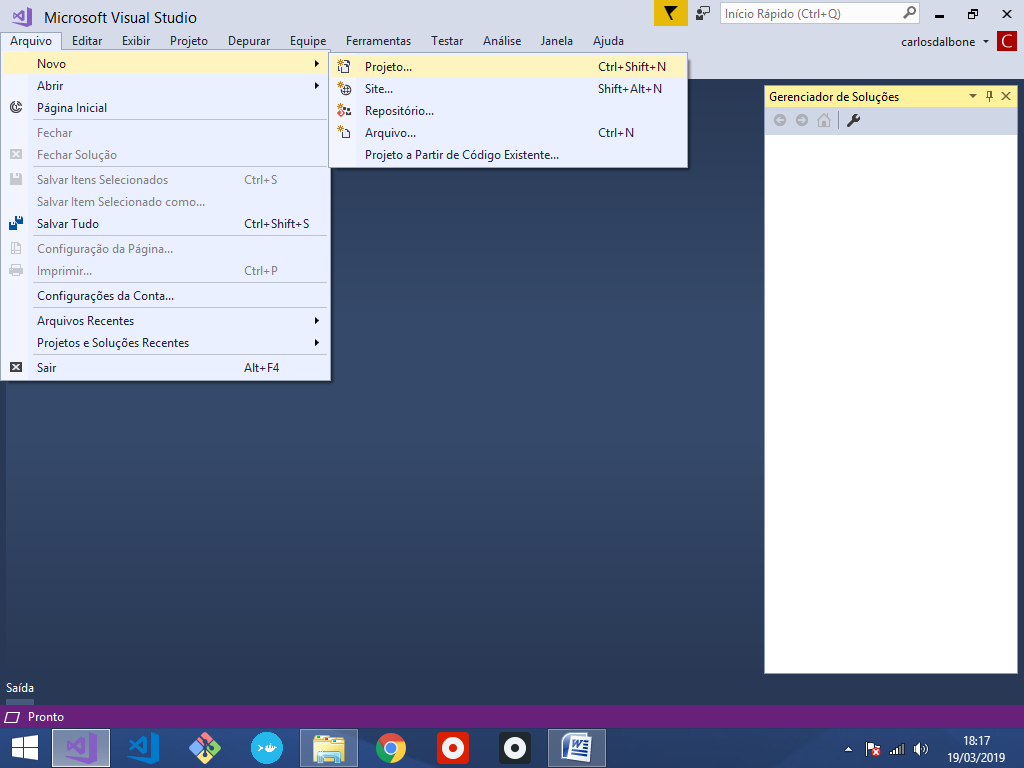
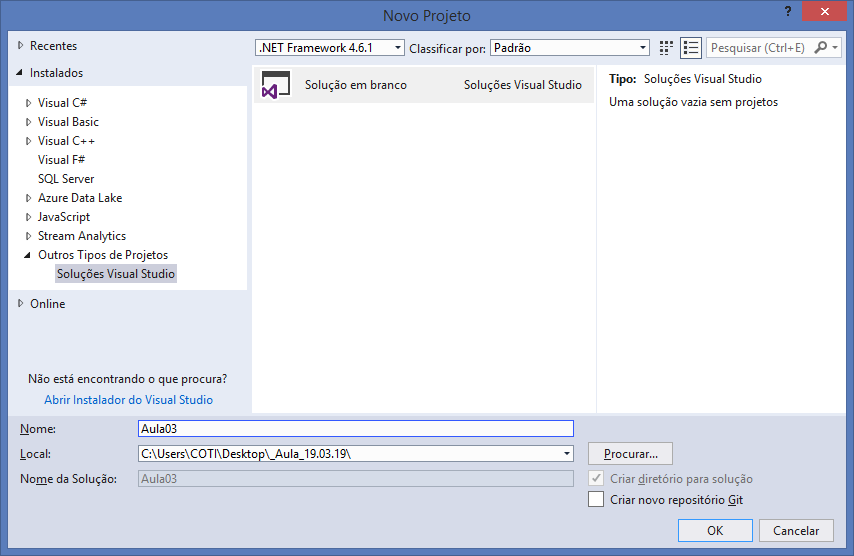
**Criando uma nova solution em branco:**





