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

DESENVOLVIMENTO DE SISTEMAS

ADO.NET

O ADO.NET é uma tecnologia de acesso a banco de dados que oferece diversas classes que fornecem inúmeros serviços de operações relacionadas a banco de dados, permitindo o acesso a diferentes plataformas, tais como: SQL Server, MySQL, Oracle, Sybase, Access, XML, arquivos textos, etc. Essas conexões podem ser realizadas de três formas diferentes: OLE DB , SQL e ODBC.

Os provedores de dados que acompanham o ADO.NET, permitem a utilização de várias classes que interagem diretamente com a base de dados, as quais são identificadas por um prefixo, conforme tabela abaixo:

Provedor	Descrição
ODBC Data Provider API Prefixo: Odbc	Geralmente usada para banco de dados mais antigos, que utilizam a interface ODBC.
OleDb Data Provider API Prefixo: OleDb	Conexão do tipo OleDb, como por exemplo o Access ou Excel;
Oracle Data Provider API Prefixo:Oracle	Para implementação de Banco de Dados Oracle.
SQL Data Provider API Prefixo:Sql	Para implementação de Banco de Dados Microsoft SQL Server.

ADO.NET – MySQL

O provedor do MySQL não faz parte diretamente da tecnologia ADO.NET, portanto, este provedor será incluído no projeto manualmente. O nome do arquivo (provedor) que será incluído é o **MySQL.Data.Dll**, que se encontra na pasta do MySQL Server instalado no computador. Por exemplo:

C:\Program Files (x86)\MySQL\Connector NET 6.7.4\Assemblies\v2.0\MySQL.Data.Dll
C:\Program Files (x86)\MySQL\Connector NET 6.7.4\Assemblies\v4.0\MySQL.Data.Dll
C:\Program Files (x86)\MySQL\Connector NET 6.7.4\Assemblies\v4.5\MySQL.Data.Dll

Antes de selecionar a biblioteca desejada, verificar a versão do Framework na qual está sendo desenvolvida a aplicação. As versões dos Framework´s são:

- ✓ **Visual Studio 2005 → Framework Versão 2.0**
- ✓ **Visual Studio 2008 → Framework Versão 3.5**
- ✓ **Visual Studio 2010 → Framework Versão 4.0**
- ✓ **Visual Studio 2012 → Framework Versão 4.5**
- ✓ **Visual Studio 2013 → Framework Versão 4.5**

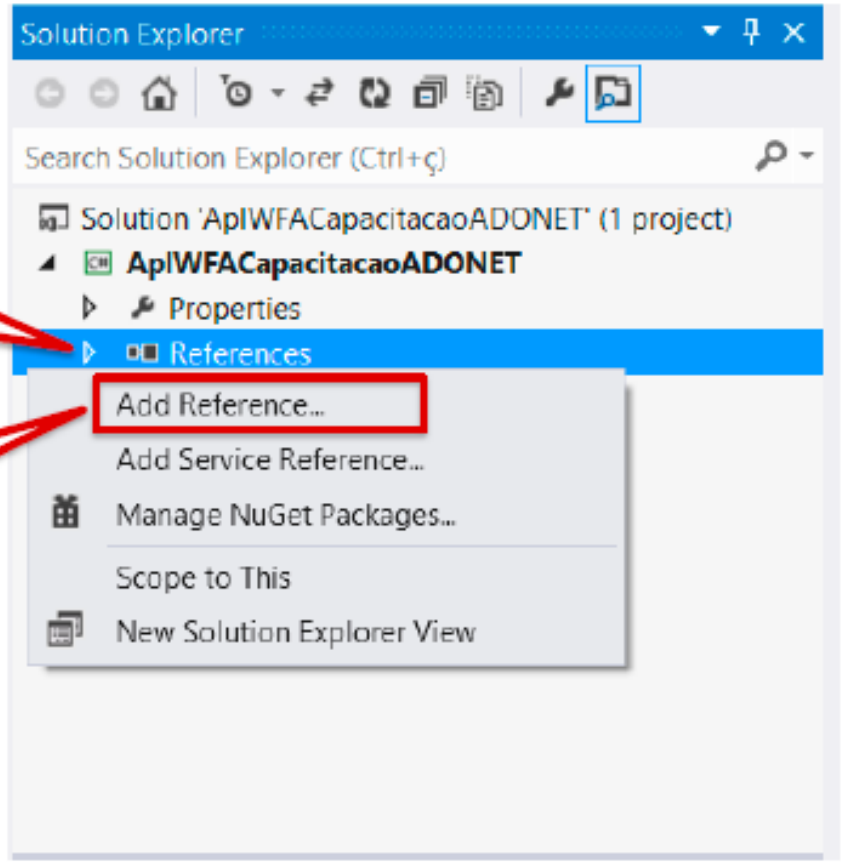
ADO.NET – MySQL

Para adicionar um provedor externo, ou seja, adicionar uma referência externa, devemos seguir os seguintes passos:

1. Na **Solution Explorer**, ao lado direito da tela, devemos clicar com o botão direito do mouse em cima da opção **References**, e aparecerá a seguinte tela:

A opção **References** permite adicionar qualquer biblioteca externa e/ou nativas do VisualStudio.

