Parte 1

ASP. NET. RAZOR PAGES. TU PRIMER PROYECTO

Introducción
ASP.NET - Razor Pages
Razor Pages
Modelos
Depuración y limpieza
Glosario

1

INTRODUCCIÓN

El objetivo de esta colección es introducirte en uno de los ecosistemas y entornos de trabajo más utilizados y extensos del mundo del desarrollo, ASP.NET. Fue creado por Microsoft, es utilizado a diario por millones de desarrolladores, y demandado por miles de empresas y organizaciones en el mundo, hecho que lo convierte en una de las herramientas más importantes del rubro.

1.1 HERRAMIENTAS

ASP.NET permite el desarrollo de distintos tipos de aplicaciones en la Web, desde sitios con pequeñas implementaciones, hasta sistemas a mayor escala, preparados para soportar grandes cantidades de trabajo y de operaciones a la vez. En esta colección verás las diferentes herramientas que ofrece este ecosistema, que varían según las posibles alternativas de desarrollo.

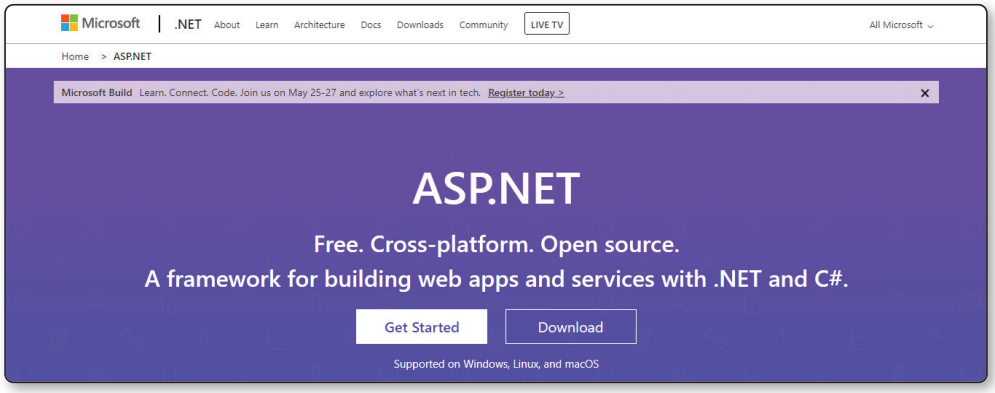


Figura 1.1. ASP.NET es una plataforma para crear sitios y aplicaciones soportadas en distintos sistemas.

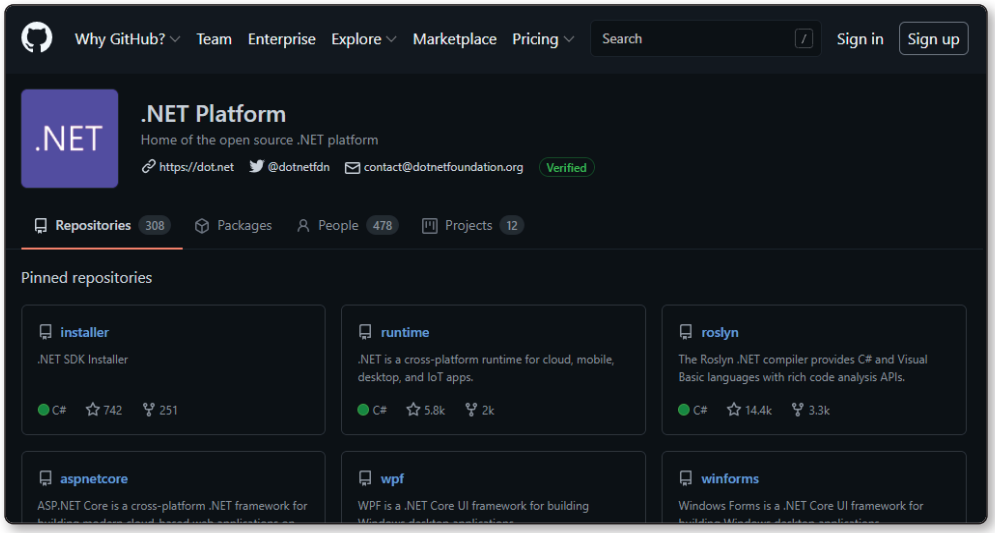


Figura 1.2. GitHub se emplea como gestor de repositorios para proyectos enormes, como .NET.

Comenzar a estudiar y desarrollar aplicaciones en ASP.NET tiene muy pocos requisitos. Además de los conocimientos previos que necesitas, solo precisas una computadora, con sistema operativo Windows o Mac, acceso a Internet y un editor de código.

Aunque puedes trabajar con cualquier editor de código o entorno de desarrollo integrado, en esta obra se utiliza Visual Studio Community. Este es un IDE creado por Microsoft, con una integración completa con la plataforma de desarrollo ASP.NET. Es gratuito, en la versión mencionada, y permite desarrollar de forma cómoda, con herramientas como detección de errores en la sintaxis, andamiaje, herramientas de testeo y de lanzamiento de aplicaciones, gestión de bases de datos e, incluso, integración con versionados de código.

1.2 ¿QUÉ DEBO SABER?

Para comenzar a trabajar en ASP.NET, debes tener conocimientos sobre C#, y aunque no es completamente necesario, sí es recomendable manejar el lenguaje de acceso a datos SQL. También se insta al lector a conocer, al menos, las bases de HTML, CSS y JavaScript.

1.2.1 C#

C# es un lenguaje de programación multiparadigma, desarrollado por Microsoft y lanzado en 2000, con orientación a la programación orientada a objetos, aunque cuenta con distintas características que lo vuelven un lenguaje muy moderno, como la programación asíncrona. Su sintaxis básica está inspirada en la de C y C++, por lo cual a aquellos programadores que conozcan estos lenguajes les resultará muy fácil su aprendizaje y uso.

ASP.NET utiliza C# como uno de sus lenguajes principales, y permite la creación de programas para distintas plataformas, como Windows, UNIX, Android e iOS, entre otros.

1.2.2 SQL

El lenguaje SQL es una sintaxis utilizada y modificada por muchos gestores de bases de datos. Se lo considera un estándar, aprovechado por distintos motores de gestión de la información, y del cual derivan muchísimos dialectos con pequeñas modificaciones. Tales son los casos de SQL Server, Oracle y MySQL, entre otros.

SQL permite el acceso y la manipulación de datos en bases de datos relacionales, que emplean tablas como soporte para la información.

1.2.3 HTML, CSS y JavaScript

HTML es el lenguaje de marcado estándar, actualizado y mantenido por el W3 Consortium, que se utiliza para la creación de páginas web y es soportado por todos los navegadores modernos. Permite almacenar y estructurar la información de modo jerárquico, en archivos con extensión .HTML.

