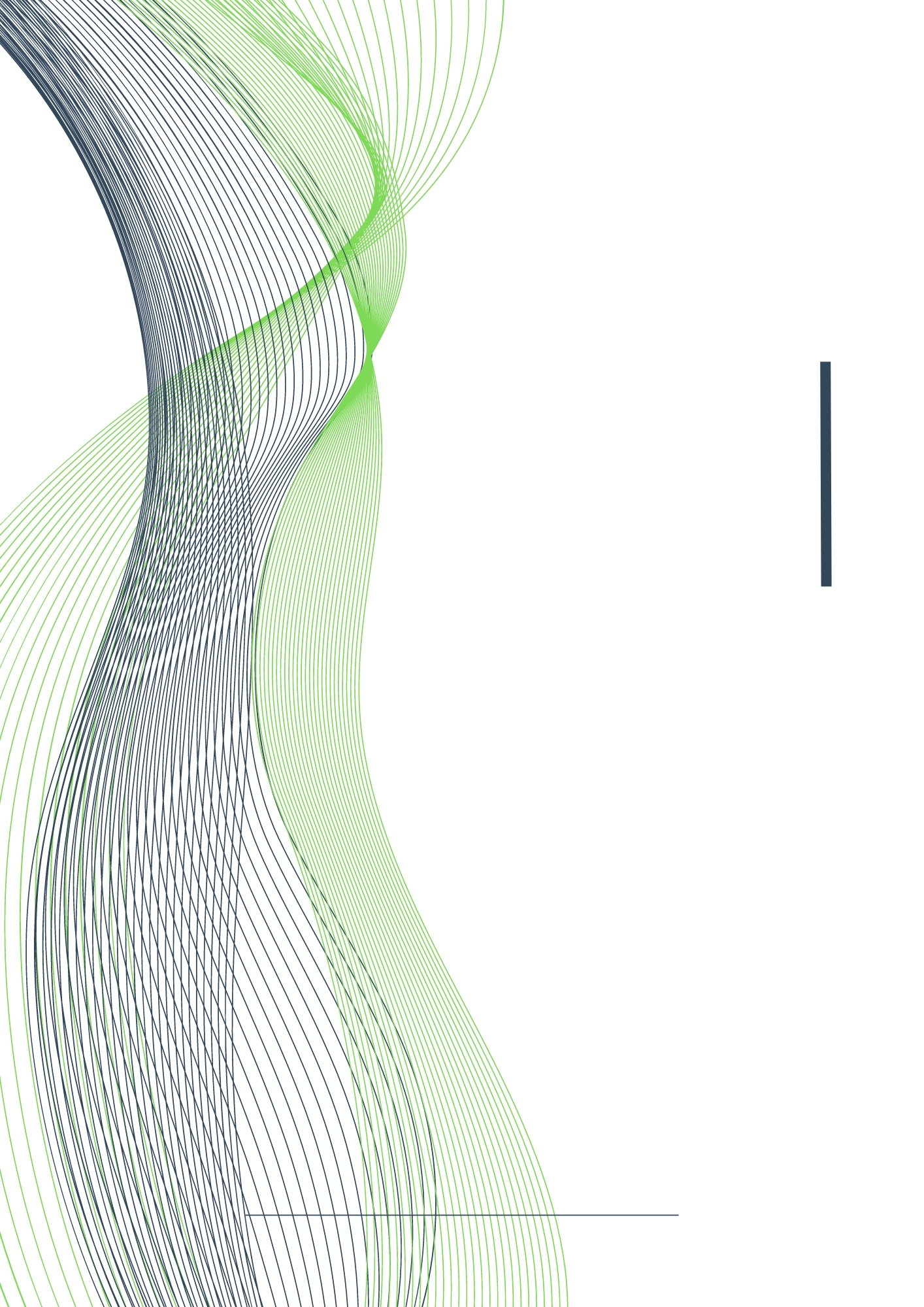
****

PROYECTO DE INTEGRACIÓN DE DATOS EN TIEMPO REAL DESDE   
UNA API HACIA   
SQL SERVER USANDO SSIS

**ING. Silva Parraguez Maximo**

1. **ENTENDIENDO LA DATA Y EL PROYECTO**

**Los datos en tiempo real se obtendrán desde la API de OpenWeatherMap**, que proporciona datos meteorológicos en tiempo real, como temperatura, humedad, velocidad del viento, etc.

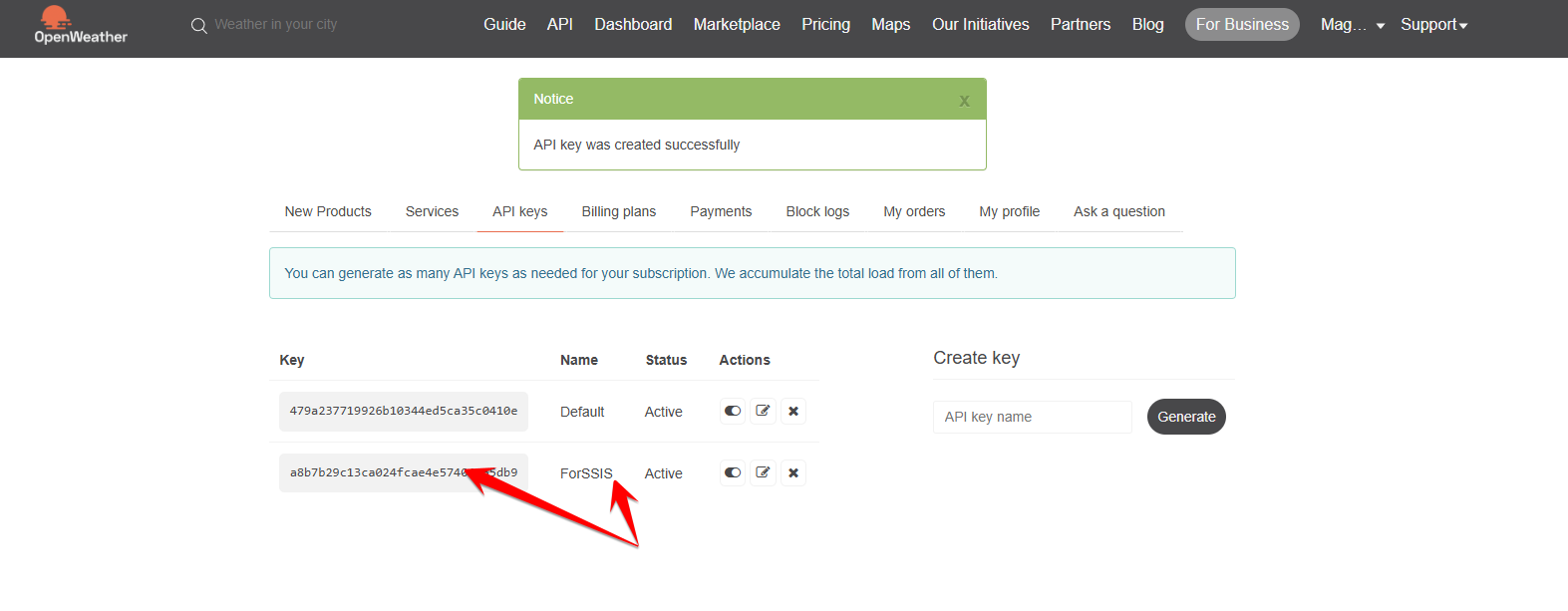
La finalidad de este proyecto es crear un proceso ETL en SSIS que capture, transforme y cargue datos en tiempo real desde una fuente de datos continua (como la API de OpenWeatherMap) de una ciudad en específico (o varias ciudades), a una base de datos SQL Server.

1. **DESARROLLO DEL PROYECTO**

**Paso 1: Obteniendo una API Key de OpenWeatherMap**

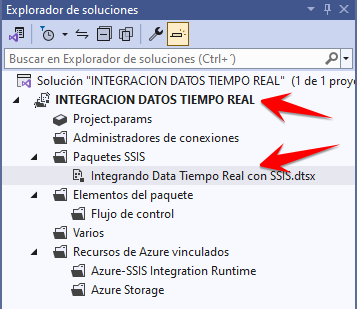
Como la data la extraeré de un API que pide una clave (API Key), necesitaré crear una cuenta en la Web que me la proporcionará, si hubiera otra API que no pidiera esa clave, por ejemplo más adelante, en el ***Paso 5: Editando el Script con el código C#,*** en lugar de pasarle la APIKey a la URL que construye la API, solo se le pasaría el enlace de la API directamente (Por ejemplo: <https://api.coindesk.com/v1/bpi/currentprice.json>) (API gratuita que brinda información del índice de precios de Bitcoin (BPI) en tiempo real).

Una vez registrado en la web de OpenWeatherMap (<https://openweathermap.org/api>), generamos una nueva API Key que utilizará para la obtención de los Datos.



**Paso 2: Creando Proyecto en SSIS**

Se creó el proyecto con Nombre: **“INTEGRACION DATOS TIEMPO REAL”** y un paquete SSIS llamado: **“Integrando Data Tiempo Real con SSIS”.**

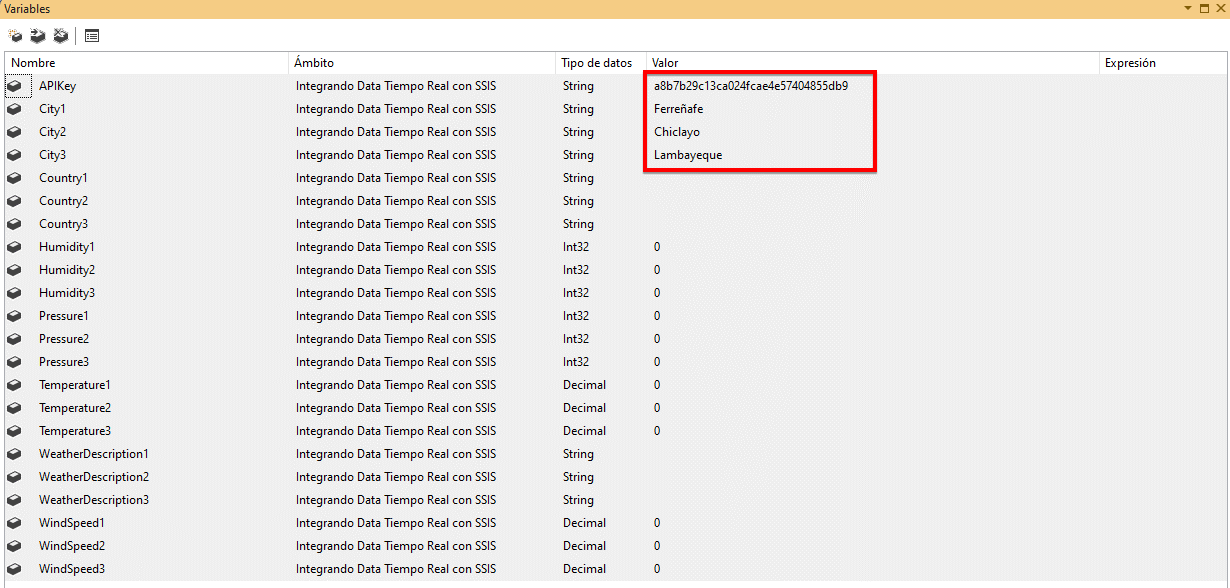


**Paso 3: Configurando Variables en SSIS**

Crearé variables que se encargarán de extraer y almacenar los datos obtenidos del API. Las variables son:

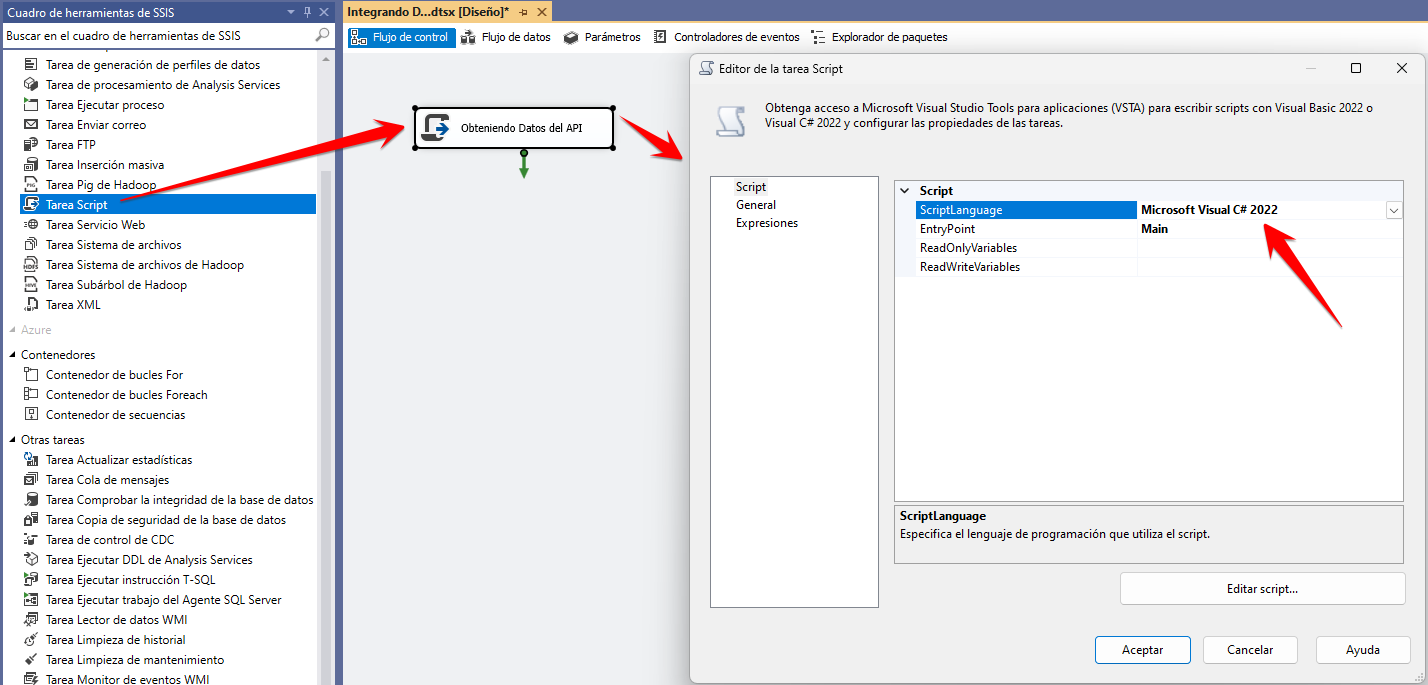
* **APIKey** (String): Almacenará el API Key de OpenWeatherMap que obtuve creando la cuenta.
* **City1** (String): Almacenará el nombre de una ciudad en específico (para este proyecto, "Ferreñafe").
* **City2** (String): Almacenará el nombre de una ciudad en específico (para este proyecto, "Chiclayo").
* **City3** (String): Almacenará el nombre de una ciudad en específico (para este proyecto, "Lambayeque").
* **Country1** (String): Almacenará el país que pertenece la ciudad 1.
* **Country2** (String): Almacenará el país que pertenece la ciudad 2.
* **Country3** (String): Almacenará el país que pertenece la ciudad 3.
* **Humidity1** (Int): Almacenará la humedad obtenida de la ciudad 1.
* **Humidity2** (Int): Almacenará la humedad obtenida de la ciudad 2.
* **Humidity3** (Int): Almacenará la humedad obtenida de la ciudad 3.
* **Pressure1** (Int): Para la presión atmosférica obtenida de la ciudad 1.
* **Pressure2** (Int): Para la presión atmosférica obtenida de la ciudad 2.
* **Pressure3** (Int): Para la presión atmosférica obtenida de la ciudad 3.
* **Temperature1** (Decimal): Almacenará la temperatura obtenida de la ciudad 1.
* **Temperature2** (Decimal): Almacenará la temperatura obtenida de la ciudad 2.
* **Temperature3** (Decimal): Almacenará la temperatura obtenida de la ciudad 3.
* **WeatherDescription1** (String): Para la descripción del clima obtenida de la ciudad 1.
* **WeatherDescription2** (String): Para la descripción del clima obtenida de la ciudad 2.
* **WeatherDescription3** (String): Para la descripción del clima obtenida de la ciudad 3.
* **WindSpeed1** (Decimal): Para la velocidad del viento obtenida de la ciudad 1.
* **WindSpeed2** (Decimal): Para la velocidad del viento obtenida de la ciudad 2.
* **WindSpeed3** (Decimal): Para la velocidad del viento obtenida de la ciudad 3.

**Solo** **APIKey, City1, City2 y City3 tienen asignado un “Valor”**, las demás variables lo dejaré vacío ya que se llenará en el Script. Si se requiere cambiar las ciudades se pueden cambiar modificando este apartado.



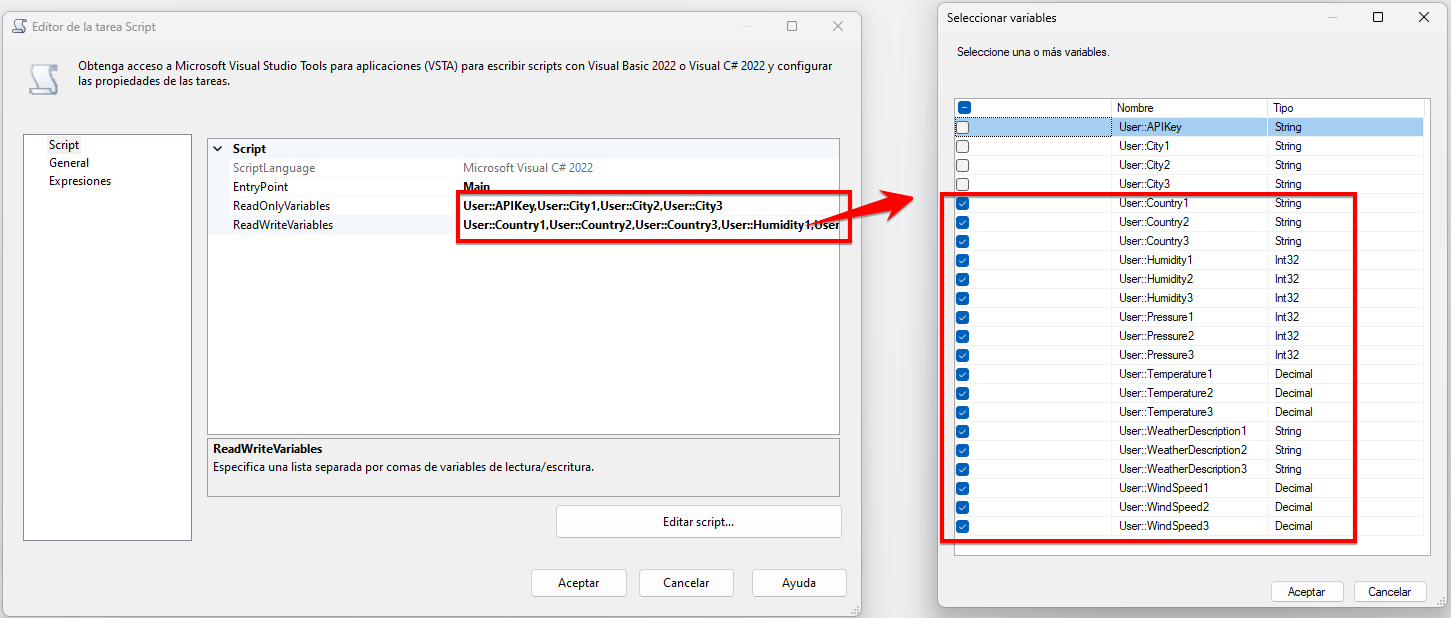
**Paso 4: Componente de Tarea de Scripts.**

Emplearé el componente de “Tarea Script” y utilizaré C# como lenguaje de programación para la configuración de la obtención de datos desde la API.



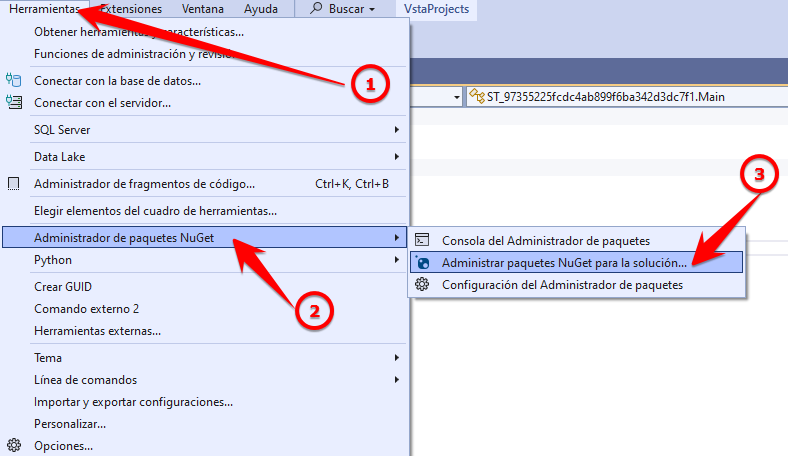
Se agregan las Variables al Script:

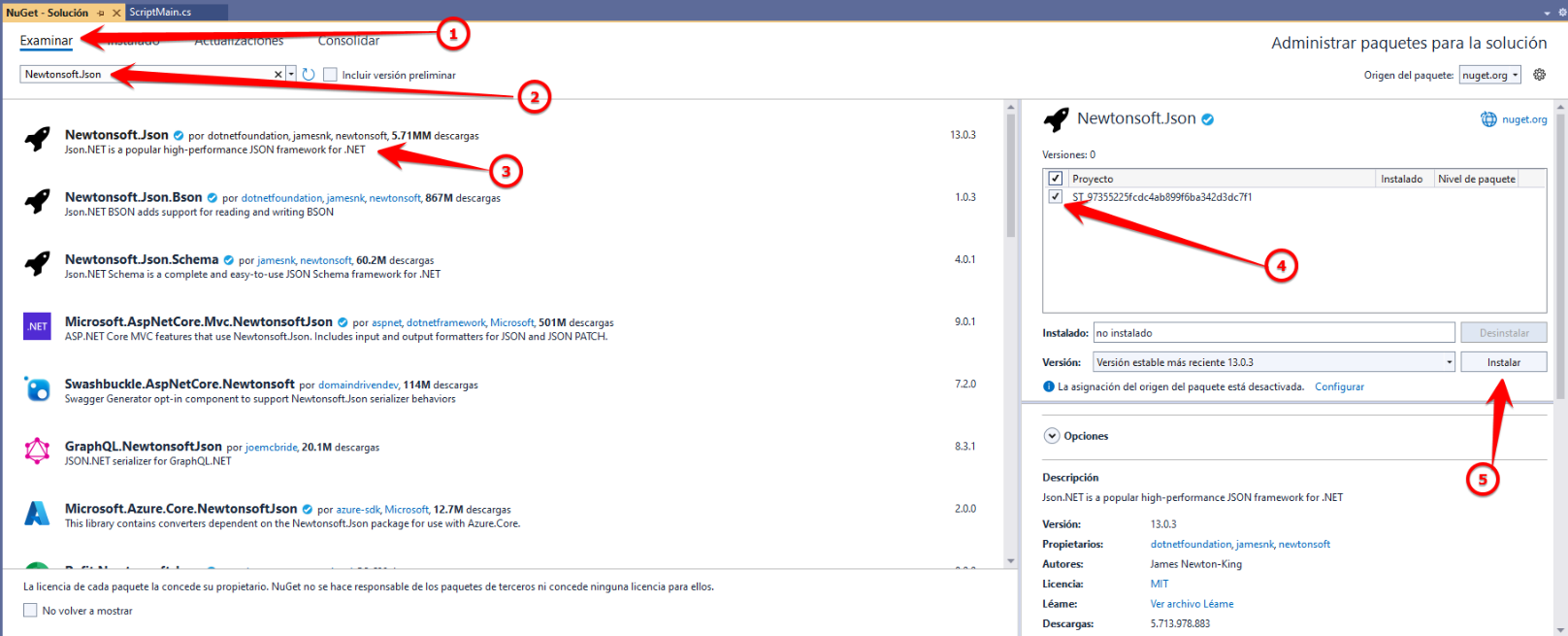
* En la pestaña "**ReadOnlyVariables**", seleccioné las variables User::APIKey, User::City1, User::City2, User::City3. (Las variables que le coloqué un valor.)
* En la pestaña "**ReadWriteVariables**", seleccioné las demas variables variables (las demás variables que no se le colocó ningún Valor).

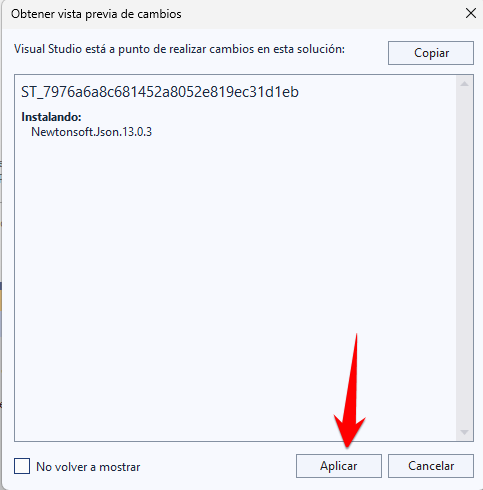


**Paso 5: Editando el Script con el código C#.**

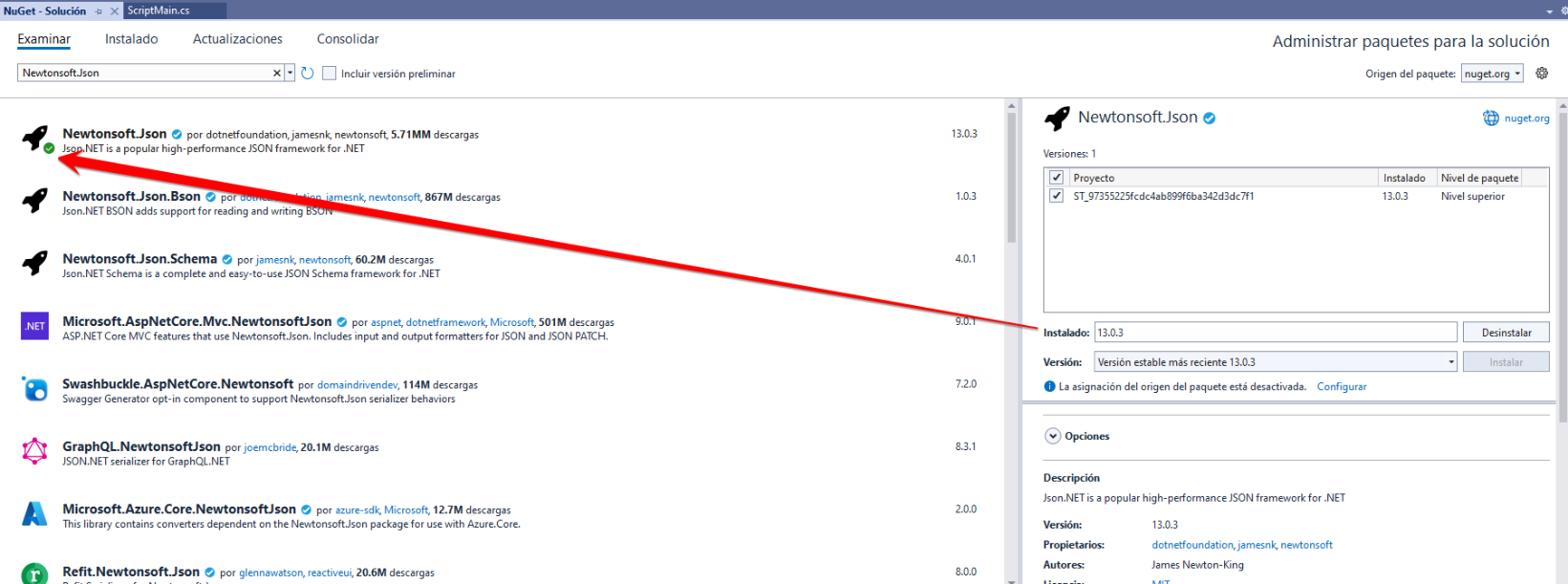
Primero instalaré una librería que usaré en el código, la librería es **Newtonsoft.Json:**



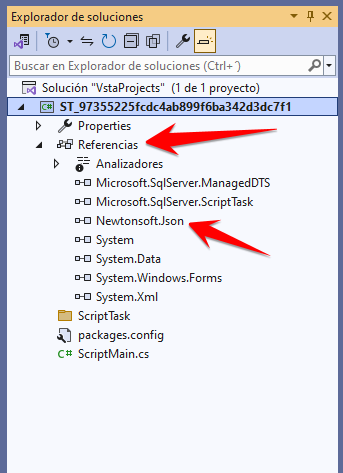




Se instaló correctamente:

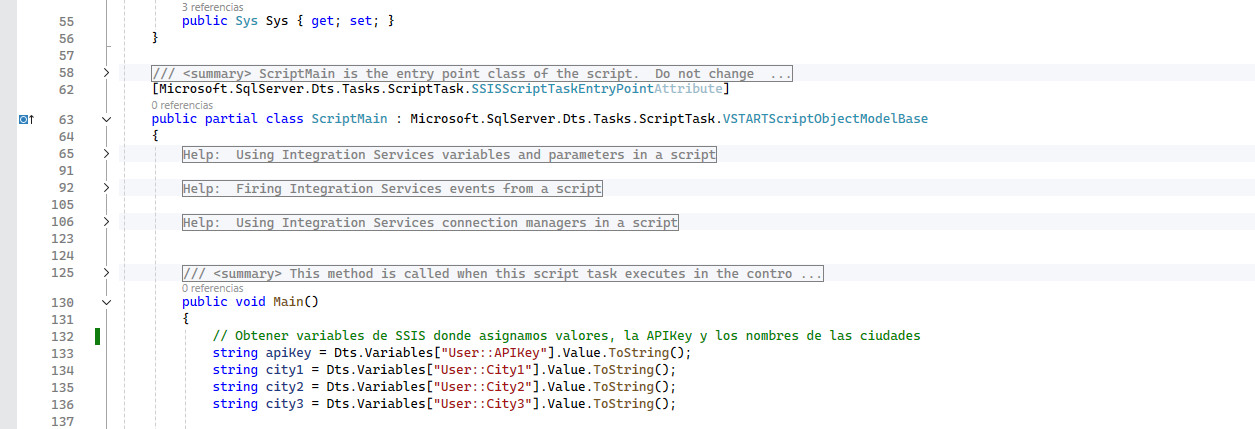


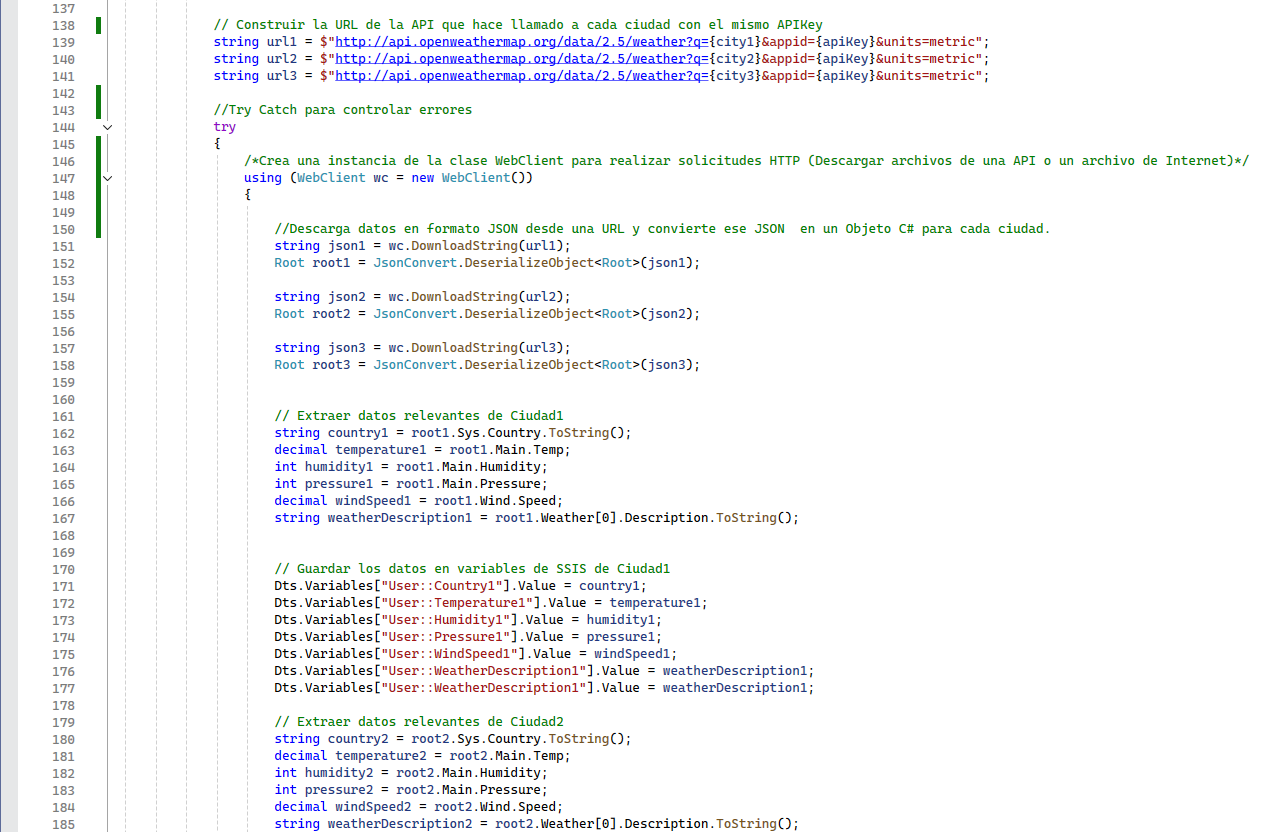
Verificamos que aparezca referenciado la librería que se agregó:

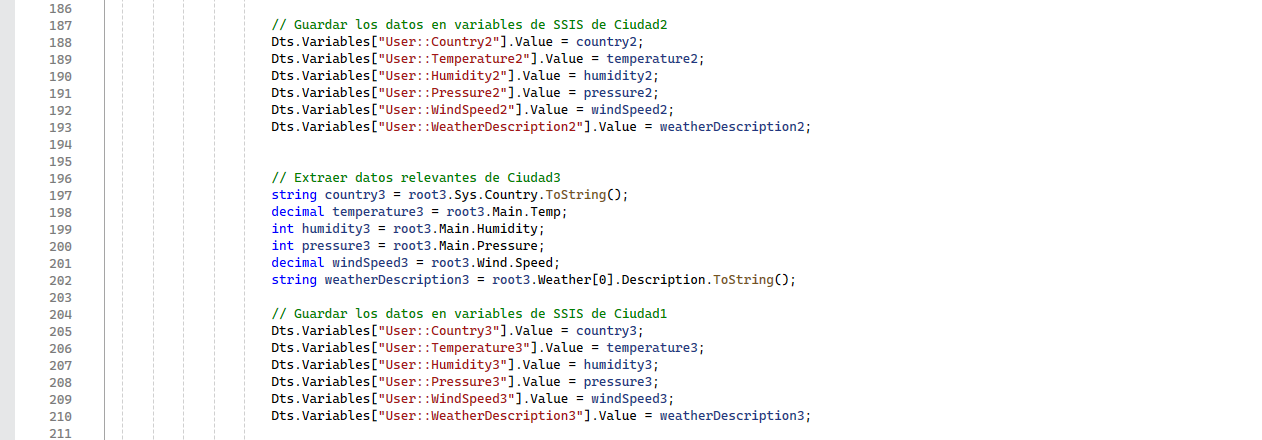


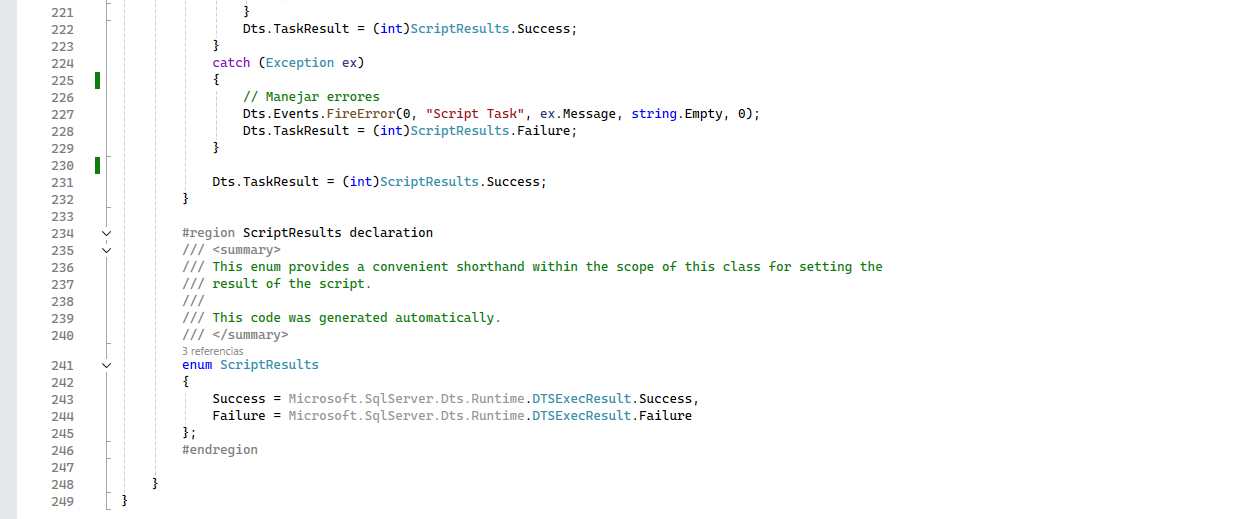
Luego escribí el código que llama a la API de OpenWeatherMap usando la API Key y 3 Ciudades, creamos las clases teniendo en cuando el JSON, luego extrae la temperatura, la humedad, la presión atmosférica, la velocidad del viento y la descripción del clima de cada ciudad brindada del JSON de respuesta y guarda los valores en las variables creadas en el SSIS.





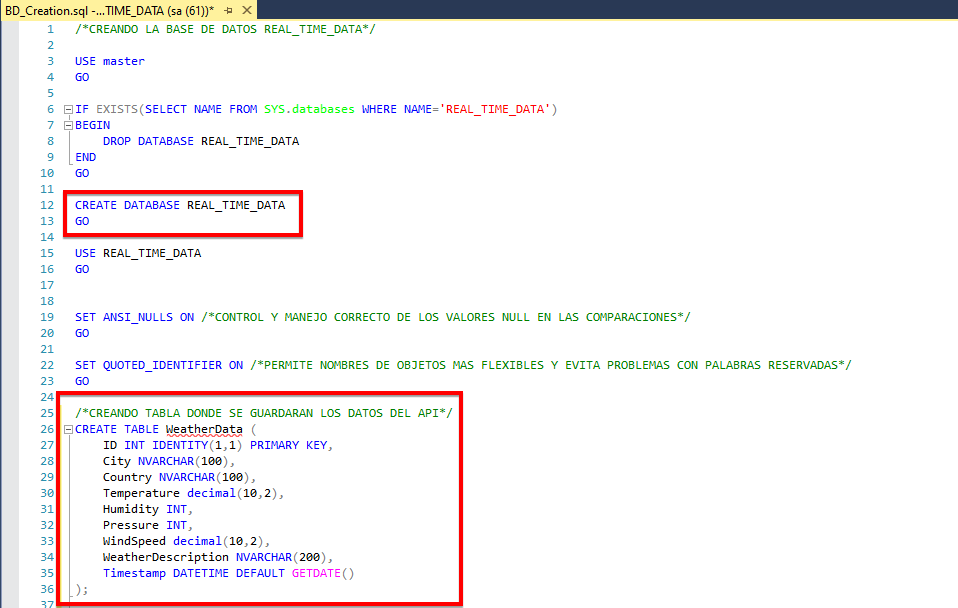






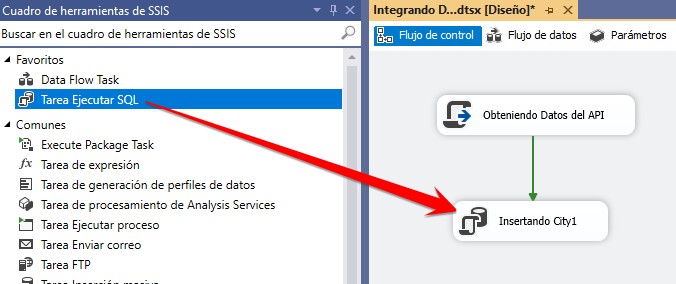
**Paso 6: Creando Tabla para almacenar los Datos.**

Crearé una Base de datos y una tabla en SQL Server donde se irán almacenando los datos extraídos del API:

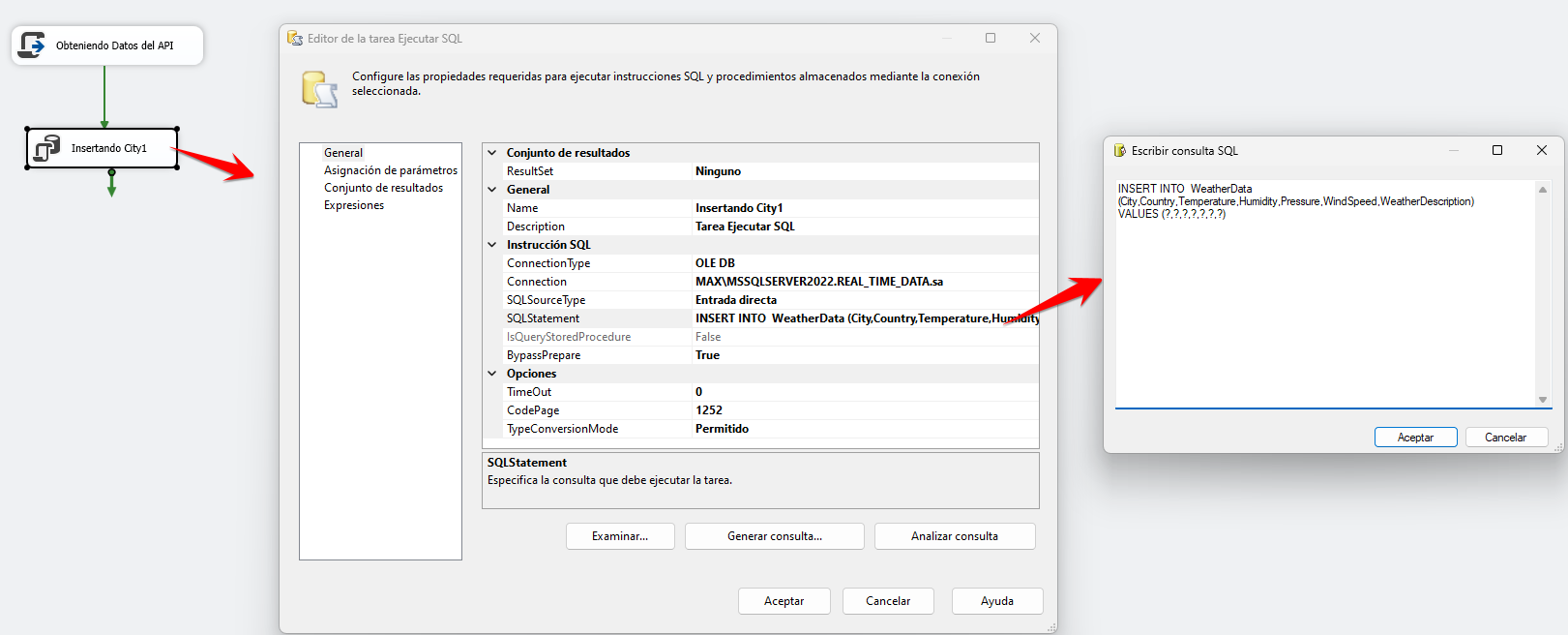


**Paso 7: Creando una Tarea Ejecutar SQL para insertar los datos a la Base de Datos.**

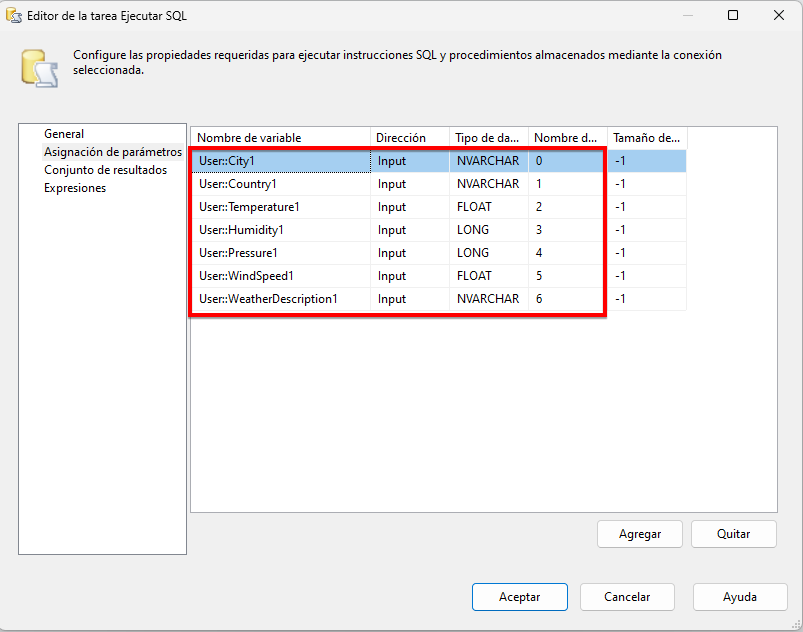
Ahora para el proceso de la carga de datos del JSON a una tabla en SQL Server utilicé el componente Tarea Ejecutar SQL para insertar los datos correspondientes a cada Ciudad.



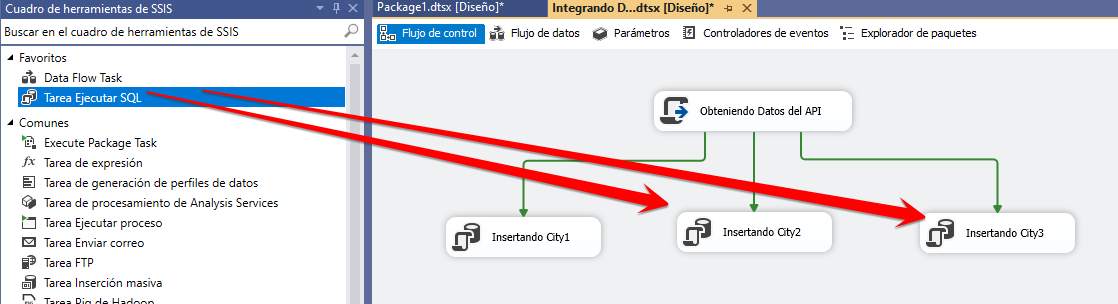
Agregué la conexión a la Base de Datos y el comando SQL donde le pasaré las variables.

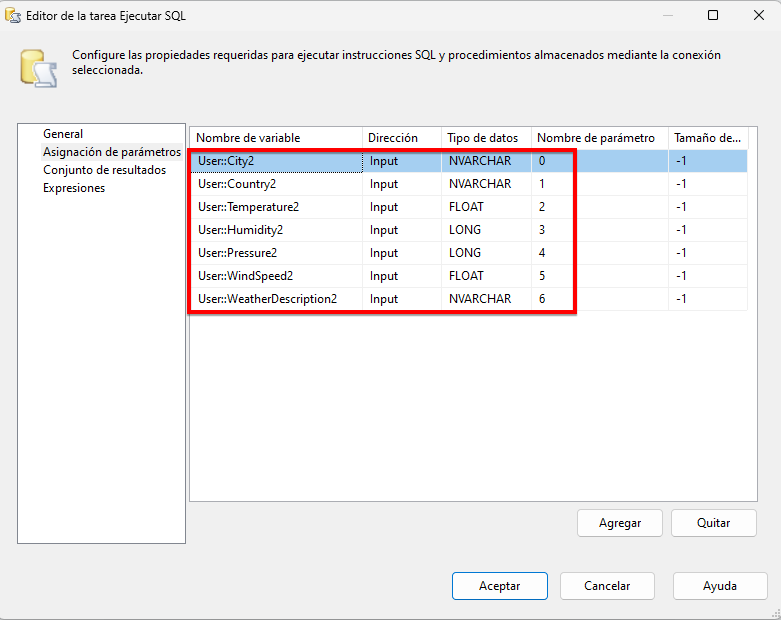
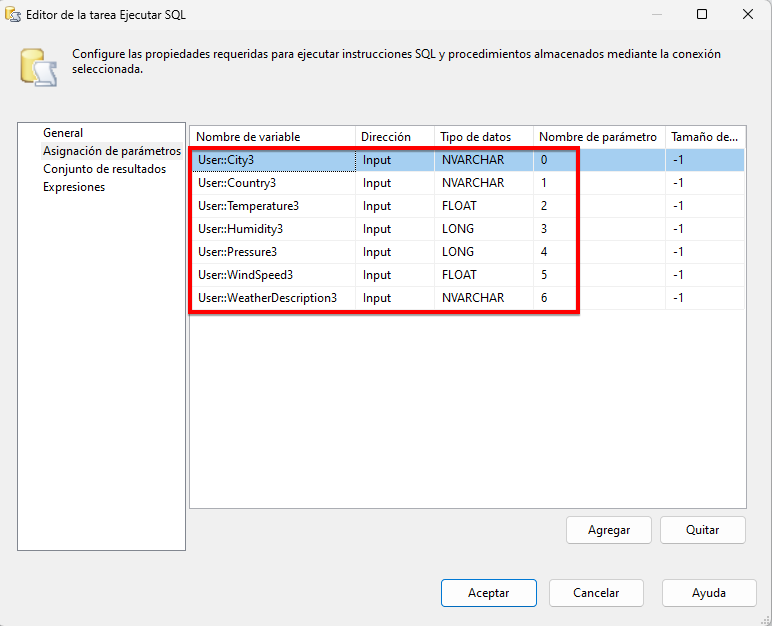


En Asignación de Variables colocamos todo lo relacionado a datos obtenidos para la ciudad 1.



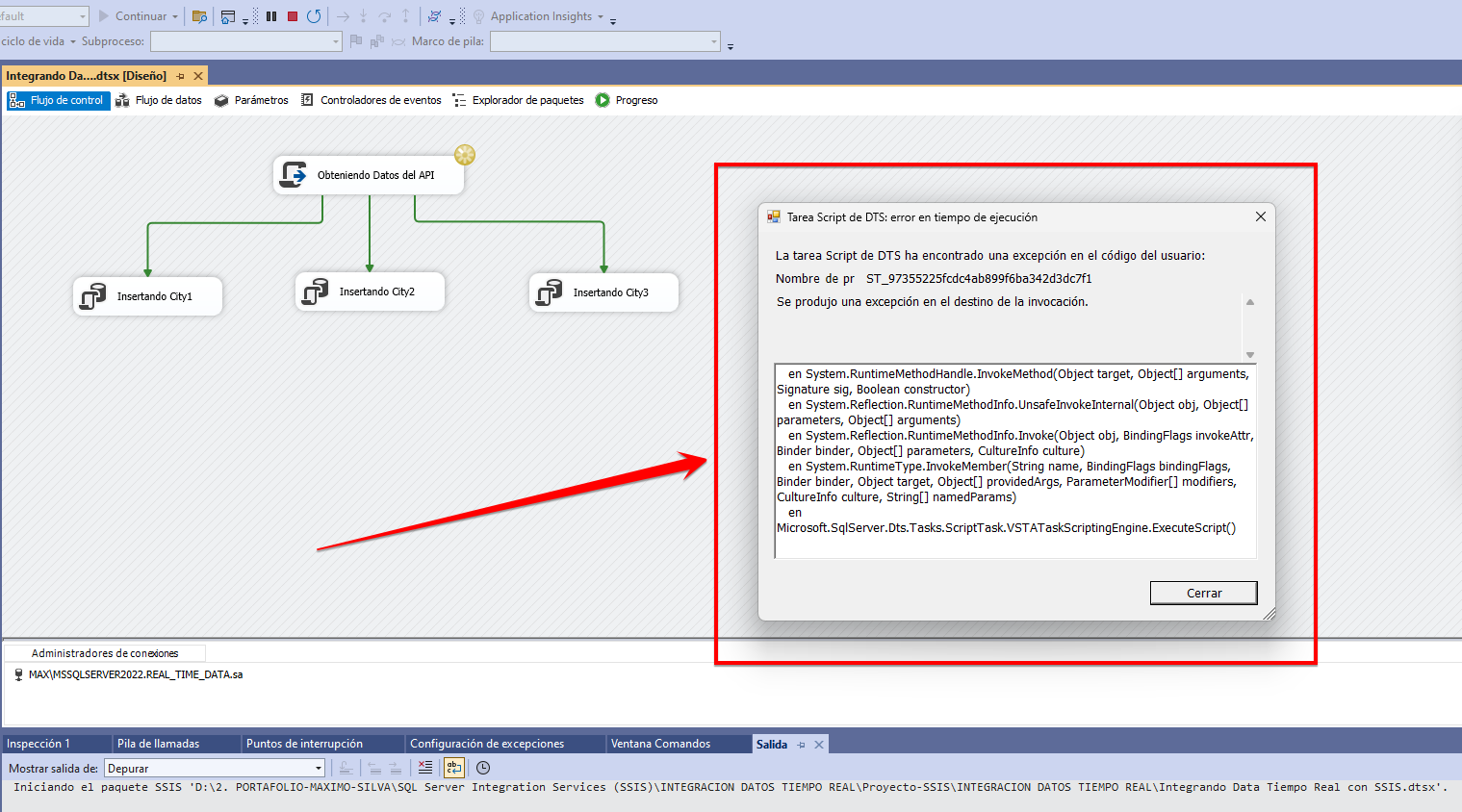
Se hizo el mismo proceso para las demás ciudades:

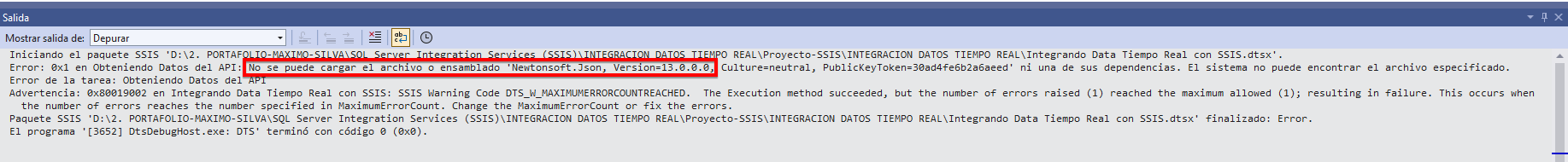


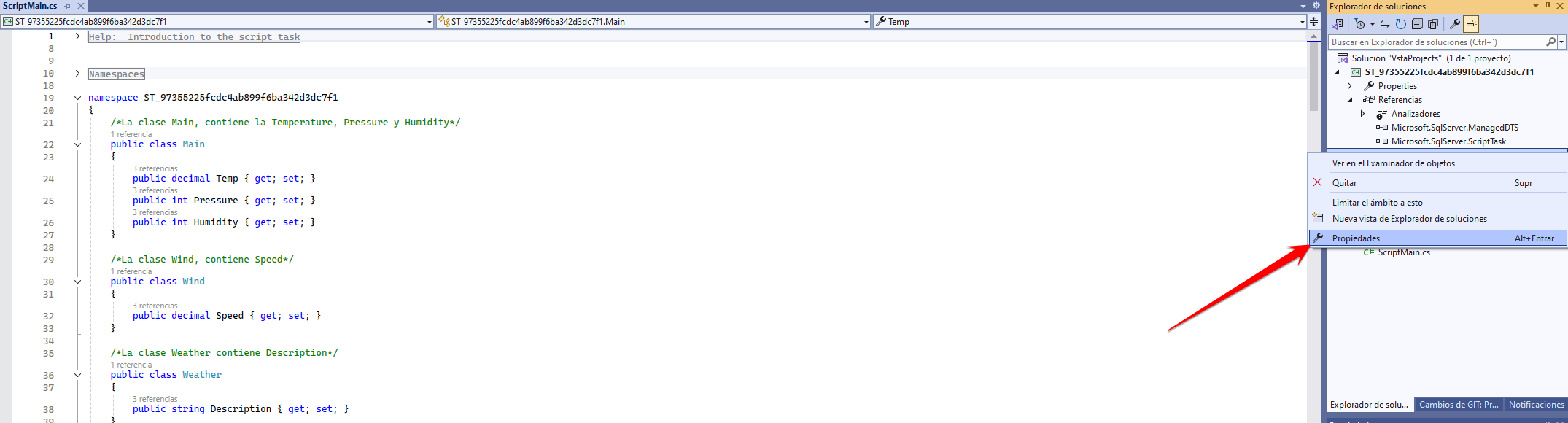
**Paso 8: Error en la Ejecución y la solución al error.**

Al dar inicio a la ejecución del proyecto **me salió el siguiente error**:

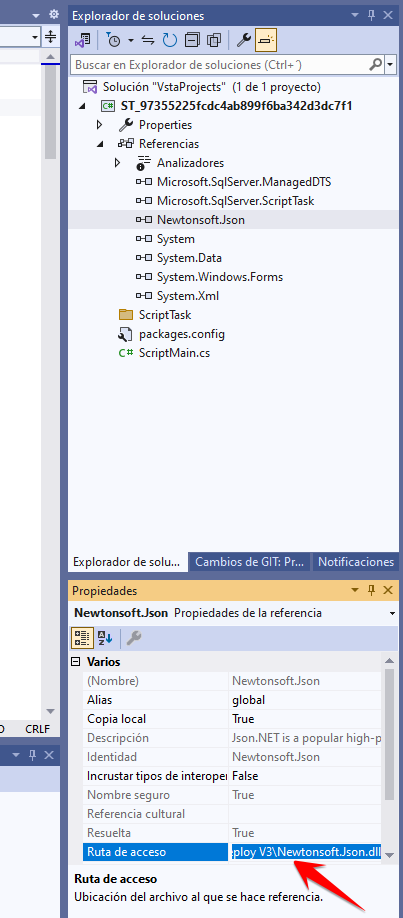




Estuve investigando y este error indica que la referencia a la librería **Newtonsoft.Json** no está registrada en la **GAC (Global Assembly Cache)** por lo que se necesita registrarla para que pueda se ubicada por el proyecto y así ser usada, para ello abrimos nuevamente donde se escribió el código C# para buscar la ruta donde está la librería descargada:



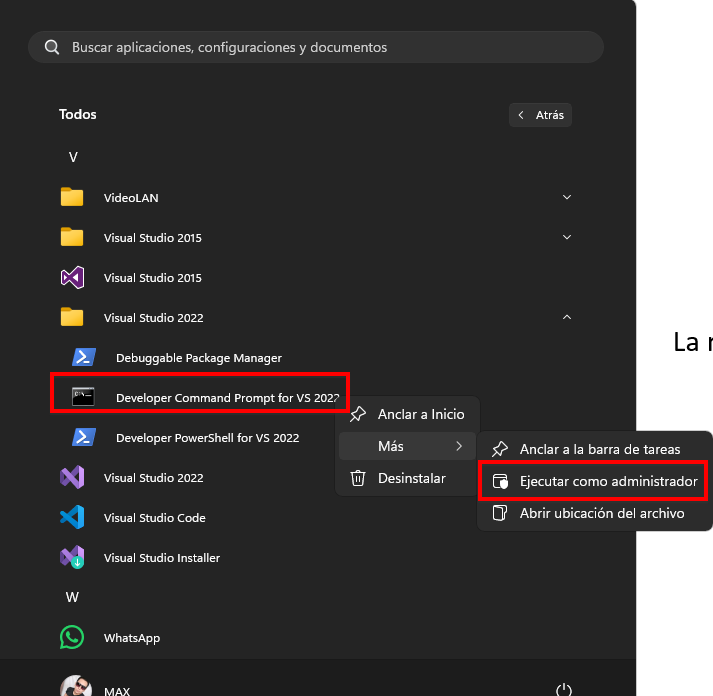
Identifiqué la Ruta de la librería y la copié:



La ruta donde se descargó fue:

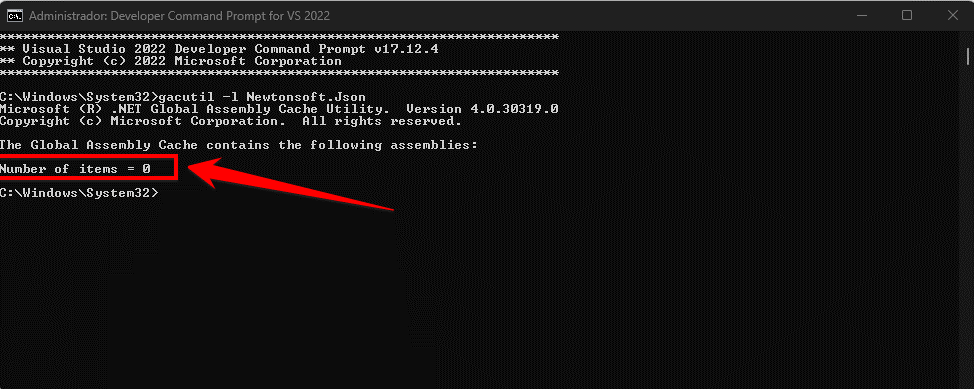
**C:\Program Files\IIS\Microsoft Web Deploy V3\Newtonsoft.Json.dll**

Luego abrí **Developer Command Prompt for VS 2022** y lo ejecuté como Administrador.



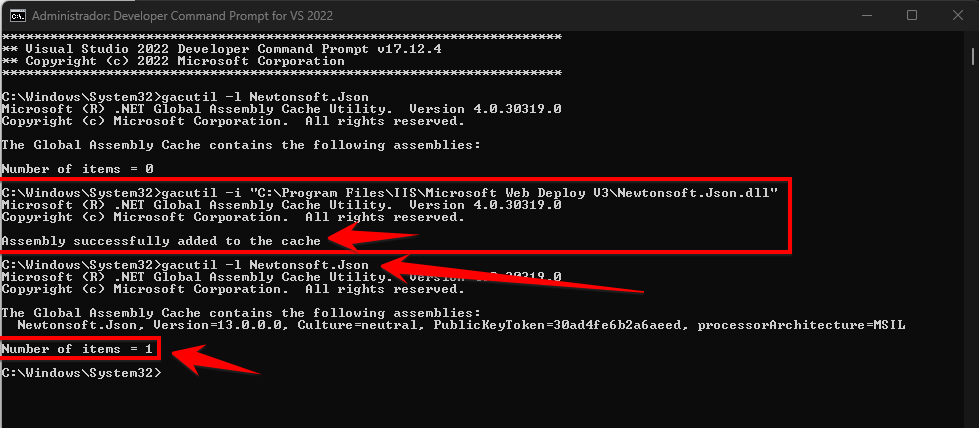
En la pantalla de comandos verificamos que no se encuentra registrada la Librería:

**gacutil -l Newtonsoft.Json**



El número 0 indica que no se tiene librerías registradas en el **GAC**, para referenciarlo **de forma global** (No solo para el proyecto, por eso se usa **gacutil**):

**gacutil -i "C:\Program Files\IIS\Microsoft Web Deploy V3\Newtonsoft.Json.dll"**



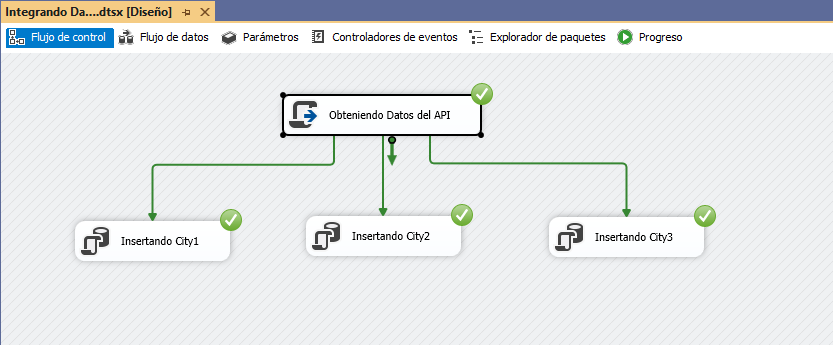
Al volver a verificar la librería, ya aparece el número 1, que indica que, si está registrado una librería, en este caso, Newtonsoft.Json. **Ya se debería haber solucionado el problema.**

Si por alguna razón requerimos borrar referencia a la librería, se hace con el siguiente comando:

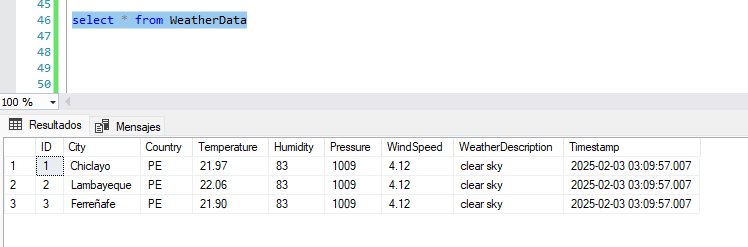
**gacutil -u Newtonsoft.Json**

**Paso 9: Ejecutando el proyecto.**

Una vez solucionado el error, al ejecutar el proyecto, lo hace sin ningún inconveniente:

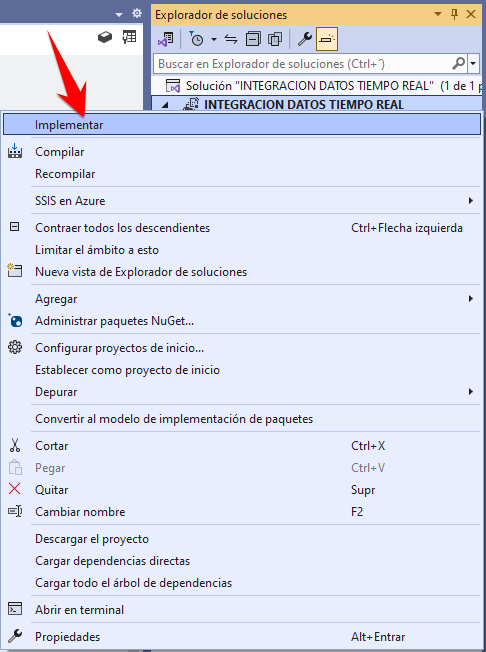
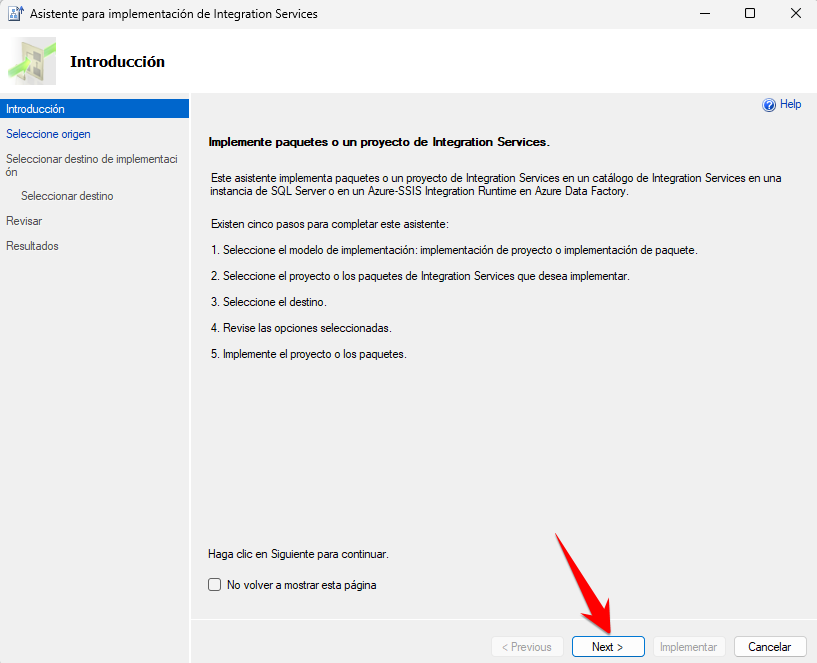


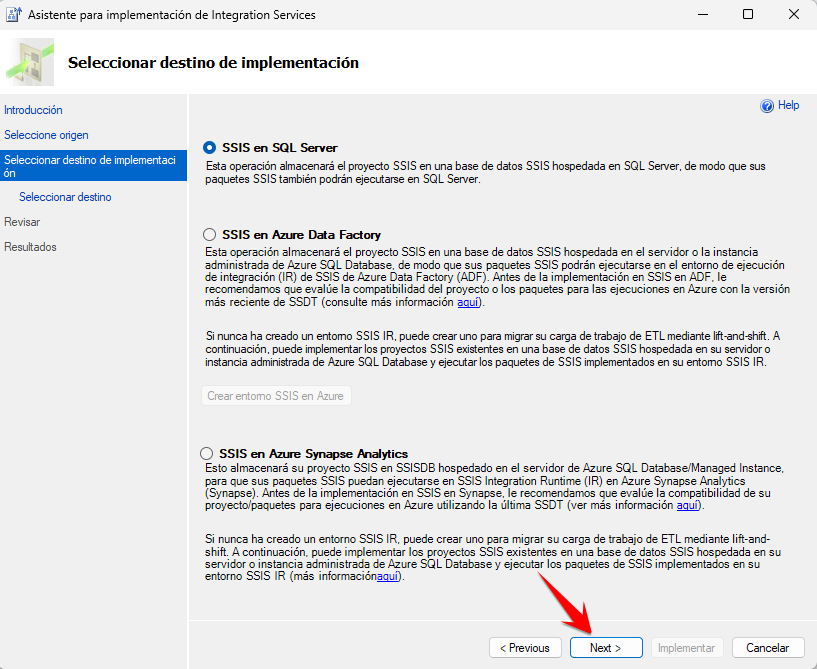
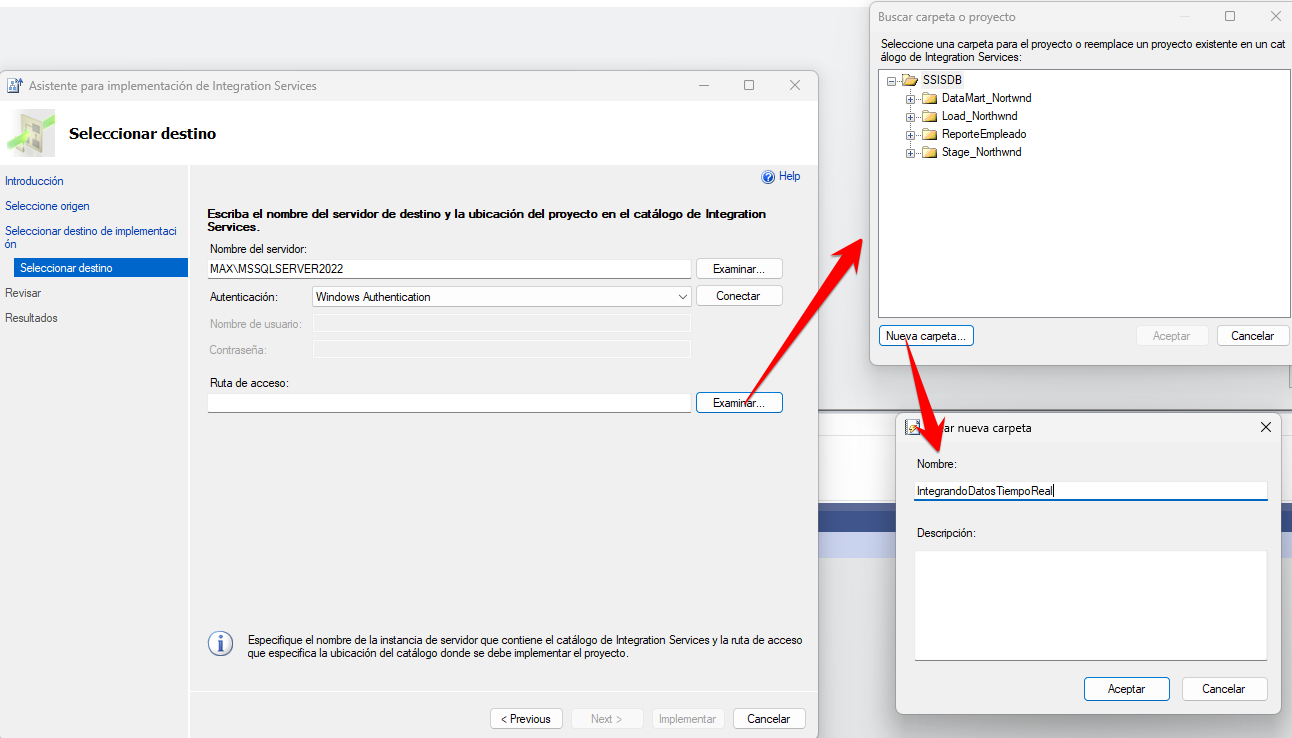
Verifico que se registraron los datos en la Base de datos:

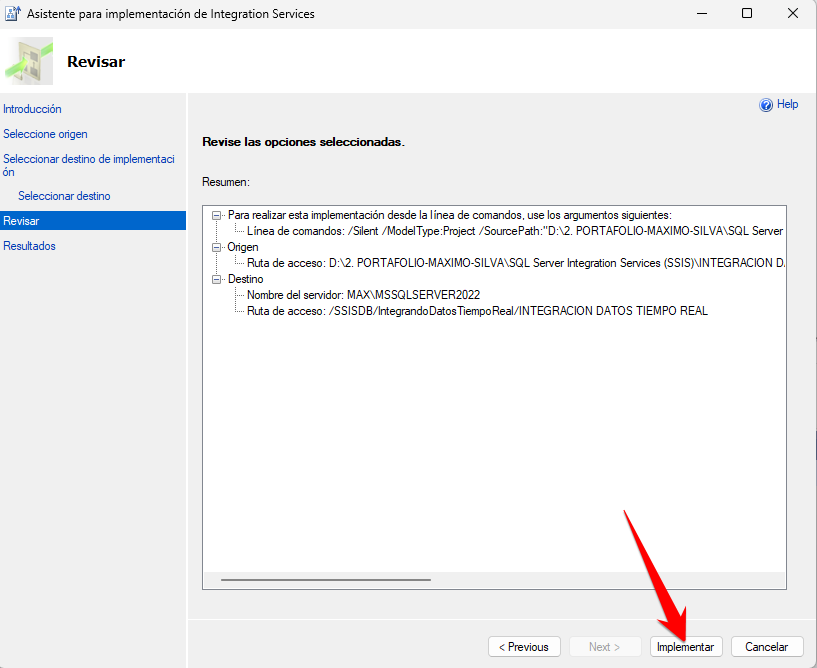
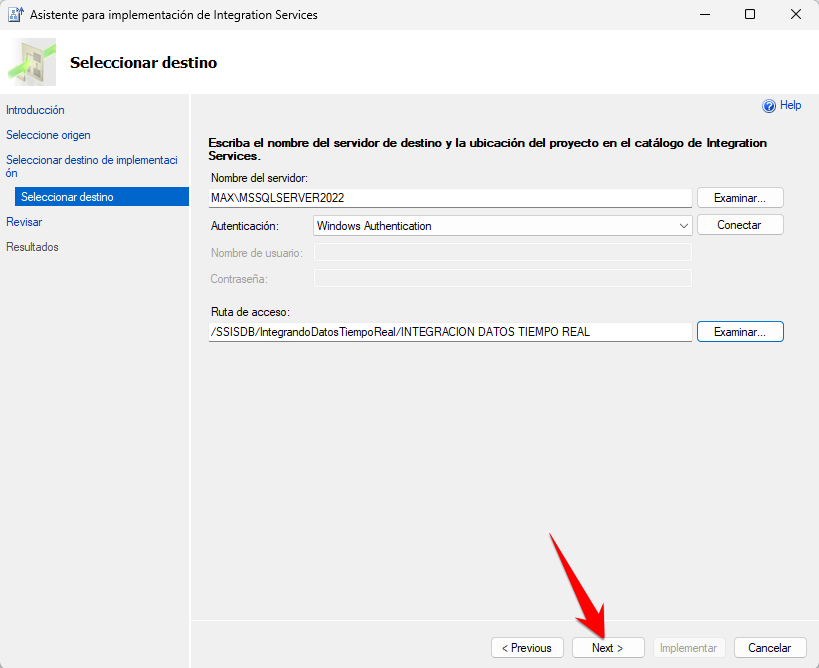


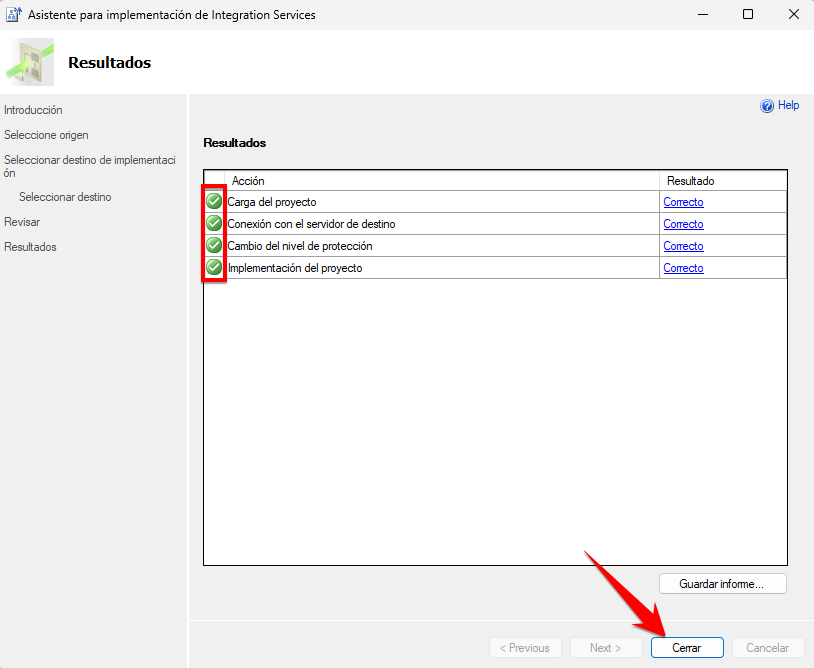
**Paso 10: Implementando el proyecto de Integration Services.**

Una vez tenemos el proyecto, empezaré con el despliegue para que aparezca en SQL Server y poder crear una Tarea Programada en el Agente de SQL Server, para ello lo implementé:

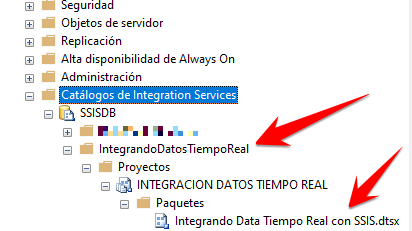
 



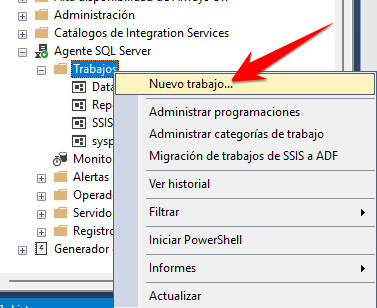
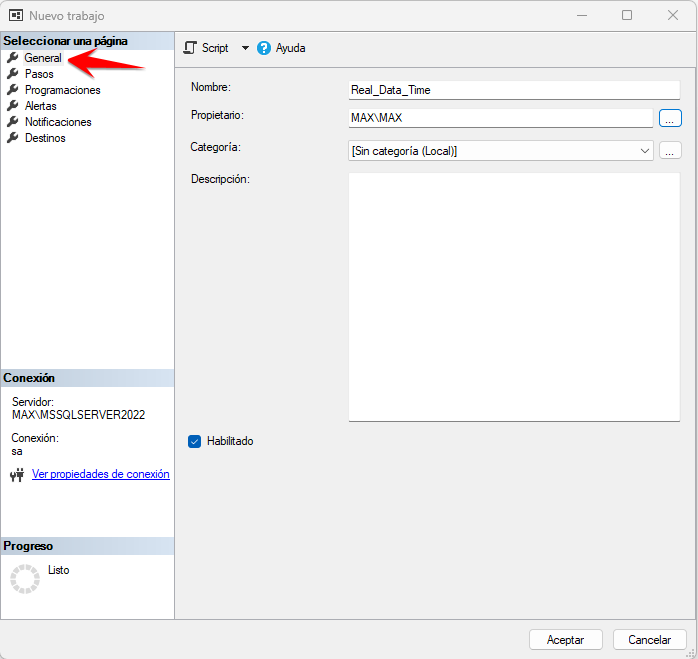


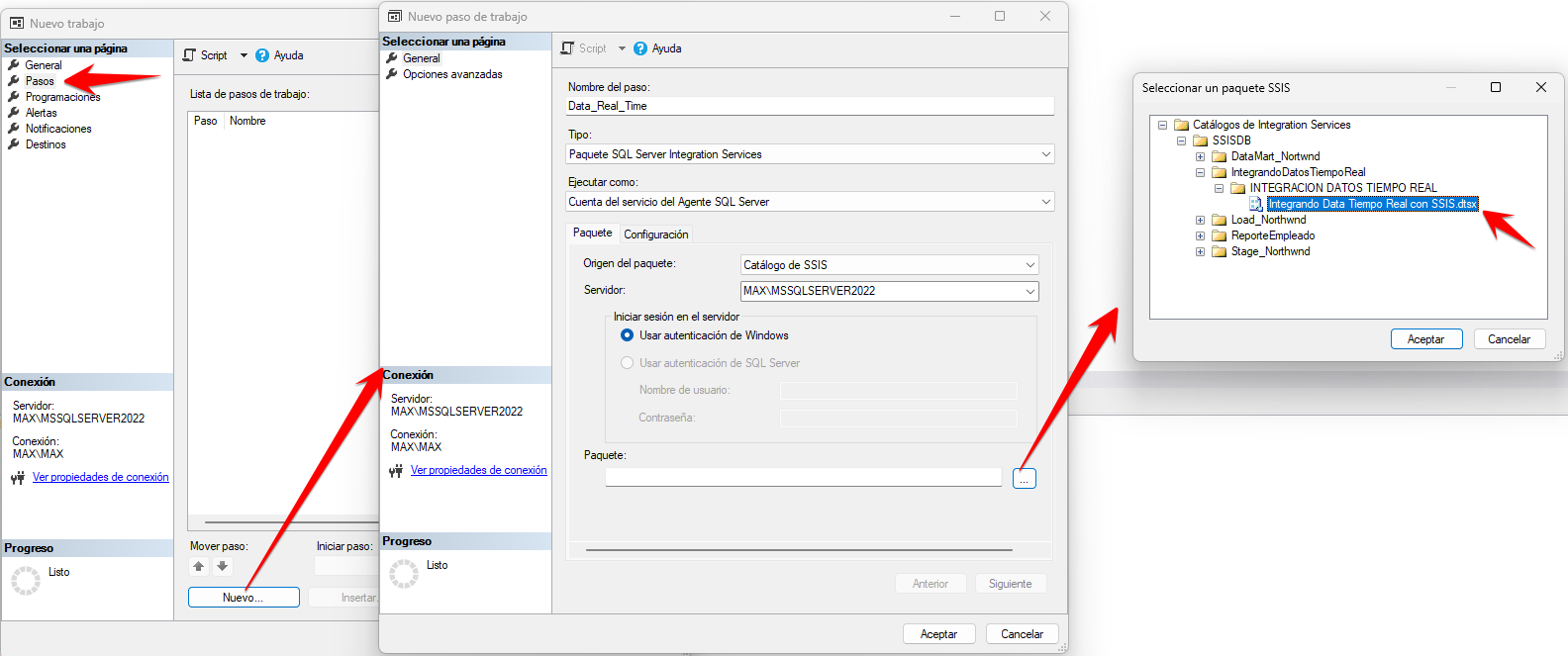
En SQL Server, en el apartado de Catálogos de Integration Services (Previamente se tuvo que crear un nuevo catálogo), ya aparece el proyecto:



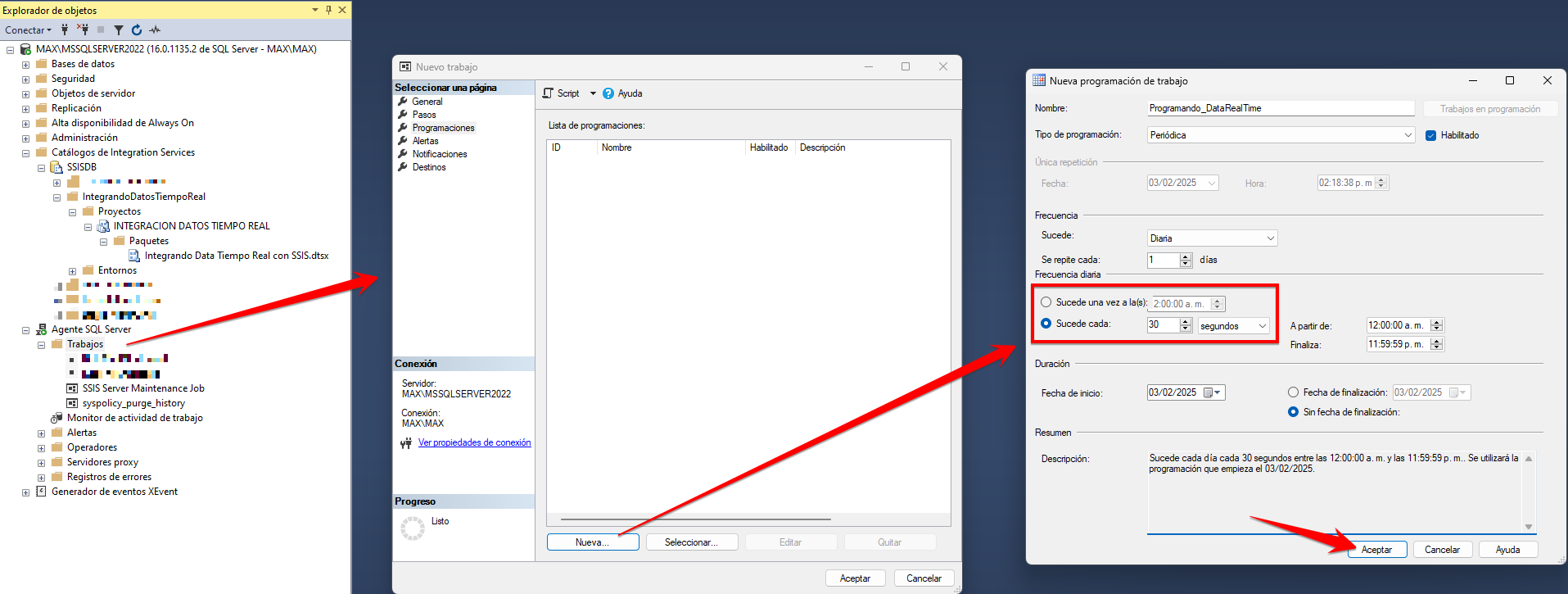
**Paso 11: Creando una tarea programada automática con Agente SQL Server.**

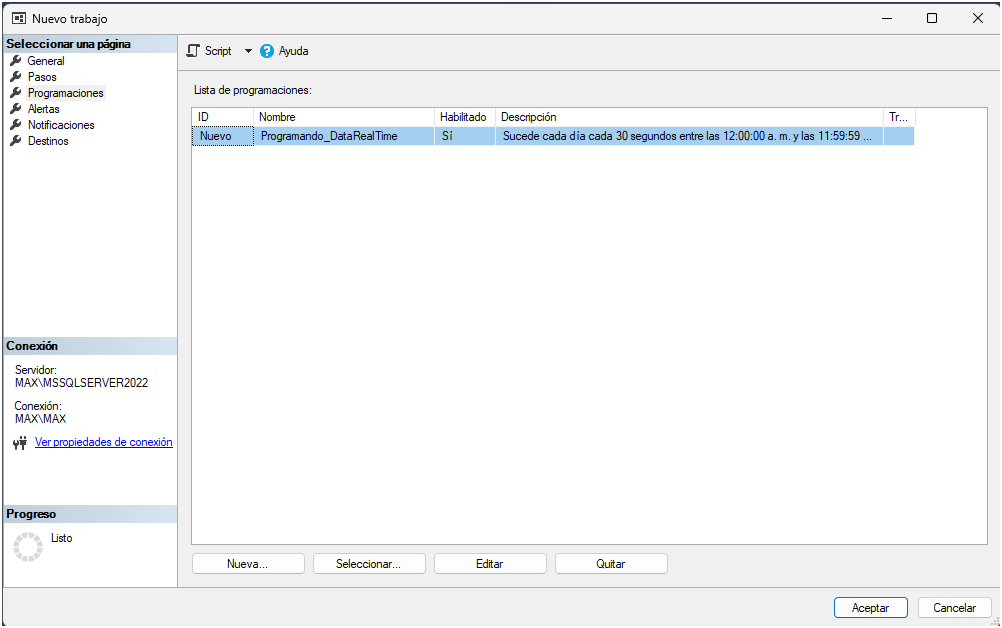
Cree un nuevo Job (trabajo) en SQL Server que ejecute el proyecto Implementado anteriormente:



Programando para que se ejecute automáticamente cada 30 segundos:

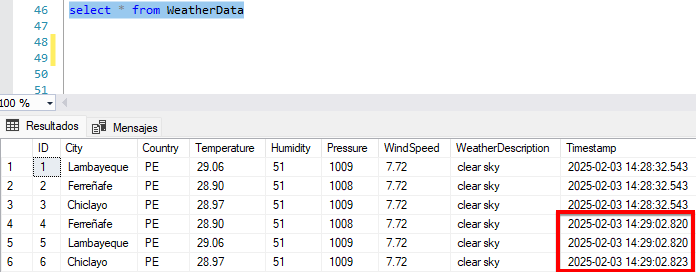




Se ejecutará automáticamente, se observa la hora de la primera ejecución:



En la segunda ejecución se visualiza que sucedió 30 segundos después tal como lo programé, pero se puede reducir ese tiempo si se desea:



De esta manera doy por concluido el proyecto donde se extrajo información de una API y se insertó en una Base de datos en SQL Server usando SSIS y que se ejecutará cada 30 segundos automáticamente.