Tópicos da Apresentação - Testes ao longo do Ciclo de Vida de Desenvolvimento de Software

Sequência:

1. *Lucas*
2. *Brenda*
3. *Gabriel*
4. *Eduarda*

Tópicos:

Lucas

2.1 Testes no contexto de um Ciclo de Vida de Desenvolvimento de Software

- Introdução: O SDLC é como um guia com etapas para criar um software do zero. Ele mostra o que precisa ser feito e em que ordem, como: planejamento, design, codificação, testes e entrega. garantindo que o projeto siga um caminho claro e oranizado do início ao fim.

2.1.1 Impacto do Ciclo de Vida de Desenvolvimento de Software nos Testes

- O SDLC impacta diretamente o planejamento e a execução dos testes, o impacto dependerá da escolha do modelo adotado, determinando como e com que intensidade os testes serão realizados, os prazos, os custos e até mesmo a colaboração entre os membros da equipe.

2.1.2 Ciclo de Vida de Desenvolvimento de Software e boas práticas de Teste

- Mesmo com diferentes modelos de SDLC, algumas boas práticas de teste são fundamentais. Para cada atividade de desenvolvimento, deve haver uma atividade de teste correspondente, garantindo cobertura e controle de qualidade em todas as etapas.
- Diferentes níveis de teste devem ser aplicados com objetivos distintos, como testes unitários, de integração e sistema, evitando redundância e aumentando a eficiência. Além disso, os testadores devem participar desde os primeiros rascunhos dos documentos, contribuindo para a identificação precoce de falhas.

2.1.3 Teste como um motivador para o desenvolvimento de software

- De forma resumida: significa usar os teste para guiar e melhorar o desenvolvimento do software, ajudando a escrever o código mais claro, seguro e focado nos requisitos desde o inicio.
- O TDD, ATDD e BDD são abordagens que podemos adotar. Essas três abordagens são motivadas por teste, e se aplicam nas fases iniciais do desenvolvimento, logo após o levantamento de requisitos e durante o planejamento e codificação. Elas seguem o princípio do teste antecipado e da estratégia shift-left.

Brenda

2.1.4 DevOps e Testes

- DevOps é uma abordagem organizacional que promove a junção de desenvolvimento e operações para alcançar um conjunto de objetivos.
- Promove várias questões, porém é importante destacar a automatização dos testes com CI (Implementação Continua) e CD (Entrega Contínua).

2.1.5 Abordagem Shift-Left

- Sugere que os testes devem ser feitos desde o inicio do SDLC (quanto antes melhor)
- Isso não significa que os testes posteriores devem ser negligenciados

2.1.6 Retrospectivas e melhoria de processos

- Com o intuito de avaliar o que foi bem sucedido e o que não foi
- Planejar o que pode ser feito para melhorar e como incorpora-los

Gabriel

2.2 Níveis de Teste e Tipos de Teste

2.2.1 Níveis de Teste

O que são:
Conjuntos organizados de atividades de teste, cada um com foco em diferentes estágios do desenvolvimento.

Principais Níveis:

- **Teste de Componente (Unidade):**
  Testa partes isoladas do código. Rápido, automatizado e feito por desenvolvedores.
- **Teste de Integração:**
  Valida a comunicação entre componentes ou sistemas. Realizado por devs ou testadores.
- **Teste de Sistema:**
  Verifica o sistema completo (end-to-end). Executado por testadores.
- **Teste de Aceite:**
  Valida o sistema com foco no uso real. Realizado por usuários, clientes e stakeholders.

Tipos de Aceite:

- **UAT:** Valida se atende ao usuário final.
- **OAT:** Valida operação e manutenção.
- **Contratual/Regulatório:** Valida exigências legais.
- **Alfa/Beta:** Feito dentro (alfa) ou fora (beta) da organização.

2.2.2 Tipos de Teste

Funcional:
Verifica **o que** o sistema faz (funções, regras de negócio).

Não Funcional:
Avalia **como** o sistema se comporta (desempenho, segurança, usabilidade).

Caixa Preta:
Foca nas entradas e saídas com base nas especificações. Não considera o código.

Caixa Branca:
Baseado na estrutura interna (código, lógica). Requer conhecimento técnico.

Eduarda

2.2.3 Testes de Confirmação

Situação comum:

Encontro um defeito → Reporto o problema → O time dev corrige → É gerado uma nova versão...

“ O que faço depois? “

- Reexecuto os testes

Etapas:

- Após encontrar e corrigir um defeito
- Reexecutar o teste que falhou
- Validar se o erro foi mesmo resolvido
- Verificar se não surgiram efeitos colaterais
  Reteste focado no erro que foi reportado.

2.2.3 Teste de Regressão

- Garante que funcionalidades antigas continuam funcionando
- Roda após correções ou novas features
- Ideal para automação, por serem repetitivos
  Toda mudança pode quebrar algo que já funcionava.

2.3 Teste de Manutenção (nível K2)

→ Feito com o sistema já em executação

Tipos:

- **Corretiva**: corrigir defeitos pós-produção
- **Adaptativa**: ajustes por mudanças no ambiente
- **Perfectiva**: melhorias de desempenho ou manutenção

  É importante sempre fazer análise de impacto antes de alterar algo

Em resumo:

- **Confirmação:** erro foi corrigido?
- **Regressão:** nada quebrou depois da mudança?
- **Manutenção:** o sistema continua funcionando bem?