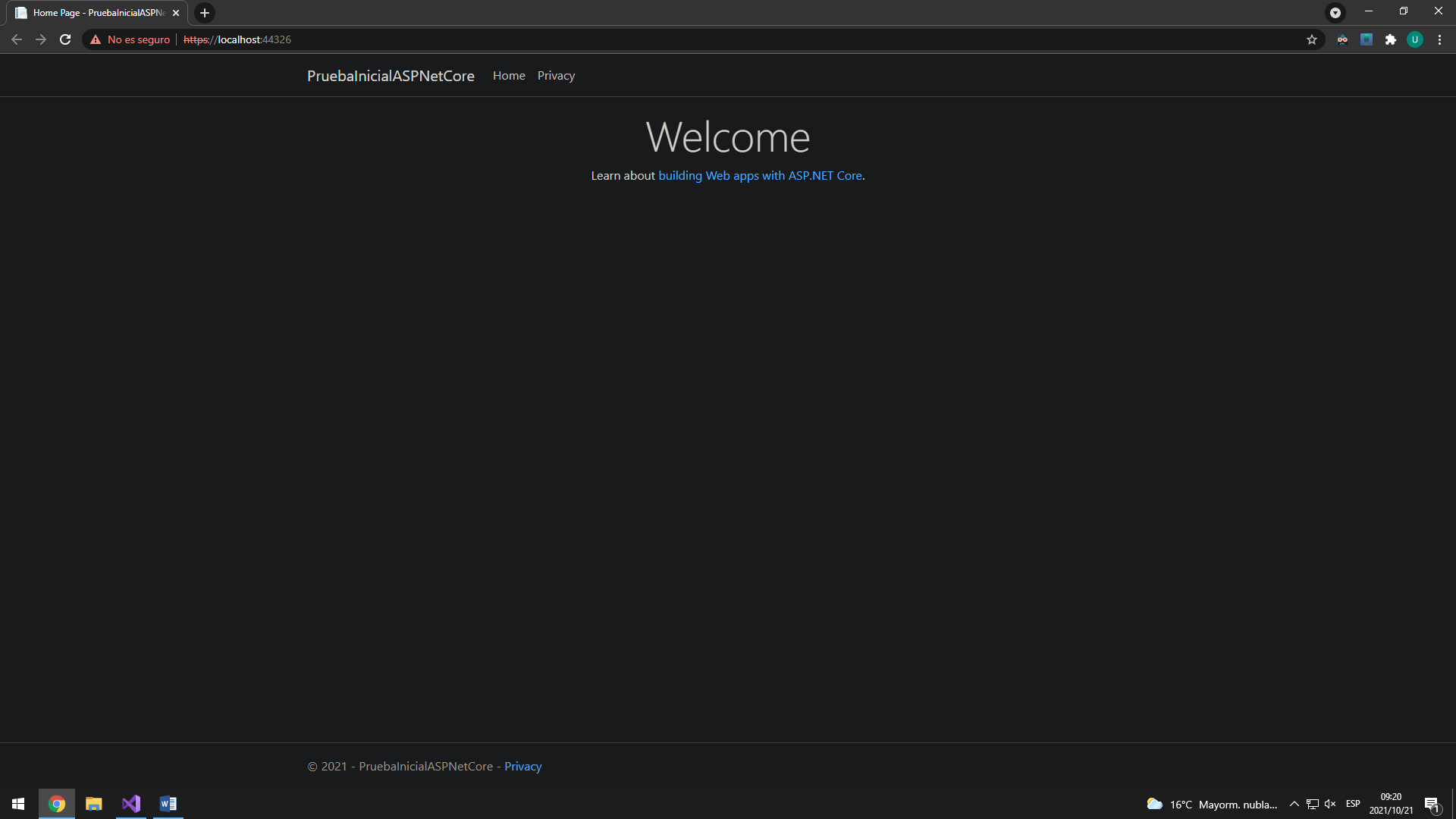
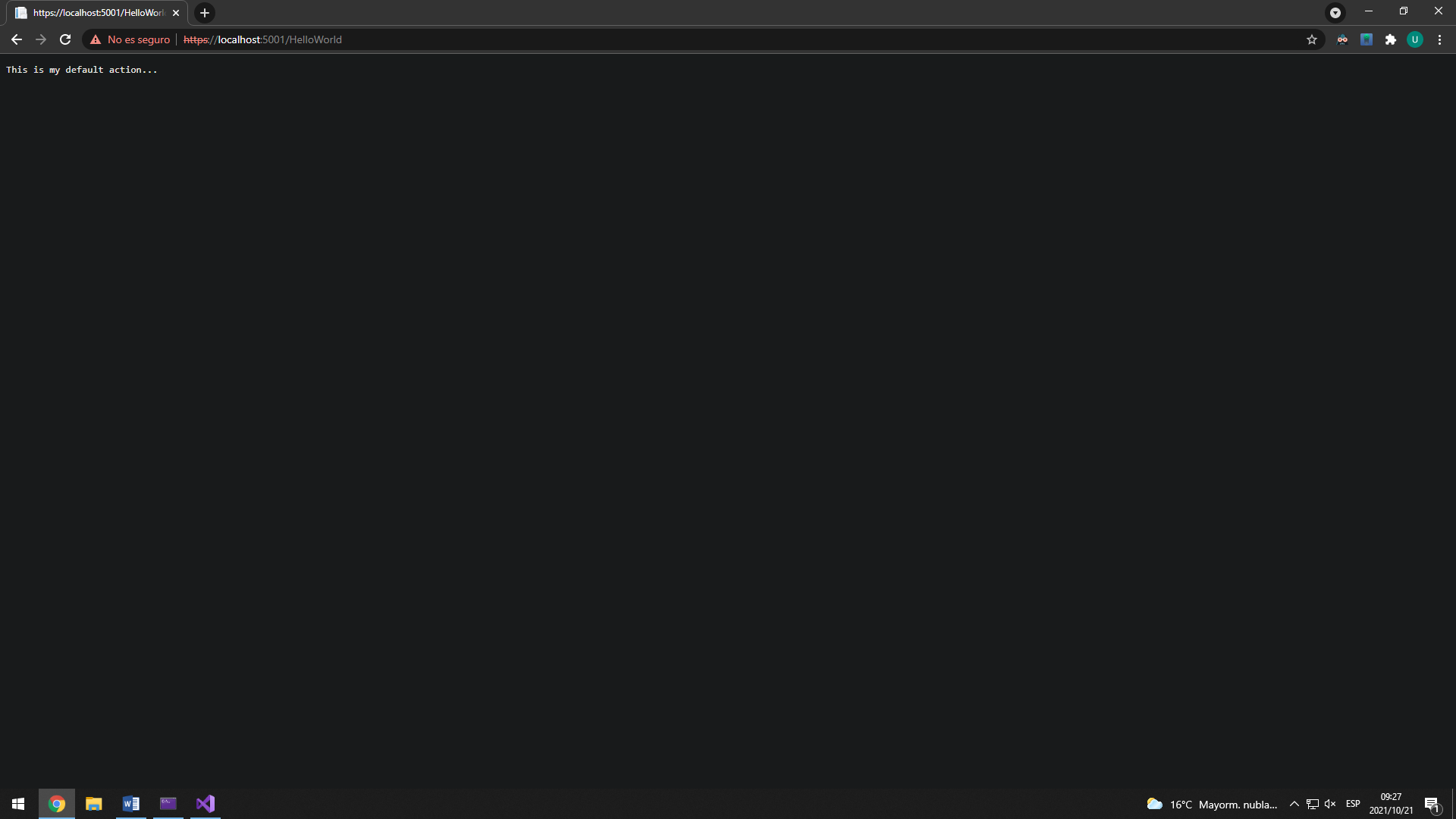
## Paso 1: Inicio

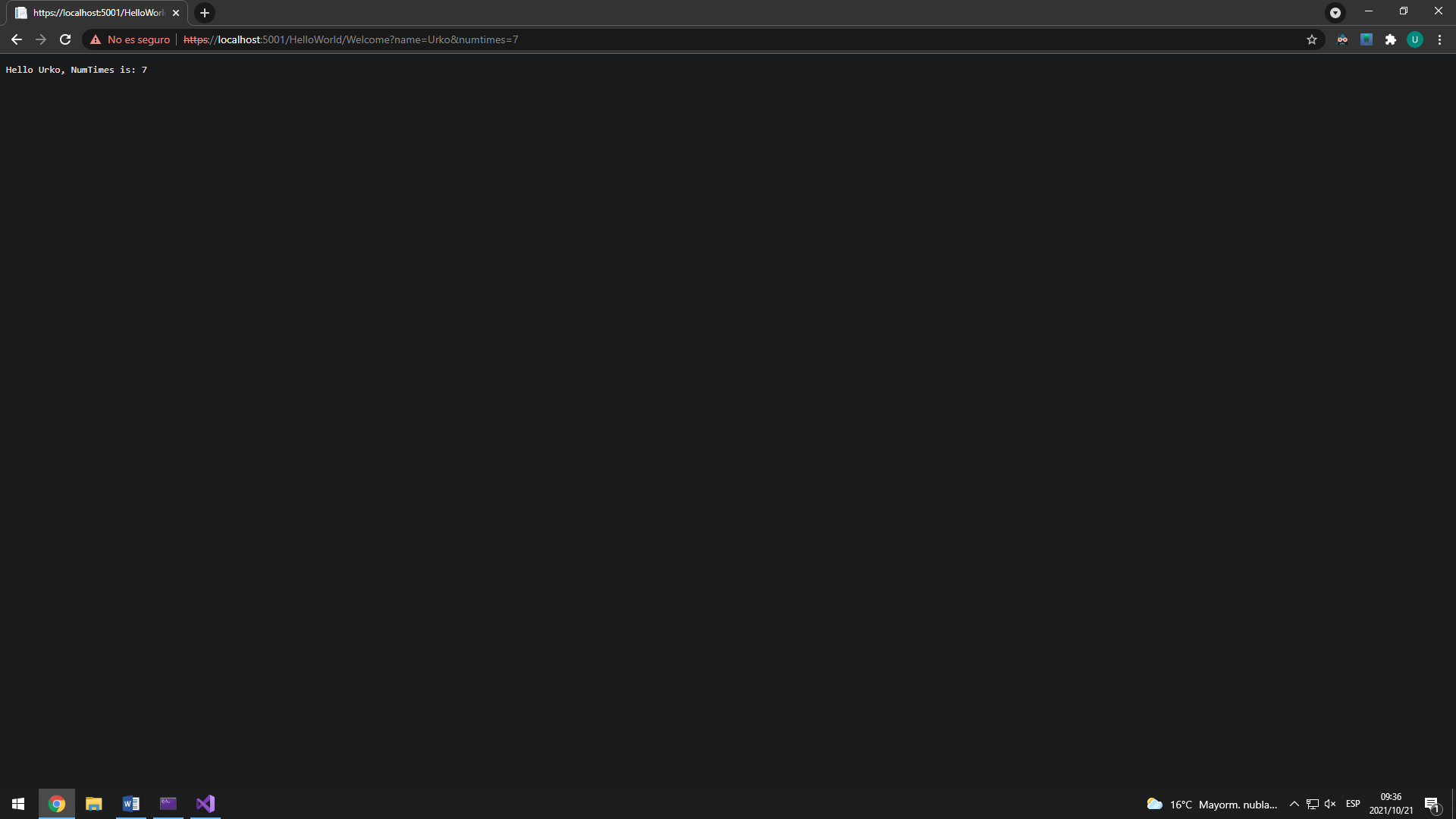
Hemos aprendido a crear un nuevo proyecto MVC con ASP .Net Core

