

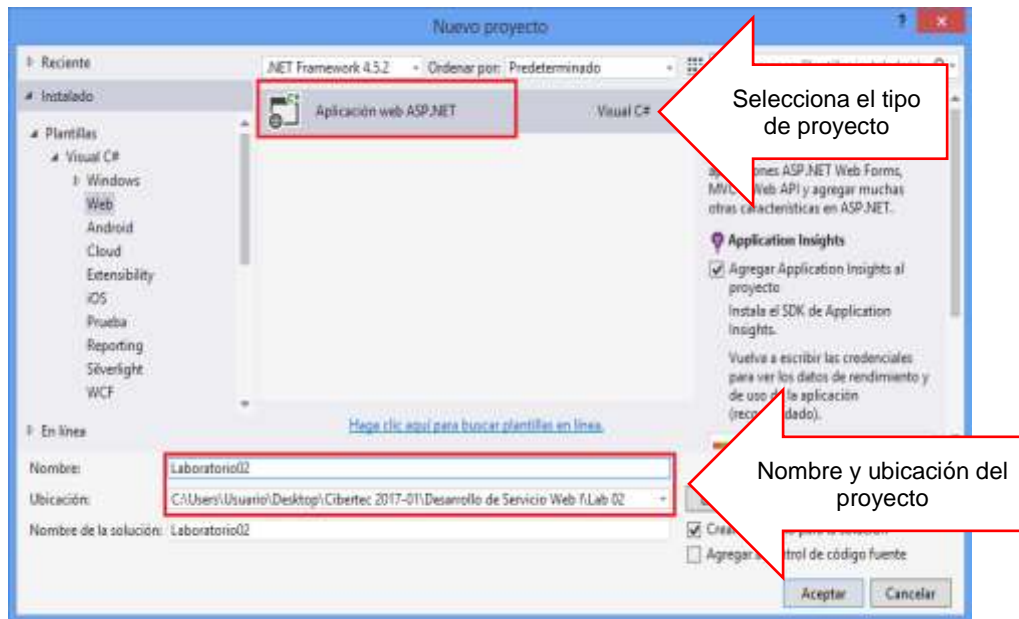
### Creando una aplicación ASP.NET MVC

Implemente un proyecto ASP.NET MVC donde permita listar y registrar los productos desde la web. Valide los datos del producto utilizando anotaciones.

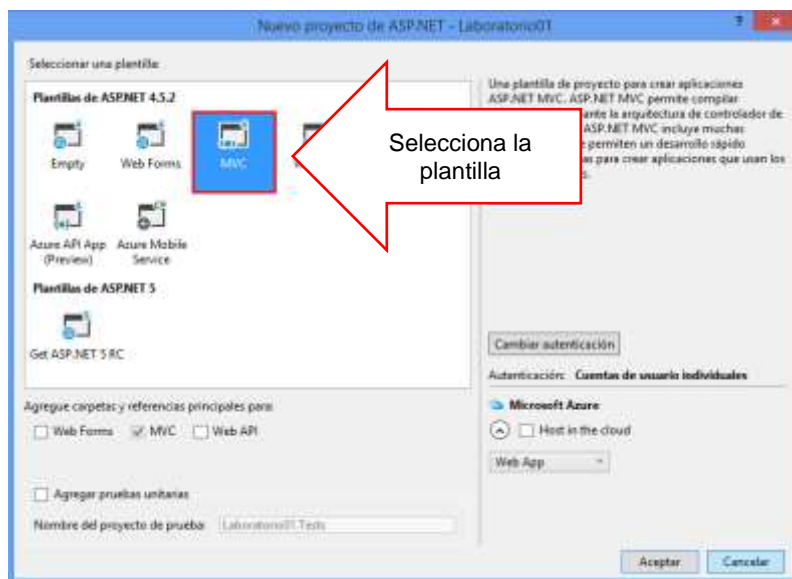
#### Creando el proyecto

Iniciamos Visual Studio 2015 y creamos un nuevo proyecto:

1. Seleccionar el proyecto Web.
2. Seleccionar el Framework: 4.5.2
3. Seleccionar la plantilla Aplicación web de ASP.NET
4. Asignar el nombre del proyecto
5. Presionar el botón ACEPTAR

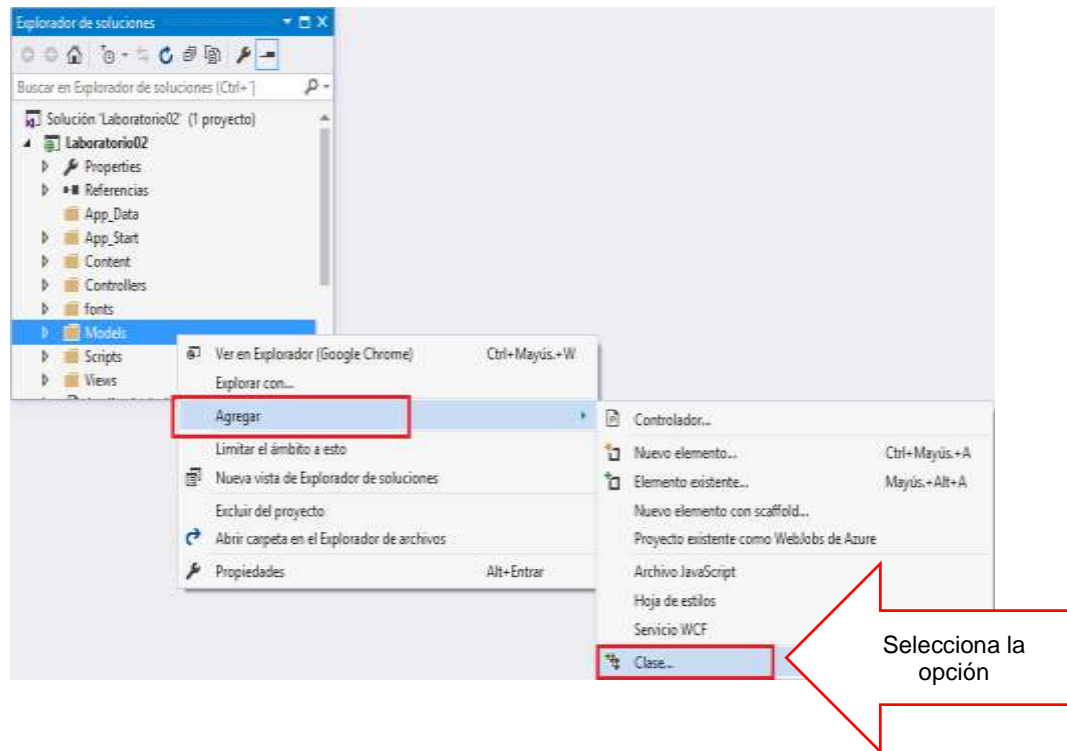


A continuación, seleccionar la plantilla del proyecto MVC. Presiona el botón ACEPTAR

