



FGA0314 – TESTES DE SOFTWARE (2025.2)

Atividade 4 – TDD

Validador de Expressões Matemáticas

Turma	01	Semestre	2025.2
Equipe	Ferrugem		
Nome		Matrícula	
Davi Marques do Egito Coelho		231030421	
Luiz Gustavo Silva de Almeida		221022669	
Vitor Guilherme Lustosa de Carvalho		232014342	
Artur Handow Krauspenhar		231034082	
Lucas de Oliveira Rodrigues		202017684	
Mauricio Ferreira de Araujo		222007021	

1. Linguagem e framework de teste unitário utilizados.

A linguagem de programação escolhida para a implementação do TDD validador de expressões matemáticas foi o Python. O framework de teste unitário utilizado foi o (**pytest ou unittest**).

2. Ciclos de TDD (Red-Green-Refactor).

Ciclo 1: O caso mais básico (Inválido).

- Teste: `deveRejeitarExpressaoVazia()`



- **RED:** foco em lidar com a string vazia. Resultado: teste falhado.

```
test_ehExpressaoValida.py atividade4_testes U X ehExpressaoValida.py atividade4_testes ESP-IDF: Search
test_ehExpressaoValida.py > test_deveRejeitarExpressaoVazia
1 from ehExpressaoValida import ehExpressaoValida
2
3 def test_deveRejeitarExpressaoVazia():
4     assert ehExpressaoValida("") == False

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ESP-IDF

test_ehExpressaoValida.py::test_deveRejeitarExpressaoVazia FAILED [100%]

===== FAILURES =====
test_deveRejeitarExpressaoVazia
def test_deveRejeitarExpressaoVazia():
> assert ehExpressaoValida("") == False
E AssertionError: assert None == False
E + where None = ehExpressaoValida('')

test_ehExpressaoValida.py:4: AssertionError
===== short test summary info =====
FAILED test_ehExpressaoValida.py::test_deveRejeitarExpressaoVazia - AssertionError: assert None == False
===== 1 failed in 0.16s =====
```

- **GREEN:** função mínima para passar. Resultado: teste bem-sucedido.

```
ehExpressaoValida.py > ehExpressaoValida
1 def ehExpressaoValida(expressao):
2     return False

PS C:\Projetos\atividade4_testes> python -m pytest test_ehExpressaoValida.py -v
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\artur\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Projetos\atividade4_testes
collected 1 item

test_ehExpressaoValida.py::test_deveRejeitarExpressaoVazia PASSED [100%]

===== 1 passed in 0.04s =====
```

Ciclo 2: Regra 6 (Ignorar Espaços).

- **Teste:** `deveRejeitarExpressaoApenasComEspacos()`
- **RED:** sem implementação, teste falhando.



```
File: test_ignoraespaco.py

1 def test_ignoraespacoespaco():
2     pass
```

```
30 de out 15:46
maubas@fedora: ~/Documentos/atividade4_testes
~/Documentos/atividade4_testes

+ atividade4_testes git:(main) X python -m pytest
===== test session starts =====
platform linux -- Python 3.13.7, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/maubas/Documentos/atividade4_testes
plugins: anyio-4.8.0
collected 2 items

tests/test_ehExpressaoValida.py . [ 50%]
tests/test_ignoraespaco.py F [100%]

===== FAILURES =====
test_ignoraespacoespaco
def test_ignoraespacoespaco():
    assert ignoraespaco(" ") == False
E     NameError: name 'ignoraespaco' is not defined

tests/test_ignoraespaco.py:2: NameError
===== short test summary info =====
FAILED tests/test_ignoraespaco.py::test_ignoraespacoespaco - NameError: name 'ignoraespaco' is not defined
===== 1 failed, 1 passed in 0.02s =====
+ atividade4_testes git:(main) X
```

- **GREEN:** função mínima para passar, teste bem-sucedido.

```
File: ehExpressaoValida.py

1 ~ def ehExpressaoValida(expressao):
2 ~     # remoção de espaços da expressão
3 ~     expr_sem_espaco = expressao.replace(" ", "")
4 ~
5 ~     # se após remover os espaços a string estiver vazia, tem que retornar fa
lso
6 ~     if expr_sem_espaco == "":
7 ~         return False
8 ~
9 ~     # retorne falso pra qualquer outra expressão
10 ~     return False
```

```
+ atividade4_testes git:(main) X python -m pytest tests/test_ehExpressaoValida.py::test_deveRejeitarExpressaoApenasComEspacos -v
===== test session starts =====
platform linux -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0 -- /usr/bin/python
cachedir: .pytest_cache
rootdir: /home/archsphinx/git/atividade4_testes
collected 1 item

tests/test_ehExpressaoValida.py::test_deveRejeitarExpressaoApenasComEspacos PASSED [100%]

===== 1 passed in 0.01s =====
```



Ciclo 3: O caso válido mais simples.

