Controlador – Vista – Modelado.

En un primer paso resolvimos toda la lógica de nuestra aplicación web sólo con el "Controlador", en un segundo paso separamos las actividades agrupando funcionalidades en el "Controlador" y la "Vista". Ahora implementaremos otro problema elemental pero introduciendo los tres elementos fundamentales de este patrón de programación: "Modelo", "Vista" y "Controlador".

Recordemos que cada una de las componentes del patrón MVC tienen un objetivo bien definido:

- Controlador: Es una clase o conjunto de clases que coordinan la comunicación entre las peticiones que hace el cliente (Navegador generalmente), el modelo que procesa los datos que llegan del cliente y comunica a las vistas para que muestren los datos peticionados por el cliente.

- Vistas: Definen como se mostrará la interfaces de usuario de la aplicación.

- Modelos: Se implementan las clases que resuelven la lógica de negocios de nuestra aplicación.

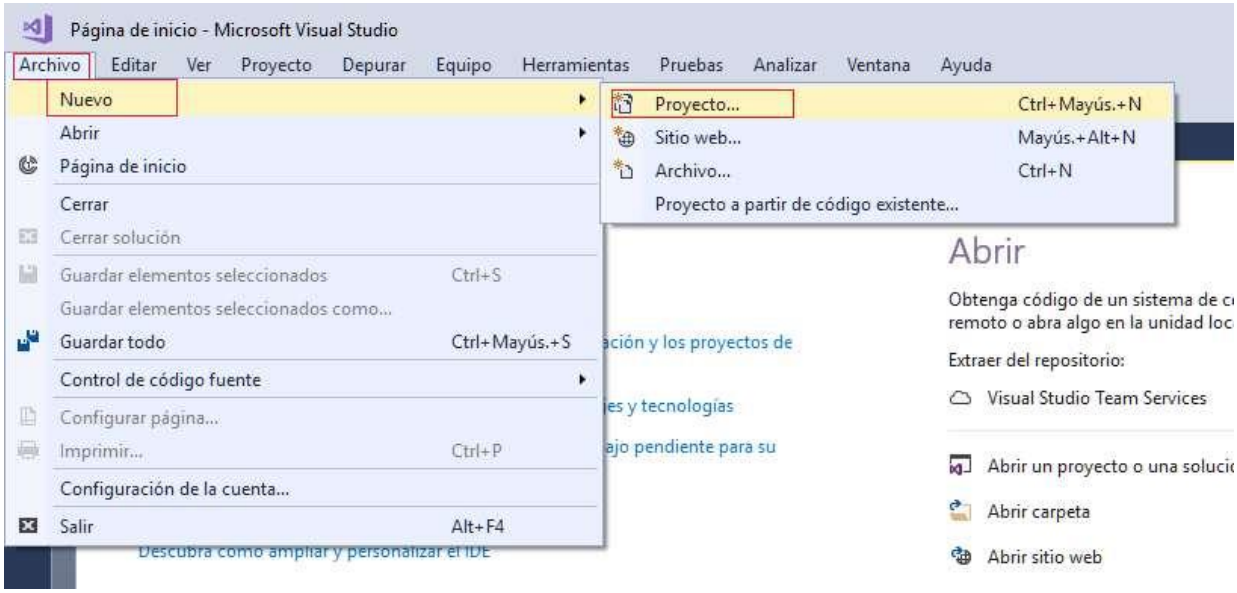
Problema

Desarrollar un libro de visitas a un sitio web. Debe haber un formulario que permita ingresar el nombre del visitante y los comentarios. Almacenar los datos en un archivo de texto.
La página principal debe tener dos hipervínculos. El primero que acceda a un formulario web para la carga del nombre del visitante y sus comentarios y el segundo hipervínculo debe mostrar todos los comentarios que han dejado los visitantes al sitio web.

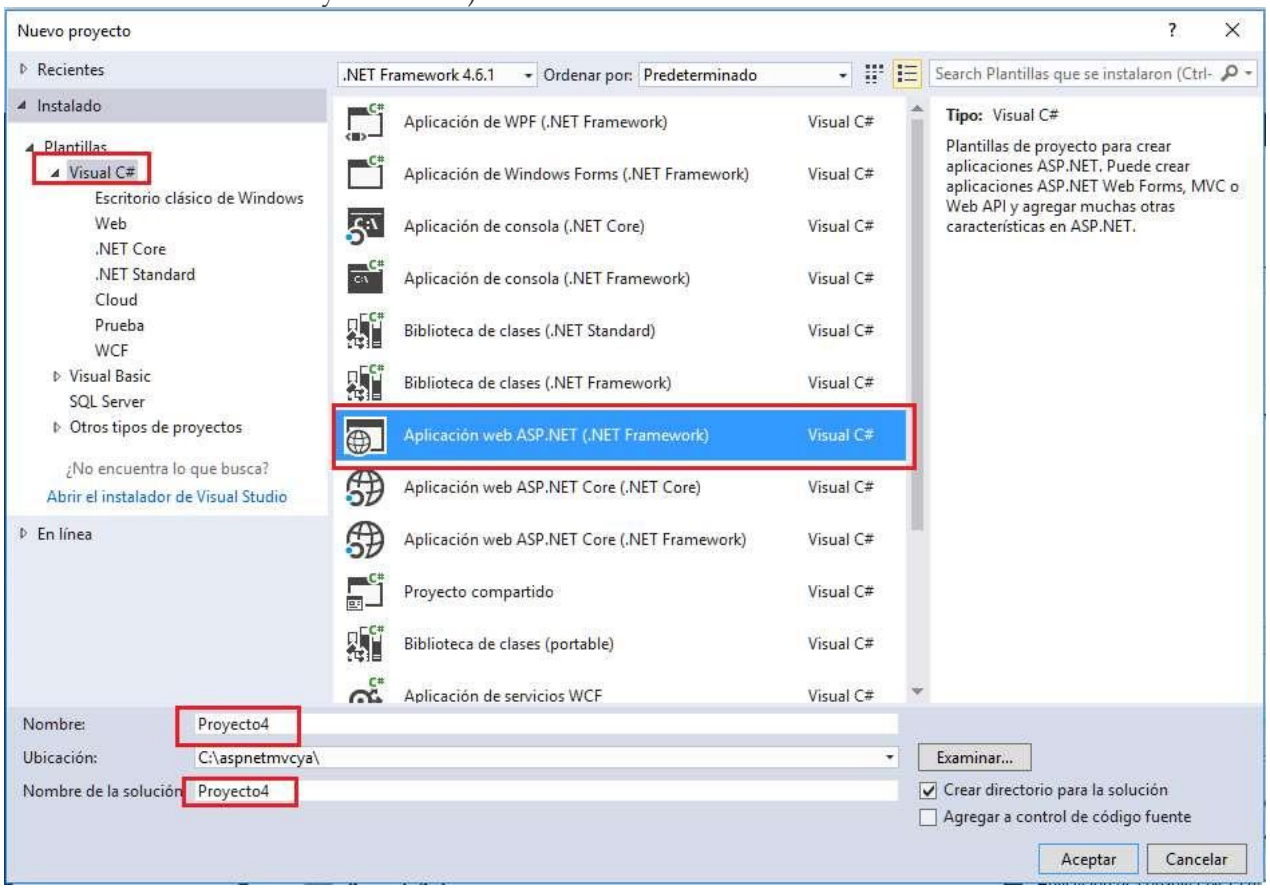
Resolveremos el problema enunciando uno a uno los pasos que debemos dar en el Visual Studio .Net para ir creando y codificando cada uno de los archivos necesarios en las carpetas "Controllers", "Views" y "Models".

