



Log In / Cadastre-se

Pesquisar



HOME DESENVOLVIMENTO FRONT-END BANCO DE DADOS EM DESTAQUE TODOS

PUBLIQUE

Desenvolvimento - C#

Guia prático para o desenvolvimento de Aplicações C# em Camadas - parte 6

Este é o sexto artigo da série onde vamos demonstrar passo-a-passo a construção de uma aplicação .Net utilizando o conceito de desenvolvimento em camadas.

por Carlos Camacho

g+1 0 Like 2

6 – Criando a interface com o usuário – Formulário de Clientes

Neste artigo vamos iniciar a implementação da camada de interface do usuário construindo um aplicativo para Microsoft Windows. Vamos criar o Formulário de Clientes utilizando todas as camadas já desenvolvidas nos artigos anteriores.

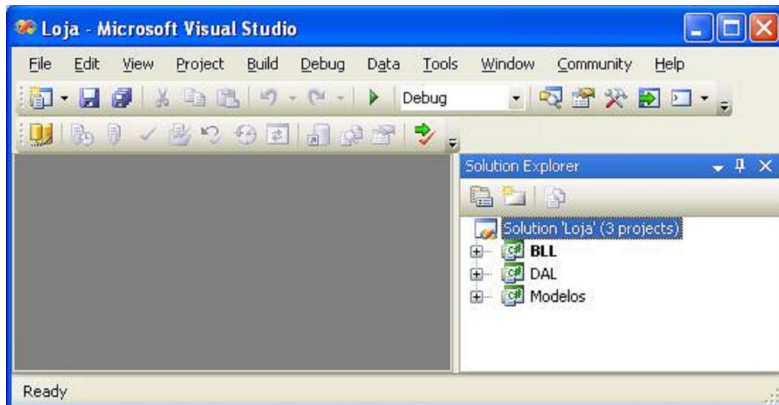
[Modelos] DAL ð BLL ð User Interface

- Abra o Microsoft Visual Studio;

- Clique em **File > Open > Project/Solution...**

- Na janela *Open Project*, selecione o arquivo da nossa Solution (**C:\Loja\Modelos\Loja.sln**) e clique em **Open** para abrir a solução.

Neste momento temos implementados os projetos Modelos, DAL e BLL:



Vamos adicionar um projeto de *user interface* para Windows chamado UIWindows.

- Clique com o botão direito na Solution Loja e escolha **Add > New Project...**

- Na janela "Add New Project" vamos informar os seguintes dados:

Project type: **Visual C#**

Template: **Windows Application**

Name: **UIWindows**

Location: **C:\LojaUIWindows**

- Clique em **Ok** para adicionar o projeto.



Publicidade

Compuware dynaTrace
É muito simples monitorar **100%** das transações do seu App

Experimente a Versão **FREE TRIAL** Compuware **APM**

REVISTAS DEVMEDIA



.net Mag 116



Easy .net mag 37

VER TODAS

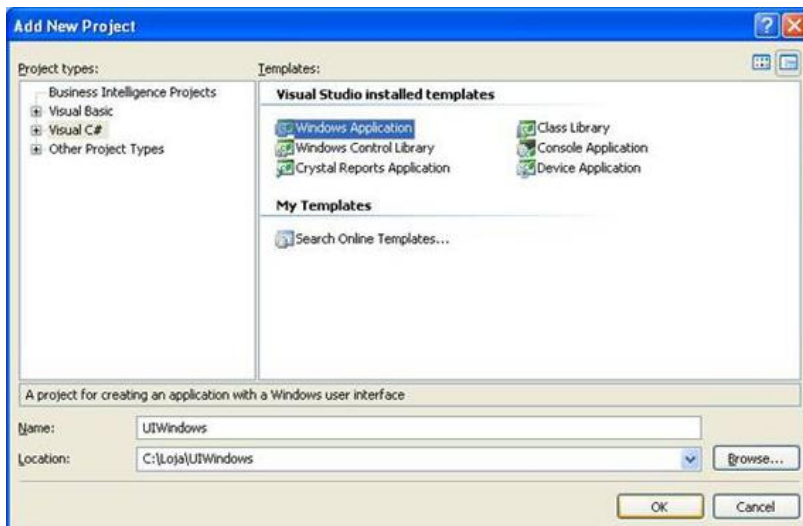
ASSINE

TOP 10 - ARTIGOS

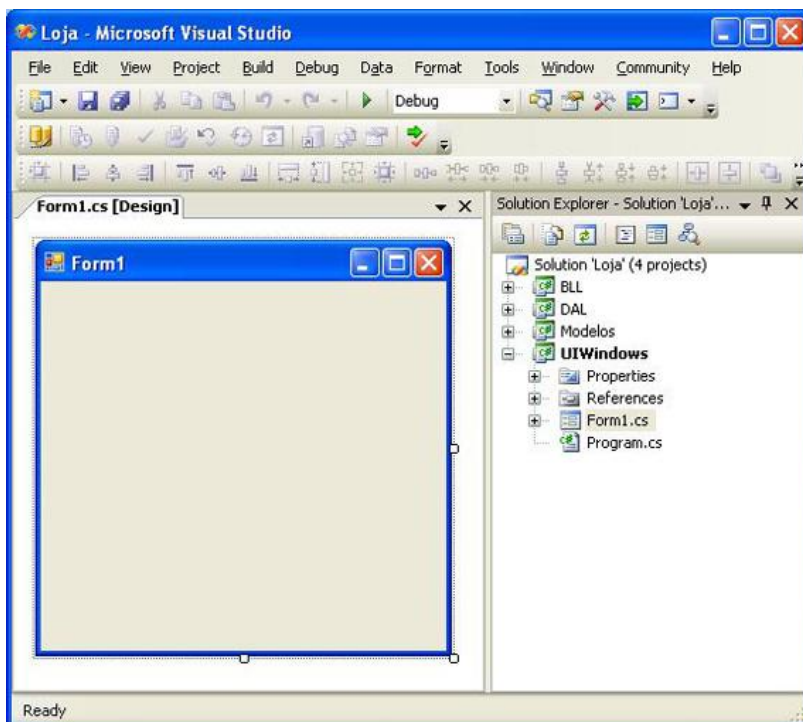
TOP 10 - AUTORES

- 1 Comandos básicos em SQL - insert, update, delete e select
- 2 HTML Básico
- 3 Menu em CSS - Menu dropdown horizontal com HTML5 e CSS3
- 4 Criando um sistema de cadastro e login com PHP e MySQL
- 5 Excel: Como verificar se existe valores duplicados
- 6 Código para background HTML e CSS
- 7 Copiando dados com o Robocopy
- 8 Formato dos registros do Sintegra
- 9 Calendário em jQuery - Criando Calendários com DatePicker
- 10 Trabalhando com Div em HTML

VER TODOS



O aplicativo para Windows UIWindows será criado:



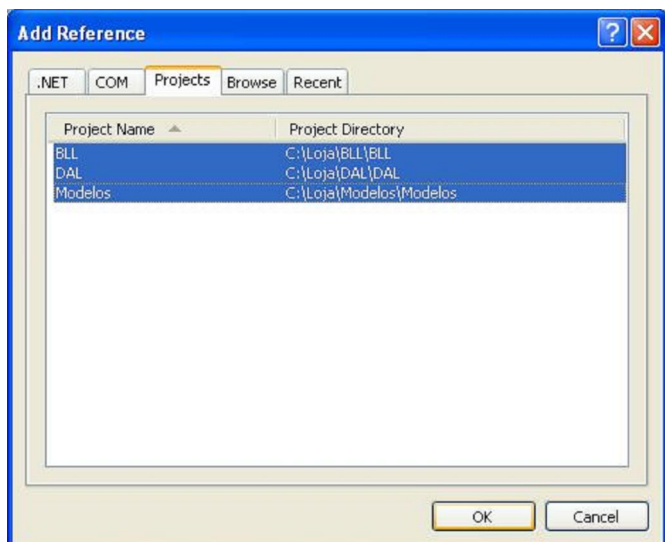
A primeira coisa a ser feita é adicionar os projetos que usaremos como referência.

Olhando para o nosso modelo de camadas abaixo:

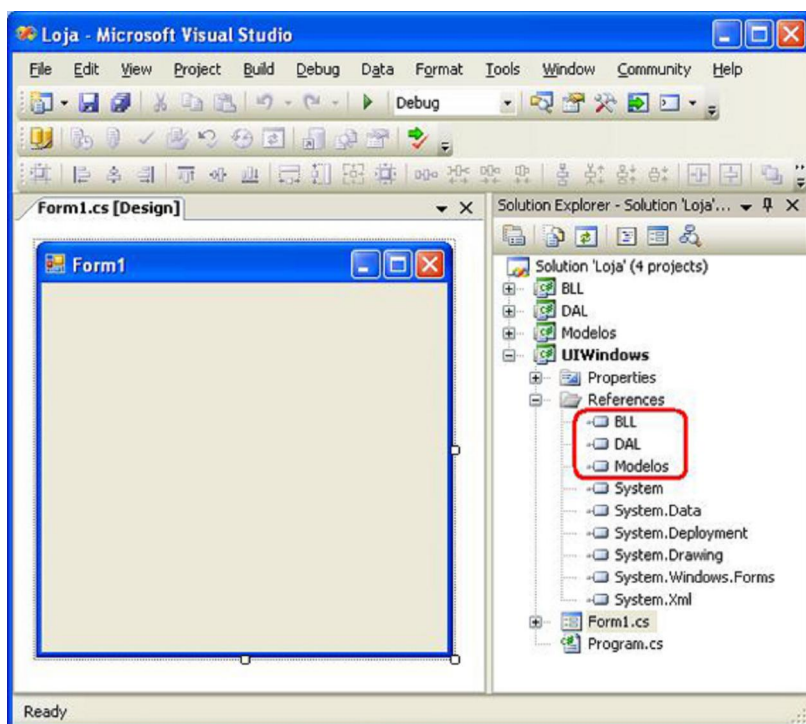
[Modelos] DAL ð BLL ð User Interface

Verificamos que é preciso adicionar referência aos projetos Modelos, DAL e BLL. Vamos fazer isso agora.

- Clique com o botão direito na pasta References do projeto BLL e escolha **Add Reference...**
- Na janela "Add Reference", clique na aba **Projects**;
- Mantenha a tecla <CTRL> pressionada e clique sobre os projetos DAL, Modelos e BLL para selecioná-los;
- Clique em **Ok** para adicionar as referências.



Podemos verificar as referências que acabamos de adicionar ao projeto UIWindows:



No nosso formulário principal, vamos criar um menu para que o usuário possa acessar os demais formulários da aplicação.

Vamos exibir a **Toolbox** (caixa de ferramentas) que contém o menu que precisaremos incluir.

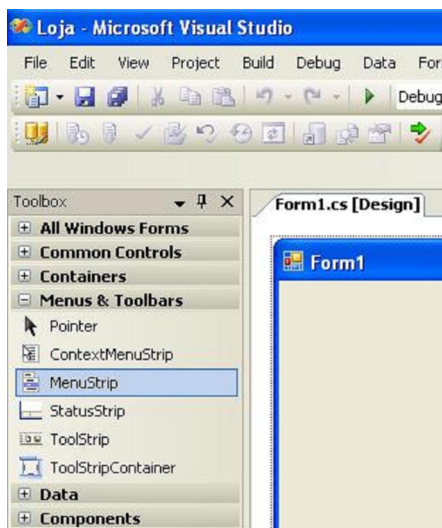
Para exibir a Toolbox você pode:

1) Utilizar a combinação de teclas **<Ctrl> + <Alt> + X** ou

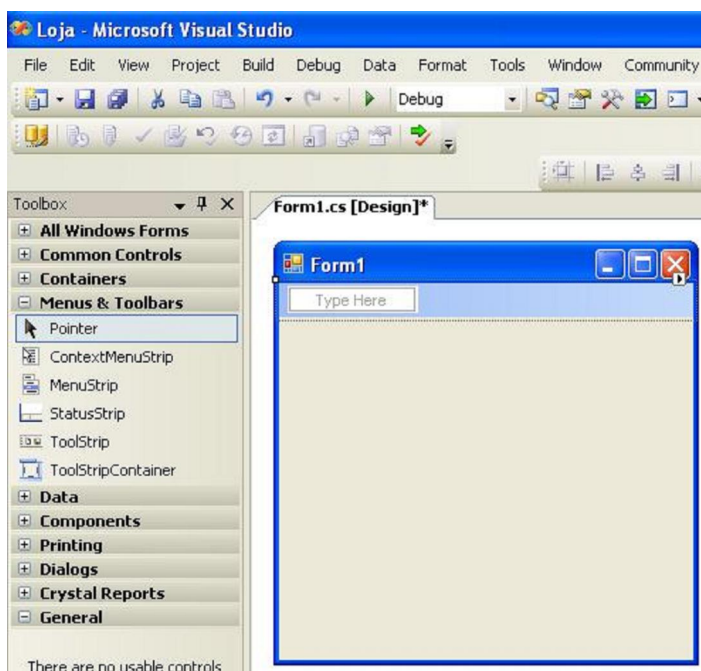
2) Utilizar o menu **View > Toolbox**

- Na Toolbox, abra o item **"Menus & Toolbars"**;

- Para criar o nosso menu, arraste o componente **MenuStrip** para o formulário **Form1**.

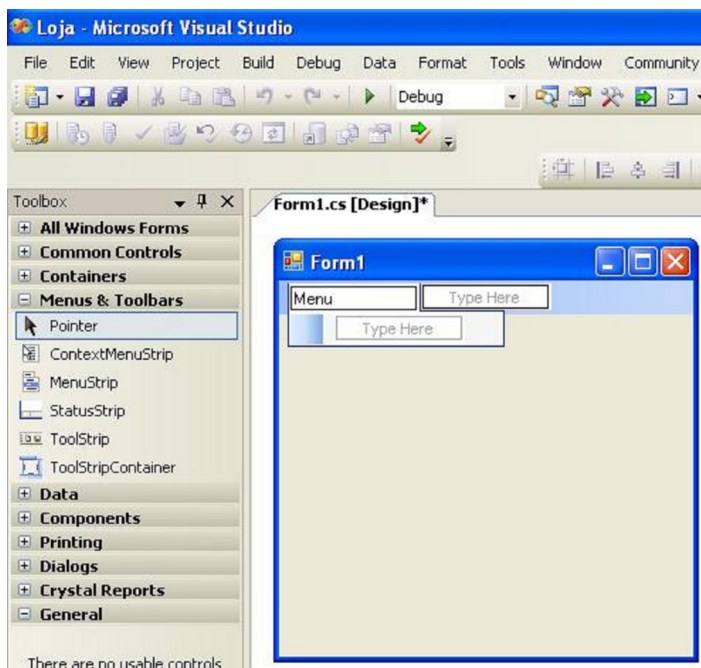


Ao soltar o *MenuStrip* dentro do formulário Form1, um “menu vazio” foi criado.



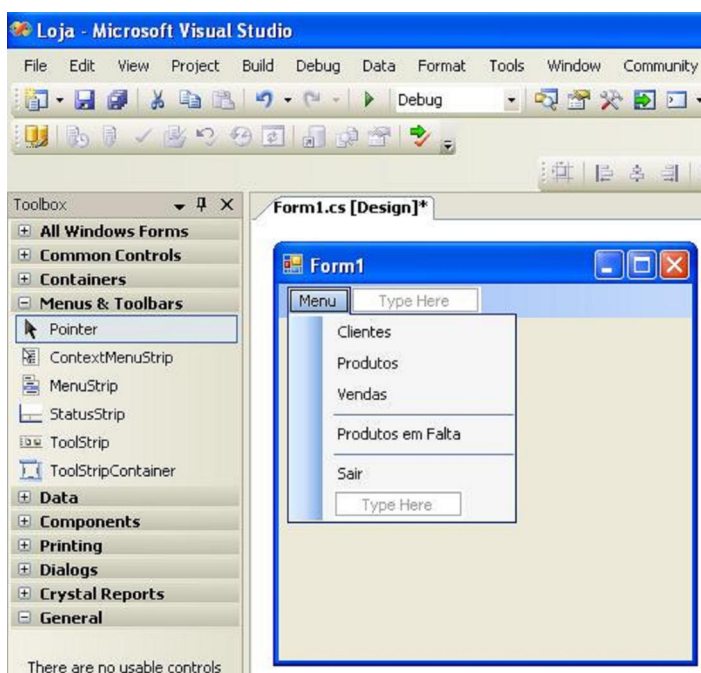
- Clique na área com a indicação “Type here” e digite **Menu**;

Veja que áreas se abrem ao lado e abaixo no caso de quisermos incluir mais itens no menu:



- Insira os itens de menu Clientes, Produtos, Vendas, Produtos em Falta e Sair como na figura abaixo:

Dica: Use o hífen (" - ") para criar os traços de divisão que você está vendo na figura.



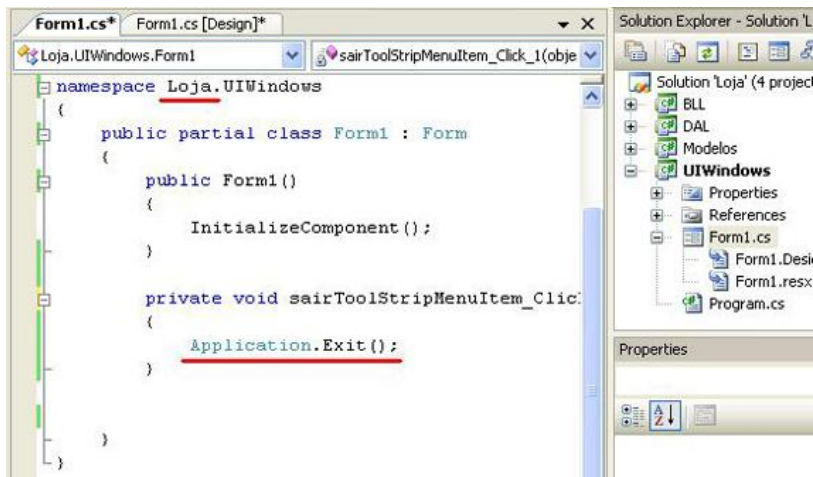
- Dê um duplo clique com o botão esquerdo do mouse sobre o item **Sair** do Menu para codificarmos o que deve acontecer quando o usuário selecionar essa opção.

- Dentro do método do item de menu Sair que se abrirá, insira o seguinte código:

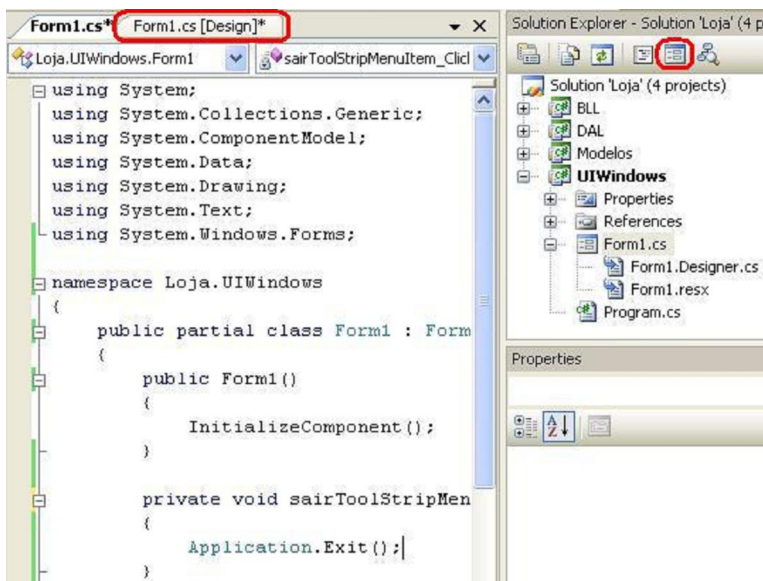
```
Application.Exit();
```

Dessa forma, sempre que o usuário clicar na opção Sair a aplicação será encerrada.

- Acrescente também o nome do projeto **Loja** no namespace como a seguir:



- Agora vamos para o modo de Design para visualizar o formulário novamente. Para ir para o modo de design utilize a aba [Design] do *Form1* ou o botão “View Design” do Solution Explorer como indicado a seguir:

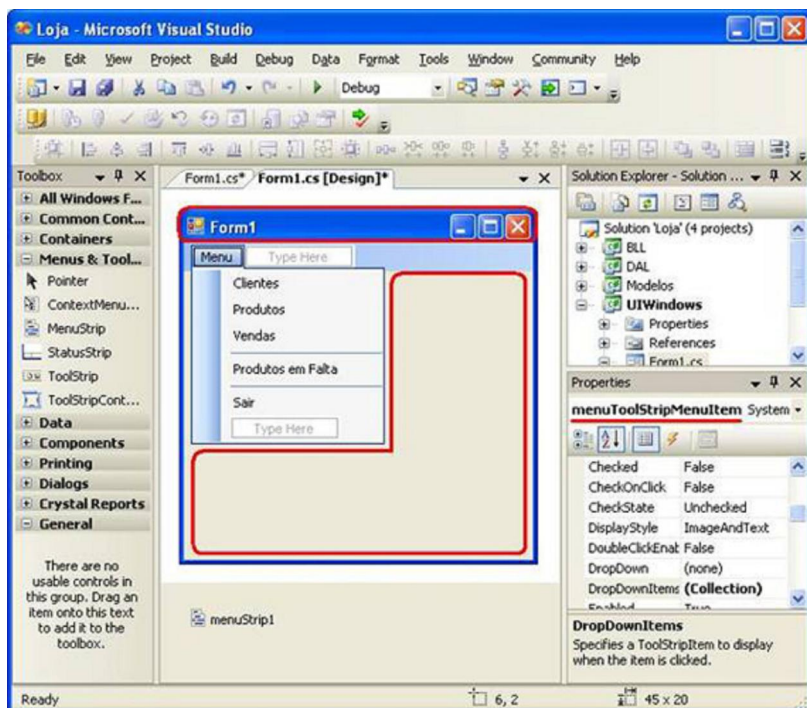


No modo de Design, é o componente de Menu que está selecionado. Por esse motivo, na janela de propriedades (*Properties*) no canto inferior direito do MS Visual Studio, são as propriedades do objeto de Menu que estão sendo exibidas.

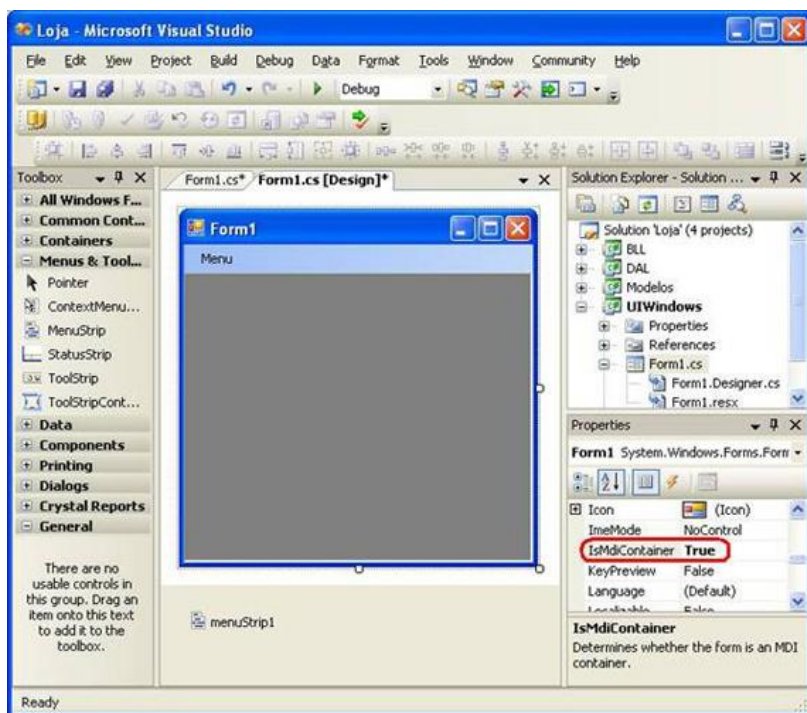
Agora nós queremos alterar uma propriedade do objeto *Form1*. Para selecioná-lo e fazer com que as propriedades do formulário *Form1* sejam exibidas na janela de propriedades, clique na barra de títulos ou em qualquer área do formulário *Form1* fora do objeto Menu conforme destacado abaixo:

- **Clique em uma área do formulário *Form1*** para alterarmos uma de suas propriedades.

Dica: Se a janela de propriedades não estiver visível, pressione <F4> para exibí-la.



- Na janela de propriedades do **Form1**, altere a propriedade **IsMdiContainer** para **True**;

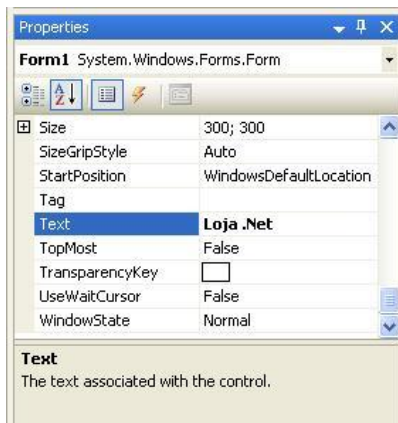


Isso significa que este formulário será o Menu Principal da nossa aplicação. Essa alteração indica que todos os formulários chamados pelo *Form1* serão abertos dentro dele, e não em uma nova janela.

Assim, se fecharmos o formulário to Menu Principal todos os formulários que estiverem abertos dentro da aplicação (exemplo: formulário de clientes, formulário de produtos) também serão fechados.

Agora vamos alterar o nome da nossa aplicação de *Form1* para "**Loja .Net**"

- Na janela de propriedades do *Form1*, altere a propriedade **Text** para "**Loja .Net**".



Isso altera automaticamente o nome do formulário exibido na barra de títulos.

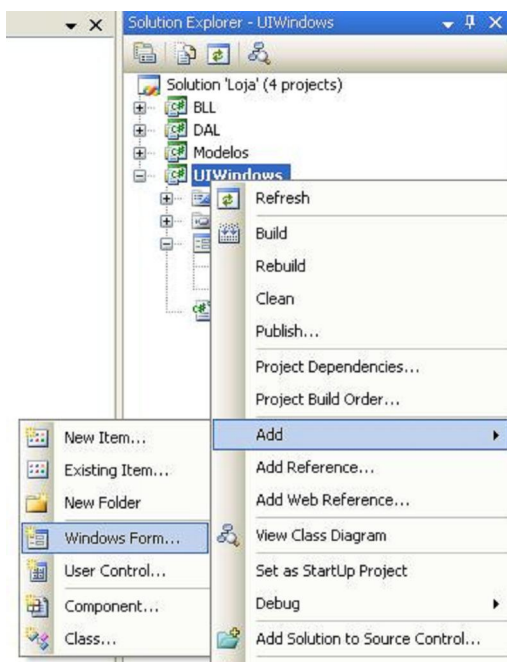


Para testarmos tudo o que já desenvolvemos até agora vamos criar 4 formulários. São eles:

- Clientes;
- Produtos;
- Vendas;
- Produtos em Falta.

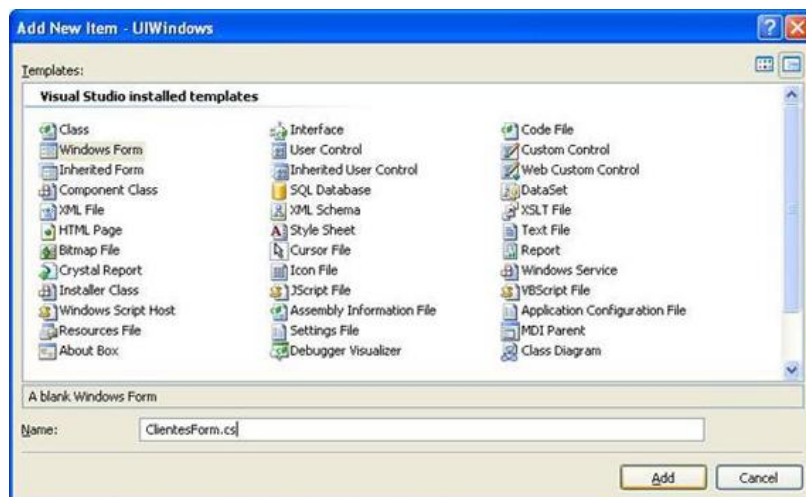
6.1) Criando o formulário de Clientes

- Clique com o botão direito sobre o projeto **UIWindows** e escolha **Add > Windows Form...**



- Na janela "Add New Item", selecione o template **Windows Form** e nomeie o novo formulário como **ClientesForm.cs**

- Clique em **Add** para adicionar o formulário.



(- Certifique-se de que o namespace esteja como Loja.UIWindows tanto em ClientesForm.cs quanto em ClientesForm.Designer.cs;

Colocar using Loja.BLL; no codebehind ClientesForm.cs)

- Na janela de propriedades do formulário **ClientesForm**, defina as propriedades como abaixo:

Size: **448; 512**

- A partir do menu "Common Controls" da Toolbox, arraste quatro objetos do tipo **Label** para o formulário **ClientesForm**.

- Defina as propriedades dos 4 labels como a seguir:

Name: codigoLabel
Location: 16;32
Text: Código:

Name: nomeLabel
Location: 16;64
Text: Nome:

Name: emailLabel
Location: 16;96
Text: E-mail:

Name: telefoneLabel
Location: 16;128
Text: Telefone:

Neste momento, nosso formulário de clientes estará com esta aparência:



- A partir do menu "Common Controls" da Toolbox, arraste quatro objetos do tipo **TextBox** para o formulário **ClientesForm**.

- Defina as propriedades dos 4 TextBox como a seguir:

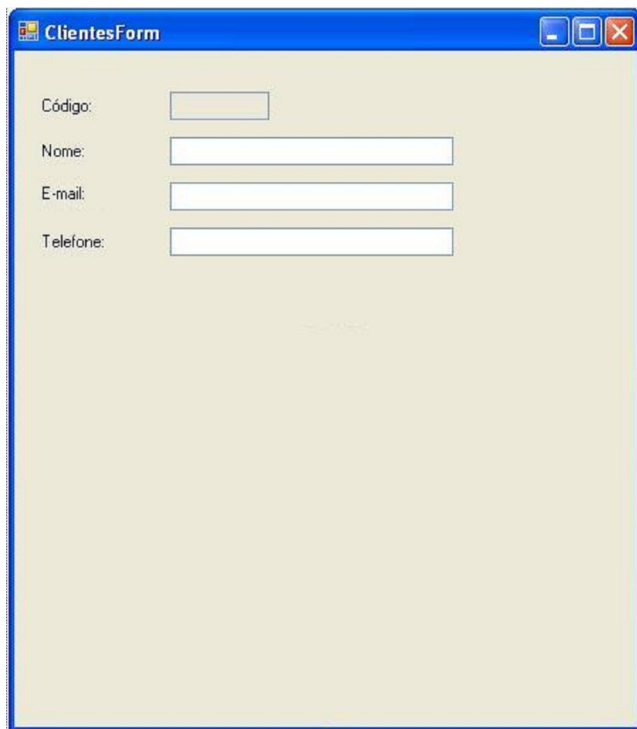
Name: codigoTextBox
Enable: False
Location: 110;29
ReadOnly: True
Size: 70;20

Name: nomeTextBox
Location: 110;61
Size: 200;20

Name: emailTextBox
Location: 110;93
Size: 200;20

Name: telefoneTextBox
Location: 110;125
Size: 200;20

Nosso formulário de clientes agora está com essa aparência:



- A partir do menu "Common Controls" da Toolbox, arraste cinco objetos do tipo **Button** para o formulário **ClientesForm**.

- Defina as propriedades dos 5 Button como a seguir:

Name: limparButton
Location: 110;159
Size: 75;23
Text: Limpar

Name: incluirButton
Location: 191;159
Size: 75;23
Text: Incluir

Name: alterarButton
Location: 272;159
Size: 75;23
Text: Alterar

Name: excluirButton
Location: 353;159
Size: 75;23
Text: Excluir

Name: lerButton
Anchor: Bottom, Right
Location: 353; 447

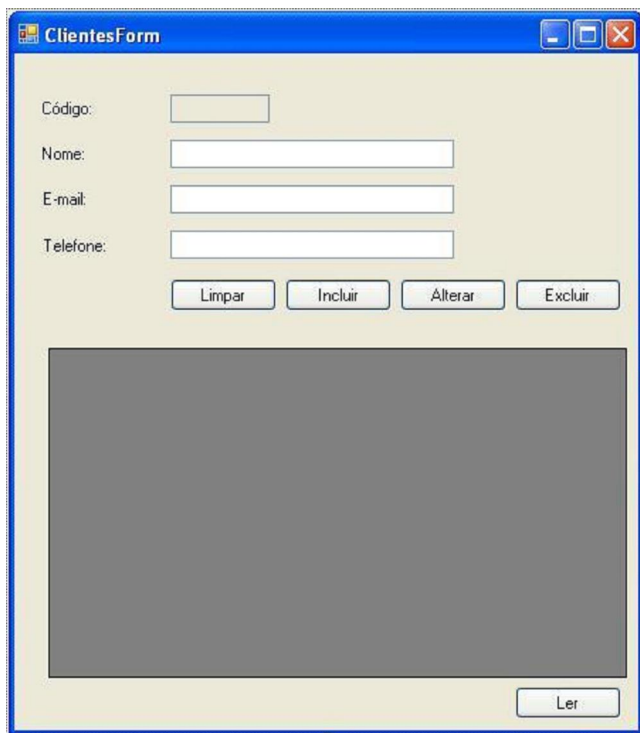
Size: 75;23
Text: Ler

- A partir do menu "Data" da Toolbox, arraste um objeto do tipo **DataGridView** para o formulário **CientesForm**.

- Defina as propriedades do DataGridView como a seguir:

Name: clientesDataGridView
Anchor: Top, Bottom, Left, Right
Location: 24; 208
Size: 408; 233

Agora nosso formulário de clientes está assim:



Muito bem! O *design*, ou seja, a parte gráfica do nosso formulário de clientes está finalizada. Agora precisamos programar as ações que desejamos que aconteça quando o usuário abrir esse formulário e interagir clicando nos botões. Vamos lá?!

- Para ver o código-fonte (também chamado de *code behind*) clique com o botão direito sobre o formulário **CientesForm** no Solution Explorer e escolha a opção "**View Code**".

Após o método `CientesForm_Load()` vamos criar o método `AtualizaGrid()`. No `AtualizaGrid` faremos a comunicação com a camada de regras de negócio (BLL) para que possamos preencher os valores dos objetos `Textbox`.

Dentro do método `CientesForm_Load()`, que é chamado toda vez que o formulário de clientes é carregado em memória, vamos fazer uma chamada para o método `AtualizaGrid()` e também posicionaremos o cursor no primeiro campo do formulário.

- Copie e cole o trecho de código para o code behind do formulário `CientesForm`:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.ComponentModel;
```

```
using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using Loja.BLL;

namespace Loja.UIWindows
{
    public partial class ClientesForm : Form
    {
        public ClientesForm()
        {
            InitializeComponent();
        }

        public void AtualizaGrid()
        {
            // Comunicação com a Camada BLL

            ClientesBLL obj = new ClientesBLL();

            clientesDataGridView.DataSource = obj.Listagem();

            // Atualizando os objetos TextBox

            codigoTextBox.Text = clientesDataGridView[0,
            clientesDataGridView.CurrentRow.Index].Value.ToString();

            nomeTextBox.Text = clientesDataGridView[1,
            clientesDataGridView.CurrentRow.Index].Value.ToString();

            emailTextBox.Text = clientesDataGridView[2,
            clientesDataGridView.CurrentRow.Index].Value.ToString();

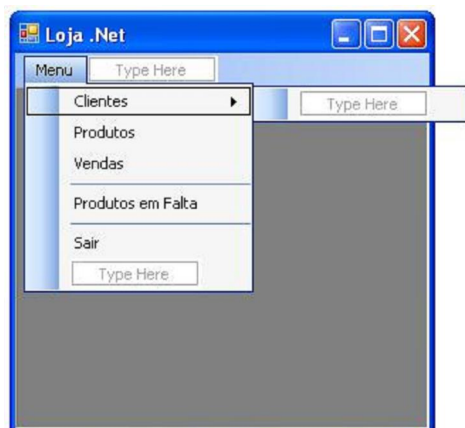
            telefoneTextBox.Text = clientesDataGridView[3,
            clientesDataGridView.CurrentRow.Index].Value.ToString();
        }

        private void ClientesForm_Load(object sender, EventArgs e)
        {
            AtualizaGrid();

            nomeTextBox.Focus();
        }
    }
}
```

Para testarmos se essa exibição inicial dos dados dos clientes está funcionando, precisamos criar uma chamada para este formulário a partir do menu principal.

- **Abra o Form1** em modo de design e **clique sobre o Menu** para que possamos ver as opções existentes:



- Dê um **duplo clique** sobre a opção **Clientes** para codificarmos a chamada para o formulário correspondente.

- O MS Visual Studio abrirá o código do formulário com o cursor dentro do método que será executado quando o usuário clicar no item Clientes do Menu. Dentro deste método (chamado *clientesToolStripMenuItem_Click*) digite o seguinte código:

```
ClientesForm obj = new ClientesForm();
```

```
obj.MdiParent = this;
```

```
obj.Show();
```

A listagem completa do nosso code behind Form1.cs ficou assim:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.ComponentModel;
```

```
using System.Data;
```

```
using System.Drawing;
```

```
using System.Text;
```

```
using System.Windows.Forms;
```

```
namespace Loja.UIWindows
```

```
{
```

```
    public partial class Form1 : Form
```

```
    {
```

```
        public Form1()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        private void sairToolStripMenuItem_Click_1(object sender, EventArgs e)
```

```
        {
```

```
            Application.Exit();
```

```
        }
```

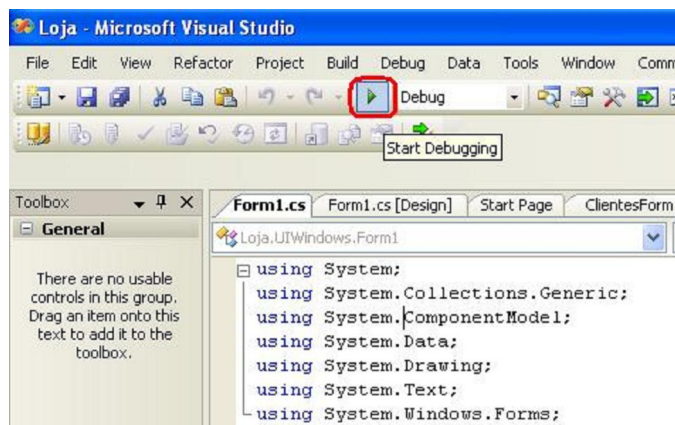
```
        private void clientesToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
```

```
            ClientesForm obj = new ClientesForm();
```

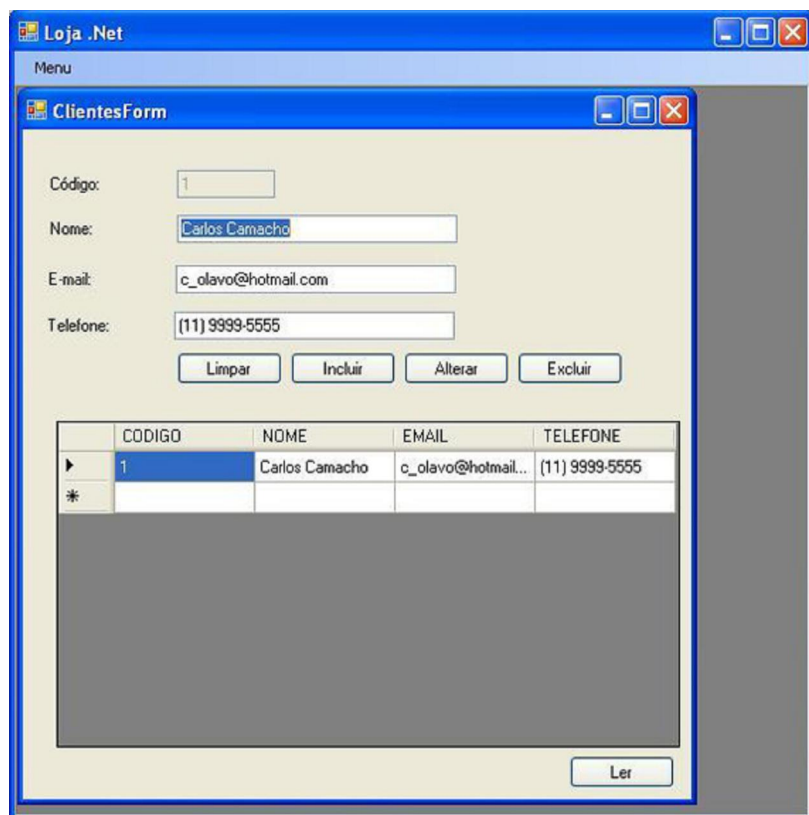
```
obj.MdiParent = this;  
  
obj.Show();  
  
}  
  
}  
  
}
```

- Dê um **"Start Debugging"** clicando na seta verde da barra de botões conforme indicado abaixo. Essa opção compila o projeto e, se tudo estiver certo, já executa a nossa aplicação exibindo o menu principal.



Observações: Se a aplicação não abrir, clique com o botão direito sobre o projeto **UIWindows** e selecione a opção **"Set as StartUp Project"**. Com isso estamos indicando que, na nossa solução, o projeto UIWindows será o primeiro a ser executado.

- Na aplicação Loja .Net em execução, escolha **Menu > Clientes**:



Podemos ver que a carga inicial de dados funcionou conforme o esperado.

Agora vamos codificar as funcionalidades dos botões do formulário de clientes.

- Dê um **duplo clique** no **botão Limpar** para codificarmos o evento click;
- Dentro do evento clique do botão Limpar, copie e cole o código abaixo:

```
codigoTextBox.Text = "";

nomeTextBox.Text = "";

emailTextBox.Text = "";

telefoneTextBox.Text = "";
```

- Dê um **duplo clique** no **botão Incluir** para codificarmos o evento click;
- Dentro do evento clique do botão Incluir, copie e cole o código abaixo:

```
try

{

    ClienteInformation cliente = new ClienteInformation();

    cliente.Nome = nomeTextBox.Text;

    cliente.Email = emailTextBox.Text;

    cliente.Telefone = telefoneTextBox.Text;

    ClientesBLL obj = new ClientesBLL();

    obj.Incluir(cliente);

    MessageBox.Show("O cliente foi incluído com sucesso!");

    codigoTextBox.Text = Convert.ToString(cliente.Codigo);

    AtualizaGrid();

}

catch (Exception ex)

{

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

}
```

- Dê um **duplo clique** no **botão Alterar** para codificarmos o evento click;
- Dentro do evento clique do botão Alterar, copie e cole o código abaixo:

```
if (codigoTextBox.Text.Length == 0)

{

    MessageBox.Show("Um cliente deve ser selecionado para alteração.");

}

else

try

{

    ClienteInformation cliente = new ClienteInformation();

    cliente.Codigo = int.Parse(codigoTextBox.Text);

    cliente.Nome = nomeTextBox.Text;

    cliente.Email = emailTextBox.Text;

    cliente.Telefone = telefoneTextBox.Text;

    ClientesBLL obj = new ClientesBLL();

    obj.Alterar(cliente);

}
```

```
        MessageBox.Show("O cliente foi alterado com sucesso!");

        AtualizaGrid();
    }

    catch (Exception ex)

    {

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

    }
```

- Dê um **duplo clique** no **botão Excluir** para codificarmos o evento click;
- Dentro do evento clique do botão Excluir, copie e cole o código abaixo:

```
if (codigoTextBox.Text.Length == 0)

{

    MessageBox.Show("Um cliente deve ser selecionado antes da exclusão.");

}

else

    try

    {

        int codigo = Convert.ToInt32(codigoTextBox.Text);

        ClientesBLL obj = new ClientesBLL();

        obj.Excluir(codigo);

        MessageBox.Show("O cliente foi excluído com sucesso!");

        AtualizaGrid();

    }

    catch (Exception ex)


    {

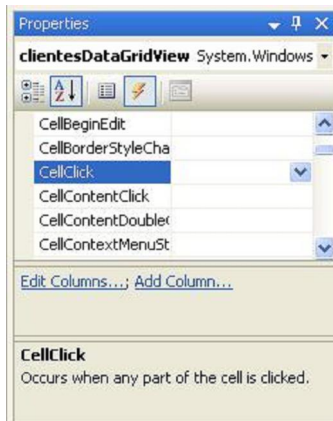
        MessageBox.Show(ex.Message);

    }
```

- Dê um **duplo clique** no **botão Ler** para codificarmos o evento click;
- Dentro do evento clique do botão Ler, copie e cole o código abaixo:

```
AtualizaGrid();
```

- Clique sobre o dataGridView;
- Na janela de propriedades, clique no ícone que tem um relâmpago () para exibir a lista de eventos;
- Dê um duplo clique no evento **CellClick** para inserirmos o código para esse evento:



- Dentro do evento **CellClick**, copie e cole o código abaixo:

```
// Atualizando os objetos TextBox

codigoTextBox.Text = clientesDataGridView[0,
clientesDataGridView.CurrentRow.Index].Value.ToString();

nomeTextBox.Text = clientesDataGridView[1,
clientesDataGridView.CurrentRow.Index].Value.ToString();

emailTextBox.Text = clientesDataGridView[2,
clientesDataGridView.CurrentRow.Index].Value.ToString();

telefoneTextBox.Text = clientesDataGridView[3,
clientesDataGridView.CurrentRow.Index].Value.ToString();
```

- O code behind completo do nosso ClientesForm.cs ficou assim:

```
/*
 * File....: ClientesForm.cs
 * Author.: Carlos Camacho - www.camachojunior.com.br
 * Content: Formulário de Clientes
 * Subject: Guia prático para o desenvolvimento de Aplicações C# em Camadas
 */

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using Loja.BLL;

using Loja.DAL;

using Loja.Modelos;

namespace Loja.UIWindows
{
    public partial class ClientesForm : Form
    {
        public ClientesForm()
    }
}
```



```
{  
    InitializeComponent();  
}  
  
public void AtualizaGrid()  
{  
    // Comunicação com a Camada BLL  
  
    ClientesBLL obj = new ClientesBLL();  
  
    clientesDataGridView.DataSource = obj.Listagem();  
  
    // Atualizando os objetos TextBox  
  
    codigoTextBox.Text = clientesDataGridView[0,  
clientesDataGridView.CurrentRow.Index].Value.ToString();  
  
    nomeTextBox.Text = clientesDataGridView[1,  
clientesDataGridView.CurrentRow.Index].Value.ToString();  
  
    emailTextBox.Text = clientesDataGridView[2,  
clientesDataGridView.CurrentRow.Index].Value.ToString();  
  
    telefoneTextBox.Text = clientesDataGridView[3,  
clientesDataGridView.CurrentRow.Index].Value.ToString();  
}  
  
private void ClientesForm_Load(object sender, EventArgs e)  
{  
    AtualizaGrid();  
  
    nomeTextBox.Focus();  
}  
  
private void limparButton_Click(object sender, EventArgs e)  
{  
    codigoTextBox.Text = "";  
    nomeTextBox.Text = "";  
    emailTextBox.Text = "";  
    telefoneTextBox.Text = "";  
}  
  
private void incluirButton_Click(object sender, EventArgs e)  
{  
    try  
    {  
        ClienteInformation cliente = new ClienteInformation();  
  
        cliente.Nome = nomeTextBox.Text;  
  
        cliente.Email = emailTextBox.Text;  
  
        cliente.Telefone = telefoneTextBox.Text;  
  
        ClientesBLL obj = new ClientesBLL();
```

```
obj.Incluir(cliente);

MessageBox.Show("O cliente foi incluído com sucesso!");

codigoTextBox.Text = Convert.ToString(cliente.Codigo);

AtualizaGrid();
}

catch (Exception ex)
{
    MessageBox.Show("Erro: " + ex.Message);
}

}

private void alterarButton_Click(object sender, EventArgs e)
{
    if (codigoTextBox.Text.Length == 0)
    {
        MessageBox.Show("Um cliente deve ser selecionado para alteração.");
    }
    else
    {
        try
        {
            ClienteInformation cliente = new ClienteInformation();

            cliente.Codigo = int.Parse(codigoTextBox.Text);

            cliente.Nome = nomeTextBox.Text;

            cliente.Email = emailTextBox.Text;

            cliente.Telefone = telefoneTextBox.Text;

            ClientesBLL obj = new ClientesBLL();

            obj.Alterar(cliente);

            MessageBox.Show("O cliente foi alterado com sucesso!");

            AtualizaGrid();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Erro: " + ex.Message);
        }
    }
}

private void excluirButton_Click(object sender, EventArgs e)
{
    if (codigoTextBox.Text.Length == 0)
    {
        MessageBox.Show("Um cliente deve ser selecionado antes da exclusão.");
    }
}
```

```
    }

    else

        try

        {

            int codigo = Convert.ToInt32(codigoTextBox.Text);

            ClientesBLL obj = new ClientesBLL();

            obj.Excluir(codigo);

            MessageBox.Show("O cliente foi excluído com sucesso!");

            AtualizaGrid();

        }

        catch (Exception ex)

        {

            MessageBox.Show(ex.Message);

        }

    }

    private void lerButton_Click(object sender, EventArgs e)

    {

        AtualizaGrid();

    }

    private void clientesDataGridView_CellClick(object sender, DataGridViewCellEventArgs e)

    {

        // Atualizando os objetos TextBox

        codigoTextBox.Text = clientesDataGridView[0,
clientesDataGridView.CurrentRow.Index].Value.ToString();

        nomeTextBox.Text = clientesDataGridView[1,
clientesDataGridView.CurrentRow.Index].Value.ToString();

        emailTextBox.Text = clientesDataGridView[2,
clientesDataGridView.CurrentRow.Index].Value.ToString();

        telefoneTextBox.Text = clientesDataGridView[3,
clientesDataGridView.CurrentRow.Index].Value.ToString();

    }

}
```

Parabéns! Você acabou de implementar o Formulário de Clientes do nosso projeto Loja .Net. Agora é só executar o projeto, acessar o Formulário de Clientes e testar todas as funcionalidades que você implementou.

No próximo artigo vamos implementar o Formulário de Produtos.

Até o próximo artigo!



Carlos Camacho - Carlos Olavo de Azevedo Camacho Júnior é mestrando em



Tecnologias da Inteligência e Design Digital pela Pontifícia Universidade Católica de São Paulo PUCSP. Pós-graduado em Análise e Projeto de Sistemas pela Universidade Paulista UNIP. Bacharel em Ciência da Computação pela Universidade Paulista UNIP e possui Licenciatura Plena em Matemática pelas Faculdades Oswaldo Cruz.
MCP .Net, MCP SQL Server, Carlos Camacho leciona disciplinas técnicas na área de Ciências Exatas e é Consultor em Tecnologia da Informação para Instituições Financeiras.



g+1 0

Like 2

Leia também

Lambda Expressions x SQL: Comparando a sintaxe de consultas comuns
C#

List: trabalhando com listas genéricas em C#
C#

Criando Gráficos usando C# e API do Google
C#

Imprimindo um Panel com C#
C#

Consumindo um Web API em C#
C#

Estamos aqui:    

Linha de Código faz parte do grupo Web-03

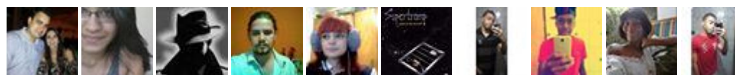
[Política de privacidade e de uso](#) | [Anuncie](#) | [Cadastre-se](#) | [Fale conosco](#)



Linha de Código

Like

9,980 people like Linha de Código.



Facebook social plugin

© 2014 Linha de Código. Todos os direitos reservados