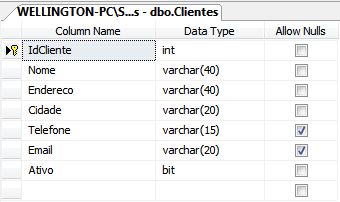
**Criação do Banco e do Projeto: CRUD Básico em VB.NET – Parte 1**

Olá pessoal, nesta série de 3 artigos veremos como fazer um **CRUD** Básico em **Windows Forms** usando a linguagem **VB.NET** (quem disse que este blog não tem artigos sobre VB.NET?) e o banco de dados **SQL Server**. Acompanhem:

**Introdução –** Este artigo é feito já considerando que o leitor esteja habituado com o básico dos conceitos do **ADO.NET**. Se você não se enquadra neste perfil, pesquise no próprio blog, que tem diversos artigos publicados sobre o assunto.

Antes de qualquer coisa vamos criar a tabela **Clientes** no SQL Server, veja a estrutura da tabela abaixo, na **Figura 01**:

[](https://programandodotnet.wordpress.com/2011/08/15/crud-basico-vb-net-parte-1/clientestable-3/)

*Figura 01 – Tabela Clientes*

 Segue abaixo o script para criação da tabela:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | CREATE TABLE Clientes    (    IdCliente   INT IDENTITY(1,1) NOT NULL    , Nome        VARCHAR(40)       NOT NULL    , Endereco    VARCHAR(40)       NOT NULL    , Cidade      VARCHAR(20)       NOT NULL    , Telefone    VARCHAR(15)           NULL    , Email       VARCHAR(20)           NULL    , Ativo       BIT               NOT NULL    , CONSTRAINT PK\_IdCliente PRIMARY KEY (IdCliente)    ) |

Tabela criada, crie um projeto em **Windows Form** (lembrando que estou usando a linguagem **VB.NET** para este artigo) e adicione a ele uma classe, dando a ela o nome de **DadosClientes**, essa classe irá conter toda a lógica do projeto.

Sem mais delongas, comece declarando nesta classe o namespace **System.Data.SqlClient**:

|  |  |
| --- | --- |
| 1 | Imports System.Data.SqlClient |

Agora dentro de nossa classe iremos criar outra classe, responsável por conter as variáveis que representarão as colunas de nosso banco de dados (se o projeto fosse em camadas poderíamos dizer que esta classe seria a camada de **Modelo** dos dados). Veja a classe:

Public Class ClientesDTO

Private \_idCliente As Integer

Private \_nome As String

Private \_endereco As String

Private \_cidade As String

Private \_telefone As String

Private \_email As String

Private \_ativo As Boolean

Public Property IdCliente As Integer

Get

Return \_idCliente

End Get

Set(value As Integer)

\_idCliente = value

End Set

End Property

Public Property Nome As String

Get

Return \_nome

End Get

Set(value As String)

\_nome = value

End Set

End Property

Public Property Endereco As String

Get

Return \_endereco

End Get

Set(value As String)

\_endereco = value

End Set

End Property

Public Property Cidade As String

Get

Return \_cidade

End Get

Set(value As String)

\_cidade = value

End Set

End Property

Public Property Telefone As String

Get

Return \_telefone

End Get

Set(value As String)

\_telefone = value

End Set

End Property

Public Property Email As String

Get

Return \_email

End Get

Set(value As String)

\_email = value

End Set

End Property

Public Property Ativo As Boolean

Get

Return \_ativo

End Get

Set(value As Boolean)

\_ativo = value

End Set

End Property

End Class

Assim, ao chamar os métodos não preciso ficar passando diversos parâmetros, apenas passo esta classe.

Dando continuidade crie algumas variáveis globais, que iremos usar nos métodos:

|  |  |
| --- | --- |
| Imports System.Data.SqlClient  'Variáveis globais para todo o projeto  Module GlobalDAL  Public Const strConexao As String = "Data Source=NOTE;Initial Catalog=CRUD;Integrated Security=true"  Public strInstrucao As String = String.Empty  Public objConexao As New SqlConnection(strConexao)  Public objCommand As New SqlCommand(strInstrucao, objConexao)  End Module |  |

Note que, fazendo desta forma, não precisamos declará-las em cada método que precisarmos usá-las. As variáveis são referentes aos objetos das classes do **ADO.NET**, a **string de conexão** com o banco, além de uma variável que irá receber a **instrução SQL** (dependendo do método), todas essenciais para nosso projeto.

