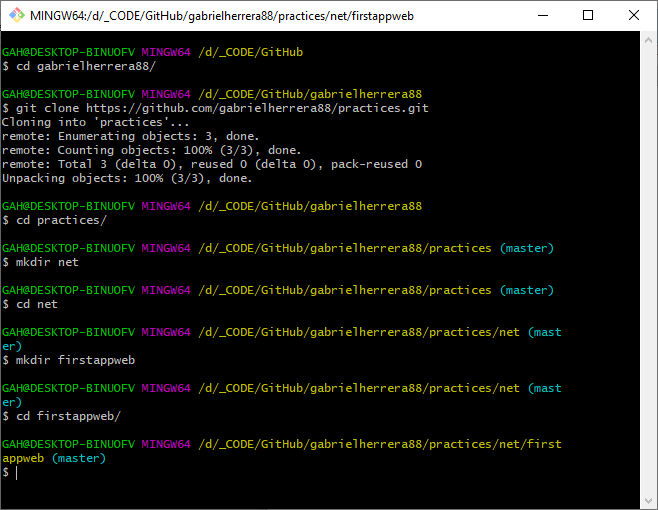
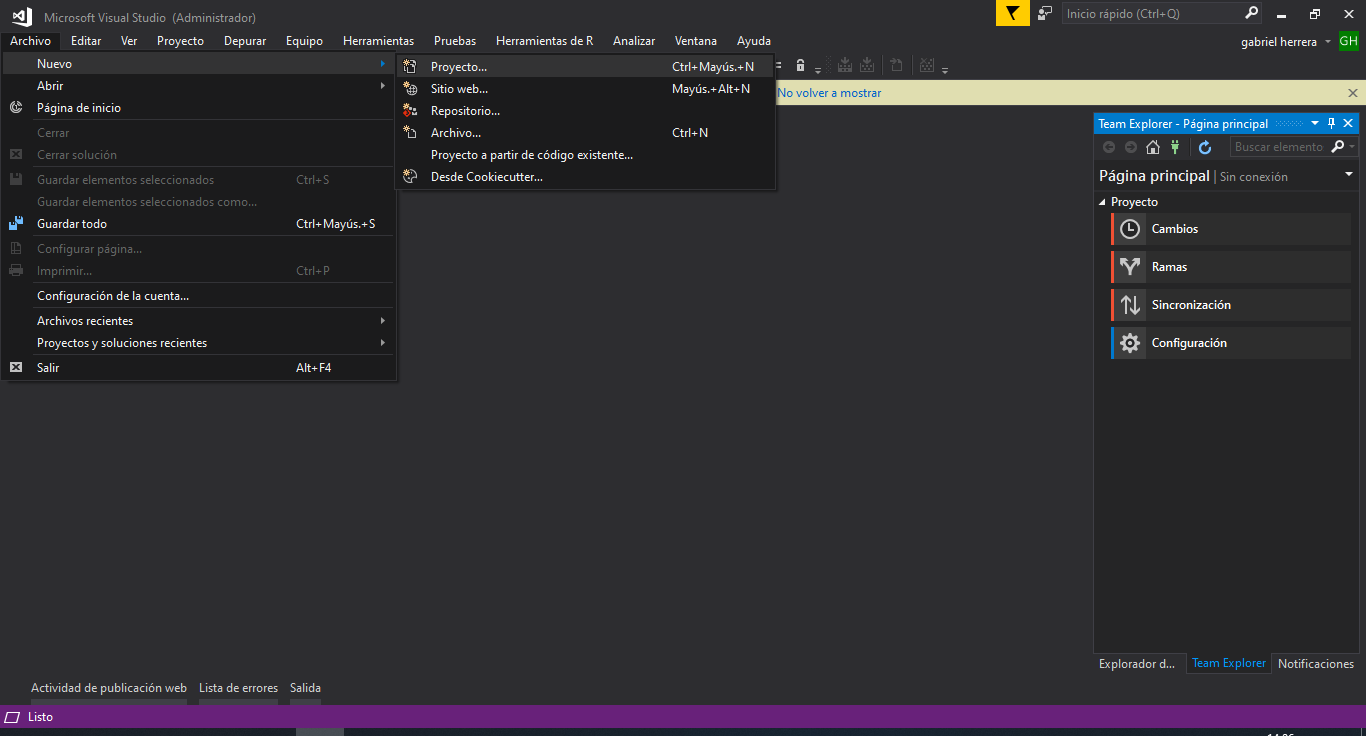
**Armando el Entorno:**

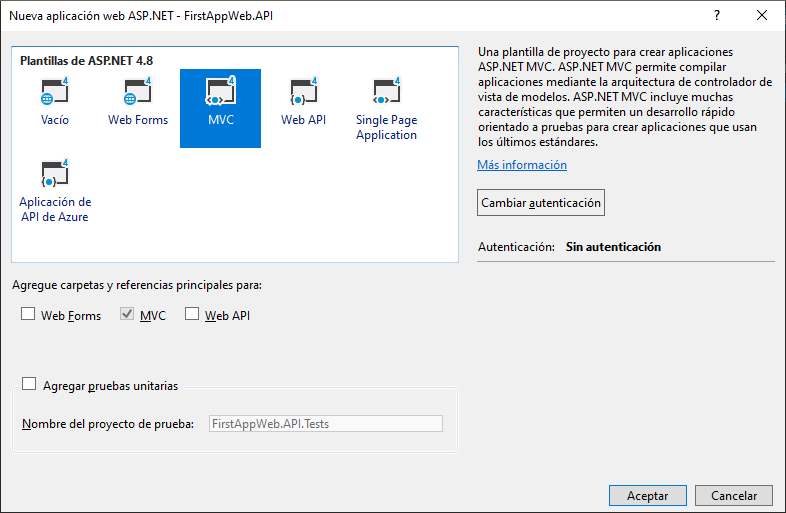
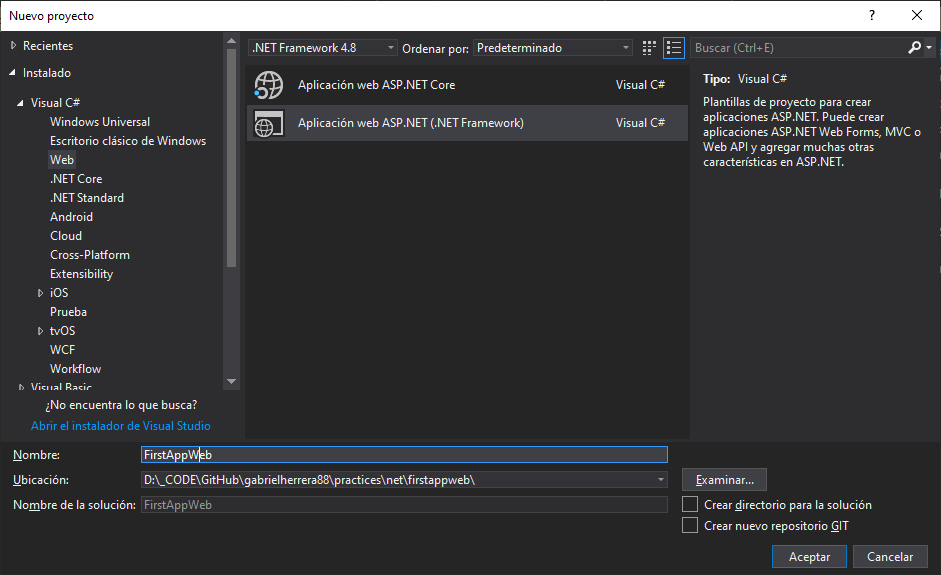
Consola Git Bash, clonamos el repo y creamos la estructura de directorio de nuestra primera app.



Creamos la solución y agregamos el primer proyecto. Una solución puede contener varios proyectos, cada proyecto responde a resolver un tema específico, que puede ser un cliente web, una capa de servicios, acceso a datos, etc.

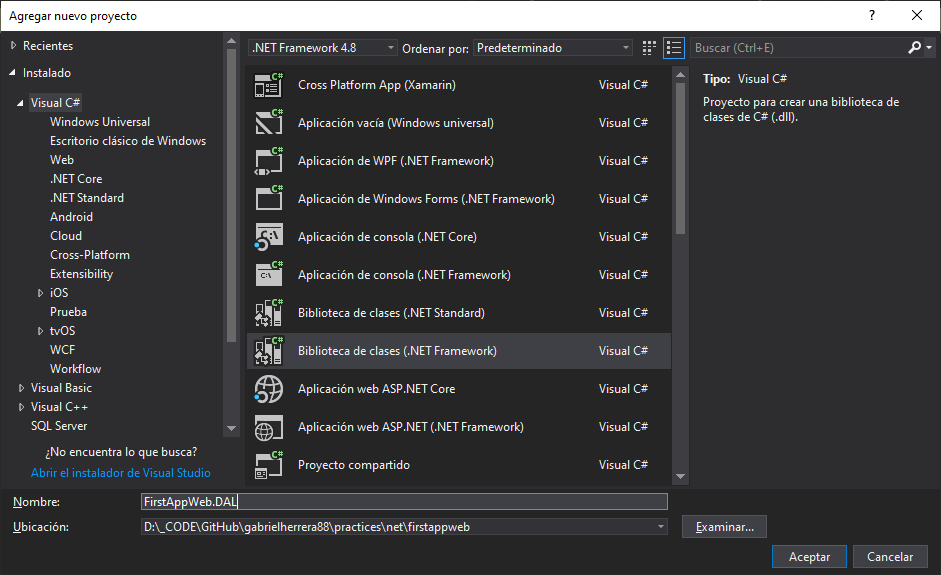


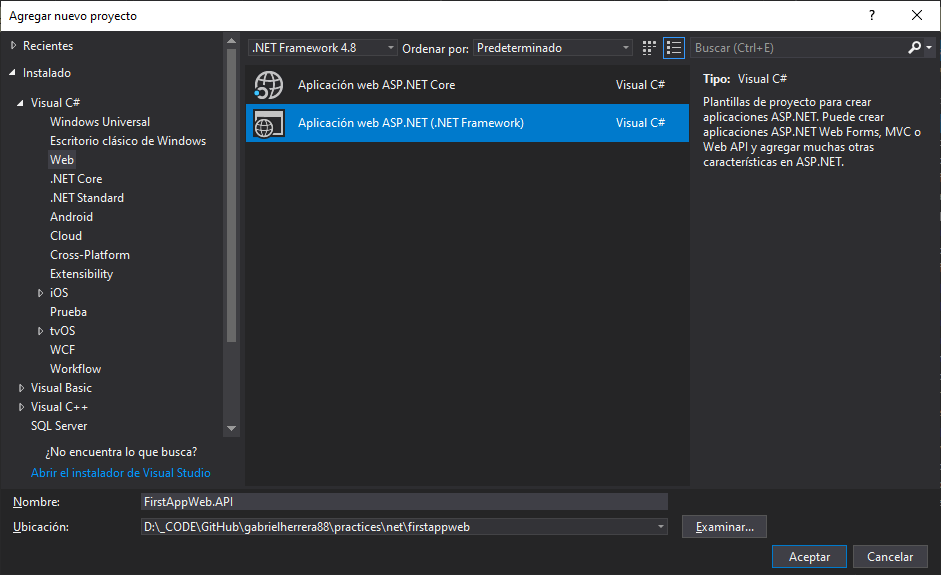
El primer proyecto que vamos a agregar va a ser de tipo C#, Web y Net Framework, va a ser la UI y de tipo MVC.

