Desenvolvimento de Sistemas

Testes de *software*

Introdução

Em desenvolvimento de *software* sempre se ouve falar de testes, porém eles nem sempre ganham a importância que deveriam. Mas, afinal, o que são testes?

Pode-se dizer, de maneira informal, que teste é uma forma de fazer uma verificação e/ou validação. E na prática, como se pode definir o teste?

Realizar o processo de testes, ou seja, testar o *software*, é executá-lo a procura de defeitos. O objetivo é encontrar defeitos, todos ou uma boa parte deles.

No conteúdo a seguir, serão explorados conceitos, aspectos e definições sobre testes de *software*, ressaltando sua relevância em um projeto de *software*.

Figura 1 – Representação gráfica dos profissionais de teste de *software*

Fonte: FireWork (2021)

Conceito de teste

Pode-se dizer que teste de *software* é um processo normatizado que usa técnicas de execução de sistema para encontrar defeitos antes que este seja entregue ao cliente/usuário final. Essas técnicas, combinadas com o conhecimento do analista de testes, aumentam a confiabilidade do usuário no *software* que está sendo testado, fazendo com que o usuário tenha a garantia de que o sistema foi validado em relação às suas regras de negócio e de que está atendendo às suas expectativas.

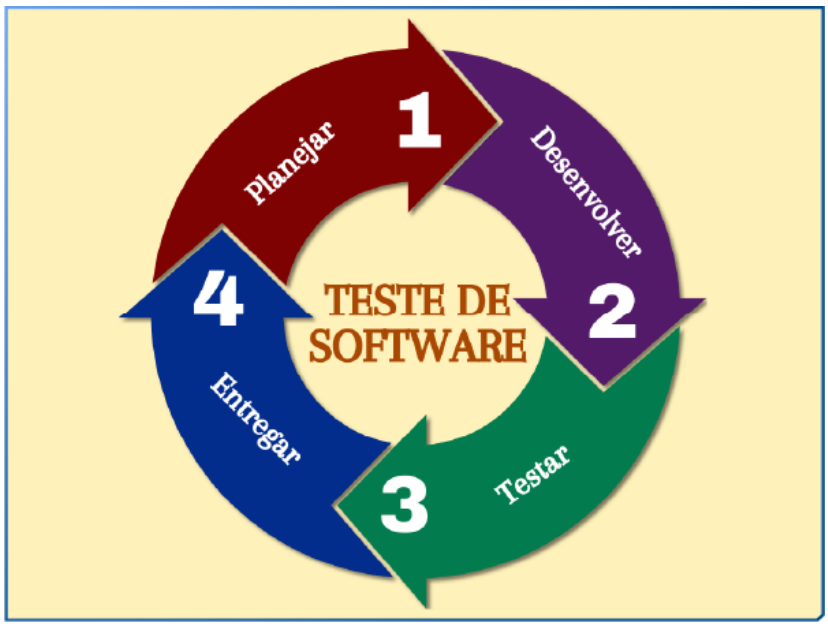

Figura 2 – Ciclo do teste de *software*
Fonte: Senac EAD (2023)

Como é possível ver na imagem, testar é um ciclo. O teste de *software* permite que ele esteja sempre em constante verificação, pois erros são comuns. Porém, essa prática faz com o que o *software* possa ser usado constantemente, sem necessidade de parar o trabalho do usuário para que sejam corrigidas as falhas (ou erros) que acabaram passando despercebidos, em razão da falta de teste.

O responsável por esses testes, chamado de analista de testes, tem a função de explorar falhas de um sistema e tentar, por meio de combinações incorretas, expor alguma vulnerabilidade do *software*. Por isso, é possível dizer que as atividades de teste têm característica “destrutiva”, porém esses testes aumentam a segurança e a confiabilidade do sistema, por meio de exposição desses erros e dessas falhas.

Caso de teste

Segundo o glossário da ISTQB (International Software Testing Qualifications Board), um selo de qualidade para testadores de *software*, define-se caso de teste como um conjunto de **valores de entrada, pré-condições de execução, resultados esperados e pós-condições de execução**, desenvolvido para um objetivo particular ou uma condição de teste.

Sabendo disso, como posso usar o caso de teste na prática?

Entenda, agora, um pouco melhor esse conceito.

