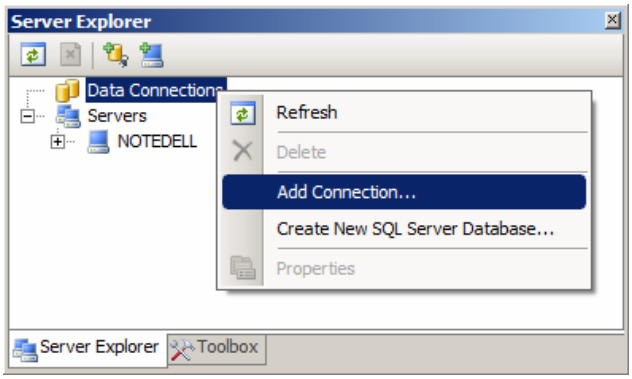
1. Criar novo projeto chamado ConsultaProdutos

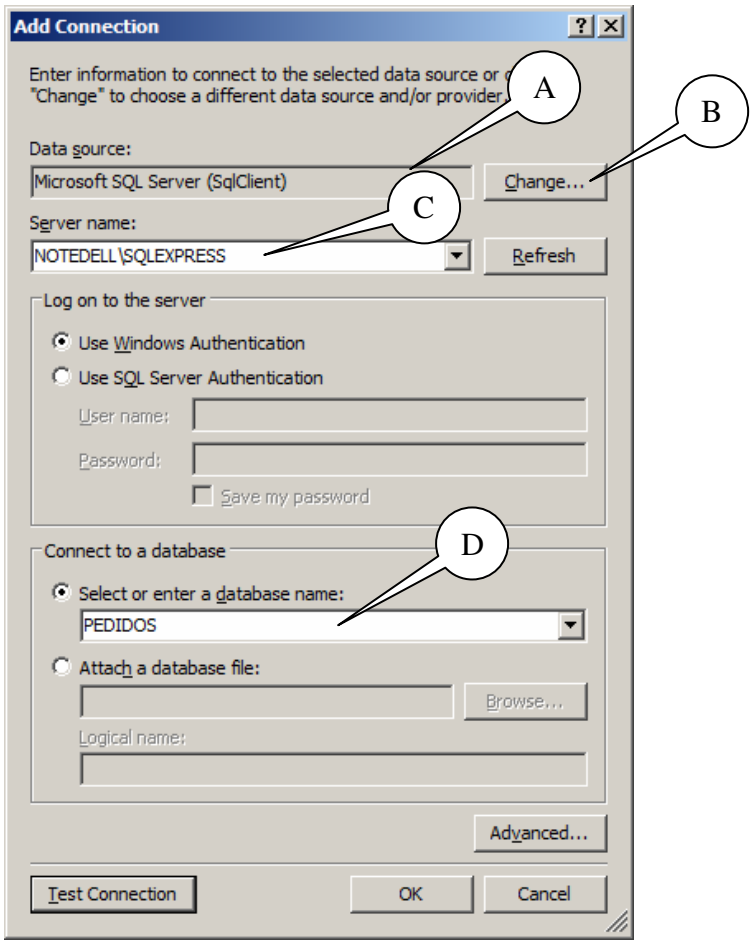
   1.1. Definir o string de conexão para o banco de dados.

   Se o Server Explorer não estiver visível abra-o utilizando a opção View - Server Explorer". Nós vamos utilizá-lo para definir um string de conexão para o banco de dados PEDIDOS.

   Com um clique direito sobre Data Connections, selecione Add Connection:



   Aparecerá a tela a seguir:



A. Driver de acesso ao banco. Este que está selecionado nos obrigará a utilizar a classe SqlConnection para configurar e estabelecer a conexão. Esta classe se conecta somente a bancos de dados MS-SQL Server.

B. Permite alterar o driver. Selecione Microsoft Sql Server (Sql Client). Este driver permite apenas conexões com Microsoft Sql Server.

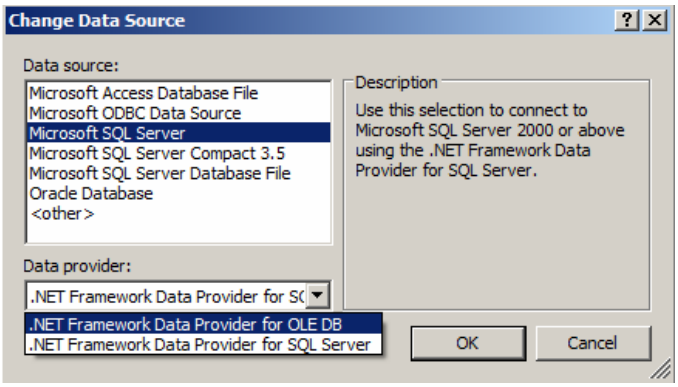
C. Nome do servidor sql.

