ENGENHARIA DE SOFTWARE

AULA 6

Prof. Alex Mateus Porn

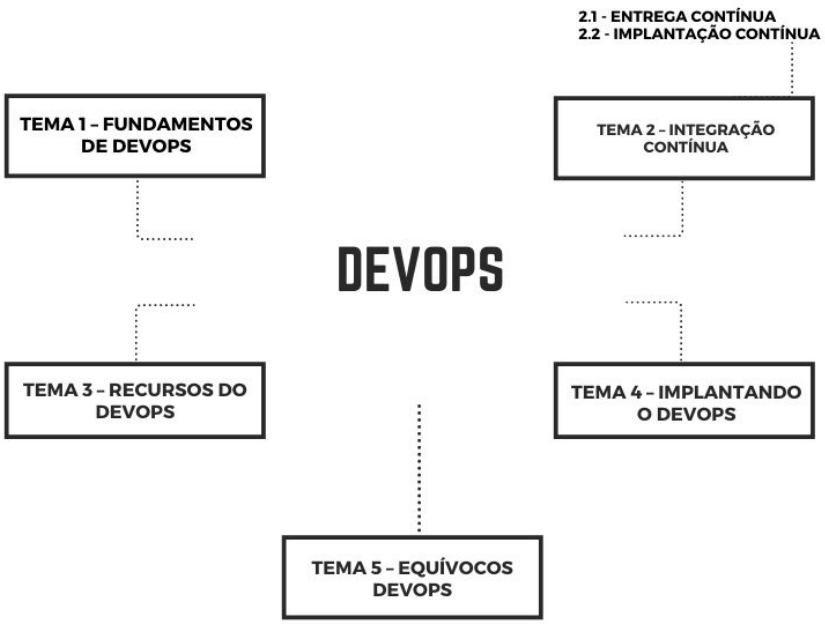
CONVERSA INICIAL

Olá! Nesta aula estudaremos os conceitos da cultura de desenvolvimento e operações (DevOps), suas características, critérios de implementação, desenvolvimento integrado, benefícios e mitos. O termo *DevOps*, como veremos com mais detalhes durante nossa aula, refere-se pois à junção de dois conceitos: desenvolvimento (*Dev*) e operações (*Ops*) e muito mais à origem do DevOps do que propriamente a sua forma de implementação – o termo pode dar uma falsa impressão de se tratar de um novo tipo de tecnologia, ferramenta ou método de desenvolvimento. Porém, como veremos adiante DevOps é uma cultura que deve ser implementada e trabalhada, ao longo do tempo, em uma organização.

Várias atividades são desencadeadas com a implantação da cultura DevOps, geralmente vinculadas à integração e ao trabalho em equipe, como melhoria contínua de processos, integração contínua, implementação contínua e entrega contínua. Tais atividades estão associadas a todos os conteúdos que estudamos desde a nossa primeira aula, principalmente a aplicação de métodos ágeis de desenvolvimento. Nesta aula, perceberemos que todos os conteúdos já estudados, como processos de software, testes de software, projeto e arquitetura de software, entre outros, são aplicados no DevOps, de forma colaborativa.

Ao longo desta aula serão trabalhados os conteúdos discriminados na Figura 1.

Figura 1 – Temas e subtemas da aula



TEMA 1 – FUNDAMENTOS DA CULTURA DEVOPS

O termo *DevOps*, conforme apresentam Muniz et al. (2020), refere-se a dois termos em inglês que identificam as equipes envolvidas nas atividades de construção e implantação de softwares. *Dev* remete a *development* (*desenvolvimento*) e *Ops*, a *operations* (*operações*). Portanto, DevOps é a união de pessoas, processos e tecnologias para fornecer continuamente valor aos clientes.

Ainda de acordo com Muniz et al. (2020), DevOps refere-se a:

- **Desenvolvimento (*Dev*)**: equipe responsável pela identificação dos requisitos com o cliente, pela análise, pelo projeto, pela codificação e pelos testes.
- **Operações (*Ops*)**: equipe responsável pela implantação em produção, pelo monitoramento e pela solução de incidentes e problemas.

Cabe destacar aqui a definição da cultura DevOps, também apresentada por Muniz et al. (2020):

> DevOps é uma cultura fortemente colaborativa entre as equipes de Desenvolvimento e Operações para entregar o software funcionando em produção de forma ágil, segura e estável. Mais do que um conceito, é importante destacar que DevOps é uma jornada de aproximação entre as pessoas com ações práticas de automação para acelerar as implantações com qualidade, considerando o ponto de vista de todos os envolvidos, a tão falada empatia.

