ALINE FRANÇA

Mentoria Seja Tester

Academy – Aula 5

Testes ao longo do Ciclo de Vida de Desenvolvimento de Software- Parte 01

ALINE FRANÇA

SUMÁRIO

Quais assuntos vamos aprender nessa aula?

- Testes no Ciclo de Vida do Desenvolvimento de Software
- Como o Ciclo de Vida do Desenvolvimento de Software Afeta os Testes
- Desenvolvimento de Software e Boas Práticas de Teste
- Teste como um Estímulo para Criar o Software
- DevOps e Testes
- Como o DevOps ajuda nos testes?

ALINE FRANÇA

SUMÁRIO

Quais assuntos vamos aprender nessa aula?

- O que é a Abordagem Shift-Left (ou “testar mais cedo”)
- Como aplicar esse “testar mais cedo” na prática?
- Retrospectivas e melhoria de processos

ALINE FRANÇA

TESTES NO CICLO DE VIDA DO DESENVOLVIMENTO DE SOFTWARE

Quando uma equipe cria um software (como um aplicativo ou site), ela segue um passo a passo para planejar, desenvolver, testar e entregar esse software. Esse passo a passo é chamado de ciclo de vida do desenvolvimento de software (ou SDLC, na sigla em inglês).

Esse ciclo é como uma receita que mostra quais etapas devem ser seguidas e em qual ordem, desde a ideia inicial até o momento em que o software está pronto para ser usado. Existem diferentes formas de organizar essas etapas. Por exemplo:

ALINE FRANÇA

SDLC...

- Em alguns casos, tudo é feito em sequência, etapa por etapa, como em uma linha de produção. (Exemplos: modelo em cascata e modelo em V).
- Em outros, as etapas se repetem várias vezes, para melhorar o software aos poucos. (Exemplos: modelo em espiral, prototipagem).
- Há também modelos que entregam partes do software em etapas menores, sempre melhorando aos poucos. (Exemplo: Processo Unificado).

ALINE FRANÇA

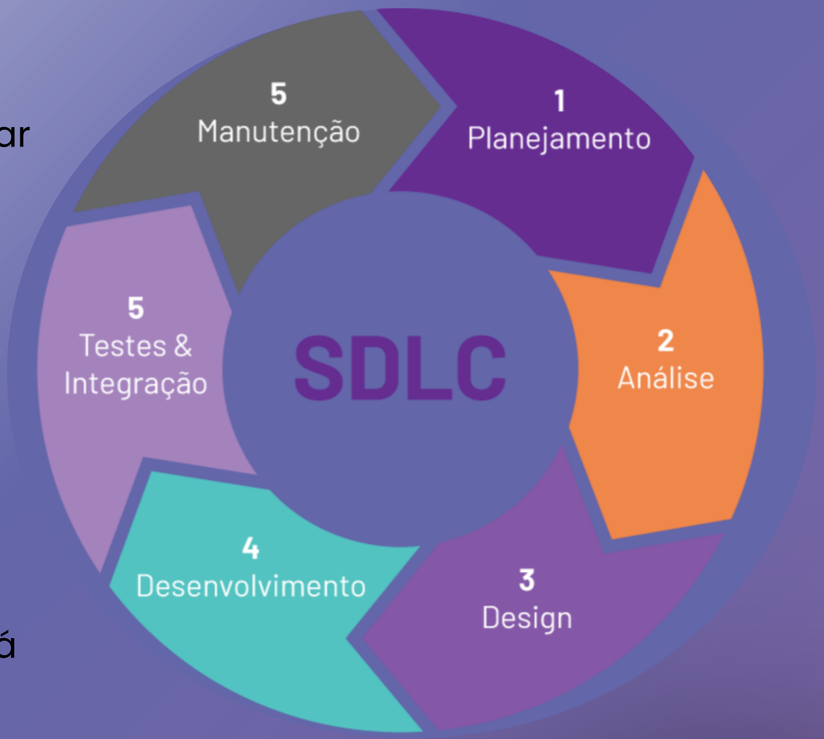
SDLC...

Além desses modelos mais gerais, existem métodos e práticas mais detalhadas para ajudar no desenvolvimento. Esses métodos definem como o time trabalha no dia a dia.

