Integração entre aplicações

Arquitetura baseada em WebServices

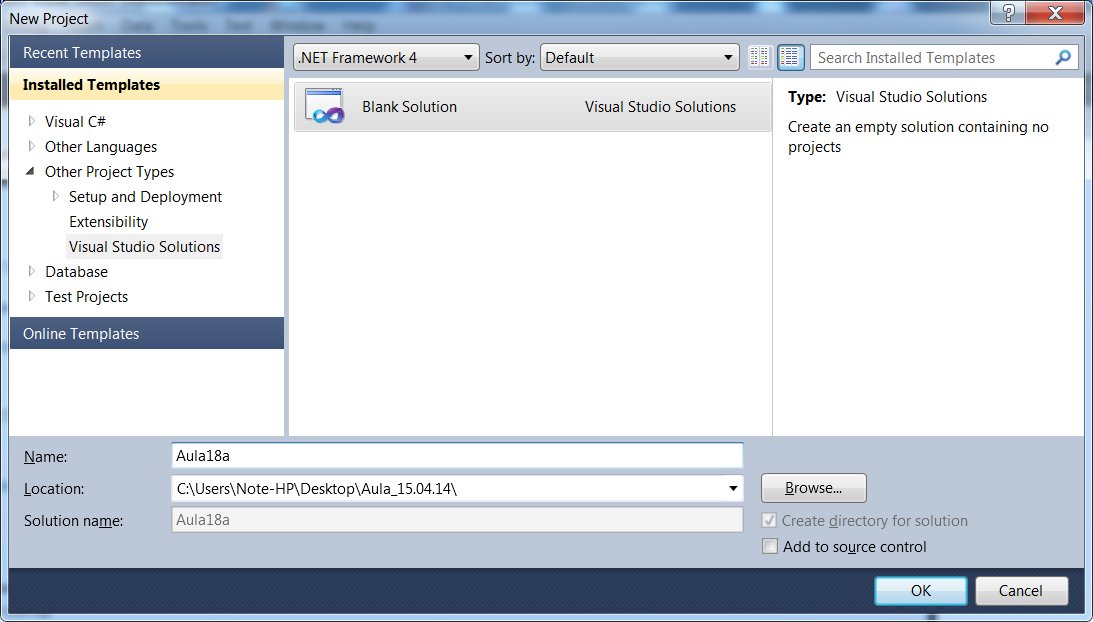
* WebServices

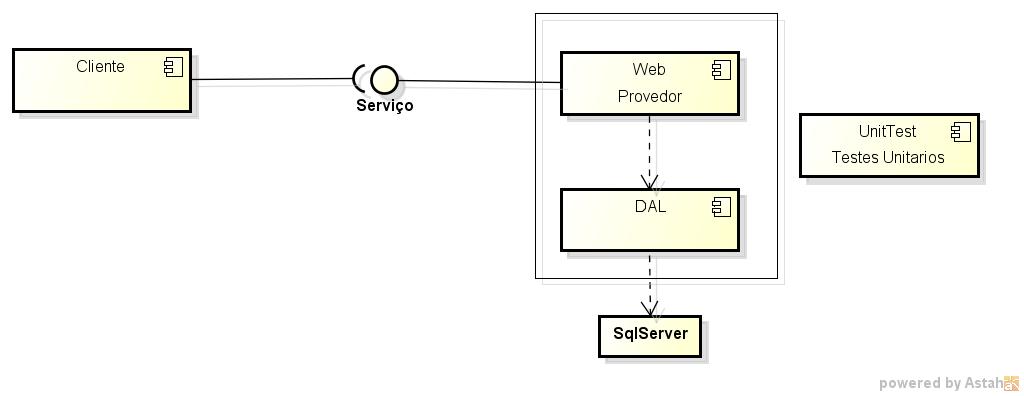
Tecnologia utilizada para integração de projetos web. Permite comunicar e compartilhar recursos e serviços entre aplicações de plataformas distintas (Java, NET, Php, etc...)

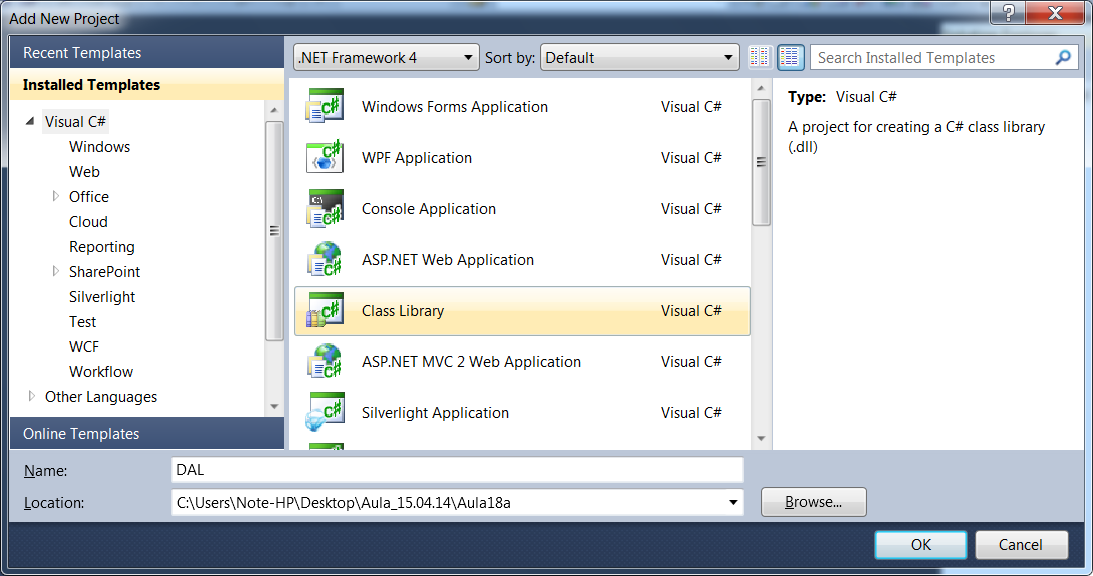
* + Provedor
    - Aplicação que disponibiliza o serviço, a funcionalidade que será compartilhada
  + Cliente
    - Aplicação que consome e acessa o serviço do provedor
  + Serviço
    - Funcionalidade compartilhada entre as aplicações

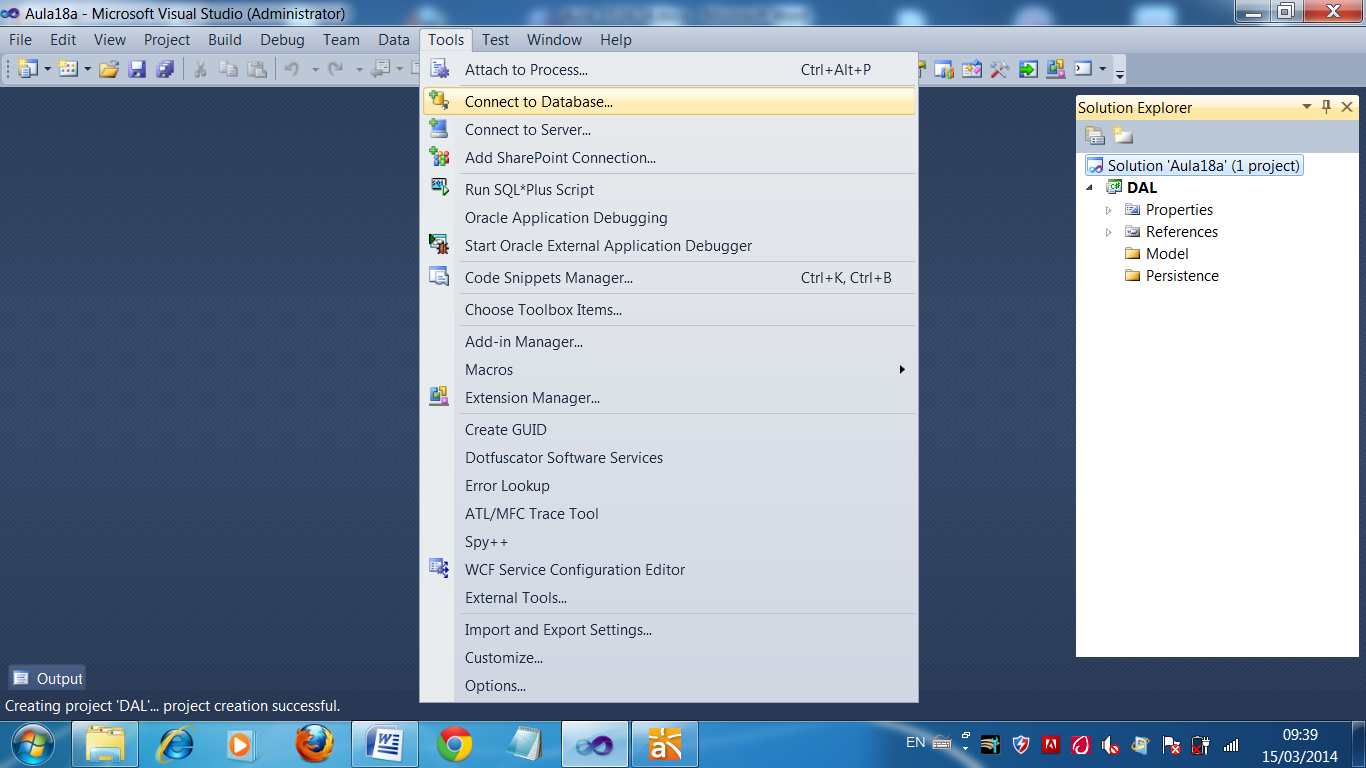
Parte 1:

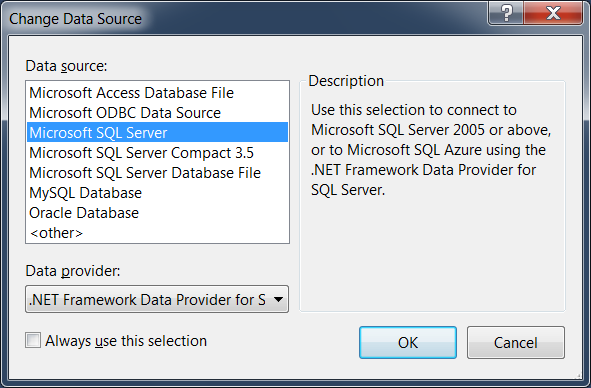
**Provedor do Serviço**

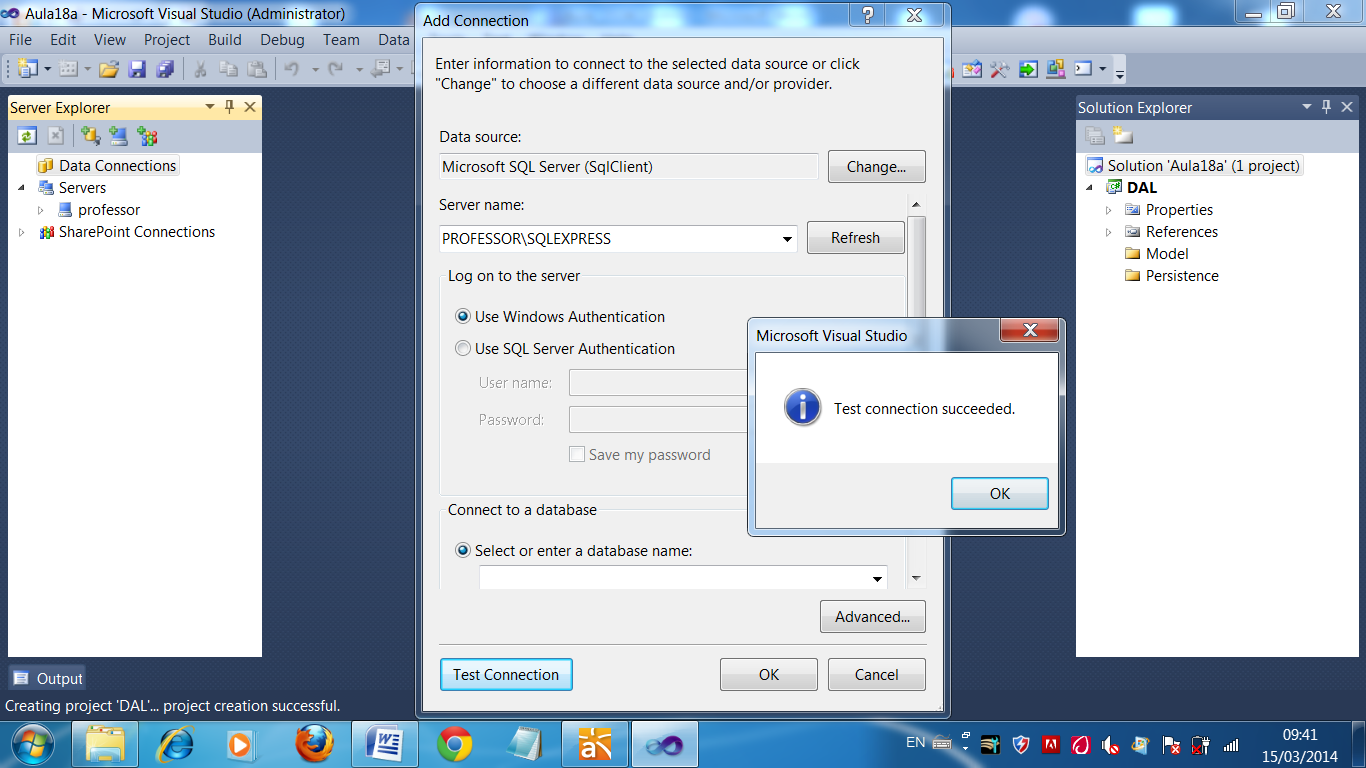


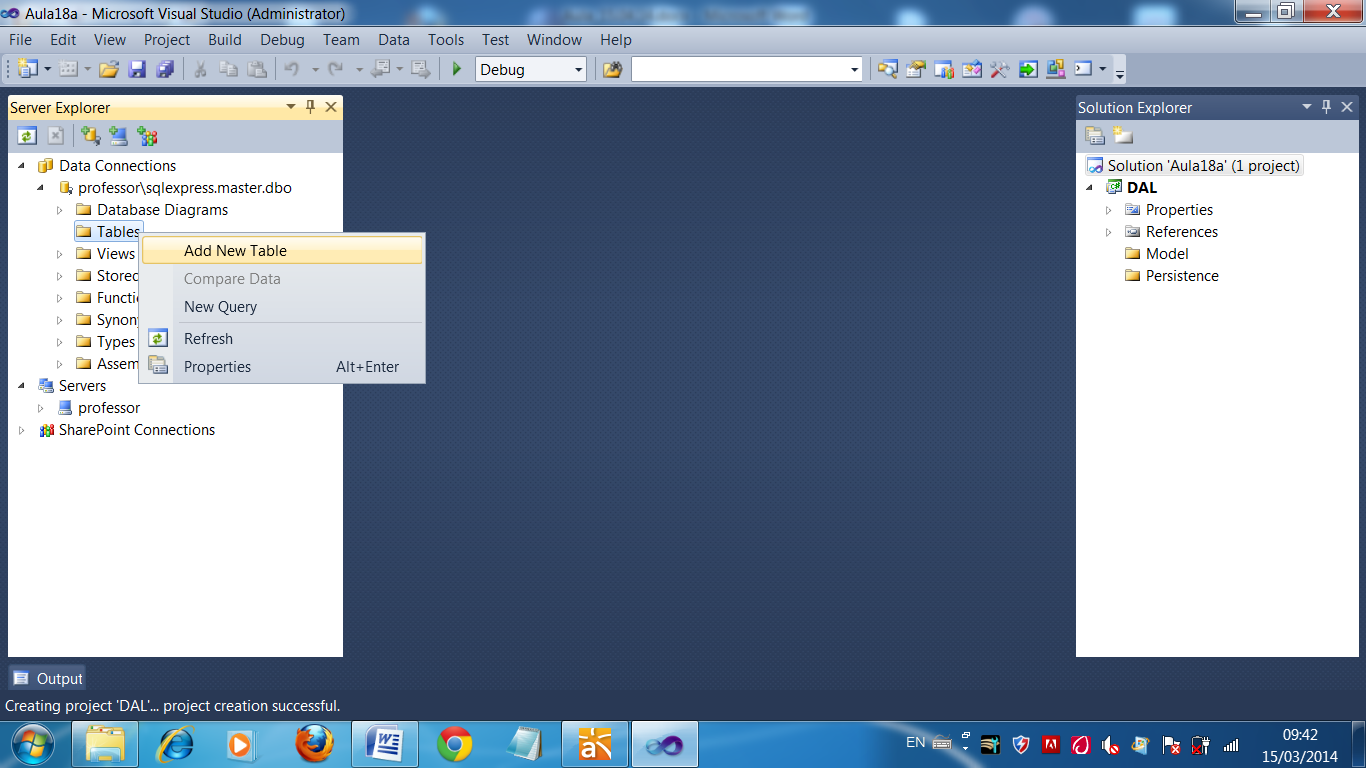


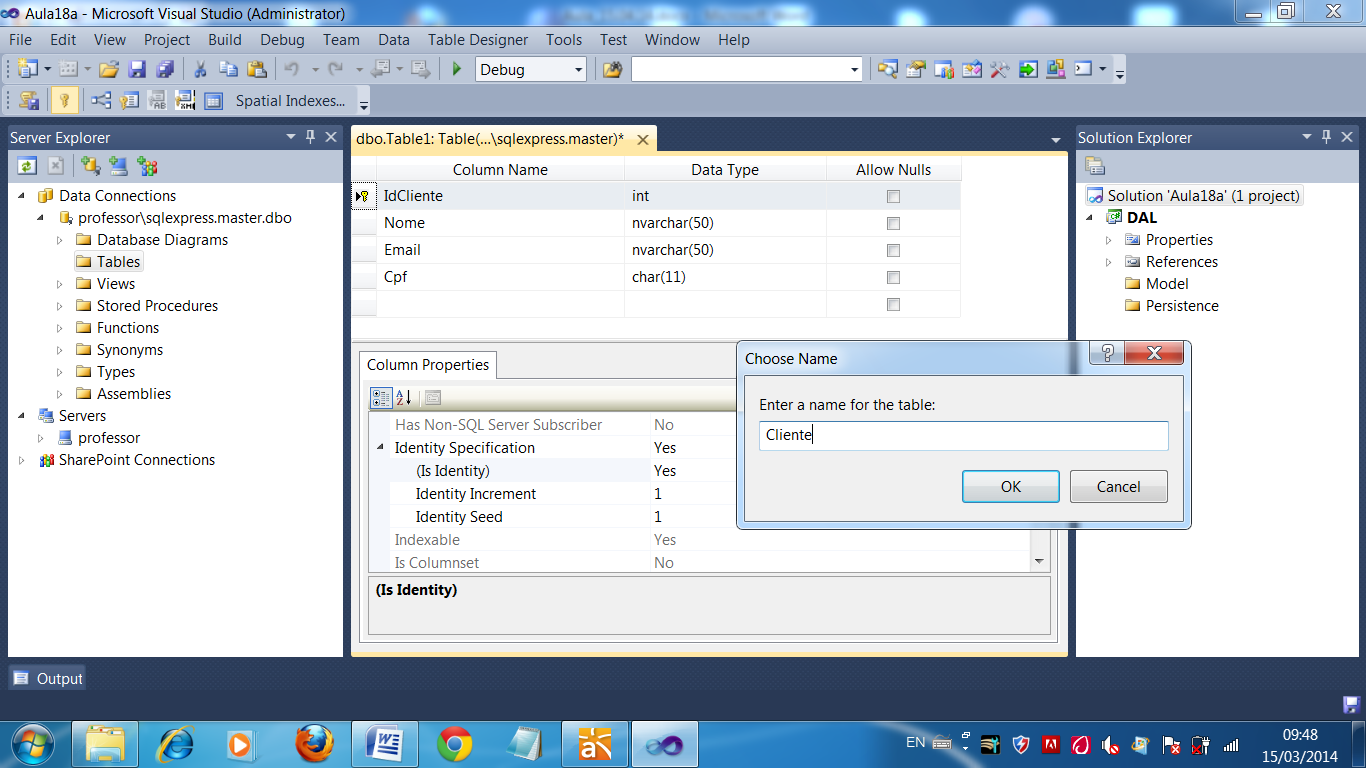


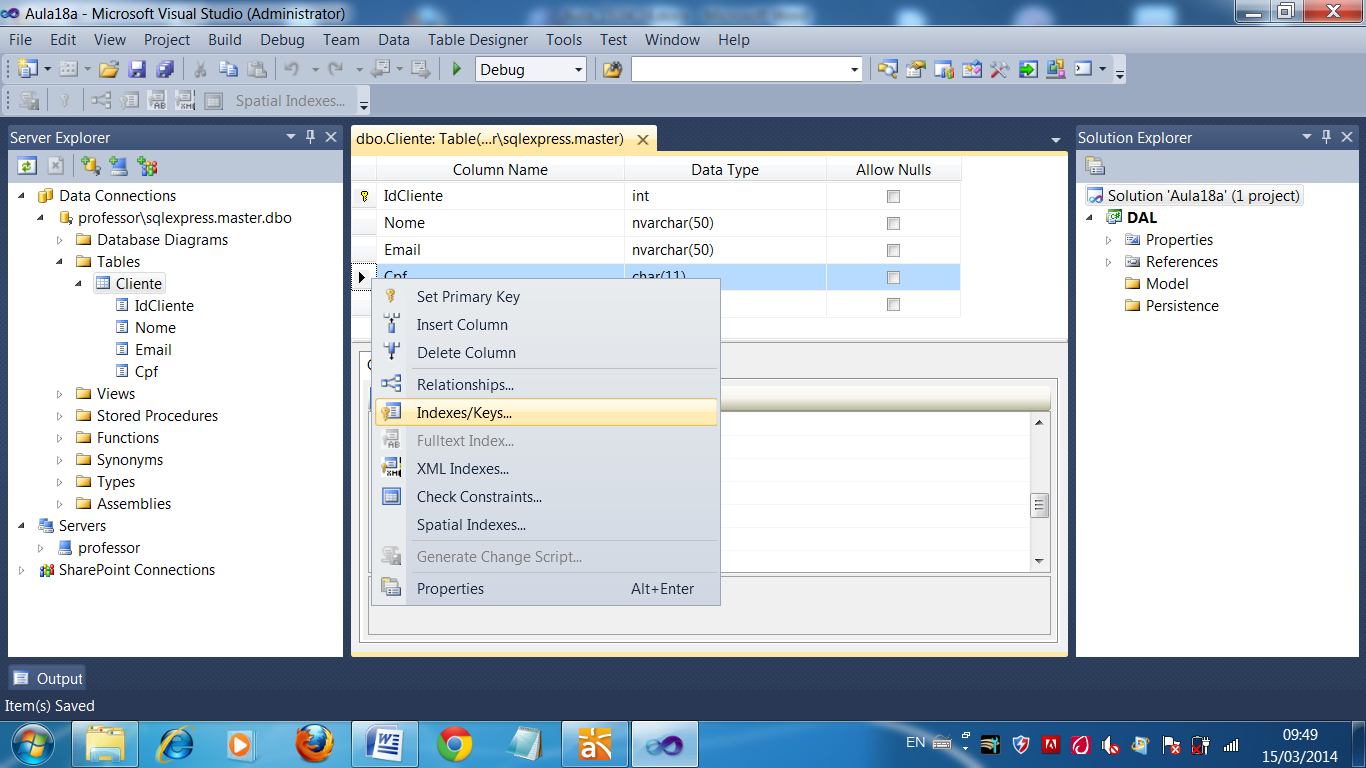


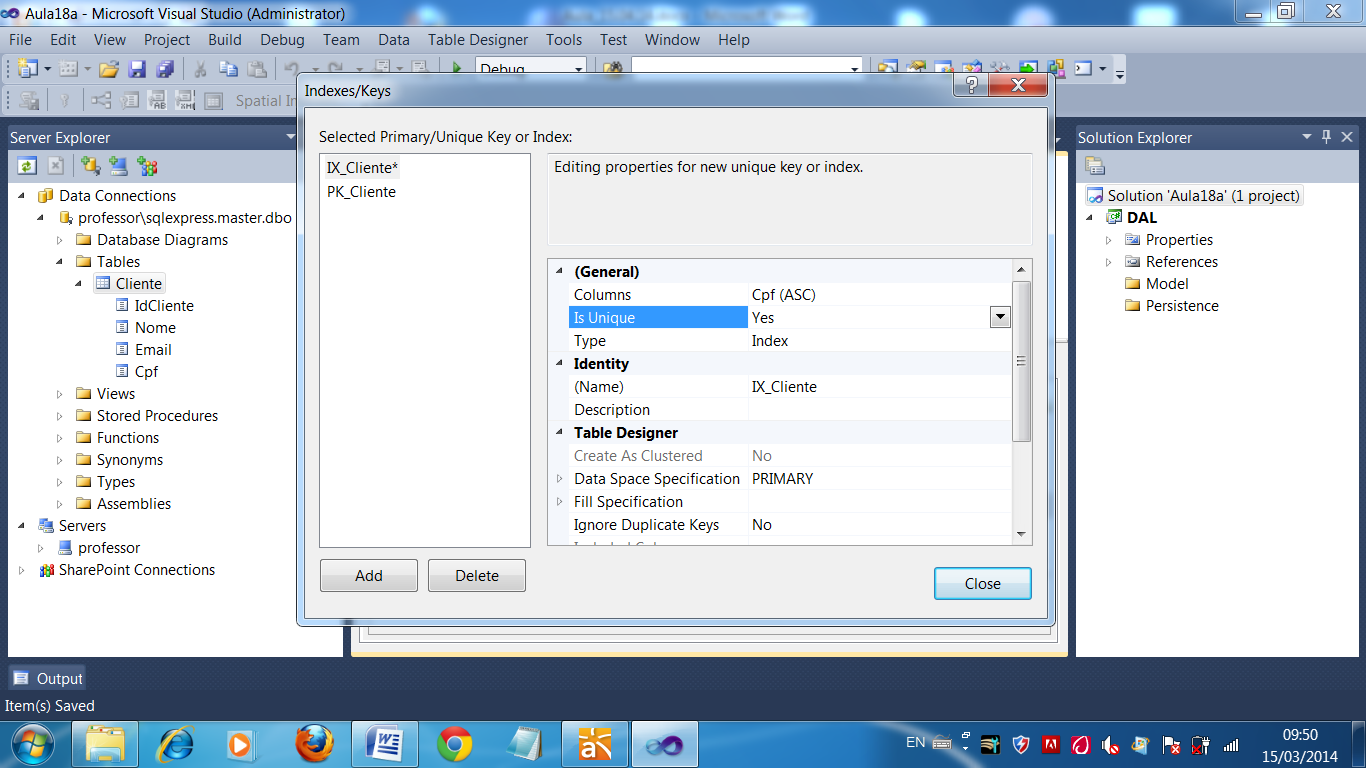












Modelagem da Classe Cliente...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace DAL.Model

{

public class Cliente

{

//Atributos

private int idCliente;

private string nome;

private string email;

private string cpf;

//Construtor padrão [ctor] + 2x[tab]

public Cliente()

{

//Default

}

//Sobrecarga de Métodos (Overloading)

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

{

this.idCliente = idCliente;

this.nome = nome;

this.email = email;

this.cpf = cpf;

}

public int IdCliente

{

set { idCliente = value; }

get { return idCliente; }

}

public string Nome

{

set { nome = value; }

get { return nome; }

}

public string Email

{

set { email = value; }

get { return email; }

}

public string Cpf

{

set { cpf = value; }

get { return cpf; }

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient; //acesso ao sqlserver

namespace DAL.Persistence

{

public class Conexao

{

//Atributos

protected SqlConnection Con; //Conexão com o SqlServer

protected SqlCommand Cmd; //Executar comandos SQL

protected SqlDataReader Dr; //Ler registros de consultas

protected SqlTransaction Tr; //Commit / Rollback (transações)

protected void AbrirConexao()

{

Con = new SqlConnection(@"Data Source=PROFESSOR\SQLEXPRESS;

Integrated Security=True");

Con.Open();

}

protected void FecharConexao()

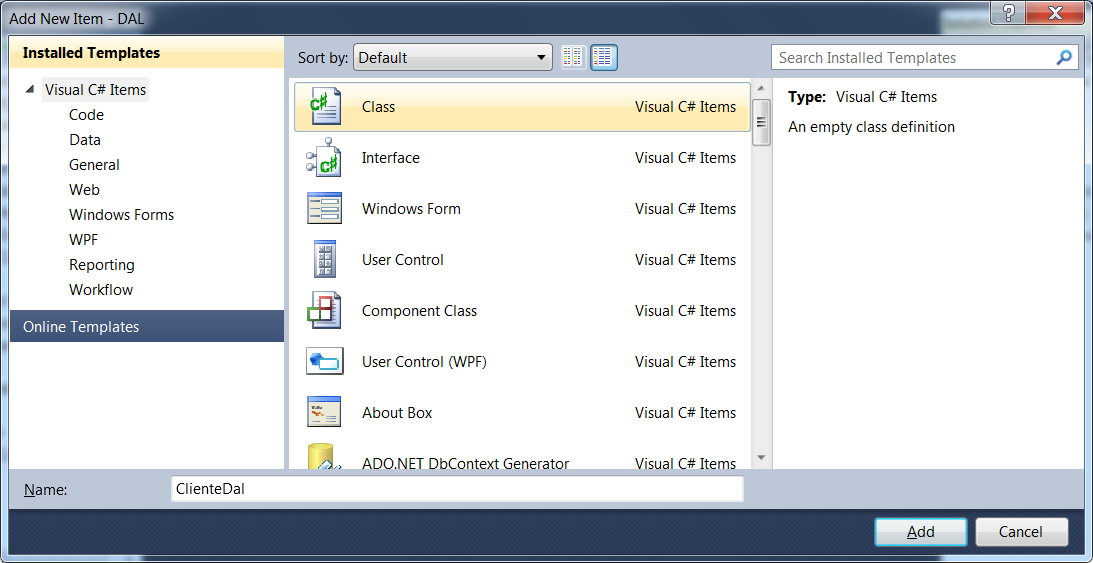
{

Con.Close();

