

Sumário

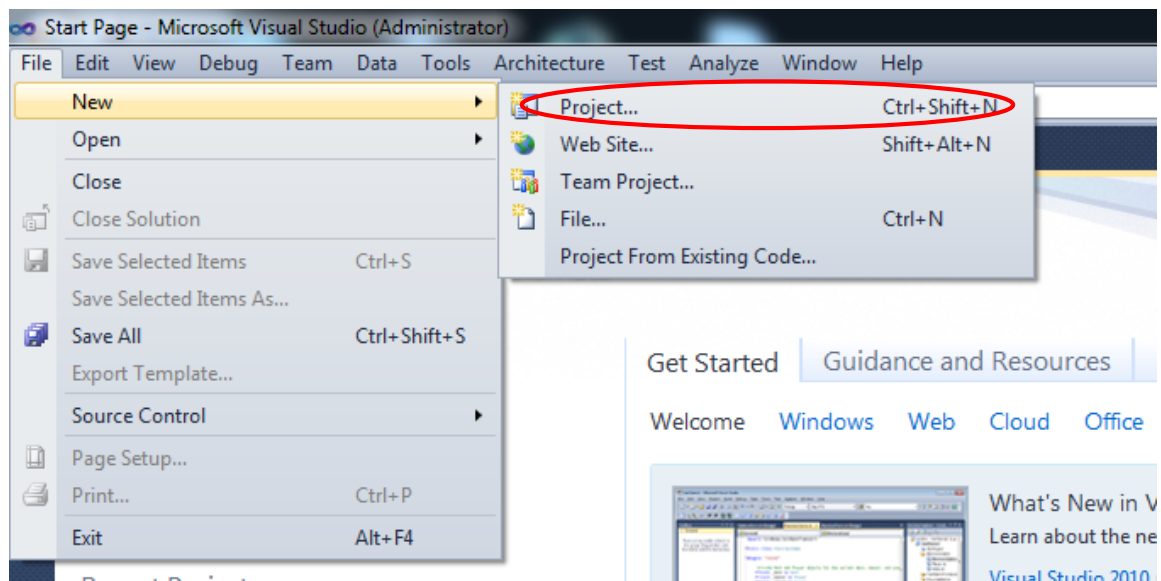
1. Introdução	2
Apresentação IDE e criação do primeiro projeto Windows Form	2
Exercícios Complementares em aula.	7
2. Caixas de diálogos em C# Windows Forms – MessageBox	8
3. Usando os controles radioButton, checkBox, listBox e comboBoxução	11
Exercícios Complementares em aula.	19
4. Objeto de controle MenuStrip	20
5. Criando aplicação com vários Forms.....	22
6. Variáveis Globais	24
Exercícios Complementares em aula.	27
7. Ligando C# com SGBD MS SQL Server	28
8. Criando Cadastros Básicos	35
9. Criando Consultas.....	43
10. Criando Relatórios.....	44

1. Introdução

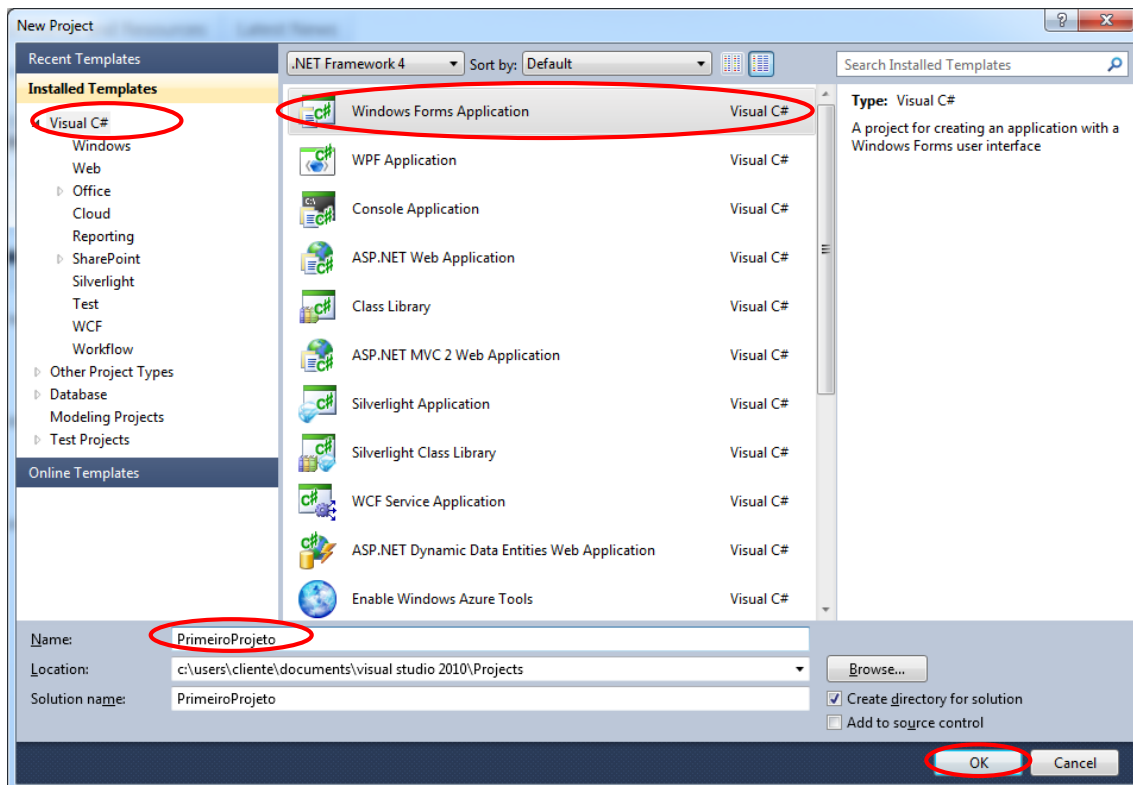
Na disciplina de Desenvolvimento de Software o aluno desenvolverá aplicações **C# Windows Form**, ou seja, aplicações com interface gráfica. A IDE que utilizaremos será o **Microsoft Visual Studio 2010**.

Apresentação IDE e criação do primeiro projeto Windows Form

- Abra o MS VS 2010 e Inicie um novo Projeto;



- Escolha Windows Forms Application;
- Informe o **name** do projeto e clique em OK;

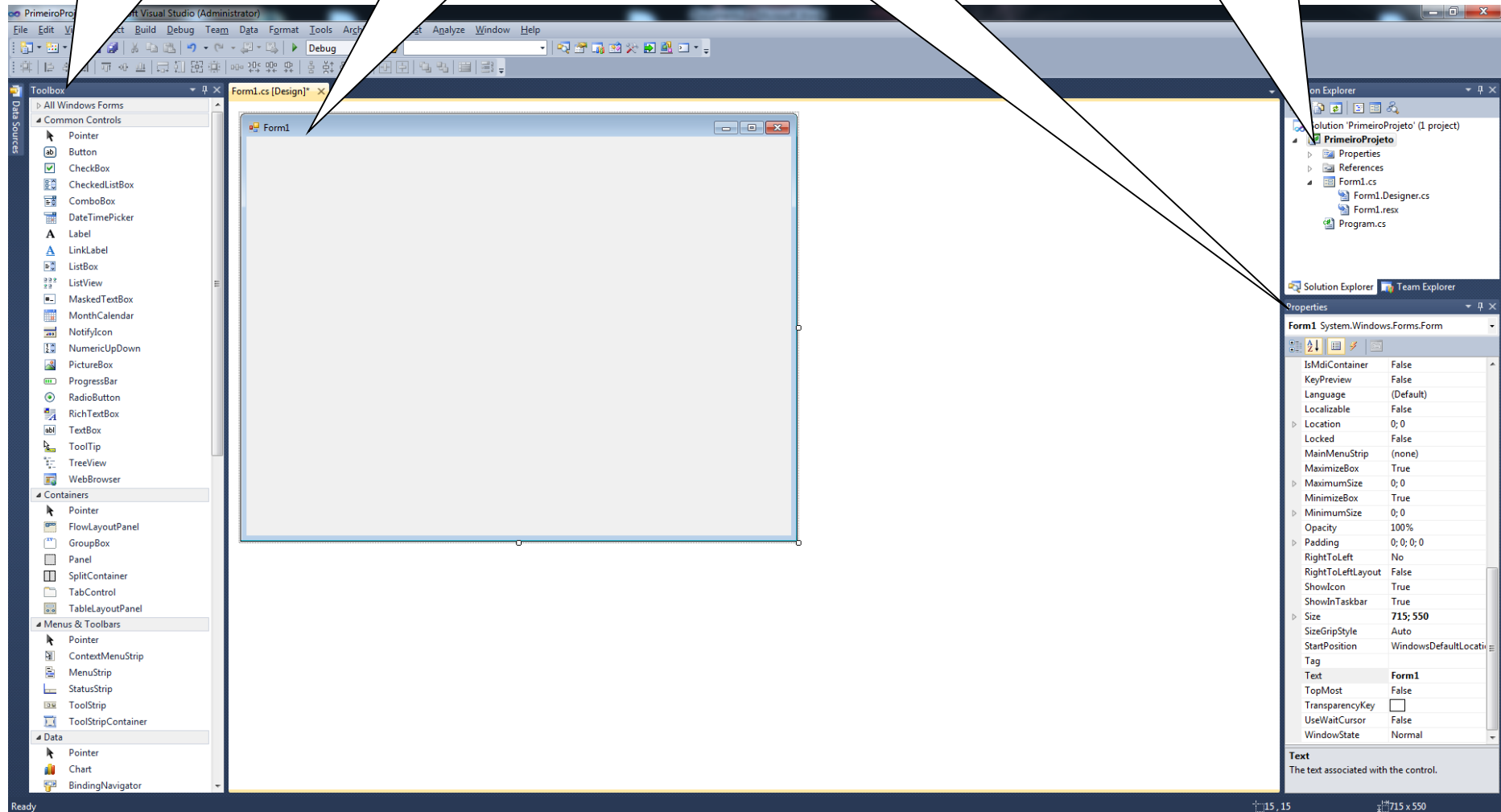


Aqui se encontram os **Objetos de controle**, como Caixas de Textos, Rótulos. Botões. etc.

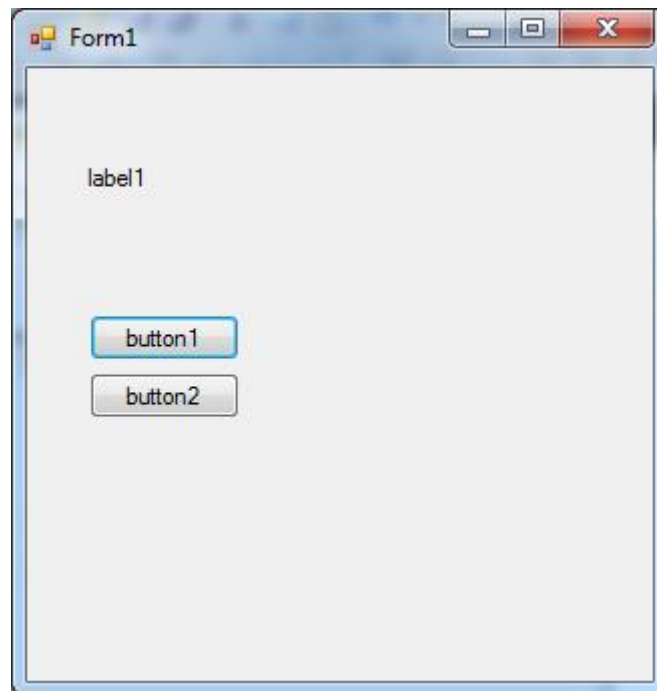
Este é o **Formulário**, onde será desenhada a Interface.

Aqui se encontram as **propriedades** (características) de cada objeto da interface.

Aqui se encontram os arquivos de todo o projeto, como formulários, imagens, sons, classes. etc.

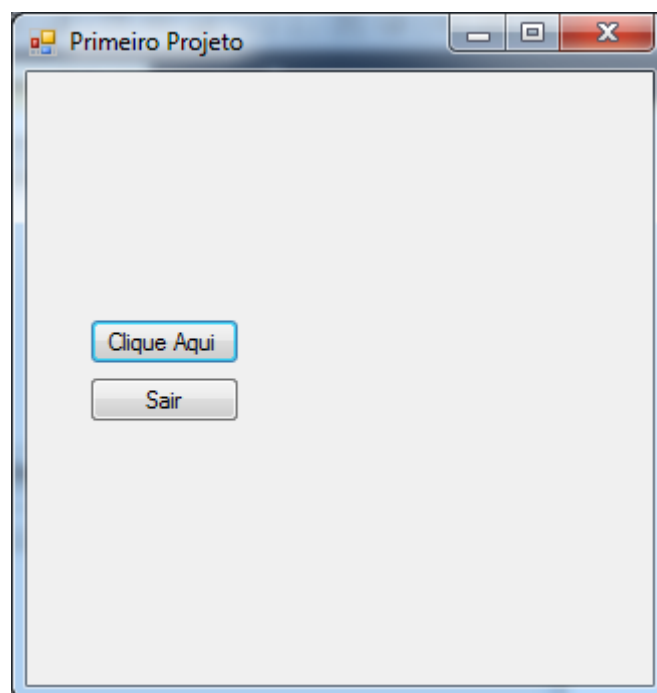


- Na guia **ToolBox** escolha e adicione um objeto **Label** e dois objetos **Button** ao Form1;



- Clique na label1 e na guia **Properties** apague a propriedade **Text**;
- Clique no button1 e na guia **Properties** altera a propriedade **Text** para **Clique Aqui**;
- Clique no button2 e na guia **Properties** altera a propriedade **Text** para **Sair**;
- Clique no Form1 e na guia **Properties** altera a propriedade **Text** para **Primeiro Projeto**;

O formulário ficará assim:



- Dê dois Cliques no button1 e o código fonte será exibido:

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

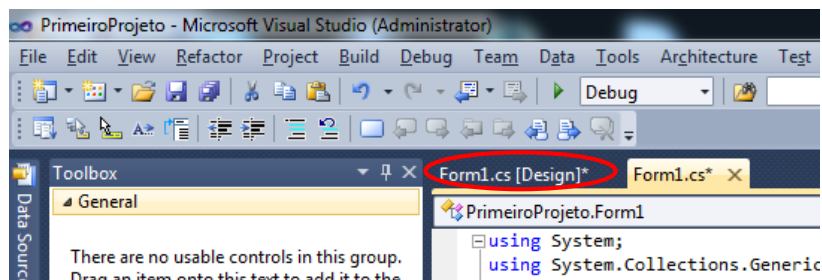
```

Observe que o button1 recebeu o sufixo **_Click**, isso significa que a programação que for inserida será executada quando o **evento** Click for acionado.

- Insira o seguinte código dentro do button1_Click:

```
label11.Text = "Olá, Você acaba de clicar no botão";
```

- Volte no form1 clicando na guia Form1.cs [Design]



- Dê dois Cliques no button2;
- Insira o seguinte código dentro do button2_Click:

```
this.Close();
```

O código fonte ficará assim:

```
private void button1_Click(object sender, EventArgs e)
{
    label11.Text = "Olá, Você acaba de clicar no botão";
}

private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

```

- Agora pressione a tecla **F5** para executar e testar o seu projeto;

Agora acesse o canal www.youtube.com/MestreCarlosVA e assista a **Vídeo Aula 01** e **Vídeo Aula 02**.

Exercícios Complementares em aula.

[illegible]

2. Caixas de diálogos em C# Windows Forms – MessageBox

Caixas de diálogos são partes vitais em aplicações do Windows. Um diálogo típico comum fornece uma funcionalidade interna para fora-da-caixa semelhante aos diálogos comuns encontradas no sistema operacional Windows. Conhecer a estrutura das caixas de diálogos é fundamental para o desenvolvimento de aplicações Windows, pois o uso correto deste recurso possibilita uma forma de comunicação com o usuário.

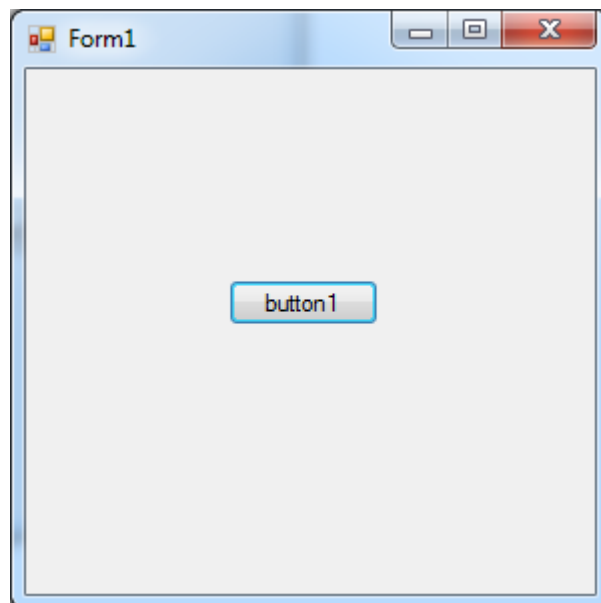
Neste capítulo você aprenderá como usar a caixa de diálogos **MessageBox** que normalmente é disparada por um evento da aplicação exibindo uma mensagem com o texto especificado. Uma caixa de mensagem pode ter algumas opções adicionais, incluindo uma legenda, ícone e botões de ajuda.

A classe MessageBox possui um método Show sobrecarregado estático que é usado para exibir uma mensagem. Aqui estão a maioria das formas de MessageBox.

MessageBox simples

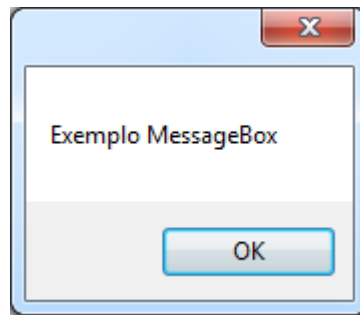
A forma mais simples de um MessageBox é um diálogo com um botão de texto e OK. A seguir criaremos um MessageBox simples.

1- Crie um projeto no Visual Studio 2010 do tipo WindowsFormsApplication e arraste um Button para o Formulário, em seguida dê um duplo clique sobre o botão e codifique.



