Definição do processo de DevOps: Um estudo de caso para o Projeto de Extensão do CESJF com o Banco Alimentario de La Plata

Eduardo Gaio Mancio [1], Evaldo de Oliveira da Silva[1]

Curso de Engenharia de Software
Centro de Ensino Superior de Juiz de Fora (CES/JF) – Campus Academia
36016-000 – Juiz de Fora – MG– Brasil

{edu.gaio58,evaldo.oliveira}@gmail.com

Abstract. *Delivering new features quickly and without errors, and integration of the sectors involved in software development are some of the benefits found in DevOps. DevOps can be considered as an area of software engineering, aiming to unify software development and the infrastructure that maintains software operations. In this context, this article aims to present how a process based on DevOps can be specified, citing its main concepts and characteristics, as well as its steps and what each step provides for the development of software. The implementation of the DevOps process specified in this article was carried out in a case study based on the need to implement the managerial indicators of the CESJF extension project with the Banco Alimentario de La Plata.*

Resumo. *Entregar novas funcionalidades de forma rápida e sem erros, além da integração dos setores envolvidos no desenvolvimento de um software, são alguns dos benefícios encontrados no DevOps. O DevOps pode ser considerado como uma área da Engenharia de Software, visando unificar o desenvolvimento de software e a infraestrutura que mantém as operações de software. Neste contexto, este artigo tem o objetivo de apresentar como um processo baseado no DevOps pode ser especificado, citando seus principais conceitos e características, bem como suas etapas e o que cada etapa proporciona para o desenvolvimento de um software. A implementação do processo de DevOps especificado neste artigo, foi realizada em um estudo de caso tendo como base a necessidade de implantação dos indicadores gerenciais do projeto de extensão do CESJF com o Banco Alimentario de La Plata.*

1. Introdução

Em diferentes ambientes e organizações de desenvolvimento de software os processos realizados entre a solicitação da demanda de manutenção e a instalação das soluções de software em produção são executados manualmente. Nestes ambientes, mesmo que existam soluções que apoiem as ações de coordenação e gerência dos processos de software, o processo de entrega manual faz com que muitas organizações desenvolvam software tenham um ciclo mais longo para gerar uma nova versão da aplicação (ALMEIDA e SILVA, 2014).

Os processos de DevOps estão relacionados às atividades de desenvolvimento e manutenção de Software que permite a colaboração entre desenvolvimento e operação de software permitindo a integração das tarefas de desenvolvimento de aplicações e serviços utilizados no processo de software (BRAGA, 2015).

De acordo com a Microsoft, o DevOps é uma prática que unifica pessoas, processos e tecnologias entre desenvolvimento e Tecnologia da Informação (TI) em 5 (cinco) principais práticas: a-) planejamento e acompanhamento; b-) desenvolvimento; c-) build e teste; d-) implantação e entrega e; e-) monitoramento, gerenciamento e operações (MICROSOFT, 2019). Desta forma, a criação de um processo de DevOps permite a redução do tempo da entrega do desenvolvimento, permitindo mapear por meio de um processo automatizado o desenvolvimento de uma aplicação. Além destas práticas, é possível relacionar outras características tais como, a preparação da infraestrutura de servidores e serviços para automação da entrega do software.

A empresa Atlassian, que distribui o software JIRA também possui soluções integradas para implementação de processos de DevOps. Neste caso, o serviço chamado Bamboo é encarregado de integrar a abertura de *issues* por meio do JIRA e o versionamento de software feito pelo BitBucket, visando a automatização das entregas (BAMBOO, 2019).

Recentemente, os cursos de Bacharelados em Sistemas de Informação e Engenharia de Software do CESJF foram contemplados com as licenças de todas soluções fornecidas pela Atlassian, facilitando o aprendizado de diferentes soluções para manutenção, teste e desenvolvimento de software nas diferentes disciplinas técnicas, incluindo o serviço Bamboo usado para DevOps. Deste modo, permitiu-se o desenvolvimento do conhecimento mais aplicado das técnicas de desenvolvimento além das ferramentas da Atlassian poderem ser utilizadas em projetos de iniciação científica, por exemplo, o projeto de geração de indicadores gerenciais para o Banco Alimentário de La Plata, que é uma ONG (Organização Não Governamental) Argentina (CESJF).

De acordo com as definições e descrições acima, este artigo tem o objetivo de apresentar como um processo baseado no DevOps pode ser especificado, citando seus principais conceitos e características, bem como suas etapas e o que cada etapa proporciona para o desenvolvimento de um software, especificamente. A implementação do processo de DevOps especificado neste artigo, foi realizada em um estudo de caso tendo como base a necessidade de implantação dos indicadores gerenciais do projeto de extensão do CESJF com o Banco Alimentario de La Plata.

É importante ressaltar que este artigo visa a implantação do processo de DevOps para automação da entrega dos indicadores gerenciais, e com base na infraestrutura oferecida pelas soluções da Atlassian. A infraestrutura de servidores e instalação dos serviços das aplicações da Atlassian foram realizadas e configuradas juntamente com o setor de tecnologia da informação do CESJF.

O restante do trabalho se encontra organizado em seções. A Seção 2 descreve o referencial teórico que permitiu o embasamento de conceitos abordados neste trabalho, tais como, DevOps, ferramentas e técnicas para implantação de processos nessa área. A Seção 3 propõe um processo para implantação de DevOps. A Seção 4 mostra a implementação do processo proposto utilizando as soluções da Atlassian e finalmente, a Seção 5 são descritas as considerações finais e trabalhos futuros.

2. Referencial Teórico

DevOps é uma cultura que engloba diversas práticas e ferramentas que tem como principal objetivo integrar os setores de desenvolvimento e de operações. Surgiu da necessidade do mercado de realizar entregas mais rápidas e constantes de novos produtos, mas mantendo a qualidade nas entregas. A cultura está constantemente associada ao uso de ferramentas e tem grande foco em metodologias ágeis. (AMAZON, 2019a)

2.1. Conceitos sobre DevOps

Para implementar a cultura DevOps, é necessário estar atento aos seus principais pilares, sendo eles a integração, entrega e o feedback contínuo.

A integração continua consiste em manter o código de uma aplicação atualizado para que todos os envolvidos no projeto possam ter acesso a ele enquanto o produto é desenvolvido. Isso ajuda na detecção de bugs mais rapidamente e melhora a qualidade do software, o que proporciona um desenvolvimento mais rápido e eficiente (AMAZON, 2019b).

A entrega contínua consiste em um conjunto de práticas que tem como objetivo garantir que um código esteja apto para ser enviado para o ambiente de produção a qualquer momento, ou seja, o código além de estar sempre atualizado por causa das práticas de integração continua, também precisa ser confiável, livre de erros e acessível a qualquer momento (4LINUX, 2019a).

