Programmer's®
Beyond IT

APOSTAR EM **QUALIDADE** VAI DIMINUIR O *TIME TO MARKET* DA SUA APLICAÇÃO

INTRODUÇÃO

Atualmente o segredo para o sucesso de uma nova ideia é lançar o produto, e ou, serviço antes que alguém chegue primeiro no mercado. Porém já ouviu aquele ditado que a pressa é inimiga da perfeição? Pois é, todo desenvolvedor sabe muito bem a importância que tem os testes e a qualidade do seu código para garantir o sucesso das aplicações digitais.

Mas e você, sabe o valor que isso tem para o seu negócio e como isso afeta o ***TIME TO MARKET*** do seu software?

É comum que as organizações na ânsia de colocar suas ideias no mercado acabem focando mais no tempo de entrega do que na qualidade delas. E priorizar durante todo o projeto – do começo até o fim – os testes e a qualidade do processo, não só acelera a comercialização do software como evita "gambiarras" que em curto prazo poderia gerar *bugs*, retrabalho e dificuldade na manutenção.

Neste e-book, vamos te ajudar a entender como apostar em **QUALIDADE** pode impulsionar a inovação e evitar erros que podem comprometer o sucesso do seu produto/serviço.

BOA LEITURA!

ÚLTIMA ETAPA, CERTO? ERRADO!

Quando falamos em desenvolvimento de software o produto nasce de uma ideia com o objetivo de resolver determinada necessidade. Mas da ideia até sua realização, existe um caminho que deve ser percorrido ao mesmo tempo que há a urgência de agregar valor ao negócio.

Com os métodos ágeis, como o Scrum, essa necessidade de colocar o produto rapidamente no mercado é possível graças aos ciclos curtos de desenvolvimento, as *sprints*, que proporcionam ao final de cada ciclo uma entrega da aplicação para uso.

E neste contexto de implementações rápidas e contínuas de valor, qualidade e testes são essenciais, pois sem isso, será gasto muito tempo com retrabalho ou as entregas terão uma frequência menor. Por isso, qualidade e testes devem ser efetuadas durante todo o desenvolvimento e não apenas na etapa final.

Vamos lá, imagine que em uma *sprint* apareça algo não previsto (e acredite isso vai acontecer!) e que acabe tomando mais tempo do que o planejado e que a qualidade e testes foram deixados para o fim do ciclo. Consequentemente isso diminuirá o período de testes e poderá prejudicar a qualidade do seu código. Consegue visualizar as falhas e problemas?

Portanto, no ágil a responsabilidade em garantir qualidade e testes durante todo o desenvolvimento passa a ser de **TODOS** do Time de Desenvolvimento e não apenas do *Quality Assurance* (QA). Afinal, a mentalidade dever ser: **prevenir *bugs* ao invés de achá-los**.

CULTURA DEVSECOPS

E como fazer com que todo o time e mais outras peças fundamentais no processo estejam cientes e presentes para que o software seja seguro, automatizado e integrado? Os métodos ágeis até mudaram a forma como se desenvolvia um software, mas o DevSecOps melhorou ainda mais o processo.

Segundo a Microsoft DevSecOps é a:

> união de pessoas, processos e tecnologias para fornecer continuamente valor ao usuário final.”

Ou seja, **PESSOAS** que precisam ter objetivos comuns para seguirem processos com foco na melhoria da aplicação. **TECNOLOGIAS** que auxiliam o Time de Desenvolvimento a colaborarem, aumentarem a produtividade e facilitarem a automação dos processos de ponta a ponta. E o mais importante **VALOR CONTÍNUO** que traz agilidade no *feedback* sobre o produto.

DevSecOps é uma cultura que gerencia os fluxos de trabalho de forma coordenada e colaborativa em busca de produtos melhores e mais confiáveis, entregas mais rápidas e usuários finais muito mais satisfeitos.

A execução operacional eficiente do ciclo DevSecOps durante a implementação vai garantir a continuidade e o aperfeiçoamento constante de toda a estruturação realizada: práticas, processos e automações que envolvem essa nova cultura organizacional. E prevenção ou antecipação de possíveis problemas com a responsabilidade compartilhada com todo o time, tornando as aplicações mais seguras.

Benefícios de implementar essa cultura:

VELOCIDADE: Atualizações mais rápidas, acelerando assim o tempo da comercialização do produto.

ESCALA: Opera e gerencia sistemas complexos ou dinâmicos com eficiência e risco reduzido.

CONFIABILIDADE: É possível acompanhar em tempo real o desempenho do software, podendo identificar eventuais erros. Mantendo a estabilidade e a confiabilidade do sistema.

SEGURANÇA: É uma temática importante que vem dando o que falar. Com o DevSecOps você pode definir e rastrear a conformidade em escala.

QUALIDADE ALÉM DOS *BUGS*

É importante ressaltar que o DevSecOps não é focado em *bugs*, mas sim em fornecer valor continuamente. Por isso, é preciso ir além disso.

Qualidade precisa ser mais que testar o software, encontrar bugs, documentá-los. definir a severidade deles e o quão rápido eles precisam ser corrigidos. Avaliar e validar o que foi desenvolvido é apenas uma das atividades que devem ser desenvolvidas neste processo que passa a ser focado em prevenção e não em falhas.

Abaixo listamos as principais etapas que exercemos na maioria dos projetos por aqui:

- Análise de Requisitos: Esta etapa trata-se do processo de entender e identificar as necessidades que o produto/serviço precisará atingir. Então, junto com o Product Owner (P.O.) se faz um levantamento de requisitos definindo, o escopo do projeto.

EXEMPLO: Imagine um e-commerce que precisa desenvolver a página de carrinho de compras os requisitos poderiam ser:

- Permitir o usuário aumentar a quantidade do(s) item(ns) adicionado(s);
- Permitir o usuário remover o(s) item(ns) adicionado(s);
- Permitir o usuário consultar prazo de entrega para o CEP informado;
- Permitir o usuário consultar valor de frete.

- Apoio na definição dos critérios de aceites: Também junto com o P.O. são escritos os critérios de aceites que precisam ser atingidos para atender os requisitos levantados na etapa anterior.

EXEMPLO: Seguindo a ideia do e-commerce e da página de carrinho de compras, os critérios de aceite poderiam ser:

- Usuário deve realizar a compra de um ou mais item;
- Usuário não deve realizar compra de mercadoria com quantidade <= 0;
- Usuário não deve realizar compra de mercadoria com quantidade > que o estoque disponível;
- Usuário deve conseguir consultar prazo de entrega/valor de frete.

- Identificar cenários: Durante uma *sprint* ocorrem algumas reuniões para planejar, pontuar as stories e identificar cenários que resumidamente é a etapa na qual o Time de Desenvolvimento compreende o que precisa ser feito e definir o que deve ser testado, pensando em diversas situações que a aplicação pode passar.

EXEMPLO: Em uma planning o P.O. explica como ele quer que funcione o carrinho de compras do seu e-commerce. Neste momento o Time de Desenvolvimento começa a identificar os cenários que neste caso poderiam ser:

- Colocar quantidade negativa de mercadorias;
- Quantidade maior que o estoque permitido e etc.

- Casos de testes: O próximo passo é documentar detalhadamente como serão feitos os testes que foram levantados em cada cenário. A ideia aqui é que qualquer um do Time de Desenvolvimento consiga ler e testar se tornando praticamente um guia.

EXEMPLO: Vamos pegar o cenário imaginado acima "Carrinho com mercadoria com quantidade negativa" e descrever o caso de teste dele:

Numeração	Passo para reproduzir	Comportamento esperado
1	Acesse o e-commerce	Página principal do e-commerce deve ser exibida
2	Por meio da lupa, pesquise uma mercadoria	A página deve ser atualizada com o resultado da pesquisa e a mercadoria deve ser apresentada
3	Clique no botão "Comprar" para esta mercadoria	A página deve ser redirecionada para a página de carrinho de compras
4	No campo de quantidade, digite quantidade < 0	Não deve ser possível selecionar quantidade < 0

BOAS PRÁTICAS = QUALIDADE DO CÓDIGO

Outro fator que irá te ajudar a tomar cuidado em evitar a produção de *bugs* é se preocupando com a qualidade do código que está sendo desenvolvido. É preciso apostar em códigos: limpos e bem escritos.

Atingir essa qualidade no código ajudará a prevenir retrabalho, conseguindo utilizar o tempo de desenvolvimento para trazer novas funcionalidades para o produto em vez de ter que perder tempo refazendo o código para conseguir que o software opere do jeito que já deveria estar funcionando.

Mas como efetivamente eu garanto a qualidade do código? Por meio de boas práticas e princípios que se aplicados corretamente tornam não só o código limpo, mas o software todo, com manutenção evolutiva e de fácil manuseio.

SOLID

é um acrônimo que carrega cinco princípios – que podem ser aplicados a qualquer linguagem de POO (Programação Orientada a Objetos) –, que facilita a manutenção, entendimento e organização do código.

Os princípios são: Single Responsibility Principle (SRP); Open/Closed Principle (OCP); Liskov Substituion Principle (LSP); Interface Segregation Principle (ISP) e Dependency Inversion Principle (DIP).

CLEAN CODE

são boas práticas para criar um código limpo que possui o objetivo de manter o código apenas com dados relevantes, nomes autoexplicativos para variáveis, métodos, classes e não deixar comentários desnecessários no código.

Uma vez aplicado facilita a escrita e leitura de um código.

PROCESSOS CONTÍNUOS

O conteúdo do tema sobre DevSecOps é bem extenso, mas vale abordarmos os focos que entendemos serem os fundamentais para a evolução da cultura. Como *Continuos Integration* (CI) e *Continuous Delivery* (CD) – em tradução livre Integração Contínua e Entrega Contínua – que são práticas que quando utilizadas de forma coordenadas, criam a base da cultura DevSecOps. Essas práticas auxiliam acelerar, automatizar e criar processos contínuos que veremos a seguir:

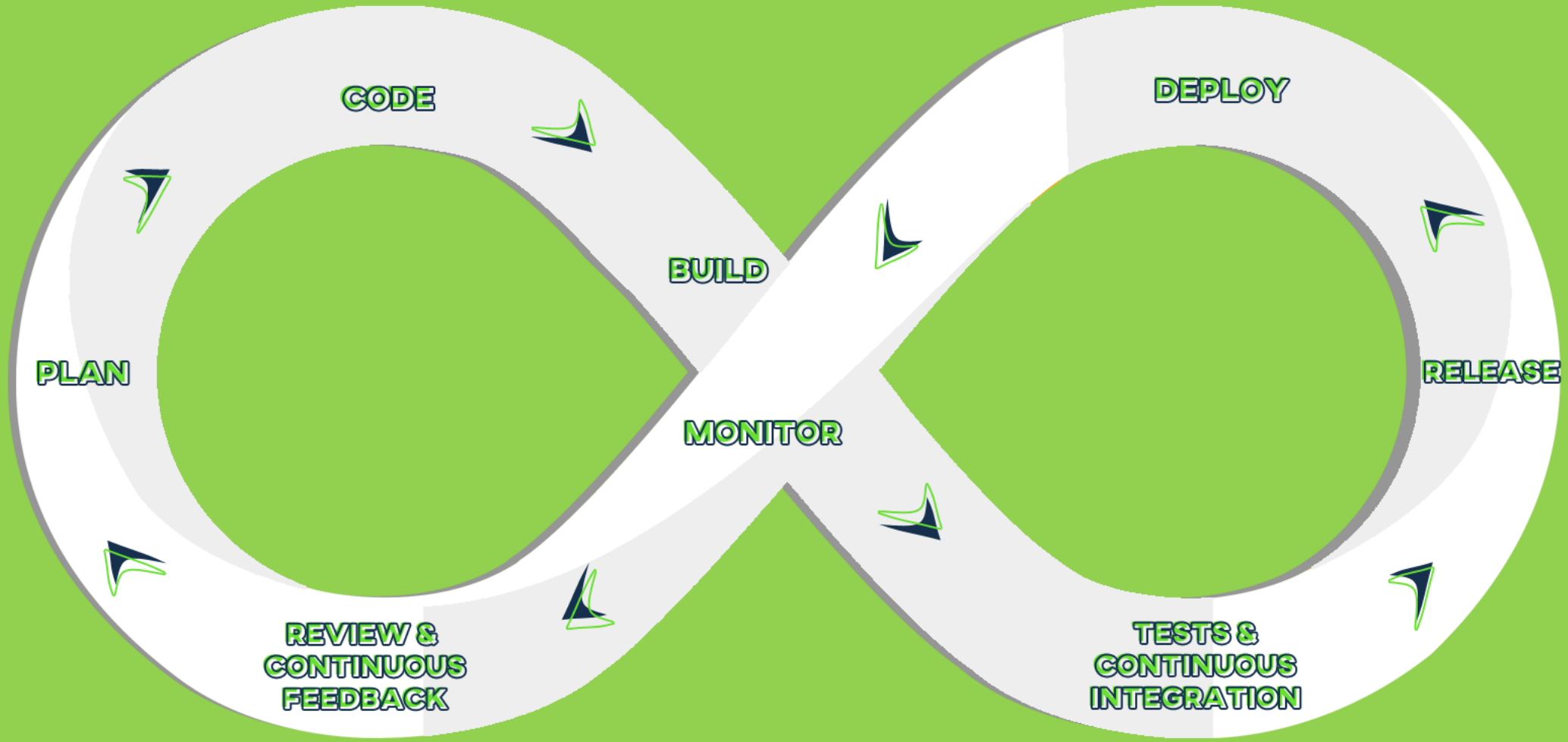
- **Continuos Integration (CI):** Na maioria das vezes os desenvolvedores trabalham em seus códigos individualmente, e com isso eles precisam no pós-desenvolvimento integrar seu código com a do resto do Time de Desenvolvimento em um repositório. Com isso, o CI cria e valida automaticamente as mudanças na versão do último código, permitindo a identificação de erros com antecedência e de reparos de falhas com rapidez.

  O que faz: Build, testes, zonas lint, pentest e vulnerabilidades.

- **Continuous Delivery (CD):** Antes as novas atualizações de um software geravam liberações não confiáveis que no fim se não fosse feita a refatoração do código mais erros e atrasos apareceriam. O CD é a capacidade que você tem de fazer o Deploy do sistema em produção. Ou seja, ele cria, testa e executa o software de forma automática e contínua, para garantir que ele possa ser lançado a qualquer momento e com mais frequência.

  O que faz: Infraestrutura code, testes automatizados (BDD), performance/lead testes e deployment.

COMO FUNCIONA



QUALIDADE E SEGURANÇA DURANTE TODO O PROCESSO!

POR FIM, FIQUE SEMPRE ALERTA!

Ao final de toda *sprint* é entregue uma parte do software para ser usado pelo usuário final. E mesmo priorizando durante todo o desenvolvimento a qualidade e tendo tudo validado pelos testes ainda assim é possível que algum incidente aconteça – e aqui pode ser algo que você nem imaginou que poderia ocorrer ou até algo que antes estava funcionando como deveria e agora não mais.

Uma coisa é certa, aplicações complexas – compostas muitas vezes por microsserviços – não possuem 100% de confiabilidade e falhas surgirão a longo prazo. E isso ocorre exatamente por serem complexas se tornando assim de suma importância ferramentas de monitoramento de ambientes e aplicações.

E utilizar técnicas visando alta disponibilidade e redundância, e melhor aproveitamento de recursos é fundamental para obter a flexibilidade necessária nestes cenários de falha.

No fim, entregar valor ao usuário final também consiste em manter a integridade e segurança adequada do produto em execução, e a única maneira de obter isso é estar sempre alerta monitorando de perto a aplicação para conseguir responder rapidamente a incidentes.

Em outras palavras, é ter alguém responsável em monitorar anomalias e ao detectá-las, focar para conseguir mitigar as falhas para que os usuários finais não sejam afetados.

CONCLUSÃO

A cultura em qualidade tem um papel importante no *Time To Market* de produtos, e ou, serviços, como de sua constante evolução. Se preocupar com qualidade durante todas as etapas do processo - desde escrever bons códigos até o monitoramento das entregas - aplicando testes, boas práticas e ferramentas, ajudam a evitar *bugs*, retrabalho e problemas na manutenção.

As dificuldades em aplicá-las até existem, mas reduzir os riscos, possuir *timelines* melhoradas e os outros diversos resultados gerados, são muito mais benéficos e garantimos: evita muita dor de cabeça.

E foi pensando em melhorar nossos serviços, que a Programmer's desenvolveu a solução **AGILE EXPERIENCE**, com abordagem baseada em métodos ágeis e com time multidisciplinar – combinamos diferentes capacidades e experiências - de forma a garantir o sucesso do desenvolvimento da aplicação digital que se preocupa em entregas rápidas, contínuas de valor e com qualidade.

Tem uma ideia de uma aplicação digital e precisa ter um rápido retorno sobre o investimento? Conheça mais nossa solução **AGILE EXPERIENCE** falando com um de nossos consultores!

SOBRE A PROGRAMMER'S

A Programmer’s é uma empresa que desenvolve soluções de negócio com inovação, tecnologia especializada em desenvolvimento e inteligência de dados.

Há anos, auxiliamos grandes marcas em sua transformação digital, acumulando expertise no desenvolvimento ágil por meio de tecnologias avançadas, como Big Data, Machine Learning, Inteligência Artificial, Analytics, Cloud Computing, entre outros.

Comprometidos com a qualidade de entrega e inovação que agregue valor, consolidamos parcerias concretas com clientes no mercado nacional e internacional, atuando na transformação digital de empresas do Brasil, EUA, Japão, Portugal, Itália e Austrália.

São Paulo
Av. das Nações Unidas, 12901 | 25 and - Cidade Monções
+ 55 (11) 3504-1100

Campinas
Av. John Dalton, 301 – Edificio 3, Cj. 13A - Techno Park Campinas
+ 55 (19) 3242-8033

Matão
Av. XV de Novembro, 1368 - Centro
+55 (16) 3384-3053

Araraquara
Av. Dr. Gastão Vidigal, 139 - Jardim Primavera
+55 (16) 3461-3088

contato: comercial@programmers.com.br

Programmer's®
Beyond IT