- **Teste:** `deveAceitarNumeroUnico()`
- **RED:** sem implementação, teste falhando.

```
from src.ehExpressaoValida import ehExpressaoValida

def test_deveAceitarNumeroUnico():
    assert ehExpressaoValida(5) == True
```

```
30 de out 15:50
maubas@fedora: ~/Documentos/atividade4_testes
maubas@fedora:~/Documentos/atividade4_testes x vim tests/test_ignoraespaco.py — vim tests/test_ignoraespaco.py

+ atividade4_testes git:(main) X python -m pytest
===== test session starts =====
platform linux -- Python 3.13.7, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/maubas/Documentos/atividade4_testes
plugins: anyio-4.8.0
collected 3 items

tests/test_aceitaNumeroUnico.py F [ 33%]
tests/test_ehExpressaoValida.py . [ 66%]
tests/test_ignoraespaco.py F [100%]

===== FAILURES =====
test_deveAceitarNumeroUnico
    def test_deveAceitarNumeroUnico():
    > assert ehNumeroUnico(5) == True
    E NameError: name 'ehNumeroUnico' is not defined

tests/test_aceitaNumeroUnico.py:4: NameError
test_ignoraespacosespaco
    def test_ignoraespacosespaco():
    > assert ignoraespaco(" ") == False
    E NameError: name 'ignoraespaco' is not defined

tests/test_ignoraespaco.py:2: NameError
===== short test summary info =====
FAILED tests/test_aceitaNumeroUnico.py::test_deveAceitarNumeroUnico - NameError: name 'ehNumeroUnico' is not defined
FAILED tests/test_ignoraespaco.py::test_ignoraespacosespaco - NameError: name 'ignoraespaco' is not defined
===== 2 failed, 1 passed in 0.02s =====
+ atividade4_testes git:(main) X
```

- **GREEN:** teste passando após a função implementada.

```
+ atividade4_testes git:(main) X python -m pytest
===== test session starts =====
platform linux -- Python 3.13.7, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/maubas/Documentos/atividade4_testes
plugins: anyio-4.8.0
collected 4 items

tests/test_aceitaNumeroUnico.py . [ 25%]
tests/test_ehExpressaoValida.py .. [ 75%]
tests/test_validador_expressao.py . [100%]

===== 4 passed in 0.03s =====
```

- **BLUE:** refatoração.

```
def ehExpressaoValida(expressao):
    if isinstance(expressao, int):
        return True

    # remoção de espaços da expressão
    expr_sem_espaco = expressao.replace(" ", "")

    # se após remover os espaços a string estiver vazia, tem que retornar falso
    if expr_sem_espaco == "":
        return False

    # retorne falso pra qualquer outra expressão
    return False
```

Ciclo 4: Regra 3 (Começar/Terminar com Operador).

- **Teste:** `deveRejeitarSeComecarOuTerminarComOperador()`
- **RED:** código do teste implementado, com o teste falhado.

test_validador_expressao.py ×

validador_expressao.py

tests > test_validador_expressao.py

```
1 import sys, os
2 sys.path.append(os.path.join(os.path.dirname(__file__), "..", "src"))
3
4 from src.validador_expressao import eh_expressao_valida
5
6 def test_deveRejeitarSeComecarOuTerminarComOperador():
7     casos_invalidos = ["+1", "1*", "/"]
8     for expr in casos_invalidos:
9         assert eh_expressao_valida(expr) is False
10
```

```
C:\Users\lalde\OneDrive\Desktop\AAE-4\atividade4_testes>python -m pytest tests/test_validador_expressao.py
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\lalde\OneDrive\Desktop\AAE-4\atividade4_testes
plugins: anyio-4.11.0, socket-0.7.0, xdist-3.8.0
collected 1 item

tests\test_validador_expressao.py F [100%]

===== FAILURES =====
_____ test_deveRejeitarSeComecarOuTerminarComOperador _____

    def test_deveRejeitarSeComecarOuTerminarComOperador():
        casos_invalidos = ["+1", "1*", "/"]
        for expr in casos_invalidos:
            assert eh_expressao_valida(expr) is False
>           AssertionError: assert True is False
E             + where True = eh_expressao_valida('+1')

tests\test_validador_expressao.py:9: AssertionError
===== short test summary info =====
FAILED tests\test_validador_expressao.py::test_deveRejeitarSeComecarOuTerminarComOperador - AssertionError: assert True is False
===== 1 failed in 0.19s =====
```



