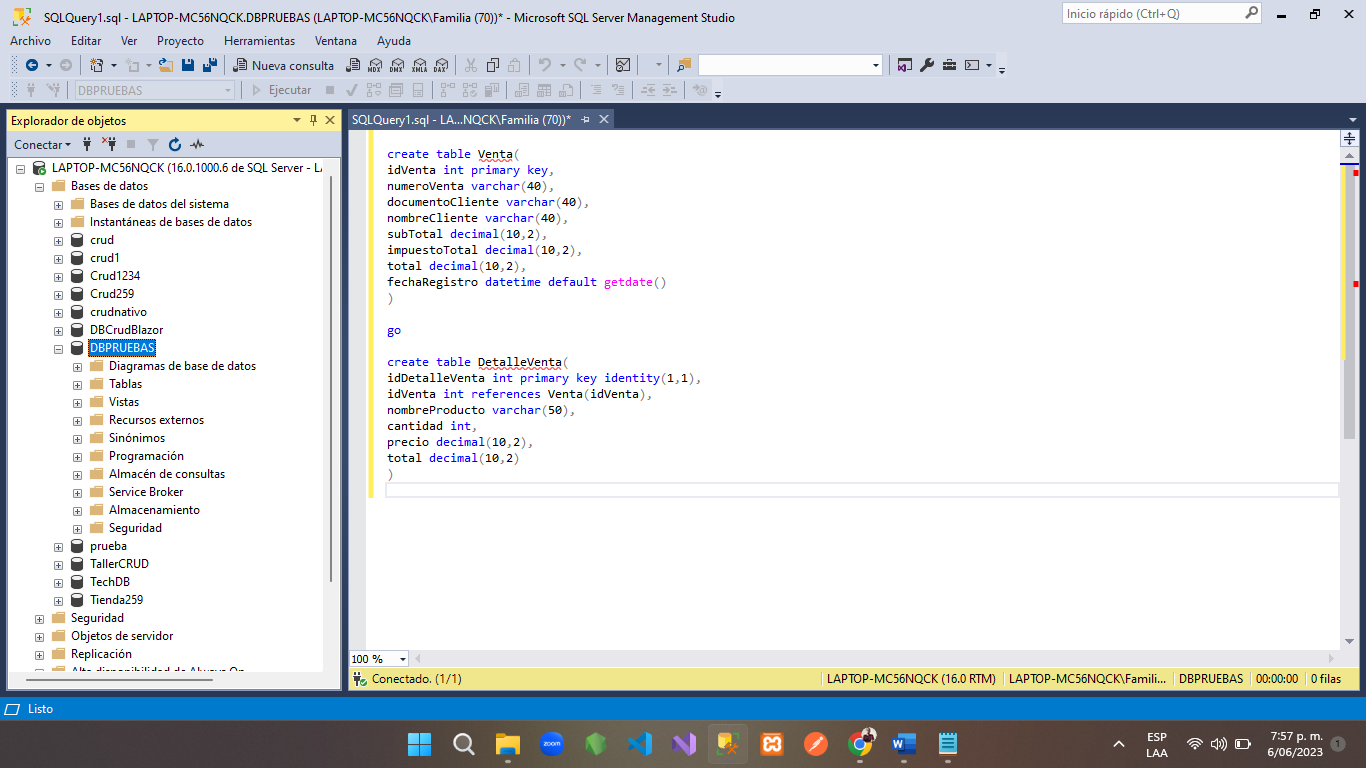
GRAFICOS

Creamos la bd e insertamos datos a la bd



Insertamos registros con el siguiente código:

USE DBPRUEBAS

-- Inserción de registros en la tabla Venta

INSERT INTO Venta (idVenta, numeroVenta, documentoCliente, nombreCliente, subTotal, impuestoTotal, total)

VALUES

(1, 'V0001', '12345678A', 'Cliente A', 100.00, 21.00, 121.00),

(2, 'V0002', '87654321B', 'Cliente B', 200.00, 42.00, 242.00),

(3, 'V0003', '12312312C', 'Cliente C', 150.00, 31.50, 181.50),

(4, 'V0004', '32132132D', 'Cliente D', 250.00, 52.50, 302.50),

(5, 'V0005', '45645645E', 'Cliente E', 300.00, 63.00, 363.00),

(6, 'V0006', '65465465F', 'Cliente F', 350.00, 73.50, 423.50),

(7, 'V0007', '78978978G', 'Cliente G', 120.00, 25.20, 145.20),

(8, 'V0008', '98798798H', 'Cliente H', 400.00, 84.00, 484.00),

(9, 'V0009', '15915915I', 'Cliente I', 275.00, 57.75, 332.75),

(10, 'V0010', '95195195J', 'Cliente J', 500.00, 105.00, 605.00);

INSERT INTO DetalleVenta (idVenta, nombreProducto, cantidad, precio, total)

