Gustavo Fratoni Boeing

Desn 2024 2V1

15/07/2025

TESTES UNITÁRIOS

1. Quem faz esse tipo de teste?
Os desenvolvedores são os principais responsáveis por criar e executar os testes unitários, pois eles se concentram em pequenas partes isoladas do código, como funções, métodos ou classes.

2. Em que fase isso acontece?
Os testes unitários são realizados durante a fase de desenvolvimento, geralmente seguindo práticas como:

TDD (Desenvolvimento Orientado a Testes) – Os testes são escritos antes do código.

Integração Contínua (CI) – Os testes são executados automaticamente em pipelines de CI/CD.

3. Qual é o objetivo desse tipo de teste?
Verificar se as unidades de código funcionam isoladamente.

Garantir que cada parte do sistema opere corretamente antes da integração.

Facilitar a identificação rápida de bugs nas etapas iniciais.

Servir como uma documentação viva do comportamento esperado do código.

4. Exemplos de ferramentas para aplicar TDD
Algumas ferramentas populares para TDD e testes unitários incluem:

JUnit (Java)

pytest (Python)

Jest (JavaScript/Node.js)

RSpec (Ruby)

PHPUnit (PHP)

NUnit (.NET)

Sistemas e Testes de Aceitação: Ferramentas
Como mencionado, os testes de sistema e aceitação avaliam a aplicação já desenvolvida em um contexto mais amplo, simulando o uso real. Algumas ferramentas comuns são:

1. Selenium
O que testa?

Automação de testes em navegadores (Teste de UI).

Valida interações do usuário em páginas web (cliques, formulários, etc.).

2. Cypress
O que testa?

Testes E2E (End-to-End) em aplicações web.

Mais moderno que o Selenium, com execução direta no navegador.

3. Postman/Newman
O que testa?

Testes de APIs (requisições HTTP, respostas, autenticação).

4. Cucumber
O que testa?

Testes baseados em comportamento (BDD - Desenvolvimento Orientado a Comportamento).

Usa linguagem natural (ex.: Gherkin) para descrever cenários.

5. JMeter
O que testa?

Testes de carga e desempenho (como o sistema se comporta sob tráfego intenso).

6. Appium
O que testa?

Testes de aceitação em aplicativos móveis (iOS e Android).

TESTES DE INTEGRAÇÃO

1. Quem faz esse tipo de teste?
Os desenvolvedores e engenheiros de QA (Quality Assurance) são os responsáveis pelos testes de integração, geralmente trabalhando em conjunto, já que isso envolve checar a comunicação entre módulos, serviços ou sistemas.

2. Em que etapa do processo isso acontece?
Os testes de integração são realizados depois dos testes unitários e antes dos testes de sistema, em fases como:

Desenvolvimento contínuo (quando novos módulos são integrados).

Pipelines de CI/CD (usando ferramentas como Jenkins, GitHub Actions).

Preparação para homologação (antes dos testes de aceitação).

3. Qual é o propósito desse tipo de teste?
Verificar se módulos ou serviços independentes funcionam corretamente quando são combinados.

Identificar erros de comunicação (por exemplo: APIs, bancos de dados, chamadas entre serviços).

Assegurar que os contratos entre componentes (como schemas de APIs) sejam respeitados.

Validar fluxos de negócio que dependem de várias partes do sistema.

4. Exemplos de ferramentas para testes de integração
Algumas ferramentas populares incluem:

JUnit + Mockito (Java, para simular dependências).

pytest (Python, com fixtures para integração).

Postman/Newman (testes de APIs REST).

TestContainers (testes com bancos de dados e serviços em containers).

RestAssured (Java, para APIs HTTP).

Ferramentas para Testes de Integração
1. Postman
O que testa?

Chamadas HTTP entre serviços (APIs).

Valida respostas, códigos de status e contratos (como JSON Schema).

2. TestContainers
O que testa?

Integração com bancos de dados (PostgreSQL, MySQL), filas (Kafka) e outros serviços em containers Docker.

3. WireMock
O que testa?

Simula APIs externas (mocking) para testar integrações sem depender de sistemas reais.

4. Pact
O que testa?

Testes de contrato entre serviços (Consumer-Driven Contracts), garantindo que sejam compatíveis.

5. Cypress (para integração front-end/back-end)
O que testa?

Chamadas de API a partir da interface do usuário e respostas renderizadas no navegador.

TESTES DE SISTEMAS

1. Quem faz esse tipo de teste?
Os testadores de QA (Quality Assurance) ou engenheiros de teste são os principais responsáveis, mas em equipes ágeis, os desenvolvedores também podem entrar na dança. O objetivo é garantir que o sistema funcione perfeitamente de acordo com os requisitos.