Assim finalizo o artigo. Na próxima parte começaremos a codificar os métodos relativos ao **CRUD**, não percam!

**Criação dos Métodos da Classe: CRUD Básico em VB.NET – Parte 2**

Olá pessoal, continuando nosso artigo com o **CRUD** Básico com a linguagem **VB.NET** vamos criar os principais métodos em nossa classe, para que na próxima parte apenas adicionemos os detalhes finais referentes aos formulários do projeto. Acompanhem:

Comece criando o método que irá **gravar** o cliente na base de dados:

Public Function GravarCliente(ByVal clientes As ClientesDTO) As Integer

Try

strInstrucao = "INSERT INTO Clientes(Nome,Endereco,Cidade,Telefone,Email,Ativo) " &

" VALUES (@Nome,@Endereco,@Cidade,@Telefone,@Email,@Ativo)"

objCommand.CommandText = strInstrucao

objCommand.Connection = objConexao

If objCommand.Parameters.Contains("@Nome") = False Then

objCommand.Parameters.AddWithValue("@Nome", clientes.IdCliente)

Else

objCommand.Parameters.Item(1).Value = clientes.Nome

End If

If objCommand.Parameters.Contains("@Endereco") = False Then

objCommand.Parameters.AddWithValue("@Endereco", clientes.Endereco)

Else

objCommand.Parameters.Item(2).Value = clientes.Endereco

End If

If objCommand.Parameters.Contains("@Cidade") = False Then

objCommand.Parameters.AddWithValue("@Cidade", clientes.Cidade)

Else

objCommand.Parameters.Item(3).Value = clientes.Cidade

End If

If objCommand.Parameters.Contains("@Telefone") = False Then

objCommand.Parameters.AddWithValue("@Telefone", clientes.Telefone)

Else

objCommand.Parameters.Item(4).Value = clientes.Telefone

End If

If objCommand.Parameters.Contains("@Email") = False Then

objCommand.Parameters.AddWithValue("@Email", clientes.Email)

Else

objCommand.Parameters.Item(5).Value = clientes.Email

End If

If objCommand.Parameters.Contains("@Ativo") = False Then

objCommand.Parameters.AddWithValue("@Ativo", clientes.Ativo)

Else

objCommand.Parameters.Item(6).Value = clientes.Ativo

End If

If objConexao.State = ConnectionState.Closed Then

objConexao.Open()

End If

Return objCommand.ExecuteNonQuery()

Catch ex As Exception

Throw New Exception(ex.Message)

Finally

objConexao.Close()

End Try

End Function

O método para atualizar o cliente será praticamente o mesmo, apenas com a adição do **IdCliente**, que será nosso parâmetro para o **Where** de nosso **Update** (lembrando que **nunca deveremos criar um UPDATE ou DELETE sem WHERE!**)

Public Function AtualizarCliente(ByVal clientes As ClientesDTO) As Integer

Try

strInstrucao = "UPDATE Clientes SET Nome = @Nome," &

"Endereco = @Endereco," &

"Cidade = @Cidade," &

"Telefone = @Telefone," &

"Email = @Email," &

"Ativo = @Ativo " &

"WHERE IdCliente = @IdCliente"

objCommand.CommandText = strInstrucao

objCommand.Connection = objConexao

If objCommand.Parameters.Contains("@IdCliente") = False Then

objCommand.Parameters.AddWithValue("@IdCliente", clientes.IdCliente)

Else

objCommand.Parameters.Item(0).Value = clientes.IdCliente

End If

If objCommand.Parameters.Contains("@Nome") = False Then

objCommand.Parameters.AddWithValue("@Nome", clientes.Nome)

Else

objCommand.Parameters.Item(1).Value = clientes.Nome

End If

If objCommand.Parameters.Contains("@Endereco") = False Then

objCommand.Parameters.AddWithValue("@Endereco", clientes.Endereco)

Else

objCommand.Parameters.Item(2).Value = clientes.Endereco

End If

If objCommand.Parameters.Contains("@Cidade") = False Then

objCommand.Parameters.AddWithValue("@Cidade", clientes.Cidade)