Valores de entrada são dados que os seres humanos, ou uma máquina, fornecem para o *software*. O *software* recebendo esses dados fará o processamento e dará uma resposta que é comparada com os resultados esperados. Em resumo: você passará um grupo de valores de entrada, o *software* processará e, quando devolver o resultado, você analisará se os resultados esperados são iguais aos resultados processados.

Para que se possa enviar uma informação ao *software* e para que ele possa processar de uma maneira especifica, é necessário que o sistema esteja em um estado inicial ou mais avançado, mas que já tenha um escopo desenvolvido, que foi criado com base na regra de negócio. É possível citar algumas **pré-condições** para o teste: que já se tenha um usuário cadastrado, ou que já tenha saldo na conta, ou que já tenha algum requisito específico já configurado no *software*.

Já a **pós-condição** é qualquer tipo de informação que acontecerá no *software* depois que um determinado teste for executado.

Contador:	**001**
Localização:	**Tela de abertura NFE.**
Criticidade:	Alta
Objeto de Teste:	Verificar funcionamento da tela de abertura emissão nfe.
Caso de Teste:	Testar o funcionamento do botão "Emissão NF-e e campo fiscal"
Pré - Condição:	**1.Operação devidamente cadastrada.**
Procedimento:	**1.** Clicar no campo "Emissão de NF-e" **2.** Digite o código da operação ou clique F3 para pesquisar, depois de identificar a operação desejada. Pressione ENTER. 2.1 Digite um carácter, por exemplo, letra "A", no campo operação. 3. Digite o código da empresa ou pressione F3 para pesquisar, depois de selecionada a empresa pressione ENTER. 4. Digite o código do emitente ou pressione F3 para pesquisar, depois de identificado o emitente, pressione ENTER. **5.** Clicar no botão "Salvar".
Resultado Esperado:	**1.** O sistema deverá abrir a tela de abertura para NF-e. **2.** O sistema deverá passar para campo EMPRESA carregar dados. 2.1 O Sistema deverá mostrar mensagem de erro. **3.** O sistema deverá passar para campo EMITENTE carregar dados. 4. O sistema deverá autenticar o emitente selecionado. 5 . O sistema deverá abrir a tela para emissão da nota fiscal eletrônica.

Figura 3 – Exemplo de caso de teste

Fonte: Brasil Escola (2014)

Caso de teste *versus* cenário de teste

Quando se vai realizar testes em alguma aplicação, não se deve confundir as definições de caso de teste e de cenário de teste. Muitas vezes, por não entender muito do assunto, acaba-se achando que os dois são a mesma coisa, porém não são.

Caso de teste é um conjunto de ações executadas para verificar algum recurso ou alguma função da aplicação. **Cenário de teste** é qualquer funcionalidade da aplicação que possa ser testada. O caso de teste foca mais em como testar, já o cenário de teste concentra-se no que testará.

A equipe de controle de qualidade e desenvolvimento da aplicação é a responsável por escrever os casos de teste. Já em um cenário de teste, os responsáveis pela sua criação são os analistas e/ou gerentes de negócios.

Falando em escrita, veja, agora, como escrever um caso de teste.

Escrita do caso de teste

Para realizar a escrita de um caso de teste, deve-se, primeiro, conhecer o que se está testando e, depois disso, criar uma espécie de documento que pode conter todas (ou algumas) as etapas descritas a seguir:

1. ID: é um número único e exclusivo do caso de teste. Isso fará com que cada caso de teste se diferencie do outro.
2. Descrição: é um texto breve, porém adequado ao que se está testando (recurso, unidade, função etc.), com detalhes importantes do caso de teste.
3. Pré-condições: como visto anteriormente, é tudo que deve ser observado antes da execução do teste.
4. Etapas de teste: é muito importante descrever cada passo (cada etapa) usando uma linguagem de fácil compreensão.
5. Resultado esperado/desejado: nesta etapa, são explicados os resultados esperados pelo caso de teste.
6. Pós-condições: aqui, serão descritas as condições que devem ser atendidas após a execução com sucesso do caso de teste.
7. Estado: com base nos resultados obtidos, deve-se mencionar se o *status* do caso de teste é aprovado, caso tenham sido obtidos os resultados desejados, ou reprovado, caso os resultados não tenham sido os esperados.

