

# **APOSTILA DE VISUAL STUDIO 2008**

**NUPEDS**

**Núcleo de Pesquisa e Desenvolvimento de Sistemas**

**FACULDADES INTEGRADAS CLARETIANAS**



**RIO CLARO**

**2009**

## SUMÁRIO

1. Visão geral do Visual Studio 2008 .....	4
1.1 Visual Studio 2008.....	4
1.2 O que há de novo no Visual Studio 2008.....	4
- O compilador do Visual Basic e a linguagem .....	4
1.3 O que é possível fazer com o Visual Studio 2008 .....	5
1.4 Ambiente de Desenvolvimento Integrado.....	5
2. Programando em Visual Basic .....	7
2.1 Criar um Formulário .....	7
2.2 Criar um novo Projeto.....	7
2.3 Escrevendo o primeiro programa.....	8
2.3.1 Alguns Conceitos .....	8
2.3.2 Programa que faz operação matemática simples .....	9
2.3.3 Executando aplicações Visual Basic.....	11
2.3.4 Salvando aplicações Visual Basic.....	11
3. Estruturas de Controle.....	11
3.1 If...Then.....	11
3.2.1 Programa com estrutura de controle If...Then .....	12
3.2.2 Editando o código do programa .....	13
3.2 Select Case .....	14
3.2.1 Programa com estrutura de controle Select Case.....	15
4. Estruturas de Repetição.....	17
4.1 For...Next .....	17
4.1.2 Programa com estrutura de repetição For...Next .....	18
4.2 While.....	20
4.2.1 Programa com estrutura de repetição Do .....	20
4.3 Do.....	21
4.3.1 Programa com estrutura de repetição Do .....	21
5. Formulários e Menus .....	22
5.1 Adicionando novos formulários.....	22
5.2 Adicionando Menu.....	24
6. Trabalhando com vários tipos de controles da Toolbox .....	27
6.1 Construindo o programa de compras .....	27
6.2 Outro método para se fazer uma Calculadora.....	34
7. Banco de Dados .....	40
7.1 A biblioteca ADO .....	40
7.2 Objeto Connection .....	40
7.3 O Objeto Recordset.....	41
7.4 Objeto Field.....	42
7.5 Objeto OleDbCommand.....	42
7.6 Objeto Error .....	42
8. A Biblioteca ADO.NET .....	42
8.2 O que é o ADO.NET.....	42
8.3 Componentes e objetos do ADO.NET .....	43
8.4 Arquitetura da ADO. NET .....	44
8.4.1 Objetos de acesso a uma base de dados .....	44
8.5 Conhecendo outras classes do ADO.NET .....	45
8.5.1 DataSet.....	45
8.5.2 DataTable .....	46
8.5.3 DataAdapter .....	46

8.5.4 DataView .....	46
8.5.5 DataRelation.....	46
8.6 Componente Binding Source .....	46
8.7 O componente BindingNavigator .....	47
8.8 O componente TableAdapterManager .....	47
9. Criando acesso a banco através do assistente .....	47
10. Acessando dados usando o MS-Access .....	54
10.1 Como trabalhar com o Microsoft Access usando o OleDb Data Provider .....	54
10.2 Criar uma consulta ao banco de dados usando parâmetros (por código).....	56
10.3 Criar uma consulta com escolha de parâmetros (Através do Assistente) .....	58
10.4 Fazer consultas por parte do Nome .....	63
10.5 Consulta com data .....	64
11. Diferença entre DataSet e DataTable.....	65
12. Objeto DataRow.....	66
12.1 Preencher objeto DataSet ( via código).....	66
13. Diferença entre DataGrid e DataGridView .....	68
14. Data Binding .....	69
Incluir, excluir e alterar dados na tabela .....	71
15. Relatório.....	73
16. Bibliografia .....	75

# 1. Visão geral do Visual Studio 2008

## 1.1 Visual Studio 2008

O Microsoft® Visual Studio® 2008 permite que os desenvolvedores criem com muita rapidez aplicativos, proporcionar experiências de usuário com a mais alta qualidade e riqueza. O Visual Studio 2008, junta ferramentas com as quais as organizações sentirão maior facilidade em capturar e analisar informações, o que significa a melhor tomada de decisões de negócios. O Visual Studio 2008 possibilita que organizações de todos os tamanhos criem aplicativos mais seguros, gerenciáveis e confiáveis que tiram proveito do Windows Vista™ e do Office System 2007.

O Visual Studio 2008 se baseia em três pilares para proporcionar melhor experiência para os programadores: Melhorias na produtividade do desenvolvedor; Gerenciamento do ciclo de vida do aplicativo; e Utilização das mais recentes tecnologias.

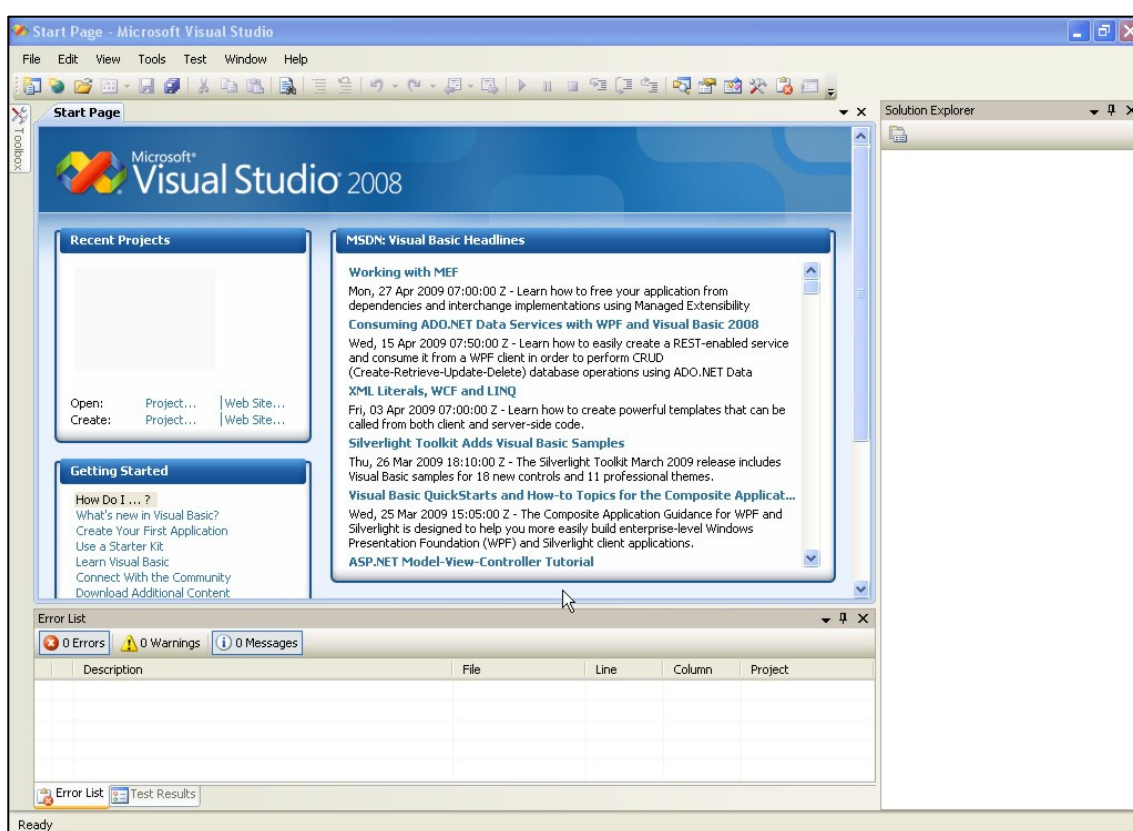


Figura 1: Tela de Abertura Visual Studio 2008

## 1.2 O que há de novo no Visual Studio 2008

### - O compilador do Visual Basic e a linguagem

As melhorias de linguagem no Visual Basic 2008 simplificam o código-fonte e permitem a interação com os componentes que usam recursos avançados.

### - Suporte para novos tipos de projeto

No Visual Studio 2008, o Project Designer oferece suporte a novo Windows Presentation Foundation (WPF) e tipos de projeto de aplicativos Web (WAP).

#### - Suporte para novas versões .NET Framework

Multitargeting permite direcionar o código para uma versão específica do .NET Framework:

#### - Implantação do Windows Installer

A implantação do Windows Installer foi atualizada para oferecer suporte ao Windows Vista e as versões mais recentes do .NET Framework

### 1.3 O que é possível fazer com o Visual Studio 2008

Visual Studio é um conjunto completo de ferramentas de desenvolvimento para construção de aplicações Web ASP. NET, serviços Web XML, aplicações da área de trabalho 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 mistas. 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.

### 1.4 Ambiente de Desenvolvimento Integrado

A família de produtos Visual Studio compartilha um simples ambiente de desenvolvimento integrado (IDE) que é composto de vários elementos: a barra de ferramentas menu, barra de ferramentas Padrão, diversas ferramentas Windows ancorada ou auto-ocultas à esquerda, inferior e direito, bem como o editor de espaço. A janela de ferramenta, menus e barras de ferramentas disponíveis dependem do tipo de projeto ou arquivo que estiver trabalhando.

#### ToolBox

No lado esquerdo do IDE, você vê uma guia vertical marcada como Caixa de Ferramentas. Também é inicialmente em branco, mas enquanto você trabalha, ela será preenchida com os itens que podem ser usados para a tarefa que você está trabalhando atualmente.

#### Barra de menus e barra de ferramentas

Na parte superior do IDE existe uma barra de menus e uma barra de ferramentas. Os menus disponíveis e a barra de ferramentas de botões mudam com base em sua tarefa atual.

#### Solution Explorer

Os arquivos são exibidos em uma exibição hierárquica. Por padrão, o Solution Explorer está localizado no lado direito do IDE.

Quando você cria um novo aplicativo Windows Forms, uma solução Windows Application aparece no Solution Explorer. A solução contém dois nós: My project e Form1.vb.

**My Project** abre o Project Designer ao clicar duas vezes nele. O Project Designer fornece acesso a propriedades do projeto, configurações e recursos.

**Form1.vb** é o Windows Form em sua solução. Você pode exibir esse arquivo no modo Design, que permite que você veja o formulário e os controles que você adicionou a ele. Você também pode visualizar esse arquivo no Editor de Código, que permite que você veja o código associado com o aplicativo que está criando.

## Janela Properties (Propriedades)

Será exibida embaixo da Janela Gerenciador de Soluções (Solution Explorer). Onde irá definir as várias propriedades que definem a aparência e comportamento do formulário e seus controles.

## Code Editor (Editor de Código)

Se você clicar duas vezes em um formulário ou controle, uma nova janela chamada Editor de Código será aberta. É onde você escreve o código real para o seu aplicativo.

O Editor de códigos usa uma tecnologia conhecida como o **IntelliSense** para ajudar você a escrever código, fornecendo informações relevantes à medida que você digita.

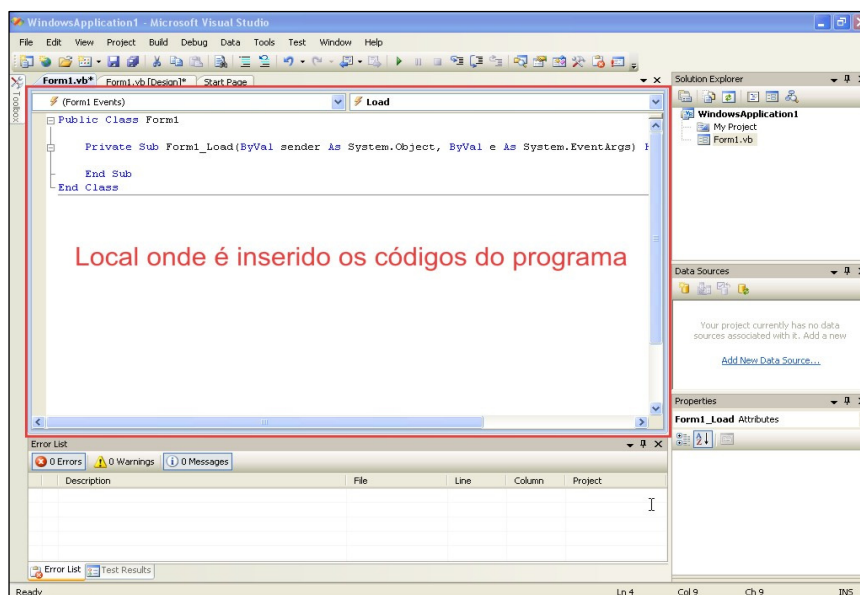


Figura 2 - Tela do Code Editor (Editor de Código)

## Modo Design

No modo design, a Página Inicial é coberta por outra janela conhecida como o Criador de Formulário, que é basicamente uma tela em branco que representa a interface do usuário para seu aplicativo.

Quando o criador de formulário estiver visível, a Caixa de Ferramentas (ToolBox) contém muitos controles (representações de botões, campos de texto, grades e assim por diante) que podem ser adicionados ao formulário e organizados como desejar.

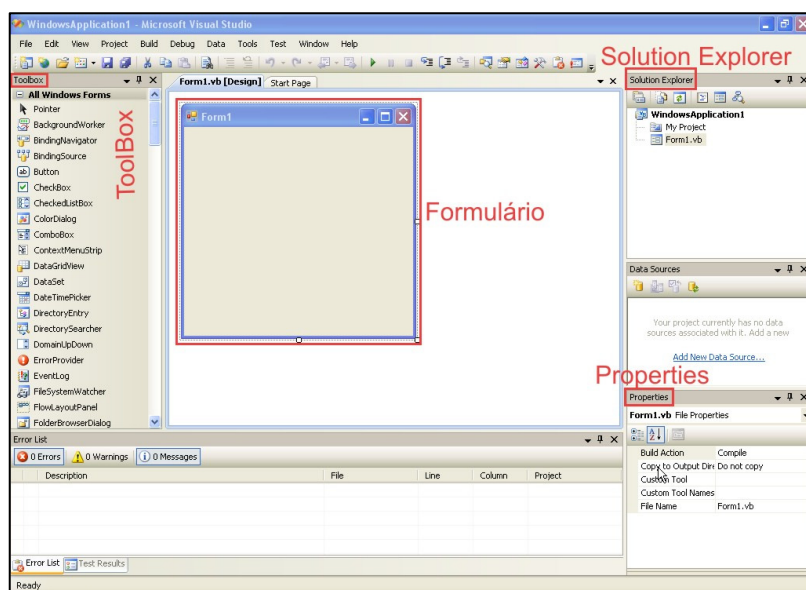


Figura 3 - Tela em Modo design

## 2. Programando em Visual Basic

Visual Basic 2008 é uma evolução da linguagem do Visual Basic, projetada para compilar com produtividade aplicações fortemente tipadas e orientadas a objeto. O Visual Basic permite aos desenvolvedores terem como alvo o Windows, Web e dispositivos móveis. Como todas as linguagens para Microsoft .NET Framework, programas escritos em Visual Basic se beneficiam da segurança e interoperabilidade da linguagem.

Esta geração do Visual Basic continua a tradição de fornecer uma maneira rápida e fácil de criar aplicativos baseados no .NET Framework.

### 2.1 Criar um Formulário

1. Com o Visual Studio 2008 aberto, no menu file, clique em new Project. A caixa de Diálogo New Project abre.
2. A caixa de diálogo New Project fornece acesso aos principais tipos de projetos disponíveis para escrever aplicações Windows.
3. Clique no ícone de Aplicação Windows na área de Templates da caixa de diálogo.
4. Clique em OK para iniciar o projeto no Visual Studio.

Após isso será aberto o Visual Studio em modo design.

### 2.2 Criar um novo Projeto

1. No menu File, clique em New Project. Caso esteja com o outro Projeto aberto, aparecerá uma janela solicitando para que salve o projeto antigo.

2. Irá aparecer a janela de New Project com já visto anteriormente.

## 2.3 Escrevendo o primeiro programa

### 2.3.1 Alguns Conceitos

**Instruções de programa:** é uma linha de código em um programa Visual Basic, executada pelo compilador que realiza trabalho útil da aplicação. Sempre seguindo as regras de sintaxe do compilador.

Instruções podem ser palavras-chave, propriedades, nome de objetos, variáveis, números, símbolos especiais...

**Variável:** armazena dados em um programa temporariamente.

Em Visual Basic cria variáveis utilizando a palavra-chave “Dim”.

**Controle:** ferramenta utilizada para criar objetos. *Exemplo:* Botão.

Utiliza a maioria dos controles para criar elementos de interface de usuário, caixa de figuras, caixa de listagem...

**Propriedades:** é um valor, ou característica, possuído por um objeto.

*Exemplo:* Objeto Botão tem uma propriedade Text, para especificar o texto que aparece no botão.

### Configurando a propriedade no Editor de Código (Code Editor)

Objeto.Propriedade = Valor

Onde:

Objeto = Nome do Objeto que está personalizando.

Propriedade = Característica que quer alterar.

Valor = A nova configuração da propriedade.

Button1.Text = “OK”

Foi usada a Propriedade *Text* para colocar a palavra OK no Button1.

### Procedimento de Evento

Um bloco de código que é executado quando um objeto é manipulado em um programa.

*Exemplo:* Button1

Quando ele é clicado, o procedimento de evento *Button1\_Click* é executado e automaticamente o Visual Studio adiciona as primeiras linhas do procedimento de evento.

*Private Sub Button1\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) Handles Button1.Click*



Obs: A linha do código abaixo foi quebrada para permanecer dentro da margem da página.

### **Método**

Instrução especial que realiza uma ação ou um serviço para um objeto particular em um programa.

### ***Configurando a propriedade no código (Code Editor)***

Objeto.Método(Valor)

Onde:

Objeto = Objeto com o qual você quer trabalhar.

Método = Ação que quer realizar.

Valor = Argumento a ser utilizado pelo método.

`ListBox1.Items.Add("Computador")`

Método Add para colocar a palavra Computador na caixa de listagem ListBox1

### **2.3.2 Programa que faz operação matemática simples (soma de dois números)**

1. Com o Visual Studio 2008 aberto, crie um novo projeto com o nome de Calculadora simples.  
(Consultar como criar novo projeto).

2. Dê duplo clique no controle TextBox na Toolbox.

O Visual Studio cria um objeto TextBox de tamanho padrão no formulário.

3. Crie mais dois controles TextBox no formulário.

4. Agora dê duplo clique no controle Button (botão) na Toolbox .

O Visual Studio cria um objeto Botão de tamanho padrão no formulário.

5. Crie três Controles Label da Toolbox para criar rótulo para as TextBox.

Arraste os controles criados para que fiquem como a imagem abaixo.

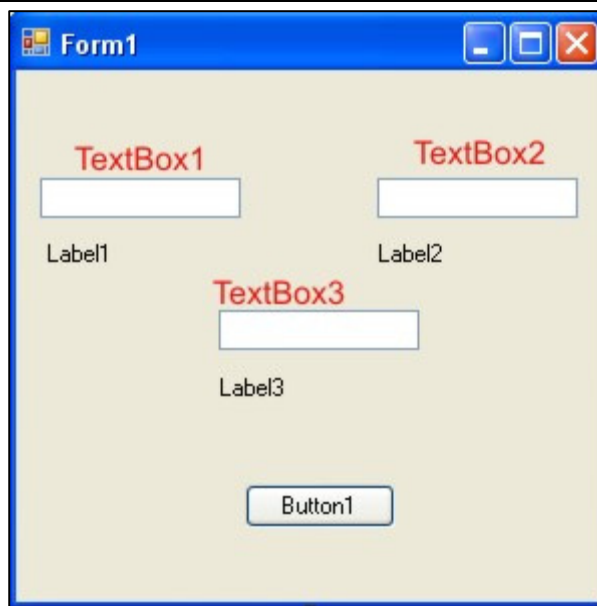


Figura 4 – Formulário do primeiro programa

6. Dê um clique para selecionar a Label1, vá à janela properties, altere a propriedade Text para Primeiro número.

7. Faça o mesmo com a Label2 e altere para Segundo número e Label3 para Soma.

8. Também altere a propriedade Text do Botão para Somar.

Os Objetos ainda estão sem funcionalidades, é preciso editar o código do programa (Code Editor).

9. Dê duplo clique no formulário para abrir o editor de código.

Será necessário criar duas variáveis.

Adicione o código embaixo de **Public Class Form1**

*Dim PrimeiroNumero, SegundoNumero As Double*

Foi declarada uma variável chamada PrimeiroNumero e outra chamada SegundoNumero do tipo Double.

10. Dê duplo clique no controle Button (Botão), será aberto o procedimento de evento Button1\_Click

Adicione o código: (OBS. Inicie o código de onde o cursor estiver).

*PrimeiroNumero = TextBox1.Text*

*SegundoNumero = TextBox2.Text*

*TextBox3.Text = PrimeiroNumero + SegundoNumero*

A TextBox1 recebe um número e armazena na variável PrimeiroNumero, a TextBox2 recebe outro número e armazena na variável SegundoNumero.

Quando o Button1 (Botão Somar), for clicado, a TextBox3 recebe a soma do número da TextBox1 com o número da TextBox2.

### 2.3.3 Executando aplicações Visual Basic

Para executar um programa do Visual Basic a partir do ambiente de desenvolvimento, siga qualquer um destes passos:

- Clique em *start Debugging* no menu *Debug*.
- Clique no botão *Start Debbugging* na barra de ferramentas standard.
- Pressione F5.

### 2.3.4 Salvando aplicações Visual Basic

Clique no botão Save All na barra de ferramentas Standard para salvar o projeto.

O Visual Studio agora solicita um nome e uma localização para o projeto.

