

Linguagem de Programação Orientada a Objetos

Ambiente de Programação Microsoft .Net

C# (C sharp)

Conectando banco de dados MySQL

Prof. Laércio Silva

- Hoje há novas necessidades e desafios nos modelos de programação para acesso a dados em relação ao antigo modelo “cliente/servidor”.
- Há um crescimento no número de aplicações Web que rodam no cliente, sendo assim não se pode manter uma conexão aberta durante muito tempo.
- As aplicações hoje são desenvolvidas em multicamadas e há grandes desafios de como passar esses dados entre essas camadas.

- O modelo de programação atual pede que nós tenhamos um modelo desconectado.
- O ADO.net foi criado pensando justamente nessas novas necessidades e desafios do novo modelo de programação.
- Constitui-se, basicamente, em um conjunto de classes para trabalhar com dados.
- Inclui ferramentas para se conectar a um banco de dados, executar comandos e recuperar os resultados. Os resultados são processados diretamente, colocados em um objeto DataSet e exibidos para o usuário .
- ADO.NET é uma evolução do ADO (ActiveX Data Objects).

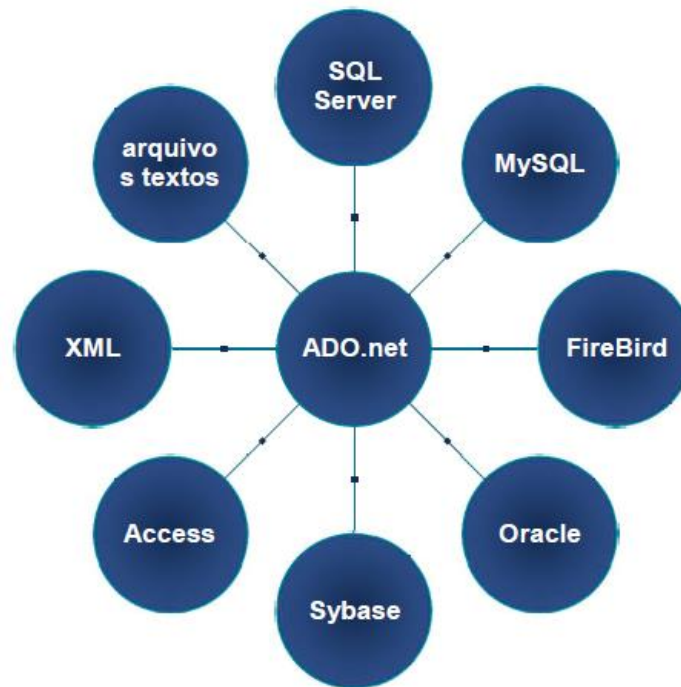
Características

- É compatível com aplicações de cenários conectados;
- É compatível com aplicações de cenários desconectados;
- É um modelo de programação com suporte avançado para o XML;
- Permite trabalhar com múltiplas fontes de dados diferentes ao mesmo tempo e de uma mesma maneira;

- Totalmente integrado ao .NET Framework;
- Como é nativo, sua API pode ser utilizada com as diversas linguagens que fazem parte do .NET Framework: Visual Basic, C#, entre outras;
- É sucessor do ADO, porém mais flexível;
- Possui um conjunto de Classes, Interfaces, Estruturas e Enumerações que gerem o acesso a dados dentro do .NET Framework;

Compatibilidade com fontes de dados.

Permite a comunicação com qualquer banco de dados relacionais entre outras fontes como arquivos CVS e Excel e etc.



