



Colégio
Pedro II

PROGRAMAÇÃO O.O. (C#)



Aula 25: Aplicações Multiformulário e Enumerações
Professor: João Luiz Lagôas



Aplicativo do Windows Forms (.NET Framework)



- Para desenvolver um aplicativo que contenha formulários iniciamos criando um projeto de *Windows Forms* no *Visual Studio*.
- Automaticamente uma classe `Form1.cs` é criada e nós temos acesso a ela através do **Gerenciador de Soluções**.
- Podemos então editar essa classe **adicionando atributos** (labels, buttons, textboxes, progressbars, comboboxes, etc).
- Podemos também **adicionar métodos** à classe `Form1` que respondem a determinados eventos (clique do button, mudança de valor na textbox, carregamento do form, etc).

Aplicativo do Windows Forms (.NET Framework)



MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções

Solução 'MeuProjeto' (1 projeto)

- MeuProjeto
 - Properties
 - Referências
 - App.config
 - Form1.cs
 - Form1.Designer.cs
 - Form1.resx
 - Program.cs

Program.cs* Form1.cs [Design]*

Exemplo de Formulário

Carregar Texto Limpar Texto

Caixa de Ferramentas

- DirectorySearcher
- DomainUpDown
- ErrorProvider
- EventLog
- FileSystemWatcher
- FlowLayoutPanel
- FolderBrowserDialog
- FontDialog
- GroupBox
- HelpProvider
- HScrollBar
- ImageList
- Label
- LinkLabel
- ListBox
- ListView
- MaskedTextBox
- MenuStrip
- MessageQueue
- MonthCalendar
- NotifyIcon

Saída

Operações de Ferramentas de Dados Saída

Pronto

Adicionar ao Controle do Código-Fonte

Adicionamos alguns atributos nesse formulário.

Classe automaticamente criada.

Aplicativo do Windows Forms (.NET Framework)



MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

João Luiz Lagôas

Debug Any CPU Iniciar

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções

Solução 'MeuProjeto' (1 projeto)

- MeuProjeto
 - Properties
 - Referências
 - App.config
 - Form1.cs
 - Form1.Designer.cs
 - Form1.resx
 - Program.cs

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MeuProjeto
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Código para fazer alguma coisa em resposta ao clique do button
        }
    }
}
```

Adicionamos métodos para responder a eventos.

107 %

Saída

Operações de Ferramentas de Dados Saída

Pronto Li 22 Col 62 Car 62 INS Adicionar ao Controle d 2 novas notificações

Aplicativo do Windows Forms (.NET Framework)



O que torna todas essas classes diferentes?
Por que essas classes são semelhantes?

Aplicativo do Windows Forms (.NET Framework)



- Todos os formulários que vimos no slide anterior são diferentes porque cada um tem um conjunto de atributos e métodos próprios!
- Todos os formulários que vimos no slide anterior são semelhantes porque... ora, porque todos são formulários!



Aplicativo do Windows Forms (.NET Framework)



MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

João Luiz Lagôas

Debug Any CPU Iniciar

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções

Solução 'MeuProjeto' (1 projeto)

- MeuProjeto
 - Properties
 - Referências
 - App.config
 - Form1.cs
 - Form1.Designer.cs
 - Form1.resx
 - Program.cs

Form1.cs* Program.cs* Form1.cs [Design]*

MeuProjeto

