



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 5

Este é o quinto 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



+1

0

5 -BLL – Camada de Regras de Negócio.

Neste artigo 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:



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

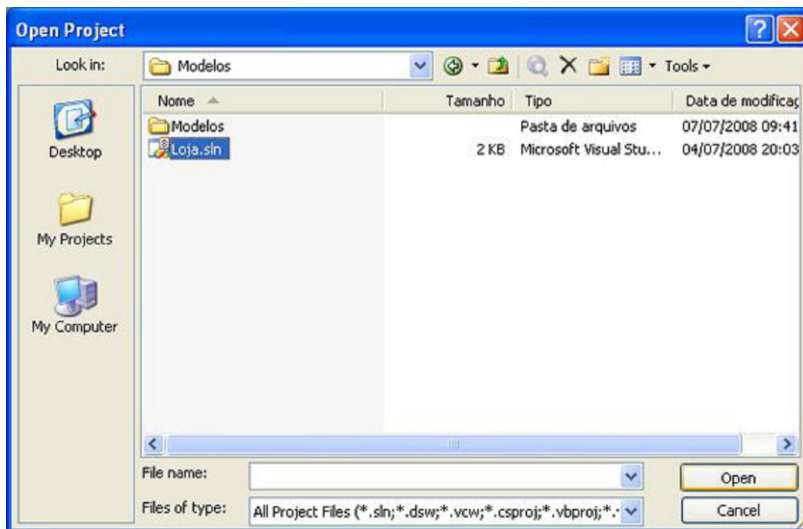
ASSINE

TOP 10 - ARTIGOS

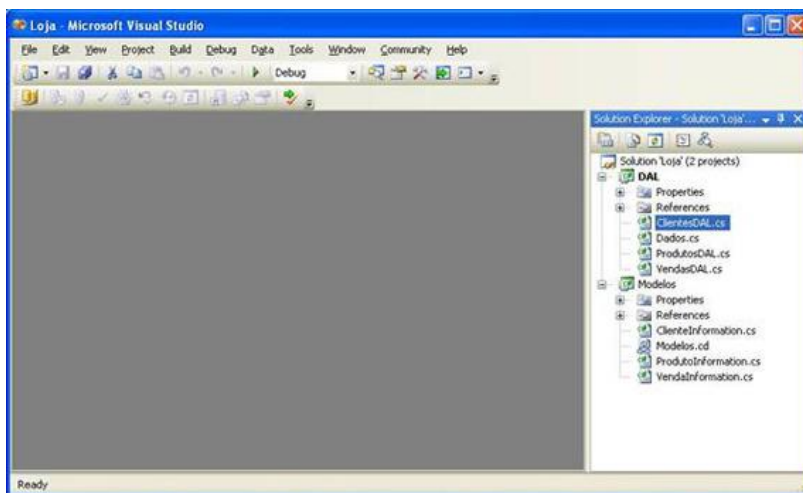
TOP 10 - AUTORES

- 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



Neste ponto do projeto temos implementados os projetos Modelos e DAL:



Vamos adicionar o projeto BLL à solução.

