



Centro Universitário de Adamantina - UNIFAI

Credenciada nos termos da Portaria CEE/GP nº 235, de 13/07/2016
Autarquia Municipal - CNPJ: 03.061.303/0001-02

Rua Nove de Julho, 730 - CEP: 17800-000 - Adamantina/SP
Fone: (18) 3502-7010 - www.fai.com.br



SOFTWARE PARA GERENCIAMENTO DE SORVETERIA

Autores:

RA	ALUNO
0590/19	Júlio Henrique Conceição de Lima
0093/19	Vinicius Mesquini de Oliveira

ADAMANTINA – SP

2020

SUMÁRIO

Resumo do sistema.....	3
Levantamento de requisito.....	4
Métricas do sistema.....	5
Esforço estimado.....	6
Casos de teste.....	7
Diagrama de classes.....	13

1 - RESUMO DO SISTEMA

1.1 - Software Sorveteria

Trabalho feito para criação de um novo sistema para uma sorveteria, no qual consiste em, trazer agilidade nos processos de lançamento de vendas, e organização do empreendimento, no sistema estará atuando tanto na parte da frente, ajudando e auxiliando no atendimento com vendas, e cadastro de clientes, registro de pedidos, que leva um peso maior, e também estará atuando no “back” da empresa, controlando estoque, saídas, registros de cargas novas, estimativa de volume de produtos ainda existente para auxiliar na hora de escolher quais produtos comprar para repor as matérias primas; o sistema se encarregará de fechar o dia e fazer o balanço financeiro da empresa.

2- LEVANTAMENTO DE REQUISITOS

Identificação	Descrição	importância	Dependência	Peso
RF1	Cadastrar clientes	baixa		2
RF2	Registrar pedidos de fornecimento	media	RF1	3
RF3	Gerar Venda	Alta		5
RF4	Gerar nota do pedido completo fornecido	alta	RF3	5
RF5	Acionar aviso de dia de entrega e hora próximo	baixa	RF2	2
RF6	Cadastrar entrada de matérias primas	media		2
RF7	Consultar matérias primas	media	RF6	2
RF8	Registrar produtos feitos	alta		3
RF9	Descontar na matéria prima os gastos	alta	RF8	2
RF10	Acionar aviso fim da matéria prima	alta	RF9	2
RF11	Diminuir estoque produtos após venda	alta	RF3	3
RF12	Acionar aviso fim de produtos estoque	alta	RF11	3
RF13	Gerar fechamento de caixa diário	media		3
RF14	Relatório do fechamento do mês	media	RF13	3
RF15	Calcular Salários e descontos	alta	RF16	4
RF16	Cadastrar Funcionários	alta		4
RF17	Gerar relatório de produtos mais vendidos mês/ano	baixa		3
RFN1	Programa intuitivo e sequencial	media		3
RFN2	Facilidade para todos mexerem	media	RFN1	2
RFN3	velocidade na criação de relatórios	media		3
RFN4	Funcionar sem conexão também	alta		2
RFN5	Poder lançar pedidos por device mobile	media		3
RFN6	Agilidade na troca de telas para consulta	alta		2
RFN7	Boa organização das informações dos relatorios	Alta		3
Total (APF)				69

3 - MÉTRICAS PARA O PROJETO

3.1 - Métricas utilizada:

Análise de pontos de função (APF): mede o tamanho funcional do software, subsídios para o cálculo da produtividade do processo de desenvolvimento com base na funcionalidade ou utilidade dos programas.

3.2 - Calculo:

Cada requisito tem seu peso diante do sistema, o que mostra seu nível de importância e grau de impacto para um projeto funcional e de possível uso por parte do cliente, quanto mais itens de peso maior concluídos, mais perto do MVP (Minimum Viable Product).

3.3 - Exemplo:

Identf	peso
RF1	2
RF2	3
RF3	5
RF4	5
RF5	2
RF6	2
RF7	2
RF8	3
RF9	2
RF10	2
RF11	3
RF12	3
RF13	3
RF14	3
RF15	4
RF16	4
RF17	3
total	51

s 69 pontos de

RFN1	3
RFN2	2
RFN3	3
RFN4	2
RFN5	3
RFN6	2
RFN7	3
total	18

RFN1 RFN2 RFN3 RFN4 RFN5 RFN6 RFN7 total

RF1 RF2 RF3 RF4

Identf	peso
RF1	2
RF2	3
RF3	5
RF4	5

quisitos e 74% do

4 – CALCULANDO ESFORÇO

4.1 - Estimativa do esforço no desenvolvimento

Com base em todo o projeto e utilizando nossos 69 pontos de peso é possível calcular o esforço e quantas pessoas são necessárias no projeto. A media de produtividade é 3 PM/kloc, iremos trabalhar em um sistema com 16 mil linhas de código.

$$PM = (69 * (1 - 0 / 100)) / 3$$

$$PM = (69 * 1) / 3$$

$$PM = 69 / 3$$

$$PM = 23$$

Para concluirmos esse sistema será necessárias 23 pessoas.