Exemplificando:

O cliente João realizará uma transferência para Maria (será a favorecida neste exemplo). Não será necessário ir até o banco, pois tudo será feito *on-line*, por meio de um *software* específico. Essa transferência será feita em razão de um processo judicial no qual João deve pagar a quantia de R$ 800,00 para Maria. No ato da transferência, João tinha essa quantia em sua conta e Maria criou uma conta para esse fim, a qual se encontra sem nenhum dinheiro.

Como essa transferência se deve a um processo judicial, João terá de realizá-la usando um *software* específico. Nesse sistema, ele informará os próprios dados de *login*. Após isso, irá na aba de **Transferência Judicial**, informará os dados da conta de Maria e confirmará a transferência. Feito isso, João sairá do sistema, para que o seu usuário não fique “logado”. Se tudo der certo, Maria ficará com o saldo de R$ 800,00 e o processo judicial terá sido executado com sucesso.

Com base no cenário anterior, é possível montar um caso de teste, conforme o exemplo a seguir.

Exemplo de caso de teste

ID: 001
Descrição: Transferência de valor com saldo positivo
Pré-condições de execução:
- O cliente João precisa ter saldo de R$ 800,00.
- A favorecida (pessoa que receberá o dinheiro) precisa ter saldo de R$ 0,00.

Valores de entradas:
- Maria como favorecida
- R$ 800,00 como valor de transferência

Etapas de teste:

Etapa	Descrição	Resultado esperado
1	Faça *login* com os dados do cliente	Tela secreta apresentada
2	Inicie uma transferência de valores	Foco no campo **Favorecido**
3	Informe R$ 800,00 para Maria	Mensagem de confirmação é apresentada
4	Confirme a transferência	Mensagem de sucesso na transferência é apresentada
5	Faça *logoff* do sistema	Tela de *login* é apresentada

Resultados esperados:
- Sucesso ao transferir a quantia

Pós-condições de execução:
- Novo saldo do cliente que realizou a transferência será de R$ 0,00.
- Novo saldo da favorecida (Maria) será de R$ 800,00.

Estado: Aprovado

Tabela 1 – Exemplo de caso de teste

Fonte: Senac EAD (2023)

Veja, a seguir, outro exemplo.

Gustavo finalizou a programação de uma tela de cadastro que será implementada em sistema *desktop* já existente. Na empresa onde esse sistema *desktop* é usado, somente alguns usuários poderão preencher essa tela de cadastro e salvar os dados. Foi solicitado que um usuário com a devida permissão faça o cadastro e tente salvar. O usuário em questão acessou o sistema, colocou os próprios dados de *login*, acessou a tela de cadastro e conseguiu realizar um cadastro com sucesso.

Baseando-se no cenário anterior, é possível montar um caso de teste, conforme o exemplo a seguir.

Exemplo de caso de teste

ID: 002
Descrição: Preenchimento e salvamento de dados em uma tela de cadastro
Pré-condições de execução:
- O usuário precisa ter permissão para acessar a tela de cadastro.
- O *login* e a senha do usuário precisam ser válidos.

Valores de entrada:
- Usuário com acesso à tela de cadastro.
- Usuário fornece os dados de *login* dele.

Etapas de teste:

Etapa	Descrição	Resultado esperado
1	Faça *login* com os dados do cliente	Direcionamento para o menu de opções
2	Acesse a tela de cadastro	Tela de cadastro é apresentada
3	Preencha todos os campos do cadastro	Os dados estão corretos
4	Salve o cadastro	Mensagem de cadastro realizado com sucesso é apresentada
5	Faça *logoff* do sistema	Tela de *login* é apresentada

Resultados esperados:
- Cadastro realizado com sucesso.

Pós-condições de execução:
- Usuário autorizado consegue preencher a tela de cadastro e salvar.

Estado: Aprovado

Tabela 2 – Exemplo de caso de teste

Fonte: Senac EAD (2023)

Pratique por meio do desafio a seguir.

