



Habilitação Profissional Técnica de Nível Médio de **TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS**

DESENVOLVIMENTO DE SISTEMAS

Introdução ao Visual Studio

Visual Studio é um conjunto completo de ferramentas de desenvolvimento para construção de aplicações Web ASP.NET, serviços Web XML, aplicações desktop e aplicativos móveis.

Visual Basic, Visual C# e Visual C++ todos usam o mesmo ambiente de desenvolvimento integrado (IDE), que permite o compartilhamento de ferramentas e facilita a criação de soluções de linguagens.

Introdução ao Visual Studio

Além disso, essas linguagens usam a funcionalidade do *.NET Framework*, que fornece acesso às tecnologias-chaves que simplificam o desenvolvimento de aplicativos Web em ASP e serviços Web XML.

O que posso fazer com o .NET?

O .NET permite desenvolver soluções como:

- Aplicativos Web
- Aplicativos para Servidores
- Aplicativos *Smart Client*
- Aplicativos de Console
- Aplicativos de Banco de Dados
- Serviços Windows
- Serviços Web

TERMOS DA PLATAFORMA

CLR - Sigla de *Common Language Runtime*. Base comum a todas as linguagens .NET, o CLR é o ambiente que gerencia a execução de código escrito em qualquer linguagem. Faz parte do *Framework*.

FRAMEWORK - É o modelo da plataforma .NET para construir, instalar e rodar qualquer aplicação, no desktop ou na Internet. Para executar um programa .NET, é preciso ter o *Framework* instalado.

TERMOS DA PLATAFORMA

IDE COMPARTILHADO - Ambiente integrado de programação (*Integrated Development Environment*) do Visual Studio.NET. Diferentes linguagens usam o mesmo editor de código e depurador e compilam executáveis na linguagem MSIL. Além das linguagens da Microsoft, já há mais de 20 outras (Perl, Cobol, Pascal, etc) que podem usar esse ambiente.

MSIL - *Microsoft Intermediate Language*. Quando se compila uma aplicação .NET, ela é convertida para uma linguagem intermediária, a MSIL, um conjunto de instruções independentes de CPU. Na hora de executar o programa, um novo compilador, chamado *Just-in-time (JIT) Compiler*, o converte para o código nativo, ou seja, específico para o processador da máquina.

MANAGED CODE - Código administrado, ou seja, código escrito para rodar com o *runtime* do VS.NET. No VS.NET, somente o C++ produz programas que não dependem do *runtime*, o chamado *Unmanaged code*.

Visão geral do .NET Framework

É uma tecnologia que dá suporte à compilação e à execução da próxima geração de aplicativos e serviços Web XML. O *.NET Framework* foi criado para atender os seguintes objetivos:

- Para fornecer um ambiente de programação orientada a objetos consistente, quer o código objeto seja armazenado e executado localmente ou remotamente.
- Fornecer um ambiente de execução que minimize conflitos de versionamento de publicação.
- Fornecer um ambiente de execução que promova a execução segura de código criado por desconhecidos ou código de terceiros com baixo nível de confiança
- Para fornecer um ambiente de execução que elimina os problemas de desempenho dos ambientes interpretados ou com scripts.
- Para tornar a experiência do desenvolvedor consistente, através dos diversos tipos de aplicativos, como aplicativos baseados no Windows e Web.
- Para executar toda comunicação usando padrões da indústria, assim garantindo que códigos baseados no *.NET Framework* possam se integrar a qualquer outro código.

CLR (*Common Language Runtime*)

O .NET Framework tem um ambiente de tempo de execução chamado de *Common Language Runtime*, que executa o código e provê serviços que tornam o processo de desenvolvimento mais fácil.

Compiladores e ferramentas expõem as funcionalidades do CLR e habilitam você escrever código que se beneficia desse ambiente de execução gerenciado.

Código que você desenvolve com um compilador de linguagem que tem como alvo o *runtime* é chamado de código gerenciado; ele se beneficia de recursos como integração entre linguagens, tratamento de exceção entre linguagens, segurança aprimorada, suporte a versionamento e implantação, um modelo simplificado para interação entre componentes, e serviços de depuração e de perfil.

CLR (*Common Language Runtime*)

O *runtime* automaticamente trata o leiaute de objetos e gerencia referências a objetos, liberando-os quando eles não estão sendo usados. Objetos cujos tempos de vida são gerenciados dessa forma são chamados de dados gerenciados.

A coleta de lixo elimina perdas de memória, bem como alguns outros erros de programação comuns. Se o código for gerenciado, você poderá usar dados gerenciados, dados não gerenciados ou ambos no seu aplicativo do *.NET Framework*.

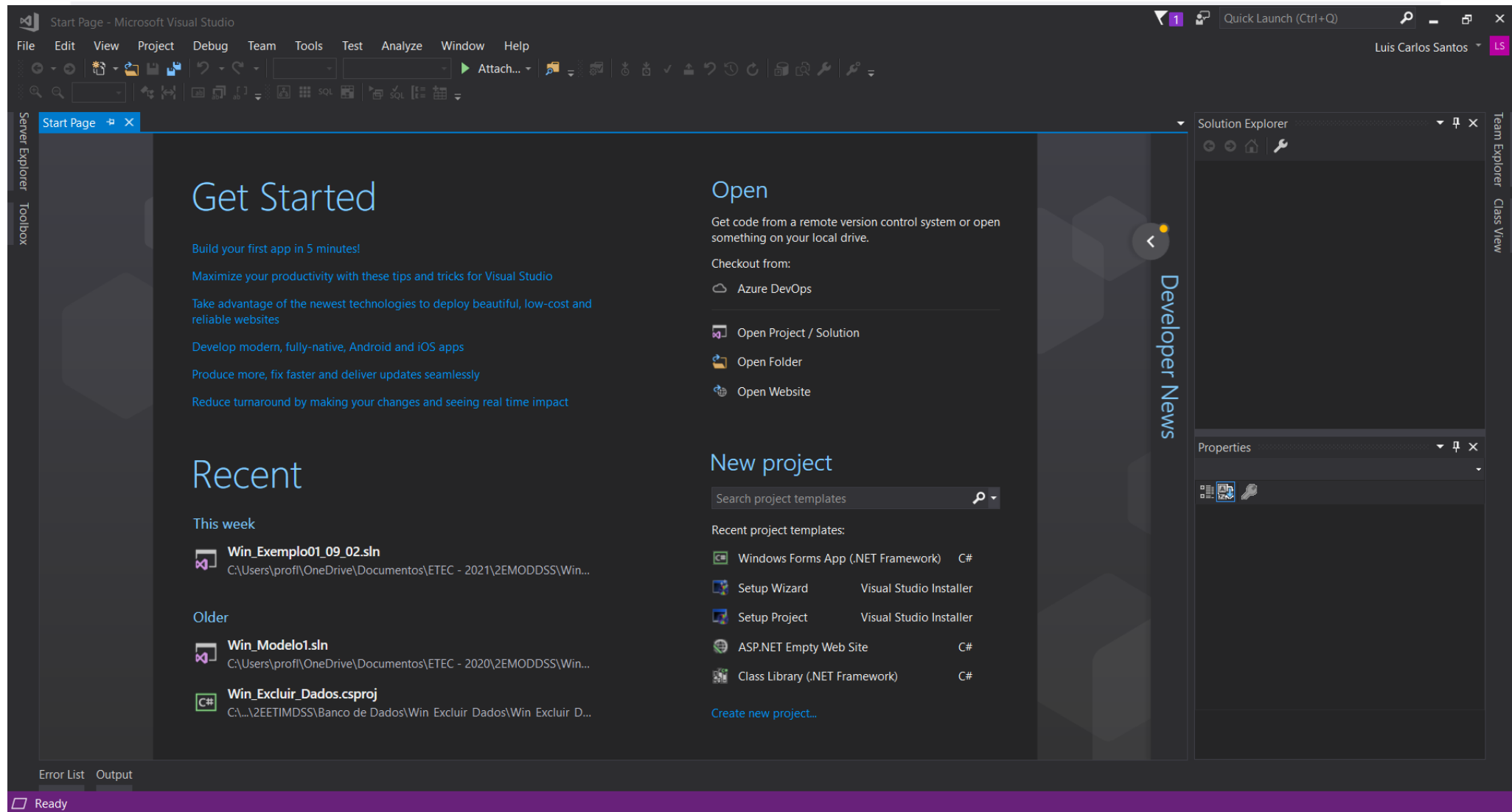
Devido ao fato de compiladores de linguagens fornecerem seus próprios tipos, como tipos primitivos, você nem sempre pode saber (ou precisa saber) se seus dados estão sendo gerenciados.