1. Creamos el proyecto (Proyecto4)

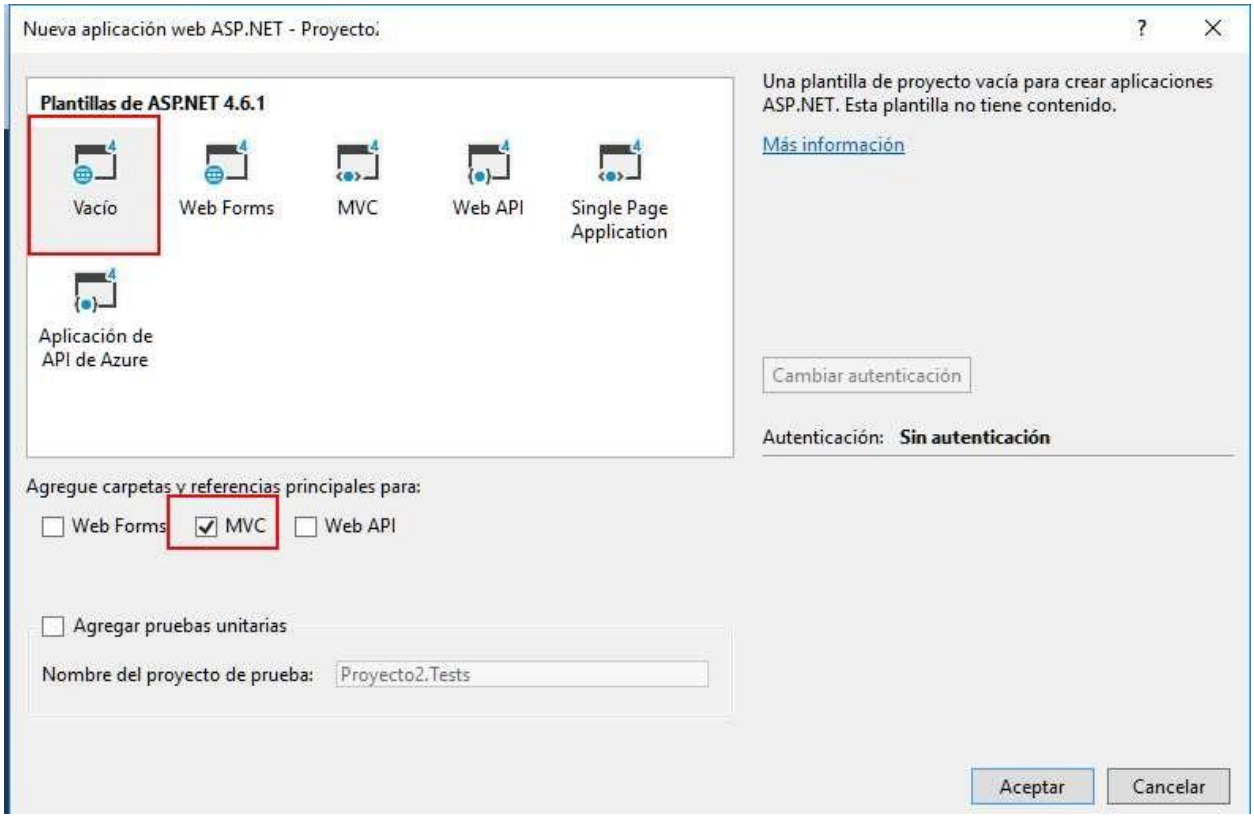
   Creación del proyecto. Para esto seleccionamos desde el menú la opción "Archivo" -> "Nuevo" -> "Proyecto..."



Aparece un diálogo donde debemos indicar del lado izquierdo que utilizaremos el lenguaje Visual C# y del lado de la derecha seleccionamos "Aplicación web ASP.NET (.Net Framework)" y en la parte inferior definimos el "nombre", "ubicación" y "nombre de la solución" (podemos usar el mismo texto para el "nombre de la solución" y "nombre"):

