

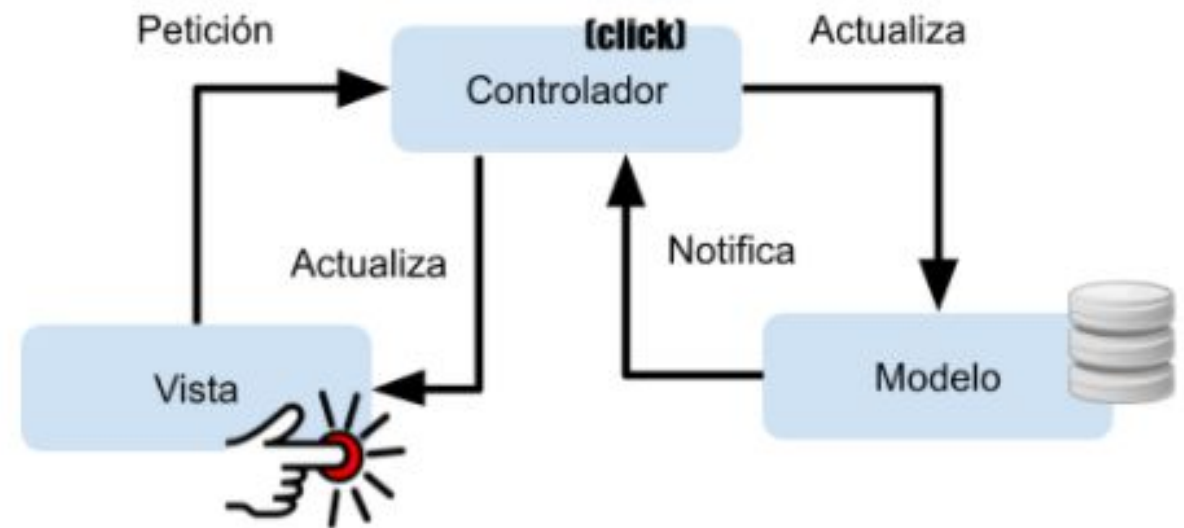
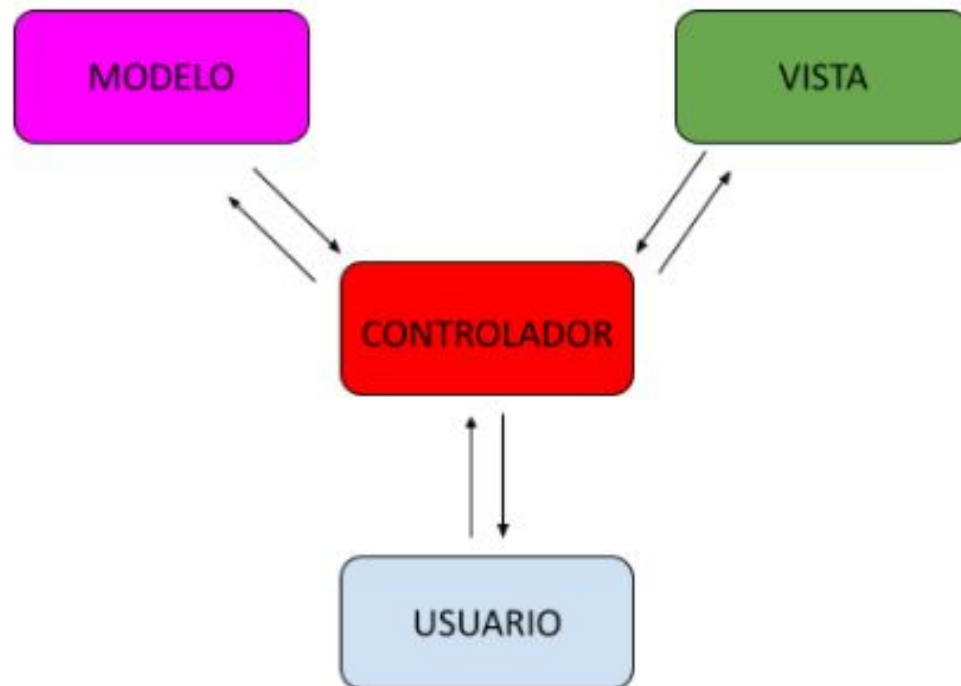


MVC

Model View Controller

Fundamentos del Modelo Vista Controlador MVC

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.



Fundamentos del Modelo Vista Controlador MVC

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La **Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste.
- El **Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".

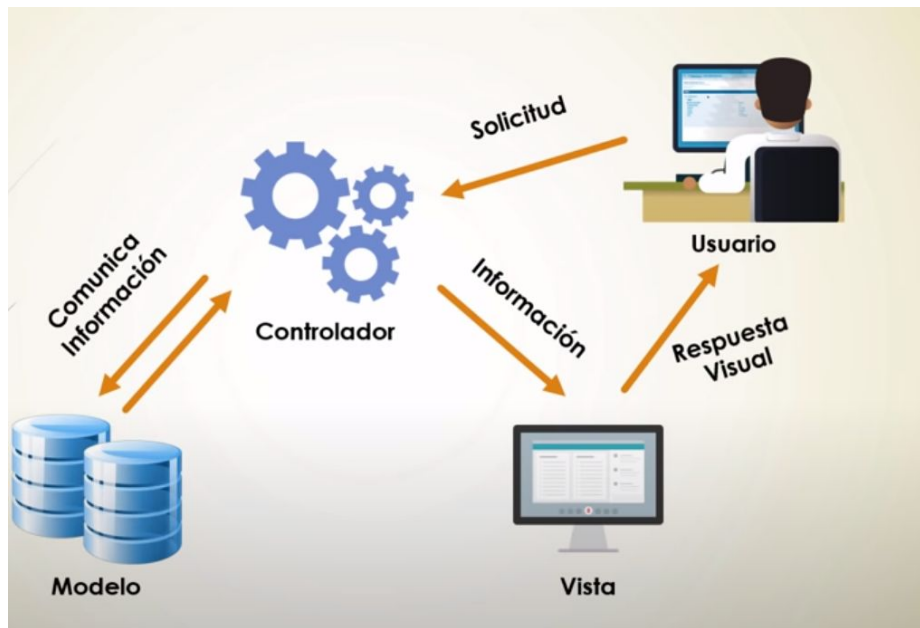
El controlador

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.
- Cada nombre de clase controller debe terminar con la palabra "Controller". Además, cada clase controller debe estar ubicada en la carpeta Controller de la estructura de carpetas MVC.

