

Curso básico de C#

Thiago Paiva Medeiros

Introdução

O que é

- Linguagem de programação orientada a objetos;
- Multiparadigma;
- Tipagem forte;
- Desenvolvida pela Microsoft;
- Versão atual : 7.0. (18/07/2018)

Primeiro programa

```
using System;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Declaração de variáveis

Declaração de variáveis

- Há vários tipos de dados em C#. Cada variável usa um tipo de dado.
- `int` inteiro;
- `char` character;
- `double` d;

Tipos de dados primitivos

Nome Abreviado	Classe .NET	Tipo	Bits	Valores
byte	Byte	Inteiro Não Assinado	8	0 até 255
sbyte	SByte	Signed inteiro	8	-128 até 127
int	Int32	Signed inteiro	32	-2.147.483.648 até 2.147.483.647
uint	UInt32	Inteiro Não Assinado	32	0 até 4294967295
short	Int16	Signed inteiro	16	-32.768 até 32.767

Tipos de dados primitivos

Nome Abreviado	Classe .NET	Type (Tipo)	Bits	Range (BITS)
ushort	UInt16	Inteiro Não Assinado	16	0 até 65535
long	Int64	Signed inteiro	64	- 922337203685477508 to até9223372036854775 07
ulong	UInt64	Inteiro Não Assinado	64	0 até 184467440737095516 15
float	Single	Ponto flutuante	32	-3.402823E38 até 3.402823E38
double	Double	Ponto flutuante Precisão dupla	64	1.79769313486232e30 8 até1.79769313486232 e308

Tipos de dados primitivos

Nome Abreviado	Classe .NET	Tipo	Bits	Valores
char	Char	Único caracter Unicode	16	Caracteres
bool	Boolean	Tipo Lógico	8	true ou false
object	Object	Base de todos os outros tipos		
string	String	Uma sequência de caracteres		
decimal	Decimal	Números decimais	128	$10e-28 \times \pm 1.0$ até $10e28 \times \pm 7.9$

Captura de dados

Captura de dados

- A captura de dados é feita através do comando `Console.ReadLine()`;

```
static void Main(string[] args)
{
    string s;

    Console.WriteLine("Escreva a frase \n");
    s = Console.ReadLine();

    Console.WriteLine(" A frase escrita foi -> " + s + ".\n");
}
```

Captura de dados

- O comando `Console.ReadLine()` retorna uma string. Para outros tipos de dados, será necessário realizar conversões.

```
static void Main(string[] args)
{
    int i;

    Console.WriteLine("Escreva o número \n");
    i = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine(" O número escrito foi -> " + i + ".\n");
}
```

```
static void Main(string[] args)
{
    double d;

    Console.WriteLine("Escreva o número \n");
    d = Convert.ToDouble(Console.ReadLine());

    Console.WriteLine(" O número escrito foi -> " + d + ".\n");
}
```

Exercício - Calculadora

Exercício - Calculadora

- Implementar uma calculadora que some, subtraia, multiplique e divida o número 1 e o número 2.

Comandos condicionais

Condicional if / else if / else

- Serve para executar blocos de códigos em determinadas condições.

```
class Program
{
    static void Main(string[] args)
    {
        int numero = 2;

        if (numero == 2)
            Console.WriteLine("O número é 2");

        else
            Console.WriteLine("O número não é 2");
    }
}
```


Condicional if / else if / else

```
class Program
{
    static void Main(string[] args)
    {
        int numero = 2;

        if (numero == 1)
            Console.WriteLine("O número é 1");

        else if (numero == 2)
            Console.WriteLine("O número é 2");

        else
            Console.WriteLine("O número não é 2");
    }
}
```

Exercício – Dias da semana

Exercício - Dias da semana

- Fazer um programa que exiba os dias da semana de acordo com a escolha do usuário.

Exercício – Mega Sena

Exercício – Mega Sena

- Mensalmente ocorre o sorteio da Mega Sena. São 6 dezenas sorteadas e quem acertar esses 6 números ganha um prêmio em dinheiro. Faça um programa que dado 6 números (predefinidos e digitados pelo usuário) mostre se o usuário ganhou ou não.

Condicional switch case

Condicional switch case

- Execução de um bloco de código a partir da escolha de uma opção.