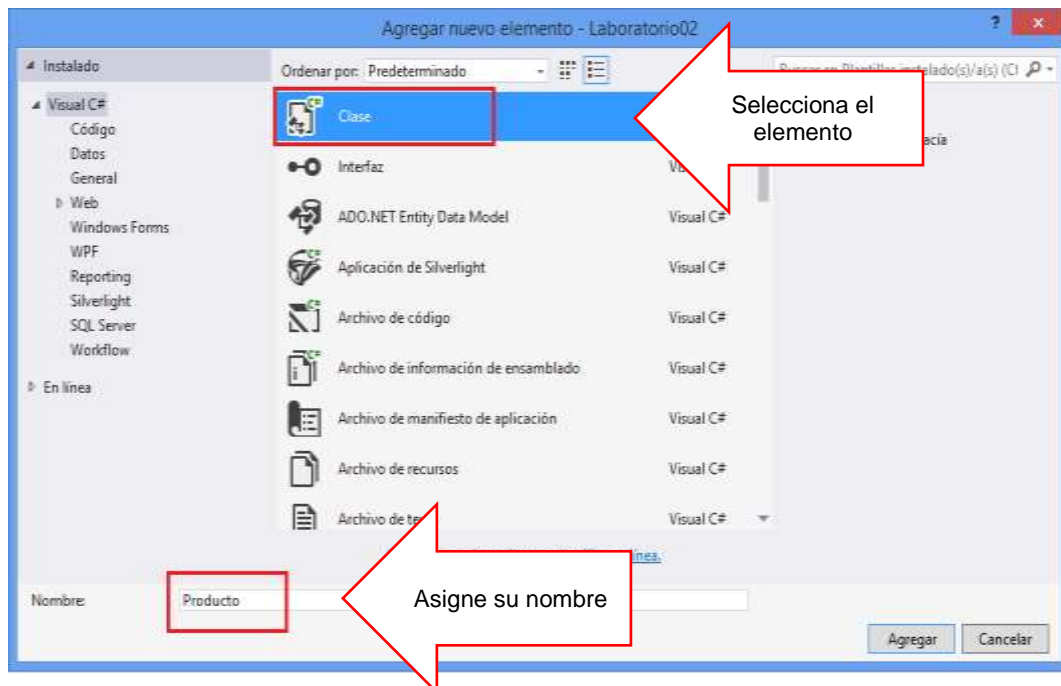


### Trabajando con el Modelo

Primero, creamos una clase en la carpeta Models: Agregar una clase llamada Producto, tal como se muestra

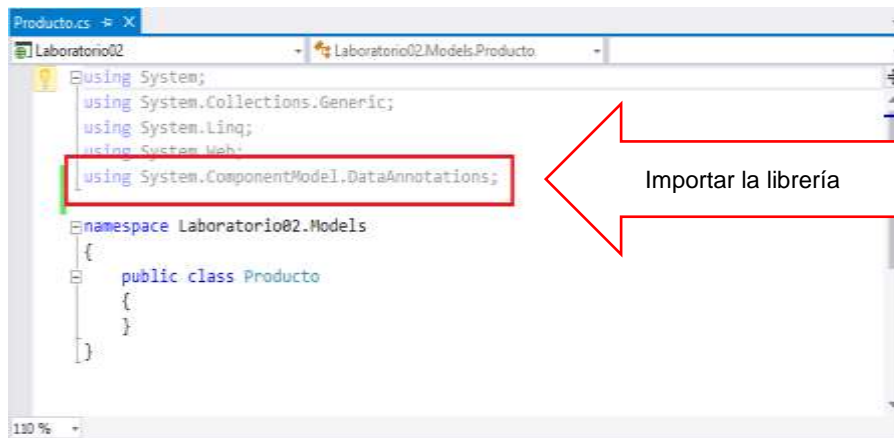


En la ventana **Agregar nuevo elemento**, selecciona el elemento **Clase** y asigne el nombre: Producto, presiona el botón Agregar



## Desarrollo de Servicios Web I

En la clase importar la librería de Anotaciones y Validaciones

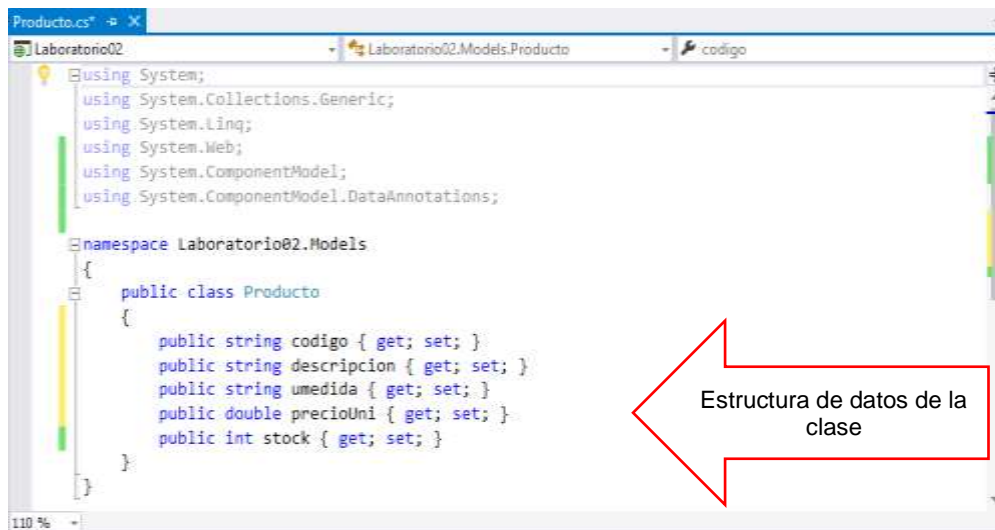


The screenshot shows the Visual Studio IDE with the file 'Producto.cs' open. The code includes several 'using' statements. A red box highlights the line 'using System.ComponentModel.DataAnnotations;'. A red arrow points from the text 'Importar la librería' to this line.