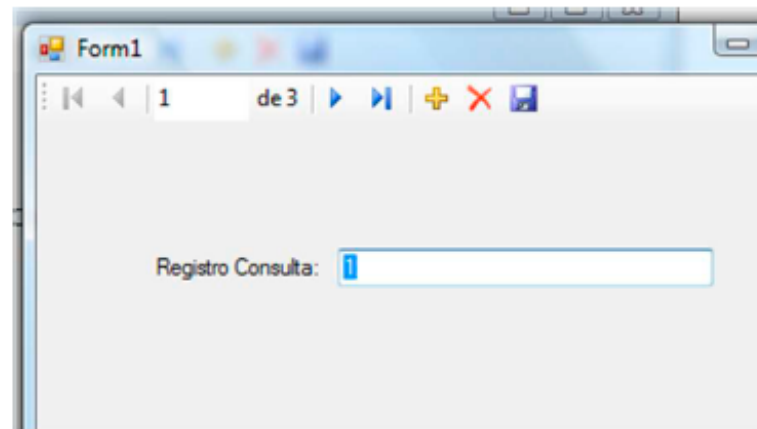
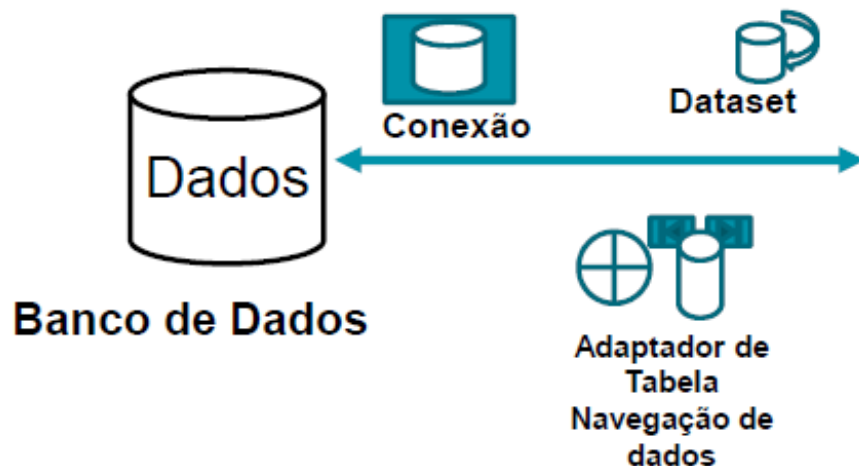
Terminologia de Banco de Dados

- Totalmente integrado ao .NET Framework;
- Como é nativo, sua API pode ser utilizada com as diversas linguagens que fazem parte do .NET Framework: Visual Basic, C#, entre outras;
- É sucessor do ADO, porém mais flexível;
- Possui um conjunto de Classes, Interfaces, Estruturas e Enumerações que gerem o acesso a dados dentro do .NET Framework;

