



Integración AdminLTE con ASP.Net Core
ACCIÓN FORMATIVA

At. cliente: 950 23 22 11 | E-mail: central@almerimatik.es
Más info en nuestra web: almerimatik.es



Creación de Aplicaciones ASP.NET Core MVC

Contenido

1. Objetivo del presente documento.....	2
2. Por qué integrar Bootstrap en el proyecto ASP.Net Core MVC	2
3. ¿Qué es AdminLTE?	2
4. Obtener AdminLTE.....	3
5. Estructura de carpetas	3
6. Configuración del proyecto ASP.Net Core MVC con autenticación.	4
7. Descripción de diseños y vistas parciales.....	6
8. Integración de AdminLTE con ASP.Net Core	8
9. Copia de los recursos necesarios.	9
10. Adición de páginas de diseño y vistas parciales.....	9
11. Agregar navegación	14
12. Indicador de navegación	15
13. Integración de la interfaz de usuario con la autenticación existente.	16
14. Habilitación de la autenticación	21
15. Menú basado en estado autenticado	21
16. Corrección del complemento filterzr de AdminLTE	23
17. Resumen	24

1. Objetivo del presente documento

El objetivo que pretendemos cubrir con la presente acción formativa es aprender a crear una aplicación que contemple los siguientes aspectos:

- Integración con una plantilla de arranque de terceros
- Uso limpio y separación de diseños, vistas y vistas parciales
- Integración de la autenticación de usuario.
- Diseño por perfiles de usuario
- Inicios de sesión, registro, cierre de sesión y más.

2. Por qué integrar Bootstrap en el proyecto ASP.Net Core MVC

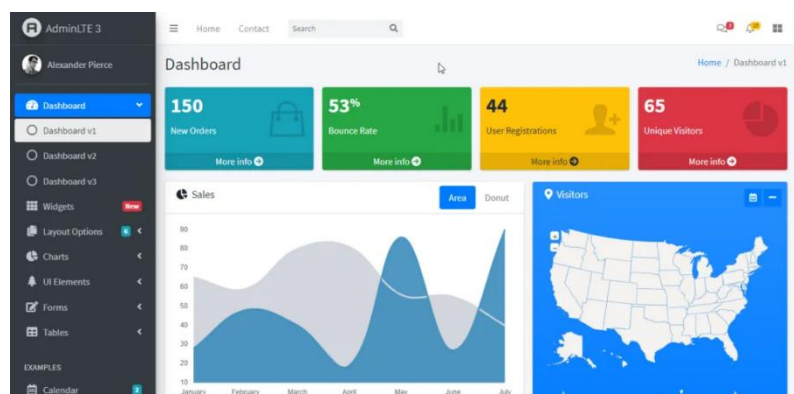
Si usted es un desarrollador de back-end que posea conocimientos avanzados para crear plantillas HTML5 / CSS3 / JS completa desde cero, probablemente desee buscar otras opciones que hagan que su aplicación se vea 100 veces más profesional. Hoy en día, hay bastantes opciones, incluidas las plantillas de pago, que le permiten crear aplicaciones con un aspecto 100% moderno y profesional.

Para el presente curso, utilizaremos una plantilla de código abierto que es bastante popular, AdminLTE. Las ventajas de usar una plantilla de código abierto a construida son las siguientes:

- Presenta interfaz de usuario profesional.
- Está probado.
- En la mayoría de los casos tiene un diseño Responsivo (sensible al área de presentación del dispositivo donde se abre el cliente)
- Acceso a un amplio numero de componentes reutilizables como tablas de datos, formularios, cajas de texto, deslizadores, casillas de verificación, calendarios y muchos más, para que no tengas que reinventar la rueda.

3. ¿Qué es AdminLTE?

AdminLTE es una plantilla de panel de administración de código abierto que se basa en Bootstrap. Está repleto de una gran cantidad de componentes receptivos y de uso común que se integran muy fácilmente con sus aplicaciones web. Para obtener una mejor imagen, haga clic [aquí](#) para ver una demostración de AdminLTE en acción.



Como puede apreciar en la imagen la plantilla presenta un aspecto totalmente profesional que hará que nuestras aplicaciones luzcan 100% profesionales y amigables para el usuario.

Si probamos la respuesta de la página a la vista de diseño adaptable en nuestro navegador, podremos observar que se ajusta perfectamente a los diferentes tamaños que utilicemos, de forma que éste será un aspecto menos del que nos tendremos que preocupar a la hora de construir nuestra aplicación.

Por ahora, estas páginas no están vinculadas a ninguna aplicación del lado del servidor, es simplemente un archivo HTML 5. En este artículo, integraremos esta interfaz de usuario con nuestra aplicación ASP.NET Core MVC e incluiremos en el diseño algunas buenas prácticas para la creación de aplicaciones MVC 100% profesionales.

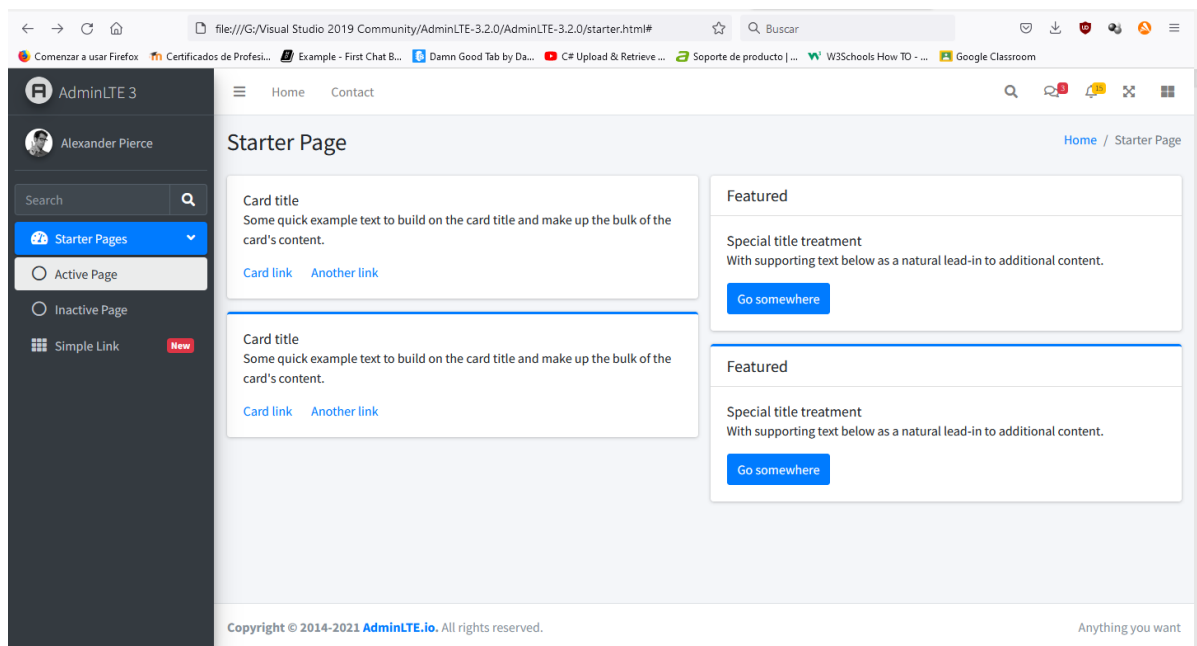
4. Obtener AdminLTE

AdminLTE es completamente **GRATIS**. [Siga este enlace para comenzar a descargar](#) . Al momento de escribir este documento, 3.2.0 es la última versión disponible. Haga clic en el enlace al [código fuente](#) para descargar el archivo comprimido en su máquina.

5. Estructura de carpetas

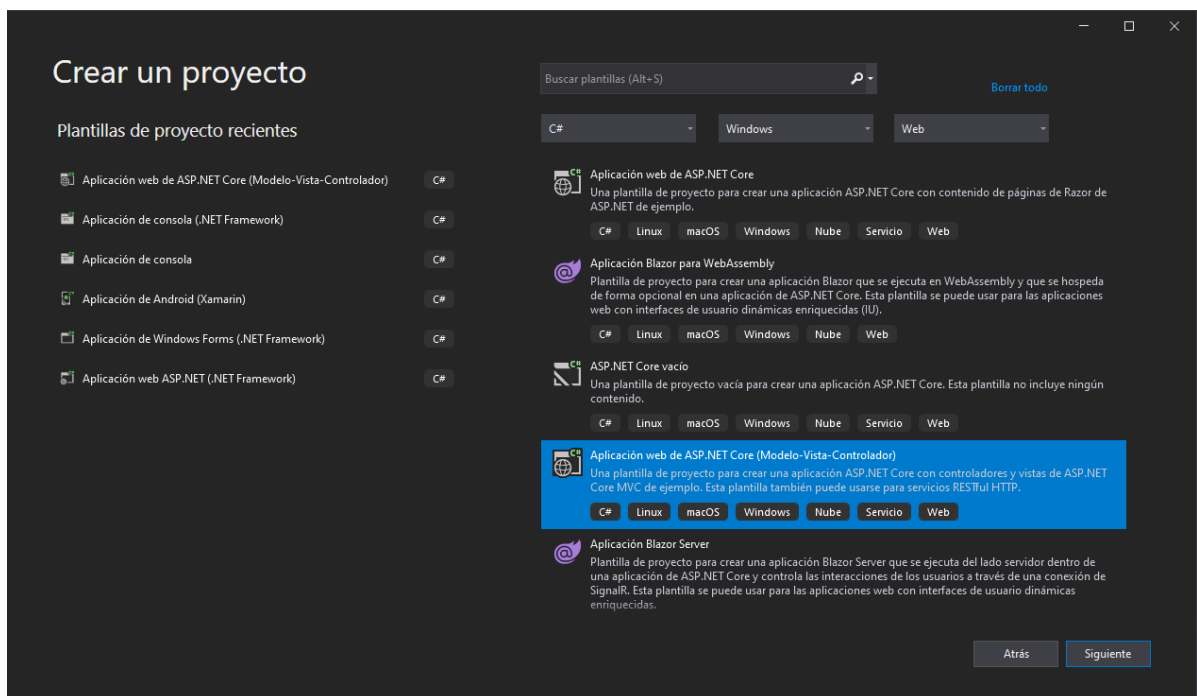
Una vez descargado, extraiga el archivo comprimido. Aquí encontrará un montón de carpetas y archivos. No tendremos que tocar todos y cada uno de los archivos, sino solo unos pocos. Le daré una breve descripción de lo que contiene cada carpeta.

