# Programação Visual

## Aula 02

Prof. Ederson Cichaczewski

## Conversa inicial

Nesta aula, serão abordadas as ferramentas para desenvolvimento de interfaces gráficas para Windows. Iremos conhecer o MS Visual Studio e seus recursos para desenvolvimento de aplicação em forms, incluindo as linguagens de programação C# (C-Sharp) e Visual Basic .NET (VB.NET). Por fim, trabalharemos em exemplos de aplicações gráficas em .NET.

Veja o vídeo do professor Ederson em que ele apresenta o que mais vamos estudar nesta aula. Acompanhe no seu material virtual.

## Contextualizando

Em nosso dia a dia, utilizamos diversos softwares, e no que diz respeito ao sistema operacional Windows da Microsoft, a interface gráfica é seu ponto forte. Mas como esses aplicativos são desenvolvidos? Utiliza-se um ambiente de programação, que é uma coleção de ferramentas para o desenvolvimento de software, inclui basicamente um editor de código, um editor de telas gráficas e um compilador. Este ambiente é chamado de IDE (*Integrated Development Environment*), que é uma plataforma integrada de desenvolvimento.

O Visual Studio .NET da Microsoft é um exemplo de uma IDE, a qual iremos trabalhar nesta aula. E como estamos tratando de programação visual, daremos ênfase ao desenvolvimento da GUI (*Graphic User Interface*), que é a interface gráfica do usuário. A tela de um software é chamada de “Form”, portanto, focaremos em Windows Forms.

Iremos ver exemplos utilizando as linguagens de programação C# e VB.NET. O Visual Studio .NET tem sua versão gratuita disponível na internet. Caso você queira fazer o *download*, acesse o site a seguir.

https://www.visualstudio.com/downloads/download-visual-studio-vs

O professor Ederson vai contextualizar o assunto desta aula no vídeo disponível no seu material *on-line*.

## Pesquise

### Tema 01: Programação Visual com Microsoft Visual Studio IDE

Vamos conhecer, primeiramente, a ferramenta de desenvolvimento que utilizaremos em nossa aula. Mas antes de falarmos do Visual Studio .NET, é importante saber que para poder desenvolver e rodar aplicativos nas linguagens Visual Basic .NET e C#, é necessário ter instalado o .NET Framework, que é basicamente uma plataforma multilinguagem e uma máquina virtual, que faz com que a aplicação seja independente do sistema operacional.

Portanto, para a configuração do nosso ambiente de desenvolvimento, será necessário ter instalado o .NET Framework e a IDE Visual Studio .NET, de preferência em suas versões mais atuais. Ao instalar o Visual Studio, este inclui todo o pacote, inclusive o framework. Contudo, se quiser rodar uma aplicação .NET em um computador que não tenha o ambiente de desenvolvimento instalado, é necessário instalar o pacote do .NET framework.

É imprescindível ter certa familiaridade com a língua inglesa, pois o Visual Studio não está disponível no idioma Português. Também é necessário conhecer alguns conceitos utilizados pelo Visual Studio: *solution*, *project* e *project item*.

Uma *solution* (solução) é um conjunto de projetos e seus arquivos relacionados que serão parte integrante da aplicação. A primeira ação ao se iniciar uma nova aplicação é criar uma solução. O arquivo da solução tem extensão .SLN. Em aplicações simples, uma solução contém apenas um projeto.

O *project* (projeto) está associado ao tipo de linguagem de programação, seu arquivo terá uma extensão correspondente, no caso para C# é .CSPROJ e para VB é .VBPROJ. O projeto é um conjunto de itens, que são os *project items*, como código fonte, forms, classes, entre outros, que representam os componentes da aplicação, sendo que o código fonte (*source code*) possuirá uma extensão específica, no caso de C# é .cs e para VB é .vb.

Agora, será apresentada a IDE Microsoft Visual Studio .NET 2015 versão *Community*, ou simplesmente Visual Studio Community 2015.

A Figura 1, a seguir, apresenta a tela inicial do Visual Studio.

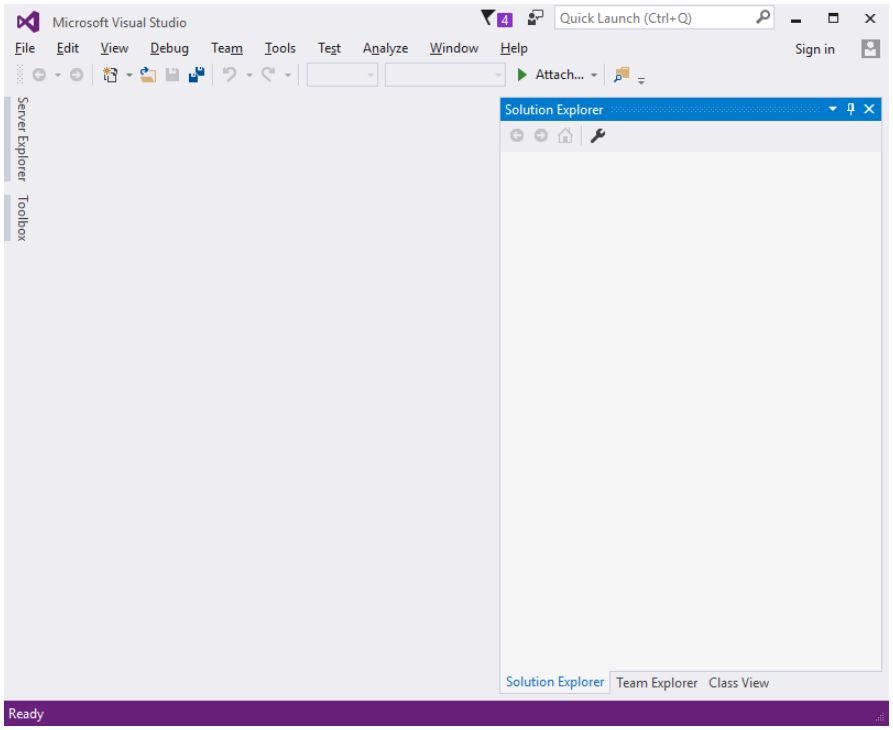


Figura 1 – Tela inicial do Visual Studio.

Para iniciar um novo projeto, seguir os seguintes passos:

- Clicar no menu **File**
- Ir na opção **New**
- Clicar no subitem **Project...**

A Figura 2 apresenta a opção de criar um novo projeto.

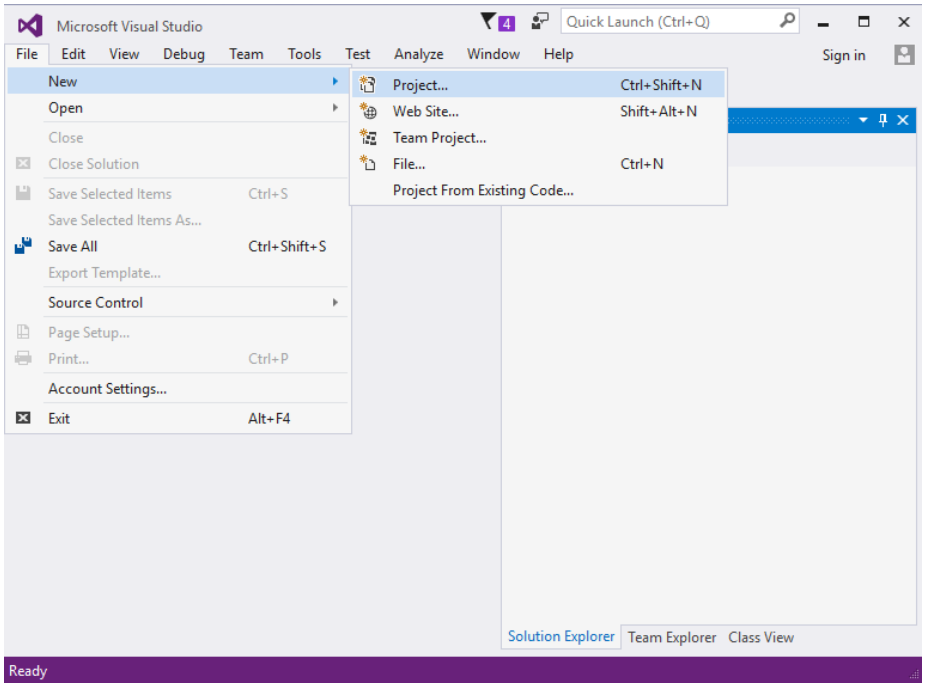


Figura 2 – Opção de criar um novo projeto.

Na sequência, é apresentada a tela com as opções de projeto para as diferentes linguagens de programação, conforme apresentado na Figura 3.

Na coluna mais à esquerda são apresentadas as opções de linguagem de programação disponíveis, em *Templates*.

Ao clicar em uma opção de linguagem de programação, como, por exemplo, C#, na região central são listadas as opções de diferentes tipos de projeto para a linguagem de programação escolhida. Na região mais à direita, é apresentada uma descrição de cada uma dessas opções ao clicar nelas.

Mais abaixo dessa tela, estão os campos para dar nome ao projeto (em *Name*), escolher a pasta onde será gravada a aplicação (em *Location*) e dar nome à solução (em *Solution name*). Por padrão, o nome do projeto é igual ao da solução, mas é possível alterar.

