**DESARROLLO DE UN PROYECTO WEB (CAPA DE PRESENTACION)**

**CON CONECCIÓN A UNA WEB API**

**(CAPA DE DATOS)**

**Y BASE DE DATOS EN SQL SERVER PARA ITALIKA**

Contenido

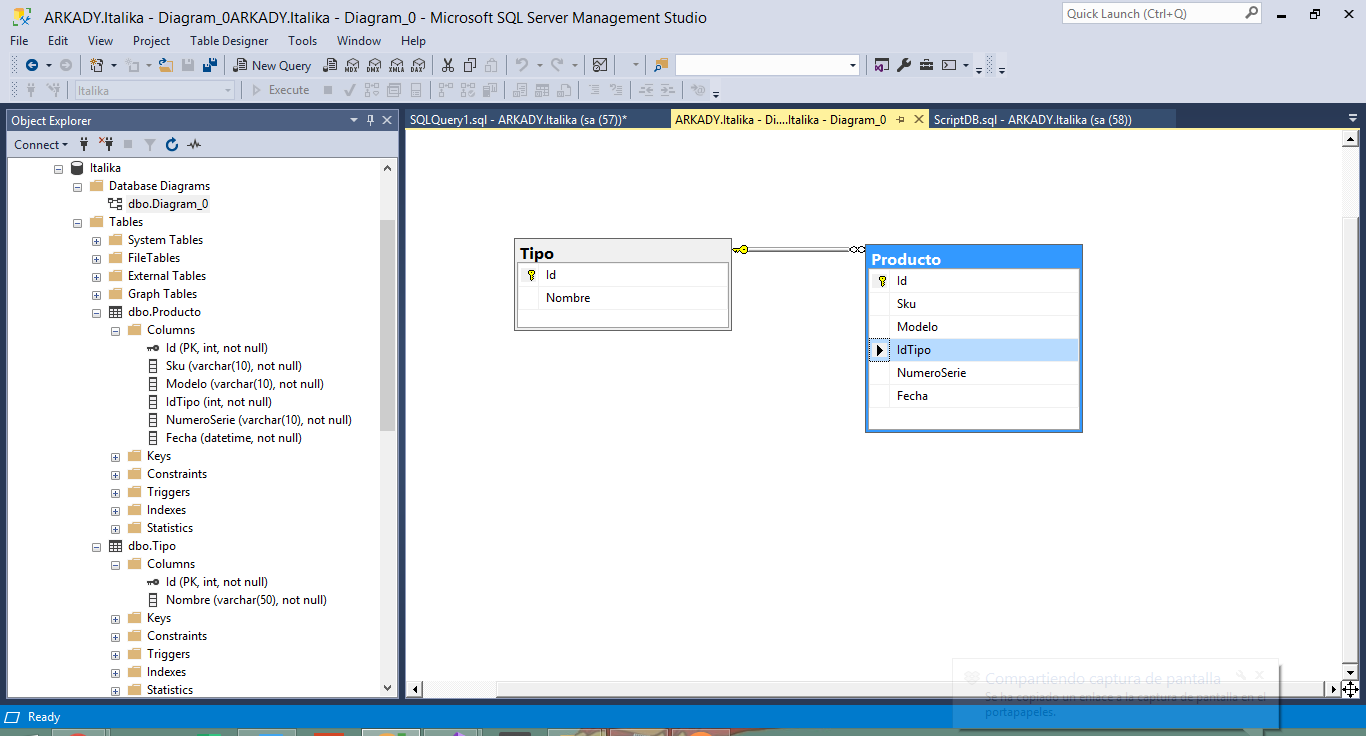
[Desarrollo de la Base de Datos 2](#_Toc73474439)

[Desarrollo de la Web Api 3](#_Toc73474440)

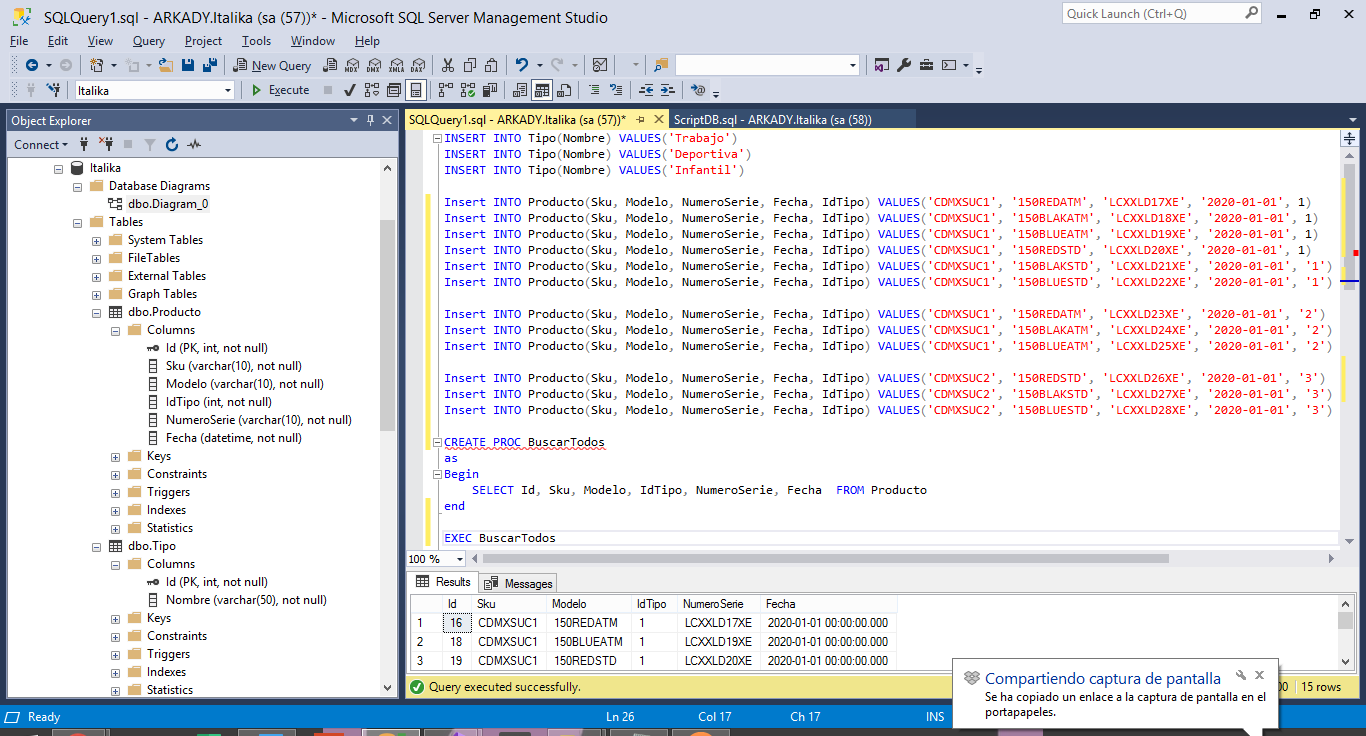
[Desarrollo del portal Web MVC 5](#_Toc73474441)

# Desarrollo de la Base de Datos

Se creo una Base de Datos en Sql Server 2019 una tabla Producto que representa a las Motocicletas de la cadena Italika con las especificaciones básicas como Numero de Serie y SKU además del ID y con su respectiva relación hacia un catalogo de Tipo donde específicamente se generaron 3 tipos de Motocicletas: (TRABAJO,DEPORTIVA,INFANTIL)

