**Solução para o Exercício TDD - Cálculo de Apdex**

**Implementação usando TDD (Test-Driven Development)**

**Passo 1: Definir os requisitos**

Com base no documento fornecido, precisamos criar uma função que calcule o índice Apdex com os seguintes parâmetros:

* satisfactory\_responses: número de respostas satisfatórias (≤ 500ms)
* tolerable\_responses: número de respostas toleráveis (>500ms e ≤2000ms)
* frustrated\_responses: número de respostas frustrantes (>2000ms)

A fórmula do Apdex é:

Apdex = (satisfactory\_responses + (tolerable\_responses / 2)) / total\_responses

**Passo 2: Criar os testes unitários primeiro**

import unittest

from apdex\_calculator import calculate\_apdex

class TestApdexCalculator(unittest.TestCase):

def test\_all\_satisfactory(self):

self.assertAlmostEqual(calculate\_apdex(100, 0, 0), 1.0)

def test\_all\_tolerable(self):

self.assertAlmostEqual(calculate\_apdex(0, 100, 0), 0.5)

def test\_all\_frustrated(self):

self.assertAlmostEqual(calculate\_apdex(0, 0, 100), 0.0)

def test\_mixed\_responses(self):

# Exemplo do documento: 270 satisfatórias, 70 toleráveis, 60 frustradas

self.assertAlmostEqual(calculate\_apdex(270, 70, 60), 0.7625)

def test\_no\_responses(self):

with self.assertRaises(ValueError):

calculate\_apdex(0, 0, 0)

def test\_negative\_values(self):

with self.assertRaises(ValueError):

calculate\_apdex(-10, 5, 3)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**Passo 3: Implementar a função (apdex\_calculator.py)**

def calculate\_apdex(satisfactory\_responses, tolerable\_responses, frustrated\_responses):

"""

Calcula o índice Apdex baseado nas respostas categorizadas.

Args:

satisfactory\_responses: Número de respostas ≤ 500ms

tolerable\_responses: Número de respostas >500ms e ≤2000ms

frustrated\_responses: Número de respostas >2000ms

Returns:

float: Valor do Apdex entre 0 e 1

Raises:

ValueError: Se qualquer parâmetro for negativo ou se todos forem zero

"""

# Validação dos parâmetros

if any(x < 0 for x in [satisfactory\_responses, tolerable\_responses, frustrated\_responses]):

raise ValueError("Números de respostas não podem ser negativos")

total = satisfactory\_responses + tolerable\_responses + frustrated\_responses

if total == 0:

raise ValueError("Pelo menos uma resposta deve ser fornecida")

# Cálculo do Apdex

apdex = (satisfactory\_responses + (tolerable\_responses / 2)) / total

return round(apdex, 4) # Arredonda para 4 casas decimais

**Passo 4: Executar os testes e refatorar**

Execute os testes para garantir que todos passem. Se algum teste falhar, ajuste a implementação até que todos os testes sejam aprovados.

**Passo 5: Exemplo de uso**

from apdex\_calculator import calculate\_apdex

# Exemplo do documento

satisfactory = 270

tolerable = 70

frustrated = 60

apdex = calculate\_apdex(satisfactory, tolerable, frustrated)

print(f"Apdex: {apdex}") # Deve imprimir: Apdex: 0.7625

# Interpretação

if apdex >= 0.94:

print("Excelente")

elif apdex >= 0.85:

print("Bom")

elif apdex >= 0.7:

print("Razoável")

elif apdex >= 0.5:

print("Ruim")

else:

print("Inaceitável")

**Classificação do Apdex**

| **Faixa do Apdex** | **Classificação** |
| --- | --- |
| 0.94 - 1.00 | Excelente |
| 0.85 - 0.93 | Bom |
| 0.70 - 0.84 | Razoável |
| 0.50 - 0.69 | Ruim |
| 0.00 - 0.49 | Inaceitável |

No exemplo fornecido no documento (Apdex = 0.76), a classificação seria "Razoável", indicando que há espaço para melhorias no tempo de resposta do sistema.

**Considerações finais**

Esta implementação segue rigorosamente a metodologia TDD:

1. Primeiro foram escritos os testes que definem o comportamento esperado
2. Depois foi implementada a função para passar nos testes
3. Por fim, os testes foram executados para validar a implementação

A função inclui tratamento de erros para casos inválidos (valores negativos ou zero respostas) e retorna o valor do Apdex arredondado para 4 casas decimais.