Este artigo demonstrará como desenvolver um [**CRUD básico**](https://www.devmedia.com.br/criando-um-crud-em-csharp-com-xml/28453). CRUD vem do inglês Create Read Update e Delete que em tradução livre para o português seria Criar, Ler, Atualizar, e Excluir. O CRUD é composto pelas operações básicas que uma aplicação faz a um [**banco de dados**](https://www.devmedia.com.br/cursos/banco-de-dados).

O CRUD que será mostrado neste artigo foi desenvolvido de uma forma bem simples para aqueles que realmente estão começando a dar seus primeiros passos, mais que já tenham um conhecimento [**básico em C#**](https://www.devmedia.com.br/curso/minha-primeira-aplicacao-csharp/2173) e em SQL e SQL Server, como declarar variáveis e alguns comandos da sintaxe e executar [**instruções SQL**](https://www.devmedia.com.br/sql-select-guia-para-iniciantes/29530).

Será usado neste artigo o Visual Studio 2010, [**SQL Server**](https://www.devmedia.com.br/curso/curso-sql-server/406) e o SQL Server Management Studio.

Por ser destinados a iniciantes, algumas pessoas que já tem alguma experiência poderão notar que algumas partes do código poderiam ser feitas de forma diferente, utilizando melhores práticas e reaproveitamento de código, mas o objetivo deste artigo é ensinar a aquelas que nunca fizeram um **CRUD** ou que possuem dúvidas de como fazer, então foi feito de maneira bem simples para que todos possam entender.

O primeiro passo é criar o banco de dados que será utilizado. A **Listagem 1** contém o código SQL para a criação do mesmo.

/\* CRIAR BANCO DE DADOS \*/

CREATE DATABASE BDCADASTRO

/\* SELECIONA O BANCO DE DADOS \*/

USE BDCADASTRO

/\* CRIACÇÃO DA TABELA \*/

CREATE TABLE CLIENTE (

ID INT NOT NULL IDENTITY,

NOME VARCHAR(50) NOT NULL,

ENDERECO VARCHAR(50),

CEP VARCHAR(9),

BAIRRO VARCHAR(50),

CIDADE VARCHAR(50),

UF VARCHAR(2),

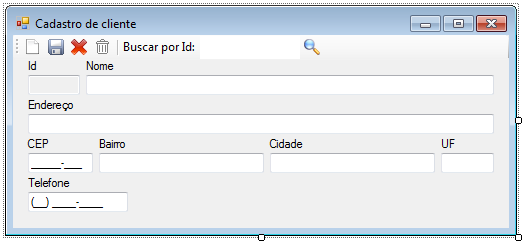
TELEFONE VARCHAR(15),

CONSTRAINT PK\_CLIENTE PRIMARY KEY(ID)

)

**Listagem 1**. Comandos SQL para criação do banco de dados e tabela

Após a criação do banco de dados, crie um formulário conforme a **Figura 1**.

**Figura 1.** Tela de cadastro

Abaixo segue a **Tabela 1** com todos os controles necessário para o formulário.

|  |  |
| --- | --- |
| Cadastro de cliente | Tipo: Form Name: frmCadastroCliente StartPosition: CenterScreen |
| Id | Tipo: TextBox Name: txtId ReadOnly: True |
| Nome | Tipo: TextBox Name: txtNome CharacterCasing: Upper MaxLength: 50 |
| Endereço | Tipo: TextBox Name: txtEndereco CharacterCasing: Upper MaxLength: 50 |
| CEP | Tipo: MaskedTextBox Name: mskCEP Mask: 00000-999 |
| Bairro | Tipo: TextBox Name: txtBairro CharacterCasing: Upper MaxLength: 50 |
| Cidade | Tipo: TextBox Name: txtCidade CharacterCasing: Upper MaxLength: 50 |
| UF | Tipo: TextBox Name: txtUf CharacterCasing: Upper MaxLength: 2 |
| Telefone | Tipo: MaskedTextBox Name: mskTelefone Mask: (99) 0000-0000 |
| Barra de botões | Tipo: ToolStrip Name: toolStrip1 |
| Novo | Tipo: ToolStripButton Name: tsbNovo Text: Novo |
| Salvar | Tipo: ToolStripButton Name: tsbSalvar Text: Salvar |
| Cancelar | Tipo: ToolStripButton Name: tsbCancelar Text: Cancelar |
| Excluir | Tipo: ToolStripButton Name: tsbExcluir Text: Excluir |
| Separador | Tipo: ToolStripSeparator Name: toolStripSeparator1 |
| Buscar por Id: | Tipo: ToolStripLabel Name: tsbBuscaPorId Text: Buscar por Id: |
|  | Tipo: ToolStripTextBox Name: tstId |
| Buscar | Tipo: ToolStripButton Name: tsbBuscar Text: Buscar |

