C# - Cadastro de Clientes no MySQL CRUD (ADO .NET)

Programar se aprende programando, certo ?

E nada melhor para aprender do que ter um protótipo funcional com o código fonte a partir do qual podemos estudar.

Este singelo projeto feito na linguagem C# realiza o acesso e as operações de inclusão, edição e exclusão em uma base de dados MySQL.

Então vamos ao que interessa...

O projeto poderá ser aberto nos seguintes **IDEs:**

- Visual Studio 2010 *(você pode baixar uma versão trial)*
- Visual C# 2010 Express Edition - (é grátis)
- SharpDevelop 4.0 - (é grátis)

Além disso você deverá instalar os seguintes recursos:

- **MysQL** -> http://dev.mysql.com/downloads/mysql/
- **Connector/NET** -> http://www.mysql.com/products/connector/

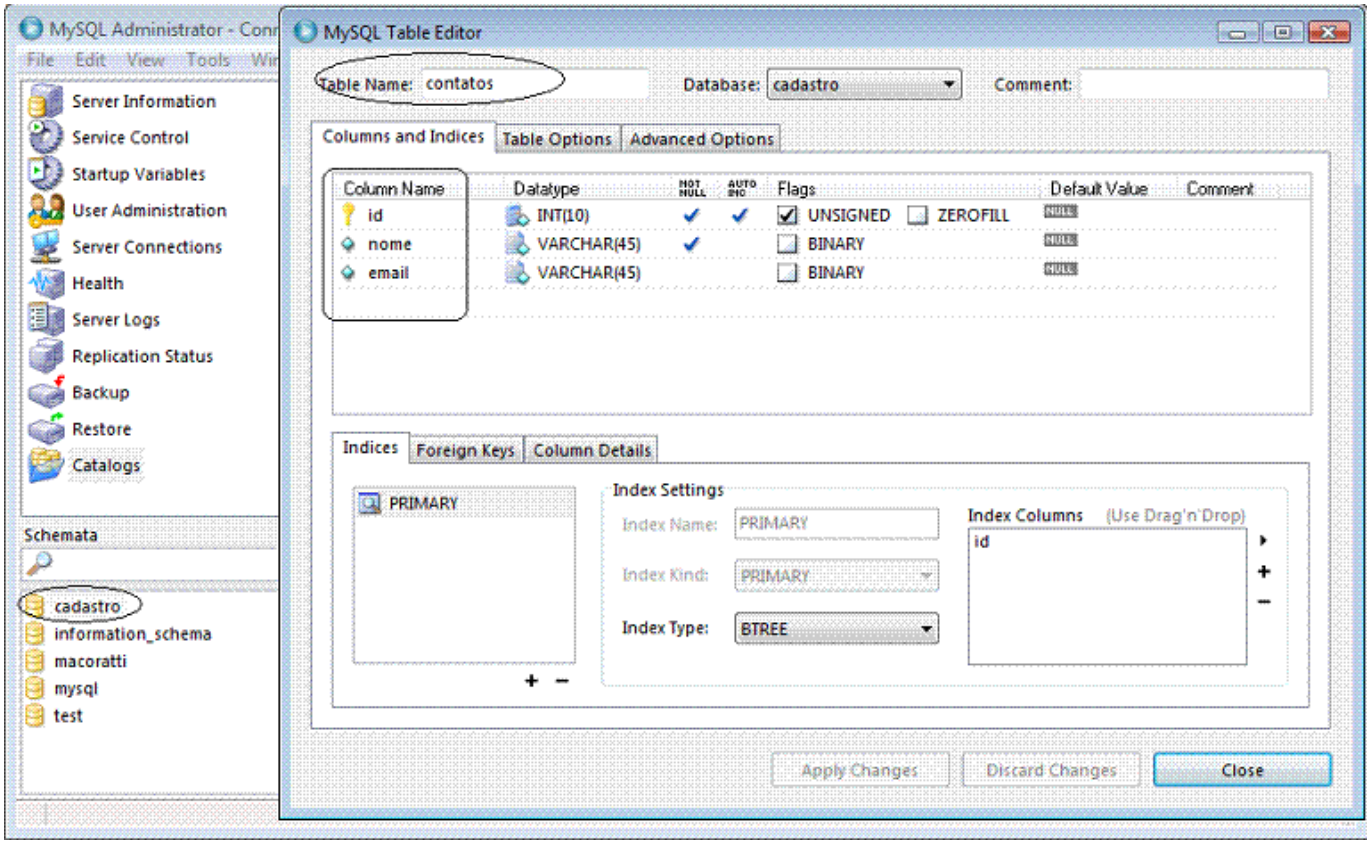
As telas e o código que eu vou mostrar no artigo foram obtidos a partir do **SharpDevelop 4.0.**

