

Manual de Usuario

Nortwind Database

Autor: Enrique Andrés Chávez Martínez

Introducción

En este proyecto he usado la base de datos Northwind para crear un back end en donde podemos modificarla desde Swagger o Postman. Se uso la base de datos SQL Server Express 2019 con un proyecto en Microsoft Visual Studio en NET 5.

Es recomendable usar Microsoft Visual Studio para poder abrir la solución Northwind Database.sln y así poder usar el back end.

Una vez que se corra el programa en Visual Studio se abrirá un localhost que se podrá conectar a un frontend, se abrirá también swagger donde se podrán probar los verbos REST de cada tabla que se soporta de la base de datos de Northwind, estas tablas son Employee, Product y Supplier.

Se puede usar este localhost también para conectarlo a postman y probar los end points ahí.

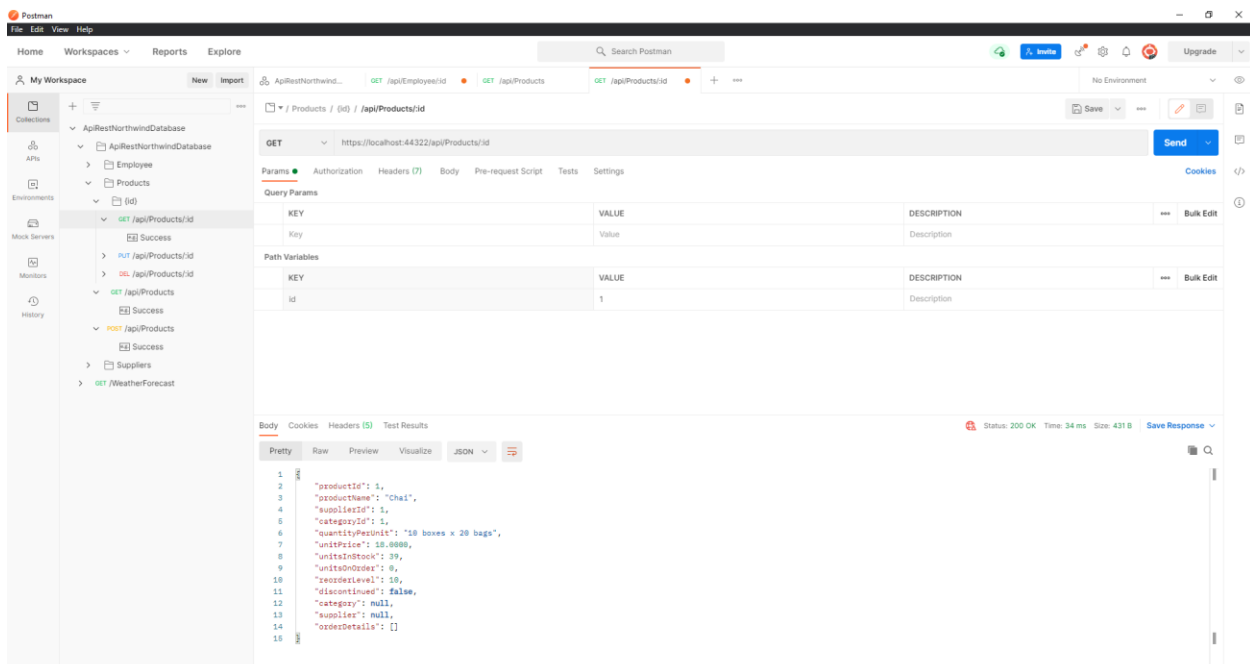
Documentación de cada endpoint en Postman en la tabla de Products de la base de datos Northwind:

GET ID

URL: <https://localhost:44322/api/Products/:id>

Se nos muestra la opción de PATH Variables donde podemos poner el id del producto que queremos ver en el apartado de VALUE, en este caso el ID 1.

No hay parámetros en body que llenar.



SQL Query2.sql - MS-D (MS) (53) - Microsoft SQL Server Enterprise Manager

Object Explorer: MSSQLSERVER (SQL Server 15.0.2000) > Databases > NORTHWIND > Tables > Products

SQL Query2.sql - MS-D (MS) (53) > SQL Query1.sql - MS-D (MS) (53) (50)

```

/***** Script for SelectTop1000s command from SSRS *****/
SELECT TOP (1000) [ProductID]
      , [ProductName]
      , [SupplierID]
      , [CategoryID]
      , [QuantityPerUnit]
      , [UnitPrice]
      , [UnitsInStock]
      , [UnitsOnOrder]
      , [ReorderLevel]
      , [Discontinued]
FROM [NORTHWIND].[dbo].[Products]

```

Results: 100% Messages

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Oat	1	1	10 boxes x 20 bags	18.00	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 500 ml bottles	10.00	13	70	25	0
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
6	6	Garden of Eatin' Boyberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0
7	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0
8	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
9	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1
10	10	Ika	4	8	12 - 200 ml jars	31.00	31	0	0	0
11	11	Queso Cabrales	5	4	1 kg pkg.	21.00	22	30	30	0
12	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38.00	86	0	0	0
13	13	Korbu	6	8	2 kg box	6.00	24	0	5	0
14	14	Tofu	6	7	40 - 100 g pkgs.	23.25	35	0	0	0
15	15	Garden of Eatin' Shoupsi	6	2	24 - 250 ml bottles	15.50	39	0	5	0
16	16	Pavlova	7	3	32 - 500 g boxes	17.45	29	0	10	0
17	17	Alice Mutton	7	6	20 - 1 kg tins	39.00	0	0	0	1
18	18	Camaron Tigris	7	8	16 kg pkg.	62.50	42	0	0	0
19	19	Tesame Chocolate Biscuits	8	3	10 boxes x 12 pieces	9.20	25	0	5	0

Query executed successfully.

Podemos ver en la base de datos que los datos que se pidieron son correctos.

PUT ID

URL: <https://localhost:44322/api/Products/id>

Igual que en GET ID ponemos el id del producto que queremos modificar su nombre , en este caso el 1.

Postman

Home Workspaces Reports Explore

My Workspace

Collection: ApiRestNorthwindDatabase

Environments: No Environment

Mock Servers: Success

History: Success

Products: Success

Suppliers: Success

WeatherForecast: Success

Products / (id) / Api/Products/id

PUT https://localhost:44322/api/Products/id

Params: Authorization Headers (10) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
id	1	Description

Path Variables

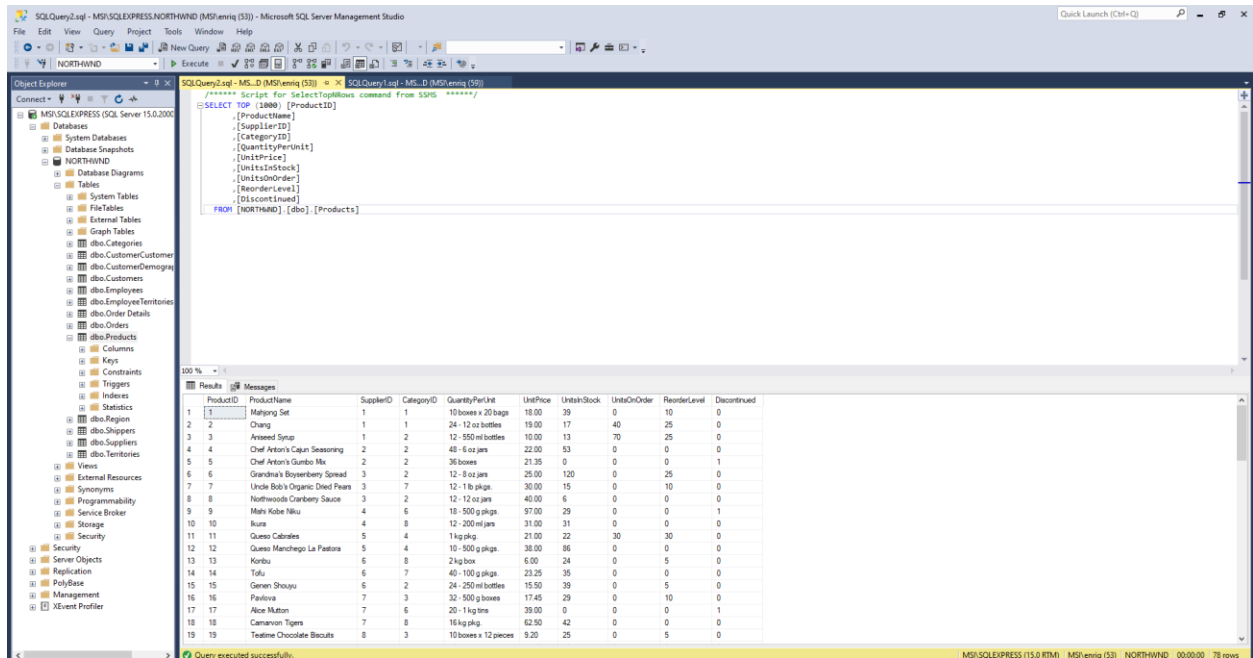
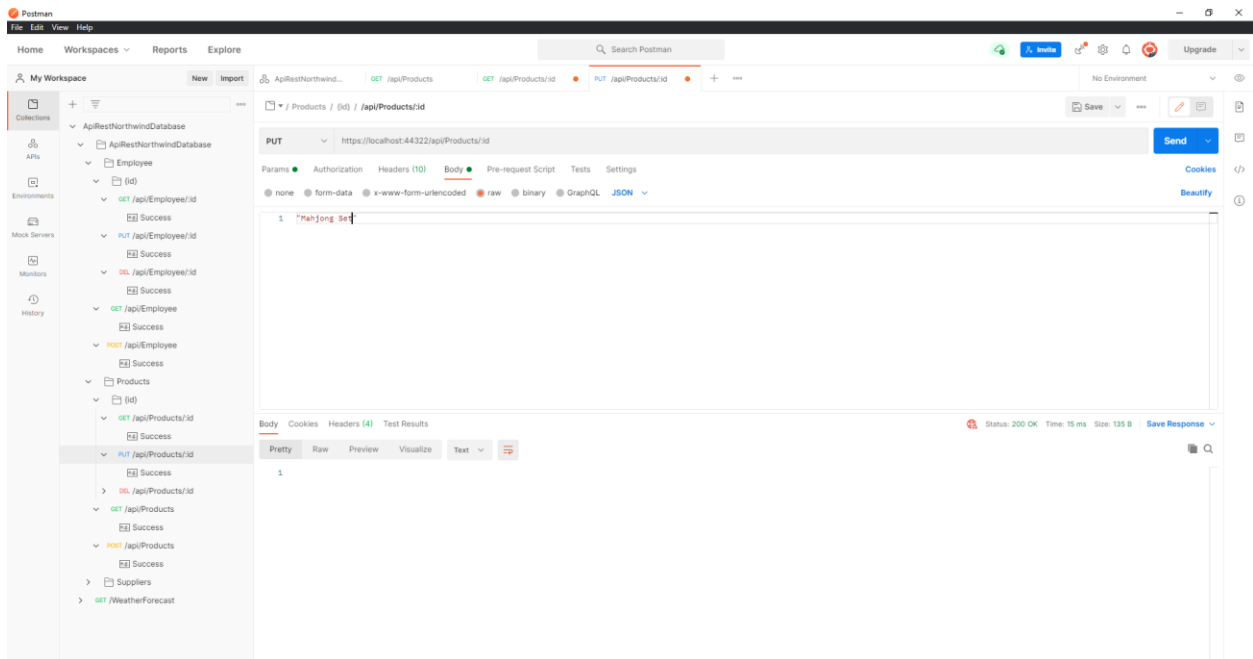
KEY	VALUE	DESCRIPTION
id	1	(Required)