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

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

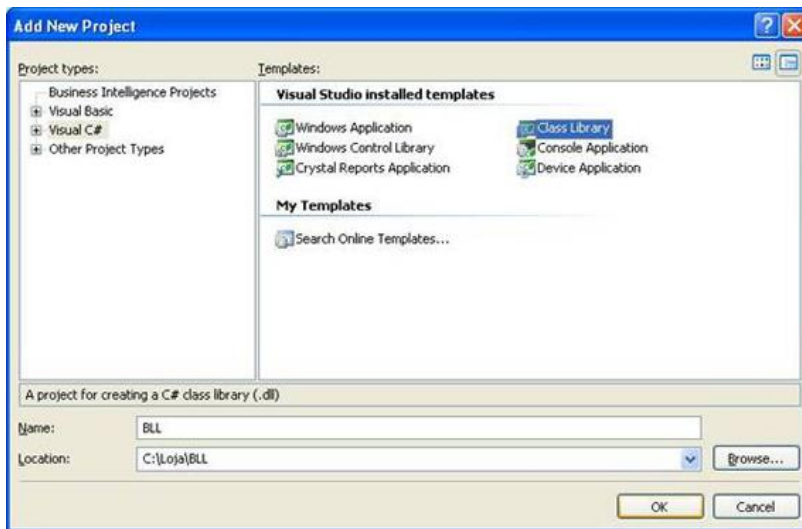
Project type: **Visual C#**

Template: **Class Library**

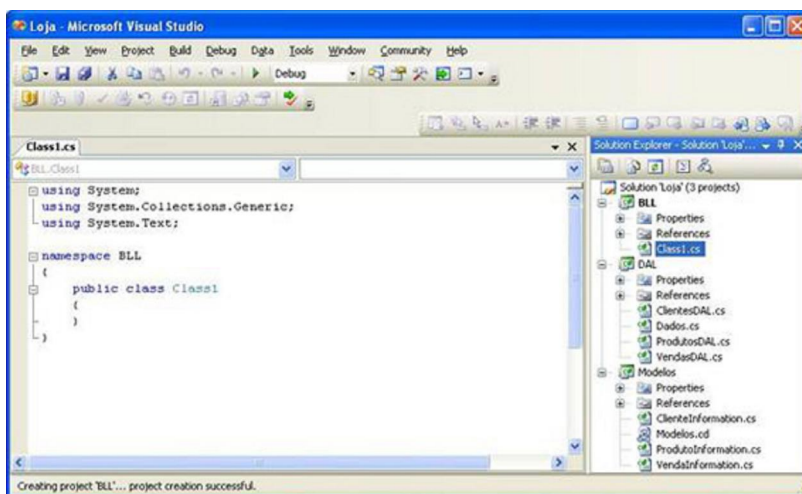
Name: **BLL**

Location: **C:\Loja\BLL**

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



O projeto BLL será criado:

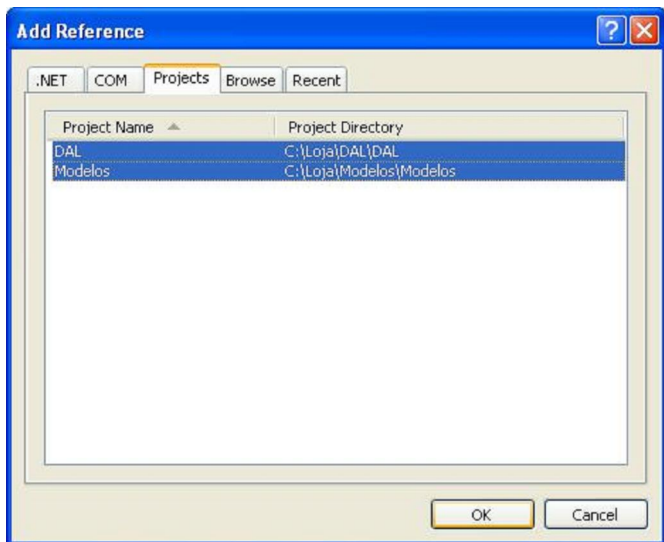


Analisando o nosso modelo em camadas abaixo:

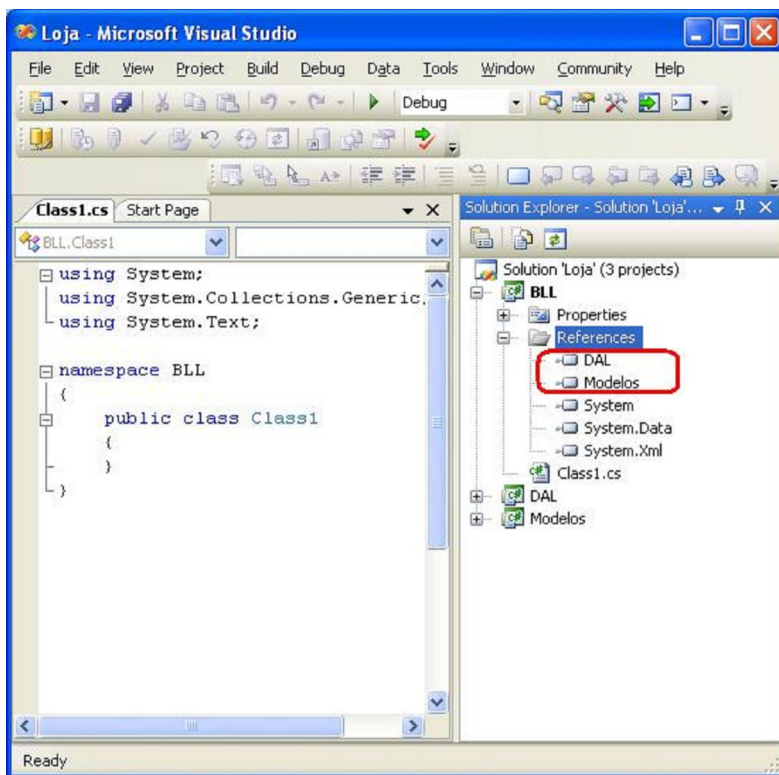
[Modelos] DAL ð BLL ð User Interface

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 seleccionálos;
- Clique em **Ok** para adicionar as referências.

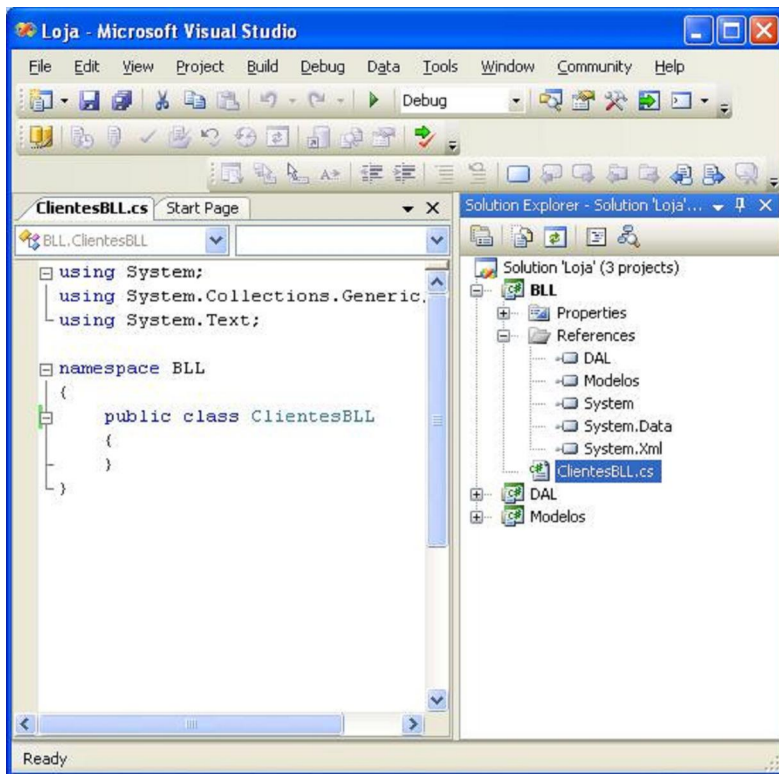


Podemos verificar as referências que acabamos de adicionar ao projeto BLL:



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**:



Copie e cole o seguinte código para a classe ClientesBLL.cs:

```
using System;
using System.Data;
using Loja.Modelos;
using Loja.DAL;
namespace Loja.BLL
{
    public class ClientesBLL
    {
        public void Incluir(ClienteInformation cliente)
        {
            //O nome do cliente é obrigatório
            if (cliente.Nome.Trim().Length == 0)
            {
                throw new Exception("O nome do cliente é obrigatório");
            }

            //E-mail é sempre em letras minúsculas
            cliente.Email = cliente.Email.ToLower();

            //Se tudo está Ok, chama a rotina de inserção.
            ClientesDAL obj = new ClientesDAL();
            obj.Incluir(cliente);
        }

        public void Alterar(ClienteInformation cliente)
```

```

{
    ClientesDAL obj = new ClientesDAL();

    obj.Alterar(cliente);
}

public void Excluir(int codigo)
{
    if (codigo < 1)
    {
        throw new Exception("Selecione um cliente antes de excluí-lo.");
    }

    ClientesDAL obj = new ClientesDAL();

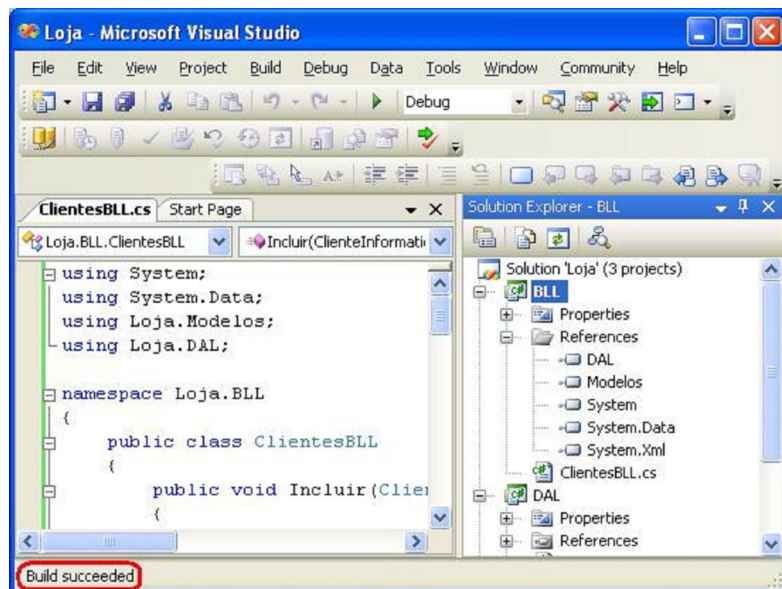
    obj.Excluir(codigo);
}

public DataTable Listagem()
{
    ClientesDAL obj = new ClientesDAL();

    return obj.Listagem();
}
}
}

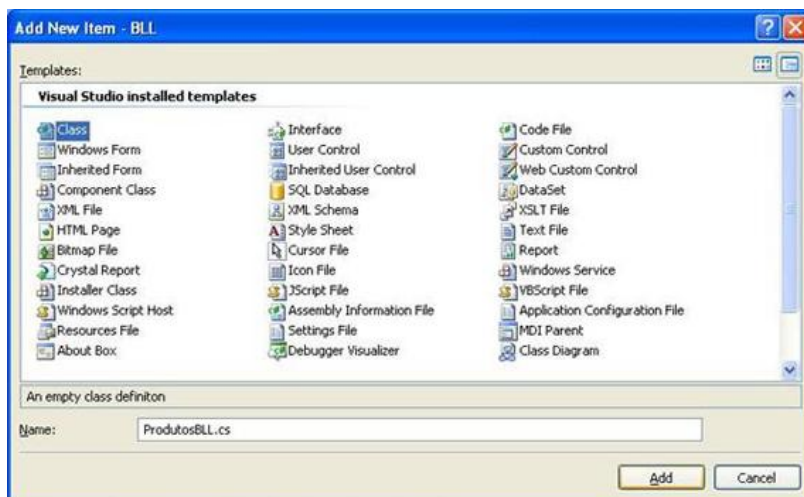
```

- Dê um **Build** no projeto BLL para confirmar a inexistência de erros na codificação:



Agora vamos criar a classe ProdutosBLL.cs.

- Adicione uma nova classe ao projeto BLL nomeando-a como **ProdutosBLL.cs**:



Copie e cole o seguinte código para a classe ProdutosBLL.cs:

```
using System;

using System.Data;

using System.Collections.Generic;

using System.Text;

using System.Collections;

using Loja.Modelos;

using Loja.DAL;

namespace Loja.BLL
{
    public class ProdutosBLL
    {
        public ArrayList ProdutosEmFalta()
        {
            ProdutosDAL obj = new ProdutosDAL();

            return obj.ProdutosEmFalta();
        }

        public void Incluir(ProdutoInformation produto)
        {
            // Nome do produto é obrigatório

            if (produto.Nome.Trim().Length == 0)
            {
                throw new Exception("O nome do produto é obrigatório.");
            }

            // O preço do produto não pode ser negativo

            if (produto.Preco < 0)
            {
                throw new Exception("Preço do produto não pode ser negativo.");
            }
        }
    }
}
```

```
    }

    // O estoque do produto não pode ser negativo

    if (produto.Estoque < 0)

    {

        throw new Exception("Estoque do produto não pode ser negativo.");

    }

    //Se tudo estiver ok, chama a rotina de gravação

    ProdutosDAL obj = new ProdutosDAL();

    obj.Incluir(produto);

}

public void Alterar(ProdutoInformation produto)

{

    ProdutosDAL obj = new ProdutosDAL();

    obj.Alterar(produto);

}

public void Excluir(int codigo)

{

    ProdutosDAL obj = new ProdutosDAL();

    obj.Excluir(codigo);

}

public DataTable Listagem()

{

    ProdutosDAL obj = new ProdutosDAL();

    return obj.Listagem();

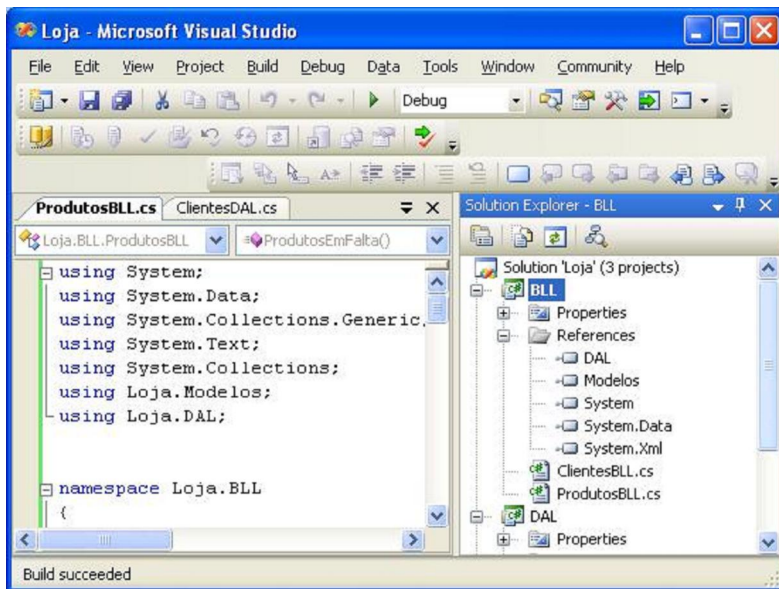
}

}

}
```

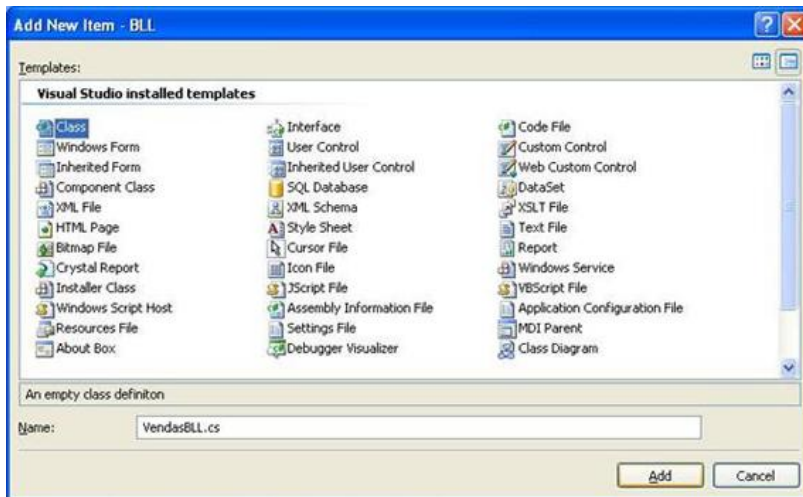
- Dê um **Build** no projeto BLL para confirmar a inexistência de erros na codificação:





Agora vamos implementar a classe VendasBLL.cs.

- Adicione uma nova classe ao projeto BLL nomeando-a como **VendasBLL.cs**:



Copie e cole o seguinte código para a classe VendasBLL.cs:

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Data;

using Loja.DAL;

using Loja.Modelos;

namespace Loja.BLL
{
    public class VendasBLL
    {
        //Este é um campo privado para armazenar uma instância da classe DAL.

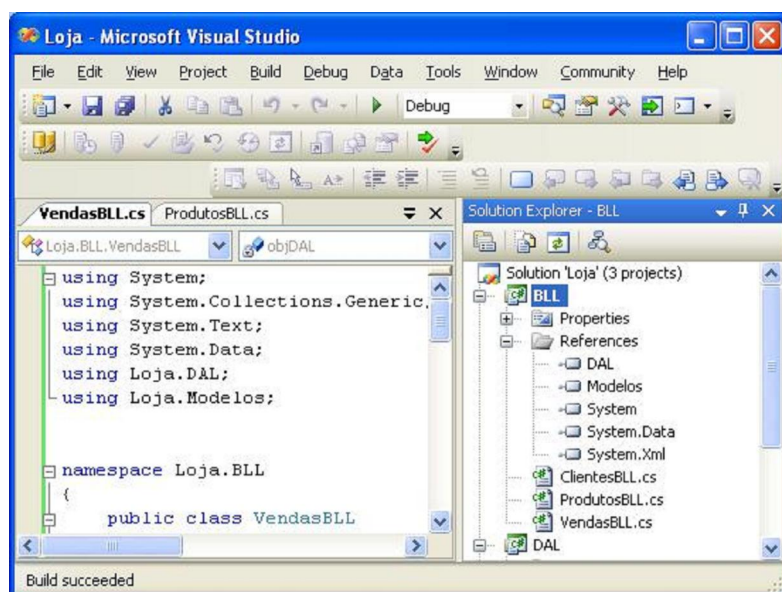
        private VendasDAL objDAL;

        //Esse é o construtor da classe VendasBLL

        public VendasBLL()
```

```
{  
  
    objDAL = new VendasDAL();  
  
}  
  
public DataTable ListaDeProdutos  
{  
  
    get  
  
    {  
  
        return objDAL.ListaDeProdutos;  
  
    }  
  
}  
  
public DataTable ListaDeClientes  
{  
  
    get  
  
    {  
  
        return objDAL.ListaDeClientes;  
  
    }  
  
}  
  
public void Incluir(VendaInformation venda)  
{  
  
    objDAL.Incluir(venda);  
  
}  
  
}  
  
}
```

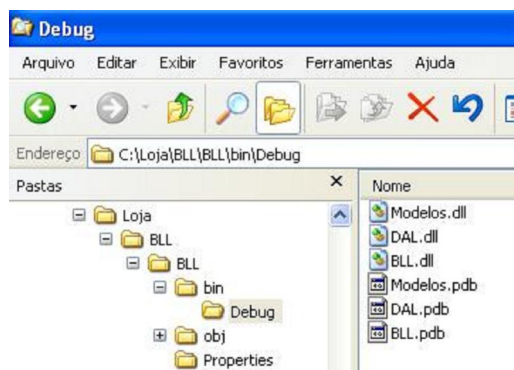
- Dê um **Build** no projeto BLL para confirmar a inexistência de erros na codificação:



Parabéns! Finalizamos a implementação da camada BLL – Regras de negócio.

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.



Compare isso com o nosso modelo em camadas a seguir:

[Modelos] DAL ð BLL ð User Interface

Vemos que 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.

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.



## 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: [f](#) [s](#) [t](#) [g+](#)


Linha de Código faz parte do grupo Web-03

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

**Linha de Código**

Like

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



Facebook social plugin

© 2014 Linha de Código. Todos os direitos reservados