| Curso | Engenharia da Computação | Período | 10A |
|---|---|---|---|
| Disciplina | Engenharia de Software | Data | 09/10/2024 |
| Professor | Prof. Marcílio | | |

**Exercícios sobre Testes – Engenharia de Software**

**Parte 1)** Baseado no que foi visto sobre TDD (*Test-Driven Domain*), implemente o que se pede nos exercícios a seguir. Você deverá tirar prints que comprovem a **falha** dos testes e os **sucessos** dos testes.

Dica: Se necessário, consulte os seguintes links para maiores informações de como utilizar o módulo *unittest* do Python:

**Link 1:** Unit testing framework (https://docs.python.org/3/library/unittest.html)

**Link 2:** Testes Unitários com Python usando unittest (https://blog.formacao.dev/testes-unitarios-em-python-usando-unittest/)

**Link 3:** Como utilizar unittest (https://medium.com/itautech/testes-unit%C3%A1rios-em-python-como-utilizar-o-unittest-e-execut%C3%A1-los-na-aws-70d13193e42b)

**Exercício 1)** Dado o contexto de uma aplicação que gerencia uma **biblioteca de livros**, onde os usuários podem:

- **Adicionar um livro** ao sistema.
- **Buscar um livro** pelo título.
- **Remover um livro** do sistema.

Você deverá criar uma classe chamada Biblioteca que terá as seguintes funcionalidades:

- **Adicionar livro**: O método adicionar_livro adiciona um livro ao sistema.
- **Buscar livro por título**: O método buscar_livro permite buscar um livro pelo seu título.
- **Remover livro**: O método remover_livro permite remover um livro da biblioteca.

Casos de Teste a serem implementadas:

- **Teste de Adição de Livro**: Verificar se um livro pode ser adicionado com sucesso ao sistema.
- **Teste de Busca de Livro**: Verificar se é possível buscar um livro pelo título.
- **Teste de Remoção de Livro**: Verificar se um livro pode ser removido da biblioteca.

**Exercício 2)** Baseado no contexto de uma aplicação de conta bancária, onde os usuários podem:

- **Depositar dinheiro** na conta.
- **Sacar dinheiro** da conta.
- **Verificar o saldo** da conta.

Implemente uma classe para fazer os testes unitários necessários, conforme abaixo:

- **Teste de Depósito**: Verificar se o depósito aumenta o saldo corretamente.
- **Teste de Saque**: Verificar se o saque reduz o saldo corretamente e se impede saques maiores que o saldo disponível.
- **Teste de Saldo**: Verificar se o saldo está correto após uma série de depósitos e saques.

Após a implementação dos testes unitários, implemente uma classe ***ContaBancaria*** com as funcionalidades necessárias para que todos os testes sejam validados e passem.

**Exercício 3)** Baseado nas ODS apresentadas abaixo, escolha um tema e proponha, ao menos 3 casos de testes para uma eventual aplicação. Você deverá pensar no contexto para isso. Anexe os códigos na tarefa do Google Classroom.


**Parte 2)** Baseado na leitura do pdf disponibilizado na atividade do Google Classroom, responda – com suas palavras – o que se pede abaixo. Você deverá anexar um **GOOGLE DOCS** na atividade, o qual deve conter o seu nome, seu RA e as respostas dos exercícios abaixo.

**Exercício 3)** Explique o que é Verificação e Validação (V&V).

**Exercício 4)** Explique a diferença entre testes de componentes e testes de integração.

**Exercício 5)** Explique o que é como funciona o teste caixa-branca e o teste caixa-preta para o cenário de testes de componentes e de nível de integração.