Sabendo que o Esforço = $A * Tamanho^B * M$ e levando em conta valores padrões de $B = 1,29$ e $M = 1,63$ (Intermediário)

$$Esforço = 3 * 16^{1,29} * 1,63$$

$$Esforço = 3 * 35,75 * 1,63$$

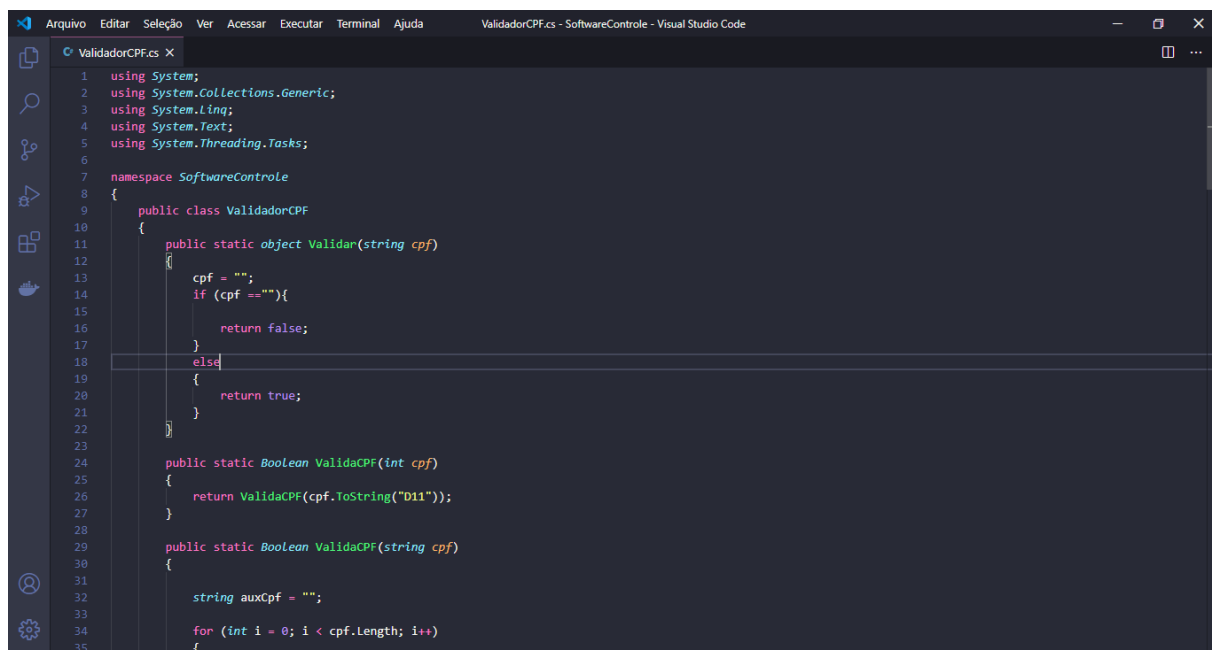
$$Esforço = 174,8175.$$

5 – CASOS DE TESTE

5.1 - NUnit

Teste unitário é uma técnica que consiste de colocar à prova as diversas funcionalidades de um sistema, testando-as em suas menores unidades. O NUnit, que foi usado nos testes desse sistema, é uma estrutura de teste de unidade para todas as linguagens .net. Inicialmente portado do JUnit, a versão de produção atual foi reescrita com muitos recursos novos e suporte para uma ampla variedade de plataformas.

