UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
MBA EM GESTÃO DA TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

PEDRO HENRIQUE DE SOUZA FREITAS

PRÁTICAS DE DEVOPS NA MELHORIA DA QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2018

PEDRO HENRIQUE DE SOUZA FREITAS

PRÁTICAS DE DEVOPS NA MELHORIA DA QUALIDADE NO DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada como requisito parcial à obtenção do título de Especialista em Gestão da Tecnologia da Informação e Comunicação, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná

Orientador: Prof. MSc. Luiz Pinheiro Junior

CURITIBA

2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
IV CURSO DE ESPECIALIZAÇÃO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO

TERMO DE APROVAÇÃO

Práticas de *DevOps* na Melhoria da Qualidade no Desenvolvimento de Software

Por – **Pedro Henrique de Souza Freitas**

Esta monografia foi apresentada às **20h30** do dia **22/11/2018** como requisito parcial para a obtenção do título de Especialista no CURSO DE ESPECIALIZAÇÃO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO, da Universidade Tecnológica Federal do Paraná, **Câmpus Curitiba**. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho:

1		Aprovado
2		Aprovado condicionado às correções Pós-banca, postagem da tarefa e liberação do Orientador.
3		Reprovado

_____
Prof. Msc. Alexandre Jorge Miziara
UTFPR - Examinador

_____
Prof. Msc. Luiz Pinheiro Junior
UTFPR – Orientador

_____
Prof. Msc. Alexandre Jorge Miziara
UTFPR – Coordenador do Curso

Obs.: O termo original encontra-se assinado na coordenação do curso.

AGRADECIMENTOS

Agradeço a Deus por ter me guiado na realização do curso. Agradeço aos professores por terem disseminado o conhecimento durante o decorrer do curso, com dedicação, paciência e amor.

Agradeço aos meus familiares pelo apoio.

RESUMO

O objetivo da presente pesquisa é explorar como as práticas de DevOps estão auxiliando na melhoria da qualidade no desenvolvimento de software. Este objetivo destaca-se nos benéficos que a adoção da filosofia DevOps traz na gestão de ciclo de vida de um software. Este objetivo se alcança através do entendimento dos processos de desenvolvimento de software e da análise dos processos que possuem um potencial de adoção de práticas DevOps. Para isso foi realizada uma revisão da literatura sobre os processos de Desenvolvimento de Software com ênfase em metodologias ágeis, práticas DevOps dentro de uma abordagem filosófica e cultura, e a qualidade de software. Foi utilizado um estudo de caso comparando dois sistemas de software, apresentado os benéficos que se obteve com a adoção das práticas DevOps utilizando ferramentas de automatização de processos de implantação e testes. Com as informações levantadas, os critérios analisados foram a quantidade de falhas no decorrer do desenvolvimento do software, a velocidade de execução de testes e massa, e a velocidade de implantação e disponibilização do software para as partes interessadas. Conclui-se que as práticas de DevOps utilizadas na melhoria da qualidade do desenvolvimento do software apresentam uma melhoria relativa nas atividades repetitivas que fazem parte do ciclo de desenvolvimento, trazendo redução de custo e otimização de entrega e testes. Como sugestões de estudos futuros recomenda-se verificar como a adoção das práticas de DevOps impactam na cultura da empresa e na redução dos postos de trabalho.

PALAVRAS-CHAVE: *DevOps*; Metodologias Ágeis; Qualidade de Software.

ABSTRACT

The purpose of this research is to explore how DevOps practices are helping to improve quality in software development. This goal stands out in the benefits that the adoption of the DevOps philosophy brings in the software life cycle management. This goal is achieved through the understanding of software development processes and the analysis of processes that have a potential to adopt DevOps practices. For this, a review of the literature on Software Development processes with emphasis on agile methodologies, DevOps practices within a philosophical approach and culture, and the quality of software was carried out. We used a case study comparing two software systems, presented the benefits obtained with the adoption of DevOps practices using automation tools for the deployment and testing processes. With the information collected, the criteria analyzed were the number of failures during software development, the speed of execution of tests and mass, and the speed of deployment and availability of the software for the interested parties. It is concluded that the DevOps practices used to improve the quality of software development show a relative improvement in the repetitive activities that are part of the development cycle, bringing cost reduction and optimization of delivery and testing. As suggestions for future studies it is recommended to check how the adoption of DevOps practices impact on company culture and the reduction of jobs.

KEYWORDS: DevOps; Agile Methods; Software Quality.

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
BDD	Behavioral Driven Development
CMM	Modelo de Maturidade em Capacitação
DevOps	Desenvolvimento e Operações
HTTP	Hypertext Transfer Protocol
IES	Instituição de Ensino Superior
ISO	Organização Internacional de Normalização
NBR	Norma Regulamentadora
P.O.	Product Owner
PBQP	Programa Brasileiro de Qualidade e Produtividade em Software
T.I.	Tecnologia da Informação
TDD	Test Driven Development
XP	Extreme Programming

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A demanda pelo desenvolvimento de soluções de *softwares* mais rápidas e com qualidade vem se tornando cada vez maior pelo mercado. Com isso, a busca de uma melhoria continua dentro dos processos de desenvolvimento e entrega de soluções, vem se tornando mais presente no ambiente de tecnologia da informação.

A tecnologia da informação se faz presente no âmbito da estratégia das empresas pela sua capacidade de trazer benefícios ao produto final e aos processos dentro do ambiente corporativo. Essa presença vem se tornando cada vez maior pois as empresas estão tratando a tecnologia da informação, como um grande diferencial estratégico para sua competição dentro do mercado, e para isso, a entrega de soluções precisam ter mais velocidade e assertividade.

Com essa demanda de soluções de *software* mais assertivas e com mais qualidade que as empresas precisam fornecer, dentro dos processos da tecnologia da informação, surge uma nova filosofia, com o propósito de aumentar a velocidade das entregas de novas soluções para o mercado. Essa filosofia quebra um paradigma pois visa incentivar a junção dos setores de desenvolvimento de *software* e as operações que os mantém.

Baseado nessa necessidade de melhora do ambiente de desenvolvimento de soluções e também, visando elaborar um material acadêmico com dados coletados através de pesquisa de diferentes fontes e um estudo de caso, este trabalho tem como tema principal uma análise sobre as práticas de desenvolvimento e operações, na melhoria da qualidade do desenvolvimento de *software*.

Considerou-se a utilização de metodologias ágeis e a automatização de alguns processos de determinadas fases do ciclo de desenvolvimento de *software*, como forma de redução de tempo de desenvolvimento e de melhoria na entrega das soluções de *software*.

Dentro de uma abordagem mais ampla, o entendimento da qualidade de *software* em diversos aspectos, se faz presente neste estudo a fim de apresentar como esses pontos são essências para a melhoria da qualidade de um *software*.

1.1 Problema de Pesquisa

Com o movimento de *software* ágil, implementado para agilizar a aumentar a assertividade da entrega de *software*, surgiu um desequilíbrio entre o desenvolvimento de software e as operações que os mantém, surgindo o seguimento *DevOps*.

O *DevOps* é uma filosofia que vem sendo trabalhada nas empresas a fim de apresentar uma nova forma de se desenvolver e gerenciar *softwares*. Dentro dessa filosofia, existe a gestão da qualidade do *software*, de uma maneira automatizada e otimizada.

Desta forma, o presente estudo tem a proposta de responder à seguinte pergunta: ***Como as práticas do DevOps podem contribuir na qualidade do desenvolvimento de Software?***

Para responder esta pergunta de pesquisa, o trabalho contém os objetivos gerais e específicos a seguir.

1.1.1 Objetivo Geral

O objetivo geral do presente trabalho é verificar como as práticas de *DevOps* podem contribuir na qualidade do desenvolvimento de *software.*

1.1.2 Objetivos Específicos

Com o intuito de atingir o objetivo geral deste trabalho, destacam-se os seguintes objetivos específicos:

1. Identificar práticas de *DevOps* no desenvolvimento de *software*;
2. Verificar se as práticas de *DevOps* propiciam qualidade no desenvolvimento de *software*;
3. Explicar como as práticas de *DevOps* propiciam qualidade no desenvolvimento de *software*.

2 REFERENCIAL TEÓRICO

Este trabalho está divido em cinco capítulos, os quais contém as seguintes abordagens descritas a seguir.

No primeiro capitulo é feita a introdução do conteúdo, com o problema de pesquisa e seus objetivos gerais e específicos.

No segundo capitulo é apresentado o embasamento teórico obtido em realização ao Desenvolvimento de *software*, o DevOps e a qualidade de *software.* No âmbito do desenvolvimento de software é apresentando os conceitos das metodologias ágeis e suas aplicações com o Scrum e o Extreme Programming. Na sessão do DevOps são abordadas as práticas, o seu surgimento e seus princípios. Na sessão sobre a qualidade de software, é abordo o seu conceito, o custo, o controle e os testes automatizados.

