

# Desenvolvimento de Sistemas

Professor Everton Sette

# 06 Integração com Banco de Dados





**Nome do projeto deverá ser o  
RM do Aluno ou da dupla**

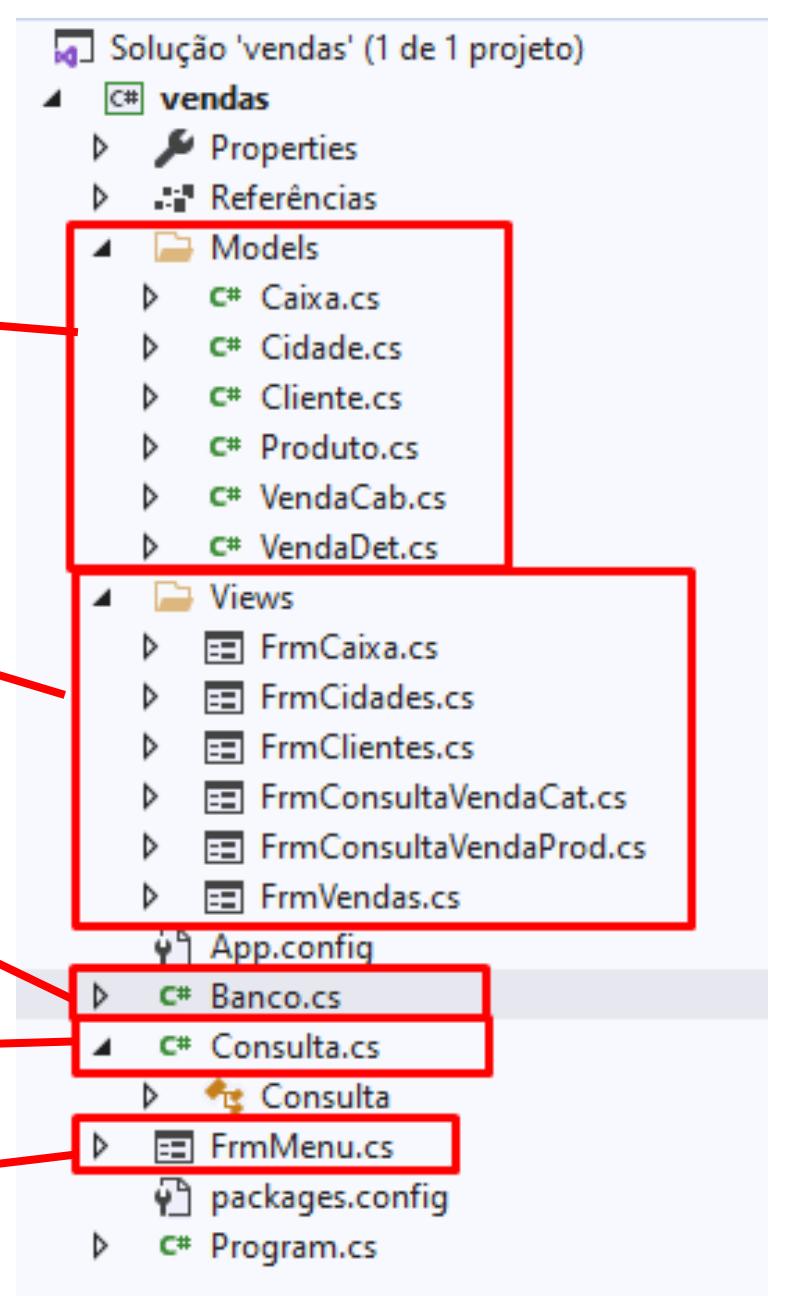
Pasta contendo todas as classes de modelagem (atributos, métodos)

Pasta contendo todos os formulários do sistema

Classe de conexão com o banco de dados

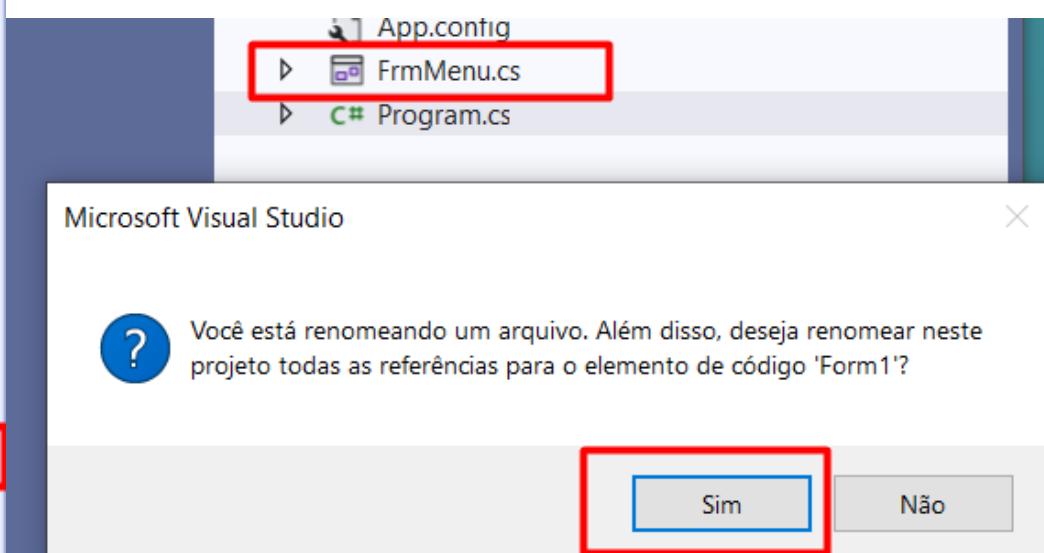
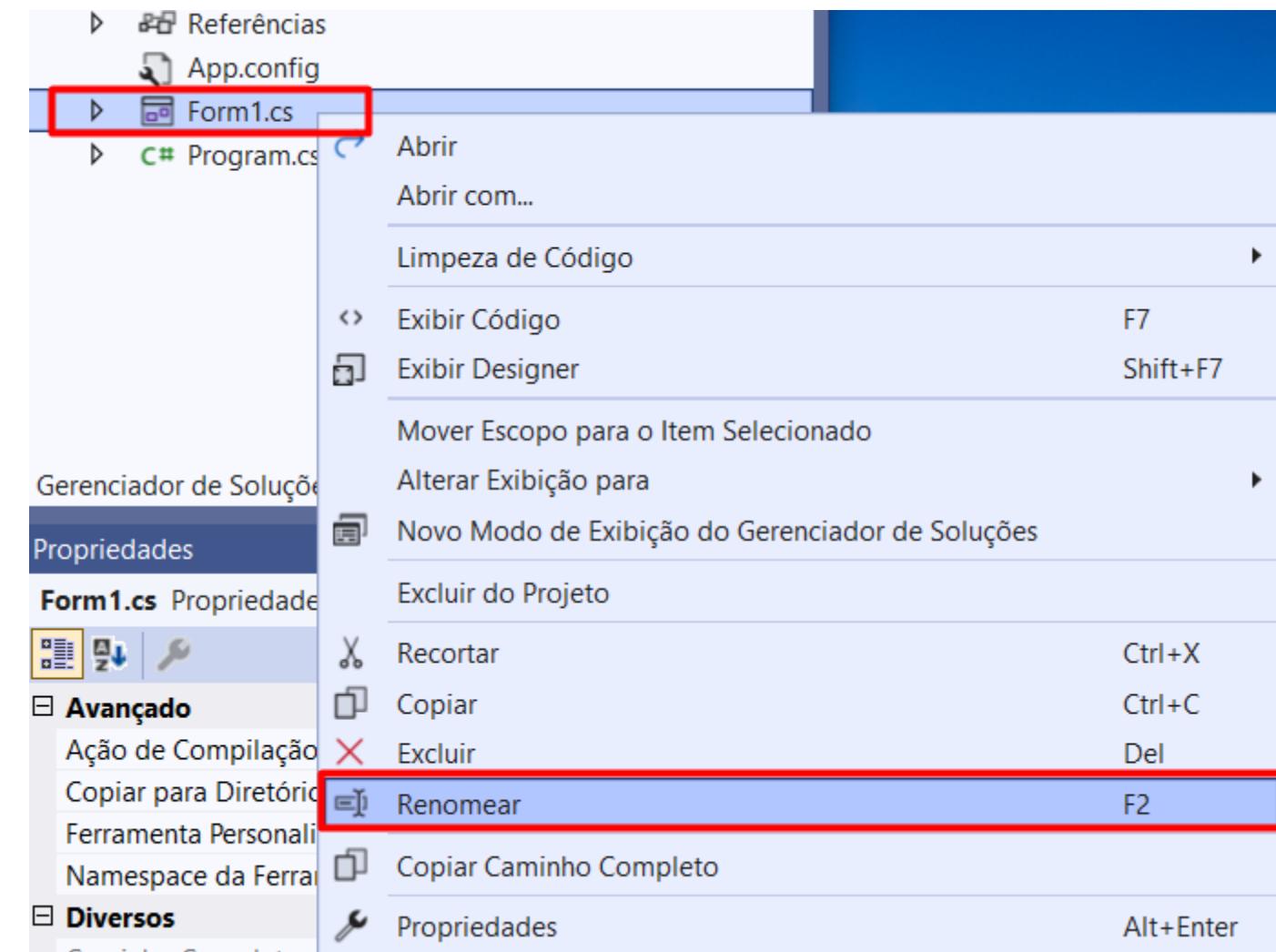
Classe contendo todas as consultas para relatórios e gráficos

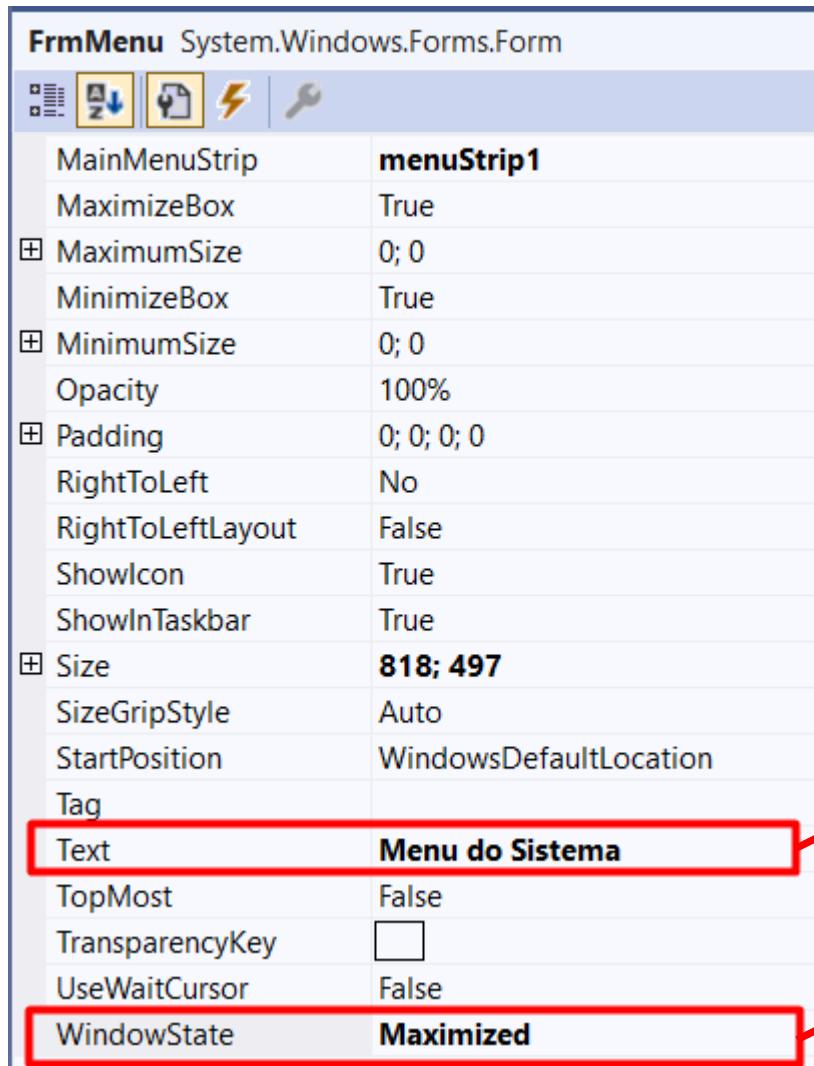
Menu principal do sistema



# Renomeando um Formulário

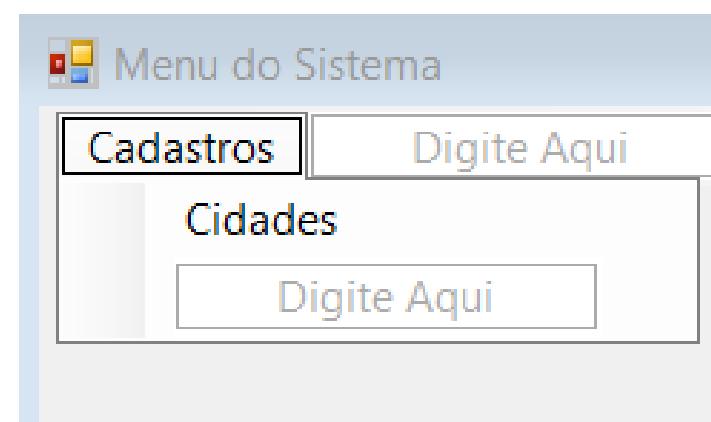
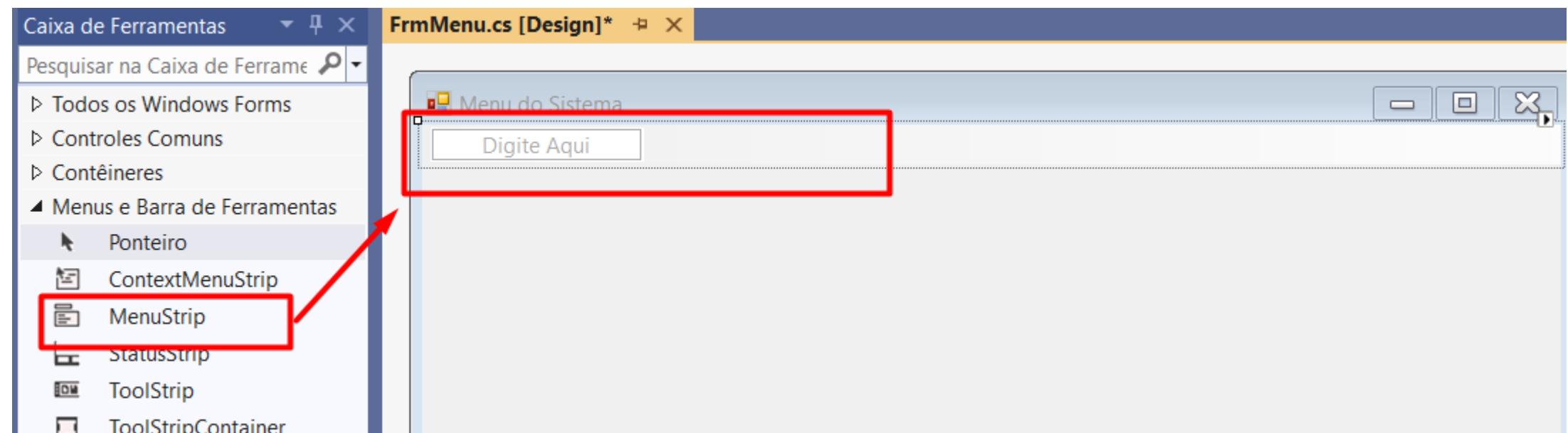
05