Body Cookies Headers (4) Test Results

Body: {}

Status: 200 OK Time: 15 ms Size: 135 B Save Response

En body modificamos el nombre del producto al que queramos, en este caso de Chai a Mahjong Set y damos enviar.

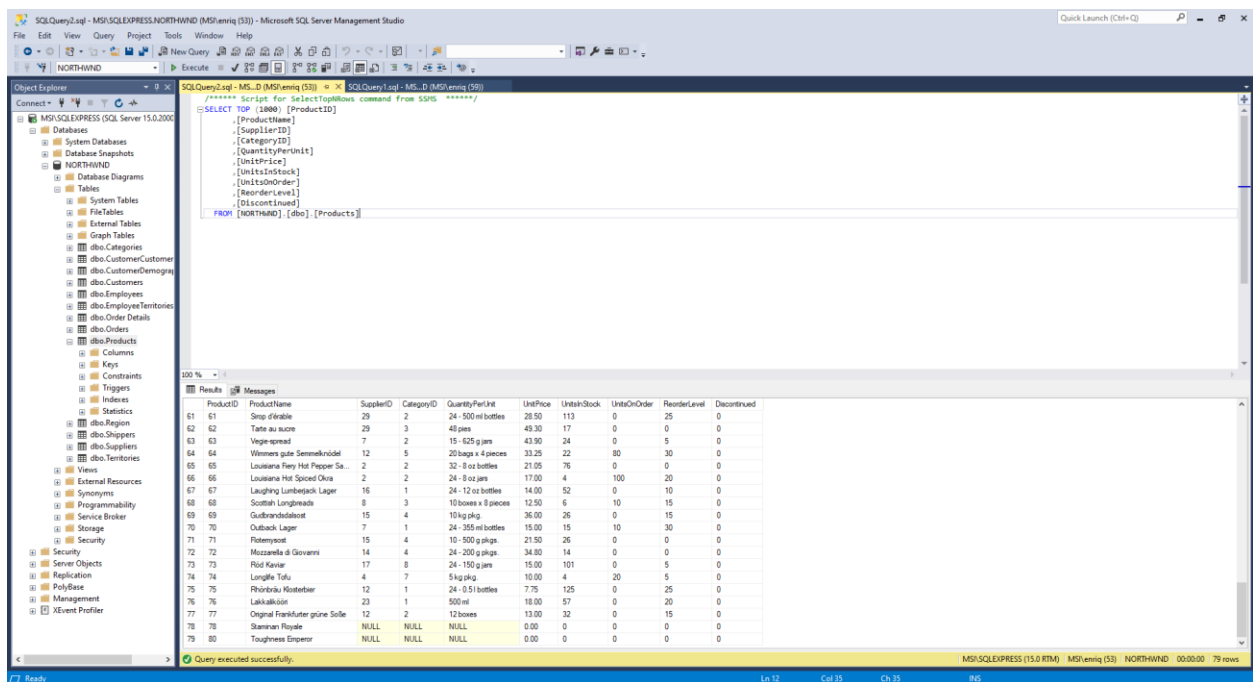
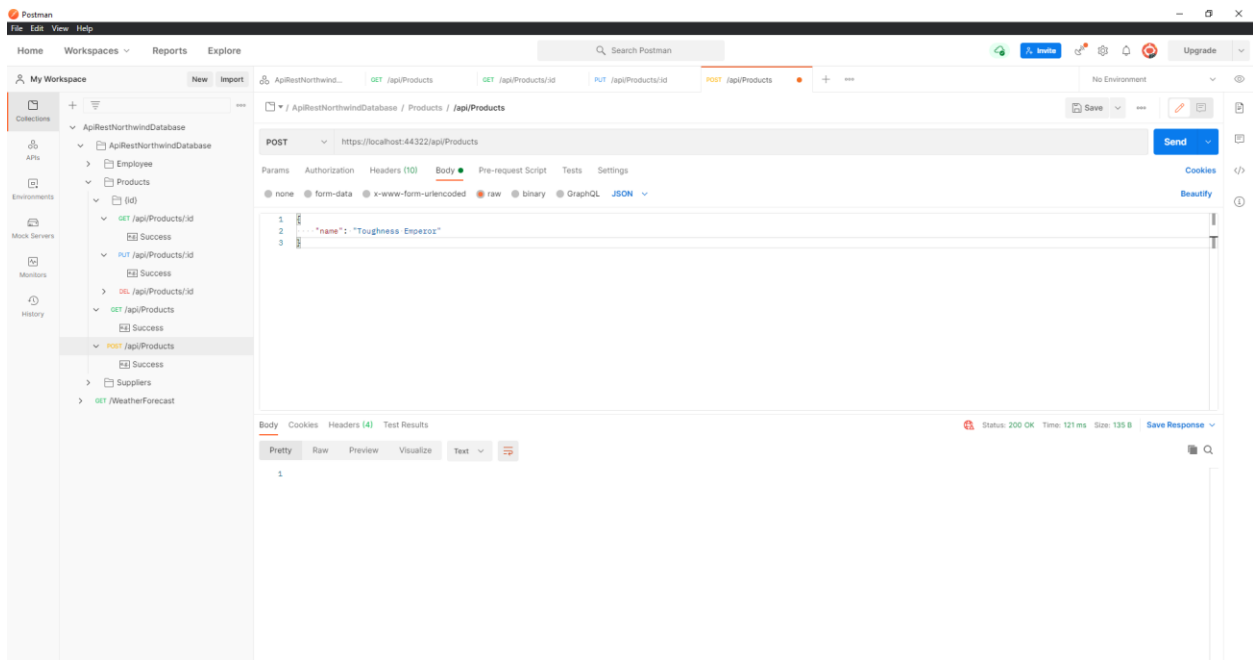


Podemos ver en la base de datos que se modificó correctamente.

POST

URL: <https://localhost:44322/api/Products>

En body ponemos los datos que se van a usar para crear un nuevo producto en la tabla, en este caso solo el nombre del producto que en este ejemplo es Toughness Emperor.