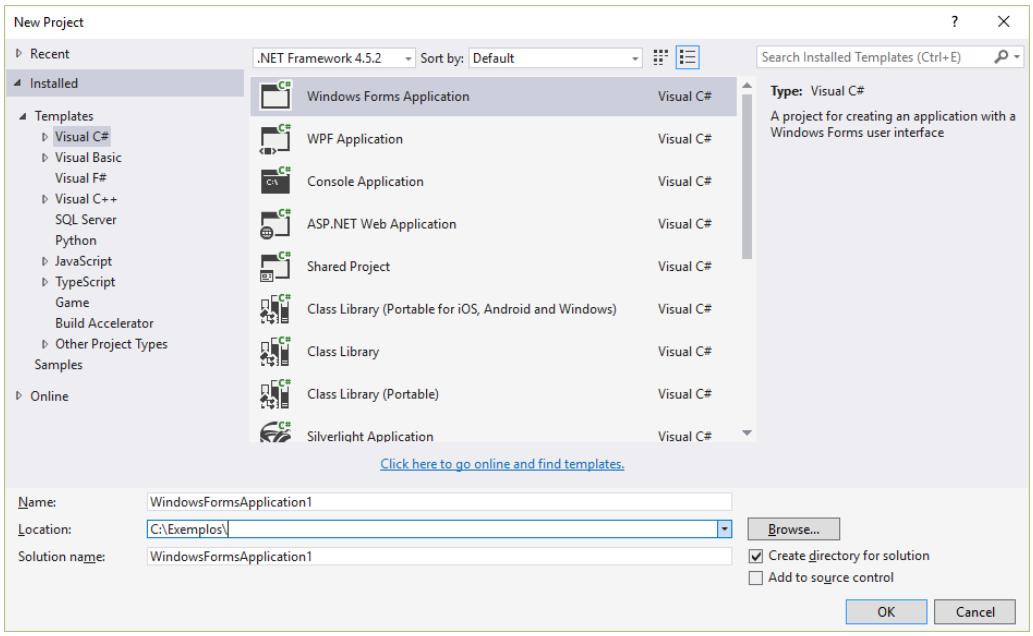


Figura 3 – Opções para criação de um novo projeto.

Como exemplo, vamos criar um projeto C# Windows Forms, clicando no botão OK na tela de novo projeto. A tela do Visual Studio se modifica, apresentando em partes da tela as funcionalidades de acesso rápido, no caso para acessar os arquivos do projeto é em *Solution Explorer,* no canto superior direito, para acessar as propriedades dos recursos é em *Properties* e para acessar os recursos que podem ser adicionados ao projeto é em *Toolbox*, conforme pode-se ver na Figura 4.

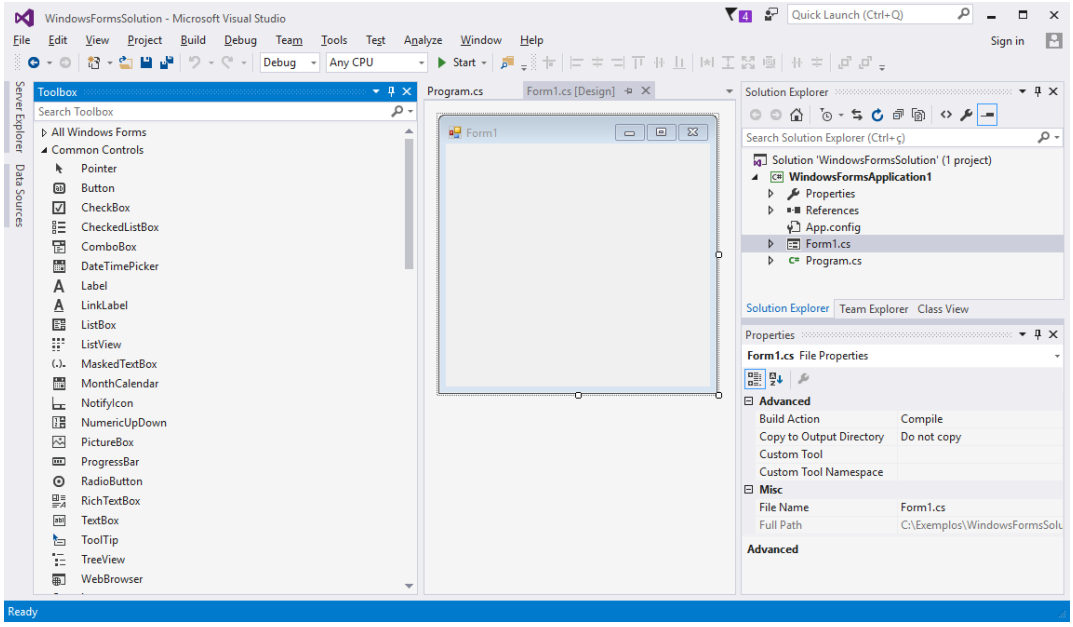


Figura 4 – Tela de desenvolvimento de um programa em C#.

Então é possível trabalhar no projeto, fazendo alterações no form (acessado por meio de um duplo clique em Form1.cs no Solution Explorer) ou no código fonte (acessado por meio de um duplo clique em Program.cs no Solution Explorer).

Para saber se não há nenhum erro após realizar as alterações e implementações desejadas, pode-se clicar no menu *Build* e então clicar em *Build Solution*. Esse processo irá compilar e lincar o projeto, gerando na sua pasta um arquivo executável (extensão .exe), que é a aplicação em si, podendo rodar em qualquer pasta do computador ou qualquer outro computador que tenha instalado o .NET framework. Após a compilação, é possível verificar se houve algum erro (*error*), aviso (*warning*) ou sucesso (*succeeded*) na janela de saída (*Output*).

Para ver a aplicação rodando, clicar no menu *Debug* e então clicar em *Start Debugging* ou simplesmente apertar, no teclado, a tecla F5, que é uma tecla de atalho para esta função. É possível fazer com que a aplicação pare em um determinado ponto do código, adicionando um *breakpoint* na linha de

código desejada, apertando a tecla F9 no teclado e, então, avaliar os valores de variáveis ou outros elementos de código em tempo de execução.

Até aqui foi apresentada de forma básica e introdutória a ferramenta de desenvolvimento de software Visual Studio .NET.

**Saiba mais:** saiba um pouco da história da criação do Visual Studio até chegar na versão 2015 acessando o link a seguir:

https://channel9.msdn.com/Events/Visual-Studio/Visual-Studio-2015-Final-Release-Event/Building-Visual-Studio-2015

**Saiba mais:** você pode conhecer mais sobre o Visual Studio nos seus canais oficiais, confira os sites a seguir:

https://msdn.microsoft.com/library/dd831853(v=vs.140).aspx

https://channel9.msdn.com/VisualStudio

Vamos aprender mais sobre este tema assistindo ao vídeo do professor Ederson no material virtual. Acompanhe!

## Tema 02: Windows Forms

Biblioteca gráfica incluída no .NET Framework, o Windows Forms proporciona uma plataforma para desenvolver aplicações para computadores pessoais, inclusive *tablets* baseados em Windows. O aplicativo em Windows Forms, de um modo geral, apresenta dados na forma gráfica e é orientado a eventos, isso significa que o programa normalmente fica esperando uma interação do usuário.

Um *form* (formulário) nada mais é do que uma superfície visual que mostra informações ao usuário. Normalmente, se constrói uma aplicação adicionando controles a um ou mais forms e implementando em código as respostas às ações do usuário, como cliques do mouse ou pressionamento de teclas. A Figura 5 apresenta um form novo, ainda sem nenhum componente ou controle adicionado.

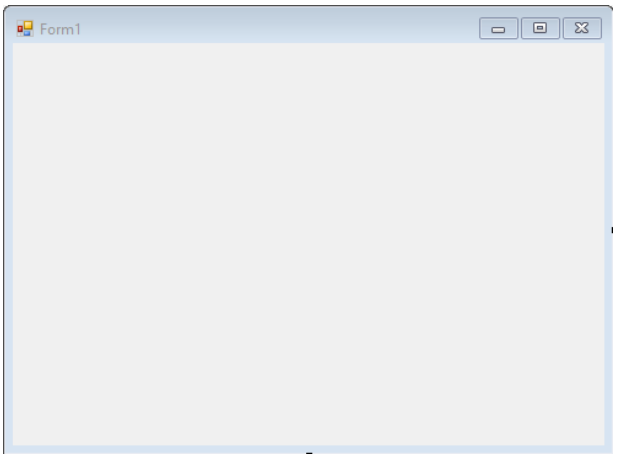


Figura 5 – Windows Form novo.

Um controle (*control*) é um componente da interface do usuário (UI) que mostra dados ou aceita uma entrada de dados.

Os controles mais básicos são os nativos do Windows, como botão (*button*), caixa de texto (*textbox*), caixa de seleção (*checkbox*), entre outros, e estão arranjados em uma caixa de ferramentas (uma janela específica do Visual Studio chamada *Toolbox*), conforme apresentado na Figura 6, a seguir, e acessíveis por meio do recurso de segurar e arrastar com o mouse. As suas configurações incluem posicionamento, tamanho, fonte, texto, e também eventos, como clicar ou passar por cima com o mouse, acessíveis por meio da janela de propriedades (*Properties*).