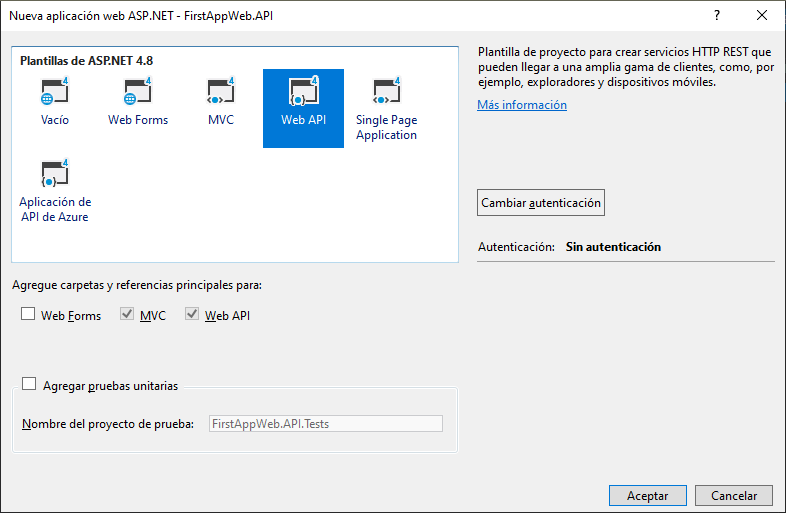


Luego desde el explorador de soluciones podremos ir agregando más proyectos.

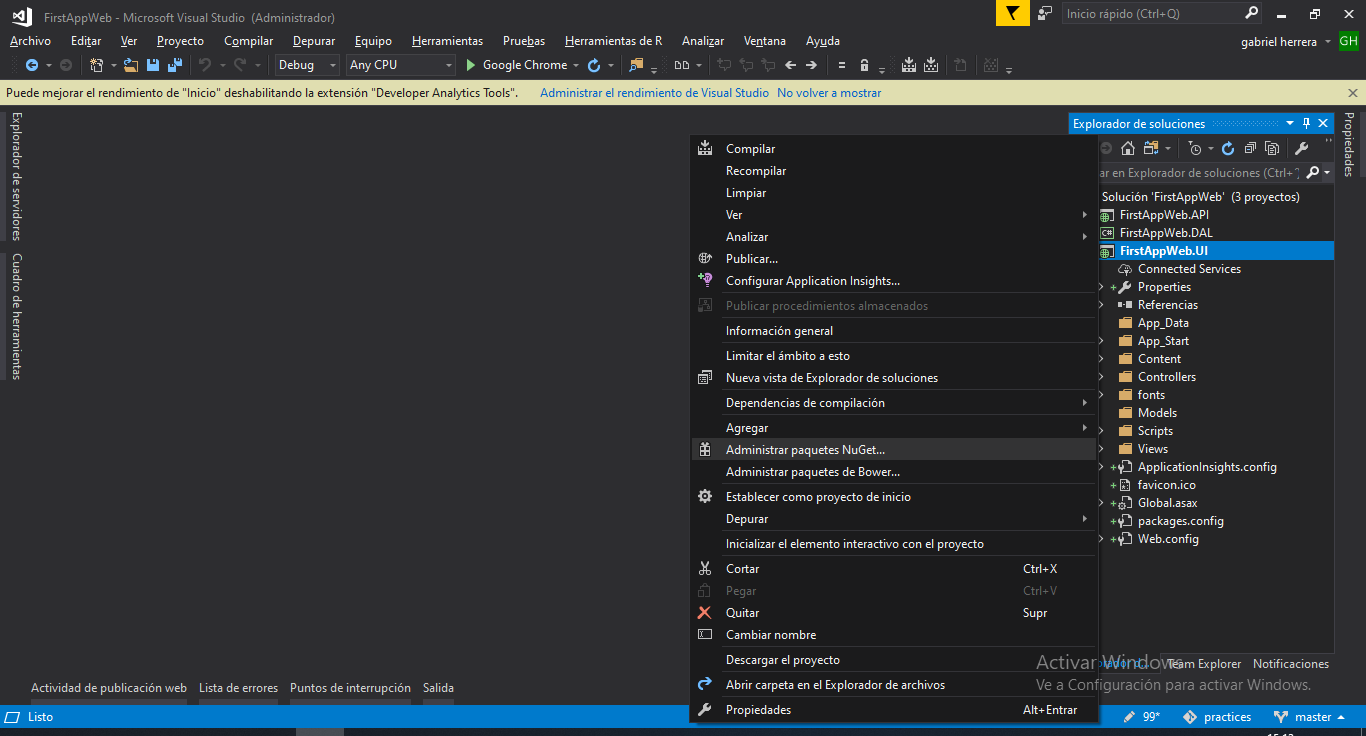
En este caso vamos a agregar un proyecto de tipo C#, Biblioteca de clases y Net Framework, esto quiere decir que al crearse el empaquetado de dicho proyecto vamos a tener como resultado un componente de tipo dll. Dicha dll va a ser consumida por otros proyectos, los componentes de tipo dll no hacen nada por sí solo, solo actúan cuando son instanciados.

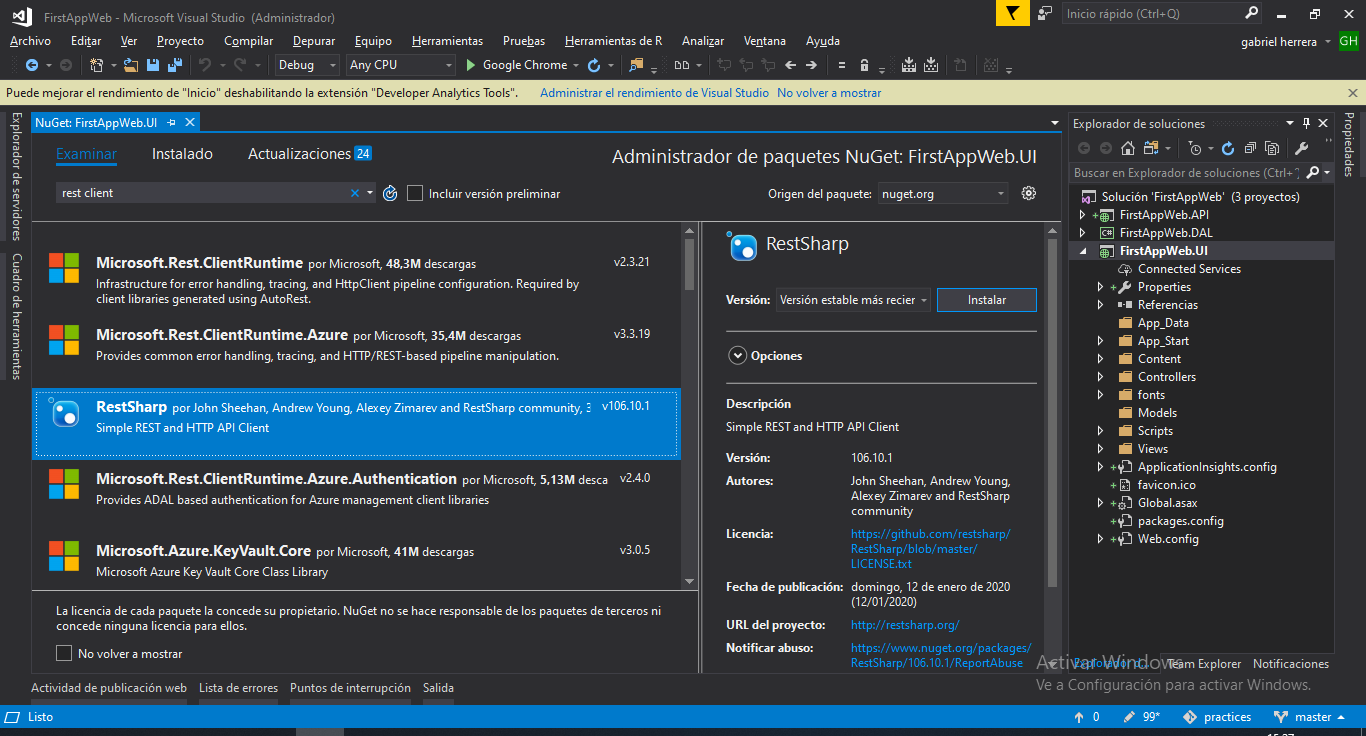


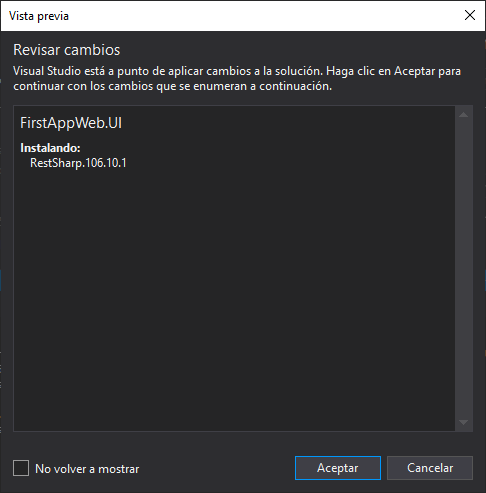
Agregamos un nuevo proyecto de tipo C#, Web y Net Framework va a ser la API que expondrá servicios RESTful mediante WebApi.