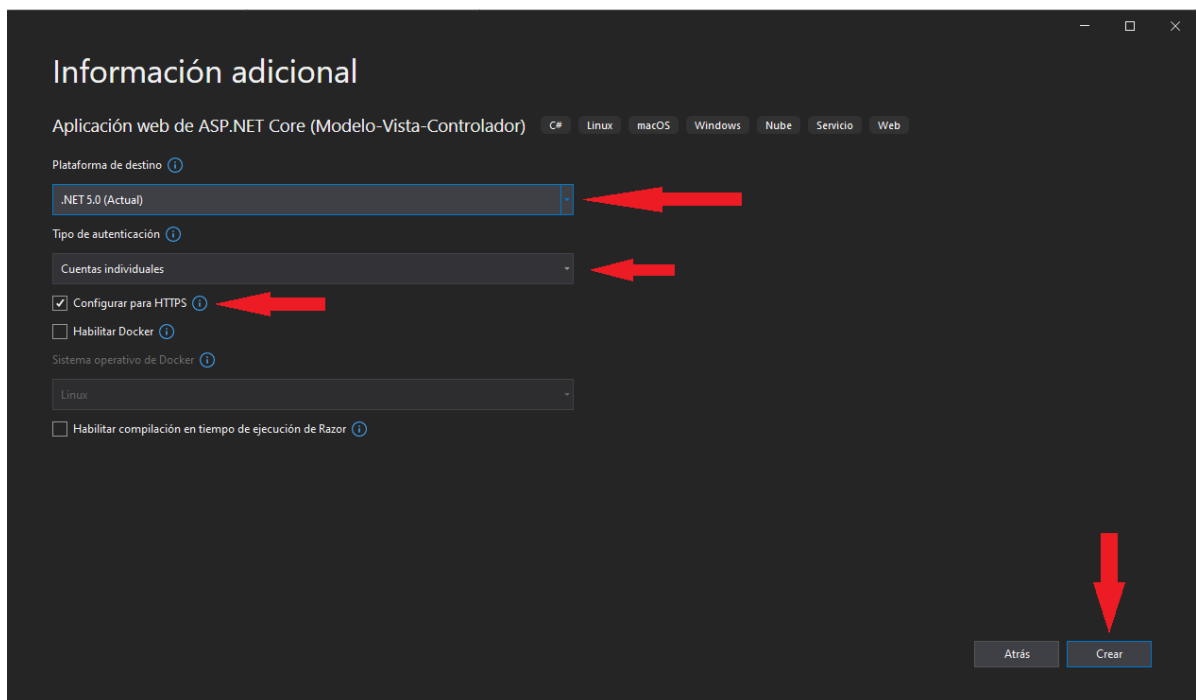
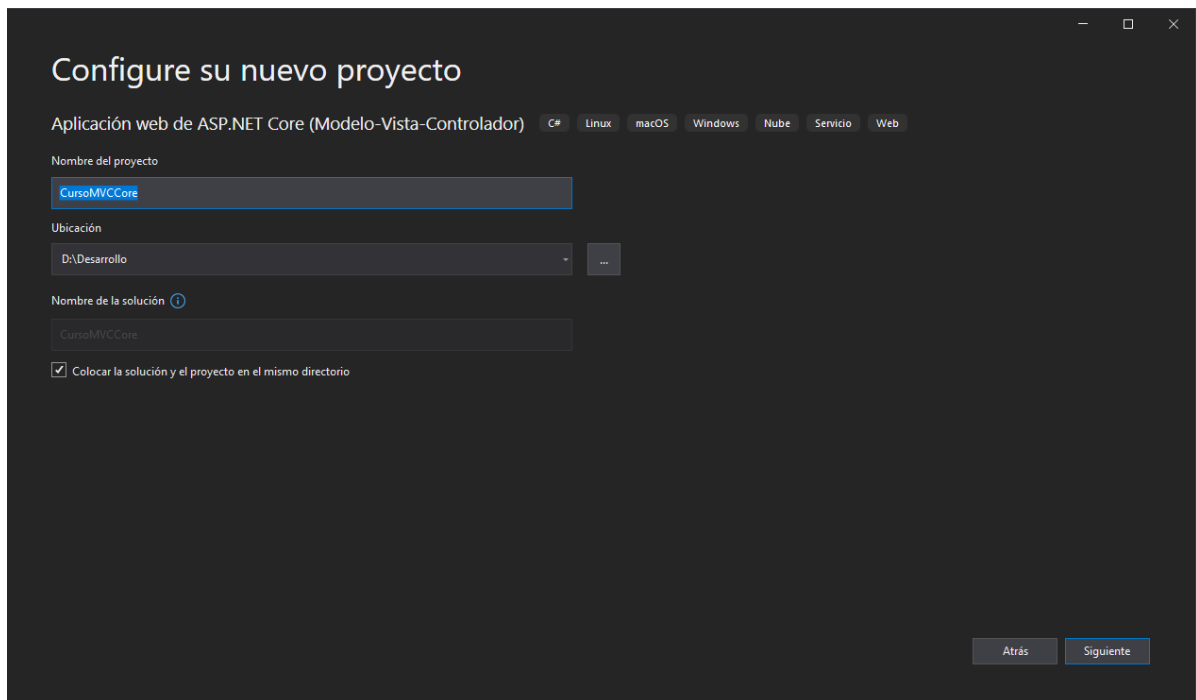
- **dist/**: esta es la carpeta de distribución que contiene todos los archivos css y js, o sea, principalmente todos los archivos estáticos de la aplicación. Esta carpeta la tendremos que copiar esta carpeta dentro de la carpeta `wwwroot` de nuestro Proyecto MVC más adelante.
- **pages/**: En esta carpeta tenemos toda una lista de archivos HTML prediseñados que usaremos principalmente como ejemplos de uso. Esta es una sección bastante importante ya que utiliza todos los componentes disponibles y que nos va a resultar especialmente útil para verificar cómo se utilizan los componentes.
- **plugins/**: aquí se incluyen complementos JS de terceros como select2, jquery, datatables, etc. Necesitaremos esta carpeta también.
- **starter.html**: En este archivo tenemos una estructura mínima del archivo HTML de inicio. Nosotros vamos a usar esta página para generar el archivo `_layout.cshml` de nuestra aplicación ASP.NET Core MVC. En la imagen adjunta podemos ver una captura de pantalla de la salida del archivo `starter.html`.



6. Configuración del proyecto ASP.Net Core MVC con autenticación.

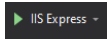
Vamos a crear una nueva aplicación ASP.NET Core con la plantilla Model-View-Controller (MVC). **Asegúrese de seleccionar el modo de autenticación para cuentas de usuario individuales.** Esto nos permite usar la Autenticación integrada (usando Microsoft Identity).