No terceiro capitulo é apresentado a metodologia de pesquisa utilizada para a realização do trabalho.

No quarto capitulo é apresentado a analise de dados realizada com em uma empresa que se utiliza as práticas *DevOps*. Nesta análise é possível entender como os princípios do *DevOps* estão implementados e quais são os benefícios desta implementação em relação a qualidade do software.

Por último, no quinto capítulo encerra-se o trabalho através de uma conclusão que cita os benefícios da utilização das práticas DevOps.

2.1 Desenvolvimento de *Software*

Existem inúmeras metodologias no mercado para se utilizar no desenvolvimento de um software. Como parte dessas metodologias, o DevOps surge a fim de otimizar e melhorar o desenvolvimento de um software. Deste modo, é preciso entender como os processos de desenvolvimento de software e suas metodologias, suportam a inclusão do DevOps em seus processos.

Os processos de desenvolvimento de software vêm sofrendo alterações e melhorias ao longo da história. Com a presente dependência dos softwares na estratégia competitiva das empresas, é necessário ter uma atenção especial aos processos que podem melhorar e otimizar o desenvolvimento de um software.

A definição de um processo de desenvolvimento de software, Segundo (PRESSMAN, 2011), é um conjunto de atividades, ações e tarefas realizadas para a criação de uma tarefa de trabalho. Se tratando de desenvolvimento de software, esta abordagem muda a forma de se ver estes processos, trazendo a responsabilidade de sua escolha para a equipe de desenvolvimento do software. Esta atribuição visa a entrega do software e a qualidade suficiente para satisfazer a necessidade daqueles que o patrocinaram.

Uma metodologia de processo de desenvolvimento de software (PRESSMAN, 2011), estabelece uma base para os processos e pequenas atividades no desenvolvimento do software. Uma metodologia genérica necessita de cinco atividades: Comunicação; Planejamento; Modelagem; Construção; Emprego.

Com este conjunto de atividades, podemos apontar como sendo de um modelo clássico de desenvolvimento, uma vez que cada etapa só começa apenas depois que a etapa anterior for finalizada. Considerada como modelo cascata, este modelo é inflexível pois restringem a divisão do projeto em fases distintas.

2.1.1 Ciclo do Desenvolvimento de Software

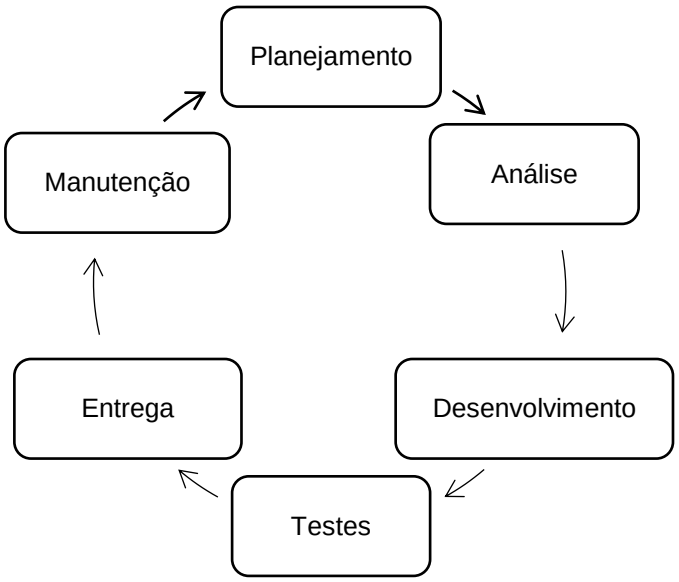
O desenvolvimento de um software é composto por várias etapas. É essencial ter isso mapeado pois existe algumas etapas onde a equipe de desenvolvimento atua e outras onde as operações atua.

Em um cenário comum, conforme apresentado na figura 1, o desenvolvimento de um software possuí 6 etapas. Cada etapa possuí o seu responsável e a sua equipe de execução. Para se executar essas etapas, trabalhamos com alguma metodologia de desenvolvimento de software, que faz com que as etapas fluam de uma maneira orquestrada.

A etapa de planejamento é a primeira faze do ciclo e é responsável por planejar as pessoas e datas para a realização do Software. A parte de análise é a segunda fase trabalhada e é responsável pelo detalhamento das funcionalidades. A próxima fase é responsável pelo o desenvolvimento do software. Após o desenvolvimento é iniciado a fase de testes de software, e esta fase pode ser executada paralelamente ao desenvolvimento do software. Após a finalização o desenvolvimento e testes é feito

a entrega do software para as partes interessadas. A fase de manutenção do sistema finaliza o ciclo de desenvolvimento de software.

Figura 1 - Ciclo de Desenvolvimento de um Software



Fonte: Autoria Própria

2.1.2 Metodologia Ágil

Com base no Manifesto Ágil (BECK; BEEDLE; BENNEKUM, 2018), foi identificado uma outra maneira de se exergar o desenvolvimento de software, beseado na Valorazação dos seguintes pontos:

- **Indivíduos e interações** mais que processos e ferramentas.
- **Software em funcionamento** mais que documentação abrangente.
- **Colaboração com o cliente** mais que negociação de contratos.
- **Responder a mudanças** mais que seguir um plano.

Para atender esses pontos declarados dentro do manifesto ágil, surgiu alguns frameworks de desenvolvimento e gestão de software. Será descrito na próxima seção, os frameworks Scrum e Extreme Programming.

2.1.2.1 SCRUM

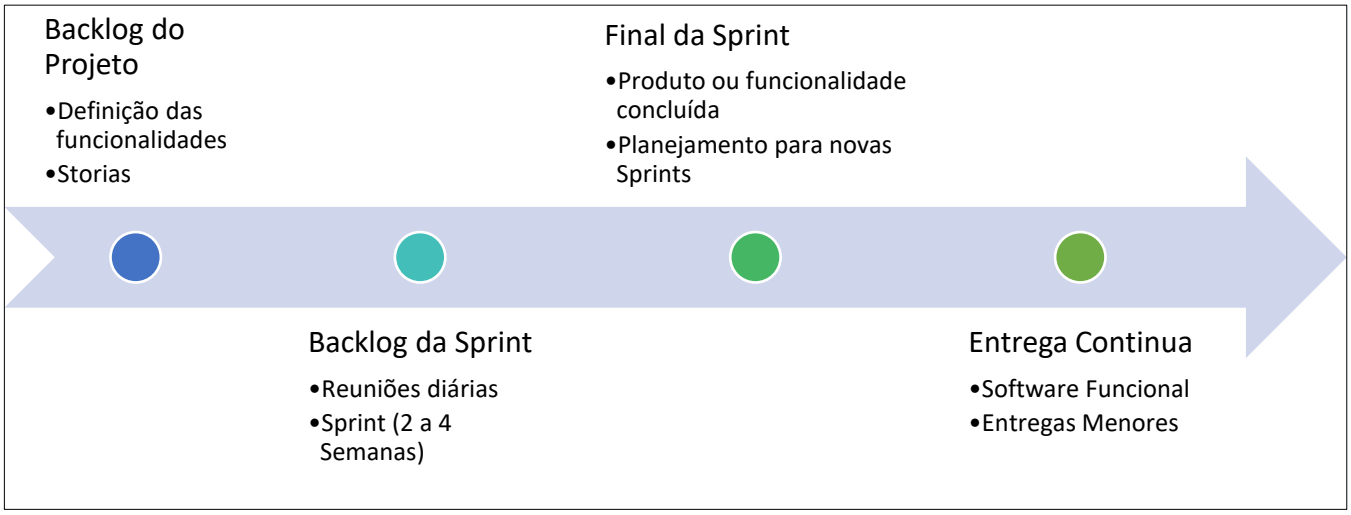
O SCRUM, segundo (SCHWABER; SUTHERLAND, 2013), é uma framework aplicado no desenvolvimento dos softwares, utilizado para gerenciar fluxos de trabalhos na criação de produtos que surgir no final dos anos 80.

O SCRUM, ainda (SCHWABER; SUTHERLAND, 2013), aplica vários processos e técnicas que visa gerenciar e melhorar o desenvolvimento do software, de forma interativa e incremental. Trazendo uma melhoria continua no decorrer do desenvolvimento do produto.

O modelo de trabalho do scrum se baseia em times pequenos, que são auto gerenciáveis e que atendem os pilares do SCRUM, que são eles: Transparência, Inspeção e adaptação (SCHWABER; SUTHERLAND, 2013).

Na figura 2, é apresentado as etapas essenciais do scrum com algumas características.