Nesse contexto, Davis e Daniels (2016) apresentam os quatro pilares da cultura DevOps para os quais qualquer equipe ou organização que pretenda implementá-la precisará despender tempo e recursos:

1. **Colaboração**: significa construir um resultado específico com interações de pessoas com diferentes experiências e um propósito comum.
2. **Afinidade**: propõe a construção de relações interdependentes fortes entre os times interfuncionais, para que todos vejam sentido ao navegarem por objetivos organizacionais complementares e sintam-se interessados naturalmente pelo sentimento de empatia e aprendizagem contínua.
3. **Ferramentas**: funcionam como um acelerador, impulsionando a mudança com base na cultura atual.

4. **Escala**: esse dimensionamento leva em conta como os outros três pilares podem ser aplicados à medida que as organizações crescem, amadurecem e até encolhem, considerando questões técnicas e culturais.

Com base nesses quatro pilares, podemos observar que a cultura DevOps é implantada de forma singular em cada organização, não existindo um caminho único a ser seguido para a sua implantação, que sirva a todas as organizações. A implantação do DevOps permite que funções anteriormente isoladas, como desenvolvimento, operações de tecnologia da informação (TI), engenharia da qualidade e segurança, atuem de forma coordenada e colaborativa para gerar produtos melhores e mais confiáveis, na organização.

Ao adotar a cultura DevOps em conjunto com as práticas e ferramentas de DevOps, as equipes de trabalho ganham a capacidade de responder melhor às necessidades dos clientes, aumentar a confiança nos aplicativos que constroem e cumprir as metas empresariais mais rapidamente. Com base nessa abordagem, o DevOps influencia o ciclo de vida de um aplicativo em todas as fases do planejamento, do desenvolvimento, da entrega e da operação, conforme destacam Soares e Silveira ([S.d.]):

- **Planejamento**: as equipes DevOps idealizam, definem e descrevem os recursos e as funcionalidades dos aplicativos e sistemas que estão construindo. Elas acompanham o seu progresso em níveis altos e baixos de granularidade, desde tarefas de produto único até tarefas que abrangem portfólios de vários produtos. Exemplos: criar lista de pendências, acompanhar *bugs*, gerenciar o desenvolvimento de software ágil com Scrum, usar quadros Kanban e visualizar o progresso com *dashboards*.

- **Desenvolvimento**: inclui todos os aspectos da codificação – gravação, teste, revisão e integração do código pelos membros da equipe –, bem como a compilação do código em artefatos de compilação, que podem ser implementados em vários ambientes. As equipes DevOps buscam inovar rapidamente sem sacrificar a qualidade, a estabilidade e a produtividade. Para fazer isso, elas usam ferramentas altamente produtivas, automatizam etapas elementares e manuais e iteram em pequenos incrementos, por meio de testes automatizados e integração contínua.

- **Entrega**: é o processo de implantação de aplicativos nos ambientes de produção, de maneira consistente e confiável. A fase de entrega também inclui a implantação e a configuração da infraestrutura fundamental totalmente governada, que compõem aqueles ambientes. Na fase de entrega, as equipes definem um processo de gerenciamento de versão com estágios claros de aprovação manual. Elas também estabelecem portões automatizados que movem os aplicativos entre os estágios, até que sejam disponibilizados aos clientes. A automação desses processos os torna escalonáveis, repetíveis e controlados. Dessa forma, as equipes que praticam o DevOps com frequência podem atuar e entregar projetos com facilidade, confiança e tranquilidade.
- **Operação**: envolve manter, monitorar e solucionar problemas de aplicativos em ambientes de produção. Ao adotar as práticas DevOps, as equipes trabalham para garantir a confiabilidade do sistema, a alta disponibilidade e o objetivo de tempo de inatividade igual a zero, reforçando a segurança e a governança. As equipes DevOps buscam identificar os problemas antes que eles afetem a experiência do cliente e mitigar os problemas rapidamente, quando ocorrem. Manter esse nível de vigilância requer telemetria avançada, alertas acionáveis e visibilidade total dos aplicativos e do sistema subjacente.

Nesse contexto, conforme abordam Muniz et al. (2020), o DevOps pode ser considerado uma continuação natural dos métodos ágeis, conforme já estudamos, visando entregar valor ao cliente com foco em adaptabilidade e aprendizado contínuo. Nessa perspectiva, Muniz et al. (2020) ainda destacam que “Um dos benefícios da adoção de DevOps com métodos ágeis é a mobilização de todas as equipes que participam do fluxo de valor em uma abordagem ponta a ponta, desde o levantamento de requisitos até a entrega do software no ambiente de produção.”