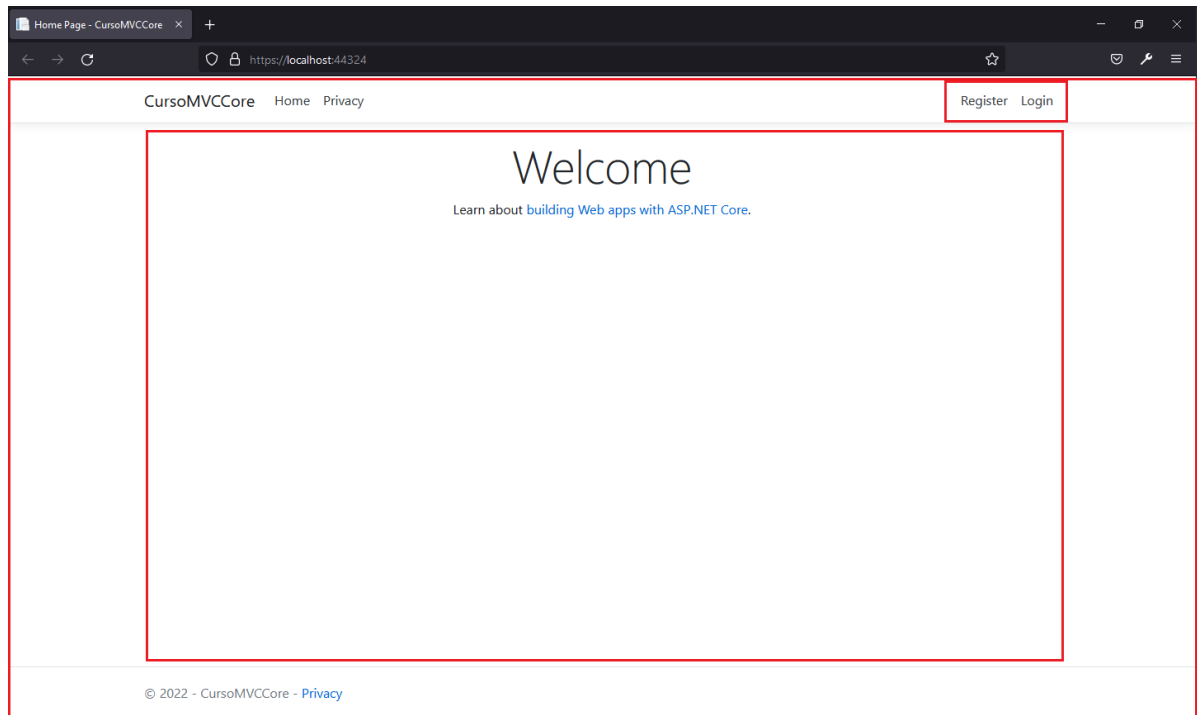




Ahora que tenemos listos nuestros archivos AdminLTE y la aplicación ASP.NET Core, comencemos a integrarlos. Antes de comenzar, veamos cómo funciona el diseño predeterminado de ASP.NET Core.

7. Descripción de diseños y vistas parciales.

Pulsamos la flecha  o la tecla de función F5 y generamos y ejecutamos la aplicación. Este es el diseño predeterminado que viene con las aplicaciones web ASP.NET Core 5.0



La página se divide de la siguiente forma.

- Página de diseño principal: este es el diseño general de nuestra aplicación. Contendrá la vista común a todo el diseño de nuestra aplicación y está definido en el documento `_layout.cshtml`¹ que se encuentra en la ubicación `“/Views/Shared/”`
- Panel de navegación: Se encuentra dentro del diseño principal, se define una referencia de vista parcial que llama a la página `_LoginPartial.cshtml`.
- Cuerpo del contenido: aquí es donde vamos a mostrar el contenido de cada una de las secciones de nuestra aplicación.

Ahora bien, ¿por qué este tipo de separación?

En ASP.NET Core MVC, puede generar vistas (CHTML) mediante controladores. Imagine una aplicación de la vida real que tenga múltiples controladores y vistas. Realmente no desea definir el HTML completo para cada vista, ¿verdad? Porque ya sabemos que tendremos una plantilla común para todo el sitio. Entonces, lo que hace este concepto de diseño es que define el diseño cshtml solo una vez y agrega el contenido de cada una de las demás páginas dinámicamente dentro de la página de diseño.

¹ ASP.NET Core usa la extensión `“.cshtml”` (archivos de marcado Razor) para sus páginas

Por lo tanto, define toda la parte HTML que, en esencia, sería común en toda la aplicación, y separa el contenido dinámico en otra página cshtml. De esta forma, podemos hacer que la aplicación sea mucho más fácil de mantener y reducir el tamaño de toda la aplicación.

Examinemos el archivo donde se define el diseño predeterminado. Dado que creamos una aplicación con plantilla MVC, podemos encontrar este archivo en ../Views/Shared/_Layout.cshtml en la estructura de la solución. Puede ver que esta página comienza con una etiqueta HTML. Esto sugiere que se trata de una página HTML completa con todas las etiquetas de cuerpo y encabezado.

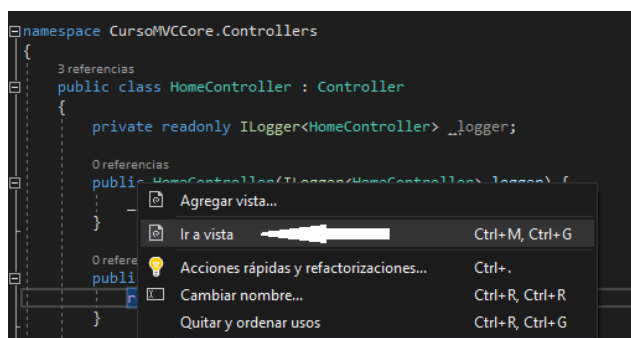
Examinando el código en la línea 33 vamos a encontrar el siguiente bloque de código:

```
<div class="container">
  <main role="main" class="pb-3">
    @RenderBody()
  </main>
</div>
```

@RenderBody se usa para representar el contenido de la página secundaria. Cuando ejecuta la aplicación ASP.NET Core, navega a la raíz de la aplicación, es decir, localhost:XXXX/Home e invoca el método de "Index" en "HomeController" (ya que está configurado como la ruta predeterminada). Veamos qué contiene este método.

```
public IActionResult Index() {
    return View();
}
```