D. Nome do banco de dados.

Na janela de propriedades veremos a propriedade Connection String contendo

Data Source=<nomeDoServidor>\SQLEXPRESS;Initial Catalog=PEDIDOS;Integrated Security=True

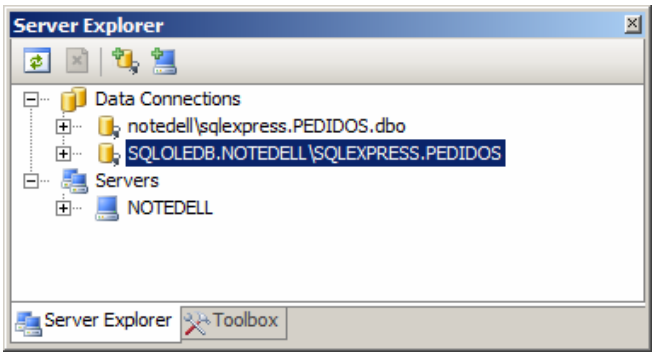
Crie uma nova conexão, mas agora clique no botão "Change" (B) para mudar o driver de conexão.



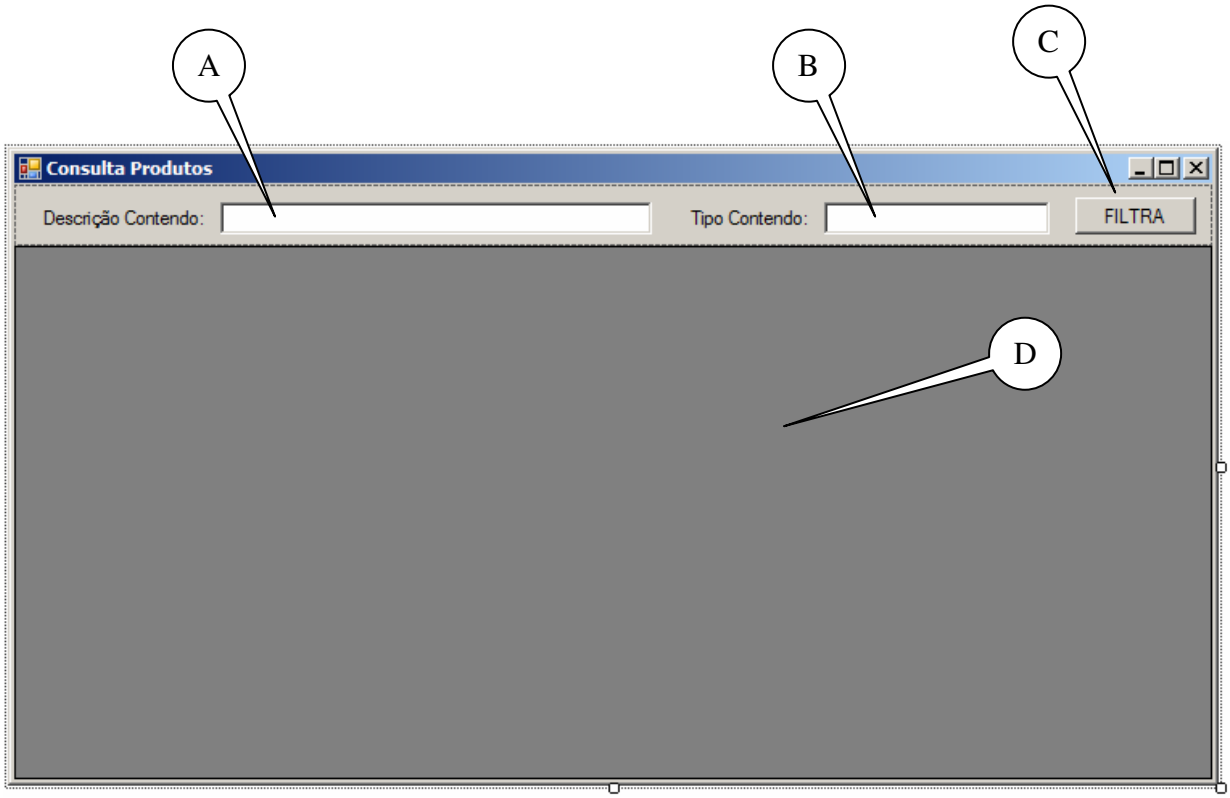
Mantenha "Microsoft SQL Server" selecionado, mas altere o Data provider para ".Net Framework Data Provider for OLE DB".