```
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace MeuProjeto
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void button1_Click(object sender, EventArgs e)
21         {
22             // Código para fazer alguma coisa e
23         }
24     }
25 }
26
```

button1_Click(object sender, EventArgs e)

Qualquer classe de formulário que criamos, seja ela Form1.cs, Form2.cs, QuestaoForm.cs, MainForm.cs, etc, são filhas de uma classe predefinida no .NET! Uma classe chamada Form!

107 %

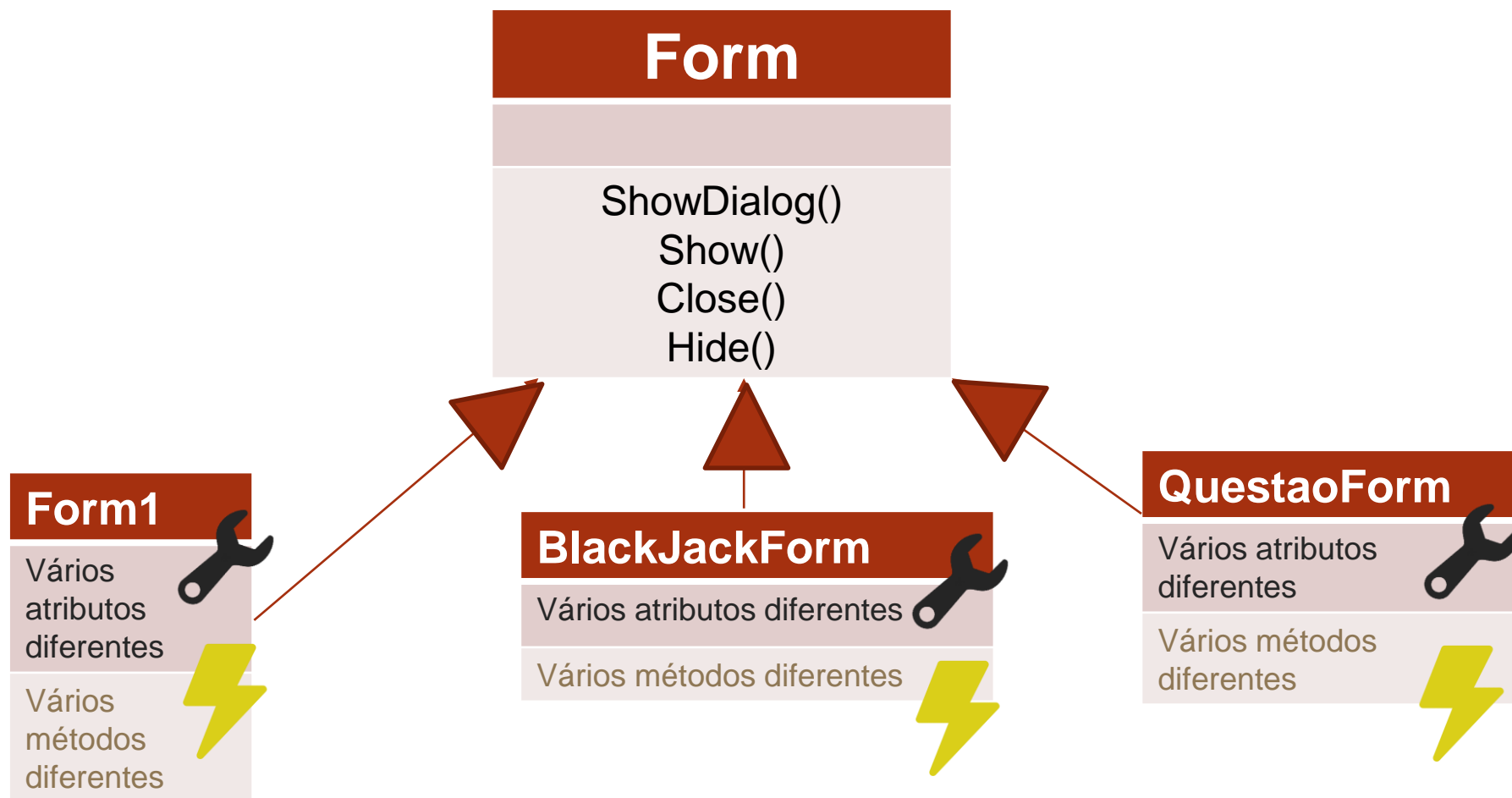
Saída

Operações de Ferramentas de Dados Saída

Pronto Li 22 Col 62 Car 62 INS

Adicionar ao Controle 2 novas notificações

Aplicativo do Windows Forms (.NET Framework)

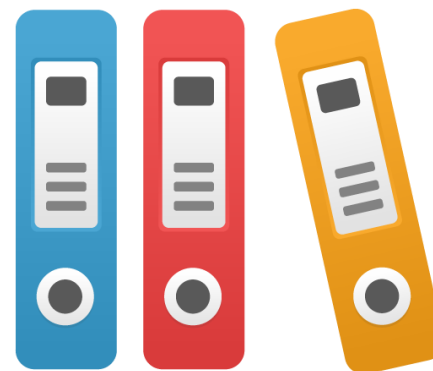


Aplicativo do Windows Forms (.NET Framework)



- Entendendo que todas as classes de formulário que criamos até agora são filhas da classe Form, então significa que IMPLICITAMENTE, essas classes herdaram vários atributos e métodos.
- Você pode conhecer todos os atributos/propriedades e métodos da classe Form na própria documentação oferecida pela Microsoft:

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms?view=netframework-4.7.2>



Atributos e Métodos da Classe Form



Form

ShowDialog()
Show()
Close()
Hide()

Métodos	Descrição
ShowDialog()	Exibe o formulário em modo de diálogo (isso significa que formulários que estão em background não podem ser ativados).
Show()	Exibe o formulário.
Close()	Fecha o formulário
Hide()	Esconde o formulário. <u>Nota:</u> o formulário não é liberado da memória! Ele apenas está “escondido” e pode ser aberto através do método Show() a qualquer momento.

Criando mais de um Formulário

MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Formatar Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Form2.cs Form1.cs [Design]* Form2.cs [Design] Form1.cs* Program.cs

Propriedades

btnFecharAplicacao System.Windows.Forms.Button

Acessibilidade

AccessibleDescription	
AccessibleName	
AccessibleRole	Default

Aparência

BackColor	<input type="checkbox"/> Control
BackgroundImage	<input type="checkbox"/> (nenhum/a)
BackgroundImageLayout	Tile
Cursor	Default
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif; 8pt
ForeColor	<input type="checkbox"/> ControlText
Image	<input type="checkbox"/> (nenhum/a)
ImageAlign	MiddleCenter
ImageIndex	<input type="checkbox"/> (nenhum/a)
ImageKey	<input type="checkbox"/> (nenhum/a)
ImageList	(nenhum)
RightToLeft	No
Text	Fechar Aplicação
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackColor	True
UseWaitCursor	False

Text

O texto associado ao controle.

Form1

IstAlunos Campus

Inserir um Aluno

Fechar Aplicação

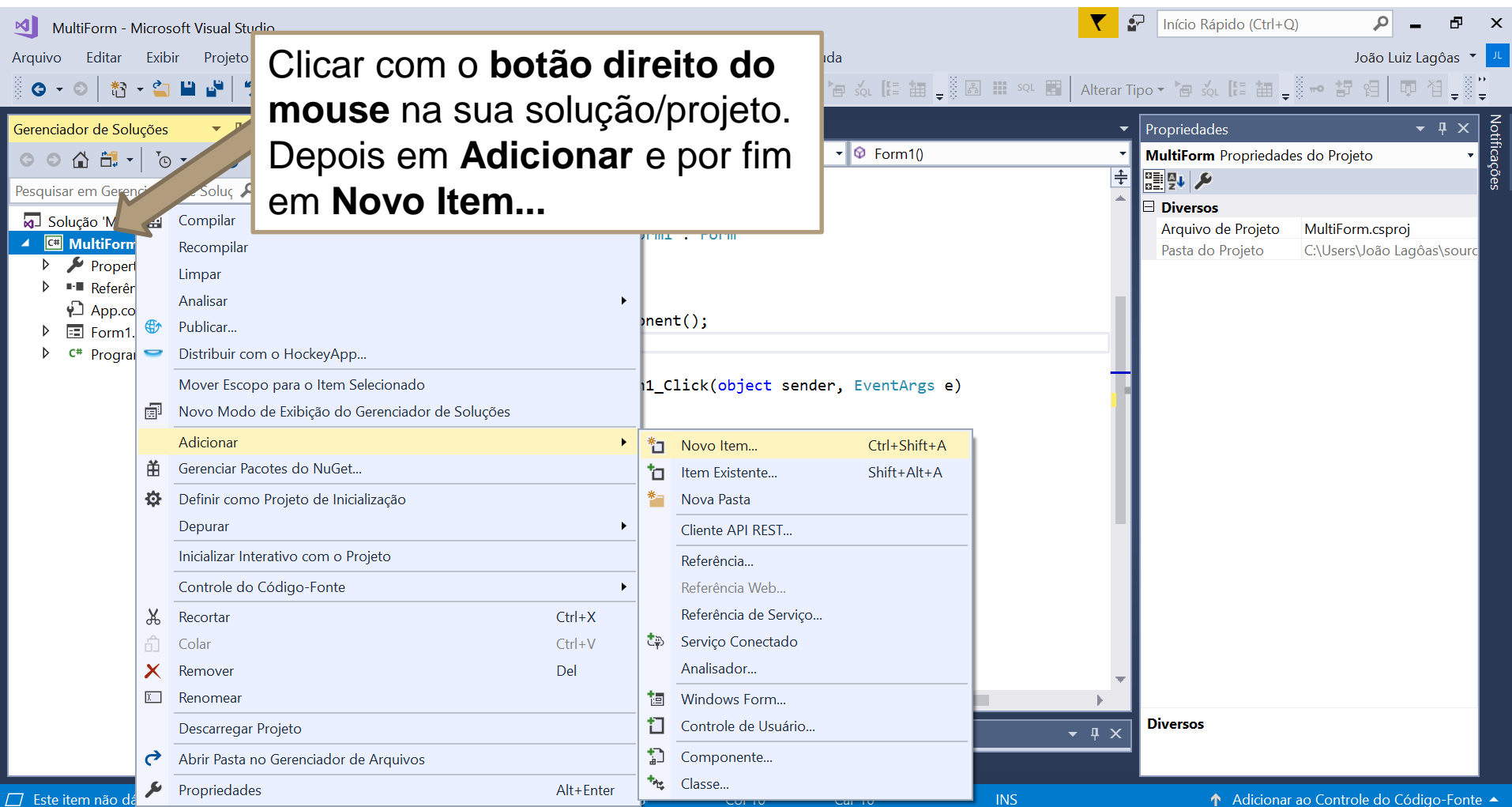
Saída

Operações de Ferramentas de Dados Saída

Crie o formulário da classe Form1 no Designer como você sempre fez e implemente seus métodos.

Criando mais de um Formulário

Clicar com o **botão direito do mouse** na sua solução/projeto. Depois em **Adicionar** e por fim em **Novo Item...**



