TESTES DE SOFTWARE

Bibliotecas de Testes Unitários e Abordagens de Teste

Integrantes: Luiza Fernanda, Rayanne Conde, Urbano Neto, Moacyr Junio, Kalyl Cordeiro, Lucas Cesar

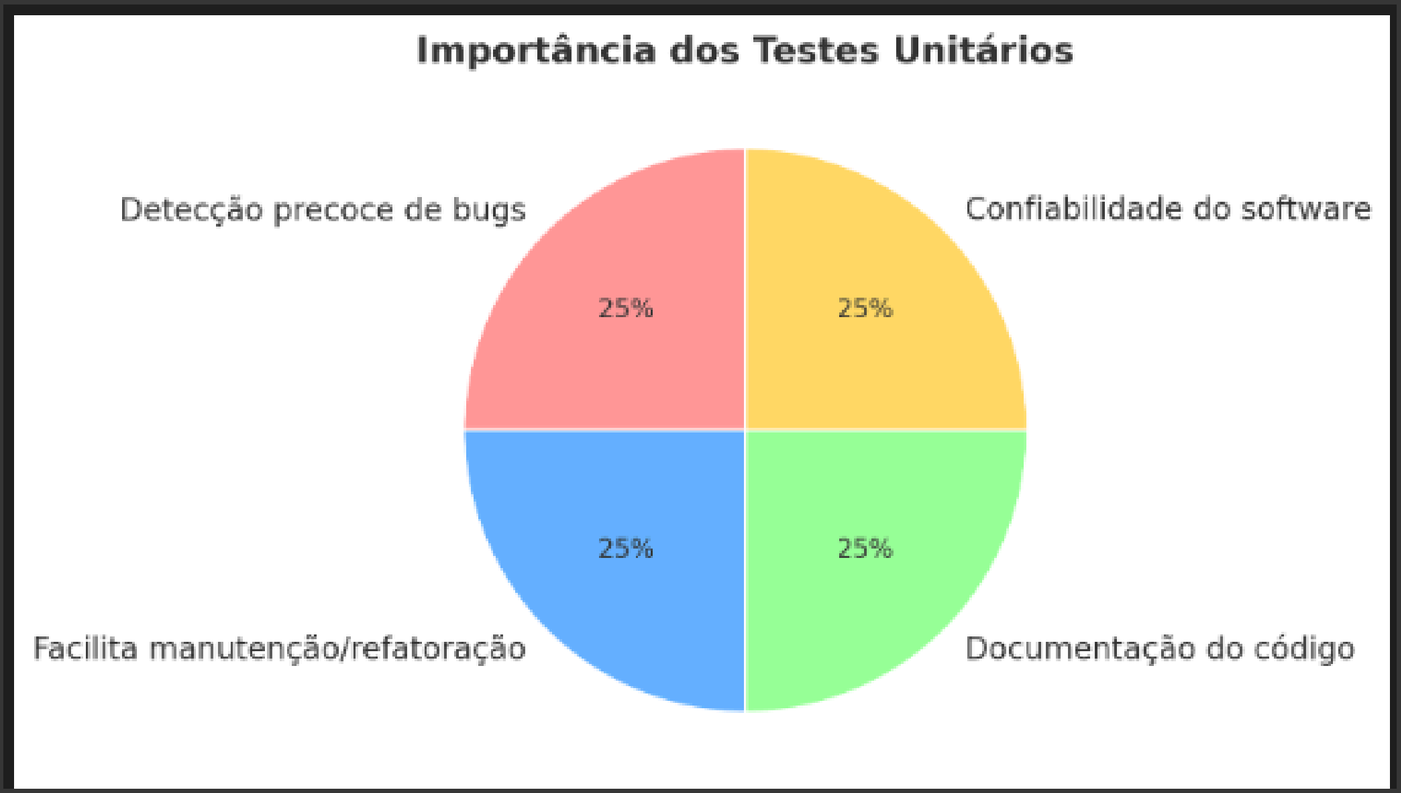
INTRODUÇÃO AOS TESTES UNITÁRIOS:

O que são?

- Testes que verificam pequenas partes do código (funções, métodos ou classes) de forma isolada.

Por que usar?

- Encontrar erros cedo
- Ajudar na manutenção e refatoração
- Servir como documentação
- Aumentar a confiança no software



BIBLIOTECA UNITTEST:

O que é?

Framework de testes do Python, inspirado no JUnit (Java).
Já vem na biblioteca padrão (não precisa instalar).

Pontos Positivos:

- Fácil de usar para quem conhece JUnit.
- Permite organizar testes com classes e métodos.
- Suporte a fixtures (setUp/tearDown).

Pontos Negativos:

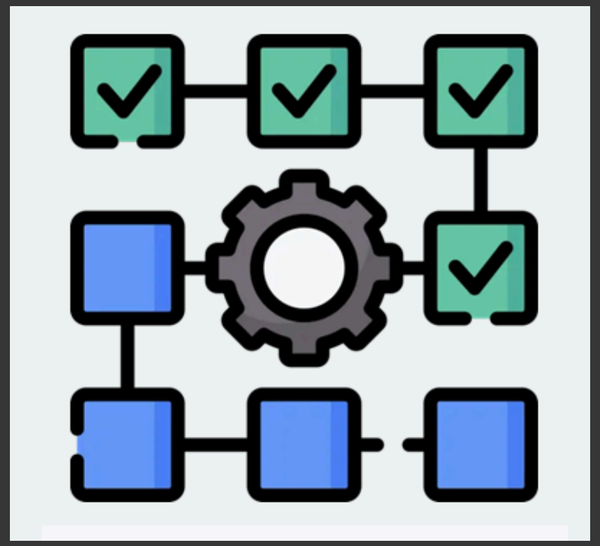
- Sintaxe mais verbosa que o Pytest.
- Precisa herdar unittest.TestCase.
- Relatórios de teste menos amigáveis.