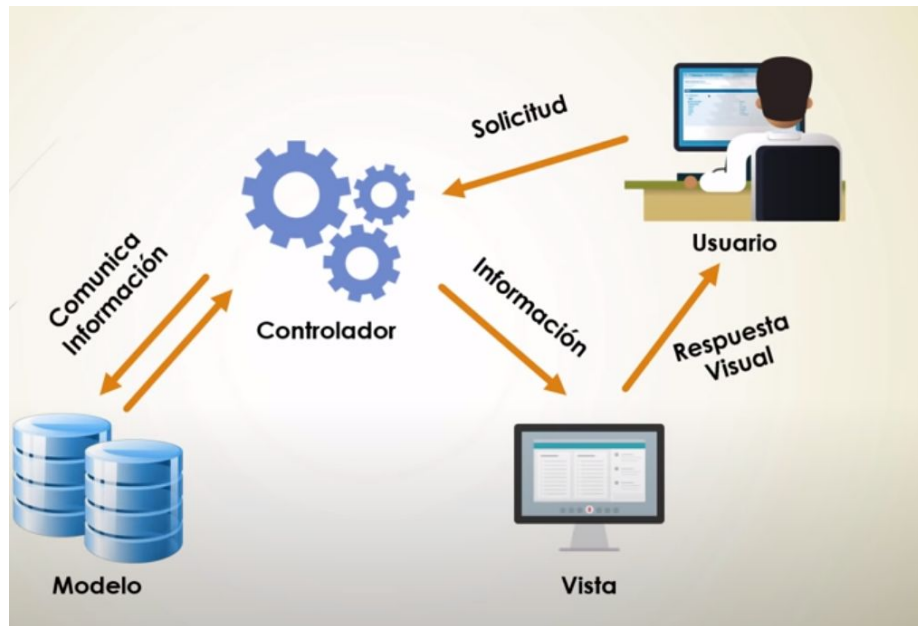
Las vistas son responsables de:

- Recibe el modelo (a través del controlador) y lo muestra al usuario en un formato adecuado.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

El flujo que sigue el control generalmente es el siguiente:



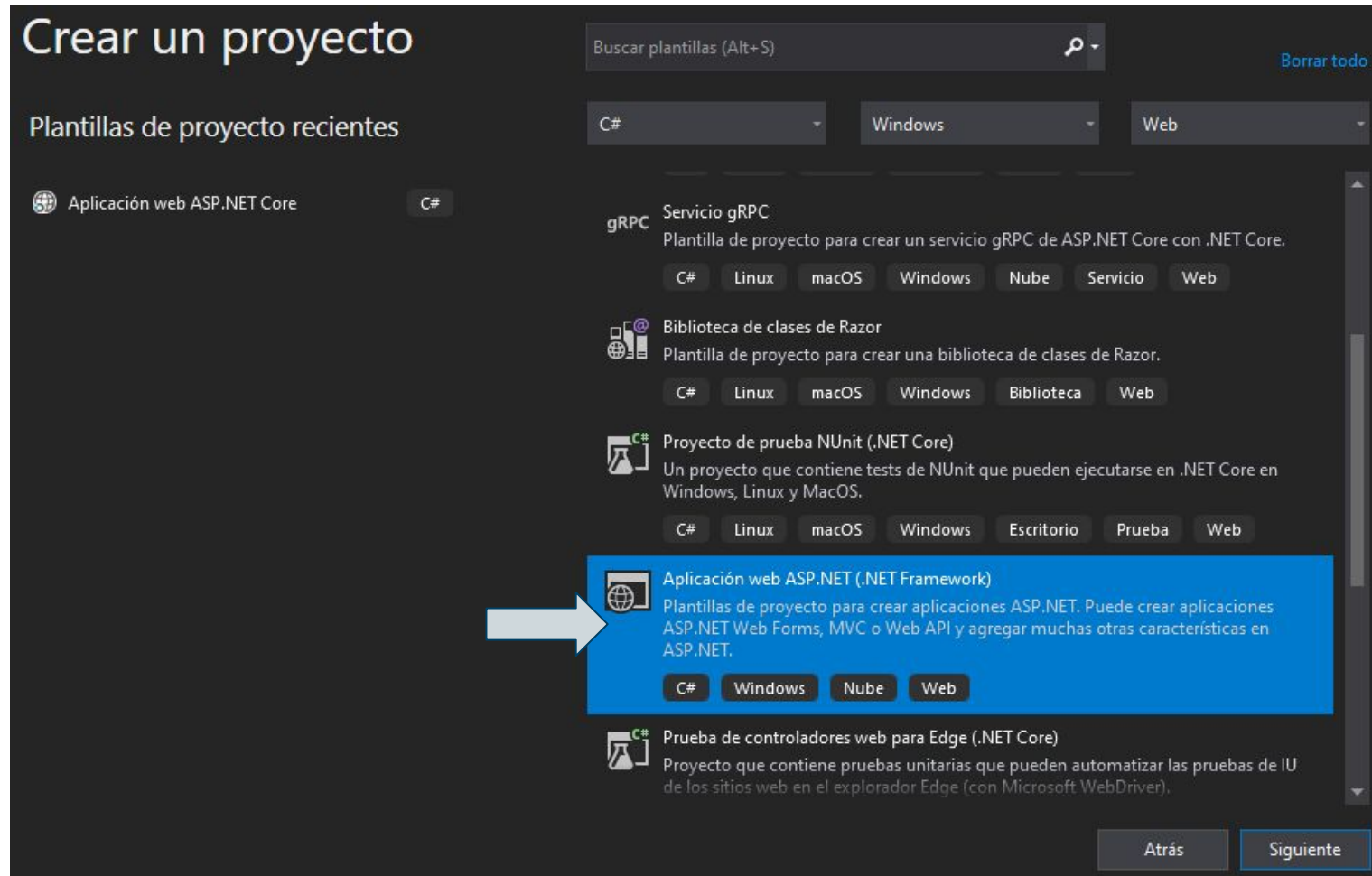
1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.



