Teste de Software

Professor: Euclides Paim
euclidespaim@gmail.com

Testes de Integração

Professor: Euclides Paim
euclidespaim@gmail.com

Teste de Software

Testes de integração

Sumário

- Objetivos
- Definição
- Importância
- Tipos
- Exemplos

Teste de Software

Testes de integração

Objetivos da Aula

Nesta aula, vamos entender o que são testes de integração, qual a sua importância no desenvolvimento de software e como eles se diferenciam dos testes unitários. Também veremos os principais tipos de integração, as abordagens mais comuns para aplicar os testes e exemplos práticos que ajudam a consolidar o conteúdo.

Teste de Software

Testes de integração

Recapitulando o que já vimos

- Fundamentos: garantir qualidade, detectar erros cedo, reduzir custo.
- Aspectos psicológicos: viés do desenvolvedor, resistência a testes.
- Mandamentos do testador: pensar como o usuário, buscar falhas, repetir testes.
- Tipos de testes: caixa branca/preta, unitário, integração, sistema, aceitação etc.

Teste de Software

Testes de integração

O que são Testes de Integração

Testes de integração são uma abordagem de teste de software que visa verificar se diferentes módulos ou componentes de um sistema funcionam corretamente quando combinados. Em outras palavras, eles garantem que as interfaces entre esses componentes e suas interações estejam corretas e que os dados sejam transferidos sem problemas entre eles.

Teste de Software

Testes de integração

Por que são Importantes

Mesmo que os módulos funcionem bem individualmente (testes unitários), problemas podem surgir quando eles precisam interagir. Testes de integração ajudam a detectar falhas na comunicação entre funções, classes, APIs ou até sistemas inteiros, garantindo mais estabilidade e confiança na aplicação.

Teste de Software

Testes de integração

Diferença entre Teste Unitário e de Integração

O teste unitário foca em uma parte isolada do código, como uma função ou método. Já o teste de integração avalia a combinação de duas ou mais partes, testando o comportamento coletivo. Ambos são essenciais, mas cumprem papéis diferentes dentro do processo de garantia de qualidade.

Tipos de Integração

A integração pode ser feita de diversas formas. A mais comum é a integração entre módulos internos do sistema, mas também é possível testar integrações com bancos de dados, APIs externas, bibliotecas de terceiros e até com interfaces gráficas. Cada uma exige uma abordagem específica de testes.

Abordagens de Teste de Integração

Existem várias abordagens para conduzir testes de integração: de cima para baixo (testa os módulos superiores primeiro), de baixo para cima (começa pelos módulos mais básicos), big bang (testa tudo de uma vez) e abordagem híbrida. Cada estratégia tem vantagens e desvantagens conforme o contexto do projeto.

Ferramentas de Apoio

No desenvolvimento moderno, várias ferramentas ajudam na automação dos testes de integração, como JUnit, pytest, Mocha, Postman e ferramentas de CI/CD como GitHub Actions e Jenkins. Elas ajudam a garantir que os testes sejam executados continuamente e de forma confiável.

Exemplos Práticos

Um exemplo comum de teste de integração é verificar se uma função de login consegue se conectar corretamente ao banco de dados e autenticar um usuário. Outro exemplo é testar se uma API retorna os dados corretos ao ser chamada por um front-end. Esses testes revelam problemas que não aparecem nos testes unitários.

Teste de Software

Testes de integração

Resumo

Testes de integração verificam se os módulos de um sistema funcionam corretamente quando combinados. Eles detectam **falhas na comunicação entre partes do software**, complementam os testes unitários e ajudam a garantir a estabilidade do sistema. Conhecer suas abordagens e aplicar ferramentas adequadas é essencial para o desenvolvimento profissional de software.

Referências

Referências Básicas

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. ***Algoritmos: teoria e prática*****.** 3. ed. Rio de Janeiro: Elsevier, 2013.

SEBESTA, Robert W. ***Conceitos de linguagens de programação***. 10. ed. São Paulo: Pearson, 2018.

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. ***Algoritmos: lógica para desenvolvimento de programação de computadores*****.** 27. ed. São Paulo: Érica, 2016.

DOWNEY, Allen B. ***Pense em Python: como pensar como um cientista da computação***. Tradução de Cássio de Souza Costa. 2. ed. São Paulo: Novatec, 2016.

Referências Complementares

IEPSEN, Edécio Fernando. ***Lógica de programação e algoritmos com JavaScript***. 2. ed. São Paulo: Novatec, 2022.

Referências na Internet

https://www.w3schools.com/python/