Nesse contexto, de acordo com Sharma e Coyne (2017, p. 6), o movimento DevOps produziu vários princípios que evoluíram ao longo do tempo e ainda estão evoluindo. Vários fornecedores de soluções desenvolveram suas próprias variantes dessa cultura. Todos esses princípios, no entanto, têm uma abordagem holística e organizações de todos os tamanhos podem adotá-los. Esses princípios são:

- **Desenvolver e testar sistemas semelhantes ao de produção**: o objetivo desse princípio é permitir que as equipes de desenvolvimento e de garantia da qualidade desenvolvam e testem sistemas que se comportem como o sistema de produção, para que possam ver como o aplicativo se comporta e funciona bem antes de estar pronto para implantação.
- **Implantar o aplicativo com processos confiáveis e repetíveis**: esse princípio propicia que o desenvolvimento e as operações suportem um processo de desenvolvimento de software ágil em todo o seu caminho até a produção. A automação é essencial para criar processos iterativos, frequentes, repetíveis e confiáveis; portanto, a organização deve criar uma segmentação de entrega que permita implantação e testes contínuos e automatizados.
- **Monitorar e validar a qualidade operacional**: esse princípio propõe monitoramento desde o início do ciclo de vida do projeto, exigindo que o teste automatizado seja feito logo no início e frequentemente no ciclo de vida do aplicativo para monitorar as suas características funcionais e não funcionais. Sempre que um aplicativo é implantado e testado, as suas métricas de qualidade devem ser capturadas e analisadas. O monitoramento frequente fornece um aviso prévio sobre problemas operacionais e de qualidade que podem ocorrer na produção.
- **Amplificar os *loops* de feedback**: um objetivo do DevOps é permitir que as organizações reajam e façam mudanças mais rapidamente. Na entrega de softwares, essa meta requer que uma organização obtenha feedback rápido e aprenda rapidamente com cada ação que realiza. Esse princípio exige que as organizações criem canais de comunicação que permitam a todas as partes interessadas acessar e agir de acordo com o feedback.

De acordo com os fundamentos da cultura DevOps, fica nítido que ela vai além de ferramentas específicas ou métodos padronizados para sua implementação, tal como estudamos até o momento com modelos de processos, métodos ágeis, arquiteturas de projetos de software, técnicas de teste, entre outros elementos. As características da cultura DevOps, tais quais integração contínua, entrega contínua, melhoria contínua, trabalho em equipe, assim como utilização de metodologias ágeis, são um processo de implantação e

amadurecimento do conhecimento durante todo o período, não sendo possível estipular um prazo final para determinar quando o DevOps está concluído.

TEMA 2 – INTEGRAÇÃO CONTÍNUA

A integração contínua é um requisito fundamental para o DevOps. Essa técnica foi desenvolvida em 1999 como parte integrante do método ágil programação extrema (XP), que já estudamos. Em sua abordagem, Fowler (2006, tradução nossa) destaca que:

> A integração contínua é uma prática do desenvolvimento de software em que cada participante do time integra seu trabalho pelo menos uma vez no dia. Cada integração é verificada por um *build* automatizado para detectar erros de imediato e permitir que as atividades de desenvolvimento de software tenham mais qualidade e agilidade.

Com base na afirmação de Fowler, Muniz et al. (2020) ressaltam que a integração contínua estabelece que o código seja compilado para cada mudança e execute testes automatizados minimamente confiáveis, ao contrário dos métodos tradicionais, que adiam a integração até o final do desenvolvimento.

Em sua abordagem, Fowler (2006) propõe 11 princípios que possibilitam que a integração contínua seja mais efetiva, durante o seu desenvolvimento:

1. manter um repositório de origem único;
2. automatizar a versão;
3. fazer o autoteste de construção (testes unitários);
4. todos devem fazer *commit* pelo menos uma vez por dia;
5. todo *commit* deve ser centralizado em uma máquina de integração;
6. corrigir quebra de código imediatamente;
7. manter a construção rápida;
8. testar em um ambiente que seja o mais próximo possível do ambiente de produção;
9. garantir que qualquer um possa obter o executável mais recente;
10. possibilitar que todo mundo veja o que está acontecendo;
11. automatizar a implantação.

Já para Humble e Farley (2014), a integração contínua somente é praticada quando todas as ações a seguir são realizadas:

a. O time integra o código pelo menos uma vez por dia, em um único tronco.

b. A segmentação de implementação é iniciada automaticamente a cada mudança de código e executa validações, análise estática de padrões de codificação e testes.
c. Quando uma versão falha, na maioria das vezes é corrigida em até 10 minutos.

