



INTEGRANTES: CUAYCAL DIEGO, SIMBAÑA OMAR, QUILUMBA LIZBETH

FECHA: SÁBADO, 1 DE FEBRERO DE 2025

TUTOR: MSC. DIEGO TREJO

TEMA: APlicación API REST EN AZURE

Tabla de contenido

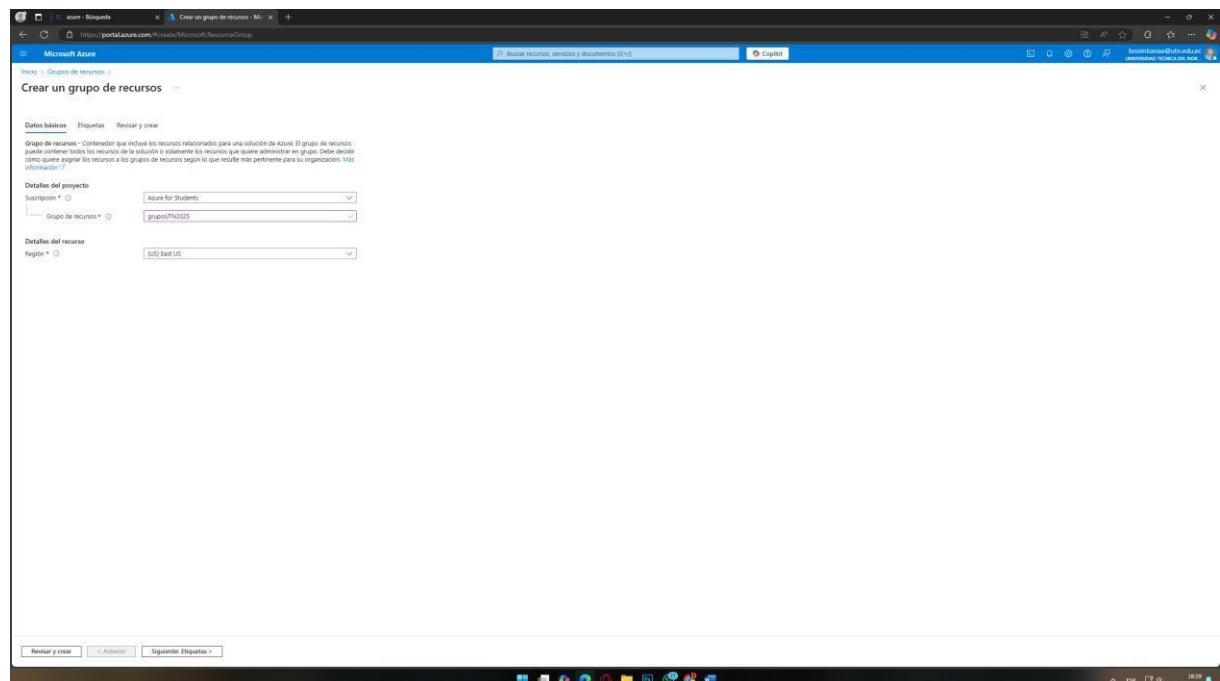
BASE DATOS.....	2
CREACIÓN DE BASE DE DATOS	2
API REST	8
CREACIÓN DE API REST	8
PUBLICACIÓN DE API REST	12
LINK DEL API REST.....	13
DOCUMENTACIÓN CON SWAGGER UI.....	13
APLICACIÓN MVC Y CONSUMO DE API.....	14
CONFIGURACIÓN DE LA BASE DE DATOS EN EL SERVIDOR	14
MÓDULO DE PAQUETES	15
API CONSUMER: CONSUMO DE LA API REST.....	16
PUBLICACIÓN DEL API REST	17
CREACIÓN DEL PROYECTO MVC.....	18
VISUALIZACIÓN INICIAL	24
PUBLICACIÓN MVC.....	28
AUTENTICACION.....	32
LINK DE LA PÁGINA WEB.....	37
CONCLUSIONES	37

APLICACIÓN DE DELIVERY DE COMIDA

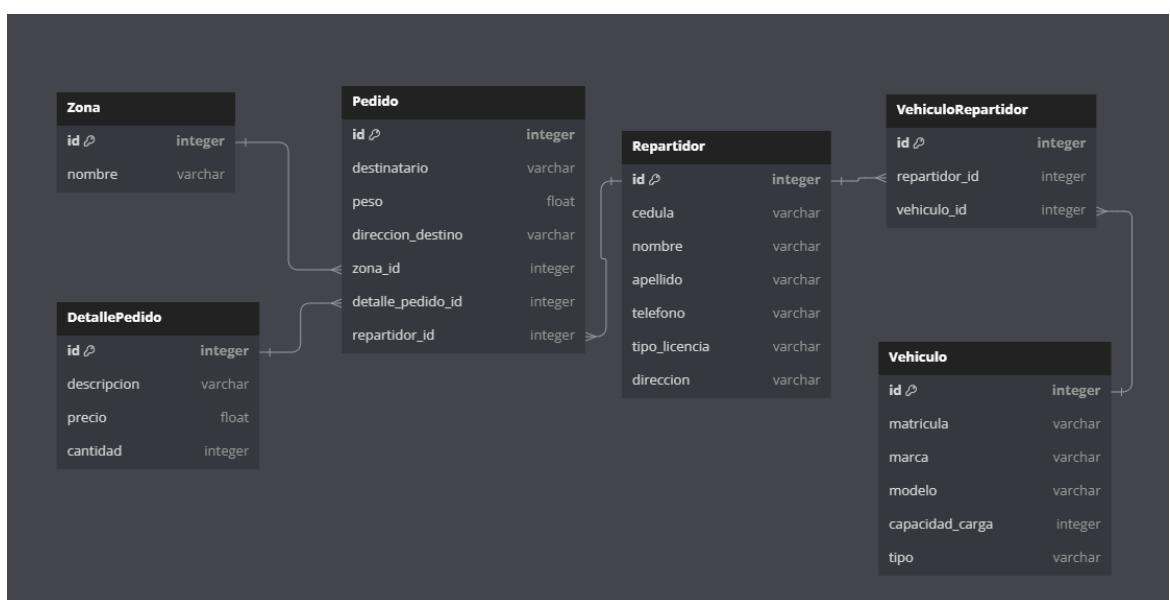
BASE DATOS

CREACIÓN DE BASE DE DATOS

1. En el portal de Microsoft Azure crear el grupo de recursos en donde debe llenar con el nombre del grupo y la región.



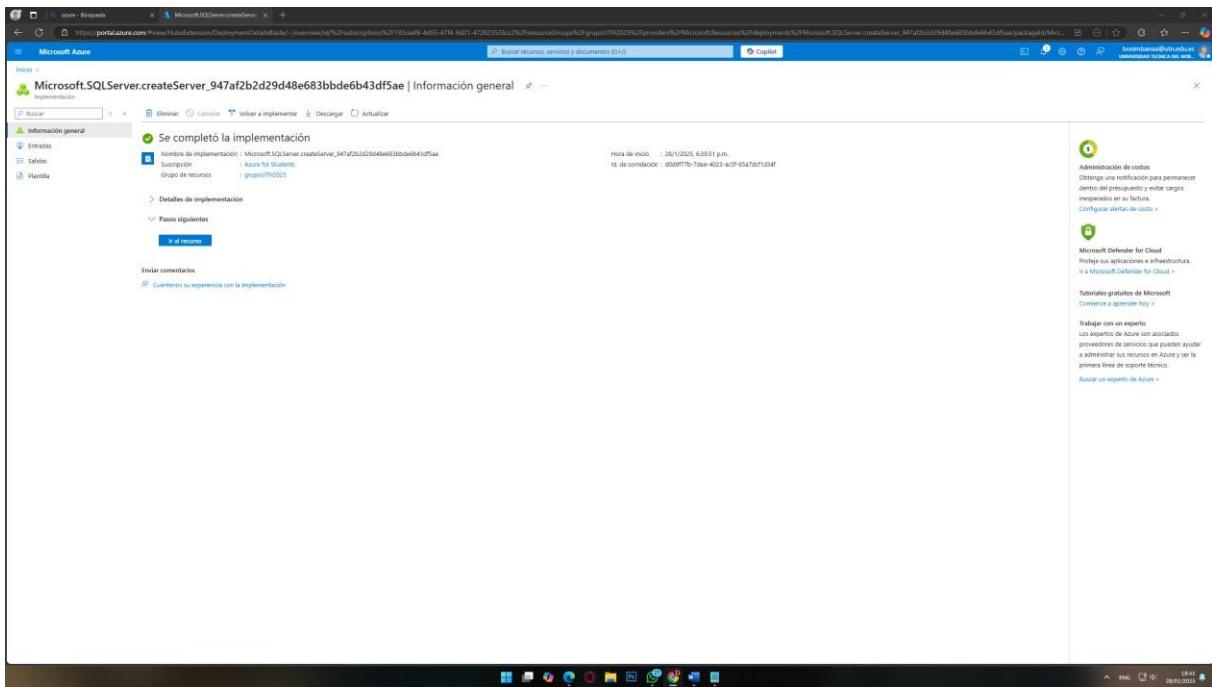
Modelo Entidad – Relación



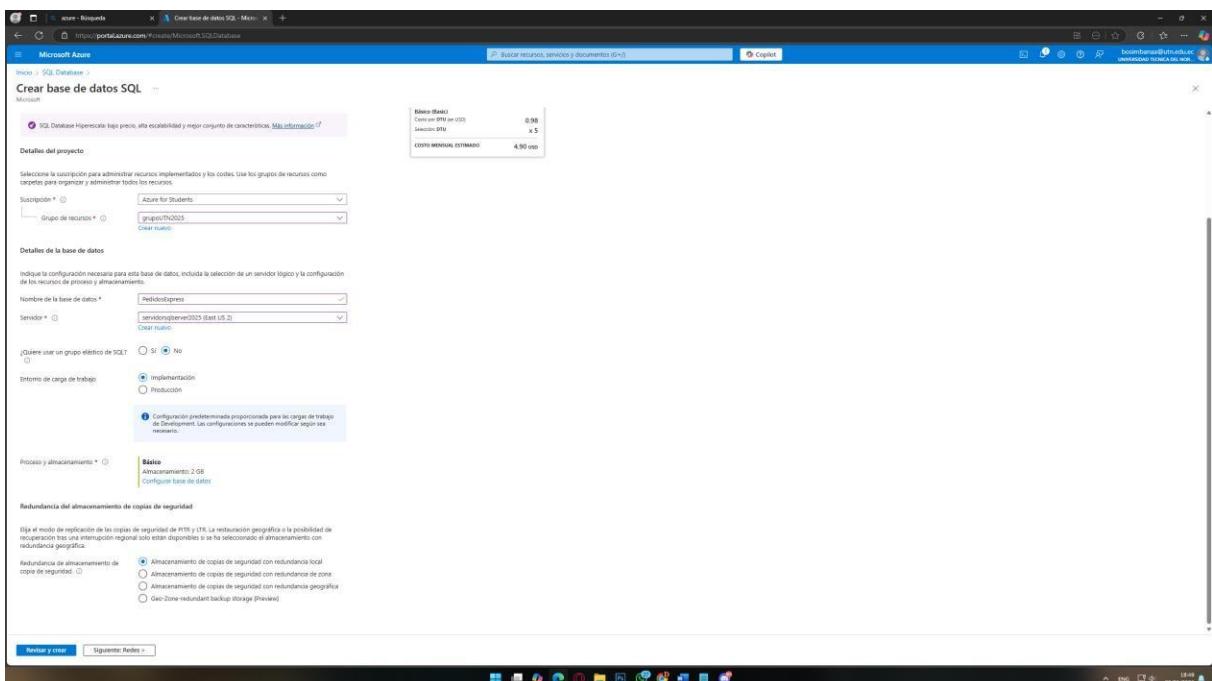
2.- En la sección de grupos de recursos se observa el grupo que se creó anteriormente con el nombre de **grupoUTN2025**.

3.- Crear un servidor SQL Database llenando la información básica que se observa en la imagen.

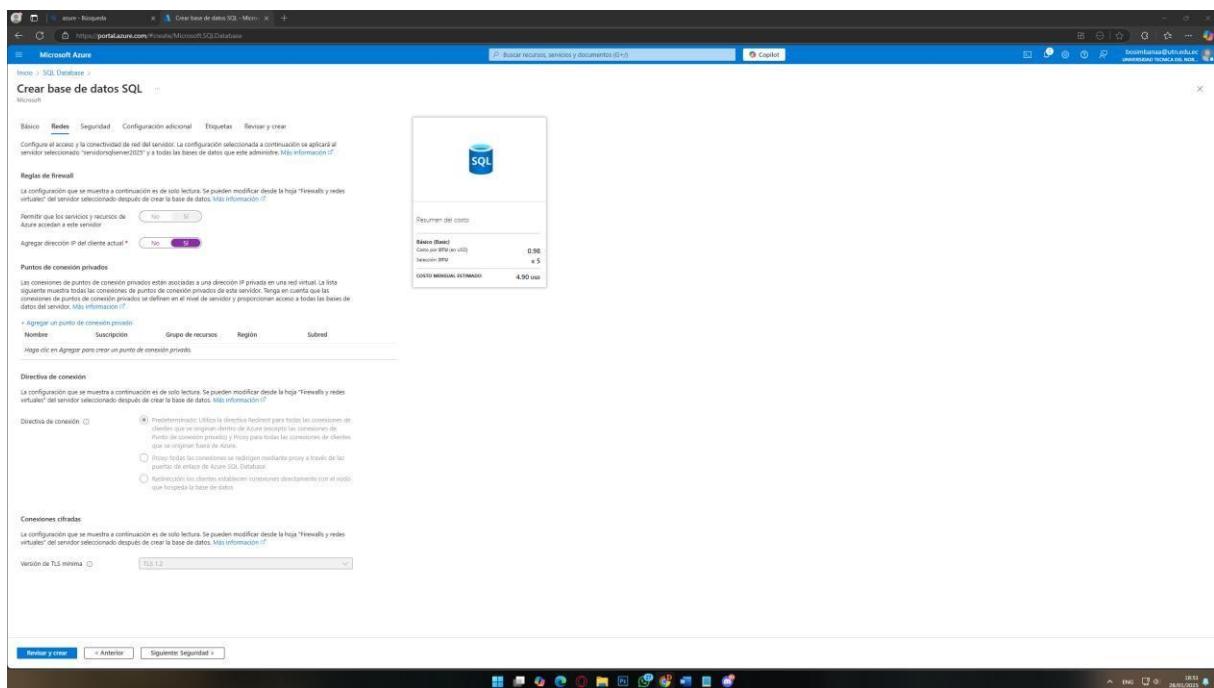
4.- Una vez llenado toda la información necesaria para crear el servidor se puede observar que se completa la implementación.



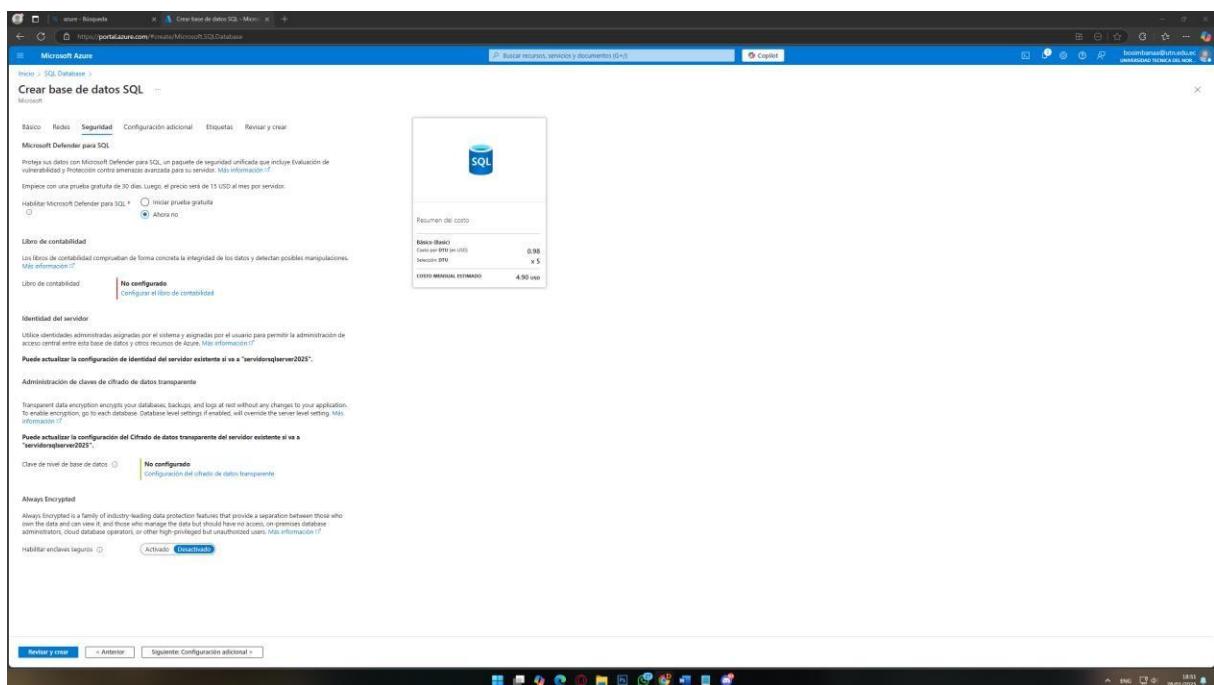
5.- Se procede a crear la base de datos SQL, en la sección básico llenar toda la información necesaria.