O feedback contínuo é importante, pois toda a equipe envolvida em todas as fases do processo é capaz de dar feedbacks constantes sobre ele. Os feedbacks permitem que ajustes sejam feitos no processo mais rapidamente e, consequentemente, de forma mais econômica (IBM, 2019).

2.2. Modelagem de Processos para Devops

A área de desenvolvimento de software está muito presente no negócio, podendo, a todo momento, demandar novas funcionalidades para atender ao mercado. Para que isso se tornasse possível, métodos ágeis foram implementados nessa área, aprimorando-a para atender ao mercado. Em contrapartida, a área de operações requer o mínimo de alterações possíveis na infraestrutura de um software, já que cada nova alteração pode gerar uma instabilidade no sistema (4LINUX, 2019b).

Para que a área de operações possa acompanhar as rápidas mudanças do desenvolvimento, o DevOps propõe algumas técnicas a serem aplicadas, permitindo que a infraestrutura de um software também seja ágil. Algumas dessas técnicas são: a implementação e o gerenciamento de ambientes de forma automatizada; a criação de testes automatizados para validar esses ambientes e o controle de versão desses ambientes (4LINUX, 2019b).

Um processo DevOps é extenso e envolve diversas pessoas em vários setores de uma organização, além de um número grande de ferramentas e, por isso, modelar esse processo é de extrema importância e relevância (MICROFOCUS, 2019).

Quando o processo DevOps é bem modelado, deixará claro o que será feito em cada uma das etapas e quais ferramentas deverão ser utilizadas.

Segundo a Micro Focus, um modelo de processo DevOps completo e eficiente contém oito módulos em sua composição (MICROFOCUS, 2019):

1. Definição e planejamento: onde é feita toda a gestão de quais atividades serão realizadas, quais são mais importantes no momento, quanto tempo será gasto na sua realização, qual é o valor monetário necessário para sua execução, quais os planos de testes que deverão ser criados e como será a arquitetura do projeto.
2. Integração contínua: esse módulo contempla toda a parte de desenvolvimento das atividades definidas anteriormente. Os desenvolvedores irão dar inicio ao processo de codificação com base no planejamento. É importante que todos tenham acesso aos requisitos de suas tarefas e a mesma versão do código, portanto, é necessário o uso de repositórios de código. Realização de testes e builds também acontecem nesse modulo.
3. *Deploy* contínuo e release: nesse módulo é importante a presença de técnicas de *deploy* contínuo. Essas técnicas são criadas e configuradas através de ferramentas que realizarão o processo de *deploy* assim que uma ação ocorrer. Geralmente é quando há um merge do código para o repositório principal.
4. Teste contínuo: com o *deploy* feito em um ambiente de testes, vários testes automatizados devem ser executados para prevenção de erros no ambiente de produção. Testes de performance, funcionais e segurança são exemplos que podem ser automatizados e executados.
5. Segurança contínua: nesse módulo é importante a presença de testes de vulnerabilidade ao longo do ciclo de vida do software.
6. Operação contínua: etapa onde o código está em produção. É necessário que haja monitoramento continuo da aplicação para detectar possíveis erros e problemas de performance na medida em que o software é utilizado.
7. Colaboração contínua: caso algum problema ou bug seja encontrado no ambiente de produção, a equipe de suporte deverá se comunicar com a equipe de desenvolvimento para que o problema possa ser relatado e corrigido.
8. Avaliação contínua: nesse módulo é importante sempre avaliar todo o processo de DevOps e como está o andamento de todas as etapas. O objetivo é determinar quais etapas estão apresentando gargalos para que uma ação possa ser tomada com a intenção de melhorar o processo como um todo.

Para todos esses módulos existem ferramentas que auxiliam a equipe em cada etapa. Para que o processo seja eficiente é importante que as ferramentas estejam presentes, pois elas ajudam na gestão de cada etapa.

2.3. Ferramentas para implementação de Devops

Atualmente existem várias ferramentas que auxiliam na implementação de todas as fases do DevOps. Existem tanto ferramentas *open source* quanto de código proprietário. A seguir serão listadas algumas das ferramentas mais usadas na implementação da cultura DevOps.

- Etapa de planejamento: Asana, Trello, Slack, Confluence, JIRA

- Etapa de codificação: Git, GitHub, GitLab, Bitbucket, Mercurial
- Etapa de *build*: Bamboo, Maven, Docker, ANT, Travis CI
- Etapa de teste: Selenium, JUnit, xUnit.net, JMeter
- Etapa de *deploy*: Bamboo, AWS CodeDeploy, Jenkins
- Etapa de monitoramento: Google analicts, Hotjar

2.4. A ferramenta JIRA

Nesse trabalho uma das ferramentas que será utilizada é o JIRA, que é um software desenvolvido pela empresa Atlassian. Sua principal função é auxiliar no planejamento e no acompanhamento de todos os tipos de trabalhos.

O JIRA permite que a equipe de desenvolvimento planeje e distribua as atividades que serão realizadas em uma *sprint*. Também permite o gerenciamento de projetos e atividades, assim como uma visão geral do andamento do projeto, através de relatórios de desempenho e status das sprints (JIRA, 2019).

2.5. Bamboo, ferramenta de integração contínua

O Bamboo também é uma ferramenta desenvolvida pela Atlassian. É um servidor que permite a implementação do processo de entrega contínua. Nele é possível criar um processo de integração contínua para um software assim como realizar testes e criar *deploys* de forma automatizada (BAMBOO, 2019).

Configurando as etapas da integração contínua, é possível configurar planos de compilação automatizados que podem ser disparados por diversas triggers, que também são configuradas no plano. Isso manterá o código sempre compilado e atualizado.

A ferramenta permite que o usuário cadastre diversos testes automatizados que verificam a integridade do código sempre que uma compilação é iniciada e, caso a compilação não passe em algum teste, o informa para que uma ação possa ser tomada.

É possível configurar diversas formas de *deploy* utilizando a ferramenta. Quando a compilação é feita e os testes são concluídos com sucesso, é possível também configurar como e onde o código terá o seu *deploy* realizado.

Finalmente, pode-se ressaltar a importância do JIRA e demais pacotes de software da Atlassian como ferramentas úteisde implementação do DevOps, pois permite a automatização de diversas etapas do processo de forma simples e centralizada. Na unidade 4 desse trabalho, a ferramenta será melhor explicada com um exemplo prático de seu uso.

3. Definição do processo de Devops: Um estudo de caso para o Projeto de Extensão do CESJF com o Banco Alimentario de La Plata

Um processo de DevOps bem definido tem o objetivo de integrar as equipes de desenvolvimento com a equipe de operações conforme visto na seção 2 deste trabalho.