Else

objCommand.Parameters.Item(3).Value = clientes.Cidade

End If

If objCommand.Parameters.Contains("@Telefone") = False Then

objCommand.Parameters.AddWithValue("@Telefone", clientes.Telefone)

Else

objCommand.Parameters.Item(4).Value = clientes.Telefone

End If

If objCommand.Parameters.Contains("@Email") = False Then

objCommand.Parameters.AddWithValue("@Email", clientes.Email)

Else

objCommand.Parameters.Item(5).Value = clientes.Email

End If

If objCommand.Parameters.Contains("@Ativo") = False Then

objCommand.Parameters.AddWithValue("@Ativo", clientes.Ativo)

Else

objCommand.Parameters.Item(6).Value = clientes.Ativo

End If

If objConexao.State = ConnectionState.Closed Then

objConexao.Open()

End If

Return objCommand.ExecuteNonQuery()

Catch ex As Exception

Throw New Exception(ex.Message)

Finally

objConexao.Close()

End Try

End Function

Dito isto, não irei exibir aqui o método para **atualizar** o cliente, já que o projeto completo será disponibilizado ao final da **parte 3** do artigo.

Veja abaixo o método para **excluir** o cliente da base de dados:

Public Function ExcluirCliente(ByVal idCliente As Integer) As Integer

Try

strInstrucao = "DELETE FROM Clientes WHERE IdCliente = @IdCliente"

objCommand.CommandText = strInstrucao

objCommand.Connection = objConexao

If objCommand.Parameters.Contains("@IdCliente") = False Then

objCommand.Parameters.AddWithValue("@IdCliente", idCliente)

Else

objCommand.Parameters.Item(0).Value = idCliente

End If

If (objConexao.State = ConnectionState.Closed) Then

objConexao.Open()

End If

Return objCommand.ExecuteNonQuery()

Catch ex As Exception

Throw New Exception(ex.Message)

Finally

objConexao.Close()

End Try

End Function

Bem simples né? Seguindo a mesma lógica dos outros.

Finalizando, veja o método para **consultar** os clientes, um pouco diferentes dos outros:

Public Function ConsultarClientes() As DataTable

Dim dt As New DataTable

Dim ds As New DataSet

Try

strInstrucao = "SELECT IdCliente,Nome,Endereco,Cidade,Telefone,Email,Ativo FROM Clientes"

objCommand.CommandText = strInstrucao

objCommand.Connection = objConexao

If (objConexao.State = ConnectionState.Closed) Then

objConexao.Open()

End If

Dim da As New SqlDataAdapter(objCommand)

da.Fill(ds)

dt = ds.Tables(0)

Return dt

Catch ex As Exception

Throw New Exception(ex.Message)

Finally

objConexao.Close()

End Try

End Function

Public Function ConsultarPorCriterio(ByVal criterio As String, ByVal pesquisa As String) As DataTable

Try

Dim dt As New DataTable

Dim ds As New DataSet

objCommand.Connection = objConexao

objCommand.CommandType = CommandType.Text

If (criterio = "ID") Then

strInstrucao = "SELECT \* FROM Clientes WHERE IdCliente = @IdCliente"

If (objCommand.Parameters.Contains("@IdCliente") = False) Then

objCommand.Parameters.AddWithValue("@IdCliente", Convert.ToInt32(pesquisa))

Else

objCommand.Parameters.Item(0).Value = pesquisa

End If

ElseIf (criterio = "Nome") Then

strInstrucao = "SELECT \* FROM Clientes WHERE Nome LIKE '%" + pesquisa + "%'"

End If

objCommand.CommandText = strInstrucao

If (objConexao.State = ConnectionState.Closed) Then

objConexao.Open()

End If

Dim da As New SqlDataAdapter(objCommand)

da.Fill(ds)

dt = ds.Tables(0)

Return dt

Catch ex As Exception

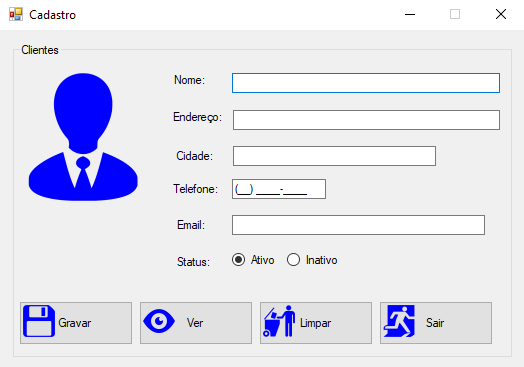
Throw New Exception(ex.Message)

End Try

End Function

Note que aqui usei **DataTable**, **DataSet** e **DataAdapter**, conceitos muito importantes do **ADO.NET**. Poderia até usar o **SqlDataReader**, mais para este caso específico preferi não usá-lo.