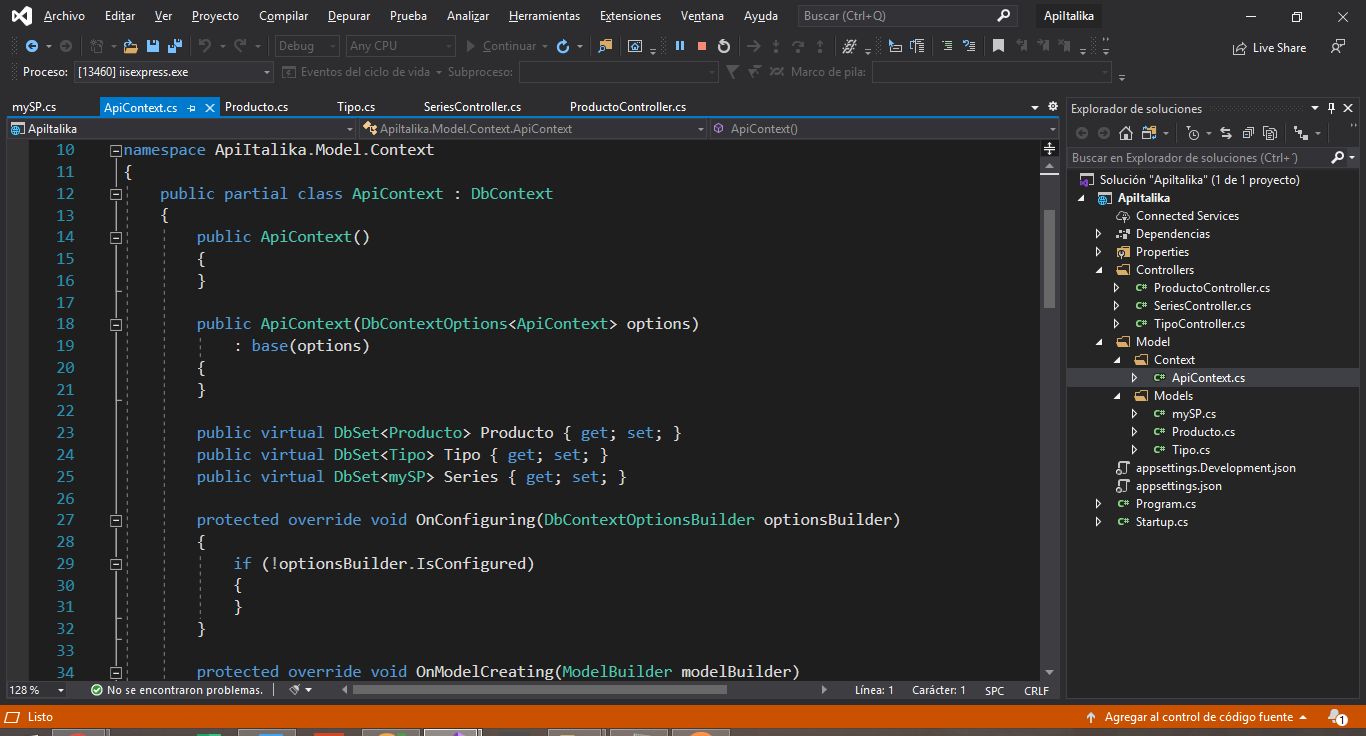


Se Setearon algunos datos y se generaron algunos Stored Procedures para que más adelante las consultas fuesen más eficientes.



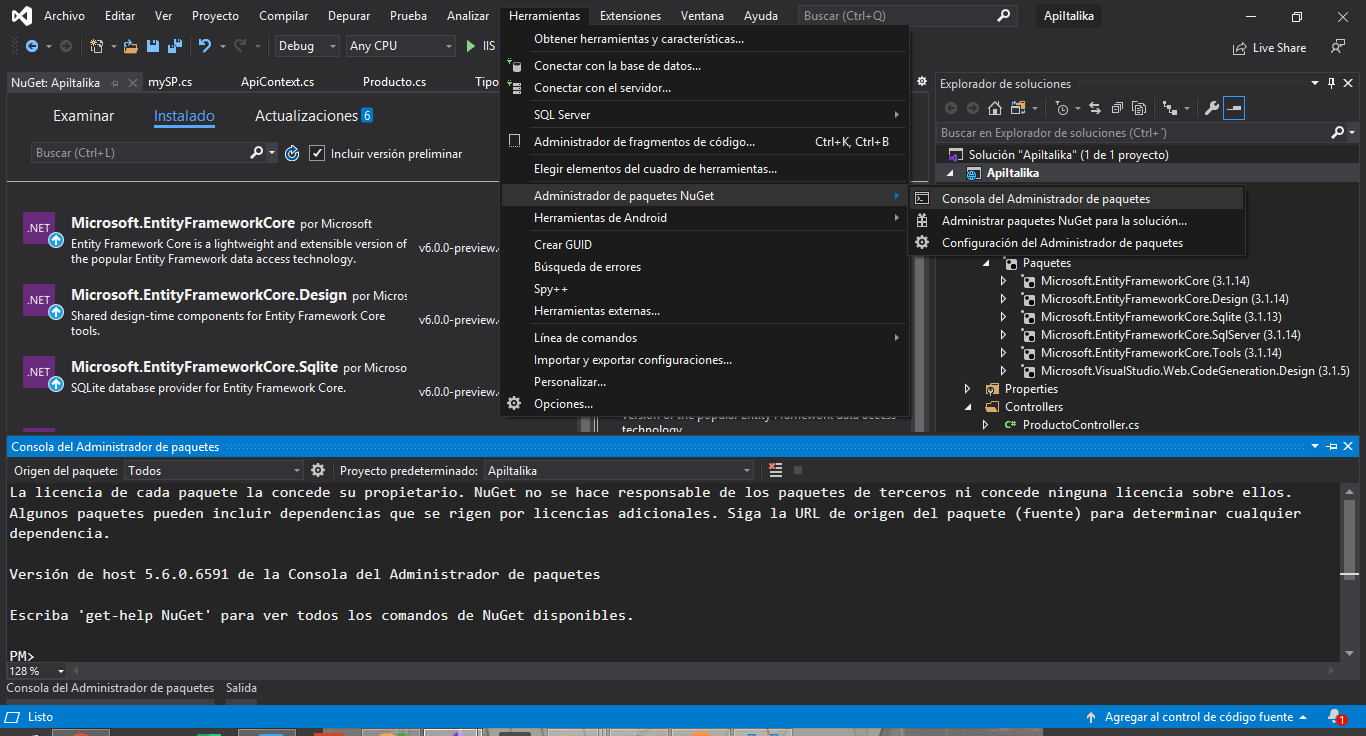
# Desarrollo de la Web Api

Para la manipulación de los datos de forma asíncrona, se creo un nuevo proyecto Web Api en el Framework de .Net Core para administrar los datos de la base de datos previamente creada. Con ayuda de la herramienta ORM Entity Framework el proceso de construcción de un modelado de datos Cotextual, así como las tablas o ‘DBSets’, fue rápido y sin errores.