Para a execução dos processos de DevOps é necessária a definição de um processo que esteja alinhado às necessidades do projeto de software. Nesse contexto, esse trabalho mostra um processo de DevOps modelado e apresentado na Figura 1. Este processo é proposto para a automatização de rotinas de criação de indicadores

gerenciais para o Banco Alimentário de La Plata, o qual possui um acordo de cooperação com o CESJF. Esse acordo de cooperação prevê a implementação de 20 indicadores gerenciais, e um processo de integração contínua torna-se relevante ser implementado neste contexto. Tais indicadores estão em fase de construção, e as rotinas de software em desenvolvimento estão especificadas também como trabalho de conclusão de curso (TCC) no CESJF, defendido e aprovado pelo aluno Guilherme Latuque no ano de 2019.

É importante que a integração contínua seja automatizada e representada por meio de um processo a fim de torná-la documentada, principalmente em equipes que utilizam metodologias ágeis e práticas de entrega contínua. As etapas do processo de DevOps são apresentadas na Figura 1.

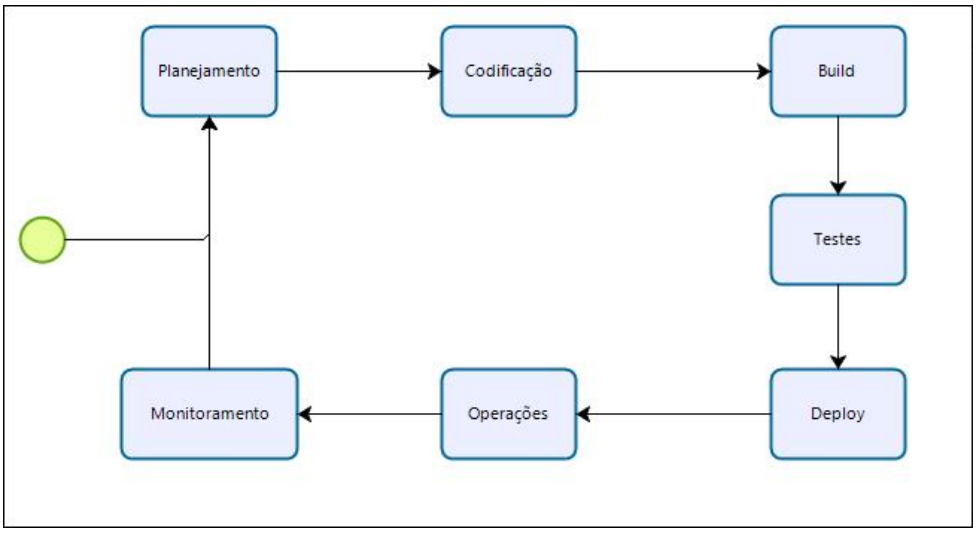

Figura 1. Modelo de processo para integração continua.

1. Etapa de planejamento: onde a equipe realiza reuniões para levantamento de quais serão as *features* (funcionalidades) entregues ao final do processo. Por ter forte influência da metodologia ágil, as entregas geralmente são planejadas em *sprints* que duram de 15 a 30 dias. As *features* são definidas baseadas nos objetivos do produto e no feedback da equipe de operações.

2. Etapa de codificação: com o planejamento pronto, a equipe de desenvolvimento divide as tarefas e começa a codificar. É preciso que as tarefas sejam acessíveis para todos os membros da equipe, pois é importante que a equipe saiba como está o andamento do projeto. O framework Scrum é bastante utilizando nessa etapa, pois ele permite maior clareza no andamento das etapas que estão sendo desenvolvidas.

3. Etapa de build: nessa etapa, o código já pronto é compilado gerando uma versão do software com todas as funcionalidades realizadas. Desta forma a integração contínua torna-se muito importante, pois é uma prática onde os desenvolvedores, frequentemente, juntam suas alterações de código em um repositório central. Isso garante que o código da equipe esteja sempre atualizado, ajudando a detectar bugs mais rapidamente, através de testes unitários, reduzindo o tempo de validação e lançamento de novas versões do software. A integração

continua permite que a equipe de desenvolvimento seja mais produtiva além de reduzir o número de erros e *bugs* encontrados pelo cliente.

4. Etapa de teste: com o código compilado, a equipe de teste realiza os processos de testes para garantir que as funcionalidades estejam condizentes com o que foi planejado. Como podem ser muitos desenvolvedores que trabalham no mesmo projeto, os testes também são úteis para garantir que não houve nenhum problema relacionado aos diversos *merges* feitos. Caso alguma inconsistência seja identificada, a equipe de desenvolvimento deve ser notificada para corrigi-la. Com a inconsistência resolvida, o fluxo tem continuidade.

5. Etapa de *deploy*: com o código testado e aprovado, está na hora de fazer o *deploy* no ambiente de produção. Nessa etapa é implementada a prática de entrega contínua, que tem como principal função, garantir que as alterações do código sejam automaticamente preparadas para serem liberadas no ambiente de produção. Uma das grandes vantagens da entrega contínua é permitir que um código pronto para *deploy* seja preparado facilmente, isso agiliza muito o processo de publicação.

6. Etapas de operações e monitoramento: com o código no ambiente de produção, a equipe de operações fica responsável por resolver os incidentes, caso apareçam. Também é função da equipe de operações, monitorar a performance do código nos servidores de produção e dar feedbacks para a equipe de desenvolvimento, para que eles possam planejar as novas *features* baseando-se, também, em como está o feedback do produto.

Para que o processo de DevOps ocorra corretamente e seja eficaz, é importante que as equipes tenham em mente que, ambas, são dependentes e que o sucesso de uma depende do sucesso da outra. Uma comunicação ativa entre os setores envolvidos é extremamente importante.

É importante frisar que esse processo é feito através de várias ferramentas que automatizam várias partes do setor de desenvolvimento até o de operações conforme apresentado na Figura 1.

4. Implementação do processo de Devops utilizando a ferramenta JIRA

No projeto Banco Alimentario de La Plata, o processo de DevOps será implementado utilizando as ferramentas da empresa Atlassian. A proposta da implantação é automatizar um processo de build como parte da aplicação da técnica de DevOps. As ferramentas utilizadas para tal implementação serão listadas a seguir.

A etapa de planejamento utilizou-se as ferramentas Confluence e JIRA Software. Com o Confluence é possível documentar e planejar o roteiro de desenvolvimento das atividades (CONFLUENCE, 2019).

Utilizando o JIRA Software (JIRA, 2019) a equipe montará as sprints baseando-se no roteiro criado no Confluence (CONFLUENCE, 2019). Serão criados itens que representarão as atividades desenvolvidas na Sprint.

Para criar os itens utilizando o JIRA é necessário entrar na ferramenta através de um login e senha e ir na opção “Criar” que está disponível na barra do menu inicial (Figura 2).