VALUES

(1, 'Producto 1A', 2, 50.00, 100.00),

(2, 'Producto 2A', 1, 200.00, 200.00),

(3, 'Producto 3A', 3, 50.00, 150.00),

(4, 'Producto 4A', 5, 50.00, 250.00),

(5, 'Producto 5A', 3, 100.00, 300.00),

(6, 'Producto 6A', 7, 50.00, 350.00),

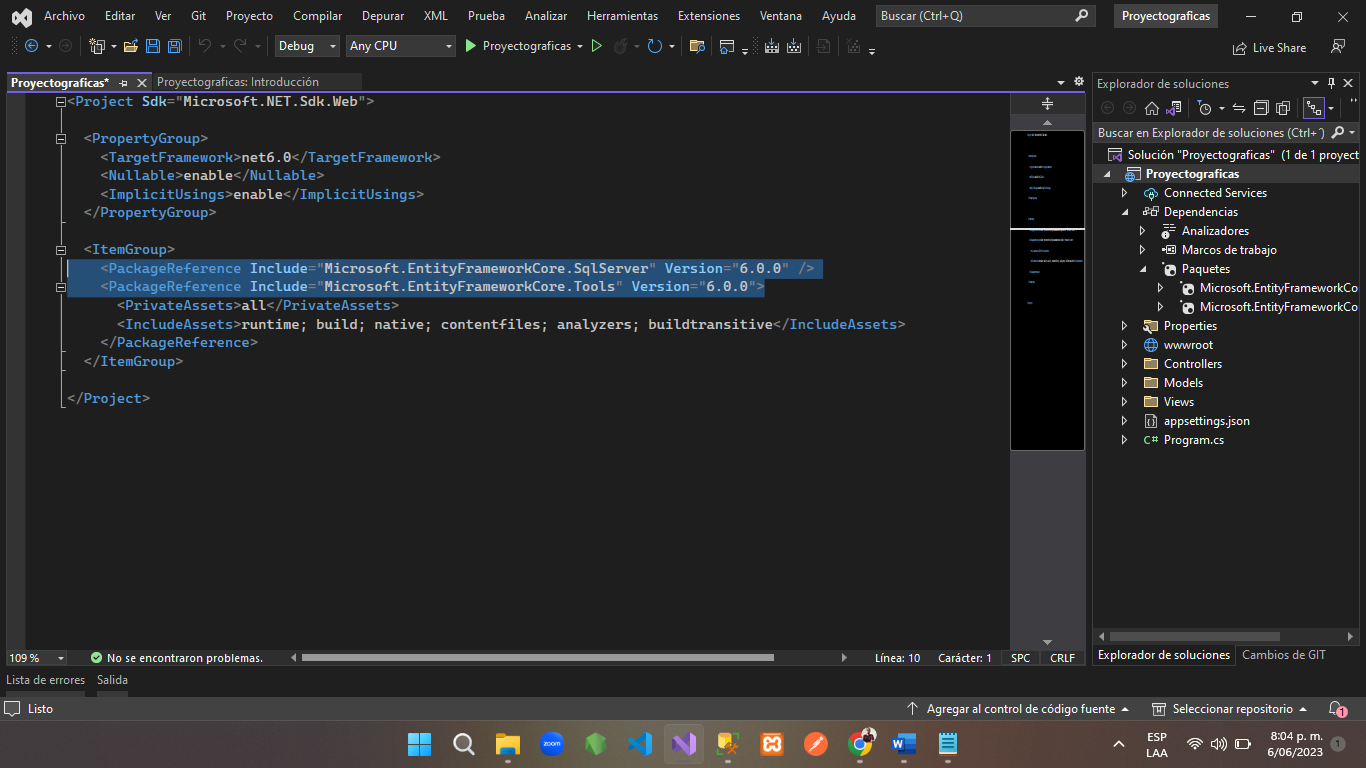
(7, 'Producto 7A', 4, 30.00, 120.00),

(8, 'Producto 8A', 8, 50.00, 400.00),

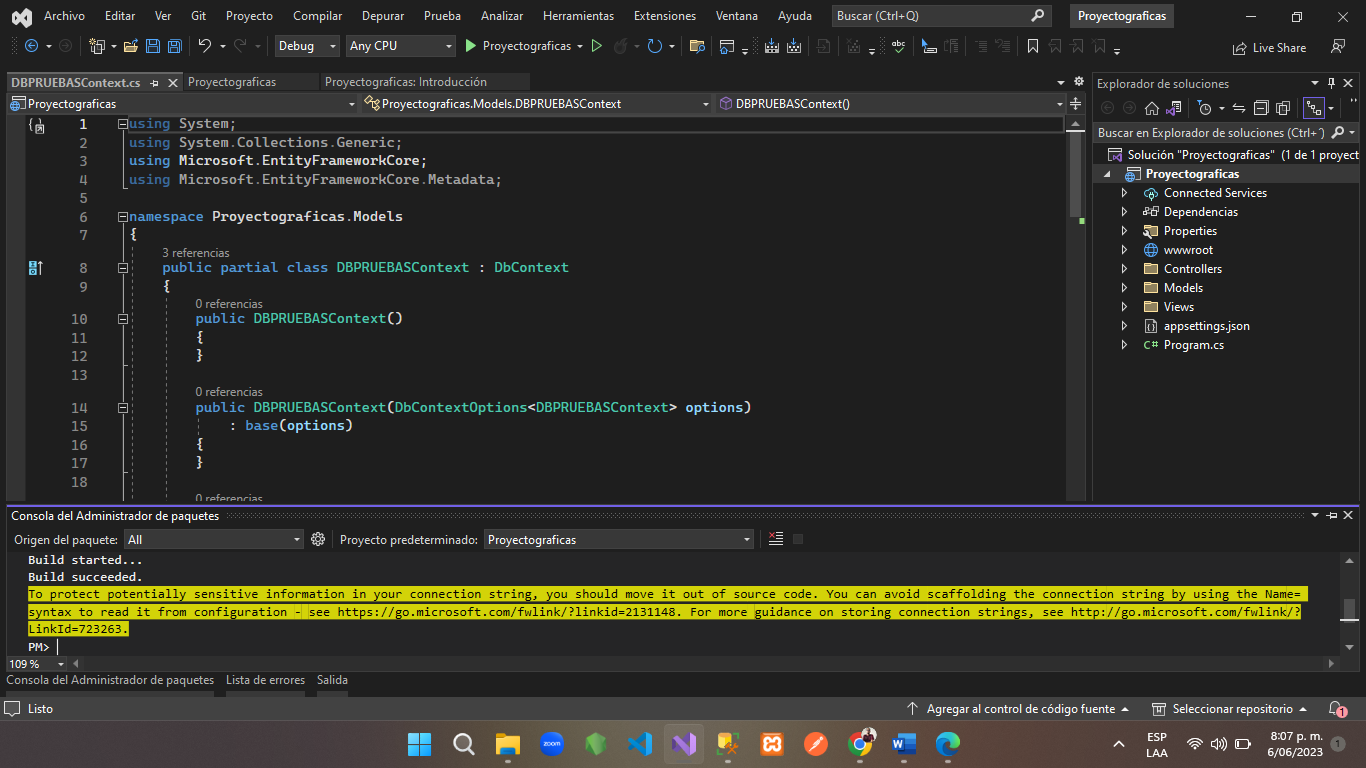
(9, 'Producto 9A', 5, 55.00, 275.00),

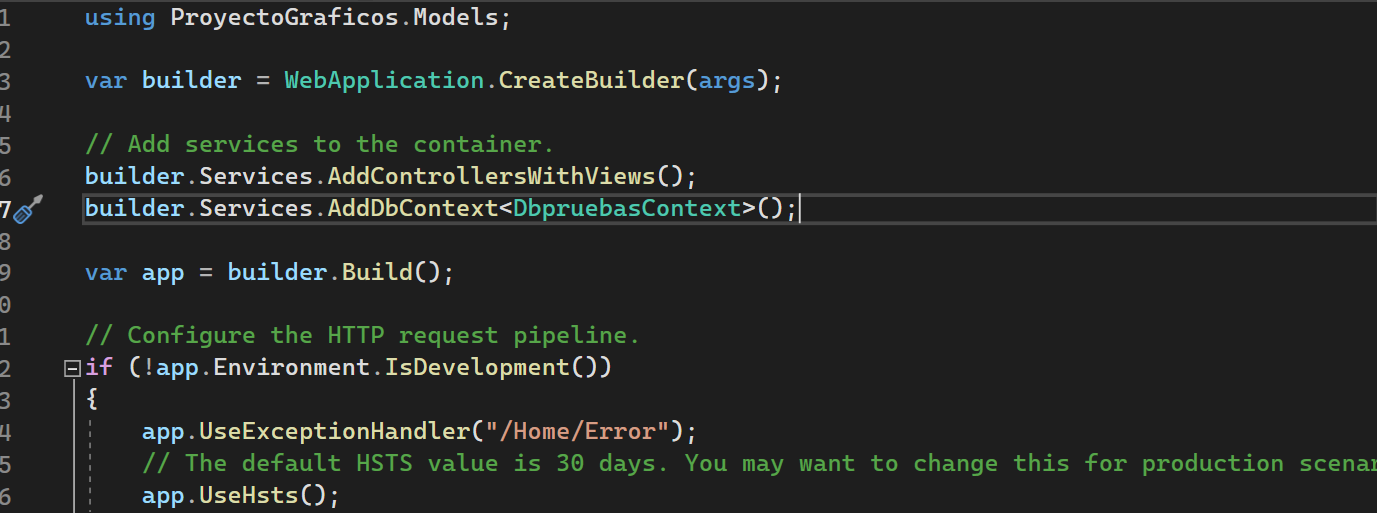
(10, 'Producto 10A', 10, 50.00, 500.00);