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Data.SqlClient;

using DAL.Model;

namespace DAL.Persistence

{

public class ClienteDal : Conexao

{

//Método para salvar um cliente na tabela

public void Salvar(Cliente c)

{

try

{

AbrirConexao();

Tr = Con.BeginTransaction(); //iniciando uma transação

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

values(@v1, @v2, @v3)", Con, Tr);

Cmd.Parameters.AddWithValue("@v1", c.Nome);

Cmd.Parameters.AddWithValue("@v2", c.Email);

Cmd.Parameters.AddWithValue("@v3", c.Cpf);

Cmd.ExecuteNonQuery(); //escreve o comando Sql na base

Tr.Commit(); //executar a transação

}

catch(Exception e)

{

Tr.Rollback(); //desfaz a transação

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

}

finally

{

FecharConexao();

}

}

//Método para obter 1 Cliente pelo CPF

public Cliente ObterPorCpf(string cpf)

{

try

{

AbrirConexao();

Cmd = new SqlCommand("select \* from Cliente

where Cpf = @v1", Con);

Cmd.Parameters.AddWithValue("@v1", cpf);

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

if(Dr.Read())

{

Cliente c = new Cliente();

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

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

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

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

return c; //retornar o cliente

}

return null; //retornar vazio

}

catch(Exception e)

{

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

}

finally

{

FecharConexao();

}

}

}

}

Teste de Software

Qualidade do produto de software

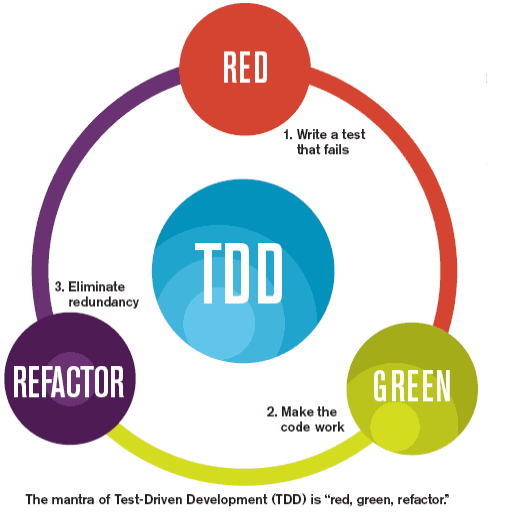
O principal aspecto para verificar e validar um software é a sua conformidade com os requisitos inicialmente estabelecidos.