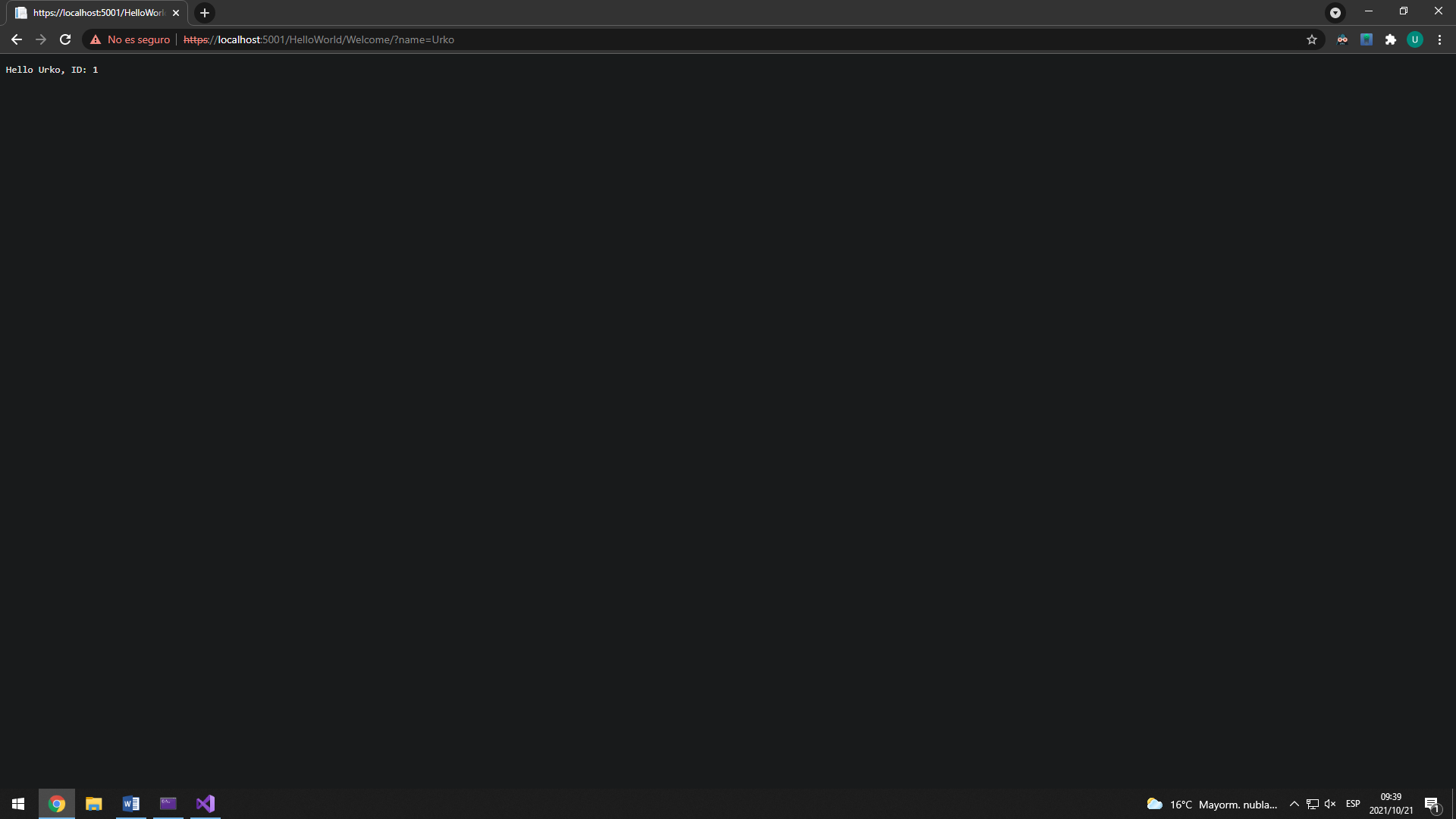


## Paso 2: Controlador Hello World

Hemos aprendido a crear un controlador que reciba algunos parámetros desde la url, modificarlo y acceder a su vista

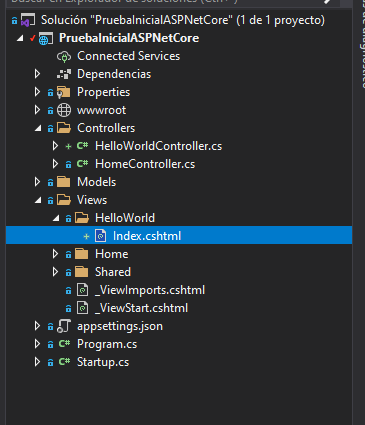
https://localhost:5001/HelloWorld/ 

https://localhost:5001/HelloWorld/Welcome?name=Urko&numtimes=7

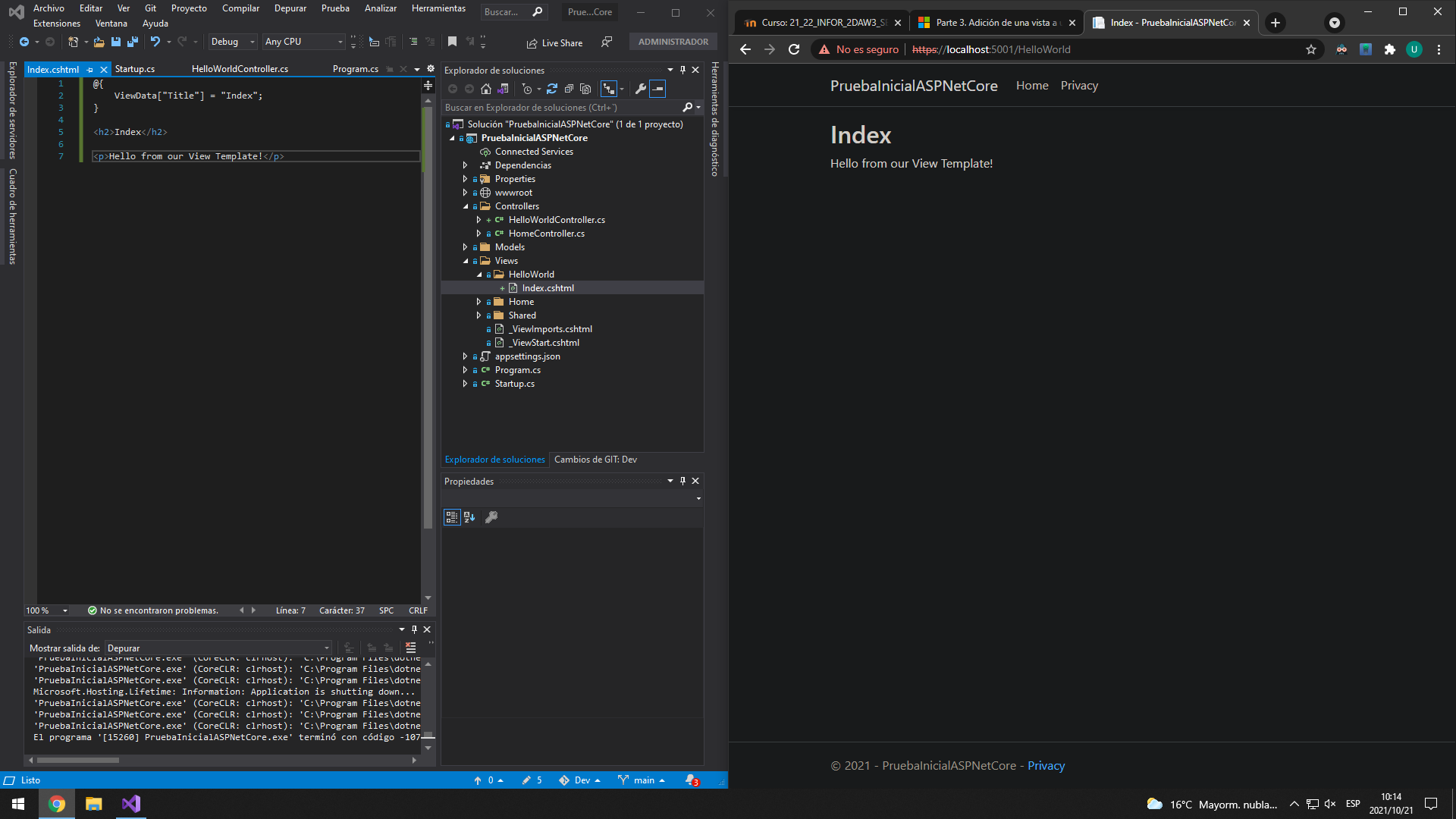
[https://localhost:5001HelloWorld/Welcome/?name=Urko](https://localhost:5001HelloWorld/Welcome/?name=Urko)

