

**Tema**

TALLER WEB CON MVC DOTNET

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU.CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

14/12/2022

SOCKETS EN APACHE NETBEANS

[1. INTRODUCCIÓN 4](#_Toc121955073)

[2. OBJETIVOS 4](#_Toc121955074)

[2.1. OBJETIVO GENERAL 4](#_Toc121955075)

[2.2. OBJETIVOS ESPECÍFICOS 4](#_Toc121955076)

[3. MARCO TEÓRICO 4](#_Toc121955077)

[3.1 PATRÓN MODELO VISTA CONTROLADOR 4](#_Toc121955078)

[3.1.1 MODELO 5](#_Toc121955079)

[3.1.2 VISTA 5](#_Toc121955080)

[3.1.3 CONTROLADOR 5](#_Toc121955081)

[3.2 PLATAFORMA .NET 6](#_Toc121955082)

[3.2.1 MARCO DE DESARROLLO ASP.NET FRAMEWORK 6](#_Toc121955083)

[3.2.2 RAZOR 7](#_Toc121955084)

[3.2.3 VISUAL STUDIO 7](#_Toc121955085)

[4. PARTE PRÁCTICA 8](#_Toc121955086)

[4.1 CREACIÓN DEL PROYECTO 8](#_Toc121955087)

[4.2 ESTRUCTURA DE LA APLICACIÓN 24](#_Toc121955088)

[4.3 EJECUCIÓN DEL PROYECTO 25](#_Toc121955089)

[5. CONCLUSIONES 26](#_Toc121955090)

[6. RECOMENDACIONES 27](#_Toc121955091)

[7. REFERENCIAS 27](#_Toc121955092)

**INDICE DE IMÁGENES**

[Figura 1. Arquitectura MVC 5](#_Toc121955093)

[Figura 2. Logotipo identificativo de .NET 6](#_Toc121955094)

[Figura 3. Logotipo identificativo de ASP.NET 6](#_Toc121955095)

[Figura 4. Logotipo identificativo de Razor 7](#_Toc121955096)

[Figura 5. Logotipo identificativo de Visual Studio 8](#_Toc121955097)

[Figura 6. Ventana inicial Visual Studio 8](#_Toc121955098)

[Figura 7. Búsqueda de la plantilla para el proyecto 9](#_Toc121955099)

[Figura 8. Configuración de la plantilla del Proyecto 9](#_Toc121955100)

[Figura 9. Estructura inicial del Proyecto con MVC 10](#_Toc121955101)

[Figura 10. Estructura del proyecto para la práctica 10](#_Toc121955102)

[Figura 11. Creación de Clase en el Modelo 11](#_Toc121955103)

[Figura 12. Nombre para la Clase en el Modelo 12](#_Toc121955104)

[Figura 13. Codificación de la clase DatosEncuesta 12](#_Toc121955105)

[Figura 14. Codificación de la clase HomeController 15](#_Toc121955106)

[Figura 15. Codificación de vista \_Layout 17](#_Toc121955107)

[Figura 16. Codificación de la vista Index 17](#_Toc121955108)

[Figura 17. Añadir vista al controlador 20](#_Toc121955109)

[Figura 18. Selección del tipo de vista 20](#_Toc121955110)

[Figura 19. Propiedades para la vista Salida 21](#_Toc121955111)

[Figura 20. Codificación de la vista Salida 21](#_Toc121955112)

[Figura 21. Añadir nueva carpeta resources para imágenes 23](#_Toc121955113)

[Figura 22. Añadir imágenes a la carpeta resources 23](#_Toc121955114)

[Figura 23. Selección de imágenes para la carpeta resources 24](#_Toc121955115)

[Figura 24. Lista de imágenes en la carpeta resources 24](#_Toc121955116)

[Figura 25. Estructura final de la aplicación web con MVC 25](#_Toc121955117)

[Figura 26. Ejecución del proyecto 25](#_Toc121955118)

[Figura 27. Vista del formulario en el navegador 26](#_Toc121955119)

[Figura 28. Vista salida en el navegador 26](#_Toc121955120)

**ÍNDICE DE TABLAS**

[Tabla 1. Codificación del archivo DatosEncuesta.cs 13](file:///C:\FORMULARIOS_CHUQUIMARCA\PARCIAL2\04.%20WEB_CON_MVC_DOTNET_GRUPO5\DOCUMENTACION\08_MODELO_CONCEPTUAL_GRUPO5.docx#_Toc121955131)