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

namespace Laboratorio02.Models
{
    public class Producto
    {
    }
}
```

En la clase Producto, primero, defina la estructura de datos de la clase, tal como se muestra.

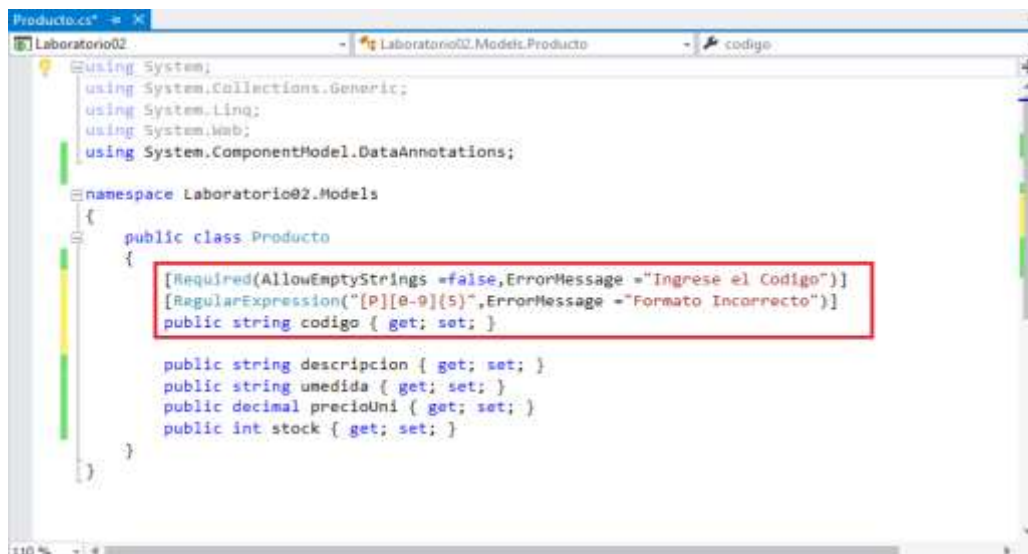


The screenshot shows the Visual Studio IDE with the file 'Producto.cs' open. The code defines the 'Producto' class with several public properties. A red box highlights the property declarations. A red arrow points from the text 'Estructura de datos de la clase' to this box.

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

namespace Laboratorio02.Models
{
    public class Producto
    {
        public string codigo { get; set; }
        public string descripcion { get; set; }
        public string unedida { get; set; }
        public double precioUni { get; set; }
        public int stock { get; set; }
    }
}
```

A continuación validamos el campo código: no debe estar vacío y su formato es P99999, tal como se muestra



The screenshot shows the Visual Studio IDE with the file 'Producto.cs' open. The code defines the 'Producto' class with several public properties. A red box highlights the validation attributes for the 'codigo' property. A red arrow points from the text 'A continuación validamos el campo código: no debe estar vacío y su formato es P99999, tal como se muestra' to this box.

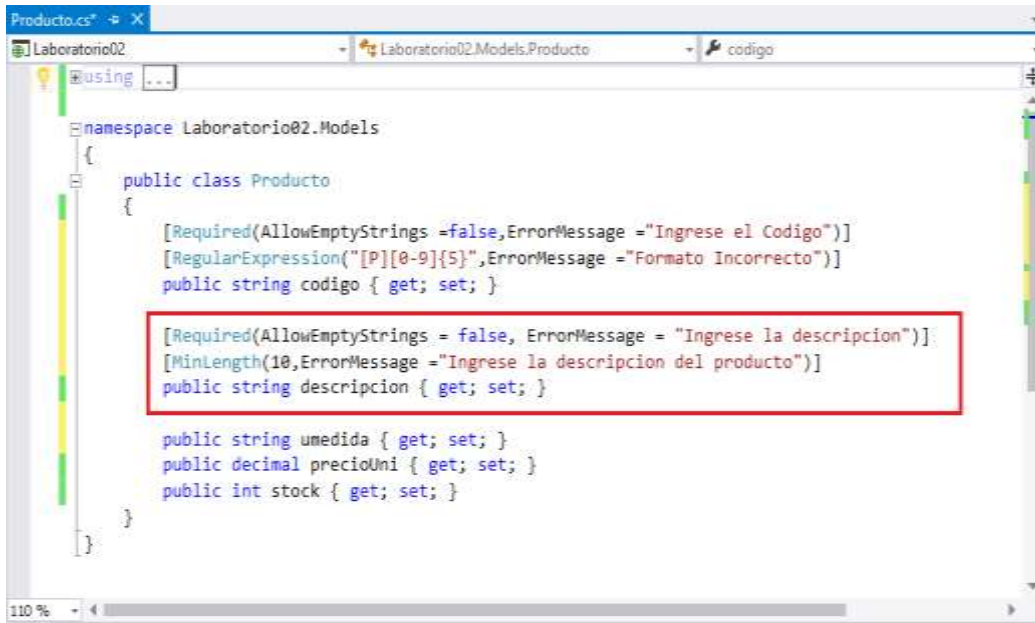
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace Laboratorio02.Models
{
    public class Producto
    {
        [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese elCodigo")]
        [RegularExpression("[P][0-9]{5}", ErrorMessage = "Formato Incorrecto")]
        public string codigo { get; set; }

        public string descripcion { get; set; }
        public string unedida { get; set; }
        public decimal precioUni { get; set; }
        public int stock { get; set; }
    }
}
```

## Desarrollo de Servicios Web I

Luego validamos el campo descripción: no debe estar vacío y la longitud mínima de caracteres es 10, tal como se muestra



```
Productos.cs -> X
Laboratorio02 -> Laboratorio02.Models.Producto -> codigo

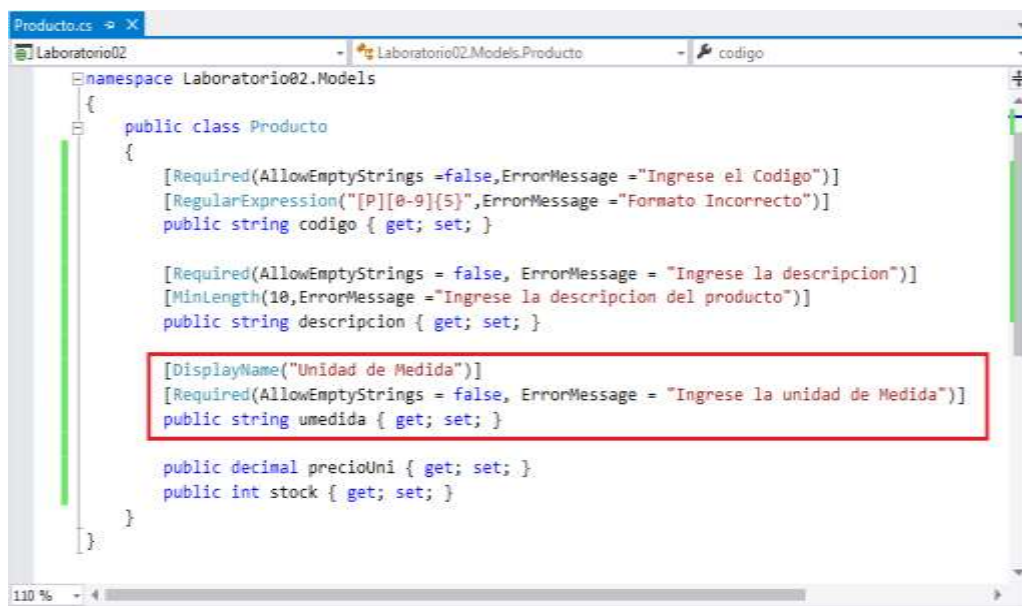
using ...

