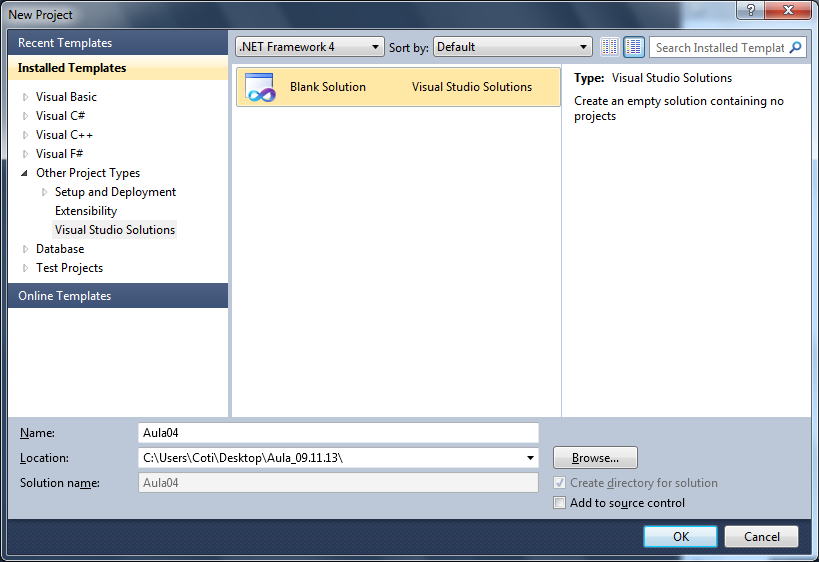
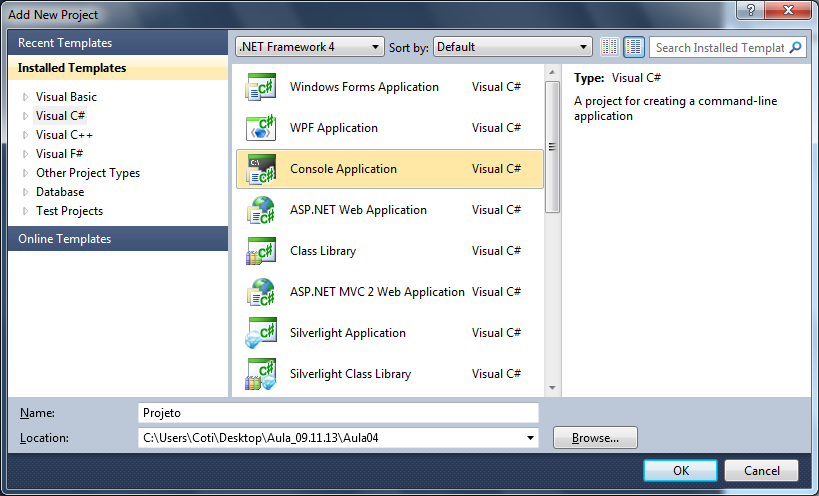
Passo 1: Criação da Solution...

(Ambiente de trabalho)

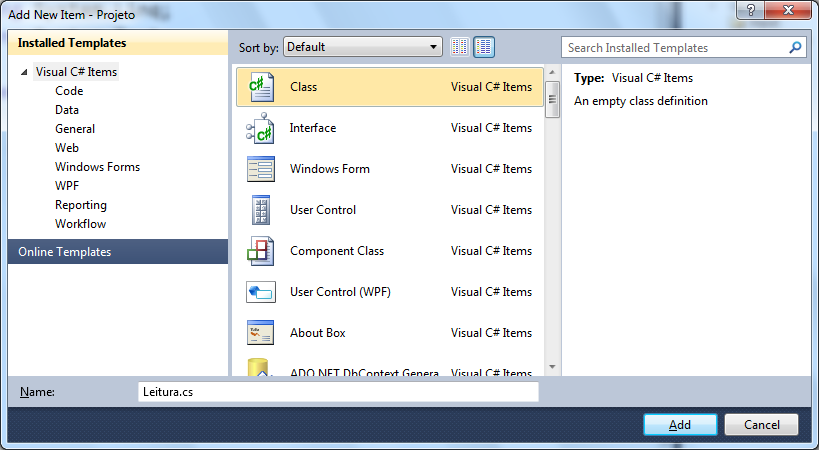


Criação do Projeto

Console Application (Ambiente DOS 🡪 Prompt de Comando)



Classe para leitura de dados   
relacionado a Cliente...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Projeto.Input

{

public class Leitura

{

//Método para ler e retornar um Id informado pelo prompt

public int LerIdCliente()

{

Console.Write("Informe o Id do Cliente...: ");

return Convert.ToInt32(Console.ReadLine());

}

//Método para ler e retornar o Nome do Cliente informado

public string LerNome()

{

Console.Write("Informe o Nome do Cliente.: ");

return Console.ReadLine();

}

//Método para ler e retornar o Email do Cliente informado

public string LerEmail()

{

Console.Write("Informe o Email do Cliente: ");

return Console.ReadLine();

}

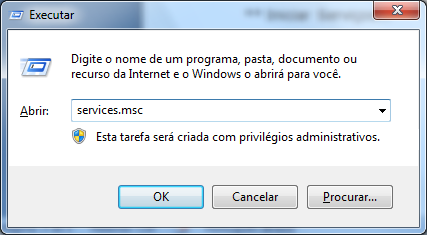
}

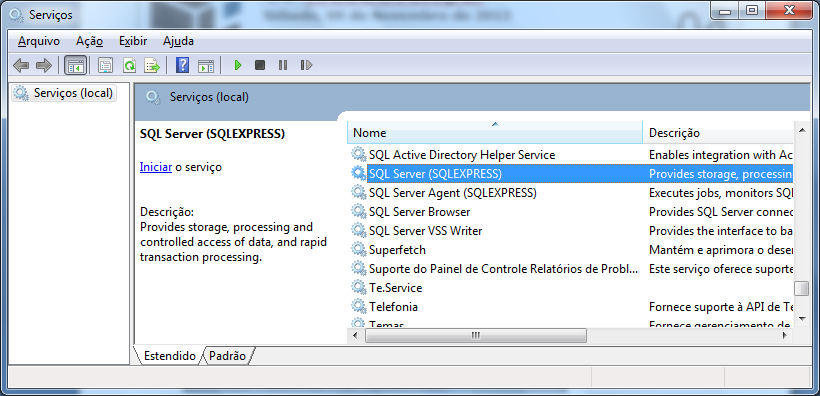
}

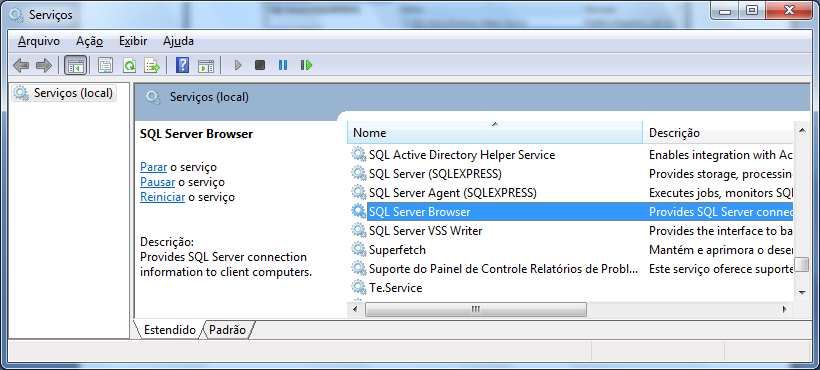
SqlServer

Banco de dados

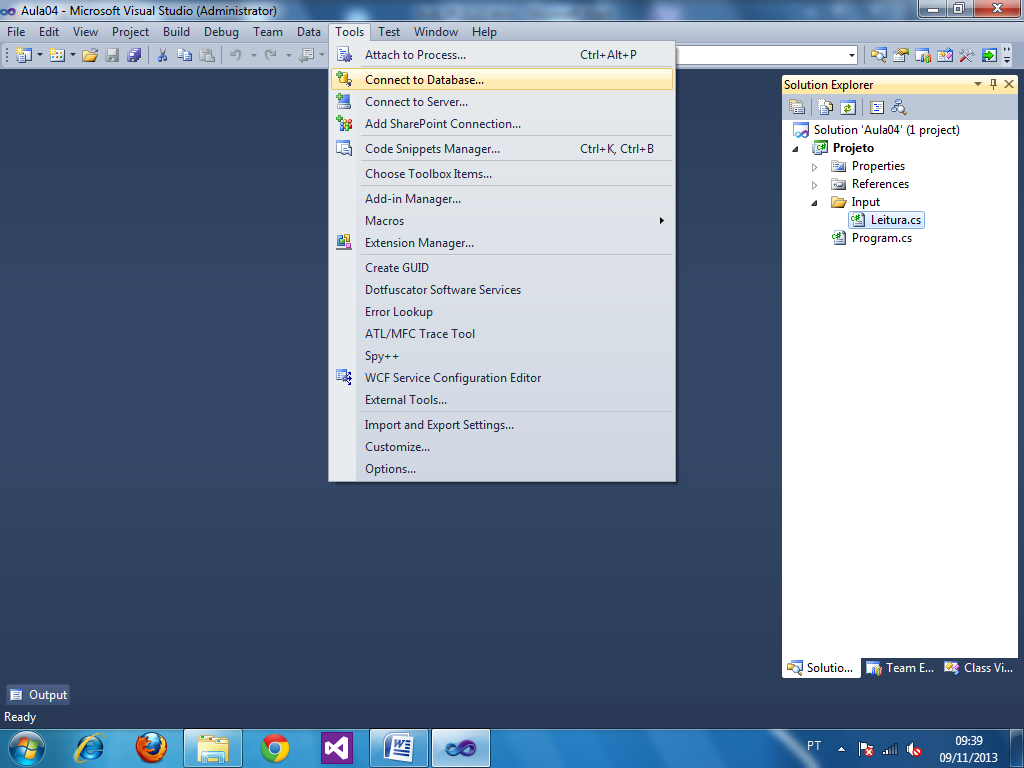
\*\* Iniciar Serviços...

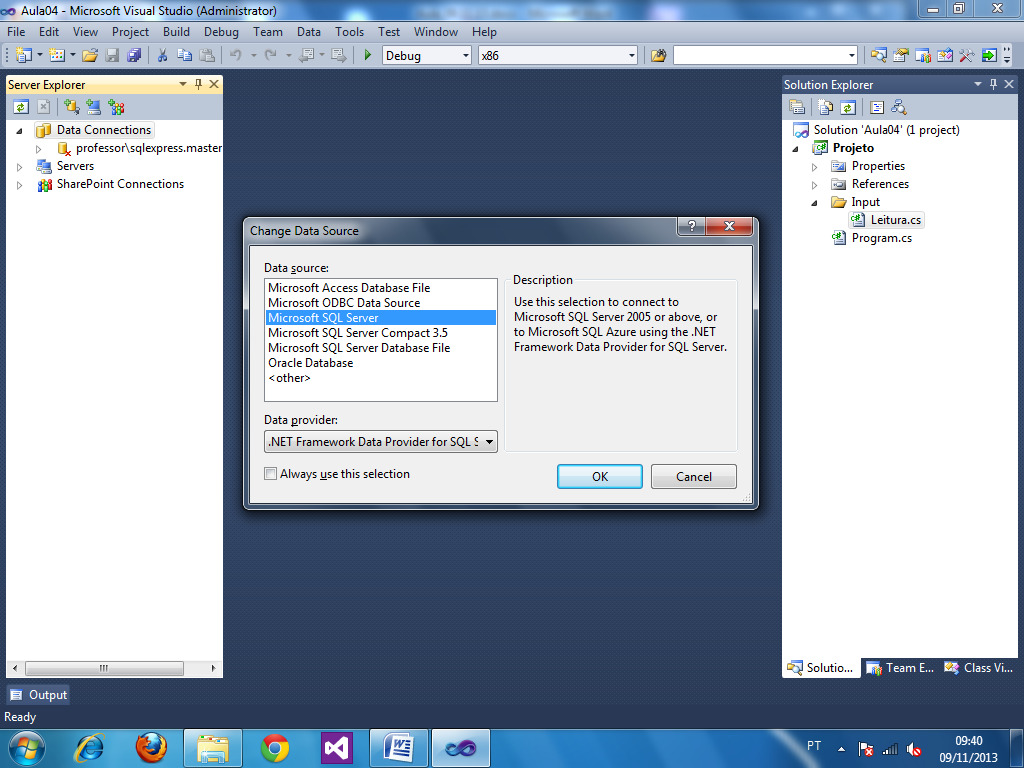


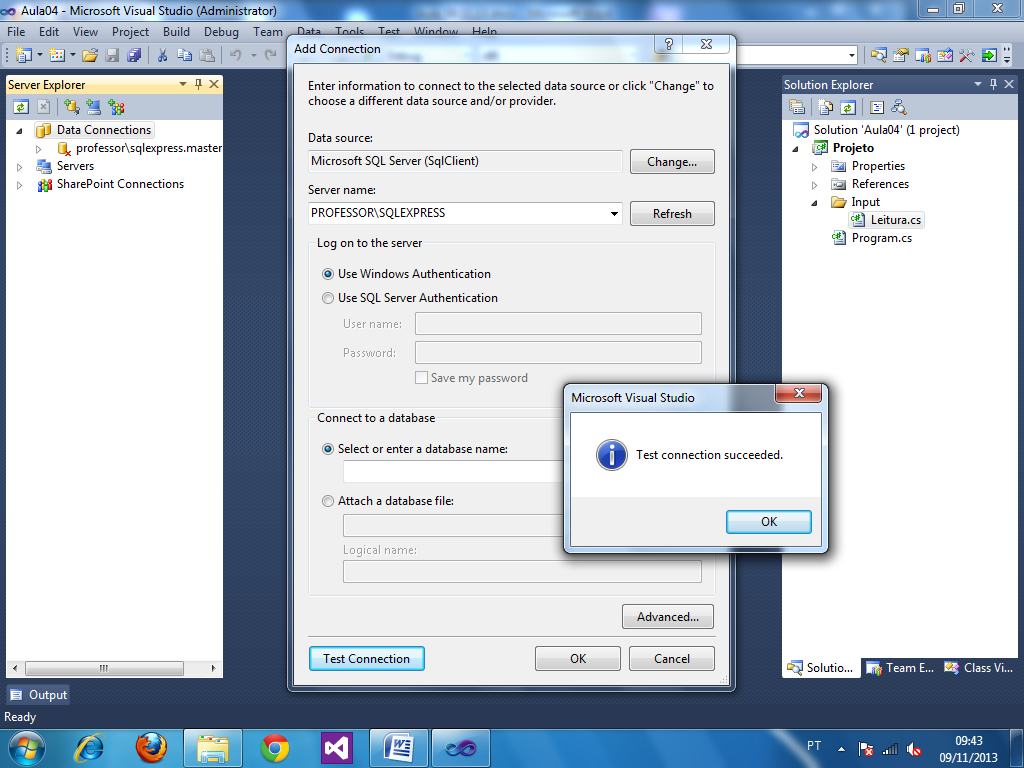




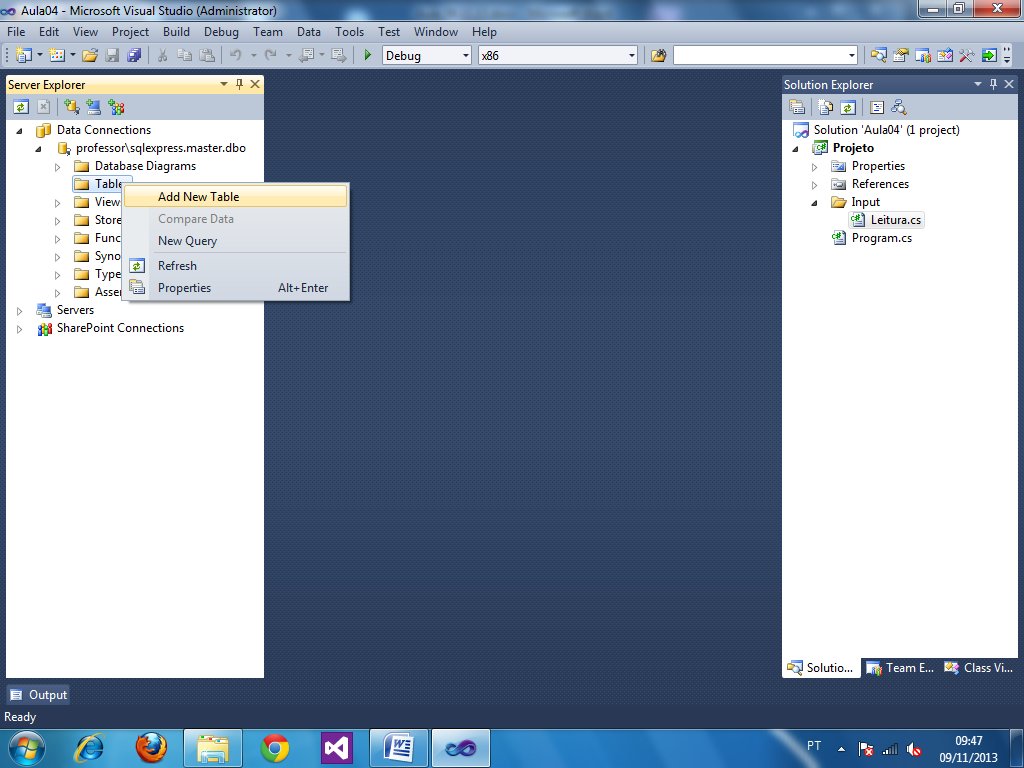
Acessar o SqlServer através do VisualStudio...

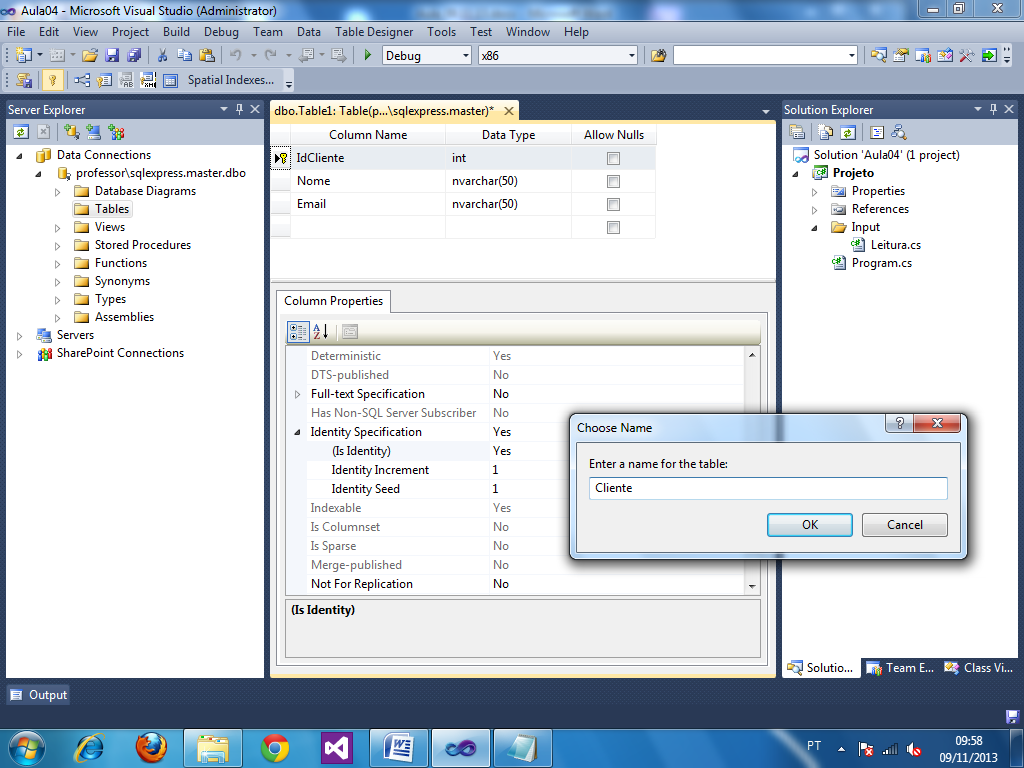






Criando a tabela de Clientes...





Data Access Layer (DAL)

Camada de acesso a dados

Em aplicações .NET o recomendado é que certos aspectos do projeto como acesso a banco de dados, regras de negócio, etc, sejam desenvolvidos em camadas separadas do projeto principal.

* **Desenvolvimento baseado em camadas**

Segundo a Microsoft, todo projeto criado em .NET poderia ser dividido em 3 camadas:

**Data Access Layer (DAL)**

Camada de acesso, conexão, transações e consultas à base de dados.

**Business Logic Layer (BLL)**

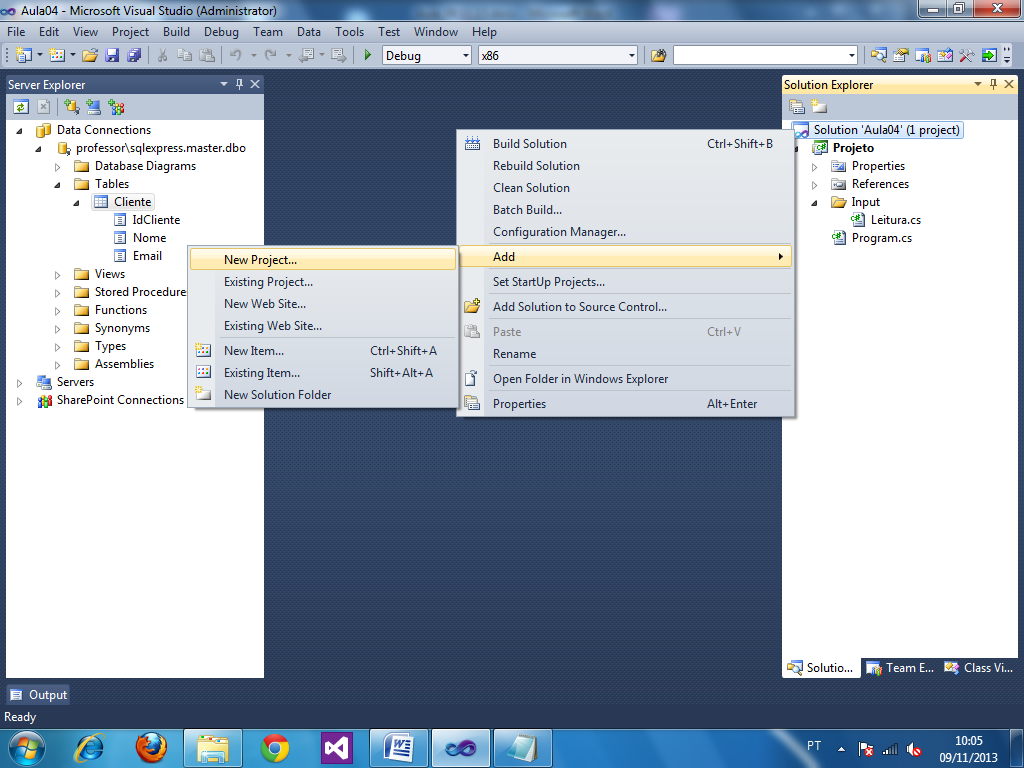
Camada para tratamento de regras de negócio

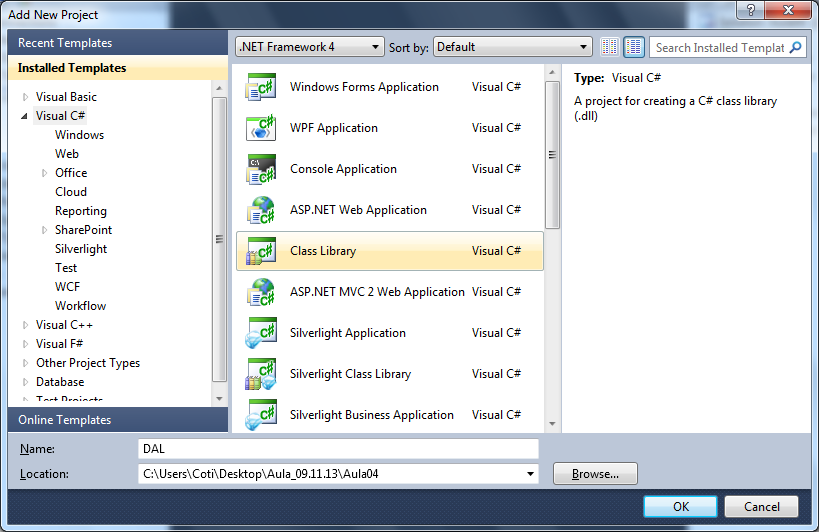
**Presentation Layer (PL)**

Camada de apresentação do sistema, por exemplo: Asp.NET

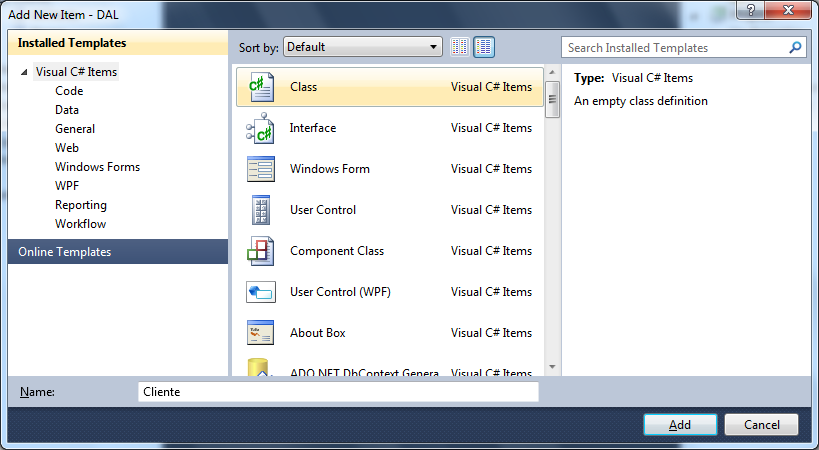
Projeto: Class Library

Biblioteca de código





Criando a Classe de Modelo   
para a entidade Cliente...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace DAL.Model

{

public class Cliente

{

#region Atributos

private int idCliente;

private string nome;

private string email;

#endregion

#region Construtores

//[ctor] + 2x[tab]

public Cliente() //Construtor default

{

//Vazio

}

//Sobrecarga de Construtores

public Cliente(int idCliente, string nome, string email)

{

this.idCliente = idCliente;

this.nome = nome;

this.email = email;

}

#endregion

#region Propriedades

public int IdCliente

{

set { idCliente = value; } //entrada

get { return idCliente; } //saída

}

public string Nome

{

set { nome = value; }

get { return nome; }

}

public string Email

{

set { email = value; }

get { return email; }

}

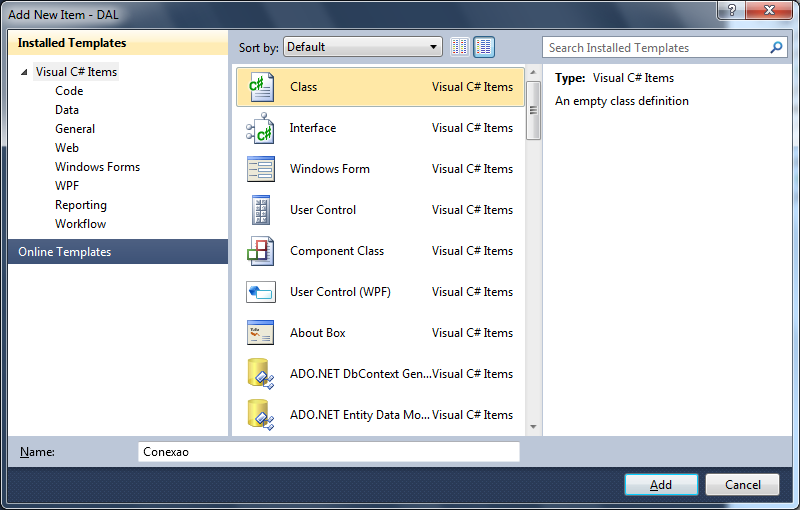
#endregion

}

}

Próximo Passo: Conexão com o SqlServer...

\*\* Persistência de dados (**Armazenamento**)



API: System.Data.SqlClient

Biblioteca padrão do C# para acesso a bases   
de dados do SqlServer

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient; //API para o SqlServer

namespace DAL.Persistence

{

public class Conexao

{

//Atributos | Visibilidade -> [protected]

//Permite o acesso ao elemento por meio de herança

//Utilizado para estabeler a conexão com a base de dados

protected SqlConnection Con;

//Utilizado para escrever rotinas SQL e executá-las no banco

protected SqlCommand Cmd;

//Utilizado para capturar e varrer resultados de consultas ao BD

protected SqlDataReader Dr;

//Métodos

//Abrir conexão com o banco de dados

protected void AbrirConexao()

{

//inicializar o atributo SqlConnection

Con = new SqlConnection();

Con.Open(); //conexão aberta!

}

//Fechar conexão com o banco de dados

protected void FecharConexao()

{

Con.Close(); //conexão fechada!

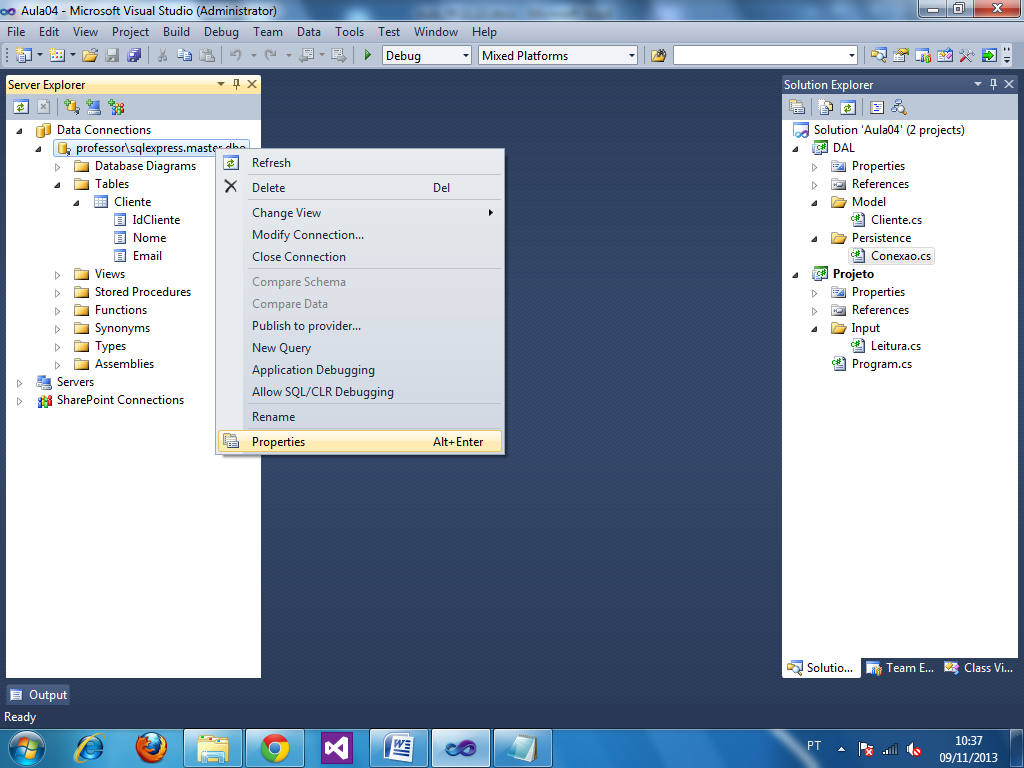
}

}

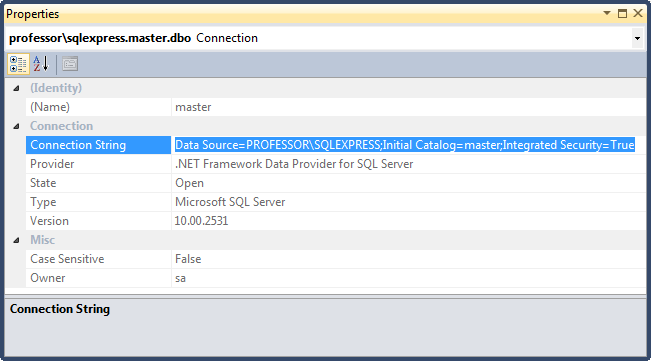
}

ConnectionString

Linha de texto que define o caminho e informações básicas para que uma API possa estabelecer conexão com uma base de dados.



Obtendo a String de Conexão...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient; //API para o SqlServer

namespace DAL.Persistence

{

public class Conexao

{

//Atributos | Visibilidade -> [protected]

//Permite o acesso ao elemento por meio de herança

//Utilizado para estabeler a conexão com a base de dados

protected SqlConnection Con;

//Utilizado para escrever rotinas SQL e executá-las no banco

protected SqlCommand Cmd;

//Utilizado para capturar e varrer resultados de consultas ao BD

protected SqlDataReader Dr;

//Métodos

//Abrir conexão com o banco de dados

protected void AbrirConexao()

{

//inicializar o atributo SqlConnection

Con = new SqlConnection(**@"Data Source=PROFESSOR\**

**SQLEXPRESS;Initial Catalog=master;**

**Integrated Security=True"**);

Con.Open(); //conexão aberta!

}

//Fechar conexão com o banco de dados

protected void FecharConexao()

{

Con.Close(); //conexão fechada!

}

}

}

Classe de persistência para   
a entidade Cliente...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient; //API para o SqlServer

using DAL.Model; //namespace da Classe -> Cliente

namespace DAL.Persistence

{

public class ClienteDal : Conexao

{

//Método para gravar um registro na tabela de Cliente

public void Salvar(Cliente c)

{

try

{

AbrirConexao(); //executando o método da Classe PAI

Cmd = new SqlCommand("insert into Cliente(Nome, Email)

values(@Nome, @Email)", Con);

Cmd.Parameters.AddWithValue("@Nome", c.Nome);

Cmd.Parameters.AddWithValue("@Email", c.Email);

Cmd.ExecuteNonQuery(); //executar

}

catch (Exception e)

{

//Informar o erro à camada executora

throw new Exception("Erro ao cadastrar Cliente: " + e.Message);

}

finally //finalizador

{

//tudo que é colocado no finally sempre é executado

FecharConexao();

}

}

//Método para atualizar os dados do Cliente

public void Atualizar(Cliente c)

{

try

{

AbrirConexao();

Cmd = new SqlCommand("update Cliente set Nome=@Nome, Email=@Email

where IdCliente=@IdCliente", Con);

Cmd.Parameters.AddWithValue("@Nome", c.Nome);

Cmd.Parameters.AddWithValue("@Email", c.Email);

Cmd.Parameters.AddWithValue("@IdCliente", c.IdCliente);

Cmd.ExecuteNonQuery(); //executar

}

catch(Exception e)

{

throw new Exception("Erro ao atualizar Cliente: " + e.Message);

}

finally

{

FecharConexao();

}

}

//Método para excluir o Cliente

public void Excluir(int IdCliente)

{

try

{

AbrirConexao();

Cmd = new SqlCommand("delete from Cliente

where IdCliente=@IdCliente", Con);

Cmd.Parameters.AddWithValue("@IdCliente", IdCliente);

Cmd.ExecuteNonQuery(); //executar

}

catch(Exception e)

{

throw new Exception("Erro ao excluir Cliente: " + e.Message);

}

finally

{

FecharConexao();

}

}

//Rotina para Obter 1 Cliente pelo Id

public Cliente ObterPorId(int IdCliente)

{

try

{

AbrirConexao();

Cmd = new SqlCommand("select \* from Cliente

where IdCliente=@IdCliente", Con);

Cmd.Parameters.AddWithValue("@IdCliente", IdCliente);

Dr = Cmd.ExecuteReader();

//executa a consulta e lê os registros obtidos

if(Dr.Read()) //se algum registro foi encontrado

{

Cliente c = new Cliente();

//instanciando um objeto da Classe Cliente

c.IdCliente = Convert.ToInt32( Dr["IdCliente"] );

c.Nome = Convert.ToString( Dr["Nome"] );

c.Email = Convert.ToString( Dr["Email"] );

return c; //retornar o Cliente

}

else

{

return null; //vazio (sem espaço de memória)

}

}

catch(Exception e)

{

throw new Exception("Erro ao obter Cliente: " + e.Message);

}

finally

{

FecharConexao();

}

}

//Método para listar todos os Clientes cadastrados

public List<Cliente> ListarTodos()

{

try

{

AbrirConexao();

Cmd = new SqlCommand("select \* from Cliente", Con);

Dr = Cmd.ExecuteReader(); //executa a consulta e lê os registros

List<Cliente> lista = new List<Cliente>(); //lista vazia

while(Dr.Read()) //enquanto houver registros

{

Cliente c = new Cliente();

//instanciando um objeto da Classe Cliente

c.IdCliente = Convert.ToInt32(Dr["IdCliente"]);

c.Nome = Convert.ToString(Dr["Nome"]);

c.Email = Convert.ToString(Dr["Email"]);

lista.Add(c); //adiciono o Cliente na lista

}

//retornar a lista

return lista;

}

catch (Exception e)

{

throw new Exception("Erro ao consultar cliente: " + e.Message);

}

finally

{

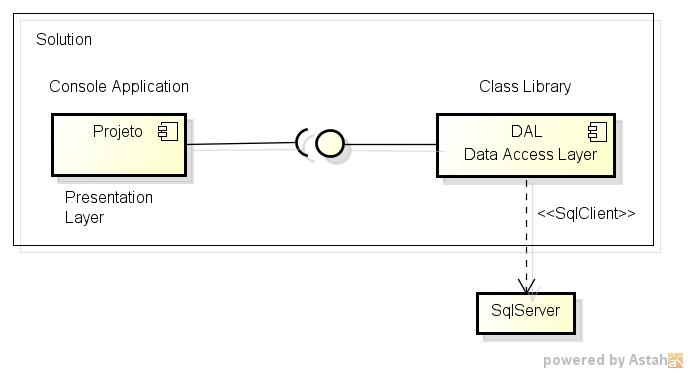
FecharConexao();

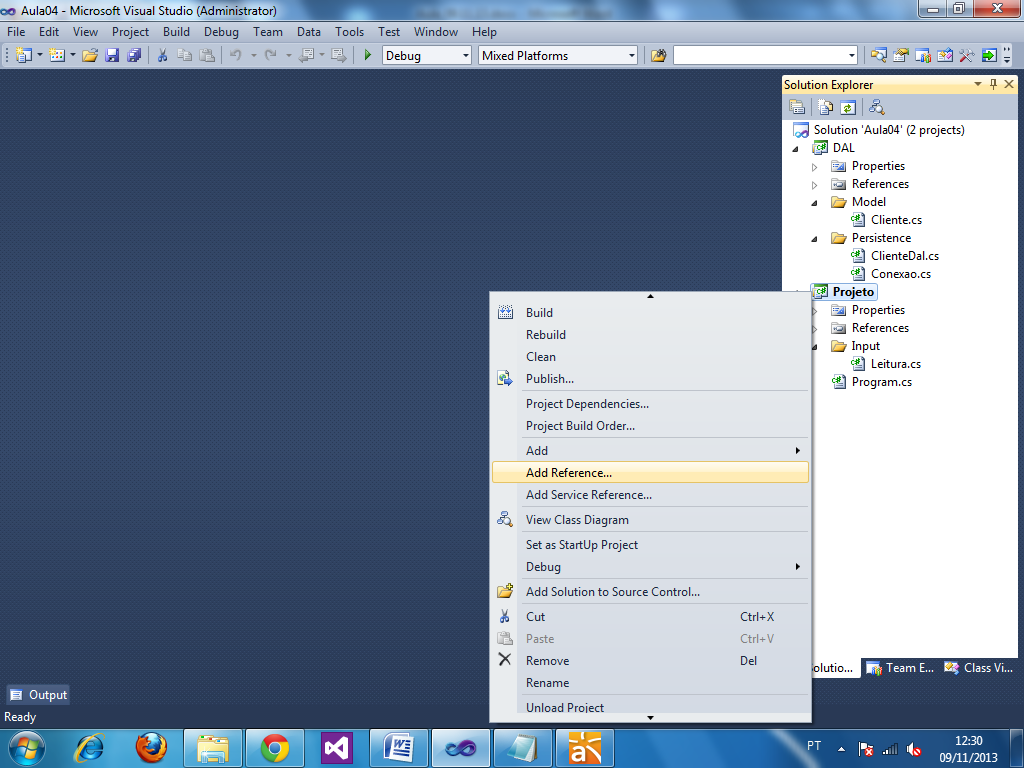
}

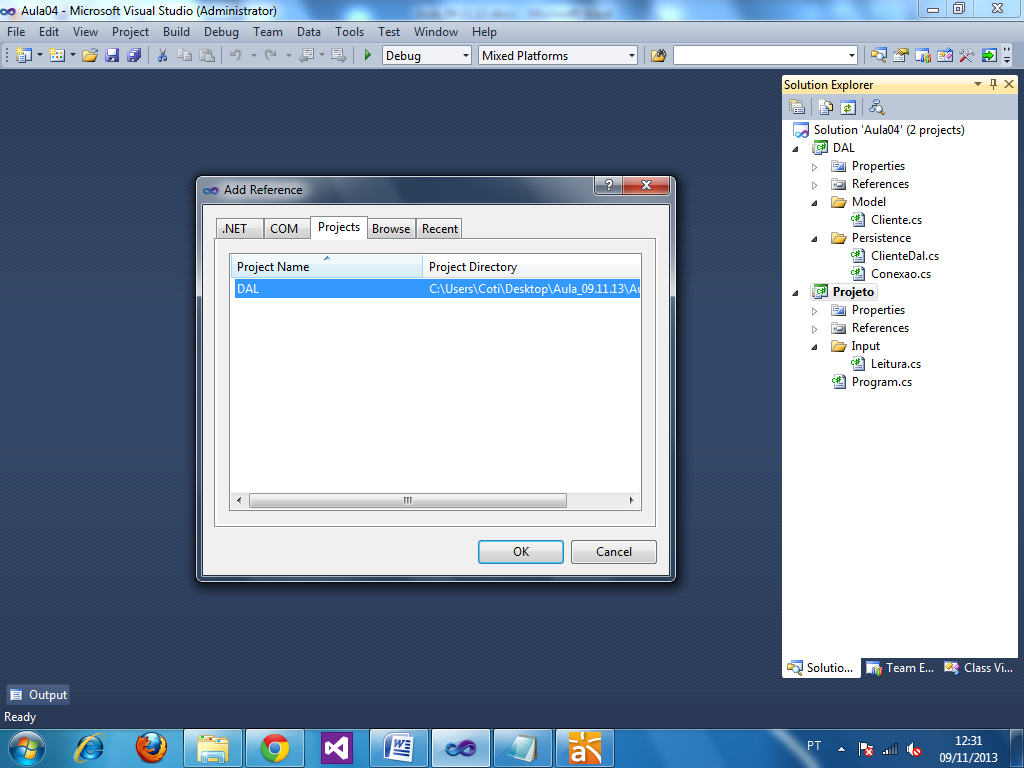
}

}

}







Testando as operações...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using DAL.Model;

using DAL.Persistence;

using Projeto.Input;

namespace Projeto

{

public class Program

{

static void Main(string[] args)

{

try

{

Leitura l = new Leitura();

Cliente c = new Cliente();

c.Nome = l.LerNome();

c.Email = l.LerEmail();

ClienteDal d = new ClienteDal();

d.Salvar(c); //gravação

Console.WriteLine("\nDados gravados com sucesso\n");

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

//Pausar

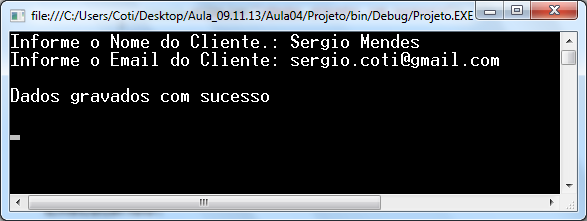
Console.ReadKey();

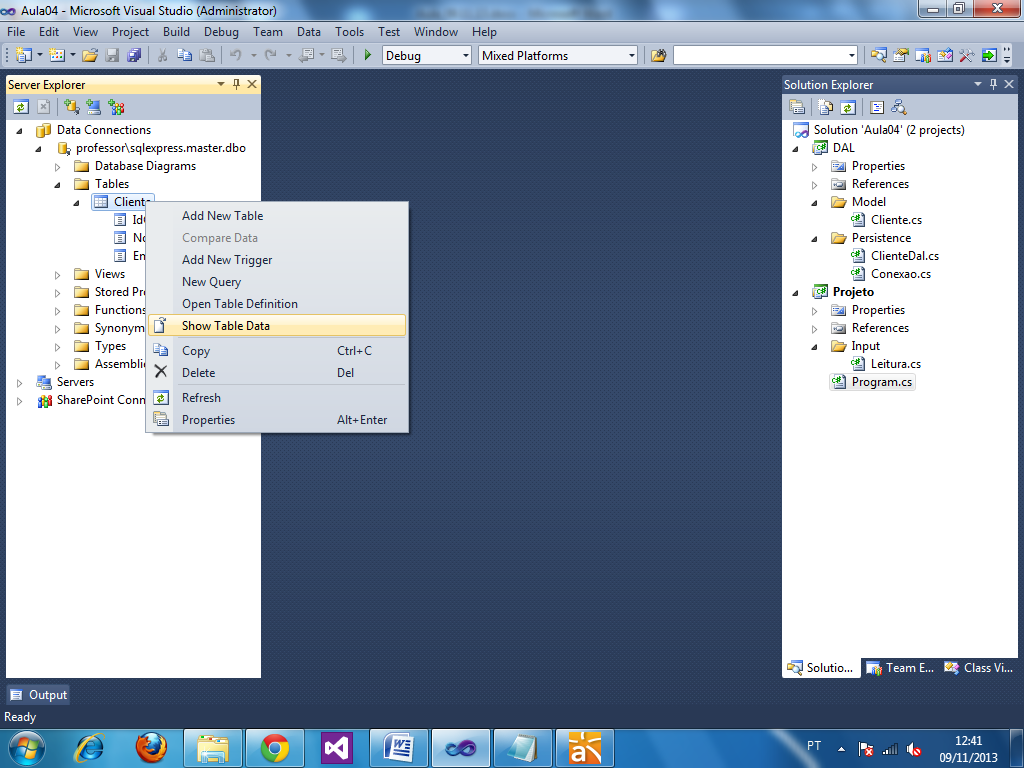
}

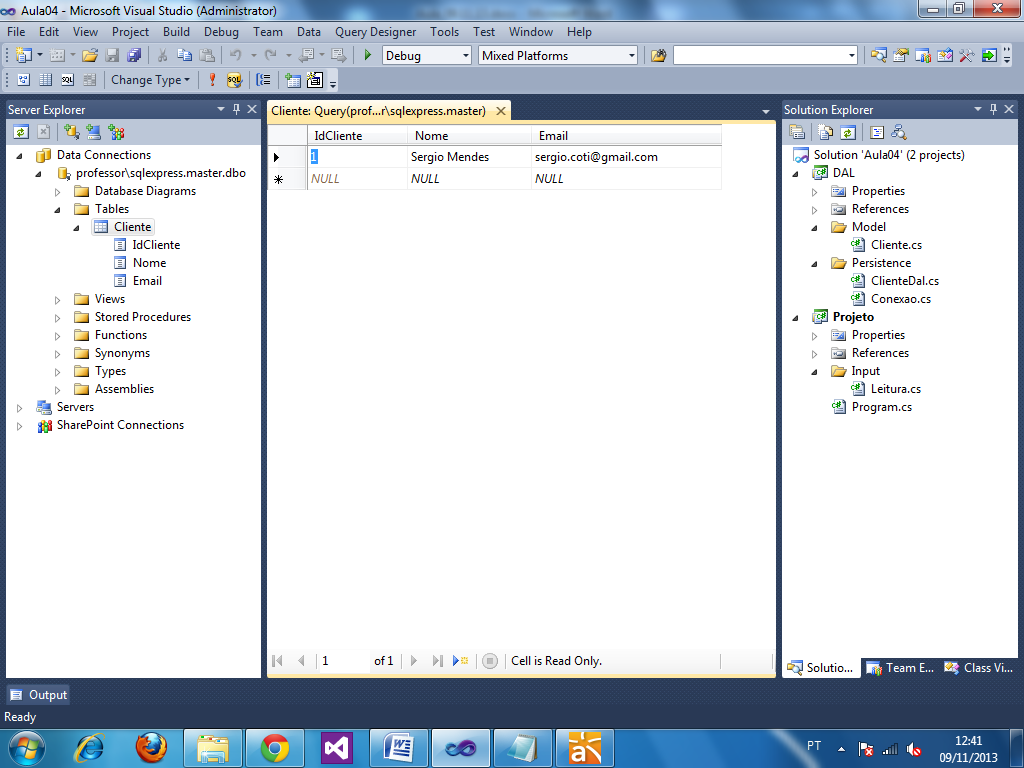
}

}

Executando:







Testando a Consulta...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using DAL.Model;

using DAL.Persistence;

using Projeto.Input;

namespace Projeto

{

public class Program

{

static void Main(string[] args)

{

try

{

Leitura l = new Leitura();

Cliente c = new Cliente();

c.Nome = l.LerNome();

c.Email = l.LerEmail();

ClienteDal d = new ClienteDal();

d.Salvar(c); //gravação

Console.WriteLine("\nDados gravados com sucesso\n");

//Varrer a listagem de Clientes obtido

**foreach(Cliente item in d.ListarTodos())**

{

Console.WriteLine("Id....: " + item.IdCliente);

Console.WriteLine("Nome..: " + item.Nome);

Console.WriteLine("Email.: " + item.Email);

Console.WriteLine("\n"); //quebra de linha

}

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

//Pausar

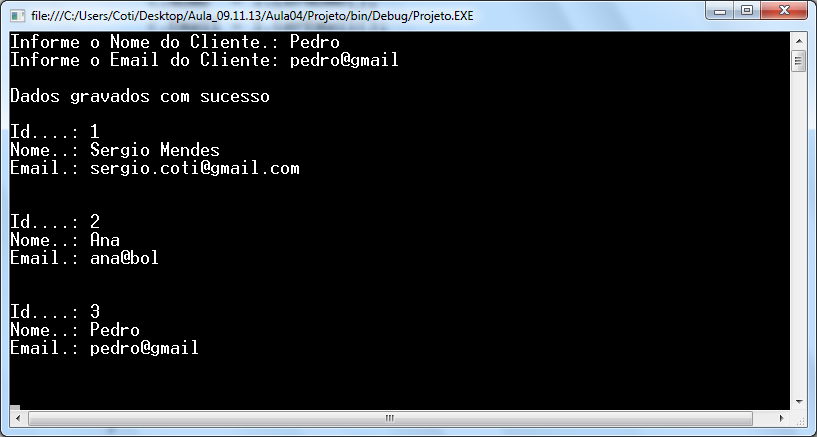
Console.ReadKey();

}

}

}

Resultado...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using DAL.Model;

using DAL.Persistence;

using Projeto.Input;

namespace Projeto

{

public class Program

{

static void Main(string[] args)

{

try

{

Leitura l = new Leitura();

Cliente c = new Cliente();

//c.Nome = l.LerNome();

//c.Email = l.LerEmail();

ClienteDal d = new ClienteDal();

//d.Salvar(c); //gravação

int IdCliente = l.LerIdCliente();

**if (d.ObterPorId(IdCliente) != null)**

{

**d.Excluir(IdCliente);**

Console.WriteLine("\nCliente excluido com sucesso.\n");

}

else

{

Console.WriteLine("\nCliente nao encontrado.\n");

}

//Console.WriteLine("\nDados gravados com sucesso\n");

//Varrer a listagem de Clientes obtido

foreach(Cliente item in d.ListarTodos())

{

Console.WriteLine("Id....: " + item.IdCliente);

Console.WriteLine("Nome..: " + item.Nome);

Console.WriteLine("Email.: " + item.Email);

Console.WriteLine("\n"); //quebra de linha

}

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

//Pausar

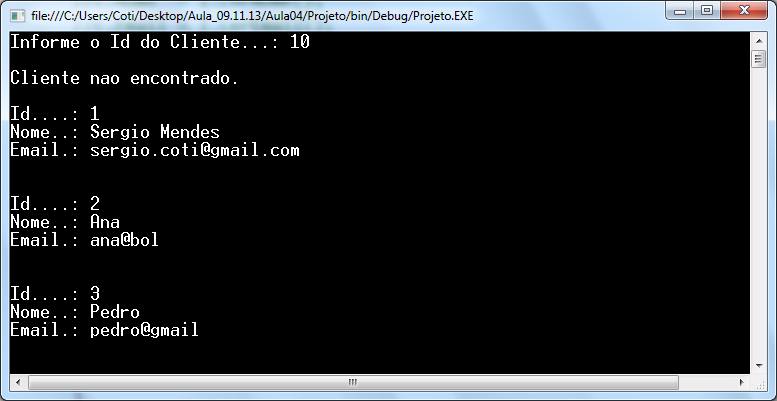
Console.ReadKey();

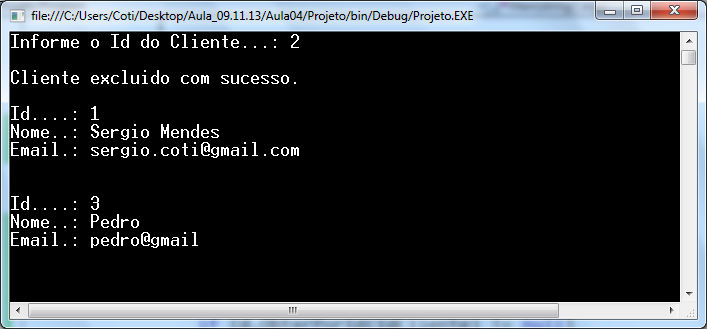
}

}

}

Executando a Exclusão...





Testando a atualização...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using DAL.Model;

using DAL.Persistence;

using Projeto.Input;

namespace Projeto

{

public class Program

{

static void Main(string[] args)

{

try

{

Leitura l = new Leitura();

Cliente c = new Cliente();

//c.Nome = l.LerNome();

//c.Email = l.LerEmail();

ClienteDal d = new ClienteDal();

//d.Salvar(c); //gravação

//int IdCliente = l.LerIdCliente();

//if (d.ObterPorId(IdCliente) != null)

//{

// d.Excluir(IdCliente);

// Console.WriteLine("\nCliente excluido com sucesso.\n");

//}

//else

//{

// Console.WriteLine("\nCliente nao encontrado.\n");

//}

**c.IdCliente = l.LerIdCliente();**

**c.Nome = l.LerNome();**

**c.Email = l.LerEmail();**

**d.Atualizar(c);**

**Console.WriteLine("\nCliente atualizado com sucesso\n");**

//Console.WriteLine("\nDados gravados com sucesso\n");

//Varrer a listagem de Clientes obtido

foreach(Cliente item in d.ListarTodos())

{

Console.WriteLine("Id....: " + item.IdCliente);

Console.WriteLine("Nome..: " + item.Nome);

Console.WriteLine("Email.: " + item.Email);

Console.WriteLine("\n"); //quebra de linha

}

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

//Pausar

Console.ReadKey();

}

}

}

Executando...