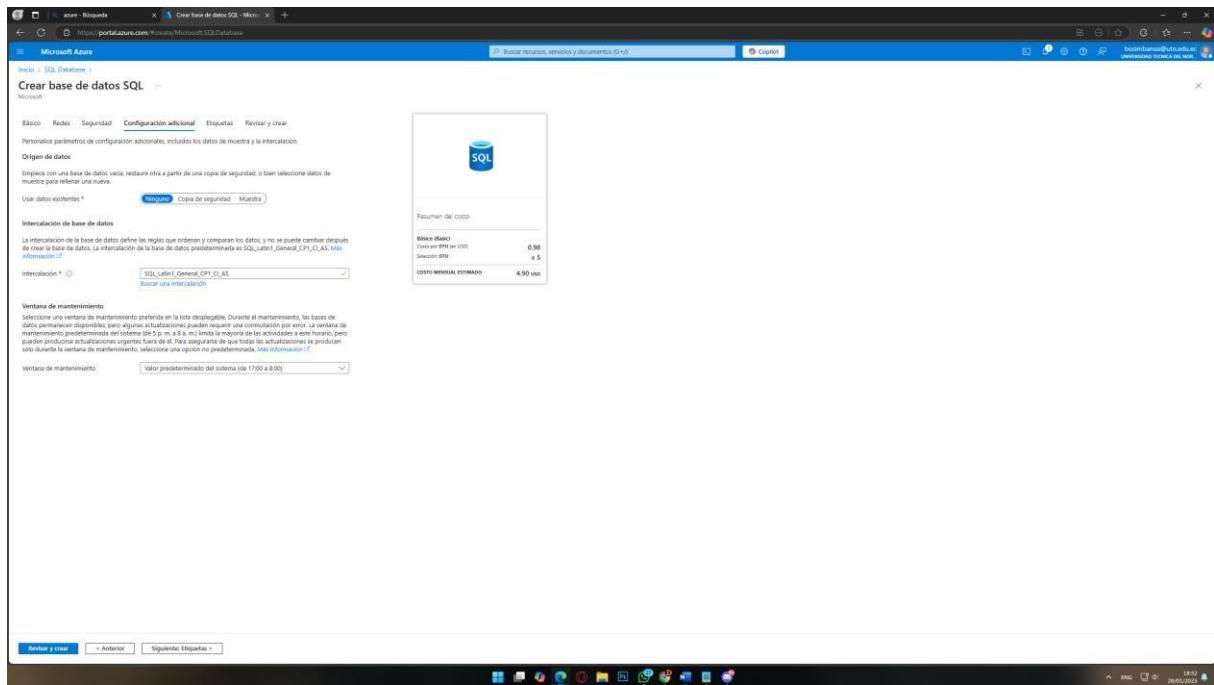
6.- En la sección de redes llenar toda la información que le muestra en la imagen.



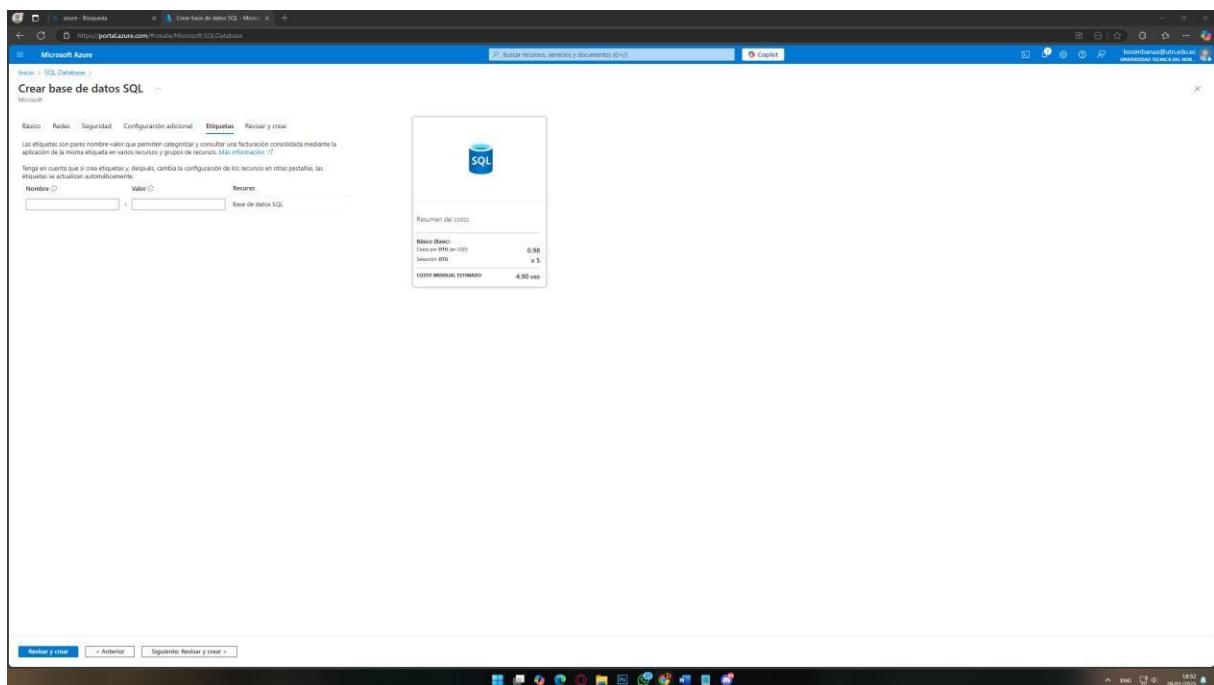
7.- En la sección de seguridad de igual manera habilitar todo lo necesario.



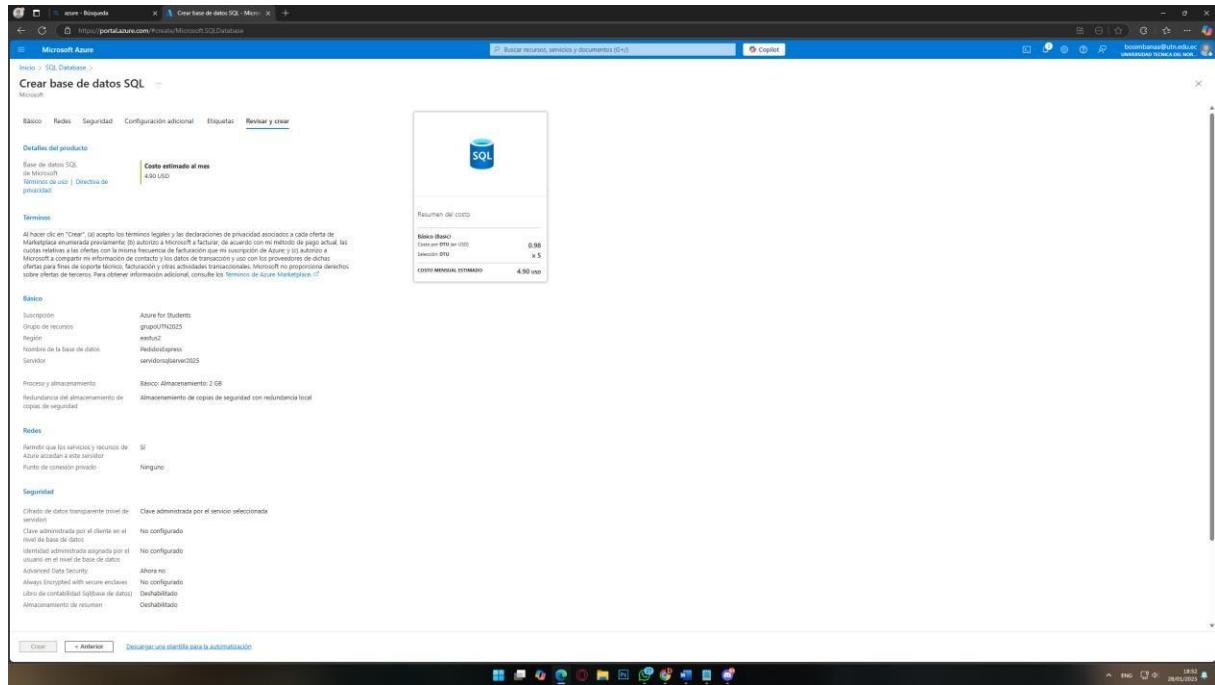
8.- En la sección de configuración adicional llenar toda la información necesaria.



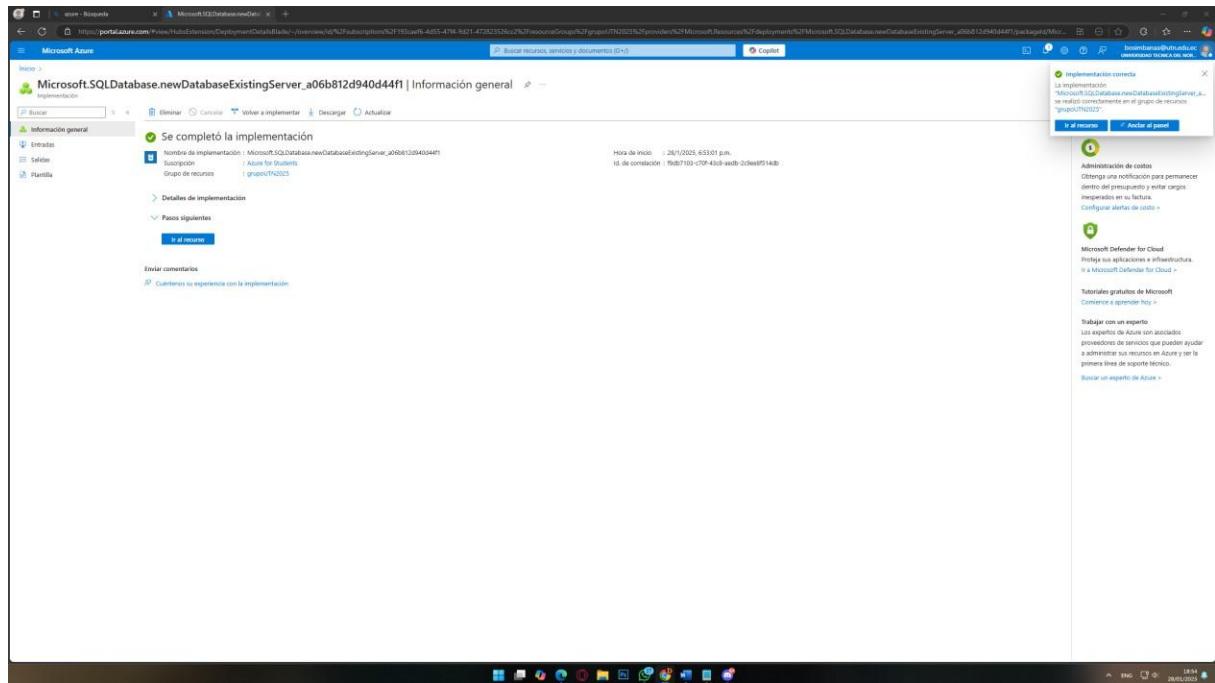
9.- En la sección de etiquetas no seleccionar nada ya que no es necesario.



10.- En la sección de revisar y crear, revisar que toda la configuración que se hizo anteriormente sea la correcta, seguidamente click en crear.



11.- Una vez realizado click en crear se observa que la implementación se ha completado.



11.- Se observa la base de datos que se creó.

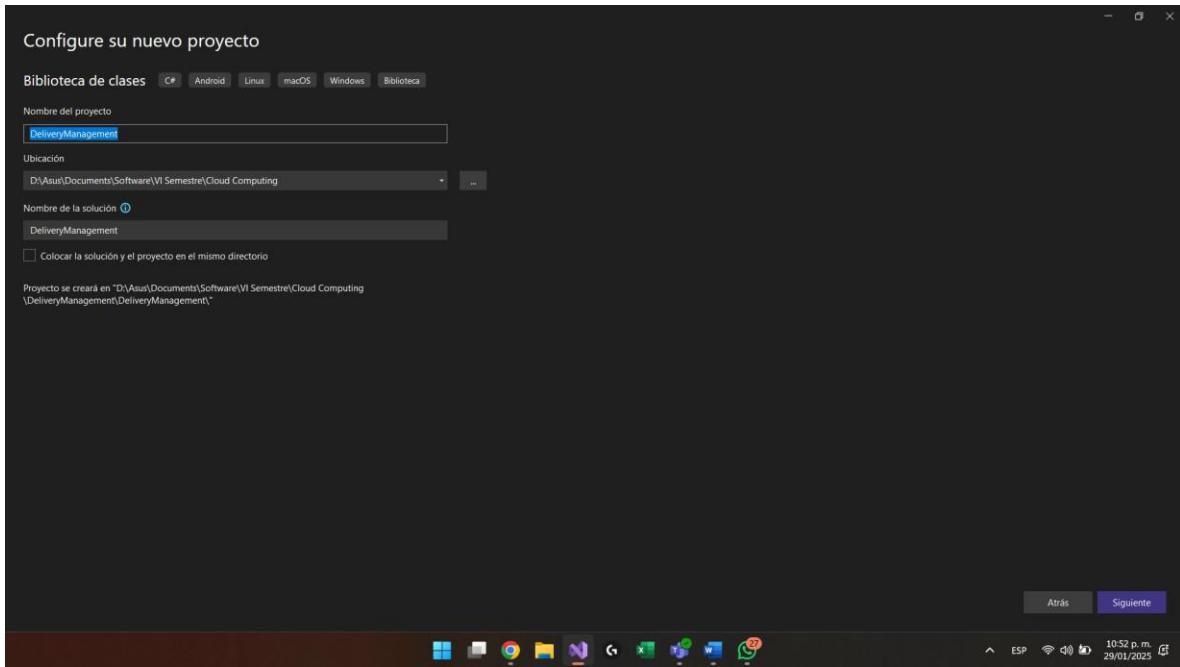
API REST

CREACIÓN DE API REST

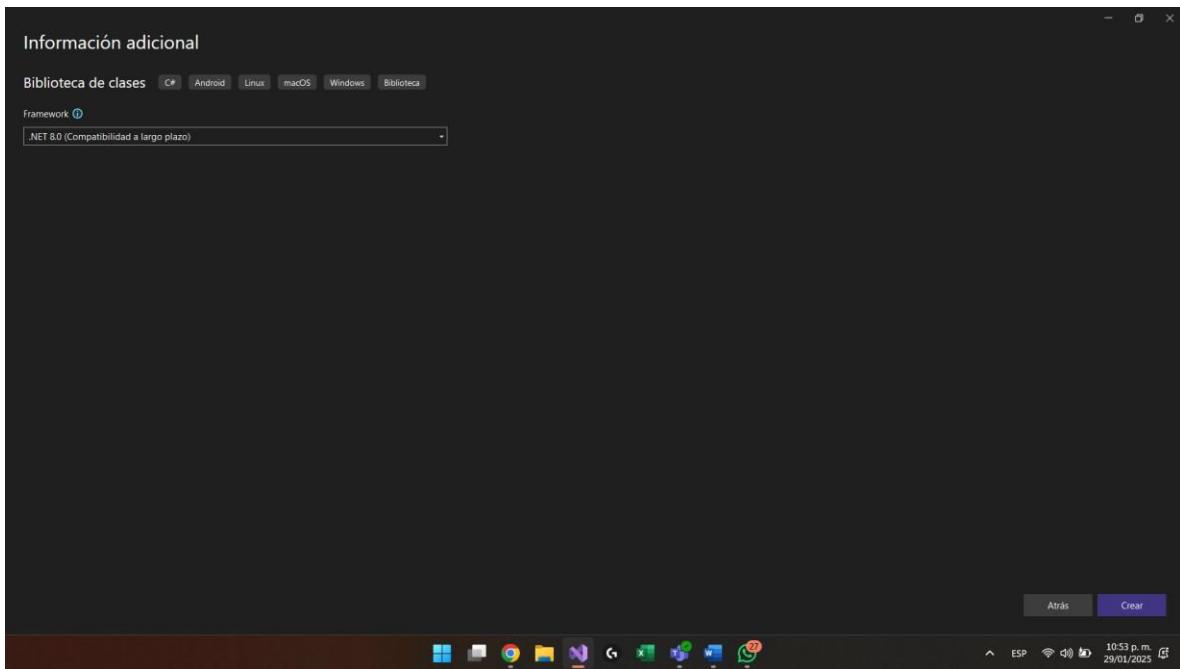
Selección de plantillas para crear un nuevo proyecto en **Visual Studio**. Las opciones disponibles abarcan varios tipos de proyectos comunes para el desarrollo en .NET.

12.- Abrir Visual Studio seleccionar biblioteca de clases, click en siguiente.

13.- Dar un nombre al cual en este es DeliveryManagement.



14.- Seleccionar la versión de Framework en este caso es el .NET 8.0 y crear.



15.- Visualizamos la creación del proyecto y la entidad Restaurante con sus respectivos atributos.

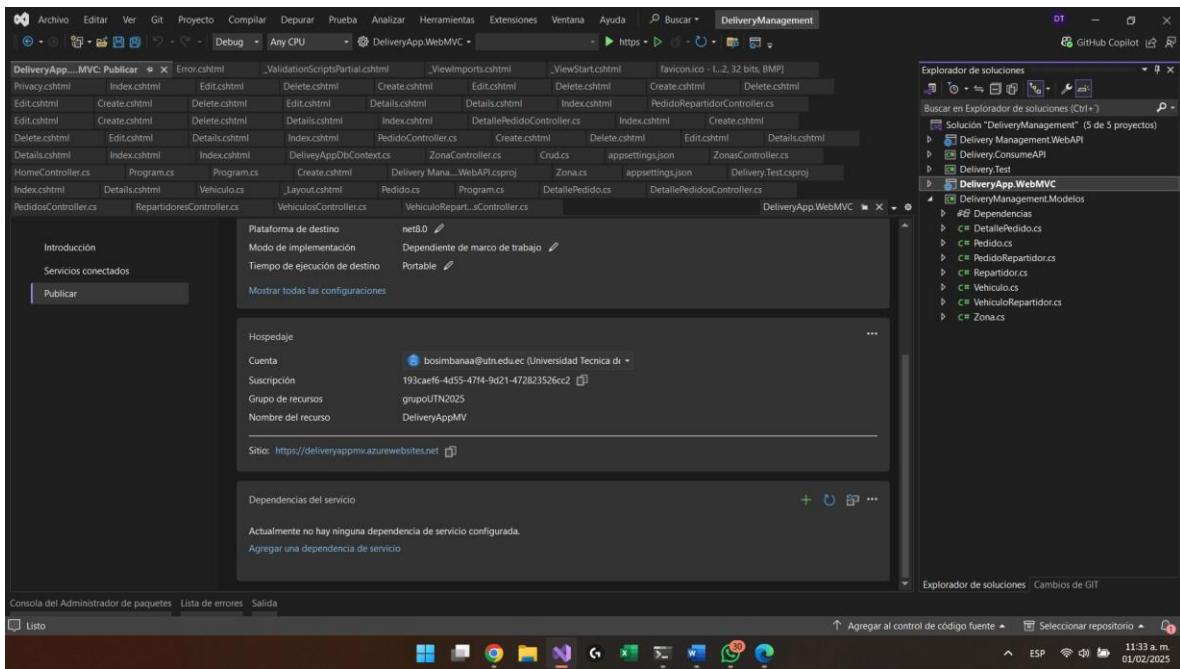
```

using System.ComponentModel.DataAnnotations;

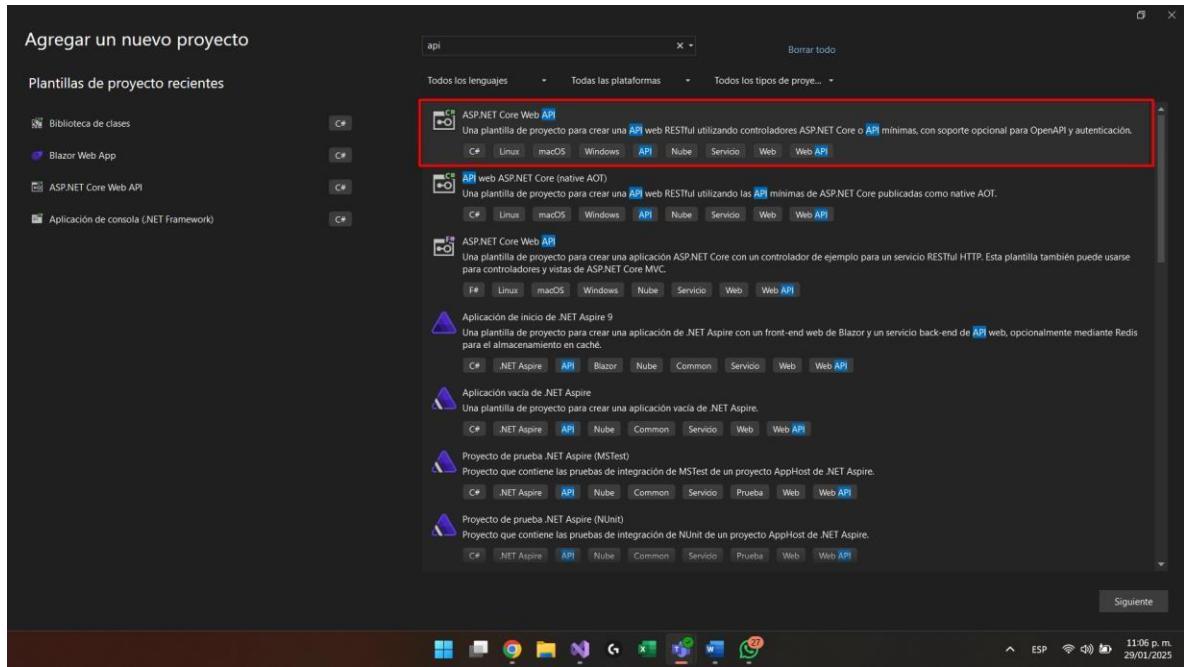
namespace DeliveryManagement
{
    public class Restaurante
    {
        [Key]
        public int RestauranteId { get; set; }
        public string Nombre { get; set; }
        public string Direccion { get; set; }
        public string Telefono { get; set; }
    }
}