namespace Laboratorio02.Models
{
    public class Producto
    {
        [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el Codigo")]
        [RegularExpression("[P][0-9]{5}", ErrorMessage = "Formato Incorrecto")]
        public string codigo { get; set; }

        [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la descripcion")]
        [MinLength(10, ErrorMessage = "Ingrese la descripcion del producto")]
        public string descripcion { get; set; }

        public string umedida { get; set; }
        public decimal precioUni { get; set; }
        public int stock { get; set; }
    }
}
```

Continuamos con la validación del campo umedida, el cual será obligatorio, y asignamos el nombre a visualizar (DisplayName), importar la librería System.ComponentModel



```
Productos.cs -> X
Laboratorio02 -> Laboratorio02.Models.Producto -> codigo

namespace Laboratorio02.Models
{
    public class Producto
    {
        [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el Codigo")]
        [RegularExpression("[P][0-9]{5}", ErrorMessage = "Formato Incorrecto")]
        public string codigo { get; set; }

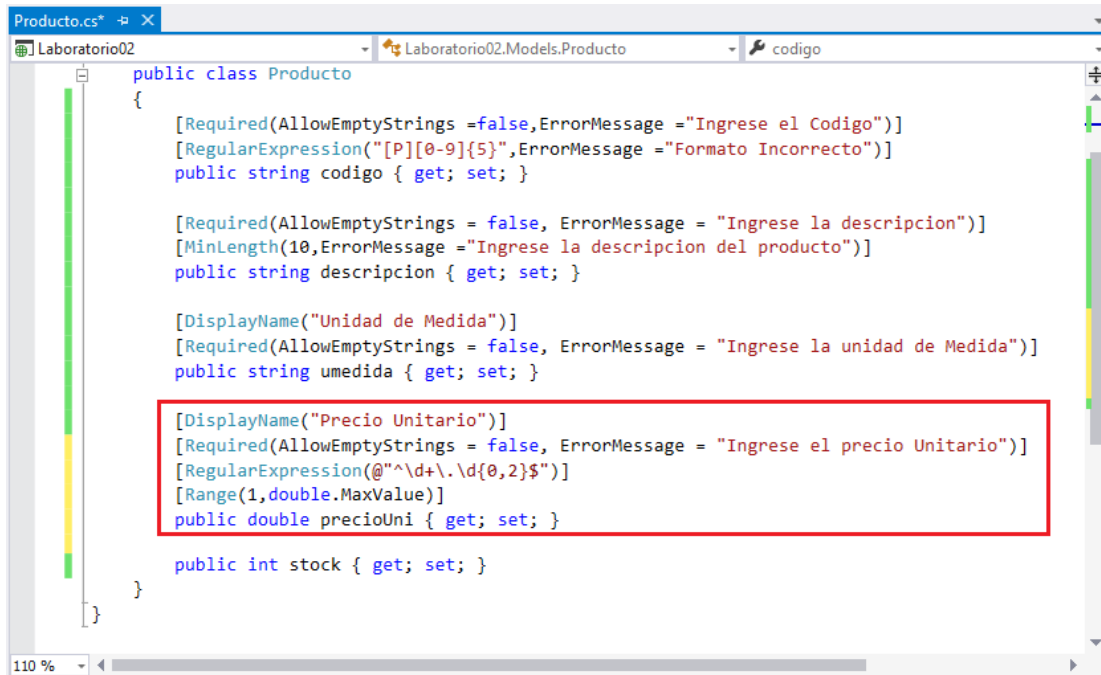
        [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la descripcion")]
        [MinLength(10, ErrorMessage = "Ingrese la descripcion del producto")]
        public string descripcion { get; set; }

        [DisplayName("Unidad de Medida")]
        [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la unidad de Medida")]
        public string umedida { get; set; }

        public decimal precioUni { get; set; }
        public int stock { get; set; }
    }
}
```

## Desarrollo de Servicios Web I

Luego validamos el campo preUni: asignar un Nombre para visualizarlo, indicar que el campo obligatorio y solo debe contener dos decimales (RegularExpression) y su rango de valores se encuentra entre 1 al valor máximo del tipo de dato.



```
Producto.cs* X
Laboratorio02
Laboratorio02.Models.Producto
codigo

public class Producto
{
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese elCodigo")]
    [RegularExpression("[P][0-9]{5}", ErrorMessage = "Formato Incorrecto")]
    public string codigo { get; set; }

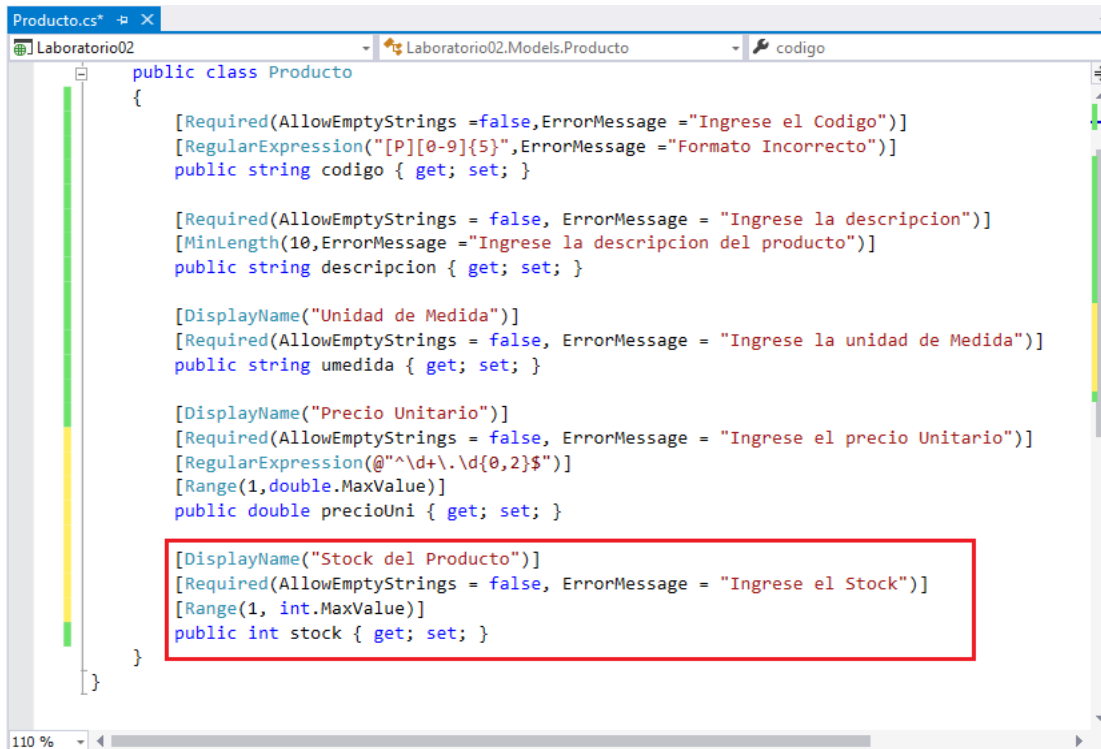
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la descripcion")]
    [MinLength(10, ErrorMessage = "Ingrese la descripcion del producto")]
    public string descripcion { get; set; }

