Apostila de Linguagem de Programação I

Professor Eduardo Rosalém Marcelino

(2017 / 2018)

eduardormbr@gmail.com

Conteúdo

Principais teclas de atalho

Teclas de atalho que podem ser usadas quando o Visual Studio tela estiver configurada como na imagem acima

Descrição	Tecla de Atalho
Janela de propriedades do objeto	F4
Compilar	F6
Executar o programa	F5
Ir para o código fonte do formulário	F7
Retornar ao formulário	Shift + F7
Adicionar namespace	Alt + Shift + F10
Criar propriedade	CTRL + R + E
Depurar sem entrar nos métodos	F10
Depurar entrando nos métodos	F11
Adicionar Break Point	F9
Pesquisar pelo nome de um arquivo no projeto	CTRL + ,
Pesquisar no conteúdo de um arquivo	CTRL + F
Pesquisar no conteúdo de todos os arquivos do projeto	CTRL + SHIFT + F

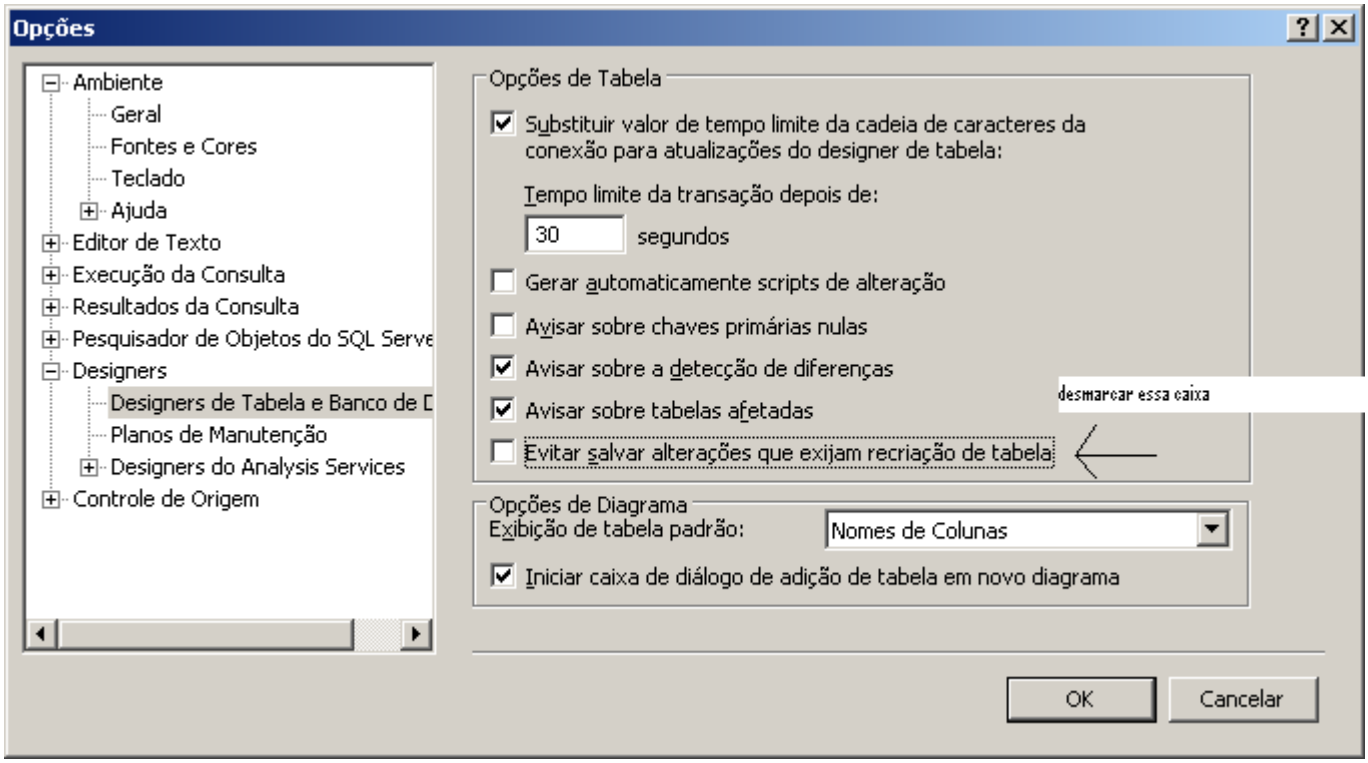
1. Criando um cadastro de Aluno – parte 1

Crie um novo projeto Windows Forms chamado CadastroAlunoBD.

Instrução SQL para criar a tabela aluno:

```
CREATE  TABLE Alunos (
      Id  int NOT NULL PRIMARY KEY,
      nome  varchar(50) NULL,
      mensalidade decimal (18, 2) NULL,
      cidadeId int NULL,
      DataNascimento datetime NULL
)
```

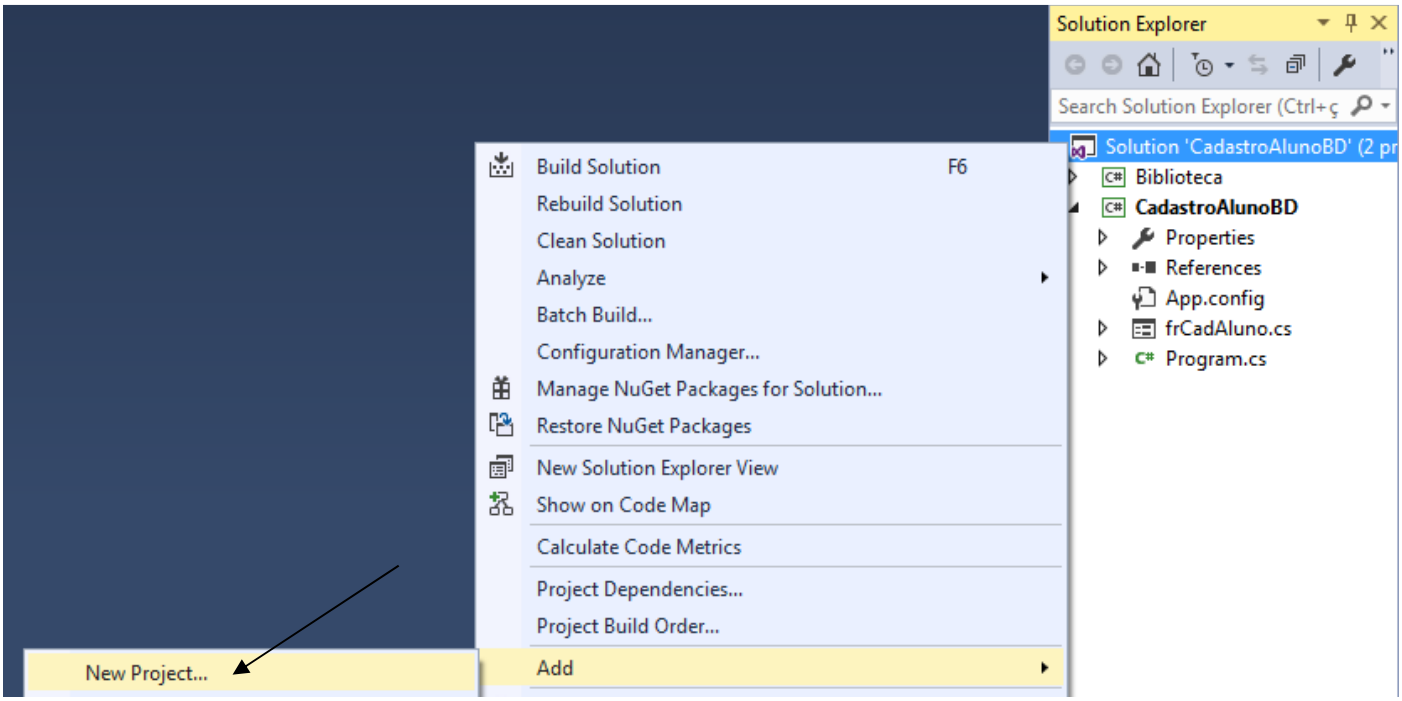
Caso tente alterar a estrutura de uma tabela no modo visual, o **SQL Server** pode dar um erro dizendo que não é possível alterar a estrutura da tabela. Para permitir essa alteração, entre nas opções do SQL Server e desmarque a seguinte opção:

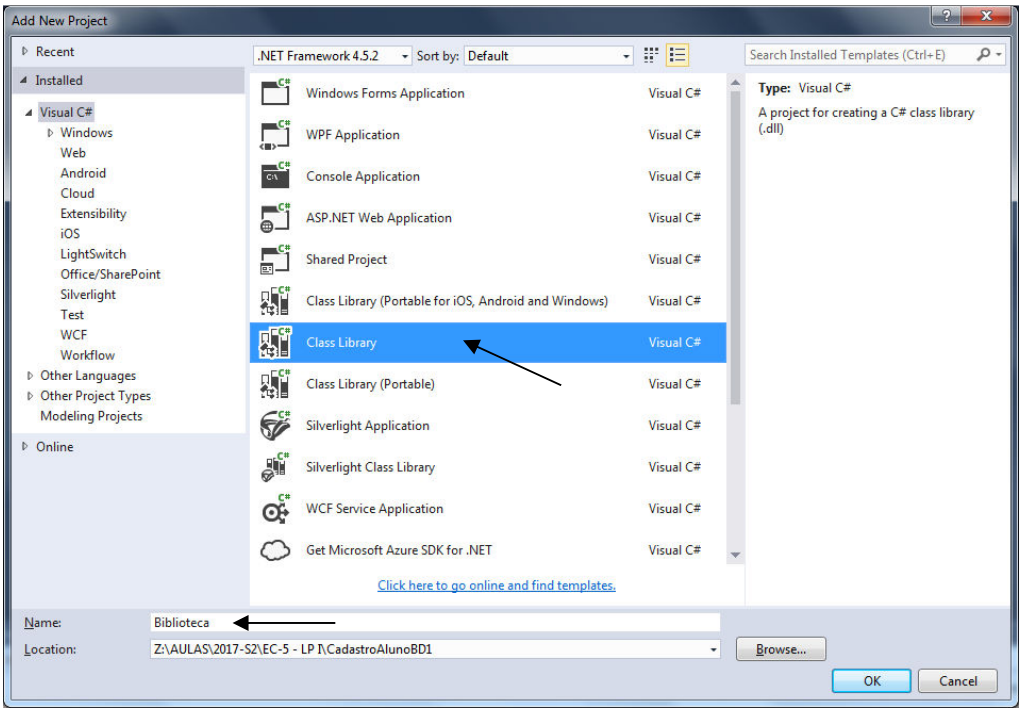


Separação dos projetos

Vamos criar um projeto chamado “biblioteca” para separar as classes de negócio das interfaces do usuário.

No “Solution explorer”, clique com o botão direito na solução e adicione um novo projeto, com o nome biblioteca. O tipo do projeto é Class Library.





No projeto Biblioteca, através da “Solution Explorer”, crie a pasta “Exceptions” e dentro dela crie a classe “ValidacaoException”.

Código fonte da classe:

```
using System;

namespace Biblioteca.Exceptions
{
    public class ValidacaoException : Exception
    {
        public ValidacaoException(string msg ) : base(msg)
        {
        }
    }
}
```

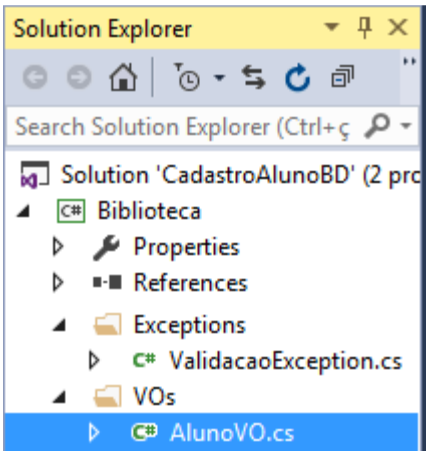
Também no projeto Biblioteca, crie o enumerador TipoMensagemEnum na pasta Enumeradores, como no código abaixo:

```
namespace Biblioteca.Enumeradores
{
    public enum TipoMensagemEnum { alerta, erro, informacao, pergunta }
}
```

Classe VO (Value Object) ou DTO (Data Transfer Object)

Classe que irá conter todos os atributos da tabela Aluno. Com o objetivo de encapsular os atributos, podemos utilizar métodos de acesso públicos (Getters e Setters) para cada atributo. Esta classe facilita o transporte das informações entre as camadas da aplicação.

Crie uma pasta chamada VOs no projeto biblioteca, e dentro desta pasta crie a classe AlunoVO:




```
using Biblioteca.Exceptions;

using System;

namespace Biblioteca.VOs
```

```
{
    public class AlunoVO
    {
        private int id;
        private string nome;
        private double mensalidade;
        private int cidadeId;
        private DateTime dataNascimento;


        public int Id
        {
            get
            {
                return id;
            }

            set
            {
                if (value < 0)
                    throw new ValidacaoException("Id não pode ser negativo!");
                else
                    id = value;
            }
        }



        public string Nome
        {
            get
            {
                return nome;
            }
            set
            {
                if (string.IsNullOrEmpty(value))
                    throw new ValidacaoException("Informe o nome!");
                else
                    nome = value;
            }
        }



        public double Mensalidade
        {
            get
            {
                return mensalidade;
            }

            set
            {
                if (value < 0)
                    throw new ValidacaoException("Mensalidade não pode ser negativa!");
                else
                    mensalidade = value;
            }
        }


        public int CidadeId
        {
            get
            {
                return cidadeId;
            }
            set
            {
                if (value < 0)
                    throw new ValidacaoException("Código da cidade não ser negativo!");
                else
                    cidadeId = value;
            }
        }


        public DateTime DataNascimento
```

```
        {
            get
            {
                return dataNascimento;
            }

            set
            {
                if (value  > DateTime.Now)
                    throw new ValidacaoException("Mas a pessoa nem nasceu ainda!");
                else
                    dataNascimento = value;
            }
        }
    }
}
```

Classe de Conexão com o Banco de dados

A aplicação deve ter apenas uma classe de conexão com banco de dados. Sempre que for necessário utilizar uma conexão, devemos acessar essa classe e executar o método que devolve a conexão aberta. Para se conectar ao banco de dados é necessária uma string de conexão.

No projeto “Biblioteca”, crie uma pasta chamada “DAO” e crie uma classe chamada “ConexaoBD”

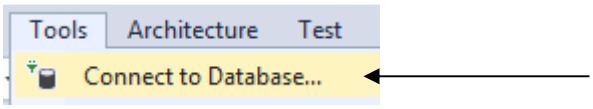
```
using System.Data.SqlClient;

namespace Biblioteca.DAO
{
    public static class ConexaoBD
    {
        /// <summary>
        /// Método Estático que retorna um conexao aberta com o BD
        /// </summary>
        /// <returns>Conexão aberta</returns>
        public static SqlConnection GetConexao()
        {
            string strCon = "Data Source=LOCALHOST;Initial Catalog=AULADB;user id=sa; password=123456";
            SqlConnection conexao = new SqlConnection(strCon);
            conexao.Open();
            return conexao;
        }
    }
}
```

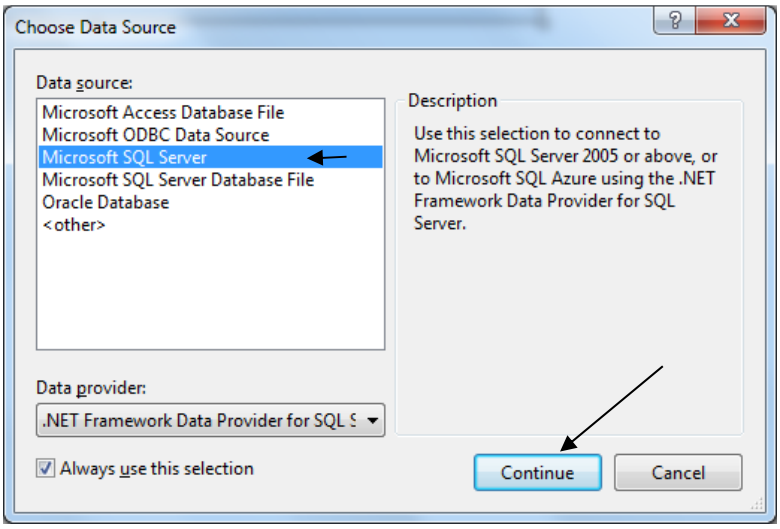
Caso não saiba a senha, tente se conectar por autenticação Windows. A string neste caso fica assim:

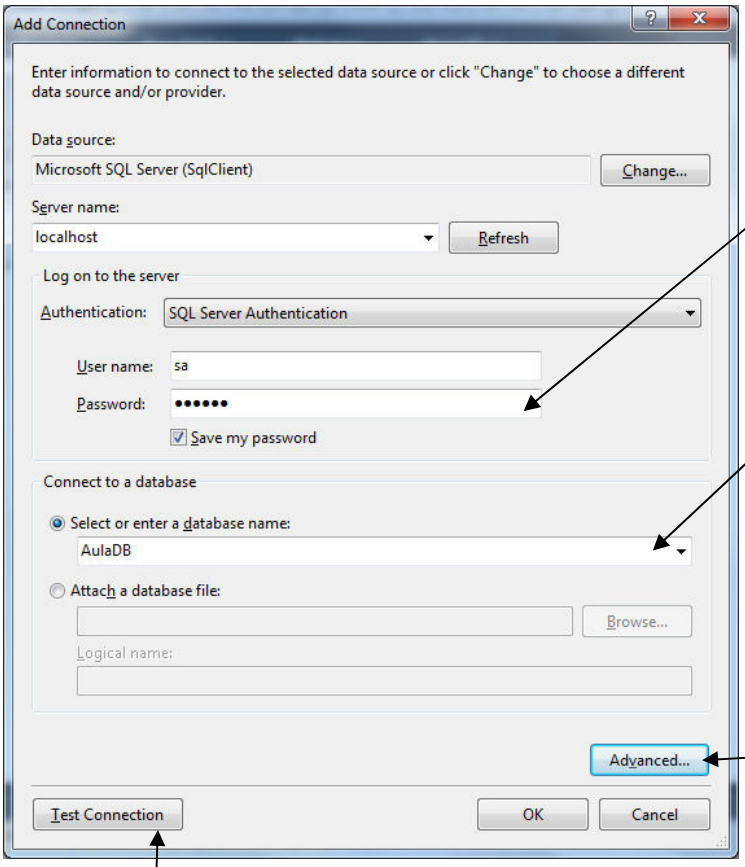
```
"Data Source=LOCALHOST;Initial Catalog=AULADB; integrated security=true";
```

Caso não esteja se conectando com êxito, abaixo segue uma forma de testar sua conexão com o banco de dados:

Acesse o menu abaixo.







Caso não saiba a senha, você pode tentar se conectar via autenticação Windows. Veja mas abaixo como fica a string de conexão para este tipo de conexão.

Se as informações fornecidas anteriormente estiverem corretas, os bancos de dados disponíveis irão aparecer nesta lista.

Selecione o banco de dados e clique em OK

Clique neste botão para ver a string de conexão.

Clique neste botão para testar a conexão.

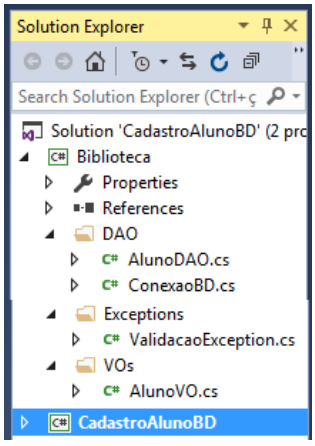
Classe DAO para realização das operações de acesso ao banco de dados

DAO (acrônimo de Data Access Object), é um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados. O padrão DAO consiste em abstrair o mecanismo de persistência utilizado na aplicação. A camada de negócios acessa os dados persistidos sem ter conhecimento se os dados estão em um banco de dados relacional ou um arquivo XML. O padrão DAO esconde os detalhes da execução da origem dos dados.

Por enquanto iremos incluir apenas o método para incluir alunos:

Na pasta DAO, crie a classe AlunoDAO

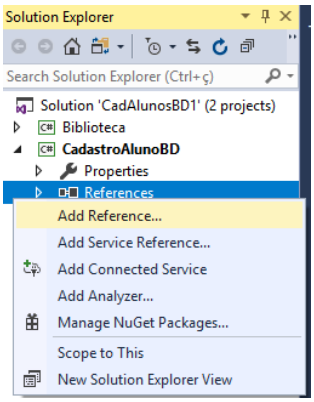
```
using Biblioteca.VOs;
using System;
using System.Data.SqlClient;

namespace Biblioteca.DAO
{
    public static class AlunoDAO
    {
        /// <summary>
        /// Método estático para inserir um aluno no BD
        /// </summary>
        /// <param name="aluno">objeto aluno com todas os atributos preenchidos</param>
        public static void Inserir(AlunoVO aluno)
        {
            SqlConnection conexao = ConexaoBD.GetConexao();
            try
            {
                //devemos substituir a ',' por '.'
                string mensalidade = aluno.Mensalidade.ToString().Replace(',', '.');
                // set dateformat dmy; este comando serve para alterar a
                //forma como o SQL Server entende o formato de data
                string sql = String.Format("set dateformat dmy; " +
                "insert into alunos(id, nome, mensalidade, cidadeId, dataNascimento)" +
                "values ( {0}, '{1}', {2}, {3}, '{4}')", aluno.Id,
                aluno.Nome, mensalidade, aluno.CidadeId, aluno.DataNascimento);
                SqlCommand comando = new SqlCommand(sql, conexao);
                comando.ExecuteNonQuery();
            }
            finally
            {
                conexao.Close();
            }
        }
    }
}
```



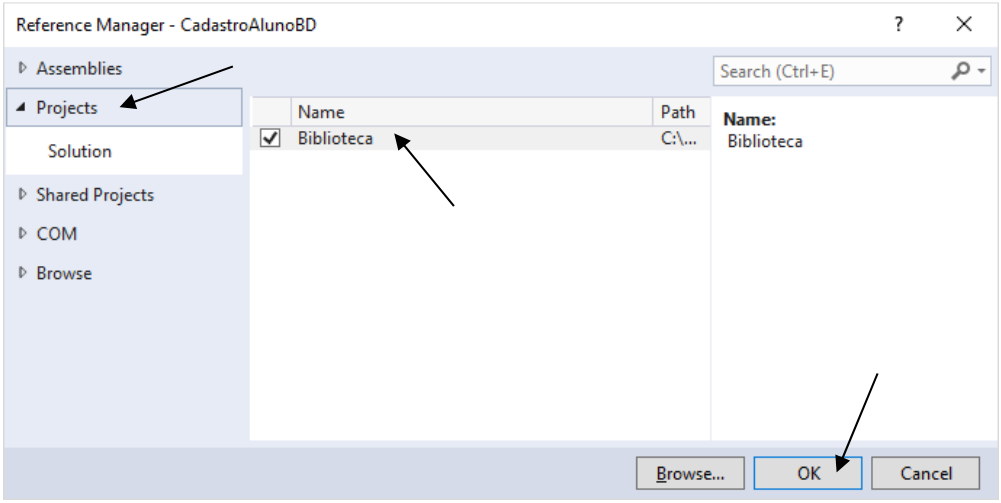
Adicionando uma referência ao projeto

Para que um projeto consiga utilizar as classes e outros elementos criados em outro projeto, primeiro precisamos adicionar o projeto como referência.