Para facilitar nossa compreensão sobre como implantar a integração contínua no desenvolvimento de softwares seguindo os 11 princípios de Fowler (2006) e as 3 ações necessárias de Humble e Farley (2014), vamos destacar aqui o exemplo de integração no desenvolvimento de uma pequena parte de um software, proposto por Fowler (2006).

a. Suponhamos que será implementada uma pequena função de um sistema.
b. O primeiro passo para isso consiste em se obter uma cópia do código-fonte atual do sistema e colocá-lo na máquina de desenvolvimento (exemplo: obter uma cópia do GitHub).
c. O próximo passo consiste no desenvolvimento e em testes da nova função.
d. Finalizada a terceira etapa, é necessário criar uma nova versão automatizada (*build*), compilada e transformada em uma versão executável (essa etapa somente é finalizada se em todos os testes não forem encontrados erros).
e. Com a nova versão pronta, o próximo passo consiste em adicioná-la ao repositório de onde foi pega a versão original do código-fonte, na primeira etapa (se houver mudanças feitas por outros desenvolvedores no repositório, primeiro é necessário atualizar a versão com as alterações realizadas pelos outros desenvolvedores e criar um novo *build*; se houver conflitos com as alterações dos outros desenvolvedores, serão exibidos erros tanto na compilação quanto nos testes, sendo, portanto, necessárias a correção dos erros e a repetição do processo até que o novo *build* esteja sincronizada com a versão principal do software.
f. Com o novo *build* criado e sincronizado com a versão principal do software, o próximo passo consiste na realização do *commit* e na atualização do repositório com a versão mais atual do sistema.

O gerenciamento da configuração e controle de versão, tal qual estudamos anteriormente, quando abordamos o uso da ferramenta Git e do repositório GitHub, assim como dos métodos ágeis, são fundamentais na implantação da integração contínua no DevOps, principalmente quando os indivíduos de uma equipe atuam em lugares distintos.

2.1 Entrega contínua

Muniz et al. (2020) destacam que a entrega contínua é uma evolução natural quando existe o interesse de expandir os benefícios da automação dos testes e do feedback imediato para os próximos estágios que não são cobertos pela integração contínua.

Com o intuito de definir o conceito de entrega contínua, Humble e Farley (2014) afirmam que “A entrega contínua é a capacidade de disponibilizar mudanças de forma segura e rápida garantindo que o código esteja sempre pronto para implantação, mesmo diante de milhares de desenvolvedores fazendo alterações diariamente.”

2.2 Implantação contínua

Conforme Muniz et al. (2020), a implantação contínua é a evolução natural da entrega e consiste na implantação automática em produção após a execução com sucesso dos testes automatizados e das validações previstas.

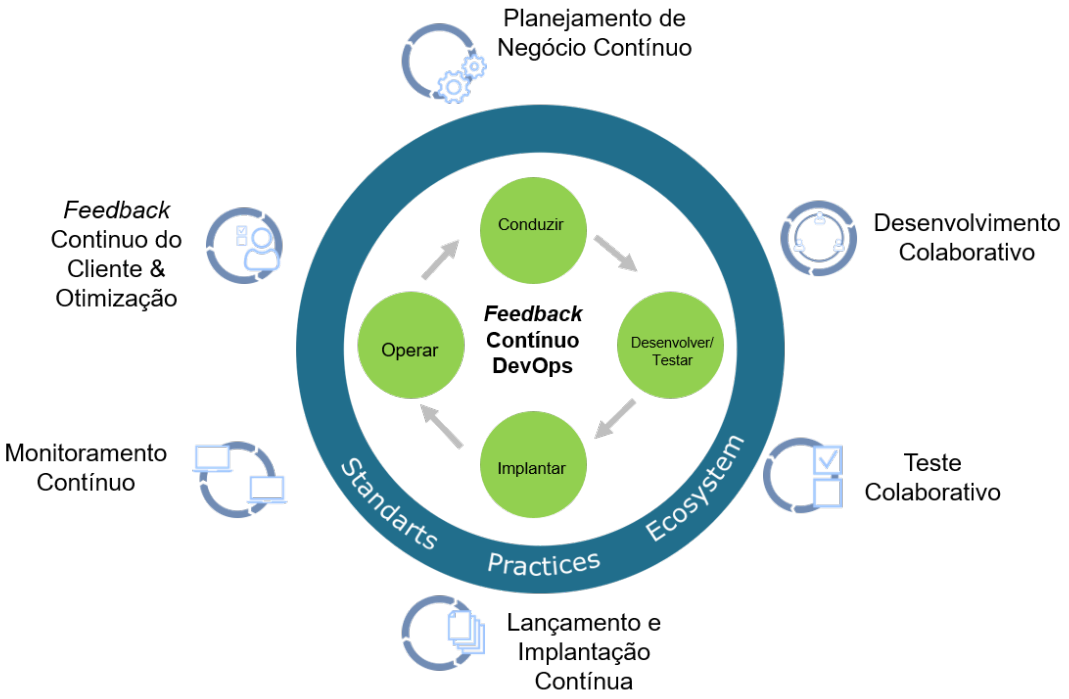
Diante dessa abordagem, Muniz et al. (2020) ainda afirmam que:

> A implantação contínua é usada principalmente para aplicações web e aplicativos, mas pode ser experimentada em qualquer tipo de tecnologia. A ideia é disponibilizar pequenas mudanças em produção com a maior frequência possível. É também possível replicar os ambientes de forma confiável, assim os desenvolvedores podem ensaiar a implantação do código em produção em um ambiente onde a infraestrutura já é muito parecida, e o mais próximo possível, do ambiente de produção.

TEMA 3 – RECURSOS DO DEVOPS

Os recursos que compõem o DevOps são um amplo conjunto que abrange o ciclo de vida da entrega do software. Na Figura 2 é apresentada a arquitetura de referência DevOps, na perspectiva dos principais recursos que ele pretende fornecer.

Figura 2 – Arquitetura de referência DevOps



Fonte: Elaborado com base em Sharma; Coyne, 2017, p. 10.

Conforme observado na Figura 2, Sharma e Coyne (2017) apresentam que o DevOps propõe quatro caminhos de adoção a seguir:

1. **Conduzir**: esse caminho foca em estabelecer os objetivos do negócio e ajustá-los com base no feedback dos clientes. Inclui a prática de **planejamento contínuo do negócio**.
2. **Desenvolver/testar**: esse caminho envolve duas práticas:

a. **desenvolvimento colaborativo**: permite que os profissionais trabalhem juntos, fornecendo um conjunto comum de práticas e uma plataforma comum que eles podem usar para criar e entregar um software; a atividade central do desenvolvimento colaborativo é a integração contínua;

b. **teste contínuo**: significa testar mais cedo e continuamente ao longo do ciclo de vida do software, o que resulta em custos reduzidos, ciclos de teste mais curtos e feedback contínuo sobre a qualidade do produto.

3. **Implantar**: esse caminho abrange a prática da liberação e da implantação contínuas, que têm como objetivo lançar novos recursos para clientes e usuários o mais rápido possível.

4. **Operar**: esse caminho inclui duas práticas que permitem às empresas monitorar o desempenho dos aplicativos lançados em produção e receber feedback dos clientes:

a. **monitoramento contínuo**: fornece dados e métricas para o pessoal de operações, controle de qualidade, desenvolvimento, linhas de negócios e outras partes interessadas, sobre aplicativos em diferentes estágios do seu ciclo de entrega;

b. **feedback e otimização contínua do cliente**: esse feedback permite que diferentes partes interessadas tomem as ações apropriadas para melhorar os aplicativos e aprimorar a experiência do cliente; esse ciclo de feedback contínuo é um componente essencial do DevOps, propiciando que as empresas sejam mais ágeis e responsivas às necessidades do cliente.

TEMA 4 – IMPLANTANDO O DEVOPS

Conforme destacam Sharma e Coyne (2017, p. 15, tradução nossa):

> Embora o nome DevOps sugira recursos baseados em desenvolvimento e operações, DevOps é um recurso corporativo que abrange todas as partes interessadas em uma organização, incluindo proprietários de negócios, arquitetura, *design*, desenvolvimento, garantia de qualidade, operações, segurança, parceiros e fornecedores. A exclusão de qualquer parte interessada – interna ou externa – leva a uma implementação incompleta do DevOps.

Vale destacar aqui que o DevOps não é um objetivo, mas sim uma cultura para ajudar a atingir os objetivos organizacionais. Conforme abordam Sharma e Coyne (2017, p. 16), a primeira tarefa na criação de uma cultura é seguirem todos numa mesma direção, identificando objetivos de negócios comuns à equipe e à organização como um todo. Quando as pessoas sabem com qual objetivo comum estão trabalhando e como seu progresso em direção a esse objetivo será medido, existem menos desafios para equipes ou profissionais que têm suas próprias prioridades.

Nesse contexto, Sharma e Coyne (2017, p. 16) afirmam que as maiores fontes de ineficiências no segmento de entregas de software foram categorizadas da seguinte forma:

- **sobrecarga desnecessária**: necessidade de comunicar várias vezes a mesma informação e conhecimento;

- **retrabalho desnecessário**: defeitos são descobertos em testes ou na produção, forçando a equipe a voltar ao desenvolvimento;
- **superprodução**: funcionalidades desenvolvidas que não foram requeridas.