```

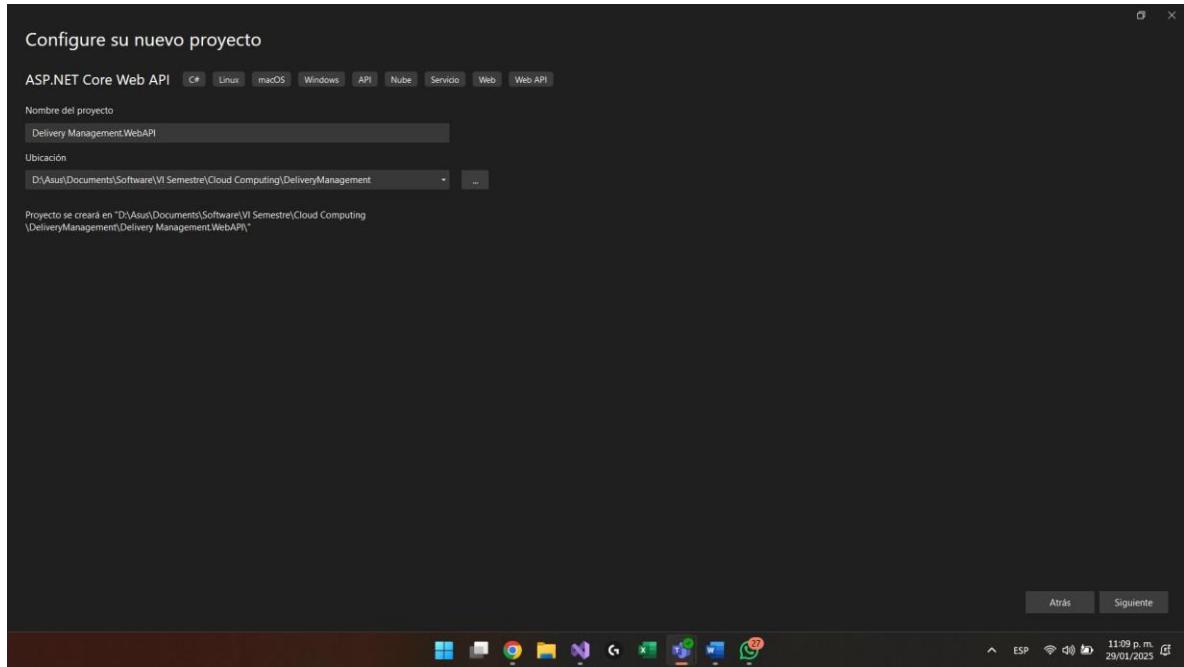
16.- Agregar los todos lo modelos necesarios como se observa en la imagen.



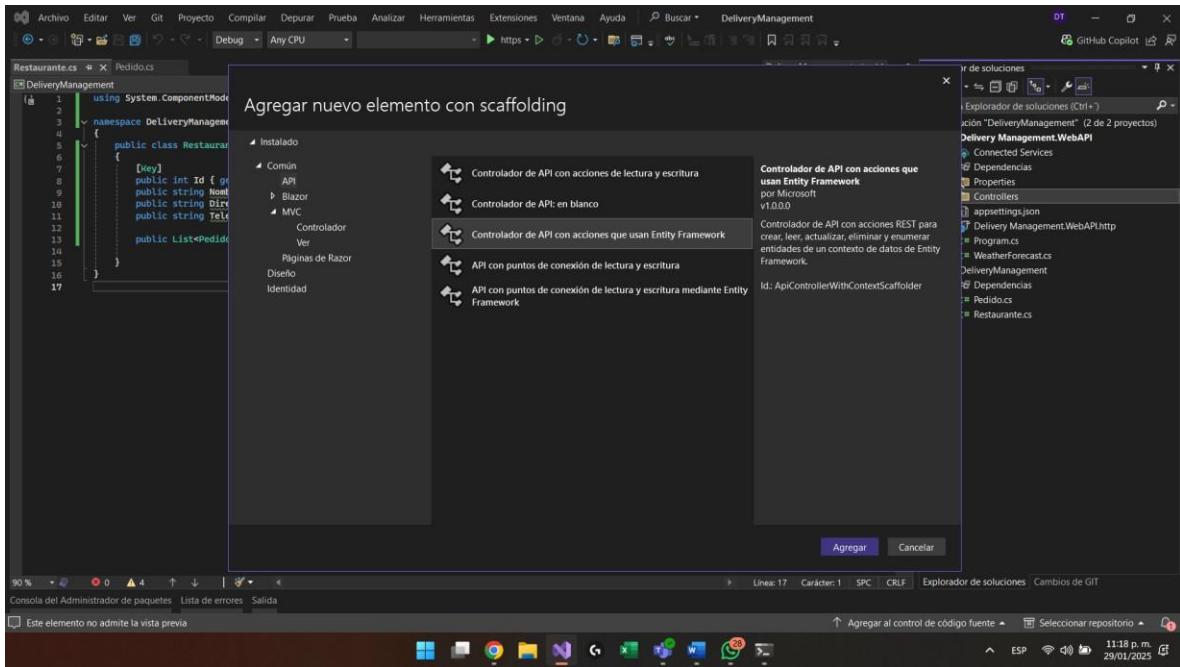
17.- Clic en DeliveryApp.WebMVC y agregar un nuevo proyecto, en el buscador escribir api y la pantalla seleccionar ASP.NET Core Web API para crear una API RESTful con controladores minimalistas en ASP.NET Core.



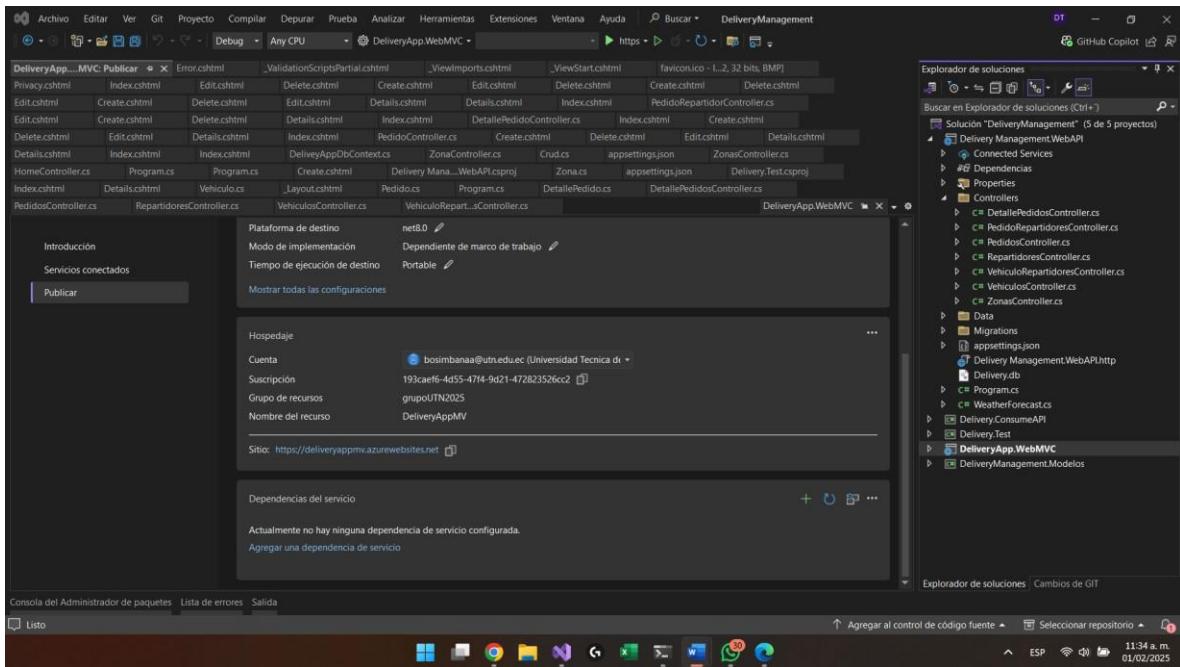
17.- En configure su nuevo proyecto escribir el nombre del proyecto Delivery Management.WebAPI.



18.- Una vez creado el nuevo proyecto, hacer clic en Controllers y agregar nuevo elemento con scaffolding y seleccionar controlador de API con acciones que usan Entity Framework, en esta sección se agregara todos los controladores necesarios. Se utiliza la funcionalidad de **scaffolding** en Visual Studio para agregar un nuevo elemento al proyecto **DeliveryManagement.WebAPI**.

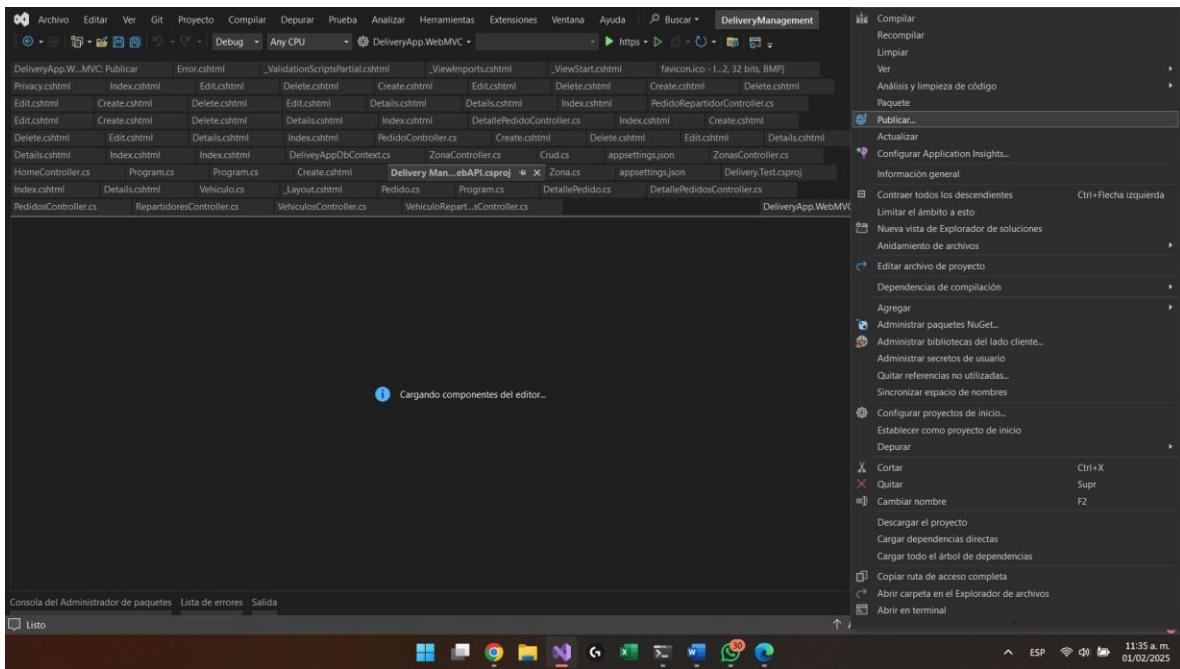


19.- Revisar que todos los controladores no contengan errores y de esa poder realizar el despliegue de la API REST



PUBLICACIÓN DE API REST

20.- Realizar la publicación de la API REST.



LINK DEL API REST

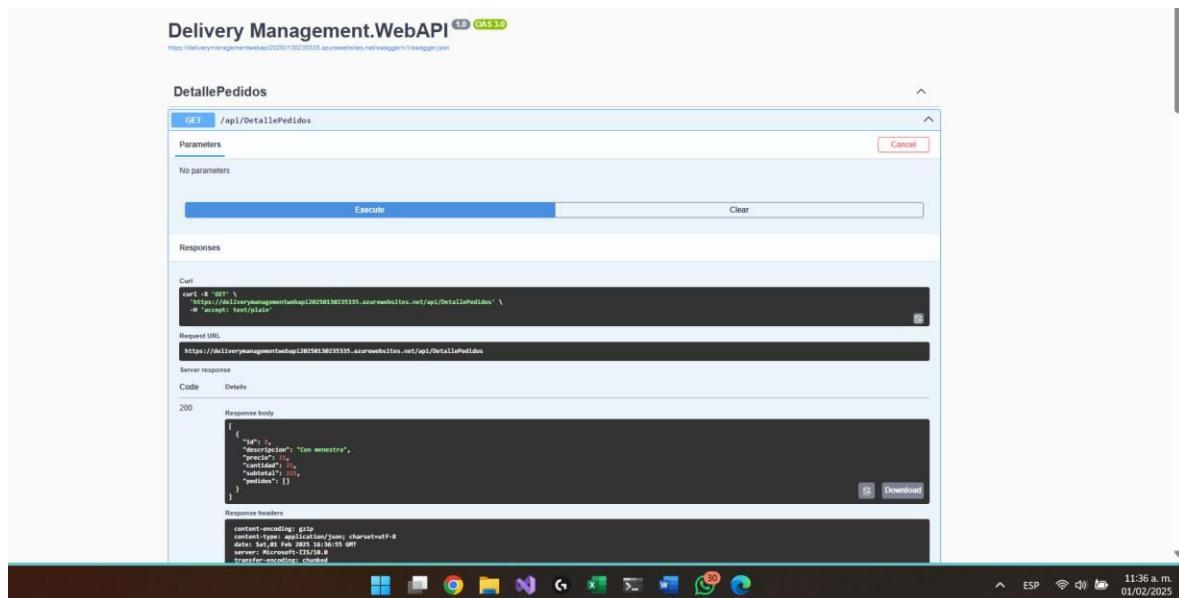
21.- Obtenemos el link de API REST

[Swagger UI](#)

DOCUMENTACIÓN CON SWAGGER UI

22.- Ejecutar el enlace y se visualiza todos los esquemas que tenemos.

23.- Se realiza a todas las funciones de Crear, Leer, Actualizar y Eliminar. Se han realizado pruebas exhaustivas de todos los endpoints mediante herramientas como **Postman** y **Swagger UI**. Se verificó que cada solicitud devuelve las respuestas esperadas y que se manejen adecuadamente los errores en caso de datos incorrectos o no encontrados.

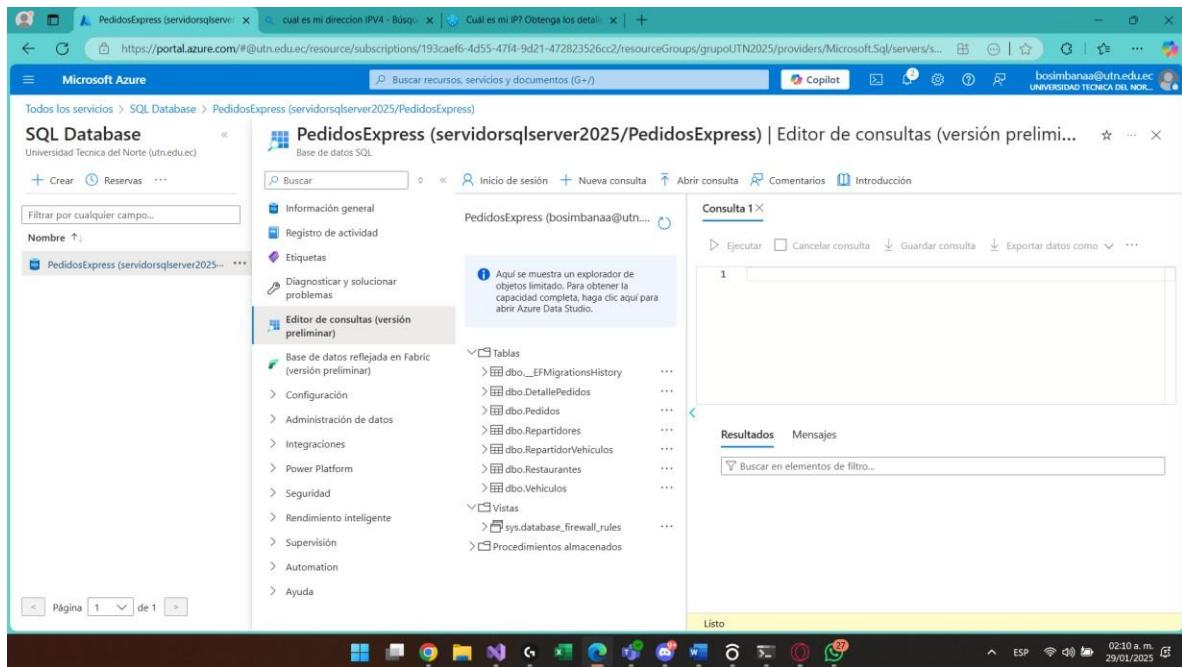


APLICACIÓN MVC Y CONSUMO DE API

CONFIGURACIÓN DE LA BASE DE DATOS EN EL SERVIDOR

La base de datos almacena todos los registros de pedidos, productos y usuarios, asegurando que la aplicación pueda operar sin problemas. Se implementaron mecanismos de respaldo automático y optimización de consultas para mejorar el rendimiento del sistema.