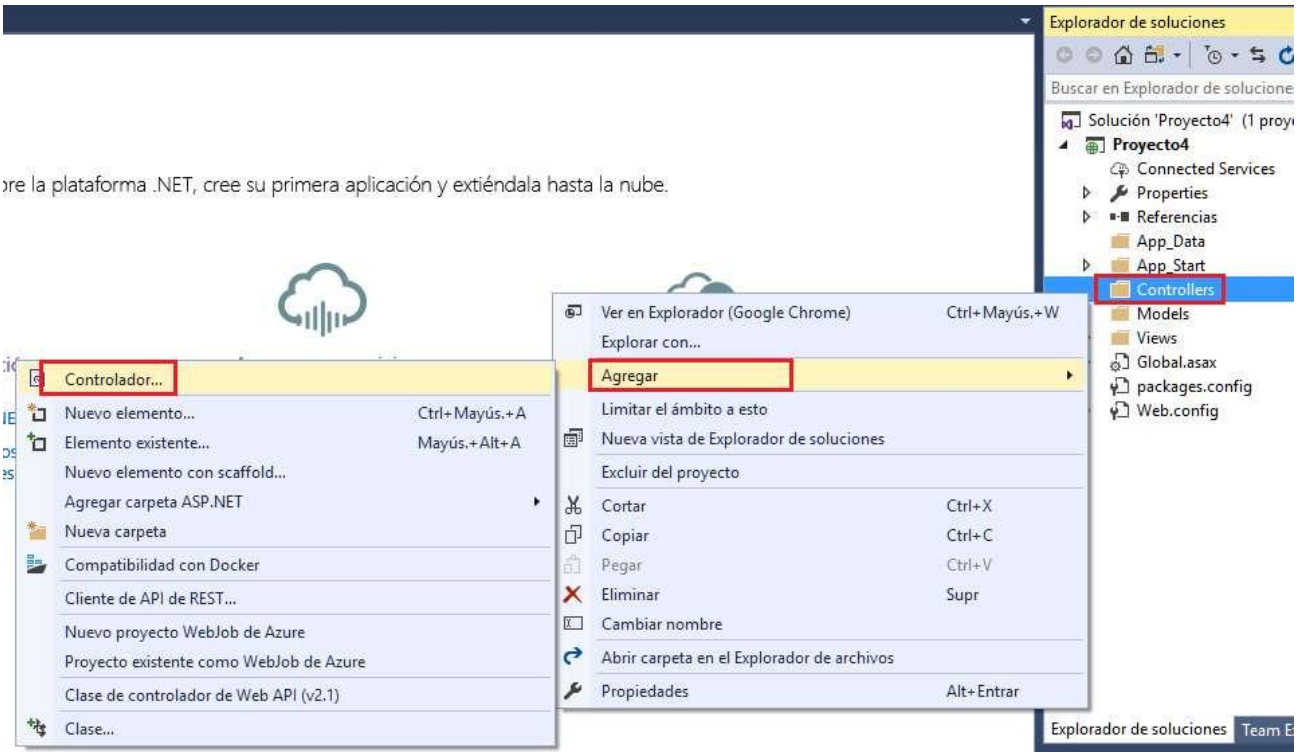


Aparece un segundo diálogo donde seleccionaremos que cree un proyecto vacío y utilice el patrón MVC:

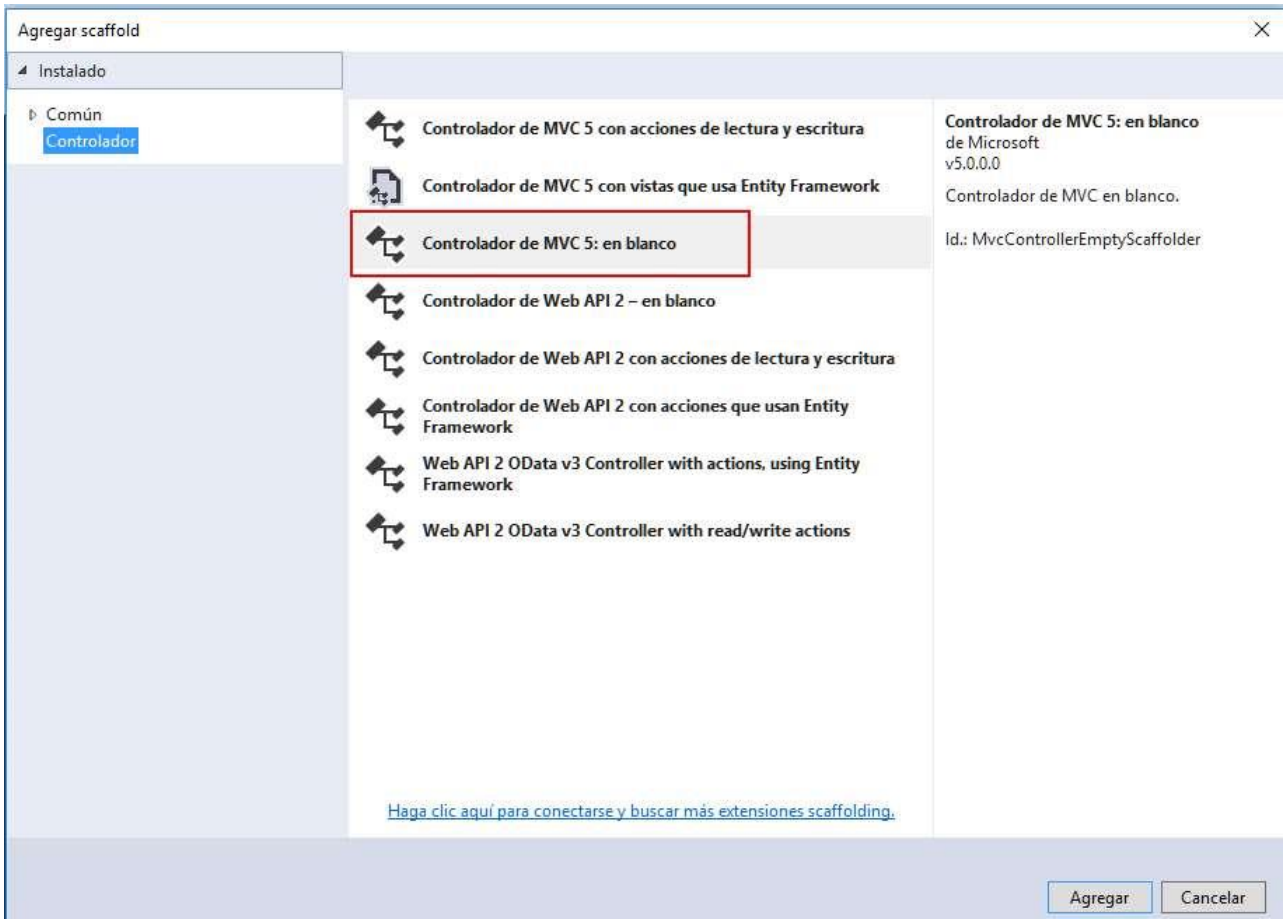


2. Ahora creamos el "Controlador" como hicimos en conceptos anteriores.

   El controlador principal siempre lo llamaremos "Home", para esto presionamos el botón derecho del mouse sobre la carpeta "Controllers" y seleccionar "Agregar" -> Controlador...:

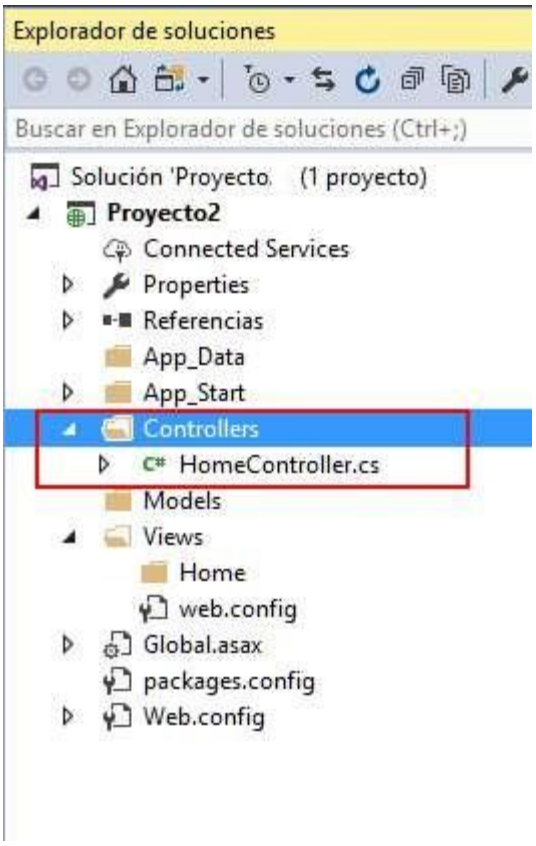


En el diálogo seleccionamos "Controlador de MVC 5: en blanco":



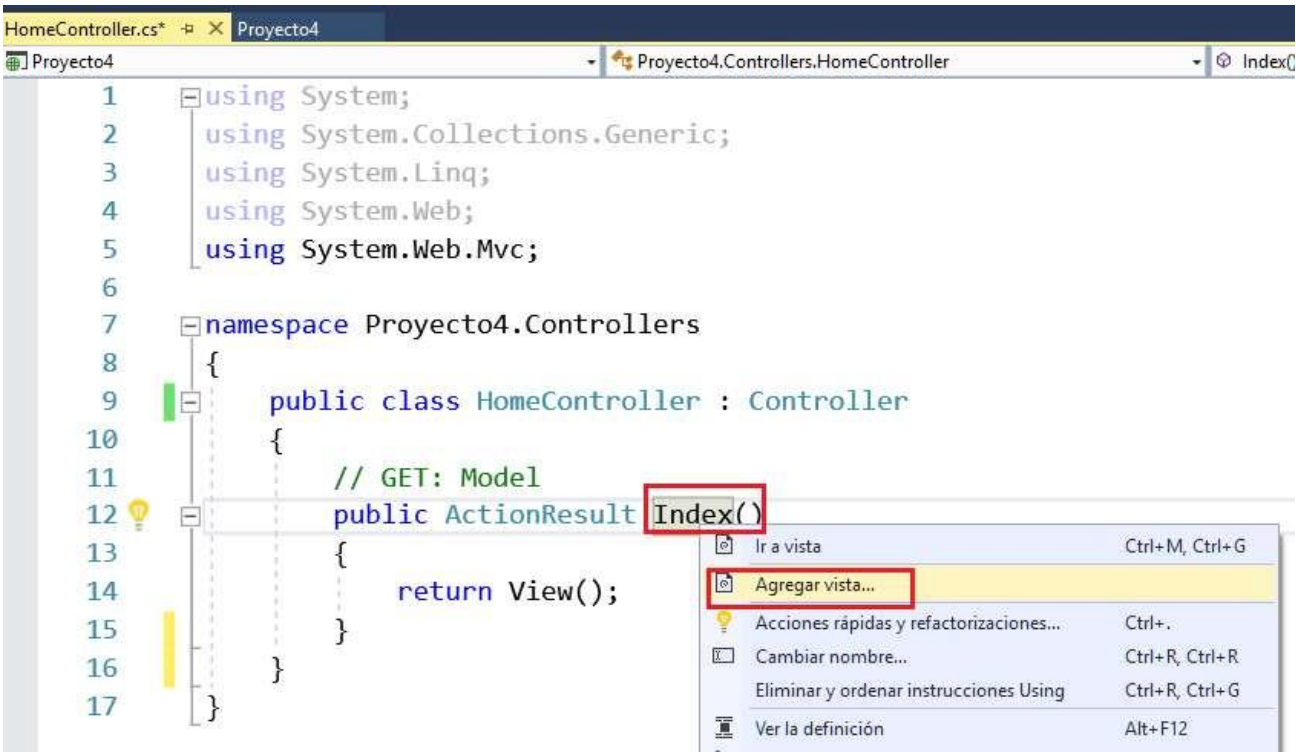
En el diálogo siguiente damos como nombre "HomeController" (por convención en ASP.NET MVC todos los controladores terminan con la palabra "Controller"):

Ahora si vemos el "Explorador de soluciones" podremos comprobar que en la carpeta "Controllers" tenemos un archivo llamado "HomeController.cs":



También se creó una carpeta llamada Home en la carpeta Views, en esta carpeta guardaremos las vistas para cada acción que definamos en el controlador.

3. Ahora generaremos la vista para la página principal del sitio, como sabemos tenemos que abrir el archivo HomeController y presionar el botón derecho del mouse sobre el método Index:



El nombre de la vista la dejamos con el nombre propuesto "Index", la plantilla debe ser "Empty (sin modelo) y finalmente sacamos el Check del "Usar página de diseño" (para evitar que se agreguen otros archivos que veremos más adelante y que hace más complejo el problema):

Si vemos ahora el "Explorador de soluciones" podemos observar que se ha creado un archivo llamado "Index.cshtml" en la carpeta "Views", subcarpeta "Home".

