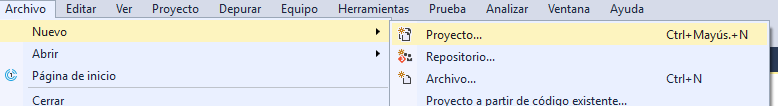
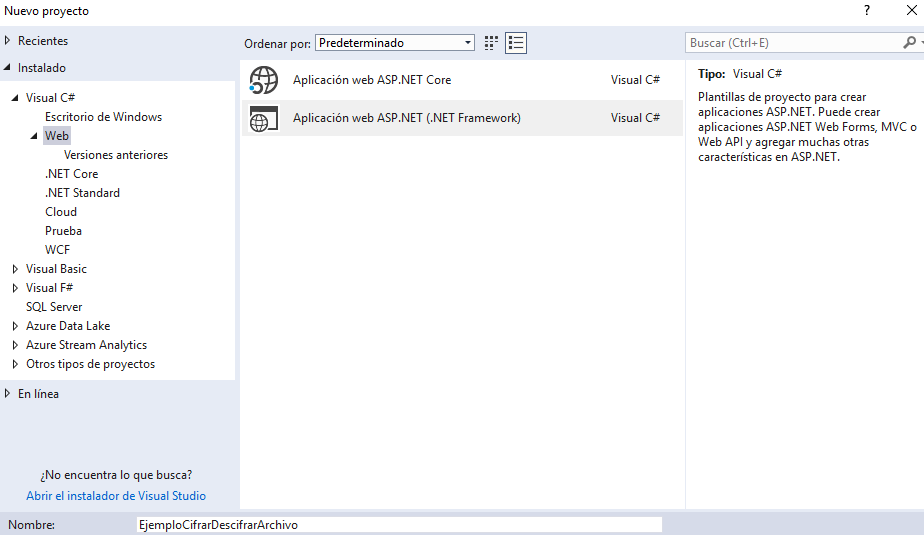
Este post es una guía paso a paso, de cómo crear un proyecto *ASP.NET MVC* funcional, que permite la encriptación y la desencriptación de un fichero.

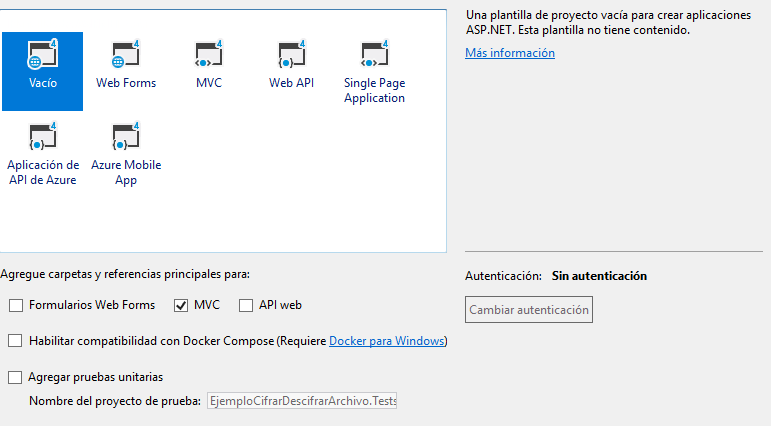
Lo primero de todo es abrir *Visual Studios*, crear un nuevo proyecto.



