Prof. Roberto de Castro

Escola Técnica Estadual "São Paulo" - ETESP
Linguagem C# - Visual Studio 2019

E agora estudaremos o terceiro modelo de acesso a dados, utilizando conceitos da Programação Orientada a Objetos.

O projeto que iremos desenvolver realizará a "manutenção" na "tabela de usuários". Estudaremos o modelo de desenvolvimento utilizando "duas camadas": a de apresentação - (view/formulário) e a de acesso/manipulação da tabela no "Banco de Dados".

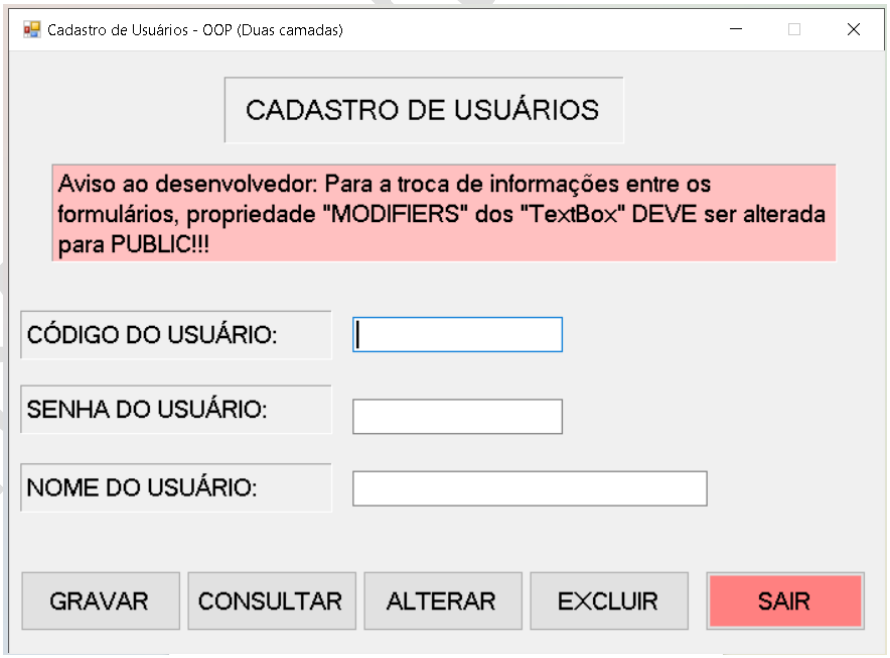
OBS: Em outro projeto estudaremos o modelo de três cadamas.

Tabela de usuário (nome Usuarios). OBS: Criar no Access!!!

Após a finalização deste projeto, faremos uma alteração: o acesso pelo SQL Server!!!

Campo	Tipo	Observação
CodUsuario	Número	Campo Chave
Senha	Número	Requerido / Regra Validação > 0
NomeUsuario	Texto	Tamanho 50 / Requerido

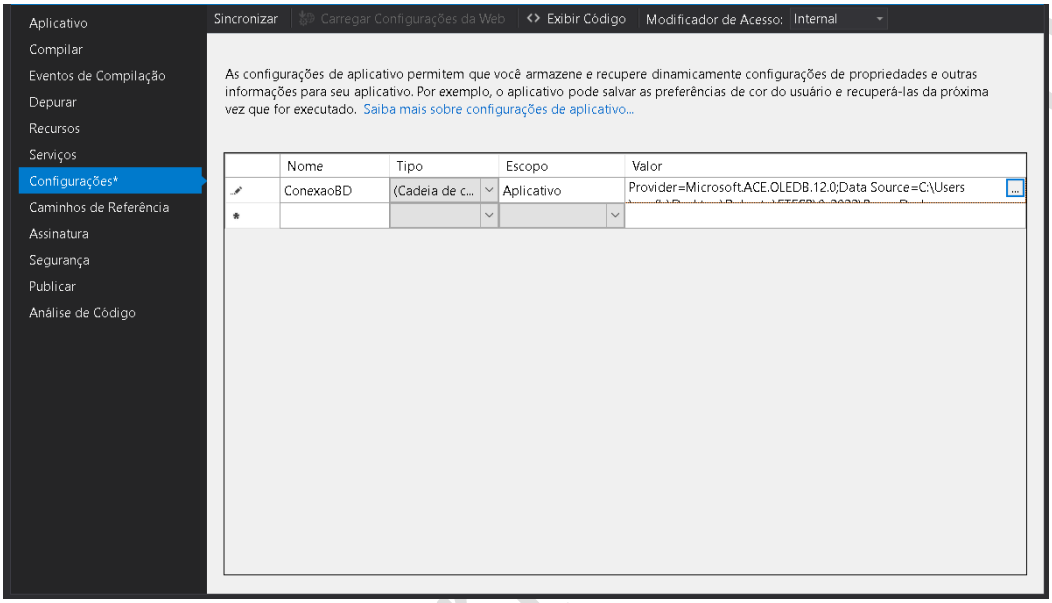
Vamos iniciar um novo projeto no Visual Studio (EXE039_BD05) e desenhar o formulário "FrmUsuarios":



Considerações iniciais sobre o formulário "FrmUsuarios":

1. A propriedade "MODIFIERS" das caixas de texto (TextBox) deverá ser alterada para "PUBLIC", para que seja possível a troca de informações entre os formulários de "Cadastro" e "Consulta".
2. O formulário será exibido e já será possível a digitação de "dados" para gravação. As caixas de texto, neste momento, estarão destravadas. O travamento/destravamento poderá ser feito oportunamente.

3. Ao clicar em “Consultar” o projeto exibirá o formulário “FrmConsulta” (ver modelo abaixo).
4. As opções “Alterar” e “Excluir” somente terão “funcionalidade” após o usuário fazer uma consulta e o sistema “trazer” os dados para o formulário “FrmUsuarios”.
5. Vamos aproveitar que estamos com este formulário aberto (FrmUsuarios), para gerar uma informação importante: a “Connection String” (variável que irá conter o nome e o caminho do Banco de Dados).
   Clique em Projeto → Propriedades (ou Project → Properties) → Configurações (Settings) → e vamos adicionar a variável de “Conection String”:



<u>Agora, vamos adicionar ao projeto, o Formulário “FrmConsulta”:</u>

Clique em Projeto (Project) → Adicionar Formulário (Add Form). Altere o nome para “FrmConsulta”.

<u>Considerações iniciais sobre o formulário “FrmConsulta”:</u>

1. O usuário poderá consultar/listar o cadastro através das opções:

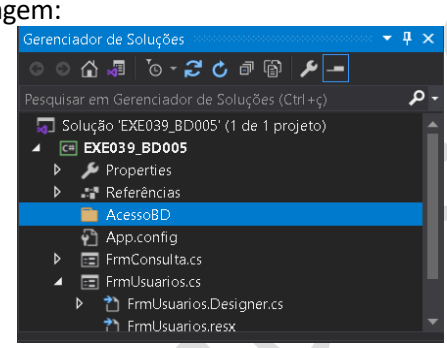
i. Listar Geral → todos os usuários serão exibidos no DataGridView.

ii. Consulta por código → somente o usuário com o "código" informado, será exibido.

iii. Consulta por nome → uma "consulta" avançada" que exibirá todos os usuários com o nome "parcial" ou "completo".

2. Name do DatagridView → DgvLista.
3. A propriedade "SelectMode" do DataGridView foi alterada para "FullRowSelect", que significa que toda a linha do "DataGridView" será "selecionada" quando o "click" for acionado.
4. Ao clicar em uma "linha" do DataGridView os dados do registro serão exibidos nos "TextBox" do formulário "FrmUsuario", que poderá Excluir ou Alterar os dados.

Vamos "adicionar" uma "pasta" (Folder) em nosso projeto, para "sinalizar visualmente" que iremos incluir uma nova camada. Pressione o botão direito sobre o nome do projeto e clique em "Adicionar → nova pasta" (ou Add New Folder). Renomeie a pasta para "AcessoBD". Neste momento o "Gerenciador de Soluções" (Solution Explorer) deverá ter a imagem:

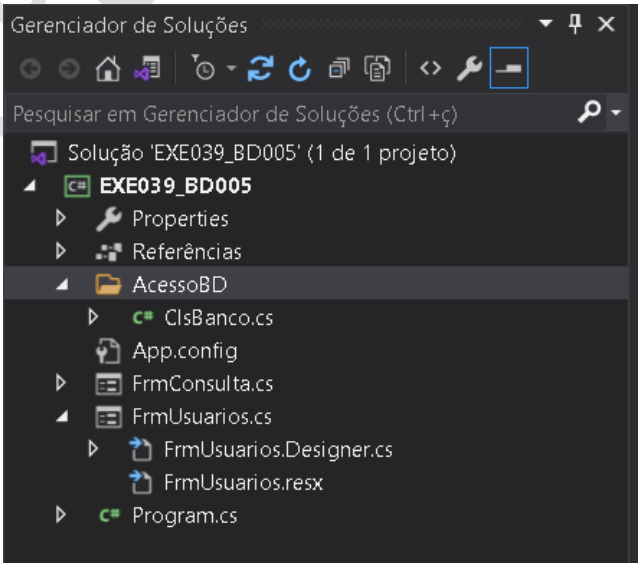


Vamos inserir uma "classe", dentro da pasta "AcessoBD". Pressione o botão direito sobre a pasta "AcessoBD", selecione a opção: "Adicionar → Classe".

IMPORTANTE: MUITA ATENÇÃO NESTE MOMENTO PARA NÃO ADICIONAR ALGO DIFERENTE!!!

Nomeie a classe para: **ClsBanco.**

Neste momento, se observar, o "Gerenciador de Soluções" deverá exibir a imagem:



Faremos uma programação “parcial” na classe “ClsBanco” (que está dentro da pasta “AcessoBD”). Incluiremos os campos e as propriedades (gets/sets) de nossa tabela e também o método construtor. Abra a classe e insira a programação abaixo:

```
using System;
//Observe a inclusão do NameSpace para acessar Banco de Dados
using System.Data;
using System.Data.OleDb;

//e também do Namespace para utilizar o MessageBox
using System.Windows.Forms;


namespace EXE039_BD005.AcessoBD
{
   class ClsBanco
   {
      //Declaração da "variável" que contém informações
      //sobre o Banco de Dados: tipo, nome, caminho...
      string strConexao = Properties.Settings.Default.ConexaoBD;

      //Declaração da "variável" (objeto) de Conexão
      OleDbConnection conn = null;


      //Declaração dos campos (que constam na tabela de usuarios)
      //OBS: O atributo "private" é opcional. Se nada for declarado
      //o "private" é assumido!!
      private int codUsuario;
      private int senha;
      private string nomeUsuario;


      //Declaração das "propriedades" (Gets/Sets).
      //Botão direito sobre cada campo declarado acima, ações rápidas e
      //refatoração --> Encapsular Campos
      public int CodUsuario { get => codUsuario; set => codUsuario = value; }
      public int Senha { get => senha; set => senha = value; }
      public string NomeUsuario { get => nomeUsuario; set => nomeUsuario = value; }

      //Declaração do método construtor. Método executado automaticamente
      //no momento em que a classe é "instanciada". Abertura do Banco de Dados
      public ClsBanco()
      {
         conn = new OleDbConnection(strConexao);
         try
         {
            conn.Open();
         }
         catch (Exception ex)
         {
            MessageBox.Show("Erro ao abrir o arquivo: " + ex.ToString(), "ATENÇÃO");
```

```
      }
    }
  }
}
```

Programação do botão "Salvar" (Formulario FrmUsuarios)

```
private void BtnGravar_Click(object sender, EventArgs e)
        {
            //Atenção: Falta validar as caixas de TexBox!!!

            //Instancia a classe ClsBanco (neste momento o método
            //construtor é executado
            //*** Importante: o using precisará ser declarado para poder utilizar a classe
              ClsBanco que
            //está dentro da pasta AcessoBD --> using EXE039_BD005.AcessoBD;
            ClsBanco objBanco = new ClsBanco();

            //Transfere os dados das caixas de texto para as propriedades
            //sets da classe
            objBanco.CodUsuario = Convert.ToInt32(TxtCodigo.Text);
            objBanco.Senha = Convert.ToInt32(TxtSenha.Text);
            objBanco.NomeUsuario = TxtNome.Text;

            //executa o método gravar da classe ClsBanco.
            //*** Neste momento o método não existe, porém, o Visual Studio
            //exibe a opção de cria-lo. E nós utilizaremos esta opção
            //de criação!!

            int status;
            status = objBanco.Gravar();

            //Verifica o retorno da gravação
            if (status != 0)
            {
                MessageBox.Show("Gravação bem sucedida!!", "SUCESSO");
            }
            else
            {
                MessageBox.Show("Erro na gravação!!", "*** ERRO ***");
            }
            LimparCampos();
        }


        private void LimparCampos()
        {
            TxtCodigo.Text = "";
            TxtSenha.Text = "";
            TxtNome.Text = "";
            TxtCodigo.Focus();
        }
```

Programação do método "Gravar" da classe ClsBanco:

```
public int Gravar()
        {
            //declaração do objeto/variável "command"
            OleDbCommand comando = new OleDbCommand();

            comando.CommandType = CommandType.Text;


            //declaração dos parametros/variáveis que serão utilizadas
            //no comando SQL. Os parâmetros recebem os dados dos Gets
            comando.Parameters.Add("@varCodigo", oleDbType: OleDbType.Integer).Value =
                    codUsuario;
            comando.Parameters.Add("@varSenha", oleDbType: OleDbType.Integer).Value = senha;
            comando.Parameters.Add("@varNome", oleDbType: OleDbType.VarChar, 50).Value =
                    nomeUsuario;

            //comando SQL
            comando.CommandText="Insert into Usuarios (CodUsuario, Senha, NomeUsuario)
                values (@varCodigo,@varSenha, @varNome)";

            comando.Connection = conn;

            //Executa a instrução
            int status = comando.ExecuteNonQuery();

            //Fecha o banco
            conn.Close();
            return status;
        }
```

Faremos agora a implementação do botão "Consultar" que está no formulário "FrmUsuarios". Ao clicar no botão "Consultar" o projeto deverá exibir o formulário de listagem "FrmConsulta".

IMPORTANTE: A chamada do formulário "FrmConsulta" terá uma "novidade", pois, além de "abrir" o formulário "FrmConsulta", deverá, também, enviar o "FrmUsuarios" como parâmetro, para que o retorno dos dados, que estarão no "FrmConsulta", possam ser exibidos no formulário "FrmUsuarios".

Veja a programação do botão "Consultar":

```
private void BtnConsultar_Click(object sender, EventArgs e)
    {
      //Observe o parametro "This" ao final. Indica que estamos
      //transferindo o formulário atual (FrmUsuarios) como
      //parametro para o formulário "FrmConsulta".
      //No momento da escrita desta instrução um erro será sinalizado
      //pois, no momento, o FrmConsulta não está "preparado" para
      //receber dados de parametro. Esta implementação do
      //parametro será feita logo em seguida no próprio formulário
      //FrmConsulta
      FrmConsulta MyFrmConsulta = new FrmConsulta(this);
      MyFrmConsulta.ShowDialog();
    }
```

E agora o “ajuste” necessário no formulário “FrmConsulta” para receber o “FrmUsuarios” como parâmetro e a programação completa:

```
using System;

using System.Windows.Forms;

//O using abaixo é necessário para poder acessar
//a classe ClsBanco que está dentro da pasta "AcessoBD
using EXE039_BD005.AcessoBD;

namespace EXE039_BD005
{
    public partial class FrmConsulta : Form
    {
        //Declaração de um objeto do tipo "FrmUsuarios"
        FrmUsuarios instanciaDoForm1;

        //O parametro (FrmUsuarios frm1) foi inserido, com o objetivo de
        //informar que o formulário FrmConsulta receberá um formulário
        //como parametro
        public FrmConsulta(FrmUsuarios frm1)
        {
            InitializeComponent();

            //instrução incluída
            instanciaDoForm1 = frm1;
        }

        private void BtnVoltar_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void BtnPesquisar_Click(object sender, EventArgs e)
        {
            //Gera a instancia da classe "ClsBanco".
            //Neste momento o construtor é executado e o banco aberto
            ClsBanco objeto = new ClsBanco();

            //Declara uma variável do tipo Byte
            byte tipoConsulta = 1;

            if (RdoCodigo.Checked == true)
            {
                tipoConsulta = 2;
                TxtNome.Text = "";
            }
            else if (RdoNome.Checked == true)
            {
                tipoConsulta = 3;
                TxtCodigo.Text = "";
            }

            //tenta converter o conteúdo da caixa de texto TxtCodigo
            //se der erro (nada foi preenchido), o código será Zero
            int codigo;
            try
            {
                codigo = Convert.ToInt32(TxtCodigo.Text);
```

```
            }
            catch (Exception)
            {
                codigo = 0;
            }

            //OBS: No momento em que a instrução abaixo é digitada
            //um erro ocorrerá, informando que o método "ConsultarDados"
            //não existe. O método poderá ser criado pelo próprio Visual
            //Studio e "ajustado" manualmente
            DgvLista.DataSource = objeto.ConsultarDados(tipoConsulta, codigo, TxtNome.Text);

            if (DgvLista.RowCount == 0)
            {
                MessageBox.Show("Nenhum usuario encontrado!!", "Atenção");
            }
        }


        private void DgvLista_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            //importante: No Form1 (FrmUsuarios) precisamos alterar as propriedades
            //MODIFIERS dos textbox para public para permissão de acesso
            instanciaDoForm1.TxtCodigo.Text = DgvLista.CurrentRow.Cells[0].Value.ToString();
            instanciaDoForm1.TxtSenha.Text = DgvLista.CurrentRow.Cells[1].Value.ToString();
            instanciaDoForm1.TxtNome.Text = DgvLista.CurrentRow.Cells[2].Value.ToString();

            this.Close();
        }
    }
}
```

Programação do método "ConsultarDados" que está na classe "ClsBanco":

```
//A linha abaixo foi criada pelo Visual Studio, porém, precisamos
//escreve-la corretamente
//internal object ConsultarDados(byte tipoConsulta, int v, string text)
public DataTable ConsultarDados(byte tipoConsulta, int codigo, string nome)
{
           string strSql = "Select * from Usuarios";
            // tipoConsulta = 1 --> lista geral
            // tipoConsulta = 2 --> Consulta por código
            // tipoConsulta = 3 --> Consulta por qualquer parte do nome

            if (tipoConsulta == 2)
            {
                //declaração dos parametros/variáveis que serão utilizadas
                //no comando SQL

                comando.Parameters.Add("@varCodigo", oleDbType: OleDbType.Integer).Value =
                   codigo;
                strSql += " where CodUsuario = @varCodigo";
            }
            else if (tipoConsulta == 3)
            {
                comando.Parameters.Add("@varNome", oleDbType: OleDbType.VarChar, 50).Value =
                  "%" + nome + "%";
                strSql += " where NomeUsuario like @varNome";
            }
```

```
            strSql += " order by CodUsuario";

             //atribui a variavel strSql ao comando
             comando.CommandType = CommandType.Text;
             comando.CommandText = strSql;

             comando.Connection = conn;

             DataSet myDataset = new DataSet();

             //cria e executa o comando SQL para preencher o Dateset
             OleDbDataAdapter myAdapter = new OleDbDataAdapter(comando);

             myAdapter.Fill(myDataset, "tabelaDataset");

             conn.Close();
             return myDataset.Tables["tabelaDataset"];
        }
```

Seguiremos agora com a programação dos botões “Alterar” e “Excluir”.

IMPORTANTE: Estas opções (Alterar / Excluir) somente poderão ser executadas após uma consulta, com o preenchimento das caixas de texto.

```
//IMPORTANTE: Este método de "Alterar" somente poderá ser
//executado após a realizaçaõ de uma consulta,
//com o preenchimento das caixas de texto.
private void BtnAlterar_Click(object sender, EventArgs e)
{
  //Atenção: Falta validar as caixas de TexBox!!!

  //Instancia a classe ClsBanco (neste momento o método
  //construtor é executado
  //*** Importante: o using precisará ser declarado para poder utilizar a classe //ClsBanco que
  //está dentro da pasta AcessoBD --> using EXE039_BD005.AcessoBD;
  ClsBanco objBanco = new ClsBanco();

  //Transfere os dados das caixas de texto para as propriedades
  //sets da classe.
  objBanco.CodUsuario = Convert.ToInt32(TxtCodigo.Text);
  objBanco.Senha = Convert.ToInt32(TxtSenha.Text);
  objBanco.NomeUsuario = TxtNome.Text;

  //executa o método "Alterar" da classe ClsBanco.
  //*** Neste momento o método não existe, porém, o Visual Studio
  //exibe a opção de cria-lo. E nós utilizaremos esta opção
  //de criação e faremos os ajustes necessários

  int status;
  status = objBanco.Alterar();

  //Verifica o retorno da alteração
  if (status != 0)
```

```
        {
            MessageBox.Show("Alteração bem sucedida!!", "SUCESSO");
        }
        else
        {
            MessageBox.Show("Erro na alteração dos dados!!", "*** ERRO ***");
        }
        LimparCampos();
    }


    //IMPORTANTE: Este método de "Excluir" somente poderá ser
    //executado após a realização de uma consulta,
    //com o preenchimento das caixas de texto.
    private void BtnExcluir_Click(object sender, EventArgs e)
    {
        //Atenção: Falta validar as caixas de TexBox!!!

        //Instancia a classe ClsBanco (neste momento o método construtor é executado
        //*** Importante: o using precisará ser declarado para poder utilizar a classe //ClsBanco que
        //está dentro da pasta AcessoBD --> using EXE039_BD005.AcessoBD;
        ClsBanco objBanco = new ClsBanco();

        //Transfere os dados das caixas de texto para as propriedades
        //sets da classe. Para a EXCLUSÃO, somente o código é necessário
        objBanco.CodUsuario = Convert.ToInt32(TxtCodigo.Text);

        //executa o método "Excluir" da classe ClsBanco.
        //*** Neste momento o método não existe, porém, o Visual Studioexibe a opção de cria-lo.
        //E nós utilizaremos esta opção de criação e faremos os ajustes necessários

        int status;
        status = objBanco.Excluir();

        //Verifica o retorno da exclusão
        if (status != 0)
        {
            MessageBox.Show("Exclusão bem sucedida!!", "SUCESSO");
        }
        else
        {
            MessageBox.Show("Erro na exclusão do usuário!!", "*** ERRO ***");
        }
        LimparCampos();
    }
```

Para finalizar este projeto, a programação dos métodos “ALTERAR” E “EXCLUIR” que estão na classe ClsBanco:

```
public int Alterar()
```

```
{
    comando.CommandType = CommandType.Text;

    //declaração dos parametros/variáveis que serão utilizadas no comando SQL

    comando.Parameters.Add("@varSenha", oleDbType: OleDbType.Integer).Value = senha;
    comando.Parameters.Add("@varNome", oleDbType: OleDbType.VarChar, 50).Value = nomeUsuario;
    comando.Parameters.Add("@varCodigo", oleDbType: OleDbType.Integer).Value = codUsuario;
    //Observe que os parametros DEVERÃO ser declarados na mesma ordem em que serão utlizados na
    //instrução SQL. A chave primária "CodUsuario" não poder ser alterada, mas será utilizada no SQL
    //com uma condição "where".

    //comando SQL
    comando.CommandText = "Update Usuarios set Senha = @varSenha, NomeUsuario = @varNome
                where CodUsuario = @varCodigo";

    comando.Connection = conn;

    //Executa a instrução
    int status = comando.ExecuteNonQuery();

    //Fecha o banco
    conn.Close();
    return status;
}


public int Excluir()
{
    comando.CommandType = CommandType.Text;

    //declaração dos parametros/variáveis que serão utilizadas
    //no comando SQL

    comando.Parameters.Add("@varCodigo", oleDbType: OleDbType.Integer).Value = codUsuario;

    //comando SQL
    comando.CommandText = "Delete from Usuarios where CodUsuario = @varCodigo";

    comando.Connection = conn;

    //Executa a instrução
    int status = comando.ExecuteNonQuery();

    //Fecha o banco
    conn.Close();
    return status;
}
```