Terminologia de Banco de Dados

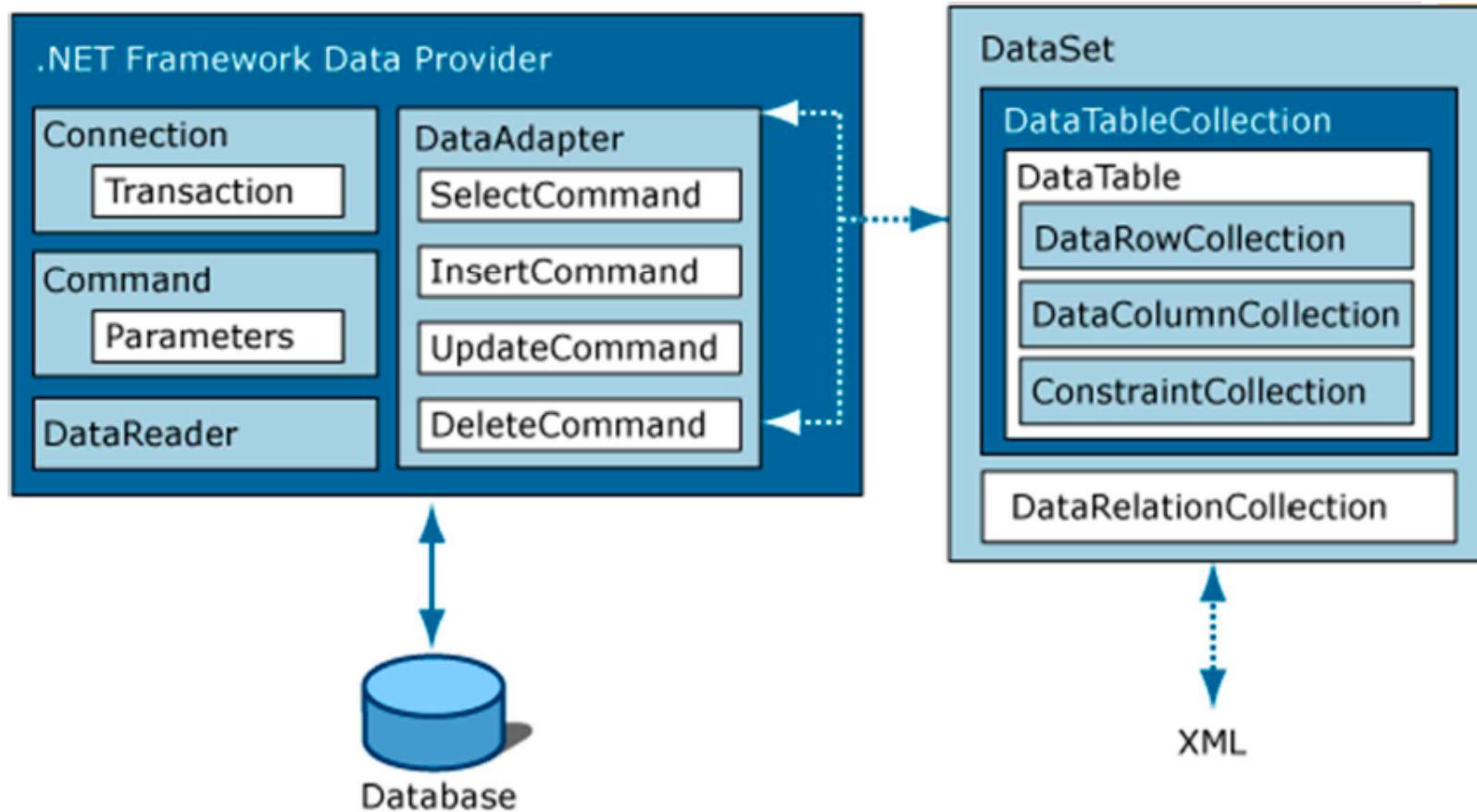
Camada de Dados

Camada da Aplicação



**Formulário de controles
Vinculados em tempo de
execução.**

Arquitetura do ADO.NET



Arquitetura do ADO.net - DataProvider

- São bibliotecas que possibilitam uma maneira comum de interagir com uma fonte específica de dados.
- Cada biblioteca possui um prefixo que indica qual provedor ela suporta.

DataProvider – Principais provedores

- São bibliotecas que possibilitam uma maneira comum de interagir com uma fonte específica de dados.
- Cada biblioteca possui um prefixo que indica qual provedor ela suporta.

Nome do Provedor	API prefixo	Descrição
ODBC Data Provider	Odbc	Fonte de dados com uma interface ODBC interface. Geralmente usada para banco de dados antigos
OleDb Data Provider	OleDb	Fonte de dados que expõe uma interface OleDb interface, ou seja: Access ou Excel
Oracle Data Provider	Oracle	Para banco de dados Oracle
SQL Data Provider	Sql	Para interação com o Microsoft SQL Server

Connection: Usado para se conectar a base de dados

Command: Usado para criar comandos dentro da fonte de dados

DataReader: Fornece um acesso conectado somente-leitura e somente para-frente a uma fonte de dados.

O **DataReader** efetua uma única passagem pelo conjunto de registros do banco da forma mais eficiente possível. São objetos que não podem ser diretamente instanciados. O **DataReader** é retornado pelo método **ExecuteReader** do objeto Command.

DataAdapter: Componentes encarregados de chamar os comandos de leitura, atualização, inclusão e exclusão de informações em uma fonte de dados pode ser usado para preencher um **DataReader** ou um **DataSet**.

O **DataAdapter** promove associação entre o provedor de dados .NET e o **DataSet**. É um mediador, que facilita a comunicação entre o banco de dados e o **DataSet**. O **DataAdapter** lê os dados do banco e os armazena dentro do **DataSet** através do método Fill. Ele também escreve as alterações efetuadas nos dados, a partir do **DataSet**, para que elas posteriormente sejam gravadas no banco através do método Update.

DataSet: Fornece uma representação relacional em memória de dados, sendo um conjunto completo de dados que incluem tabelas que contêm dados, restrições de dados e relacionamentos entre as tabelas. O acesso é desconectado.

É o principal componente da arquitetura ADO.NET, armazenando dados do banco no lado do cliente. Funciona como uma coleção de uma ou mais tabelas, que forma os **objetos DataTable**. Cada **DataTable** corresponde a uma única tabela ou visão. Assim, um objeto do tipo **DataSet** simula a estrutura da base de dados à qual se efetuou a consulta.