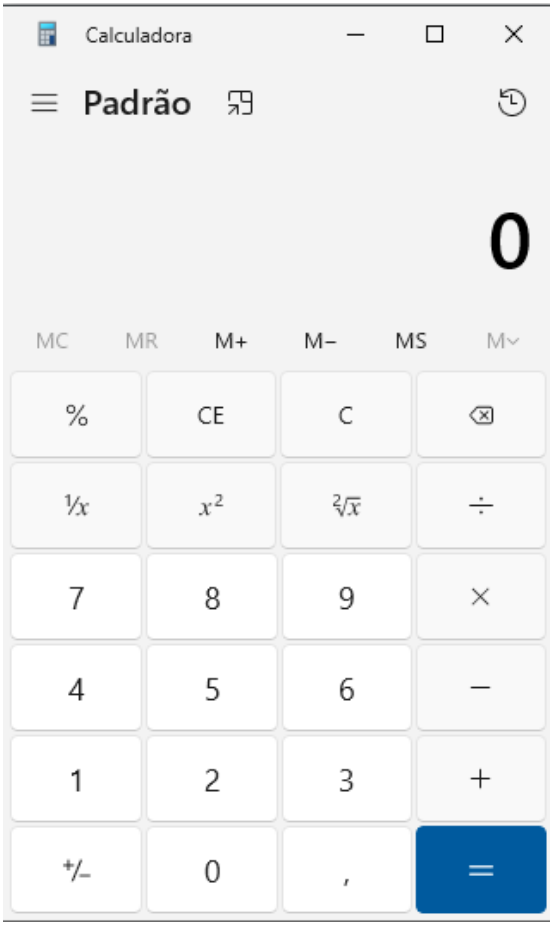

Figura 4 – Calculadora do Windows

Fonte: Senac EAD (2023)

A equipe de teste da empresa Senac Tecnologia foi contratada para testar um *software* que simula uma calculadora virtual. Devem ser testadas todas as funcionalidades apresentadas pela calculadora para efetuar os cálculos (não precisa testar teclas nem abas de função). A equipe deverá preparar um documento de teste contendo pelo menos três casos de teste.

Erro, defeito e falha: entendendo a diferença

Antes de conceituar essas três palavras, veja um pequeno cenário criado em que, na prática, erro, defeito e falha acontecerão.

Jorge foi contratado por uma empresa para criar um cadastro de produtos dentro de um sistema que está em andamento. Ao codificar, Jorge se esqueceu de alguns detalhes muito importantes em seu código, mas, mesmo assim, enviou suas linhas de código para os outros desenvolvedores, que implementaram no sistema. Logo após isso, o sistema apresentou **erro** ao cadastrar, em razão do código incompleto. Os desenvolvedores não sabiam o que fazer e executaram o sistema, que apresentou um grande **defeito**, em razão do erro de Jorge. Após essa execução não ter sido bem sucedida, o sistema **falha**.

Resumindo:

- **Erro:** foi cometido por Jorge, ao raciocinar o que precisava fazer.
- **Defeito:** foi inserido pelo código errado de Jorge.
- **Falha:** foi causada porque o sistema passou pelo fluxo defeituoso.

A seguir, veja as definições formais dessas palavras.

Erro: estado inconsistente, observado em momento de execução, de um produto ou sistema de *software*. Esse erro pode ou não causar uma falha.

Defeito/*bug*: passo, processo ou definição de dados incorretos; ou, ainda, qualquer problema existente no *software* ou no sistema sob teste que possa causar sua falha em atender às expectativas do usuário. Em outras palavras, um defeito é uma fonte potencial de insatisfação com o produto.

Falha: estado inesperado do produto ou do sistema de *software*, ocasionado por um erro.

Figura 5 – Explicação de erro, defeito e falha

Fonte: adaptado de Jordan (2018)

Plano de teste

O plano de teste contém os casos de uso, as telas e o projeto que serão testados, os responsáveis pelo teste, por elaborar roteiros e criar ambiente, os erros que podem ser encontrados, os riscos do processo etc. É o planejamento inicial do teste sobre a demanda enviada para a equipe.

Dependendo do tamanho do projeto de desenvolvimento, é possível ter diversos planos de teste para um mesmo projeto de teste de *software*. Esses planos podem ser separados por módulos, funcionalidades ou requisitos, ou, ainda, por um grupo de cada um desses elementos.