**Projeto Console Application**

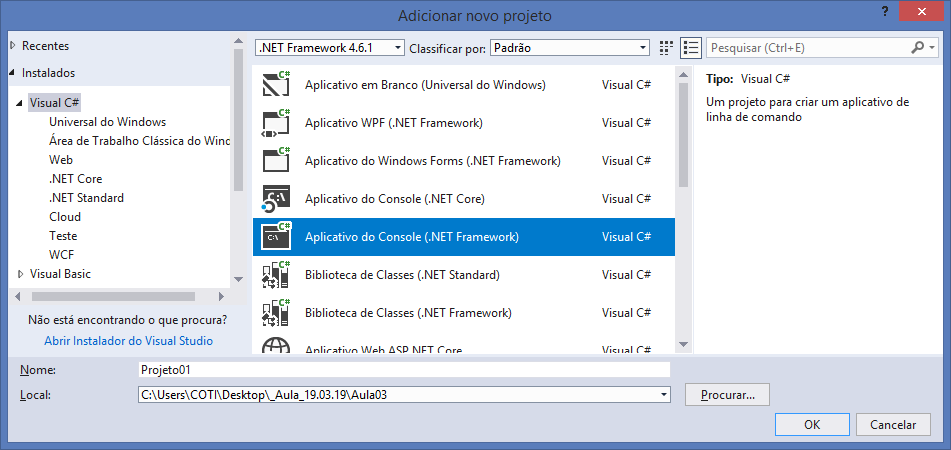
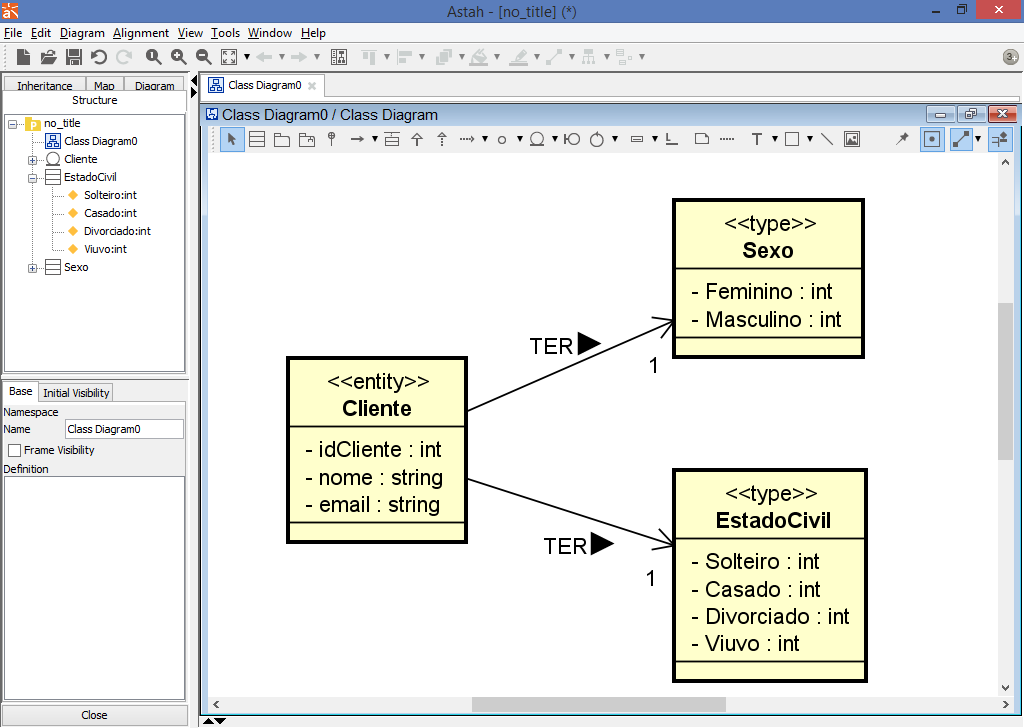


Diagrama de Classes

Modelagem de entidades



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto01.Entities.Types

{

public enum Sexo

{

Feminino = 'F',

Masculino = 'M'

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto01.Entities.Types

{

public enum EstadoCivil

{

Solteiro = 1,

Casado = 2,

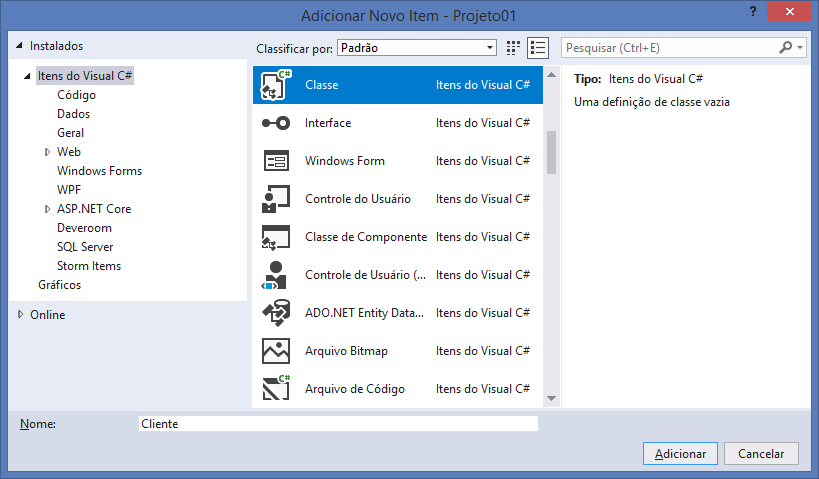
Divorciado = 3,

Viuvo = 4

}

}

**Criando a entidade Cliente:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities.Types; //importando

namespace Projeto01.Entities

{

public class Cliente

{

//prop + 2x[tab]

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public Sexo Sexo { get; set; }

public EstadoCivil EstadoCivil { get; set; }

}

}

Objeto:

Representa a instância de uma classe. Consiste de uma variável inicializada a partir do espaço de memória de uma classe. Por exemplo:

**Cliente cliente = new Cliente();**

[Classe] [Objeto] [**Método Construtor**]

Construtores

Toda classe criada em C# precisa de um método construtor para ser inicializada. O construtor é a função chamada após o uso da palavra reservada **new**.

**Cliente cliente = new Cliente();**

[Classe] [Objeto] [**Método Construtor**]

Por padrão, toda classe em C# já possui um construtor default, sem que este esteja escrito no código da classe. Se quisermos, podemos escrever o método construtor de forma explicita.

Por exemplo:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities.Types; //importando

namespace Projeto01.Entities

{

public class Cliente

{

//prop + 2x[tab]

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public Sexo Sexo { get; set; }

public EstadoCivil EstadoCivil { get; set; }

**//ctor + 2x[tab]**

**public Cliente()**

**{**

**//método construtor**

**}**

}

}

Sobrecarga de Métodos (Overloading)

Ocorre quando uma classe em C# declara métodos com o mesmo nome, porem com entrada de argumentos (parâmetros) diferentes.