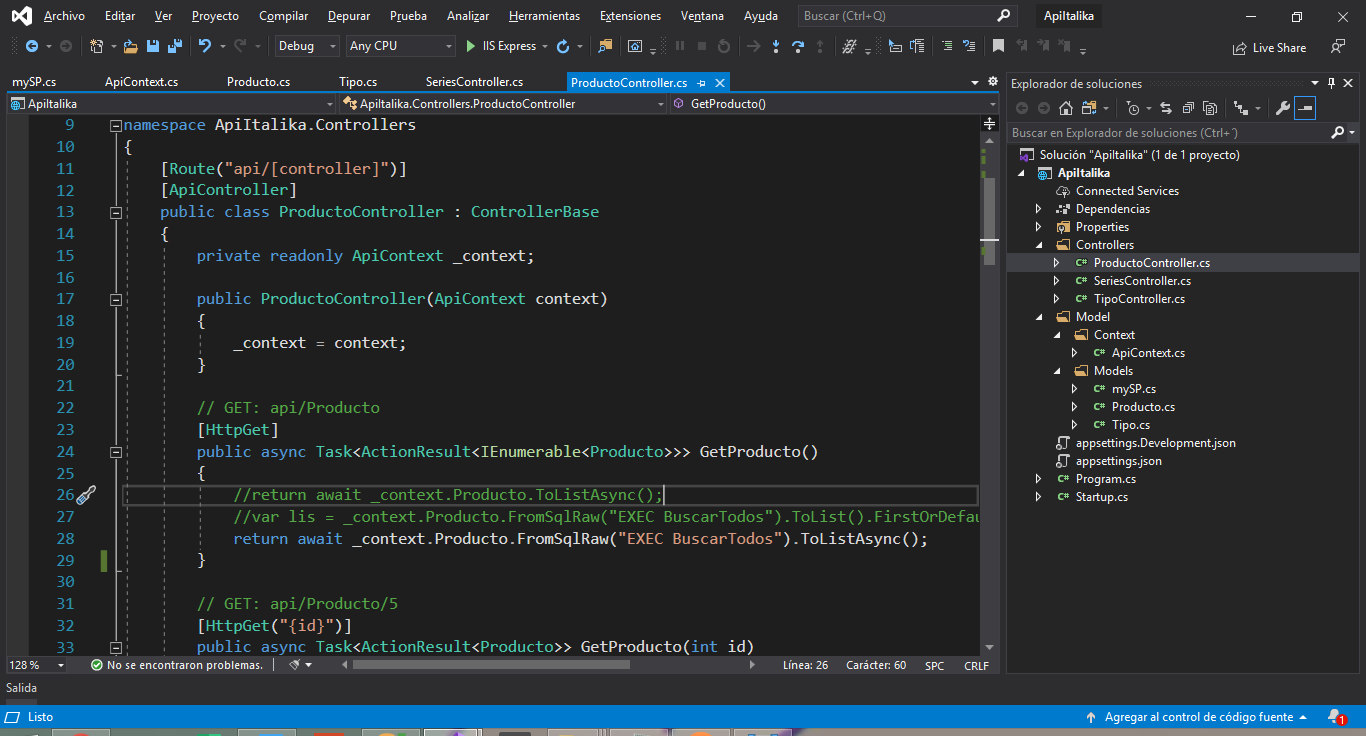


Cabe mencionar la cadena usada para ejecutar el ORM Entity fue la siguiente

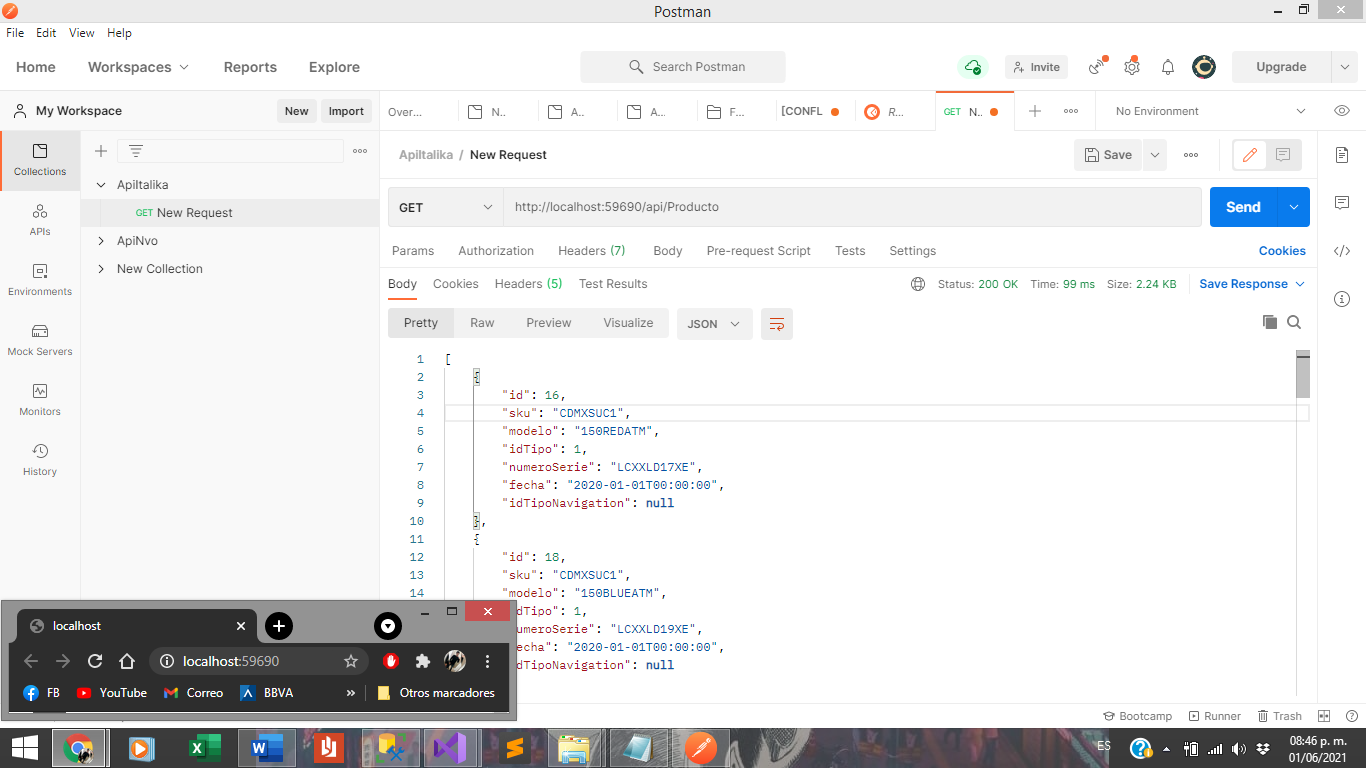
**scaffold-DbContext "Server=ARKADY;Database=Italika;User Id=sa;Password=\*\*\*\*\*\*\*\*;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models/Models -ContextDir Models/Context -Context SecContext -Force**Una vez instalados los paquetes Nuget del Entity Framework Adecuados y aperturada la consola de comandos del Administrador de paquetes de Nuget que nos proporciona Visual Studio



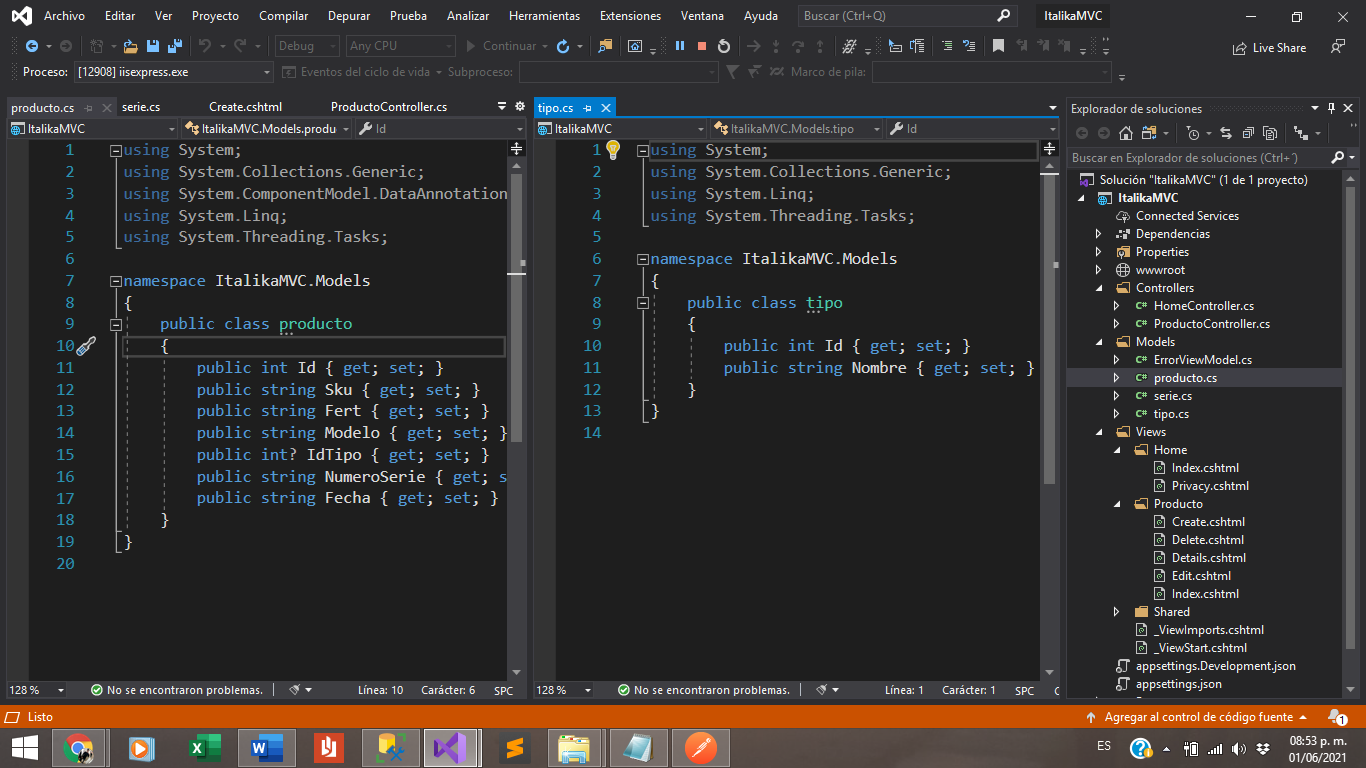
Se generan los Controladores y se hacen los cambios pertinentes para que cada petición funcione de forma correcta.



