

PLANO DE TESTE SISTEMA IMC NUTRIVITTA

Afrânio Dimas da Silva

Contendo todas as informações relacionadas ao teste como, objetivo, escopo, cronograma e ferramentas.

Versão 1.0 – 2022

Seção Plano de teste	Descrição da seção	Modelo básico de plano de teste	Modelo padrão de plano de teste	Modelo de plano de teste de software	Modelo de plano de teste ágil	Modelo de plano de teste de sistema	Modelo de plano de teste de segurança do aplicativo da web	Modelo de plano de teste de software integrado
Resumo	Este plano de teste foi desenvolvido com o objetivo de definir as etapas, a equipe de teste e informando a mesma sobre o escopo do teste, sua abordagem, ambiente de teste, cronograma, resultados esperados e elaboração do relatório final de teste.	X	X	X	X	X	X	X
Escopo do Plano de Teste	Planejar as atividades a serem realizadas, definir os métodos a serem empregados, estabelecer as métricas e formas de acompanhamento do processo.			X				
Objetivos de Negócios	Melhorar o UX/UI do sistema.		X	X		X	X	X

Objetivos do Teste	Verificar o funcionamento do sistema a procura de possíveis erros e inconsistência.		X	X		X	X	X
Documentos Normativos e Informativos	Normas e diretrizes da Obeso e da Organização Mundial de Saúde (OMS).			X		X		X
Revisão Formal	Será criada uma equipe que ficará encarregada de analisar e manter a qualidade do teste.		X	X		X	X	X

Requisitos	Verificar o funcionamento do sistema a procura de possíveis erros e inconsistência.	X		X		X	X	X
Links de Coleção de Requisitos	Não há.		X	X	X	X		X
Links de Plano de Desenvolvimento	Não há.		X		X			
Avaliação de Risco	Possível lentidão de navegação do site em função de sobrecarregamento.		X	X		X		X
Planejamentos de Teste	Obter os dados a serem utilizados (nome da pessoa, sua altura e peso); inserir os dados no sistema; anotar os resultados obtidos e criar um relatório final do teste seguindo as datas previamente estabelecidas.	X	X	X		X	X	X
Estimativa de Teste	Será realizado dois testes a cada semana com dados de três pessoas diferentes.		X	X		X	X	X
Ambientes de Teste	Espaço interno com computador acessado a Internet, rodando o editor de texto Visual Studio e linguagem C# e Excel (para produção das documentações necessárias).		X	X	X	X	X	X

Detalhes do Ambiente de Teste de Software	Os testes serão realizados em ambiente interno, com iluminação e climatização e segurança adequados e contendo os recursos necessários a realização do teste.			X				
Estratégia de Amostragem	Como tamanho mínimo da amostra será estipulado o uso de dado de três (3) pessoas diferentes.					X		
Instalação de Testes	As etapas para preparar o plano de teste serão preparadas com antecedência de mínimo três (3) e não mais que cinco (5) dias.							X
Condições de Teste	Teste será realizado em ambiente interno, atendendo os requisitos necessários a realização do teste (ferramentas) e seguindo as normas aplicáveis da Obeso e OMS.					X		
Dados de Teste	Os dados coletados em decorrência do teste serão transferidos para um relatório final de Resultado de Teste.							X
Identificação do Teste	Teste Unitário - testará cada componente do sistema individualmente, verificando se esteja funcionando corretamente.			X				

Estratégia de Teste	Serão realizados três (3) teste a cada ciclo de teste.							X
Equipe de Teste	Teste realizado pelo desenvolvedor: Afrânio Dimas da Silva.		X	X		X	X	X
Instrumentos e Equipamento de Teste	O teste será realizado em um computador com Windows 10 versão 64 bits com processador Intel i7 com 2.9 GHz e 8GB de RAM e softwares de testes.					X		