Esta é uma aplicação para quem esta iniciando com a linguagem C# e pretende acessar e realizar as operações ***CRUD (Create, Read, Update e Delete)*** em um banco de dados.

O banco de dados MySQL

A aplicação acessa a tabela **Contatos** de um banco de dados **MySQL** definido como **Cadastro**.

Na figura abaixo temos a estrutura da tabela Contatos exibida no **MySQL Administrator:**



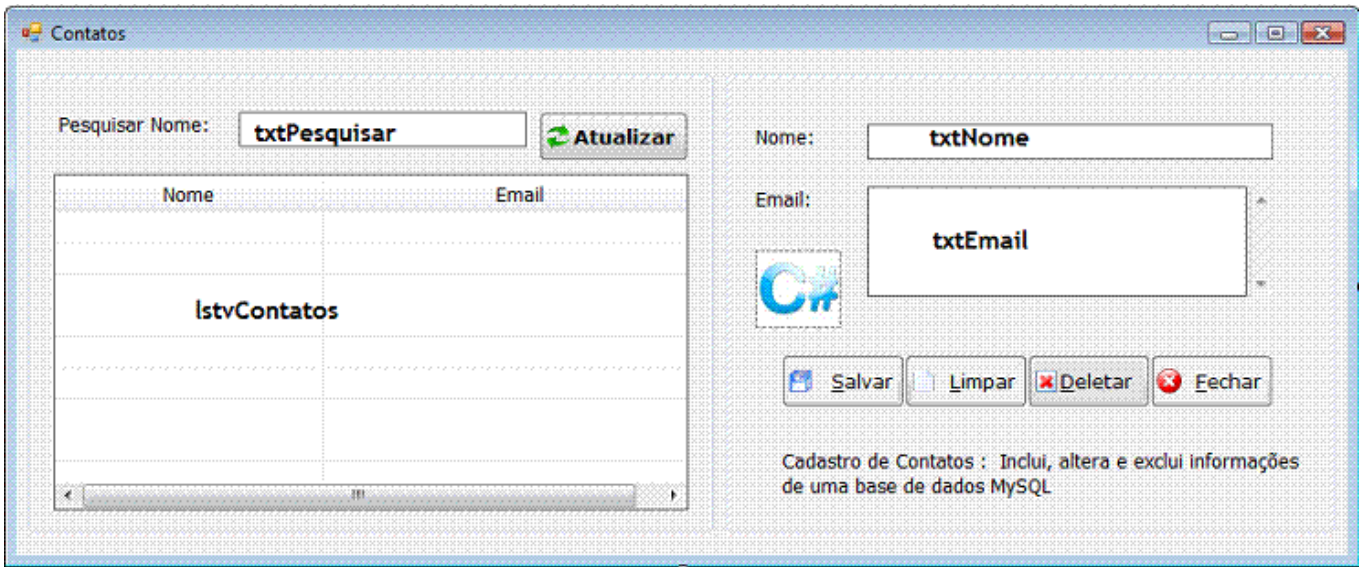
O script para criar o banco de dados e a tabela no MySQL é dado a seguir:

```
CREATE DATABASE /*!32312 IF NOT EXISTS*/ Cadastro;
USE Cadastro;
--
-- Table structure for table `Cadastro`.`Contatos`
--
DROP TABLE IF EXISTS `Contatos`;
CREATE TABLE `Contatos` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `nome` varchar(50) default NULL,
  `email` varchar(100) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='InnoDB free: 3072 kB';
```