CSS es un lenguaje de hojas de estilo creado para interactuar con HTML y modificar la manera en la cual los navegadores muestran la información de los archivos HTML. Se encarga de estilizar la información almacenada en el lenguaje de marcado HTML, para separar de modo organizado la información de los estilos y el diseño. JavaScript es el lenguaje estándar de programación de los navegadores web. Es un dialecto o derivado del estándar ECMAScript, el cual fue desarrollado en 1996, con inspiración en lenguajes como Java y C.

Este lenguaje es un estándar dentro de los navegadores porque todos poseen un motor de intérprete para él, que les permite ejecutar las instrucciones de los programas creados en JavaScript.

1.3 ¿DÓNDE APRENDER TODO ESTO?

El punto de partida en el cual te basarás para adquirir o reforzar conocimientos en uno o más de los temas mencionados es el sitio web RedUSERS Premium: **https://premium.redusers.com**. Con la experiencia y las décadas de evolución del ecosistema de la información, RedUSERS Premium cuenta con manuales, e-books y guías necesarias para potenciar tu conocimiento básico y así entender a la perfección esta obra.

1.3.1 C#

RedUsers Premium te ofrece la posibilidad de aprender C# de forma completa, en su obra **C# Guía total del programador**. Podrás ver conceptos iniciales del lenguaje, desde la forma en la que se declaran variables y la manera de realizar programas básicos; hasta conceptos mucho más avanzados, como la programación orientada a objetos.

Si te sientes listo para iniciarte en temas mucho más complejos del lenguaje, puedes continuar con **C# Avanzado**, una obra orientada a aquellas personas que deseen ver a fondo todas las características de C#.

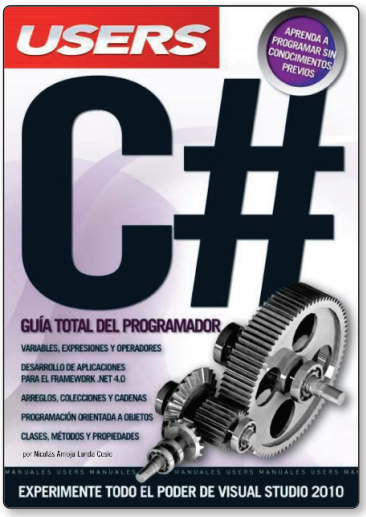


Figura 1.3. C# La guía totaldel programador.

Figura 1.4. C# Avanzado.

1.3.2 HTML, CSS y JavaScript

Puedes aprender sobre HTML en alguna de las obras que RedUSERS Premium te ofrece:

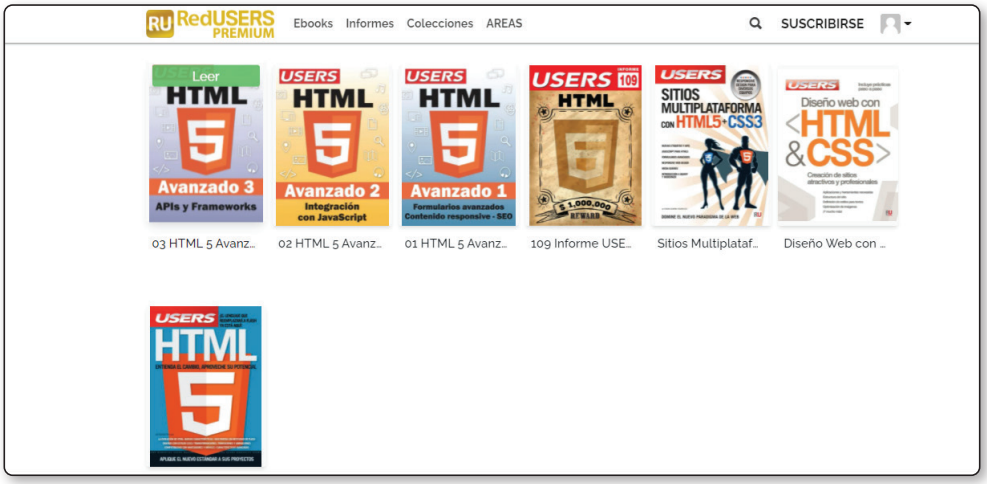


Figura 1.5. *HTML* en RedUSERS.

CSS es un lenguaje orientado al estilo y el diseño, y puedes aprenderlo en conjunto con HTML en RedUSERS Premium:

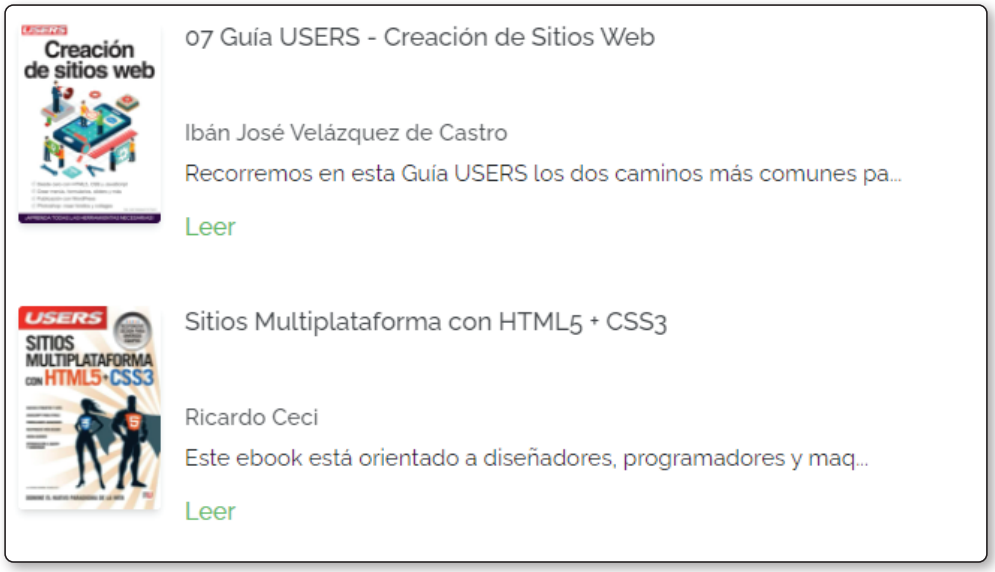


Figura 1.6. *CSS* en RedUSERS.

El lenguaje de programación de la Web es uno de los más demandados en la actualidad, y RedUSERS tiene varias entregas que te serán útiles para aprenderlo, tanto si ya tienes conocimientos como si partes desde cero.

Figura 1.7. *JavaScript* en RedUSERS.

1.3.3 SQL

Si deseas comenzar a estudiar sobre SQL y cómo implementarlo, puedes aprovechar la obra Programador Web Full Stack, en su entrega 14, donde se ve MySQL.

