JOÃO PAULO BARBOSA PENA

IMPLEMENTANDO A CULTURA DEVOPS DENTRO DE UMA ORGANIZAÇÃO

LAVRAS – MG

2021

JOÃO PAULO BARBOSA PENA

IMPLEMENTANDO A CULTURA DEVOPS DENTRO DE UMA ORGANIZAÇÃO

Relatório de Estágio apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para obtenção do título de Bacharel

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG

2021

JOÃO PAULO BARBOSA PENA

IMPLEMENTANDO A CULTURA DEVOPS DENTRO DE UMA ORGANIZAÇÃO

Relatório de Estágio apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Ciência da Computação, para obtenção do título de Bacharel

APROVADA em 05 de Novembro de 2021.

Prof. Dr. Maurício Ronny de Almeida Souza	UFLA
Profa. Dra. Renata Teles Moreira	UFLA
Prof. Dr. Rafael Serapilha Durelli	UFLA

Prof. Dr. Maurício Ronny de Almeida Souza
Orientador

LAVRAS – MG
2021

AGRADECIMENTOS

Agradeço à Deus pelas bênçãos e glórias que colocou em minha vida.

Agradeço também aos meus pais, Cleide Barbosa e Paulo César Pena, que são a minha base. Sempre me apoiaram e acreditaram nos meus sonhos, me ajudando em todas as dificuldades da vida. Algum dia Deus irá retribuir todo o sacrifício que eles fizeram por mim para que eu alcançasse o meu sucesso. Eu amo vocês!

Agradeço a minha incrível namorada, por melhorar minha vida, sempre me apoiar e incentivar em momentos de fraqueza. Espero um dia poder retribuir tudo isso. Amo você.

Agradeço, ainda, a toda a minha família,pais, irmãos, tios e tias, primos e minha namorada. Minha família é a minha base e sem eles eu não estaria aqui. Poder compartilhar tudo com eles é uma honra.

Agradeço ao meu orientador, professor Maurício, por todo o tempo gasto me ajudando, compartilhado seu conhecimento no longo dessa jornada, e também por ser um grande amigo.

Agradeço à cafeina, sob forma de café e Coca-Cola e também aos energéticos, que me ajudaram a não dormir ou cochilar quando eu mais necessitava e menos podia.

Nenhum esforço faz sentido, se você não acreditar em si mesmo
(Maito Gai(Naruto Shippunden))

RESUMO

Uma iniciativa da Agência Zetta, para o ano de 2020 foi implantar a cultura Devops na organização. Devops é a combinação dos termos "desenvolvimento"e "operações". No entanto, ela representa um conjunto de ideias e práticas que ultrapassam o significado desses dois termos. O Devops inclui segurança, maneiras colaborativas de trabalhar, análise de dados e muitas outras práticas e conceitos. Neste cenário, esse relatório de estágio supervisionado, descreve as atividades realizadas pelo autor nos estudos e implantações de Devops na Agência Zetta, no período de maio a dezembro de 2020.

Palavras-chave: DevOps. Integração Contínua. Entrega Contínua. *Deploy*. *Pipiline*

ABSTRACT

An initiative of the Zetta Agency for the year 2020 was to implement the Devops culture in the organization. Devops is the combination of the terms "development"and "operations". However, it represents a set of ideas and practices that go beyond the meaning of these two terms. Devops includes security, collaborative ways of working, data analysis, and many other practices and concepts. In this scenario, this supervised internship report describes the activities performed by the author in the studies and deployments of Devops at Zetta Agency, from May to December 2020.

Keywords: DevOps. Continuous integration. Continuous delivery. Deploy. Pipiline

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este relatório técnico supervisionado é realizado para disciplina PRG610 - Estágio Supervisionado/TCC, com objetivo de conclusão do curso de graduação de Ciência da Computação pela Universidade Federal de Lavras – UFLA.

O estágio foi realizado na Zetta - Agência UFLA de Inovação, Geotecnologia e Sistemas Inteligentes, que se localiza em Lavras/MG, O estágio teve a duração entre 18 de Maio de 2020 a 31 de Dezembro de 2020, com total de 1008 horas. Essa foi uma excelente oportunidade na ingressão ao mercado de trabalho, onde foi possível aperfeiçoar e complementar habilidades e competências adquiridas ao decorrer da graduação.

Este relatório técnico relata as atividades realizadas ao decorrer do estágio, de como se implementar uma cultura DevOps em uma organização, utilizando algumas ferramentas e metodologias.

1.1 Objetivos do estágio

O objetivo do estágio foi a implantação de DevOps na organização e, como consequência, ensinar ao estagiário como ingressar no mercado de trabalho de DevOps, aprendendo a cultura e ferramentas DevOps e validar o conteúdo apresentado no decorrer da graduação e nos cursos realizados durante o estágio. Durante o estágio foram realizadas algumas atividades em que podem ser dividas em 5 etapas: (i) Treinamento, (ii) Estudos sobre DevOps (iii) Execução do projeto Piloto , (iv)Implantação de DevOps no Projeto Análise automatizada dos cadastros em Santa Catarina, e (v) Replicação em outros projetos.

Na primeira etapa, o estagiário realizou um treinamento da organização, que consistiu em assistir vídeo aulas sobre ferramentas que a empresa utilizava na plataforma Alura [1]. Na segunda etapa, o estagiário realizou dois cursos na área de DevOps para poder aplicar o conhecimento adquirido dentro da organização. Na terceira etapa, o conhecimento adquirido através dos cursos realizados foram aplicados em um projeto piloto pensando em sua replicação para outros projetos. Na quarta etapa, o estagiário implantou a cultura DevOps em um projeto da organização, e por fim na última etapa, foi replicado a cultura DevOps em outros dois projetos na Empresa.

[1] https://www.alura.com.br/

O principal objetivo é relatar os resultados e conclusões do estágio de modo a sintetizar o que foi aprendido e fazer um paralelo com o curso de Ciência da Computação, visando abordar os pontos que podem trazer uma comunicação maior entre o bacharelado e o ingresso no mercado de trabalho.

1.2 Organização do trabalho

Além deste capítulo introdutório, este trabalho foi organizado em quatro capítulos contendo as principais atividades realizadas durante o estágio:

- Capítulo 2 Sobre a empresa: tem o objetivo de descrever a organização em que o estagiário trabalhou.
- Capítulo 3 Conceitos e Tecnologias utilizadas: possui o intuito de contextualizar o leitor quanto às tecnologias utilizadas no estágio.
- Capítulo 4 Atividades realizadas: descreve as atividades realizadas pelo estagiário durante o estágio.
- Capítulo 5 Conclusão: foca em relatar as conclusões traçando um paralelo entre o estágio e o curso de bacharelado de Ciência da Computação.

2 SOBRE A EMPRESA

Este capítulo tem o intuito de descrever a organização em que o estágio foi efetuado, uma breve contextualização de sua história, assim como a área de atuação e objetivo.

2.1 Agência Zetta

A Agência Zetta é uma agência da UFLA de inovação, geotecnologias e sistemas inteligentes vinculada à Universidade Federal de Lavras, uma das melhores instituições públicas de ensino do país. Ela é formada por pessoas que trabalham todos os dias buscando soluções tecnológicas ambientais que possam contribuir com um mundo melhor (ZETTA, 2021).

As pessoas que trabalham na Agência Zetta são comprometidas com a busca de soluções inovadoras para o Brasil. Tem como legado um projeto visionário que teve início há mais de 25 anos, de um sonho de semear concepções que possam contribuir para a melhoria da sociedade, com tecnologia, inovação e, acima de tudo, com responsabilidade social. É motivado por desafios que os fizeram crescer e os tornaram referência na área de atuação, com a confiança de quem sabe o que faz (ZETTA, 2021).