**Caixa de seleção Create Directory For Solution** = Quando ela está selecionada o Visual Studio cria uma pasta para o projeto a ser salvo.

Após escolhido o localização para o arquivo, clique em Save para salvar o projeto e seus arquivos.

## 3. Estruturas de Controle

### 3.1 If...Then

Quando uma expressão condicional for utilizada em um bloco especial de instruções chamado de Estrutura de Decisão, ela controla se outras instruções no programa são executadas e em que ordem são executadas.

*Sintaxe:*

**If** *condição* **Then** *instrução*

Exemplo:

*If Pontos >= 100 Then Label1.Text = "Você Ganhou!"*

**Condição:** Pontos >= 100

**Instrução:** *Label1.Text = "Você Ganhou!"*

Para determinar se o programa deve configurar a propriedade *Text* do *objeto* Label1 com "Você ganhou!", se a variável pontos contiver um valor maior ou igual a 100, o Visual Basic configura a propriedade *Text*; caso contrário, ele pula a instrução de atribuição e executa a próxima linha no procedimento de evento. Esse tipo de comparação sempre resulta em um valor true (verdadeiro) ou false (falso). Uma expressão condicional nunca resulta em talvez.

Tabela 2- Operadores Aritméticos

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação

/	Divisão
\	Divisão de inteiro (N° Inteiro)
Mod	Divisão de resto
^	Exponenciação (elevar a uma potência)
&	Concatenação de strings (Combinação)

Tabela 2 - Operadores de Comparação

Operador de Comparação	Significado
=	Igual a
<>	Não igual a
>	Maior que
<	Menor que
>=	Maior que ou igual a
<=	Menor que ou igual a

### 3.2.1 Programa com estrutura de controle If....Then

#### (Matemática Básica: Soma, Subtração, Multiplicação, Divisão)

1. Com o Visual Studio 2008 aberto, no menu File, clique em New Project.  
A caixa de diálogo New Project abre.
2. Crie um novo projeto Windows Form Application chamado Calculadora.
3. Crie dois controles label da ToolBox.  
Altere a propriedade Text da Label1 para Primeiro Número e da Label2 para Segundo Número.
4. Altere a propriedade size do formulário para 405; 210.
5. Crie dois controles TextBox da ToolBox.
6. Crie um controle GroupBox .  
Altere a propriedade Text para Operador e Size para 133; 116
7. Crie 4 controles RadioButton. Coloque dentro do controle GroupBox1.  
Altere a Propriedade Text:  
RadioButton1 = Adição ( + )  
RadioButton2 = Subtração ( - )  
RadioButton3 = Multiplicação ( \* )  
RadioButton4 = Divisão ( / )
8. Crie mais uma Label e uma TextBox, altere a propriedade Text do controle Label para Resultado.
9. Crie dois controles Button (Botão), Altere a propriedade Text:  
Button1 = Calcular

Button2 = Sair

Mova os Controles para que fiquem assim:

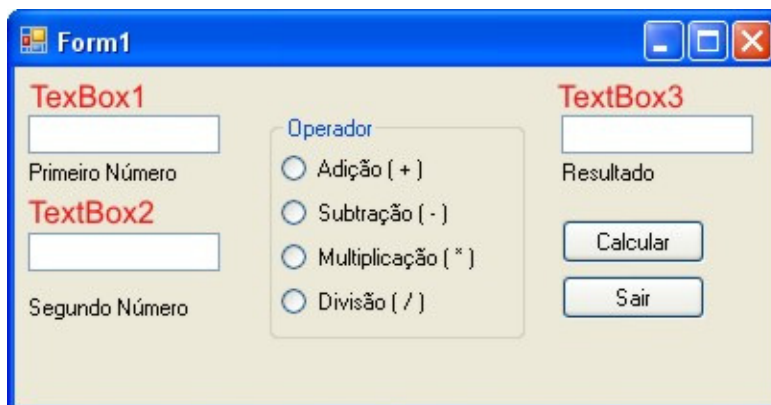


Figura 4 – Formulário do programa da calculadora

### 3.2.2 Editando o código do programa

10. Dê um clique no botão Calcular do formulário.

O code editor exibe o procedimento de evento `Button1_click`. Role para a parte superior do formulário, onde será declarado duas variáveis do tipo *Double* (embaixo de **Public Class Form1**):

*Dim PrimeiroNumero, SegundoNumero As Double*

11. Role para baixo novamente até o procedimento de evento `Button1_Click`.

Adicione o código a seguir:

```
PrimeiroNumero = TextBox1.Text
SegundoNumero = TextBox2.Text
PrimeiroNumero = TextBox1.Text
SegundoNumero = TextBox2.Text

If RadioButton1.Checked = True Then
    TextBox3.Text = PrimeiroNumero + SegundoNumero

ElseIf RadioButton2.Checked = True Then
    TextBox3.Text = PrimeiroNumero - SegundoNumero

ElseIf RadioButton3.Checked = True Then
    TextBox3.Text = PrimeiroNumero * SegundoNumero

Else : RadioButton4.Checked = True
    TextBox3.Text = PrimeiroNumero / SegundoNumero

End If
```

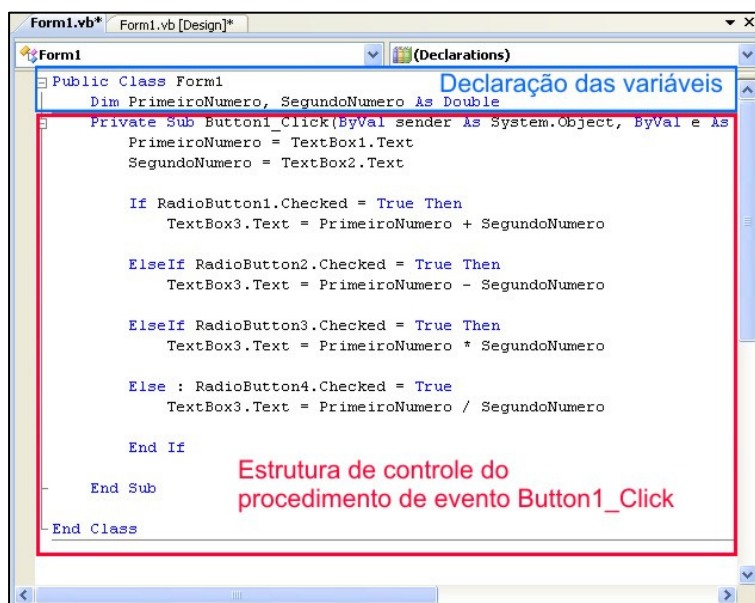


Figura 5 – Codificação do programa

As duas primeiras instruções no procedimento de evento transferem dados inseridos nos objetos caixa de texto para as variáveis *PrimeiroNumero* e *SegundoNumero*.

Depois que os valores de caixa de texto são atribuídos às variáveis, o procedimento de evento determina qual o botão de opção foi selecionado, calcula a fórmula matemática e exibe o resultado em uma terceira caixa de texto.

Somente um *RadioButton* (objeto botão de opção) pode ser selecionado em um Objeto caixa de grupo (*GroupBox*).

Pode dizer se um objeto *RadioButton* foi ou não selecionado avaliando a propriedade *Checked*. Se for *True* (verdadeiro), o botão foi selecionado. Se a propriedade *Checked* for *False* (False), o botão não foi selecionado.

**12.** Volte para *Form1.vb [Design]*. Dê duplo clique no botão sair.

Será aberto o procedimento de evento *Button2\_Click*. Adicione o código:

*End*

**13.** Clique no botão *Save All* na barra de ferramenta *Standard* para salvar seu projeto.

**14.** Agora é só executar o programa.

### 3.2 Select Case

Também pode controlar a execução de instruções em programas utilizando estruturas de controle *Select Case*.

Sintaxe:

Select Case variável

Case valor1

Instruções executadas se valor1 corresponder á variável

Case valor2

Instruções executadas se valor2 corresponder á variável

...

Case Else

Instruções executadas se nenhuma correspondência for encontrada.

End Select

Uma estrutura *Select Case* inicia com as palavras-chave *Select Case* e termina com as palavras-chave *End Select*. Substitui a *variável* pela variável, propriedade ou outra expressão que tiver o valor-chave, ou caso de teste, para estrutura. Substitui valor1, valor2 por números, strings, ou outros valores relacionados ao caso de teste sendo considerado. Se um dos valores corresponderem à variável, a instrução abaixo da cláusula *Case* são executadas e, então, o Visual Basic pula para a linha depois da instrução *End Select* e retoma a execução a partir daí.

### 3.2.1 Programa com estrutura de controle Select Case (Processa entrada de uma caixa de listagem)

1. Com o Visual Studio 2008 aberto, no menu File, clique em New Project.  
A caixa de diálogo New Project abre.
2. Crie um novo projeto Windows Form Application chamado Select Case.
3. Clique no controle Label da ToolBox e desenhe um rótulo próximo à parte superior do formulário para exibir um título para o programa.
4. Utilize o controle Label para criar um segundo objeto rótulo abaixo do primeiro.  
Utilizará esse rótulo como um título para a caixa de listagem.
5. Clique no controle ListBox da ToolBox e crie uma caixa de listagem abaixo do segundo rótulo.
6. Utilize o controle Label para desenhar dois rótulos abaixo da caixa de listagem para exibir a saída do programa.
7. Utilize o controle Button para criar um pequeno botão na parte inferior do formulário.
8. Abra a janela properties e configure as propriedades mostradas a seguir:

Objeto	Propriedade	Configuração
Form1	Text	Saudação
Label1	Font	Arial; 10pt; style=Bold
	Text	Bem Vindo ao Programa Internacional
Label2	Text	Escolha o País

Label3	Text	(Vazia)
Label4	Autosize BorderStyle ForeColor Text	False Fixed3d Red (vazia)
Button1	Text	Sair

O Formulário deverá ficar como este:

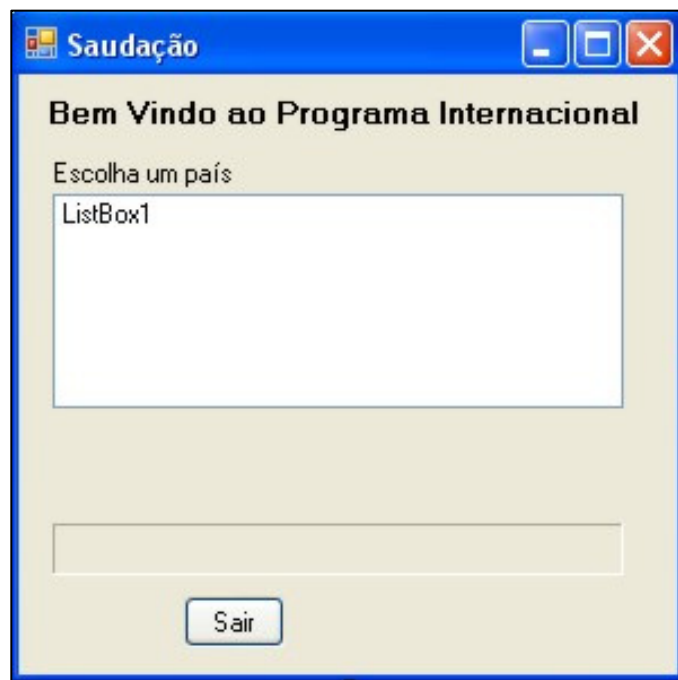


Figura 6 – Formulário para estrutura Case

Agora será inserido o código do programa para inicializar a caixa de listagem

**09.** Vá à janela propriedade da ListBox1 e altere a propriedade *Items*

Dê um clique na reticências e adicione o seguintes itens:

Inglaterra  
Alemanha  
México  
Itália  
Português

**10.** Dê duplo clique no objeto ListBox1 no formulário para editar seu procedimento de evento.

O procedimento de evento *ListBox1\_SelectedIndexChanged* aparece no code editor.

**11.** Digite o seguinte código:

```
Label3.Text = ListBox1.Text
Select Case ListBox1.SelectedIndex
```



```
Case 0
    Label4.Text = "Hello, Programmer"
Case 1
    Label4.Text = "Hallo, Programmierer"
Case 2
    Label4.Text = "Hola, Programador"
Case 3
    Label4.Text = "Ciao, Programmatore"
Case 4
    Label4.Text = "Olá, Programador"
End Select
```

A primeira linha copia o nome do item de caixa de listagem selecionado para a propriedade *Text* do terceiro rótulo no formulário. A propriedade mais importante utilizada na instrução é *Listbox1.Text*, que contém o texto exato do item selecionado na caixa de listagem. As instruções restantes fazem parte da estrutura de decisão *Select Case*. A estrutura utiliza a propriedade *Listbox1.SelectIndex* como uma variável no caso de teste e a compara com vários valores. A propriedade *SelectIndex* sempre contém o número do item selecionado na caixa de listagem:

```
Primeiro Item = 0 (Zero)
Segundo Item = 1
Terceiro Item = 2
```

E assim por diante.

Utilizando a *SelectIndex*, a estrutura *Select Case* pode identificar rapidamente a escolha do usuário e exibir a saudação correta no formulário.

**12.** Exiba o formulário e dê duplo clique no *Button1* (Sair).

O procedimento de evento *Button1\_Click* aparece no code editor.

Digite o código a seguir:

```
End
```

**13.** Clique no botão *Save All* na barra de ferramenta *Standard* para salvar seu projeto.

**14.** Agora execute o Programa.

## 4. Estruturas de Repetição

### 4.1 For...Next

Com um *Loop For...Next*, você pode executar um grupo específico de instruções de programa um número predefinido de vezes em um procedimento de evento ou em um módulo de código.

Sintaxe:

*For* variável = início *To* fim

Instruções a serem repetidas

*Next* [variável]

*For*, *To*, *Next* são palavras-chave obrigatórias. Substitui a *variável* pelo nome de uma variável numérica que monitora a contagem atual do loop (a variável após *Next* é opcional) e substitui o *início* e o *fim* por valores numéricos que representam o ponto inicial e final para o loop. (Deve declarar a variável antes de ela ser utilizada na instrução *For...Next*.) A linha ou linhas entre as instruções *For* e *Next* são as instruções que são repetidas toda vez que o loop é executado.

#### 4.1.2 Programa com estrutura de repetição *For...Next* (Exibindo uma variável contadora em um controle *TextBox*)

1. Com o Visual Studio 2008 aberto, no menu *File*, clique em *New Project*.

A caixa de diálogo *New Project* abre.

2. Crie um novo projeto *Windows Form Application* chamado *For Next*.

3. Dê duplo clique no controle *Button* na *ToolBox*.

4. Arraste o objeto botão para a direita e centralize próximo da parte superior do formulário.

5. Abra a janela *properties* e então configure a propriedade *text* do botão como “*Loop*”.

6. Dê um duplo clique no controle *TextBox* na *ToolBox*.

Configure a propriedade *Multiline* do objeto *TextBox* com *True* e então configure a propriedade *ScrollBars* do objeto caixa de texto com *vertical*.

Essas configurações preparam a caixa de texto para exibir mais de uma linha de texto.

7. Mova caixa de texto para baixo do botão e a expanda ela para que ocupe boa parte do formulário.

O formulário deverá ficar assim:

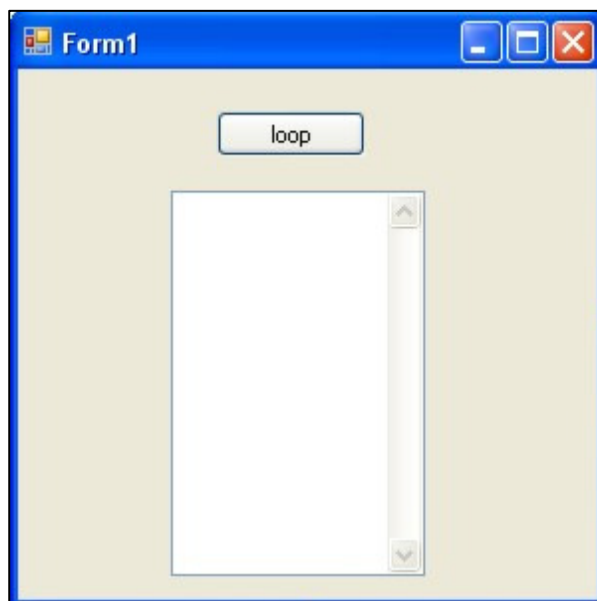


Figura 7 – Formulário do exercício para estrutura de repetição *For/Next*

**8.** Dê duplo clique no botão Loop no formulário.

O procedimento de evento *Button1\_Click* aparece no code Editor.

Digite as seguintes instruções de programa:

```
Dim i As Integer
```

```
Dim Wrap As String
```

```
Wrap = Chr(13) & Chr(10)
```

```
For i = 1 To 10
```

```
    TextBox1.Text = TextBox1.Text & "Linha " & i & Wrap
```

```
Next i
```

Esse procedimento de evento declara duas variáveis, uma do tipo *Integer* ( *i* ), e uma do tipo *string* ( *Wrap* ). Ele então atribui um valor string representando o caractere de retorno de carro à segunda variável.

**Obs:** Caractere de retorno de carro é o equivalente a pressionar a tecla *Enter* do teclado. Foi criada uma variável especial para esse caractere no código de programa, que é composto dos elementos quebra de linha e retorno, para tornar a codificação de um retorno de carro menos complicado. O elemento retorno, *Chr(13)* move o cursor em forma de | para o início da linha. O elemento quebra de linha, *Chr(10)*, remanescente de uma máquina de escrever antiga, move o cursor em forma de | para a próxima linha.

Depois da declaração e atribuição de variável, utilizou um Loop *For..Next* para exibir a linha X 10 vezes na caixa de texto, onde X é o valor atual da variável contadora. Os caracteres de concatenação de strings ( & ) unem as partes componentes de cada linha na caixa de texto. Primeiro, o valor inteiro da caixa de texto, que é armazenado na propriedade *Text*, é adicionado ao objeto de modo que as linhas anteriores não sejam descartadas quando novas forem adicionadas. Em seguida a string "Linha", o número de linha atual e o caractere de retorno (*Wrap*) são combinados para exibir uma nova linha e mover o cursor em forma de | para a margem esquerda e para baixo uma linha. A instrução *Next* Completa o Loop.

**09.** Clique no botão Save All na barra de ferramenta Standard para salvar seu projeto.

**10.** Agora execute o Programa.

O Loop *For..Next* exibe 10 linha na caixa de texto.

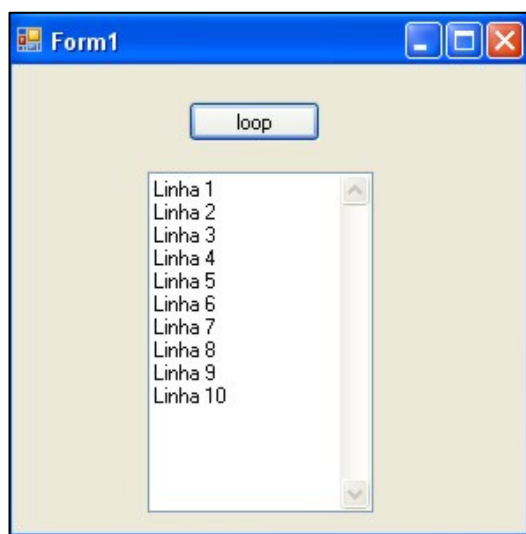


Figura 8 – Execução do programa

## 4.2 While

Use uma estrutura de While quando você desejar repetir um conjunto de instruções um número indefinida de vezes, desde que uma condição permaneça True (verdadeira).

Se desejar mais flexibilidade Utiliza-se a função Do.

Sintaxe básica:

*While Condição*

*Bloco de instruções a ser executado*

*End While*

### 4.2.1 Programa com estrutura de repetição Do (Conversão de temperaturas)

1. Com o Visual Studio 2008 aberto, no menu File, clique em New Project.  
A caixa de diálogo New Project abre.

2. Crie um novo projeto Windows Form Application chamado Temperatura.

3. Dê duplo clique no formulário.

O Procedimento de evento *Form1\_Load* aparece no code editor.

4. Digite as seguintes instruções:

*Dim FTemp, Celsius As Single*

*Dim strFTemp As String*

*Dim Prompt As String = "Entre com a temperatura Fahrenheit."*

*strFTemp = InputBox(Prompt, "Fahrenheit para Celsius")*

*While strFTemp <> ""*

*If strFTemp <> "" Then*

*FTemp = CSng(strFTemp)*

*Celsius = Int((FTemp + 40) \* 5 / 9 - 40)*

```
MsgBox(Celsius, , "Temperatura em Celsius")  
strFTemp = InputBox(Prompt, "Fahrenheit para Celsius")  
End If
```

*End While*

A primeira linha declara duas variáveis de precisão simples, para armazenar as temperaturas Fahrenheit, Celsius. A segunda linha declara uma variável String chamada strFTemp que armazena uma versão string da temperatura Fahrenheit. A terceira linha declara uma variável string chamada Prompt, que será utilizado na função InputBox, e lhe atribui um valor inicial . O *While* solicita repetidamente ao usuário uma temperatura Fahrenheit, converte-a para Celsius e, em seguida, exibe-a na tela utilizando a função *MsgBox*.

Como se trata de um cálculo matemático com o valor inserido, strTemp deve ser convertido em um número. A função CSng é utilizada para converter uma string no tipo de dados Single. CSng é uma de muitas funções de conversão que pode utilizar para converter uma string no tipo de dados diferente. O valor individual convertido então é armazenado na variável Ftemp.

