## Aplicação em camadas utilizando

## C# e Conceitos de POO

**Parte 1**

**Objetivo**: praticar o desenvolvimento de uma aplicação para que você adquira habilidades para a construção ou manutenção de aplicações mais complexas.

**Conceitos Trabalhados:**

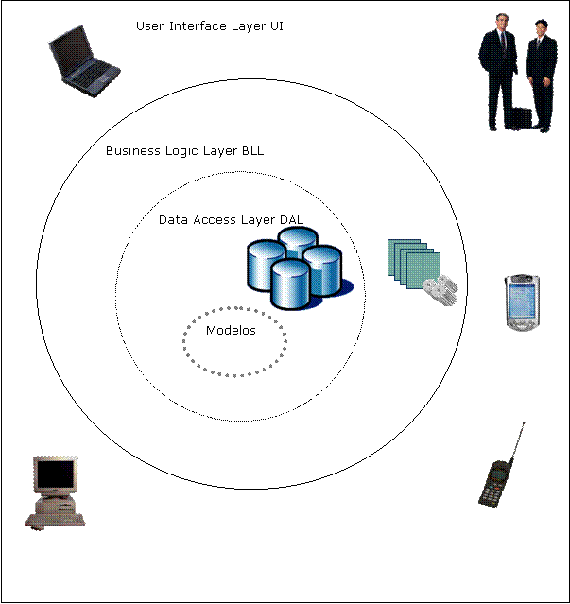
* Programação orientada a objetos;
* Integridade referencial de banco de dados;
* Uso de transações via linguagem de programação C# .Net;
* Implementação de algoritmos de segurança no desenvolvimento de aplicações;

**Conhecimentos a serem adquiridos:**

* Comunicação entre as camadas;
* Vantagens do modelo de desenvolvimento em camadas;
* Controle de transações do banco de dados com o ADO .Net;
* Construção de uma aplicação para Windows;
* Construção de uma aplicação para a Web;
* A importância da segurança no desenvolvimento de aplicações.

**A metodologia utilizada será o desenvolvimento em três camadas. Veremos a diferença entre:**

* Camada de acesso à dados ou Data Access Layer (DAL);
* Camada de regras de negócio ou Business Logic Layer (BLL); e
* Camada de interface do usuário ou User Interface (UI).



**Parte 2 – Criando a infraestrutura**

**Nome do Banco**: Loja

**Tabelas**: Clientes

CREATE TABLE [CLIENTES] (

[CODIGO] [int] IDENTITY (1, 1) NOT NULL ,

[NOME] [varchar] (100) ,

[EMAIL] [varchar] (100) ,

[TELEFONE] [varchar] (80) ,

CONSTRAINT [PK\_CLIENTES] PRIMARY KEY CLUSTERED

(

[CODIGO]

) ON [PRIMARY]

) ON [PRIMARY]

GO

Insira um registro através do código abaixo:

insert into clientes(nome,email,telefone)

values ("Teste","teste@hotmail.com","(31) 8888-3333")

**Tabelas**: Produtos

CREATE TABLE [PRODUTOS] (

[CODIGO] [int] IDENTITY (1, 1) NOT NULL,

[NOME] [varchar] (100),

[PRECO] decimal(10,2) ,

[ESTOQUE] [int] ,

CONSTRAINT [PK\_PRODUTOS] PRIMARY KEY CLUSTERED

(

[CODIGO]

) ON [PRIMARY]

) ON [PRIMARY]

GO

Agora vamos inserir três produtos utilizando o código abaixo:

insert into produtos (nome, preco, estoque)

values ("Computador Dell Inspiron 3647-B20M com Intel","2312.10","15")

insert into produtos (nome, preco, estoque)

values ("Notebook Samsung ATIV Book 3 370E4K-KD4 com Intel Core i3","1399.90","150")

insert into produtos (nome, preco, estoque)

values ("Impressora HP Officejet Pro 6230 ePrinter com Conexão WiFi","289.99","50")

**Tabela**: Vendas

CREATE TABLE [VENDAS] (

[CODIGO] [int] IDENTITY (1, 1) NOT NULL ,

[DATA] [datetime],

[QUANTIDADE] [int],

[FATURADO] bit,

[CODIGOCLIENTE] [int],

[CODIGOPRODUTO] [int],

CONSTRAINT [PK\_VENDAS] PRIMARY KEY CLUSTERED

(

[CODIGO]

) ON [PRIMARY],

CONSTRAINT [FK\_Codigo\_Cliente] FOREIGN KEY

(

[CODIGOCLIENTE]

) REFERENCES [Clientes] (

[Codigo]

),

CONSTRAINT [FK\_Codigo\_Produto] FOREIGN KEY

(

[CODIGOPRODUTO]

) REFERENCES [Produtos] (

[Codigo]

)

) ON [PRIMARY]

GO

A tabela de Vendas que acabamos de criar possui duas foreign keys (chaves estrangeiras). Essas duas chaves são os campos: CodigoCliente e CodigoProduto.

Isso significa que quando um registro for inserido nesta tabela, o campo CodigoCliente incluído deverá existir na tabela de Clientes, assim como o campo CodigoProduto deverá existir na tabela Produtos.

Se alguém tentar incluir valores nestes campos que não existam nas tabelas citadas o Microsoft SQL Server vai informar a ocorrência de um erro e não permitirá a inclusão.

Esse mecanismo do sistema gerenciador de banco de dados existe para manter a integridade dos dados. Chamamos isso de integridade referencial.

Uma regra que vamos usar no nosso projeto é que ao realizar uma venda, atualizaremos o estoque do produto na tabela de Produtos. Faremos com que o estoque fique atualizado com a seguinte fórmula:

**Estoque = Estoque – Quantidade Vendida**

Essa tarefa será realizada juntamente com a tarefa de inclusão da Venda na tabela de Vendas.

Assim, para manter a integridade dos dados, precisaremos garantir que as duas coisas aconteçam:

1) a inclusão na tabela de Vendas; e

2) a atualização do estoque na tabela de Produtos.

Para isso usaremos uma Transação. Com a transação podemos garantir a execução das duas tarefas. Se uma delas for bem sucedida e a outra falhar, a transação desfaz a primeira tarefa mantendo assim a integridade referencial.

Em outras palavras, não exisitirá na tabela de Vendas um registro que tenha o código do produto vendido mas que o estoque deste produto na tabela de Produtos esteja desatualizado.

Veremos isso com mais detalhes no próximo artigo chamado “Camada de Acesso a Dados (Data Access Layer)”. Implementaremos as classes de acesso a dados e faremos esse controle transacional no método Incluir da classe de Vendas.

**OBSERVAÇÃO:**

Para futura utilização no nosso projeto, precisaremos guardar as seguintes informações sobre o banco de dados:

* Nome do servidor de banco de dados que estamos usando;
* Nome do banco de dados onde as três tabelas foram criadas;

Essa informação é importante para a construção da nossa string de conexão ou connectionstring.

A connectionstring será usada para a conexão com o banco de dados através da aplicação que desenvolveremos neste projeto.

“server=PC-Priscila\SQLEXPRESS;database=Loja;Integrated Security”

## Parte 3 – Camada de Acesso a Dados

Essa camada é normalmente também chamada de DAL (Data Access Layer). Nessa camada vamos implementar os métodos de inserção, atualização, exclusão e listagem referentes a todas as tabelas existentes no nosso projeto.

Essa é uma tarefa simples, já que para criar cada classe usaremos os nomes dos campos da respectiva tabela existente no banco de dados.

Começaremos o desenvolvimento da aplicação da camada mais interna até a mais externa. Dessa forma, iniciaremos a implementação com a camada de acesso a dados (DAL).

Dentro da camada DAL nós temos o projeto Modelos. Vamos criar esse projeto.

No projeto chamado Modelos iremos implementar as classes:

· Cliente.cs

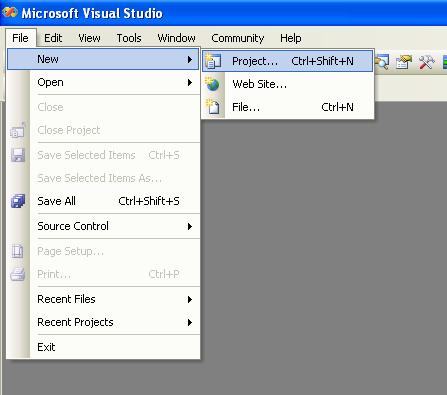
· Produto.cs e

· Venda.cs

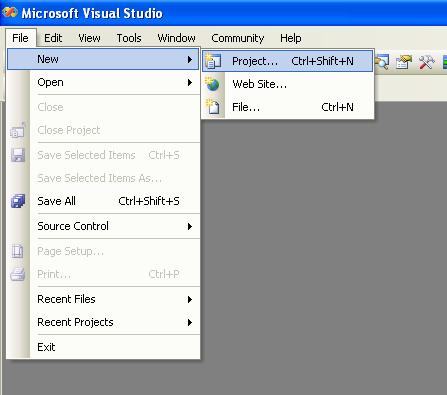
É necessário criar uma classe para cada tabela do nosso projeto.