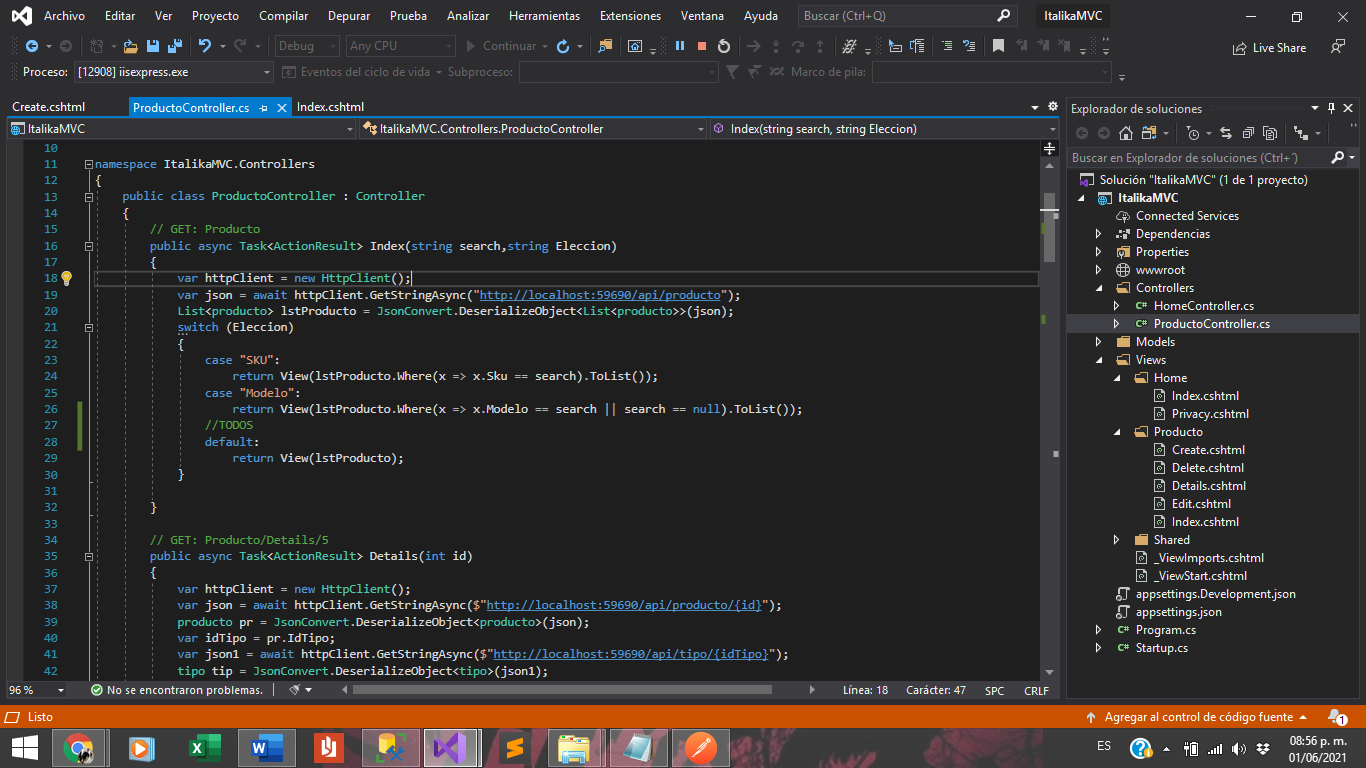
Ya con la compilación correcta y la ejecución de la web api usamos **Postman** para probar la funcionalidad correcta de cada petición.



# Desarrollo del portal Web MVC

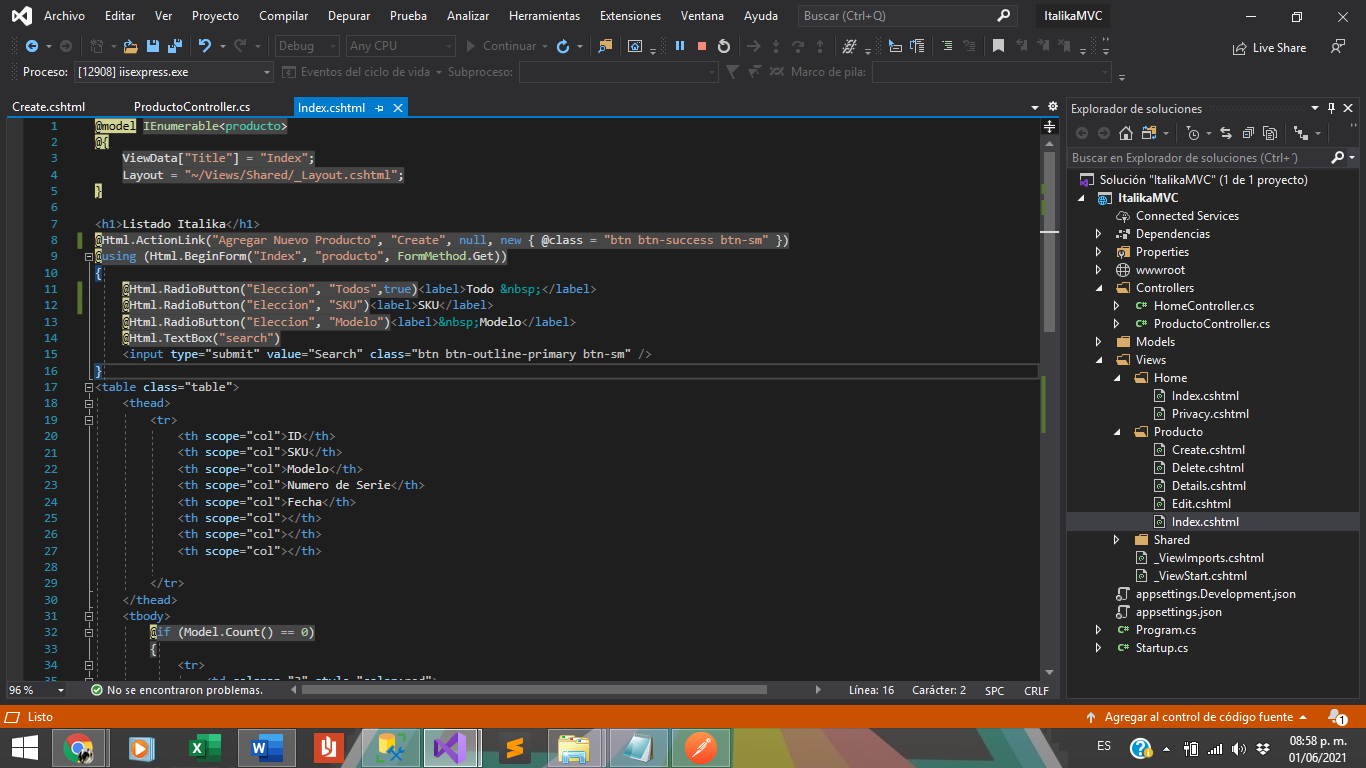
En la primera parte del proyecto se desarrollan los modelos 

Se generan los Controladores en base a los modelos de la capa de presentación

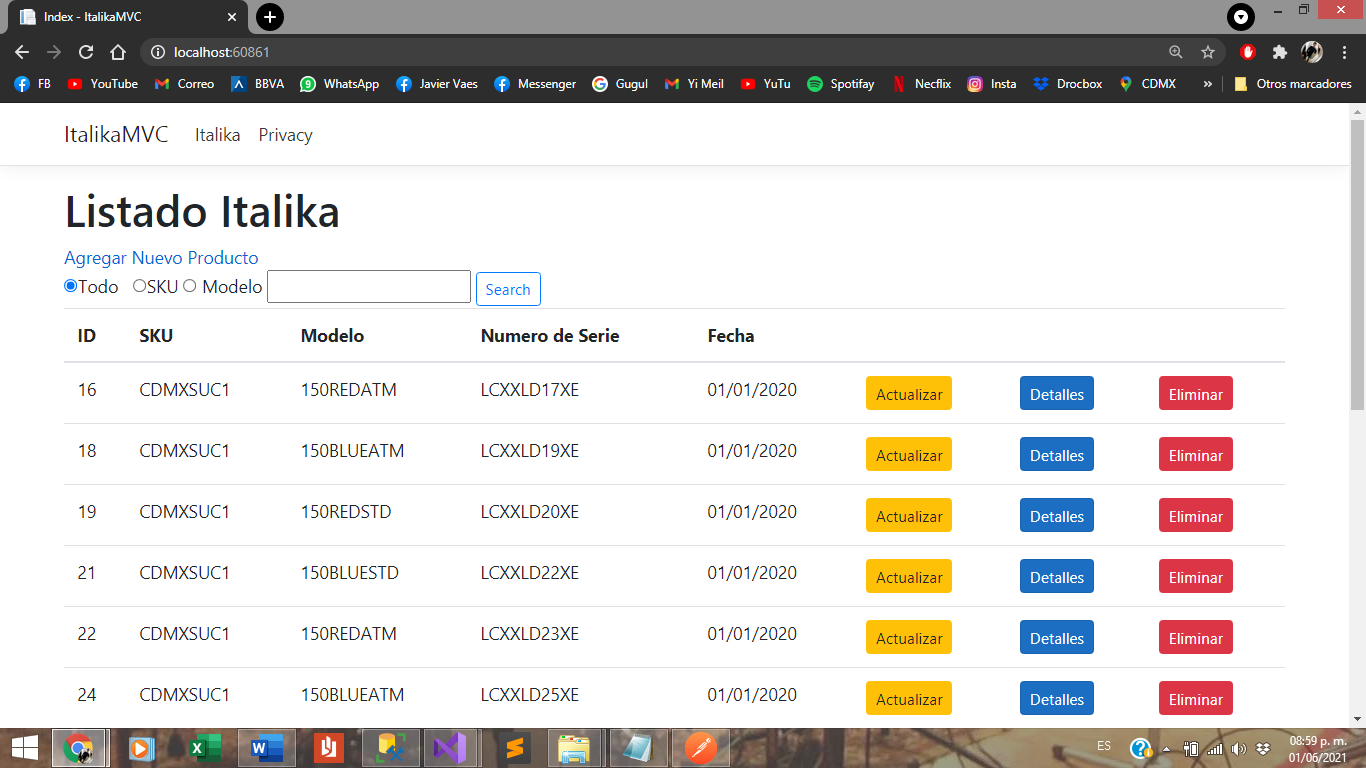


Se generar los métodos para la manipulación de los datos y las vistas uno por uno, en especial detalle con la de editar y la de agregar, para usar el Store Procedure en base a los modelos y presentar los diferentes tipos de motocicletas en el catalogo, así como una edición correcta.