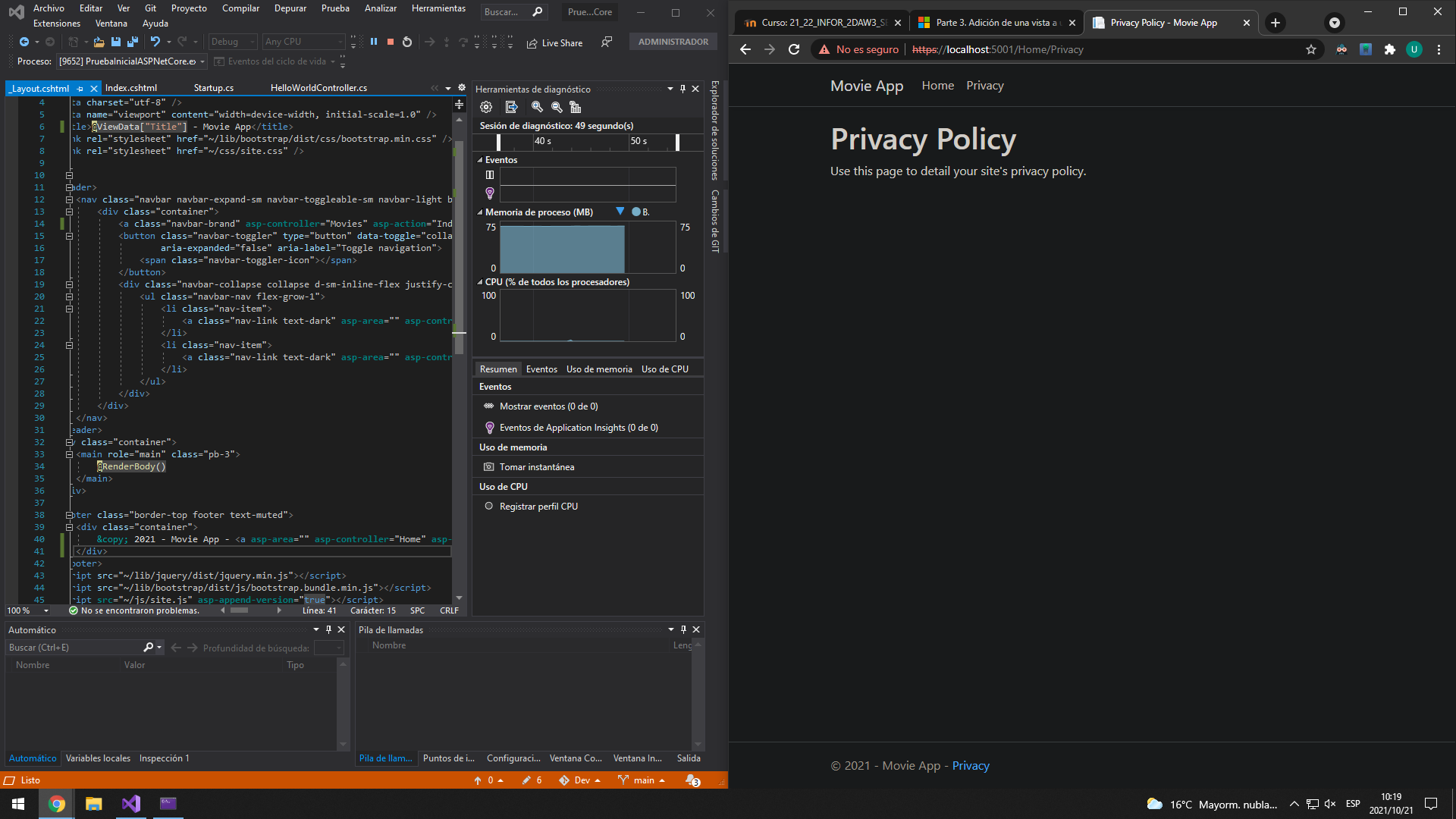
## Paso 3

Hemos aprendido a crear una vista personalizada asociada al controlador HelloWorld, y a cambiar el título de la aplicación, así como otros apartados que aparecerán en todas las páginas de la web. Ahora la página welcome está customizada y extiende del layout general de la aplicación.