- **GREEN:** código da funcionalidade implementada no ciclo, com o teste bem-sucedido.

```
test_validador_expressao.py 1, U  validador_expressao.py U X

src > validador_expressao.py > eh_expressao_valida

1  def eh_expressao_valida(expressao: str) -> bool:
2
3      expr = expressao.replace(" ", "")
4      if not expr:
5          return False
6
7      operadores = {"+", "-", "*", "/"}
8      if expr[0] in operadores or expr[-1] in operadores:
9          return False
10
11     return True
```

```
C:\Users\lalm\OneDrive\Desktop\AAE-4\atividade4_testes>python -m pytest tests/test_validador_expressao.py
===== test session starts =====
platform win32 -- Python 3.12.0, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\lalm\OneDrive\Desktop\AAE-4\atividade4_testes
plugins: anyio-4.11.0, socket-0.7.0, xdist-3.8.0
collected 1 item

tests\test_validador_expressao.py . [100%]

===== 1 passed in 0.04s =====
```

Ciclo 5: Regra 2 (Operadores Seguidos).

- **Teste:** `deveRejeitarOperadoresDuplos()`
- **RED:** código do teste implementado, com o teste falhando.

```
PS C:\Users\lukin\Downloads\att4_testes> C:\Users\lukin\Downloads\att4_testes\.venv\Scripts\python.exe -m pytest tests/test_validador_expressao.py -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\lukin\Downloads\att4_testes\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\lukin\Downloads\att4_testes
collected 1 item

tests\test_validador_expressao.py::test_deveRejeitarOperadoresDuplos FAILED [100%]

===== FAILURES =====
test_deveRejeitarOperadoresDuplos

def test_deveRejeitarOperadoresDuplos():
    # Test case 1: Adjacent plus operators
    > assert not validador_expressao("1+2"), "Should reject expression with consecutive plus operators"
E   AssertionError: Should reject expression with consecutive plus operators
E   assert not True
E   + where True = validador_expressao("1+2")

tests\test_validador_expressao.py:6: AssertionError

===== short test summary info =====
FAILED tests\test_validador_expressao.py::test_deveRejeitarOperadoresDuplos - AssertionError: Should reject expression with consecutive plus operators
===== 1 failed in 0.25s =====
PS C:\Users\lukin\Downloads\att4_testes>
```



```
===== FAILURES =====
test_deveRejeitarOperadoresDuplos

def test_deveRejeitarOperadoresDuplos():
    # Test case 1: Adjacent plus operators
    assert not validator_expressao("1+2"), "Should reject expression with consecutive plus operators"

    # Test case 2: Adjacent division and multiplication operators
    > assert not validator_expressao("5 / * 3"), "Should reject expression with consecutive division and multiplication operators"
    E AssertionError: Should reject expression with consecutive division and multiplication operators
    E     assert not True
    E     + where True = validator_expressao('5 / * 3')

tests/test_validador_expressao.py:9: AssertionError
===== short test summary info =====
FAILED tests/test_validador_expressao.py::test_deveRejeitarOperadoresDuplos - AssertionError: Should reject expression with consecutive division and multiplication operators
```

Ciclo 6: Regra 1 (Parênteses Balanceados).

- **Teste:** `deveRejeitarParentesesDesbalanceados()`
- **RED:** código do teste implementado, com o teste falhado.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python -m pytest tests/test_ehExpressaoValida.py::test_deveRejeitarParentesesDesbalanceados

PS C:\Users\vgui4\atividade4_testes>
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\vgui4\atividade4_testes
collected 0 items

===== no tests ran in 0.03s =====
ERROR: not found: C:\Users\vgui4\atividade4_testes\tests\test_ehExpressaoValida.py::test_deveRejeitarParentesesDesbalanceados
(no match in any of [*Module test_ehExpressaoValida.py*])

PS C:\Users\vgui4\atividade4_testes>
```

```
test_ehExpressaoValida.py • ehExpressaoValida.py • validator_expressao.py • test_aceitaNumeroUnico.cpython-311
tests > test_ehExpressaoValida.py > test_deveRejeitarParentesesDesbalanceados
1 from src.ehExpressaoValida import ehExpressaoValida
2
3
4 def test_deveRejeitarExpressaoVazia():
5     assert ehExpressaoValida("") == False
6
7
8 def test_deveRejeitarExpressaoApenasComEspacos():
9     assert ehExpressaoValida(" ") == False
10
11 def test_deveRejeitarParentesesDesbalanceados():
12     casos_invalidos = ["(1+2", "1+2)", "("]
13     for expr in casos_invalidos:
14         assert ehExpressaoValida(expr) == False
```

- **GREEN:** código da funcionalidade implementada no ciclo, com o teste bem-sucedido.

