

Desenvolvimento de Sistemas

Professor Everton Sette

06 Integração com Banco de Dados





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

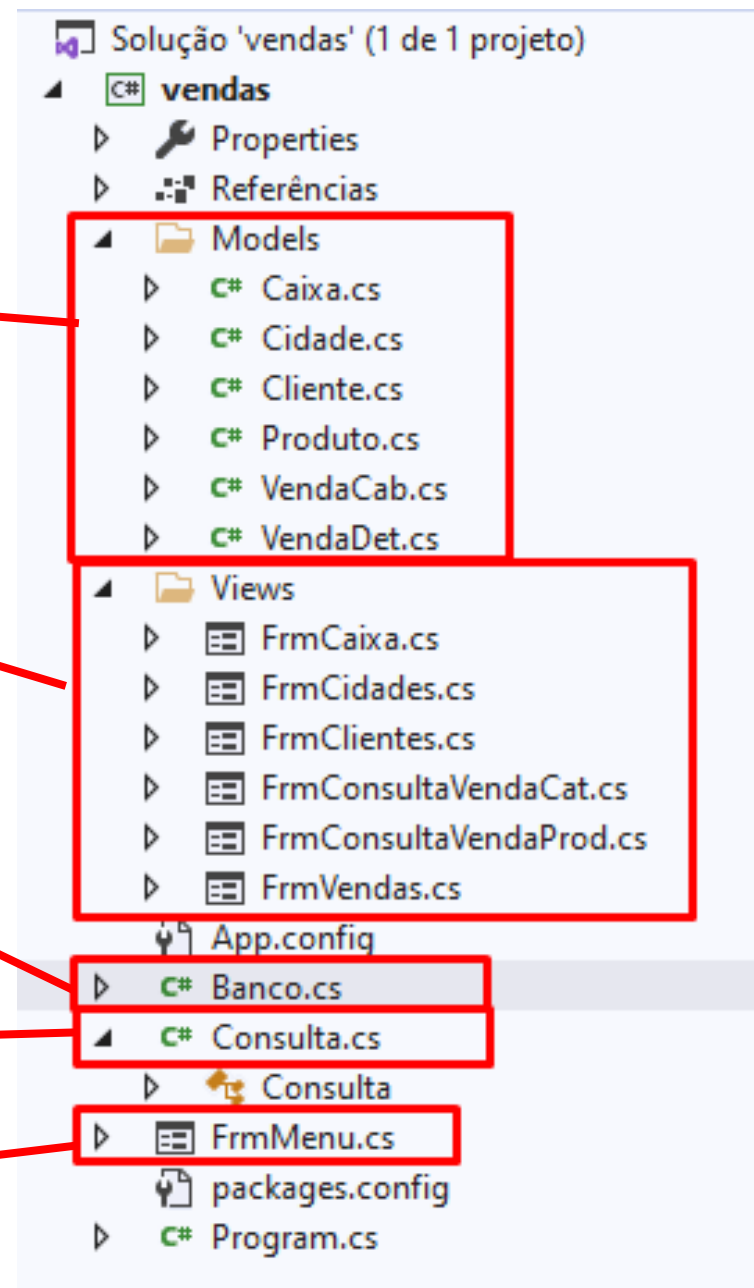
**Pasta contendo todas as
classes de modelagem
(atributos, métodos)**

**Pasta contendo todos os
formulários do sistema**

**Classe de conexão com o
banco de dados**

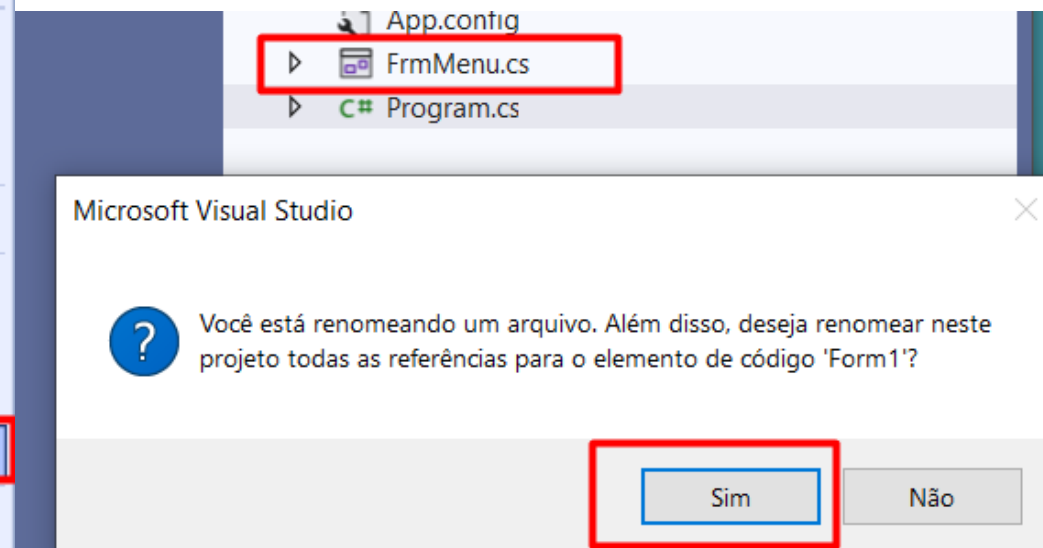
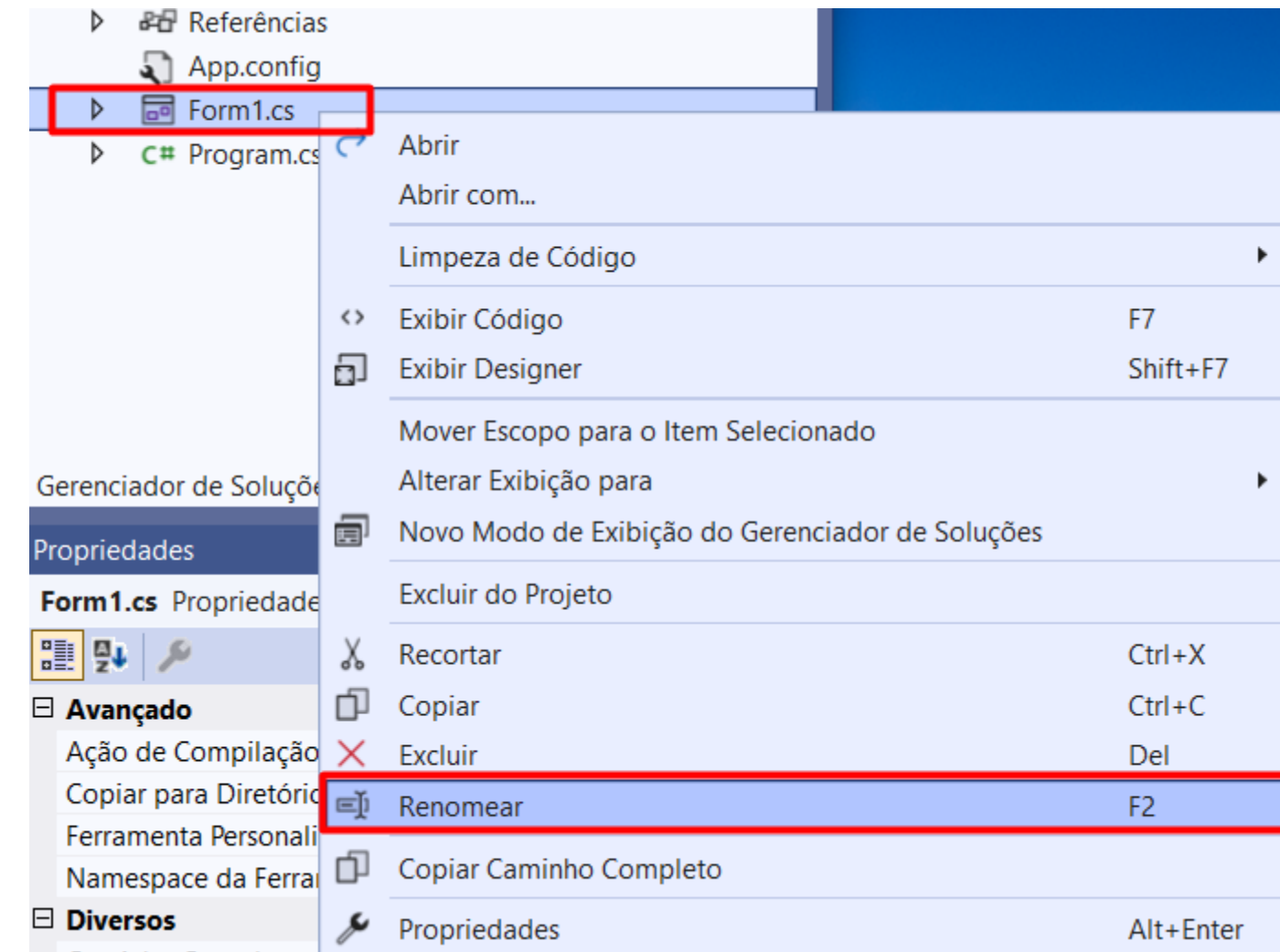
**Classe contendo todas as consultas
para relatórios e gráficos**