Figura 2 - Fluxo do Scrum



Fonte: Autoria própria

2.1.2.2 *EXTREME PROGRAMMING*

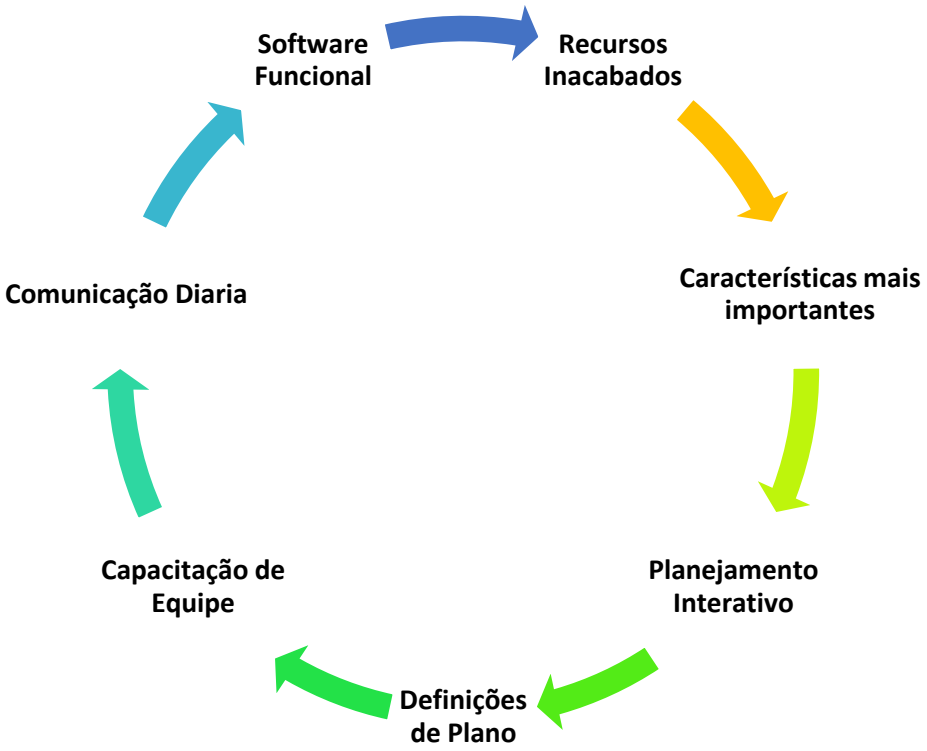
O *Extreme Programming* (XP) é um processo de desenvolvimento de *software* que implementa o desenvolvimento ágil.

O XP é recomendado, segundo (TELES, 2004), nos cenários de desenvolvimento de *software*, onde os requisitos não são bem definidos e mudam com frequência, utiliza-se da orientação objeto como modelo de desenvolvimento do

software, as equipes de desenvolvimento de *software* são pequenas, limitadas a até 12 desenvolvedores e o desenvolvimento é incremental, ou seja, desde o começo do projeto o sistema já é implementado e vai ganhando novas *features*.

Para que este modelo seja implementado, a empresa precisar possuir alguns valores, que são eles: *Feedbacks*, Comunicação, Simplicidade e Coragem. Esses valores são a essência para a adoção deste modelo de desenvolvimento de *software*.

Figura 3 - Fluxo do Extreme Programming



Fonte: Adaptado de EXTREME PROGRAMING (2013)

2.2 *DevOps* – Desenvolvimento e Operações

Com a necessidade cada vez maior de se desenvolver soluções de *software* com mais rapidez, para que atendam as demandas do mercado o mais rápido possível, as metodologias ágeis foram se tornando a principal metodologia utilizada no desenvolvimento de *software*.

No desenvolvimento ágil, a filosofia que se prega é que os desenvolvedores e clientes se unam, para que a fluidez do desenvolvimento do *software* seja voltada para a real necessidade do cliente. Com isso, o software começa a se tornar um produto que o cliente consiga acompanhar e moldar, com o decorrer do desenvolvimento. Nos modelos de desenvolvimentos tradicionais isso não é possível, uma vez que os

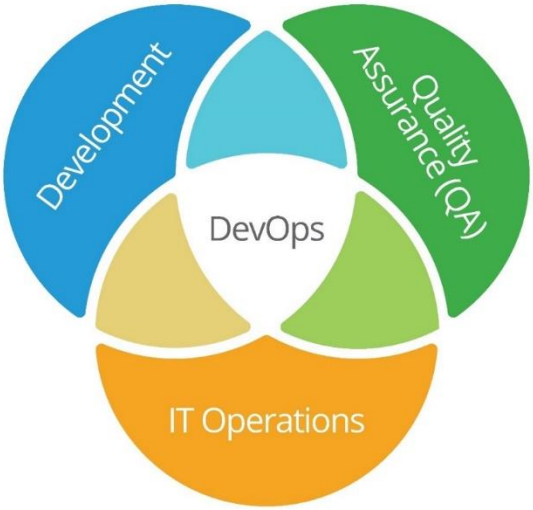
softwares eram desenvolvidos a partir de especificações longas e as entregas eram que enormes e inúmeras funcionalidades.

Com a adoção do desenvolvimento ágil, as empresas e equipes de T.I. começaram a ter um problema com o ciclo de vida do *software*. A partir do momento que as iterações entre as equipes de desenvolvimento e os clientes começam aumentar, as entregas precisaram seguir no mesmo ritmo. Neste momento, começou a surgir o *DevOps*.

DevOps é um termo referente a combinação entre “Desenvolvimento e Operações”, onde o objetivo é unificar as partes no desenvolvimento de um *software* (ZENTGRAF, 2012). Para que o *DevOps* seja adotado, é preciso uma mudança cultura na organização, a fim de buscar um trabalho cooperativo entre as áreas de negócios, operações e desenvolvimento.

Na figura 4 é apresentado uma abordagem onde o *DevOps* é o centro de um universo onde se encontra Desenvolvedores, Garantia da Qualidade e Operações de Tecnologia da Informação.

Figura 4 - Abordagem DevOps



Fonte: (SMARTSHEET, 2018)

2.2.1 Surgimento do DevOps

O termo *DevOps* surgiu no ano de 2009, quando Patrick Debois organizou uma conferência denominada *DevOpsDays*, em Gent, na Bélgica. Nesta conferência foi apresentada um conjunto de práticas que visavam aumentar a cooperação entre as esquipes de desenvolvimento e negócios (RATO, 2017).

Anualmente este termo vem sendo apresentado e debatido em diversos eventos. A página web www.devopsdays.org centraliza as series de eventos a nível mundial, trazendo tópicos sobre desenvolvimento de *software*, operações de infraestrutura de T.I. e como essas duas abordagens se relacionam.

2.2.2 Praticas DevOps

Com o surgimento do termo *DevOps* e os cenários corporativos ficando cada vez mais competitivos, o mercado começou a adotar alguns processos que visam otimizar os ambientes de desenvolvimento de *software*.

O DevOps começou a ser associados a alguns termos nestes ambientes de desenvolvimento, que são eles: Integração Continua (*Continuos Integration*), Implementação Continua (*Continuos Deploymeny*) e Entregas Contínuas (*Continuos Delivery*) (RATO, 2017). Estes itens são como uma base para identificar o DevOps nos ambientes de desenvolvimento, uma vez que o *DevOps* é uma forma de integrar processos e pessoas dentro destes ambientes.

Com a implementação destes 3 termos, é possível colocar em pratica a integração e compromisso das partes envolvidas nos processos de desenvolvimento. Desta maneira, é possível atingir os objetivos de entrega do *software* com mais assertividade e velocidade.

2.2.2.1 INTEGRAÇÃO CONTINUA

A integração contínua se resume em uma prática de integração entre o código desenvolvido e testes. Essa prática pode, muitas vezes, ter o intuito de concretizar uma outra prática, a de entrega continua (RATO, 2017).

O processo que envolve essa prática funciona com a junção de códigos de *software* que são desenvolvidos, com uma frequência alta, chegando a ter várias junções no decorrer de um dia de desenvolvimento. Essas junções, mais tarde, podem se tornar um entregável do *software*, de acordo com o *feedback* da equipe de desenvolvimento do *software* (RATO, 2017).

2.2.2.2 ENTREGA CONTINUA

Uma prática muito importante no ciclo de vida de um *software* é a entrega continua. Essa prática é fundamental no acompanhamento do crescimento de um *software*. Para Sharma e Coyne (SHARMA; COYNE, 2015), esta prática viabiliza que as funcionalidade desenvolvidos no *software*, estejam disponíveis para os usuários o mais rápido possível.

Está prática, tem uma importância especial pois se deu origem ao movimento *DevOps*. Isso ocorreu, pois, a integração continua possibilita uma evolução na automatização dos processos de *software*, fornecendo a possibilidade de uma implantação automatizada de um *software*. É fundamental nesta prática, implementar os testes automatizados, pois isso leva a uma detecção mais rápida de problemas. (LEVITA; NETO, 2018).

2.2.2.3 IMPLEMENTAÇÃO CONTÍNUA

No final do desenvolvimento do *software*, começa a fase de implementação e entrega. Com o modelo automatizado, a implementação continua permite que todo o desenvolvimento do *software*, após ter passado pelos processos de testes automatizados, são automaticamente enviados ao sistema de produção, sendo assim, finalizando o ciclo do desenvolvimento do *software* (HUMBLE; FARLEY, 2015).