A missão da Agência Zetta é Potencializar o conhecimento gerado por pesquisadores da UFLA, transformando-o em produtos inovadores. Gerar frutos que ultrapassam os limites da pesquisa e promovem o desenvolvimento sustentável do País (ZETTA, 2021).

A partir do conhecimento especializado proveniente de nossa Universidade mãe, tem-se um grupo de pesquisadores e técnicos focados em Pesquisa e Inovação. Trabalhando para o desenvolvimento de soluções inovadoras para o agronegócio brasileiro em busca da prosperidade sustentável, ética e responsável. Assim, trilhando o caminho através de uma ampla experiência na execução de centenas de projetos espalhados por todo Brasil (ZETTA, 2021).

A Agência Zetta tem como quatro pilares: (i) criatividade, (ii) inovação, (iii) tecnologia e, principalmente, (iv) sociedade. Conforme ZETTA (2021), esses pilares são definidos como:

- A Criatividade, ela é guiada na busca de soluções criativas e estratégias arrojadas para atender de forma plena aos desafios colocados por nossos parceiros;

- A Inovação, ela é o pilar e simboliza a vocação pela ciência e pelas pesquisas na constante busca de novas maneiras de enxergar os problemas, criando métodos inovadores que possam impactar de forma positiva a sociedade;

- A Tecnologia, tão essencial nos dias atuais, este é o motor de propulsão! Utilizando da tecnologia para materializar os sonhos dos parceiros. Com a busca diária por novas tecnologias e a constante capacitação dos colaboradores, perseguimos a excelência tecnológica para dar suporte ao desenvolvimento sustentável da sociedade;

- A Sociedade, é um grande ecossistema e cada qual exerce uma função peculiar para sua manutenção! A nossa função é de usar a criatividade, a inovação e a tecnologia para melhorar a vida das pessoas, esse é o nosso objetivo e, também, a nossa maior finalidade;

Era escassa a interação entre os times de desenvolvimento e infraestrutura. Esta só acontecia quando ambos os times necessitavam da ajuda um do outro. Com o intuito de introduzir a cultura DevOps em alguns projetos, foi proposto a realização de uma reunião quinzenal, a fim de discutir assuntos referentes ao projeto que envolvia tanto o estagiário, quanto as equipes de desenvolvimento e infraestrutura. Com isto, as equipes puderam interagir melhor, contribuindo com a evolução da cultura DevOps no desenrolar do projeto.

Os dados sensíveis da Agência Zetta eram alocados em um repositório privado do Gitlab juntamente com o projeto. Este repositório contém variáveis de ambiente e links para sites internos da própria organização para baixar dependências dos projetos. Apenas os responsáveis pelo projeto tinham acesso a este repositório.

3 CONCEITOS E TECNOLOGIAS UTILIZADAS

Este capítulo tem como objetivo contextualizar as tecnologias e conceitos utilizados pelo estagiário. Somado a isto, também estão descritos processos utilizados no desenvolvimento web e na cultura DevOps aplicados durante o estágio.

3.1 DevOps

O DevOps é a combinação de filosofias culturais, práticas e ferramentas que aumentam a capacidade de uma organização de distribuir aplicativos e serviços em alta velocidade: otimizando e aperfeiçoando produtos em um ritmo mais rápido do que o das organizações que usam processos tradicionais de desenvolvimento de software e gerenciamento de infraestrutura.(AWS, 2021b) Essa velocidade permite que as organizações atendam melhor aos seus clientes e consigam competir de modo mais eficaz no mercado. (AWS, 2021b)

O DevOps tem como princípio criar um processo de entrega confiável, previsível e passível de repetições, gerar grandes reduções no tempo de ciclo e entrega novas funcionalidades e correções aos usuários rapidamente, e gerando valor ao produto. (GAEA, 2021) Organizações que adotam o DevOps podem ter vantagens sobre organizações que ainda adotam em relação aos objetivos do DevOps: (i) Redução de custos, (ii) Aumento da confiabilidade, (iii) Agilidade nas entregas, (iv) Colaboração entre equipes, e (v) Segurança. (GAEA, 2021)

- **Redução de custos:** O desenvolvimento de software foi, por um longo tempo, considerado uma atividade cara, não apenas pela tecnologia, mão de obra especializada e ferramentas exigidas para o trabalho, mas também pelos problemas ao decorrer do desenvolvimento dos projetos. Com a cultura DevOps, vários aspectos do desenvolvimento de software são melhorados, como feedback, comunicação, integração, entre outros. O resultado disso é uma redução no tempo de entrega, mais qualidade do produto final e mais resultado. Isso tem um impacto direto sobre o custo da implementação de sistemas, já que existe uma redução significativa de erros, que demandam tempo de trabalho dos times, adequações pela falta de comunicação e retrabalho, poupando recursos que podem ser redirecionados para novos projetos ou incrementações. (GAEA, 2021)

- **Aumento da confiabilidade:** Outro ponto muito interessante da cultura DevOps é o aumento da confiabilidade. Isso se dá devido ao registro e ao armazenamento de logs de desenvolvimento, contendo toda e qualquer alteração realizada em código. Sendo assim, sempre que um erro for identificado na produção, é possível rastreá-lo rapidamente apenas verificando os logs dos arquivos envolvidos, aplicando as correções e publicando o novo código. Além disso, o aumento da confiabilidade também é derivado da automatização de testes, que são rodados sempre que uma nova integração é realizada. Isso garante uma minimização dos riscos de erros em produção. (GAEA, 2021)

- **Agilidade nas entregas:** Entre as vantagens mais conhecidas do DevOps, está a agilidade e a Integração Contínua. Isso é possível graças à automatização do processo de publicação de software em produção. Por meio de ferramentas de implantação de sistemas, é possível publicar de forma muito mais simples e rápida todas as alterações de código realizadas pelo time de desenvolvimento. (GAEA, 2021)

- **Colaboração entre equipes:** A comunicação interna é fundamental para o sucesso do desenvolvimento de software, uma vez que as equipes são formadas por vários profissionais de áreas diferentes. A ausência desse contato pode levar a problemas, atrasos e até mesmo a prejuízos devido à falta de entregas. A integração promovida por uma estratégia DevOps acaba eliminando essa situação. Todo o time de desenvolvimento é aproximado, desfazendo barreiras que poderiam criar ruídos de comunicação entre os vários profissionais envolvidos, como analistas, desenvolvedores, testers, entre outros. Assim, todos podem observar as atividades realizadas pelos outros integrantes dos projetos e dos times, organizando melhor o seu trabalho e garantindo a entrega. (GAEA, 2021)

- **Segurança:** Nos últimos anos, houve um grande aumento no número de ataques virtuais, sendo que os cibercriminosos estão cada vez mais refinados em suas técnicas, utilizando de toda e qualquer brecha para invadir aplicações e roubar dados. Ao aplicar uma cultura DevOps, pode-se rastrear e garantir a conformidade de código, mantendo um padrão de qualidade esperado dentro da segurança da informação. (GAEA, 2021)

Dadas essas características, as subseções a seguir apresentam alguns conceitos e atividades chaves do DevOps aplicados durante o estágio.

3.1.1 Integração Contínua

Segundo Ebert et al. (2016), Integração Contínua (*Continuous Integration*, ou CI) é um conjunto de práticas de desenvolvimento de software que visam verificar e manter a qualidade do produto. Geralmente os desenvolvedores, reúnem suas alterações de código em um repositório central. Com a Integração Contínua, os desenvolvedores confirmam um repositório compartilhado usando um sistema de controle de versão, como o Git. Após cada confirmação no repositório, o código será submetido a construção e passará por teste unitário, teste de componentes, teste rápidos para verificar e manter a qualidade do código.

