Plano de Teste

André Moura Pedroso

Planejamento de Teste para um sistema de cálculo de IMC (Índice de Massa Corporal) para a empresa NutriVitta

Testful

SENAI/SP

Jun - 2022

Sumário

INTRODUÇÃO

1.Modelo

Cálculo do IMC (Índice de Massa Corporal) para verificar o grau de obesidade.

2.Resumo sobre o sistema

O Índice de Massa Corporal (IMC) é utilizado para o controle de peso que apresenta uma escala de classificação para ser considerado normal e saudável. A tabela de IMC define os valores que indicam se o paciente está abaixo do peso, com peso normal ou acima do peso (Figura 1).

Categoria	IMC
Abaixo do peso	Abaixo de 18,5
Peso normal	18,5 - 24,9
Sobrepeso	25,0 - 29,9
Obesidade Grau I	30,0 - 34,9
Obesidade Grau II	35,0 - 39,9
Obesidade Grau III	40,0 e acima

Figura 1 – classificação do IMC (2017)

O cálculo para alcançar o valor a ser comparado na classificação do IMC deve ser feito considerando o peso (Kg) e altura (m) do paciente (Figura 2).

$$IMC = Peso\ (Kg) / Altura^2\ (m)$$

Figura 2 – fórmula para calcular o valor do IMC

3.Funcionalidades do sistema

Desta forma, o sistema a ser testado deve permitir que o usuário insira valores de peso e altura, apresentar um processamento com o cálculo de IMC,

comparação do resultado do cálculo com a classificação do IMC e uma resposta sobre o grau de abaixo do peso, normal, sobrepeso ou obesidade.

ESCOPO

1.Funcionalidades do sistema

O sistema apresenta os valores peso (racional), altura (racional) e IMC (racional). O peso e a altura serão fornecidos pelo usuário. O IMC será resultado do peso dividido pela altura ao quadrado. A classificação do IMC vai fornecer informações de saúde baseado no valor do IMC.

2.Testes

Os testes deverão verificar a funcionalidade do método desenvolvido para o cálculo do IMC e a comparação do valor do IMC com a classificação do IMC.

OBJETIVOS

1.Objetivo Geral

Testar as funcionalidades do sistema que define a saúde do paciente considerando o IMC.

2.Objetivo Específico 1

Testar o método de cálculo do IMC.

3.Objetivo Específico 2

Testar o método que compara o resultado do IMC com a classificação do IMC.

ESPECIFICAÇÃO DO PROJETO DE TESTE

1.Requisitos de teste

1.1.Teste Unitário 1 – Cálculo do IMC.

Peso e altura de indivíduos que representem todas as faixas de IMC na classificação do IMC devem ser verificados.

1.1.1.Casos de teste e Cenário esperado

1. **Peso** = 50, **Altura** = 1.70, **Resultado**: 17.301038062283737.
2. **Peso** = 60, **Altura** = 1.70, **Resultado**: 20.761245674740486.
3. **Peso** = 60, **Altura** = 1.50, **Resultado**: 26.666666666666668.
4. **Peso** = 110, **Altura** = 1.80, **Resultado**: 33.95061728395061.
5. **Peso** = 120, **Altura** = 1.80, **Resultado**: 37.03703703703704.
6. **Peso** = 130, **Altura** = 1.70, **Resultado**: 44.98269896193772.

1.1.2.Teste com erro

Erro: **Peso** = 50, **Altura** = 1.70, **Resultado**: 26.666666666666668.

1.2.Teste Unitário 2 – Comparação na Classificação do IMC.

Para a classificação do IMC foram utilizados valores numéricos que representam os níveis de obesidades presentes na tabela (Figura 3).

1	Abaixo do peso.
2	Peso Normal.
3	Sobrepeso.
4	Obesidade Grau I.
5	Obesidade Grau II.
6	Obesidade Grau III.

Figura 3 – valores numéricos presentes no teste para cada grau de obesidade.

1.2.1.Casos de teste e Cenário esperado

1. **IMC**: 17.301038062283737 – **Classificação**: 1
2. **IMC**: 20.761245674740486 – **Classificação**: 2
3. **IMC**: 26.666666666666668 – **Classificação**: 3
4. **IMC**: 33.95061728395061 – **Classificação**: 4
5. **IMC**: 37.03703703703704 – **Classificação**: 5
6. **IMC**: 44.98269896193772 – **Classificação**: 6

1.2.2.Teste com erro

Erro : **IMC**: 26.666666666666668 – **Classificação**: 4.