Esta prática se repete ao longo do ciclo de vida de um software, pois é a última etapa que a equipe de desenvolvimento fica responsável. Também é possível considerar que ao final deste processo, o cliente pode ver um valor agregado no software, pois é nesse momento que ele começa a se tornar útil para o cliente.

2.2.2.4 MONITORAMENTO CONTINUO

A prática de Monitoramento Continuo é de grande auxilio no acompanhamento da evolução de um sistema. Esta prática permite que métricas do desempenho e desenvolvimento do software estejam disponíveis aos envolvidos após durante e após o desenvolvimento do software. Com isso, é possível ter uma visão clara de como está o progresso do projeto, podendo fornecer insumos para entender o desempenho dos processos e auxiliar na tomada de decisão em eventuais gargalos (LEVITA; NETO, 2018).

2.2.3 Princípios do DevOps

Segundo o projeto *The Phoenix Project* (KIM; BEHR; SPAFFORD, 2013), existem 3 princípios para a adoção do DevOps dentro de uma metodologia de desenvolvimento de software.

O primeiro princípio visa criar um modelo de trabalho otimizado que funcione com o time de desenvolvimento e operações. Para isso, pode se pensar que os processos podem ser quebrados em pequenas partes, para otimizar aos poucos um pacote global do fluxo de trabalho, prezando o controle e segurança das mudanças que podem ocorrer durante o trabalho.

O segundo princípio é a doção de uma cultura de *feedbacks* em forma de *loops*, ou seja, contínuos. Estes modelos de *feedback* visam corrigir a fonte de desenvolvimento, evitando problemas e retrabalho. Estes *loops* também criam a qualidade da origem do processo, nas partes que estão atuando constantemente no desenvolvimento e nas operações.

O terceiro princípio visa trabalhar com uma cultura que estimula a experimentação e a mentalidade de que a repetição de processos é essencial para trabalhar se ter o domínio. Esta cultura estimula o fazer e não a análise aprofundada.

2.2.4 DevOps e a otimização do desenvolvimento do software

Dentro das etapas do ciclo de desenvolvimento de *software*, existem 3 etapas com a integração com o *DevOps* pode interagir, a etapa de desenvolvimento, testes e entrega.

Nestas etapas, o *DevOps* está associado a automatização de alguns processos que são corriqueiros. Existe várias ferramentas que implementa a automatização nessas etapas. Duas ferramentas bem conhecidas no mercado são utilizadas para implementar essas automatizações:

1. Microsoft Team Foudation Server
2. Jenkins

Essas ferramentas, em poder da equipe de operações, traz a possibilidade da implementação das práticas de *DevOps*.

2.3 Qualidade de Desenvolvimento de Software

A qualidade de desenvolvimento de um *software* é um assunto que vem se tornando cada vez mais um importante dentro dos processos de desenvolvimento de um *software*. A assertividade das entregas de um produto de *software* e de suas implementações, são fundamentais para tornar a experiência dos usuários mais agradáveis.

Duas abordagens de qualidade relacionadas à *software* estão presentes em dois momentos: na qualidade do desenvolvimento do *software* e na qualidade do produto final. Para atingir esses dois modelos existem alguns processos que precisam ser adotados no decorrer do desenvolvimento.

A norma ISO 9000 apresenta a gestão da qualidade e auxilia na implementação de um modelo de qualidade aceito dentro do mercado Brasileiro. Estas diretrizes abordam a qualidade de software.

As normas ISO 9001, 9002 e CMM são os modelos principais que devem ser adotados para atingir as recomendações de qualidade dos processos que envolvem o desenvolvimento de um *software* (RINCONM, 2006).

Para a implementação da qualidade dentro desses processos, é possível elencar alguns pontos de atenção (BUENO, 2006):

- Definição da qualidade
- Controle de qualidade
- Garantia de qualidade
- Custo da qualidade

2.3.1 Definição de Qualidade de Software

Segundo a Norma ISO 9000, os clientes exigem que as características dos seus produtos, satisfação as suas necessidades e expectativas. Isso é possível quando o produto atende os requisitos previamente estabelecidos pelos clientes (ABNT, 2005).

Partindo deste princípio, o software pode ser um produto que precisa atender as expectavas do cliente, e quando isto acontece, o software já adquiriu um grau de qualidade.

2.3.2 Controle da Qualidade

O controle da qualidade no decorrer do desenvolvimento envolve a integração de várias ações para garantir que os padrões e os procedimentos estão sendo seguidos (RODRIGUES, 2015).

Para realizar este controle, são realizadosrevisões, inspeções e testes continuamente enquanto um software é desenvolvimento. É possível realizar o controle da qualidade com o auxílio do cliente e das documentações de regras do negócio. Tudo o que não estiver dentro dos critérios de qualidade que o cliente espera, deve ser tomado como um ponto de atenção para quem gerencia o projeto (RODRIGUES, 2015).

O foco do controle da qualidade é garantir um software estável na entrega final do projeto. Uma entrega com qualidade pode garantir com que o custo do projeto não fique fora do previsto, uma vez que a correção de problemas após a finalização de um software pode gerar prejuízo.

2.3.3 Custo da Qualidade

Um software possuí uma característica muito importante que deve ser levado em consideração na hora de planejar e controlar o custo da sua qualidade. É praticamente impossível entregar um software em perfeitas condições, ou seja, sem bugs (RODRIGUES, 2015).

É comum um software ter problemas, então, é preciso definir o grau de qualidade que o software precisa atingir para satisfazer as necessidades e expectativas do cliente.

Os custos que envolvem a qualidade de software podem ser categorizados entre custo da falha, custo da prevenção e custo da análise (RODRIGUES, 2015).

O custo da falha é mais difícil de prever pois depende muito do tamanho da falha e o que esta falha significa para os usuários do software. O custo da prevenção é um

orçamento que deve estar prevendo as ações para garantir a qualidade de um software. O custo da análise está atrelado a garantia da qualidade do software, uma vez que depende das ações tomadas do decorrer do desenvolvimento do software, como os testes para a identificação de erros (RODRIGUES, 2015).

2.3.4 Qualidade do Produto de Software

Para entender e medir a qualidade do produto de software, segundo o programa Brasileiro de Qualidade e Produtividade em Software (CRUZ; ANDRADE; FIGUEIREDO, 2010), é preciso passar por alguns critérios de avaliação para definir o grau de qualidade de produto.

Com o incentivo do Ministério da Ciência e Tecnologia e a Secretaria de Política de Informática, O PBQP (Programa Brasileiro de Qualidade e Produtividade em Software) de software promove projetos com o foco melhoria dos processos, projetos e serviços de software. Para entender a qualidade dos produtos, o PBQP de software possuí critérios de avaliação, com seus conjuntos de atributos e pesos. Os critérios são (CRUZ; ANDRADE; FIGUEIREDO, 2010):

- Relevância
- Impacto
- Abrangência
- Inovação
- Qualidade da Apresentação do Artigo
- Resultados Obtidos pelo Projeto

Com a análise de cada critério, levando em consideração seus pesos, é possível medir o grau de qualidade que o produto se encaixa, em relação ao programa. Vale salientar que a avaliação da qualidade dos produtos de software seguem as series NBR ISO/IEC 9126 (Tecnologia de Informação – Avaliação de Produto de Software – Características de qualidade e diretrizes para o seu uso) e NBR ISSO/IEC 14598 (Tecnologia de informação - Pacotes de software - Testes e requisitos de qualidade), que estão sendo substituídas pelas normas da série NBR ISO/IEC 25000 - Engenharia de Software - Requisitos e avaliação da qualidade de produtos de software (SQuaRE) (CRUZ; ANDRADE; FIGUEIREDO, 2010).

2.3.5 Testes Automatizados

Uma característica utilizada no modelo de desenvolvimento ágil, são as entregas continuas. Estas entregas surgem com o intuíto de o software ser desenhado em pequenas fatias e entregas menores, para atender o negócio a medida com que as mudanças no mercado vão ocorrendo.

Neste modelo de entrega continua, se encaixa a automatização de testes. As práticas de automatização de testes são bem aceitas no modelo ágil (LIMA; DANTAS; VASCONCELOS, 2012).

Os testes automatizados são programas ou scripts, que são desenvolvidos junto ao software. Eles são utilizados para exercitar as funcionalidades do sistema, de maneira rápida. Uma vantagem de utilizar este modelo é poder passar por todos os casos de testes previamente estabelecidos (LIMA; DANTAS; VASCONCELOS, 2012). A repetição destes testes, se for aplicada por humanos, pode levar ao erro, mas quanto executada por uma ferramenta de automatização, é possível inibir estes erros e encontra-los de forma mais rápida. Outro ponto relevante em utilizar este modelo, é a garantia da qualidade na manutenção dos *softwares*, uma vez que ele já possuiu todos os casos de testes automatizados.

A garantia das entregas com qualidade, que é resultado de uma automatização de testes, é uma pratica do DevOps que se encaixa dentro da entrega continua.