    [DisplayName("Unidad de Medida")]
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la unidad de Medida")]
    public string umedida { get; set; }

    [DisplayName("Precio Unitario")]
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el precio Unitario")]
    [RegularExpression(@"^\d+\.\d{0,2}$")]
    [Range(1, double.MaxValue)]
    public double precioUni { get; set; }

    public int stock { get; set; }
}
```

Y por ultimo validamos el campo stock: asignamos un nombre, campo obligatorio y cuyo rango de valores es 1 hasta el valor máximo del tipo de dato



```
Producto.cs* X
Laboratorio02
Laboratorio02.Models.Producto
codigo

public class Producto
{
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese elCodigo")]
    [RegularExpression("[P][0-9]{5}", ErrorMessage = "Formato Incorrecto")]
    public string codigo { get; set; }

    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la descripcion")]
    [MinLength(10, ErrorMessage = "Ingrese la descripcion del producto")]
    public string descripcion { get; set; }

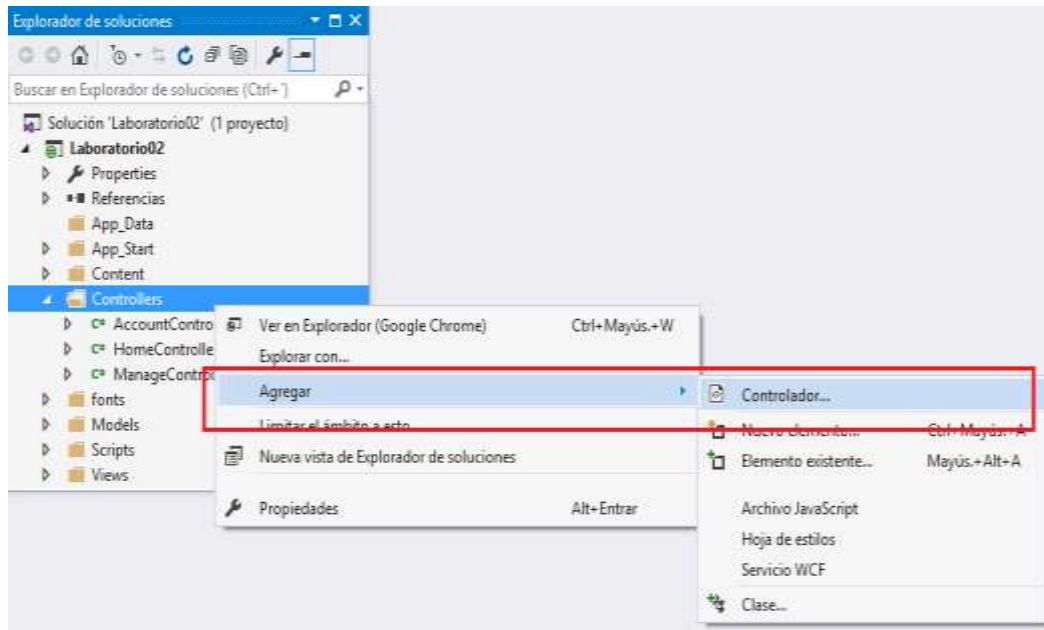
    [DisplayName("Unidad de Medida")]
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese la unidad de Medida")]
    public string umedida { get; set; }

    [DisplayName("Precio Unitario")]
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el precio Unitario")]
    [RegularExpression(@"^\d+\.\d{0,2}$")]
    [Range(1, double.MaxValue)]
    public double precioUni { get; set; }