Como podemos ver el método Index simplemente devuelve una vista (.cshtml). Haga clic derecho en la vista y haga clic en Ir a la vista. Esto nos llevará al archivo CSHTML asociado



```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
  <h1 class="display-4">Welcome</h1>
  <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

Este contenido se carga mediante la etiqueta @RenderBody, de modo que en tiempo de ejecución obtiene la página web completa (incluido el contenido HTML estático y el código dinámico generado por C#).

Ahora, hablemos un poco sobre las vistas parciales. En la página _Layout.cshtml, puede encontrar esto en algún lugar de la línea 20 más o menos.

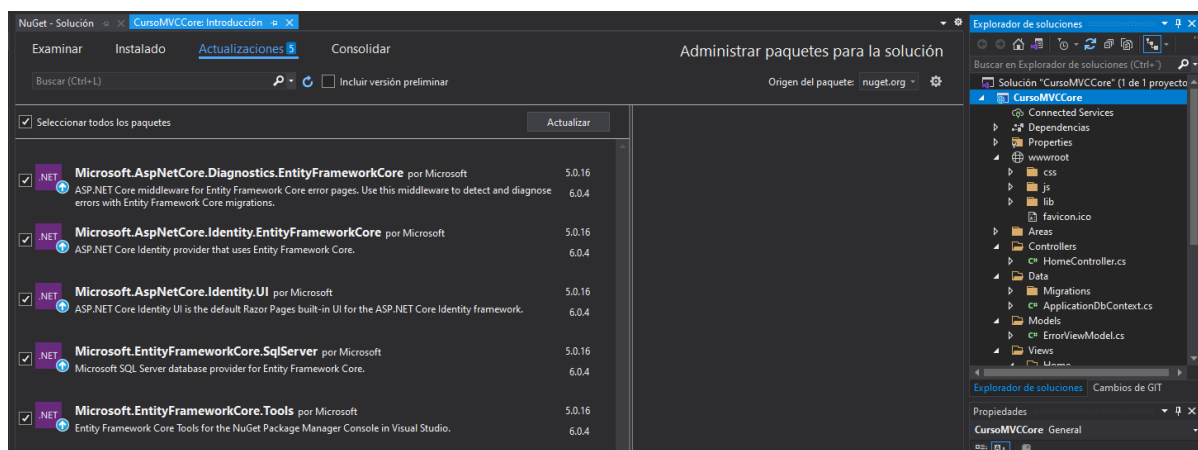
```
<partial name="_LoginPartial"/>
```

A diferencia de las vistas, las vistas parciales en ASP.NET Core representan una salida HTML dentro de la salida representada de otra vista. Puede haber casos en los que sus aplicaciones tengan componentes que se puedan reutilizar en cualquier lugar dentro de la aplicación.

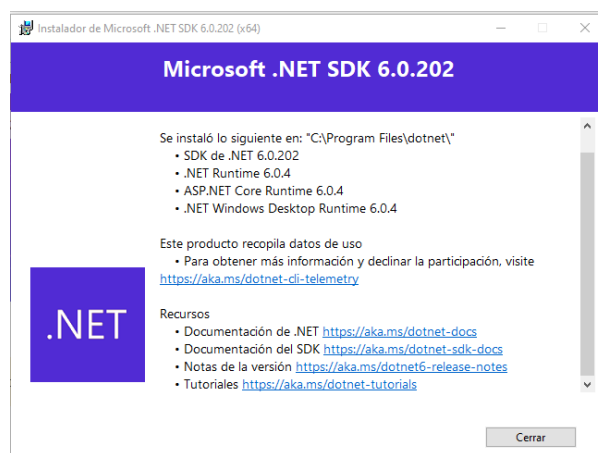
En nuestro caso tenemos la barra de navegación superior que es bastante común a través de la aplicación. También puede encontrar _LoginPartial.cshtml en la carpeta compartida.

8. Integración de AdminLTE con ASP.Net Core

Ahora que hemos repasado los conceptos básicos de diseños, vista y vistas parciales y los tenemos claros, integrar en nuestras aplicaciones ASP.NET Core un diseño HTML CSS3 y JavaScript de terceros será más sencilla. Antes de proceder con la integración Actualicemos los paquetes NuGet referenciados en nuestro proyecto.



En nuestro caso ya disponemos de la última versión de los paquetes existentes para la plataforma .NET 5. Visual Studio 2019 tan solo viene acompañado del SDK de .NET Core 5 si deseamos desarrollar nuestra aplicación para .NET Core 6, tendremos que descargar el SDK desde el sitio [Web de .Net de Microsoft](https://aka.ms/dotnet6-release-notes) e instalarlo en nuestro sistema, Instalar Visual Studio Code y/o Visual Studio 2022 y crear nuestro proyecto ahí.



9. Copia de los recursos necesarios.

Como se mencionó anteriormente, AdminLTE se basa en Bootstrap. Por lo tanto, también contiene muchas integraciones de jquery y js. No solo estaríamos copiando el contenido HTML, sino también los recursos relacionados, como css, imágenes, bibliotecas, archivos js, etc.

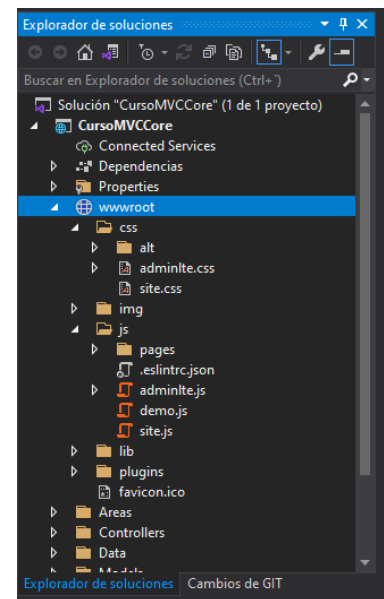
Descomprima el archivo zip que ha descargado desde el sitio web de AdminLTE fuera del proyecto. Una vez haya finalizado el proceso, entre en la carpeta AdminLTE, navegue hasta \dist\css y copie el contenido en la carpeta wwwroot\css en nuestro Visual Studio. Haga lo mismo con el contenido de la carpeta js².

A continuación, copie toda la carpeta img a la carpeta wwwroot. Vuelva a la raíz de la carpeta AdminLTE y copie también la carpeta de plugins en la carpeta wwwroot.

Así es como se debería ver la carpeta wwwroot después de que terminemos de copiar el contenido.

Tenga en cuenta que, aunque no usaremos todos los complementos incluidos en la carpeta plugins por ahora, vamos a conservarlos al menos hasta que hayamos finalizado la construcción de la aplicación y tan solo eliminaremos los archivos si nos vemos en la necesidad de ahorrar espacio a la hora de publicar nuestra aplicación.

Ahora que disponemos de todos los recursos que necesitamos en nuestro proyecto para desarrollar nuestra aplicación, vamos a modificar las páginas del diseño para hacer funcionar nuestra aplicación.



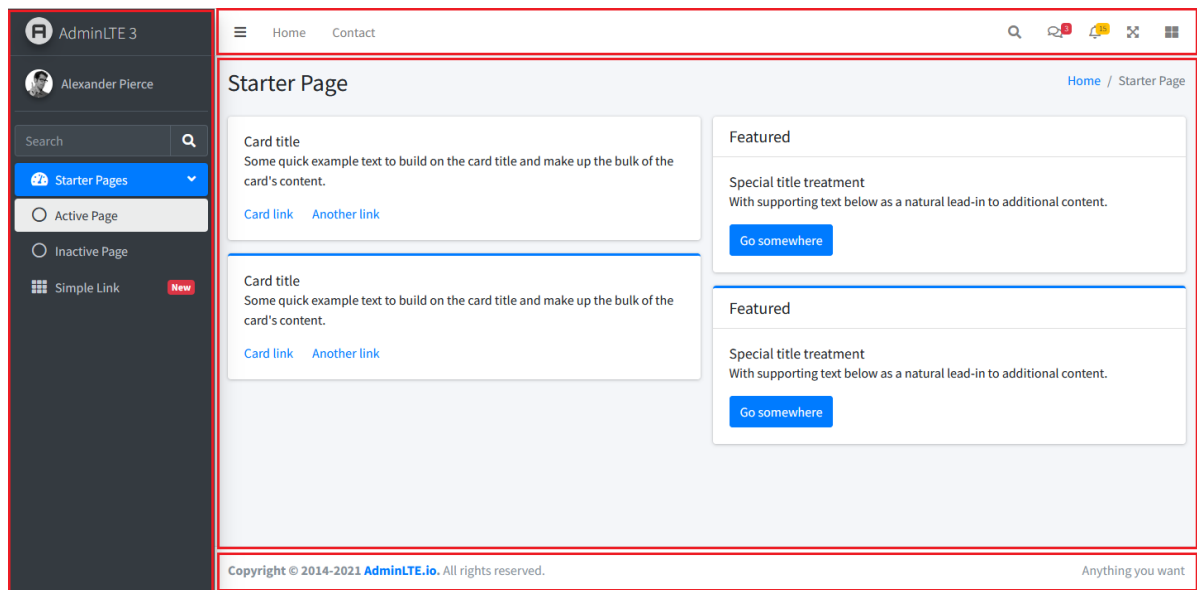
10. Adición de páginas de diseño y vistas parciales.

Una aplicación puede contener, y de hecho contendrá, varias páginas de diseño. En este curso no modificaremos la página “_Layout.cshtml” existente, sino que construiremos una específica para AdminLTE.

En la Carpeta shared, cree una nueva carpeta denominada AdminLTE. Aquí es donde vamos a ir colocando todos los archivos “.cshtml” relacionados con AdminLTE que vayamos a crear.

Antes de construir las páginas de diseño y las vistas parciales, decidamos cómo separaremos el contenido HTML. Tenga en cuenta que usaremos la página starter.html de AdminLTE para crear el diseño. Abra la página starter.html en su navegador.

² Para copiar el contenido, simplemente copie la carpeta/archivo seleccionado y vaya a Visual Studio. Aquí puede encontrar una carpeta wwwroot. Esta es la carpeta destinada a contener los archivos estáticos de las aplicaciones ASP.NET Core. Simplemente pegue el archivo copiado aquí con un simple comando CTRL+V.

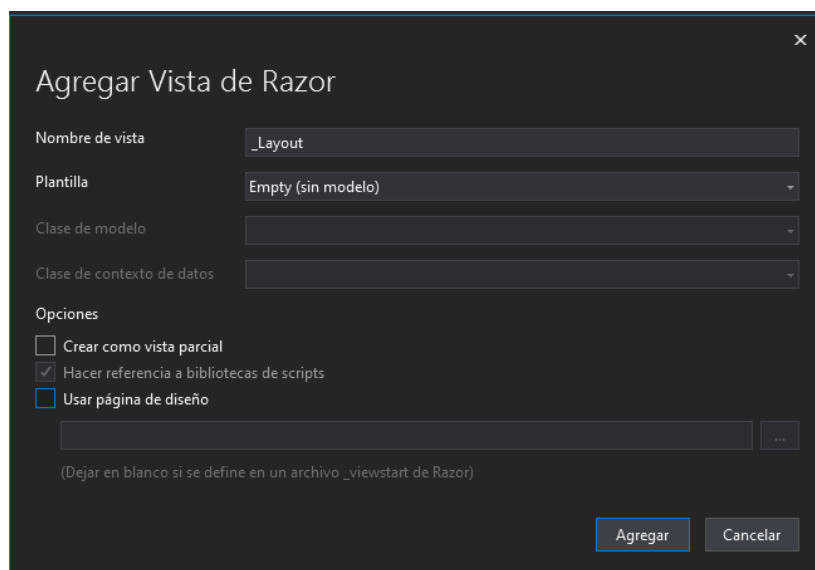


De acuerdo con la imagen podríamos dividir la página web en las siguientes secciones:

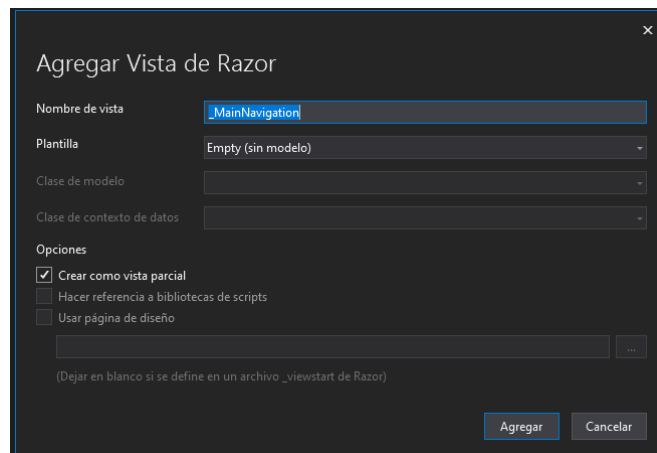
- Navegación lateral
- Navegación superior.
- Cuerpo
- Pie de página

Además de las citadas anteriormente, vamos a agregar dos vistas parciales más que van a contener las referencias a los archivos CSS y JS. De esta forma va a ser más sencillo ordenar y administrar las referencias a los recursos de la aplicación.

Empezamos entonces creando una nueva Vista en la carpeta AdminLTE que vamos a llamar “_Layout.cshtml”. las opciones de vista parcial y diseño de página deben estar desmarcadas.

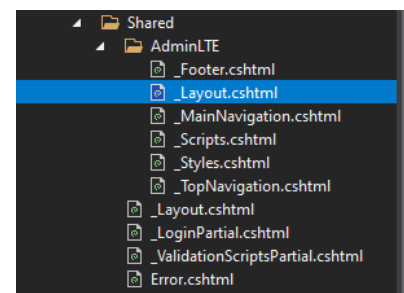


A continuación, comencemos a agregar los archivos de vista parcial. En la misma carpeta AdminLTE, agregue una nueva vista y asígnele el nombre `_MainNavigation.cshtml`. Esta vez, marcaremos la opción "crear como vista parcial".



De la misma forma vamos a añadir otras vistas parciales con los siguientes nombres de archivo dentro de la carpeta AdminLTE:

- `_TopNavigation`
- `_Footer`
- `_Scripts`
- `_Styles`



Una vez creados los archivos vamos a agregar contenido a cada uno de ellos.

Ahora que ya tenemos la idea de cómo dividiremos el contenido del archivo HTML. Abrimos el archivo "starter.html" en el editor que prefieras, yo suelo utilizar Notepad++, y comenzamos a mover el código HTML contenido en él (más de trescientas líneas) a cada uno de los archivos que hemos creado uno por uno.

Scripts: Desplácese hasta el final del archivo. Encima de la etiqueta `"body"`, puede encontrar algunas de las referencias al script allí. Córtelo y péguelo en el archivo `_Scripts.cshtml`. sustituya en el archivo `starter.html` con la siguiente etiqueta:

```
<partial name="AdminLTE/_Scripts"/>
```

Footer: Justo encima de donde se definieron los scripts, puede encontrar el contenedor de pie de página. Corte esto y muévase a `_Footer.cshtml`. En su lugar, agregue lo siguiente en `starter.html`. Moveremos los códigos de aquí a `_Layout.cshtml` una vez que hayamos terminado con las vistas parciales.

```
<partial name="AdminLTE/_Footer"/>
```

Body: En el archivo Starter.html busca una etiqueta div de clase “content” que está comentada como “Main Content”. Elimínala y sustituye el contenido con el siguiente código:

```
<!-- Main content -->
<div class="content">
  <div class="container-fluid">
    <div class="row">
      <main role="main" class="pb-3">
        @RenderBody()
      </main>
    </div>
  <!-- /.row -->
</div>
<!-- /.container-fluid -->
</div>
<!-- /.content -->
```

MainNavigation: Busque en el archivo Starter.html una etiqueta de la clase “main-sidebar”. Corte todo el contenido y muévelo al archivo _MainNavigation.cshtml. Sustituya el texto cortado con la referencia al archivo parcial.

TopNavigation: Busque la clase “main-header” y muévelo al archivo _TopNavigation.cshtml. Al igual que en el caso anterior, sustituya el texto que ha movido por la referencia al archivo parcial.

Styles: Bajo la etiqueta “title” podemos encontrar un montón de referencias a hojas de estilo, las cortamos y las copiamos en el archivo parcial _Scripts.cshtml, reemplazándolo por la referencia a dicho archivo.

Bien, ahora que hemos movido cada uno de los posibles componentes a las vistas parciales, lo que quede debe parecerse al siguiente bloque. Cópelo dentro del archivo _layout.cshtml

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="x-ua-compatible" content="ie=edge" />
  <title>AdminLTE 3 | Starter</title>
  <partial name="AdminLTE/_Styles" />
</head>
<body class="hold-transition sidebar-mini">
  <div class="wrapper">
    <!-- Navbar -->
    <partial name="AdminLTE/_TopNavigation" />
    <!-- /.navbar -->
    <!-- Main Sidebar Container -->
    <partial name="AdminLTE/_MainNavigation" />
    <!-- /.sidebar -->
  <!-- sigue -->
```

```

<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
  <!-- Content Header (Page header) -->
  <div class="content-header">
    <div class="container-fluid">
      <div class="row mb-2">
        <div class="col-sm-6">
          <h1 class="m-0">Starter Page</h1>
        </div><!-- /.col -->
        <div class="col-sm-6">
          <ol class="breadcrumb float-sm-right">
            <li class="breadcrumb-item"><a href="#">Home</a></li>
            <li class="breadcrumb-item active">Starter Page</li>
          </ol>
        </div><!-- /.col -->
      </div><!-- /.row -->
    </div><!-- /.container-fluid -->
  </div>
  <!-- /.content-header -->
  <!-- Main content -->
  <div class="content">
    <div class="container-fluid">
      <div class="row">
        <main role="main" class="pb-3">
          @RenderBody()
        </main>
      </div>
      <!-- /.row -->
    </div>
    <!-- /.container-fluid -->
  </div>
  <!-- /.content -->
</div>
<!-- /.content-wrapper -->
<!-- Control Sidebar -->
<aside class="control-sidebar control-sidebar-dark">
  <!-- Control sidebar content goes here -->
  <div class="p-3">
    <h5>Title</h5>
    <p>Sidebar content</p>
  </div>
</aside>
<!-- /.control-sidebar -->
<!-- footer -->
<partial name="AdminLTE/_Footer" />
<!-- ./footer -->
</div>
<!-- ./wrapper -->
<!-- scripts -->
<partial name="AdminLTE/_Scripts" />
<!-- ./scripts -->
</body>
</html>

```

Felicidades, hemos separado los HTML en diseños y vistas parciales. Ahora compila y ejecuta la aplicación. No verías absolutamente ningún cambio. ¿Por qué? Porque no hemos mencionado ningún lugar para usar nuestro nuevo diseño, ¿verdad? Para esto, vaya a Views/Home/Index.cshtml y mencione el diseño manualmente.

Si ejecutamos la aplicación nuevamente, vas a ver muchos cambios, pero serán decepcionantes, ya que la página resultado está “aparentemente” rota. Sin embargo, puede ver que podemos mostrar el contenido que pasamos desde la Vista. Ahora vamos a arreglar la página.

Si nos paramos a pensar por qué nuestra página se muestra mal, si tenemos algún conocimiento de HTML5 CSS3 y JavaScript, será bastante sencillo llegar a la conclusión de que nuestra página no puede encontrar los archivos que contienen los estilos, y efectivamente así es. Hemos copiado los estilos y scripts a las Vistas parciales, pero no cambiamos las referencias. Cambiemos las rutas para que apunten a los archivos específicos en la carpeta wwwroot.

Una vez que hayamos hecho los cambios observaremos que la página comienza a verse muy bien, tan solo vemos que tenemos algún problemilla con las imágenes cuyo vínculo aparece roto. Bien, podemos solucionarlo de manera similar yendo a _mainNavigation.cshtml y arreglando las referencias como hicimos antes.

11. Agregar navegación

Agreguemos un menú de navegación a nuestro _MainNavigation.cshtml. Lo que tenemos que hacer es simple. Eliminamos las rutas estáticas y agregaremos enlaces a nuestros métodos de controlador. Por ahora, agregaremos 2 elementos de navegación. Inicio y Privacidad.

```
<!-- Brand Logo -->
<a asp-controller="Home" asp-action="Index" class="brand-link">
  
  <span class="brand-text font-weight-light">AdminLTE 3</span>
</a>
.
.
.
<ul class="nav nav-treeview">
  <li class="nav-item">
    <a asp-controller="Home" asp-action="Index" class="nav-link active">
      <i class="nav-icon fa fa-home"></i>
      <p>Inicio</p>
    </a>
  </li>
  <li class="nav-item">
    <a asp-controller="Home" asp-action="Privacy" class="nav-link">
      <i class="nav-icon fa fa-lock"></i>
      <p>Privacidad</p>
    </a>
  </li>
</ul>
```

Tal y como vemos en el fragmento de código adjunto hemos sustituido el código de la referencia href del logo por nuestra referencia al controlador `Home` y a la acción del controlador `Index` y dentro de la etiqueta de la clase `"nav nav-treeview"` hemos añadido los enlaces a nuestro controlador y sus métodos.

Tras comprobar que eso también esta funcionando observamos que hay un detalle que se nos escapa. El indicador de navegación. No hay ninguna indicación en nuestra barra lateral sobre la página que se está viendo actualmente. Por ahora, al disponer únicamente de 2 elementos de navegación puede parecer poco importante, pero por lo común es bastante interesante indicar qué página es la que estamos viendo en cada momento. Ya que esto es lo ideal, arreglémoslo primero.

12. Indicador de navegación

Para que esto funcione, necesitamos datos de nuestro ASP.NET Core con respecto al controlador actual y el método de acción. Y en base a esto, debemos cambiar la clase del elemento de navegación correspondiente a activo. Activo significa la página actual.