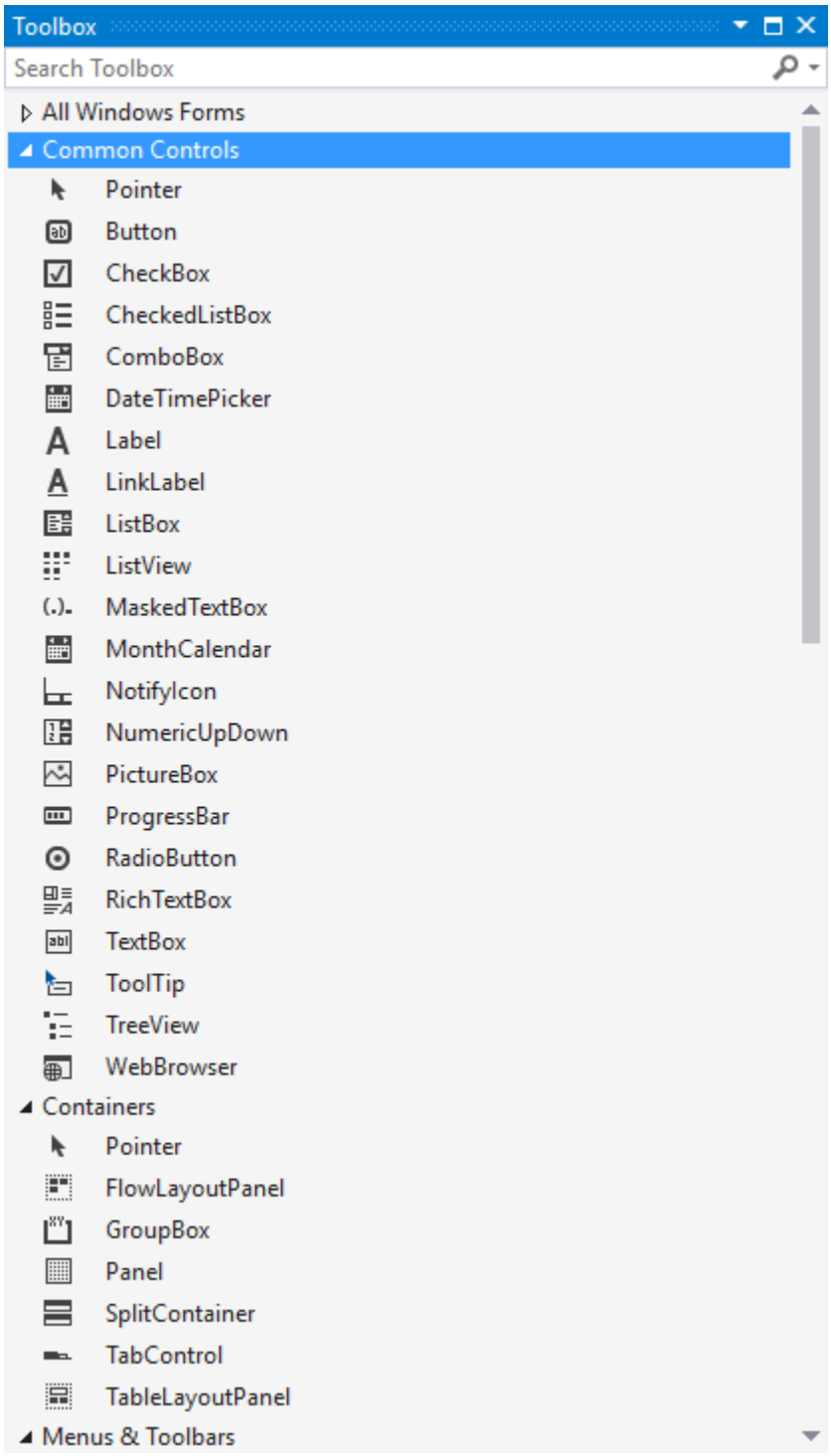


Figura 6 – Caixa de ferramentas de controles para form.

Quando o usuário faz alguma interação com o form ou com alguns dos seus controles, esta ação gera um evento. O aplicativo reage a este evento por meio de um código implementado que manipula este evento (*event handler*), e processa o evento quando ele ocorre. Cada form e cada controle possui um conjunto de eventos pré-definidos, para os quais pode ser feita uma programação. Quando o evento ocorre, e tem um código implementado no manipulador deste evento (*event handler*), este código é invocado. Além dos eventos associados às ações do usuário, há também eventos relacionados ao processo de inicialização (*startup*) da aplicação e seu fechamento (*shutdown*).

Vamos ver um pouco sobre os objetos principais em uma programação visual Windows Form.

## Janelas

A janela é basicamente o form em si. O aplicativo pode ter uma janela principal e janelas secundárias. Nela estão os objetos ou controles da aplicação, com os quais o usuário interage. Normalmente as janelas possuem uma barra superior de menus com as opções do aplicativo e uma barra inferior com algumas informações de status e dicas sobre as funções da aplicação. Com relação ao tamanho, a janela pode ou não ser redimensionada e caso tenha um conteúdo extenso, poderá conter uma barra lateral ou inferior de rolagem.

Além das janelas principal e secundárias, há alguns outros tipos de janelas, como caixas de diálogo (em que se faz uma pergunta ao usuário, a resposta pode ser escolher um botão ou digitar um texto), caixas de mensagens (em que se apresenta uma mensagem ao usuário), paletas de cores e *pop-up* de clique com o botão direito do mouse. A Figura 7 apresenta um exemplo de janela principal, janela secundária e caixa de mensagem.

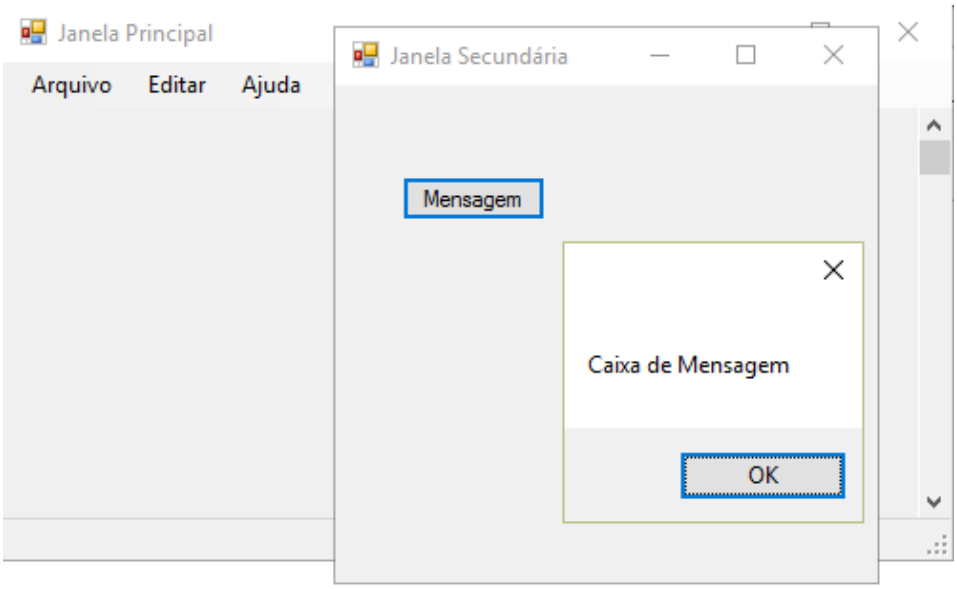


Figura 7 – Exemplo de janela principal, janela secundária e caixa de mensagem.

## Componentes

Abaixo, segue uma lista de tipos de componentes disponíveis:

- Rótulos, *Labels* ou Textos
- Caixas de texto
- Caixas de seleção - *CheckBox*
- Botões de opção - *OptionButton*
- Caixas de Listas - *ListBox*
- Caixas combinadas - *ComboBox*
- Componentes do tipo botões de comando - *Command Button*
- Barras de Rolagem - *ScrollBar*
- Componentes que exibem figuras e gráficos
- Caixas de figuras - *PictureBox*
- Imagens
- Formas e linhas - *Shape, Line*

- Menus
- Caixas de diálogo comum - *Common Dialog*
- Caixas de Diálogo ou Mensagem - *Dialog Box*

A Figura 8 apresenta exemplos de componentes em um form.

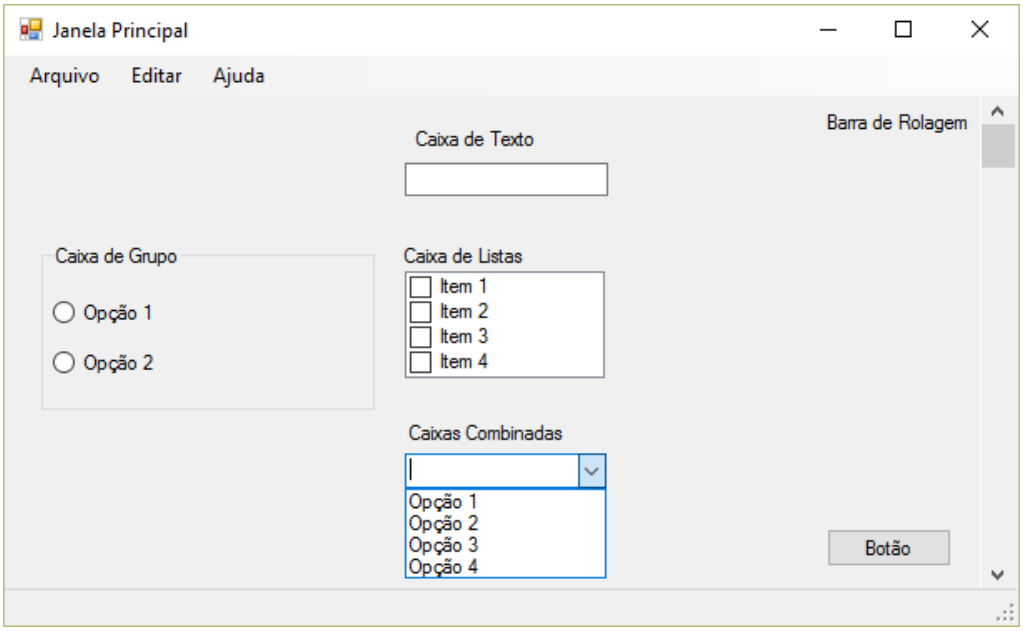


Figura 8 – Exemplos de componentes em um form.

A construção do form e a implementação dos eventos têm características específicas da linguagem em que se está programando, portanto, mais detalhes serão apresentados a seguir para as linguagens C# e VB.NET.

O professor Ederson vai explicar mais detalhes sobre o Windows Forms no vídeo disponível na sua rota *on-line*. Confira!

