UniSENAI

Trilha 02

Tipos de Testes

Instruções para a melhor prática de Estudo

1. **Leia atentamente todo o conteúdo:** Antes de iniciar qualquer atividade, faça uma leitura detalhada do material fornecido na trilha, compreendendo os conceitos e os exemplos apresentados.

2. **Não se limite ao material da trilha:** Utilize o material da trilha como base, mas busque outros materiais de apoio, como livros, artigos acadêmicos, vídeos, e blogs especializados. Isso enriquecerá o entendimento sobre o tema.

3. **Explore a literatura:** Consulte livros e publicações reconhecidas na área, buscando expandir seu conhecimento além do que foi apresentado. A literatura acadêmica oferece uma base sólida para a compreensão de temas complexos.

4. **Realize todas as atividades propostas:** Conclua cada uma das atividades práticas e teóricas, garantindo que você esteja aplicando o conhecimento adquirido de maneira ativa.

5. **Evite o uso de Inteligência Artificial para resolução de atividades:** Utilize suas próprias habilidades e conhecimentos para resolver os exercícios. O aprendizado vem do esforço e da prática.

6. **Participe de debates:** Discuta os conteúdos estudados com professores, colegas e profissionais da área. O debate enriquece o entendimento e permite a troca de diferentes pontos de vista.

7. **Pratique regularmente:** Não deixe as atividades para a última hora. Pratique diariamente e revise o conteúdo com frequência para consolidar o aprendizado.

8. **Peça feedback:** Solicite o retorno dos professores sobre suas atividades e participe de discussões sobre os erros e acertos, utilizando o feedback para aprimorar suas habilidades.

Essas instruções são fundamentais para garantir um aprendizado profundo e eficaz ao longo das trilhas.

1. E2E

E2E é a sigla para End-to-End, que em português significa de ponta a ponta. No contexto de desenvolvimento de software e testes, E2E refere-se a um tipo de teste ou abordagem que verifica o funcionamento completo de um sistema, simulando um cenário real de uso do início ao fim.

Exemplo: Verifica se um usuário consegue realizar uma compra com sucesso no site, garantindo que todas as interações e sistemas (frontend, backend, banco de dados, API de pagamento) funcionem corretamente.

---

2. Testes Unitários

Os testes unitários são testes automatizados escritos por desenvolvedores para validar a funcionalidade de um único módulo, classe ou função de um sistema. Eles são executados independentemente de outras partes do software e normalmente fazem parte do processo de desenvolvimento ágil.

Exemplo: Imagine que você está desenvolvendo um sistema de e-commerce, e uma das funcionalidades é calcular o valor total de um pedido. Essa função deve somar os preços dos produtos e aplicar um desconto, caso exista.

Cenário:

Uma função chamada **calcularTotalPedido** recebe uma lista de produtos e um percentual de desconto. Ela deve retornar o valor final do pedido.

Para garantir que essa função funcione corretamente em diferentes situações, é necessário aplicar testes unitários. Isso ajudará a verificar se:

- O cálculo da soma dos produtos é feito corretamente.
- O desconto é aplicado corretamente ao valor total.
- O comportamento da função está correto para pedidos vazios ou com valores inválidos.

---

3. Testes de Integração

Os testes de integração verificam a interação entre diferentes partes do software, como módulos, serviços, APIs e banco de dados. Eles garantem que a comunicação entre esses componentes funcione conforme o esperado, identificando problemas de compatibilidade e inconsistências.

Exemplo: Imagine que você está desenvolvendo um sistema de e-commerce, uma das funcionalidades é a **finalização de um pedido**, uma etapa essencial para garantir uma boa experiência de compra para os clientes. Esse processo precisa ser rápido, seguro e

eficiente, pois qualquer falha pode gerar frustração para o usuário e prejuízo para a empresa.

A finalização de um pedido envolve **diversas etapas**, que precisam funcionar de forma integrada para garantir que a compra seja concluída com sucesso. Primeiro, o sistema deve **verificar a disponibilidade dos produtos no estoque**, evitando que um cliente compre um item que não está mais disponível. Em seguida, é necessário **calcular o valor total da compra**, considerando o preço dos produtos, taxas adicionais, descontos e possíveis promoções.

Depois dessa etapa, o sistema deve **processar o pagamento**, garantindo que o valor seja cobrado corretamente e que a transação seja validada de forma segura. Caso o pagamento seja aprovado, o pedido deve ser **confirmado e registrado no sistema**, gerando um número de pedido e notificando o cliente sobre a compra. Além disso, o estoque precisa ser atualizado para refletir a quantidade real de produtos disponíveis.