```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Exemplo MessageBox");
}
```


2-Execute a aplicação e clique no botão para visualizar a MessageBox



3-Para a execução da aplicação;

Para personalizar a MessageBox com um título basta informar o título desejado como segundo parâmetro para o método Show, ex:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Exemplo MessageBox", "Título");
}
```



MessageBox com botões

Um MessageBox pode ter diferentes tipos de combinação de botões, como **Yes, No** ou **OK, Cancel**. A enumeração **MessageBoxButtons** representa os botões a serem exibidos em uma **MessageBox** e tem os seguintes valores:

- OK
- OKCancel
- AbortRetryIgnore
- YesNoCancel
- YesNo
- RetryCancel

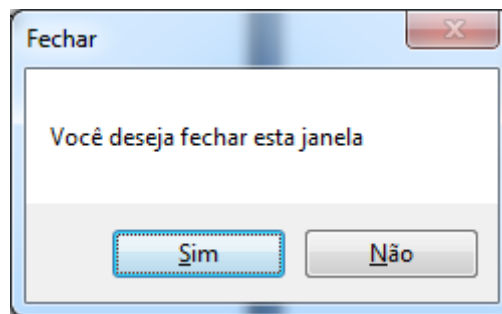
Para exemplificar vamos alterar o nosso código para usar a enumeração **MessageBoxButtons**

4-Altere o código do button1 para:

```
private void button1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Você deseja fechar esta janela", "Fechar", MessageBoxButtons.YesNo)
        == DialogResult.Yes)
    {
        this.Close();
    }
    else
    {
        //Faz qualquer outra coisa
    }
}
```

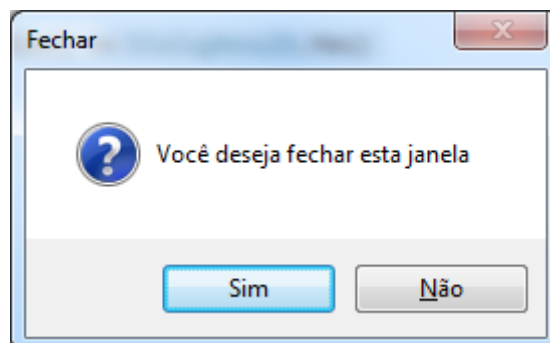
O trecho de código cria um MessageBox com um título e botões Sim e Não. Este é um típico MessageBox você pode chamar quando você quiser fechar uma aplicação. Se o botão Sim é clicado, o aplicativo será fechado

5-Teste a aplicação



NOTA1: Para inserir um ícone na MessageBox, basta inserir a enumeração **MessageBoxIcon** no último parâmetro. Ex:

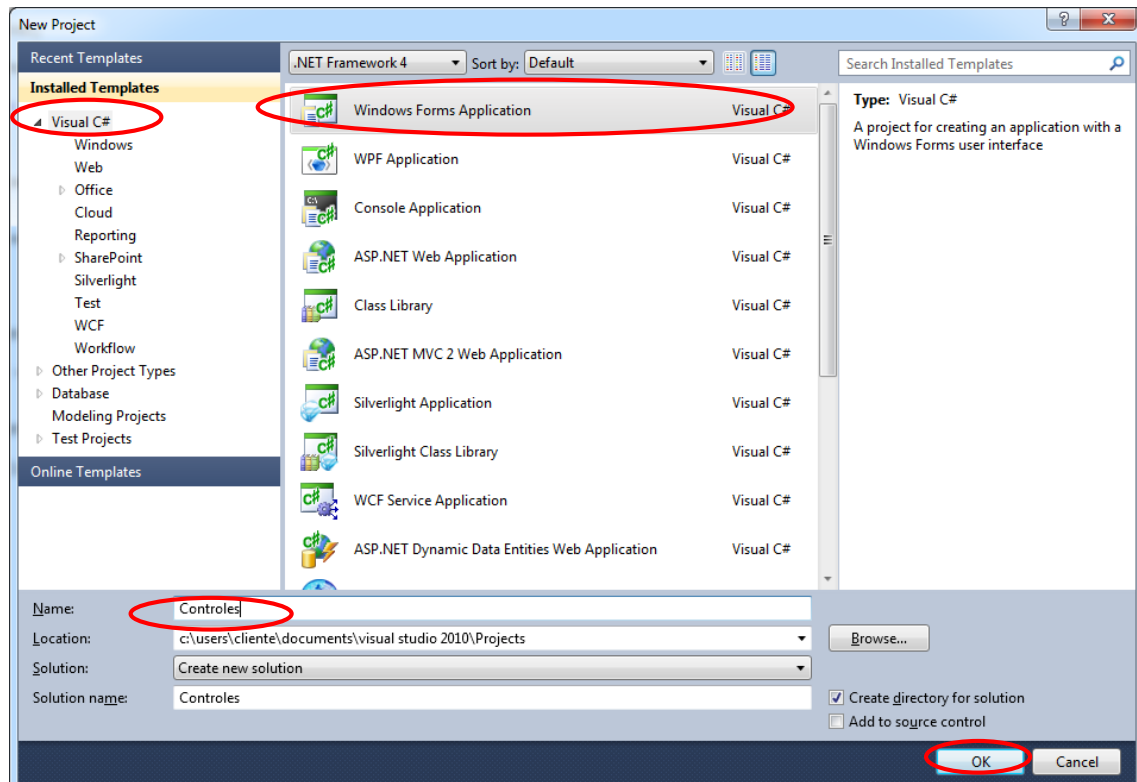
```
private void button1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Você deseja fechar esta janela", "Fechar",
        MessageBoxIcon.Question, MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        this.Close();
    }
    else
    {
        //Faz qualquer outra coisa
    }
}
```



3. Usando os controles radioButton, checkBox, listBox e comboBoxução

Vamos compreender agora como utilizar os controles RadioButton, CheckBox, ListBox e ComboBox para receber informações dos usuários. Para isso vamos criar uma simples aplicação que usa os quatro tipos de controles para você compreender quais as diferenças entre eles, quando e como aplicá-los.

1 - Crie um novo projeto do tipo Windows Forms Application chamado Controles.



2 - Arraste para o Form1 os seguintes controles:

- 2 GroupBox
- 3 RadioButton
- 3 CheckBox
- 1 ListBox
- 1 ComboBox
- 8 Label

3 - Organize-os como a figura:

4 - Mude a propriedade **Text** do **GroupBox1** para *Computadores (apenas um)*.

5 - Mude a propriedade **Text** do **GroupBox2** para *Escritório (0-3)*.

6 - Mude a propriedade **Text** do **RadioButton1** para *PC*.

7 - Mude a propriedade **Text** do **RadioButton2** para *MAC*.

8 - Mude a propriedade **Text** do **RadioButton3** para *Notebook*.

9 - Mude a propriedade **Text** do **CheckBox1** para *FAX*.

10 - Mude a propriedade **Text** do **CheckBox2** para *Calculadora*.

11 - Mude a propriedade **Text** do **CheckBox3** para *Copiadora*.

12 - Mude a propriedade **Text** do **Label1** para *Periféricos (apenas um)*.

13 - Mude a propriedade **Text** do **Label2** para *Produtos escolhidos*.

14 - Mude a propriedade **Text** dos Labels 3 a 8 para "vazio".

Deve ficar como a seguinte imagem:

The screenshot shows a Windows Form titled "Form1". It contains two main sections of controls. The first section, "Computadores (apenas um)", contains three radio buttons labeled "PC", "MAC", and "NoteBook". The second section, "Escritório (0-3)", contains three checkboxes labeled "FAX", "Calculadora", and "Copiadora". To the right of these, there is a section labeled "Periféricos (apenas um)" which contains a "listBox1" and a dropdown menu. The text "Produtos escolhidos" is visible on the right side of the form.

Os 3 **RadioButton** devem ficar dentro do **GroupBox1**, assim como os 3 **CheckBox** devem ficar dentro do **GroupBox2**, o **GroupBox** agrupa os controles.

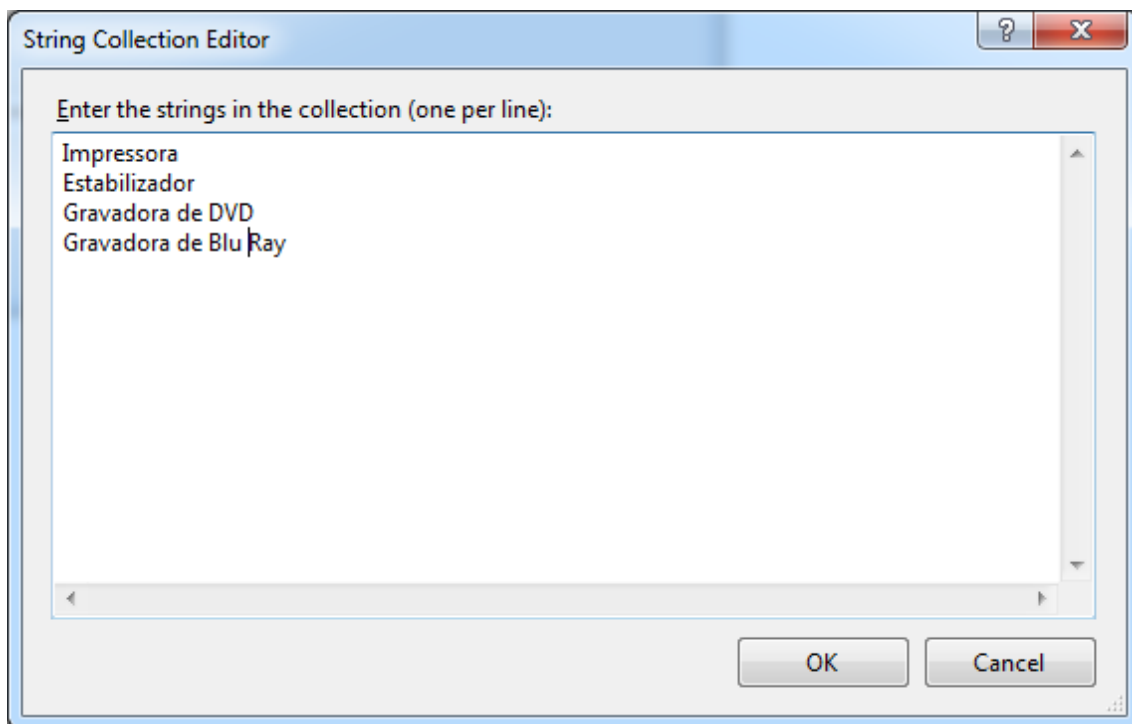
Vamos agora inserir valores na **ListBox1**.

15 - Clique na **ListBox1** e na janela **Properties**, localize a propriedade **Items**.

Clique no botão com as reticências nesta propriedade, deve abrir a seguinte caixa:

The screenshot shows the "String Collection Editor" dialog box. It has a title bar with a question mark and a close button. The main area contains the text "Enter the strings in the collection (one per line):" followed by a large text area for input. At the bottom, there are "OK" and "Cancel" buttons.

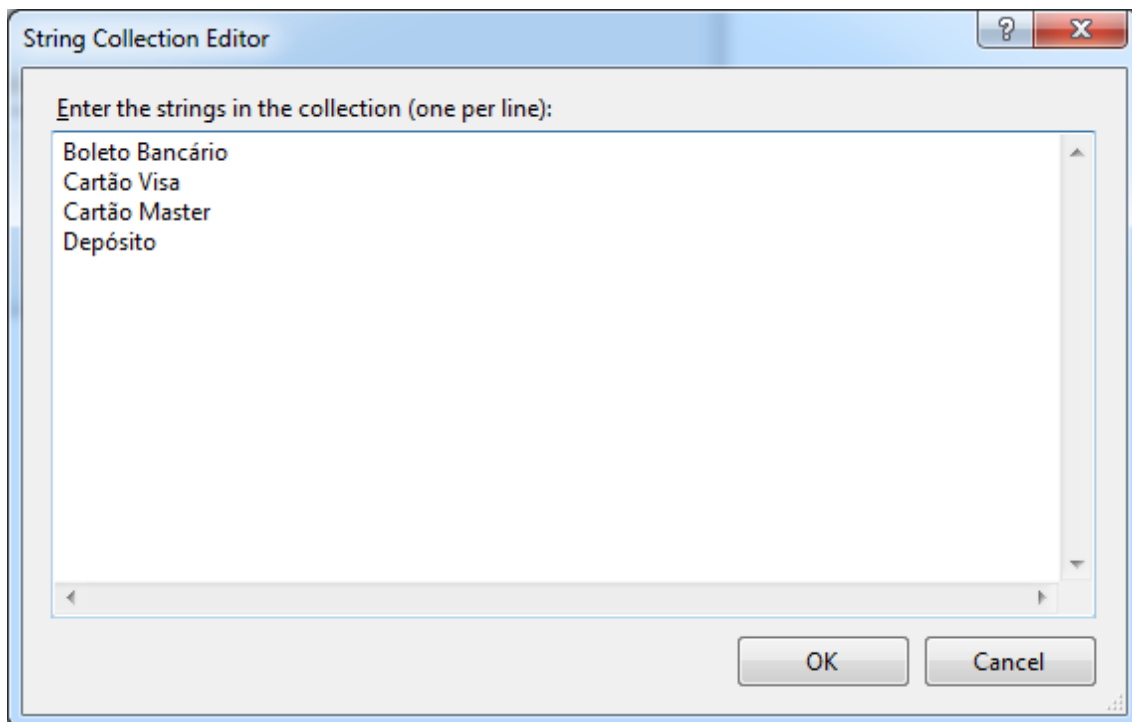
16 - Na janela **String Collection Editor** digite os items conforme a figura abaixo:



7 - Clique em OK.

Isso deve adicionar os items que digitamos na **ListBox1**.

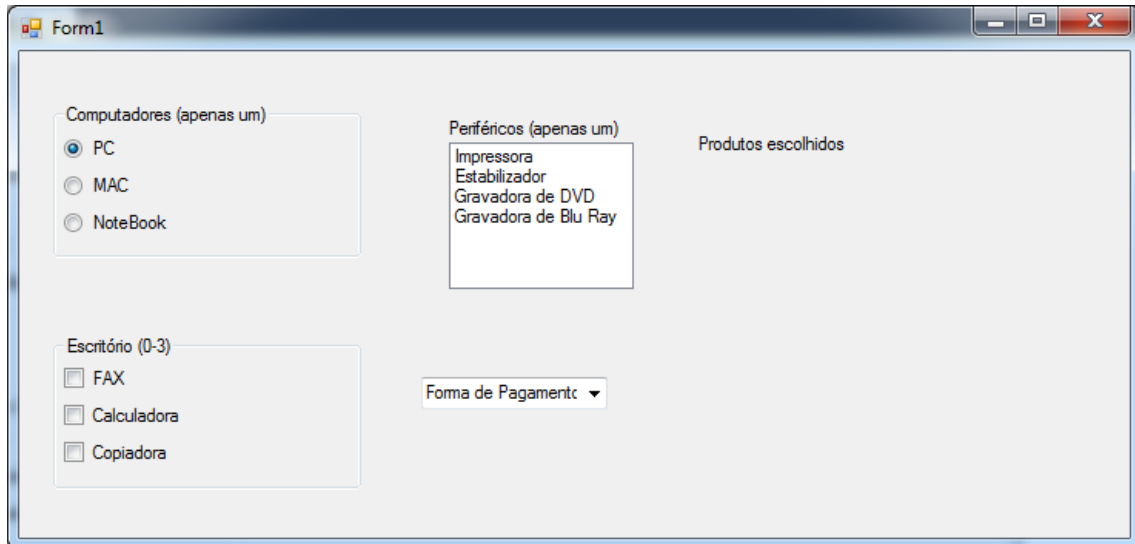
18 - Faça o mesmo para o **ComboBox1**, adicione os seguintes items:



19 - Mude a propriedade **Text** do **ComboBox1** para Forma de Pagamento.

20 - Execute a aplicação

Seu **Form1** deve estar semelhante a seguinte imagem:



21 - Pare a execução do programa.

A primeira funcionalidade que iremos implementar agora é a dos **RadioButton**s. Eles são usados sempre que o usuário precisa fazer uma **escolha única**, já que não permite que mais de um item seja marcado. Podemos também colocar uma escolha que já aparece marcada como padrão, para isso você deve mudar a propriedade **Checked** do **RadioButton** em questão para **True**.

22 - Mude a propriedade **Checked** do **RadioButton1** para **True**.

23 - Execute novamente a aplicação. Agora o item PC já aparece marcado.

24 - Pare novamente a execução do programa. Vamos agora programar o código para os **RadioButton**s.