CLR (*Common Language Runtime*)

O tempo de execução oferece os seguintes benefícios:

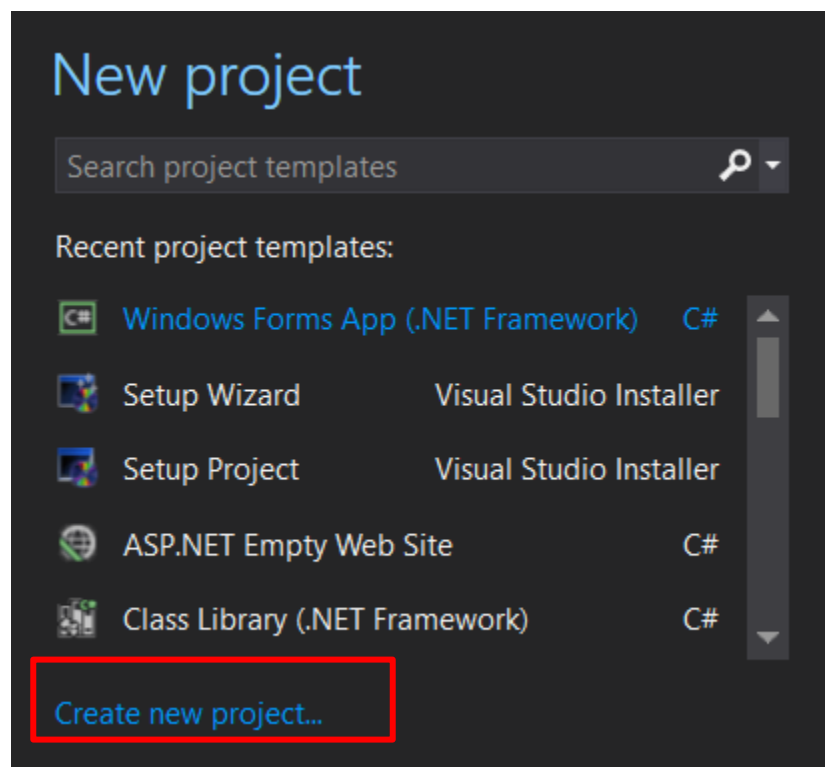
- Melhorias de desempenho.
- A capacidade de utilizar facilmente componentes desenvolvidos em outras línguas.
- Tipos extensíveis fornecidos por uma biblioteca de classes.
- Recursos da linguagem como a herança, interfaces, e sobrecarga para programação orientada a objeto.
- Suporte a definição explícita de threads que permite a criação de aplicações *multi-thread* escalonáveis .
- Suporte a manipulação estruturada de exceções .
- Suporte a atributos personalizados.
- Coleta de lixo.
- Uso de *delegates* em vez de ponteiros de função para aumentar a segurança de tipos e segurança.