Por exemplo:

public class A

{

**public void Imprimir()**

{

Console.WriteLine("Olá");

}

**public void Imprimir(string nome)**

{

Console.WriteLine("Olá " + nome);

}

**public void Imprimir(string nome, string sobrenome)**

{

Console.WriteLine("Olá " + nome + " " + sobrenome);

}

**public void Imprimir(int numero)**

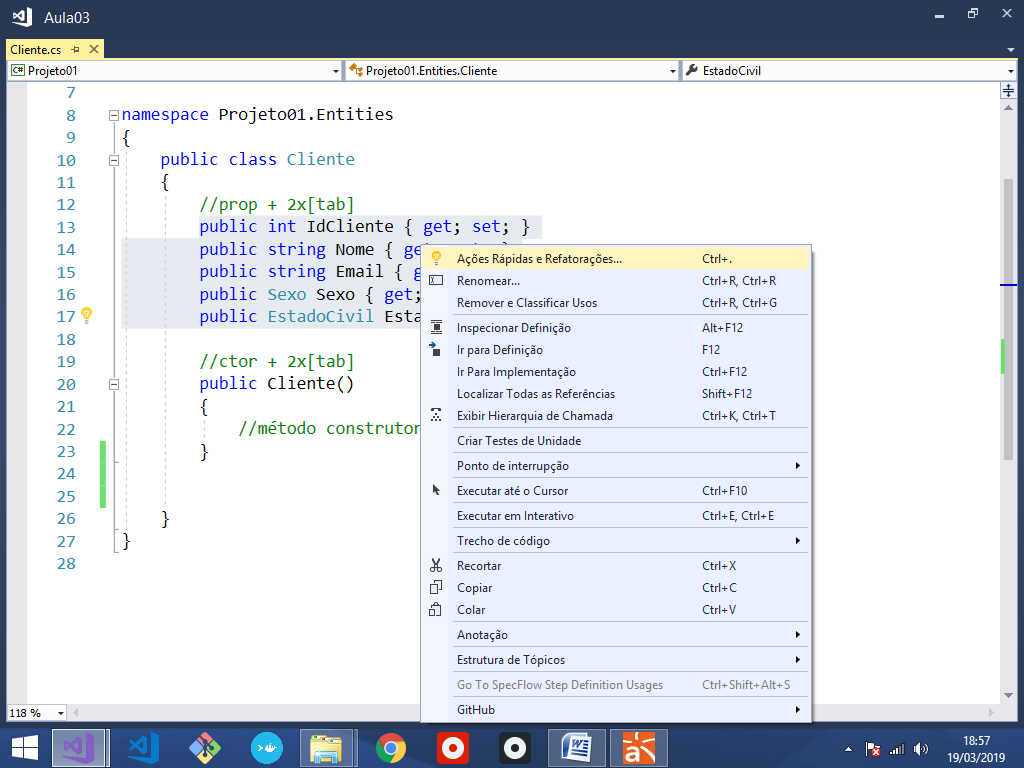
{

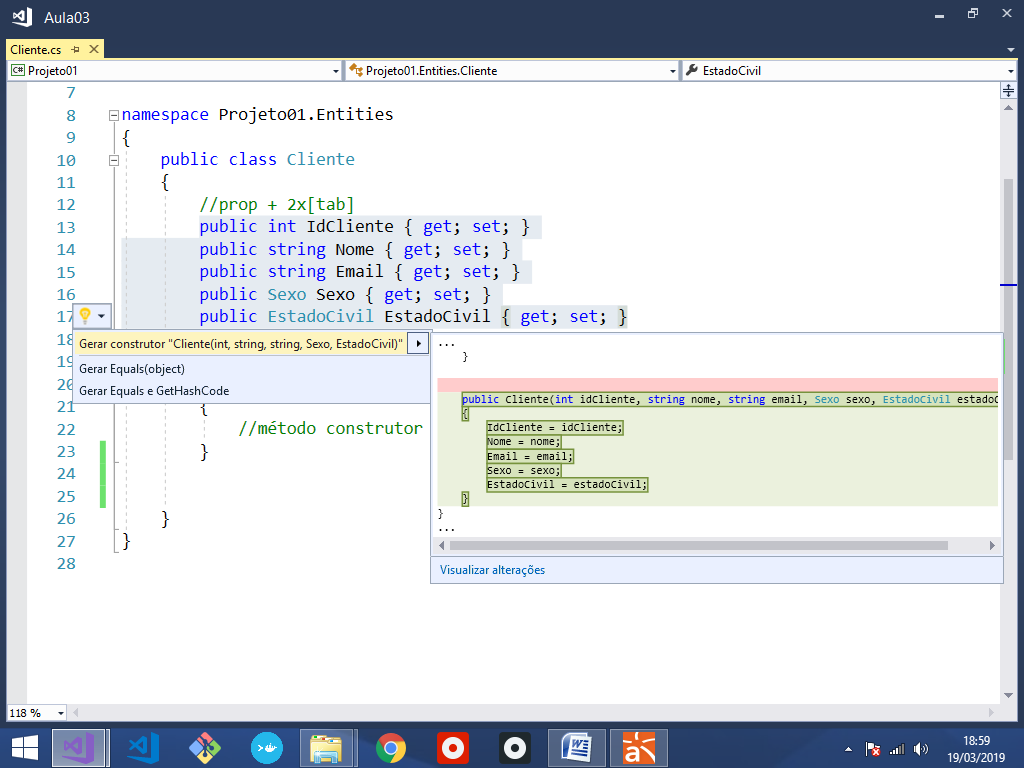
Console.WriteLine("Olá " + numero);

}

}

Podemos também fazer uma sobrecarga dos métodos construtores, ou seja, declarar em uma classe mais de um construtor contanto que eles tenham o mesmo nome porem entrada de argumentos diferentes.





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities.Types; //importando

namespace Projeto01.Entities

{

public class Cliente

{

//prop + 2x[tab]

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public Sexo Sexo { get; set; }

public EstadoCivil EstadoCivil { get; set; }

//ctor + 2x[tab]

public Cliente()

{

//método construtor

}

**//Sobrecarga do método Construtor**

**public Cliente(int idCliente, string nome, string email,**

**Sexo sexo, EstadoCivil estadoCivil)**

**{**

**IdCliente = idCliente;**

**Nome = nome;**

**Email = email;**

**Sexo = sexo;**

**EstadoCivil = estadoCivil;**

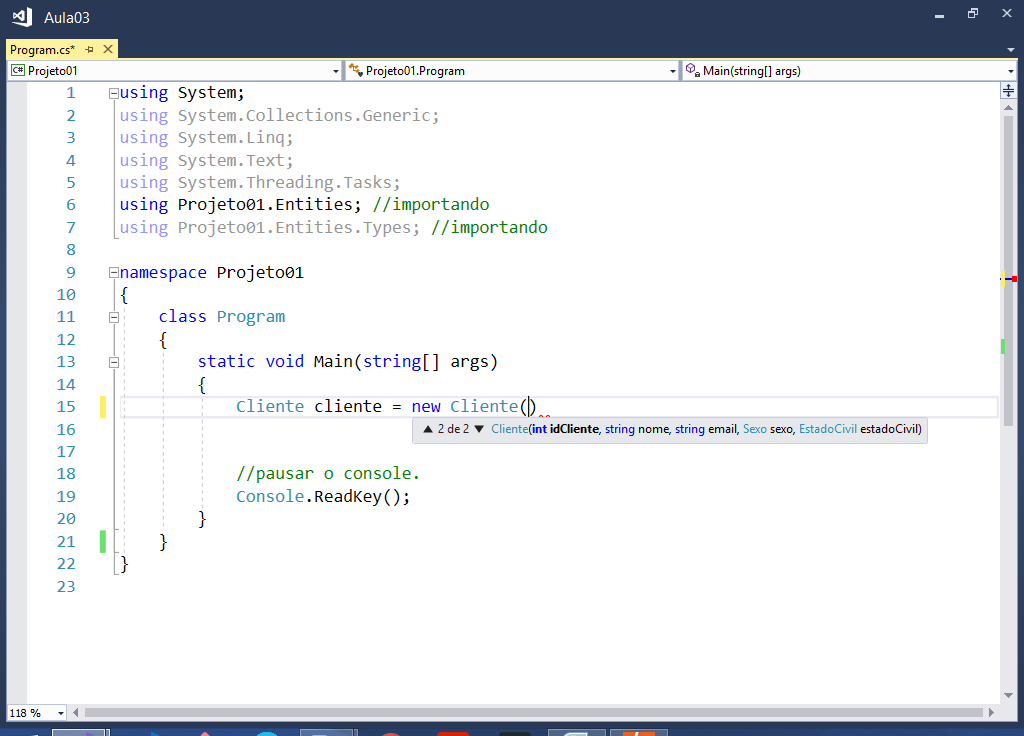
**}**

}

}

**Utilizando o construtor com entrada de argumentos:**

Voltando no método Main()



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities; //importando

using Projeto01.Entities.Types; //importando

namespace Projeto01

{

class Program

{

static void Main(string[] args)

{

**Cliente cliente = new Cliente(1, "Sergio Mendes",**

**"sergio.coti@gmail.com",**

**Sexo.Masculino, EstadoCivil.Solteiro);**

//pausar o console.

Console.ReadKey();

}

}

}

Sobrescrita de Métodos (Override)

Ocorre quando uma subclasse (classe-filha) sobrepõe / substitui um método da sua superclasse (classe-pai). Sobrescrita de métodos depende do uso de Herança entre classes.

**virtual**

Palavra reservada utilizada para permitir que um método seja sobrescrito por uma ou mais subclasses. Somente métodos **virtual** podem ter sobrescrita.

Por exemplo:

**public class A**

**{**

**public virtual void Imprimir()**

**{**

**Console.WriteLine("Imprime A");**

**}**

**}**

**public class B : A**

**{**

**public override void Imprimir()**

**{**

**Console.WriteLine("Imprime B");**

**}**

**}**

Toda classe em C# implicitamente HERDA **System.Object**.

Portanto, toda classe criada em C# já herda os métodos: ToString(), Equals(), GetHashCode() e GetType()

Na classe Object, os seguintes métodos são virtual:

(Ou seja, podemos sobrescrevê-los)

Estes métodos são:

* ToString()
* Equals()
* GetHashCode()

ToString()

Método utilizado para retornar o valor de um objeto em formato de texto.

Nas classes de entidade, podemos sobrescrevê-lo de forma a retornar em uma string os atributos concatenados da entidade.

Fazendo sobrescrita do método ToString()

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities.Types; //importando

namespace Projeto01.Entities

{

public class Cliente

{

//prop + 2x[tab]

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public Sexo Sexo { get; set; }

public EstadoCivil EstadoCivil { get; set; }

//ctor + 2x[tab]

public Cliente()

{

//método construtor

}

//Sobrecarga do método Construtor

public Cliente(int idCliente, string nome, string email,

Sexo sexo, EstadoCivil estadoCivil)

{

IdCliente = idCliente;

Nome = nome;

Email = email;

Sexo = sexo;

EstadoCivil = estadoCivil;

}

**//sobrescrita do método ToString()**

**public override string ToString()**

**{**

**//retornando os dados da entidade de forma concatenada..**

**return $"Id: {IdCliente}, Nome: {Nome}, Email: {Email} "**

**+ $"Sexo: {Sexo}, Estado Civil: {EstadoCivil}";**

**}**

}

}

**Testando:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities; //importando

using Projeto01.Entities.Types; //importando

namespace Projeto01

{

class Program

{

static void Main(string[] args)

{

Cliente cliente = new Cliente(1, "Sergio Mendes",

"sergio.coti@gmail.com",

Sexo.Masculino, EstadoCivil.Solteiro);

**Console.WriteLine("Cliente: " + cliente.ToString());**

//pausar o console.