Integração Contínua é uma prática recomendada ágil e de DevOps para integrar, como parte da rotina, alterações de código na ramificação principal de um repositório e testar as alterações com o máximo de antecedência e frequência possível. O ideal é que os desenvolvedores integrem os códigos todos os dias, se não várias vezes ao dia. (ATLASSIAN, 2021)

Os principais objetivos da Integração Contínua são encontrar e investigar bugs mais rapidamente, melhorar a qualidade do software e reduzir o tempo que leva para validar e lançar novas atualizações de software. No passado, os desenvolvedores de uma equipe podiam trabalhar isoladamente por um longo período e só juntar suas alterações à ramificação mestre quando concluísse seu trabalho. Dessa forma, a junção das alterações de códigos era difícil e demorada, além de resultar no acúmulo de erros sem correção por longos períodos. Estes fatores dificultavam uma distribuição de atualizações rápida para os clientes. Com a integração continuada, os desenvolvedores frequentemente confirmam um repositório compartilhado usando um sistema de controle de versão, como o Git. Antes de cada confirmação, os desenvolvedores podem escolher executar testes de unidade locais em seus códigos como uma camada de verificação extra anterior à integração. Um serviço de Integração Contínua cria e executa automaticamente testes de unidade nas novas alterações de código para destacar imediatamente todos os erros. (AWS, 2021a)

3.1.2 Entrega Contínua

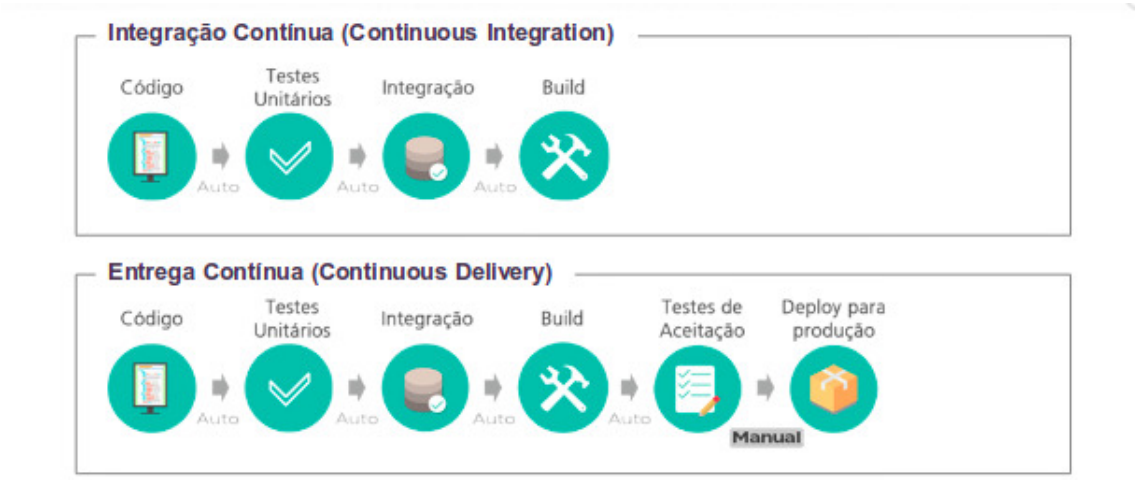
A Entrega Contínua (*Continuous Delivery*, ou CD) é uma prática de desenvolvimento de software na qual as alterações de código são automaticamente preparadas para uma liberação para produção. Um pilar do desenvolvimento de aplicativos modernos, a distribuição contínua expande com base na Integração Contínua implantando todas as alterações de código em um ambiente de teste

e/ou ambiente de produção, após o estágio de criação. Esses testes podem incluir testes de IU, carga, integração e confiabilidade de API, etc. Isso ajuda desenvolvedores a validar com maior precisão atualizações e descobrir problemas de modo preventivo. (EBERT et al., 2016)

Juntamente com a Integração Contínua, a Entrega Contínua e a Implantação Contínua, há práticas que automatizam as fases da entrega do software. São práticas que permitem às equipes de desenvolvimento lançar novos recursos, melhorias e correções para os clientes com mais velocidade, precisão e produtividade. A Entrega Contínua e a implantação contínua têm muito em comum. Para entender as diferenças entre essas práticas e descobrir qual delas você quer implementar, é necessário identificar as fases da entrega de software que podem ser automatizadas. (AZURE, 2021)

O processo de Entrega Contínua começa justamente onde a Integração Contínua acaba. Ou seja, o CD se encarrega de automatizar a entrega de aplicativos para ambientes de infraestrutura selecionados. Em outras palavras, uma vez que o novo código seja validado nos testes de integração, o CD automatiza o seu envio aos servidores correspondentes. Isso é particularmente necessário, pois no desenvolvimento de um aplicativo, é comum que as equipes trabalhem com vários ambientes e camadas diferentes (como teste, desenvolvimento etc). Dessa forma, é possível que uma base de códigos preparada para implantação em qualquer ambiente produtivo. O processo de CD garante que as alterações de código cheguem ao lugar certo. (DBACORP, 2021)

Figura 3.1 – Explicação sobre CI/CD



Fonte: https://brdsolucoes.com.br/2018/12/24/integracao-continua-voce-esta-fazendo-isso-do-jeito-certo/

Para reunir as informações sobre CI/CD, a Figura 3.1 acima apresenta dois *pipelines* e ressalta bem a diferença entre CI/CD. É importante ressaltar que a Integração Contínua CI é o processo por trás

da automação de testes e validação de alterações no código. Já a Entrega Contínua CD é o processo de automação de entrega dessas alterações na infraestrutura do código, entre outros processos adicionais, e isso é um ciclo infinito até o projeto morrer.

3.1.3 Ferramentas de DevOps

Existem algumas ferramentas para o CI/CD. Uma delas é o Jenkins[1], uma ferramenta de código aberto, com uma instalação e configuração simples. Com o Jenkins, as organizações podem dar o primeiro passo a se introduzir uma cultura DevOps, pois junto com outras ferramentas gerenciam e controlam os processos de entrega de software em todo o ciclo de vida, incluindo construção, documento, teste, pacote, estágio, implantação, análise de código estático como ilustrado na figura 3.2.

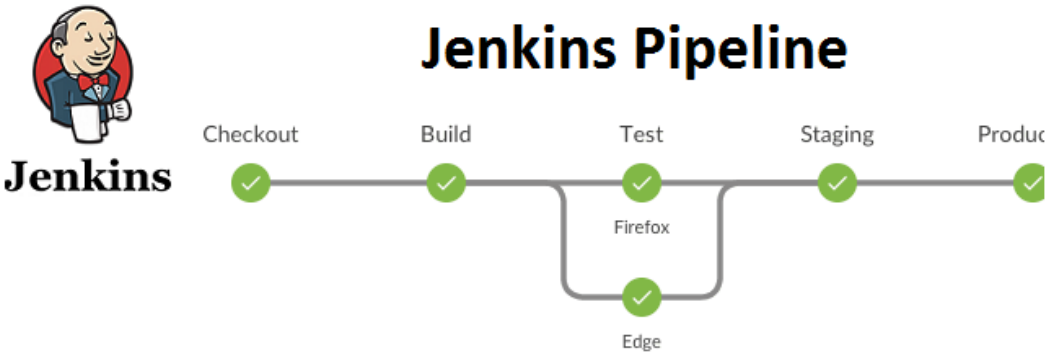
Um pipeline de CI/CD consiste em uma série de etapas a serem realizadas para a disponibilização de uma nova versão de um software. Os *pipelines* de integração e Entrega Contínuas CI/CD são uma prática que tem como objetivo acelerar a disponibilização de softwares, adotando a abordagem de DevOps ou de engenharia de confiabilidade de sites (SRE). (REDHAT, 2021)

O Jenkins é um servidor de automação de código aberto. Com o Jenkins, as organizações podem acelerar o processo de desenvolvimento de software automatizando-o como é ilustrado pela Figura 3.2. Pode-se configurar o Jenkins para observar quaisquer alterações de código em lugares como GitHub, Bitbucket ou GitLab e fazer uma construção automática com ferramentas como Maven e Gradle. Também é possível utilizar a tecnologia de contêiner, como Docker e Kubernetes, iniciar testes e, em seguida, realizar ações como retroceder ou avançar na produção. (JENKINS, 2021)