Após clicar no botão, uma janela aparecerá na tela. Nela será possível cadastrar as informações do item a ser criado. É possível selecionar a qual projeto esse item pertencerá; qual será seu tipo (melhoria, tarefa ou problema); escrever um resumo que servirá de título para o item, e também informar quem é o solicitador desse item. Essas são as informações mínimas necessárias para a criação de um item no JIRA (Figura 3).

Além das informações obrigatórias, também será possível preencher outras informações opcionais que servirão para auxiliar no desenvolvimento e no planejamento da atividade, sendo elas a descriação sobre o que deverá ser feito na atividade; qual será a prioridade da atividade, que pode variar de muito baixa a muito alta; definição de quais os rótulos a atividade fará parte; definição de um tempo estimado para a realização da atividade; determinação da data em que a atividade precisa estar concluída; definir se essa atividade tem alguma relação com outra (se ela bloqueia outra atividade ou se sua criação foi causada por conta de outra atividade, por exemplo), e também designar um responsável para a atividade já na sua criação (Figura 4).

Com o planejamento das atividades da *Sprint* concluído, a etapa de codificação é iniciada. Com as atividades definidas e separadas em itens, o desenvolvedor irá se colocar como responsável em cada item que precisará codificar e iniciará o seu trabalho.

Uma das formas de visualização disponibilizadas pela ferramenta é através de um *kanban board*. A estrutura básica do *kanban* vem com três etapas: a fazer, em progresso e finalizado. O item criado anteriormente se transforma em um cartão na visualização *kanban* e só possui algumas informações básicas da atividade, sendo elas o número identificador do cartão; seu título; quem será o responsável; o tipo de item que o cartão representa e qual é a sua prioridade (Figura 5). Para acessar o restante das informações é necessário clicar no cartão, isso fará com que uma aba lateral apareça com as informações mais detalhadas (Figura 6). A ferramenta também disponibiliza uma visualização de *scrum board*, porém foi utilizada a visualização de *kanban* por possibilitar facilmente uma visão geral sobre o andamento da *sprint*.

Figura 2. Menu da ferramenta JIRA.

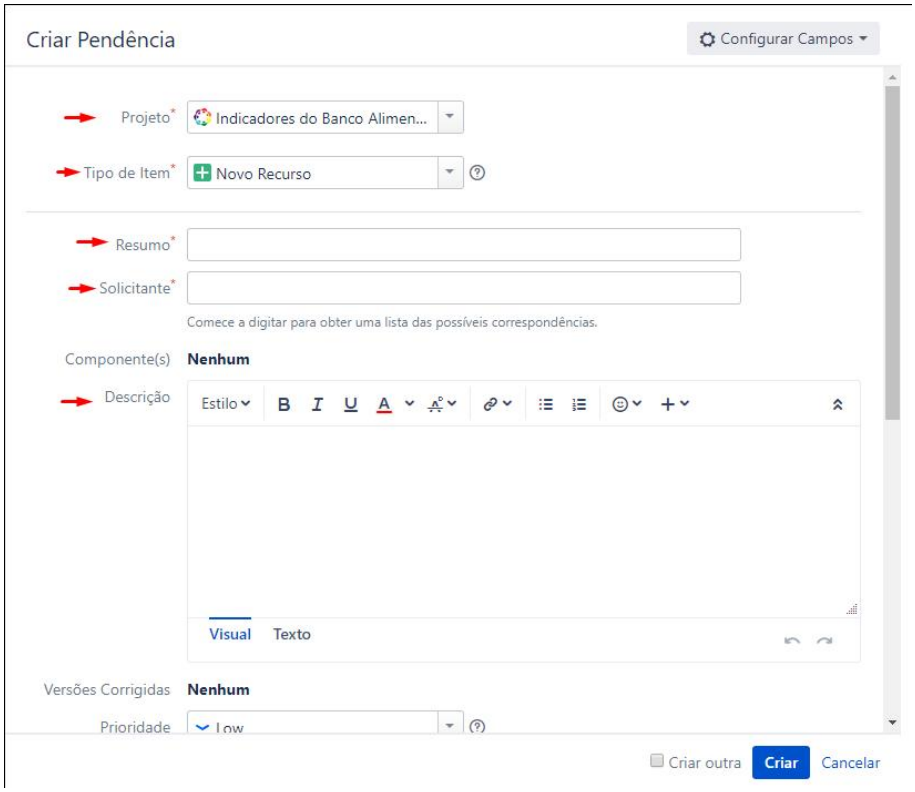

Figura 3. Janela de criação de um item no JIRA (1/2)

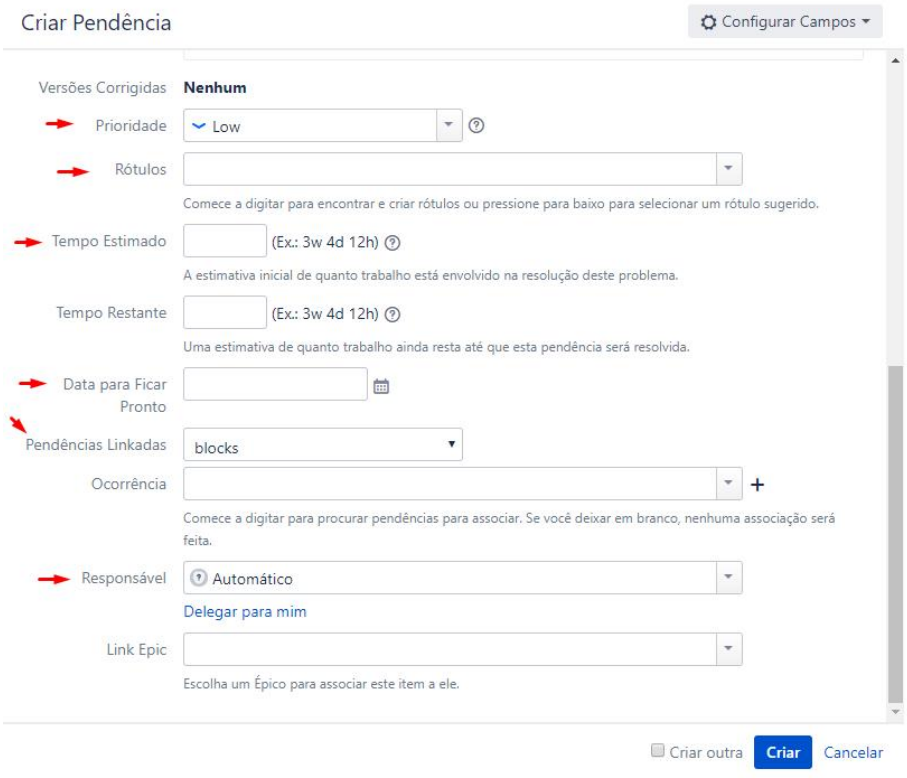

Figura 4. Janela de criação de um item no JIRA (2/2)