---

4. Testes Funcionais e Não Funcionais

Testes Funcionais: Diferente dos testes unitários e de integração, que analisam componentes isolados e sua comunicação, os testes funcionais validam o comportamento geral do sistema do ponto de vista do usuário. Eles são cruciais para garantir que as funcionalidades essenciais do software funcionem corretamente antes da entrega.

Exemplo: Testar se um novo usuário pode se registrar corretamente na tela de Cadastros de Usuário.

Testes Não Funcionais: Os testes não funcionais avaliam aspectos de qualidade do software que não estão relacionados às funcionalidades, mas sim ao desempenho, segurança, usabilidade, compatibilidade, entre outros. Eles garantem que o sistema atenda aos requisitos de qualidade exigidos pelos usuários e pelo mercado.

Exemplo: Um aplicativo bancário precisa ser testado para garantir que suporte múltiplos acessos simultâneos sem travar.

---

5. Testes de Aceitação

Os testes de aceitação são realizados para validar se o sistema atende aos critérios de aceitação estabelecidos pelos usuários finais ou clientes. Eles garantem que o software funciona conforme esperado e está pronto para ser entregue.

Exemplo: Imagine que você está desenvolvendo um sistema de e-commerce e precisa validar se o processo de finalização de compra está funcionando corretamente do ponto de vista do usuário.

O cliente definiu o seguinte critério de aceitação:

- "Como um usuário do e-commerce, quero adicionar produtos ao carrinho, inserir meus dados de pagamento e receber uma confirmação da compra para garantir que meu pedido foi processado corretamente."

Critérios de Aceitação:

- O usuário consegue adicionar produtos ao carrinho sem erro.
- O total da compra é calculado corretamente.
- O sistema processa o pagamento e retorna uma resposta válida.
- O pedido é registrado no banco de dados.
- O usuário recebe um e-mail de confirmação do pedido.

---

6. Testes de Sistema

Os testes de sistema são uma fase fundamental do ciclo de vida do desenvolvimento de software, nos quais o sistema completo é testado para verificar se atende aos requisitos especificados. Esses testes avaliam o comportamento do software em um ambiente semelhante ao de produção, considerando tanto os requisitos funcionais quanto os não funcionais, como desempenho, segurança e compatibilidade.

O objetivo principal dos testes de sistema é garantir que o software funcione conforme o esperado quando integrado em um ambiente completo. Ele ajuda a identificar falhas que podem não ser detectadas em testes unitários ou de integração, pois considera a interação entre diferentes módulos e componentes do sistema.

Exemplo: Imagine que você está desenvolvendo um sistema onde novos usuários podem se registrar criando uma conta. O teste de sistema precisa validar se todas as partes envolvidas no processo funcionam corretamente juntas.

Pré-requisitos:

- O sistema está rodando em um ambiente de testes semelhante ao de produção.
- O banco de dados está configurado e acessível.
- As APIs de autenticação e envio de e-mails estão funcionando.

Passos do Teste:

- Acessar a página de cadastro de usuário.
- Preencher os campos obrigatórios: Nome, E-mail, Senha e Confirmação de Senha.
- Clicar no botão "Criar Conta".
- O sistema deve validar os dados e verificar se o e-mail já existe.
- Se os dados estiverem corretos, o usuário deve ser registrado no banco de dados.
- O sistema deve enviar um e-mail de confirmação para o novo usuário.
- O usuário deve receber o e-mail e conseguir ativar sua conta clicando no link enviado.
- Após ativar a conta, o usuário deve conseguir fazer login com suas credenciais.

---

7. Testes de Usabilidade

O teste de usabilidade avalia a facilidade de uso, eficiência e experiência do usuário ao interagir com um sistema ou aplicativo. O objetivo é identificar barreiras que dificultam a navegação e melhorar a interface para torná-la mais intuitiva.

Para garantir que os usuários consigam usar o sistema de forma intuitiva e sem dificuldades. Para identificar problemas de navegação, acessibilidade e design antes do lançamento. Para aumentar a satisfação e retenção dos usuários.

Exemplo: Um banco lança um novo aplicativo mobile e quer garantir que os clientes consigam realizar transferências bancárias de forma fácil e rápida.

Teste aplicado: Um teste de usabilidade é conduzido com um grupo de usuários reais, que tentam completar uma transferência. Durante o teste, a equipe observa dificuldades, mede o tempo necessário para concluir a ação e coleta feedback.

Se muitos usuários demoram para encontrar a opção de transferência ou cometerem erros, ajustes na interface podem ser feitos para tornar o processo mais intuitivo.

---

8. Testes de Desempenho

Os testes de performance avaliam a rapidez, estabilidade, escalabilidade e capacidade de resposta de um sistema sob diferentes condições de carga. O objetivo é garantir que o software funcione de maneira eficiente mesmo em situações de alto uso.

