



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 3

Este é o terceiro 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



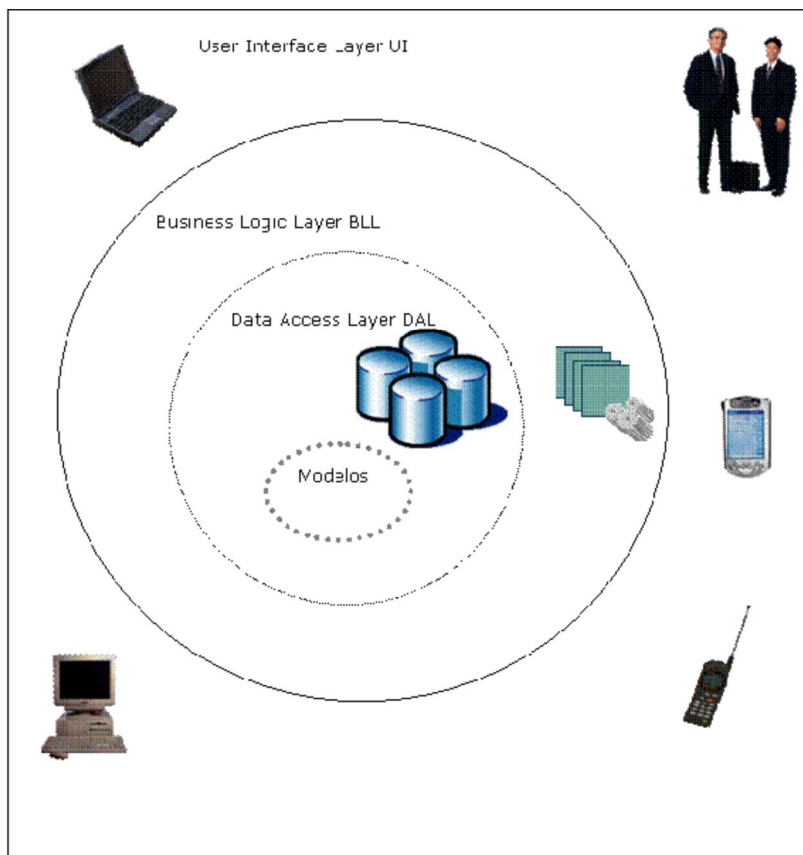
g+ 0

## 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.

Você se lembra do nosso desenho que representa as três camadas? Aqui está ele:



Nós começamos 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:

· ClienteInformation.cs



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 HTML Básico
- 2 Menu em CSS - Menu dropdown horizontal com HTML5 e CSS3
- 3 Comandos básicos em SQL - insert, update, delete e select
- 4 Criando um sistema de cadastro e login com PHP e MySQL
- 5 Código para background HTML e CSS
- 6 Copiando dados com o Robocopy
- 7 Trabalhando com Div em HTML
- 8 Excel: Como verificar se existe valores duplicados
- 9 Botão com CSS 3: Como criar um botão sem imagens
- 10 HTML Avançado