```
static void Main(string[] args)
{
    int opcao;
    Console.WriteLine("Escolha a opção \n");
    opcao = Convert.ToInt32(Console.ReadLine());

    switch (opcao)
    {
        case 1:
            Console.WriteLine("Você escolheu 1 \n");
            break;

        case 2:
            Console.WriteLine("Você escolheu 2 \n");
            break;

        case 3:
            Console.WriteLine("Você escolheu 3 \n");
            break;

        default:
            Console.WriteLine("Você escolheu uma opção inválida \n");
            break;
    }
}
```

Exercício – Conversão de temperatura

Exercício – Conversão de temperatura

- Fazer um programa que converta temperaturas de acordo com a escolha do usuário.
- Fahrenheit para Célsius ($C = (F - 32) / 1,8$)
- Célsius para Fahrenheit ($F = 1.8C + 32$)
- Kelvin para Célsius ($C = 273 - K$)
- Celsius para Kelvin ($K = C + 273$)

Exercício – Conversão de metros

Exercício - Conversão de metros

- Fazer um programa para converter metros para quilômetro ou milímetro.
- Metro para quilômetro
- $\text{Metro} / 1000$
- Metro para milímetro
- $\text{Metro} \times 1000$

Estrutura de repetição while

Estrutura de repetição while

- Repete a execução de um bloco de código até que determinada condição seja satisfeita.

```
static void Main(string[] args)
{
    int quantidade = 10;

    while (quantidade > 0 )
    {
        Console.WriteLine("O valor atual é " + quantidade);
        quantidade = quantidade - 1;
    }
}
```

Exercício - Fatorial

Exercício - Fatorial

- Faça um programa para calcular o fatorial de um número informado pelo usuário.
- Fatorial de 5
- $5 \times 4 \times 3 \times 2 \times 1$

Estrutura de repetição do while

Estrutura de repetição do while

- A estrutura do while executa um bloco de código uma vez;
- Baseado em uma condição, define-se se esse bloco será executado novamente.

```
static void Main(string[] args)
{
    char escolha = 's';
    int numero = 0;
    do
    {
        Console.WriteLine("O valor de número é {0}", numero);
        numero++;
        Console.WriteLine("Continuar incrementando número ?");
        escolha = char.Parse(Console.ReadLine());
    } while (escolha == 's');
}
```

Exercício - Média

Exercício - Média

- Calcular a média de números à medida que o usuário os informa.

Estrutura de repetição for

Estrutura de repetição for

- Repete a execução de um bloco de código até que determinada condição seja satisfeita.

```
static void Main(string[] args)
{
    int quantidade;

    for (quantidade = 0; quantidade <= 10; quantidade++)
    {
        Console.WriteLine("O valor atual é " + quantidade);
    }
}
```

Exercício – Intervalo de valores

Exercício - Intervalo de valores

- Fazer um programa em que dado 2 números, imprima os números entre eles.
- 2 e 6
- 2 3 4 5 6



Try catch finally

Try catch finally

- O bloco de código é executado no try;
- Caso haja alguma exceção (divisão por 0, por exemplo), essa exceção é tratada no catch;
- O finally é executado após de qualquer maneira ao final. Finally é muito usado para desconectar o banco de dados após todas as tarefas terem sido realizadas.

Try catch finally

```
static void Main(string[] args)
{
    char character;

    try
    {
        Console.WriteLine("Digite algo \n");
        character = Convert.ToChar(Console.ReadLine());
    }
    catch (Exception ex)
    {
        Console.WriteLine("A exceção foi : " + ex.Message.ToString());
    }
    finally
    {
        Console.WriteLine("\nFim\n");
    }
}
```

Funções

Funções (ou métodos)

- Bloco de código que pode conter várias instruções;
- Esse bloco será executado apenas quando a função correspondente for chamada;
- Funções do tipo void não possuem retorno.

Funções (ou métodos)