* Tipos de Testes:
  + Teste de Caixa Branca
  + Teste de Caixa Preta

Teste Unitário

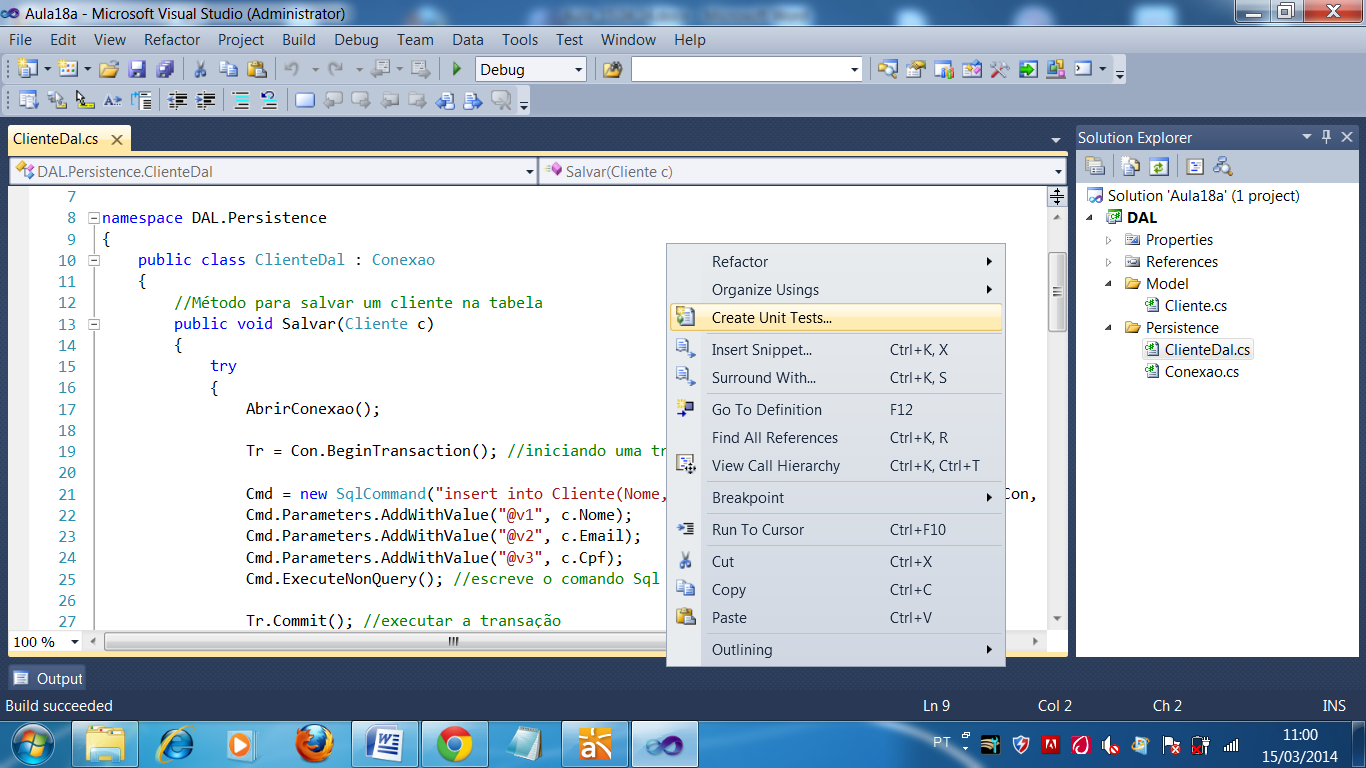
Tipo de teste de **caixa-branca** utilizado para verificar e validar cada classe, método ou unidade de um programa escrito pelo desenvolvedor.

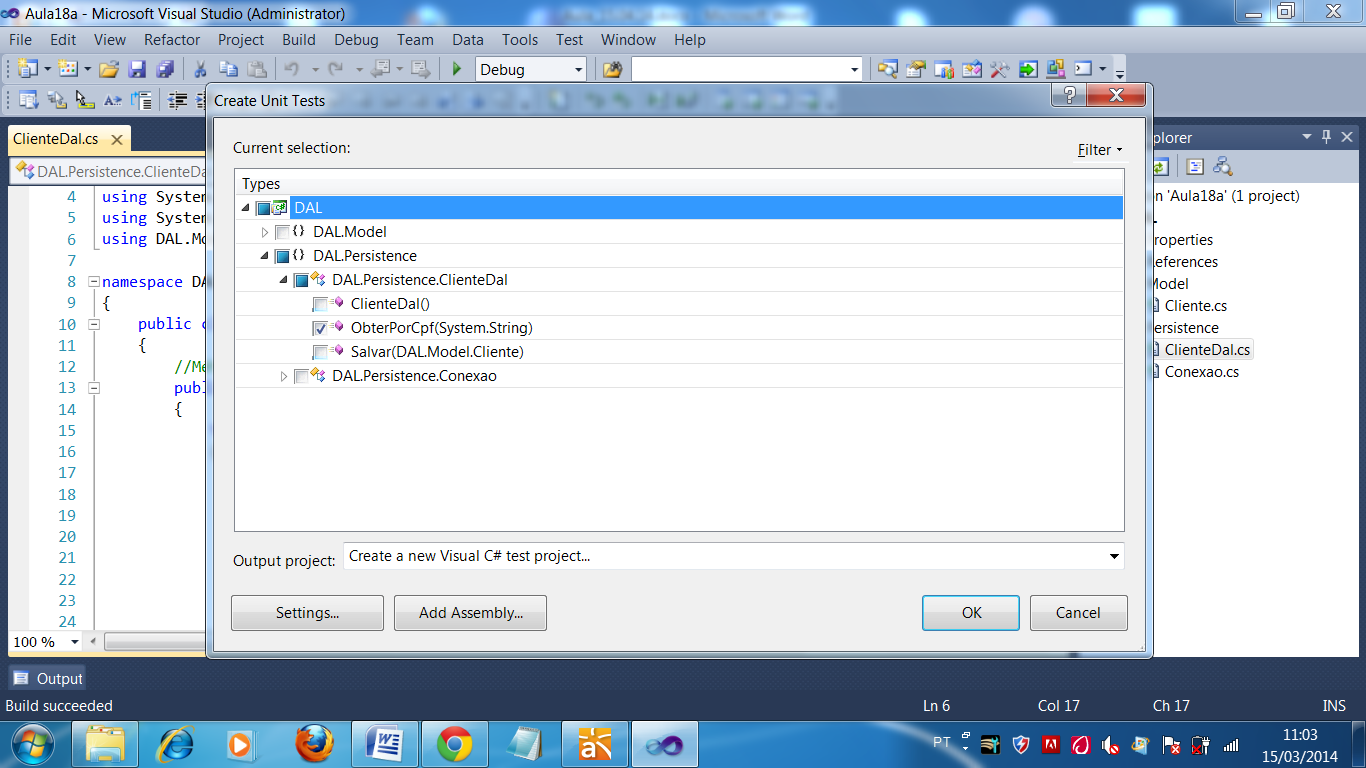
**Evolução: TDD (Desenvolvimento Dirigido a Testes)**

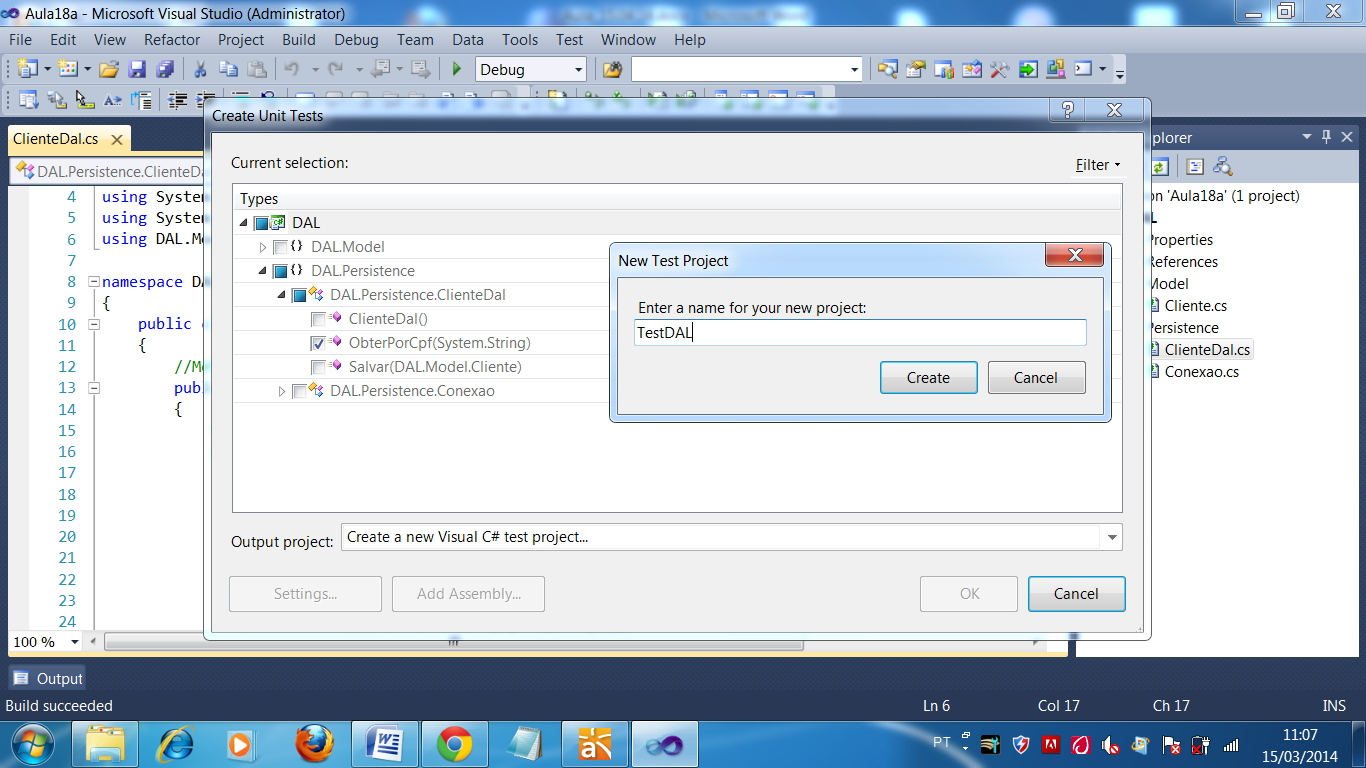


UnitTest

Framework do .NET voltado para implementação de testes unitários.







Roteiro de Teste

* **Pesquisar Cliente**
  + Dados de Entrada:
    - Nome do Cliente: Sergio Mendes
    - Email: [sergio@gmail.com](mailto:sergio@gmail.com)
    - Cpf: 0123456789
  + Passo:
    - Cadastrar o Cliente
    - Buscar o Cliente pelo Cpf
  + Resultado Esperado
    - Cliente encontrado

using DAL.Persistence;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using DAL.Model;

namespace TestDAL

{

/// <summary>

///This is a test class for ClienteDalTest and is intended

///to contain all ClienteDalTest Unit Tests

///</summary>

[TestClass()]

public class ClienteDalTest

{

private TestContext testContextInstance;

/// <summary>

///Gets or sets the test context which provides

///information about and functionality for the current test run.