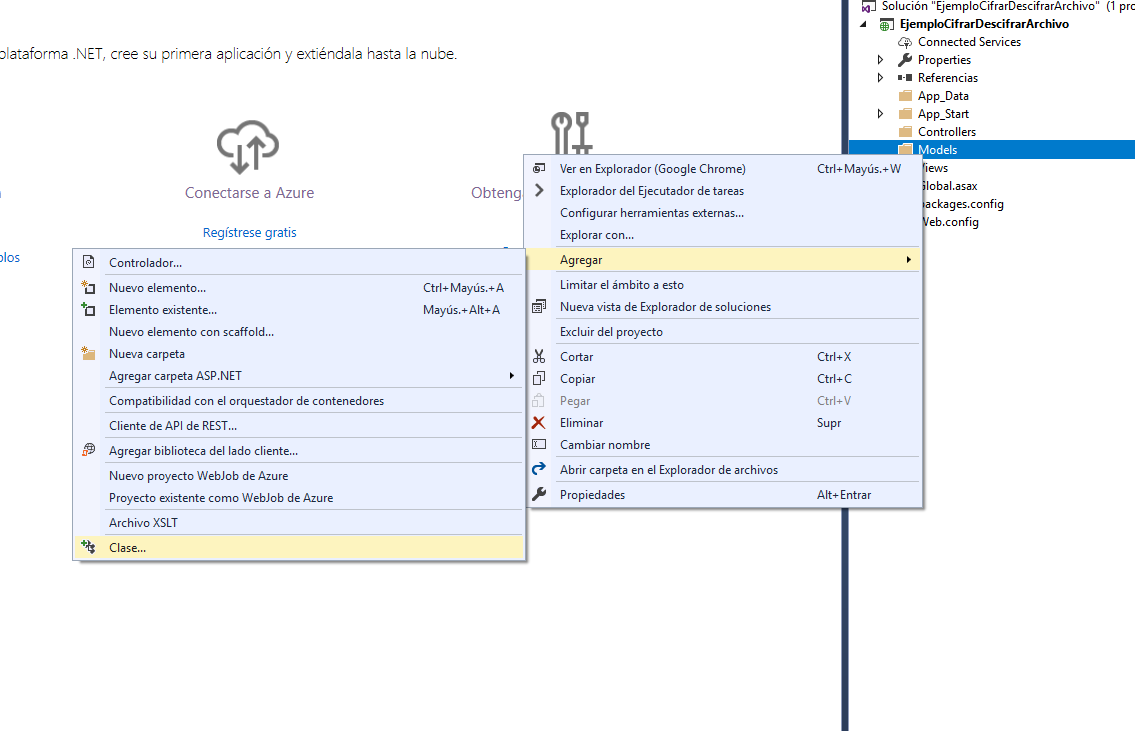
Seleccionamos *ASP.NET* y en mi caso lo he llamado **“EjemploCifrarDescifrarArchivo”**.

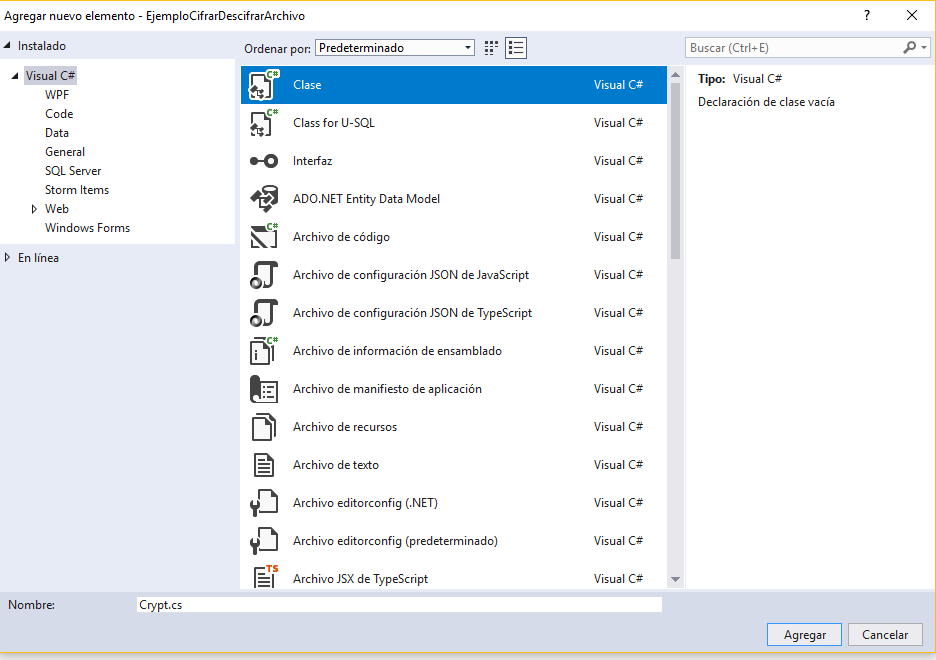


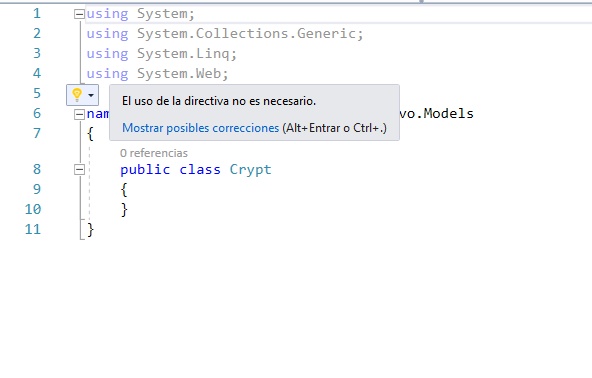
Seleccionamos *Vacío (Empty) y MVC.*



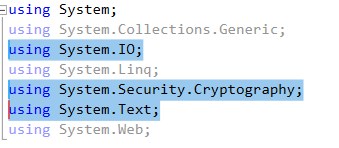
Una vez creado nuestro proyecto, sobre la carpeta **Models,** añadiremos una clase, la cual va ha ser la encargada de cifrar y descifrar los archivos. En mi caso he llamado a esta clase **“Crypt”**.



