

WCF - Windows Communication Foundation

Tecnologia para desenvolvimento de aplicações .NET baseadas em webservices. Ou seja, serviços web publicados na forma de documentos XML e que utilizam o protocolo SOAP para fazer a comunicação cliente / servidor.

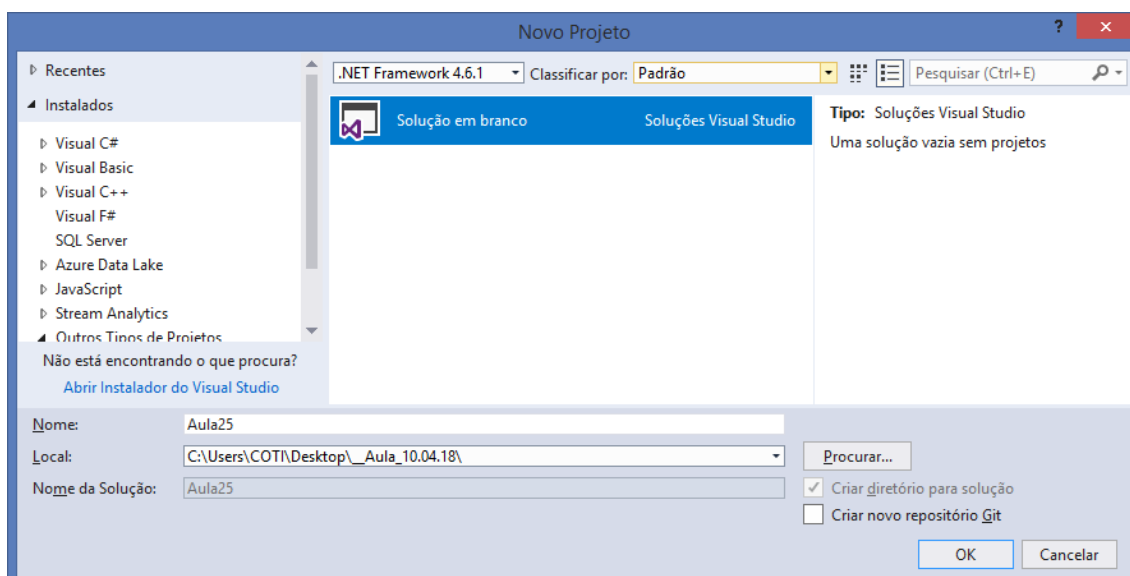
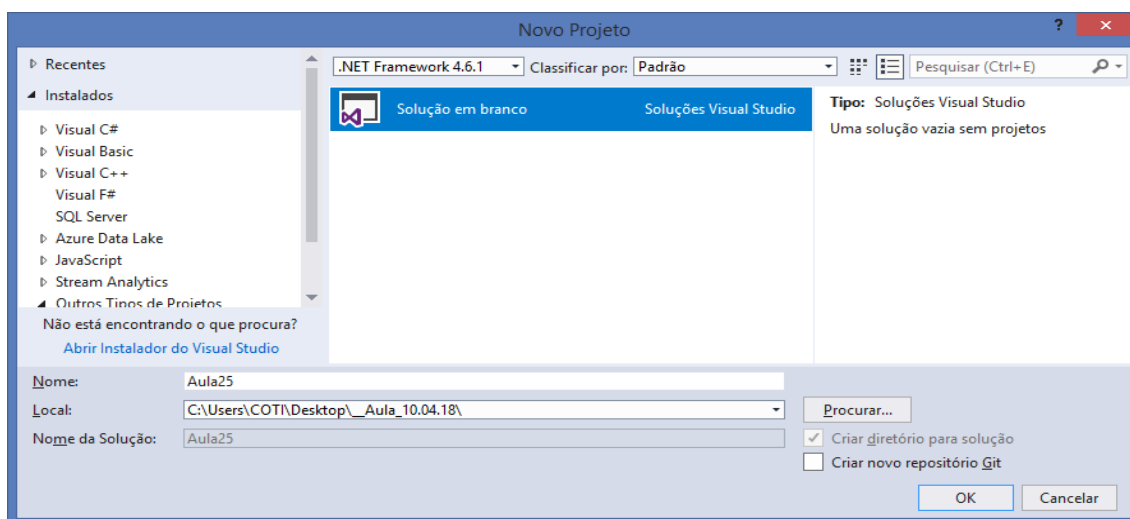
Comparativo entre **WebServices** e **WebApi**

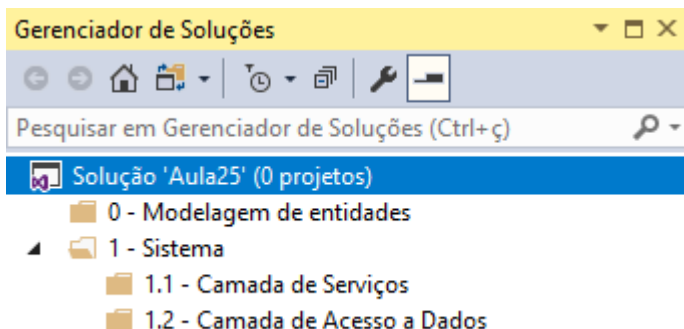
WebServices

Protocolo SOAP
Formato de dados XML
Integração deverá ser feita com outra aplicação de backend (Java, .NET, PHP, etc)

WebApi

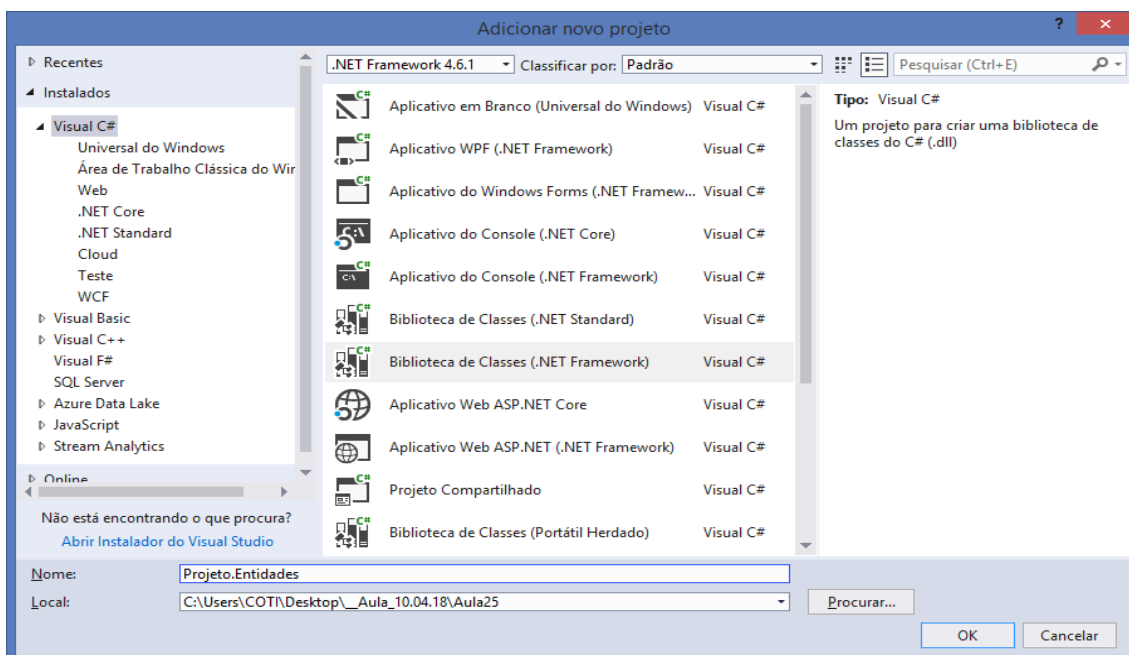
Protocolo HTTP
Formato de dados JSON
Permite integração diretamente com o frontend (javascript por exemplo)





0 - Modelagem de entidades

Biblioteca de Classes .NET Framework

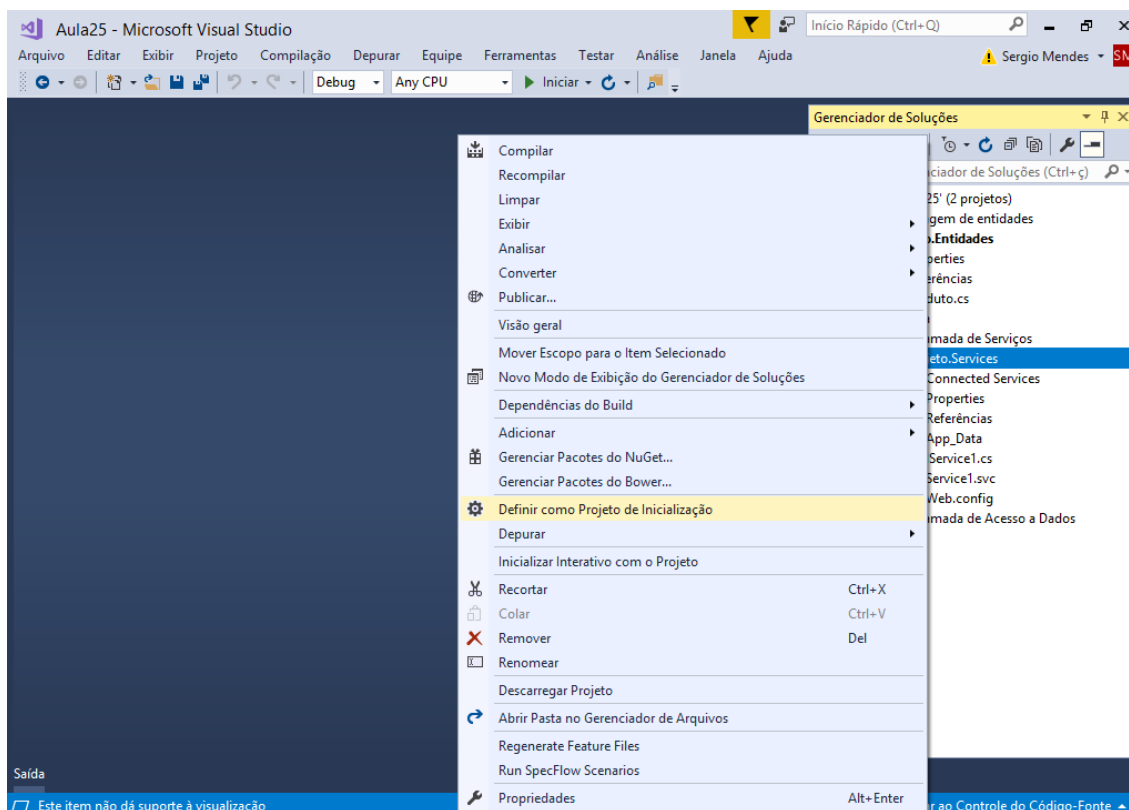
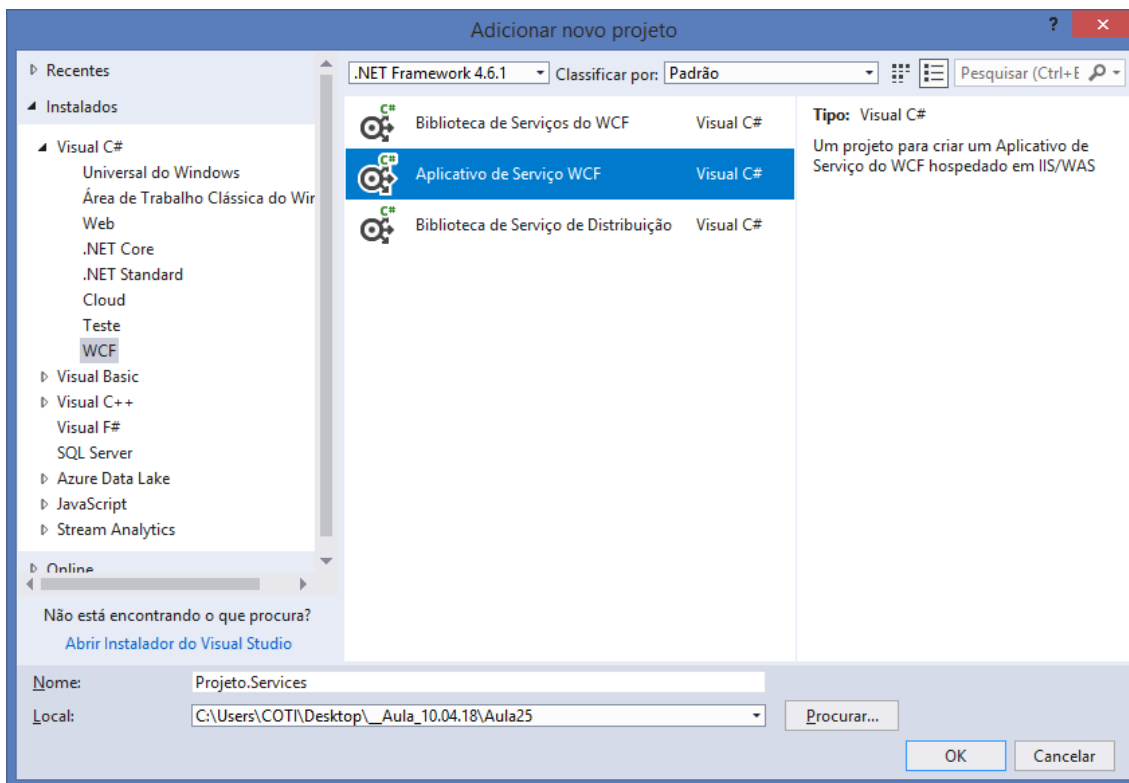