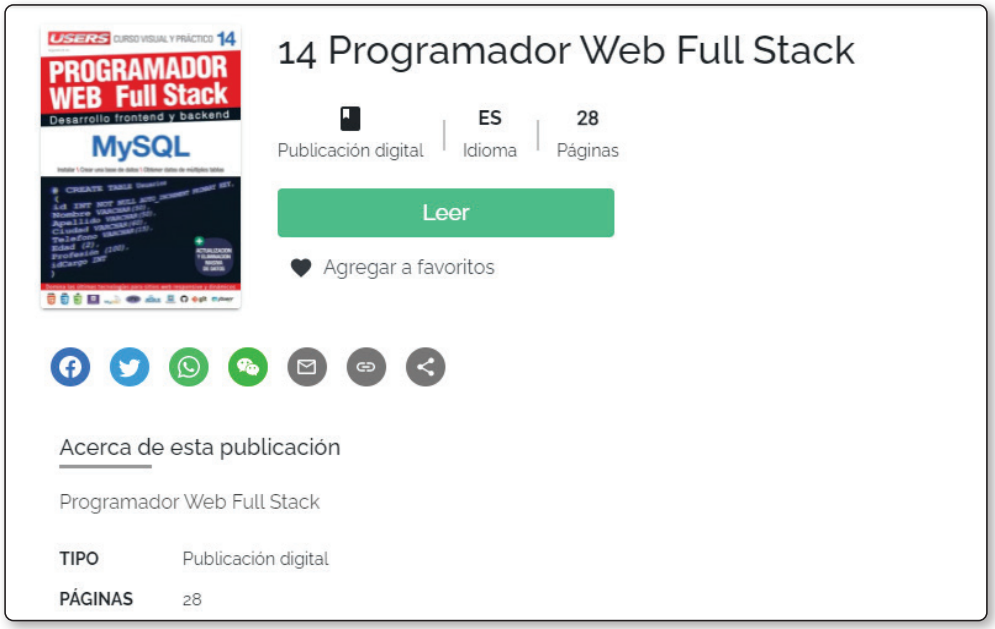


Figura 1.8. *MySQL* en RedUSERS.

1.4 ACTIVIDADES

A continuación se presentan las preguntas que deberías saber responder para considerar aprendido el capítulo.

1.4.1 Test de autoevaluación

1. ¿Para qué se puede utilizar ASP.NET?

2. ¿Qué empresa lo desarrolló y mantiene?

3. ¿Qué tecnologías usa?

4. ¿Qué lenguajes se recomiendan conocer para estudiarlo?

5. ¿Qué lenguaje principalmente utiliza ASP.NET?

6. ¿Qué paradigmas conoces que soporte C#?

2

ASP.NET – RAZOR PAGES

En este capítulo se hará una introducción al ecosistema de desarrollo de aplicaciones para la Web de Microsoft, ASP.NET, un entorno de trabajo que ha sido la apuesta, durante años, de una de las empresas más importantes del mundo del software.

2.1 CONCEPTOS IMPORTANTES

Desde el lanzamiento del sistema operativo Windows, Microsoft se ha convertido en una de las empresas de software más relevantes del mundo. Y además de destacarse por sus sistemas operativos y programas, ha sido clave por tener un ecosistema de desarrollo de aplicaciones y sistemas web muy interesante y en constante avance.

El mundo del software siempre está cambiando, y muchas empresas intentan destacarse con distintas alternativas, lenguajes de programación, **frameworks**, librerías o entornos de desarrollo completos. Ante el avance de ecosistemas muy populares, como Java (lanzado por Oracle) o Android (creado por Google), por mencionar solo algunos, Microsoft se ha planteado combatir en el ámbito del desarrollo de aplicaciones web con .NET, y para ser más precisos, con ASP.NET.

ASP.NET es un entorno que permite a los desarrolladores crear todo tipo de sistemas con orientación a la Web dentro del framework .NET. Este es un entorno de trabajo de código abierto creado por Microsoft para el desarrollo de distintos tipos de aplicaciones.

Para comenzar a trabajar con esta tecnología, se recomienda tener conocimientos sobre el lenguaje de programación **C#** o sobre **Visual Basic**, además de conocer sobre bases de datos y, al menos, manejar los conceptos básicos sobre el lenguaje universal de acceso a datos **SQL**. Si deseas aprender los lenguajes de programación C# y Visual Basic, y la manera de trabajar con ellos dentro del **framework** .NET, puedes leer la entrega 3 de Desarrollador .NET en este enlace:

https://premium.redusers.com/reader/03-desarrollador-net.

ASP.NET está pensado para la programación, desde sitios pequeños, hasta aplicaciones a gran escala. Se lanzó al mercado en 2002 junto con el **framework** .NET, como sucesor de **Active Server Pages** o **ASP Classic**.

Por lo general, se lo utiliza con Visual Basic .NET, sucesor de Visual Basic, y C#, aunque también presenta soporte para otros lenguajes de manera nativa y en formato interoperabilidad (**Figura 2.1.**).

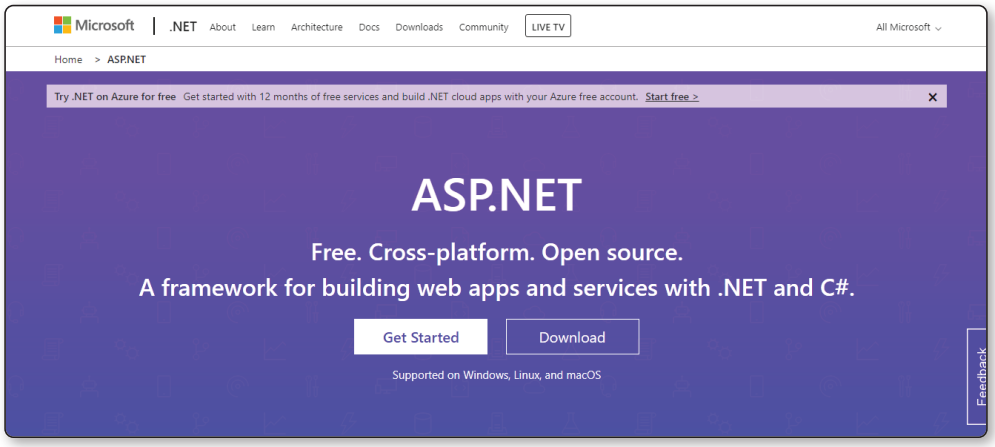


Figura 2.1. ASP.NET es la tecnología de Microsoft para desarrollo en la Web.

Una de las principales cuestiones para tener en cuenta al empezar a aprender una nueva tecnología, ya sea un lenguaje de programación o un entorno completo, es su uso a nivel global, es decir, su demanda dentro del mercado. En la actualidad, C# es el quinto lenguaje de programación más usado de acuerdo con el índice **TIOBE**, que marca la popularidad y la demanda de los lenguajes de programación, además de estar en constante crecimiento.