```
static void Main(string[] args)
{
    Console.WriteLine("A soma de 16 + 30 é \n" + soma(16, 30));
    Finalizar();
}

static private int soma (int a, int b)
{
    return a + b;
}

static private void Finalizar()
{
    Console.WriteLine("Programa encerrado. \n");
}
```

Exercício - Calculadora

Exercício - Calculadora

- Fazer uma calculadora que some, subtraia, multiplique e divida dois números. Essas operações devem ser funções.

Potência e raiz

Potência e raiz

- Para realizar a potenciação de números, basta utilizar o comando `Math.Pow (n, p);`
`Math.Pow(2, 6)`
- Para realizar a raiz quadrada, basta utilizar `Math.Sqrt(n).`
`Math.Sqrt(9)`

Exercício – Distância entre dois pontos

Exercício – Distância entre 2 pontos

- Após o usuário informar os valores de x_1 , x_2 , y_1 e y_2 , calcule a distância entre os 2 pontos.
- $D = \text{Raiz } ((x_1 - x_2)^2 + (y_1 - y_2)^2)$

Personalização do console

Personalização do console

- Se deseja mudar o título do console, use o comando `Console.Title`

```
Console.Title = "Título";
```

- Para destacar determinadas mensagens, uma das opções é deixá-la com a cor diferente das demais;
- Para determinada tarefa, basta utilizar o `Console.ForegroundColor`;

```
Console.ForegroundColor = Console.DarkMagenta;
```

Personalização do console

- Se deseja configurar o tamanho e largura da janela, pode usar o `Console.SetWindowSize`

`Console.SetWindowSize(largura, altura)`

- Mas antes use os comandos `Console.LargestWindowWidth` e `Console.LargestWindowHeight` para saber o valores máximos permitidos.

```
static void Main(string[] args)
{
    Console.Title = "Personalizando o console";
    Console.ForegroundColor = ConsoleColor.DarkMagenta;
    int largura = Console.LargestWindowWidth;
    int altura = Console.LargestWindowHeight;
    Console.WriteLine("Largura máxima -> {0} --- Altura máxima -> {1} \n", largura, altura);
    Console.SetWindowSize(Console.LargestWindowWidth / 2, Console.LargestWindowHeight / 2);
}
```

Gerando números aleatórios

Gerando números aleatórios

- Para gerar números aleatórios, basta utilizar a classe Random e então o método next

```
Random rnd = new Random()
```

```
Int numero = rnd.next();
```


Exercício – Par ou impar

Exercício – Par ou Impar

- Faça o jogo “par ou impar”;
- O usuário deve digitar um número. Um número aleatório deve ser gerado pelo computador. Então verifique quem foi o vencedor;
- Após o fim do jogo, pergunte ao usuário se o mesmo deseja jogar novamente;
- Após o fim do jogo, deve ser informado quantas vezes o jogador e o computador venceram.

Vetores

Vetores

- Conjunto de dados de um mesmo tipo;
- Cada dado é acessado através de um índice;
- Também conhecido como Array.

Vetores

```
static void Main(string[] args)
{
    string[] nomes = new string[5];

    nomes[0] = "Ana";
    nomes[1] = "Camila";
    nomes[2] = "Daniela";
    nomes[3] = "Helena";
    nomes[4] = "Leila";

    Console.WriteLine("Primeiro nome : " + nomes[0]);
    Console.WriteLine("Segundo nome : " + nomes[1]);
    Console.WriteLine("Terceiro nome : " + nomes[2]);
    Console.WriteLine("Quarto nome : " + nomes[3]);
    Console.WriteLine("Quinto nome : " + nomes[4]);
}
```

- Se tentarmos acessar uma posição inválida (10, por exemplo), a exceção `IndexOutOfRangeException` será lançada.

Capturando dados em vetores

- É necessário utilizar uma estrutura de repetição.

```
static void Main(string[] args)
{
    int[] numeros = new int[5];
    int qtd;

    for (qtd = 0; qtd < numeros.Length; qtd++)
    {
        Console.WriteLine("Digite o número " + qtd + " :");
        numeros[qtd] = Convert.ToInt32(Console.ReadLine());
    }

    Console.WriteLine("Números digitados -> \n");
    for (qtd = 0; qtd < numeros.Length; qtd++)
    {
        Console.WriteLine(numeros[qtd] + "\t");
    }
}
```

Exercício - Média

Exercício - Média

- Dado 10 números, calcular a média deles.
- Números -> 4 8 6 12
- Média -> $(4 + 8 + 6 + 12) / 4$

Exercício – Palavra palíndroma

Exercício – Palavra Palíndroma

- Dado uma palavra, verificar se ela é uma palavra palíndroma.

Matrizes

Matrizes

- Estrutura de dados multidimensional;
- É acessada pelo seu índice;
- Se tentarmos acessar uma posição inválida, a exceção `IndexOutOfRangeException` será lançada.

Matrizes