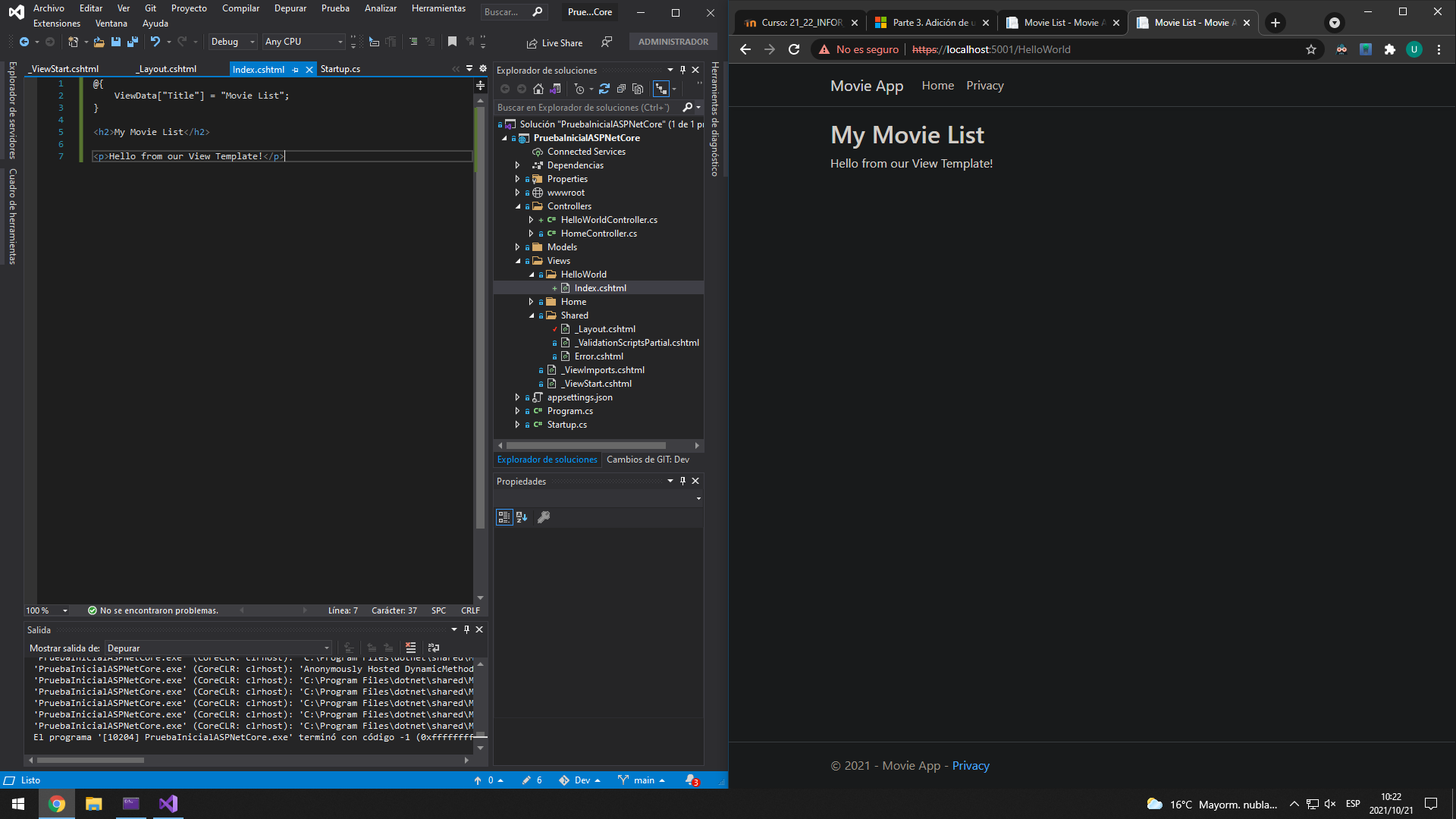


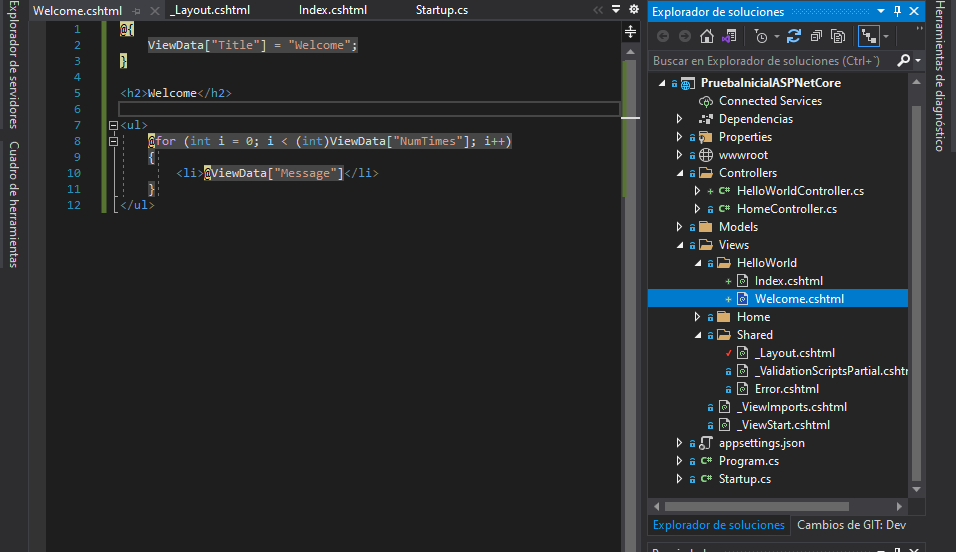
Cambiar el contenido del index.cshtml



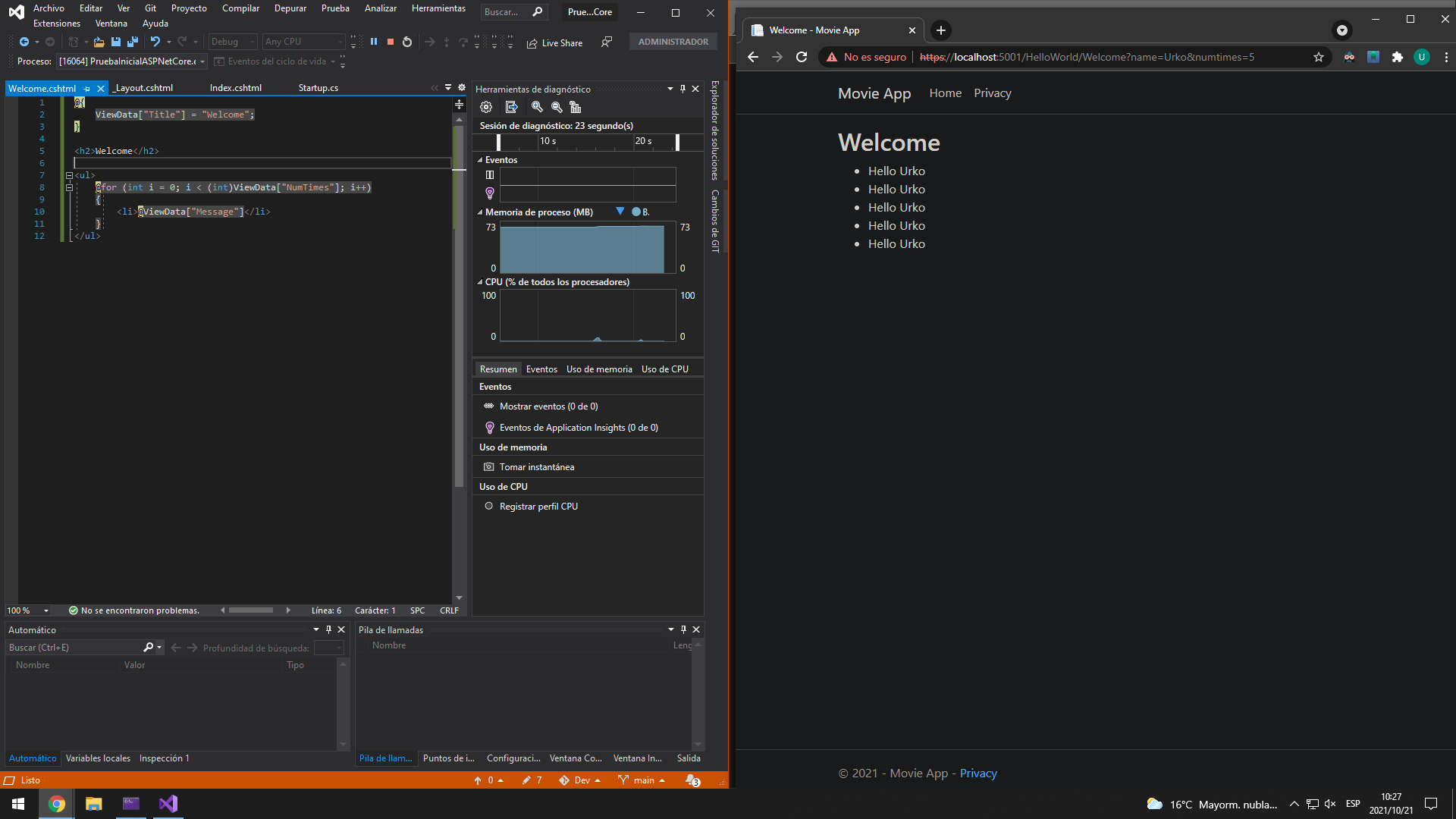
Cambiar en el \_Layout.cshtml el título de la página y algunas cosas más. Como se ve, ahora el título es “Movie App”

Cambiar el HelloWorld/Index.cshtml así, para que muestre el título “My movie List”



Crear y modificar /Views/HelloWorld/Welcome.cshtml

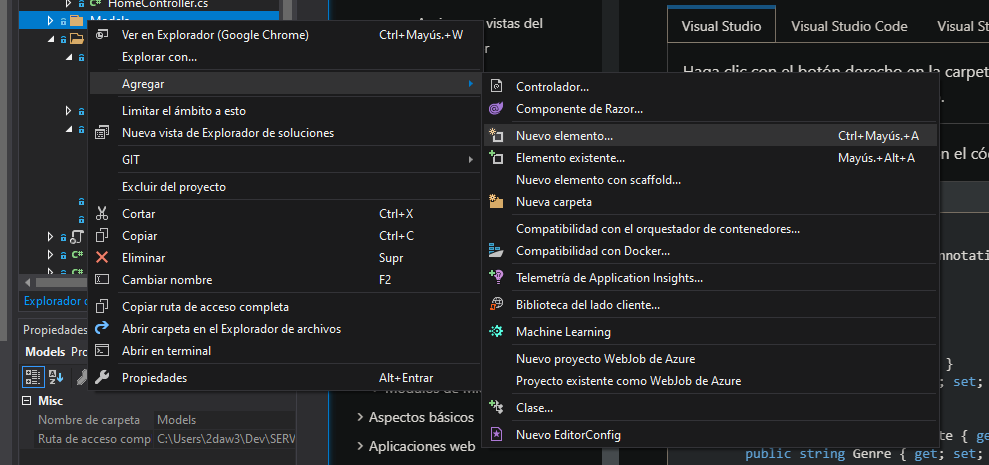
Acceder a https://localhost:5001/HelloWorld/Welcome?name=Urko&numtimes=5

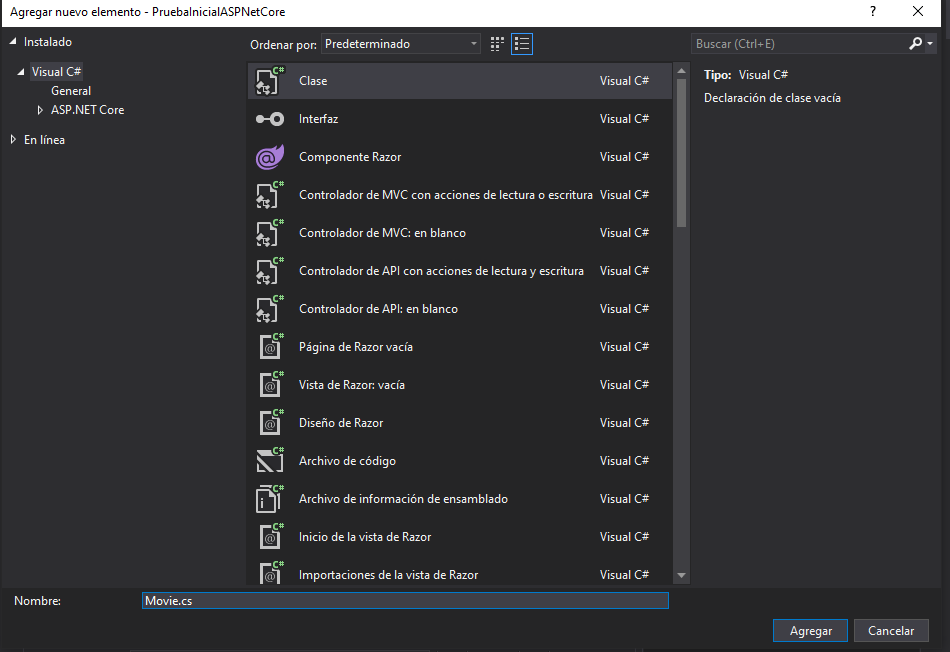


## Paso 4

Hemos aprendido a crear clases, añadir paquetes nuget y elementos con scaffold

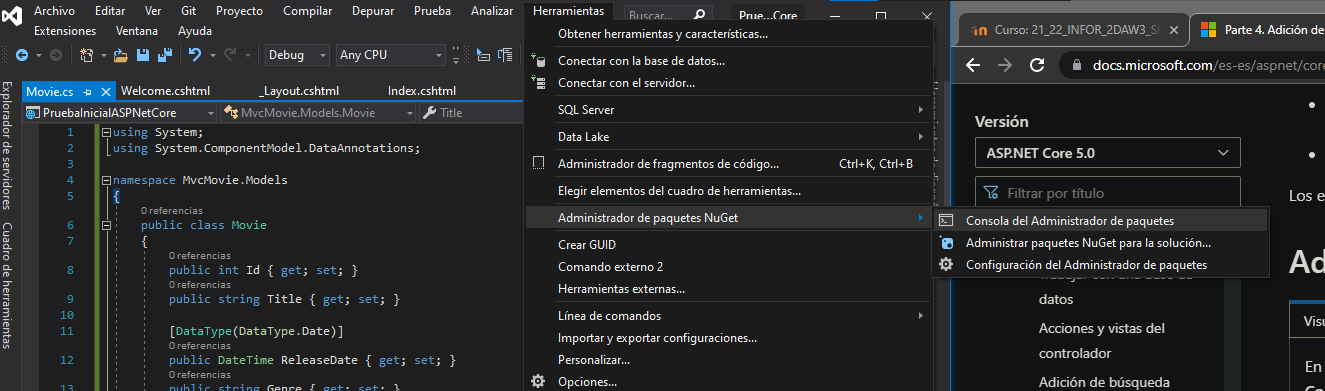
Añadir la clase Movie.cs a la carpeta modelos