Classe Validador CPF:

The image is a screenshot of the Visual Studio Code editor. The title bar at the top reads "ValidadorCPF.cs - SoftwareControle - Visual Studio Code". The editor window shows a C# file named "ValidadorCPF.cs". The code is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SoftwareControle
8 {
9     public class ValidadorCPF
10     {
11         public static object Validar(string cpf)
12         {
13             cpf = "";
14             if (cpf == ""){
15                 return false;
16             }
17             else{
18                 return true;
19             }
20         }
21
22         public static Boolean ValidaCPF(int cpf)
23         {
24             return ValidaCPF(cpf.ToString("D11"));
25         }
26
27         public static Boolean ValidaCPF(string cpf)
28         {
29             string auxCpf = "";
30             for (int i = 0; i < cpf.Length; i++)
31             {
```

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda ValidadorCPF.cs - SoftwareControl - Visual Studio Code

ValidadorCPF.cs X
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

    for (int i = 0; i < cpf.Length; i++)
    {
        if (isDigito(cpf.Substring(i, 1)))
        {
            auxCpf += cpf.Substring(i, 1);
        }
    }

    auxCpf = long.Parse(auxCpf).ToString("D11");

    if (auxCpf.Length > 11)
    {
        return false;
    }

    if (CalculaDigCPF(auxCpf.Substring(0, 9)) == auxCpf.Substring(9, 2))
    {
        return true;
    }

    return false;
}

public static string CalculaDigCPF(int cpf)
{
    return CalculaDigCPF(cpf.ToString("D9"));
}

public static string CalculaDigCPF(string cpf)
{
    string auxCpf = "";
    string digito = "";
    int aux1 = 0;
    int aux2 = 0;
```

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda ValidadorCPF.cs - SoftwareControl - Visual Studio Code

ValidadorCPF.cs X
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

    string digito = "";
    int aux1 = 0;
    int aux2 = 0;

    for (int i = 0; i < cpf.Length; i++)
    {
        if (isDigito(cpf.Substring(i, 1)))
        {
            auxCpf += cpf.Substring(i, 1);
        }
    }

    auxCpf = long.Parse(auxCpf).ToString("D9");

    if (auxCpf.Length > 9)
    {
        return null;
    }

    aux1 = 0;

    for (int i = 0; i < auxCpf.Length; i++)
    {
        aux1 += int.Parse(auxCpf.Substring(i, 1)) * (10 - i);
    }

    aux2 = 11 - (aux1 % 11);

    if (aux2 > 9)
    {
        digito += "0";
    }
    else
    {
        digito += aux2.ToString();
    }
}
```



```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda ValidadorCPF.cs - SoftwareControl - Visual Studio Code