```
static void Main(string[] args)
{
    int[,] matriz = new int[3,3];
    int linha, coluna;

    for (linha = 0; linha < 3; linha++)
    {
        for (coluna = 0; coluna < 3; coluna++)
        {
            Console.WriteLine("Digite o número da coluna " + coluna + " linha " + linha);
            matriz[coluna, linha] = Convert.ToInt32(Console.ReadLine());
        }
    }

    Console.WriteLine("\nMatriz digitada \n");

    for (linha = 0; linha < 3; linha++)
    {
        for (coluna = 0; coluna < 3; coluna++)
        {
            Console.WriteLine(matriz[coluna, linha] + " ");
        }
        Console.WriteLine("\n");
    }
}
```

Exercício – Multiplicação de Matrizes

Exercício - Multiplicação de Matrizes

➤ Fazer um programa para multiplicar 2 matrizes 2x2.

➤
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 4 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1 \times 4 + 2 \times 6 & 1 \times 5 + 2 \times 7 \\ 3 \times 4 + 4 \times 6 & 3 \times 5 + 4 \times 7 \end{bmatrix}$$

Arquivos

Arquivos

- Arquivos são um importante meio de se armazenar informações de forma permanente;
- Suas informações podem permanecer gravadas mesmo após o término do programa.

Criando um arquivo

```
string caminhoArquivo = @"C:\caminhodoArquivo\Arquivo.txt";  
File.Create(caminhoArquivo);
```

Inserindo dados em um arquivo

```
using (var streamWriter = new StreamWriter(caminhoArquivo, true))  
{  
    streamWriter.WriteLine(texto);  
}
```

Lendo um arquivo

```
using (TextReader textReader = new StreamReader(caminhoArquivo))  
{  
    Console.WriteLine(textReader.ReadToEnd());  
}
```

Excluindo um arquivo

```
File.Delete(caminhoArquivo);
```

Exercício – No Fly List

Exercício – NoFly List

- Determinadas companhias aéreas possuem a conhecida No Fly List, uma lista que possui nomes de pessoas que não podem ou possuem algum tipo de suspeita ao voar;
- Faça um programa para ler nomes em um arquivo e comparar com um nome digitado pelo usuário. No final do programa, imprima se essa pessoa é suspeita ou não.

Classes

Classes

- Abstração de algo do mundo real que possui conjunto de características;
- Pode possuir atributos, métodos e construtor(es);
- Construtores são métodos chamados assim que a classes é instanciada. Geralmente eles são responsáveis pela alocação de recursos necessários para o correto funcionamento da classe.

Classes

- Para instanciar essa classe, basta usar `Pessoa p = new Pessoa();`

```
class Pessoa
{
    string Nome;
    string CPF;
    int Idade;

    void andar() { }
    void comer() { }
    Pessoa()
    {
        Nome = "Thiago";
        CPF = "111.222.333-44";
        Idade = 29;
    }
}
```

Encapsulamento

Encapsulamento

- Permite separar o programa em partes;
- Possibilita que os detalhes internos de funcionamento dos métodos de uma classe permaneçam ocultos para outros objetos;
- Permite controlar o acesso a atributos e métodos de uma classe.

Encapsulamento

- Encapsulamento é implementado através dos modificadores de acesso
- `public` -> atributos e métodos podem ser acessados em qualquer parte do programa. É aconselhável deixar os métodos, classes e construtores como `public`.
- `private` -> atributos e métodos podem ser acessados apenas pela própria classe. É aconselhável que os atributos de uma classe sejam `private`;
- `protected` -> atributos e métodos podem ser acessados apenas pela própria classe e seus derivados

Encapsulamento

- Como acessar e inserir valores em atributos privados em outras partes do programa ?
- Utilizando getters e setters !

Encapsulamento