Com base na análise dessas ineficiências apresentadas e do foco nas pessoas para a implantação da cultura DevOps, percebe-se que essa cultura gira em torno das pessoas. Conforme abordam Sharma e Coyne (2017, p. 17), uma organização pode adotar os processos mais eficientes ou as ferramentas mais automatizadas possíveis, mas eles serão inúteis se não tiverem a adesão das pessoas que eventualmente devam executar esses processos e usar essas ferramentas.

Construir uma cultura DevOps, portanto, é o cerne da adoção do DevOps. Essa cultura é caracterizada por um alto grau de colaboração entre funções, pelo foco nos negócios em vez de nos objetivos departamentais, pela confiança e pelo alto valor atribuídos ao aprendizado por meio da experimentação.

Conforme Sharma e Coyne (2017, p. 17, tradução nossa): “Construir uma cultura não é como adotar um processo ou uma ferramenta. Requer engenharia social de equipes de pessoas, cada uma com predisposições, experiências e preconceitos únicos. Essa diversidade pode tornar a construção de uma cultura desafiadora e difícil.”

Práticas de desenvolvimento ágil como o *scrum* estão no centro do DevOps e elas podem ser aproveitadas para ajudar a adotar essa cultura.

> Construir uma cultura DevOps requer que os líderes da organização trabalhem com suas equipes para criar um ambiente e uma cultura de colaboração e compartilhamento. Os líderes devem remover quaisquer barreiras autoimpostas à cooperação. As medições típicas recompensam as equipes de operações pelo tempo de atividade e estabilidade e recompensam os desenvolvedores pelos novos recursos entregues, mas eles colocam esses grupos uns contra os outros. As operações sabem que a melhor proteção para a produção é não aceitar mudanças, por exemplo, e o desenvolvimento tem poucos incentivos para se concentrar na qualidade. Substitua essas medições pela responsabilidade compartilhada de fornecer novos recursos com rapidez e segurança. (Sharma; Coyne, 2017, p. 17, tradução nossa)

Ao se optar por uma equipe DevOps, o objetivo mais importante é garantir que ela funcione como um centro de excelência que facilite a colaboração sem adicionar uma nova camada de burocracia ou ela se tornar a equipe responsável por resolver todos os problemas de uma empresa. Conforme destacado por

Sharma e Coyne (2017), as seis principais técnicas específicas necessárias à implantação do DevOps são:

1. **Melhoria contínua**: visa atingir, ininterruptamente, resultados cada vez melhores. De acordo com Santos (2017), para que essa prática aconteça de fato, alguns princípios básicos devem ser contemplados, como:

- **ter foco no que se deseja melhorar**: é preciso saber identificar o que precisa ser melhorado;
- **estabelecer métricas para medir o desempenho**: somente é possível melhorar aquilo que se pode medir – portanto, é necessário estabelecer uma métrica para medir a qualidade do processo;
- **padronizar tarefas**: para haver melhorias, deve existir um padrão na execução das tarefas.

2. **Planejamento de liberação**: é uma função impulsionada pelas necessidades comerciais de se oferecer recursos, aos clientes, em prazos menores e exatos, em que o aproveitamento de práticas ágeis auxilia no planejamento desses prazos com mais frequência, permitindo maior foco na qualidade.
3. **Integração contínua**: conforme abordamos no Tema 2, a integração contínua agrega valor ao DevOps, permitindo que grandes equipes de desenvolvedores, trabalhando em componentes de tecnologia em vários locais, forneçam softwares de maneira ágil. A integração contínua, portanto, reduz o risco e identifica os problemas no início do ciclo de vida do desenvolvimento do software.
4. **Entrega contínua**: a integração contínua leva naturalmente à prática da entrega contínua. Normalmente, essa é a parte mais crítica da adoção do DevOps. Para muitos profissionais, o DevOps limita-se à entrega contínua; portanto, a maioria das ferramentas promovidas como ferramentas DevOps abordam apenas esse processo.
5. **Teste contínuo**: consideram-se três tipos de testes que possibilitam o teste contínuo:

- teste de provisionamento do ambiente e configuração;
- teste de gerenciamento de dados;
- teste de integração, função, desempenho e segurança – o teste de provisionamento do ambiente e o teste de gerenciamento de dados são desafios mais importantes para projetos que usam metodologias ágeis e

praticam integração contínua do que para projetos que usam metodologia em cascata e testam-nos apenas uma vez a cada poucos meses. Da mesma forma, os requisitos de teste de função e desempenho para aplicativos complexos com componentes que têm ciclos de entrega separados são diferentes daqueles de aplicativos mais simples.