Menu principal do sistema



Renomeando um Formulário

05

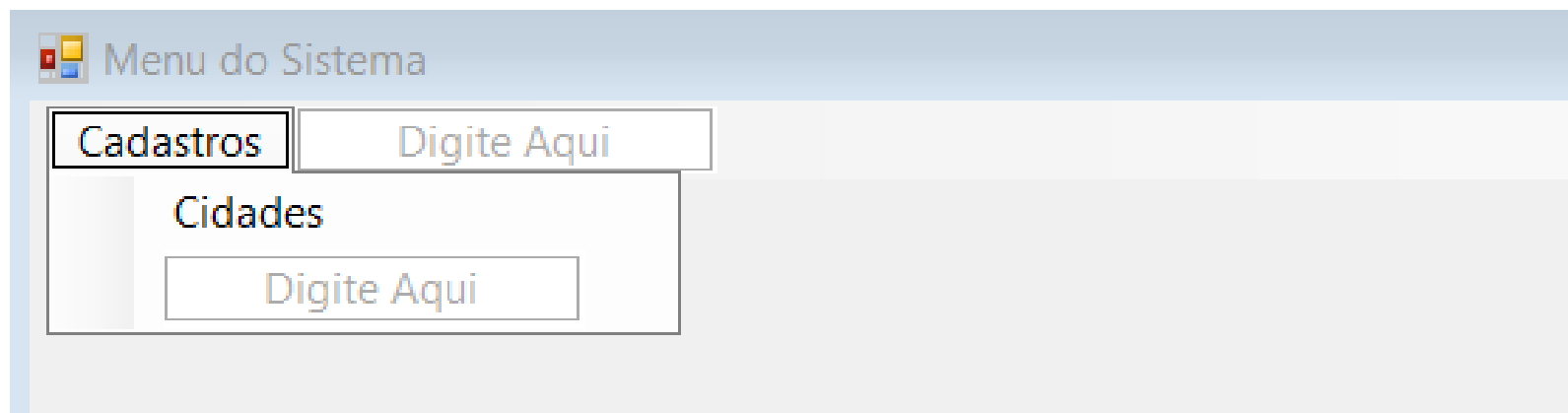
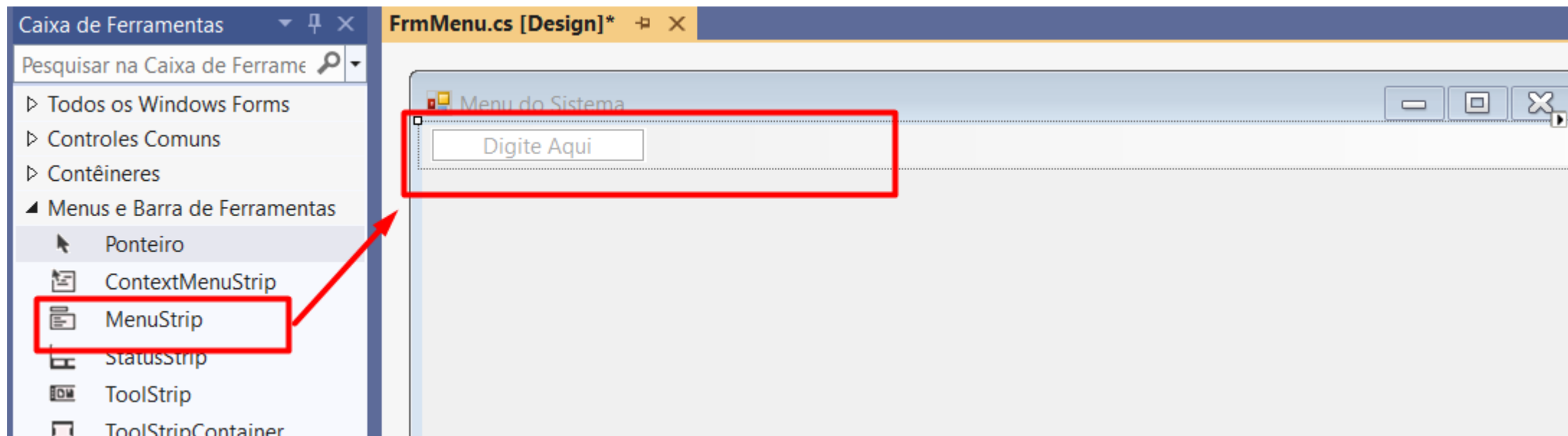