El contenido de este archivo lo modificamos para que muestre los dos enlaces de nuestro sitio web:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        <p><a href="/Home/FormularioVisita">Dejar comentarios en el libro de visit
as.</a></p>
        <p><a href="/Home/ListadoVisitas">Comentarios de visitantes.</a></p>
    </div>
</body>
</html>
```

Ya podemos ejecutar la aplicación y ver la página principal del sitio web propuesto:

Crearemos ahora la vista que muestra el formulario donde el visitante dejará sus comentarios. Lo primero que hacemos es modificar el archivo HomeController agregando otro método:

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

namespace Proyecto4.Controllers
{
   public class HomeController : Controller
   {
      // GET: Model
      public ActionResult Index()
      {
         return View();
      }

      public ActionResult FormularioVisita()
      {
         return View();
```

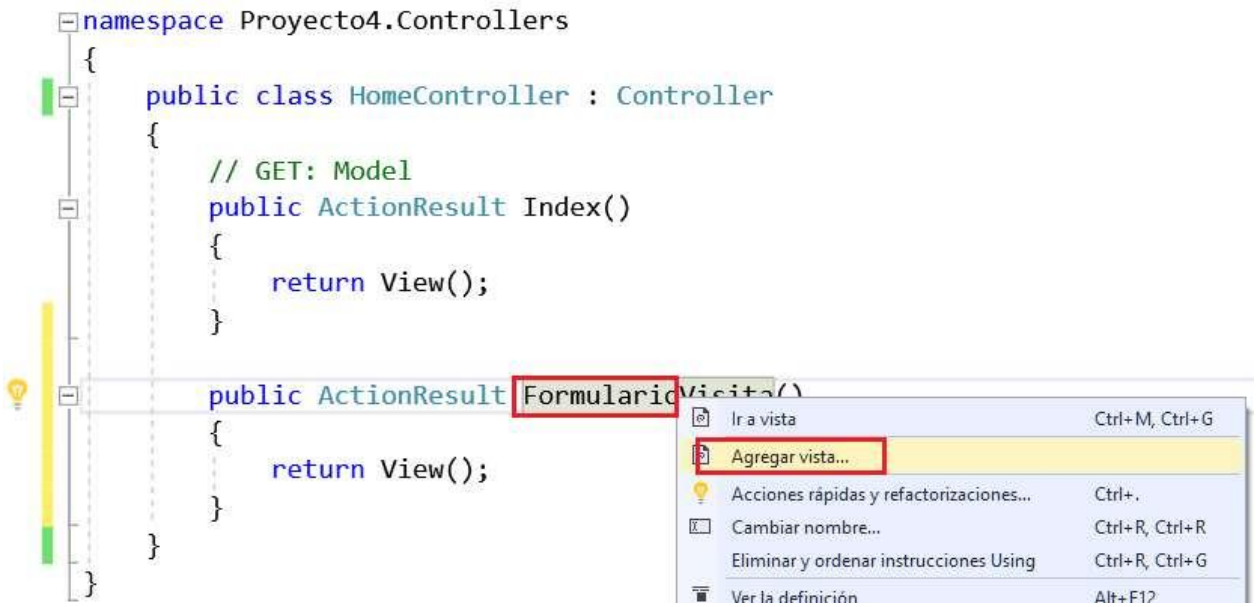
```
    }
  }
}
```

Hemos planteado el método FormularioVisita que sabemos que se ejecutará cuando el usuario ingrese la URL:

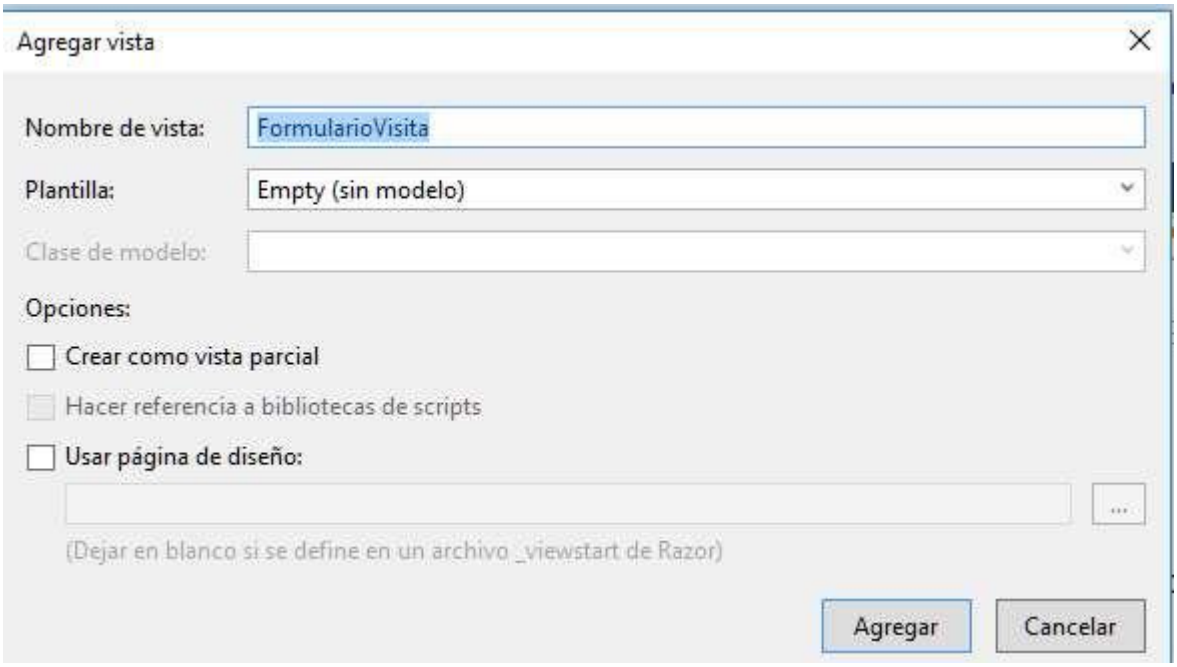
```
http://localhost/Home/FormularioVisita
```

Esto sucede cuando el visitante elige el primer enlace de la vista "Index.cshtml".

Debemos crear la vista asociada al método FormularioVisita(), como ya sabemos presionamos el botón derecho del mouse sobre el nombre del método y seleccionamos "Agregar Vista...":



El nombre de la vista la dejamos con el nombre propuesto "FormularioVisita", la plantilla debe ser "Empty (sin modelo) y finalmente sacamos el Check del "Usar página de diseño" (para evitar que se agreguen otros archivos que veremos más adelante y que hace más complejo el problema):



Si vemos ahora el "Explorador de soluciones" podemos observar que se ha creado un archivo llamado "FormularioVisita.cshtml" en la carpeta "Views", subcarpeta "Home".