Se generan las vistas nuevamente con ayuda del Scaffoldding de Visual Studio



Así se ve finalmente el index, con los botones pertinentes para el funcionamiento del CRUD



Pongo énfasis nuevamente en el código de los Radio Buttons generados para poder cumplir con las características de visualizar solo Motocicletas por modelo, o por SKU

**@using (Html.BeginForm("Index", "producto", FormMethod.Get))**

**{**

**@Html.RadioButton("Eleccion", "Todos",true)<label>Todo &nbsp;</label>**

**@Html.RadioButton("Eleccion", "SKU")<label>SKU</label>**

**@Html.RadioButton("Eleccion", "Modelo")<label>&nbsp;Modelo</label>**

**@Html.TextBox("search")**

**<input type="submit" value="Search" class="btn btn-outline-primary btn-sm" />**

**}**

ESO ENVIA AL CONTROLADOR LAS OPCIONES DEL SWITCH para que sepa que imprimir y como presentar la vista del Index.

**switch (Eleccion)**

**{**

**case "SKU":**

**return View(lstProducto.Where(x => x.Sku == search).ToList());**

**case "Modelo":**

**return View(lstProducto.Where(x => x.Modelo == search || search == null).ToList());**

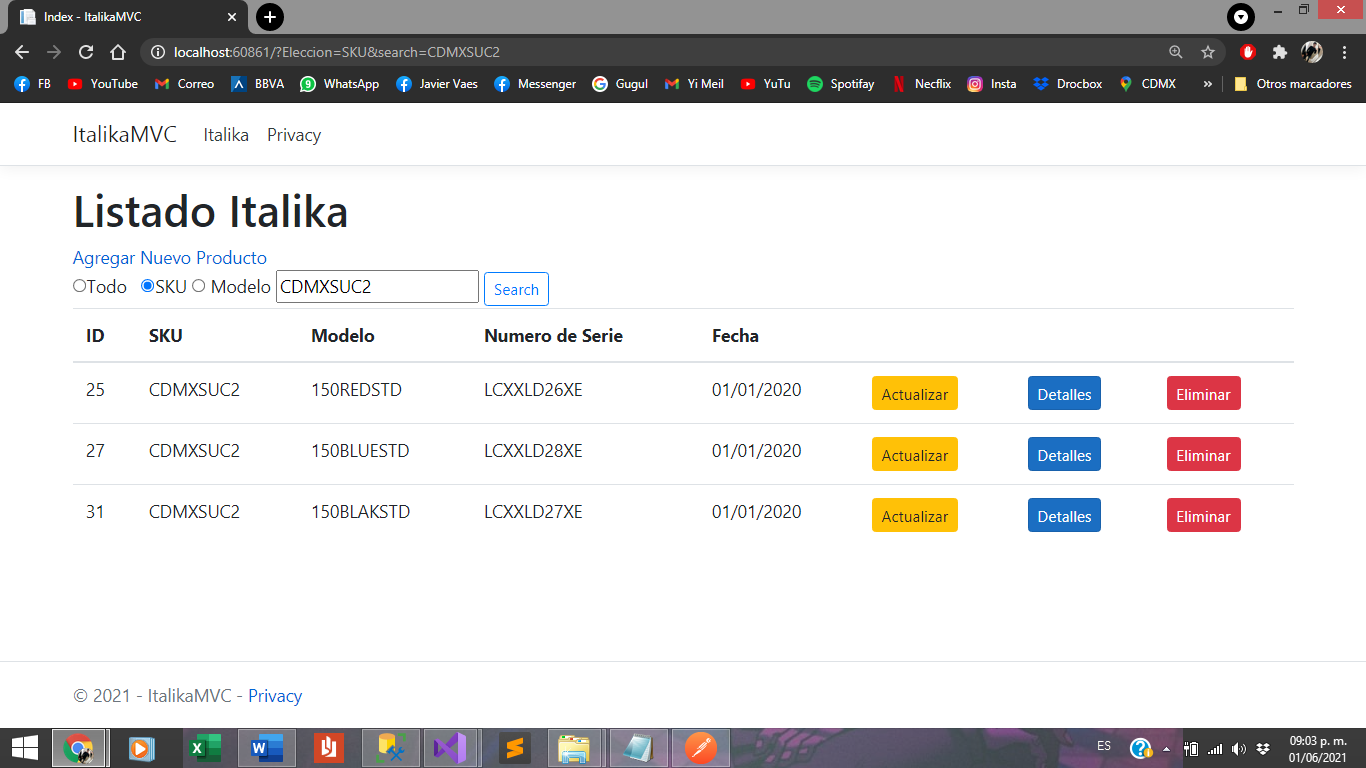
**//TODOS**

**default:**

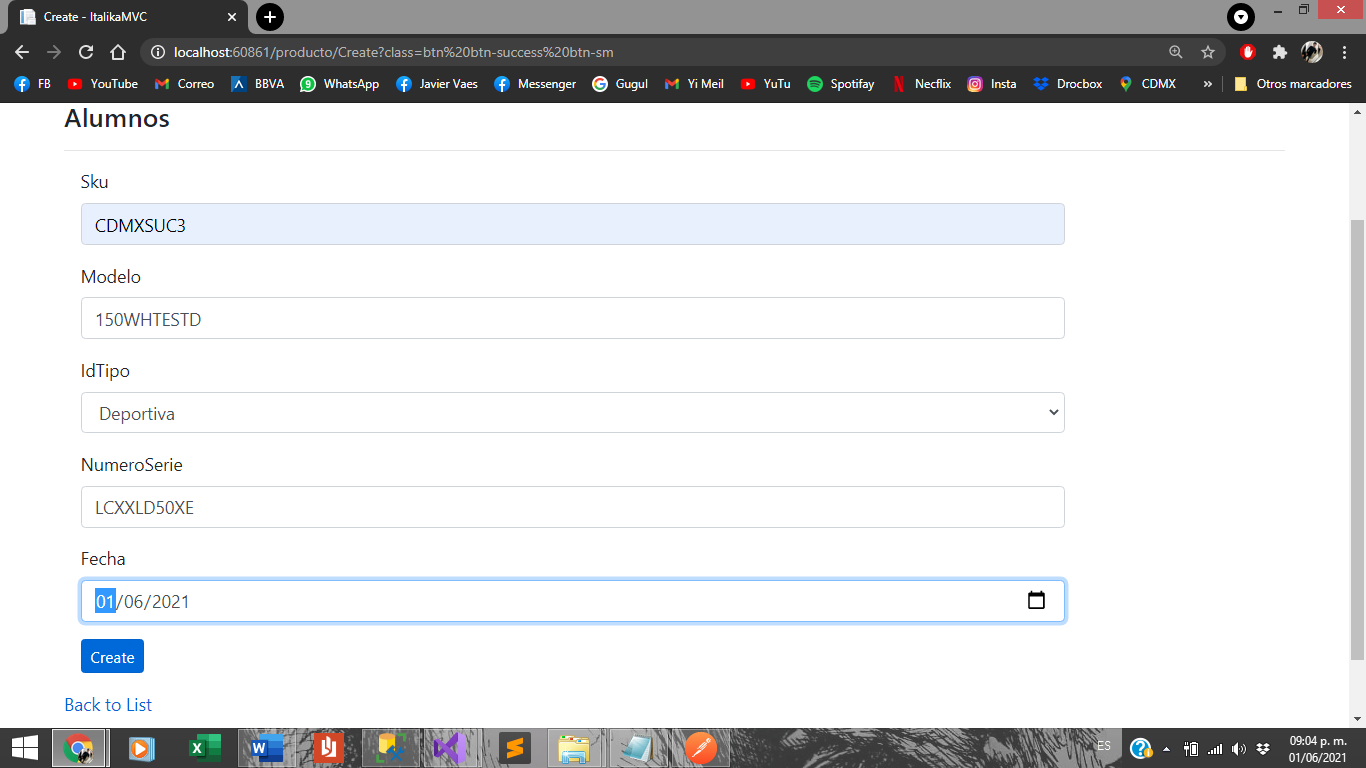
**return View(lstProducto);**

**}**

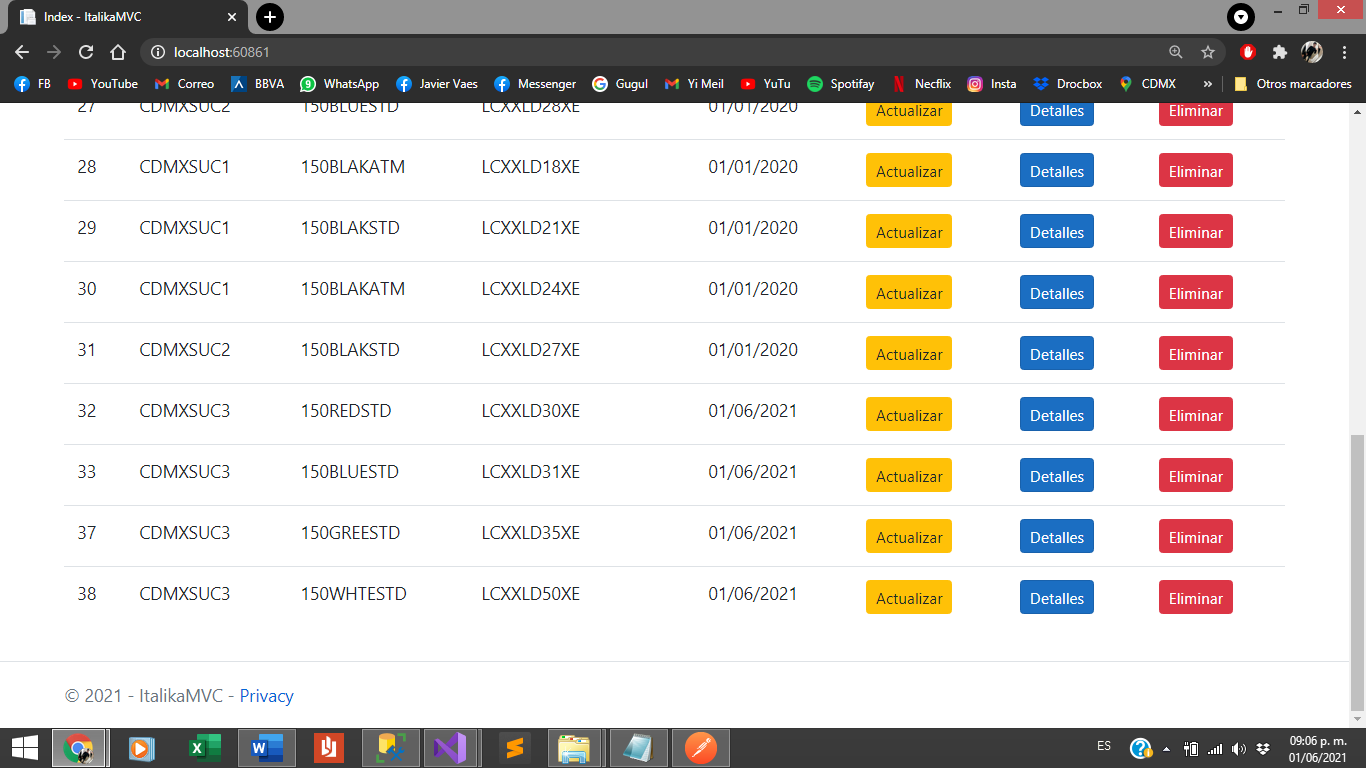
Mostrando el Index con el SKU de sucursal 2, solo los registros con ese SKU aparecen



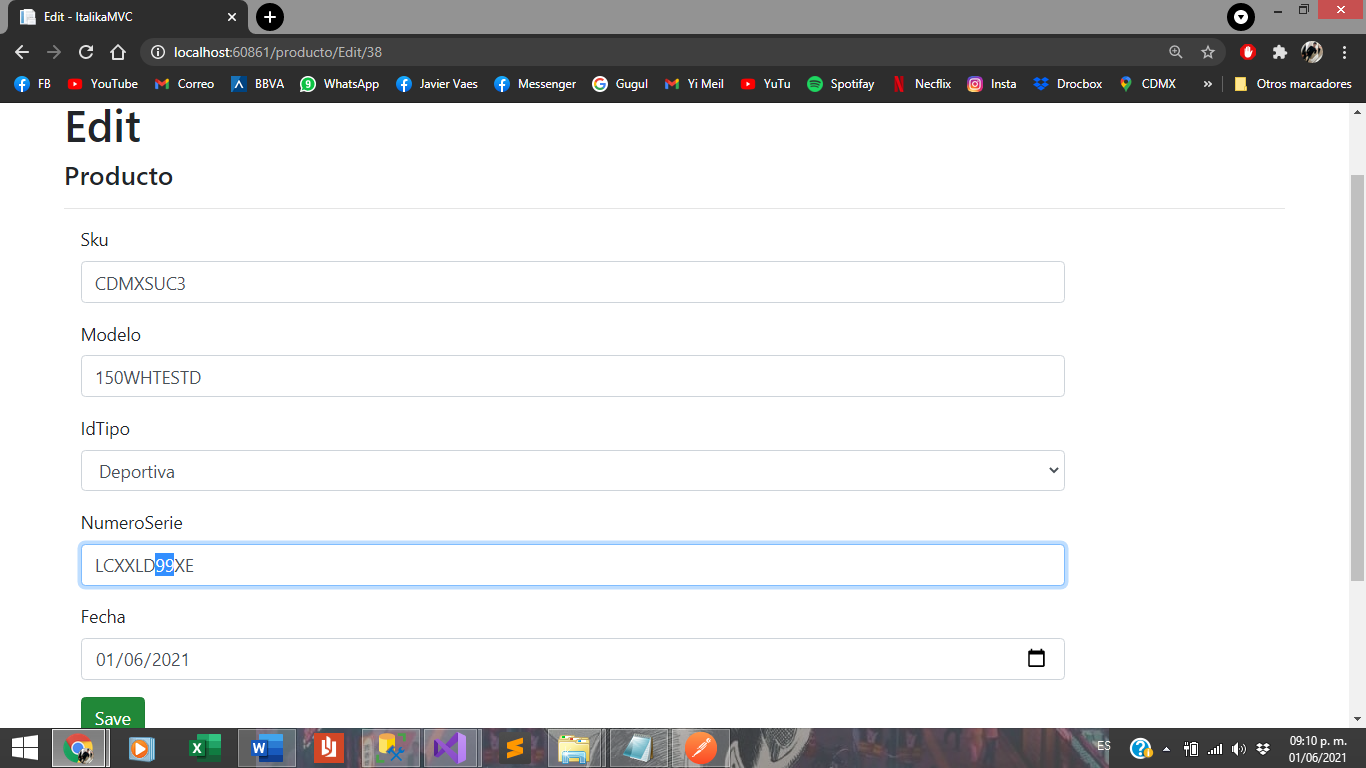
Vista de la creación de un nuevo Producto