ASP.NET permite la creación de todo tipo de sistemas web, desde aplicaciones sencillas, hasta **APIs**, sistemas complejos y a gran escala. Razor Pages es una alternativa muy interesante dentro de este ecosistema, ya que permite desarrollar páginas web de manera sencilla y rápida, con posibilidad de interactuar con código del servidor, lo que genera sitios dinámicos y, a la vez, muy escalables.

Razor representa una de las cuatro formas de proyectos que ofrece ASP.NET, acompañado por ASP.NET MVC, ASP.NET Blazor y ASP.NET API. Cada uno de ellos está orientado a un tipo de arquitectura en particular, pero al mismo tiempo, pueden coexistir dentro de un mismo sitio o sistema. Las cuatro arquitecturas utilizan el framework ASP.NET Core, un entorno de trabajo que permite desarrollar aplicaciones multiplataforma; esto quiere decir que con cualquiera de las cuatro arquitecturas puedes crear aplicaciones que corran en entornos Windows, Mac o, también, Linux, mediante el SDK .NET.

El kit de desarrollo o SDK .NET es un conjunto de librerías necesarias para compilar y ejecutar las aplicaciones creadas en .NET; podrás instalarlo siguiendo los pasos que se indican en esta obra. Este kit te permitirá desarrollar bajo un IDE como Visual Studio o con cualquier otro editor de texto, utilizando una terminal y los comandos que proporciona, conocido como **CLI** o **Command Line Interface** de .NET.

Años atrás, la forma más popular de crear sitios web sencilla y rápidamente, bajo el framework .NET, era utilizando **Web Forms**. Razor Pages es una opción más moderna y completa que viene a suplantar a la anterior, ya en desuso y que Microsoft no actualizará; solo dará soporte de seguridad, debido a su condición de obsoleta. Razor es la parte de ASP.NET que permite trabajar de forma cómoda con páginas web simples y rápidas, cuando se necesita crear un sitio poderoso, pero a la vez, sencillo; esa es su principal función.

Para usar Razor Pages en ASP.NET, es necesario contar con conocimientos, aunque sean básicos, de C#, y tener el IDE de Microsoft Visual Studio en su versión **community**, que puede descargarse sin costo desde su página oficial, en *https://visualstudio.microsoft.com/es*. Además, precisas el SDK .NET 5.0, una herramienta que se utiliza para la creación y prueba de proyectos generados bajo el **framework** .NET. Puedes descargarlo desde el siguiente link: *https://dotnet.microsoft.com/download/dotnet/5.0*.

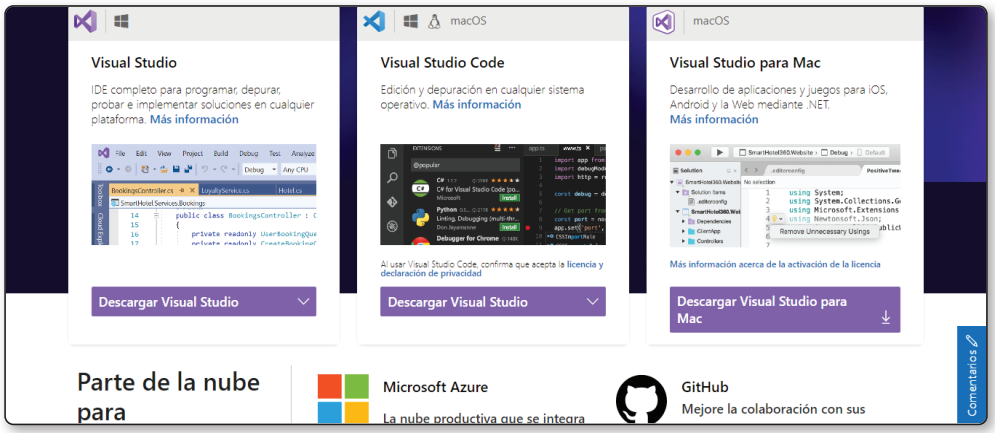


Figura 2.2. Visual Studio puede descargarse desde la página oficial de Microsoft.

Para instalar Visual Studio, solo es necesario ingresar en la página correspondiente y descargar su versión community. Este IDE es reconocido por ser uno de los entornos de desarrollo más completos del mercado, con la posibilidad de usarlo con herramientas como **Azure**, **GitHub**, **Git**, **Unity**, y soporte para lenguajes como **Python**, **C#**, **F#**, **R**, **JavaScript**, entre otras herramientas muy interesantes. Se encuentra disponible tanto para Windows como para Mac OS. Como desventaja, puede mencionarse que su consumo de recursos es más elevado que si se usaran otras alternativas.

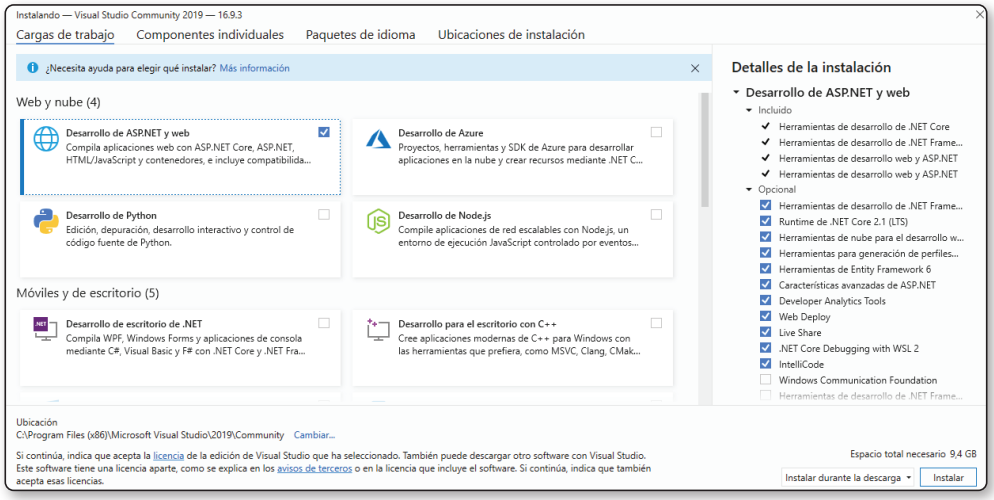
Dentro del mismo ecosistema se encuentra Visual Studio Code, un editor de texto que Microsoft lanzó como opción ligera, mucho más portátil y versátil. Brinda la posibilidad de instalar distintas herramientas, conocidas como **plugins**, que el desarrollador puede agregar o quitar, lo que evita gastar en recursos innecesarios. Visual Studio Code representa una alternativa compacta para aquellos que tengan computadoras menos poderosas o para quienes ya usen este editor con soporte para muchísimos lenguajes. Es gratuito y está disponible para Windows, Mac OS y Linux.