**Tabela 1**. Controles do formulário

As imagens para os botões podem ser encontradas no site https://www.iconfinder.com, basta apenas procurar pelo nome.

Após tudo isso feito é hora de programar.

Dê dois cliques em uma área vazia do formulário, para que seja mostrado o arquivo frmCadastroCliente.cs. Nele será criado o método frmCadastroCliente\_Load. Logo acima do método frmCadastroCliente\_Load existe o construtor da classe que é o frmCadastroCliente e logo acima dele crie as seguintes variáveis, conforme a **Listagem 2**.

string connectionString = @"Server=.\sqlexpress;Database=bdcadastro;Trusted\_Connection=True;";

bool novo;

public frmCadastroCliente()

{

InitializeComponent();

}

private void frmCadastroCliente\_Load(object sender, EventArgs e)

{

}

**Listagem 2**. Criação das variáveis

A variável connectionString armazena a string de conexão com o banco de dados e a variável novo será usada para quando salvar saber se o que será salvo é um novo registro ou uma atualização de um registro existente.

Quando o programa for executado, alguns campos deverão aparecer desabilitados e só serão habilitados ao clicar no botão Novo, permitindo assim digitar as informações, e também alguns botões da barra também estarão inacessíveis e só poderão serem clicados dependendo da situação. Essas mudanças de estado ocorrerão a cada evento; O método frmCadastroCliente\_Load é executado ao carregar o formulário e parar ter a configuração inicial do formulário coloque o seguinte código no método, conforme mostra a **Listagem 3**.

private void frmCadastroCliente\_Load(object sender, EventArgs e)

{

tsbNovo.Enabled = true;

tsbSalvar.Enabled = false;

tsbCancelar.Enabled = false;

tsbExcluir.Enabled = false;

tstId.Enabled = true;

tsbBuscar.Enabled = true;

txtNome.Enabled = false;

txtEndereco.Enabled = false;

mskCep.Enabled = false;

txtBairro.Enabled = false;

txtCidade.Enabled = false;

txtUf.Enabled = false;

mskTelefone.Enabled = false;

}

**Listagem 3**. Configuração inicial do formulário

Agora a **Listagem 4** mostra o código do botão Novo.

private void tsbNovo\_Click(object sender, EventArgs e)

{

tsbNovo.Enabled = false;

tsbSalvar.Enabled = true;

tsbCancelar.Enabled = true;

tsbExcluir.Enabled = false;

tstId.Enabled = false;

tsbBuscar.Enabled = false;

txtNome.Enabled = true;

txtEndereco.Enabled = true;

mskCep.Enabled = true;

txtBairro.Enabled = true;

txtCidade.Enabled = true;

txtUf.Enabled = true;

mskTelefone.Enabled = true;

txtNome.Focus();

novo = true;

}

**Listagem 4**. Código do botão Novo

Pode-se notar que na última linha de código está sendo atribuído true para a variável novo. Mais adiante será possível entender melhor que isso significa que ao clicar no botão Novo será incluído um novo registro.

A **Listagem 5** contém com o código do botão Salvar.

private void tsbSalvar\_Click(object sender, EventArgs e)

{

if (novo)

{

string sql = "INSERT INTO CLIENTE

(NOME,ENDERECO,CEP,BAIRRO,CIDADE,UF,TELEFONE) "

+ "VALUES ('" + txtNome.Text + "', '" + txtEndereco.Text + "', '"

+ mskCep.Text + "', '" + txtBairro.Text

+ "', '" + txtCidade.Text + "', '" + txtUf.Text + "', '"

+ mskTelefone.Text + "')";

SqlConnection con = new SqlConnection(connectionString);

SqlCommand cmd = new SqlCommand(sql, con);

cmd.CommandType = CommandType.Text;

con.Open();

try

{

int i = cmd.ExecuteNonQuery();

if (i > 0)