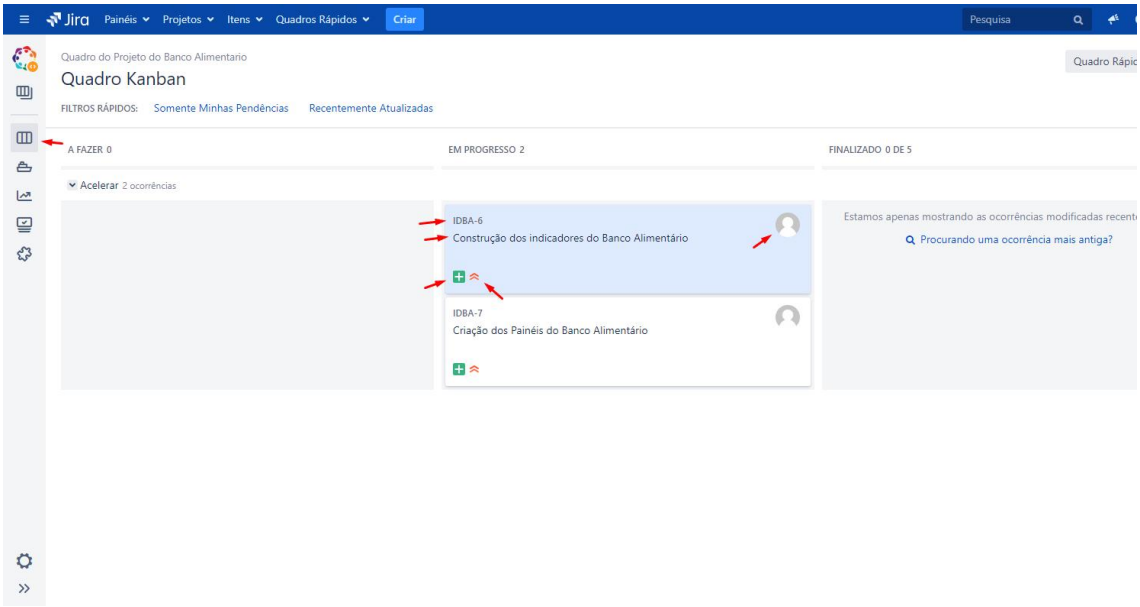

Figura 5. Visualização de Kanban.

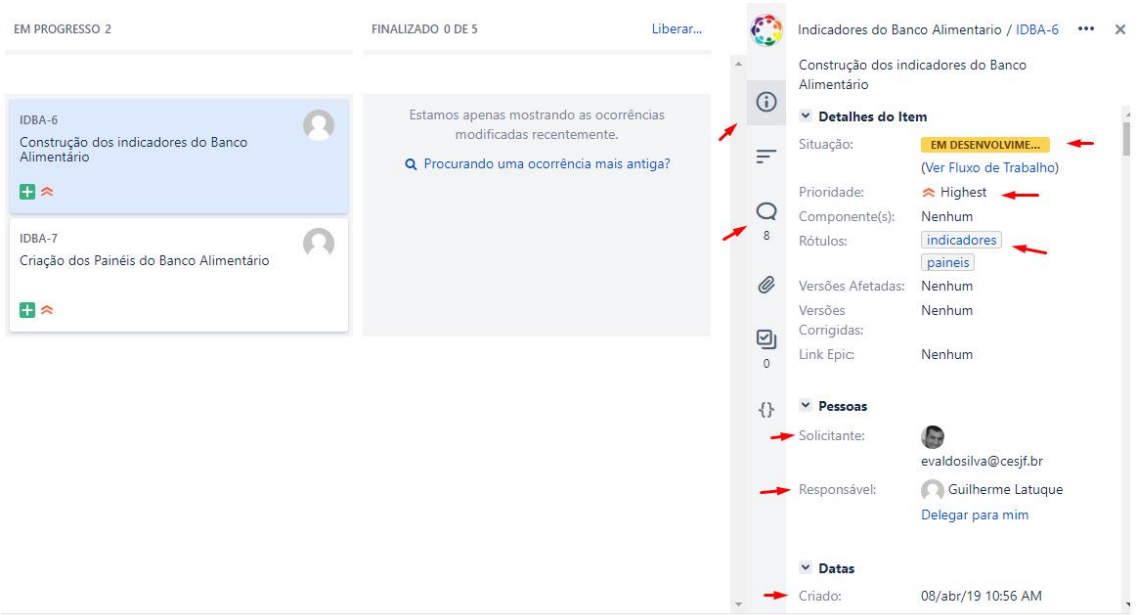

Figura 6. Visualização Kanban com informações complementares da atividade

Ao finalizar as atividades, o desenvolvedor poderá realizar o *commit* das atividades. O *commit* será enviado para o repositório de código criado no Bitbucket (BITBUCKET, 2019) que desempenhará a função de centralização do código e de controle de versão. Assim haverá a garantia de que o código estará sempre atualizado. O *commit* representa apenas o envio do código localmente. Para enviar o código ao repositório principal, haverá a necessidade de realizar um *push* que será feito por meio de *pull request*. Isso quer dizer que os *pushs* não serão imediatamente adicionados no repositório central. Será necessário a aprovação de outra pessoa para que o código possa entrar no repositório, garantindo uma validação a mais antes de enviar as modificações.

Através do Bitbucket é possível ter uma visão geral dos *commits* realizados; quem será o autor; o código identificador do *commit*; sua mensagem; data em que foi realizado; a qual item pertence e se o Build foi realizado com sucesso (Figura 7).

Com a integração entre o JIRA Software e o Bitbucket, os *commits* realizados no repositório poderão ser associados facilmente aos itens criados no JIRA. Para que a associação entre as duas ferramentas seja feita corretamente, é necessário que o desenvolvedor insira o código do item na mensagem do *commit*. O código precisa ser inserido no início da mensagem. Isso fará com que um link entre o *commit* e o item seja criado (Figura 8). Assim será possível acessar as informações do item apenas clicando no código do mesmo (Figura 9).

Após a etapa de codificação, vem a de *Build* e Testes. Utilizando a ferramenta Bamboo é possível criar um plano de *build* do projeto que será executado de forma automatizada. Para criar um plano é necessário informar o projeto, nome do plano, chave e repositório (Figura 10).

Um plano de *build* simples no Bamboo é composto por um conjunto de *jobs* que executam várias tarefas quem produzem artefatos no sucesso da execução.

A ferramenta permite que o usuário configure gatilhos que indicarão quando o *build* será iniciado. Também é possível configurar testes automatizados que verificarão a integridade do *build* sempre que ele ocorrer. Outras funcionalidades que a ferramenta disponibiliza é a possibilidade de configurar permissões de acesso ao plano, criação de notificações que serão enviadas em eventos específicos, assim como determinar se um *job* depende de outro, entre outras.