Pode-se ressaltar muitas funcionalidades, como construções periódicas, *Mailler*, integrações com plataformas de versionamento de código, ferramentas de análise de código Sonarqube e ferramentas de automatização de taferas o Ansible como ilustrado na Figura 3.3. A construção é o empacotamento do software pronto para sua instalação pelo usuário ou servidor. É a última fase do desenvolvimento de software antes de sua publicação em ambientes. As construções periódicas podem ser iniciadas de muitas maneiras, o Jenkins possibilita o agendamento cron, podendo agendar essas construções de minuto a minuto. Ele também possibilita uma integração com plataformas de

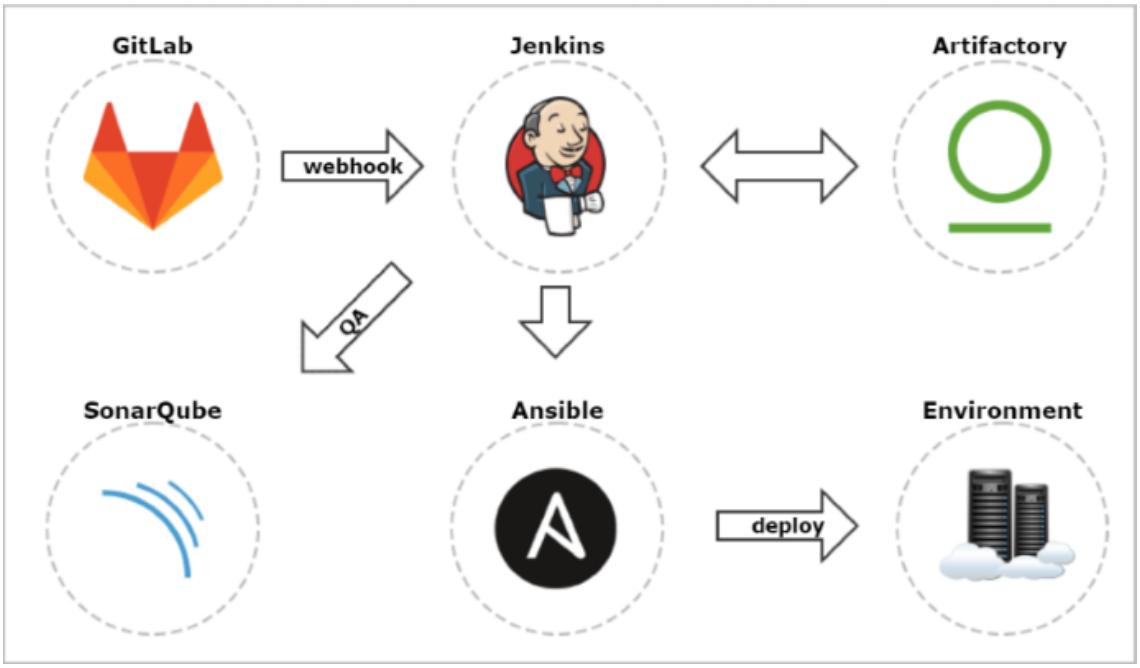
[1] https://www.jenkins.io/

Figura 3.2 – Etapas do Jenkins



Fonte: https://ichi.pro/pt/jenkins-pipeline-com-gitlab-para-projetos-java-231864011383381

Figura 3.3 – Integrações



Fonte: https://piotrminkowski.com/2017/02/10/how-to-setup-continuous-delivery-environment/

versionamento de código, de tal maneira que quando ocorrer qualquer alteração no repositório, ele identifica e realiza a construção.

O *Mailler* tem como seu principal configurar notificações por e-mail a fim de que haja resultados de construção, enviando um e-mail aos destinatários caso ocorra os seguintes problemas:

- Falha na construção;

- Construção Instável;
- Construção bem-sucedida após uma falha na construção, indicando que uma crise acabou;
- Compilação instável após uma bem-sucedida, indicando que há uma regressão;

O SonarQube[2] é uma ferramenta automática de revisão de código para detectar *bugs*, vulnerabilidades e *Code smells* em seu código. A Figura 3.5 ilustra o ciclo do Sonarquebe, onde ele pode se integrar ao seu fluxo de trabalho existente para permitir a inspeção contínua de código em todas as ramificações do projeto e solicitações de *pull*. (SONARQUBE, 2021)

O Sonarqube trabalha com várias métricas sobre como melhorar a qualidade do código, entre elas estão:

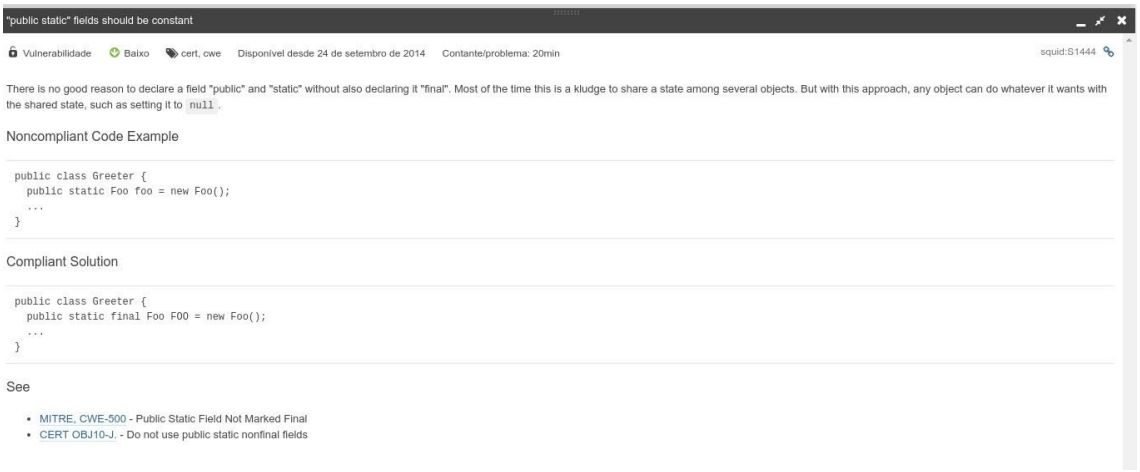
- Complexidade (ciclomática e cognitiva);
- Código duplicado;
- Quantidade de Problemas;
- Tamanho (quantidade de linhas de código, número de classes, etc...);
- Cobertura de testes;
- Índice de Manutenibilidade, Confiabilidade e Segurança;

O SonarQube se baseia em regras pré-definidas para analisar o código. Uma regra é uma boa prática e cada linguagem possui um grupo de regras relacionadas. Toda a regra possui uma descrição, normalmente, com exemplo de código e links para descrições mais detalhadas, o que ajuda o desenvolvedor a entender e resolver o problema relacionado, como mostra a Figura 3.4 .(MAUDA, 2021)

O Ansible[3] é uma ferramenta de automação de TI. Ele pode configurar sistemas, implantar software e orquestrar tarefas de TI mais avançadas, como implantações contínuas ou atualizações contínuas com tempo de inatividade zero. Os principais objetivos do Ansible são simplicidade e facilidade de uso. Ele também tem um forte foco em segurança e confiabilidade, apresentando um mínimo de partes