///</summary>

public TestContext TestContext

{

get

{

return testContextInstance;

}

set

{

testContextInstance = value;

}

}

#region Additional test attributes

//

//You can use the following additional attributes

//as you write your tests:

//

//Use ClassInitialize to run code before running the

//first test in the class

//[ClassInitialize()]

//public static void MyClassInitialize(TestContext testContext)

//{

//}

//

//Use ClassCleanup to run code after all tests in a class have run

//[ClassCleanup()]

//public static void MyClassCleanup()

//{

//}

//

//Use TestInitialize to run code before running each test

//[TestInitialize()]

//public void MyTestInitialize()

//{

//}

//

//Use TestCleanup to run code after each test has run

//[TestCleanup()]

//public void MyTestCleanup()

//{

//}

//

#endregion

/// <summary>

///A test for ObterPorCpf

///</summary>

[TestMethod()]

public void PesquisarClienteTest()

{

try

{

//Dados de Entrada

Cliente entrada = new Cliente();

entrada.Nome = "Sergio Mendes";

entrada.Email = "sergio.coti@gmail.com";

entrada.Cpf = "0123456789";

//Passos:

ClienteDal d = new ClienteDal();

d.Salvar(entrada); //Salvar o Cliente

Cliente saida = d.ObterPorCpf(entrada.Cpf);

//Verificar o resultado obtido

Assert.AreEqual(entrada.Nome, saida.Nome);

Assert.AreEqual(entrada.Email, saida.Email);

Assert.AreEqual(entrada.Cpf, saida.Cpf);

}

catch(Exception e)

{

//Registrar Falha no Teste

Assert.Fail("Falha no Teste de Pesquisa de Cliente: "

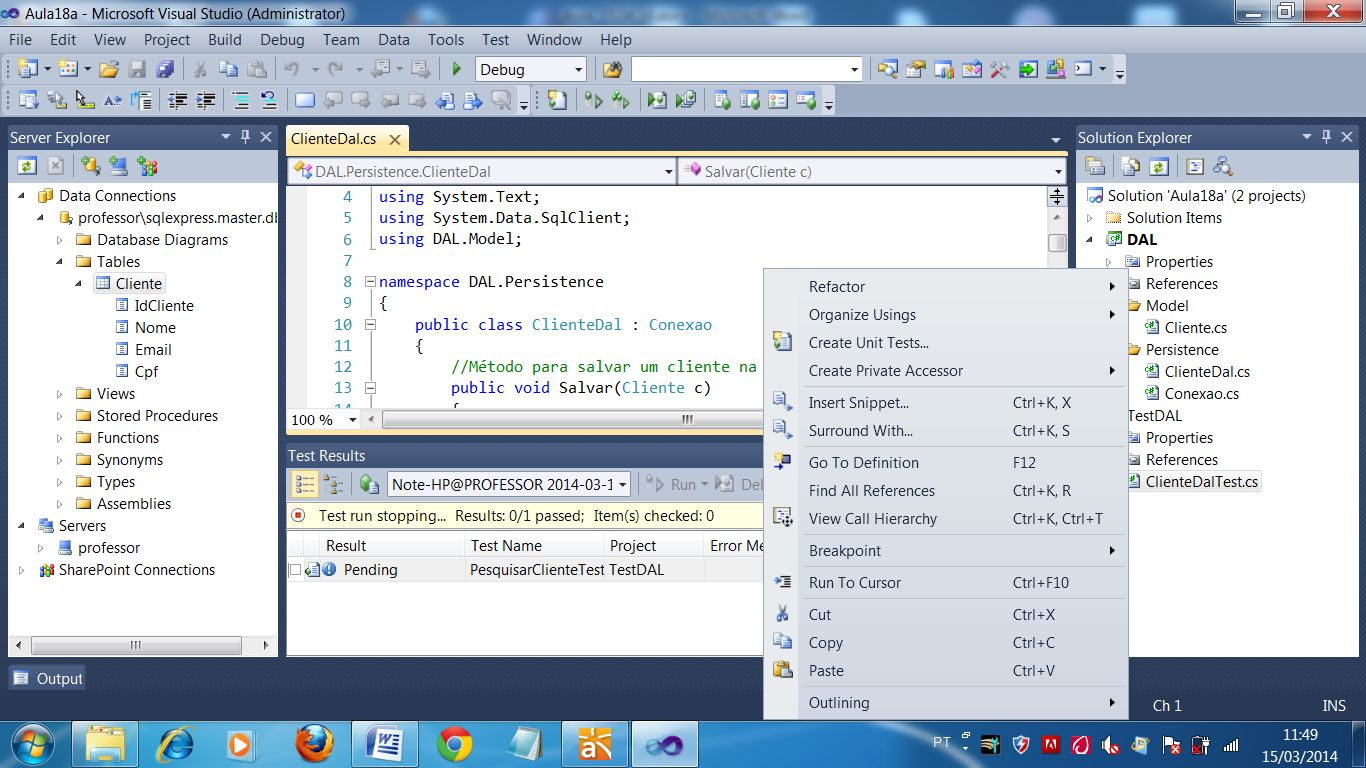
+ e.Message);

}

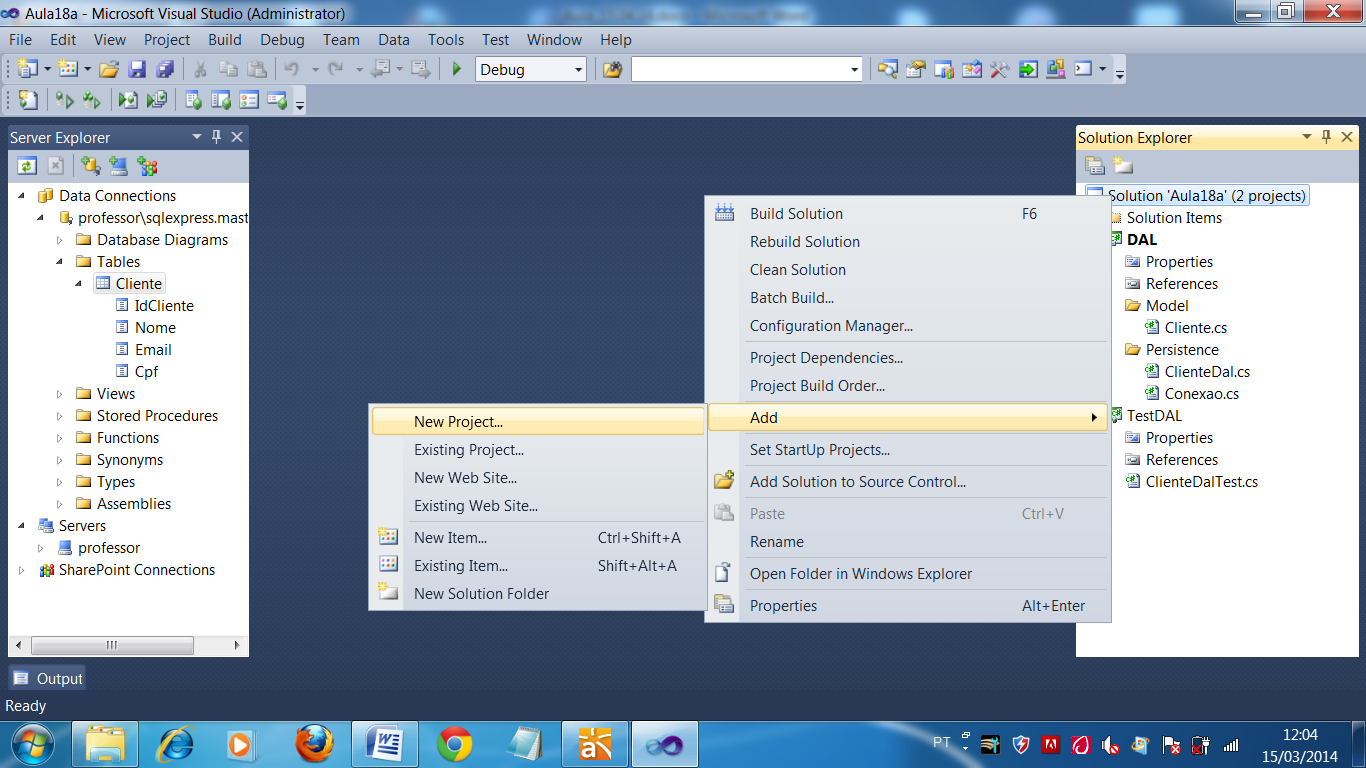
}

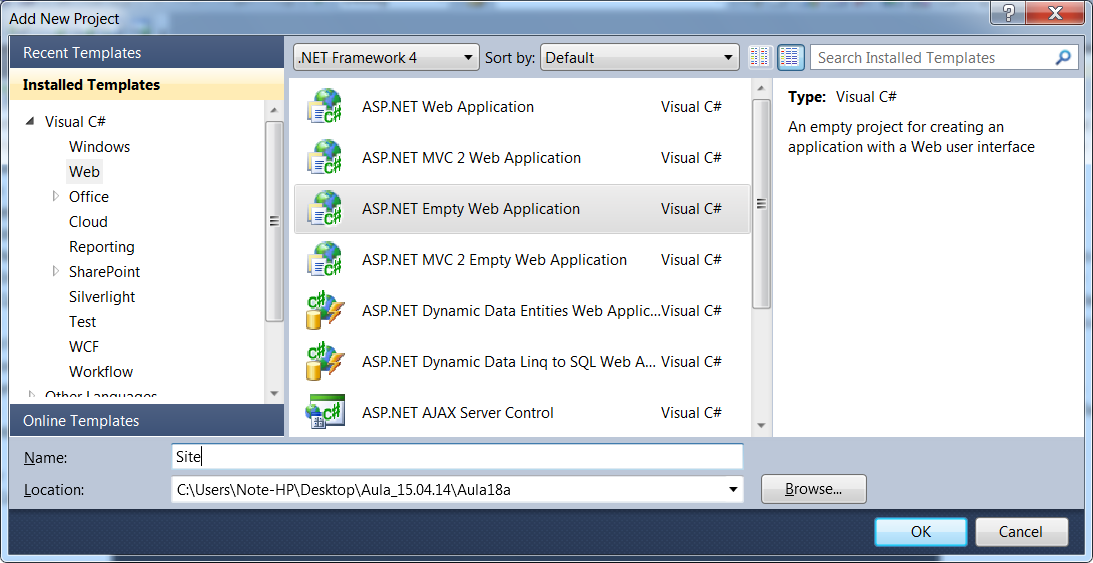
}

}

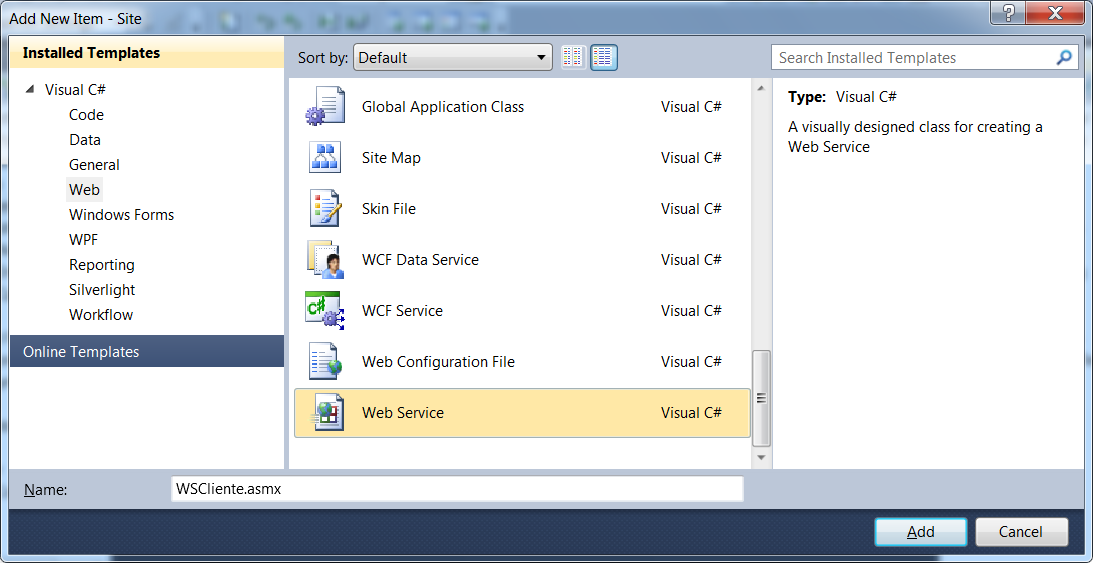


Criando o Serviço





Criando o WebService...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Services;

using DAL.Model;

using DAL.Persistence;

namespace Site.Services

{

/// <summary>

/// Summary description for WSCliente

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[System.ComponentModel.ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

// [System.Web.Script.Services.ScriptService]

public class WSCliente : System.Web.Services.WebService

{

[WebMethod]

public string PesquisarCliente(string Cpf)

{

try

{

ClienteDal d = new ClienteDal();

Cliente c = d.ObterPorCpf(Cpf); //busca

if (c != null)

{

return string.Format("{0};{1};{2};{3}", c.IdCliente,

c.Nome, c.Email, c.Cpf);

}

else //não foi encontrado!

{

return "NOT\_FOUND";

}

}

catch

{

return "ERROR";

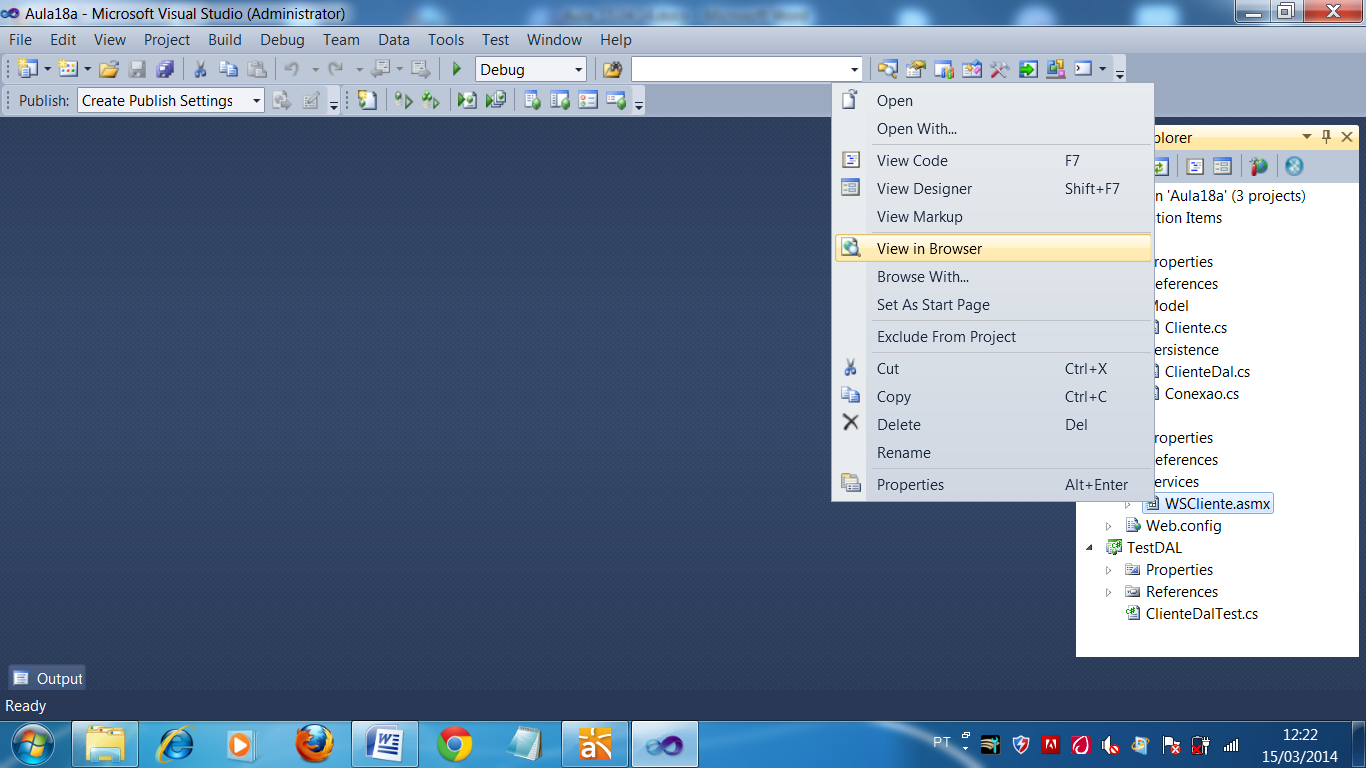
}

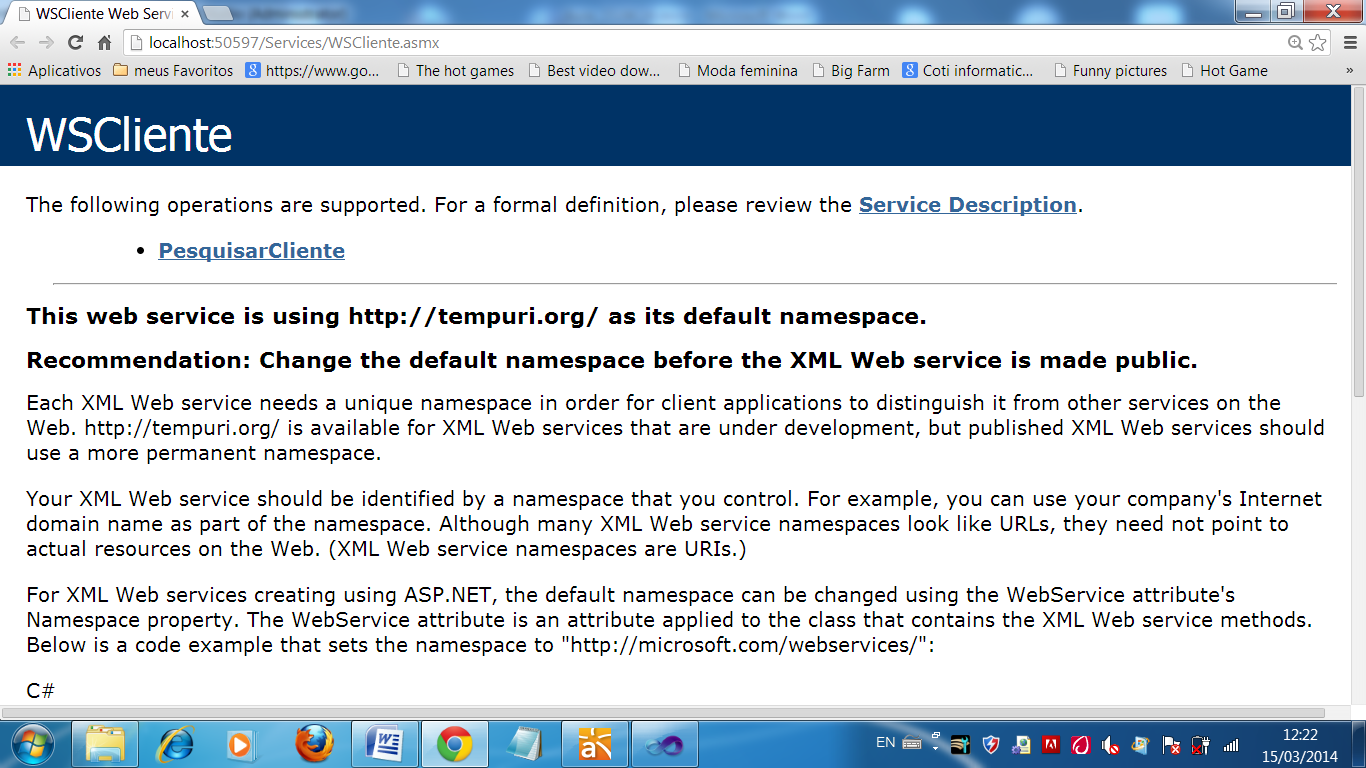