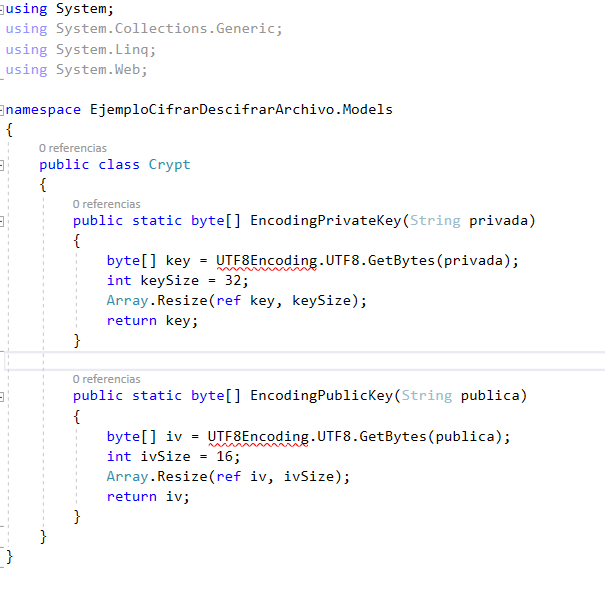




Una vez tenemos la clase creada, deberemos añadir los recursos que vamos a utilizar.



Añadiremos los métodos que nos permiten codificar las claves (convertir a byte[]) para que tengan el formato establecido por la codificación *Rijndael*.



Ahora crearemos los métodos que se encargarán de encriptar y desencriptar los ficheros que se crearán.

El método **“EncryptToFile”** contendrá cuatro parámetros:

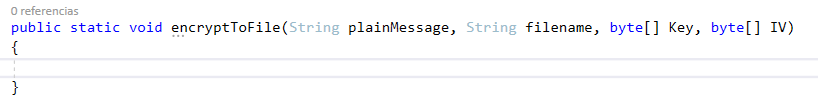
-El texto que guardaremos en el fichero(**“plainMessage”**).

-La ruta mapeada más el nombre del archivo a generar(**“filename”**).

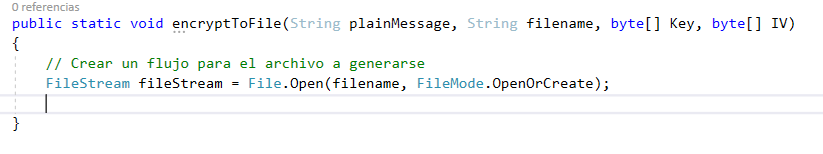
-La clave privada convertida en byte[](**“key”**).

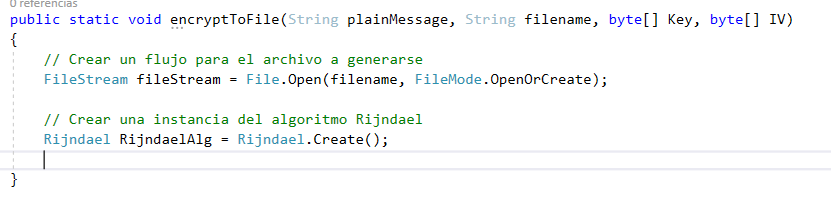
-La clave pública convertida en byte[](**“IV”**).

Será un método *void* (No devuelve nada).



A continuación, crearemos el flujo del archivo a generar.

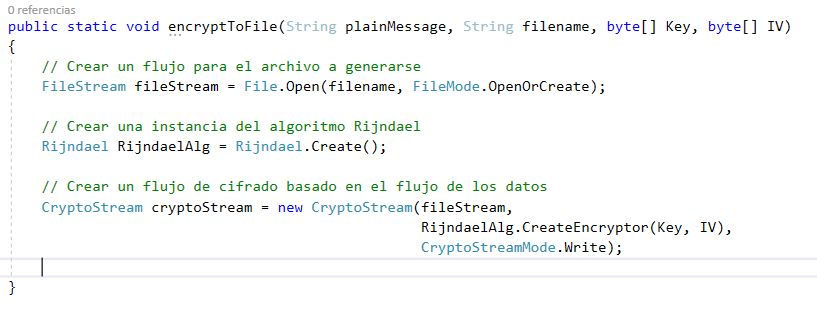


También añadiremos el flujo de la clase *Rijndael*.