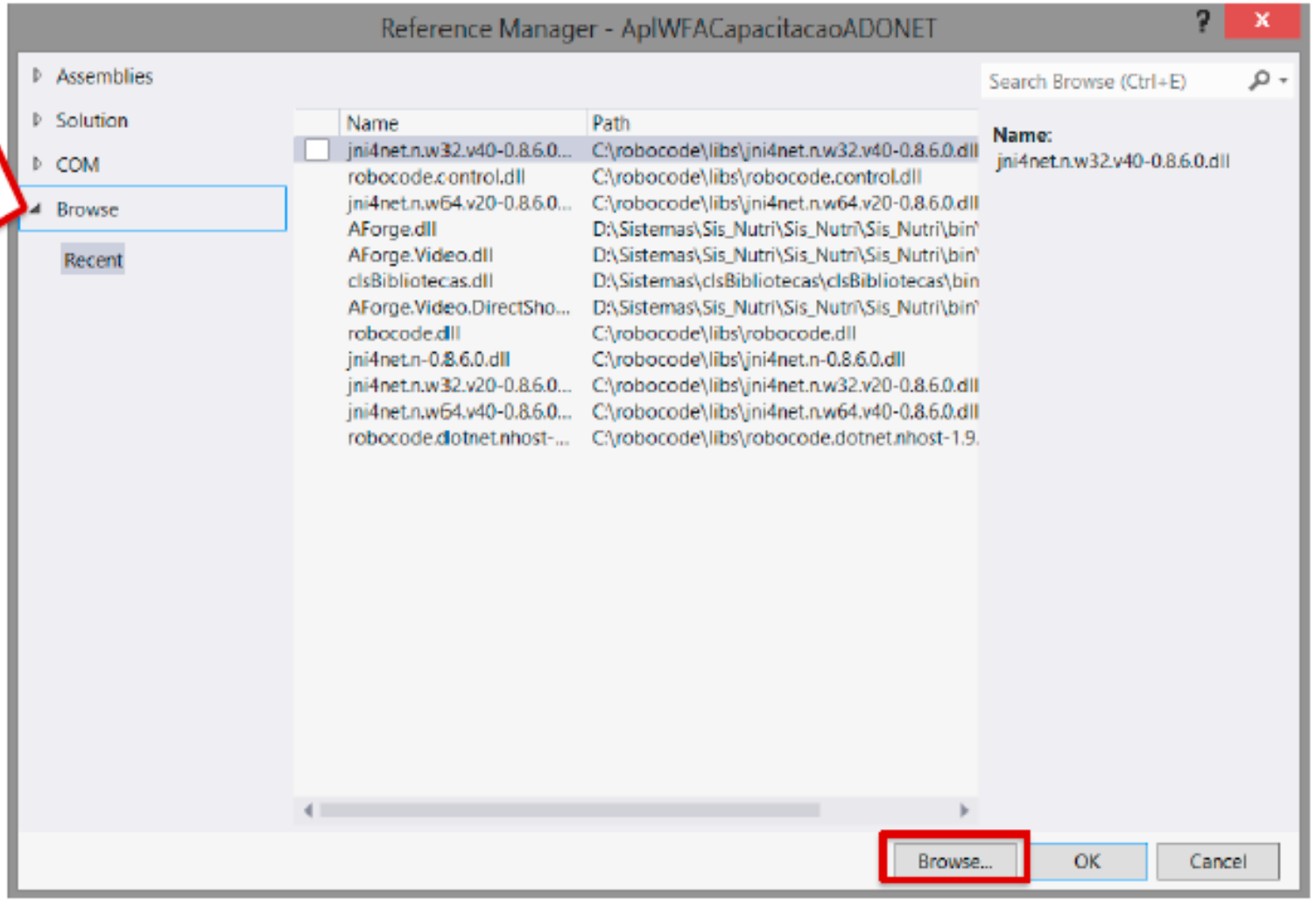
Clicar na opção **Add Reference...**, para selecionar o arquivo desejado.



ADO.NET – MySQL

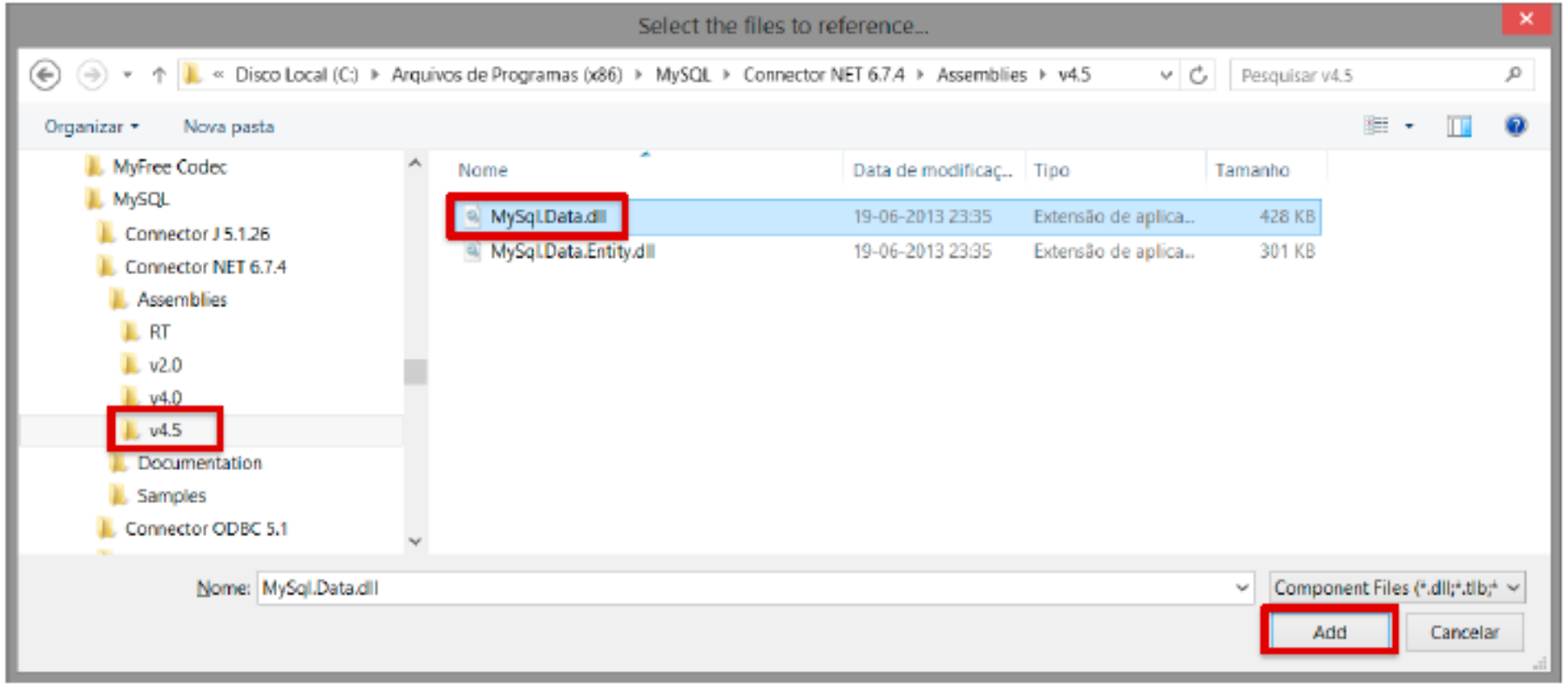
2. Após clicar no botão **Add References...**, aparecerá a seguinte tela:

Na opção Browse, mostrará todas as bibliotecas que já foram incluídas anteriormente no seu Visual Studio.