No projeto (Windows Forms) CadastroAlunoBD, clique com o botão direito sobre a guia References e escolha a opção “Add Reference...”



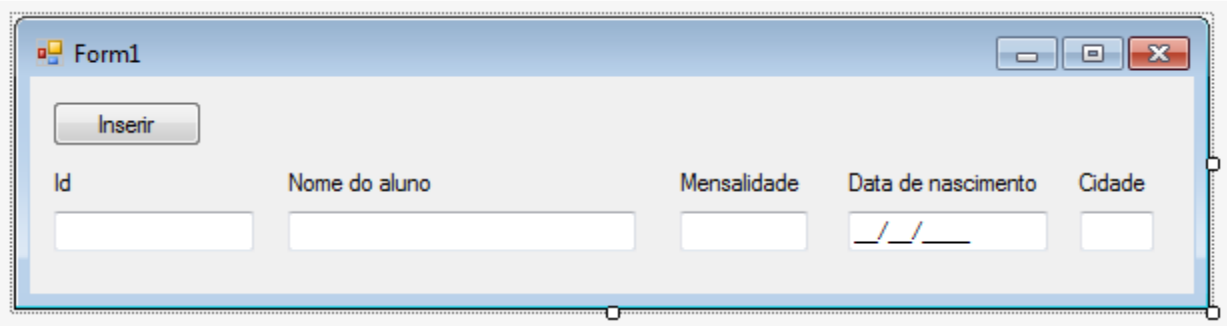
A referência do projeto Biblioteca pode ser adicionada escolhendo-se a DLL ou o projeto. Vamos fazer adicionando-se como referência o projeto.



Se você preferir adicionar a DLL, clique no botão “Browse” (na imagem acima) e selecione o arquivo DLL que está na pasta do projeto \bin\debug.

Formulário de Cadastro

O formulário para incluir terá esta aparência:



Código do botão inserir:

```
        private void btnInclui_Click(object sender, EventArgs e)
        {
            try
            {
                AlunoVO a = new AlunoVO();
                a.Id = Convert.ToInt32(txtId.Text);
                a.Nome = txtNome.Text;
                a.CidadeId = Convert.ToInt32(txtCidade.Text);
                a.DataNascimento = Convert.ToDateTime(txtData.Text);
                a.Mensalidade = Convert.ToDouble(txtMensalidade.Text);

                AlunoDAO.Inserir(a);
            }
            catch (Exception erro)
            {
                MessageBox.Show(erro.Message);
            }
        }
```

Exercícios:

1-) Adicione na classe DAO os métodos para alterar e excluir um aluno. Adicione os botões na tela para que seja possível efetuar estes novos dois métodos.

2-) Em uma nova solution, crie um crie um cadastro de jogos (inclusão, alteração e exclusão). Crie o projeto Biblioteca, as classes DAO, VO, igual fizemos no exemplo do aluno.

Tabela:

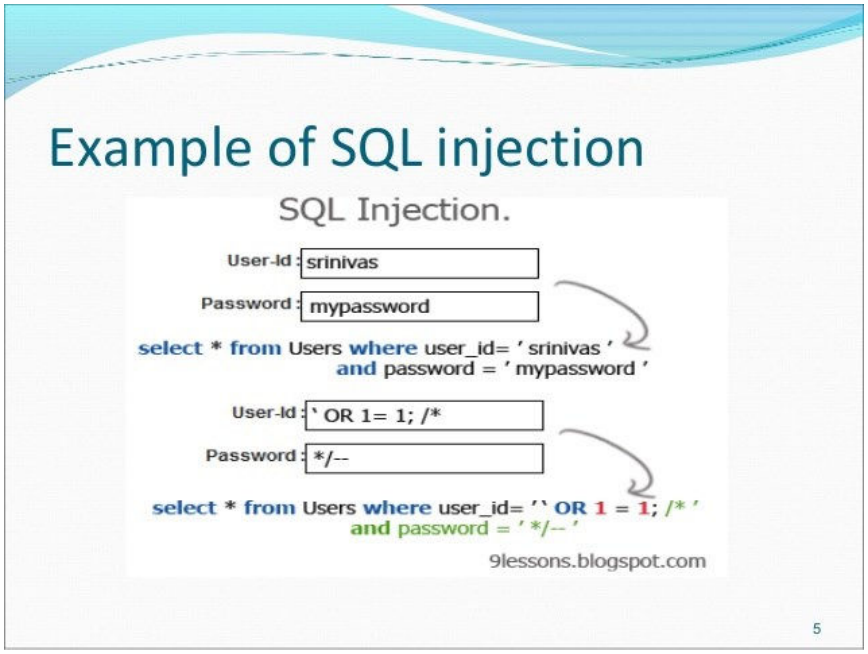
```
CREATE TABLE jogos(
       [id] [int] NOT NULL  primary key,
       [descricao] [varchar](50) NULL,
       [valor_locacao] [decimal](18, 2) NULL,
       [data_aquisicao] [datetime] NULL,
       [categoriaID] [int] NULL)
```

2. SQL Injection

A Injeção de SQL, mais conhecida através do termo americano SQL Injection, é um tipo de ameaça de segurança que se aproveita de falhas em sistemas que interagem com bases de dados via SQL. A injeção de SQL ocorre quando o atacante consegue inserir uma série de instruções SQL dentro de uma consulta (query) através da manipulação das entradas de dados de uma aplicação.

https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_de_SQL

https://www.tecmundo.com.br/tecmundo-explica/113195-sql-injection-saiba-tudo-ataque-simples-devastador.htm



3. Criando um cadastro de Aluno – parte 2

Classe DAO com parâmetros. Métodos excluir e alterar

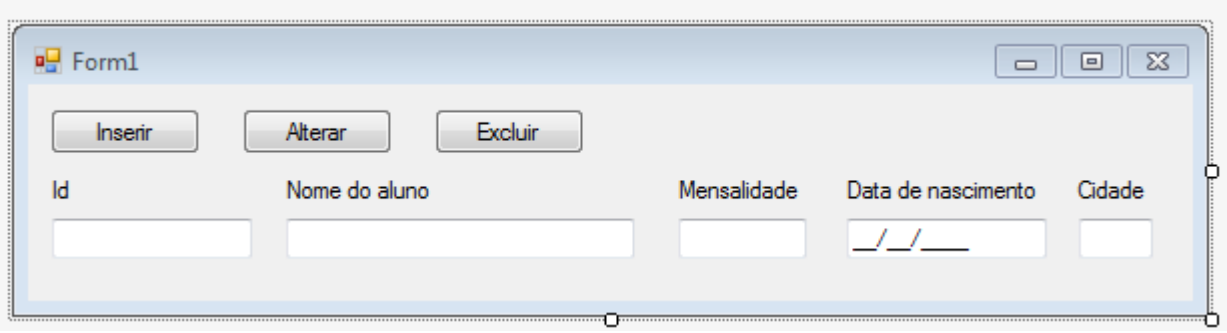
Nova Classe DAO, desta vez utilizando parâmetros para enviar os dados para o banco, evitando assim o SQL Injection. O bloco “using” faz o “dispose” dos objetos, garantindo assim que o mesmo seja destruído, fazendo às vezes do bloco try-finally. Segue abaixo a classe completa, com os métodos de inclusão, alteração e exclusão.

```
using Biblioteca.VOs;
using System;
using System.Data.SqlClient;

namespace Biblioteca.DAO
{
    public static class AlunoDAO
    {
        /// <summary>
        /// Método estático para inserir um aluno no BD
        /// </summary>
        /// <param name="aluno">objeto aluno com todas os atributos preenchidos</param>
        public static void Inserir(AlunoVO aluno)
        {
            using (SqlConnection conexao = ConexaoBD.GetConexao())
            {
                SqlParameter[] parametros = new SqlParameter[5];
                parametros[0] = new SqlParameter("id", aluno.Id);
                parametros[1] = new SqlParameter("nome", aluno.Nome);
                parametros[2] = new SqlParameter("mensalidade", aluno.Mensalidade);
                parametros[3] = new SqlParameter("cidadeId", aluno.CidadeId);
                parametros[4] = new SqlParameter("dataNascimento", aluno.DataNascimento);

                string sql =
                "insert into alunos(id, nome, mensalidade, cidadeId, dataNascimento)" +
                "values ( @id, @nome, @mensalidade, @cidadeId, @dataNascimento)";
                SqlCommand comando = new SqlCommand(sql, conexao);
                comando.Parameters.AddRange(parametros);
                comando.ExecuteNonQuery();
                conexao.Close();
            }
        }


        public static void Alterar(AlunoVO aluno)
        {
            using (SqlConnection conexao = ConexaoBD.GetConexao())
            {
                SqlParameter[] parametros = new SqlParameter[5];
                parametros[0] = new SqlParameter("id", aluno.Id);
                parametros[1] = new SqlParameter("nome", aluno.Nome);
                parametros[2] = new SqlParameter("mensalidade", aluno.Mensalidade);
                parametros[3] = new SqlParameter("cidadeId", aluno.CidadeId);
                parametros[4] = new SqlParameter("dataNascimento", aluno.DataNascimento);

                string sql =
                "update  alunos set  nome=@nome, mensalidade=@mensalidade, " +
                "cidadeId=@cidadeId, dataNascimento=@dataNascimento where id = @id";

                SqlCommand comando = new SqlCommand(sql, conexao);
                comando.Parameters.AddRange(parametros);
```

```
            comando.ExecuteNonQuery();
            conexao.Close();
        }
    }


    public static void Excluir(int id)
    {
        using (SqlConnection conexao = ConexaoBD.GetConexao())
        {
            SqlParameter[] parametros = new SqlParameter[1];
            parametros[0] = new SqlParameter("id", id);

            string sql = "delete alunos where id = @id";
            SqlCommand comando = new SqlCommand(sql, conexao);
            comando.Parameters.AddRange(parametros);
            comando.ExecuteNonQuery();
            conexao.Close();
        }
    }
}
}
```

Alterações no formulário para excluir e alterar

Para testar estes novos dois métodos, vamos alterar o formulário principal, incluindo 2 novos botões:



Código dos botões alterar e excluir:

```
    private void btnAlterar_Click(object sender, EventArgs e)
    {
        try
        {
            AlunoVO a = new AlunoVO();
            a.Id = Convert.ToInt32(txtId.Text);
            a.Nome = txtNome.Text;
            a.CidadeId = Convert.ToInt32(txtCidade.Text);
            a.DataNascimento = Convert.ToDateTime(txtData.Text);
            a.Mensalidade = Convert.ToDouble(txtMensalidade.Text);

            AlunoDAO.Alterar(a);
        }
        catch (Exception erro)
        {
            MessageBox.Show(erro.Message);
        }
    }

    private void btnExcluir_Click(object sender, EventArgs e)
    {
```

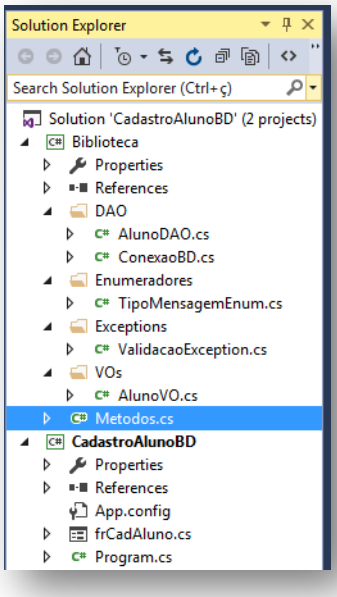
```
        try
        {
            AlunoDAO.Excluir(Convert.ToInt32(txtId.Text));
        }
        catch (Exception erro)
        {
            MessageBox.Show(erro.Message);
        }
    }
```

Exercícios:

1-) Modifique o cadastro de jogos, alterando sua classe DAO para que ele inclua, altere e exclua usando parâmetros no lugar de concatenar os valores.

Reduzindo o código repetido na classe DAO

Observe que há muito código repetido na classe DAO. Para reduzir, vamos criar uma classe estática (na raiz do projeto biblioteca) chamada Metodos, e nela iremos adicionar um método que será capaz de executar uma instrução SQL com parâmetros.




```
using Biblioteca.DAO;
using Biblioteca.Enumeradores;
using System;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace Biblioteca
{
    public static class Metodos
    {
        /// <summary>
        /// Método que executa uma instrução SQL no banco de dados
        /// </summary>
        /// <param name="sql"></param>
        /// <param name="parametros"></param>
        public static void ExecutaSQL(string sql, SqlParameter[] parametros)
        {
            using (SqlConnection conexao = ConexaoBD.GetConexao())
            {
                SqlCommand comando = new SqlCommand(sql, conexao);
                if (parametros != null)
                    comando.Parameters.AddRange(parametros);
                comando.ExecuteNonQuery();
                conexao.Close();
            }
        }
    }
}
```

Classe DAO com as modificações de redução de código

```
using Biblioteca.VOs;
using System;
using System.Data.SqlClient;

namespace Biblioteca.DAO
{
    public static class AlunoDAO
    {
```

```
        private static SqlParameter[] CriaParametros(AlunoVO aluno)
        {
            SqlParameter[] parametros = new SqlParameter[5];
            parametros[0] = new SqlParameter("id", aluno.Id);
            parametros[1] = new SqlParameter("nome", aluno.Nome);
            parametros[2] = new SqlParameter("mensalidade", aluno.Mensalidade);
            parametros[3] = new SqlParameter("cidadeId", aluno.CidadeId);
            parametros[4] = new SqlParameter("dataNascimento", aluno.DataNascimento);
            return parametros;
        }

        /// <summary>
        /// Método estático para inserir um aluno no BD
        /// </summary>
        /// <param name="aluno">objeto aluno com todas os atributos preenchidos</param>
        public static void Inserir(AlunoVO aluno)
        {
            string sql =
            "insert into alunos(id, nome, mensalidade, cidadeId, dataNascimento)" +
            "values ( @id, @nome, @mensalidade, @cidadeId, @dataNascimento)";

            Metodos.ExecutaSQL(sql, CriaParametros(aluno));
        }


        public static void Alterar(AlunoVO aluno)
        {
            string sql =
            "update  alunos set  nome=@nome, mensalidade=@mensalidade, " +
            "cidadeId=@cidadeId, dataNascimento=@dataNascimento where id = @id";
            Metodos.ExecutaSQL(sql, CriaParametros(aluno));
        }


        public static void Excluir(int id)
        {
            SqlParameter[] parametros = { new SqlParameter("id", id) };

            string sql = "delete alunos where id = @id";
            Metodos.ExecutaSQL(sql, parametros);
        }
    }
}
```

Exercícios:

2-) Aplique as reduções de código no cadastro de jogos.

4. Consultas

Para armazenarmos o conteúdo consultado de uma tabela, iremos utilizar um objeto da classe DataTable. Esta classe permite criar uma tabela “virtual” (apenas em memória), onde são armazenados os registros de uma tabela consultada.

No intuito de evitar a redundância de código, iremos criar o método para executar a instrução SQL da consulta na classe **Metodos**.

Adicione o código a seguir na classe **metodos**:

Obs: será necessário adicionar o namespace `using System.Data;`

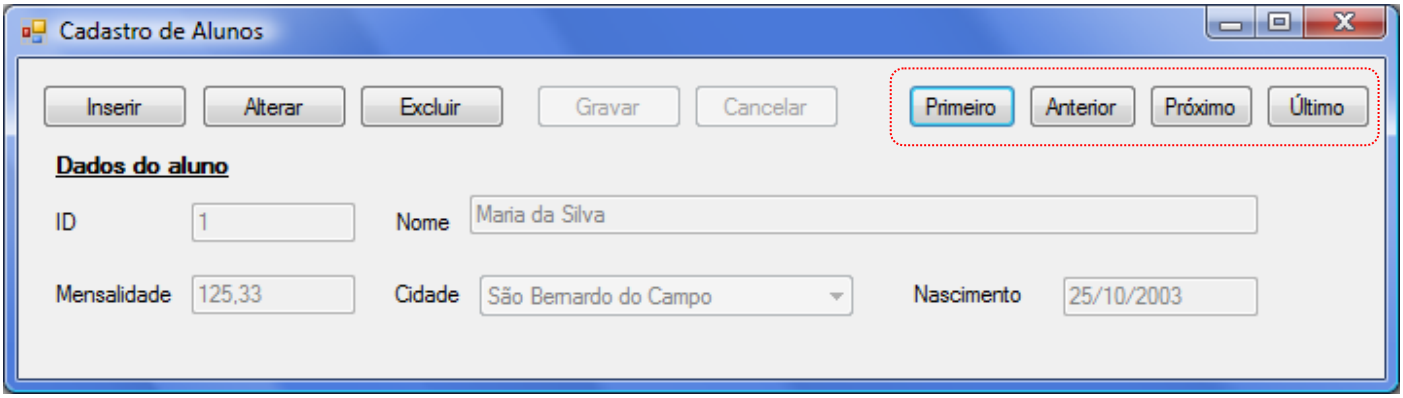
```
/// <summary>
/// Executa uma instrução Select
/// </summary>
/// <param name="sql">instrução SQL</param>
/// <returns>DataTable com os dados da instrução SQL</returns>
public static DataTable ExecutaSelect(string sql, SqlParameter[] parametros)
{
    using (SqlConnection conexao = ConexaoBD.GetConexao())
    {
        using (SqlDataAdapter adapter = new SqlDataAdapter(sql, conexao))
        {
            if (parametros != null)
                adapter.SelectCommand.Parameters.AddRange(parametros);

            DataTable tabelaTemp = new DataTable();
            adapter.Fill(tabelaTemp);
            conexao.Close();
            return tabelaTemp;
        }
    }
}
```

Para enviarmos as informações da camada DAO para a tela, vamos transformar os dados do DataTable em um VO. Crie o método abaixo na classe AlunoDAO.

```
/// <summary>
/// Recebe uma linha de um datable e preenche um objeto AlunoVO
/// </summary>
/// <param name="registro">1 registro (linha) do DataTable</param>
/// <returns>Objeto AlunoVO com os atributos  preenchidos</returns>
public static AlunoVO MontaVO(DataRow registro)
{
    AlunoVO aluno = new AlunoVO();

    aluno.Id = Convert.ToInt32(registro["id"]);
    aluno.Nome = registro["nome"].ToString();
    aluno.CidadeId = Convert.ToInt32(registro["cidadeId"]);
    aluno.Mensalidade = Convert.ToDouble(registro["mensalidade"]);
    aluno.DataNascimento = Convert.ToDateTime(registro["DataNascimento"]);

    return aluno;
}
```

Navegação nos Registros – Ajustes na classe DAO

Queremos chegar no modelo abaixo na tela de cadastro. Observe que há botões para navegar nos registros. É disso que trataremos a seguir.



Vamos criar os seguintes métodos:

- Primeiro = Vai para o primeiro registro da tabela
- Ultimo = Vai para o último registro da tabela
- Anterior = Volta para o registro anterior
- Proximo = Vai para o próximo registro

Inclua todos os métodos a seguir na classe AlunoDAO

```
public static AlunoVO Primeiro()
{
    string sql = "select top 1 * from Alunos order by id";
    DataTable tabela = Metodos.ExecutaSelect(sql, null);
    return ObjetoOuNull(tabela);
}


public static AlunoVO Ultimo()
{
    string sql = "select top 1 * from Alunos order by id desc";
    DataTable tabela = Metodos.ExecutaSelect(sql, null);
    return ObjetoOuNull(tabela);
}

public static AlunoVO Proximo(int atual)
{
    string sql = @"select  top 1 * from Alunos where id > @Atual order by id ";
    SqlParameter[] p = { new SqlParameter("Atual", atual) };
    DataTable tabela = Metodos.ExecutaSelect(sql, p);
    return ObjetoOuNull(tabela);
}


public static AlunoVO Anterior(int atual)
{
    string sql = @"select  top 1 * from Alunos where id < @Atual order by id desc";
    SqlParameter[] p = { new SqlParameter("Atual", atual) };
    DataTable tabela = Metodos.ExecutaSelect(sql, p);
    return ObjetoOuNull(tabela);
}
```

```
private static AlunoVO ObjetoOuNull(DataTable tabela)
{
    if (tabela.Rows.Count == 0)
        return null;
    else
    {
        AlunoVO a = MontaVO(tabela.Rows[0]);
        return a;
    }
}
```

Navegação nos Registros – Ajustes no formulário

Adicione os botões de navegação na tela de cadastro e o código abaixo:

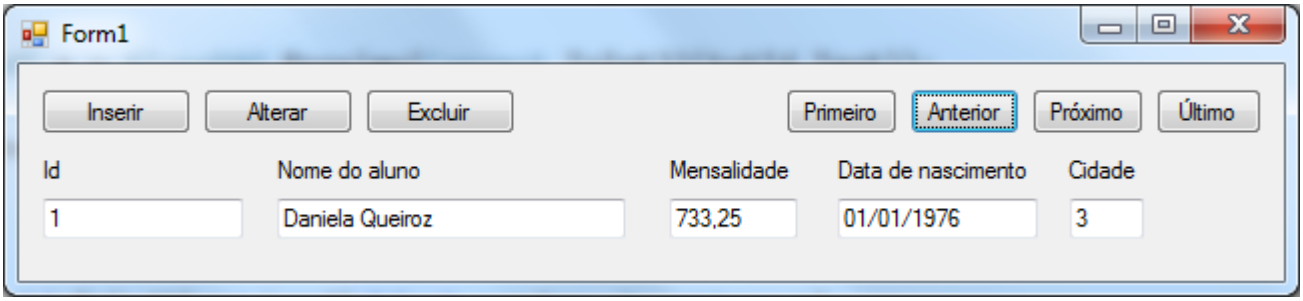
```
private void PreencheTela(AlunoVO a)
{
    if (a != null)
    {
        txtId.Text = a.Id.ToString();
        txtNome.Text = a.Nome;
        txtCidade.Text = a.CidadeId.ToString();
        txtData.Text = a.DataNascimento.ToShortDateString();
        txtMensalidade.Text = a.Mensalidade.ToString();
    }
    else
        LimpaCampos(this);
}

private void LimpaCampos(Control objeto)
{
    if (objeto is TextBox || objeto is MaskedTextBox)
        objeto.Text = "";
    else
        foreach (Control o in objeto.Controls)
            LimpaCampos(o);
}

private void btnPrimeiro_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Primeiro();
    PreencheTela(a);
}

private void btnUltimo_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Ultimo();
    PreencheTela(a);
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Anterior(Convert.ToInt32(txtId.Text));
    if (a != null)
        PreencheTela(a);
}
```

```
private void btnProximo_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Proximo(Convert.ToInt32(txtId.Text));
    if (a != null)
        PreencheTela(a);
}
```

No evento Load da tela, vamos posicionar no primeiro registro:

```
private void frCadAluno_Load(object sender, EventArgs e)
{
    btnPrimeiro.PerformClick();
}
```

A esta altura, sua dela deve estar similar a imagem abaixo:



Exercícios:

1-) Faça a navegação dos registros no cadastro de jogos.