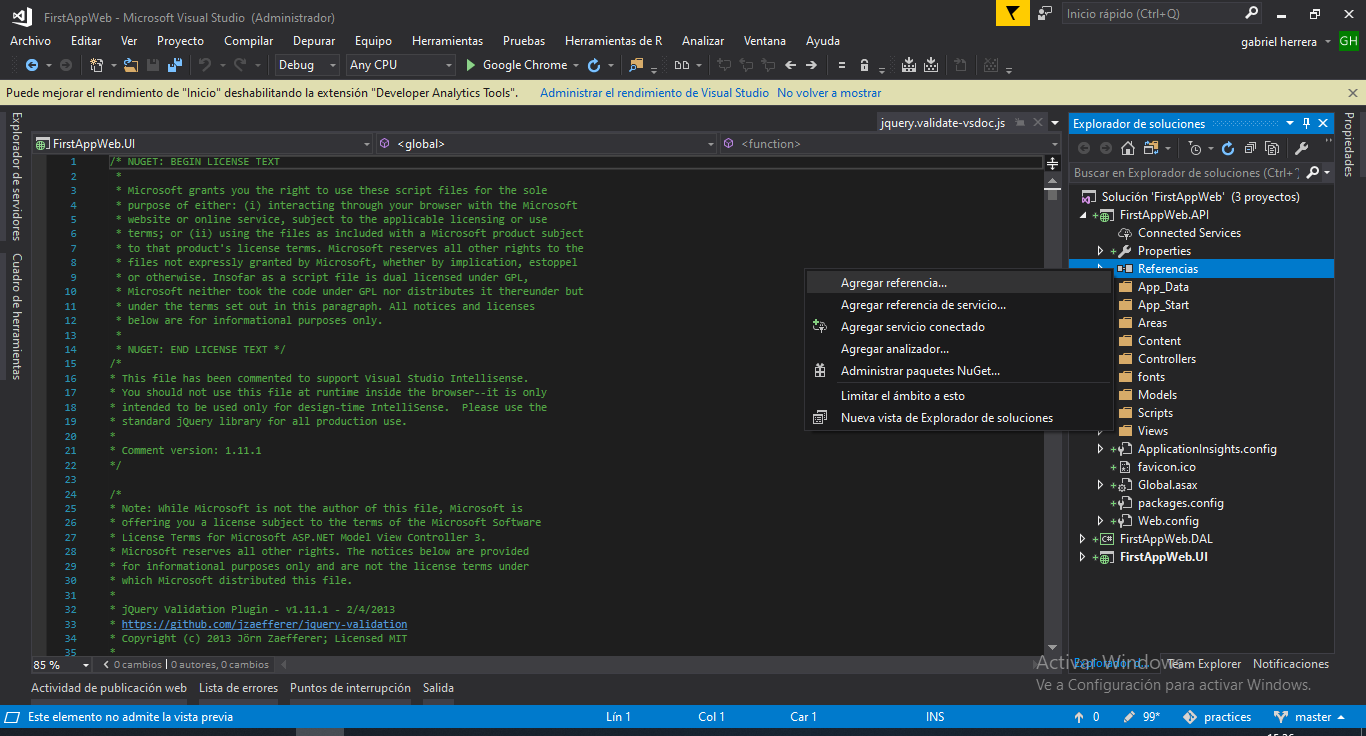
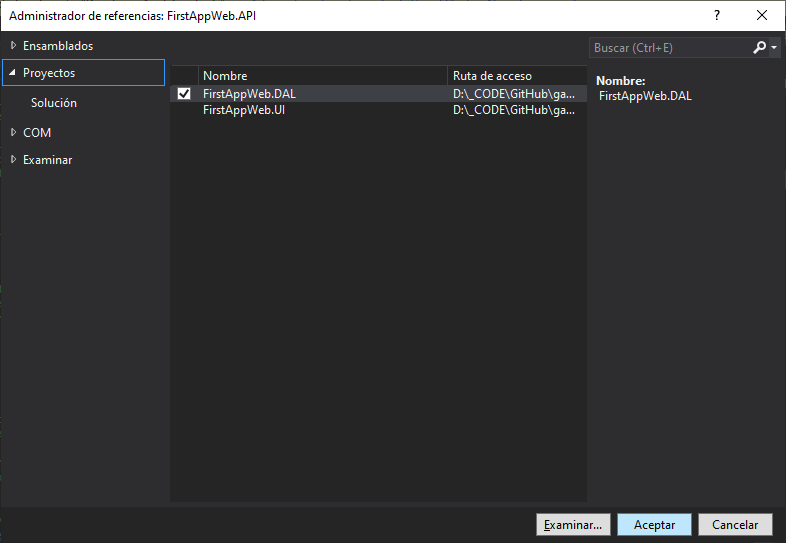
El administrador de paquetes NuGet permite agregar/actualizar/eliminar componentes externos a nuestra aplicación.

Agregaremos Bootstrap (para hacer nuestra app adaptativa, toolkit que provee clases CSS), jQuery (librería javascript que la usaremos para el control de nuestras vistas) y RestSharp (para poder consumir los servicios expuestos por la capa API).



