FGA 0238 - Testes de Software – Turma:	02	**Semestre:**	2023.2
Nome:	Raquel Temóteo Eucaria Pereira da Costa	**Matrícula:**	202045268
Equipe:	10 - Assertivos		

Atividade 4 – TDD

1 Funcionalidade

1.1 Identificação da Issue

Projeto: QuantiFGA

Issue: #199

1.2 Especificação

Descrição: Criar função que evite o uso de caminhos de forma hard coding

Objetivo: Tornar o código mais flexível para diferentes sistemas operacionais e Facilitar o manutenção

Tarefas: Usar variáveis ou configurações externas para passar os caminhos; Criar uma função que receba a string do caminho e do arquivo; Se o caminho não for encontrado deve receber uma mensagem de erro; Se o arquivo não for encontrado deve receber uma mensagem de erro

Comentários: Sugestão Usar biblioteca os

1.3 Descrição da Funcionalidade

Uma função que possibilita definir os caminhos de arquivos de forma que seja visível para qualquer sistema operacional e informe entradas inválidas e se não foi encontrado o arquivo.

1.4 Ciclos

Ciclo 1 - Verificar se a entrada é nula (class TestEncontrarArquivoCsv(unittest.TestCase)
Teste (test_encontrar_arquivo_csv_raises_value_error_for_null_input)

Ciclo 2 - Verificar se a entrada é uma string
Teste (test_encontrar_arquivo_csv_raises_value_error_para_entrada_nao_string)

Ciclo 3 - Verificar se a pasta existe
Teste (test_encontrar_arquivo_csv_folder_does_exist)

Ciclo 4 - Verificar se o arquivo foi encontrado
Teste (test_encontrar_arquivo_not_found)

2 Execução

2.1 Primeiro Ciclo - Verificar se as entradas na função são nulas.

Código base:

caminhoArquivoTDD.py U

backend > caminhoArquivoTDD.py > ...

```python
import os
import unittest

def encontrarArquivoCsv(nome_arquivo, pasta):
    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)
```

Código teste:

```python
class TestEncontrarArquivoCsv(unittest.TestCase):
    def test_encontrar_arquivo_csv_raises_value_error_for_null_input(self):
        with self.assertRaises(ValueError) as context:
            encontrarArquivoCsv(None, "pasta_qualquer")

        self.assertIn("O nome do arquivo e a pasta não podem ser nulos.", str(context.exception))
```

Execução do teste(falha):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA$ /bin/python3 /home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py
E
======================================================================
ERROR: test_encontrar_arquivo_csv_raises_value_error_for_null_input (__main__.TestEncontrarArquivoCsv)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 16, in test_encontrar_arquivo_csv_raises_value_error_for_null_input
    encontrarArquivoCsv(None, "pasta_qualquer")
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 7, in encontrarArquivoCsv
    arquivos = os.listdir(caminho_pasta)
FileNotFoundError: [Errno 2] No such file or directory: '/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/pasta_qualquer'

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (errors=1)
```

Refatoração:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

```

Execução do teste(sucesso):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA$ /bin/python3 /home/eucaria/Documentos/Github/2022-2-QuantiFGA/b
ackend/caminhoArquivoTDD.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

2.2 Segundo Ciclo - Verificar se a entrada é uma string

Código base:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

```

Código teste:

```python
class TestEncontrarArquivoCsv(unittest.TestCase):
    def setUp(self):
        self.pasta_teste = 'pasta_teste_temporaria'
        os.makedirs(self.pasta_teste)

    def tearDown(self):
        os.rmdir(self.pasta_teste)

    def test_encontrar_arquivo_csv_raises_value_error_para_entrada_nao_string(self):
        with self.assertRaises(ValueError) as context:
            encontrarArquivoCsv(123, self.pasta_teste)

        self.assertIn("O nome do arquivo e a pasta devem ser strings.", str(context.exception))

```

Execução do teste(falha):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA$ /bin/python3 /home/eucaria/Documentos/Github/2022-2-QuantiFGA/b
ackend/caminhoArquivoTDD.py
E
======================================================================
ERROR: test_encontrar_arquivo_csv_raises_value_error_para_entrada_nao_string (__main__.TestEncontrarArquivoCsv)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 43, in test_encontrar_arquivo_csv_rais
es_value_error_para_entrada_nao_string
    encontrarArquivoCsv(123, self.pasta_teste)
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 27, in encontrarArquivoCsv
    arquivos = os.listdir(caminho_pasta)
FileNotFoundError: [Errno 2] No such file or directory: '/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/pasta_teste_tempo
raria'

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (errors=1)
```

