02 – Configurando ferramenta de cobertura

O chefe da Dominique entrou em contato para elogiá-la e falar que ela está avançando muito bem nos seus aprendizados sobre desenvolvimento de projetos. Ela entendeu o que são testes automatizados, testes unitários e como utilizar o TDD para criar funcionalidades seguindo as regras de negócio.

Durante essa conversa, o chefe dela comentou sobre **cobertura de testes** e a Dominique pediu nossa ajuda para entender esse conceito. Ao comparar lado a lado os arquivos bytebank.py e test\_bytebank.py, o chefe da Dominique concluiu que existem vários testes para vários segmentos do projeto, contudo não há garantia de que todas as linhas de código estão cobertas por algum teste. Quando trabalhamos com testes, a intenção é ter 100% de cobertura por testes.

A Dominique ficou um pouco confusa: "Como assim não tem garantia?". Comparando os dois arquivos e examinando os nomes verbosos dos testes, ela sabe que o primeiro teste verifica o funcionamento do método idade, o segundo teste refere-se ao método sobrenome e assim por diante. Segundo a análise dela, cobrimos todas as partes do código.

Mas existe um problema em fazer essa análise manualmente. Conforme um projeto se torna mais extenso e complexo, fica cada vez mais difícil nos certificarmos de que cobrimos todos os cenários possíveis de todos os segmentos de código. Por isso, existem ferramentas para nos ajudar a verificar a cobertura de testes! O Pytest tem uma extensão chamada **pytest-cov**. "Cov" é uma abreviação de *coverage* — "cobertura", em inglês.

Vamos instalar a versão 3.0.0 dessa extensão pelo terminal com o pip, usando o comando pip install pytest-cov==3.0.0. Em seguida, executaremos pip freeze > requeriments.txt para atualizar o nosso arquivo requirements.txt que contém a lista com todos os pacotes instalados no ambiente virtual. Depois, podemos abrir o arquivo requirements.txt e checar que o pytest-cov==3.0.0 está listado também.

Para rodar a extensão e verificarmos qual é a cobertura do código da Dominique, vamos executar o comando pytest --cov no terminal. Assim, o pytest-cov procurará todos os testes para checar a cobertura. Na sequência, vamos examinar o resultado.

Primeiramente, o Pytest foi executado normalmente e obtemos tests/test\_bytebank.py ..... — os cinco pontos indicam que os cinco testes passaram. Em seguida, temos uma tabela que explica cada parte em que o *coverage* rodou:

| **Name** | **Stmts** | **Miss** | **Cover** |
| --- | --- | --- | --- |
| codigo/bytebank.py | 35 | 1 | 97% |
| tests/\_*init\_*.py | 0 | 0 | 100% |
| tests/test\_bytebank.py | 38 | 1 | 97% |
| **TOTAL** | **73** | **2** | **97%** |

Na coluna "*Name*", temos o nome dos arquivos em que foi feita a leitura da cobertura. O primeiro deles é o bytebank.py, onde realmente queremos que ele rode. Depois, temos o *dunder init* e o próprio arquivo de testes (test-bytebank.py).

Na segunda coluna, temos os *statements* (a quantidade de linhas do código). No bytebank.py, por exemplo, temos 35 *statements*. Depois, temos a coluna "*Miss*", que indica a quantidade de *statements* que não estão sendo testados! Ou seja, no arquivo bytebank.py, perdemos uma linha na cobertura de testes.

Por fim, temos a coluna "*Cover*" que exibe a porcentagem de cobertura de testes de cada arquivo. No bytebank.py, temos 97% de cobertura.

No momento, temos interesse em examinar apenas a cobertura de testes do arquivo bytebank.py. Então, vamos executar novamente o pytest-cov, dessa vez definindo o diretório onde queremos realizar o teste (no caso, codigo). Depois, precisamos indicar o nome da pasta de testes, que é tests:

pytest --cov=codigo tests/COPIAR CÓDIGO

Executando o comando, o pytest-cov selecionará apenas o arquivo que especificamos e fará os testes só no bytebank.py:

| **Name** | **Stmts** | **Miss** | **Cover** |
| --- | --- | --- | --- |
| codigo/bytebank.py | 35 | 1 | 97% |
| **TOTAL** | **35** | **1** | **97%** |

Sendo assim, conseguimos rodar pela primeira vez a ferramenta de cobertura de testes. Temos 97% de cobertura, o que significa que tem alguma linha do bytebank.py que não está sendo coberta, não há um teste para ela.

No próximo vídeo, vamos descobrir que linha é essa e o que podemos mudar para alcançar os 100% de cobertura!