ESPECIFICAÇÃO DO PROCEDIMENTO DE TESTES

1.Ferramentas

a) Visual Studio 2022.
b) Biblioteca de Classes.
c) .NET versão 6.0.
d) Projeto de Teste do xUnit.

2.Estratégia de Teste

A Biblioteca de Classes fornece um ambiente básico de preparação dos métodos que deverão ser testados. Serão criados dois projetos na mesma solução, um projeto com a classe dos métodos a serem testados e o projeto de teste em si.

2.1.Projeto ProjetoNutiVitta

2.1.1.Classe Operacoes

Esta classe deverá ser pública para permitir o acesso em outros projetos da mesma solução e estática para facilitar o acesso aos métodos no momento da aplicação do teste (Act).

O método CalcularImc deve conter os parâmetros pNum para o valor peso e aNum para o valor altura. O retorno será (pNum / (aNum * aNum)) e os valores deverão ter o tipo racional podendo haver muitas casas decimais.

O método CompararImc terá um parâmetro chamado iNum para número racional. Haverão condicionais para as seguintes situações: iNum < 18.5 com retorno 1, iNum > 18.5 && iNum < 24.9 com retorno 2, iNum > 25.0 && iNum < 29.9 com retorno 3, iNum > 30.0 && iNum < 34.9 com retorno 4, iNum > 35.0 && iNum < 39.9 com retorno 5 e iNum > 40.0 com retorno 6. O padrão poderá ter retorno 0.

2.2.Projeto TestXUnit1

2.2.1.Classe UnitTest1

Esta classe deverá ser pública para não restringir o acesso a outros projetos da mesma solução e irá conter o teste unitário 1 e o teste unitário 2.

2.2.1.1.Teste Unitário 1

O método CalcularImcTest vai conter valores que deverão gerar situação de erro. No Arrange os valores serão os que foram definidos no Teste Com Erro (1.1.2). O Act vai armazenar na variável *resultado* o que foi obtido no método CalcularImc da Classe Operacoes utilizando as variáveis pNum e aNum definidas no Arrange. O Assert vai comparar o rNum definido no Arrange com o *resultado* obtido no Act. Para esta configuração deverá aparecer um erro no teste.

O método CalcularImcTestLista irá reunir todos os casos de teste que vão representar os 6 tipos de graus obtidos na tabela de classificação de IMC. No Arrange serão inseridos os valores que estão definidos no Casos de teste e

Cenário esperado (1.1.1). No Act o valor do resultado do método CalcularImc da classe Operacoes será armazenado na variável *resultado*. O Assert vai comparar os valores obtidos no cálculo com o resultado esperado. Todos os resultados deverão implicar em um acerto.

2.2.1.2.Teste Unitário 2

O método CompararImcTest vai conter valores para gerar uma situação de erro. No Arrange os valores utilizados podem ser visto no Teste Com Erro (1.2.2) o valor esperado é proposital para gerar erro no teste. No Act o resultado do método CompararImcTest da classe Operacoes é armazenado na variável *resultado,* o parâmetro atribuído deve ser o valor do IMC a ser consultado*.* O Assert vai comparar o valor esperado com o que foi armazenado na variável *resultado*.

O método CompararImcTestLista vai conter valores que representam todas as possiblidades contidas na tabela para a comparação do IMC, ao todo são 6. No Arrange são definidos valores para resultados de IMC para cada classificação e a classificação de cada possibilidade representada por números de 1 a 6 (Figura 3). O Act vai armazenar o resultado do método CompararImc que utiliza o valor do IMC para verificar a classificação do IMC. O Assert vai comparar os valores armazenados na variável definida no Act com o resultado esperado definido no Arrange, todos os testes desse método devem estar corretos.

3.Sistema utilizado

Processador: Intel(R) Core(TM) i5-3337U CPU @ 1.80 GHz
RAM instalada: 8.00 GB (utilizável: 7.90 GB)
Tipo de sistema: Sistema operacional de 64 bits, processador x64

4.Equipe

André Moura Pedroso

36 anos.

Desenvolvedor Full Stack.

Análise e Desenvolvimento de Sistemas – Faculdade Descomplica.

Desenvolvedor Full Stack – SENAI.

