**La importancia del middleware en el entorno web.**

El middleware se define como lógica de intercambio de información entre aplicaciones es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, o paquetes de programas, redes, hardware y/o sistemas operativos.

# Habilitar certificados, planillas preinstaladas

# 

# *dotnet dev-certs https –trust*

# Creación de Solución MVC

# 

# *Dotnet new mvc*

# *(“Este comando se debe ejecutar en la consola desde la ruta de la carpeta que quiero que me cree la solución”)*

# Creación del Modelo MVC.

# Para empezar a crear la estructura de nuestro proyecto comenzamos con la creación de la Vistas en “Views”. Allí se podrá crear una nueva carpeta que llevará por Convención el nombre el Controlador.

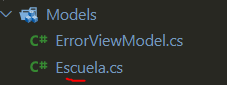
# 

# En este caso se creó una carpeta Escuela que contiene mi vista Index.cshtml

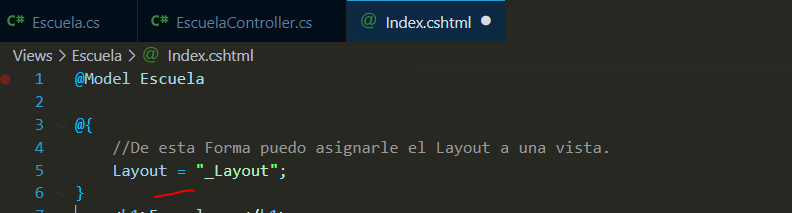
# Luego se crea el controlador dentro de la carpeta “Controllers”. Por Convención llevara el nombre de la carpeta donde están mis vistas que en este caso es Escuela.

# 

# Dentro del controlador este llevara los métodos *public IActionResult “NombreVista”*

Para la creación de Modelos se realiza desde la carpeta “Models”. Estos serán mis clases donde almacenará la información que necesita la vista para mostrar.

Para asignar mi modelo a la vista:



***VISTA GRAFICA***

***VIEWS***

***CONTROLLERS***

Folder

PerrosController.cs

Perross

Flujo de datos

Index.cshtml

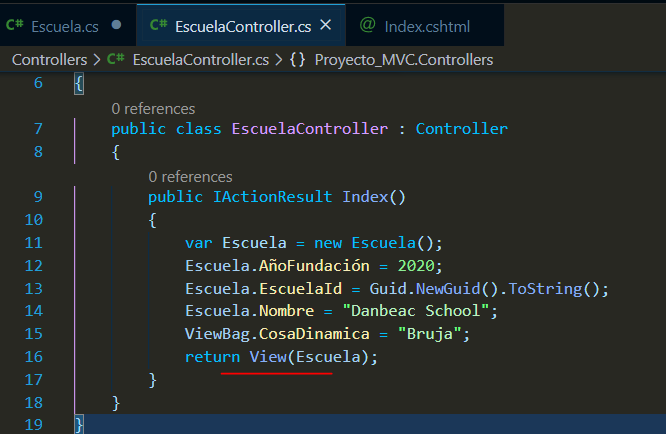
Detail.cshtml

***MODELO***

Basic.cshtml

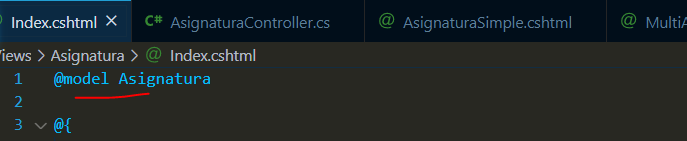
**Enviar un Modelo a Una Vista**

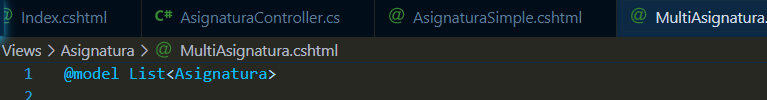
Para enviar mi modelo de datos a la vista debe ser desde el controlador. Allí cuando invoque la vista será donde yo le envié el modelo de datos.



En este caso al ser invocada la vista “Index” a través del controlador envió el modelo Escuela.

Puedo asignar el modelo que recibe mi vista de la siguiente manera.





# Asignación de Layout a una vista.

# 

# ViewBag.

# Esta es la forma en que puedo enviar información a la vista.

# 

# De esta manera se puede llamar desde la vista.

# 

# 

# Formularios Post.

# Existen varias formas de enviar los datos de un formulario por post a nuestro server:

# Definiendo parámetros como variables

# Recibir datos en el Controlador, es tan sencillo como definir un parámetro por cada variable de formulario en la Acción correspondiente. La única condición requerida, es que el nombre de las variables (atributo name= de la etiqueta HTML) coincida con el nombre del parámetro definido en la Acción (sensible a mayúsculas y minúsculas).

# 

# 

# Mediante la clase IFormCollection.

# En ciertas ocasiones, el número de variables de formulario que debemos enviar al servidor es bastante grande, para estos casos, ASP.NET MVC nos proporciona la clase IFormCollection, a través de la cual podemos acceder a una colección del tipo Clave/Valor formada por los elementos contenidos en el Form desde el que se realiza el POST.

# 

# 

# Usando un Modelo de datos (Model binding).

# Para más información:

# <https://www.rafaelacosta.net/Blog/2018/10/14/c%C3%B3mo-recibir-datos-desde-un-formulario-post-en-aspnet-mvc>

# Vistas Parciales.

# Para mostrar una vista parcial utilizamos el TagHelper “*<partial>*”

# Dentro de “*name”* especificamos como se llama la Vista que vamos a invocar.

# 

# Podemos también especificar el modelo “*model*” que se va a enviar al invocar la vista de la siguiente manera:

# 

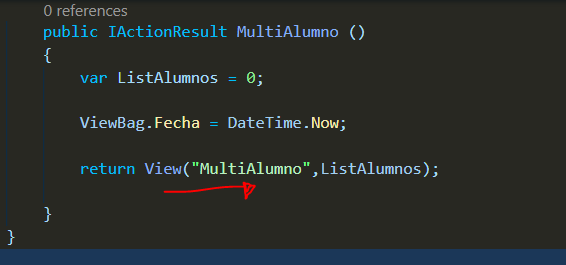
# En este caso le especificamos que el modelo que va a recibir la vista *AsignaturaSimple.cshtml* es la variable *asignatura* recorrida del Foreach.

# 

# *AsignaturaSimple.cshtml*

**Especificar vista enviada por el controlador.**

De esta manera al invocar el controlador de una vista puedo especificar la vista que devolverá:



# Introducción a entity framework.

# En nuestra consola situados en la carpeta donde tenemos nuestra solución ejecutamos los siguientes comandos para que descargue los componentes necesarios.

# 

# dotnet add package Microsoft.EntityFrameworkCore.InMemory dotnet add package Microsoft.EntityFrameworkCore.SqlServer

# Para una explicación detalla revisar el siguiente link:

# https://docs.microsoft.com/es-es/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

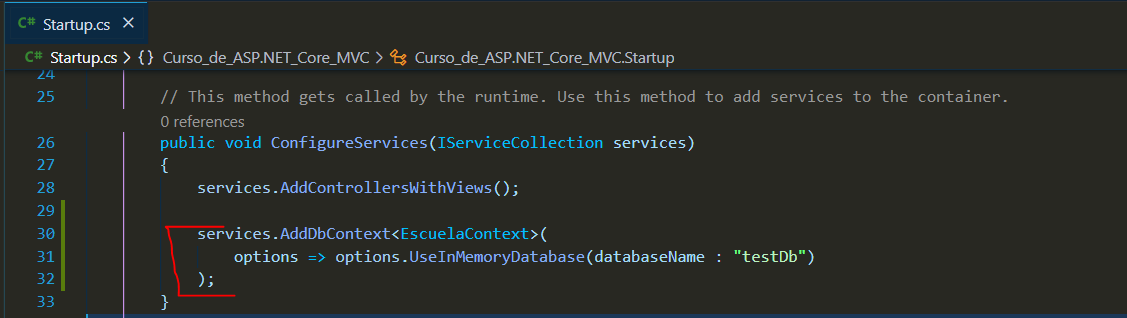
# Luego de haber instalado nuestros paquetes necesarios para hacer uso de Entity Framework, creamos nuestra clase entidad en este caso es *EscuelaContext*