5. Criando um cadastro de Aluno – parte 3

Crie o enumerador `EnumModoOperacao` no projeto Biblioteca, na pasta Enumeradores:

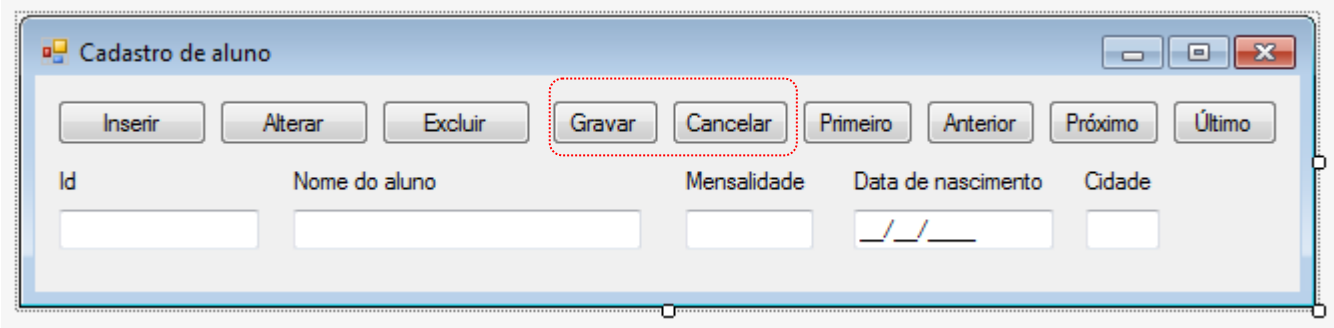
```
namespace Biblioteca.Enumeradores
{
    public enum EnumModoOperacao
    {
        Navegacao,
        Inclusao,
        Alteracao
    }
}
```

Na classe métodos, vamos adicionar mais alguns métodos utilitários para exibir mensagens, fazer validações, etc.

```
    public static class Metodos
    {
        /// <summary>
        /// Método que testa se um determinado valor string contém um inteiro válido
        /// </summary>
        /// <param name="valor">valor string a ser testado</param>
        /// <returns>true se for inteiro ou false caso contrário </returns>
        public static bool ValidaInt(string valor)
        {
            int valorConvertido;
            return int.TryParse(valor, out valorConvertido);
        }


        /// <summary>
        /// Método que testa se um determinado valor string contém um double válido
        /// </summary>
        /// <param name="valor">valor string a ser testado</param>
        /// <returns>true se for double ou false caso contrário </returns>
        public static bool ValidaDouble(string valor)
        {
            double valorConvertido;
            return double.TryParse(valor, out valorConvertido);
        }


        /// <summary>
        /// Verifica se uma data passada via parâmetro é valida
        /// </summary>
        /// <param name="data">data no formato string</param>
        /// <returns>True se for válida ou false caso contrário</returns>
        public static bool ValidaData(string data)
        {
            DateTime valorConvertido;
            return DateTime.TryParse(data, out valorConvertido);
        }

        /// <summary>
        /// Exibe uma mensagem
        /// </summary>
        /// <param name="mensagem">Texto da mensagem</param>
        /// <param name="tipoDaMensagem">tipo da mensagem</param>
        /// <returns>Quando for mensagem de confirmação, returna true se o
        /// usuário clicar em sim e retorna False caso clique em não</returns>
        public static bool Mensagem(string mensagem, TipoMensagemEnum tipoDaMensagem)
        {
            if (tipoDaMensagem == TipoMensagemEnum.alerta)
            {
                MessageBox.Show(mensagem, "Atenção", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return true;
            }
```

```
            else if (tipoDaMensagem == TipoMensagemEnum.erro)
            {
                MessageBox.Show(mensagem, "Erro", MessageBoxButtons.OK,  MessageBoxIcon.Error);
                return true;
            }
            else if (tipoDaMensagem == TipoMensagemEnum.informacao)
            {
                MessageBox.Show(mensagem, "Informação", MessageBoxButtons.OK, MessageBoxIcon.Information);
                return true;
            }
            else
            {
                if (MessageBox.Show(mensagem, "Atenção", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                    DialogResult.Yes)
                    return true;
                else
                    return false;
            }
        }
```

No formulário, adicione os botões de gravar e cancelar, como mostra a imagem abaixo:



Código completo do formulário:

```
using Biblioteca;
using Biblioteca.DAO;
using Biblioteca.Exceptions;
using Biblioteca.VOs;
using Biblioteca.Enumeradores;
using System;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace CadastroAlunoBD
{
    public partial class frCadAluno : Form
    {
        public frCadAluno()
        {
            InitializeComponent();
        }


        /// <summary>
        /// Centraliza o tratamento das exceções
        /// </summary>
        /// <param name="erro"></param>
        private void TrataErro(Exception erro)
        {
            if (erro is FormatException)
            {
```

```
            Metodos.Mensagem("Campo numérico inválido!", TipoMensagemEnum.erro);
        }
        else if (erro is ValidacaoException)
        {
            Metodos.Mensagem(erro.Message, TipoMensagemEnum.erro);
        }
        else if (erro is SqlException)
        {
            Metodos.Mensagem("Ocorreu um erro no banco de dados. Detalhes: \r\n" +
                             erro.Message, TipoMensagemEnum.erro);
        }
        else if (erro is Exception)
        {
            Metodos.Mensagem("Ocorreu um erro desconhecido!", TipoMensagemEnum.erro);
        }
    }


    /// <summary>
    /// Preenche os campos da tela com base em um objeto aluno,
    /// se o objeto for nulo, os campos serão limpos
    /// </summary>
    /// <param name="a"></param>
    private void PreencheTela(AlunoVO a)
    {
        if (a != null)
        {
            txtId.Text = a.Id.ToString();
            txtNome.Text = a.Nome;
            txtCidade.Text = a.CidadeId.ToString();
            txtData.Text = a.DataNascimento.ToShortDateString();
            txtMensalidade.Text = a.Mensalidade.ToString();
        }
        else
            LimpaCampos(this);
    }

    /// <summary>
    /// Método recursivo que limpa os objetos do tipo TextBox e MaskedTextBox
    /// </summary>
    /// <param name="objeto">objeto a ser limpo ou container</param>
    private void LimpaCampos(Control objeto)
    {
        if (objeto is TextBox || objeto is MaskedTextBox)
            objeto.Text = "";
        else
            foreach (Control o in objeto.Controls)
                LimpaCampos(o);
    }



    /// <summary>
    /// Altera a tela para se adequar a um determindo modo de operação
    /// que pode ser inclusão, alteração ou consulta
    /// </summary>
    /// <param name="modo">modo para o qual se deseja alterar</param>
    private void AlteraParaModo(EnumModoOperacao modo)
    {
        txtNome.Enabled = (modo != EnumModoOperacao.Navegacao);
        txtMensalidade.Enabled = (modo != EnumModoOperacao.Navegacao);
        txtData.Enabled = (modo != EnumModoOperacao.Navegacao);
        txtCidade.Enabled = (modo != EnumModoOperacao.Navegacao);
        btnGravar.Enabled = (modo != EnumModoOperacao.Navegacao);
        btnCancelar.Enabled = (modo != EnumModoOperacao.Navegacao);
```

```
    btnInclui.Enabled = (modo == EnumModoOperacao.Navegacao);

    btnAnterior.Enabled =
    btnProximo.Enabled =
    btnAlterar.Enabled =
    btnExcluir.Enabled = (modo == EnumModoOperacao.Navegacao) &&
                         txtId.Text.Length > 0;

    btnPrimeiro.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnUltimo.Enabled = (modo == EnumModoOperacao.Navegacao);

    if (modo == EnumModoOperacao.Inclusao)
    {
        txtId.Enabled = true;
        LimpaCampos(this);
        txtId.Focus();
    }
    else
        txtId.Enabled = false;
}



/// <summary>
/// Efetua algumas validações básicas. Outras validações são
/// realizadas pela classe AlunoVO
/// </summary>
/// <returns>True se tudo estiver OK</returns>
private bool Validacoes()
{
    if (Metodos.ValidaInt(txtId.Text) == false)
    {
        Metodos.Mensagem("Digite apenas números no campo ID.",
                         TipoMensagemEnum.erro);
        return false;
    }

    if (Metodos.ValidaDouble(txtMensalidade.Text) == false)
    {
        Metodos.Mensagem("Digite apenas números no campo Mensalidade.",
                         TipoMensagemEnum.erro);
        return false;
    }

    if (Metodos.ValidaData(txtData.Text) == false)
    {
        Metodos.Mensagem("Data de nascimento inválida.", TipoMensagemEnum.erro);
        return false;
    }

    return true;
}


private void btnInclui_Click(object sender, EventArgs e)
{
    AlteraParaModo(EnumModoOperacao.Inclusao);
}

private void btnAlterar_Click(object sender, EventArgs e)
{
    AlteraParaModo(EnumModoOperacao.Alteracao);
}

private void btnExcluir_Click(object sender, EventArgs e)
```

```
{
    if (Metodos.ValidaInt(txtId.Text) == false)
    {
        Metodos.Mensagem("Digite apenas números no campo ID.",
                         TipoMensagemEnum.alerta);
        return;
    }

    if (Metodos.Mensagem("Confirma?", TipoMensagemEnum.pergunta))
    {
        try
        {
            AlunoDAO.Excluir(Convert.ToInt32(txtId.Text));
            btnPrimeiro.PerformClick();
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }
}

private void btnPrimeiro_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Primeiro();
    PreencheTela(a);
}

private void btnUltimo_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Ultimo();
    PreencheTela(a);
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Anterior(Convert.ToInt32(txtId.Text));
    if (a != null)
        PreencheTela(a);
}

private void btnProximo_Click(object sender, EventArgs e)
{
    AlunoVO a = AlunoDAO.Proximo(Convert.ToInt32(txtId.Text));
    if (a != null)
        PreencheTela(a);
}

private void frCadAluno_Load(object sender, EventArgs e)
{
    btnPrimeiro.PerformClick();
    AlteraParaModo(EnumModoOperacao.Navegacao);
}

private void btnGravar_Click(object sender, EventArgs e)
{
    if (!Validacoes())
        return;

    try
    {
        AlunoVO a = new AlunoVO();
        a.Id = Convert.ToInt32(txtId.Text);
        a.Nome = txtNome.Text;
        a.CidadeId = Convert.ToInt32(txtCidade.Text);
        a.DataNascimento = Convert.ToDateTime(txtData.Text);
```

```
                a.Mensalidade = Convert.ToDouble(txtMensalidade.Text);

                if (txtId.Enabled)
                    AlunoDAO.Inserir(a);
                else
                    AlunoDAO.Alterar(a);

                AlteraParaModo(EnumModoOperacao.Navegacao);
            }
            catch (Exception erro)
            {
                TrataErro(erro);
            }
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            PreencheTela(AlunoDAO.Primeiro());
            AlteraParaModo(EnumModoOperacao.Navegacao);
        }

    }
}
```

Exercícios:

1-) Aplique todos os conceitos vistos até o momento no cadastro de jogos.

2-) Crie, do zero, um cadastro de Jogadores. Tabela:

```
Create table JogadorFutebol
( id  int primary key,
  NumeroCamisa int,
  Nome varchar(50),
  TimeId  int  );
```

6. Criando um cadastro de Aluno – parte 4

Evitando erro de chave duplicada

O ideal é que o próprio sistema indique o próximo ID a ser utilizado, porém, nem sempre isso é possível (o usuário pode querer informar ele mesmo o valor do campo ID). Para evitar erro de chave duplicada nestes casos, podemos verificar (antes de gravar) se o ID já não existe e, caso exista, avisamos o usuário para trocá-lo.

Primeiro, vamos criar um método para consultar um aluno na classe `AlunoDAO`:

```
/// <summary>
/// Recebe o Id do aluno e retorna um objeto AlunoVO ou null
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
public static AlunoVO Consulta(int id)
{
    string sql = "select * from alunos where id  = @id";
    SqlParameter[] parametros = { new SqlParameter("id", id) };

    DataTable tabela = Metodos.ExecutaSelect(sql, parametros);
    return ObjetoOuNull(tabela);
}
```

Agora, vamos alterar o método inserir na classe `AlunoDAO` para que ele, antes de executar a inclusão, verifique se o Id já existe.

```
/// <summary>
/// Método estático para inserir um aluno no BD
/// </summary>
/// <param name="aluno">objeto aluno com todas os atributos preenchidos</param>
public static void Inserir(AlunoVO aluno)
{
    if (Consulta(aluno.Id) != null)
        throw new ValidacaoException("Este código já está sendo utilizado!");

    string sql =
    "insert into alunos(id, nome, mensalidade, cidadeId, dataNascimento)" +
    "values ( @id, @nome, @mensalidade, @cidadeId, @dataNascimento)";

    Metodos.ExecutaSQL(sql, CriaParametros(aluno));
}
```

Sugerindo um código de aluno automaticamente

Como dito no tópico anterior, o próprio sistema indicar o próximo ID de aluno disponível durante uma inclusão.

Vamos alterar o programa para que, assim que o usuário clicar no botão incluir, o sistema automaticamente preencha o campo ID com o próximo valor disponível.

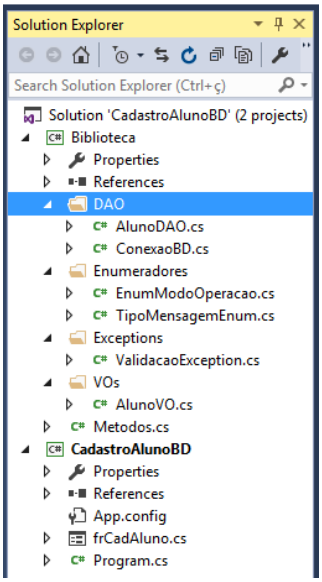
Inclua o seguinte método na classe AlunoDAO:

```
/// <summary>
/// Retorna o próximo Id
/// </summary>
/// <returns></returns>
public static int ProximoId()
{
    string sql = "select isnull(max(id) +1, 1) as 'MAIOR' from alunos";

    DataTable tabela = Metodos.ExecutaSelect(sql, null);
    return Convert.ToInt32(tabela.Rows[0]["MAIOR"]);
}
```

No botão de inserir, coloque as linhas abaixo em amarelo:

```
private void btnInserir_Click(object sender, EventArgs e)
  {
      AlteraParaModo(modoOperacao.inclusao);
      txtId.Text = AlunoDAO.ProximoID().ToString();
      txtId.Focus();
  }
```

Visão atual da solução



7. Recuperando o último valor inserido de um campo auto numeração (identity)

https://docs.microsoft.com/pt-br/sql/t-sql/functions/scope-identity-transact-sql

Por exemplo, dada a tabela abaixo, que possui um campo auto numeração (identity):

```
CREATE TABLE tbPessoa(
          id int NOT NULL primary key Identity (1,1),
          nome varchar(100) NULL)
```

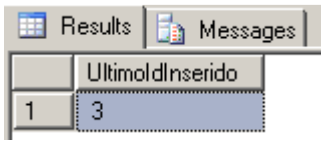
Após inserir um ou mais registros:

```
insert into tbPessoa( nome)  values ('Ana');
insert into tbPessoa( nome)  values ('Maria');
insert into tbPessoa( nome)  values ('Daniela');
```

Para saber qual foi o último valor identity inserido, execute uma consulta no banco de dados com a instrução abaixo:

```
select @@identity as 'UltimoIdInserido'
```

No C# , basta executar o comando acima como se fosse uma consulta normal, e pegar o retorno através do campo `'UltimoIdInserido'.` Observe que o comando acima irá retornar apenas 1 registro no BD:



8. Listando valores de uma tabela em um ComboBox

Para exemplificar, vamos utilizar o cadastro criado no tópico anterior, listando as cidades no combobox. Primeiro, vamos criar uma tabela para as cidades:

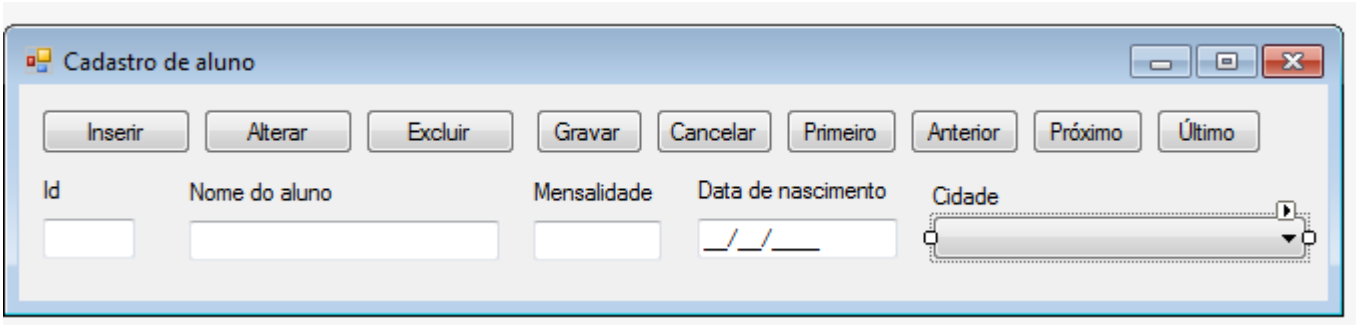
```
CREATE TABLE cidades(
       id int  NOT NULL primary key,
       nome varchar(30) NULL,
);

Insira os seguintes registros:

insert into cidades (id,nome) values (1, 'São Bernardo')
insert into cidades (id,nome) values (2, 'Santo André')
insert into cidades (id,nome) values (3, 'Maua')
insert into cidades (id,nome) values (4, 'Diadema')
insert into cidades (id,nome) values (5, 'Ribeirão Pires')
```

Alguns objetos, como o ComboBox, possuem a propriedade DataSource. Esta propriedade permite que seja associado uma lista ou um DataTable e, no caso do ComboBox, os valores serão listados.A seguir, veja as modificações que foram realizadas no programa:

Criar a classe CidadeDAO, na mesma pasta que está o AlunoDAO:

```
using System.Data;

namespace Biblioteca.DAO
{
    public static class CidadeDAO
    {
        /// <summary>
        /// Retorna um DataTable com todas as cidades por ordem alfabética
        /// </summary>
        /// <returns>DataTable com todas as cidades por ordem alfabética</returns>
        public static DataTable ListaCidades()
        {
            string sql = "select * from cidades order by nome";
            DataTable tabela = Metodos.ExecutaSelect(sql, null);
            return tabela;
        }
    }
}
```

Alterações no formulário



```
private void PreencheTela(AlunoVO a)
{
    if (a != null)
    {
        txtId.Text = a.Id.ToString();
        txtNome.Text = a.Nome;
        cbCidade.SelectedValue = a.CidadeId;
        txtData.Text = a.DataNascimento.ToShortDateString();
        txtMensalidade.Text = a.Mensalidade.ToString();
    }
    else
        LimpaCampos(this);
}


private void AlteraParaModo(EnumModoOperacao modo)
{
    txtNome.Enabled = (modo != EnumModoOperacao.Navegacao);
    txtMensalidade.Enabled = (modo != EnumModoOperacao.Navegacao);
    txtData.Enabled = (modo != EnumModoOperacao.Navegacao);
    cbCidade.Enabled = (modo != EnumModoOperacao.Navegacao);
    btnGravar.Enabled = (modo != EnumModoOperacao.Navegacao);




private void btnGravar_Click(object sender, EventArgs e)
{
    if (!Validacoes())
        return;

    try
    {
        AlunoVO a = new AlunoVO();
        a.Id = Convert.ToInt32(txtId.Text);
        a.Nome = txtNome.Text;
        a.CidadeId = Convert.ToInt32(cbCidade.SelectedValue);
        a.DataNascimento = Convert.ToDateTime(txtData.Text);


private void frCadAluno_Load(object sender, EventArgs e)
{
    cbCidade.DataSource = CidadeDAO.ListaCidades(); // datatable com todas as cidades
    cbCidade.DisplayMember = "nome"; // campo que será listado no combobox (usuário vê)
    cbCidade.ValueMember = "id"; // campo que irá representar o valor que foi escolhido

    btnPrimeiro.PerformClick();
    AlteraParaModo(EnumModoOperacao.Navegacao);
}
```

Exercícios:

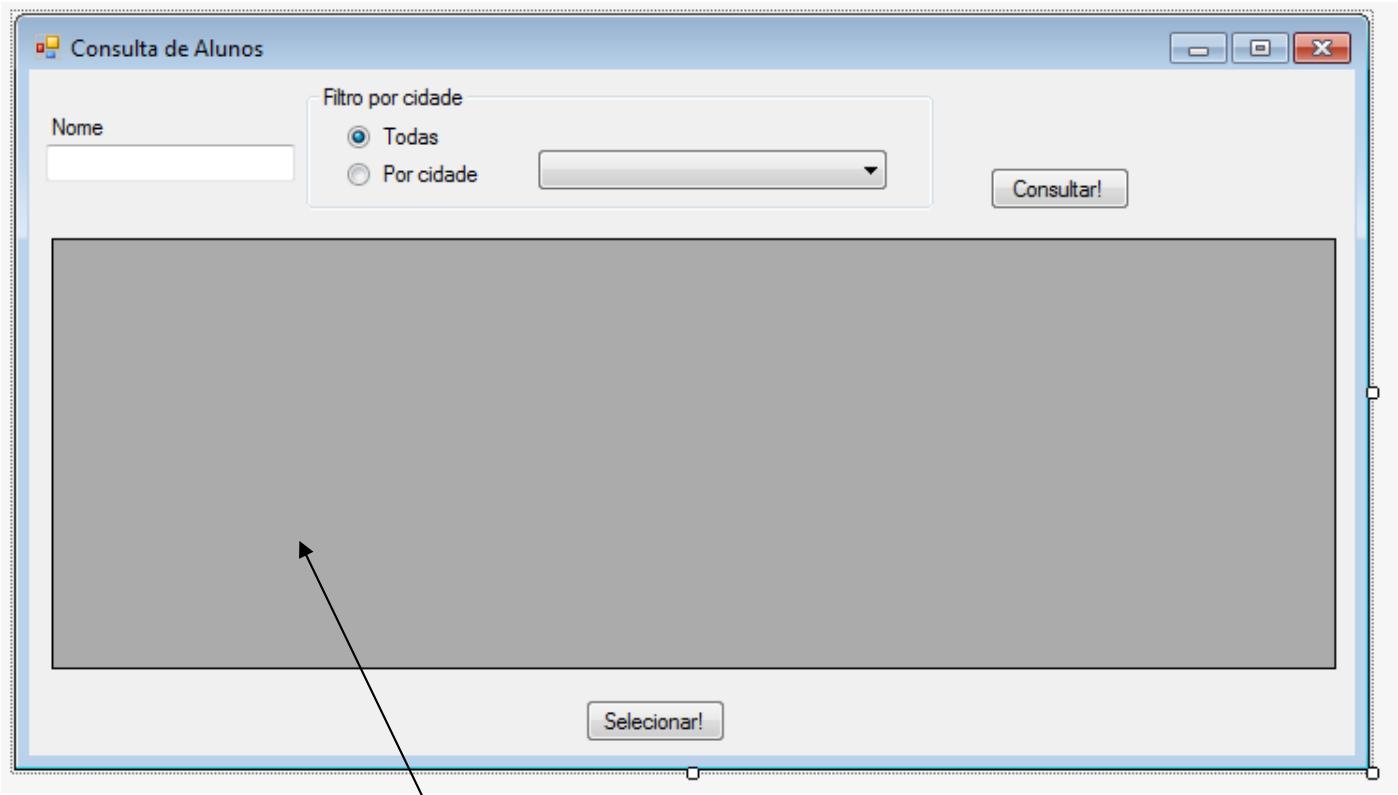
1-) Aplique no cadastro de jogos alterações para sugerir o próximo código e evitar erro de chave duplicada. Não permita Id repetido. Adicione um combobox no cadastro de jogos, para listar as categorias. Tabela de categorias:

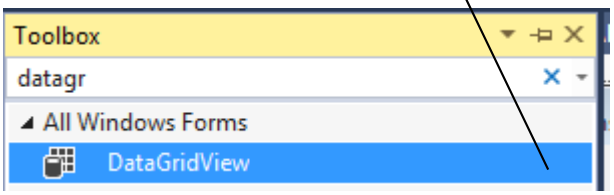
```
Create table Categorias (id int primary key, descricao  varchar(max) );
insert into Categorias values (1, 'Ação');
insert into Categorias values (2, 'RPG');
insert into Categorias values (3, 'Corrida');
insert into Categorias values (4, 'Aventura');
insert into Categorias values (5, 'Tiro');
```

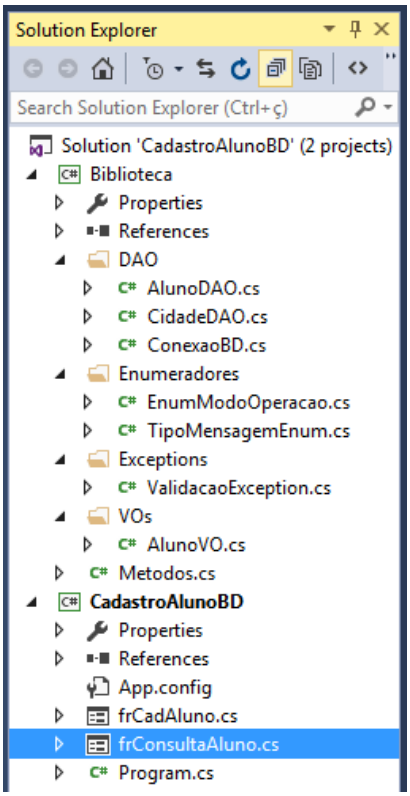
9. Criando uma tela de Consulta

Vamos criar uma tela para efetuar a pesquisa de alunos. Serão permitidos 2 filtros. Pelo nome do aluno (completo ou parcial) e por cidade (todas as cidades ou uma selecionada). O usuário poderá selecionar um aluno dentre aqueles listados no Grid e este aluno será localizado e posicionado na tela de cadastro de alunos.