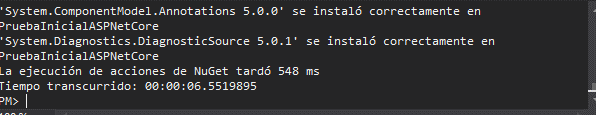


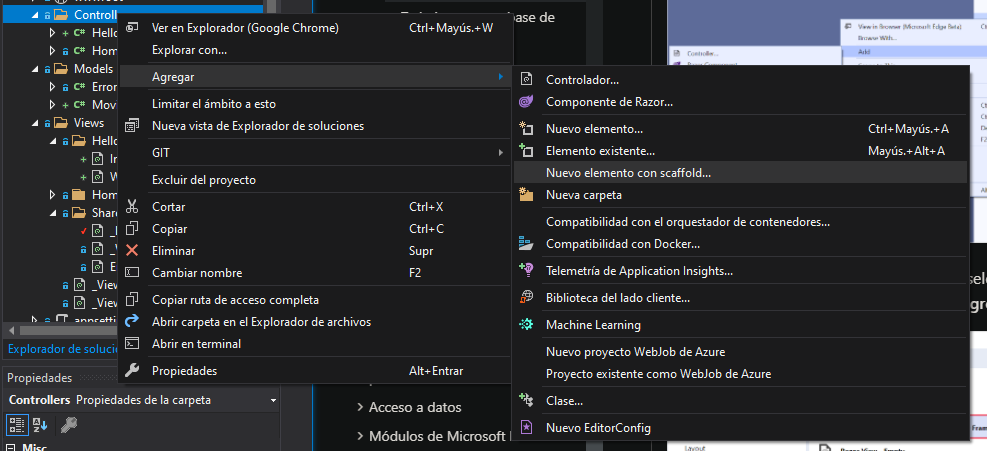


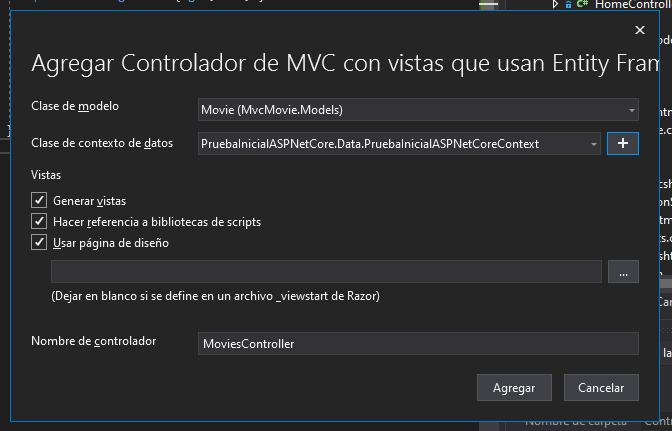
Administrar paquetes Nuget e instalar

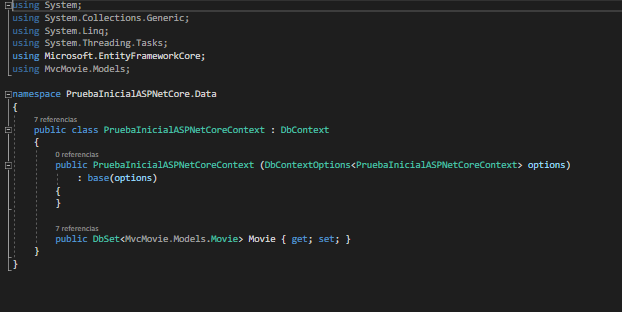
Install-Package Microsoft.EntityFrameworkCore.Design





Agregar nuevo elemento con scaffold





Add-Migration InitialCreate

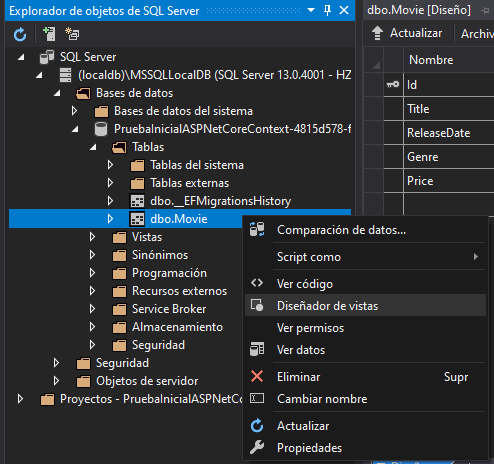


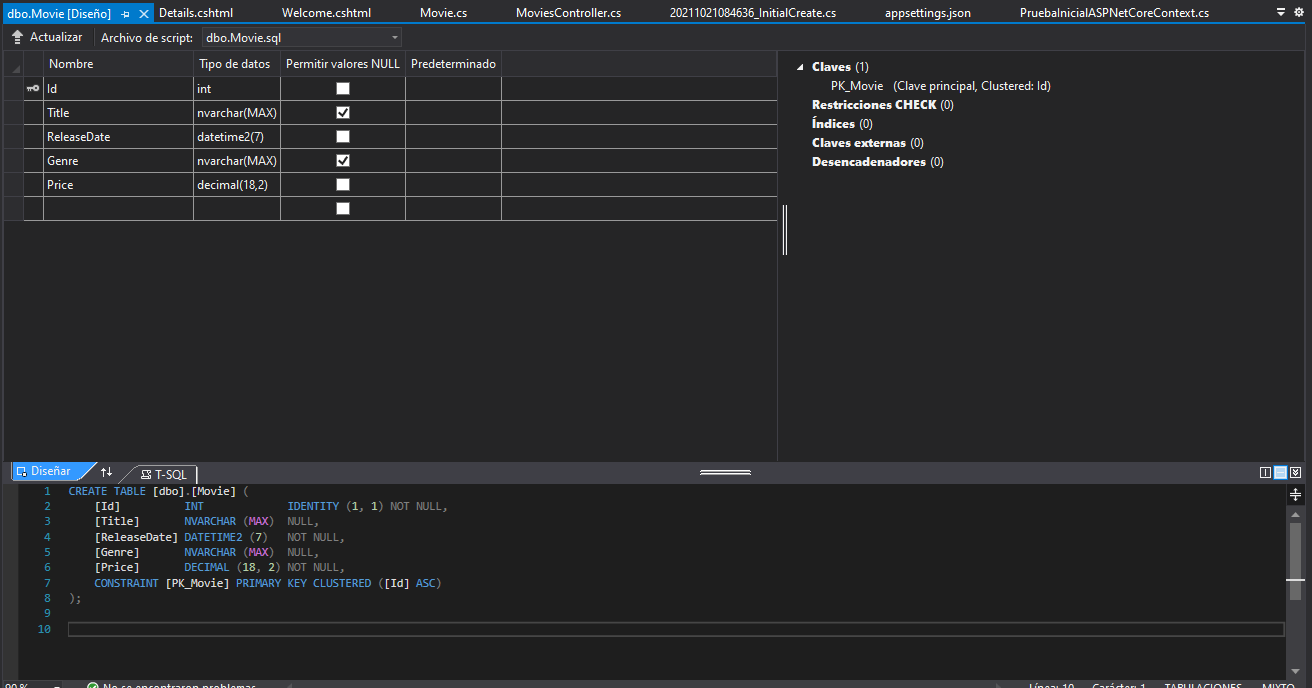
Update-Database

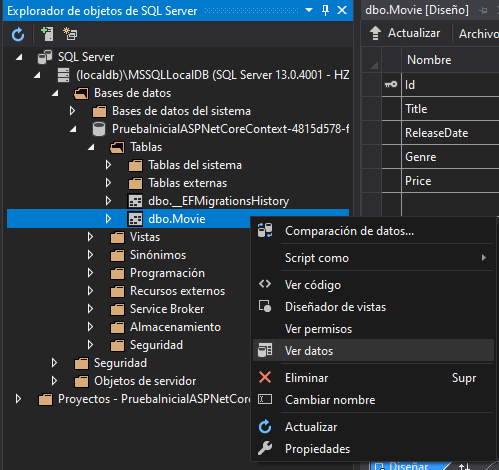


# Parte 5

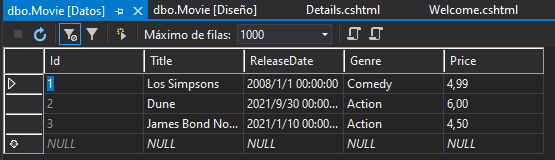
Hemos aprendido a acceder a las tablas de sql, modificarlas, crear la clase SeedData que accederá a la BD, y que si no tiene valores insertados creará nuevos. Para comprobar esto último hemos eliminado los datos que habíamos metido en un principio.