Creamos un nuevo proyecto MVC agregamos los dos paguetes nuget Sqlserver y tolos

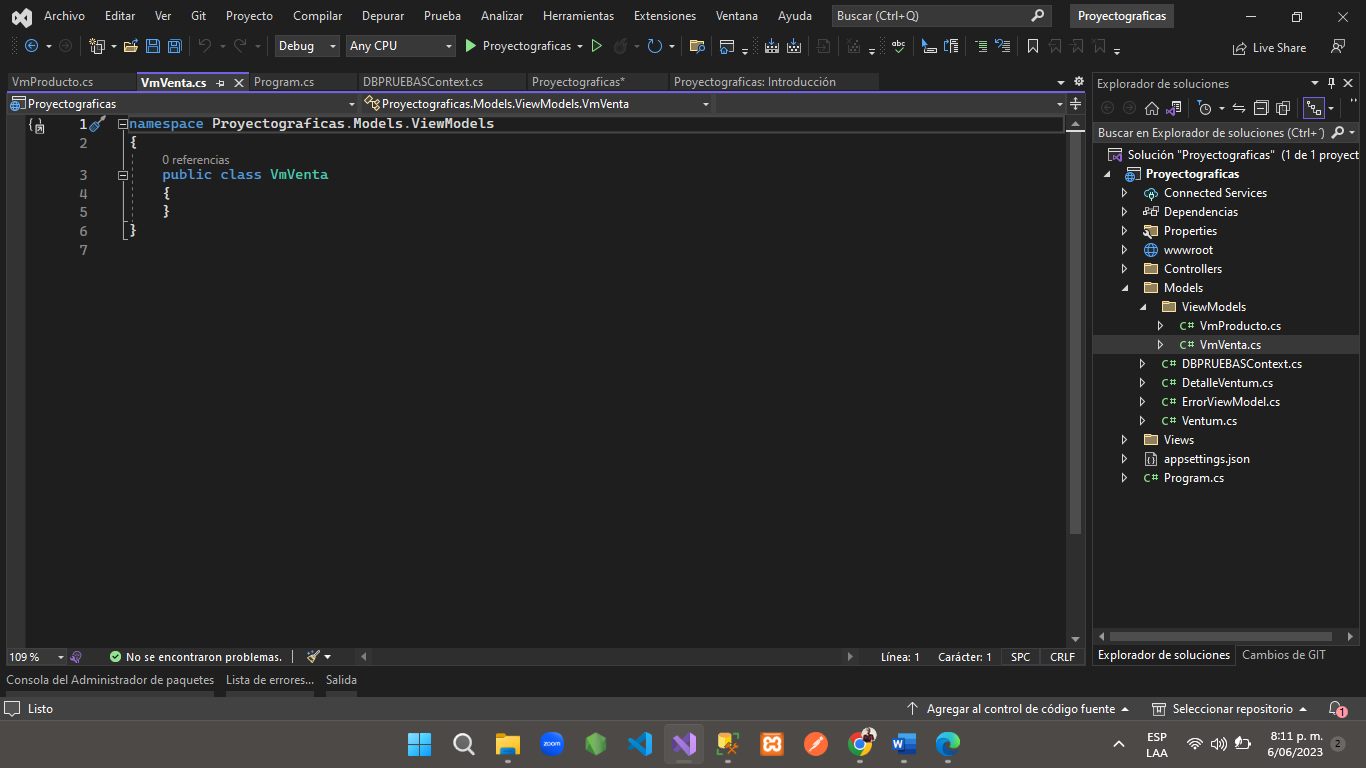


Hacemos un Scaffold-DBContext "Server=localhost;Initial Catalog=crud;integrated security=True; TrustServerCertificate=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models luego hacemos las respectivas configuraciones que nos solicita, mover la cadena de conexión para appseting. Jason y agregar el services en program