2.2 INSTALACIÓN

Para comenzar a trabajar con ASP.NET, solo necesitas dos herramientas básicas: Visual Studio en su versión community y el SDK .NET; si luego lo deseas, puedes instalar más complementos que te ayudarán a desarrollar aplicaciones más complejas.

PASO 1

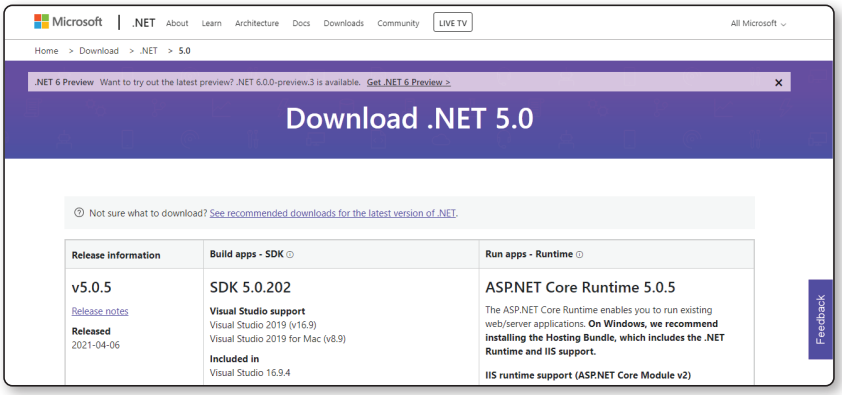
Comienza por descargar el IDE desde su página oficial y ejecuta el archivo que el sitio provee; aparecerá la pantalla que ves en la imagen. La instalación de Visual Studio te permitirá elegir entre distintas herramientas de desarrollo muy variadas.



PASO 2

Para trabajar con ASP.NET en Visual Studio, no es necesario tener instalados todos los paquetes que el asistente de la instalación ofrece al usuario; basta con tener activada la casilla de Desarrollo de ASP.NET y Web, como muestra la imagen.

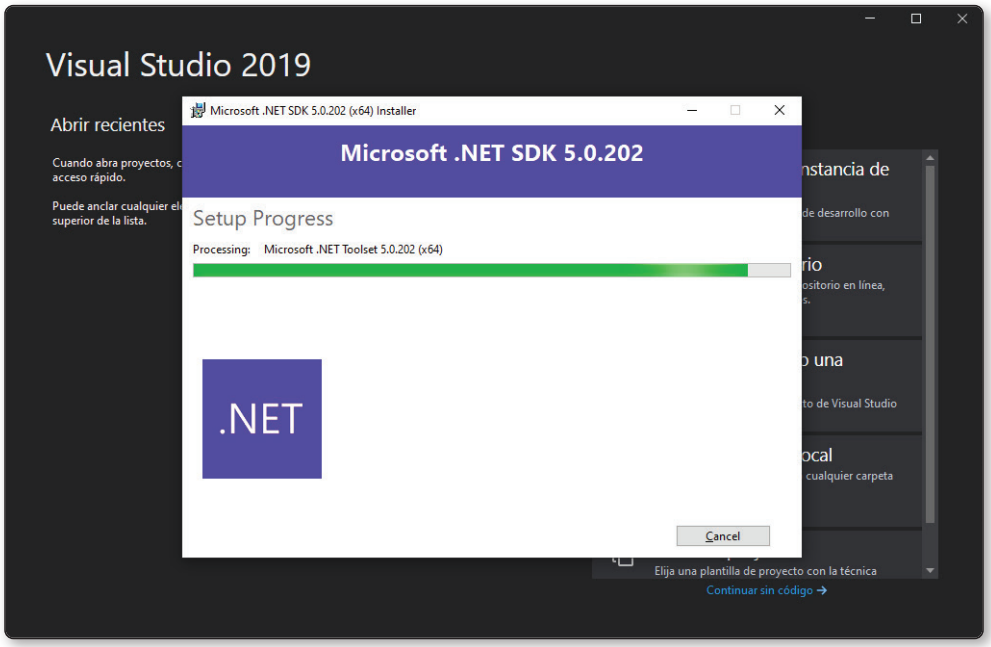
Una vez instalado Visual Studio, la herramienta de preferencia para esta entrega, debes instalar el SDK .NET, actualmente en su versión 5.0. Puedes descargarlo desde *https://dotnet.microsoft.com/download/dotnet/5.0*.



PASO 3

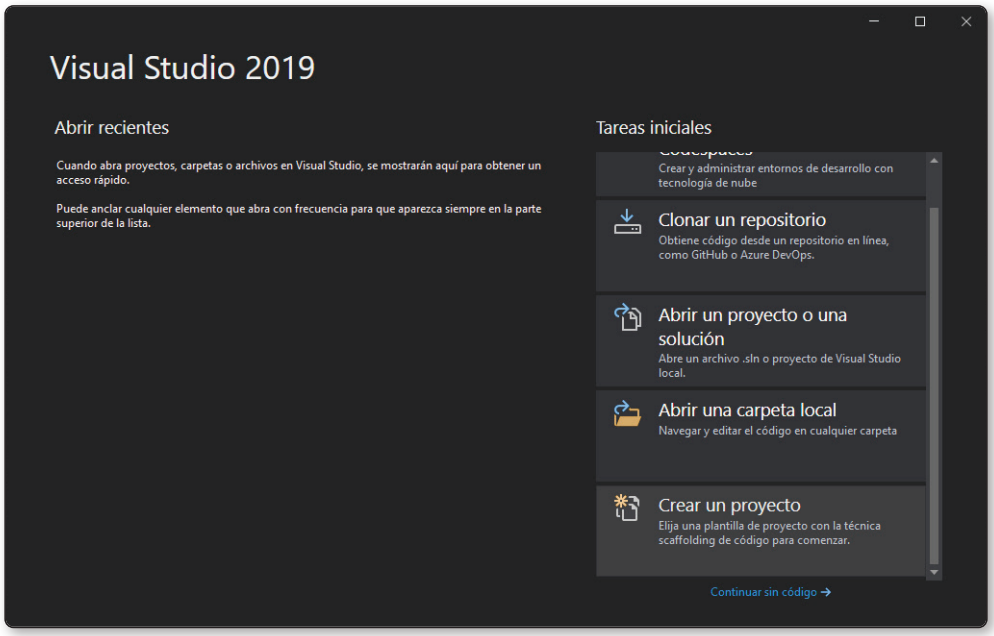
La instalación es bastante sencilla y, en pocos pasos, permite iniciar rápidamente proyectos ASP.NET de todo tipo. Una vez instaladas ambas herramientas, inicia Visual Studio para comenzar con el primer proyecto de Razor Pages.