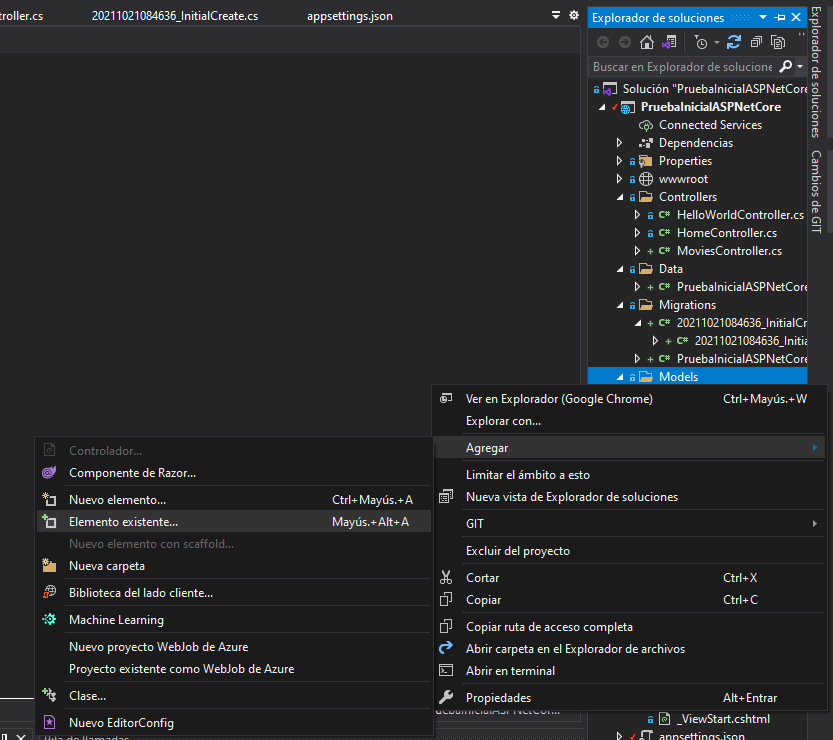




Insertar datos manualmente



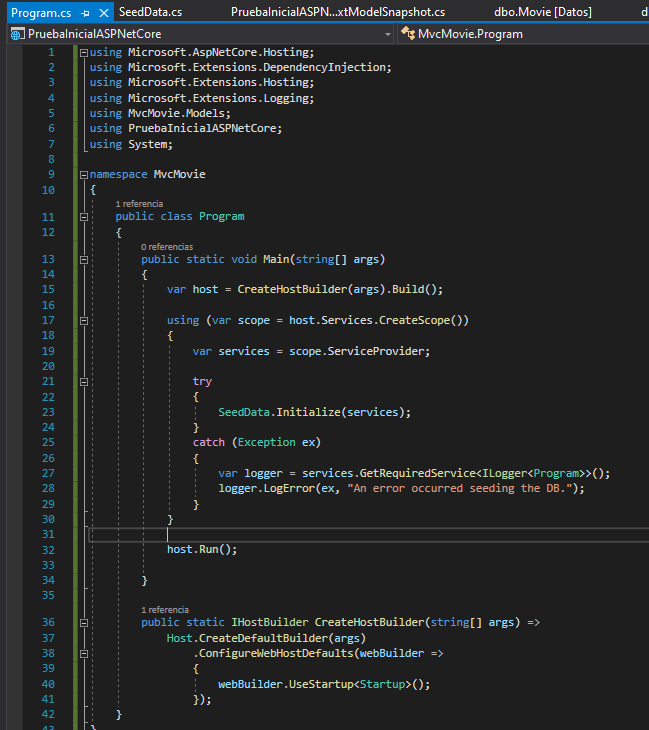
Crear clase SeedData en Models



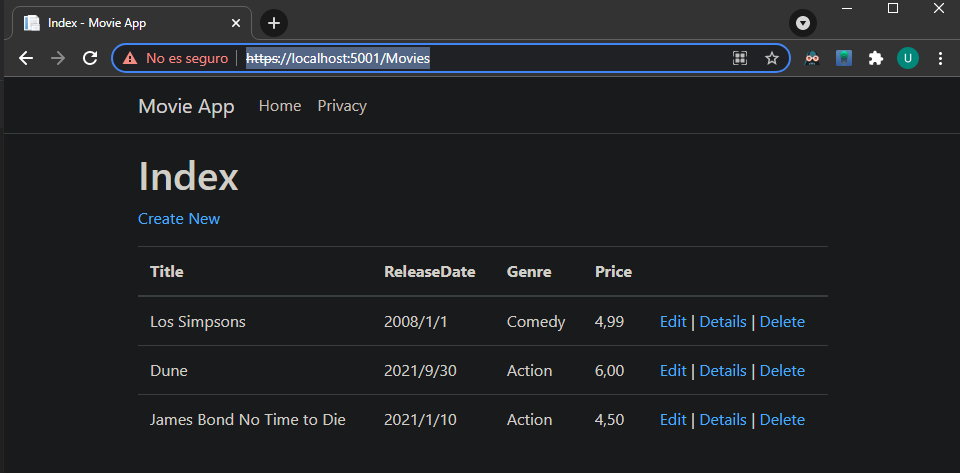
Este código en el SeedData.cs



Este código en el program.cs

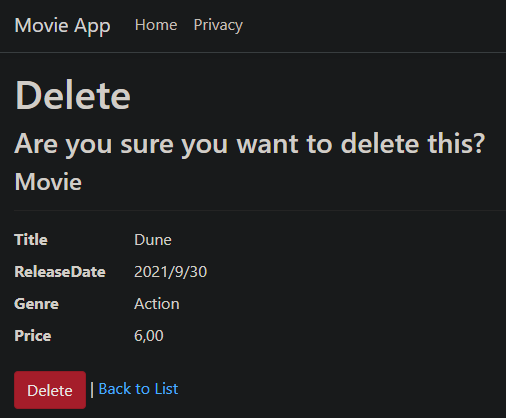


Acceder a <https://localhost:5001/Movies> y ver la lista de pelis

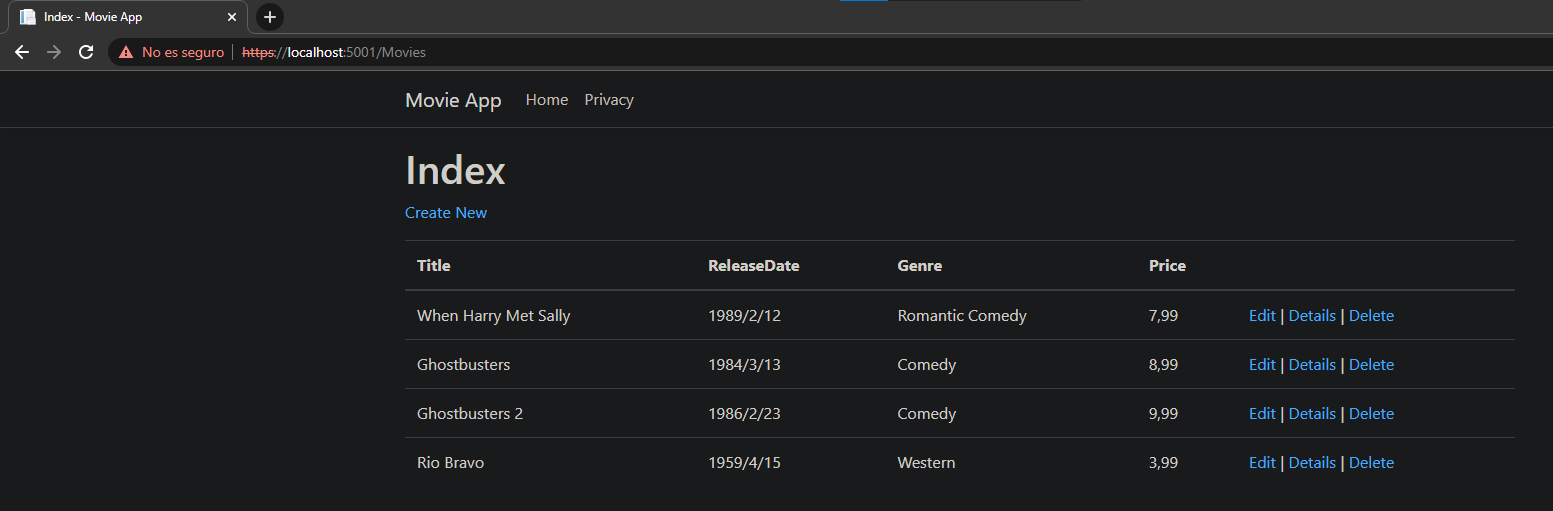


Eliminar las pelis, para que al re-ejecutar el programa cree las pelis que hay en el SeedData.cs



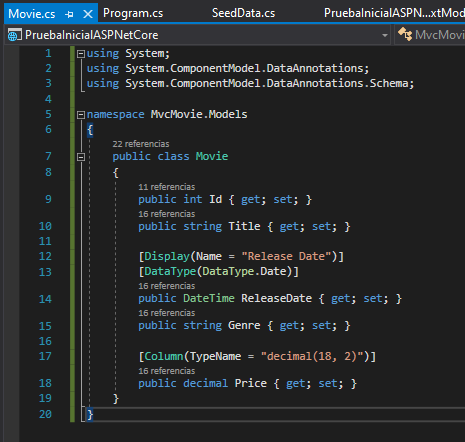


Al dejar de ejecutar y volver a ejecutar el programa, se crean las películas del SeedData.cs



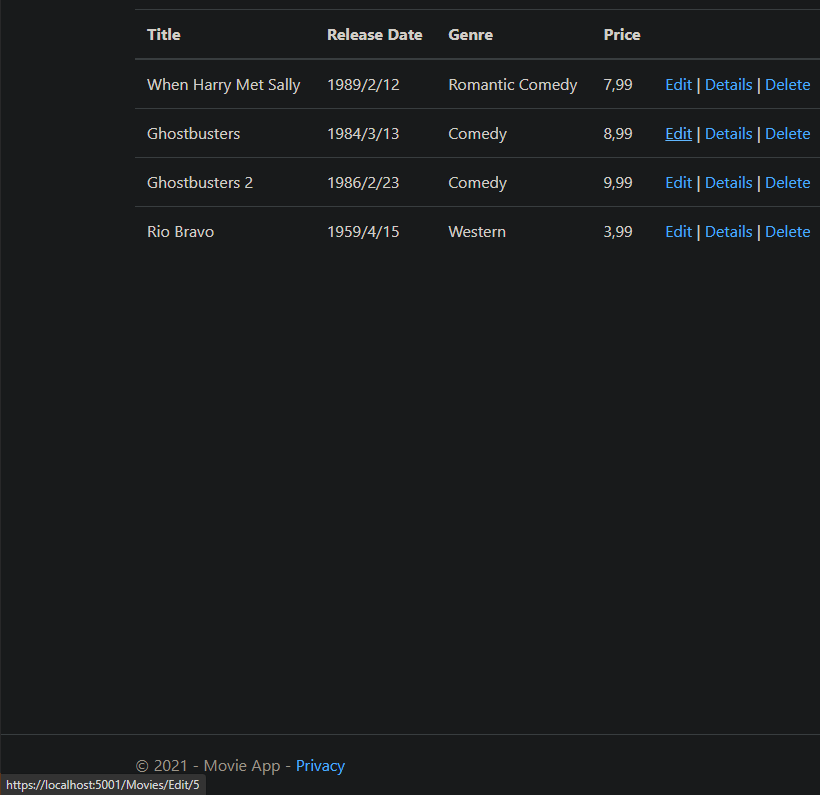
# Paso 6

Cambiamos el models/movie.cs para mostrar los datos de una forma más limpia



Ahora la fecha se muestra de forma más clara.



Si mantenemos el ratón sobre un link de editar, borrar o ver los detalles de una película, se ve abajo a la izquierda la dirección a la que accederíamos de clicar en dichos links.

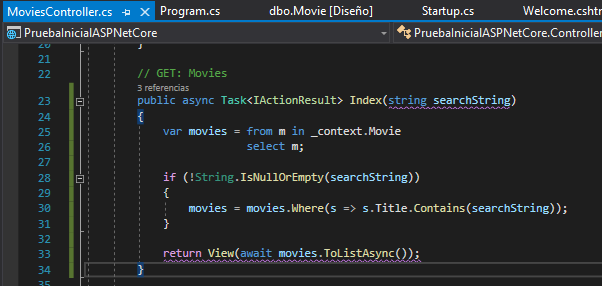
Podemos ver también la pantalla de edición, la cual validará los tipos de datos según los datos que admita sql en esos campos.