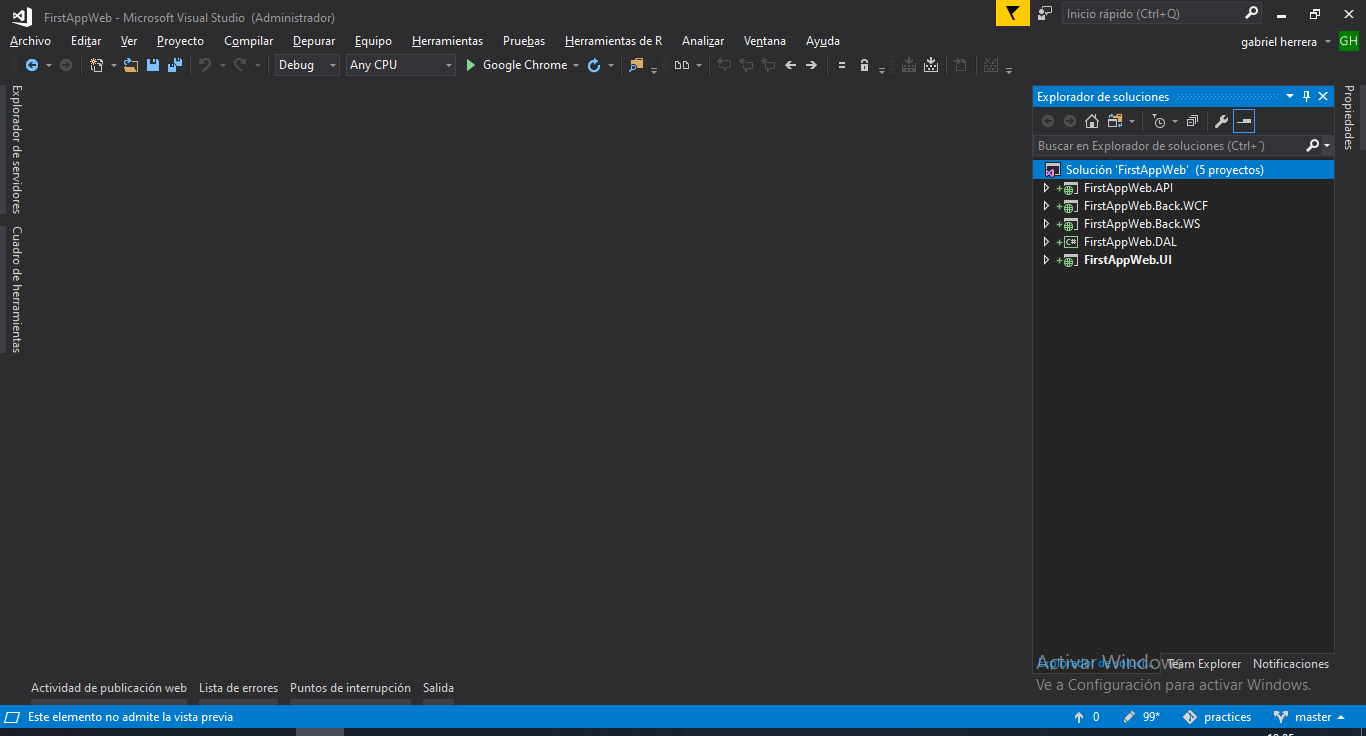


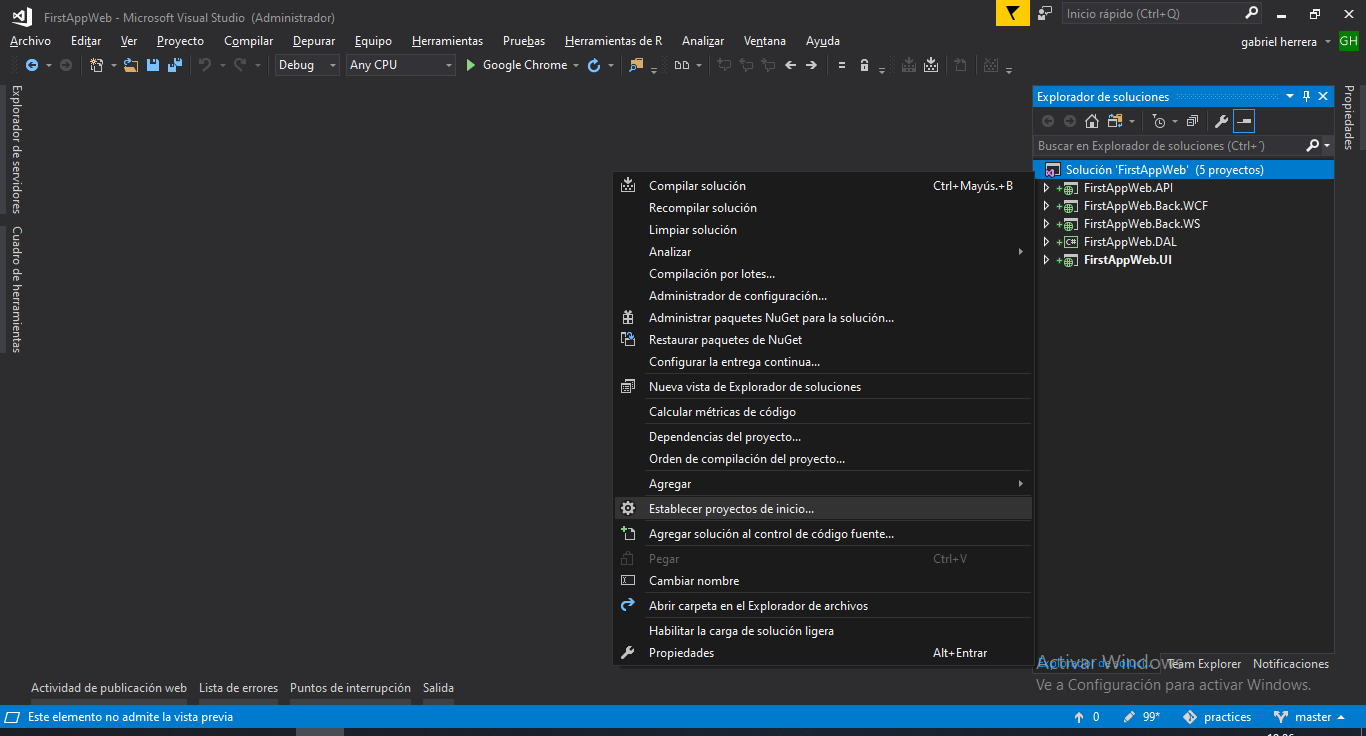
Vamos referenciar a nuestro componente DAL desde la API ya que este último se apoyara en DAL para el acceso a datos.



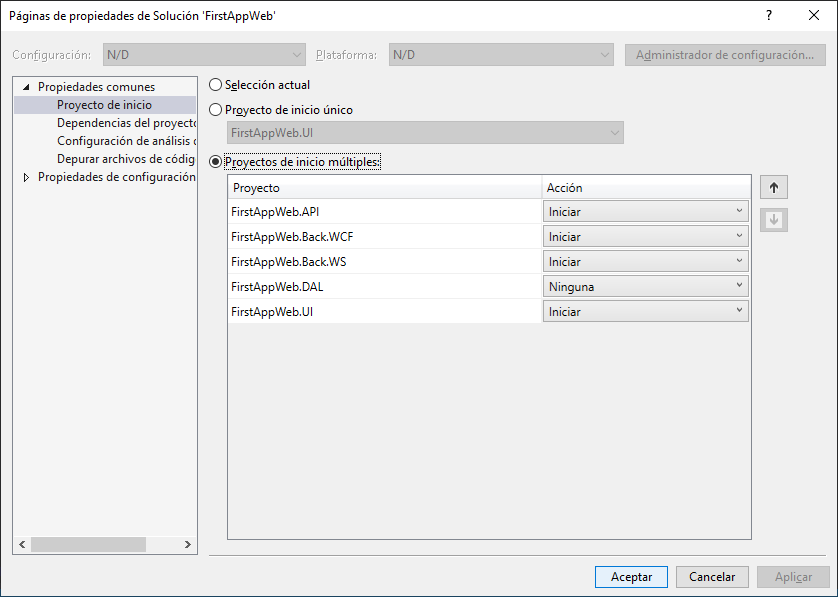
Podemos tomar componentes desde Ensamblados (Componentes del Framework), COM (Componentes que exponen SDKs o aplicaciones que tenemos instaladas) o Examinar (básicamente buscar dlls que estén en nuestra pc).

**La solución en un vistazo:**

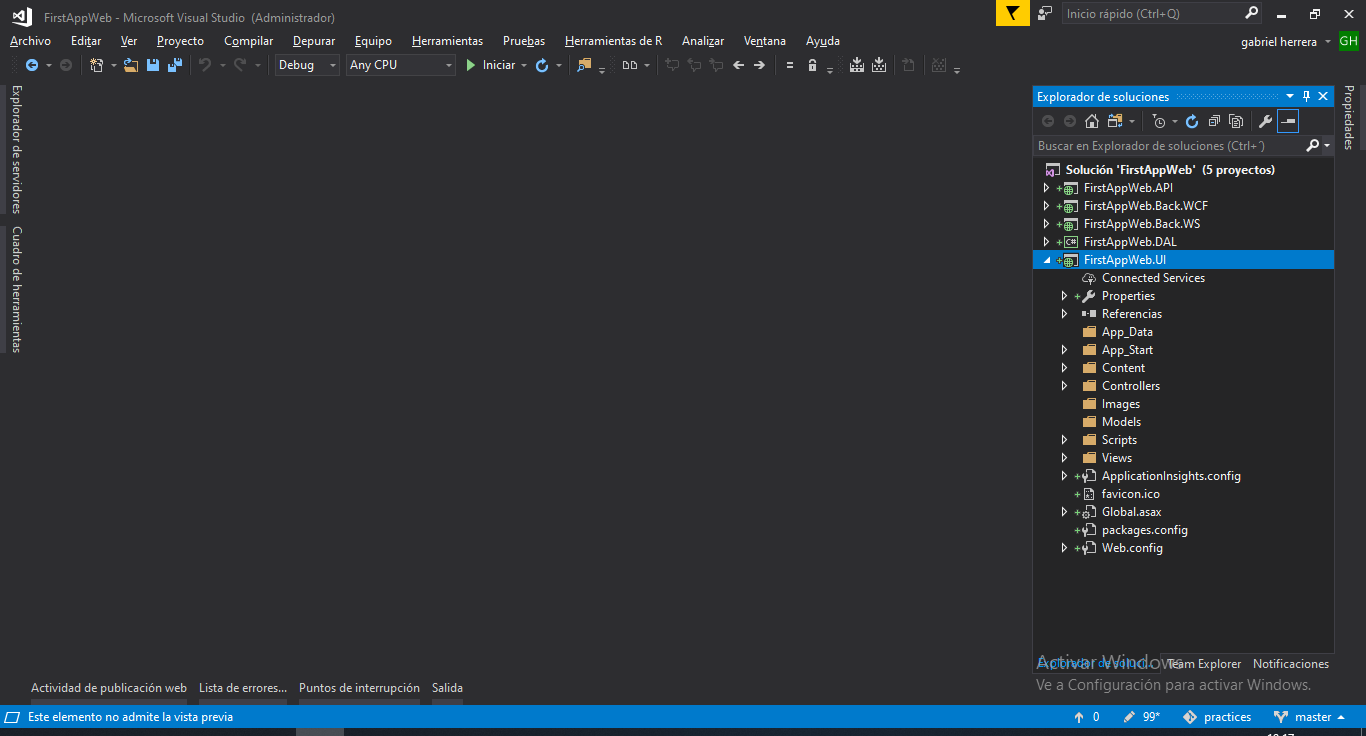


Al ser una aplicación compuesta de varios proyectos y algunos de ellos requieren que estén activos y ejecutándose para poder intercomunicarse, debemos indicarle al IDE que al momento de la ejecución levante los artefactos que le indiquemos.

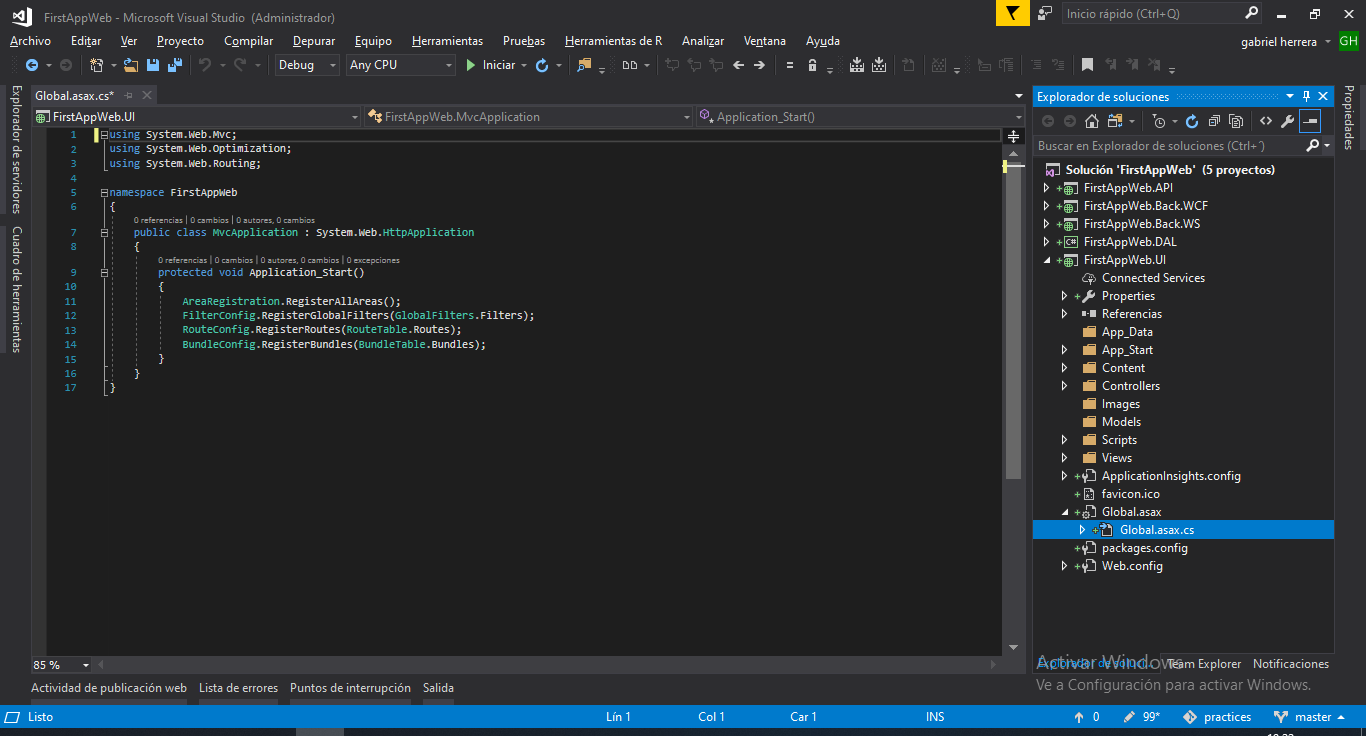
Esto hará que los proyectos indicados se inicien y mediante IIS Express estén aptos para ser consumidos mediante un puerto que el mismo IIS Express le suministra a cada uno.