Agregue una nueva carpeta en la raíz del proyecto. Nómbrala “Helpers”. Dentro de esta carpeta añada una nueva clase que vamos a llamar “NavigationIndicatorHelper”. Añada el siguiente código:

```
namespace CursoMVCCore.Helpers {
    public static class NavigationIndicatorHelper {

        public static string MakeActiveClass(this IUrlHelper urlHelper, string controller, string
action) {
            try {
                string result = "active";
                string controllerName =
urlHelper.ActionContext.RouteData.Values["controller"].ToString();
                string methodName =
urlHelper.ActionContext.RouteData.Values["action"].ToString();
                if (string.IsNullOrEmpty(controllerName)) return null;
                if (controllerName.Equals(controller, StringComparison.OrdinalIgnoreCase)) {
                    if (methodName.Equals(action, StringComparison.OrdinalIgnoreCase)) {
                        return result;
                    }
                }
                return null;
            } catch (Exception) {
                return null;
            }
        }
    }
}
```

Esto es lo que hace esta clase auxiliar. Tiene un método de extensión con URLHelper. De esta manera, también puede invocarlo en la página cshtml. Toma el controlador y el nombre del método de acción y lo verifica con los datos de la ruta actual. Si coinciden, devolvemos la cadena "activa", de lo contrario, nulo.

Ahora, vayamos a la vista de la barra de navegación lateral y hagamos las siguientes modificaciones:

```
@using static CursoMVCCore.Helpers.NavigationIndicatorHelper;
.
.
<a asp-controller="Home" asp-action="Index"
    class="nav-link @Url.MakeActiveClass("home", "index")">
.
.
<a asp-controller="Home" asp-action="Privacy"
    class="nav-link @Url.MakeActiveClass("home", "privacy")">
```

Ejecute la aplicación. Ahora tenemos un indicador de navegación que funciona. Elegante, ¿Verdad?

13. Integración de la interfaz de usuario con la autenticación existente.

Dibujemos un escenario. El requisito es que necesitamos asegurar nuestras vistas, en nuestro caso, tenemos páginas de Índice y Privacidad. No disponemos de ninguna página que esté disponible para los usuarios visitantes, así pues, únicamente vamos a permitir el acceso a las páginas a aquellos usuarios que estén identificados.

Otro escenario podría ser que la página “Index” del programa está disponible para todos los usuarios, autenticados o no, pero el resto de las páginas tan solo serían accesibles a aquellos usuarios que se haya sido identificados.

Si los visitantes intentan acceder al recurso (sin autenticación) navegando directamente a cualquiera de los recursos protegidos, lo redirigimos a una página de inicio de sesión. Así que esto es lo que haremos aquí. Este es un escenario bastante práctico, ¿Verdad?

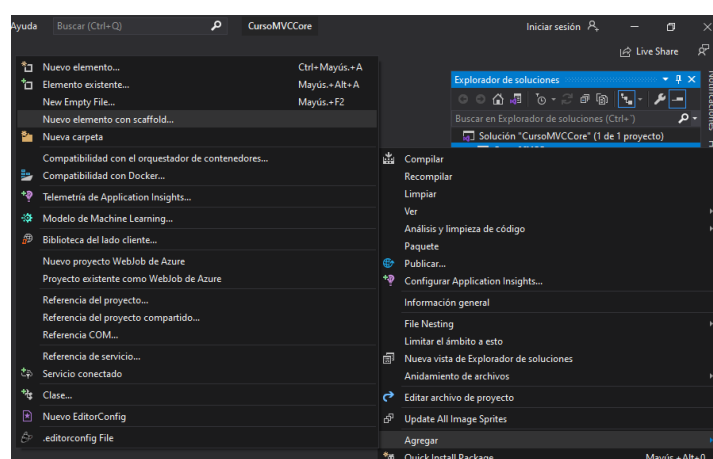
¡AdminLTE también viene con una página predeterminada de inicio de sesión y registro! Se encuentra en la carpeta “/pages/examples” como login.html y register.html.

A estas alturas, ya tendría bastante claro cómo integrar la interfaz de usuario. Pero hay una trampa aquí. Si examina nuestra aplicación ASP.NET Core MVC, no encontrará ninguna página de inicio de sesión o registro. Pero agregamos autenticación a la aplicación mientras la creamos, ¿recuerdas? Entonces, ¿cómo funciona eso?

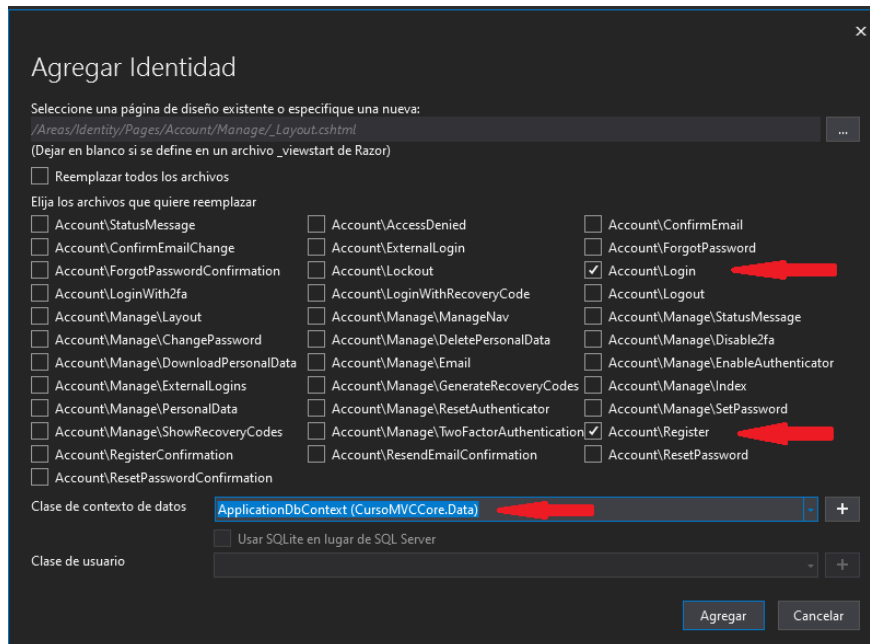
Bien, en algún momento, durante el anuncio de ASP.NET Core 2.1. Microsoft reveló que la interfaz de usuario de Identity (todas las páginas relacionadas con Microsoft Identity) se moverían a una biblioteca Razor. Por lo tanto, funciona de forma inmediata, aunque no sea visible para nosotros.

Pero, ¿y si tuviéramos que hacerle alguna modificación? Ahí es donde entra en juego **Identity Scaffold**. Siga estos pasos para recuperar las páginas de inicio de sesión y registro de Razor para que las modifiquemos.

Hagamos clic con el botón derecho en **Proyecto** -> **Agregar nuevo** -> **Nuevo elemento con scaffold**.



En el siguiente cuadro de dialogo, seleccionamos Identidad y hacemos clic en agregar y nos aparecerá el siguiente cuadro de diálogo.

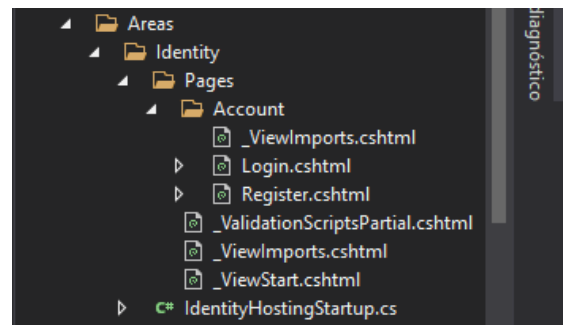


Ahora, podemos seleccionar las páginas de identidad requeridas. Para simplificar las cosas, agreguemos solo la página de inicio de sesión y registro. Asegúrese de seleccionar también la clase de datos. Haga clic en Agregar. Ahora, Visual Studio hace su magia y genera los archivos seleccionados.

En segundo plano, Visual Studio también crea una clase **DataContext** para usted y la registra en los servicios de inicio con una base de datos local predeterminada.

Una vez hecho esto, verá la siguiente carpeta con nuestras páginas de identidad seleccionadas.

Ahora integraremos estas vistas con *login.html* y *register.html*. Lo ideal es que intente hacer la integración por su cuenta, ese proceso le ayudaría a comprender mejor los escenarios. No obstante, en los cuadros adjuntos tiene el código de los dos archivos modificados.