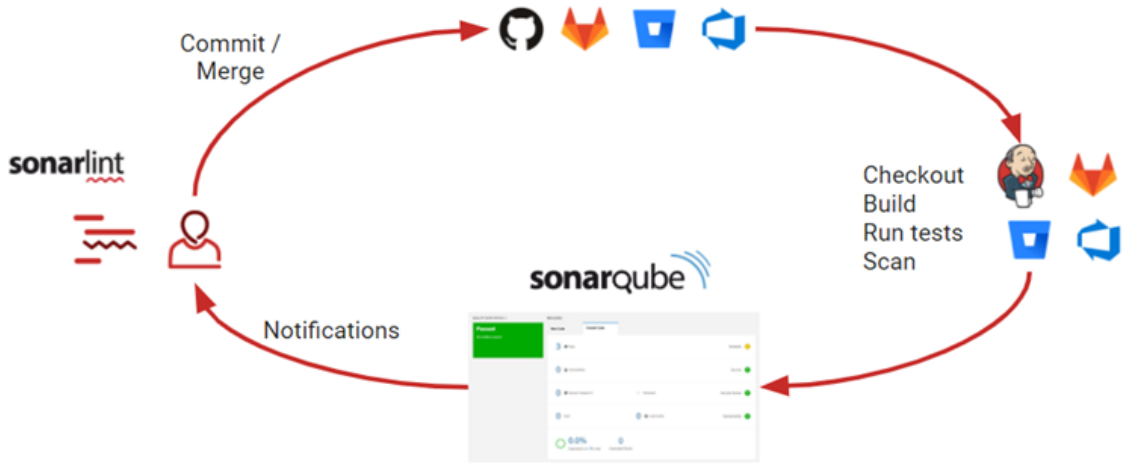
[2] https://www.sonarqube.org/
[3] https://www.ansible.com/

Figura 3.4 – Regras do Sonarquebe



Fonte: http://www.mauda.com.br/?p=2248

Figura 3.5 – Ciclo do Sonarquebe



Fonte: https://docs.sonarqube.org/latest/

móveis, uso de OpenSSH[4] para transporte (com outros transportes e modos de puxar como alternativas) e uma linguagem que é projetada em torno da auditabilidade por humanos - mesmo aqueles não familiarizados com o programa. (ANSIBLE, 2021)

[4] https://www.openssh.com/

3.2 Gerência de Configuração de Software

Gerência de configuração de software, gerência de configuração ou ainda gestão de configuração de software é uma área da engenharia de software responsável por fornecer o apoio para o desenvolvimento de software. Suas principais atribuições são o controle de versão, o controle de mudança e a auditoria das configurações. (WIKIPIDIA, 2021)

Gerência de Configuração de Software (GCS) é um conjunto de atividades de apoio que permite a absorção ordenada das mudanças inerentes ao desenvolvimento de software, mantendo a integridade e a estabilidade durante a evolução do projeto (PRONUS, 2021). Ao decorrer do estágio, foram aprendidos e utilizados duas ferramentas para Gerência de Configuração de Software:

- Git[1] é um software livre de controle de versão de arquivos. O sistema funciona de maneira descentralizada, isto é, podem existir cópias diferentes de um mesmo repositório em diferentes computadores, o que permite que um grupo trabalhe em um mesmo software simultaneamente;
- Gitlab[2] é uma plataforma para compartilhamento e hospedagem de projetos que possibilita a utilização da ferramenta git de maneira remota, simplificando a gestão dos projetos e facilitando o trabalho coletivo sobre um mesmo software.

[1] https://git-scm.com/
[2] https://about.gitlab.com/

4 ATIVIDADE REALIZADAS

Neste capitulo serão descritas as atividades realizadas pelo estagiário na Agência Zetta no periodo de Maio de 2020 até Dezembro de 2020. As atividade incluem: Treinamento (seção 4.1), Estudos de DevOps (seção 4.2), Execução de Projeto Piloto (seção 4.3), Implantação de DevOps Projeto Análise automatizada dos cadastros em SC (seção 4.4), Replicação em outros projetos (seção 4.5).

O cronograma apresentado na Figura 4.1, apresenta como as atividades foram distribuídas ao longo do estágio.

Figura 4.1 – Cronograma de atividades realizadas

CRONOGRAMA DE ATIVIDADES REALIZADAS														
ETAPAS	MAR	ABR	ABR	MAI	MAI	JUN	JUN	JUL	JUL	AGO	AGO	SET	SET	OUT
1 - Treinamento	x	x												
2 - Estudos sobre DevOps		x	x	x										
3 - Execução de Projeto Piloto					x	x	x							
4 - Implatação de DevOps Projeto Análise automatizada dos cadastros em SC								x	x	x	x			
5 - Replicação de Devops em outros Projetos												x	x	x

Fonte: Elaborada pelo autor

4.1 Treinamento

Foi realizado um treinamento, para um nivelamento de recéns chegados ao estágio. O treinamento realizado teve duração de 1 mês, com o objetivo de conhecer as ferramentas e tecnologias utilizadas pela empresa. O treinamento consistiu em um conjuto de vídeos aulas da plataforma Alura. Não houve nenhuma supervisão durante o treinamento, por conta de ser uma nova área dentro da organização. O treinamento foi dividido em dois módulos de estudo: *frontend* e *backend*.

No módulo de "*frontend*", consistiu em aprender conceitos básicos de HTML, CSS, Java*script*, e o React. Esse módulo contou com 52 aulas no total, divididas em submódulos com as tecnologias citadas acima.

No módulo de "*backend*", o objetivo foi aprender conceitos básicos controle de versão de código utilizando Gitlab, banco de dados relacional com Mysql, e por fim dois frameworks em java, Play Framework e Spring Boot. Esse módulo contou com 71 aulas no total, divididas em submódulos com as tecnologias citadas acima.

Como resultado deste treinamento, o estagiário pôde entender melhor a estrutura de projetos de desenvolvimento Web, que seriam o foco da implantação de DevOps na organização.

4.2 Estudos de DevOps

Após o treinamento, a empresa alocou o estagiário durante quase 2 meses em um estudo focado em DevOps. Essa seção é dedicada a mostrar os principais estudos realizados na área de DevOps durante o estágio.

4.2.1 Integração contínua com testes, utilizando Jenkins

O curso "Integração contínua com testes utilizando Jenkins", foi o primeiro curso realizado durante o período de estágio, com um foco mais na prática, foi o primeiro contato com a área de DevOps. O curso se encontra na plataforma da Udemy planejado pelo instrutor Francisco Wagner Costa Aquino, com 78 aulas separadas por 11 seções, com um total de 20 horas de duração.

O objetivo do curso foi aprender como criar um processo automatizado de integração e *deploy* contínuo CI/CD para as aplicações utilizando a ferramenta Jenkins. Sempre, claro, passando por vários processos de testes visando garantir que novas versões do produto entrem no ar de forma rápida e segura. Também, foi proposto como adicionar ao processo de integração contínua, testes unitários e funcionais, utilizando ferramentas populares como JUnit[1], RestAssured[2] e Selenium[3]. Foi ensinado a realizar uma análise de qualidade e cobertura de código testado, através do Sonarqube, assim como, o histórico dos testes e avisos sempre que tiver problemas na sua construção.

Como resultado deste estudo, o estagiário teve maior compreensão da ferramenta Jenkins, que seria implantada na organização, e sobre os aspectos teóricos e práticos do processo de integração contínua.