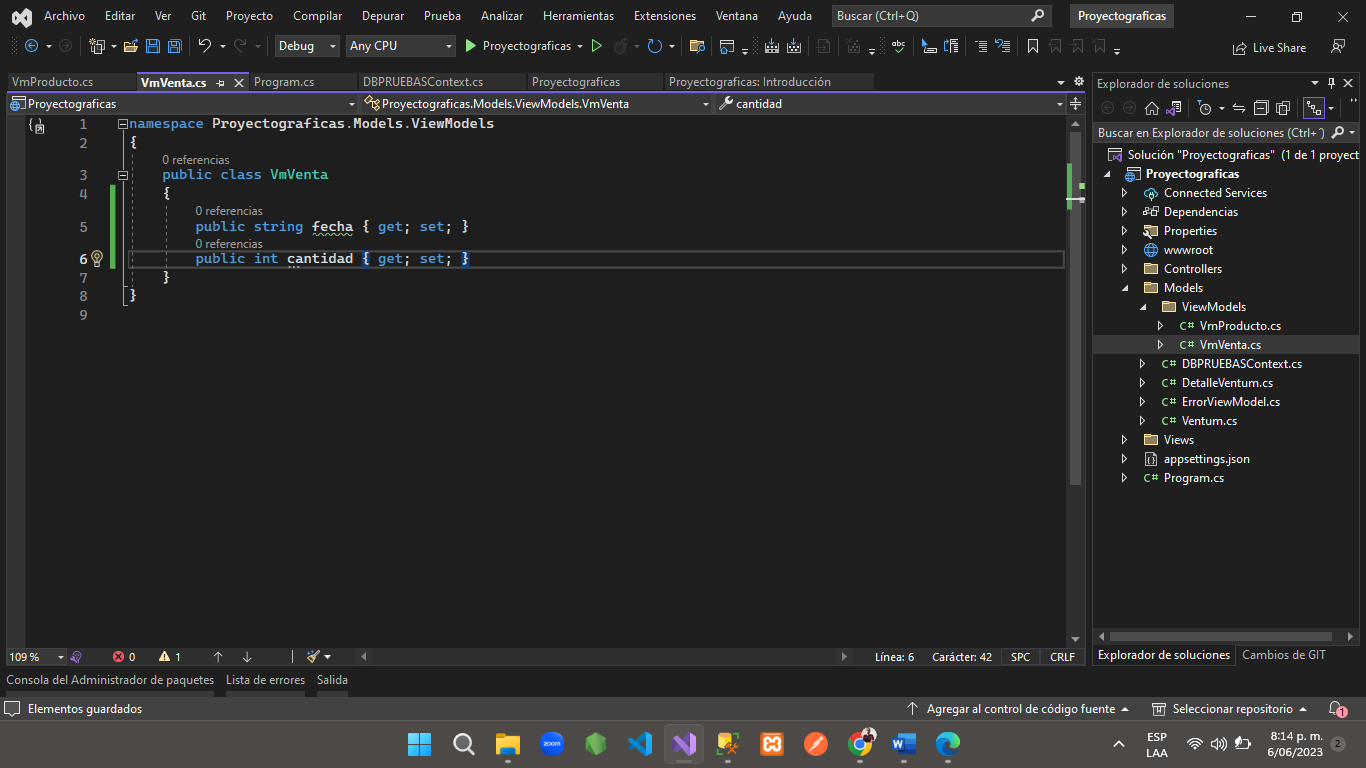




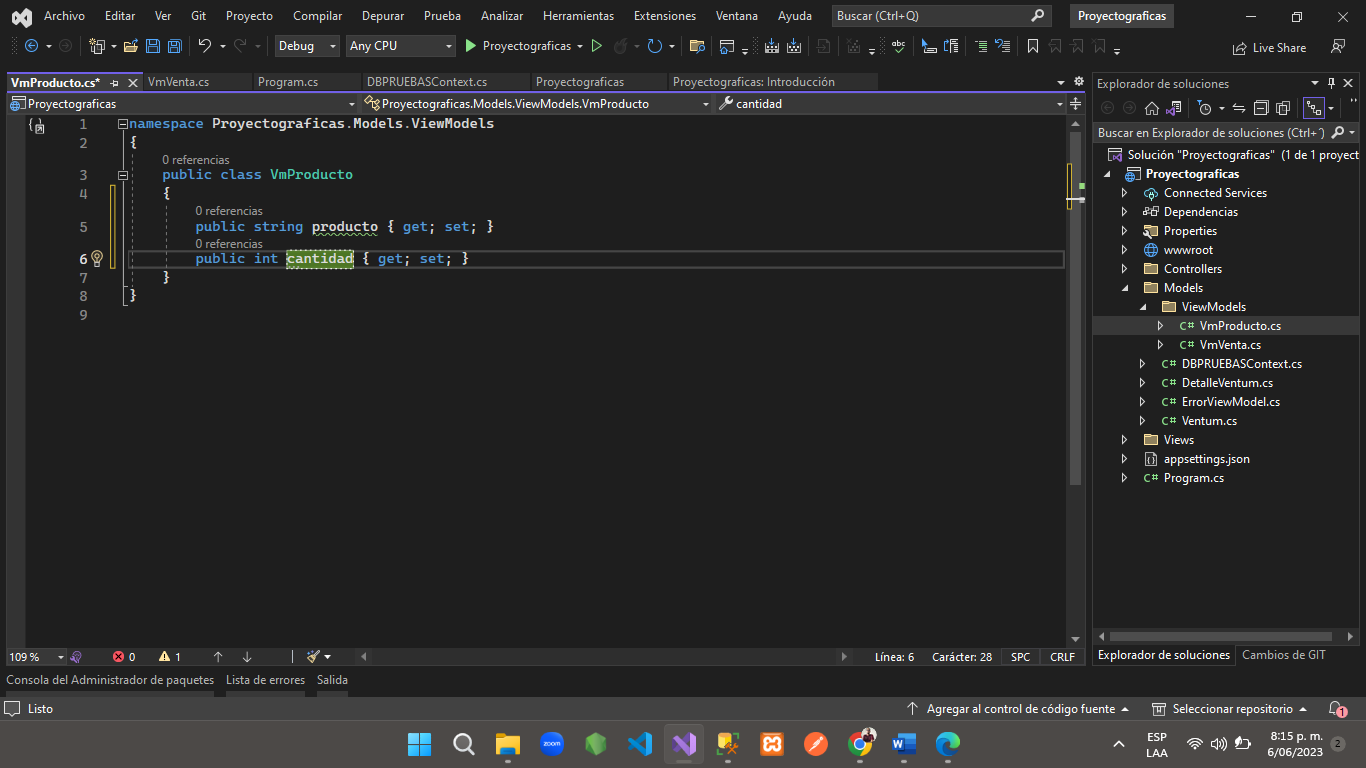
Para mayor orden en el proyecto creamos una carpeta dentro de model llamada viewmodel y dentro de la carpeta creamos dos clases OJO: corregir el nombre de las clases del modelo que seria Venta y DetalleVentas



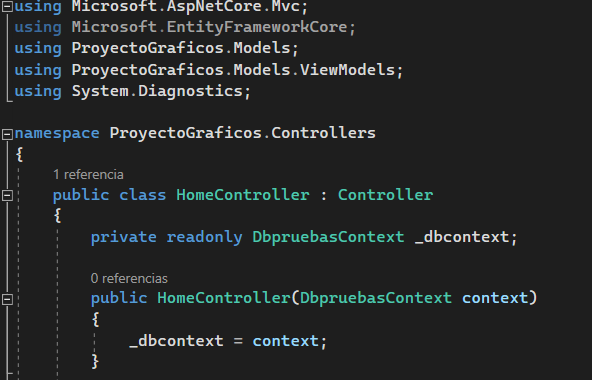
En la clase Vmventa vamos agregar dos propiedades fecha y cantidad