Desta forma teremos 2 tipos de conexão com o SQL Server, a primeira utilizando o driver nativo do SQL Server e a segunda usando OLE DB provider. A vantagem da conexão OLE DB é que dentro do Visual Studio ela utiliza uma classe chamada OleDbConnection, que é capaz de conectar a qualquer banco de dados, desde que tenhamos o correspondente OLE DB instalado. Já a conexão SqlClient usa a classe SqlConnection que conecta somente no Microsoft SQL Server.

Temos agora 2 tipos de conexão definidas:



1.2. Montagem da tela de consulta.



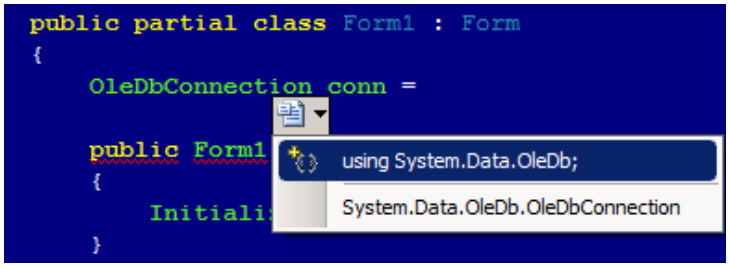
A. TextBox: Name = tbxDescricao
B. TextBox: Name = tbxTipo
C. Button: Name = btnFiltra
D. DataGridView: Name = dgv

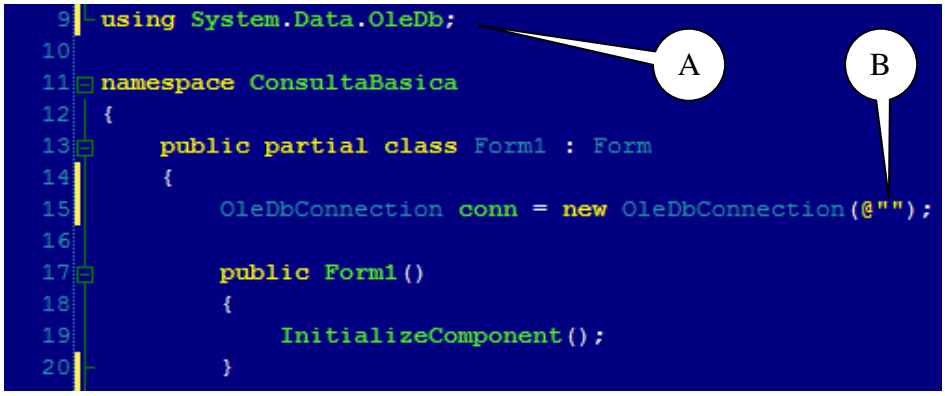
1.3. Declaração do objeto para conexão com o banco de dados.

Logo após a declaração da classe vamos declarar um objeto da classe OleDbConnection:

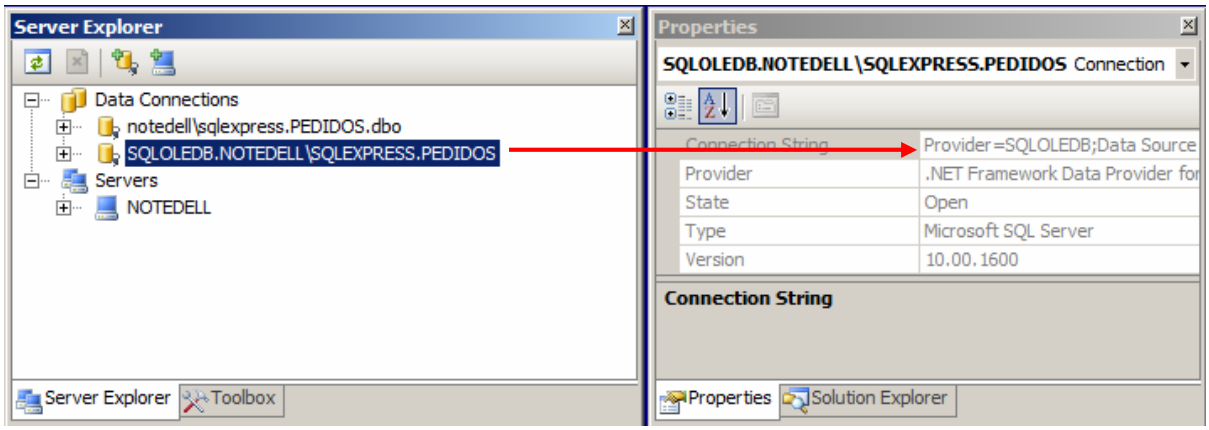
```
namespace ConsultaBasica
{
    public partial class Form1 : Form
    {
        OleDbConnection conn = new OleDbConnection(@"");

        public Form1()
        {
            InitializeComponent();
        }
...
...
```

