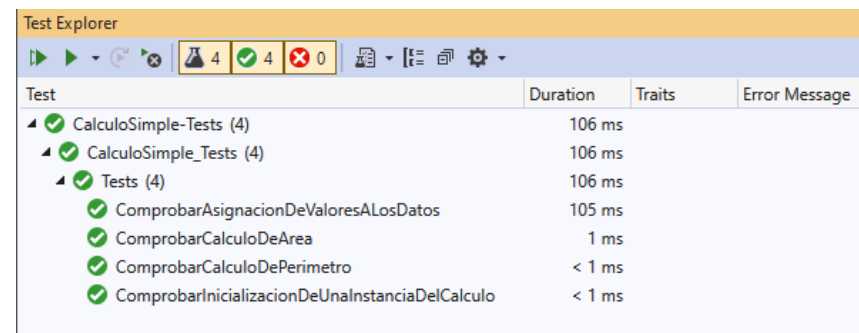
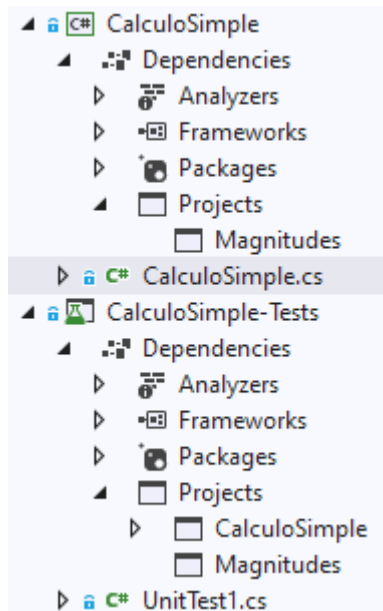


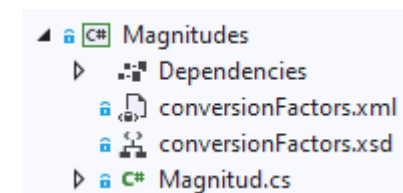
pruebas y experimentos de “multi-interfaz” de usuario

creado 20210331T1216 guardado 20210331T1505 impreso 20210331T1505

El programa base, que se va a utilizar desde diversos interfaces de usuario...