Além disso, os planos podem também ser classificados por nível de teste: unitário, integração, sistema e aceitação. Quando o sistema for pequeno, não há necessidade da criação de vários planos de teste, embora, mesmo nesses casos, possa ser razoável ter os planos de teste separados por nível de teste.

Além de definir a agenda de teste e os procedimentos necessários para testar o produto, o plano de teste define a infraestrutura necessária. Nesse sentido, no plano de teste, encontram-se também as anotações sobre os recursos humanos, de *hardware* e de *software* necessários. Essas estimativas são úteis principalmente para os gerentes de projetos e configuração que são responsáveis por garantir que esses recursos estarão disponíveis para a execução dos testes.

Um plano de testes menos formal pode ser utilizado em pequenos sistemas, mas deve ser utilizado um documento formal para auxiliar o planejamento do processo de teste. Os planos de teste não são documentos estáticos, pois evoluem durante o processo de desenvolvimento, mas existe uma padronização dos planos de teste que está descrita na norma IEEE 829 4.1.4. Essa sigla refere-se ao Instituto de Engenheiros Eletricistas e Eletrônicos.

Você pode conferir mais detalhes sobre esses documentos no conteúdo **Plano de teste** desta unidade curricular.

Teste estático e teste dinâmico

As atividades para avaliação da qualidade de *software* podem ser divididas em duas categorias amplas: teste estático e teste dinâmico.

Teste estático

Como o termo “estático” sugere, baseia-se no exame de uma série de documentos, nomeadamente documentos de requisitos, *software*, modelos, documentos de *design* e código-fonte. O teste estático tradicional inclui revisão de código, inspeção, passo a passo, análise de algoritmo e prova de veracidade. Não envolve a execução real do código em desenvolvimento. Em vez disso, ele examina o código e os motivos de todos os comportamentos possíveis que podem surgir durante o tempo de execução.

Pode-se dizer que, quando se está olhando um código e analisando-o, um teste e/ou uma análise estática estão sendo feitos. Nessa análise, é possível realizar algumas verificações, como:

- O código foi documentado?
- As variáveis, as constantes e as classes estão com uma nomenclatura aceitável?
- O código está organizado e é de fácil compreensão?
- O programador, ao codificar, obedece à arquitetura do sistema?

Teste dinâmico

O teste dinâmico de um sistema envolve a execução do programa para expor suas possíveis falhas. As propriedades comportamentais e de desempenho do programa também são observadas. Os programas são executados com valores de entrada típicos e cuidadosamente escolhidos.

Esse teste acontece dentro do ambiente de desenvolvimento e envolve verificações como:

- Entrar e sair do sistema.
- Inserir *login* e senha e verificar se autorizará.
- Abrir as opções desejadas no menu.
- Clicar em determinada opção e o *software* direcionar a ela.

Portanto, nos testes, alguns comportamentos representativos do programa são observados e é possível chegar a uma conclusão sobre a qualidade do sistema.

Ao realizar testes estáticos e dinâmicos, os profissionais desejam identificar o máximo de falhas possível, para que estas sejam corrigidas em um estágio inicial de desenvolvimento do *software*. A análise estática e a análise dinâmica são complementares por natureza e, para ter melhor eficácia, ambas devem ser realizadas de forma repetida e alternada.

Critério de aceitação

Antes de realizar um teste de aceitação (tipo de teste que acontece geralmente antes da implantação do *software* e tem como objetivo verificar se ele está pronto, aceito pelo cliente, e já pode ser utilizado para a sua finalidade), é preciso definir, de forma clara, quais critérios serão utilizados. Tais critérios são divididos em: critério de aceitação funcional e critério de aceitação não funcional.

O **critério de aceitação funcional** diz respeito ao comportamento do sistema, em como ele ajudará os seus usuários a realizar determinado trabalho. Refere-se às funções ou aos recursos que o sistema oferece.

Já o **critério de aceitação não funcional** especifica todos os requisitos. Ele avalia sobre como o sistema funciona sem levar muito em consideração os códigos, e sim os aspectos de acessibilidade, a facilidade de uso, a sua segurança e privacidade, a velocidade entre as telas, a confiabilidade, entre outros aspectos.