A partir do menu **Iniciar > Programas > Microsoft Visual Studio xxxx**, abra o Microsoft Visual Studio xxxx.

Clique no Menu File > New > Project...



Selecione as opções conforme imagem da janela abaixo:



No tipo de projeto selecione **Visual C#**;

No tipo de template selecione **Class Library**;

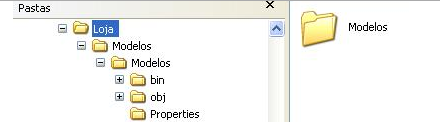
No Nome do projeto digite **Modelos**;

Na localização do projeto digite **C:\POO\Loja**;

Deixe a opção “**Create directory** for solution” selecionada e clique em Ok para criar o projeto.

Abrindo o Windows Explorer, podemos ver que a pasta Loja foi criada no drive **C:\POO**:

Dentro da pasta Loja foi criada a pasta do projeto Modelos.

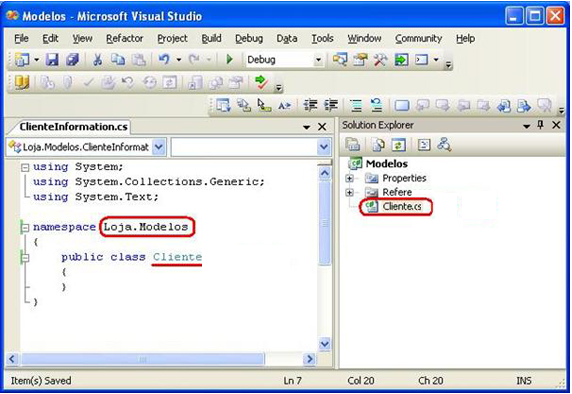


Do lado direito, do visual Studio, vemos a área do Solution Explorer.

- Clique com o botão direito sobre o arquivo Class1.cs e escolha Rename para renomear a classe para Cliente.cs;

Ao renomear o arquivo da classe, perceba que o nome da classe muda automaticamente na janela de código (public class Cliente).

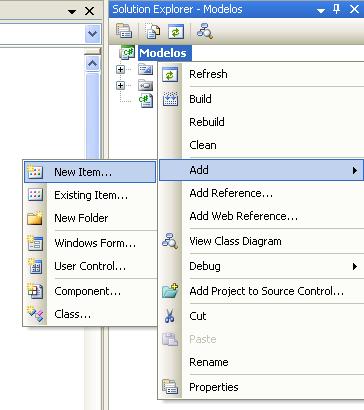
- Na janela de código à esquerda, inclua o nome do nosso projeto (Loja) no namespace, de maneira que o namespace fique Loja.Modelos como na figura abaixo:



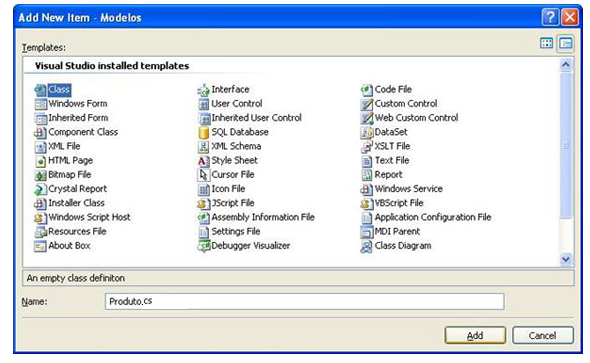
Agora codifique a classe cliente conforme imagem abaixo:

|  |
| --- |
|  |
|  |
|  |

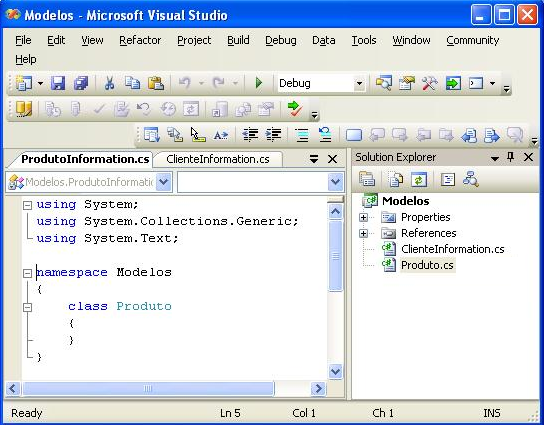
Para incrementar as demais classes, clique com o botão direito do mouse sobre o projeto Modelos e então escolha Add. Em seguida escolha New Item como na figura abaixo:



Na janela **Add New Item**, escolha o **template Class** e digite **Produto.cs** no campo **name**. Clique em **Add** para adicionar a nova classe ao projeto Modelos.



A classe está criada conforme imagem abaixo:



Os comandos echo ou print servem para escrever algo na tela. Basta utilizar o comando abaixo dentro da tag PHP:

Altere o **namespace** de modo que ele fique assim:

namespace Loja.Modelos

Feito isto, agora codifique a classe Produto conforme imagem abaixo:

|  |
| --- |
|  |
|  |
|  |

Legal! Já implementamos as classes do projeto Modelo para a tabela de Clientes e para a tabela de Produtos. Agora vamos fazer o mesmo para a tabela de Vendas:

Clique com o botão direito sobre o projeto **Modelos** e então escolha **Add**. Em seguida escolha **New Item...** como na figura abaixo:

Altere o namespace de modo que ele fique assim:

namespace Loja.Modelos

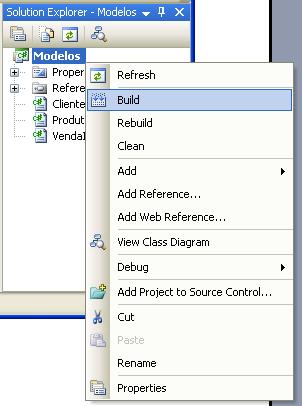
Agora digite o código abaixo dentro da classe Venda:

Para incrementar as demais classes, clique com o botão direito do mouse sobre o projeto Modelos e então escolha Add. Em seguida escolha New Item como na figura abaixo:

|  |
| --- |
|  |
|  |
|  |
|  |

Agora que implementamos no projeto Modelos as classes referentes a todas as tabelas contempladas no nosso projeto, vamos compilar o projeto através da opção **Build**.

Para compilar o projeto vamos fazer o seguinte. No Solution Explorer, clique com o botão direito sobre o projeto **Modelos** e escolha a opção **Build**:

  
  
***Se a compilação do nosso projeto for realizada com sucesso, aparecerá a mensagem “Build succeeded” na barra de status do Microsoft Visual Studio:***



Quando compilamos o projeto com a opção Build, o MS Visual Studio criou o arquivo Modelos.dll.

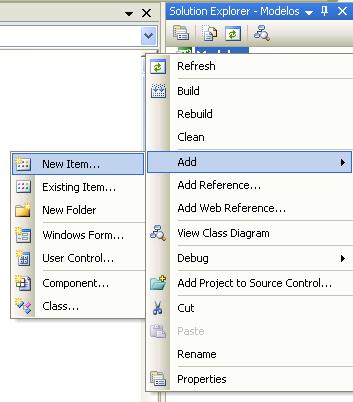
O arquivo Modelos.dll contém toda a informação que implementamos nas classes desse projeto e é ele que será usado no próximo projeto a ser implementado, que é a Camada de Acesso a Dados ou DAL (Data Access Layer).



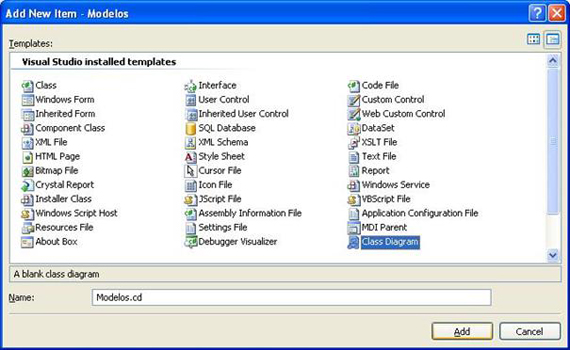
Agora que terminamos o projeto Modelos, vamos criar um Diagrama de Classes para ter uma visão melhor da nossa arte.

Para incrementar as demais classes, clique com o botão direito do mouse sobre o projeto Modelos e então escolha Add. Em seguida escolha New Item como na figura abaixo:

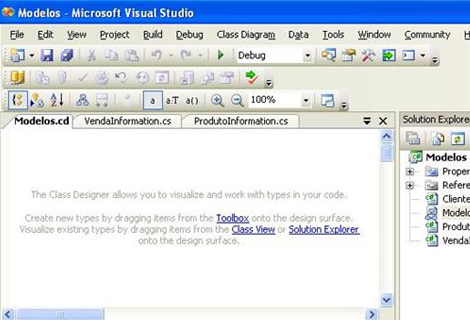
Clique com o botão direito sobre o projeto Modelos e escolha **Add** > **New Item...**

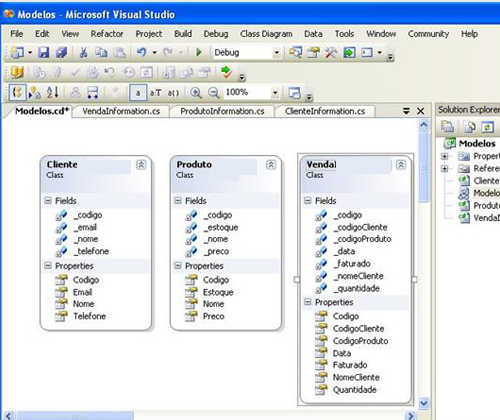


Na janela Add New Item, escolha o **template Class Diagram**. Digite **Modelos.cd** no nome e clique em **Add** para criar o nosso diagrama de classes como a seguir.



O ambiente chamado class designer é exibido.



Para a criação do diagrama vamos arrastar as classes Cliente, Produto e Venda do Solution Explorer para o Class Designer (Arquivo Modelos.cd) como a seguir:  
  


Com o Diagrama de Classes fica fácil a distinção entre os campos e as propriedades que implementamos no nosso projeto.

O asterisco ao lado do nome do diagrama Modelos.cd\* indica que ele ainda não foi salvo. Digite <Ctrl> + S para salvar o arquivo.

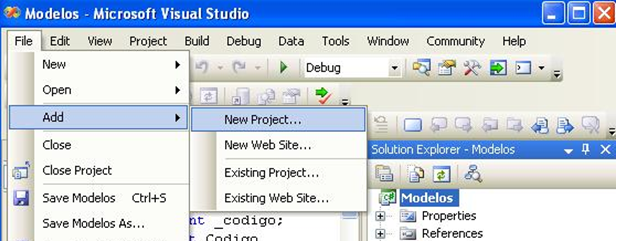
O próximo passo é implementar as classes da camada DAL. Faremos isso no próximo artigo.

## Parte 4 – Implementação da Camada da DAL

Nesta etapa criaremos as classes abaixo:

* ClientesDAL.cs → Conterá os métodos de inclusão, alteração, exclusão e consulta referente a tabela de clientes;
* ProdutosDAL.cs → Conterá os métodos de inclusão, alteração, exclusão e consulta referente a tabela de produtos;
* VendasDAL.cs → Conterá os métodos de inclusão, alteração, exclusão e consulta referente a tabela de vendas;
* Dados.cs → Será utilizada para armazenamento da string de conexão com o banco de dados;

1. Abra o Microsoft Visual Studio;
2. Abra o projeto **Modelos** que criamos no artigo anterior.
3. Vamos adicionar um novo projeto. No menu **File** do Visual Studio, clique em **Add** > **New** **Project**.



1. Na janela Add New Project:

- Escolha Visual C# para o tipo de projeto;

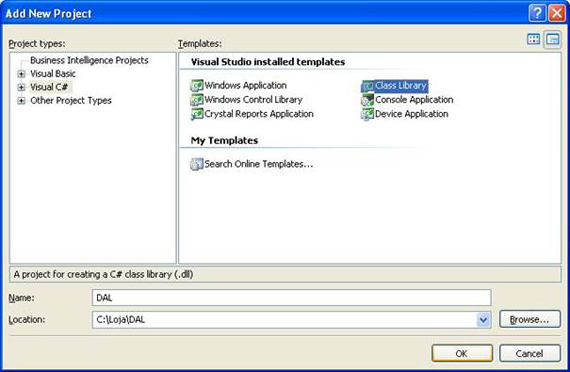
- Class Library para o template;

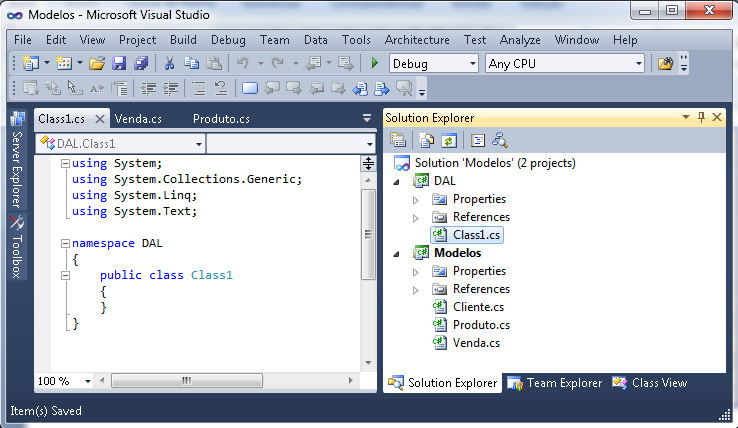
- Digite DAL no campo nome;

- Digite C:\Loja\DAL na localização;

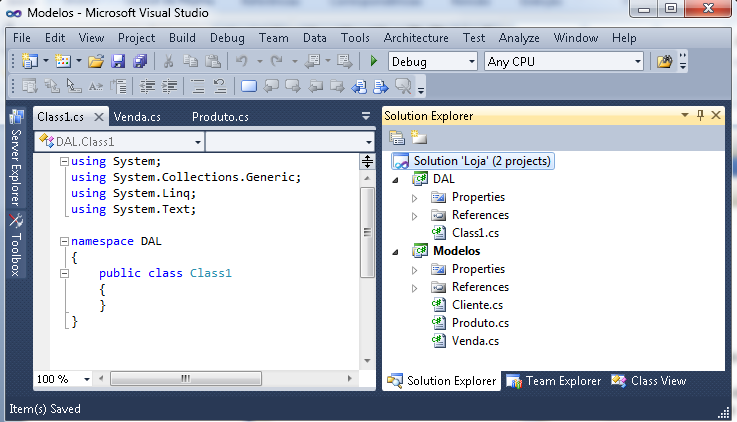
- Clique em Ok para adicionar o novo projeto à nossa Solution (solução).

1. Nota: Quando temos um conjunto de projetos reunidos para atender uma necessidade chamamos esse conjunto de Solution.

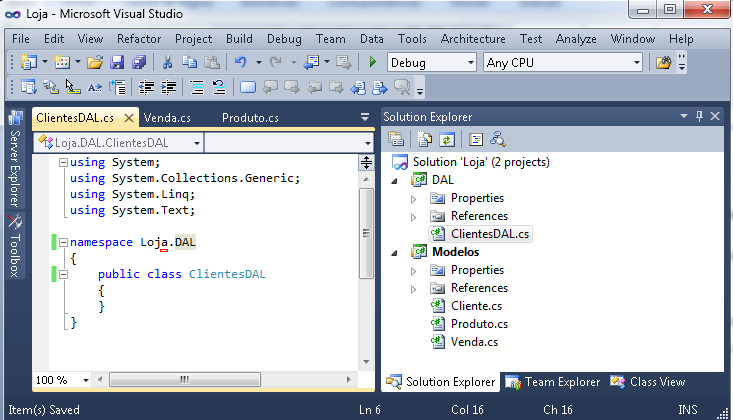




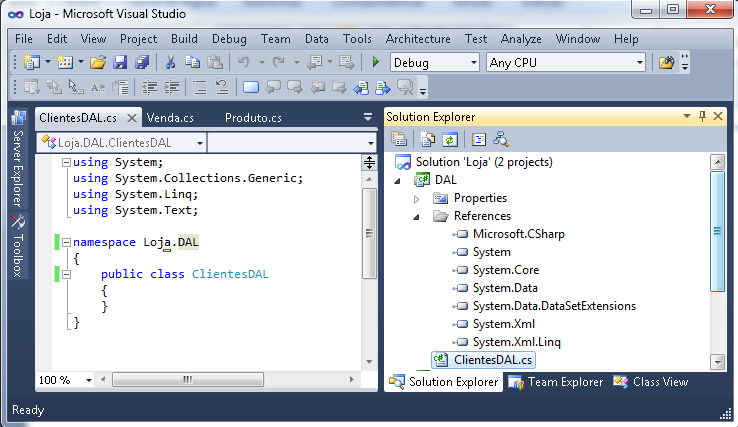
1. O nome da nossa Solution está como Modelos porque o MS Visual Studio usa o nome do primeiro projeto criado.
2. No Solution Explorer, clique com o botão direito sobre a Solution ‘Modelos’ e escolha a opção Rename. Renomeie a Solution para Loja.
3. Depois que você renomear a nossa Solution, ela ficará assim:



1. No projeto DAL, Renomeie a classe Class1.cs para ClientesDAL.cs. Em seguida altere o namespace para Loja.DAL. A nossa classe ClientesDAL ficará assim:

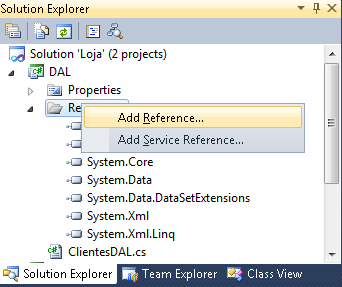


1. Como a implementação acontecerá da camada mais interna para a mais externa do desenho do nosso projeto. Ex: [Modelos] DAL BLL User Interface. Então, o projeto Modelos é o mais interno e o próximo é a camada de acesso a dados DAL. Isso significa que o nosso projeto DAL poderá usar tudo o que já foi construído no projeto Modelos!
2. Em termos de programação, dizemos que o projeto DAL faz referência ao projeto Modelos. Para que o nosso projeto DAL consiga ver o que já construímos no projeto Modelos, vamos criar essa referência.
3. Abra a pasta References do projeto DAL para ver o que já existe como referência sempre que criamos um novo projeto:

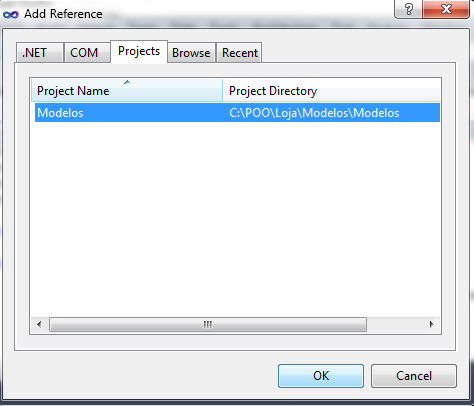


1. Vamos incluir no projeto DAL uma referência para o Modelos. Faça o seguinte:

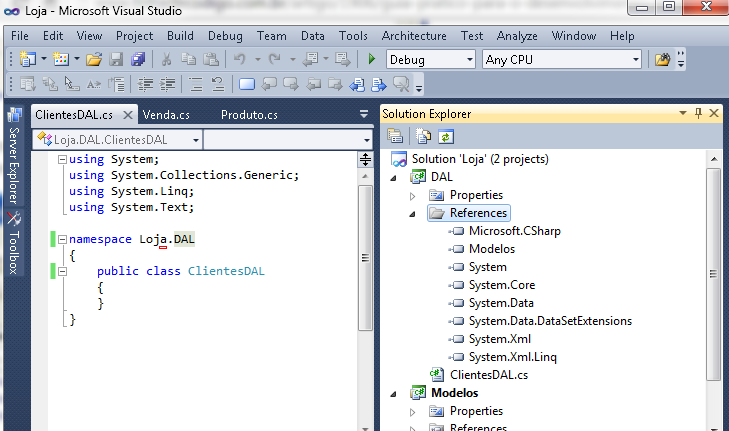
– Clique com o botão direito na pasta References do projeto DAL e escolha Add Reference...



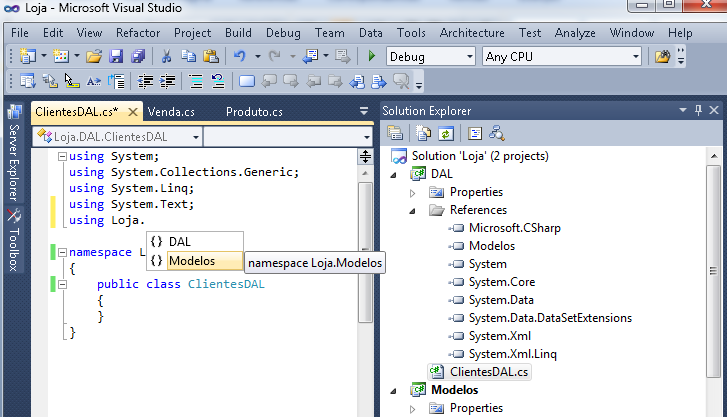
1. Na janela **Add Reference** que abrirá, clique na aba **Projects**:



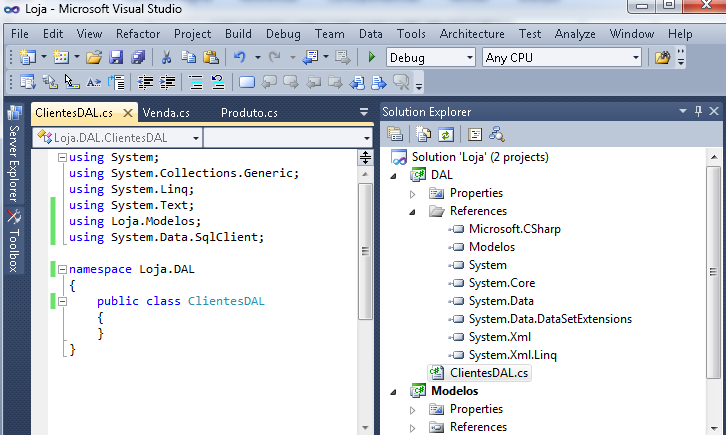
1. Conforme imagem abaixo podemos visualizar nosso projeto modelo dentro do DAL



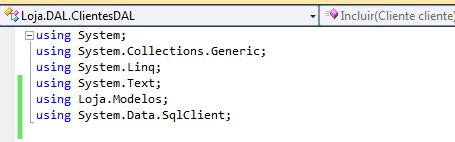
1. Para enxergar os Modelos dentro da nossa classe ClientesDAL vamos adicionar a cláusula using conforme imagem abaixo.



1. Após digitarmos using **Loja**, percebemos que o **MS** **Visual Studio** já nos mostra as referências disponíveis. Vamos escolher **Modelos**.
2. Para que tenhamos acesso as definições do namespace do MS SQL Server, vamos inserir mais uma cláusula using para System.Data.SqlClient como a seguir:



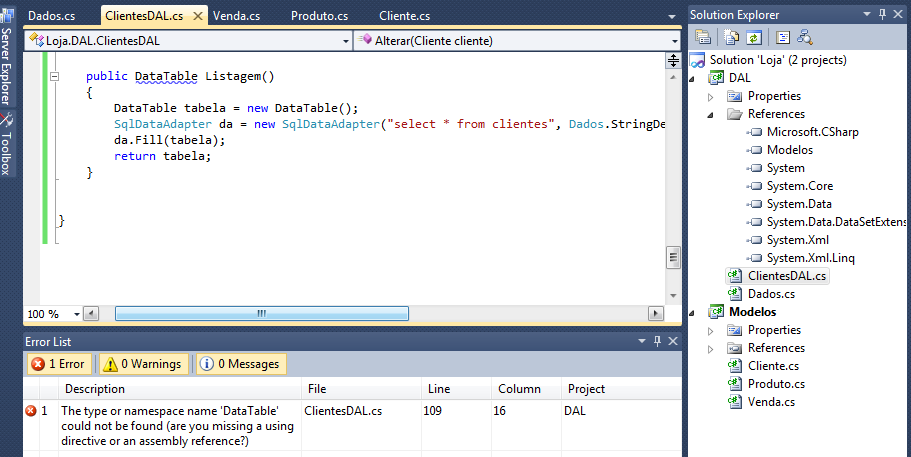
1. Após digitarmos using **Loja**, percebemos que o **MS** **Visual Studio** já nos mostra as referências disponíveis. Vamos escolher **Modelos**.
2. Para que tenhamos acesso as definições do namespace do MS SQL Server, vamos inserir mais uma cláusula using para System.Data.SqlClient como a seguir:



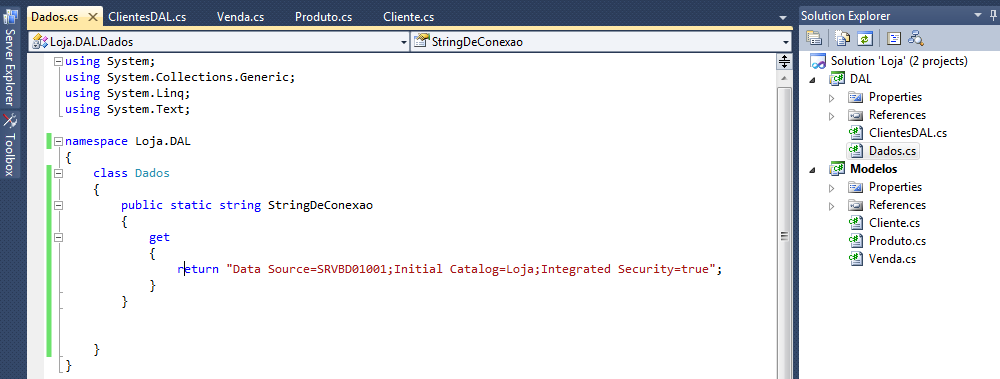
1. Faça o código da classe ClienteDALL, conforme imagem abaixo:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

1. Vamos dar um **Build** no projeto para nos certificarmos de que está tudo certo. Clique com o botão direito sobre o projeto **DAL** e escolha **BUILD**. Observe na imagem abaixo que um erro foi encontrado:



1. Na descrição do erro diz que o “***DataTable***” não foi encontrado. Isso ocorreu porque o tipo ***DataTable*** está dentro do **namespace** **System.Data** e nós não chamamos com um using. Então vamos incluir a cláusula using no início da classe. Feito isto vamos dar um build novamente e outros quatro erros aparecerão. A descrição diz que o nome Dados não foi encontrado. Então vamos codificá-las:
2. Clique com o botão direito no projeto DAL e selecione Add > New Item.. Na janela que se abrirá, vamos escolher o template Class e dar o nome de Dados.cs. Clique no botão Add para adicionar a nova classe ao projeto. Em seguida altere o namespace para **Loja.DAL.** Agora acrescente o código conforme imagem abaixo:



**Data Source**: O nome do seu servidor de banco de dados;

**Initial Catalog**: O nome do seu banco de dados;

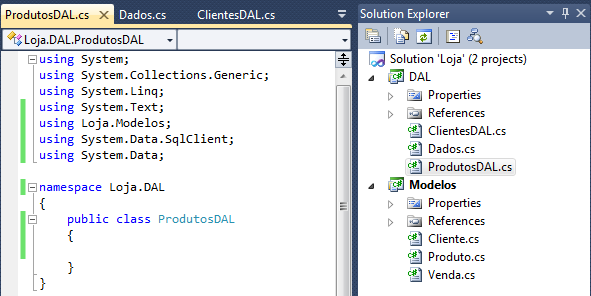
**Integrated Security=true**: Deverá ser colocado apenas se sua conexão com o banco utilizar autenticação do Windows. Senão use:

user=nome do usuário;pwd=senha de acesso ao banco";

Prontinho! Fino e chique no úrtimo! RS RS RS

Agora vamos montar a classe **ProdutosDAL.**

1. Adicione mais uma classe ao projeto DAL de nome ProdutosDAL. Faça a inclusão das using Modelos, System.Data e System.Data.SqlCliente;
2. Em seguida faça os ajustes no namespace para que fique conforme imagem abaixo:



1. Baseado na classe ClientesDAL, codifique a classe ProdutosDAL.cs
2. Código da Classe VendasDAL

|  |
| --- |
|  |
|  |
|  |

1. Código da Classe ProdutosDAL

|  |
| --- |
|  |
|  |
|  |

## Parte 5 – Implementação da Camada da BLL

Nesta parte vamos implementar as classes da Camada BLL (Business Logic Layer).

Criaremos as classes:

* + ClientesBLL.cs
  + ProdutosBLL.cs
  + VendasBLL.cs

As regras de negócio definem como o seu negócio funciona. Essas regras podem abranger diversos assuntos como suas políticas, interesses, objetivos, compromissos éticos e sociais, obrigações contratuais, decisões estratégicas, leis e regulamentações, entre outros.

No nosso projeto Loja vamos definir como regras de negócio:

**1) Regras para a inclusão de clientes:**

1.1) O nome do cliente é obrigatório;

1.2) O e-mail do cliente será armazenado em letras minúsculas;

**2) Regras para a inclusão de produtos:**

2.1) O nome do produto é obrigatório;

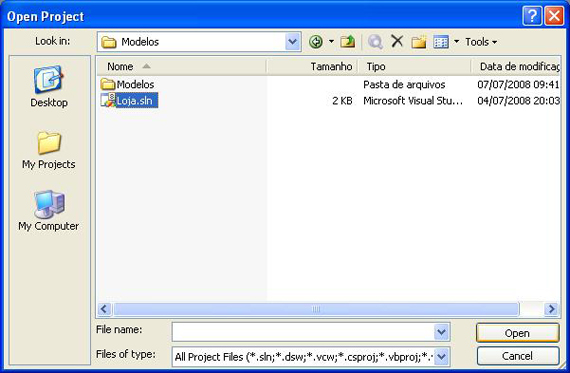
2.2) O preço não pode ser um valor negativo;

2.3) O estoque não pode ser um valor negativo;

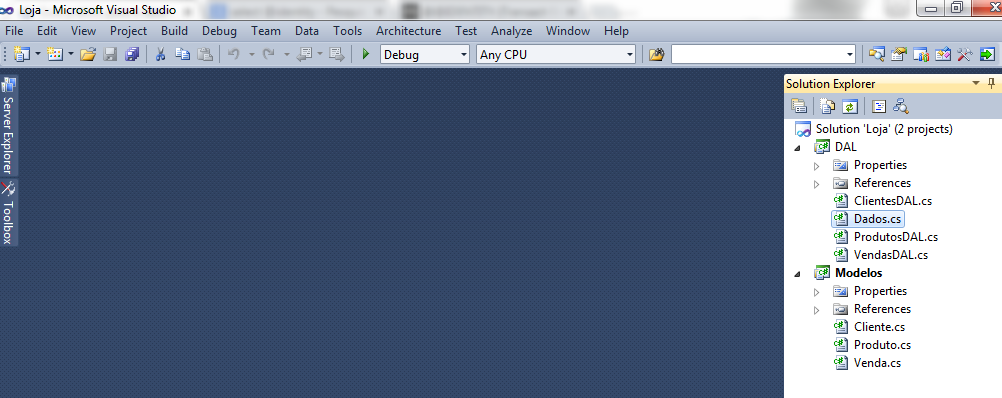
- Abra o Microsoft Visual Studio;

- 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:



Até este ponto do projeto temos implementados os projetos Modelos e DAL:

  
Vamos adicionar a solução o projeto BLL.

1. Clique com o botão direito na Solution Loja e escolha Add > New Project...
2. Na janela “Add New Project” vamos informar os seguintes dados:

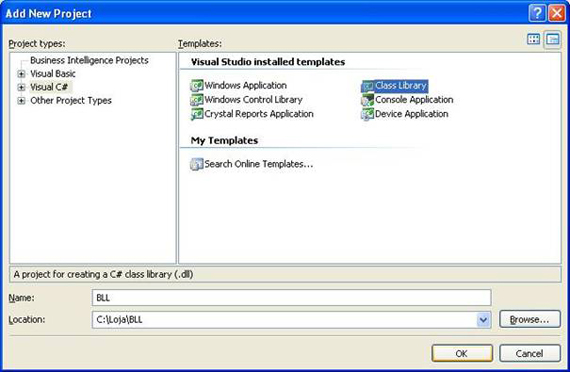
            Project type: Visual C#

            Template: Class Library

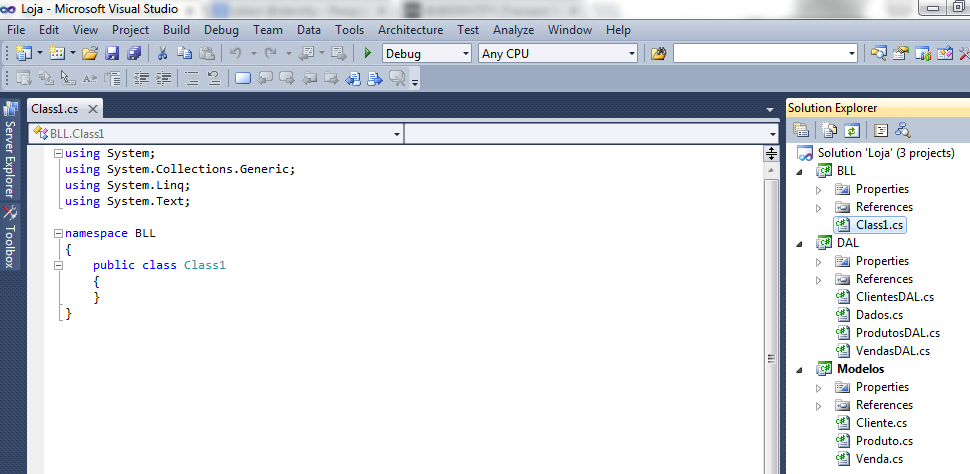
            Name: BLL

            Location: C:\Loja\BLL

1. Clique em Ok e o projeto será adicionado.

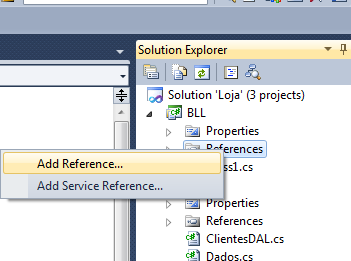


1. O projeto BLL será criado:



Analisando o nosso modelo em camadas podemos ver que a camada BLL aproveita os projetos Modelos e DAL. Sendo assim, a primeira coisa que faremos no nosso projeto BLL será adicionar esses dois projetos como referência.

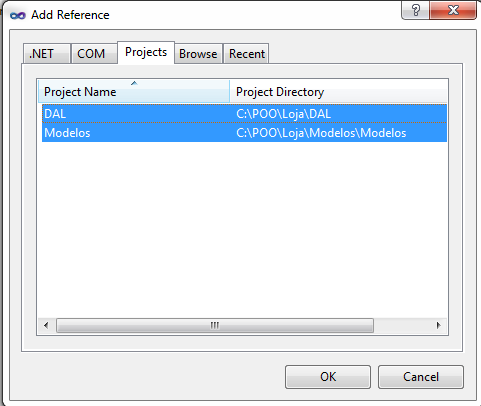
- Clique com o botão direito na pasta References do projeto BLL e escolha Add Reference...



- Na janela “Add Reference”, clique na aba Projects;

- Mantenha a tecla <CTRL> pressionada e clique sobre os projetos DAL e Modelos para selecioná-los;

- Clique em Ok para adicionar as referências.



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



Vamos renomear a classe Class1.cs para ClientesBLL.cs.

- Clique com o botão direito sobre a classe Class1.cs, escolha a opção Rename e renomeie a classe para ClientesBLL.cs:



Código da Classe ClienteBLL:

|  |
| --- |
|  |
|  |
|  |

Dê um Build no projeto BLL para confirmar a inexistência de erros na codificação.

Agora adicione uma nova classe ao projeto BLL de nome ProdutosBLL.cs e codifique conforme imagem abaixo:

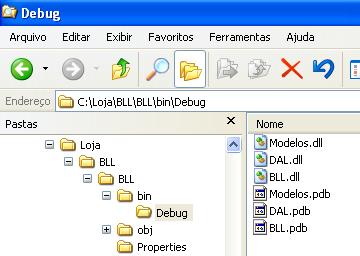
|  |
| --- |
|  |
|  |
|  |
|  |

Agora vamos implementar a classe VendasBLL.cs. Crie a classe mencionada e codifique conforme imagem abaixo:

|  |
| --- |
|  |
|  |
|  |
|  |

Dê um Build no projeto BLL para confirmar a inexistência de erros na codificação.

Se você abrir o projeto BLL no Windows Explorer, poderá verificar que na pasta de DLLs do projeto existem três arquivos com a extensão DLL. Um do próprio projeto BLL e outros para as referências que fizemos aos projetos Modelos e DAL.



A próxima camada a ser implementada é a interface com o usuário. No próximo artigo vamos construir um aplicativo para Microsoft Windows (também chamado de Desktop application) que vai utilizar todas as camadas já desenvolvidas.

## Parte 6 – Criando a Interface

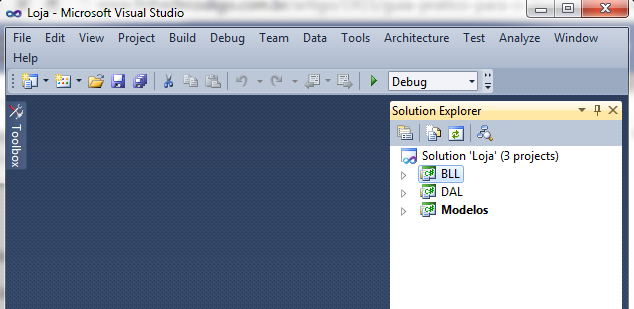
Nesta parte vamos implementar a camada de interface do usuário construindo um aplicativo para Microsoft Windows. Vamos criar o Formulário de Clientes utilizando todas as camadas já desenvolvidas nos artigos anteriores.

* + - Modelos DAL BLL 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 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:

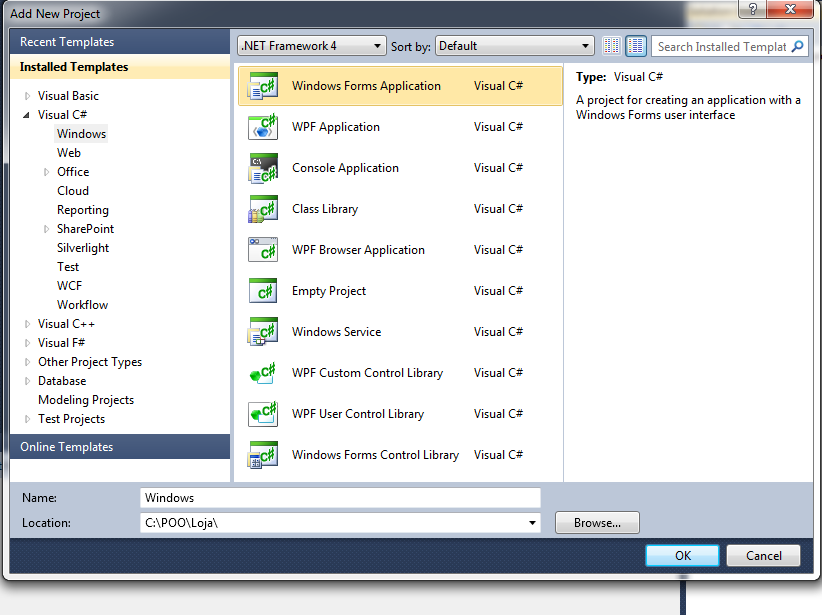
            Project type: Visual C#

            Template: Windows Application

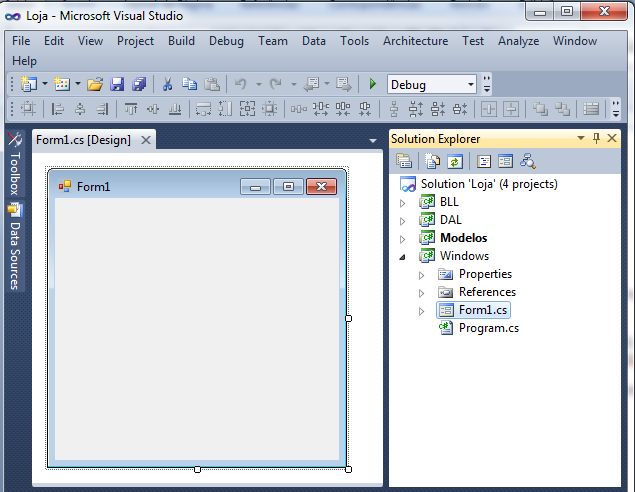
            Name: Windows

            Location: C:\Loja\Windows

- Clique em Ok para adicionar o projeto.



O aplicativo Windows 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**

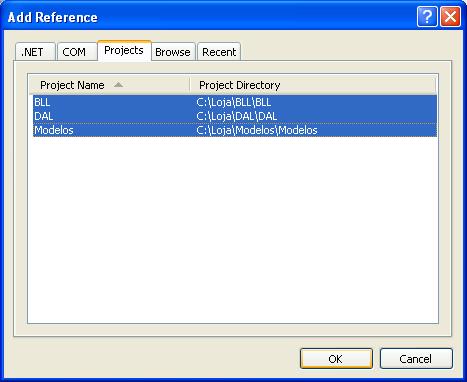
Verificamos que é preciso adicionar referência aos projetos Modelos, DAL e BLL. Vamos fazer isso agora.

- Clique com o botão direito na pasta References do projeto BLL e escolha Add Reference...

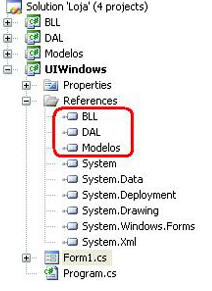
- Na janela “Add Reference”, clique na aba Projects;

- 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 adicionadas na janela solution Explorer:



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.

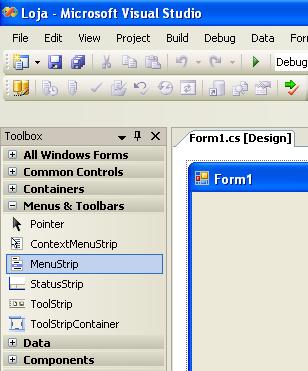
Para exibir a Toolbox você pode:

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

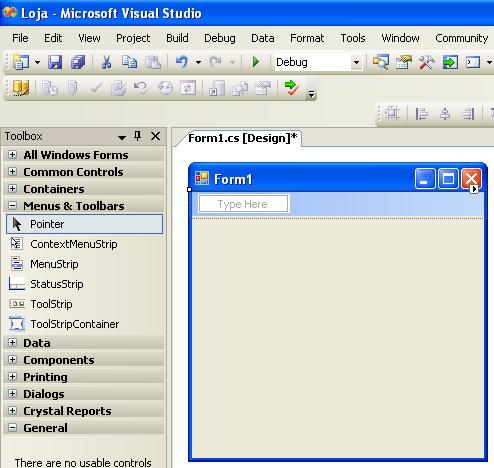
2) Utilizar o menu View > Toolbox

- Na Toolbox, abra o item “Menus & Toolbars”;

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

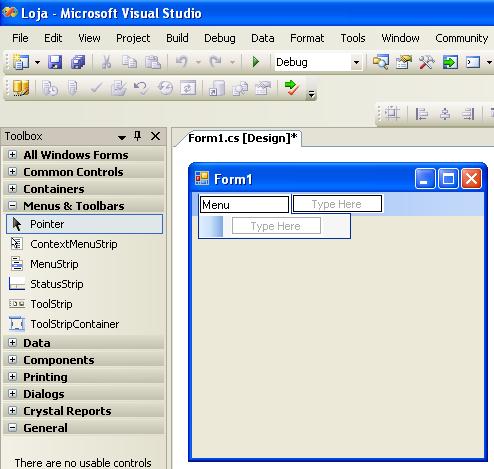


Ao soltar o **MenuStrip** dentro do formulário Form1, um “menu vazio” foi criado.



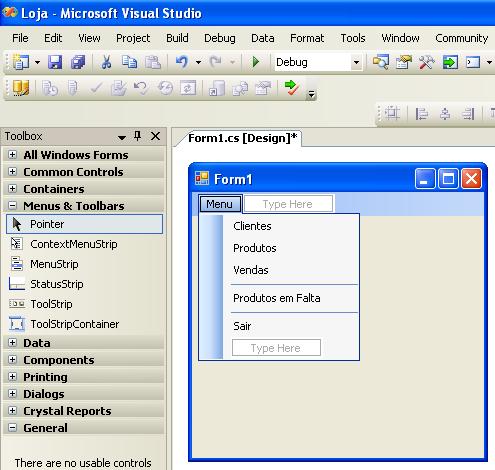
- Clique na área com a indicação “**Type here**” e digite “**Menu**”;

Veja que áreas se abrem ao lado e abaixo no caso de querermos 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.



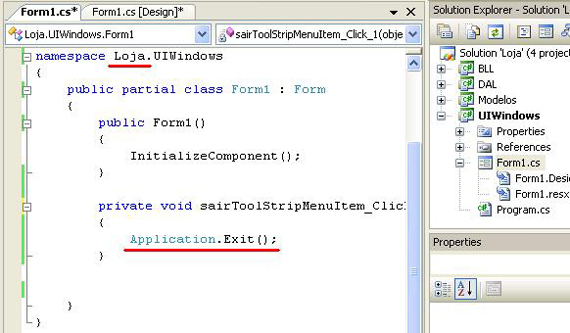
- Dê um duplo clique com o botão esquerdo do mouse sobre o item Sair do Menu para codificarmos o que deve acontecer quando o usuário selecionar essa opção.

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

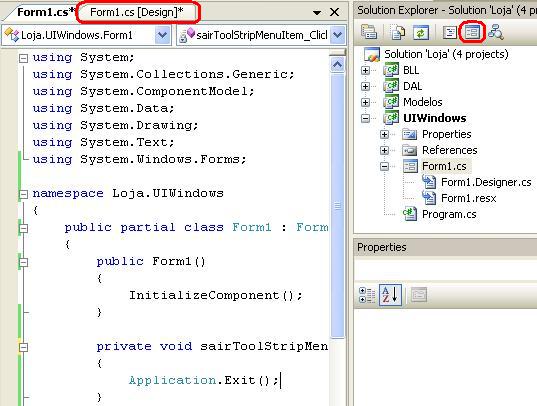
Application.Exit();

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:



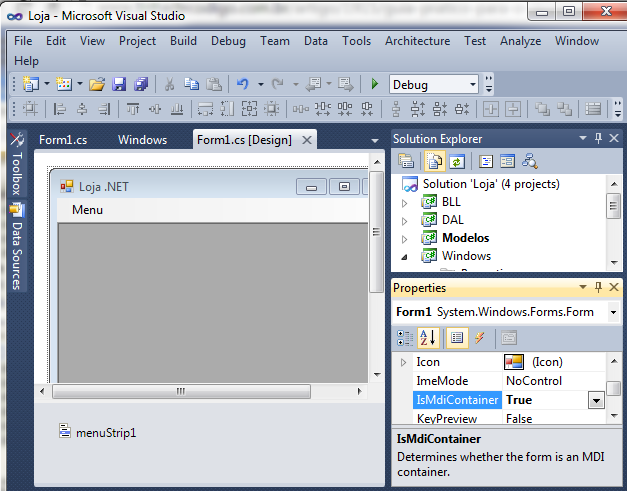
No modo de Design, é o componente de Menu que está selecionado. Por esse motivo, na janela de propriedades (**Properties**) no canto inferior direito do MS Visual Studio, são as propriedades do objeto de **Menu** que estão sendo exibidas.

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**;

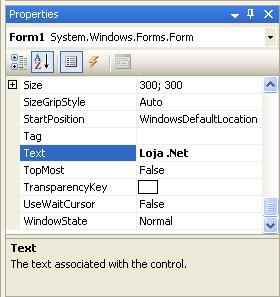


Isso significa que este formulário será o Menu Principal da nossa aplicação. Essa alteração indica que todos os formulários chamados pelo Form1 serão abertos dentro dele, e não em uma nova janela.

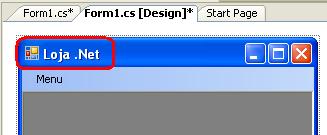
Assim, se fecharmos o formulário to Menu Principal todos os formulários que estiverem abertos dentro da aplicação (exemplo: formulário de clientes, formulário de produtos) também serão fechados.

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 formulario exibido na barra de títulos.

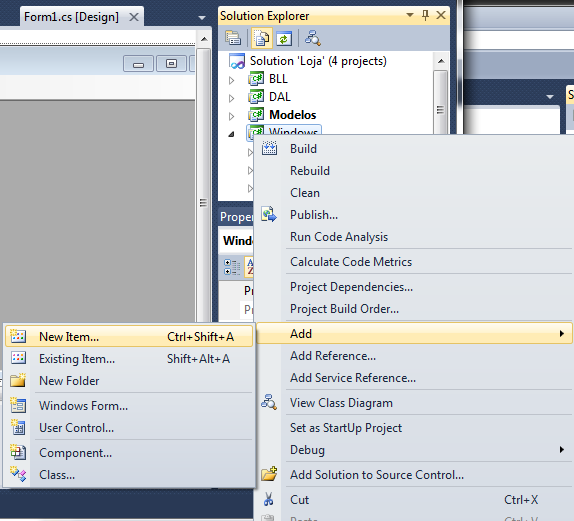


Para testarmos tudo o que já desenvolvemos até agora vamos criar 4 formulários. São eles:

* + Clientes;
  + Produtos;
  + Vendas;
  + Produtos em Falta.

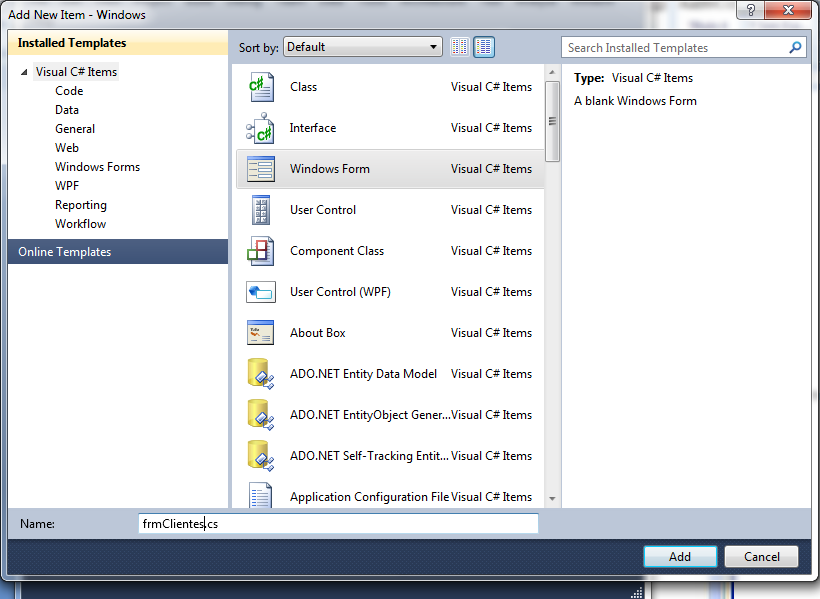
6.1) Criando o formulário de Clientes

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



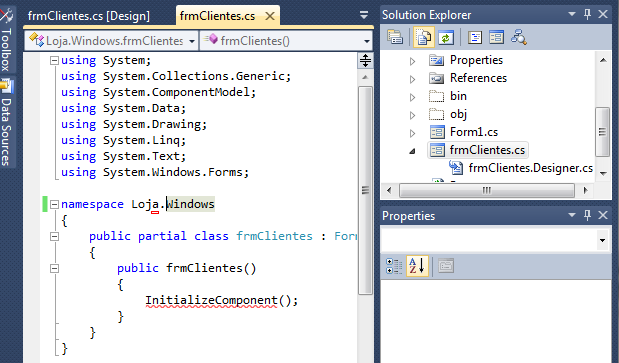
- Na janela “Add New Item”, selecione o template Windows Form e nomeie o novo formulário como frmClientes.cs

- Clique em Add para adicionar o formulário.



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

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



- Na janela de propriedades do formulário **frmClientes**, 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 **frmClientes**.

- Defina as propriedades dos 4 labels como a seguir:

|  |
| --- |
| Name: **lblcodigo** |
| Location: **16;32** |
| Text: **Código:** |

|  |
| --- |
| Name: **lblnome** |
| Location: **16;64** |
| Text: **Nome:** |

|  |
| --- |
| Name: **lblemail** |
| Location: **16;96** |
| Text: **E-mail:** |

|  |
| --- |
| Name: **lbltelefone** |
| 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: txtcodigo |
| Enable: False |
| Location: 110;29 |
| ReadOnly: True |
| Size: 70;20 |

|  |
| --- |
| Name: txtnome |
| Location: 110;61 |
| Size: 200;20 |

|  |
| --- |
| Name: txtemail |
| Location: 110;93 |
| Size: 200;20 |
| Name: txttelefone |
| 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 frmClientes.

- 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 |

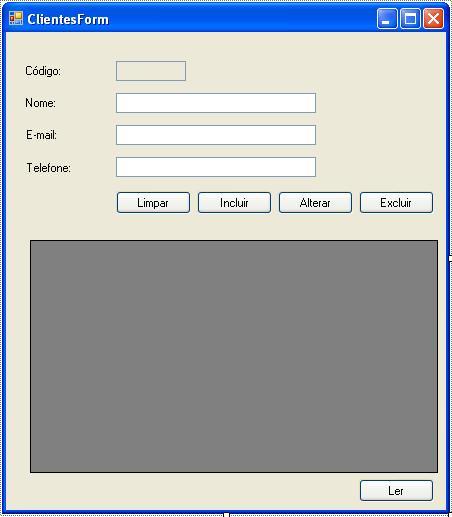
|  |
| --- |
| Name: lerButton |
| Anchor: Bottom, Right |
| Location: 353; 447 |
| Size: 75;23 |
| Text: Ler |

- A partir do menu “Data” da Toolbox, arraste um objeto do tipo DataGridView para o formulário ClientesForm.

- 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 frmClientes no Solution Explorer e escolha a opção “View Code”.

Após o método **ClientesForm\_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 **ClientesForm\_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.

|  |
| --- |
|  |
|  |
|  |

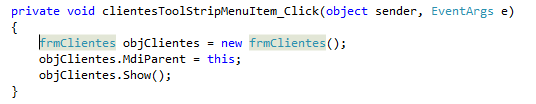
Para testarmos se essa exibição inicial dos dados dos clientes está funcionando, precisamos criar uma chamada para este formulário a partir do menu principal.

- **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.

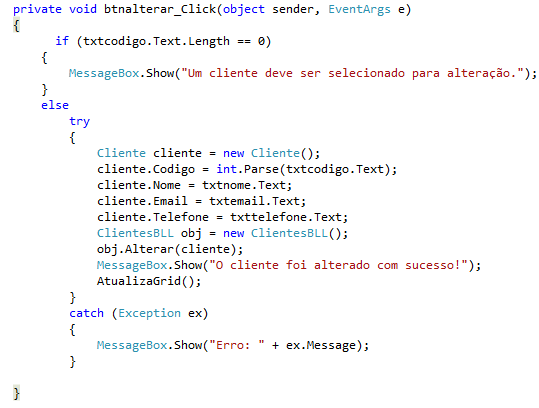
- O MS Visual Studio abrirá o código do formulário com o cursor dentro do método que será executado quando o usuário clicar no item Clientes do Menu. Dentro deste método (chamado *clientesToolStripMenuItem\_Click*) digite o seguinte código:



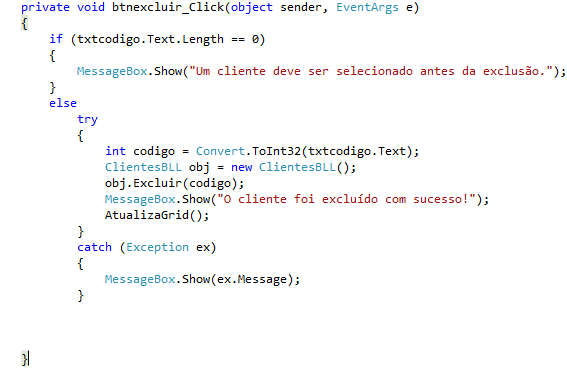
**Código do botão Incluir:**



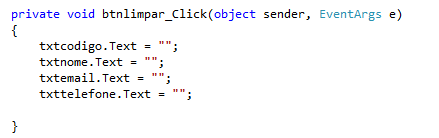
**Código do botão Alterar:**



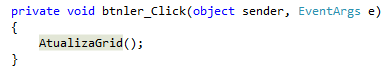
**Código do botão Excluir:**

****

**Código do botão Limpar:**



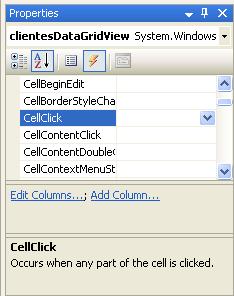
**Código do botão Ler:**

****

- Clique sobre o **dataGridView**;

- Na janela de propriedades, clique no ícone que tem um relâmpago ( http://www.linhadecodigo.com.br/artigos/img_artigos/CarlosCamacho/Camadas6/image024.jpg ) para exibir a lista de eventos;

- Dê um duplo clique no evento **CellClick** para inserirmos o código para esse evento:



Programe o evento **click** da seguinte maneira:

