MVC e C#

Aula 2

Etec
Pedro Ferreira
Alves
Mogi Mirim

Domínio: Escola

Apresentaremos um exemplo de cadastro de alunos em um domínio Escola.

O modelo utilizado será a divisão em camadas, ao todo, uma para representar o Modelo (Model), uma para representar o Banco (DAO), uma para representar os dados de visão do aluno (VIEW – usaremos formulário Windows.

Dê o nome para este projeto de WindowsBanco, por representar nossa primeira aplicação com Banco de Dados.

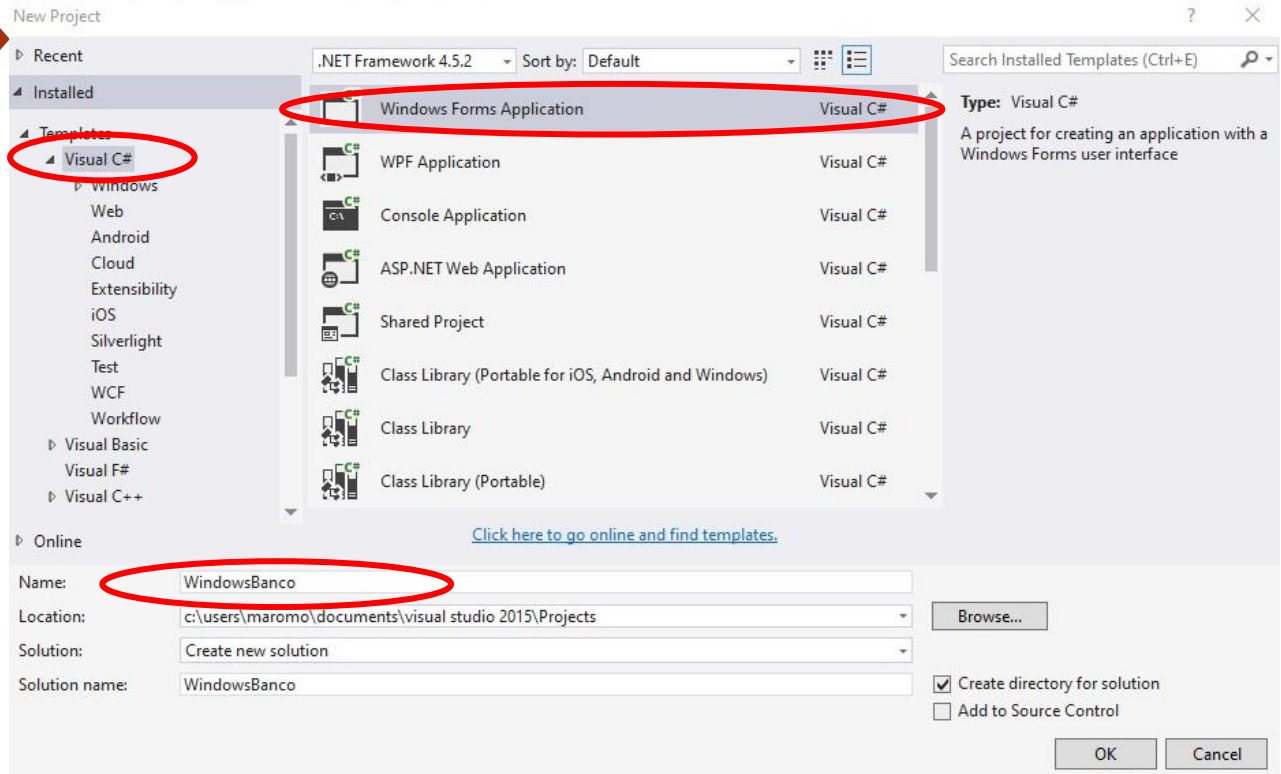
Objeto

MODEL: classes que representam as entidades do domínio, neste exemplo teremos o arquivo Aluno.cs

VIEW: classes que representam nossos formulário de visualização de dados e cadastros. Neste exemplo teremos dois, o primeiro para representar o aluno (FrmAluno) e depois criaremos um Formulário principal (MdiAplicacao) para acessar os formulários deste projeto.

DAO: Aqui teremos nossa interface IDAO genérica para determinar as operações básicas de persistências, teremos a classe AlunoDAO para persistir os dados dos aluno e uma chamada DbConnect para estabelecer a comunicação com o banco de dados criado. Escola.mdf

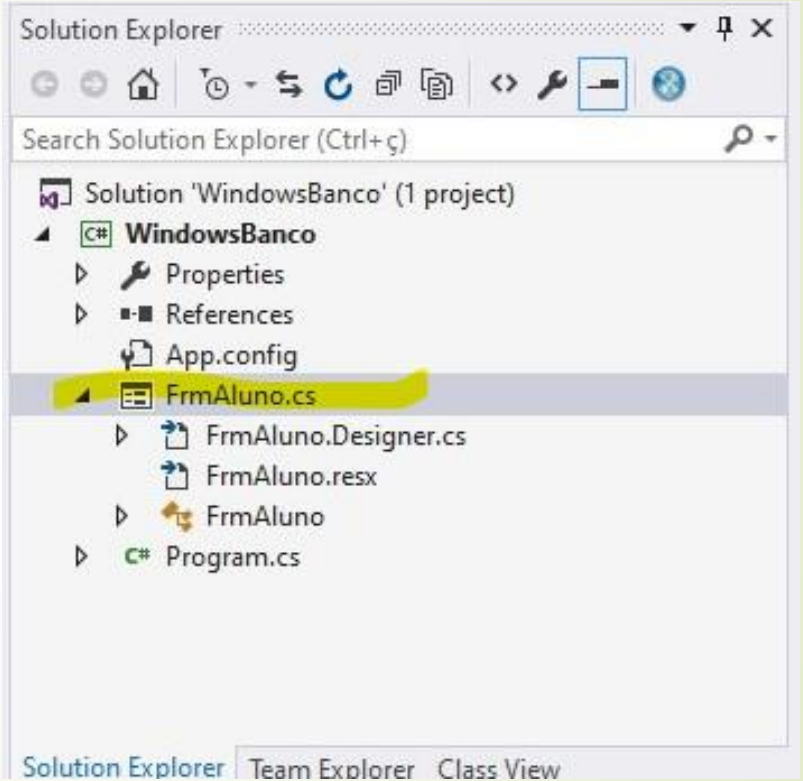
Criando Projeto



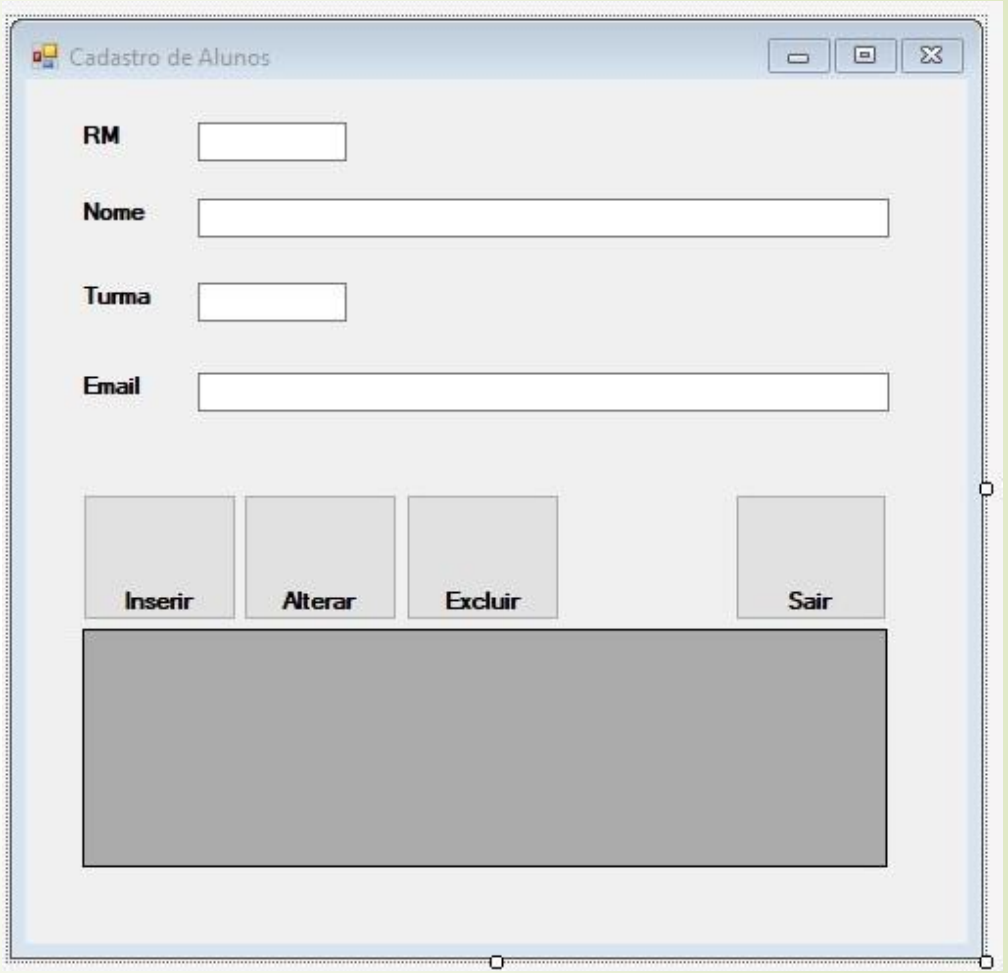
Visão

Formulários

Renomei o formulário Form1 para FrmAluno.cs

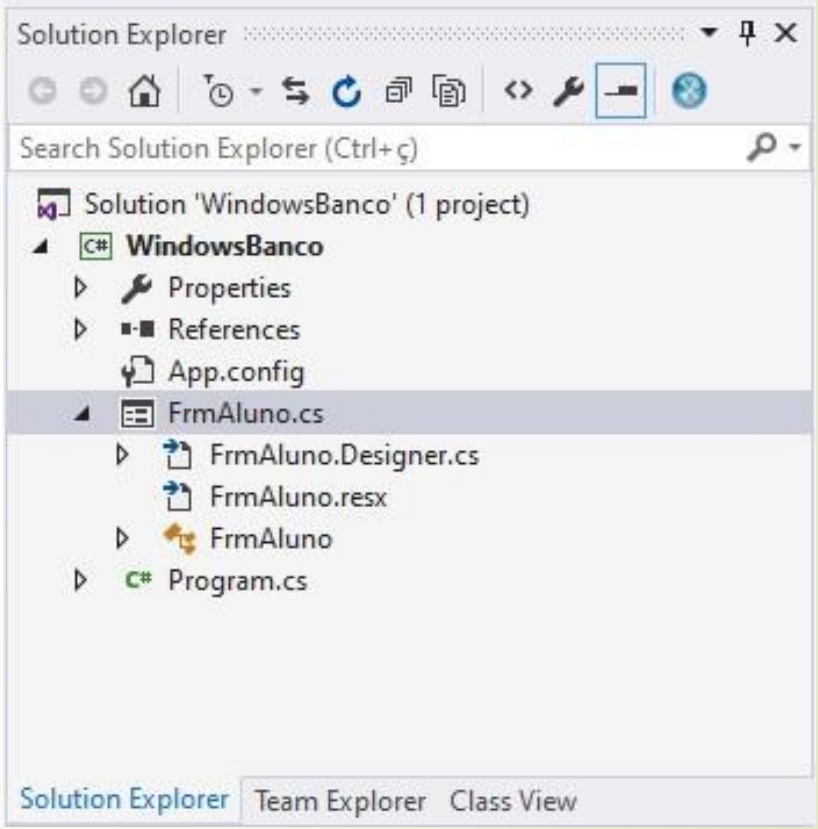


Aparência que você deve deixar o FrmAluno.cs



Nota

- Com os passos anteriores temos o nosso formulário que compõe a camada de visão, não há necessidade de organizar os formulário em uma pasta com o nome VIEW, deixe na raiz mesmo.
- Como está a janela de projeto neste momento.

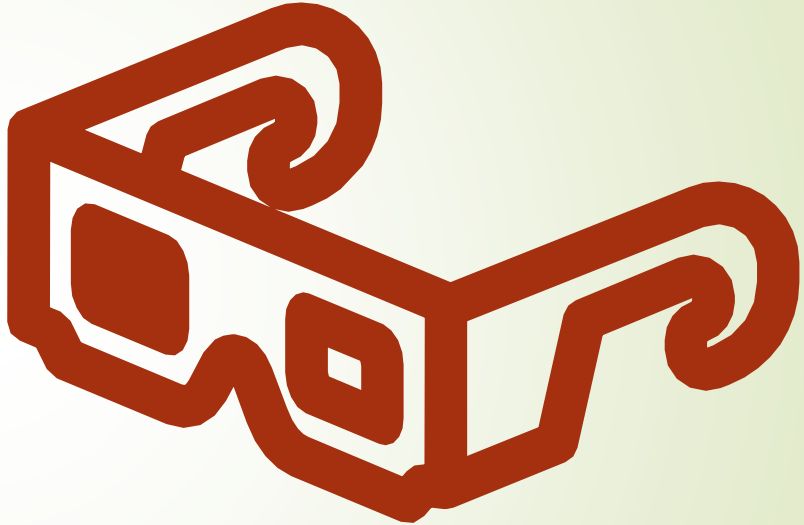


Renomeie o nome de cada elemento conforme tabela abaixo:

Elemento (Name)	Propriedade a alterar	Valor da propriedade
textBox1	**Name**	**txtRM**
textBox2	**Name**	**txtNome**
textBox3	**Name**	**txtTurma**
textBox4	**Name**	**txtEmail**
button1	**Name** **TextAlign**	**btnInserir** **BottomCenter**
button2	**Name** **TextAlign**	**btnAlterar** **BottomCenter**
button3	**Name** **TextAlign**	**btnExcluir** **BottomCenter**
button4	**Name** **TextAlign**	**btnSair** **BottomCenter**
dataGriView1	**Name**	**dataGridAlunos**

Nota

- Agora terminado o Desing (VIEW – visão do usuário), deixaremos o formulário por hora.
- Iniciaremos a modelagem do nosso modelo na próxima seção (próximo slide)



Model

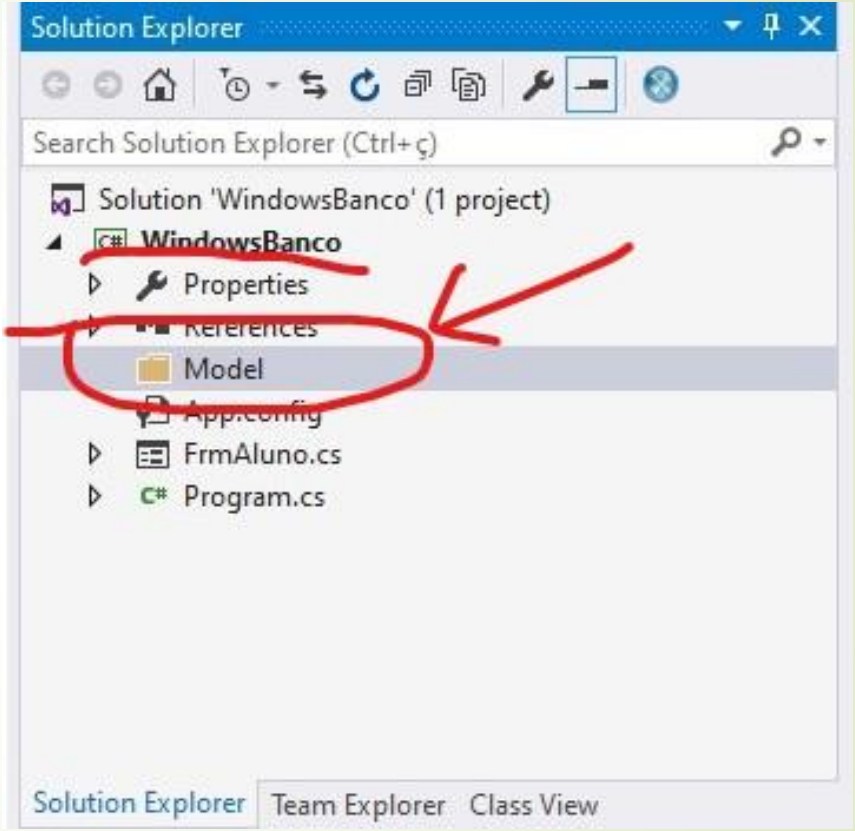
Modelo

Camada Model

- Um modelo armazena dados e notifica suas visões e persistências associadas quando há uma mudança em seu estado. Estas notificações permitem que as visões produzam saídas atualizadas e alterem o conjunto de comandos disponíveis.

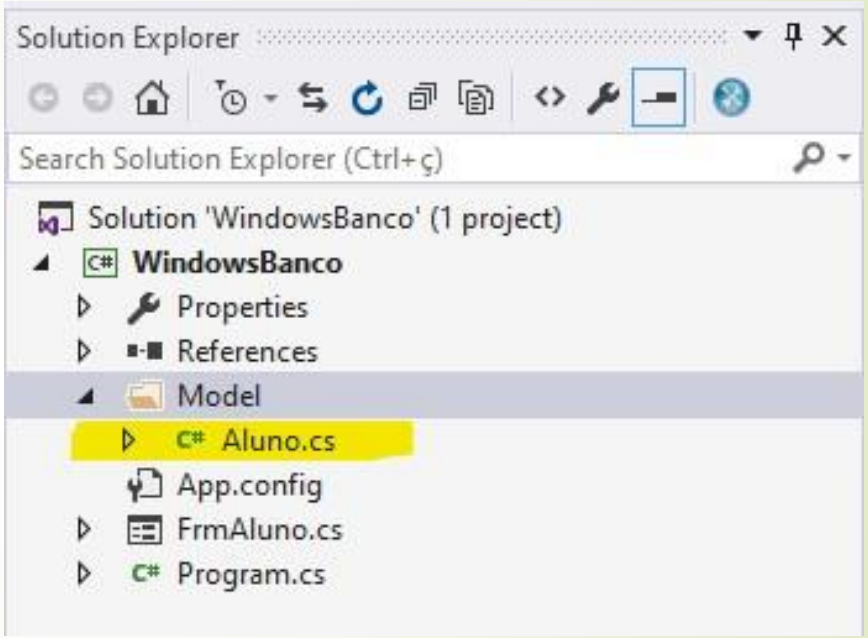
Crie uma pasta Model

- Botão direito no nome do Projeto WindowsBanco e em seguida >> Add >> New folder.
- Dê o nome de **Model.**



Modelo: Aluno

```
namespace WindowsBanco.Model
{
    public class Aluno
    {
        public int RM { get; set; }
        public String Nome { get; set; }
        public String Turma { get; set; }
        public String Email { get; set; }

    }
}
```



Adicione dentro da pasta Model

Model - Finalizado

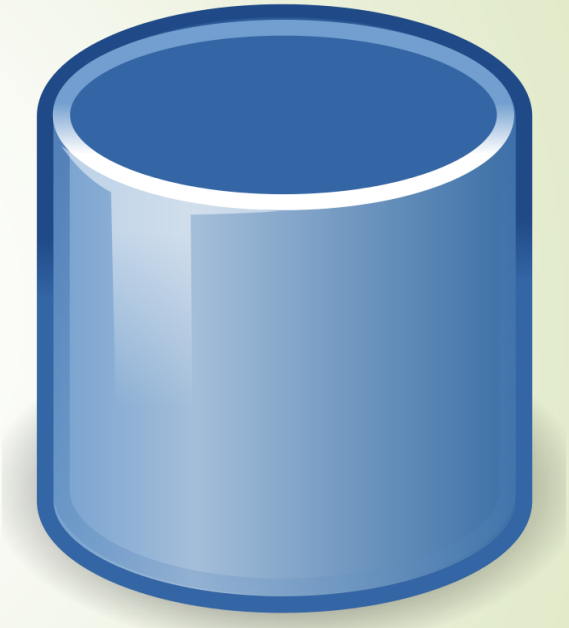
- Neste momento tudo o que precisamos para representar nossa entidade está aqui. Ela será acessada pela camada visão que assumirá o controle de um objeto Modelo, persistindo no banco acessando a camada DAO.
- Camada DAO é a próxima a ser criada, na próxima seção.

DAO

Camada de persistência com banco

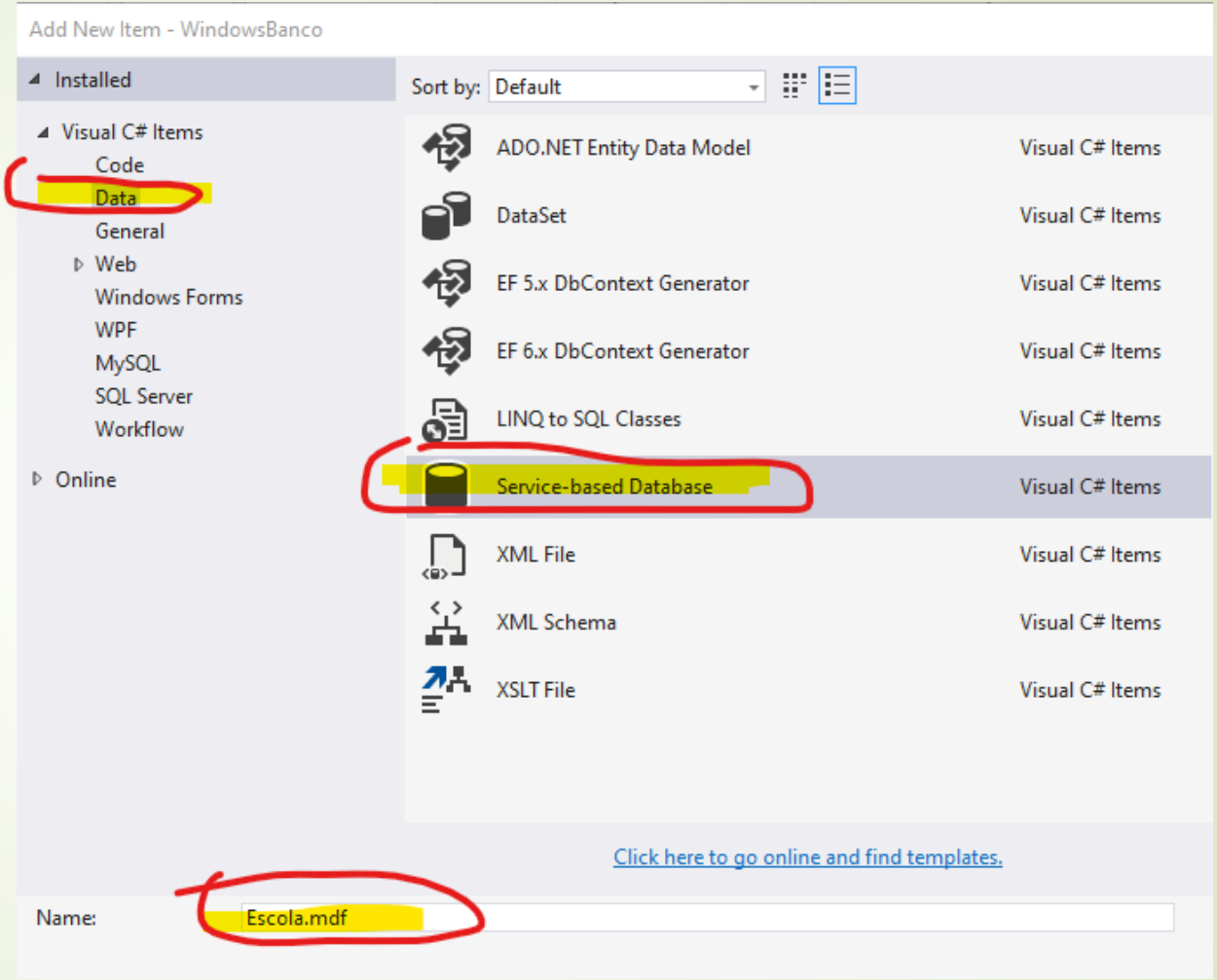
Camada DAO

- Camada DAO (Data Access Object) é a camada responsável por encapsular todas as 'idas' ao banco de dados.
- No nosso exemplo tem a classe que representa uma entidade Aluno (aluno.cs), você deve ter então uma classe como ‘AlunoDAO' que conterá alguns métodos responsáveis por realizar operações de banco de dados para a entidade Aluno.
- Como isto é um padrão que se repete para todas as entidades do negócio, poderíamos adicionar uma interface genérica para facilitar a codificação, contendo métodos abstratos como 'inserir' que seria responsável por persistir um dado no banco de dados, 'listar' que traria todas os dados persistidas no banco de dados, excluir' que apagaria um dado do banco, 'atualizar', e etc...

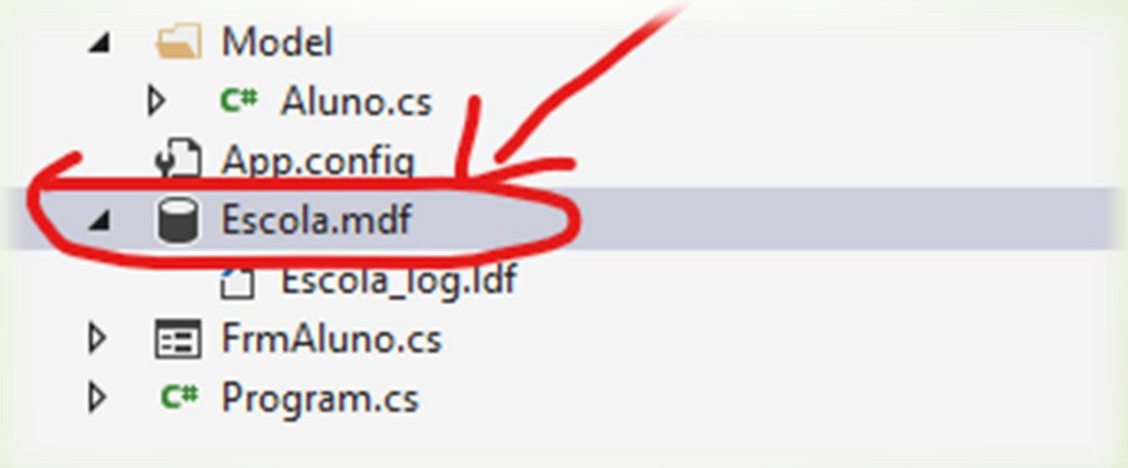


Criando o Banco de Dados

- Clique no botão direito do Mouse e escolha Add >> New Item, escolha a categoria Data e em Seguida Service-based Database.
- Banco de Dados: Escola.mdf
- Este processo pode ser um pouco lento.

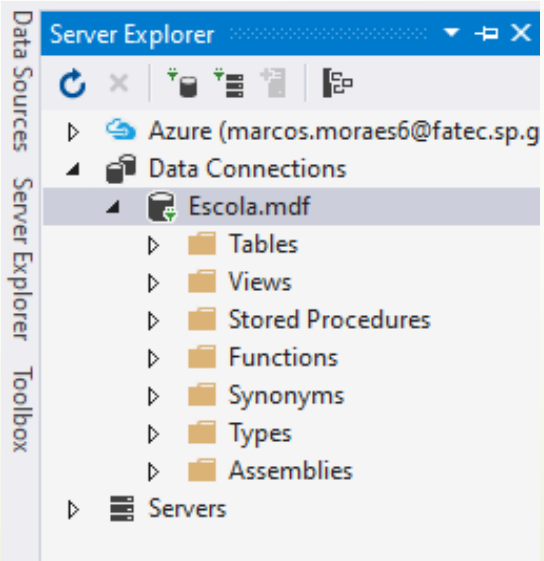


Resultado até o momento



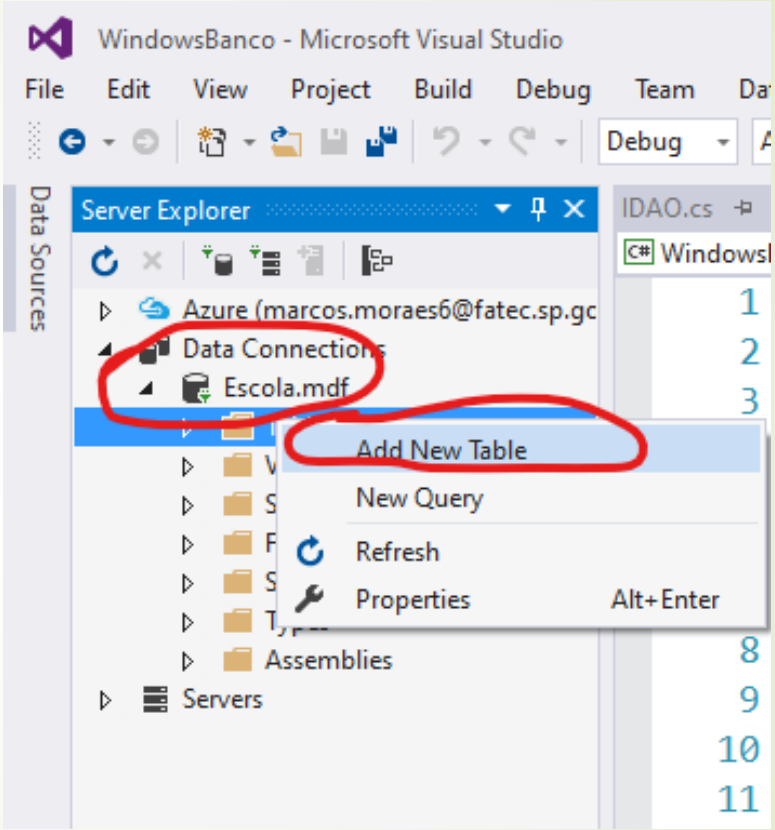
Criando o Banco

- Dê um clique duplo sobre o nome Escola.mdf, assim será ativada a janela Server Explorer abaixo:

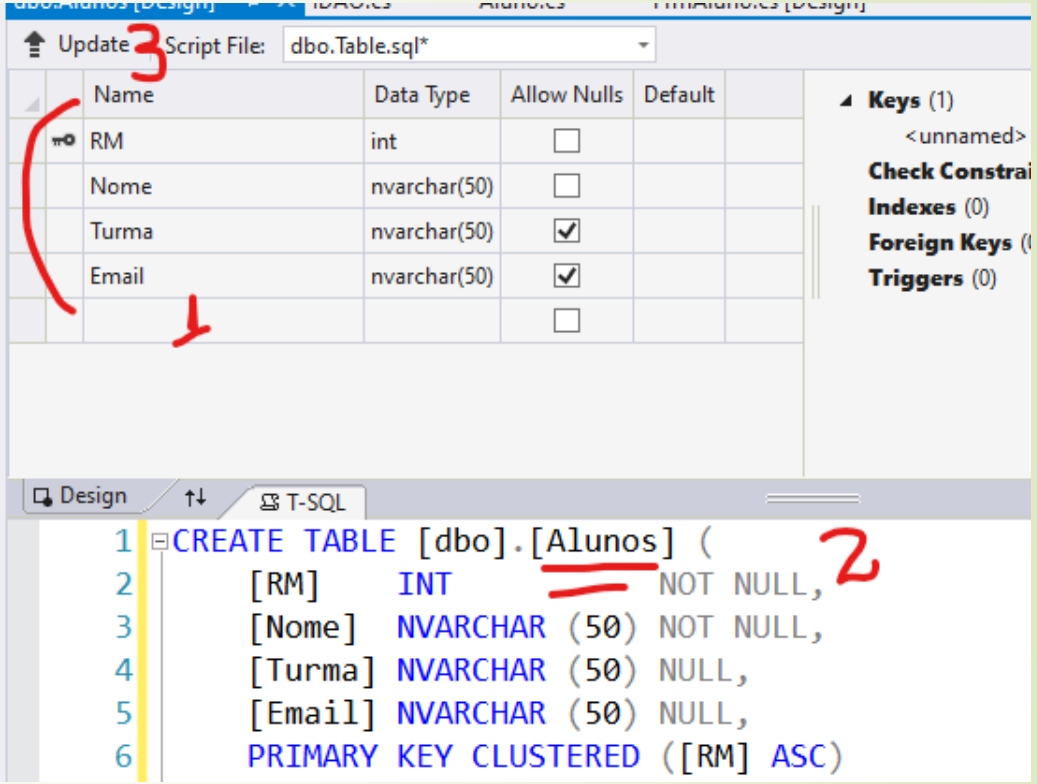


Criando tabela: Alunos

- Clique com o botão direito sobre Tables e escolha no menu de contexto a opção Add New Table

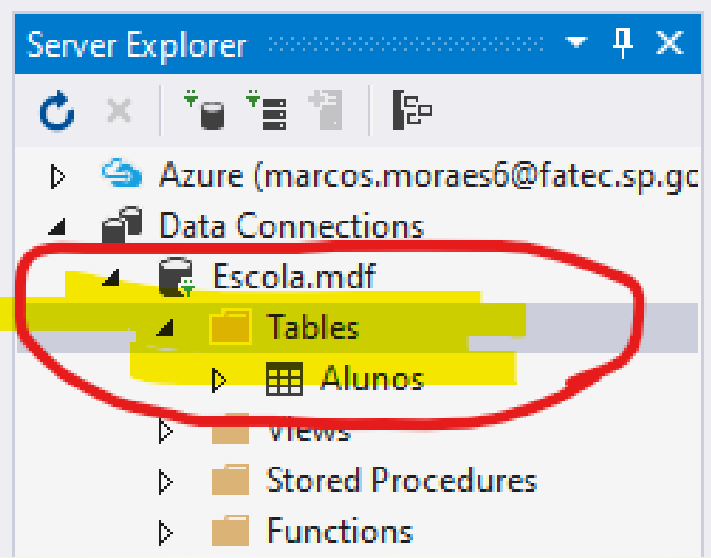


Estruture como a figura, seguindo a numeração



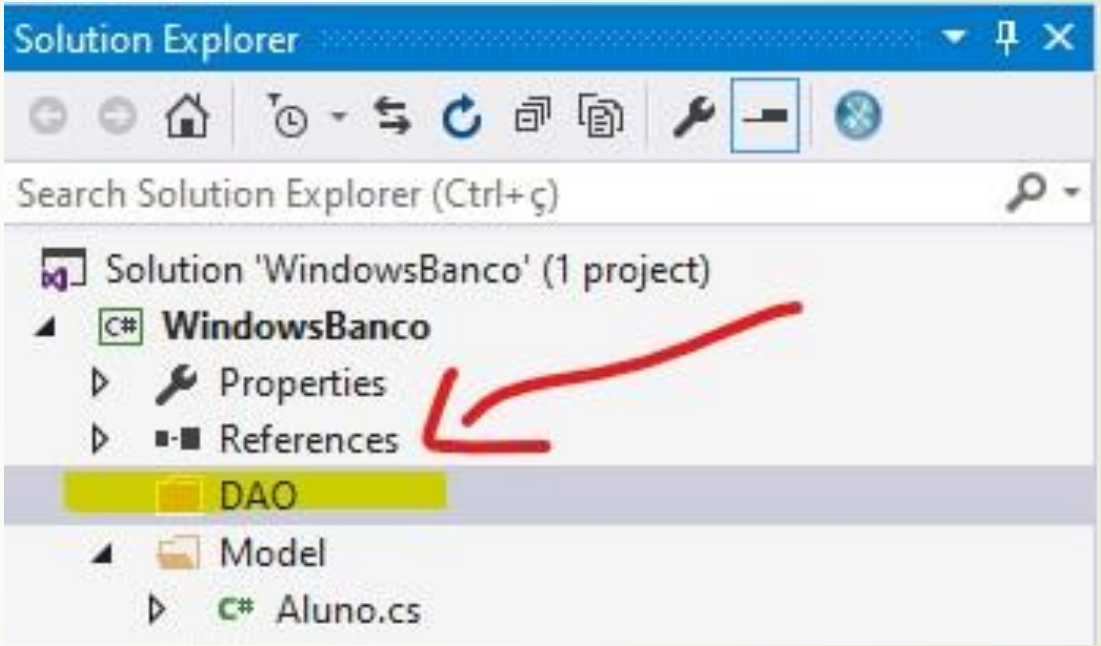
No último passo (3): confirme as janelas de atualização do script.

Resultado Final: Alunos
Expanda a pasta Tables



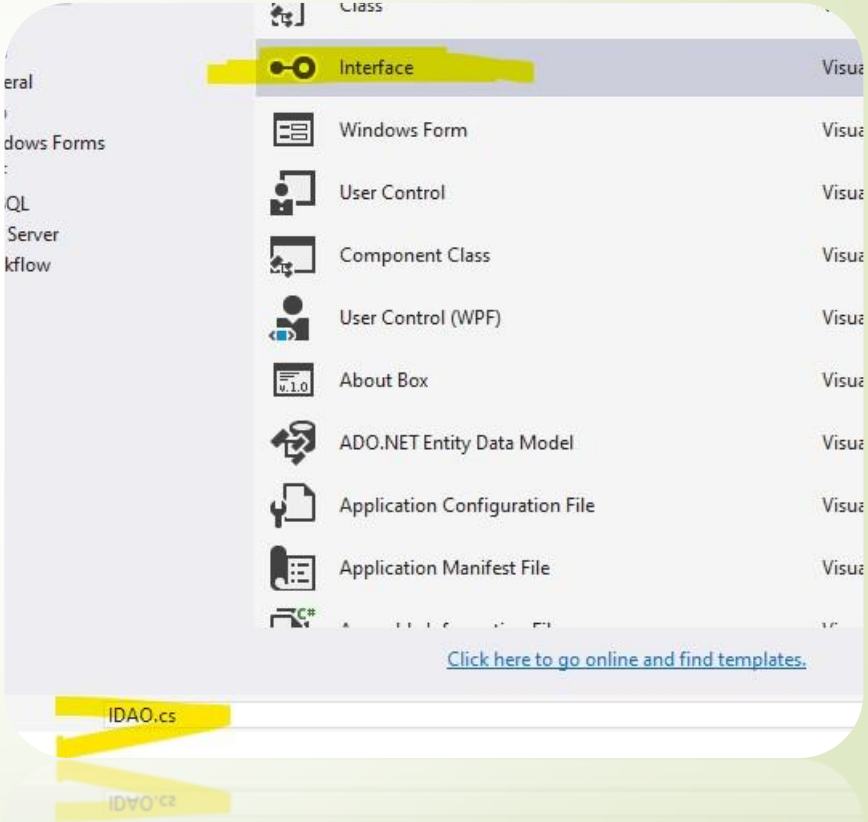
Crie uma pasta DAO

- Botão direito no nome do Projeto WindowsBanco e em seguida >> Add >> New folder.
- Dê o nome de **DAO.**



Interface IDAO

- Usando interfaces, você pode, por exemplo, incluir o comportamento de várias fontes em uma classe. Essa funcionalidade é importante em C# porque a **linguagem não dá suporte a várias heranças de classes**.
- Crie dentro da pasta DAO uma nova interface, desta forma:
  - Botão direito no nome da Pasta DAO, em seguida Add >> Add new Item. Assim abrirá a janela abaixo, selecione **Interface** e dê o nome **IDAO.cs**

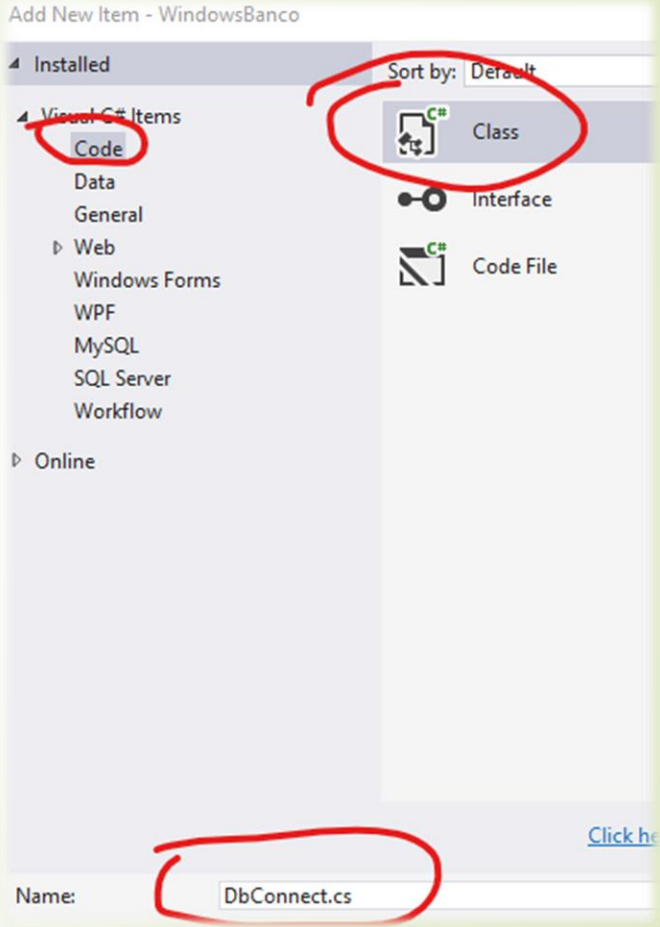


Definição da Interface Genérica IDAO.cs

```
namespace WindowsBanco.DAO
{
    interface IDAO<T> where T : new()
    {
        T buscar(int id);

        void alterar(T obj);

        void excluir(T obj);

        void inserir(T obj);

        List<T> listar();
    }
}
```

Classe DbConnect.cs

- Esta é uma classe pela qual se iniciará a conexão com o Banco de Dados, abertura da conexão amarrada com o banco de dados criado. Escola.
- Crie dentro da pasta DAO uma nova classe, desta forma:
  - Botão direito no nome da Pasta DAO, em seguida Add >> Add new Item. Assim abrirá a janela abaixo, selecione **Classe** e dê o nome **DbConnect.cs**



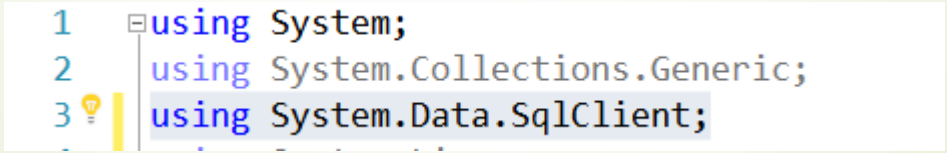
Código: DbConnect

```
public class DbConnect
  {
     //String de conexão com o banco
     private String conn =
        @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\USERS\MAROMO\DROPBOX\ETEC\TE
STES\WINDOWSBANCO\WINDOWSBANCO\ESCOLA.MDF;Integrated Security=True";

     //Objeto de conexão
     private static SqlConnection objConexao = null;
     //Construtor
     public DbConnect()
     {
        objConexao = new SqlConnection();
        objConexao.ConnectionString = conn;
        objConexao.Open();
     }
     //Método estátio que estabelece a conexao com o banco
     public static SqlConnection getConexao()
     {
        if (objConexao == null)
        {
           new DbConnect();
        }
        return objConexao;
     }
     //Método estático que fecha a conexao com o banco
     public static void fecharConexao()
     {
        objConexao.Close();
     }
  }
```

Atente as explicações no próximo slide.

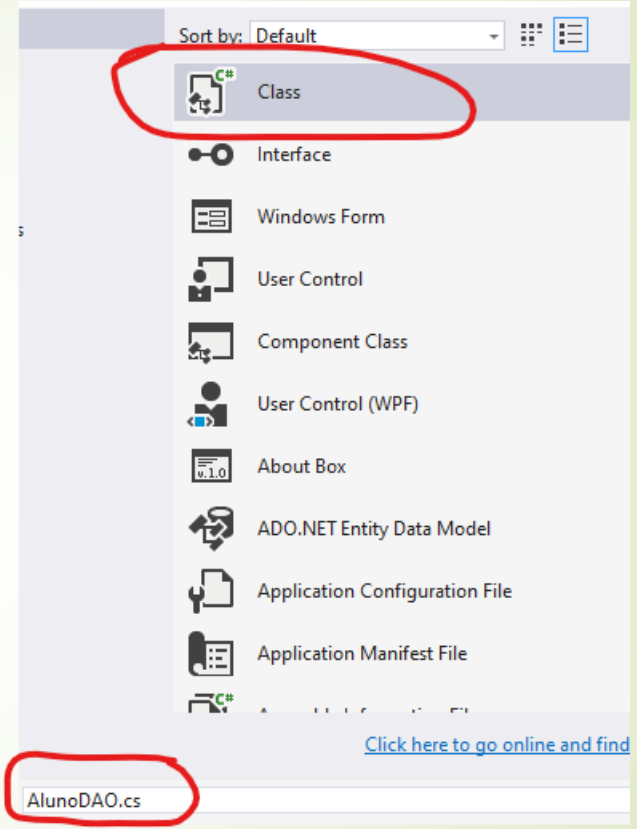
Notas

- Nota 1: você deve alterar o caminho do banco para o local onde o seu projeto está salvo. Caminho completo, nome meu caso está em **"C:\USERS\MAROMO\DROPBOX\ETEC\TESTES\WINDOWSBANCO\WINDOWSBANCO\ESCOLA.MDF". Use o caminho para o seu arquivo na sua máquina.**
- Nota 2: importe no começo do arquivo a linha e abaixo, using System.Data.SqlClient; Veja a figura:



Classe AlunoDAO.cs

- Esta é uma classe persistirá as informações de aluno na Tabela Alunos do nosso banco de dados chamado Escola.
- Crie dentro da pasta DAO uma nova classe, desta forma:
  - Botão direito no nome da Pasta DAO, em seguida Add >> Add new Item. Assim abrirá a janela abaixo, selecione **Classe** e dê o nome **AlunoDAO.cs**



Código AlunoDAO.cs
Início, as importações (using)
Slide (1/6)

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using WindowsBanco.Model;
```

Código AlunoDAO.cs (2/6) Início da classe. Método: alterar

```
public class AlunoDAO : IDAO<Aluno>
   {
        public void alterar(Aluno obj)
        {
            SqlConnection objConexao = DbConnect.getConexao();
            SqlParameter[] param = new SqlParameter[4]
            {
                new SqlParameter("@RM", SqlDbType.Int),
                new SqlParameter("@Nome", SqlDbType.NVarChar),
                new SqlParameter("@Turma", SqlDbType.NVarChar),
                new SqlParameter("@Email", SqlDbType.NVarChar)
            };
            param[0].Value = obj.RM;
            param[1].Value = obj.Nome;
            param[2].Value = obj.Turma;
            param[3].Value = obj.Email;
            string strQuery = "Update Alunos set Nome="
                + "@Nome, Turma=@Turma, "
            + " Email=@Email where RM=@RM";
            SqlCommand objCommand = new SqlCommand(strQuery, objConexao);
            objCommand.Parameters.AddRange(param);
            try
            {
                objCommand.ExecuteNonQuery();
            }
            catch (SqlException err)
            {
                throw err;
            }
        }
```

Código AlunoDAO.cs (3/6). Método: buscar

```
public Aluno buscar(int id)
 {
     SqlConnection objConexao = DbConnect.getConexao();
     SqlParameter param = new SqlParameter("RM", id);
     string strQuery = "SELECT * FROM alunos WHERE RM=@RM";
     SqlCommand objCommand = new SqlCommand(strQuery, objConexao);
     objCommand.Parameters.Add(param);
     try
     {
         SqlDataReader objReader = objCommand.ExecuteReader();
         Aluno objAluno = null;
         if (objReader.Read())
         {
             objAluno = new Aluno();
             objAluno.RM = Convert.ToInt16(objReader["RM"]);
             objAluno.Nome = Convert.ToString(objReader["Nome"]);
             objAluno.Turma = Convert.ToString(objReader["Turma"]);
             objAluno.Email = Convert.ToString(objReader["Email"]);
         }
         objReader.Close();
         return objAluno;
     }
     catch (SqlException err)
     {
         throw err;
     }
 }
```

Código AlunoDAO.cs (4/6). Método: excluir

```
public void excluir(Aluno obj)
        {
            SqlConnection objConexao = DbConnect.getConexao();
            SqlParameter param = new SqlParameter("@RM", obj.RM);
            string strQuery = "Delete from Alunos where RM=@RM";
            SqlCommand objCommand = new SqlCommand(strQuery, objConexao);
            objCommand.Parameters.Add(param);
            try
            {
                objCommand.ExecuteNonQuery();
            }
            catch (SqlException err)
            {
                throw err;
            }

        }
```

Código AlunoDAO.cs (5/6). Método: inserir

```
public void inserir(Aluno obj)
        {
            SqlConnection objConexao = DbConnect.getConexao();
            SqlParameter[] param = new SqlParameter[4]
            {
                new SqlParameter("@RM", SqlDbType.Int),
                new SqlParameter("@Nome", SqlDbType.NVarChar),
                new SqlParameter("@Turma", SqlDbType.NVarChar),
                new SqlParameter("@Email", SqlDbType.NVarChar)
            };
            param[0].Value = obj.RM;
            param[1].Value = obj.Nome;
            param[2].Value = obj.Turma;
            param[3].Value = obj.Email;
            string strQuery = "Insert into Alunos(RM," +
                " Nome, Turma, Email)" +
                " values(@RM, @Nome, @Turma, @Email)";
            SqlCommand objCommand = new SqlCommand(strQuery, objConexao);
            objCommand.Parameters.AddRange(param);
            try
            {
                objCommand.ExecuteNonQuery();
            }
            catch (SqlException err)
            {
                throw err;
            }
        }
```

Código AlunoDAO.cs (6/6). Método: listar

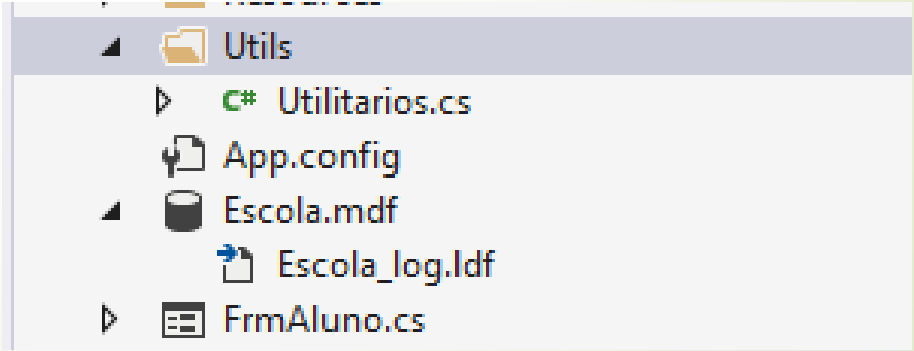
```
public List<Aluno> listar()
{
    List<Aluno> alunos = new List<Aluno>();
    SqlConnection objConexao = DbConnect.getConexao();
    string strQuery = "SELECT * FROM Alu os";
    SqlCommand objCommand = new SqlCommand(strQuery, objConexao);
    try
    {
        SqlDataReader objReader = objCommand.ExecuteReader();
        while (objReader.Read())
        {

            Aluno aluno = new Aluno();
            aluno.RM = Convert.ToInt16(objReader["RM"]);
            aluno.Nome = Convert.ToString(objReader["Nome"]);
            aluno.Turma = Convert.ToString(objReader["Turma"]);
            aluno.Email = Convert.ToString(objReader["Email"]);
            alunos.Add(aluno);
        }
        objReader.Close();
        return alunos;
    }
    catch (Exception err)
    {
        throw err;
    }
}
```

Nota Final sobre a Classe AlunoDAO

- Observe bem as aberturas e fechamento de blocos. { }
- Agora criaremos uma pasta adicional para recursos extras no nosso aplicativo. Chamaremos de Utils.
- E Em seguida faremos algumas preparações e melhorias na camada de apresentação, alterando algumas características do formulário FrmAluno.cs para finalmente codificar a camada de visão.

Pasta Utils

- Crie a pasta Utils da mesma forma que criou a camada Model. Depois crie uma classe dentro chamada Utilitarios.cs, como resultado terá o apresenta pela figura:



Código da classe: Utilitarios.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsBanco.Utils
{
    public class Utilitarios
    {
        internal static void limparTextos(Control.ControlCollection controls)
        {
            foreach (Control item in controls)
            {
                if (item is TextBox)
                {
                    TextBox texto = (TextBox)item;
                    texto.Text = "";
                }
            }
        }
    }
}
```

Camada View

Novamente

Trabalhando no formulário

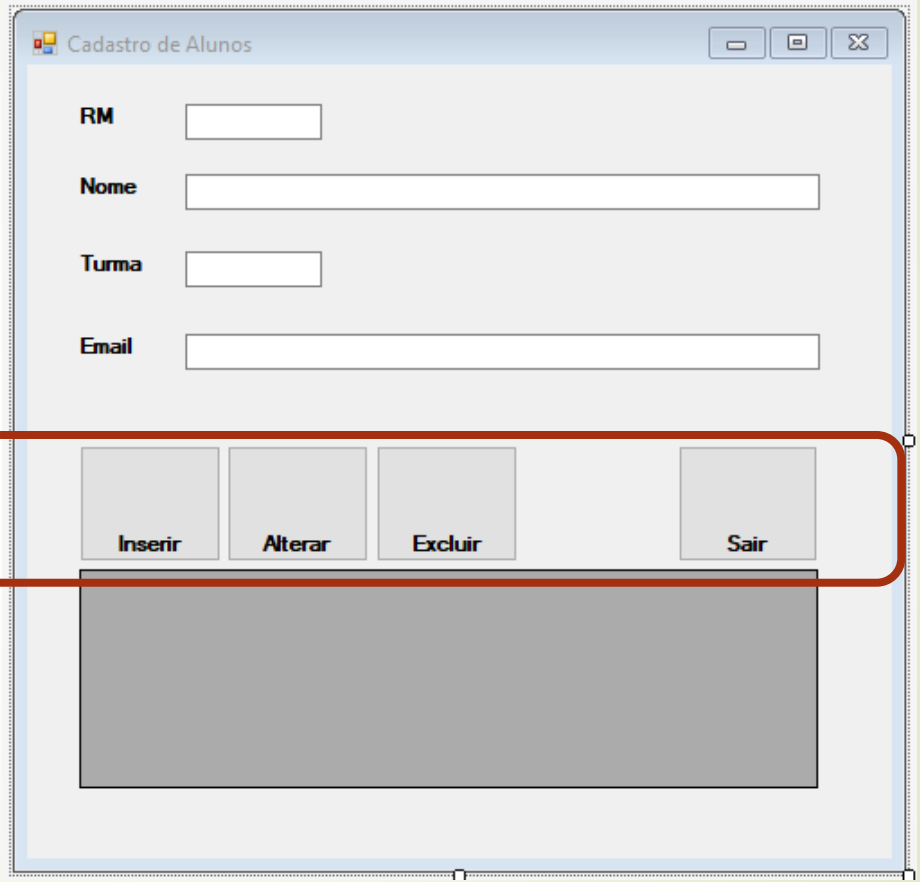
1) Inserindo imagem nos posições

2) Posicionando imagem nos botões

3) Alterando posicionamento do texto nos botões

Nosso formulário no momento:

1 Grupo Alvo



Com um dos botões selecionados

1) Localize a propriedade image e clique sobre o botão [...]

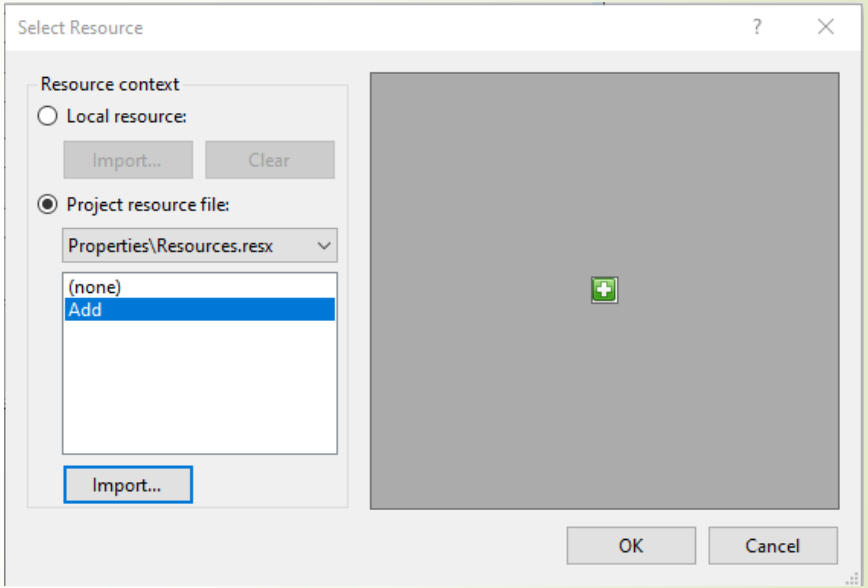
2) A caixa de Selecionar Recursos (Select Resource) será aberta.

3) Nesta caixa selecione a opção Project Resource file e

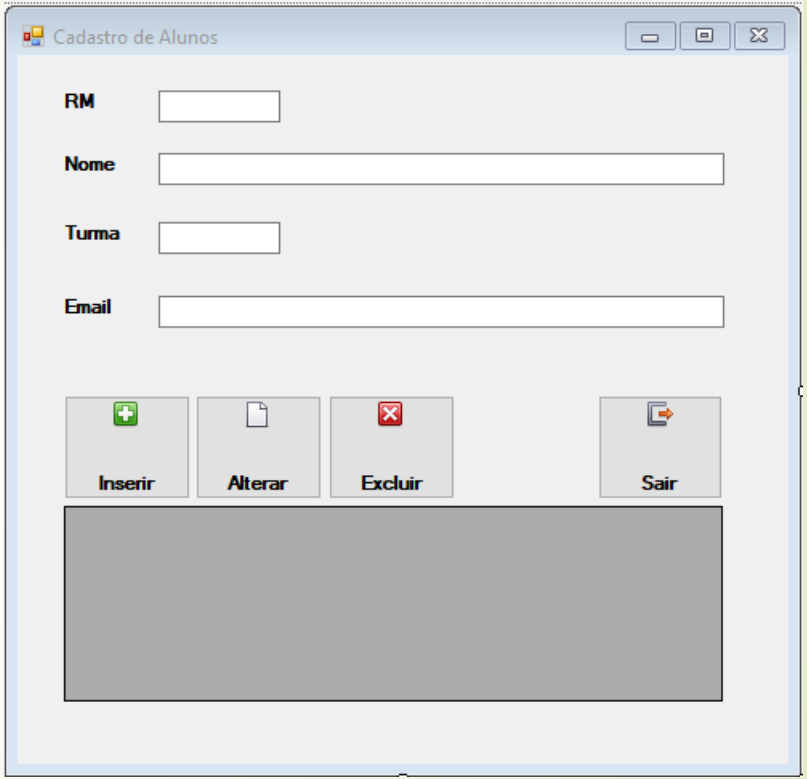
4) Clique sobre o botão Import ...

Meu botão Inserir

- Na minha máquina tenho imagens baixadas. Procure na internet por ícones para botões de formulário Windows (png) e baixe. Use-as como fiz com a imagem **Add.png**.
- Agora repita as operações para os demais botões. **Você deve acessar e alterar um a um.**
- Depois de importados e só clicar no Ok para que o desenho apareça no formulário.

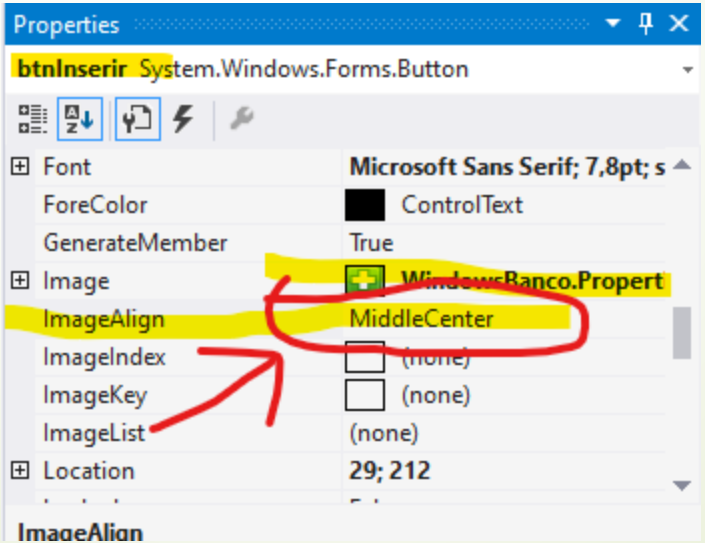


Resultado no momento

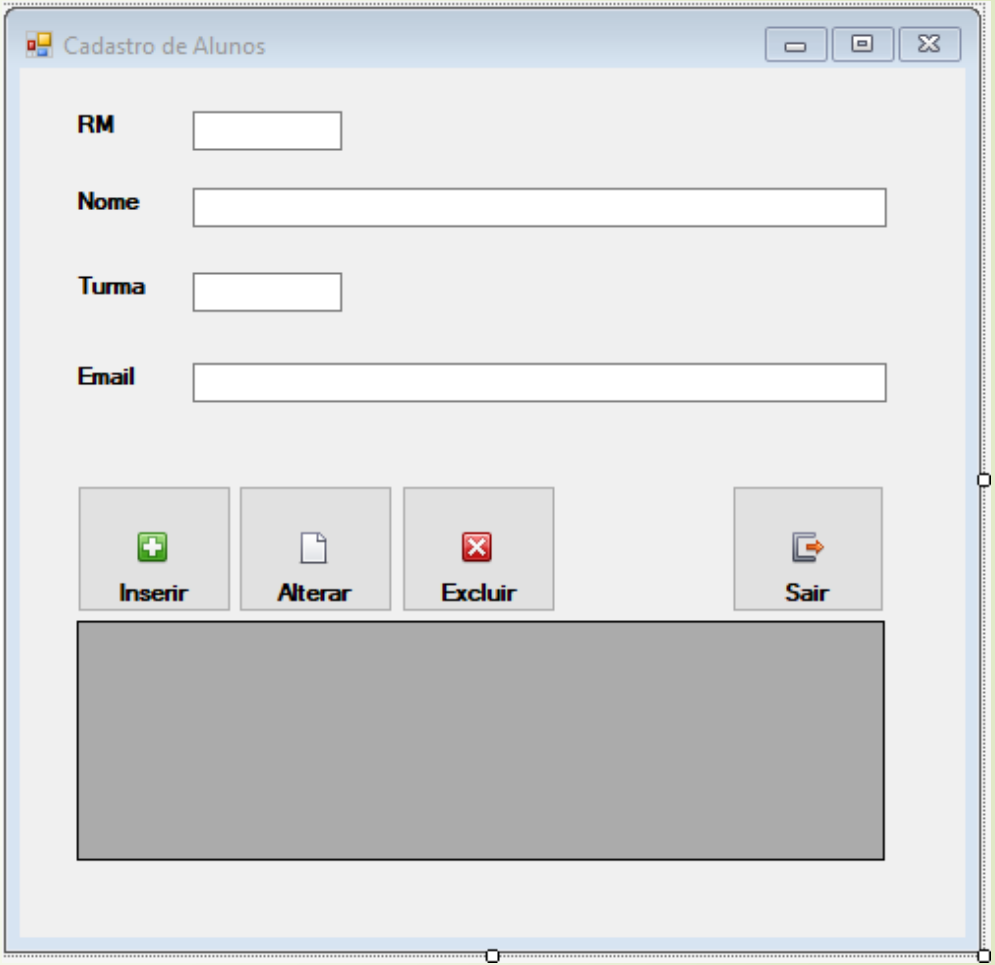


Alterando o posicionamento dos botões

- Selecione cada botão e altere a propriedade ImageAlign para MiddleCenter. Conforme imagem.



Resultado final



Finalização

Agora devemos programar (codificar) os objetos do nosso formulário para que aplicação possa ser testada.

```
public MostRecentFile[] GetFileList()
{
    if (!Loaded)
    {
        LoadMruList();
    }
    object[] array = files.ToArray();
    MostRecentFile[] mrfArray = new MostRecentFile[array.Length];
    array.CopyTo(mrfArray, 0);
    return mrfArray;
}
public bool Contains(string fileName)
{
    if (!Loaded)
    {
        LoadMruList();
    }
    string lcFileName = fileName.ToLower();

    foreach (MostRecentFile mrf in files)
    {
        string lcMrf = mrf.FileName.ToLower();

        if (0 == String.Compare(lcMrf, lcFileName))
        {
            return true;
        }
    }
```

Botão inserir (Código)
Duplo clique: btnInserir_Click

```
private void btnInserir_Click(object sender, EventArgs e)
        {
            Aluno aluno = new Aluno();
            try
            {
                aluno.RM = int.Parse(txtRM.Text);
                aluno.Nome = txtNome.Text;
                aluno.Turma = txtTurma.Text;
                aluno.Email = txtEmail.Text;
                //Acessar camada de Dados
                AlunoDAO dao = new AlunoDAO();

                dao.inserir(aluno);
                MessageBox.Show("Registro Gravado com Suceso");
                Utilitarios.limparTextos(this.Controls);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Aconteceu o erro: " + ex.ToString());
            }
}
```

Não esqueça das importações no início da classe

```
using System.Windows.Forms;
using WindowsBanco.DAO;
using WindowsBanco.Model;
using WindowsBanco.Utils;
```

Código do Botão alterar

```
private void btnAlterar_Click(object sender, EventArgs e)
        {
            Aluno aluno = new Aluno();
            try
            {
                aluno.RM = int.Parse(txtRM.Text);
                aluno.Nome = txtNome.Text;
                aluno.Turma = txtTurma.Text;
                aluno.Email = txtEmail.Text;
                //Acessar camada de Dados
                AlunoDAO dao = new AlunoDAO();

                dao.alterar(aluno);
                MessageBox.Show("Registro Alterado com Suceso");
                Utilitarios.limparTextos(this.Controls);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Aconteceu o erro: " + ex.ToString());
            }
        }
```

Botão Excluir

```
private void btnExcluir_Click(object sender, EventArgs e)
        {
            if ((txtRM.Text == null) || (txtRM.Text == ""))
            {
                MessageBox.Show("Favor indicar o RM do aluno");
                return;
            }
            try
            {
                Aluno aluno = new Aluno();
                aluno.RM = int.Parse(txtRM.Text);

                //Acessar camada de Dados
                AlunoDAO dao = new AlunoDAO();

                dao.excluir(aluno);
                MessageBox.Show("Registro Excluído com Suceso");
                Utilitarios.limparTextos(this.Controls);
            }
            catch (Exception ex)
            {
                MessageBox.Show("Aconteceu o erro: " + ex.ToString());
            }
        }
```

Botão Sair

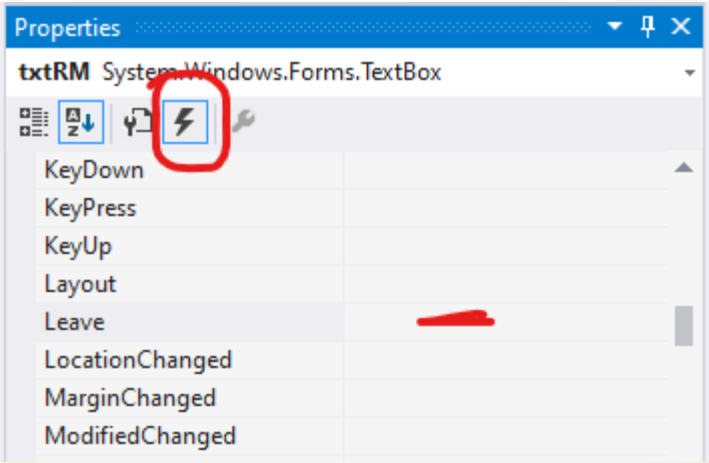
```
private void btnSair_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show(null, "Deseja sair do Aplicativo",
                "Mensagem (SAIR)", MessageBoxButtons.YesNo,
                MessageBoxIcon.Question) == DialogResult.Yes)
            {
                Application.Exit();
            }
        }
```

Para finalizar

- Quando digitamos o RM de um aluno, ao sair do campo é necessário verificar se existe aluno com este RM, caso seja verdadeiro:
- Deve-se apresentar os dados do aluno em tela, e deixar o botão alterar ativo, para que o usuário possa depois de alterar algum dado, clicar sobre o botão e realizar a alteração, mas caso o aluno não seja encontrado, deve-se:
- Deixar os demais campos vazios e habilitar o botão de Inserir para que ao final da digitação o usuário possa incluir os dados digitado do aluno.

Preparando para procura por RM de aluno.

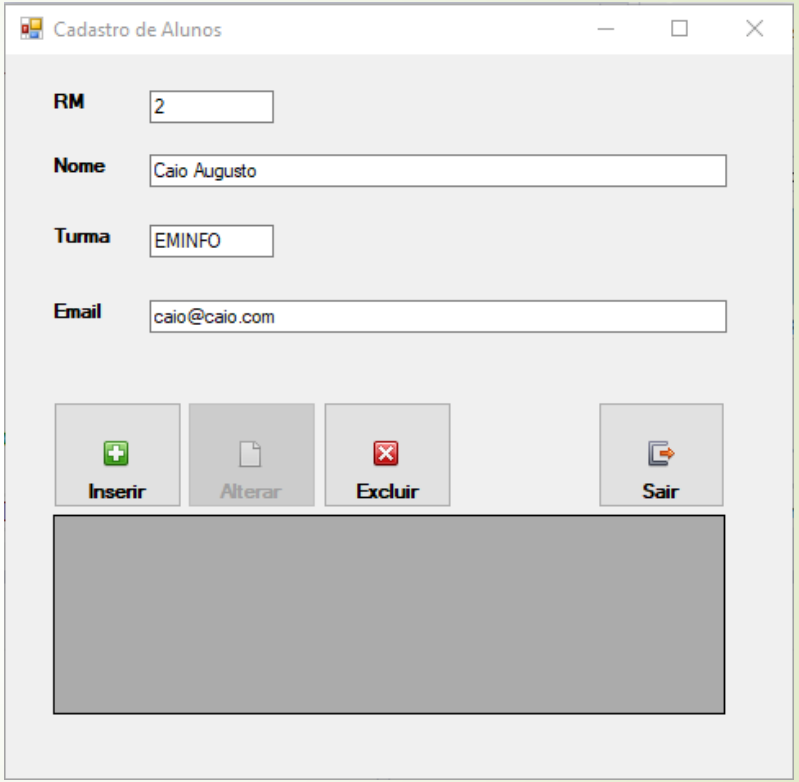
- Selecione o controle txtRM e em seguida na janela de propriedades clique sobre o botão events 
- Procure pelo evento Leave, e dê um duplo clique sobre.



Código do evento

```
private void txtRM_Leave(object sender, EventArgs e)
        {
            try
            {
              int rm = 0;
                if (txtRM.Text != "") { rm = int.Parse(txtRM.Text); }
                Aluno aluno;
                AlunoDAO dao = new AlunoDAO();
                aluno = dao.buscar(rm);
                if (aluno != null)
                {
                 txtNome.Text = aluno.Nome;
                    txtTurma.Text = aluno.Turma;
                    txtEmail.Text = aluno.Email;
                    btnInserir.Enabled = false;
                    btnAlterar.Enabled = true;
                }
                else
                {
                    btnInserir.Enabled = true;
                    btnAlterar.Enabled = false;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Aconteceu o erro: " + ex.ToString());
            }
        }
```

Testando inserir

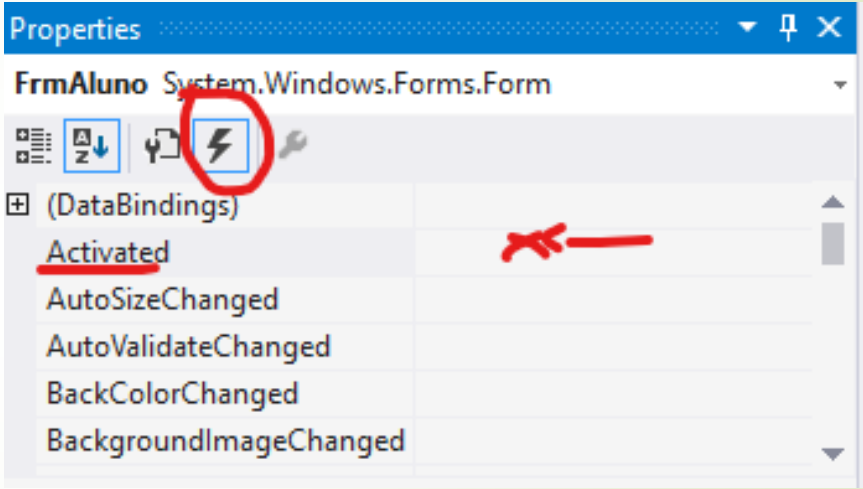


Nota

- Observe que ao digitar um RM não cadastrado fica disponível o botão inserir, pois trata-se de um novo registro. Depois de inserido se você quiser realizar alguma alteração, basta digitar o mesmo RM e perceberá que agora o botão liberado é o Alterar.
- Para terminarmos este exemplo, falta apenas ajustar o **dataGridAlunos** para que ele exiba a lista de alunos sempre atualizadas. Para isto, faça as tarefas solicitadas no próximo artigo.

Alterando o evento Form_Activate

- Este evento é disparando quando o formulário depois de carregado todos os seus componentes. Para isto dê um clique simples numa área vazia do formulário, e na janela de propriedades clique de novo no botão events e depois de localizado activate, de um clique duplo para acessar a área de código.



Código do Form_Activate

- Neste código faremos uma chamada a um método ainda não criado, que se chamará carregarGrid(), cujo código deve ser digitado na sequencia. Como mostrado abaixo:

```
private void FrmAluno_Activated(object sender, EventArgs e)
        {
            carregarGrid();
        }

        private void carregarGrid()
        {
            var lista = new List<Aluno>();
            var dao = new AlunoDAO();
            lista = dao.listar();
            dataGridAlunos.DataSource = null; //limpar
            dataGridAlunos.DataSource = lista; //carregar
        }
```

Nota do código anterior

No método carregarGrid temos uma variável do tipo List (Uma lista genérica do tipo Aluno). Outra variável chamada dao to tipo AlunoDAO, que servirá para listar os dados no list criado.

Em seguida limpa-se o grid de alunos atual, e o alimenta com a lista de alunos atualizada, usando a propriedade DataSource (origem de dados) que está associada a lista criada em memória.

Fim

Muito obrigado,

Maromo e Rodrigo