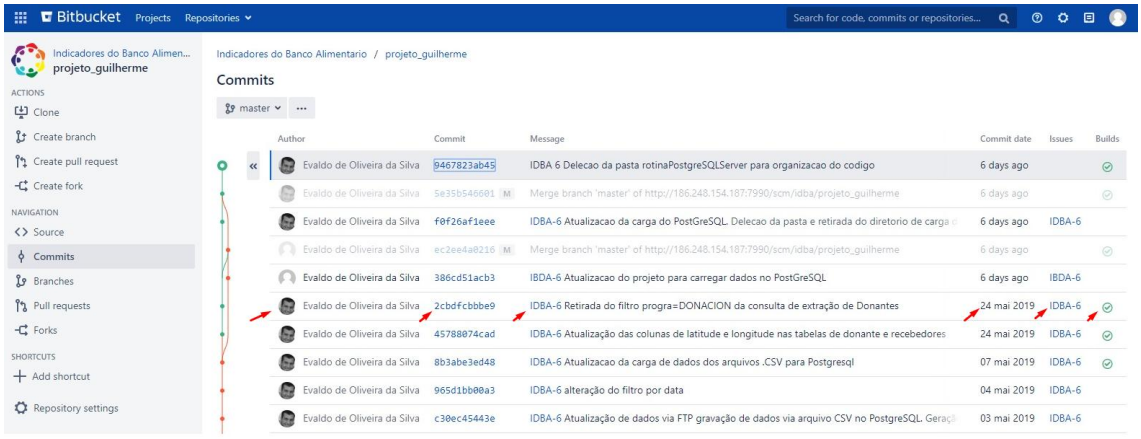

Figura 7. Tela de commits de um repositório criado no Bitbucket

Figura 8. Código do item criado no JIRA no commit do Bitbucket

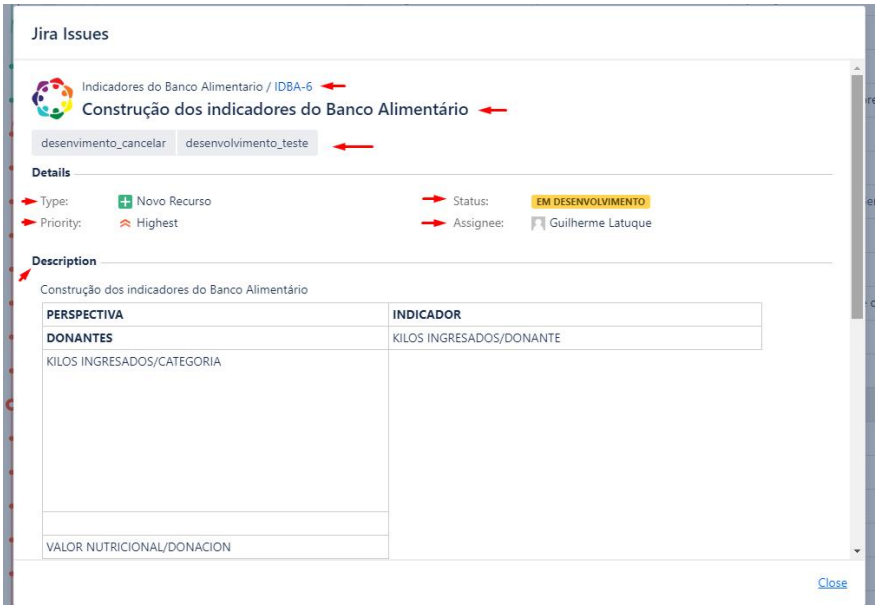

Figura 9. Janela com informações do item criado no JIRA.

Caso o *build* apresente algum problema ou não passe em alguma etapa de verificação criada na configuração, o sistema alerta o usuário para que esse possa tomar alguma decisão. É possível visualizar todos os *commits* feitos no *build*, facilitando assim, a identificação de um possível erro. Para esse trabalho, um *job* foi criado com uma tarefa de realizar o *checkout* da *branch* de um repositório criado no Bitbucket (Figura 11) e produzir artefatos .jar ao final da execução conforme configurado no *job* (Figura 12). O *job* será executado todos os dias ao meio-dia conforme configurado no gatilho (Figura 13).

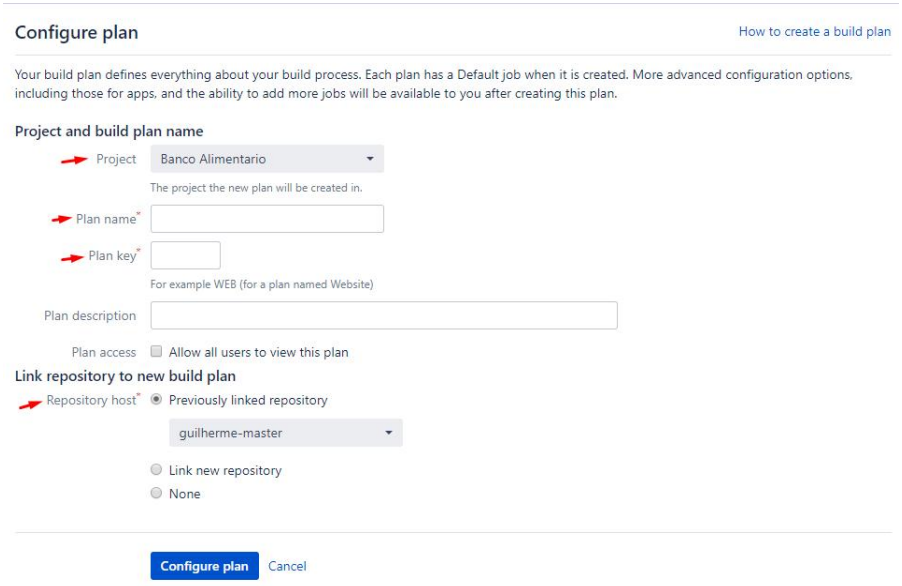

Figura 10. Tela de cadastro de um plano no Bamboo

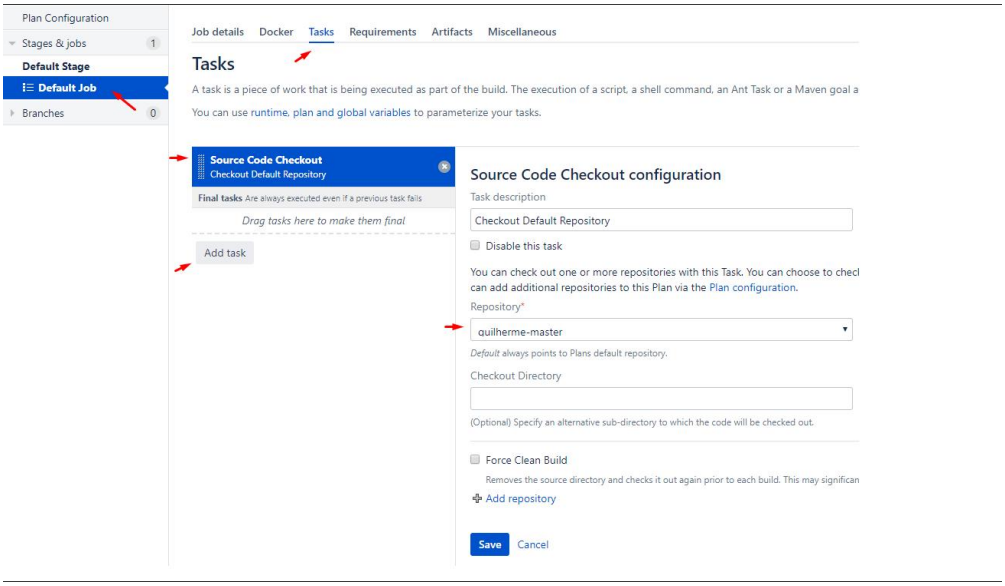

Figura 11. Tarefa responsável por dar checkout em um repositório

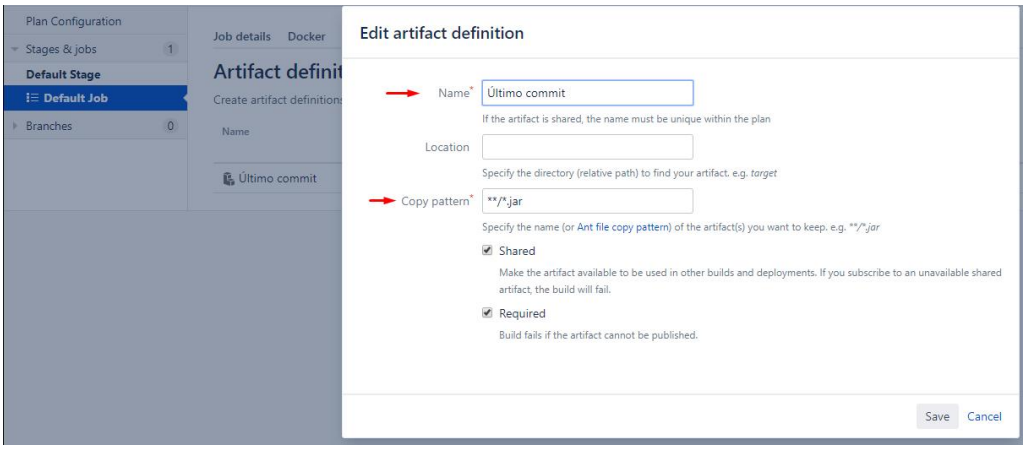

Figura 12. Configuração de um artefato que será produzido no sucesso do build

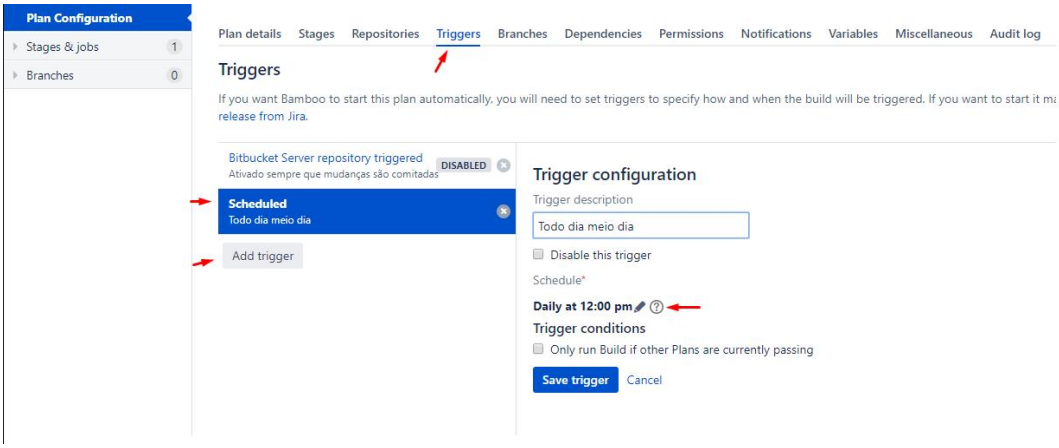

Figura 13. Configuração de gatilho de build

Se o *build* for executado com sucesso, o plano de *Deploy*, que também é configurado utilizando o Bamboo, entra em ação. Para criar um plano de *Deploy* é necessário informar um nome e qual plano de *Build* ele será referenciado (Figura 14). Após a criação, é necessário configurar o plano informando qual é o padrão de versionamento que o plano deve criar e criando tarefas informando como o *deploy* funcionará.

Nesse trabalho, o plano de *Deploy* será configurado para realizar o *deploy* dos artefatos gerados após o sucesso do *build* em um caminho, dentro do próprio servidor, preestabelecido na criação de uma tarefa do plano. (Figura 15).