Experiências:

- App para Android (Java e Flutter/Dart);
- Games (Desktop e Android) com Pixel Art (Java);
- Web Pages (Angular/Typescript);
- APIs (C#);
- Banco de Dados (SQL/Oracle e Microsoft).

CRONOGRAMA

Instalação e preparação do ambiente	07/06/2022
Classe Operacoes método CalcularImc	07/06/2022
Classe Operacoes método CompararImc	07/06/2022
Preparação XUnit	07/06/2022
Classe UnitTest1 – Teste Unitário 1 Método CalcularImcTest	07/06/2022
Classe UnitTest1 – Teste Unitário 1 Método CalcularImcTestLista	07/06/2022
Classe UnitTest1 – Teste Unitário 2 Método CompararImcTest	07/06/2022
Classe UnitTest1 – Teste Unitário 2 Método CompararImcTestLista	07/06/2022
Execução do Teste	07/06/2022
Correção de falhas	07/06/2022
Execução do Teste 2	07/06/2022
Organização dos resultados	10/06/2022
Análise dos dados	10/06/2022
Diário de teste	10/06/2022
Relatório de incidentes	10/06/2022
Resumo de teste	10/06/2022
Relatórios finais	10/06/2022

DESENHO DE TESTE

1.ProjetoNutiVitta

Projeto criado com a Biblioteca de Classes C#.

1.1.Classe Operacoes

Classe criada para o desenvolvimento dos métodos necessários para o sistema de cálculo de IMC.

Operacoes.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ProjetoNutriVitta
{
    public static class Operacoes
    {
        public static double CalcularImc(double pNum, double aNum)
        {
            return (pNum / (aNum * aNum));
        }

        public static double CompararImc(double iNum)
        {
            if(iNum < 18.5)
            {
                return 1;
            }

            else if (iNum > 18.5 && iNum < 24.9)
            {
                return 2;
            }
```

```
            else if (iNum > 25.0 && iNum < 29.9)
            {
                return 3;
            }

            else if (iNum > 30.0 && iNum < 34.9)
            {
                return 4;
            }

            else if (iNum > 35.0 && iNum < 39.9)
            {
                return 5;
            }

            else if (iNum > 40.0)
            {
                return 6;
            }

            return 0;
        }
    }
}
```

2.TestXUnit1

Projeto criado com o xUnit para o desenvolvimento dos testes necessários para o projeto.

2.1.Classe UnitTest1

Classe criada para todos os métodos de testes do Teste Unitário 1 e 2.

UnitTest1.cs

```
using ProjetoNutriVitta;
```

```
namespace TestXUnit1
{
    public class UnitTest1
    {
        //Teste Unitário 1
        [Fact]
        public void CalcularImcTest()
        {
            //Arrange

            double pNum = 50;

            double aNum = 1.70;

            double rNum = 26.666666666666668;

            //Act

            var resultado = Operacoes.CalcularImc(pNum, aNum);

            //Assert

            Assert.Equal(rNum, resultado);
        }

        //Arrange
        [Theory]
        [InlineData(50, 1.70, 17.301038062283737)]
        [InlineData(60, 1.70, 20.761245674740486)]
        [InlineData(60, 1.50, 26.666666666666668)]
        [InlineData(110, 1.80, 33.95061728395061)]
        [InlineData(120, 1.80, 37.03703703703704)]
        [InlineData(130, 1.70, 44.98269896193772)]
        public void CalcularImcTestLista(double pNum, double aNum, double
rNum)
        {
            //Act

            var resultado = Operacoes.CalcularImc(pNum, aNum);

            //Assert
```

```
            Assert.Equal(rNum, resultado);
        }

        //Teste Unitário 2
        [Fact]
        public void CompararImcTest()
        {
            //Arrange

            double iPeso = 26.666666666666668;

            double rPeso = 4;

            //Act

            var resultado = Operacoes.CompararImc(iPeso);

            //Assert

            Assert.Equal(rPeso, resultado);
        }

        //Arrange
        [Theory]
        [InlineData(17.301038062283737, 1)]
        [InlineData(20.761245674740486, 2)]
        [InlineData(26.666666666666668, 3)]
        [InlineData(33.95061728395061, 4)]
        [InlineData(37.03703703703704, 5)]
        [InlineData(44.98269896193772, 6)]
        public void CompararImcTestLista(double iPeso, double rNum)
        {
            //Act

            var resultado = Operacoes.CompararImc(iPeso);

            //Assert

            Assert.Equal(rNum, resultado);
        }
    }
}
```