[Tabla 2. Codificación del archivo HomeController.cs 16](file:///C:\FORMULARIOS_CHUQUIMARCA\PARCIAL2\04.%20WEB_CON_MVC_DOTNET_GRUPO5\DOCUMENTACION\08_MODELO_CONCEPTUAL_GRUPO5.docx#_Toc121955132)

[Tabla 3. Codificación del archivo Index.cshtml 18](file:///C:\FORMULARIOS_CHUQUIMARCA\PARCIAL2\04.%20WEB_CON_MVC_DOTNET_GRUPO5\DOCUMENTACION\08_MODELO_CONCEPTUAL_GRUPO5.docx#_Toc121955133)

[Tabla 4. Codificación del archivo Salida.cshtml 22](file:///C:\FORMULARIOS_CHUQUIMARCA\PARCIAL2\04.%20WEB_CON_MVC_DOTNET_GRUPO5\DOCUMENTACION\08_MODELO_CONCEPTUAL_GRUPO5.docx#_Toc121955134)

# INTRODUCCIÓN

las aplicaciones web deben estar desarrolladas bajo un patrón de arquitectura para que sea más sencillo corregir errores, implementar nuevas funcionalidades, realizar pruebas y mantenimiento de la aplicación. El patrón arquitectura más común es MVC (Modelo-Vista-Controlador), aunque existen otras variaciones más modernas como MVVM o MVP.

Las tecnologías y herramientas para el desarrollo de software disponen de plantillas de proyectos con una estructura y configuración inicial, por ejemplo, ASP.NET de Microsoft. El marco de ASP.NET MVC es un marco de presentación de poca complejidad que se integra con las características de ASP.NET existentes, tales como páginas maestras y la autenticación basada en pertenencia (Microsoft, 2022).

# OBJETIVOS

## OBJETIVO GENERAL

Analizar los conceptos y pasos para elaborar un formulario web implementado con el patrón de arquitectura MVC usando la plataforma NET de Microsoft y el marco de desarrollo ASP.NET Framework, para entender cómo se organiza y comunica una aplicación web con MVC.

## OBJETIVOS ESPECÍFICOS

* Analizar la arquitectura Modelo-Vista-Controlador en ASP.NET Framework para comprender como se debe organizar una aplicación web bajo esta arquitectura.
* Identificar los pasos para implementar un formulario web que registre información de un programador en ASP.NET Framework.
* Documentar todo el proceso de análisis y diseño utilizando el IDE (Entorno de desarrollo integrado) Visual Studio 2022.

# MARCO TEÓRICO

## PATRÓN MODELO VISTA CONTROLADOR

Es un patrón de arquitectura para el desarrollo de aplicaciones software donde su objetivo es el separar la lógica del negocio de la interfaz de usuario lo cual lo hace escalable facilitando la evolución por separado de sus partes, incrementa la reutilización y flexibilidad.

Tuvo su origen en 1979 y es también conocido como patrón MVC, divide una aplicación software en tres grandes partes bien diferenciadas las cuales son Modelo, Vista y Controlador.

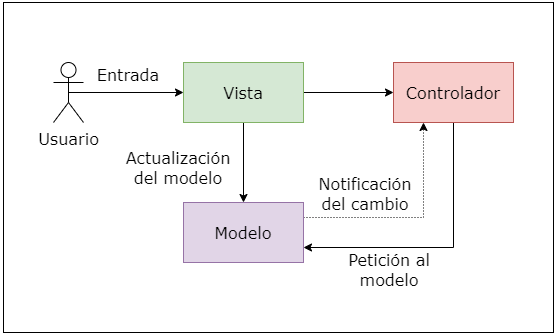


Figura . Arquitectura MVC

### **MODELO**

Es una capa donde se localiza la funcionalidad central y los datos, se comunica con el controlador y la base de datos.

Es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar otras librerías como PDO o algún ORM como Doctrine, que permite trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.

### **VISTA**

Es la capa donde se muestra la información al usuario donde es posible definir una o más vista de usuario en un software, se comunica con el controlador y el usuario.

Las vistas, como su nombre nos hacen entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más tenemos los códigos HTML y PHP que nos permite mostrar la salida.

En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generarán la salida, tal como nuestra aplicación requiera.

### **CONTROLADOR**

Es la capa donde se manejan las entradas del usuario se separa la representación interna de la información y la forma en la que se le muestra al usuario, se comunica con el modelo.