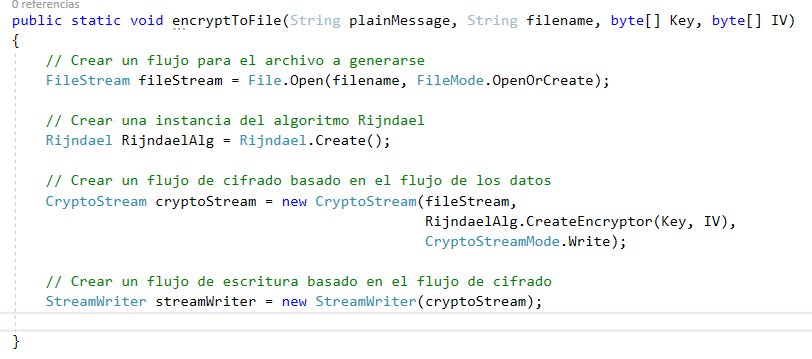
Ahora el *CryptoStream*, que es el flujo de datos cifrados.

**-Nota Importante-**

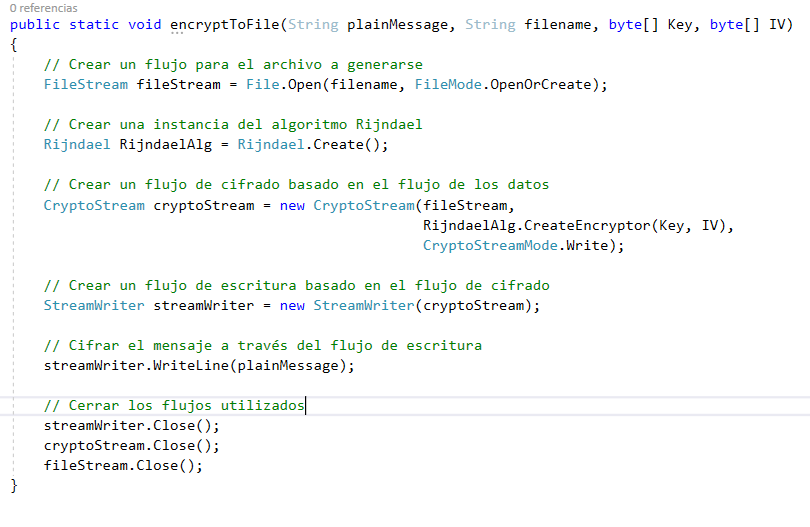
*CryptoStreamMode.****Write*** ya que estamos escribiendo en un archivo, mas adelante veremos que con ***Read***, podremos desencriptar dicho archivo.



También añadiremos el flujo de escritura *StreamWriter*.



Por último, ciframos el contenido del archivo y cerramos todos los flujos creados.



Este método está listo para su uso, pero ahora vamos a crear el método de descifrado(**“DecryptFromFile”**), que devolverá un *String*.

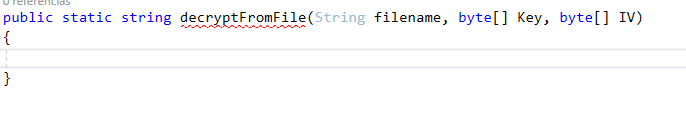
Es muy parecido, salvo qué, en vez de escribir, vamos a tener que leer.

Esta vez solo recibirá los parámedros de:

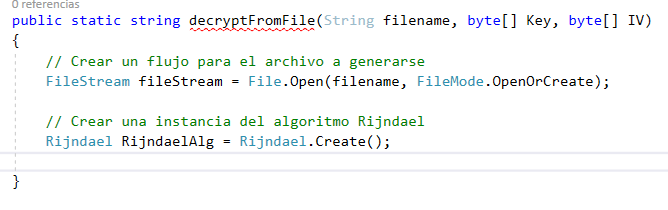
-Nombre del archivo (**“filename”**).

-La clave privada convertida en byte[](**“key”**).

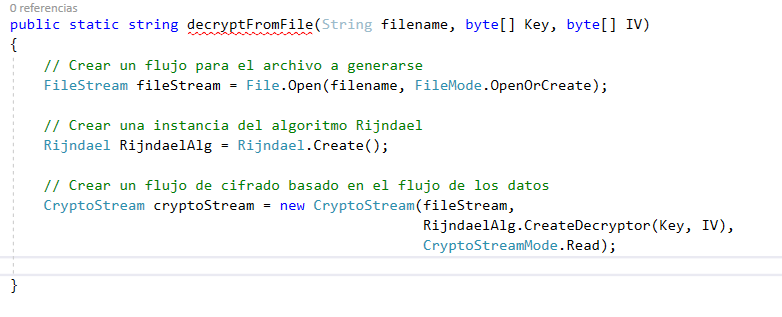
-La clave pública convertida en byte[](**“IV”**).