Conforme descrito na ISO 9000, o estabelecimento de métodos para medir a gestão da qualidade, os recursos necessários para a garantia da qualidade e a definição dos processos e responsabilidades para atendem a garantia da qualidade (ABNT, 2005), são premissas para a entrada de uma automatização de testes.

A junção dos testes e o desenvolvimento de software, pode ser feito de várias formas. Duas abordagens deste cenário é o *Test Driven Development* (TDD) and *Behavioral Driven Development* (BDD). Estes modelos são uma tendência dentro de um desenvolvimento ágil com o conceito de DevOps (ERICH; AMRIT; DANEVA, 2014).

2.3.5.1 DESENVOLVIMENTO ORIENTADO A TESTES (TDD)

Com a adoção do desenvolvimento ágil, mais especificamente o método *Extreme Programming*, surgiu uma metodologia de desenvolvimento de software, assim como existe o desenvolvimento orientado a objetos. Esta metodologia é denominada *Test-Driven Development* (Desenvolvimento Orientado a Objetos)

Esta forma de desenvolvimento consiste em desenvolver os testes automatizados antes do código funcional (JANZEN; SAIEDIAN, 2005). Todo o desenvolvimento do software é feito para satisfazer os testes automatizados escritos. Com este modelo, a qualidade de *software* é mais assertiva e a garantia do desenvolvimento dos testes automatizados é mais garantida.

2.3.5.2 DESENVOLVIMENTO ORIENTADO A COMPORTAMENTO (BDD)

O TDD é uma maneira eficiente de realizar os testes de um *software*, porém, ele possuí alguns problemas no decorrer de sua implementação. Uma vez que o responsável por desenvolver estes testes são os desenvolvedores do *software*, é complicado abstrair a essência de uma funcionalidade para realizar a aplicação deste modelo.

Com os problemas enfrentados com o TDD, o *Behavior Driven Development* (BDD) surgiu para resolve-los. O BDD é uma maneira de aplicar os testes de aceitação de um *software*, de uma maneira simples, a partir de um ponto de partida e um resultado final. Este modelo busca fazer com que todos os membros do projeto consiga ter um entendimento comum dos testes que vão ocorrer no software, através de testes de software escritos em linguagem natural. (SOEKEN; WILLE; DRECHSLER, 2012).

O BDD funciona através de cenários. Estes cenários são descritos por uma equipe de testes e implementado por desenvolvedores do *software*. Estes cenários devem ser descritos com informações suficientes para atender grande parte das possibilidades de uma funcionalidade ou fluxo de um *software*.

Com os cenários descritos, a automatização dos testes pode ser desenvolvida, trazendo assim um melhor desempenho e agilidade no desenvolvimento do *software*.

3 METODOLOGIA

Foi realizada um estudo de caso numa instituição de ensino superior (IES), localizada no Sul do Brasil. Para manter o nome da IES íntegro, utiliza se o codinome *Alpha* para mencioná-la. A IES possui um departamento de tecnologia da informação que é responsável pelo desenvolvimento de algumas aplicações, como neste caso foi investigada a aplicação de gestão de cobrança e de atestados médicos.

A pesquisa e coleta de dados foi realizada de abril/2018 até outubro/2018 onde o pesquisador, autor desta monografia faz parte do *Dev Team* e obteve acesso a informações como documentação do *software,* relatórios e até dados primários como conversas com diretores e coordenadores da área conforme o Quadro 1.

Qudro 1 - Metodologia Utilizada na Pesquisa

Metodologia	**Estratégia de Pesquisa**	**Coleta de Dados**	**Técnica de Análise**	**Validade**
Qualitativa	Estudo de Caso – IES Brasileira *Alpha*	Dados Primários e Secundários: Conversas Informais com *DevTeam*; Relatórios; Informações dos *Softwares*	Explanação Descritiva	Triangulação de Fonte de Dados

Fonte: Autoria própria

A análise dos dados utilizado nesta pesquisa é qualitativa de forma descritiva onde foi realizada uma explanação sobre o assunto em pauta (YIN, 2012) em que é explicado como o *DevOps* pode contribuir para a melhoria do desenvolvimento de *software.*

Busca-se manter a integridade dos dados, não divulgando informações que possam comprometer este caso. Sendo assim, a triangulação dos dados ocorreu por meio de fontes primárias e secundárias.

4 ANÁLISE DOS DADOS

Com a implementação do desenvolvimento ágil e algumas práticas que envolvem o DevOps na IES *Alpha* o caso de uso levantado expõe cenários onde a automatização de processos com apoio de desenvolvedores de software e equipe de operação, ajudou a aumentar a produtividade das equipes de desenvolvimento e de qualidade.

Os processos levantados foram a automatização de Entregas de *Software*, Testes Automatizados e Monitoramento da Qualidade Continuo. Por meio da ferramenta *Team Fundation Server 2015*, da empresa Microsoft, existe a possibilidade de automatização das entregas de *software*.

Os processos adotados pela a Instituição analisada, apontam uma melhora nas entregas dos produtos. Atrelado a ferramenta de automatização, existem os processos de desenvolvimento.

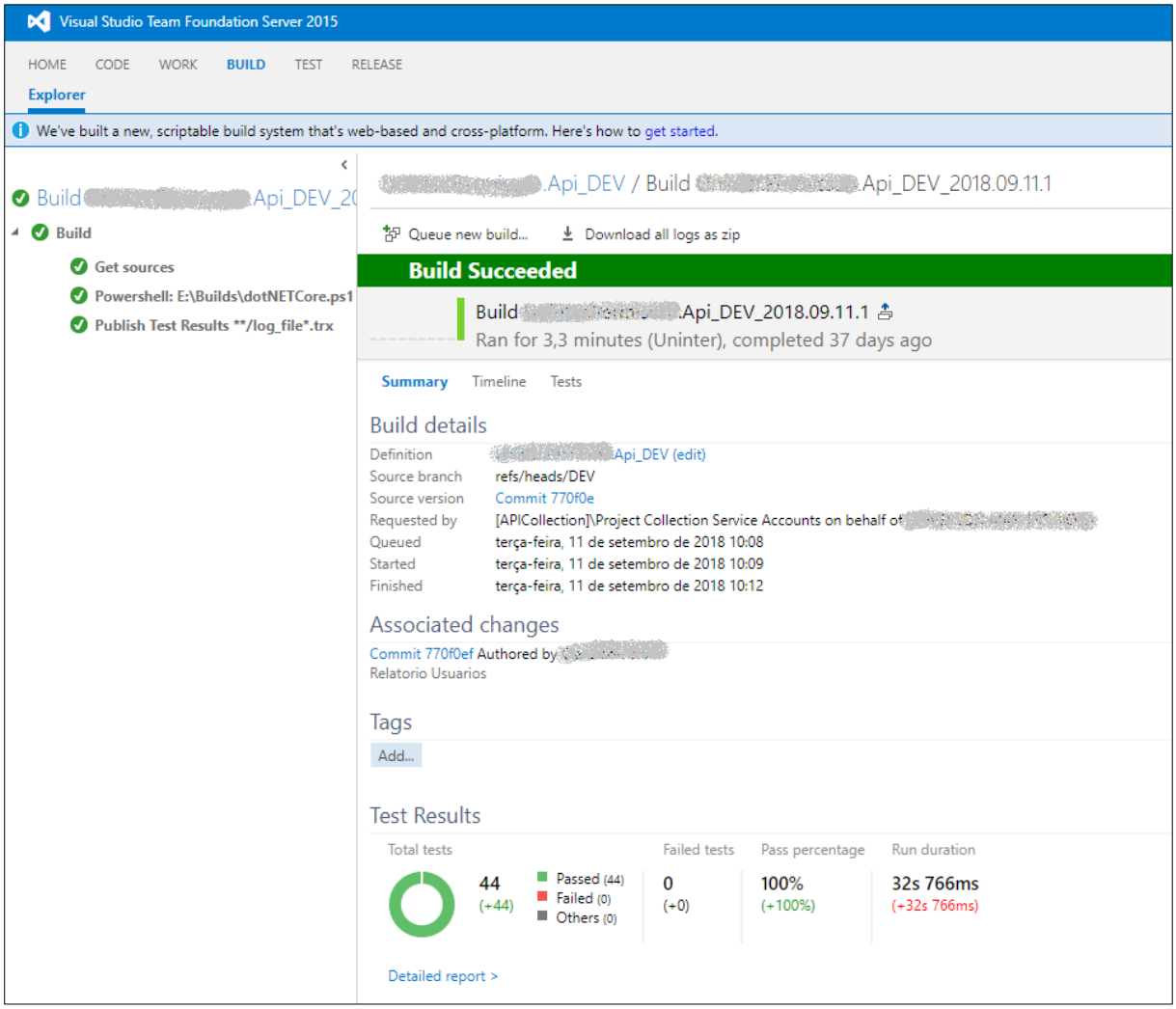
Utilizando a metodologia scrum, as equipes se organizam em tarefas previamente levantadas nas *Plannings*. As tarefas sempre fazem parte das Estórias dos usuários, definidas pelo o P.O. Neste modelo, podemos definir que um software é composto pelo desenvolvimento de tarefas que estão atreladas a necessidades dos clientes. Após o termino do desenvolvimento das tarefas, o *Dev Team* tem o objetivo de disponibilizar a funcionalidade para testes, neste momento, começam os processos de automatizações.