VER TODOS

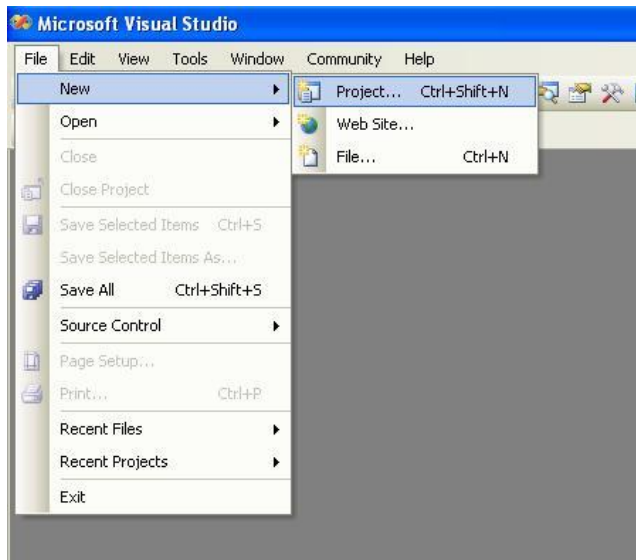
· ProdutoInformation.cs e

· VendaInformation.cs

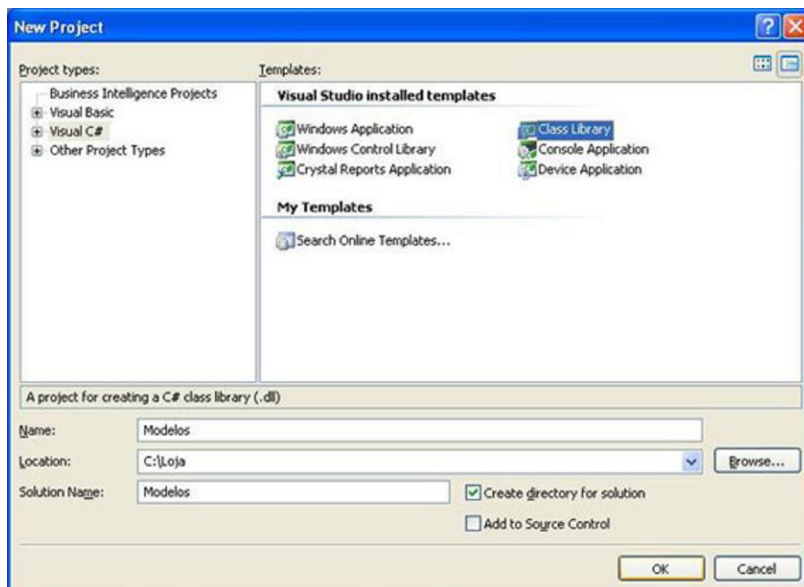
É necessário criar uma classe para cada tabela do nosso projeto.

A partir do menu Iniciar > Programas > Microsoft Visual Studio 2005, abra o **Microsoft Visual Studio 2005**.

Clique no Menu File > New > **Project...**



Acompanhe as seleções da janela *New Project*:



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

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

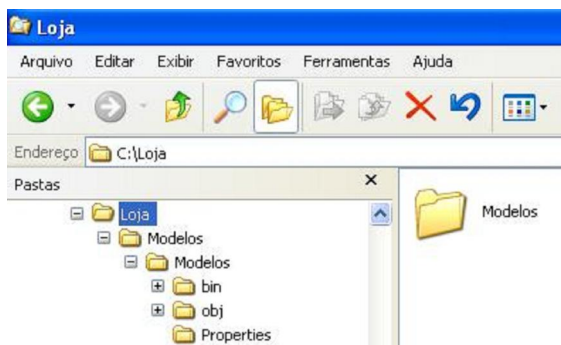
No Nome do projeto digite **Modelos**;

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

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

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

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



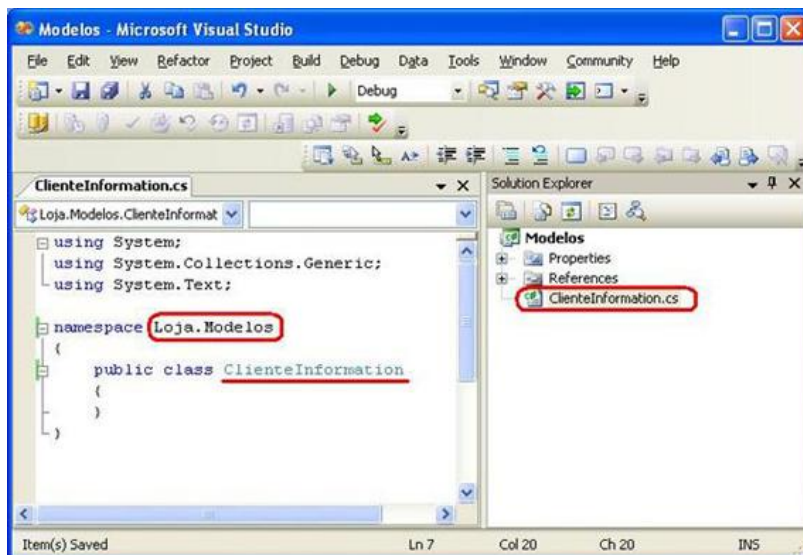
Ok, agora vamos voltar para o MS Visual Studio.

Do lado direito vemos a área do Solution Explorer.

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

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

- 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:



Vamos codificar a classe *ClienteInformation*. Para isso, copie e cole o código abaixo entre as chaves da classe *ClienteInformation*:

```
private int _codigo;

public int Codigo

{
    get { return _codigo; }

    set { _codigo = value; }
}

private string _nome;

public string Nome

{
    get { return _nome; }

    set { _nome = value; }
}
```

```
private string _email;

public string Email

{

    get { return _email; }

    set { _email = value; }

}

private string _telefone;

public string Telefone

{

    get { return _telefone; }

    set { _telefone = value; }

}
```

Na implementação da classe *ClienteInformation* vemos que estamos definindo campos e propriedades para cada campo da tabela de Clientes que criamos no banco de dados.

Agora a codificação da classe *ClienteInformation* está completa como na listagem a seguir:

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Loja.Modelos

{

    public class ClienteInformation

    {

        private int _codigo;

        public int Codigo

        {

            get { return _codigo; }

            set { _codigo = value; }

        }

        private string _nome;

        public string Nome

        {

            get { return _nome; }

            set { _nome = value; }

        }

        private string _email;

        public string Email

        {

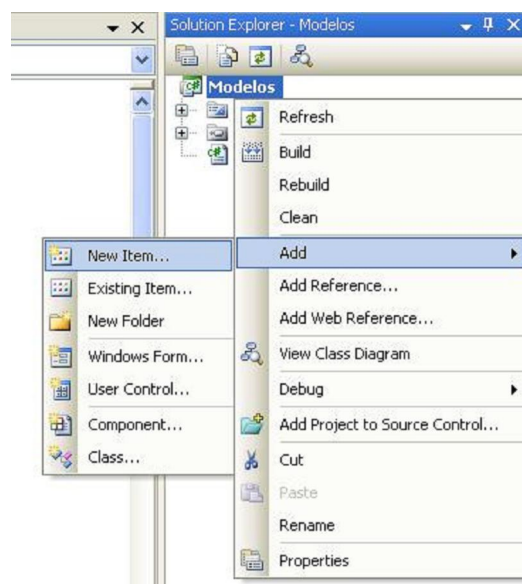
            get { return _email; }


```

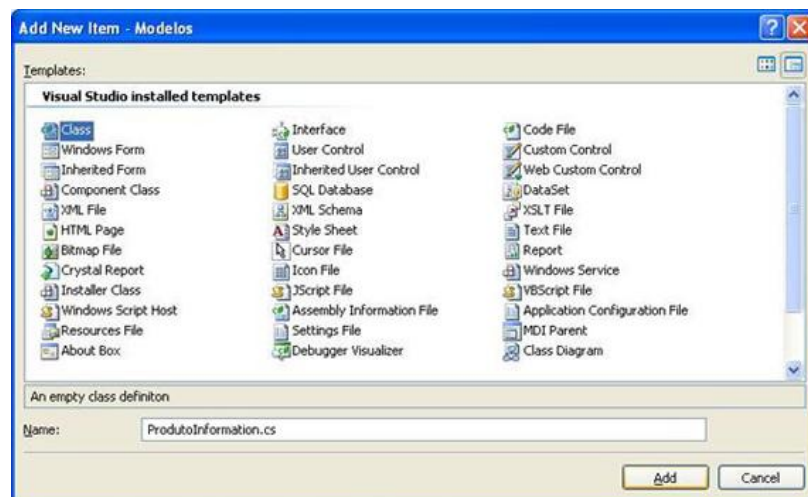
```
        set { _email = value; }  
  
    }  
  
    private string _telefone;  
  
    public string Telefone  
  
    {  
  
        get { return _telefone; }  
  
        set { _telefone = value; }  
  
    }  
  
}
```

Vamos implementar a Classe *ProdutoInformation.cs*.

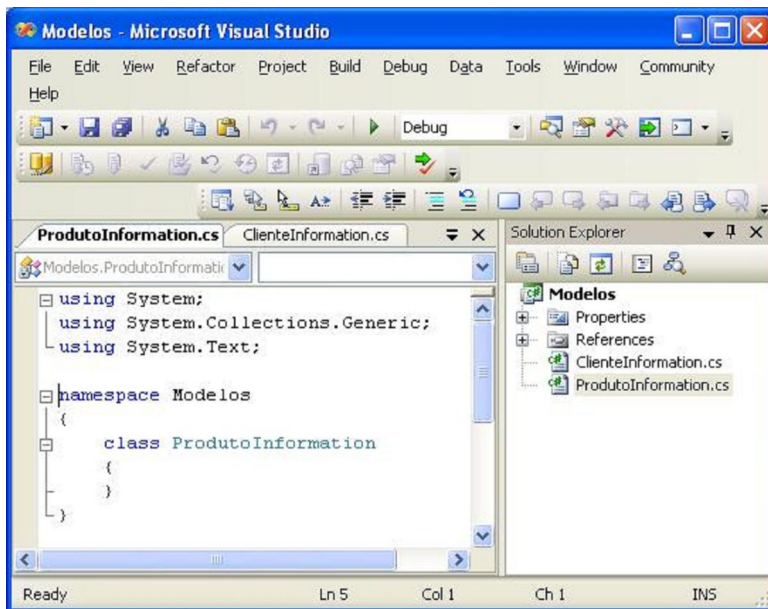
Clique com o botão direito sobre o projeto **Modelos** e então escolha **Add**. Em seguida escolha **New Item...** como na figura abaixo:



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



A nova classe *ProdutoInformation.cs* agora já esta criada:



Altere o namespace de modo que ele fique assim:

namespace Loja.Modelos

Ok. Agora copie e cole o código abaixo dentro da classe *ProdutoInformation*:

```
private int _codigo;

public int Codigo

{
    get { return _codigo; }
    set { _codigo = value; }
}

private string _nome;

public string Nome

{
    get { return _nome; }
    set { _nome = value; }
}

private decimal _preco;

public decimal Preco

{
    get { return _preco; }
    set { _preco = value; }
}

private int _estoque;

public int Estoque

{
    get { return _estoque; }
    set { _estoque = value; }
}
```

O código completo da nossa classe ProdutoInformation.cs ficará assim:

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Loja.Modelos

{

    public class ProdutoInformation

    {

        private int _codigo;

        public int Codigo

        {

            get { return _codigo; }

            set { _codigo = value; }

        }

        private string _nome;

        public string Nome

        {

            get { return _nome; }

            set { _nome = value; }

        }

        private decimal _preco;

        public decimal Preco

        {

            get { return _preco; }

            set { _preco = value; }

        }

        private int _estoque;

        public int Estoque

        {

            get { return _estoque; }

            set { _estoque = value; }

        }

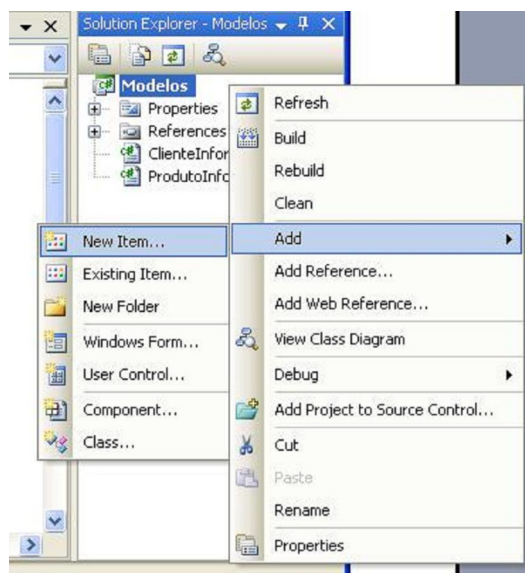
    }

}
```

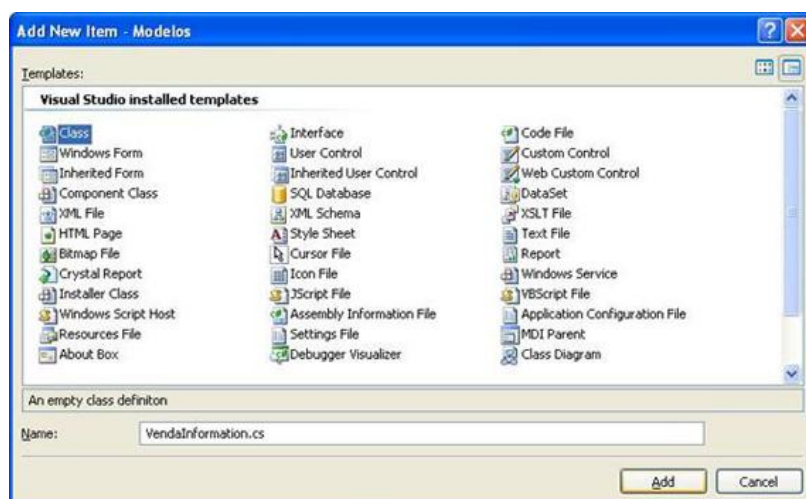
Muito bem! Já implementamos as classes do projeto Modelo para a tabela de Clientes e para a tabela de Produtos. Agora vamos fazer o mesmo para a tabela de Vendas:

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

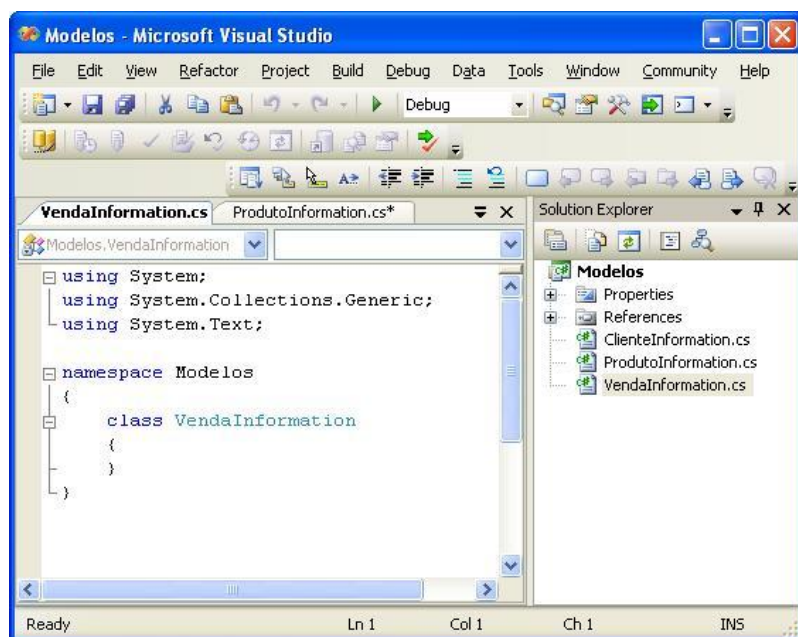




Na janela Add New Item, escolha o template **Class** e digite **VendaInformation.cs** no campo nome. Clique em **Add** para adicionar a nova classe ao projeto Modelos.



A nova classe VendaInformation.cs agora já esta criada:



Altere o namespace de modo que ele fique assim:

namespace Loja.Modelos

Ok. Agora copie e cole o código abaixo dentro da classe *VendaInformation*:



```
private int _codigo;

public int Codigo

{
    get { return _codigo; }
    set { _codigo = value; }
}

private DateTime _data;

public DateTime Data

{
    get { return _data; }
    set { _data = value; }
}

private int _quantidade;

public int Quantidade

{
    get { return _quantidade; }
    set { _quantidade = value; }
}

private bool _faturado;

public bool Faturado

{
    get { return _faturado; }
    set { _faturado = value; }
}

private int _codigoCliente;

public int CodigoCliente

{
    get { return _codigoCliente; }
    set { _codigoCliente = value; }
}

private int _codigoProduto;

public int CodigoProduto

{
    get { return _codigoProduto; }
    set { _codigoProduto = value; }
}

private string _nomeCliente;

public string NomeCliente

{
```

```
get { return _nomeCliente; }

set { _nomeCliente = value; }

}
```

O código completo da nossa classe VendaInformation.cs ficará assim:

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Loja.Modelos

{

    public class VendaInformation

    {

        private int _codigo;

        public int Codigo

        {

            get { return _codigo; }

            set { _codigo = value; }

        }

        private DateTime _data;

        public DateTime Data

        {

            get { return _data; }

            set { _data = value; }

        }

        private int _quantidade;

        public int Quantidade

        {

            get { return _quantidade; }

            set { _quantidade = value; }

        }

        private bool _faturado;

        public bool Faturado

        {

            get { return _faturado; }

            set { _faturado = value; }

        }

        private int _codigoCliente;

        public int CodigoCliente

        {

            get { return _codigoCliente; }
```

```
set { _codigoCliente = value; }

}

private int _codigoProduto;

public int CodigoProduto

{

get { return _codigoProduto; }

set { _codigoProduto = value; }

}

private string _nomeCliente;

public string NomeCliente

{

get { return _nomeCliente; }

set { _nomeCliente = value; }

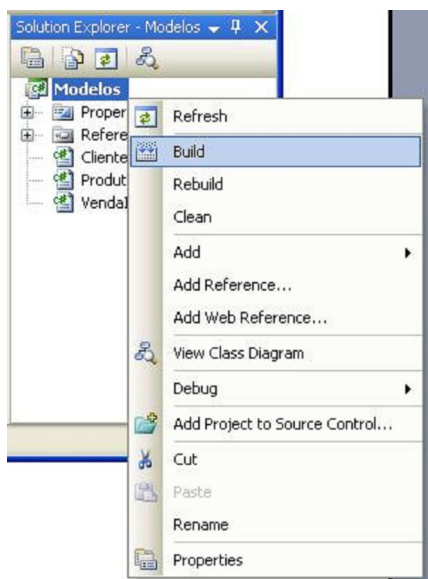
}

}

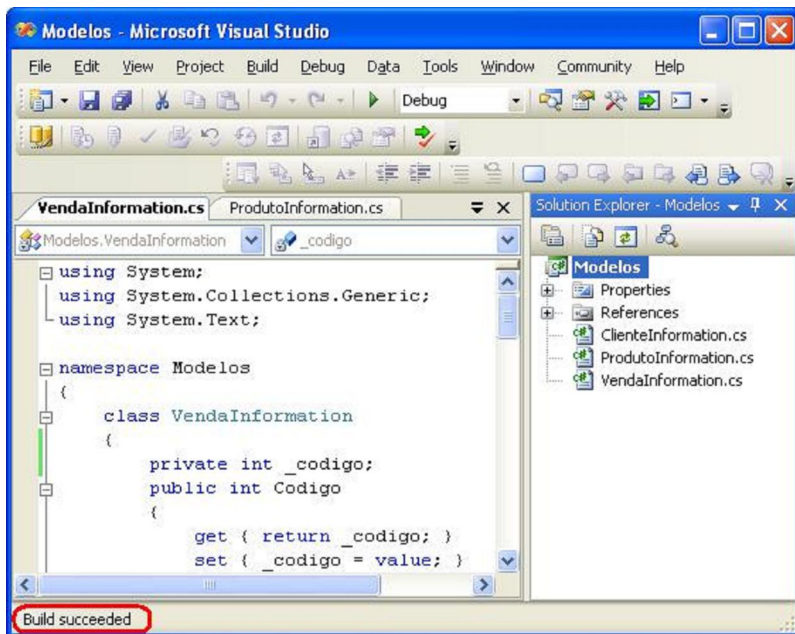
}
```

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.

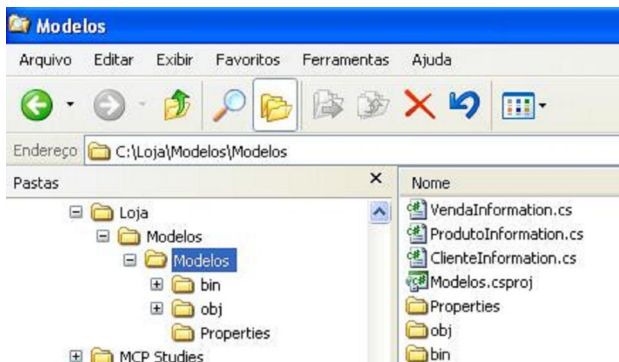
Para compilar o projeto vamos fazer o seguinte. No Solution Explorer, clique com o botão direito sobre o projeto **Modelos** e escolha a opção **Build**:



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

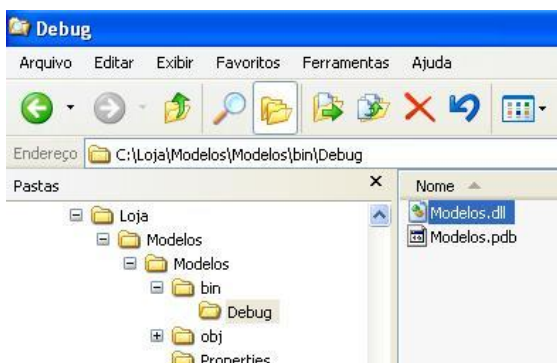


No Windows Explorer, observe que na pasta do projeto Modelos foi criado um arquivo para cada classe que implementamos.



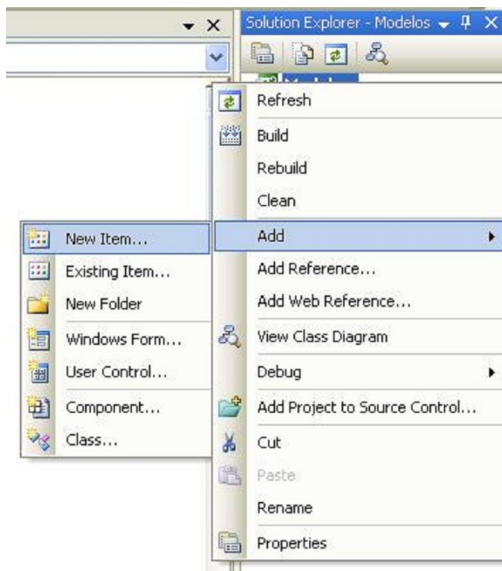
Quando compilamos o projeto com a opção Build, o MS Visual Studio criou o arquivo Modelos.dll.

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).

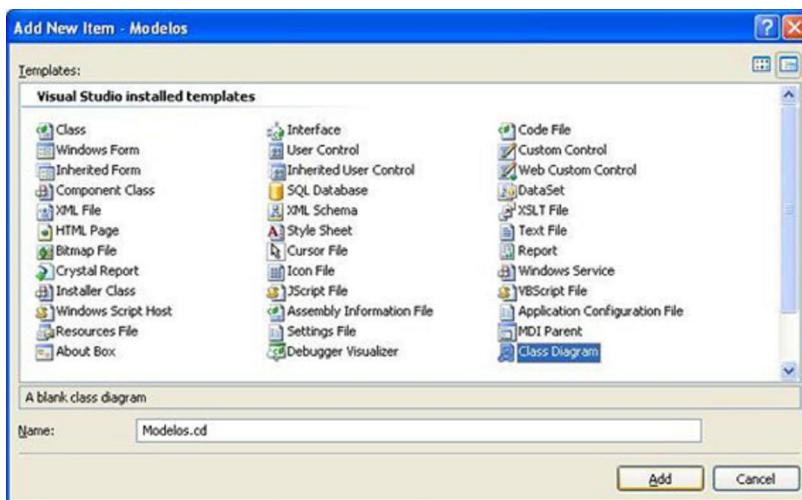


Após terminar o projeto Modelos, muitos programadores costumam criar um *Diagrama de Classes* para ter uma visão melhor da sua arte. Vamos criar o Diagrama de Classes?