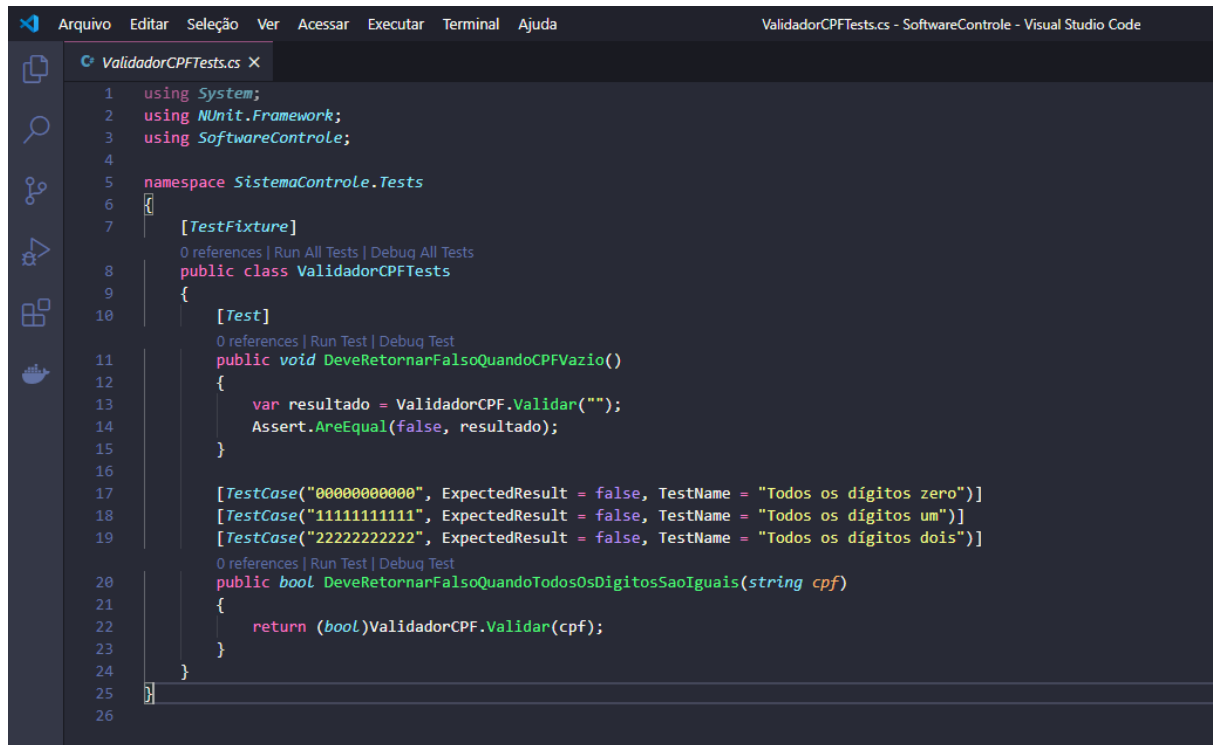
ValidadorCPF.cs X
94         if (aux2 > 9)
95         {
96             digito += "0";
97         }
98         else
99         {
100             digito += aux2.ToString();
101         }
102
103         auxCpf += digito;
104
105         aux1 = 0;
106
107         for (int i = 0; i < auxCpf.Length; i++)
108         {
109             aux1 += int.Parse(auxCpf.Substring(i, 1)) * (11 - i);
110         }
111
112         aux2 = 11 - (aux1 % 11);
113
114         if (aux2 > 9)
115         {
116             digito += "0";
117         }
118         else
119         {
120             digito += aux2.ToString();
121         }
122
123         return digito;
124     }
125
126     public static Boolean isDigito(string digito)
127     {
128         int n;
```

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda ValidadorCPF.cs - SoftwareControl - Visual Studio Code

ValidadorCPF.cs X
107         for (int i = 0; i < auxCpf.Length; i++)
108         {
109             aux1 += int.Parse(auxCpf.Substring(i, 1)) * (11 - i);
110         }
111
112         aux2 = 11 - (aux1 % 11);
113
114         if (aux2 > 9)
115         {
116             digito += "0";
117         }
118         else
119         {
120             digito += aux2.ToString();
121         }
122
123         return digito;
124     }
125
126     public static Boolean isDigito(string digito)
127     {
128         int n;
129         return Int32.TryParse(digito, out n);
130     }
131 }
132
133 }
```

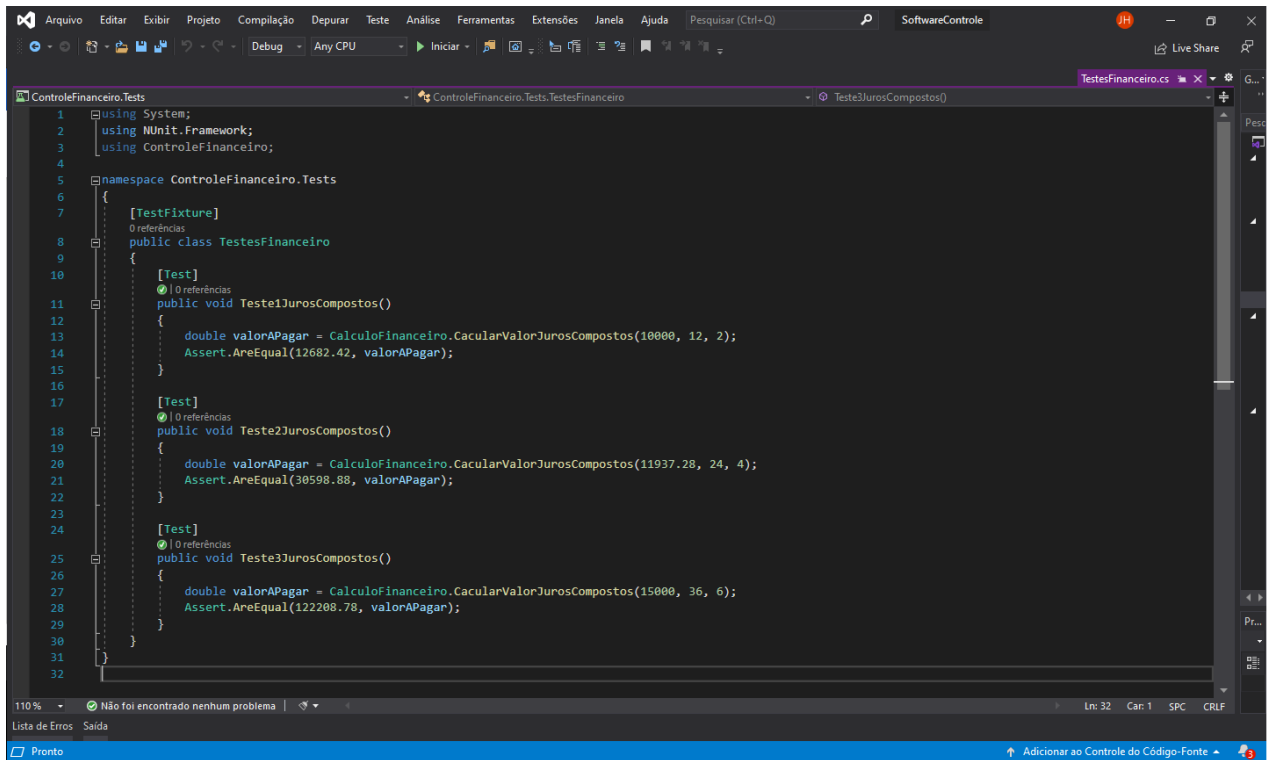
Classe de teste Validador CPF:

Obs. A anotação `TestFixture` marca uma classe como “classe de testes” e permite que ela seja reconhecida como tal no Test Explorer. Já a anotação `Test` define um método de teste.



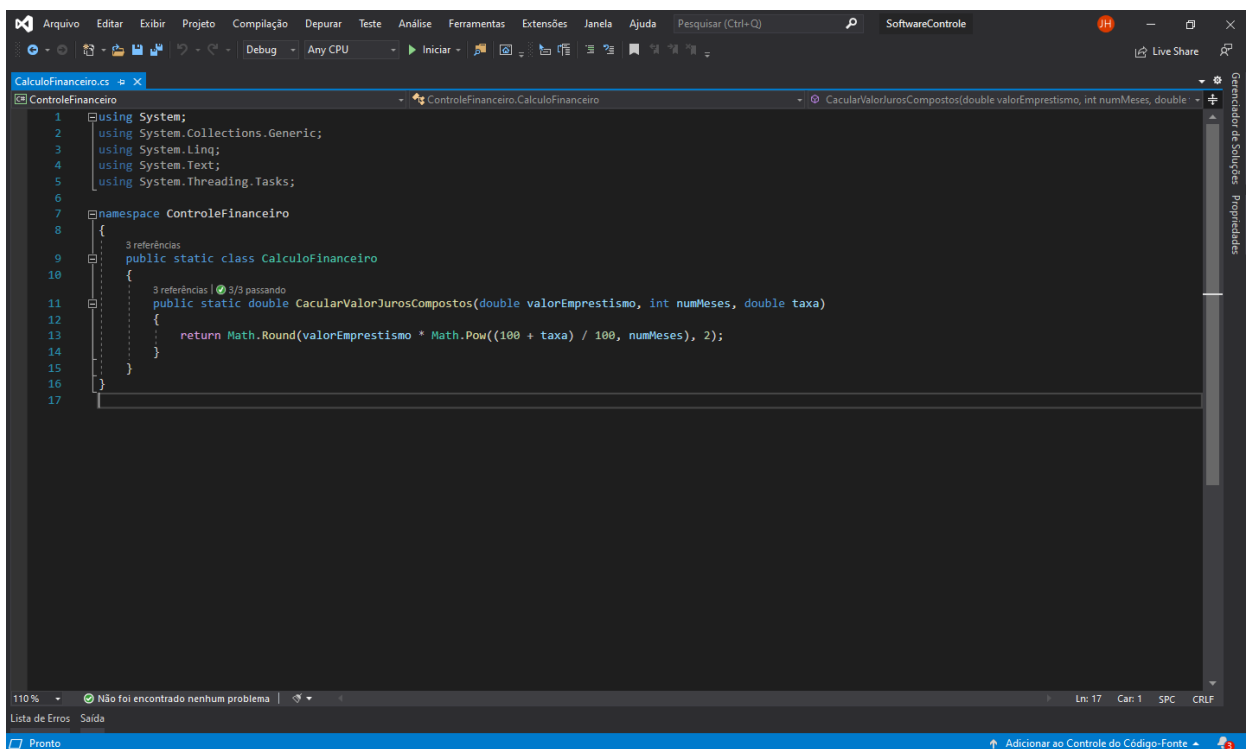
```
1 using System;
2 using NUnit.Framework;
3 using SoftwareControle;
4
5 namespace SistemaControle.Tests
6 {
7     [TestFixture]
8     public class ValidadorCPFTests
9     {
10         [Test]
11         public void DeveRetornarFalsoQuandoCPFVazio()
12         {
13             var resultado = ValidadorCPF.Validar("");
14             Assert.AreEqual(false, resultado);
15         }
16
17         [TestCase("0000000000", ExpectedResult = false, TestName = "Todos os dígitos zero")]
18         [TestCase("1111111111", ExpectedResult = false, TestName = "Todos os dígitos um")]
19         [TestCase("2222222222", ExpectedResult = false, TestName = "Todos os dígitos dois")]
20
21         public bool DeveRetornarFalsoQuandoTodosOsDigitosSaoIguais(string cpf)
22         {
23             return (bool)ValidadorCPF.Validar(cpf);
24         }
25     }
26 }
```