El SDK .NET se instala fácilmente ejecutándolo luego de descargarlo, sin ningún requerimiento extra.



PASO 4

Para realizar este procedimiento, el IDE de Microsoft te guiará por una serie de pasos. En primer lugar, una pantalla de bienvenida te preguntará si deseas crear un proyecto nuevo o trabajar con otras opciones, como abrir un proyecto existente, trabajar con repositorios, y más. Para comenzar desde cero, selecciona **Crear un proyecto**.

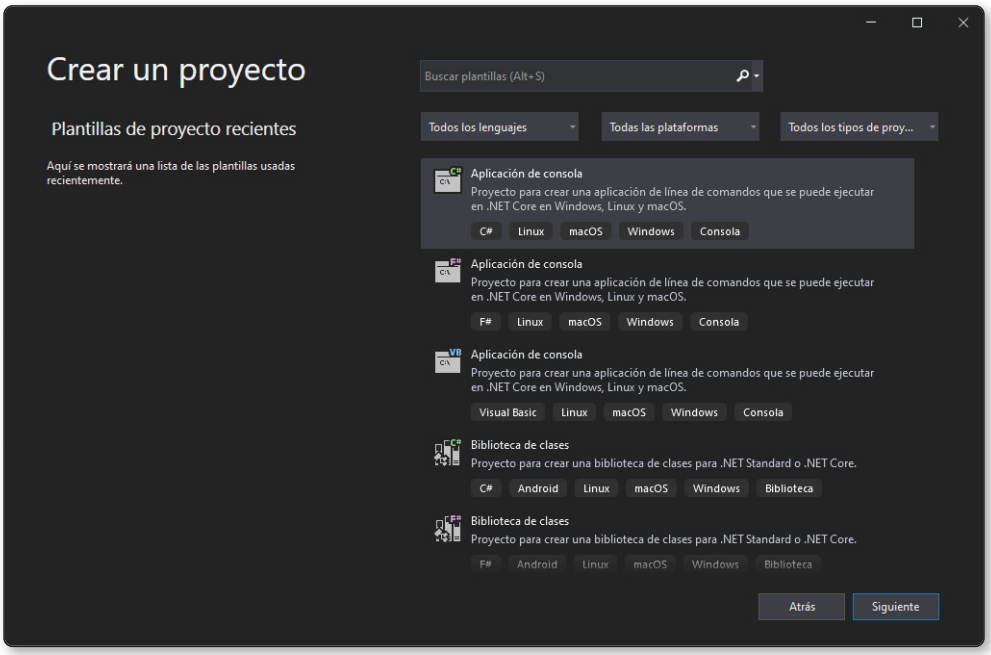


PASO 5

Para este primer proyecto de Razor, tendrás que abrir Visual Studio y seguir algunos pasos.

Apenas se inicie, se abrirá un asistente que te preguntará qué deseas hacer: **Conectarse a instancias remotas**, **Clonar un Repositorio**, **Abrir un proyecto o solución**, **Abrir una carpeta local** o **Crear un proyecto**. Selecciona **Crear un proyecto**. En la siguiente pantalla, elige el tipo de proyecto con el que vas a trabajar. Para crear un proyecto de Razor, es necesario seleccionar **Aplicación Web de ASP. NET Core**.

Como se explicó antes, un mismo proyecto puede coexistir con distintas arquitecturas, como MVC, API o Razor, sin ningún problema, dado que la misma aplicación puede ser modular y contener partes distintas. Por el momento, para crear un proyecto y trabajar con Razor, selecciona esta opción y presiona en **Siguiente**.



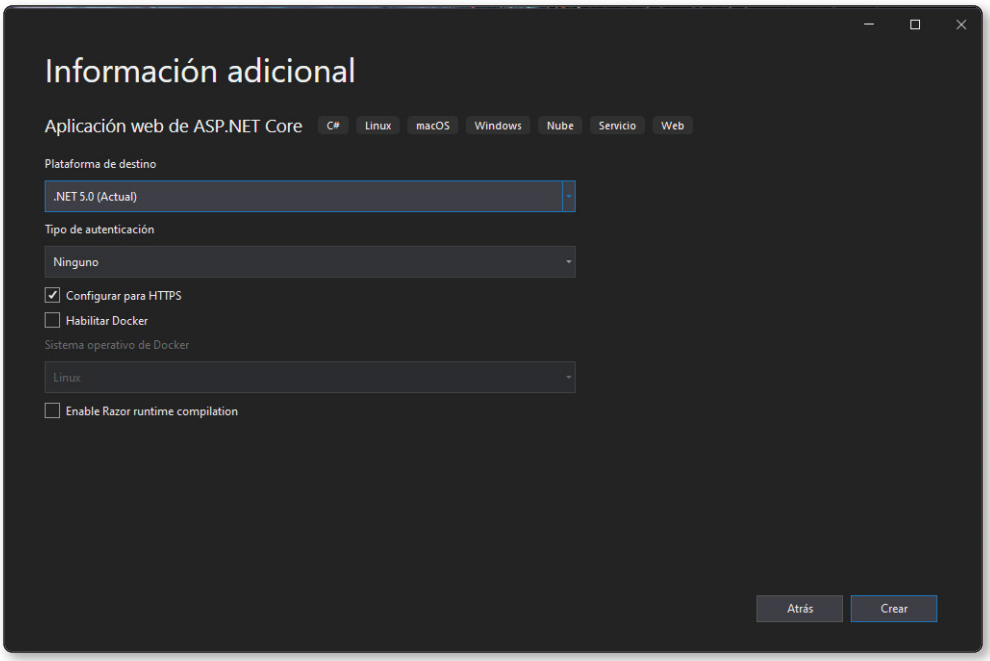
PASO 6

Antes de crear el proyecto, Visual Studio te solicitará que elijas el SDK correspondiente a la versión que tengas instalada.

El IDE te permitirá elegir entre distintas versiones del SDK, así como también si quieres crear el repositorio junto con la solución en la misma ubicación, algo que te ahorrará mucho tiempo de configuración de cara al futuro.

También te preguntará si deseas instalar el certificado HTTPS, que te permitirá trabajar con conexiones seguras y encriptadas desde que comiences a desarrollar.

Esto es algo que no está presente en muchos entornos de desarrollo, de modo que presenta una ventaja muy interesante para ASP.NET.



Si has instalado la última versión del framework .NET, que al momento de escribir este libro es la 5.0, selecciónala en la lista y luego presiona en **Crear**. Dentro de esta última ventana, el IDE ofrece una configuración por defecto bastante interesante y útil para tus primeros proyectos. Además de la versión del SDK, puedes elegir el tipo de autenticación, lo que será útil para proyectos avanzados en los cuales necesites seguridad a la hora de manipular usuarios. La configuración para HTTPS viene activa por defecto, y permite que el servidor de pruebas se ejecute con SSL, algo práctico para manipular datos sensibles. El IDE también tiene soporte para **Docker**, una plataforma destinada a empaquetar aplicaciones para que su despliegue sea mucho más sencillo. Por último, la opción **Razor runtime compilation** te permitirá guardar cambios en tu aplicación a medida que esta se ejecuta, para no detener el programa y volver a lanzarlo para ver los cambios.