4.1 Automatização de Entregas de Software

Após o termino do desenvolvimento, o desenvolvedor precisa enviar o código que está na sua máquina local, para um repositório de código, neste caso, o *Git*. Após o envio deste código, é disparado um gatilho para iniciar a implementação deste código recém enviado, para um servidor de homologação.

Na figura 5, é apresentado um exemplo de um projeto com a parte de automatização de software implementada. Neste exemplo, é utilizado a ferramenta Team Foundation Server da Microsoft. O que está sendo demonstrado é uma implantação do sistema no servidor de homologação.

Figura 5 - Processo de Build Automatizado



Fonte: Team Foudation Server – IES (2018)

Para entender a Figura 5, é possível perceber que a automatização foi executada com sucesso, conforme o texto "*Build succeeded*". Na área de "*Test Results*" são apresentados o resultado da execução dos testes automatizados. Em resumo, a ferramenta da Microsoft Team Foundation Server, fez o download do código fonte de um desenvolvedor, compilou, validou de todos os testes automatizados obtiveram sucesso e fez a implantação no servidor de homologação.

4.2 Testes automatizados

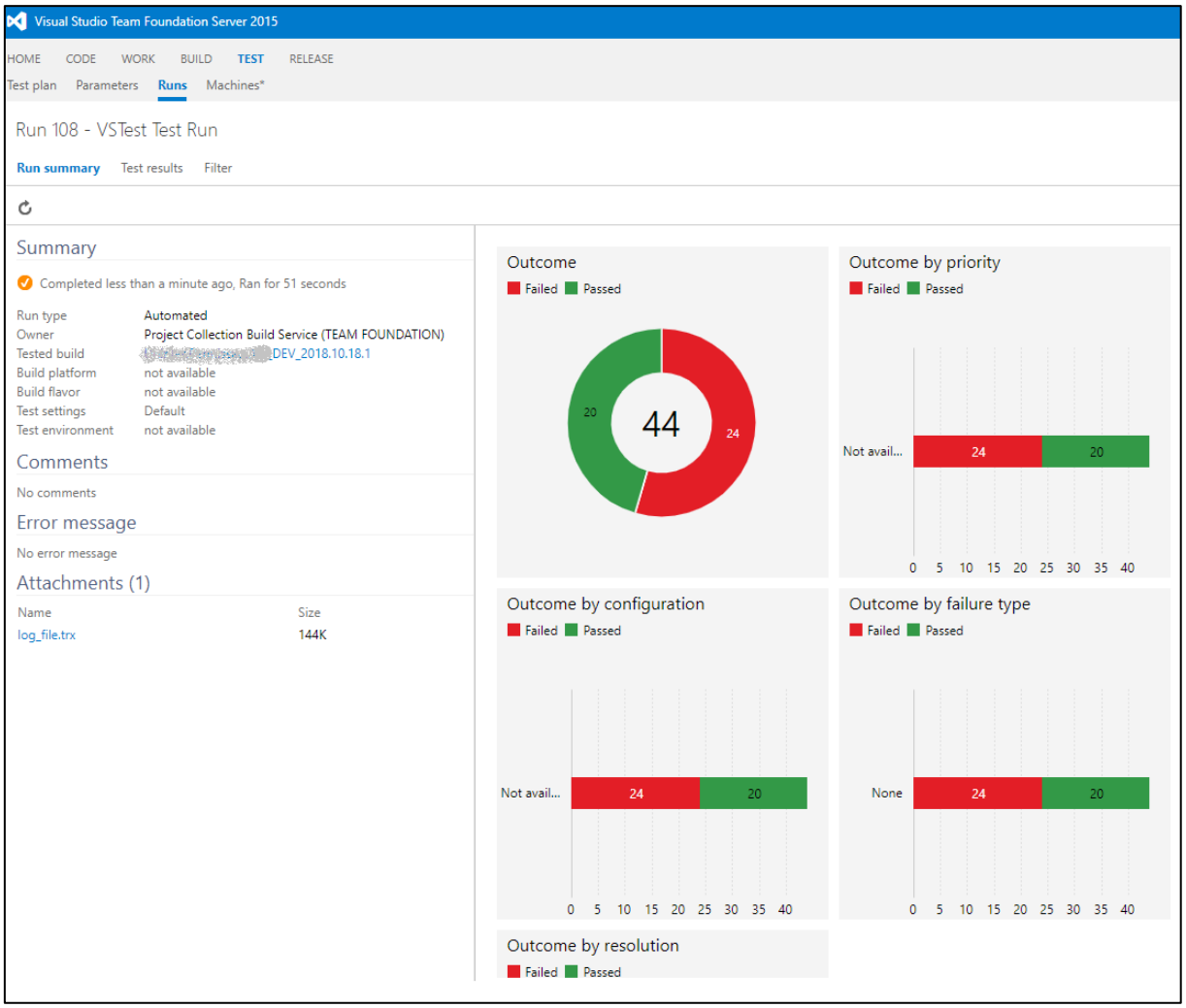
Os testes automatizados são de responsabilidade do desenvolvedor da funcionalidade. No momento que uma funcionada é criada, faz parte do escopo o desenvolvimento de um teste automatizado. A partir dos cenários levantados pelos testes, utilizando a metodologia BDD, é possível o desenvolvedor criar estes testes atendendo aos critérios dos responsáveis pelo BDD.

Os testes automatizados são executados sempre que um desenvolver envia um código para o repositório de dados. Isso garante que o código que foi enviado para é funcional. Outro ponto positivo é a velocidade com que os testes rodam, demorando em média 2 segundos.

Vale destacar, que a automatização de teste não substitui a ação de profissionais testadores de software. A automatização visa a agilidade e a garantia da manutenção do software.

Na figura 6, é apresentado um exemplo de resultado de um projeto que não obteve sucesso na execução de seus testes, impedindo o prosseguimento do software. No exemplo, de 44 testes automatizados, 20 obtiveram sucesso e os outros 24 retornaram algum erro.

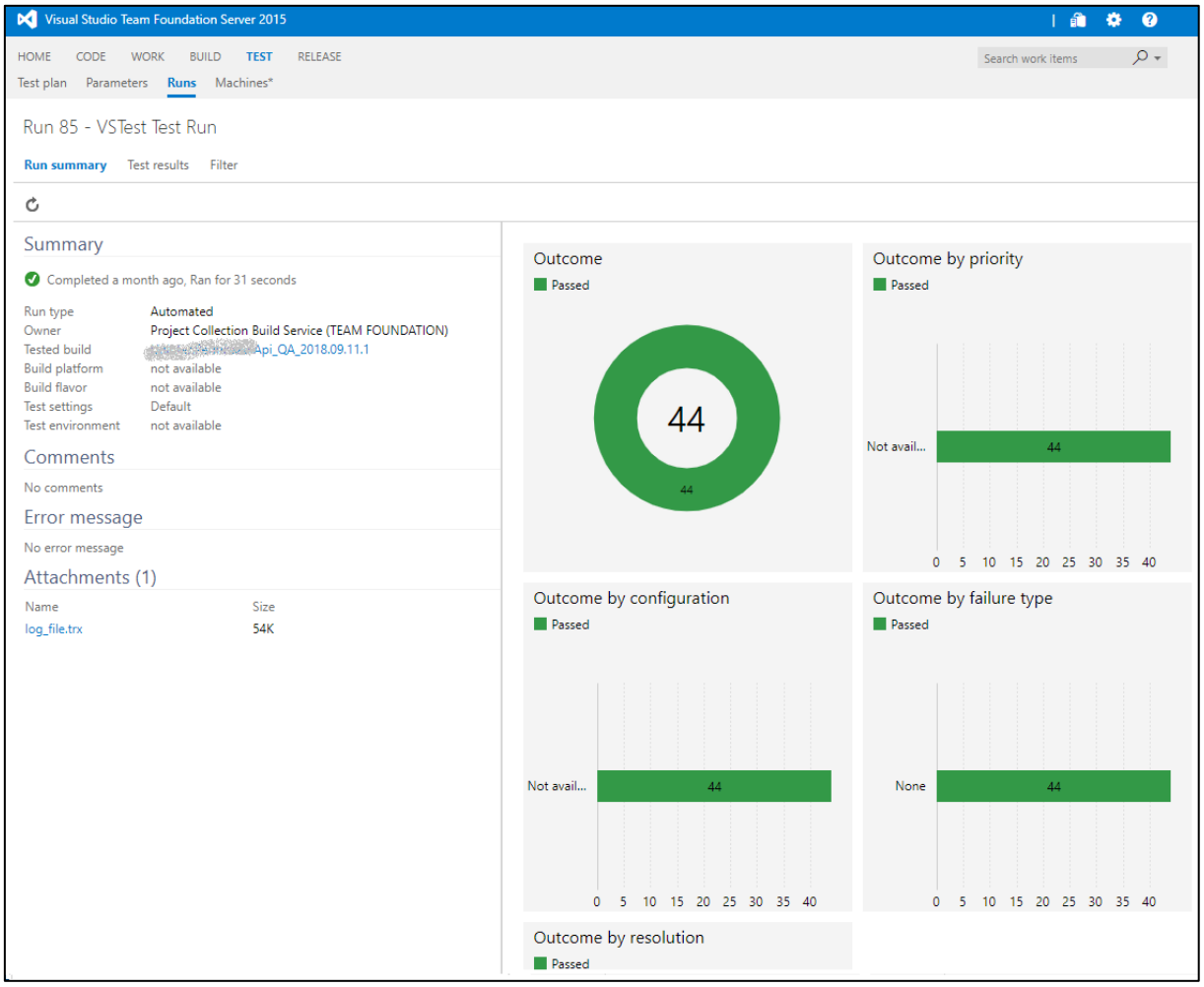
Figura 6 - Execução de Testes unário com 45% de assertividade