Adicione um novo formulário ao seu projeto: (frConsultaAluno)







Código fonte do formulário:

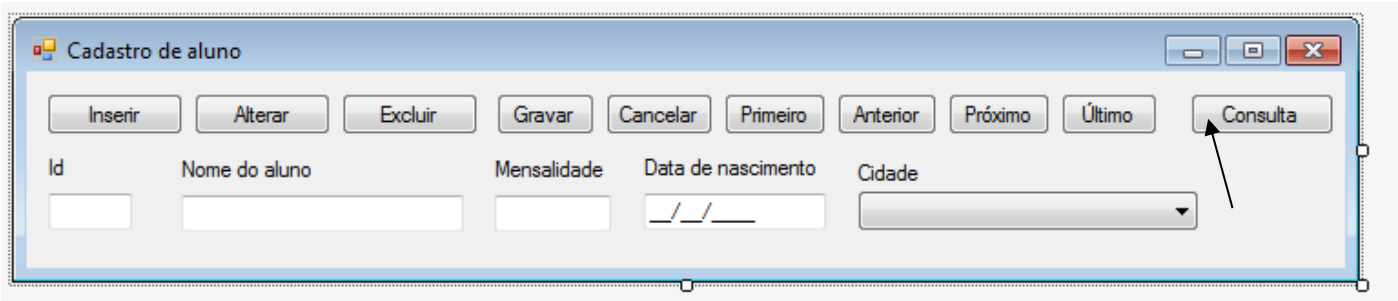
```
using Biblioteca.DAO;
using System;
using System.Data;
using System.Windows.Forms;

namespace CadastroAlunoBD
{
    public partial class frConsultaAluno : Form
    {
        public int IdSelecionado = 0;

        public frConsultaAluno()
        {
            InitializeComponent();

            cbCidade.DataSource = CidadeDAO.ListaCidades();
            cbCidade.DisplayMember = "nome";
            cbCidade.ValueMember = "id";

            gvAlunos.AllowUserToAddRows = false;
            gvAlunos.AllowUserToDeleteRows = false;
            gvAlunos.AllowUserToResizeRows = false;
            gvAlunos.MultiSelect = false;
            gvAlunos.ReadOnly = true;
            gvAlunos.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
        }

        private void rbTodas_CheckedChanged(object sender, EventArgs e)
        {
            cbCidade.Enabled = rbPorCidade.Checked;
        }

        private void rbPorCidade_CheckedChanged(object sender, EventArgs e)
        {
            cbCidade.Enabled = rbPorCidade.Checked;
        }

        private void btnPesquisa_Click(object sender, EventArgs e)
        {
            int cidade = 0;
            if (rbPorCidade.Checked)
                cidade = (int)cbCidade.SelectedValue;

            DataTable tabela = AlunoDAO.ListaAlunos(txtNome.Text, cidade);
            gvAlunos.DataSource = tabela;
        }

        private void btnSelecionar_Click(object sender, EventArgs e)
        {
            if (gvAlunos.CurrentRow != null)
            {
                IdSelecionado = (int)gvAlunos.CurrentRow.Cells[0].Value;
                Close();
            }
        }

        private void gvAlunos_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
        {
            btnSelecionar.PerformClick();
        }
    }
}
```

Na classe `AlunoDAO`, insira o seguinte método:

```
/// <summary>
/// Retorna uma datatable com os alunos que satisfaçam o filtro de pesquisa
/// </summary>
/// <param name="nome">nome do aluno, total ou parcial</param>
/// <param name="cidadeId">codigo da cidade ou Zero para todas</param>
/// <returns>datatable com os alunos que satisfaçam o filtro de pesquisa</returns>
public static DataTable ListaAlunos(string nome, int cidadeId)
{
    List<SqlParameter> parametros = new List<SqlParameter>();
    parametros.Add(new SqlParameter("nome", "%" + nome + "%"));
    string where = "";
    where = "Where alunos.nome like @nome ";
    if (cidadeId != 0)
    {
        where = where + " and cidades.id = @cidade";
        parametros.Add(new SqlParameter("cidade", cidadeId));
    }
    string sql =
    "select alunos.id as 'ID', alunos.nome as 'Nome do Aluno'," +
    "cidades.nome as 'Cidade' " +
    "from alunos " +
    "inner join cidades on alunos.cidadeId = cidades.Id " +
    where + " order by alunos.nome ";
    return Metodos.ExecutaSelect(sql, parametros.ToArray());
}
```

10. Criando um cadastro de Aluno – parte 4

Faça agora as seguintes alterações no cadastro do aluno para utilizar a tela de pesquisa:




```
        private void btnConsulta_Click(object sender, EventArgs e)
        {
            using (frConsultaAluno formConsulta = new frConsultaAluno())
            {
                formConsulta.ShowDialog();
                if (formConsulta.IdSelecionado != 0)
                    PreencheTela(AlunoDAO.Consulta(formConsulta.IdSelecionado));
            }
        }
```

Adicione no método abaixo a linha amarela:

```
        private void AlteraParaModo(EnumModoOperacao modo)
        {
            btnConsulta.Enabled = (modo == EnumModoOperacao.Navegacao);
            txtNome.Enabled = (modo != EnumModoOperacao.Navegacao);
```

Exercícios:

1-) Faça tela de uma pesquisa para o cadastro de jogos. Permita os seguintes filtros: Por código, descrição, categoria e data. Nenhum dos campos é obrigatório. Adicione o botão na tela de cadastro para chamar a pesquisa, e retornando dela, posicione no jogo selecionado.

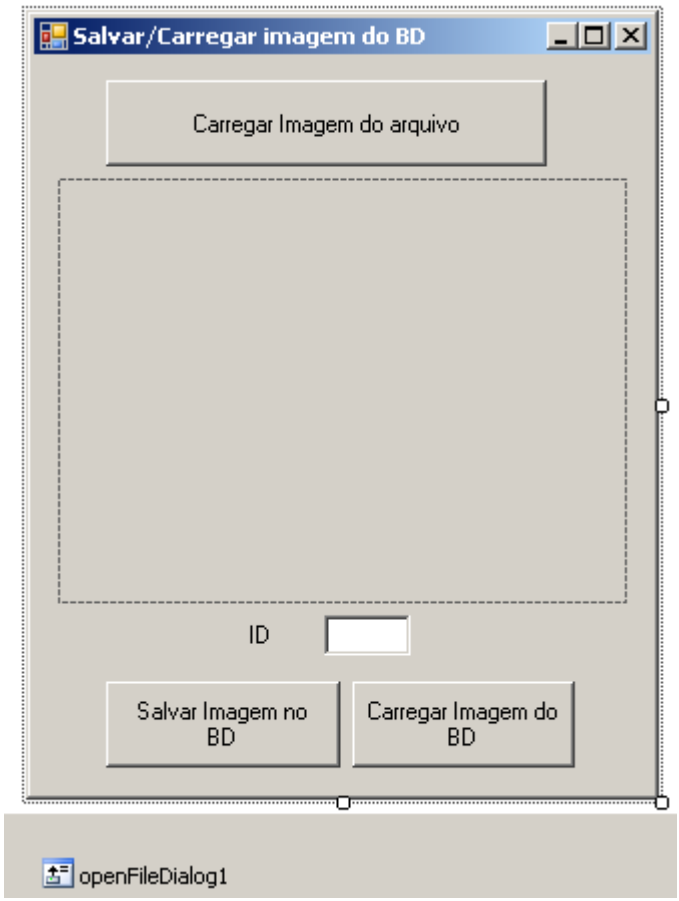
11. Salvando e Recuperando Imagens no Banco de Dados

Para realizar o exemplo abaixo, crie a seguinte tabela:

```
CREATE TABLE [dbo].[TbImagens](
          [id] [int] NOT NULL primary key,
          [imagem] [image] NULL)
```

Para fins didáticos, vamos fazer todo o código no formulário principal.

Crie uma tela como a abaixo:




```
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Data.SqlClient;

namespace ImagensCSharp
{

    public partial class Form1 : Form
    {
```

```
public Form1()
{
    InitializeComponent();
}

private void btnCarregarIMG_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        pictureBox1.ImageLocation = openFileDialog1.FileName;
    }
}


private SqlConnection GetConexao()
{
    SqlConnection conexao = new SqlConnection("Data Source=localhost;Initial " +
           "Catalog=AulaDB;Integrated Security=False; user id=sa;password=123456");
    conexao.Open();
    return conexao;
}



private void btnSalvar_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(pictureBox1.ImageLocation))
    {
        MessageBox.Show("Selecione uma imagem.");
        return;
    }
    using (SqlConnection conexao = GetConexao())
    {
        //O método abaixo obriga a ler os bytes do arquivo novamente
        //byte[] vetor = File.ReadAllBytes(pictureBox1.ImageLocation);

        //desta forma, não é necessário carregar o arquivo novamente
        ImageConverter converter = new ImageConverter();
        byte[] vetor = (byte[])converter.ConvertTo(pictureBox1.Image, typeof(byte[]));


        string sql = "insert into TbImagens (id,imagem) " +
                     "values (@id, @imagem)"; // @imagem é um parâmetro

        int id = Convert.ToInt16(txtID.Text);

        SqlParameter[] parametros = { new SqlParameter("@id", id),
                                      new SqlParameter("@imagem", vetor) };

        using (SqlCommand comando = new SqlCommand(sql, conexao))
        {
            comando.Parameters.AddRange(parametros);
            comando.ExecuteNonQuery();
        }
        MessageBox.Show("Gravado com sucesso!");
        conexao.Close();
    }
}

private void btnCarregaImg_Click(object sender, EventArgs e)
{
    //só para testar, carrega a imagem apenas do 1º registro da tabela
    using (SqlConnection conexao = GetConexao())
    {
        string sql = "select * from tbImagens where id = " + txtID.Text;
        using (SqlDataAdapter adapter = new SqlDataAdapter(sql, conexao))
```

```
                {
                    DataTable tabela = new DataTable();
                    adapter.Fill(tabela);

                    if (tabela.Rows.Count == 0)
                    {
                        MessageBox.Show("Não há registros na tabela!");
                        return;
                    }

                    // uma imagem representada na linguagem é um vetor de bytes!
                    byte[] bytesImagem = (tabela.Rows[0]["imagem"] as byte[]);
                    pictureBox1.Image = Image.FromStream(new MemoryStream(bytesImagem));
                    conexao.Close();
                }
            }
        }

    }
}
```

Exercícios:

1. Crie um CRUD de fabricantes e ferramentas, em telas separadas, incluindo todos os conceitos vistos até o momento:

```
Create Table Fabricantes (id int not null primary key identity(1,1), descricao varchar(max));

CREATE TABLE [dbo].[ferramentas](
       Id [int] NOT NULL  primary key identity (1,1),
       descricao [varchar](50) NULL,
       FabricanteId  int,
       Foto varchar(max),  -- caminho em disco da imagem
);


insert into fabricantes (descricao) values ('Bosch');
insert into fabricantes (descricao) values ('Makita');
insert into fabricantes (descricao) values ('Vonder');
insert into fabricantes (descricao) values ('Black & Decker');
insert into fabricantes (descricao) values ('DeWalt');
```

Faça um menu para que o usuário possa acessar as duas telas de cadastro.

Na tela de cadastro de ferramentas, não utilize uma caixa combo para listar os fabricantes. No lugar disso, coloque uma caixa de texto para o usuário digitar o código do fabricante e um botão de busca, que ao ser pressionado deverá exibir um formulário de consulta de fabricantes, com opção de filtro por descrição.

Crie uma tela de consulta de ferramentas, com opção de filtro por descrição e/ou fabricante. Porém, na tela, deverá haver apenas 1 campo onde o usuário irá digitar o que deseja pesquisar, e deverão ser retornados todos os registros cujo nome da ferramenta ou nome do fabricante contenham em sua descrição o filtro informado.

O usuário deverá ver no grid da consulta, além de todos os campos da ferramenta, a descrição do fabricante no lugar do seu código (fk)

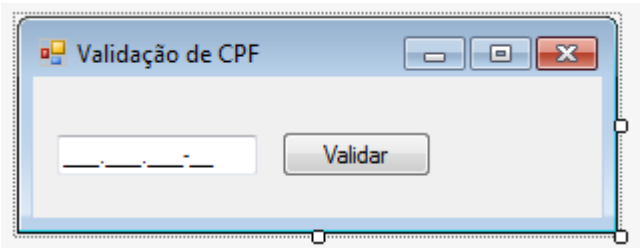
12. Executando Stored Procedures

Procedure utilizada neste exemplo: Uma procedure que valida CPF, retornando 0 se CPF inválido ou 1 caso contrário.

```
create procedure [dbo].[validaCPF] (@CPF as varchar(11) )
as
BEGIN
	-- declaração das variáveis locais
	declare @n int
	declare @soma int
	declare @multi int
	declare @digito1 int
	declare @digito2 int

	if len(rtrim(ltrim(@CPF))) <> 11
	begin
		Return 0
	end
	-- calculando o primeiro digito...
	set @soma = 0
	set @multi = 10
	set @n = 1

	WHILE (@n <= 9 )
	begin
		set @soma = @soma + cast(SUBSTRING(@cpf, @n, 1) as int) * @multi;
		set @multi = @multi -1;
		set @n = @n + 1
	end

	set @soma = @soma % 11 -- % -> módulo

	if @soma <=1
		set @digito1 = 0
	else
		set @digito1 = 11 - @soma

	--calculando o segundo digito...
	set @soma = 0
	set @multi = 11
	set @n = 1
	WHILE (@n <= 9 )
	begin
		set @soma = @soma + cast(SUBSTRING(@cpf, @n, 1) as int) * @multi;
		set @multi = @multi -1;
		set @n = @n + 1
	end
	set @soma = (@soma + @digito1 * @multi);
	set @soma = @soma % 11 -- % -> módulo
	if @soma <=1
		set @digito2 = 0
	else
		set @digito2 = 11 - @soma

	if	(cast(SUBSTRING(@cpf, 10, 1) as int) = @digito1) and
		(cast(SUBSTRING(@cpf, 11, 1) as int) = @digito2)
		Return 1
	else
		Return 0
END
```

Classe utilizada para executar as stored procedures.

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ExecutandoProcsBD
{
    static class SPs
    {
        public static bool ValidaCPF(String cpf)
        {
            using (SqlConnection cx = ConexaoBD.GetConexao())
            {
                using (SqlCommand cmd = new SqlCommand("validaCPF", cx))
                {
                    cmd.CommandType = System.Data.CommandType.StoredProcedure;
                    cmd.Parameters.Add(new SqlParameter("CPF", cpf.Replace(".", "").Replace("-", "")));
                    cmd.Parameters["CPF"].Direction = System.Data.ParameterDirection.Input;

                    cmd.Parameters.Add(new SqlParameter("retorno", SqlDbType.Bit));
                    cmd.Parameters["retorno"].Direction = System.Data.ParameterDirection.ReturnValue;

                    cmd.ExecuteNonQuery();

                    return Convert.ToBoolean(cmd.Parameters["retorno"].Value);
                }
            }
        }
    }
}
```



Código do botão validar:

```
bool retorno = SPs.ValidaCPF(maskedTextBox1.Text);
if (retorno)
  MessageBox.Show("Válido");
else
  MessageBox.Show("Inválido");
```

13. Salvando registros em lote: Passando um datatable para uma Stored Procedure

A tabela usada neste exemplo será esta:

```
CREATE TABLE Times(
       [id] [int] NOT NULL,
       [nome] [varchar](50) NULL);
```

Primeiro, precisamos criar um tipo que represente os dados que serão recebidos pela Stored Procedure. Observe que o tipo que estamos criando (TypeTime) possui os mesmos campos da tabela Times. Isso é proposital para facilitar o trabalho de não ter que mapear os campos da tabela recebida por parâmetro pela procedure.

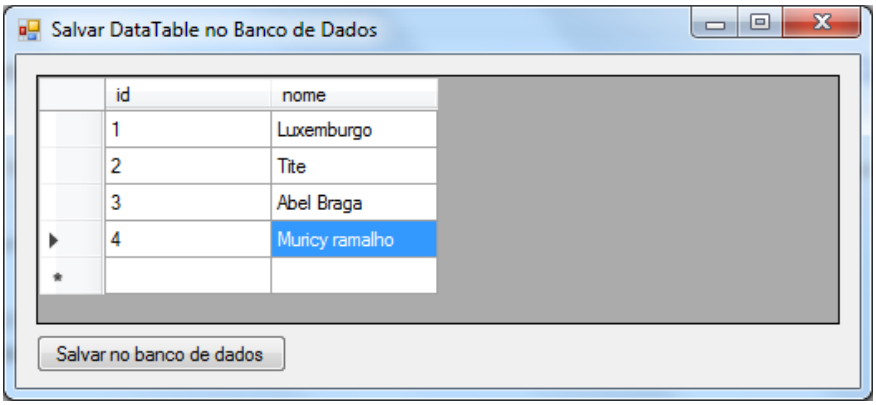
```
CREATE TYPE [TypeTime] AS TABLE(
    [Id] int NOT NULL,
    [Nome] [varchar](50) NULL);
```

Abaixo, está a stored procedure que irá receber uma variável do tipo tabela. A instrução readonly é necessária.

```
Create Procedure spIncluiTimes
    @tabela TypeTime readonly
AS
BEGIN
    select * into #TabTemp from @tabela;

    insert into times (id, nome)
       select * from #TabTemp
       where not exists(select times.id from times where id = #TabTemp.id)
END
```

Para que possamos fazer um subselect, é necessário primeiro jogar os dados para uma tabela temporária (#TabTemp). A procedure acima só irá inserir registros cujo Id não exista.

Agora, vamos montar a tela:



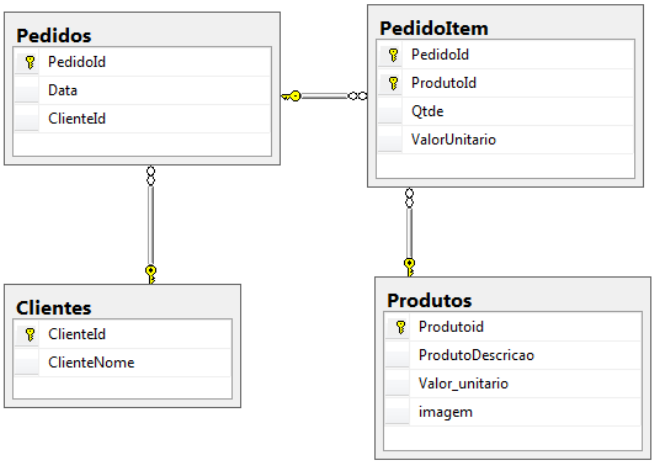
```
public partial class frCadTimes : Form
{

    public frCadTimes()
    {
        InitializeComponent();
        DataTable tabela = new DataTable();  // cria uma tabela temporária
        DataColumn c = new DataColumn("id", typeof(int)); // cria a coluna id
        tabela.Columns.Add(c);

        c = new DataColumn("nome", typeof(string));  // cria a coluna nome
        tabela.Columns.Add(c);

        dataGridView1.DataSource = tabela; // exibe no gridview
        dataGridView1.AutoGenerateColumns = true;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        using (SqlConnection  cx = ConexaoBD.GetConexao())
        {
            SqlCommand cmd = new SqlCommand("spIncluiTimes", cx);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.Add("@tabela", SqlDbType.Structured);
            cmd.Parameters[0].Value = dataGridView1.DataSource;
            cmd.ExecuteNonQuery();
        }

        MessageBox.Show("Salvo com sucesso!!!");
    }
}
```

Exercícios

1. Faça ajustes na última versão do cadastro de jogos para que a classe DAO realize todos os acessos ao banco de dados através de Stored procedures ou functions.

14. Desenvolvimento de um cadastro Mestre Detalhe

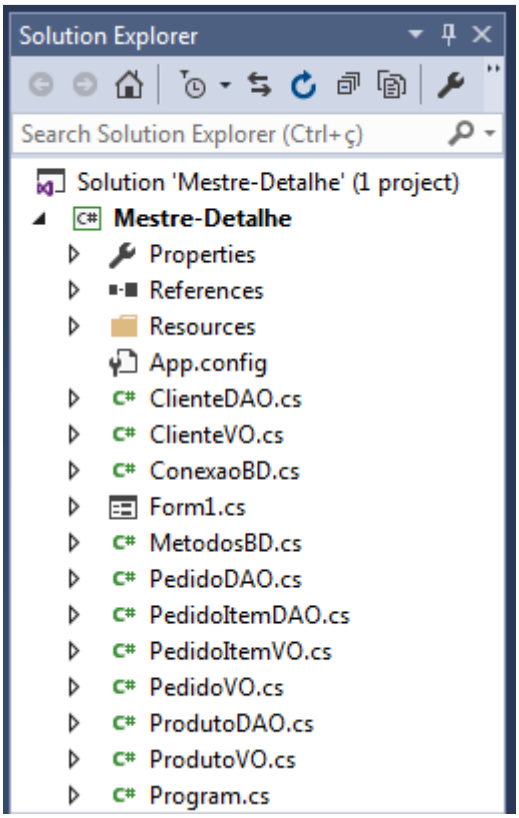
Script do Banco de dados




```
CREATE TABLE [dbo].[Clientes](
	[ClienteId] [int] NOT NULL,
	[ClienteNome] [varchar](50) NULL,
 CONSTRAINT [PK_Clientes] PRIMARY KEY CLUSTERED
(
	[ClienteId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO


CREATE TABLE [dbo].[Pedidos](
	[PedidoId] [int] NOT NULL,
	[Data] [datetime] NULL,
	[ClienteId] [int] NULL,
 CONSTRAINT [PK_Pedido] PRIMARY KEY CLUSTERED
(
	[PedidoId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO


CREATE TABLE [dbo].[Produtos](
	[Produtoid] [int] NOT NULL,
	[ProdutoDescricao] [varchar](50) NULL,
	[Valor_unitario] [decimal](18, 2) NULL,
	[imagem] [varchar](max) NULL,
 CONSTRAINT [PK_Produtos] PRIMARY KEY CLUSTERED
(
	[Produtoid] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO


CREATE TABLE [dbo].[PedidoItem](
	[PedidoId] [int] NOT NULL,
	[ProdutoId] [int] NOT NULL,
	[Qtde] [int] NOT NULL,
	[ValorUnitario] [decimal](18, 2) NOT NULL,
 CONSTRAINT [PK_PedidoItem] PRIMARY KEY CLUSTERED
(
	[PedidoId] ASC,
	[ProdutoId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

```
ALTER TABLE [dbo].[Pedidos]  WITH CHECK ADD  CONSTRAINT [FK_Pedido_Clientes] FOREIGN KEY([ClienteId])
REFERENCES [dbo].[Clientes] ([ClienteId])
GO

ALTER TABLE [dbo].[Pedidos] CHECK CONSTRAINT [FK_Pedido_Clientes]
GO

ALTER TABLE [dbo].[PedidoItem]  WITH CHECK ADD  CONSTRAINT [FK_PedidoItem_Pedido] FOREIGN KEY([PedidoId])
REFERENCES [dbo].[Pedidos] ([PedidoId])
GO

ALTER TABLE [dbo].[PedidoItem] CHECK CONSTRAINT [FK_PedidoItem_Pedido]
GO

ALTER TABLE [dbo].[PedidoItem]  WITH CHECK ADD  CONSTRAINT [FK_PedidoItem_Produtos] FOREIGN KEY([ProdutoId])
REFERENCES [dbo].[Produtos] ([Produtoid])
GO

ALTER TABLE [dbo].[PedidoItem] CHECK CONSTRAINT [FK_PedidoItem_Produtos]
GO


INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (1, N'Borracha', CAST(1.50 AS Decimal(18,
2)), N'')
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (2, N'Lápis', CAST(1.00 AS Decimal(18,
2)), N'~/imagens/lapis.png')
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (3, N'Caneta', CAST(3.50 AS Decimal(18,
2)), N'')
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (4, N'Caderno', CAST(7.00 AS Decimal(18,
2)), N'~/imagens/caderno.png')
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (5, N'Cola', CAST(3.50 AS Decimal(18, 2)),
NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (6, N'Canetinha preta', CAST(1.00 AS
Decimal(18, 2)), NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (7, N'Canetinha azul', CAST(1.00 AS
Decimal(18, 2)), NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (8, N'Canetinha rosa', CAST(1.00 AS
Decimal(18, 2)), NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (9, N'Canetinha amarela', CAST(1.00 AS
Decimal(18, 2)), NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (10, N'Pincel atômico', CAST(6.00 AS
Decimal(18, 2)), NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (11, N'Clips', CAST(0.20 AS Decimal(18,
2)), NULL)
GO
INSERT [dbo].[Produtos] ([Produtoid], [ProdutoDescricao], [Valor_unitario], [imagem]) VALUES (12, N'Grampeador', CAST(25.00 AS
Decimal(18, 2)), NULL)
GO

insert into Clientes (ClienteId, ClienteNome) values (1, 'Paola Oliveira')
insert into Clientes (ClienteId, ClienteNome) values (2, 'Cristiane Silva')
GO
```

Estrutura do projeto



Classe MetodosBD para execução de SQL

```
using System.Data;
using System.Data.SqlClient;

namespace Mestre_Detalhe
{
    public class MetodosBD
    {
        public static void ExecutaSQL(string sql, SqlParameter[] parametros)
        {
            using (SqlConnection conexao = ConexaoBD.GetConexao())
            {
                using (SqlCommand comando = new SqlCommand(sql, conexao))
                {
                    comando.Parameters.AddRange(parametros);
                    comando.ExecuteNonQuery();
                    conexao.Close();
                }
            }
        }

        public static DataTable ExecutaSelect(string sql, SqlParameter[] parametros)
        {
            using (SqlConnection cx = ConexaoBD.GetConexao())
            {
                using (SqlDataAdapter adapter = new SqlDataAdapter(sql, cx))
                {
```

```
                    if (parametros != null)
                        adapter.SelectCommand.Parameters.AddRange(parametros);
                    DataTable tabela = new DataTable();
                    adapter.Fill(tabela);

                    return tabela;
                }
            }
        }
    }
}
```

Classe ConexaoBD

```
using System.Data.SqlClient;

namespace Mestre_Detalhe
{
    public class ConexaoBD
    {
        public static SqlConnection GetConexao()
        {
            string strCon = "Data Source=localhost;Initial Catalog=AulaDB;Persist Security " +
                            "Info=True;User ID=sa;Password=123456";
            SqlConnection conexao = new SqlConnection(strCon);
            conexao.Open();
            return conexao;
        }
    }
}
```

Classes Vos

```
namespace Mestre_Detalhe
{
    class ClienteVO
    {
        public int Id { get; set; }
        public string Nome { get; set; }
    }
}


namespace Mestre_Detalhe
{
    class PedidoItemVO
    {
        public int PedidoId { get; set; }

        public int ProdutoId { get; set; }

        public int Qtde { get; set; }

        public double ValorUnitario { get; set; }


        #region Atributos que não existem na tabela
```

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

        public double ValorTotal
        {
            get
            {
                return ValorUnitario * Qtde;
            }
        }
        #endregion
    }
}
```

```
using System;
using System.Collections.Generic;

namespace Mestre_Detalhe
{
    class PedidoVO
    {
        public int Id { get; set; }

        public int ClienteId { get; set; }

        public DateTime Data { get; set; }

        public List<PedidoItemVO> ItensDoPedido { get; set; } = new List<PedidoItemVO>();
    }
}
```

```
using System;

namespace Mestre_Detalhe
{
    public class ProdutoVO
    {
        public int Codigo { get; set; }
        public String Nome { get; set; }
        public double ValorUnitario { get; set; }
    }
}
```

Classes DAO e Controle de Transação

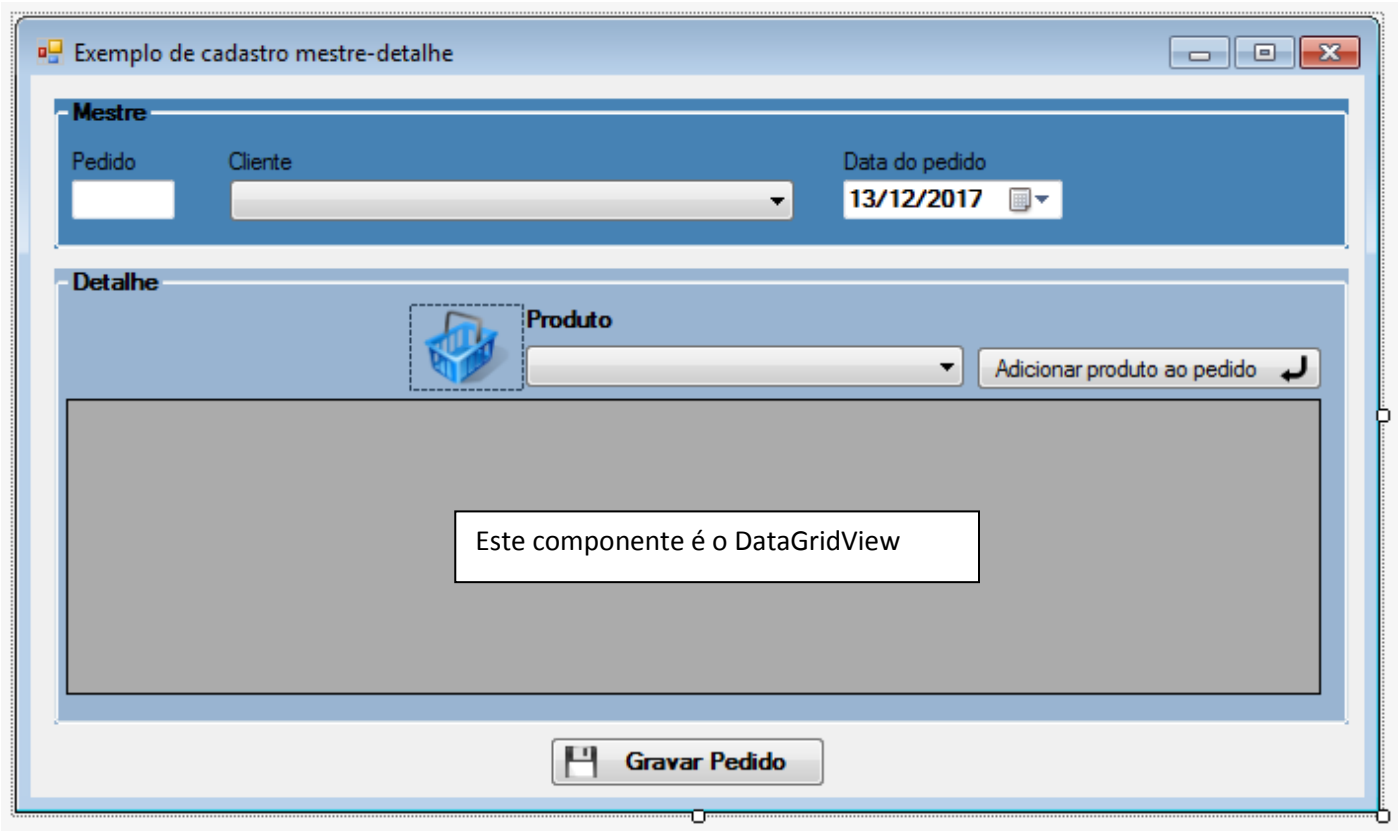
```
using System;
using System.Collections.Generic;
using System.Data;

namespace Mestre_Detalhe
{
    class ClienteDAO
    {
        public static List<ClienteVO> ConsultaClientes()
        {
            string sql = "select * from clientes order by ClienteNome";
            DataTable tabela = MetodosBD.ExecutaSelect(sql, null);
            List<ClienteVO> lista = new List<ClienteVO>();
            foreach (DataRow registro in tabela.Rows)
                lista.Add(MontaVO(registro));

            return lista;
        }

        private static ClienteVO MontaVO(DataRow registro)
        {
            ClienteVO obj = new ClienteVO();
            obj.Id = Convert.ToInt32(registro["clienteId"]);
            obj.Nome = registro["clienteNome"].ToString();
            return obj;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Data;

namespace Mestre_Detalhe
{
    public class ProdutoDAO
    {

        public static List<ProdutoVO> ConsultaProdutos()
        {
            string sql = "select * from produtos order by ProdutoDescricao";
            DataTable tabela = MetodosBD.ExecutaSelect(sql, null);
            List<ProdutoVO> lista = new List<ProdutoVO>();
            foreach (DataRow registro in tabela.Rows)
                lista.Add(MontaVO(registro));

            return lista;
        }

        private static ProdutoVO MontaVO(DataRow registro)
        {
            ProdutoVO p = new ProdutoVO();
            p.Codigo = Convert.ToInt32(registro["ProdutoId"]);
            p.Nome = registro["ProdutoDescricao"].ToString();
            p.ValorUnitario = Convert.ToDouble(registro["Valor_Unitario"]);
            return p;
        }
    }
}
```

```
using System.Data.SqlClient;

namespace Mestre_Detalhe
{
    static class PedidoDAO
    {
        /// <summary>
        /// Insere o pedido e seus itens
        /// </summary>
        /// <param name="pedido"></param>
        public static void Insere(PedidoVO pedido)
        {
              //transações
              //https://msdn.microsoft.com/pt-br/library/system.transactions.transactionscope(v=vs.110).aspx

              // É necessário adicionar a referência System.Transactions

              // início da transação (operação atômica)
              //Ou grava tudo, ou não grava nada
            using (var transacao = new System.Transactions.TransactionScope())
            {
                string sql = "insert into pedidos (pedidoId, clienteId, data) values " +
                             "(@pedidoId, @clienteid, @data)";
                MetodosBD.ExecutaSQL(sql, CriaParametros(pedido));

                foreach (PedidoItemVO item in pedido.ItensDoPedido)
                {
                    item.PedidoId = pedido.Id;
                    PedidoItemDAO.InsereItemPedido(item);
                }

                 // efetua o commit. Se der uma exception neste código, ou ele não
                 // passar por esta linha, será considerado rollback
                transacao.Complete();
            }
        }


        private static SqlParameter[] CriaParametros(PedidoVO pedido)
        {
            SqlParameter[] parametros = new SqlParameter[3];
            parametros[0] = new SqlParameter("pedidoId", pedido.Id);
            parametros[1] = new SqlParameter("clienteId", pedido.ClienteId);
            parametros[2] = new SqlParameter("data", pedido.Data);
            return parametros;
        }
    }
}
```

```
using System.Data.SqlClient;

namespace Mestre_Detalhe
{
    class PedidoItemDAO
    {
        public static void InsereItemPedido(PedidoItemVO item)
        {
            string sql = "insert into PedidoItem (PedidoId, ProdutoId,Qtde, ValorUnitario) " +
                "values (@PedidoId, @ProdutoId, @Qtde, @ValorUnitario)";
            MetodosBD.ExecutaSQL(sql, CriaParametros(item));
        }

        private static SqlParameter[] CriaParametros(PedidoItemVO item)
        {
            SqlParameter[] parametros = new SqlParameter[4];
            parametros[0] = new SqlParameter("pedidoId", item.PedidoId);
            parametros[1] = new SqlParameter("ProdutoId", item.ProdutoId);
            parametros[2] = new SqlParameter("Qtde", item.Qtde);
            parametros[3] = new SqlParameter("ValorUnitario", item.ValorUnitario);
            return parametros;
        }

    }
}
```

Formulário



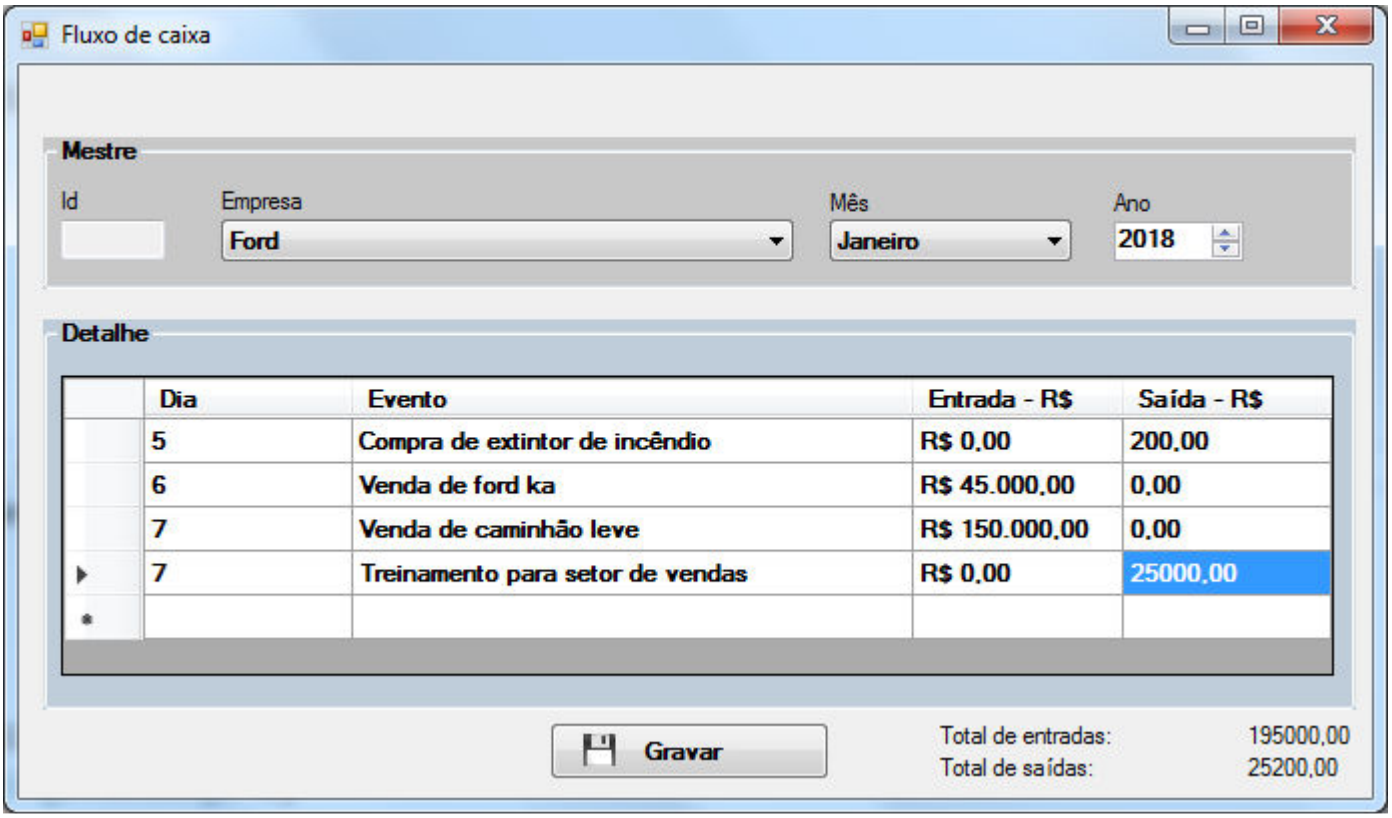
```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace Mestre_Detalhe
{
    public partial class Form1 : Form
    {
        BindingList<PedidoItemVO> carrinhoCompras = new BindingList<PedidoItemVO>();

        private void ConfiguraColunasGridView()
        {
            #region criação dinâmica das colunas do DataGridView

            DataGridViewTextBoxColumn colProdutoId = new DataGridViewTextBoxColumn();
            colProdutoId.Name = "colProdutoId";
            colProdutoId.ReadOnly = true;
            colProdutoId.Width = 100;
            colProdutoId.DataPropertyName = "ProdutoID"; // o nome do campo do datatable
            colProdutoId.HeaderText = "Produto";
            colProdutoId.DefaultCellStyle.BackColor = SystemColors.ButtonFace;

            dataGridView1.Columns.Add(colProdutoId);


            DataGridViewTextBoxColumn colProdutoDescricao = new DataGridViewTextBoxColumn();
            colProdutoDescricao.Name = "colProdutoDescricao";
            colProdutoDescricao.ReadOnly = true;
            colProdutoDescricao.Width = 200;
            colProdutoDescricao.DataPropertyName = "ProdutoDescricao"; // o nome do campo do datatable
            colProdutoDescricao.HeaderText = "Descrição";
            colProdutoDescricao.DefaultCellStyle.BackColor = SystemColors.ButtonFace;
            dataGridView1.Columns.Add(colProdutoDescricao);

            DataGridViewTextBoxColumn colQtde = new DataGridViewTextBoxColumn();
            colQtde.Name = "colQtde";
            colQtde.ReadOnly = false; // permite escrita nesse campo
            colQtde.Width = 80;
            colQtde.DataPropertyName = "Qtde"; // o nome do campo do datatable
            colQtde.HeaderText = "Quantidade";
            colQtde.DefaultCellStyle.BackColor = Color.LightBlue;
            dataGridView1.Columns.Add(colQtde);


            DataGridViewTextBoxColumn colValorUnitario = new DataGridViewTextBoxColumn();
            colValorUnitario.Name = "colValorUnitario";
            colValorUnitario.ReadOnly = true; // permite escrita nesse campo
            colValorUnitario.Width = 100;
            colValorUnitario.DataPropertyName = "ValorUnitario"; // o nome do campo do datatable
            colValorUnitario.HeaderText = "Valor Unitário";
            colValorUnitario.DefaultCellStyle.BackColor = SystemColors.ButtonFace;
            dataGridView1.Columns.Add(colValorUnitario);


            DataGridViewTextBoxColumn colValorTotal = new DataGridViewTextBoxColumn();
            colValorTotal.Name = "colValorTotal";
            colValorTotal.ReadOnly = true; // permite escrita nesse campo
            colValorTotal.Width = 100;
            colValorTotal.DataPropertyName = "ValorTotal"; // o nome do campo do datatable
            colValorTotal.HeaderText = "Valor Total";
            colValorTotal.DefaultCellStyle.BackColor = SystemColors.ButtonFace;
            dataGridView1.Columns.Add(colValorTotal);

            dataGridView1.AllowUserToAddRows = false;
            dataGridView1.AllowUserToDeleteRows = false;
            dataGridView1.AllowUserToResizeRows = false;
            dataGridView1.MultiSelect = false;
            dataGridView1.AutoGenerateColumns = false;


            dataGridView1.DataSource = carrinhoCompras;

            #endregion
        }
```

```
public Form1()
{
    InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
    ConfiguraColunasGridView();

    try
    {
        cbProduto.DisplayMember = "Nome";
        cbProduto.ValueMember = "Codigo";
        cbProduto.DataSource = ProdutoDAO.ConsultaProdutos();

        cbCliente.DisplayMember = "Nome";
        cbCliente.ValueMember = "Id";
        cbCliente.DataSource = ClienteDAO.ConsultaClientes();
        cbCliente.SelectedIndex = -1;
    }
    catch (Exception erro)
    {
        MessageBox.Show("Erro: " + erro.Message);
    }
}

private void btnAddProd_Click(object sender, EventArgs e)
{
    //pega o produto selecionado no combobox
    ProdutoVO produto = cbProduto.SelectedItem as ProdutoVO;

    PedidoItemVO produtoPesquisado =
                 carrinhoCompras.FirstOrDefault(item => item.ProdutoId == produto.Codigo);

    if (produtoPesquisado != null)
        produtoPesquisado.Qtde++; // apenas altera a quantidade
    else
    {
        //Insere o produto no carrinho (item do pedido)
        PedidoItemVO item = new PedidoItemVO();
        item.ProdutoDescricao = produto.Nome;
        item.ProdutoId = produto.Codigo;
        item.ValorUnitario = produto.ValorUnitario;
        item.Qtde = 1;
        carrinhoCompras.Add(item);
    }
    //dataGridView1.Invalidate();
}

private void dataGridView1_CellEndEdit(object sender, DataGridViewCellEventArgs e)
{
    //o objetivo deste código atualizar a coluna preço total.
    dataGridView1.InvalidateCell(dataGridView1.CurrentRow.Cells["colValorTotal"]);
}


private void dataGridView1_CellValidating(object sender,DataGridViewCellValidatingEventArgs e)
{
    if (dataGridView1.IsCurrentCellInEditMode)
    {
        //Validação para a coluna quantidade
        if (e.ColumnIndex == dataGridView1.Columns["colQtde"].Index)
        {
            try
            {
                Convert.ToInt32(e.FormattedValue);
                e.Cancel = false;
            }
            catch
            {
                e.Cancel = true;
                MessageBox.Show("Valor inválido! Pressione ESC para cancelar.", "Atenção",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

```
        private void btnGravaPedido_Click(object sender, EventArgs e)
        {
            //faça as validações aqui
            try
            {
                PedidoVO pedido = new PedidoVO();
                pedido.Id = Convert.ToInt32(txtPedidoID.Text);
                pedido.Data = txtData.Value;
                pedido.ClienteId = Convert.ToInt32(cbCliente.SelectedValue);
                pedido.ItensDoPedido = carrinhoCompras.ToList();

                PedidoDAO.Insere(pedido);
            }
            catch (Exception erro)
            {
                MessageBox.Show(erro.Message);
            }
        }
    }
}
```

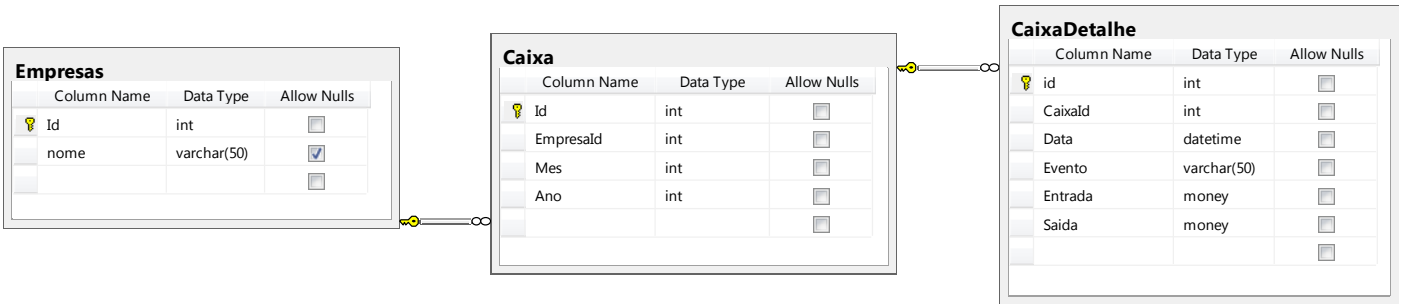
Exercícios

1. Faça um sistema de fluxo de caixa que tenha a seguinte aparência e funcionalidades:



Ao preencher ou alterar os valores no grid, atualize os labels de Total de entradas e Saídas.

Crie a tela, além das classes DAO e VO.
Utilize controle de transação.
Ao salvar, preencha na tela o Id do fluxo de caixa.



Script:

```
CREATE TABLE [Caixa](
       [Id] [int] IDENTITY(1,1) NOT NULL primary key,
       [EmpresaId] int not NULL,
       [Mes] [int] not NULL,
       [Ano] [int] not NULL
)

GO


CREATE TABLE [CaixaDetalhe](
       [id] [int] IDENTITY(1,1) NOT NULL primary key,
       [CaixaId] [int] NOT NULL,
       [Data] [datetime] NOT NULL,
       [Evento] [varchar](50) NOT NULL,
       [Entrada] [money] NOT NULL,
       [Saida] [money] NOT NULL)
GO



Create Table Empresas
(Id int  primary key,
 nome varchar(50)
 )
 GO


ALTER TABLE [dbo].[CaixaDetalhe]  WITH CHECK ADD  CONSTRAINT [FK_CaixaDetalhe_Caixa] FOREIGN
KEY([CaixaId])
REFERENCES [dbo].[Caixa] ([Id])
GO

ALTER TABLE [dbo].[CaixaDetalhe] CHECK CONSTRAINT [FK_CaixaDetalhe_Caixa]
GO



ALTER TABLE [dbo].[Caixa]  WITH CHECK ADD  CONSTRAINT [FK_Caixa_Empresas] FOREIGN KEY([EmpresaId])
REFERENCES [dbo].[Empresas] ([Id])
GO

ALTER TABLE [dbo].[Caixa] CHECK CONSTRAINT [FK_Caixa_Empresas]
GO
```

15. Herança aplicada à criação de CRUDs

Iremos a seguir criar um cadastro de técnicos utilizando conceitos de POO para redução de código.

A fim de simplificar o material da apostila, todos os arquivos foram colocados em um único projeto.

Pressuposto: A chave primária das tabelas deve ser um campo inteiro

Tabela:

```
CREATE TABLE [dbo].[Tecnicos](
	[Id] [int] NOT NULL primary key,
	[Nome] [varchar](50) NULL)
```

Enumeradores

```
public enum TipoMensagemEnum { alerta, erro, informacao, pergunta }

public enum EnumModoOperacao { Navegacao, Inclusao, Alteracao }
```

Classe Metodos

```
public static class Metodos
{
    public static void ExecutaSQL(string sql, SqlParameter[] parametros)
    {
        using (SqlConnection conexao = ConexaoBD.GetConexao())
        {
            using (SqlCommand comando = new SqlCommand(sql, conexao))
            {
                comando.Parameters.AddRange(parametros);
                comando.ExecuteNonQuery();
            }
            conexao.Close();
        }
    }



    public static DataTable ExecutaSelect(string sql, SqlParameter[] parametros)
    {
        using (SqlConnection cx = ConexaoBD.GetConexao())
        {
            using (SqlDataAdapter adapter = new SqlDataAdapter(sql, cx))
            {
                if (parametros != null)
                    adapter.SelectCommand.Parameters.AddRange(parametros);
                DataTable tabela = new DataTable();
                adapter.Fill(tabela);

                return tabela;
            }
```

```
    }
}



/// <summary>
/// método que testa se um determinado valor string contém um inteiro válido
/// </summary>
/// <param name="valor">valor string a ser testado</param>
/// <returns>true se for inteiro ou false caso contrário </returns>
public static bool ValidaInt(string valor)
{
    try
    {
        Convert.ToInt32(valor);
        return true;
    }
    catch
    {
        return false;
    }
}

/// <summary>
/// método que testa se um determinado valor string contém um double válido
/// </summary>
/// <param name="valor">valor string a ser testado</param>
/// <returns>true se for double ou false caso contrário </returns>
public static bool ValidaDouble(string valor)
{
    try
    {
        Convert.ToDouble(valor);
        return true;
    }
    catch
    {
        return false;
    }
}



/// <summary>
/// Exibe uma mensagem
/// </summary>
/// <param name="mensagem">Texto da mensagem</param>
/// <param name="tipoDaMensagem">tipo da mensagem</param>
/// <returns>Quando for mensagem de confirmação, returna true se o
/// usuário clicar em sim e retorna False caso clique em não</returns>
public static bool Mensagem(string mensagem, TipoMensagemEnum tipoDaMensagem)
{
    if (tipoDaMensagem == TipoMensagemEnum.alerta)
    {
        MessageBox.Show(mensagem, "Atenção", MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);
        return true;
    }
    else if (tipoDaMensagem == TipoMensagemEnum.erro)
    {
        MessageBox.Show(mensagem, "Erro", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        return true;
    }
    else if (tipoDaMensagem == TipoMensagemEnum.informacao)
```

```
            {
                MessageBox.Show(mensagem, "Informação", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
                return true;
            }
            else
            {
                if (MessageBox.Show(mensagem, "Atenção", MessageBoxButtons.YesNo,
                MessageBoxIcon.Question) == DialogResult.Yes)
                    return true;
                else
                    return false;
            }
        }

    }
```

Classe de conexão com Banco de dados

```
public static class ConexaoBD
  {
      /// <summary>
      /// Método Estático que retorna um conexao aberta com o BD
      /// </summary>
      /// <returns>Conexão aberta</returns>
      public static SqlConnection GetConexao()
      {
          string strCon = "Data Source=LOCALHOST;Initial Catalog=AULADB;user id=sa; password=123456";
          SqlConnection conexao = new SqlConnection(strCon);
          conexao.Open();
          return conexao;
      }
  }
```

Classe Ancestral para os todos os VOs.

Dado que toda tabela terá um campo inteiro como chave primária, iremos defini-lo aqui. Isso irá garantir que alguns algoritmos possam ser generalizados nas classes descendentes.

```
      public abstract class PadraoVO
      {
             public virtual int Id { get; set; }

      }
```

Classe VO para a tabela tecnicos

Para fazer o cadastro de técnicos, vamos criar o VO abaixo.

```
public class TecnicoVO : PadraoVO
{
    public string Nome { get; set; }
}
```

Classe DAO Padrão

Agora, vamos criar um padrão para as classes DAO. Observe que ela não é estática!!!!

```
public abstract class PadraoDAO
{
    protected string Tabela { get; set; }
    protected string Chave { get; set; } = "id";  // valor default

    protected abstract string MontaSQLInsert();
    protected abstract string MontaSQLUpdate();
    protected abstract SqlParameter[] CriaParametros(PadraoVO o);
    protected abstract PadraoVO MontaVO(DataRow dr);

    protected virtual string MontaSQLDelete()
    {
        return $"delete {Tabela} where {Chave} =  @id";
    }

    protected virtual string MontaSQLConsulta()
    {
        return $"select * from {Tabela} where {Chave} =  @id";
    }


    /// <summary>
    /// Método  para inserir um registro no BD
    /// </summary>
    public virtual void Inserir(PadraoVO o)
    {
        if (Consulta(o.Id) != null)
            throw new Exception("Este código já está sendo utilizado!");

        string sql = MontaSQLInsert();

        Metodos.ExecutaSQL(sql, CriaParametros(o));
    }


    /// <summary>
    /// Método  para alterar um regisstro
    /// </summary>
    public virtual void Alterar(PadraoVO o)
    {
        string sql = MontaSQLUpdate();
        Metodos.ExecutaSQL(sql, CriaParametros(o));
    }
```

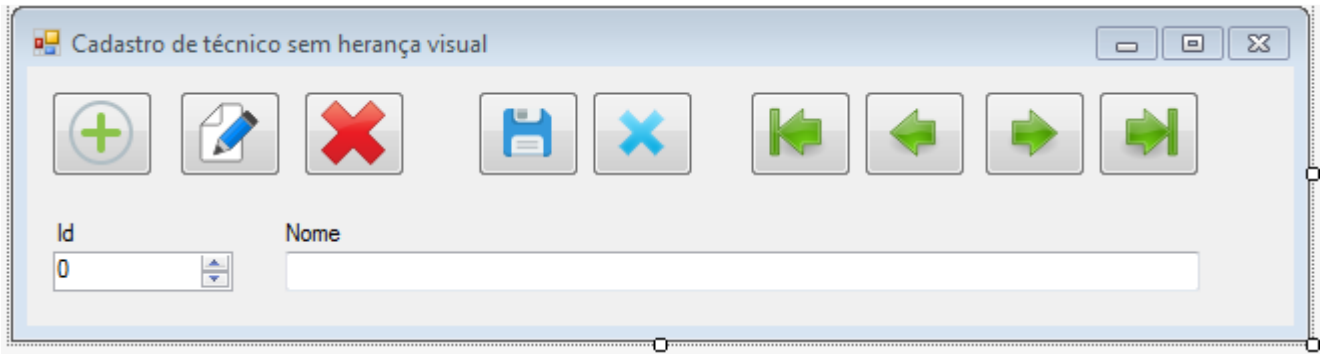
```
/// <summary>
/// Método para excluir
/// </summary>
public virtual void Excluir(int Id)
{
    string sql = MontaSQLDelete();
    SqlParameter[] parametros = new SqlParameter[1];
    parametros[0] = new SqlParameter( Chave , Id);
    Metodos.ExecutaSQL(sql, parametros);
}


/// <summary>
/// Método para consultar 1 registro
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
public PadraoVO Consulta(int id)
{
    using (SqlConnection cx = ConexaoBD.GetConexao())
    {
        string sql = MontaSQLConsulta();
        SqlParameter[] parametros =
        {
            new SqlParameter(Chave, id)
        };

        DataTable tabela = Metodos.ExecutaSelect(sql, parametros);
        if (tabela.Rows.Count == 0)
            return null;
        else
        {
            return MontaVO(tabela.Rows[0]);
        }
    }
}


/// <summary>
/// Obtem o Proximo id disponível
/// </summary>
/// <returns></returns>
public virtual int ProximoId()
{
    string sql = $"select isnull(max({Chave})+1,1) from {Tabela}";
    using (SqlConnection cx = ConexaoBD.GetConexao())
    {
        SqlCommand cmd = new SqlCommand(sql, cx);
        return Convert.ToInt32(cmd.ExecuteScalar());
    }
}


/// <summary>
/// Primeiro registro
/// </summary>
/// <returns></returns>
public virtual PadraoVO Primeiro()
{
    string sql = $"select top 1 * from {Tabela} order by {Chave}";
    return ExecutaSqlLocal(sql, null);
}
```

```
        /// <summary>
        /// Ultimo registro
        /// </summary>
        /// <returns></returns>
        public virtual PadraoVO Ultimo()
        {
            string sql = $"select top 1 * from {Tabela} order by {Chave} desc";
            DataTable tabela = Metodos.ExecutaSelect(sql, null);
            return ExecutaSqlLocal(sql, null);
        }

        /// <summary>
        /// Próximo registro
        /// </summary>
        /// <param name="atual"></param>
        /// <returns></returns>
        public virtual PadraoVO Proximo(int atual)
        {
            string sql = $"select  top 1 * from {Tabela} where {Chave} > @Atual  order by {Chave} ";
            SqlParameter[] p =
            {
                new SqlParameter("Atual", atual)
            };
            return ExecutaSqlLocal(sql, p);
        }


        /// <summary>
        /// Registro anterior
        /// </summary>
        /// <param name="atual"></param>
        /// <returns></returns>
        public virtual PadraoVO Anterior(int atual)
        {
            string sql = $"select top 1 * from {Tabela} where {Chave} < @Atual order by {Chave} desc";
            SqlParameter[] p =
            {
                new SqlParameter("Atual", atual)
            };
            return ExecutaSqlLocal(sql, p);
        }


        /// <summary>
        /// Executa uma instrução SQL
        /// </summary>
        /// <param name="sql">instrução</param>
        /// <param name="parametros">parametros</param>
        /// <returns>Objeto PadraoVO</returns>
        protected PadraoVO ExecutaSqlLocal(string sql, SqlParameter[] parametros)
        {
            DataTable tabela = Metodos.ExecutaSelect(sql, parametros);

            if (tabela.Rows.Count == 0)
                return null;
            else
                return MontaVO(tabela.Rows[0]);
        }

    }
```

Classe DAO para a tabela Tecnicos

```
    public class TecnicoDAO : PadraoDAO
    {
        public TecnicoDAO()
        {
            Tabela = "Tecnicos";
            Chave = "id"; // não precisaria informar já que o default é "id"
        }

        protected override SqlParameter[] CriaParametros(PadraoVO o)
        {
            TecnicoVO t = o as TecnicoVO;
            SqlParameter[] parametros =
            {
                new SqlParameter("id", t.Id),
                new SqlParameter("nome", t.Nome)
            };

            return parametros;
        }

        protected override string MontaSQLInsert()
        {
            return  "insert into tecnicos (id,nome) values(@id, @nome)";
        }

        protected override string MontaSQLUpdate()
        {
            return "update tecnicos set nome=@nome where id = @id";
        }

        protected override PadraoVO MontaVO(DataRow dr)
        {
            TecnicoVO t = new TecnicoVO();
            t.Id = Convert.ToInt32(dr["id"]);
            t.Nome = dr["nome"].ToString();
            return t;
        }
    }
}
```

Tela de cadastro com ajustes para suporte à classe DAO não estática



Código fonte do formulário acima:

```
    public partial class frCadTecnicoSemHeranca : Form
    {
        TecnicoDAO tecnicoDAO = new TecnicoDAO();

        public frCadTecnicoSemHeranca()
        {
            InitializeComponent();
        }


        private void Form1_Load(object sender, EventArgs e)
        {
            btnPrimeiro.PerformClick();
            AlteraParaModo(EnumModoOperacao.Navegacao);
        }


        /// <summary>
        /// Altera a tela para se adequar a um determindo modo de operação
        /// que pode ser inclusão, alteração ou consulta
        /// </summary>
        /// <param name="modo">modo para o qual se deseja alterar</param>
        private void AlteraParaModo(EnumModoOperacao modo)
        {
            txtNome.Enabled = (modo != EnumModoOperacao.Navegacao);
            txtId.Enabled = (modo != EnumModoOperacao.Navegacao);

            btnInclui.Enabled = (modo == EnumModoOperacao.Navegacao);
            btnAltera.Enabled = (modo == EnumModoOperacao.Navegacao);
            btnExclui.Enabled = (modo == EnumModoOperacao.Navegacao) &&
                                txtId.Text.Length > 0;

            btnPrimeiro.Enabled = (modo == EnumModoOperacao.Navegacao);
            btnAnterior.Enabled = (modo == EnumModoOperacao.Navegacao) && txtId.Value > 0;
            btnProximo.Enabled = (modo == EnumModoOperacao.Navegacao) && txtId.Value > 0;
            btnUltimo.Enabled = (modo == EnumModoOperacao.Navegacao);

            btnGravar.Enabled = (modo != EnumModoOperacao.Navegacao);
            btnCancelar.Enabled = (modo != EnumModoOperacao.Navegacao);

            if (modo == EnumModoOperacao.Inclusao)
            {
                txtId.Enabled = true;
                LimpaCampos(this);
```

```
            txtId.Focus();
        }
        else
            txtId.Enabled = false;
    }

    private void LimpaCampos(Control objeto)
    {
        if (objeto is TextBox ||
            objeto is MaskedTextBox)
            objeto.Text = "";
        else if (objeto is NumericUpDown)
            (objeto as NumericUpDown).Value = 0;
        else
        {
            foreach (Control o in objeto.Controls)
                LimpaCampos(o);
        }
    }

    private void btnInclui_Click(object sender, EventArgs e)
    {
        AlteraParaModo(EnumModoOperacao.Inclusao);
        txtId.Value = tecnicoDAO.ProximoId();
    }

    private void btnAltera_Click(object sender, EventArgs e)
    {
        AlteraParaModo(EnumModoOperacao.Alteracao);
    }

    private void btnExclui_Click(object sender, EventArgs e)
    {
        if (Metodos.ValidaInt(txtId.Text) == false)
        {
            Metodos.Mensagem("Digite apenas números no campo ID.",
                TipoMensagemEnum.alerta);
            return;
        }

        if (!Metodos.Mensagem("Confirma?", TipoMensagemEnum.pergunta))
            return;

        try
        {
            tecnicoDAO.Excluir(Convert.ToInt32(txtId.Text));
            btnPrimeiro.PerformClick();
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }


    private void TrataErro(Exception erro)
    {
        if (erro is FormatException)
        {
            Metodos.Mensagem("Campo numérico inválido!", TipoMensagemEnum.erro);
        }
        else if (erro is SqlException)
        {
            Metodos.Mensagem("Ocorreu um erro no banco de dados.", TipoMensagemEnum.erro);
        }
        else if (erro is Exception)
```

```
    {
        Metodos.Mensagem(erro.Message, TipoMensagemEnum.erro);
    }
}

private void btnGravar_Click(object sender, EventArgs e)
{
    try
    {
        TecnicoVO t = new TecnicoVO();
        t.Id = Convert.ToInt32(txtId.Text);
        t.Nome = txtNome.Text;

        if (txtId.Enabled)
            tecnicoDAO.Inserir(t);
        else
            tecnicoDAO.Alterar(t);

        AlteraParaModo(EnumModoOperacao.Navegacao);
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}


private void PreencheTela(PadraoVO t)
{
    try
    {
        if (t != null)
        {
            txtId.Text = (t as TecnicoVO).Id.ToString();
            txtNome.Text = (t as TecnicoVO).Nome;
        }
        else
        {
            LimpaCampos(this);
        }
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}


private void btnCancelar_Click(object sender, EventArgs e)
{
    if (txtId.Enabled)
        PreencheTela(tecnicoDAO.Primeiro());
    else
        PreencheTela(tecnicoDAO.Consulta(Convert.ToInt32(txtId.Text)));

    AlteraParaModo(EnumModoOperacao.Navegacao);
}

private void btnPrimeiro_Click(object sender, EventArgs e)
{
    PreencheTela(tecnicoDAO.Primeiro());
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    PadraoVO obj = tecnicoDAO.Anterior(Convert.ToInt32(txtId.Text));
```

```
            if (obj != null)
                PreencheTela(obj);
        }

        private void btnProximo_Click(object sender, EventArgs e)
        {
            PadraoVO obj = tecnicoDAO.Proximo(Convert.ToInt32(txtId.Text));
            if (obj != null)
                PreencheTela(obj);
        }

        private void btnUltimo_Click(object sender, EventArgs e)
        {
            PreencheTela(tecnicoDAO.Ultimo());
        }
    }
```

Exercícios

1. Utilizando os conceitos vistos neste capítulo, faça um cadastro que contemple a seguinte tabela:

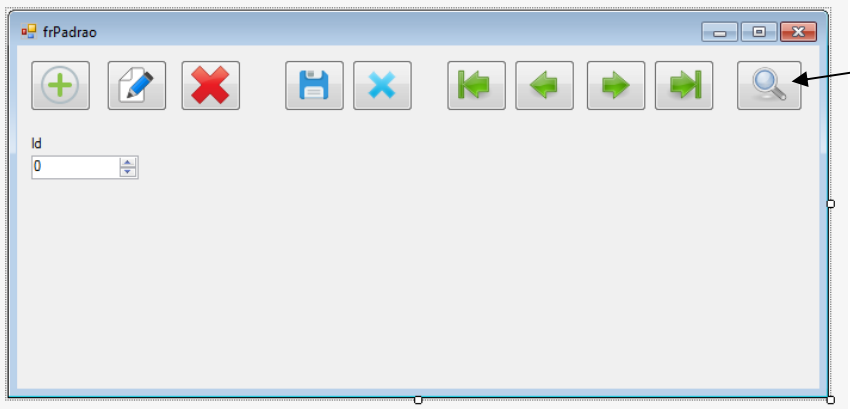
```
CREATE TABLE Produtos(
       [Produtoid] [int] NOT NULL primary key,
       [ProdutoDescricao] [varchar](50) NULL,
       [Valor_unitario] [decimal](18, 2) NULL,
       [imagem] [varchar](max) NULL
)
```

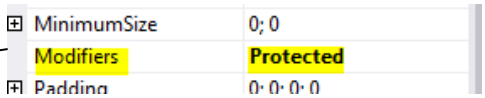
16. Herança Visual

Permite utilizar conceitos de POO para redução de código nos CRUDs.

Formulário padrão para CRUDs

A generalização na tela padrão de cadastro é grande, mas de mostrará útil quanto mais houverem descendentes. Crie um novo formulário chamado frPadrão, semelhando ao da figura abaixo.





Caso queira permitir que os eventos ou propriedades de um objeto sejam modificados nos descendentes, altere a propriedade Modifiers para Protected, como indicado na imagem.

Código fonte do formulário acima:

```
public partial class frPadrao : Form
{
    protected PadraoDAO cadastroDAO;  // deve ser preenchido nos descendentes !!!!!

    public bool SugereProximoId { get; set; } = false;

    public frPadrao()
    {
        InitializeComponent();
    }

    protected virtual void PreencheTela(PadraoVO o)
    {
        //preencher nos filhos
    }


    protected void TrataErro(Exception erro)
    {
        if (erro is FormatException)
        {
            Metodos.Mensagem("Campo numérico inválido!", TipoMensagemEnum.erro);
        }
        else if (erro is SqlException)
        {
            Metodos.Mensagem("Ocorreu um erro no banco de dados.", TipoMensagemEnum.erro);
        }
        else if (erro is Exception)
        {
            Metodos.Mensagem(erro.Message, TipoMensagemEnum.erro);
        }
    }
```

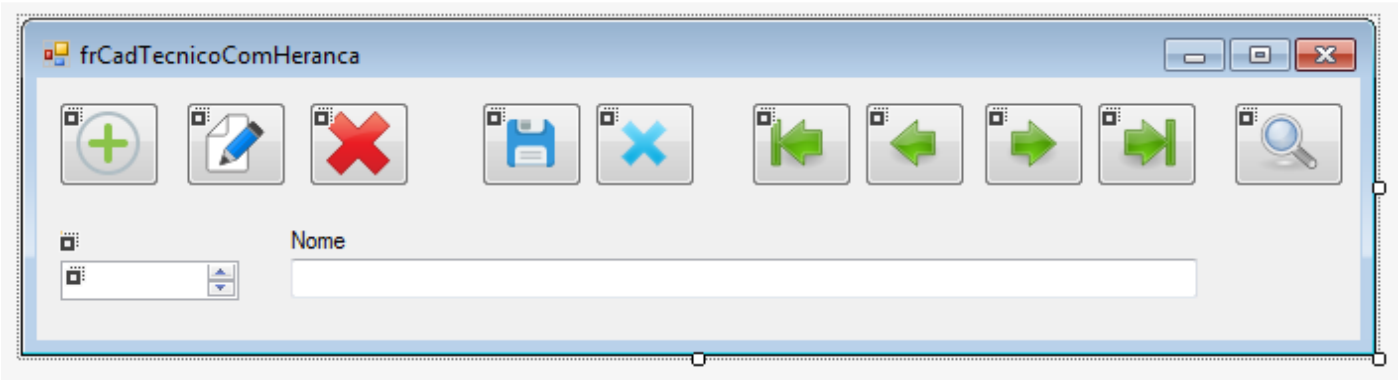
```
protected bool EstaEmModoDev()
{
    return (LicenseManager.UsageMode == LicenseUsageMode.Designtime);
}

/// <summary>
/// Altera a tela para se adequar a um determindo modo de operação
/// que pode ser inclusão, alteração ou consulta
/// </summary>
/// <param name="modo">modo para o qual se deseja alterar</param>
private void AlteraParaModo(EnumModoOperacao modo)
{
    btnInclui.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnAltera.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnExclui.Enabled = (modo == EnumModoOperacao.Navegacao) && txtId.Text.Length > 0;

    btnPrimeiro.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnAnterior.Enabled = (modo == EnumModoOperacao.Navegacao) && txtId.Value > 0;
    btnProximo.Enabled = (modo == EnumModoOperacao.Navegacao) && txtId.Value > 0;
    btnUltimo.Enabled = (modo == EnumModoOperacao.Navegacao);

    btnGravar.Enabled = (modo != EnumModoOperacao.Navegacao);
    btnCancelar.Enabled = (modo != EnumModoOperacao.Navegacao);
    btnPesquisa.Enabled = (modo == EnumModoOperacao.Navegacao);

    if (modo == EnumModoOperacao.Inclusao)
    {
        txtId.Enabled = true;
        LimpaCampos(this);
        txtId.Focus();
    }
    else
        txtId.Enabled = false;

    ControlaCamposTela(modo); // propositalmente no final para que se possa alterar alguns
                              //dos comportamentos acima nos descendentes
}


protected virtual void ControlaCamposTela(EnumModoOperacao modo)
{
    // deve ser sobrescrito
}

protected virtual PadraoVO PreencheObj()
{
    return null; // é necessário ter algo .. não é void!  Programar nos descendentes
}


protected void LimpaCampos(Control objeto)
{
    if (objeto is TextBox ||
        objeto is MaskedTextBox)
        objeto.Text = "";
    else if (objeto is NumericUpDown)
        (objeto as NumericUpDown).Value = 0;
    else
    {
        foreach (Control o in objeto.Controls)
            LimpaCampos(o);
    }
}

private void btnAltera_Click(object sender, EventArgs e)
{
```

```
    AlteraParaModo(EnumModoOperacao.Alteracao);
}

private void btnInclui_Click(object sender, EventArgs e)
{
    AlteraParaModo(EnumModoOperacao.Inclusao);
    if (SugereProximoId)
        txtId.Value = cadastroDAO.ProximoId();
}

private void btnExclui_Click(object sender, EventArgs e)
{
    if (Metodos.ValidaInt(txtId.Text) == false)
    {
        Metodos.Mensagem("Digite apenas números no campo ID.",
            TipoMensagemEnum.alerta);
        return;
    }

    if (!Metodos.Mensagem("Confirma?", TipoMensagemEnum.pergunta))
        return;

    try
    {
        cadastroDAO.Excluir(Convert.ToInt32(txtId.Text));
        btnPrimeiro.PerformClick();
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}

private void btnGravar_Click(object sender, EventArgs e)
{
    try
    {
        PadraoVO obj = PreencheObj();

        if (txtId.Enabled)
            cadastroDAO.Inserir(obj);
        else
            cadastroDAO.Alterar(obj);

        AlteraParaModo(EnumModoOperacao.Navegacao);
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}

private void btnCancelar_Click(object sender, EventArgs e)
{
    if (txtId.Enabled)
        PreencheTela(cadastroDAO.Primeiro());
    else
        PreencheTela(cadastroDAO.Consulta(Convert.ToInt32(txtId.Text)));

    AlteraParaModo(EnumModoOperacao.Navegacao);
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    try
    {
```

```
            PadraoVO o = cadastroDAO.Anterior(Convert.ToInt32(txtId.Text));
            if (o != null)
                PreencheTela(o);
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }



    private void btnPrimeiro_Click(object sender, EventArgs e)
    {
        try
        {
            PadraoVO o = cadastroDAO.Primeiro();
            PreencheTela(o);
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }

    private void btnProximo_Click(object sender, EventArgs e)
    {
        try
        {
            PadraoVO o  = cadastroDAO.Proximo(Convert.ToInt32(txtId.Text));
            if (o != null)
                PreencheTela(o);
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }

    private void btnUltimo_Click(object sender, EventArgs e)
    {
        try
        {
            PadraoVO o = cadastroDAO.Ultimo();
            PreencheTela(o);
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }

    private void btnPesquisa_Click(object sender, EventArgs e)
    {

    }

    private void frPadrao_Load(object sender, EventArgs e)
    {
        if (!EstaEmModoDev()) // modo de desenvolvimento
        {
            btnPrimeiro.PerformClick();
            AlteraParaModo(EnumModoOperacao.Navegacao);
        }
    }
}
```

Herança visual aplicada ao cadadastro de Tecnico

Agora que temos o formulário padrão, vamos refazer o formulário de cadastro de técnico, porém usando como base o formulário padrão. Crie um novo formulário, e no código fonte altere a herança para frPadrao, como indica abaixo.

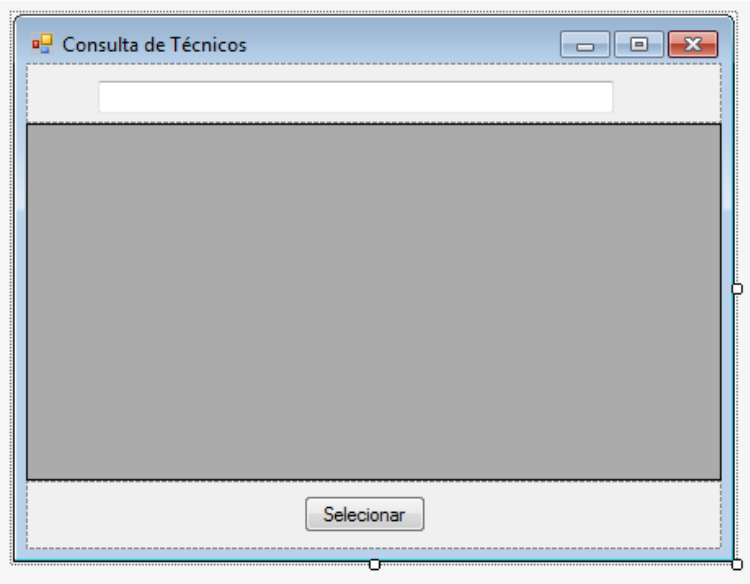



```
public partial class frCadTecnicoComHeranca : frPadrao
{
    public frCadTecnicoComHeranca()
    {
        InitializeComponent();
        cadastroDAO = new TecnicoDAO();
        SugereProximoId = true;
    }


    protected override void PreencheTela(PadraoVO o)
    {
        try
        {
            if (o != null)
            {
                txtId.Text = (o as TecnicoVO).Id.ToString();
                txtNome.Text = (o as TecnicoVO).Nome;
            }
            else
            {
                LimpaCampos(this);
            }
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }

    protected override void ControlaCamposTela(EnumModoOperacao modo)
    {
        txtNome.Enabled = modo != EnumModoOperacao.Navegacao;
    }

    protected override PadraoVO PreencheObj()
```

```
        {
            TecnicoVO t = new TecnicoVO();
            t.Id = Convert.ToInt32(txtId.Text);
            t.Nome = txtNome.Text;
            return t;
        }
    }
```

Adicionando uma tela de consulta de técnicos

Crie um novo formulário, chamado frConsultaTecnico, semelhante ao da figura abaixo.



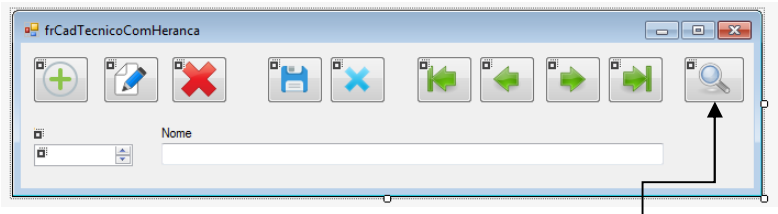
Código fonte do formulário:

```
    public partial class frConsultaTecnico : Form
    {

        public int IdSelecionado { get; private set; }
        public frConsultaTecnico()
        {
            InitializeComponent();
        }


        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            List<TecnicoVO> lista = (new  TecnicoDAO()).ListarPorNome(txtNome.Text.Trim());
            dataGridView1.DataSource = lista;
            dataGridView1.ReadOnly = true;
        }

        private void btnSelecionar_Click(object sender, EventArgs e)
        {
            if (dataGridView1.CurrentRow != null)
            {
                TecnicoVO obj = dataGridView1.CurrentRow.DataBoundItem as TecnicoVO;
                IdSelecionado = obj.Id;
                Close();
            }
        }
```

```
        private void dataGridView1_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
        {
            btnSelecionar.PerformClick();
        }
    }
```

Na classe **TecnicoDAO** adicione o método de busca pelo nome:

```
        public List<TecnicoVO> ListarPorNome(string nome)
        {
            List<TecnicoVO> retorno = new List<TecnicoVO>();
            string sql = "select * from tecnicos where nome like @nome order by nome";
            SqlParameter[] parametros = new SqlParameter[]
                {
                    new SqlParameter("nome", "%"+nome +"%")
                };

            DataTable tabela = Metodos.ExecutaSelect(sql, parametros);
            foreach(DataRow linha in tabela.Rows)
                retorno.Add(MontaVO(linha) as TecnicoVO);
            return retorno;
        }
```




```
        private void btnPesquisa_Click(object sender, EventArgs e)
        {
            using (frConsultaTecnico tela = new frConsultaTecnico())
            {
                tela.ShowDialog();
                if (tela.IdSelecionado != 0)
                {
                    TecnicoVO t = cadastroDAO.Consulta(tela.IdSelecionado) as TecnicoVO;
                    PreencheTela(t);
                }
            }
        }
```

Exercícios

1. Aplique no cadastro de produtos os conceitos vistos neste capítulo.
2. No mesmo projeto, crie outro cadastro, que contemple a tabela a seguir:

```
CREATE TABLE Clientes(
	[ClienteId] [int] NOT NULL primary key,
	[ClienteNome] [varchar](50) NULL
)
```

3. Em um novo projeto, faça ajustes nas classes DAO (incluindo a classe base) para que o CRUD seja feito inteiramente utilizando-se stored procedures, assim como já fizemos anteriormente. Refaça os exercícios 1 e 2 utilizando-se deste novo padrão DAO. Tente reutilizar as stored procedures na medida do possível.

17. Herança Visual com UserControls

UserControls são conteiners (como o Panel por exemplo) que permitem a componetização da sua aplicação, permitindo herança de código e de elementos visuais.

O projeto a seguir foi criado com base no cadastro de técnico do capítulo anterior, com a adição do cadastro de jogadores. O objetivo será fazer um único formulário com os 2 cadastros, porém de forma que o código não fique muito poluído dado a quantidade de métodos que seriam necessários para realizar tal tarefa.

Crie um novo projeto, copiando todas as classes DAO, VO, enumeradores, etc. do projeto anterior, menos os formulários.

Scripts de BD:

```
CREATE TABLE[dbo].[Tecnicos](
[Id]  [int]  NOT NULL primary key,
[Nome] [varchar](50) NULL,
[Salario] [decimal](18, 2) NULL);

GO

CREATE TABLE[dbo].[Times](
       [id] [int] NOT NULL primary key,
       [nome] [varchar](50) NULL)
GO

CREATE TABLE[dbo].[JogadorFutebol](
      [id]    [int]    NOT NULL primary key,
      [NumeroCamisa] [int]    NULL,
      [Nome]    [varchar](50) NULL,
      [TimeId]    [int]    NULL)
GO

INSERT [dbo].[Times] ([id], [nome]) VALUES (1, N'Santos')
INSERT [dbo].[Times] ([id], [nome]) VALUES (2, N'Corinthians')
INSERT [dbo].[Times] ([id], [nome]) VALUES (3, N'Palmeiras')
INSERT [dbo].[Times] ([id], [nome]) VALUES (4, N'São paulo')
INSERT [dbo].[Times] ([id], [nome]) VALUES (5, N'Santo André')
```

Classes DAO e VO

A seguir, as classes DAO e VO que você irá precisar e que não haviam no exemplo anterior:

```
public class TimeVO : PadraoVO
{
    public string Nome { get; set; }
}
```

```
public class JogadorFutebolVO : PadraoVO
{
    private int id;
    private string nome;
    private int numeroCamisa;
    private int timeId;

    public override int Id
    {
        get
        {
            return id;
        }

        set
        {
            if (value <= 0)
                throw new Exception("Código  tem que ser maior que zero.");
            id = value;
        }
    }

    public string Nome
    {
        get
        {
            return nome;
        }

        set
        {
            if (string.IsNullOrEmpty(value))
                throw new Exception("Nome obrigatório.");
            nome = value;
        }
    }

    public int NumeroCamisa
    {
        get
        {
            return numeroCamisa;
        }

        set
        {
            if (value <= 0)
                throw new Exception("Número da camisa tem que ser maior que zero.");
            numeroCamisa = value;
        }
    }
```

```
        public int TimeId
        {
            get
            {
                return timeId;
            }

            set
            {
                if (value <= 0)
                    throw new Exception("Código do time tem que ser maior que zero.");
                timeId = value;
            }
        }
    }
```

```
   public class TecnicoVO : PadraoVO
    {
        public string Nome { get; set; }
        public double Salario { get; set; }
    }
```

Classes DAO

```
    public class TimeDAO : PadraoDAO
    {
        public TimeDAO()
        {
            Tabela = "Times";
        }


        public  List<TimeVO> Lista(string nomeTimeParcial)
        {
            string sql = "select * from times where nome like @nome order by nome";
            SqlParameter[] parametros = new SqlParameter[]
                {
                    new SqlParameter("nome", "%"+nomeTimeParcial +"%")
                };

            DataTable tabela = Metodos.ExecutaSelect(sql, parametros);
            List<TimeVO> lista = new List<TimeVO>();
            foreach (DataRow dr in tabela.Rows)
            {
                lista.Add(MontaVO(dr) as TimeVO);
            }
            return lista;
        }


        protected override PadraoVO MontaVO(DataRow dr)
        {
            TimeVO t = new TimeVO();
            t.Id = Convert.ToInt16(dr["id"]);
            t.Nome = dr["nome"].ToString();
            return t;
        }

        protected override string MontaSQLInsert()
        {
            throw new NotImplementedException();
```

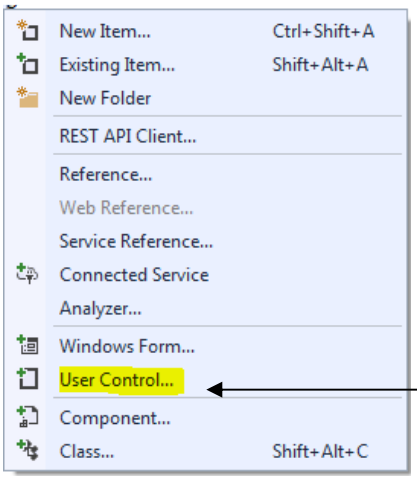
```
        }

        protected override string MontaSQLUpdate()
        {
            throw new NotImplementedException();
        }

        protected override SqlParameter[] CriaParametros(PadraoVO o)
        {
            throw new NotImplementedException();
        }
    }
```

```
public class JogadorFutebolDAO : PadraoDAO
    {
        public JogadorFutebolDAO()
        {
            Tabela = "JogadorFutebol";
        }

        protected override string MontaSQLInsert()
        {
            return
            "insert into JogadorFutebol(id, nome, NumeroCamisa, TimeId)" +
            "values ( @id, @nome, @NumeroCamisa, @TimeId)";
        }

        protected override string MontaSQLUpdate()
        {
            return
           "update jogadorFutebol set nome=@nome, NumeroCamisa=@NumeroCamisa, timeId = @timeId" +
           "Where id = @id";
        }

        protected override SqlParameter[] CriaParametros(PadraoVO o)
        {
            JogadorFutebolVO j = o as JogadorFutebolVO;

            SqlParameter[] parametros = new SqlParameter[4];
            parametros[0] = new SqlParameter("id", j.Id);
            parametros[1] = new SqlParameter("nome", j.Nome);
            parametros[2] = new SqlParameter("NumeroCamisa", j.NumeroCamisa);
            parametros[3] = new SqlParameter("timeId", j.TimeId);

            return parametros;
        }

        protected override PadraoVO MontaVO(DataRow dr)
        {
            JogadorFutebolVO a = new JogadorFutebolVO();
            a.Id = Convert.ToInt32(dr["id"]);
            a.Nome = dr["nome"].ToString();
            a.NumeroCamisa = Convert.ToInt32(dr["NumeroCamisa"]);
            a.TimeId = Convert.ToInt32(dr["timeId"]);
            return a;
        }


        public List<JogadorFutebolVO> Lista(string nomeParcial)
        {
            string sql = "select * from jogadorFutebol where nome like @nome order by nome";
            SqlParameter[] parametros = new SqlParameter[]
                {
                    new SqlParameter("nome", "%"+nomeParcial +"%")
                };
```

```
            DataTable tabela = Metodos.ExecutaSelect(sql, parametros);
            List<JogadorFutebolVO> lista = new List<JogadorFutebolVO>();
            foreach (DataRow dr in tabela.Rows)
            {
                lista.Add(MontaVO(dr) as JogadorFutebolVO);
            }
            return lista;
        }
    }
}
```

```
    public class TecnicoDAO : PadraoDAO
    {
        public TecnicoDAO()
        {
            Tabela = "Tecnicos";
        }

        protected override SqlParameter[] CriaParametros(PadraoVO o)
        {
            TecnicoVO t = o as TecnicoVO;
            SqlParameter[] parametros =
            {
                new SqlParameter("id", t.Id),
                new SqlParameter("nome", t.Nome),
                new SqlParameter("salario", t.Salario)
            };

            return parametros;
        }

        protected override string MontaSQLInsert()
        {
            return  "insert into tecnicos (id,nome,salario)" +
                    "values(@id, @nome, @salario)";
        }

        protected override string MontaSQLUpdate()
        {
            return "update tecnicos set nome=@nome, salario=@salario " +
                   "where id = @id";
        }

        protected override PadraoVO MontaVO(DataRow dr)
        {
            TecnicoVO t = new TecnicoVO();
            t.Id = Convert.ToInt32(dr["id"]);
            t.Nome = dr["nome"].ToString();
            t.Salario= Convert.ToDouble(dr["salario"]);
            return t;
        }
    }
```

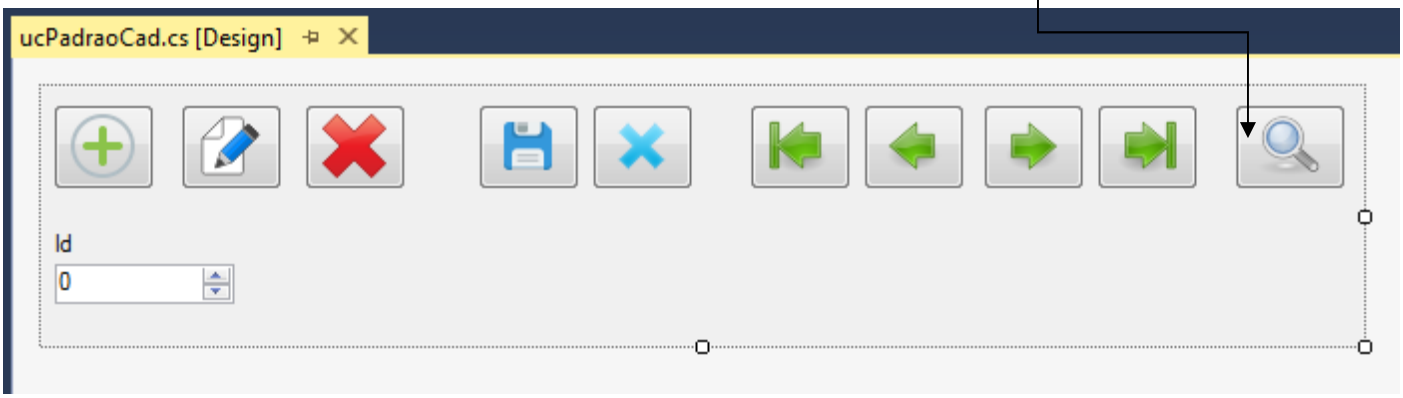
Criando um userControl padrão



Nome do user control: ucPadraoCad

Não esqueça de alterar a propriedade Modifiers para protected!

Coloque os seguintes objetos no UserControl:



Código fonte desse usercontrol. Observe que este código é praticamente o mesmo do formulário padrão.

```
public partial class ucPadraoCad : UserControl
  {
      protected PadraoDAO cadastroDAO;

      public bool SugereProximoId { get; set; } = false;


      public ucPadraoCad()
      {
          InitializeComponent();
      }


      protected bool ModoDeDesign()
      {
          bool designMode = (LicenseManager.UsageMode == LicenseUsageMode.Designtime);
          return designMode;
      }
```

```
protected virtual void PreencheTela(PadraoVO o)
{
    //preencher nos filhos
}

protected void TrataErro(Exception erro)
{
    if (erro is FormatException)
    {
        Metodos.Mensagem("Campo numérico inválido!", TipoMensagemEnum.erro);
    }
    else if (erro is SqlException)
    {
        Metodos.Mensagem("Ocorreu um erro no banco de dados.", TipoMensagemEnum.erro);
    }
    else if (erro is Exception)
    {
        Metodos.Mensagem(erro.Message, TipoMensagemEnum.erro);
    }
}



/// <summary>
/// Altera a tela para se adequar a um determindo modo de operação
/// que pode ser inclusão, alteração ou consulta
/// </summary>
/// <param name="modo">modo para o qual se deseja alterar</param>
private void AlteraParaModo(EnumModoOperacao modo)
{
    btnInclui.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnAltera.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnExclui.Enabled = (modo == EnumModoOperacao.Navegacao) &&
                         txtId.Text.Length > 0;

    btnPrimeiro.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnAnterior.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnProximo.Enabled = (modo == EnumModoOperacao.Navegacao);
    btnUltimo.Enabled = (modo == EnumModoOperacao.Navegacao);

    btnGravar.Enabled = (modo != EnumModoOperacao.Navegacao);
    btnCancelar.Enabled = (modo != EnumModoOperacao.Navegacao);
    btnPesquisa.Enabled = (modo == EnumModoOperacao.Navegacao);

    if (modo == EnumModoOperacao.Inclusao)
    {
        txtId.Enabled = true;
        LimpaCampos(this);
        txtId.Focus();
    }
    else
        txtId.Enabled = false;


    ControlaCamposTela(modo);
}


protected virtual void ControlaCamposTela(EnumModoOperacao modo)
{
    // deve ser sobrescrito
}

protected void LimpaCampos(Control objeto)
{
```

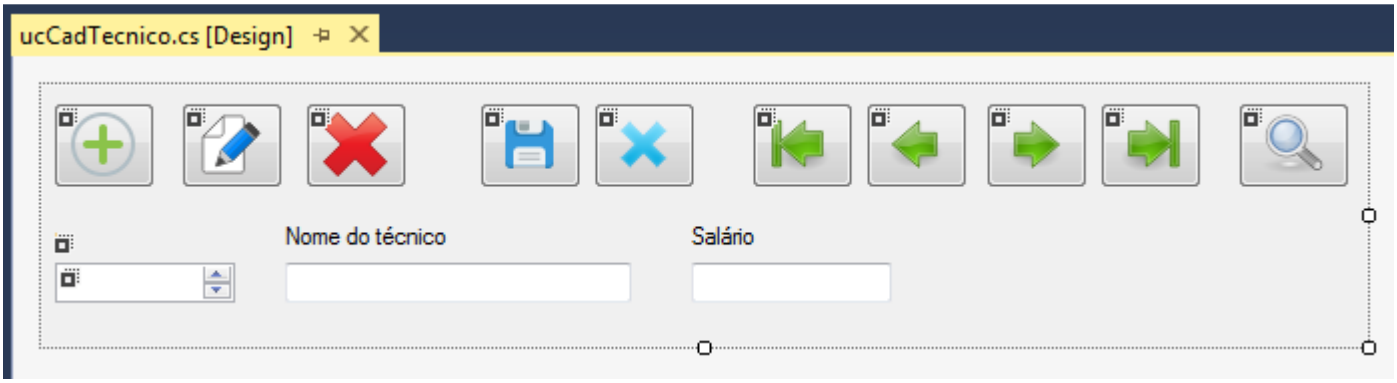
```
    if (objeto is TextBox ||
        objeto is MaskedTextBox)
        objeto.Text = "";
    else if (objeto is NumericUpDown)
        (objeto as NumericUpDown).Value = 0;
    else
    {
        foreach (Control o in objeto.Controls)
            LimpaCampos(o);
    }
}

private void btnAltera_Click(object sender, EventArgs e)
{
    AlteraParaModo(EnumModoOperacao.Alteracao);
}

private void btnInclui_Click(object sender, EventArgs e)
{
    AlteraParaModo(EnumModoOperacao.Inclusao);
    if (SugereProximoId)
        txtId.Value = cadastroDAO.ProximoId();
}

private void btnExclui_Click(object sender, EventArgs e)
{
    if (Metodos.ValidaInt(txtId.Text) == false)
    {
        Metodos.Mensagem("Digite apenas números no campo ID.",
            TipoMensagemEnum.alerta);
        return;
    }

    if (!Metodos.Mensagem("Confirma?", TipoMensagemEnum.pergunta))
        return;

    try
    {
        cadastroDAO.Excluir(Convert.ToInt32(txtId.Text));
        btnPrimeiro.PerformClick();
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}

private void btnGravar_Click(object sender, EventArgs e)
{
    try
    {
        PadraoVO obj = PreencheObj();

        if (txtId.Enabled)
            cadastroDAO.Inserir(obj);
        else
            cadastroDAO.Alterar(obj);

        AlteraParaModo(EnumModoOperacao.Navegacao);
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}
```

```
protected virtual PadraoVO PreencheObj()
{
    return null;
}

private void btnCancelar_Click(object sender, EventArgs e)
{
    if (txtId.Enabled)
        PreencheTela(cadastroDAO.Primeiro());
    else
        PreencheTela(cadastroDAO.Consulta(Convert.ToInt32(txtId.Text)));

    AlteraParaModo(EnumModoOperacao.Navegacao);
}

private void btnAnterior_Click(object sender, EventArgs e)
{
    try
    {
        PadraoVO o = cadastroDAO.Anterior(Convert.ToInt32(txtId.Text));
        if (o != null)
            PreencheTela(o);
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}



private void btnPrimeiro_Click(object sender, EventArgs e)
{
    try
    {
        PadraoVO o = cadastroDAO.Primeiro();
        PreencheTela(o);
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}

private void btnProximo_Click(object sender, EventArgs e)
{
    try
    {
        PadraoVO o = cadastroDAO.Proximo(Convert.ToInt32(txtId.Text));
        if (o != null)
            PreencheTela(o);
    }
    catch (Exception erro)
    {
        TrataErro(erro);
    }
}

private void btnUltimo_Click(object sender, EventArgs e)
{
    try
    {
        PadraoVO o = cadastroDAO.Ultimo();
        PreencheTela(o);
    }
```

```
            catch (Exception erro)
            {
                TrataErro(erro);
            }
        }

        private void ucPadraoCad_Load(object sender, EventArgs e)
        {
            if (!ModoDeDesign()) // modo de desenvolvimento
            {
                Inicializa();
            }
        }


        public virtual void Inicializa()
        {
            btnPrimeiro.PerformClick();
            AlteraParaModo(EnumModoOperacao.Navegacao);
        }

    }
```

Criando o userControl de cadastro de técnico

Crie um novo userControl (ucCadTecnico)

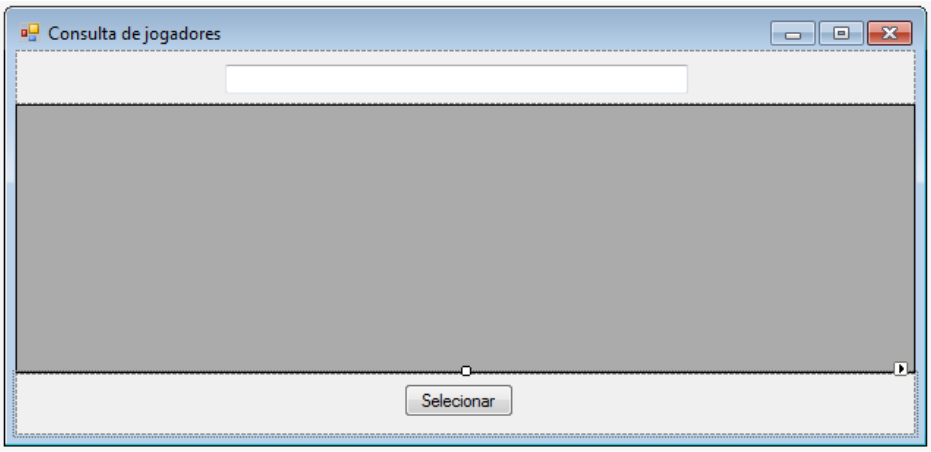


Observe que este novo user control irá herdar de ucPadraoCad.

```
public partial class ucCadTecnico : ucPadraoCad
    {
        public ucCadTecnico()
        {
            InitializeComponent();
        }

        public override void Inicializa()
        {
            cadastroDAO = new TecnicoDAO();
            SugereProximoId = true;
            base.Inicializa();
        }
```

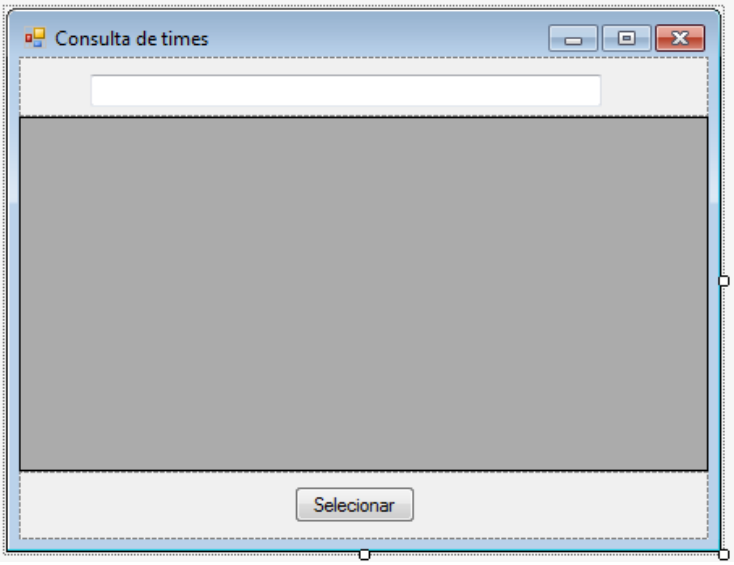
```
        protected override void ControlaCamposTela(EnumModoOperacao modo)
        {
            base.ControlaCamposTela(modo);
            txtSalario.Enabled = txtNome.Enabled = (modo != EnumModoOperacao.Navegacao);
        }

        protected override PadraoVO PreencheObj()
        {
            TecnicoVO t = new TecnicoVO();
            t.Id = Convert.ToInt32(txtId.Text);
            t.Nome = txtNome.Text;
            t.Salario = Convert.ToDouble(txtSalario.Text);
            return t;
        }


        protected override void PreencheTela(PadraoVO o)
        {
            try
            {
                if (o != null)
                {
                    TecnicoVO j = o as TecnicoVO;
                    txtId.Text = j.Id.ToString();
                    txtNome.Text = j.Nome;
                    txtSalario.Text = j.Salario.ToString();
                }
                else
                {
                    LimpaCampos(this);
                }
            }
            catch (Exception erro)
            {
                TrataErro(erro);
            }
        }

        private void btnPesquisa_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Não fiz, ainda...");
        }
    }
```

Criando a tela de consulta de jogadores



```
public partial class frConsultaJogadores : Form
{
    public int IdSelecionado { get; private set; }
    public frConsultaJogadores()
    {
        InitializeComponent();
        FazBusca();
    }

    private void txtNome_TextChanged(object sender, EventArgs e)
    {
        FazBusca();
    }

    private void FazBusca()
    {
        List<JogadorFutebolVO> lista = (new JogadorFutebolDAO()).Lista(txtNome.Text.Trim());
        dataGridView1.DataSource = lista;
        dataGridView1.ReadOnly = true;
    }

    private void btnSelecionar_Click(object sender, EventArgs e)
    {
        if (dataGridView1.CurrentRow != null)
        {
            IdSelecionado = Convert.ToInt32(dataGridView1.CurrentRow.Cells["Id"].Value);
            Close();
        }
    }

    private void dataGridView1_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        btnSelecionar.PerformClick();
    }
}
```

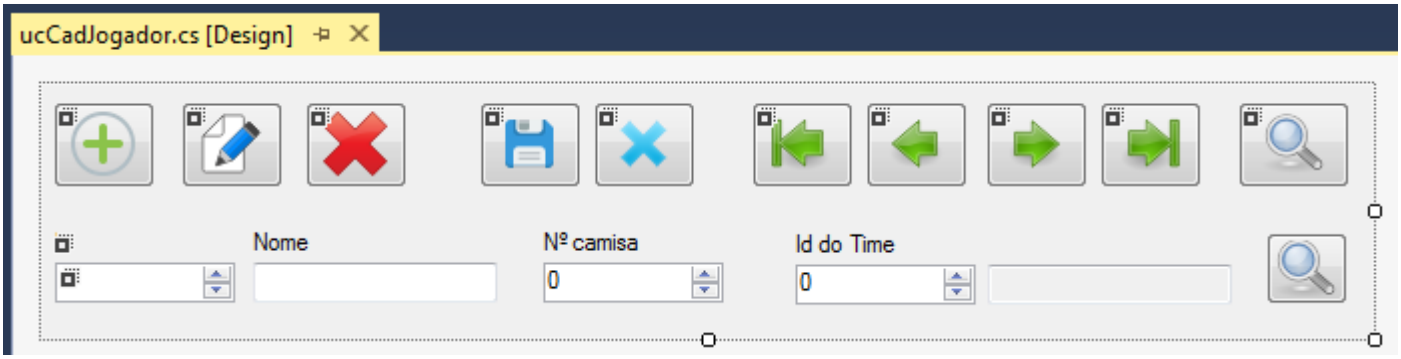
Criando a tela de consulta de times



```
public partial class frConsultaTimes : Form
{
    public int IdSelecionado { get; private set; }
    public frConsultaTimes()
    {
        InitializeComponent();
        FazBusca();
    }


    private void textBox1_TextChanged(object sender, EventArgs e)
    {
        FazBusca();
    }


    private void FazBusca()
    {
        List<TimeVO> lista = (new TimeDAO()).Lista(txtNome.Text.Trim());
        dataGridView1.DataSource = lista;
        dataGridView1.ReadOnly = true;
    }

    private void btnSelecionar_Click(object sender, EventArgs e)
    {
        if (dataGridView1.CurrentRow != null)
        {
            IdSelecionado = Convert.ToInt32(dataGridView1.CurrentRow.Cells["Id"].Value);
            Close();
        }
    }

    private void dataGridView1_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        btnSelecionar.PerformClick();
    }
}
```

Criando o userControl de cadastro de jogadores

Agora iremos criar um userControl para o cadastro de jogadores (ucCadJogador). Este usercontrol também irá herdar de ucPadrao.



```
public partial class ucCadJogador : ucPadraoCad
{
    public ucCadJogador()
    {
        InitializeComponent();
    }

    public override void Inicializa()
    {
        cadastroDAO = new JogadorFutebolDAO();
        base.Inicializa();
    }

    protected override void ControlaCamposTela(EnumModoOperacao modo)
    {
        base.ControlaCamposTela(modo);
        txtNome.Enabled = (modo != EnumModoOperacao.Navegacao);
        txtNumeroCamisa.Enabled = (modo != EnumModoOperacao.Navegacao);
        txtIdTime.Enabled = (modo != EnumModoOperacao.Navegacao);
    }

    protected override PadraoVO PreencheObj()
    {
        JogadorFutebolVO j = new JogadorFutebolVO();
        j.Id = Convert.ToInt32(txtId.Text);
        j.Nome = txtNome.Text;
        j.NumeroCamisa = Convert.ToInt32(txtNumeroCamisa.Text);
        j.TimeId = Convert.ToInt32(txtIdTime.Text);
        return j;
    }


    protected override void PreencheTela(PadraoVO o)
    {
        try
        {
            JogadorFutebolVO j = o as JogadorFutebolVO;
            if (j != null)
            {
                txtId.Text = j.Id.ToString();
                txtNome.Text = j.Nome;
                txtNumeroCamisa.Value = j.NumeroCamisa;
                txtIdTime.Value = j.TimeId;
                TimeVO t = (new TimeDAO()).Consulta((int)txtIdTime.Value) as TimeVO;
                if (t != null)
                    txtNomeTime.Text = t.Nome;
                else
                    txtNomeTime.Clear();
            }
            else
            {
                LimpaCampos(this);
            }
        }
        catch (Exception erro)
        {
            TrataErro(erro);
        }
    }


    private void btnBuscaTime_Click(object sender, EventArgs e)
    {
        using (frConsultaTimes form = new frConsultaTimes())
        {
            form.ShowDialog();
```

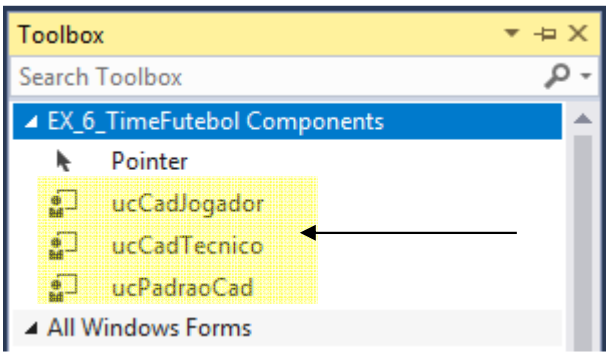
```
            if (form.IdSelecionado != 0)
                txtIdTime.Value = form.IdSelecionado;
        }
    }


    private void txtIdTime_Leave(object sender, EventArgs e)
    {
        TimeVO t = (new TimeDAO()).Consulta((int)txtIdTime.Value) as TimeVO;
        if (t != null)
            txtNomeTime.Text = t.Nome;
        else
            txtNomeTime.Text = "";

    }

    private void btnPesquisa_Click_1(object sender, EventArgs e)
    {
        using (frConsultaJogadores form = new frConsultaJogadores())
        {
            form.ShowDialog();
            if (form.IdSelecionado != 0)
            {
                var j = cadastroDAO.Consulta(form.IdSelecionado);
                PreencheTela(j as JogadorFutebolVO);
            }
        }
    }
}
```

Formulário principal para exibir os usercontrols de cadastro

Compile seu projeto, até que dê sucesso!

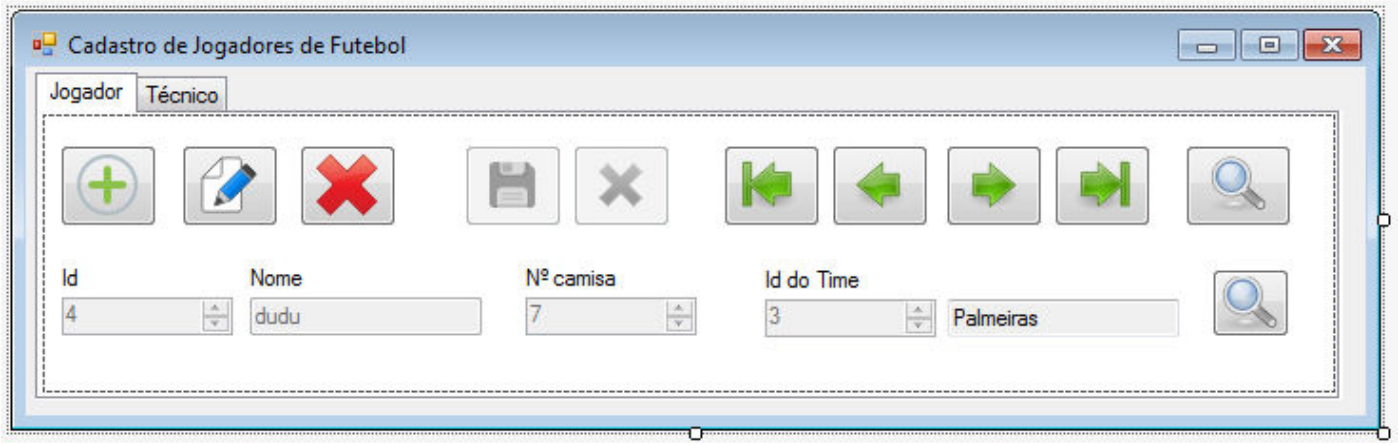
Crie um novo formulário (ou user o Form1 caso ele já exista no seu projeto) e adicione um componente TabControl.

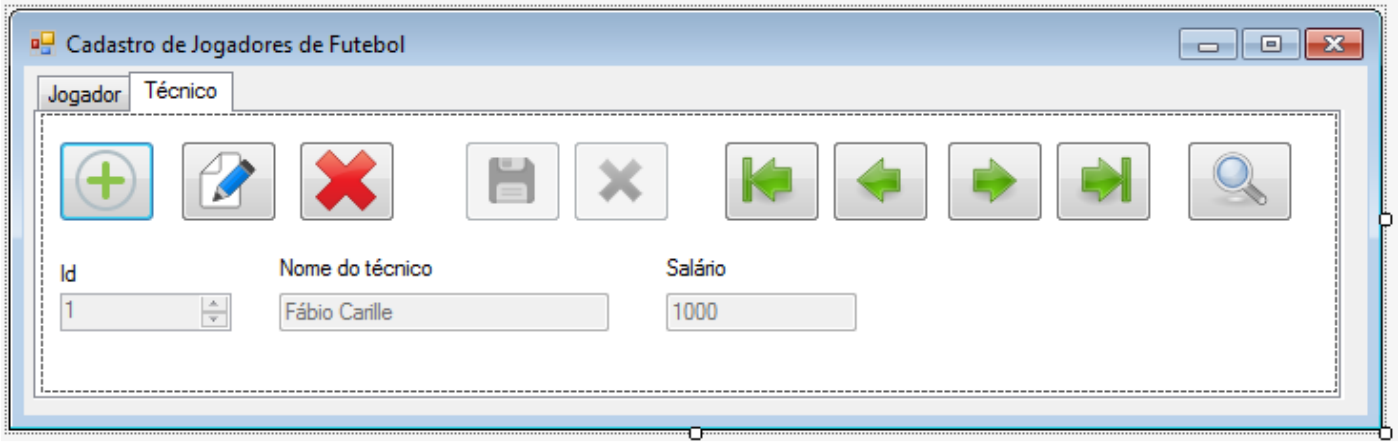
Observe que os usercontrols foram adicionados na sua ToolBox:



Atenção: A cada mudança que fizer no seu usercontrol, você precisa recompilar o projeto inteiro (CTRL + SHIFT +B)

Arraste o ucCadJogador para a primeira aba do seu TabControl e o ucCadTecnico para a segunda aba. Veja na imagem a seguir:





Este é o código fonte:

```
public partial class frTelaCadastro : Form
    {
        public frTelaCadastro()
        {
            InitializeComponent();
        }
    }
```

É só isso mesmo :)