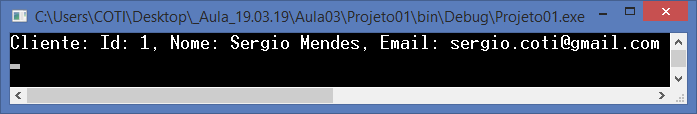
Console.ReadKey();

}

}

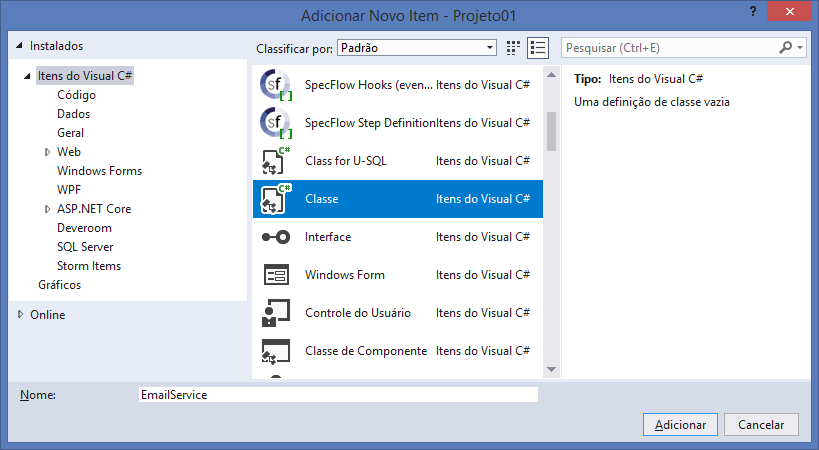
}

**Saída:**



Criando uma classe para enviar um email para o cliente

informando que o mesmo foi criado no projeto:



App.config.xml

Arquivo de configuração utilizado para definir parâmetros

que poderão ser utilizados e acessados por classes do projeto.

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

**<!-- Mapear as configurações para envio de email -->**

**<appSettings>**

**<add key="EMAIL" value="cotiexemplo@gmail.com"/>**

**<add key="SENHA" value="@coticoti@"/>**

**<add key="SMTP" value="smtp.gmail.com"/>**

**<add key="PORTA" value="587"/>**

**</appSettings>**

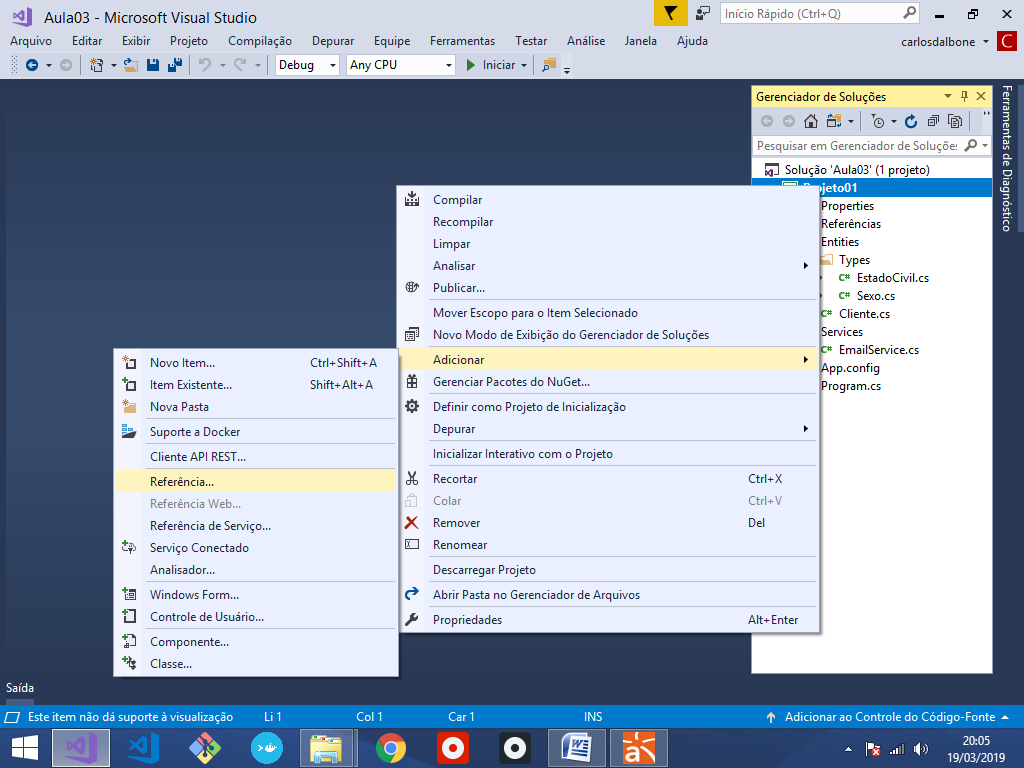
<startup>

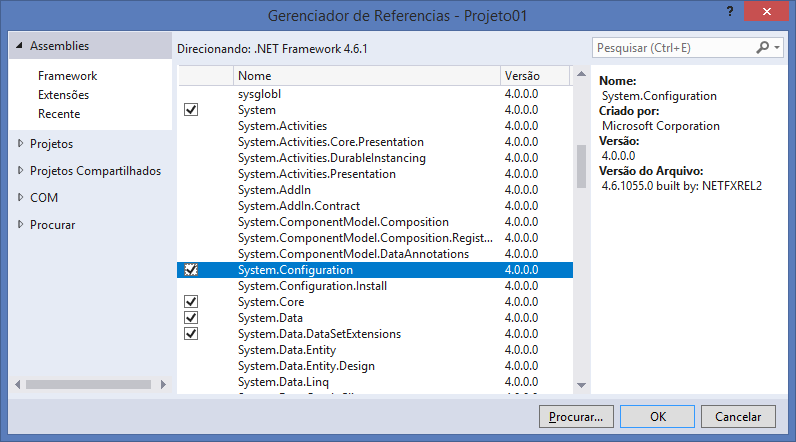
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />

</startup>

</configuration>

Para que possamos acessar o arquivo **App.config.xml** de forma a ler os valores declarados, precisamos adicionar no projeto uma referência para a biblioteca: System.Configuration





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities; //importando..

using System.Net; //importando..

using System.Net.Mail; //importando..

using System.Configuration; //importando..

namespace Projeto01.Services

{

public class EmailService

{

public void EnviarMensagem(Cliente cliente)

{

//definir o remetente da mensagem

MailAddress from = new MailAddress(ConfigurationManager

.AppSettings["EMAIL"]);

//definir o destinatário da mensagem

MailAddress to = new MailAddress(cliente.Email);

//Montar um email

MailMessage msg = new MailMessage(from, to);

//assunto da mensagem

msg.Subject = "Seja bem vindo ao Sistema";

//corpo da mensagem

msg.Body = $"Olá {cliente.Nome}\nSeu cadastro

foi realizado com sucesso.";

//enviar a mensagem..

SmtpClient smtp = new SmtpClient();

smtp.Host = ConfigurationManager.AppSettings["SMTP"];

smtp.Port = int.Parse(ConfigurationManager.AppSettings["PORTA"]);

//autenticar na conta de email do remetente

NetworkCredential credential = new NetworkCredential();

credential.UserName = ConfigurationManager.AppSettings["EMAIL"];

credential.Password = ConfigurationManager.AppSettings["SENHA"];

//enviando..

smtp.Credentials = credential;

smtp.EnableSsl = true; //segurança

smtp.Send(msg); //disparando a mensagem!

}

}

}

**Executando:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto01.Entities; //importando

using Projeto01.Entities.Types; //importando

using Projeto01.Services; //importando

namespace Projeto01

{

class Program

{

static void Main(string[] args)

{

Cliente cliente = new Cliente(1, "Sergio Mendes",

"sergio.coti@gmail.com",

Sexo.Masculino, EstadoCivil.Solteiro);

Console.WriteLine("Cliente: " + cliente.ToString());

**try**

**{**

**EmailService emailService = new EmailService();**

**emailService.EnviarMensagem(cliente);**

**Console.WriteLine("\nEmail enviado com sucesso!");**

**}**

**catch(Exception e)**

**{**

**Console.WriteLine("\nOcorreu um erro: " + e.Message);**

**}**

//pausar o console.

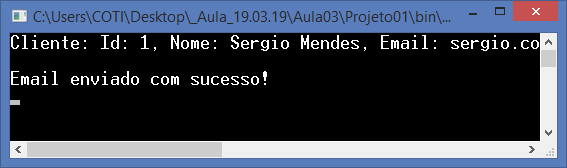
Console.ReadKey();