## Tema 03: Interface Gráfica com C#

O C# (*C sharp*) é uma linguagem de programação baseada em C, C++ e Java que é visual, dirigida por eventos e totalmente orientada a objetos para o desenvolvimento de aplicações que rodam sobre o .NET Framework. Essa

linguagem usa o conceito de máquina virtual, portanto, a aplicação não tem envolvimento direto com o sistema operacional, mas sim com a CLR (*Common Language Runtime*) da plataforma .NET. Contudo, visto que a máquina virtual .NET trabalha com diversas linguagens de programação diferentes, a CLR não pode executar diretamente o código C#, ela precisa executar uma linguagem intermediária comum a todas as linguagens da plataforma .NET, que é a CIL (*Common Intermediate Language*), por isso é necessário passar o código C# por um compilador da linguagem. O compilador lê o arquivo com o código fonte do programa e o traduz para o código intermediário que será executado pela máquina virtual.

A seguir, veremos como desenvolver uma aplicação Windows Form com C#. Vamos criar um novo projeto chamado WindowsFormsAppCsharp e dar nome para a solução de WindowsFormsSolCsharp, conforme a Figura 9.

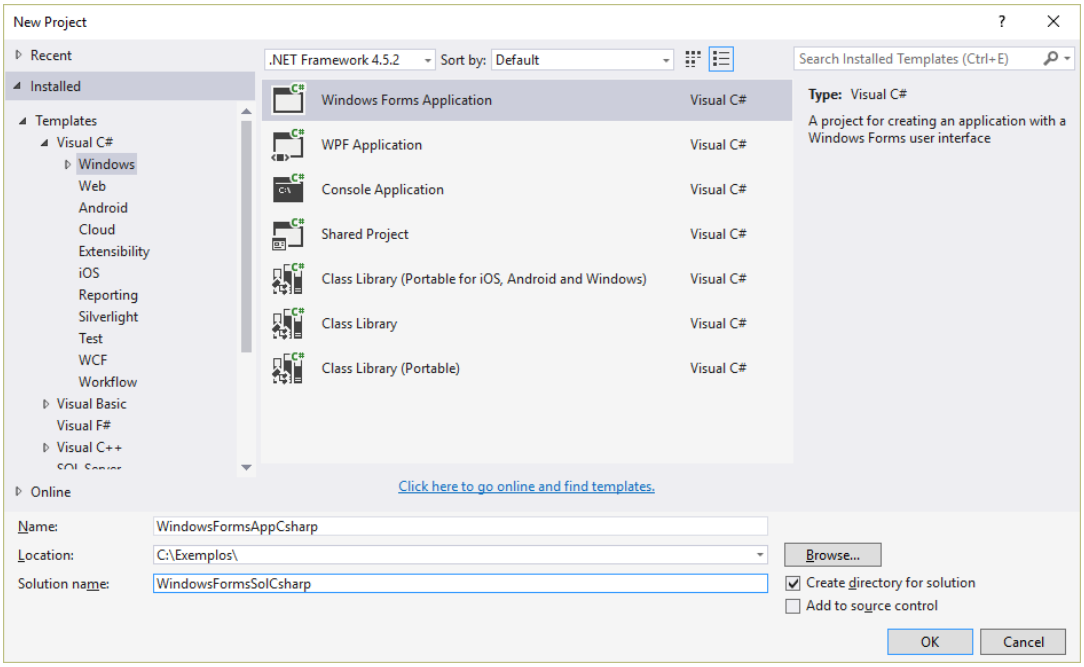


Figura 9 – Novo projeto C# Windows Forms.

Após criado o projeto, a tela do ambiente de desenvolvimento fica conforme a Figura 10.

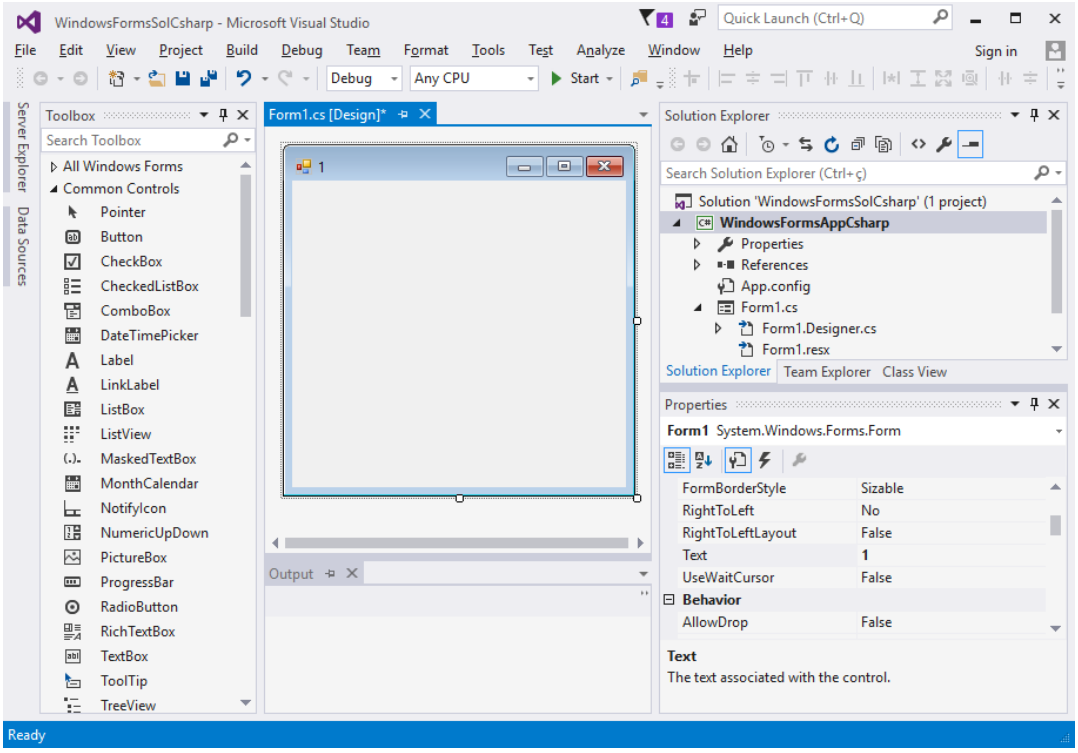


Figura 10 – Ambiente de desenvolvimento C#.

Vamos colocar um botão e editar o evento de clique deste botão para mostrar uma mensagem “Olá Mundo!!!”.

Basta ir ao *Toolbox* e dar um duplo clique sobre o *Button* ou arrastá-lo ao form. Então na janela *Properties* ir na propriedade *Text* e alterar para “Mensagem”, conforme a Figura 11.

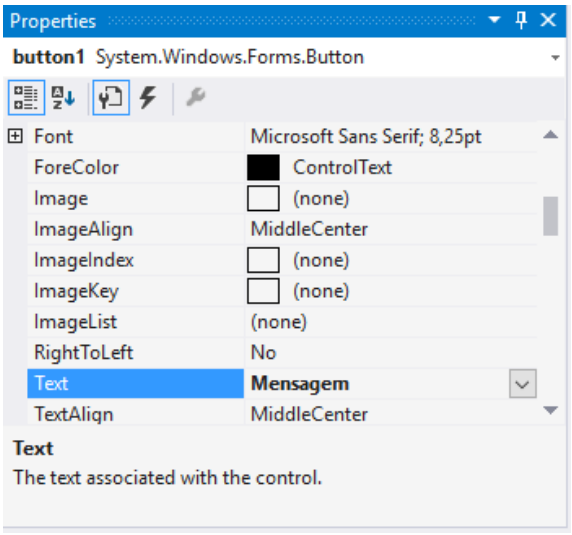


Figura 11 – Propriedade Text do botão.

Em seguida, para implementar a ação do botão, é possível ir até a janela *Properties*, clicar no botão do raio para mudar para a área de eventos, então ir no item *Action*, conforme a Figura 12, e clicar duas vezes em *Click*, ou clicar duas vezes no botão inserido.

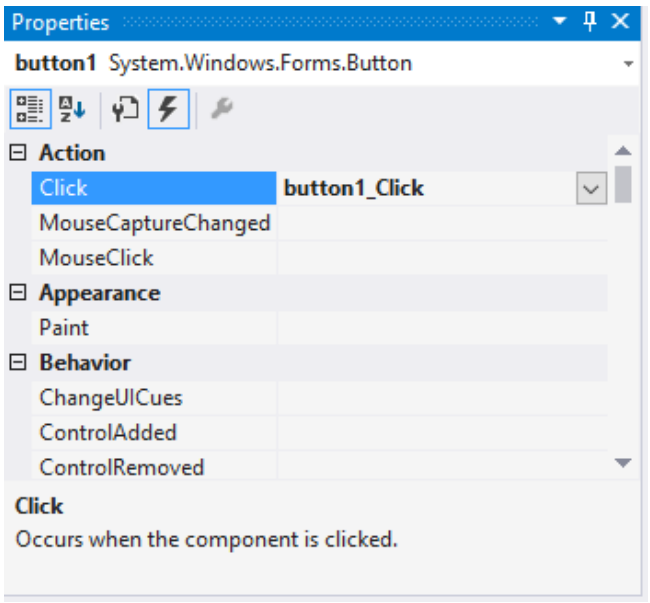


Figura 12 – Evento *Click* do botão.

Será aberto o arquivo de código fonte do form, chamado Form1.cs, na função de clique do botão, então escrever a seguinte linha de código: MessageBox.Show(“Olá Mundo!!!”);

Esta área de código é apresentada na Figura 13.