4.2.2 DevOps Master

Ao contrário do primeiro curso, o segundo curso realizado foi totalmente teórico, planejado pela HNZ Consultoria. As aulas foram ministradas pelo professor Heinz Dieter Nevermann Zamorano. O curso teve duração de 20 horas, em 5 aulas ao vivo, de segunda a sexta, dividido em 5 tópicos: (i)

[1] https://mvnrepository.com/artifact/junit/junit
[2] https://rest-assured.io/
[3] https://www.selenium.dev/

Adoção do DevOps; (ii) Planejamento, Requisitos e Desenho; (iii) Desenvolvimento e Implantação; (iv) Operação e Dimensionamento; (v) Condições de Fim de Vida de um Produto ou Serviço;

O curso teve como objetivos:

- desenvolver competências para avaliar, mensurar as diversas teorias atuais da TI, seus sintomas e causas raiz, para determinar ações que aumentem a eficácia dos processos;
- Desenvolver competências para avaliar que cuidados e necessidades culturais precisam ser integrados, bem como que características tornam as equipes altamente produtivas em um ambiente mutante, e o que anula o trabalho ágil nas práticas DevOps;
- Investigar que práticas convencionais, como ITSM, Agile Scrum, atividades de desenvolvimento e gerenciamento de projetos, precisam de adaptações para o DevOps funcionar eficazmente e por fim Influenciar a organização com boas práticas do DevOps Orientar e desenvolver planos de melhorias baseado nos principais sintomas da organização Capacitar aos alunos para efetuar o exame Exin DevOps Master;

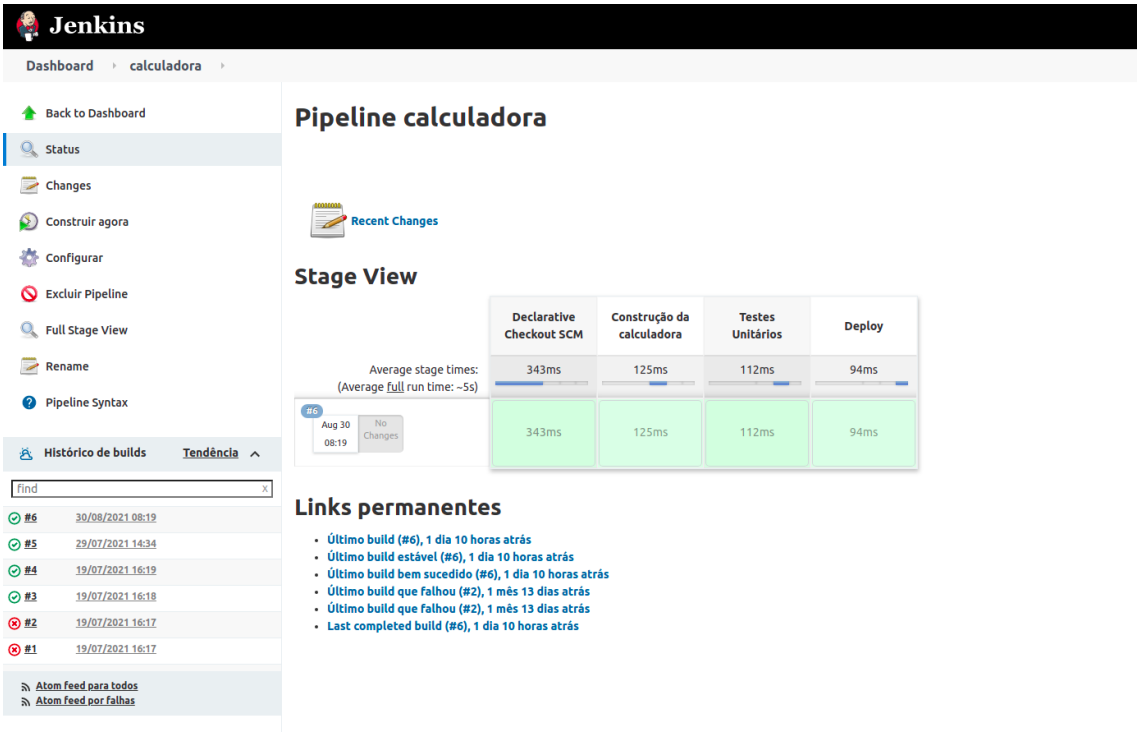
Como resultado deste estudo, o estagiário pôde identificar boas práticas e conceitos gerais do DevOps. Este conhecimento adquirido foi fundamental para a realização da implatação de DevOps nos projetos da organização.

4.3 Execução de Projeto Piloto

Foi realizado um projeto piloto, aplicando todos os conhecimentos adquiridos durante os treinamentos e cursos realizados. Para isso, Foi construída uma calculadora, utilizando java com o framework Spring Boot. Para sua construção foi utilizado o Maven, e seu controle de versão foi realizado com o Github.

Após a calculadora estar finalizada, foi elaborado um teste unitário com a dependência Junit do Maven, para testar o método de soma. Com o teste finalizado, foi iniciada a fase de construção do *pipeline*, com o objetivo de realizar o CI/CD e automação do *deploy*. Com o auxílio da ferramenta Jenkins, foi projetado um *pipeline* que após testado e validado, seria replicado em outro projetos da organização, mas adaptando em cada projeto. O *pipeline* da calculadora na Figura 4.2, apresenta as jobs do Jenkins e seus estágios finalizados.

Figura 4.2 – Pipeline do Projeto Piloto



Fonte: Elaborada pelo autor

O *pipeline* se inicia no primeiro estágio, *"Declarative Checkout SCM"*, onde cada minuto o Jenkins realiza uma verificação no repositório do projeto para saber se ocorreu qualquer alteração no código da calculadora submetida ao o repositório do Github. O *pipeline* está conFigurado no arquivo Jenkinsfile e mantido no repositório do Projeto. O arquivo Jenkinsfile foi escrito na linguagem YML, seu conteúdo são os comandos para se efetuar a automatização do projeto.

O segundo estágio do *pipeline* é a *build* da calculadora, isto é, executar um *script* para construir o projeto e validar se o código não está instável. Após esse estágio, é notório um controle maior sobre as instabilidades do projeto em novas *features*, pois, se algum desenvolvedor da equipe, submeter algum trecho de código que torne o projeto instável, o *pipeline* irá detectar e falhar imediatamente sua *job*, mandando um Email para esse responsável do projeto. Caso o *build* esteja correto, não tenha nada instabilizando o projeto, pode-se seguir para o próximo estágio.

O terceiro estágio é o teste unitário, apenas para testar o método principal da calculadora, sua soma. O *pipeline* irá executar o teste unitário e verificar se o teste está correto. Caso o teste unitário

esteja correto, o *pipeline* irá para seu próximo estágio, caso falhe, mandará uma notificação por Email, para o responsável pelo projeto.

No último estágio, é quase garantido que o projeto não está instável ou apresentando erros em seus métodos, agora será seu *deploy*. Com auxílio da ferramenta de containers Docker, toda vez que for realizado o *deploy*, tem-se três comandos Docker para isso. O primeiro comando foi para construir a imagem Docker, o segundo comando foi para parar a imagem atual que está em execução num container, e o último passo foi ativar a nova imagem e gerar um novo container, e com isso, foi finalizado o estágio do *deploy*.

4.4 Implantação de DevOps no Projeto Análise automatizada dos cadastros em Santa Catarina

Esse foi o primeiro projeto a ser introduzido uma integração e entrega contínua seguindo a cultura DevOps. O principal motivo por esse projeto ter sido escolhido como o primeiro, foi que ele é um projeto em que a equipe de desenvolvedores responsáveis, enfrentavam dificuldades em realizar o *deploy* para ambiente de desenvolvimento e homologação.

Partindo dessa dificuldade, foi construído um *pipeline* no intuito de facilitar o *deploy* do projeto, e também, foi adicionado uma análise estática do código para ajudar os desenvolvedores nas melhores práticas de desenvolvimento do projeto.

Ao longo da criação do *pipeline* do projeto da piloto, notou-se que alguns estágios vão se repetir em outros projetos. Partindo disso, as explicações dos estágios dos próximos projetos, serão apenas nas diferenças em relação ao *pipeline* do projeto piloto.

Como o projeto foi construído com as mesmas tecnologias utilizadas pelo projeto piloto, o primeiro e segundo estágio do *pipeline* são idênticos. A única diferença é que o projeto tem um *script* que além de realizar o build do *backend*, ele também realiza o build do *frontend* e cria uma nova versão do Projeto.