Classe de Testes Controle Financeiro:



```
1 using System;
2 using NUnit.Framework;
3 using ControleFinanceiro;
4
5 namespace ControleFinanceiro.Tests
6 {
7     [TestFixture]
8     public class TestesFinanceiro
9     {
10         [Test]
11         public void Teste1JurosCompostos()
12         {
13             double valorAPagar = CalculoFinanceiro.CacularValorJurosCompostos(10000, 12, 2);
14             Assert.AreEqual(12682.42, valorAPagar);
15         }
16
17         [Test]
18         public void Teste2JurosCompostos()
19         {
20             double valorAPagar = CalculoFinanceiro.CacularValorJurosCompostos(11937.28, 24, 4);
21             Assert.AreEqual(38598.88, valorAPagar);
22         }
23
24         [Test]
25         public void Teste3JurosCompostos()
26         {
27             double valorAPagar = CalculoFinanceiro.CacularValorJurosCompostos(15000, 36, 6);
28             Assert.AreEqual(122288.78, valorAPagar);
29         }
30     }
31 }
```

Classe cálculo financeiro:

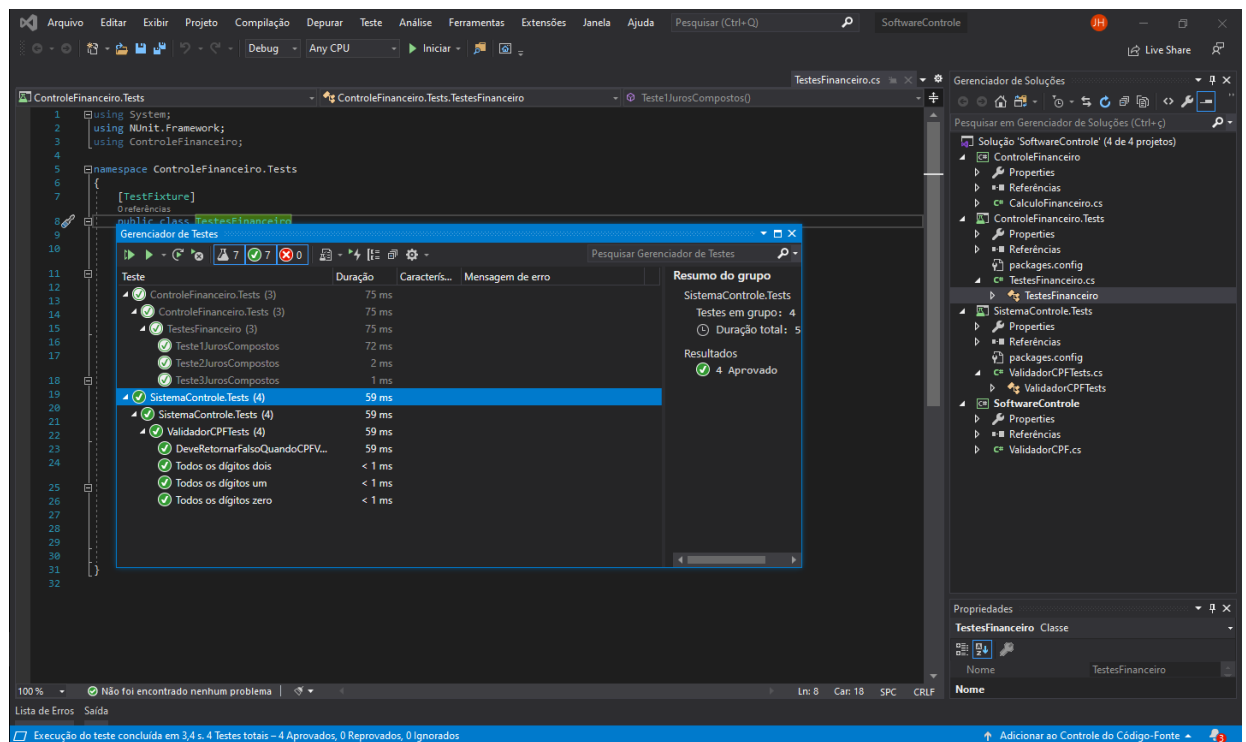