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad, es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

## PLATAFORMA .NET

Microsoft .NET es una plataforma para el desarrollo de software lanzada por Microsoft con la finalidad de fusionar su amplio catálogo de productos, que va desde sus múltiples sistemas operativos hasta herramientas de desarrollo (Pellicer, 2021). Se utiliza para crear aplicaciones para dispositivos web, móviles, escritorio, juegos y de IoT (Internet de las cosas). El marco .NET proporciona un modelo de programación coherente y un entorno de tiempo de ejecución que simplifica el desarrollo y la implementación de aplicaciones. También incluye una gran biblioteca de componentes prediseñados que se pueden usar para crear e implementar rápidamente aplicaciones con un código mínimo.



Figura . Logotipo identificativo de .NET

### **MARCO DE DESARROLLO ASP.NET FRAMEWORK**

ASP.NET es un marco de aplicación web desarrollado por Microsoft que permite a los desarrolladores crear aplicaciones y servicios web dinámicos (Carmona, 2022). Proporciona un marco sólido y escalable para crear aplicaciones y servicios web, con funciones como administración y autenticación de datos integradas, así como compatibilidad con el desarrollo de front-end y back-end.



Figura . Logotipo identificativo de ASP.NET

### **RAZOR**

Razor es una sintaxis de marcado para incrustar código basado en servidor en páginas web, es lo suficientemente inteligente para saber cuándo termina el código de servidor, sin necesidad de que lo explicitemos (Tomás, 2011). Se usa comúnmente en ASP.NET, el marco de desarrollo web de Microsoft, para crear sitios web dinámicos e interactivos. Razor usa una combinación de HTML y C# para crear el marcado final que se envía al navegador web del cliente, sus archivos se identifican por la extensión cshtml.

El motor Razor es el componente del marco ASP.NET que procesa el código Razor y genera el HTML final que se envía al navegador. Analiza el código de Razor y ejecuta cualquier bloque de código del lado del servidor, generando la salida resultante como parte de la página HTML. El motor Razor está diseñado para ser liviano, eficiente y fácil de usar, lo que lo convierte en una opción popular para crear aplicaciones web dinámicas con ASP.NET.



Figura . Logotipo identificativo de Razor

### **VISUAL STUDIO**

Visual Studio es un entorno de desarrollo integrado con el cual el desarrollador podrá crear y desarrollar softwares como aplicaciones web y móviles, sitios o servicios web en entornos compatibles con la plataforma .NET (Milagros, 2022). Visual Studio proporciona una interfaz fácil de usar y una gran cantidad de herramientas y características que facilitan a los desarrolladores la escritura, la depuración y la prueba del código.

Algunas de las características clave de Visual Studio incluyen:

* Soporte para una gran variedad de lenguajes de programación, incluidos C#, F#, VB.NET y C++.
* IntelliSense proporciona sugerencias y finalización de código inteligente para ayudar a los desarrolladores a escribir código de forma más rápida y precisa.
* Un editor de código integrado con funciones avanzadas como resaltado de sintaxis, refactorización de código y herramientas de depuración
* Integración con sistemas de control de fuente populares como Git, lo que permite a los desarrolladores administrar y realizar un seguimiento de los cambios en su base de código
* Compatibilidad con la creación e implementación de aplicaciones en una amplia gama de plataformas, incluidas Windows, macOS, Linux, iOS y Android
* Un amplio ecosistema de extensiones y herramientas de terceros que se pueden usar para personalizar y ampliar la funcionalidad de Visual Studio.



Figura . Logotipo identificativo de Visual Studio

# PARTE PRÁCTICA

Para esta práctica se implementará un formulario con los siguientes campos: Nombre, Apellido, Lenguajes de Programación que maneja, y una imagen en un archivo JSP. La información ingresada será enviada a una pestaña de salida JSP usando un controlador Servlet.

## CREACIÓN DEL PROYECTO

Para crear un nuevo proyecto abrir Visual Studio y clic en la opción “Create a new project”.

Text

Description automatically generated

Figura . Ventana inicial Visual Studio

Para esta práctica se debe buscar la plantilla del proyecto “ASP.NET Web Application (.NET Framework)”, seleccionar la plantilla y clic en Next.

Text

Description automatically generated

Figura . Búsqueda de la plantilla para el proyecto

En la siguiente ventana se muestra la configuración del proyecto, escoger la opción de MVC y clic en Create.