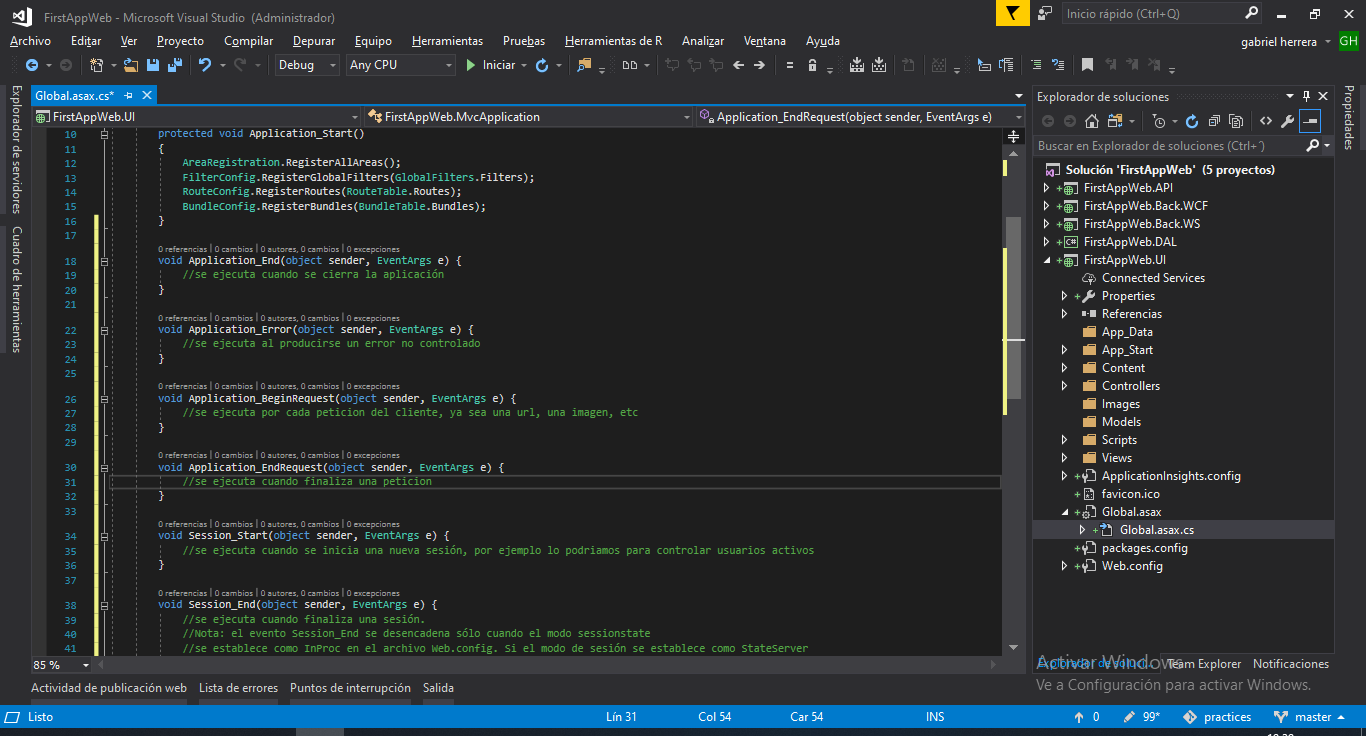
La UI por dentro, Content es el directorio que tendrá los estilos (CSS), Controllers es el directorio que tendrá los controladores de MVC, Images es el directorio que tendrá todas las imágenes de nuestra aplicación, Models es el directorio que tendrá los diferentes modelos que servirán datos a nuestras vistas, Scripts es el directorio que tendrá los javascript de nuestra aplicación, Views es el directorio que tendrá todas nuestras vistas.

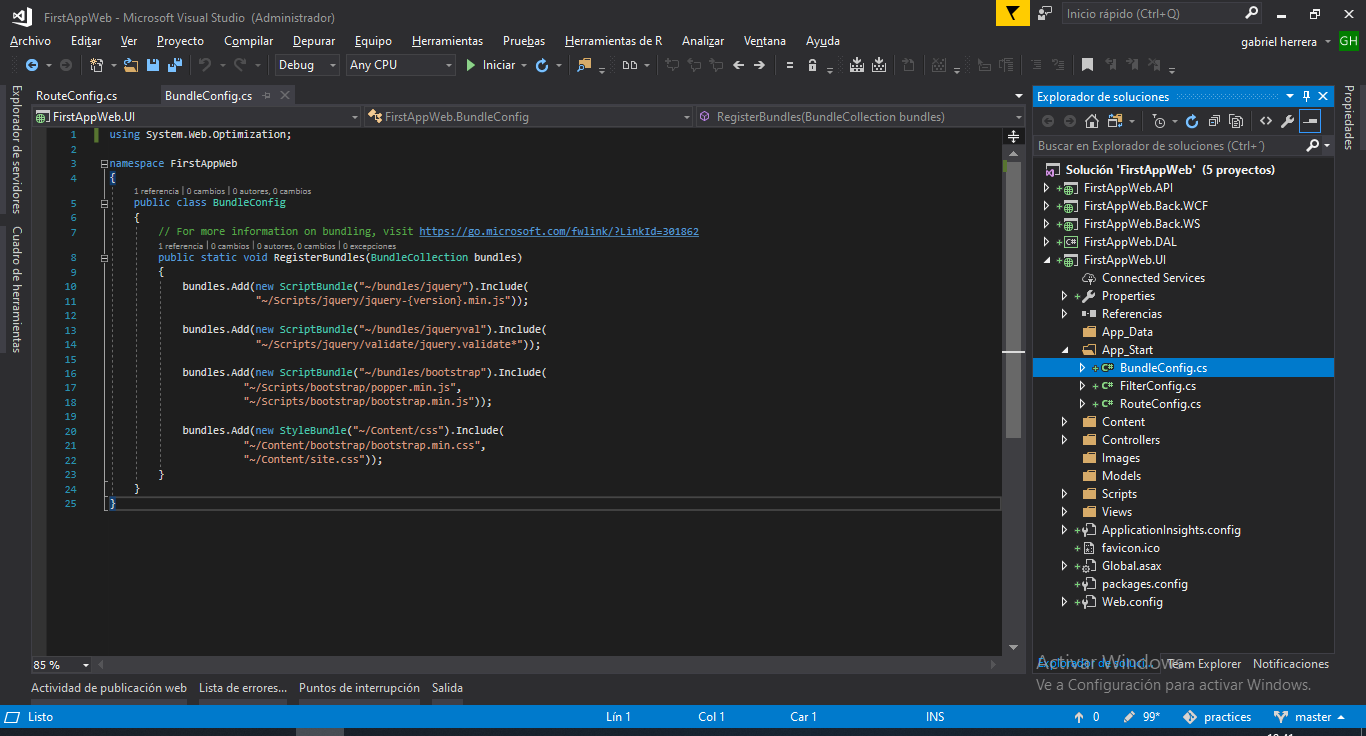


El Global.asax es una clase especial en el cual se declaran y manejan eventos y objetos a nivel de aplicación. En tiempo de ejecución antes de la llegada de la primera ejecución (y cada vez que se reinicia el sitio mediante el reinicio de IIS o del Application Pools) este archivo es analizado y compilado. No se puede realizar peticiones directamente al Global.asax.

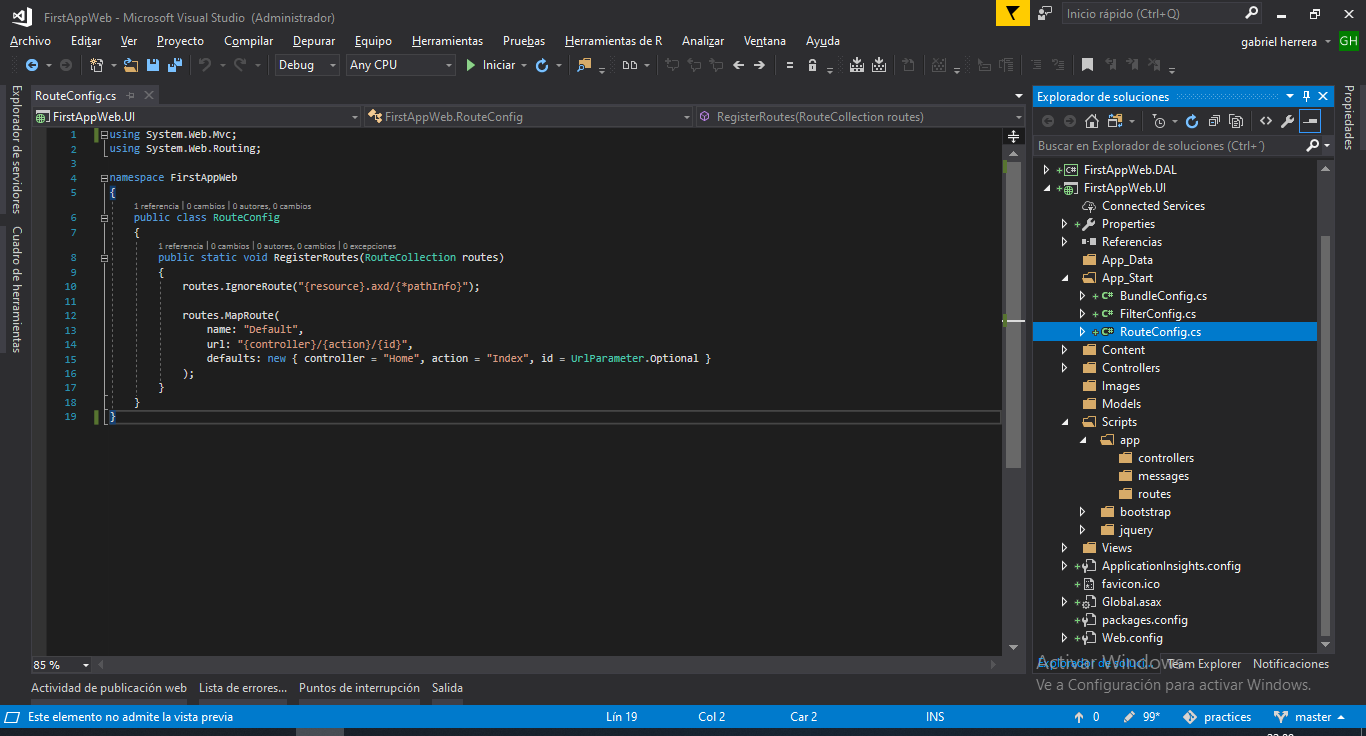


Los eventos más importantes y su uso.