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ControleFinanceiro
8 {
9     public static class CalculoFinanceiro
10     {
11         public static double CacularValorJurosCompostos(double valorEmprestimo, int numMeses, double taxa)
12         {
13             return Math.Round(valorEmprestimo * Math.Pow((100 + taxa) / 100, numMeses), 2);
14         }
15     }
16 }
```

Resultado testes:

Validação do funcionamento dos métodos, o método de validação de CPF deve retornar falso quando o parâmetro for vazio, quando contiver letras ou caracteres diferentes de ponto e traço, quando o comprimento for maior ou menor que o esperado.

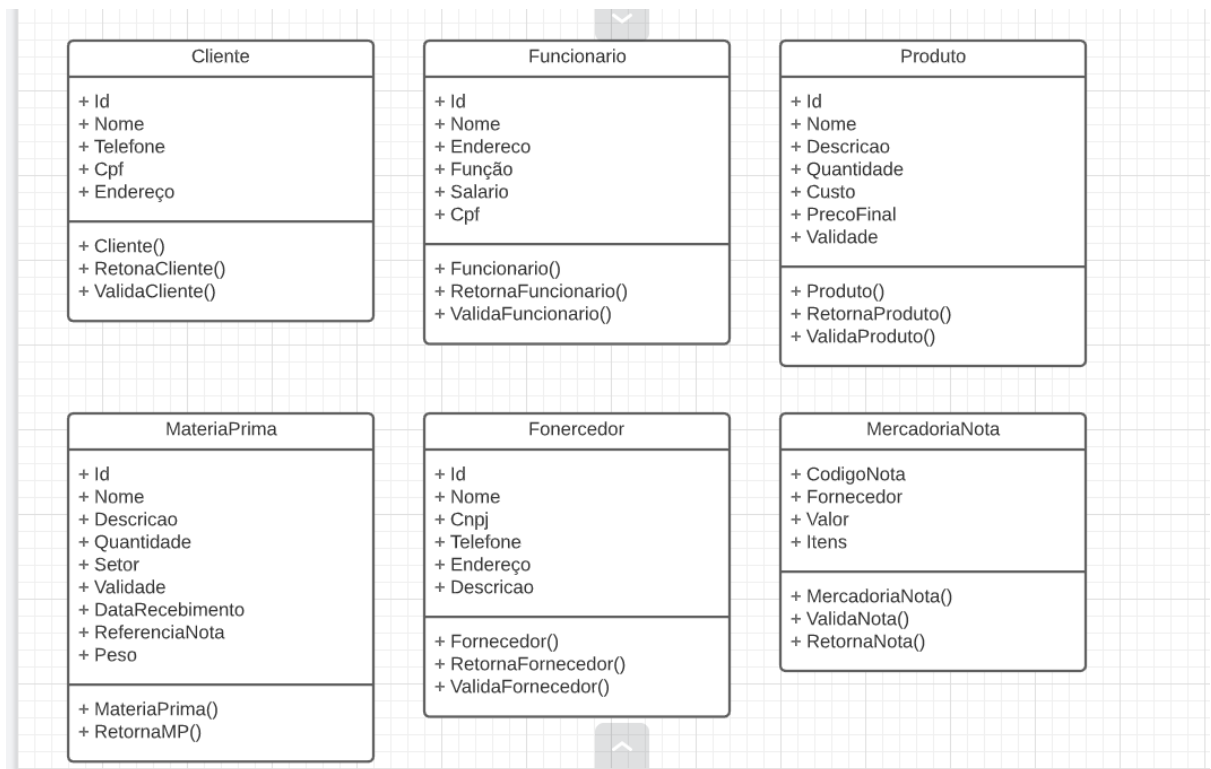
No caso do método de validação controle de financeiro, são calculados os juros compostos é verificado e testados 3 casos diferentes de inserção de valores o todos os testes são aceitos.



Classes desenvolvidas no Visual Studio Code e Teste realizados no Visual Studio.

6 – DIAGRAMA DE CLASSES

6.1 - Diagrama de classes UML – Lucidichart



6.2 - Classes implementadas em C#

```
C# Cliente.cs
C# Fornecedor.cs
C# Funcionario.cs
C# MateriaPrima.cs
C# MercadoriaNota.cs
C# Produto.cs
```

Classes desenvolvidas no Visual Studio

6.3 - Classes implementadas em C#

```
C# Cliente.cs X C# Funcionario.cs C# Produto.cs C# MateriaPrima.cs
C# Cliente.cs > {} ProjetoClass
1 namespace ProjetoClass
2 {
3     0 references
4     public class Cliente
5     {
6         0 references
7         int id;
8         0 references | 0 references | 0 references | 0 references
9         string nome, endereco, telefone, cpf;
10        0 references
11        public Cliente(){ ...
12    }
13    0 references
14    public void RetornaCliente(){ ...
15    }
16    0 references
17    public void ValidadeCliente(){ ...
18    }
19 }

C# Funcionario.cs > {} ProjetoClass > ProjetoClass.Funcionario > Funcionario()
1 namespace ProjetoClass
2 {
3     0 references
4     public class Funcionario
5     {
6         0 references
7         int id;
8         0 references | 0 references | 0 references | 0 references | 0 references
9         string nome, endereco, telefone, cpf, funcao;
10        0 references
11        double salario;
12    }
13    0 references
14    public Funcionario(){ ...
15    }
16    0 references
17    public void RetornaFuncionario(){ ...
18    }
19 }

C# Cliente.cs C# Funcionario.cs C# Produto.cs X C# MateriaPrima.cs
C# Produto.cs > {} ProjetoClass > ProjetoClass.Produto > ValidaProduto()
1 namespace ProjetoClass
2 {
3     0 references
4     public class Produto
5     {
6         0 references
7         int id;
8         0 references | 0 references | 0 references | 0 references
9         string nome, descricao, quantidade, validade;
10        0 references | 0 references
11        double custo, precoFinal;
12    }
13    0 references
14    public Produto(){ ...
15    }
16    0 references
17    public void RetornaProduto(){ ...
18    }
19    0 references
20    public void ValidaProduto(){ ...
21    }
22 }

C# Cliente.cs C# Funcionario.cs C# Produto.cs C# MateriaPrima.cs X Fornec
C# MateriaPrima.cs > {} ProjetoClass > ProjetoClass.MateriaPrima > RetornaMP()
1 namespace ProjetoClass
2 {
3     0 references
4     public class MateriaPrima
5     {
6         0 references
7         int id;
8         0 references
9         float peso;
10        0 references | 0 references | 0 references | 0 references | 0 references | 0 references
11        string nome, descricao, quantidade, setor, validade, refNota;
12    }
13    0 references
14    public MateriaPrima(){ ...
15    }
16    0 references
17    public void RetornaMP(){ ...
18    }
19 }

C# Fornecedor.cs X C# MercadoriaNota.cs
C# Fornecedor.cs > {} ProjetoClass > ProjetoClass.Fornecedor > ValidaForneced
1 namespace ProjetoClass
2 {
3     0 references
4     public class Fornecedor
5     {
6         0 references
7         int id;
8         0 references | 0 references | 0 references | 0 references | 0 references
9         string nome, endereco, telefone, cnpj, descricao;
10        0 references
11        public Fornecedor(){ ...
12    }
13    0 references
14    public void RetornaFornecedor(){ ...
15    }
16    0 references
17    public void ValidaFornecedor(){ ...
18    }
19 }

C# MercadoriaNota.cs X
C# MercadoriaNota.cs > {} ProjetoClass > ProjetoClass.MercadoriaNota
1 namespace ProjetoClass
2 {
3     0 references
4     public class MercadoriaNota
5     {
6         0 references | 0 references | 0 references
7         string codigoNota, fornecedor, itens;
8         0 references
9         double valor;
10    }
11    0 references
12    public MercadoriaNota(){ ...
13    }
14    0 references
15    public void RetornaNota(){ ...
16    }
17    0 references
18    public void ValidadeNota(){ ...
19    }
20 }
```