Conhecendo o ambiente

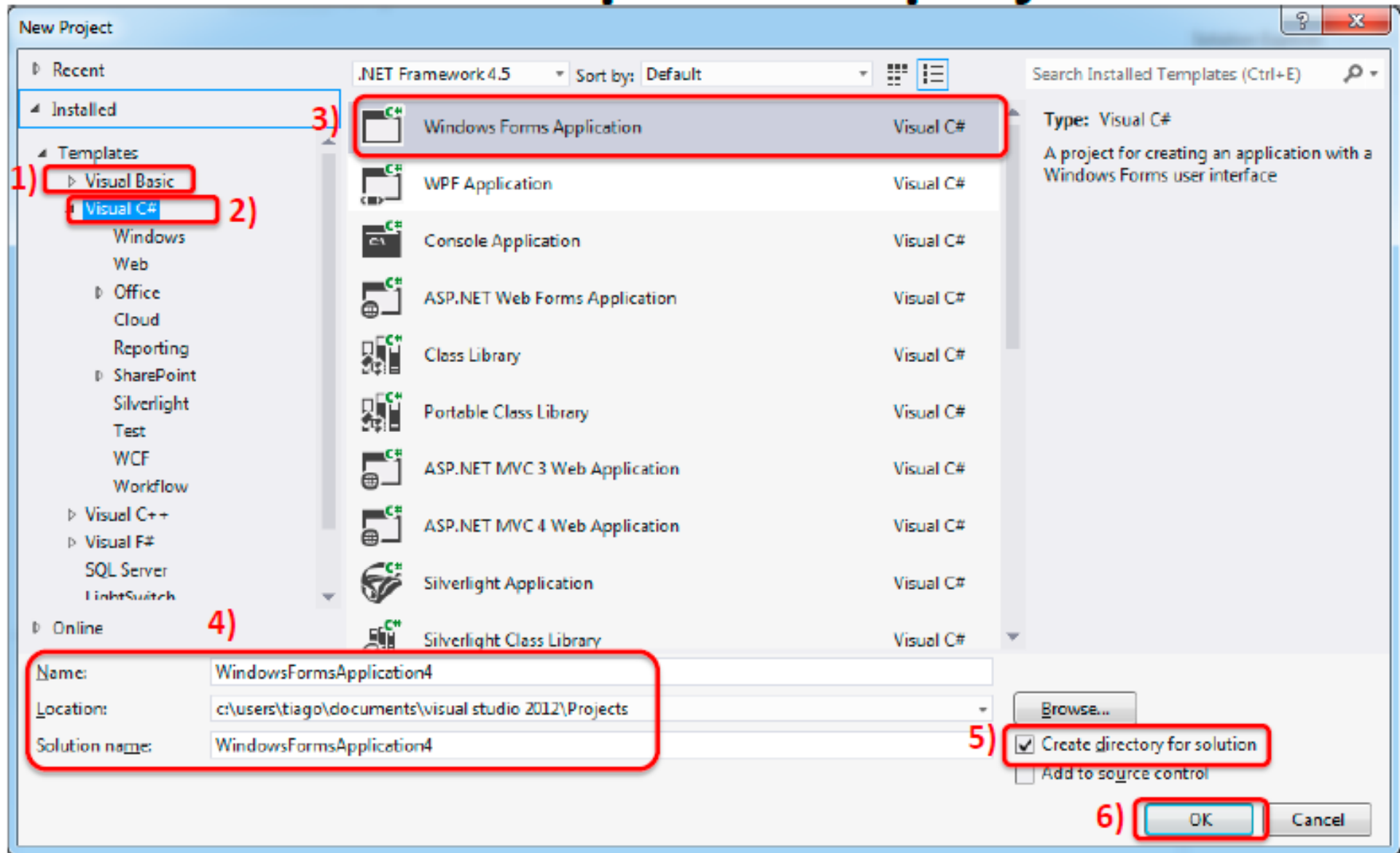


Criando o primeiro projeto

- Clique em *New Project...*



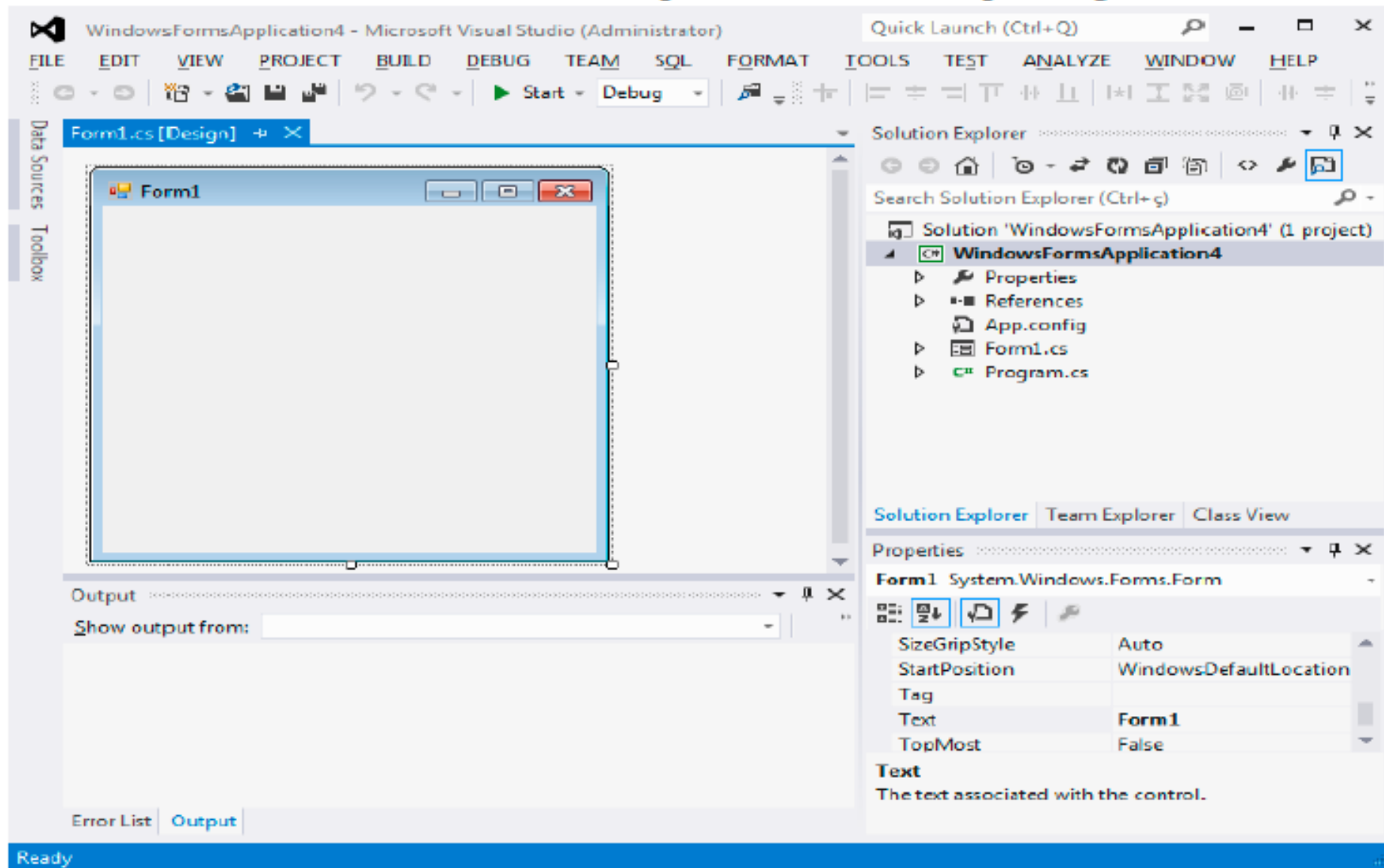
Criando o primeiro projeto



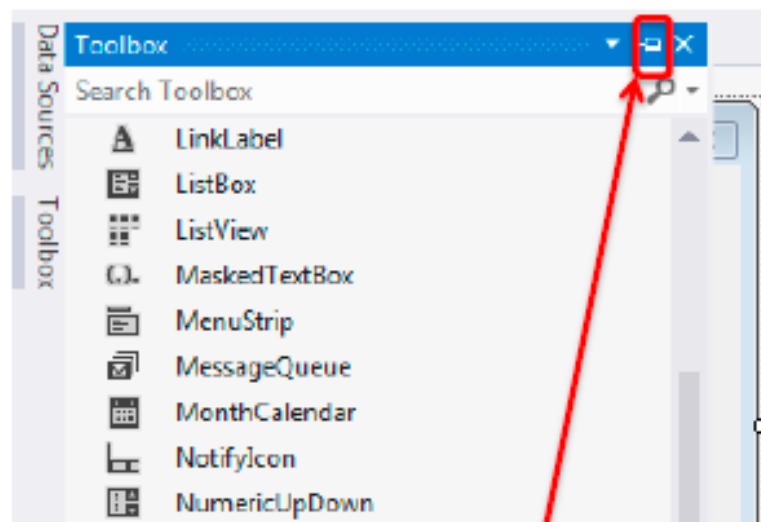
Criando o primeiro projeto

- 1) Selecione **Visual C#**
- 2) Selecione **Windows**
- 3) Selecione **Windows Forms Application**
- 4) Coloque um nome para o projeto e indique o local para salvá-lo.
- 5) Selecione **Create directory for solution** (Deixar selecionado para criação automática da estrutura do projeto).
- 6) Depois clique em **OK**