A interface da aplicação

A seguir temos a interface da aplicação no formulário form1.cs que utiliza os seguintes controles:

1. ListView - lstvContatos;
2. TextBox - txtPesquisar, txtNome e txtEmail;
3. Button - btnAtualizar, btnSalvar, btnLimpar, btnDeletar e btnFechar;



O código da aplicação

No início do formulário temos a definição dos namespaces usados no projeto

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using MySql;
using MySql.Data;
using MySql.Data.MySqlClient;
```

Note que estamos referenciando o namespace para acessar as classes ADO .NET para o MySQL.

A seguir logo após a declaração do formulário temos a definição das variáveis ADO .NET para conexão com o banco de dados **MySQL**:

```
MySqlConnection conMySQL = new MySqlConnection("uid=root; password=*****; database=cadastro");
MySqlCommand cmdMySQL = new MySqlCommand();
MySqlDataReader reader;
```

Se você predente usar outro banco de dados basta alterar os nomes dessas variáveis conforme o provedor pertinente.

1 - Código do botão Salvar

```
private void btnSalvar_Click(object sender, System.EventArgs e)
{
   try
   {
            if(txtNome.Text=="")
            {
               MessageBox.Show("Informe o nome do
contato.","Warning",MessageBoxButtons.OK,MessageBoxIcon.Warning);
                txtNome.Focus();
            }
            else if(txtEmail.Text=="")
            {
                MessageBox.Show("Informe o email do
contato.","Warning",MessageBoxButtons.OK,MessageBoxIcon.Warning);
```

```
                txtNome.Focus();
            }
            else
            {
             if(status=="novo")
            {
               cmdMySQL.CommandText = "INSERT INTO Contatos(nome,email)
VALUES('"+txtNome.Text+"','"+txtEmail.Text+"')";
               cmdMySQL.ExecuteNonQuery();
               cmdMySQL.Dispose();
               MessageBox.Show("Registro salvo com
sucesso.","Salvar",MessageBoxButtons.OK,MessageBoxIcon.Information);
            }
            else if(status=="editar")
            {
               cmdMySQL.CommandText = "UPDATE Contatos SET nome='"+txtNome.Text+"',
email='"+txtEmail.Text+
                                          "' WHERE
id='"+lstvContatos.Items[lstvContatos.FocusedItem.Index].Text+"'";
               cmdMySQL.ExecuteNonQuery();
               MessageBox.Show("Registro atualizado com
sucesso.","Atualizar",MessageBoxButtons.OK,MessageBoxIcon.Information);
            }
            carregaVars();
            btnLimpar.PerformClick();
            }
    }
    catch(Exception ex)
   {
      MessageBox.Show(ex.ToString());
   }
}
```

O código do botão Salvar verificamos o conteúdo da variável status e se o valor for igual a 'novo' realizamos uma inclusão usando a instrução SQL:

```
cmdMySQL.CommandText = "INSERT INTO Contatos(nome,email)
VALUES('"+txtNome.Text+"','"+txtEmail.Text+"')";
cmdMySQL.ExecuteNonQuery();
cmdMySQL.Dispose();
```

O objeto **Command** da ADO.NET fornece o método ExecuteNonQuery para executar consultas que não retornam linhas (registros). Apesar de não retornar registros , qualquer parâmetro de saída ou valores retornados mapeados para parâmetros do objeto Comando são preenchidos com dados.

O método ExecuteNonQuery retorna o número de linhas afetados pelas operações de Insert , Update e Delete. Para todas as demais consultas o valor retornado é **-1**.

Quando uma consulta falha na execução o provedor gerenciado dispara uma exceção que você pode capturar no seu código.

Os provedores gerenciados ADO.NET possuem classes que fazem o tratamento das exceções. Esta classe de exceção é criada e disparada quando um erro é

encontrado.

Se o valor da variável status for igual a editar realizamos uma alteração usando a seguinte instrução SQL:

```
cmdMySQL.CommandText = "UPDATE Contatos SET nome='"+txtNome.Text+"', email='"+txtEmail.Text+
                       "' WHERE id='"+lstvContatos.Items[lstvContatos.FocusedItem.Index].Text+"'";
cmdMySQL.ExecuteNonQuery();
```

2 - Código do botão Deletar

```
private void bttnDeletar_Click(object sender, System.EventArgs e)
{
  if (MessageBox.Show("Deseja encerrar a aplicação?", "Encerrar", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
  {
     try
     {
         cmdMySQL.CommandText = "DELETE FROM Contatos WHERE id='" +
lstvContatos.Items[lstvContatos.FocusedItem.Index].Text + "'";
         cmdMySQL.ExecuteNonQuery();
          MessageBox.Show("Registro deletado com sucesso.", "Deletar", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
          carregaVars();
          btnLimpar.PerformClick();
     }
     catch (Exception ex) { MessageBox.Show(ex.ToString()); }
    }
}
```

O código do botão Deletar utiliza a instrução SQL :

```
cmdMySQL.CommandText = "DELETE FROM Contatos WHERE id='" +
lstvContatos.Items[lstvContatos.FocusedItem.Index].Text + "'";
cmdMySQL.ExecuteNonQuery();
```

Note que o Id do contato esta sendo obtido a partir do contato selecionado a partir do controle ListView.

3 - Código do evento Click do ListView

```
 private void lstvContatos_Click(object sender, System.EventArgs e)
{
 try
 {
            txtNome.Text = lstvContatos.Items[lstvContatos.FocusedItem.Index].SubItems[1].Text;
            txtEmail.Text = lstvContatos.Items[lstvContatos.FocusedItem.Index].SubItems[2].Text;
            status="editar";
 }
 catch(Exception){MessageBox.Show("Não existem registros na
lista.","Warning",MessageBoxButtons.OK,MessageBoxIcon.Warning);}
}
```

No evento Click do controle ListView estamos obtendo os valores para o nome e o email do contato e exibindo nos controles de formulário.

4- Código da rotina carregaVars

```
/// <summary>
/// CarregaVars - Preenche o controle ListView com os dados da tabela Contatos
/// </summary>
void carregaVars()
{
   try
   {
     lstvContatos.Items.Clear();
      if(txtPesquisar.Text=="")
      {
           cmdMySQL.CommandText = "SELECT * FROM Contatos ORDER BY nome ASC";
      }
     else
     {
       cmdMySQL.CommandText = "SELECT * FROM Contatos WHERE nome LIKE '"+txtPesquisar.Text+"%'
ORDER BY nome ASC";
     }
    reader = cmdMySQL.ExecuteReader();
    while(reader.Read())
    {
            ListViewItem list = new ListViewItem(reader[0].ToString());
            list.SubItems.Add(reader[1].ToString());
            list.SubItems.Add(reader[2].ToString());
            lstvContatos.Items.AddRange(new ListViewItem [] {list});
    }
    reader.Close();
   }
  catch(Exception ex){MessageBox.Show(ex.ToString());}
}
```

A rotina **carregaVars** preenche o controle **ListView** usando uma consulta SQL da seguinte forma:

Se a caixa de texto txtPesquisar.Text estiver vazia serão selecionados todos os registros da tabela Contatos caso contrário será feita uma consulta filtrando os dados obtidos pelo caractere que for digitado na caixa de texto;

```
if(txtPesquisar.Text=="")
{
    cmdMySQL.CommandText = "SELECT * FROM Contatos ORDER BY nome ASC";
}
else
{
    cmdMySQL.CommandText = "SELECT * FROM Contatos WHERE nome LIKE '"+txtPesquisar.Text+"%'
ORDER BY nome ASC";
}
```

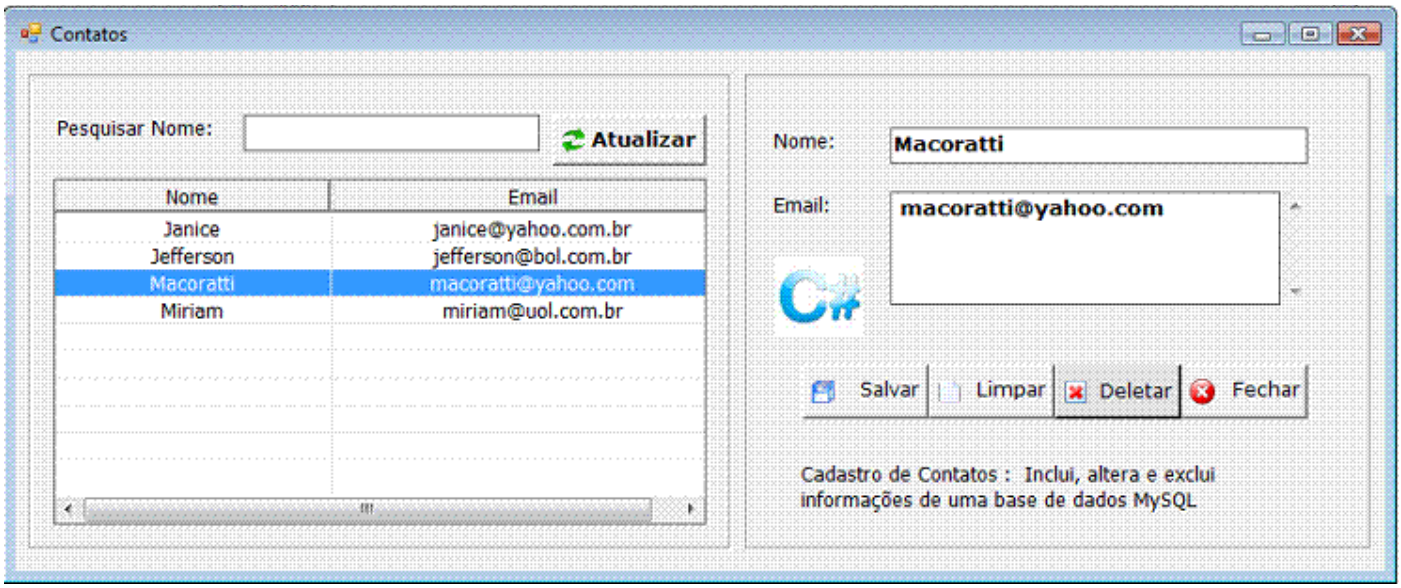
5- O código do evento Click do botão Atualizar e do evento TextChanged da caixa de Texto TxtPesquisar

```
private void btnAtualizar_Click(object sender, System.EventArgs e)
{
            txtPesquisar.Text="";
            carregaVars();
}

private void txtPesquisar_TextChanged(object sender, System.EventArgs e)
{
   carregaVars();
}
```

O código acima chama a rotina **carregaVars** quando o botão for clicado atualizado o **ListView** ou realizar um filtro conforme o caractere digitado.

Executando o projeto iremos obter:



Enfim uma aplicação simples mas que mostra como você pode realizar a manutenção dos dados acessando um banco de dados.

Pegue o projeto completo aqui: **Contatos_MySQL.zip**

Eu sei é apenas C# , mas eu gosto...

Referências:

- **Seção C# do site Macoratti.net**
- **Super DVD .NET - Centenas de aplicativos com código fonte para você aprender**

José Carlos Macoratti