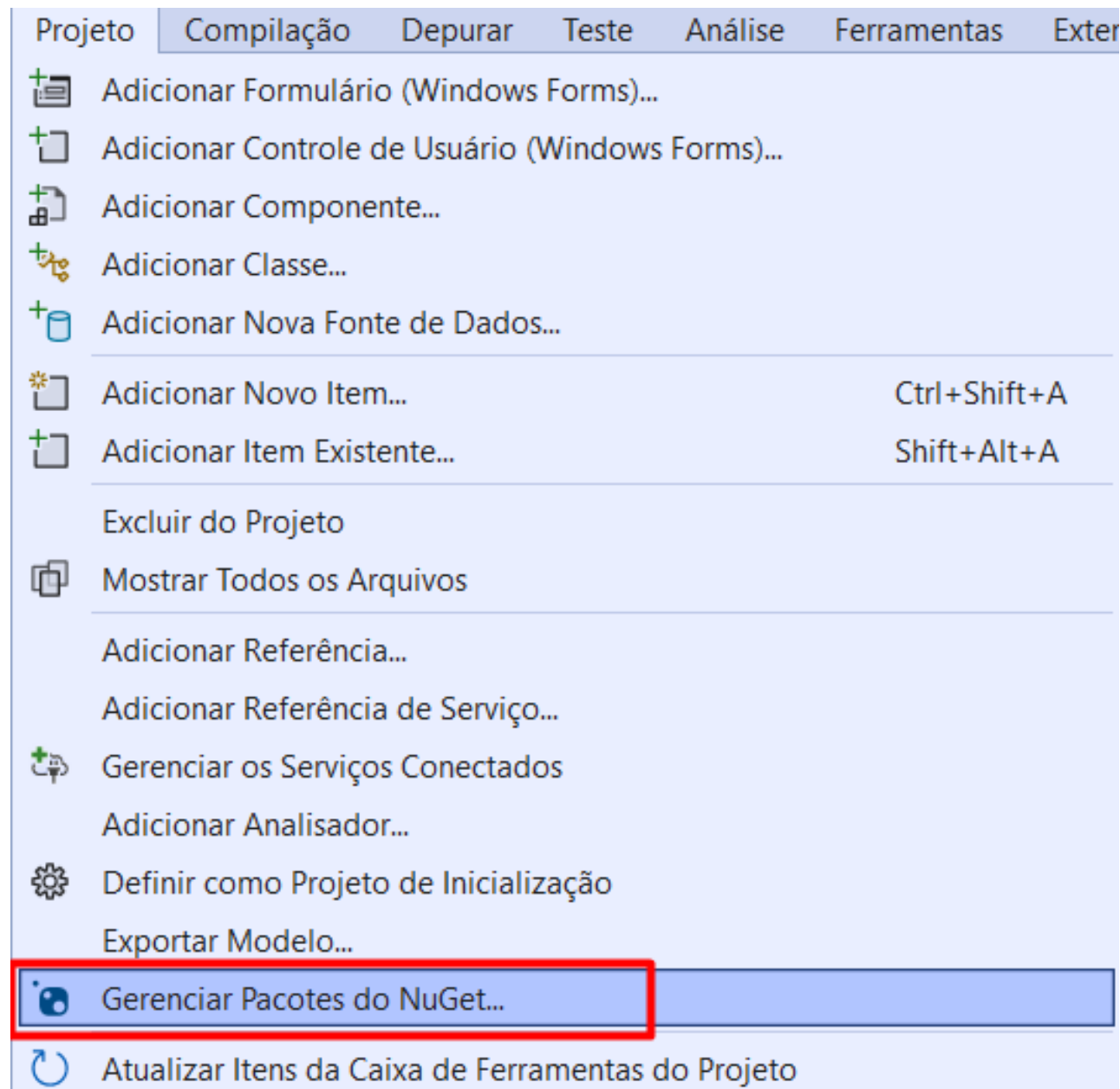
FrmMenu System.Windows.Forms.Form

MainMenuStrip	menuStrip1
MaximizeBox	True
MaximumSize	0; 0
MinimizeBox	True
MinimumSize	0; 0
Opacity	100%
Padding	0; 0; 0; 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
Size	818; 497
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	Menu do Sistema
TopMost	False
TransparencyKey	<input type="checkbox"/>
UseWaitCursor	False
WindowState	Maximized

Altera o título do Formulário

Inicializa o Form Maximizado





NuGet: vendas* FrmMenu.cs [Design]*

Procurar Instalado Atualizações

mysql ☐ Incluir pré-lançamento

Origem do pacote: nuget.org

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

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

Versão: Estável mais recente 8 **Instalar**

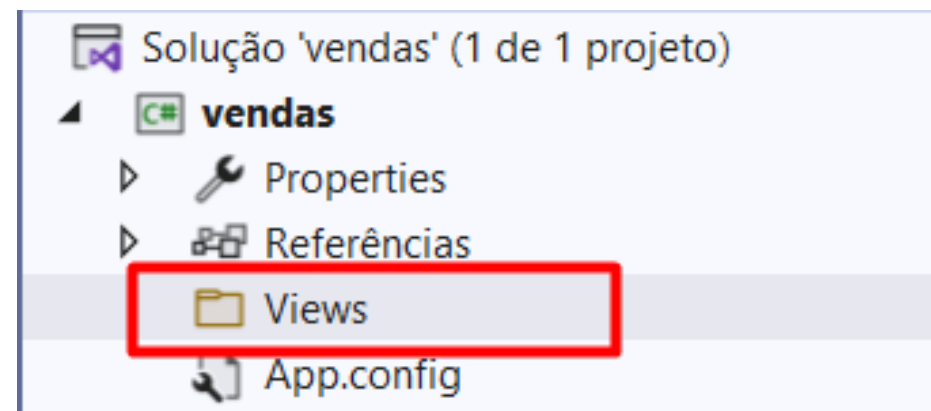
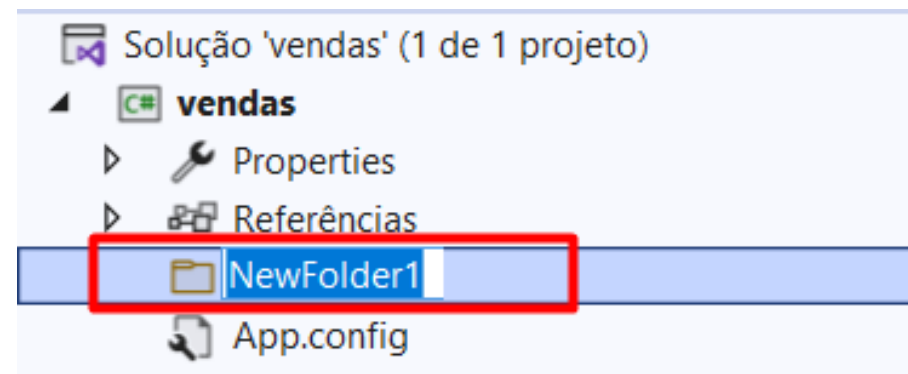
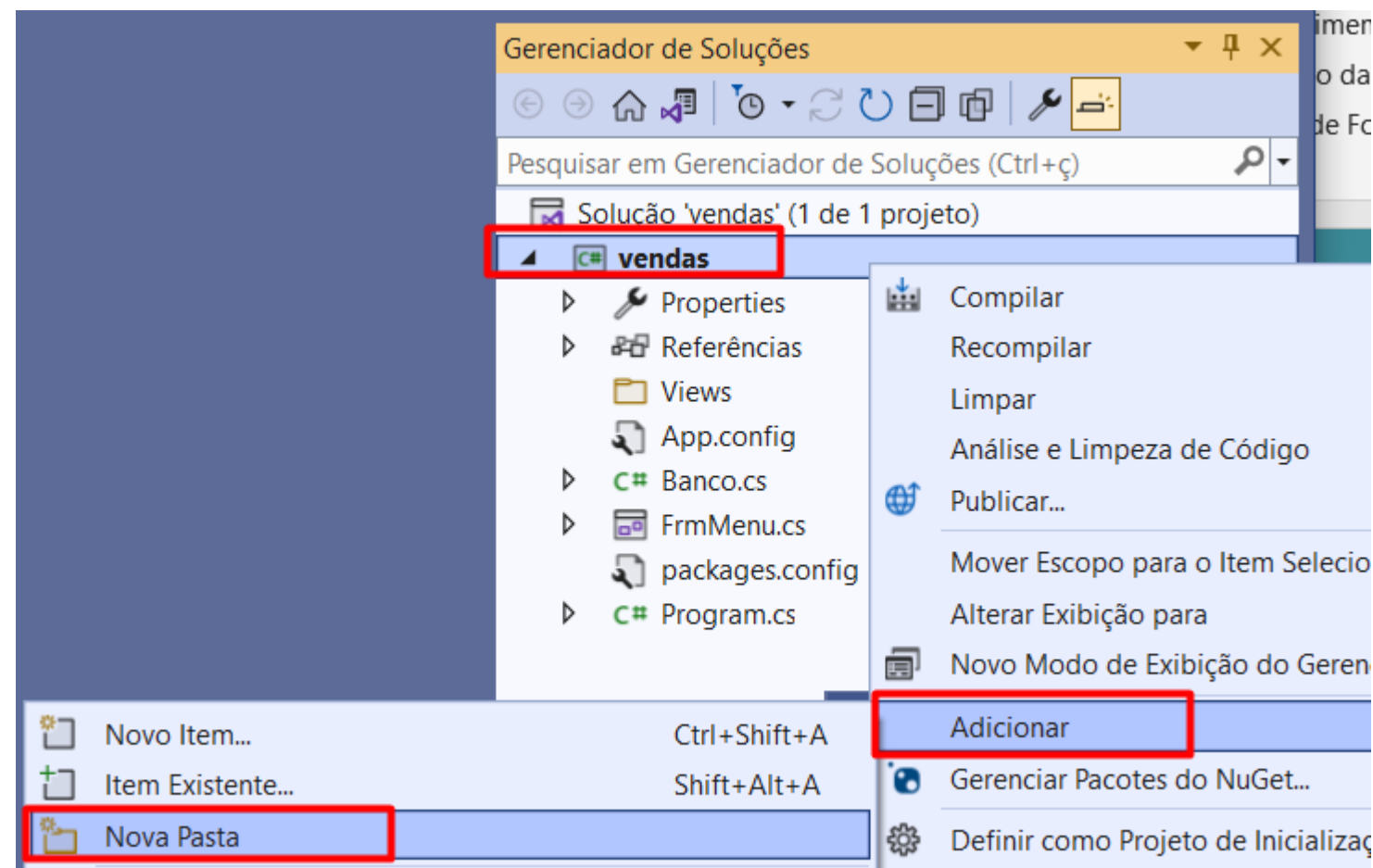
Opções