MessageBox.Show("Cadastro realizado com sucesso!");

}

catch (Exception ex)

{

MessageBox.Show("Erro: " + ex.ToString());

}

finally

{

con.Close();

}

}

else

{

string sql = "UPDATE CLIENTE SET NOME='" + txtNome.Text + "',

ENDERECO='" + txtEndereco.Text + "', " +

"CEP='" + mskCep.Text + "', BAIRRO='" + txtBairro.Text + "',

CIDADE='" + txtCidade.Text + "', " +

"UF='" + txtUf.Text + "', TELEFONE='" + mskTelefone.Text + "' WHERE ID=" + txtId.Text";

SqlConnection con = new SqlConnection(connectionString);

SqlCommand cmd = new SqlCommand(sql, con);

cmd.CommandType = CommandType.Text;

con.Open();

try

{

int i = cmd.ExecuteNonQuery();

if (i > 0)

MessageBox.Show("Cadastro atualizado com sucesso!");

}

catch (Exception ex)

{

MessageBox.Show("Erro: " + ex.ToString());

}

finally

{

con.Close();

}

}

tsbNovo.Enabled = true;

tsbSalvar.Enabled = false;

tsbCancelar.Enabled = false;

tsbExcluir.Enabled = false;

tstId.Enabled = true;

tsbBuscar.Enabled = true;

txtNome.Enabled = false;

txtEndereco.Enabled = false;

mskCep.Enabled = false;

txtBairro.Enabled = false;

txtCidade.Enabled = false;

txtUf.Enabled = false;

mskTelefone.Enabled = false;

txtId.Text = "";

txtNome.Text = "";

txtEndereco.Text = "";

mskCep.Text = "";

txtBairro.Text = "";

txtCidade.Text = "";

txtUf.Text = "";

mskTelefone.Text = "";

}

**Listagem 5**. Código do botão salvar

Neste código há um if que testa o valor da variável novo para saber se é um registro novo e executa o código.

Quando novo=true, a variável SQL (que é uma string) recebe uma instrução SQL de INSERT que está sendo montada com os valores dos campos.

Logo após é criado um objeto com do tipo SqlConnection que é a conexão com o banco e que recebe como parâmetro no momento da instanciação a variável connectionString que foi criada anteriormente e que contém a string de conexão.

Depois é criado o objeto cmd que é um SqlCommand que recebe como parâmetro no seu construtor a variável SQL que contém a [**instrução SQL**](https://www.devmedia.com.br/10-instrucoes-sql-para-manipulacao-de-dados/4832) e o objeto con que é a conexão.

O objeto cmd será responsável por executar a instrução SQL.

Depois é definido o tipo de comando do objeto cmd, no caso Text, logo em seguida é aberta a conexão.

Dentro do bloco try é executada a instrução SQL do objeto cmd através do método ExecuteNonQuery que retorna o número de linha afetadas, que no caso é armazenada na variável inteira i.

Logo após testa-se se o valor de i é maior que 0, se sim significa que o registro foi incluído com sucesso.

Voltando um pouco, se quando o if testa a variável novo e ela for false então executa o código contido no bloco else que significa que está fazendo uma atualização de um registro.

O código é praticamente o mesmo, a não ser a instrução SQL que agora é um UPDATE.

Logo depois é alterado o estado dos controles do formulário novamente.

A **Listagem 6** contém o código do botão Cancelar, que só alterada o estado dos controles e limpa os TextBox’s.

private void tsbCancelar\_Click(object sender, EventArgs e)

{

tsbNovo.Enabled = true;

tsbSalvar.Enabled = false;

tsbCancelar.Enabled = false;

tsbExcluir.Enabled = false;

tstId.Enabled = true;

tsbBuscar.Enabled = true;

txtNome.Enabled = false;

txtEndereco.Enabled = false;

mskCep.Enabled = false;

txtBairro.Enabled = false;

txtCidade.Enabled = false;

txtUf.Enabled = false;

mskTelefone.Enabled = false;

txtId.Text = "";

txtNome.Text = "";

txtEndereco.Text = "";

mskCep.Text = "";

txtBairro.Text = "";

txtCidade.Text = "";

txtUf.Text = "";

mskTelefone.Text = "";