5. Clique no botão Save All na barra de ferramenta Standard para salvar seu projeto.

6. Agora execute o Programa.

### 4.3 Do

Loop DO executa um grupo de instruções até que certa condição seja True (verdadeira). As vezes não dá para saber quantas vezes um loop deve se repetir.

A sintaxe mais comum é:

Do While Condição

Bloco de instruções a ser executado

Loop

#### 4.3.1 Programa com estrutura de repetição Do (Conversão de temperaturas)

1. Com o Visual Studio 2008 aberto, no menu File, clique em Open Project.

A caixa de diálogo OpenProject abre.

2. Selecione a pasta onde foi salvo o projeto anterior chamado Temperatura.

3. Dê duplo clique no formulário.

O Procedimento de evento *Form1\_Load* aparece no code editor.

4. Altere o code editor com as seguintes instruções:

*Dim FTemp, Celsius As Single*

```
Dim strFTemp As String
Dim Prompt As String = "Entre com a temperatura Fahrenheit."
Do
    strFTemp = InputBox(Prompt, "Fahrenheit para Celsius")
    If strFTemp <> "" Then
        FTemp = CSng(strFTemp)
        Celsius = Int((FTemp + 40) * 5 / 9 - 40)
        MsgBox(Celsius, , "Temperatura em Celsius")
    End If
Loop While strFTemp <> ""
End

End Sub
```

O Loop Do solicita repetidamente ao usuário uma temperatura Fahrenheit, converte-a para Celsius e, em seguida, exibe-a na tela utilizando a função *MsgBox*.

5. Clique no botão Save All na barra de ferramenta Standard para salvar seu projeto.
6. Agora execute o Programa.

## 5. Formulários e Menus

### 5.1 Adicionando novos formulários

Se precisar trocar informações adicionais com o usuário de uma maneira personalizada, pode acrescentar formulários adicionais ao programa.

#### Trabalhando com múltiplos formulários

1. Inicie o visual Studio e abra o projeto que foi criado anteriormente *Select Case*.
2. Exiba o formulário principal (form1.vb) no designer.
3. Clique no comando Add Windows Form no menu Project para adicionar um segundo formulário ao projeto.  
Aparecerá a caixa de diálogo mostrada a seguir:

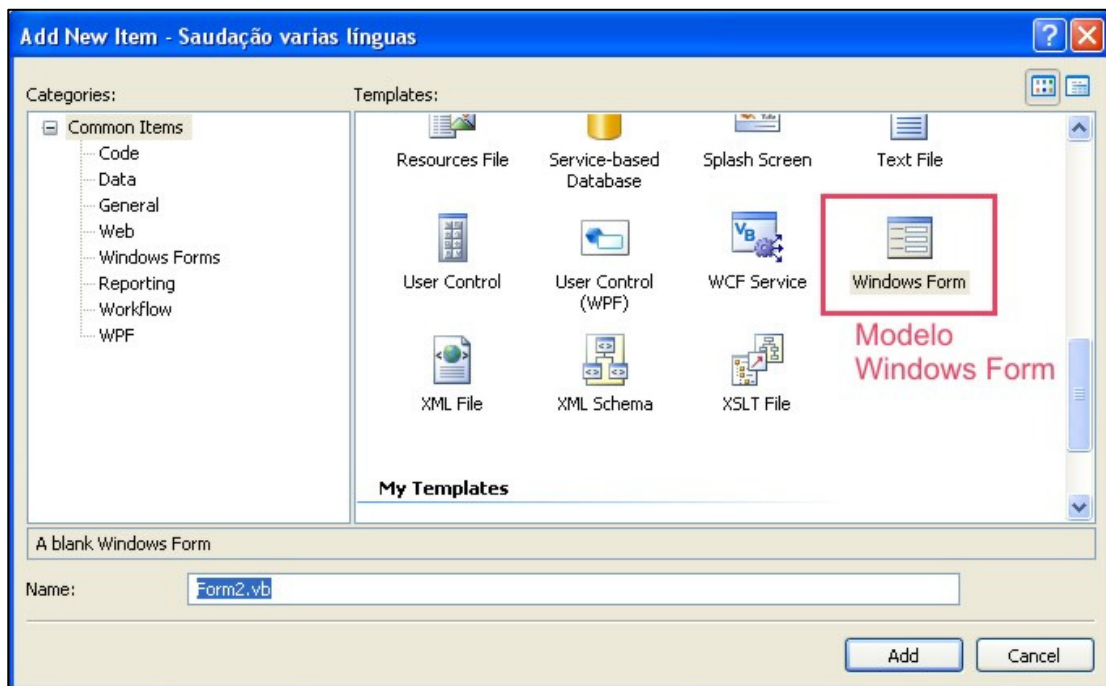


Figura 9 – Incluindo novo formulário ao projeto.

4. Selecione o modelo Windows Form e clique em Add.

Um segundo formulário chamado form2.vb é adicionado ao projeto Select Case e o formulário abre no solution Explorer.

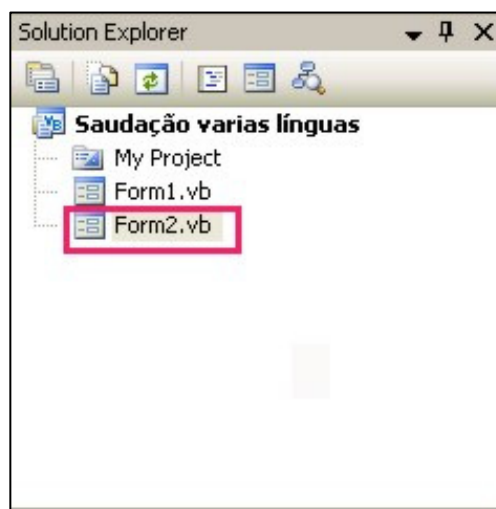


Figura 10 – Selecionando o formulário inserido

Configure a propriedade Text do Form2.vb como “Informações”.

Agora será adicionado alguns controles ao formulário Form2.vb.:

5. Adicione um controle *Label* na parte superior do formulário Form2.vb.

Configure a propriedade Text como “Informações do programa”

**6.** Adicione um controle *TextBox*.

Configure a propriedade *Multiline* com True para poder redimensionar o objeto facilmente.

Configure a propriedade *Scrollbars* com vertical

Redimensione a *TextBox* de modo que ela cubra a maior parte do formulário.

**7.** Adicione um controle *Button* na parte inferior do formulário.

Configure a propriedade *Text* com "OK".

**8.** Dê duplo clique no botão OK para exibir o procedimento de evento *Button1\_Click* no code editor.

Adicione o seguinte código:

*End*

**9.** Dê duplo clique no formulário *Form2.vb* para abrir o procedimento de evento *Form2\_Load*.

Adicione o seguinte código:

```
TextBox1.Text = "O PROGRAMA EXIBE A SAUDAÇÃO EM INGLÊS, ALEMÃO, MEXICANO, ITALIANO E PORTUGUÊS"
```

**10.** Vá para o *Form1.vb[Design]*.

**11.** Adicione um controle *Button* na parte inferior do lado direito do formulário.

Configure a propriedade *Text* do Botão como Info.

**12.** Dê duplo clique no botão Info.

Será aberto o procedimento de evento *Button2\_Click*

Adicione o seguinte código:

```
Form2.Show()
```

**13.** Clique no botão *Save All* na barra de ferramenta *Standard* para salvar seu projeto.

**14.** Agora execute o Programa.

## 5.2 Adicionando Menu

O controle *Menu Strip* é uma ferramenta que adiciona menus aos programas. Com o *MenuStrip*, pode adicionar novos menus, modificar, reordenar menus existentes, excluir menus antigos entre outras coisa.

Como os outros controles, o *MenuStrip* cria a parte visível dos menus, ainda é preciso escrever procedimentos de evento que processam suas funcionalidades.

### Trabalhando com Menus

**1.** Com o Visual Studio 2008 aberto, no menu *File*, clique em *New Project*.



A caixa de diálogo New Project abre.

2. Crie um novo projeto Windows Form Application chamado Menu.

3. Crie um controle Label e altere suas propriedades para:

AutoSize = False

BorderStyle = FixedSingle

Font = Arial; 12pt; style=Bold

ForeColor = azul

Size = 157;23

Text = deixar vazio

TextAlign = MiddleCenter

4. Crie um controle *MenuStrip* na ToolBox.

O Visual Studio cria um objeto MenuStrip na bandeja de componentes (nele pode configurar suas propriedades ou excluí-los a partir desse painel).

O Visual Studio configura a localização do menu automaticamente no Formulário.

5. Clique na Tag Type Here.

Digite Data e Hora e pressione Enter.

A palavra Data e Hora é inserida como nome de seu primeiro menu e dois tags Type Here adicionais aparecem. Com elas pode criar itens de submenu ou título de menu adicionais.

6. Crie um comando Data para o menu Relógio e pressione Enter.

7. Na tag abaixo de Data, crie um comando Hora para o menu Relógio e pressione Enter.

O Formulário deverá ficar assim:

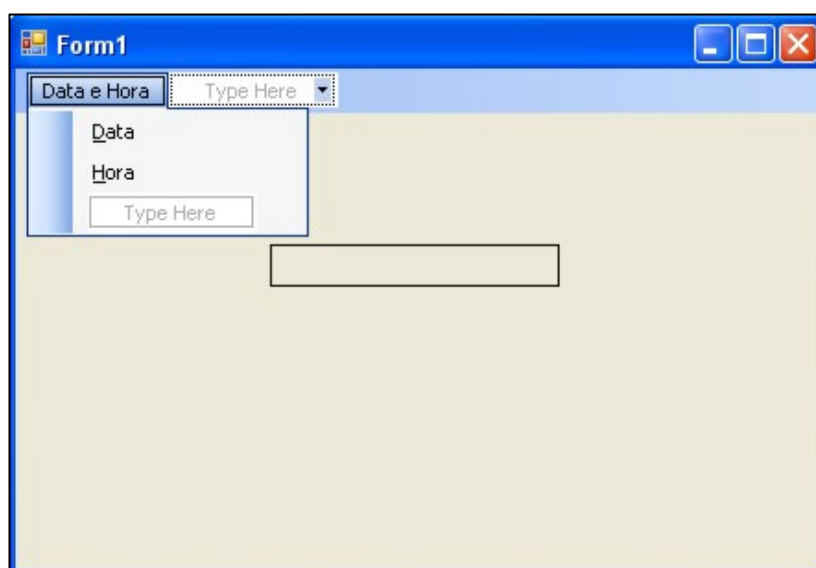


Figura 11 – Formulário com itens de menu

O menu está criado, agora é só escrever os procedimentos de evento que processam suas funcionalidades.

**8. Dê um clique no menu Data e Hora.**

Selecione a tag Data e dê duplo clique. O procedimento de evento *DataToolStripMenuItem\_Click* abre no code editor.

Digite a seguinte instrução:

*Label1.Text = DateTime.Now.ToString("dd/MM/yyyy")*

Essa instrução de programa exibe a data atual (do relógio do sistema) na propriedade Text do objeto Label1 quando o comando Data do menu Data e Hora é clicado.

**9. Dê um clique no menu Data e Hora.**

Selecione a tag Hora e dê duplo clique. O procedimento de evento *HoraToolStripMenuItem\_Click* abre no code editor.

Digite a seguinte instrução:

*Label1.Text = DateTime.Now.ToString("HH:mm:ss")*

Essa instrução de programa exibe a hora atual (do relógio do sistema) na propriedade Text do objeto Label1 quando o comando Hora do menu Data e Hora é clicado.

Também pode adicionar teclas de atalho para abrir os comandos do menu Data e Hora

Ex: Selecione o comando Data do menu Data e Hora. Vá na janela properties e altere a propriedade ShortcutKeys.

Clique na seta do campo onde se encontra o valor da propriedade.

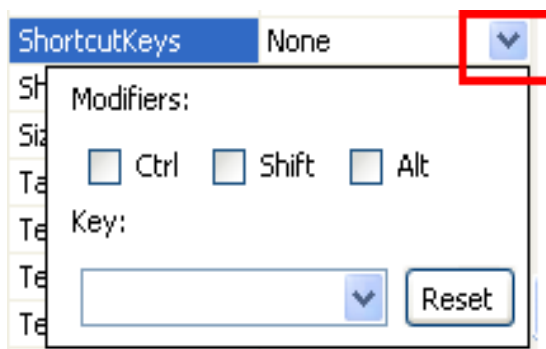


Figura 12 – Alterando a propriedade ShortcutKeys

Em Modifiers selecione Ctrl.

Em Key selecione D.

Agora para executar o comando Data não é necessário ir até o Menu Data e Hora, é só pressionar Ctrl + D.

10. Clique no botão Save All na barra de ferramenta Standard para salvar seu projeto.

11. Agora execute o Programa.

## 6. Trabalhando com vários tipos de controles da Toolbox

### (Pequeno programa de Compras)

Esse capítulo será abordado conceitos já visto e alguns conceitos novos.

O programa é capaz de buscar os produtos e mostrar a imagem do produto. Também é possível fazer uma simulação de compra. O cliente escolhe o produto e a forma de pagamento , então o programa gera seu preço à vista, o valor da parcela e o valor à prazo.

### 6.1 Construindo o programa de compras

1. Com o Visual Studio 2008 aberto, no menu File, clique em New Project.

A caixa de diálogo New Project abre.

2. Crie um novo projeto Windows Form Application chamado Compra.

3. Altere a propriedade do Formulário como a abaixo:

Size = 615; 412

Text = Produtos

4. Crie 3 controles Label. Altere a propriedade como abaixo:

#### **Label1**

Text = Digite 1 para Computador

Location = 12; 58

#### **Label2**

Text = Digite 2 Para Calculadora

Location = 12; 83

#### **Label3**

Text = Digite 3 Para Cartucho

Location = 12; 109

5. Crie um controle TextBox. Altere a propriedade como abaixo:

Location = 12; 144

Size = 49; 20

6. Crie três controle Button. Altere a propriedade como abaixo:

#### **Button1**

Location = 12; 170

Text = Buscar

## Button2

Location = 12; 245

Text = Comprar

## Button3

Location = 169; 245

Text = Sair

7. Iremos criar um controle PictureBox. Esse controle ainda não foi utilizado, ele se encontra também na ToolBox.

O controle PictureBox serve para adicionar imagem ao programa.

Altere a propriedade do controle PictureBox como abaixo:

Location = 320; 67

Size = 250; 200

SizeMode = StretchImage

Visible = False

O Formulário deve ficar como a imagem abaixo.

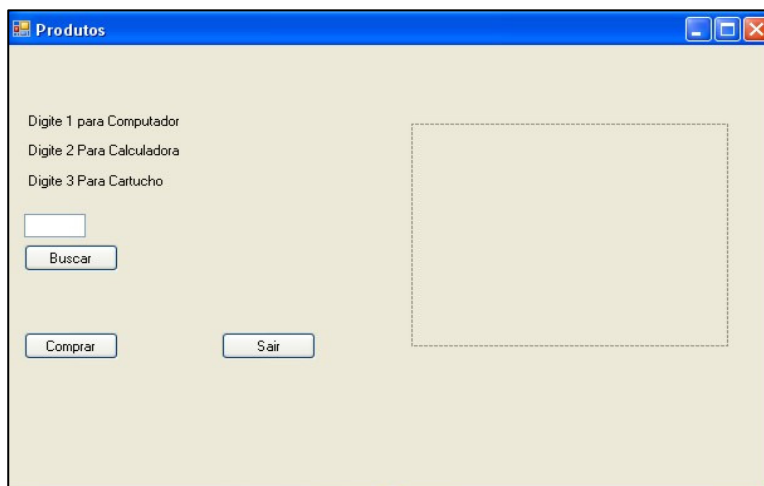


Figura 13 – Formulário do programa de compras

O primeiro formulário já está pronto. Antes de escrever os procedimentos de eventos iremos criar outro formulário, o formulário para a simulação de compras

8. Clique no comando Add Windows Form no menu Project para adicionar um segundo formulário ao projeto.

9. Altere a propriedade do Formulário como a abaixo:

Size = 615; 412

Text = Compras

10. Crie 8 controles Label. Altere a propriedade com a tabela abaixo:

Controle	Propriedade
Label1	Location = 13; 52

	Text = Escolha o Produto
Label2	Location = 377; 302 Text = Preço a vista:
Label3	Location = 455; 302 Text = 0
Label4	Location = 471; 330 Text = 0
Label5	Location = 13; 140 Text = Forma de Pagamento
Label6	Location = 377; 330 Text = Valor da Parcela:
Label7	Font = 6,75pt Location = 389; 363 Text = Total a prazo:
Label8	Font = 6,75pt Location = 455; 363 Text = 0

11. Crie dois controle ComboBox. Esse controle ainda não foi utilizado, ele se encontra também na ToolBox.

#### Adicionando itens á ComboBox:

Na propriedade Items da ComboBox, dê um clique na reticências.

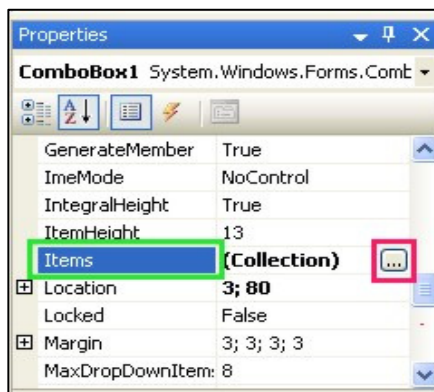
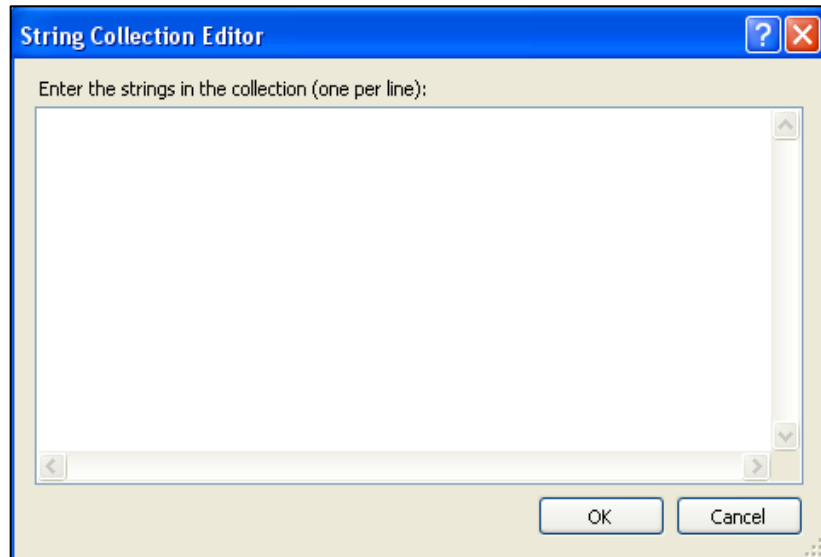


Figura 14 – Propriedades da ComboBox.

Irá abrir a janela para adicionar os itens.



Os itens devem ser adicionados um um embaixo do outro.

**Altere as propriedades como abaixo:**

**ComboBox1**

Items

COMPUTADOR

CALCULADORA

CARTUCHO

Location = 3; 80

**ComboBox2**

Items

1x sem juros

2x sem juros

3x sem juros

4x com juros

5x com juros

Location = 3; 165

**12.** Crie dois controles Button. Altere as propriedades como abaixo:

**Button1**

Location = 12; 292

Text = Finalizar Compra

**Button2**

Location = 12; 325

Text = Voltar

**13.** Crie um controle PictureBox. Altere as propriedades como abaixo:

Location = 320; 67

Size = 250; 200

SizeMode = StretchImage

Visible = False

O segundo formulário está pronto. Ele deve ficar com a imagem abaixo:

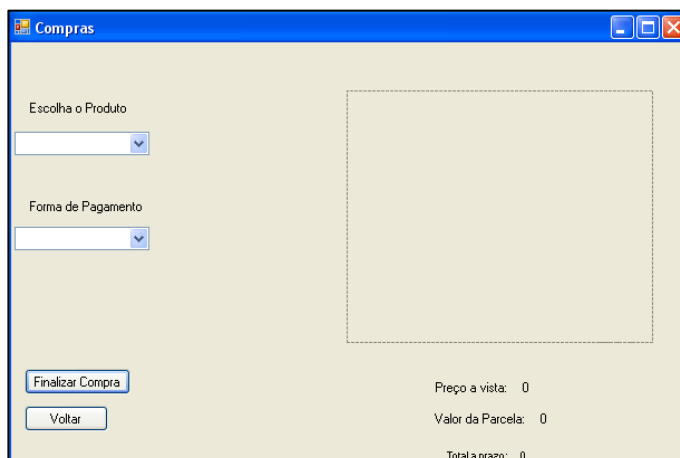


Figura 14 – Fomulário de compras

Agora é necessário escrever as instruções para o programa. Volte para o Form1.vb[Design].

14. Dê duplo clique no botão Buscar. O procedimento de evento *Button1\_Click* abre no code editor. Escreva o seguinte código:

```
Select Case TextBox1.Text
    Case 1
        PictureBox1.Load("C:\Documents and Settings\Nupeds03\Meus documentos\Minhas
imagens\computador.jpg")
        PictureBox1.Visible = True
    Case 2
        PictureBox1.Load("C:\Documents and Settings\Nupeds03\Meus documentos\Minhas
imagens\calculadora.jpg")
        PictureBox1.Visible = True
    Case 3
        PictureBox1.Load("C:\Documents and Settings\Nupeds03\Meus documentos\Minhas
imagens\cartucho.jpg")
        PictureBox1.Visible = True
End Select
```

#### Leitura do código:

Caso o texto do controle TextBox1 for 1, no controle PictureBox1 aparecerá a imagem do computador.  
 Caso o texto do controle TextBox1 for 2, no controle PictureBox1 aparecerá a imagem da calculadora.  
 Caso o texto do controle TextBox1 for 3, no controle PictureBox1 aparecerá a imagem do cartucho.

**Método Load** = O método Load é utilizado para carregar a imagen na PictureBox1.

**Método Visible** = O método Visible é utilizado para que a imagem seja visualizada na PictureBox1.

15. Volte para o Form1.vb[Design]. Dê duplo clique no botão Sair. O procedimento de evento *Button3\_Click* abre no code editor. Escreva o seguinte código:

```
Close()
```

16. Volte para o Form1.vb[Design]. Dê duplo clique no botão Comprar. O procedimento de evento *Button2\_Click* abre no code editor. Escreva o seguinte código:

```
Form2.Show()
```

As instruções do Primeiro Formulário (Form1) já estão prontas. Agora só falta instruções do segundo Formulário (Form2).

17. Com o Form2 aberto. Dê duplo clique na ComboBox1. O procedimento de evento *ComboBox1\_SelectedIndexChanged* é aberto no formulário, escreva o seguinte código:

```
Select Case ComboBox1.Text
    Case "COMPUTADOR"
        PictureBox1.Load("C:\Documents and Settings\Nupeds03\Meus documentos\Minhas
imagens\computador.jpg")
        PictureBox1.Visible = True
        Label3.Text = CStr("900,00")
    Case "CALCULADORA"
        PictureBox1.Load("C:\Documents and Settings\Nupeds03\Meus documentos\Minhas
imagens\calculadora.jpg")
        PictureBox1.Visible = True
        Label3.Text = CStr("130,00")
    Case "CARTUCHO"
        PictureBox1.Load("C:\Documents and Settings\Nupeds03\Meus documentos\Minhas
imagens\cartucho.jpg")
        PictureBox1.Visible = True
        Label3.Text = CStr("90,00")
End Select
```

#### Leitura do código:

Caso seja selecionado o texto “COMPUTADOR” do controle ComboBox1, no controle PictureBox1 aparecerá a imagem do computador e o texto no controle Label3 será “900,00”.

Caso seja selecionado o texto “CALCULADORA” do controle ComboBox1, no controle PictureBox1 aparecerá a imagem da calculadora e o texto no controle Label3 será “130,00”.

Caso seja selecionado o texto “CARTUCHO” do controle ComboBox1, no controle PictureBox1 aparecerá a imagem do cartucho e o texto no controle Label3 será “90,00”.

**Obs:** CStr é utilizado para converter os números em string.

18. Volte para o Form2.vb[Design]. Dê duplo clique no controle ComboBox2. O procedimento de evento *ComboBox2\_SelectedIndexChanged* é aberto no formulário. Role para cima até *Public Class Form2*, pois será necessário declarar alguns valores. Declare os seguintes valores:

```
Dim x = 0.03
Dim y = 0.05
Dim z = 0.06
Dim A As Decimal
```



Volte para o procedimento de evento `ComboBox2_SelectedIndexChanged`. Escreva o seguinte código:

*Select Case ComboBox2.Text*

```
Case "1x sem juros"
    Label4.Text = Label3.Text
    Label8.Text = Label3.Text
Case "2x sem juros"
    Label4.Text = (Label3.Text) / 2
    Label8.Text = Label3.Text
Case "3x sem juros"
    A = (Label3.Text) / 3
    A = Math.Round(A, 2)
    Label4.Text = A
    Label8.Text = Label3.Text
Case "4x com juros"
    A = (Label3.Text + (Label3.Text * x)) / 4
    A = Math.Round(A, 2)

    Label4.Text = A
    Label8.Text = (Label3.Text + (Label3.Text * (4 * x)))
Case "5x com juros"
    A = (Label3.Text + (Label3.Text * y)) / 5
    A = Math.Round(A, 2)

    Label4.Text = A
    Label8.Text = (Label3.Text + (Label3.Text * (5 * y)))
End Select
```

#### **Leitura do código:**

Caso seja selecionado o texto “1x sem juros” do controle `ComboBox1`, o valor da `label4` e `label8` será igual ao valor da `label3`.

Caso seja selecionado o texto “2x sem juros” do controle `ComboBox1`, a `label4` recebe o valor da `label3`, e esse valor é dividido por 2 e a `label8` também recebe o valor da `label3`, mas não ocorre a operação matemática divisão.

Caso seja selecionado o texto “3x sem juros” do controle `ComboBox1`, a variável “A” recebe o valor da `label3`, e esse valor é dividido por 3, o valor dessa variável é arredondado para o número especificado de casas decimais, a `Label4` recebe o valor da variável “A” já arredondado, e a `label8` também recebe o valor da `label3`, mas não ocorre a operação matemática divisão.

Caso seja selecionado o texto “4x com juros” do controle `ComboBox1`, a variável “A” recebe o valor da `label3` somada com o juros (valor do juros declarado na variável x) e esse valor é dividido por 4, o valor dessa variável é arredondado para o número especificado de casas decimais, a `Label4` recebe o valor da variável “A” já arredondado, e a `label8` também recebe o valor da `label3`, mas somada com o juros multiplicado em 4.

Caso seja selecionado o texto “5x com juros” do controle `ComboBox1`, a variável “A” recebe o valor da `label3` somada com o juros (valor do juros declarado na variável y) e esse valor é dividido por 5, o valor dessa variável é arredondado para o número especificado de casas decimais, a `Label4` recebe o valor da

variável "A" já arredondado, e a label8 também recebe o valor da label3 somada com o juros multiplicado em 5.

**Obs:** Math.Round = Arredonda um valor para o inteiro mais próximo ou número especificado de casas decimais.

19. Volte para o form2.vb[Design]. Dê duplo clique no botão Finalizar Compra, o procedimento de evento *Button1\_Click* abre no code editor.

Digite o seguinte código:

```
MsgBox("COMPRA FINALIZADA")
```

Quando o botão Finalizar Compra for clicado, uma janela de mensagem aparecerá na tela com a frase "COMPRA FINALIZADA".

20. Volte para o form2.vb[Design]. Dê duplo clique no botão voltar, o procedimento de evento *Button2\_Click* abre no code editor.

Digite o seguinte código:

```
Close()
```

21. Clique no botão Save All na barra de ferramenta Standard para salvar seu projeto.

22. Agora execute o Programa.

## 6.2 Outro método para se fazer uma Calculadora

1. Com o Visual Studio 2008 aberto, no menu File, clique em New Project.

A caixa de diálogo New Project abre.

2. Crie um novo projeto Windows Form Application chamado Calc.

3. Altere a propriedade do Formulário como a abaixo:

Size = 328; 358

Text = Calculadora

4. Crie um controle TextBox. Altere as propriedades como abaixo:

Size = 290; 20

Location = 18; 34

5. Crie 10 controles Button. Altere as propriedades como abaixo:

Text button0 à button9 = 0 à 9.

Name button0 à button9 = btnZero, btnUm, btnDois, btnTres, btnQuatro, btnCinco, btnSeis, btnSete, btnOito e btnNove.

Size (para button1 à button9) = 50; 23.

Size (button10) = 162; 23.

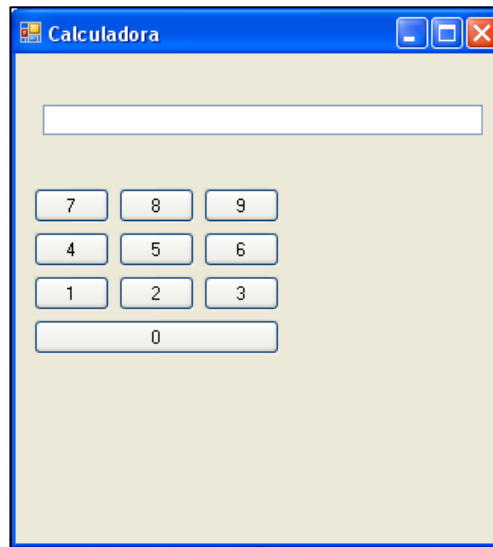


Figura 16 – Formulário da calculadora

**6.** Crie mais 5 controle Button. Altere a propriedade como abaixo:

Text (button11 à button 15) = na seguinte ordem + - \* / %.

Name (button11 à button 15) = na seguinte ordem Adicao, Subtracao, Multiplicacao, Divisao, Porcent.

Size (button11 à button15) = 50; 23.

**7.** crie mais 1 controle Button. Altere a propriedade como abaixo:

Text “ = “

Name = Igual

Size = 66; 52

Organize os controles no formulário para que fiquem como a imagem abaixo:

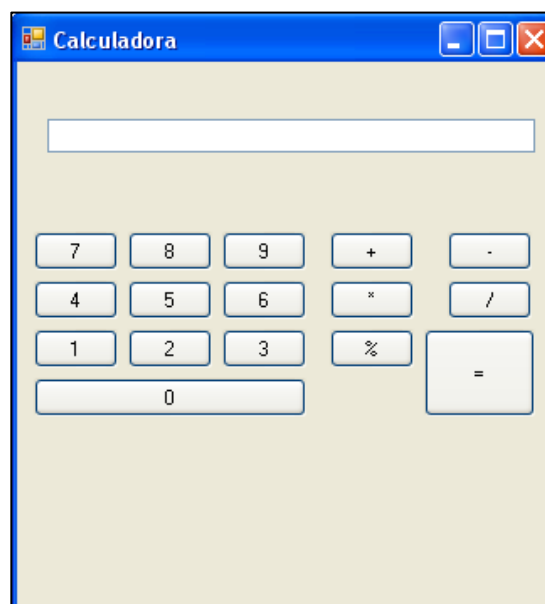


Figura 17 – Formulário da calculador com todos os botões

O Formulário está pronto agora é necessário adionar as funcionalidades dos controles.

**8.** Dê duplo clique no Formulário. Será aberto o procedimento de evento *Form1\_Load*.

Embaixo de Public Class Form1, escreva o seguinte código:

```
Dim limpaExibicao As Boolean  
Dim operando1, operando2 As Double  
Dim operador As String  
Dim operador2 As String  
Dim porcentagem As Double
```

Foram declaradas 6 variáveis.

limpaExibição do tipo Booleana;

operando1, operando2 e porcentagem do tipo Double;

operador, operador2 do tipo String.

**9.** Volte para o Form1.vb[Design]. Dê duplo clique no botão 0 (Zero). Será aberto o procedimento de evento *btnZero\_Click*.

Escreva o seguinte código:

```
If limpaExibicao Then  
    TxtTela.Clear()  
    limpaExibicao = False  
End If  
    txtTela.Text = TxtTela.Text + sender.Text
```

Escreva o mesmo código para os botões 1,2,3,4,5,6,7,8,9.

**10.** Volte para o Form1.vb[Design]. Dê duplo clique no botão + . Será aberto o procedimento de evento *Adicao\_Click*.

Escreva o seguinte código:

```
operando1 = Val(TxtTela.Text)  
operador = "+"  
LimpaExibicao = True
```

Dê duplo clique no botão - . Será aberto o procedimento de evento *Subtracao\_Click*.

Escreva o seguinte código:

```
operando1 = Val(TxtTela.Text)  
operador = "-"  
LimpaExibicao = True
```

Dê duplo clique no botão \* . Será aberto o procedimento de evento *Multiplicacao\_Click*.

Escreva o seguinte código:

```
operando1 = Val(TxtTela.Text)
operador = "*"
limpaExibicao = True
```

Dê duplo clique no botão / . Será aberto o procedimento de evento *Divisao\_Click*.

Escreva o seguinte código:

```
operando1 = Val(TxtTela.Text)
operador = "/"
limpaExibicao = True
```

Dê duplo clique no botão % . Será aberto o procedimento de evento *Porcent\_Click*.

Escreva o seguinte código:

```
operando2 = Val(TxtTela.Text)
operador2 = "%"
limpaExibicao = True
```

Quando o botão "+" for clicado a variável *operando1* recebe o valor que está no controle *TxtTela.Text*. A variável *operador* recebe um caractere "+" e a variável *limpaExibicao*, como o nome já diz, limpa o controle *TxtTela.Text*.

A mesma coisa acontece para O botão -, \* e / o que muda é o caractere que a variável *operador* recebe. No caso da porcentagem (%) a variável *operador2* é que recebe o caractere %.

**Função Val:** Retorna os números contidos numa cadeia de caracteres como um valor numérico do tipo apropriado.

11. Volte para o *Form1.vb[Design]*. Dê duplo clique no botão = . Será aberto o procedimento de evento *Igual\_Click*.

Escreva o seguinte código:

*Dim resultado As Double*

```
operando2 = Val(TxtTela.Text)
```

```
If operador2 = "%" Then
```

```
    If operador = "+" Then
```

```
        porcentagem = operando2 / 100
```

```
        resultado = operando1 * (porcentagem + 1)
```

```
    ElseIf operador = "-" Then
```

```
        porcentagem = operando2 / 100
```

```
        resultado = operando1 - (operando1 * porcentagem)
```

```
    Else
```

```
        porcentagem = operando2 / 100
```

```
        resultado = operando1 * porcentagem
```

*End If*

```
Elseif operador = "+" Then
    resultado = operando1 + operando2
Elseif operador = "-" Then
    resultado = operando1 - operando2
Elseif operador = "*" Then
    resultado = operando1 * operando2
Else
    If operando2 <> "0" Then
        resultado = operando1 / operando2
    Else
        TxtTela.Text = "ERRO: Divisão por zero"
        limpaExibicao = True
```

Foi declarada uma variável resultado do tipo Double.

## Cálculo de Porcentagem

O programa verifica se o botão Porcentagem foi clicado.

O usuário entra com um valor, após isso é preciso clicar nos botões + , - ou \* , depois será necessário entrar com outro valor na TxtTela, o operando2 recebe esse segundo valor.

Após entrar com o segundo valor é necessário clicar no botão porcentagem. Se o botão porcentagem for clicado, então o programa verifica o valor do operador se é +, - ou \* para fazer ou seguinte cálculo:

Se o operador for igual a +

```
porcentagem = operando2 / 100
resultado = operando1 * (porcentagem + 1)
```

Se o operador for igual a –

```
porcentagem = operando2 / 100
resultado = operando1 - (operando1 * porcentagem)
```

Senão

```
porcentagem = operando2 / 100
resultado = operando1 * porcentagem
```

Quando o usuário clicar nos botões + , - , \* , / ou % , será necessário entrar com outro valor na TxtTela, o operando2 recebe esse segundo valor.

Anteriormente foi declarado uma variável chamada operador, conforme o usuário clica nos botões + , - , \* ou / , essa variável recebe um caractere.

Quando o botão "=" é clicado o programa verifica o caractere do operador, caso o caractere do operador for "+ , - ou \* " , a variável resultado recebe o valor do operando1 somado, subtraído ou multiplicado com o valor do operando2 e a TxtTela recebe o valor da variável resultado.

caso o caractere do operador for "/", o programa verifica se o valor da variável operando2 é diferente de 0(zero). Se for diferente de 0(Zero), a variável resultado recebe o valor do operando1 e divide pelo valor do operando2 e a TxtTela recebe o valor da variável resultado, senão, a TxtTela mostra a seguinte frase "ERRO: Divisão por zero".

## 7. Banco de Dados

### 7.1 A biblioteca ADO

ADO é baseada em uma tecnologia proposta pela Microsoft: o OLE DB ( Object Linking and Embedding Data Base), que por sua vez é parte de um conjunto mais amplo: o COM (Component Object Model). Assim um objeto OLE também é um objeto COM.

Cinco mais importantes objetos do ADO:

1. Connection
2. Recordset
3. Field
4. Command
5. Error

### 7.2 Objeto Connection

Este objeto faz conexão com a fonte de dados através do nome de um provedor OLE DB ou uma DSN de ODBC. O objeto Connection possui métodos e propriedades que devem ser empregadas em uma conexão. A tabela a seguir mostra os principais métodos deste componente para um objeto definido como objConn.

*Public objCon as ADODB.Connection*

Adicionalmente o objeto Connection possui duas coleções:

Erros: Contém os erros gerados na conexão

Properties: Contém o conjunto de propriedades para uma conexão.

Método	Exemplo	Descrição
Open	objCon.Open strCon	Tenta abrir uma conexão definida na string strCon
Close	objCon.Close	Fecha a conexão aberta e todos os componentes que dela dependem
Execute	objCon.Execute "instrução"	Executa uma instrução SQL, uma consulta, um comando, ou uma stored procedure.
BeginTrans	objCon.BeginTrans	Inicia uma transação no banco de dados
ComminTrans	objCon.CommnTrans	Salva as alterações e encerra uma conexão iniciada.



A seguir algumas propriedades do objeto Connection da ADO:

Propriedade	Exemplo	Descrição
Provider	objCon.Provider	Indica o nome do provedor de acesso à fonte de dados
State	ObjCon.State	Retorna o status da conexão, indicando se foi ativada (1) ou não (0).
ConnectionString	objCon.ConnectionString	Contém informação da string de banco de dados.
Mode	ObjCon.Mode	Indica as permissões que estão disponíveis em uma conexão

### 7.3 O Objeto Recordset

Um recordset representa um conjunto de registros de uma tabela gerado através de uma consulta. Como no objeto Connection, o Recordset da ADO se comporta de maneira semelhante ao Recordset da DAO. É possível criar este objeto sem ter que criar previamente o objeto Connection, e se o objeto recordset for usado para manipular dados no provedor, esse conjunto de registro é denominado cursor.

A tabela a seguir mostra os principais métodos de recordset criado através da sintaxe:

*Public ObjTab as ADODB.Recordset*

Método	Exemplo	Descrição
AddNew	objTab.AddNew	Cria um novo registro na tabela
Close	ObjTab.Close	Fecha um recordset
Delete	ObjTab.Delete	Exclui o registro atual
Move	ObjTab.Move	Muda a posição do registro atual
Open	ObjTab.Open	Abre um recordset (cursor)

Algumas propriedades do objeto Recordset

Objeto	Exemplo	Descrição
Bof	objTab.Bof	Indica se atingiu o início do arquivo
Eof	ObjTab.Eof	Indica se o cursor atingiu o fim do arquivo
RecordCount	ObjTab.RecordCount	Indica o número de registros de um recordset
CursorType	ObjTab.CursorType	Indica o tipo de cursor do recordset

**Exemplo de uso de um Recordset:**

```
Dim ObjCon As ADODB.Connection
Dim ObjTab As ADODB.Recordset
Dim strCon as String
Dim strSQL as String
Set objCon = New Connection
strCon = "Provider = MyProv; User ID = MyID; password = MySenha; DataSource = MinhaDSN"
strSql = "Select * from Empregados"
Set ObjTabEmp = New Recordset
ObjTab.Open strSQL, objCon, adOpenKeyset
```

**7.4 Objeto Field**

O objeto Field da ADO representa a coluna (campo) de um objeto RecordSet. Todos os objetos Field estão contidos na coleção "Fields" de um Recordset.

**7.5 Objeto OleDbCommand**

Este objeto contém definição de uma instrução SQL ou de uma Stored Procedure armazenada num servidor, ou ainda uma ação que deve ser executada em uma fonte de dados. Para criar um objeto Command com o nome de objComm a sintaxe é a seguinte:

```
Dim objCmd As ADODB.Command
```

Neste objeto a propriedade CommandText armazena o texto que contém a instrução SQL.

Ex: objCmd.CommandText

**7.6 Objeto Error**

O objeto Error, auxilia na depuração de erros gerados durante operações que envolvam objetos da ADO. A sintaxe para criar um objeto deste tipo é a seguinte:

```
Dim objErr as ADODB.Error
```

**8. A Biblioteca ADO.NET****8.2 O que é o ADO.NET**

Segundo site MSDN ([www.msdn.com.br](http://www.msdn.com.br)), ADO.NET são várias classes que expõe serviços de acesso à dados para o programador .NET. O ADO.NET fornece um rico conjunto de componentes para a criação de aplicações distribuídas e compartilhamento de dados. O ADO.NET é parte integrante do NET Framework, que dá acesso relacional ao XML e à aplicação de dados. ADO.NET suporta uma variedade de

necessidades de desenvolvimento, incluindo a criação de front-end com base de dados clientes e objetos empresariais usados pelas aplicações, ferramentas, linguagens, ou navegadores de Internet.

Fonte: [http://msdn.microsoft.com/pt-br/library/e80y5yhx\(VS.80\).aspx](http://msdn.microsoft.com/pt-br/library/e80y5yhx(VS.80).aspx)

A diferença básica é que o ADO.NET é uma evolução do ADO tradicional e mais poderosa. Agora é possível trabalhar com dados desconectados da fonte de dados, pois os dados são manipulados através de um **DataSet**, não necessitando de acesso ao disco. ADO.Net também manipula e transmite dados no formato XML o que facilita a troca de documentos.

### 8.3 Componentes e objetos do ADO.NET

Componentes são controles pré-construídos que podem ser instanciados através do process de arrastar e soltar o mouse para dentro do formulário.

Os elementos do tipo **SQL** são empregados exclusivamente para acesso à base de dados SQL Server, e os elementos do tipo **OleDB** são empregados para acesso a outros tipos de base de dados que possuam um provedor **OleDB**, como por exemplo Access e Oracle.

Os namespaces primários do ADO.NET são **System.Data**, **System.Data.OleDb** e **System.Data.SqlClient**. Esses namespaces contêm classes para trabalhar com banco de dados e outros tipos de fonte de dados como os arquivos XML. **System.Data** é o namespace raiz da API do ADO.NET. **System.Data.OleDb** e **System.Data.SqlClient** contêm classes que habilitam os programadores a conectar-se com fontes de dados e modificá-las.