Refatoração:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

```

Execução do teste(sucesso):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA$ /bin/python3 /home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

2.3 Terceiro Ciclo - Verificar se a pasta existe

Código base:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

```

Código teste:

```python
class TestEncontrarArquivoCsv(unittest.TestCase):
    def setUp(self):
        self.pasta_teste = 'pasta_teste_temporaria'
        os.makedirs(self.pasta_teste)

    def tearDown(self):
        os.rmdir(self.pasta_teste)

    def test_encontrar_arquivo_csv_folder_does_exist(self):
        with self.assertRaises(ValueError) as context:
            encontrarArquivoCsv('arquivo.csv', 'pasta_inexistente')

        self.assertIn("A pasta pasta_inexistente não foi encontrada.", str(context.exception))

```

Execução do teste(falha):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA$ /bin/python3 /home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py
E
======================================================================
ERROR: test_encontrar_arquivo_csv_folder_does_exist (__main__.TestEncontrarArquivoCsv)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 73, in test_encontrar_arquivo_csv_folder_does_exist
    encontrarArquivoCsv('arquivo.csv', 'pasta_inexistente')
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 57, in encontrarArquivoCsv
    arquivos = os.listdir(caminho_pasta)
FileNotFoundError: [Errno 2] No such file or directory: '/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/pasta_inexistente'

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (errors=1)
```

Refatoração:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    if not os.path.exists(caminho_pasta):
            raise ValueError(f"A pasta {pasta} não foi encontrada.")

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)
```

Execução do teste(sucesso):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA$ /bin/python3 /home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

2.4 Quarto Ciclo - Verificar se o arquivo existe

Código base:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    if not os.path.exists(caminho_pasta):
            raise ValueError(f"A pasta {pasta} não foi encontrada.")

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)
```

Código teste:

```python
class TestEncontrarArquivoCsv(unittest.TestCase):
    def setUp(self):
        self.pasta_teste = 'pasta_teste_temporaria'
        os.makedirs(self.pasta_teste)

    def tearDown(self):
        os.rmdir(self.pasta_teste)

    def test_encontrar_arquivo_not_found(self):
        resultado = encontrarArquivoCsv('arquivo_inexistente.csv', self.pasta_teste)

        mensagem_esperada = f'O arquivo arquivo_inexistente.csv não foi encontrado na pasta {self.pasta_teste}'
        self.assertEqual(resultado, mensagem_esperada)
```

Execução do teste(falha):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA/backend$ python3 -m unittest caminhoArquivoTDD.py
F
======================================================================
FAIL: test_encontrar_arquivo_not_found (caminhoArquivoTDD.TestEncontrarArquivoCsv)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/eucaria/Documentos/Github/2022-2-QuantiFGA/backend/caminhoArquivoTDD.py", line 109, in test_encontrar_arquivo_not_fou
nd
    self.assertEqual(resultado, mensagem_esperada)
AssertionError: None != 'O arquivo arquivo_inexistente.csv não fo[40 chars]aria'

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (failures=1)
```

Refatoração:

```python
def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    if not os.path.exists(caminho_pasta):
            raise ValueError(f"A pasta {pasta} não foi encontrada.")

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

    return f'O arquivo {nome_arquivo} não foi encontrado na pasta {pasta}'
```