# 

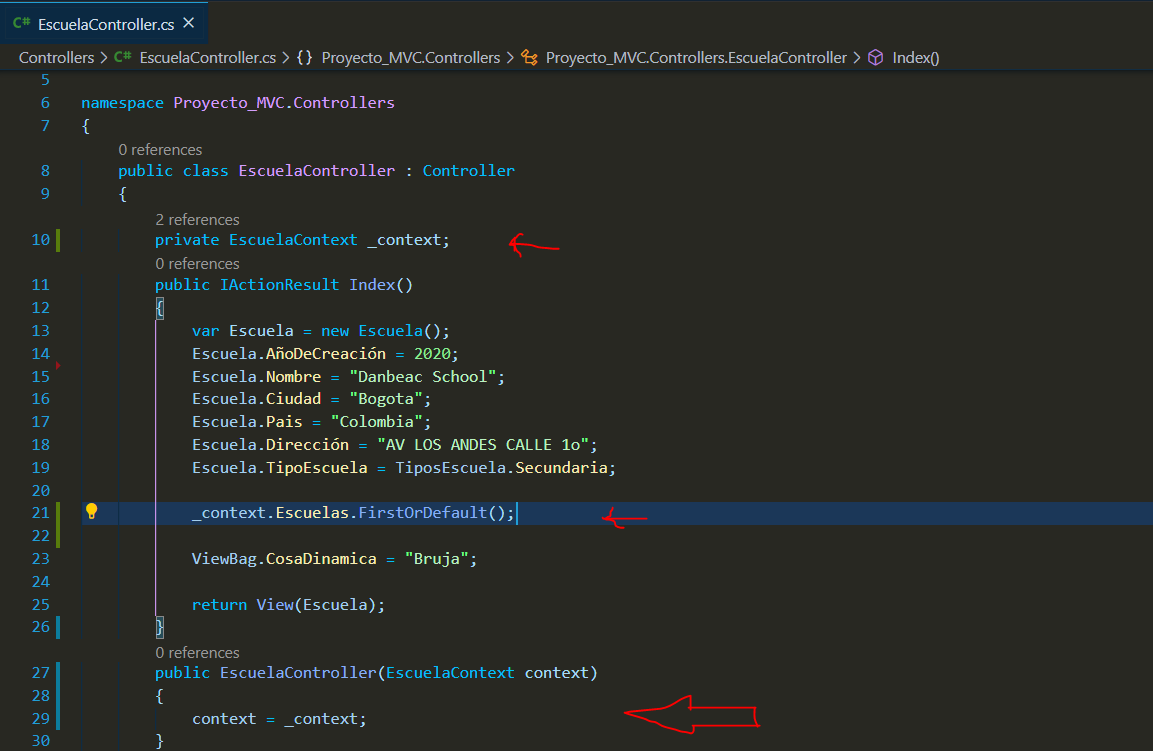
# Luego le construimos el constructor especial quedando de la siguiente forma.

# 

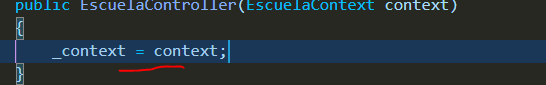
1. Después configuramos el servicio en el archivo del Proyecto *Startup.cs*. en el método ConfigureServices.



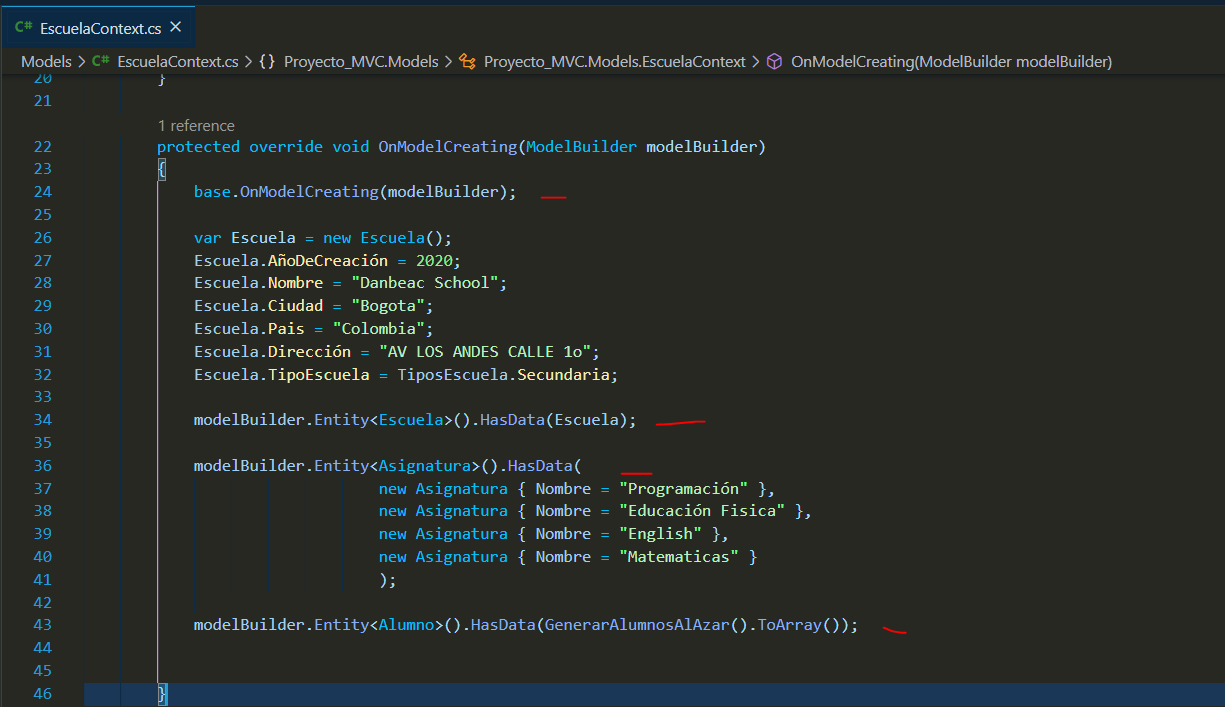
1. Luego en el controlador en el que vamos a hacer uso de EscuelaContext modificaremos el constructor recibiéndolo como un parámetro al iniciarse.



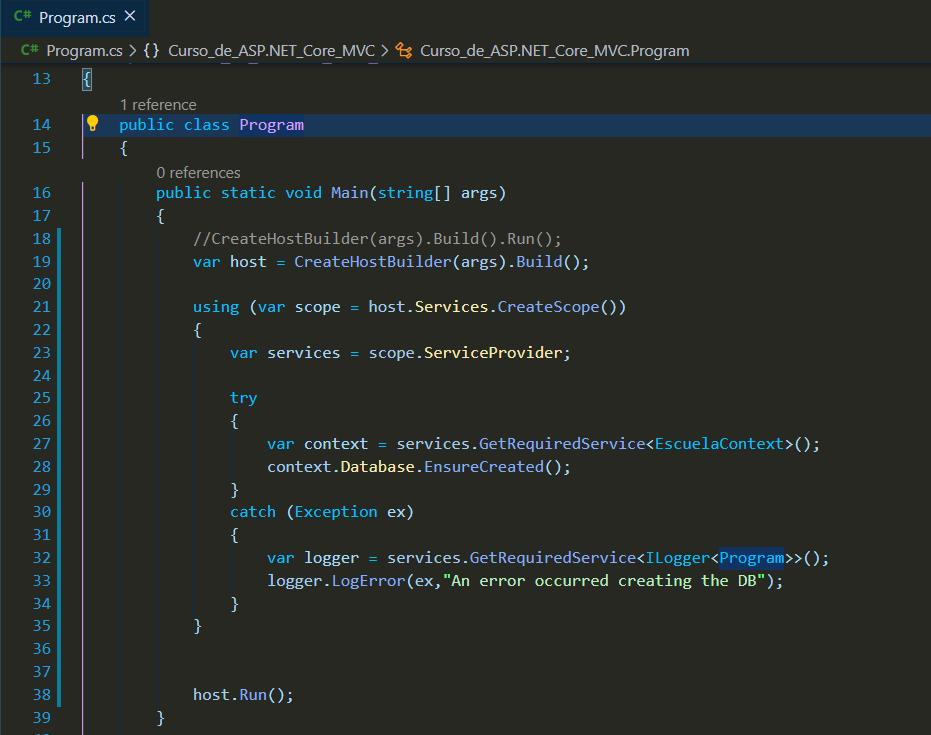
***Corrección: la asignación del context es inversa.***



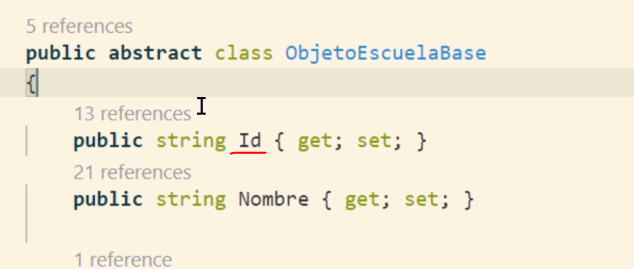
1. Luego sembraremos datos en nuestro contexto. Para ello vamos a sobreescribir un método OnModelCreating en EscuelaContext, este método se ejecuta cuando se esta creando la base de datos entonces será allí donde crearemos nuestra información de Prueba.