As instâncias da classe **System.Data.DataSet** que consistem em um conjunto de **DataTables** e relações entre esses **DataTables** representam um cache de dados, ou seja, dados que o programa armazena temporariamente na memória local. Uma vantagem de usar o **DataSet** é que ela é desconectada, o programa não necessita de uma conexão persistente ao banco de dados para trabalhar com os dados em um **DataSet**.

As instâncias da classe **OleDbConnection** do namespace **System.Data.OleDb** representam uma conexão a uma fonte de dados. As instâncias da classe **OleDbDataAdapter** conectam-se a uma fonte de dados através de uma instância da classe **OleDbConnection** e podem preencher os **DataSets** com os dados de uma fonte de dados.

## 8.4 Arquitetura da ADO. NET

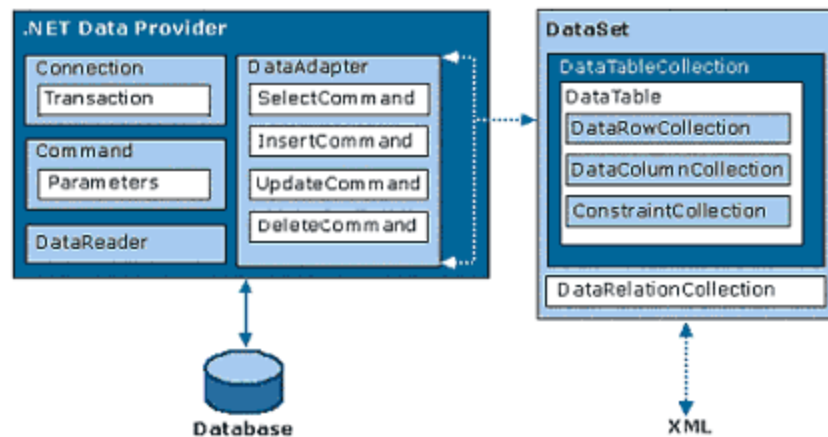


Figura 18 Arquitetura dos componentes da plataforma.NET  
(fonte: [http://www.msdnbrasil.com.br/tecnologias/adonet\\_intro.aspx](http://www.msdnbrasil.com.br/tecnologias/adonet_intro.aspx))

### 8.4.1 Objetos de acesso a uma base de dados

Os principais objetos do tipo SQL são:

- SqlConnection
- SqlCommand
- SqlCommandBuilder
- SqlDataAdapter
- SqlDataSource

Para o tipo OleDb existem os tipos principais a seguir:

- OleDbConnection
- OleDbCommand
- OleDbDataReader
- OleDbDataAdapter
- OleDbParameter

Além destes objetos existem outros que auxiliam na manipulação/exibição dos dados armazenados nas bases de dados:

- DataSet: Representa uma coleção de tabelas na memória
- DataTable: Representa uma tabela
- DataRow: Representa um registro da tabela
- DataColumn: Representa uma coluna (campo) de uma tabela
- DataView: Representa uma visão dos registros de uma tabela.

**Connection:** Classe de conexão com sua fonte de Dados.

**Command:** Classe de execução de declarações SQL e retorno dos Dados da fonte para o cliente.

**DataReader:** Classe que fornece a listagem dos Dados recebidos de uma fonte de Dados. Para quem conhece o ADO podemos comparar o DataReader com o Recordset, porém o DataReader possui cursor somente forward-only. Podemos dizer que o DataReader é o nosso leitor dos dados recebidos de uma fonte de dados. O DataReader necessita de um objeto Connection em estado de conexão aberta para poder realizar suas operações.

Essas classes terão seus nomes variados de acordo com sua opção de namespace utilizado para acessar sua fonte de Dados.

Se você estiver trabalhando com uma fonte de dados SQLServer podemos utilizar o namespace: System.Data.SqlClient

Usaremos as classes:

- **SqlConnection**
- **SqlCommand**
- **SqlDataReader**
- **SqlDataAdapter**

Se você estiver trabalhando com outra fonte de Dados como Oracle ou Sybase por exemplo devemos utilizar o namespace: System.Data.OleDb

Usaremos as classes:

- **OleDbConnection**
- **OleDbCommand**
- **OleDbDataReader**
- **OleDbDataAdapter**

## 8.5 Conhecendo outras classes do ADO.NET

### 8.5.1 DataSet

O DataSet é uma classe "chave" dentro da arquitetura ADO.NET. O DataSet é como nosso banco de dados, porém em memória. Dentro dele podemos armazenar vários DataTable. É possível você adicionar um ou mais DataTable e de diferentes fontes de dados (SQLServer, Oracle, Sybase, XML, etc...). Imagine que você necessita buscar informações de uma tabela em um servidor com o SQL Server e outra a partir de um Oracle, porém esses bancos não conversam entre si, estão em redes que não se comunicam. Você pode buscar as informações dos servidores e adicionar os retornos dentro de um DataSet. Você estará "virtualmente" criando um Banco de Dados local, na sua memória, e ainda podendo estabelecer os

relacionamentos entre as tabelas, caso elas possuam algum tipo de relacionamento com o objeto `DataRelation`.

### 8.5.2 DataTable

O `DataTable` nada mais é que um objeto que possui Dados, para os familiarizados com o ADO, podemos compará-lo com o objeto `Recordset`. Após buscarmos os dados de uma fonte, a partir de um objeto `command`, por exemplo, devemos armazenar esse resultado dentro de um objeto `DataTable`. A diferença entre o `DataTable` e o `Recordset` é que o `DataTable` sempre trabalha com os dados desconectados da fonte, estáticos e do lado do cliente. O conceito do `DataTable` é que ele é para o usuário como uma tabela que está em memória no cliente.

### 8.5.3 DataAdapter

O `DataAdapter` serve como um elo de ligação entre a fonte de Dados e o `DataSet`. Podemos usar o `DataAdapter` para adicionar o resultado de uma declaração SQL dentro de um `DataSet`. O `DataAdapter` executará sua declaração SQL na conexão que for passada para ele.

### 8.5.4 DataView

O `DataView` é um objeto que permite que a partir de um `DataTable` você crie outros tipos de visões partindo daquele resultado local. Uma vez que você tem as informações locais em seu `DataTable` é possível customizar as visões dessas informações.

### 8.5.5 DataRelation

O `DataRelation` é um objeto de relacionamento entre os `DataTables`, quando você possuir mais de um `DataTable` dentro de um `DataSet` você pode querer definir um relacionamento entre esses `DataTables`, para isso você pode utilizar o objeto `DataRelation` e estabelecer um ligação entre essas tabelas.

## 8.6 Componente Binding Source

Encapsula uma fonte de dados para ligação a um controle.

O componente `BindingSource` serve dois propósitos. Primeiro, ele fornece uma camada que vincula os controles de um formulário para os dados. Isto é realizado através da ligação do componente `BindingSource` para a sua fonte de dados e, em seguida, os controles obrigatórios no seu formulário para o componente `BindingSource`. Todas as outras interações com os dados, incluindo a navegar, classificar, filtrar, e atualizar, são realizadas com chamadas para o componente `BindingSource`.

O componente de `BindingSource` atua como a Origem de dados para alguns ou todos controles no formulário. No Visual Studio, a `BindingSource` pode ser Ligado a um Controle por meio da propriedade `DataBindings`, que é acessível da janela de `Properties`.

Em tempo de design, algumas ações, como arrastar uma tabela de banco de dados de uma janela de dados para um formulário em branco, irão criar o Componente de `BindingSource`, vincular-à Origem de dados subjacente, e Adicionar controles ciente a dados em uma operação.



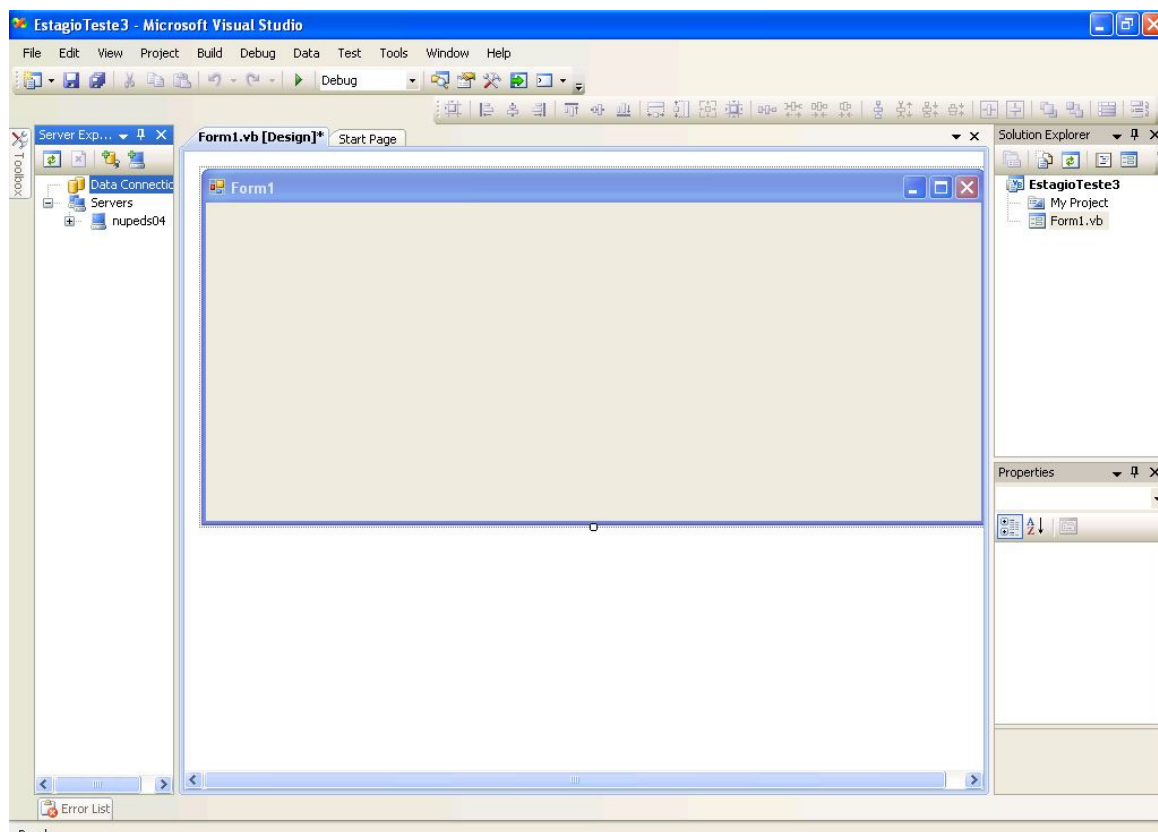


Figura 20 – Formulário no Visual Studio

Em seguida, clique no menu Data e escolha Add New Data Source. Escolha a opção Database e clique no botão Next.

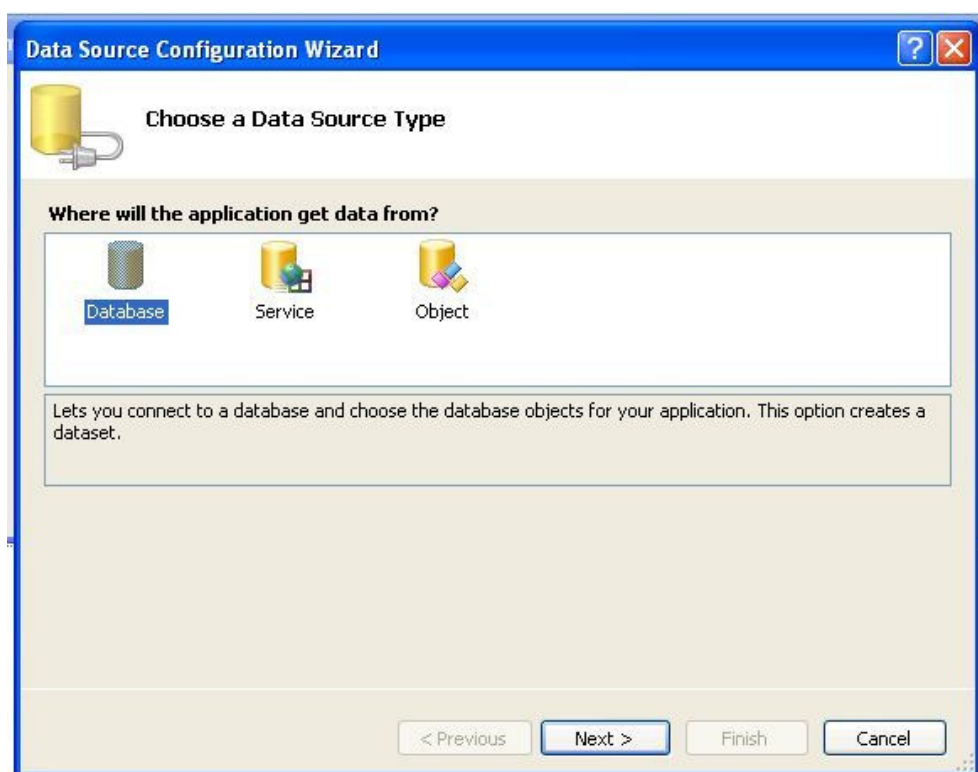


Figura 21 – Assistente de Banco de Dados



Em seguida, clique no botão New Connection para criar sua conexão com o banco de dados.

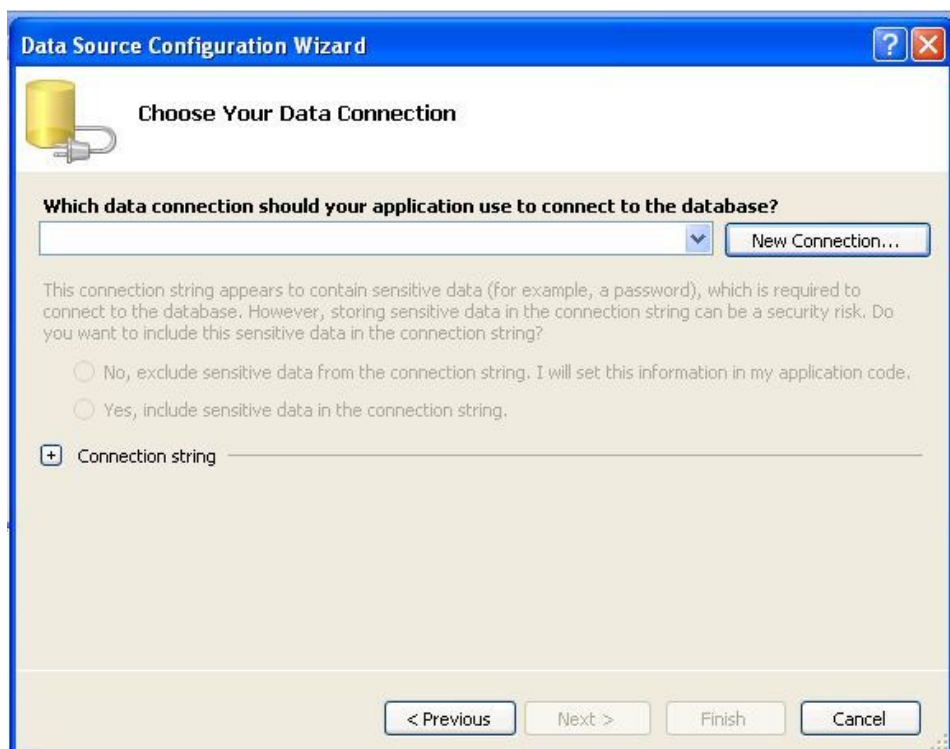


Figura 22 – 2ª Etapa do Assistente de Banco de Dados

Na janela seguinte, clique no botão Browse e escolha o arquivo que deseja conectar.



Figura 23 – 3ª Etapa do Assistente de Banco de Dados

Mantenha o nome de usuário como Admin e password, deixe em branco. Marque a opção "Save my password".

Clique no botão "Test Connection" para testar sua conexão com o banco de dados.

Confirme em OK.

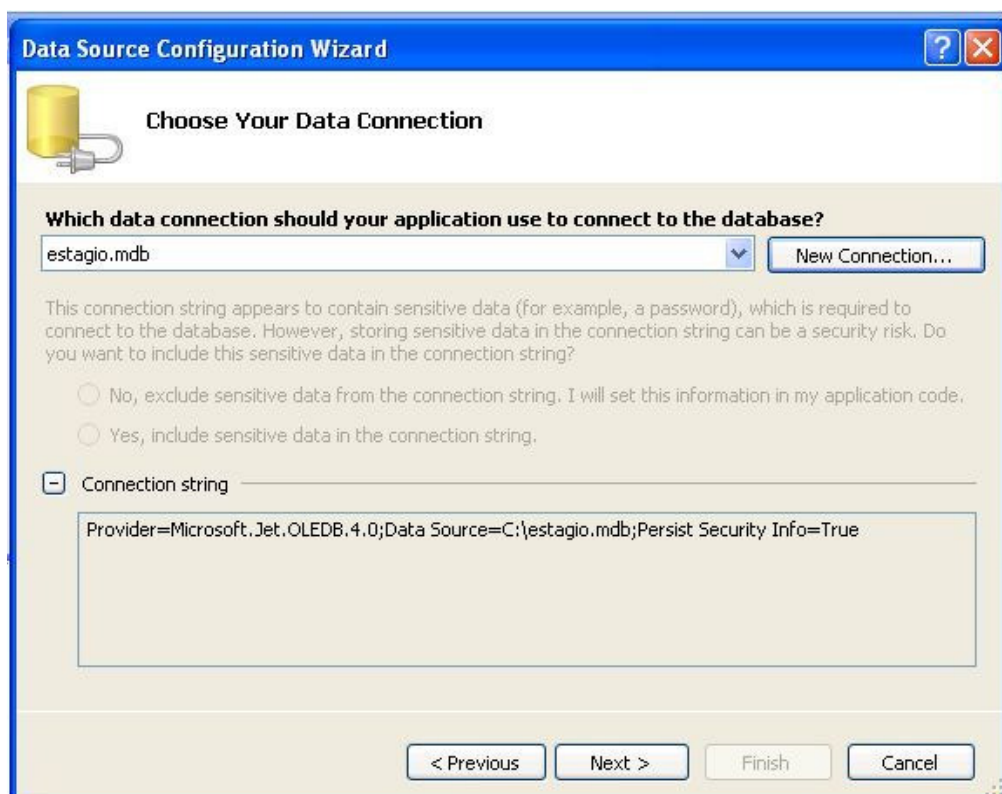


Figura 24 – 4ª Etapa do Assistente de Banco de Dados

Em Connection String verifique a string de conexão que foi criada.  
Clique em Next.

Salve o nome da String de conexão.

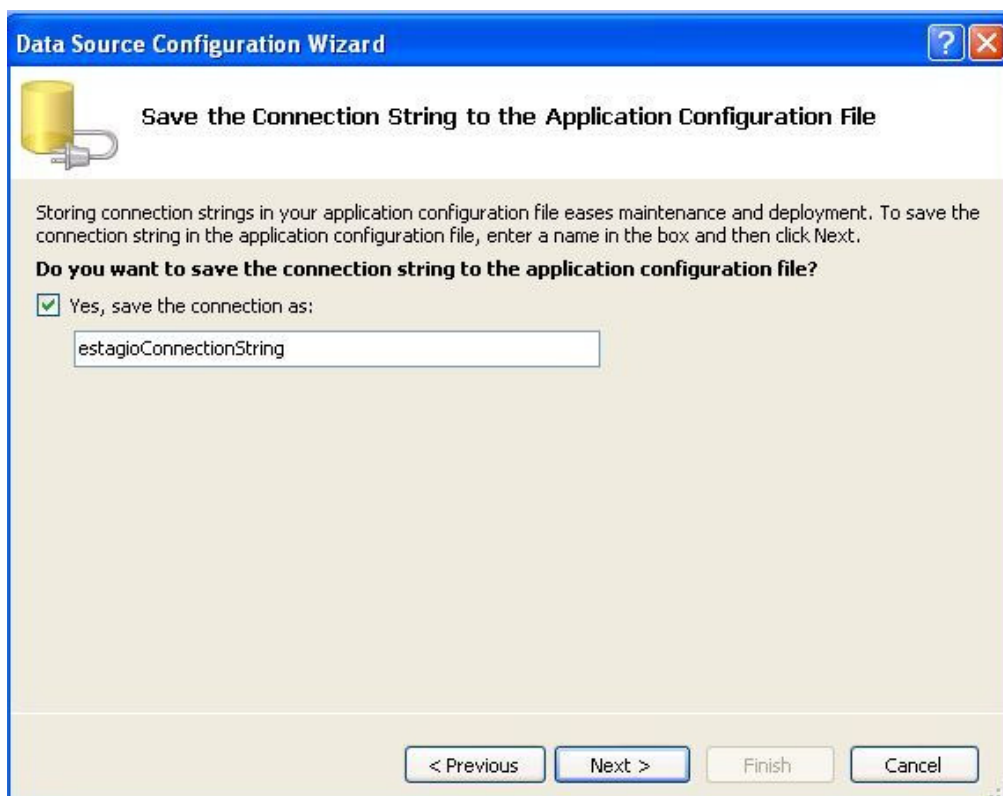


Figura 25 – 5ª Etapa do Assistente de Banco de Dados

Na etapa seguinte, escolha as tabelas que farão parte da sua conexão e o nome do DataSet

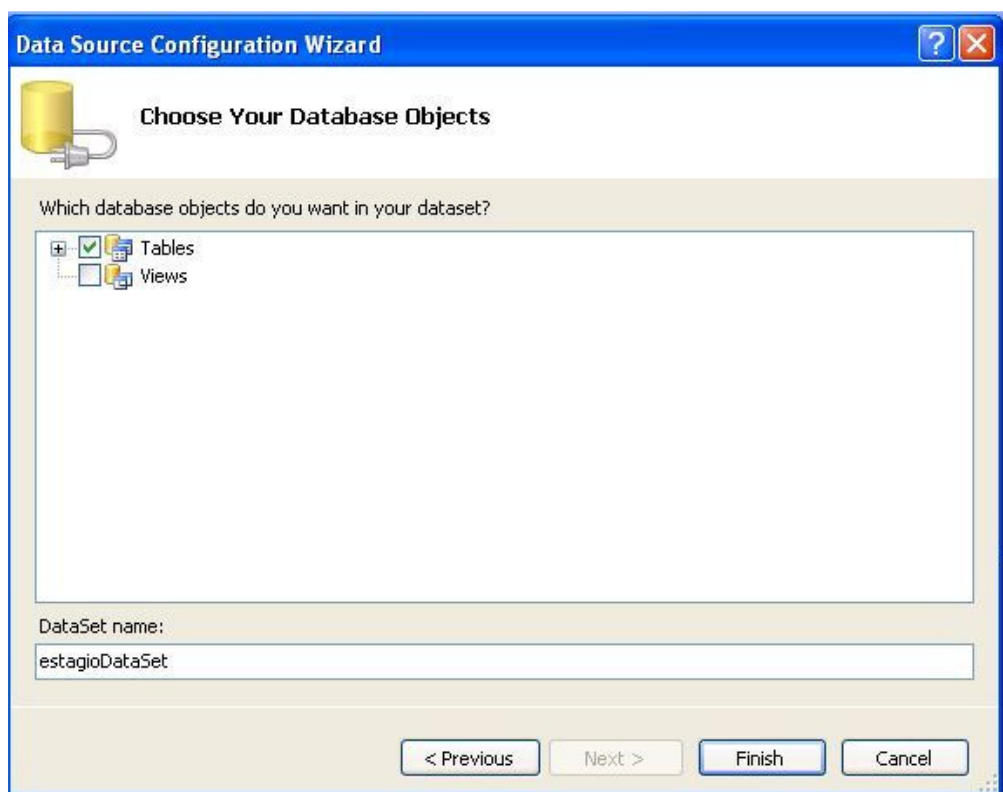


Figura 26 – 6ª Etapa do Assistente de Banco de Dados

Clique então no botão Finish

Após a conclusão deste processo, o Visual Studio irá conectar o banco de dados ao projeto.