The screenshot shows the Microsoft Visual Studio interface with a project named 'MultiForm'. The 'Gerenciador de Soluções' (Solution Explorer) on the left shows the project structure. A right-click context menu is open over the project, with the 'Adicionar' (Add) option selected. A sub-menu is displayed, showing 'Novo Item...' (New Item...) as the first option, which is highlighted. The 'Propriedades' (Properties) window on the right shows the project properties, including the project name 'MultiForm' and the project path 'C:\Users\João Lagôas\source\...'. The status bar at the bottom indicates 'Este item não dá' (This item does not give) and 'Adicionar ao Controle do Código-Fonte' (Add to Source Control).

MultiForm - Microsoft Visual Studio

Arquivo Editar Exibir Projeto

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções

Solução 'MultiForm'

MultiForm

Propriedades

MultiForm Propriedades do Projeto

Diversos

Arquivo de Projeto MultiForm.csproj

Pasta do Projeto C:\Users\João Lagôas\source\...

Novo Item... Ctrl+Shift+A

Item Existente... Shift+Alt+A

Nova Pasta

Cliente API REST...

Referência...

Referência Web...

Referência de Serviço...

Serviço Conectado

Analizador...

Windows Form...

Controle de Usuário...

Componente...

Classe...

Este item não dá

Adicionar ao Controle do Código-Fonte

Criando mais de um Formulário

MultiForm - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Início Rápido (Ctrl+Q)

João Luiz Lagôas

Adicionar Novo Item - MultiForm

Seleção Windows Form

Pesquisar em Gerenciador de Soluções

- Solução 'MultiForm'
- MultiForm
 - Propriedades
 - Referências
 - Aplicativo
 - Formulários
 - Programa

Itens do Visual C#

- Código
- Dados
- Geral
- Web
- Windows Forms
- WPF
- Asp.NET Core
 - SQL Server
 - Storm Items
- Gráficos
- Online

Itens do Visual C#

- Classe para U-SQL
- Classe
- Interface
- Windows Form**
- Controle do Usuário
- Classe de Componente
- Controle de Usuário (WPF)
- ADO.NET Entity Data Model
- Arquivo Bitmap
- Arquivo de Código
- Arquivo de Configuração de Aplicativo

Itens do Visual C#

Tipo: Itens do Visual C#

Um Windows Form em branco

Pesquisar (Ctrl+E)

Dê um nome para a sua classe que representará o novo formulário.

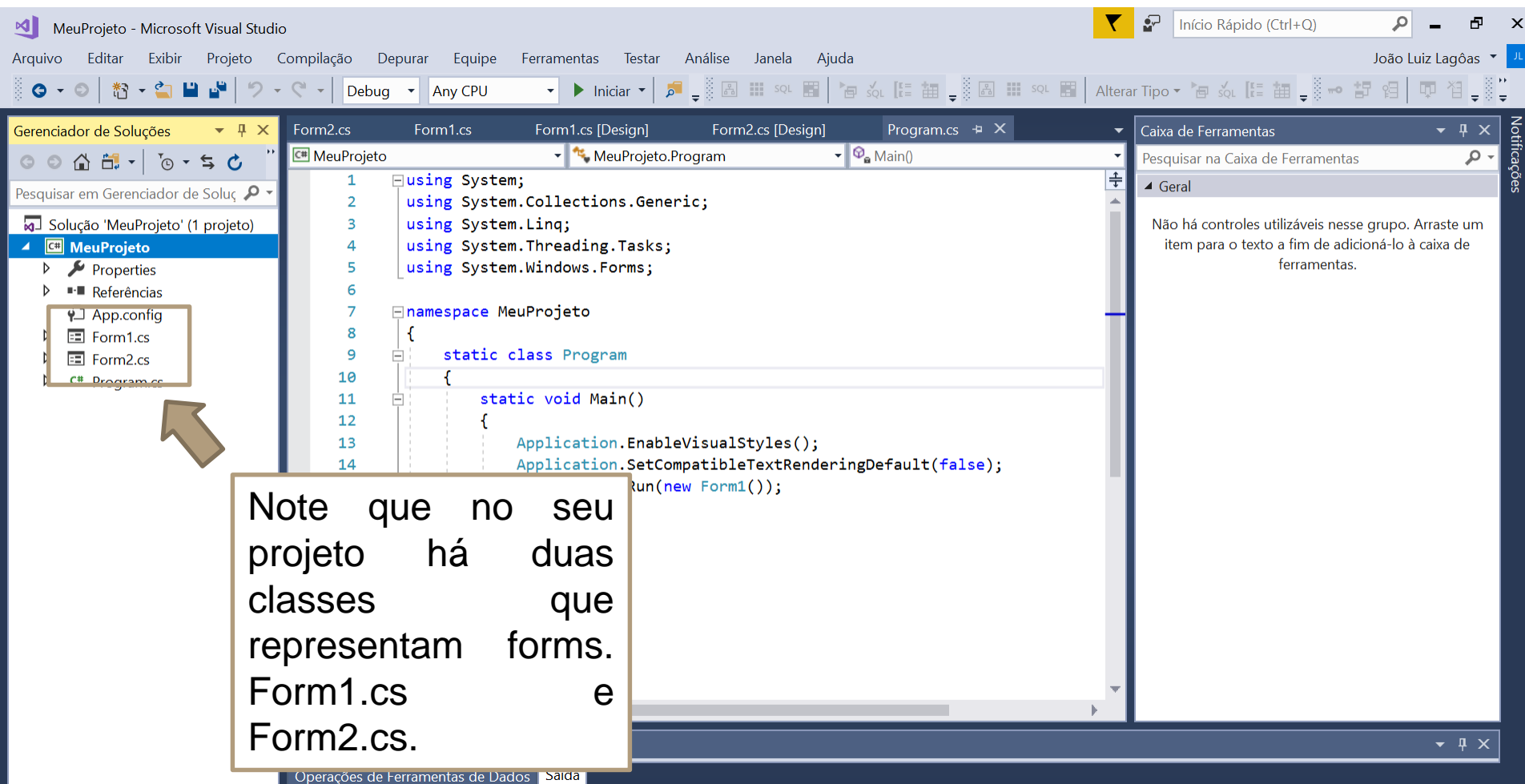
Nome: Form2.cs

Adicionar Cancelar

Este item não

Código-Fonte

Criando mais de um Formulário



MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Form2.cs Form1.cs Form1.cs [Design] Form2.cs [Design] Program.cs

MeuProjeto MeuProjeto.Program Main()

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using System.Windows.Forms;
6
7 namespace MeuProjeto
8 {
9     static class Program
10     {
11         static void Main()
12         {
13             Application.EnableVisualStyles();
14             Application.SetCompatibleTextRenderingDefault(false);
15             Run(new Form1());
16         }
17     }
18 }
```