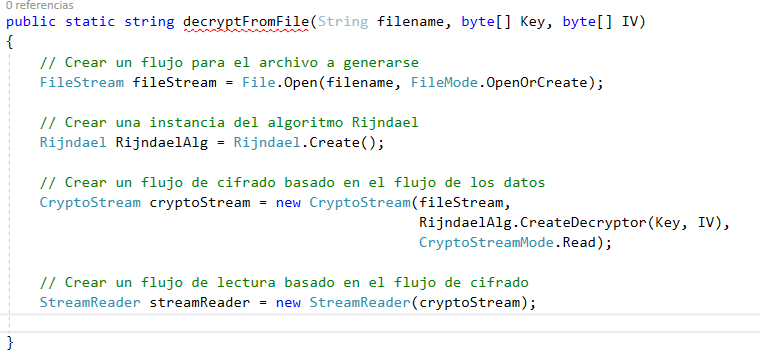
Abrimos los flujos *FileStream* y *Rijndael*.



Añadimos el flujo de *CryptoStream*, salvo que esta vez con el *CyptoStreamMode***.*Read.***

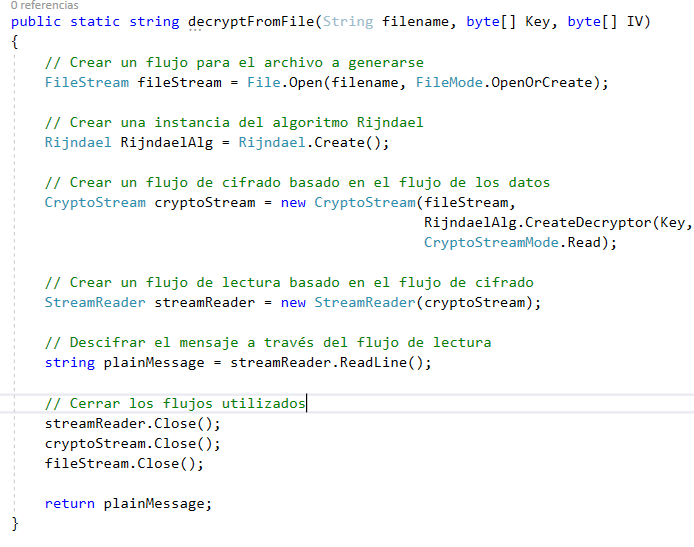


Ahora deberemos abrir el flujo *StreamReader*.



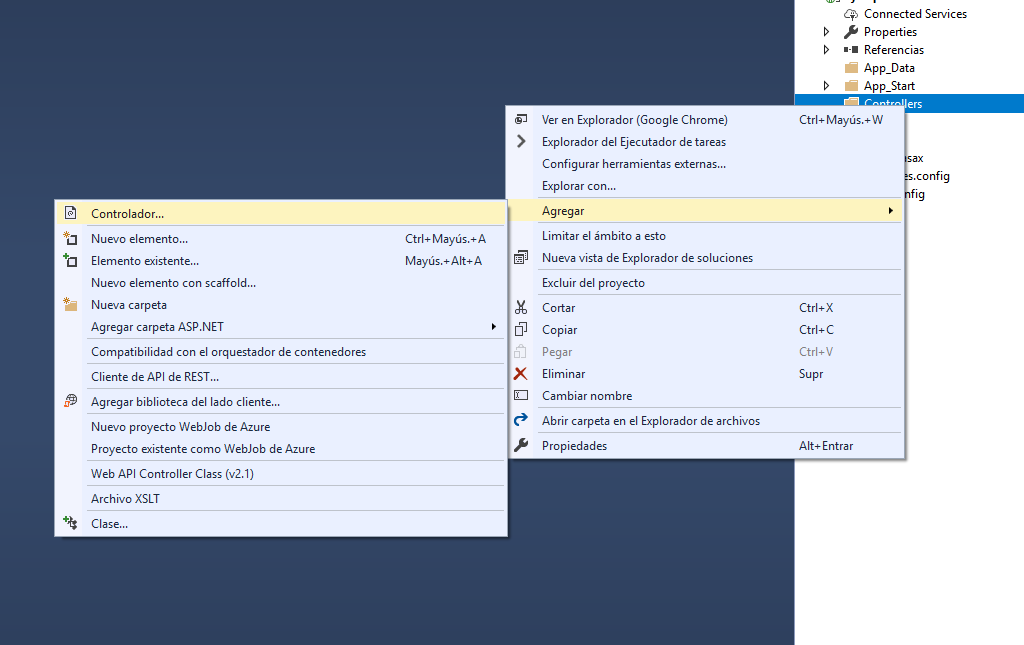
Por último, al igual que antes; Deberemos desencriptar el archivo y cerrar todos los flujos usados.