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projeto.Entidades
{
    public class Produto
    {
        //propriedades
        public virtual int IdProduto { get; set; }
        public virtual string Nome { get; set; }
        public virtual decimal Preco { get; set; }
        public virtual int Quantidade { get; set; }
    }
}
```

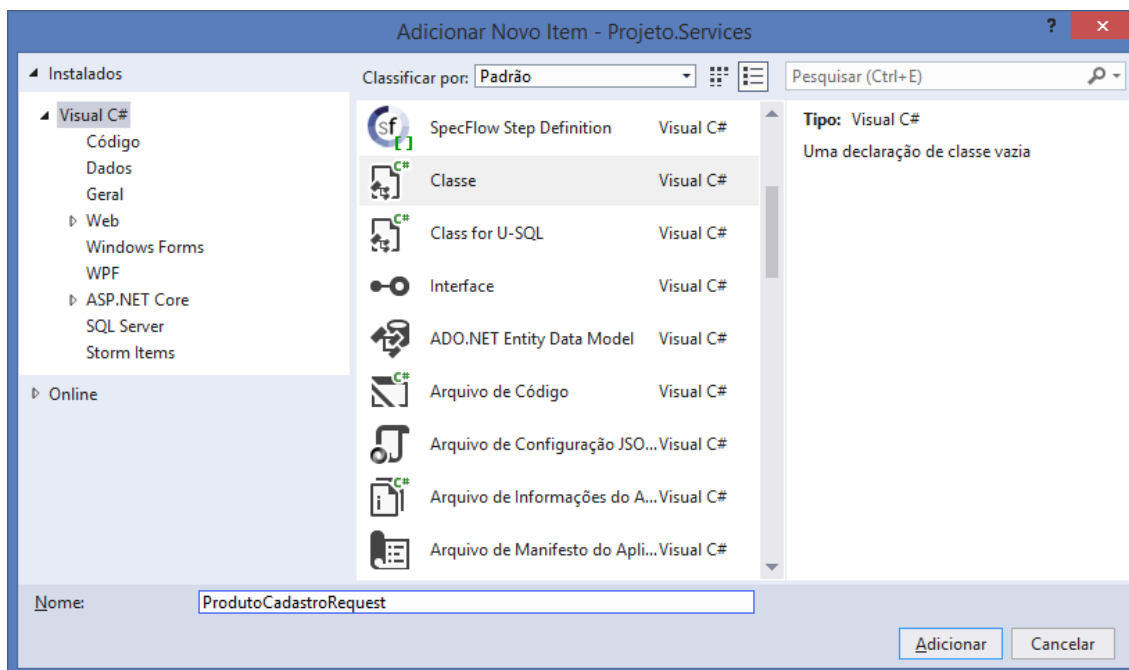
1.1 - Camada de Serviços

Projeto WCF - Windows Communication Foundation



Criando serviços:

Primeiro passo: Criando classes de modelo para operação de serviço. Basicamente, todo serviço tem uma entrada de dados (Request) e uma saída de dados (response)

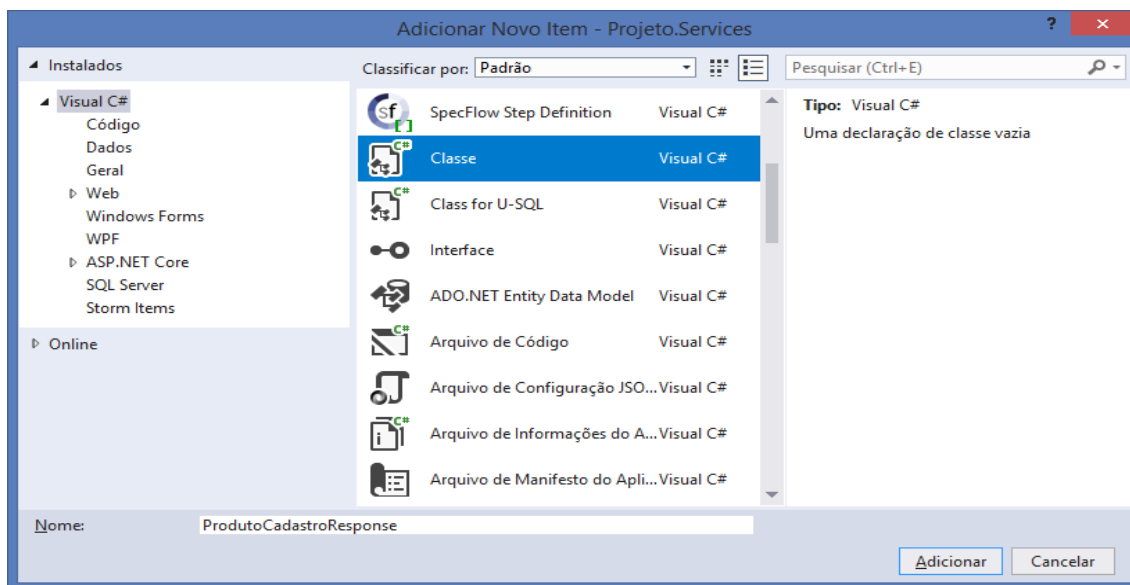


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Runtime.Serialization;

namespace Projeto.Services.Models
{
    [DataContract]
    public class ProdutoCadastroRequest
    {
        [DataMember(IsRequired = true)]
        public string Nome { get; set; }

        [DataMember(IsRequired = true)]
        public decimal Preco { get; set; }

        [DataMember(IsRequired = true)]
        public int Quantidade { get; set; }
    }
}
```

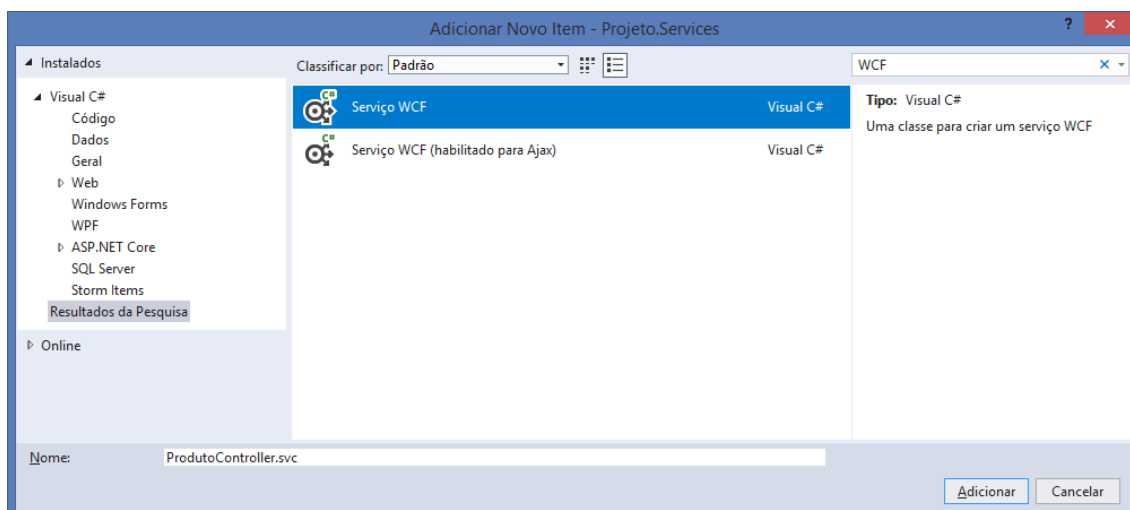


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Runtime.Serialization;

namespace Projeto.Services.Models
{
    [DataContract]
    public class ProdutoCadastroResponse
    {
        [DataMember]
        public string Status { get; set; }

        [DataMember]
        public string Mensagem { get; set; }
    }
}
```

Criando uma classe WCF para programar os serviços:



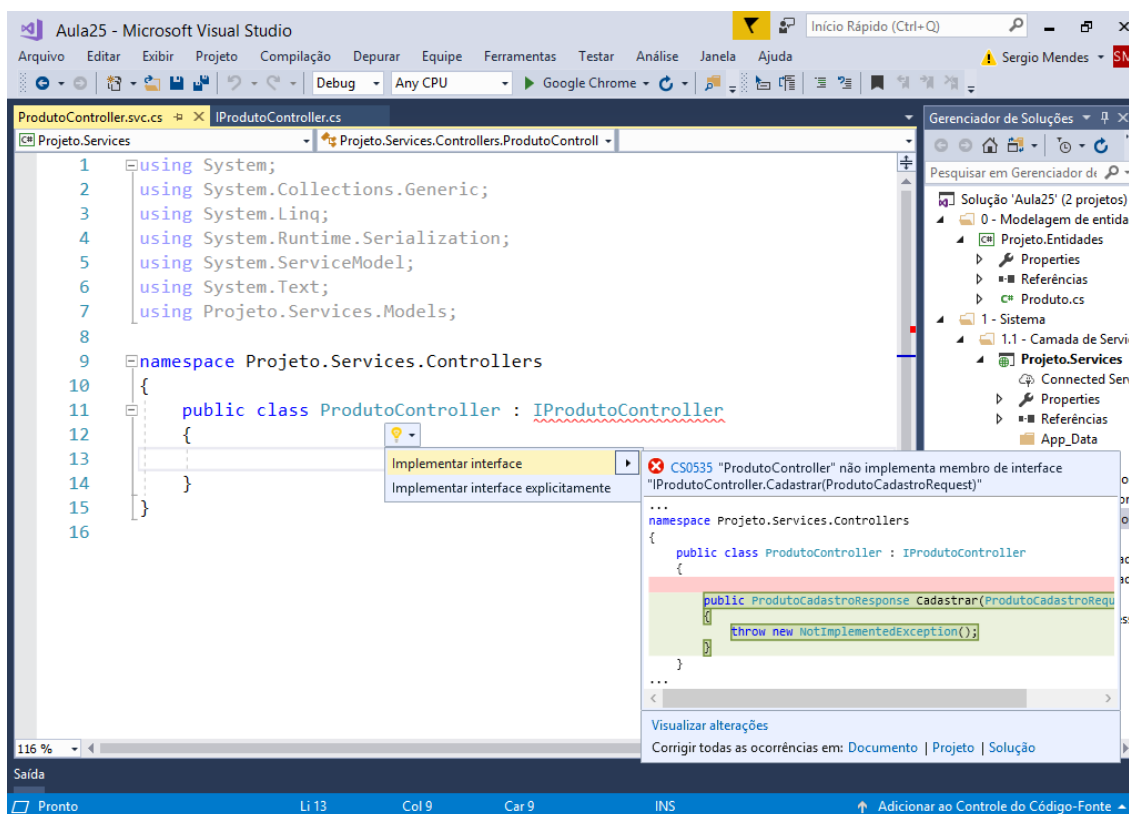
Todo serviço WCF é criado em 2 partes:

- Primeiro é criado uma interface que irá conter os métodos (funções) que o serviço irá disponibilizar.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using Projeto.Services.Models; //camada de modelo..

namespace Projeto.Services.Controllers
{
    [ServiceContract]
    public interface IProdutoController
    {
        [OperationContract]
        ProdutoCadastroResponse Cadastrar(ProdutoCadastroRequest request);
    }
}
```