Inicio de Sesión: Login.cshtml

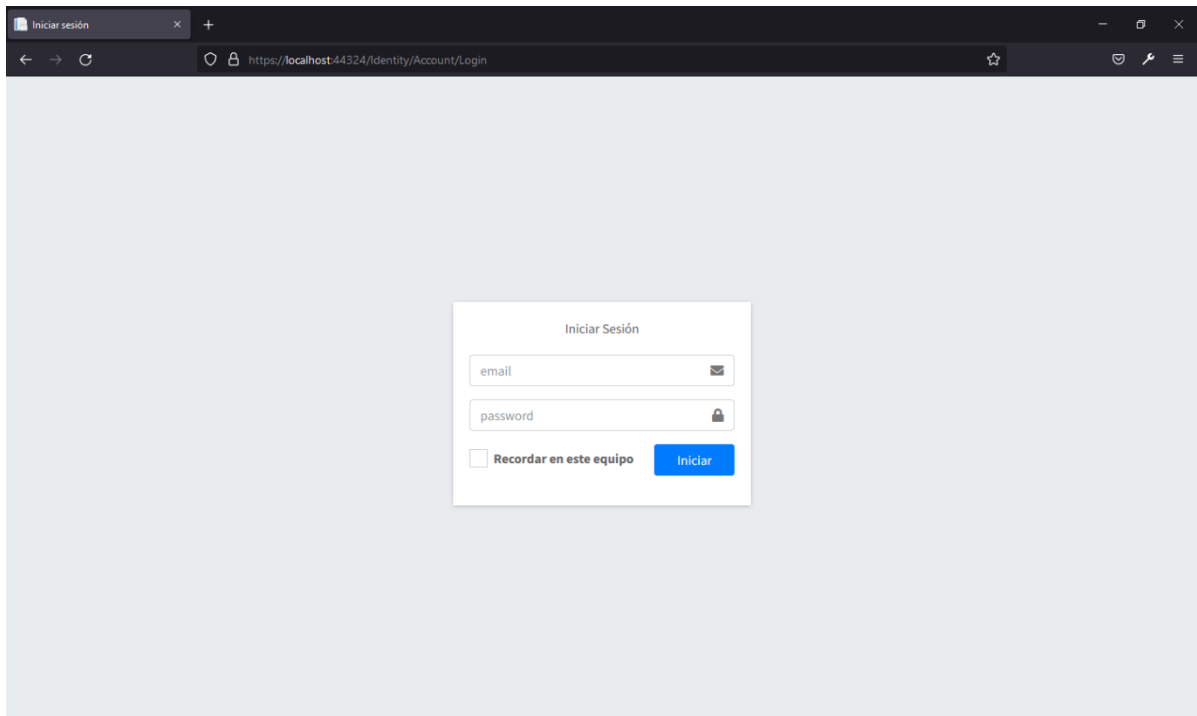
```
@page
@model LoginModel
@{
    Layout = null;
    ViewData["Title"] = "Iniciar sesión";
}
<!DOCTYPE html>
<html lang="es-es">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Iniciar sesión</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="~/plugins/fontawesome-free/css/all.min.css">
    <link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
    <link rel="stylesheet" href="~/plugins/ichex-bootstrap/ichex-bootstrap.min.css">
    <link rel="stylesheet" href="~/css/adminlte.min.css">
    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">
</head>
<body class="hold-transition login-page">
    <div class="login-box">
        <div class="login-logo"></div>
        <div class="card">
            <div class="card-body login-card-body">
                <p class="login-box-msg">Iniciar Sesión</p>
                <form id="account" method="post">
                    <span asp-validation-for="Input.Email" class="text-danger small"></span>
                    <div class="input-group mb-3">
                        <input asp-for="Input.Email" class="form-control" placeholder="email" />
                        <div class="input-group-append">
                            <div class="input-group-text">
                                <span class="fas fa-envelope"></span>
                            </div>
                        </div>
                    </div>
                    <span asp-validation-for="Input.Password" class="text-danger small"></span>
                    <div class="input-group mb-3">
                        <input asp-for="Input.Password" class="form-control" placeholder="password" />
                        <div class="input-group-append">
                            <div class="input-group-text">
                                <span class="fas fa-lock"></span>
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-8">
                            <div class="icheck-primary">
                                <input type="checkbox" id="remember">
                                <label for="remember">
                                    Recordar en este equipo
                                </label>
                            </div>
                        </div>
                        <div class="col-4">
                            <button type="submit" class="btn btn-primary btn-block">Iniciar</button>
                        </div>
                    </div>
                    <div asp-validation-summary="All" class="text-danger small"></div>
                </form>
            </div>
        </div>
    </div>
    <script src="~/plugins/jquery/jquery.min.js"></script>
    <script src="~/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/adminlte.min.js"></script>
    @section Scripts {
        <partial name="_ValidationScriptsPartial" />
    }
</body>
</html>
```

Registro: Register.cshtml

```
@page
@model RegisterModel
@{
    Layout = null;
    ViewData["Title"] = "Nuevo Usuario";
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Nuevo Usuario</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="~/plugins/fontawesome-free/css/all.min.css">
    <link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
    <link rel="stylesheet" href="~/plugins/ichack-bootstrap/ichack-bootstrap.min.css">
    <link rel="stylesheet" href="~/css/adminlte.min.css">
    <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">
</head>
<body class="hold-transition register-page">
    <div class="register-box">
        <div class="register-logo">
            <a href="~/Home"><b>Admin</b>LTE</a>
        </div>

        <div class="card">
            <div class="card-body register-card-body">
                <form asp-route-returnUrl="@Model.ReturnUrl" method="post">
                    <h4>Creando una nueva cuenta.</h4>
                    <hr />
                    <div class="form-group">
                        <label asp-for="Input.Email"></label>
                        <input asp-for="Input.Email" class="form-control" />
                        <span asp-validation-for="Input.Email" class="text-danger"></span>
                    </div>
                    <div class="form-group">
                        <label asp-for="Input.Password"></label>
                        <input asp-for="Input.Password" class="form-control" />
                        <span asp-validation-for="Input.Password" class="text-danger"></span>
                    </div>
                    <div class="form-group">
                        <label asp-for="Input.ConfirmPassword"></label>
                        <input asp-for="Input.ConfirmPassword" class="form-control" />
                        <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
                    </div>
                    <button type="submit" class="btn btn-primary">Registrarse</button>
                </form>
                <div class="social-auth-links text-center">
                    <p>- O BIEN -</p>
                    <a href="#" class="btn btn-block btn-primary">
                        <i class="fab fa-facebook mr-2"></i>
                        Regístrese mediante Facebook
                    </a>
                    <a href="#" class="btn btn-block btn-danger">
                        <i class="fab fa-google-plus mr-2"></i>
                        Regístrese mediante Google+
                    </a>
                </div>
                <a href="~/login.html" class="text-center">Ya soy un usuario registrado</a>
            </div>
        </div>
    </div>
    <script src="~/plugins/jquery/jquery.min.js"></script>
    <script src="~/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/adminlte.min.js"></script>
    @section Scripts {
        <partial name="_ValidationScriptsPartial" />
    }
</body>
</html>
```

Estas son las páginas resultantes. Bastante limpio, ¿Verdad?



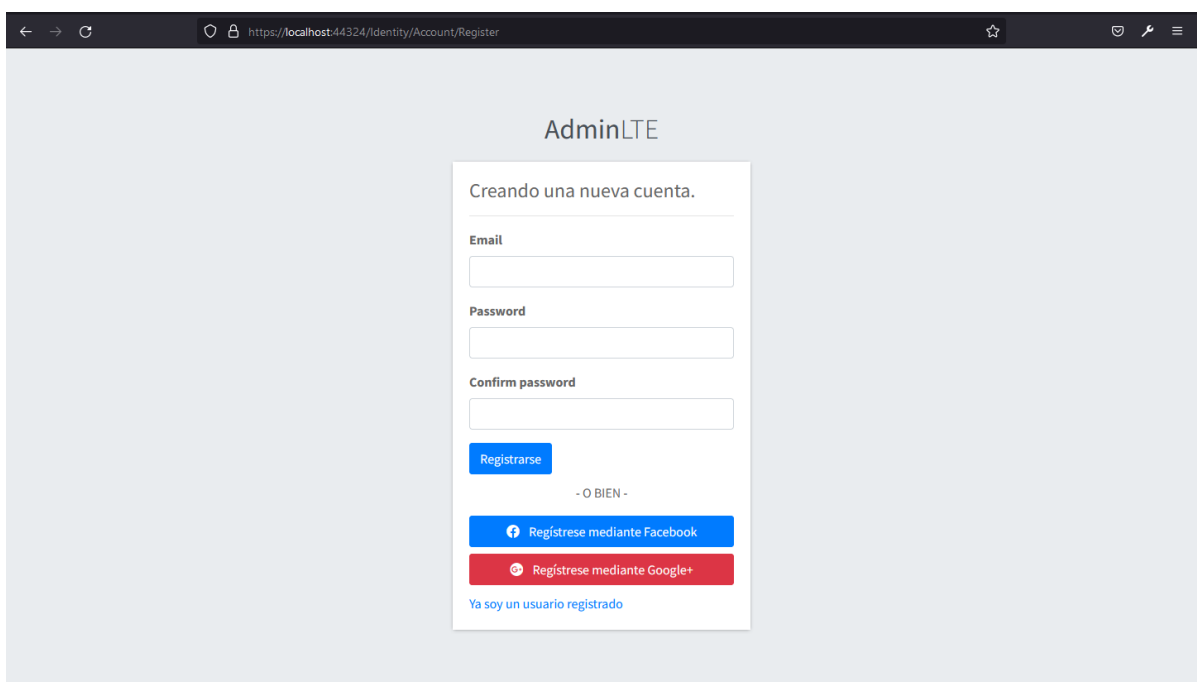
Iniciar Sesión

email

password

☐ Recordar en este equipo

Iniciar



AdminLTE

Creando una nueva cuenta.

Email

Password

Confirm password

Registrarse

- O BIEN -

Regístrese mediante Facebook

Regístrese mediante Google+

[Ya soy un usuario registrado](#)

14. Habilitación de la autenticación

Ahora que tenemos nuestras páginas de identidad listas, configuremos nuestra aplicación ASP.NET Core para habilitar la autenticación.

Protegeremos todos los métodos del controlador de forma predeterminada. Según nuestro requisito, debemos permitir que cualquier visitante aleatorio use el método Inicio/Índice.

Navigate hasta el método Startup.cs/ConfigureServices y agregue estas líneas al final del método:

```
services.AddMvc(o => {  
    var policy = new AuthorizationPolicyBuilder()  
        .RequireAuthenticatedUser()  
        .Build();  
    o.Filters.Add(new AuthorizeFilter(policy));  
});
```

Esto agregará una nueva política a la aplicación ASP.NET Core MVC de que cada método necesita un usuario autenticado, a menos que lo definamos como [AllowAnonymous].

Ahora, vaya a Home Controller y agregue [AllowAnonymous] arriba del método Index. Esto significa que cualquiera puede acceder al método.

15. Menú basado en estado autenticado

Bien, ahora que hemos habilitado la seguridad, no queremos que el usuario anónimo vea las opciones privadas en la barra de navegación, en nuestro caso el enlace Privacidad. Para ocultar el artículo, vaya a _MainNavigation.cshtml.

Aquí está la lógica simple. Tendremos que agregar una condición (si) para verificar si el usuario actual está autenticado, en caso afirmativo, mostraremos los enlaces privados de la aplicación. En caso contrario únicamente los que tengan permitido el acceso anónimo. Así es como se codifica.

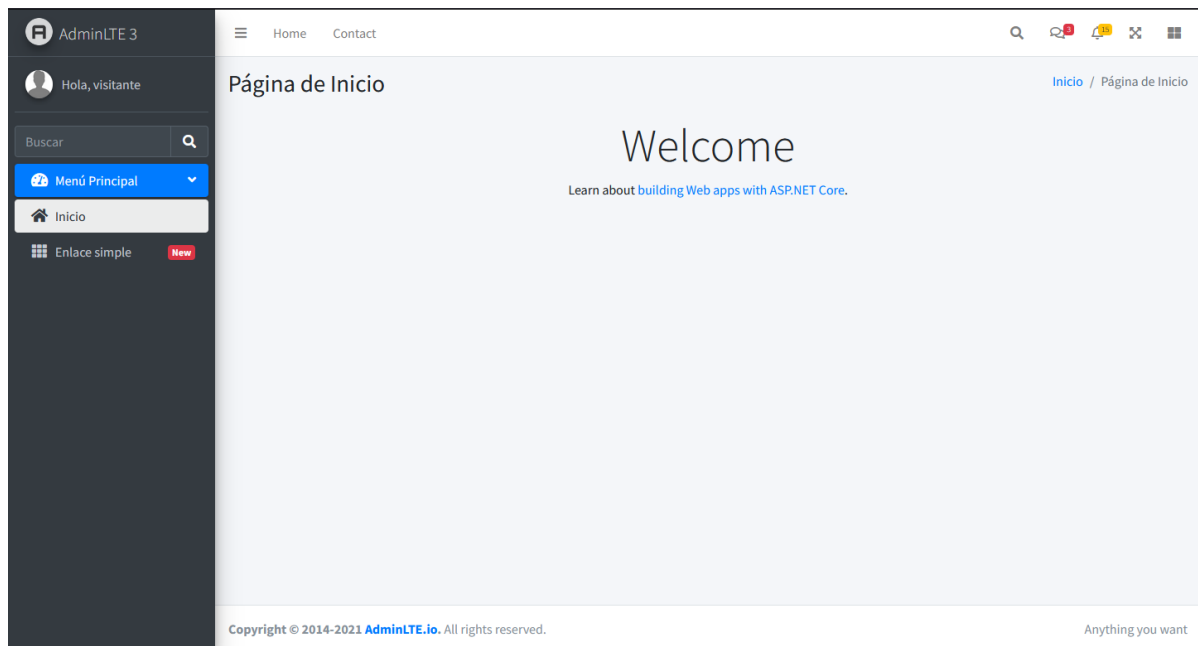
```
@if (User.Identity.IsAuthenticated) {  
    <li class="nav-item">  
        <a asp-controller="Home" asp-action="Privacy"  
            class="nav-link" @Url.MakeActiveClass("home", "privacy")">  
            <i class="nav-icon fa fa-lock"></i>  
            <p>Privacidad</p>  
        </a>  
    </li>  
}
```

Ahora, hay una cosa más que podemos agregar aquí usando la verificación condicional de autenticación. En el lateral, ¿ves una imagen aleatoria con un nombre aleatorio? ¿Tiene sentido agregar una condición similar allí? ¿Sí claro? Entonces, lo que haremos será mostrar el nombre del usuario autenticado. Si no está autenticado, mostremos un "Hola, visitante".

Modifique `_MainNavigation.cshtml` como se muestra a continuación.

```
@if (User.Identity.IsAuthenticated) {
    <div class="user-panel mt-3 pb-3 mb-3 d-flex">
        <div class="image">
            
        </div>
        <div class="info">
            <a href="#" class="d-block">Hola, @User.Identity.Name</a>
        </div>
    </div>
} else {
    <div class="user-panel mt-3 pb-3 mb-3 d-flex">
        <div class="image">
            
        </div>
        <div class="info">
            <a href="#" class="d-block">Hola, visitante</a>
        </div>
    </div>
}
```

Compilamos y ejecutamos la aplicación y obtendremos lo siguiente:



Creo que el resultado salta a la vista, ¿No? Bien, para finalizar agreguemos los enlaces del inicio de sesión y de registro en el menú superior.

```
@if (User.Identity.IsAuthenticated) {
    <li class="nav-item d-none d-sm-inline-block">
        <form class="form-inline asp-area="Identity" asp-page="/Account/Logout"
            asp-route-returnUrl="@Url.Action("Index", "Home", new { area = "" })">
            <button type="submit" class="nav-link btn btn-link text-dark">Logout</button>
        </form>
    </li>
} else {
    <li class="nav-item d-none d-sm-inline-block">
        <a asp-area="Identity" asp-page="/Account/Login" class="nav-link">Iniciar Sesión</a>
    </li>
    <li class="nav-item d-none d-sm-inline-block">
        <a asp-area="Identity" asp-page="/Account/Register" class="nav-link">Registrarse</a>
    </li>
}
```

Y con esto ya está todo completado para iniciar nuestra aplicación integrada perfectamente con AdminLTE. Ahora vamos a comprobar si todo funciona bien. Registremos una nueva cuenta.

A database operation failed while processing the request.

SqlException: Cannot open database "aspnet-CursoMVCCore-AEB50072-D732-47A3-9B0C-C4C09B55F695" requested by the login. The login failed. Login failed for user 'BREE\Desarrollo'.

Applying existing migrations may resolve this issue

There are migrations that have not been applied to the following database(s):

ApplicationDbContext

- 0000000000000000_CreatelIdentitySchema

Apply Migrations

In Visual Studio, you can use the Package Manager Console to apply pending migrations to the database:

PM> Update-Database

Alternatively, you can apply pending migrations from a command prompt at your project directory:

```
> dotnet ef database update
```

¡Ups! ¿Qué ha pasado? ¡Ah ya se! tenemos que aplicar las migraciones todavía. Al registrarse por primera vez, Visual Studio le informa sobre esto.

Haga clic en Aplicar Migraciones. Una vez hecho esto, recibirá un mensaje que dice: "Intente actualizar la página"... Recarga la página.

Register confirmation

This app does not currently have a real email sender registered, see [these docs](#) for how to configure a real email sender. Normally this would be emailed:

[Click here to confirm your account](#)

En esta página, haga clic en confirmar su cuenta. Luego navegue a localhost: xxx/Home

Y ya puede iniciar sesión con las credenciales que ha introducido.

16. Corrección del complemento filterzr de AdminLTE

Desde la versión 3.0.5 de AdminLTE 3.0.5 se ha podido apreciar que al compilar la aplicación se producida una excepción parecida a esto.

Build: The module `"/FilterizrOptions/defaultOptions"` has no exported member `'RawOptionsCallbacks'`. Did you mean `'import RawOptionsCallbacks from "/FilterizrOptions/defaultOptions"'` instead.

Con AdminLTE 3.0.5, el complemento filterizr, por alguna razón, todavía tiene sus propios archivos de código fuente, lo que puede causar errores de TypeScript en Visual Studio si intenta ejecutar el proyecto. Si enfrenta un problema similar, elimine cuidadosamente el plugin de su solución, o elimine todo en la carpeta filterizr excepto los siguientes archivos:

- filterizr.min.js
- jquery.filterizr.min.js
- vainilla.filterizr.min.js

17. Resumen

En esta guía detallada para principiantes, hemos aprendido Integración de AdminLTE con ASP.NET Core 5 MVC. Y hemos cubierto varios temas, incluidos diseños, vistas, vistas parciales, autenticación, identidad, indicador de navegación y mucho más.

Siguiendo los contenidos de esta guía vamos a poder crear nuestras propias aplicaciones con un aspecto muy profesional desde el primer día y lo que hemos aprendido nos servirá de base para integrar otras plantillas de Bootstrap diferentes a la que ha sido objeto de este documento.