Alguns exemplos:

- **ATDD**: o time começa desenvolvendo com base no que o cliente espera que funcione.
- **BDD**: o foco é no comportamento do sistema, como ele deve reagir.
- **DDD**: a equipe organiza o sistema com base nos conceitos e regras do negócio.
- **XP e FDD**: são formas de trabalhar com entregas rápidas e com foco em qualidade.
- **Kanban, Lean e Scrum**: ajudam a organizar o trabalho do time e acompanhar o que está sendo feito.
- **TDD**: antes de escrever o código, o time escreve testes para garantir que tudo funcione direitinho.

Essas práticas também ajudam a garantir que o software seja bem testado e funcione como esperado antes de ser entregue para quem vai usar.



ALINE FRANÇA

COMO O CICLO DE VIDA DO DESENVOLVIMENTO DE SOFTWARE AFETA OS TESTES

Para que os testes de um software funcionem bem e encontrem problemas de verdade, eles precisam seguir o mesmo caminho que o time está usando para criar o software, ou seja, precisam se adaptar à forma como o projeto está sendo feito.

A maneira escolhida para desenvolver o software (o modelo de SDLC) influencia coisas como:

- O que será testado e quando os testes vão acontecer;
- Quanta documentação será feita para os testes;
- Quais técnicas e estratégias de teste serão usadas;
- O quanto os testes vão ser automáticos;
- Quais são as tarefas e responsabilidades da pessoa que testa (o Testador).

ALINE FRANÇA

CONTINUAÇÃO...

Por exemplo::

- Nos modelos que seguem uma ordem fixa, como o modelo em **cascata**, o testador geralmente entra no começo para revisar os planos e requisitos e só vai testar de verdade (com o sistema funcionando) no final, quando o código estiver pronto. Ou seja, os testes “dinâmicos”, aqueles em que se executa o software, só acontecem depois.
- Nos modelos que entregam o software em partes ou ciclos, como os modelos **iterativos** e **incrementais**, cada nova versão (ou “incremento”) do sistema já pode ser testada logo. Isso quer dizer que dá para testar tanto antes quanto depois do código ser escrito, e em várias etapas. Como as entregas acontecem com frequência, os testes precisam ser rápidos e repetidos várias vezes para garantir que nada que já estava funcionando quebre, isso se chama teste de regressão.

ALINE FRANÇA

CONTINUAÇÃO...

Por exemplo::

Nos métodos ágeis, que são mais flexíveis e aceitam mudanças durante o projeto, os testes também precisam se adaptar. Por isso:

- A documentação é mais simples e direta (sem excesso de papéis);
- Muitos testes são automáticos, para dar respostas rápidas e facilitar os testes repetidos;

ALINE FRANÇA

DESENVOLVIMENTO DE SOFTWARE E BOAS PRÁTICAS DE TESTE

Algumas práticas ajudam a garantir que o software seja bem testado, independentemente de qual modelo de desenvolvimento (SDLC) o time esteja usando. Essas boas práticas incluem:

- Para cada etapa do desenvolvimento, existe uma etapa de teste equivalente. Isso significa que tudo o que está sendo criado é também verificado, ajudando a manter a qualidade em todas as fases do projeto.
- Existem diferentes tipos de testes, com objetivos diferentes. Isso permite que os testes cubram várias áreas do sistema sem repetir desnecessariamente as mesmas verificações.
- O planejamento dos testes começa junto com o desenvolvimento. Ou seja, o time não espera o sistema estar pronto para só então pensar em como testá-lo. Esse cuidado antecipado ajuda a encontrar problemas mais cedo, o que economiza tempo e esforço depois.

ALINE FRANÇA

DESENVOLVIMENTO DE SOFTWARE E BOAS PRÁTICAS DE TESTE

- As pessoas responsáveis por testar (os testadores) participam das revisões dos documentos assim que esses documentos ficam prontos, mesmo que ainda estejam em rascunho. Isso também ajuda a identificar falhas logo no começo e segue uma estratégia chamada “shift-left”, que significa trazer os testes para as fases iniciais do projeto, em vez de deixar para o final.

ALINE FRANÇA

TESTE COMO UM ESTÍMULO PARA CRIAR O SOFTWARE

Existem algumas formas de desenvolver software em que os testes são criados antes mesmo do código. Nesses casos, os testes servem como um guia que mostra o que precisa ser feito. Essas abordagens ajudam a começar os testes bem no início do projeto, chamado de teste antecipado e fazem parte de uma estratégia conhecida como shift-left, que significa testar o quanto antes. Essas formas de trabalhar são baseadas em ciclos de melhoria contínua e costumam ser usadas em projetos com entregas frequentes. Vamos entender cada uma delas de forma simples:

TDD – Desenvolvimento Orientado por Testes: Antes de escrever qualquer código, o desenvolvedor escreve testes pequenos que dizem o que o sistema deve fazer. Depois, ele escreve o código necessário só para passar nesses testes. Por fim, melhora o código (isso se chama "refatorar"), mantendo os testes funcionando.