4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Crear aplicación ASP.NET MVC

En Microsoft Visual Studio **Community 2019** y .NET Framework 4.7.2



Crear aplicación ASP.NET MVC

Configure su nuevo proyecto

Aplicación web ASP.NET (.NET Framework) C# Windows Nube Web

Nombre del proyecto

Programa CLIP_2020_1

Ubicación

C:\Users\Luis\source\repos

Nombre de la solución ⓘ

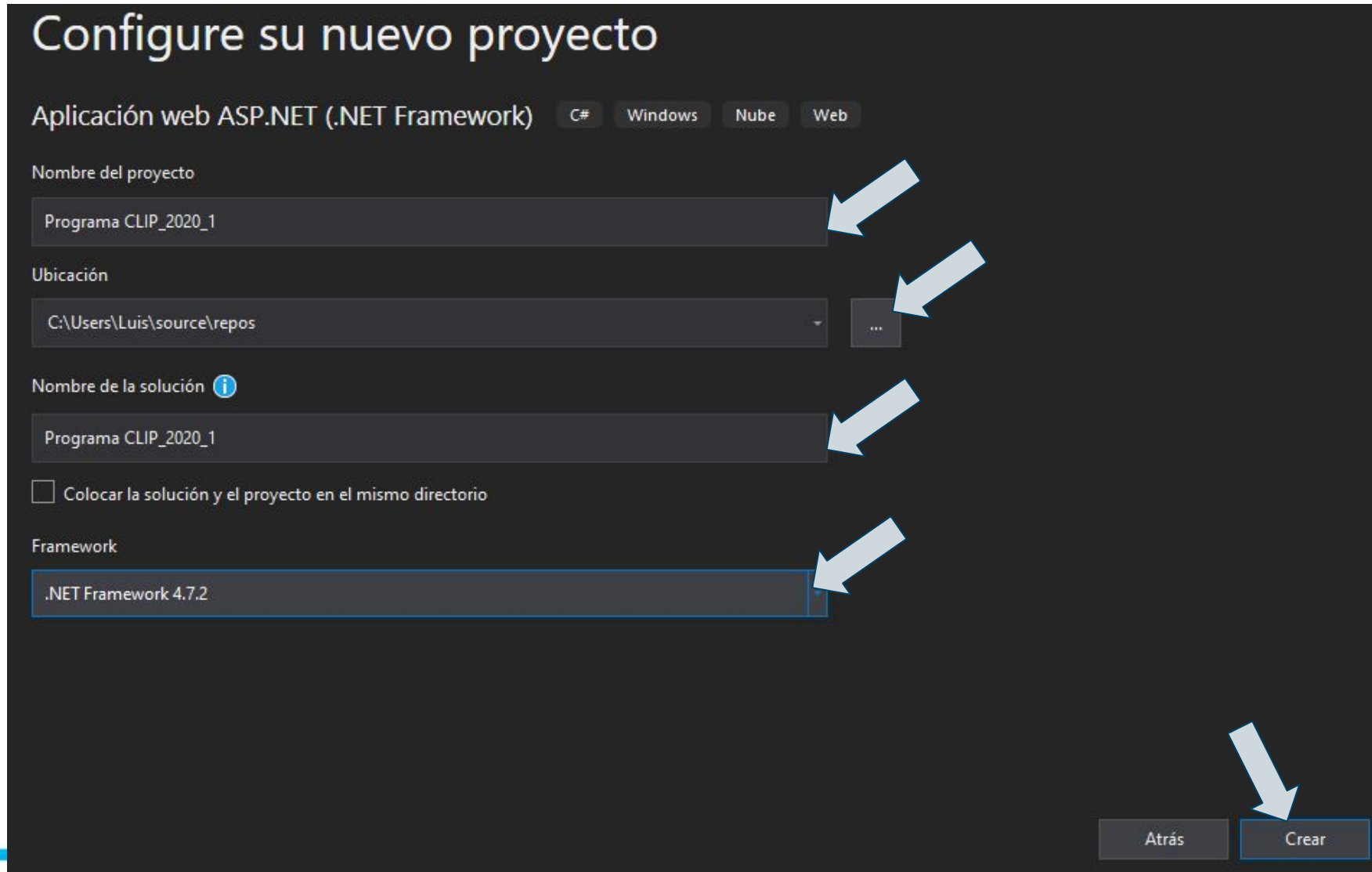
Programa CLIP_2020_1

☐ Colocar la solución y el proyecto en el mismo directorio

Framework

.NET Framework 4.7.2

Atrás Crear



Crear aplicación ASP.NET MVC

Crear una aplicación web ASP.NET



Vacío

Una plantilla de proyecto vacía para crear aplicaciones ASP.NET. Esta plantilla no tiene contenido.



Web Forms

Una plantilla de proyecto para crear aplicaciones de ASP.NET Web Forms. ASP.NET Web Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlados por la interfaz de usuario y con acceso a datos.



MVC

Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.



API web

Plantilla de proyecto para crear servicios HTTP REST que pueden llegar a una amplia gama de clientes, como, por ejemplo, exploradores y dispositivos móviles.



Aplicación de página única

Una plantilla de proyectos para crear aplicaciones HTML5 atractivas controladas por JavaScript del lado cliente mediante ASP.NET Web API. Las aplicaciones de una sola página proporcionan una experiencia de usuario atractiva que incluye interacciones del lado cliente mediante HTML5, CSS3 y JavaScript.