Execução do teste(sucesso):

```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA/backend$ python3 -m unittest caminhoArquivoTDD.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

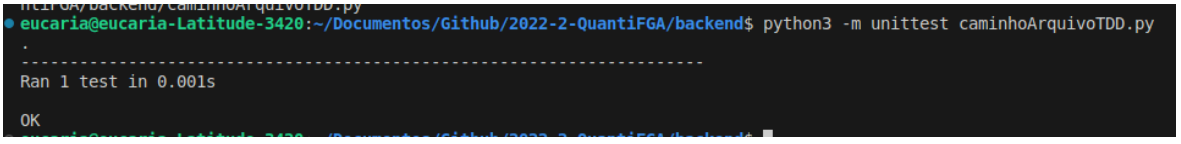
3 Código Fonte Testes

Versão final do código:

```python
import os
import unittest

def encontrarArquivoCsv(nome_arquivo, pasta):
    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

class TestEncontrarArquivoCsv(unittest.TestCase):
    def test_encontrar_arquivo_csv_raises_value_error_for_null_input(self):
        with self.assertRaises(ValueError) as context:
            encontrarArquivoCsv(None, "pasta_qualquer")

        self.assertIn("O nome do arquivo e a pasta não podem ser nulos.", str(context.exception))

def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

class TestEncontrarArquivoCsv(unittest.TestCase):
    def setUp(self):
        self.pasta_teste = 'pasta_teste_temporaria'
        os.makedirs(self.pasta_teste)

    def tearDown(self):
        os.rmdir(self.pasta_teste)

    def test_encontrar_arquivo_csv_raises_value_error_para_entrada_nao_string(self):
        with self.assertRaises(ValueError) as context:
            encontrarArquivoCsv(123, self.pasta_teste)

        self.assertIn("O nome do arquivo e a pasta devem ser strings.", str(context.exception))

def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)
```

```python
class TestEncontrarArquivoCsv(unittest.TestCase):
    def setUp(self):
        self.pasta_teste = 'pasta_teste_temporaria'
        os.makedirs(self.pasta_teste)

    def tearDown(self):
        os.rmdir(self.pasta_teste)

    def test_encontrar_arquivo_csv_folder_does_exist(self):
        with self.assertRaises(ValueError) as context:
            encontrarArquivoCsv('arquivo.csv', 'pasta_inexistente')

        self.assertIn("A pasta pasta_inexistente não foi encontrada.", str(context.exception))


def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    if not os.path.exists(caminho_pasta):
            raise ValueError(f"A pasta {pasta} não foi encontrada.")

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

class TestEncontrarArquivoCsv(unittest.TestCase):
    def setUp(self):
        self.pasta_teste = 'pasta_teste_temporaria'
        os.makedirs(self.pasta_teste)

    def tearDown(self):
        os.rmdir(self.pasta_teste)

    def test_encontrar_arquivo_not_found(self):
        resultado = encontrarArquivoCsv('arquivo_inexistente.csv', self.pasta_teste)

        mensagem_esperada = f'O arquivo arquivo_inexistente.csv não foi encontrado na pasta {self.pasta_teste}'
        self.assertEqual(resultado, mensagem_esperada)

def encontrarArquivoCsv(nome_arquivo, pasta):

    if nome_arquivo is None or pasta is None:
        raise ValueError("O nome do arquivo e a pasta não podem ser nulos.")

    if not isinstance(nome_arquivo, str) or not isinstance(pasta, str):
        raise ValueError("O nome do arquivo e a pasta devem ser strings.")

    caminho_pasta = os.path.join(os.path.dirname(os.path.abspath(__file__)), pasta)

    if not os.path.exists(caminho_pasta):
            raise ValueError(f"A pasta {pasta} não foi encontrada.")

    arquivos = os.listdir(caminho_pasta)

    for arquivo in arquivos:
        if nome_arquivo in arquivo:
            return os.path.join(caminho_pasta, arquivo)

    return f'O arquivo {nome_arquivo} não foi encontrado na pasta {pasta}'
```

4 Resultado Final Execução Testes



```
eucaria@eucaria-Latitude-3420:~/Documentos/Github/2022-2-QuantiFGA/backend$ python3 -m unittest caminhoArquivoTDD.py
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