Segurança do Aplicativo	Como segurança durante os testes serão utilizados antivírus além de Firewall.						X	
Objetivos de Qualidade	Os objetivos de qualidade para uma liberação, no formato de tabela.		X	X		X	X	X
CrITÉrios de Entrada	Inserção dos dados necessários para realização dos testes.		X	X		X	X	X
CrITÉrios de Saída	Resultado do cálculo do IMC que é a divisão do peso (em kg) pela altura ao quadrado (em metros).		X	X		X	X	X
Conjuntos de Testes	Será utilizado o peso e a altura de três pessoas diferentes.	X	X		X	X		X
Etapas de Teste	Inserção corretas dos dados e analisar retorno; Inserção de dados Inválidos e analisar retorno; inserção conjunta de dados válidos e inválidos e analisar retorno; Ausência de dados e analisar retorno.	X	X	X	X	X	X	X
Recursos	Computador com acesso à internet e softwares específicos ao teste.		X	X		X	X	X
Anexos	Serão adicionados documentos da Abeso e da Organização Mundial de Saúde (OMS).	X	X	X		X	X	X

❖ Caso de Testes

Os casos e testes foram estabelecidos com o objetivo de identificar a presença de erros e ou inconsistência que no sistema e na existência dos mesmos realizar sua correção.

Os erros e inconsistência que podem serem encontrados são:

- ✓ O recebimento de mensagem de erro ao inserir dados inválidos, ausência de um ou mais dados necessários ao teste e inserção simultânea de dados válido com inválidos.
- ✓ Comparação de dados diferentes;
- ✓ Retorno de resultados incompatíveis com os dados inseridos;
- ✓ Falha no retorno de dados em função de inconsistência nos operadores.

❖ Tabela de casos de teste

Os testes são baseados no cálculo do Índice de Massa Corporal - IMC definido pela Obeso que é calculado dividindo o peso (em kg) pela altura ao quadrado (em metros) e sua classificação (Figura 1) em função do resultado encontrado.

Figura 1 Classificação do IMC da Obeso.

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

Cenário	Caso de Teste	Entradas	Resultados Esperados Conforme IMC
Cenário 1	Calcular ICM da pessoa1	Inserir altura (1,70 metros)	26,64 = Sobrepeso
		Inserir peso (77 kg)	

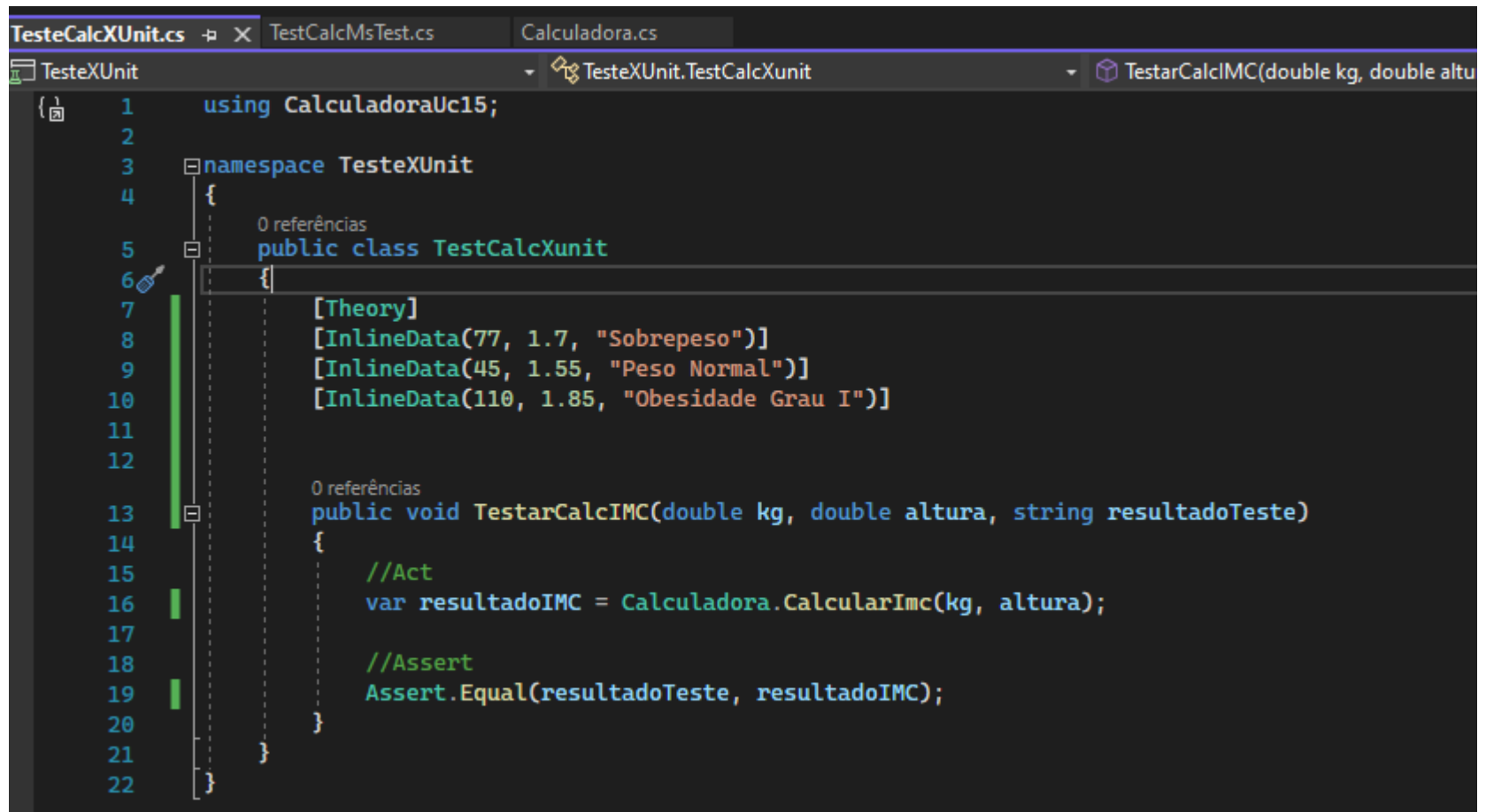
Cenário	Caso de Teste	Entradas	Resultados Esperados Conforme IMC
Cenário 2	Calcular ICM da pessoa2	Inserir altura (1,55 metros)	18,73 = Peso normal
		Inserir peso (45 kg)	