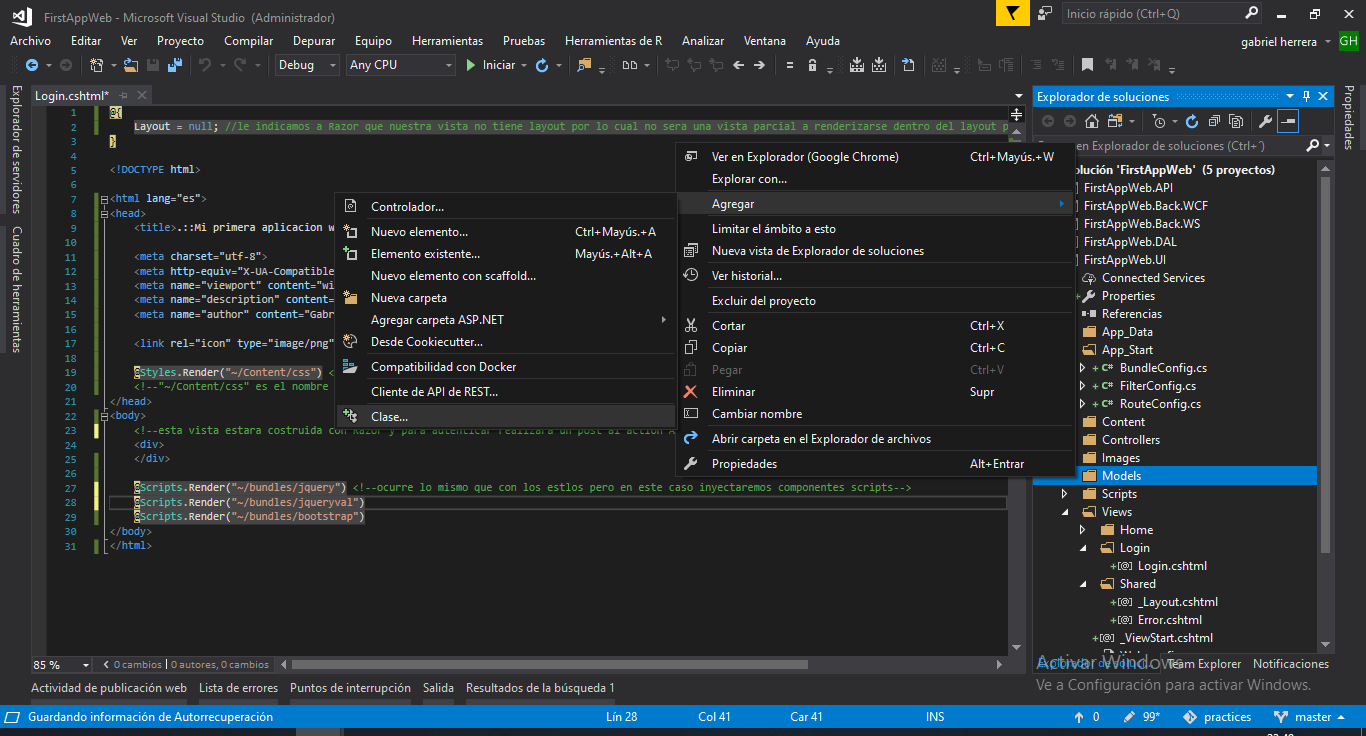
La clase BundleConfig nos permite agrupar css y js bajo un alias, básicamente sirve para agrupar una cantidad dada de componentes bajo un alias y al invocar dicho alias en nuestras vistas se inyectaran automáticamente todos los componentes que estén declarados dentro de dicho alias tanto para css como para js.

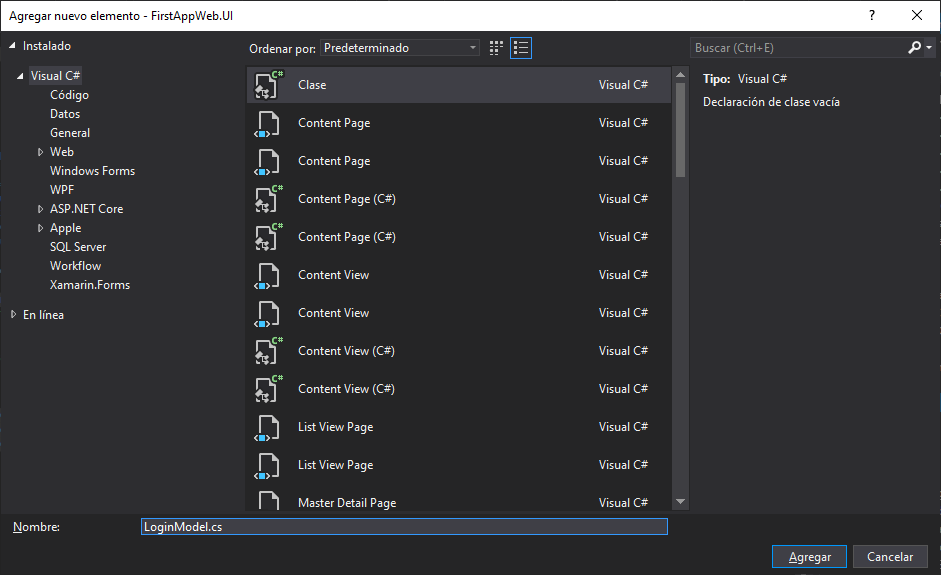
La clase RouteConfig nos permite definir la forma en la que se van a invocar a nuestros recursos mediante las URLs, llámese recursos a una vista, una imagen, un método que recibe parámetros e inserta datos en un catálogo, etc. Siguiendo la ruta por defecto vemos que está conformada de la siguiente manera: controller/action/id, cuando detallemos la construcción de un controller volveremos a hacer referencia a las rutas.



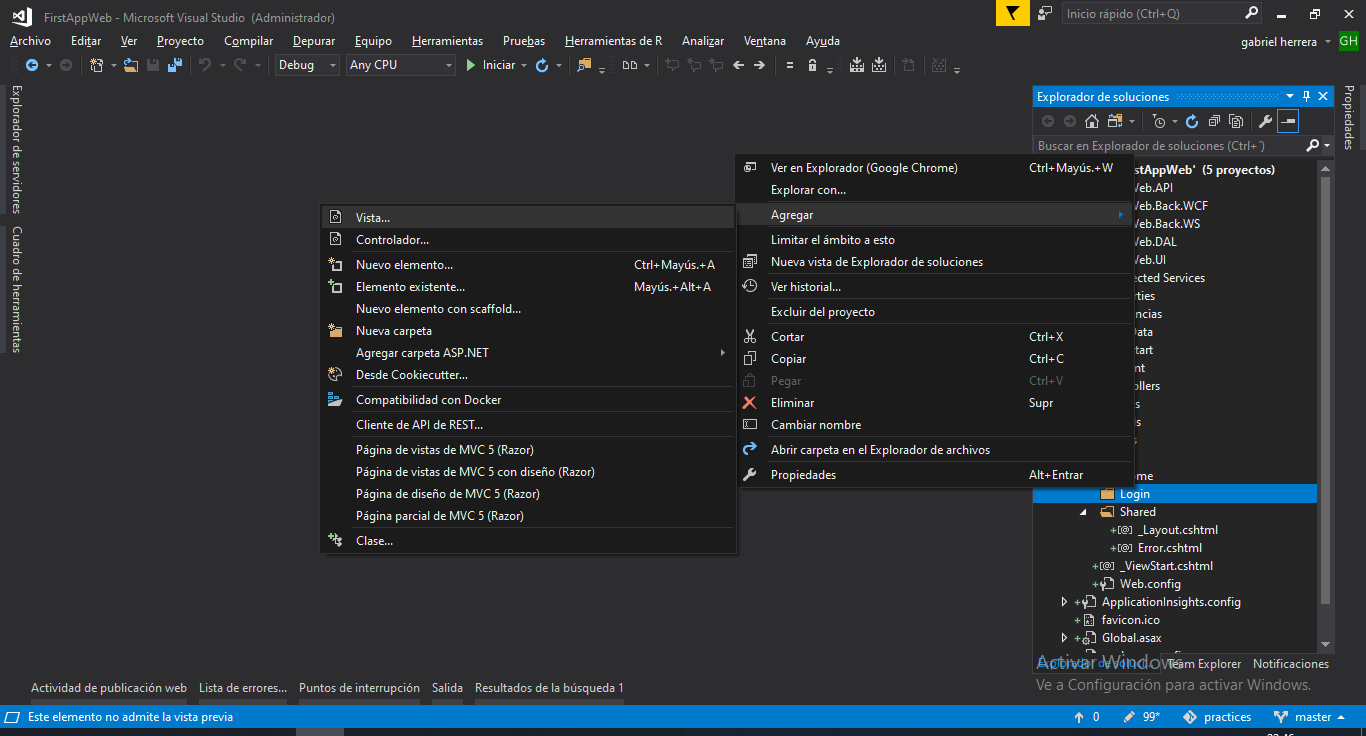
**Manos a la obra**:

Todas las vistas, clases, modelos y demás componentes están comentados por lo cual para mayor comprensión lo mejor es ir siguiendo este manual junto con el código.

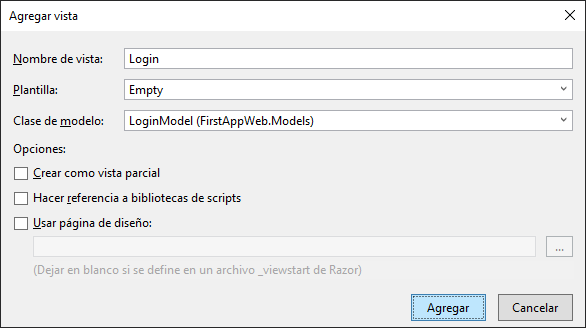
Agregaremos un modelo dentro del directorio Models, llamado LoginModel que servirá para cargar los datos de la vista y transportar dichos datos hacia el controlador.



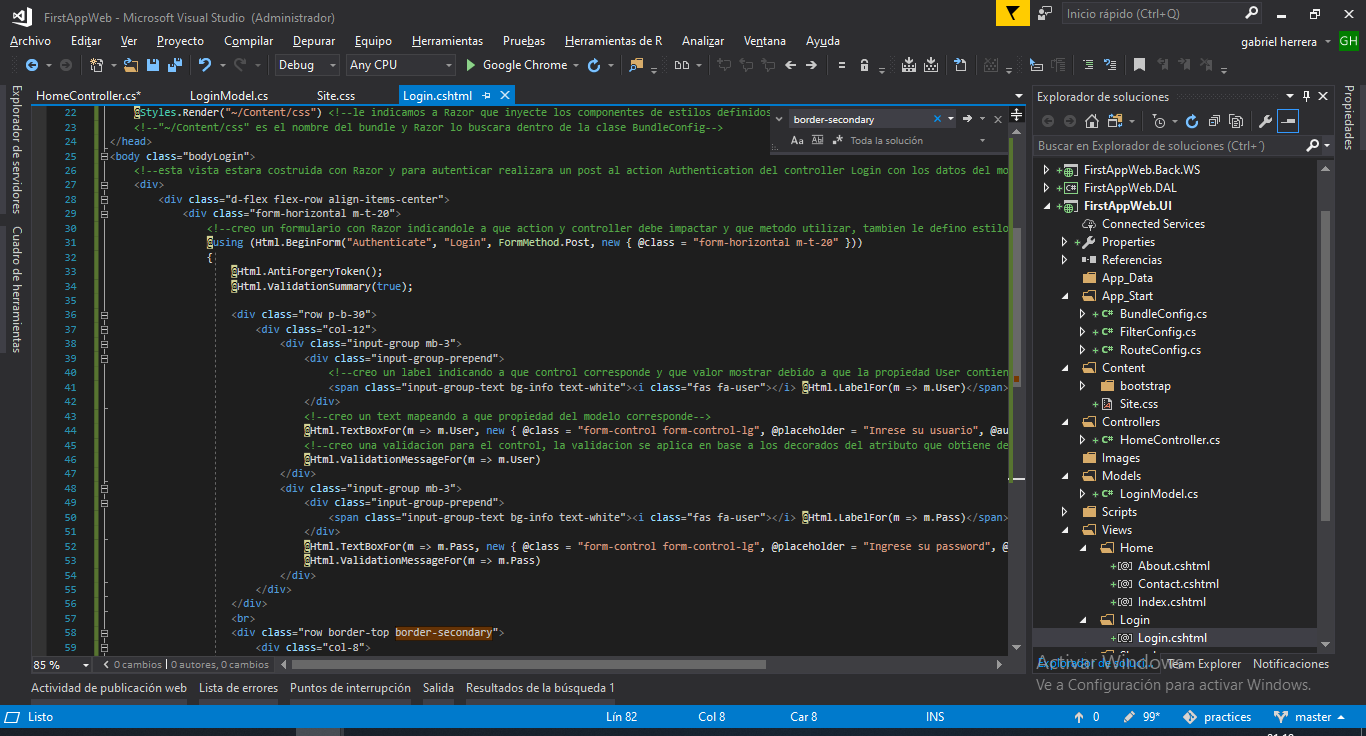
Agregaremos un directorio dentro de Views que se llamara Login y dentro una vista que se llamara de igual nombre.



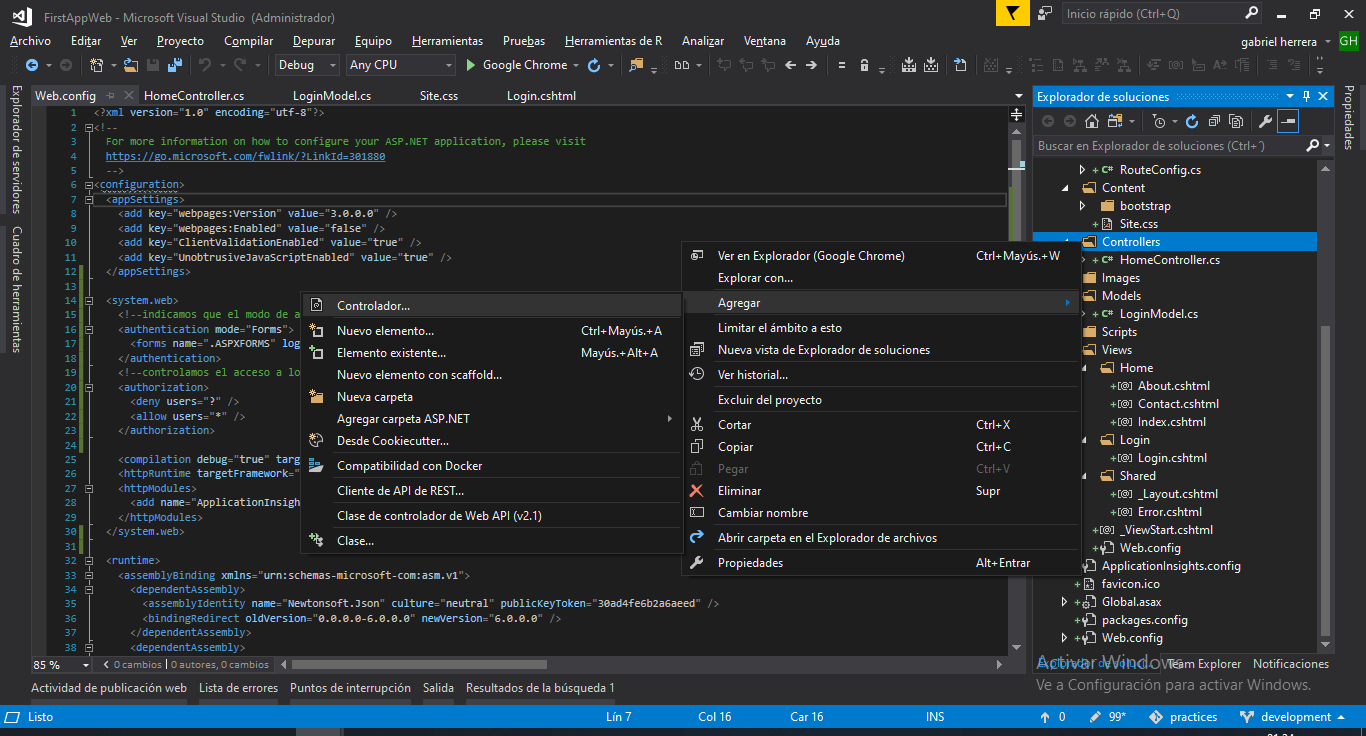
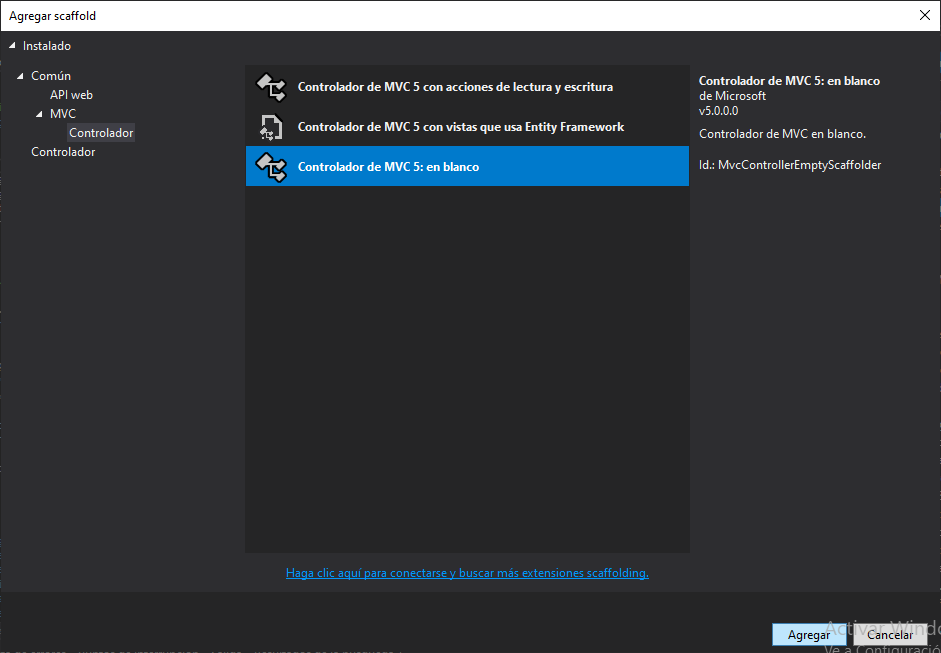
De esta manera agregaremos una vista con un HTML completo y que no depende de un Layout, ya veremos qué es esto. También le indicaremos con qué modelo trabajar.



Mediante el uso de Razor armaremos nuestra vista, agregaremos estilos y scripts con los Renders, armaremos un formulario con dos text y un botón para generar el submit al recurso indicado por el formulario.