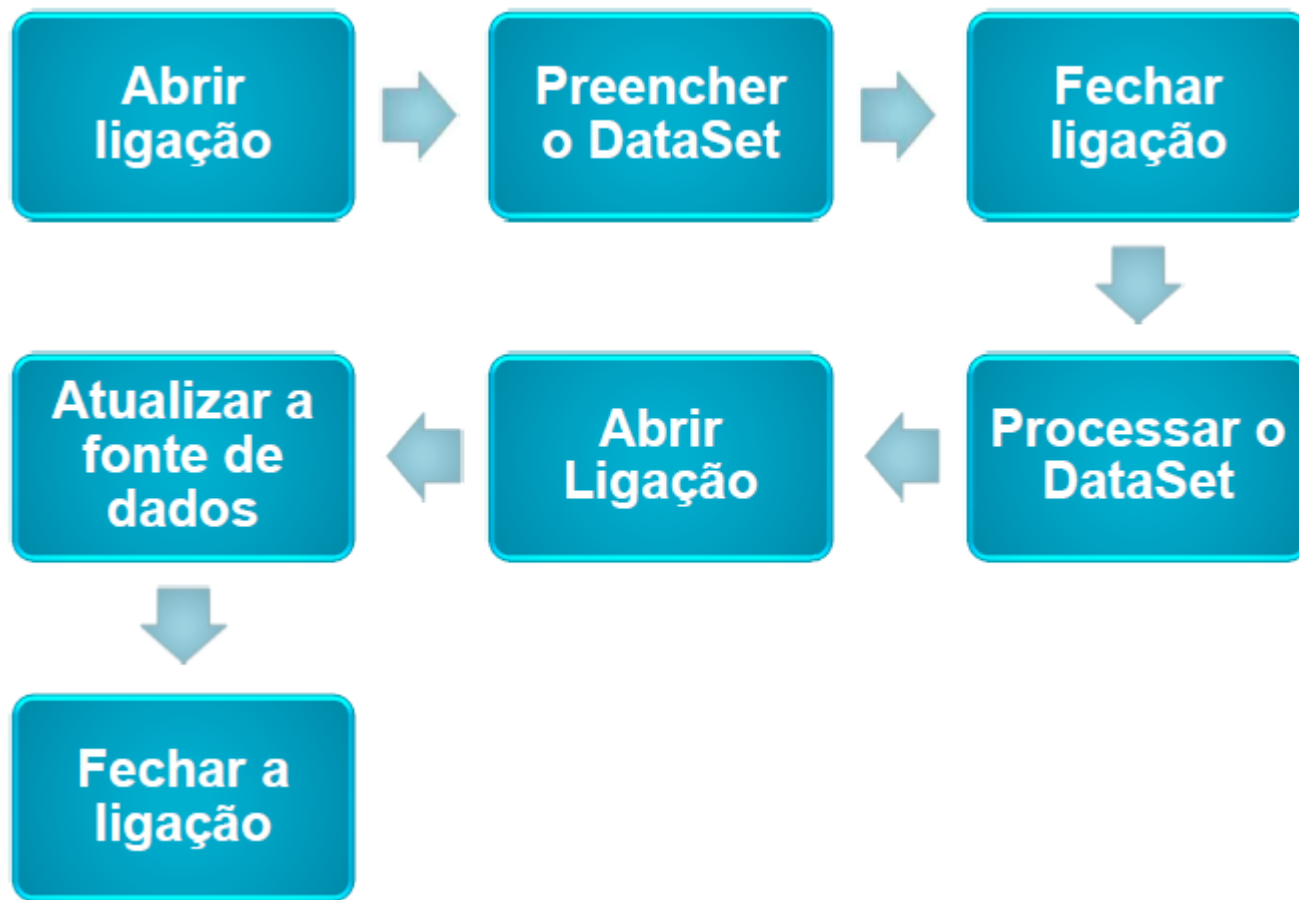
Arquitetura do ADO.net – DataSet

- Criada para manipular os dados independente da fonte;
- Pode receber fontes de dados de bancos de dados através do **DataAdapter**;
- Pode trabalhar diretamente com arquivos XML;
- Pode ainda trabalhar com fontes de dados diferentes dentro de um mesmo **DataSet**;
- Permite manipular os dados, efetuando leituras e alterações necessárias no modelo desconectado, sem a necessidade de manter a conexão aberta com o banco de dados.

DataRelationCollection: Permite a criação de relacionamentos entre as **DataTables** contidas dentro do **DataSet**.

Com tudo isso tem-se um modelo desconectado em memória algo muito parecido com que você possui em um banco de dados.

