03 – Implementando uma funcionalidade

O chefe da Dominique pediu para ela adicionar mais uma funcionalidade à classe Funcionario, então chegou a hora de aplicarmos nosso conhecimento de TDD.

A Bytebank está passando por um período financeiramente instável, por isso os sócios e os diretores da empresa concordaram em fazer um decréscimo de 10% em seus próprios salários para ajudar nessa situação, visto que todos eles ganham um salário mínimo de R$100.000,00. Então, a Dominique será responsável por implementar o método que identificará os diretores e fará esse decréscimo.

Seguindo a metodologia do TDD, a primeira etapa são os testes. Vamos abrir o arquivo test\_bytebank.py e começar a codar o teste para essa nova funcionalidade.

Primeiramente, escolheremos o nome do método, lembrando que ele deve começar com test\_ e ser o mais verboso possível. Mantendo o padrão usado anteriormente, o nomearemos test\_quando\_decrescimo\_salario\_recebe\_100000\_deve\_retornar\_90000 — pois, ao tirar 10% de um salário de R$100.000,00, o resultado é R$90.000,00.

Em seguida, escreveremos o código aplicando a metodologia Given-When-Then. A entrada do salário será 100000 e o resultado esperado é 90000:

def test\_quando\_decrescimo\_salario\_recebe\_100000\_deve\_retornar\_90000(self):

entrada\_salario = 100000 #given

esperado = 90000COPIAR CÓDIGO

Ou seja, o *Given* é a entrada do salário. Depois, vamos instanciar o funcionário:

def test\_quando\_decrescimo\_salario\_recebe\_100000\_deve\_retornar\_90000(self):

entrada\_salario = 100000 #given

esperado = 90000

funcionario\_teste = Funcionario('Teste', '11/11/2000', entrada\_salario)COPIAR CÓDIGO

No vídeo, há um pequeno erro de digitação na variável funcionario\_teste, mas ele já foi corrigido nas transcrições.

Na sequência, chamaremos o método responsável pelo decréscimo e o resultado deve ser igual ao salário do funcionário:

def test\_quando\_decrescimo\_salario\_recebe\_100000\_deve\_retornar\_90000(self):

entrada\_salario = 100000 #given

esperado = 90000

funcionario\_teste = Funcionario('Teste', '11/11/2000', entrada\_salario)

funcionario\_teste.decrescimo\_salario() # when

resultado = funcionario\_teste.salarioCOPIAR CÓDIGO

Em outras palavras, o *When* é a chamada do método decrescimo\_salario. Por fim, com o assert, verificaremos se o resultado é igual ao esperado — este será o *Then*:

def test\_quando\_decrescimo\_salario\_recebe\_100000\_deve\_retornar\_90000(self):

entrada\_salario = 100000 #given

esperado = 90000

funcionario\_teste = Funcionario('Teste', '11/11/2000', entrada\_salario)

funcionario\_teste.decrescimo\_salario() # when

resultado = funcionario\_teste.salario

assert resultado == esperado # thenCOPIAR CÓDIGO

É possível que haja outros funcionários na Bytebank que ganhem R$100.000,00 e não sejam diretores. Então, para garantir que apenas o salário dos diretores será afetado, vamos incluir mais uma entrada, com o nome de um diretor:

def test\_quando\_decrescimo\_salario\_recebe\_100000\_deve\_retornar\_90000(self):

entrada\_salario = 100000 #given

entrada\_nome = 'Paulo Bragança'

esperado = 90000

funcionario\_teste = Funcionario(entrada\_nome, '11/11/2000', entrada\_salario)

funcionario\_teste.decrescimo\_salario() # when

resultado = funcionario\_teste.salario

assert resultado == esperado # thenCOPIAR CÓDIGO

Note que também modificamos a instanciação do funcionario\_teste, passando entrada\_nome para o parâmetro nome.

Com o teste pronto, vamos rodá-lo usando o comando pytest -v no terminal. Ele falhará, porque ainda não chegamos na parte de codar. Seguindo as etapas do ciclo do TDD, primeiro desenvolvemos o teste e eles falharão, então faremos a implementação.

A seguir, ajudaremos a Dominique a implementar esse método. Já conhecemos as regras de negócio, respeitadas pelo teste: se temos um salário de R$100.000,00 e o nome de um dos diretores, o método deve ser chamado e haverá um decréscimo de 10% no salário desse funcionário. Agora, temos que codar um método que faça o teste passar.

No arquivo bytebank.py, após o método sobrenome, criaremos o método decrescimo\_salario:

def decrescimo\_salario(self):

if self.\_salario >= 100000:

decrescimo = self.\_salario \* 0.1

self.\_salario = self.\_salario - decrescimoCOPIAR CÓDIGO

De início, incluímos uma condicional: se o salário do funcionário for maior ou igual a R$100.000,00, o decréscimo deve ocorrer. Em seguida, declaramos a variável decrescimo, que será igual a 10% desse salário. Por fim, subtraímos o valor do decréscimo.

Além de verificar se a pessoa ganha R$100.000,00 ou mais, é preciso checar se ela também está na lista de diretores e sócios. A Dominique possui uma lista dos sobrenomes deles, então vamos inseri-la no código:

def decrescimo\_salario(self):

sobrenomes = ['Bragança', 'Windsor', 'Bourbon', 'Yamato', 'Al Saud', 'Khan', 'Tudor', 'Ptolomeu']

if self.\_salario >= 100000:

decrescimo = self.\_salario \* 0.1

self.\_salario = self.\_salario - decrescimoCOPIAR CÓDIGO

Na sequência, adicionaremos mais uma condição. Para que o decréscimo aconteça, além de ganhar mais de R$100.000,00, o sobrenome do funcionário deve estar na lista sobrenomes:

def decrescimo\_salario(self):

sobrenomes = ['Bragança', 'Windsor', 'Bourbon', 'Yamato', 'Al Saud', 'Khan', 'Tudor', 'Ptolomeu']

if self.\_salario >= 100000 and (self.sobrenome() in sobrenomes):

decrescimo = self.\_salario \* 0.1

self.\_salario = self.\_salario - decrescimoCOPIAR CÓDIGO

Perceba que, para selecionar o sobrenome do funcionário, usamos o método sobrenome que desenvolvemos anteriormente. Dessa forma, geramos o método que fará nosso teste passar. Vamos chamar o pytest -v no terminal novamente.

Todos os testes passarão! Quer dizer que completamos a segunda etapa do ciclo do TDD. Elaboramos os testes e eles falharam, depois desenvolvemos o código e os testes passaram. Podemos partir para a refatoração.