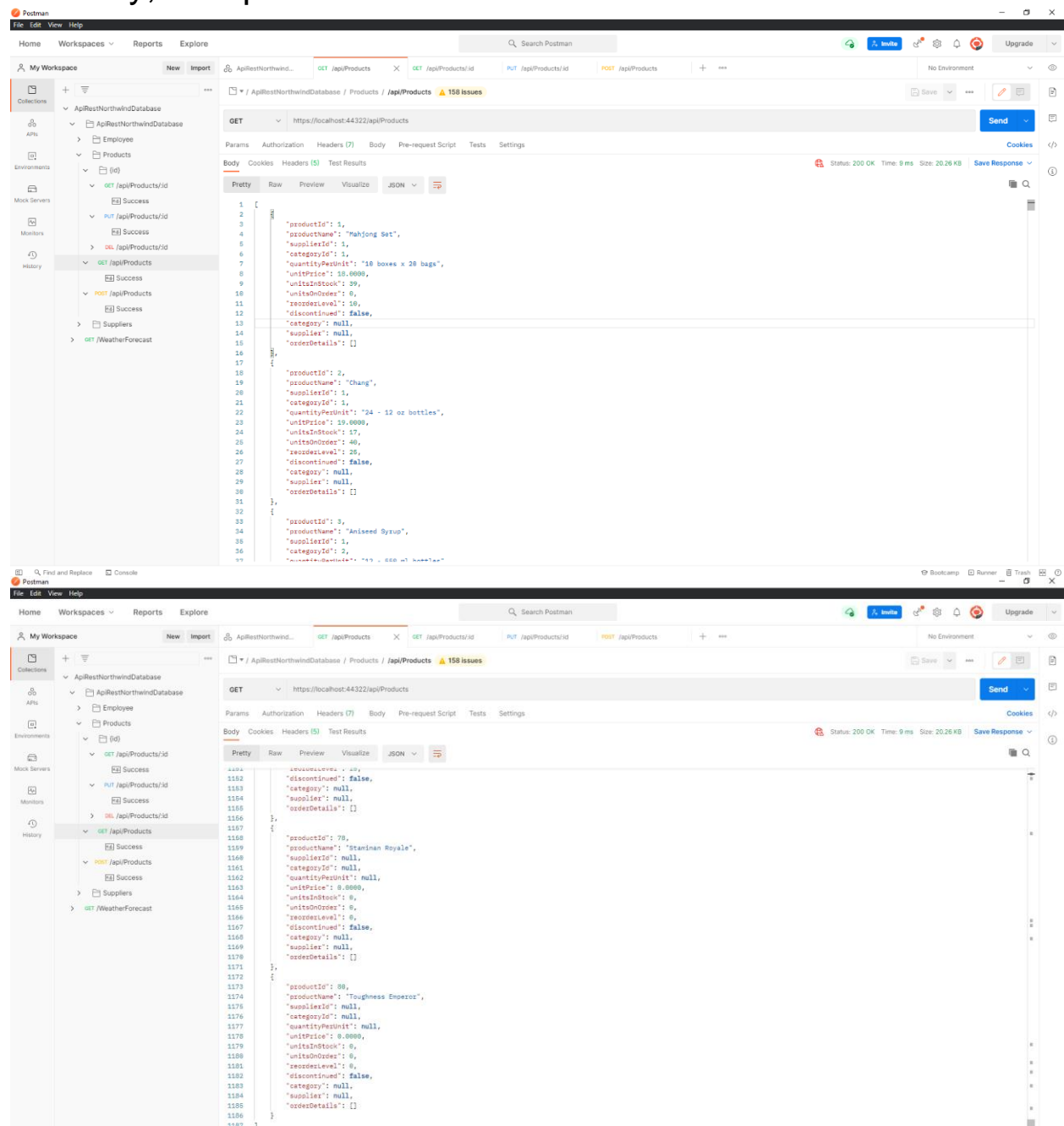


Como podemos ver en la base de datos, se agregó el producto deseado.

GET ALL

URL: <https://localhost:44322/api/Products>

En este endpoint no se necesita modificar nada de parámetros o del body, solo pedimos la tabla.



SQLQuery2.sql - MS-SQLSERVER.NORTHWND (MS)enmq (53) - Microsoft SQL Server Management Studio

Object Explorer: MSSQLSERVER (SQL Server 15.0.2000) > Databases > NORTHWND > Tables > dbo.Products

Results: Messages

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	Mahjong Set	1	1	10 boxes x 20 bags	18.00	39	0	10	0
2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
3	Ananas Spruz	1	2	12 - 160 ml bottles	10.00	13	70	25	0
4	Chef Anton's Cajun Seasoning	2	2	40 - 5 oz jars	22.00	53	0	0	0
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1
10	Ikuu	4	8	12 - 200 ml jars	31.00	31	0	0	0
11	Que Pasa	5	4	1 kg pkg.	21.00	22	30	30	0
12	Que Pasa Manchego La Pastora	5	4	10 - 500 g pkgs.	38.00	86	0	0	0
13	Korbu	6	8	2 kg box	6.00	24	0	5	0
14	Tofu	6	7	40 - 100 g pkgs.	23.25	35	0	0	0
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.50	39	0	5	0
16	Pavane	7	3	32 - 500 g boxes	17.45	29	0	10	0
17	Alice Mutton	7	6	20 - 1 kg tins	39.00	0	0	0	1
18	Camaron Tigers	7	8	16 kg pkg.	62.50	42	0	0	0
19	Toutaine Chocolate Biscuits	8	3	10 boxes x 12 pieces	9.20	25	0	5	0
20	Sir Rodney's Marmalade	8	3	30 gft boxes	81.00	40	0	0	0
21	Sir Rodney's Scones	8	3	24 pkgs x 4 pieces	10.00	3	40	5	0
22	Gustaf's Knackebrod	9	5	24 - 500 g pkgs.	21.00	104	0	25	0
23	Turnbird	9	5	12 - 250 g pkgs.	9.00	61	0	25	0
24	Guaraná Fantástica	10	1	12 - 355 ml cans	4.50	20	0	0	1
25	NuNuCa Nu-Nu-Cajun Creme	11	3	20 - 450 g pkgs	14.00	76	0	30	0
26	Gumbur Lumbreakchen	11	3	100 - 250 g pkgs	31.23	15	0	0	0
27	Schoggi Schokolade	11	3	100 - 100 g pieces	43.90	49	0	30	0
28	Rössle Sauerkraut	12	7	25 - 825 g cans	45.60	26	0	0	1
29	Thüringer Rostbratenst.	12	6	50 bags x 30 sausages	123.79	0	0	0	1
30	Nord-Ost Magerwurst	13	5	10 - 200 g packages	25.99	10	0	15	0
31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12.50	0	70	20	0
32	Mascarpone Fabbri	14	4	24 - 200 g pkgs.	32.00	9	40	25	0
33	Gerdot	15	4	500 g	2.50	112	0	20	0
34	Sasquatch Ale	16	3	24 - 12 oz bottles	14.00	11	0	15	0
35	Steeleye Stout	16	1	24 - 12 oz bottles	18.00	20	0	15	0
36	Inland NW	17	8	24 - 250 g jars	19.00	112	0	20	0
37	Grand Iux	17	8	12 - 500 g pkgs.	26.00	11	50	25	0
38	Côte de Boie	18	1	12 - 75 cl bottles	263.50	17	0	15	0
39	Chateau d'Ay	18	1	750 cl per bottle	18.00	69	0	5	0
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.40	123	0	30	0
41	Jack's New England Clam Cho...	19	8	12 - 12 oz cans	9.65	85	0	10	0

Query executed successfully.

MS-SQLSERVER (15.0 RTM) MS)enmq (53) NORTHWND 00:00:00 79 rows

SQLQuery2.sql - MS-SQLSERVER.NORTHWND (MS)enmq (53) - Microsoft SQL Server Management Studio

Object Explorer: MSSQLSERVER (SQL Server 15.0.2000) > Databases > NORTHWND > Tables > dbo.Products

Results: Messages

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
39	Chateau d'Ay	18	1	750 cl per bottle	18.00	69	0	5	0
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.40	123	0	30	0
41	Jack's New England Clam Cho...	19	8	12 - 12 oz cans	9.65	85	0	10	0
42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14.00	26	0	0	1
43	Ispah Coffee	20	1	16 - 500 g tins	46.00	17	10	25	0
44	Gula Melacca	20	2	20 - 2 kg bags	19.45	27	0	15	0
45	Rössle sild	21	8	1 kg pkg.	9.50	5	70	15	0
46	Spegefeld	21	8	4 - 450 g glasses	12.00	95	0	0	0
47	Zaenen koeken	22	3	10 - 4 oz boxes	9.50	36	0	0	0
48	Chocolade	22	3	10 pkgs	12.75	15	70	25	0
49	Maidulau	23	3	24 - 50 g pkgs.	20.00	10	60	15	0
50	Valkonen suklaa	23	3	12 - 100 g bars	16.25	65	0	30	0
51	Margrop Dried Apples	24	7	50 - 300 g pkgs.	53.00	20	0	10	0
52	Pila Miu	24	5	16 - 2 kg boxes	7.00	38	0	25	0
53	Petit Pâtisier	24	6	48 pieces	32.80	0	0	0	1
54	Toutaine	25	6	16 pies	7.45	21	0	10	0
55	Pâté chèvres	25	6	24 boxes x 2 pies	24.00	115	0	20	0
56	Gnocchi di nonna Alice	26	5	24 - 250 g pkgs.	38.00	21	10	30	0
57	Raviole Angole	26	5	24 - 250 g pkgs.	19.50	36	0	20	0
58	Escaragots de Bourgogne	27	8	24 pieces	13.25	62	0	20	0
59	Raclette Courdaivault	28	4	5 kg pkg.	55.00	79	0	0	0
60	Camembert Fromst	28	4	15 - 300 g mounds	34.00	19	0	0	0
61	Snap d'Italie	29	2	24 - 500 ml bottles	28.50	113	0	25	0
62	Tarte au sucre	29	3	48 pies	49.30	17	0	0	0
63	Veggie spread	7	2	15 - 625 g jars	43.90	24	0	5	0
64	Wimmers gute Semmelknödel	12	5	20 bags x 4 pieces	33.25	22	80	30	0
65	Louisiana Hot Hot Pepper Sa...	2	2	32 - 8 oz bottles	21.05	76	0	0	0
66	Louisiana Hot Spiced Ochs	2	2	24 - 8 oz jars	17.00	4	100	20	0
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.00	52	0	10	0
68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12.50	6	10	15	0
69	Gulbrandellakost	15	4	10 kg pkg.	36.00	26	0	15	0
70	Outback Lager	7	1	24 - 355 ml bottles	15.00	15	10	30	0
71	Pilsener	15	4	10 - 500 g pkgs.	21.50	26	0	0	0
72	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34.80	14	0	0	0
73	Röd Kaviar	17	8	24 - 150 g jars	15.00	101	0	5	0
74	Longlife Tofu	4	7	5 kg pkg.	10.00	4	20	5	0
75	Philly Phanitza Hotsterber	12	1	24 - 0.5 l bottles	7.75	125	0	25	0
76	Lakkalikööri	23	1	500 ml	18.00	57	0	20	0
77	Original Frankfurt grüne Soße	12	2	12 bottles	13.00	32	0	15	0
78	Stamman Royale	NULL	NULL	NULL	0.00	0	0	0	0
80	Touffaine Emperor	NULL	NULL	NULL	0.00	0	0	0	0

Query executed successfully.

