UniSENAI

Trilha 03

Técnicas 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.

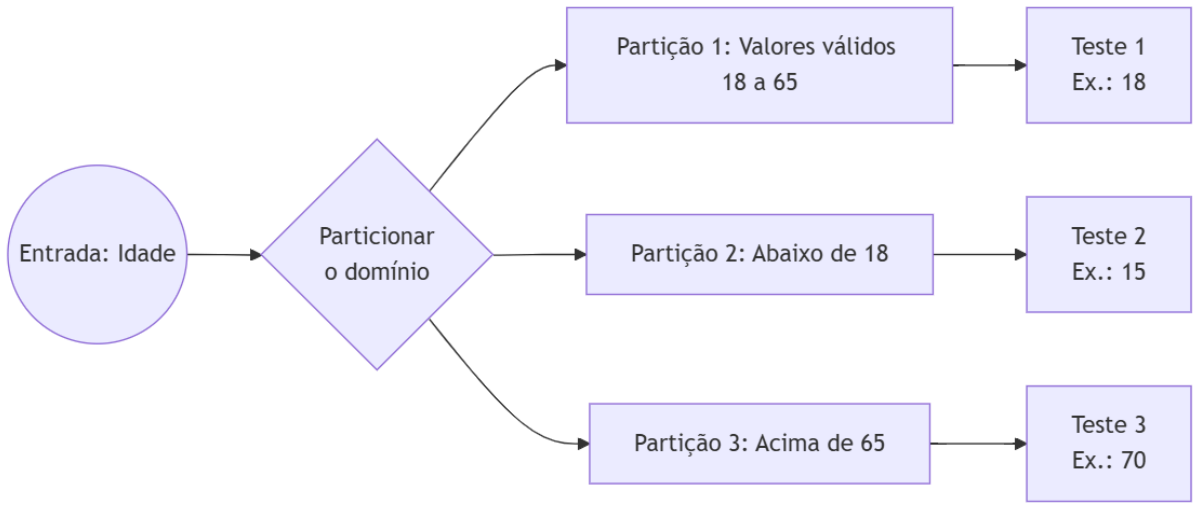
1. Partição de Equivalência

A Partição de Equivalência é uma técnica de teste de software que se baseia na ideia de que, se um programa funciona corretamente para um valor de um conjunto, ele provavelmente funcionará para outros valores desse mesmo conjunto. A ideia é dividir os dados de entrada em partições de equivalência de tal forma que, se o sistema se comporta corretamente para um valor de uma partição, ele se comportará corretamente para todos os valores dessa partição.

Exemplo: Vamos supor que estamos testando uma função de cadastro de idade em um sistema que só permite cadastros de pessoas com idades entre 18 e 65 anos. A função de cadastro poderia ser testada utilizando a partição de equivalência da seguinte forma:

- **Idade válida (18 a 65 anos)**: Esta é uma partição de equivalência válida, onde qualquer valor entre 18 e 65 deve ser aceito pelo sistema.
- **Idade inválida (abaixo de 18 anos)**: Uma partição de equivalência inválida, onde qualquer valor abaixo de 18 anos não deve ser aceito.
- **Idade inválida (acima de 65 anos)**: Uma outra partição inválida, onde qualquer valor acima de 65 anos não deve ser aceito.

Exemplo Visual:



2. Análise de Valor Limite

A Análise de Valor Limite é uma técnica de teste de software que visa identificar erros em pontos críticos de um sistema, que são os valores nos limites de entrada. A técnica foca em testar os valores extremos das entradas para garantir que o sistema se comporte corretamente dentro e fora dos limites especificados. O princípio da BVA é que os erros são mais propensos a ocorrer nas bordas ou valores próximos aos limites das variáveis.