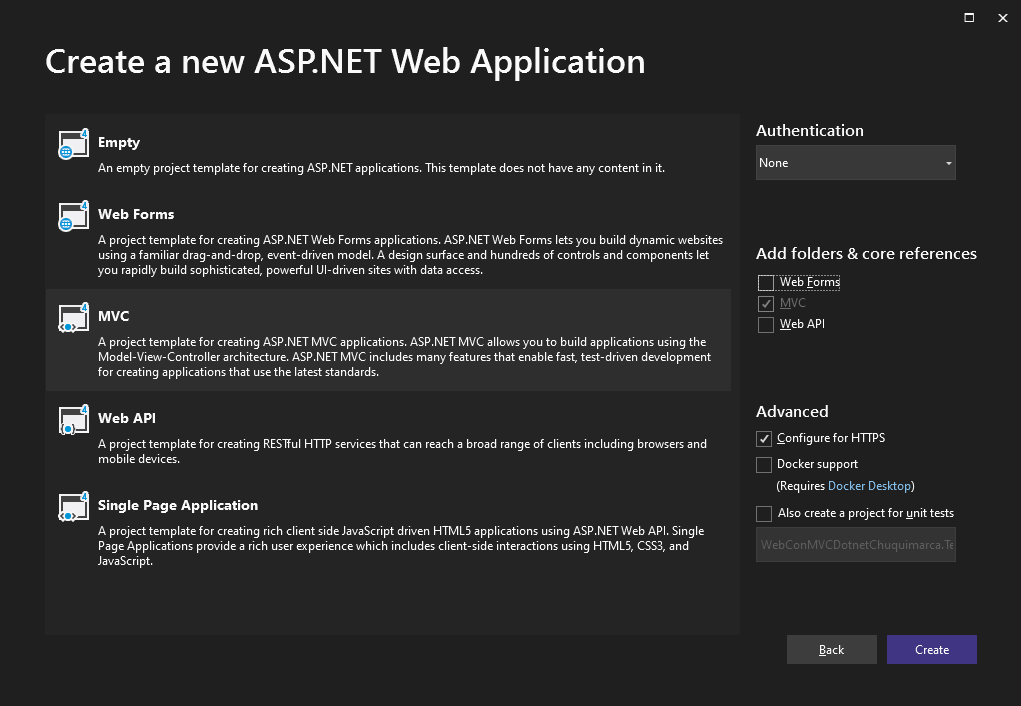


Figura . Configuración de la plantilla del Proyecto

Una vez creado el proyecto, se muestra una lista de carpetas y archivos para organizar y clasificar los elementos de nuestro proyecto, principalmente las carpetas de Views, Models y Controllers.

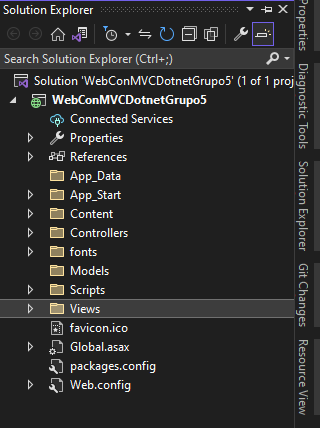


Figura . Estructura inicial del Proyecto con MVC

De forma predeterminada se crean algunos archivos para presentar una página de muestra inicial, borrar los archivos que no sean necesarios en las carpetas Views, Models y Controllers hasta que se tenga los que se presentan en la Figura 10.

Text

Description automatically generated

Figura . Estructura del proyecto para la práctica

Para guardar los datos que se ingresen y envíen desde el formulario se crea una clase llamada DatosEncuesta en la carpeta Models.

Text

Description automatically generated

Figura . Creación de Clase en el Modelo

Se verifica que el tipo de archivo sea Class C#, definir el nombre del archivo que será DatosEncuesta y clic en Next.