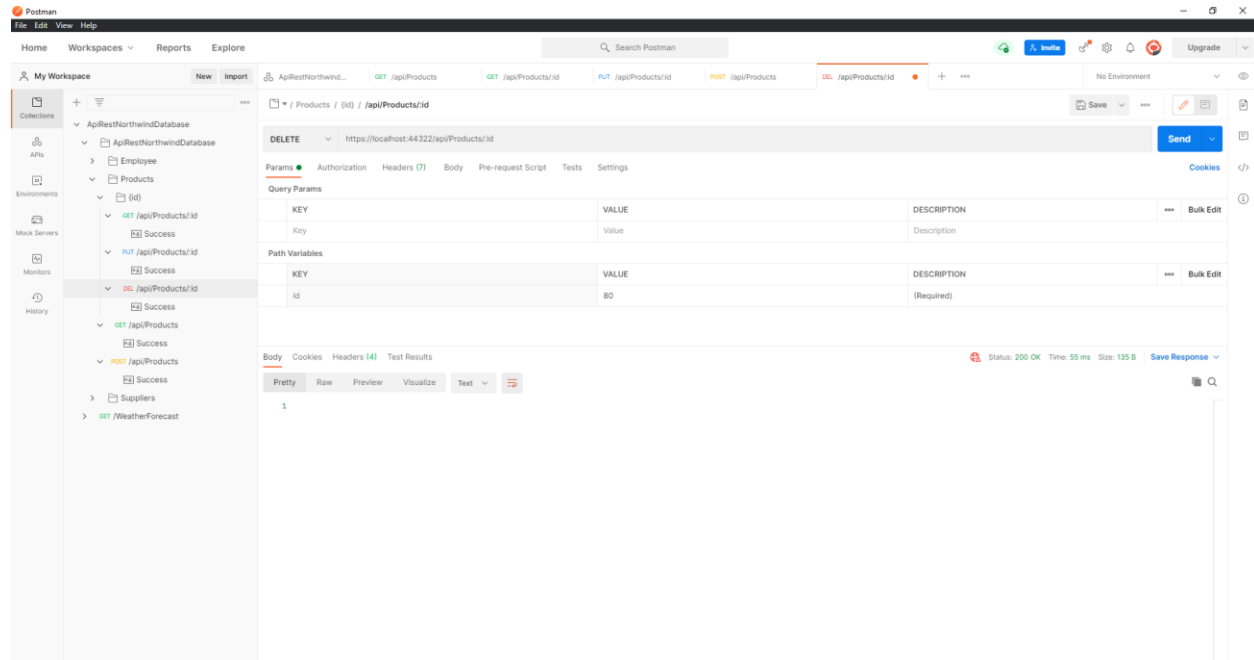
MS-SQLSERVER (15.0 RTM) MS)enmq (53) NORTHWND 00:00:00 79 rows

Y así obtenemos todos los datos de la tabla de products.

DELETE ID

URL: <https://localhost:44322/api/Products/:id>

Este endpoint solo pide el ID y nada que llenar en el body, vamos a poner el id 80 del producto Toughness Emperor.



The screenshot shows the 'Products' table in the 'NORTHWIND' database. The table has 80 rows and 13 columns: ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, and Discontinued. The data includes various products like 'Chateau d'Aux', 'Boston Crab Meat', 'Jack's New England Clam Chowder', 'Singaporean Hokkien Fried Mee', 'Gula Melacca', 'Rogede sild', 'Spegesild', 'Zaaron kookien', 'Chocolade', 'Maailku', 'Valomonen suklaa', 'Margsup Dried Apples', 'Rip Ma', 'Pethi Pasties', 'Touffaine', 'Pala chivos', 'Gnocchi di nonna Alice', 'Ravioi Angelo', 'Escargots de Bourgogne', 'Raclette Courdaulat', 'Carmenbet Pienot', 'Snop d'able', 'Tarte au sucre', 'Vogel-spread', 'Wimmers gute Semmelknödel', 'Louisiana Hot Hot Pepper Sauce', 'Louisiana Hot Spiced Omelette', 'Laughing Lumberjack Lager', 'Scottish Longbreads', 'Gulbrandsenbakkost', 'Outback Lager', 'Polmosmykist', 'Mozzarella di Giovanni', 'Röd Kaviar', 'Longlife Tofu', 'Pilsbrower Klotzebeer', 'Lakaleiköl', 'Original Frankfurt grüne Soße', 'Stammen Pilske', and 'Toughness Emperor'.

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
39	Chateau d'Aux	18	1	750 cc per bottle	18.50	69	0	5	0
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.40	123	0	30	0
41	Jack's New England Clam Chowder	19	8	12 - 12 oz cans	9.65	85	0	10	0
42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14.00	26	0	0	1
43	Ispah Coffee	20	1	16 - 500 g tins	46.00	17	10	25	0
44	Gula Melacca	20	2	20 - 2 kg bags	19.45	27	0	15	0
45	Rogede sild	21	8	1k pkg	9.50	5	70	15	0
46	Spegesild	21	8	4 - 450 g glasses	12.00	95	0	0	0
47	Zaaron kookien	22	3	10 - 4 oz boxes	9.50	36	0	0	0
48	Chocolade	22	3	10 pkgs	12.75	15	70	25	0
49	Maailku	23	3	24 - 50 g pkgs.	20.00	10	60	15	0
50	Valomonen suklaa	23	3	12 - 100 g bars	16.25	65	0	30	0
51	Margsup Dried Apples	24	7	50 - 300 g pkgs.	53.00	20	0	10	0
52	Rip Ma	24	5	16 - 2 kg boxes	7.00	38	0	25	0
53	Pethi Pasties	24	6	40 pieces	32.80	0	0	0	1
54	Touffaine	25	6	16 pies	7.45	21	0	10	0
55	Pala chivos	25	6	24 boxes x 2 pies	24.00	115	0	20	0
56	Gnocchi di nonna Alice	26	5	24 - 250 g pkgs.	38.00	21	10	30	0
57	Ravioi Angelo	26	5	24 - 250 g pkgs.	19.50	36	0	20	0
58	Escargots de Bourgogne	27	8	24 pieces	13.25	62	0	20	0
59	Raclette Courdaulat	28	4	5 kg pkg	55.00	79	0	0	0
60	Carmenbet Pienot	28	4	15 - 300 g rounds	34.00	19	0	0	0
61	Snop d'able	29	2	24 - 500 ml bottles	28.50	113	0	25	0
62	Tarte au sucre	29	3	40 pies	49.30	17	0	0	0
63	Vogel-spread	7	2	15 - 625 g jars	43.90	24	0	5	0
64	Wimmers gute Semmelknödel	12	5	20 bags x 4 pieces	33.25	22	80	30	0
65	Louisiana Hot Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05	76	0	0	0
66	Louisiana Hot Spiced Omelette	2	2	24 - 8 oz jars	17.00	4	100	20	0
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.00	52	0	10	0
68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12.50	6	10	15	0
69	Gulbrandsenbakkost	15	4	10 kg pkg	36.00	26	0	15	0
70	Outback Lager	7	1	24 - 355 ml bottles	15.00	15	10	30	0
71	Polmosmykist	15	4	10 - 500 g pkgs.	21.50	26	0	0	0
72	Mozzarella di Giovanni	14	4	24 - 200 g pkgs.	34.80	14	0	0	0
73	Röd Kaviar	17	8	24 - 150 g jars	15.00	101	0	5	0
74	Longlife Tofu	4	7	5 kg pkg	10.00	4	20	5	0
75	Pilsbrower Klotzebeer	12	1	24 - 0.5 l bottles	7.75	125	0	25	0
76	Lakaleiköl	23	1	500 ml	18.00	57	0	20	0
77	Original Frankfurt grüne Soße	12	2	12 boxes	13.00	32	0	15	0
78	Stammen Pilske	NULL	NULL	NULL	0.00	0	0	0	0
79	Toughness Emperor	NULL	NULL	NULL	0.00	0	0	0	0

SQLQuery2.sql - MSSQLSERVER.NORTHWND (MSDieniq (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Object Explorer

Connect to MSSQLSERVER (SQL Server 15.0.2000)

System Databases
Database Snapshots
NORTHWND
Database Diagrams
Tables
System Tables
File Tables
External Tables
Graph Tables
dbo.Categories
dbo.CustomerCustomer
dbo.CustomerDemographics
dbo.Customers
dbo.Employees
dbo.EmployeeTerritories
dbo.Order Details
dbo.Orders
dbo.Products
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Region
dbo.Shippers
dbo.Suppliers
dbo.Territories
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler

Results Messages

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
38	Côte de Blaye	18	1	12 - 75 cl bottles	263.50	17	0	15	0
39	Chateau d'Ay	18	1	750 cc per bottle	18.00	69	0	5	0
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.40	123	0	30	0
41	Jack's New England Clam Chowder	19	8	12 - 12 oz cans	9.65	85	0	10	0
42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14.00	26	0	0	1
43	Ispah Coffee	20	1	16 - 500 g tins	46.00	17	10	25	0
44	Gula Malacca	20	2	20 - 2 kg bags	19.45	27	0	15	0
45	Pigeon's milk	21	8	1/4 pkgs.	9.50	5	70	15	0
46	Soupstock	21	8	4 - 450 g glasses	12.00	95	0	0	0
47	Zaansse Koeken	22	3	10 - 4 oz boxes	9.50	36	0	0	0
48	Chocolade	22	3	10 pkgs.	12.75	15	70	25	0
49	Manila	23	3	24 - 50 g pkgs.	20.00	10	60	15	0
50	Vakuumverpakte	23	3	12 - 100 g jars	16.25	65	0	30	0
51	Margnap Dried Apples	24	7	50 - 300 g pkgs.	53.00	20	0	10	0
52	Flo Mix	24	5	16 - 2 kg boxes	7.00	38	0	25	0
53	Petit Pasties	24	6	48 pieces	32.80	0	0	0	1
54	Tourtiere	25	6	16 pies	7.45	21	0	10	0
55	Fish chowder	26	6	24 boxes x 2 pies	24.00	115	0	20	0
56	Gnocchi di nonna Alice	26	5	24 - 250 g pkgs.	38.00	21	10	30	0
57	Ravioli Angelo	26	5	24 - 250 g pkgs.	19.50	36	0	20	0
58	Escargots de Bourgogne	27	8	24 pieces	13.25	62	0	20	0
59	Raclette Courmoulay	28	4	5 kg pkg.	55.00	79	0	0	0
60	Camembert Fromat	28	4	15 - 300 g rounds	34.00	19	0	0	0
61	Simp d'érable	29	2	24 - 500 ml bottles	28.50	113	0	25	0
62	Tarte au sucre	29	3	48 pies	49.30	17	0	0	0
63	Vierge-repand	7	2	15 - 150 g jars	42.30	24	0	5	0
64	Wimmers gute Semmelknödel	12	5	20 bags x 4 pieces	33.25	22	80	30	0
65	Louisiana Firey Hot Pepper Sauce	2	2	32 - 8 oz bottles	21.05	76	0	0	0
66	Louisiana Hot Spiced Okra	2	2	24 - 8 oz jars	17.00	4	100	20	0
67	Laughing Lumberjack Lager	16	1	24 - 12 oz bottles	14.00	52	0	10	0
68	Scottish Longbreads	8	3	10 boxes x 8 pieces	12.50	6	10	15	0
69	Gudbrandsdalsost	15	4	10 kg pkg.	36.00	26	0	15	0
70	Outback Lager	7	1	24 - 355 ml bottles	15.00	15	10	30	0
71	Pilsener Beer	15	4	10 - 500 g pkgs.	21.50	26	0	0	0
72	Mazzetta di Giovanni	14	4	24 - 200 g pkgs.	34.80	14	0	0	0
73	Rid Kaviar	17	8	24 - 150 g jars	15.00	101	0	5	0
74	Longlife Tofu	4	7	5 kg pkg.	10.00	4	20	5	0
75	Rhinbräu Klotzebeer	12	1	24 - 0.5 l bottles	7.75	125	0	25	0
76	Lakeland	23	1	500 ml	18.00	57	0	20	0
77	Original Frankfurt grüne Soße	12	2	12 boxes	13.00	32	0	15	0
78	Stamnan Royale	NULL	NULL	NULL	0.00	0	0	0	0

Query executed successfully.

MSSQLSERVER (15.0 RTM) MSDieniq (53) NORTHWND 00:00:00 78 rows

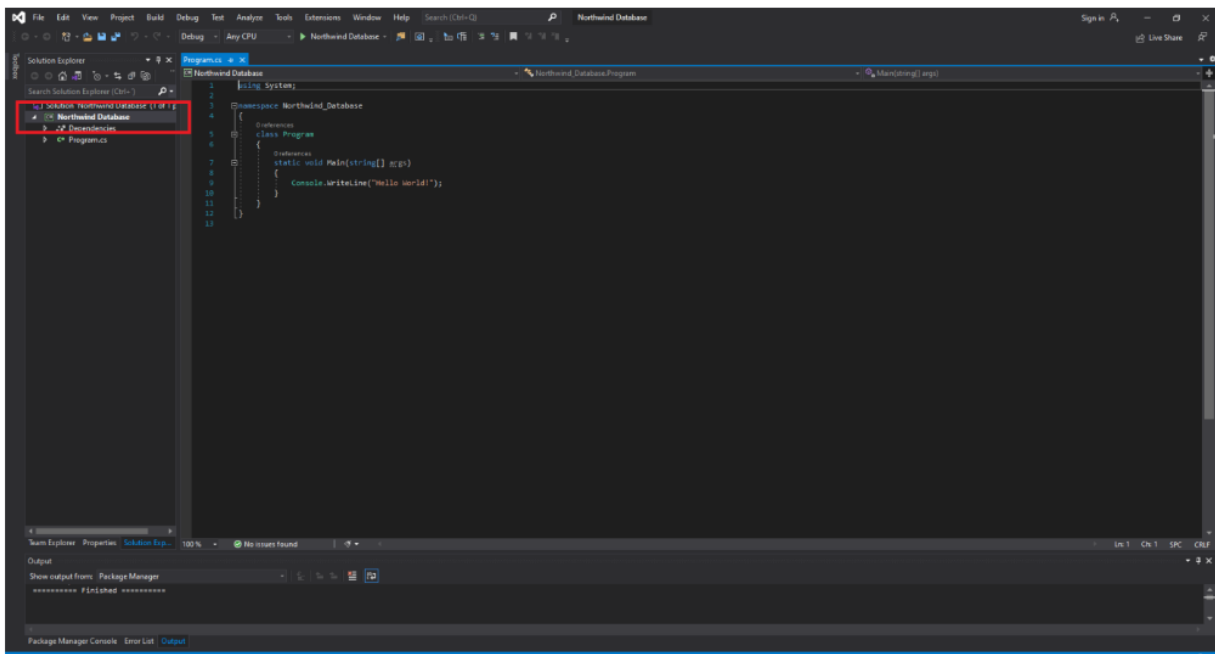
Ready Lin 12 Col 35 Ch 25 R6

Y como podemos ver eliminamos exitosamente el producto Toughness Emperor.

Documentación del desarrollo:

Al empezar el proyecto primero descargamos la base de datos Northwind que se uso para conectarse al back end, y en tanto al back end se creó un proyecto nuevo en Microsoft Visual Studio con el template de Console App en NET 5.

Se ha llamado este proyecto Northwind Database.

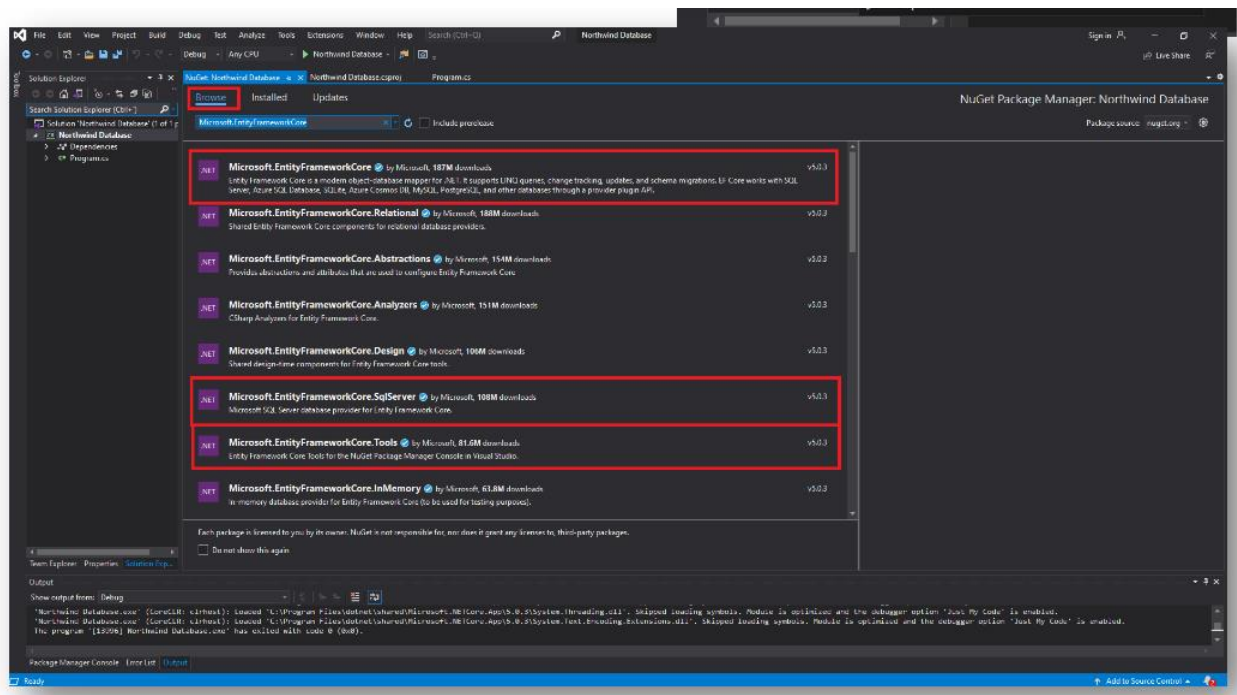


Después de esto se descargo los siguiente paquete NuGet para poder conectarse a la base de datos:

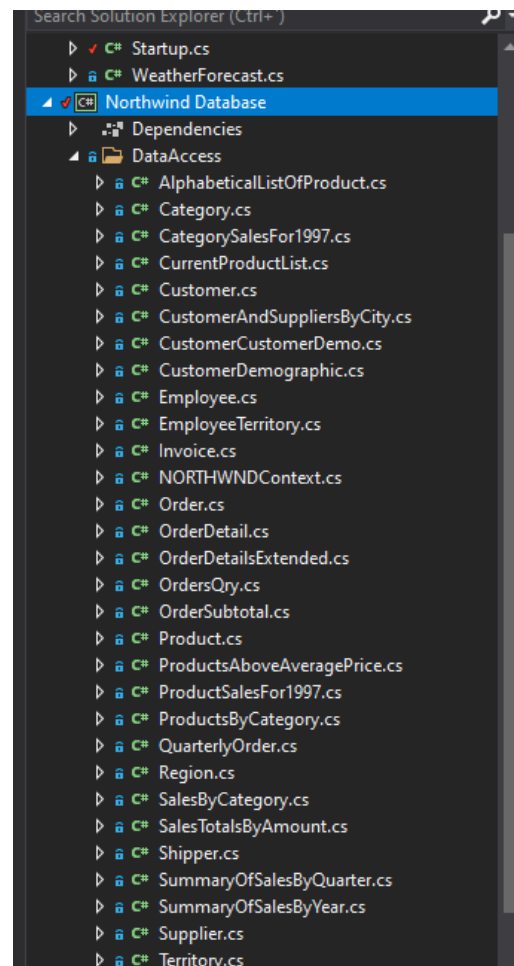
Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

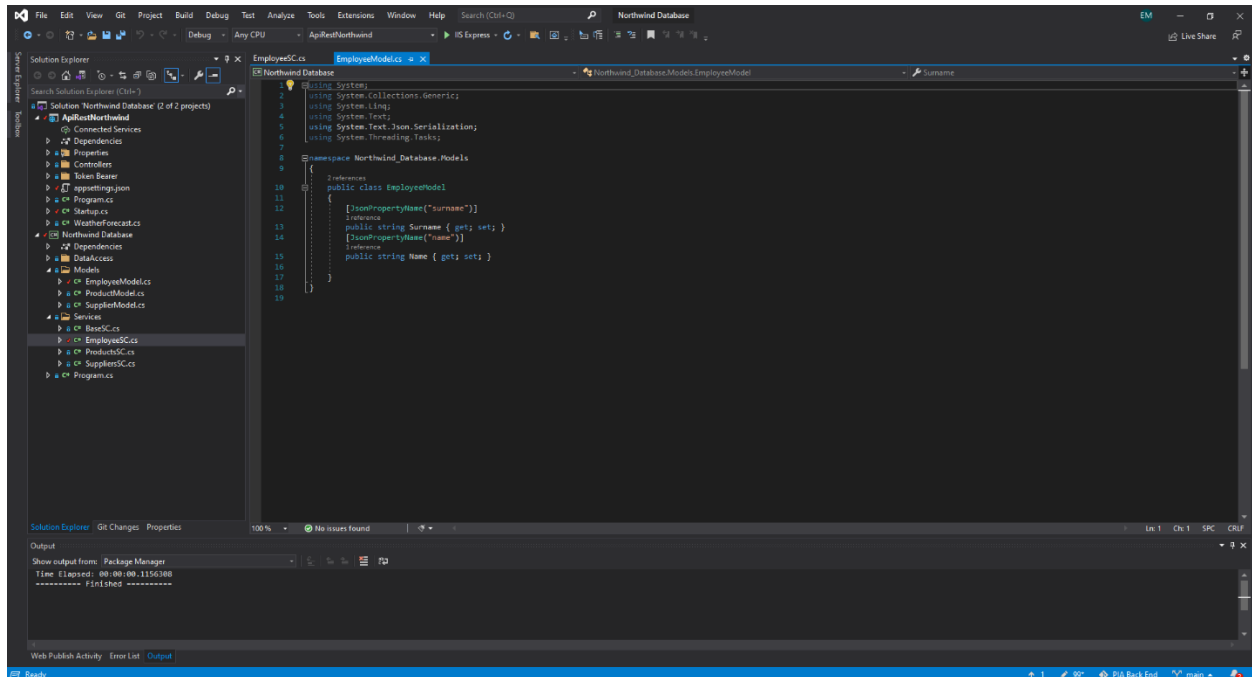


Después de eso utilizamos una comando en la consola para hacer un scaffolding con la base de datos Nortwind y con eso ya tenemos la conexión y toda una carpeta con el data context y el data access para cada tabla de la base de datos a usar.

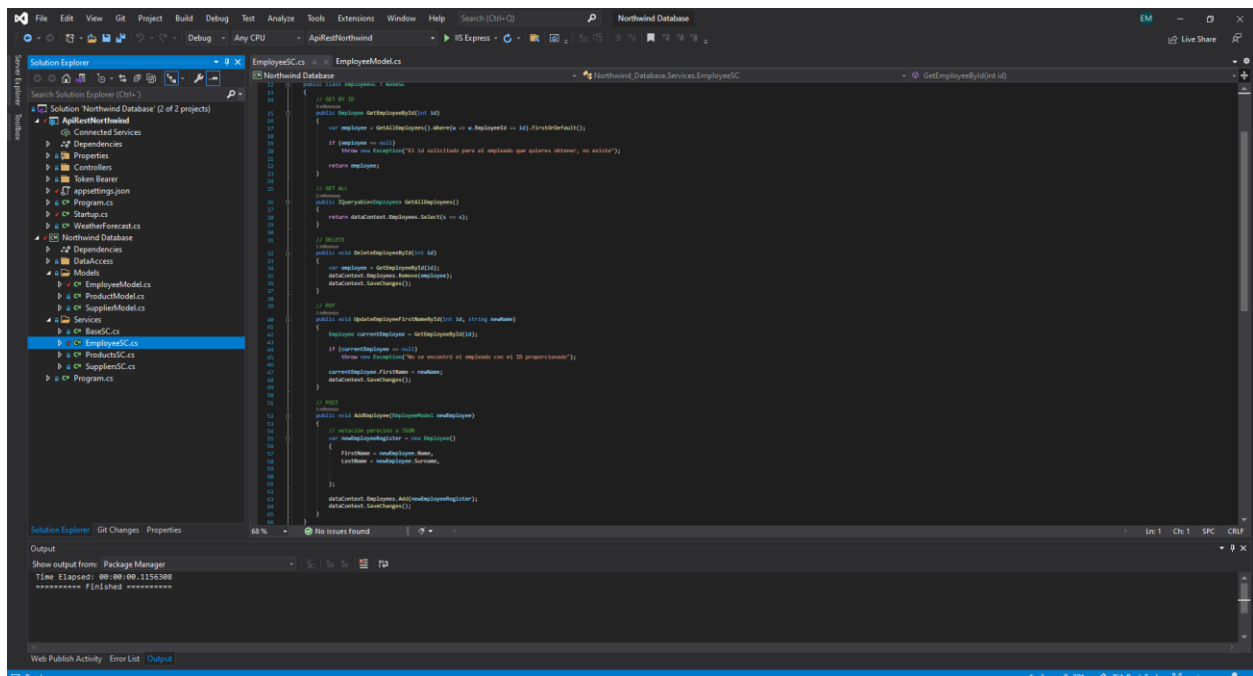


Después de esto se creo una carpeta de modelos y otra de servicios que se van a usar para el back end y que es lo que podemos modificar en él.

En el modelo de Employee por ejemplo tiene todos los datos que queremos sacar de este para poder usarlos en el back end, en este caso solo el nombre y el apellido.



En un servicio como el de Employee tenemos los verbos como GET, POST, DELETE y PUT que se usaran para el back end cuando se conecten a una api rest. En esta clase esta le código para poder correr estos verbos aunque no directamente desde Postman o Swagger, antes de eso hay que hacer una API REST.



Si checamos en una clase de servicio más a detalle podemos ver cómo funcionan cada verbo empezando con GET BY ID que por medio de un ID se busca en la base de datos el empleado.

```
// GET BY ID
3 references
public Employee GetEmployeeById(int id)
{
    var employee = GetAllEmployees().Where(w => w.EmployeeId == id).FirstOrDefault();

    if (employee == null)
        throw new Exception("El id solicitado para el empleado que quieres obtener, no existe");

    return employee;
}
```

En caso de que no se encuentre se le notifica al usuario que no existe y si se encuentra se manda el empleado con sus datos.