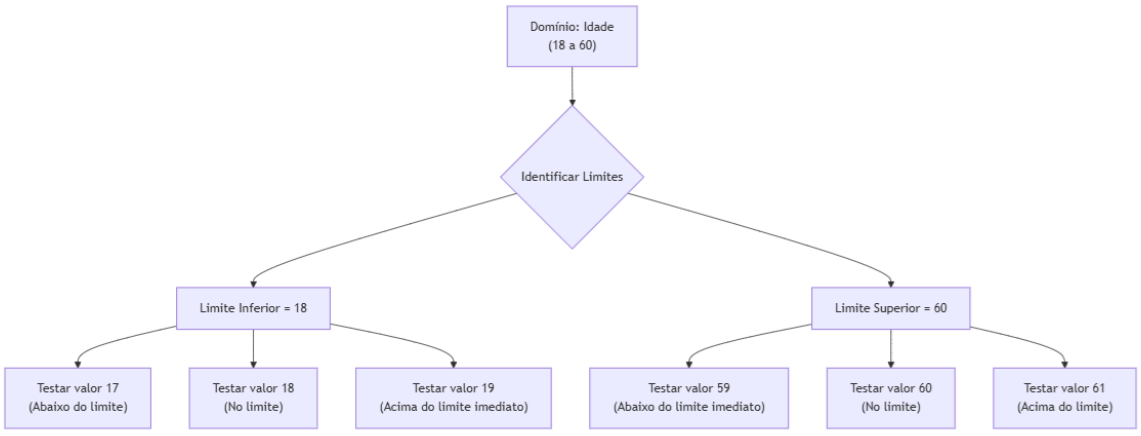
A análise de valores limite é baseada na ideia de que, se um sistema se comporta corretamente nas extremidades (limites) de um intervalo de dados, ele provavelmente também funcionará bem nos valores internos.

Exemplo: Vamos supor que estamos testando uma funcionalidade de cadastro de um sistema que solicita a idade de um usuário. O campo de idade permite valores entre 18 e 60 anos (inclusive). A Análise de Valor Limite ajudaria a testar as bordas desse intervalo para garantir que o sistema funcione corretamente.

- Valor mínimo válido: 18 anos (limite inferior).
- Valor mínimo inválido: 17 anos (abaixo do limite inferior).
- Valor máximo válido: 60 anos (limite superior).
- Valor máximo inválido: 61 anos (acima do limite superior).
- Valor dentro do intervalo: 30 anos (valor arbitrário dentro do intervalo válido).
- Com esses cenários, a BVA ajuda a garantir que o sistema não tenha erros quando o valor da idade está na borda ou fora dela.

Exemplo Visual:

Nesse fluxo testamos algumas variantes para melhor simular o exemplo descrito acima, com isso, podemos elaborar melhor o fluxo de testes a serem seguidos.



3. Testes de Casos de Uso

O Teste de Caso de Uso é uma técnica utilizada para testar a funcionalidade de um sistema, validando que ele atende aos requisitos de negócios e objetivos do usuário. O objetivo do teste de caso de uso é garantir que o sistema funcione conforme esperado nas situações reais de uso. Ele foca em interações e processos completos do ponto de vista do usuário, incluindo as condições e caminhos alternativos possíveis.

Exemplo: Validar se o sistema permite que um usuário acesse a plataforma inserindo um nome de usuário e senha válidos, segue o fluxo:

- O usuário abre a página de login.
- O usuário insere seu nome de usuário e senha.
- O sistema verifica a validade das credenciais.
- Se as credenciais forem válidas, o sistema direciona o usuário para a página principal.

---

4. Tabela de Decisão

A Tabela de Decisão de Uso é uma técnica de modelagem usada para representar de forma clara e estruturada os diferentes cenários de tomada de decisão dentro de um sistema ou processo. Cada linha da tabela representa uma combinação específica de condições e as ações correspondentes que devem ser tomadas com base nessas condições. Esse método facilita a visualização e a análise de sistemas complexos que envolvem várias condições e ações.

Exemplo: Imagine que um sistema de vendas aplica descontos de acordo com o tipo de cliente e o valor da compra. A Tabela de Decisão seria usada para modelar as regras de decisão sobre o valor do desconto.

Condições:

- Cliente VIP (Sim/Não)
- Valor da Compra (Maior que 100 reais/ Menor que 100 reais)

Ações:

- Aplicar 20% de desconto.
- Aplicar 10% de desconto.
- Não aplicar desconto.

Tabela de Decisão de Uso para Cálculo de Desconto:

Condição	Cliente VIP	Valor da Compra > 100	Ação (Desconto)
Cenário 1	Sim	Sim	20% de desconto
Cenário 2	Sim	Não	10% de desconto
Cenário 3	Não	Sim	10% de desconto
Cenário 4	Não	Não	Nenhum desconto

5. Teste de Mutação

O Teste de Mutação é uma técnica de teste de software usada para avaliar a qualidade e eficácia de um conjunto de testes automatizados. Ele envolve a introdução de pequenas modificações no código-fonte do sistema (chamadas de "mutações") e a execução dos testes para verificar se os testes existentes são capazes de detectar essas falhas introduzidas. O objetivo é avaliar o "poder de detecção" dos testes, ou seja, a capacidade dos testes em identificar erros quando o código é ligeiramente alterado.