Exemplo: Um e-commerce realiza promoções sazonais e precisa garantir que seu site aguente um grande volume de acessos sem travar.

Teste aplicado: Um teste de carga é realizado simulando milhares de usuários acessando o site ao mesmo tempo. A equipe mede o tempo de resposta, o uso de recursos do servidor e a estabilidade da aplicação.

Se houver lentidão ou falhas, ajustes são feitos na infraestrutura e no código para garantir que a plataforma funcione corretamente mesmo em picos de acesso.

---

9. Testes de Segurança

Os testes de segurança avaliam a capacidade do software de resistir a ataques maliciosos, proteger dados sensíveis e garantir a integridade das informações. Eles verificam vulnerabilidades, falhas de autenticação, exposição de dados e outros riscos que possam comprometer a segurança do sistema.

Exemplo: Uma fintech lança um novo aplicativo bancário e precisa garantir que os dados dos clientes estejam protegidos contra acessos não autorizados.

Teste aplicado: A equipe realiza um teste de penetração (pentest), simulando ataques hackers para tentar invadir o sistema. O objetivo é identificar falhas de segurança, como senhas fracas, falhas de autenticação ou exposição de dados sensíveis.

Se uma vulnerabilidade for descoberta, a equipe pode corrigi-la antes que um atacante real a explore, garantindo a segurança dos usuários.

---

10. Testes de Regressão

O teste de regressão tem o objetivo de verificar se uma nova alteração no software não causou impactos negativos em funcionalidades já existentes. Ele garante que correções de bugs, novas features ou atualizações não introduzam novos defeitos.

Exemplo: Uma equipe de desenvolvimento adiciona um novo campo de "cupom de desconto" no checkout de um e-commerce.

Teste aplicado: Antes de lançar a atualização, um teste de regressão é realizado para garantir que a nova funcionalidade não quebrou o fluxo de pagamento, cálculo de valores ou a exibição dos produtos no carrinho.

Sem esse teste, o risco de um erro crítico afetar as compras dos usuários seria alto, impactando a receita e a experiência do cliente.

---

11. Testes de Caixa-Preta e Caixa-Branca

Aqui está um comparativo entre Teste de Caixa Preta e Teste de Caixa Branca, incluindo suas descrições, motivos para utilização, vantagens, desvantagens e exemplos práticos.

Teste de Caixa Preta: O teste de caixa preta avalia a funcionalidade do software sem considerar sua implementação interna. O testador fornece entradas e analisa as saídas para verificar se estão corretas.

Exemplo: Neste caso, o testador não vê o código-fonte, apenas interage com o sistema como um usuário comum para verificar se a funcionalidade funciona conforme esperado.

Teste de Caixa Branca: O teste de caixa branca analisa a lógica e estrutura interna do código. O testador verifica fluxos de controle, condições e loops para garantir que o código funcione corretamente.

Exemplo: Como o testador tem acesso ao código, ele pode criar testes unitários para cobrir diferentes caminhos de execução de alguma função, como por exemplo **validarSenha**.

Lista de Exercícios de Fixação

Questões Dissertativas

1. Explique o que são os testes End-to-End (E2E) e sua importância no processo de garantia de qualidade de um sistema. Dê um exemplo prático de quando esse tipo de teste deve ser aplicado.
2. Qual a principal vantagem dos testes unitários no desenvolvimento de software? Explique como esses testes podem ajudar a reduzir custos e retrabalho no desenvolvimento de um sistema.
3. Por que os testes de integração são essenciais para o funcionamento adequado de um sistema? Explique um possível problema que pode ocorrer quando os testes de integração não são realizados corretamente.
4. Qual é a principal diferença entre testes funcionais e testes não funcionais?
5. Os testes de aceitação são fundamentais para validar se um sistema atende aos requisitos do usuário final. Explique como esses testes são conduzidos e por que a participação do cliente é essencial nesse processo.
6. Qual a importância dos testes de sistema no desenvolvimento de software?
7. Por que os testes de usabilidade são essenciais para a experiência do usuário? Explique como esses testes são aplicados e como os resultados podem influenciar melhorias no design e na funcionalidade de um sistema.
8. Os testes de desempenho ajudam a avaliar a estabilidade e a eficiência de um sistema sob diferentes condições de carga. Explique os principais tipos de testes de desempenho e como eles podem evitar falhas em momentos críticos.
9. A segurança da informação é uma preocupação crescente no desenvolvimento de software. Explique como os testes de segurança ajudam a prevenir vulnerabilidades e quais são as técnicas mais comuns utilizadas nesses testes.
10. Explique a diferença entre testes de caixa preta e testes de caixa branca.