24.- Se muestra el **Editor de consultas** para la base de datos **PedidosExpress** alojada en **Microsoft Azure SQL Database**.



MÓDULO DE PAQUETES

Se desarrolló un módulo específico para la gestión de paquetes dentro del sistema. Este módulo permite:

- Registrar nuevos paquetes con información detallada.
- Consultar el estado actual de cada paquete.
- Modificar y actualizar la información de los paquetes.
- Eliminar paquetes en caso de ser necesario.

25.- Se implementa el método **GetRestaurantes** en el controlador **RestaurantesController** dentro del proyecto **DeliveryManagement.WebAPI**. Los puntos destacados son:

- **Método HTTP:** [HttpGet] define una acción para obtener datos de restaurantes.
- **Retorno:** Una lista de restaurantes (IEnumerable<Restaurante>) incluyendo la relación con pedidos mediante. `Include(r => r.Pedidos)`.
- **Consulta asincrónica:** Utiliza `ToListAsync()` para una ejecución eficiente.

The screenshot shows the Visual Studio IDE interface. In the top navigation bar, the following items are visible: Archivo, Editar, Ver, Git, Proyecto, Compilar, Depurar, Prueba, Analizar, Herramientas, Extensiones, Vigneta, Ayuda, Buscar, and DeliveryManagement. The title bar says "DeliveryManagement".

The code editor displays the following C# code:

```

19     {
20         _context = context;
21     }
22
23     // GET: api/Restaurantes
24     [HttpGet]
25     public async Task<ActionResult><IEnumerable<Restaurante>> GetRestaurante()
26     {
27         return await _context
28             .Restaurantes
29                 .Include(r => r.Pedidos)
30                 .ToListAsync();
31     }

```

The Solution Explorer on the right shows the project structure for "DeliveryManagement" (2 de 2 proyectos):

- Delivery Management WebAPI
 - Connected Services
 - Properties
 - Controllers
 - Data
 - Migrations
 - appsettings.json
 - DeliveryManagement.WebAPI.csproj
 - DeliveryManagement.db

The NuGet Package Manager console at the bottom shows the following output:

```

La licencia de cada paquete la concede su propietario. NuGet no se hace responsable de los paquetes de terceros ni concede ninguna licencia sobre ellos. Algunos paquetes pueden incluir dependencias que se rigen por licencias adicionales. Siga la URL de origen del paquete (fuente) para determinar cualquier dependencia.

Versión de host 6.12.2.1 de la consola del Administrador de paquetes

Escriba 'get-help NuGet' para ver todos los comandos de NuGet disponibles.

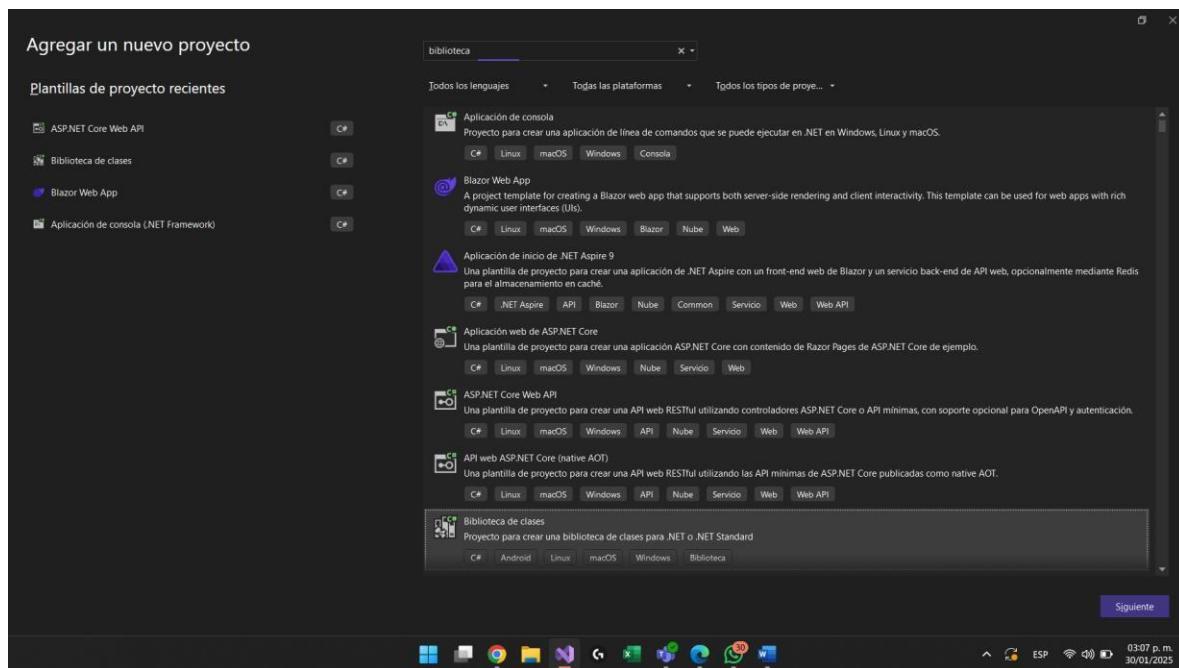
PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 8.0.11
Restaurando paquetes para D:\Asus\Documents\Software\VI Semestre\Cloud Computing\Delivery Management\Delivery Management.WebAPI\Delivery Management.WebAPI.csproj...

```

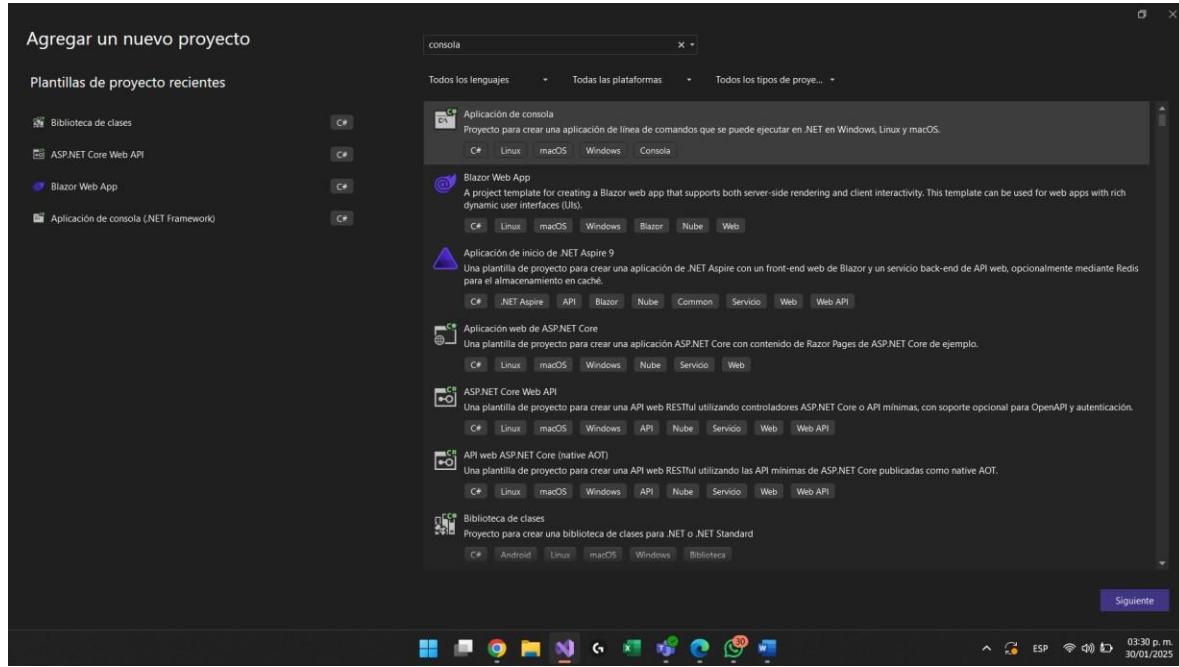
API CONSUMER: CONSUMO DE LA API REST

Para que la aplicación MVC pueda interactuar con el API REST, se implementó un **API Consumer** que se encarga de realizar las solicitudes necesarias. El API Consumer se comunica con la API REST utilizando peticiones HTTP, manejando los datos obtenidos y presentándolos en la interfaz de usuario de la aplicación MVC.

26.- Agregar nuevo proyecto y en el buscador escribir biblioteca y seleccionar Biblioteca de clases y dar el nombre del proyecto y crear proyecto.

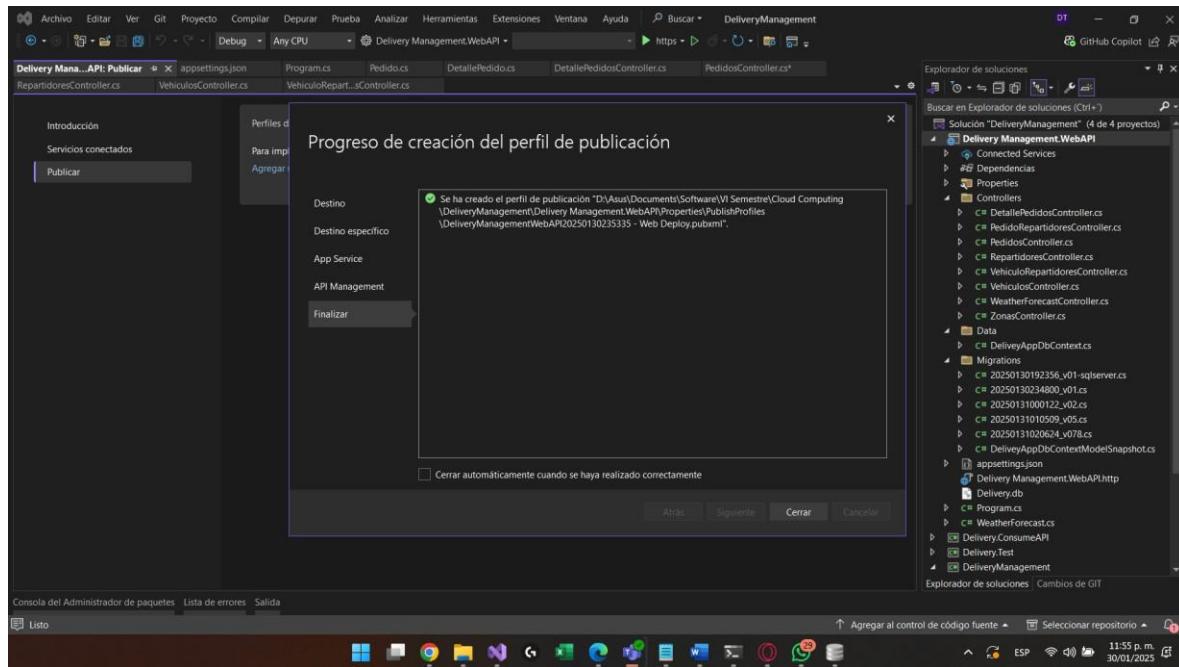


27.- Seguidamente agregar un nuevo proyecto y en el buscador escribir consola y seleccionar Aplicación de consola, dar el nombre del proyecto y crear.

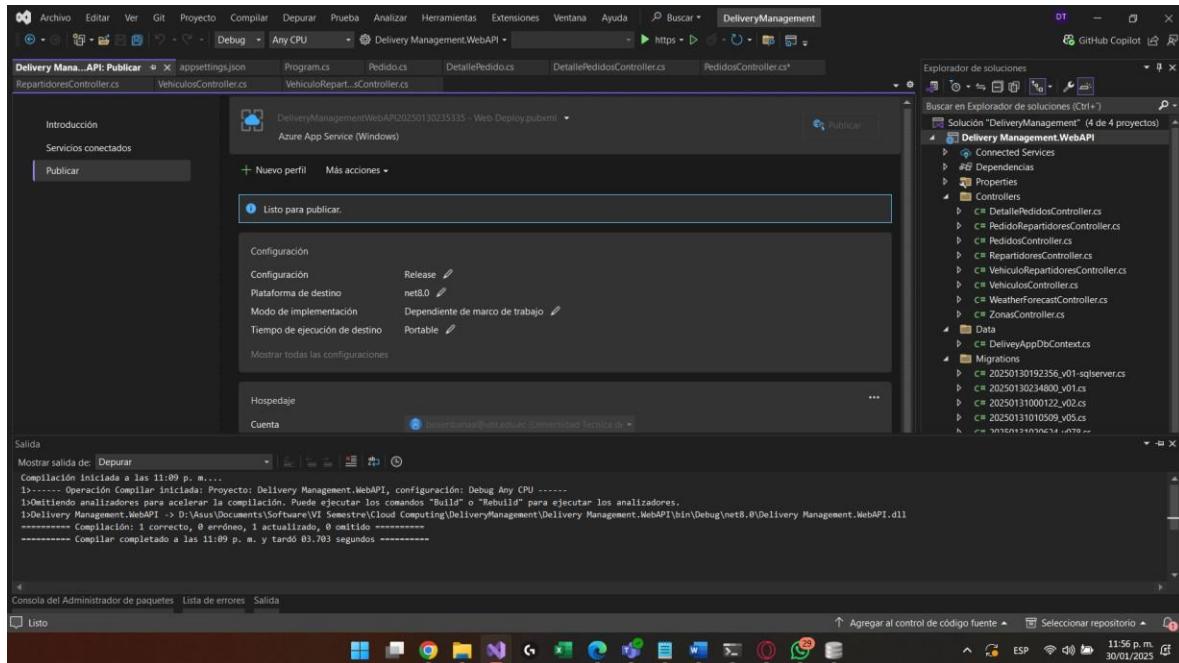


PUBLICACIÓN DEL API REST

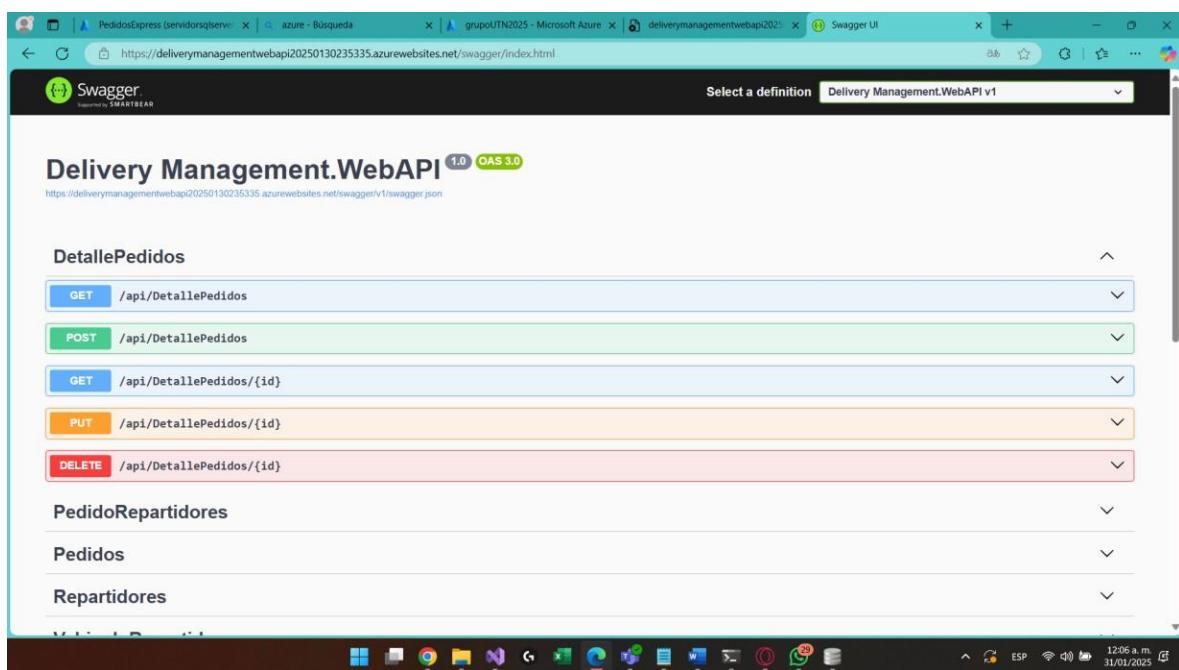
28.- Una vez finalizada la implementación del API Consumer y la validación de su correcto funcionamiento, se crea un perfil de publicación.



29.- Se procedió a la publicación del API REST en el servidor, asegurando su accesibilidad para la aplicación MVC y otros clientes.

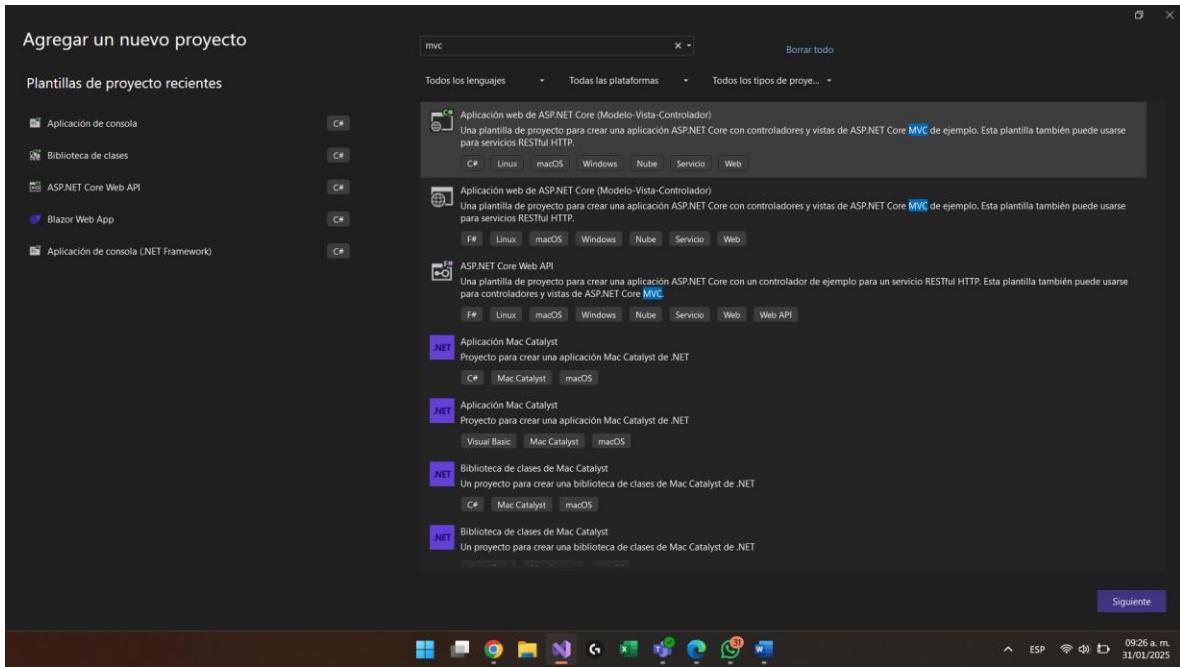


30.- Se observa la publicación de la API REST con los esquemas que se implementaron anteriormente.