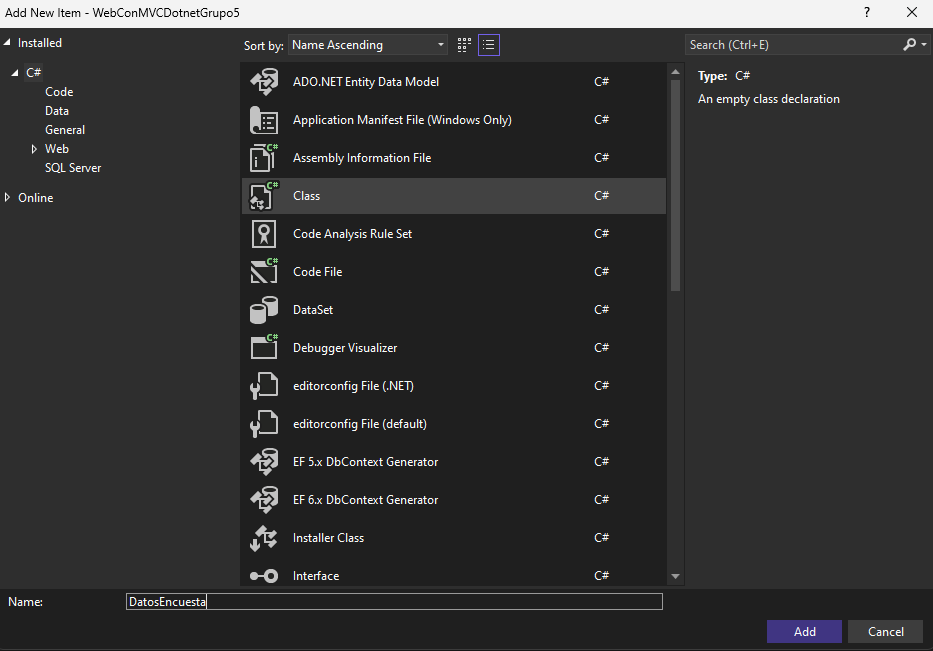


Figura . Nombre para la Clase en el Modelo

Una vez creada la clase se definen los campos de Nombre, Apellido, Lenguajes de programación e imagen con sus respectivos métodos, constructor y de acceso (Getters y Setters).

Text

Description automatically generated

Figura . Codificación de la clase DatosEncuesta

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace WebConMVCDotnetGrupo5.Models

{

public class DatosEncuesta

{

private string txtNombre;

private string txtApellido;

private string[] chkProgLeng;

private string imgProfile;

public string getTxtNombre()

{

return txtNombre;

}

public void setTxtNombre(string txtNombre)

{

this.txtNombre = txtNombre;

}

public string getTxtApellido()

{

return txtApellido;

}

public void setTxtApellido(string txtApellido)

{

this.txtApellido = txtApellido;

}

Tabla . Codificación del archivo DatosEncuesta.cs

public string[] getChkProgLeng()

{

return chkProgLeng;

}

public void setChkProgLeng(string[] chkProgLeng)

{

this.chkProgLeng = chkProgLeng;

}

public string getImgProfile()

{

return imgProfile;

}

public void setImgProfile(string imgProfile)

{

this.imgProfile = imgProfile;

}

}

}

En la clase HomeController que se creó por defecto en la carpeta Controllers se deben definir los Actions, estos son métodos en un controlador, accesibles por una URL gracias al sistema de enrutamiento y normalmente regresan otra acción con un resultado.

Se define un Action Salida en el controlador HomeController, este será la encargada de recuperar los datos del formulario y enviarlos a la vista Salida.

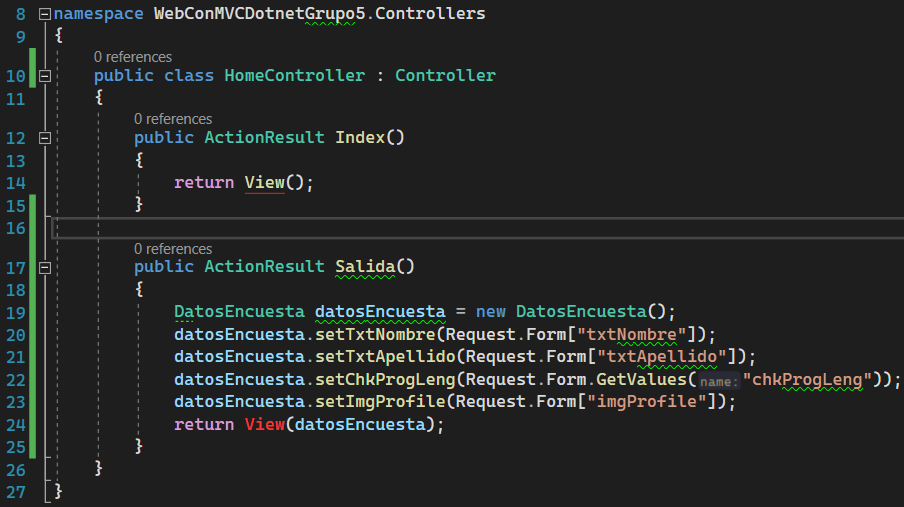


Figura . Codificación de la clase HomeController

Tabla . Codificación del archivo HomeController.cs

using System.Web.Mvc;

using WebConMVCDotnetGrupo5.Models;

namespace WebConMVCDotnetGrupo5.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

public ActionResult Salida()

{

DatosEncuesta datosEncuesta = new DatosEncuesta();

datosEncuesta.setTxtNombre(Request.Form["txtNombre"]);

datosEncuesta.setTxtApellido(Request.Form["txtApellido"]);

datosEncuesta.setChkProgLeng(Request.Form.GetValues("chkProgLeng"));

datosEncuesta.setImgProfile(Request.Form["imgProfile"]);

return View(datosEncuesta);

}

}

}

Para renderizar las vistas de Index y Salida se debe modificar el archivo \_Layout.cshtml como se muestra en la Figura 15 usando la sintaxis de Razor.