Cenário	Caso de Teste	Entradas	Resultados Esperados Conforme IMC
Cenário 3	Calcular ICM da pessoa3	Inserir altura (1,85 metros)	32,14 = Obesidade Grau I
		Inserir peso (110 kg)	

❖ Relatório final de teste

Após a finalização do teste, os resultados obtidos e as conclusões formuladas serão transferidas para um relatório final e este definirá se o produto no caso a calculadora do ICM irá ser produzido.

❖ Desenho do Teste



```
TesteCalcXUnit.cs  TestCalcMsTest.cs  Calculadora.cs
TesteXUnit
  TesteXUnit.TestCalcXUnit
    TestarCalcIMC(double kg, double altura)

1  using CalculadoraUc15;
2
3  namespace TesteXUnit
4  {
5      public class TestCalcXUnit
6      {
7          [Theory]
8          [InlineData(77, 1.7, "Sobrepeso")]
9          [InlineData(45, 1.55, "Peso Normal")]
10         [InlineData(110, 1.85, "Obesidade Grau I")]
11
12
13         public void TestarCalcIMC(double kg, double altura, string resultadoTeste)
14         {
15             //Act
16             var resultadoIMC = Calculadora.CalcularImc(kg, altura);
17
18             //Assert
19             Assert.Equal(resultadoTeste, resultadoIMC);
20         }
21     }
22 }
```

```
CalcIMC  CalculadoraUc15.Calculadora
1 namespace CalculadoraUc15
2 {
3     3 referências
4     public static class Calculadora
5     {
6
7         1 referência
8         public static string CalcularImc(double kg, double altura)
9         {
10             double resultado = kg / (altura * altura);
11
12             if (resultado < 18.5)
13             {
14                 return "Abaixo do peso";
15             }
16             if (resultado < 24.9)
17             {
18                 return "Peso normal";
19             }
20             if (resultado < 29.9)
21             {
22                 return "Sobre peso";
23             }
24             if (resultado < 34.9)
25             {
26                 return "Obesidade grau 1";
27             }
28         }
29     }
30 }
```



```
27     if (resultado < 39.9)
28     {
29         return "Obesidade grau 2";
30     }
31     else
32     {
33         return "Obesidade grau 3";
34     }
35
36     //Para classes comuns
37     //Calculadora calc = new Calculadora;
38     //cal.Somar();
39
40
41     //Para classes staticas (static)
42     //Claculadora.Somar();
43
44
45 }
46
47 }
```

❖ **Teste de integração do projeto API Web**

Cenário	Caso de Teste
Cenário 1	Teste do usuário com retorno inválido

Cenário	Caso de Teste
Cenário 2	Teste do usuário com retorno token

Cenário	Durante o teste não houve a necessidade de outros cenários de teste.
Cenário 3	

❖ Print dos Casos de Testes Realizados

```
namespace TesteIntegracao
{
    0 referências
    public class LoginControllerTeste
    {
        [Fact]
        0 referências
        public void LoginController_Retornar_Usuario_Invalido()
        {
            //Arrange - Preparação
            var repositoryEspelhado = new Mock<IUsuarioRepository>();

            repositoryEspelhado.Setup(x => x.Login(It.IsAny<string>(), It.IsAny<string>())).Returns((Usuario)null);

            var controller = new LoginController(repositoryEspelhado.Object);

            LoginViewModel dadosUsuario = new LoginViewModel();
            dadosUsuario.email = "batata@email.com";
            dadosUsuario.senha = "batata";

            //Act -Ação
            var resultado = controller.Login(dadosUsuario);

            //Assert - Verificação
            Assert.IsType<UnauthorizedObjectResult>(resultado);
        }
    }
}
```

[Fact]

0 referências

public void LoginController.Retornar_Token()

{

//Arrange - Preparação

Usuario usuarioRetornado = new Usuario();

usuarioRetornado.Email = "email@email.com";

usuarioRetornado.Senha = "1234";

usuarioRetornado.Tipo = "0";

usuarioRetornado.Id = 1;

var repositoryEspelhado = new Mock<IUsuarioRepository>();

repositoryEspelhado.Setup(x => x.Login(It.IsAny<string>(), It.IsAny<string>())) .Returns(usuarioRetornado);

LoginViewModel dadosUsuario = new LoginViewModel();

dadosUsuario.email = "batata@email.com";

dadosUsuario.senha = "batata";

var controller = new LoginController(repositoryEspelhado.Object);

string issuerValido = "chapter.webapi";

//Act - Ação

OkObjectResult resultado = (OkObjectResult)controller.Login(dadosUsuario);

string tokenString = resultado.Value.ToString().Split(' ')[3];

var jwtHandler = new JwtSecurityTokenHandler();

var tokenJwt = jwtHandler.ReadJwtToken(tokenString);

//Assert - Verificação

Assert.Equal(issuerValido, tokenJwt.Issuer);

}

}

Visual Studio interface showing a C# test file and test results.

Code Editor (LoginControllerTeste.cs):

```
53  
54     string issuerValido = "chapter.webapi";  
55  
56     //Act - Ação  
57     OkObjectResult resultado = (OkObjectResult)controller.Login(dadosUsuario);  
58  
59     string tokenString = resultado.Value.ToString().Split(' ')[3];  
60  
61     var jwtHandler = new JwtSecurityTokenHandler();  
62     var tokenJwt = jwtHandler.ReadJwtToken(tokenString);  
63  
64     //Assert - Verificação  
65     Assert.Equal(issuerValido, tokenJwt.Issuer);  
66  
67  
68 }
```

Gerenciador de Testes (Test Explorer):

Execução de teste anulada: 0 Testes (0 Aprovados, 0 Com falha, 0 Ignorados) executados em < 1 ms

Teste	Duração	Caracterís...	M
TesteIntegracao (2)	1,1 min		
TesteIntegracao (2)	1,1 min		
LoginControllerTeste (2)	1,1 min		
LoginController_Retomar_Tok...	3,5 s		
LoginController_Retomar_Usu...	1,1 min		

Resumo do grupo:

- TesteIntegracao
- Testes em grupo: 2
- Duração total: 1,1 min
- Resultados: 2 Aprovado

Solução 'ChapterBET6' (2 de 2 projetos):

- ChapterBET6
 - Connected Services
 - Dependências
 - Properties
 - Contexts
 - Controllers
 - Interfaces
 - Models
 - Repositories
 - ViewModels
 - appsettings.json
 - Program.cs
- TesteIntegracao
 - Dependências
 - LoginControllerTeste.cs
 - Usings.cs

Modelo de plano de teste clássico
X
X
X
X

X
X
X
X
X
X



X
X
X
X
X
X
X
X