Test	Duration	Traits	Error Message
CalculoSimple-Tests (4)	106 ms		
CalculoSimple_Tests (4)	106 ms		
Tests (4)	106 ms		
ComprobarAsignacionDeValoresALosDatos	105 ms		
ComprobarCalculoDeArea	1 ms		
ComprobarCalculoDePerimetro	< 1 ms		
ComprobarInicializacionDeUnaInstanciaDelCalculo	< 1 ms		



```
using System;
```

```
namespace CalculoSimple
{
    public class CalculoSimple
    {
        private Magnitudes.Magnitud unlado;
        private Magnitudes.Magnitud otrolado;

        private Magnitudes.Magnitud perimetro;
        private Magnitudes.Magnitud area;
```

```
public CalculoSimple()
{
    unlado = new Magnitudes.Magnitud();
    otrolado = new Magnitudes.Magnitud();

    perimetro = new Magnitudes.Magnitud();
    area = new Magnitudes.Magnitud();
}

private void ReCalcular()
{
    if (!double.IsNaN(unlado.valor) && !double.IsNaN(otrolado.valor))
    {
        perimetro.valor = 2.0 * unlado.valor + 2.0 * otrolado.getCopiaConvertidaA(unlado.unidaddemedida).valor;
        perimetro.unidaddemedida = unlado.unidaddemedida;

        area.valor = unlado.valor * otrolado.getCopiaConvertidaA(unlado.unidaddemedida).valor;
        area.unidaddemedida = unlado.unidaddemedida + "2";
    }
}

public void setDato_unlado(Magnitudes.Magnitud dato)
{
    unlado = dato;
    ReCalcular();
}

public void setDato_otrolado(Magnitudes.Magnitud dato)
{
    otrolado = dato;
    ReCalcular();
}

public Magnitudes.Magnitud getResultado_perimetro()
{
    return perimetro;
}

public Magnitudes.Magnitud getResultado_area()
{
    return area;
}
```

```
        public Magnitudes.Magnitud getDato_unlado()
        {
            return unlado;
        }
        public Magnitudes.Magnitud getDato_otrolado()
        {
            return otrolado;
        }
    }
}
```

```
using NUnit.Framework;
```

```
namespace CalculoSimple_Tests
{
```

```
    public class Tests
    {
```

```
        [SetUp]
        public void Setup()
        {
        }
    }
```

```
    [Test]
    public void ComprobarInicializacionDeUnaInstanciaDelCalculo()
    {
        CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

        Assert.IsTrue(double.IsNaN(calculadora.getDato_unlado().valor));
        Assert.AreEqual(Magnitudes.Magnitud.DESCONOCIDA, calculadora.getDato_unlado().unidadmedida);

        Assert.IsTrue(double.IsNaN(calculadora.getDato_otrolado().valor));
        Assert.AreEqual(Magnitudes.Magnitud.DESCONOCIDA, calculadora.getDato_otrolado().unidadmedida);
    }
}
```

```
[Test]
public void ComprobarAsignacionDeValoresALosDatos()
{
    CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

    calculadora.setDato_unlado(new Magnitudes.Magnitud(valor: 2, unidadDeMedida: "m"));
    Assert.AreEqual(2.0, calculadora.getDato_unlado().valor);
    Assert.AreEqual("m", calculadora.getDato_unlado().unidaddemedida);

    calculadora.setDato_otrolado(new Magnitudes.Magnitud(valor: 300, unidadDeMedida: "cm"));
    Assert.AreEqual(300.0, calculadora.getDato_otrolado().valor);
    Assert.AreEqual("cm", calculadora.getDato_otrolado().unidaddemedida);
}

[Test]
public void ComprobarCalculoDePerimetro()
{
    CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

    Assert.IsTrue(double.IsNaN(calculadora.getResultado_perimetro().valor));
    Assert.AreEqual(Magnitudes.Magnitud.DESCONOCIDA, calculadora.getResultado_perimetro().unidaddemedida);

    calculadora.setDato_unlado(new Magnitudes.Magnitud(valor: 2, unidadDeMedida: "m"));
    Assert.IsTrue(double.IsNaN(calculadora.getResultado_perimetro().valor));
    Assert.AreEqual(Magnitudes.Magnitud.DESCONOCIDA, calculadora.getResultado_perimetro().unidaddemedida);

    calculadora.setDato_otrolado(new Magnitudes.Magnitud(valor: 300, unidadDeMedida: "cm"));
    Assert.AreEqual(10, calculadora.getResultado_perimetro().valor);
    Assert.AreEqual("m", calculadora.getResultado_perimetro().unidaddemedida);
}

[Test]
public void ComprobarCalculoDeArea()
{
    CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

    Assert.IsTrue(double.IsNaN(calculadora.getResultado_area().valor));
    Assert.AreEqual(Magnitudes.Magnitud.DESCONOCIDA, calculadora.getResultado_area().unidaddemedida);

    calculadora.setDato_unlado(new Magnitudes.Magnitud(valor: 2, unidadDeMedida: "m"));
    Assert.IsTrue(double.IsNaN(calculadora.getResultado_area().valor));
    Assert.AreEqual(Magnitudes.Magnitud.DESCONOCIDA, calculadora.getResultado_area().unidaddemedida);
}
```

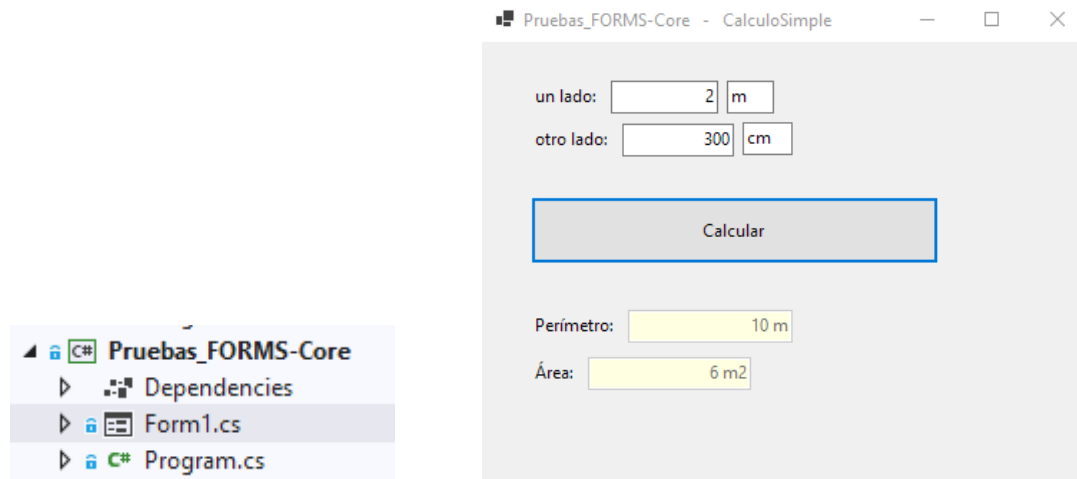
```

        calculadora.setDato_otrolado(new Magnitudes.Magnitud(valor: 300, unidadDeMedida: "cm"));
        Assert.AreEqual(6, calculadora.getResultado_area().valor);
        Assert.AreEqual("m2", calculadora.getResultado_area().unidaddemedida);
    }

}
}

```

Utilizado desde un formulario Windows Forms



```

using System;
using System.Windows.Forms;

namespace Pruebas_FORMS_Core
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

    try
    {
        calculadora.setDato_unlado(new Magnitudes.Magnitud(Double.Parse(txtUnLado_valor.Text),
                                                                txtUnLado_unidadmedida.Text));
        calculadora.setDato_otrolado(new Magnitudes.Magnitud(Double.Parse(txtOtroLado_valor.Text),
                                                                txtOtroLado_unidadmedida.Text));
    }
    catch (ArgumentOutOfRangeException ex)
    {
        MessageBox.Show("Alguna de las unidades de medida no está contemplada en las conversiones: " + ex.Message);
    }
    catch (System.IO.FileNotFoundException ex)
    {
        MessageBox.Show("No se ha encontrado el archivo con la lista de conversiones entre unidades de medida: " + ex.Message);
    }
    catch (ArgumentException ex)
    {
        MessageBox.Show("Problemas leyendo el archivo con la lista de conversiones entre unidades de medida: " + ex.Message);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Problemas en alguna conversión entre unidades de medida: " + ex.Message);
    }

    txtPerimetro.Text = calculadora.getResultado_perimetro().ToString();
    txtArea.Text = calculadora.getResultado_area().ToString();
}
}
```

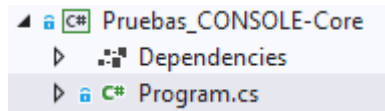
Utilizado desde linea de comandos

```
PS C:\Users\Public\Documents\01_PROGRAMACION\Pruebas_WEB-Core\Pruebas_CONSOLE-Core\bin\Debug\net5.0> dir

Directorio: C:\Users\Public\Documents\01_PROGRAMACION\Pruebas_WEB-Core\Pruebas_CONSOLE-Core\bin\Debug\net5.0

Mode                LastWriteTime         Length Name
----                -
d-----          31/03/2021    10:42             ref
-a----          31/03/2021    10:42           5120 CalculoSimple.dll
-a----          19/10/2020    18:40          27528 Microsoft.Win32.SystemEvents.dll
-a----          31/03/2021    10:42           3655 Pruebas_CONSOLE-Core.deps.json
-a----          31/03/2021    10:42           6144 Pruebas_CONSOLE-Core.dll
-a----          31/03/2021    10:42          142848 Pruebas_CONSOLE-Core.exe
-a----          31/03/2021    10:42           10040 Pruebas_CONSOLE-Core.pdb
-a----          31/03/2021    10:42           238  Pruebas_CONSOLE-Core.runtimeconfig.dev.json
-a----          31/03/2021    10:42           147  Pruebas_CONSOLE-Core.runtimeconfig.json
-a----          16/02/2021     22:00          173456 System.Drawing.Common.dll

PS C:\Users\Public\Documents\01_PROGRAMACION\Pruebas_WEB-Core\Pruebas_CONSOLE-Core\bin\Debug\net5.0> .\Pruebas_CONSOLE-Core.exe 2 m 300 cm
Perimetro=10 m   Area=6 m2
PS C:\Users\Public\Documents\01_PROGRAMACION\Pruebas_WEB-Core\Pruebas_CONSOLE-Core\bin\Debug\net5.0> .\Pruebas_CONSOLE-Core.exe 2 m
Forma de uso:   Pruebas_CONSOLE-Core.exe unLado_valor unLado_unidad otroLado_valor otroLado_unidad
por ejemplo:   Pruebas_CONSOLE-Core.exe 2 m 300 cm
PS C:\Users\Public\Documents\01_PROGRAMACION\Pruebas_WEB-Core\Pruebas_CONSOLE-Core\bin\Debug\net5.0> .\Pruebas_CONSOLE-Core.exe 2 m 300 kg
Alguna de las unidades de medida no está contemplada en las conversiones: Specified argument was out of the range of valid values. (Parameter '300 kg = ? m')
Perimetro=NaN Desconocida-Unknow   Area=NaN Desconocida-Unknow
```



```
using System;

namespace Pruebas_CONSOLE_Core
{
    class Program
    {
        static void Main(string[] args)
        {
            if (args.Length != 4)
            {
                Console.WriteLine("Forma de uso:   Pruebas_CONSOLE-Core.exe unLado_valor unLado_unidad otroLado_valor otroLado_unidad");
                Console.WriteLine("por ejemplo:   Pruebas_CONSOLE-Core.exe 2 m 300 cm");
            }
            else
            {
                CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

                try
                {
                    calculadora.setDato_unlado(new Magnitudes.Magnitud(Double.Parse(args[0]), args[1]));
                    calculadora.setDato_otrolado(new Magnitudes.Magnitud(Double.Parse(args[2]), args[3]));
                }
            }
        }
    }
}
```

```

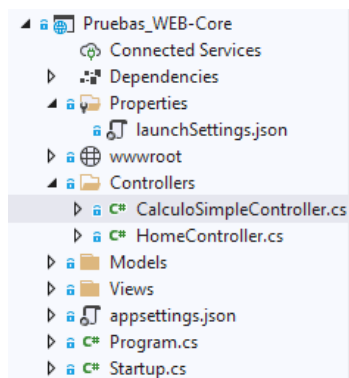
    }
    catch (ArgumentOutOfRangeException ex)
    {
        Console.WriteLine("Alguna de las unidades de medida no está contemplada en las conversiones: " + ex.Message);
    }
    catch (System.IO.FileNotFoundException ex)
    {
        Console.WriteLine("No se ha encontrado el archivo con la lista de conversiones entre unidades de medida: " + ex.Message);
    }
    catch (ArgumentException ex)
    {
        Console.WriteLine("Problemas leyendo el archivo con la lista de conversiones entre unidades de medida: " + ex.Message);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Problemas en alguna conversión entre unidades de medida: " + ex.Message);
    }

    Console.WriteLine("Perimetro=" + calculadora.getResultado_perimetro().ToString()
        + "      Area=" + calculadora.getResultado_area().ToString());
}

}
}

```

Utilizado para dar unos servicios web RESTfull




```
using Microsoft.AspNetCore.Mvc;

namespace Pruebas_WEB_Core.Controllers
{
    public class ParametrosDelCalculoSimple
    {
        public Magnitudes.Magnitud unLado { get; set; }
        public Magnitudes.Magnitud otroLado { get; set; }
    }

    public class ResultadosDelCalculoSimple
    {
        public Magnitudes.Magnitud perimetro { get; set; }
        public Magnitudes.Magnitud area { get; set; }
    }

    [ApiController]
    [Route("api/[controller]")]
    public class CalculoSimpleController : ControllerBase
    {
        // nota: por URL, los parametros se pasan algo así como:
        // https://localhost:44310/api/CalculoSimple?unLado_valor=2&unLado_unidadmedida=m&otroLado_valor=300&otroLado_unidadmedida=cm

        [HttpGet]
        public ResultadosDelCalculoSimple Get(double unLado_valor, string unLado_unidadmedida,
            double otroLado_valor, string otroLado_unidadmedida)
        {
            CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

            calculadora.setDato_unlado(new Magnitudes.Magnitud(unLado_valor, unLado_unidadmedida));
            calculadora.setDato_otrolado(new Magnitudes.Magnitud(otroLado_valor, otroLado_unidadmedida));

            ResultadosDelCalculoSimple resultados = new ResultadosDelCalculoSimple();
            resultados.perimetro = calculadora.getResultado_perimetro();
            resultados.area = calculadora.getResultado_area();
            return resultados;
        }
    }
}
```

```
// nota: en JSON el resultado se devuelve algo así como:
// {
//   "perimetro": {
//     "valor": xxxxx
//     , "unidaddemedida": "xxxxxx"
//   }
//   , "area": {
//     "valor": xxxxx
//     , "unidaddemedida": "xxxxxxx"
//   }
// }
//Escribiendolo en forma compacta: { "perimetro":{ "valor":10,"unidaddemedida":"m"},"area":{ "valor":6,"unidaddemedida":"m2"} }
```

```
// nota: en JSON hay que pasar los parametros algo asi como:
// {
//   "unLado": {
//     "valor": xxxxx
//     , "unidaddemedida":"xxxxxxx"
//   }
//   , "otroLado": {
//     "valor": xxxxx
//     , "unidaddemedida":"xxxxxxx"
//   }
// }
//Escribiendolo en forma compacta: {"unLado":{"valor":2,"unidaddemedida":"m"},"otroLado":{"valor":300,"unidaddemedida":"cm"}}
```

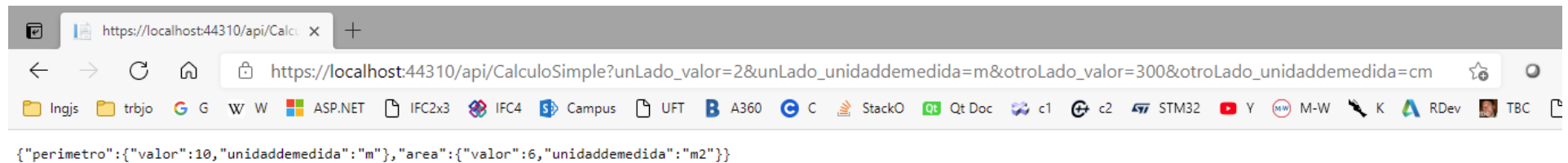
```
[HttpPost]
public ResultadosDelCalculoSimple Post([FromBody] ParametrosDelCalculoSimple parametros)
{
    CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

    calculadora.setDato_unlado(parametros.unLado);
    calculadora.setDato_otrolado(parametros.otroLado);

    ResultadosDelCalculoSimple resultados = new ResultadosDelCalculoSimple();
    resultados.perimetro = calculadora.getResultado_perimetro();
    resultados.area = calculadora.getResultado_area();
    return resultados;
}
```

```
}
}
```