Text

Description automatically generated

Figura . Codificación de vista \_Layout

En la vista Index.cshtml se define la estructura del formulario, eliminar el contenido que viene por defecto y construir el formulario con las etiquetas HTML como se muestra en la Figura 15. En el campo action de la etiqueta form se define la ruta del controlador como Home/Salida, de esta forma se envía y recupera la información del formulario.

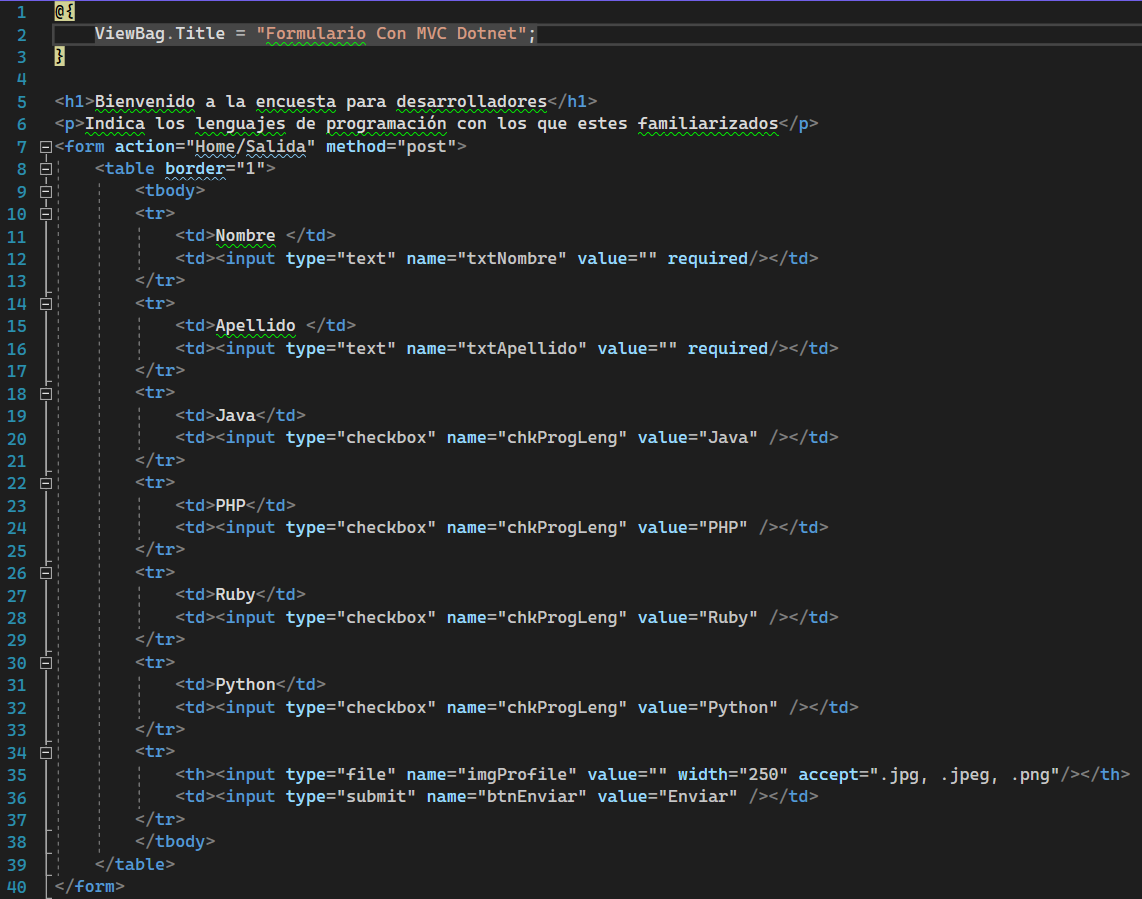


Figura . Codificación de la vista Index

Tabla . Codificación del archivo Index.cshtml

@{

ViewBag.Title = "Formulario Con MVC Dotnet";