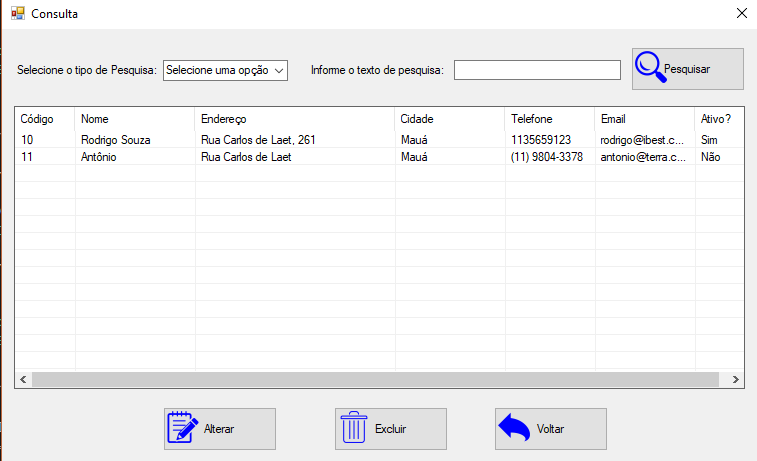
Crie um formulário e dê a ele o nome de **Cadastro**. Nele, adicione alguns controles para que o mesmo fique como a **Figura 01** nos mostra:



*Figura 01 – Formulário de Cadastro*

 Bem simples. Esse botão **Ver** nos levará para a tela de **Consulta**, que criaremos logo abaixo.

Apenas altere a propriedade **Modifiers** dos **TextBoxes** e dos **RadioButtons** para **Public**. Você entenderá o porque disso na próxima parte do artigo.No form de **Consulta**, adicione um ListView e três Buttons. No ListView configure as propriedades **GridLines** para True, **FullRowSelect** para True, **MultiSelect** para False e **View** para Details. Adicione também nele 5 colunas. Veja como o formulário ficará na **Figura 02**:



*Figura 02 – Formulário de Consulta*

 Finalizando, vá à página de códigos do form de Cadastro e declare uma variável global (que será “enxergada” em todo o form) do tipo **Integer**:

Public idCliente As Integer = 0

Ela irá receber o **Id** do Cliente do form de **Consulta**, para assim definirmos quando for Alteração ou Gravação de dados.

Assim finalizo o artigo. Na próxima e última parte faremos a codificação nos forms criados e os testes práticos, não percam!

**Codificação dos Formulários e Resultado Final: CRUD Básico em VB.NET – Parte 3**

Olá pessoal, continuando nosso artigo com o **CRUD** Básico em **Windows Forms** com a linguagem **VB.NET** vamos criar a codificação para os formulários do projeto e testarmos o funcionamento dos métodos de nossa classe. Acompanhem:

Como de costume em artigos sobre ADO.NET, a parte 3 ficou enorme hehe.

Camada de Negócios

Imports DAL

Imports DTO

'Camada de Negócios (BLL)

Public Class ClientesBLL

Public Function GravarCliente(ByVal cliente As ClientesDTO) As Integer

Dim db As New ClientesDAL

Dim cod As Integer

cod = db.GravarCliente(cliente)

Return cod

End Function

Public Function AtualizarCliente(ByVal cliente As ClientesDTO) As Integer

Dim db As New ClientesDAL

Dim cod As Integer

cod = db.AtualizarCliente(cliente)

Return cod

End Function

Public Function ExcluirCliente(ByVal idcliente As Integer) As Integer

Dim db As New ClientesDAL

Dim cod As Integer

cod = db.ExcluirCliente(idcliente)

