Disciplina: Arquitetura de Software

Professor: Sidney Loyola

Aluna: Tássia Nascimento – 202212166 – Turma B

Testes automatizados com Python

Introdução

Este trabalho tem como objetivo aplicar conceitos de desenvolvimento de software relacionados à correção de bugs e criação de testes automatizados utilizando Python. Foram fornecidos três arquivos com classes que continham métodos apresentando falhas na implementação. As atividades realizadas envolveram:

Identificar e corrigir os bugs nos métodos das classes.
Escrever testes automatizados para validar os métodos corrigidos.
Utilizar a biblioteca unittest ou pytest para criar e executar os testes.

Arquivo: **FibonacciGenerator.py**
Contém a classe FibonacciGenerator, responsável por gerar a sequência de Fibonacci e obter o enésimo número da sequência. Apresentava 2 problemas:

generate_sequence: Não calculava corretamente a sequência.
get_nth_number: O loop não continha a quantidade necessária de iterações para retornar o enésimo número.
Correções feitas:
Ajustado o loop em generate_sequence para somar os dois últimos números corretamente.
Corrigido o loop em get_nth_number para percorrer o número correto de iterações.

Testes Implementados
Objetivo: validar a sequência gerada para diferentes valores de n.
Validar o enésimo número para diferentes entradas.
Os testes garantem que os métodos funcionem corretamente para diferentes casos, incluindo entradas inválidas.

Arquivo: **StringUtils.py**
O arquivo contém a classe StringUtils, responsável por manipulação de strings. Um dos métodos apresentava problemas:

reverse_string: Não realizava a inversão da string.
Correções feitas:
Implementado o uso de slicing ([::-1]) para reverter a string.

Testes Implementados
Validação da inversão de diferentes strings.
Verificação de strings: palíndromos ou não.
Os testes verificam se a inversão é realizada corretamente e se o método de detecção de palíndromos está funcional.

Arquivo: **UserManager.py**
O arquivo contém a classe UserManager, responsável por gerenciar uma lista de usuários. Um dos métodos apresentava problemas:

remove_user: Não tratava o caso em que o usuário não existia na lista.

Correções Feitas:
Foi adicionado tratamento para evitar erro ao tentar remover um usuário inexistente.

Testes Implementados
Validação e adição de usuários únicos.
Garantia de que a remoção de usuários funcione, incluindo o tratamento de erros para usuários inexistentes.
Os testes garantem que o método **remove_user** lida corretamente com os casos válidos e inválidos.

Conclusão
O trabalho realizado demonstrou a importância de testes automatizados para garantir a qualidade do código. Após identificar e corrigir os bugs nos métodos, os testes implementados validaram a funcionalidade dos mesmos, abrangendo diferentes cenários, inclusive entradas inválidas.