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções

Solução 'MeuProjeto' (1 projeto)

MeuProjeto

- Properties
- Referências
- App.config
- Form1.cs
- Form2.cs
- Program.cs

Caixa de Ferramentas

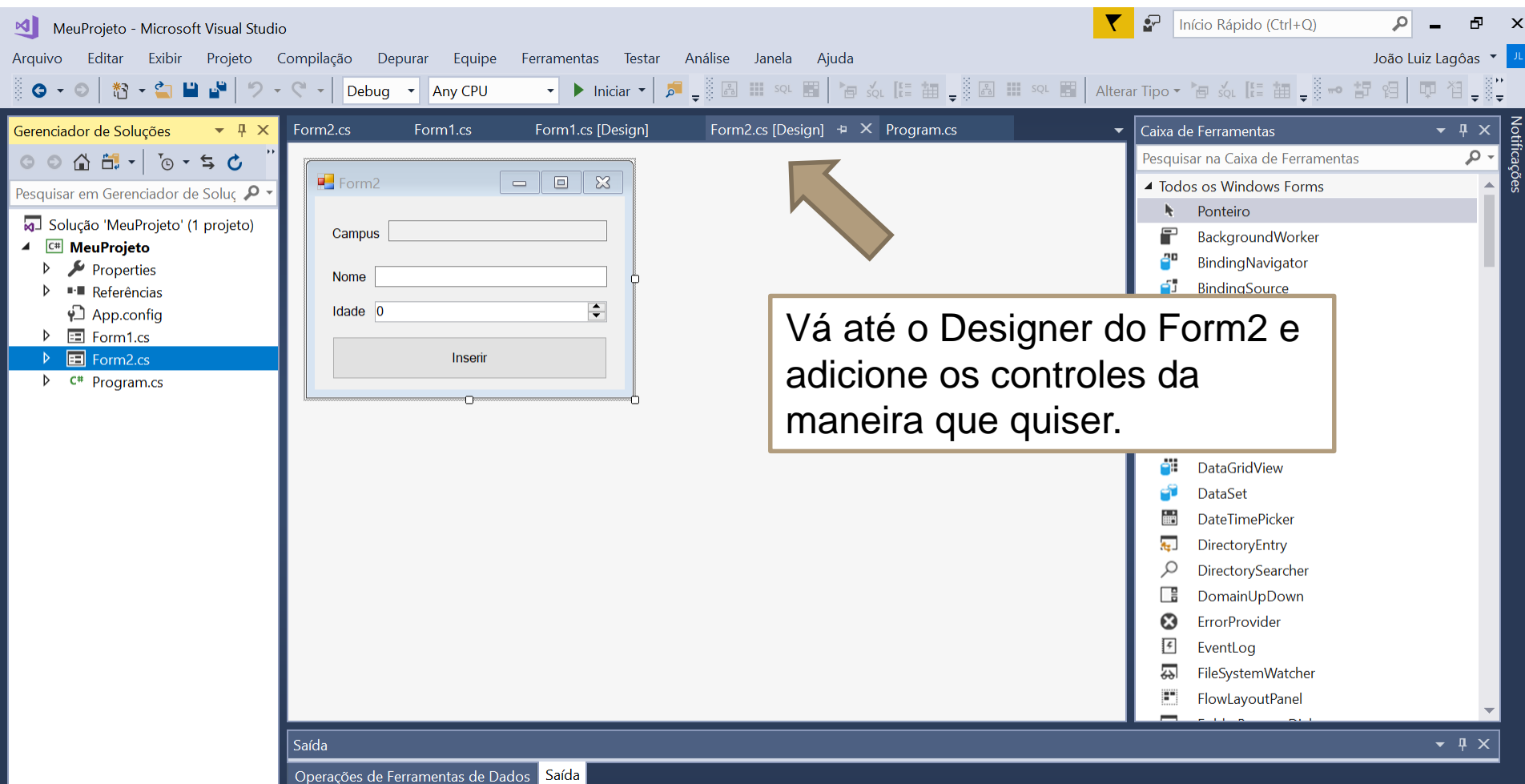
Pesquisar na Caixa de Ferramentas

Geral

Não há controles utilizáveis nesse grupo. Arraste um item para o texto a fim de adicioná-lo à caixa de ferramentas.

Note que no seu projeto há duas classes que representam forms. Form1.cs e Form2.cs.

Criando mais de um Formulário



MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Form2.cs Form1.cs Form1.cs [Design] Form2.cs [Design] Program.cs

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções

Solução 'MeuProjeto' (1 projeto)

- MeuProjeto
 - Properties
 - Referências
 - App.config
 - Form1.cs
 - Form2.cs
 - Program.cs

Form2

Campus

Nome

Idade

Inserir

Caixa de Ferramentas

Pesquisar na Caixa de Ferramentas

Todos os Windows Forms

- Ponteiro
- BackgroundWorker
- BindingNavigator
- BindingSource
- DataGridView
- DataSet
- DateTimePicker
- DirectoryEntry
- DirectorySearcher
- DomainUpDown
- ErrorProvider
- EventLog
- FileSystemWatcher
- FlowLayoutPanel

Vá até o Designer do Form2 e adicione os controles da maneira que quiser.

Saída

Operações de Ferramentas de Dados Saída

Criando mais de um Formulário



MeuProjeto - Microsoft Visual Studio

Arquivo Editar Exibir Projeto Compilação Depurar Equipe Ferramentas Testar Análise Janela Ajuda

Debug Any CPU Iniciar

Gerenciador de Soluções

Solução 'MeuProjeto' (1 projeto)

- MeuProjeto
 - Properties
 - Referências
 - App.config
 - Form1.cs
 - Form2.cs
 - Program.cs

Form2.cs Form1.cs Form1.cs [Design] Form2.cs [Design] Program.cs

MeuProjeto MeuProjeto.Form2 Form2(string campus)

```
10
11 namespace MeuProjeto
12 {
13     public partial class Form2 : Form
14     {
15         public Form2(string campus)
16         {
17             InitializeComponent();
18             txtCampus.Text = campus;
19         }
20
21         private void button1_Click(object sender, EventArgs e)
22         {
23             DialogResult = DialogResult.OK;
24             Close();
25         }
26
27         public string GetAlunoInfo()
28         {
29             string nome = txtNome.Text;
30             decimal idade = numIdade.Value;
31         }
32
33     }
```

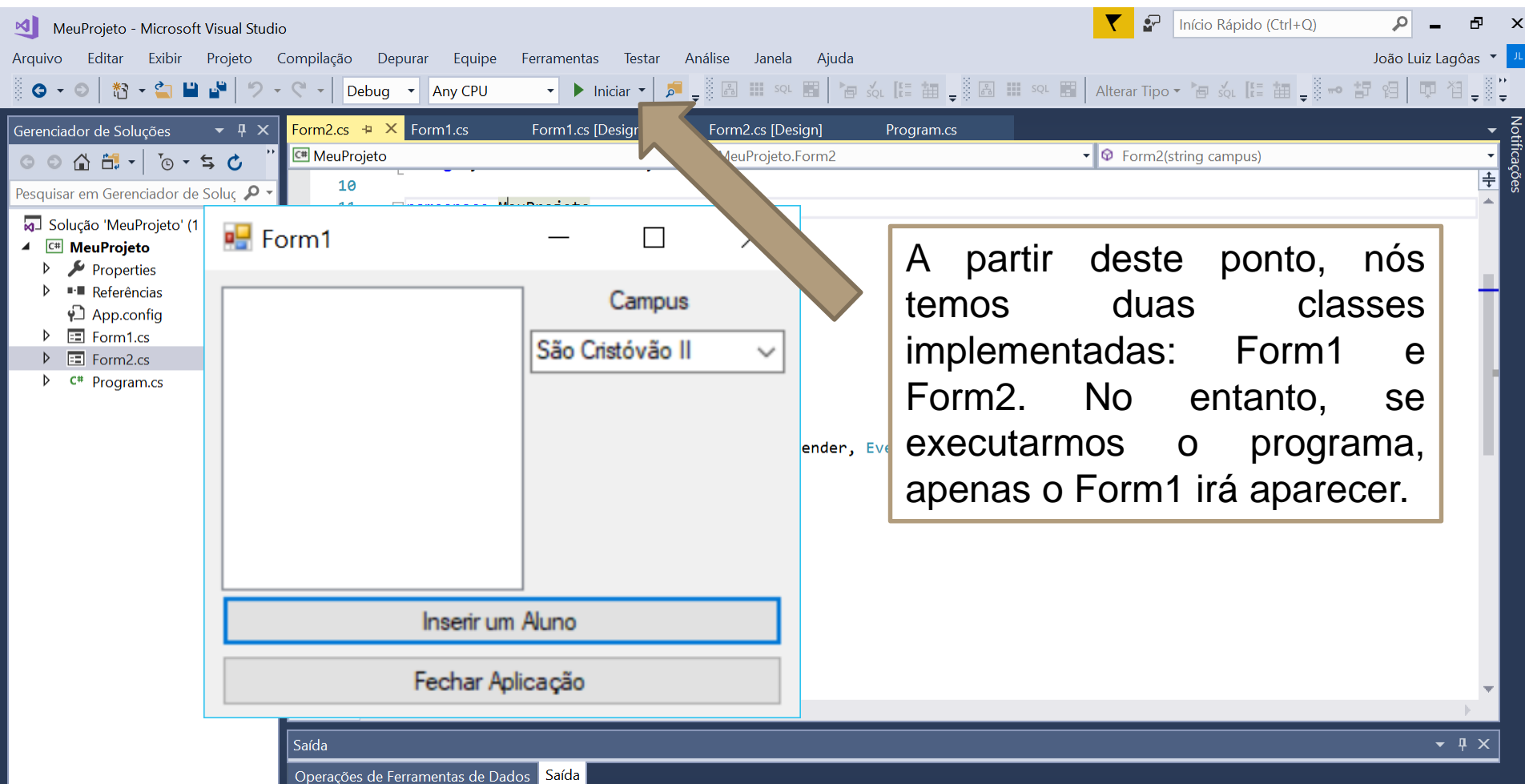
Obviamente implemente tudo o que for necessário no Form2: seus métodos e funcionalidades.