BIBLIOTECA PYTEST:

O que é?

Framework de testes para Python

Sintaxe simples + recursos avançados



Pontos Positivos

- Sintaxe curta e fácil de aprender
- Não precisa herdar classes de teste
- Descobre testes automaticamente
- Fixtures poderosas e flexíveis
- Plugins para estender funcionalidades
- Relatórios mais claros

Pontos Negativos

- Requer instalação (pip install pytest)
- Pode ser mais complexo em projetos grandes

COMPARAÇÃO: UNITTEST VS PYTEST

unittest

- Sintaxe mais verbosa, baseada em classes
- Já vem no Python (não precisa instalar)
- Fixtures com setUp e tearDown por classe

pytest

- Sintaxe curta e simples, baseada em funções
- Precisa instalar com pip install pytest
- Fixtures mais flexíveis e poderosas

TESTES DE CAIXA BRANCA:

O que são?

- Também chamados de testes estruturais.
- Avaliam a estrutura interna, design e implementação do sistema.
- Exigem conhecimento do código, algoritmos e fluxo de controle.

Como aplicar?

- Cobertura de código → garantir que todas as linhas sejam testadas.
- Teste de caminho → validar todos os caminhos de execução.
- Teste de loop → verificar execução correta dos laços.
- Teste de condição → checar condições lógicas.

Ferramentas

- Estática: SonarQube, Checkmarx
- Dinâmica: OWASP ZAP, Burp Suite
- Cobertura: Cobertura.py, JaCoCo
- Depuração: PyCharm, Visual Studio Debugger

TESTES DE CAIXA PRETA:

Testes de Caixa Preta avaliam o software pelo comportamento e saídas, sem conhecer o código interno.

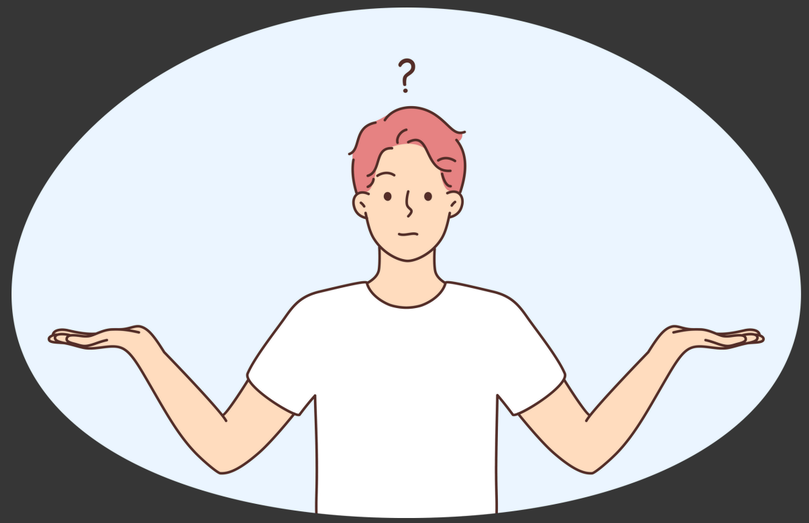
Aplicações: verificar requisitos, evitar falhas em mudanças (regressão), avaliar usabilidade e desempenho.

Ferramentas: Selenium, Cypress, Postman, JMeter, entre outras.

COMPARAÇÃO: CAIXA BRANCA VS CAIXA PRETA:

Caixa Branca: precisa conhecer o código, foca na lógica interna, feito por desenvolvedores/testadores técnicos, aplicado em testes unitários e de integração.

Caixa Preta: não precisa conhecer o código, foca na funcionalidade e comportamento externo, feito por testadores/usuários/QA, aplicado em testes de sistema e aceitação.



FERRAMENTAS PARA TESTES:

1. Testes de Caixa Branca (internos, analisam o código):

- Análise estática de código: SonarQube, Checkmarx
- Cobertura de código: JaCoCo (Java), Coverage.py (Python)
- Debuggers: PyCharm Debugger, Visual Studio Debugger

2. Testes de Caixa Preta (externos, analisam funcionalidade):

- Automação de UI: Selenium, Cypress
- Testes de API: Postman (REST), SoapUI (SOAP)
- Testes de carga/desempenho: JMeter, LoadRunner

CONCLUSÃO:

Testes unitários garantem a qualidade de cada parte do código.

Unittest e Pytest são bibliotecas para implementar testes unitários em Python.

Testes de caixa branca analisam a estrutura interna do código.

Testes de caixa preta avaliam o comportamento externo do software.

O ideal é combinar ambos para maior cobertura e qualidade.

Escolher as ferramentas corretas otimiza o desenvolvimento e melhora o produto final.

Obrigado pela atenção!