```
PS C:\Users\vgui4\atividade4_testes> python -m pytest tests/
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.4.2, pluggy-1.6.0
rootdir: C:\Users\vgui4\atividade4_testes
collected 4 items

tests/test_aceitaNumeroUnico.py . [ 25%]
tests/test_ehExpressaoValida.py .. [ 75%]
tests/test_validador_expressao.py . [100%]

===== 4 passed in 0.09s =====

PS C:\Users\vgui4\atividade4_testes>
```

```
1 def eh_expressao_valida(expressao: str) -> bool:
2     expr = expressao.replace(" ", "")
3     balanceamento = 0
4     for char in expr:
5         if char == '(':
6             balanceamento += 1
7         elif char == ')':
8             balanceamento -= 1
9
10        if balanceamento < 0:
11            return False
12    if balanceamento != 0:
13        return False
```

- **BLUE/Refactor:** código de produção e arquivo de testes refatorados.

```
validador_expressao.py X
src > validador_expressao.py > validador_expressao
1 def validador_expressao(expressao):
2
3     # ciclo 1 - tipo inválido
4     if expressao is None:
5         return False
6
7     # converte tudo pra string (caso número passe direto)
8     if isinstance(expressao, int):
9         return True
10
11    # ciclo 2 - regra 6
12    expr = expressao.replace(" ", "")
13    if not expr:
14        return False
15
16    # ciclo 3 - número único válido
17    if expr.isdigit():
18        return True
19
20    # ciclo 4 - regra 3
21    operadores = {"+", "-", "*", "/"}
22    if expr[0] in operadores or expr[-1] in operadores:
23        return False
24
25    # ciclo 6 - regra 1
26    balanceamento = 0
27    for char in expr:
28        if char == '(':
29            balanceamento += 1
30        elif char == ')':
31            balanceamento -= 1
32
33        if balanceamento < 0:
34            return False
35    if balanceamento != 0:
36        return False
37
38    # demais casos ainda não implementados (ciclos 5+)
39    return False
```

- https://github.com/daviegito/atividade4_testes/blob/main/src/validador_expressao.py

test_validador_expressao.py X

tests > test_validador_expressao.py > ...

```
1  from src.validador_expressao import validador_expressao
2
3  # ciclo 1
4  def test_deveRejeitarExpressaoVazia():
5      |   assert validador_expressao("") is False
6
7  # ciclo 2
8  def test_deveRejeitarExpressaoApenasComEspacos():
9      |   assert validador_expressao("") is False
10
11 # ciclo 3
12 def test_deveAceitarNumeroUnico():
13     |   assert validador_expressao(5) is True
14
15 # ciclo 4
16 def test_deveRejeitarSeComecarOuTerminarComOperador():
17     |   casos_invalidos = ["+1", "1*", "/"]
18     |   for expr in casos_invalidos:
19     |       |   assert validador_expressao(expr) is False
20
21 # ciclo 6
22 def test_deveRejeitarParentesesDesbalanceados():
23     |   casos_invalidos = ["(1+2", "1+2)", ")("]
24     |   for expr in casos_invalidos:
25     |       |   assert validador_expressao(expr) is False
26
```

- https://github.com/LuizGust4vo/saleor_testes/blob/main/saleor/graphql/account/tests/mutations/authentication/test_token_create.py



3. Reflexão sobre a prática realizada (o que funcionou bem? o que foi mais difícil?)

Durante a prática de TDD, uma das maiores dificuldades que o grupo enfrentou foi a consecução dos diferentes ciclos, tendo cada um tendo de começar imediatamente após o término do outro para que o código gradualmente fosse incrementado. Isso colocou a comunicação do grupo à prova, uma vez que usamos um repositório para fazer essa atividade e tivemos de ter cuidado com as diferentes versões e commits feitos pelo outro para garantir que o código fosse gradualmente crescendo da versão inicial.

Tivemos algumas pequenas dificuldades iniciais com a importação de módulos em Python, sobre como seria feita a refatoração e a hierarquia de pastas que tivemos, mas eventualmente começamos a sanar esses problemas. Um dos nossos membros teve problemas com a questão da biblioteca Pytest, mas que eventualmente foi sanado também.