ADO.NET – MySQL

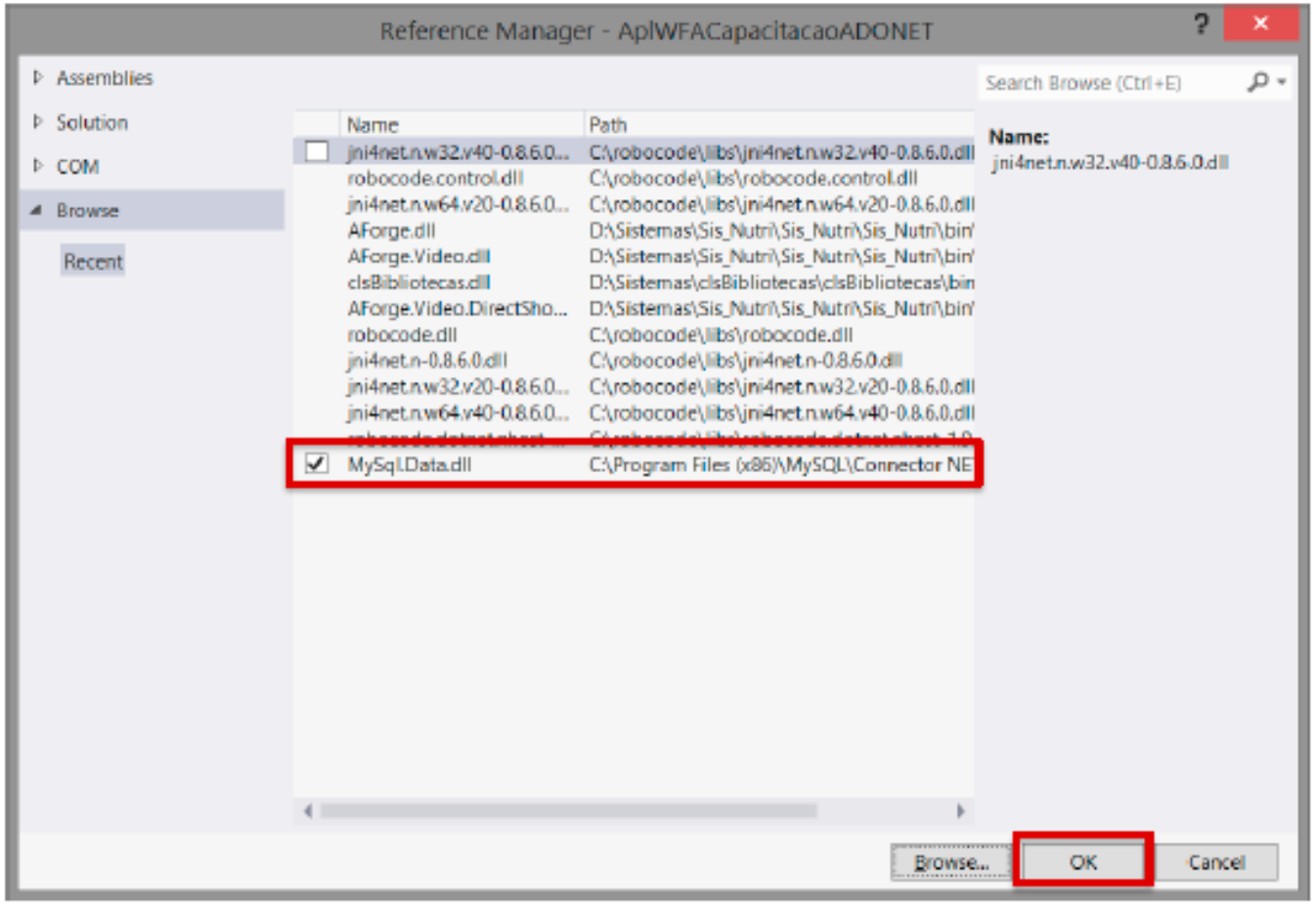
3. Após clicar no botão **Browse**, aparecerá a seguinte tela:
   a) Selecione o arquivo **MySQL.Data.Dll**, a partir da pasta que estão instalados os conectores do MySQL.
   b) Selecionar o conector para o ambiente **.NET**, como mostra a figura.
   c) Selecionar a versão do conector de acordo com a versão do Visual Studio **(Ver slide 5)**. Em seguida clicar no botão **Add**.



ADO.NET – MySQL

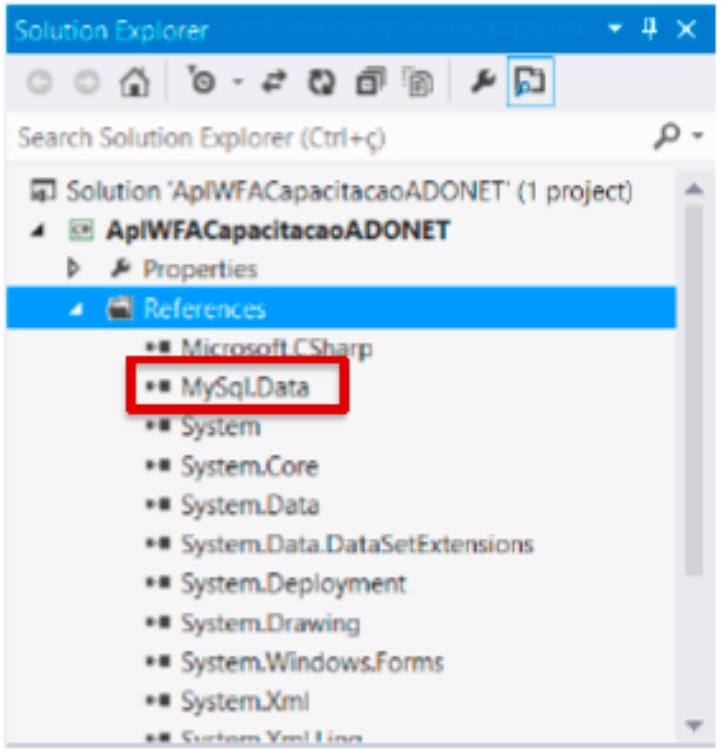
3. Após a seleção do arquivo (.dll), irá aparecer no gerenciamento de referências, como mostra a tela abaixo:

   a. Em seguida clicar no botão **OK**.



ADO.NET – MySQL

3. Após a seleção do arquivo (.dll), o mesmo irá aparecer na relação de referências, como mostra a tela abaixo:



A partir de agora, a biblioteca está pronta para ser usada no desenvolvimento.

Sempre que abrir o mesmo projeto, não precisará mais adicionar a referência MySQL.Data.Dll sempre que abrir o projeto.

ADO.NET – MySQL

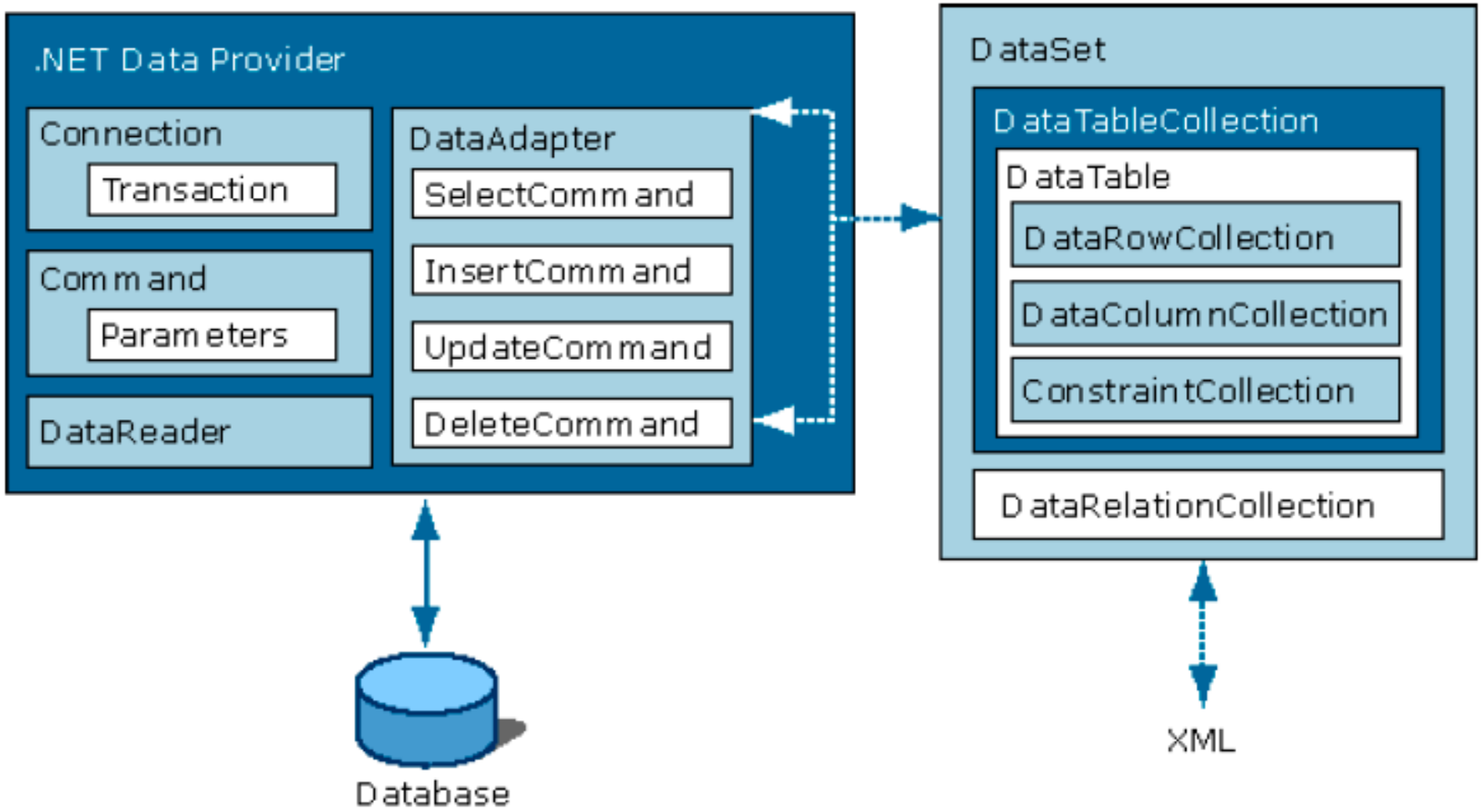
Para importarmos a biblioteca para o ambiente de desenvolvimento, devemos utilizar a instrução **using**, como mostra a figura abaixo:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace AplWFACapacitacaoADONET
{
    public partial class frmBancoDados : Form
    {
        public frmBancoDados()
        {
            InitializeComponent();
        }

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

****** IMPORTANTE ******

Sem adicionar a referência no projeto, NÃO será possível importar a classe (Provedor) MySQL.

ADO.NET – Arquitetura



ADO.NET – Modo Conectado

A arquitetura ADO.NET disponibiliza dois modos de conexão: **Conectado e Desconectado**.

No modo **Conectado** utilizamos os seguintes componentes:

- ✓ **Connection** → Classe (objeto) disponível para conexão com diversos bancos de dados.
- ✓ **Command** → Classe (objeto) disponível para enviar as instruções (select, insert, update, delete, etc) ao banco de dados. Para executar uma **Stored Procedure**, devemos utilizar uma propriedade **Parameters**, para enviar os parâmetros de entrada da **Stored Procedure**.
- ✓ **DataReader** → Classe (objeto) disponível para **receber os dados** de uma tabela após a execução de um **Select**. Este componente só funciona com a instrução **Select**, porque após a busca dos dados no banco a instrução devolverá uma resposta, se encontrou ou não, o que foi solicitado anteriormente. Toda resposta de um **Select** sempre será associado a um DataReader.

ADO.NET – Modo Desconectado

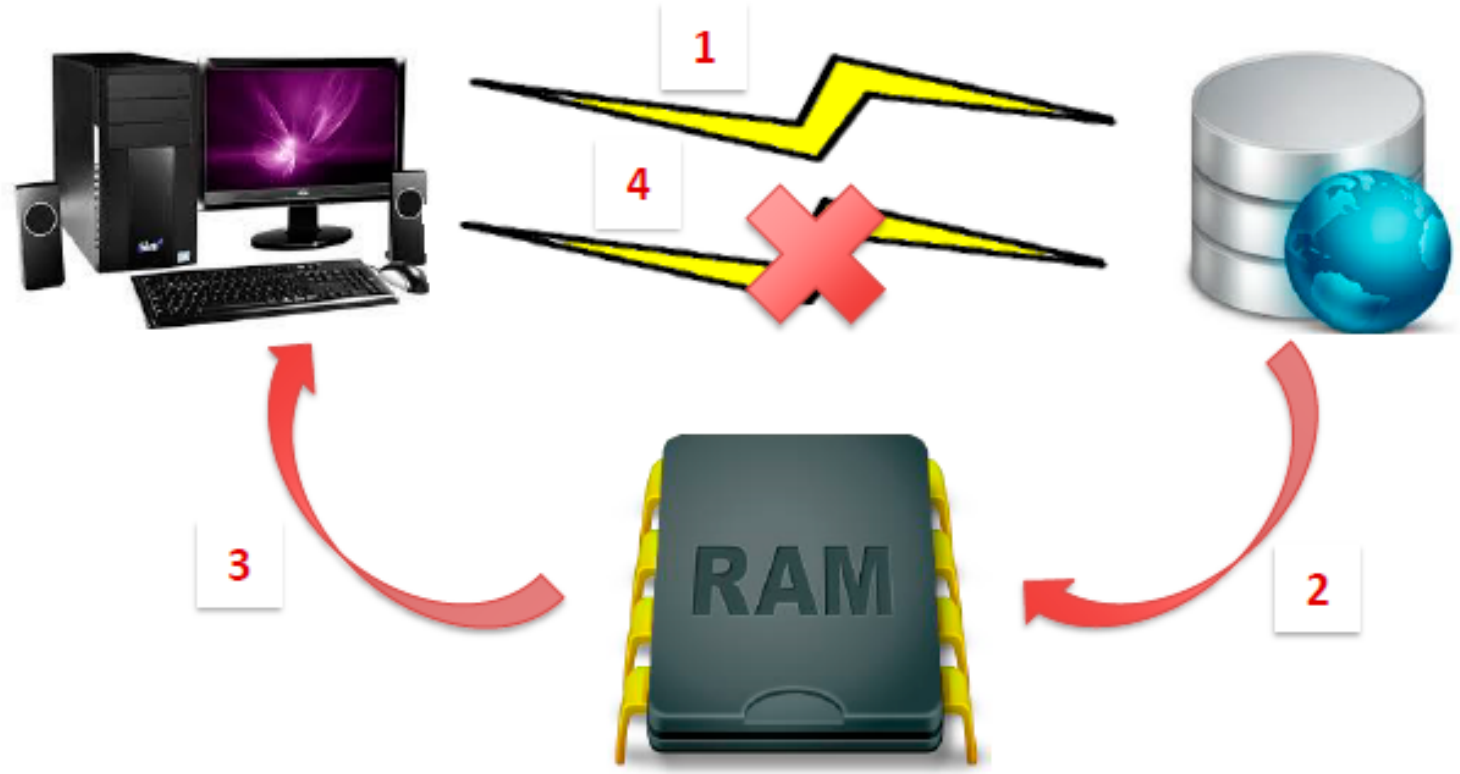
No modo **Desconectado** utilizamos os seguintes componentes:

- ✓ **Connection** → Classe (objeto) disponível para conexão com diversos bancos de dados.
- ✓ **Command** → Classe (objeto) disponível para enviar as instruções (select, insert, update, delete, etc) ao banco de dados. Para executar uma **Stored Procedure**, devemos utilizar uma propriedade **Parameters**, para enviar os parâmetros de entrada da **Stored Procedure**.
- ✓ **DataAdapter** → Classe (objeto) disponível para **receber os dados** de uma tabela após a execução de um **Select**.
- ✓ **DataSet** → O objeto Recordset (ADO), que armazena somente uma coleção de tabelas, entra em desvantagem com o DataSet, membro do System.Data, o qual passa a controlar uma cópia do banco de dados em memória, representando um conjunto de dados em memória cache que não está conectado com o banco de dados.
- ✓ **DataTable** → O objeto DataTable pode representar uma ou mais tabelas de dados, as quais permanecem alocadas em memória e pode ser manipulado através de métodos.
- ✓ **DataView** → As operações de pesquisa, ordenação e navegação pelos dados, podem ser feitas através do DataView, que permite a ligação da fonte de dados com a interface do usuário, portanto, utilizamos um DataView para visualizar as informações contidas em DataTable.

ADO.NET – Modo Desconectado

No modo **desconectado** fazemos:

1. Conexão com o Banco de Dados;
2. Realizamos a busca através do DataAdpater pelo SelectCommand;
3. Os dados retornados ficam disponíveis na memória RAM, para consulta, entre outros;
4. Desconecta-se do banco da dados;



ADO.NET – Principais Pacotes

Os principais pacotes utilizados pelo ADO.NET, são:

- ✓ **System.Data:** contém as classes que representam tabelas, colunas, linhas e também a classe DataSet de todos os provedores, além das interfaces: IDbCommand, IDbConnection, e IDbDataAdapter que são usadas por todos os provedores de conexão.
- ✓ **System.Data.Commom:** Define as classes para os provedores de dados: DbConnection e DbDataAdapter.
- ✓ **System.Data.OleDb:** Fonte de dados OLE DB usando o provedor .NET OleDb.
- ✓ **System.Data.Odbc:** Fonte de dados ODBC usando o provedor .NET Odbc.
- ✓ **System.Data.SqlTypes:** Dados específicos para o SQL Server.

Além disso o ADO.NET, oferece classes referenciadas:

- ✓ **Disconnected:** Fornece classes que são capazes de armazenar dados sem a dependência da fonte de dados de um determinado provedor, como por exemplo: DataTable.
- ✓ **Shared:** São classes que podem ser acessadas por todos os provedores.
- ✓ **Data Providers:** São classes utilizadas em diferentes fontes de dados, para gerenciamento.

SQL – Grupos

As instruções SQL são divididas por grupos. São eles:

- ✓ DTC (Data Type Commands)
- ✓ DDL (Data Definition Language)
- ✓ DQL (Data Query Language)
- ✓ DML (Data Manipulation Language)
- ✓ DCL (Data Control Language)
- ✓ SRC (Stored Routines Commands)

DTC – Data Type Commands

- ✓ **BIGINT** ➔ Números inteiros.
- ✓ **BINARY** ➔ Números binários.
- ✓ **BIT** ➔ Número inteiro. Só armazena 0 ou 1.
- ✓ **CHAR** ➔ Caractere.
- ✓ **DATETIME** ➔ Armazena data no formato "yyyy-mm-dd" e/ou hora no formato "HH:mm:ss"
- ✓ **DECIMAL** ➔ Números reais. Usado para campos monetários.
- ✓ **FLOAT** ➔ Números reais.
- ✓ **INT** ➔ Números inteiros.
- ✓ **SMALINT** ➔ Números inteiros.
- ✓ **TEXT** = Sequência de caracteres. Campo MEMO.
- ✓ **TINYINT** ➔ Números inteiros.
- ✓ **VARCHAR** ➔ Cadeia de caracteres.

DDL - Data Definition Language

- ✓ **ALTER TABLE:** alterar a estrutura de uma tabela.
- ✓ **ALTER VIEW:** alterar a estrutura da tabela virtual.
- ✓ **CREATE DATABASE:** criar banco de dados.
- ✓ **CREATE INDEX:** criar um índice para a tabela.
- ✓ **CREATE TABLE:** criar uma tabela no banco de dados.
- ✓ **CREATE VIEW:** criar uma tabela virtual cujo conteúdo (colunas e linhas) é definido por uma consulta. Use esta instrução para criar uma exibição dos dados em uma ou mais tabelas no banco de dados.
- ✓ **DROP DATABASE:** eliminar um banco de dados.
- ✓ **DROP INDEX:** eliminar um índice de uma tabela.
- ✓ **DROP TABLE:** eliminar uma tabela do banco de dados.
- ✓ **DROP VIEW:** eliminar uma tabela virtual.
- ✓ **USE:** Abrir um banco de dados.

DQL - Data Query Language

- ✓ **SELECT:** permite ao usuário especificar uma consulta ("query") como uma descrição do resultado desejado.

DML – Data Manipulation Language

- ✓ **SELECT:** permite ao usuário especificar uma consulta ("query") como uma descrição do resultado desejado.
- ✓ **INSERT** é usada para inserir um registro (formalmente uma tupla) a uma tabela existente.
- ✓ **UPDATE** para mudar os valores de dados em uma ou mais linhas da tabela existente.
- ✓ **DELETE** permite remover linhas existentes de uma tabela.
- ✓ **TRUNCATE:** remove rapidamente todas as linhas da tabela, esvaziando-a.
- ✓ **COMMIT:** efetiva a transação atualmente executada.
- ✓ **ROLLBACK:** desfaz a transação corrente, fazendo com que todas as modificações realizadas pela transação sejam rejeitadas.

DCL – Data Control Language

- ✓ **GRANT:** concede privilégios a um ou mais usuários para acessar ou realizar determinadas operações em um objetos de dados.
- ✓ **REVOKE:** revoga (remove) ou restringe a capacidade de um usuário de executar operações.
- ✓ **SET:** Define parâmetros em tempo de execução, como por exemplo, o tipo de codificação do cliente e o estilo de representação de data e hora.
- ✓ **LOCK:** Bloqueia explicitamente uma tabela fazendo o controle de acessos concorrente.


Listagem de Produtos
Produtos
Código
Produto
Descrição
Valor
Fornecedor
1 CANETA AZUL CANETA BIC AZUL 4,95 1
2 CANETA VERMELHA CANETA BIC VERMELHA 4,50 1
3 CANETA VERDE CANETA BIC VERDE 4,50 1
4 CANETA PRETA CANETA BIC PRETA 4,50 1
5 MARCADOR TEXTO MARCADOR TEXTO AZUL 5,55 2
6 MARCADOR TEXO MARCADOR TEXTO AMARELO 5,55 2
7 SULFITE A4 PACOTE C/500 FLS 14,55 2
8 SULFITE A3 PACOTE C500 17,67 3
9 PINCEL PILOT PINCEL P QUADRO BRANCO AZUL 5,55 4
10 PINCEL PILOT PINCEL P QUADRO BRANCO PRETO 5,55 4
11 PINCEL PILOT PINCEL P QUADRO BRANCO VERMELHO 5,55 4
12 PINCEL PILOT PINCEL P QUADRO BRANCO VERDE 5,55 4
13 RECARGA PINCEL RECARGA PINCEL AZUL 4,00 4
Exibir
Limpar
Sair

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10  using MySql.Data.MySqlClient;
11
12  namespace Win_Lista_POO
13  {
14      public partial class FrmListaPOO : Form
15      {
16          CldBancoDados bd = new CldBancoDados();
17          MySqlDataReader objDados;
18          StringBuilder strQuery = new StringBuilder();
19
20          public FrmListaPOO()
21          {
22              InitializeComponent();
23          }
24
```

```
25      private void FrmListaPOO_Load(object sender, EventArgs e)
26      {
27          strQuery.Append("Select * from TabProd");
28          objDados = bd.RetornaDataSet(strQuery.ToString());
29          while (objDados.Read())
30          {
31              dgvDados.Rows.Add(objDados["CodProd"].ToString(),
32                                objDados["NomeProd"].ToString(),
33                                objDados["DescProd"].ToString(),
34                                objDados["Valor"].ToString(),
35                                objDados["CodFor"].ToString());
36          }
37          if (!objDados.IsClosed) { objDados.Close(); strQuery.Remove(0, strQuery.Length); }
38      }
39
40      private void btnExibir_Click(object sender, EventArgs e)
41      {
42          FrmListaPOO_Load(sender, e);
43      }
44
45      private void btnLimpar_Click(object sender, EventArgs e)
46      {
47          dgvDados.Rows.Clear();
48      }
```

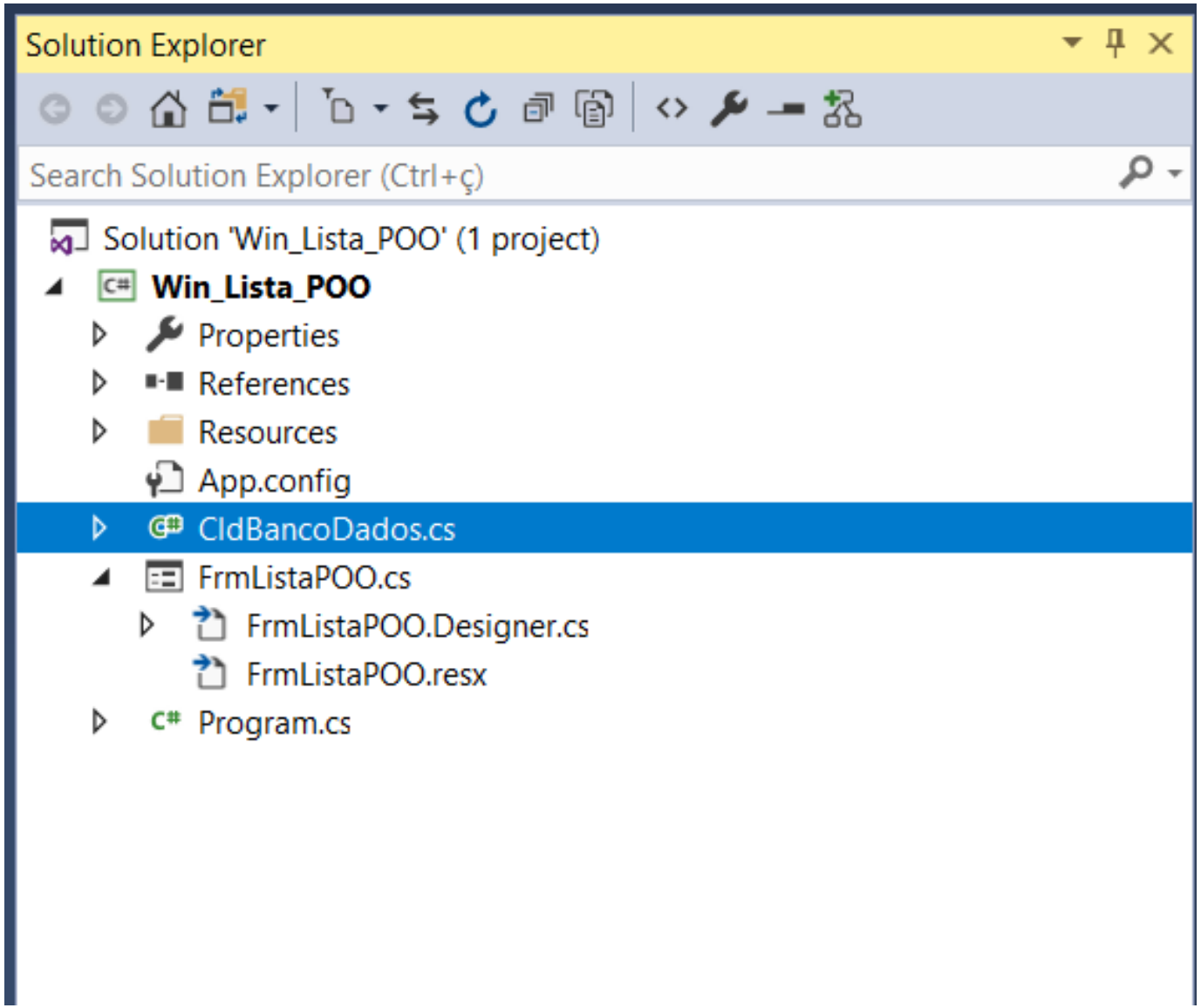
```
49
50          private void btnSair_Click(object sender, EventArgs e)
51          {
52              if(MessageBox.Show("Deseja Sair da Consulta Geral?",
53                                 "<<< FINALIZANDO >>>",
54                                  MessageBoxButtons.YesNo,
55                                  MessageBoxIcon.Question) == DialogResult.Yes)
56              {
57                  this.Close();
58              }
59          }
60      }
61  }
```

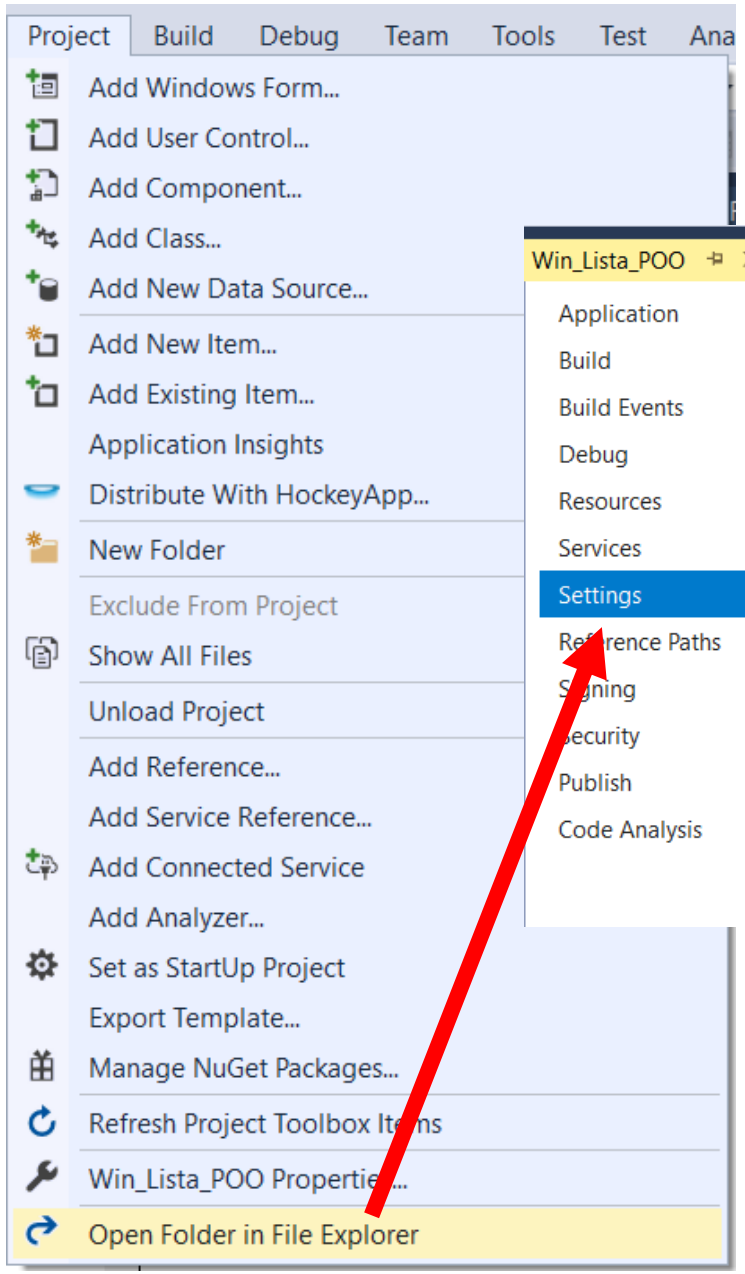
CldBancoDados.cs | FrmListaPOO.cs | FrmListaPOO.cs [Design]

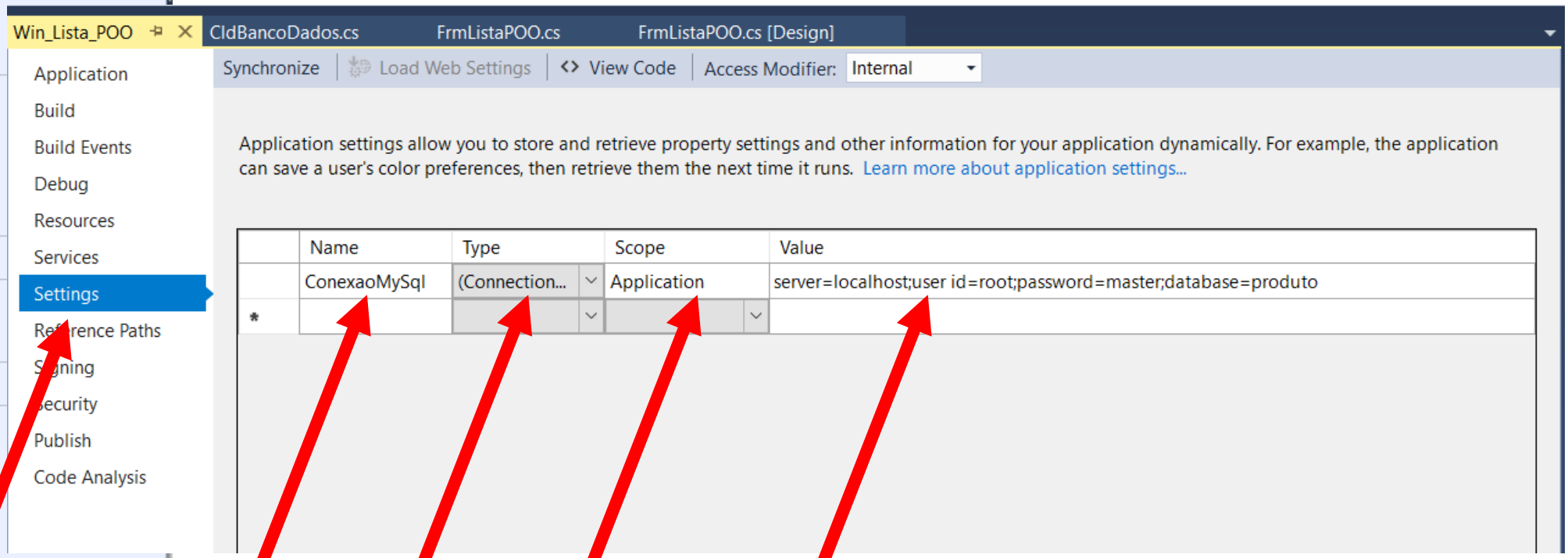
Win_Lista_POO | Win_Lista_POO.CldBancoDados | RetornaDataSet(string strQuery)

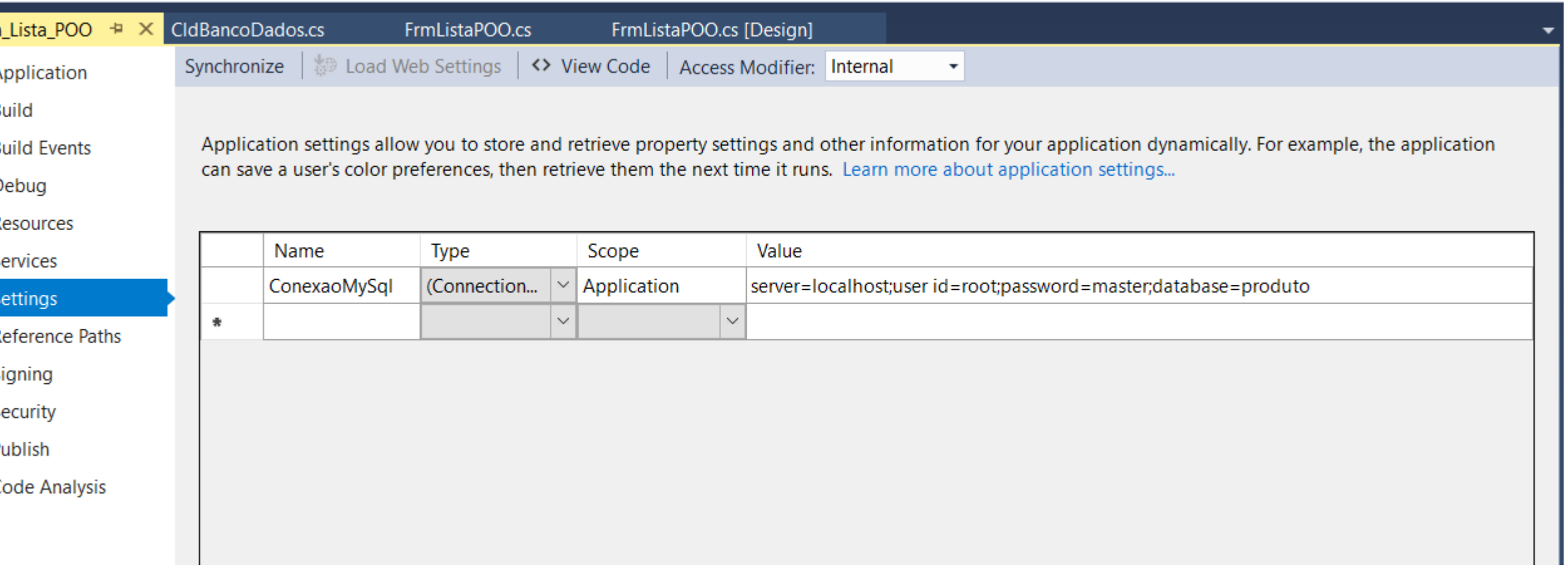
```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using System.Data;
7   using MySql.Data.MySqlClient;
8
9   namespace Win_Lista_POO
10  {
11      class CldBancoDados
12      {
13          //string con = Properties.Settings.Default.ConexaoMySql;
14          string con = "server = localhost; user id = root; password=master;database=produto;";
15          private MySqlConnection AbreBanco()
16          {
17              MySqlConnection ocon = new MySqlConnection();
18              ocon.ConnectionString = this.con;
19              ocon.Open();
20              return ocon;
21          }
22
23          private void FechaBanco(MySqlConnection ocon)
24          {
25              if (ocon.State == ConnectionState.Open)
26              {
27                  ocon.Close();
28              }
```

```
29          }
30
31          public MySqlDataReader RetornaDataSet(string strQuery)
32          {
33              MySqlConnection ocon = new MySqlConnection();
34              try
35              {
36                  ocon = AbreBanco();
37                  MySqlCommand cmdComando = new MySqlCommand();
38                  cmdComando.CommandText = strQuery;
39                  cmdComando.CommandType = CommandType.Text;
40                  cmdComando.Connection = ocon;
41                  return cmdComando.ExecuteReader();
42              }
43              catch (MySqlException ex)
44              {
45                  throw ex;
46              }
47          }
48
49      }
50  }
```


Solution Explorer
Search Solution Explorer (Ctrl+ç)
Solution 'Win_Lista_POO' (1 project)
Win_Lista_POO
Properties
References
Resources
App.config
CldBancoDados.cs
FrmListaPOO.cs
FrmListaPOO.Designer.cs
FrmListaPOO.resx
Program.cs


Project
Build
Debug
Team
Tools
Test
Add Windows Form...
Add User Control...
Add Component...
Add Class...
Add New Data Source...
Add New Item...
Add Existing Item...
Application Insights
Distribute With HockeyApp...
New Folder
Exclude From Project
Show All Files
Unload Project
Add Reference...
Add Service Reference...
Add Connected Service
Add Analyzer...
Set as StartUp Project
Export Template...
Manage NuGet Packages...
Refresh Project Toolbox Items
Win_Lista_POO Properties...
Open Folder in File Explorer


Win_Lista_POO
ClsBancoDados.cs
FrmListaPOO.cs
FrmListaPOO.cs [Design]
Application
Build
Build Events
Debug
Resources
Services
Settings
Reference Paths
Signing
Security
Publish
Code Analysis
Synchronize
Load Web Settings
View Code
Access Modifier:
Internal
Application settings allow you to store and retrieve property settings and other information for your application dynamically. For example, the application can save a user's color preferences, then retrieve them the next time it runs. Learn more about application settings...
Name
Type
Scope
Value
ConexaoMySql
(Connection...
Application
server=localhost;user id=root;password=master;database=produto


_Lista_POO
CldBancoDados.cs
FrmListaPOO.cs
FrmListaPOO.cs [Design]
Application
Build
Build Events
Debug
Resources
Services
Settings
Reference Paths
Signing
Security
Publish
Code Analysis
Synchronize
Load Web Settings
View Code
Access Modifier:
Internal
Application settings allow you to store and retrieve property settings and other information for your application dynamically. For example, the application can save a user's color preferences, then retrieve them the next time it runs. Learn more about application settings...
Name
Type
Scope
Value
ConexaoMySql
(Connection...
Application
server=localhost;user id=root;password=master;database=produto