}

}

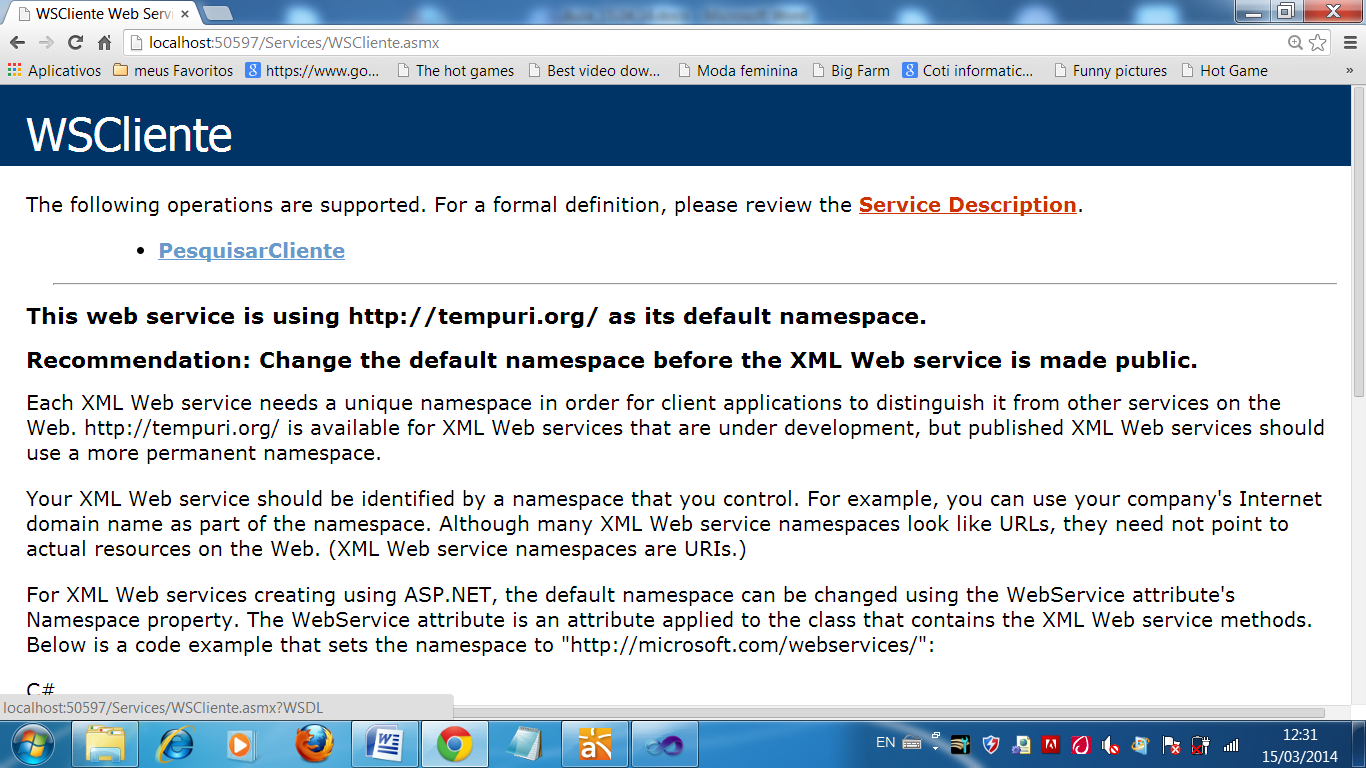
}

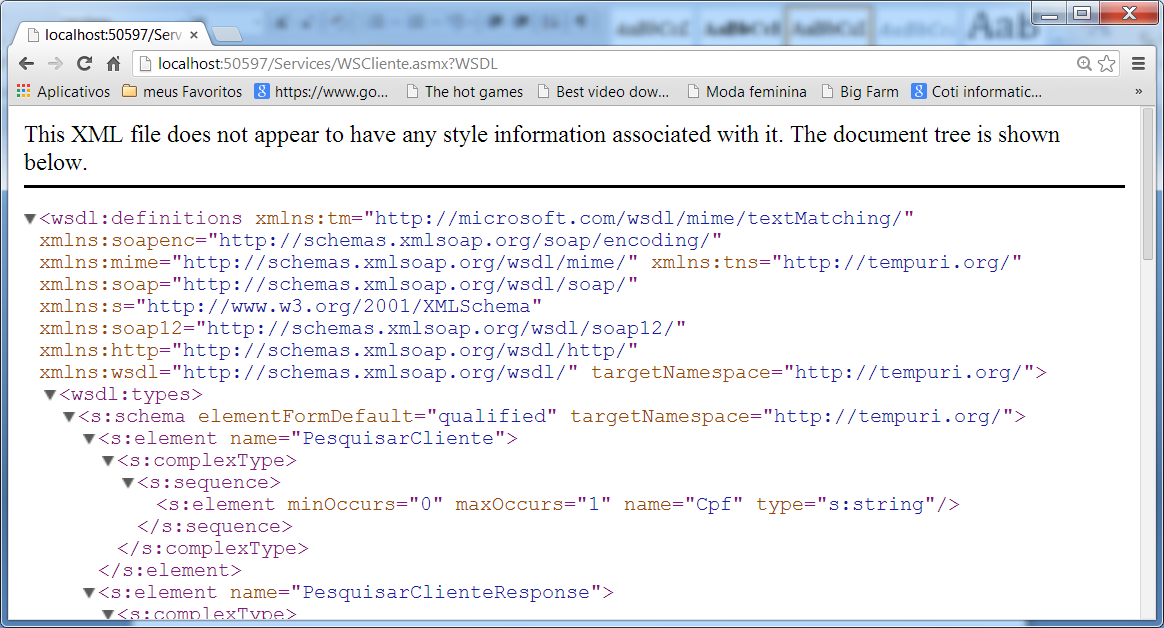
* Testar o Serviço





Para que possamos acessar um serviço web, é necessário obter o endereço deste serviço na forma de um documento XML denominado WSDL (WebService Description Language)

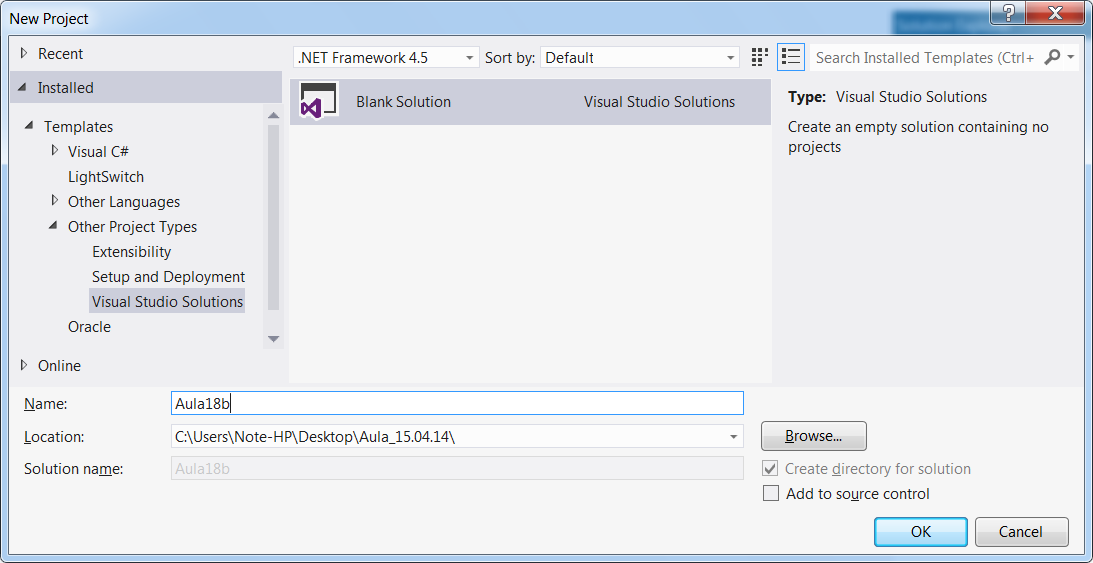


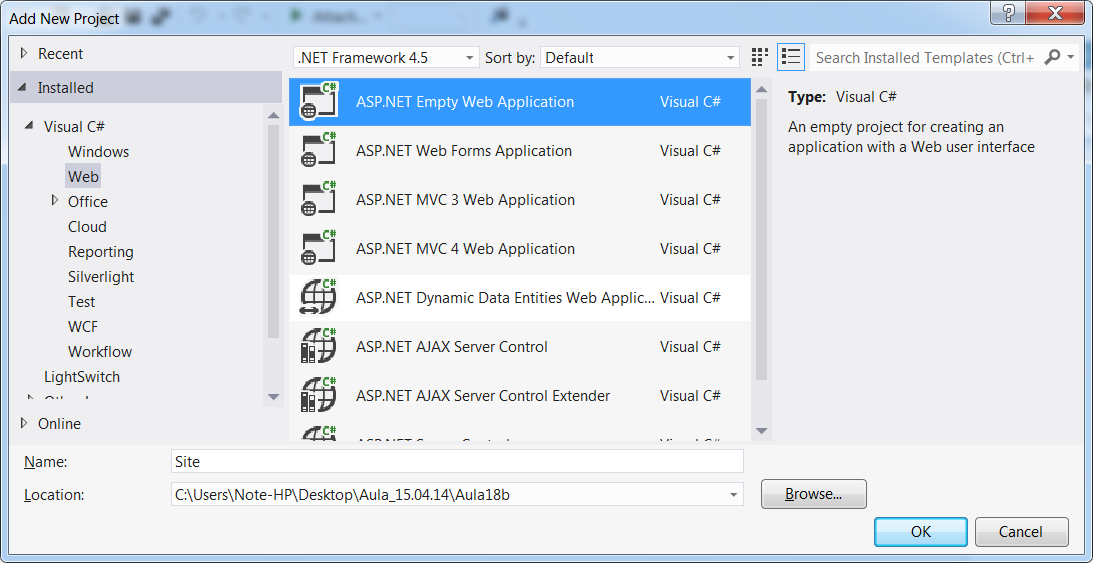


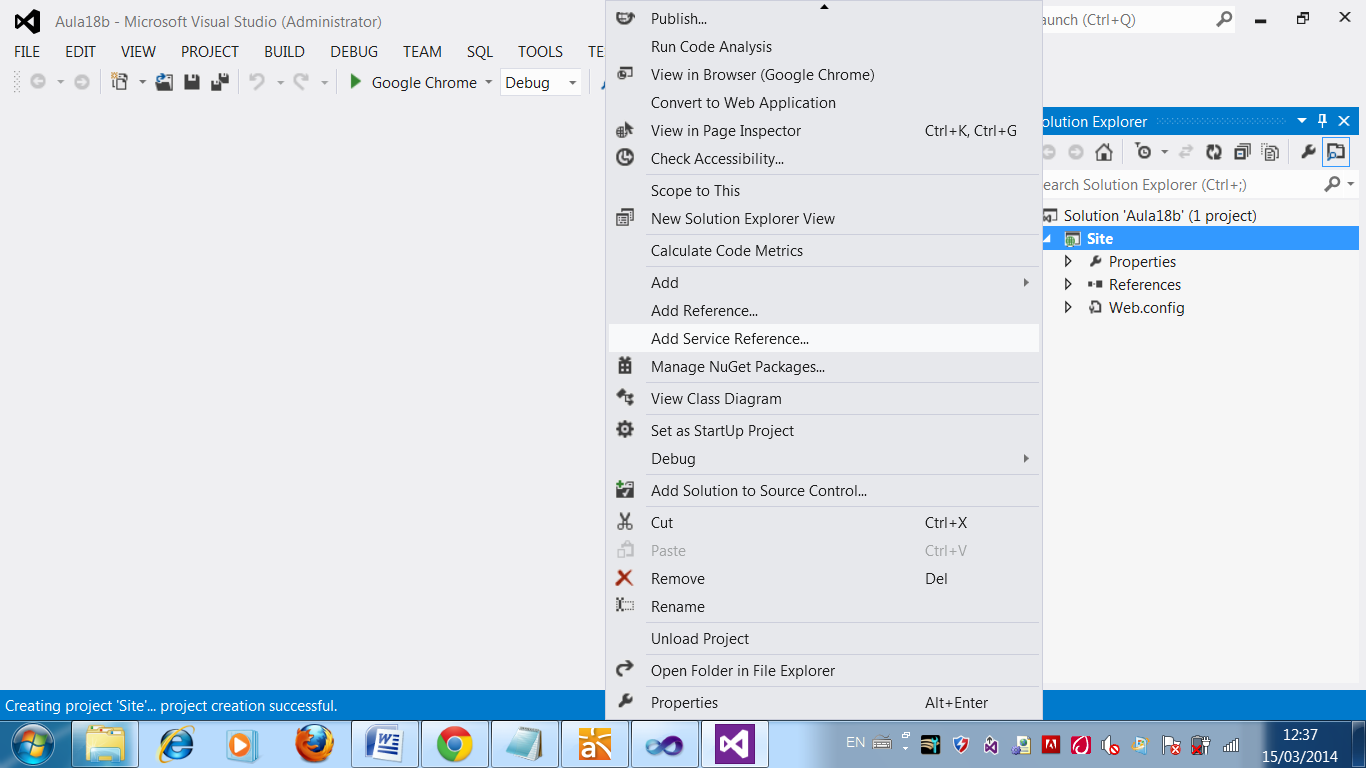
URL do Serviço

<http://localhost:50597/Services/WSCliente.asmx?WSDL>

Criando o Cliente do Serviço...