Utilizando el servicio HttpGet a través de una url



Utilizando el servicio HttpPut a través de un programilla de linea de comando

```
C:\Users\Public\Documents\01_PROGRAMACION\Pruebas_WEB-Core\zz-paraProbarServicioWebPost\bin\Debug\net5.0>
zz-paraProbarServicioWebPost.exe 2 m 300 cm
PARAMETROS: {"unLado":{"valor":2,"unidadmedida":"m"},"otroLado":{"valor":300,"unidadmedida":"cm"}}
RESULTADO: {"perimetro":{"valor":10,"unidadmedida":"m"},"area":{"valor":6,"unidadmedida":"m2"}}
Perimetro = 10 m
Area = 6 m2
```

using System;

```
namespace zz_paraProbarServicioWebPost
{
```

```
    public class Magnitud
    {
        public double valor { get; set; }
        public string unidadmedida { get; set; }
    }
    public class ParametrosDelCalculoSimple
    {
        public Magnitud unLado { get; set; }
        public Magnitud otroLado { get; set; }
    }
    public class ResultadosDelCalculoSimple
    {
        public Magnitud perimetro { get; set; }
        public Magnitud area { get; set; }
    }
}
```

```
class Program
{
    static async System.Threading.Tasks.Task Main(string[] args)
    {
        System.Net.Http.HttpClient clienteWeb = new System.Net.Http.HttpClient();

        if (args.Length != 4)
        {
            Console.WriteLine("Forma de uso:  zz-paraProbarServicioWebPost.exe unLado_valor unLado_unidad otroLado_valor otroLado_unidad");
            Console.WriteLine("por ejemplo:  paraProbarServicioWebPost.exe 2 m 300 cm");
        }
        else
        {
            Magnitud unLado = new Magnitud();
            unLado.valor = double.Parse(args[0]);
            unLado.unidadmedida = args[1];
            Magnitud otroLado = new Magnitud();
            otroLado.valor = double.Parse(args[2]);
            otroLado.unidadmedida = args[3];

            ParametrosDelCalculoSimple parametros = new ParametrosDelCalculoSimple();
            parametros.unLado = unLado;
            parametros.otroLado = otroLado;
            string parametrosJson = System.Text.Json.JsonSerializer.Serialize(parametros);

            Console.WriteLine("PARAMETROS: " + parametrosJson);

            try
            {
                System.Net.Http.HttpContent parametrosCodificados =
                    new System.Net.Http.StringContent(content: parametrosJson,
                                                        encoding: System.Text.Encoding.UTF8,
                                                        mediaType: "application/json");

                System.Net.Http.HttpResponseMessage respuesta = await clienteWeb.PostAsync("https://localhost:44310/api/CalculoSimple",
                                                                                          parametrosCodificados);

                respuesta.EnsureSuccessStatusCode();
                string resultadoEnBruto = await respuesta.Content.ReadAsStringAsync();

                Console.WriteLine("RESULTADO: " + resultadoEnBruto);

                ResultadosDelCalculoSimple resultados =
                    (ResultadosDelCalculoSimple)System.Text.Json.JsonSerializer.Deserialize(resultadoEnBruto,
                                                                                             typeof(ResultadosDelCalculoSimple));

                Console.WriteLine("Perimetro = " + resultados.perimetro.valor.ToString() + " " + resultados.perimetro.unidadmedida);
                Console.WriteLine("      Area = " + resultados.area.valor.ToString() + " " + resultados.area.unidadmedida);
            }
            catch { }
        }
    }
}
```

```
}  
catch (Exception ex)  
{  
    Console.WriteLine("ERROR: " + ex.Message);  
}  
}  
}  
}
```

Utilizando el servicio HttpPut desde una página web HTML/JavaScript

The screenshot displays a web browser window titled "CalculoSimple vía HTML" with the address bar showing "https://localhost:44310/Pruebas_HTML.html". The browser's address bar includes navigation buttons (back, forward, refresh, home) and a search bar. Below the address bar, there are several icons representing different web services or applications: Ingjs, trbjo, G, W, W, ASP.NET, IFC2x3, IFC4, and Campus. The main content area of the browser shows a heading "Usando el cálculo desde una página HTML estandard". Below the heading, there is a form titled "Calculo Simple:" with two input fields: "Un lado: 2" and "Otro lado: 300". The first field has a unit "m" and the second field has a unit "cm". Below the input fields, there are two buttons: "Calcular (HttpGet)" and "Calcular (HttpPost) y mostrar resultados". The results of the calculation are displayed below the buttons: "Perímetro: 10 m" and "Área: 6 m2".

Visual Studio is open on the left side of the screen, showing the project structure of "Pruebas_WEB-Core". The project structure includes:

- Connected Services
- Dependencies
- Properties
- launchSettings.json
- wwwroot
 - css
 - js
 - lib
 - favicon.ico
 - Pruebas_HTML.html
- Controllers
- Models
- Views
- appsettings.json
- Program.cs
- Startup.cs

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>CalculoSimple vía HTML</title>
</head>
<body>
    <h2>Usando el cálculo desde una página HTML estandard</h2>

    <div>
        <form name="frmCalculoSimple" action="/api/CalculoSimple" target="_self">
            <fieldset>
                <legend>Calculo Simple:</legend>
                <label for="unLado_valor unLado_unidadmedida">Un lado: </label>
                <input type="number" id="unLado_valor" name="unLado_valor" required value="2" />
                <input type="text" id="unLado_unidadmedida" name="unLado_unidadmedida" size="4" placeholder="cm" required value="m" />
                <br />
                <label for="otroLado_valor otroLado_unidadmedida">Otro lado: </label>
                <input type="number" id="otroLado_valor" name="otroLado_valor" required value="300" />
                <input type="text" id="otroLado_unidadmedida" name="otroLado_unidadmedida" size="4" placeholder="cm" required value="cm"/>
                <br />
                <br />
                <input type="submit" value="Calcular (HttpGet)" formmethod="get" />
                <button type="button" onclick="CalcularYMostrarResultados()">Calcular (HttpPost) y mostrar resultados</button>
            </fieldset>
            <br />
            <fieldset>
                <label for="perimetro">Perimetro: </label>
                <output id="perimetro"></output>
                <br />
                <label for="area">Área: </label>
                <output id="area"></output>
            </fieldset>
        </form>
    </div>

    <script>
        function CalcularYMostrarResultados() {

            var solicitud = new XMLHttpRequest();

            solicitud.onload = function () {
                var respuesta = solicitud.responseText;

                if (solicitud.statusText = "OK") {
                    var resultados = JSON.parse(respuesta)

```

```
        document.getElementById("perimetro").value = resultados.perimetro.valor + " " + resultados.perimetro.unidaddemedida;
        document.getElementById("area").value = resultados.area.valor + " " + resultados.area.unidaddemedida;
    }
}

var esAsincrono = true;
solicitud.open("POST", "/api/CalculoSimple", esAsincrono);

solicitud.setRequestHeader("Content-Type", "application/json;charset=UTF-8");

var parametros = JSON.stringify(
    {
        "unLado": {
            "valor": document.getElementById("unLado_valor").value
            , "unidaddemedida": document.getElementById("unLado_unidaddemedida").value
        }
        , "otroLado": {
            "valor": document.getElementById("otroLado_valor").value
            , "unidaddemedida": document.getElementById("otroLado_unidaddemedida").value
        }
    }
)

solicitud.send(parametros);
}
</script>
</body>
</html>
```

Utilizado desde una página web RAZOR/C#

The screenshot shows a web browser window at <https://localhost:44307> displaying a web application titled "Pruebas_PAGES_Core". The application has a navigation bar with "Home" and "Privacy" links. The main content area is titled "Calculo Simple:" and contains two input fields for "Un lado:" (value 2, unit m) and "Otro lado:" (value 300, unit cm). Below these is a button labeled "Calcular y mostrar resultados". The results section shows "Perímetro: 10 m" and "Área: 6 m2". At the bottom, there is a footer: "© 2021 - Pruebas_PAGES_Core - [Privacy](#)".

On the left, the Visual Studio file explorer shows the project structure for "Pruebas_PAGES-Core". The "Pages" folder is expanded, showing "Shared" and "Index.cshtml.cs" (which is selected). Other files include "appsettings.json", "Program.cs", and "Startup.cs".

index.cshtml

```
@page
@model IndexModel
@{
    ViewData["Title"] = "CalculoSimple vía RAZOR";
}

<div>
    <form method="post">
        <fieldset>
            <legend>Calculo Simple:</legend>
```



```
<label for="unLado_valor unLado_unidadmedida">Un lado: </label>
<input type="number" asp-for="unLado_valor" required />
<input type="text" asp-for="unLado_unidadmedida" size="4" placeholder="cm" required />
<br />
<label for="otroLado_valor otroLado_unidadmedida">Otro lado: </label>
<input type="number" asp-for="otroLado_valor" required />
<input type="text" asp-for="otroLado_unidadmedida" size="4" placeholder="cm" required />
<br />
<br />
<button type="submit">Calcular y mostrar resultados</button>
</fieldset>
<br />
<fieldset>
  <label for="perimetro">Perímetro: </label>
  <input asp-for="perimetro" disabled />
  <br />
  <label for="area">Área: </label>
  <input asp-for="area" disabled />
</fieldset>
</form>
</div>
```

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Logging;
using System;

namespace Pruebas_PAGES_Core.Pages
{
    public class IndexModel : PageModel
    {
        [BindProperty]
        public double unLado_valor { get; set; }
        [BindProperty]
        public string unLado_unidadmedida { get; set; }
        [BindProperty]
        public double otroLado_valor { get; set; }
        [BindProperty]
        public string otroLado_unidadmedida { get; set; }

        [BindProperty]
        public string perimetro { get; private set; }
    }
}
```

```
[BindProperty]
public string area { get; private set; }

public IActionResult OnGet()
{
    unLado_valor = 2;
    unLado_unidadmedida = "m";
    otroLado_valor = 300;
    otroLado_unidadmedida = "cm";

    return Page();
}

public IActionResult OnPost()
{
    CalculoSimple.CalculoSimple calculadora = new CalculoSimple.CalculoSimple();

    try
    {
        calculadora.setDato_unlado(new Magnitudes.Magnitud(unLado_valor, unLado_unidadmedida));
        calculadora.setDato_otrolado(new Magnitudes.Magnitud(otroLado_valor, otroLado_unidadmedida));
    }
    catch (ArgumentOutOfRangeException ex)
    {
        //MessageBox.Show("Alguna de las unidades de medida no está contemplada en las conversiones: " + ex.Message);
    }
    catch (System.IO.FileNotFoundException ex)
    {
        //MessageBox.Show("No se ha encontrado el archivo con la lista de conversiones entre unidades de medida: " + ex.Message);
    }
    catch (ArgumentException ex)
    {
        //MessageBox.Show("Problemas leyendo el archivo con la lista de conversiones entre unidades de medida: " + ex.Message);
    }
    catch (Exception ex)
    {
        //MessageBox.Show("Problemas en alguna conversión entre unidades de medida: " + ex.Message);
    }

    perimetro = calculadora.getResultado_perimetro().ToString();
    area = calculadora.getResultado_area().ToString();

    return Page();
}
```

```
}  
}
```