5 Código Fonte da Funcionalidade Implementada

adiconando a função TDD — Navegar pelos arquivos

raqueleucaria comprometida 9 minutos atrás — 1 pai fac7207 cometer a20cc35

Mostrando **1 arquivo alterado** com **131 adições** e **0 exclusões** . — Espaço em branco | Ignorar espaços em branco | Dividir | Unificado

131 backend/caminhoArquivoTDD.py

@@ -0,0 +1,131 @@

```
+ importar  sistema operacional
+ importar  teste unitário
+
+ def  encontrarArquivoCsv ( nome_arquivo , pasta ):
+     caminho_pasta  =  os . caminho . join ( os . path . dirname ( os . path . abspath ( __file__ )), pasta )
+
+     arquivos  =  os . listdir ( caminho_pasta )
+
+     para  arquivo  em  arquivos :
+         if  nome_arquivo  no  arquivo :
+             retornar  os . caminho . juntar ( caminho_pasta , arquivo )
+
+ classe  TestEncontrarArquivoCsv ( unitest.TestCase ) : _
+     def  test_encontrar_arquivo_csv_raises_value_error_for_null_input ( self ):
+         consigo  mesmo . assertRaises ( ValueError ) como  contexto :
+             encontrarArquivoCsv ( None , "pasta_qualquer" )
+
+         auto . assertIn ( "O nome do arquivo e a pasta não podem ser nulos." , str ( context . exceção ))
+
+ def  encontrarArquivoCsv ( nome_arquivo , pasta ):
+
+     se  nome_arquivo  for  None  ou  pasta  for  None :
+         raise  ValueError ( "O nome do arquivo e a pasta não podem ser nulos." )
+
+     caminho_pasta  =  os . caminho . join ( os . path . dirname ( os . path . abspath ( __file__ )), pasta )
+
+     arquivos  =  os . listdir ( caminho_pasta )
+
+     para  arquivo  em  arquivos :
+         if  nome_arquivo  no  arquivo :
+             retornar  os . caminho . juntar ( caminho_pasta , arquivo )
+
+ classe  TestEncontrarArquivoCsv ( unitest.TestCase ) : _
+     def  configuração ( self ):
+         auto . pasta_teste  =  'pasta_teste_temporaria'
+         os . makedirs ( self . pasta_teste )
+
+     def  tearDown ( self ):
+         os . rmdir ( self . pasta_teste )
+
+     def  test_encontrar_arquivo_csv_raises_value_error_para_entrada_nao_string ( self ):
+         consigo  mesmo . assertRaises ( ValueError ) como  contexto :
+             encontrarArquivoCsv ( 123 , self . pasta_teste )
+
+         auto . assertIn ( "O nome do arquivo e a pasta devem ser strings." , str ( context . exceção ))
+
+ def  encontrarArquivoCsv ( nome_arquivo , pasta ):
+
+     se  nome_arquivo  for  None  ou  pasta  for  None :
+         raise  ValueError ( "O nome do arquivo e a pasta não podem ser nulos." )
+
+     se  não for  isinstance ( nome_arquivo , str ) ou  não for  isinstance ( pasta , str ):
+         raise  ValueError ( "O nome do arquivo e a pasta devem ser strings." )
+
+     caminho_pasta  =  os . caminho . join ( os . path . dirname ( os . path . abspath ( __file__ )), pasta )
+
+     arquivos  =  os . listdir ( caminho_pasta )
+
+     para  arquivo  em  arquivos :
+         if  nome_arquivo  no  arquivo :
+             retornar  os . caminho . juntar ( caminho_pasta , arquivo )
+
+ classe  TestEncontrarArquivoCsv ( unitest.TestCase ) : _
+     def  configuração ( self ):
+         auto . pasta_teste  =  'pasta_teste_temporaria'
+         os . makedirs ( self . pasta_teste )
+
+     def  tearDown ( self ):
+         os . rmdir ( self . pasta_teste )
+
+     def  test_encontrar_arquivo_csv_folder_does_exist ( self ):
+         consigo  mesmo . assertRaises ( ValueError ) como  contexto :
+             encontrarArquivoCsv ( 'arquivo.csv' , 'pasta_inexistente' )
+
+         auto . assertIn ( "Uma pasta pasta_inexistente não foi encontrada." , str ( context . exceção ))
+
+
+ def  encontrarArquivoCsv ( nome_arquivo , pasta ):
+
+     se  nome_arquivo  for  None  ou  pasta  for  None :
+         raise  ValueError ( "O nome do arquivo e a pasta não podem ser nulos." )
+
+     se  não for  isinstance ( nome_arquivo , str ) ou  não for  isinstance ( pasta , str ):
+         raise  ValueError ( "O nome do arquivo e a pasta devem ser strings." )
+
+     caminho_pasta  =  os . caminho . join ( os . path . dirname ( os . path . abspath ( __file__ )), pasta )
+
+     se  não  for . caminho . existe ( caminho_pasta ):
+             raise  ValueError ( f"Uma pasta { pasta } não foi encontrada." )
+
+     arquivos  =  os . listdir ( caminho_pasta )
+
+     para  arquivo  em  arquivos :
+         if  nome_arquivo  no  arquivo :
+             retornar  os . caminho . juntar ( caminho_pasta , arquivo )
+
+ classe  TestEncontrarArquivoCsv ( unitest.TestCase ) : _
+     def  configuração ( self ):
+         auto . pasta_teste  =  'pasta_teste_temporaria'
+         os . makedirs ( self . pasta_teste )
+
+     def  tearDown ( self ):
+         os . rmdir ( self . pasta_teste )
+
+     def  test_encontrar_arquivo_not_found ( self ):
+         resultado  =  encontrarArquivoCsv ( 'arquivo_inexistente.csv' , self . pasta_teste )
+
+         mensagem_esperada  =  f'O arquivo arquivo_inexistente.csv não foi encontrado na pasta { self . pasta_teste } '
+         auto . assertEqual ( resultado , mensagem_esperada )
+
```

```
+
+ def  encontrarArquivoCsv ( nome_arquivo , pasta ):
+
+     se  nome_arquivo  for  None  ou  pasta  for  None :
+         raise  ValueError ( "O nome do arquivo e a pasta não podem ser nulos." )
+
+     se  não for  isinstance ( nome_arquivo , str ) ou  não for  isinstance ( pasta , str ):
+         raise  ValueError ( "O nome do arquivo e a pasta devem ser strings." )
+
+     caminho_pasta  =  os . caminho . join ( os . path . dirname ( os . path . abspath ( __file__ )), pasta )
+
+     se  não  for . caminho . existe ( caminho_pasta ):
+             raise  ValueError ( f"Uma pasta { pasta } não foi encontrada." )
+
+     arquivos  =  os . listdir ( caminho_pasta )
+
+     para  arquivo  em  arquivos :
+         if  nome_arquivo  no  arquivo :
+             retornar  os . caminho . juntar ( caminho_pasta , arquivo )
+
+     return  f'O arquivo { nome_arquivo } não foi encontrado na pasta { pasta } '
+
```

Link do projeto: https://github.com/raqueleucaria/2022-2-QuantiFGA/
Link do commit: https://github.com/fga-eps-mds/2022-2-QuantiFGA/commit/a20cc357c961f5f88b6a4c58de46b40cec4c8596?diff=unified&w=0

6 Pull Request

Link do PR: https://github.com/fga-eps-mds/2022-2-QuantiFGA/pull/201

7 Conclusão

A incorporação do Desenvolvimento Orientado por Testes (TDD) transformou minha abordagem no desenvolvimento de software. Ao pensar mais nos requisitos e escrever testes antes da implementação das funcionalidades, não apenas assegurou a qualidade do código, mas também adotou uma mentalidade proativa em relação às possíveis falhas. Essa prática não apenas proporciona feedback imediato, mas também exerce uma influência positiva na arquitetura do código, gerando soluções mais robustas e testáveis.