Ao começar digitar você vai perceber que o VS não está reconhecendo a classe OleDbConnection (falta o Namespace).





A. Namespace inserido.
B. Copie para cá o string de conexão OLE DB da propriedade Connection String:




```
namespace ConsultaBasica
{
  public partial class Form1 : Form
  {
    OleDbConnection conn =
     new OleDbConnection(@"Provider=SQLOLEDB;Data Source=NOTEDELL\SQLEXPRESS;" +
                         "Integrated Security=SSPI;Initial Catalog=PEDIDOS");
     // Objeto DataTable para armazenar o retorno da consulta
     DataTable tbProdutos = new DataTable();
     // Objeto para gerenciar o ponteiro e transferir os dados para a tela
     BindingSource bsProdutos = new BindingSource();

     public Form1()
     {
         InitializeComponent();
     }
...
...
```

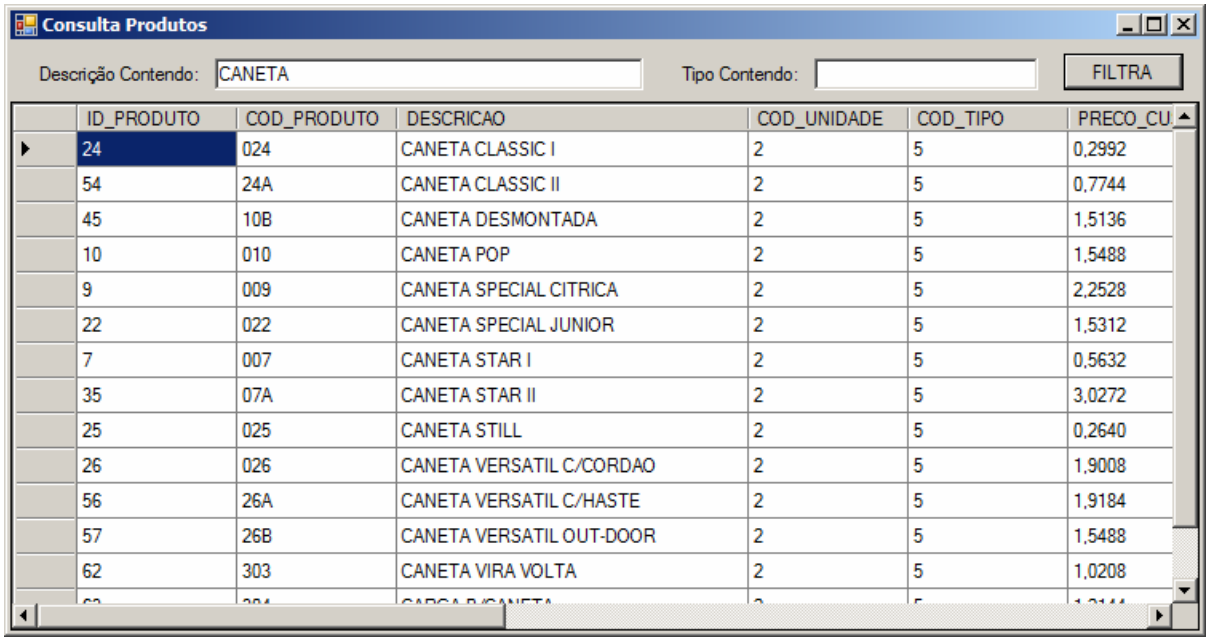
1.4. Método Click do botão btnFiltra:

```
private void btnFiltra_Click(object sender, EventArgs e)
{
    OleDbCommand cmd = conn.CreateCommand();

    cmd.CommandText = @"SELECT P.ID_PRODUTO, P.COD_PRODUTO, P.DESCRICAO, T.TIPO, U.UNIDADE,
                               P.PRECO_VENDA, P.QTD_REAL
                        FROM PRODUTOS P
                             JOIN TIPOPRODUTO T ON P.COD_TIPO = T.COD_TIPO
                             JOIN UNIDADES U ON P.COD_UNIDADE = U.COD_UNIDADE
                        WHERE P.DESCRICAO LIKE ? AND
                              T.TIPO LIKE ?
                        ORDER BY P.DESCRICAO";

cmd.Parameters.AddWithValue("descr", "%" + tbxDescricao.Text + "%")
cmd.Parameters.AddWithValue("tipo", "%" + tbxTipo.Text + "%")

    OleDbDataAdapter da = new OleDbDataAdapter(cmd);

    tbProdutos.Clear()
    da.Fill(tbProdutos);
    bsProdutos.DataSource = tbProdutos;
    dgv.DataSource = bsProdutos;
}
```