Analisaremos o que ocorreu com o projeto:

A janela Solution Explorer agora apresenta os componentes da nova conexão.

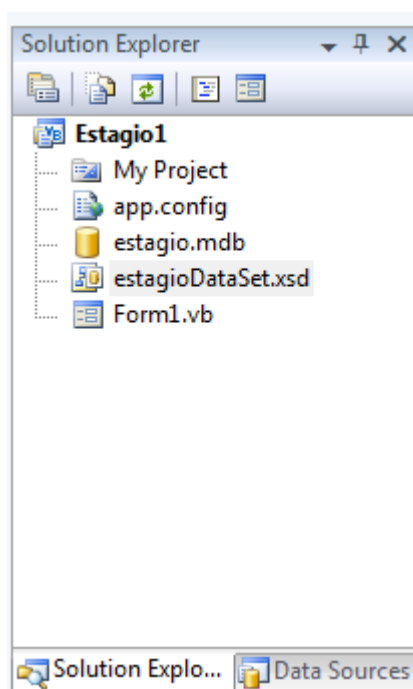


Figura 27 – Janela Solution Explorer

Nesta janela, existe o componente de conexão estagio.mdb que representa o arquivo do Microsoft Access que está conectado ao projeto.

Logo abaixo vê-se o componente estagioDataSet.vxd que representa a conexão do banco de dados com as tabelas. Lembre-se que o DataSet é uma conexão local, em memória.

Um duplo clique neste DataSet apresenta a janela a seguir, mostrando as tabelas do banco de dados que estão conectadas.



Figura 28 – DataSet Designer

O componente TableAdapter que aparece nesta janela apresenta o método Fill, e o método Getdata() que apresentam todos os registros desta tabela através de uma expressão em SQL. Para ver esta expressão, clique o botão direito neste método e escolha a opção *Configure*.

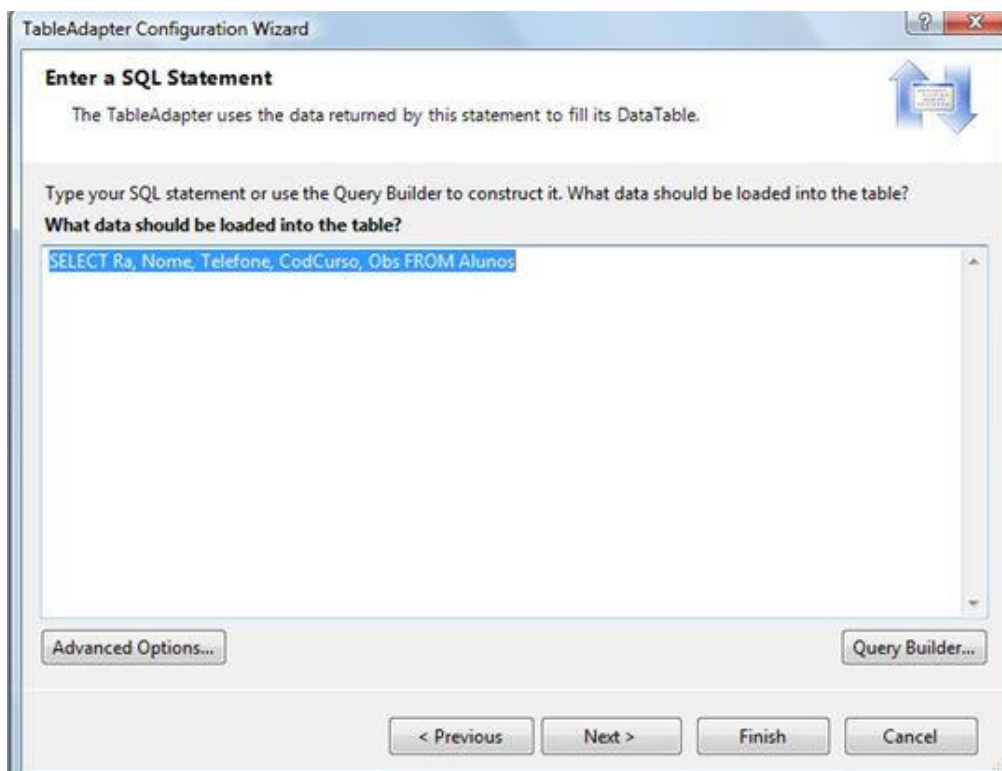


Figura 29 – String SQL

### Adicionando a tabela alunos ao formulário

Clique no menu Data e escolha Show DataSources para visualizar a janela com as tabelas do banco de dados.

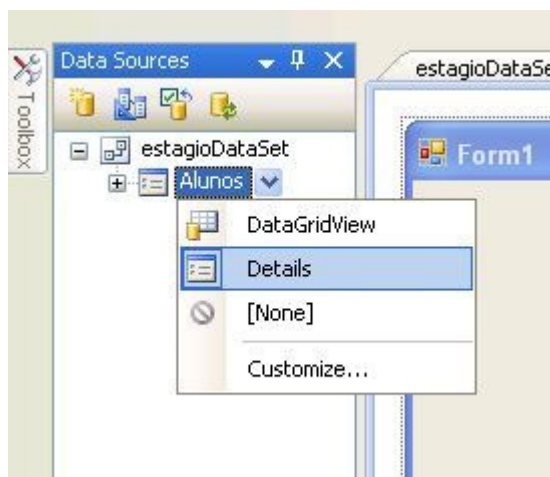


Figura 30 – Data Source

Clique na tabela Alunos e escolha a opção Details para que o formulário apresente um registro por página. Em seguida, arraste a tabela Alunos para dentro do formulário.

Serão apresentados os campos da tabela e também a barra de navegação (navigator). Observe que logo abaixo do formulário surge a barra de componentes.

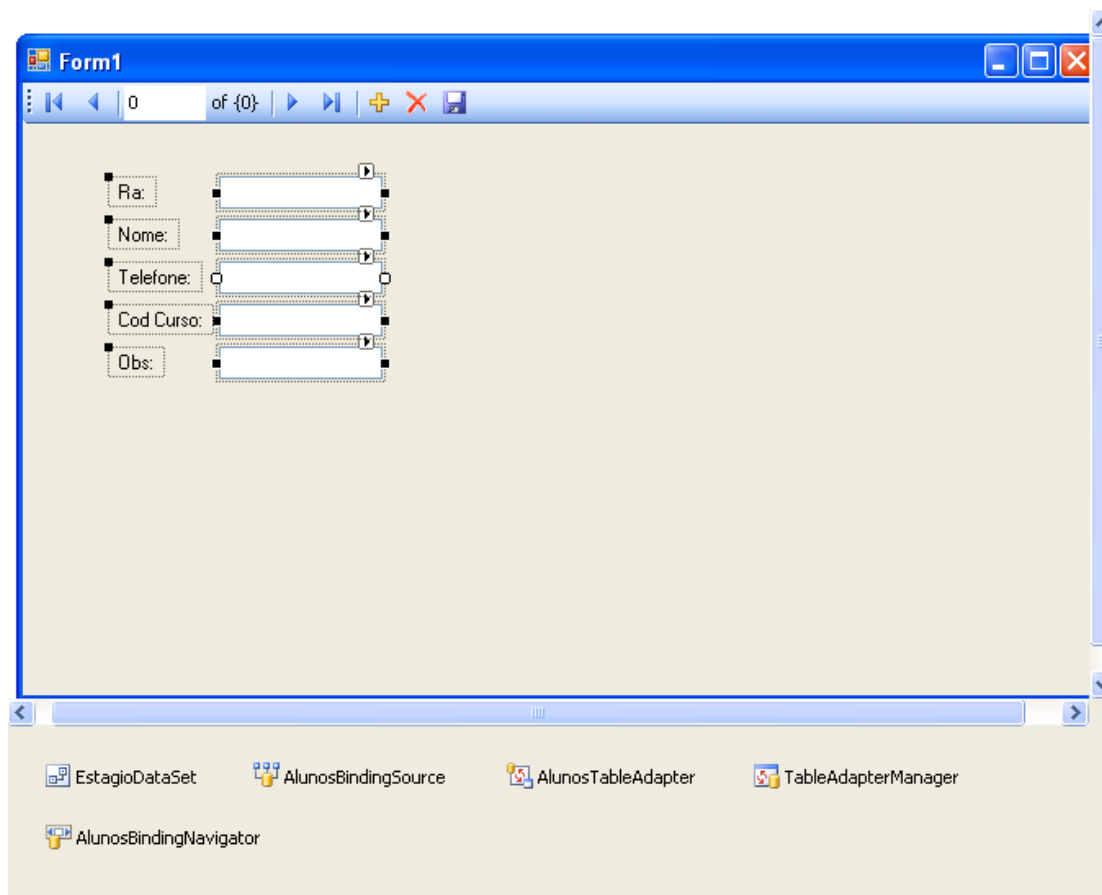


Figura 31 – Formulário com os campos

Salve o projeto e rode o programa para ver o resultado.

Veja que foram criados os seguintes componentes:

- EstagioDataset
- AlunosBindingSource
- AlunosTableAdapter
- TableAdapterManager
- AlunosBindingNavigator

## 10. Acessando dados usando o MS-Access

### 10.1 Como trabalhar com o Microsoft Access usando o OleDb Data Provider

O exemplo a seguir mostra como acessar um banco de dados através do OleDb. Para isto usaremos um banco de dados criado através do MS-Access.

Crie um banco de dados chamado **EMP** no Access e grave-o no drive C:\ do computador. Crie dentro deste banco a tabela **Alunos** com os campos AINo, AINome e Cidade. Insira alguns registros neste banco de dados.

Crie um formulário com três controles TextBox (TextBox1, TextBox2, TextBox3). O Objetivo é recuperar os dados da **Alunos** e apresentá-las no TextBox quando o usuário clicar no botão **Button1**.

**Veja o código a seguir.** Este código deve ser colocado nas declarações gerais do projeto

```
Imports System.Data
Imports System.Windows.Forms.Form
Imports System.Data.OleDb
```

O código abaixo deve ser criado dentro da Classe Form1

```
Public Class Form1

    Dim cn As OleDbConnection
    Dim cmd As OleDbCommand
    Dim dr As OleDbDataReader

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Try
            cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data
Source = c:\aula.mdb;")
            cn.Open()
            cmd = New OleDbCommand("Select * from alunos", cn)
            dr = cmd.ExecuteReader
            While dr.Read()
                TextBox1.Text = dr(0)
                TextBox2.Text = dr(1)
                TextBox3.Text = dr(2)

            End While
        Catch
        End Try
        dr.Close()
        cn.Close()
    End Sub
End Class
```

O resultado será o formulário criado abaixo:

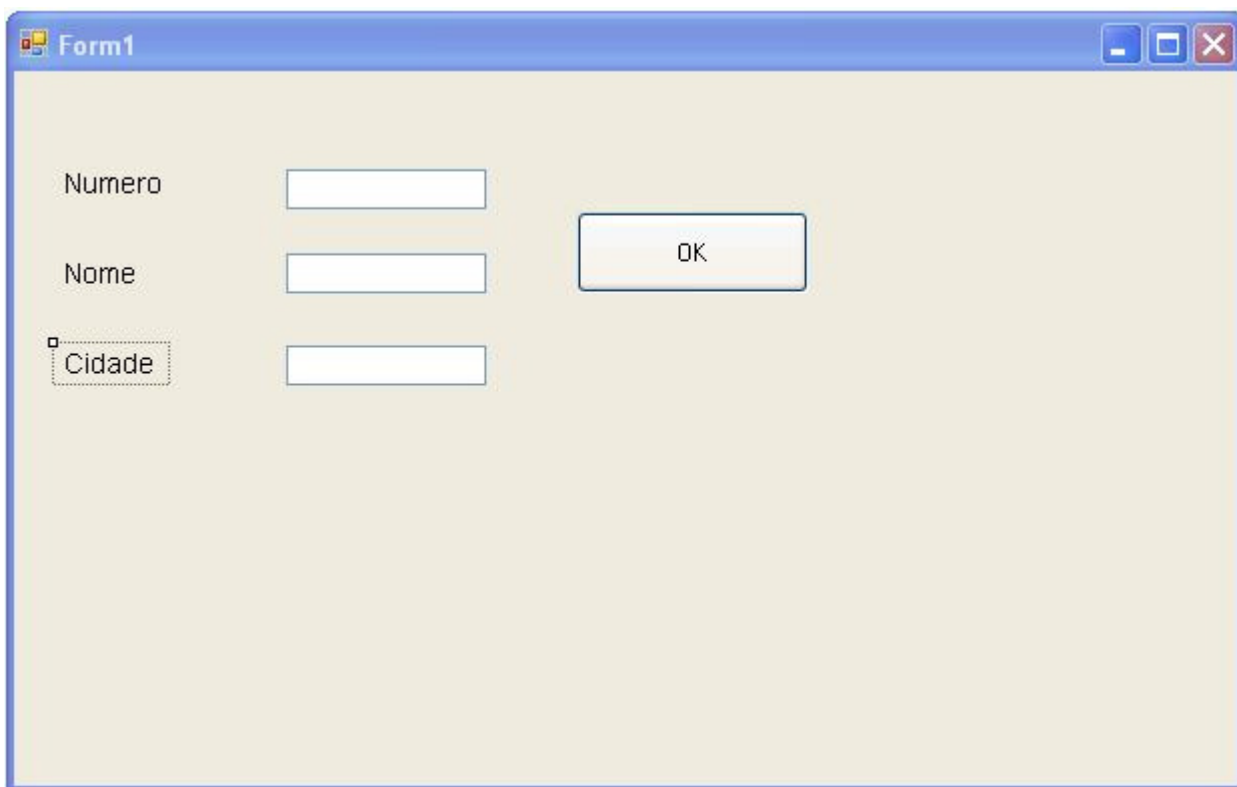


Figura 32 – Ao clicar no botão OK o código é executado e são recuperados os dados da alunos do banco de dados Emp.

Observação: consulte o projeto AcessoDados1 da pasta de exemplos.

## 10.2 Criar uma consulta ao banco de dados usando parâmetros (por código)

Nesta etapa vamos criar um formulário para consulta no banco de dados usando parâmetros.

Será usado para isso o mesmo banco de dados do exemplo anterior – Aula.mdb.

Crie através do assistente um formulário que apresente a Alunos do banco de dados Aula.mdb. Esta tabela deve ser apresentada no modo DataGrid.

Em seguida crie uma caixa de texto e um botão para efetuar a busca.

O formulário deve ter o seguinte aspecto:



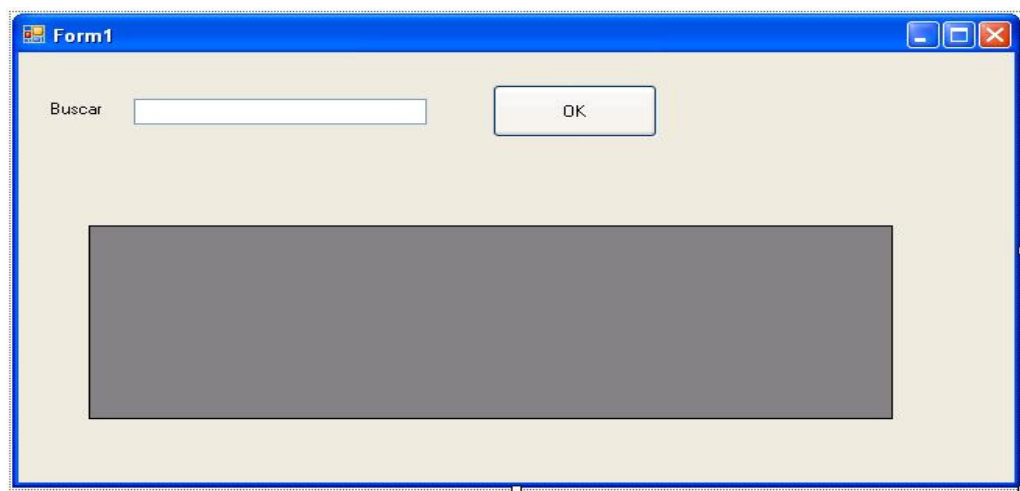


Figura 33 - Formulário para acesso a dados

Em seguida inclua no projeto o seguinte código:

```
Imports System.Windows.Forms.Form
Imports System.Data.OleDb

Public Class Form1
    Dim cn As OleDbConnection
    Dim cmd As OleDbCommand
    Dim nome As OleDbParameter
    Dim tb As DataTable
    Dim dr As OleDbDataReader

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Try
            cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data
Source = c:\aula.mdb;")
            cn.Open()
        Catch
        End Try

        cn.Close()

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        cmd = New OleDbCommand("Select * from alunos where AlNome=@nome")
        cmd.Connection = cn
        cmd.Parameters.Add("@nome", OleDbType.VarWChar, 15).Value =
TextBox1.Text
        Dim da As New OleDbDataAdapter(cmd)
        Dim ds As New DataSet()

        tb = New DataTable
        da = New OleDb.OleDbDataAdapter
        tb.Clear()
        da.SelectCommand = cmd
        da.Fill(tb)

        DataGridView1.DataSource = tb
    End Sub
End Class
```

```
End Sub
End Class
```

Observação: consulte o projeto AcessoDados2 da pasta de exemplos.

### 10.3 Criar uma consulta com escolha de parâmetros (Através do Assistente)

Neste exercício vamos criar um formulário para realizar a consulta de alunos usando os critérios nome, número ou cidade. O usuário irá escolher o critério desejado.

Usando o mesmo banco de dados do exercício anterior, crie um DataGrid para a tabela de Alunos. Veja a figura abaixo:

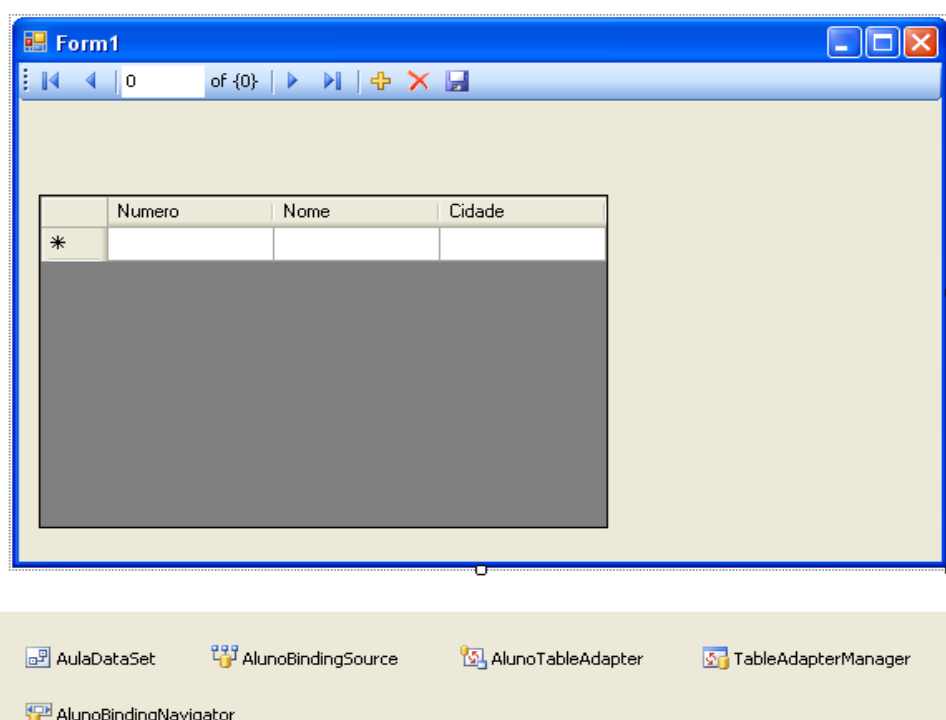


Figura 34 – Data Grid para acesso a dados

Em seguida, crie uma *comboBox* para listar os critérios de busca. Na propriedade Items da combobox, insira as palavras: Número, Nome e Cidade.