- Segundo é implementar os métodos da interface



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using Projeto.Services.Models;

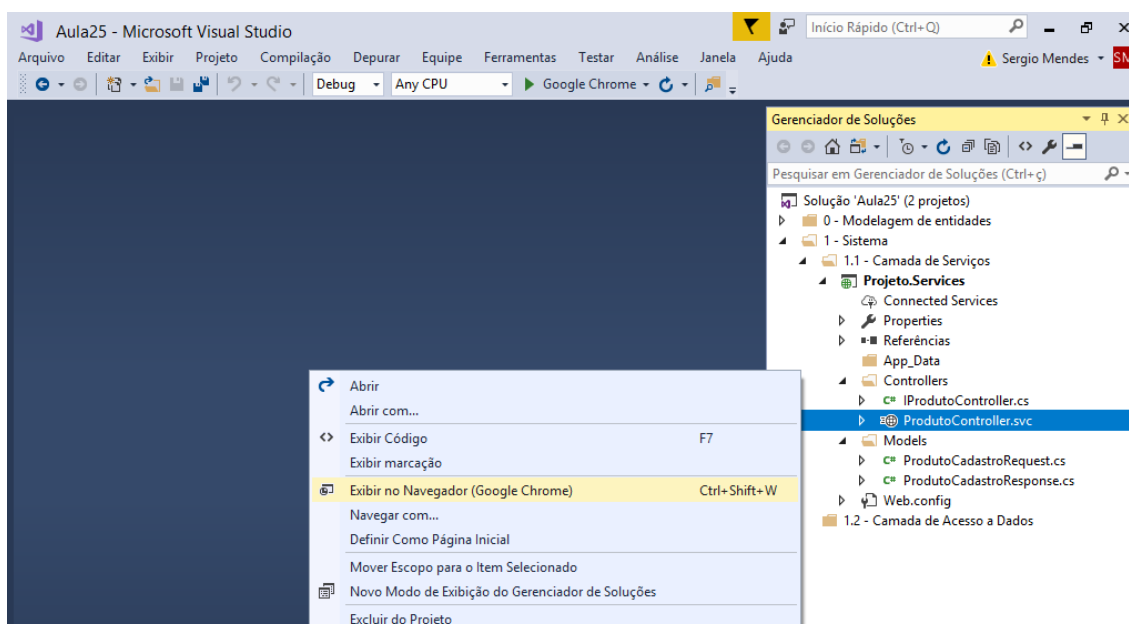
namespace Projeto.Services.Controllers
{
    public class ProdutoController : IProdutoController
    {
        public ProdutoCadastroResponse Cadastrar(ProdutoCadastroRequest request)
        {
            ProdutoCadastroResponse response = new ProdutoCadastroResponse();

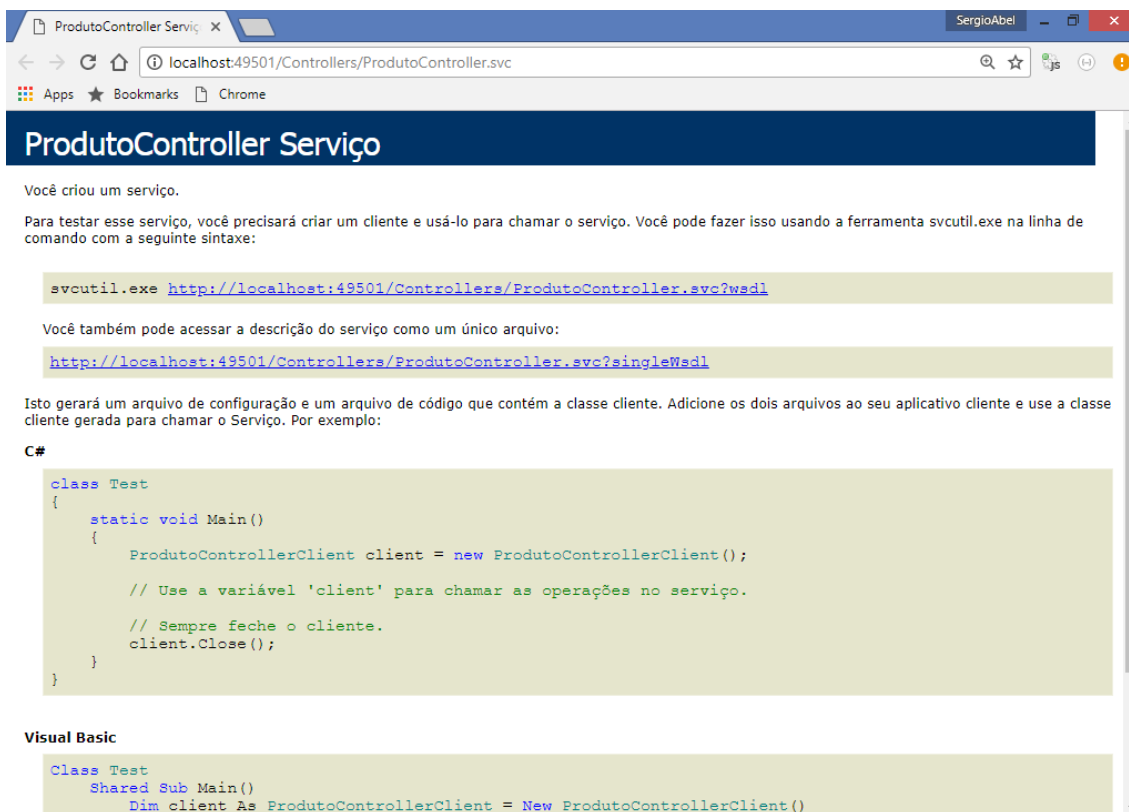
            try
            {
                //TODO..

                //sucesso..
                response.Status = "Sucesso";
                response.Mensagem = $"Produto {request.Nome},
                                    cadastrado com sucesso.";
            }
            catch (Exception e)
            {
                //erro..
                response.Status = "Erro";
                response.Mensagem = $"Ocorreu um erro: {e.Message}";
            }

            return response;
        }
    }
}
```

Executando o serviço:





ProdutoController Serviço

Você criou um serviço.

Para testar esse serviço, você precisará criar um cliente e usá-lo para chamar o serviço. Você pode fazer isso usando a ferramenta svcutil.exe na linha de comando com a seguinte sintaxe:

```
svcutil.exe http://localhost:49501/Controllers/ProdutoController.svc?wsdl
```

Você também pode acessar a descrição do serviço como um único arquivo:

```
http://localhost:49501/Controllers/ProdutoController.svc?singleWsdl
```

Isto gerará um arquivo de configuração e um arquivo de código que contém a classe cliente. Adicione os dois arquivos ao seu aplicativo cliente e use a classe cliente gerada para chamar o Serviço. Por exemplo:

C#

```
class Test
{
    static void Main()
    {
        ProdutoControllerClient client = new ProdutoControllerClient();

        // Use a variável 'client' para chamar as operações no serviço.

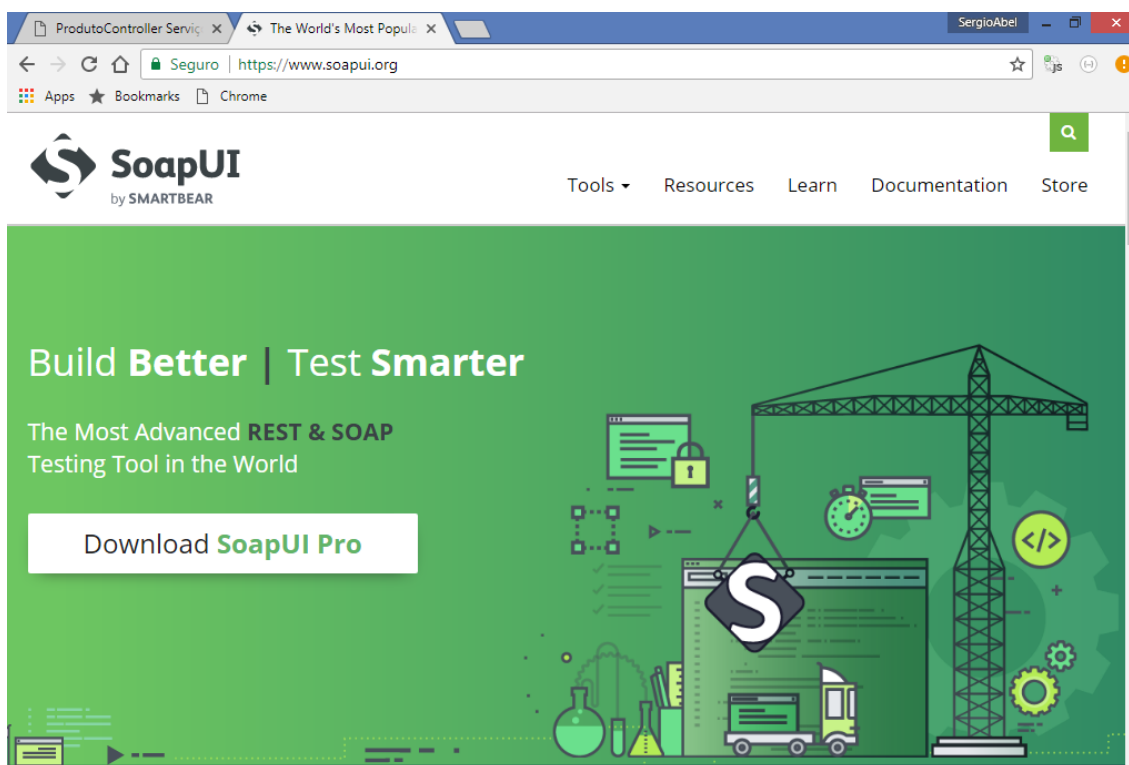
        // Sempre feche o cliente.
        client.Close();
    }
}
```

Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As ProdutoControllerClient = New ProdutoControllerClient()
```

SOAP UI (<https://www.soapui.org/>)


Ferramenta para teste de webservices.



ProdutoController Serviço x The World's Most Popular x SergioAbel

← → ↻ ⌂ Seguro | <https://www.soapui.org> ☆ ⌵ ⌵ ⌵

Apps ★ Bookmarks □ Chrome

 **SoapUI**
by SMARTBEAR

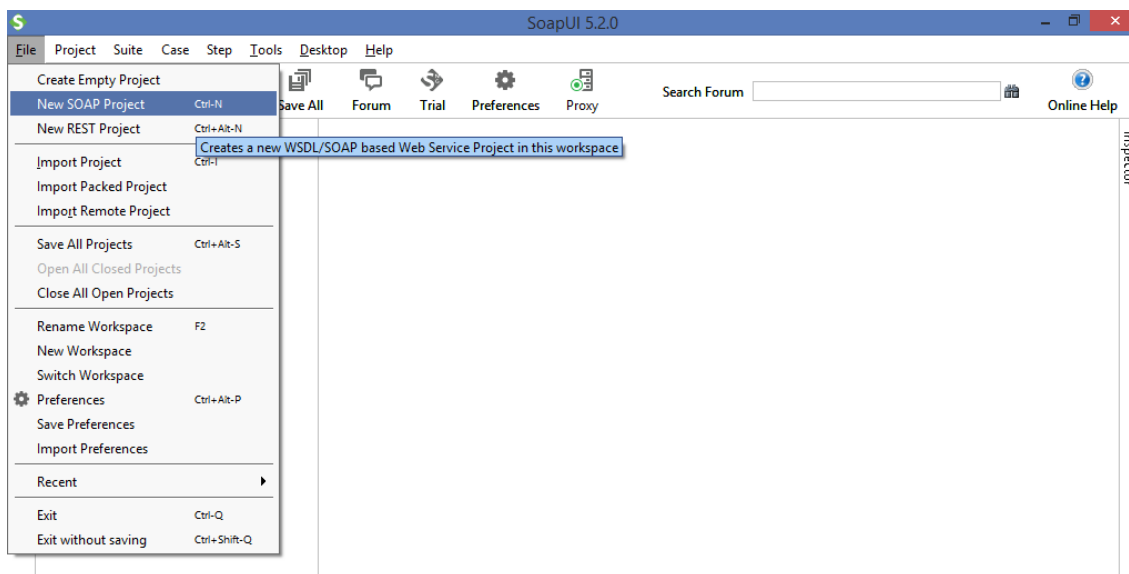
Tools ▾ Resources Learn Documentation Store