107 %

Saída

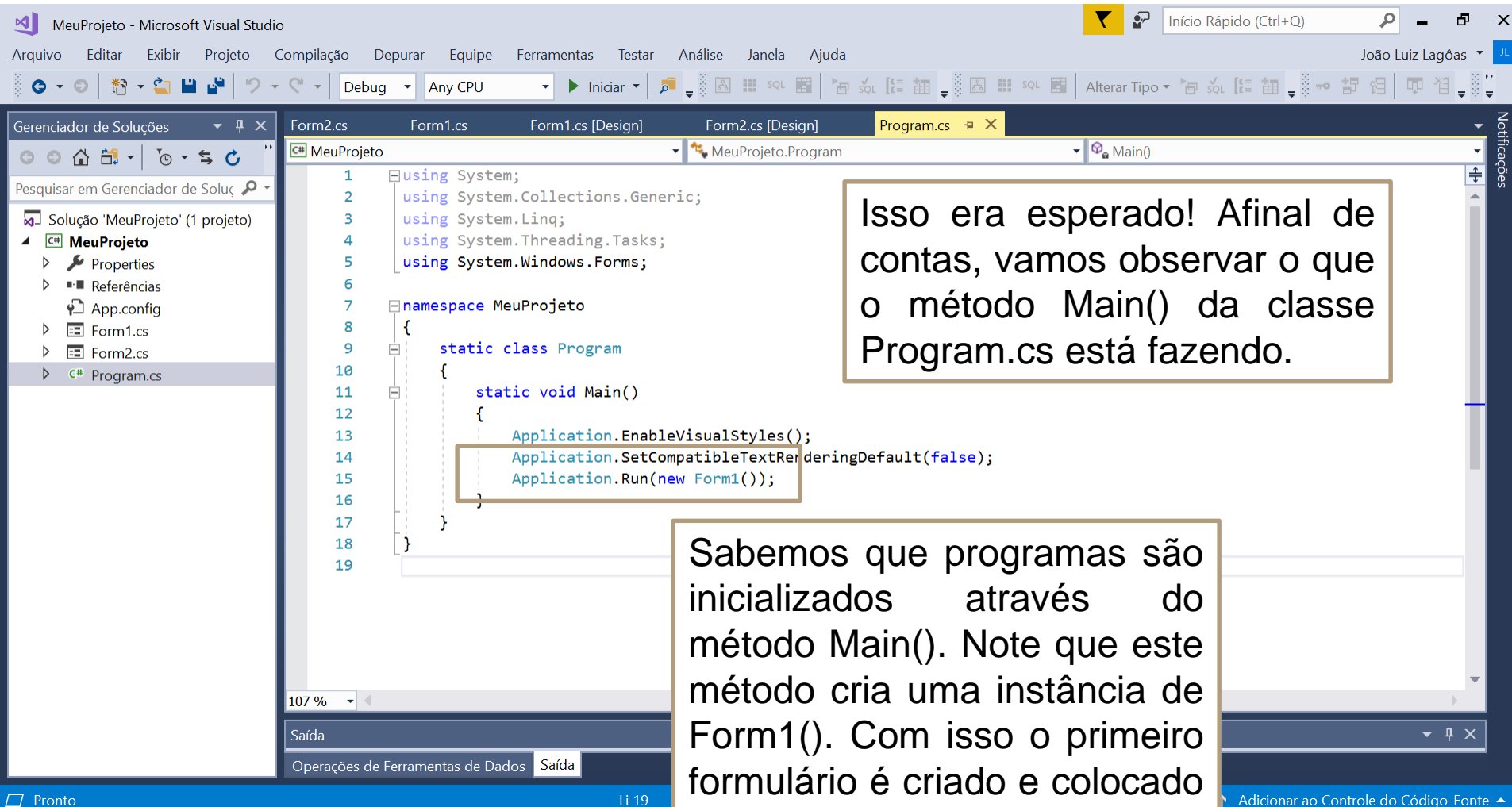
Operações de Ferramentas de Dados Saída

Criando mais de um Formulário



A partir deste ponto, nós temos duas classes implementadas: Form1 e Form2. No entanto, se executarmos o programa, apenas o Form1 irá aparecer.

Criando mais de um Formulário



Para abrir um novo formulário em resposta ao clique (ou qualquer evento) de um outro formulário, basta instanciar um objeto do tipo Form2 dentro de um método de Form1 e então chamar ShowDialog().

```
16 {
17     InitializeComponent();
18 }
19
20 private void btnInserirAluno_Click(object sender, EventArgs e)
21 {
22     using (Form2 form2 = new Form2())
23     {
24         form2.ShowDialog();
25     }
26 }
27
28 private void btnFecharAplicacao_Click(object sender, EventArgs e)
```

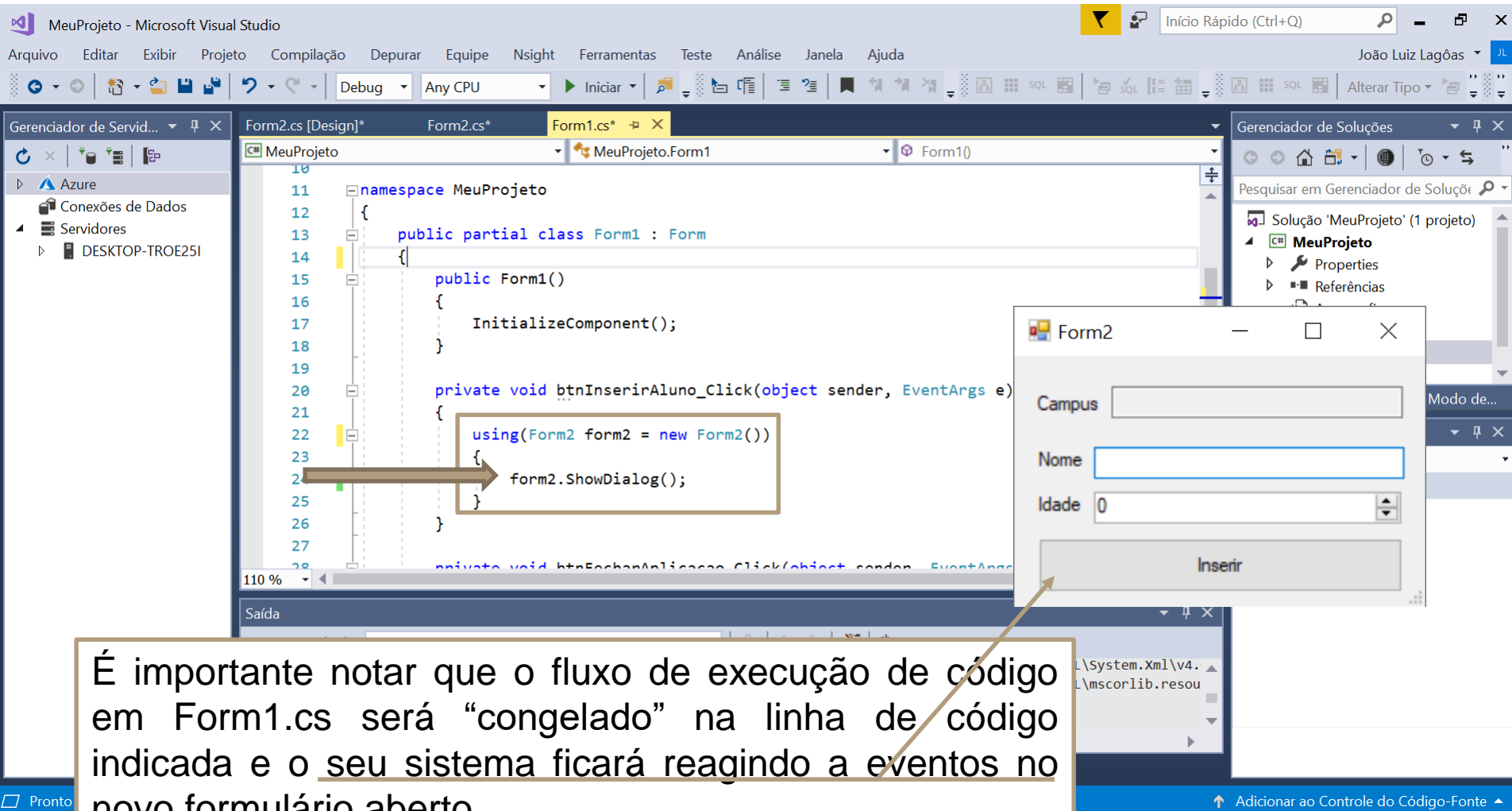
Saída

Mostrar saída de: Depuração

```
"MeuProjeto.exe" (CLR v4.0.30319: MeuProjeto.exe): Carregado "C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\System.Xml\v4.0.30319\System.Xml.dll"
"MeuProjeto.exe" (CLR v4.0.30319: MeuProjeto.exe): Carregado "C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\mscorlib.resources\v4.0.30319\mscorlib.resources.dll"
O programa "[10172] MeuProjeto.exe" foi fechado com o código 0 (0x0).
```

Lista de Erros Data Tools Operations Saída

Criando mais de um Formulário



The screenshot shows the Microsoft Visual Studio IDE with a project named 'MeuProjeto'. The code editor displays the 'Form1.cs' file, which contains the following code:

```
11 namespace MeuProjeto
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void btnInserirAluno_Click(object sender, EventArgs e)
21         {
22             using (Form2 form2 = new Form2())
23             {
24                 form2.ShowDialog();
25             }
26         }
27
28         private void btnEscherAplicacao_Click(object sender, EventArgs e)
29         {
30         }
31     }
32 }
```

A brown box highlights the code block from line 22 to 25, and a brown arrow points from this box to a preview of the 'Form2' dialog box. The 'Form2' dialog box has the following controls:

- Campus:
- Nome:
- Idade:
- Inserir:

At the bottom of the image, a brown box contains the following text:

É importante notar que o fluxo de execução de código em Form1.cs será “congelado” na linha de código indicada e o seu sistema ficará reagindo a eventos no novo formulário aberto.

Como passar informação **entre** formulários?



- Uma pergunta muito comum quando se trabalha com Forms é:

Como que eu passo informações entre dois formulários?

- Essa mesma pergunta costuma ser feita através das seguintes frases:
 - Como eu passo dados entre dois formulários?
 - Como eu acesso dados do Form1 no Form2?
 - Como eu obtenho o conteúdo de uma TextBox, ComboBox, etc, de um outro formulário?

Como passar informação **entre** formulários?



- Todo programa é diferente e apresenta suas particularidades.
- No entanto, o problema de se trocar informações entre formulários tem uma solução prática e comum.

Curiosidade: em programação O.O., soluções genéricas bem estáveis mas que se aplicam a vários programas diferentes são denominadas **Padrões de Projeto**.

Como passar informação entre formulários?



Form1

Campus

São Cristóvão II

Inserir um Aluno

Fechar Aplicação

O Form1 que criou o Form2 pode acessar os dados do Form2 através de métodos *Getters*.

Form2

Campus São Cristóvão II

Nome Pedro

Idade 14

Inserir

O Form2 que foi criado pelo Form1, pode receber dados do Form1 através de métodos *Setters* ou através de seu construtor.

Como passar informação **entre** formulários?



- Sendo assim, iremos fazer algumas alterações no Form2.cs de modo que o seu construtor possa receber uma informação.
- Iremos também definir uma variável denominada Retorno para armazenar um número inteiro. Dependendo de qual botão foi clicado em Form2, iremos atribuir um diferente inteiro para esta variável.

Para que isso?!

Para que através desta variável, Form1 seja capaz de saber qual botão em Form2 foi clicado!

Como passar informação entre formulários?

Primeiramente observe o código da classe Form2



```
public partial class Form2 : Form
```

```
{
```

```
    private int Retorno = 0;
```

```
    public Form2(string campus)
```

```
{
```

```
        InitializeComponent();
```

```
        txtCampus.Text = campus;
```

```
}
```

```
    private void button1_Click(object sender, EventArgs e)
```

```
{
```

```
        Retorno = 1;
```

```
        Close();
```

```
}
```

```
    public string AlunoInfo()
```

```
{
```

```
        string campus = txtCampus.Text;
```

```
        string nome = txtNome.Text;
```

```
        decimal idade = numIdade.Value;
```

```
        return $"{nome} / {campus} / {idade}";
```

```
}
```

```
    public int GetRetorno()
```

```
{
```

```
        return Retorno;
```

```
}
```

```
}
```

Note que o construtor de Form2 espera receber uma string e bota essa string na sua TextBox txtCampus.

Esse método só determina que o atributo Retorno vale 1 e depois fecha o formulário. Esse atributo vai ser utilizado posteriormente pelo Form1 para saber qual botão foi clicado no Form2

Como passar informação entre formulários?

Primeiramente observe o código da classe Form2



```
public partial class Form2 : Form
{
    private int Retorno = 0;

    public Form2(string campus)
    {
        InitializeComponent();

        txtCampus.Text = campus;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Retorno = 1;

        Close();
    }

    public string AlunoInfo()
    {
        string campus = txtCampus.Text;
        string nome = txtNome.Text;
        decimal idade = numIdade.Value;

        return $"{nome} / {campus} / {idade}";
    }

    public int GetRetorno()
    {
        return Retorno;
    }
}
```

Observe que esse método só existe para devolver informações a respeito dos dados do Form2. Ele serve para enviar dados para o formulário que o referencia.

Já este método só é utilizado para devolver o valor armazenado na variável Retorno.

Como passar informação entre formulários?

Passando dados do Form1 para o Form2

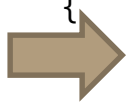


```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnInserirAluno_Click(object sender, EventArgs e)
    {
        using (Form2 form2 = new Form2(cmbCampus.Text))
        {
            form2.ShowDialog();

            if (form2.GetRetorno() == 1)
            {
                string linha = form2.AlunoInfo();
                lstAlunos.Items.Add(linha);
            }
        }
    }

    private void btnFecharAplicacao_Click(object sender, EventArgs e)
    {
        Close();
    }
}
```



Criação de um instância de Form2. Note que é passado para o seu construtor a string que estava armazenada na ComboBox cmbCampus do Form1.

Como passar informação entre formulários?

Passando dados do Form1 para o Form2



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnInserirAluno_Click(object sender, EventArgs e)
    {
        using (Form2 form2 = new Form2(cmbCampus.Text))
        {
            form2.ShowDialog();

            if (form2.GetRetorno() == 1)
            {
                string linha = form2.AlunoInfo();
                lstAlunos.Items.Add(linha);
            }
        }
    }

    private void btnFecharAplicacao_Click(object sender, EventArgs e)
    {
        Close();
    }
}
```

O método ShowDialog do objeto form2 é chamado. Isso significa que a janela será aberta. A continuação da execução do código (**em Form1**) só acontecerá depois que o método ShowDialog terminar sua execução (através do fechamento {método Close()} de form2).

Além disso, quando o form2 for fechado, o seu atributo Retorno estará valendo 1. Dessa maneira, Form1 será capaz de saber qual botão foi clicado nemo form2!

Como passar informação entre formulários?

Passando dados do Form1 para o Form2



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnInserirAluno_Click(object sender, EventArgs e)
    {
        using(Form2 form2 = new Form2(cmbCampus.Text))
        {
            form2.ShowDialog();

            ➡ if(form2.GetRetorno() == 1)
            {
                string linha = form2.AlunoInfo();
                lstAlunos.Items.Add(linha);
            }
        }

        private void btnFecharAplicacao_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Se no form2 o botão clicado foi Inserir, então eu irei acessar os dados do form2 (através do método AlunoInfo()) e exibi-los no form1.

Como passar informação entre formulários?

Passando dados do Form1 para o Form2



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnInserirAluno_Click(object sender, EventArgs e)
    {
        using(Form2 form2 = new Form2(cmbCampus.Text))
        {
            form2.ShowDialog();

            if(form2.GetRetorno() == 1)
            {
                string linha = form2.AlunoInfo();
                lstAlunos.Items.Add(linha);
            }
        }

        private void btnFecharAplicacao_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Como Form1 criou o objeto form2, ele pode acessar seus métodos e atributos públicos. Sendo assim, essa linha de código acessa o método AlunoInfo() de form2 que irá retornar informações armazenadas.

Como passar informação entre formulários?

Passando dados do Form1 para o Form2



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnInserirAluno_Click(object sender, EventArgs e)
    {
        using (Form2 form2 = new Form2(cmbCampus.Text))
        {
            form2.ShowDialog();

            if (form2.GetRetorno() == 1)
            {
                string linha = form2.AlunoInfo();
                lstAlunos.Items.Add(linha);
            }
        }

        private void btnFecharAplicacao_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Dessa forma conseguimos implementar tanto a passagem de dados do form chamador para o form chamado e vice-versa!

Como passar informação **entre** formulários?

Melhorias no código (**Enumerações**)



- A forma como Form2 se fala com Form1 é através da variável Retorno. Esta, por sua vez, é apenas um número inteiro.
- Em situações onde temos números inteiros simplesmente para traduzir alguma resposta (como o clique no botão inserir), podemos utilizar o conceito de **Enumeração**.

Uma **enumeração** é simplesmente um tipo interno a uma classe que define um intervalo de valores válidos.

Enumerações

Exemplos



```
public enum Botão  
{  
    Nenhum,  
    Inserir  
}
```

Nenhum, → 0
Inserir → 1

Nenhum e Inserir são apenas representações mais significativas de 2 inteiros diferentes.

Form2

Campus São Cristóvão II

Nome Pedro

Idade 14

Inserir

Enumerações

Exemplos



```
public enum Direção
```

```
{
```

```
    Norte, _____> 0
```

```
    Sul, _____> 1
```

```
    Leste, _____> 2
```

```
    Oeste, _____> 3
```

```
}
```

Norte, Sul, Leste e Oeste
são apenas representações
mais significativas de 4
inteiros diferentes.



Enumerações

Exemplos



```
public enum Estado
```

```
{
```

```
    Parado, —————> 0
```

```
    Pulando, —————> 1
```

```
    Andando —————> 2
```

```
}
```

Parado, Pulando e Andando são apenas representações mais significativas de 2 inteiros diferentes.



Enumerações



```
public partial class Form2 : Form
{
    public enum Botão
    {
        Nenhum,
        Inserir
    }

    private Botão Retorno = Botão.Nenhum;

    public Botão GetRetorno()
    {
        return Retorno;
    }

    public Form2(string campus)
    {
        InitializeComponent();

        txtCampus.Text = campus;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Retorno = Botão.Inserir;

        Close();
    }

    public string AlunoInfo()
    {
        string campus = txtCampus.Text;
        string nome = txtNome.Text;
        decimal idade = numIdade.Value;

        return $"{nome} / {campus} / {idade}";
    }
}
```

Criação da **enumeração**. Repare que definimos uma gama de possibilidades para uma variável do “tipo” Botão. Ela pode assumir o valor “Nenhum” ou o valor “Inserir”.

Enumerações

```
public partial class Form2 : Form
{
    public enum Botão
    {
        Nenhum,
        Inserir
    }

    private Botão Retorno = Botão.Nenhum;

    public Botão GetRetorno()
    {
        return Retorno;
    }

    public Form2(string campus)
    {
        InitializeComponent();

        txtCampus.Text = campus;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Retorno = Botão.Inserir;

        Close();
    }

    public string AlunoInfo()
    {
        string campus = txtCampus.Text;
        string nome = txtNome.Text;
        decimal idade = numIdade.Value;

        return $"{nome} / {campus} / {idade}";
    }
}
```

A variável Retorno não é mais do tipo int. Ela agora é do tipo Botão e, portanto, só pode assumir dois valores!

A atribuição de uma variável do tipo enum ocorre conforme mostrado no código (Primeiro o nome da enum seguido do nome do seu valor).

Enumerações



```
public partial class Form2 : Form
{
    public enum Botão
    {
        Nenhum,
        Inserir
    }

    private Botão Retorno = Botão.Nenhum;

    public Botão GetRetorno()
    {
        return Retorno;
    }

    public Form2(string campus)
    {
        InitializeComponent();

        txtCampus.Text = campus;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Retorno = Botão.Inserir;

        Close();
    }

    public string AlunoInfo()
    {
        string campus = txtCampus.Text;
        string nome = txtNome.Text;
        decimal idade = numIdade.Value;

        return $"{nome} / {campus} / {idade}";
    }
}
```

Como o método GetRetorno devolver o valor armazenado em Retorno e, esta, por sua vez, agora é uma variável do tipo Botão, então basta modificarmos o tipo de retorno do método de int para Botão.

Enumerações



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void btnInserirAluno_Click(object sender, EventArgs e)
    {
        using(Form2 form2 = new Form2(cmbCampus.Text))
        {
            form2.ShowDialog();

            if(form2.GetRetorno() == Form2.Botão.Inserir)
            {
                string linha = form2.AlunoInfo();
                lstAlunos.Items.Add(linha);
            }
        }
    }

    private void btnFecharAplicacao_Click(object sender, EventArgs e)
    {
        Close();
    }
}
```

O if destacado agora fica mais intuitivo! Está claro que eu só vou entrar nessa condicional se o usuário clicou no botão Inserir do form2.

Aplicação de Multiformulários

Formulário de Credenciais



- Iremos criar uma aplicação que consiste de dois formulários.
- A ideia é carregar um segundo formulário caso o usuário passe para o primeiro formulário condições de credenciais válidas.