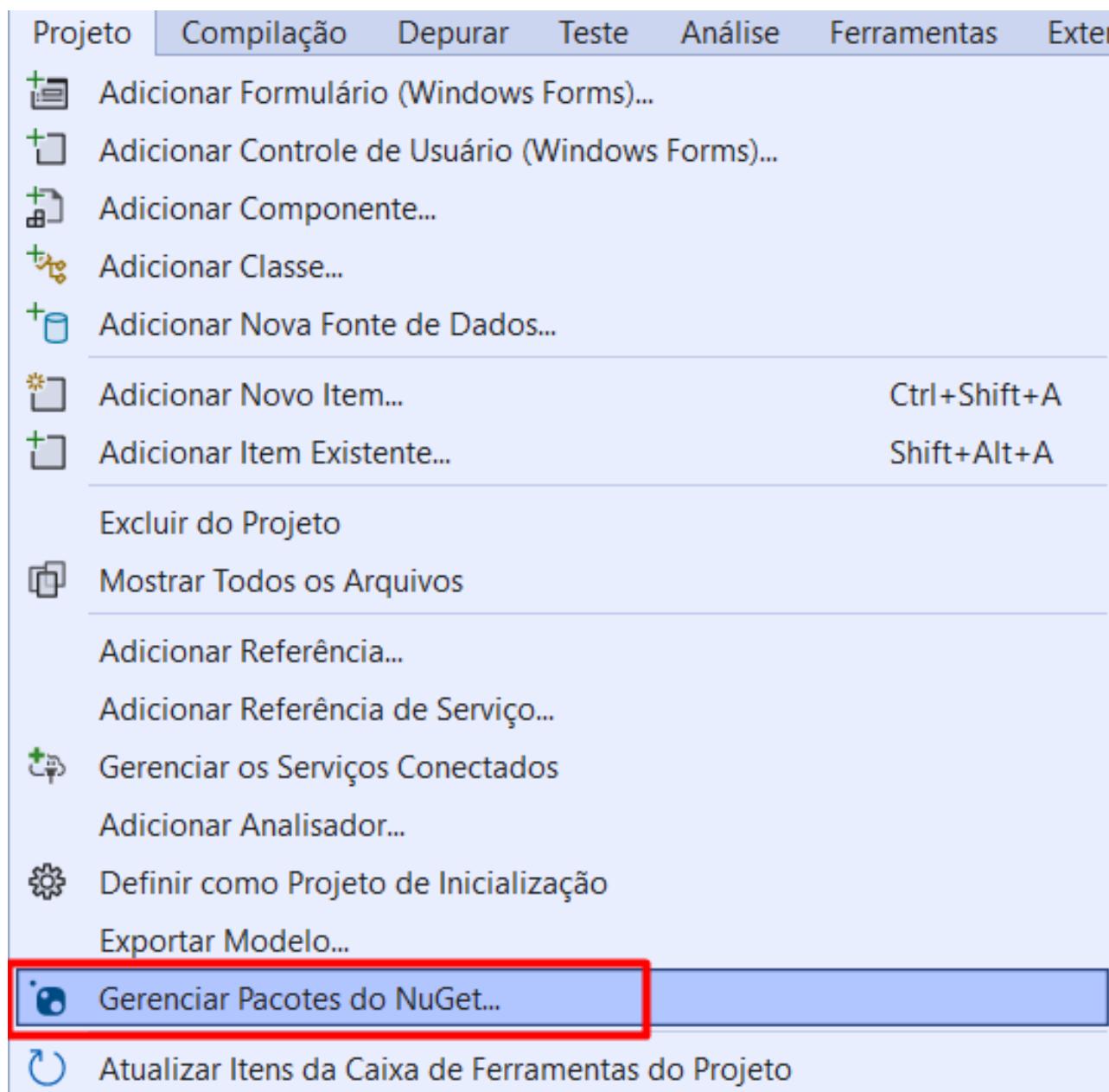




Altera o título do Formulário

Inicializa o Form Maximizado





# Importando a biblioteca do MySQL

09

NuGet: vendas\* ▾ X FrmMenu.cs [Design]\*

Procurar Instalado Atualizações

mysql x 🔍 Incluir pré-lançamento

Gerenciador de Pacotes do NuGet: vendas

Origem do pacote: nuget.org ⚙

**MySQL.Data** por Oracle, 37,8M downloads  
MySql.Data.MySqlClient .Net Core Class Library 8.0.28

**Pomelo.EntityFrameworkCore.MySql** por Laurents Meyer, Caleb Lloyd, Yuko Zh 6.0.1  
Pomelo's MySQL database provider for Entity Framework Core.

Ao clicar em "Aceitar", você concorda com os termos de licença para os pacotes listados acima. Se não concordar com os termos de licença, clique em "Recusar".

Aceitar Recusar

MySQL.org

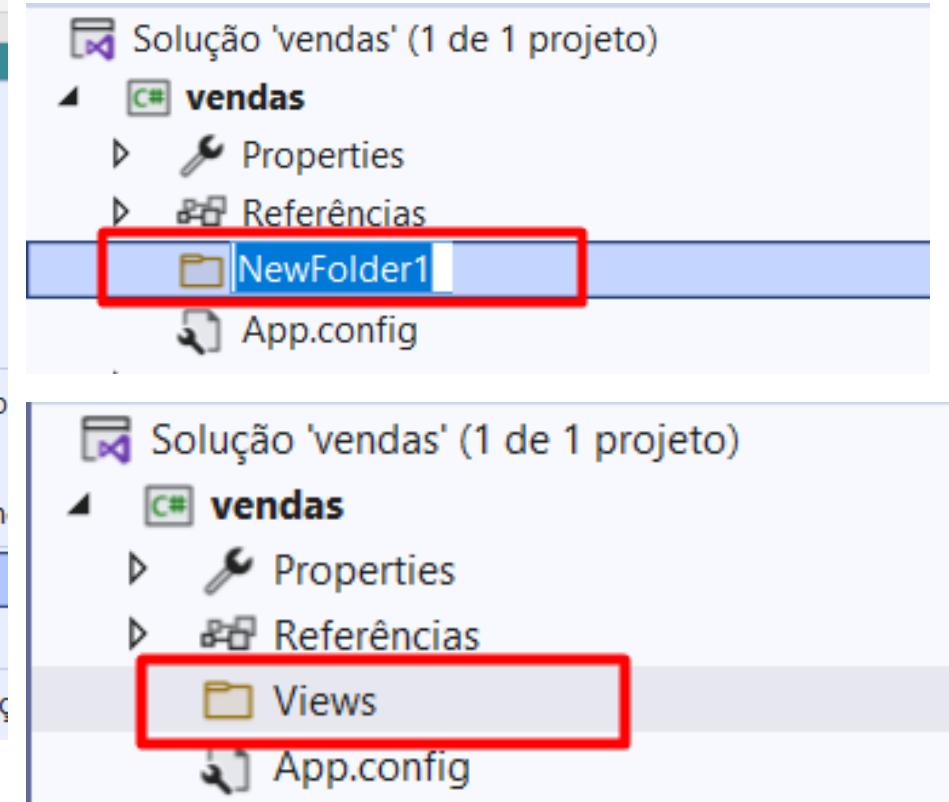
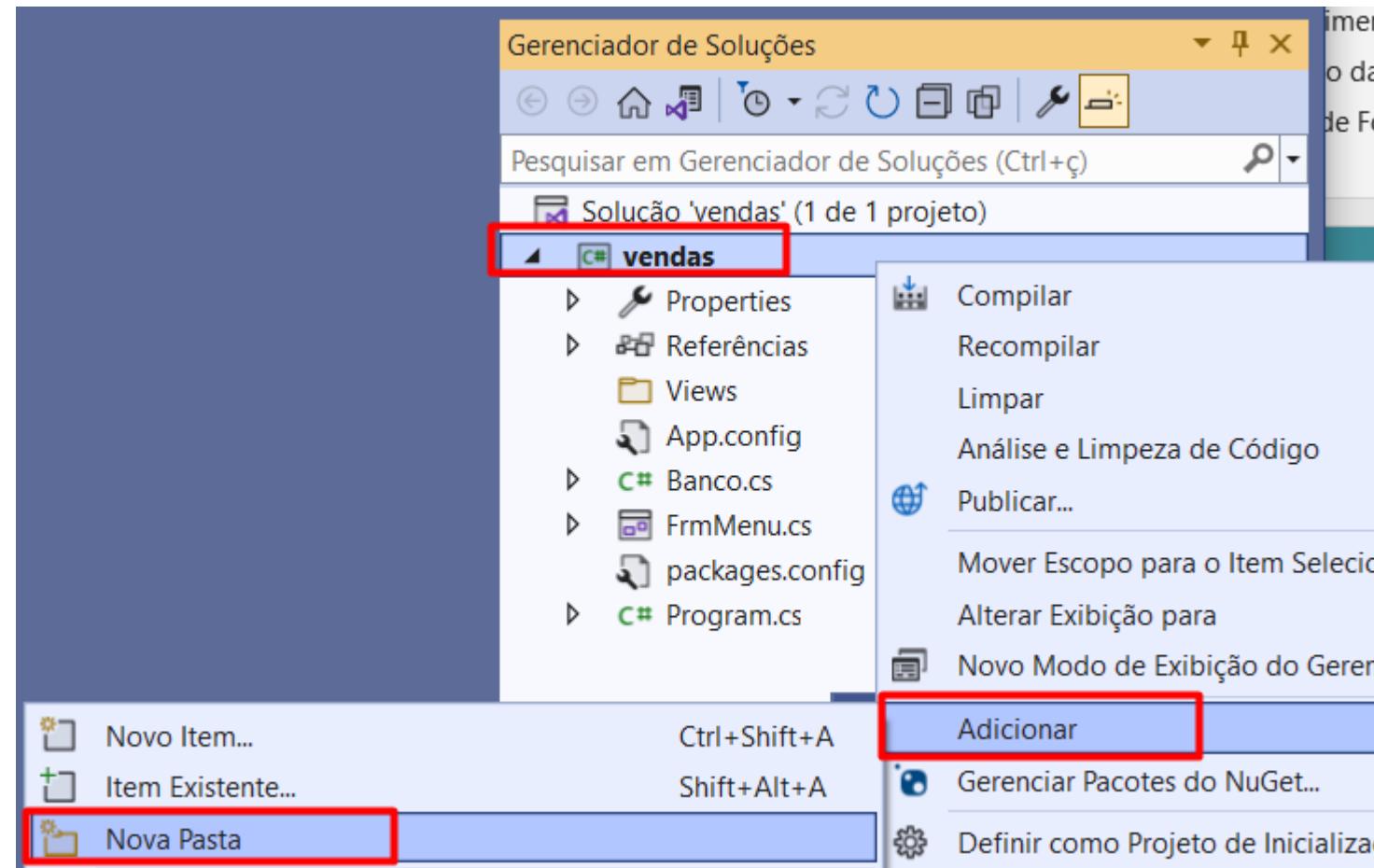
MySQL.Data

Versão: Estável mais recente & Instalar

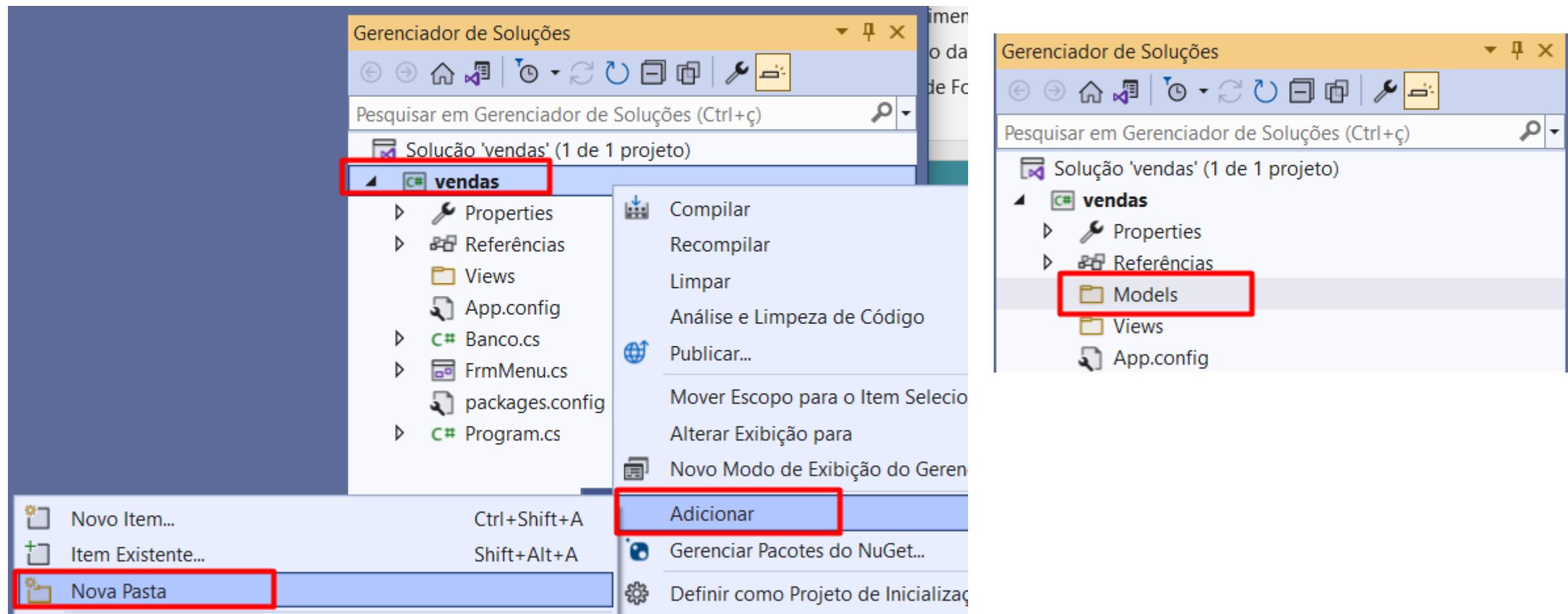
Opções

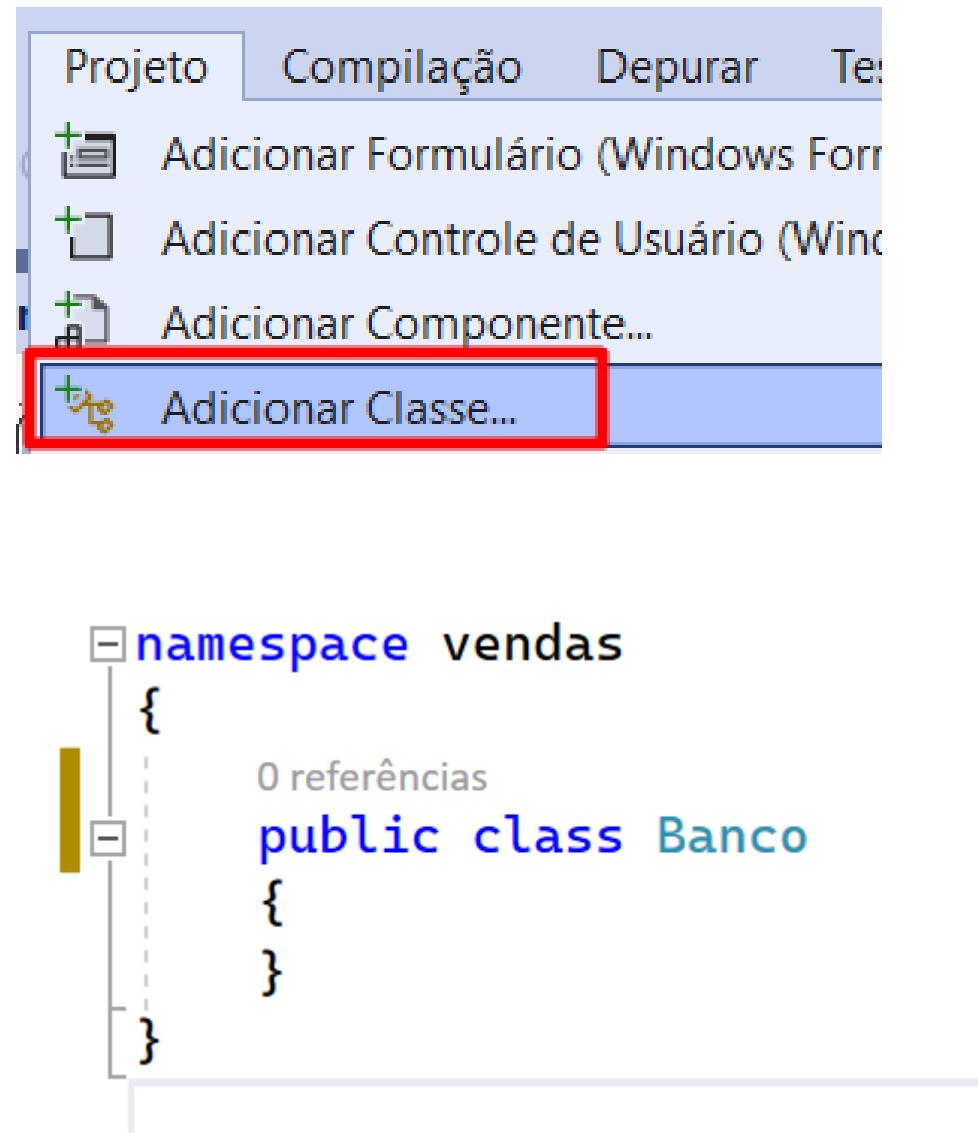
# Adicionando uma pasta para nossos Forms

10

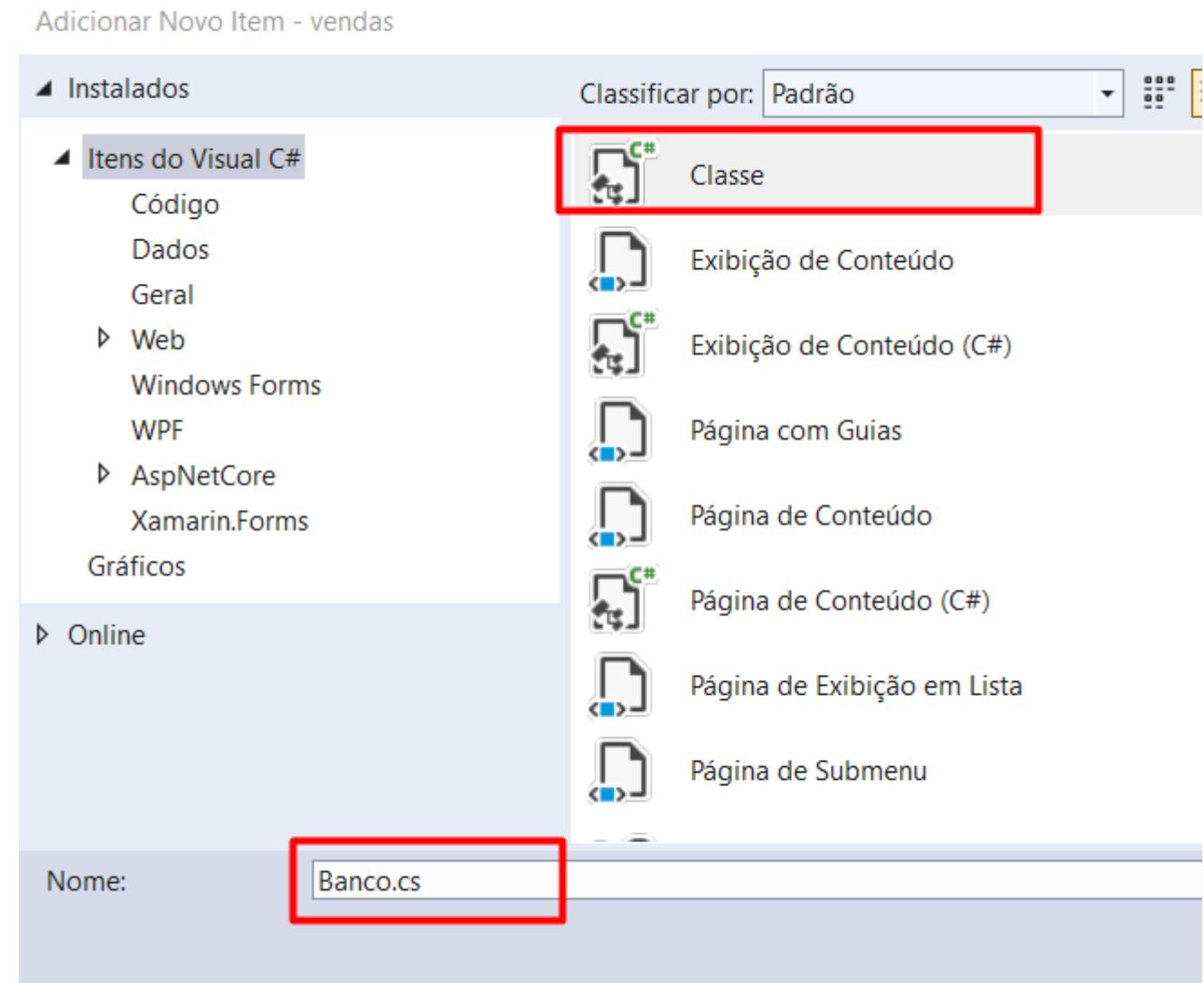


# Adicionar pasta para classes de modelagem





```
namespace vendas
{
    0 referências
    public class Banco
    {
    }
}
```

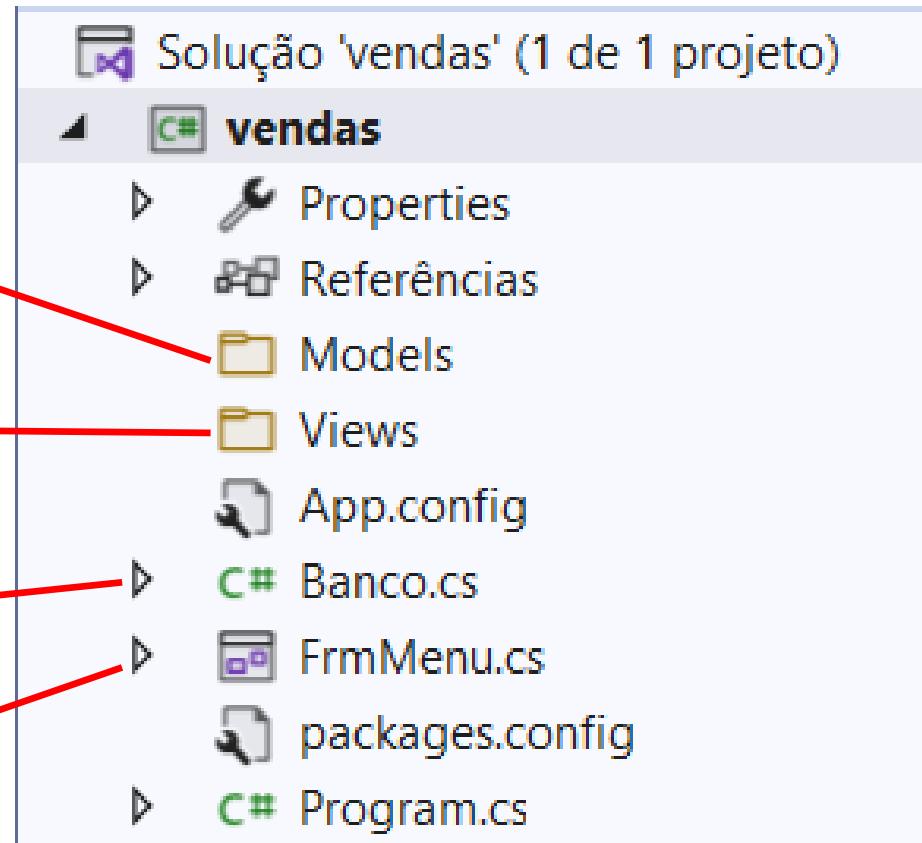


Pasta onde ficará nossas classes de modelagem

Pasta onde ficará nossos Forms

Classe de conexão com o banco

Menu Principal



```
using System;
using System.Data;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace vendas
{
    99+ referências
    public class Banco
    {
        // Criando as variáveis públicas para conexão e consulta serão usadas em todo o projeto
        // Connection responsável pela conexão com o MySQL
        public static MySqlConnection Conexao;
        // Command responsável pelas instruções SQL a serem executadas
        public static MySqlCommand Comando;
        // Adapter responsável por inserir dados em um dataTable
        public static MySqlDataAdapter Adaptador;
        // DataTable responsável por ligar o banco em controles com a propriedade DataSource
        public static DataTable datTabela;
    }
}
```

Essa classe será responsável para estabelecer conexão com o banco de dados

1 referência

```
public static void AbrirConexao()
{
    try
    {
        // Estabelece os parâmetros para a conexão com o banco
        Conexao = new MySqlConnection("server=localhost;port=3307;uid=root;pwd=etecjau");

        // Abre a conexão com o banco de dados
        Conexao.Open();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Método (Função) responsável por estabelecer a conexão com o banco de dados.

1 referência

```
public static void FecharConexao()
{
    try
    {
        // Fecha a conexão com o banco de dados
        Conexao.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Método (Função) responsável por encerrar a conexão com o banco de dados.

```
1 referência
public static void CriarBanco()
{
    try
    {
        //Chama a função para abertura de conexão com o banco
        AbrirConexao();

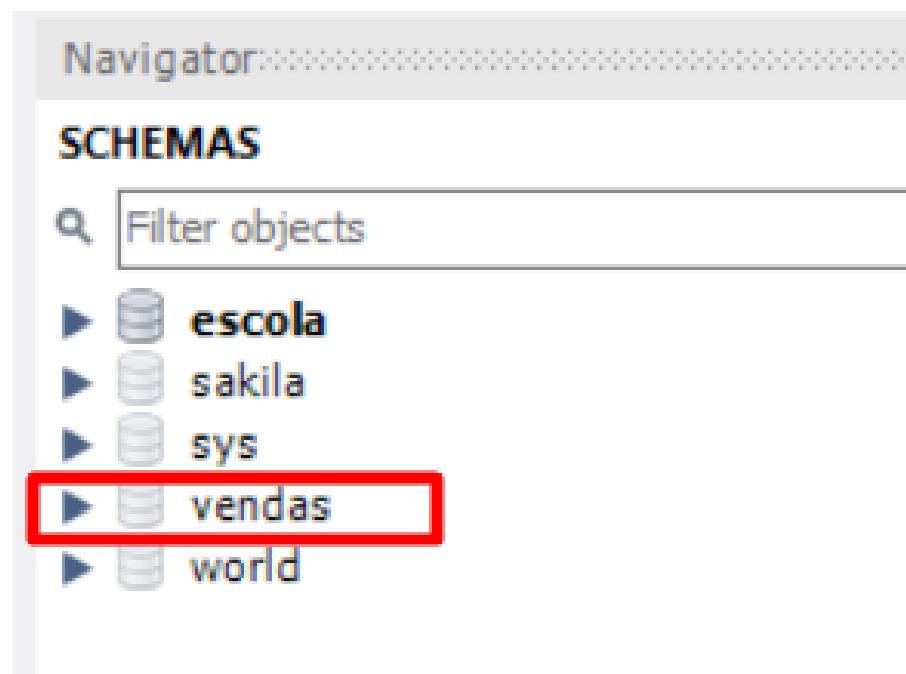
        // Informa a instrução SQL
        Comando = new MySqlCommand("CREATE DATABASE IF NOT EXISTS vendas; USE vendas", Conexao);
        // Executa a Query no MySQL (Raio do Workbench)
        Comando.ExecuteNonQuery();

        // Chama a função para fechar a conexão com o banco
        FecharConexao();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Método (Função) responsável por criar nosso banco e estruturas de tabelas

1 referência

```
private void FrmMenu_Load(object sender, EventArgs e)
{
    Banco.CriarBanco();
}
```





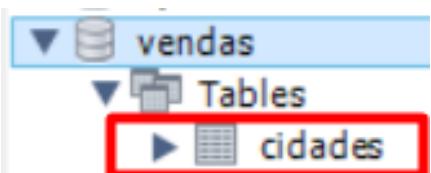
```
1 referência
public static void CriarBanco()
{
    try
    {
        // Chama a função para abertura de conexão com o banco
        AbrirConexao();

        // Informa a instrução SQL
        Comando = new MySqlCommand("CREATE DATABASE IF NOT EXISTS vendas; USE vendas", Conexao);
        // Executa a Query no MySQL (Raio do Workbench)
        Comando.ExecuteNonQuery();

        Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Cidades " +
            "(id integer auto_increment primary key, " +
            "nome char(40), " +
            "uf char(02))", Conexao);
        Comando.ExecuteNonQuery();

        // Chama a função para o fechamento de conexão com o banco
        FecharConexao();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

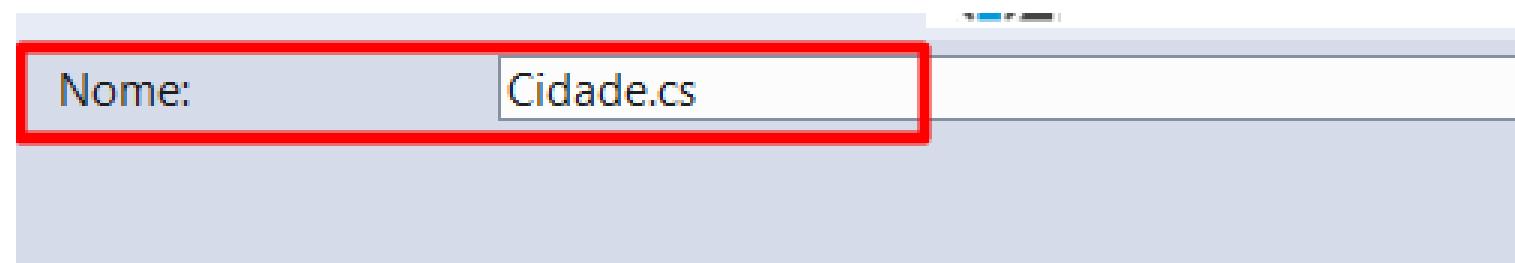
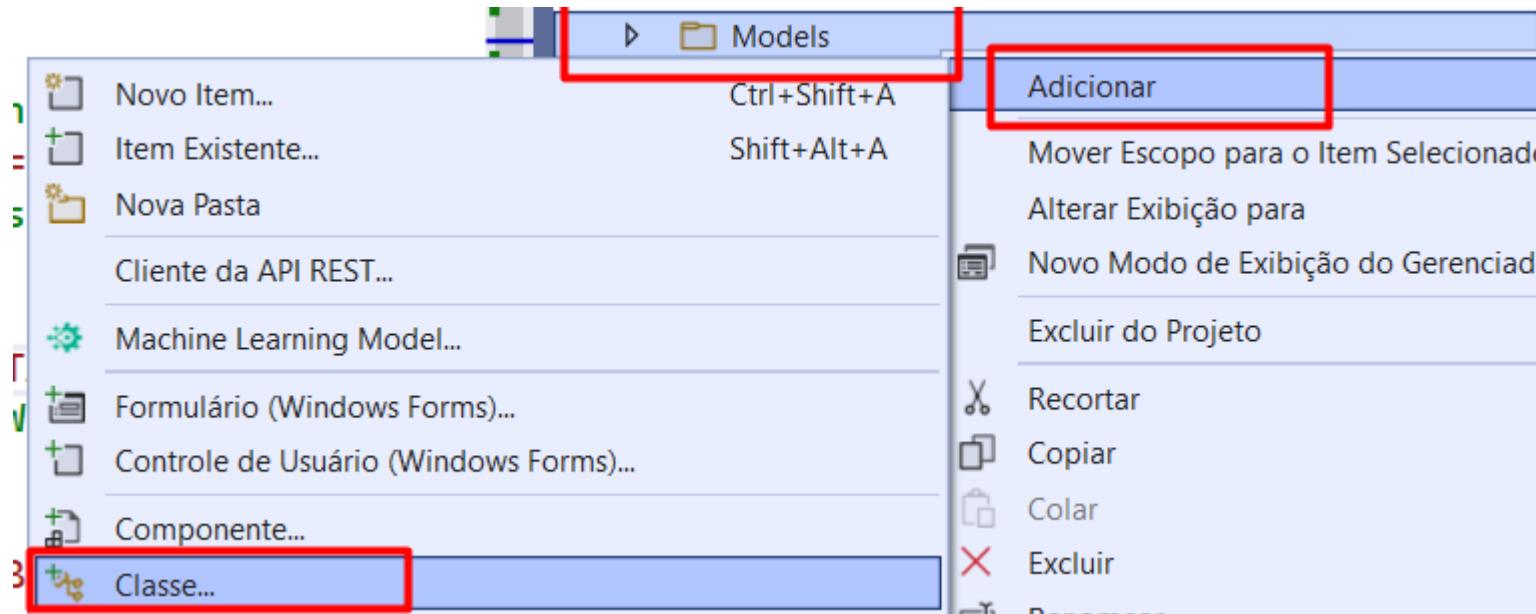
Devemos modificar nossa função onde criamos o banco, inserindo a instrução para criar nossa tabela de cidades.



A finalidade da modelagem de classes é descrever objetos. Um objeto é um conceito, abstração ou coisa com identidade que possui significado para a aplicação. Os objetos normalmente aparecem como nomes próprios ou referências específicas nas descrições de problemas e discussões com os usuários.

Um objeto é uma instância – ou ocorrência – de uma classe. Uma classe descreve um grupo de objetos com as mesmas propriedades (atributos), comportamentos (métodos).

Devemos criar nossa classe de modelagem de cidades dentro da pasta Models do nosso projeto



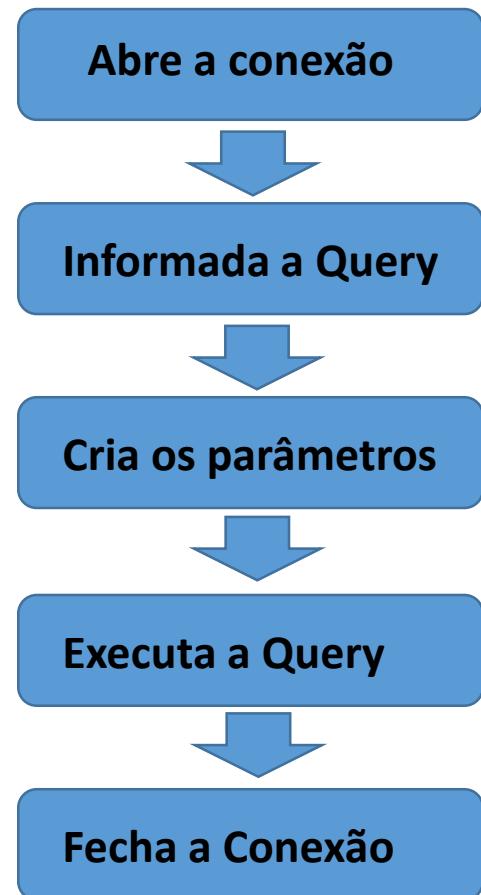
```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Windows.Forms;

namespace vendas.Models
{
    7 referências
    public class Cidade
    {
        4 referências
        public int id { get; set; }
        6 referências
        public string nome { get; set; }
        4 referências
        public string uf { get; set; }
    }
}
```

Primeiramente devemos criar nossos atributos na classe de modelagem, referenciando cada campo da tabela do banco de dados.

```
public void Incluir()
{
    try
    {
        // Abre a conexão com o banco
        Banco.AbrirConexao();
        // Alimenta o método Command com a instrução desejada e indica a conexão utilizada
        Banco.Comando = new MySqlCommand("INSERT INTO cidades (nome, uf) VALUES (@nome, @uf)", Banco.Conexao);
        // Cria os parâmetros utilizados na instrução SQL com seu respectivo conteúdo
        Banco.Comando.Parameters.AddWithValue("@nome", nome); // Parâmetro String
        Banco.Comando.Parameters.AddWithValue("@uf", uf);
        // Executa o Comando, no MYSQL, tem a função do Raio do Workbench
        Banco.Comando.ExecuteNonQuery();
        // Fecha a conexão
        Banco.FecharConexao();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

**Função na classe de modelagem responsável pela inclusão das cidades no banco de dados.**

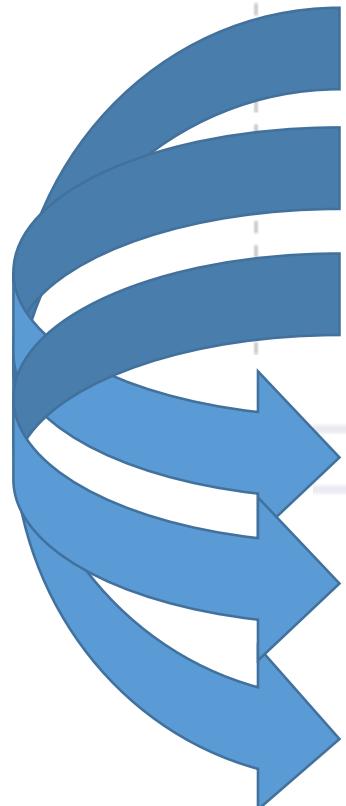


```
public class Cidade
{
    public int id { get; set; }
    public string nome { get; set; }
    public string uf { get; set; }
}
```

4 referências

6 referências

4 referências



```
Banco.Comando.Parameters.AddWithValue("@nome", nome);
```

```
Banco.Comando.Parameters.AddWithValue("@uf", uf);
```

```
Banco.Comando.Parameters.AddWithValue("@id", id);
```



**Slide apenas informativo**

```
public void Alterar()
{
    try
    {
        // Abre a conexão com o banco
        Banco.AbrirConexao();
        // Alimenta o método Command com a instrução desejada e indica a conexão utilizada
        Banco.Comando = new MySqlCommand("Update cidades set nome = @nome, uf = @uf where id = @id", Banco.Conexao);
        // Cria os parâmetros utilizados na instrução SQL com seu respectivo conteúdo
        Banco.Comando.Parameters.AddWithValue("@nome", nome); // Parâmetro String
        Banco.Comando.Parameters.AddWithValue("@uf", uf);
        Banco.Comando.Parameters.AddWithValue("@id", id);
        // Executa o Comando, no MYSQL, tem a função do Raio do Workbench
        Banco.Comando.ExecuteNonQuery();
        // Fecha a conexão
        Banco.FecharConexao();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

**Função na classe de modelagem responsável pela alteração das cidades no banco de dados.**

```
public void Excluir()
{
    try
    {
        // Abre a conexão com o banco
        Banco.AbrirConexao();
        // Alimenta o método Command com a instrução desejada e indica a conexão utilizada
        Banco.Comando = new MySqlCommand("delete from cidades where id = @id", Banco.Conexao);
        // Cria os parâmetros utilizados na instrução SQL com seu respectivo conteúdo
        Banco.Comando.Parameters.AddWithValue("@id", id);
        // Executa o Comando, no MYSQL, tem a função do Raio do Workbench
        Banco.Comando.ExecuteNonQuery();
        // Fecha a conexão
        Banco.FecharConexao();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

**Função na classe de modelagem responsável pela exclusão das cidades no banco de dados.**

```
public DataTable Consultar()
{
    try
    {
        Banco.AbrirConexao();
        Banco.Comando = new MySqlCommand("SELECT * FROM Cidades where nome like @Nome " +
                                         "order by nome", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@Nome", nome + "%");
        Banco.Adaptador = new MySqlDataAdapter(Banco.Comando);
        Banco.datTabela = new DataTable();
        Banco.Adaptador.Fill(Banco.datTabela);
        Banco.FecharConexao();
        return Banco.datTabela;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
}
```

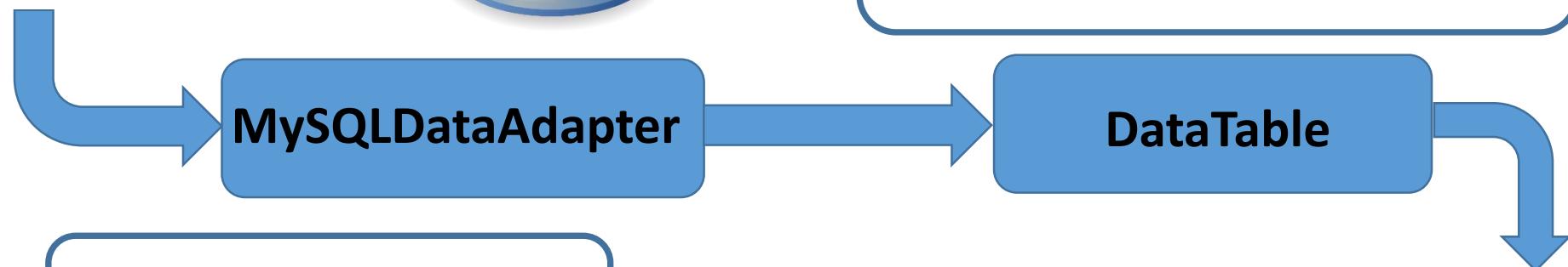
Função criada na classe de modelagem para consultar os registros da tabela de cidades

|   | id   | nome   | uf   |
|---|------|--------|------|
| ▶ | 1    | JAU    | SP   |
| ● | 2    | BARIRI | SP   |
| * | NULL | NULL   | NULL |



```
Adaptador.Fill(datTabela = new DataTable());
```

Fonte de dados que permite a população de consultas em controles com propriedade DataSource



Traz a Consulta SQL em forma de tabela para a memória

```
Adaptador = new MySqlDataAdapter(comando);
```



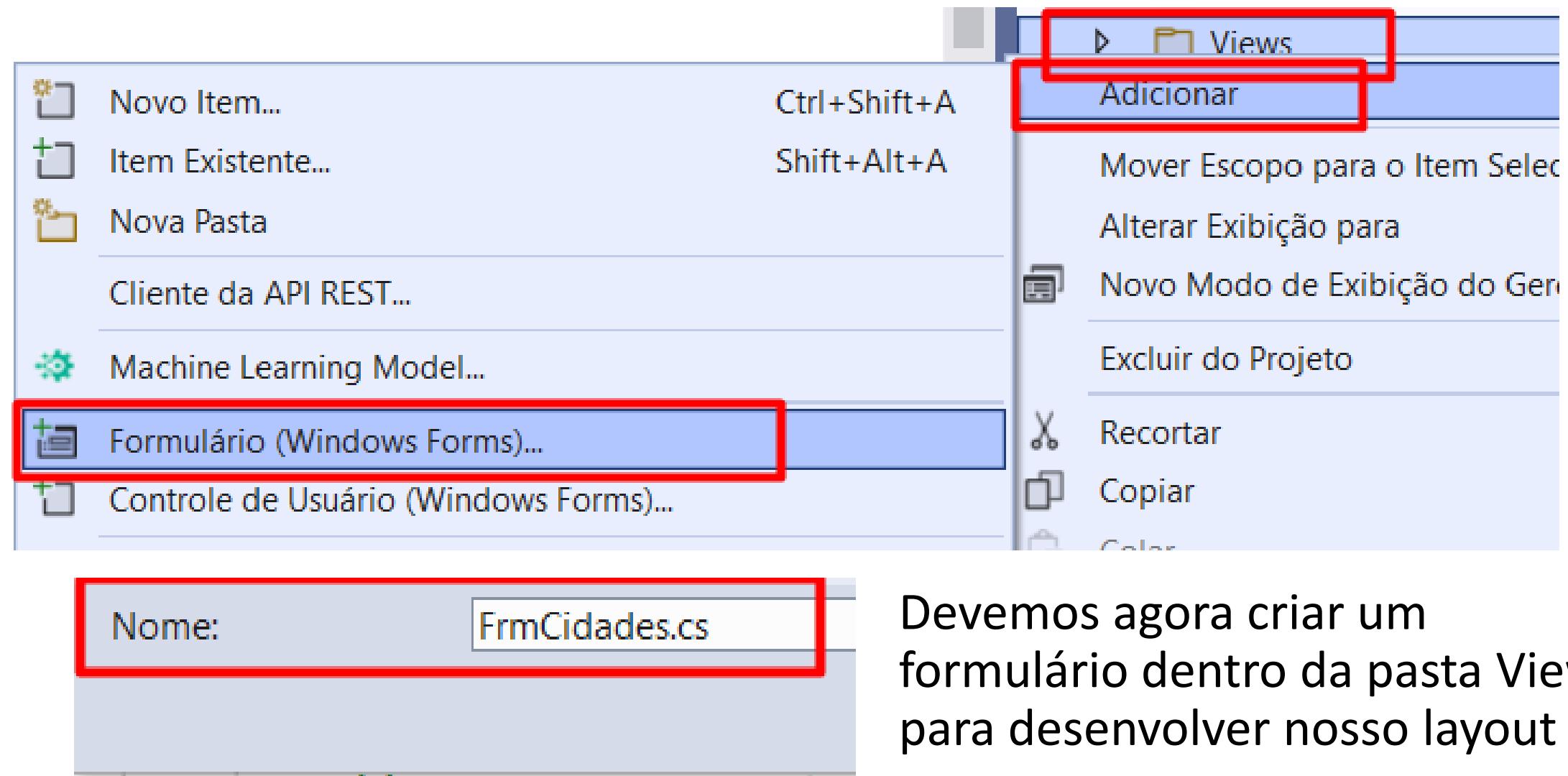
|   | id | nome   | uf |
|---|----|--------|----|
| ▶ | 2  | BARIRI | SP |
| ● | 1  | JAU    | SP |

DataGridView – Controle com a propriedade DataSource

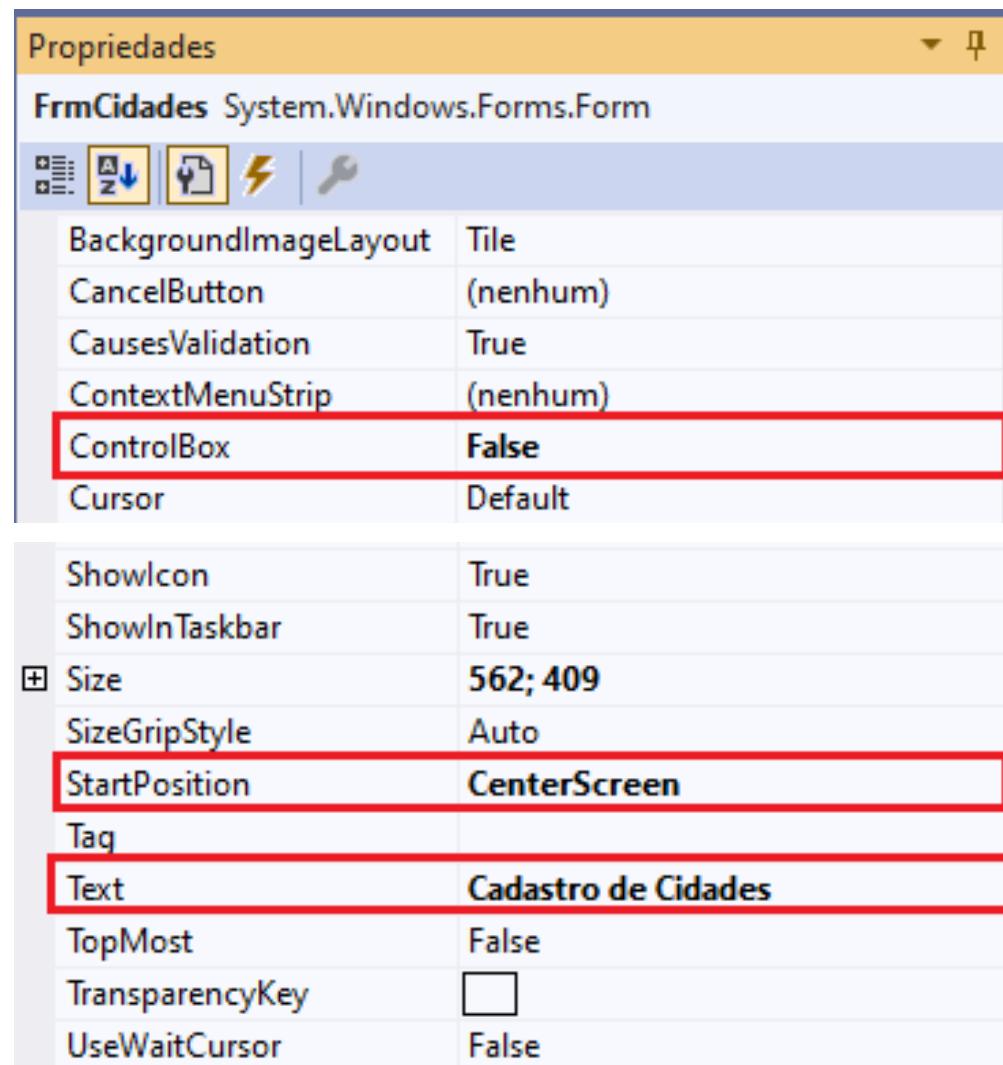
```
dgvCidades.DataSource = datTabela;
```

# Adicionando formulário de cidades

31

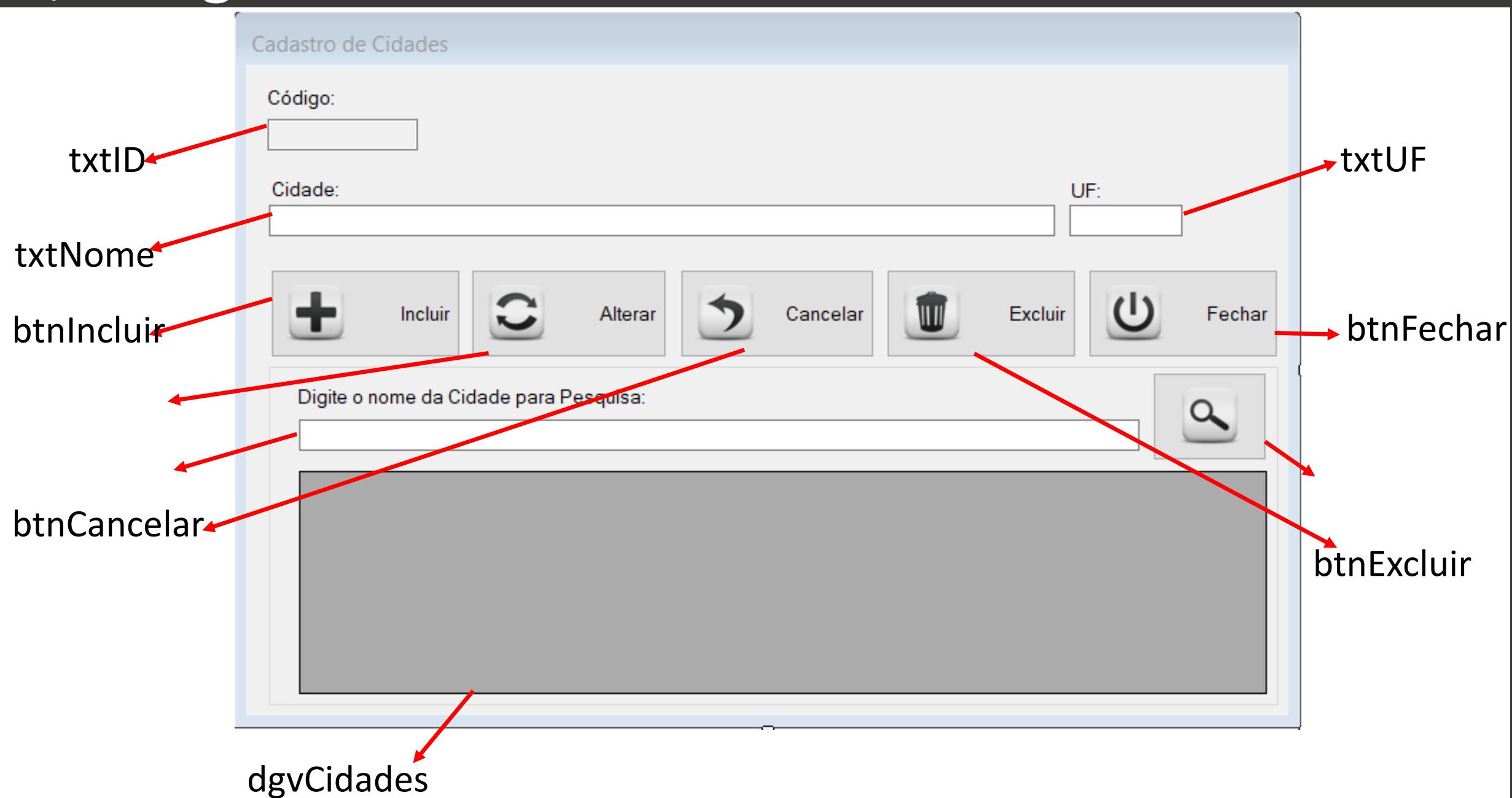


Devemos agora criar um formulário dentro da pasta Views para desenvolver nosso layout



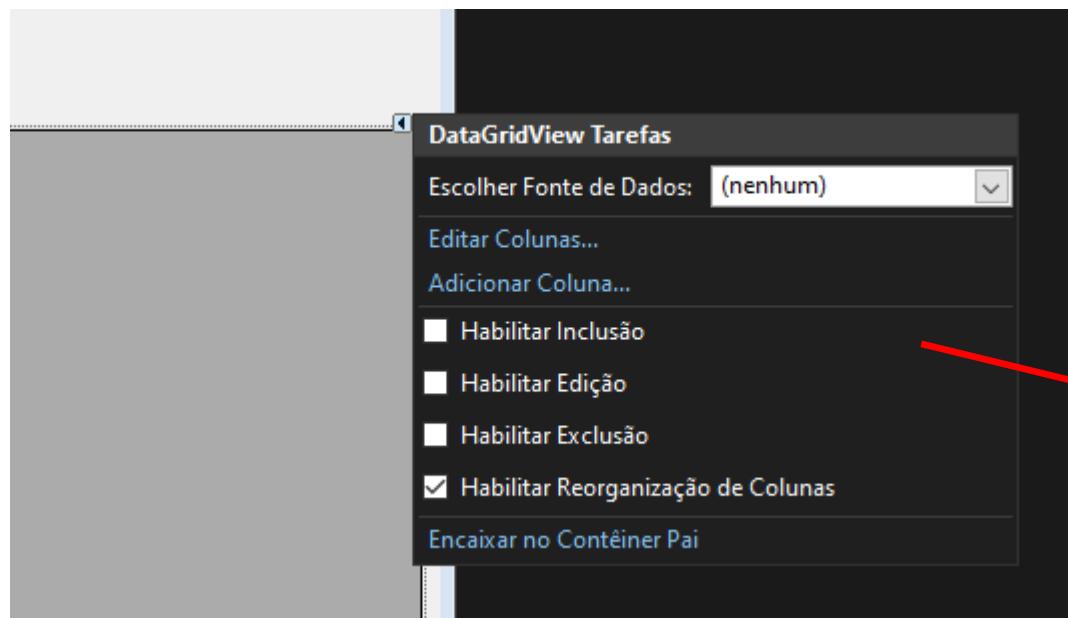
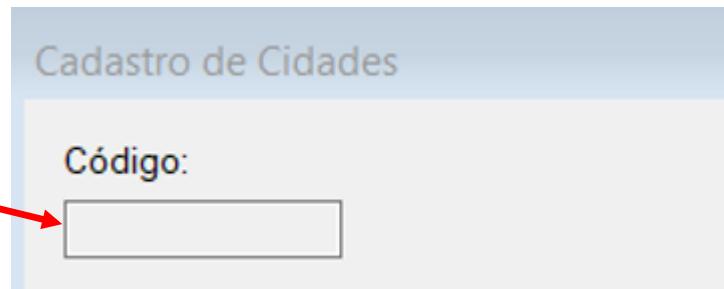
# X Design do Formulário de Cidades

33

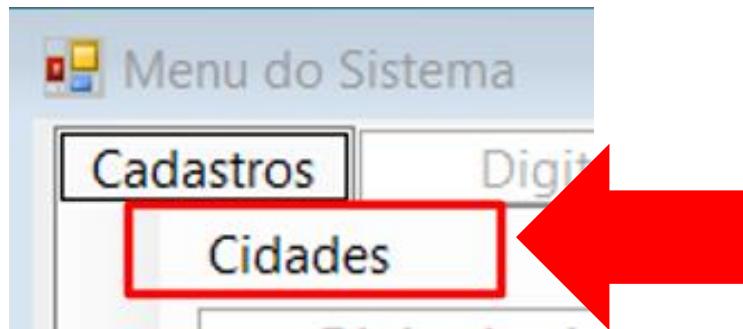




A propriedade enabled da caixa referente ao ID deverá estar como false



Nosso grid deverá ter essas configurações  
As colunas dos cadastros NÃO serão criadas manuais, a própria conexão com o banco irá criar elas automaticamente



1 referência

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

```
using System;
using System.Windows.Forms;
using vendas.Views;

namespace vendas
{
    3 referências
    public partial class FrmMenu : Form
    {
        1 referência
        public FrmMenu()
        {
            InitializeComponent();
        }

        1 referência
        private void cidadesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            FrmCidades form = new FrmCidades();
            form.Show();
        }
    }
}
```

Criaremos  
uma variável  
para a nossa  
classe de  
modelagem

```
-> using System;
    using System.Windows.Forms;
    using vendas.Models;

-> namespace vendas.Views
{
    5 referências
    public partial class FrmCidades : Form
    {
        Cidade c;
        1 referência
        public FrmCidades()
        {
            InitializeComponent();
        }
}
```

```
public FrmCidades()
{
    InitializeComponent();
}
```

5 referências

```
void limpaControles()
{
    txtId.Clear();
    txtNome.Clear();
    txtUF.Clear();
    txtPesquisa.Clear();
}
```

**Função para limpar nossos controles do formulário.**

**Como essa ação acontecerá várias vezes, é muito interessante criar a função para otimizarmos nosso código**

Criação do  
objeto na  
memória

```
6 referências
void carregarGrid(string pesquisa)
{
    Cidade c = new Cidade()
    {
        nome = pesquisa
    };
    DgvCidades.DataSource = c.Consultar();
}
```

1 referência

```
private void FrmCidades_Load(object sender, EventArgs e)
{
    limpaControles();
    carregarGrid("");
}
```

# X Formulário de Cidades

40

Cadastro de Cidades

Código:

Cidade:

 UF:

 Incluir    Alterar    Cancelar    Excluir    Fechar

Digite o nome da Cidade para Pesquisa:

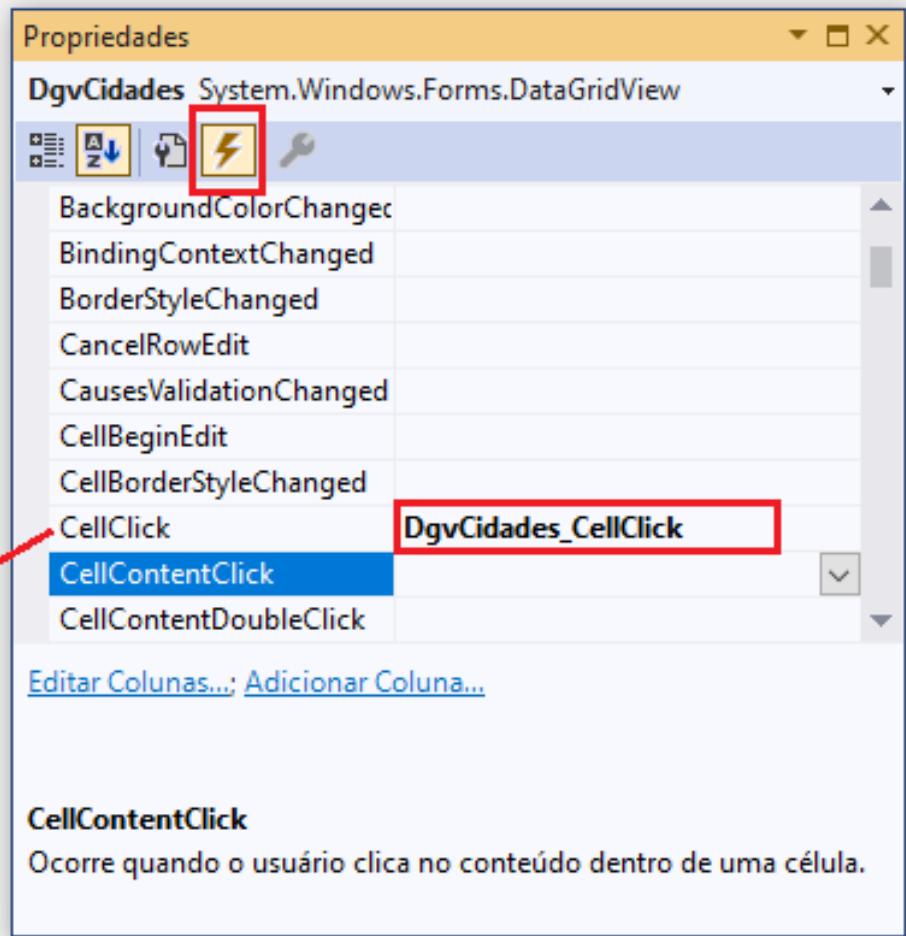
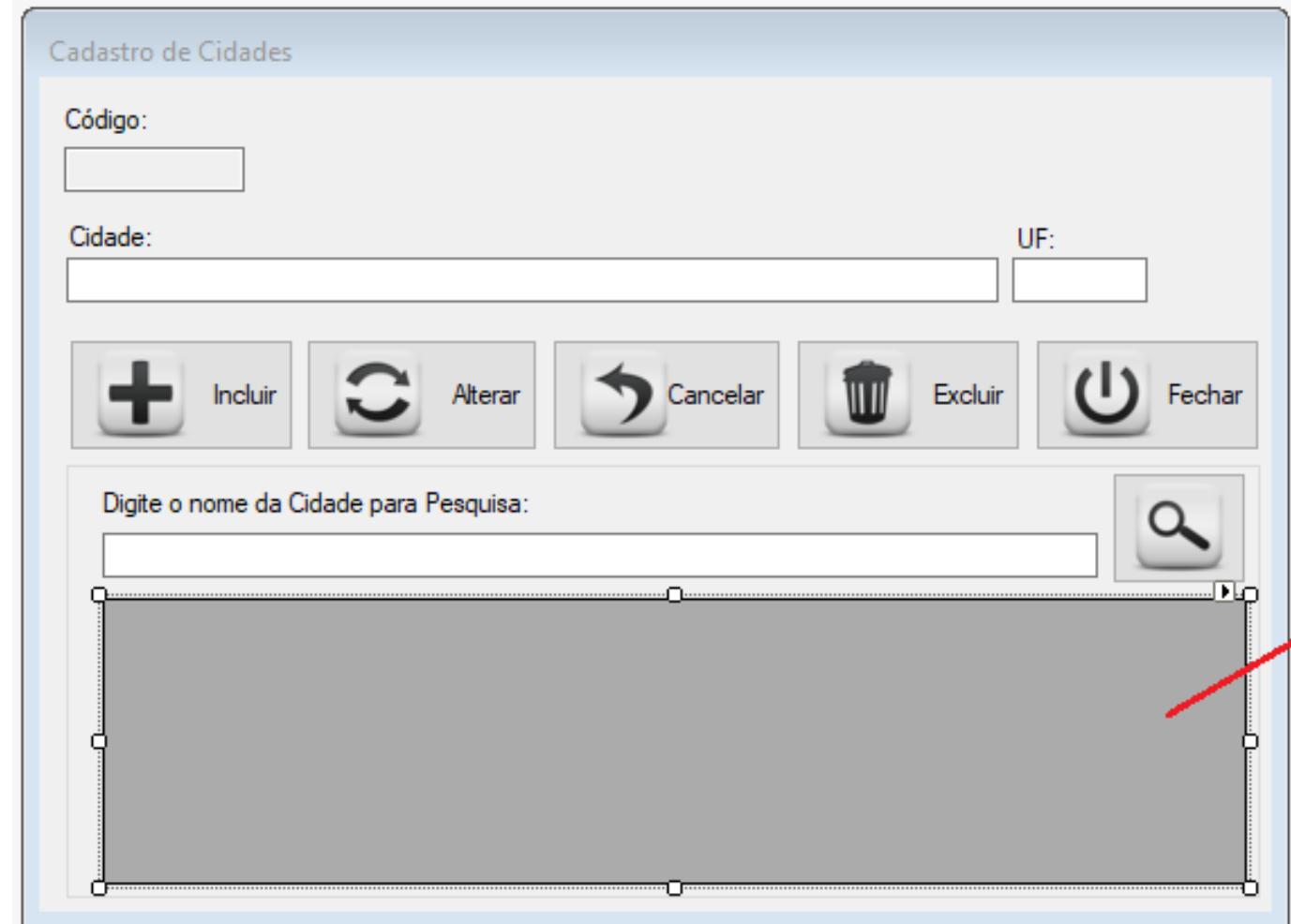
|  | <b>id</b> | <b>nome</b> | <b>uf</b> |
|--|-----------|-------------|-----------|
|--|-----------|-------------|-----------|

1 referência

```
private void btnIncluir_Click(object sender, EventArgs e)
{
    if (txtNome.Text == String.Empty) return;

    c = new Cidade()
    {
        nome = txtNome.Text,
        uf = txtUF.Text
    };
    c.Incluir();

    limpaControles();
    carregarGrid("");
}
```



1 referência

```
private void DgvCidades_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (DgvCidades.RowCount > 0)
    {
        txtId.Text = DgvCidades.CurrentRow.Cells["id"].Value.ToString();
        txtNome.Text = DgvCidades.CurrentRow.Cells["nome"].Value.ToString();
        txtUF.Text = DgvCidades.CurrentRow.Cells["uf"].Value.ToString();
    }
}
```

```
private void btnAlterar_Click(object sender, EventArgs e)
{
    if (txtId.Text == String.Empty) return;

    c = new Cidade()
    {
        id = int.Parse(txtId.Text),
        nome = txtNome.Text,
        uf = txtUF.Text
    };
    c.Alterar();

    limpaControles();
    carregarGrid("");
}
```

```
public class Cidade  
{  
    4 referências  
    public int id { get; set; }  
    6 referências  
    public string nome { get; set; }  
    4 referências  
    public string uf { get; set; }  
  
    c = new Cidade()  
    {  
        id = int.Parse(txtId.Text),  
        nome = txtNome.Text,  
        uf = txtUF.Text  
    };
```



Slide apenas informativo

c.Alterar();

```
public void Alterar()
{
    try
    {
        // Abre a conexão com o banco
        Banco.AbrirConexao();
```



**Slide apenas  
informativo**

```
private void btnExcluir_Click(object sender, EventArgs e)
{
    if (txtId.Text == "") return;

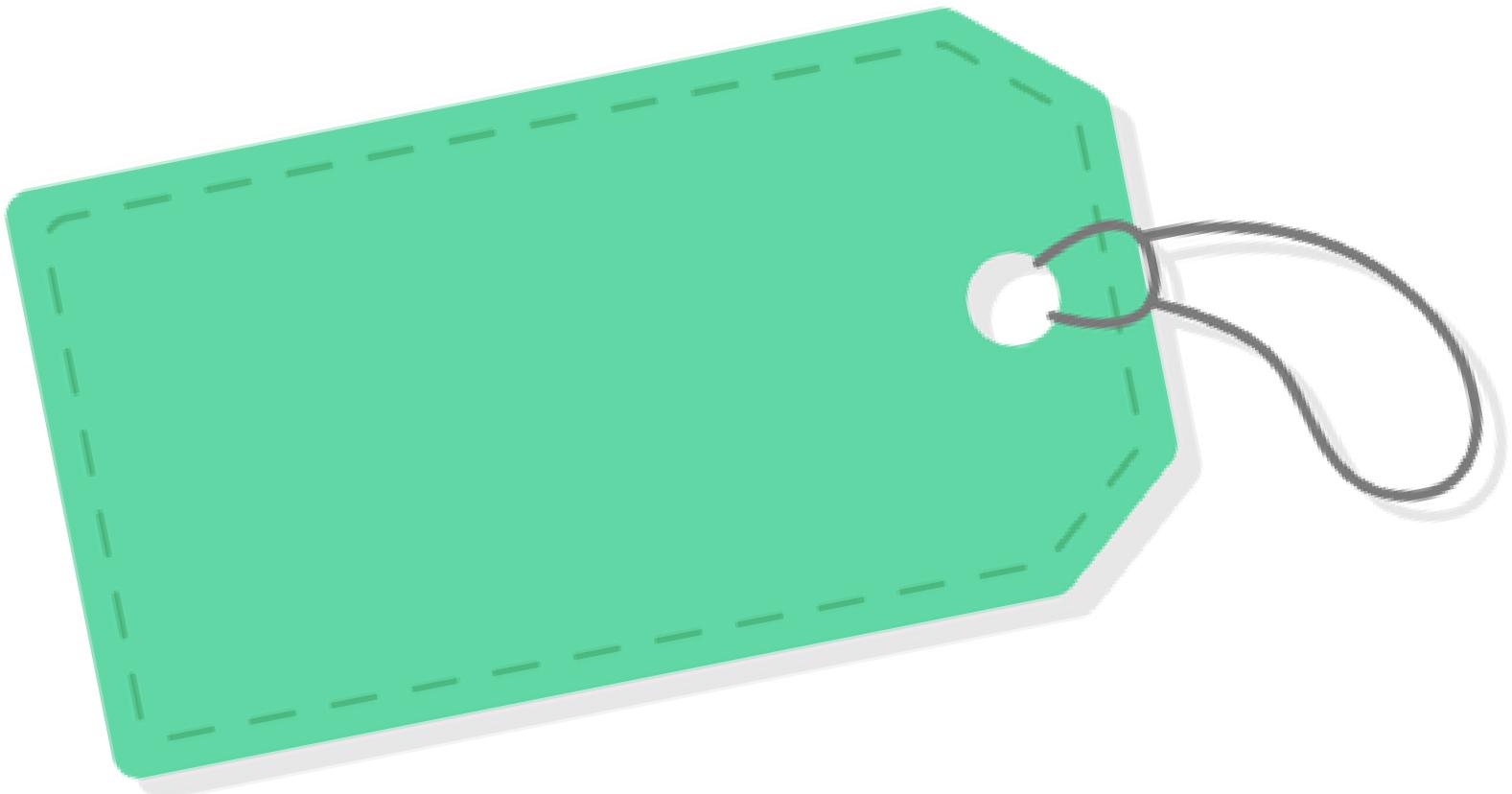
    if (MessageBox.Show("Deseja excluir a cidade?", "Exclusão",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {
        c = new Cidade()
        {
            id = int.Parse(txtId.Text)
        };
        c.Excluir();

        limpaControles();
        carregarGrid("");
    }
}
```

```
private void btnCancelar_Click(object sender, EventArgs e)
{
    limpaControles();
    carregarGrid("");
}

private void btnConsultar_Click(object sender, EventArgs e)
{
    carregarGrid(txtPesquisa.Text);
}

private void btnFechar_Click(object sender, EventArgs e)
{
    Close();
}
```



```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Cidades " +
    "(id integer auto_increment primary key, " +
    "nome char(40), " +
    "uf char(02))", Conexao);
Comando.ExecuteNonQuery();
```

```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Marcas " +
    "(id integer auto_increment primary key, " +
    "marca char(20))", Conexao);
Comando.ExecuteNonQuery();
```



```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Cidades " +
    "(id integer auto_increment primary key, " +
    "nome char(40), " +
    "uf char(02))", Conexao);
```

```
Comando.ExecuteNonQuery();
```

---

```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Marcas " +
    "(id integer auto_increment primary key, " +
    "marca char(20))", Conexao);
```

```
Comando.ExecuteNonQuery();
```

```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Categorias " +
    "(id integer auto_increment primary key, " +
    "categoria char(20))", Conexao);
```

```
Comando.ExecuteNonQuery();
```



```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Categorias " +
    "(id integer auto_increment primary key, " +
    "categoria char(20))", Conexao);
Comando.ExecuteNonQuery();
```

```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Clientes " +
    "(Id integer auto_increment primary key, " +
    "nome char(40), " +
    "idCidade integer," +
    "dataNasc date," +
    "renda decimal(10,2), " +
    "cpf char(14), " +
    "foto varchar(100), " +
    "venda boolean)", Conexao);
Comando.ExecuteNonQuery();
```

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Windows.Forms;

namespace vendas.Models
{
    7 referências
    public class Cliente
    {
        4 referências
        public int id { get; set; }
        6 referências
        public string nome { get; set; }
        4 referências
        public int idCidade { get; set; }
        4 referências
        public DateTime dataNasc { get; set; }
        4 referências
        public double renda { get; set; }
        4 referências
        public string cpf { get; set; }
        4 referências
        public string foto { get; set; }
        4 referências
        public bool venda { get; set; }
```

```
public void Incluir()
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand
            ("INSERT INTO clientes (nome, idCidade, dataNasc, renda, cpf, foto, venda) " +
             "VALUES (@nome, @idCidade, @dataNasc, @renda, @cpf, @foto, @venda)", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@nome", nome);
        Banco.Comando.Parameters.AddWithValue("@idCidade", idCidade);
        Banco.Comando.Parameters.AddWithValue("@dataNasc", dataNasc);
        Banco.Comando.Parameters.AddWithValue("@renda", renda);
        Banco.Comando.Parameters.AddWithValue("@cpf", cpf);
        Banco.Comando.Parameters.AddWithValue("@foto", foto);
        Banco.Comando.Parameters.AddWithValue("@venda", venda);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

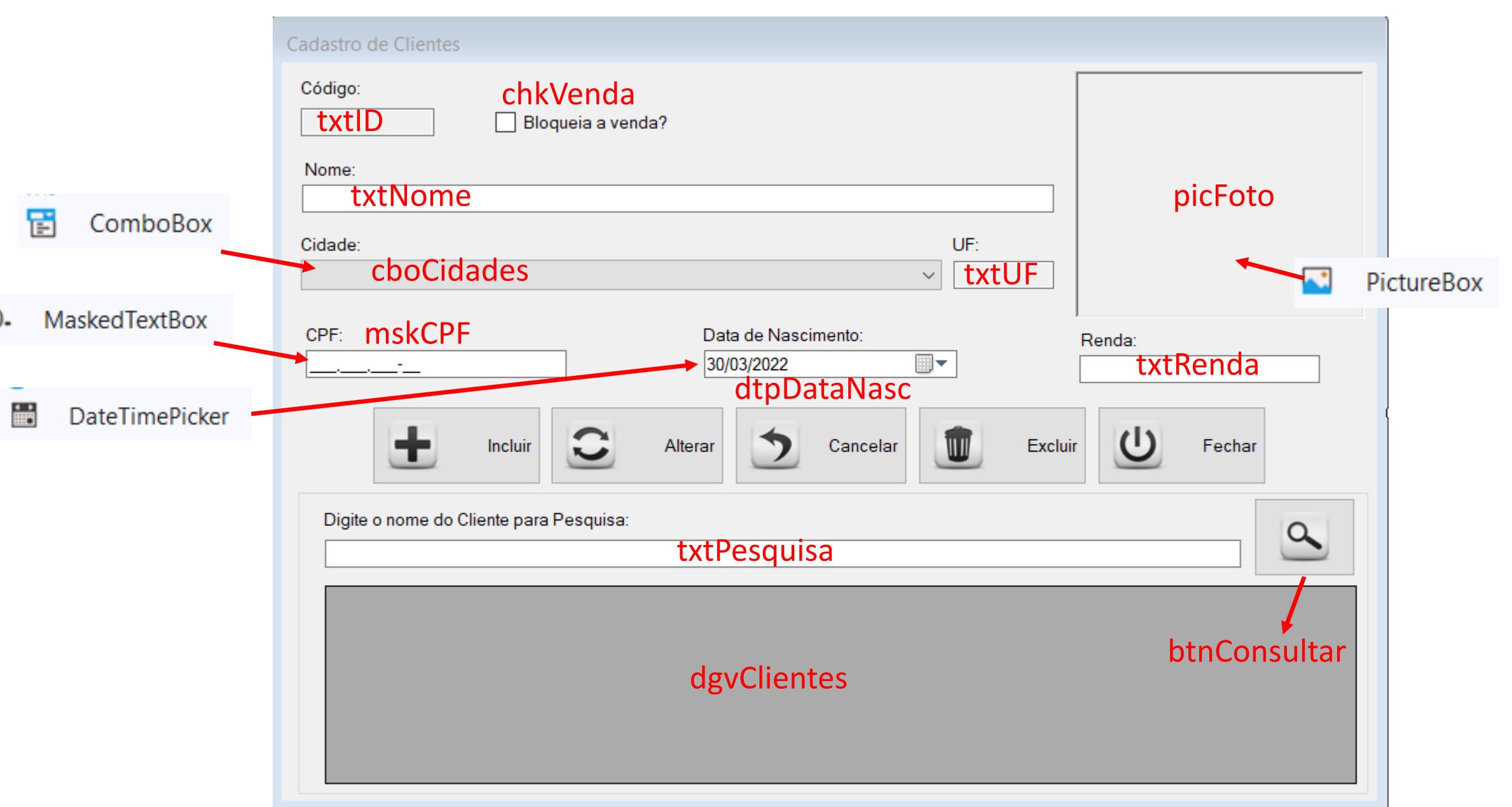
```
public void Alterar()
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand
        ("UPDATE clientes SET nome = @nome, idCidade = @idCidade, dataNasc = @dataNasc, " +
         "renda = @renda, cpf = @cpf, foto = @foto, venda = @venda where id = @id", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@nome", nome);
        Banco.Comando.Parameters.AddWithValue("@idCidade", idCidade);
        Banco.Comando.Parameters.AddWithValue("@dataNasc", dataNasc);
        Banco.Comando.Parameters.AddWithValue("@renda", renda);
        Banco.Comando.Parameters.AddWithValue("@cpf", cpf);
        Banco.Comando.Parameters.AddWithValue("@foto", foto);
        Banco.Comando.Parameters.AddWithValue("@venda", venda);
        Banco.Comando.Parameters.AddWithValue("@id", id);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

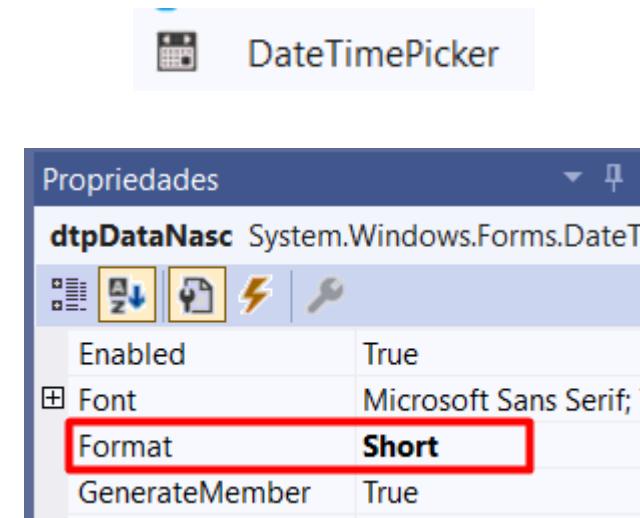
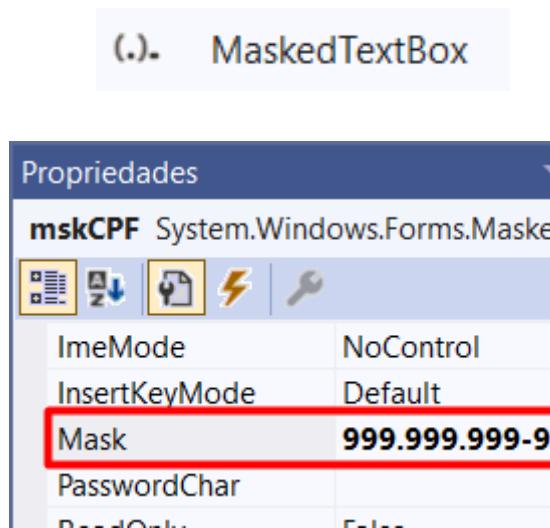
```
public void Excluir()
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand("delete from clientes where id = @id", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@id", id);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

```
public DataTable Consultar()
{
    try
    {
        Banco.Comando = new MySqlCommand("SELECT cl.*, ci.nome cidade, " +
            "ci.uf FROM Clientes cl inner join Cidades ci on (ci.id = cl.idCidade) " +
            "where cl.nome like ?Nome order by cl.nome", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@Nome", nome + "%");
        Banco.Adaptador = new MySqlDataAdapter(Banco.Comando);
        Banco.datTabela = new DataTable();
        Banco.Adaptador.Fill(Banco.datTabela);
        return Banco.datTabela;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
}
```

# Formulário de clientes

60





```
using System;
using System.Data;
using System.Windows.Forms;
using vendas.Models;

namespace vendas.Views
{
    5 referências
    public partial class FrmClientes : Form
    {
        Cidade ci;
        Cliente cl;

        1 referência
        public FrmClientes()
        {
            InitializeComponent();
        }
    }
}
```

5 referências

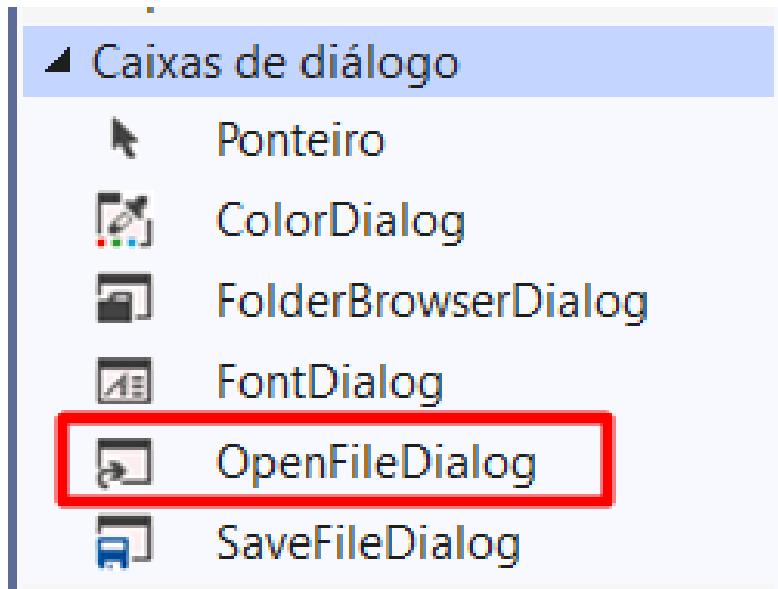
```
void LimpaControles()
{
    txtId.Clear();
    txtNome.Clear();
    cboCidades.SelectedIndex = -1;
    txtUF.Clear();
    mskCPF.Clear();
    txtRenda.Clear();
    dtpDataNasc.Value = DateTime.Now;
    picFoto.ImageLocation = "";
    chkVenda.Checked = false;
}
```

```
private void FrmClientes_Load(object sender, EventArgs e)
{
    // Cria um objeto do tipo cidade
    // E alimenta o comboBox
    ci = new Cidade();
    cboCidades.DataSource = ci.Consultar();
    cboCidades.DisplayMember = "nome";
    cboCidades.ValueMember = "id";

    limpaControles();
    carregarGrid("");

    // Deixa invisivel colunas do Grid
    DgvClientes.Columns["idCidade"].Visible = false;
    DgvClientes.Columns["foto"].Visible = false;
}
```

```
private void cboCidades_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cboCidades.SelectedIndex != -1)
    {
        DataRowView reg = (DataRowView)cboCidades.SelectedItem;
        txtUF.Text = reg["uf"].ToString();
    }
}
```

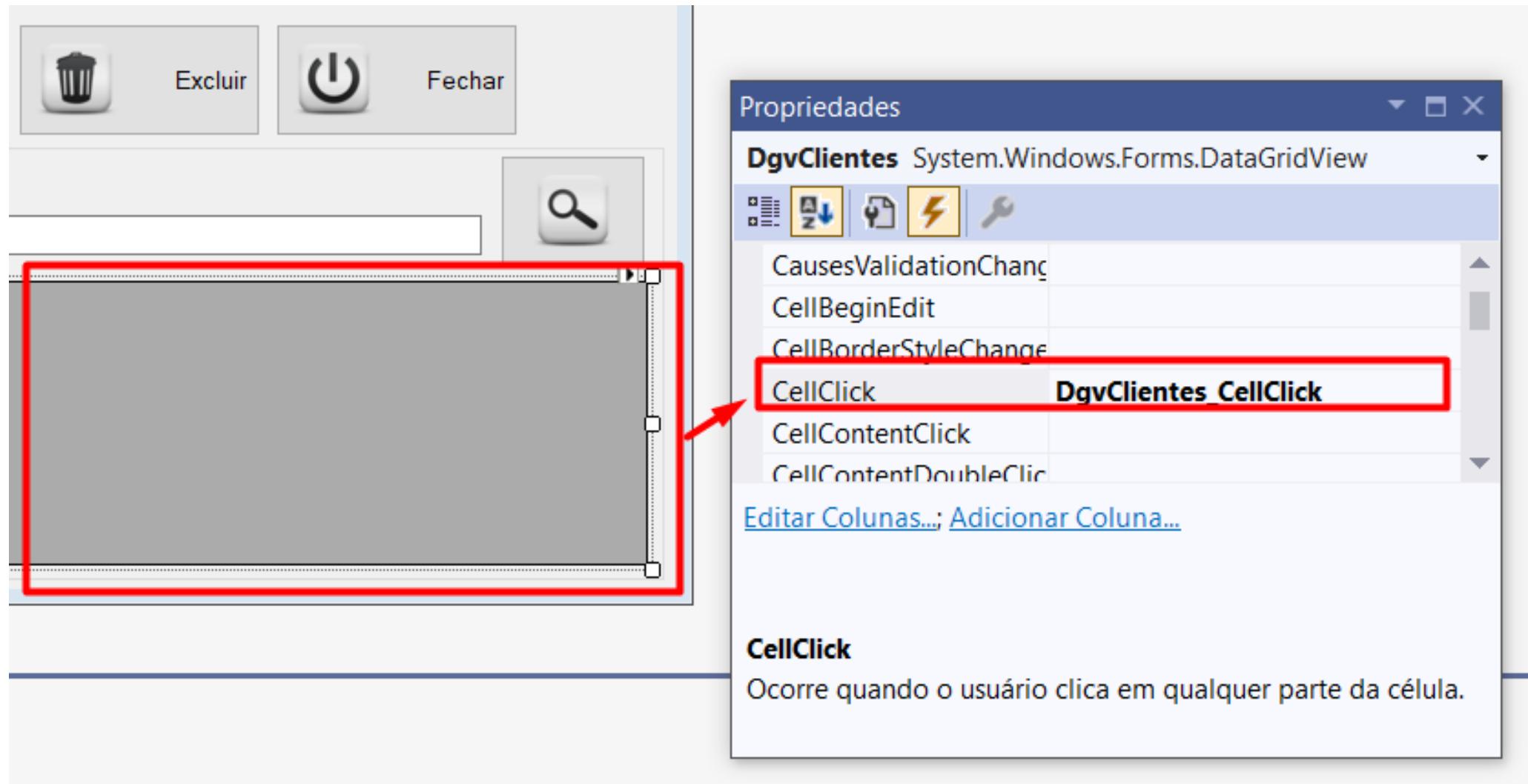


```
private void picFoto_Click(object sender, EventArgs e)
{
    ofdArquivo.InitialDirectory = "D:/fotos/clientes/";
    ofdArquivo.FileName = "";
    ofdArquivo.ShowDialog();
    picFoto.ImageLocation = ofdArquivo.FileName;
}
```

```
private void btnIncluir_Click(object sender, EventArgs e)
{
    if (txtNome.Text == "") return;

    cl = new Cliente()
    {
        nome = txtNome.Text,
        idCidade = (int) cboCidades.SelectedValue,
        dataNasc = dtpDataNasc.Value,
        renda = double.Parse(txtRenda.Text),
        cpf = mskCPF.Text,
        foto = picFoto.ImageLocation,
        venda = chkVenda.Checked
    };
    cl.Incluir();

    limpaControles();
    carregarGrid("");
}
```



```
private void DgvClientes_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (DgvClientes.RowCount > 0)
    {
        txtId.Text = DgvClientes.CurrentRow.Cells["id"].Value.ToString();
        txtNome.Text = DgvClientes.CurrentRow.Cells["nome"].Value.ToString();
        cboCidades.Text = DgvClientes.CurrentRow.Cells["cidade"].Value.ToString();
        txtUF.Text = DgvClientes.CurrentRow.Cells["uf"].Value.ToString();
        chkVenda.Checked = (bool)DgvClientes.CurrentRow.Cells["venda"].Value;
        mskCPF.Text = DgvClientes.CurrentRow.Cells["cpf"].Value.ToString();
        dtpDataNasc.Text = DgvClientes.CurrentRow.Cells["dataNasc"].Value.ToString();
        txtRenda.Text = DgvClientes.CurrentRow.Cells["renda"].Value.ToString();
        picFoto.ImageLocation = DgvClientes.CurrentRow.Cells["foto"].Value.ToString();
    }
}
```

```
private void btnAlterar_Click(object sender, EventArgs e)
{
    if (txtId.Text == "") return;

    cl = new Cliente()
    {
        id = int.Parse(txtId.Text),
        nome = txtNome.Text,
        idCidade = (int)cboCidades.SelectedValue,
        dataNasc = dtpDataNasc.Value,
        renda = double.Parse(txtRenda.Text),
        cpf = mskCPF.Text,
        foto = picFoto.ImageLocation,
        venda = chkVenda.Checked
    };
    cl.Alterar();

    limpaControles();
    carregarGrid("");
}
```

```
private void btnExcluir_Click(object sender, EventArgs e)
{
    if (txtId.Text == "") return;

    if (MessageBox.Show("Deseja excluir o cliente?", "Exclusão",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
    {
        cl = new Cliente()
        {
            id = int.Parse(txtId.Text)
        };
        cl.Excluir();

        limpaControles();
        carregarGrid("");
    }
}
```

```
private void btnCancelar_Click(object sender, EventArgs e)
{
    limpaControles();
    carregarGrid("");
}

private void btnConsultar_Click(object sender, EventArgs e)
{
    carregarGrid(txtPesquisa.Text);
}

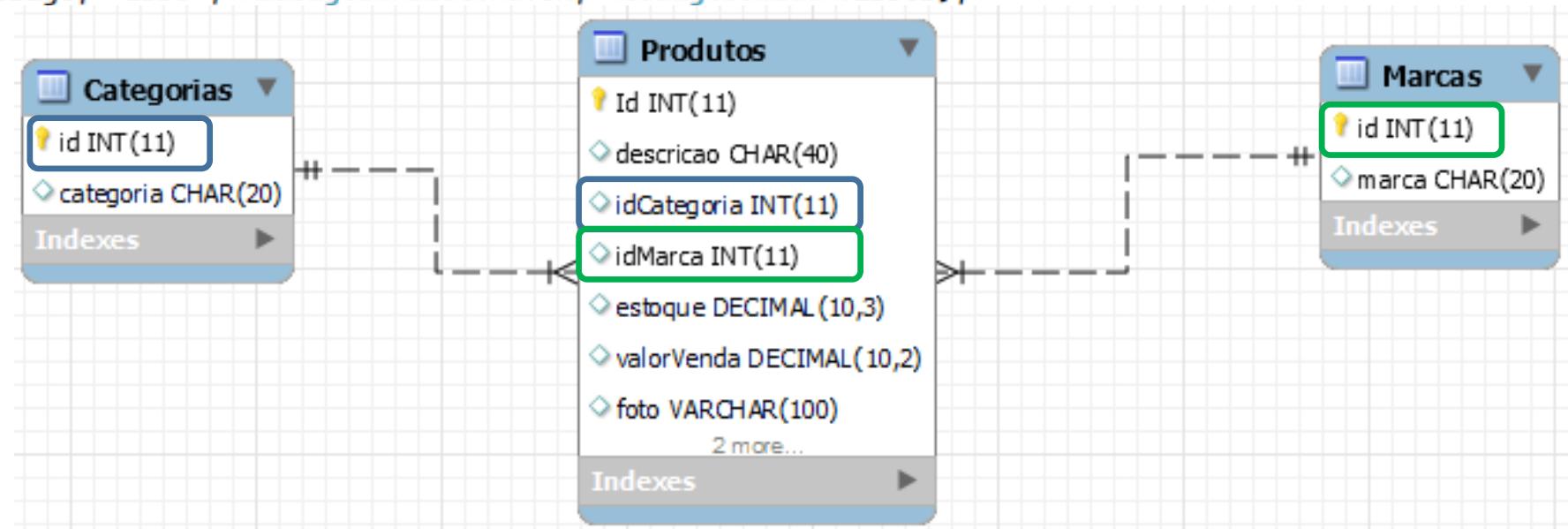
private void btnFechar_Click(object sender, EventArgs e)
{
    Close();
}
```



```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Produtos " +
    "(Id integer auto_increment primary key,
     descricao char(40), " +
     "idCategoria integer," +
     "idMarca integer," +
     "estoque decimal(10,3), " +
     "valorVenda decimal(10,2), " +
     "foto varchar(100))", Conexao);

Comando.ExecuteNonQuery();
```

```
public DataTable Consultar()
{
    try
    {
        Banco.Comando = new MySqlCommand("SELECT p.*, m.marca, c.categoria FROM " +
            "Produtos p inner join Marcas m on (m.id = p.idMarca) " +
            "inner join Categorias c on (c.id = p.idCategoria) " +
            "where p.descricao like @descricao order by p.descricao", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@descricao", descricao + "%");
        Banco.Adaptador = new MySqlDataAdapter(Banco.Comando);
        Banco.datTabela = new DataTable();
        Banco.Adaptador.Fill(Banco.datTabela);
        return Banco.datTabela;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return null;
    }
}
```



# Formulário de Produtos

77

Cadastro de Produtos

Código:

Descrição:

Categoria:

Marca:

Valor de Venda:  Estoque:

 Incluir  Alterar  Cancelar  Excluir  Fechar

Digite o Produto para Pesquisa:  



```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS VendaCab " +
    "(Id integer auto_increment primary key, " +
    "idCliente int, " +
    "data date, " +
    "total decimal(10,2))", Conexao);
Comando.ExecuteNonQuery();

Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS VendaDet " +
    "(Id integer auto_increment primary key, " +
    "idVendaCab int, " +
    "idProduto int, " +
    "qtde decimal(10,3), " +
    "valorUnitario decimal(10,2))", Conexao);
Comando.ExecuteNonQuery();
```

```
using MySql.Data.MySqlClient;
using System;
using System.Windows.Forms;

namespace vendas.Models
{
    public class VendaCab
    {
        public int Id { get; set; }
        public int idCliente { get; set; }
        public DateTime data { get; set; }
        public double total { get; set; }
    }
}
```

```
public int Incluir()
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand(
            "INSERT INTO vendaCab (idCliente, data, total) " +
            "VALUES (@idCliente, @data, @total)", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@idCliente", idCliente);
        Banco.Comando.Parameters.AddWithValue("@data", data);
        Banco.Comando.Parameters.AddWithValue("@total", total);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
        return (int)Banco.Comando.LastInsertedId;
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return 0;
    }
}
```



```
[-] using MySql.Data.MySqlClient;
[+] using System;
[+] using System.Windows.Forms;

[-] namespace vendas.Models
{
    2 referências
    [-] public class VendaDet
    {
        0 referências
        [-] public int Id { get; set; }
        2 referências
        [-] public int idVendaCab { get; set; }
        2 referências
        [-] public int idProduto { get; set; }
        2 referências
        [-] public double qtde { get; set; }
        2 referências
        [-] public double valorUnitario { get; set; }
    }
}
```

```
public void Incluir()
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand(
            "INSERT INTO vendaDet (idVendaCab, idProduto, qtde, valorUnitario) " +
            "VALUES (@idVendaCab, @idProduto, @qtde, @valorUnitario)", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@idVendaCab", idVendaCab);
        Banco.Comando.Parameters.AddWithValue("@idProduto", idProduto);
        Banco.Comando.Parameters.AddWithValue("@qtde", qtde);
        Banco.Comando.Parameters.AddWithValue("@valorUnitario", valorUnitario);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

# Formulário de Vendas

84

Formulário de Vendas

| Cliente:                                                                                                                                                             | cboClientes          |        |          |           |            |        |         |      |       |             |  |  |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|--------|----------|-----------|------------|--------|---------|------|-------|-------------|--|--|--|
| Cidade:                                                                                                                                                              | txtCidade            | txtUF  |          |           |            |        |         |      |       |             |  |  |  |
| Nascimento:                                                                                                                                                          | mskDataNasc          | CPF:   | mskCPF   | Renda:    | txtRenda   |        |         |      |       |             |  |  |  |
| <input type="checkbox"/> Bloqueia a venda                                                                                                                            |                      |        | chkVenda | Confirmar |            |        |         |      |       |             |  |  |  |
| Produto:                                                                                                                                                             | cboProdutos          |        |          |           |            |        |         |      |       |             |  |  |  |
| Marca:                                                                                                                                                               | txtMarca             |        |          |           |            |        |         |      |       |             |  |  |  |
| Categoria:                                                                                                                                                           | txtCategoria         |        |          |           |            |        |         |      |       |             |  |  |  |
| Quantidade:                                                                                                                                                          | txtQuantidade        | Preço: | txtPreco | Estoque:  | txtEstoque |        |         |      |       |             |  |  |  |
| <table border="1"><thead><tr><th>CÓDIGO</th><th>PRODUTO</th><th>QTDE</th><th>VALOR</th></tr></thead><tbody><tr><td colspan="4">dgvProdutos</td></tr></tbody></table> |                      |        |          |           |            | CÓDIGO | PRODUTO | QTDE | VALOR | dgvProdutos |  |  |  |
| CÓDIGO                                                                                                                                                               | PRODUTO              | QTDE   | VALOR    |           |            |        |         |      |       |             |  |  |  |
| dgvProdutos                                                                                                                                                          |                      |        |          |           |            |        |         |      |       |             |  |  |  |
| Total:                                                                                                                                                               | 0,00 <b>lblTotal</b> |        |          |           |            |        |         |      |       |             |  |  |  |
| <b>btnInserir</b> <b>btnRemover</b>                                                                                                                                  |                      |        |          |           |            |        |         |      |       |             |  |  |  |
| <b>btnGravar</b> <b>btnCancelar</b> <b>btnFechar</b>                                                                                                                 |                      |        |          |           |            |        |         |      |       |             |  |  |  |
| <b>Gravar</b> <b>Cancelar</b> <b>Fechar</b>                                                                                                                          |                      |        |          |           |            |        |         |      |       |             |  |  |  |

```
- using System;
using System.Data;
using System.Windows.Forms;
using vendas.Models;

namespace vendas.Views
{
    5 referências
    public partial class FrmVendas : Form
    {
        double total;

        Cliente c;
        Produto p;
        VendaCab vc;
        VendaDet vd;

        1 referência
        public FrmVendas()
        {
            InitializeComponent();
        }
    }
}
```

2 referências

```
void LimpaProduto()
{
    cboProdutos.SelectedIndex = -1;
    txtEstoque.Clear();
    txtPreco.Clear();
    txtQuantidade.Clear();
    txtMarca.Clear();
    txtCategoria.Clear();
    picProduto.ImageLocation = "";
}
```

```
private void btnCancelar_Click(object sender, EventArgs e)
{
    dgvProdutos.RowCount = 0;
    cboClientes.SelectedIndex = -1;
    txtCidade.Clear();
    txtUF.Clear();
    txtRenda.Clear();
    mskCPF.Clear();
    mskDataNascimento.Clear();
    chkVenda.Checked = false;
    picCliente.ImageLocation = "";
    total = 0;
    lblTotal.Text = total.ToString("C");
    grbClientes.Enabled = true;
    grbProdutos.Enabled = false;
    LimpaProduto();
}
```

```
private void FrmVendas_Load(object sender, EventArgs e)
{
    c = new Cliente();
    cboClientes.DataSource = c.Consultar();
    cboClientes.DisplayMember = "nome";
    cboClientes.ValueMember = "id";

    p = new Produto();
    cboProdutos.DataSource = p.Consultar();
    cboProdutos.DisplayMember = "descricao";
    cboProdutos.ValueMember = "id";

    btnCancelar.PerformClick();
}
```

```
private void cboClientes_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cboClientes.SelectedIndex != -1)
    {
        DataRowView reg = (DataRowView)cboClientes.SelectedItem;
        txtCidade.Text = reg["CIDADE"].ToString();
        txtUF.Text = reg["UF"].ToString();
        txtRenda.Text = reg["RENDASALARIO"].ToString();
        mskCPF.Text = reg["CPF"].ToString();
        mskDataNasc.Text = reg["DATANASCIMENTO"].ToString();
        picCliente.ImageLocation = reg["FOTO"].ToString();
        chkVenda.Checked = (bool)reg["VENDA"];
    }
}
```

```
private void btnConfirmar_Click(object sender, EventArgs e)
{
    if (cboClientes.SelectedIndex != -1)
    {
        if (chkVenda.Checked)
        {
            MessageBox.Show("Cliente bloqueado para venda", "Vendas",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            btnCancelar.PerformClick();
            return;
        }
        grbClientes.Enabled = false;
        grbProdutos.Enabled = true;
    }
}
```

```
private void cboProdutos_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cboProdutos.SelectedIndex != -1)
    {
        DataRowView reg = (DataRowView)cboProdutos.SelectedItem;
        txtEstoque.Text = reg["estoque"].ToString();
        txtPreco.Text = reg["valorVenda"].ToString();
        txtMarca.Text = reg["Marca"].ToString();
        txtCategoria.Text = reg["Categoria"].ToString();
        picProduto.ImageLocation = reg["foto"].ToString();
    }
}
```

```
private void btnInserir_Click(object sender, EventArgs e)
{
    double quantidade = double.Parse(txtQuantidade.Text);
    double estoque = double.Parse(txtEstoque.Text);

    if (quantidade > estoque)
    {
        MessageBox.Show("Estoque insuficiente", "Vendas",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtQuantidade.SelectAll();
        return;
    }

    dgvProdutos.Rows.Add(cboProdutos.SelectedValue, cboProdutos.Text,
        txtQuantidade.Text, txtPreco.Text);

    double preco = double.Parse(txtPreco.Text);

    total += quantidade * preco;
    lblTotal.Text = total.ToString("C");
    limpaProduto();
}
```

```
private void btnRemover_Click(object sender, EventArgs e)
{
    if (dgvProdutos.RowCount > 0)
    {
        double quantidade = double.Parse(dgvProdutos.CurrentRow.Cells[2].Value.ToString());
        double preco = double.Parse(dgvProdutos.CurrentRow.Cells[3].Value.ToString());

        total -= quantidade * preco;
        lblTotal.Text = total.ToString("C");

        dgvProdutos.Rows.RemoveAt(dgvProdutos.CurrentRow.Index);
    }
}
```

```
public void atualizaEstoque(double qtde)
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand(
            "Update produtos set estoque = estoque - @qtde where id = @id", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@qtde", qtde);
        Banco.Comando.Parameters.AddWithValue("@id", Id);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

```
private void btnGravar_Click(object sender, EventArgs e)
{
    vc = new VendaCab()
    {
        idCliente = (int)cboClientes.SelectedValue,
        data = DateTime.Now,
        total = total
    };

    int idVenda = vc.Incluir();
```

```
foreach (DataGridViewRow linha in dgvProdutos.Rows)
{
    vd = new VendaDet()
    {
        idVendaCab = idVenda,
        idProduto = Convert.ToInt32(linha.Cells[0].Value),
        qtde = Convert.ToDouble(linha.Cells[2].Value),
        valorUnitario = Convert.ToDouble(linha.Cells[3].Value)
    };
    vd.Incluir();

    p = new Produto()
    {
        Id = (int)linha.Cells[0].Value
    };
    p.atualizaEstoque(Convert.ToDouble(linha.Cells[2].Value));
}

btnCancelar.PerformClick();
}
```



```
Comando = new MySqlCommand("CREATE TABLE IF NOT EXISTS Caixa " +
    "(Id integer auto_increment primary key, " +
    "idVendaCab int, " +
    "dinheiro decimal(10,2), " +
    "pix decimal(10,2), " +
    "cartao decimal(10,2), " +
    "cheque decimal(10,2), " +
    "boleto decimal(10,2))", Conexao);
Comando.ExecuteNonQuery();
```

```
using MySql.Data.MySqlClient;
using System;
using System.Windows.Forms;

namespace vendas.Models
{
    0 referências
    public class Caixa
    {
        0 referências
        public int Id { get; set; }
        1 referência
        public int idVendaCab { get; set; }
        1 referência
        public double dinheiro { get; set; }
        1 referência
        public double pix { get; set; }
        1 referência
        public double cartao { get; set; }
        1 referência
        public double cheque { get; set; }
        1 referência
        public double boleto { get; set; }
```

```
public void Incluir()
{
    try
    {
        Banco.Conexao.Open();
        Banco.Comando = new MySqlCommand(
            "INSERT INTO Caixa (idVendaCab, dinheiro, pix, cartao, cheque, boleto) " +
            "VALUES (@idVendaCab, @dinheiro, @pix, @cartao, @cheque, @boleto)", Banco.Conexao);
        Banco.Comando.Parameters.AddWithValue("@idVendaCab", idVendaCab);
        Banco.Comando.Parameters.AddWithValue("@dinheiro", dinheiro);
        Banco.Comando.Parameters.AddWithValue("@pix", pix);
        Banco.Comando.Parameters.AddWithValue("@cartao", cartao);
        Banco.Comando.Parameters.AddWithValue("@cheque", cheque);
        Banco.Comando.Parameters.AddWithValue("@boleto", boleto);
        Banco.Comando.ExecuteNonQuery();
        Banco.Conexao.Close();
    }

    catch (Exception e)
    {
        MessageBox.Show(e.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```



# Formulário Caixa

```
using System;
using System.Drawing;
using System.Windows.Forms;
using vendas.Models;

namespace vendas.Views
{
    5 referências
    public partial class FrmCaixa : Form
    {
        double totalVenda, pago, troco, dinheiro,
            cartao, cheque, pix, boleto;

        1 referência
        public FrmCaixa(int idVendaCab, int idCliente, double total, string nome)
        {
            InitializeComponent();

            txtidCliente.Text = idCliente.ToString();
            txtNomeCliente.Text = nome;
            txtValor.Text = total.ToString("C");
            txtVenda.Text = idVendaCab.ToString();
            totalVenda = total;

            calcularTroco();
        }
    }
}
```

```
void ... calcularTroco()
{
    if (txtDinheiro.Text == "") dinheiro = 0;
    else dinheiro = double.Parse(txtDinheiro.Text);

    if (txtCheque.Text == "") cheque = 0;
    else cheque = double.Parse(txtCheque.Text);

    if (txtCartao.Text == "") cartao = 0;
    else cartao = double.Parse(txtCartao.Text);

    if (txtPIX.Text == "") pix = 0;
    else pix = double.Parse(txtPIX.Text);

    if (txtBoleto.Text == "") boleto = 0;
    else boleto = double.Parse(txtBoleto.Text);

    pago = dinheiro + cartao + cheque + pix + boleto;
    troco = pago - totalVenda;
    txtTroco.Text = troco.ToString("C");

    if (troco < 0) txtTroco.ForeColor = Color.Red;
    else txtTroco.ForeColor = Color.Blue;
}
```

```
private void txtBoleto_TextChanged(object sender, EventArgs e)
{
    calcularTroco();
}

private void txtPIX_TextChanged(object sender, EventArgs e)
{
    calcularTroco();
}

private void txtCartao_TextChanged(object sender, EventArgs e)
{
    calcularTroco();
}

private void txtCheque_TextChanged(object sender, EventArgs e)
{
    calcularTroco();
}

private void txtDinheiro_TextChanged(object sender, EventArgs e)
{
    calcularTroco();
}
```

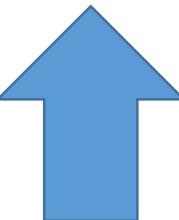
```
private void btnCaixa_Click(object sender, EventArgs e)
{
    if (troco < 0 )
    {
        MessageBox.Show("Valor pago meno que o valor da Venda", "Caixa",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }

    Caixa c = new Caixa()
    {
        idVendaCab = int.Parse(txtVenda.Text),
        dinheiro = dinheiro,
        cartao = cartao,
        cheque = cheque,
        boleto = boleto,
        pix = pix
    };
    c.Incluir();
    Close();
}
```

```
FrmCaixa form =  
    new FrmCaixa(idVenda,  
                 (int)cboClientes.SelectedValue,  
                 total,  
                 cboClientes.Text);  
    form.ShowDialog();
```

```
btnCancelar.PerformClick();
```

```
}
```



Esse código deverá ser feito no Botão  
gravar do Formulário de VENDAS



Consulta de Venda de Produtos

|   | Venda      | Data                      | Cliente | Quantidade | Valor |
|---|------------|---------------------------|---------|------------|-------|
| 3 | 22/06/2021 | ANTONIO APARECIDO RIBEIRO | 5       | 29,00      |       |
| 4 | 22/06/2021 | ANTONIO APARECIDO RIBEIRO | 2       | 29,00      |       |
| 5 | 22/06/2021 | ANTONIO APARECIDO RIBEIRO | 10      | 29,00      |       |
| 7 | 22/06/2021 | ANTONIO APARECIDO RIBEIRO | 1       | 29,00      |       |
| 8 | 22/06/2021 | ANTONIO APARECIDO RIBEIRO | 10      | 29,00      |       |
| 9 | 22/06/2021 | ANTONIO APARECIDO RIBEIRO | 10      | 29,00      |       |

Total Vendido: **38**

Produto: ARROZ TIO JOAO 5KG

Periodo: 22/06/2021 à: 22/06/2021

Consultar

Fechar

Labels and arrows:

- cboProdutos
- dtpInicio
- dtpFinal
- btnConsultar
- dgvConsulta
- lblTotal
- btnFechar



# X Consulta / Gráfico / Relatório

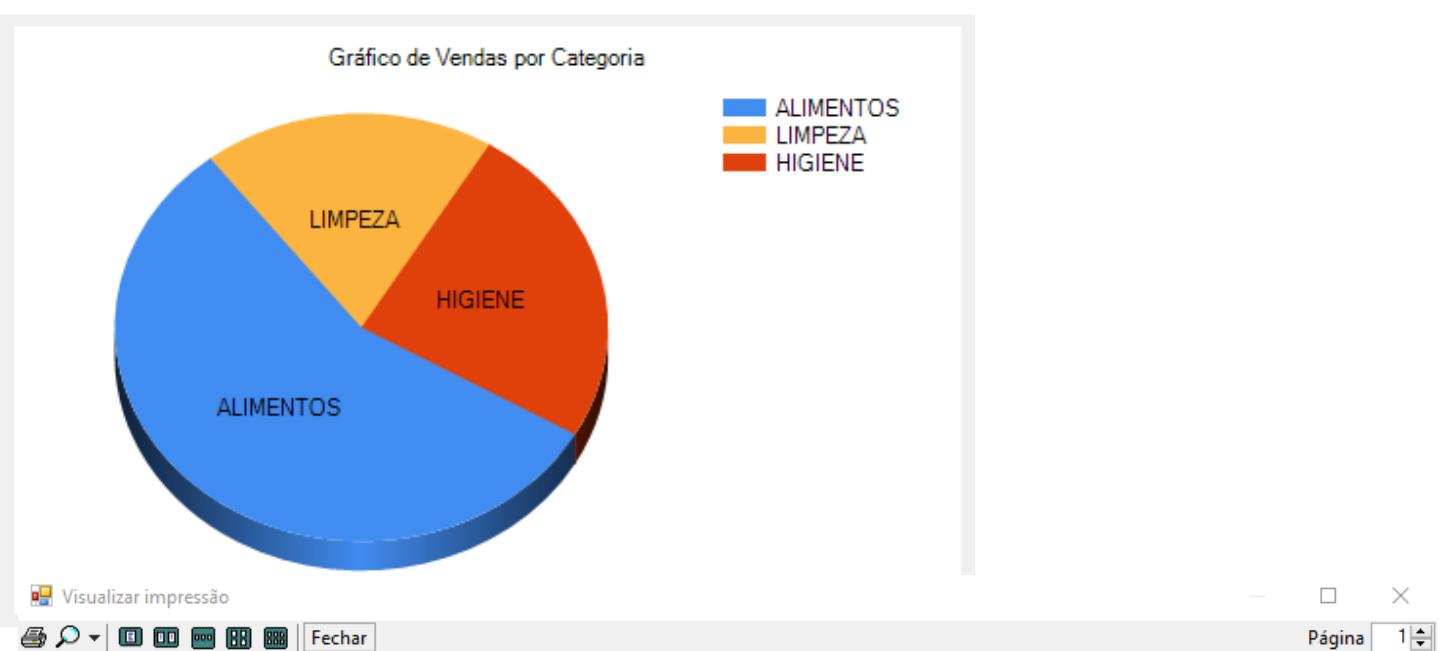
110

Consulta de Vendas por Categoria

Periodo: 01/05/2021 à: 24/06/2021 Consultar

|   | Categoria | Quantidade |
|---|-----------|------------|
| ▶ | ALIMENTOS | 41         |
|   | LIMPEZA   | 14         |
|   | HIGIENE   | 18         |

Gráfico Imprimir Fechar



## Relatório de Vendas

| Categoria | Quantidade |
|-----------|------------|
| ALIMENTOS | 41         |
| LIMPEZA   | 14         |
| HIGIENE   | 18         |

# Consulta / Gráfico / Relatório

111