Exemplo Prático

Caso de Uso: "Cálculo de Imposto de Venda"

Vamos considerar um sistema de cálculo de imposto de venda, onde a regra é aplicar 10% de imposto sobre o valor total da compra. A função de cálculo de imposto é simples:

```
function imposto(valor) {
    if(valor > 100) {
        return valor * 0.10;
    } else {
        return valor * 0.05;
    }
}
```

Agora, vamos aplicar o Teste de Mutação para melhorar a cobertura de testes e avaliar a eficácia dos testes existentes.

- Mutação 1 - Mudança no valor do imposto: Modificamos o valor do imposto de 0.10 para 0.15 na condição de **valor > 100**.

```
function imposto(valor) {
    if(valor > 100) {
        return valor * 0.15; // Mudança do imposto de 10% para 15%
    } else {
        return valor * 0.05;
    }
}
```

Passo a Passo para Teste de Mutação:

Teste Original: Escreva testes unitários para verificar o cálculo do imposto:

- Se o valor for maior que 100, o imposto deve ser 10%.
- Se o valor for menor ou igual a 100, o imposto deve ser 5%.

```
const { imposto } = require('../src/math');

Add only
describe('Cáculo de Imposto de Venda', () => {
    Add only
    it('Apenas um teste.', () => {
        expect(imposto(150)).toBe(15);
    });
});
```

Aplicando o Teste de Mutação: Após introduzir as mutações no código, execute os testes:

- Se os testes passarem para a mutação 1 (valor de imposto de 15%), significa que o teste detecta a falha.

Analisando os Resultados:

- Se o teste falhar em algumas mutações, é necessário ajustar ou adicionar novos testes para melhorar a cobertura.

---

6. Testes Exploratórios

O Teste Exploratório é uma abordagem de teste de software onde os testadores exploram livremente o sistema, sem um conjunto de testes previamente definido. Em vez de seguir um roteiro rigoroso de testes, o testador utiliza seu conhecimento do sistema, intuição e experiência para identificar falhas e descobrir comportamentos inesperados. O foco principal do Teste Exploratório é entender o funcionamento do sistema e descobrir possíveis erros, em vez de simplesmente validar que os requisitos estão sendo atendidos.

Exemplo: Suponha que você está testando um aplicativo de login que permite aos usuários se autenticarem com nome de usuário e senha. O sistema tem algumas funcionalidades básicas de autenticação e recuperação de senha.

Passos do Teste Exploratório:

Exploração Inicial: Comece fazendo o login com credenciais válidas e inválidas, sem ter um roteiro predefinido. Tente casos como:

- Inserir nome de usuário válido e senha incorreta.
- Inserir nome de usuário incorreto e senha válida.
- Testar a recuperação de senha com um e-mail válido e inválido.
- Verificar o comportamento do sistema quando o campo de senha é deixado em branco.
- Inserir caracteres especiais ou muito longos no nome de usuário e senha.

Exploração de Condições de Erro: Durante o processo de login, tente explorar o comportamento do sistema em condições extremas, como:

- Inserir dados muito longos (ex: 1000 caracteres no nome de usuário).
- Deixar o nome de usuário ou senha em branco.
- Testar a recuperação de senha com e-mail não registrado no sistema.
- Testar o login simultâneo em vários dispositivos com a mesma conta de usuário.

Testando o Fluxo do Sistema: Em seguida, explore como o sistema lida com o comportamento do usuário em cenários reais, como:

- Testar a resposta do sistema quando há uma falha na conexão com a internet.
- Explorar o tempo de resposta do sistema após múltiplas tentativas de login com falhas.

Lista de Exercícios de Fixação

Questões Dissertativas

1. Explique o conceito de Partição de Equivalência e como essa técnica pode reduzir a quantidade de testes necessários sem comprometer a qualidade da validação do sistema.
2. Por que a Análise de Valor Limite é considerada uma abordagem eficiente para identificar erros em sistemas? Explique sua importância e forneça um exemplo prático onde essa técnica poderia ser aplicada para evitar falhas em um software de validação de idade.
3. Os Testes de Casos de Uso são utilizados para validar funcionalidades do sistema do ponto de vista do usuário. Explique como esse tipo de teste é estruturado e qual a sua importância na garantia da experiência do usuário.
4. Qual é a principal vantagem de utilizar uma Tabela de Decisão em testes de software? Explique como essa técnica pode ajudar na modelagem de regras de negócio e dê um exemplo de como poderia ser aplicada em um sistema de concessão de descontos para clientes.
5. Os Testes Exploratórios são uma abordagem dinâmica e flexível para a detecção de falhas. Explique como essa técnica se diferencia dos testes tradicionais e cite três vantagens desse tipo de teste.