Como definir os critérios de aceitação

Após entender os conceitos dos critérios de aceitação, você aprenderá como definir esses critérios, que incluem os requisitos de desempenho e as condições essenciais que serão atendidas antes de se realizar a entrega do projeto. Os critérios determinam sob quais situações o cliente aceitará o resultado final do projeto. Assim, é possível medir e provar ao cliente que o trabalho está completo.

Algumas técnicas podem ser usadas para definir critérios de aceitação. Para funcionalidades em que é possível definir um cenário de uso, o esquema “dado que/quando/então” (*given/when/then*) pode ser útil.

- “Dado que” alguma pré-condição
- “Quando” uma ação é executada
- “Então” espera-se algum resultado

Veja, a seguir, um exemplo dessa técnica aplicada a um caso de uso para um caixa eletrônico.

História de usuário: Como usuário, quero ser capaz de solicitar o saque de um terminal de caixa eletrônico de maneira que eu possa receber o dinheiro da minha conta bancária rapidamente em diferentes locais.	
Cenário 1: Solicitar o saque de uma conta com crédito	
Dado que	A conta possui saldo
E	O cartão é válido
E	O caixa eletrônico tem notas
Quando	O usuário solicita o saque
Então	Assegure-se que a conta seja debitada no valor solicitado
E	Assegure-se que as notas sejam disponibilizadas pela máquina
E	Assegure-se que o cartão é liberado ao usuário

Tabela 3 – Exemplo do esquema “dado que/quando/então”

Fonte: Senac EAD (2023)

Modernize a história de usuário exemplificada anteriormente trocando caixa eletrônico por Pix. Considere o cenário em que o usuário vai transferir dinheiro a outra pessoa usando uma chave Pix.

Em outros casos mais técnicos, em que seja difícil aplicar o esquema “dado que/quando/então”, como a interação com uma interface gráfica, pode ser necessário usar um esquema orientado a regras. Nesse formato, usa-se geralmente uma *checklist* de regras que descrevem o comportamento de um sistema ou de uma funcionalidade específica. Com base nessas regras, pode-se desenhar cenários mais específicos.

Veja, no exemplo a seguir, uma história de usuário para um sistema *web* que faz busca por hotéis.

História de usuário: Como usuário, quero ser capaz de usar um campo de pesquisa no sistema para encontrar opções de hotel, informando cidade, nome ou rua.
Critérios básicos de aceitação para busca: ◆ O campo de busca está posicionado no topo da tela. ◆ A busca inicia ao clicar no ícone **Lupa**. ◆ O campo está vazio. ◆ A busca é realizada se um usuário informar uma cidade, um nome de hotel, uma rua ou todos essas informações combinadas. ◆ A busca ocorre em português. ◆ O usuário não pode digitar mais que 200 caracteres.

Tabela 4 – Exemplo de busca

Fonte: Senac EAD (2023)

Outros formatos próprios para escrita de critério de aceitação podem ser aplicados, como tabelas com valores possíveis para uma funcionalidade ou texto explicativo do que deve ou não ocorrer em um cenário de uso específico.

Veja, a seguir, alguns exemplos de critérios de aceitação não funcionais.

1. **Requisito**: O sistema deve ser capaz de processar a listagem de dados automaticamente.
   Critério de aceitação: A falha no processamento de dados não pode ultrapassar 5%;
2. **Requisito**: O processo de *backup* e restauração devem estar em perfeito funcionamento.
   Critério de aceitação: O processo de *backup* e restauração foi testado com sucesso.
3. **Requisito**: A funcionalidade de *login* não deve permitir que um usuário não cadastrado no banco de dados faça *login* no sistema.
   Critério de aceitação: Somente usuários autenticados (cadastrados no banco de dados) acessam o sistema.

Características dos critérios de aceitação

- Os critérios de aceitação devem ser fáceis de serem testados pelo usuário. Seus resultados devem ser de simples interpretação. Dentro do teste de aceitação, as respostas esperadas seriam: sim ou não, aprovado ou reprovado, aprovado ou resultados inconclusivos etc.

- Os critérios de aceitação devem ser compreensíveis e resumidos. Não há necessidade de uma extensão documentação; a finalidade é tentar ser o mais claro e breve possível.

- Os critérios de aceitação devem ser compreendidos por todos os usuários.

- Os critérios de aceitação devem fornecer uma perspectiva do usuário. Eles farão com que o problema seja percebido pelo ponto de vista do usuário, e isso com certeza trará resultados positivos para o sistema.