Return cod

End Function

Public Function ConsultarCliente() As DataTable

Dim db As New ClientesDAL

Dim dt As DataTable

dt = db.ConsultarClientes()

Return dt

End Function

Public Function ConsultarPorCriterio(ByVal criterio As String, ByVal pesquisa As String) As DataTable

Try

Dim db As New ClientesDAL

Dim dt As DataTable

dt = db.ConsultarPorCriterio(criterio, pesquisa)

Return dt

Catch ex As Exception

Throw New Exception(ex.Message)

End Try

End Function

Public Function ValidarCampos(ByVal cliente As ClientesDTO) As Boolean

Dim blnResult = False

If (cliente.Nome.Equals(String.Empty)) Then

blnResult = False

ElseIf (cliente.Endereco.Equals(String.Empty)) Then

blnResult = False

ElseIf (cliente.Cidade.Equals(String.Empty)) Then

blnResult = False

ElseIf (cliente.Email.Equals(String.Empty)) Then

blnResult = False

ElseIf (cliente.Ativo.Equals(String.Empty)) Then

blnResult = False

ElseIf (cliente.Telefone.Equals("( ) -")) Then

blnResult = False

Else

blnResult = True

End If

Return blnResult

End Function

End Class

Voltando de onde paramos, veja abaixo a codificação do método que será chamado no botão **Gravar** do form **Cadastro**(não perderei muito tempo com validações já que o intuito do artigo não é este, apesar delas terem sido feitas no projeto):

Private Sub Gravar()

Dim classe As New ClientesBLL

Dim objeto As New ClientesDTO

objeto.Nome = txtNome.Text.Trim()

objeto.Endereco = txtEndereco.Text.Trim()

objeto.Cidade = txtCidade.Text.Trim()

objeto.Email = txtEmail.Text.Trim()

objeto.Telefone = mskTelefone.Text.Trim()

If rbtAtivo.Checked Then

objeto.Ativo = True

Else

objeto.Ativo = False

End If

If (classe.ValidarCampos(objeto) = False) Then

MessageBox.Show("Verifique se existe algum dado não informado", "Advertindo", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

Exit Sub

End If

If (idCliente > 0) Then

objeto.IdCliente = idCliente

If (classe.AtualizarCliente(objeto) > 0) Then

MessageBox.Show("Cliente alterado com sucesso", "Cliente", MessageBoxButtons.OK, MessageBoxIcon.Information)

End If

Else

If (classe.GravarCliente(objeto) > 0) Then

MessageBox.Show("Cliente inserido com sucesso", "Cliente", MessageBoxButtons.OK, MessageBoxIcon.Information)

txtNome.Text = ""

txtEndereco.Text = ""

txtEmail.Text = ""

txtCidade.Text = ""

mskTelefone.Text = ""

rbtAtivo.Checked = True

rbtInativo.Checked = False

idCliente = 0

End If

End If

End Sub

Note que usando a classe tudo fica mais fácil e nosso código não fica muito “poluído”. O que fiz acima foi instanciar a classe em uma variável e instanciar a classe criada dentro de nossa classe principal em outra variável. Depois atribuí a cada objeto de minha classe **ClientesObj** o valor nos campos, seguido do método **Trim()**, que remove os espaços em branco. Ao final verifico o valor da variável **IdCliente**. Se for maior que **0** chamo o método **AtualizarCliente**,passando o Id pro objeto. Senão chamo o **GravarCliente** passando o objeto carregado com os valores respectivos.

Agora vá ao form de **Consulta** e declare uma variável pública que irá receber o índice da linha do ListView:

'Recebe o indice da listview

Public index As Integer

Para que façamos isso devemos usar o evento **Click** do ListView. Chame-o e faça a seguinte codificação:

Private Sub lsvDados\_Click(sender As Object, e As EventArgs) Handles lsvDados.Click

If (lsvDados.SelectedItems.Count > 0) Then

'Coleta o item selecionado

index = lsvDados.SelectedItems(0).Index

End If

End Sub

Agora veja o método que irá popular nosso ListView com os dados do banco:

Private Sub CarregarListView()

Dim objDados As New ClientesBLL

Dim dt As New DataTable

dt = objDados.ConsultarCliente()