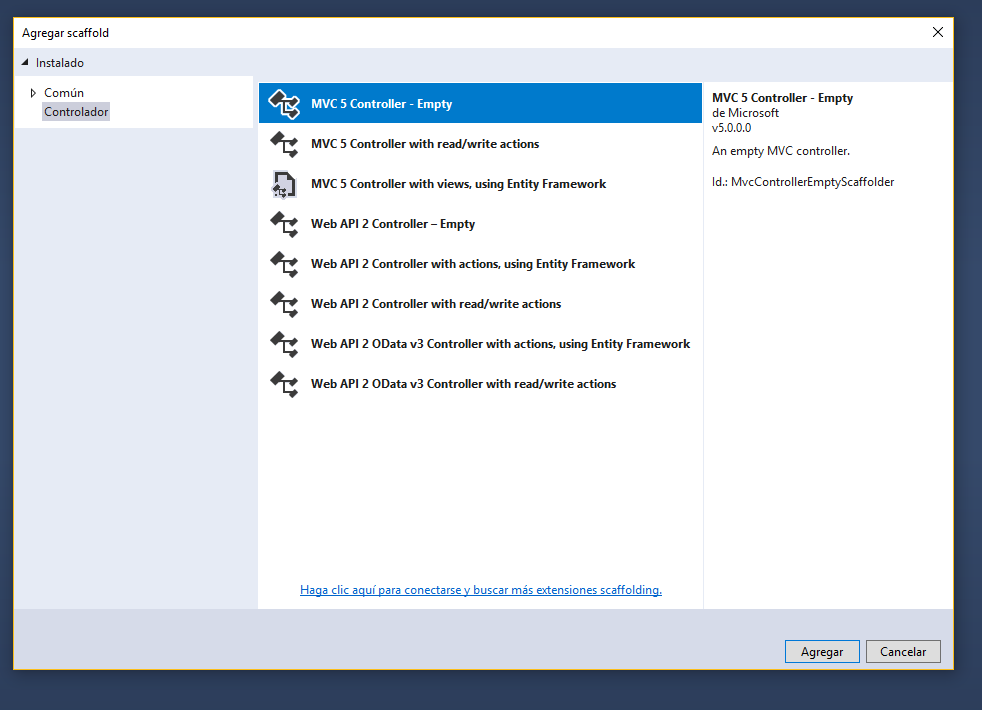
Este método tiene que devolver un *String,* muy importante no olividar el *return*.



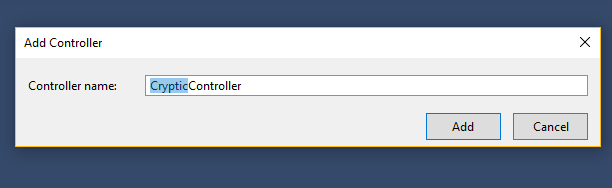
Con esto ya hemos terminado con el archivo **“Crypt”.**

Es hora de añadir un controlador en la carpeta **Controllers,** el cual será el encargado de devolver las vistas y el que recibirá la información necesaria mediante el método *Post.*

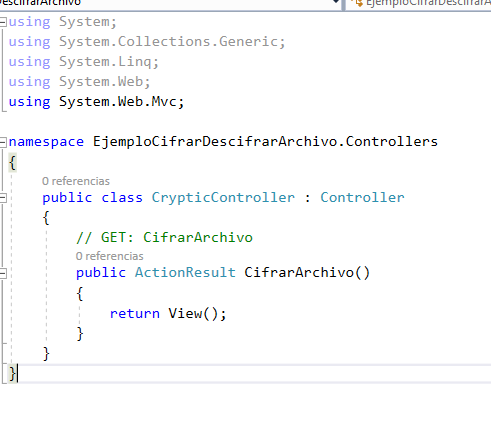




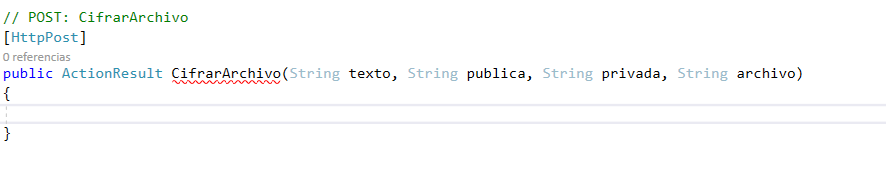
En mi caso he llamado al controlador **“CrypticController”.**

****

Una vez añadido, deberemos cambiar el nombre al método *Index* (no es necesario) por **“CifrarArchivo”**.