}

**Listagem 6**. Código do botão cancelar

A **Listagem 7** mostra o código do botão Excluir, que é bem parecido ao do botão Salvar, só que agora com a instrução de DELETE.

private void tsbExcluir\_Click(object sender, EventArgs e)

{

string sql = "DELETE FROM CLIENTE WHERE ID=" + txtId.Text;

SqlConnection con = new SqlConnection(connectionString);

SqlCommand cmd = new SqlCommand(sql, con);

cmd.CommandType = CommandType.Text;

con.Open();

try

{

int i = cmd.ExecuteNonQuery();

if (i > 0)

MessageBox.Show("Registro excluído com sucesso!");

}

catch (Exception ex)

{

MessageBox.Show("Erro: " + ex.ToString());

}

finally

{

con.Close();

}

tsbNovo.Enabled = true;

tsbSalvar.Enabled = false;

tsbCancelar.Enabled = false;

tsbExcluir.Enabled = false;

tstId.Enabled = true;

tsbBuscar.Enabled = true;

txtNome.Enabled = false;

txtEndereco.Enabled = false;

mskCep.Enabled = false;

txtBairro.Enabled = false;

txtCidade.Enabled = false;

txtUf.Enabled = false;

mskTelefone.Enabled = false;

txtId.Text = "";

txtNome.Text = "";

txtEndereco.Text = "";

mskCep.Text = "";

txtBairro.Text = "";

txtCidade.Text = "";

txtUf.Text = "";

mskTelefone.Text = "";

}

**Listagem 7**. Código do botão Excluir

A **Listagem 8** mostra o código do botão Buscar.

private void tsbBuscar\_Click(object sender, EventArgs e)

{

string sql = "SELECT \* FROM CLIENTE WHERE ID=" + tstId.Text;

SqlConnection con = new SqlConnection(connectionString);

SqlCommand cmd = new SqlCommand(sql, con);

cmd.CommandType = CommandType.Text;

SqlDataReader reader;

con.Open();

try

{

reader = cmd.ExecuteReader();

if (reader.Read())

{

tsbNovo.Enabled = false;

tsbSalvar.Enabled = true;

tsbCancelar.Enabled = true;

tsbExcluir.Enabled = true;

tstId.Enabled = false;

tsbBuscar.Enabled = false;

txtNome.Enabled = true;

txtEndereco.Enabled = true;

mskCep.Enabled = true;

txtBairro.Enabled = true;

txtCidade.Enabled = true;

txtUf.Enabled = true;

mskTelefone.Enabled = true;

txtNome.Focus();

txtId.Text = reader[0].ToString();

txtNome.Text = reader[1].ToString();

txtEndereco.Text = reader[2].ToString();

mskCep.Text = reader[3].ToString();

txtBairro.Text = reader[4].ToString();

txtCidade.Text = reader[5].ToString();

txtUf.Text = reader[6].ToString();

mskTelefone.Text = reader[7].ToString();

novo = false;

}

else

MessageBox.Show("Nenhum registro encontrado com o Id

informado!");

}

catch (Exception ex)

{

MessageBox.Show("Erro: " + ex.ToString());

}

finally

{

con.Close();

}

tstId.Text = "";

}

**Listagem 8**. Código do botão Buscar

No código do botão Buscar contém uma instrução de SELECT que retorna o registro com o Id informado.

Uma diferença é que além do objeto cmd agora também um objeto reader do tipo SqlDataReader que armazena o conteúdo retornado da consulta, como é mostrado na **Listagem 9**.

reader = cmd.ExecuteReader();

**Listagem 9**. Objeto reader recebendo o resultado da consulta executada pelo objeto cmd

Em seguida, o if testa se há registro no reader com o método Read. Se sim, executa-se o bloco.

O que há de diferente é que logo após alterar os estados dos controles o foco do cursor é posicionado no campo txtNome através do método Focus e é atribuído a cada campo o valor correspondente que está no reader.

Para acessar os campos no reader é utilizado o índice do campo. Esse índice é a ordem dos campos, para poder vê-los basta executar um SELECT no SQL Server Management Studio e ver o retorno, o primeiro campo inicia no índice 0.

Pronto, o CRUD está pronto, basta agora executar o programa e testar.

Como dito no início do artigo, existem maneiras melhores de fazer esse CRUD, mais o intuito foi fazê-lo da maneira mais simples para um fácil entendimento.