```
public class Pessoa
{
    private string _Nome;
    private string _CPF;
    private int _Idade;

    public string Nome { get => _Nome; set => _Nome = value; }
    public string CPF { get => _CPF; set => _CPF = value; }
    public int Idade { get => _Idade; set => _Idade = value; }

    public void andar() { }
    public void comer() { }

    public Pessoa()
    {
        Nome = "Thiago";
        CPF = "111.222.333-44";
        Idade = 29;
    }
}
```

The background features abstract, overlapping geometric shapes in various shades of purple and magenta, primarily concentrated on the right side of the frame. The central area is a plain, light gray.

Herança

Herança

- Permite herdar atributos e métodos já definidos em outras classes;
- Possibilita reutilização de código;
- A classe pai é conhecida como classe base;
- A classe filha (aquela que herda atributos e métodos) é conhecida como classe derivada.

Herança

```
public class Pessoa
{
    private string _Nome;
    private int _Idade;
    private double _Peso;
    private double _Altura;

    public Pessoa() { }

    public string RetornaDados()
    {
        string dados = String.Format("{0}, {1} anos, pesando {2} e {3} de altura",
            _Nome, _Idade, _Peso, _Altura);
        return dados;
    }
}
```

```
public class PessoaFísica : Pessoa
{
    private string CPF;
}
```

```
public class PessoaJurídica : Pessoa
{
    private string CNPJ;
}
```

Polimorfismo

Polimorfismo

- Permite que um método tenha vários comportamentos.
- Polimorfismo estático ou sobrecarga : O método é implementado várias vezes na classe. A assinatura do métodos sobrecarregados que determinará qual método será executado.
- Polimorfismo Dinâmico ou sobreposição : Ocorre na herança quando o método na classe filha sobrepõe o método na classe pai. O método a ser chamado será escolhida em tempo de execução.

Polimorfismo estático ou sobrecarga

```
public class Metodos
{
    public void CadastrarPessoa(string Nome) { }

    public void CadastrarPessoa(string Nome, int Idade) { }

    public void CadastrarPessoa(string Nome, int Idade, double Peso) { }

    public void CadastrarPessoa(string Nome, int Idade, double Peso, double Altura) { }
}
```

Polimorfismo dinâmico ou sobreposição

```
public class Pessoa
{
    private string Nome;
    private string CPF;
    private int Idade;
    private double Peso;
    private double Altura;

    public void Cadastrar() { }
}
```

```
public class PessoaFísica : Pessoa
{
    private string CPF;

    public new void Cadastrar() { }
}
```

```
public class PessoaJurídica : Pessoa
{
    private string CNPJ;

    public new void Cadastrar() { }
}
```

Interfaces

Interfaces

- Classe que possui apenas assinaturas de métodos, propriedades, eventos e indexadores;
- Outra classe deve implementar os métodos descritos na interface;
- Interfaces possuem um 'I' antes do seu nome. IPessoa e IProduto são exemplos de interfaces;
- Para implementá-la, basta utilizar Classe : NomeInterface

Interfaces

```
public interface IPessoa
{
    void PreencherDados();
    int PegarIdade();
    double PegarPeso();
    string PegarCPF();
}
```