O terceiro estágio é uma verificação estática do código, para isso, foi conFigurado previamente um projeto na ferramenta Sonarqube, que realiza inspeções contínuas da qualidade do código. Após essa conFiguração, o Sonarqube gera de forma automática um *script* para ser executado pelo *pipeline*. O Sonarqube tem diversas métricas para qualidade do código que geralmente são escolhidas pela equipe de desenvolvedores e a equipe de teste. O projeto só passa para próximo estágio do *pipeline* se o resultado da análise realizada pelo Sonarqube estiver positivo, mas como se trata de um projeto

legado, é bastante complicado e custoso em relação a tempo de realizar as melhorias sugeridas pelo Sonarqube. O resultado dado pela análise do Sonarqube é ignorado, somente mostrando as melhorias sugeridas para os desenvolvedores ficarem atentos a não cometer más práticas de programação.

O último estágio do *pipeline* é o *deploy* automatizado, diferente do projeto piloto, que utilizava o Docker para seu *deploy*, nesse projeto é utilizada a ferramenta de automatização de tarefas, o Ansible automatizando as tarefas ou comandos utilizados pelo sistema operacional. O processo do *deploy* manual, era realizar o *build* do projeto, gerar a nova versão, realizar um acesso remoto ao servidor onde está alocado o projeto. Como o projeto está em um serviço do linux, então é preciso parar o serviço, copiar via SSH da máquina do desenvolvedor a nova versão do projeto para a máquina onde está o projeto, e por fim, iniciar novamente o serviço. Mas agora não há necessidade de realizar esse procedimento de forma manual, basta o projeto chegar até o estágio de *deploy* do *pipeline*, que será realizado com sucesso e automatizado o *deploy*. Como mencionado, esse estágio de deploy é apenas para as branches de desenvolvimento e homologação do repositório, pois não faz sentido realizar o *deploy* em branches que não são oficiais para a realização de testes ou homologação.

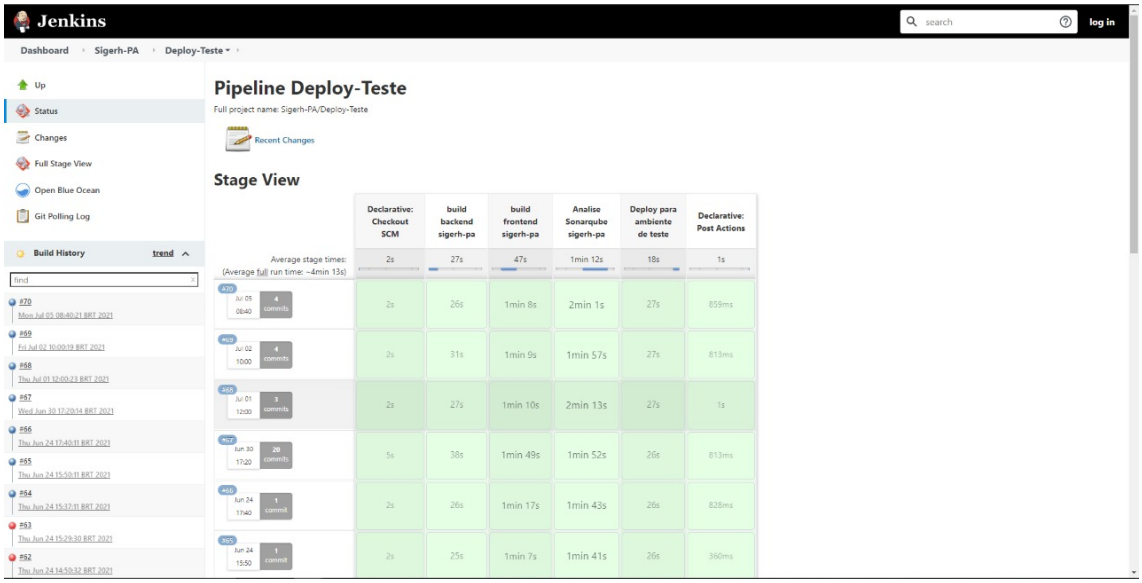
4.5 Replicação de DevOps em outros Projetos

O *pipeline* do projeto anterior, acabou se tornando a referência para a organização. Todos os *pipelines* posteriores foram desenvolvidos com o mesmo princípio, contendo em seu esqueleto, os estágios de *build*, análise estática e *deploy*. Entretanto, cada pipeline foi adaptado de acordo com a necessidade do projeto.

O próximo projeto que foi implementado a cultura DevOps, foi o Sistema de gerenciamento dos procedimentos gerais de recursos hídrico do estado do Pará, conhecido como SIGERH-PA. Seguindo o projeto anterior, temos os mesmos estágios do *pipeline*, Sendo que no estágio de construção, as tecnologias utilizadas são diferentes. Enquanto a Análise Automatizada utiliza Spring Boot, o SIGERH-PA utiliza dotnet framework para o *backend*, e framework de java*script* Angular para o *frontend*. Para a Análise Automatizada sua construção do *frontend* e do *backend* são um único estágio, agora no projeto do SIGERH-PA, são um estágio para cada, como é apresentado na Figura 4.3.

Após a implementação com sucesso do DevOps no Projeto SIGERH-PA, foi realizado também o projeto Sistema de Outorga de Água de Santa Catarina (SIOUT-SC), que é um clone do projeto SIGERH-PA.

Figura 4.3 – Pipeline do Projeto SIGERH-PA



Fonte: Elaborada pelo autor

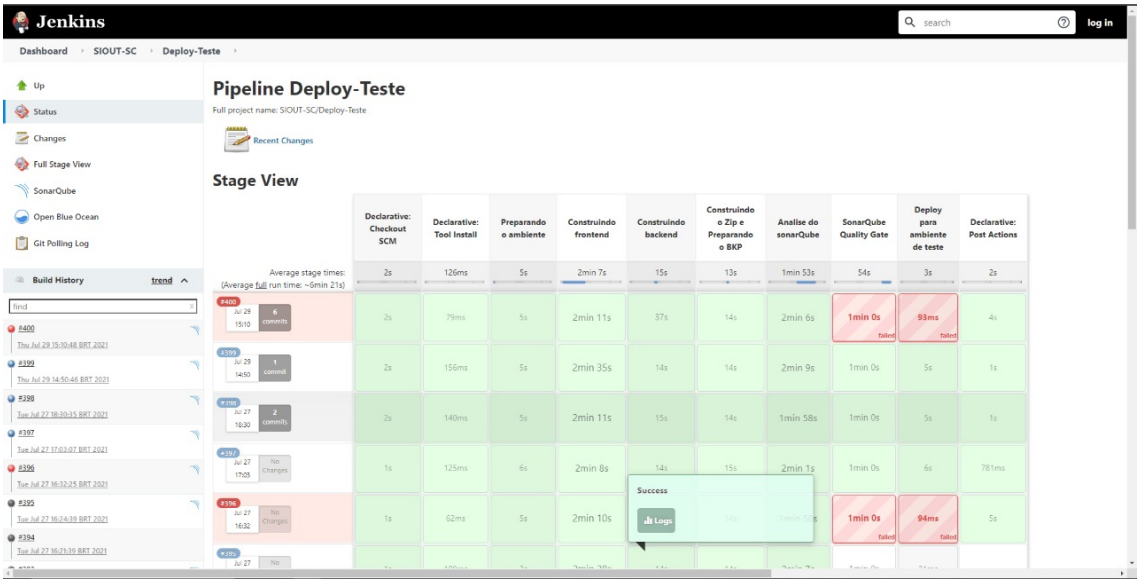
Porém, foram adicionados alguns estágios para o devido funcionamento de seu *pipeline*. Começando pela preparação do ambiente, que é responsável por baixar algumas dependências do projeto para sua construção. Após isso temos os estágios de construção do *backend* e do *frontend*. Neste projeto SIOUT-SC é necessário realizar backups, mas apenas quando o *pipeline* estiver executando nas branches de desenvolvimento ou homologação pois elas são as branches onde se realiza o *deploy*.