6. **Monitoramento e feedback contínuos**: o feedback do cliente aparece de diferentes formas, como por meio de solicitações realizadas por clientes, reclamações informais e classificações de produtos. O feedback também resulta dos dados de monitoramento. Esses dados podem provir dos servidores que executam o aplicativo ou de ferramentas de métricas incorporadas ao aplicativo.

TEMA 5 – EQUÍVOCOS DEVOPS

Muitos equívocos comuns surgem diante de várias dificuldades que as equipes apresentam para esclarecer e articular crenças e valores sobre o DevOps. Nesse contexto, Davis e Daniels (2016) apresentam alguns dos problemas mais comuns que os profissionais enfrentam ao tentar estabelecer essa cultura em suas organizações, baseados nas crenças de que:

- **DevOps é apenas para desenvolvedores e administradores de sistemas**: embora o nome esteja relacionado a desenvolvimento e operações, como já explicado, ele refere-se mais à origem do movimento do que a uma definição estrita dele. Os conceitos e ideias do DevOps abrangem todas as funções existentes em uma organização. Não há uma lista definitiva de quais equipes ou indivíduos devem ser envolvidos, assim como não existe uma maneira única de “fazer” DevOps.
- **DevOps é um time**: criar uma equipe chamada *DevOps* ou renomear uma equipe existente não é necessário nem suficiente para criar a cultura. Se a organização estiver em um estado em que as equipes de desenvolvimento e operações não possam se comunicar, uma equipe adicional provavelmente causará mais problemas de comunicação.
- **DevOps é uma função**: esse equívoco surgiu com uma possível especificação do cargo de engenheiro DevOps. Normalmente, não faz muito sentido ter um diretor de DevOps ou alguma outra posição que coloque uma pessoa no comando. Devops é, em sua essência, um

movimento cultural, e suas ideias e princípios precisam ser usados em organizações em seu todo, para serem eficazes.

- **DevOps é relevante apenas para *startups web***: o produto principal das empresas baseadas na *web* é o aplicativo que o usuário vê ao navegar no *site* da empresa. É fácil ver por que o DevOps faz sentido para essas empresas, pois o movimento ajuda a quebrar as barreiras que podem impedir o desenvolvimento e a implantação desses aplicativos. Porém, as ideias culturais do DevOps podem ser aplicadas mesmo em empresas e organizações governamentais.
- **É necessária uma certificação DevOps**: DevOps é sobre cultura – como certificar cultura? Não existe um exame de 60 minutos que possa certificar a eficácia com que uma pessoa se comunica com outras, como as equipes trabalham juntas ou como uma organização aprende. As certificações fazem sentido quando aplicadas a tecnologias específicas que requerem um alto nível de conhecimento para serem usadas, como softwares ou hardwares específicos. Os exames de certificação testam o conhecimento em que há respostas certas ou erradas, o que o DevOps geralmente não tem. O que funciona melhor para uma empresa não será necessariamente ideal para outra.
- **DevOps significa fazer todo o trabalho com metade da equipe**: é um equívoco comum pensar que DevOps é uma maneira de obter um desenvolvedor de softwares e um administrador de sistemas em uma única pessoa. Durante os estágios iniciais de consolidação, é comum uma empresa se beneficiar do trabalho de desenvolvedores que entendam o suficiente sobre operações para lidar também com implantações. Depois que uma organização ultrapassa o ponto em que cada funcionário deve ter vários cargos por pura necessidade, é impossivel esperar que uma pessoa ocupe duas funções que lhe exijam dedicação em tempo integral.
- **Existe um jeito certo e um errado de aplicar o DevOps**: só porque uma empresa apresenta sua estratégia de DevOps como bem-sucedida não significa que esses mesmos processos serão a maneira certa de fazer DevOps em todos os ambientes. O DevOps incentiva o pensamento crítico sobre processos, ferramentas e práticas. Ser uma organização que aprende exige questionar e iterar os processos, não aceitando as coisas como a única maneira verdadeira ou como as coisas sempre foram feitas.