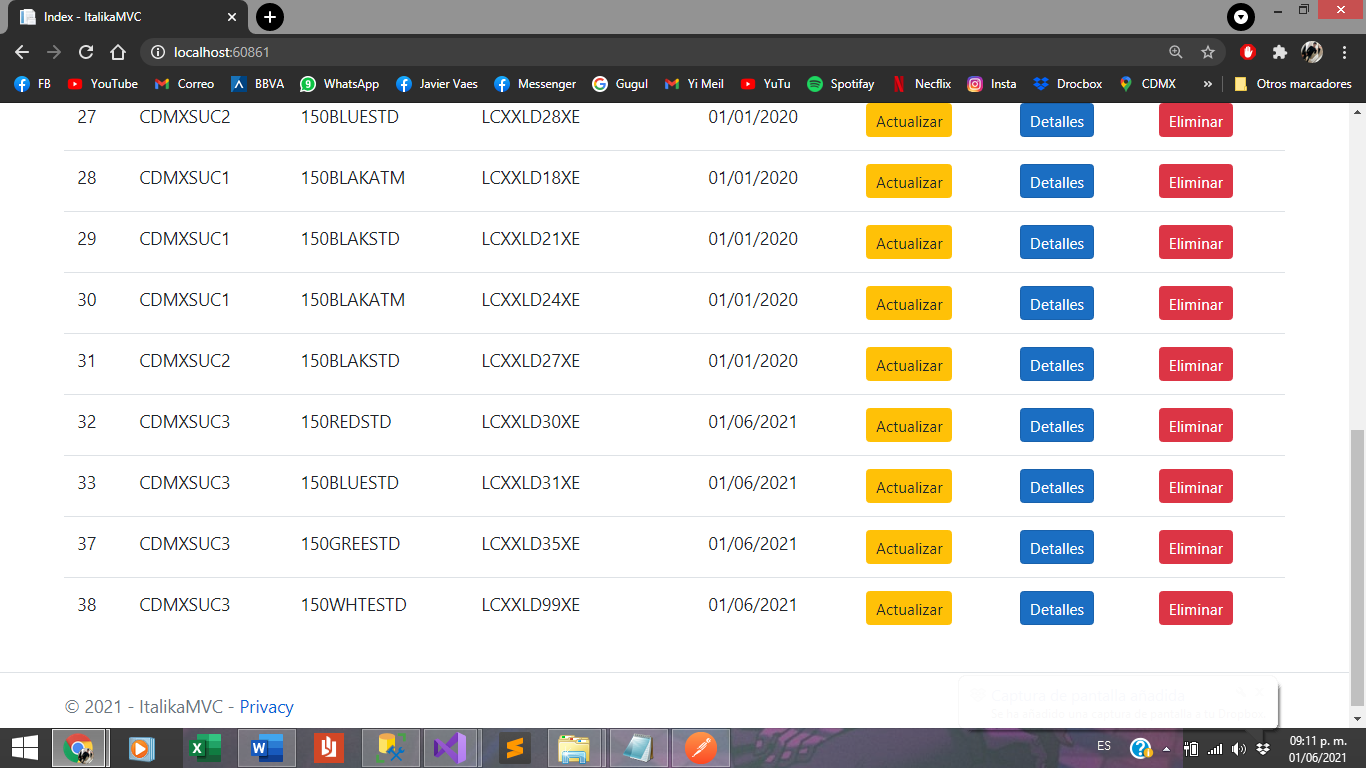


Redireccionamiento y presentación del nuevo registro al fondo de la tabla

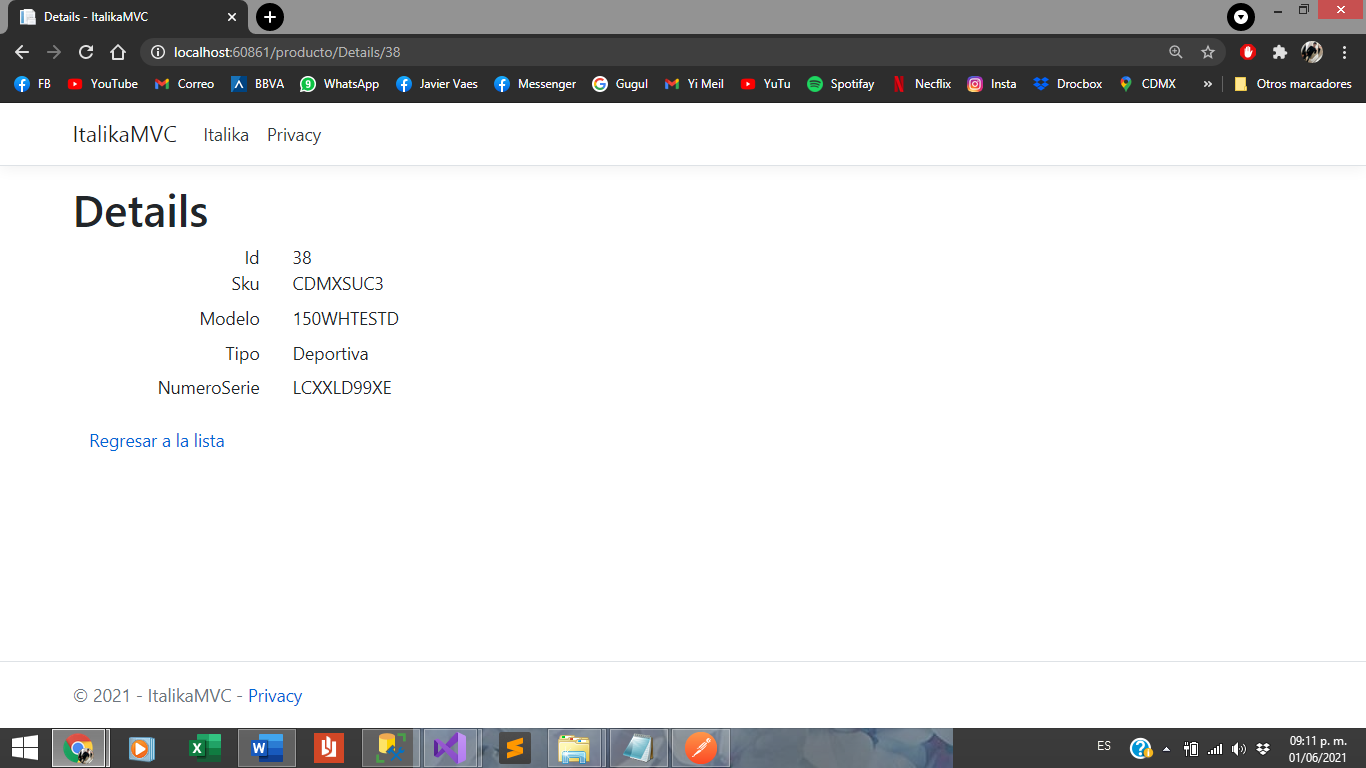
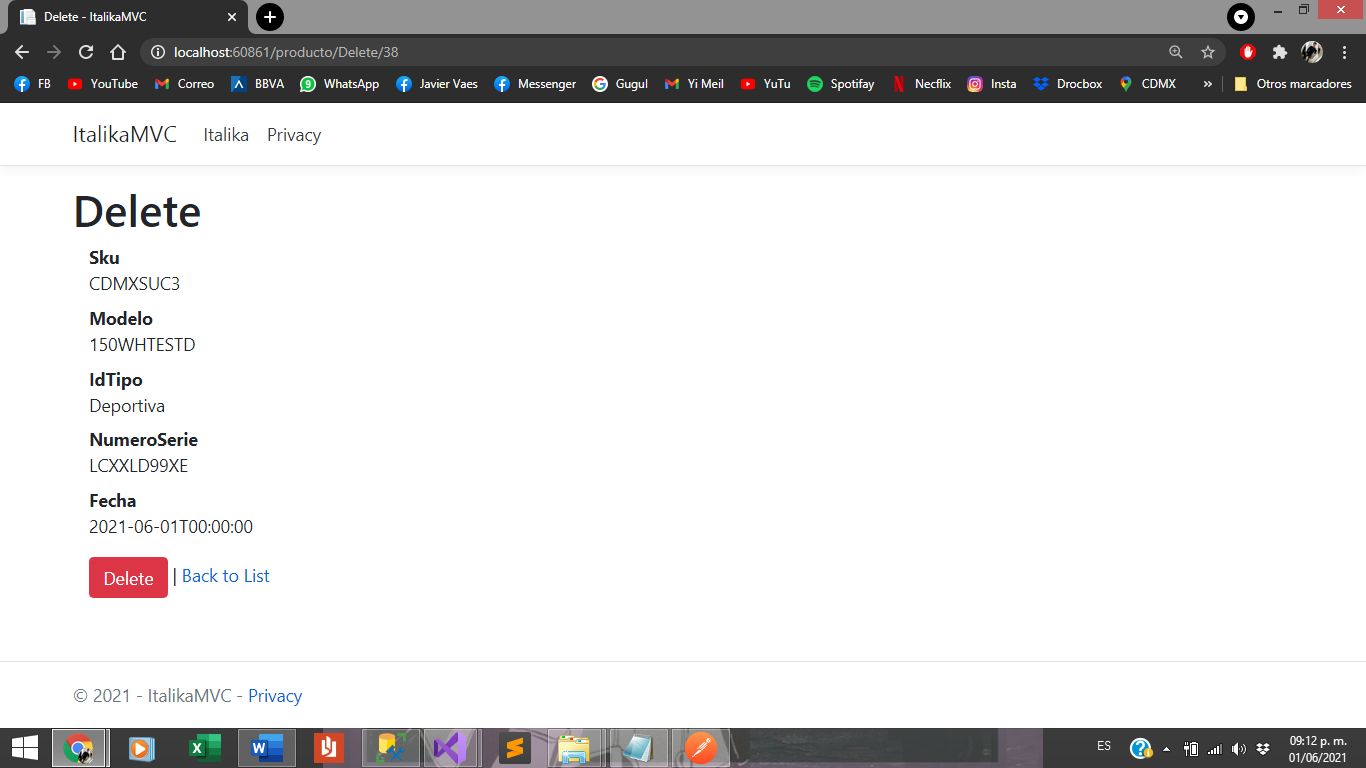