Luego de unos momentos, Visual Studio creará un proyecto web y, dentro del editor, podrás ver las distintas carpetas y ficheros que este posee.

2.3 ESTRUCTURA DE UN PROYECTO

Siempre que se comienza a trabajar con una nueva tecnología, sobre todo dentro del mundo del desarrollo, es una buena práctica empezar viendo las diferentes partes que lo componen, es decir, cómo está diseñada su arquitectura entre directorios y ficheros.

La creación de un proyecto web de ASP.NET genera una aplicación con una pequeña plantilla, comúnmente llamada **template**, con la cual puedes comenzar a interactuar. Si seguiste el paso a paso anterior, a la derecha verás el explorador de soluciones con el proyecto que acabas de crear.

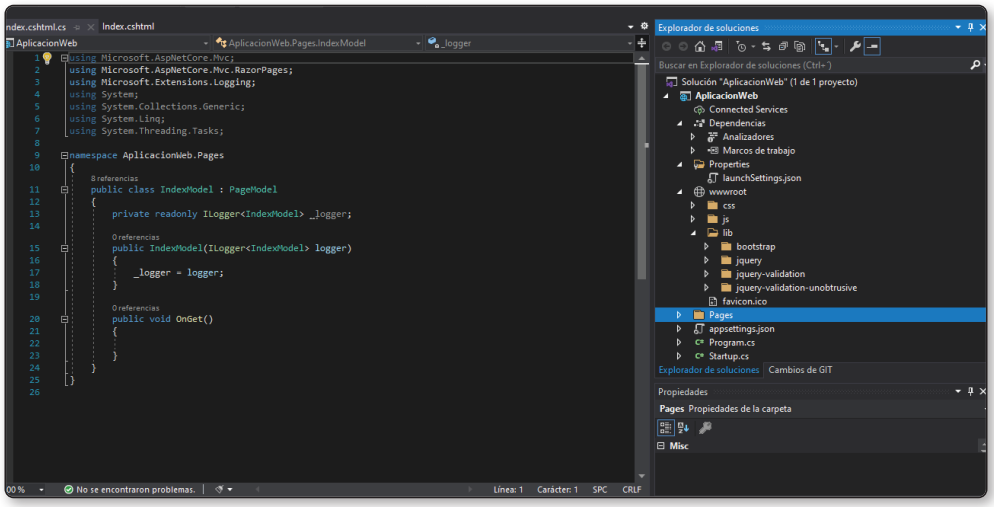


Figura 2.3. Un proyecto web de ASP.NET nuevo suele verse de esta forma.

ASP.NET es un entorno que funciona con C#, y como muchos de los **frameworks** y lenguajes actuales más populares en el mercado, se trabaja con programación orientada a objetos. El proyecto contiene distintas clases que se utilizan para configurar el proyecto, iniciarlo, conectarlo con distintos programas (como motores de bases de datos) e iniciar las plantillas, entre muchísimas otras opciones.

2.3.1 Raíz del proyecto/archivo Startup.cs

Una de las clases más importantes del entorno ASP.NET está en la raíz del proyecto, en el archivo **Startup.cs**.

Este posee en su interior una clase denominada como el nombre del archivo, junto con su método constructor, que es llamado cuando el proyecto se inicia y se pone en funcionamiento.

De la misma manera en que un programa de escritorio necesita iniciarse, los proyectos ASP.NET también necesitan de un comando que los inicia, para poder visualizarlos en el navegador. Verás en detalle cómo se realiza esta tarea más adelante en este capítulo.

Si prestas atención a la clase **Startup**, en su método constructor, encontrarás una instancia de la interfaz **IConfiguration**. Visual Studio te permite visualizar estas clases, que son parte del framework .NET.

2.3.2 Archivo Program.cs

Además de estos archivos, en la raíz del proyecto encontrarás también la clase **Program**, que tiene una llamada al método **public static void Main**, encargado de lanzar o ejecutar el programa:

```
public class Program
{
   public static void Main(string[] args)
   {
   CreateHostBuilder(args).Build().Run();
   }
...
}
```

2.3.3 Archivo appsettings.json

También encontrarás un archivo con extensión **json**, referente a la configuración del proyecto, el cual no es necesario que edites, ya que solo se encarga de configuraciones para el host y el servidor de pruebas.

2.3.4 Carpeta Pages

Dentro del proyecto, hay una carpeta llamada **Pages**, que almacena todos los archivos **cshtml** del proyecto, es decir, todas las vistas o código que va a renderizarse a HTML para ser enviado al navegador luego de compilarse. Los archivos **cshtml** son archivos con lenguaje de marcado **HTML** estándar, pero además, permiten el uso de directivas y código C#, algo muy útil para separar la lógica de la aplicación,

de la parte visual. Este es un concepto conocido como modularización, y su objetivo es crear aplicaciones mucho más escalables, fáciles de mantener y de entender por una persona.

Como podrás ver, el proyecto cuenta con cinco archivos **cshtml**, de los cuales tres —**Index**, **Privacy** y **Error**— son renderizados en la aplicación.

Estos tres archivos, además de contener el código **HTML** de las vistas, funcionan bajo un sistema de herencia de vistas. Verás este tema en detalle más adelante. Además, cada vista requiere de una clase que funciona como gestor de las peticiones que se le hacen a la aplicación.

2.3.5 Carpeta wwwroot

Volviendo a la raíz del proyecto, hay otra carpeta, llamada **wwwroot**, que contiene todos los archivos y carpetas pertenecientes a las librerías necesarias. Como la mayoría de los entornos de trabajo para desarrollar aplicaciones web, no se acostumbra escribir desde cero todo el código **HTML**, **CSS** y **JavaScript**, sino que se recurre a distintas librerías que agilizan este trabajo. En el caso de ASP.NET, las librerías por defecto son **Bootstrap**, como librería de CSS y JavaScript; y **JQuery** para **JavaScript**. El código de ambas está dentro de este directorio y, en caso de utilizar nuevas librerías o cualquier tipo de archivos con material estático (como CSS o imágenes, entre otros), lo normal sería colocarlas dentro de este. También es importante aclarar que todo lo que se ubique en esta carpeta, al momento de publicar o desplegar el sitio a producción, será colocado en la raíz del sitio, con lo cual las llamadas a estos archivos pueden hacerse de manera más sencilla, un procedimiento que pronto verás en detalle.