For Each linha As DataRow In dt.Rows

Dim lista As New ListViewItem

lista.Text = linha("IdCliente").ToString()

lista.SubItems.Add(linha("Nome").ToString())

lista.SubItems.Add(linha("Endereco").ToString())

lista.SubItems.Add(linha("Cidade").ToString())

lista.SubItems.Add(linha("Telefone").ToString())

lista.SubItems.Add(linha("Email").ToString())

If linha("Ativo").ToString().Equals("True") Then

lista.SubItems.Add("Sim")

Else

lista.SubItems.Add("Não")

End If

lsvDados.Items.Add(lista)

Next

End Sub

Apenas instanciei um **DataTable**, que recebeu meu método que é do mesmo tipo, fiz um laço **foreach**, para pesquisar em cada linha do **DataTable**, dentro dele instanciei um **ListViewItem** e fui adicionando nele os valores do **DataRow**. Ao final jogo em meu ListView os itens adicionados a variável do ListViewItem.

Você pode chamar este método no evento **Load** do formulário, só dar 2 cliques no form para que o evento seja chamado.

Crie o método **Excluir**, você precisará de apenas 2 linhas de código:

Private Function ExcluirCliente(ByVal idcliente As Integer) As Integer

Dim objDados As New ClientesBLL

Return objDados.ExcluirCliente(idcliente)

End Function

Como o método é do tipo **Integer**, você deve chama-lo passando o Id do Cliente que você deseja excluir. Para isso codifique o evento **click** do botão **Excluir**:

Private Sub btnExcluir\_Click(sender As Object, e As EventArgs) Handles btnExcluir.Click

If lsvDados.SelectedItems.Count > 0 Then

If (MessageBox.Show("Deseja excluir este cliente?", "Excluir", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2) = DialogResult.Yes) Then

If (ExcluirCliente(Integer.Parse(lsvDados.Items(index).Text)) > 0) Then

MessageBox.Show("Cliente excluído com sucesso!!", "Exclusão", MessageBoxButtons.OK, MessageBoxIcon.Information)

lsvDados.Items.Clear()

CarregarListView()

End If

End If

Else

MessageBox.Show("Favor para excluir selecione um cliente específico", "Advertindo", MessageBoxButtons.OK, MessageBoxIcon.Warning)

End If

End Sub

Note que eu passo o valor da variável **index**, que é a coluna **IdCliente**.

Finalizando, vamos ao método **Alterar**. Para ficar tudo mais fácil, se o usuário clicar em **Alterar** ele volta pra tela de Cadastro com os dados preenchidos e clica em **Gravar**. Lembra que nossos campos da tela de Cadastro tem a propriedade **Modifiers** como Public? Então, é agora que entenderemos o porquê disso.

Veja a codificação no evento click do botão **Alterar**:

Private Sub btnAlterar\_Click(sender As Object, e As EventArgs) Handles btnAlterar.Click

If lsvDados.SelectedItems.Count > 0 Then

'Dim instancia As New FrmCadastro()

FrmCadastro.idCliente = Integer.Parse(lsvDados.Items(index).Text)

FrmCadastro.txtNome.Text = lsvDados.Items(index).SubItems(1).Text

FrmCadastro.txtEndereco.Text = lsvDados.Items(index).SubItems(2).Text

FrmCadastro.txtCidade.Text = lsvDados.Items(index).SubItems(3).Text

FrmCadastro.mskTelefone.Text = lsvDados.Items(index).SubItems(4).Text

FrmCadastro.txtEmail.Text = lsvDados.Items(index).SubItems(5).Text

If (lsvDados.Items(index).SubItems(6).Text.Equals("Sim")) Then

FrmCadastro.rbtAtivo.Checked = True

Else

FrmCadastro.rbtInativo.Checked = True

End If

Me.Close()

Else

MessageBox.Show("Para alterar selecione um cliente específico", "Advertindo", MessageBoxButtons.OK, MessageBoxIcon.Warning)

End If

End Sub

Botão **Pesquisar** implementação:

Private Sub btnPesquisar\_Click(sender As Object, e As EventArgs) Handles btnPesquisar.Click

Dim clienteBLL As New ClientesBLL

Dim dt As New DataTable

If (cmbTipoPesquisa.SelectedIndex = -1) Then

MessageBox.Show("Selecione um critério específico para a pesquisa", "Advertindo", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