Obs.: O ? dentro do comando SQL é um parâmetro, uma variável cujo valor é passado posteriormente ao comando.

1.5. Crie um método para o evento Load do formulário, forçaremos a execução do SELECT quando o formulário for aberto.

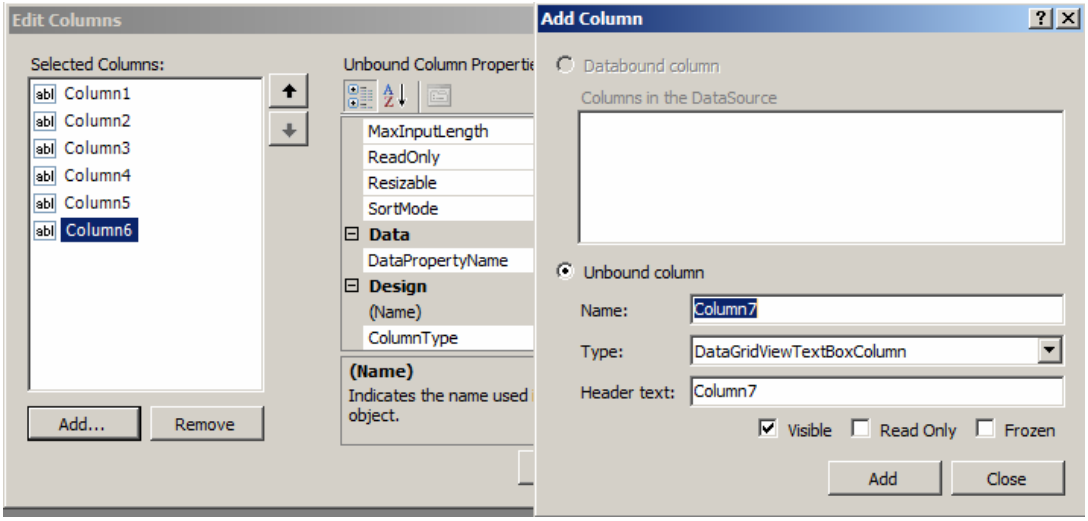
```
private void Form1_Load(object sender, EventArgs e)
{
    btnFiltra.PerformClick();
}
```



ID_PRODUTO	COD_PRODUTO	DESCRICAO	COD_UNIDADE	COD_TIPO	PRECO_CU
24	024	CANETA CLASSIC I	2	5	0,2992
54	24A	CANETA CLASSIC II	2	5	0,7744
45	10B	CANETA DESMONTADA	2	5	1,5136
10	010	CANETA POP	2	5	1,5488
9	009	CANETA SPECIAL CITRICA	2	5	2,2528
22	022	CANETA SPECIAL JUNIOR	2	5	1,5312
7	007	CANETA STAR I	2	5	0,5632
35	07A	CANETA STAR II	2	5	3,0272
25	025	CANETA STILL	2	5	0,2640
26	026	CANETA VERSATIL C/CORDAO	2	5	1,9008
56	26A	CANETA VERSATIL C/HASTE	2	5	1,9184
57	26B	CANETA VERSATIL OUT-DOOR	2	5	1,5488
62	303	CANETA VIRA VOLTA	2	5	1,0208