Fonte: Team Foudation Server – IES (2018)

Na figura 7, é apresentado um exemplo de execução com 100% de assertividade nos testes. Os 44 testes unitários desenvolvidos, foram executados com sucesso, sem nenhum erro. Quando este cenário acontece, o software está apto para ser implantado em um servidor de desenvolvimento.

Figura 7 - Execução de Testes Unários com 100% de assertividade



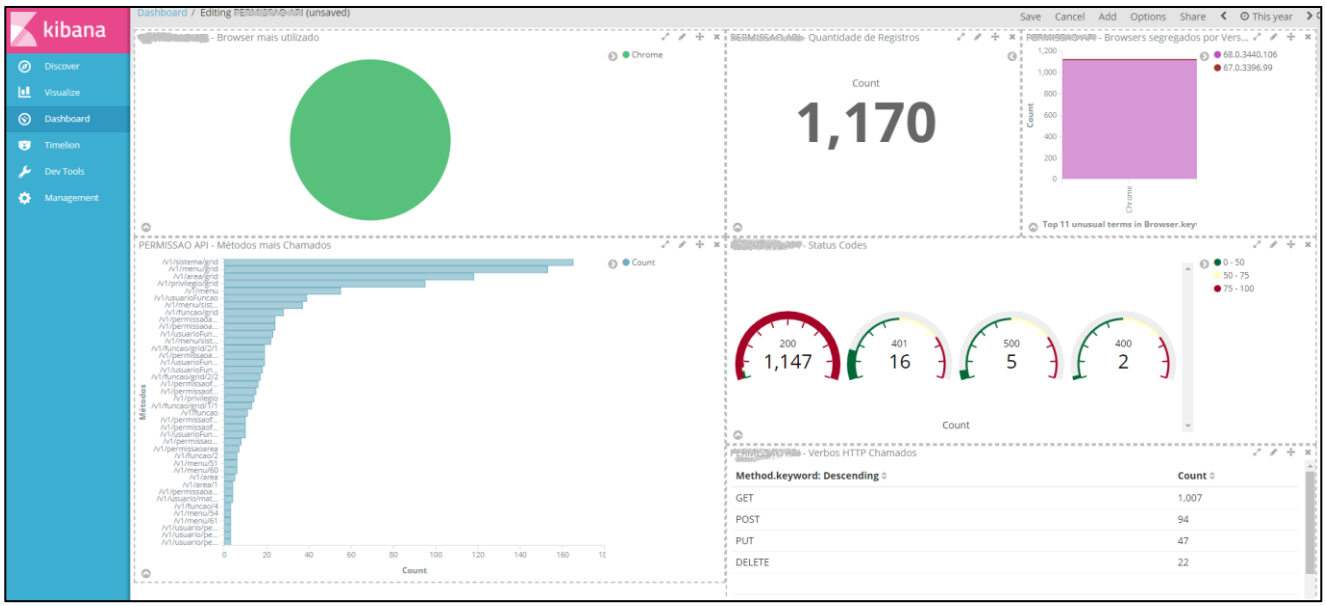
Fonte: Team Foudation Server – IES (2018)

4.3 Monitoramento continuo

O monitoramento continuo desenvolvido para os sistemas da empresa *Alpha* tem o objetivo de garantir a disponibilidade da aplicação, bem como o registro de informações para futuras melhorias.

Na Figura 8 é apresentado o *dashboard* com as informações de monitoramento da aplicação, é utilizado a plataforma Kibana que interpreta os dados de um banco não relacional chamado *Elastic Searc.*

Figura 8 - Figura 8 - Dashboard de Monitoramento Continuo



Fonte: Kibana – IES (2018)

Na figura 8 é possível verificar as seguintes informações:

- Porcentagem de *Browser* que acessam o Sistema em um gráfico de Pizza
- A Quantidade de Registros
- A Versão dos *Browser* que estão acessando o Sistema
- Os métodos que são mais utilizados
- Os *status codes* retornados pelo sistema
- Os verbos HTTP retornados

Com essas informações é possível identificar se a aplicação está com erro e quais o *browser* são mais utilizados. Outra informação importando são os métodos para identificar quais telas do sistema são mais utilizados.

Na figura 9 é um exemplo de *dashboard* de uma outra aplicação, com mais informações para análise. As ferramentas utilizadas são as mesmas do exemplo anterior.

Figura 9 - Exemplo mais robusto de Dahsboard de monitoramento

Fonte: Kibana – IES (2018)

Na Figura 9 é possível identificar as seguintes informações:

- Quantidade de Acesso por dia
- *Browser* mais utilizados
- Versões de *browser* utilizados
- Métodos mais utilizados
- *Status Codes* retornados
- Quantidade de registros mais utilizados

Com essas informações também é possível identificar se o sistema está retornando erro para os clientes e quais são as telas mais utilizados. Também é possível identificar quais versões de *browser* são mais utilizadas.

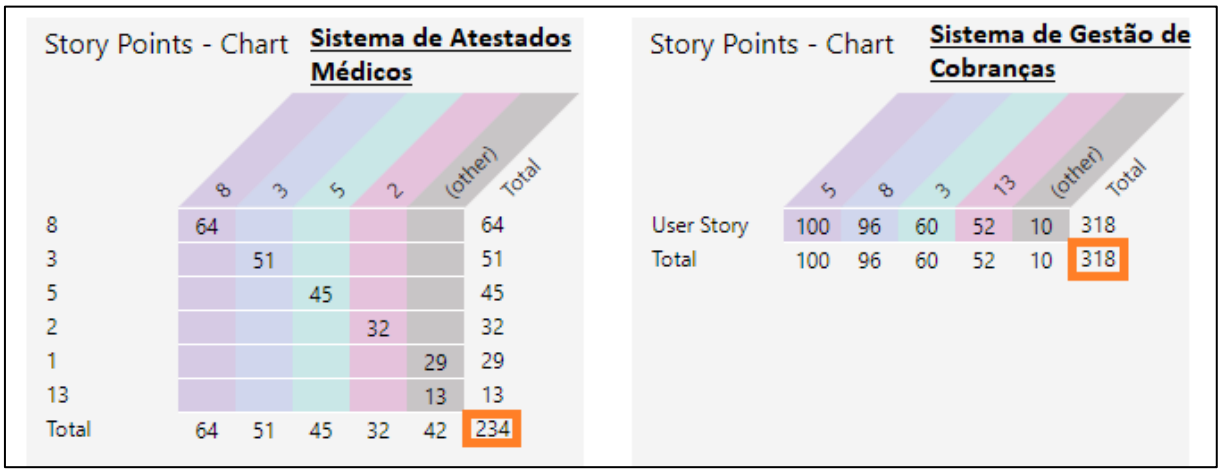
4.4 Analise de aumentado de Qualidade com Práticas DevOps

Para entender o impacto das práticas de DevOps na qualidade dos sistemas, foi feita uma comparação entre dois sistemas, o primeiro sistema de “Atestados Médicos” é um sistema que não possuí testes automatizados desenvolvidos. O segundo sistema é o de “Gerenciamento de Cobranças” que utiliza testes automatizados.

O primeiro ponto analise é o tamanho dos dois sistemas. A métrica utilizada para dimensionar o tamanho destes sistemas é o *Story Points* que é uma pontuação que a equipe de desenvolvimento fornece com base no nível de complexidade das estórias do cliente. Essa métrica faz parte da metodologia SCRUM é se utiliza da sequência Fibonacci.