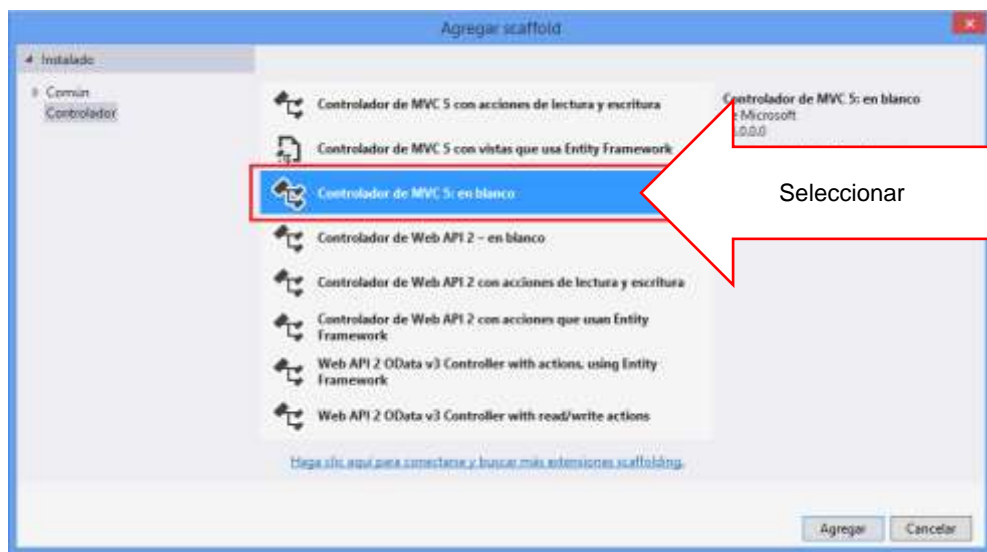
    [DisplayName("Stock del Producto")]
    [Required(AllowEmptyStrings = false, ErrorMessage = "Ingrese el Stock")]
    [Range(1, int.MaxValue)]
    public int stock { get; set; }
}
```

## Desarrollo de Servicios Web I

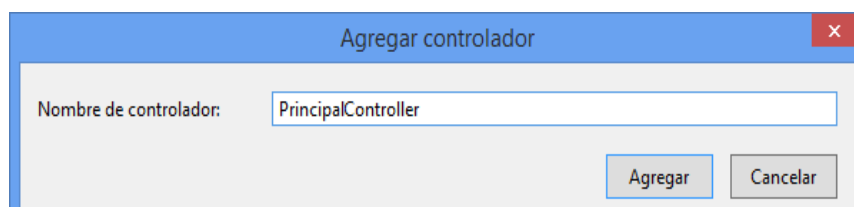
A continuación agregar en la carpeta Controller un controlador, tal como se muestra



En la ventana Scaffold, selecciona el controlador MVC 5 en blanco, tal como se muestra



A continuación ingrese el nombre del Controlador, presiona al botón Agregar

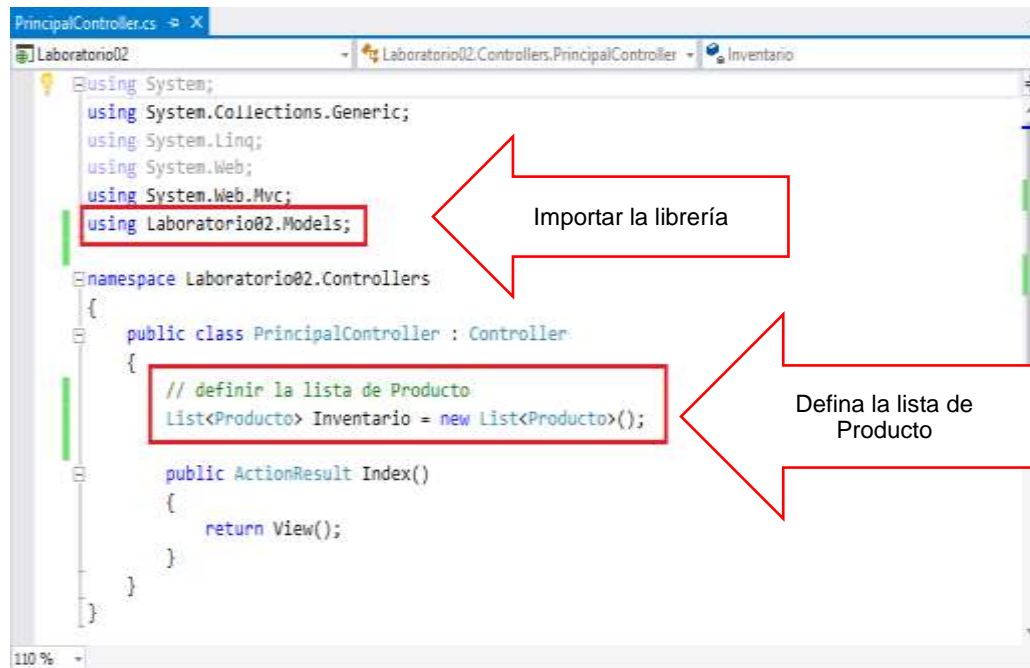




## Desarrollo de Servicios Web I

En la ventana de código PrincipalController:

- Importar la carpeta Models, la cual almacena la clase Producto
- Definir una lista de Producto, a nivel controlador



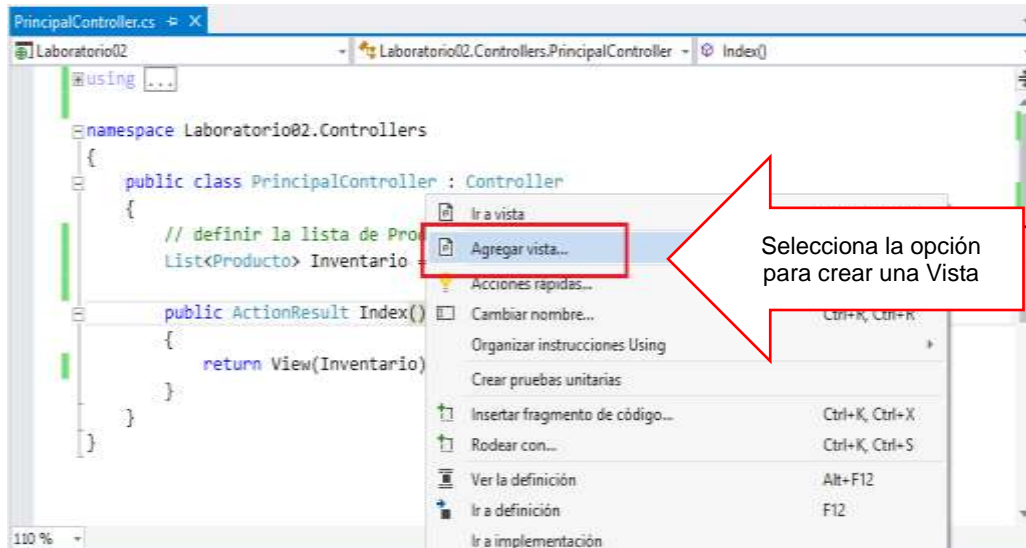
En el ActionResult Index(), enviar a la vista la lista de Producto, llamado Inventario. La sintaxis es:

```
static List<Producto> Inventario = new List<Producto>();
```



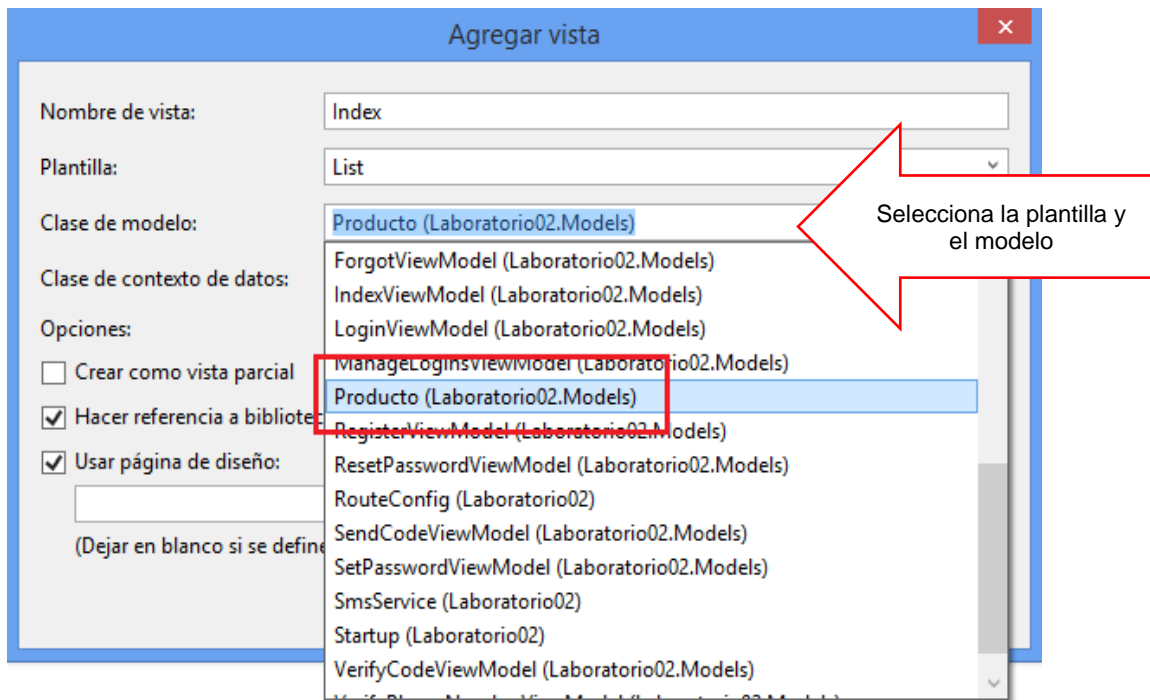
## Desarrollo de Servicios Web I

A continuación agregar una Vista a la acción Index: hacer click derecho a la acción y selecciona Agregar Vista, tal como se muestra



En la ventana Agregar Vista, aparece el nombre de la Vista: Index. No cambiar el nombre de la vista.

- Selecciona la plantilla: List
- Selecciona la clase de modelo: Producto, el cual listaremos los productos almacenados en la colección.
- Presionar el botón Agregar



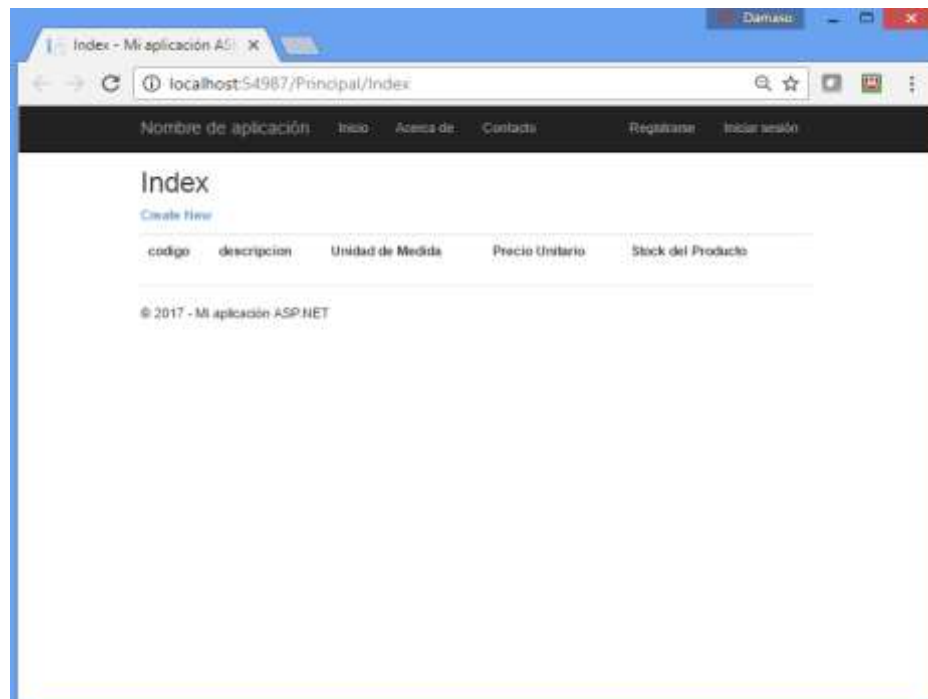


## Desarrollo de Servicios Web I

Generada la Vista, se visualiza tal como se muestra.

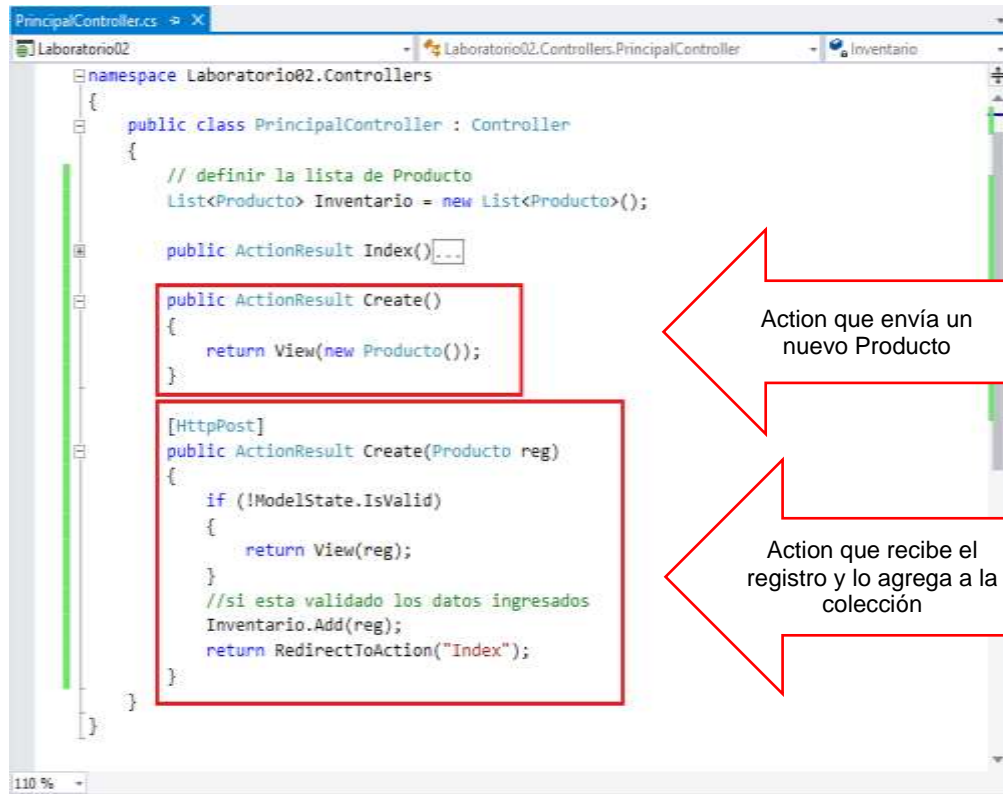


Ejecute la Vista Ctrl + F5, visualizando la ventana Index.

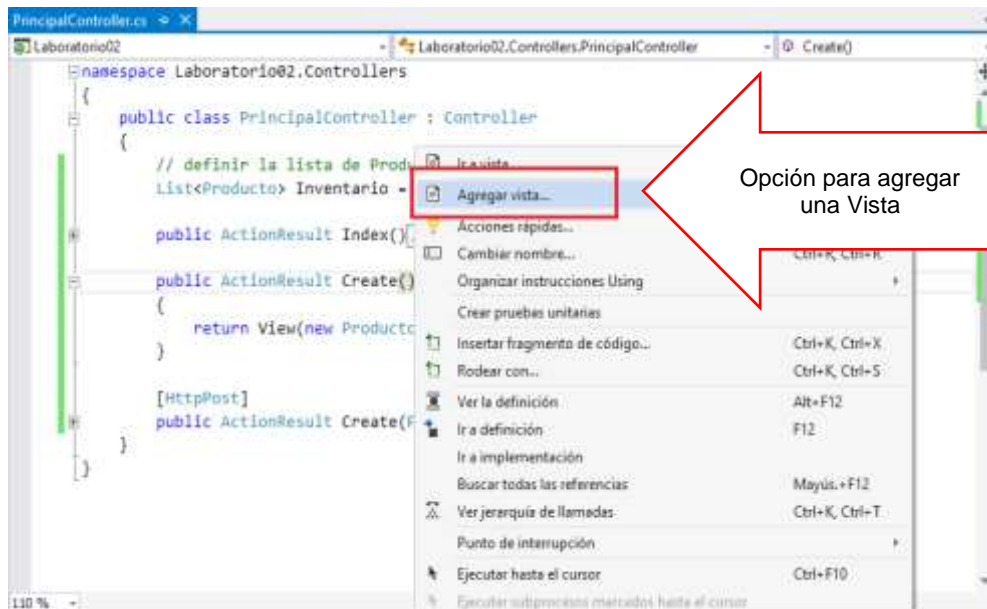


## Desarrollo de Servicios Web I

En el controlador Principal, defina la acción Create, el cual envía la estructura de la clase Producto, y recibe los datos para ser validados y agregados a la colección.



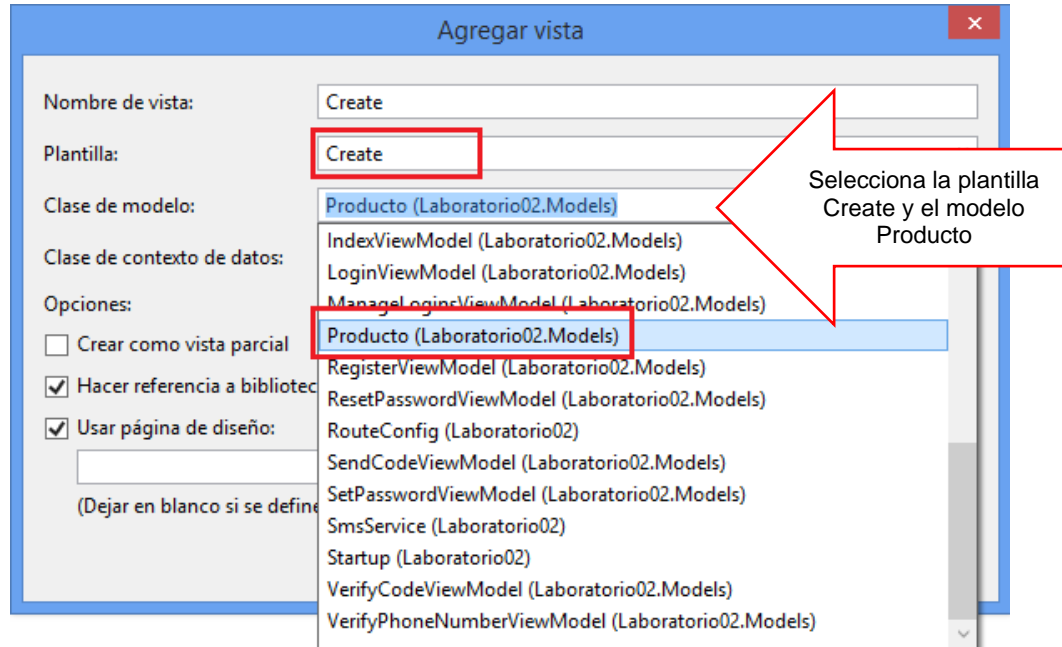
Definida la acción Create, agregar una vista: hacer click derecho a la acción y seleccionar la opción Agregar Vista, tal como se muestra



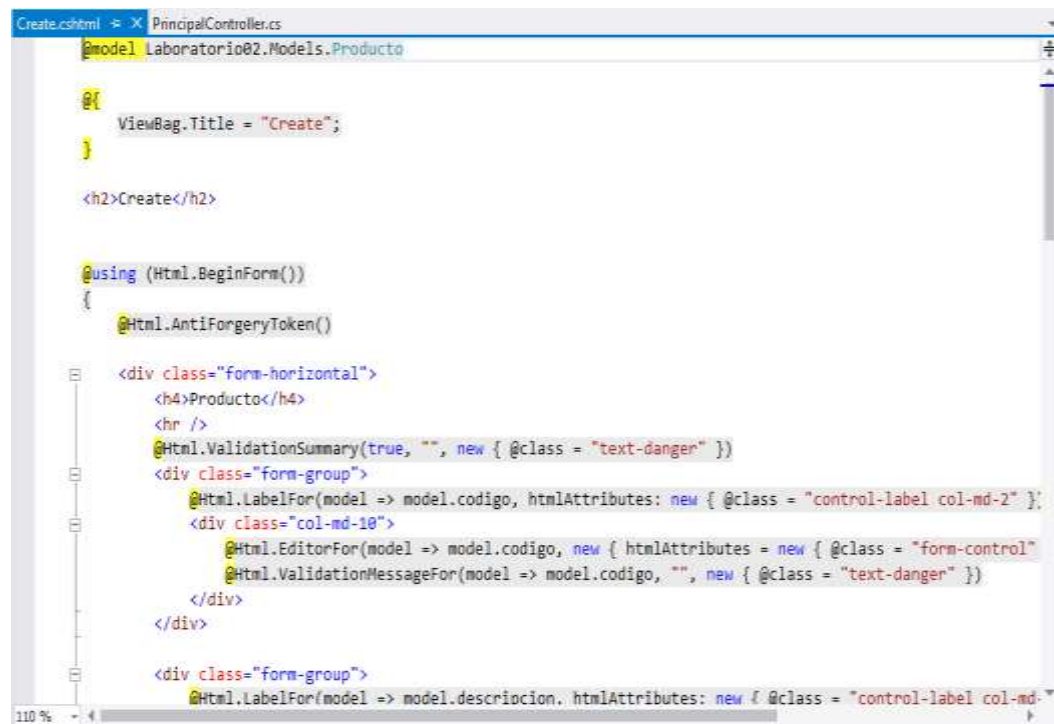