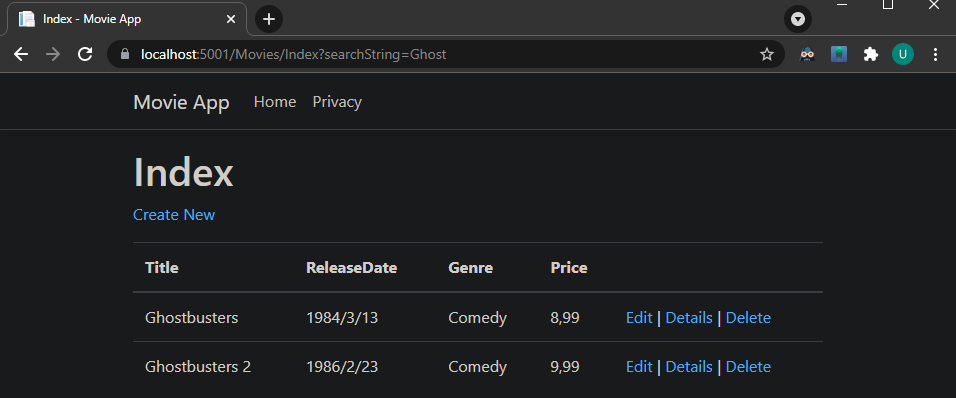
# Paso 7

Hemos aprendido a filtrar los títulos de las películas con una query en la url, y luego añadido un buscador para que sea más simple de hacerlo. Luego añadido un dropdown para seleccionar el género de la película

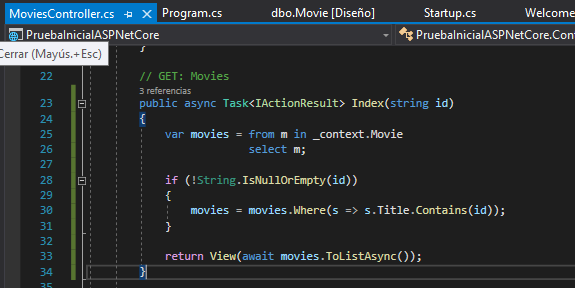
Cambiar el método Index de Controllers/MoviesController.cs por el siguiente código



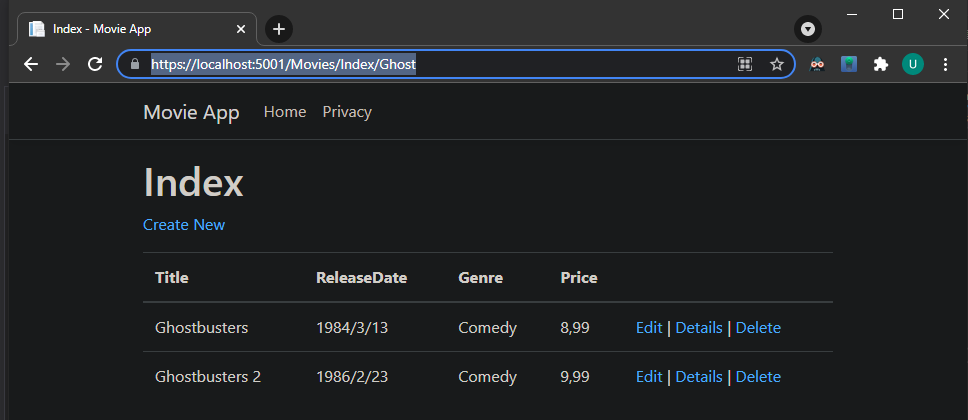
Ahora, si tras reiniciar el proyecto vamos a: <https://localhost:5001/Movies/Index?searchString=Ghost> veremos que se puede filtrar el resultado de la página



Sin embargo, si cambiamos el código del Index de Controllers/MovieController por el siguiente…

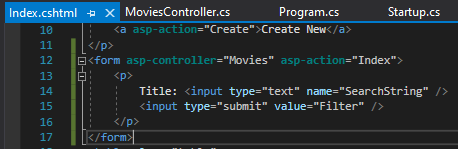


Podremos hacer la misma búsqueda solamente escribiendo después de “index” en la url, de la siguiente forma <https://localhost:5001/Movies/Index/Ghost>

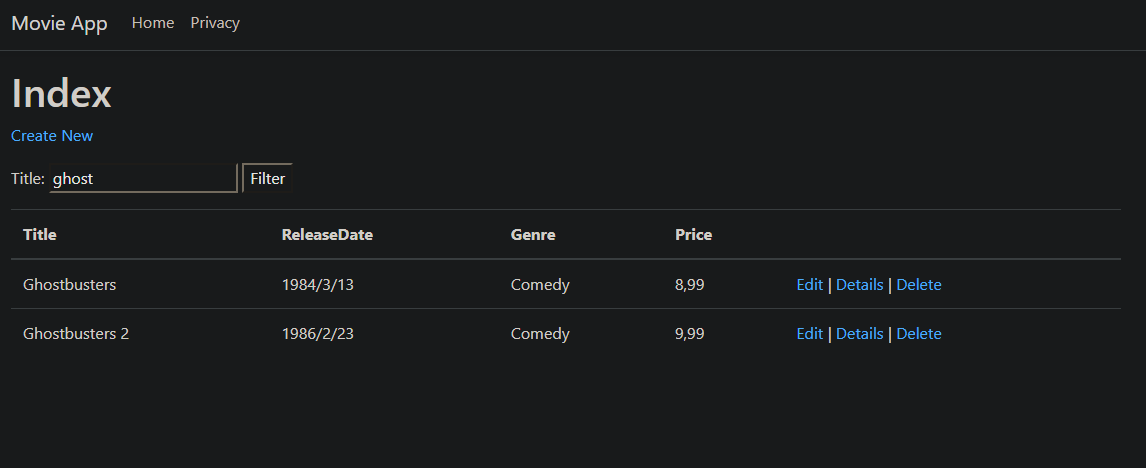


Sin embargo, como los usuarios no usarán este método para filtrar, volveremos a escribirlo de la forma anterior en el Index, y añadiremos un buscador

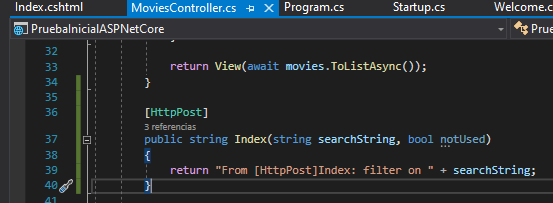
Para añadir el buscador, se hará con el siguiente código en el views/movies/index.cshtml



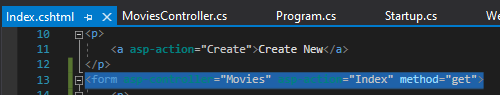
El resultado será el siguiente: buscará en el título la palabra escrita en la barra de búsqueda



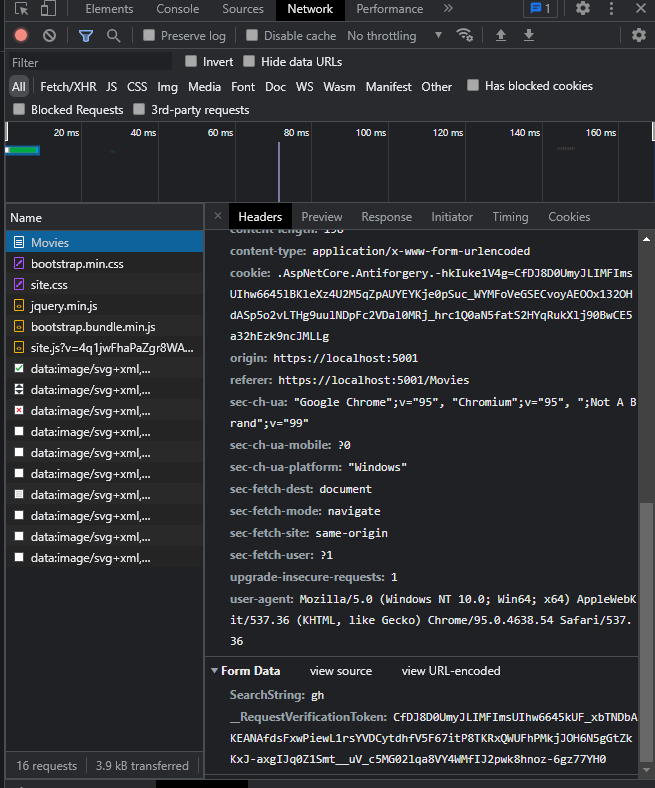
Modificamos el código al MoviesController.cs



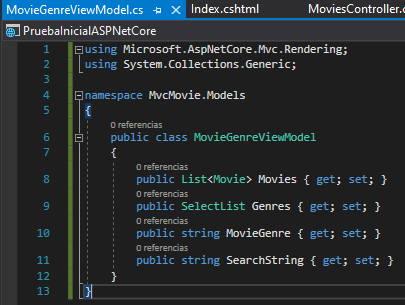
Y cambiar la siguiente línea en el index.cshtml



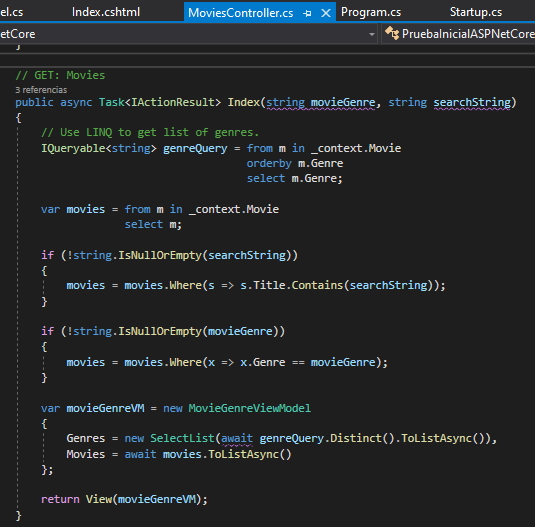
Podemos monitorizar la consulta en del formulario en las herramientas de desarrollador del navegador



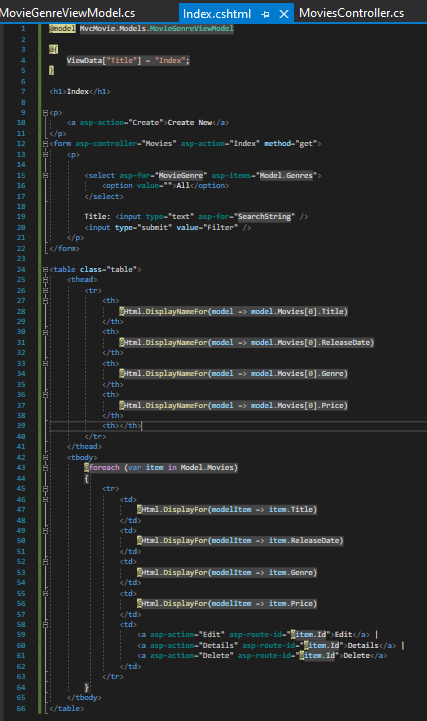
Crear la clase /Models/MovieGenreViewModel.cs



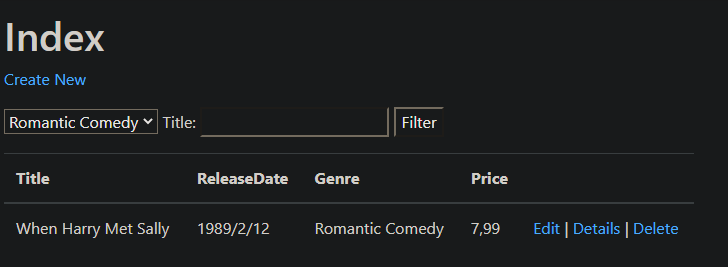
Reemplazar el método index de Controllers/MoviesController.cs