Por fim, temos o estágio de Sonarqube *Quality Gates*, como já mencionado no projeto anterior, depois que a análise estática do Sonarqube é realizada, caso o projeto esteja dentro das métricas definidas, ele poderá passar para o estágio de *deploy*. Se por acaso o projeto quebrar alguma métrica estabelecida pela Sonarqube, o estágio de Quality Gate saberá e irá abortar o *pipeline* falhando como é apresentado na Figura 4.4.

4.6 Dificuldades enfrentadas

No início do estágio foi complicado, pois quase ninguém sabia a função exercida pelo estagiário, mas depois de ver alguns resultados positivos, a organização começou a entender como funcionava a cultura DevOps e seus benefícios. Alguns meses depois, foi ministrado um Workshop para mais entendimento da área, destacando alguns pontos sobre a função exercida pelo estagiário no decorrer do estágio.

Figura 4.4 – Pipeline do Projeto SIOUT-SC



Fonte: Elaborada pelo autor

No decorrer do desenvolvimento dos *pipelines*, ocorreram algumas dificuldades, principalmente na parte de comunicação, pois o estágio foi em Home Office, e a organização só ter uma pessoa alocada para iniciar a cultura DevOps.

As maiores dificuldades foram saber quem procurar na hora de projetar o *pipeline*, pois era necessário saber como realizar o *build* do projeto, em qual servidor o projeto está alocado para o *deploy*, obter acesso às dependências do projeto, aos servidores e colaboradores do projeto. Por fim, houve uma certa resistência de uma equipe que trabalhava mais tempo em um projeto em querer aprender sobre a cultura DevOps, mas depois eles levaram em consideração os benefícios se aceitassem trabalhar de uma forma um pouco diferente do costume. Enfim, essa comunicação foi difícil no início, mas depois, com decorrer do estágio foi ficando tranquilo, as equipes foram se adaptando a essa mudança de cultura e cada vez mais entrando dentro da cultura DevOps.

O autor desse trabalho, está ciente que o DevOps não se limita apenas a Integração Contínua e Entrega Contínua. A cultura DevOps é algo extramente complexo de se aplicar dentro de uma organização, mas nesse estágio foi inserido a cultura Devops até nesse estágio de CI/CD.

5 CONCLUSÃO

Este documento descreveu a expeciência do discente João Paulo Barbosa Pena no estágio na Agência Zetta. O estágio teve duração de 9 meses. Neste período, ocorreram três grandes etapas realizadas pelo estagiário. Na primeira etapa foi realizado um treinamento pela plataforma da Alura, para o conhecimento das ferramentas utilizadas pela organização, com objetivo do estagiário aprender as principais ferramentas para o decorrer do estágio. Para a segunda etapa, ocorreu outro treinamento, mas voltado para área de DevOps. O estagiário realizou dois cursos, um teórico e outro prático com objetivo de aprender a cultura DevOps e poder replicar esse conhecimento adquirido em projetos na organização. A última fase teve como objetivo aplicar o conhecimento da cultura e ferramentas DevOps em projetos da organização.

O curso de Bacharelado em Ciência da Computação da Universidade Federal de Lavras provê uma base em infraestrutura que ajudou bastante o estagiário durante o período estágio, auxiliando o estagiário a entender assuntos como servidores Linux, Redes de Computadores, Sistemas Distribuídos, Gerência de Configuração de Software, pois os sistemas que o estagiário trabalhou são definidos como sistemas web, como por exemplo o Jenkins, que está alocado em um servidor, e aplica alguns comandos via SSH em outros servidores da empresa. Entender o funcionamento de protocolos de troca de mensagem e as especificações de um sistema distribuído em rede foram facilitadores para o desenvolvimento e aplicação da cultura DevOps em projetos durante o curso do estágio.

A maior divergência do bacharelado em contraste com o estágio é a cultura DevOps e suas ferramentas. O curso de Ciência da Computação deveria buscar meios de possibilitar aos alunos terem contato com novas tecnologias do mercado de trabalho e também conhecimento sobre a cultura e ferramentas de DevOps, com minicursos sobre algumas ferramentas mais atuais do mercado de trabalho, disciplinas mais focadas no mercado de trabalho. Pois, durante o periodo da graduação, em nenhuma disciplina foi citado algo relacionado as ferramentas ou cultura DevOps.

Após a aplicação da cultura DevOps dentro da organização, notou-se que, introduzir a cultura DevOps em uma organização que ainda não o possui, é algo extremamente difícil e complicado, pois precisa do apoio e incentivo, tanto da alta gerência quanto de seus funcionários. Cada organização tem seu ambiente, sua metodologia de trabalho, e assim o DevOps deve se adaptar e flexibilizar à organização.

O estágio proporcionou, muitas oportunidades de vivenciar na prática conteúdos que foram abordados na graduação, possibilitando a obtenção de conhecimentos relacionados à área de DevOps. O conhecimento técnico adquirido pode ser aplicado tanto para a vida pessoal, quanto para a vivência acadêmica e profissional. Também, obteve-se crescimento com relação à organização, proatividade e trabalho em equipe. As lições aprendidas no estágio podem ser aplicadas na vida de uma maneira geral.

REFERÊNCIAS

ANSIBLE. **Documentação Ansible**. 2021. Disponível em: <https://docs.ansible.com/ansible/latest/index.html>. Acesso em: 4 out. 2021.

ATLASSIAN. **Integração contínua**. 2021. Disponível em: <https://www.atlassian.com/br/agile/software-development/continuous-integration>. Acesso em: 13 set. 2021.

AWS. **O que significa integração contínua??** 2021. Disponível em: <https://aws.amazon.com/pt/devops/continuous-integration/>. Acesso em: 13 set. 2021.

AWS. **O que é o DevOps?** 2021. Disponível em: <https://aws.amazon.com/pt/devops/what-is-devops/>. Acesso em: 17 ago. 2021.

AZURE. **Entrega contínua vs. implantação contínua**. 2021. Disponível em: <https://azure.microsoft.com/pt-br/overview/continuous-delivery-vs-continuous-deployment/>. Acesso em: 18 set. 2021.

DBACORP. **CI/CD - Entrega Contínua: conheça tudo a respeito**. 2021. Disponível em: <https://blog.dbacorp.com.br/2020/05/29/cicd-entrega-continua-conheca-tudo-a-respeito/>. Acesso em: 18 set. 2021.

EBERT, C. et al. Devops. **IEE3**, v. 1, n. 1, p. 94–100, 2016.

GAEA. **Benefícios do DevOps conheça os 5 principais**. 2021. Disponível em: <https://gaea.com.br/beneficios-do-devops-conheca-os-5-principais/>. Acesso em: 08 setembro. 2021.

JENKINS. **O que é Jenkins?** 2021. Disponível em: <https://www.cloudbees.com/jenkins/what-is-jenkins>. Acesso em: 21 set. 2021.

MAUDA. **Introdução ao SonarQube**. 2021. Disponível em: <http://www.mauda.com.br/?p=2248>. Acesso em: 13 nov. 2021.

PRONUS. **O que é Gerência de Configuração de Software?** 2021. Disponível em: <https://blog.pronus.io/posts/controle-de-versao/o-que-eh-gerencia-de-configuracao-de-software/>. Acesso em: 22 set. 2021.

REDHAT. **O que é um pipeline de CI/CD?** 2021. Disponível em: <https://www.redhat.com/pt-br/topics/devops/what-cicd-pipeline>. Acesso em: 22 set. 2021.

SONARQUBE. **Documentação SonarQube**. 2021. Disponível em: <https://docs.sonarqube.org/latest/>. Acesso em: 30 set. 2021.

WIKIPIDIA. **Gerência de configuração de software**. 2021. Disponível em: <https://pt.wikipedia.org/wiki/Ger%C3%AAncia_de_configura%C3%A7%C3%A3o_de_software>. Acesso em: 22 set. 2021.

ZETTA. **SOMOS ZETTA**. 2021. Disponível em: <https://agenciazetta.ufla.br/institucional/>. Acesso em: 22 set. 2021.