1. Luego configuramos en donde se inicia nuestra App, asegurándonos que cuando queramos consultar nuestros datos estén creados correctamente. Para ello haremos uso del siguiente código.

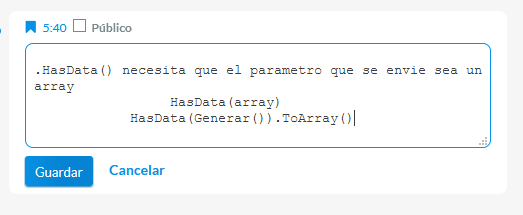


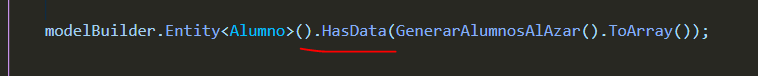
1. Debemos asegurarnos que nuestros Objetos cuenten correctamente con los ***Id.*** Por convención podremos colocar NombreObjeto**Id** o solo **Id.**



*En este caso también podríamos colocar* ***ObjetoEscuelaBaseId***

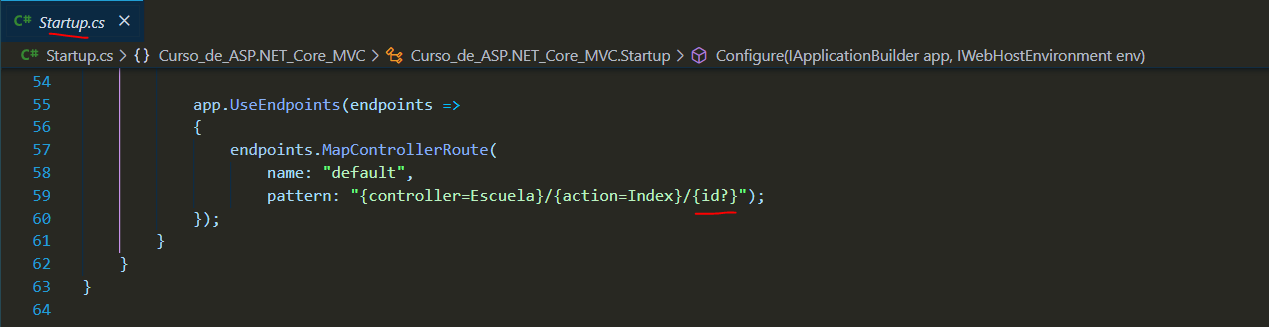
***HasData()***



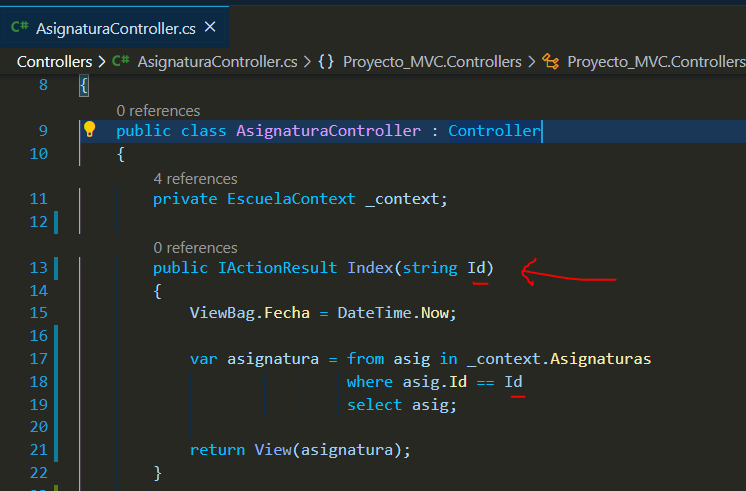


# *Ajustes de enrutamiento.*

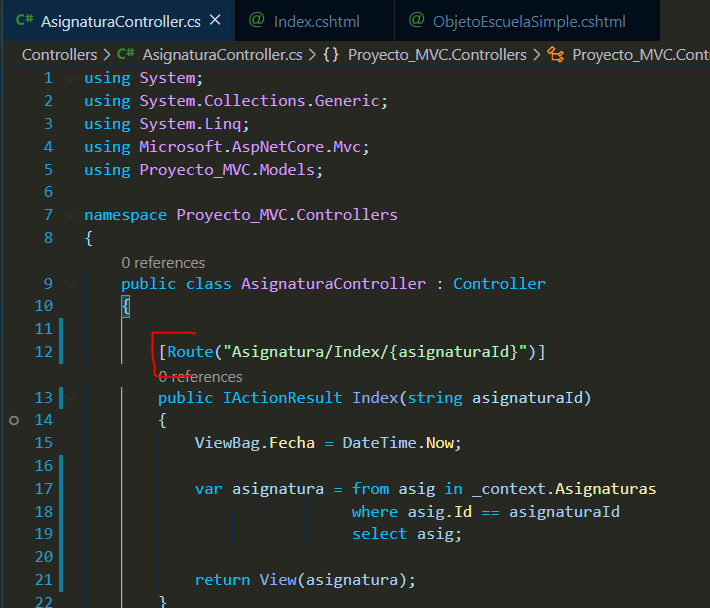
Para modificar que una vista me reciba parámetros debo revisar la ruta de enrutamiento desde el archivo ***Startup.cs***



En este caso desde la configuración tengo un parámetro con símbolo de interrogación -***id?***-que simboliza que es un parámetro opcional. Ahora desde el controlador de la vista agregando el parámetro con ese mismo nombre lo reconocerá:

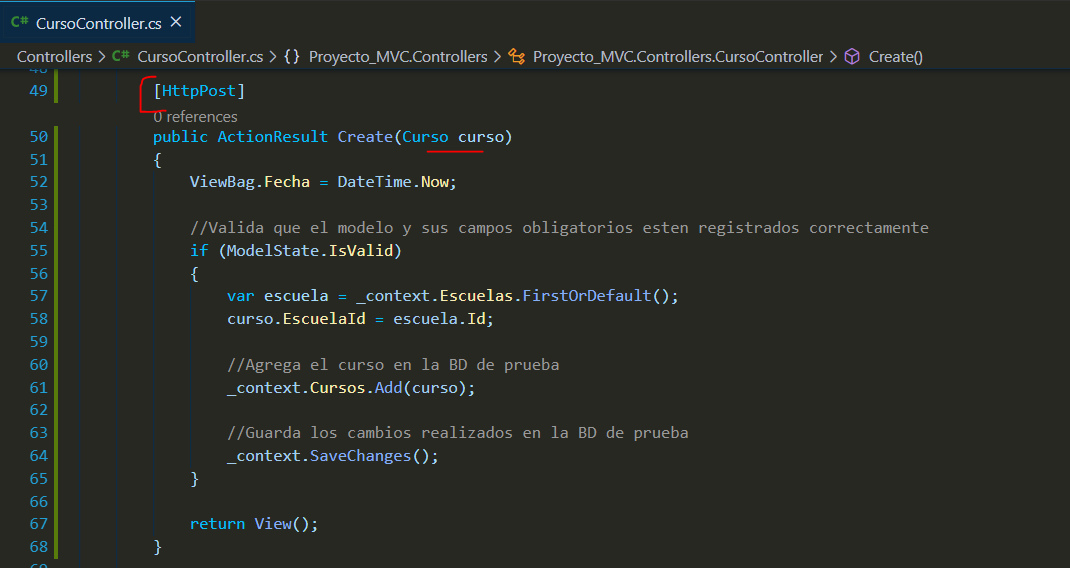


Sin embargo existe otra manera de cambiar el enrutamiento desde el controlador, quedando de la siguiente manera:

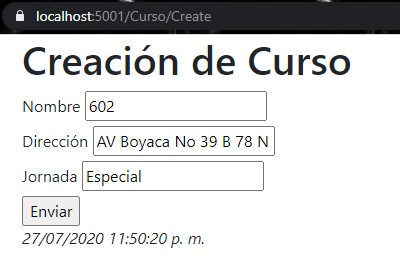


***Formularios Post***

Desde el controlador debo crear otro método ActionResult con la etiqueta [HttpPost] y configuro el parámetro que va a recibir. En este caso el formulario guardará un curso.



Desde la vista puedo configurar lo que me va a guardar de acuerdo con los campos de los objetos.

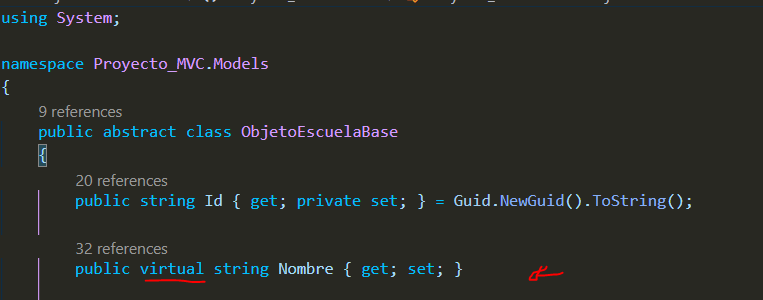
 

Para consultar mas formas de enviar los datos por POST consultar:

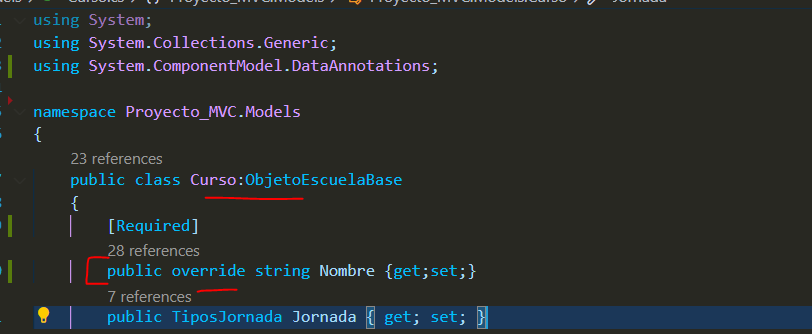
<https://www.rafaelacosta.net/Blog/2018/10/14/c%C3%B3mo-recibir-datos-desde-un-formulario-post-en-aspnet-mvc>

***Sobreescribir atributos Padre – Hijo.***

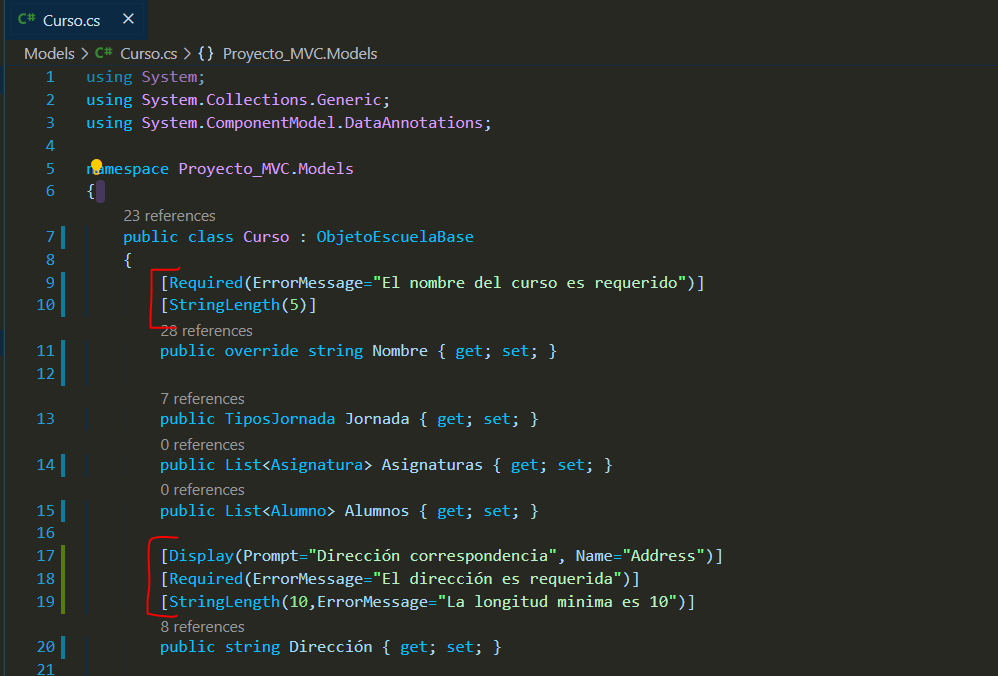
Identificando el atributo con la palabra clave “***virtual***” me permitira que en sus objetos hijos este atributo podrá ser reescribido.

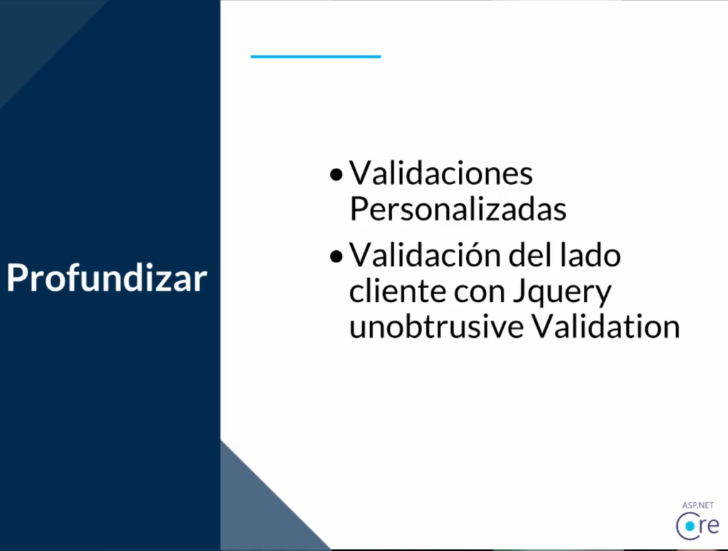


En la clase hija puedo sobreescribir el atributo con la palabra clave ***override*** y en este caso fue sobreescrito para obligar que el Nombre solo para mis Cursos oblique a que el campo sea requerido [Required]



***Data Annotations.***

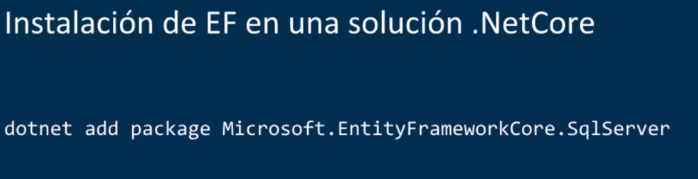




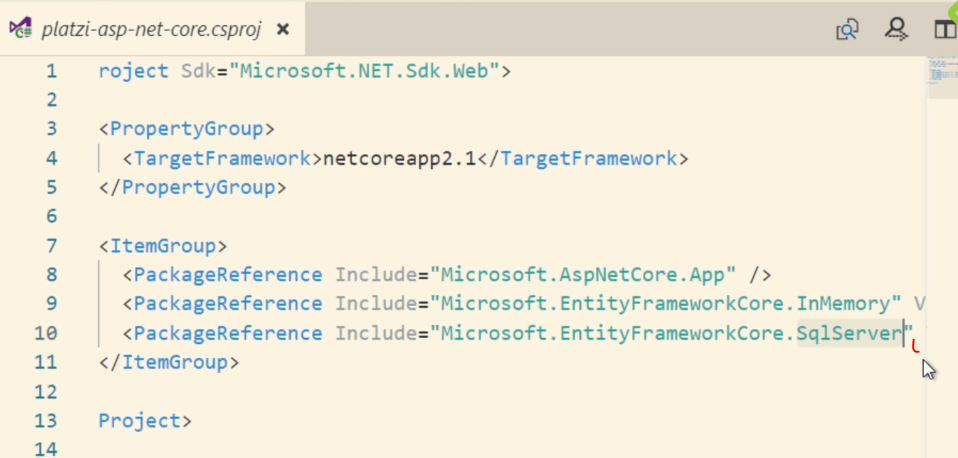
[*https://exceptionnotfound.net/asp-net-mvc-demystified-unobtrusive-validation/*](https://exceptionnotfound.net/asp-net-mvc-demystified-unobtrusive-validation/)

***Configurar proyecto a base de datos.***

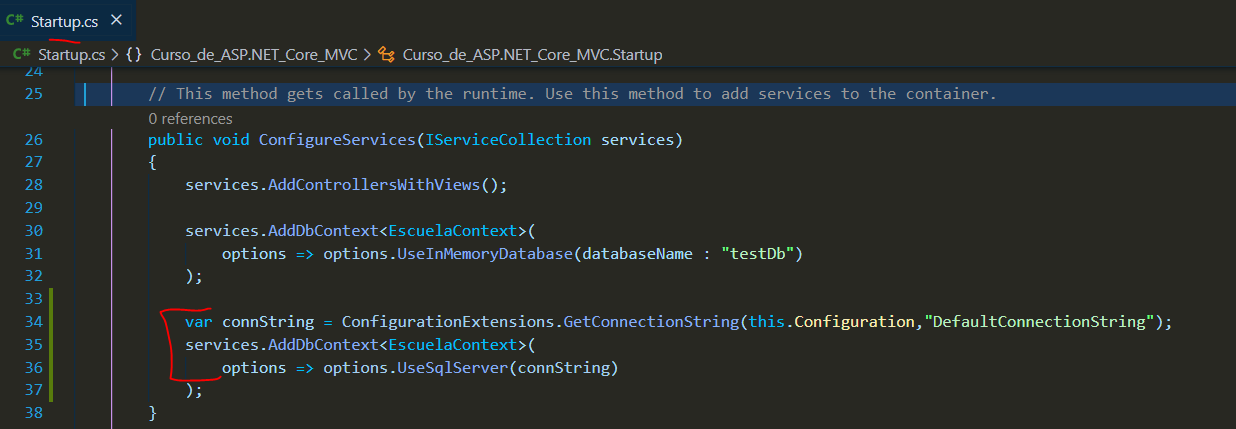
1. Instalamos el paquete de SQL Server



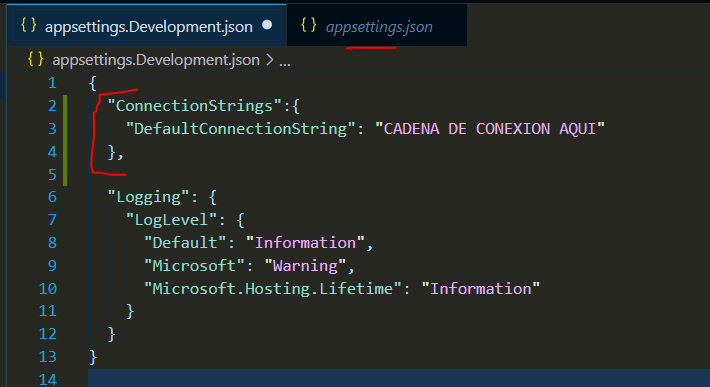
*dotnet add package Microsoft.EntityFrameworkCore.SqlServer*



1. En el archivo *Startup.cs agregamos el servicio:*



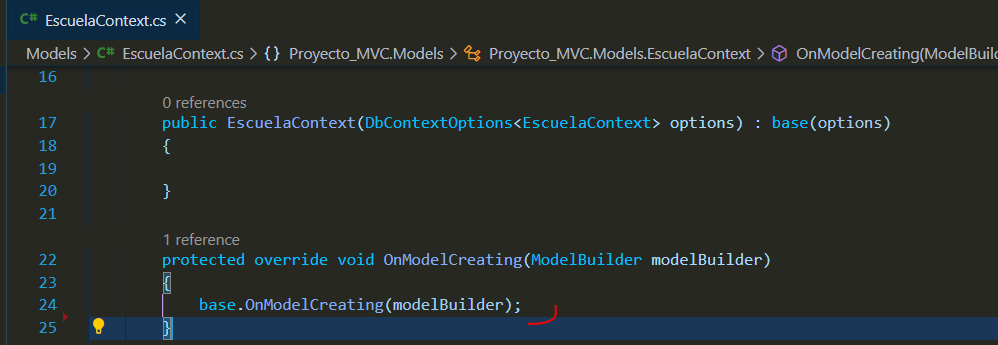
1. Luego configuraremos las cadenas de conexión en nuestros dos archivos desarrollo/producción, agregando el código que se remarca:



<https://www.connectionstrings.com/sql-server/>

Link sobre cadena de conexión.

1. Luego de agregar la cadena de conexión ya no es necesario la siembra de datos por lo que dejaremos vacío nuestro OnModelCreating de nuestro contexto, por lo que tendremos que ya tener creada nuestras tablas en la BD para que no nos genera ninguna inconsistencia.



# *Scaffolding.*

# Nos permite crear la estructura necesaria de nuestro proyecto de una manera mucho mas rápida.

# 

# dotnet tool install -g dotnet-aspnet-codegenerator dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design

# dotnet add package Microsoft.EntityFrameworkCore.Design dotnet restore

# En los comandos a ejecutar “*-g*” indica que al ejecutarlo se instalarán los componentes globalmente por lo que el primer comando no es necesario ejecutarlo en cada proyecto ya que se instala globalmente en el equipo.

# 

# dotnet aspnet-codegenerator controller -name AlumnoController -m Alumno -dc EscuelaContext --relativeFolderPath Controllers --useDefaultLayout --referenceScriptLibraries -f

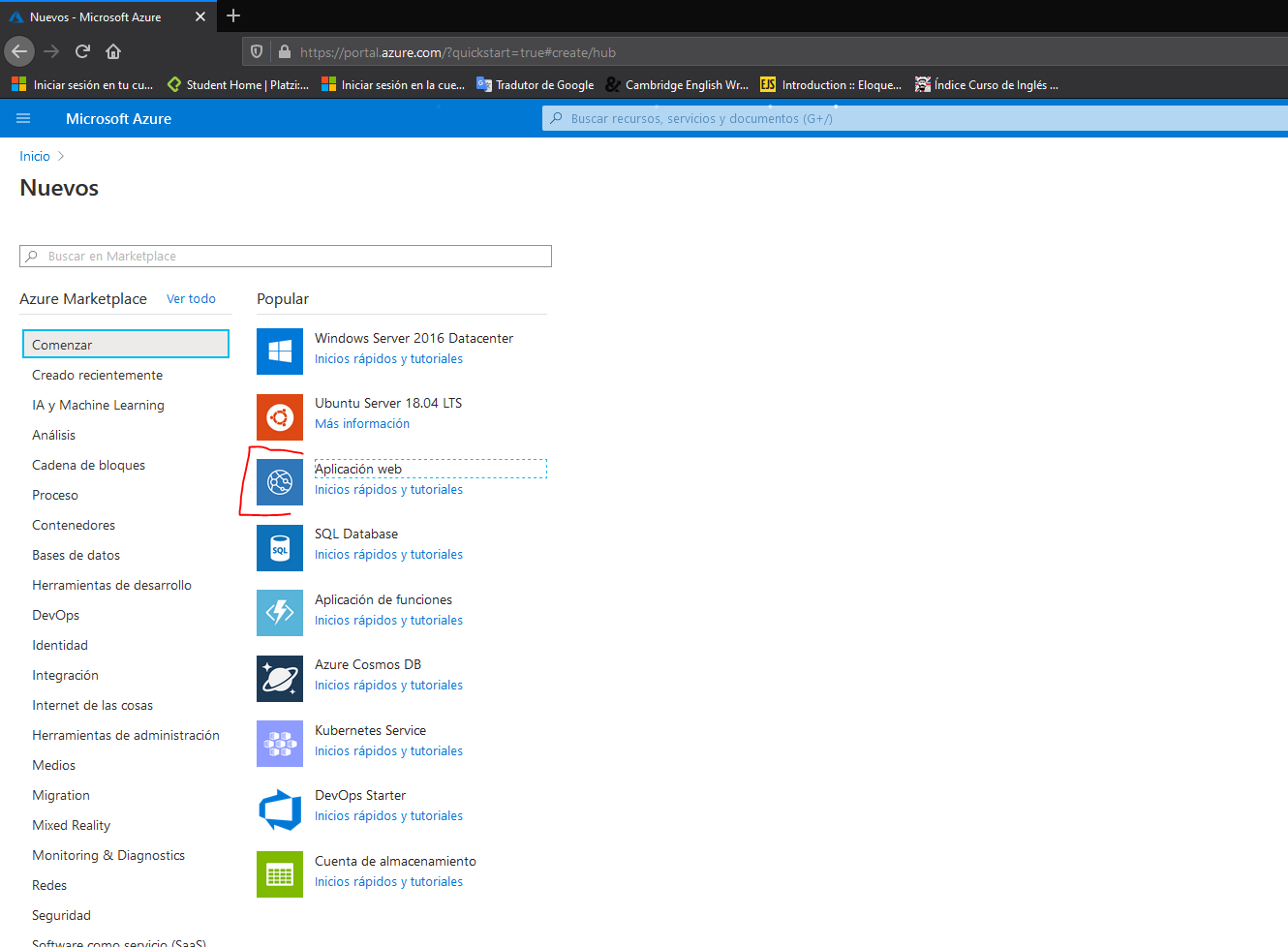
# En caso de generar algún error en este punto por versionamiento, dirigirse a asp.net y descargar el SDK otra vez.

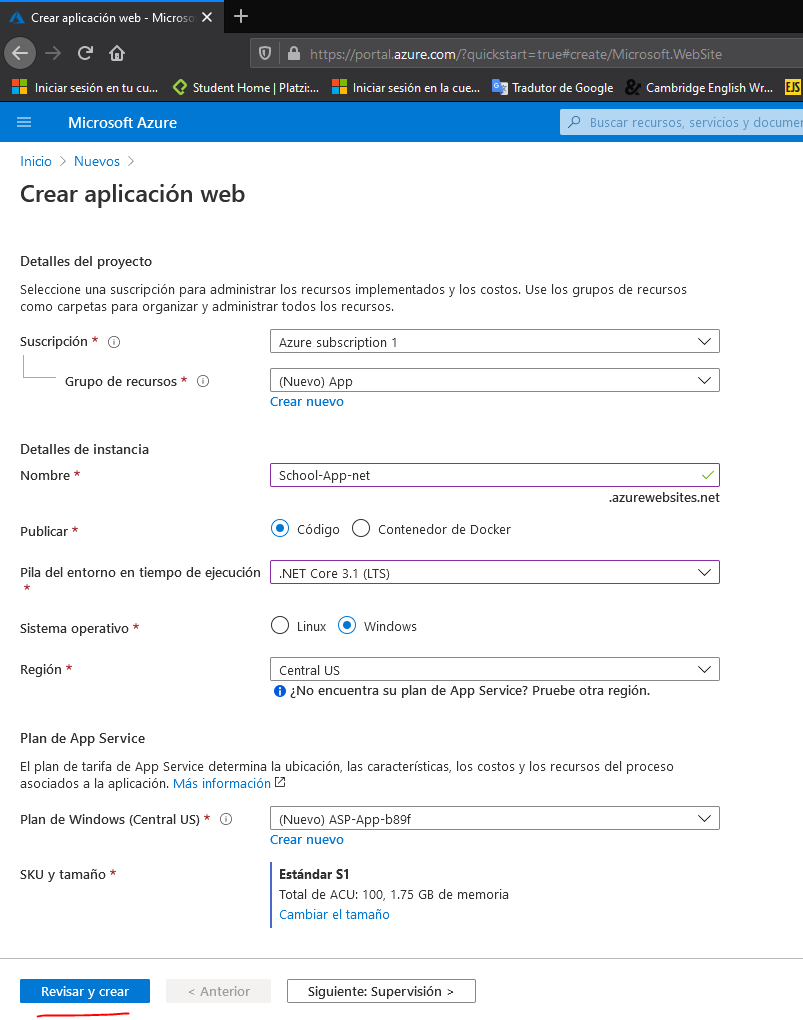
# 

# 

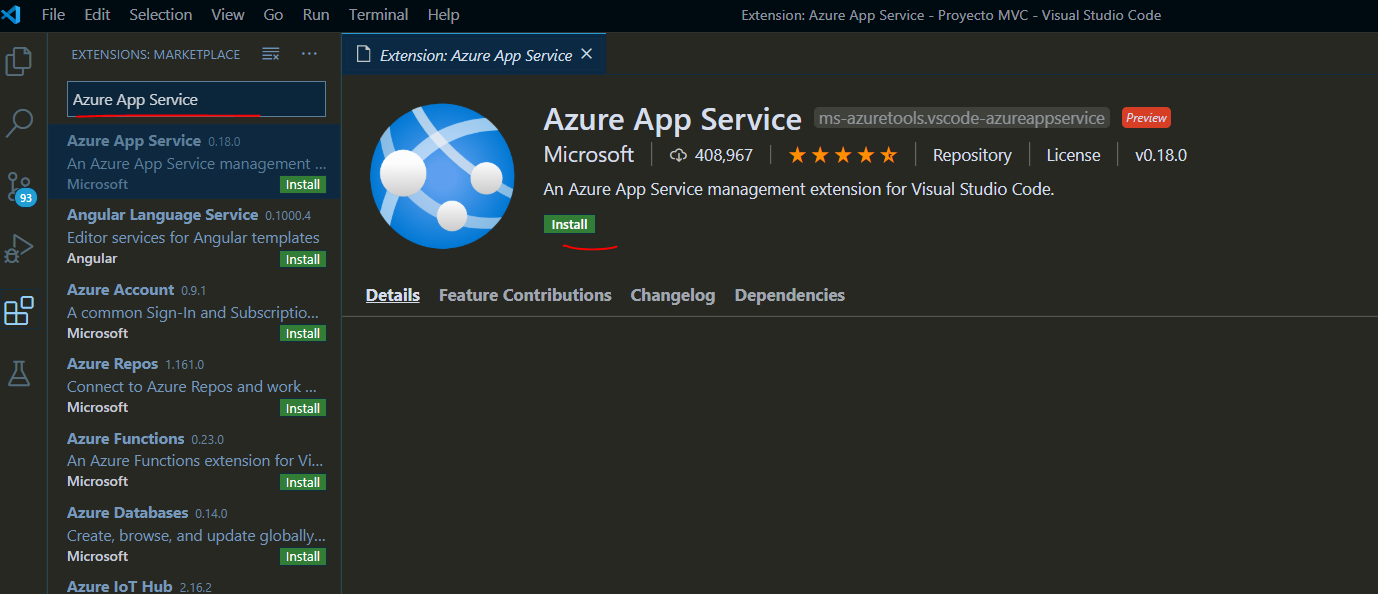
***Publicación del WebSite.***

1. Ingresamos a la cuenta de Azure y creamos una nueva Web App o una Aplicación Web.

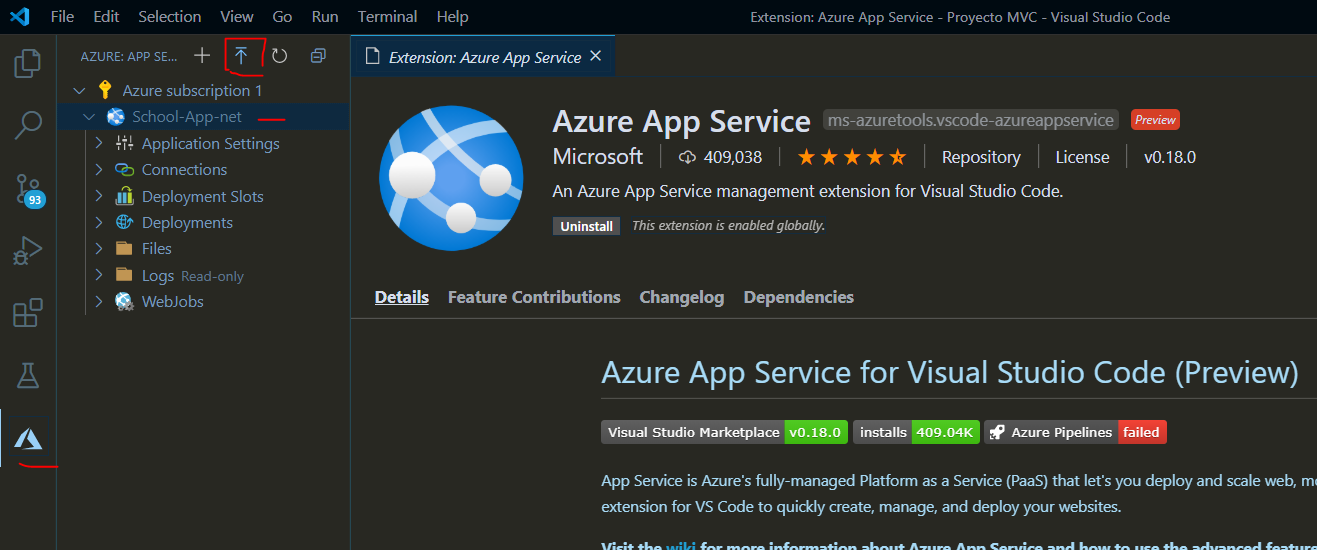




1. Revisamos que nuestro VS Code tenga instalada la extensión de Azure.

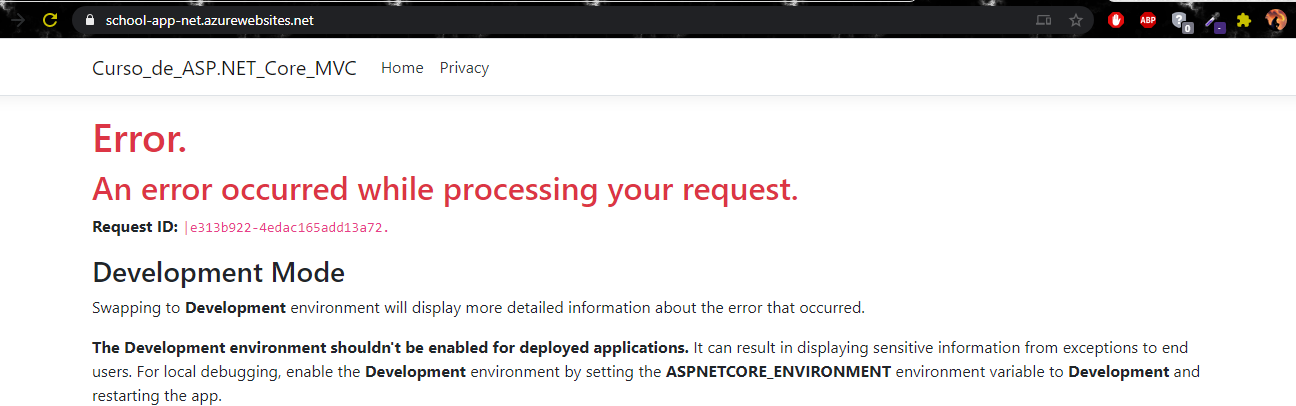


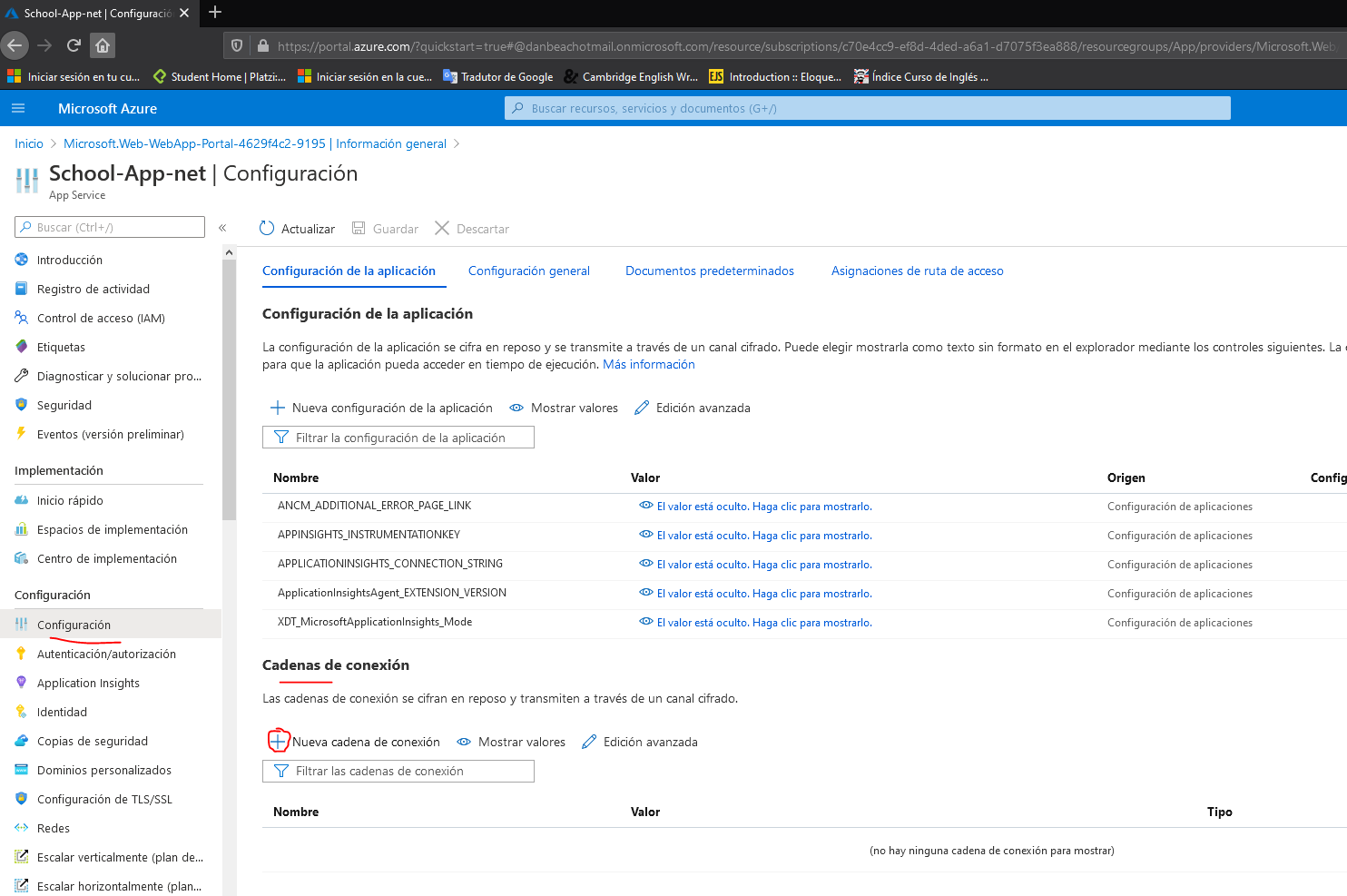
1. Desde el icono de Azure, iniciamos sesión y buscamos el sitio que creamos anteriormente y le damos *Upload* en la flecha como se muestra:

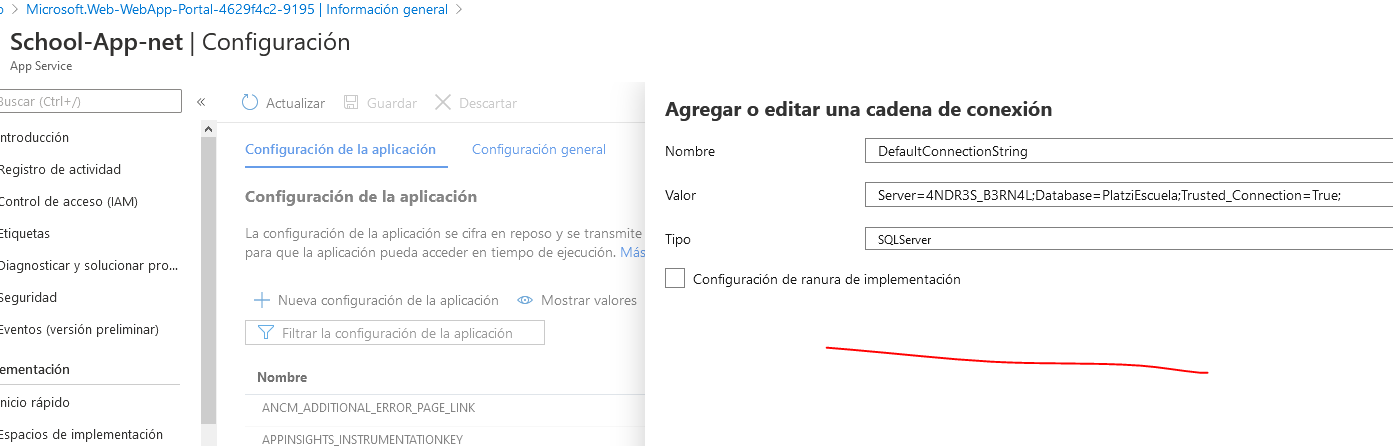


En caso del siguiente error debemos configurar las cadenas de conexión desde Azure:

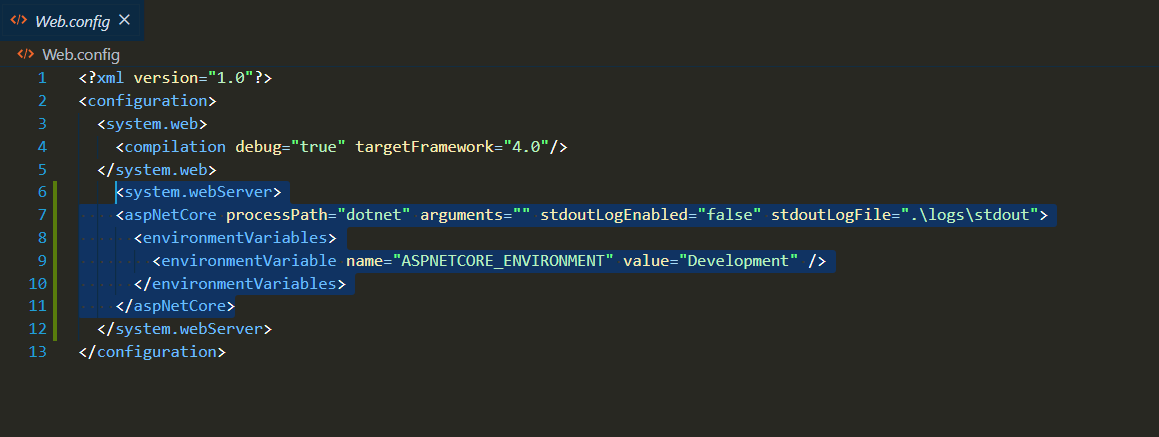
Error:







En caso de que no se solucione, para que las alertas se muestren en producción debemos cambiar el web config agregando lo resaltado y luego hacer release nuevamente :



<system.webServer>

    <aspNetCore processPath="dotnet" arguments="" stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout">

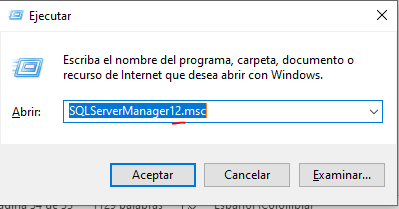
      <environmentVariables>

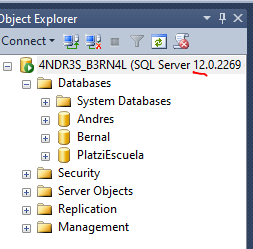
        <environmentVariable name="ASPNETCORE\_ENVIRONMENT" value="Development" />