Artefatos de testes

Cada empresa e cada metodologia definidas no começo do processo de desenvolvimento de *software* pedem um tipo de artefato para resolver as necessidades do projeto/sistema. Não há um documento padrão seguido por todas as equipes e empresas do mundo. É preciso saber como o processo de teste na empresa foi montado para, então, ver quais documentos devem ser produzidos.

O artefato de teste é uma parte integrante do teste do *software*, e o conjunto de documentos que o compõe fazem parte do ciclo de vida do teste do *software*.

Sabendo disso, há alguns tipos de documentos que podem/devem ser usados para entender se a necessidade do projeto/sistema foi, de fato, resolvida. Veja, a seguir, informações sobre alguns desses documentos.

Estratégia de teste

Geralmente é desenvolvida pelo gerente de projeto. Uma estratégia de teste é um documento que lista os detalhes sobre como todo o projeto prosseguirá. Isso inclui requisitos e recursos para o projeto, estratégias de teste envolvidas, *design* de teste, fases incrementais envolvidas, processos de comunicação com o cliente etc. É o resumo do documento que descreve a abordagem de teste do ciclo de desenvolvimento de *software* e lista como alcançar o resultado esperado usando os recursos disponíveis.

Para desenvolver essa estratégia, há vários pontos que devem ser considerados. A seguir, veja alguns deles.

- Qual é o principal objetivo do teste e o motivo pelo qual você deseja realizá-lo?
- Quais são as diretrizes que devem ser seguidas para a realização de testes?
- Quais são todos os requisitos necessários para o teste, como requisitos funcionais, cenários de teste, recursos etc.?
- Quais são as funções e as responsabilidades de cada função e cada gerente de projeto para concluir o teste?
- Quais são os diferentes níveis de teste?
- Qual será a principal entrega desse teste?
- Quais riscos existem em relação aos testes com os riscos do projeto?
- Existe algum método para resolver problemas que possam surgir?

Plano de teste

Como citado anteriormente, é um documento de planejamento do processo de teste do *software*.

Cenário de teste

É uma condição criada para realizar testes de ponta a ponta bem-sucedidos. Vários casos de teste vêm em um cenário de teste.

Às vezes, também se refere a uma condição ou uma possibilidade de teste. Um cenário de teste resolve o dilema da aplicação do *software* na vida real.

Casos de teste

Este artefato de teste foi citado dentro deste conteúdo.

Dados de teste

Para executar os casos de teste, o engenheiro de controle de qualidade precisa de alguns dados sobre os quais o caso de teste foi aprovado ou reprovado para decidir. Para isso, é criado um documento que contém todos os dados de teste necessários para a execução dos casos de teste criados.

Relatório de defeito/*bug*

Um relatório de defeitos é um documento criado após a execução de todos os casos de teste e o registro dos resultados. Ele lista todos os *bugs* encontrados durante o teste. Isso torna mais fácil, para a equipe de desenvolvimento, a correção de *bugs*.

Relatório de resumo de teste

Ao final de cada um dos ciclos de teste completos, um relatório final é criado. Esse relatório inclui os detalhes de todo o processo de teste. Por exemplo: detalhes sobre casos de teste, resultados obtidos, *bugs* corrigidos, *status* atual do projeto etc.

Esse documento é importante e deve ser apresentado às partes interessadas para mantê-las atualizadas sobre o andamento do projeto. É um documento formal e tudo está claramente documentado para que qualquer pessoa possa entender o relatório.

Guia do usuário

Com todo *software* desenvolvido e pronto para ser implantado no mercado, o guia do usuário é criado ao final. Esse guia é útil para usuários finais e fornece informações detalhadas sobre o *software* e sobre o uso deste.

Encerramento

Testar um *software* é muito importante, porém ainda mais importante é saber o que testar. Um *software* sem defeito não significa que está perfeito, pode significar também que não foi totalmente testado ou foi testado de maneira errada.

Dentro de qualquer projeto, é muito importante sempre verificar a documentação, para compreender melhor o *software* e também o usuário final. Dentro dos testes de *software*, isso é essencial, pois é o usuário final quem utilizará o sistema no dia a dia, então é quem pode melhor dizer se há erros ou não.