25 - No modo Design, de um clique duplo sobre o **RadioButton1** e digite o seguinte código no procedimento de evento criado:

```
label3.Text = "PC";
```

26 - Faça o mesmo para o **RadioButton2** e **RadioButton3**, digitando as seguintes linhas de código respectivamente:

```
label3.Text = "MAC";
```

```
label3.Text = "Notebook";
```

Deve ficar assim:

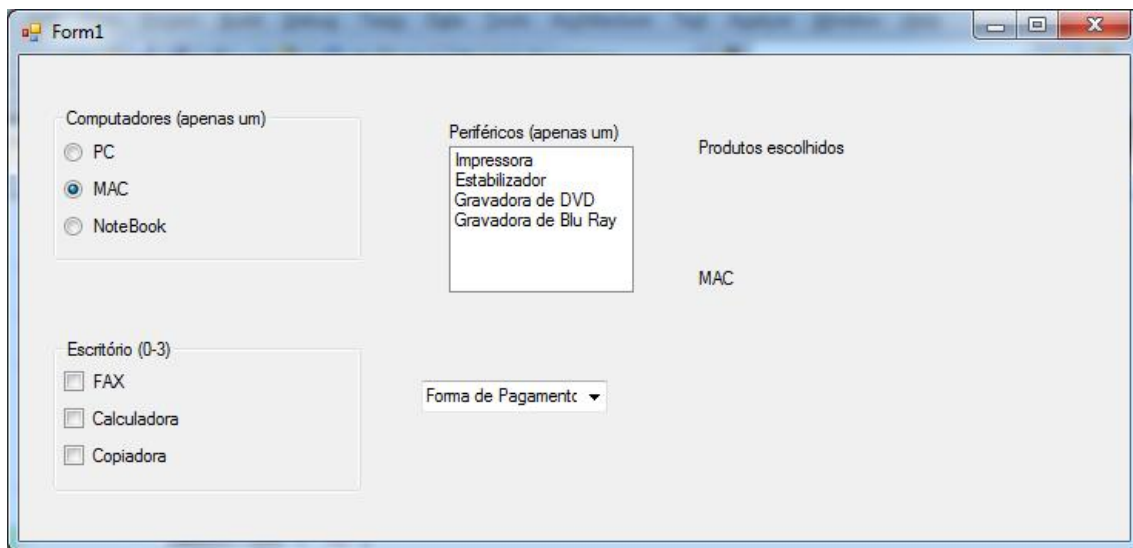
```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    label13.Text = "PC";
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    label13.Text = "MAC";
}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    label13.Text = "Notebook";
}
```

O evento padrão para o **RadioButton** é o **CheckedChanged**, que ocorre sempre que há uma mudança na propriedade **Checked**. Quando você clica nele o **Checked** fica como **True**, então o evento é disparado e muda a propriedade **Text** do **Label3**.

27 - Execute a aplicação e clique sobre as escolhas do **RadioButtons** para verificar a funcionalidade que acabamos de colocar.



28 - Pare a execução do programa.

Vamos agora programar o código para os **CheckBoxes**. Eles são semelhantes ao **RadioButtons**, no entanto permitem mais de uma escolha. Novamente você pode mudar a propriedade **Checked** de alguns deles para já iniciarem marcados. São usados sempre para perguntas que requerem nenhum ou várias respostas como no nosso exemplo. O usuário pode não querer nenhum item para o escritório, pode querer um, dois ou todos os itens.

29 - De um clique duplo sobre o **CheckBox1** e digite o seguinte código:

```
if (checkBox1.Checked == true)
{
    label6.Text = "Fax";
}
else
{
    label6.Text = "";
}
```

30 - Faça o mesmo para o **CheckBox2** e digite o seguinte código:

```
if (checkBox2.Checked == true)
{
    label7.Text = "Calculadora";
}
else
{
    label7.Text = "";
}
```

31 - Faça o mesmo para o **CheckBox3** e digite o seguinte código:

```
if (checkBox3.Checked == true)
{
    label8.Text = "Copiadora";
}
else
{
    label8.Text = "";
}
```

32 - Execute sua aplicação. Marque as opções FAX e Copiadora, por exemplo, como disse podemos marcar mais de um item.

Form1

Computadores (apenas um)

☒ PC

☐ MAC

☐ NoteBook

Escritório (0-3)

☒ FAX

☒ Calculadora

☒ Copiadora

Periféricos (apenas um)

Impressora

Estabilizador

Gravadora de DVD

Gravadora de Blu Ray

Forma de Pagamento ▼

Produtos escolhidos

PC

Fax

Calculadora

Copiadora

33 - Pare a execução do programa.

34 - De um clique duplo sobre o **ListBox1** e digite o seguinte código:

```
label4.Text = listBox1.SelectedItem.ToString();
```

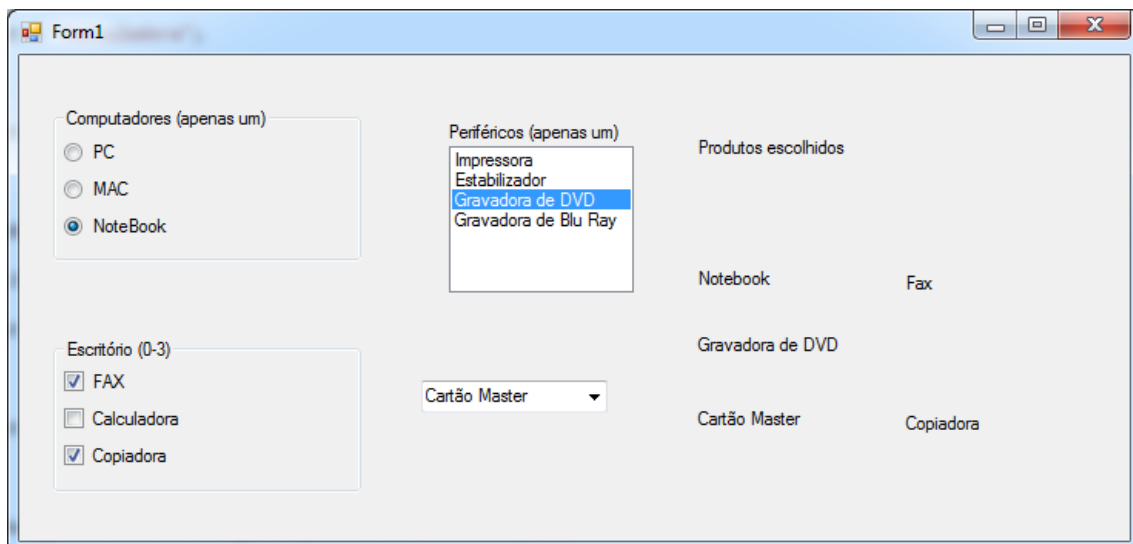
Isso escreve o conteúdo do item selecionado na **Label4**.

35 - Vamos já programar o código para a **ComboBox** também, de um clique duplo sobre a mesma no modo Design para abrir o procedimento de evento padrão e digite o seguinte código:

```
label5.Text = comboBox1.SelectedItem.ToString();
```

36 - Execute sua aplicação.

37 - Teste todas as escolhas.



Agora você já é capaz de pegar informações dos usuários de várias maneiras.

Procure utilizar sempre os controles aprendidos neste capítulo, eles impedem muitos erros nos programas por impedirem entradas de dados inválidas.

NOTA1: Quando for necessário utilizar a **ListBox** como método de entrada, o método que deve ser utilizado é o **listBox1.Items.Add**.

Ex:

```
private void button1_Click(object sender, EventArgs e)
{
    listBox1.Items.Add("Texto exemplo");
}
```

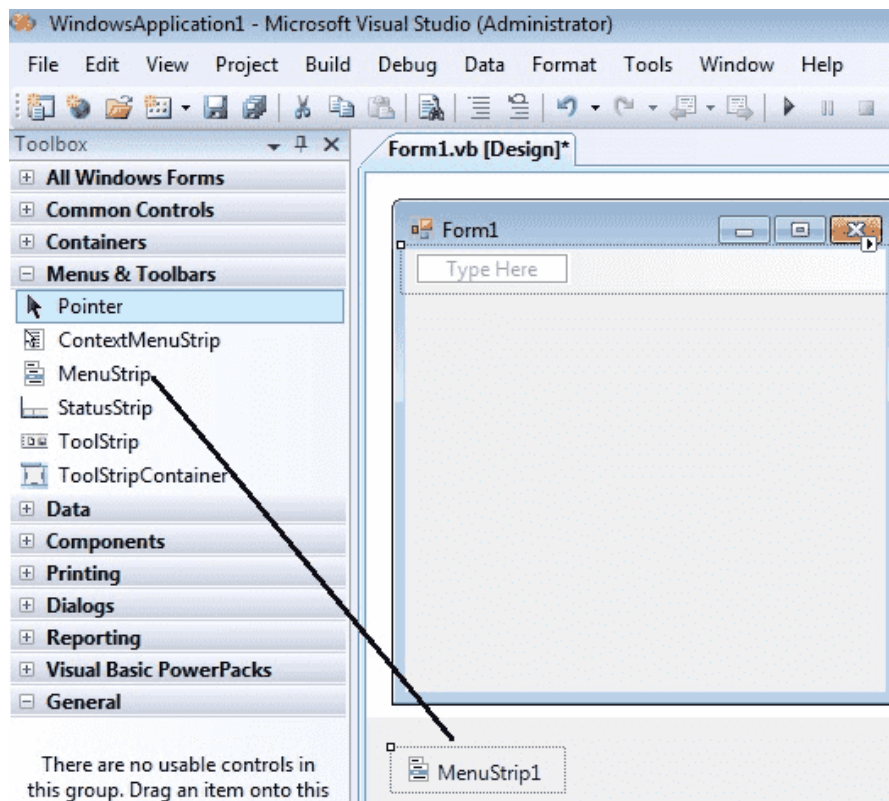
NOTA2: Para apagar o conteúdo de uma **listBox** o método é **ListBox1.Clear()**;

Exercícios Complementares em aula.

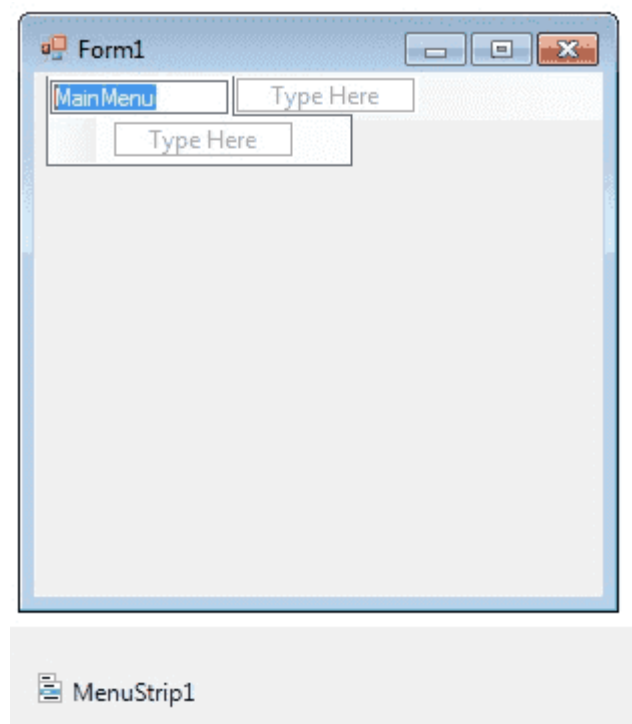
[illegible]

4. Objeto de controle MenuStrip

Para criar um menu, insira o objeto de controle **MenuStrip** no formulário

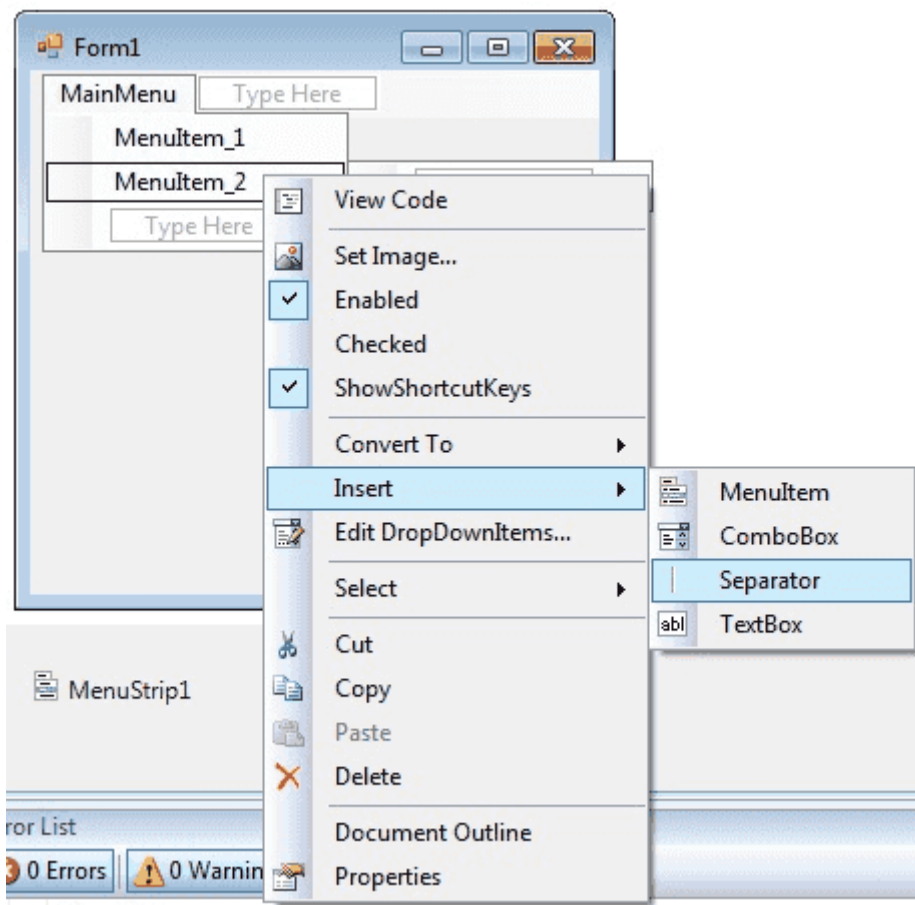


Depois basta digitar o nome dos menus, e submenus em “Type Here”



Se necessário, insira um separador, clicando com o botão direito e depois em

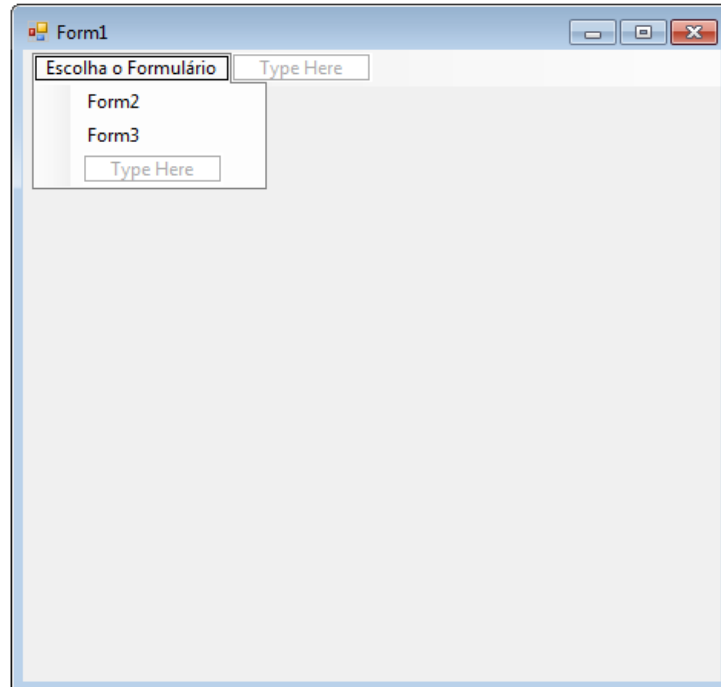
Insert > Separator



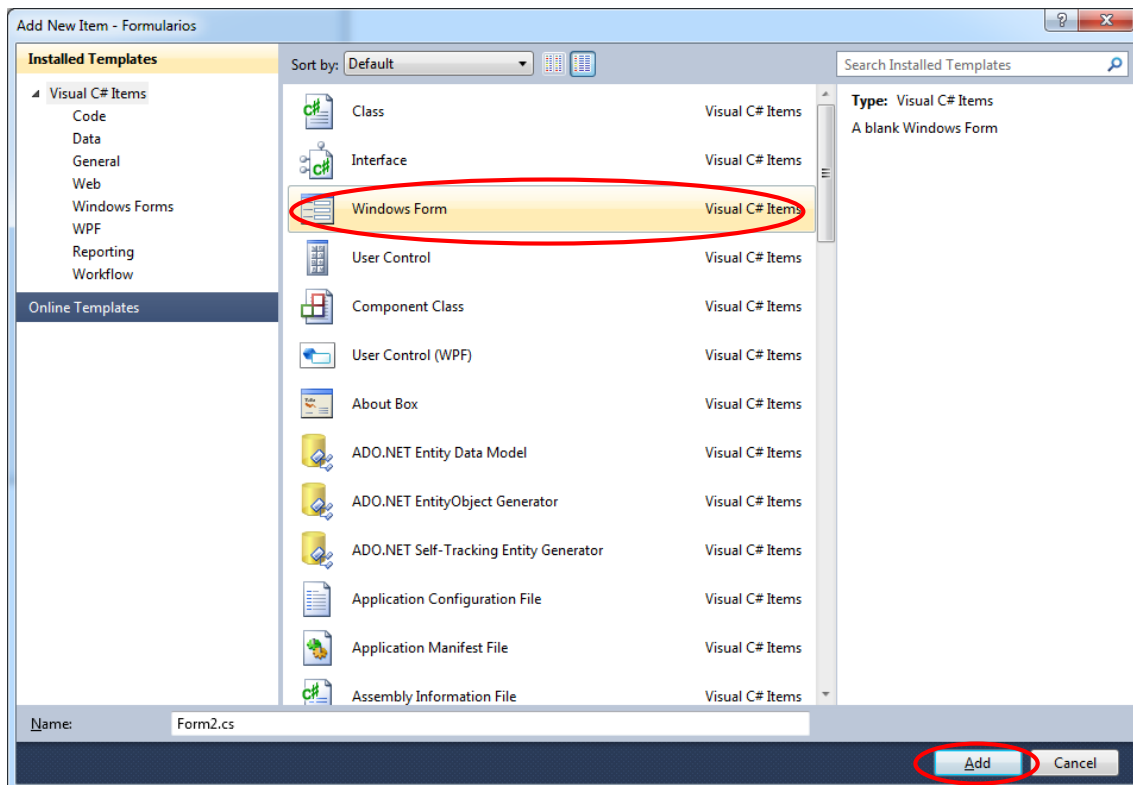
Depois de criar o menu no formulário, dê dois cliques em cada menu e escreva a programação desejada.