2. Em que fase ele é realizado?

Os testes de sistema acontecem depois dos testes de integração e antes dos testes de aceitação, em etapas como:

- Fase de testes dedicada (em um ambiente de staging ou homologação).
- Pipelines de CI/CD (antes do deploy em produção).
- Preparação para release (validação final antes da entrega).

3. Qual é o objetivo desse tipo de teste?
Verificar se o sistema, como um todo, atende aos requisitos funcionais e não funcionais.

Validar fluxos completos de usuário (por exemplo: cadastro → login → compra).

Testar performance, segurança e usabilidade em condições reais.

Garantir que a aplicação funcione no ambiente de implantação (não apenas em desenvolvimento).

4. Exemplos de ferramentas para testes de sistema
Algumas ferramentas populares incluem:

- Selenium WebDriver (automação de UI em navegadores).
- Cypress (testes E2E modernos para web).
- JMeter (testes de carga e performance).
- OWASP ZAP (testes de segurança).
- SoapUI (testes de APIs SOAP/REST).

Ferramentas para Testes de Sistema

1. Selenium WebDriver
O que testa?
Interações de usuário em interfaces web (cliques, formulários, navegação).

Compatibilidade entre navegadores (Chrome, Firefox, Edge).

2. Cypress
O que testa?
Testes end-to-end (E2E) com execução em tempo real no navegador.

Valida desde APIs até a renderização da UI.

3. JMeter
O que testa?
Performance e escalabilidade (por exemplo: como o sistema se comporta com 1.000 usuários simultâneos).

Carga em APIs, bancos de dados e páginas web.

4. OWASP ZAP
O que testa?
Vulnerabilidades de segurança (SQL injection, XSS, tokens expostos).

5. SoapUI
O que ele testa?

APIs REST/SOAP, incluindo autenticação e contratos de resposta.

Resumo das Diferenças
Tipo de Teste	Escopo	Exemplo de Ferramenta
Unitário	Código isolado (funções/métodos)	JUnit, pytest
Integração	Comunicação entre componentes	Postman, TestContainers
Sistema	Sistema completo (UI, APIs, perf.)	Selenium, JMeter
Aceitação	Requisitos do usuário final	Cucumber, UAT manuais

Quando usar testes de sistema?
Para validar cenários completos (por exemplo, "um usuário deve conseguir finalizar uma compra").

Antes de entregas em produção (para garantir que tudo funcione no ambiente real).

Para testes não funcionais (como segurança, performance e acessibilidade).

TESTES DE ACEITAÇÃO

1. Quem faz esse tipo de teste?
Os testadores de QA, os Product Owners (POs) e até mesmo os usuários finais, especialmente durante os testes de aceitação do usuário (UAT). O principal objetivo é garantir que o sistema atenda às expectativas tanto do negócio quanto dos clientes.

2. Em que fase do processo isso acontece?
Os testes de aceitação são realizados:

No final do ciclo de desenvolvimento, logo após os testes de sistema.

Em ambientes que imitam a produção (homologação/UAT).

Podem ser parte de entregas ágeis (no final de cada sprint) ou em projetos mais tradicionais (antes do lançamento).

3. Qual é a finalidade desse tipo de teste?
Validar se o sistema cumpre os requisitos de negócio (não apenas os técnicos).

Assegurar que a solução realmente resolve o problema do usuário.

Verificar se está em conformidade com regras específicas (por exemplo, validações de um fluxo financeiro).

Garantir que a experiência do usuário (UX) seja satisfatória.

Ferramentas para Testes de Aceitação
1. Cucumber (+ Gherkin)
O que testa?

Fluxos baseados em regras de negócio, utilizando uma linguagem natural (ex.: "Dado que um usuário está logado, Quando ele adiciona um produto ao carrinho, Então o valor total deve ser atualizado").

Perfeito para BDD (Desenvolvimento Orientado a Comportamento).

2. Selenium (para automação de UAT)
O que testa?

Cenários completos de interface do usuário que simulam as ações do usuário final (ex.: preenchimento de formulários, navegação entre telas).

3. Postman/Newman (para APIs)
O que testa?

APIs em cenários de aceitação (ex.: "Ao enviar um pedido, o status deve mudar para 'processando'").

4. Jira/Xray (gestão de UAT)
O que testa?

Não é uma ferramenta de execução, mas ajuda a organizar casos de teste e aprovações dos clientes.

5. TestRail
O que testa?

Gerenciamento de casos de teste manuais e acompanhamento das aprovações.