Autenticación

Sin autenticación

[Cambiar](#)

Agregar carpetas y referencias principales

- ☐ Formularios Web Forms
- ☒ MVC
- ☐ API web

Avanzado

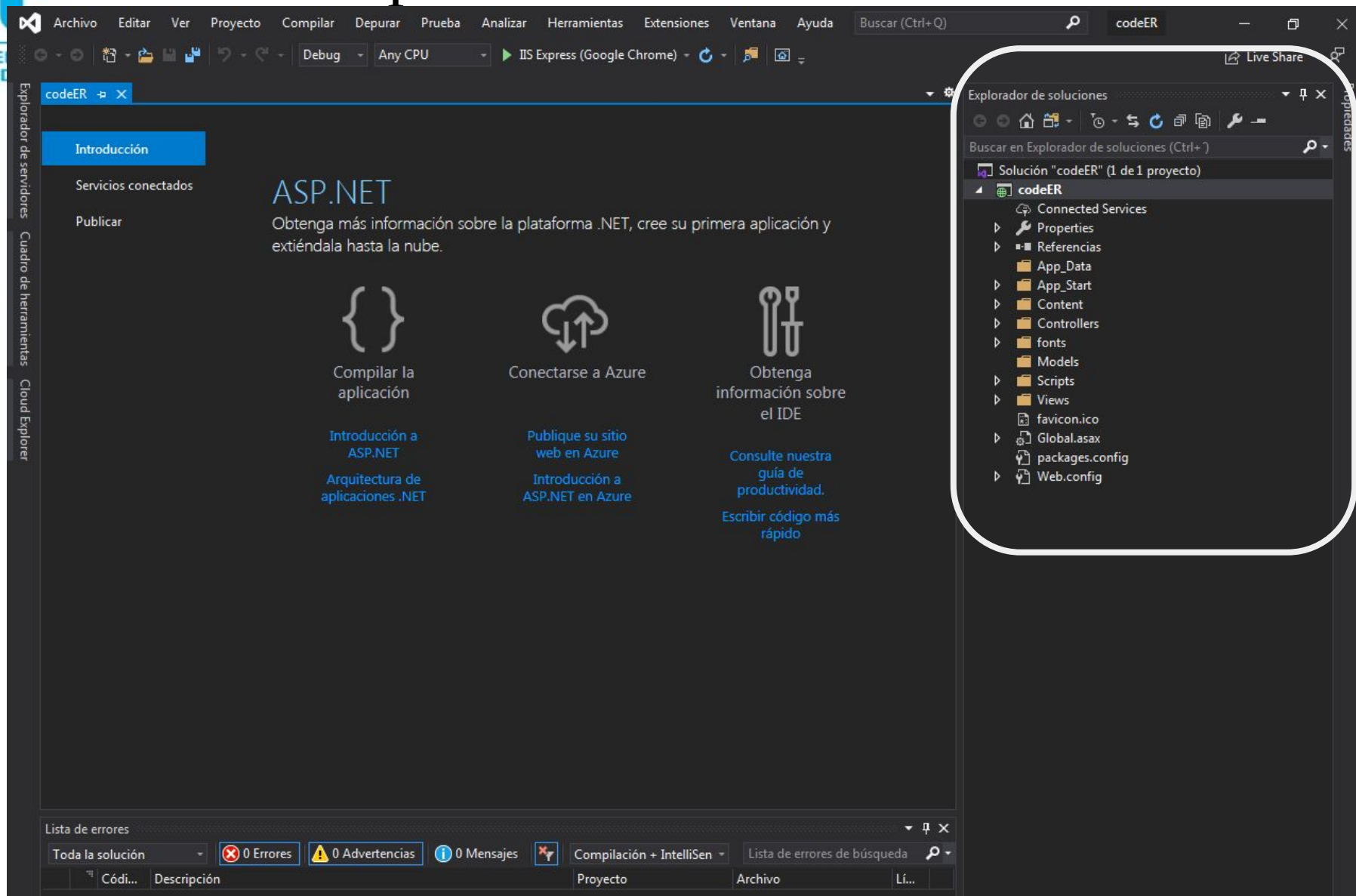
- ☒ Configurar para HTTPS
- ☐ Compatibilidad con Docker
(Requiere [Docker Desktop](#))
- ☐ Crear también un proyecto para pruebas unitarias