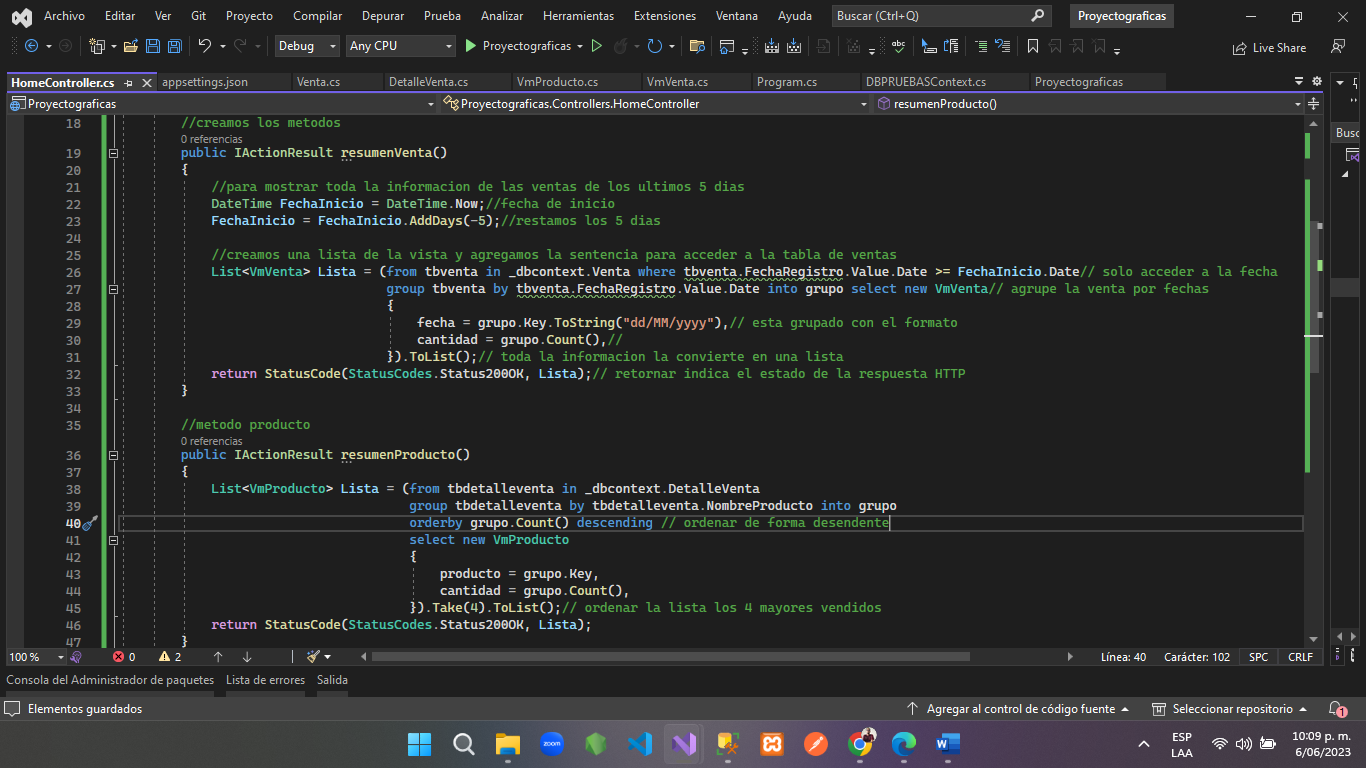


En la clase VmProducto agregamos las propiedades producto y cantidad

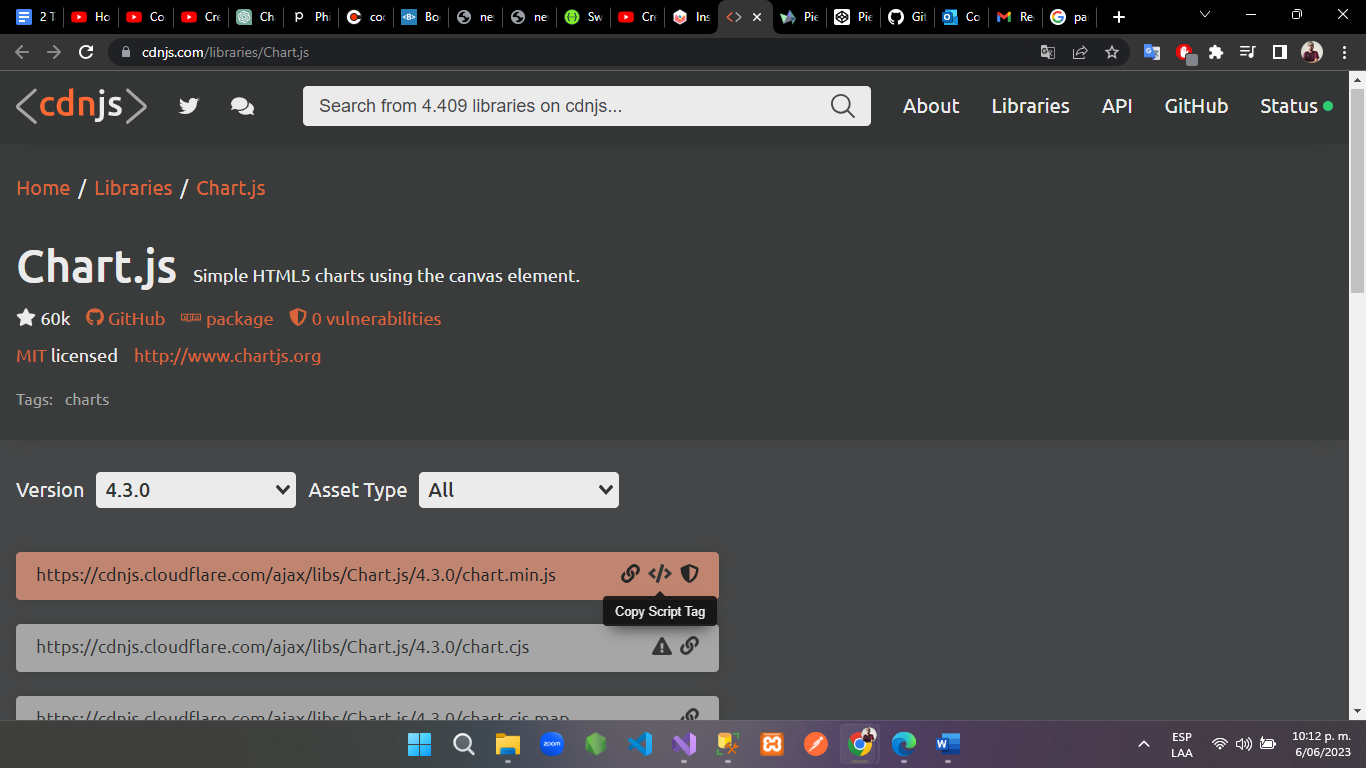


Nos dirigimos al controlador Home y modificamos y agregamos el context y el constructor y creamos los métodos para mostrar en el grafico

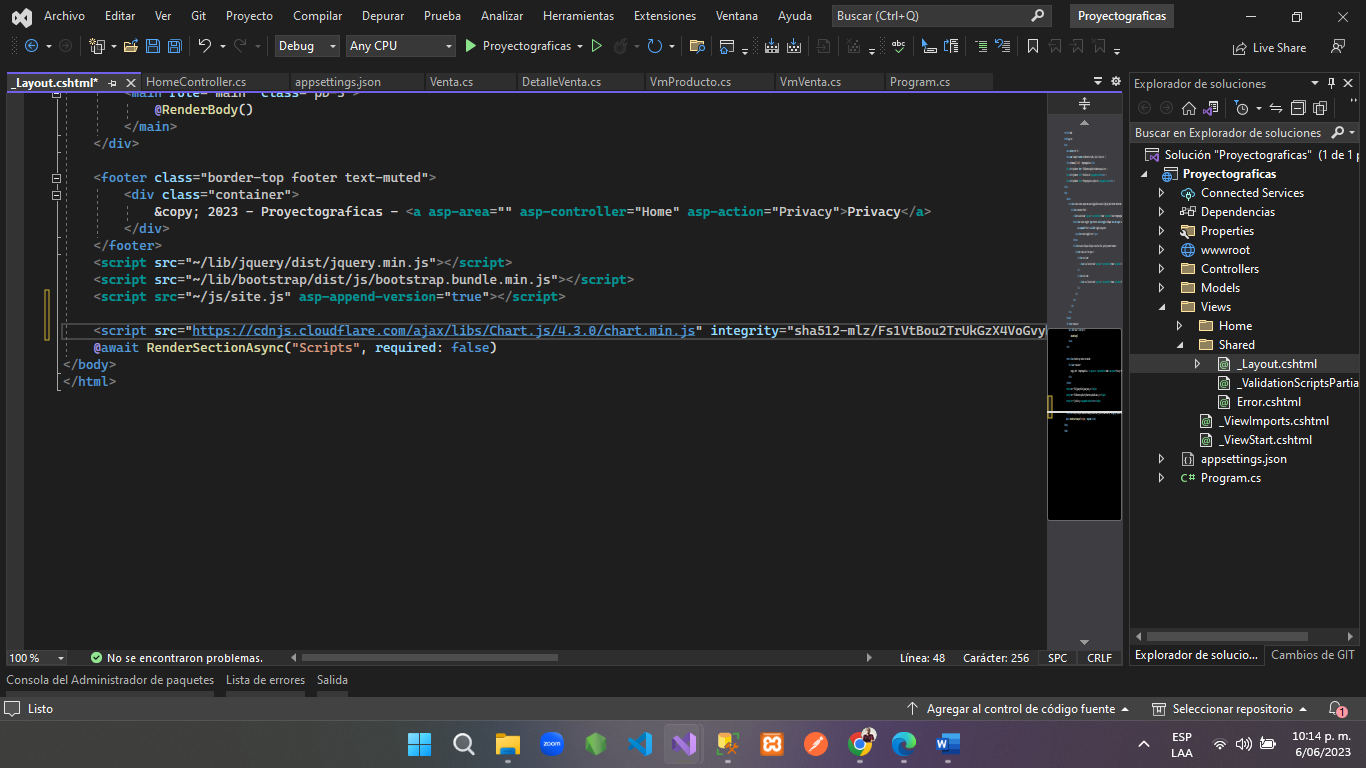




Ahora trabajaremos la parte visible con la librería chart js <https://www.chartjs.org/docs/latest/getting-started/installation.html> y luego ingresamos al siguiente link <https://cdnjs.com/libraries/Chart.js> y copiamos el primer script. Se les recomienda la 3.8.0.



Ahora nos dirigimos a view shared \_layout y pegamos el link



Ahora nos dirigimos a view home index y agregamos diseño de html y css para crear dos tarjetas, luego creamos las funciones de javascript.