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

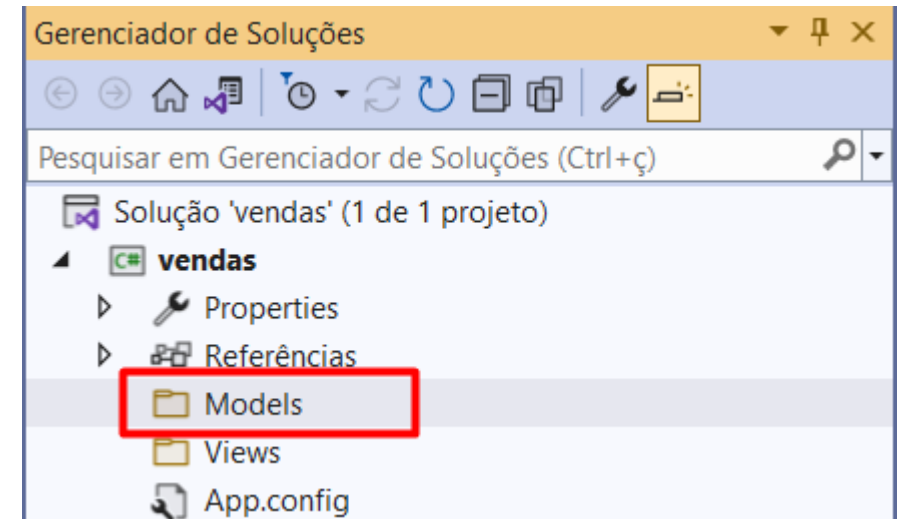
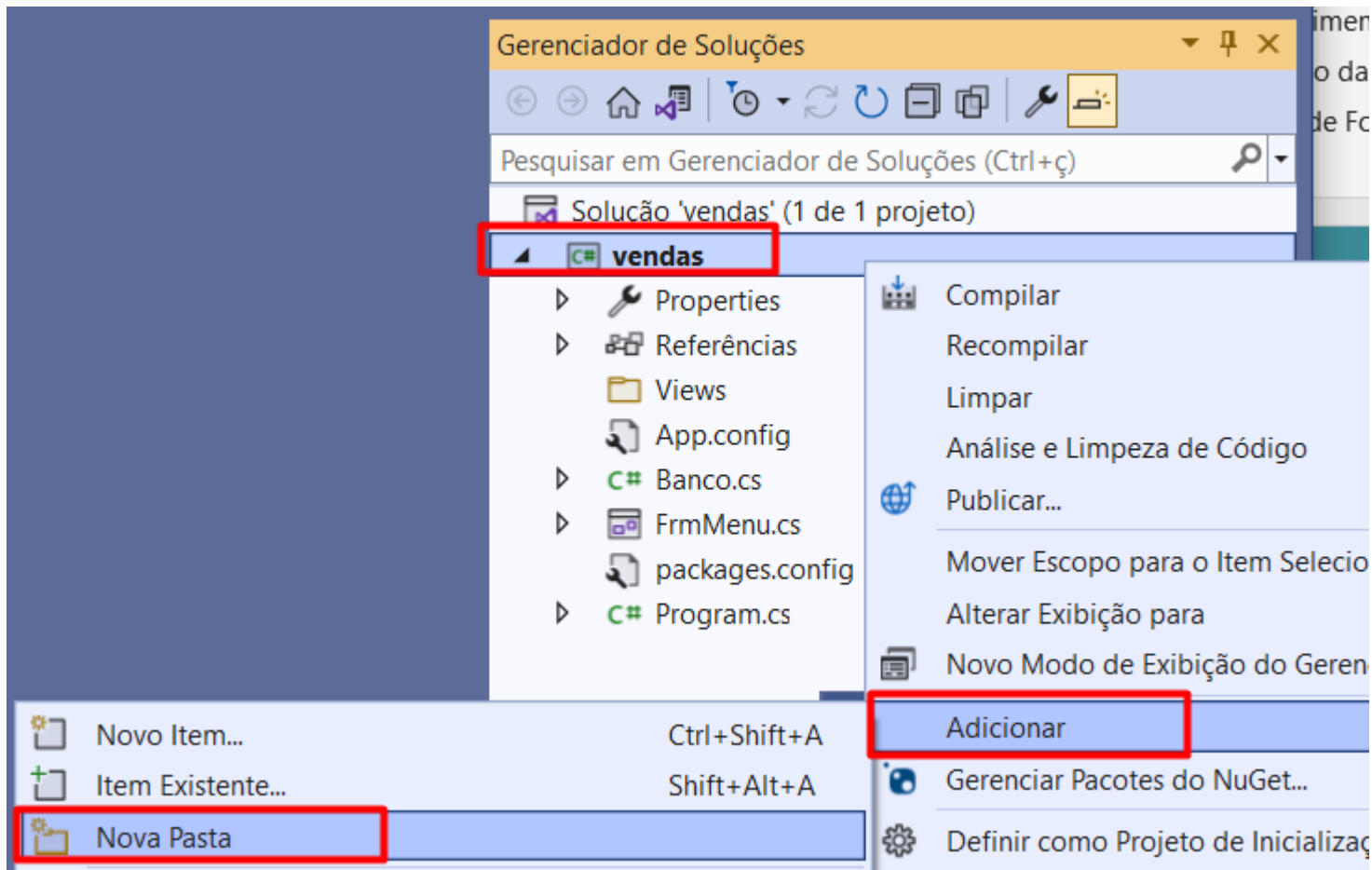
Adicionando uma pasta para nossos Forms

10



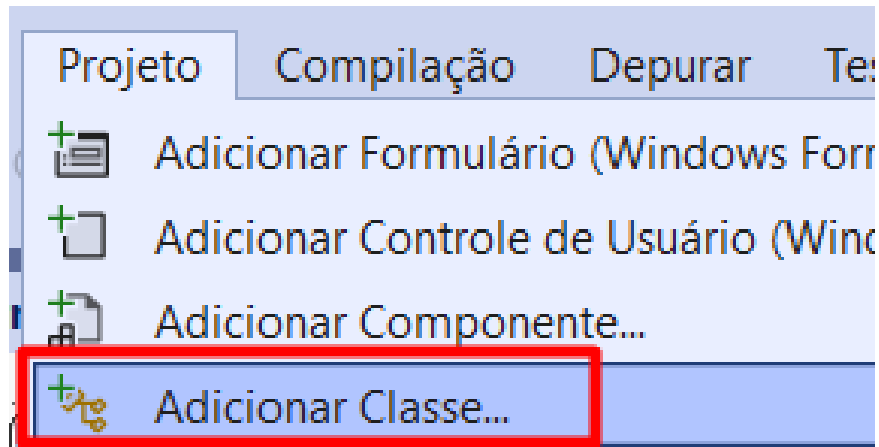
Adicionar pasta para classes de modelagem

11

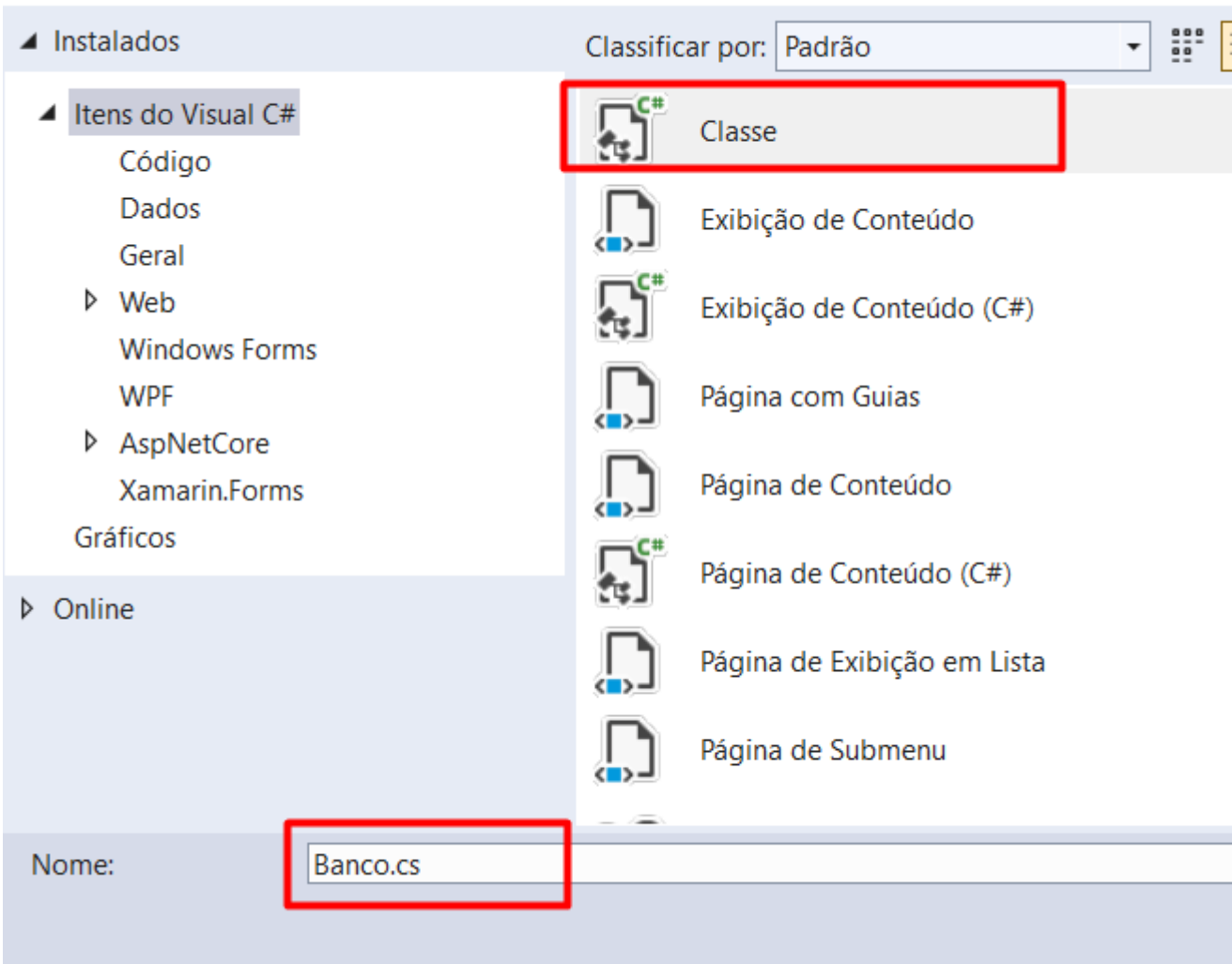


Criar classe para conexão com o banco de dados

12



Adicionar Novo Item - vendas



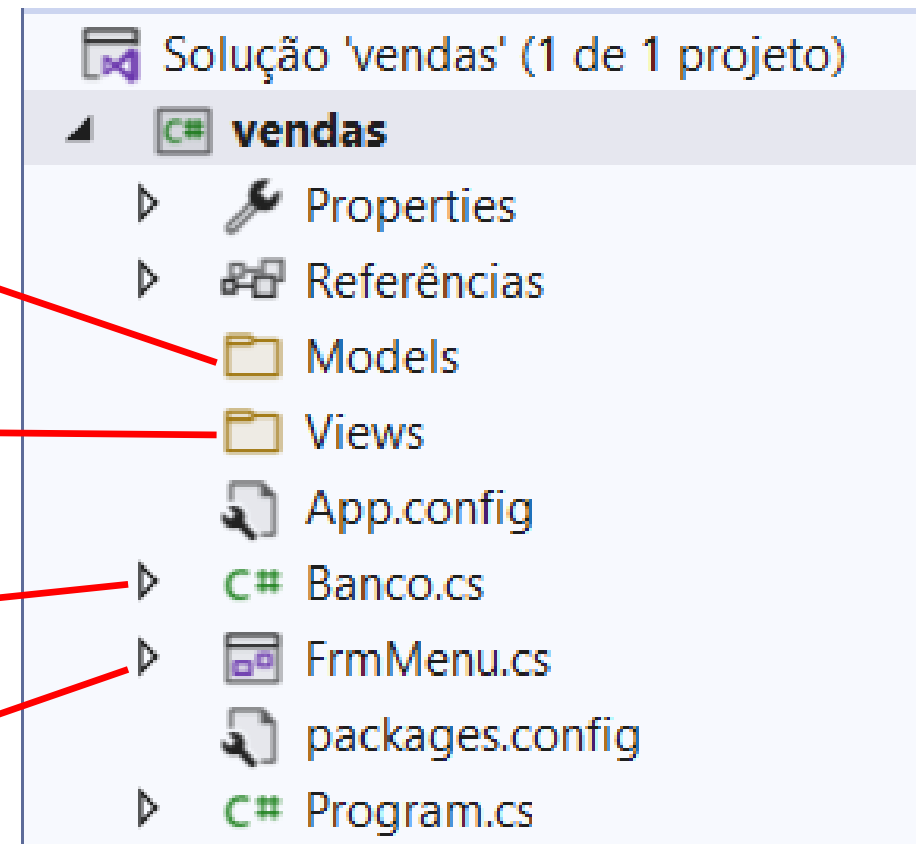
```
namespace vendas
{
    0 referências
    public class Banco
    {
    }
}
```

**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 publicas 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();

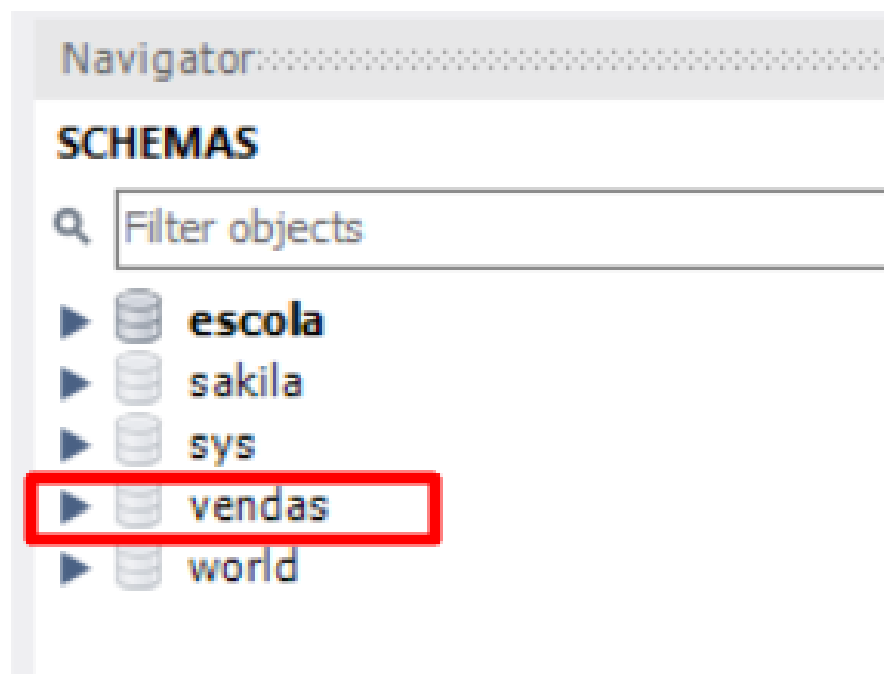
        // Informa a instrução SQL
        Comando = new MySqlCommand("CREATE DATABASE IF NOT EXISTS vendas; USE vendas", Conexao);
        // Executa a Query no MySQL (Raio do Workbench)
        Comando.ExecuteNonQuery();

        // 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

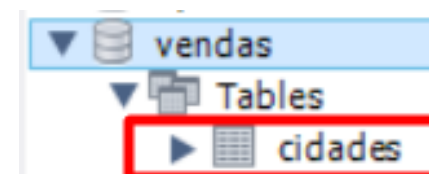
1 referência

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





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



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

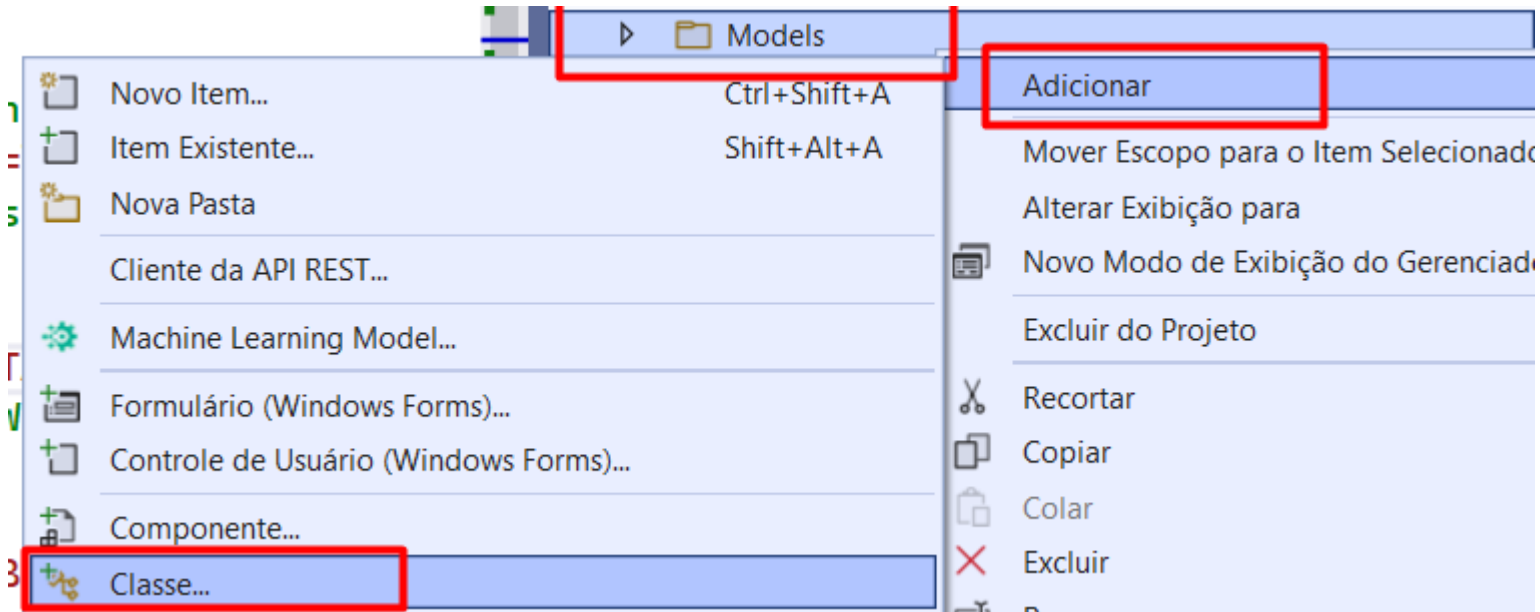
        // Informa a instrução SQL
        Comando = new MySqlCommand("CREATE DATABASE IF NOT EXISTS vendas; USE vendas", Conexao);
        // Executa a Query no MySQL (Raio do Workbench)
        Comando.ExecuteNonQuery();

        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);
    }
}
```

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

Nome: Cidade.cs

```
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; }
```

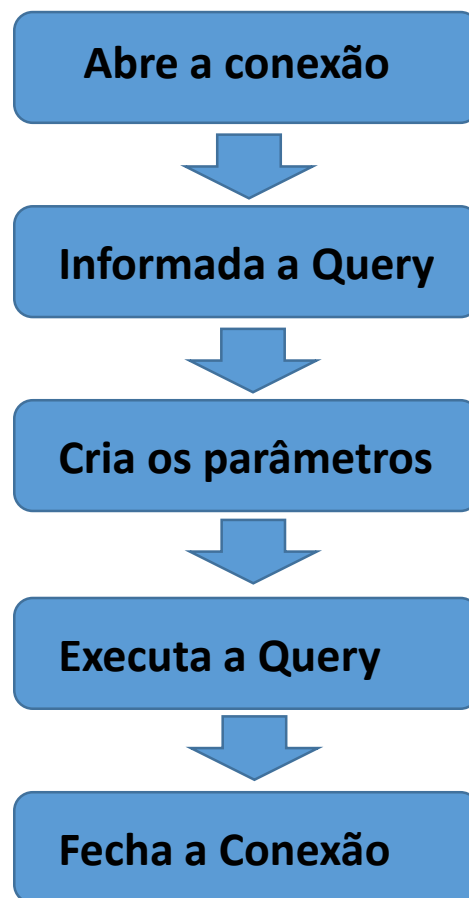
4 referências

```
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  
{
```

4 referências

```
public int id { get; set; }
```

6 referências

```
public string nome { get; set; }
```

4 referências

```
public string uf { get; set; }
```

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

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

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



```
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

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

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

	id	nome	uf
▶	1	JAU	SP
	2	BARIRI	SP
•	NULL	NULL	NULL

MySQLDataAdapter

DataTable

Traz a Consulta SQL em forma de tabela para a memória

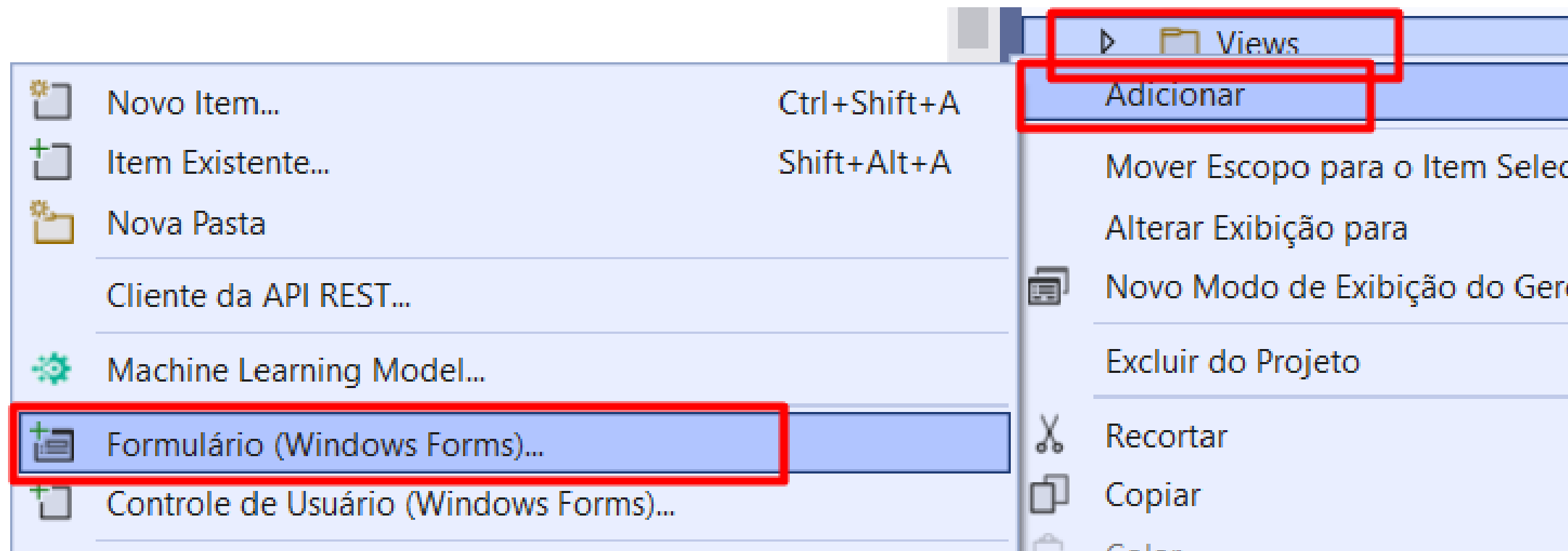
```
Adaptador = new MySqlDataAdapter(comando);
```



	id	nome	uf
▶	2	BARIRI	SP
	1	JAU	SP

DataGridView – Controle com a propriedade DataSource

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



Nome:

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

Propriedades

FrmCidades System.Windows.Forms.Form

BackgroundImageLayout Tile

CancelButton (nenhum)

CausesValidation True

ContextMenuStrip (nenhum)

ControlBox False

Cursor Default

ShowIcon True

ShowInTaskbar True

Size 562; 409

SizeGripStyle Auto

StartPosition CenterScreen

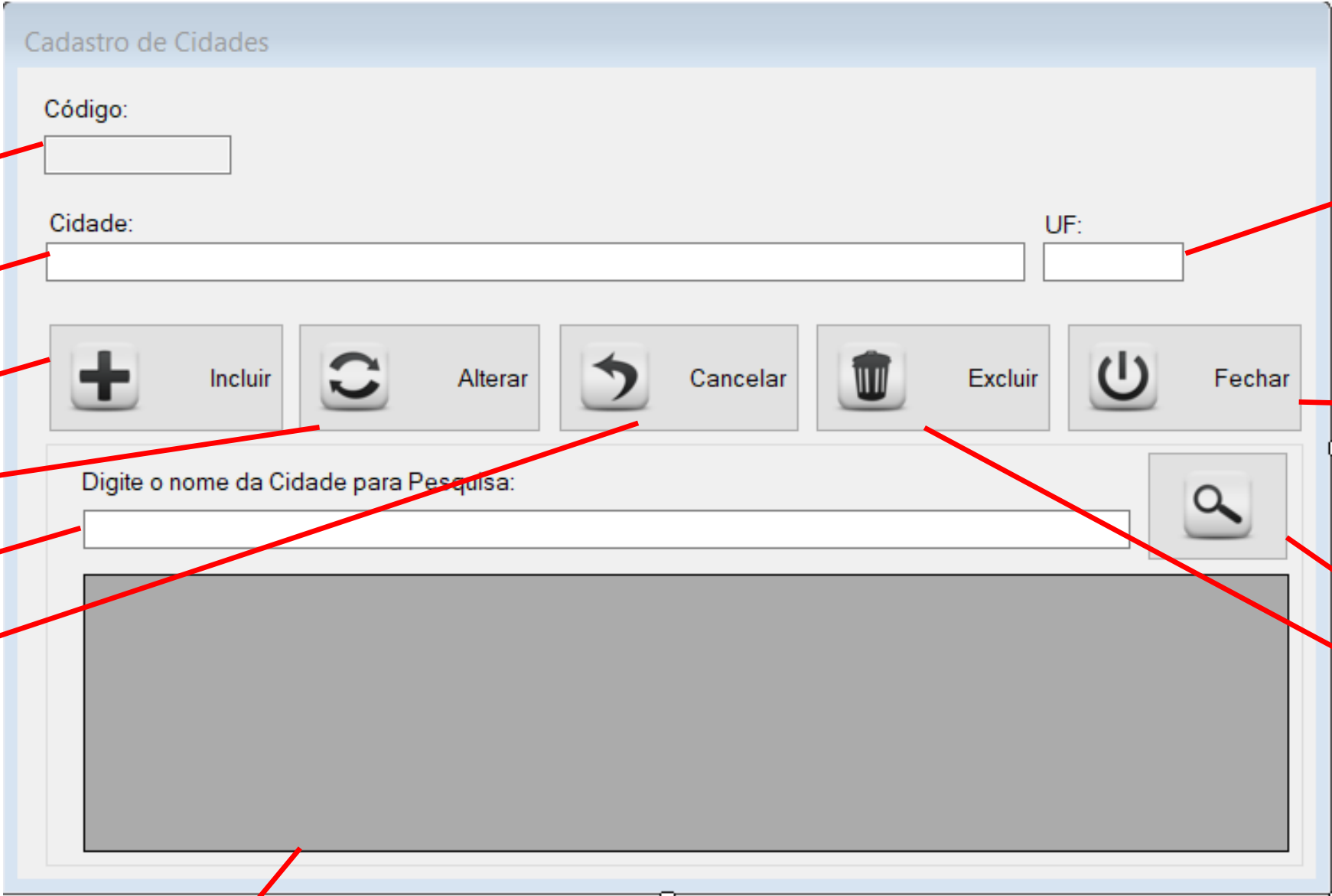
Tag

Text Cadastro de Cidades

TopMost False

TransparencyKey ☐

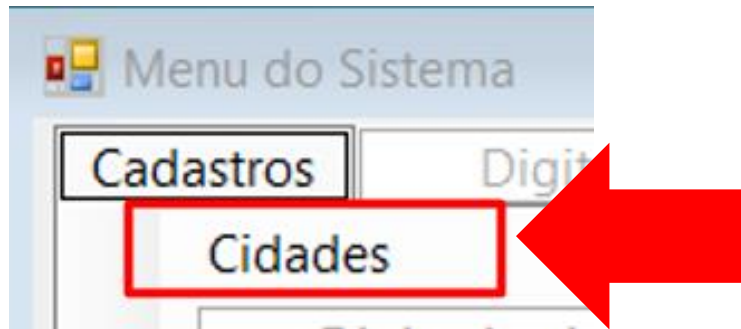
UseWaitCursor False



The image shows a software form titled "Cadastro de Cidades". It includes several input fields and a set of action buttons. Red arrows point from text labels to specific UI elements:

- txtID** points to the "Código:" input field.
- txtNome** points to the "Cidade:" input field.
- txtUF** points to the "UF:" input field.
- btnIncluir** points to the "Incluir" button (plus icon).
- btnCancelar** points to the "Cancelar" button (undo icon).
- btnExcluir** points to the "Excluir" button (trash icon).
- btnFechar** points to the "Fechar" button (power icon).
- dgvCidades** points to the large gray rectangular area at the bottom, which is a data grid.

Other visible elements include a search bar labeled "Digite o nome da Cidade para Pesquisa:" with a magnifying glass icon, and a toolbar with buttons labeled "Incluir", "Alterar", "Cancelar", "Excluir", and "Fechar".



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)
{
    c = new Cidade()
    {
        nome = pesquisa
    };
    DgvCidades.DataSource = c.Consultar();
}
```






1 referência


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

Cadastro de Cidades

Código:

Cidade: UF:

 Incluir  Alterar  Cancelar  Excluir  Fechar

Digite o nome da Cidade para Pesquisa: 

	id	nome	uf
--	----	------	----

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("");
}
```

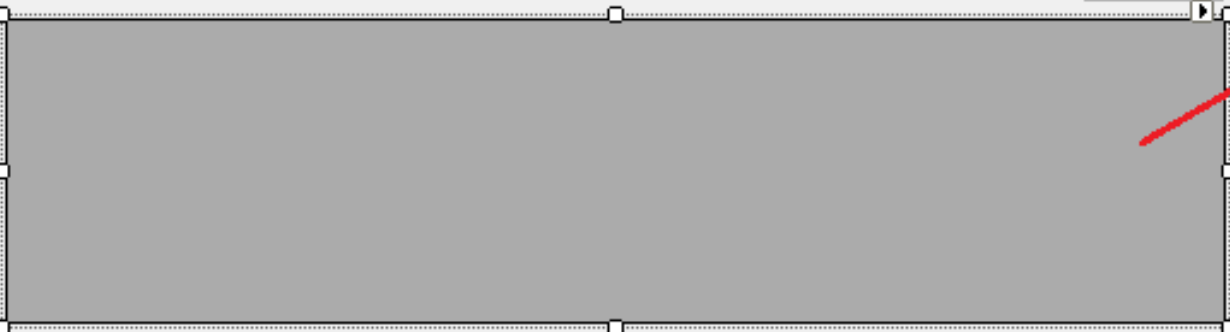

Cadastro de Cidades

Código:

Cidade: UF:


 Incluir  Alterar  Cancelar  Excluir  Fechar

Digite o nome da Cidade para Pesquisa: 



Propriedades

DgvCidades System.Windows.Forms.DataGridView



- BackgroundColorChanged
- BindingContextChanged
- BorderStyleChanged
- CancelRowEdit
- CausesValidationChanged
- CellBeginEdit
- CellBorderStyleChanged
- CellClick **DgvCidades_CellClick**
- CellContentClick
- CellContentDoubleClick

[Editar Colunas...](#) [Adicionar Coluna...](#)

CellContentClick
Ocorre quando o usuário clica no conteúdo dentro de uma célula.

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
};
```



```
c.Alterar();

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

```
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();
}
```