Build Better | Test Smarter

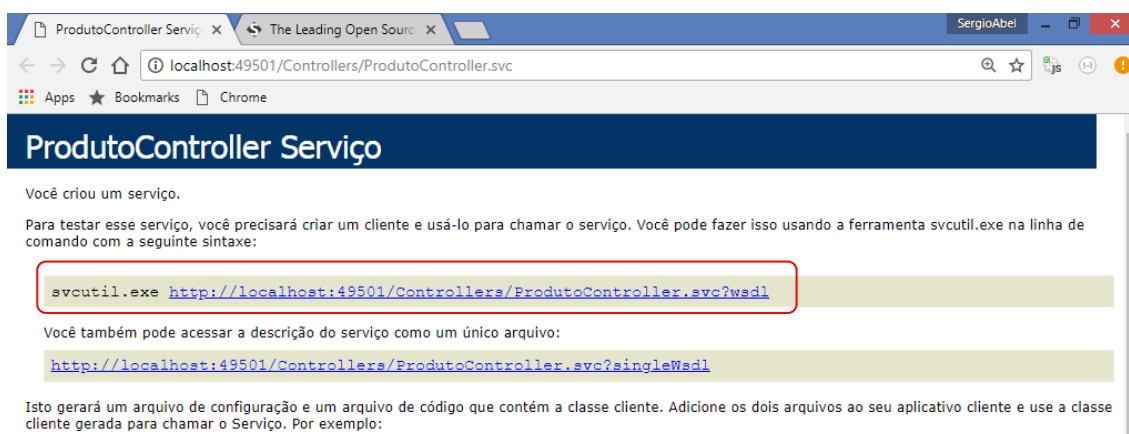
The Most Advanced **REST & SOAP** Testing Tool in the World

Download **SoapUI Pro**

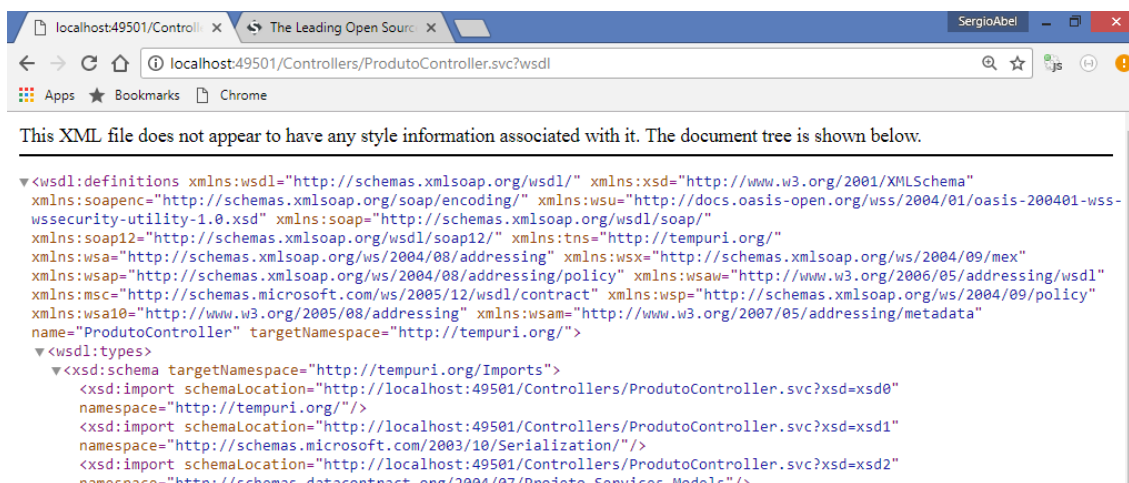
Para que possamos testar um webservice é necessário obter o endereço **WSDL (WebService Description Language)** do webservice.



Obtenha o endereço WSDL do serviço:

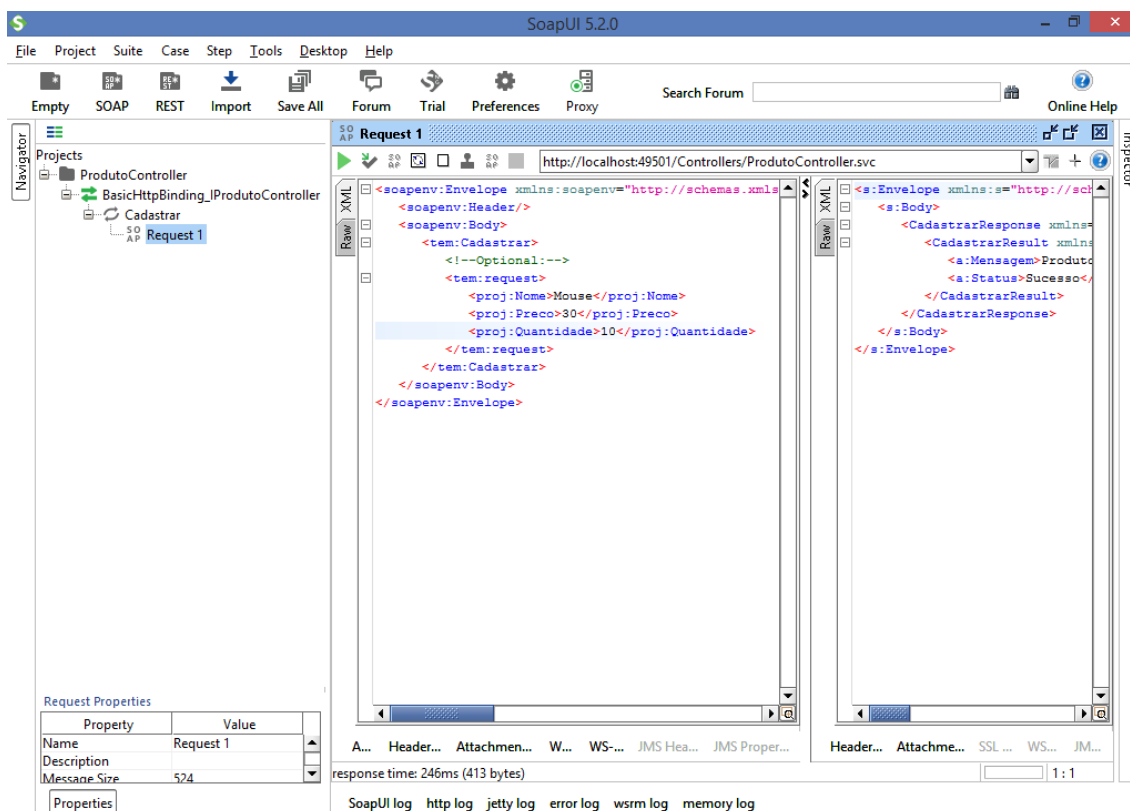
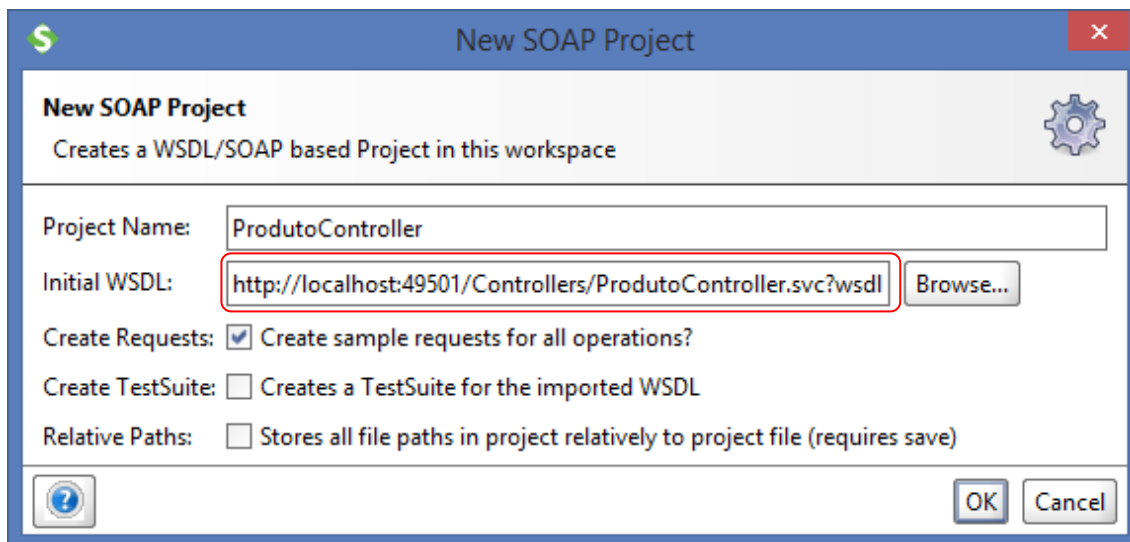


<http://localhost:49501/Controllers/ProdutoController.svc?wsdl>



Cole o endereço e crie um novo projeto no SOAPUI:

<http://localhost:49501/Controllers/ProdutoController.svc?wsdl>



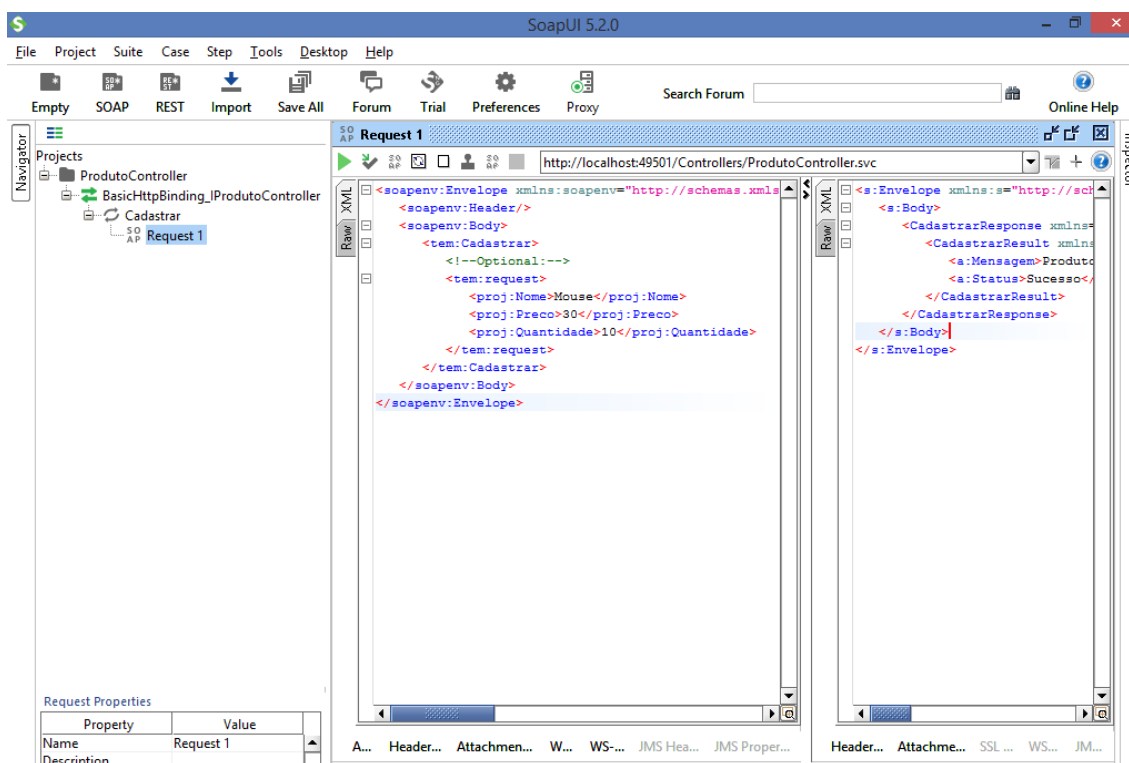
XML de Request:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tem="http://tempuri.org/"
xmlns:proj="http://schemas.datacontract.org/2004/07/Projeto.Services.Models">
```

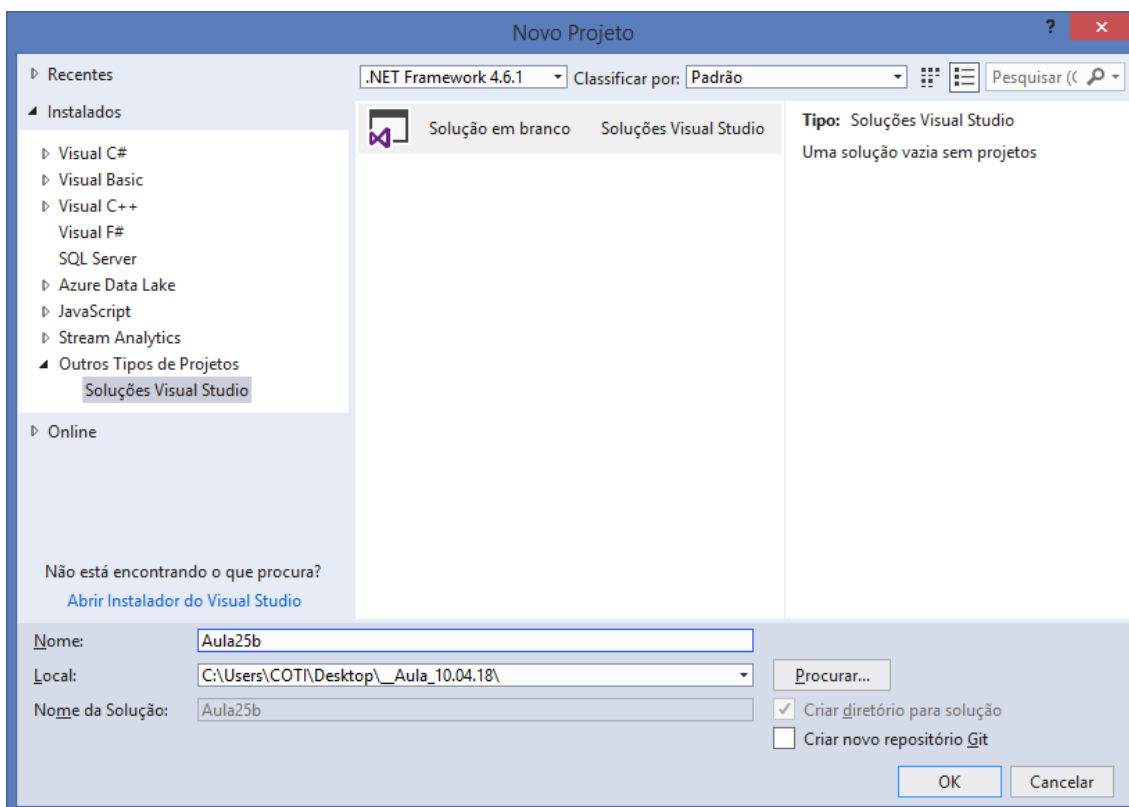
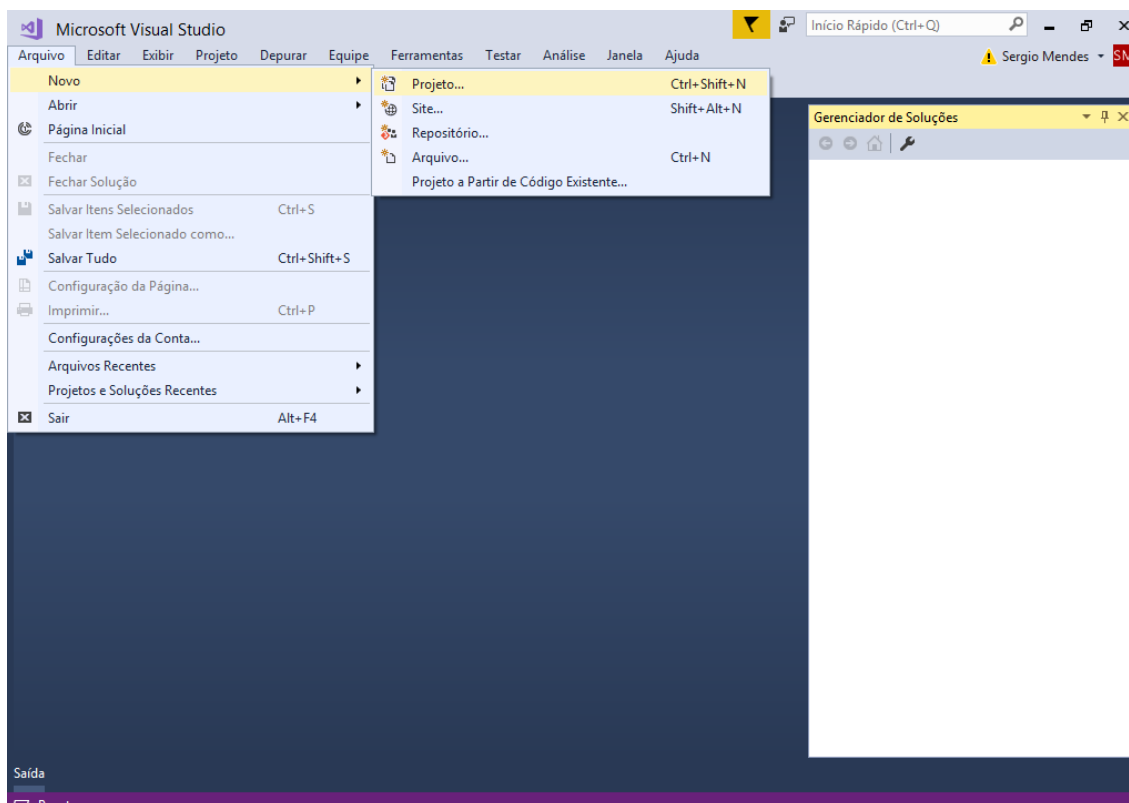
```
<soapenv:Header/>
<soapenv:Body>
  <tem:Cadastrar>
    <!--Optional:-->
    <tem:request>
      <proj:Nome>Mouse</proj:Nome>
      <proj:Preco>30</proj:Preco>
      <proj:Quantidade>10</proj:Quantidade>
    </tem:request>
  </tem:Cadastrar>
</soapenv:Body>
</soapenv:Envelope>
```

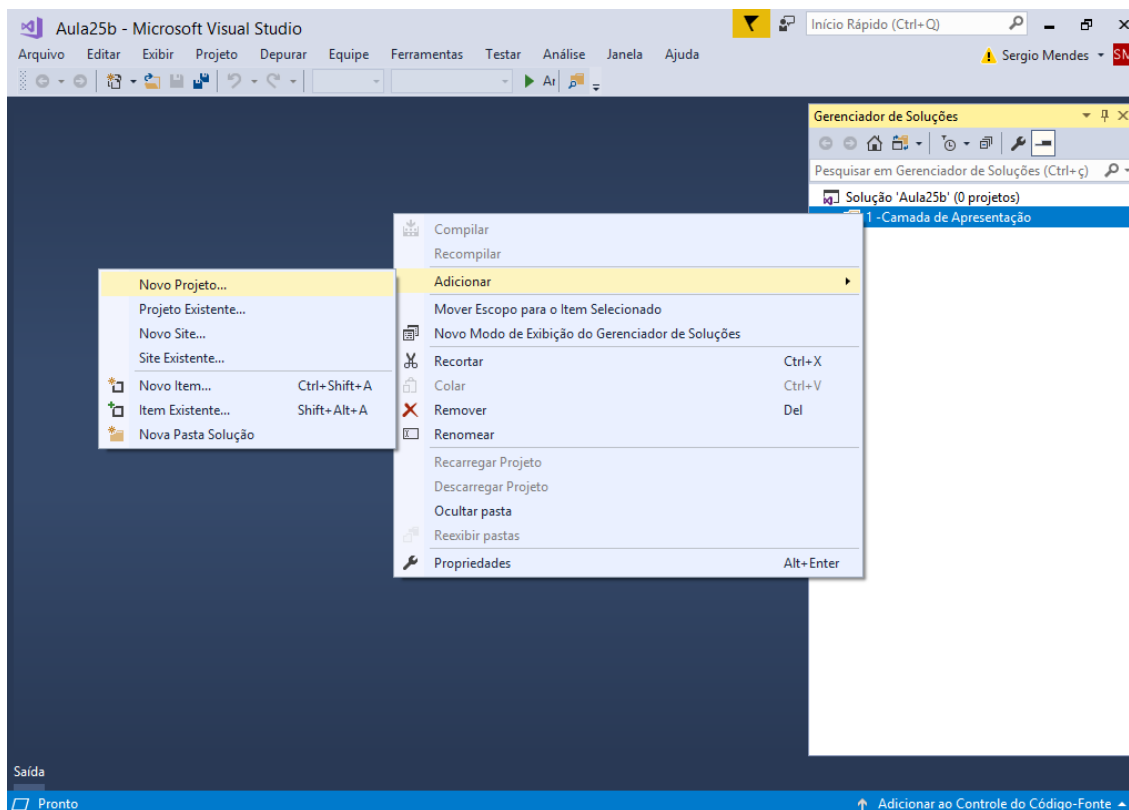
XML de Response:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <CadastrarResponse xmlns="http://tempuri.org/">
      <CadastrarResult xmlns:a="http://schemas.datacontract.org/
        2004/07/Projeto.Services.Models"
        xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:Mensagem>Produto Mouse, cadastrado
          com sucesso.</a:Mensagem>
        <a>Status>Sucesso</a>Status>
      </CadastrarResult>
    </CadastrarResponse>
  </s:Body>
</s:Envelope>
```

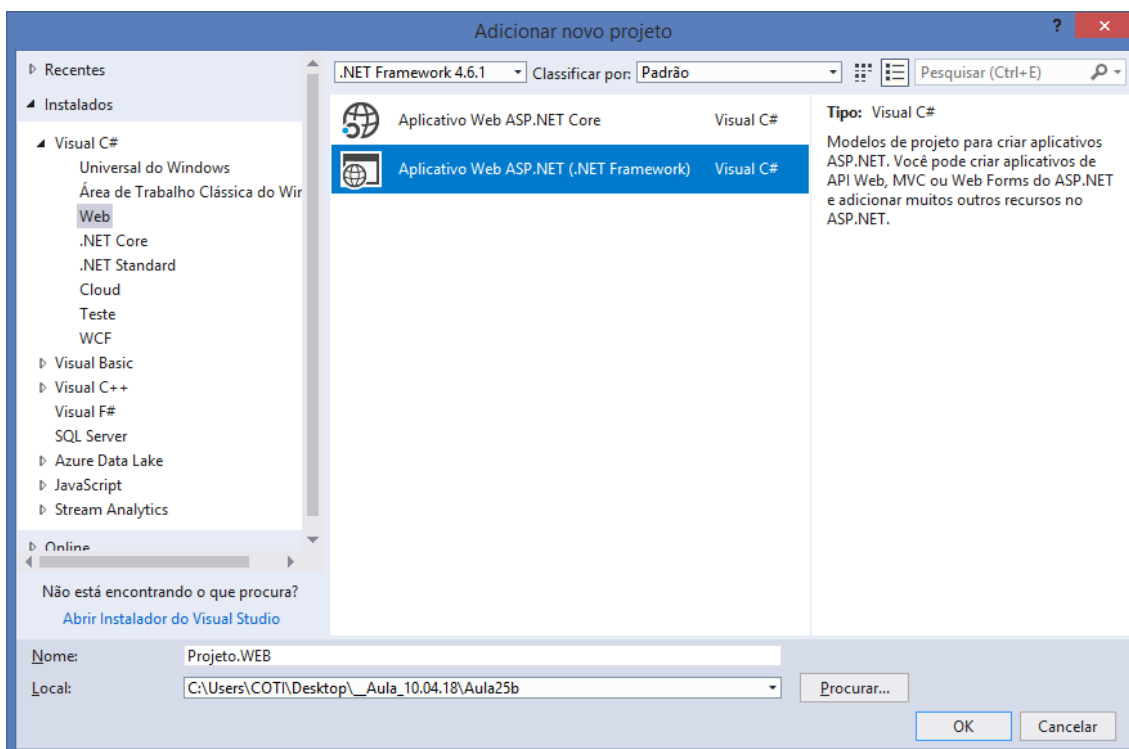


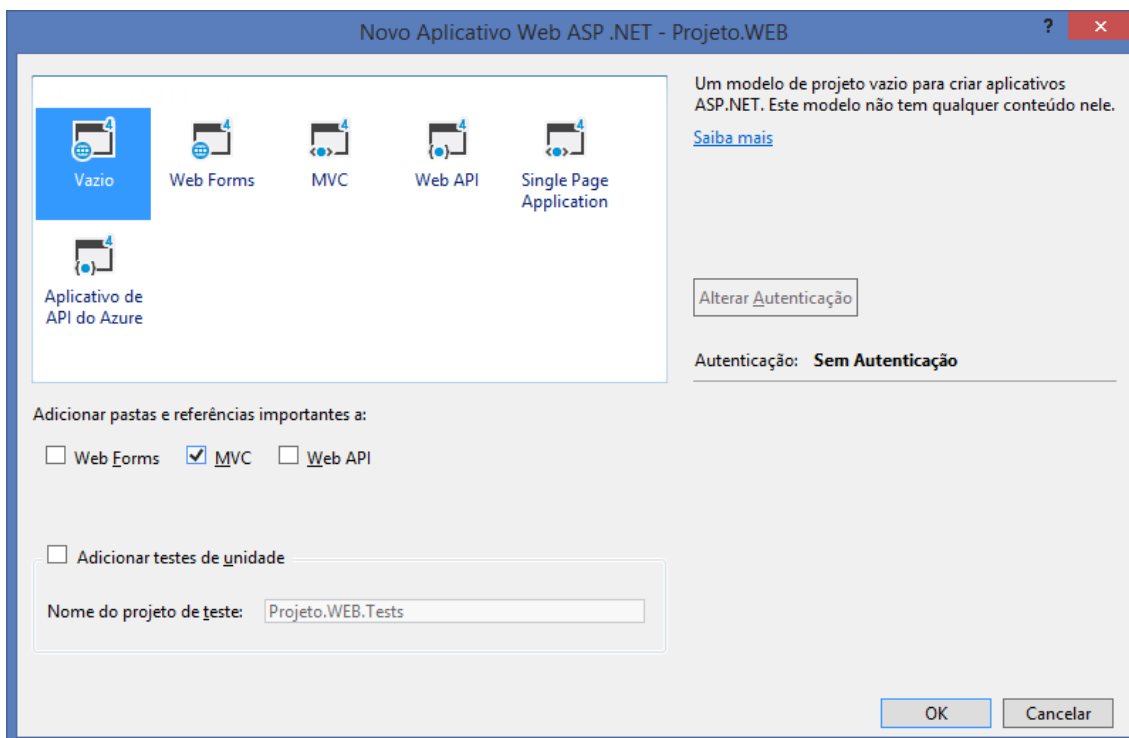
Criando o Cliente do Webservice:





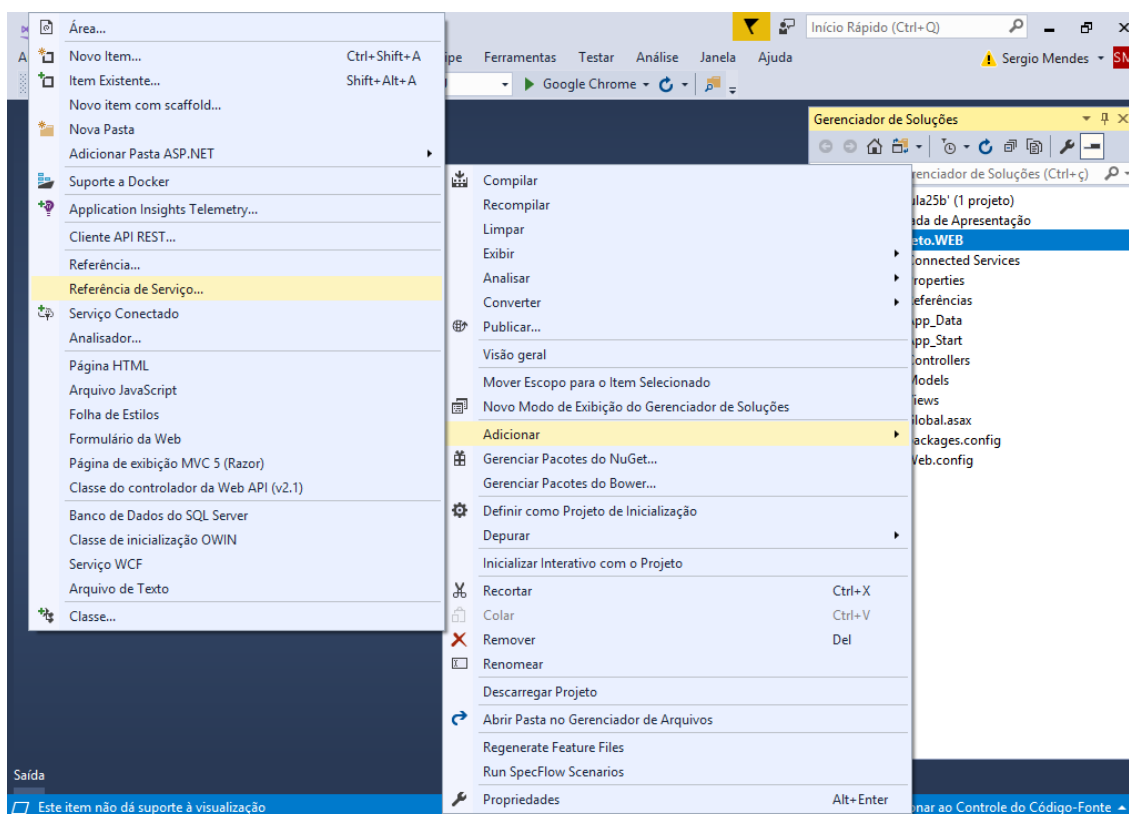
Criando um projeto Asp.Net MVC



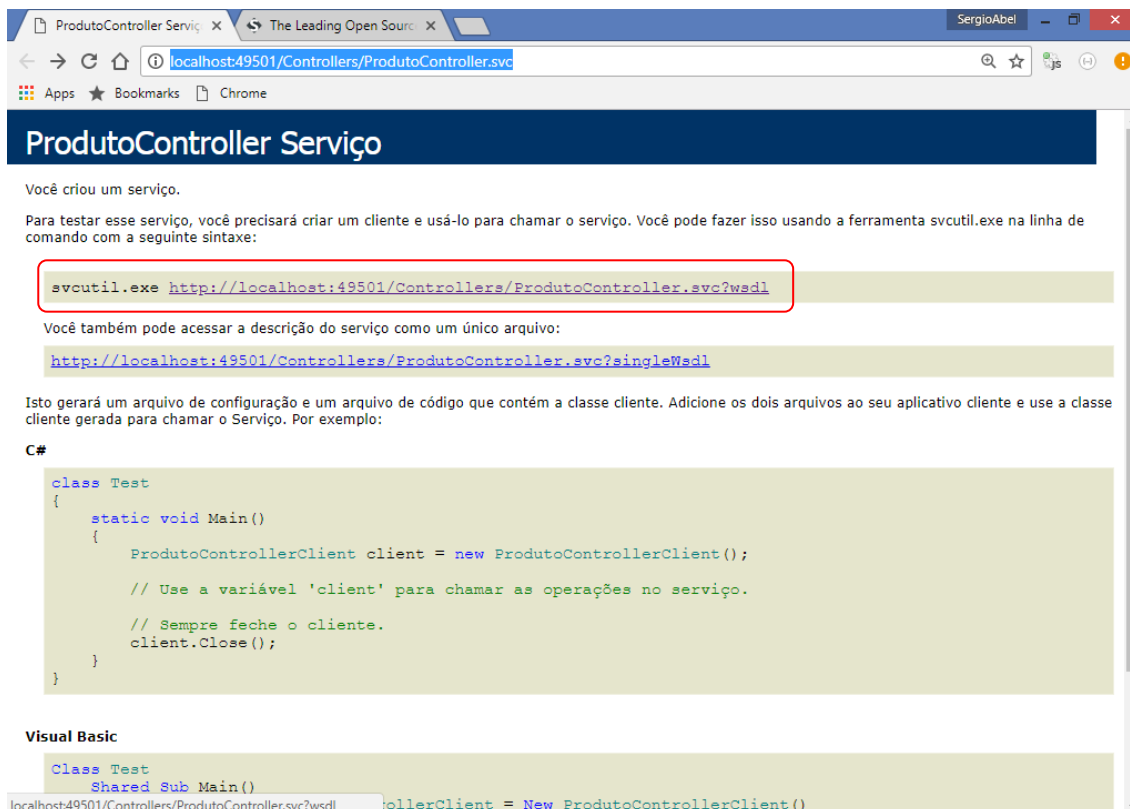


Adicionando referencia para o Webservice:

- Adicionar
- Referencia de Serviço



Cole o endereço WSDL do serviço:



Você criou um serviço.

Para testar esse serviço, você precisará criar um cliente e usá-lo para chamar o serviço. Você pode fazer isso usando a ferramenta svcutil.exe na linha de comando com a seguinte sintaxe:

```
svcutil.exe http://localhost:49501/Controllers/ProdutoController.svc?wsdl
```

Você também pode acessar a descrição do serviço como um único arquivo:

```
http://localhost:49501/Controllers/ProdutoController.svc?singleWsdl
```

Isto gerará um arquivo de configuração e um arquivo de código que contém a classe cliente. Adicione os dois arquivos ao seu aplicativo cliente e use a classe cliente gerada para chamar o Serviço. Por exemplo:

C#

```
class Test
{
    static void Main()
    {
        ProdutoControllerClient client = new ProdutoControllerClient();

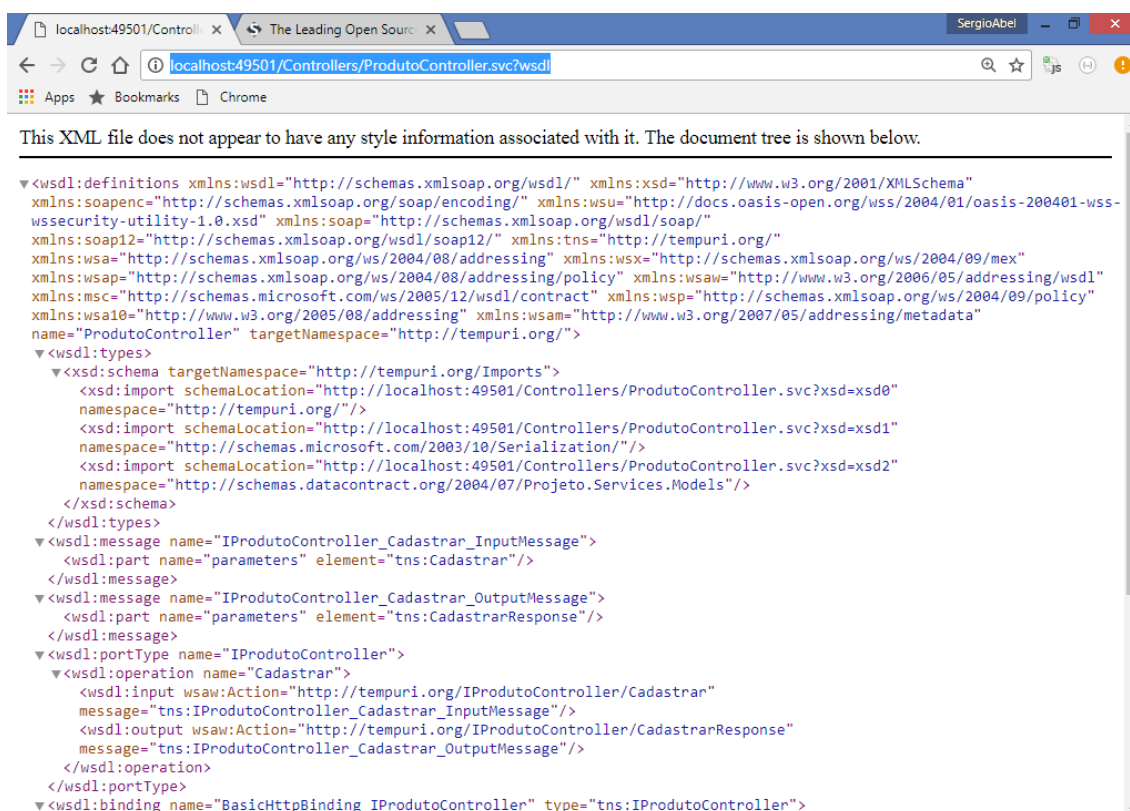
        // Use a variável 'client' para chamar as operações no serviço.

        // Sempre feche o cliente.
        client.Close();
    }
}
```

Visual Basic

```
Class Test
    Shared Sub Main()
        controllerClient = New ProdutoControllerClient()
```

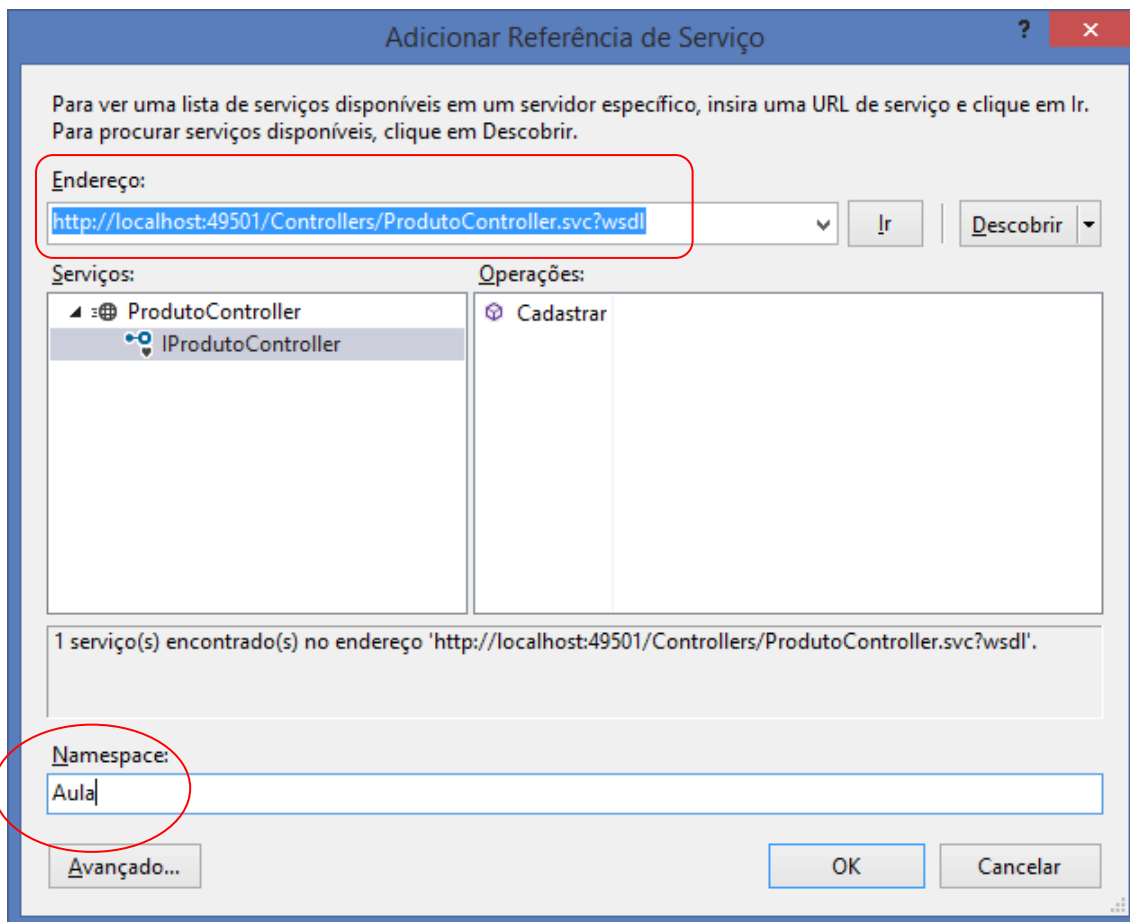
<http://localhost:49501/Controllers/ProdutoController.svc?wsdl>



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version='1.0' encoding='utf-8'?>
<wsdl:definitions xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/' xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/' xmlns:wsu='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/' xmlns:soap12='http://schemas.xmlsoap.org/wsdl/soap12/' xmlns:tns='http://tempuri.org/'
xmlns:wsa='http://schemas.xmlsoap.org/ws/2004/08/addressing' xmlns:wsx='http://schemas.xmlsoap.org/ws/2004/09/mex'
xmlns:wsap='http://schemas.xmlsoap.org/ws/2004/08/addressing/policy' xmlns:wsaw='http://www.w3.org/2006/05/addressing/wsdl'
xmlns:msc='http://schemas.microsoft.com/ws/2005/12/wsdl/contract' xmlns:wsp='http://schemas.xmlsoap.org/ws/2004/09/policy'
xmlns:wsai0='http://www.w3.org/2005/08/addressing' xmlns:wsam='http://www.w3.org/2007/05/addressing/metadata'
name='ProdutoController' targetNamespace='http://tempuri.org/'>
  <wsdl:types>
    <xsd:schema targetNamespace='http://tempuri.org/Imports'>
      <xsd:import schemaLocation='http://localhost:49501/Controllers/ProdutoController.svc?xsd=xsd0'
namespace='http://tempuri.org/'/>
      <xsd:import schemaLocation='http://localhost:49501/Controllers/ProdutoController.svc?xsd=xsd1'
namespace='http://schemas.microsoft.com/2003/10/Serialization/'/>
      <xsd:import schemaLocation='http://localhost:49501/Controllers/ProdutoController.svc?xsd=xsd2'
namespace='http://schemas.datacontract.org/2004/07/Projeto.Services.Models/'/>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name='IProdutoController_Cadastrar_InputMessage'>
    <wsdl:part name='parameters' element='tns:Cadastrar'/>
  </wsdl:message>
  <wsdl:message name='IProdutoController_Cadastrar_OutputMessage'>
    <wsdl:part name='parameters' element='tns:CadastrarResponse'/>
  </wsdl:message>
  <wsdl:portType name='IProdutoController'>
    <wsdl:operation name='Cadastrar'>
      <wsdl:input wsaw:Action='http://tempuri.org/IProdutoController/Cadastrar'
message='tns:IProdutoController_Cadastrar_InputMessage'/>
      <wsdl:output wsaw:Action='http://tempuri.org/IProdutoController/CadastrarResponse'
message='tns:IProdutoController_Cadastrar_OutputMessage'/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name='BasicHttpBinding_IProdutoController' type='tns:IProdutoController'>
  </wsdl:binding>
</wsdl:definitions>
```

Coloque o endereço do WSDL do serviço
e também adicione um nome de namespace:



Adicionar Referência de Serviço

Para ver uma lista de serviços disponíveis em um servidor específico, insira uma URL de serviço e clique em Ir.
Para procurar serviços disponíveis, clique em Descobrir.

Endereço:

Serviços:

- ProdutoController
- IProdutoController

Operações:

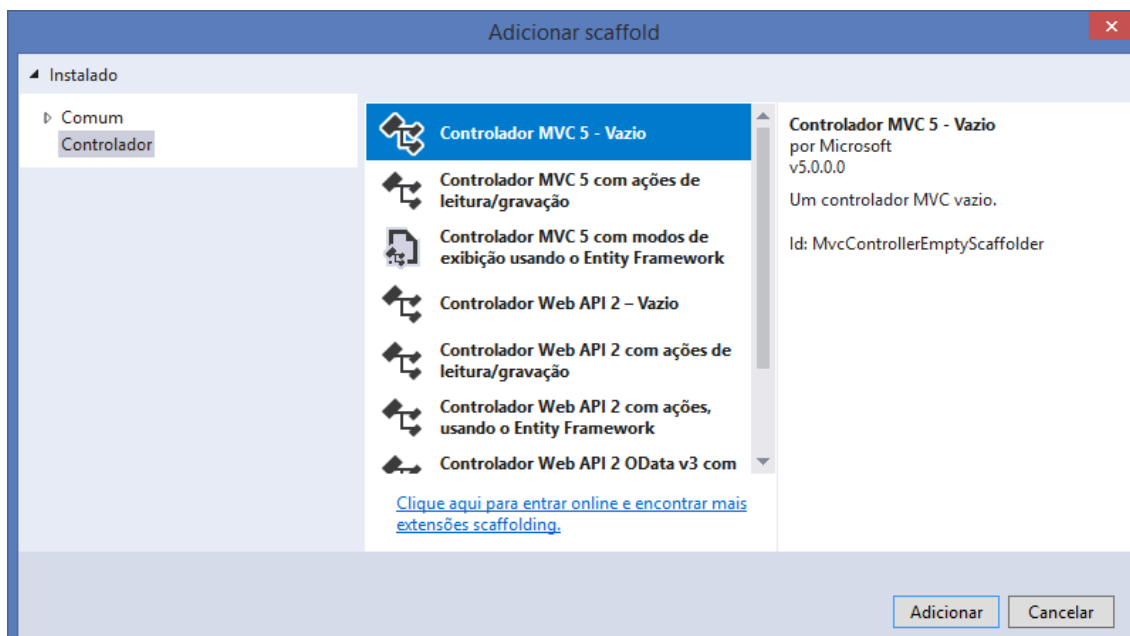
- Cadastrar

1 serviço(s) encontrado(s) no endereço 'http://localhost:49501/Controllers/ProdutoController.svc?wsdl'.

Namespace:

Avançado... OK Cancelar

Criando uma classe de controle de produtos:



Adicionar scaffold

Instalado

- Comum
- Controlador

Controlador MVC 5 - Vazio

Controlador MVC 5 com ações de leitura/gravação

Controlador MVC 5 com modos de exibição usando o Entity Framework

Controlador Web API 2 - Vazio

Controlador Web API 2 com ações de leitura/gravação

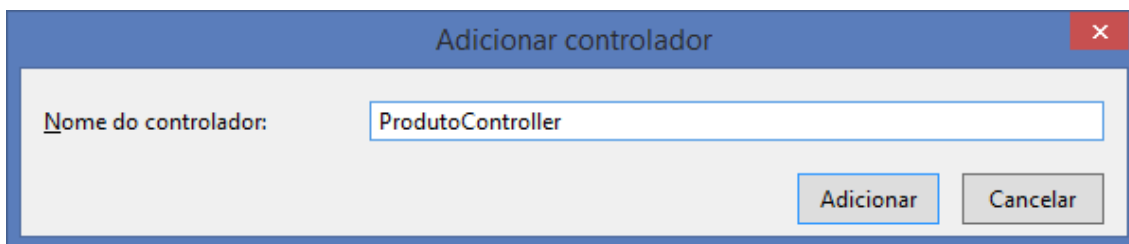
Controlador Web API 2 com ações, usando o Entity Framework

Controlador Web API 2 OData v3 com

[Clique aqui para entrar online e encontrar mais extensões scaffolding.](#)

Controlador MVC 5 - Vazio
por Microsoft
v5.0.0.0
Um controlador MVC vazio.
Id: MvcControllerEmptyScaffolder

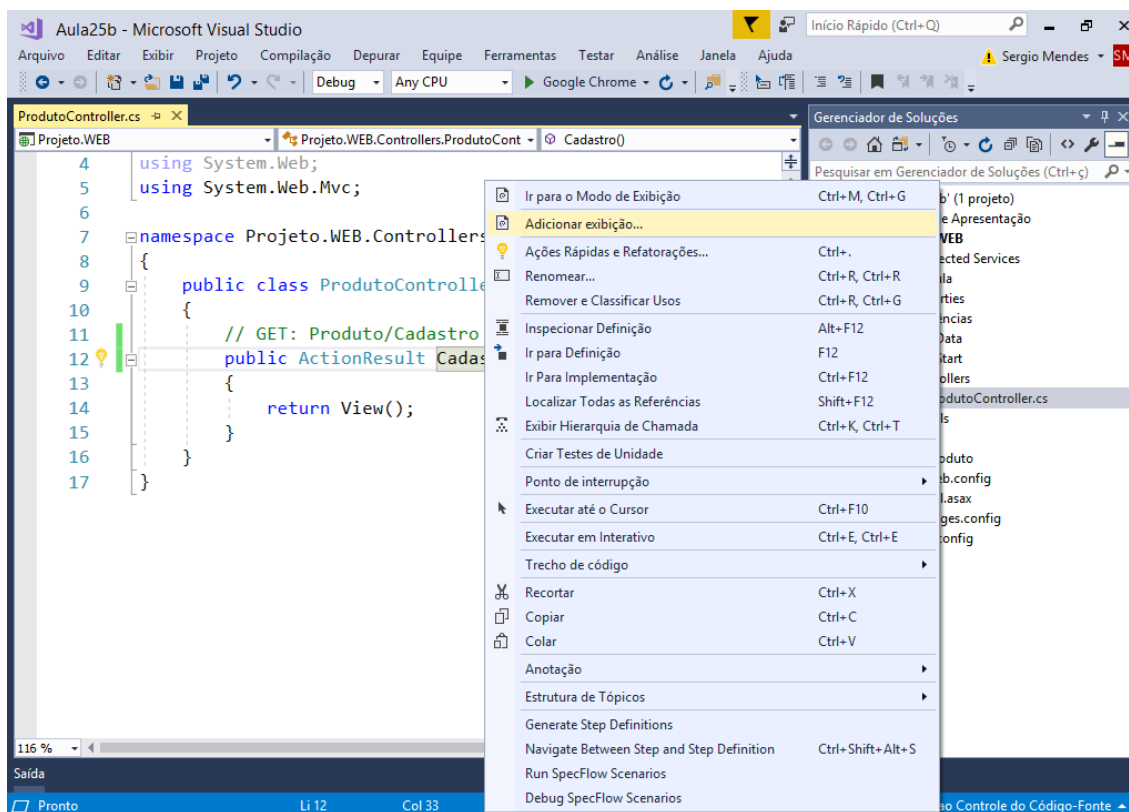
Adicionar Cancelar



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Projeto.WEB.Controllers
{
    public class ProdutoController : Controller
    {
        // GET: Produto/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }
    }
}
```

Adicionando a página:



Adicionar modo de exibição

Nome do modo de exibição:

Modelo:

Classe do modelo:

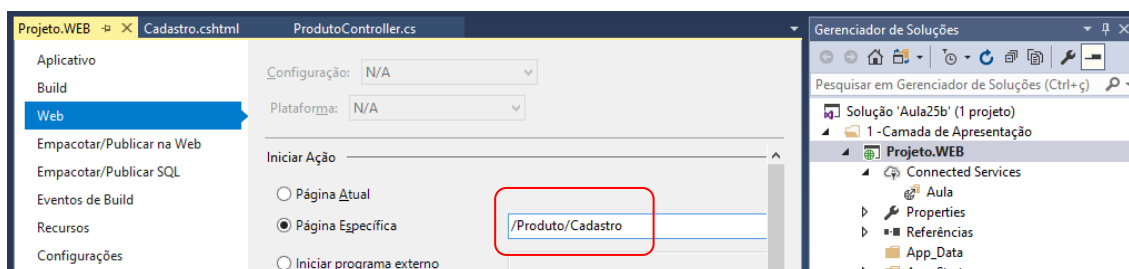
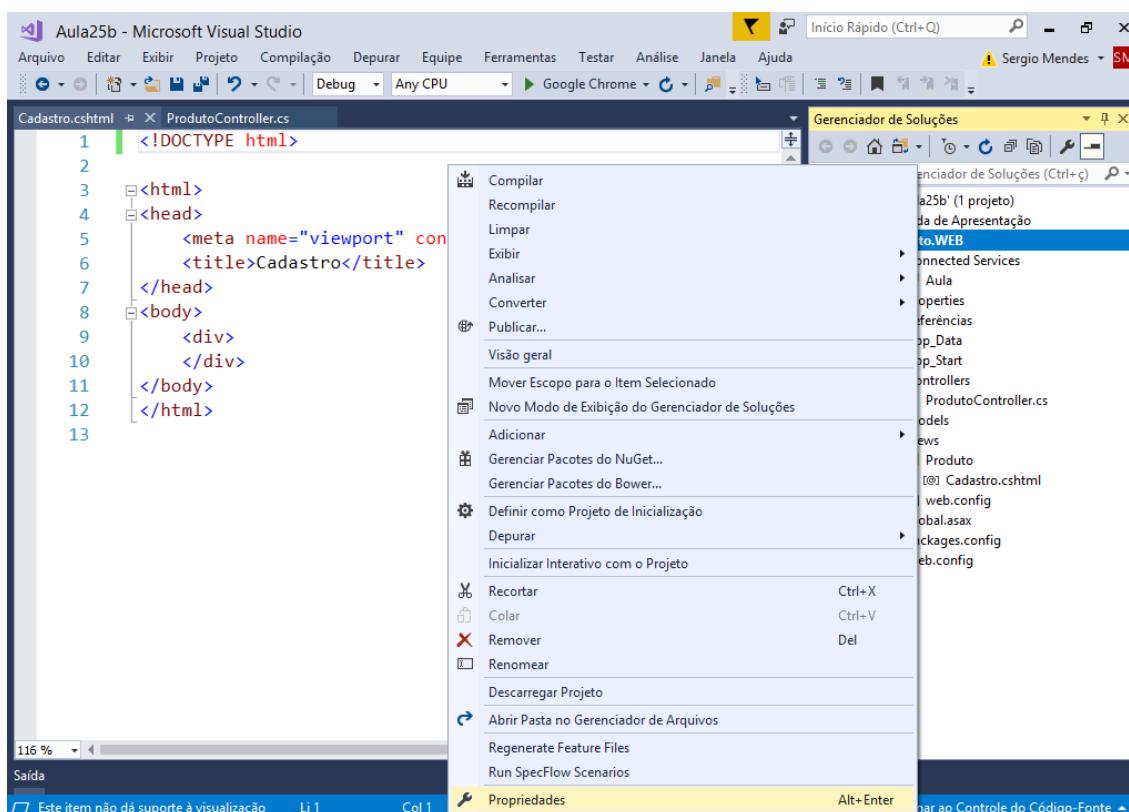
Opções:

☐ Criar como um Modo de exibição parcial

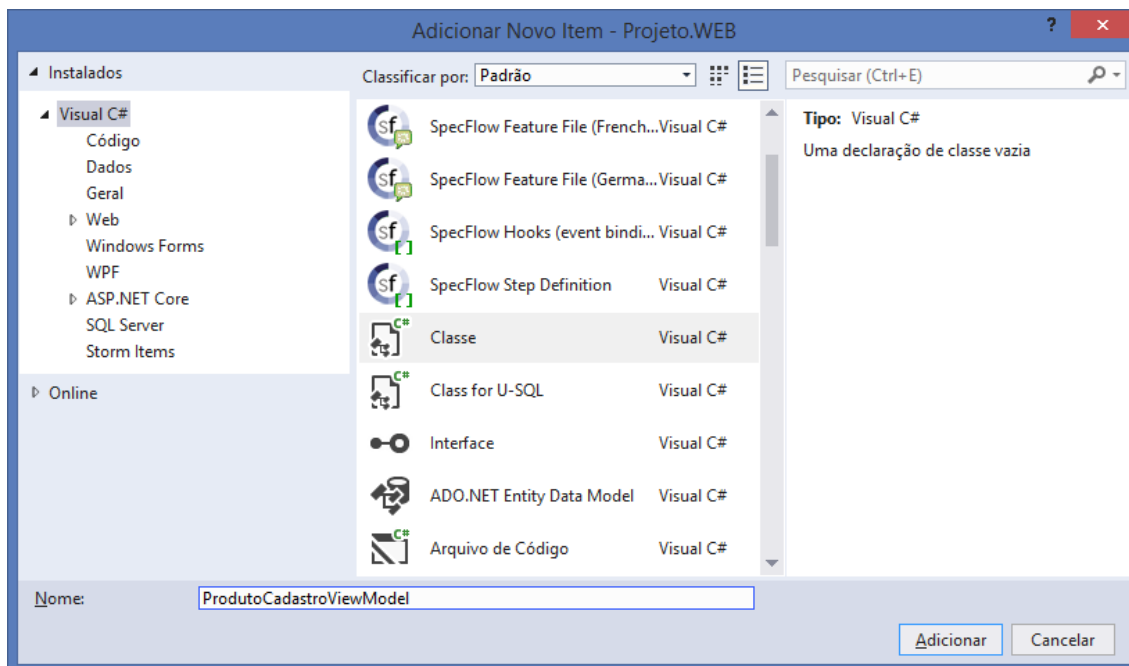
☒ Bibliotecas de scripts de referência

☐ Usar uma página de layout:

(deixe em branco se ele estiver definido em um arquivo Razor _viewstart)



Classe de modelo para criar um formulário de cadastro de produto:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Projeto.WEB.Models
{
    public class ProdutoCadastroViewModel
    {
        [Required(ErrorMessage = "Por favor, informe o nome do produto.")]
        public string Nome { get; set; }

        [Required(ErrorMessage = "Por favor, informe o preço do produto.")]
        public decimal Preco { get; set; }

        [Required(ErrorMessage = "Por favor, informe a quantidade do produto.")]
        public int Quantidade { get; set; }
    }
}
```

Criando o formulário para cadastro de produtos:

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Cadastro</title>
</head>
<body>
    <div>
        <h2>Formulário de Cadastro de Produto</h2>
        <hr/>
    </div>
</body>
</html>
```

```
@model Projeto.WEB.Models.ProdutoCadastroViewModel

@using (Html.BeginForm())
{
    <label>Nome do Produto:</label> <br/>
    @Html.TextBoxFor(model => model.Nome)
    <br/><br />

    <label>Preço:</label> <br/>
    @Html.TextBoxFor(model => model.Preco)
    <br/><br />

    <label>Quantidade:</label> <br/>
    @Html.TextBoxFor(model => model.Quantidade)
    <br /><br />

    <input type="submit" value="Cadastrar Produto"/>
    <br /><br />

    @ViewBag.Mensagem
}

</div>
</body>
</html>
```

Voltando ao controller:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Projeto.WEB.Models; //camada de modelo..
using Projeto.WEB.Aula; //webservice..

namespace Projeto.WEB.Controllers
{
    public class ProdutoController : Controller
    {
        // GET: Produto/Cadastro
        public ActionResult Cadastro()
        {
            return View();
        }

        // POST: Produto/Cadastro
        [HttpPost]
        public ActionResult Cadastro(ProdutoCadastroViewModel model)
        {
            try
            {
                ProdutoCadastroRequest request = new ProdutoCadastroRequest();
                request.Nome = model.Nome;
                request.Preco = model.Preco;
                request.Quantidade = model.Quantidade;

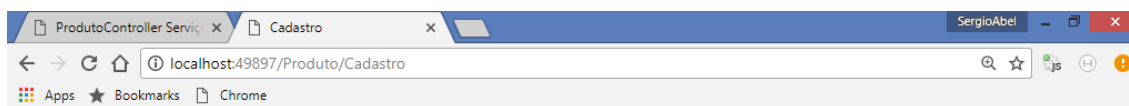
                ProdutoControllerClient client = new ProdutoControllerClient();
                ProdutoCadastroResponse response = client.Cadastrar(request);
            }
            catch { }
        }
    }
}
```

```
        ViewBag.Mensagem = response.Mensagem;
        ModelState.Clear();

        client.Close();
    }
    catch (Exception e)
    {
        ViewBag.Mensagem = e.Message;
    }

    return View();
}
}
```

Executando:



Formulário de Cadastro de Produto

Nome do Produto:

Preço:

Quantidade:

Cadastrar Produto

Produto Mouse, cadastrado com sucesso.