Con GET ALL simplemente se hace lo mismo que con GET BY ID pero no ponemos una cláusula where.

```
// GET ALL
2 references
public IQueryable<Employee> GetAllEmployees()
{
    return dbContext.Employees.Select(s => s);
}
```

Con DELETE usamos el método de GET BY ID para encontrar el empleado por el ID y lo removemos de la base de datos guardando los cambios.

```
// DELETE
1 reference
public void DeleteEmployeeById(int id)
{
    var employee = GetEmployeeById(id);
    dbContext.Employees.Remove(employee);
    dbContext.SaveChanges();
}
```

Con PUT usamos el método GET BY ID para buscar el empleado a modificar y con los datos nuevos modificarlo, en este caso cambiar el nombre. Si no hay empleado con ese ID, se le notifica al usuario.

```
// PUT
1 reference
public void UpdateEmployeeFirstNameById(int id, string newName)
{
    Employee currentEmployee = GetEmployeeById(id);

    if (currentEmployee == null)
        throw new Exception("No se encontró el empleado con el ID proporcionado");

    currentEmployee.FirstName = newName;
    dbContext.SaveChanges();
}
```

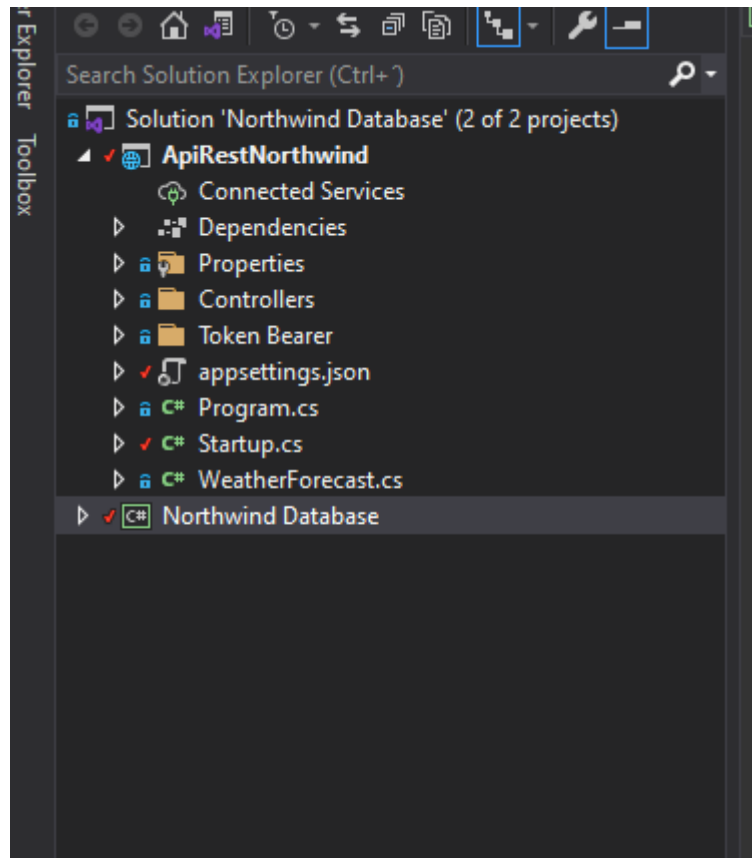
Por ultimo en el verbo POST vamos a crear un nuevo empleado, usando los datos que se solicitaron se crea un nuevo objeto de tiempo employee que tiene todas las características de la base de datos, se agregan los datos y se agrega a la base de datos el nuevo empleado. Se guardan los cambios al final.

```
// POST
1 reference
public void AddEmployee(EmployeeModel newEmployee)
{
    // notación parecida a JSON
    var newEmployeeRegister = new Employee()
    {
        FirstName = newEmployee.Name,
        LastName = newEmployee.Surname,
    };

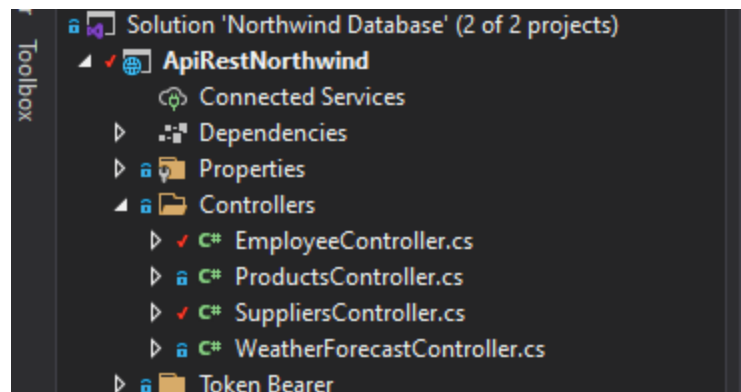
    dataContext.Employees.Add(newEmployeeRegister);
    dataContext.SaveChanges();
}
```


Después de esto se creó una API REST en Visual Studio creando un proyecto de tipo ASP NET Core Web API en C#. Y con esto se creó el proyecto API REST llamado ApiRestNorthwind.

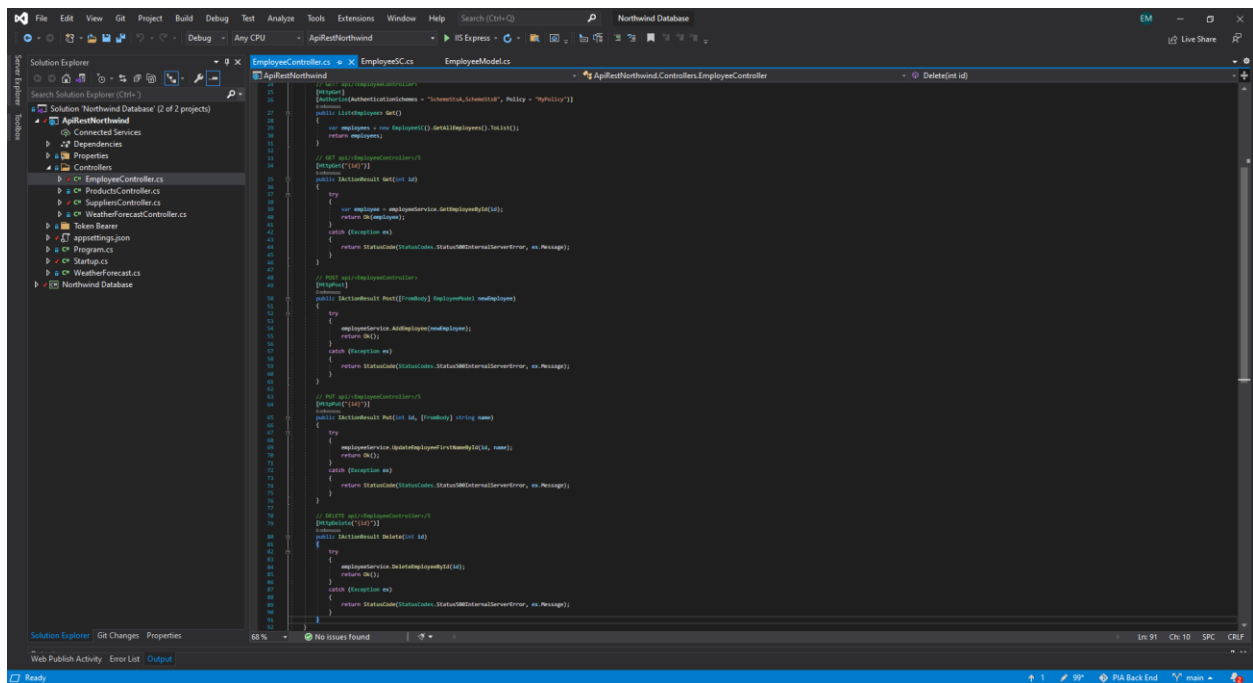
Después de tener la API rest se hizo los controladores.



Se crearon los controladores para las clases Employee, Products y Suppliers.



Cada controlador usa los servicios y modelos que hemos hecho para poder crear los endpoints que se podrán utilizar en POSTMAN y swagger y así poder modificar la base de datos.



Hasta este punto se puede hacer la documentación que se hizo al principio de los endpoints en postman.

Checando los controladores a detalle podemos ver cómo funciona cada verbo, usando las clases de modelos y servicios que se han hecho y explicado tenemos lo siguiente.

```
// GET: api/<ProductsController>
[HttpGet]
0 references
public List<Product> Get()
{
    var product = new ProductsSC().GetAllProducts().ToList();
    return product;
}
```

Con GET ALL usamos el método que hicimos en el servicio de productos, que funciona casi exactamente igual que el que se hizo con employees solo que con datos diferentes, y lo que nos regresa lo hacemos una lista y lo regresamos.

```
// GET api/<ProductsController>/5
[HttpGet("{id}")]
0 references
public IActionResult Get(int id)
{
    try
    {
        var product = productService.GetProductById(id);
        return Ok(product);
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}
```

Con GET BY ID usamos el método del servicio y regresamos exactamente el producto que se pidió por su ID. Si hay un problema se regresa un código de error 500.

```
// POST api/<ProductsController>
[HttpPost]
0 references
public IActionResult Post([FromBody] ProductModel newProduct)
{
    try
    {
        productService.AddProduct(newProduct);
        return Ok();
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}
```

Con POST y todos los demás se van a usar el método correspondiente en su servicio pero con la diferencia que solo se regresa una confirmación de que se hizo el cambio correctamente.

```

// PUT api/<ProductsController>/5
[HttpPut("{id}")]
0 references
public IActionResult Put(int id, [FromBody] string Name)
{
    try
    {
        productService.UpdateProductNameById(id, Name);
        return Ok();
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

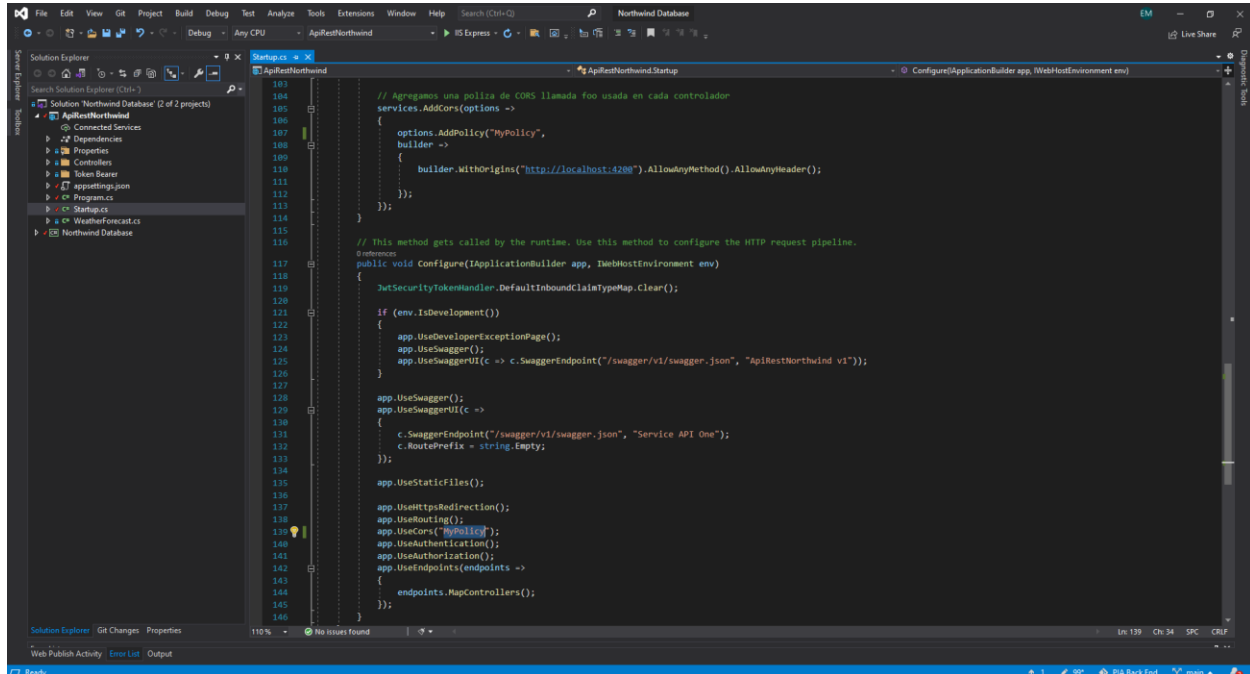
// DELETE api/<ProductsController>/5
[HttpDelete("{id}")]
0 references
public IActionResult Delete(int id)
{
    try
    {
        productService.DeleteProductById(id);
        return Ok();
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}

```

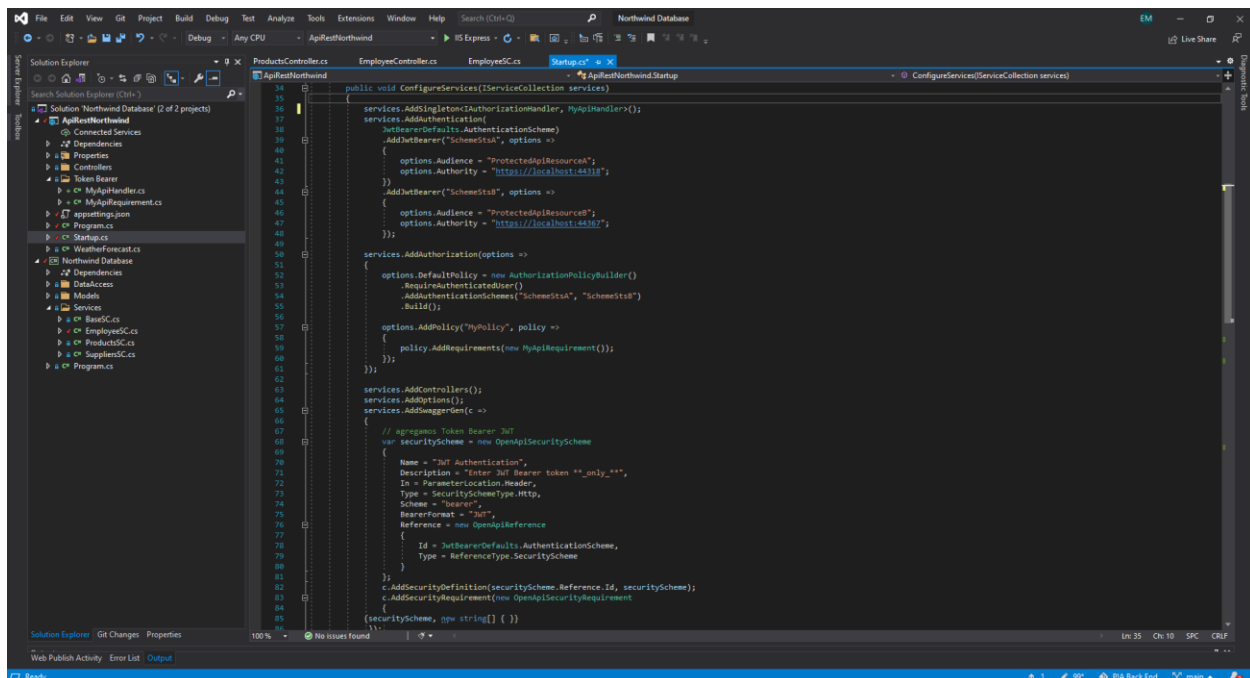
Igual es el caso con PUT y DELETE que simplemente regresan que se confirmó el movimiento.

Después de esto se configuró el CORS en Startup.cs que simplemente no dejará que otro origen que no sea el local host especificado pueda usar el back end, esta póliza se llama MyPolicy, esto se escribe en el método ConfigureServices y por el otro lado en el método configure se agrega el CORS con la póliza que hize. En cada controlador se agregó esta póliza para

que no se pueda modificar ninguna si no es el origen esperado, que idealmente seria el front end especificado.



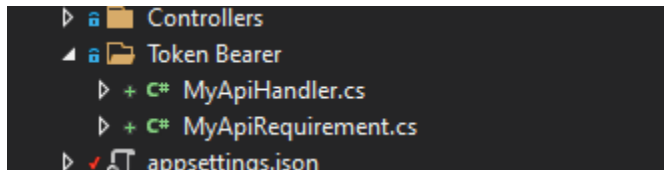
```
103 // Agregamos una politica de CORS llamada Foo usada en cada controlador
104 services.AddCors(options =>
105 {
106     options.AddPolicy("MyPolicy",
107         builder =>
108         {
109             builder.WithOrigins("http://localhost:4200").AllowAnyMethod().AllowAnyHeader();
110         });
111     });
112
113 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
114 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
115 {
116     JwtSecurityTokenHandler.DefaultInboundClaimTypeMap.Clear();
117
118     if (env.IsDevelopment())
119     {
120         app.UseDeveloperExceptionPage();
121         app.UseSwagger();
122         app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "ApiRestNorthwind v1"));
123     }
124
125     app.UseSwagger();
126     app.UseSwaggerUI(c =>
127     {
128         c.SwaggerEndpoint("/swagger/v1/swagger.json", "Service API One");
129         c.RoutePrefix = string.Empty;
130     });
131
132     app.UseStaticFiles();
133
134     app.UseHttpsRedirection();
135     app.UseRouting();
136     app.UseCors("MyPolicy");
137     app.UseAuthentication();
138     app.UseAuthorization();
139     app.UseEndpoints(endpoints =>
140     {
141         endpoints.MapControllers();
142     });
143
144 }
```



```
35 public void ConfigureServices(IServiceCollection services)
36 {
37     services.AddSingleton<AuthorizationHandler, MyAuthHandler>();
38     services.AddAuthentication(
39         JwtBearerDefaults.AuthenticationScheme
40         .AddJwtBearer("SchemeStsA", options =>
41         {
42             options.Audience = "ProtectedApiResourceA";
43             options.Authority = "https://localhost:44318";
44         })
45         .AddJwtBearer("SchemeStsB", options =>
46         {
47             options.Audience = "ProtectedApiResourceB";
48             options.Authority = "https://localhost:44318";
49         })
50     );
51
52     services.AddAuthorization(options =>
53     {
54         options.DefaultPolicy = new AuthorizationPolicyBuilder()
55             .RequireAuthenticatedUser()
56             .AddAuthenticationSchemes("SchemeStsA", "SchemeStsB")
57             .Build();
58
59         options.AddPolicy("MyPolicy", policy =>
60         {
61             policy.AddRequirements(new MyApiRequirement());
62         });
63     });
64
65     services.AddControllers();
66     services.AddOptions();
67     services.AddSwaggerGen(c =>
68     {
69         // agregamos Token Bearer JWT
70         var securityScheme = new OpenApiSecurityScheme
71         {
72             Name = "JWT Authentication",
73             Description = "Enter JWT Bearer token **_only_**",
74             In = ParameterLocation.Header,
75             Type = SecuritySchemeType.Http,
76             Scheme = "bearer",
77             BearerFormat = "JWT",
78             Reference = new OpenApiReference
79             {
80                 Id = JwtBearerDefaults.AuthenticationScheme,
81                 Type = ReferenceType.SecurityScheme
82             }
83         };
84         c.AddSecurityDefinition(securityScheme.Reference.Id, securityScheme);
85         c.AddSecurityRequirement(new OpenApiSecurityRequirement
86         {
87             {securityScheme, new string[] { }}
88         });
89     });
90 }
```

Para agregar el Token Bearer JWT primero modificaremos ConfigureServices agregándoles Singleton, Authentication donde

podemos tomar el Token , Authorization para crear la póliza que se deberá respetar a la hora de modificar un controlador que este controlado por ella, y modificaremos AddSwaggerGen para que nos de la opción de poder poner el Token para modificar lo que queramos.



Crearemos dos clases que se encargaran de hacer funcionar el Token Bearer JWT, primero empezamos con MyApiRequirement donde simplemente vamos a herenciar

IAuthorizationRequirement que es una clase de un paquete que se debe instalar llamado Microsoft.AspNetCore.Authorization, esto manejará los requerimientos para autorizar el uso de los controladores, el requerimiento siendo el Token.

En la siguiente clase llamada MyApiHandler se usa para comprobar que el Token entregado sea válido, y si lo es se manda una comprobación que lo es.