Exit Sub

ElseIf (txtPesquisa.Text = String.Empty) Then

MessageBox.Show("Digite uma pesquisa específica", "Advertindo", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

Exit Sub

ElseIf (cmbTipoPesquisa.SelectedItem.Equals("ID")) Then

dt = clienteBLL.ConsultarPorCriterio("ID", txtPesquisa.Text)

ElseIf (cmbTipoPesquisa.SelectedItem.Equals("Nome")) Then

dt = clienteBLL.ConsultarPorCriterio("Nome", txtPesquisa.Text)

End If

lsvDados.Items.Clear()

For Each linha As DataRow In dt.Rows

Dim lista As New ListViewItem

lista.Text = Convert.ToInt32(linha("IdCliente"))

lista.SubItems.Add(linha("Nome").ToString())

lista.SubItems.Add(linha("Endereco").ToString())

lista.SubItems.Add(linha("Cidade").ToString())

lista.SubItems.Add(linha("Telefone").ToString())

lista.SubItems.Add(linha("Email").ToString())

If linha("Ativo").ToString().Equals("True") Then

lista.SubItems.Add("Sim")

Else

lista.SubItems.Add("Não")

End If

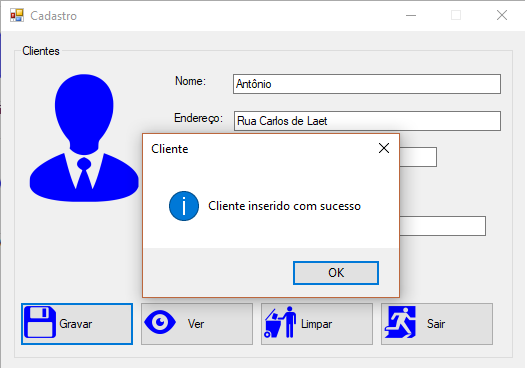
lsvDados.Items.Add(lista)

Next

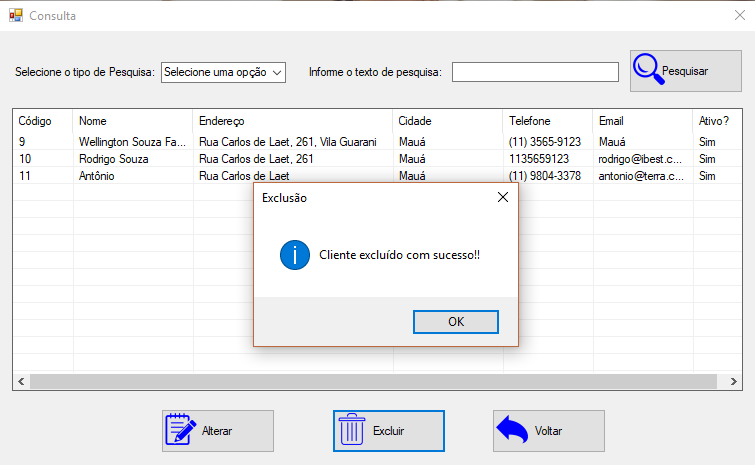
End Sub

Entenderam? Dessa forma eu passo para os campos do form de **Cadastro** os valores de meu **ListView** (como este é um teste não passei todos os valores).

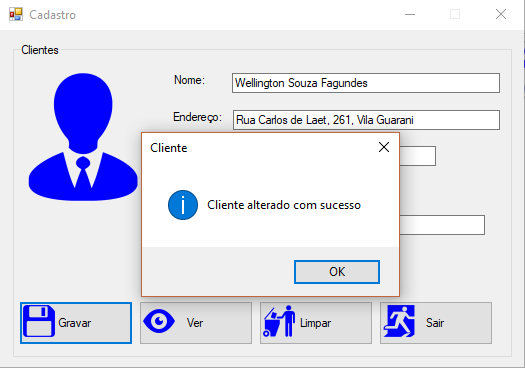
Finalizando, faça os testes. Veja os mesmos nas Figuras **01**, **02** e **03**:



*Figura 01 – Método* ***GravarCliente***



*Figura 02 – Método* ***ConsultarClientes*** *e* ***ExcluirCliente***



*Figura 03 – Método* ***AlterarCliente***