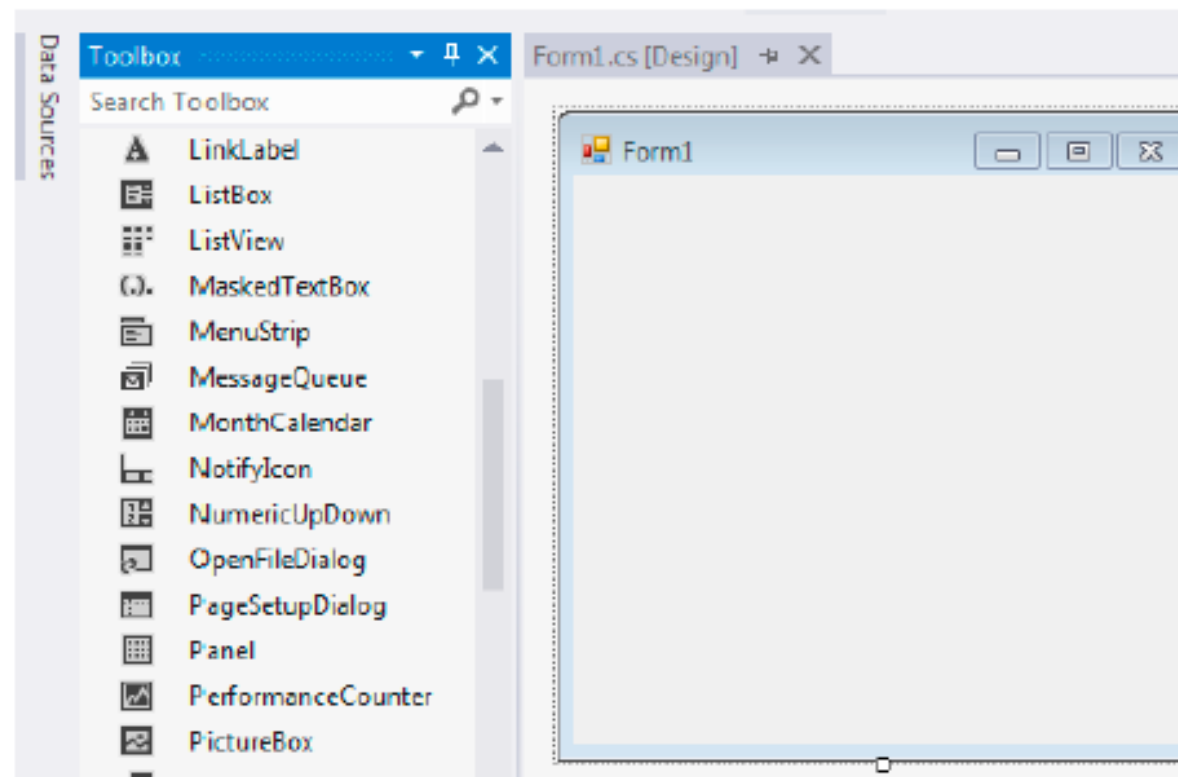
Ambiente do primeiro projeto



Exibir Caixa de Ferramentas

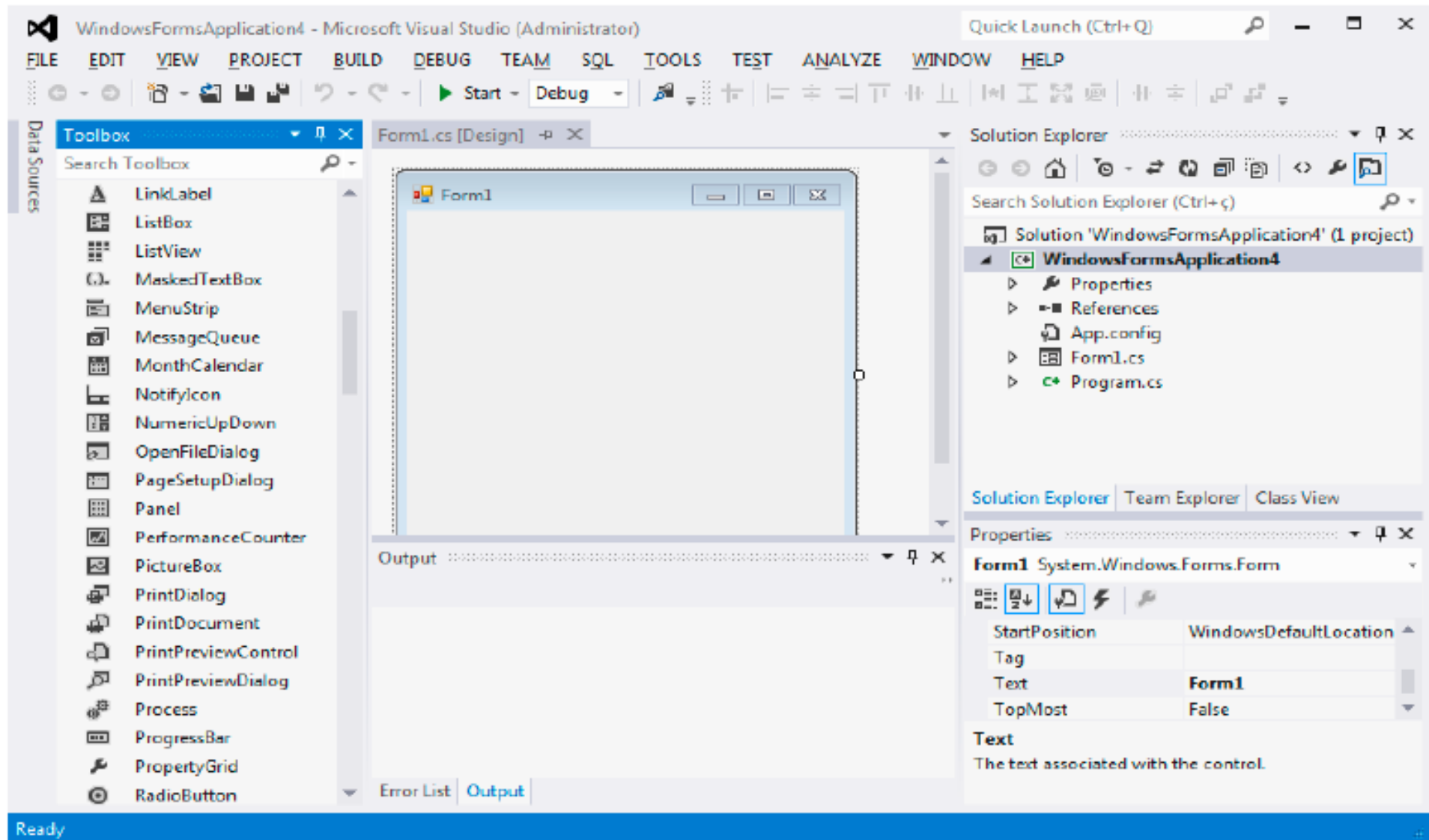


Clique em *auto hide*, para fixar a barra de ferramentas na janela



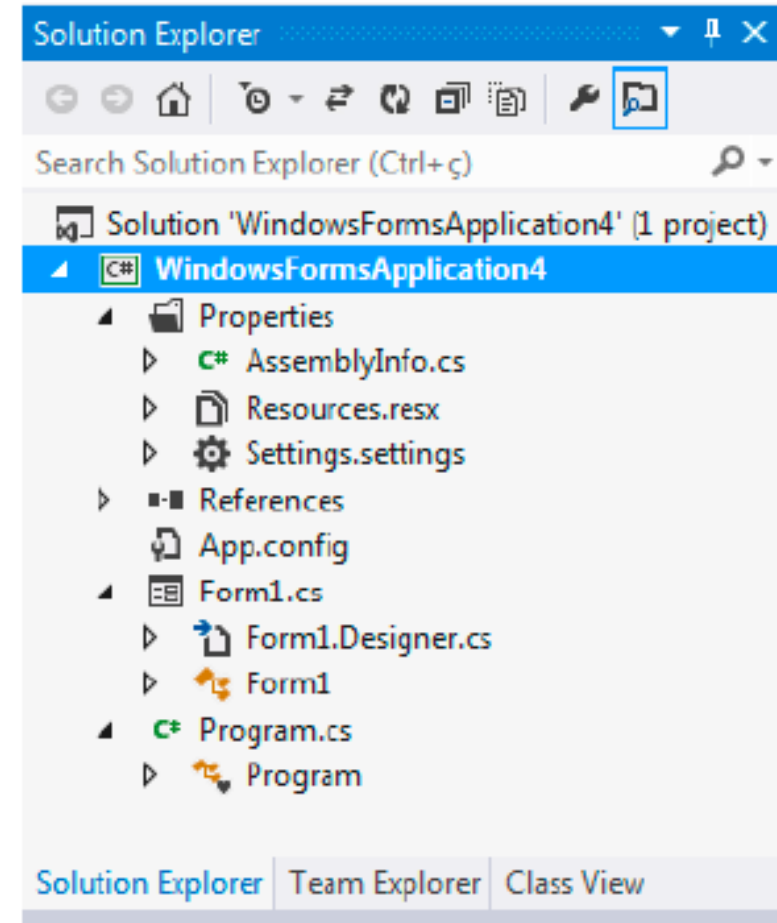
Logo após o clique, a barra de ferramentas ficará fixada na janela.

Identificando as Áreas do Ambiente



Solution Explorer

Esta janela mostra a estrutura de pastas e os arquivos que fazem parte do seu projeto. Seu comportamento é como o Explorer do Windows, permitindo criar, excluir e importar arquivos.

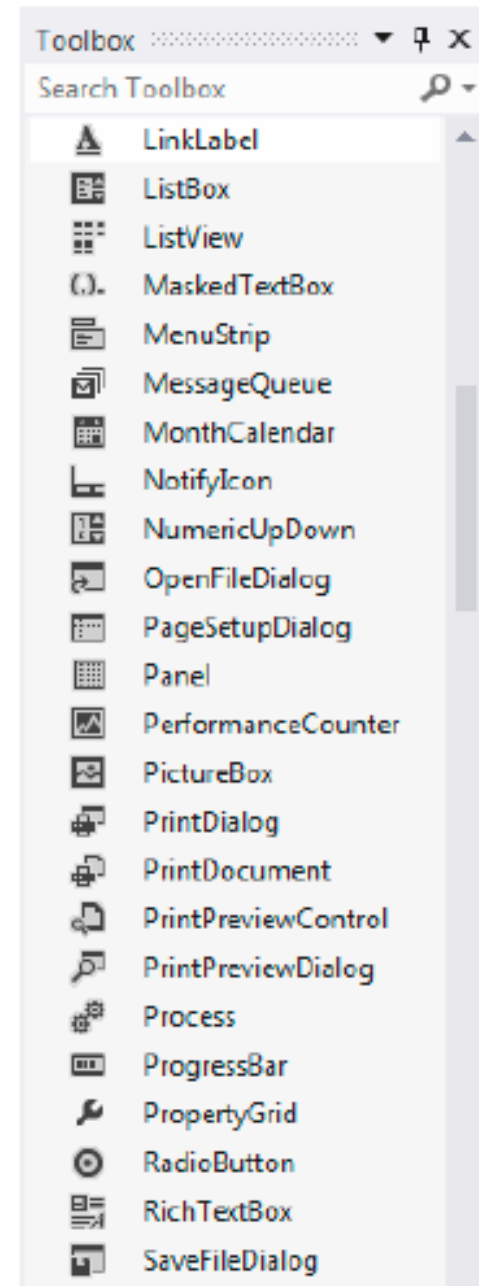


Solution Explorer

- O arquivo principal de uma aplicação é armazenado no disco como um arquivo do tipo “.sln”.
- Um projeto é armazenado em um arquivo do tipo “.csproj”.
- A seção Solution Explorer de um projeto possui quatro itens:
 - **Properties** (propriedades) – contém arquivos de configuração da solução e do projeto como o *AssemblyInfo.cs*, que define informações de configuração do projeto.
 - **References** (referências) – a lista de fragmentos de códigos compilados (assemblies) referenciados pelo projeto.
 - **Form1.cs** – um arquivo contendo a classe Form criada por padrão para a aplicação.
 - **Program.cs** – Um arquivo contendo a classe do programa criada por padrão para inicializar a aplicação.

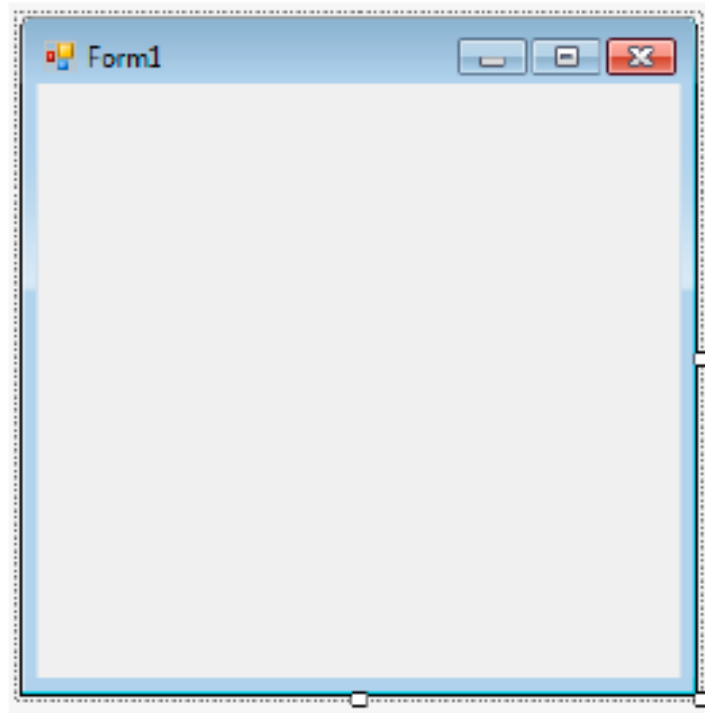
Toolbox

Esta janela contém os componentes necessários para o desenvolvimento de formulários.



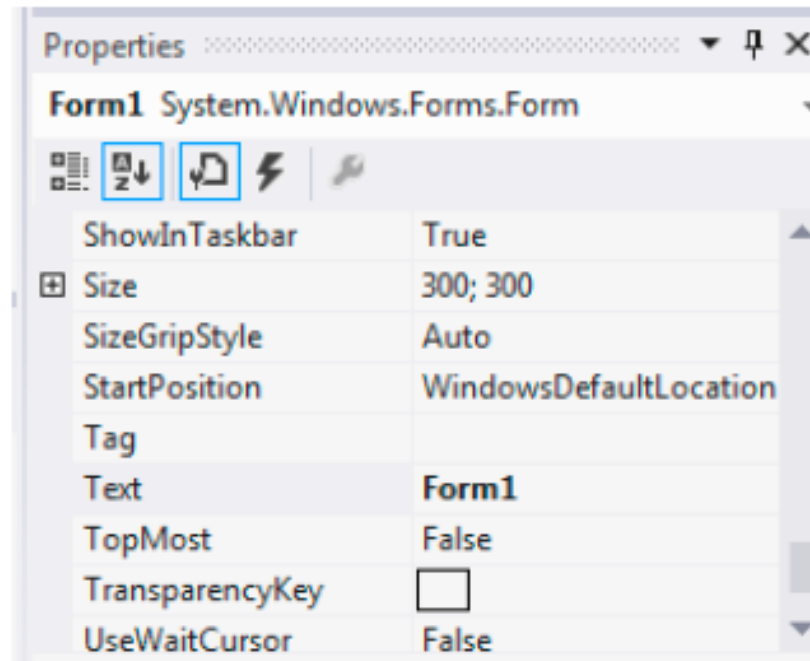
Form

Esta janela é a nossa aplicação, a qual receberá a programação e os componentes da *toolbox*.



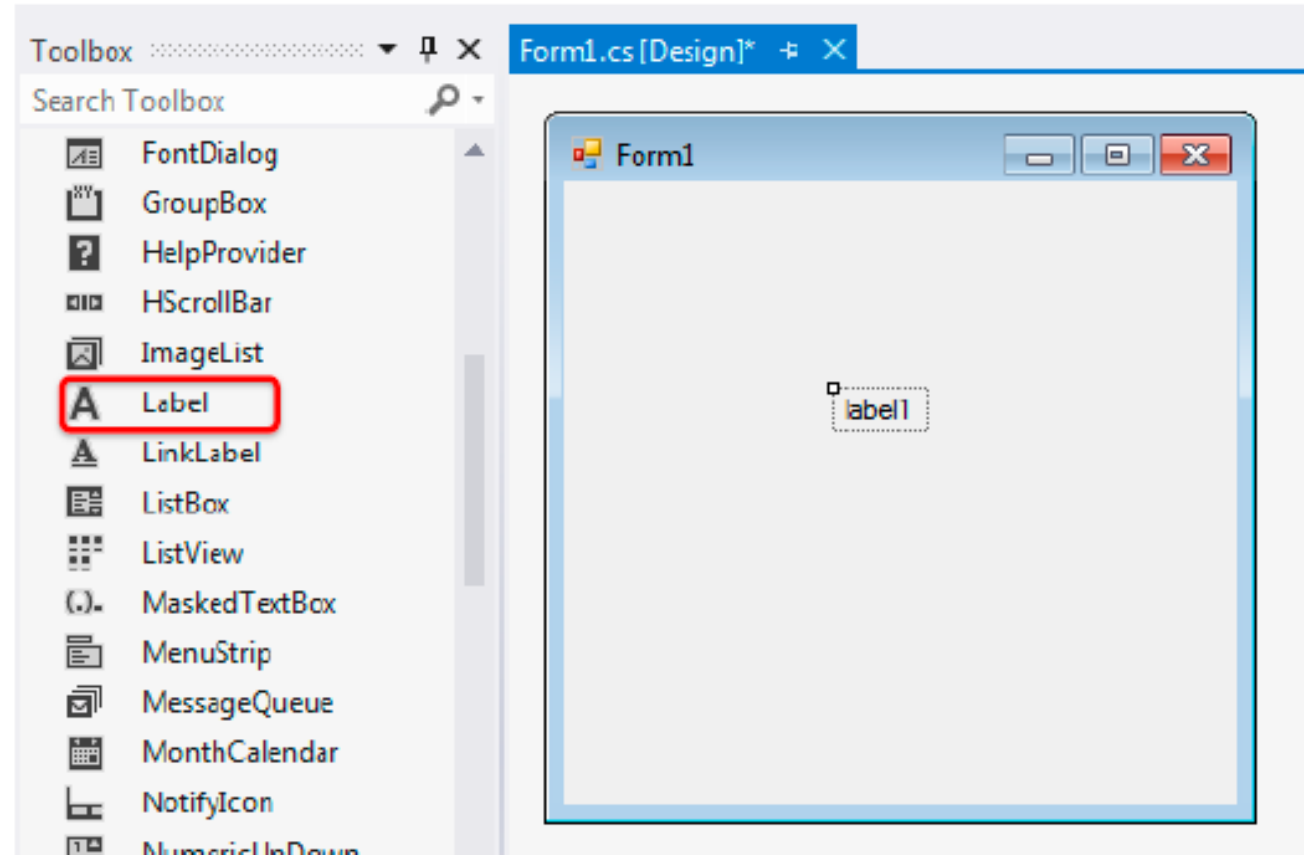
Properties

Esta janela permite alterar as propriedades dos componentes, as quais poderão estar organizadas por categoria, ordem alfabética, propriedades ou eventos.

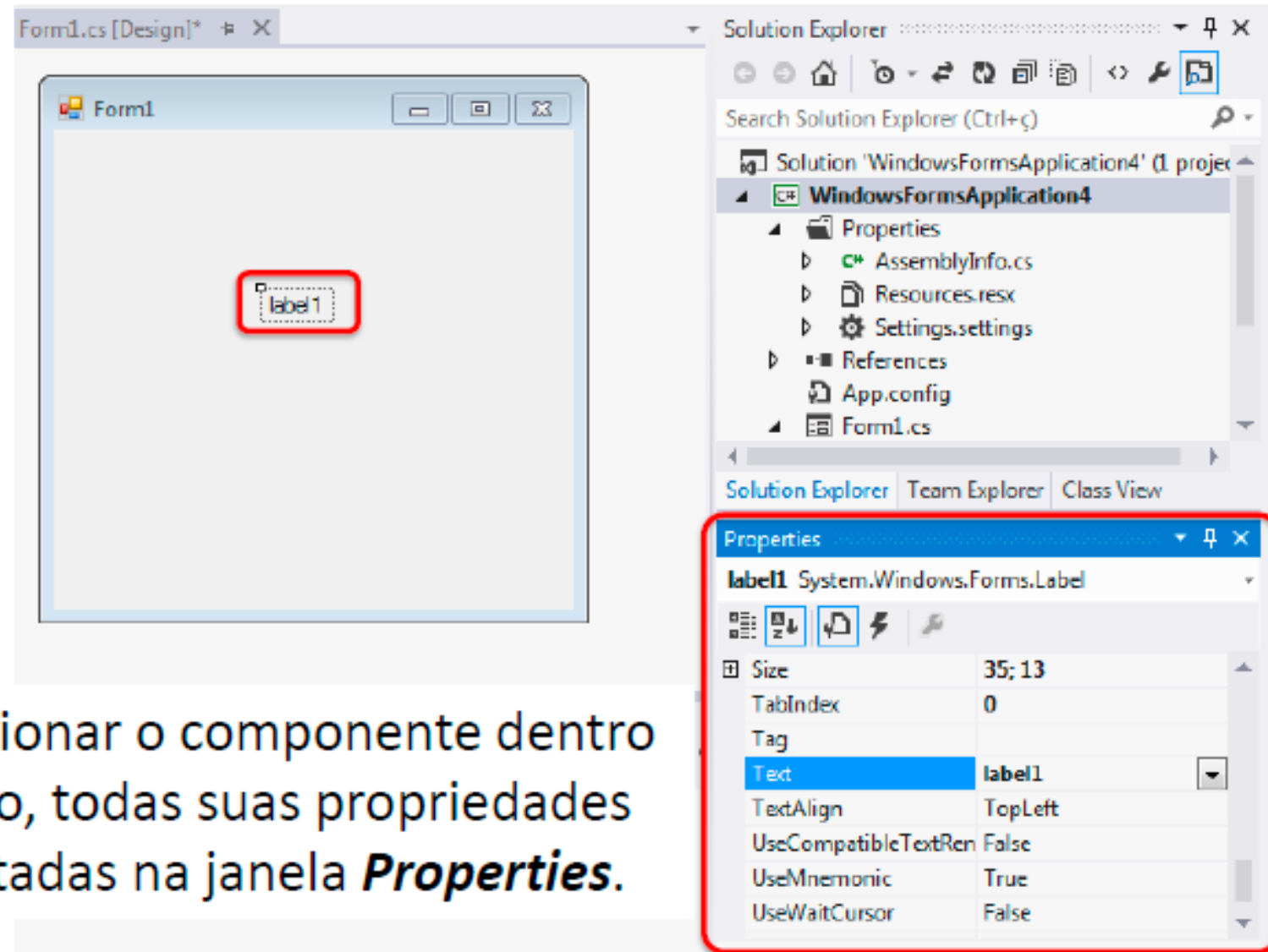


Inserindo um Componente

Para inserir um componente, basta selecionar algum na Caixa de Ferramentas (*Toolbox*) e depois clicar dentro do formulário (Form1). Ou um duplo clique para inserir diretamente. Neste exemplo foi escolhido o componente **Label**.

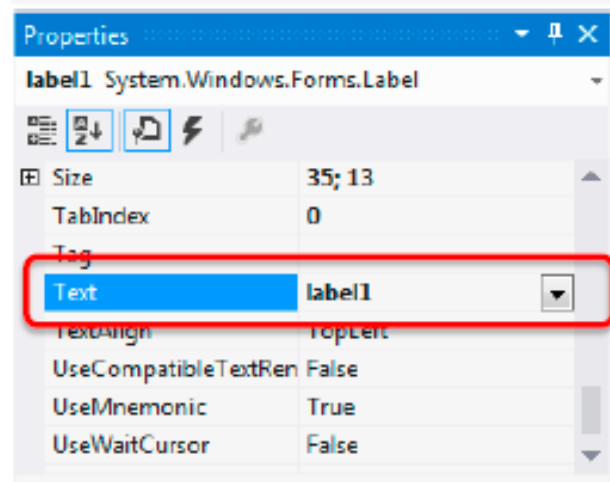
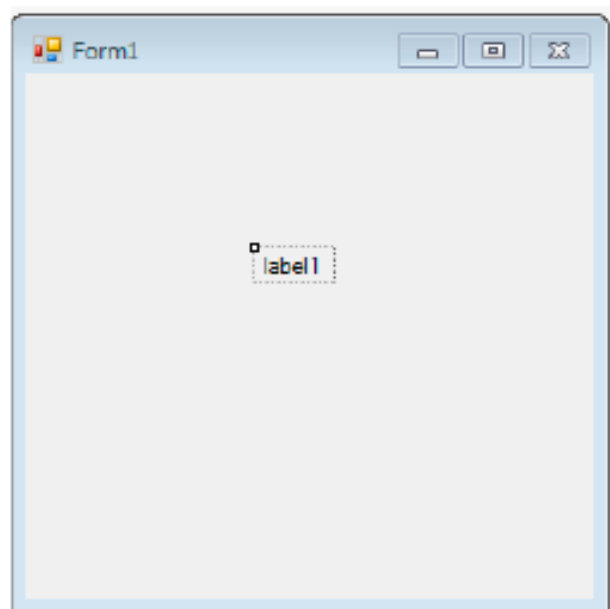


Propriedades do Componente



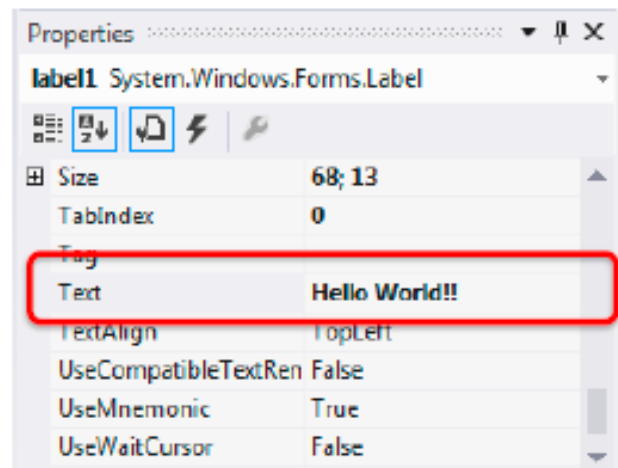
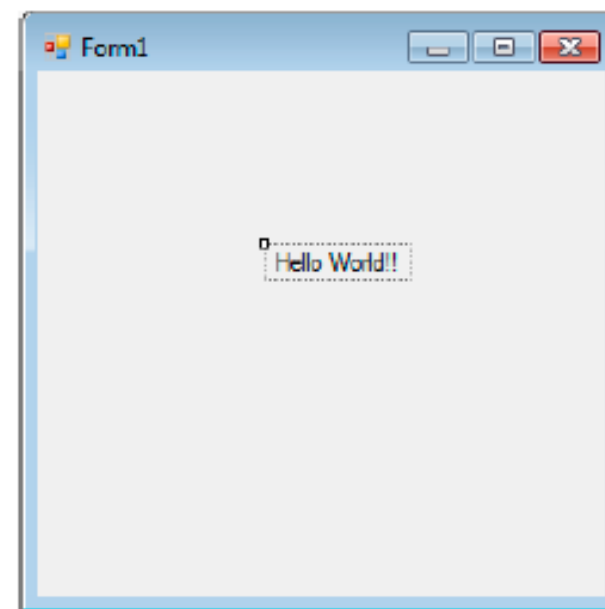
Quando selecionar o componente dentro do formulário, todas suas propriedades são apresentadas na janela **Properties**.

Alterando a propriedade *Text*



Vamos alterar o texto deste **Label**, colocando o famoso “Hello Word!!!”.

Para isso na caixa de propriedades basta alterar o campo **Text**, conforme indicado.



Algumas propriedades do *Label*

AutoSize – alterar para **False** permitindo que o tamanho do *Label* possa ser definido pelo programador.

(name) – para definir um nome para o componente, que visa facilitar a identificação durante a programação

BackColor – define a cor de fundo do componente

Font – define a fonte, estilo e tamanho do texto

ForeColor – define a cor de texto do componente

TextAlign – define o alinhamento do texto dentro do label

Image – permite incluir uma imagem no label

Alterando algumas propriedades

Alterar as seguintes propriedades com os respectivos valores:

AutoSize = False


(name) = minhaLabel





BackColor = selecione a cor Laranja

Font = Size = 20

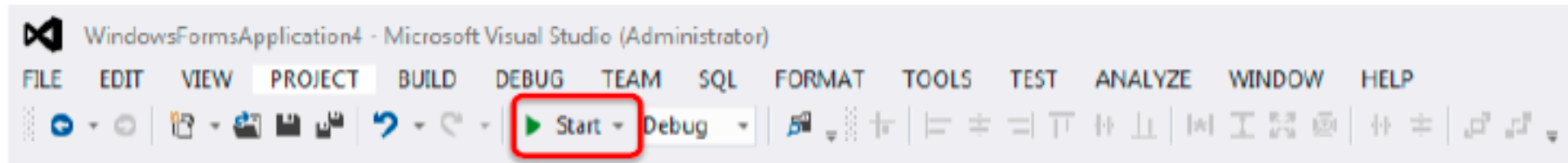
ForeColor = selecione a cor Preta

TextAlign = MiddleCenter

(Name)	minhaLabel
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	False
BackColor	 255; 128; 0
BorderStyle	None
CausesValidation	True
ContextMenuStrip	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
Font	Microsoft Sans Serif; 8,25
Name	ab Microsoft Sans Serif
Size	8,25
Unit	Point
Bold	False
GdiCharSet	0
GdiVerticalFont	False

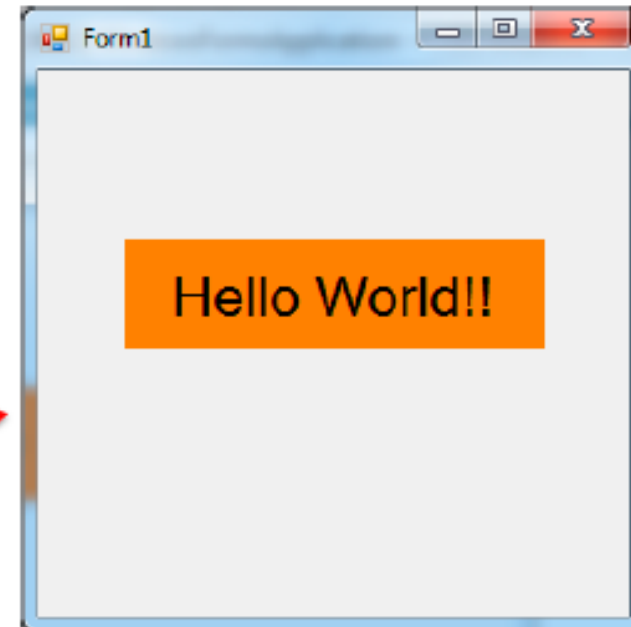
Underline	False
ForeColor	 ControlText
GenerateMember	True
Image	 (none)
ImageAlign	MiddleCenter
ImageIndex	 (none)
ImageKey	 (none)
ImageList	(none)
Location	116; 89
Locked	False
Margin	3; 0; 3; 0
MaximumSize	0; 0
MinimumSize	0; 0
Modifiers	Private
Padding	0; 0; 0; 0
RightToLeft	No
Size	100; 23
TabIndex	0
Tag	
Text	Hello World!!
TextAlign	MiddleCenter
UseCompatibleTextRendering	False
UseMnemonic	True
UseWaitCursor	False
Visible	True

Executando o Programa



Para que possamos executar o programa, basta clicar em *Start* na barra de ferramentas.

E se estiver tudo OK, será exibida a seguinte janela, contendo o formulário Form1.



Aprimorando o Hello World!!!

Para testar suas habilidades, modifique algumas propriedades deste formulário, como por exemplo, alterar a cor de fundo do formulário, o texto do formulário, a cor do texto da Label, o tipo de fonte do texto entre outras.

Na sequência adicione um botão, que servirá para finalizar a aplicação, como por exemplo.....



Evento no Botão

Clicando duas vezes no botão, abrirá a área de codificação do componente.

Nesta área iremos programar o evento (ação) **click** que servirá para finalizar a aplicação.

```
namespace WindowsFormsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

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

Evento no Botão

Dentro do evento Click, iremos colocar o comando
Application.Exit();
que será responsável por finalizar a aplicação.

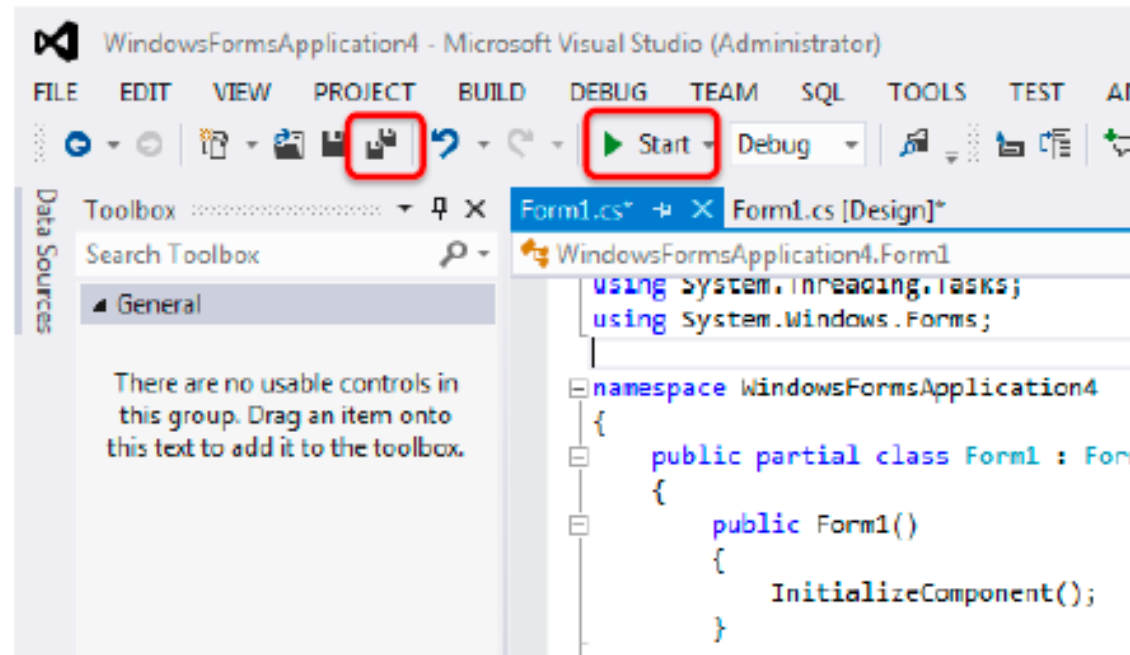
```
namespace WindowsFormsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Salvando e executando o programa

Clique em **Salvar todos**.

Em seguida clique em **Start** ou pressione a tecla F5 para executar. Dessa vez, quando clicar em **Sair** a aplicação será finalizada.



A propriedade (*name*)

A propriedade (*name*) dos componentes serve para determinarmos um nome específico queremos adotar como referência ao componente, para o tratamento de eventos e alteração de alguma propriedade através da codificação.

Não podemos definir um mesmo nome para mais de um componente.

Sendo esta a maneira de tornar cada componente, único na aplicação.

Mais Eventos

Vamos agora criar outro projeto, onde deverá contêr os seguintes componentes, de acordo com a imagem.

Label

Alterar as propriedades:

(name) = minhaLabel

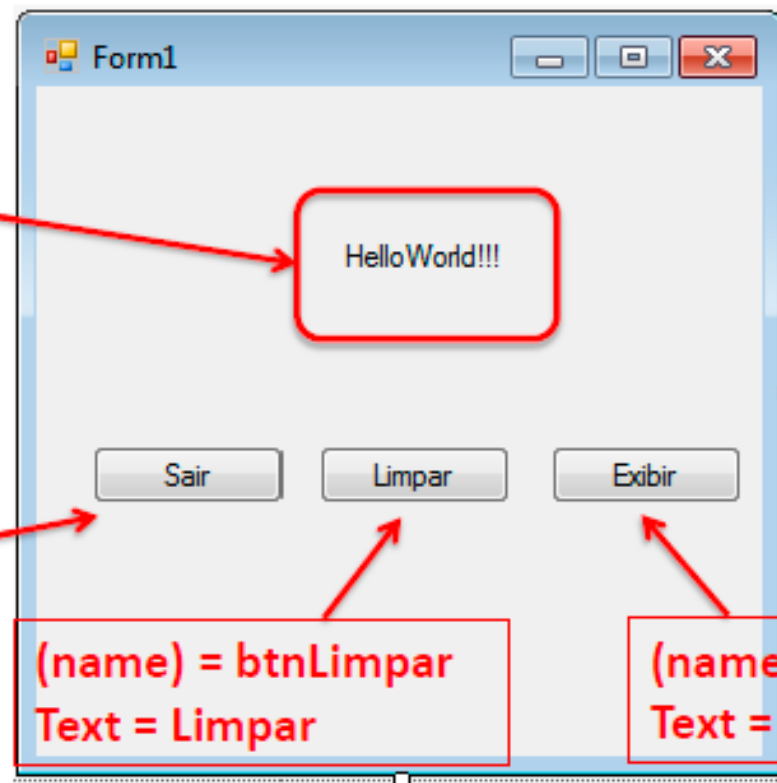
Text = Hello World!!!

Button

Alterar as propriedades:

(name) = btnSair

Text = Sair



(name) = btnLimpar
Text = Limpar

(name) = btnExibir
Text = Exibir

Aplicando Ações

As ações que serão aplicadas aos eventos dos respectivos botões são:

Sair: Deverá ser encerrada a aplicação

Limpar: Deverá limpar o conteúdo da *minhaLabel*

Exibir: Deverá exibir o texto “Meu primeiro programa!!!” na *minhaLabel*

Aplicando Ações

Dando um duplo clique em cada botão, abrirá a área de codificação para implementarmos as seguintes ações.

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

    private void btnSair_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void btnLimpar_Click(object sender, EventArgs e)
    {
        minhaLabel.Text = "";
    }

    private void btnExibir_Click(object sender, EventArgs e)
    {
        minhaLabel.Text = "Meu primeiro programa!!!";
    }
}
```