Exemplo simples: Se o sistema precisa somar dois números, o desenvolvedor escreve um teste que verifica se 2 + 2 = 4. Só depois ele cria o código que faz essa conta funcionar.

ALINE FRANÇA

TESTE COMO UM ESTÍMULO PARA CRIAR O SOFTWARE

ATDD – Desenvolvimento Guiado por Testes de Aceite: Os testes são baseados em critérios de aceitação, ou seja, nas regras que o cliente ou usuário espera que o sistema siga. Os testes são definidos antes da parte do sistema ser construída, garantindo que o que for desenvolvido atenda exatamente ao que o cliente espera.

Exemplo simples: Se o cliente quer que um botão "Enviar" funcione só quando todos os campos estiverem preenchidos, o teste é escrito com essa condição antes do botão ser programado.

BDD – Desenvolvimento Guiado pelo Comportamento: O foco aqui é descrever como o sistema deve se comportar, mas usando uma linguagem mais próxima do dia a dia, que qualquer pessoa entenda. Normalmente, os testes são escritos no formato: Dado (uma situação inicial), Quando (uma ação acontece), Então (o resultado esperado).Esses testes escritos em linguagem natural podem ser convertidos automaticamente em testes que o sistema pode executar.

Exemplo simples: “Dado que o usuário está logado, quando ele clicar em ‘sair’, então ele deve voltar para a tela de login.”

- Esses testes podem continuar sendo usados depois, como testes automáticos, para garantir que nada se quebre caso o sistema seja alterado ou melhorado no futuro.

ALINE FRANÇA

DEVOPS E TESTES

DevOps é uma forma moderna de organizar o trabalho em empresas de tecnologia. A ideia é que quem desenvolve o sistema (incluindo quem testa) e quem cuida da parte técnica depois que ele está no ar (as operações) trabalhem juntos, como um único time, com os mesmos objetivos. Para isso dar certo, a empresa precisa mudar a cultura interna, valorizando igualmente todos os envolvidos, tanto quem cria o sistema quanto quem garante que ele funcione bem depois de pronto.

O DevOps valoriza:

- Equipes mais independentes, que conseguem tomar decisões por conta própria;
- Feedback rápido, ou seja, respostas rápidas sobre o que está funcionando ou não;
- Ferramentas conectadas entre si, facilitando o trabalho do início ao fim;
- Práticas como a Integração Contínua (CI) e a Entrega Contínua (CD), que permitem criar, testar e liberar versões do sistema com mais qualidade e rapidez.

Tudo isso é feito através de um pipeline de entrega, que é como uma esteira automática que vai levando o software pelas etapas de desenvolvimento, testes e publicação.

ALINE FRANÇA

COMO O DEVOPS AJUDA NOS TESTES?

- Ele dá respostas rápidas sobre a qualidade do sistema, mostrando se uma mudança estragou algo que já funcionava.
- Com o uso do CI, os desenvolvedores são incentivados a testar o que fazem desde o início (isso é o que chamamos de "testar cedo" ou shift-left).
- Os ambientes para testar ficam mais estáveis e automáticos, graças às ferramentas de CI/CD.
- Ajuda a cuidar da qualidade geral do sistema, como velocidade e confiabilidade.
- Reduz a necessidade de testes manuais repetitivos, porque muita coisa passa a ser feita por testes automáticos.
- Diminui os riscos de problemas quando algo antigo para de funcionar por causa de mudanças (isso é chamado de teste de regressão).

ALINE FRANÇA

COMO O DEVOPS AJUDA NOS TESTES?

Mas nem tudo é tão simples...

- A empresa precisa definir e configurar essa esteira automática (a pipeline).
- As ferramentas de CI/CD precisam ser escolhidas, configuradas e mantidas com cuidado
- A automação dos testes exige tempo, conhecimento e recursos, e pode ser difícil de manter funcionando sempre bem.

Mesmo com muitos testes automáticos, os testes manuais ainda são importantes, especialmente aqueles que simulam a experiência real de um usuário usando o sistema.