- **São necessárias “x” semanas para implantar o DevOps**: DevOps é um processo contínuo. É a jornada e não o destino. Algumas partes dele terão um ponto final fixo, como configurar um sistema de gerenciamento de configuração e certificar-se de que todos os servidores da empresa estão sendo gerenciados por ele, mas a manutenção contínua, o desenvolvimento e o uso do gerenciamento de configuração se perpetuarão.
- **DevOps é todo o conjunto de ferramentas**: embora as ferramentas sejam valiosas, o DevOps não impõe nem exige nenhuma em particular. Em um ambiente, as ferramentas utilizadas fazem parte da cultura. Antes de decidir sobre uma mudança, cabe reconhecer as ferramentas no ambiente que fazem parte da cultura existente em uma organização, entender as experiências dos indivíduos com essas ferramentas e observar o que é semelhante e diferente entre as experiências dos outros. A tecnologia impacta a velocidade e as estruturas organizacionais. Fazer mudanças drásticas nas ferramentas, embora elas possam ter valor para um indivíduo ou equipe, pode acarretar um custo que desacelere a organização como um todo.
- **DevOps é sobre automação**: muitas inovações em ferramentas adjacentes ao DevOps ajudam a codificar esse entendimento, preenchendo as lacunas entre as equipes e aumentando a velocidade por meio da automação. Se houver tarefas repetitivas que possam ser automatizadas, essa automação ajudará a se alcançar mais eficiência. A automação pode tornar o trabalho mais rápido; mas, para ele ser mais eficaz, também se deve aumentar a transparência, a colaboração e a compreensão.
- **DevOps é passageiro**: como DevOps não é específico a uma tecnologia, ferramenta ou processo, é menos provável que se torne obsoleto, ou seja, substituído por algo novo. Um movimento para melhorar a eficácia organizacional e a felicidade individual do funcionário pode ser absorvido pelo uso comum, mas não irá envelhecer. Uma das principais diferenças entre DevOps e metodologias como Itil e Agile é que estas têm definições estritas, com o contexto sendo mudado ao longo do tempo. DevOps, por outro lado, é um movimento definido por histórias e ideias de indivíduos,

equipes e organizações. São as conversas contínuas e a evolução dos processos e ideias que levam ao crescimento e à mudança.

FINALIZANDO

Essa foi nossa sexta e última aula, em que abordamos os conceitos da cultura DevOps. Nela, estudamos todos os conceitos para desenvolvimento dessa cultura em uma organização, mais precisamente relacionada ao desenvolvimento de softwares. Pudemos compreender que o DevOps não é uma tecnologia específica, que está vinculada a um tipo ou conjunto de ferramentas, à automatização de um processo, entre outros aspectos, mas sim uma cultura que faz uso de todos os conceitos da engenharia de software que estudamos nas aulas anteriores.

Aprendemos que o principal princípio DevOps está relacionado à integração contínua, que engloba tanto a entrega quanto a implantação e a melhoria contínua dos processos. Percebemos, após o Tema 3, ao estudar a arquitetura de referência DevOps, que essa cultura abrange todo o ciclo de vida do desenvolvimento de um software, com ênfase na integração contínua e no apoio a métodos ágeis.

Também vimos nesta aula que as principais fontes de ineficiência no desenvolvimento de softwares, que podem justificar a implementação do DevOps, estão relacionadas principalmente a retrabalho e produção desnecessária, normalmente ocasionados por falhas nos requisitos, que, por sua vez, podem estar associadas a sobrecarga desnecessária, ou seja, a ineficiência na comunicação.

Em contrapartida, finalizamos esta aula abordando os principais equívocos relacionados ao DevOps, ou seja, interpretações realizadas que podem dificultar o desenvolvimento dessa cultura em uma organização, muitas vezes pela realização equivocada de atividades desnecessárias, ou mesmo impedir o seu desenvolvimento por não se atingir resultados satisfatórios devido a falhas do seu processo de implementação.

REFERÊNCIAS

DAVIS, J.; DANIELS, K. **Effective DevOps**: Building a culture of collaboration, affinity, and tooling at scale. Sebastopol: O'Reilly Media, 2016.

FOWLER, M. Continuous Integration. **MartinFowler.com**, 1 maio 2006. Disponível em: <https://martinfowler.com/articles/continuousIntegration.html>. Acesso em: 17 fev. 2021.

HUMBLE, J.; FARLEY, D. **Entrega contínua**: como entregar software de forma rápida e confiável. Porto Alegre: Bookman, 2014.

MUNIZ, A. et al. **Jornada DevOps**: unindo cultura ágil, *lean* e tecnologia para entrega de software com qualidade. 2. ed. Rio de Janeiro: Brasport, 2020.

SANTOS, V. F. M. Melhoria contínua: o que é? Como implementá-la? **FM2S**, 16 mar. 2017. Disponível em: <https://www.fm2s.com.br/melhoria-continua>. Acesso em: 17 fev. 2021.

SHARMA, S.; COYNE, B. **DevOps for Dummies**. 3. ed. Hoboken: John Wiley & Sons, 2017.

SOARES, V. D.; SILVEIRA, C. DevOps do zero à produção. **Coffee and It**, [S.d.]. Disponível em: <https://drive.google.com/file/d/161A3PBnbVH2N7Uu_hdbxLHqvFSP-EcUc/view>. Acesso em: 17 fev. 2021.