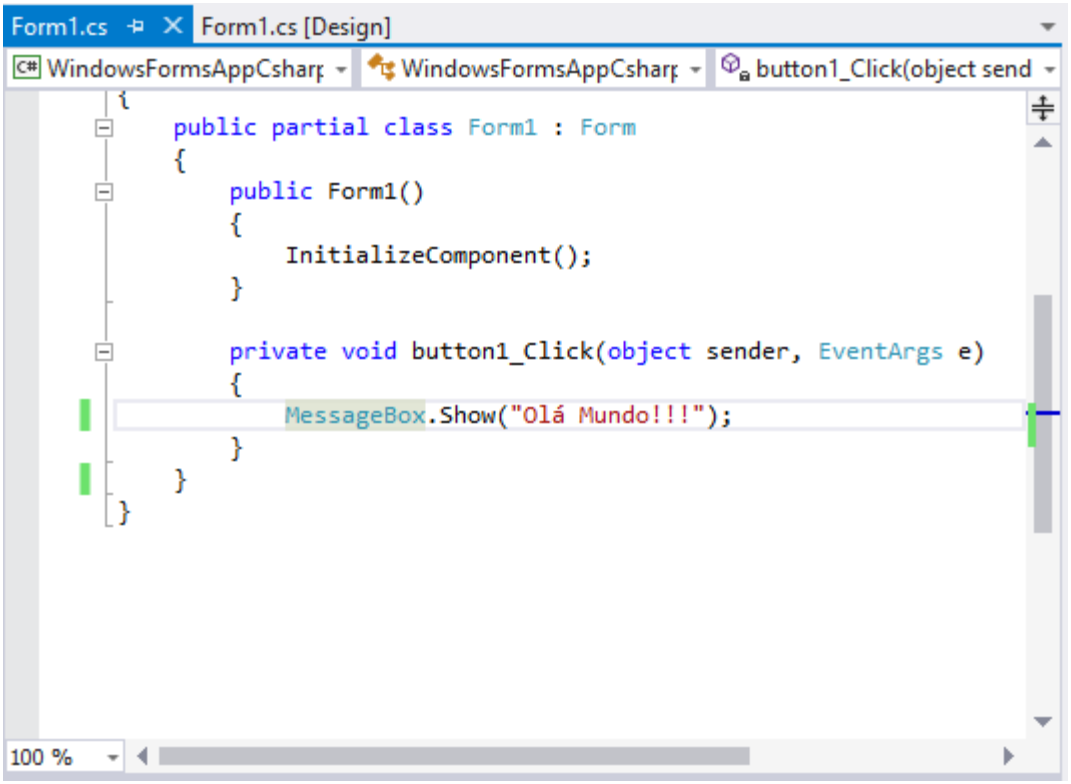


Figura 13 – Código fonte implementado no botão.

Agora é só compilar e rodar o programa, apertando a tecla F5. Irá aparecer o form com um botão, e apertando nesse botão aparece a mensagem. O resultado é apresentado na Figura 14.

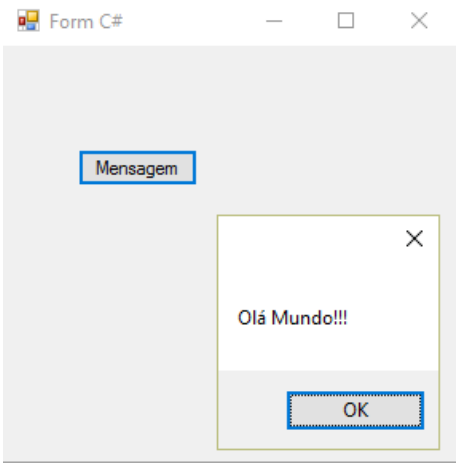


Figura 14 – Form com um botão e mensagem após clique neste botão.

Pronto! Agora é só explorar os demais controles e itens disponíveis no *Toolbox*, assim como suas respectivas propriedades e eventos em *Properties*, para desenvolver a sua aplicação visual em C#.

**Saiba mais:** no link a seguir, você encontra um exemplo de implementação de um aplicativo de teste de matemática com cronômetro em C#:

https://msdn.microsoft.com/pt-BR/library/dd492172.aspx

O professor Ederson vai mostrar como iniciar o desenvolvimento de uma aplicação utilizando o C#. Acompanhe no vídeo disponível na rota.

## Tema 04: Interface Gráfica com VB.NET

O Visual Basic .NET é uma linguagem de programação visual para Windows, orientada a objetos e dirigida a eventos, baseada no .NET Framework. Trata-se de uma evolução da linguagem BASIC (*beginner's all-purpose symbolic instruction code* – código de instrução simbólico de propósito geral para iniciantes).

A seguir, veremos como desenvolver uma aplicação Windows Form com VB.NET.

Vamos criar um novo projeto chamado WindowsFormsAppVBNET e dar

nome para a solução de WindowsFormsSolVBNET, conforme a Figura 15.

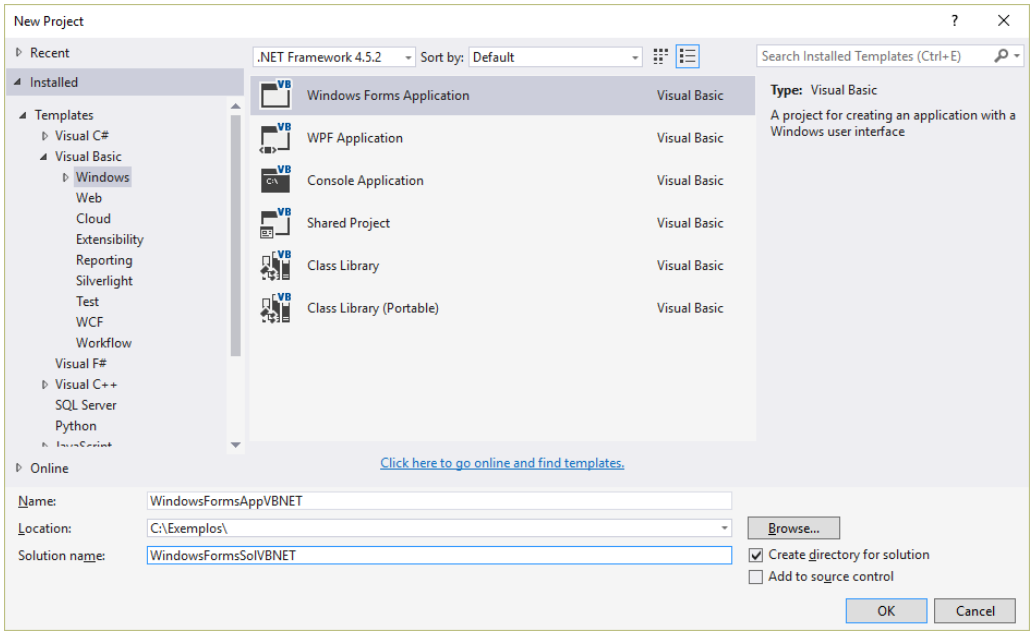


Figura 15 – Novo projeto VB.NET Windows Forms.

Após criado o projeto, a tela do ambiente de desenvolvimento fica conforme a Figura 16.

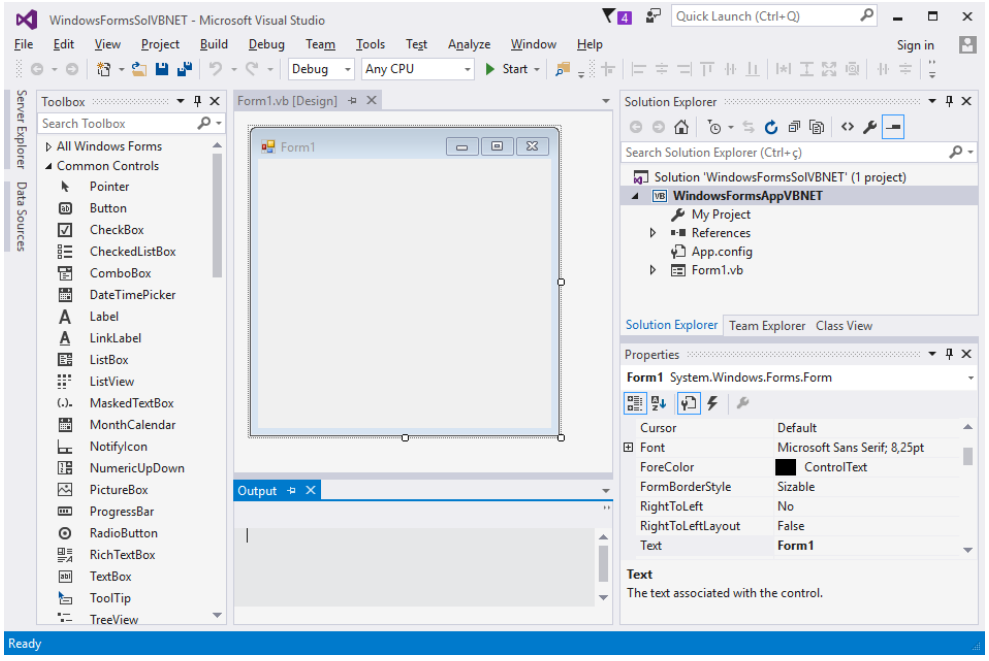


Figura 16 – Ambiente de desenvolvimento VB.NET.

Vamos colocar um botão e editar o evento de clique deste botão para mostrar uma mensagem “Olá Mundo!!!”.

Basta ir ao *Toolbox* e dar um duplo clique sobre o *Button* ou arrastá-lo ao form. Então, na janela *Properties* ir em propriedade *Text* e alterar para “Mensagem”, conforme a Figura 17.

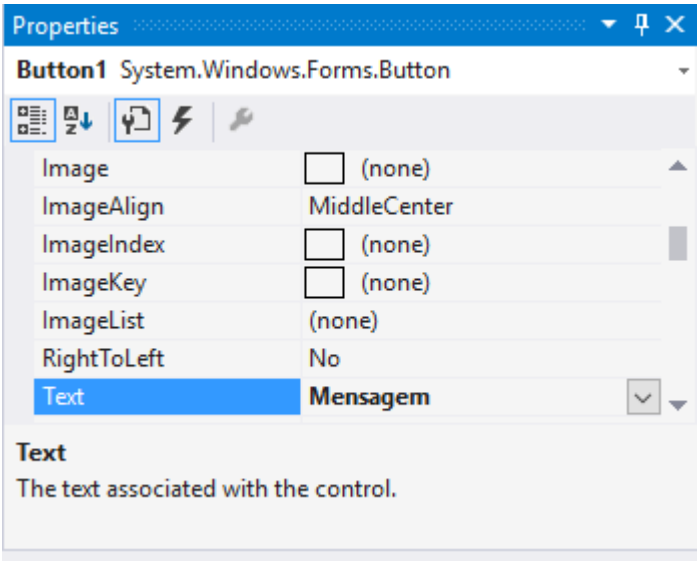


Figura 17 – Propriedade *Text* do botão.