      </environmentVariables>

    </aspNetCore>

Revisar también que todos los servicios de la BD estén funcionando correctamente. Para ingresar a el Configuration Manager de Sql:

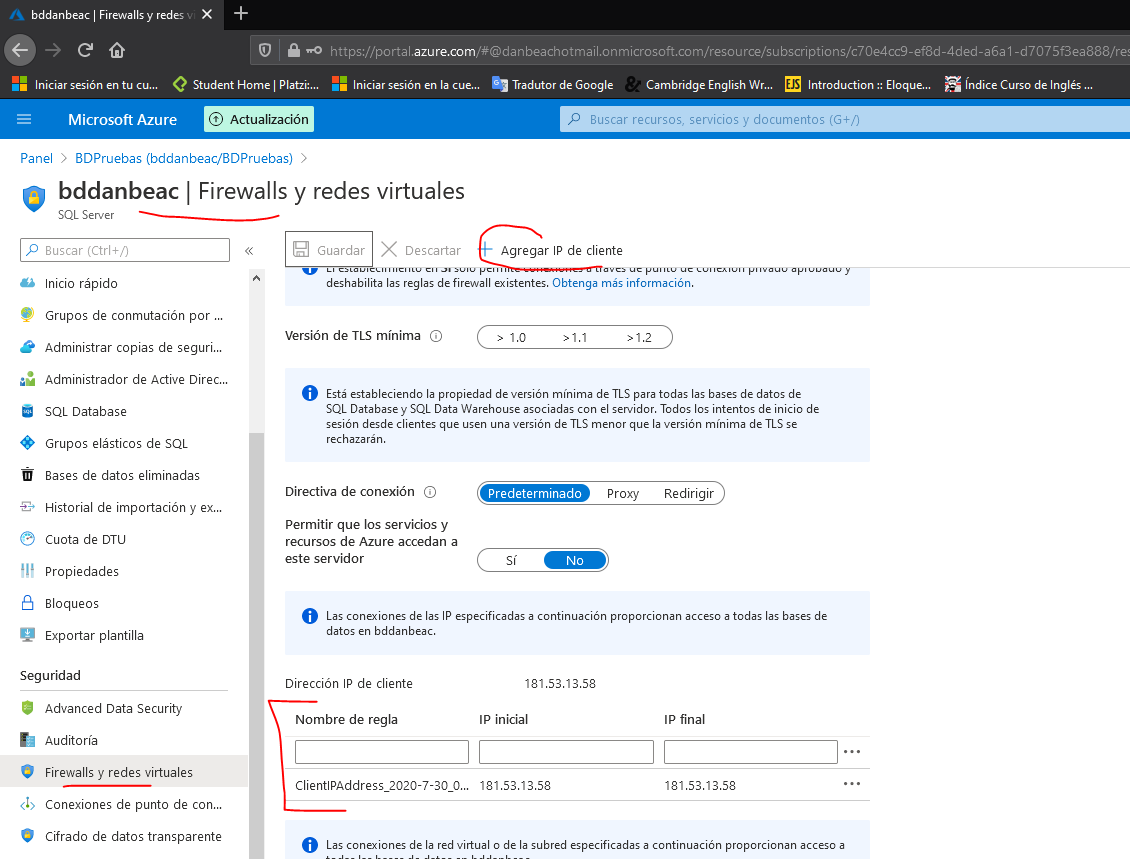




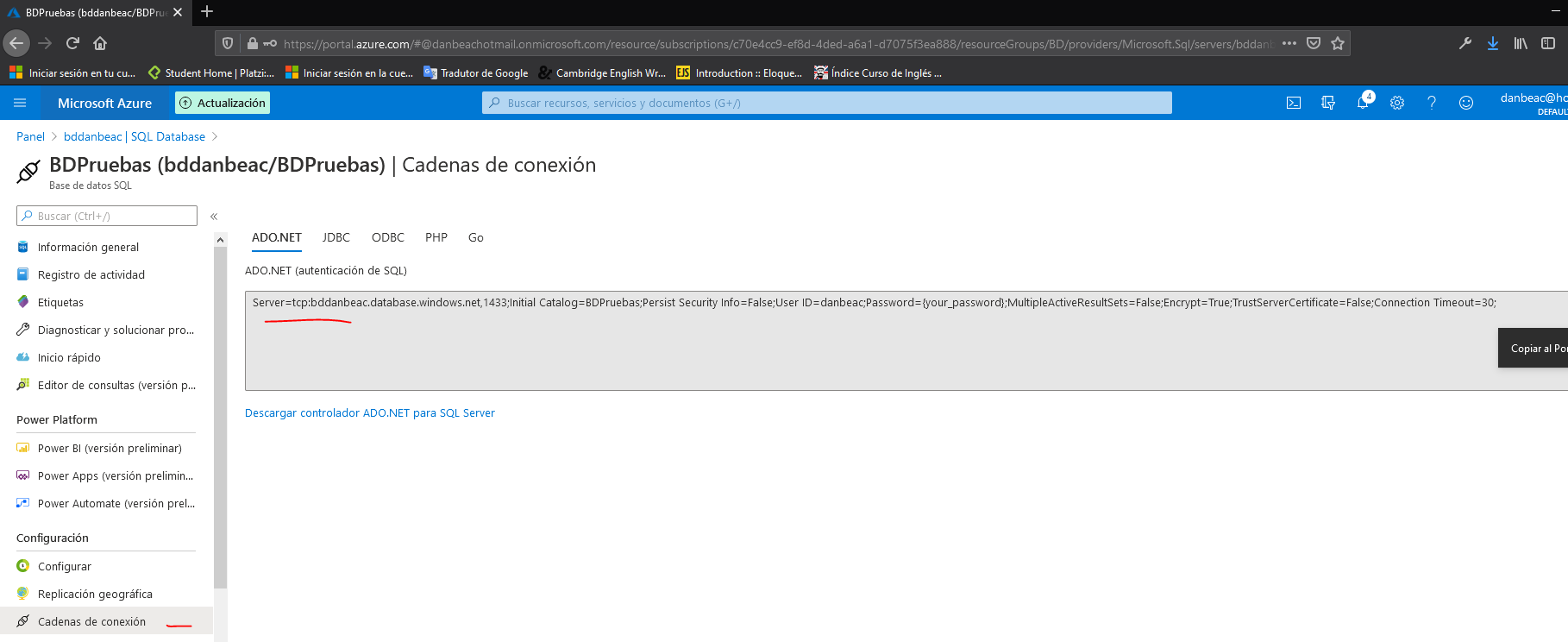
El numero 12 puede variar dependiendo de la versión de SQL que tengamos.

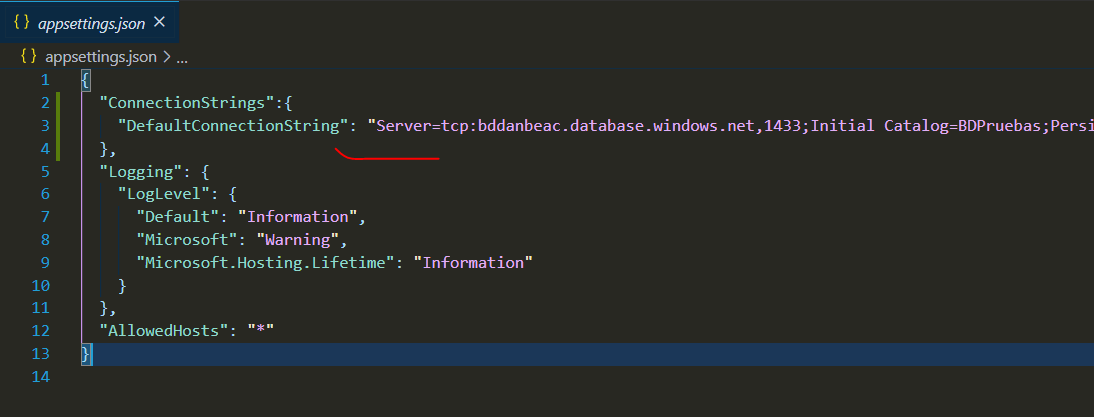
***Implementación BD Azure.***

Luego de crear mi base de datos y mi servidor en Azure debo configurar las IP a las cuales podrán acceder.



Luego copiamos la cadena de conexión a nuestro proyecto.





***Sincronizar Base de Datos Azure con una Local.***

<https://www.sqlshack.com/es/como-sincronizar-bases-de-datos-sql-de-azure-y-bases-de-datos-locales-con-sincronizacion-de-datos-sql/>