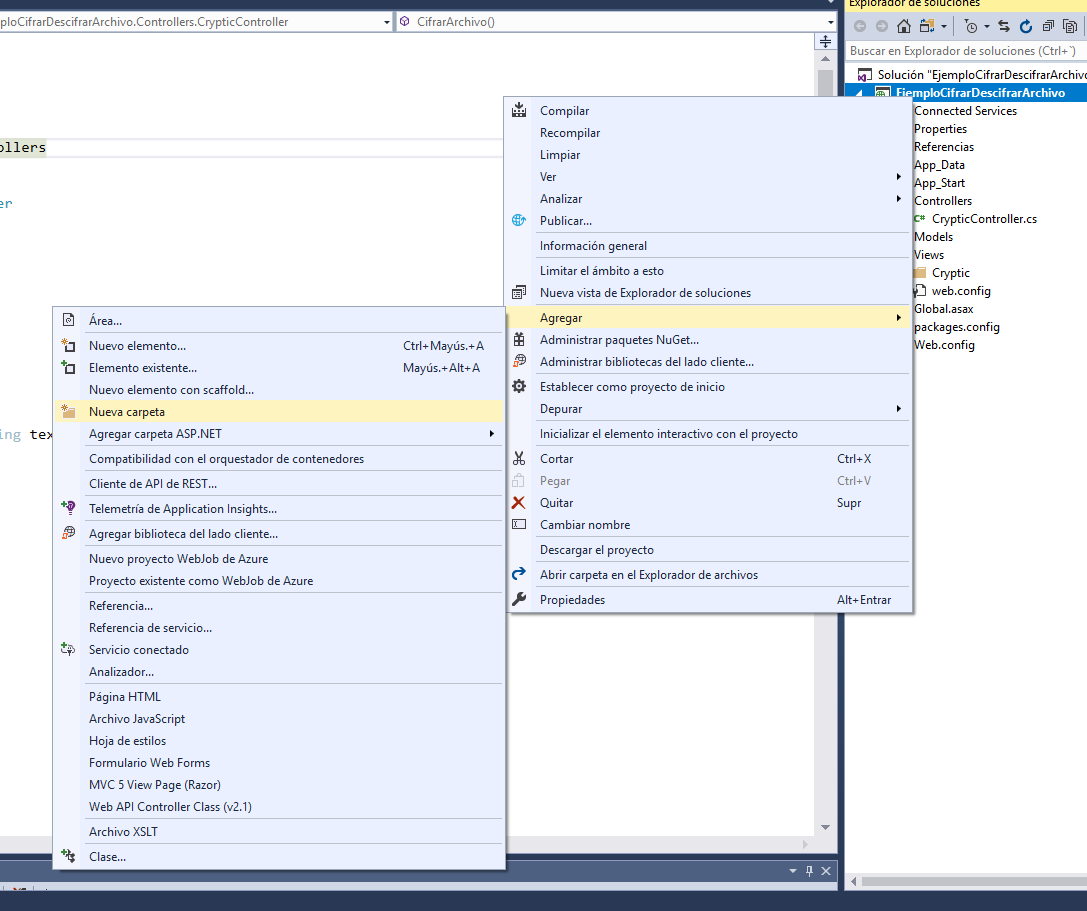
Añadimos el método *Post***,** que es el que realmente va a obtener los datos de la vista.



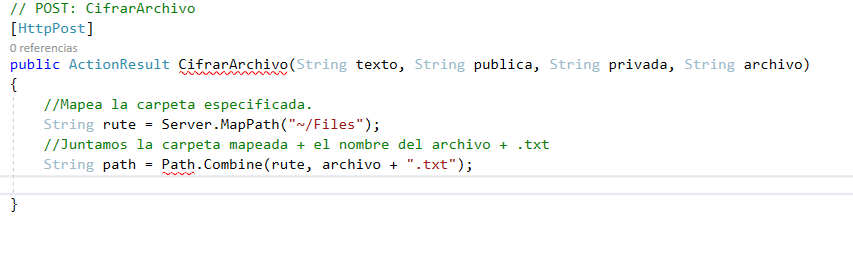
Muy importante que tenga el mismo nombre que el *Get.*

Los parámetros que contiene serán ***<input>*** de la vista cuyo *name* serán los mismos que los que estamos viendo.

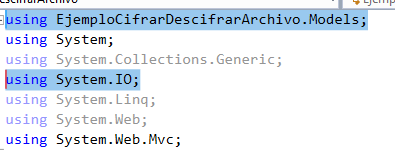
Creamos una carpeta al mismo nivel que el proyecto llamada **“Files”**.



En el método *Post* de **“CifrarAchivo”,** añadiremos el siguiente código para mapear la carpeta anteriormente creada.