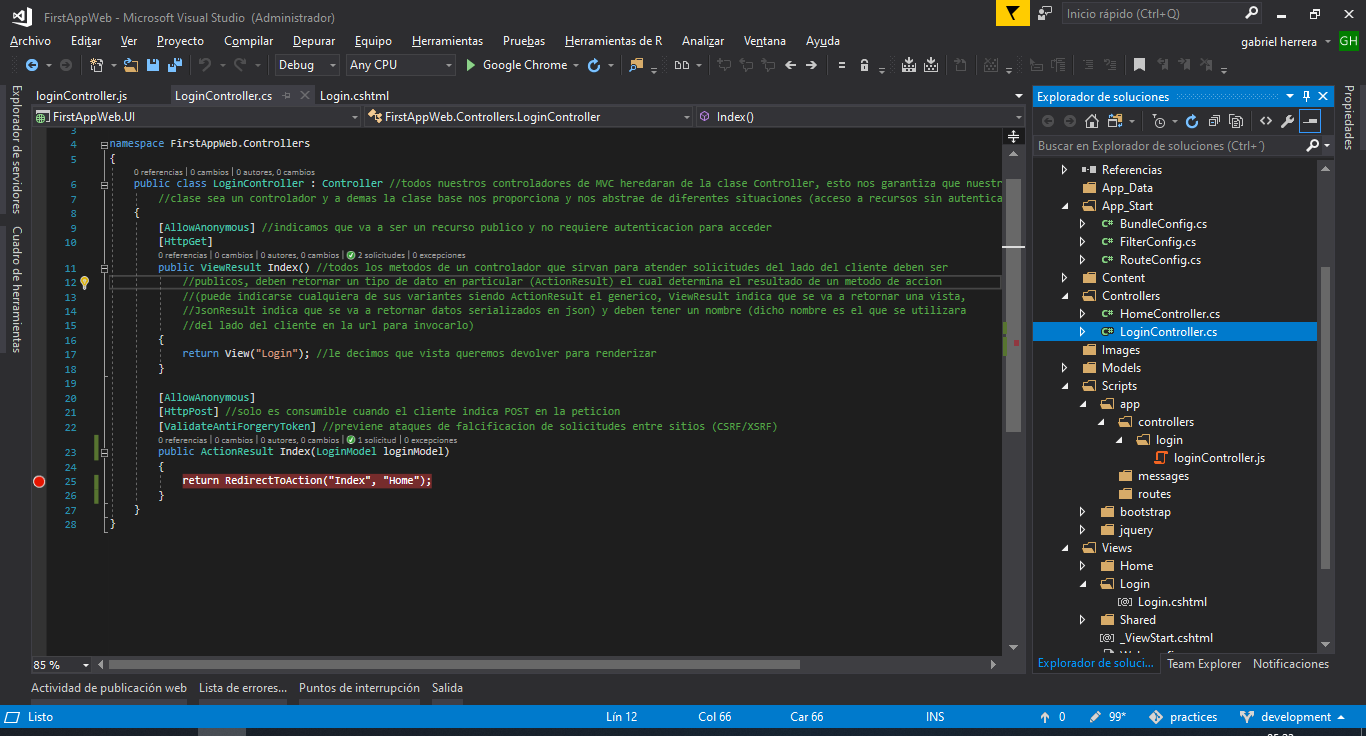


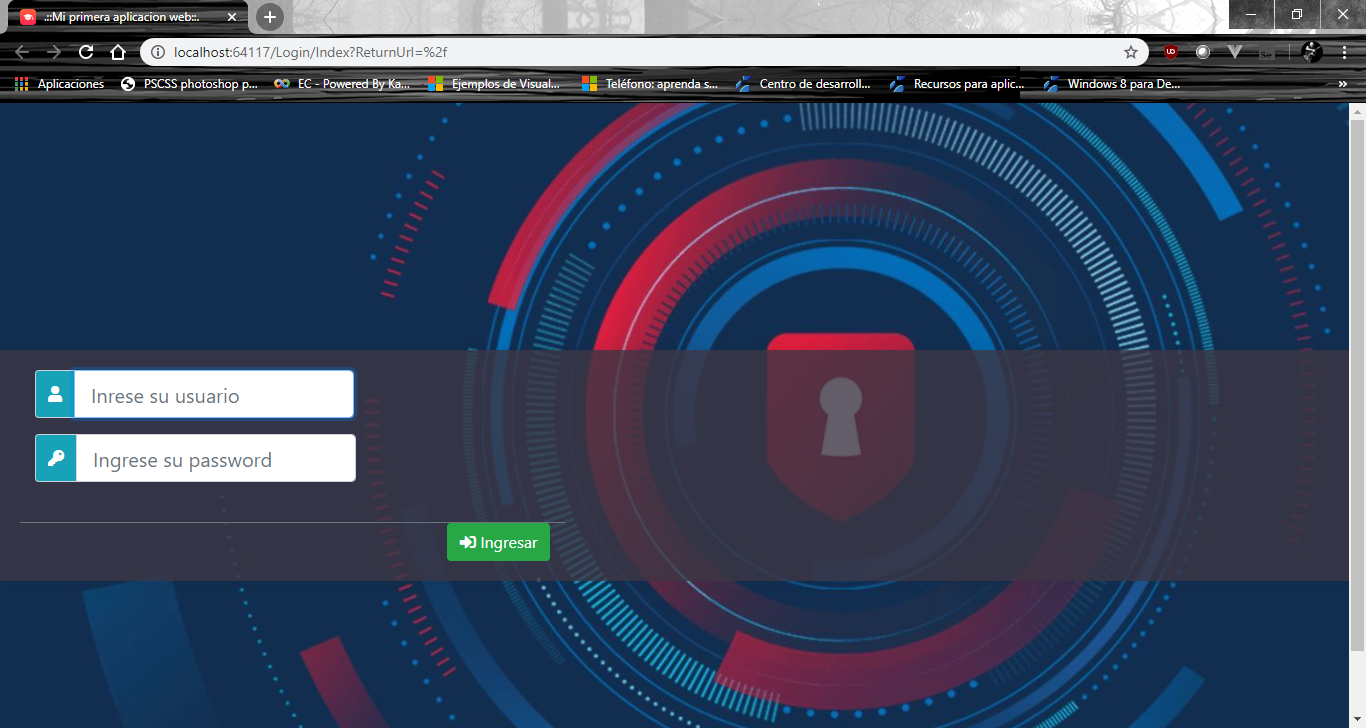
Antes de poder ver nuestra vista en acción deberemos agregar un controlador, será el encargado de devolver al cliente la vista a renderizar cuando el cliente solicite el recurso y también atenderá la solicitud de autenticación.



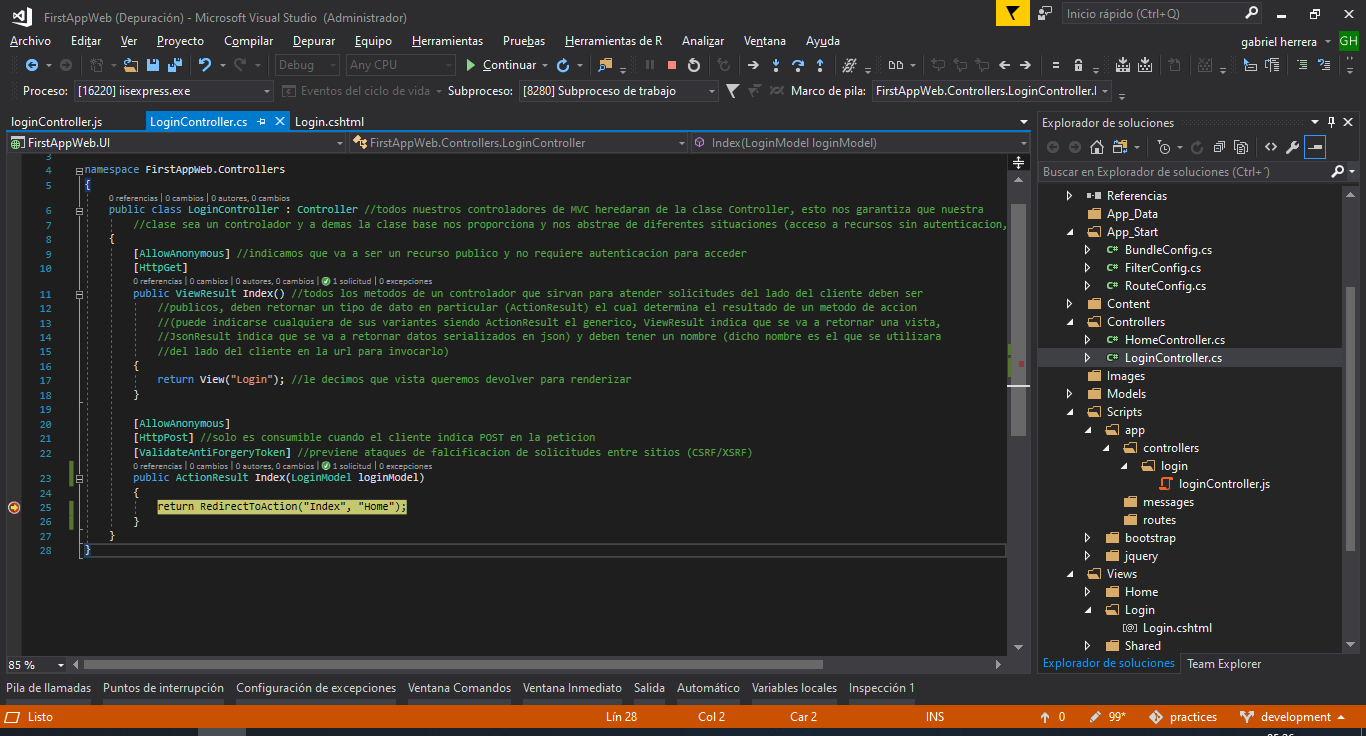
De manera automática vamos a encontrarnos con que el IDE nos recomienda que nuestros controladores terminen con el sufijo Controller, esto se debe a responder a las nomenclaturas propuestas por Microsoft para este tipo de desarrollos.

En principio nuestro controlador tendrá dos acciones una para retornar la vista del Login y otra para la autenticación.



**Nuestra primera ejecución:**

Por medio del uso del formulario en el Login, al completar los datos y hacer click en el botón Ingresar podremos enviar los datos al servidor para su procesamiento.



**URLs**:

<https://dotnet.microsoft.com/download/visual-studio-sdks?utm_source=getdotnetsdk&utm_medium=referral>

[https://jquery.com](https://jquery.com/)

[https://getbootstrap.com](https://getbootstrap.com/)

[https://www.w3schools.com](https://www.w3schools.com/)