```
public class Pessoa : IPessoa
{
    public string PegarCPF()
    { /**/ }

    public int PegarIdade()
    { /**/ }

    public double PegarPeso()
    { /**/ }

    public void PreencherDados()
    { /**/ }
}
```

Exercício - Cadastro de Pessoas

Exercício – Cadastro de Pessoas

- Um cliente deseja ter uma forma de armazenar informações de seus clientes e fornecedores. Faça um programa para o cadastro de pessoas.
- O banco de dados deve ser simulado através de um arquivo.
- O programa deve :
 - Listar todas as pessoas;
 - Verificar se uma pessoa já está cadastrada;
 - Permitir o cadastro de pessoas físicas e jurídicas;
 - Permitir a exclusão do banco de dados.

Introdução à Banco de dados

Introdução Banco de Dados

- Coleções de dados interligados entre si e organizados para fornecer informações;
- Possibilitam o armazenamento de informações;
- Utilizando um banco de dados relacional, as informações são armazenadas em forma de tabelas.

Introdução Banco de Dados

	peessoaid [PK] integer	nome character varying (80)	cpf character varying (20)	idade integer	peso integer
1	1	Thiago	166.888.777-55	30	74
2	2	Ana	177.999.844-32	28	60
3	3	Beatriz	066.777.963-88	23	57
4	4	Helena	555.888.444-55	24	66
5	5	Sabrina	122.333.777-55	18	55

Introdução Banco de Dados

- Usa-se o `CREATE DATABASE NomeBancoDados` para a criação do banco de dados;
- `CREATE DATABASE Produtos`

Introdução Banco de Dados

- Usa-se o `CREATE TABLE NomeTabela` para se criar a tabela;
- Toda tabela deve ter uma chave primária (`PRIMARY KEY`). Através dessa chave primária é que será possível diferenciar os registros;
- `CREATE TABLE Pessoa (Pessoald serial PRIMARY KEY, Nome VARCHAR(50) NOT NULL);`

Introdução Banco de Dados

- Para a inserção de dados, usa-se o `INSERT INTO NomeTabela (Coluna1, Coluna2, ColunaN) VALUES (Valor1, Valor2, ValorN);`
- `INSERT INTO Pessoa (Nome, Idade) VALUES ('Thiago', 30);`

Introdução Banco de Dados

- Para seleccionar valores, utilizamos
- `SELECT Coluna1, Coluna2, ColunaN FROM NomeTabela WHERE (condição)`
- `SELECT Nome, CPF FROM Pessoa WHERE Idade >= 18`

Introdução Banco de Dados

- Para atualizar o valor de uma linha, utilizamos o
- `UPDATE NomeTable SET Coluna1 = Valor1, Coluna2 = Valor2, ColunaN = ValorN WHERE (condição)`
- `UPDATE Pessoa SET Nome = 'Camila', Idade = 28 WHERE CPF = '111.555.444-89'`

Introdução Banco de Dados

- Para excluir uma linha inteira, usamos `DELETE FROM NomeTabela WHERE (condição);`
- `DELETE FROM Pessoa WHERE Idade < 18`

Exercício – Banco de dados

Exercício - Banco de dados

- Uma empresa precisa de um banco de dados para armazenar seus produtos.
- Faça um script para :
 - Criar um banco de dados com o nome “Armazém”;
 - Criar uma tabela com nome “Produtos”. Ela deve possuir as colunas ProdutoId, Nome, Marca, Quantidade em estoque, Preço unitário, Peso, Setor;
 - Selecionar todos os produtos;
 - Inserir 10 registros na tabela;

Exercício – Banco de dados

- Após alguns dias, a empresa entrou em contato e informou o seguinte :
- Houve um erro na inserção do produto com ID 8, o mesmo deve ser atualizado para “Smartphone”, “XYZ”, 10, 700.00, 0.5, “Tecnologia”;
- O produto com ID 4 parou de ser vendido. Exclua-o do banco.

Introdução – Windows Forms

Introdução Windows Forms

- Tecnologia utilizada para construção de softwares para desktop;
- Permite conectar a vários bancos de dados;
- Possui vários elementos, tais como labels, textboxes, grids e etc

Introdução Windows Forms

Produtos armazenados

Dados do projeto

Identificador

Nome: Creme Dental Marca: Colgate

Quantidade: 50 Preço: 2,99

Peso: 0,1 Setor: Enxaguantes

Salvar

	ProdutoId	Nome	Marca	Quantidade	Preço	Peso	Setor
	1	Tenis	Adidas	10	70,99	1,2	Calçados
	2	Bone	QuikSilver	20	10,99	0,2	Bonés
	3	Iphone	Apple	20	1000,99	0,7	Tecnologia
	5	Monitor	Samsung	9	349,99	1,2	Tecnologia
	6	Pneu	Michelan	20	889,99	10,2	Automotivo
	7	Shampoo	Fructis	50	8,99	0,1	Comésticos
	8	SmartPhone	XYZ	10	700	0,5	Tecnologia
	9	Panela	Tramontina	20	39,99	0,5	Acessórios
▶	10	Creme De...	Colgate	50	2,99	0,1	Enxaguant...
	11	Shampoo	Fructis	9	899	1	Cosméticos
	12	Shampoo	Loreal	9	899	1	Cosméticos
	13	Tablet	Samsung	9	89999	1	Tecnologia
	14	Mouse	Razer	9	9999	1	Tecnologia
	15	Mouse	Logitech	9	999	1	Tecnologia

Atualizar Excluir

Introdução – ASP NET CORE MVC 2