Na figura 10 é apresentada a quantidade de *story points* para cada aplicação. A aplicação de Atestados médicos possuí **234** *story points* e a de sistema de gestão de cobranças possuí **318** *story points*. Logo a aplicação de sistema de cobrança supera em **26%** o tamanho da aplicação de atestados médicos.

Figura 10 - Quantidade de Story Points das Aplicações de Atestados e de Cobranças

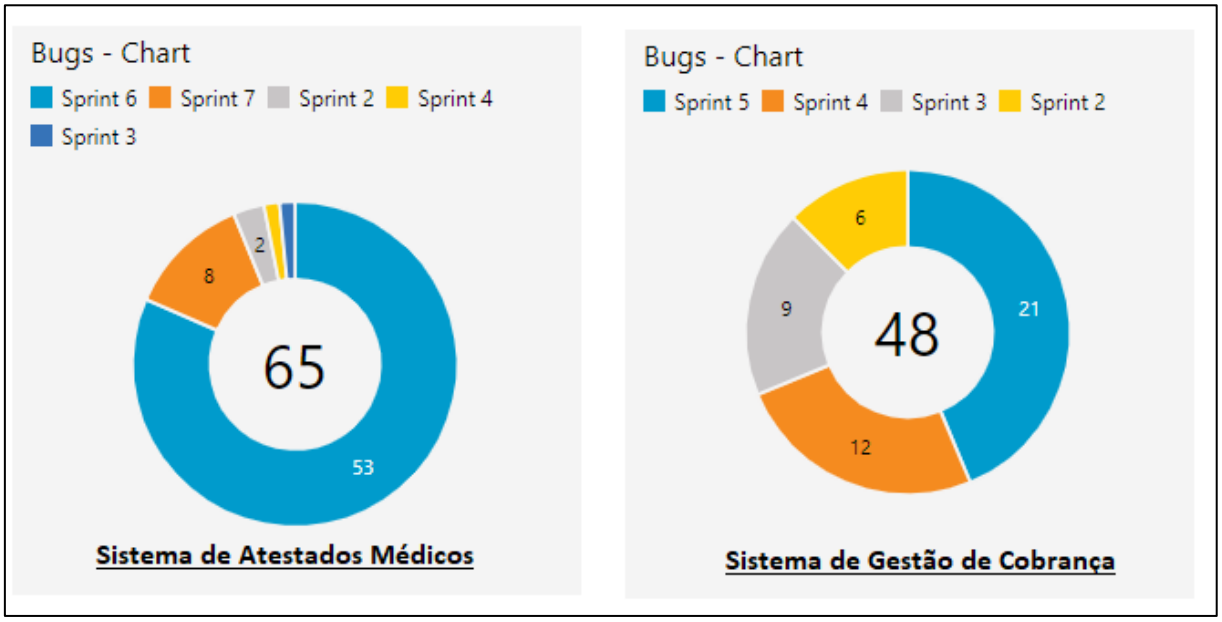


Fonte: Adaptado de Team Foudation Server – IES (2018)

É importante reforçarmos que a aplicação de Gestão de Cobranças é maior e mais complexa que a de atestados médicos, conforme evidenciado pela quantidade de *story points*.

Na figura 11 é possível identificar a quantidade de *bugs* abertos para a aplicação de “Atestado Médico” e o “Sistema de Gestão de Cobranças”. O sistema de “Atestado Médico” possuí **65** bugs e a aplicação de “Gestão de Cobranças” possuí 48.

Figura 11 - Quantidade de Bugs na Aplicação de Atestados e Cobranças



Fonte: Adaptao de Team Foudation Server – IES (2018)

Com base nesta comparação é possível concluirmos que o sistema de cobrança gerou **27%** de bugs a menos que o sistema de atestados médicos. Também podemos considerar que a aplicação de gestão de cobranças é maior que a de atestado.

De forma resumida, podemos concluir que a aplicação de gestão de cobranças, além de ser maior e mais complexa que a de atestados médicos, gerou menos bugs.

5 CONSIDERAÇÕES FINAIS

A presente pesquisa buscou responder a seguinte pergunta: Como as práticas do DevOps podem contribuir na qualidade do desenvolvimento de Software?

Com os assuntos abordados neste trabalho, buscou-se apresentar uma perspectiva genérica sobre o desenvolvimento de software e a metodologia ágil, são fundamentais para identificar como o DevOps interage dentro do ciclo de vida um software. A qualidade de software obteve uma abordagem mais ampla por ser do principal potencial competitivo nos produtos de software.

Para complementar o estudo, foi feita a apresentação de um estudo de caso onde, a partir de uma comparação de dois cenários, um sem as práticas de DevOps e outra com. Esse estudo apresentou os benefícios que o DevOps pode trazer para o desenvolvimento de software.

O tema DevOps é um assunto relativamente novo, mas se revela promissor por trazer novas possibilidades na construção de softwares. Suas práticas se encaixam dentro da arquitetura de um sistema com os princípios da colaboração, iteração e automatização. Diferente de uma metodologia de desenvolvimento de software, não existe padrões para a sua implementação, e por isso o termo está atrelado a uma filosofia, muito mais do que um modelo para o desenvolvimento.

Como sugestões de estudos futuros recomenda-se verificar como que o *DevOps* está substituindo postos de trabalho, por meio da massiva automatização dos processos de desenvolvimento de *software,* e o que era para atuar na melhoria do desenvolvimento está produzindo impactos colaterais como a robotização dos testes e desenvolvimento, o que foi verificado de modo empírico nesta pesquisa.

REFERÊNCIAS

ABNT (2005). **NBR ISO 9000-2005 - Sistemas de gestão da qualidade - Fundamentos e vocabulário**. Rio de Janeiro, Sede da ABNT, 2005.

BECK, Kent.; BEEDLE, Mike.; BENNEKUM, Arie van (2018). **Manifesto para Desenvolvimento Ágil de Software**. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html> Acesso em Julho de 2018.

BUENO, Cassiane de Fátima dos; CAMPELO, Gustavo Bueno. **Qualidade de Software**. Recife.

CRUZ, Cláudio Silva da; ANDRADE Edméia Leonor Pereira de; FIGUEIREDO, Rejane Maria da Costa. **Programa Brasileiro da Qualidade e Produtividade em Software.** Brasilia: PBQP Software, 2008. cap 2.

ERICH, Floris; AMRIT, Chintan; DANEVA, Maya. **DevOps Literature Review**. Enschede, 6, Outubro de 2014.

HUMBLE, Jezz; FARLEY, David. **Continuous Delivery.** Boston: Person Education, 2011.

JANZEN, David; SAIEDIAN, Hossein (2005). **Test-Driven Development: Concepts, Taxonomy,and Future Direction**. IEEE Computer. Setembro, 2005.

KIM, Gene; BEHR, Kevin; SPAFFORD, George. **The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win**. Portlan: IT Revolution Press, 2013.

LEVITA, Carlos de Amorim.; NETO, João Augusto Mattar. **Proposta de Modelo para Avaliação da Maturidade DevOps**. São Paulo, 06, Novembro de 2017.

LIMA, TLL; DANTAS, Ayla; VASCONCELOS, Livia M. R. **Usando o SilkTest para automatizar testes: um Relato de Experiência**. João Pessoa, 2012.

PRESSMAN, Roger S.. **Engenharia de Software - Uma Abordagem Profissional - 7º Edição**. Porto Alegre: Bookman, 2011. Cap 1, p. 29-51.

RATO, Francisco. **DevOps em Sistemas de Informação:** Implementação em Operações de Tecnologias de Informação. 2017. Universidade Nova de Lisboa, Lisboa, 2017.

RINCONM, André Mesquita. **Qualidade de Software**. Goiânia, 2006.

RODRIGUES, Milena (2015). **Introdução ao Gerenciamento de Qualidade**. DEVMIDIA. Disponível em: <https://www.devmedia.com.br/introducao-ao-gerenciamento-de-qualidade/33435> Acesso em Outubro de 2018.

SCHWABER, Ken; SUTHERLAND, Jeff. **Um guia definitivo para o Scrum: As regras do jogo**. 2013.

SHARMA, Sanjeev; COYNE, Bernie. **DevOps for Dummies**. Hoboken: John Wiley e Sons, 2015.

SMARTSHEET (2018). **The Way of DevOps: A Primer on DevOps Principles and Practices**. Disponível em: <https://www.smartsheet.com/devops> Acesso em Setembro de 2018.

SOEKEN, Mathias; WILLE, Robert; DRECHSLER, Rolf. **Assisted behavior driven development using natural language processing**. Bremen, 2012.

TELES, Vinícius Manhães. **Extreme Programming : Aprenda como encantar os seus usuários desenvolvendo software com agilidade e alta qualidade**. Novatec, 2004. cap 1, p 21-29.

ZENTGRAF, Dan. **Definindo a implementação distribuível em DevOps**. 23, Outubrod e 2012.

Extreme Programming (2013). **Extreme Programming: A Gentle Introduction**. Disponível em <http://www.extremeprogramming.org/> Acesso em Outubro de 2018.