Vista de edición





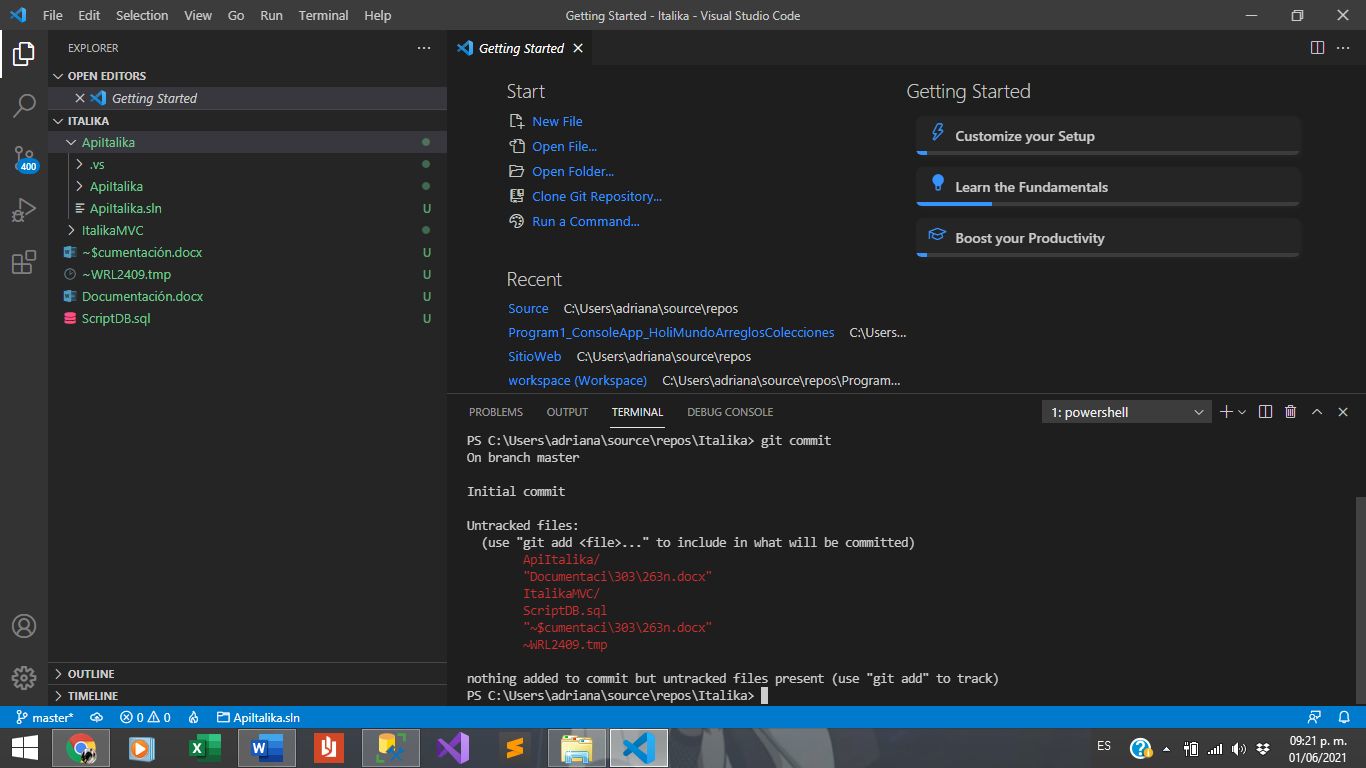
DETALLES ELIMINAR

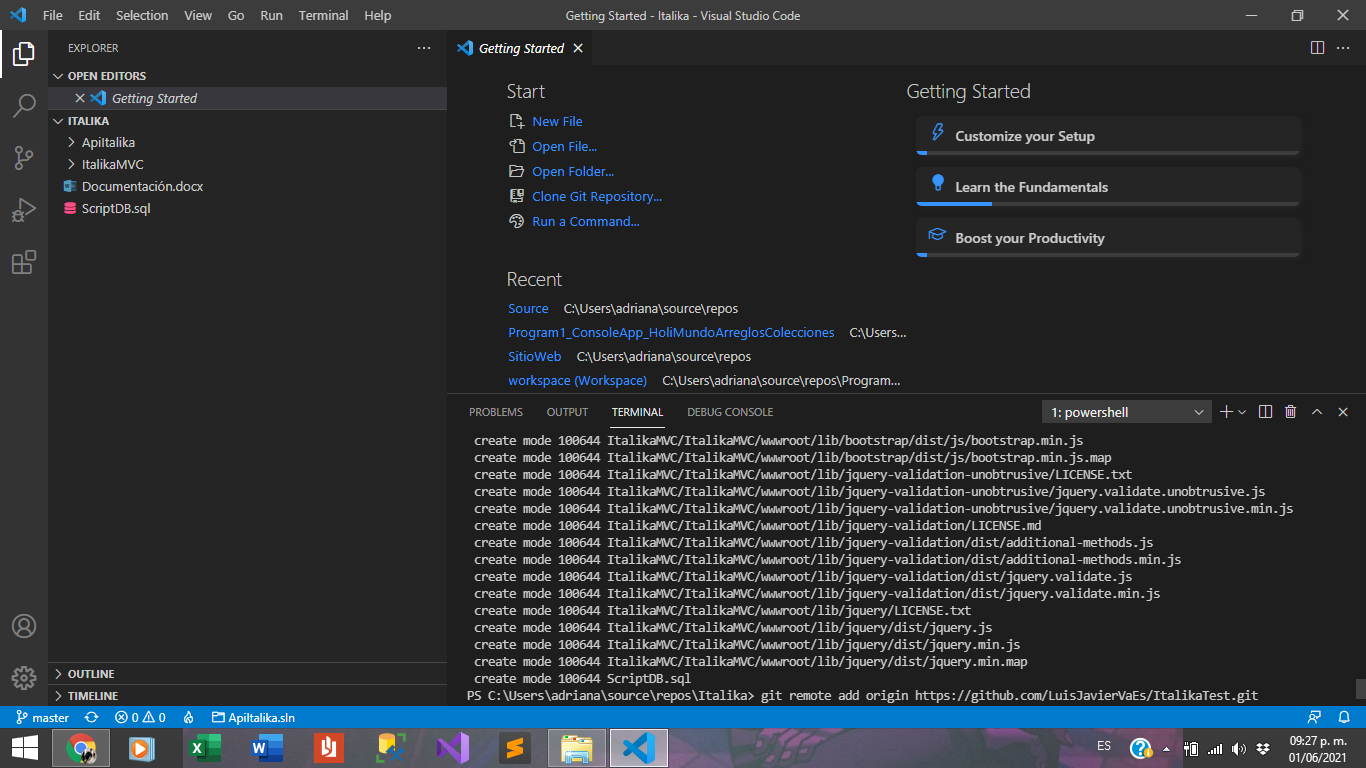
 

Así bien vemos como se refresca de forma correcta, dinámicamente y Asíncrona

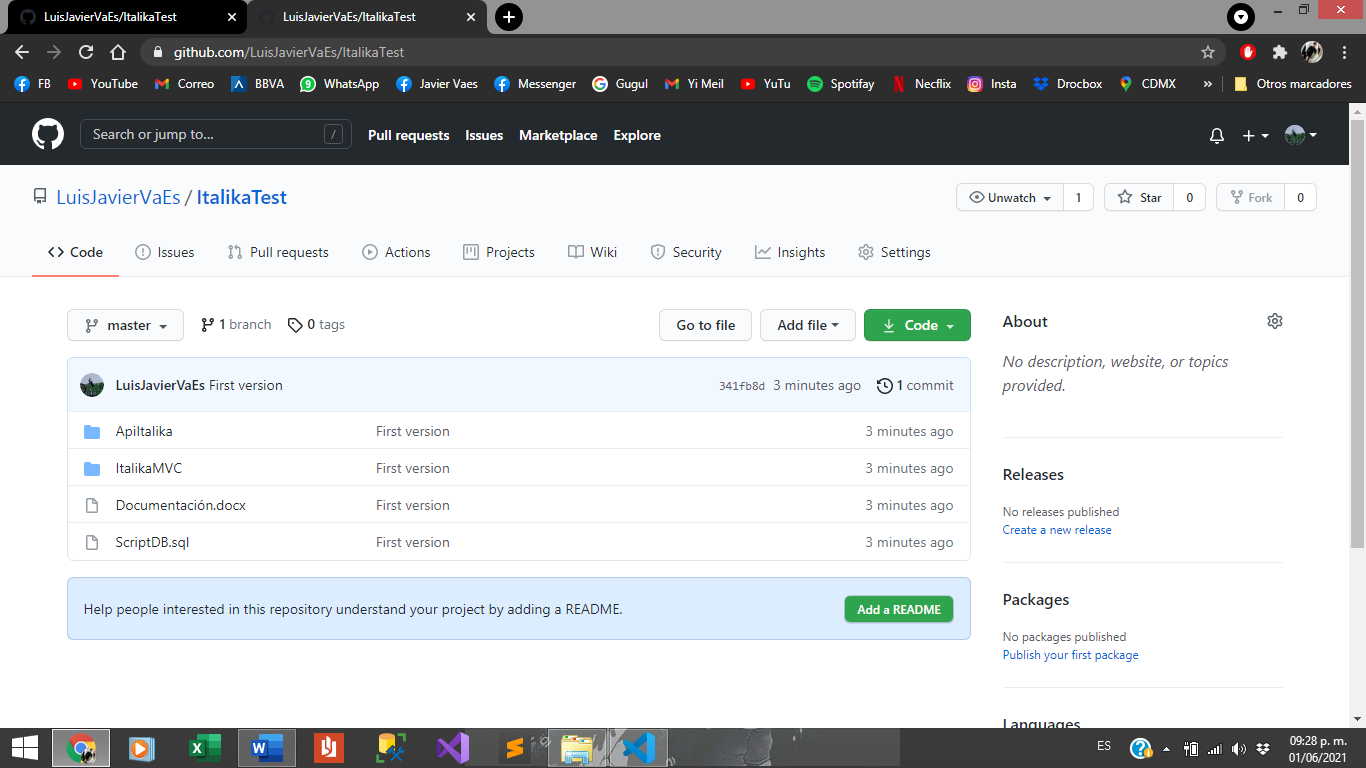
# Link de Repositorio

Subida con Visual Studio Code y Git en GitHub





GITHUB



Repositorio: <https://github.com/LuisJavierVaEs/ItalikaTest.git>