Programa CLIP_2020_1.Tests

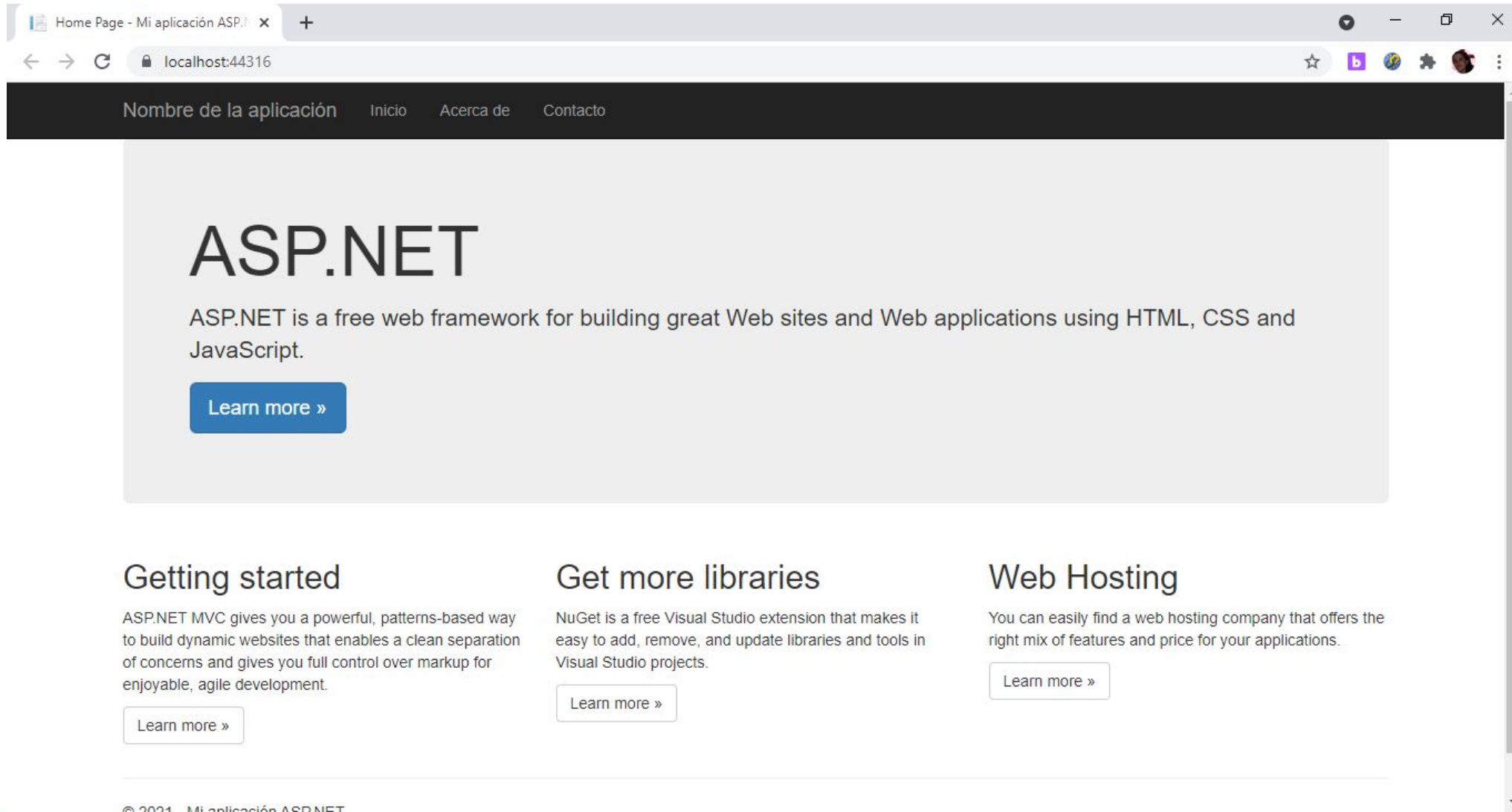
Atrás

Crear

Crear aplicación ASP.NET MVC

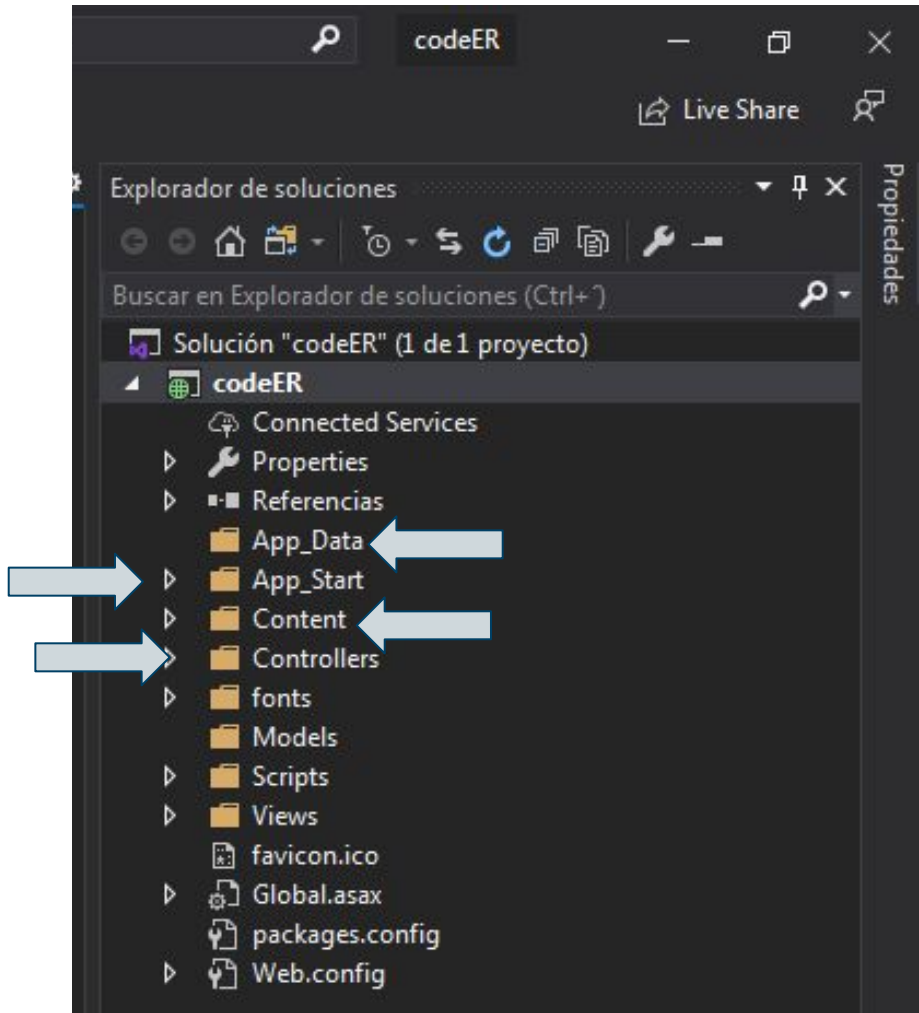


Aplicación creada y ejecutada en local

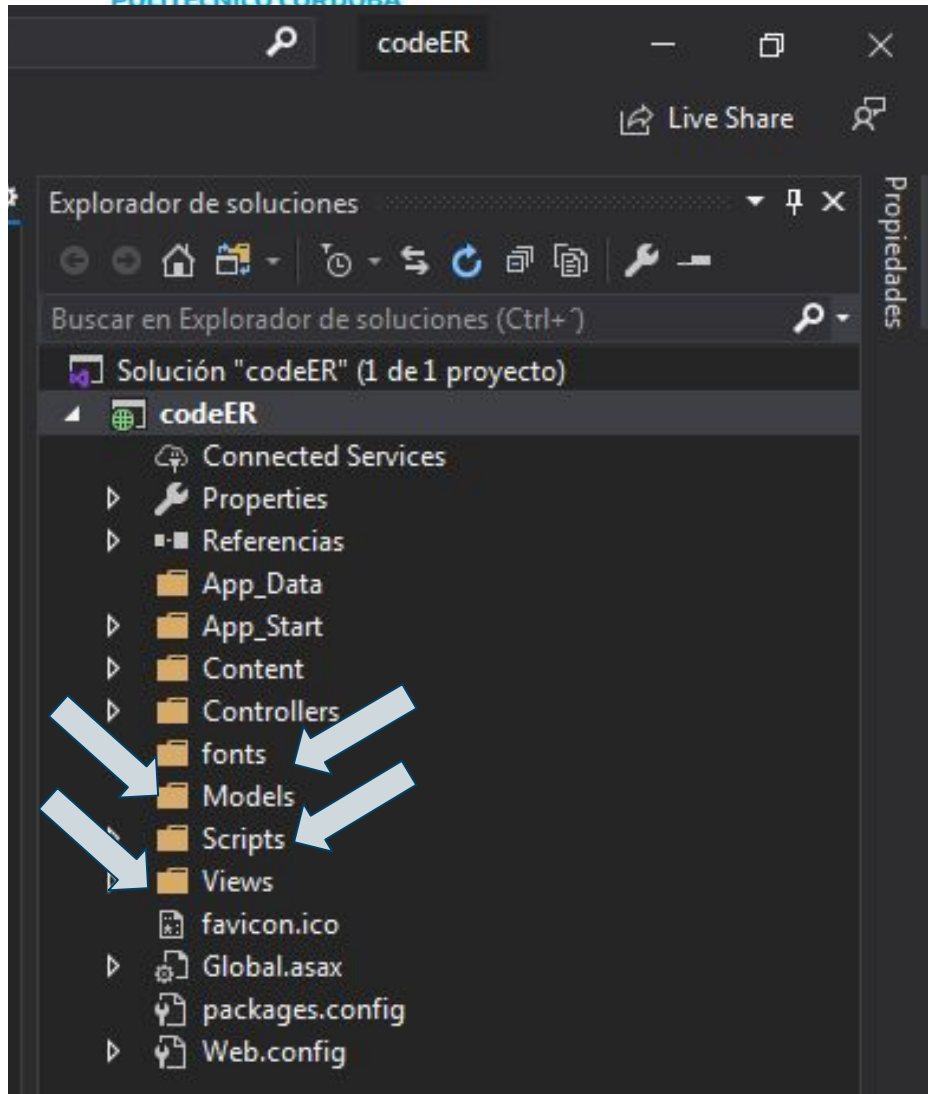


DEMO

Estructura de carpetas ASP.NET MVC



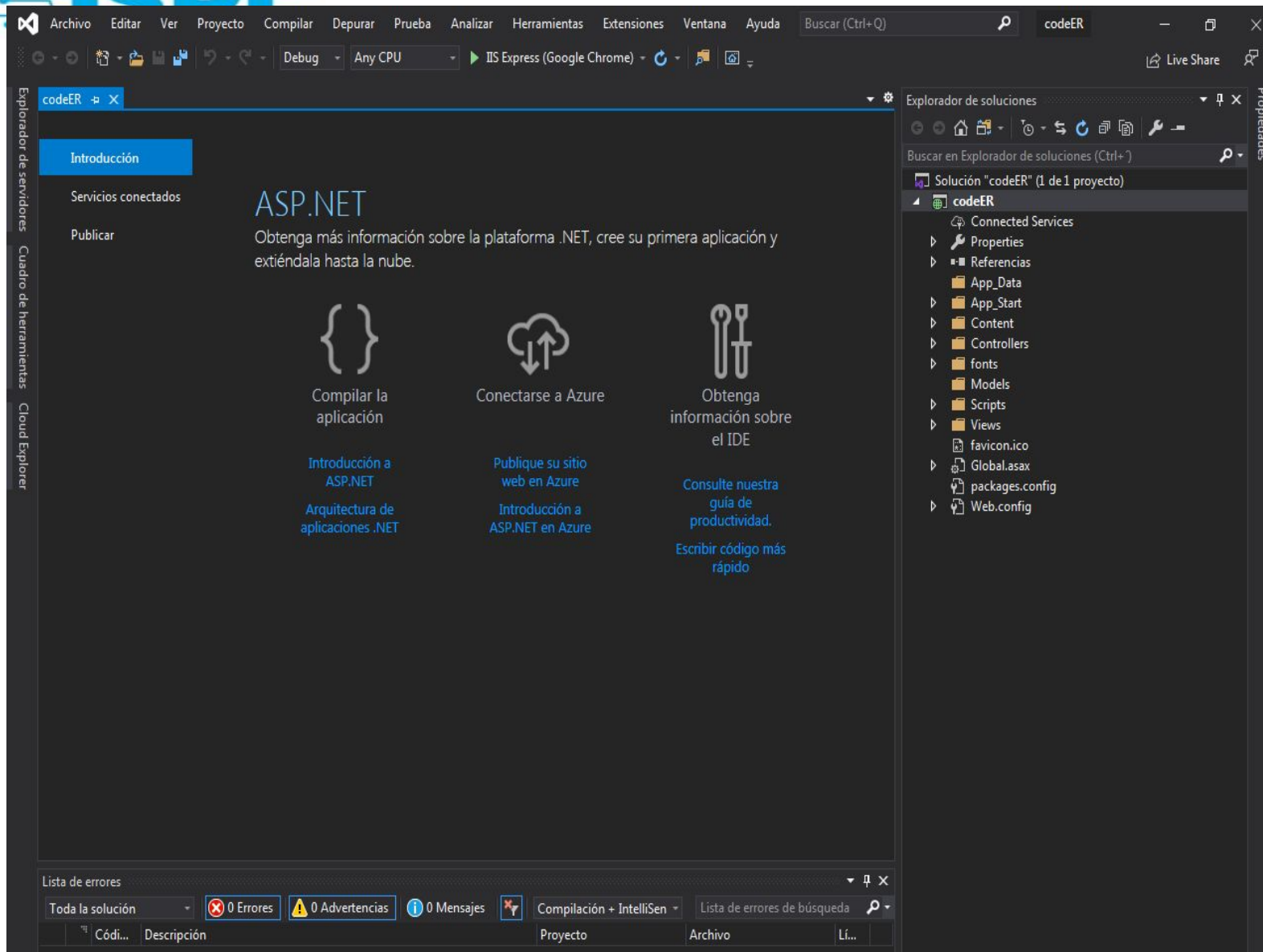
- La carpeta **App_Data** puede contener archivos de datos de aplicaciones como LocalDB, archivos .mdf, archivos XML y otros archivos relacionados con datos.
- La carpeta **App_Start** puede contener archivos de clase que se ejecutarán cuando se inicie la aplicación. Normalmente, estos serían archivos de configuración como AuthConfig.cs, BundleConfig.cs, FilterConfig.cs, RouteConfig.cs, etc. MVC 5 incluye BundleConfig.cs, FilterConfig.cs y RouteConfig.cs de forma predeterminada.
- La carpeta **Content** contiene archivos estáticos como archivos CSS, imágenes y archivos de iconos. La aplicación MVC 5 incluye bootstrap.css, bootstrap.min.css y Site.css de forma predeterminada.
- La carpeta **Controllers** contiene archivos de clase para los controladores. El Controlador maneja la solicitud de los usuarios y devuelve una respuesta. MVC requiere que el nombre de todos los archivos del controlador termine con "Controller".



- La carpeta de **fonts** contiene archivos de fuentes personalizados para su aplicación.
- La carpeta **Models** contiene archivos de clases de modelos. Normalmente, la clase de modelo incluye propiedades públicas, que serán utilizadas por la aplicación para almacenar y manipular los datos de la aplicación.
- La carpeta **Scripts** contiene archivos JavaScript o VBScript para la aplicación. MVC 5 incluye archivos javascript para bootstrap, jquery 1.10 y modernizador de forma predeterminada.
- La carpeta **Views** contiene archivos HTML para la aplicación. Normalmente, el archivo de visualización es un archivo .cshtml en el que se escribe código HTML y C # o VB.NET.

La carpeta Views incluye una carpeta separada para cada controlador. Por ejemplo, todos los archivos .cshtml, que serán procesados por HomeController estarán en Views> Home.

La carpeta Shared en la carpeta Views contiene todas las vistas compartidas entre diferentes controladores, por ejemplo, archivos de diseño.



Archivos de configuración:

Global.asax

El archivo Global.asax le permite escribir código que se ejecuta en respuesta a eventos de nivel de aplicación, como Application_BeginRequest, application_start, application_error, session_start, session_end, etc.

Packages.config

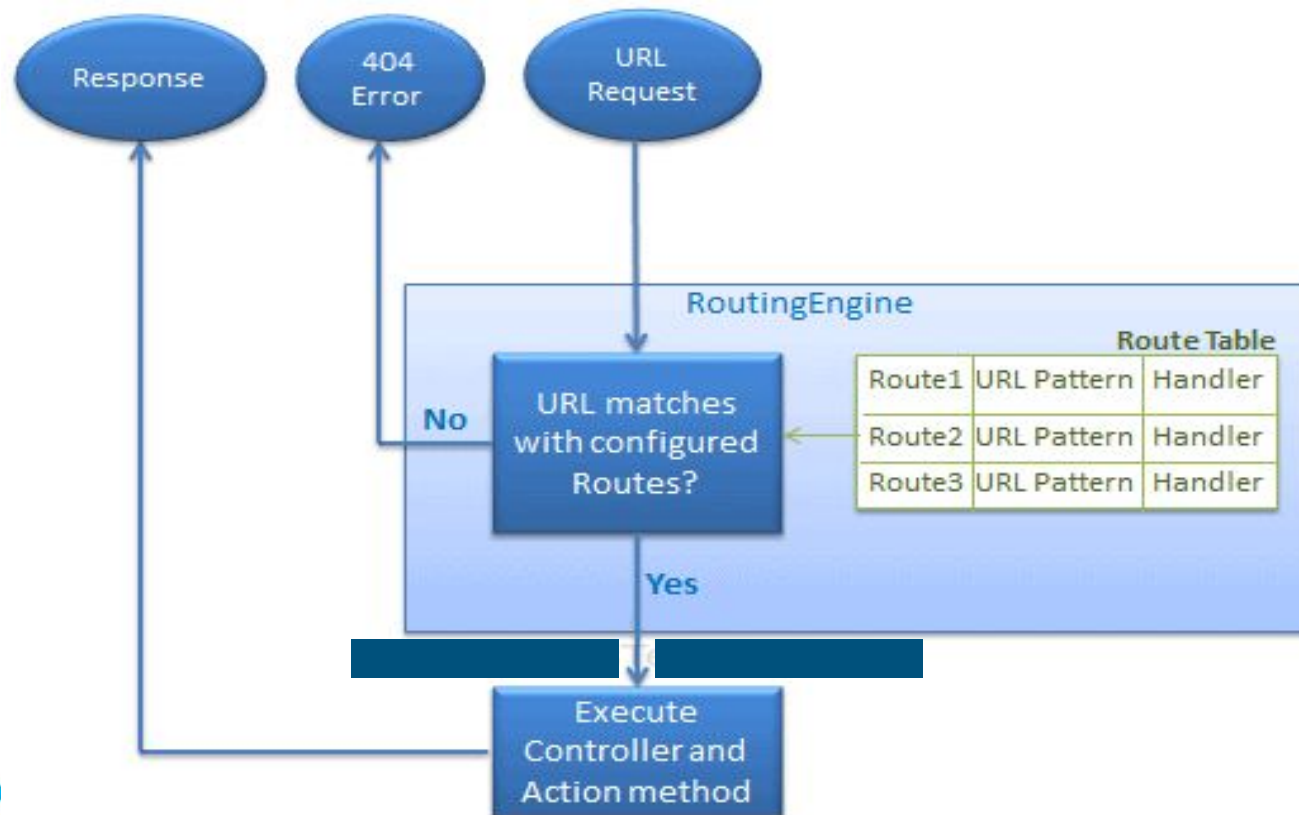
NuGet administra el archivo Packages.config para realizar un seguimiento de los paquetes y versiones que ha instalado en la aplicación.

Web.config

El archivo Web.config contiene configuraciones a nivel de aplicación.