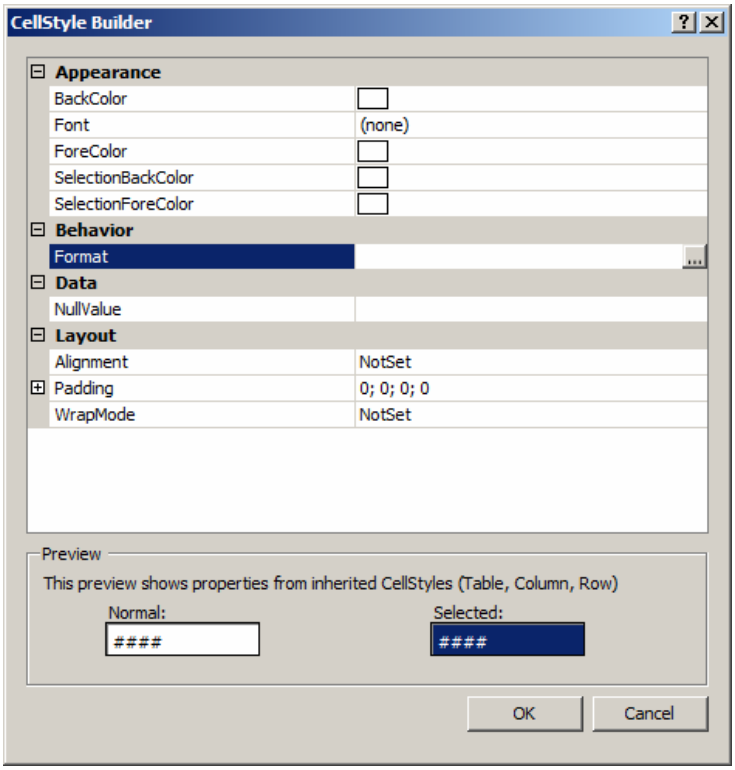
Execute o projeto para testar

1.6. Formatar colunas do grid: Com um clique direito sobre o DataGridView, selecione "Edit Columns" e adicione 6 colunas.



Propriedades importantes de cada coluna:

DataPropertyName: Nome do campo que será exibido na coluna.
HeaderText: Título da coluna.
Width: Largura da coluna.
DefaultCellStyle: Formatação das células. Abre a seguinte janela:



Configuração de Column1:

DataPropertyName: ID_PRODUTO
HeaderText: ID:
Width: 30

Configuração de Column2:

DataPropertyName: COD_PRODUTO
HeaderText: Código
Width: 50

Configuração de Column3:

DataPropertyName: DESCRICAO
HeaderText: Descrição
Width: 250

Configuração de Column4:

DataPropertyName: TIPO
HeaderText: Tipo
Width: 200

Configuração de Column5:

DataPropertyName: UNIDADE
HeaderText: Unidade
Width: 100

Configuração de Column6:

DataPropertyName: PRECO_VENDA
HeaderText: Pr. Venda
Width: 60
DefaultCellStyle.Format: #,##0.00
DefaultCellStyle.Alignment: MiddleRight

Configuração de Column7:

DataPropertyName: QTD_REAL
HeaderText: Qtd.Estoque:
Width: 70
DefaultCellStyle.Format: #,##0
DefaultCellStyle.Alignment: MiddleRight

Execute o projeto para testar

1.7. Criando uma classe estática para gerar o objeto de conexão. Isso facilitará a alteração do string de conexão porque esta classe poderá ser usada em todos os formulários do projeto.

Adicione um novo item ao projeto e selecione "Class" e de o nome Conexoes.cs.

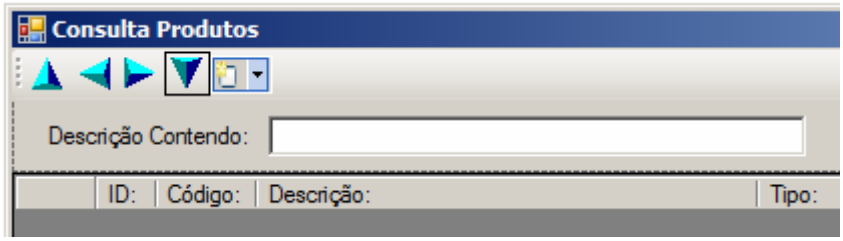
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data.OleDb;

namespace ConsultaBasica
{
    static class Conexoes
    {
        public static SqlConnection ConexaoSql()
        {
            return new SqlConnection(
            @"Data Source=localhost\sqlexpress;Initial Catalog=PEDIDOS;Integrated Security=True");
        }

        public static OleDbConnection ConexaoOle()
        {
            return new OleDbConnection(
   @"Provider=SQLOLEDB;Data Source=localhost\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=PEDIDOS");
        }
    }
}
```

- Observe que temos um método para gerar uma conexão SqlConnection e outro para conexão OleDb.
- Copie o string de conexão que criamos anteriormente para dentro dos métodos.
- Altere a criação do objeto de conexão criado no formulário para:

OleDbConnecion conn = Conexoes.ConexaoOle()

Execute o projeto para testar

- Coloque uma ToolStrip no topo do formulário e acrescente 4 botões:



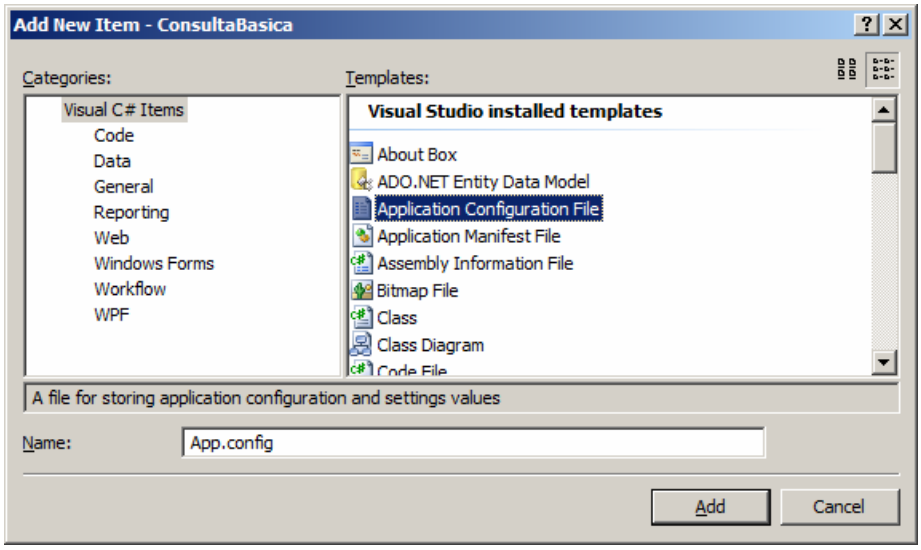
- Altere os nomes dos botões para tbtnPrimeiro, tbtnAnterior, tbtnProximo e tbtnUltimo respectivamente.
- Utilize a propriedade Image de cada botão para selecionar a figura. Essas figuras podem ser encontradas na **pasta 1_Images** disponibilizada pelo instrutor.
- Faça o evento Click de cada um deles:

```
private void tbtnPrimeiro_Click(object sender, EventArgs e)
{
    bsProdutos.MoveFirst();
}

private void tbtnAnterior_Click(object sender, EventArgs e)
{
    if (bsProdutos.Position > 0)
        bsProdutos.MovePrevious();
    else
        MessageBox.Show("Início do arquivo...");
}

private void tbtnProximo_Click(object sender, EventArgs e)
{
    if (bsProdutos.Position < tbProdutos.Rows.Count - 1)
        bsProdutos.MoveNext();
    else
        MessageBox.Show("Fim do arquivo...");
}

private void tbtnUltimo_Click(object sender, EventArgs e)
{
    bsProdutos.MoveLast();
}
```

Execute o projeto para testar

1.8. Criando arquivo App.config para configurar a conexão com o banco fora do executável.

- Click direito sobre o projeto "Add - New Item"
- Selecione Application Configuration File



- O arquivo terá o seguinte conteúdo inicialmente.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
</configuration>
```

- Altere como mostrado a seguir:.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="conexaoOleDb"
    connectionString="copie aqui o string de conexão OLE DB"
    providerName="System.Data.OleDb" />
  </connectionStrings>
</configuration>
```

- Altere a classe Conexoes de modo a utilizar o arquivo de configuração:

  Inclua o namespace:

```
using System.Configuration;
```

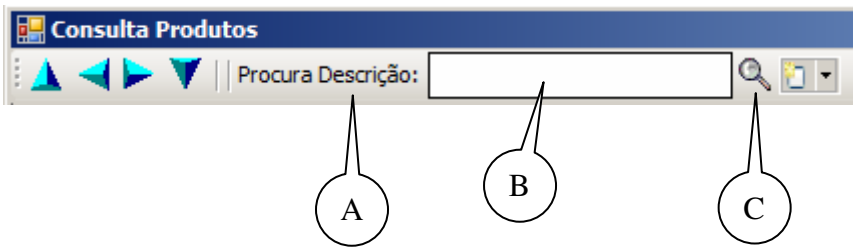
Obs.: Precisa adicionar referência (DLL) System.configuration.

```
public class Conexoes
{
  public static SqlConnection ConexaoSql()
  {
     string connSQL = ConfigurationManager.ConnectionStrings["conexaoSQL"].ConnectionString;
     return new SqlConnection(connSQL);
  }

  public static OleDbConnection ConexaoOle()
  {
      string connOleDb = ConfigurationManager.ConnectionStrings["conexaoOleDb"].ConnectionString;
      return new OleDbConnection(connOleDb);
  }
}
```

Execute o projeto para testar

1.9. Criando uma opção de busca:

- Altere a toolStrip como mostra a figura a seguir.



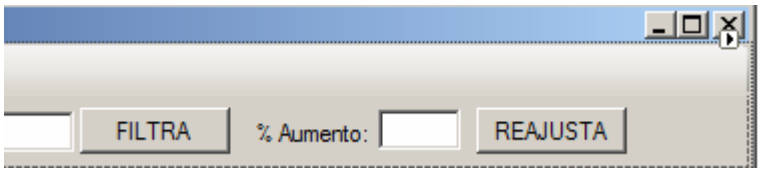
A. ToolStripLabel
B. ToolStripTextBox: Name = ttbxDec
C. ToolStripButton: Name tbtbProcura

- Evento click do botão tbtnProcura:

```
private void tbtnProcura_Click(object sender, EventArgs e)
{
    int pos = -1;
    for (int i = 0; i < tbProdutos.Rows.Count; i++)
    {
        string conteudoCampo = tbProdutos.Rows[i]["DESCRICAO"].ToString();
        if (conteudoCampo.ToUpper().StartsWith(ttbxDesc.Text.ToUpper()))
        {
            pos = i;
            bsProdutos.Position = pos;
            break;
        }
    }
    if (pos < 0) MessageBox.Show(ttbxDesc.Text + " Não Encontrado...");
}
```

1.10. Alterando os preços dos produtos filtrados:

- Altere a tela incluindo os seguintes controles:



- Evento click do botão REAJUSTA

```
private void btnReajusta_Click(object sender, EventArgs e)
{
    OleDbCommand cmd = conn.CreateCommand();
    double fator = 1 + Convert.ToDouble(tbxPorc.Text)/100;
    cmd.CommandText = "UPDATE PRODUTOS SET PRECO_VENDA = PRECO_VENDA * " +
                            fator.ToString().Replace(",", ".") +
                      " FROM PRODUTOS P " +
                      "     JOIN TIPOPRODUTO T ON P.COD_TIPO = T.COD_TIPO " +
                      "     JOIN UNIDADES U ON P.COD_UNIDADE = U.COD_UNIDADE " +
                      " WHERE P.DESCRICAO LIKE '%" + tbxDescricao.Text + "%' AND " +
                      "T.TIPO LIKE '" + tbxTipo.Text + "%' " ;
    if (conn.State == ConnectionState.Closed) conn.Open();
     cmd.ExecuteNonQuery();
    conn.Close();
    btnFiltra.PerformClick();
}
```

EXERCÍCIO 1:

Criar novo projeto chamado "Clientes" com as seguintes características:

- Permitir consulta aos seguintes campos da tabela CLIENTES:

  CODCLI
  NOME
  ENDERECO
  BAIRRO
  CIDADE
  ESTADO
  CEP
  FONE1
  FAX
  E_MAIL
  CNPJ
  INSCRICAO

- Deve permitir filtrar por:

  Nome contendo...
  Cidade contendo
  Estado (UF) começando com

- Deve ter botões de navegação.

EXERCÍCIO 2:

Criar projeto para mostrar seguintes campos da tabela EMPREGADOS:

CODFUN
NOME
DATA_ADMISSAO
SALARIO
DEPTO (TABELADEP)
CARGO (TABELACAR)

A tela deve permitir filtrar por:

Nome contendo
Data de admissão entre data inicial e data final
Salário entre salário inicial e salário final
Depto começando com
Cargo começando com

Permitir o reajuste de salário dos funcionários do filtro.