## Desarrollo de Servicios Web I

En la ventana Agregar Vista, aparece el nombre de la Vista: Create. No cambiar el nombre de la vista.

- Selecciona la plantilla: Create
- Selecciona la clase de modelo: Producto, el cual ingresamos los datos de un producto a la página.
- Presiona el botón Agregar



Página Create creada por la plantilla Create.



## Desarrollo de Servicios Web I

Ejecuta el proyecto, ingresar a la vista Create, los datos de los productos. Sus campos se encuentran validados desde la definición de la clase, tal como se muestra.

The screenshot shows a web browser window with the address bar displaying 'localhost:54967/Principal/Create'. The page has a dark navigation bar with links: 'Nombre de aplicación', 'Inicio', 'Acerca de', 'Contacto', 'Regístrate', and 'Iniciar sesión'. The main content area is titled 'Create' and 'Producto'. Below this, there are five input fields with their respective validation messages:

- codigo**: Ingrese el Código
- descripcion**: Ingrese la descripción
- Unidad de Medida**: Ingrese la unidad de Medida
- Precio Unitario**: El campo Precio Unitario debe coincidir con la expresion regular ^d+\.?d(0,2)\$.
- Stock del Producto**: El campo Stock del Producto debe estar entre 1 y 2147483647.

At the bottom of the form, there is a 'Create' button and a 'Back to List' link.