

## Sumário

PROJETO CRUD - Cidade & Estado .....	2
Sobre o Projeto.....	2
Explicando o Processo de Criação .....	2
Criando as Tabelas:.....	2
Criando as Telas:.....	3
Explicando o Processo de Criação (Código) .....	5
Variáveis utilizadas no projeto: .....	5
private void TestarConexao() .....	5
txtCidade, txtBoxEstado, txtBoxEstadoSigla .....	5
btnCadastrar() .....	6
bntListar() .....	7
bntDeletar() .....	7

# PROJETO CRUD - Cidade & Estado

## Sobre o Projeto

Foi feito para conseguir um Estágio ou CLT na empresa SISTEMAS BR que se reside em JALES-SP.

Procurei usar todos os meus conhecimentos de Programação Desktop (C#, *Windows Forms*, *Banco de Dados*, *UI/UX*) para criar um sistema de Cadastro de Cidade e Estado tendo uma relação com Banco de dados (ambas estão conectadas) e o principal **Criar, Listar e Deletar os dados**.

## Explicando o Processo de Criação

### Criando as Tabelas:

Para a criação das tabelas utilizei o banco de dados da Microsoft, **SQL SERVER MANAGEMENT STUDIO 2.0**, nele criei as tabelas Estado e Cidade, possuindo as seguintes colunas como podemos ver na imagem abaixo:

```
CREATE TABLE Cidade
(
    idcidade INT IDENTITY(1,1),
    nomecidade VARCHAR(50) NOT NULL,
    idestado INT NOT NULL,
    CONSTRAINT FK_estado_cidade FOREIGN KEY (idestado)
    REFERENCES Estado(idestado) ON DELETE CASCADE
);

CREATE TABLE Estado
(
    idestado INT IDENTITY(1,1) PRIMARY KEY,
    nomeestado VARCHAR(50) NOT NULL,
    sigla VARCHAR(2) NOT NULL
);
```

Imagem das tabelas que foram criadas

Testei inserindo alguns valores no mesmo, para testar se estava realmente interligas e funcionando perfeitamente:

```
-- Estado
INSERT INTO Estado(nomeestado, sigla) VALUES ('Sao Paulo', 'SP');
INSERT INTO Estado(nomeestado, sigla) VALUES ('Rio de Janeiro', 'RJ');

-- Cidade
INSERT INTO Cidade(nomecidade, idestado) VALUES ('Sao Paulo', 1);
INSERT INTO Cidade(nomecidade, idestado) VALUES ('Rio de Janeiro', 2);
INSERT INTO Cidade(nomecidade, idestado) VALUES ('Jales', 1);
INSERT INTO Cidade(nomecidade, idestado) VALUES ('Rio Preto', 1);
```

## Criando as Telas:

No meu caso resolvi fazer em uma única tela, mas contendo tudo que foi relatado pela empresa SISTEMAS BR.

A primeira coisa que fiz foi criar o visual da aplicação, fundo cinza + uma PictureBox vazio com a cor branca (White) para dar um contraste e realçar o que é importante na aplicação.

A segunda parte foi criar as labels (que irá descrever o que os TextBoxs irão fazer), aproveitei e coloquei a logo da SISTEMAS BR no lado superior esquerdo para dar identidade ao projeto que estou realizando, como podemos ver abaixo:



### ***Cidade - Estado***

Cadastre uma Cidade:

Cadastre um Estado:

Sigla:

Após eu ter feito isso inserir os botões que será as ações que o usuário poderá fazer no programa desktop, elas são “Cadastrar, Listar, Deletar”, veja a imagem abaixo ilustrando isso:



### ***Cidade - Estado***


Cadastre uma Cidade:

Cadastre um Estado:

Sigla:

Cadastrar	Listar	Deletar
-----------	--------	---------

Agora entra a parte do elemento que ira exibir as tabelas o famoso dataGridView, inseri ele logo abaixo no centro do programa, que podemos ver abaixo:



## Cidade - Estado

Cadastre uma Cidade:

Cadastre um Estado:

Sigla:

O ícone do programa eu peguei de um site gratuitos de ícones, o nome do title text eu coloquei “Cadastro de Cidade & Estado”, como podemos ver logo abaixo (olhe para lado superior canto esquerdo da aplicação):

Cadastro de Cidade & Estado



## Cidade - Estado

Cadastre uma Cidade:

Cadastre um Estado:

Sigla:

## Explicando o Processo de Criação (Código)

### Variáveis utilizadas no projeto:

Criei as seguintes variáveis no projeto, sendo a principal a string de conexão do banco de dados, veja abaixo:

```
private string connectionString = @"Server=DESKTOP-60P28C8\SQLEXPRESS;Database=Cadastro_Cidade_Estado;Integrated Security=True;";  
private string nomecidade;  
private string nomeestado;  
private string estadosigla;
```

Todos foram criados fora da classe de elementos, fazendo assim variáveis globais

### private void TestarConexao()

Aqui a primeira classe do programa, testando a conexão do banco de dados/abrindo o mesmo, resolvi fazer desse jeito para ficar melhor na leitura e manutenção do código depois (por isso é o primeiro que aparece, a instancia **public Cidade()** nela está chamando a classe)

```
1 referência  
public Cidade()  
{  
    InitializeComponent();  
    TestarConexao();  
}  
  
1 referência  
private void TestarConexao()  
{  
    using (SqlConnection connection = new SqlConnection(connectionString))  
    {  
        connection.Open();  
        // MessageBox.Show("O banco está conectado", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information);  
    }  
}
```

### txtCidade, txtBoxEstado, txtBoxEstadoSigla

Essa parte são os TextBox citados na parte "[Criando as telas](#)" aqui usarei as variáveis usadas para guardar as informações digitadas pelo usuário para cadastrar e deletar:

```
1 referência  
private void txtCidade_TextChanged(object sender, EventArgs e)  
{  
    nomecidade = txtCidade.Text;  
}  
  
1 referência  
private void txtBoxEstado_TextChanged(object sender, EventArgs e)  
{  
    nomeestado = txtBoxEstado.Text;  
}  
  
1 referência  
private void txtBoxEstadoSigla_TextChanged(object sender, EventArgs e)  
{  
    estadosigla = txtBoxEstadoSigla.Text.ToUpper();  
}
```

Coloquei um tratamento de string no **txtBoxEstadoSigla** usando o método **.ToUpper()** para que qualquer coisa que o usuário escreva seja convertida para letras maiúsculas (por ser SIGLAS de ESTADOS isso é extremamente importante)

## btnCadastrar()

Aqui já começamos a entrar nas partes das funções dos botões, sendo o primeiro de cadastrar o que foi inserido pelo usuário, fiz tratamento caso o usuário tente inserir nada no banco de dados, além de exibir em uma MessageBox se a Cidade e Estado foram cadastrados com sucesso ou não:

```
1 referência
private void btnCadastrar_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrEmpty(nomecidade) && !string.IsNullOrEmpty(nomeestado) && !string.IsNullOrEmpty(estadosigla))
    {
        try
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                SqlTransaction transaction = connection.BeginTransaction();

                try
                {
                    int idestado;

                    // Verifica se o estado já existe no banco
                    string queryCheckEstado = "SELECT idestado FROM Estado WHERE nomeestado = @nomeestado AND sigla = @estadosigla";
                    using (SqlCommand commandCheckEstado = new SqlCommand(queryCheckEstado, connection, transaction))
                    {
                        commandCheckEstado.Parameters.AddWithValue("@nomeestado", nomeestado);
                        commandCheckEstado.Parameters.AddWithValue("@estadosigla", estadosigla);
                        var result = commandCheckEstado.ExecuteScalar();

                        if (result != null)
                        {
                            idestado = Convert.ToInt32(result); // Estado já existe, pega o ID
                        }
                        else
                        {
                            // Insere o estado e recupera o ID gerado
                            string queryInsertEstado = "INSERT INTO Estado (nomeestado, sigla) VALUES (@nomeestado, @estadosigla); SELECT SCOPE_IDENTITY()";
                            using (SqlCommand commandInsertEstado = new SqlCommand(queryInsertEstado, connection, transaction))
                            {
                                commandInsertEstado.Parameters.AddWithValue("@nomeestado", nomeestado);
                                commandInsertEstado.Parameters.AddWithValue("@estadosigla", estadosigla);
                                idestado = Convert.ToInt32(commandInsertEstado.ExecuteScalar());
                            }
                        }
                    }

                    // Insere a cidade associando ao idestado
                    string queryInsertCidade = "INSERT INTO Cidade (nomecidade, idestado) VALUES (@nomecidade, @idestado)";
                    using (SqlCommand commandInsertCidade = new SqlCommand(queryInsertCidade, connection, transaction))
                    {
                        commandInsertCidade.Parameters.AddWithValue("@nomecidade", nomecidade);
                        commandInsertCidade.Parameters.AddWithValue("@idestado", idestado);
                        commandInsertCidade.ExecuteNonQuery();
                    }

                    transaction.Commit();
                    MessageBox.Show("Cidade & Estado cadastrada com sucesso!", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    // Limpa os campos após o cadastro
                    txtCidade.Text = "";
                    txtBoxEstado.Text = "";
                    txtBoxEstadoSigla.Text = "";

                }
                catch (Exception ex)
                {
                    transaction.Rollback();
                    MessageBox.Show("Erro ao cadastrar cidade: " + ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Erro de conexão com o banco de dados: " + ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    else
    {
        MessageBox.Show("Preencha todos os campos antes de cadastrar!", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

## bntListar()

Aqui entra o botão para listar toda alteração do banco de dados, irá exibir nome da cidade, nome do estado e principalmente a sigla do mesmo:

```
private void bntListar_Click(object sender, EventArgs e)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string query = @"
SELECT C.nomecidade, C.idestado, E.nomeestado, E.sigla
FROM Cidade C
INNER JOIN Estado E ON C.idestado = E.idestado;";

            using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))
            {
                DataTable dataTable = new DataTable();
                adapter.Fill(dataTable);
                dataGridViewCidade.DataSource = dataTable;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro ao listar cidades: " + ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

## bntDeletar()

Aqui temos o botão que pode fazer duas ações quando clicar no mesmo:

1. O primeiro clique no botão os labels que estavam escritos **“Cadastre uma Cidade:”** **“Cadastre um Estado:”** iram mudar para **“Deletar Cidade”** **“Deletar Estado”**, logo depois pega as variáveis criadas e atribui novamente os valores digitados pelo usuário:

```
1 referência
private void btnDeletar_Click(object sender, EventArgs e)
{
    if (lblCidade.Text != "Deletar Cidade" && lblEstado.Text != "Deletar Estado")
    {
        // Primeira vez que clica: altera os labels e armazena os valores dos TextBox
        lblCidade.Text = "Deletar Cidade";
        lblEstado.Text = "Deletar Estado";

        nomecidade = txtCidade.Text;
        nomeestado = txtBoxEstado.Text;
        estadosigla = txtBoxEstadoSigla.Text;
    }
}
```

2. O segundo clique já faz ação que o botão foi feito, deletar o que já esta escrito no banco de Dados (para isso precisa clicar no botão listar para ver o que já esta no banco de dados):

```
// Segunda vez que clica: deleta do banco de dados
if (!string.IsNullOrWhiteSpace(nomecidade) && !string.IsNullOrWhiteSpace(nomeestado) && !string.IsNullOrWhiteSpace(estadosigla))
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            SqlTransaction transaction = connection.BeginTransaction();

            try
            {
                // Deleta TODAS as cidades associadas ao estado
                string queryDeleteCidades = "DELETE FROM Cidade WHERE idestado IN (SELECT idestado FROM Estado WHERE nomeestado = @nomeestado AND sigla = @estadosigla)";
                using (SqlCommand cmdDeleteCidades = new SqlCommand(queryDeleteCidades, connection, transaction))
                {
                    cmdDeleteCidades.Parameters.AddWithValue("@nomeestado", nomeestado);
                    cmdDeleteCidades.Parameters.AddWithValue("@estadosigla", estadosigla);
                    cmdDeleteCidades.ExecuteNonQuery();
                }

                // Agora que não há cidades, deleta o estado
                string queryDeleteEstado = "DELETE FROM Estado WHERE nomeestado = @nomeestado AND sigla = @estadosigla";
                using (SqlCommand cmdDeleteEstado = new SqlCommand(queryDeleteEstado, connection, transaction))
                {
                    cmdDeleteEstado.Parameters.AddWithValue("@nomeestado", nomeestado);
                    cmdDeleteEstado.Parameters.AddWithValue("@estadosigla", estadosigla);
                    cmdDeleteEstado.ExecuteNonQuery();
                }

                transaction.Commit();
                MessageBox.Show("Cidade(s) e Estado deletados com sucesso!", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch { transaction.Rollback(); }
        }
    }
}
```

```
        catch (Exception ex)
        {
            transaction.Rollback();
            MessageBox.Show("Erro ao deletar: " + ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    catch (Exception ex)
    {
        MessageBox.Show("Erro de conexão com o banco de dados: " + ex.Message, "Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    // Limpa os labels e variáveis
    lblCidade.Text = "Cadastre uma Cidade";
    lblEstado.Text = "Cadastre um Estado";
    nomecidade = "";
    nomeestado = "";
    estadosigla = "";
    txtCidade.Clear();
    txtBoxEstado.Clear();
    txtBoxEstadoSigla.Clear();
}

else
{
    MessageBox.Show("Nenhum dado foi armazenado para deletar!", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}
```