Modelo Desconectado

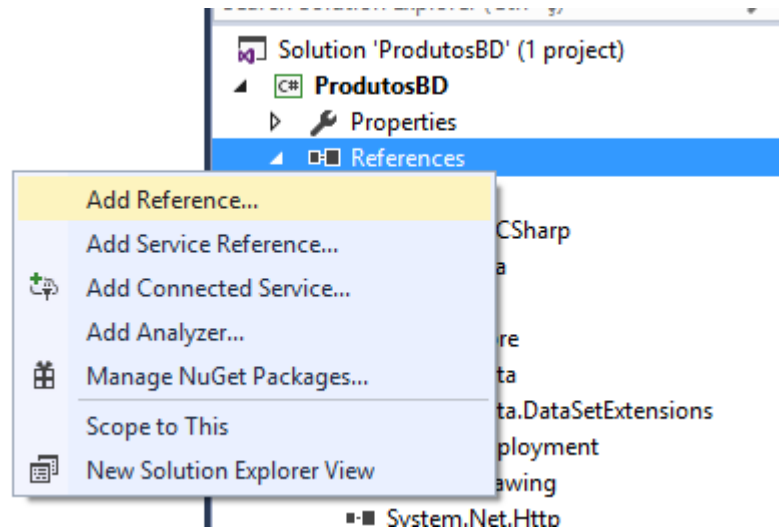




MySQL Connector/NET

Conector do MySQL para Visual Studio

<https://dev.mysql.com/downloads/connector/net/>

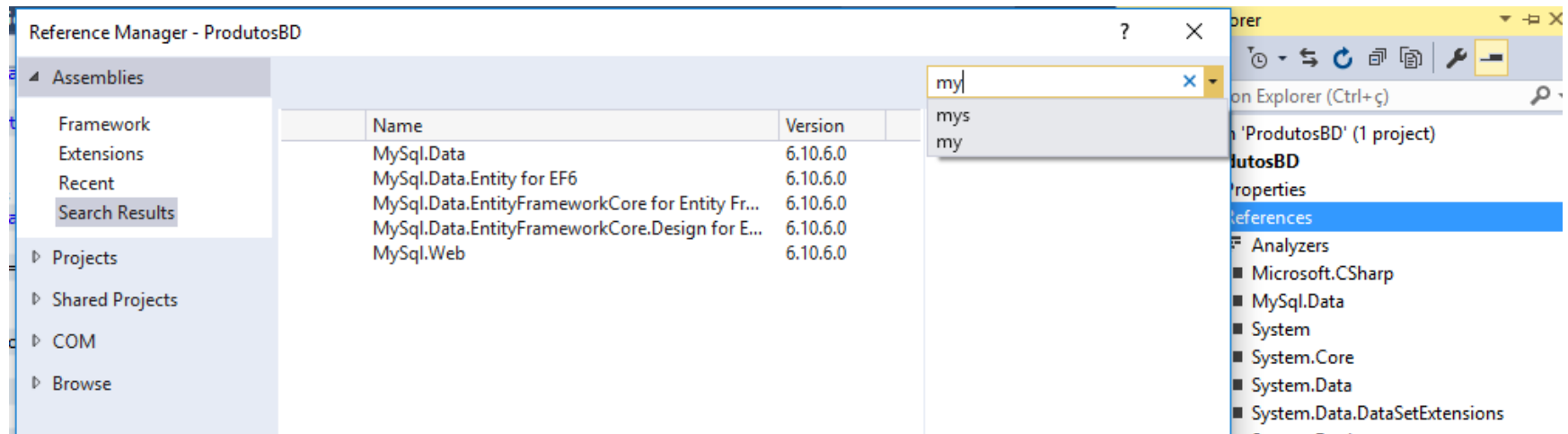




MySQL Connector/NET

Conector do MySQL para Visual Studio

<https://dev.mysql.com/downloads/connector/net/>

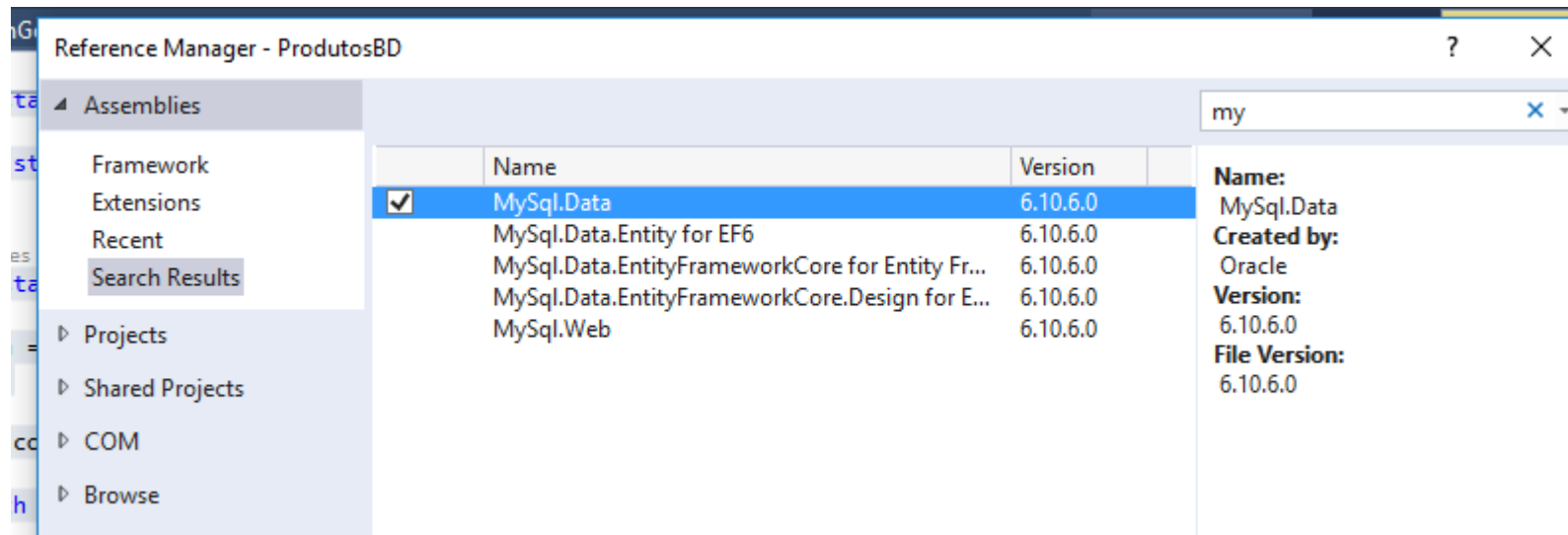




MySQL Connector/NET

Conector do MySQL para Visual Studio

<https://dev.mysql.com/downloads/connector/net/>

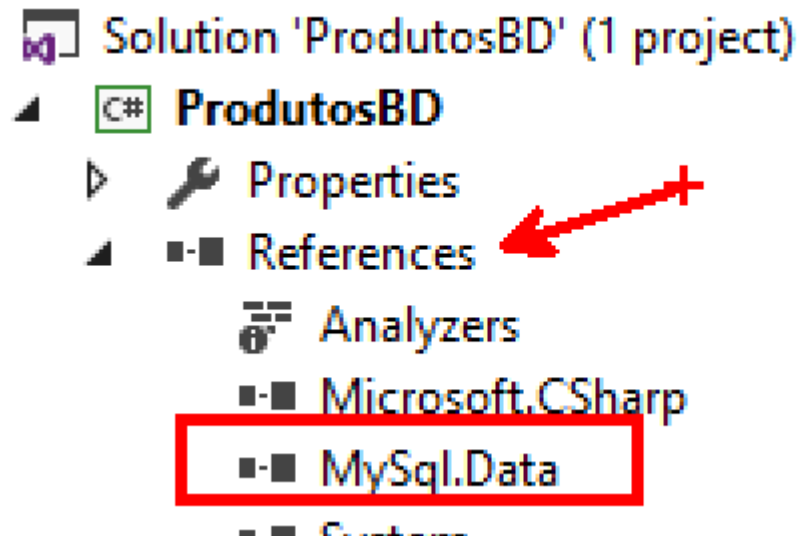




MySQL Connector/NET

Conector do MySQL para Visual Studio

<https://dev.mysql.com/downloads/connector/net/>





String de conexão MySql

Conexão - interna

String de conexão para acesso ao banco de dados local

```
MySqlConnection con = new MySqlConnection();  
con.ConnectionString = "Server=localhost;Port=3306;Database=dbti88;Uid=root;Pwd =''";  
con.Open();  
MessageBox.Show("Conexão Aberta!!!");  
con.Close();  
MessageBox.Show("Conexão Fechada!!!");
```



String de conexão MySql

Classe de conexão - externa

```
class Conexao
{
    private static string connString = "Server=localhost;Database=nomedobancomedados;Uid=usuario;Pwd='senha'";

    private static MySqlConnection conn = null;

    public static MySqlConnection obterConexao()
    {
        conn = new MySqlConnection(connString);
        try
        {
            conn.Open();
        }
        catch (MySqlException)
        {
            conn = null;
        }
        return conn;
    }

    public static void fecharConexao()
    {
        if (conn != null)
        {
            conn.Close();
        }
    }
}
```



Exemplo de Insert

```
MySQLCommand comm = new MySQLCommand();
    comm.CommandText = "insert into produtos (DESCRICAO,PVENDA,PCOMPRA,ESTOQUEATUAL)VALUES
(@DESCRICAO, @PVENDA, @PCOMPRA, @ESTOQUEATUAL); ";
    comm.CommandType = CommandType.Text;

    comm.Parameters.Clear();

    comm.Parameters.Add("@DESCRICAO", MySQLDbType.VarChar, 100).Value = txtDescr.Text;
    comm.Parameters.Add("@PVENDA", MySQLDbType.Decimal, 18).Value = txtPreVend.Text;
    comm.Parameters.Add("@PCOMPRA", MySQLDbType.Decimal, 18).Value = txtPreCompr.Text;
    comm.Parameters.Add("@ESTOQUEATUAL", MySQLDbType.Decimal, 18).Value = txtEstAtual.Text;

    comm.CommandType = CommandType.Text;
    comm.Connection = Conexao.obterConexao();

    int res = comm.ExecuteNonQuery();
    MessageBox.Show("Valores Inseridos com Sucesso!!! " + res);

    Conexao.fecharConexao();
    lblInformacao.Text = "";
```


Exemplo de Update

```
MySQLCommand comm = new MySQLCommand();
    comm.CommandText = "UPDATE produtos SET
descricao=@descricao,pvenda=@pvenda, pcompra = @pcompra,estoqueatual = @estoqueatual
WHERE codigo = " + txtCodigo.Text;
    comm.CommandType = CommandType.Text;
    comm.Connection = Conexao.obterConexao();
    comm.Parameters.Clear();
    comm.Parameters.Add("@descricao", MySqlDbType.VarChar, 50).Value =
txtDescr.Text;
    comm.Parameters.Add("@pvenda", MySqlDbType.Decimal, 18).Value =
txtPreVend.Text;
    comm.Parameters.Add("@pcompra", MySqlDbType.Decimal, 18).Value =
txtPreCompr.Text;
    comm.Parameters.Add("@estoqueatual", MySqlDbType.Decimal, 18).Value =
txtEstAtual.Text;

    int res = comm.ExecuteNonQuery();
    MessageBox.Show("Valores Alterados com Sucesso!!! " + res);

    Conexao.fecharConexao();
```

Exemplo de Delete

```
MySQLCommand comm = new MySQLCommand();
comm.CommandText = "DELETE FROM produtos WHERE codigo=@codigo";
comm.CommandType = CommandType.Text;
comm.Connection = Conexao.obterConexao();
comm.Parameters.Clear();
comm.Parameters.Add("@codigo", MySqlDbType.Int32).Value =
txtCodigo.Text;

vresp = MessageBox.Show("Deseja Realizar a Exclusão?", "Mensagem do
Sistema", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2);

if (vresp == DialogResult.Yes)
{
    int res = comm.ExecuteNonQuery();
    MessageBox.Show("Registro Excluido com Sucesso!!! " + res);
    //cancelarCampos();
}
else if (vresp == DialogResult.No)
{
    //cancelarCampos();
}

Conexao.fecharConexao();
```



Exemplo de Select

```
MySqlCommand comm = new MySqlCommand();  
    comm.CommandText = "SELECT CODIGO + 1 FROM produtos order by  
codigo desc;";  
    comm.CommandType = CommandType.Text;  
    comm.Connection = Conexao.obterConexao();  
  
MySqlDataReader DR;  
DR = comm.ExecuteReader();  
DR.Read();  
  
txtCodigo.Text = Convert.ToString(DR.GetInt32(0));  
  
ativarCampos();  
botaoqueestouclicando = 1;  
btnNovo.Enabled = false;  
btnAlterar.Enabled = false;  
btnExcluir.Enabled = false;  
lblInformacao.Text = "Você clicou no botão novo!!!";
```

Exemplo de Select

```
MySqlCommand comm = new MySqlCommand();
    comm.CommandText = "select * from produtos where descricao
like '%" + txtDescricao.Text + "%'";
    comm.CommandType = CommandType.Text;
    comm.Connection = Conexao.obterConexao();

    MySqlDataReader DR;
    DR = comm.ExecuteReader();


    ltbProd.Items.Clear();

    while (DR.Read())
    {
        ltbProd.Items.Add(DR.GetInt32(0) + " - " +
DR.GetString(1));
    }

    //txtDescricao.Text = "";
    //txtDescricao.Focus();
```