Em seguida, para implementar a ação do botão, é possível ir na janela *Properties*, clicar no botão do raio para mudar para a área de eventos, então ir no item *Action*, conforme a Figura 18, e clicar duas vezes em Click, ou clicar duas vezes no botão inserido.

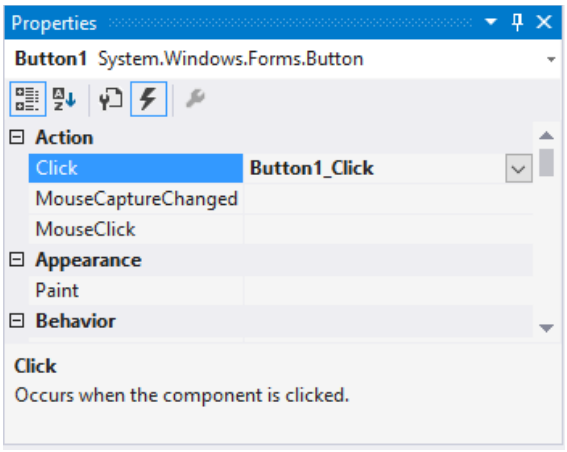


Figura 18 – Evento Click do botão.

Será aberto o arquivo de código fonte do form, chamado Form1.vb, na função de clique do botão, então escrever a seguinte linha de código: MessageBox.Show(“Olá Mundo!!!”)

Esta área de código é apresentada na Figura 19.

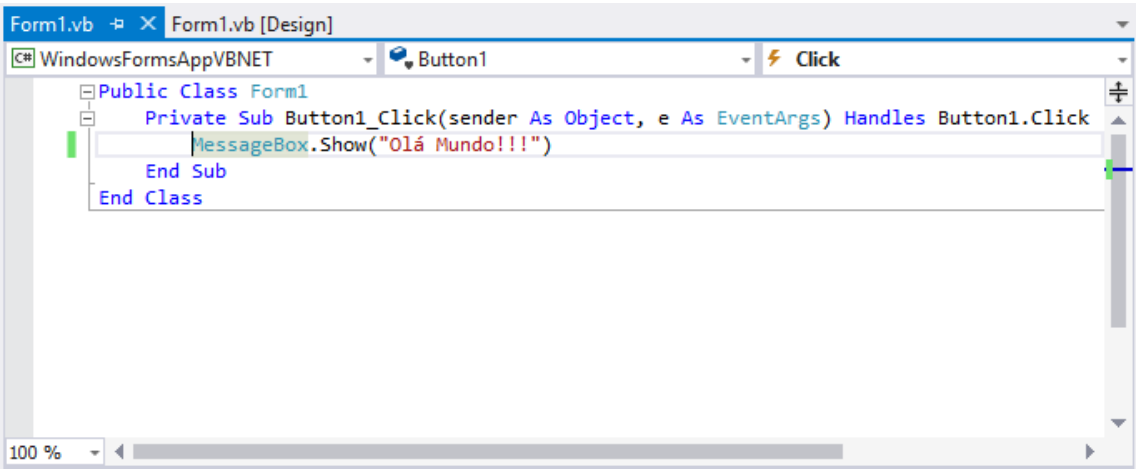


Figura 19 – Código fonte implementado no botão.

Agora é só compilar e rodar o programa apertando a tecla F5. Irá aparecer o form com um botão, e apertando no botão aparece a mensagem. O resultado é apresentado na Figura 20.

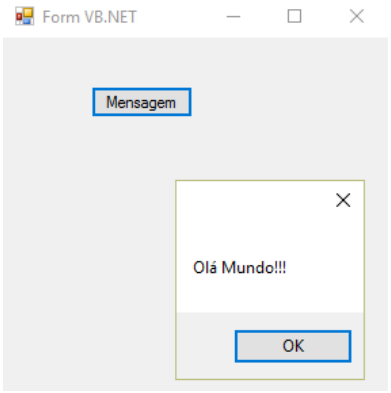


Figura 20 – Form com um botão e mensagem após clique neste botão.

Pronto! Agora é só explorar os demais controles e itens disponíveis no *Toolbox*, assim como suas respectivas propriedades e eventos em *Properties*, para desenvolver a sua aplicação visual em VB.NET.

**Saiba mais:** no site a seguir, você encontra a página oficial do Visual Basic no MSDN Microsoft. Acesse e conheça!

https://msdn.microsoft.com/pt-BR/library/2x7h1hfk.aspx

Agora, veja o vídeo do professor Ederson em que ele mostra um exemplo de aplicação do Windows Form utilizando a linguagem Visual Basic VB.NET.

## Tema 05: Desenvolvimento de Aplicações Gráficas

Agora, vamos ver um exemplo completo de programação visual com Windows Forms.

Vamos fazer um exemplo de projeto em C#, que consiste em um visualizador de imagens. Como a parte visual é igual para um projeto em VB.NET, iremos apenas comentar onde for referente ao código fonte qual seria o código em VB.NET. Dessa forma, este exemplo vale para as duas linguagens de programação.

**1º Passo:** Criar um novo projeto Visual C# do tipo Windows Forms Application, conforme já visto anteriormente na Figura 9. Chame este projeto de PictureViewer.

**2º Passo:** Vamos definir as propriedades do Form1.cs, clicando em qualquer lugar dentro do form, então vá na janela *Properties*, encontre a propriedade *Text* e escreva *Picture Viewer*, conforme a Figura 21. Também vá na propriedade *Size* e deixe com 550 x 350. Teste a aplicação até aqui apertando a tecla F5.

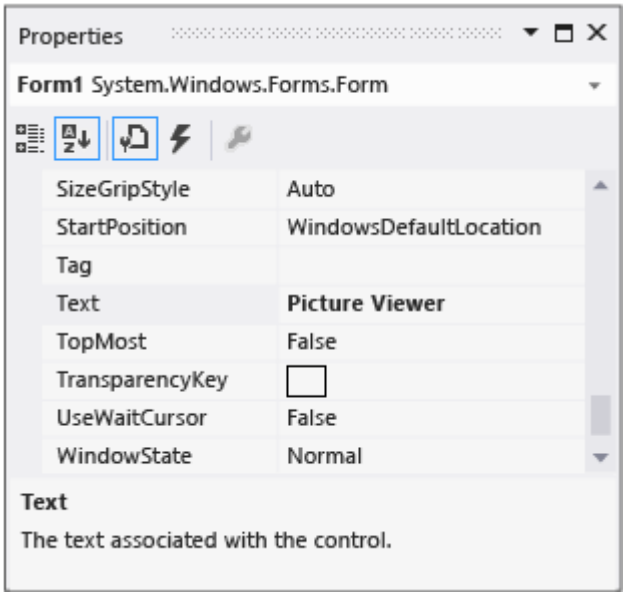


Figura 21 – Propriedade *Text* do aplicativo *Picture Viewer*.

**3º Passo:** Vamos organizar o *layout* do form adicionando o componente TableLayoutPanel. Vá à janela *Toolbox*, em *Containers* clique e arraste para o form o controle TableLayoutPanel. A Figura 22 apresenta o *Toolbox* e a Figura 23 o form com o controle em sua forma padrão. Vá na janela *Properties* e no item *Dock* deixe como *Fill*, conforme a Figura 24. Então, clique no componente TableLayoutPanel, veja que no seu canto superior direito aparece uma setinha para a direita bem pequena, clique nela, irá aparecer um menu de opções, então clique em *Edit Rows and Comuns...*, conforme a Figura 25. Então, selecione *Show Columns* e deixe a coluna 1 com 15% e a coluna 2 com 85%, conforme a Figura 26. Depois, selecione *Show Rows* e deixe a *row 1* com 90% e a *row 2* com 10%. O form com o componente TablelLayoutPanel corretamente configurado é mostrado na Figura 27.

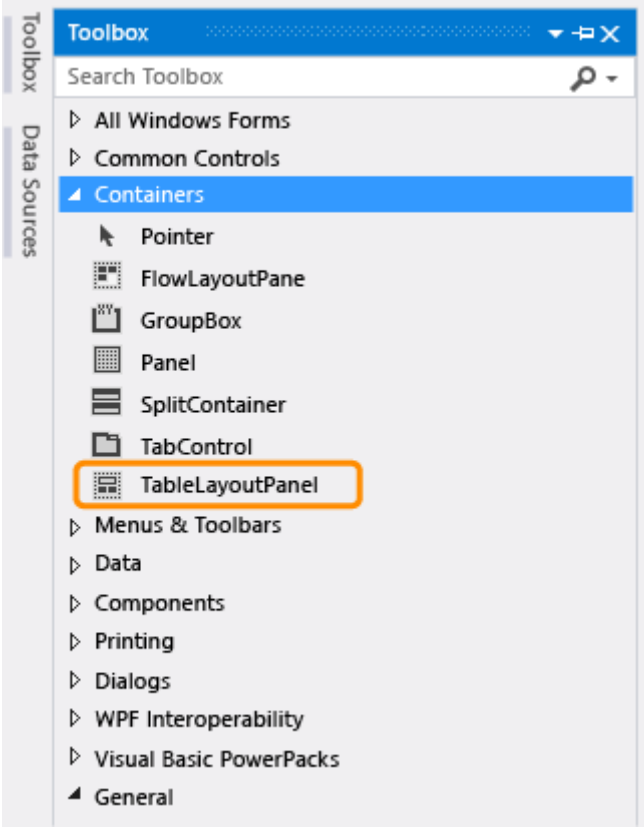


Figura 22 – Componente TableLayoutPanel na janela Toolbox.