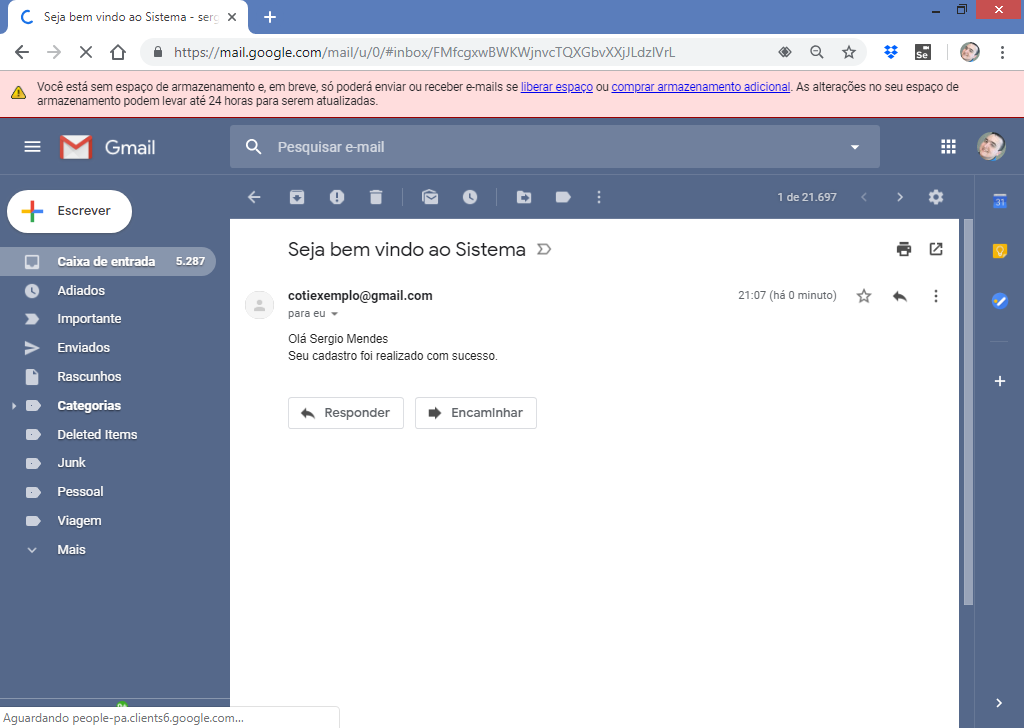
}

}

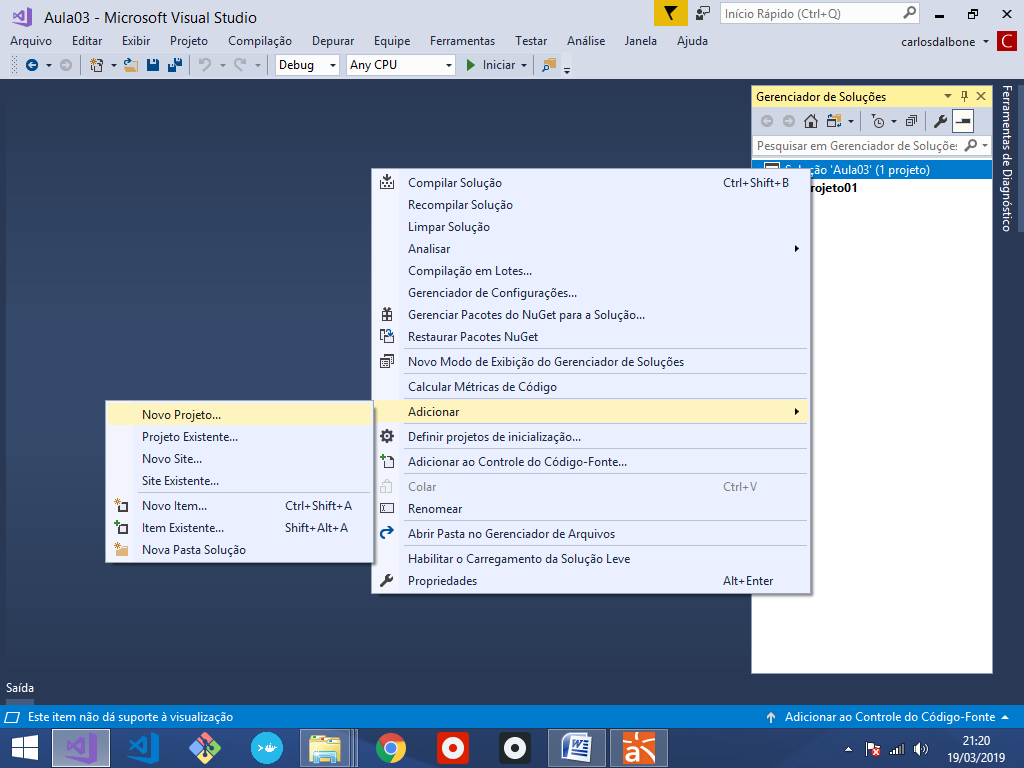
}

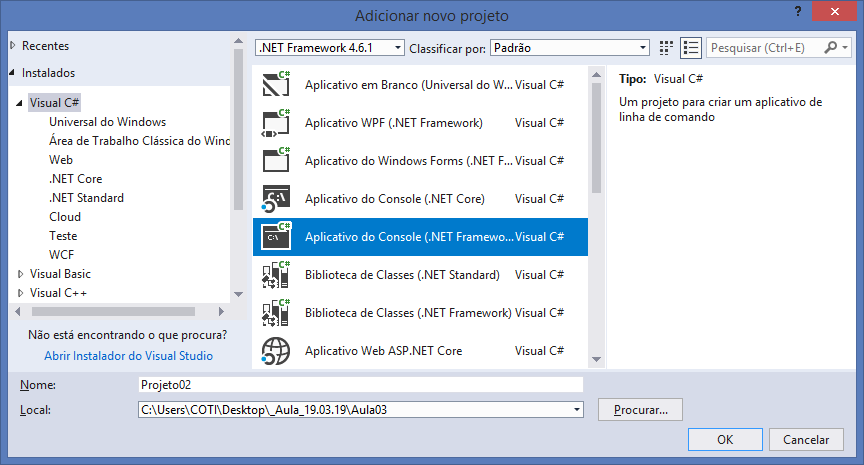
Saída do programa:

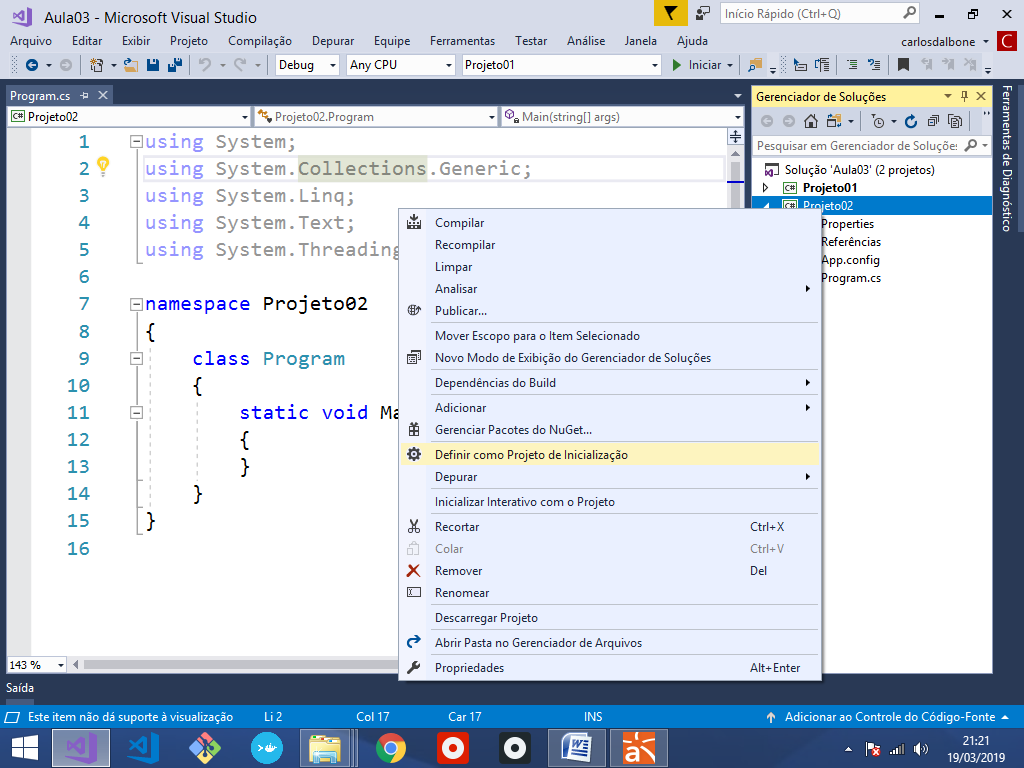




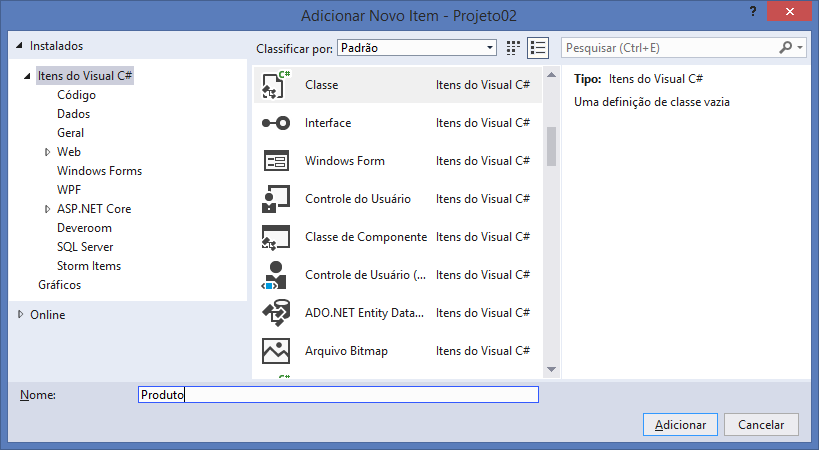
**Criando um novo projeto:**







**Entidade:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Projeto02.Entities

{

public class Produto

{

//prop + 2x[tab]

public int IdProduto { get; set; }

public string Nome { get; set; }

public decimal Preco { get; set; }

public int Quantidade { get; set; }

//ctor + 2x[tab]

public Produto()

{

//construtor default

}

//Sobrecarga de método construtor (Overloading)

public Produto(int idProduto, string nome, decimal preco, int quantidade)

{

IdProduto = idProduto;

Nome = nome;

Preco = preco;

Quantidade = quantidade;

}

//sobrescrita do método ToString()

public override string ToString()

{

return $"Id: {IdProduto}, Nome: {Nome},

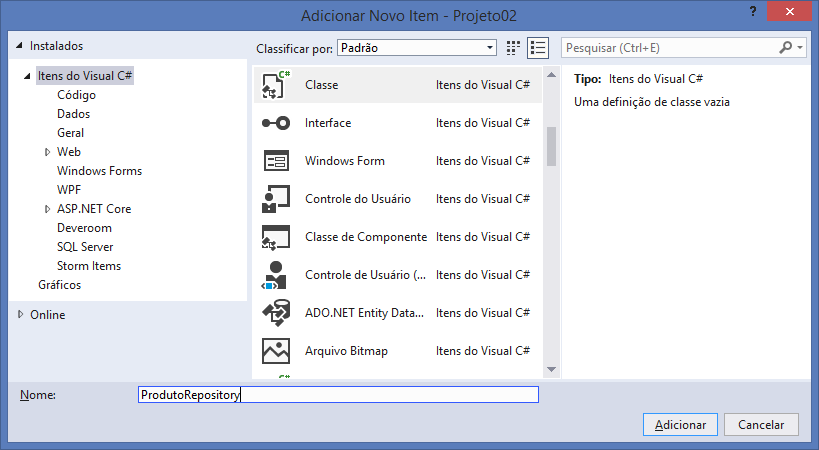
Preço: {Preco}, Qtd: {Quantidade}";

}

}

}

**Criando uma classe para exportar os dados   
do Produto para um arquivo XML:**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto02.Entities; //importando

using System.Xml; //importando

namespace Projeto02.Repositories

{

public class ProdutoRepository

{

public void ExportarParaXml(Produto produto)

{

//abrir um arquivo XML..

using (XmlWriter xmlWriter = XmlWriter

.Create("c:\\temp\\produto.xml"))

{

//abrir o documento xml..

xmlWriter.WriteStartDocument();

xmlWriter.WriteStartElement("PRODUTO"); //abrindo a TAG principal

xmlWriter.WriteElementString("IDPRODUTO",

produto.IdProduto.ToString());

xmlWriter.WriteElementString("NOME",

produto.Nome);

xmlWriter.WriteElementString("PRECO",

produto.Preco.ToString());

xmlWriter.WriteElementString("QUANTIDADE",

produto.Quantidade.ToString());

xmlWriter.WriteEndElement(); //fechando a TAG principal

}

}

}

}

**Executando e testando:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Projeto02.Entities; //importando..

using Projeto02.Repositories; //importando..

namespace Projeto02

{

class Program

{

static void Main(string[] args)

{

Produto produto = new Produto(1, "Mouse", 25.0m, 10);

try

{

ProdutoRepository repository = new ProdutoRepository();

repository.ExportarParaXml(produto);

Console.WriteLine("\nXML gerado com sucesso!");

}

catch(Exception e)

{

Console.WriteLine("Erro: " + e.Message);

}

Console.ReadKey();

}

}

}

**Executando:**