ALINE FRANÇA

O QUE É A ABORDAGEM SHIFT-LEFT (OU “TESTAR MAIS CEDO”)

A ideia de “shift-left” significa começar a testar o software o mais cedo possível no processo de criação. Em vez de esperar o sistema estar quase pronto para começar os testes, a proposta aqui é começar a verificar tudo desde o início, até mesmo antes do código estar escrito. Mas atenção: testar cedo não significa parar de testar depois. Os testes das fases finais continuam sendo importantes também.

Por que testar mais cedo vale a pena? Mesmo que essa abordagem exija mais tempo, esforço e até custo no começo, a ideia é que ela ajude a economizar bastante lá no final, evitando retrabalho, falhas e atrasos. Para que tudo funcione bem, é importante que as pessoas envolvidas no projeto entendam e aceitem essa forma de trabalhar, pois mudar a ordem tradicional pode causar resistência no início.

ALINE FRANÇA

COMO APLICAR ESSE “TESTAR MAIS CEDO” NA PRÁTICA?

Algumas boas práticas para isso incluem:

- Ler e revisar os documentos do projeto com olhar de testador.
- Isso ajuda a encontrar problemas como erros de escrita, coisas que estão mal explicadas ou ideias que se contradizem antes mesmo do código ser feito.
- Criar os testes antes do código. Assim, o código já nasce preparado para funcionar corretamente e para ser testado enquanto está sendo feito.
- Usar ferramentas de automação que testam o código sempre que ele é enviado para o sistema central (isso é o que chamamos de Integração Contínua - CI - e Entrega Contínua - CD). Isso dá respostas rápidas se algo estiver errado.
- Analisar o código mesmo sem executá-lo (isso se chama análise estática). É como dar uma olhada no código e encontrar falhas só lendo, sem precisar rodar o sistema ainda.
- Testar coisas como desempenho e segurança logo nas partes menores do sistema (componentes), sem esperar que tudo esteja pronto. Esses testes normalmente são feitos mais tarde, mas começar antes ajuda a evitar problemas grandes depois.

ALINE FRANÇA

RETROSPECTIVAS E MELHORIA DE PROCESSOS

As retrospectivas são reuniões feitas no final de um projeto ou de uma etapa do projeto (como um ciclo de trabalho), ou sempre que for necessário. Elas servem para a equipe conversar sobre como foi o trabalho até aquele momento. Essas reuniões podem acontecer em diferentes momentos, dependendo de como o time organiza seu trabalho (modelo de desenvolvimento usado).

Participam da retrospectiva várias pessoas do time, como:

- Quem testa o sistema (Testadores),
- Quem desenvolve (Desenvolvedores),
- Arquitetos de software,Dono do produto (Product Owner),
- Analistas de negócios, entre outros.

ALINE FRANÇA

RETROSPECTIVAS E MELHORIA DE PROCESSOS

Nessas reuniões, o grupo discute:

- O que deu certo e deve continuar sendo feito da mesma forma?
- O que deu errado ou poderia ter sido melhor?
- O que podemos fazer diferente para melhorar da próxima vez e manter o que funcionou bem?
- Essas conversas são anotadas e geralmente viram parte de um relatório final do projeto ou da fase de testes.

Por que essas retrospectivas são importantes? Elas ajudam muito na ideia de melhoria contínua, que é fazer cada vez melhor. Mas não adianta só conversa, é importante acompanhar depois se as melhorias estão sendo aplicadas mesmo.

ALINE FRANÇA

RETROSPECTIVAS E MELHORIA DE PROCESSOS

Alguns benefícios que costumam aparecer com as retrospectivas:

- Os testes ficam mais eficientes, pois surgem boas ideias para melhorar os processos.
- Os materiais de teste (como roteiros e casos de teste) ficam mais bem feitos, graças às revisões feitas em grupo.
- O time aprende junto, porque todos têm a chance de falar o que pensam e sugerir melhorias.
- A qualidade do que será testado também melhora, porque problemas nos requisitos (as instruções do que o sistema deve fazer) são identificados e corrigidos.
- A comunicação entre quem desenvolve e quem testa melhora, já que a colaboração entre as áreas é avaliada e ajustada com frequência.

ALINE FRANÇA

- DÚVIDAS?

- SUGESTÕES?

- RECLAMAÇÃO?

- ELOGIO?

ALINE FRANÇA

VAMOS JUNTOS(A) NESSA!