CREACIÓN DEL PROYECTO MVC

31.- Agregar un nuevo proyecto y en el buscador escribir mvc, seleccionar Aplicación web de ASP.NET Core (Modelo-Vista-Controlador), dar el nombre al proyecto y crear.

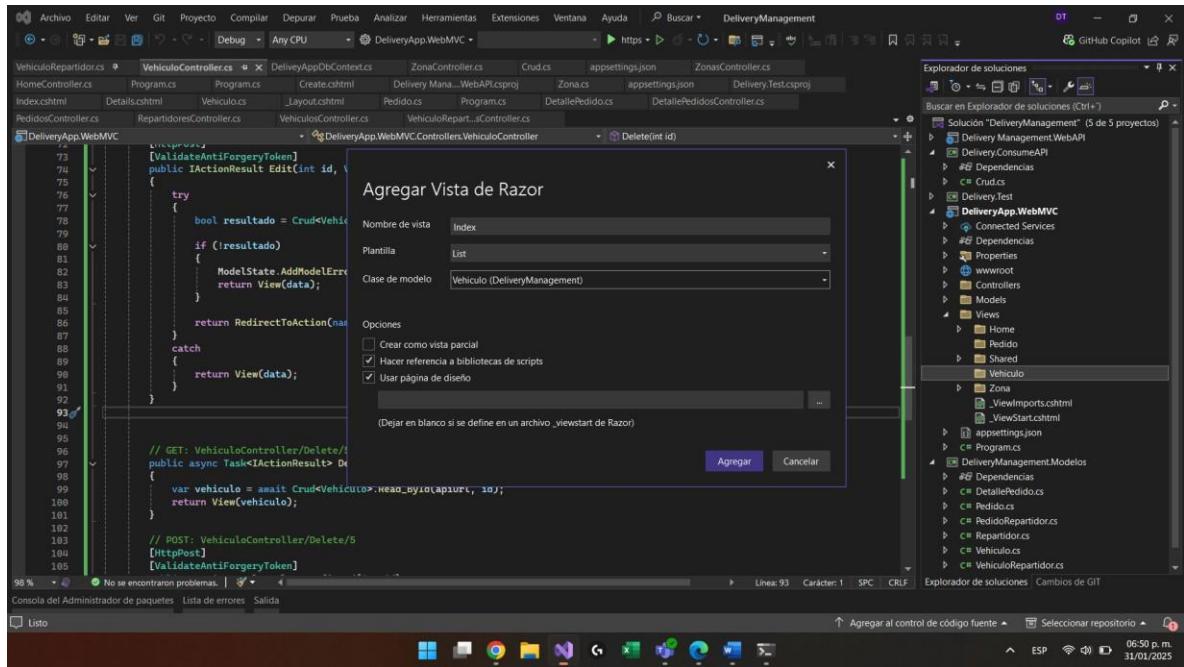


32.- En el proyecto DeliveryApp.WebMVC agregar una nueva **Vista de Razor** en el proyecto.

Detalles principales:

- **Nombre de la vista:** Index.
- **Plantilla:** Basada en el modelo Vehiculo del namespace DeliveryManagement.
- **Opciones seleccionadas:**
 - Crear como vista parcial.
 - Hacer referencia a bibliotecas de scripts.
 - Usar página de diseño predeterminada.

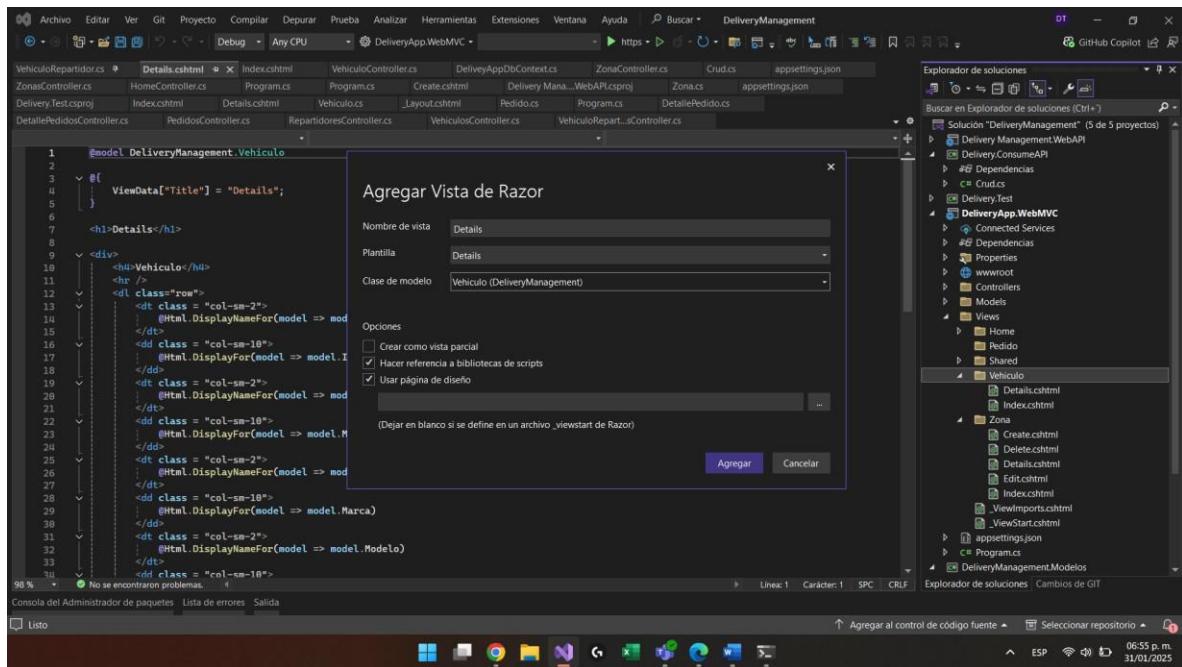
Esta configuración permite crear una vista Razor asociada al controlador, facilitando la generación dinámica de contenido para la interfaz de usuario.



33.- Creación y edición de una Vista de Razor llamada Details dentro del proyecto DeliveryApp.WebMVC:

- **Nombre:** Details.
- **Modelo asociado:** Vehiculo del namespace DeliveryManagement.
- **Contenido:** La vista utiliza la propiedad ViewData["Title"] para el título y muestra detalles del modelo Vehiculo en un diseño estructurado con etiquetas HTML y clases CSS.
- **Opciones habilitadas:**
 - Referencia a bibliotecas de scripts.
 - Uso de página de diseño.

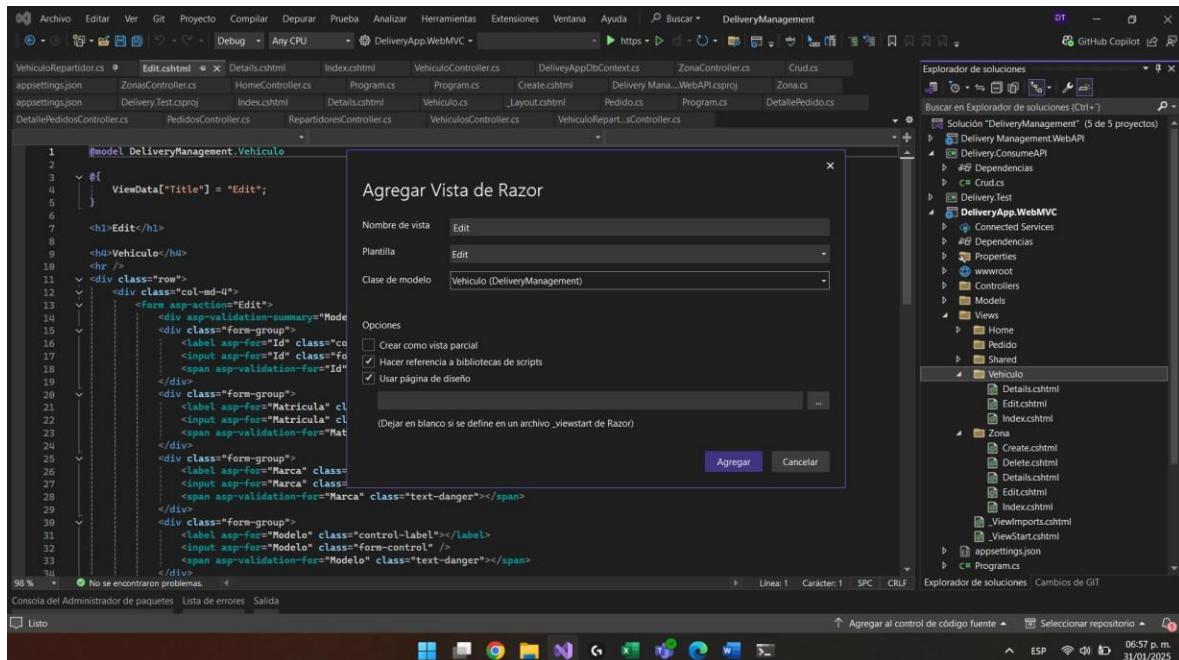
Esta vista permite visualizar información detallada de un objeto Vehiculo en la aplicación web.



34.- Creación y edición de la Vista de Razor llamada Edit en el proyecto DeliveryApp.WebMVC:

- **Nombre:** Edit.
- **Modelo asociado:** Vehiculo del namespace DeliveryManagement.
- **Contenido:** La vista utiliza etiquetas HTML y helpers de Razor como asp-for para generar un formulario interactivo que permite editar las propiedades de un objeto Vehiculo, incluyendo:
 - Matrícula.
 - Marca.
 - Modelo.
- **Opciones habilitadas:**
 - Referencias a bibliotecas de scripts.
 - Uso de página de diseño.

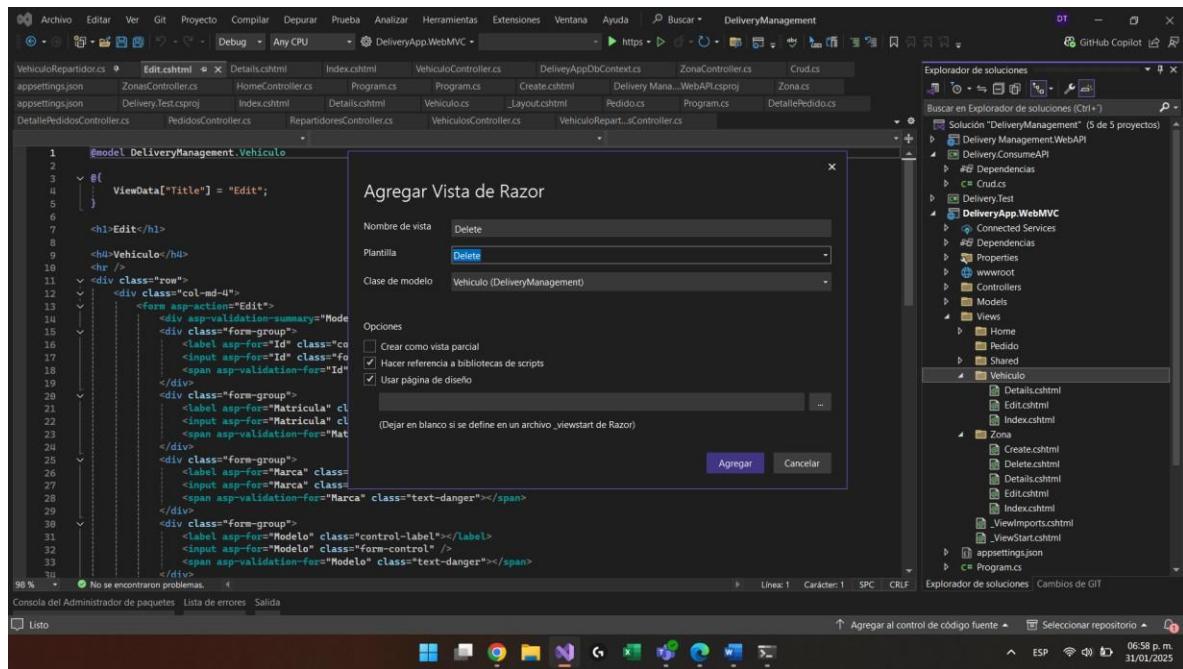
Esta vista facilita la edición de información en la capa de presentación de la aplicación web.



35.- Creación de la Vista de Razor llamada Delete en el proyecto DeliveryApp.WebMVC:

- **Nombre:** Delete.
- **Modelo asociado:** Vehiculo del namespace DeliveryManagement.
- **Opciones habilitadas:**
 - Referencias a bibliotecas de scripts.
 - Uso de página de diseño.

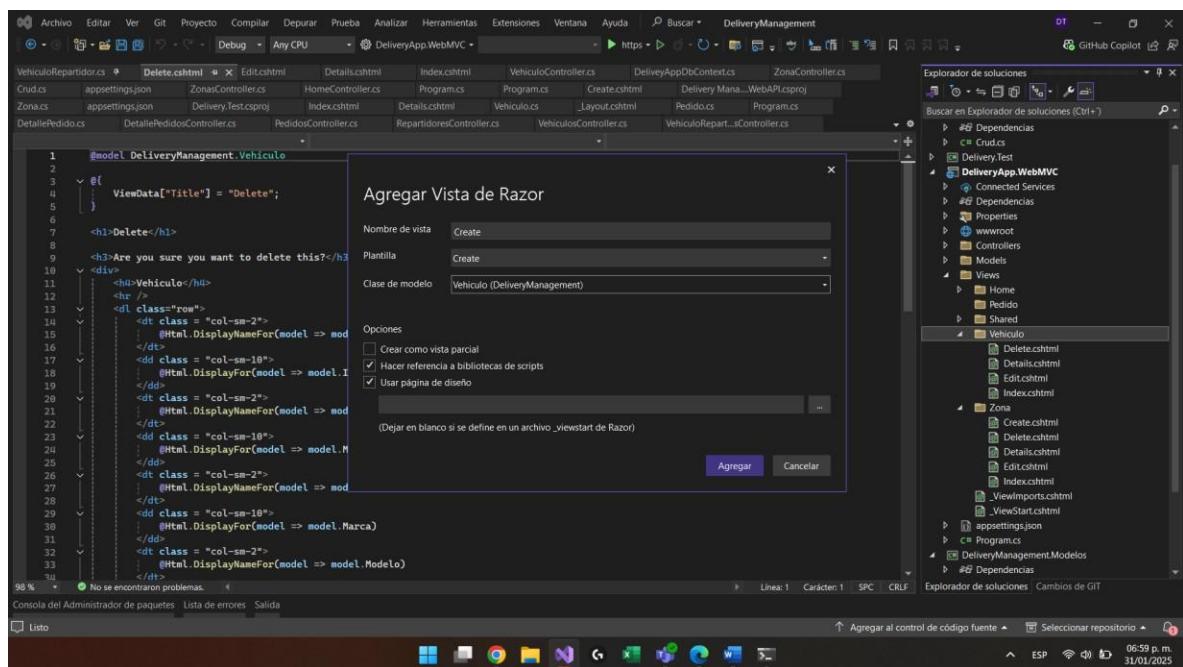
Esta vista servirá para confirmar y procesar la eliminación de un objeto Vehiculo, integrándose con el controlador correspondiente para realizar la acción de eliminación en la base de datos.



36.- Crear la Vista de Razor llamada Create dentro del proyecto DeliveryApp.WebMVC:

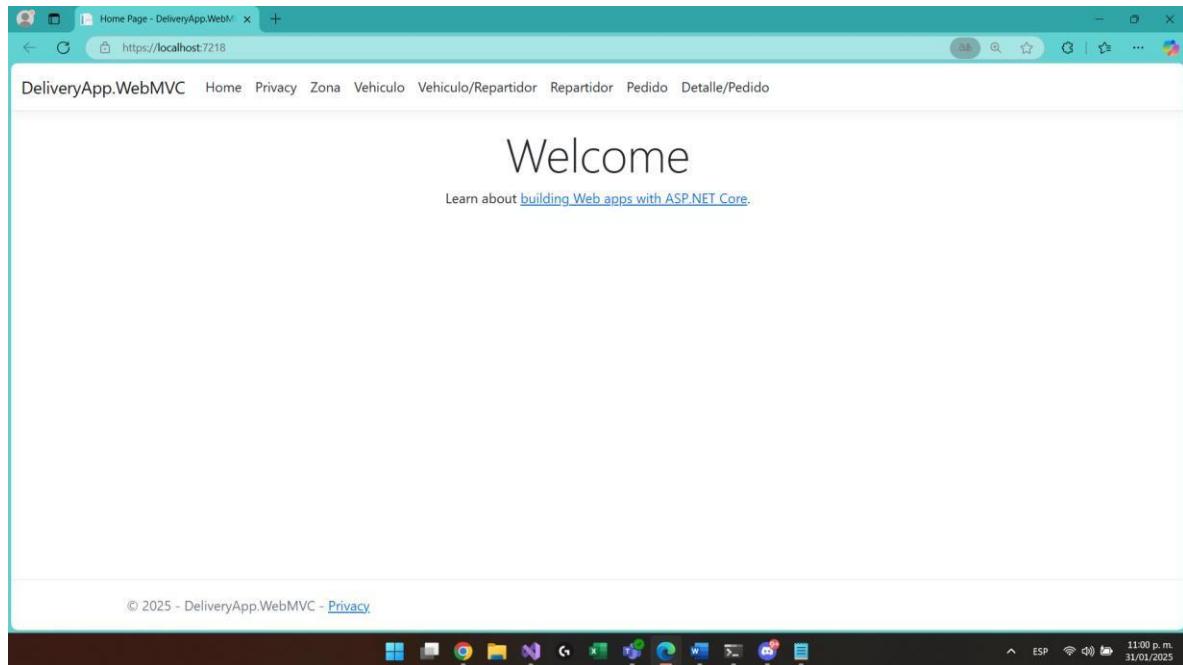
- Nombre:** Create.
- Modelo asociado:** Vehiculo del namespace DeliveryManagement.
- Opciones habilitadas:**
 - Referencias a bibliotecas de scripts.
 - Uso de página de diseño.

Esta vista permitirá generar un formulario para añadir nuevos registros de vehículos, interactuando con el controlador correspondiente para guardar los datos en la base de datos.



VISUALIZACIÓN INICIAL

37.- Se puede observar la pantalla inicial de la página web.



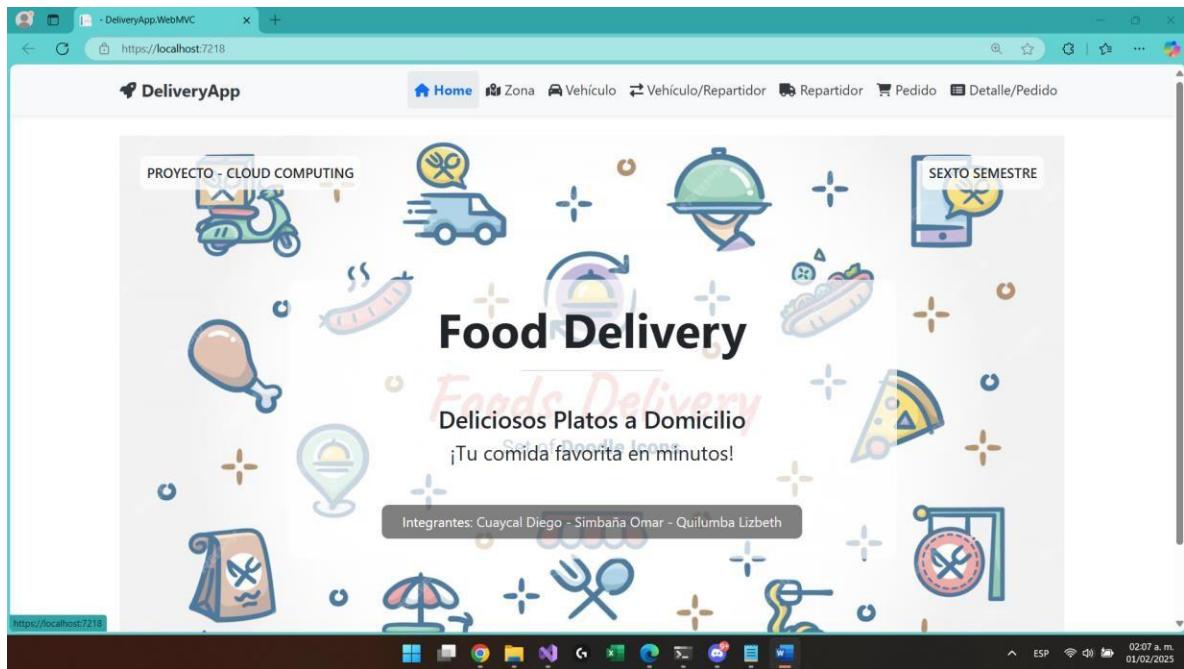
38.- Se visualiza el Crud de Vehículos.

Index						
Create New						
Id	Matricula	Marca	Modelo	CapacidadCarga	Tipo	
1	MACA	BUENA	A	0	12	Edit Details Delete
2	string	string	string	0	string	Edit Details Delete
5	CDBA2	Mazda	Caminion	2	fa	Edit Details Delete

© 2025 - DeliveryApp.WebMVC - Privacy

11:00 p.m.
31/01/2025

39.- Vista con todas las opciones funcionales.



40.- Visualización de Zona de entregas.

A screenshot of the 'Zonas de Entrega' (Delivery Areas) list view. The title bar reads 'Index - DeliveryApp.WebMVC' and the address bar shows 'https://localhost:7218/Zona'. The page has a background pattern of food-related icons. It displays a table with one row of data:

ID	Nombre	Acciones
17	Ibarra	Editar Detalles Eliminar

The top navigation bar includes links for Home, Zona, Vehículo, Vehículo/Repartidor, Repartidor, Pedido, and Detalle/Pedido. A 'Nueva Zona' (New Zone) button is located at the top right of the table area.

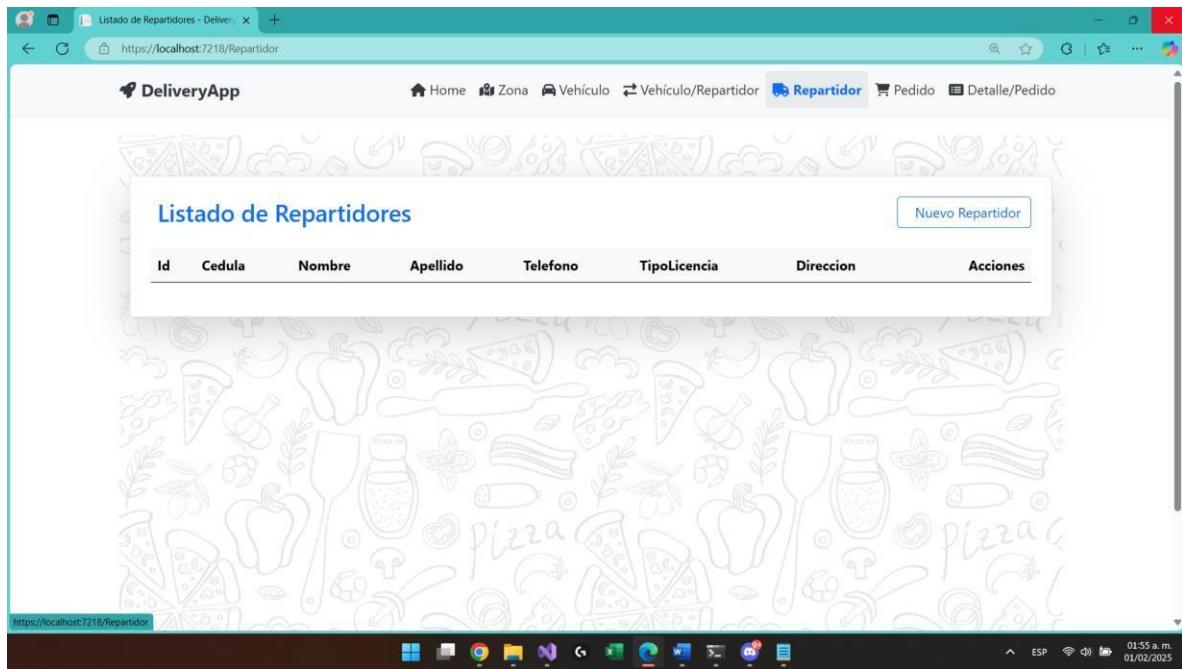
41.- Visualización de Listado de vehículos.

The screenshot shows a web browser window titled "Listado de Vehículos - DeliveryApp". The URL is <https://localhost:7218/Vehiculo>. The page has a header with the DeliveryApp logo and navigation links: Home, Zona, Vehículo (which is highlighted in blue), Vehículo/Repartidor, Repartidor, Pedido, and Detalle/Pedido. Below the header is a decorative banner with pizza-related illustrations. The main content area is titled "Listado de Vehículos" and contains a table with two rows of vehicle data. The columns are: Id, Matrícula, Marca, Modelo, CapacidadCarga, and Tipo. The first row has Id=2, Matrícula="string", Marca="string", Modelo="string", CapacidadCarga="0", and Tipo="string". The second row has Id=5, Matrícula="CDBA2", Marca="Mazda", Modelo="Caminion", CapacidadCarga="2", and Tipo="fa". Each row has three buttons in the "Acciones" column: "Editar" (yellow), "Detalles" (blue), and "Eliminar" (red). A "Nuevo Vehículo" button is located in the top right corner of the table area. The bottom of the screen shows a Windows taskbar with various icons and the system tray indicating the date and time.

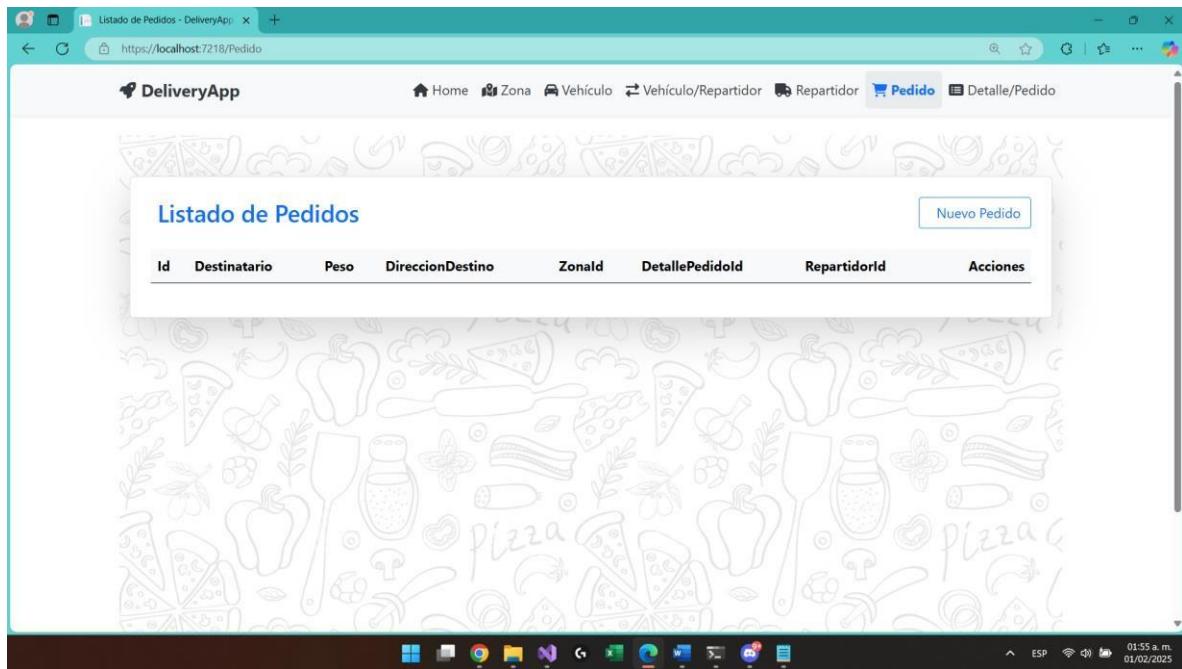
42.- Visualización de Asignaciones de vehículos.

The screenshot shows a web browser window titled "Asignaciones de Vehículos - DeliveryApp". The URL is <https://localhost:7218/VehiculoRepartidor>. The page has a header with the DeliveryApp logo and navigation links: Home, Zona, Vehículo, Vehículo/Repartidor (which is highlighted in blue), Repartidor, Pedido, and Detalle/Pedido. Below the header is a decorative banner with pizza-related illustrations. The main content area is titled "Asignaciones de Vehículos" and contains a table with two columns: Id and RepartidorId. The first row has Id and RepartidorId both empty. A "Nueva Asignación" button is located in the top right corner of the table area. The bottom of the screen shows a Windows taskbar with various icons and the system tray indicating the date and time.

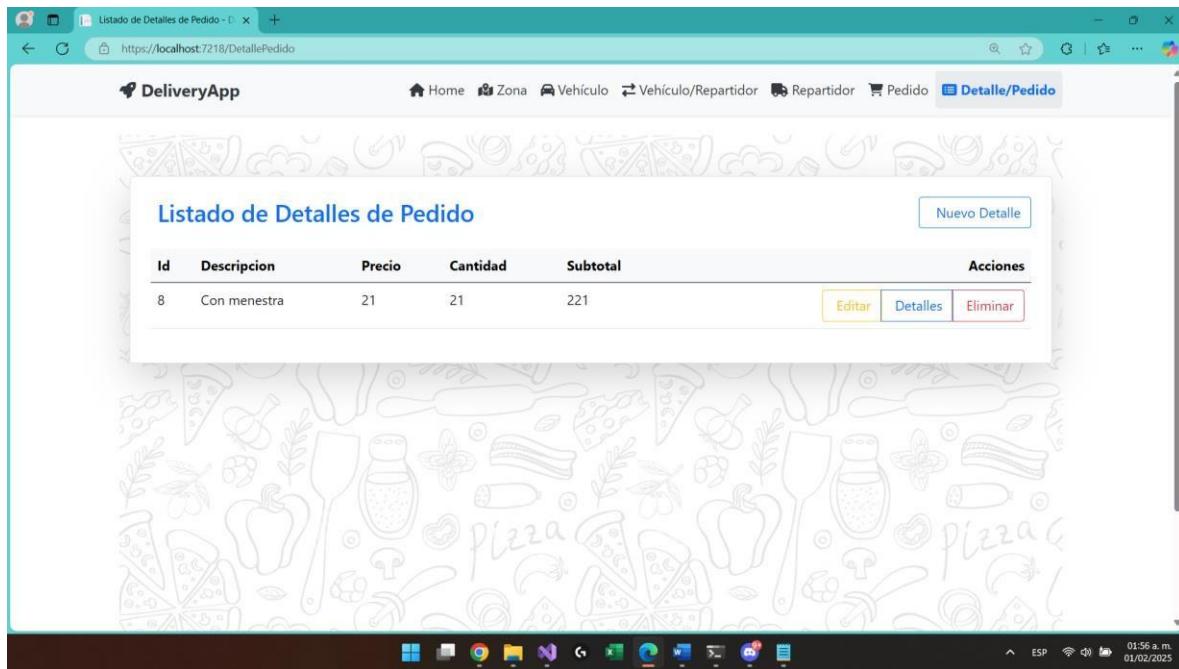
43.- Visualización de Listado de repartidores.



44.- Visualización de Listado de pedidos.



45.- Visualización de Listado de Detalles de pedido.

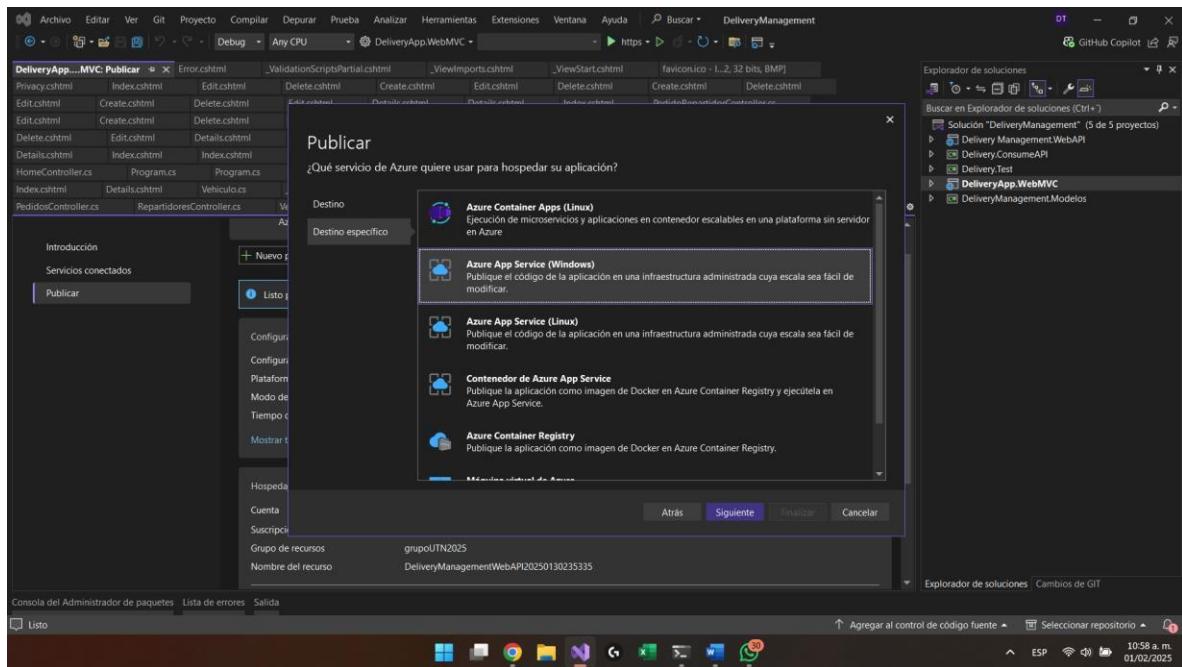


PUBLICACIÓN MVC

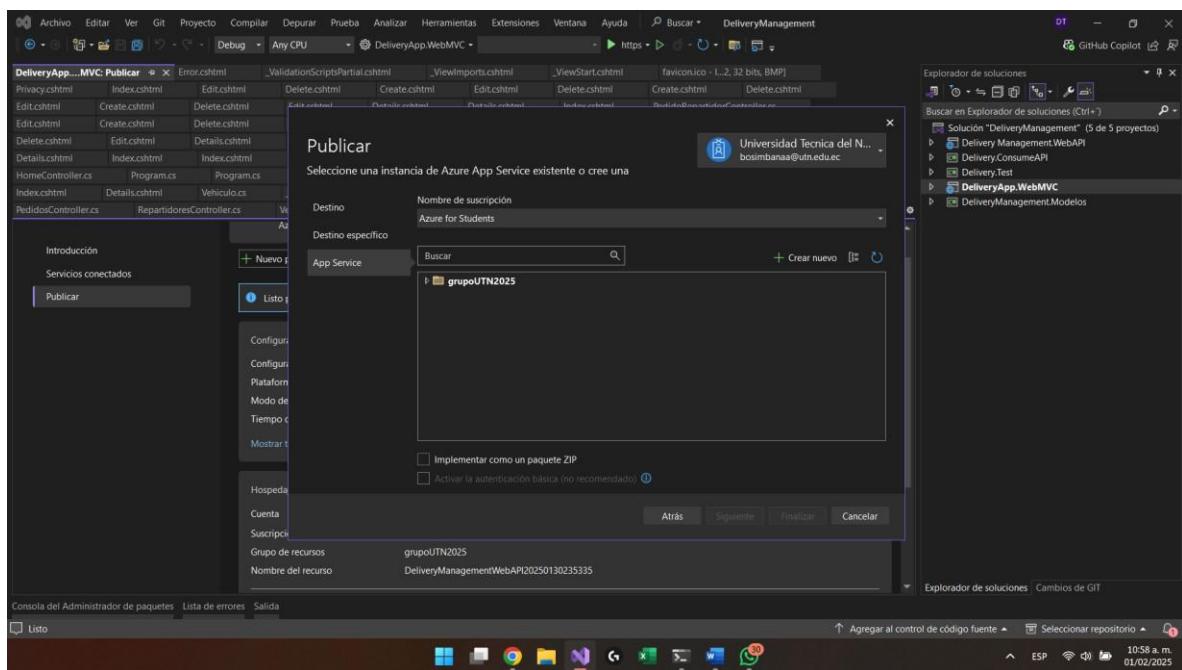
46.- Realizar el proceso de la publicación de la aplicación **DeliveryApp.WebMVC** en **Azure**. Los detalles son:

- **Opciones de servicios de Azure:**
 - **Azure Container Apps (Linux):** Para ejecutar microservicios en contenedores escalables.
 - **Azure App Service (Windows/Linux):** Infraestructura administrada para aplicaciones web.
 - **Azure Container Registry:** Hospedaje de imágenes de contenedores Docker.
- **Grupo de recursos:** grupoUTN2025.
- **Nombre del recurso:** DeliveryManagementWebAPI202310233533.

Este paso configura el despliegue de la aplicación en un servicio de Azure adecuado según las necesidades del proyecto.



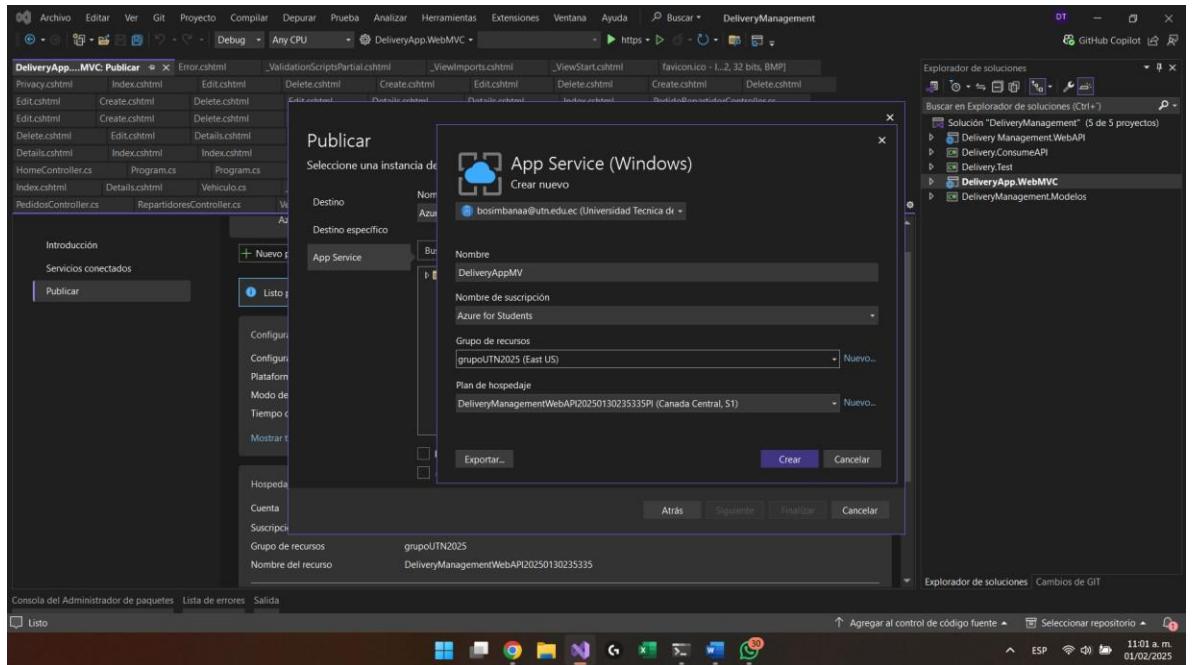
47.- Seleccionar una instancia de Azure App Service existente o puede crear una nueva.



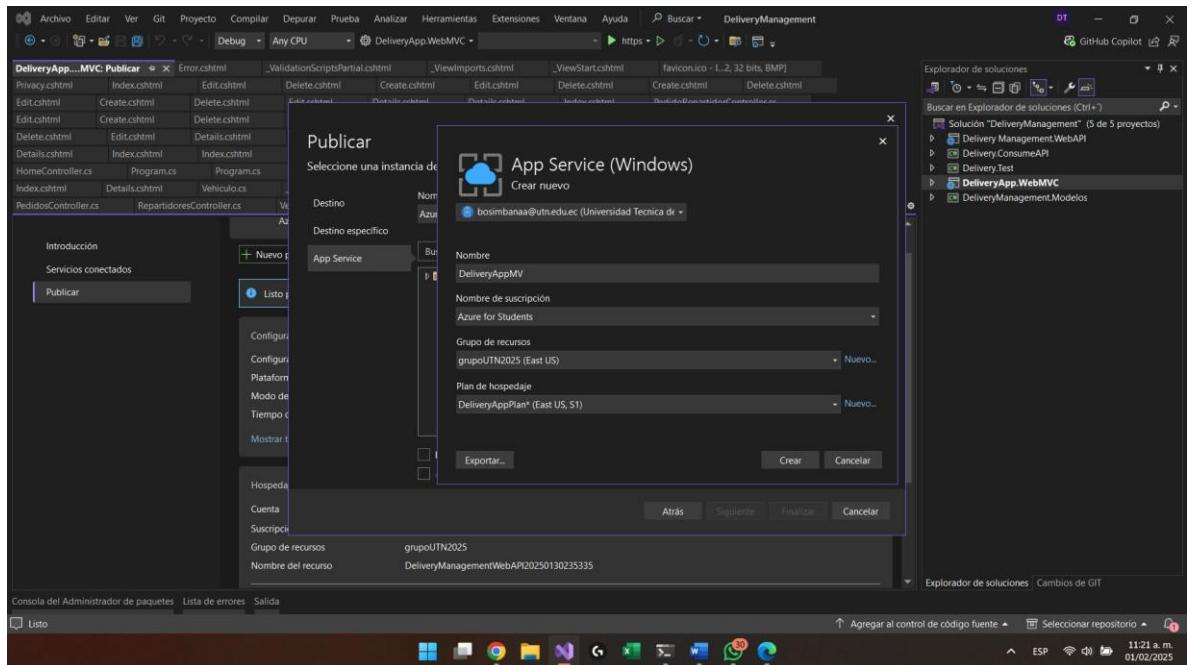
48.- Configurar la publicación de la aplicación **DeliveryApp.WebMVC** en **Azure App Service (Windows)**. Los detalles son:

- **Nombre del recurso:** DeliveryAppMV.
- **Suscripción:** Azure for Students (Universidad Técnica de Norte).
- **Grupo de recursos:** grupoUTN2025.
- **Región:** Canada Central.
- **Plan de hospedaje:** DeliveryManagementWebAPI202510233533.

Esta configuración define el entorno para desplegar la aplicación en una infraestructura administrada por Azure, optimizando la accesibilidad y escalabilidad del sistema.



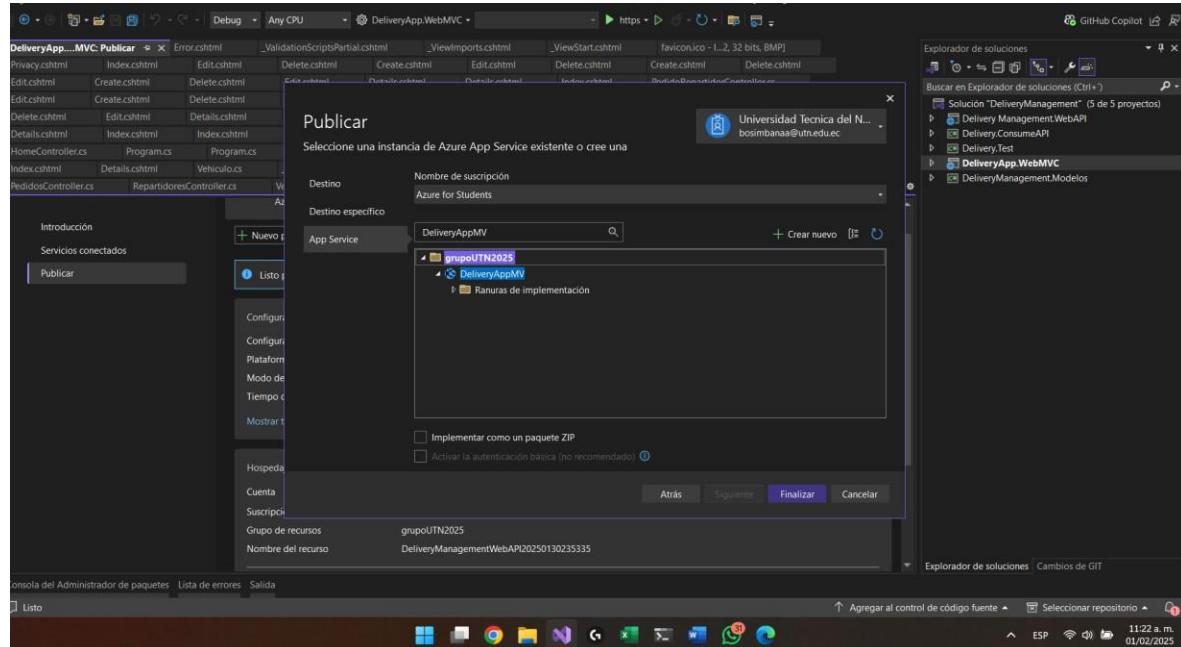
49.- Verificar que todo este correcto elegir el plan de hospedaje.



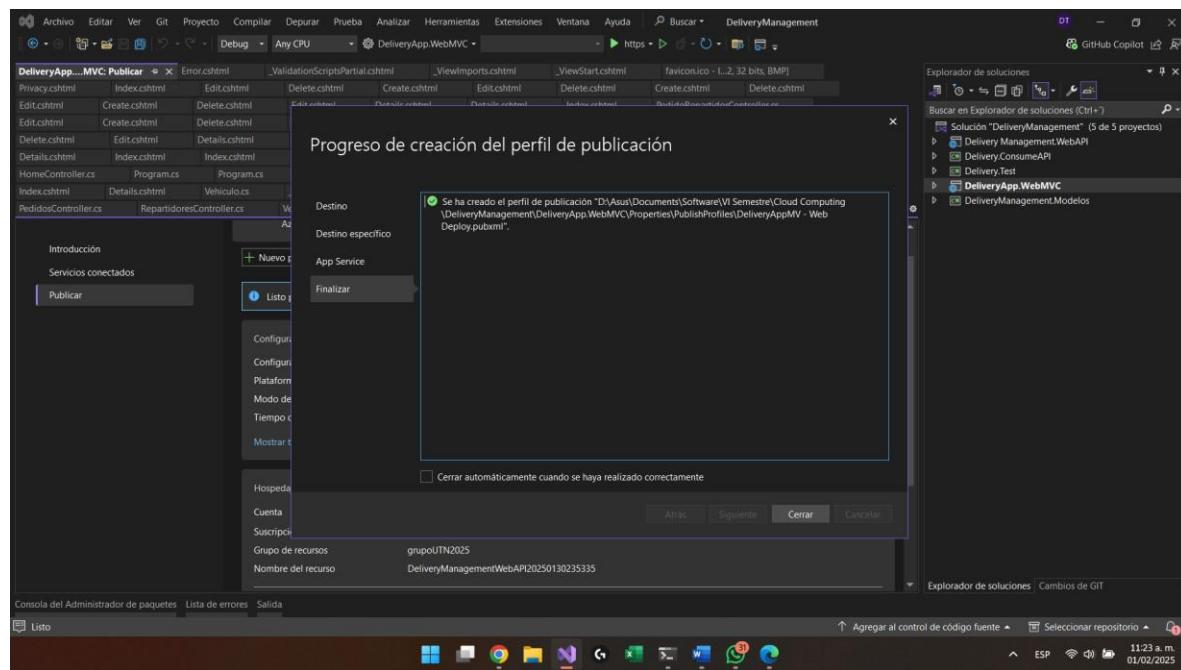
50.- Seleccionar de una instancia de Azure App Service para publicar la aplicación DeliveryApp.WebMVC:

- **Suscripción:** Azure for Students.
- **Grupo de recursos:** grupoUTN2025.
- **Instancia seleccionada:** DeliveryAppMV (ya existente).
- **Opción adicional:** Permite implementar como un paquete ZIP.

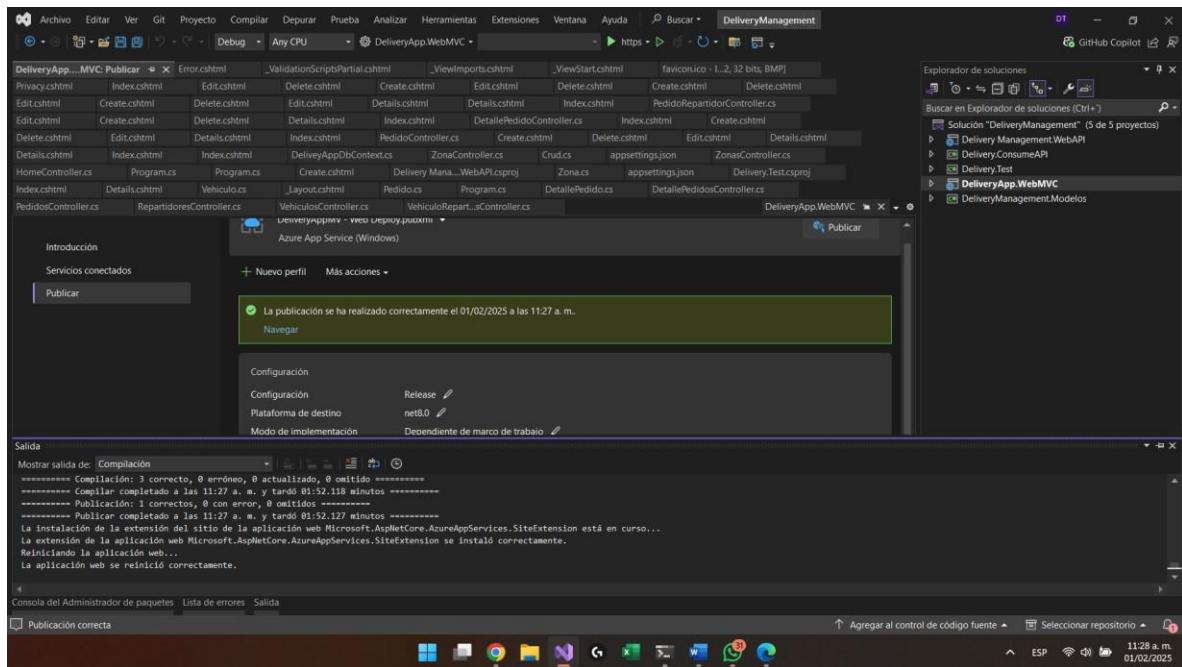
Este paso confirma que la aplicación se desplegará en un servicio ya configurado en Azure, optimizando el uso de recursos existentes.



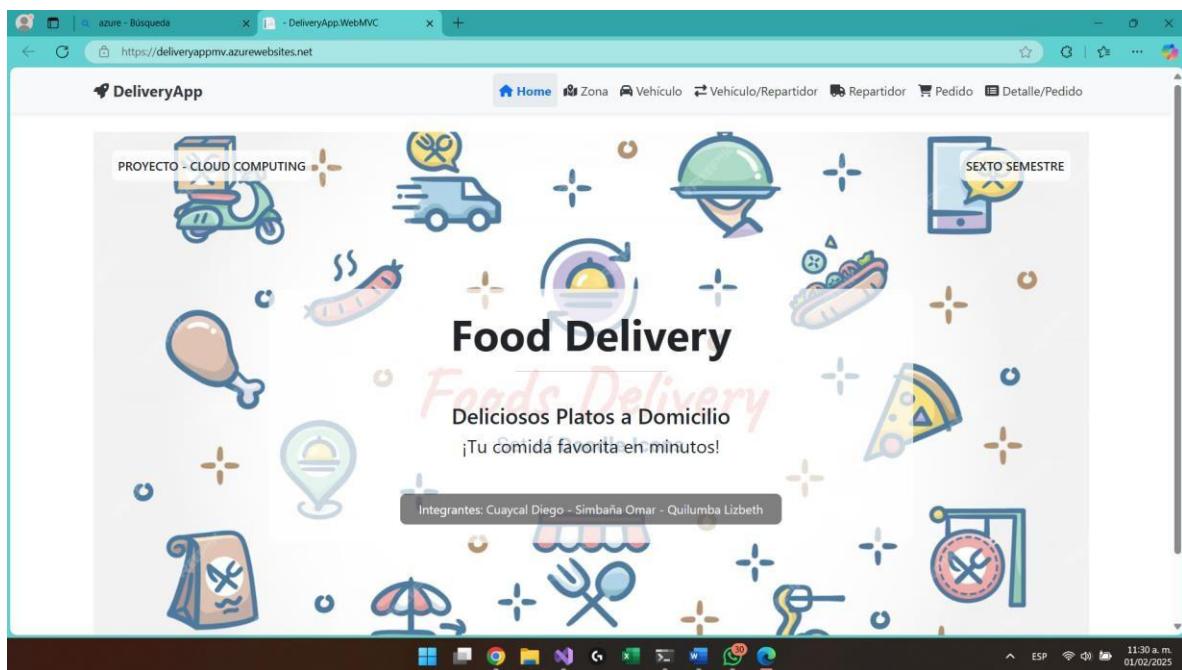
51.- Crear un perfil de publicación.



52.- Se observa que la publicación se ha generado correctamente.



53.- Visualización de la pantalla inicial.

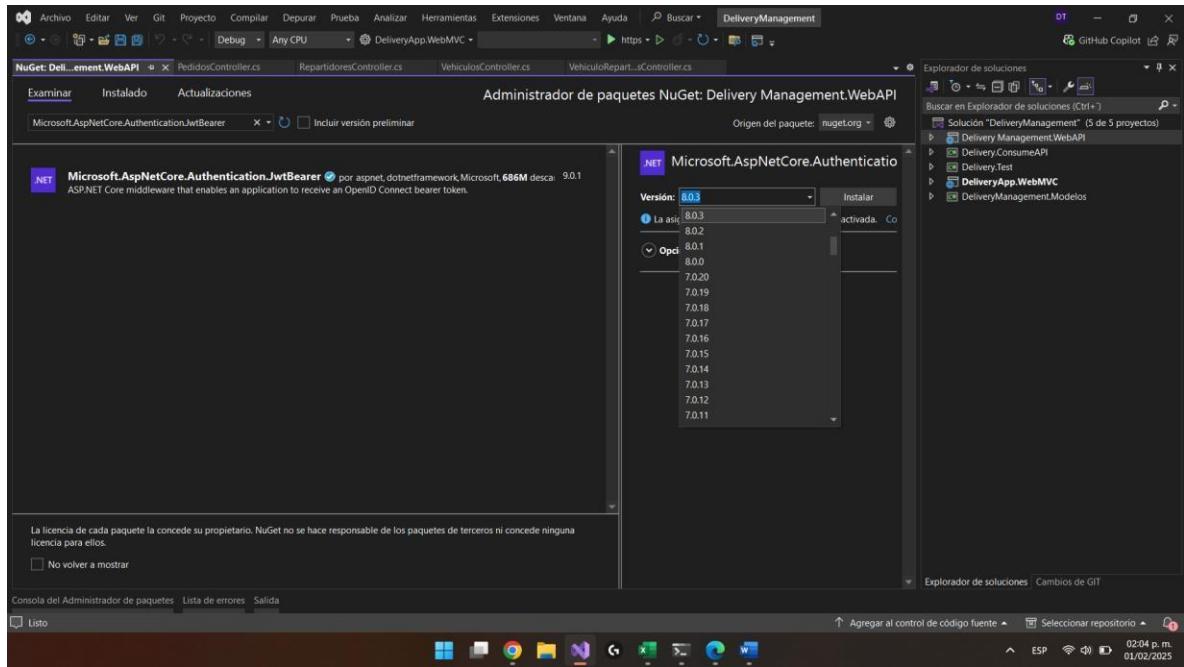


AUTENTICACION

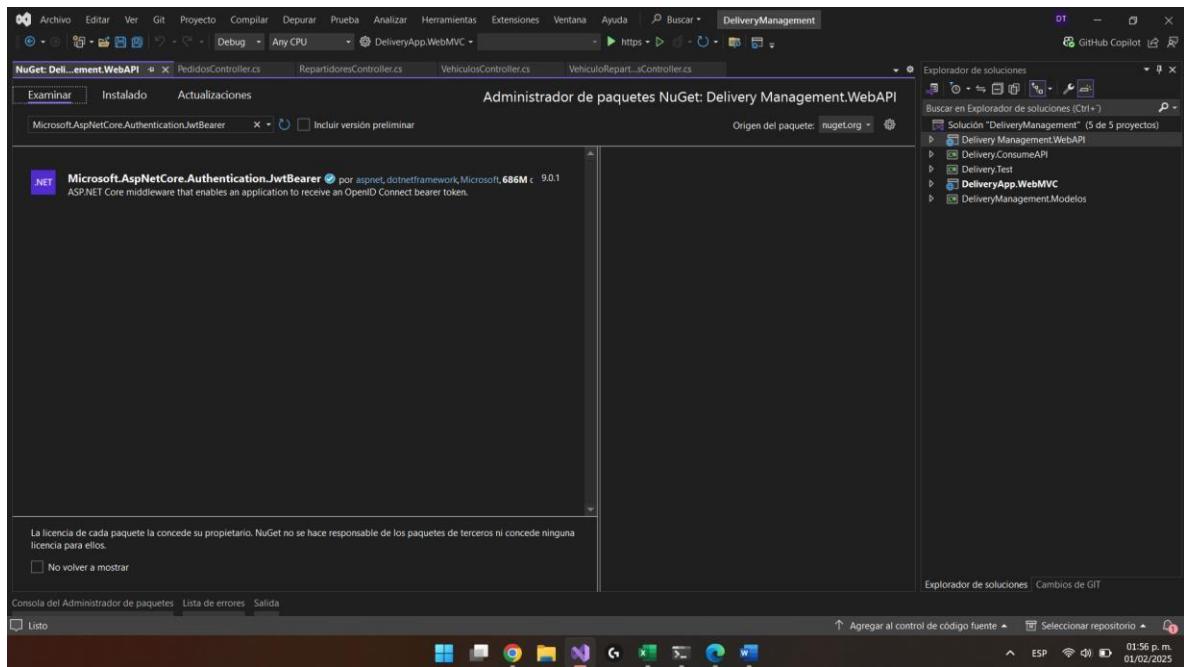
Implementación la parte de medidas de seguridad básicas, como la autenticación y la autorización

54.- En nuestra API REST en estos pasos se puede observar cómo fue creado y las diferentes pruebas que se hizo

Instalación de paquetes compatible con nuestro .net 8

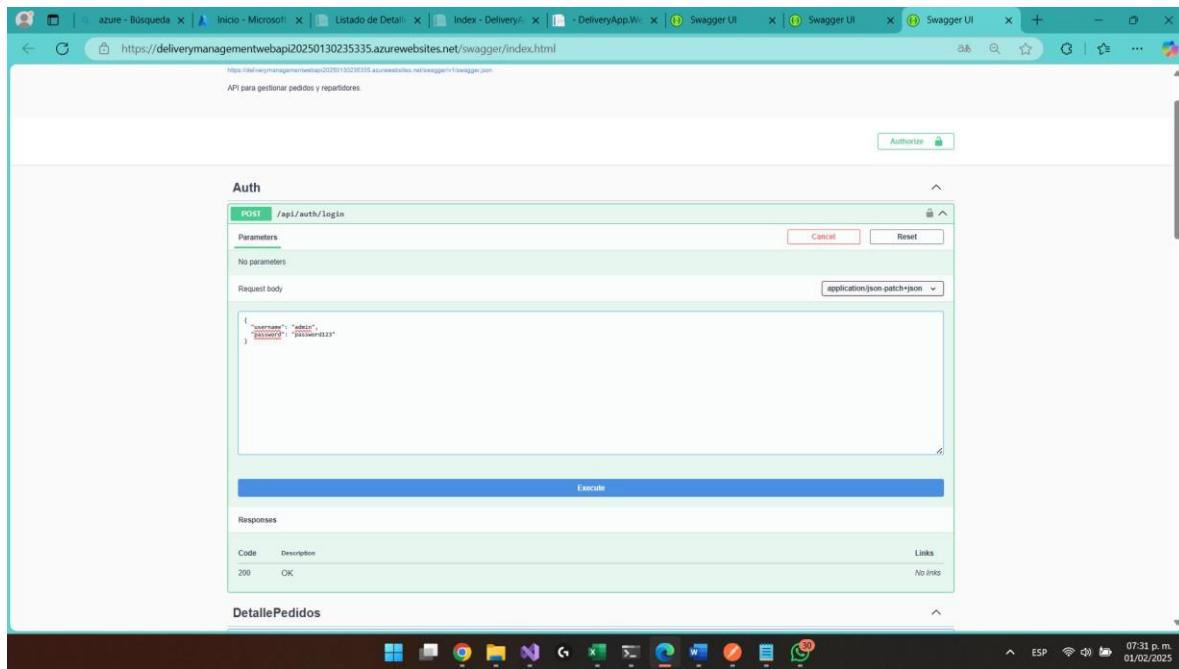


55.- Se instala el paquete de Microsoft.AspNetCore.Authentication.JwtBearer.



56.- Con las siguientes credenciales puede acceder a la API REST donde genera un token para poder ingresar, caso contrario si las credenciales son incorrectas no se podrá ingresar.

user.Username == "admin" && user.Password == "password123



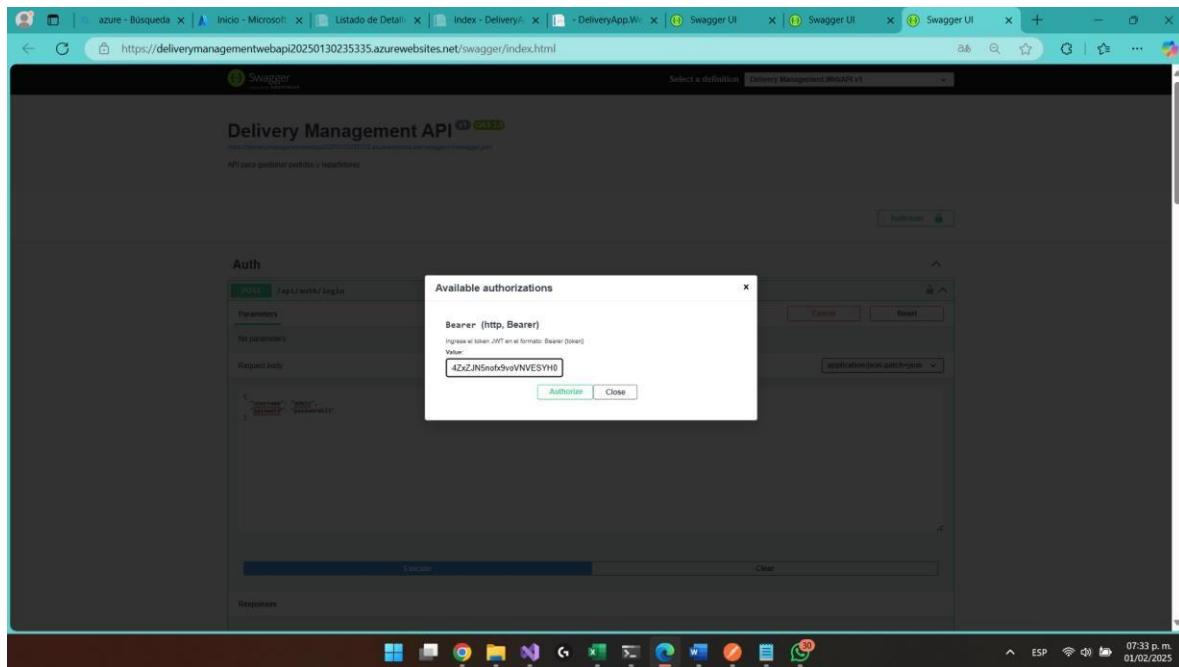
57.- Como se puede observar las credenciales son correctas y por ende me genera el token.

```

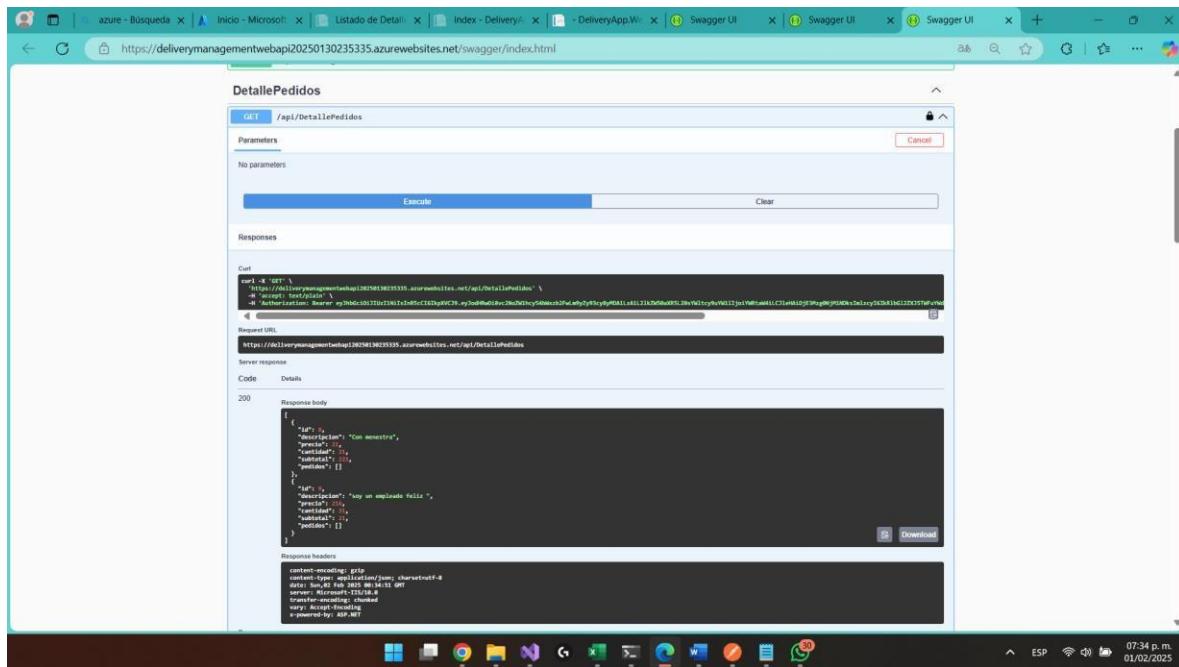
{
  "token": "eyJhbGciOiJITUzIiN1TsInR5cIi6IkpXVCj9-eyJzdWIiOiJ8vc2h0Zmlhcj54blnxzb1FwLwdy2y83cy8yDAll-zA1L21kZW50aXR5L2NsYWhcy9uW11Ijo1YWRtaW4lC1eHA10JE3Mzg0NjM1NDksIelzcyc16IkR1bG32Z0STWFvYHd1bWudfFQ5St1mF12C1G1kR1bG122CTwfUyWd1bWVudENsawadCj9.K5FRY4kKr3kO1ZXKUbkg4zxZ3NmrxFxvovWVE5YH0"
}

```

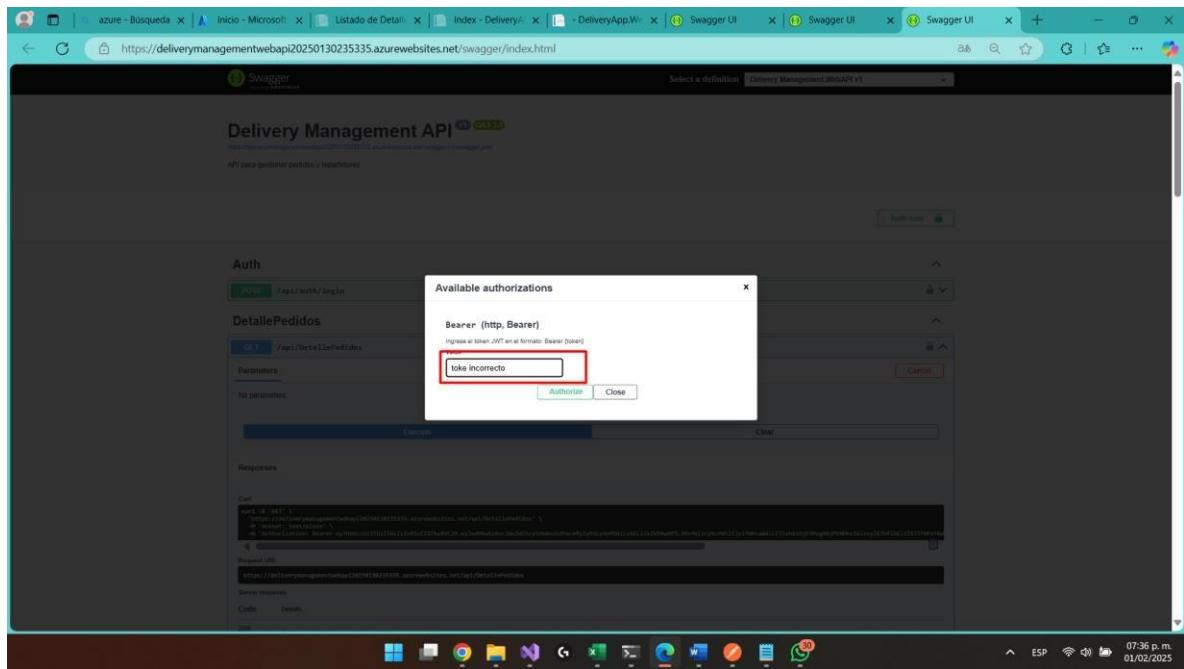
58.- Una vez generado el token se procede a ingresar para realizar las acciones de autorizar o cerrar.



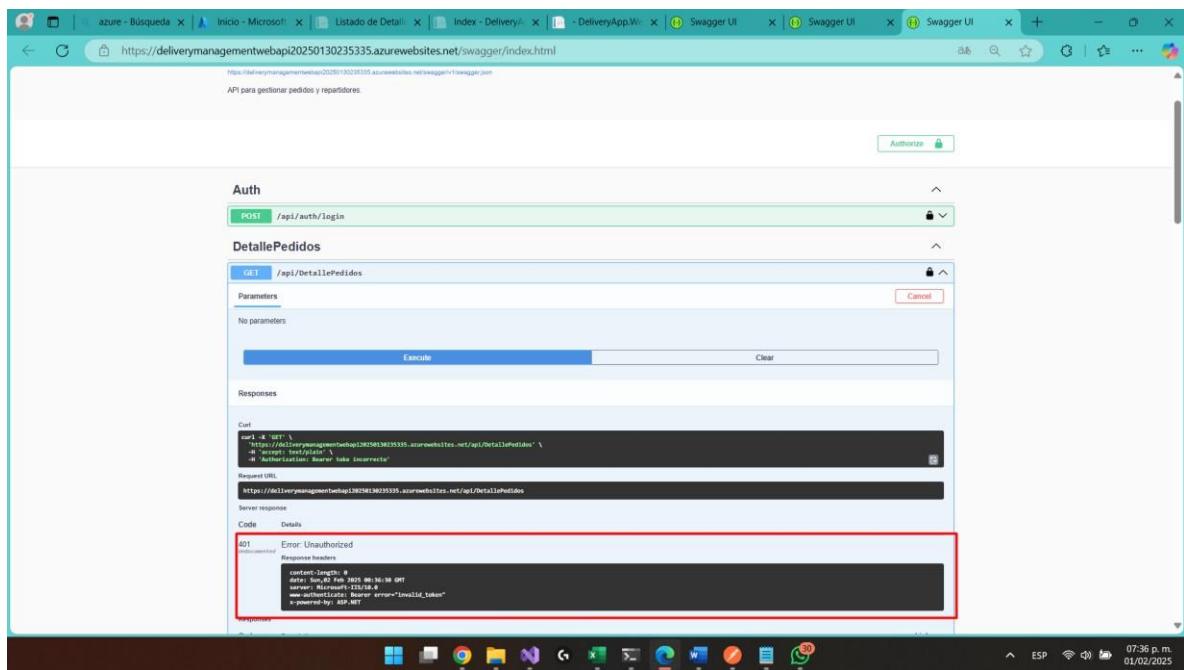
60.- Token ingresado y por ende puedo tener acceso a los datos de los endpoints como se observa en la imagen se obtiene todos los datos de detalle pedido



61.- Segunda prueba con un token incorrecto por ende los endpoints ya no deben dar sus valores porque el usuario ingreso un token incorrecto.



62.- Se visualiza el error 401 de que no está conectado a la API REST.



63.- Última prueba con credenciales incorrectas al rato de querer generar el token:

No pudo generar su token por credenciales incorrectas, ya que las únicas credenciales correctas son:

user.Username == "admin"

Password == "password123"

The screenshot shows the Swagger UI interface for a REST API. In the 'Request body' section, there is a JSON object:

```
{
  "username": "User",
  "password": "123"
}
```

In the 'Responses' section, a failed request is shown with a status code of 401 (Unauthorized). The response body contains the message "Credenciales incorrectas".

LINK DE LA PÁGINA WEB

[- DeliveryApp.WebMVC](#)

CONCLUSIONES

En el proyecto ha logrado:

- Desarrollar un **API REST** funcional y bien documentado con Swagger UI.
- Crear una **aplicación MVC** que consume la API REST y presenta una interfaz para la gestión de pedidos.
- Publicar con éxito ambas aplicaciones en un entorno de producción.

Con esto, se ha construido un sistema escalable y bien estructurado, listo para su uso y futuras mejoras.

La implementación de una API REST junto con una aplicación web basada en MVC permitió la creación de un sistema modular y escalable. Esto facilita la separación de responsabilidades y mejora la mantenibilidad del proyecto.

Se logró desplegar tanto la API REST como la aplicación web en Azure App Services, asegurando una infraestructura confiable y accesible. La base de datos SQL de Azure se configuró correctamente, garantizando una integración eficiente con la API y la aplicación web.

Se desarrollaron cinco endpoints en la API REST que permiten realizar operaciones CRUD sobre los datos almacenados en la base de datos. La correcta gestión de los códigos de estado HTTP y mensajes de error mejora la experiencia del usuario.