El contenido de este archivo lo modificamos para que muestre el formulario HTML que permita ingresar el nombre de una persona y sus comentarios:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>FormularioVisita</title>
</head>
<body>
    <div>
        <form method="post" action="/Home/CargaDatos" >
```

```
        <p>Nombre:<input type="text" name="nombre" /></p>
        <p>Comentarios<br />
        <textarea rows="10" cols="120"  name="comentarios"></textarea></p>
        <p><input type="submit" value="Confirmar" /></p>
      </form>
   </div>
</body>
</html>
```

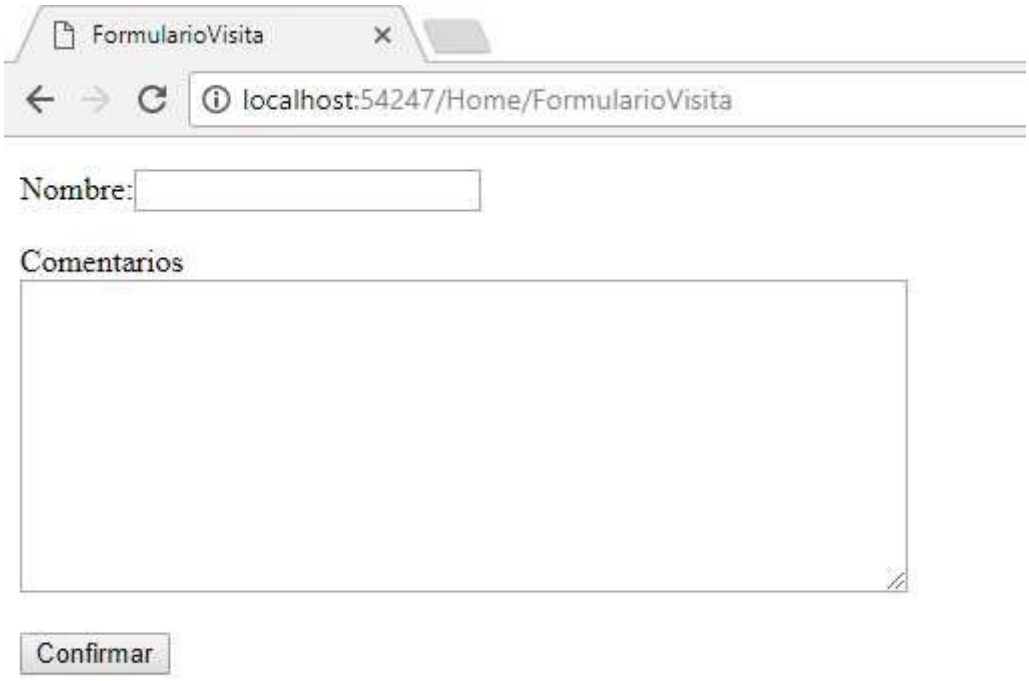
Hemos codificado un formulario HTML con un control "input" de tipo "text" y un control de tipo "textarea". Además del botón de tipo "submit" que envía los datos al servidor.

La propiedad action del elemento form indica que URL de nuestro controlador debe atrapar el envío de los datos del formulario que carga el operador:

```
      <form method="post" action="/Home/CargaDatos" >
```

Significa que en el controlador HomeController debemos plantear un método llamado CargaDatos.

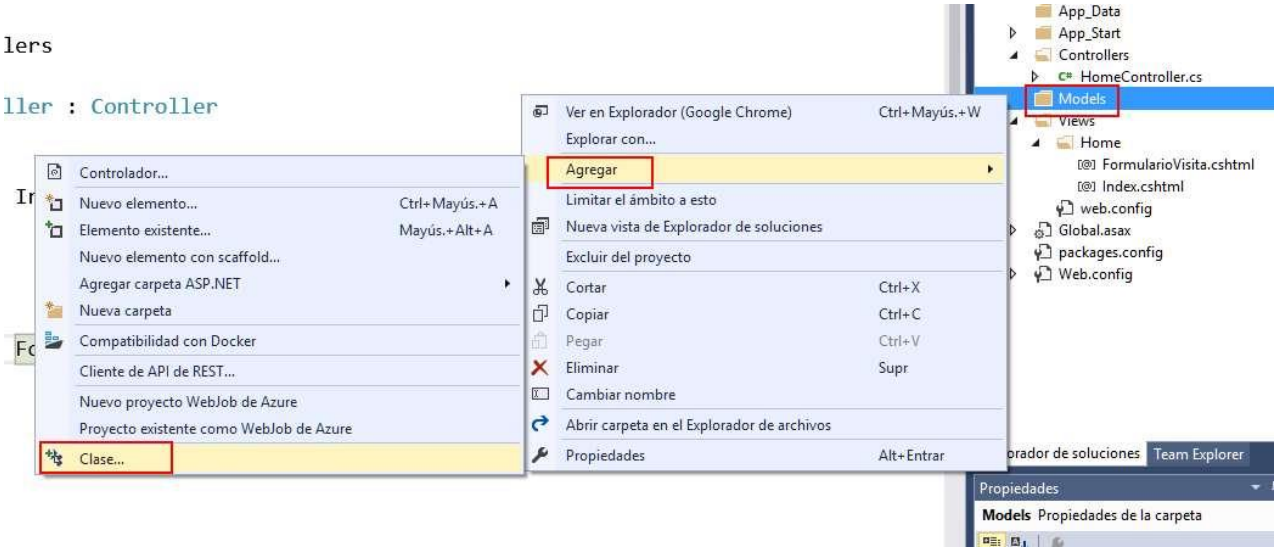
Si ejecutamos la aplicación veremos que podemos ver el formulario:



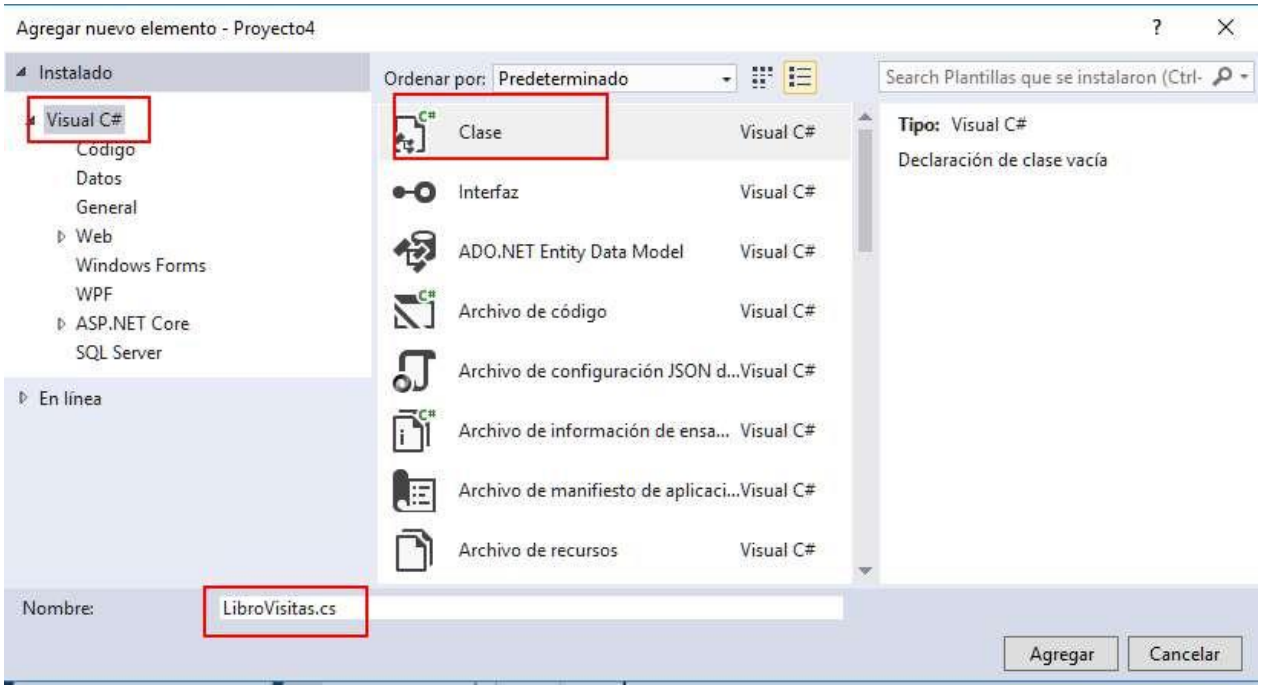
Pero cuando presionamos el botón de "Confirmar" nos muestra el error 404 que retorna el servidor:

4. Primero crearemos el modelo que tiene como responsabilidad almacenar los datos del visitante que serán comunicados por el controlador.

Presionamos el botón derecho del mouse sobre la carpeta Models y seleccionamos la opción "Agregar" -> "Clase...":



Creamos la clase "LibroVisitas.cs":



Codificamos la clase para permitir almacenar los datos en el archivo de texto:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.Hosting;
using System.IO;

namespace Proyecto4.Models
{
    public class LibroVisitas
    {
        public void Grabar(string nombre, string comentarios)
        {
            StreamWriter archivo = new StreamWriter(HostingEnvironment.MapPath("
~") + "/App_Data/datos.txt", true);
            archivo.WriteLine("Nombre:" + nombre + "<br>Comentarios:" + comentario
s + "<hr>");
            archivo.Close();
        }

    }
}
```

El método Grabar recibe dos string y dentro del mismo procedemos a crear un objeto de la clase StreamWriter que se encuentra en el espacio de nombres System.IO (Debemos disponer el using respectivo)

Al constructor de la clase StreamWriter le indicamos el path o camino donde se almacena el archivo de texto llamado "datos.txt", utilizamos el método MapPath del objeto HostingEnvironment que nos devuelve la ruta donde se almacena nuestro proyecto. Le concatenamos la carpeta "App_Data" donde se debe almacenar el archivo de texto.
El segundo parámetro de MapPath al pasar un true indicamos que si ya existe el archivo lo abra para añadir datos al final.

Mediante el método WriteLine procedemos a grabar en el archivo de texto los parámetros nombre y comentarios.

Finalmente cerramos el archivo llamando al método Close.

5. Ahora crearemos otra acción en el archivo HomeController donde llamaremos al modelo para que efectúe la carga de datos. Abrimos el archivo HomeController y codificamos:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Proyecto4.Models;

namespace Proyecto4.Controllers
{
    public class HomeController : Controller
    {
        // GET: Model
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult FormularioVisita()
        {
            return View();
        }

        public ActionResult CargaDatos()
        {
            string nombre = Request.Form["nombre"].ToString();
            string comentarios = Request.Form["comentarios"].ToString();
            LibroVisitas libro = new LibroVisitas();
            libro.Grabar(nombre, comentarios);
```

```
34.             return View();
35.         }
36.     }
37. }
```

La acción CargaDatos se ejecuta cuando el visitante confirma los datos del formulario de visitas.
Primero recuperamos los datos ingresados por el visitante en los dos controles HTML:

```
            string nombre = Request.Form["nombre"].ToString();
            string comentarios = Request.Form["comentarios"].ToString();
```

Seguidamente creamos un objeto de la clase LibroVisitas y llamamos al método Grabar:

```
            LibroVisitas libro = new LibroVisitas();
            libro.Grabar(nombre, comentarios);
```

Esta es la forma que tiene el "Controlador" de comunicarse con el "Modelo", en este caso para pedir que almacene los datos ingresados en el formulario.

Nos falta ahora crear una vista para la acción CargaDatos. Presionamos el botón derecho del mouse sobre el nombre del método "CargaDatos()" y seleccionamos "Agregar vista", luego codificamos el archivo CargaDatos.cshtml:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>CargaDatos</title>
</head>
<body>
```

```
    <div>
        <p>Los datos ingresados fueron almacenados.</p>
        <p>Gracias por sus comentarios.</p>
        <p><a href="/">Retornar</a></p>
    </div>
</body>
</html>
```

Podemos ahora seleccionar nuevamente el archivo Index.cshtml y ejecutar el proyecto. Cargamos los datos en el formulario y presionamos el botón de Confirmar:

El navegador muestra la vista, pero también en el servidor se almacenaron los datos cargados en el formulario HTML. Si vemos la carpeta "App_Data" encontraremos que se ha creado el archivo "datos.txt", que va creciendo a medida que se agregan comentarios.

38. Nos queda resolver la página que muestra todos los comentarios dejados por los visitantes.

    Primero codificaremos la responsabilidad de leer el archivo de texto que le corresponde al "Modelo". Abrimos el archivo "LibroVisitas.cs" y codificamos el método "Leer":

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Hosting;
using System.IO;

namespace Proyecto4.Models
{
    public class LibroVisitas
    {
        public void Grabar(string nombre, string comentarios)
        {
            StreamWriter archivo = new StreamWriter(HostingEnvironment.MapPath("
~") + "/App_Data/datos.txt", true);
            archivo.WriteLine("Nombre:" + nombre + "<br>Comentarios:" + comentario
s + "<hr>");
            archivo.Close();
        }

        public string Leer()
        {
            StreamReader archivo = new StreamReader(HostingEnvironment.MapPath(
"~") + "/App_Data/datos.txt");
            string todo = archivo.ReadToEnd();
            archivo.Close();
            return todo;
        }
    }
}
```

El método Leer procede a crear un objeto de la clase StreamReader e indicamos el path donde se encuentra el archivo a leer llamado "datos.txt".

Midente el método "ReadToEnd" recuperamos todos los caracteres almacenados en el archivo de texto y finalmente luego de cerrar el archivo procedemos a retornar un string con todos los caracteres leidos.

Ahora debemos agregar en el "Controlador" un método que responda al segundo hiperínculo de la página principal del sitio llamado "ListadoVisitas":

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Proyecto4.Models;

namespace Proyecto4.Controllers
{
   public class HomeController : Controller
   {
      // GET: Model
      public ActionResult Index()
      {
         return View();
      }

      public ActionResult FormularioVisita()
      {
         return View();
      }

      public ActionResult CargaDatos()
      {
         string nombre = Request.Form["nombre"].ToString();
         string comentarios = Request.Form["comentarios"].ToString();
```

```
            LibroVisitas libro = new LibroVisitas();
            libro.Grabar(nombre, comentarios);
            return View();
        }

        public ActionResult ListadoVisitas()
        {
            LibroVisitas libro = new LibroVisitas();
            string todo = libro.Leer();
            ViewData["libro"] = todo;
            return View();
        }
    }
}
```

En el método ListadoVisitas procedemos a crear un objeto de la clase LibroVisitas y recuperamos el contenido almacenado llamando al método Leer:

```
            LibroVisitas libro = new LibroVisitas();
            string todo = libro.Leer();
```

Mediante el diccionario ViewData podemos pasar datos a la vista, creamos una entrada en el diccionario y le asignamos el string recuperado desde el "Modelo":

```
            ViewData["libro"] = todo;
            return View();
```

Paso siguiente debemos crear la "Vista" para el método "ListadoVisitas", para ello presionamos el botón derecho del mouse y creamos la vista como hemos estado trabajando hasta ahora.

Modificamos el archivo "ListadoVisitas.cshtml" para que muestre todos los comentarios que se han dejado en el sitio web:

```
@{
    Layout = null;
}
```

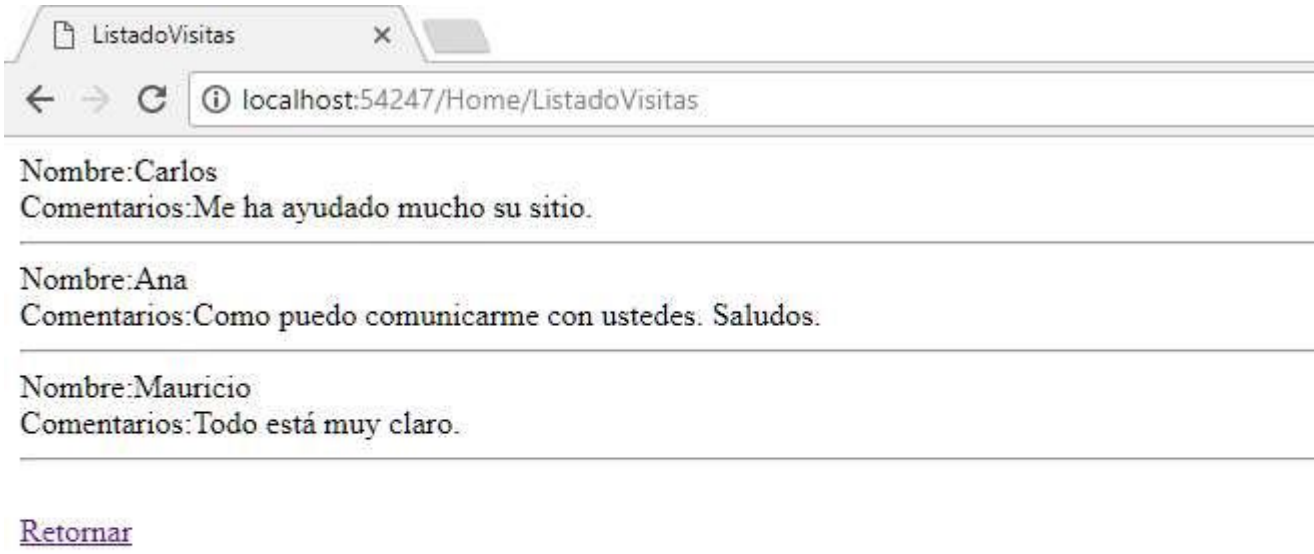
```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ListadoVisitas</title>
</head>
<body>
    <div>
        @Html.Raw(ViewData["libro"])
        <br />
        <a href="/">Retornar</a>
    </div>
</body>
</html>
```

Ahora mediante Razor podemos recuperar los datos almacenados en el diccionario ViewData que cargamos en el "Controlador", si bien podemos escribir:

```
    <div>
        @ViewData["libro"]
        <br />
        <a href="/">Retornar</a>
    </div>
```

Luego como tenemos marcas HTML en el archivo de texto no aparecen correctamente. Debemos utilizar el método Raw de HTML para su correcta visualización.

El resultado de esta vista es:



Ya tenemos correctamente funcionando nuestro pequeño sitio web que permite dejar comentarios y ver los comentarios de otros visitantes. Todo esto utilizando el patrón del Modelo Vista Controlador.

Es necesario practicar para sentir más natural esta forma de organizar el código de nuestro proyecto en distintas carpetas, con distintas clases que tienen asignadas distintas responsabilidades.