5. Criando aplicação com vários Forms

- 1- Crie um novo projeto do tipo Windows Forms Application chamado **Formularios**.
- 2- Insira um **MenuStrip** e crie o menu conforme a figura a seguir:



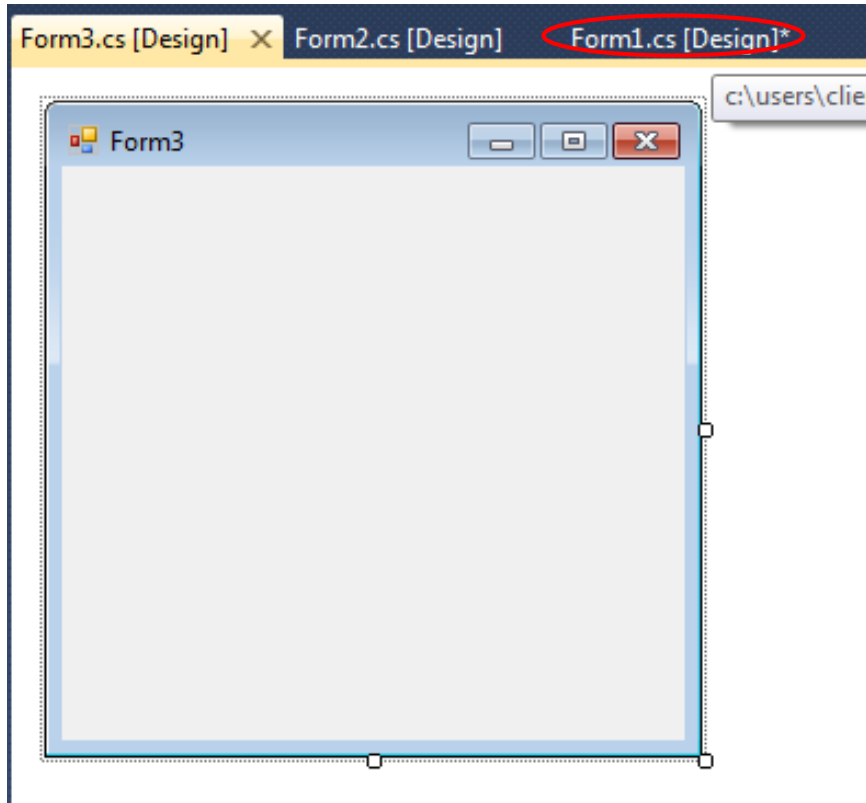
- 3- Pressione as teclas de atalho **CTRL+SHIFT+A**, escolha a opção **Windows Form** e Click em **Add**



4- Repita o passo anterior para adicionar o Form3;

Agora nossa aplicação possui três formulário, o Form1 que possui o Menu, e o form2 e form3.

5- Volte para o Form1.cs [Design]



6- Dê dois cliques no **SubMenu Form2** do **MenuStrip** presente no Form1 e codifique:

```
private void form1ToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form2 ProximoForm = new Form2();
    ProximoForm.ShowDialog();
}
```

7- Volte para o Form1.cs [Design]

8- Dê dois cliques no **SubMenu Form3** do **MenuStrip** presente no Form1 e codifique:

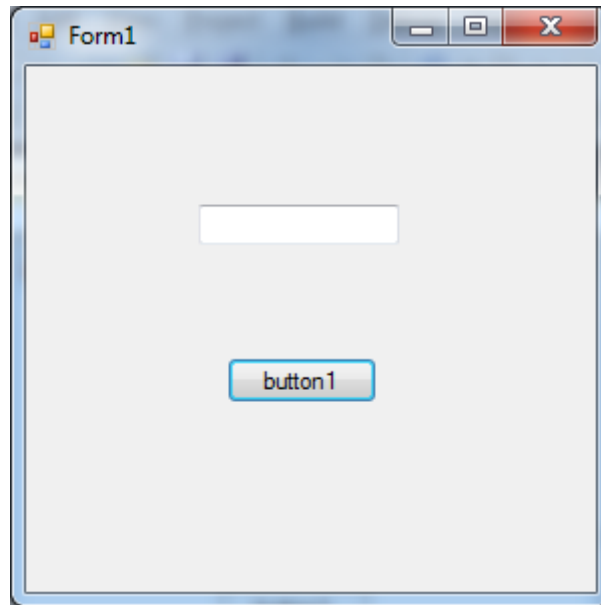
```
private void form2ToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form3 ProximoForm = new Form3();
    ProximoForm.ShowDialog();
}
```

9-Teste sua aplicação.

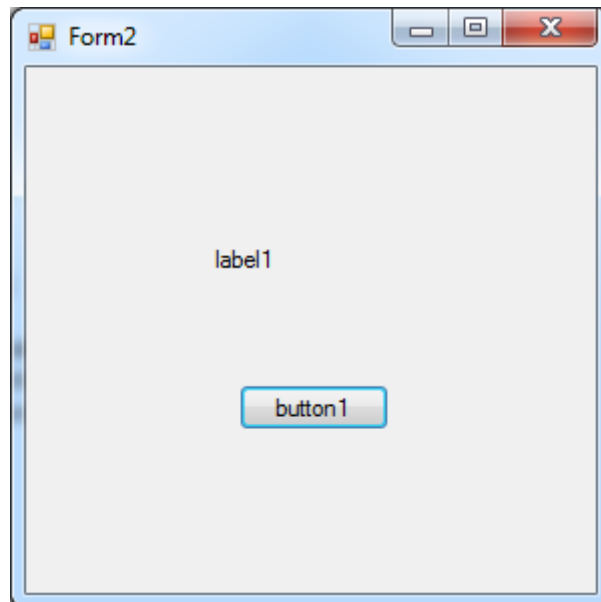
6. Variáveis Globais

O uso de variáveis globais é aconselhado quando temos que trocar valores entre formulários. Para exemplificar vamos criar uma aplicação que passará de um Form outro.

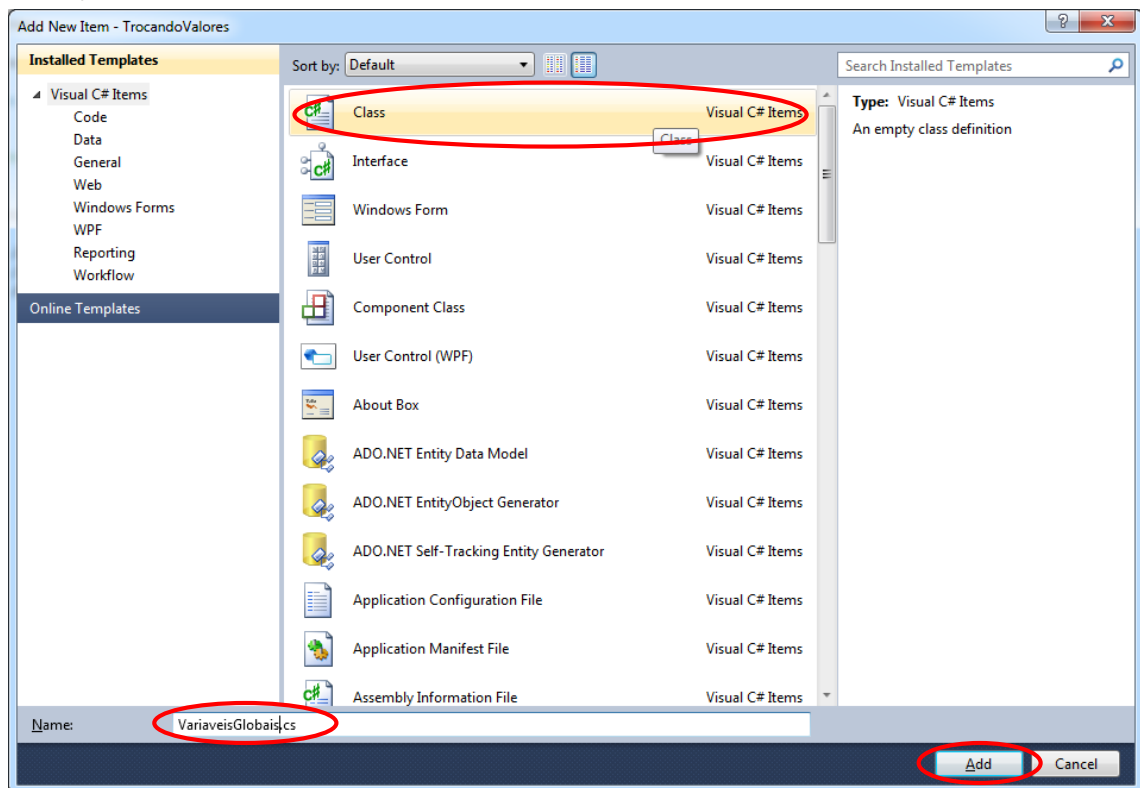
- 1- Crie um novo projeto(CTRL+SHIFT+N) do tipo Windows Forms Application chamado **TrocandoValores**.
- 2- No form1 insira uma textbox e um Button;



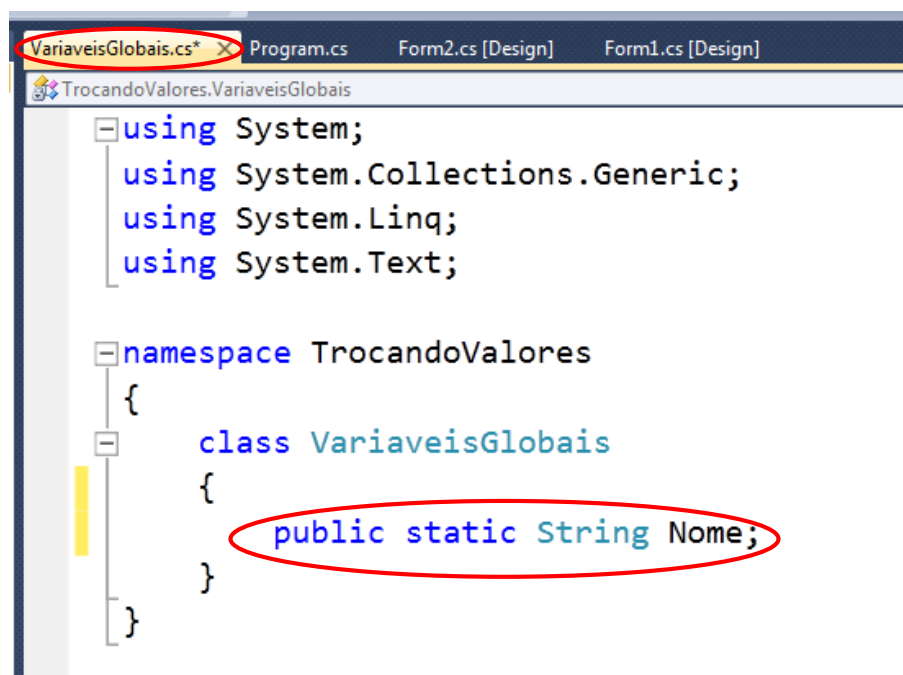
- 3- Insira um novo form(CTRL+SHIFT+A);
- 4- Nesse novo form2 insira um Button e uma label;



- 5- Agora vamos inserir uma nova Classe, pressione CTRL+SHIFT+A
- 6- Selecione a opção **Class**;
- 7- Altere o Name para **VariaveisGlobais**;
- 8- De **OK**;

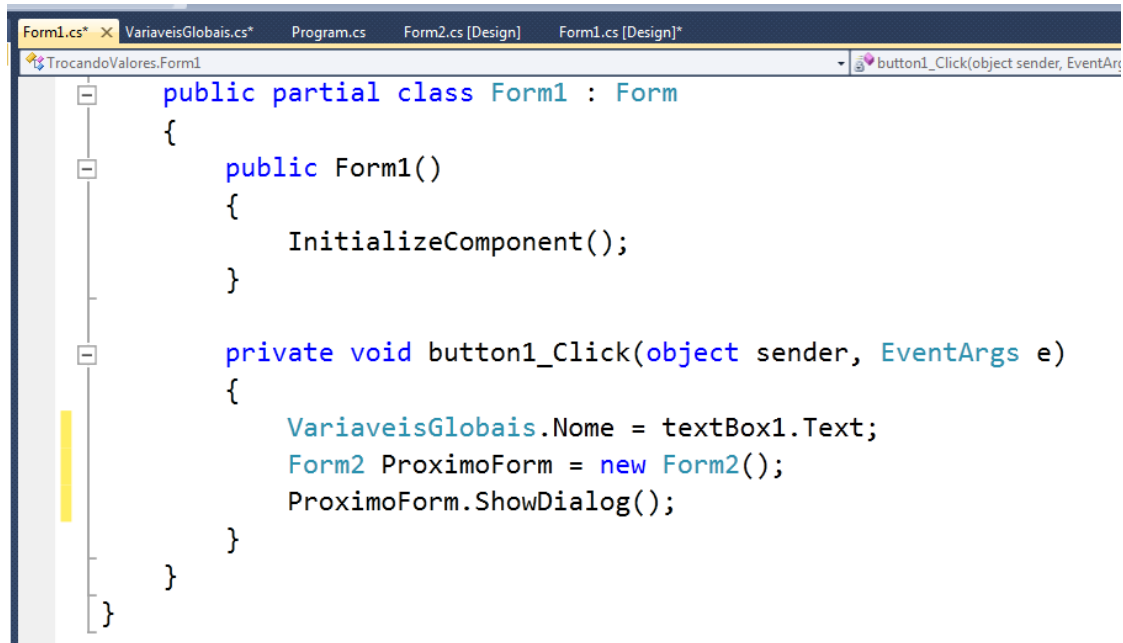


- 9- Dentro dessa nova classe, vamos declarar uma variável chamada **Nome** do tipo **String**:



Agora a variável Nome esta visível para todo nosso projeto.

10- Volte para o **form1** e codifique o **button1**:



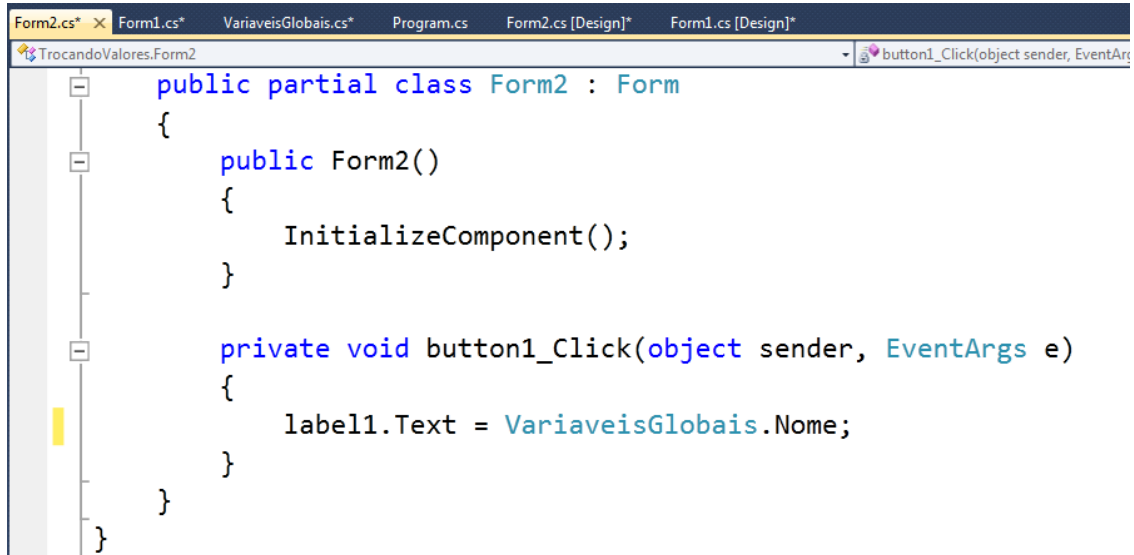
```
Form1.cs* x VariaveisGlobais.cs* Program.cs Form2.cs [Design] Form1.cs [Design]*
TrocandoValores.Form1
button1_Click(object sender, EventArgs e)

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

    private void button1_Click(object sender, EventArgs e)
    {
        VariaveisGlobais.Nome = textBox1.Text;
        Form2 ProximoForm = new Form2();
        ProximoForm.ShowDialog();
    }
}
```

Observe que quando o button1 for clicado, a **variável global Nome** receberá o texto presente na **textbox1**, logo em seguida o **form2** será chamado.

11- Agora codifique o **button1** do **form2**:



```
Form2.cs* x Form1.cs* VariaveisGlobais.cs* Program.cs Form2.cs [Design]* Form1.cs [Design]*
TrocandoValores.Form2
button1_Click(object sender, EventArgs e)

public partial class Form2 : Form
{
    public Form2()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        label1.Text = VariaveisGlobais.Nome;
    }
}
```

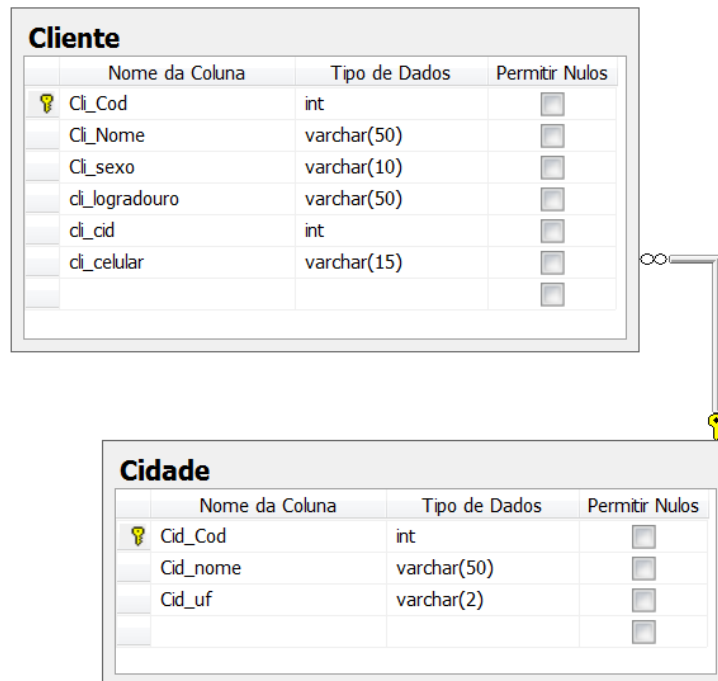
12- Teste sua aplicação.

Exercícios Complementares em aula.

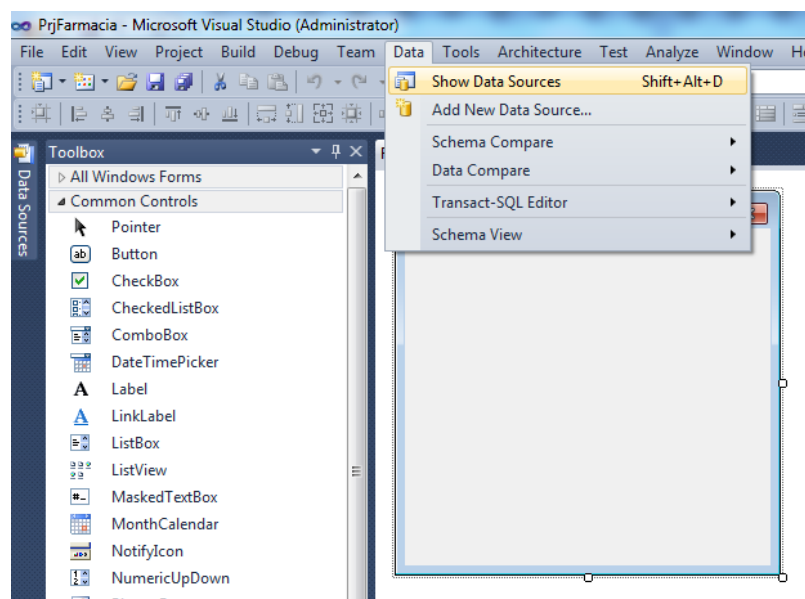
[illegible]

7. Ligando C# com SGBD MS SQL Server

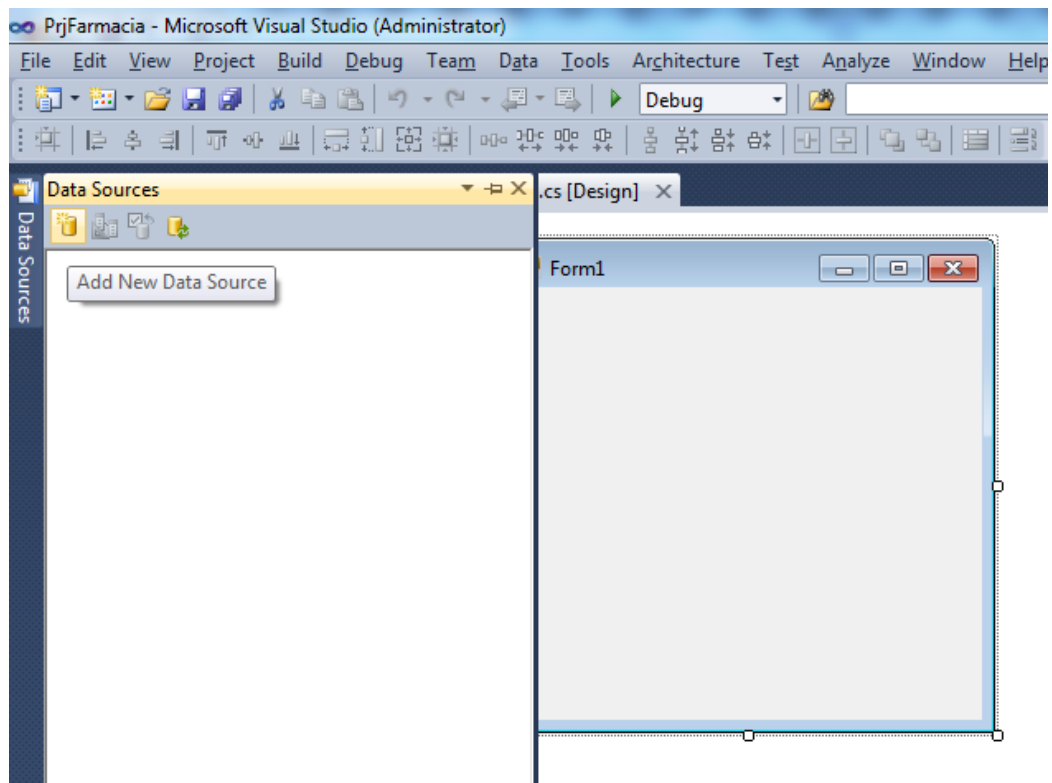
Para demonstração inicial, trabalharemos com um banco de dados que possui apenas duas tabelas, Cliente e Cidade, relacionadas entre si:



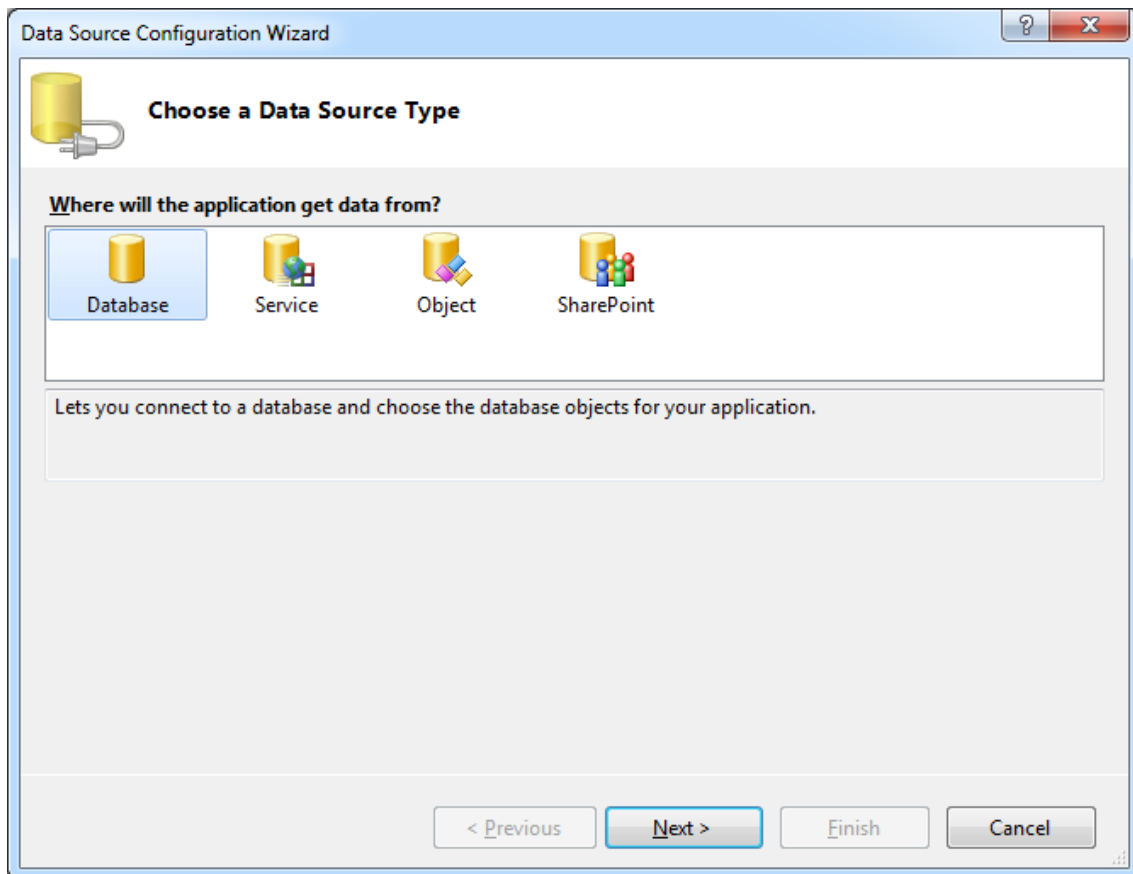
- 1- Crie um novo projeto(CTRL+SHIFT+N) do tipo Windows Forms Application chamado **PrjFarmacia**
- 2- Clique no **Menu Data** e selecione **Show Data Sources**(SHIFT+ALT+D);



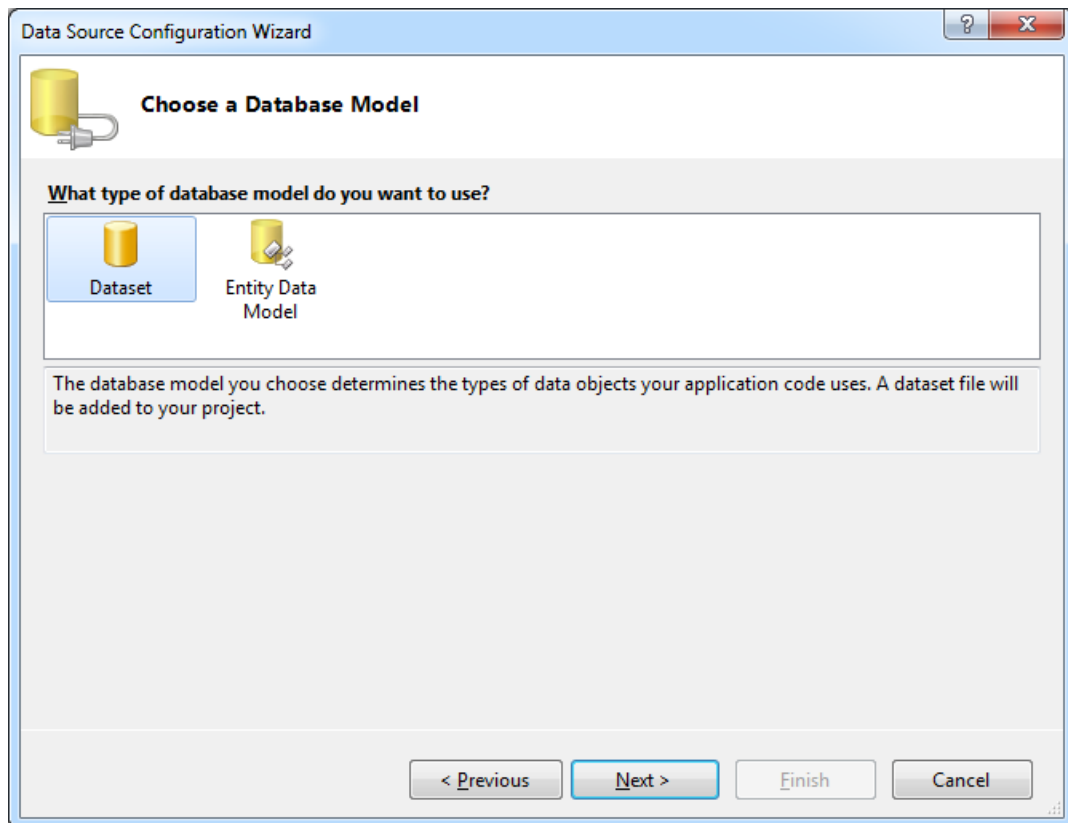
- 3- Click em **Add New Data Source**



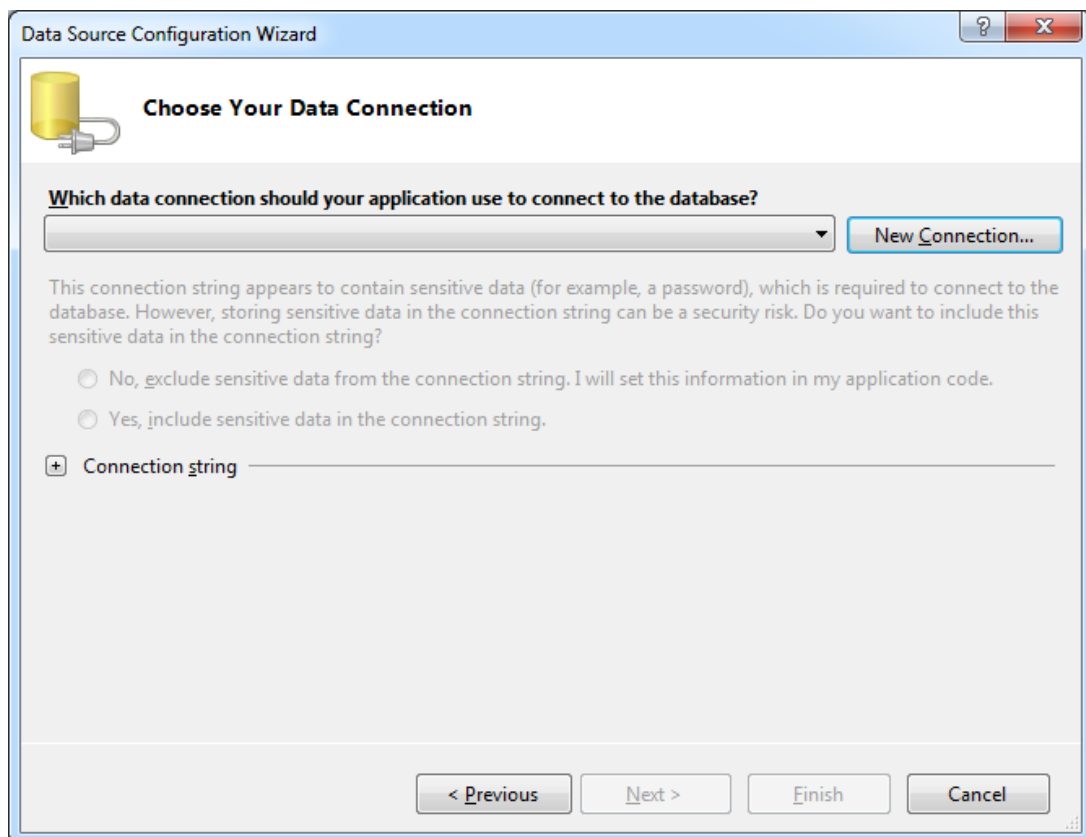
- 4- Seleccione **Database** e click em **Next**



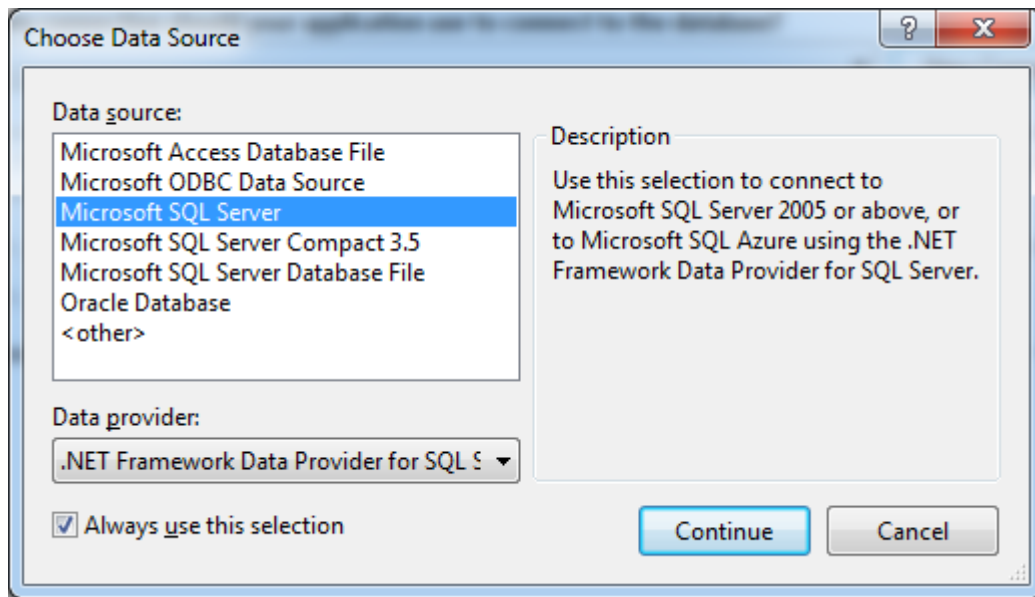
- 5- Seleccione **Dataset** e click em **Next**



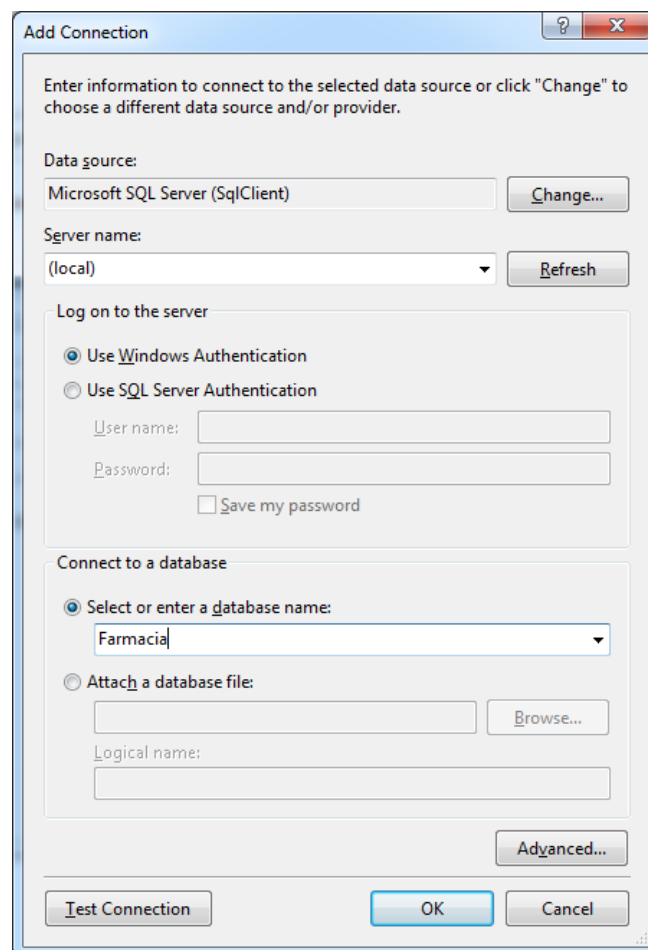
- 6- Click em **New Connection...**



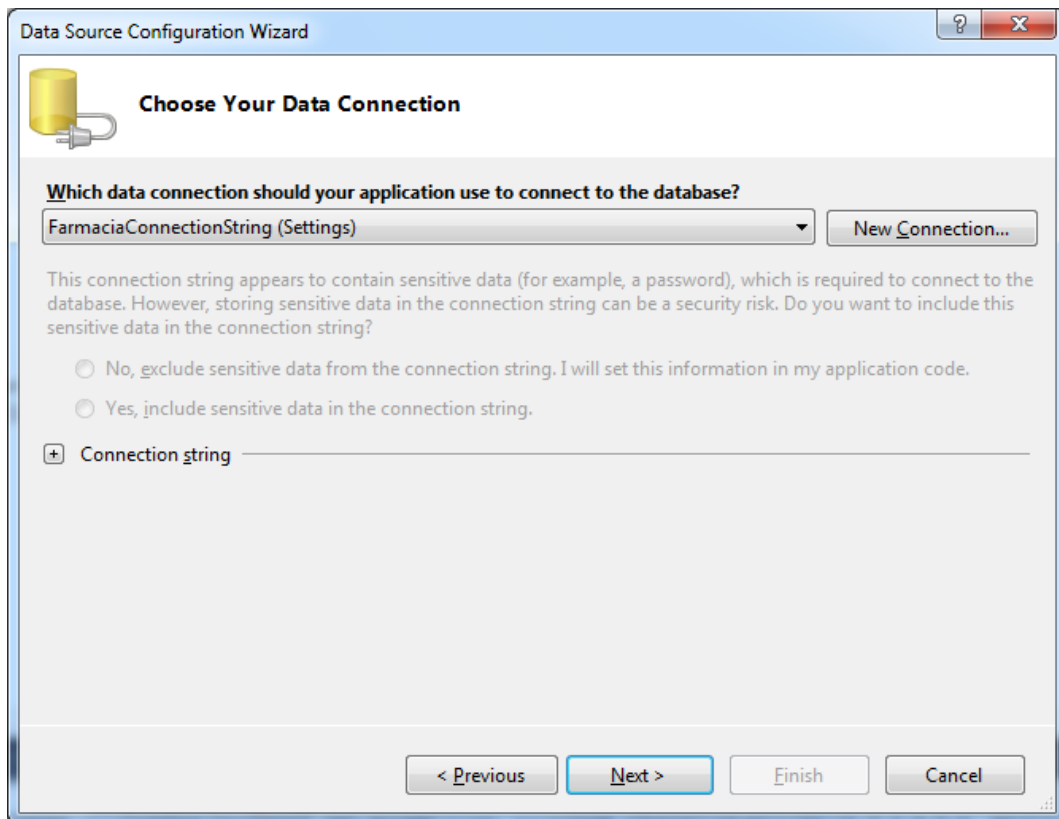
- 7- Selecione **Microsoft SQL Server** e click em **Continue**



- 8- Escolha o Server Name(**local**)
- 9- Escolha o **database name**, no nosso caso é o **Farmacia**
- 10- Click em **OK**



11- Click em **Next**



The screenshot shows the 'Data Source Configuration Wizard' window, specifically the 'Choose Your Data Connection' step. The window has a title bar with a question mark and a close button. The main area features a yellow database icon and the title 'Choose Your Data Connection'. Below this, a question asks which data connection the application should use. A dropdown menu shows 'FarmaciaConnectionString (Settings)', and a 'New Connection...' button is to its right. A paragraph explains that the connection string might contain sensitive data like a password, posing a security risk. Two radio buttons are provided: 'No, exclude sensitive data from the connection string. I will set this information in my application code.' and 'Yes, include sensitive data in the connection string.' The 'Yes' option is selected. Below the radio buttons is a section labeled 'Connection string' with a plus icon and a text input field. At the bottom, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

Data Source Configuration Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

FarmaciaConnectionString (Settings) New Connection...

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

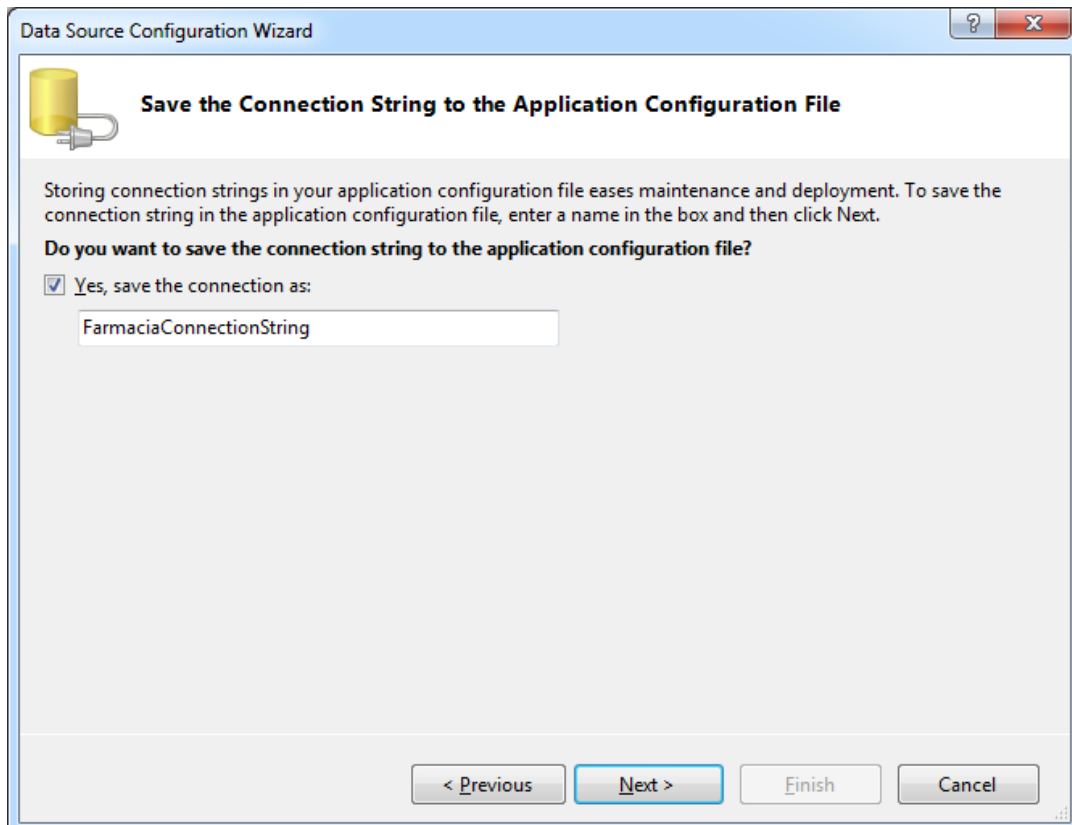
☐ No, exclude sensitive data from the connection string. I will set this information in my application code.

☒ Yes, include sensitive data in the connection string.

+ Connection string

< Previous Next > Finish Cancel

12- Click em **Next**



The screenshot shows the 'Data Source Configuration Wizard' window, specifically the 'Save the Connection String to the Application Configuration File' step. The window has a title bar with a question mark and a close button. The main area features a yellow database icon and the title 'Save the Connection String to the Application Configuration File'. Below this, a paragraph explains that storing connection strings in the application configuration file eases maintenance and deployment. A question asks if the user wants to save the connection string to the application configuration file. A checkbox labeled 'Yes, save the connection as:' is checked. Below the checkbox is a text input field containing 'FarmaciaConnectionString'. At the bottom, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

Data Source Configuration Wizard

Save the Connection String to the Application Configuration File

Storing connection strings in your application configuration file eases maintenance and deployment. To save the connection string in the application configuration file, enter a name in the box and then click Next.

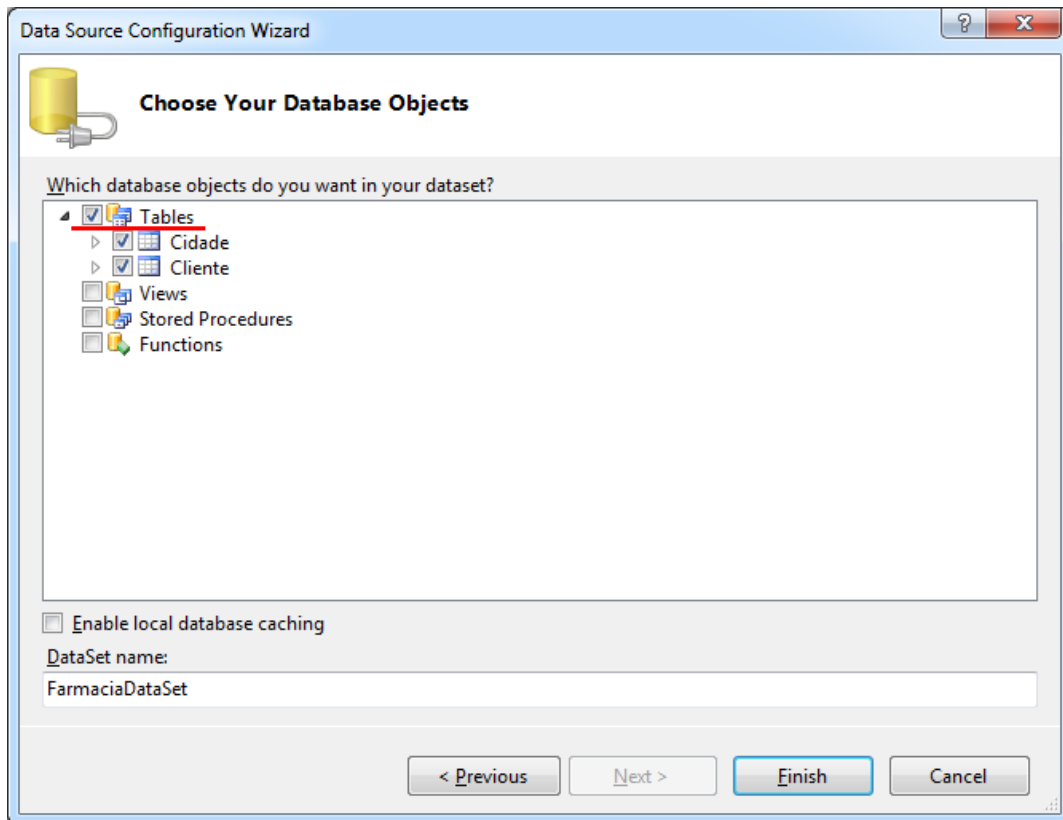
Do you want to save the connection string to the application configuration file?

☒ Yes, save the connection as:

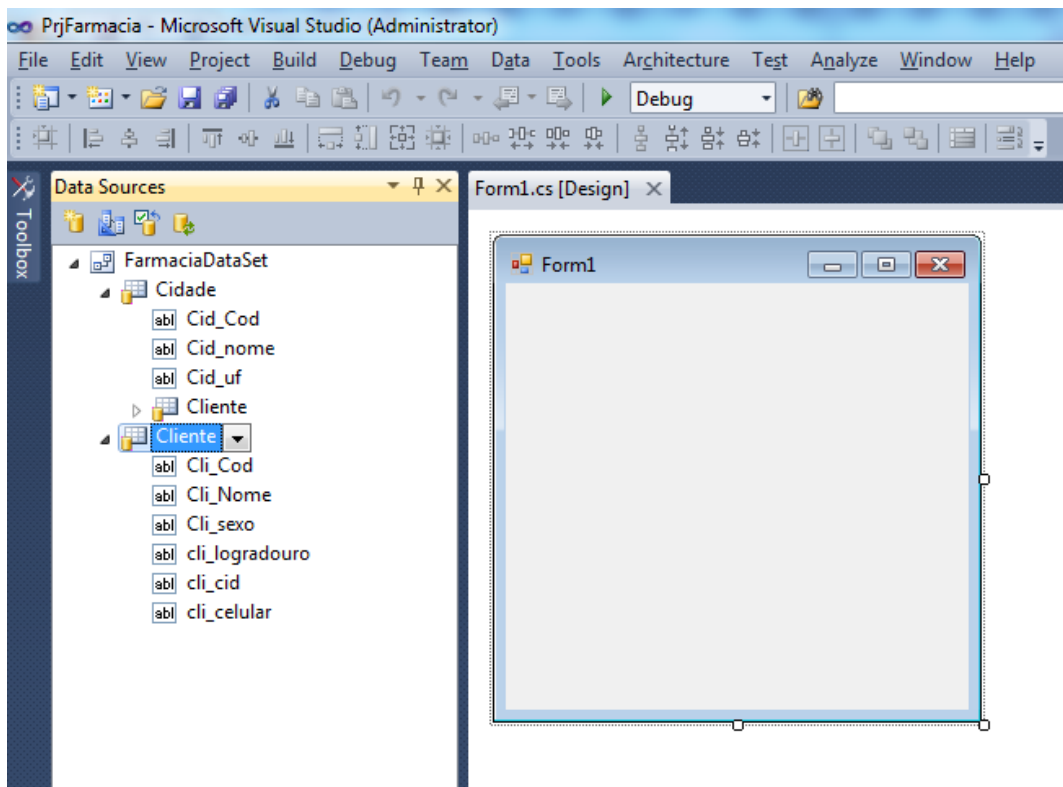
FarmaciaConnectionString

< Previous Next > Finish Cancel

- 13- Click em **Tables** para selecionar todas as tabelas do Banco de Dados
- 14- Click em **FINISH**



Agora o projeto está relacionado com o Banco de Dados Farmacia, as tabelas e campos podem ser vistos na guia **Data Sources**.

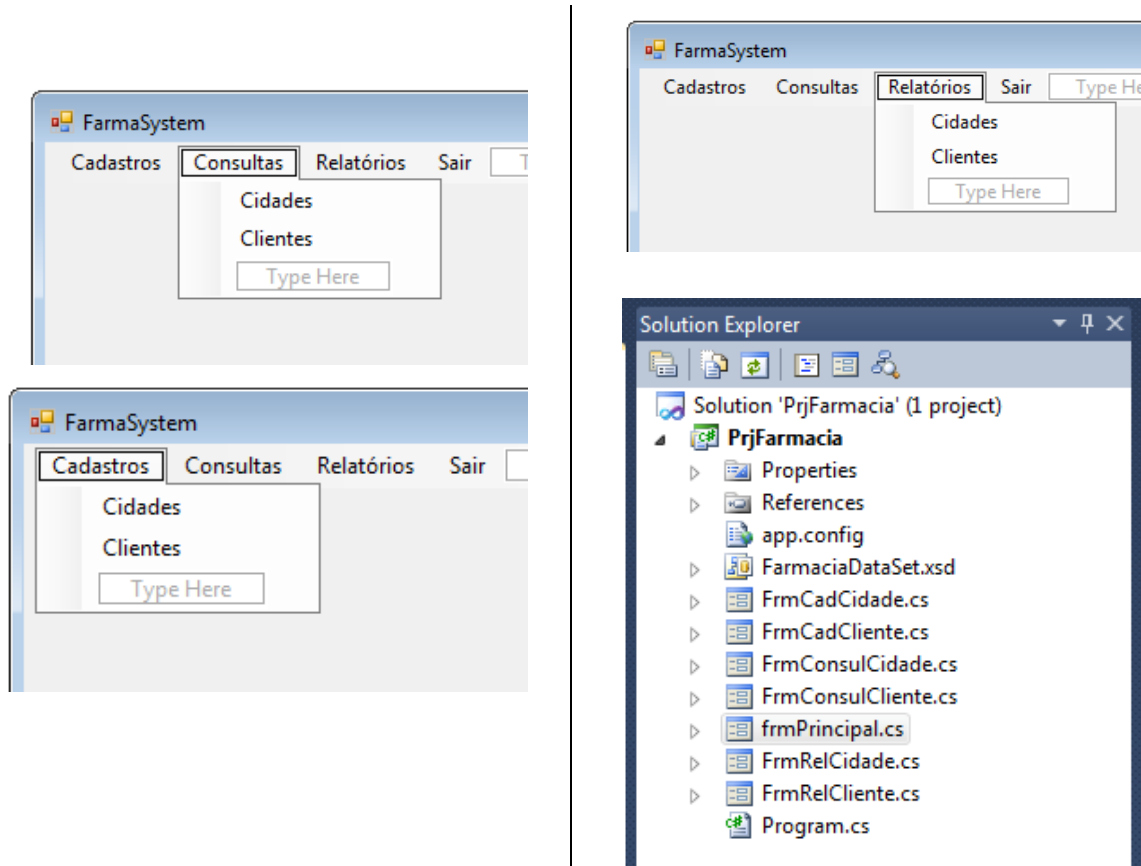


ATENÇÃO: A conexão é feita apenas uma vez, porém, nos laboratórios da nossa ETEC, a conexão tem que ser refeita a cada REINICIALIZAÇÃO do Windows, pois os HDs são “Congelados/Freezados”.

8. Criando Cadastros Básicos

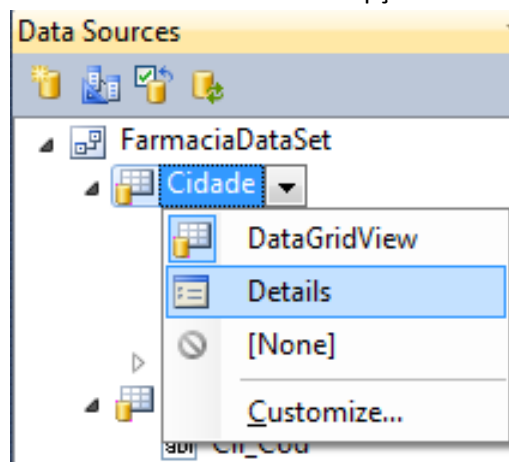
Dando continuidade ao capítulo anterior, iremos agora criar o **cadastro básico** da tabela **Cidade**.

Antes de iniciarmos, certifique-se que seu projeto possui o seguinte layout e forms criados:

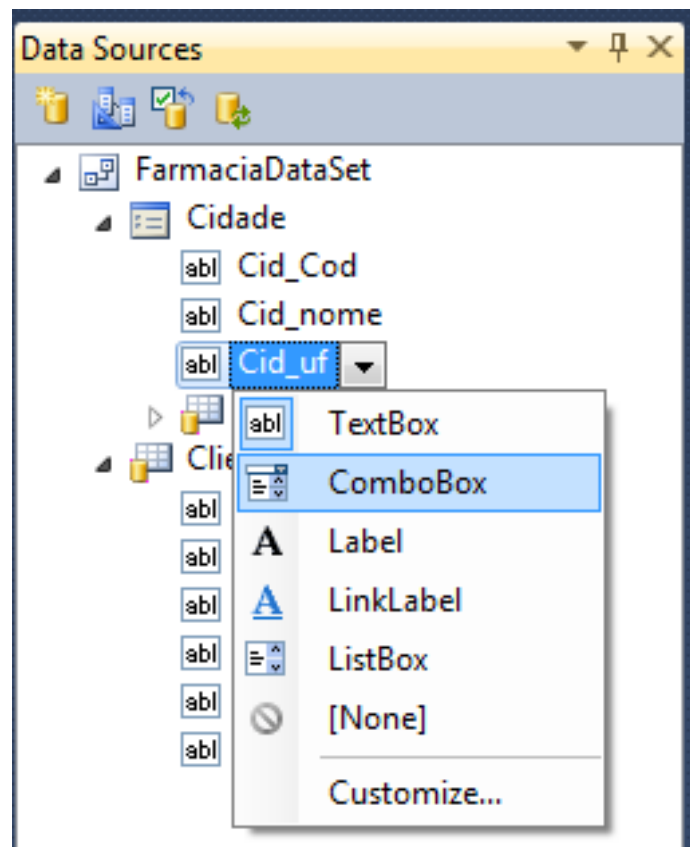


Cadastro de Cidades

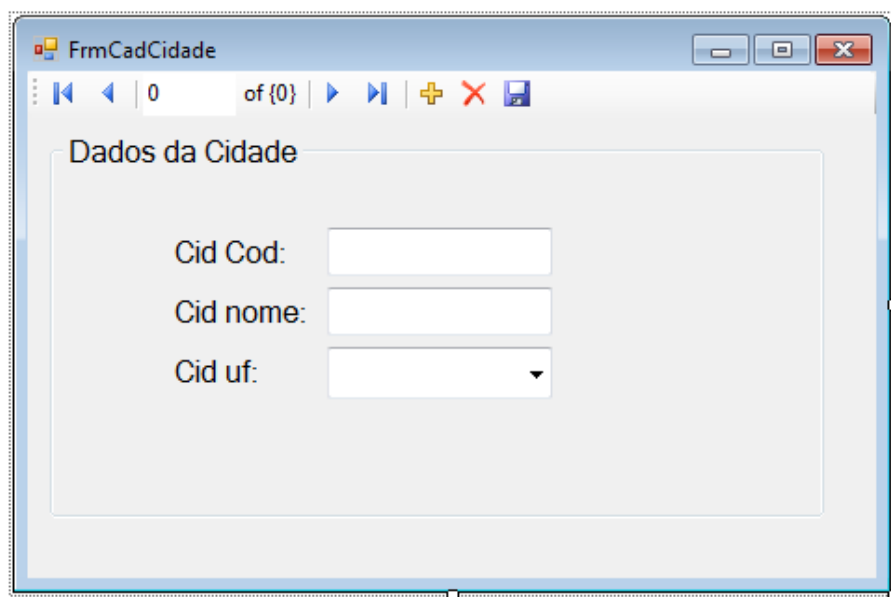
1. Abra o **frmCadCidade** [Design];
2. Insira uma **Groupbox** e altere o **text** para **Dados da Cidade**;
3. Na guia **Data Sources** click em **Cidade** e selecione a opção **Details**



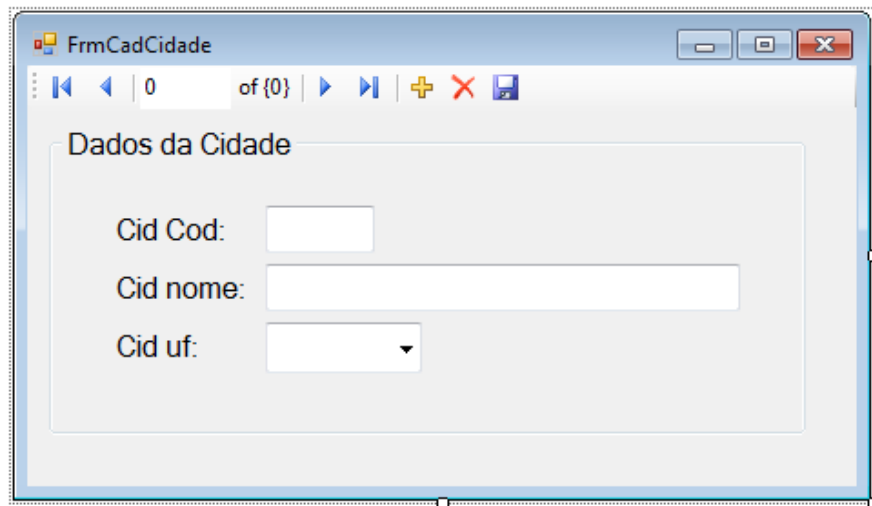
4. Click no campo **Cid_Uf** e selecione a opção **ComboBox**.



5. **Click e Arraste** a tabela Cidade para o **frmCadCidade**, em cima da **GroupBox1**;



6. Altere a propriedade **ENABLED** da **cid_CodTextBox** para **False**;
7. Insira as **siglas** dos Estados no itens da **cid_ufComboBox**;
8. Ajuste o **tamanho** das **TextBoxes**;
9. Altere a propriedade **StartPosition** do form para **CenterScreen**;



10. Altera a propriedade **DeleteItem** do **cidadeBindingNavigator** para **None**
11. De duplo clique no botão **Delete** do **bindingNavigator** e insira a seguinte programação:

```
try
{
    if (MessageBox.Show("confirma exclusão do registro atual",
        "FarmaSystem", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
        DialogResult.Yes)
    {
        cidadeBindingSource.RemoveCurrent();
        cidadeTableAdapter.Update(farmaciaDataSet.Cidade);
    }
}
catch (Exception ex)
{
    MessageBox.Show("registro não pode ser excluído " + ex.Message);
    this.cidadeTableAdapter.Fill(this.farmaciaDataSet.Cidade);
}
```

12. De duplo clique no botão **Save** do **bindingNavigator** e altera a programação existente para os códigos abaixo:

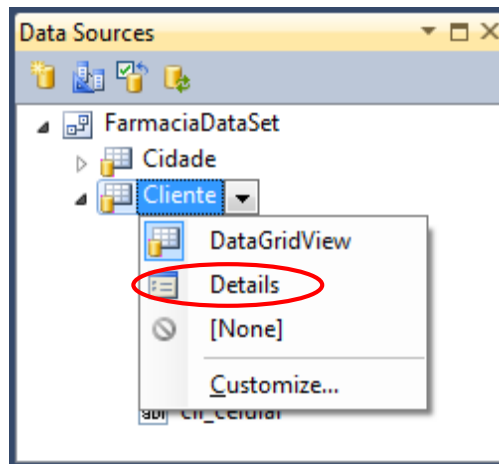
```
try
{
    this.Validate();
    this.cidadeBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.farmaciaDataSet);
    MessageBox.Show("Registro Salvo", "FarmaSystem", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
catch (Exception ex)
{
    MessageBox.Show("Erro ao salvar registro" + ex.Message);
}
```

13. Execute e adicione/remova alguns registros no formulário Cidade.

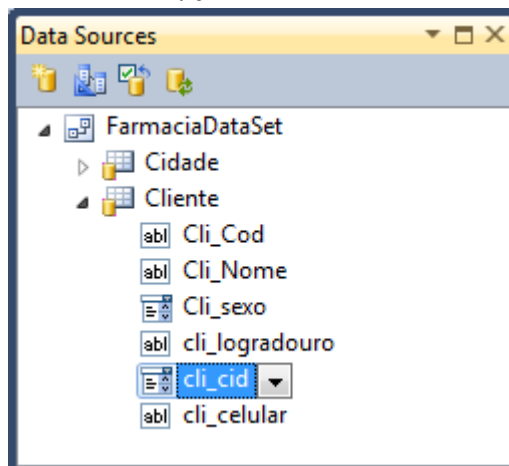
Cadastro de Clientes

No Cadastro de Clientes, a principal diferença em relação ao cadastro de cidade será no campo Cli_Cid, pois o formulário deverá “trazer” as cidades cadastradas para o usuário selecionar durante o cadastro de um novo cliente.

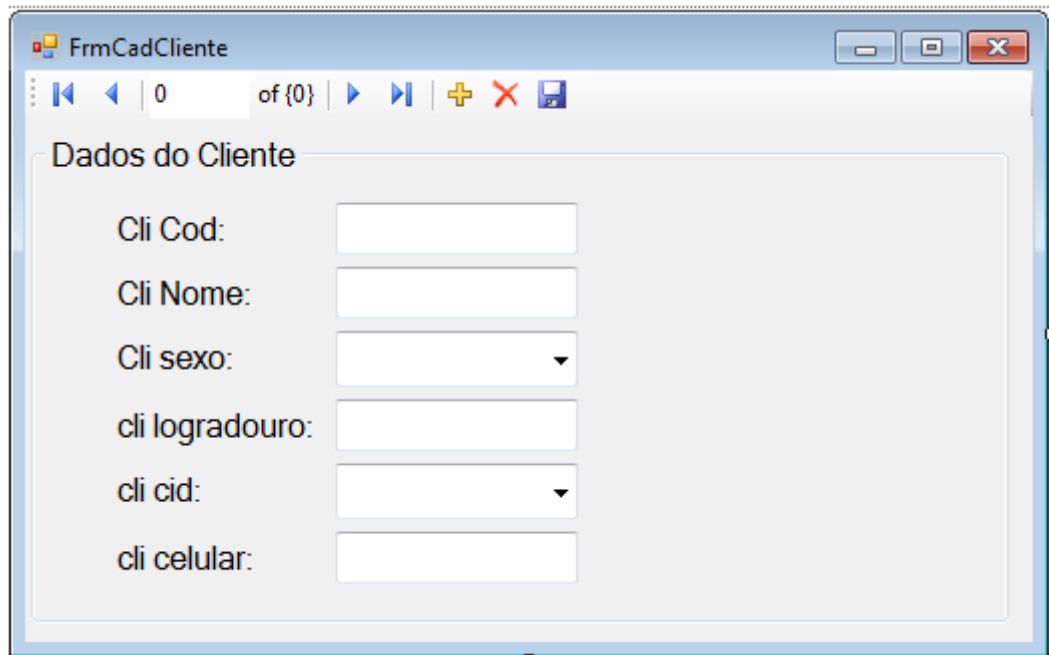
1. Abra o **frmCadCliente** [Design];
2. Insira uma **GroupBox** e altere o **text** para **Dados do Cliente**;
3. Na guia **Data Sources** click em **Cliente** e selecione a opção **Details**



4. Click no campo **Cli_Sexo** e selecione a opção **ComboBox**;
5. Click no campo **Cli_Cid** e selecione a opção **ComboBox**;



6. **Click e Arraste** a tabela **Cliente** para o **frmCadCliente**, em cima da **GroupBox1**;

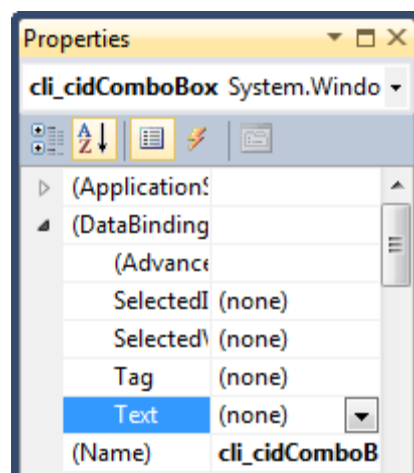


The screenshot shows a Windows form titled "FrmCadCliente". Inside the form, there is a section titled "Dados do Cliente" which contains several input fields: "Cli Cod:" (text box), "Cli Nome:" (text box), "Cli sexo:" (dropdown menu), "cli logradouro:" (text box), "cli cid:" (dropdown menu), and "cli celular:" (text box). The form has standard Windows window controls (minimize, maximize, close) in the top right corner.

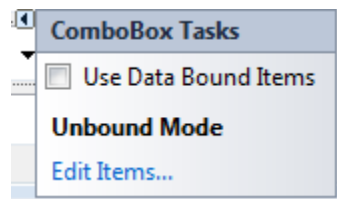
7. Altere a propriedade **ENABLED** da **Cli_CodTextBox** para **False**;
8. Insira as **siglas** os valores **Masculino** e **Feminino** no itens da **Cli_SexoComboBox**;
9. Altere a propriedade **StartPosition** do form para **CenterScreen**;

ATENÇÃO: Agora precisamos buscar os dados da tabela cidade e carrega-los na **Cli_CidComboBox**, faça:

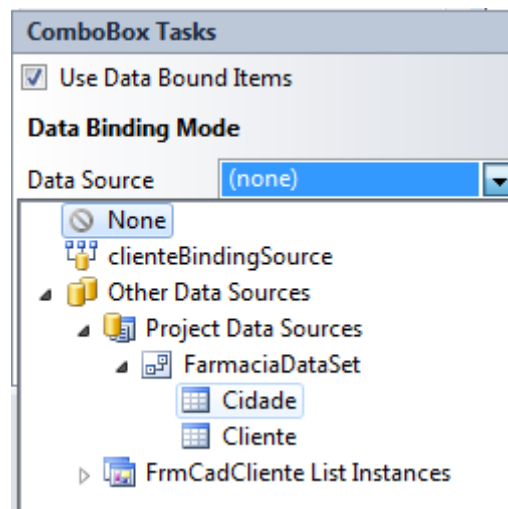
1. Altere a propriedade **DataBindings>Text** da **Cli_CidComboBox** para **none**



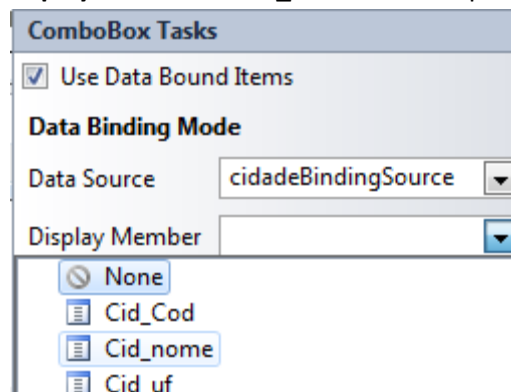
2. Click no **SmartButton** da **Cli_CidComboBox** e selecione a opção **Use Data Bound Items**;



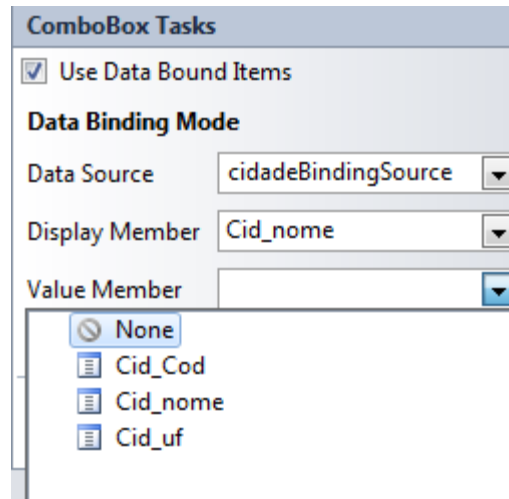
3. Altere a propriedade **Data Source** da **Cli_CidComboBox** para **Cidade**



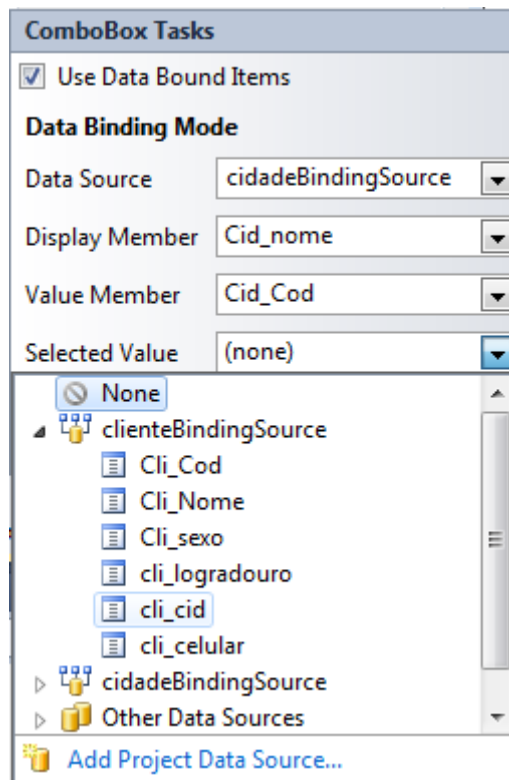
4. Altera a propriedade **Display Member** da **Cli_CidComboBox** para **Cid_Nome**



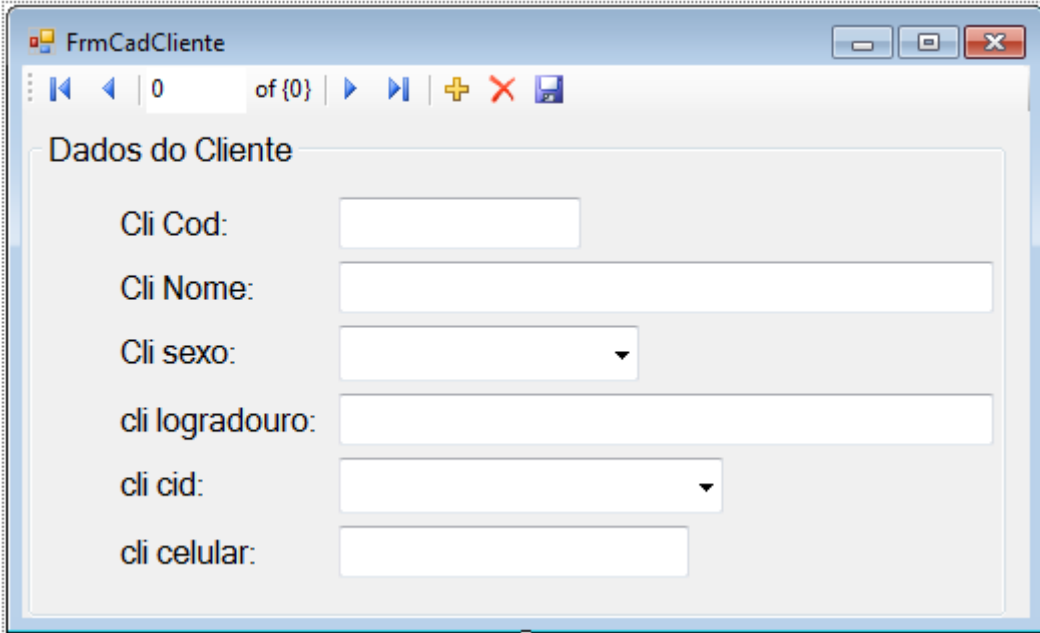
5. Altera a propriedade **Value Member** da **Cli_CidComboBox** para **Cid_Cod**



6. Altera a propriedade **Selected Value** da **Cli_CidComboBox** para **Cli_Cid**



10. Ajuste o **tamanho** das **TextBoxes**;



The screenshot shows a Windows form titled "FrmCadCliente". At the top, there is a status bar with navigation icons and a text box showing "0 of {0}". Below this is a section titled "Dados do Cliente" containing several input fields: "Cli Cod:" (a small text box), "Cli Nome:" (a larger text box), "Cli sexo:" (a dropdown menu), "cli logradouro:" (a text box), "cli cid:" (a dropdown menu), and "cli celular:" (a text box).

10. Altera a propriedade **DeleteItem** do **clienteBindingNavigator** para **None**

11. De duplo clique no botão **Delete** do **bindingNavigator** e insira a seguinte programação:

```
try
{
    if (MessageBox.Show("confirma exclusão do registro atual",
        "FarmaSystem", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
        DialogResult.Yes)
    {
        clienteBindingSource.RemoveCurrent();
        clienteTableAdapter.Update(farmaciaDataSet.Cliente);
    }
}
catch (Exception ex)
{
    MessageBox.Show("registro não pode ser excluído " + ex.Message);
    this.clienteTableAdapter.Fill(this.farmaciaDataSet.Cliente);
}
```

12. De duplo clique no botão **Save** do **bindingNavigator** e altera a programação existente para os códigos abaixo:

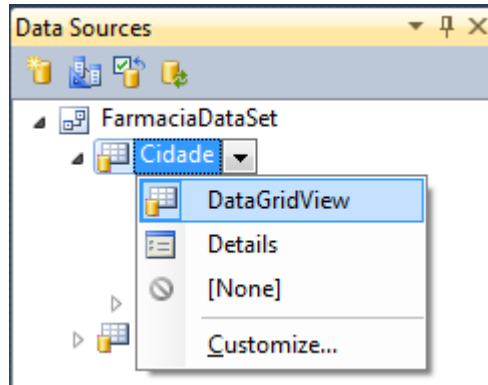
```
try
{
    this.Validate();
    this.clienteBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.farmaciaDataSet);
    MessageBox.Show("Registro Salvo", "FarmaSystem", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
catch (Exception ex)
{
    MessageBox.Show("Erro ao salvar registro" + ex.Message);
}
```

13. Execute e adicione/remova alguns registros no formulário Cidade.

9. Criando Consultas

A consulta será criada através do evento FILTER do BindingSource

1. Abra o **frmConsulCidade** [Design];
2. **Arraste** a tabela **Cidade** no modo **DataGridView**(Não em Details) para o formulário;



3. **Delete** o **BindingNavigator**;
4. Adicione uma **TextBox** e Renomeie como **txtConsulNome**;
5. Adicione um **button** e renomeie para **btExibirTodos**;
6. De dois cliques na **txtConsulNome** e na programação do evento **CHANGE** inclua a seguinte programação:

```
private void TxtConsulNome_TextChanged(object sender, EventArgs e)
{
    cidadeBindingSource.Filter = "Cid_Nome like '" + TxtConsulNome.Text + "%'";
}
```

7. Volte ao Formulário e de dois cliques do **btExibirTodos** e inclua a programação:

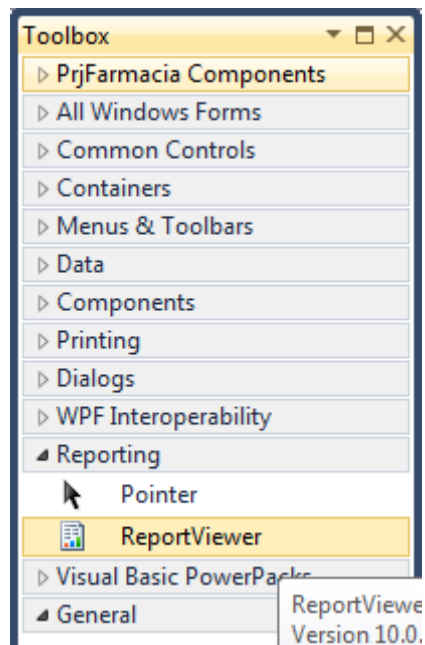
```
private void btExibirTodos_Click(object sender, EventArgs e)
{
    cidadeBindingSource.RemoveFilter();
    TxtConsulNome.Clear();
}
```

8. Teste sua Consulta;

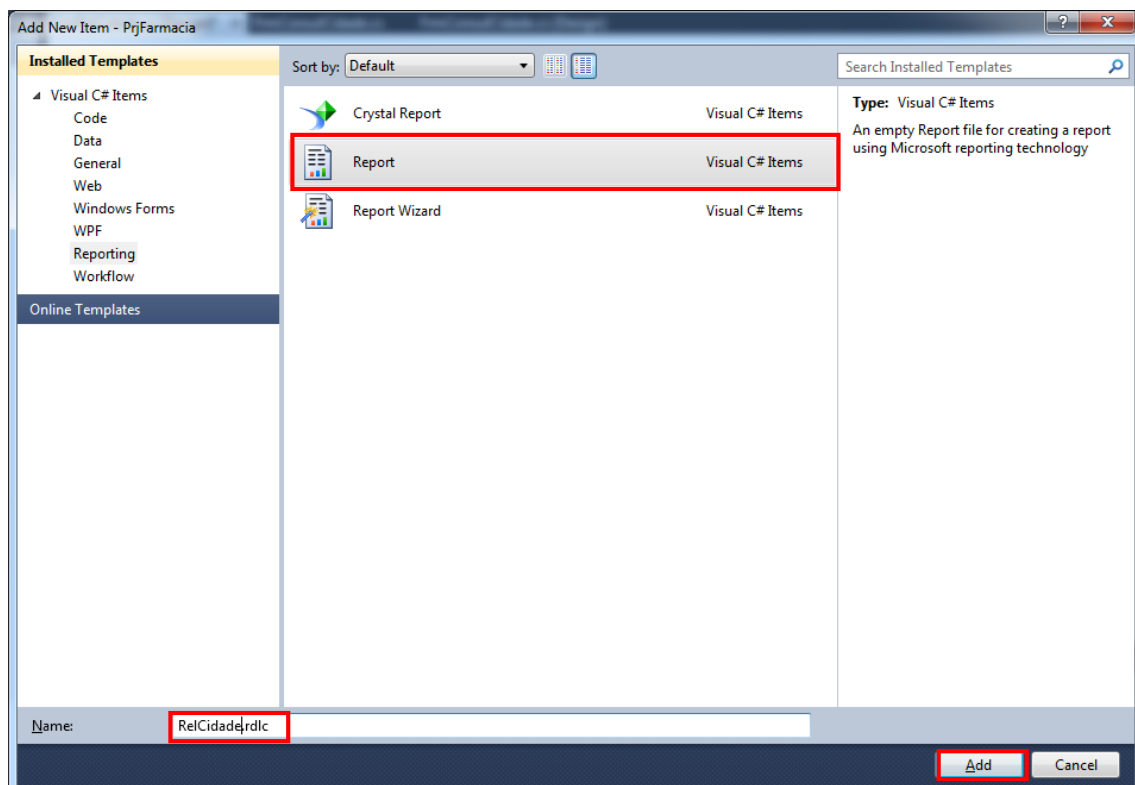
Atenção: O exemplo acima foi baseado no BindingSource CIDADE, mas funcionará para qualquer outra tabela. Apenas tenha em mente que onde houver CIDADE, deverá ser substituído pela outra tabela em questão.

10. Criando Relatórios

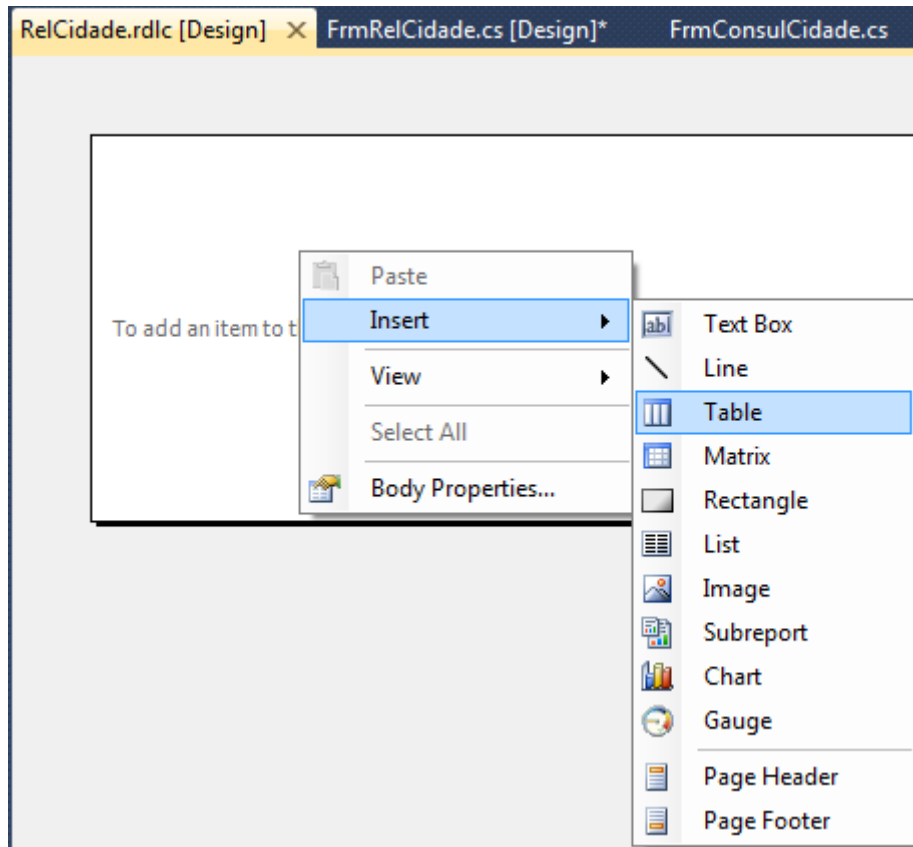
1. Adicionar um objeto **ReportViewer** ao formulário **frmRelCidade**



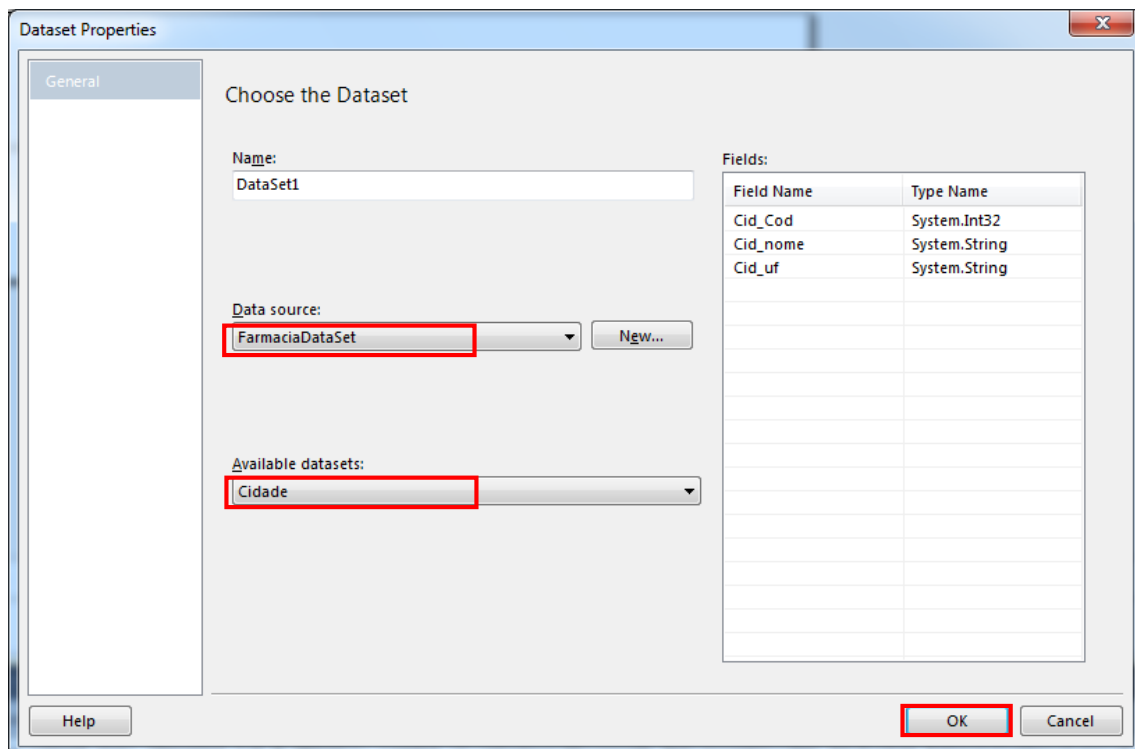
2. Ajustar o ReportViewer1 ao **tamanho** do formulário;
3. Alterar a **propriedade Anchor** do ReportViewer1 para **Top, Bottom, Left e Right**;
4. Pressione **CTRL+SHIFT+A** e Adicione um **Report** da guia **Reporting** com o nome de RelCidade;



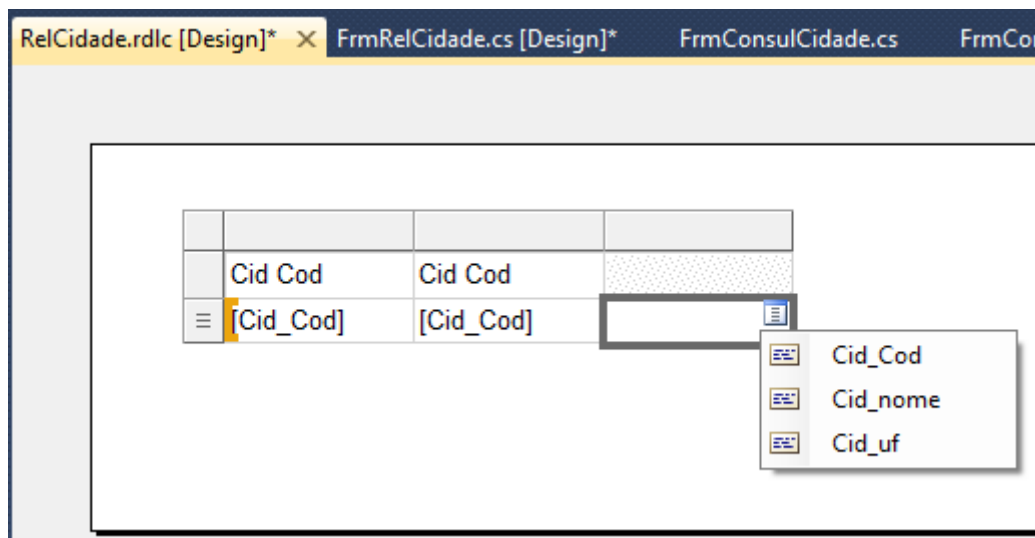
5. No **RelCidade.rdlc** [Design] clicar com o botão direito no **Report** depois em **Insert>Table**



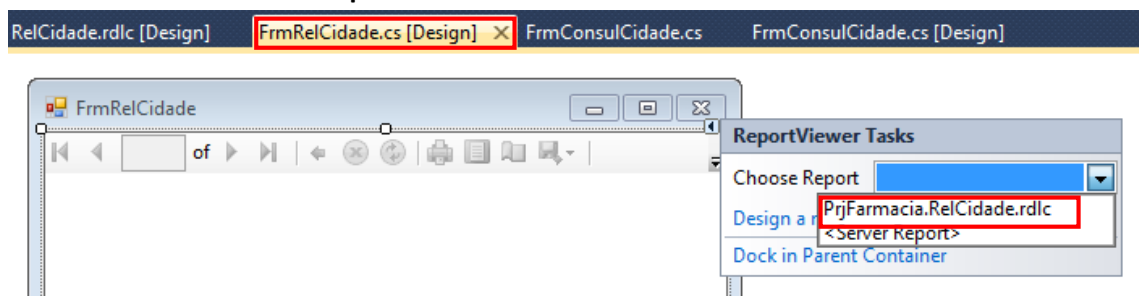
6. Escolher o **Data Source FarmaciaDataSet**;
7. Selecionar a Tabela **Cidade** em **Available datasets**;
8. **Click** em **Ok**



9. Inserir as **Colunas** na **Table** conforme imagem abaixo:



10. Formate a Tabela a seu gosto;
11. Volte ao **FrmRelCidade** [Design];
12. Selecione o **RelCidade** no **ReportViewer1**



13. No Evento **FormClosed** do **FrmRelCidade** insira a linha de código:

```
private void FrmRelCidade_FormClosed(object sender, FormClosedEventArgs e)
{
    this.Dispose();
}
```

14. Execute e visualize seus relatórios;