Autenticacao

Insira suas credenciais abaixo

Login

Senha

Entrar



ProdutoForm

Preço Mínimo

Preço Máximo

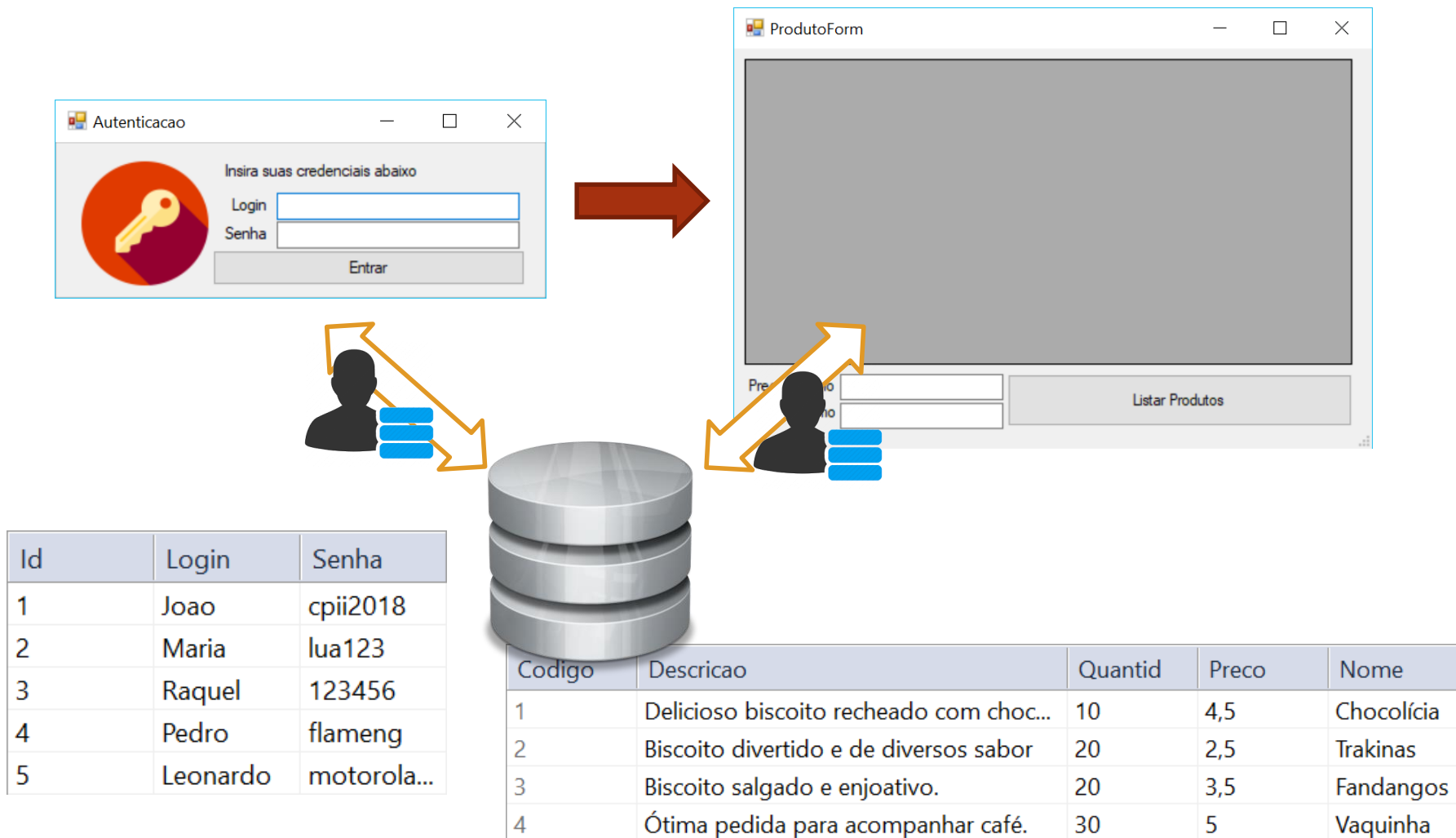
Listar Produtos

Formulários de Autenticação



- Suponha que temos o seguinte cenário:
 - Uma aplicação com duas classes de formulários:
 - **LoginForm.cs**;
 - **ProdutoForm.cs**;
 - Uma classe **DataBaseManager.cs** para se comunicar com o banco de dados.
 - Um banco de dados chamado **Mercado.mdf** com as tabelas:
 - **Usuario**(Id, Login, Senha)
 - **Produto**(Codigo, Descricao, Quantidade, Preco, Nome)

Formulários de Autenticação



Realizando Login

- Suponha que no banco de dados há uma tabela denominada Usuario como. Uma instância dessa tabela é mostrada abaixo.

Id	Login	Senha
1	Joao	cpil2018
2	Maria	lua123
3	Raquel	123456
4	Pedro	flamengo14
5	Leonardo	motorola1718

int

varchar(50)

varchar(50)

Realizando Login

- Que consulta SQL iria validar se um determinado usuário pertence a esse banco de dados?

Id	Login	Senha
1	Joao	cpil2018
2	Maria	lua123
3	Raquel	123456
4	Pedro	flamengo14
5	Leonardo	motorola1718

Várias consultas podem ser usadas aqui! Basta validarmos que um determinado Login está associado a uma determinada senha! Vejamos um exemplo:

```
SELECT COUNT(*) AS Quantidade FROM Usuario  
WHERE Login = 'Leonardo' AND Senha = 'motorola1718';
```

Quantidade
1

Note que se a condição do WHERE for satisfeita, então o resultado da consulta será uma tabela com uma coluna e um registro armazenando o valor 1.

Realizando Login

- Que consulta SQL iria validar se um determinado usuario pertence a esse banco de dados?

Id	Login	Senha
1	Joao	cpil2018
2	Maria	lua123
3	Raquel	123456
4	Pedro	flamengo14
5	Leonardo	motorola1718

Várias consultas podem ser usadas aqui! Basta validarmos que um determinado Login está associado a uma determinada senha! Vejamos um exemplo:

```
SELECT COUNT(*) AS Quantidade FROM Usuario  
WHERE Login = 'Leonardo' AND Senha = 'sapeoufj98sj9d8fh';
```

Quantidade
0

Se a senha ou o usuário não estiverem corretas, a consulta irá retornar uma tabela com uma coluna e um registro armazenando o valor 0.

Realizando Login



Como que então poderíamos utilizar a ideia apresentada agora para realizar um login?

- Podemos utilizar essa consulta que acabamos de discutir para realizar uma autenticação no nosso banco de dados!
- Já sabemos que o método ConsultarBanco da nossa classe DataBaseManager retorna um DataTable que foi gerada de acordo com a consulta SELECT que enviamos para o banco.
- Ora, basta então observar o valor armazenado no registro da primeira linha e da primeira coluna da DataTable que o método retornou!

Realizando Login



txtLogin

txtSenha

btnEntrar

Realizando Login



```
private void button1_Click(object sender, EventArgs e)
{
    string login = txtLogin.Text;
    string senha = txtSenha.Text;

    string stringDeAutenticacao = "SELECT COUNT(*) FROM Usuario " +
                                   "WHERE Login = '" + login + "' AND " +
                                   "Senha = '" + senha + "'";

    DataTable tabela = gerenciador.ConsultarBanco(stringDeAutenticacao);

    int resultado = (int) tabela.Rows[0][0];

    if (resultado == 1)
        MessageBox.Show("Login realizado com sucesso!", "Tentativa de
                          Autenticação", MessageBoxButtons.OKCancel, MessageBoxIcon.None);
    else
        MessageBox.Show("Não foi possível realizar a autenticação. Usuário e
                          senha inválidas.", "Tentativa de Autenticação",
                          MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
}
```


Realizando Login



```
private void button1_Click(object sender, EventArgs e)
```

```
{
```

```
    string login = txtLogin.Text;
```

```
    string senha = txtSenha.Text;
```

```
    string stringDeAutenticacao = "SELECT COUNT(*) FROM Usuario " +  
                                   "WHERE Login = '" + login + "' AND " +  
                                   "Senha = '" + senha + "'";
```

```
    DataTable tabela = gerenciador.ConsultarBa
```

```
    int resultado = (int) tabela.Rows[0][0];
```

```
    if (resultado == 1)
```

```
    {
```

```
        MessageBox.Show("Login realizado com sucesso!", "Tentativa de Autenticação",  
                        MessageBoxButtons.OKCancel, MessageBoxIcon.None);
```

```
        using (ProdutoForm produtoForm = new ProdutoForm())  
            produtoForm.ShowDialog();
```

```
    }
```

```
    else
```

```
        MessageBox.Show("Não foi possível realizar a autenticação. Usuário e senha  
                        inválidas.", "Tentativa de Autenticação",  
                        MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
```

```
}
```

Repare que dessa forma, o outro formulário será aberto apenas se o usuário conseguir se logar!