2.3.6 Carpeta properties

Por otro lado, la carpeta **Properties** solo contendrá otro archivo con extensión **json** que será utilizado para configurar la ejecución de la aplicación; en la carpeta **dependencies** se encontrarán llamadas a los componentes de ASP.NET Core y sus dependencias.

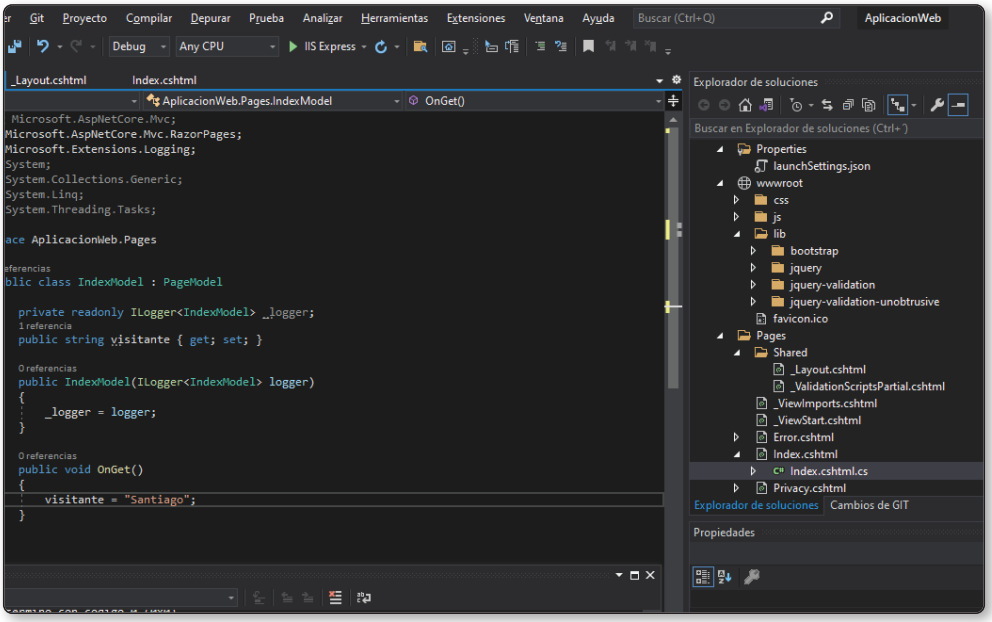


Figura 2.4. Podrás observar que ASP.NET está fuertemente orientado a objetos.

Una vez analizada la estructura del proyecto general, es momento de lanzarlo y ver el proyecto básico que ASP.NET generó como plantilla de trabajo. Para esto, Visual Studio implementa herramientas de compilado que trabajan en conjunto con el SDK. En la parte superior del editor verás una sección dedicada a la depuración y el lanzamiento. La opción más sencilla es hacer clic en **IIS Express**, lo cual lanzará el proyecto actual en tu computadora, de forma local y como prueba, para que lo veas en el navegador.

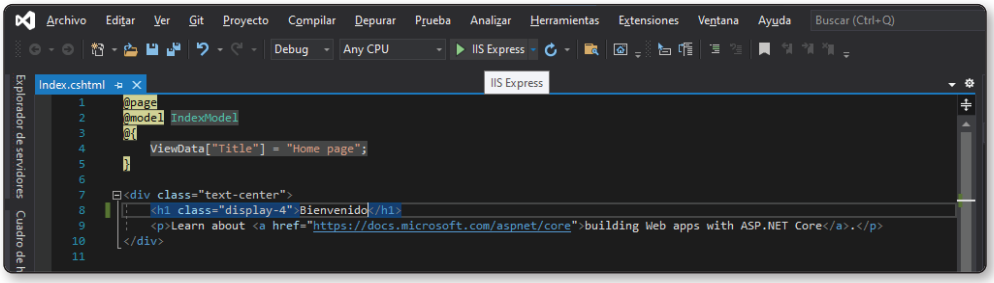


Figura 2.5. Para ejecutar el proyecto, haz clic en el botón verde triangular junto a IIS Express.

Haciendo clic en el botón verde, se lanzará el proyecto y, una vez que termine la compilación, podrás ver la pantalla inicial del proyecto.

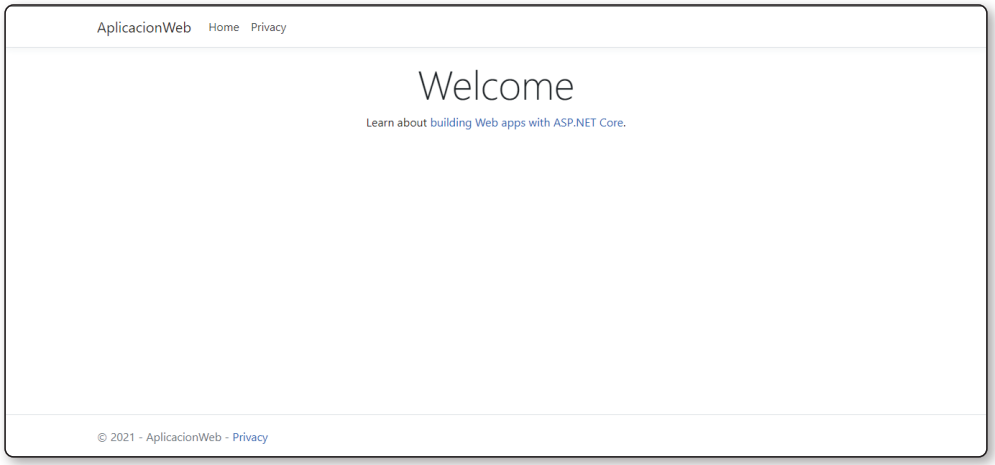


Figura 2.6. Luego de abrir el navegador, se presenta la siguiente pantalla.

Como podrás observar, el proyecto es bastante simple y se compone de las vistas que podías ver dentro de la carpeta **Pages**, un **index** (que funciona como página Home o Inicio), una página de privacidad y una de error, que se lanza en el momento en que hay algún tipo de excepción.

Una vez que hayas visto el pequeño sitio de ejemplo, dirígete de nuevo a Visual Studio, detén la ejecución del programa y abre el archivo **Index. cshtml**, para ver su contenido. En su interior hay algo del código que viste en la página Home, aunque no todo; comienza por cambiar el título **<h1>** de la página, por Bienvenido a mi sitio:

```
@{
   ViewData[“Title”] = “Home page”;
}

<div class=”text-center”>
   <h1 class=”display-4”>Bienvenido a mi sitio web</h1>
</div>
```

Debajo verás un elemento **<p>** con un link a un sitio que tiene la documentación de ASP.NET. Reemplaza este texto por otro en español que dé la bienvenida al visitante, por ejemplo, como si fuera un sitio de compras. Una vez hecho esto, vuelve a ejecutar el programa con IIS Express.