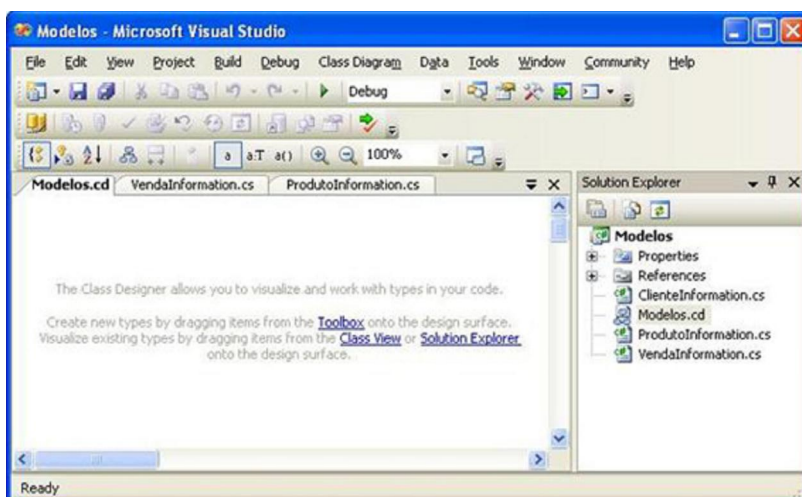
De volta ao Visual Studio, clique com o botão direito sobre o projeto **Modelos** e escolha **Add > New Item...**



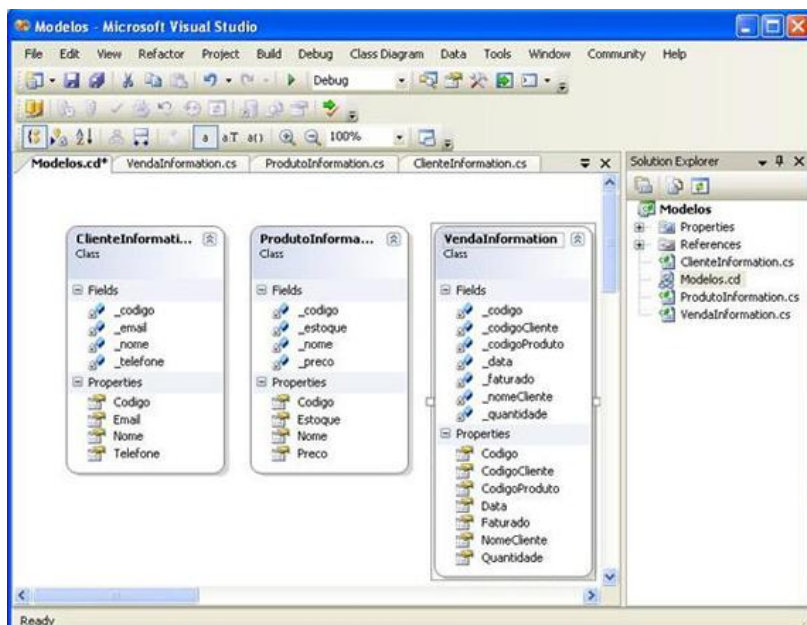
Na janela Add New Item, escolha o template **Class Diagram**. Digite **Modelos.cd** no nome e clique em **Add** para criar o nosso diagrama de classes como a seguir.



O ambiente chamado *Class Designer* é exibido.



Para a criação do diagrama vamos arrastar as classes *ClienteInformation*, *ProdutoInformation* e *VendaInformation* 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.

Se olharmos para o nosso desenho das camadas, vamos verificar que o próximo passo é implementar as classes da camada DAL. Faremos isso no próximo artigo.

Até breve!



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.



## 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#

**Linha de Código**

Like

9,981 people like [Linha de Código](#).



Facebook social plugin