Veja figura abaixo:

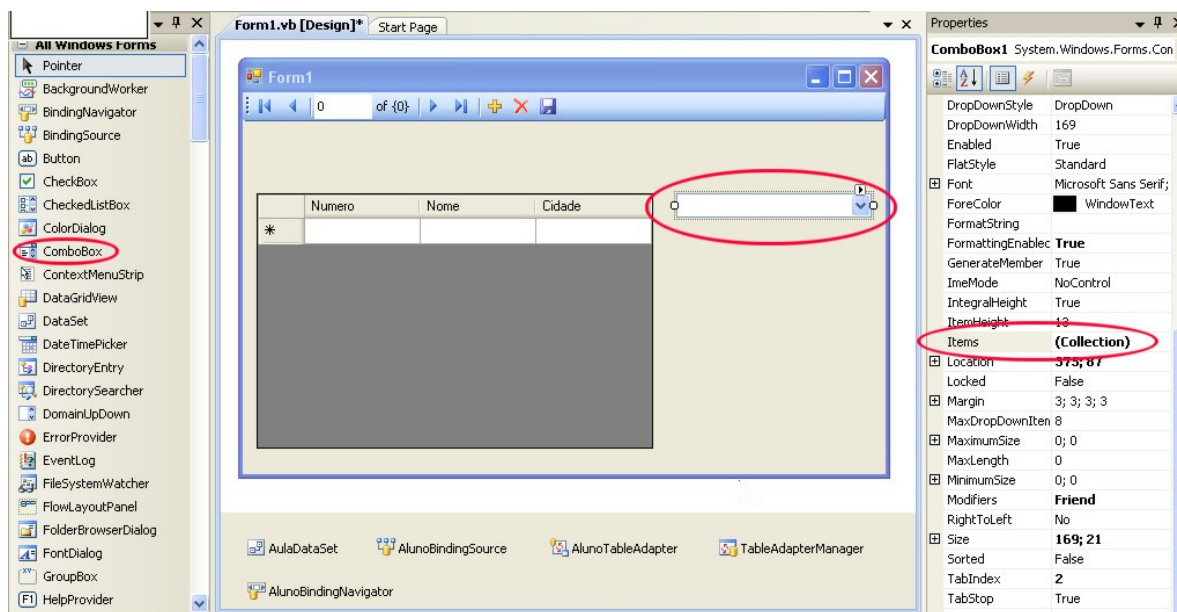


Figura 35 – Criação de Combobox para consulta.

Crie uma caixa de texto e um botão com o título: Buscar.

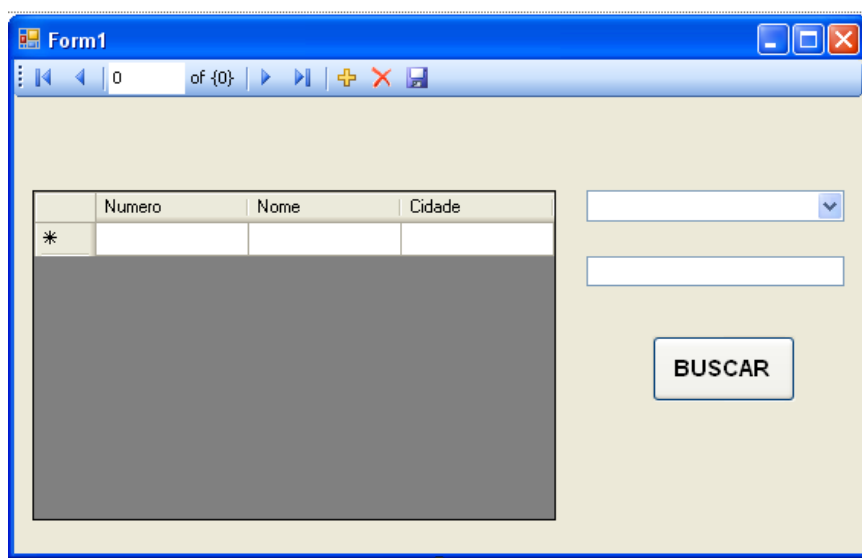


Figura 36 – Criação de Combobox para consulta.

Agora serão definidos os critérios para realizar a consulta. Vá no Solution Explorer dê dois cliques em *AulaDataSet.xsd*.

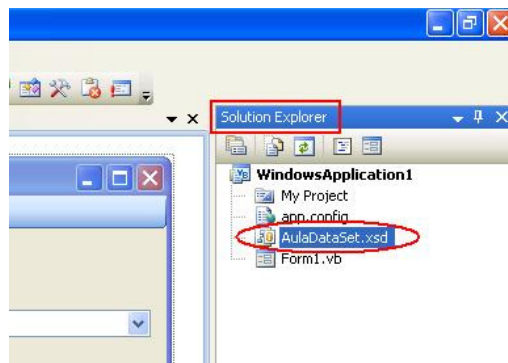


Figura 37 – Acesso ao DataSet

Será aberta a seguinte janela.

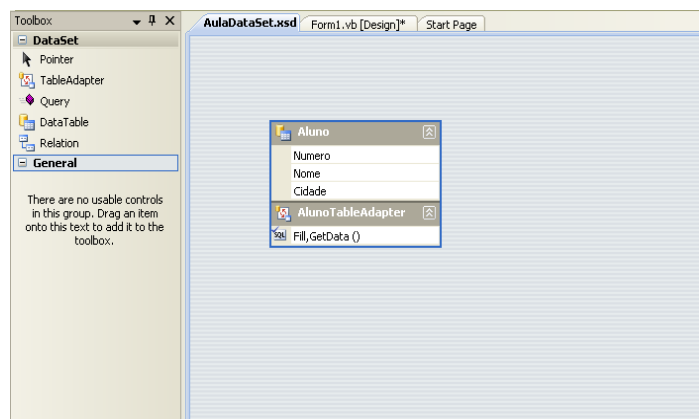


Figura 38 – DataSet.

Dê um clique com o botão direito sobre *Fill, GetData()*. Clique em *Add Query*

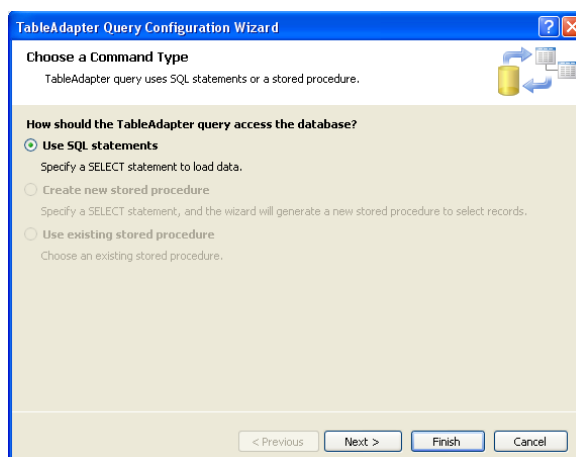


Figura 39 – Início do assistente para consulta.

Clique em Next.

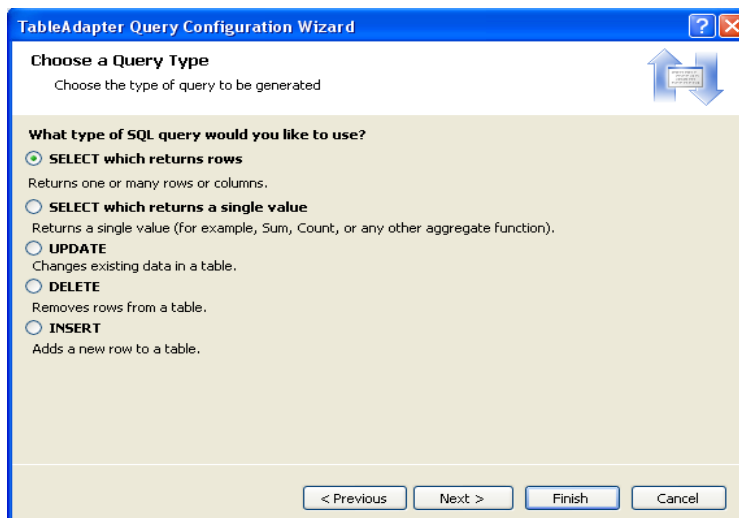


Figura 40 – 2ª etapa

do assistente de consulta.

Selecione a primeira opção *Select which returns rows*, clique em Next. Abrirá a seguinte janela.

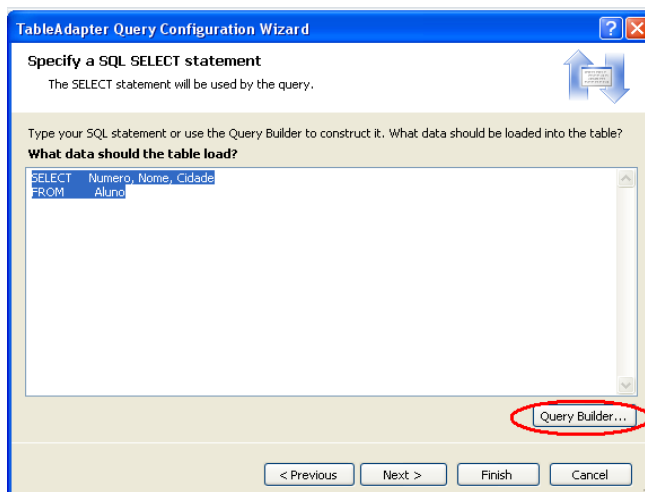


Figura 41 – 3ª etapa do assistente para consulta.

A próxima janela é onde são definidos os critérios de busca.

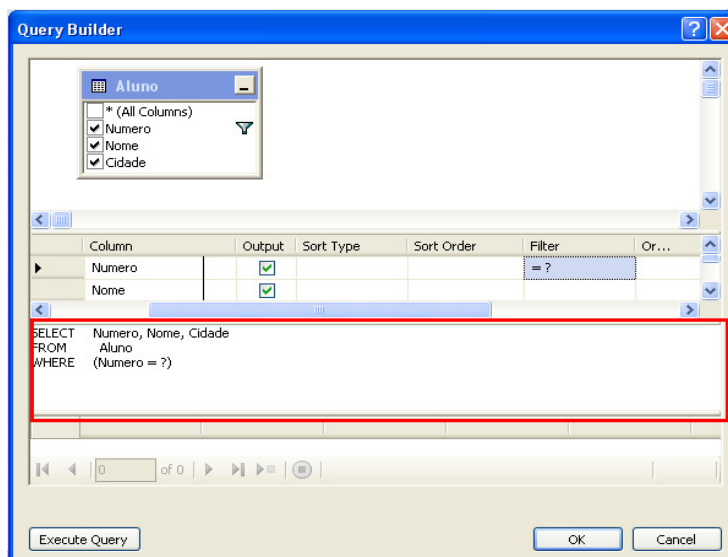


Figura 42 – 4ª etapa do assistente para consulta.

Acrescente o código com a imagem acima.

*WHERE (Numero = ?)*

Dê OK, Clique em Next para adicionar um nome a consulta. Coloque o nome como ConsultaNumero, clique em Next e em Seguida Finish.

Voltará para a janela *AulaDataSet.xsd*, mas com uma nova consulta.

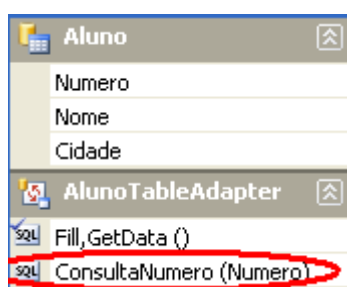


Figura 43 – consulta criada ao final do assistente

Repita o procedimento, mas acrescentando o seguinte código.

*WHERE (Nome LIKE ?)*

Coloque o nome da consulta como ConsultaNome.

Repita o procedimento novamente. Acrescente o seguinte código

*WHERE (Cidade LIKE ?)*

Coloque o nome da consulta como ConsultaCid

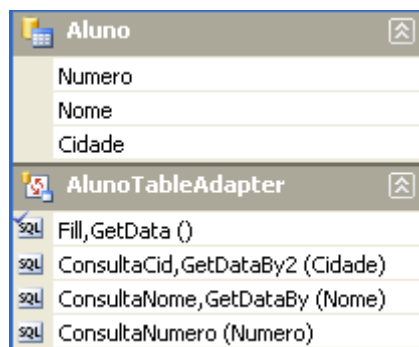


Figura 44 – Todas as consultas criadas

Volte para o Form1, onde será adicionado o código para as consultas terem funcionalidade.

Dê dois cliques no botão Buscar. Adicione o seguinte código:

```

If ComboBox1.SelectedItem = "Número" Then
    Try
        Me.AlunoTableAdapter.ConsultaNumero(Me.AulaDataSet.Aluno,
        TextBox1.Text)

        Catch ex As Exception
        End Try

ElseIf ComboBox1.SelectedItem = "Nome" Then
    Try
        Me.AlunoTableAdapter.ConsultaNome(Me.AulaDataSet.Aluno, TextBox1.Text)

        Catch ex As Exception
        End Try

Else
    Try
        Me.AlunoTableAdapter.ConsultaCid(Me.AulaDataSet.Aluno, TextBox1.Text)

        Catch ex As Exception
        End Try

End If

```

## 10.4 Fazer consultas por parte do Nome

Abra o mesmo projeto do exemplo acima. Clique no botão buscar para abrir as instruções desse botão.

No procedimento de evento que foi aberto *Button1\_click*, localize onde está escrito *TextBox1.Text*.



Figura 45 – Consulta por parte do nome

Altere para "%" + Me.TextBox1.Text + "%"

Agora será possível consultar pela inicial do nome quanto pelo final do nome.

## 10.5 Consulta com data

Será necessário criar um pequeno banco de dados com campos que contenham datas. Exemplo: Abra o Access, crie uma tabela chamada estágio. Criar os campos abaixo:

Nome = do tipo texto

dataInicio = do tipo Data/Hora

dataFim = do tipo Data/Hora

Preencher alguns dados na tabela.

*O exemplo abaixo possibilitará encontrar estágios que estão para terminar.*

No Visual Studio criar um novo projeto chamado Estagio. Fazer a conexão com o banco de dados estagio. Arraste a tabela estagio para o form1 como *DataGridView*. Acrescente algumas ferramentas para que o formulário fique assim:

Figura 46 – Exemplo de consulta com data

Fazer a consulta de datas adicionando o seguinte código:

*WHERE (DataFim BETWEEN ? AND ?)*



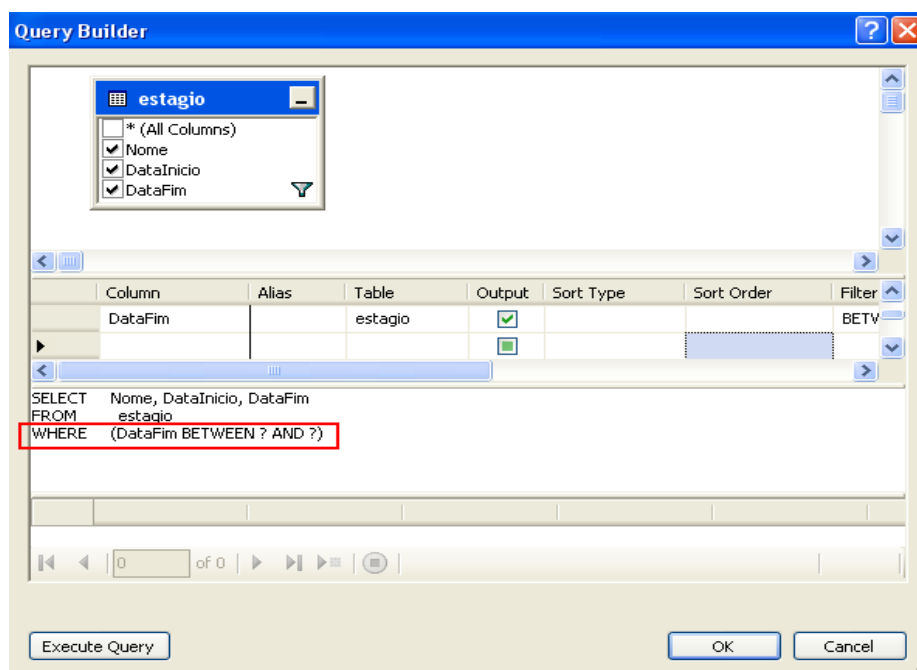


Figura 47 – Consulta com datas

Dê o nome da consulta de *FillTermino*. Volte para o Form1.

Dê dois cliques no botão consultar para adicionar as instruções que fazem a consulta ter funcionalidade.

Digite o seguinte código:

```
Me.EstagioTableAdapter.FillTermino(Me.EstagioDataSet.estagio, TextBox1.Text,
    TextBox2.Text)
```

O programa irá buscar as datas de fim do estágio, entre TextBox1 e TextBox2.

## 11. Diferença entre DataSet e DataTable

### DataSet

O mais importante é a classe DataSet pois contém toda a funcionalidade para gerenciar dados armazenados na memória através de um cache de dados desconectado .

Podemos ver que o DataSet contém , tabelas , relacionamentos entre tabelas e cada tabela contém um conjunto de linhas e colunas.

Perceba que o DataSet esta relacionado com Tabelas ( Tables ) e Relacionamentos ( Relations ) e que as tabelas contidas no DataSet pertence a classe DataTable.

### DataTable

É um objeto chave dentro do namespace System.Data. O objeto DataTable expõe as seguintes propriedades :

1. Columns – A coleção Columns contém uma lista de todas as colunas contidas em uma tabela;
2. Constraints – Regras de dados que são aplicadas a um tabela;

3. ChildRelations – É uma coleção de relacionamentos que define as relações entre as tabelas filhas e o DataTable.;
4. PrimaryKey – É uma matriz de objetos DataColumn que representa todas as colunas com as chaves primárias de um DataTable particular;
5. Rows – Fornecem os dados atuais contidos em uma tabela. Contém uma coleção de objetos DataRow.
6. TableName – Representa o nome do DataTable;

## 12. Objeto DataRow

O objeto DataRow é um componente essencial do objeto DataTable . É o container para as linhas de dados dentro de uma tabela e permite as funcionalidades: Criar, Atualizar, Obter e Excluir. A seguir temos os métodos e propriedades mais importantes da classe DataRow :

1. RowState – Indica o estado atual do DataRow podendo ser um dos seguintes valores:
  - Added – A linha foi incluída na tabela e o método AcceptChanges() ainda não foi chamado.
  - Deleted – A linha foi excluída da tabela através do método Delete().
  - Detached – A linha atualmente não faz parte da tabela.
  - Modified – Os dados dentro das linhas foram modificados e AcceptChanges() ainda não foi chamado.
  - Unchanged – Os dados da linha não foram alterados desde a última chamada a AcceptChanges().
2. BeginEdit() – Função que ativa o modo de edição do DataRow permitindo ao código modificar os dados de mais de uma linha de uma vez.
3. CancelEdit() – Desativa o modo de edição do DataRow e descarta as alterações feitas desde que a chamada a BeginEdit() foi feita.
4. Delete() – Exclui a linha atual.
5. EndEdit() – Completa o modo de edição para a linha atual , salvando as alterações no DataSet desde que o método BeginEdit() foi invocado.
6. AcceptChanges() – Invoca implicitamente o método EndEdit() e dependendo do status de Rowstate as alterações são descartadas ou efetivadas.

### 12.1 Preencher objeto DataSet ( via código)

Pode-se usar um DataSet para armazenar dados de um banco de dados e pode-se também mover os dados de um DataSet para um banco de dados , mas o próprio DataSet não faz conexão alguma com o banco de dados , ele nem mesmo tem um objeto para realizar tal conexão . A ligação entre o banco de dados e DataSet é feita pelo objeto DataAdapter (Adaptador de dados).

Inicie o Visual Studio 2008. Insira no formulário os seguintes componentes: DataGrid, TextBox, Botão.

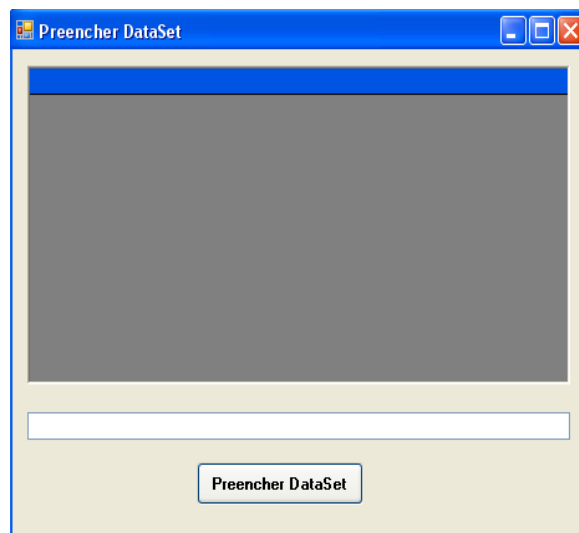


Figura 48 – Formulário com os objetos criados

Nas declarações gerais insira:

```
Imports System.Data
Imports System.Windows.Forms.Form
Imports System.Data.OleDb ' Permite usar os objetos que irão criar uma conexão
com
                           o banco de dados.
```

Volte para o formulário. Dê dois cliques no botão Preencher DataSet para abrir o procedimento de evento button1\_click. Insira o seguinte código:

```
'define a string de conexão com o banco de dados
Dim strConn As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source ="
& TextBox1.Text

'define o objeto OleDbConnection usando a string de conexão
Dim conexao As New OleDbConnection(strConn)

'define a instrução SQL que será usada para extrair as linhas da tabela
Estagio
Dim sql As String = "Select * FROM estagio"

'cria o objeto OleDbCommand
Dim comando As New OleDbCommand(sql, conexao)

'Cria o objeto DataAdapter
Dim adaptador As New OleDbDataAdapter(comando)

'Cria o objeto DataSet
Dim dsEstagio As New DataSet()

'preenche o dataset
adaptador.Fill(dsEstagio, "estagio")

'exibe os dados em um datagrid
DataGrid1.DataSource = dsEstagio
```

Veja abaixo o resultado da exibição do DataSet no DataGrid.

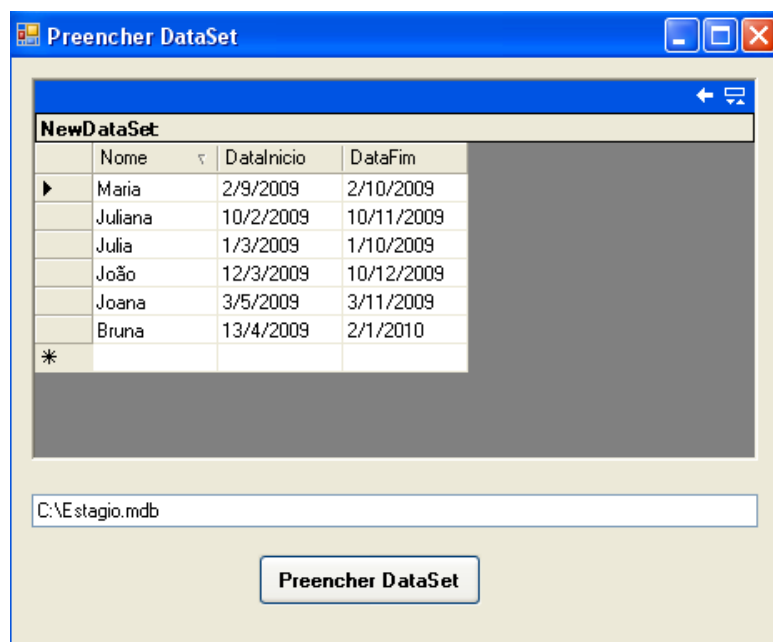


Figura 49 – Exibição dos dados no DataGrid

### 13. Diferença entre DataGrid e DataGridView

O controle DataGridView Windows Forms é um completamente novo controle .NET. O recurso de apresentação de dados da tabela ficou muito mais fácil de ser implementado em .NET Windows. As características mais atraentes do novo controle são: permite misturar os dados associados e não associados, as colunas virtuais no GridView. Ele dá a capacidade de facilmente aplicar estilos para diferentes colunas, linhas, células, ou uma tabela toda fonte de dados.

Já o DataGrid, Para a maioria das situações, funciona mais rápido do que o DataGridView. Também o controle dá-lhe capacidade de exibir dados hierárquicos. Esta é uma grande vantagem do DataGrid. NET Windows Forms. A quantidade de empregos que você pode querer alcançar com o controle DataGrid Windows Forms são, inesperadamente, mais problemático do que se poderia esperar. Principalmente, é porque o Windows Forms controle DataGrid é baseado em colunas, ao invés de célula-base. Como resultado, para atingir a maioria das tarefas, você tem que trabalhar com as colunas, e não as próprias células.

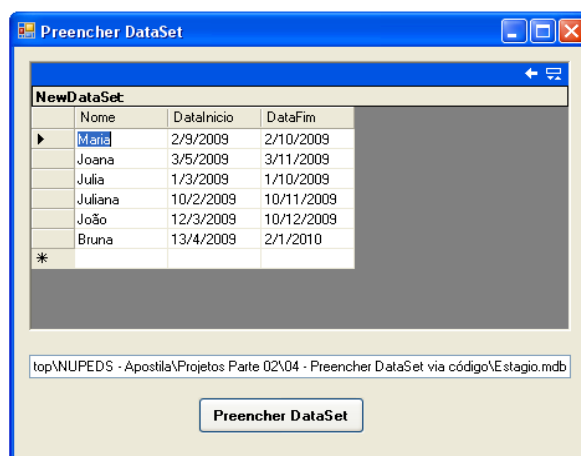
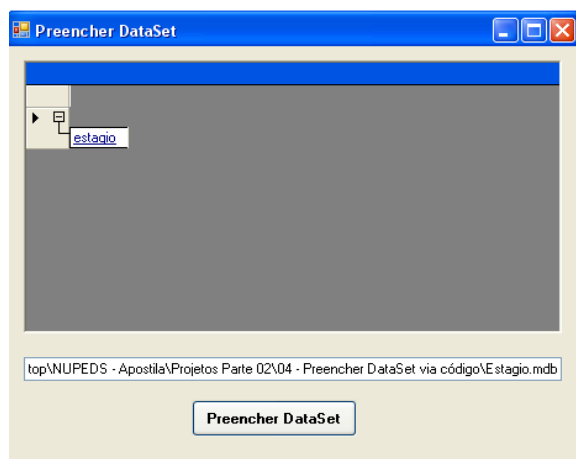


Figura 50 – Exemplo de DataGrid

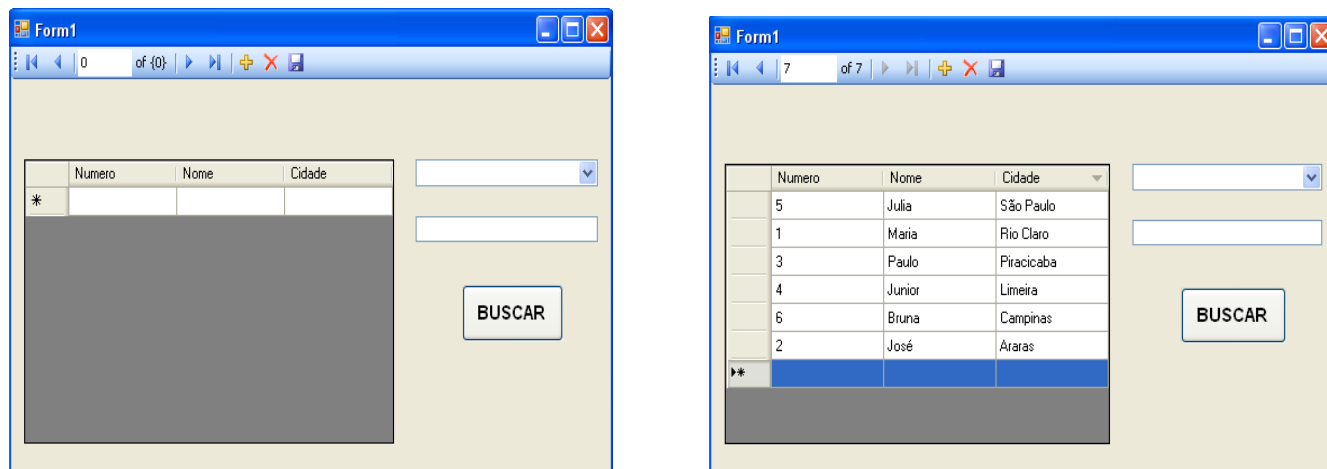


Figura 51 – Exemplo do DataGrid View

## 14. Data Binding

Será feito um exemplo utilizando alguns recursos de *Data Binding*.

Inicie um novo projeto chamado de *DataBinding* e adicione os controles de acordo com a imagem e descrição abaixo:

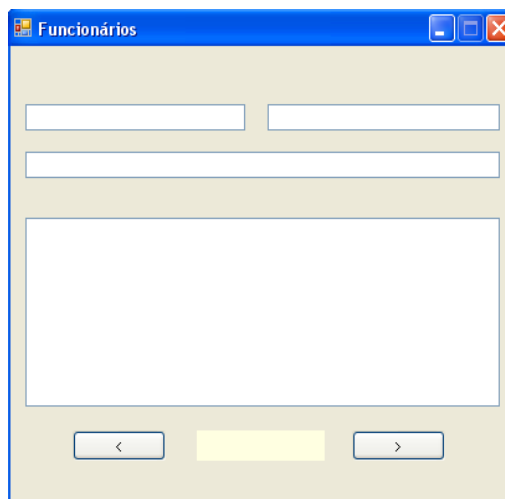


Figura 52 – Formulário para uso do DataBinding

OBJETO	PROPRIEDADE	VALOR
TextBox1	Name	txtPrimeiroNome
TextBox2	Name	txtultimoNome
TextBox3	Name	txtCargo
TextBox4	Name	txtNotas
Button1	Name Text	btnVoltar <
Button2	Name Text	btnAvancar >

Label	Name Backcolor Text	lblPosicao Info ""
Form1	Name	FrmFuncionarios

Vá a Server Explorer e crie uma nova conexão com o banco de dados funcionarios. Arreste a tabela para o form1.

### Agora vamos ao código:

No evento Load será verificado a posição e quantidade de dados:

```
Me.txtPrimeiroNome.DataBindings.Add("Text", Me.FuncionariosDataSet.funcionarios,
"PrimeiroNome")
Me.txtUltimoNome.DataBindings.Add("Text", Me.FuncionariosDataSet.funcionarios,
"Ultimonome")
Me.txtCargo.DataBindings.Add("Text", Me.FuncionariosDataSet.funcionarios,
"Cargo")
Me.txtNotas.DataBindings.Add("Text", Me.FuncionariosDataSet.funcionarios,
"Notas")

'Obtem a quantidade de registros
RecordCount =
Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Count

'Obtem a posição atual
Position =
Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Position

AtualizaLabel()

'Verifica se existem registros
If RecordCount <= 1 Then
    Me.btnAvancar.Enabled = False
    Me.btnVoltar.Enabled = False

Exit Sub
End If
```

No evento Click do botão voltar adicionar o seguinte código:

```
'Analisa se podemos voltar

If sender.Equals(Me.btnVoltar) Then
    If RecordCount > 1 Then
        Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Position -=
1
    Else
        Me.btnVoltar.Enabled = False
    End If
End If

'Analisa de podemos avançar

If sender.Equals(Me.btnAvancar) Then
    If RecordCount > Position Then
```

```

1         Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Position +=
Else
    Me.btnAvancar.Enabled = False
End If
End If

'Obtem a nova posição
Position = Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Position +
1

'Recalcula os status dos botões de navegação

If RecordCount = Position Then
    btnAvancar.Enabled = False
Else
    btnAvancar.Enabled = True
End If

If Position = 1 Then
    btnVoltar.Enabled = False
Else
    btnVoltar.Enabled = True
End If

'Atualiza o label que mostra a posição atual
AtualizaLabel()

```

Adicionar o mesmo código no evento Click do botão avançar.

E finalmente a sub que atualiza o Label com a informação da posição no registro:

```

Private Sub AtualizaLabel()

RecordCount = Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Count
Position = Me.BindingContext(Me.FuncionariosDataSet.funcionarios).Position + 1

If RecordCount <= 1 Then
    lblPosicao.Text = "Sem Registro"
Else
    lblPosicao.Text = "Registro " & Position & " de " & RecordCount
End If

```

## Incluir, excluir e alterar dados na tabela

No mesmo projeto acima (Data Binding), Será adicionado três funções: Incluir, excluir e alterar dados da tabela. Adicione 3 Botões no form1 – incluir/excluir/alterar.

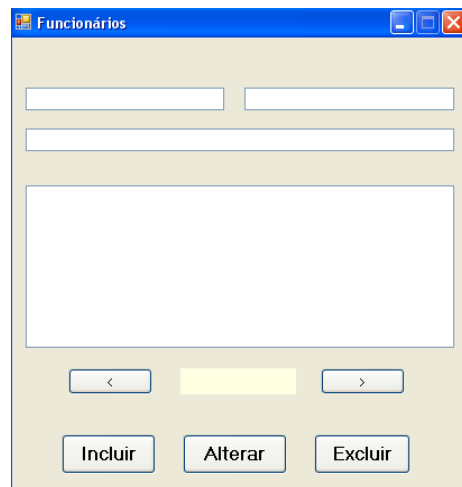


Figura 52 – Botões do Incluir, Alterar e Excluir

## Botão Incluir

Será necessário criar outro Formulário.

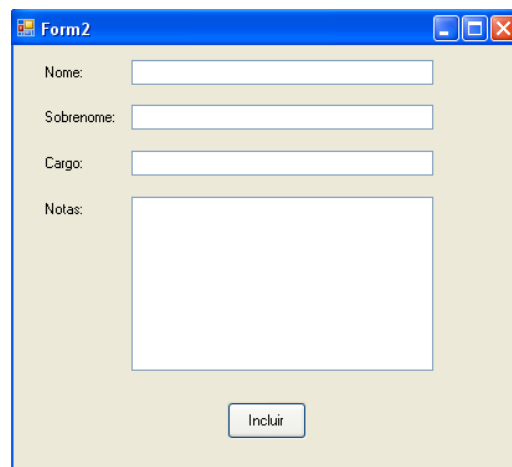


Figura 53 – Botão Incluir

No botão incluir do Form2 acrescente o seguinte código:

```
'define a string de conexão com o banco de dados
Dim strConn As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source = C:\
funcionarios.mdb;"

'define o objeto OleDbConnection usando a string de conexão
Dim conexao As New OleDbConnection(strConn)

'define a instrução SQL que será usada para extrair as linhas da tabela
funcionarios
Dim sql As String = "Select * FROM funcionarios"

'cria o objeto OleDbCommand
Dim comando As New OleDbCommand(sql, conexao)
```



```
'Cria o objeto DataAdapter
Dim adaptador As New OleDbDataAdapter(comando)

'Cria o objeto DataSet
Dim dsFuncionarios As New DataSet()

'preenche o dataset
adaptador.Fill(dsFuncionarios, "funcionarios")

Dim oDR As DataRow
oDR = dsFuncionarios.Tables("funcionarios").NewRow

'preenche os datarow com os valores
oDR("PrimeiroNome") = TextBox1.Text
oDR("UltimoNome") = TextBox2.Text
oDR("Cargo") = TextBox3.Text
oDR("Notas") = TextBox4.Text

'adiciona o datarow ao dataset
dsFuncionarios.Tables("funcionarios").Rows.Add(oDR)

'usa o objeto Command Bulder para gerar o comando Insert dinamicamente
Dim oCB = New OleDbCommandBuilder(adaptador)

'Atualiza a tabela funcionarios
adaptador.Update(dsFuncionarios, "funcionarios")

Close()
```

Volte para o form1. No botão incluir adicione o seguinte código: `Form2.Show()`

## 15. Relatório

Será utilizado o mesmo projeto que foi realizada a consulta com data. Abra o projeto, Adicione mais um botão (botão que irá gerar o relatório).

Crie um novo formulário para receber o relatório. Na Toolbox clique em MicrosoftReportViewer. Deixar o objeto MicrosoftReportViewer do tamanho do formulário como abaixo:

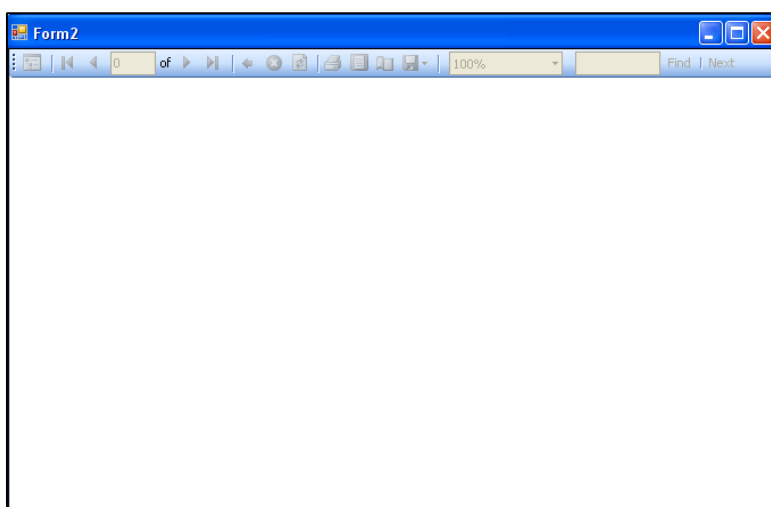


Figura 54 – Tela do Relatório

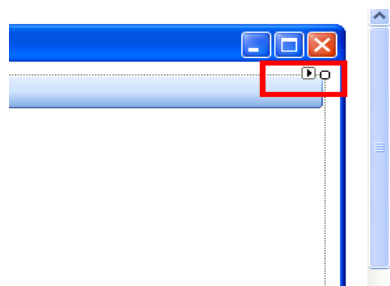


Figura 55 – Clique na seta com mostra a imagem ao lado.

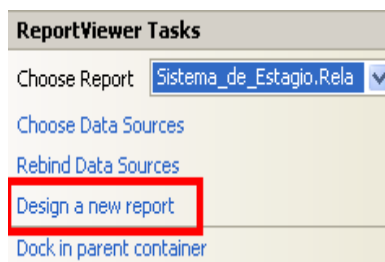


Figura 56 – Selecione a opção *Design a new report*.

Abrirá uma nova janela, onde selecionará a tabela que será gerada no relatório. Selecione a tabela e clique em Next. Selecione o tipo de relatório (Tabular) e clique Next. Na próxima janela mostrará os campos disponíveis, clique em Next. Agora será selecionado o layout do relatório, selecione a primeira opção e clique em Next, o último passo é escolher o estilo, selecione um e clique em Next. Coloque o nome como Relatório e clique em Finish.

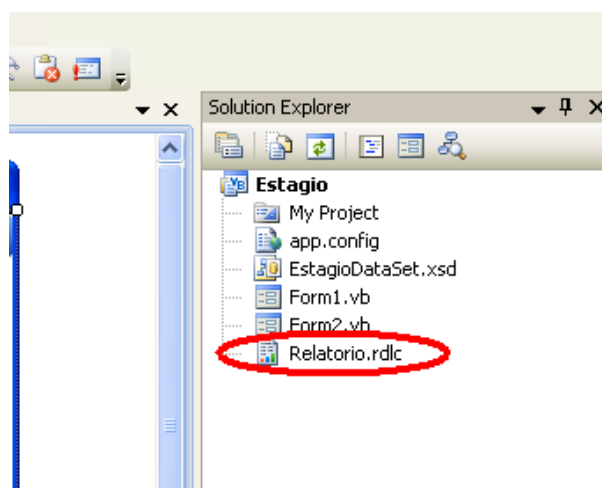


Figura 57 – Para customizar o relatório é só clicar nele em *Solution explorer*.

Clique novamente na setinha. Selecione a opção Choose Report e escolha Estagio.Relatorio (o que acabou de ser criado).

Finalmente volte para o form1. Dê dois cliques no botão Relatório e digite o seguinte código:

Form2.Show()

Pronto. Agora é só rodar o programa.

## 16. Bibliografia

HALVORSON, M. Microsoft Visual Basic 2008 Passo a Passo. São Paulo. Editora Artmed. 2008

ALVORSON, Michael. Microsoft Visual Studio. Wikipédia. Disponível em [http://pt.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](http://pt.wikipedia.org/wiki/Microsoft_Visual_Studio). Acesso em 28/04/2009.

INTRODUÇÃO AO VISUAL STUDIO - MSDN. Disponível em: <http://msdn.microsoft.com/pt-br/library/fx6bk1f4.aspx> . Acesso em 28/04/2009.

O QUE HÁ DE NOVO NO VISUAL BASIC – MSDN. Disponível em: <http://msdn.microsoft.com/pt-br/library/we86c8x2.aspx>. Acesso em 28/04/2009

TOUR RÁPIDO DO AMBIENTE DE DESENVOLVIMENTO INTEGRADO – MSDN. Disponível em <http://msdn.microsoft.com/pt-br/library/ms165088.aspx>. Acesso em 28/04/2009.

VISÃO GERAL SOBRE DESENVOLVIMENTO DE APLICATIVOS COM VISUAL BASIC EXPRESS. Disponível em : <http://msdn.microsoft.com/pt-br/library/ds00eztk.aspx>. Acesso em 29/04/09.

REFERÊNCIA SOBRE LINGUAGEM VISUAL BASIC. Disponível em: <http://msdn.microsoft.com/pt-br/library/zh1f56zs.aspx>. Acesso em 08/05/2009.

Outros sites consultados:

<http://www.forumweb.com.br/desenvolvimento/aspnet/introducao-ao-ado-net/>

<http://www.forumweb.com.br/desenvolvimento/aspnet/introducao-ao-ado-net/>

[http://www.macoratti.net/vbn\\_dts1.htm](http://www.macoratti.net/vbn_dts1.htm)

<http://www.wwwcoder.com/parentid/459/tabid/68/type/art/site/6454/default.aspx>

<http://msdn.microsoft.com/pt-br/library/cc517967.aspx>