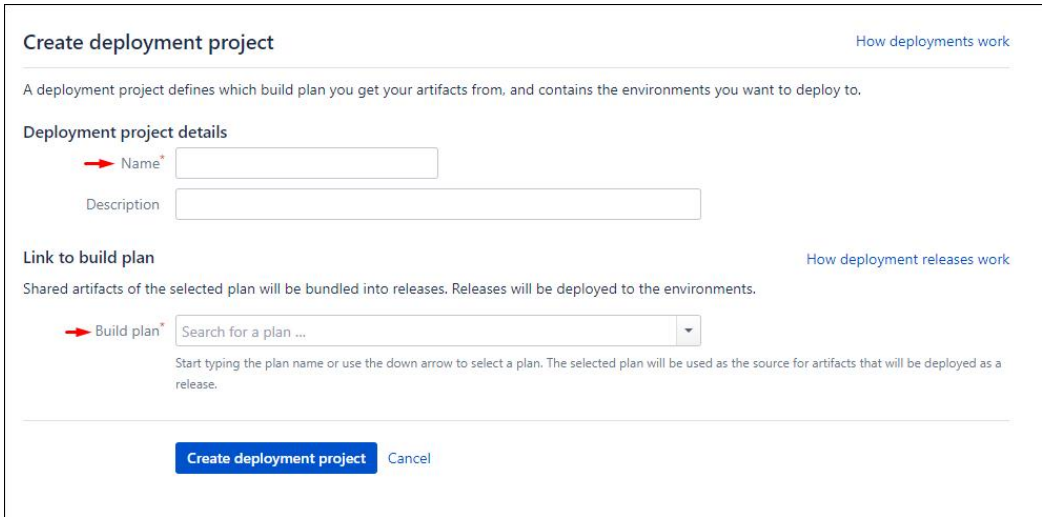

Figura 14. Criação de um plano de Deploy

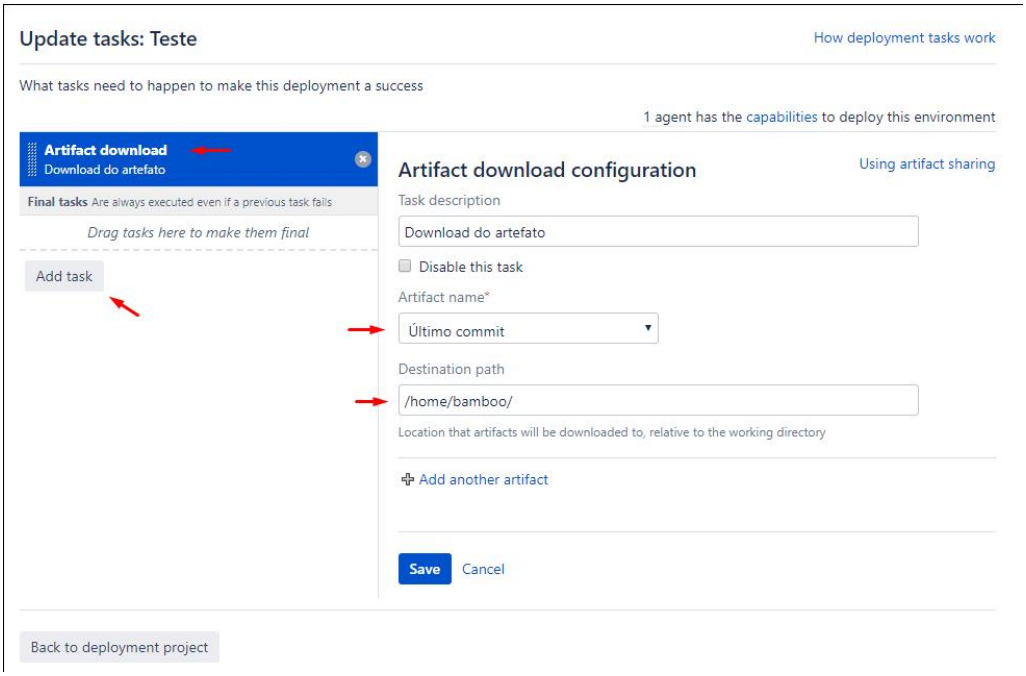

Figura 15. Configuração da tarefa de deploy

O processo citado acima foi feito implementando as configurações mais simples disponibilizadas pelas ferramentas utilizadas. O resultado obtido ao final do processo é a geração diária de um arquivo .JAR com as atualizações de código do sistema. O arquivo tem o objetivo de realizar a extração, transformação e carga de dados para a geração dos indicadores gerencias do Banco Alimentario de La Plata.

5. Considerações Finais

A área de DevOps contempla várias áreas de uma organização. Seu principal objetivo é automatizar o processo de um software para que as suas entregas sejam mais rápidas e livres de erros. Para que um processo DevOps seja bem executado, é extremamente importante que a equipe de desenvolvimento e de operações tenham um bom relacionamento uma com a outra.

Existem muitas ferramentas que auxiliam na implementação de um processo DevOps em uma organização. Em cada etapa do processo, diversas ferramentas podem ser utilizadas para resolver um mesmo problema. Determinar quais ferramentas serão utilizadas é importante. Por isso é recomendado que um estudo das ferramentas disponíveis no mercado seja feito com o objetivo de identificar quais atendem melhor as necessidades do projeto a ser criado.

O estudo de caso apresentado na unidade 4 possibilitou a implementação de um processo DevOps até a etapa de *deploy* de forma simples. A fácil implementação da integração continua e da entrega continua disponibilizadas pela ferramenta estudada, foram essenciais para conseguir desenvolver um processo simples.

Futuramente pretende-se estudar e implementar ferramentas para ajudar na análise dos estágios de operações e monitoramento, e aprofundar nas etapas de *build* e *deploy* pois, existem outras configurações e opções que podem ser exploradas, tais como a criação de dependências no plano de build, configuração de permissões de acesso, configuração de notificações do processo, possibilidade de configurar um ambiente Docker, a aplicação de testes que ocorrerão na etapa de *build* além da possibilidade de expandir o processo para outras *branches*.

Referências

4LINUX. Diferenças entre Integração, deploy e entrega contínua. Disponível em: https://www.4linux.com.br/diferencas-entre-integracao-deploy-e-entrega-continua. Acesso em 05 de junho de 2019a.

4LINUX. O que é Infraestrutura Ágil. Disponível em: https://www.4linux.com.br/o-que-e-infraestrutura-agil. Acesso em 25 de junho de 2019b.

ALMEIDA, A. ; SILVA, Evaldo de Oliveira ; MARTINS, D. . Automação do Processo de Implantação de Software. In: Escola Regional de Informática SBC Mato Grosso, 2014, Barra dos Graças. Anais da V Escola Regional de Informática de Mato Grosso - 2014. Barra do Garças: Título, 2014. v. 5. p. 135-140.

AMAZON. O que é DevOps? Disponível em: https://aws.amazon.com/pt/devops/what-is-devops/. Acesso em 05 de junho de 2019 a.

AMAZON. O que significa integração contínua? Disponível em: https://aws.amazon.com/pt/devops/continuous-integration. Acesso em 05 de junho

de 2019b .BAMBOO. Compile, teste e implemente. Disponível em: https://br.atlassian.com/software/bamboo. Acesso em 30 de maio de 2019.

BRAGA, Filipe Antônio Motta. Um panorama sobre o uso de práticas DevOps nas indústrias de software. 2015. Dissertação de Mestrado. Universidade Federal de Pernambuco.

CESJF. CES/JF faz Parceria com ONG na Argentina Disponível em: https://www.cesjf.br/sistemas-de-informacao-noticias/4597-ces-jf-faz-parceria-com-ong-na-argentina.html. Acesso em 24 de junho de 2019.

CONFLUENCE. O Confluence é um espaço de trabalho aberto e compartilhado. Disponível em: https://br.atlassian.com/software/confluence. Acesso em 30 de maio de 2019.

IBM. DevOps Entrega contínua de inovações promovidas por software. Disponível em: https://www.ibm.com/developerworks/br/devops/feedback.html. Acesso em 05 de junho de 2019.

JIRA. A ferramenta número 1 de desenvolvimento de software usada por equipes ágeis. Disponível em: https://br.atlassian.com/software/jira. Acesso em 30 de maio de 2019.

MICROFOCUS. Professional Services Model Offices Real-life use case modeling based on IT4IT™ to accelerate your deployment and time to value. Disponível em: https://www.microfocus.com/en-us/services/use-case-modeling. Acesso em 09 de junho de 2019.

MICROSOFT. Visão Geral da Tecnologia de DevOps. Disponível em: https://azure.microsoft.com/pt-br/overview/devops/. Acesso em 10 de junho de 2019.