Exemplo de Select

```
MySqlCommand comm = new MySqlCommand();  
    comm.CommandText = "select * from produtos where codigo = "  
+ txtDescricao.Text + ";";  
    comm.CommandType = CommandType.Text;  
    comm.Connection = Conexao.obterConexao();  
  
MySqlDataReader DR;  
DR = comm.ExecuteReader();  
DR.Read();  
  
txtCodigo.Text = Convert.ToString(DR.GetInt32(0));  
txtDescr.Text = DR.GetString(1);  
txtPreVend.Text = Convert.ToString(DR.GetDecimal(2));  
txtPreCompr.Text = Convert.ToString(DR.GetDecimal(3));  
txtEstAtual.Text = Convert.ToString(DR.GetDecimal(4));  
  
lrbProd.Items.Add(DR.GetInt32(0) + " - " + DR.GetString(1));  
  
Conexao.fecharConexao();
```

 Gerenciar Produtos


Pesquisa por:


☐ Código ☐ Nome



Pesquisar


Descrição


Dados do Produto


Novo


Alterar


Excluir


Gravar


Cancelar

Código

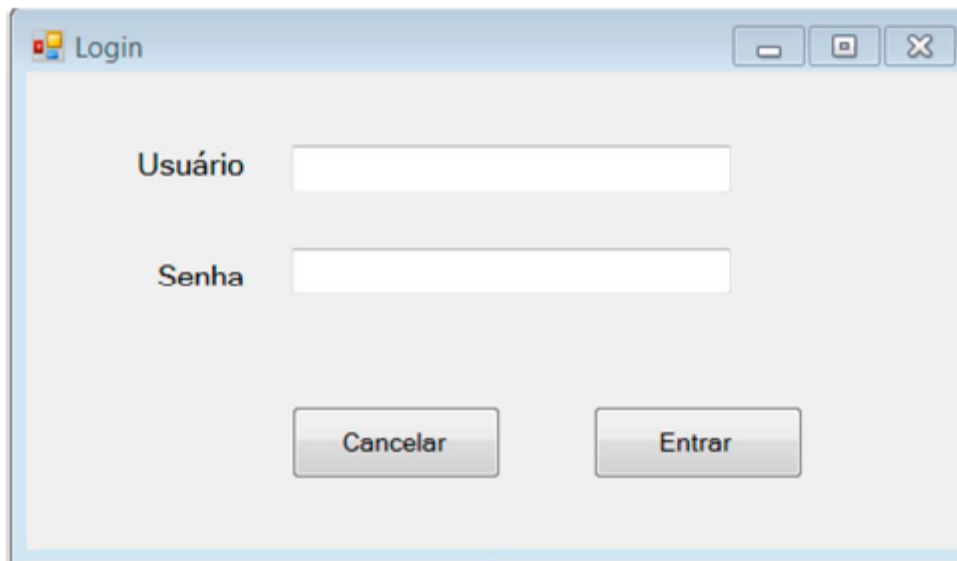
Descrição

Preço de Venda

Preço de Compra

Estoque Atual na Loja

Modelo de Sistema - Login

A mockup of a login window titled "Login". It features two input fields: "Usuário" (Username) and "Senha" (Password). Below the fields are two buttons: "Cancelar" (Cancel) and "Entrar" (Enter). The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Usuário

Senha



Modelo de Sistema - Login

```
using System.Data.SqlClient;  
namespace SistemaComBancoDeDados  
{  
    public partial class Login : Form  
    {  
        public Login()  
        {  
            InitializeComponent();  
        }  
        private bool Logado = false;
```




Modelo de Sistema - Login

```
bool AcessoSistema()
{
    bool resultado = false;
    conexao.Open();
    MySqlCommand comm = new MySqlCommand();
    comm.CommandText = "SELECT * FROM tbusuarios WHERE username='" + textBox1.Text + "'  
AND senha='" + textBox2.Text + "'";
    comm.CommandType = CommandType.Text;
    comm.Connection = conexao;
    MySqlDataReader DR;
    DR = comm.ExecuteReader();
    resultado = DR.HasRows;
    MessageBox.Show("O resultado do HasRows = "+ resultado);
    conexao.Close();
    return resultado;
}
```



Modelo de Sistema - Login

```
private void button2_Click(object sender, EventArgs e)
{
    bool result =AcessoSistema();
    Logado = result;
    if (result)
    {
        MessageBox.Show("Seja bem vindo!");
        Produtos abrirLogin = new Produtos();
        abrirLogin.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("Usuário ou senha incorreto!");
    }
}
```



Modelo de Sistema - Login

```
private void Login_FormClosed(object sender, FormClosedEventArgs e)
{
    if (Logado){
        this.Close();
    }
    else
    {
        Application.Exit();
    }
}

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