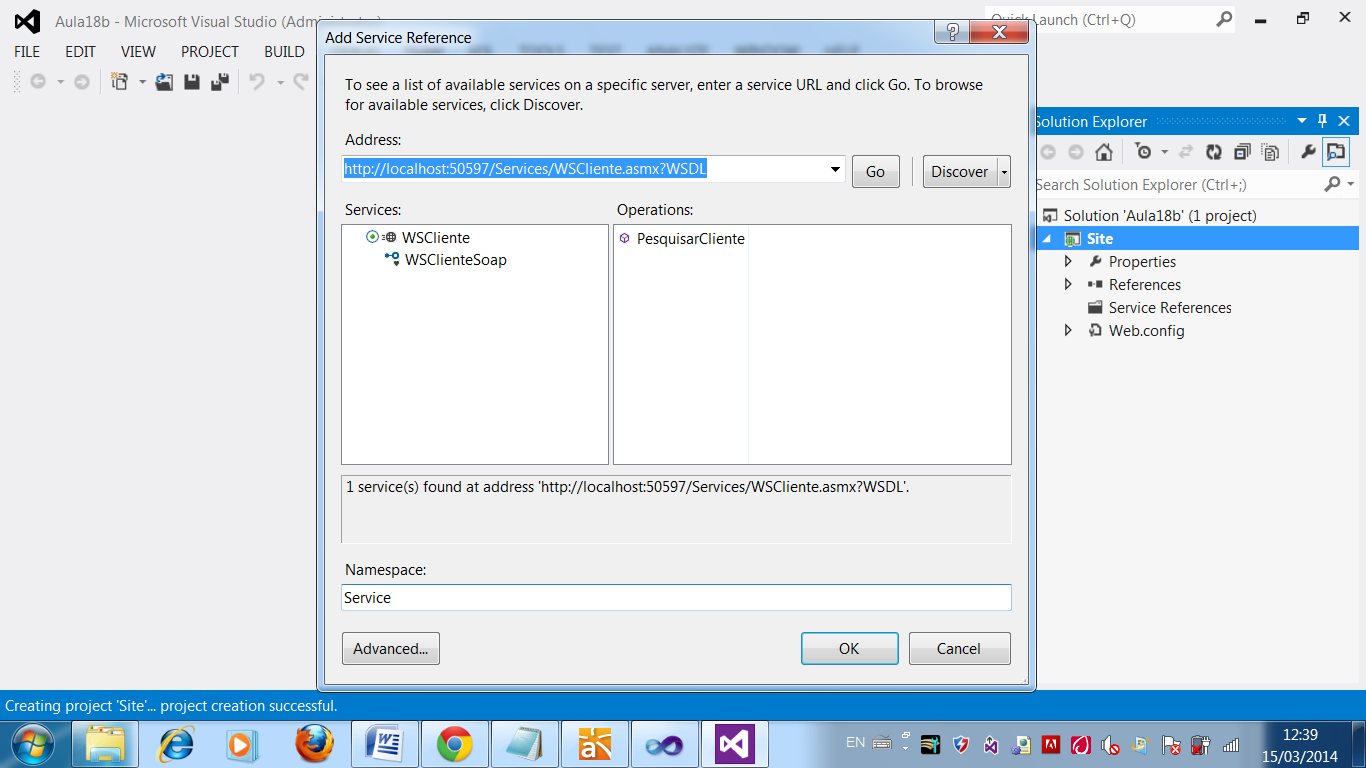


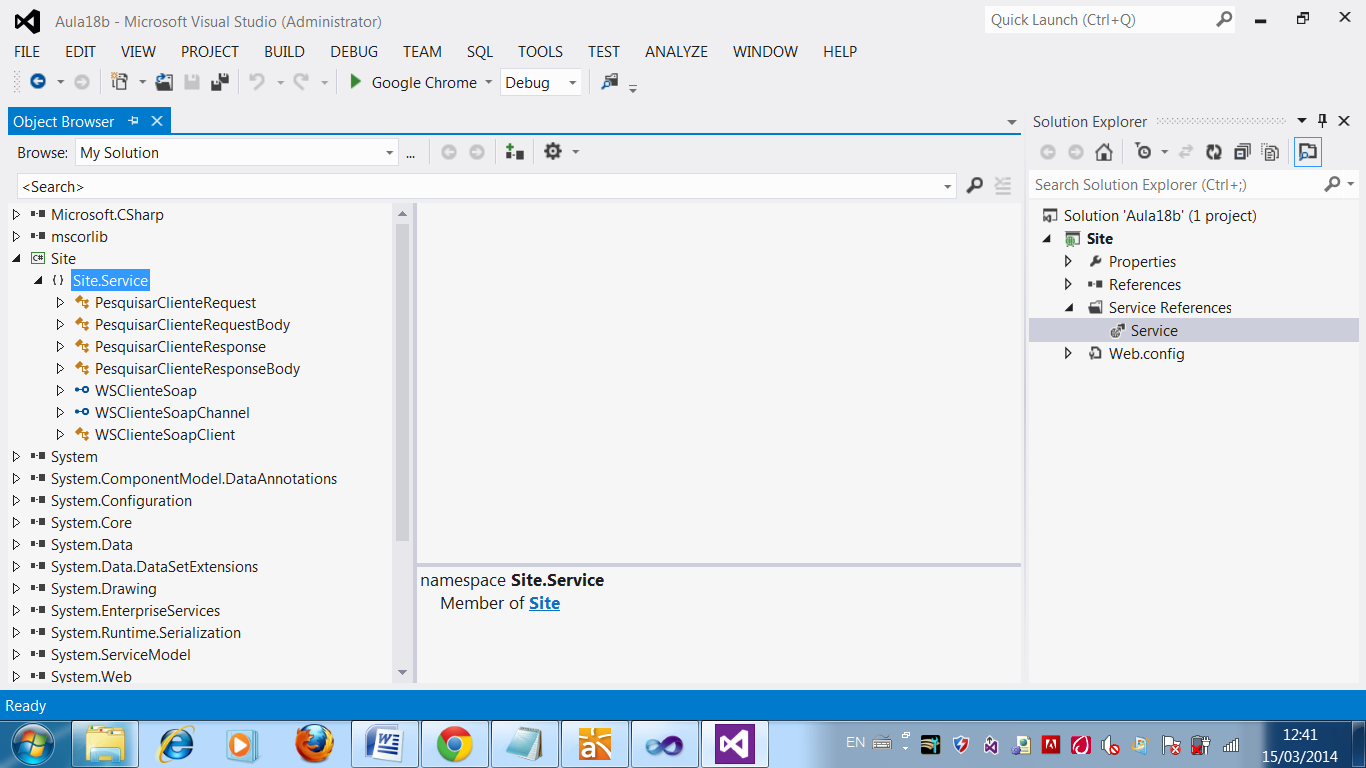


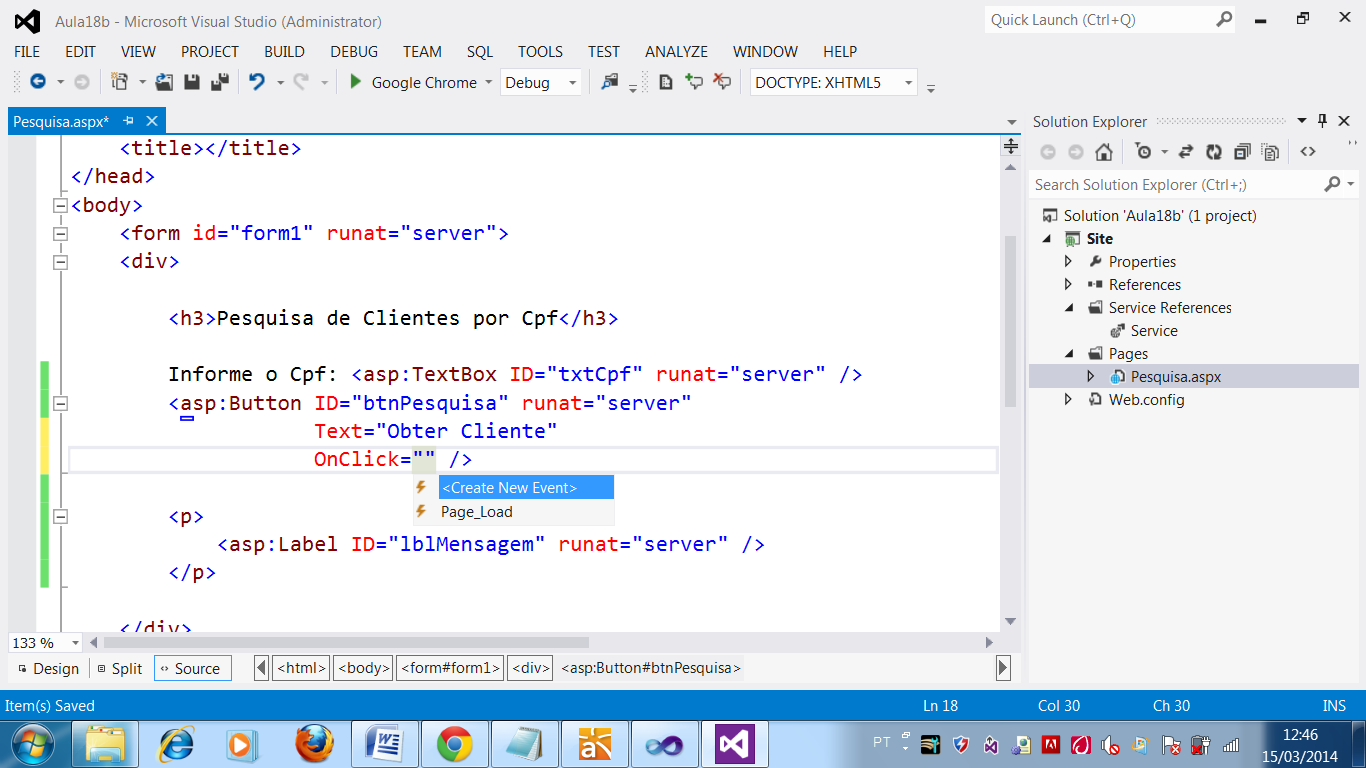


SOAP – Simple Object Access Protocol

Nome do protocolo utilizado para acesso a serviços web.







<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Pesquisa.aspx.cs" Inherits="Site.Pages.Pesquisa" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>Pesquisa de Clientes por Cpf</h3>

Informe o Cpf: <asp:TextBox ID="txtCpf" runat="server" />

<asp:Button ID="btnPesquisa" runat="server"

Text="Obter Cliente"

OnClick="btnPesquisa\_Click" />

<p>

<asp:Label ID="lblMensagem" runat="server" />

</p>

</div>

</form>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using Site.Service;

namespace Site.Pages

{

public partial class Pesquisa : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void btnPesquisa\_Click(object sender, EventArgs e)

{

try

{

WSClienteSoapClient ws = new WSClienteSoapClient();

string resultado = ws.PesquisarCliente(txtCpf.Text);

if(resultado.Equals("ERROR"))

{

lblMensagem.Text = "Erro ao obter cliente.";

}

else if (resultado.Equals("NOT\_FOUND"))

{

lblMensagem.Text = "Cliente não encontrado.";

}

else

{

lblMensagem.Text = "Cliente: " + resultado;

}

}

catch(Exception ex)

{

lblMensagem.Text = ex.Message;

}

}

}

}