Enrutamiento

ASP.NET introdujo el enrutamiento para eliminar la necesidad de mapear cada URL con un archivo físico. El enrutamiento nos permite definir un patrón de URL que se asigna al controlador de solicitudes. Este controlador de solicitudes en MVC, es la clase controller y el action method. Por ejemplo, `http: // dominio / estudiantes` se puede asignar al método `Index action` de MVC.



La ruta define el patrón de URL y la información del controlador. Todas las rutas configuradas de una aplicación se almacenan en `RouteTable` y serán utilizadas por el motor de enrutamiento para determinar la clase de controlador apropiado o el archivo para una solicitud entrante.

Configurar Ruta

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

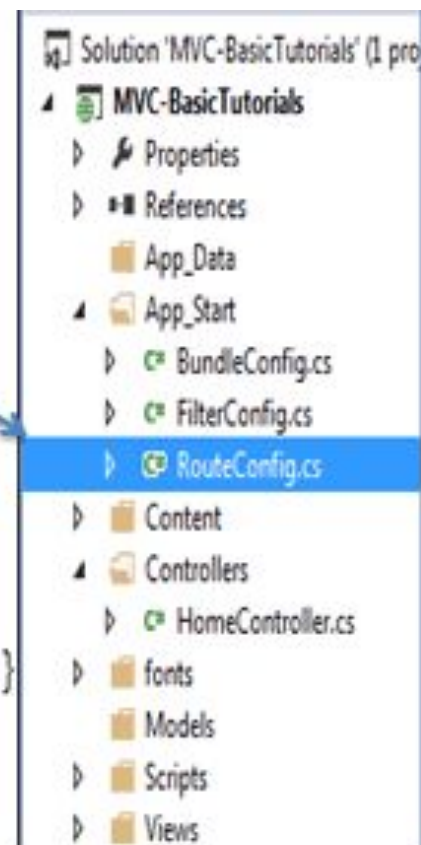
Route to ignore

Route name

URL Pattern

Defaults for Route

RouteConfig.cs



Patrón URL

Controller *Action method*
`http://localhost:1234/home/index/100` *Id parameter value*

Controller *Action method*
`http://localhost:1234/home/index`

URL	Controller	Action	Id
http://localhost/home	HomeController	Index	null
http://localhost/home/index/123	HomeController	Index	123
http://localhost/home/about	HomeController	About	null
http://localhost/home/contact	HomeController	Contact	null
http://localhost/student	StudentController	Index	null
http://localhost/student/edit/123	StudentController	Edit	123

Múltiples Rutas

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Student",
            url: "students/{id}",
            defaults: new { controller = "Student", action = "Index" }
        );

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
        );
    }
}
```

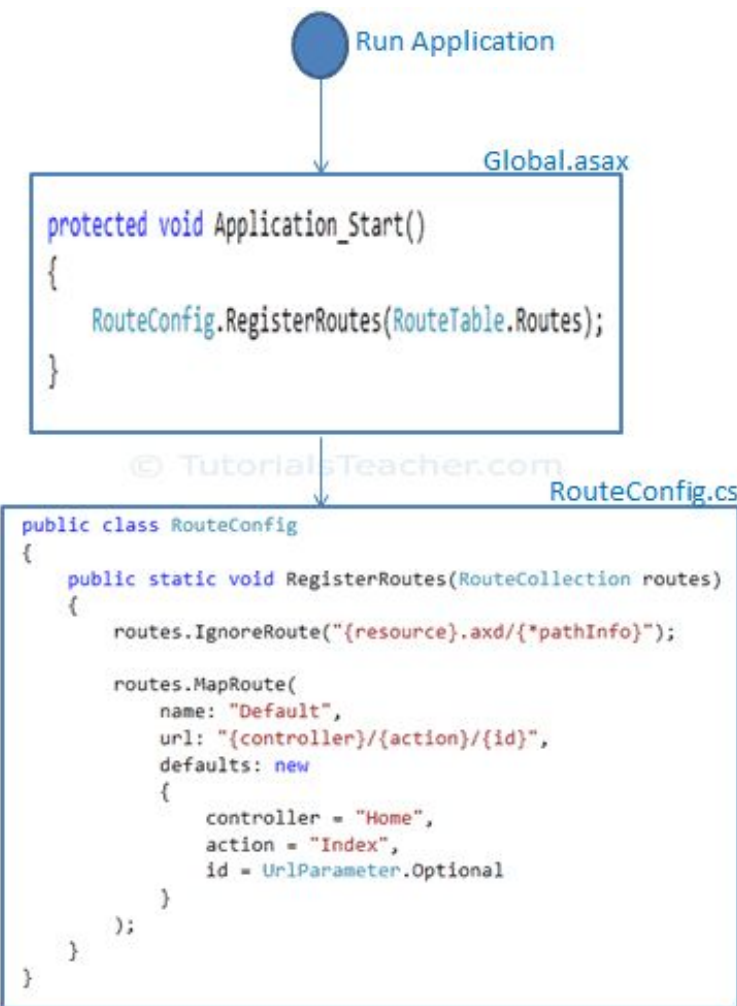
Registrar Rutas

Ahora, después de configurar todas las rutas en la clase RouteConfig, también debe registrarlo en el evento Application_Start() del Global.asax para que incluya todas sus rutas en el RouteTable.

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        RouteConfig.RegisterRoutes(RouteTable.Routes);
    }
}
```

Enrutamiento

1. El enrutamiento juega un papel importante en el marco MVC. El enrutamiento asigna la URL a una clase controller.
2. La ruta contiene información del controlador y del patrón de URL. El patrón de URL comienza después del nombre de dominio.
3. Las rutas se pueden configurar en la clase RouteConfig. También se pueden configurar varias rutas personalizadas.
4. Las restricciones de ruta aplican restricciones sobre el valor de los parámetros.
5. La ruta debe registrarse en el evento Application_Start en el archivo Global.ascx.cs.



Repaso de Controller

1. El controlador maneja las solicitudes de URL entrantes. El enrutamiento MVC envía solicitudes al controlador apropiado y al action method según la URL y las rutas configuradas.
2. Todos los métodos públicos de la clase Controller se denominan métodos de acción o action methods.
3. El nombre de la clase de controlador debe terminar con "Controller".