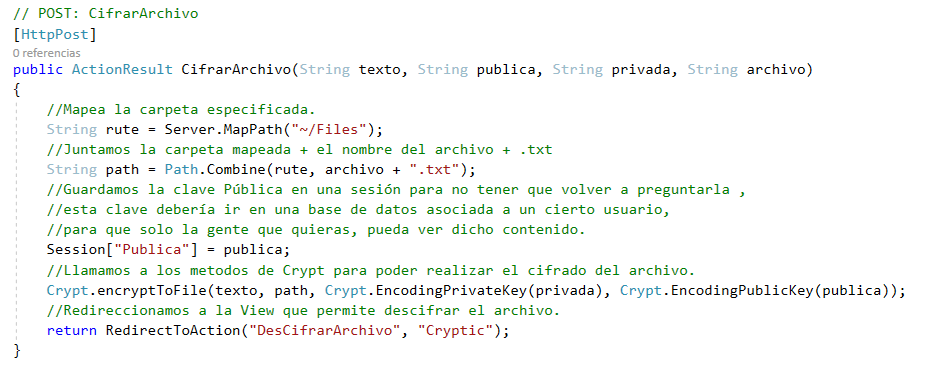


Necesitaremos incluir los siguiente *using:*



Invocamos el método **“EncryptToFile”** generado en la case **“Crypt”.**

Hay que tener en cuenta que las claves las estamos recogiendo en formato *String,* por lo que habrá que usar los métodos creados con anterioridad para pasarlos a *byte[].*



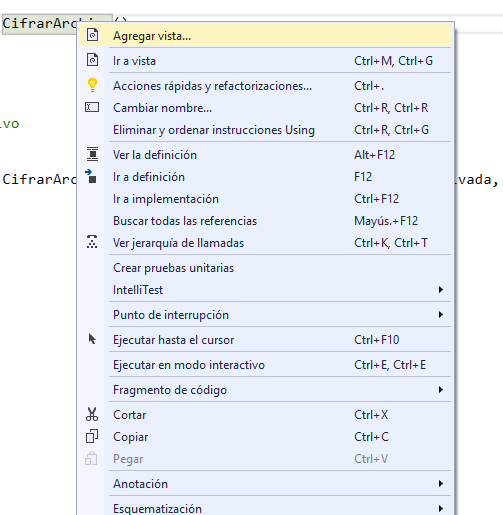
**-Nota Importante-**

En este ejemplo estoy usando *Session* para guardar la clave pública, pero esta clave debería ir en una base de datos asociada a un nombre de archivo, para que la clave no se pierda y siempre se pueda consultar, pero para no dificultar el ejemplo, he decidido utilizar *Session*.

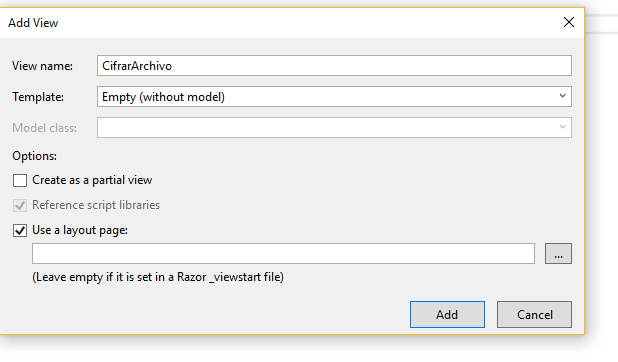
Por lo tanto, si cierra el navegado cuando esté ejecutando el programa, el archivo al no tener una clave pública ejecutará una excepción.

También se han omitido las validaciones para poder centrar este Tutorial a su finalidad final: **Cifrar y descifrar un archivo con Asp.Net MVC.**

Añadimos la **View** de la siguiente forma**.** (click derecho sobre el nombre del método)

****

Dejamos todo por defecto.



Una vez Creada la vista introduciremos el siguiente código.



Creamos los métodos *Get* y *Post* de **“DescifrarArchivo”,** muy parecidos a los métodos anteriores.

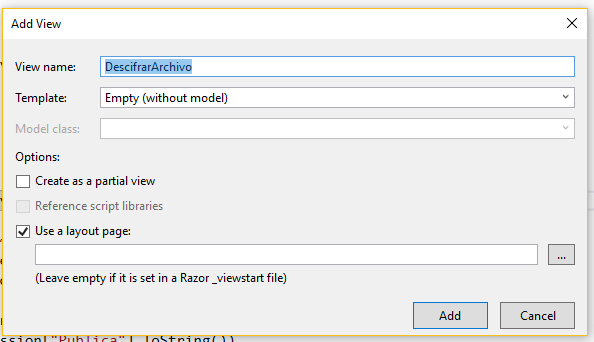


Esta vez solo recibirá los parámetros del nombre del archivo a descifrar y su clave privada.

Como el método **“DecryptFromFile”** devuelve un *String*, podemos guardarlo en un *ViewBag* para mostrar la información en un ***<textarea>*** por ejemplo.

(Recuerda que la clave pública está almacenada en *Session.*)

Añadimos la vista como antes.



Incluimos el *HTML* en la vista que hemos creado.



Con esto abríamos terminado todo lo necesario para el correcto funcionamiento del programa.

Les dejo la dirección de donde está subido el proyecto para que puedan probarlo.

También les dejo un vídeo complementario en el que les muestro el funcionamiento.

Muchas gracias por seguir el tutorial y espero que os haya ayudado.