}

<h1>Bienvenido a la encuesta para desarrolladores</h1>

<p>Indica los lenguajes de programación con los que estes familiarizados</p>

<form action="Home/Salida" method="post">

<table border="1">

<tbody>

<tr>

<td>Nombre </td>

<td><input type="text" name="txtNombre" value="" required/></td>

</tr>

<tr>

<td>Apellido </td>

<td><input type="text" name="txtApellido" value="" required/></td>

</tr>

<tr>

<td>Java</td>

<td><input type="checkbox" name="chkProgLeng" value="Java" /></td>

</tr>

<tr>

<td>PHP</td>

<td><input type="checkbox" name="chkProgLeng" value="PHP" /></td>

</tr>

<tr>

<td>Ruby</td>

<td><input type="checkbox" name="chkProgLeng" value="Ruby" /></td>

</tr>

<tr>

<td>Python</td>

<td><input type="checkbox" name="chkProgLeng" value="Python" /></td>

</tr>

<tr>

<th><input type="file" name="imgProfile" value="" width="250" accept=".jpg, .jpeg, .png"/></th>

<td><input type="submit" name="btnEnviar" value="Enviar" /></td>

</tr>

</tbody>

</table>

</form>

Una vez implementado el formulario, lo siguiente es definir la vista que mostrara la información enviada, para esto dirigirse a la clase HomeController, clic derecho sobre el Action Salida y clic en Add View.

A screenshot of a computer

Description automatically generated with medium confidence

Figura . Añadir vista al controlador

Verificar que el archivo sea de tipo MVC 5 View y clic en Add.

Graphical user interface

Description automatically generated

Figura . Selección del tipo de vista

A continuación, se muestra una ventana, ingresar los valores en los campos como se muestra en la Figura 19 y clic en Add.

Graphical user interface, text, application, email

Description automatically generated

Figura . Propiedades para la vista Salida

Para recuperar los datos en la vista Salida.cshtml se usa la notación @Model.nombreDelAtributo o método de recuperación (Getter), codificar el archivo como se muestra en la Figura 20 y guardar los cambios.

Graphical user interface, text

Description automatically generated

Figura . Codificación de la vista Salida

Tabla . Codificación del archivo Salida.cshtml

@model WebConMVCDotnetGrupo5.Models.DatosEncuesta

@{

Layout = null;

ViewBag.Title = "Datos de la encuesta";

}

<h2>Gracias por llenar la encuesta</h2>

<p>@Model.getTxtNombre() @Model.getTxtApellido() , has respondido que estas familiarizado con los siguientes lenguajes de programación:</p>

<img src="~/resources/@Model.getImgProfile()" name="datosEncuesta" property="imgProfile" width="200" height="250" alt="Archivo no seleccionado"/>

<ul>

@if (@Model.getChkProgLeng() != null)

{

for (int i = 0; i < @Model.getChkProgLeng().Length; i++)

{

<li>

@Model.getChkProgLeng()[i]

</li>

}

}

</ul>

En el formulario se define un campo para cargar imágenes, para añadir imágenes locales en este campo se debe crear una carpeta llamada resources, aquí se guardarán todas las imágenes que se usarán en el campo imagen.

Para crear una nueva carpeta hacer clic derecho sobre el proyecto, seleccionar la opción Add y clic en New Folder.

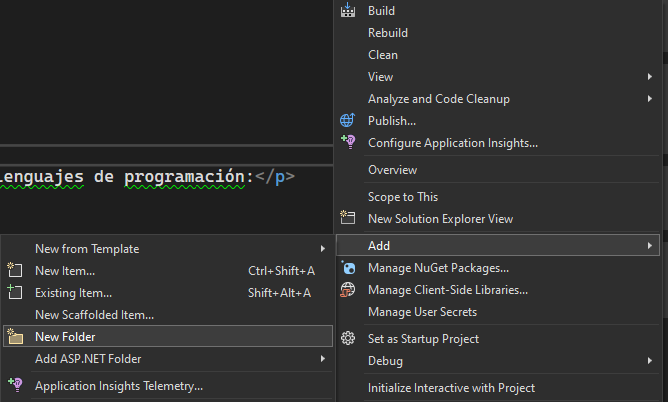


Figura . Añadir nueva carpeta resources para imágenes

Cambiar el nombre de la carpeta a resources, dar clic derecho sobre la carpeta creada y seleccionar la opción Add y dar clic en “Exiting Item…” para añadir las imágenes al proyecto.