Buscamos el grafico en la siguiente dirección. Y copiamos la config y el setup

@{

ViewData["Title"] = "Home Page";

}

<div class="row mt-2">

<div class="col-sm-6">

<div class="card">

<div class="card-header">Ventas ultimos 5 dias</div>

<div class="card-body">

<canvas id="chartVentas" style="max-height:250px"></canvas>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-header">Productos mayor vendidos</div>

<div class="card-body">

<canvas id="chartProductos" style="max-height:250px"></canvas>

</div>

</div>

</div>

</div>

@section Scripts{

<script>

function resumenVenta() {

fetch("Home/resumenVenta")

.then((response) => {

return response.ok ? response.json() : Promise.reject(response);

})

.then((dataJson) => {

const labels = dataJson.map((item) => { return item.fecha })

const values = dataJson.map((item) => { return item.cantidad })

const data = {

labels: labels,

datasets: [{

label: 'cantidad',

data: values,

backgroundColor: 'rgb(54, 162, 235)',

borderWidth: 1

}]

};

const config = {

type: 'bar',

data: data,

options: {

scales: {

y: {

beginAtZero: true

}

}

},

};

const canvasBarras = document.getElementById("chartVentas");

const graficoBarras = new Chart(canvasBarras, config)

}).catch((error) => {

console.log("error")

})

}

function resumenProducto() {

fetch("Home/resumenProducto")

.then((response) => {

return response.ok ? response.json() : Promise.reject(response);

})

.then((dataJson) => {

const labels = dataJson.map((item) => { return item.producto })

const values = dataJson.map((item) => { return item.cantidad })

console.log(labels);

console.log(values);

const data = {

labels: labels,

datasets: [{

label: 'Cantidad',

data: values,

backgroundColor: [

'rgb(255, 99, 132)',

'rgb(54, 162, 235)',

'rgb(255, 205, 86)',

"#01AF40"

],

hoverOffset: 4

}]

};

const config = {

type: 'doughnut',

data: data,

};

const canvasDona = document.getElementById("chartProductos");

const graficoDona = new Chart(canvasDona, config)

}).catch((error) => {

console.log("error")

})

}

$(document).ready(() => {

resumenVenta()

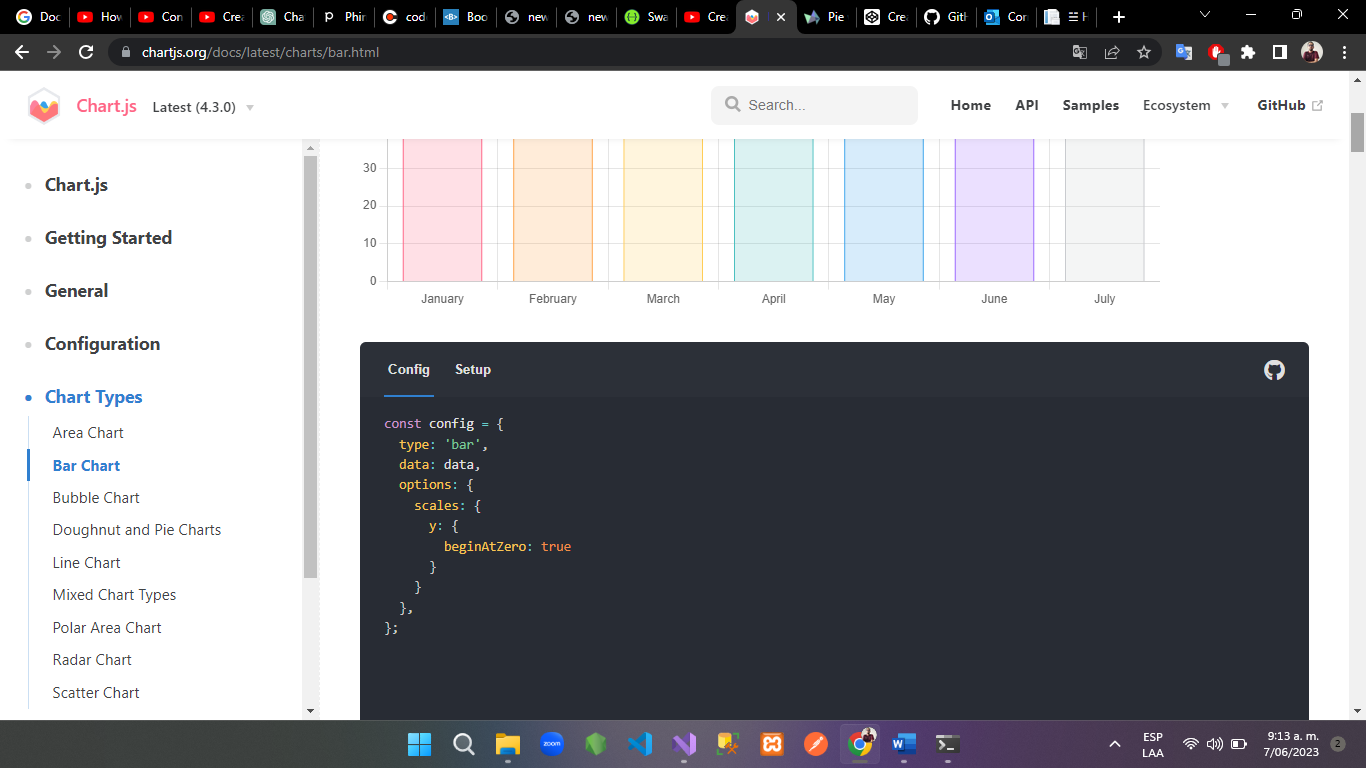
resumenProducto()

})

</script>

}

<https://www.chartjs.org/docs/latest/charts/bar.html>



Ejecutamos el programa y verificamos los resultados

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente