me acompanhou nesta jornada desde o primeiro período, compartilhando as alegrias e desafios da jornada acadêmica. Sua amizade e apoio foram inestimáveis.

Ao professor Vinicius Cardoso Garcia, meu orientador, sou grato por sua orientação, paciência e dedicação ao longo deste processo.

Não posso deixar de mencionar todo o corpo docente que contribuiu para a minha formação. Seus ensinamentos e conhecimentos moldaram meu entendimento e abriram portas para meu desenvolvimento acadêmico.

Este trabalho é o resultado do esforço coletivo de muitas pessoas, e estou profundamente grato a todos que fizeram parte dessa jornada. Obrigado por tornarem possível a realização deste sonho.

"A verdadeira sabedoria é saber que você não sabe tudo." - Sócrates

RESUMO

Os pipelines de CI/CD desempenham um papel importante na construção de um software, garantindo rapidez, qualidade e confiabilidade nas entregas de novas funcionalidades. No entanto, a segurança em pipelines de CI/CD é um pouco negligenciada, seja pelo custo operacional ou pela mão de obra necessária para tal. Essa negligência pode resultar em violações de dados e interrupções de serviço e perda de reputação. Identificar e resolver problemas de segurança de forma eficaz nesse contexto é de extrema importância. Dito isto, o objetivo deste trabalho é abordar essa problemática da segurança com foco em realizar uma análise e exploração dessas falhas, além da proposição de um chatbot para auxiliar na detecção de anomalias e fornecer alertas e notificações. O estudo envolve a seleção e configuração de ferramentas adequadas, a implementação de um protótipo de chatbot e a análise dos resultados obtidos, com o objetivo de melhorar a detecção precoce de vulnerabilidades, reduzir o tempo de resposta e contribuir para o avanço das práticas de segurança nesse contexto crítico de desenvolvimento de software.

Palavras-chave: Segurança, Pipelines, Chatbot, CI/CD.

ABSTRACT

The CI/CD pipelines play an important role in the software construction, ensuring speed, quality and reliability in the delivery of new features. However, security in CI/CD pipelines is often overlooked either by operational costs or the labor force needed to do it. This negligence can result in data violations, service interruptions and reputation loss. Effectively identifying and resolving security issues in this context is very important. Thus, the objective of this work is to address this security problem by focusing on analyzing and exploring these vulnerabilities in addition to proposing the implementation of a chatbot to assist in anomalies detection and provide real-time alerts and notifications. The study involves selecting and configuring appropriate tools, implementing the chatbot and analyzing the obtained results with the aim to improve early vulnerabilities detection, reducing response time and contributing to the advancement of security practices in this critical software development context.

Keywords: Security, Pipelines, Chatbot, CI/CD

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES

CI/CD	Continuous Integration / Continuous Delivery
EC	Entrega Contínua
IC	Integração Contínua

SUMÁRIO

1 INTRODUÇÃO

Com os computadores se tornando mais acessíveis a partir dos anos 2000 e por consequência a popularização da internet a demanda por softwares só aumentou. Com esse crescente aumento, métodos de desenvolvimentos de softwares como o modelo cascata e o modelo em V acabaram se tornando obsoletos, seja pela ineficiência em atender a exigência do mercado ou pelo alto risco que as metodologias possuíam devido a segregação das equipes. Como dito por Mayank et al, “No ciclo de vida de desenvolvimento convencional, equipes diferentes desempenham seus papéis em seus níveis.”(GOKARNA, 2021). Equipes separadas tornam o ciclo de vida do produto mais longo e também a comunicação entre as equipes se torna pobre (GOKARNA, 2021). Nos últimos anos, o desenvolvimento de software tem passado por uma transformação significativa, impulsionado pela adoção de práticas de integração contínua e entrega contínua (CI/CD) que advém das metodologias ágeis. DevOps é uma versão estendida das metodologias ágeis (GOKARNA, 2021). O que o DevOps visa é a Integração Contínua (CI), Entrega Contínua (CD), feedbacks com mais velocidade e segurança (GOKARNA, 2021). Os pipelines de CI/CD são componentes fundamentais nesse processo, permitindo a automação de tarefas de compilação, teste e implantação de software. No entanto, à medida que esses pipelines se tornam mais complexos e sofisticados, a segurança se torna uma preocupação cada vez mais premente. Em um mundo digital cada vez mais conectado e dependente de software, garantir a segurança dos pipelines de CI/CD se torna uma necessidade imperativa.

Dito isto, este trabalho abordará as principais vulnerabilidades e desafios de segurança enfrentados nos pipelines de CI/CD, além de investigar as ferramentas disponíveis no mercado para análise e detecção de anomalias. Será proposta uma arquitetura para o chatbot de segurança, considerando a integração com as ferramentas existentes para facilitar as análises realizadas pelo projeto.

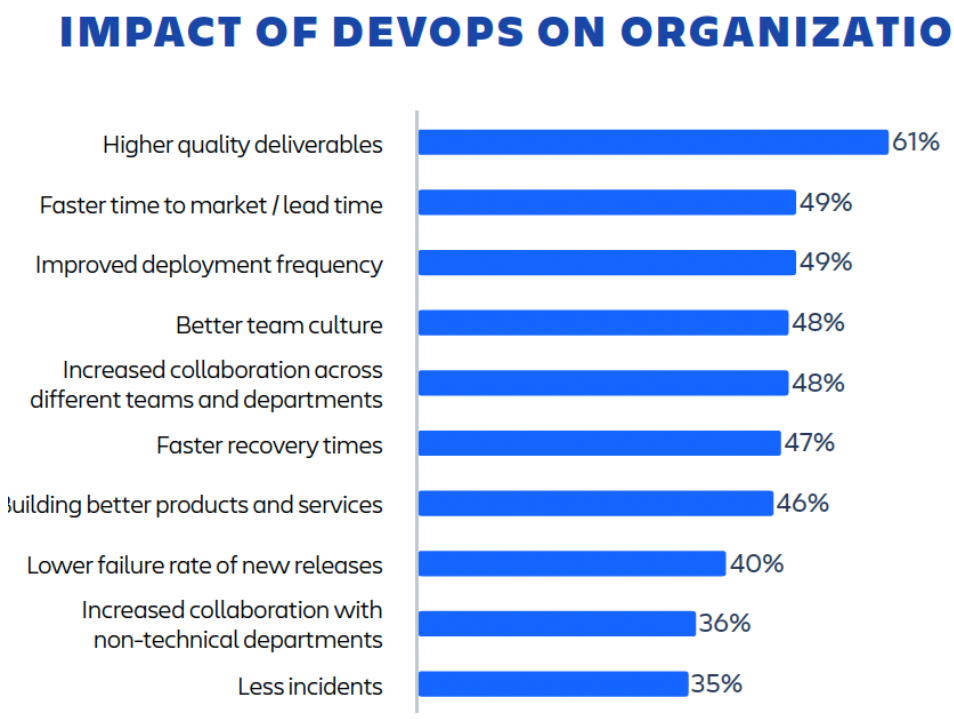
Ao final do estudo, espera-se que as soluções propostas contribuam para aprimorar a segurança dos pipelines de CI/CD, proporcionando uma detecção mais

eficiente de problemas de segurança, redução do tempo de resposta e maior confiabilidade no processo de desenvolvimento de software. Além disso, o trabalho pretende incentivar a discussão e a conscientização sobre a importância da segurança em pipelines de CI/CD, fornecendo insights valiosos para profissionais e pesquisadores da área.

1.1 Motivação

Até certo ponto, a inclusão da entrega contínua e integração contínua, pipelines (CI/CD), no fluxo de desenvolvimento se tornaram uma garantia para a qualidade do projeto (PULGAR, 2022); Esse fenômeno pode ser notado na Figura 1, que é uma representação gráfica extraída de uma pesquisa conduzida em 2020 pela Atlassian, em colaboração com a CITE Research, envolvendo 500 profissionais do campo em questão. De acordo com os resultados dessa pesquisa, impressionantes 61% dos participantes relataram um aumento notável na qualidade das entregas.

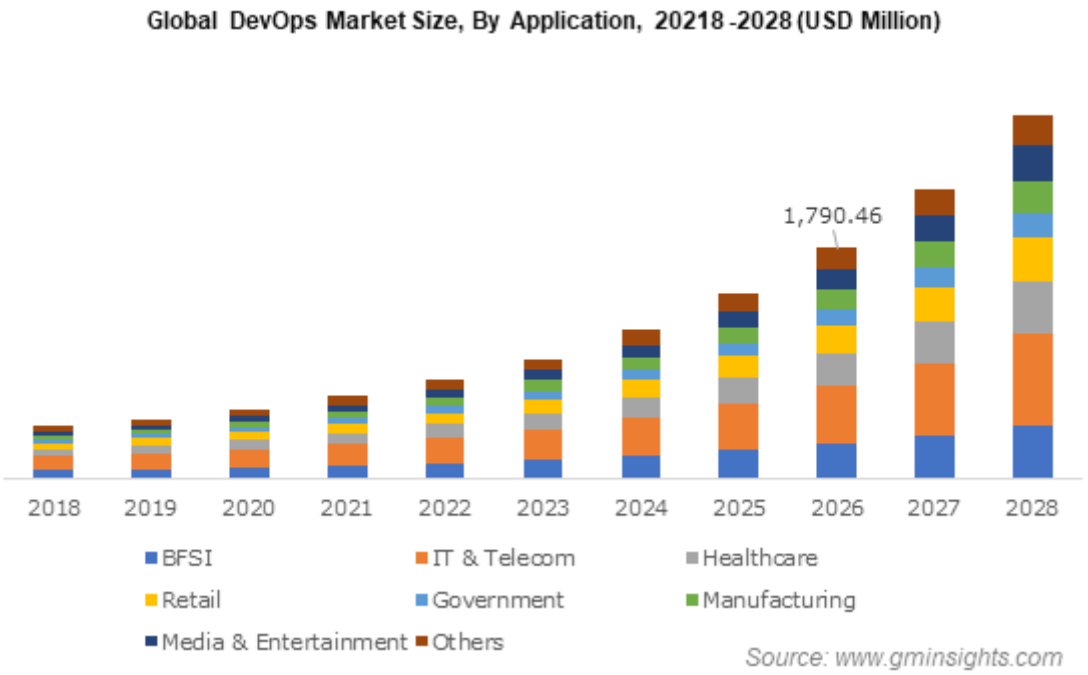
Figura 1: Impacto do DevOps nas organizações



Fonte: (Atlassian, 2020)

Uma pesquisa realizada pelo GMI (Global Market Insights) mostrou que o mercado de soluções DevOps está previsto para ultrapassar os $30 bilhões de dólares em 2028; Isso devido ao crescimento da adoção de desenvolvimento ágil dos softwares, redução do ciclo de desenvolvimento e a ideia de uma comunicação mais eficiente. Na figura 2, retirada do GMI é possível notar o crescimento do mercado DevOps em diversos segmentos de mercado.

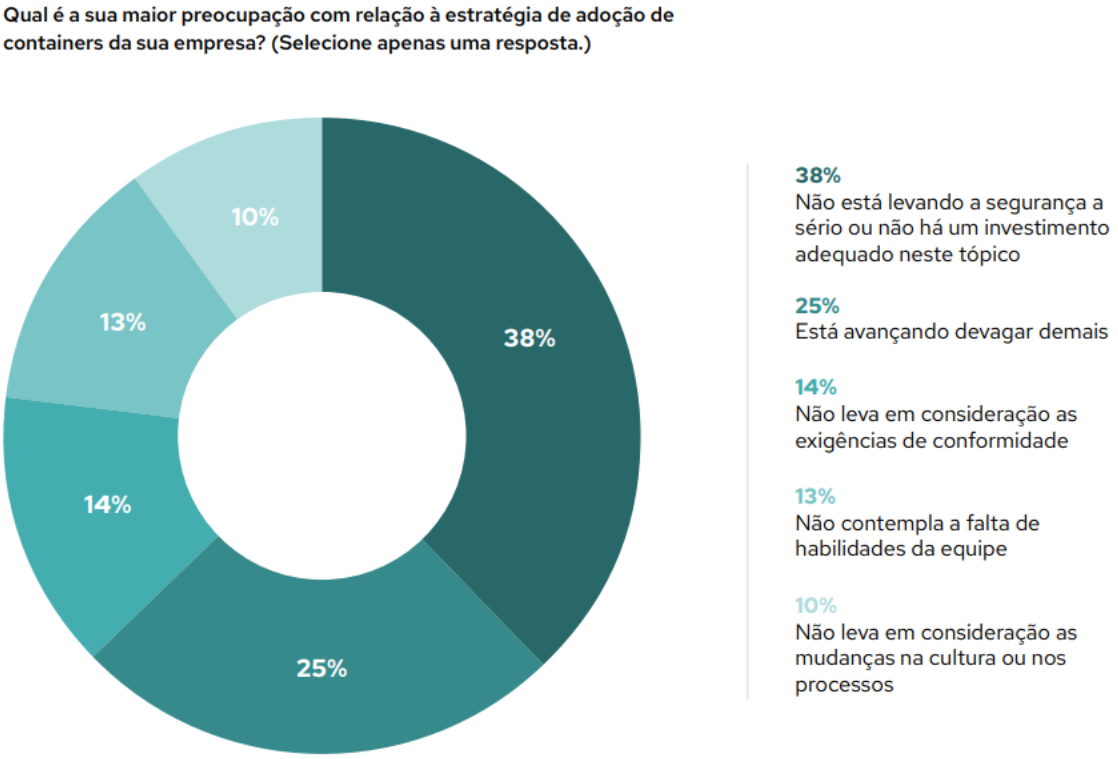
Figura 2: Previsão de crescimento do mercado DevOps até 2028



Fonte: (GMI, 2022)

Ainda com todos esses dados, quando se trata de segurança é possível notar um certo desdém das organizações para com este tópico, um relatório da redhat sobre o estado da segurança do Kubernetes, realizado em 2023, demonstrou que dos 600 profissionais de DevOps, engenharia e segurança entrevistados 38% demonstraram que a segurança não está sendo levada a sério ou que o investimento está inadequado, como visto na figura 3.

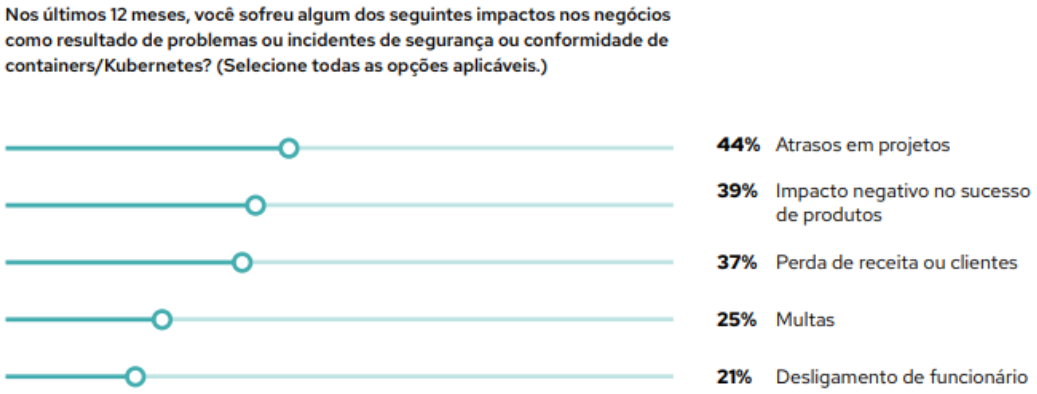
Figura 3: Preocupação com a segurança de containers



Fonte: (RedHat, 2023)

O preço que a organização pode pagar por isso é o atraso em lançamento de aplicações assim como a exposição de brechas para ataques, incidentes em ambiente de execução e/ou perda de receita ou clientes

Figura 4: Resultado dos impactos devido a problemas de segurança



Fonte: (RedHat, 2023)

Além disso, as pipelines por si só passam por um ciclo contínuo e incremental de evolução, uma vez que, por essência, são parte do fluxo de trabalho definido pelos princípios DevOps (PULGAR, 2022). Segundo o autor U. Zdun et al (ZDUN, 2023), apesar de diversos estudos publicados abordando as melhores práticas, a arquitetura de microservices é um desafio quando se trata da segurança. Isso, devido ao tamanho e a complexidade dos sistemas de micro serviços, sua natureza políglota e a necessidade de evolução contínua e o frequente lançamento desses sistemas (ZDUN, 2023).

Segundo (GRUHN, 2013) para sistemas de multi-instâncias os problemas de segurança podem ser até mais prejudiciais uma vez que uma instância maliciosa pode atacar várias outras instâncias devido ao sistema de CI.

Dito isso, nesse contexto, como devemos fazer ou o que devemos seguir para prevenir e identificar precocemente possíveis falhas de segurança nas pipelines de CI/CD? A questão é que não existe bem uma receita de bolo para ser seguida; existem sim diversos estudos abordando como melhorar as pipelines seguindo as melhores práticas, como visto em Flora (2020), porém às vezes devido ao contexto é preciso optar por uma abordagem diferente dessas práticas. Neste trabalho, através da seleção e configuração adequada de ferramentas.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo principal deste trabalho é compreender as complexidades da segurança em pipelines de CI/CD e propor uma arquitetura de chatbot que quando implementada, servirá como uma ferramenta de suporte para os desenvolvedores. Na seção 4, discutiremos a implementação detalhada e apresentaremos um protótipo da ferramenta**.**

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Desenvolver uma Arquitetura de Chatbot: Projetar uma arquitetura robusta e eficiente para o chatbot que seja capaz de interagir com os pipelines de CI/CD; Que seja possível de garantir e acompanhar a evolução do projeto de maneira orgânica e organizada.
- Implementar um Protótipo Funcional do Chatbot: Desenvolver e demonstrar um protótipo funcional da ferramenta chatbot que ilustra sua capacidade de detectar anomalias e fornecer notificações.
- Avaliar a Eficácia do Chatbot na Melhoria da Segurança: Conduzir testes e análises para avaliar como a presença do chatbot influencia a análise de segurança dos pipelines de CI/CD, testando sua capacidade de reduzir riscos e melhorar a resposta a incidentes.

Esses objetivos específicos guiarão nossa pesquisa e esforços de implementação, permitindo uma abordagem sistemática para alcançar nosso objetivo geral.

2 FUNDAMENTAÇÃO TEÓRICA

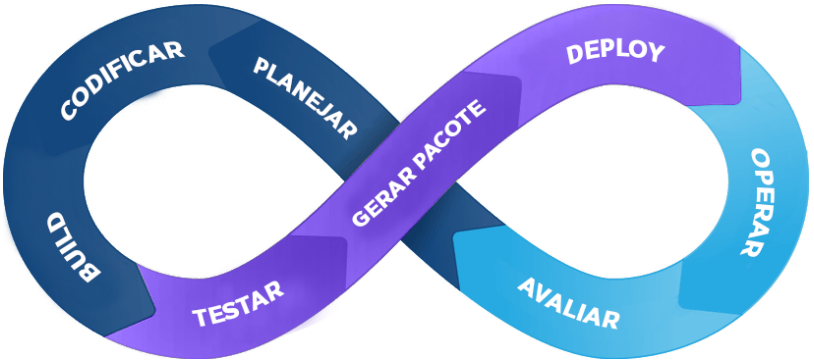
Esta seção pretende detalhar um pouco do escopo de DevOps e DevSecOps entrando em alguns pontos importantes sobre a segurança em pipelines, pontos estes que serão necessários para o entendimento da solução bem como o entendimento da motivação do trabalho proposto.

Com isso em mente, abordaremos nesta seção as ferramentas escolhidas para o desenvolvimento deste trabalho, juntamente com a motivação por trás de suas escolhas.

2.1 Cultura DevOps

DevOps unifica os times de desenvolvimento e operações (GOKARNA, 2021). A cultura DevOps é uma abordagem colaborativa que visa promover uma integração mais estreita entre equipes de desenvolvimento (Dev) e operações (Ops) no processo de entrega de software, conforme discutido por Jabbari et al. Essa cultura enfatiza a comunicação, a colaboração, a automação e a melhoria contínua conforme destacado também por Jabbari et al em seu trabalho. Com isso, as equipes de desenvolvimento e operações trabalham de forma contínua, compartilhando responsabilidades e objetivos em comum. Geralmente esse ciclo de trabalho é atrelado ao símbolo do infinito para representar a ideia de continuidade como visto na figura 5, onde é possível observar o fluxo contínuo da prática DevOps.

Figura 5: Ciclo DevOps



Fonte: Solvimm (2023)

Na metodologia observam-se diversos termos frequentemente usados pelos desenvolvedores. Para este trabalho é válido destacar os seguintes:

- Entrega Contínua (EC) | Continuous Delivery (CD)
- Integração Contínua (IC) | Continuous Integration (CI)

2.1.1 Integração Contínua

Nos últimos 15 anos, a integração contínua se tornou uma prática importante no desenvolvimento de software (GRUHN, 2013). Com isso os desenvolvedores são

capazes de manter os códigos sempre atualizados várias vezes ao dia no mesmo repositório central. DevOps permite a integração contínua de todos os processos envolvidos no desenvolvimento do produto e assim todo o processo é feito por uma única equipe durante todo o ciclo (EBERT, 2016). Essa abordagem permite a prematura identificação de erros de integração, através das execuções automatizadas dos testes, facilitando a resolução dos mesmos e evitando deploys problemáticos para produção.

2.1.2 Entrega Contínua

Entrega contínua, como o nome já diz, é uma prática que vai além da IC, envolve a automatização de todo o processo de desenvolvimento do software. Aqui já observamos a implementação de pipelines automatizadas que permitem a entrega em produção de forma rápida e confiável. Nessa pipeline, está incluso a execução dos testes automatizados, a implantação em ambientes de desenvolvimento e produção de forma controlada e o controle dos ambientes e acessos. Os benefícios em adotar a entrega contínua estão relacionados a acelerar o tempo de lançamento para o mercado, criar o produto correto, melhor produtividade e eficiência, lançamentos confiáveis, melhor qualidade no produto e satisfação do cliente (CHEN, 2017).

2.2 DevSecOps

DevSecOps é uma abordagem inovadora para o desenvolvimento de software que combina três conceitos fundamentais: Desenvolvimento (Dev), Segurança (Sec) e Operações (Ops). DevSecOps refere-se à engenharia de uma plataforma de desenvolvimento e entrega de software que incorpora segurança em todas as fases. (SCANLON, 2022). Ou seja, enquanto o DevOps está relacionado a parte de operações, o DevSecOps tem seu foco na segurança.

DevSecOps trata do uso da metodologia DevOps para segurança. (CARTER, 2022), ou seja, além de englobar os passos existentes no DevOps, aqui a camada

de segurança se vê mais presente, por isso que muitas vezes o DevSecOps é atrelado a evolução do DevOps, uma vez que a segurança é um dos principais pilares do desenvolvimento e entrega contínua de software nesse contexto ágil.

2.3 Chatbots

Chatbots são programas de computador que interagem com os usuários por meio de linguagens naturais. Esta tecnologia começou na década de 1960; o objetivo era ver se os sistemas de chatbot poderiam enganar os usuários, dizendo que eram humanos reais. (SHAWAR, 2007).

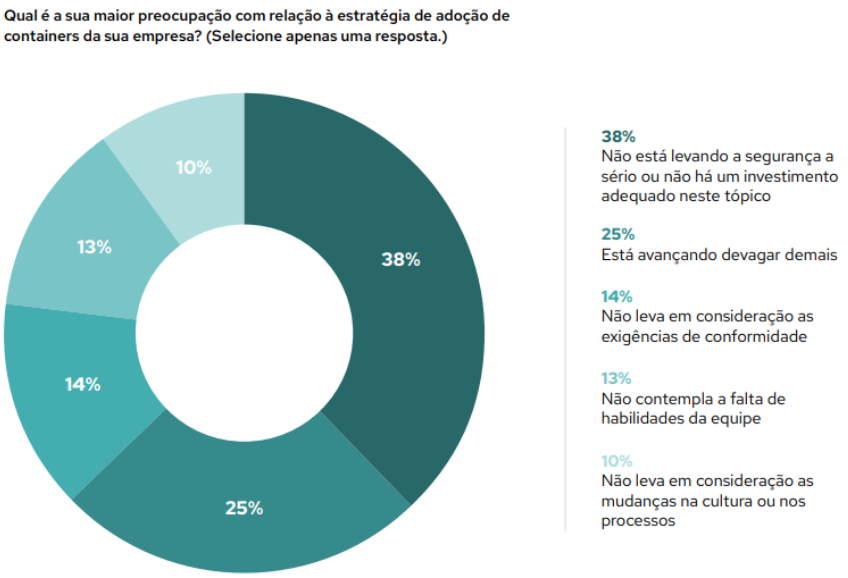
Os chatbots têm uma longa história de evolução, desde os primeiros sistemas baseados em regras como a ELIZA, até os mais inteligentes alimentados por IA como o chat-GPT. A integração de chatbots em ambientes de desenvolvimento pode ser uma extensão dessa evolução, proporcionando uma interface de comunicação eficiente entre equipes de desenvolvimento e sistemas automatizados.

2.2 Importância e desafios da segurança em pipelines de CI/CD

A segurança em pipelines de CI/CD (Integração Contínua e Entrega Contínua) é de extrema importância para garantir a integridade, confiabilidade e proteção dos projetos de software. CI/CD é uma prática de desenvolvimento de software que automatiza o processo de integração, teste e entrega de código em produção, permitindo que as equipes de desenvolvimento entreguem novas funcionalidades e correções com mais frequência e confiabilidade.

Ainda assim é possível observar inúmeros casos de atrasos em projetos devido a preocupações com a segurança, seja por não ter sido levada a sério ou por investimento inadequado neste tópico como apresentado na figura 6 retirada de uma pesquisa da redhat com cerca de 500 profissionais da área de TI.

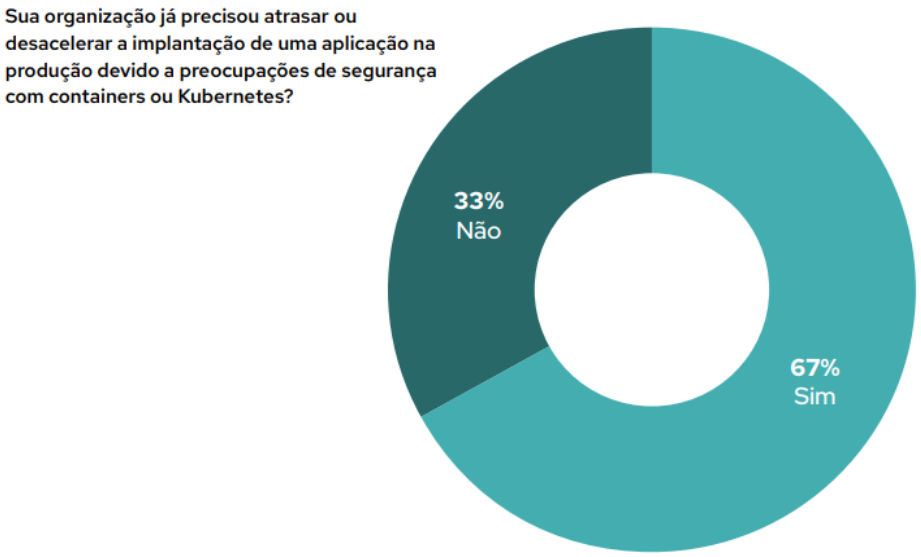
Figura 6: Preocupação com a estratégia de adoção de containers na empresa



Fonte: (RedHat, 2023)

Outrossim, a seguinte figura retirada da mesma fonte anterior, demonstra uma porcentagem bastante expressiva de profissionais da área de TI, que afirmaram que a organização já precisou atrasar ou desacelerar a implantação de um aplicativo devido a questões de segurança.

Figura 7: Atrasos devido a preocupação com segurança

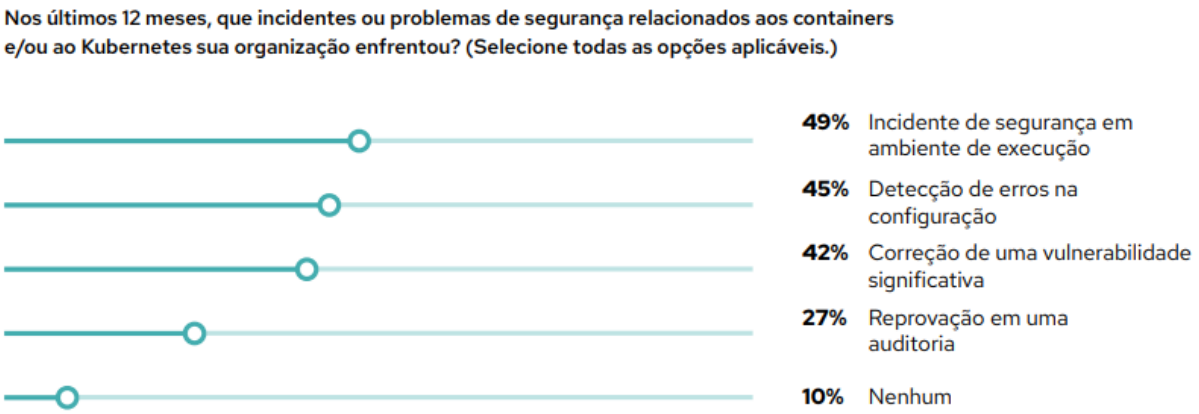


Fonte: (RedHat, 2023)

2.2.1 Detecção precoce de vulnerabilidades

A integração contínua permite que as equipes integrem seu trabalho e verifiquem (com testes de unidade) se tudo está em condições ideais antes de avançar para a próxima etapa (ROMERO, 2022). A segurança no pipeline de CI pode ajudar a detectar vulnerabilidades no código o mais cedo possível, antes que elas sejam enviadas para a produção. Isso permite que os desenvolvedores corrijam as falhas antes que elas se tornem um problema real. No entanto, ainda é possível observar que muitos dos problemas continuam a ocorrer em ambiente de produção como demonstrado na figura a seguir.

Figura 8: Porcentagem de Incidentes ocorridos em diferentes etapas



Fonte: (Redhat, 2023)

2.2.2 Automação de testes de segurança

Incorporar testes automatizados de segurança no pipeline de CI/CD ajuda a identificar problemas de segurança com mais facilidade e rapidez. Testes como análise de código estático, análise de composição de software, análise de vulnerabilidades, entre outros, podem ser automatizados e integrados ao processo de desenvolvimento. Uma ferramenta como o SonarCloud pode ser usada para uma

análise de código estática. De acordo com a própria documentação do sonarcloud “O SonarCloud usa técnicas de ponta em análise de código estático para encontrar problemas e possíveis problemas no código que você e sua equipe escrevem.” (SONARCLOUD, 2023)

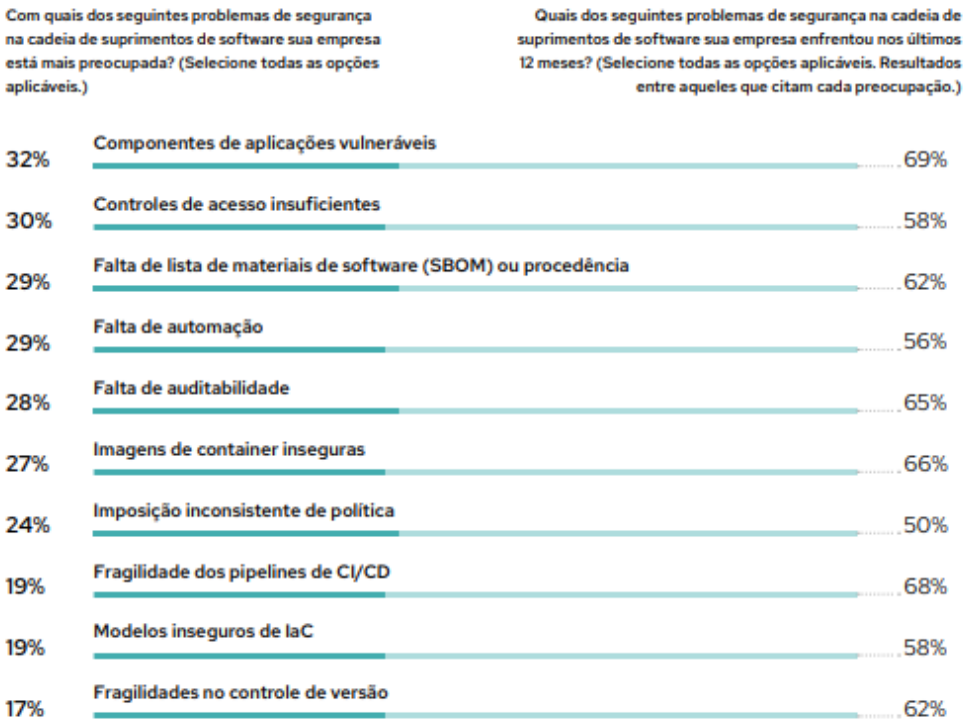
2.2.3 Monitoramento de ameaças em tempo real

Como abordado na seção 2.2.1, quase metade dos incidentes relatados acontecem em ambiente de execução; Dito isto, um chatbot que esteja integrado com uma pipeline de CI/CD pode ser configurado para alertar os desenvolvedores e administradores do sistema sobre atividades suspeitas, como tentativas de acesso não autorizado ou outras atividades maliciosas. Isso permite uma resposta rápida e reduz a janela de oportunidade para possíveis ataques.

2.2.4 Controle de acesso

Os pipelines de CI/CD frequentemente exigem o uso de credenciais e segredos para acessar ambientes de desenvolvimento, sistemas de controle de versão e serviços em nuvem, por isso a assegurar que apenas as pessoas autorizadas tenham acesso aos recursos e ferramentas do pipeline pode evitar que indivíduos mal-intencionados interfiram no processo de desenvolvimento. A gestão adequada dessas informações sensíveis é essencial para evitar vazamentos e comprometer a segurança. A seguinte figura representa justamente essa problemática na cadeia de suprimentos de algumas organizações, onde demonstra uma porcentagem expressiva no problema do controle de acesso e algumas outras fragilidades concomitantes.

Figura 9: Problemas recorrentes na cadeia de suprimentos



Fonte: (RedHat, 2023)

2.2.5 Resiliência a ataques

Com uma abordagem de segurança robusta, os pipelines de CI/CD podem ser projetados para serem mais resilientes a ataques. Isso pode incluir implementar práticas de desenvolvimento seguro, bem como o uso de tecnologias de isolamento e segurança.

2.3 Escolha das ferramentas

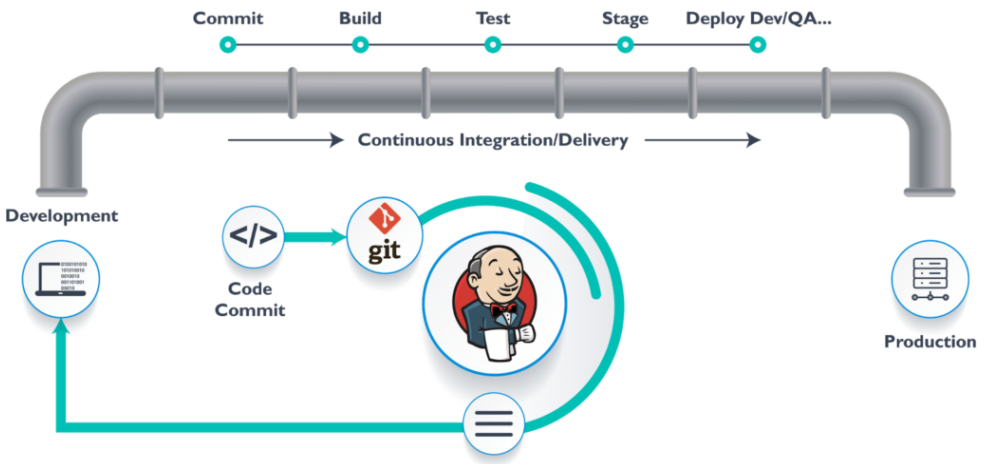
Nesta seção, será abordado os motivos e as ferramentas que foram escolhidas para a realização deste trabalho.

2.3.1 Jenkins

O Jenkins é uma ferramenta de código aberto amplamente utilizada para automação de CI/CD. Ele permite que você automatize o processo de integração contínua, entrega contínua e implementação contínua, facilitando a construção, teste e implantação de código de maneira mais rápida e confiável. Além disso, possui uma vasta quantidade de plugins disponíveis que o tornam altamente customizável e adequado para várias necessidades de desenvolvimento. O Jenkins foi escolhido por sua confiabilidade, escalabilidade e vasta comunidade de usuários. Sua natureza de código aberto também permite uma integração mais fácil com outras ferramentas e sistemas, tornando-o uma escolha popular para equipes de desenvolvimento de todos os tamanhos.

A figura a seguir apresenta uma demonstração visual do fluxo do jenkins e seu processo de pipeline.

Figura 10: Jenkins Integração Contínua



Fonte: Medium - Continuous integration with jenkins and other tools(AWS vproject)

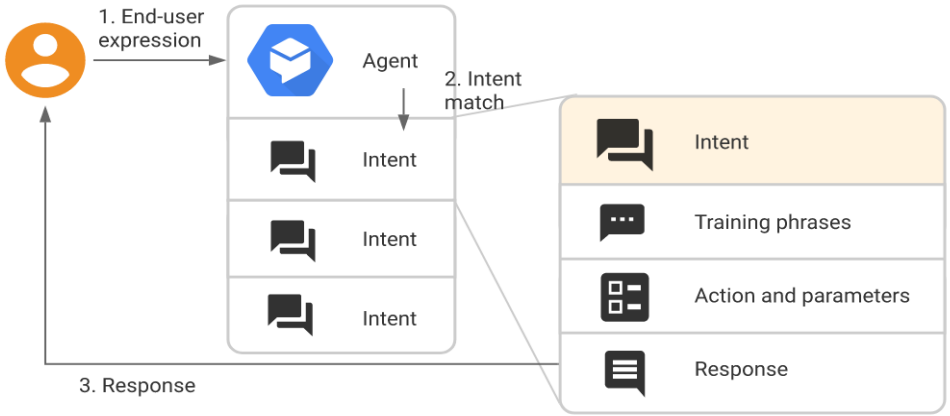
2.3.2 Dialogflow

O Dialogflow, agora conhecido como "Google Cloud Dialogflow", é uma plataforma de desenvolvimento de chatbots e agentes de conversação alimentada por IA (Inteligência Artificial). Ele oferece recursos avançados de processamento de linguagem natural (PLN) que permitem criar interfaces de conversação para aplicativos, sites e dispositivos.

O Dialogflow foi escolhido por sua capacidade de permitir a interação do usuário através de comandos de voz e texto, facilitando a comunicação entre os usuários e o sistema que está sendo desenvolvido. Sua integração com outras ferramentas do Google Cloud também facilita a construção de aplicações mais ricas em recursos de conversação.

Abaixo está uma representação dessa arquitetura do dialogflow, onde é possível observar o fluxo de interação entre o usuário e o chatbot, onde o usuário interage com o agente do dialogflow configurado e esse agente é responsável por fazer o match das intenções, processamento da linguagem, identificar parâmetros e ações e retornar a resposta para o usuário final.

Figura 11: Arquitetura Dialogflow



Fonte: Google Cloud, Documentação do Dialogflow.

Para aprimorar ainda mais a usabilidade da ferramenta e permitir ações avançadas e personalizadas, a plataforma oferece a capacidade de integrar a webhooks, o qual foi utilizado neste projeto. “Webhook é uma função de callback baseada em HTTP que viabiliza a comunicação entre duas interfaces de programação de aplicações (APIs). Os webhooks são usados por várias web apps para receber pequenos volumes de dados de outras aplicações e também podem ser usados para acionar workflows de automação em ambientes GitOps.” (RedHat, 2023).

Desse modo, para facilitar a realização das análises e centralizar as informações obtidas de maneira mais amigável para o usuário, um webhook foi configurado para lidar com as demandas vindas do chatbot.

2.3.3 GitHub

O Github é uma plataforma de hospedagem de código baseada em Git, que oferece controle de versão distribuído e recursos de colaboração para equipes de desenvolvimento (GITHUB, 2023). É amplamente utilizado para hospedar repositórios de código, revisar e discutir mudanças de código (pull requests) e gerenciar projetos de software. O Github é uma escolha popular para hospedagem de código devido à sua interface amigável, facilidade de colaboração entre desenvolvedores e recursos de controle de versão robustos. Além disso, a integração com outras ferramentas de CI/CD, como o Jenkins, é bem estabelecida, permitindo uma experiência de desenvolvimento contínua e integrada.

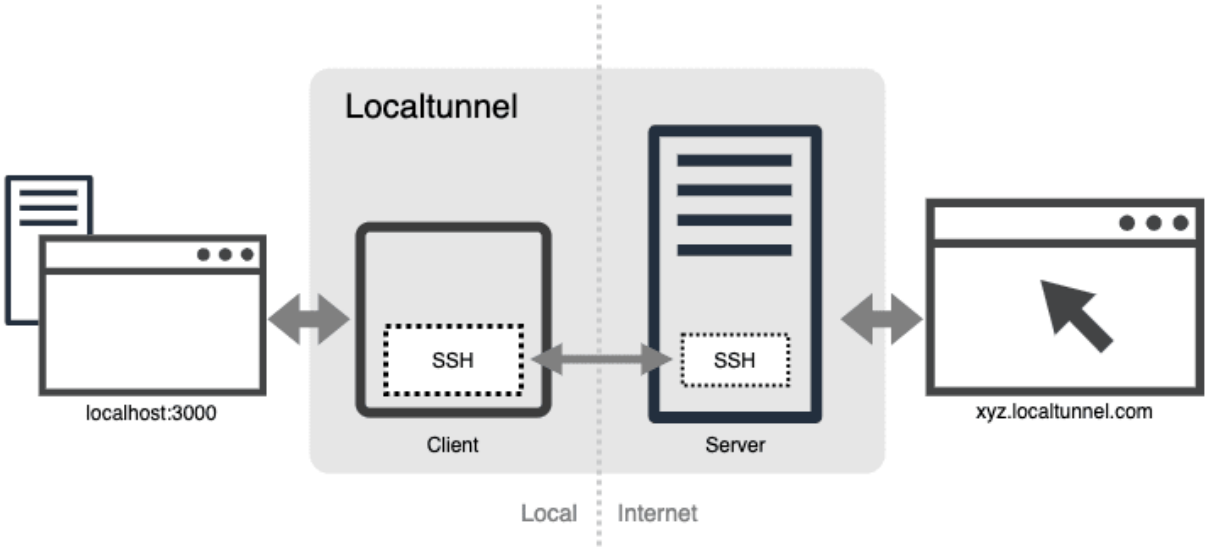
2.3.4 Ngrok

O Ngrok é uma ferramenta que permite criar túneis seguros para expor localmente um servidor ou aplicativo para a internet (NGROK, 2023). Ele facilita o compartilhamento temporário de aplicativos hospedados localmente, permitindo que outras pessoas acessem esses recursos através de URLs públicas temporárias.

Abaixo está uma demonstração visual dessa arquitetura que é simples, onde o usuário expõe seu localhost para a web usando o tunelamento pelos servidores do

ngrok, os quais já possuem toda uma camada de segurança como certificados ssh, https e o handshake; Apesar de ter toda essa segurança envolvida o ngrok é recomendado para testes e não para ser usado em uma aplicação real.

Figura 12: Arquitetura do Ngrok



Fonte: Dev Community, Building your own Ngrok in 130 lines

O Ngrok é uma escolha útil para testar e compartilhar temporariamente recursos locais com outras pessoas, como chatbots em desenvolvimento ou servidores que ainda não foram implantados em produção. Sua facilidade de uso e a capacidade de criar túneis seguros foram decisivas para a escolha dessa ferramenta.

3 METODOLOGIA

3.1 Visão Geral Arquitetura

Toda a arquitetura foi montada com base nas ferramentas escolhidas, adotando também o modelo “Clean Architecture” proposto por Robert Cecil Martin no seu livro Clean Architecture: A Craftsman's Guide to Software Structure and Design, devido a limitações não seguiremos à risca o que é proposto mas utilizaremos os padrões mais bem vistos e usados no mercado. A arquitetura proposta tem como

objetivo aprimorar a segurança em Pipelines de Integração Contínua e Entrega Contínua (CI/CD) por meio da utilização de um Chatbot Inteligente, proporcionando uma interação mais eficiente com os desenvolvedores e auxiliando na detecção de possíveis anomalias e riscos de segurança durante todo o processo de desenvolvimento e entrega de software.

3.1.1 Chatbot vs Chatbot inteligente

Para entendermos melhor a escolha de um chatbot inteligente, primeiro é preciso entender que não há diferenças significativas entre um chatbot e um chatbot inteligente;

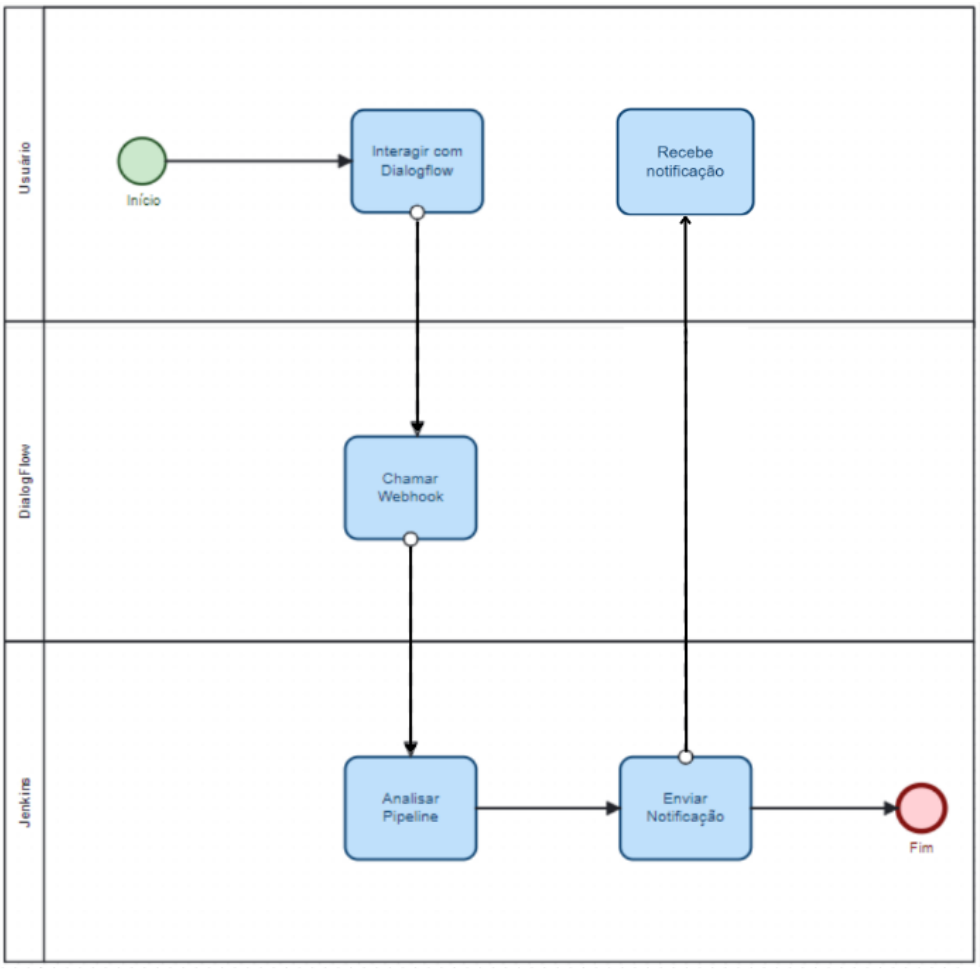
Um sistema chatbot é um programa de software que interage com os usuários usando linguagem natural. (SHAWAR, 2007). Ele pode ser programado para responder a perguntas e executar tarefas específicas com base em comandos fornecidos pelo usuário. No entanto, sistemas primitivos como ELIZA usavam correspondência de palavras-chave e identificação mínima de contexto e não tinham a capacidade de manter uma conversa acontecendo. (RADZIWILL, 2017)

Um chatbot inteligente, por outro lado, é apenas um tipo mais avançado de chatbot que utiliza tecnologias de inteligência artificial (IA) para compreender e responder às consultas dos usuários de maneira mais sofisticada. Esses chatbots são capazes de processar linguagem natural, o que significa que podem compreender e interpretar o contexto das mensagens dos usuários, em vez de simplesmente corresponder palavras-chave.

3.1.1 Input do Usuario

O fluxo de input referente a figura 13 foi mapeado com base no input do usuário. Assim que o usuário dispara alguma intenção específica para o chatbot o webhook identifica a ação e procede executando uma ação para a API do jenkins a qual realiza o que foi solicitado.

Figura 13: Fluxo Input do Usuário



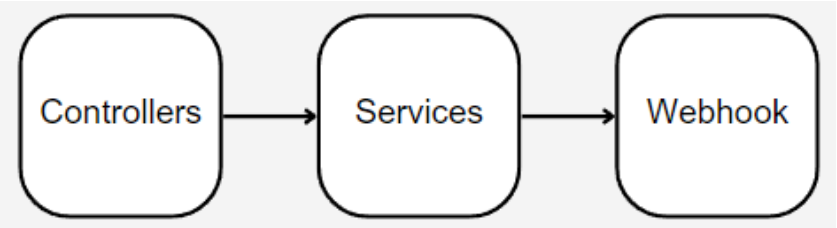
Fonte: O Autor (2023)

Explicação do fluxo: O usuário solicita ao bot para que uma build da pipeline seja executada; O dialogflow captura essa intenção e envia para o webhook; O webhook faz a identificação da ação e realiza uma chamada http para a API do Jenkins; O Jenkins executa a build da pipeline gerando um relatório básico; O relatório é retornado para o usuário.

3.2 Componentes da Arquitetura

Além das ferramentas escolhidas é proposto também uma organização de pastas para o webhook assim como a definição de dois módulos que se dividem em: módulo de detecção de anomalias e o módulo de detecção de vulnerabilidades;

Figura 14: Arquitetura Proposta



Fonte: O autor (2023)

A modelagem da arquitetura vista na figura 14 foi elaborada com base nos princípios da “Clean Architecture”, possuindo 3 camadas de abstração.

- **Controllers:** Nesta camada, colocaremos os controladores responsáveis por gerenciar as solicitações recebidas pelo Webhook. Isso inclui a tarefa de lidar com a interação com o Chatbot, ou seja, quando o sistema recebe uma solicitação, o controlador neste nível determina como essa solicitação deve ser tratada e como o Chatbot deve ser acionado.
- **Services:** A camada de serviços abriga os módulos principais do Webhook, onde a maior parte da lógica da aplicação está localizada. É aqui que ocorre o processamento das informações recebidas e a tomada de decisões sobre como proceder com base nos dados recebidos. Os serviços são responsáveis por executar ações específicas, como analisar anomalias e riscos, bem como interagir com outros componentes, como o Chatbot.
- **Webhook**: O Webhook é responsável por enviar notificações e informações relevantes sobre anomalias e riscos detectados pelo sistema. O Webhook atua como um elo de ligação entre o sistema e o Chatbot, garantindo que as informações sejam entregues e processadas corretamente.

3.2.1 Possíveis benefícios da Arquitetura Proposta:

Melhoria na Comunicação: A utilização do Chatbot Inteligente melhora a comunicação entre os desenvolvedores e o sistema de CI/CD. O Chatbot deve fornecer informações claras e em tempo real sobre o status do pipeline, permitindo uma visão geral e detalhada do processo de desenvolvimento. Detecção Precoce de Anomalias e Riscos: Com o monitoramento contínuo do Chatbot Inteligente, é

possível detectar e notificar os desenvolvedores sobre anomalias e riscos de segurança de forma proativa, antes que eles se tornem problemas maiores. Agilidade na Resolução de Problemas: Ao receber notificações em tempo real sobre falhas ou riscos, a equipe de desenvolvimento pode agir rapidamente, reduzindo o tempo de recuperação e aumentando a segurança do pipeline de CI/CD. Automatização de Tarefas: O Chatbot Inteligente automatiza a comunicação e notificações, permitindo que a equipe se concentre em tarefas mais complexas e críticas.

3.3 Pipeline de CI/CD

A definição dos estágios da pipeline não segue um padrão rígido, uma vez que cada projeto apresenta suas particularidades e, portanto, os estágios são configurados de acordo com as necessidades específicas de cada um. Neste contexto, para a realização dos testes neste trabalho, foram criadas duas pipelines destinadas a dois projetos distintos. Para alcançar uma varredura completa e obter os melhores resultados possíveis, é fundamental a instalação de alguns plugins no Jenkins. Isso justifica, em parte, a escolha dessa ferramenta para a execução desse processo.

3.4 Pesquisa e Modelo de Testagem

A pesquisa realizada neste trabalho teve como foco a segurança em pipelines de Integração Contínua e Entrega Contínua (CI/CD). Para atingir os objetivos propostos, uma abordagem de pesquisa mista, combinando elementos qualitativos e quantitativos foi realizada.

Coleta de Dados:

A coleta de dados para esta pesquisa envolveu várias etapas. Inicialmente, foi realizada uma revisão bibliográfica, explorando fontes acadêmicas, artigos científicos e recursos online relacionados à segurança em pipelines de CI/CD. Isso permitiu compreender o estado da arte, identificar tendências e mapear as vulnerabilidades comuns existentes. A coleta de dados para esta pesquisa também

se beneficiou da análise de fóruns especializados e da documentação das ferramentas utilizadas no projeto.

Modelo de testagem:

Para a realização dos testes do projeto, a arquitetura do chatbot foi integrada às ferramentas de detecção de anomalias previamente identificadas. Durante essa etapa, a integração foi testada para garantir que os dados fossem corretamente transmitidos entre o chatbot e as ferramentas de detecção, e que as notificações fossem geradas de maneira adequada.

Testes de Simulação de Anomalias: Para avaliar a eficiência na detecção do chatbot, conduzimos testes de simulação de anomalias. Nesses cenários simulados, foram replicadas atividades anômalas, como a execução de pipelines com duração excepcionalmente longa, a interrupção de builds e a ocorrência de falhas no processo de construção. Estes testes permitiram verificar a capacidade do chatbot em identificar e alertar prontamente sobre comportamentos e situações não usuais, contribuindo para aprimorar a segurança e a confiabilidade dos pipelines de CI/CD.

Testes de simulação de Risco: Para avaliar o nível de detecção de possíveis brechas de segurança, foram conduzidos testes simulando situações críticas. Nesses testes, recuperamos informações cruciais do Jenkins e do SonarQube, incluindo alertas sobre plugins desatualizados e possíveis vulnerabilidades presentes nessas versões. Além disso, acessamos informações do SonarQube relacionadas a "code smells" (mau cheiro de código), bugs e vulnerabilidades em aberto. Esses testes de simulação de risco visam testar a capacidade do sistema em identificar e alertar sobre ameaças potenciais, ajudando a aprimorar a segurança dos pipelines de CI/CD.

Os resultados dos testes podem ser vistos na seção 5 “Testes e Resultados” acompanhados por figuras retiradas da implementação demonstrando as detecções realizadas.

4 IMPLEMENTAÇÃO

4.1 Requisitos Funcionais

O requisito funcional descreve os principais recursos desejados de um sistema (SHAH, 2016). Ou seja, são requisitos / funcionalidades essenciais para a prototipação de um mvp de projeto.

Quadro 1: Requisitos Funcionais

ID	Requisitos Funcionais
RF1	O chatbot permite ao usuário escolher o projeto do Jenkins
RF2	O chatbot deve ser capaz de obter informações sobre a pipeline escolhida
RF3	O usuário deve ser capaz de integrar com outras ferramentas a partir do chatbot
RF4	O chatbot deve ser capaz de receber inputs do usuário por meio de um webhook
RF5	O chatbot deve ser capaz de analisar os dados do Jenkins e notificar o usuário de possíveis anomalias
RF6	O chatbot deve ser capaz de analisar os dados do Jenkins e notificar o usuário de possíveis riscos de segurança
RF7	O chatbot deve ser capaz de sugerir melhorias para mitigar os riscos

Fonte: O autor (2023)

4.2 Requisitos Não Funcionais

Requisitos não funcionais cobrem todas as demandas adicionais, que são especificadas para um software e que não influenciam diretamente a saída resultante dos dados (KUGELE, 2008)

Quadro 2: Requisitos não Funcionais

ID	Requisitos Funcionais
RNF1	O chatbot deve ser capaz de se integrar com o Jenkins
RNF2	O chatbot deve responder de maneira rápida e eficiente a consultas e eventos
RNF3	O sistema deve permitir atualizações regulares e a incorporação de novos recursos
RNF4	A arquitetura do projeto deve possuir uma boa arquitetura, seguido de padrões limpos de código permitindo uma boa escalabilidade do projeto
RNF5	O sistema deve ser capaz de entender a linguagem do usuário

Fonte: O autor (2023)

4.3 Dialogflow

Nesta seção será discutida a implementação do Dialogflow como parte integrante da solução de chatbot para aprimorar a segurança em pipelines de CI/CD. O Dialogflow desempenha um papel fundamental na interação com os desenvolvedores e na interpretação das intenções dos usuários, gerando respostas relevantes e ações apropriadas.

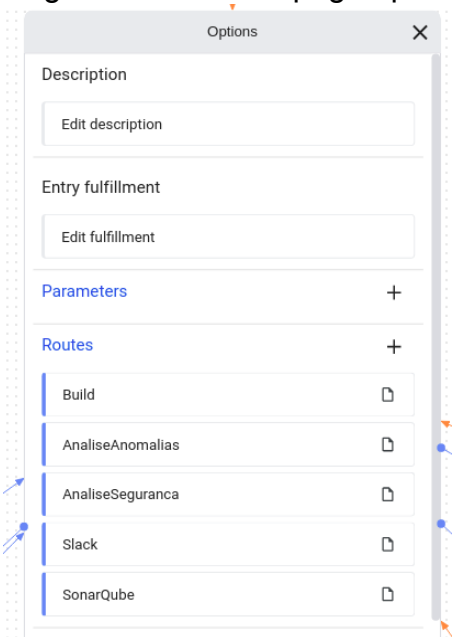
4.3.1 Configuração do Agent

Inicialmente, foi criado um agente no Dialogflow, denominado "Intellibot", que será responsável por receber e processar as solicitações dos usuários. Foram definidas as intenções específicas do agente, mapeando as possíveis consultas e comandos relacionados à segurança nos pipelines de CI/CD. Isso inclui intenções para consulta de status e solicitação de informações sobre incidentes.

4.3.2 Pages e Intents

Para a comunicação com o dialogflow e o desenvolvimento do fluxo foi necessário criar diversas opções de intents no agent para que fosse possível entender perfeitamente as solicitações dos usuários. As imagens ilustradas a seguir foram retiradas do console do dialogflow, onde foi realizado toda a parte de configuração das intenções. Cada um dos cards ilustrado em cada imagem representa uma page (página). Uma conversa do Dialogflow CX pode ser descrita e visualizada como uma máquina de estado. Os estados de uma sessão do CX são representados por páginas. (DIALOGFLOW, 2023). As páginas quando combinadas formam um fluxo inteiro de conversação para qual o fluxo foi projetado. A figura abaixo ilustra como são os cards das pages quando clicados.

Figura 15: Card da page options



Fonte: O autor (2023)

Observando a imagem acima, é possível observar alguns campos e os mais significativos para nós são os seguintes: 'Entry Fulfillment', 'Parameters' e 'Routes'.

- **Entry Fulfillment:** Este campo permite configurar a resposta que o chatbot fornecerá caso não seja necessária uma consulta ao webhook. Isso é especialmente útil em cenários em que uma resposta imediata pode ser gerada sem a necessidade de acessar dados externos ou executar lógica personalizada.

- **Parameters:** A seção 'Parameters' é crucial para capturar e fazer referência a valores fornecidos pelo usuário durante uma sessão de conversação. Isso possibilita a personalização das respostas e a capacidade de manter contextos ao longo da interação.

- **Routes:** O campo 'Routes' (rotas) é fundamental para definir o direcionamento das interações. Ele permite estabelecer regras e condições para guiar o fluxo da conversa com base nas intenções identificadas, nas ações realizadas e nas informações coletadas durante a interação, garantindo uma experiência de conversação fluida e personalizada para o usuário. As rotas levam a uma outra page.

Com esse breve entendimento de como o dialogflow pode ser configurado, a seguir será explicado brevemente sobre o que cada page configurada no fluxo proposto para esse projeto realiza, seguida pela imagem do console do dialogflow para fins visuais.

Start Page - Assim que o usuário interage com o chatbot, é disparada uma mensagem de boas-vindas;

URL - Page responsável por solicitar a informação da URL do jenkins para o usuário;

JenkinsUrl - Responsável por salvar a informação da URL e redirecionar o usuário para a opção de fornecer as informações para se conectar com o jenkins;

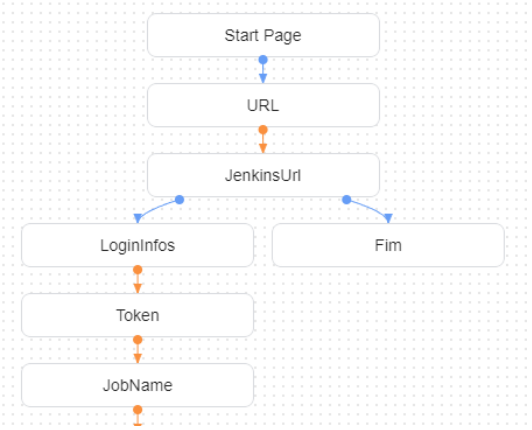
LoginInfos: Recupera o login do usuário (necessário para a api do jenkins)

Token: Solicita a API token do jenkins do usuário (necessário para fazer as requisições para api)

JobName: Recupera e armazena o nome do job da pipeline que o usuário queira analisar.

Fim: Encerra a conversação caso o usuário não queira interagir com o bot.

Figura 16: Fluxo de Intents configuradas no dialogflow



Fonte: O autor (2023)

Após a etapa anterior prosseguimos para as opções (as figuras foram divididas para uma melhor visualização);

AnaliseSeguranca, AnaliseAnomalias - Pages principais do projeto, responsáveis por gerar uma solicitação para o webhook configurado, e retornar os dados e as análises gerados pelo mesmo;

Build - Realiza o build remoto da pipeline;

SonarQube - Page que permite ao usuário integrar com o sonarqube, caso o usuário pretenda realizar essa integração ele pode escolher a opção disponibilizada e ser redirecionado para a próxima etapa;

PossuiSonar - Essa page identifica que o usuário deseja se integrar com o sonar e solicita algumas informações necessárias para realizar a conexão;

SonarToken - Page que solicita o API token do sonar para poder se conectar;

SonarProject - Pergunta ao usuário de qual projeto ele deseja que seja recuperado as informações;

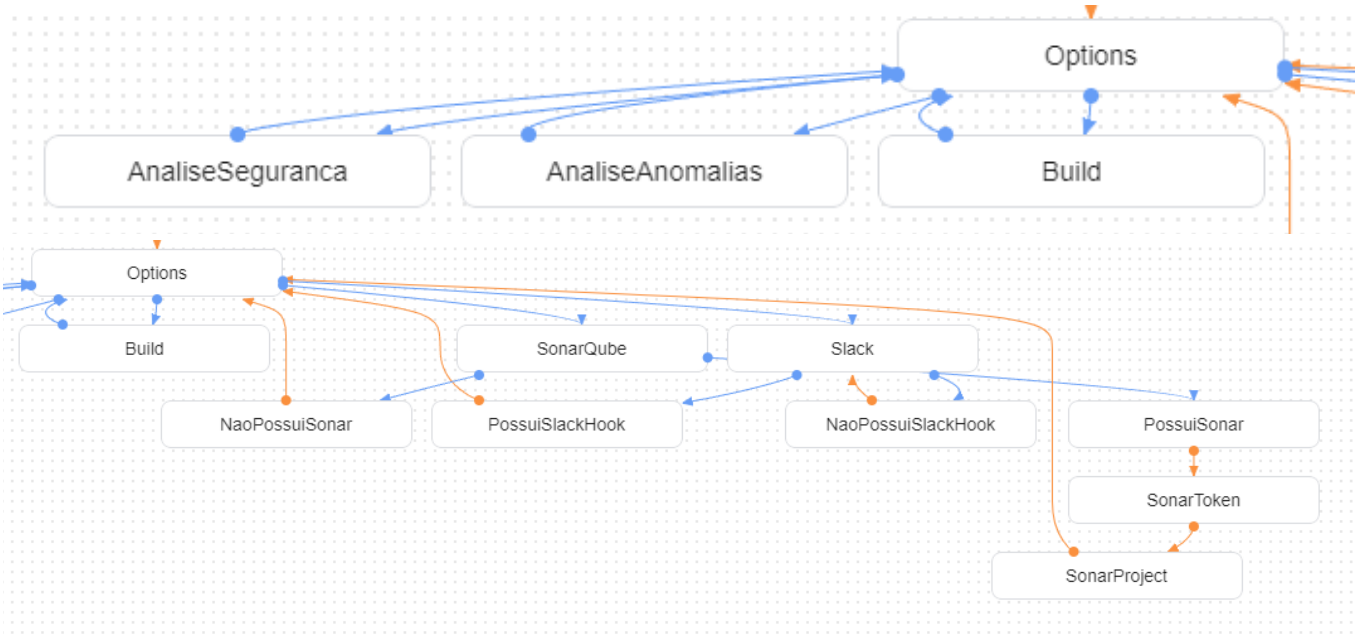
NaoPossuiSonar - Esta intent é ativada caso o usuário indique que não possua um sonar configurado, aqui ele recebe algumas dicas de como o fazer;

Slack - Opção que permite ao usuário integrar com o Slack, assim que o usuário escolhe se deseja integrar ou não, ele é redirecionado para alguma das outras duas intents;

PossuiSlackHook - Se o usuário confirmar que possui o slack e desejar receber as mensagens de análise do webhook em um dos seus canais do slack, essa opção é chamada;

NaoPossuiSlackHook - Essa intent é ativado caso o usuário indique que não possui um slack configurado, aqui o usuário também recebe dicas de como o fazer caso queira;

Figura 17: Fluxo de Intents configuradas no dialogflow (Opções)



Fonte: O autor (2023)

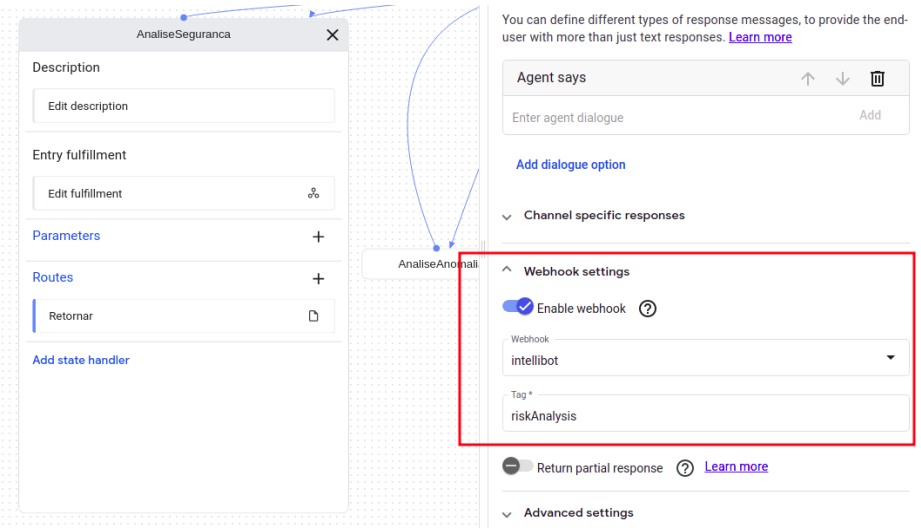
4.4 Webhook

Como citado na seção TAL, foi necessário configurar um webhook no Dialogflow para permitir a integração do chatbot e o Jenkins, além das outras ferramentas, sonarqube e slack. O webhook atua como uma ponte de comunicação, permitindo que o chatbot acesse informações em tempo real dos pipelines de CI/CD e forneça respostas aos usuários. O webhook foi programado para interpretar as intenções dos usuários, acionar consultas ao Jenkins e processar os dados recebidos para responder às solicitações dos usuários de forma adequada. Um

website simples para o webhook também foi desenvolvido para que o usuário consiga interagir com o agent do dialogflow a partir do dialogflow messenger.

Para que o webhook fosse capaz de identificar as solicitações do usuário, foi configurado no campo “entry fulfillment” as tags que a api do dialogflow enviará para o webhook. A figura abaixo, demonstra como foi implementado a identificação das intents configuradas no dialogflow, através do atributo “tag” definido na parte do “entry fulfillment”;

Figura 18: Ativando webhook para uma page



Fonte: O autor (2023)

Abaixo segue a ilustração do código implementado no webhook que identifica as intents;

Figura 19: Código demonstrando a implementação de identificação das intents

```
app.post("/webhook", async (req, res) => {
  const intentName = req.body.intentInfo?.displayName

  if(intentName === "Default Welcome Intent" ) {···
  }

  if(req.body.fulfillmentInfo?.tag === "jenkinsUrl") {···
  }

  if(req.body.fulfillmentInfo?.tag === "login") {···
  }

  if(req.body.fulfillmentInfo?.tag === "apiToken") {···
  }

  if(req.body.fulfillmentInfo?.tag === "jobName") {···
  }

  if(req.body.fulfillmentInfo?.tag === "sonarURL") {···
  }

  if(req.body.fulfillmentInfo?.tag === "sonarToken") {···
  }

  if(req.body.fulfillmentInfo?.tag === "sonarProject") {···
  }

  if(req.body.fulfillmentInfo?.tag === "slackUrl") {···
  }

  if(req.body.fulfillmentInfo?.tag === "anomalyAnalysis") {···
  }

  if(req.body.fulfillmentInfo?.tag === "riskAnalysis") {···
  }

  if(req.body.fulfillmentInfo?.tag === "buildPipeline") {···
  }
```

Fonte: O autor (2023)

4.4.1 Estrutura

Como dito na seção 3, foi utilizado como base a estrutura da “Clean Architecture” para a implementação do webhook, essa estrutura foi utilizada com o objetivo de facilitar a modularização e a manutenção do código. A estrutura está organizada da seguinte maneira, com adição da camada de pages:

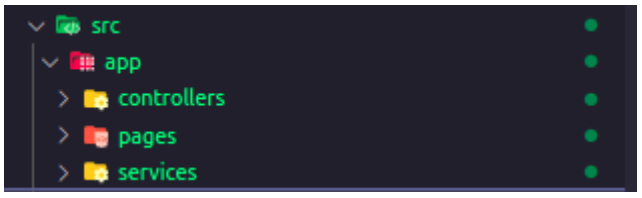
Controllers: Esta camada é responsável por receber as solicitações do Dialogflow, interpretar as intenções dos usuários e acionar as ações apropriadas.

Services: Os serviços são os componentes mais cruciais do webhook. Os dois módulos principais, "Análise de Anomalias" e "Análise de Riscos", são implementados nesta camada. Eles são responsáveis por consultar e processar informações em tempo real do Jenkins, realizar análises de segurança e retornar respostas ao Dialogflow.

Webhook: Camada responsável por se comunicar com o Dialogflow e retornar às solicitações;

Pages: A camada de páginas é responsável por abrigar o conteúdo front-end da aplicação.

Figura 20: Organização de pastas do projeto

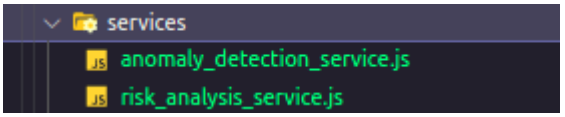


Fonte: O autor (2023)

4.4.2 Módulos

Os módulos mais importantes da implementação do webhook são:

Figura 21: Módulos de análise



Fonte: O autor (2023)

Módulo de Análise de Anomalias: Este módulo é dedicado à detecção de anomalias nos pipelines de CI/CD. Assim que o usuário solicita essa análise, o módulo dá início a uma cadeia de eventos recuperando algumas informações cruciais do Jenkins, como falhas de build, implantações mal sucedidas e outliers na pipeline e notifica a equipe / usuário por meio do chatbot.

Módulo de Análise de Riscos: Este módulo realiza a análise de riscos nos pipelines de CI/CD. Ele avalia ameaças e vulnerabilidades potenciais, prioriza riscos e fornece recomendações de segurança. Se integrado com o slack e com sonar, ele pode fornecer informações detalhadas sobre os riscos identificados.

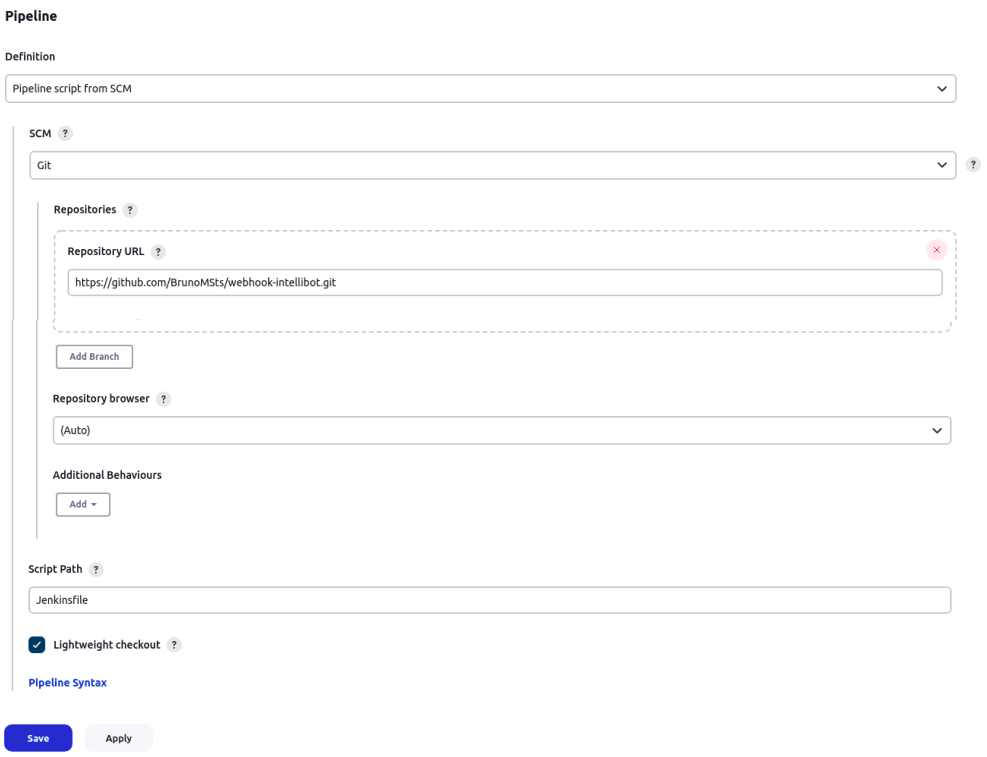
4.5 Jenkins

Nesta seção, será abordada a implementação no Jenkins, componente fundamental para a execução e automação de pipelines de CI/CD. Será descrito como os projetos de teste foram configurados e quais plugins utilizados, como SonarQube e OWASP Dependency-Check, para auxiliar o leitor ou possíveis desenvolvedores na implementação de práticas de segurança.

4.5.1 Configuração de Projetos de Teste

O jenkins permite possui diversos meios para a criação da pipe, neste trabalho adotamos o modelo utilizando o “Jenkinsfile” nas pastas dos projetos escolhidos;

Figura 22: Configuração da Pipeline usando o repositório Github



Fonte: O autor (2023)

Para a implementação, foram criados dois projetos de teste no Jenkins, representando aplicativos ou serviços que fazem parte dos pipelines de CI/CD. Esses projetos foram configurados com as seguintes etapas:

- Compilação: Implementamos a compilação do código-fonte para gerar artefatos executáveis ou pacotes a serem implantados posteriormente.
- Análise Estática de Código com SonarQube: Integrando o SonarQube, configuramos análises estáticas de código para identificar possíveis problemas de qualidade e segurança. Isso inclui a detecção de vulnerabilidades de segurança, duplicação de código e padrões de codificação inadequados.
- Análise de Dependências com OWASP Dependency-Check: Utilizamos o plugin OWASP Dependency-Check para verificar as dependências do projeto em busca de vulnerabilidades conhecidas. Isso ajuda a identificar e abordar potenciais ameaças de segurança relacionadas às bibliotecas utilizadas.

Figura 23: Etapas configurada para as Pipelines

Fonte: O autor (2023)

4.6 SonarQube

Nesta seção, discutiremos a importância do SonarQube como uma etapa opcional, mas crucial, para melhorar a qualidade e a segurança do código. Também abordaremos a criação de um token necessário para usar a API do SonarQube, que é uma prática recomendada para garantir a segurança e a autenticação adequada durante a integração.

4.6.1 Importância do SonarQube

O SonarQube é uma plataforma de análise de código-fonte que desempenha um papel crítico na avaliação da qualidade, segurança e manutenibilidade do código. É uma ferramenta valiosa para equipes de desenvolvimento, permitindo a identificação precoce de problemas de código e vulnerabilidades de segurança.
As principais funcionalidades do SonarQube incluem:

- Análise Estática de Código: O SonarQube verifica o código-fonte em busca de padrões e práticas inadequadas, identificando possíveis problemas de qualidade e segurança.
- Detecção de Vulnerabilidades: A ferramenta pode identificar vulnerabilidades de segurança conhecidas em bibliotecas e dependências.
- Integração Contínua: O SonarQube pode ser integrado diretamente aos pipelines de CI/CD, permitindo análises automáticas do código em cada build e implantação.

4.6.2 Configurando o SonarQube

Para utilizar a API do SonarQube, é recomendável criar um token de autenticação. Isso é importante para garantir a segurança e o controle de acesso aos recursos do SonarQube. Aqui estão os passos para criar um token de API no SonarQube:

- Fazer Login no SonarQube: Acesse a interface do SonarQube e fazer login na conta.
- Acessar a Configuração de Segurança: No menu principal, navegar até a seção de configuração de segurança ou credenciais. A localização exata pode variar dependendo da versão do SonarQube.
- Criar um Token de Acesso: Na seção de tokens ou credenciais, encontrar a opção para criar um novo token. Escolher um nome descritivo para o token que indique seu propósito.
- Salvar o Token: Após a criação do token, ele será gerado e exibido na tela. É crucial copiar e armazenar o token com segurança, pois ele geralmente é exibido apenas uma vez. O token é usado como um meio de autenticação para acessar a API do SonarQube.
- Utilizar o token no chatbot: Ao fazer chamadas à API do SonarQube, o token de autenticação precisa ser enviado no cabeçalho da solicitação, geralmente usando o cabeçalho "Authorization". Por causa disto, o chatbot solicita o token ao usuário
- O uso de um token de API garante que apenas usuários autorizados possam acessar e interagir com o SonarQube por meio da API. Isso é fundamental para garantir a segurança e o controle adequados sobre as operações de análise e avaliação do código.

4.7 Slack

Nesta seção, abordaremos a integração também opcional com o Slack. Falaremos sobre a criação de um webhook no Slack, uma etapa importante para permitir que o sistema envie notificações e mensagens para canais do Slack,

fornecendo informações relevantes sobre eventos e atividades importantes. E onde as mensagens poderão ficar salvas para uma posterior análise.

4.7.1 Importância da Integração com o Slack

A integração com o Slack pode ser uma adição valiosa para o sistema de CI/CD e segurança. Ela oferece diversos benefícios:

- Notificações: A integração com o Slack permite que o sistema envie notificações para canais específicos ou para equipes inteiras, mantendo todos informados sobre eventos críticos ou atualizações de segurança.
- Colaboração eficiente: Os canais do Slack facilitam a colaboração entre equipes de desenvolvimento, operações e segurança, tornando mais fácil o compartilhamento de informações e a discussão de questões de segurança.

4.7.2 Criação de um Webhook no Slack

Para permitir a comunicação entre o sistema e o Slack, é necessário criar um webhook. As etapas para criar um webhook no Slack são:

- Acessar: https://api.slack.com/apps/ e fazer login com a conta do slack;
- Após logar: Escolher o projeto criado e procurar na aba da esquerda por “Incoming Webhooks”
- Na seção “Incoming Webhooks”: Clicar em “Add new webhook to Workspace”, escolher o canal ou conversa para a qual as mensagens serão enviadas. Uma url para o hook será gerada.
- Depois de criar o hook: No chatbot ao escolher a opção de se conectar com o slack, fornecer a url criada no passo anterior. Desse modo o chatbot vai redirecionar as mensagens para o canal do Slack sempre que o usuário solicitar um evento, como análise de segurança e anomalias. Isso facilitará a comunicação e a colaboração entre as equipes e permitirá ações rápidas em resposta a eventos críticos.

5 TESTES E RESULTADOS

Para avaliar os resultados do chatbot, alguns aspectos foram avaliados como:

Análise de Riscos: A identificação e avaliação de ameaças potenciais ao software são fundamentais para a criação de estratégias de segurança eficazes. Isso envolve a análise de possíveis vulnerabilidades, ataques e seus impactos.

Detecção de Anomalias em Tempo Real: A detecção de anomalias é essencial para identificar comportamentos incomuns que podem indicar uma violação de segurança. Monitorar métricas de desempenho, logs de sistema e tráfego de rede é crucial para identificar atividades suspeitas.

Chatbot Inteligente para Notificações: A integração de um chatbot inteligente permite a comunicação em tempo real com desenvolvedores e partes interessadas. O chatbot pode notificar sobre riscos de segurança, relatar anomalias, sugerir ações corretivas e até mesmo automatizar tarefas de segurança.

Integração com Ferramentas de Segurança: A integração com ferramentas de segurança, como scanners de vulnerabilidades, sistemas de detecção de intrusões e gerenciamento de identidade, é fundamental para fortalecer a segurança no Pipeline de CI/CD.

5.1 Fluxo Principal

O fluxo principal é dividido na seguinte ordem:

O usuário interage com o chatbot: Etapa inicial onde o usuário tem suas primeiras interações com o chat;

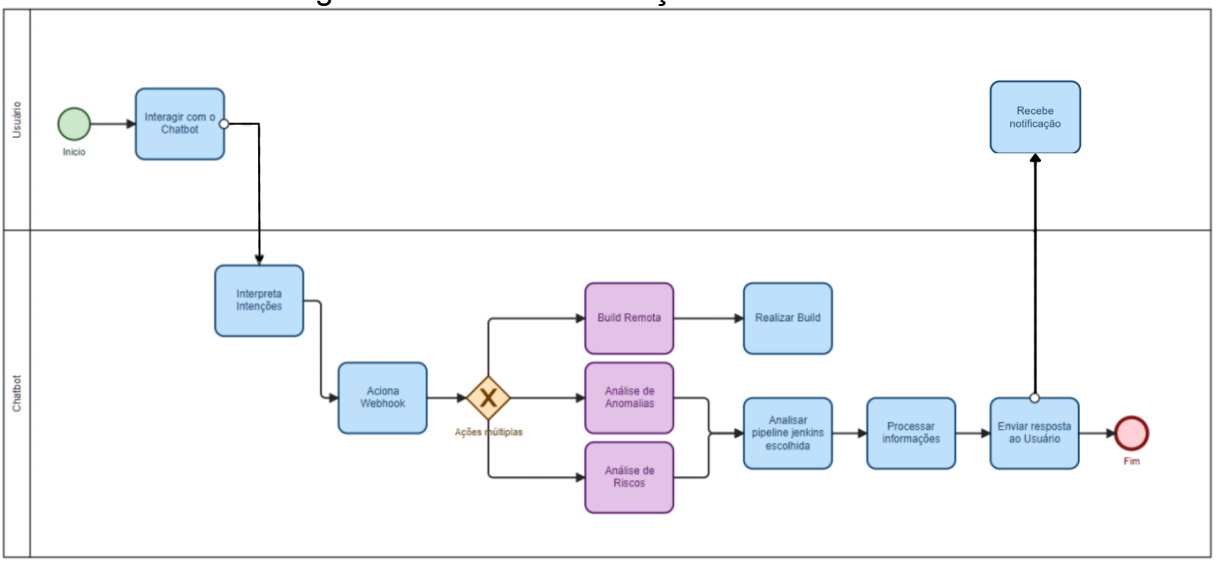
Chatbot interpreta intenções: Etapa onde o chatbot identifica o que o usuário deseja;

Webhook é acionado: Após a identificação das intenções do usuário o webhook é acionado ; E então é realizado algum dos três fluxos escolhidos pelo usuário. É válido salientar que, além dos três (Análise de risco, Análise de anomalia e Build) contidos na figura 16, existe o fluxo do usuário se conectar com o Slack e com o Sonar, o qual não foi colocado nesta figura por motivos didáticos; Porém será apresentado na próxima seção;

Webhook obtém informações do Jenkins: Assim que o usuário selecionar alguma das opções, em seguida o webhook começará a coletar os dados necessários, seja anomalias na pipeline, falta de atualizações nos plugins, possíveis outliers no processamento da pipe ou realizar uma build;

Retorno para o Usuário: Após a obtenção das informações, a resposta é formatada para o formato que o dialgoflow entenda e consiga exibir de maneira limpa para o usuário; Após todo o processamento realizado de formatação, a resposta é enviado para o agent do dialogflow exibindo assim os resultados para o usuário;

Figura 24: Fluxo de interação com o ChatBot



Fonte: O autor (2023)

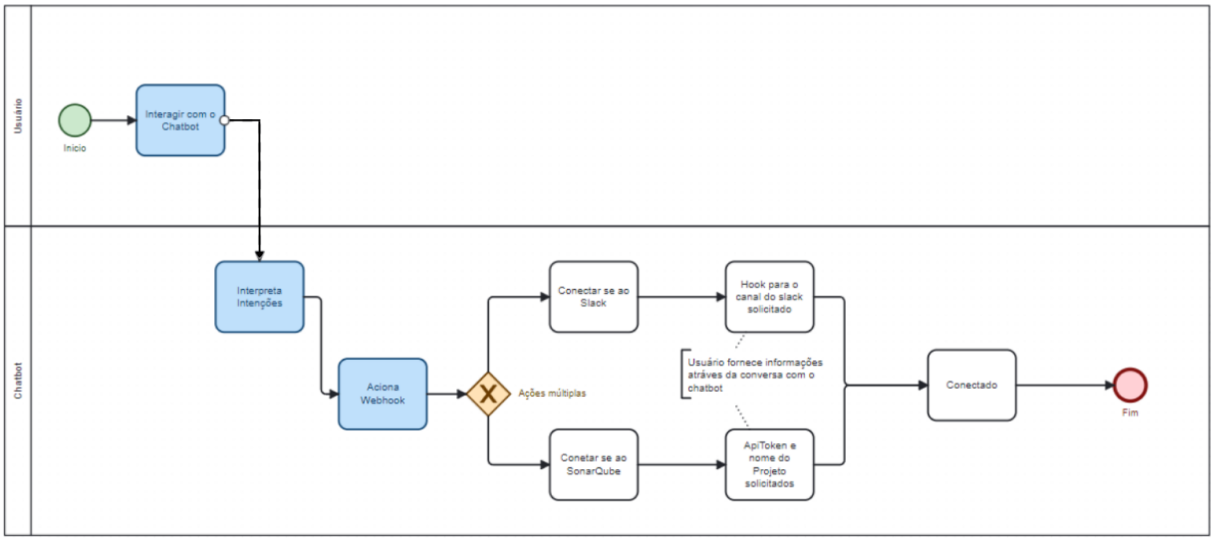
5.2 Fluxo secundário

O fluxo secundário, é para caso o usuário queira mais informações. O fluxo segue da seguinte forma;

Conectar-se ao Slack - O usuário informa ao chatbot que deseja se conectar com o slack; Então o chatbot solicita o webhook para qual o canal que o vai ser conectado para receber as informações. Ao fazer essa conexão, ele não receberá mais as mensagens pelo dialogflow messenger, as mensagens serão redirecionadas para o slack com mais detalhes;

Conectar-se ao Sonar - O usuário informa que deseja conectar ao sonar; Então o webhook solicita o token para a conexão com a api do sonar e qual o nome do projeto que o usuário deseja analisar, desse modo o webhook é capaz de recuperar informações mais detalhadas sobre o estado do codigo / projeto;

Figura 25: Fluxo Secundário, conexão com Slack e Sonar



Fonte: O autor (2023)

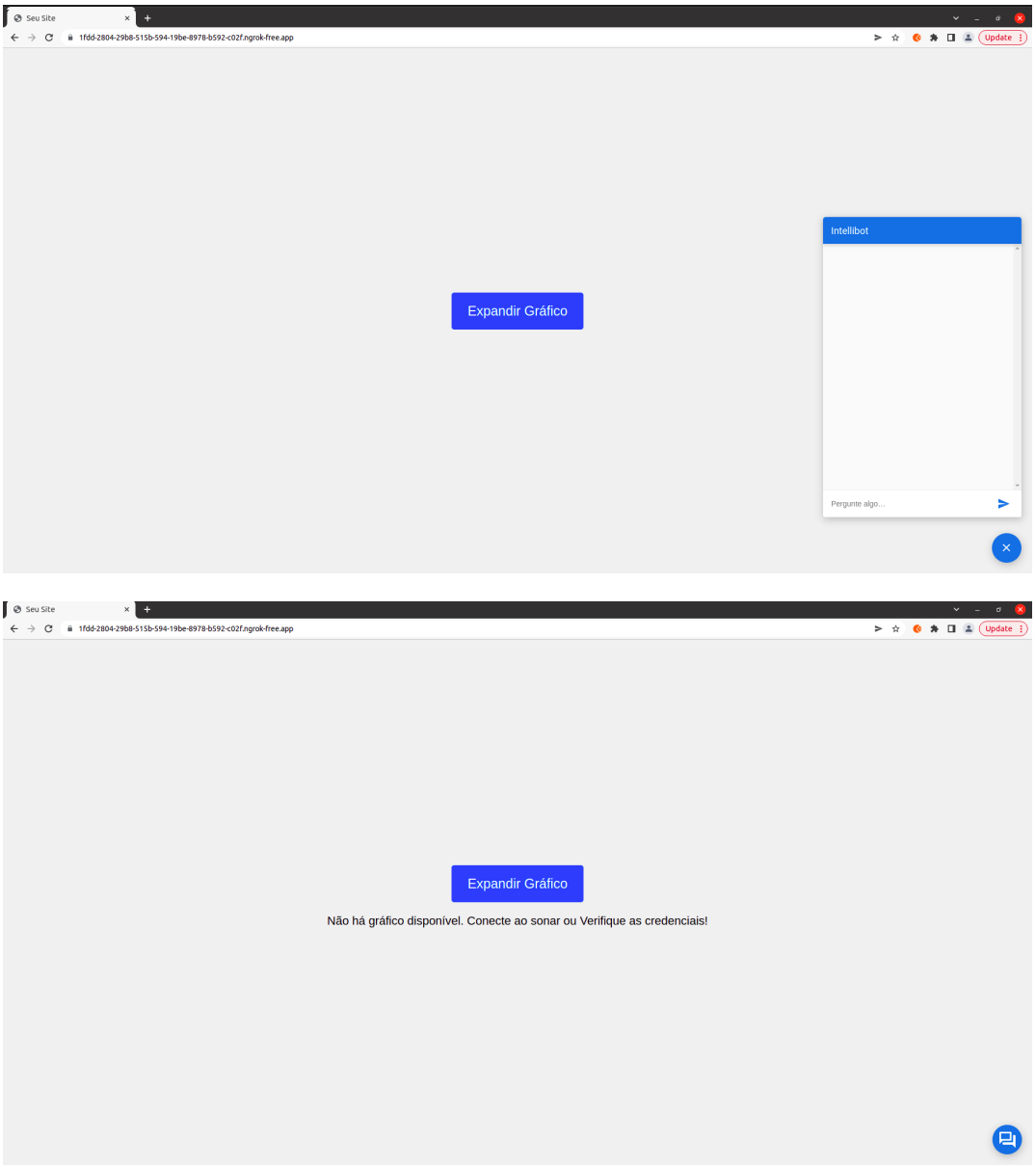
5.3 Aplicação

Demonstração prática do fluxo principal da aplicação web utilizando o Google Chrome como navegador.

5.3.1 Tela inicial

As figuras a seguir demonstram a tela inicial da aplicação, com o botão para “Expandir Gráfico” que é responsável por exibir o gráfico gerado na aplicação, caso não haja um gráfico uma mensagem é exibida para o usuário como demonstrado na figura, junto a isso é possível perceber o botão de chat no canto inferior direito, o qual tem a função para expandir o chat de conversação.

Figura 26: Tela inicial



Fonte: O autor (2023)

5.3.2 Interação com o ChatBot

Esta seção será inteiramente dedicada a mostrar a interação com o chatbot através dos prints retirados da conversação, será abordado diversos fluxos como solicitar análises, realizar builds remotas, integrar com o sonar e slack. Aqui será mostrado o fluxo completo da aplicação; Passando pelos seguintes pontos:

- Interação inicial com o chatbot
- Interação com o board de opções do chatbot
- Realização de Build Remota
- Solicitação de análise de anomalia
- Solicitação de análise de segurança
- Integração com Slack
- Mensagens chegando no Slack
- Integração com o Sonar e recuperação dos dados

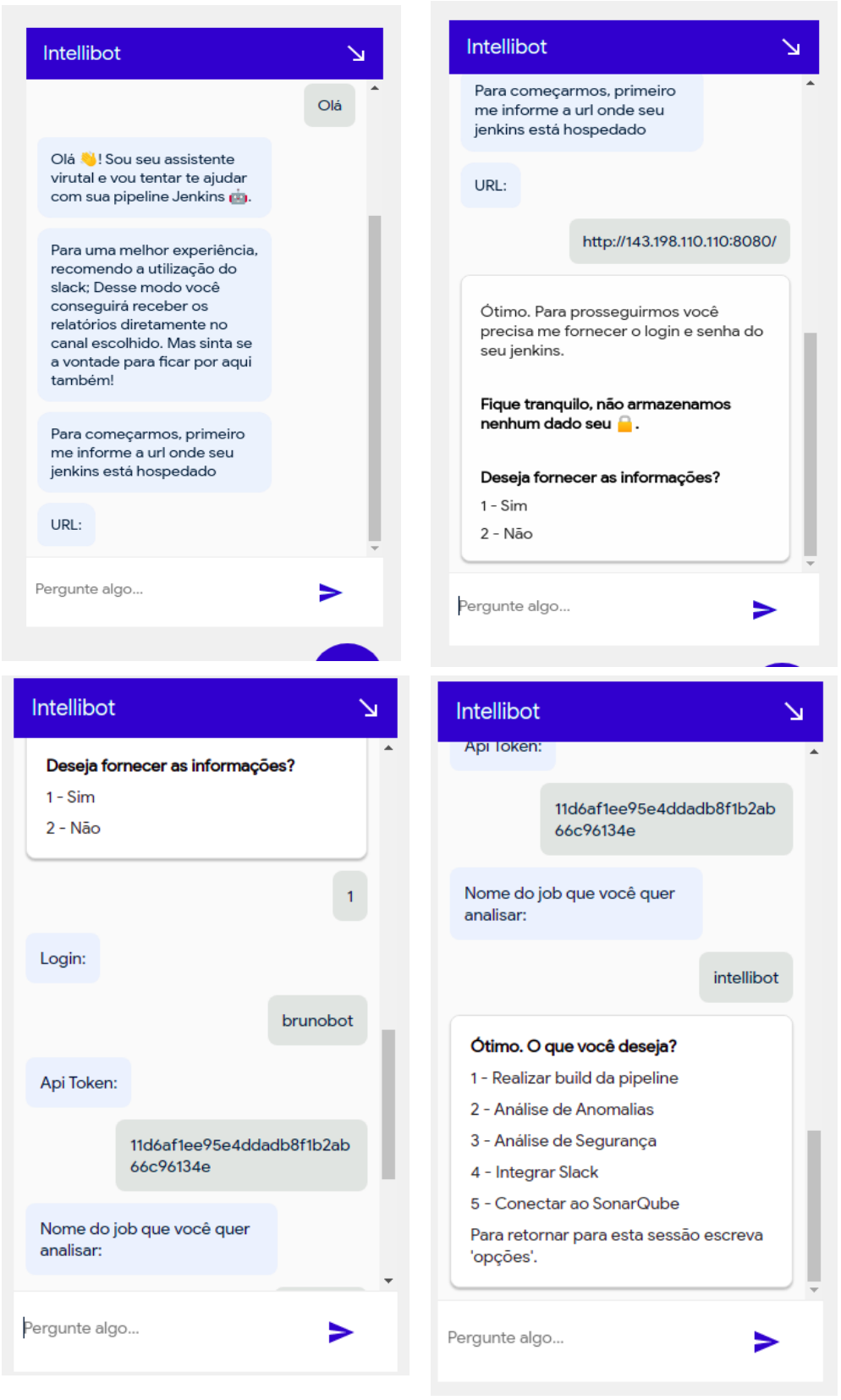
5.3.3 - Teste 1 (Intellibot)

O primeiro teste será realizado utilizando o próprio projeto do chatbot, apelidado de “Intellibot”. A pipeline do jenkins foi configurada como demonstrado na seção 4.5.

5.3.3.1 Interação Inicial

As imagens a seguir demonstram a primeira interação com o chatbot, onde o usuário tem a iniciativa. Assim que o usuário interage, o chatbot retorna desejando uma espécie de “boas-vindas” e solicita algumas informações para o usuário, como a URL do seu Jenkins, login, ApiToken e o nome do job / pipeline, informações necessárias para que o webhook configurado consiga recuperar os dados do Jenkins. Assim que estiver tudo certo o chatbot retorna para o usuário um quadro de opções com diversas operações que o mesmo pode estar realizando.

Figura 27: Interações iniciais com o chatbot

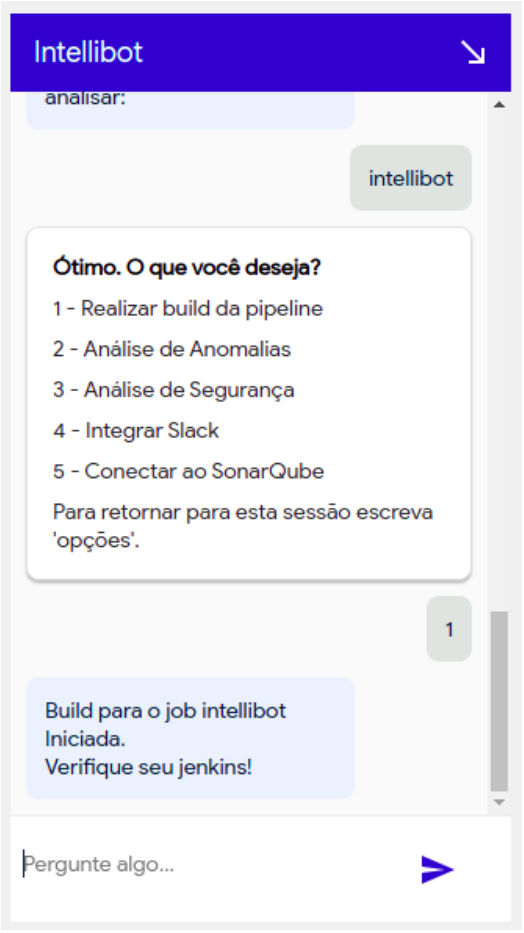


Fonte: O autor (2023)

5.3.3.2 Build Remota

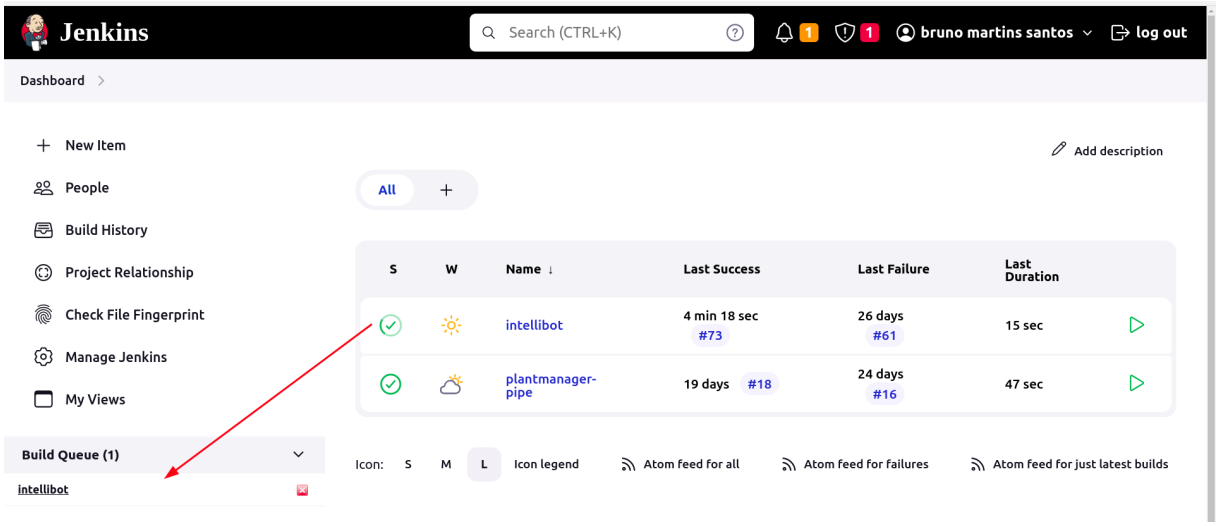
Nessa etapa o usuário faz a solicitação para o chatbot realizar uma build remota da pipeline e logo em seguida é possível observar que o jenkins coloca na sua fila de execução e assim que estiver tudo pronto o build da pipeline é realizado, juntamente a isso, caso o slack já esteja configurado uma notificação é enviada para o canal escolhido como demonstrado na imagem 30.

Figura 28: Solicitação de build remota



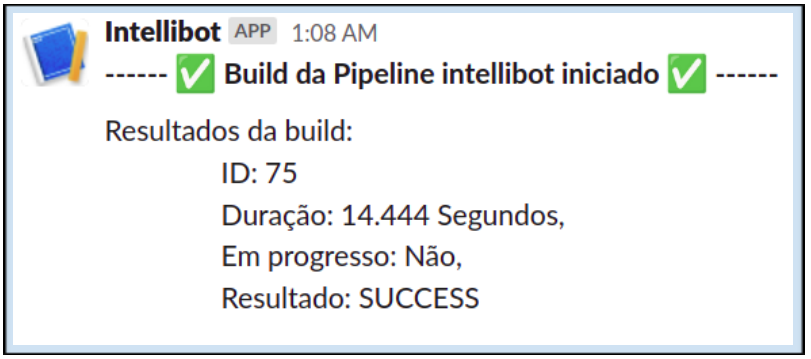
Fonte: O autor (2023)

Figura 29: Jenkins colocando o build solicitado na fila



Fonte: O autor (2023)

Figura 30: Notificação da build remota

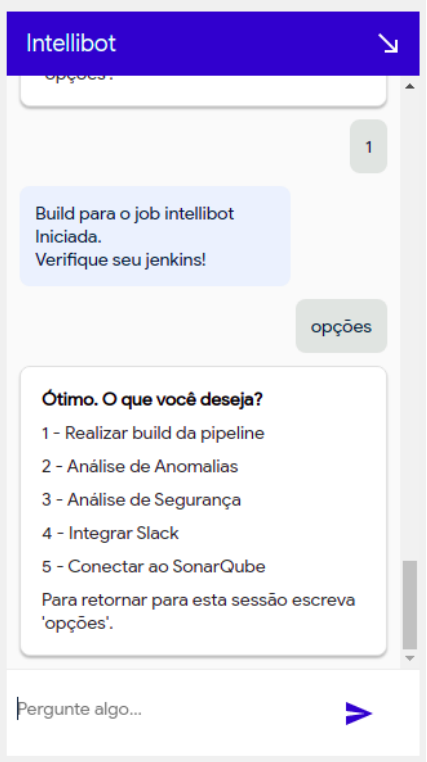


Fonte: O autor (2023)

5.3.3.3 Retornando para as Opções

Assim que o usuário executa uma ação ele é capaz de retornar para o quadro de opções, simplesmente escrevendo “opções”, sendo assim capaz de realizar uma outra ação novamente, como demonstrado na ilustração abaixo.

Figura 31: Retornando para as opções



Fonte: O autor (2023)

5.3.3.4 Análise de anomalias

Assim que o usuário retorna para as opções e solicita a análise de anomalias o chatbot envia a informação para o webhook o qual possui a lógica de realizar esta etapa. Assim que a análise é realizada os dados são enviados novamente para o chatbot, essa etapa de análise dura alguns segundos.
As informações obtidas e suas motivações são:

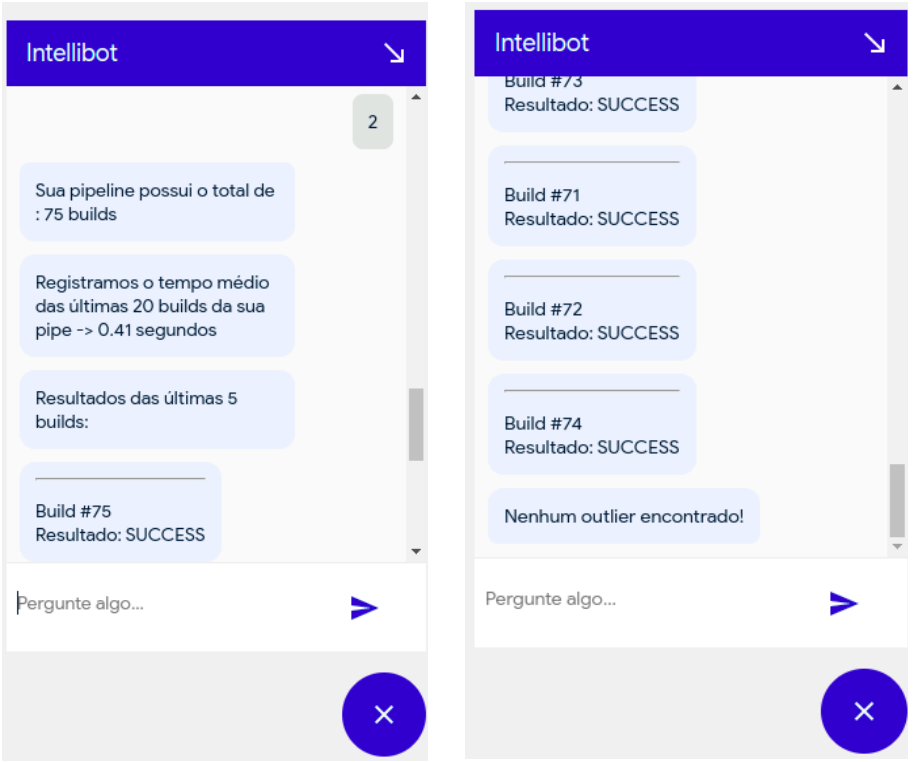
Quantidade de Builds: Monitorar a quantidade de builds ajuda a identificar mudanças significativas na atividade de desenvolvimento, que podem indicar anomalias.

Tempo Médio das Builds: Acompanhar o tempo médio das builds ajuda a detectar variações que podem ser causadas por problemas de desempenho ou eficiência.

Resultados das Últimas Builds: Avaliar os resultados das builds, como sucesso ou falha, ajuda a identificar sequências incomuns de eventos que podem indicar anomalias.

Identificação de Outliers: Encontrar valores atípicos nos dados, como builds que levam muito mais tempo do que o normal, pode ser um indicativo valioso de anomalias.

Figura 32: Solicitando análise de anomalias

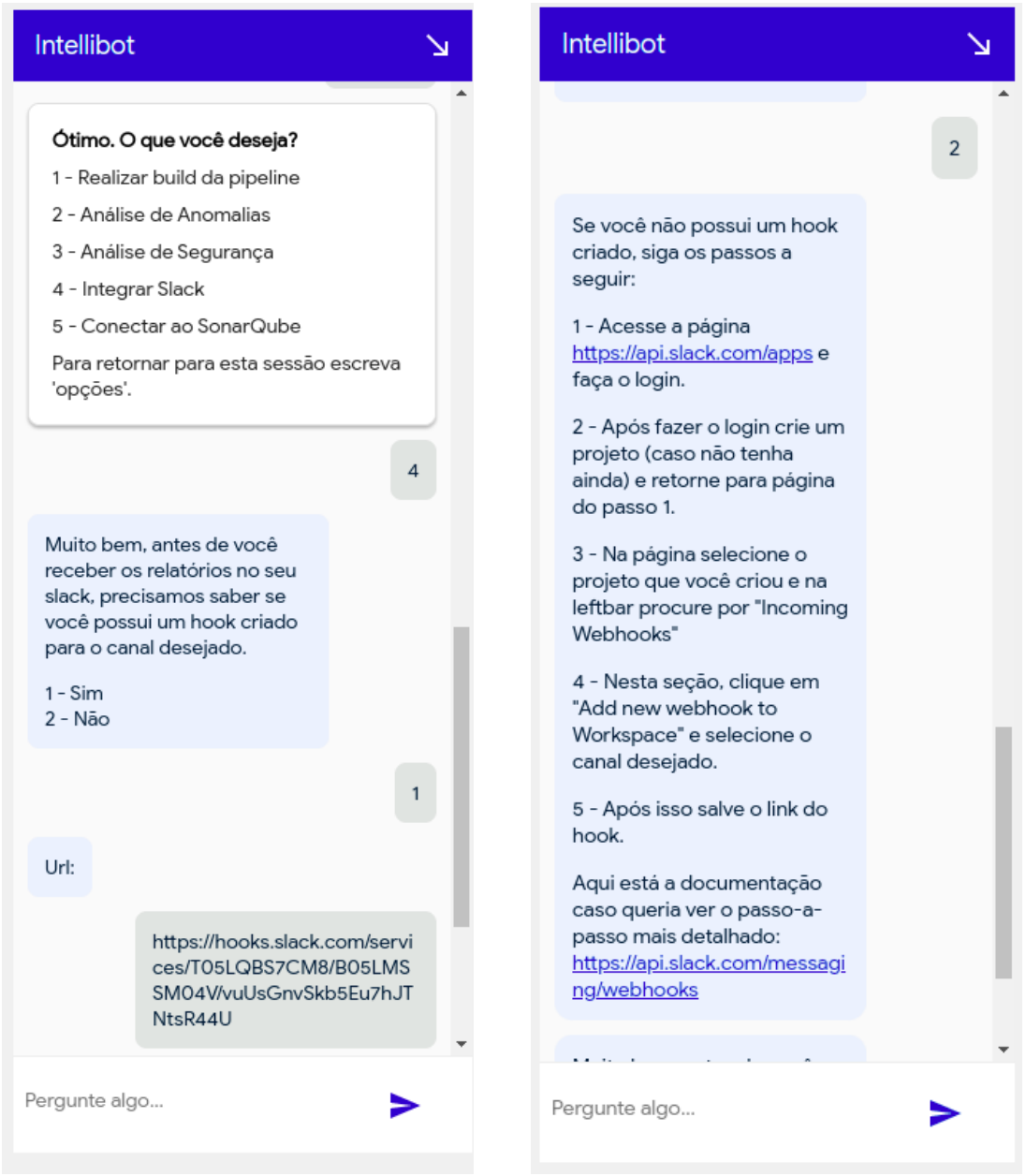


Fonte: O autor (2023)

5.3.3.5 Integrando com o Slack

Esta etapa demonstra a integração que o usuário pode realizar com o slack, caso o mesmo deseje manter as informações retornadas pelo chatbot salvas em algum lugar. As imagens a seguir demonstram as opções 1 (possuir hook para o canal do slack) e 2 (não possuir esse hook), caso o usuário se encontre na etapa 2, o chatbot informa todo um passo a passo de como ele pode fazer a criação do hook.

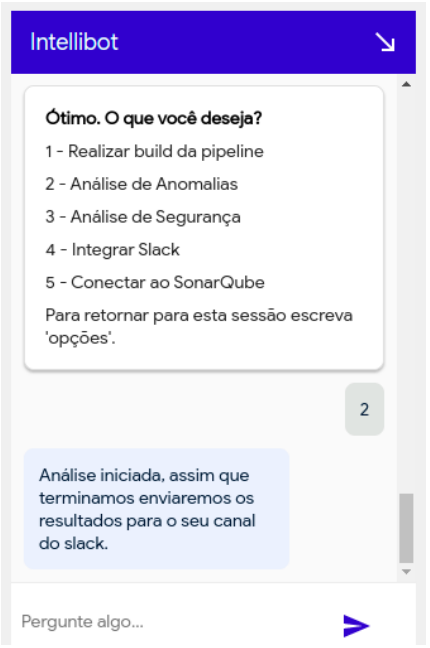
Figura 33: Integração com Slack (opções 1 e 2)



Fonte: O autor (2023)

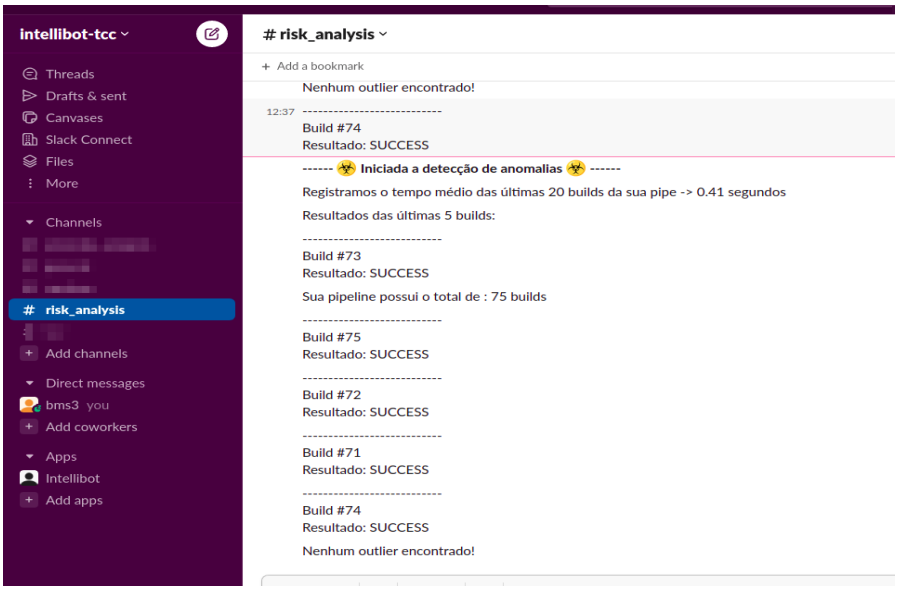
Assim que o usuário faz a integração com o slack, e solicita novamente a análise de anomalia, o chatbot apenas retorna uma mensagem de que as análises foram enviadas para o slack, e as informações vão chegando no slack de modo formatado e legível para todos que estão no canal. Como demonstrado respectivamente nas figuras 34 e 35.

Figura 34: Mensagem ao solicitar análise quando o Slack estiver conectado



Fonte: O autor (2023)

Figura 35: Notificações recebidas no slack (quando integrado)



Fonte: O autor (2023)

5.3.3.6 Análise de Riscos / Segurança

Indo para o segundo, mas não menos importante, módulo de análise de riscos ou segurança. Assim que solicitado a análise de segurança o webhook processa toda a lógica e retorna para o usuário os seguintes dados:

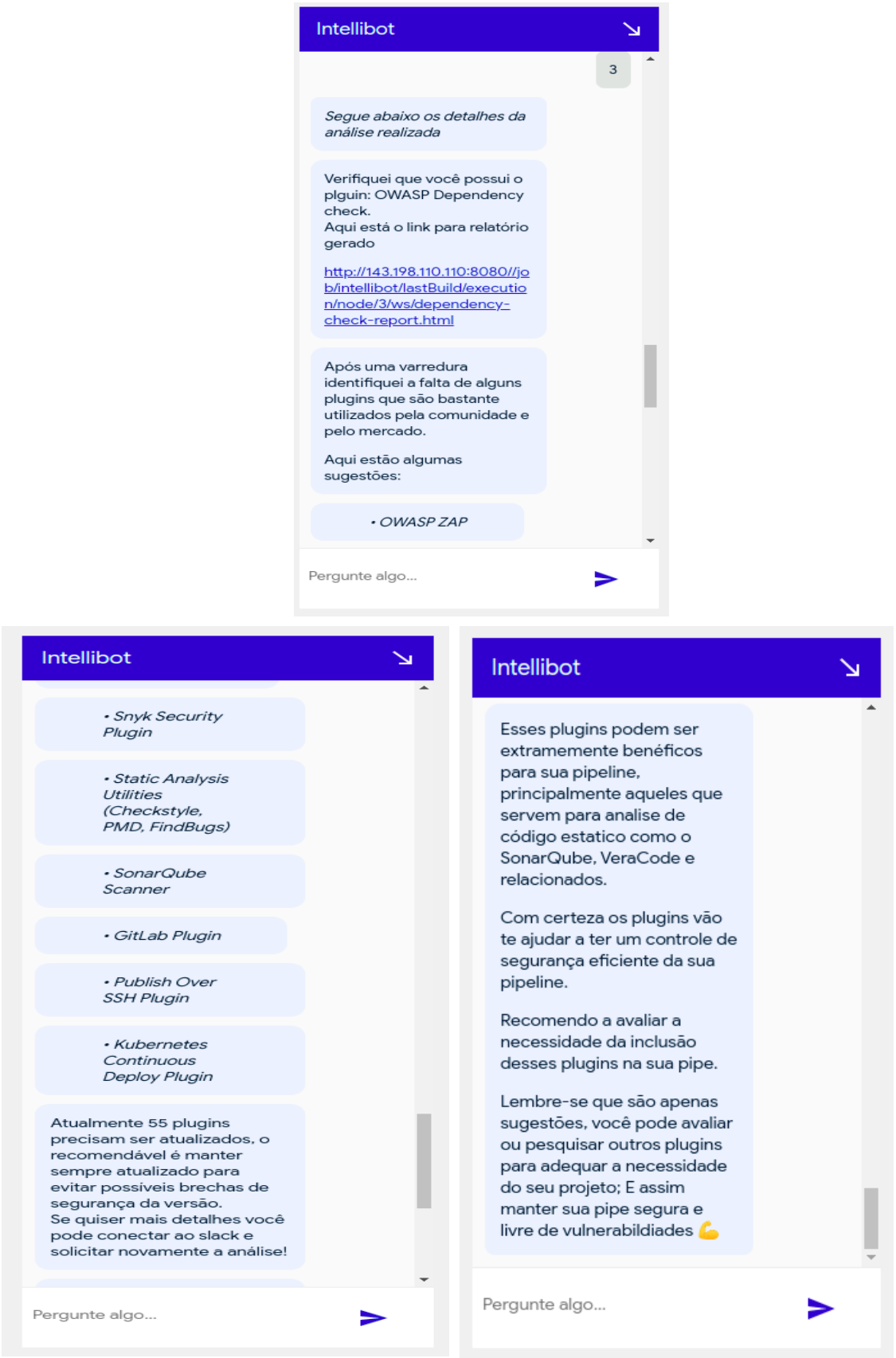
Relatório do OWASP Dependency Check: Este relatório revela vulnerabilidades em bibliotecas e dependências do projeto, permitindo a identificação e avaliação de ameaças à segurança do software.

Plugins Recomendados: A recomendação de plugins pode fortalecer a segurança do ambiente de desenvolvimento, incluindo ferramentas para análise estática de código e verificação de vulnerabilidades de terceiros.

Plugins que Precisam ser Atualizados: A identificação de plugins que necessitam de atualização é crucial para manter um ambiente de desenvolvimento seguro, garantindo a aplicação de correções de segurança e atualizações de recursos.

Sugestões sobre Benefícios dos Plugins: Compreender por que esses plugins são benéficos ajuda a equipe de desenvolvimento a entender sua importância, pois podem melhorar a qualidade do código, reduzir vulnerabilidades e acelerar a detecção de ameaças à segurança.

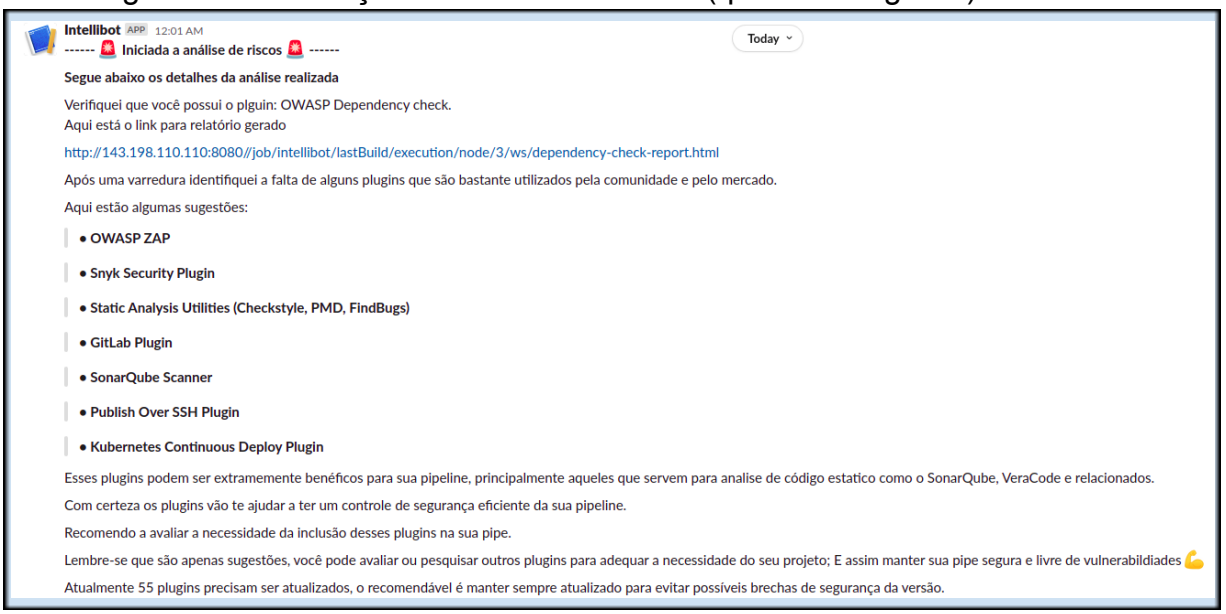
Figura 36: Solicitação e retorno da análise de riscos



Fonte: O autor (2023)

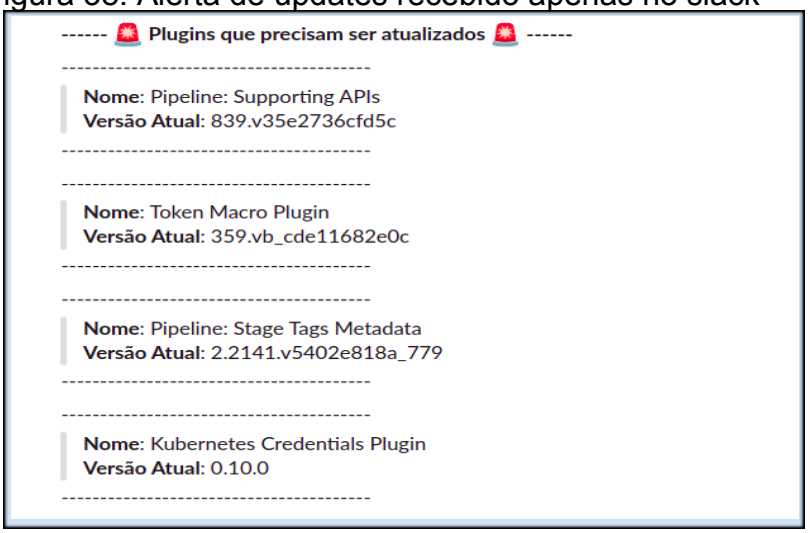
A figura abaixo demonstra como as mensagens estão sendo retornadas para o slack; As informações serão as mesmas retornadas para o chatbot caso o usuário não se conecte ao slack, porém, com a adição de alguns alertas que são notificados apenas pelo slack. Como visto na imagem 38, onde é demonstrado alguns plugins que precisam ser atualizados e qual a sua versão atual, e na imagem 39 onde foi identificado uma possível vulnerabilidade em um dos plugins instalados no Jenkins e uma outra vulnerabilidade na própria versão do Jenkins.

Figura 37: Notificações recebidas no slack (quando integrado)



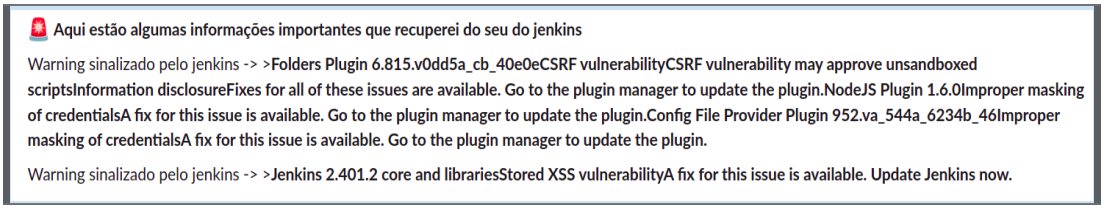
Fonte: O autor (2023)

Figura 38: Alerta de updates recebido apenas no slack



Fonte: O autor (2023)

Figura 39: Alerta de vulnerabilidade sinalizado



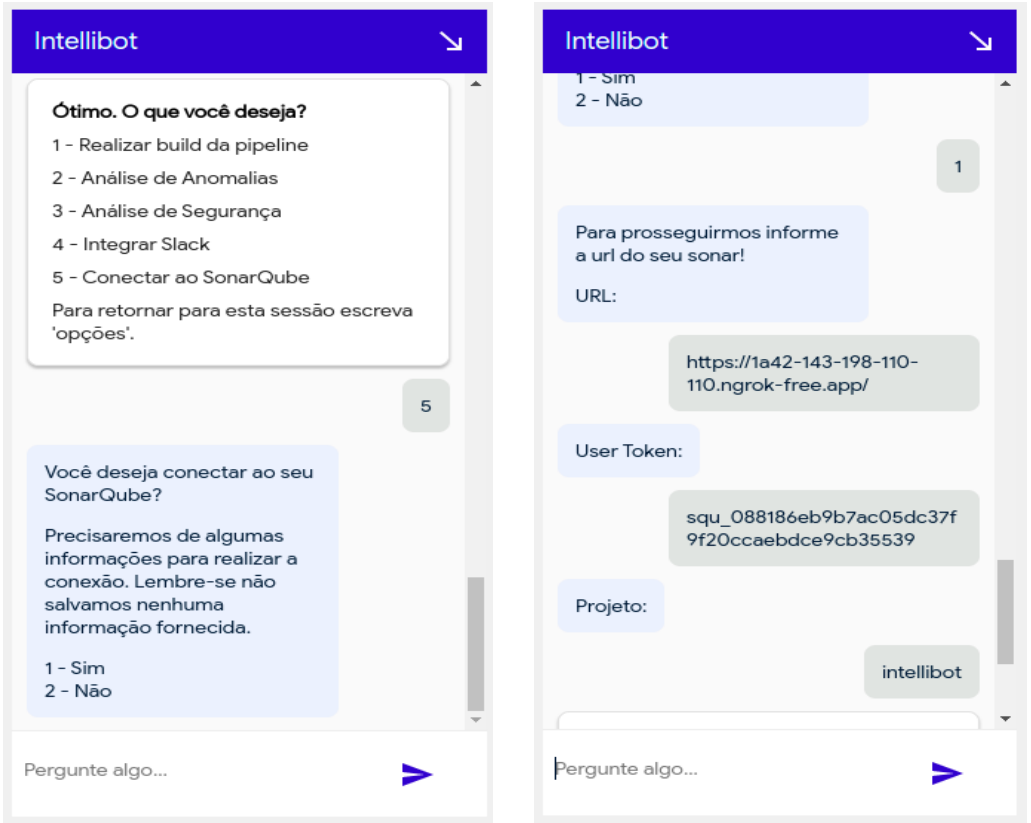
Fonte: O autor (2023)

5.3.3.7 Integrando com o SonarQube

Aqui nesta etapa é demonstrado a realização da conexão / integração com o SonarQube, plataforma responsável por realizar uma análise estática do código.

A integração é bem simples, o usuário apenas precisa fornecer algumas informações que o webhook do chatbot faz todo o trabalho, o interessante é que o usuário é capaz de escolher qualquer projeto conectado no seu Sonar, só basta colocar o nome referente ao projeto que ele queira analisar.

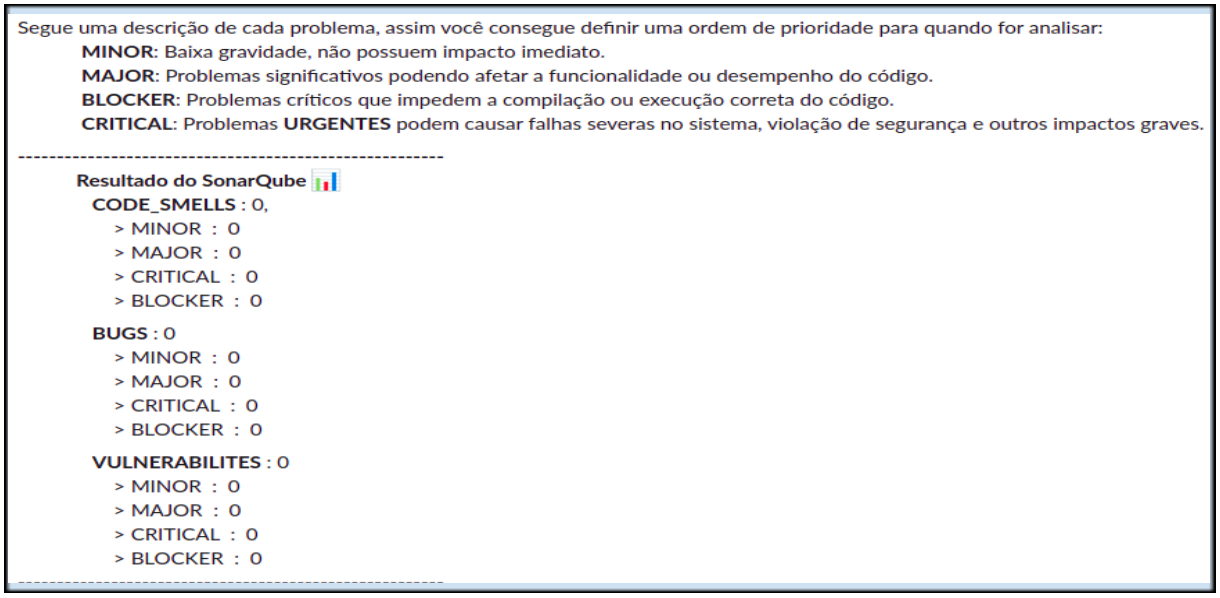
Figura 40: Integrando com o Sonar



Fonte: O autor (2023)

Na figura a seguir para fins didáticos já está sendo ilustrado diretamente o retorno para o slack de algumas informações importantes recuperadas do sonarqube, como bugs, codesmells e vulnerabilidades bem como o tipo / criticidade de cada um. É possível notar que os dados retornam vazios, isso se deve ao fato do projeto analisado não possuir nenhum tipo das três informações citadas anteriormente.

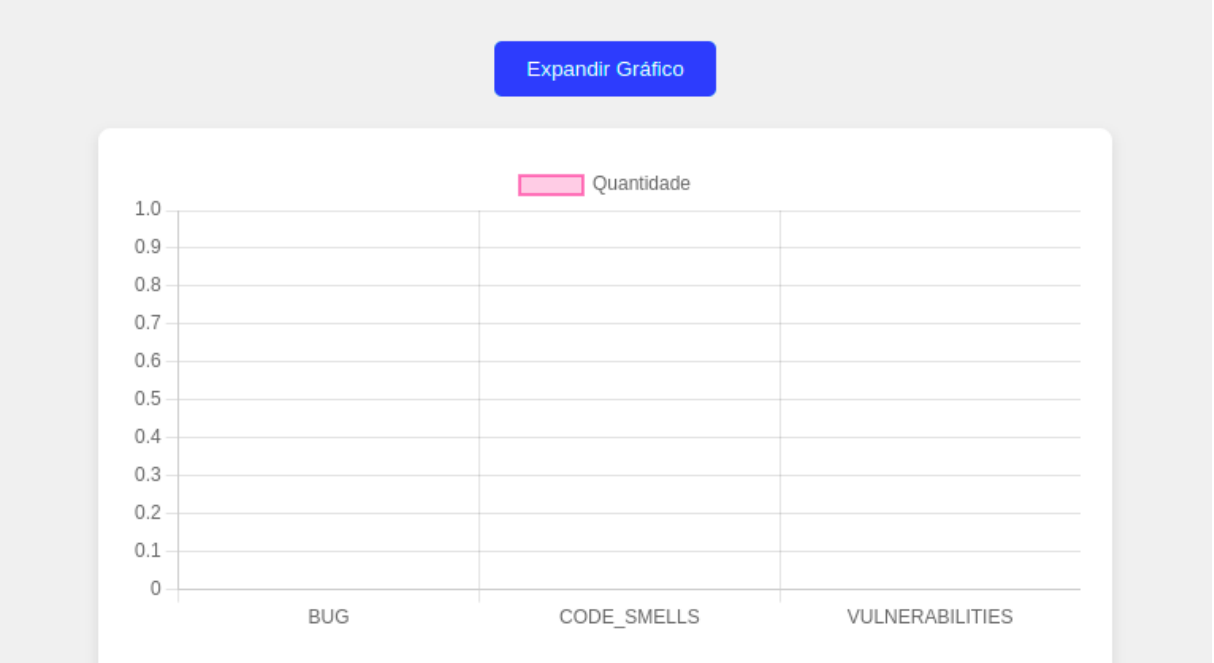
Figura 41: Informações só visíveis para quem se conectar com o sonar



Fonte: O autor (2023)

A ilustração a seguir demonstra o gráfico gerado na tela inicial do chatbot que no momento está vazio, pois o projeto não possui nenhum problema apontado, o usuário pode usar esse gráfico para diversos fatores, como levar como métrica para o seu time. No próximo projeto teste veremos um exemplo de gráfico plotado.

Figura 42: Gráfico gerado a partir das informações obtidas no sonar (vazio pois o sonar não detectou nenhum problema no código)



Fonte: O Autor (2023)

Os testes foram realizados de maneira abrangente neste primeiro projeto, revelando os seguintes resultados:

- Foi identificada uma vulnerabilidade na versão do Jenkins utilizada.
- No entanto, não foram detectadas anomalias aparentes em outras áreas do projeto, indicando uma boa estabilidade geral.
- Além disso, durante a análise, foi observado que alguns plugins precisam de atualização para manter o ambiente do Jenkins atualizado e seguro.
- Um relatório detalhado foi gerado pelo Dependency Check, fornecendo informações cruciais para a avaliação da segurança e integridade das dependências do projeto.

Esses resultados são fundamentais que podem direcionar os desenvolvedores para a correção da vulnerabilidade identificada e realizar a atualização dos plugins necessários para manter a segurança e o desempenho do sistema.

5.3.4 Teste no Projeto 2 (PlantManager)

Esta etapa de teste foi conduzida utilizando um outro projeto chamado “PlantManager” esse foi um projeto desenvolvido em um bootcamp da Rocketseat, empresa de educação e ensino de programação com foco em web, que o autor participou.

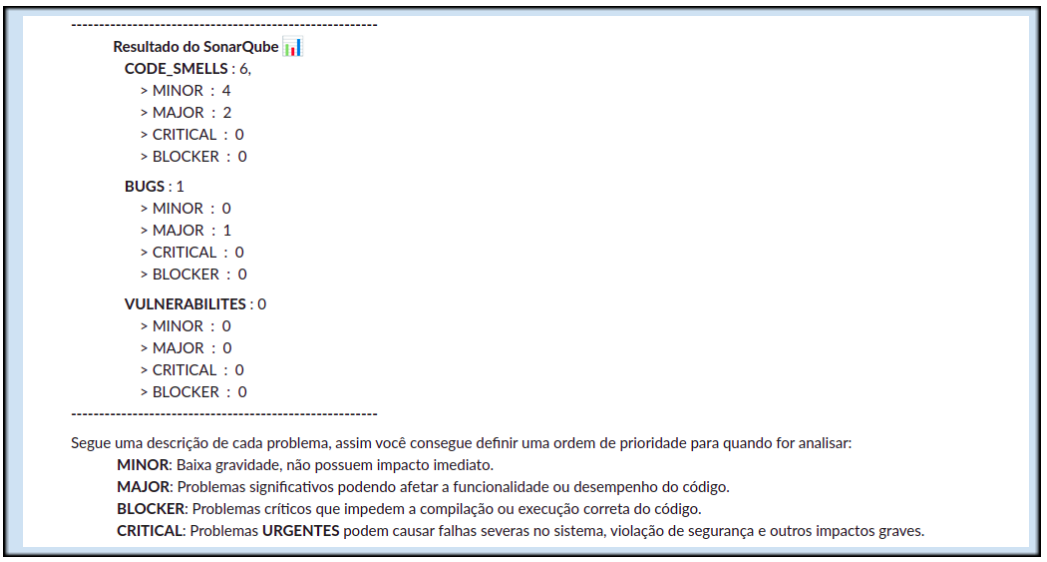
E assim como o projeto anterior esse projeto também possui sua pipeline configurada seguindo os passos da seção 4.5; Neste teste iremos reduzir bastante fluxos, devido à semelhança com o que já foi mostrado anteriormente. Nesta seção será abordado apenas o retorno dos fluxos mais importantes já conectado com o slack e com o sonar para poder demonstrar que o chatbot é dinâmico e capaz de obter informações de diferentes projetos.

5.3.4.1 Análise de Risco / Segurança

Esta etapa demonstra a análise realizada, onde o sonarqube identificou alguns problemas no código do projeto. O chatbot foi capaz de obter essas informações e retornar para o usuário de maneira limpa, separando os dados por criticidade, assim o desenvolvedor é capaz de ter uma visão mais clara e priorizar o que vai atuar primeiro.

Por limitações de escopo, não foi possível no momento colocar a solução do problema encontrado e o link do sonarqube para cada um dos problemas encontrados.

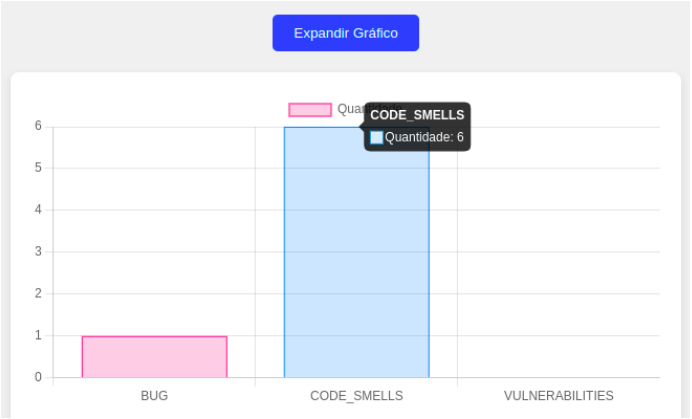
Figura 43: Slack recebendo notificação do chatbot com dados do sonarqube



Fonte: O autor (2023)

A imagem abaixo ilustra o gráfico gerado na primeira tela do chatbot, aqui já é possível observar o gráfico de barras que é gerado para o usuário;

Figura 44: Gráfico plotado com as infos do sonarqube na página inicial do chatbot

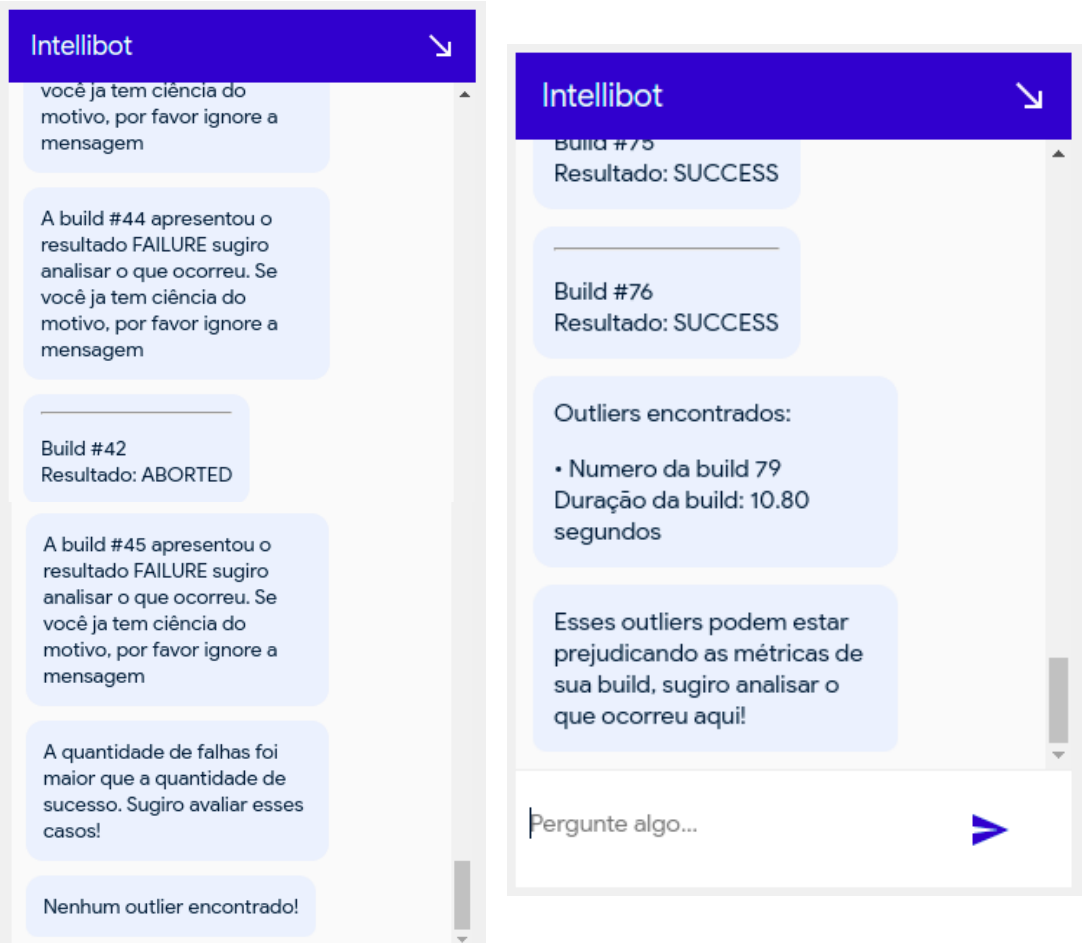


Fonte: O autor (2020)

5.3.4.2 Análise de Anomalias

A análise de anomalias realizada neste projeto, trouxe algumas informações interessantes para serem analisadas, é possível observar que algumas builds possuíam o status de “FAILURE” ou “ABORTED” além de um outlier que foi identificado como ilustrado nas figuras abaixo, ou seja, alguma coisa ocorreu nas builds da pipeline desse projeto. Com essas informações o algoritmo do projeto também faz uma contabilização das falhas e traz para o usuário que houve mais casos de falha do que sucesso em tentativas de realizar a build da pipeline. Com isso conseguimos observar que o chatbot é capaz de identificar padrões de anomalias em diferentes projetos.

Figura 45: Registros recuperados de falhas e anomalias nas pipelines



Fonte: O autor (2023)

5.4 Consolidação

Com isso foi demonstrado que o chatbot é capaz de se integrar eficazmente em diversos projetos e diferentes tecnologias, adquirindo informações pertinentes que podem ser de grande auxílio aos desenvolvedores. Essa interação não apenas fortalece a capacidade do chatbot de fornecer suporte valioso, mas também promove a eficiência e a colaboração entre equipes multidisciplinares. Os testes de simulação de ameaças envolveram a criação de cenários fictícios que replicavam situações críticas, como builds de pipelines com builds interrompidas e falhas no processo de construção, até mesmo com teste de outlier. Estes testes permitiram avaliar a capacidade do chatbot em identificar e alertar sobre comportamentos anômalos em condições controladas.

É importante reconhecer que a validação completa em um ambiente de produção com equipes reais seria uma etapa subsequente de desenvolvimento. No entanto, os resultados obtidos até o momento demonstram uma possível capacidade promissora do protótipo em identificar anomalias e contribuir para a segurança no processo de desenvolvimento de software.

5.4 Limitações e Dificuldades

Durante a condução deste trabalho, várias limitações e desafios surgiram, influenciando o escopo e a qualidade da implementação. É fundamental abordar essas limitações para um entendimento completo do trabalho realizado. As principais limitações e dificuldades incluíram:

- **Pouco espaço de tempo:** O período acadêmico reduzido disponível para o projeto tornou desafiador dedicar tempo suficiente à pesquisa, desenvolvimento e testes completos da solução. Isso pode ter impactado a profundidade da análise e a qualidade da implementação.
- **Falta de conhecimento técnico:** A necessidade de utilizar um chatbot de terceiros (Dialogflow) exigiu a aprendizagem de uma nova tecnologia. A falta de conhecimento técnico prévio nessa área pode ter limitado a capacidade de

explorar todo o potencial do chatbot e integrá-lo de maneira ideal à solução de segurança.

- **Falta de conhecimento em IA:** A aplicação de análises de segurança e detecção de anomalias mais robustas exigiria um conhecimento técnico profundo em inteligência artificial e aprendizado de máquina. A falta de experiência nessa área pode ter limitado a capacidade de implementar soluções mais avançadas e complexas.
- **Limitações de Escopo:** Devido ao foco na conclusão do trabalho de conclusão de curso, pode ter havido limitações no escopo da implementação. Alguns recursos ou funcionalidades desejáveis podem não ter sido totalmente desenvolvidos ou testados.
- **Disponibilidade de Dados Limitada:** A qualidade das análises de segurança e detecção de anomalias depende da disponibilidade de dados relevantes. A falta de acesso a dados adequados pode ter limitado a eficácia dessas análises, prejudicando a capacidade de detecção de ameaças reais.
- **Escassez de Literatura Científica e Estudos:** A pesquisa em chatbots inteligentes voltados para segurança em pipelines de CI/CD é um campo emergente e extremamente carente de recursos acadêmicos e estudos de caso relevantes. A falta de referências e artigos especializados neste tópico pode ter dificultado a obtenção de insights e melhores práticas para a implementação. Por isso que nesse contexto desafiador, como autor da pesquisa, vi a necessidade de realizar estudos e navegar entre disciplinas como segurança da informação, aprendizado de máquina, processos de desenvolvimento de software e automação de pipelines de CI/CD. Esse esforço multidisciplinar foi necessário para desenvolver uma abordagem satisfatória que atendesse às demandas específicas nesse contexto.

No entanto, essas limitações e dificuldades são oportunidades valiosas de aprendizado e servem como base para futuras melhorias e aprofundamento nas áreas técnicas necessárias para aprimorar soluções de segurança em pipelines de CI/CD. Cada desafio enfrentado durante o projeto oferece insights valiosos para o desenvolvimento de habilidades técnicas e o planejamento de projetos futuros.

6 CONCLUSÃO

Em resumo, este trabalho abordou a importante questão da segurança em pipelines de CI/CD, oferecendo uma solução na forma de uma arquitetura e a implementação de um protótipo de chatbot inteligente. Ao longo do processo, várias dificuldades foram enfrentadas, como restrições de tempo, limitações de conhecimento técnico e desafios na concepção da arquitetura. No entanto, apesar dessas barreiras, conseguimos desenvolver com sucesso um protótipo funcional que integra o Dialogflow, Jenkins, SonarQube e Slack.

A integração com o Jenkins permitiu automatizar análises de segurança, identificar anomalias e fornecer notificações em tempo real. A colaboração com o SonarQube permitiu a realização de análises estáticas de código e a detecção de problemas relacionados à qualidade e segurança. A integração com o Slack possibilitou a entrega eficaz de notificações para equipes de desenvolvimento e segurança, aprimorando assim a conscientização e a resposta a questões de segurança no ciclo de desenvolvimento.

Além disso, vale ressaltar que a integração com o Dialogflow desempenhou um papel fundamental na incorporação do sistema. Isso se deve ao fato de que o DialogFlow já oferece uma interface de chatbot inteligente pronta, juntamente com seu robusto processamento de linguagem natural. A presença dessa interface de chatbot, que já estava preparada e otimizada, acelerou significativamente a implementação.

Dessa forma, este trabalho representa um passo importante na melhoria da segurança em pipelines de CI/CD, demonstrando a viabilidade de um chatbot inteligente como uma ferramenta eficaz de auxílio aos desenvolvedores nesse contexto, contribuindo para a construção de sistemas mais seguros e confiáveis.

O código fonte do projeto pode ser encontrado no seguinte link do github: https://github.com/BrunoMSts/webhook-intellibot

6.1 Trabalhos Futuros

Como proposta para trabalhos futuros foram pontuadas várias oportunidades de melhoria e direções. São elas:

- Aprimoramento da Análise de Segurança: Investir mais tempo em pesquisas adicionais e desenvolver modelos de análise de segurança mais avançados, como a detecção de vulnerabilidades em tempo real e análises de comportamento suspeito.
- Inteligência Artificial Mais Avançada: Desenvolver recursos de IA mais robustos para permitir a compreensão de linguagem natural avançada e respostas mais contextuais.
- Expansão para Outras Plataformas de CI/CD: Estender a solução para ser compatível com uma variedade de plataformas de CI/CD além do Jenkins.
- Integração com Mais Ferramentas de Segurança: Integrar com mais ferramentas de segurança, como scanners de vulnerabilidades, firewalls de aplicativos da web (WAF) e gerenciamento de identidade e acesso (IAM).
- Análise de Dados Mais Profunda: Realizar análises de dados mais aprofundadas para identificar tendências e padrões de segurança, ajudando na prevenção proativa de ameaças.
- Estudos de Usabilidade: Realizar estudos de usabilidade para garantir que o chatbot seja intuitivo e eficaz para os desenvolvedores proporcionando uma experiência de alta qualidade.
- Documentação Completa: Fornecer documentação detalhada sobre a solução, integração e melhores práticas de segurança.

Em resumo, este trabalho serviu como um ponto de partida para abordar a segurança em pipelines de CI/CD com o desenvolvimento de um protótipo de chatbot inteligente. O potencial para melhorias é significativo, e futuros projetos podem se concentrar em aprimorar ainda mais a automação, a análise de segurança e a colaboração entre equipes de desenvolvimento e segurança.

REFERÊNCIAS

CARTER, K. Francois Raynaud on DevSecOps. IEEE Software, vol. 34, no. 5, pp. 93-96, 2017. Disponível em: <https://doi.org/10.1109/MS.2017.3571578>

CHEN, L. Continuous Delivery: Overcoming adoption challenges. Journal of Systems and Software, Volume 128, 2017, Pages 72-86. Disponível em: <https://doi.org/10.1016/j.jss.2017.02.013> Acesso em: 19 jun. 2023

DANG, Y.; LIN, Q.; HUANG, P. AIOps: Real-World Challenges and Research Innovations. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 2019, pp. 4-5. Disponível em: <https://ieeexplore.ieee.org/document/8802836>

DIALOGFLOW, Dialogflow. Documentação Dialogflow. Disponível em: <https://cloud.google.com/dialogflow/docs?hl=pt-br>

EBERT, C.; GALLARDO, G.; HERNANTES, J.; SERRANO, N. DevOps. IEEE Software, vol. 33, no. 3, pp. 94-100, May-June 2016. Disponível em: <https://doi.org/10.1109/MS.2016.68>

FLORA, J. Improving the security of microservice systems by detecting and tolerating intrusions. In 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, Washington, DC, USA, 131–134. Disponível em: <https://doi.org/10.1109/ISSREW51248.2020.00051>

GITHUB, Github. Documentação Github. Disponível em: <https://github.com>

GMI Insights. 2020 DevOps Trends Survey. Disponível em: <https://www.gminsights.com/pressrelease/devops-market>. Acesso em: 13/09/2023

GOKARNA, M.; SINGH, R. DevOps: A Historical Review and Future Works. In: 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 366-371. Disponível em: <https://doi.org/10.1109/ICCCIS51004.2021.9397235>

GRUHN, V.; HANNEBAUER, C.; JOHN, C. Security of Public Continuous Integration Services. v. 15, p. 1–10, Aug 2013. Disponível em: <https://doi.org/10.1145/2491055.2491070>

JABBARI, R.; ALI, N. B.; PETERSEN, K.; TANVEER, B. What is DevOps? A Systematic Mapping Study on Definitions and Practices. In: Proceedings of the Scientific Workshop Proceedings of XP2016 (XP '16 Workshops). Association for Computing Machinery, New York, NY, USA, Article 12, 1–11. Disponível em: <https://doi.org/10.1145/2962695.2962707>

KUGELE, Stefan et al. Optimizing Automatic Deployment Using Non-functional Requirement Annotations. Communications in Computer and Information Science. 17. 400-414. Disponível em: <https://doi.org/10.1007/978-3-540-88479-8_28>

NGROK, Ngrok. Documentação Ngrok. Disponível em: <https://ngrok.com/docs/>

MARTIN, Robert C. Arquitetura limpa: O guia do artesão para estrutura e design de software. 1. ed. São Paulo: Alta Books, 2019.

PULGAR, C. Eat your own DevOps: a model driven approach to justify continuous integration pipelines. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (MODELS '22). Association for Computing Machinery, New York, NY, USA, 225–228. Disponível em: <https://doi.org/10.1145/3550356.3552395>

RADZIWILL, Nicole; BENTON, Morgan. Evaluating Quality of Chatbots and Intelligent Conversational Agents. 2017. Disponível em:

<https://www.researchgate.net/publication/316184347_Evaluating_Quality_of_Chatbots_and_Intelligent_Conversational_Agents>

REDHAT. O que é um Webhook?. Disponível em: <https://www.redhat.com/pt-br/topics/automation/what-is-a-webhook.> Acesso em: 25/09/2023.

REDHAT, State of Kubernetes Security Report. 2023. Disponível em: <https://www.redhat.com/rhdc/managed-files/cl-state-kubernetes-security-report-262667-202304-a4-ptbr.pdf>. Acesso em: 14 set, 2023.

ROMERO, E. E. et al. Integration of DevOps Practices on a Noise Monitor System with CircleCI and Terraform. ACM Trans. Manage. Inf. Syst. 13, 4, Article 36 (December 2022), 24 pages. Disponível em: <https://doi.org/10.1145/3505228>

SCANLON, Thomas; MORALES, Jose. Revelations from an Agile and DevSecOps Transformation in a Large Organization: An Experiential Case Study. In: Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering (ICSSP'22), Association for Computing Machinery, New York, NY, USA, 77–81. Disponível em: <https://doi.org/10.1145/3529320.3529329>

SHAH, T.; PATEL, S. V. A Novel Approach for Specifying Functional and Non-functional Requirements Using RDS (Requirement Description Schema). Procedia Computer Science, Volume 79, 2016, Pages 852-860. Disponível em: <https://doi.org/10.1016/j.procs.2016.03.083>

SHAWAR, Bayan; ATWELL, Eric. Chatbots: Are they Really Useful?. LDV Forum, v. 22, p. 29-49, 2007. Disponível em: <https://www.researchgate.net/publication/220046725_Chatbots_Are_they_Really_Useful>

SONARCLOUD. Disponível em: <https://docs.sonarcloud.io/?_gl=1*1ksy22p*_gcl_au*NTQ3NjAzMDUxLjE2OTAwNTkwNDY.*_ga*MTcxNjIyMDQ4Ni4xNjkwMDU9MA..*_ga_3BQP5VHE6N*MTY5MDA1OTA0Ni4xLjAuMTY5MDA1OTA0Ni4wLjAuMA..*_ga_9JZ0GZ5TC6*MTY5MDA1OTA0Ni4xLjAuMTY5MDA1OTA0Ni42MC4wLjA>

ZDUN, U. et al. Microservice Security Metrics for Secure Communication, Identity Management, and Observability. ACM Trans. Softw. Eng. Methodol. 32, 1, Article 16 (January 2023), 34 pages. Disponível em: <https://doi.org/10.1145/3532183>