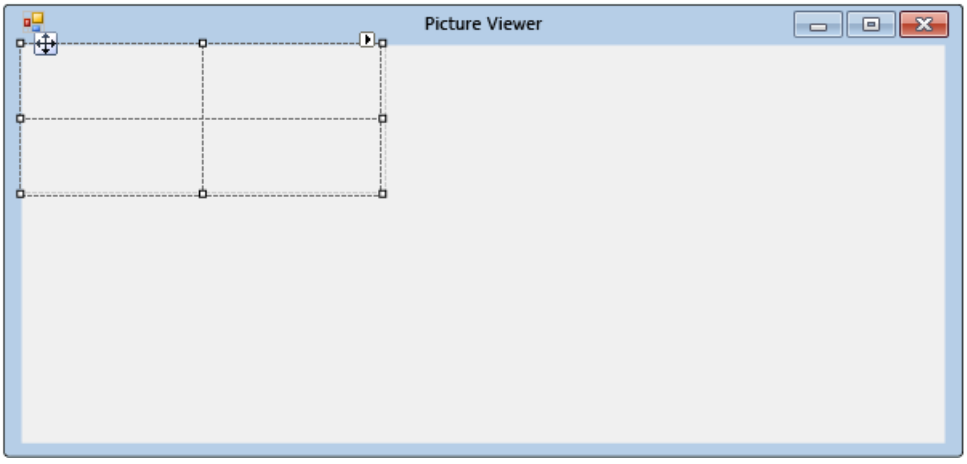


Figura 23 – Form da aplicação com o controle TableLayoutPanel.

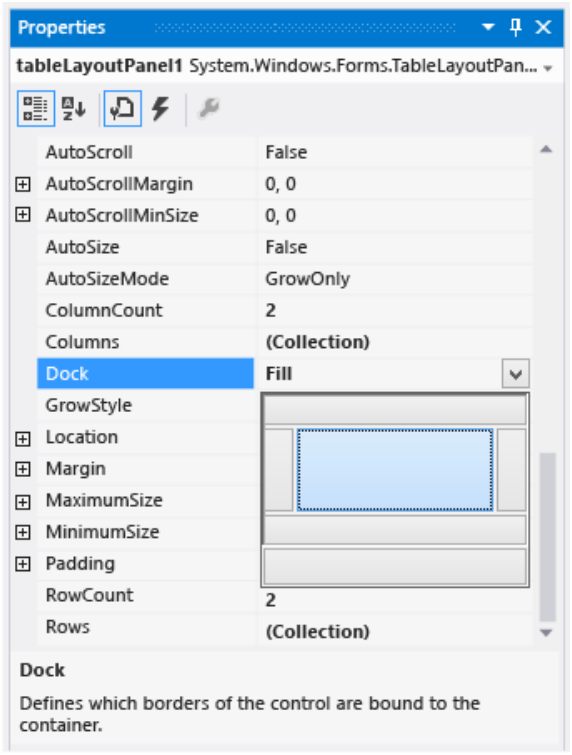


Figura 24 – Item *Dock* como *Fill* na janela *Properties*.

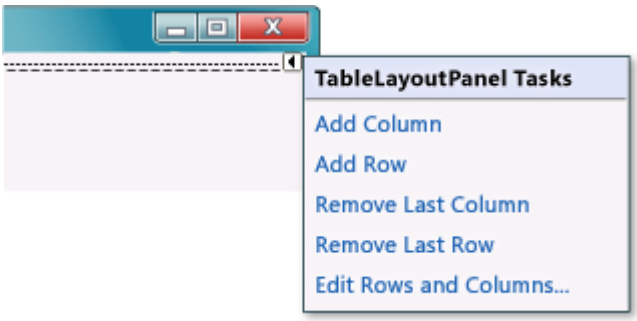


Figura 25 – Opções do componente TableLayoutPanel.

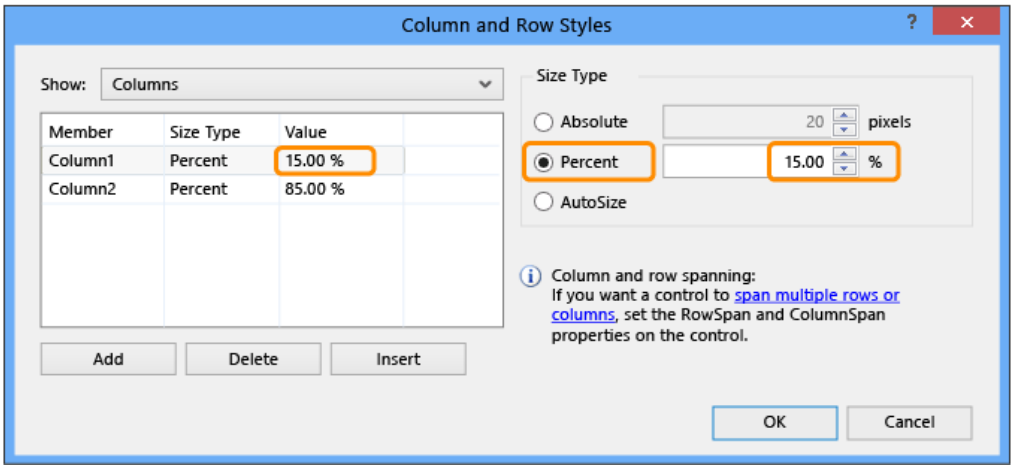


Figura 26 – Configuração de porcentagem de colunas do TableLayoutPanel.

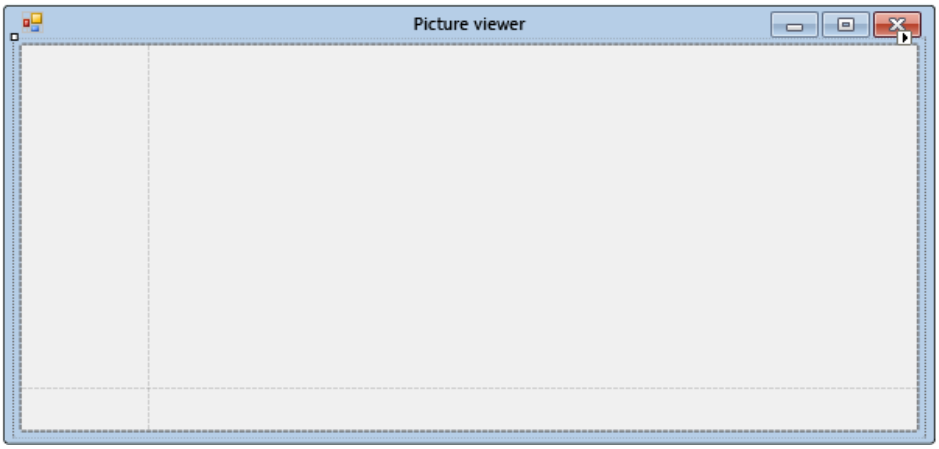


Figura 27 – Form com o componente TableLayoutPanel redimensionado.

**4º Passo:** Vamos inserir um componente de imagem. Na janela *Toolbox*, puxe para o form o item *PictureBox*. Vamos ajustar o seu posicionamento, clique no componente *PictureBox*, irá aparecer uma setinha no seu canto superior direito, clique nela e clique em *Dock in parental container*, conforme a Figura 28, para que ocupe toda a célula do TableLayoutPanel. Para que ocupe também a célula ao lado, vá na janela *Properties* e deixe o item *ComunSpan* com valor 2.

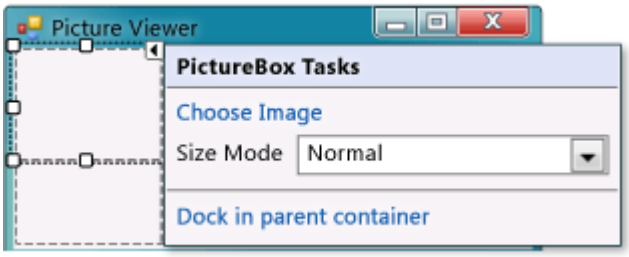


Figura 28 – Opções do componente *PictureBox*.

Agora, adicione um componente de caixa de seleção, vá até a janela *Toolbox* e puxe para o form o item *CheckBox*. Então, vá para a janela *Properties* e deixe *Text* com *Stretch* (que será uma opção de alongar a imagem ou fazer com que ela caiba inteira no espaço de visualização), conforme a Figura 29.

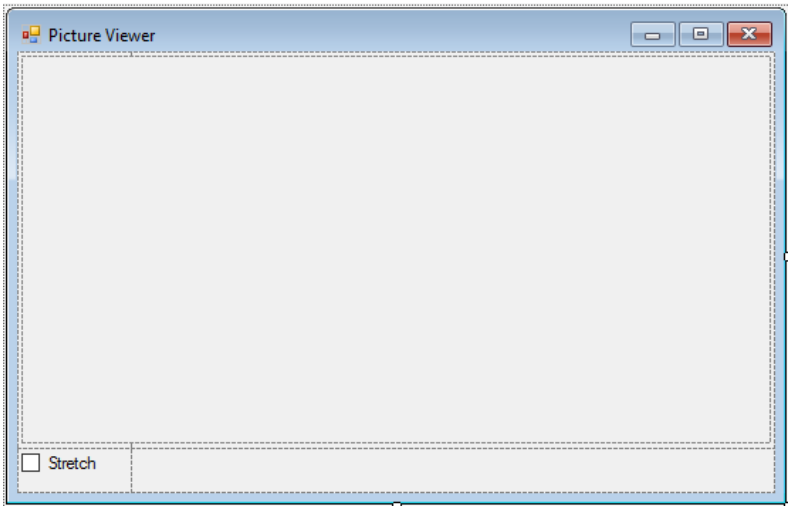


Figura 29 – Componente *CheckBox* adicionado ao form.

Agora, vamos adicionar botões, e para organizá-los, vamos adicionar o componente FlowLayoutPanel, puxando para o form do grupo *Containers* na janela *Toolbox*. Então, configure a sua propriedade *Dock* para *Fill* na janela *Properties*. Vá para a Janela *Toolbox* e puxe 4 botões para o form sobre o FlowLayoutPanel. Na propriedade *Text* de cada botão, escreva sua função, do botão 1 ao 4, conforme a Figura 30, deixe: *Close, Set the background color,*

*Clear the Picture, Show a picture*. Clique no controle FlowLayoutPanel e defina sua propriedade *FlowDirection* para *RightToLeft*. Agora, selecione os botões 1 a 4 todos juntos, segurando a tecla CTRL, então, defina a propriedade *AutoSize* como *True*.

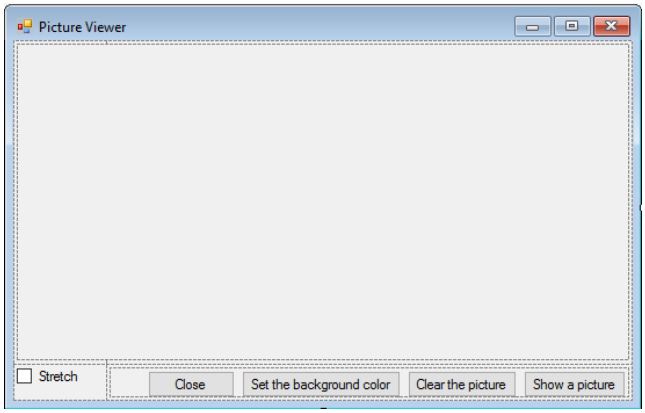


Figura 30 – Botões adicionados ao form.

Agora, para ficar mais fácil de trabalhar com os diferentes botões, é importante dar nome a eles, visto que apenas alteramos o texto que eles mostram, mas não seu nome como será usado no código fonte do programa. Para cada botão, selecione a sua propriedade *Name*, deixando-os assim: closeButton, backgroundButton, clearButton e showButton.

**5º Passo:** Vamos adicionar componentes de diálogo à aplicação. Vá em *Toolbox* e no grupo *Dialog* dê um duplo clique em OpenFileDialog para adicioná-lo ao form. Faça o mesmo para o componente ColorDialog. Eles apareceram abaixo do form, conforme a Figura 31. Clique sobre o OpenFileDialog adicionado, vá na janela *Properties* e no item *Filter* digite: JPEG Files (*.jpg)|*.jpg|PNG *Files* (*.png)|*.png|BMP *Files* (*.bmp)|*.bmp|All files (*.*)|*.*. Também no item *Title* escreva: *Select an image*.

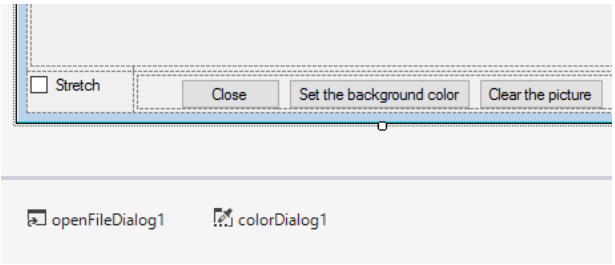


Figura 31 – Componentes de caixa de diálogo.

**6º Passo:** Vamos adicionar código para manipular os eventos da aplicação. Vamos fazer com que o componente PictureBox apresente uma imagem carregada do computador por meio da opção abrir arquivo. Dê um duplo clique no botão *Show a picture*, então digite o código abaixo em C#:

```
// Show the Open File dialog. If the user clicks OK, load the
// picture that the user chose.
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
    pictureBox1.Load(openFileDialog1.FileName);
}
```

Caso a aplicação seja em VB.NET, digite o código abaixo em VB.NET:

```
' Show the Open File dialog. If the user clicks OK, load the
' picture that the user chose.
If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
    PictureBox1.Load(OpenFileDialog1.FileName)
End If
```

Para adicionar o código que configura a cor de fundo, dê um duplo clique no botão *Set the background color*, então digite o código abaixo em C#:

```
// Show the color dialog box. If the user clicks OK, change the
// PictureBox control's background to the color the user chose.
if (colorDialog1.ShowDialog() == DialogResult.OK)
    pictureBox1.BackColor = colorDialog1.Color;
```

Caso a aplicação seja em VB.NET, digite o código abaixo em VB.NET:

```
' Show the color dialog box. If the user clicks OK, change the
' PictureBox control's background to the color the user chose.
If ColorDialog1.ShowDialog() = DialogResult.OK Then
    PictureBox1.BackColor = ColorDialog1.Color
End If
```

Para fechar a aplicação, dê um duplo clique no botão Close, então digite o código abaixo em C#:

```
// Close the form.
this.Close();
```

Caso a aplicação seja em VB.NET, digite o código abaixo em VB.NET:

```
' Close the form.
Close()
```

Para limpar a imagem, dê um duplo clique no botão *Clear the picture*, então digite o código abaixo em C#:

```
// Clear the picture.
pictureBox1.Image = null;
```

Caso a aplicação seja em VB.NET, digite o código abaixo em VB.NET:

```
' Clear the picture.
PictureBox1.Image = Nothing
```

Para alongar a imagem, vá no evento *CheckedChanged* do componente *CheckBox*, dê um duplo clique, então digite o código abaixo em C#:

```
// If the user selects the Stretch check box,
// change the PictureBox's
// SizeMode property to "Stretch". If the user clears
// the check box, change it to "Normal".
if (checkBox1.Checked)
    pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
else
    pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
```

Caso a aplicação seja em VB.NET, digite o código abaixo em VB.NET:

```
' If the user selects the Stretch check box, change
' the PictureBox's SizeMode property to "Stretch". If the user
' clears the check box, change it to "Normal".
If CheckBox1.Checked Then
PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
Else
PictureBox1.SizeMode = PictureBoxSizeMode.Normal
End If
```

Agora é só apertar a tecla F5 para compilar e rodar a aplicação. Teste clicando no botão *Show a picture*, escolha uma imagem para abrir, deixe marcada a opção *Strech*, o resultado pode ser viso na Figura 32.


Figura 32 – Aplicação final com uma imagem aberta.

Vamos aprender mais com o professor Ederson, ele mostra a aplicação gráfica mais completa utilizando o Windows Forms. Confira no seu material virtual.

## Trocando ideias

Nesta aula, iniciamos com o desenvolvimento de aplicações para o Sistema Operacional Windows, utilizando os conceitos de programação visual. Para tanto, conhecemos o IDE chamado Visual Studio .NET e seus recursos para construir interfaces gráficas. Pudemos perceber que há disponíveis em uma janela de ferramentas todas as funcionalidades que normalmente encontramos nos aplicativos que usamos em nosso dia a dia, como botões, menus, barra de status, caixa de seleção etc.

Algumas funcionalidades são dependentes de uma linguagem de programação específica, como a codificação para o tratamento de eventos, mas a parte gráfica compartilha, em sua maioria, os mesmos controles entre as linguagens C# e VB.NET que vimos nesta aula.

Busque explorar mais recursos gráficos disponíveis para ambas as linguagens de programação no Visual Studio .NET. Lembre-se que aplicativos .NET necessitam da máquina virtual .NET Framework para rodar, o que garante para a aplicação sua independência de hardware e de sistema operacional.

O conhecimento adquirido nesta aula é fundamental para a continuidade desta disciplina, assim como será utilizado ao longo de todo o curso. Por isso, não fique com dúvidas sobre o assunto desta aula, estude o tema consultando as referências bibliográficas desta aula e também outras fontes de pesquisa.

## Na prática

A programação visual engloba uma diversa gama de aplicações, desde aplicativos para imagens, música, vídeos, edição de textos, planilhas, leitura etc. Uma das áreas que tem bastante desenvolvimento é a de jogos. O que você acha de desenvolver um jogo com os conceitos e ferramentas aprendidos nesta aula e ainda explorar novos recursos?

Desenvolva um jogo de memória, escolhendo a linguagem C# ou VB.NET. A tela deve ter uma matriz 4x4, em que imagens ficam ocultas, ao clicar em uma célula da matriz ela irá mostrar a imagem nela; então deve-se clicar em outra célula para encontrar outra imagem igual à esta. Caso as imagens sejam iguais, elas ficam aparecendo na tela, caso contrário as imagens voltam a ficar ocultas. Ao descobrir todos os pares de imagens informe ao usuário que ele chegou ao fim do jogo, ou seja, o jogador ganhou.

Para referência da solução em como implementar esta aplicação, consulte o link a seguir:

https://msdn.microsoft.com/pt-br/library/dd553235.aspx

## Síntese

Nesta aula, trabalhamos a programação visual com MS Visual Studio IDE, Windows Forms, interfaces gráficas com C#, interfaces gráficas com VB.NET e desenvolvimento de aplicações gráficas.

O bom entendimento desta aula é fundamental, visto que trata dos recursos básicos mais utilizados em programação visual para aplicativos baseados em Windows.

O conhecimento adquirido é fundamental para a continuidade desta disciplina, assim como será utilizado ao longo de todo o curso. Por isso, não fique com dúvidas sobre o assunto desta aula, estude o tema consultando as referências bibliográficas desta aula e também outras fontes de pesquisa.

Assista ao último vídeo desta aula em que o professor Ederson comenta os principais pontos estudados.

## Referências

DEITEL, H. M. **C# - Como programar**. São Paulo: Pearson Education. 2003.

DEITEL, H. M. **Visual Basic .NET - Como programar**. São Paulo: Pearson Education. 2004.

MICROSOFT. **Guia de Introdução ao Visual Studio**. Disponível em: <https://msdn.microsoft.com/pt-br/library/ms165079.aspx>. Acesso em: 10 ago. 2016.