A screenshot of a computer

Description automatically generated with medium confidence

Figura . Añadir imágenes a la carpeta resources

En la siguiente ventana dirigirse a la carpeta creada resources y pegar las imágenes que se agregaran al proyecto, seleccionar todas con Ctrl + A y clic en Add.

A screenshot of a computer screen

Description automatically generated with medium confidence

Figura . Selección de imágenes para la carpeta resources

Para verificar que las imágenes se agregaron con éxito al proyecto, clic sobre la carpeta resources y verificar que las imágenes aparezcan en la carpeta.

Graphical user interface, text, application

Description automatically generated

Figura . Lista de imágenes en la carpeta resources

Una vez realizados todos los pasos anteriores guardar todos los cambios y verificar que Visual Studio no muestre ninguna notificación de error.

## ESTRUCTURA DE LA APLICACIÓN

Text

Description automatically generated with medium confidence

Figura . Estructura final de la aplicación web con MVC

## EJECUCIÓN DEL PROYECTO

Para ejecutar el proyecto dirigirse a la barra de herramientas superior y dar clic en ISS Express (Microsoft Edge).

A computer screen capture

Description automatically generated with medium confidence

Figura . Ejecución del proyecto

A continuación, se abre una pestaña en el navegador con el formulario que se implemento en la vista Index.cshtml.

Graphical user interface, application

Description automatically generated

Figura . Vista del formulario en el navegador

Para comprobar que la aplicación funciona correctamente, llenar los campos del formulario y dar clic en el botón Enviar.

Por último, se muestra con éxito la vista Salida.cshtml con los datos enviados desde el formulario de Index.cshtml, todo este proceso utilizando la arquitectura MVC.

Graphical user interface, application

Description automatically generated

Figura . Vista salida en el navegador

# CONCLUSIONES

* El patrón de arquitectura MVC permite estructurar aplicaciones de software para que sea más fácil y rápido implementar nuevas características, realizar pruebas, reutilizar y mantener el código del proyecto.
* Dotnet conocido actualmente como .NET es una plataforma de desarrollo de software propiedad de Microsoft, permite no solo construir aplicaciones web, sino todo tipo de aplicaciones para casi todas las plataformas existentes. Incluye plantillas con configuraciones y estructuras como MVC para crear aplicaciones de forma rápida.
* Al igual que Java EE, .NET dispone de un marco de desarrollo para aplicaciones web conocido como ASP.NET, entre sus principales características esta el poder incrustar código C# con Razor, tal y como ocurre con JSP, pero presenta una mayor facilidad y legibilidad para definir los Actions del controlador a diferencia de los Servlets de Java.

# RECOMENDACIONES

* Al crear un nuevo proyecto Web se presenta una gran cantidad de opciones, se recomienda indagar para entender las diferentes herramientas que dispone .NET al crear un proyecto nuevo.
* No crear proyectos de .NET en carpetas que contengan una ruta muy grande o se encuentren almacenados en varias subcarpetas, pues se presenta un error al ejecutar el aplicativo con el archivo System.Runtime.InteropServices.RuntimeInformation.dll.
* La base de .NET es C#, por lo tanto, es muy recomendable seguir un curso para comprender la sintaxis y características especificas de este lenguaje de programación de alto nivel.

# REFERENCIAS

* Carmona, J. (18 de Marzo de 2022). *Qué es ASP.NET*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-aspnet-y-cuales-son-sus-puntos-fuertes/#:~:text=ASP.NET%20es%20un%20framework,aplicaciones%20y%20servicios%20web%20modernos.
* Microsoft. (21 de Septiembre de 2022). *Información general sobre ASP.NET MVC*. Obtenido de ASP.NET Microsoft: https://learn.microsoft.com/es-es/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview
* Milagros, B. (20 de Marzo de 2022). *¿Qué es Visual Studio? ¡El desarrollo de software nunca fue más fácil!* Obtenido de crehana: https://www.crehana.com/blog/transformacion-digital/que-es-visual-studio/
* Pagoada, A. (2015). *Curso de ASP NET MVC en C# Creando el primer controlador y vista #1*. Obtenido de YouTube: https://www.youtube.com/watch?v=f835Kwe9XHU
* Pellicer, P. (21 de Octubre de 2021). *¿Qué es el .NET? ¿Para qué sirve?* Obtenido de emagister: https://www.emagister.com/blog/que-es-el-net-para-que-sirve/
* Tomás, E. (03 de Mayo de 2011). *El motor de vistas Razor*. Obtenido de desarrolloweb: https://desarrolloweb.com/articulos/motor-vistas-razor-dotnet.html