Cambiar el código del Views/Movies/index.cshtml



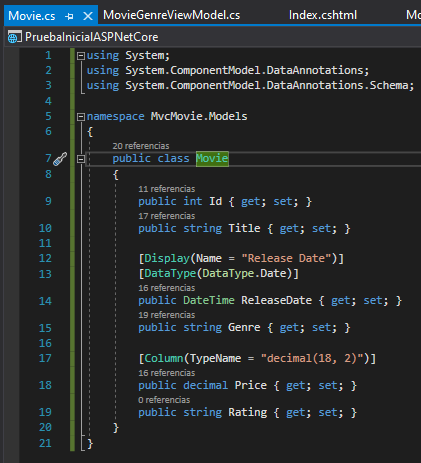
Dropdown con el filtro por género:



# Paso 8

Hemos aprendido a añadir un nuevo campo al objeto movie, y mostrarlo en cada una de sus vistas, así como poder editarlo también

Añadir un nuevo campo al Models/Movie.cs

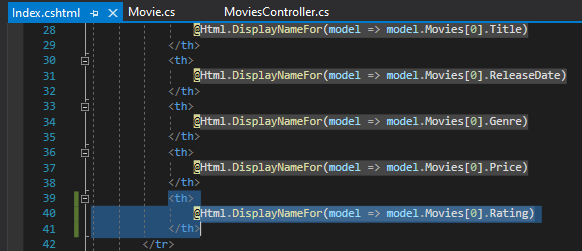


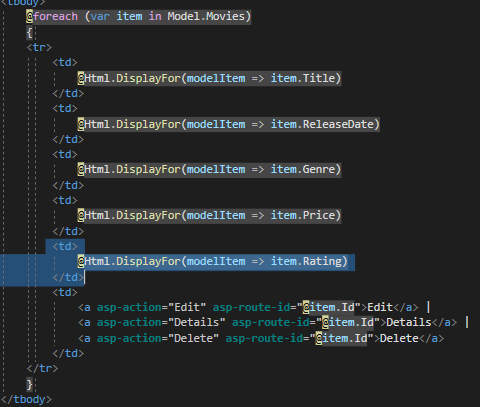
Actualizar con el nuevo campo los Bind de edit y create en el MoviesController.cs

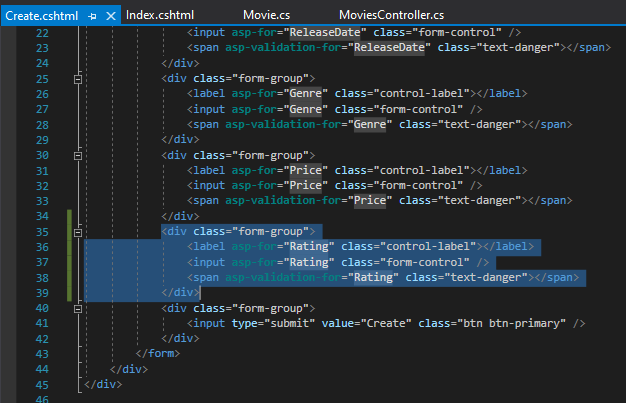




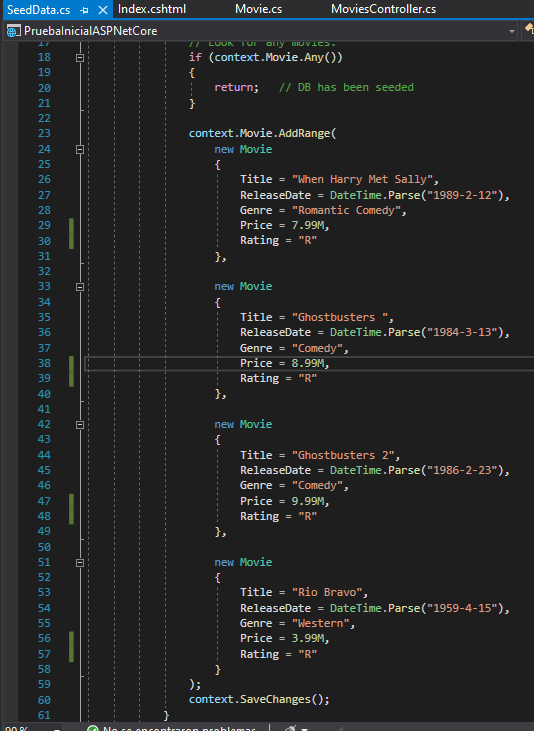
Añadir los th y td correspondientes en el Views/Movies/index.cshtml



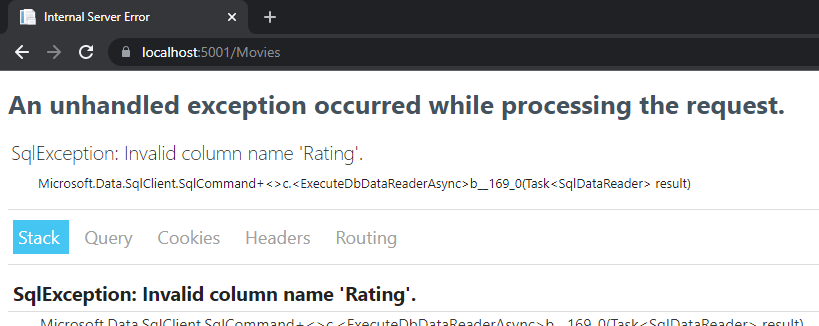


Idem en el Views/Movies/Create.cshtml

En la clase SeedData añadir el campo rating a cada película



Si se ejecuta la aplicación ahora, da el error de sql “Invalid column name ‘rating’”

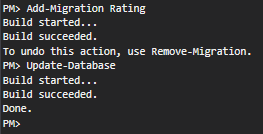


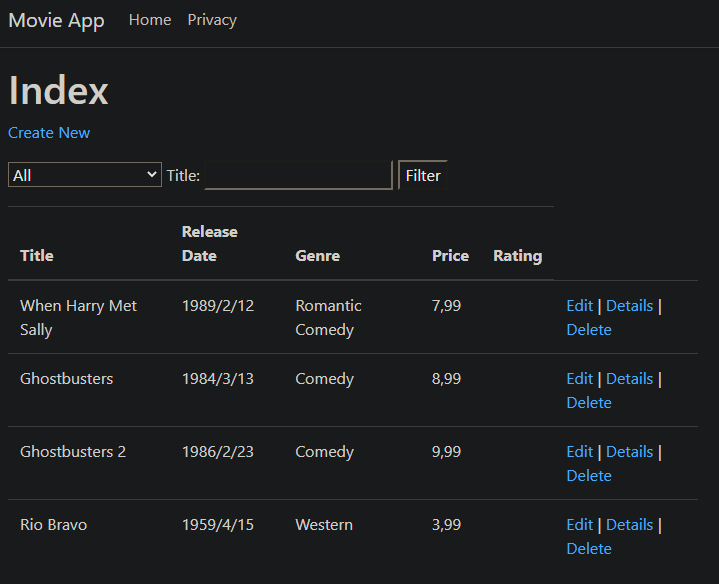
Se podría solucionar de varias formas, como recreando la base de datos, pero vamos a usar Migraciones de Code First

Para instalarlo, ejecutamos

Add-Migration Rating

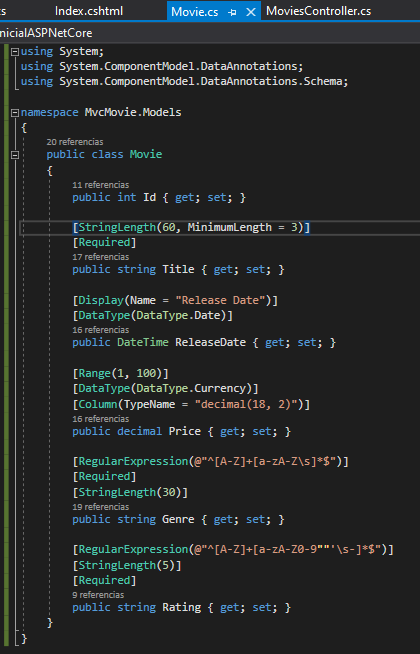
Update-Database



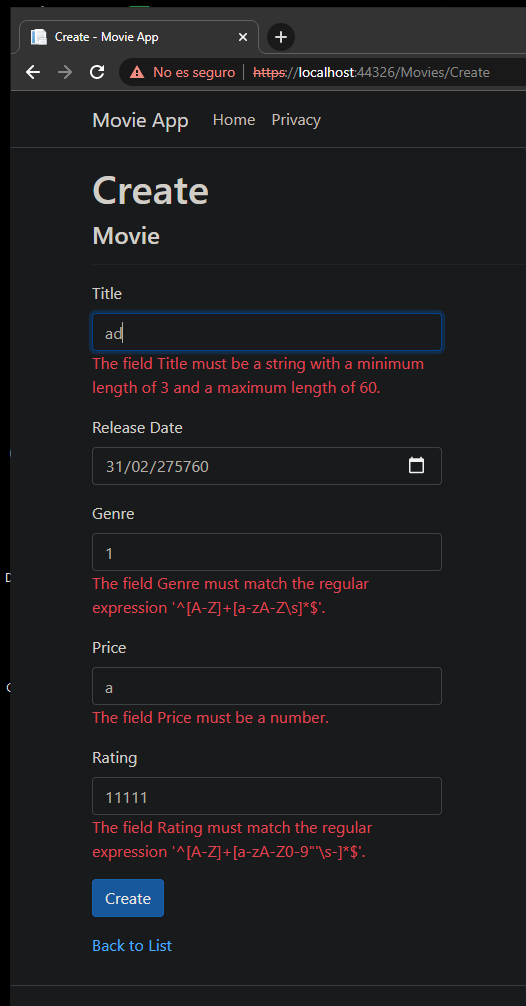
Como vemos, se ha creado el campo correctamente

# Paso 9

Cambiar el Models/Movie.cs por el siguiente código. Integrando así validaciones a los campos



Con esas validaciones, al crear una nueva película la web, dará error al no cumplirlas



(La fecha debería dar error también…)