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

ING. Silva Parraguez Maximo

I. 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.

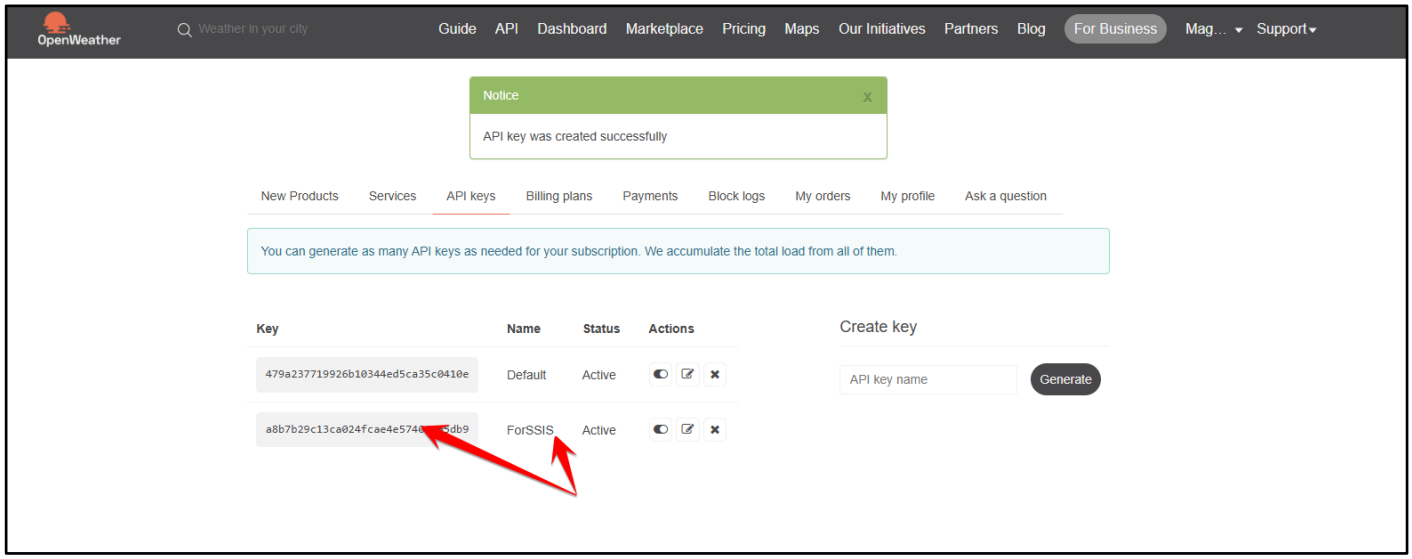
La finalidad de este proyecto es crear un proceso ETL en SSIS que capture, transforme y cargue datos en tiempo real desde una fuente de datos continua (como la API de OpenWeatherMap) de una ciudad en específico (o varias ciudades), a una base de datos SQL Server.

II. DESARROLLO DEL PROYECTO

Paso 1: Obteniendo una API Key de OpenWeatherMap

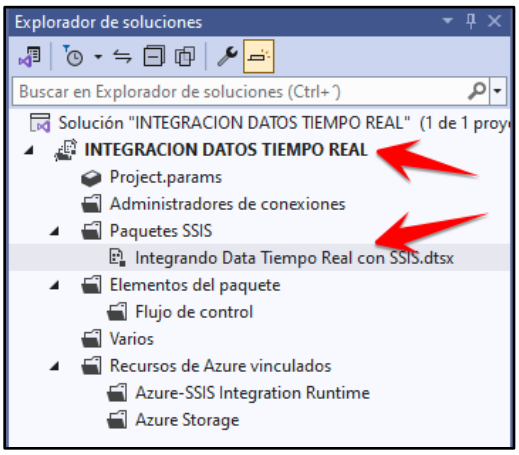
Como la data la extraeré de un API que pide una clave (API Key), necesitaré crear una cuenta en la Web que me la proporcionará, si hubiera otra API que no pidiera esa clave, por ejemplo más adelante, en el ***Paso 5: Editando el Script con el código C#,*** en lugar de pasarle la APIKey a la URL que construye 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".**

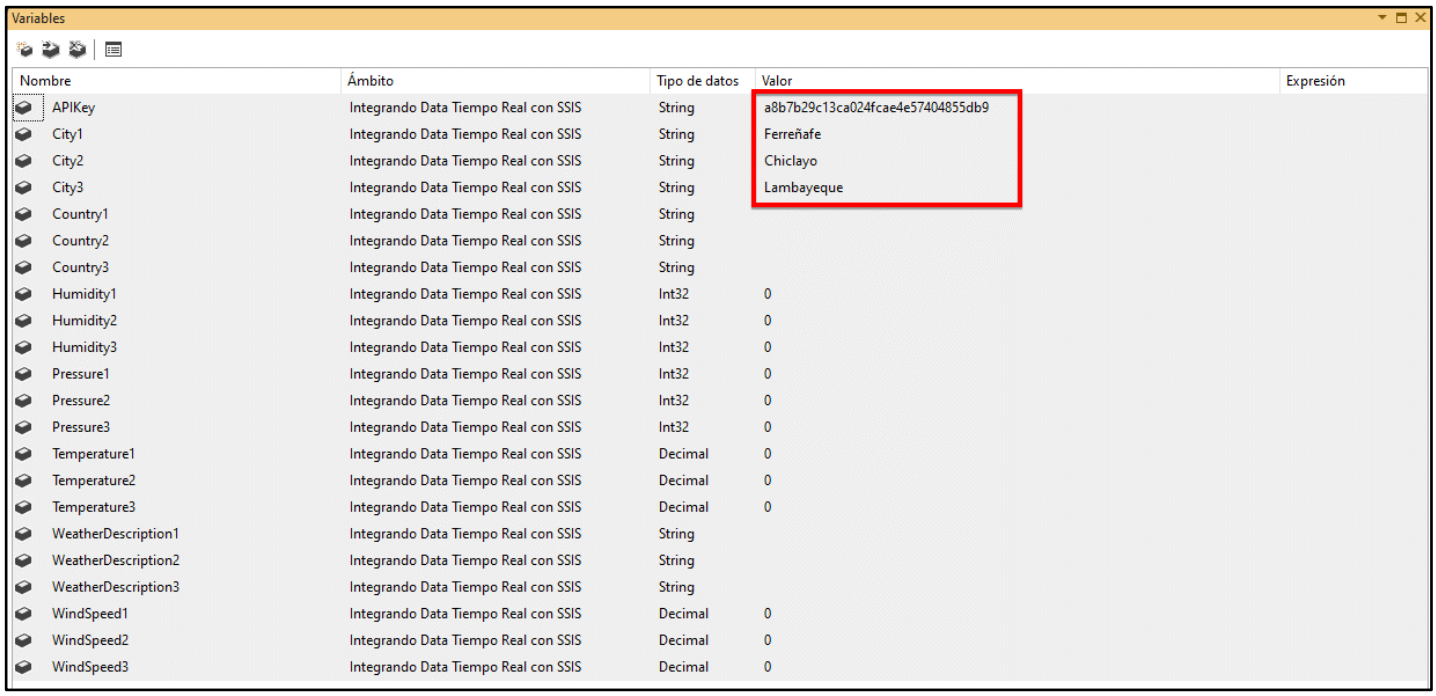


Paso 3: Configurando Variables en SSIS

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

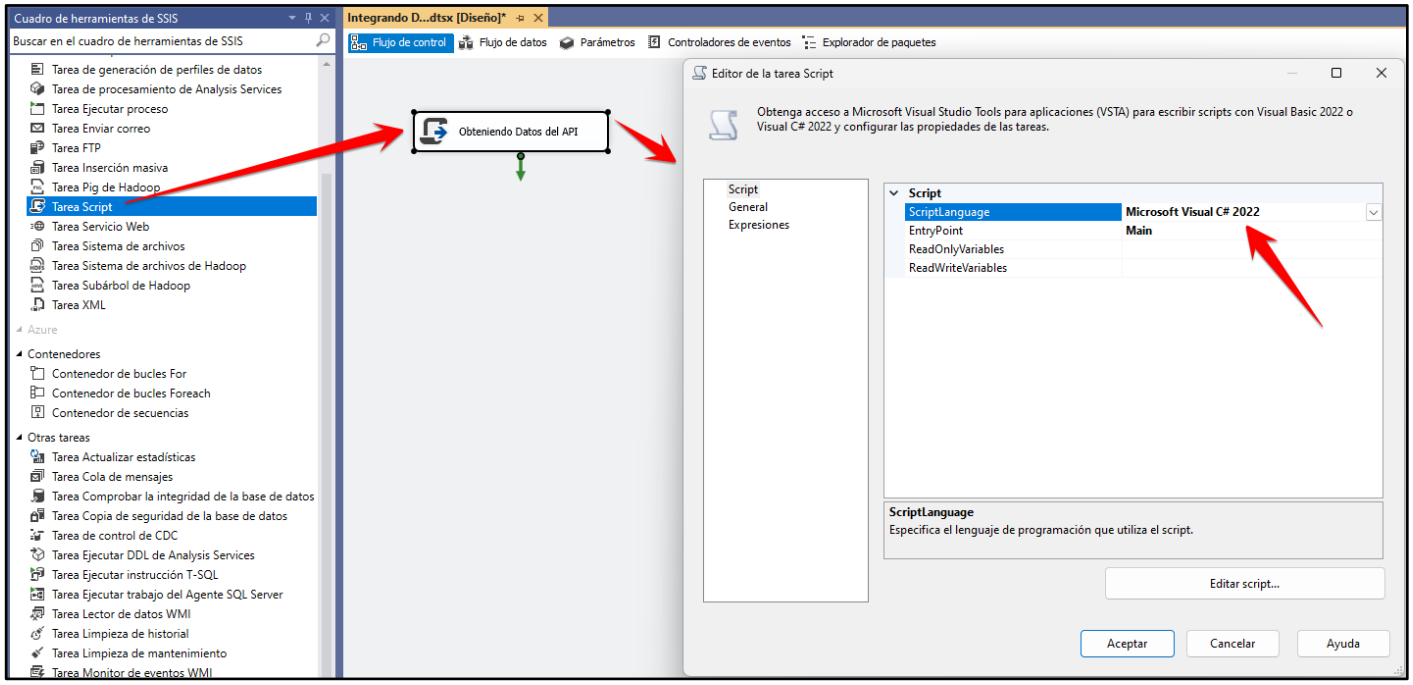
- **APIKey** (String): Almacenará el API Key de OpenWeatherMap que obtuve creando la cuenta.
- **City1** (String): Almacenará el nombre de una ciudad en específico (para este proyecto, "Ferreñafe").
- **City2** (String): Almacenará el nombre de una ciudad en específico (para este proyecto, "Chiclayo").
- **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.



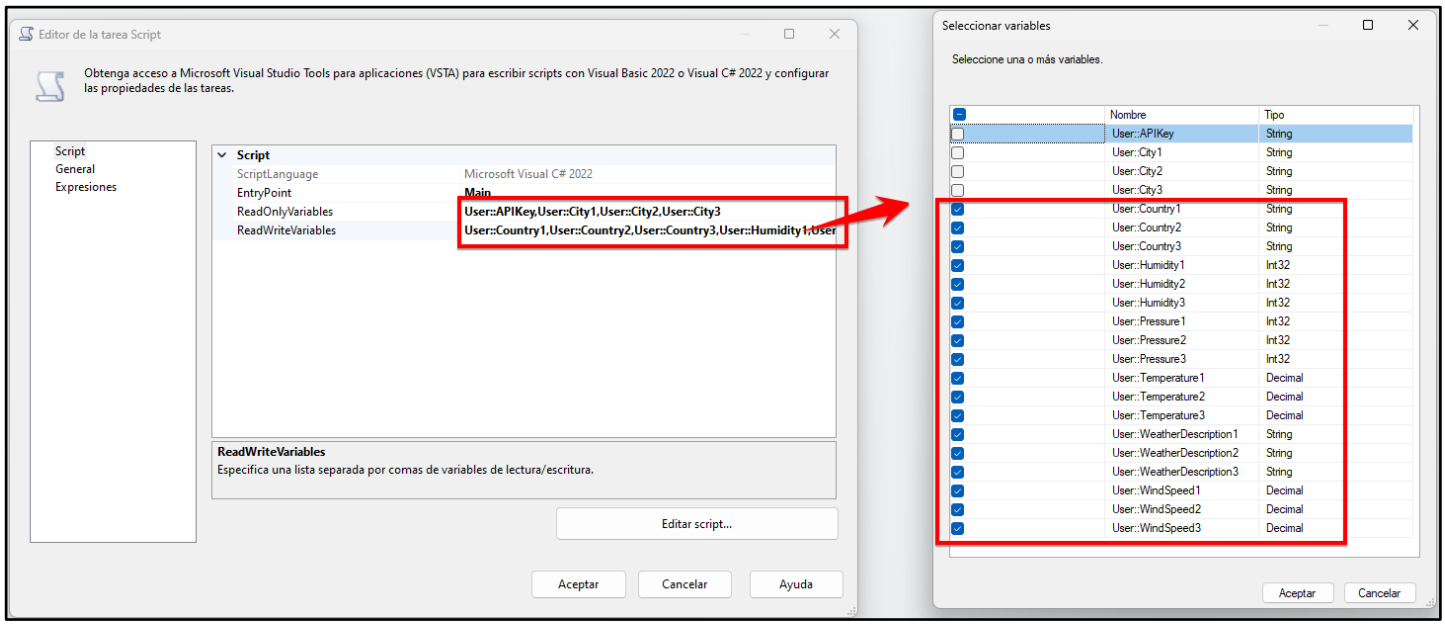
Paso 4: Componente de Tarea de Scripts.

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



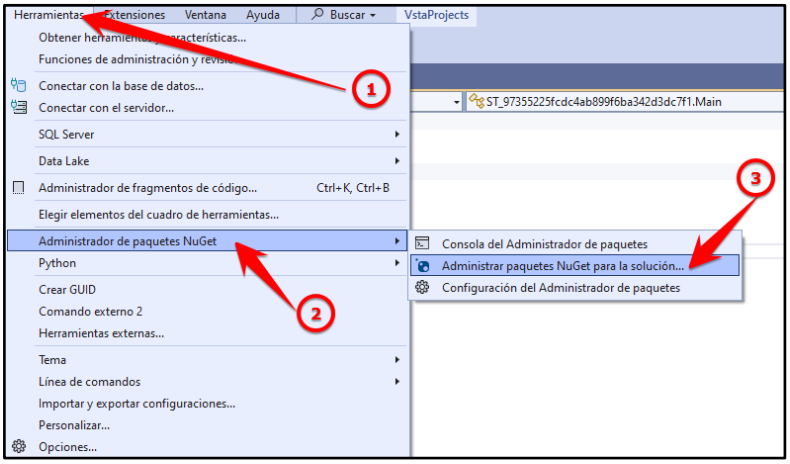
Se agregan las Variables al Script:

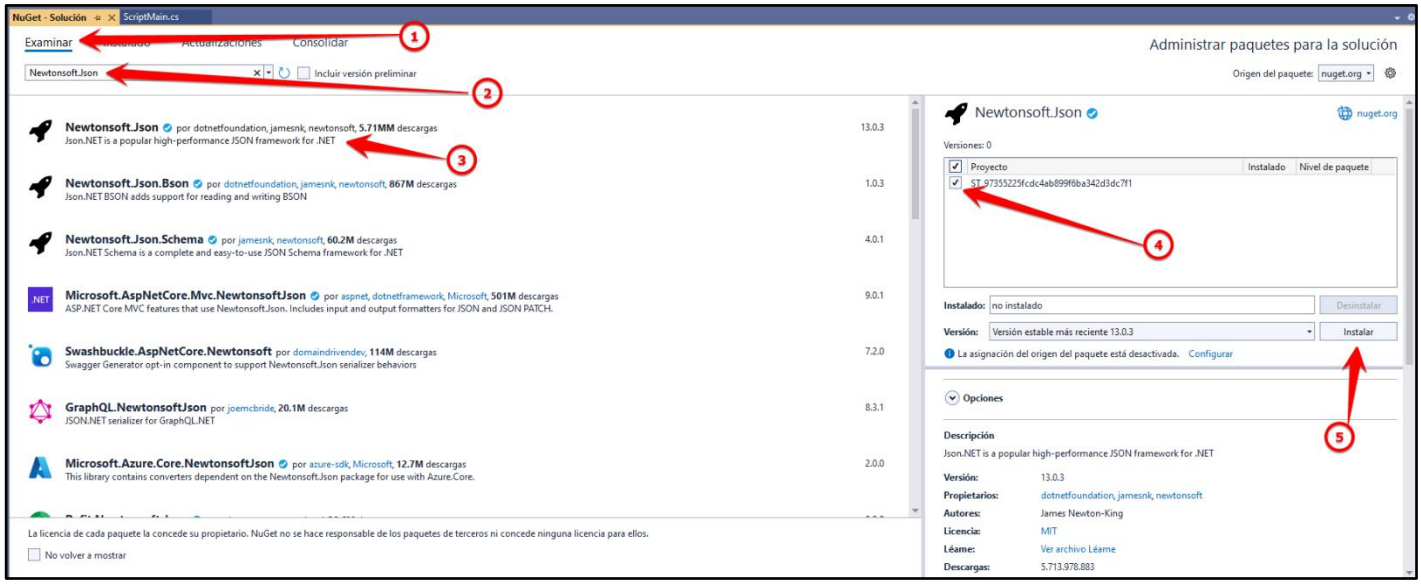
- En la pestaña "**ReadOnlyVariables**", seleccioné las variables User::APIKey, User::City1, User::City2, User::City3. (Las variables que le coloqué un valor.)
- En la pestaña "**ReadWriteVariables**", seleccioné las demas variables variables (las demás variables que no se le colocó ningún Valor).

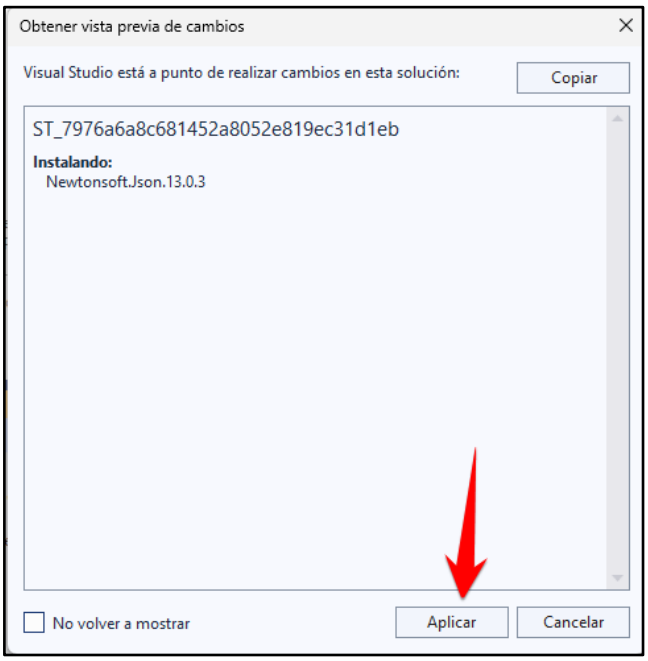


Paso 5: Editando el Script con el código C#.

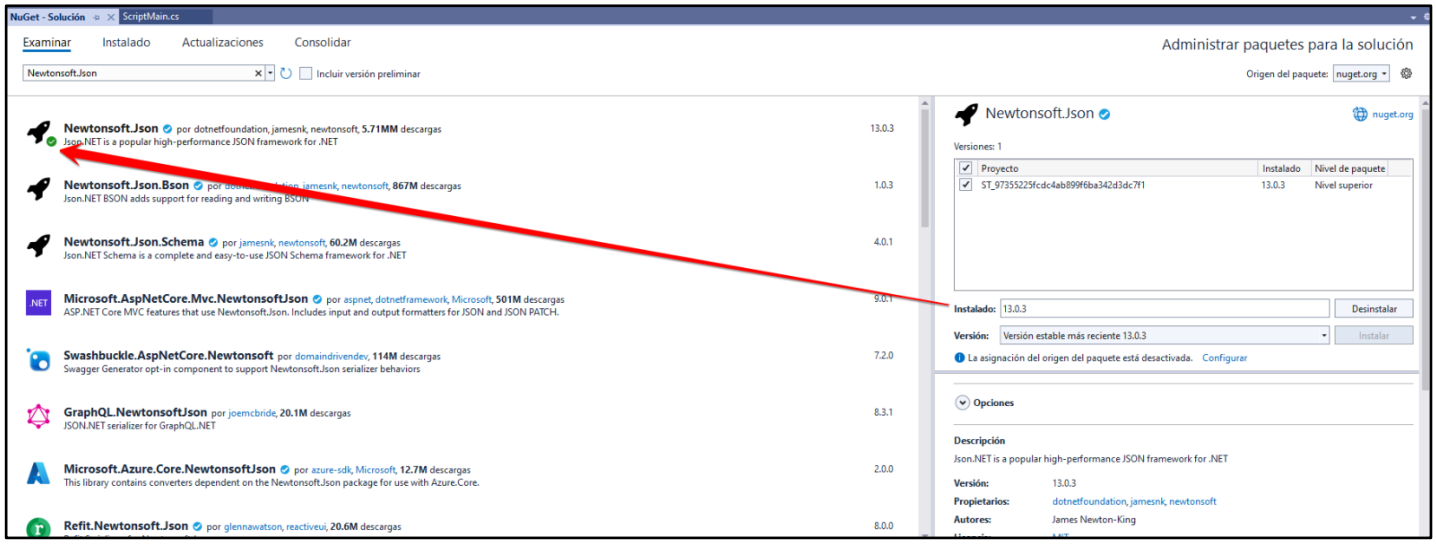
Primero instalaré una librería que usaré en el código, la librería es **Newtonsoft.Json:**



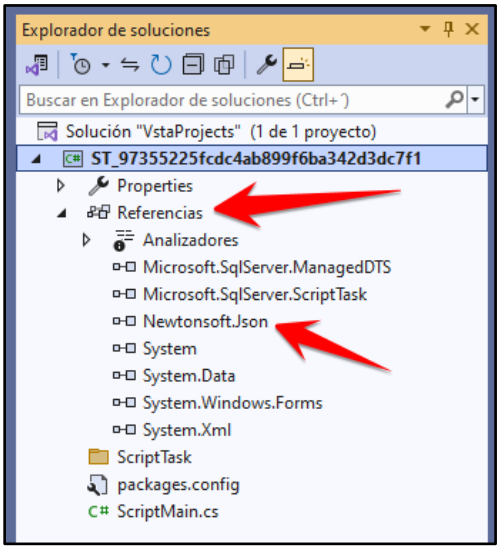




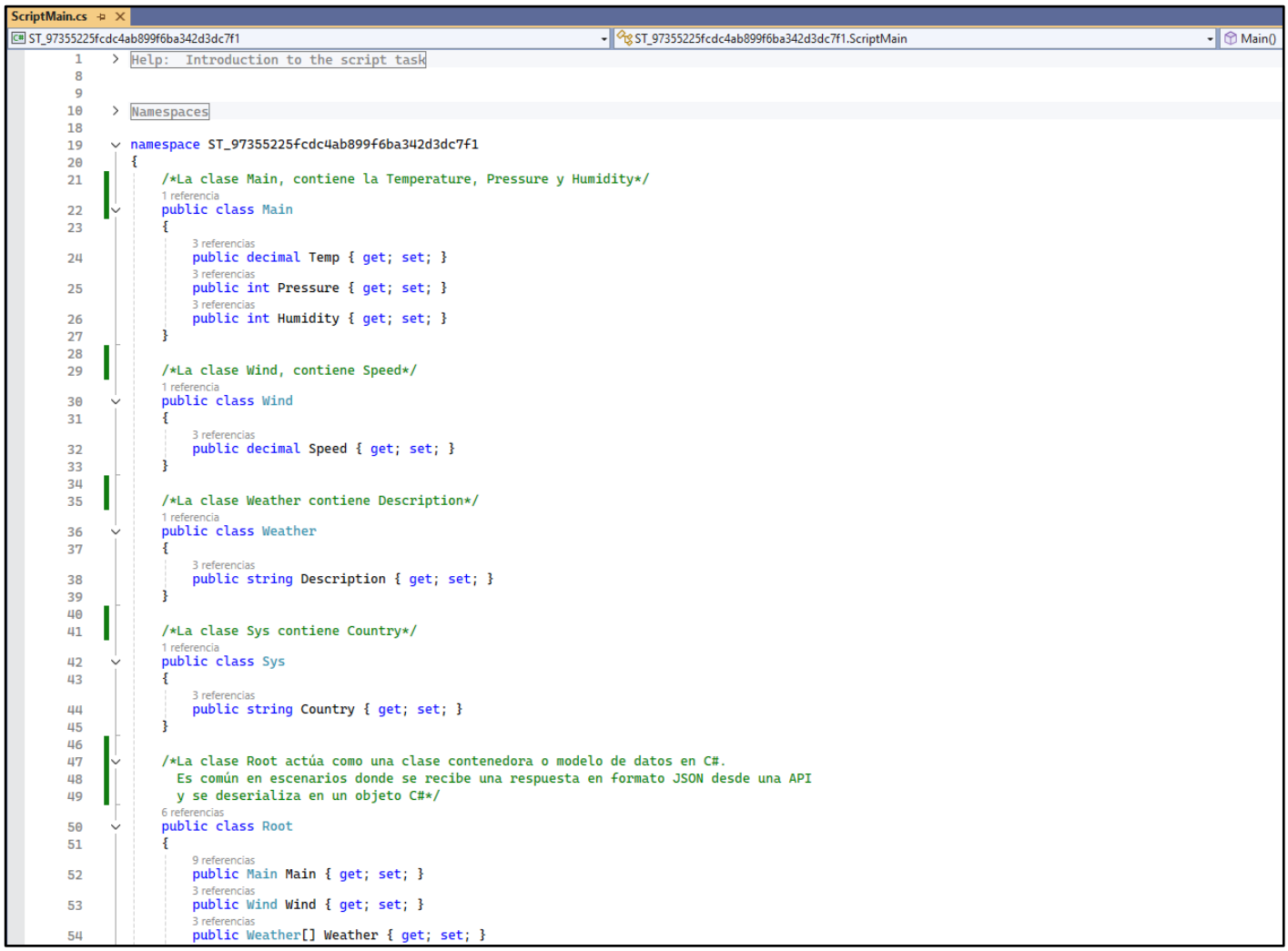
Se instaló correctamente:

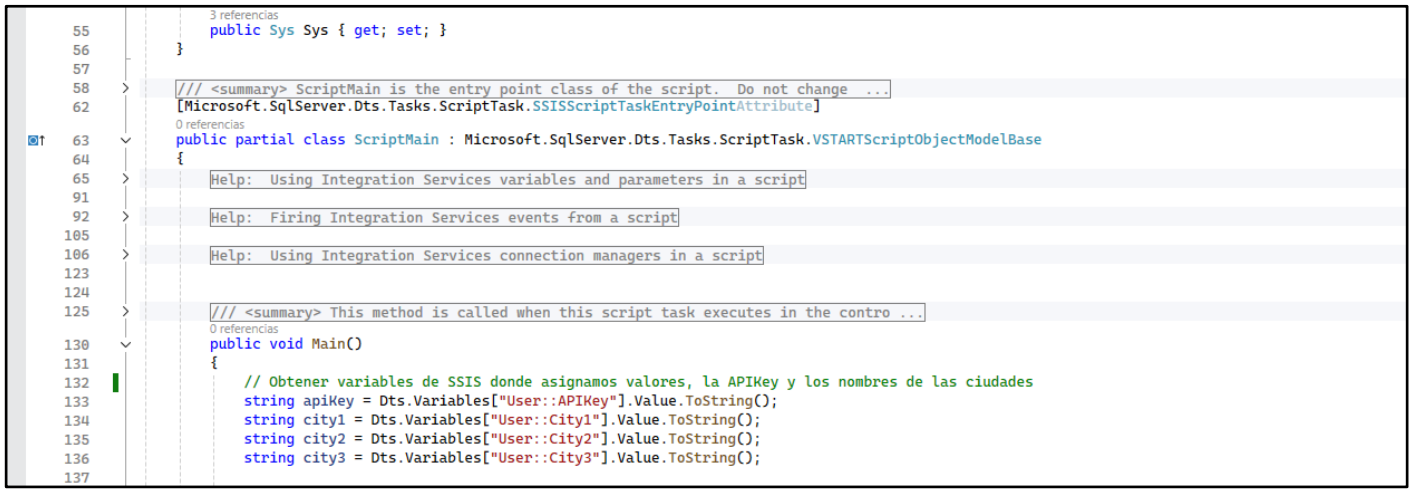


Verificamos que aparezca referenciado la librería que se agregó:



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.





```
            // Construir la URL de la API que hace llamado a cada ciudad con el mismo APIKey
            string url1 = $"http://api.openweathermap.org/data/2.5/weather?q={city1}&appid={apiKey}&units=metric";
            string url2 = $"http://api.openweathermap.org/data/2.5/weather?q={city2}&appid={apiKey}&units=metric";
            string url3 = $"http://api.openweathermap.org/data/2.5/weather?q={city3}&appid={apiKey}&units=metric";

            //Try Catch para controlar errores
            try
            {
                /*Crea una instancia de la clase WebClient para realizar solicitudes HTTP (Descargar archivos de una API o un archivo de Internet)*/
                using (WebClient wc = new WebClient())
                {

                    //Descarga datos en formato JSON desde una URL y convierte ese JSON  en un Objeto C# para cada ciudad.
                    string json1 = wc.DownloadString(url1);
                    Root root1 = JsonConvert.DeserializeObject<Root>(json1);

                    string json2 = wc.DownloadString(url2);
                    Root root2 = JsonConvert.DeserializeObject<Root>(json2);

                    string json3 = wc.DownloadString(url3);
                    Root root3 = JsonConvert.DeserializeObject<Root>(json3);


                    // Extraer datos relevantes de Ciudad1
                    string country1 = root1.Sys.Country.ToString();
                    decimal temperature1 = root1.Main.Temp;
                    int humidity1 = root1.Main.Humidity;
                    int pressure1 = root1.Main.Pressure;
                    decimal windSpeed1 = root1.Wind.Speed;
                    string weatherDescription1 = root1.Weather[0].Description.ToString();


                    // Guardar los datos en variables de SSIS de Ciudad1
                    Dts.Variables["User::Country1"].Value = country1;
                    Dts.Variables["User::Temperature1"].Value = temperature1;
                    Dts.Variables["User::Humidity1"].Value = humidity1;
                    Dts.Variables["User::Pressure1"].Value = pressure1;
                    Dts.Variables["User::WindSpeed1"].Value = windSpeed1;
                    Dts.Variables["User::WeatherDescription1"].Value = weatherDescription1;
                    Dts.Variables["User::WeatherDescription1"].Value = weatherDescription1;

                    // Extraer datos relevantes de Ciudad2
                    string country2 = root2.Sys.Country.ToString();
                    decimal temperature2 = root2.Main.Temp;
                    int humidity2 = root2.Main.Humidity;
                    int pressure2 = root2.Main.Pressure;
                    decimal windSpeed2 = root2.Wind.Speed;
                    string weatherDescription2 = root2.Weather[0].Description.ToString();
```

```
                    // Guardar los datos en variables de SSIS de Ciudad2
                    Dts.Variables["User::Country2"].Value = country2;
                    Dts.Variables["User::Temperature2"].Value = temperature2;
                    Dts.Variables["User::Humidity2"].Value = humidity2;
                    Dts.Variables["User::Pressure2"].Value = pressure2;
                    Dts.Variables["User::WindSpeed2"].Value = windSpeed2;
                    Dts.Variables["User::WeatherDescription2"].Value = weatherDescription2;


                    // Extraer datos relevantes de Ciudad3
                    string country3 = root3.Sys.Country.ToString();
                    decimal temperature3 = root3.Main.Temp;
                    int humidity3 = root3.Main.Humidity;
                    int pressure3 = root3.Main.Pressure;
                    decimal windSpeed3 = root3.Wind.Speed;
                    string weatherDescription3 = root3.Weather[0].Description.ToString();

                    // Guardar los datos en variables de SSIS de Ciudad1
                    Dts.Variables["User::Country3"].Value = country3;
                    Dts.Variables["User::Temperature3"].Value = temperature3;
                    Dts.Variables["User::Humidity3"].Value = humidity3;
                    Dts.Variables["User::Pressure3"].Value = pressure3;
                    Dts.Variables["User::WindSpeed3"].Value = windSpeed3;
                    Dts.Variables["User::WeatherDescription3"].Value = weatherDescription3;

```

```
                }
                Dts.TaskResult = (int)ScriptResults.Success;
            }
            catch (Exception ex)
            {
                // Manejar errores
                Dts.Events.FireError(0, "Script Task", ex.Message, string.Empty, 0);
                Dts.TaskResult = (int)ScriptResults.Failure;
            }

            Dts.TaskResult = (int)ScriptResults.Success;
        }

        #region ScriptResults declaration
        /// <summary>
        /// This enum provides a convenient shorthand within the scope of this class for setting the
        /// result of the script.
        ///
        /// This code was generated automatically.
        /// </summary>
        3 referencias
        enum ScriptResults
        {
            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        };
        #endregion

    }
}
```

Paso 6: Creando Tabla para almacenar los Datos.

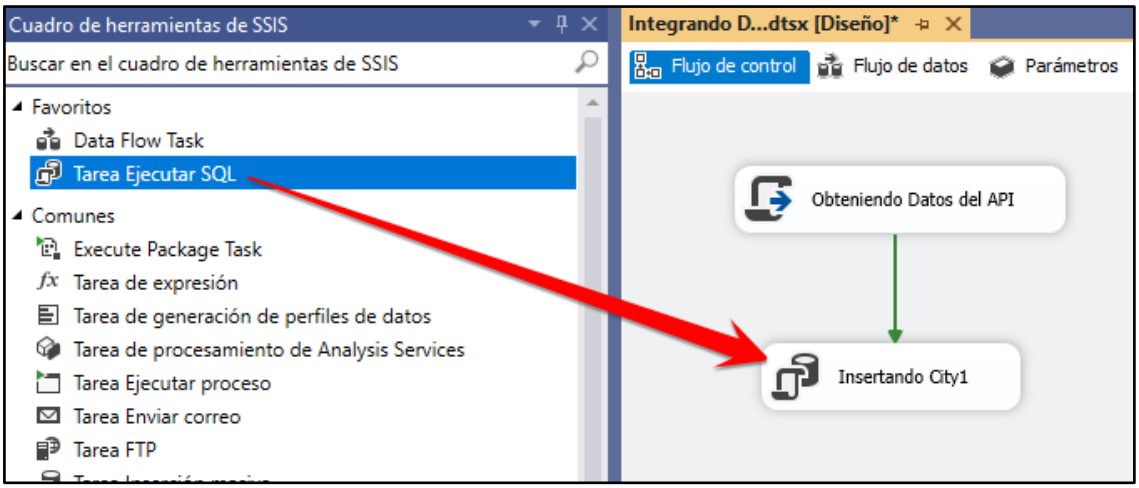
Crearé una Base de datos y una tabla en SQL Server donde se irán almacenando los datos extraídos del API:

BD_Creation.sql -...TIME_DATA (sa (61))*

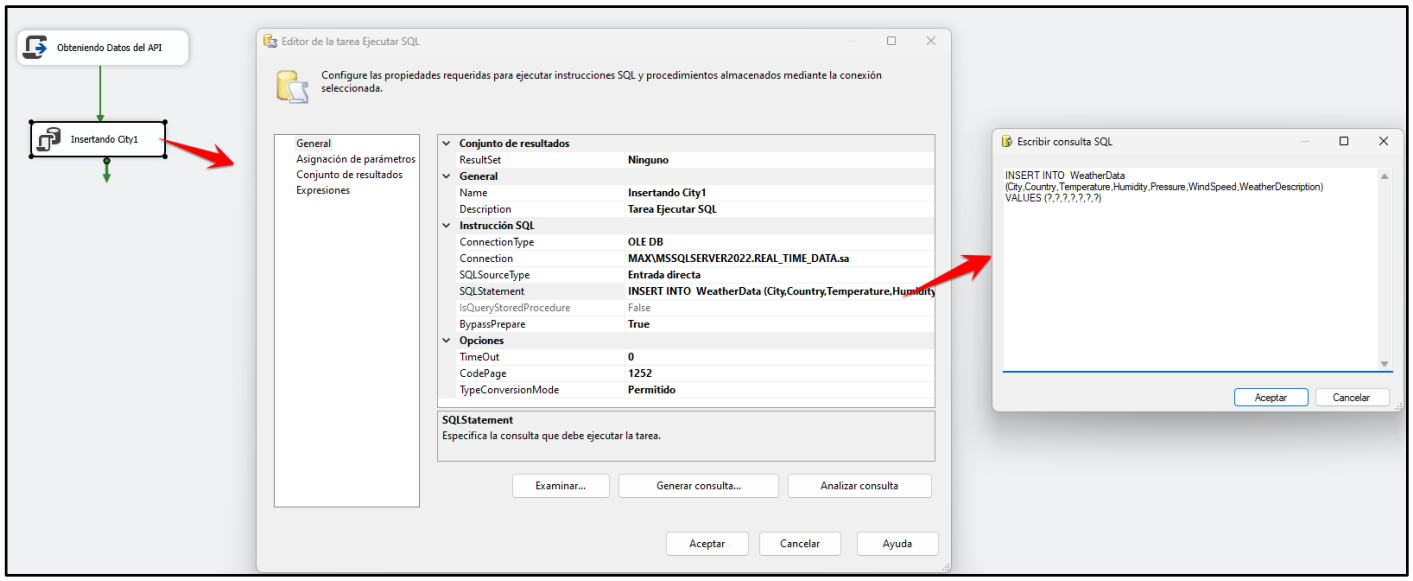
```
/*CREANDO LA BASE DE DATOS REAL_TIME_DATA*/

USE master
GO

IF EXISTS(SELECT NAME FROM SYS.databases WHERE NAME='REAL_TIME_DATA')
BEGIN
    DROP DATABASE REAL_TIME_DATA
END
GO

CREATE DATABASE REAL_TIME_DATA
GO

USE REAL_TIME_DATA
GO


SET ANSI_NULLS ON /*CONTROL Y MANEJO CORRECTO DE LOS VALORES NULL EN LAS COMPARACIONES*/
GO

SET QUOTED_IDENTIFIER ON /*PERMITE NOMBRES DE OBJETOS MAS FLEXIBLES Y EVITA PROBLEMAS CON PALABRAS RESERVADAS*/
GO

/*CREANDO TABLA DONDE SE GUARDARAN LOS DATOS DEL API*/
CREATE TABLE WeatherData (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    City NVARCHAR(100),
    Country NVARCHAR(100),
    Temperature decimal(10,2),
    Humidity INT,
    Pressure INT,
    WindSpeed decimal(10,2),
    WeatherDescription NVARCHAR(200),
    Timestamp DATETIME DEFAULT GETDATE()
);
```

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

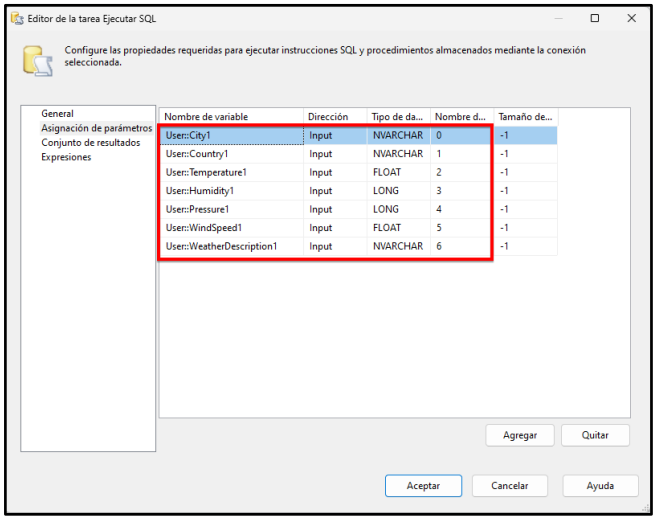
Ahora para el proceso de la carga de datos del JSON a una tabla en SQL Server utilicé el componente Tarea Ejecutar SQL para insertar los datos correspondientes a cada Ciudad.



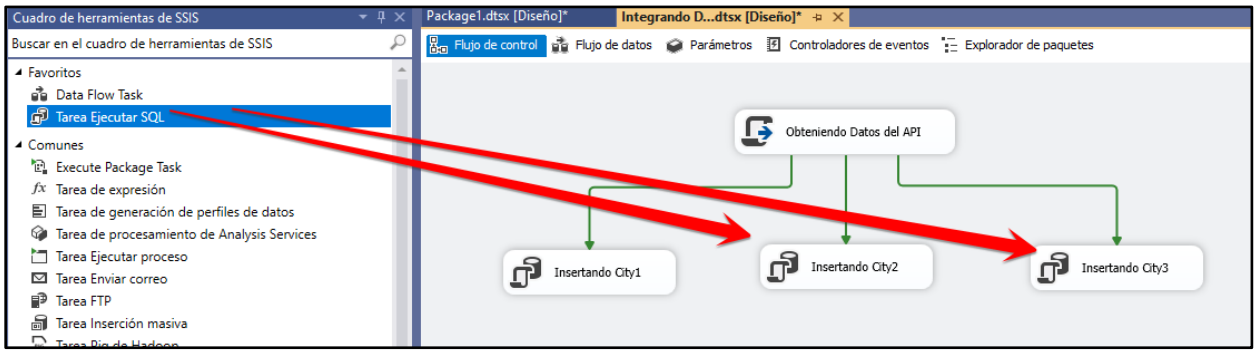
Agregué la conexión a la Base de Datos y el comando SQL donde le pasaré las variables.

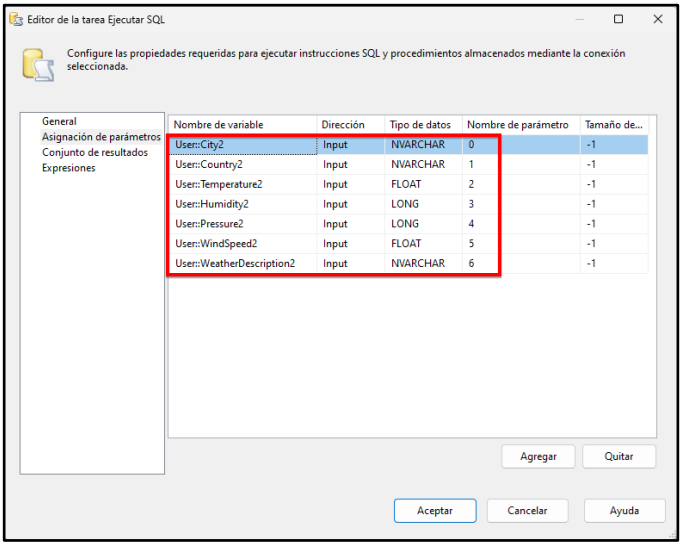


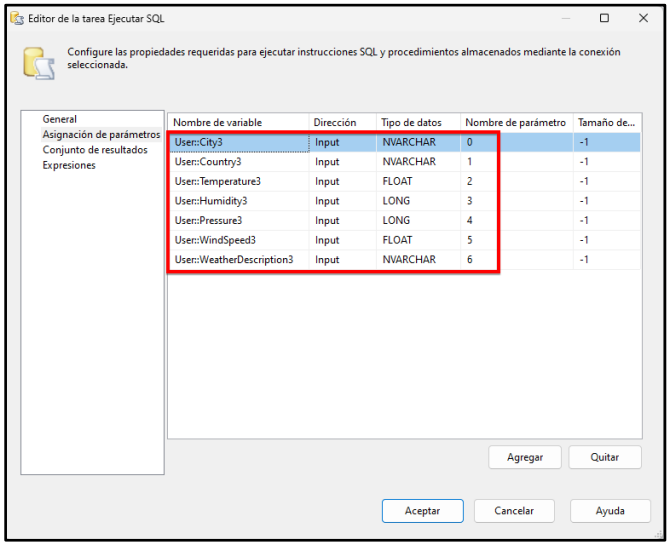
En Asignación de Variables colocamos todo lo relacionado a datos obtenidos para la ciudad 1.



Se hizo el mismo proceso para las demás ciudades:

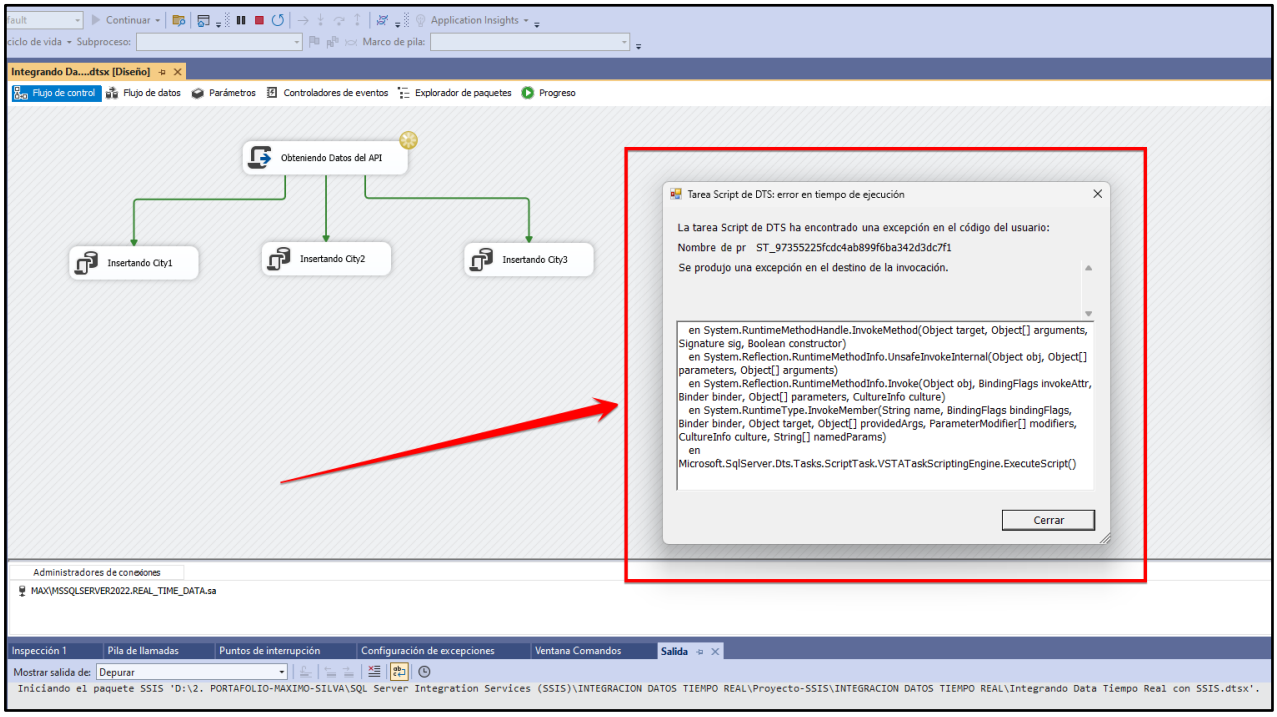


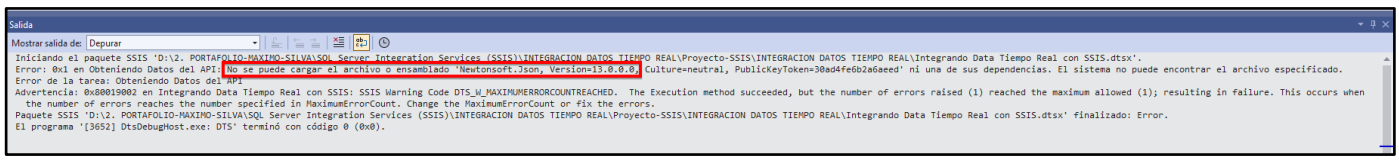




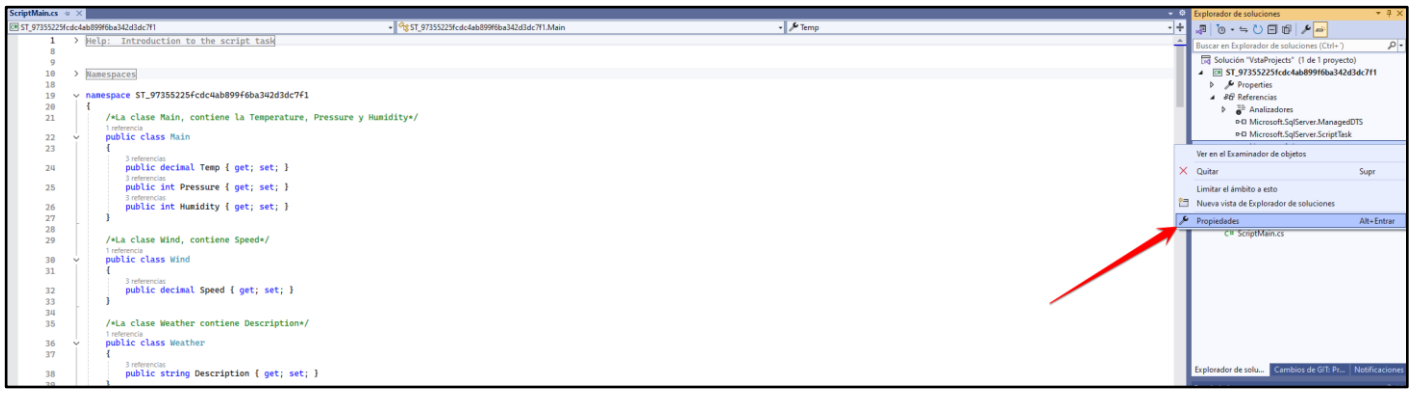
Paso 8: Error en la Ejecución y la solución al error.

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

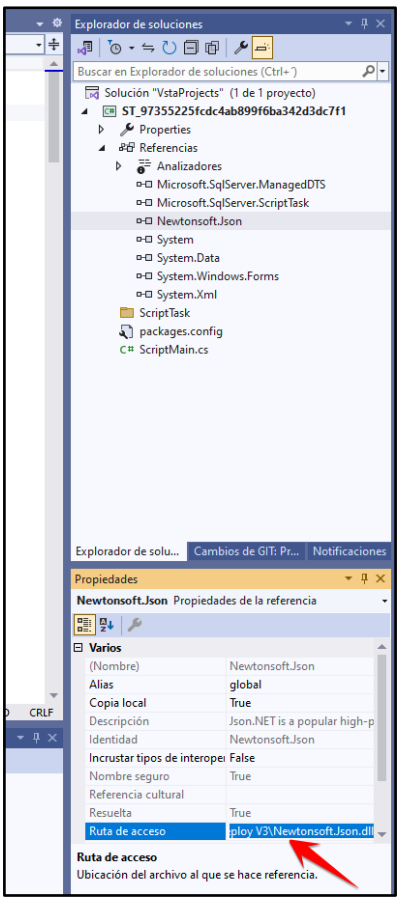




Estuve investigando y este error indica que la referencia a la librería **Newtonsoft.Json** no está registrada en la **GAC (Global Assembly Cache)** por lo que se necesita registrarla para que pueda se ubicada por el proyecto y así ser usada, para ello abrimos nuevamente donde se escribió el código C#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



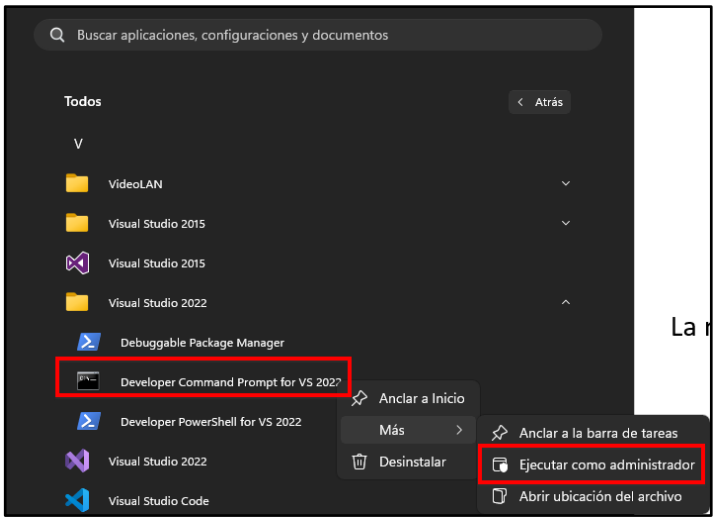
Identifiqué la Ruta de la librería y la copié:



La ruta donde se descargó fue:

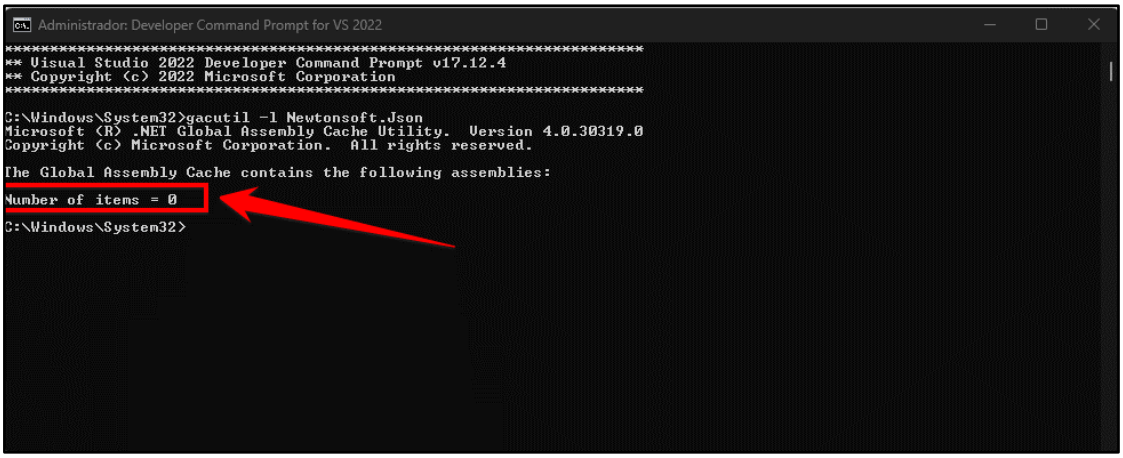
C:\Program Files\IIS\Microsoft Web Deploy V3\Newtonsoft.Json.dll

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



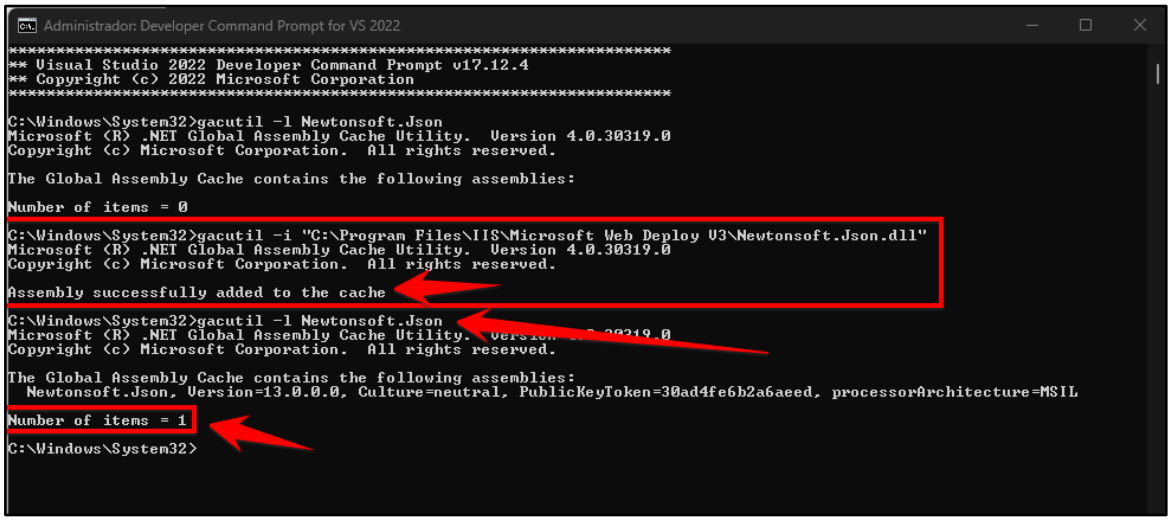
En la pantalla de comandos verificamos que no se encuentra registrada la Librería:

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"



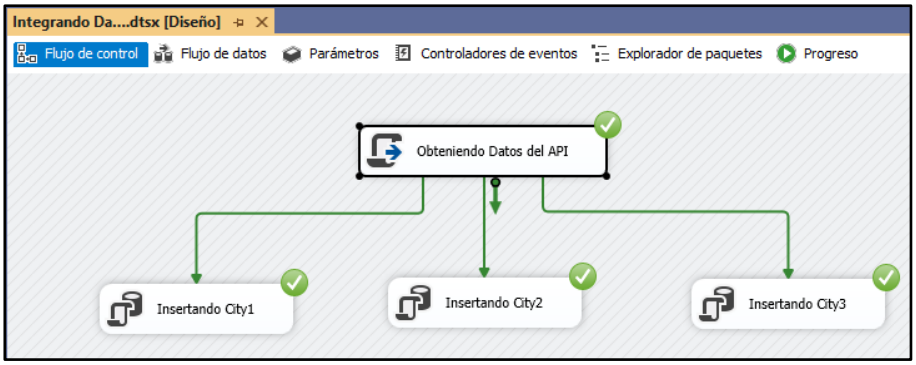
Al volver a verificar la librería, ya aparece el número 1, que indica que, si está registrado una librería, en este caso, Newtonsoft.Json. **Ya se debería haber solucionado el problema.**

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

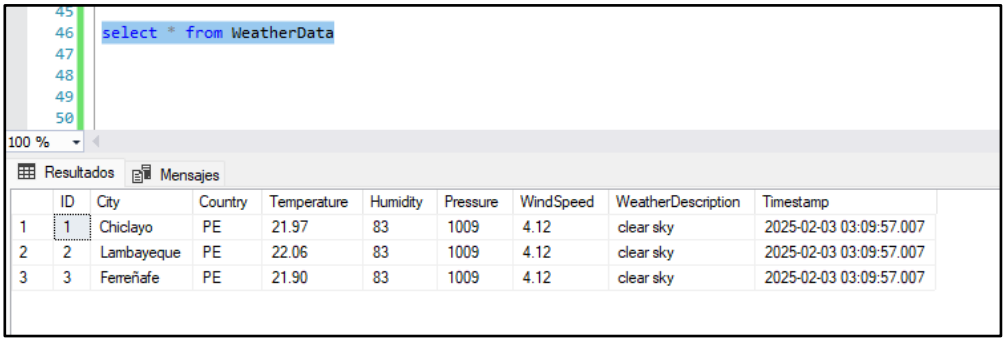
gacutil -u Newtonsoft.Json

Paso 9: Ejecutando el proyecto.

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



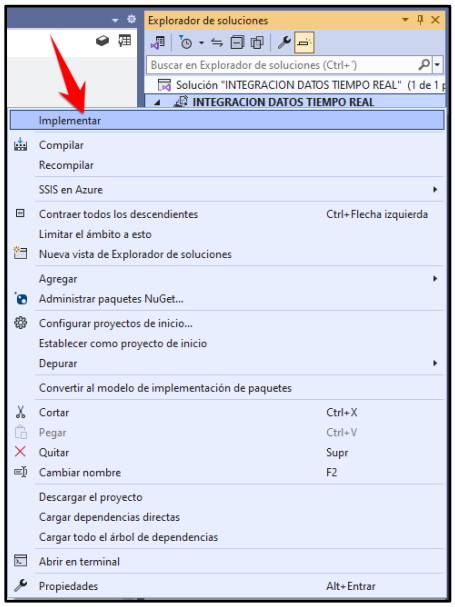
Verifico que se registraron los datos en la Base de datos:

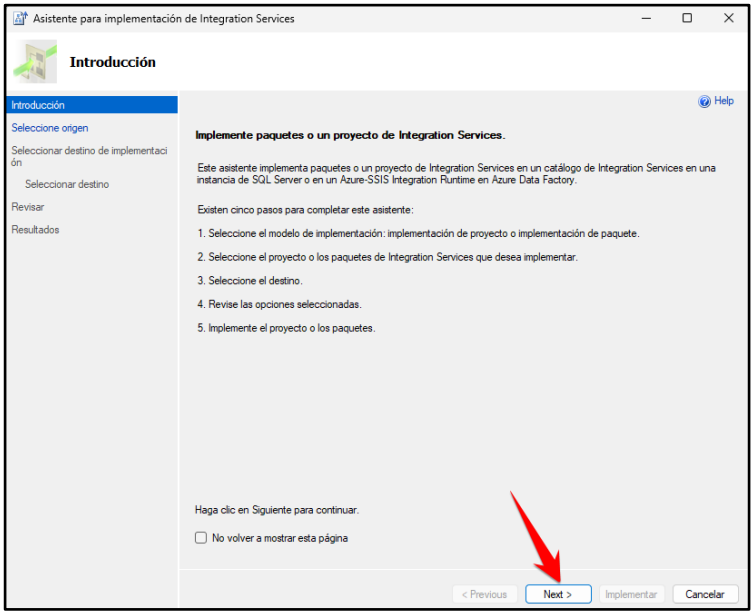


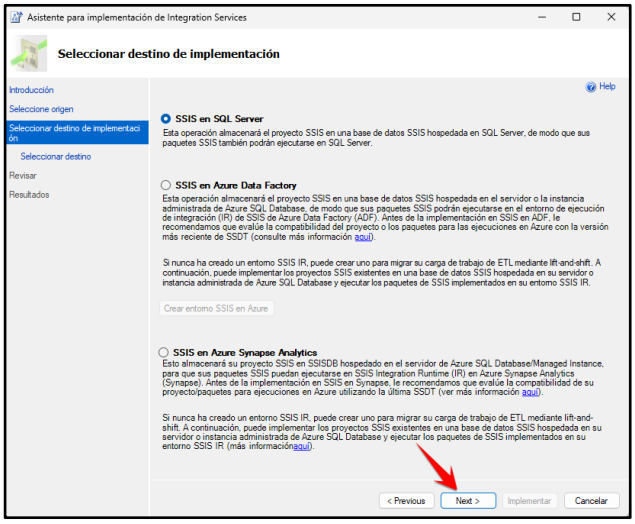
	ID	City	Country	Temperature	Humidity	Pressure	WindSpeed	WeatherDescription	Timestamp
1	1	Chiclayo	PE	21.97	83	1009	4.12	clear sky	2025-02-03 03:09:57.007
2	2	Lambayeque	PE	22.06	83	1009	4.12	clear sky	2025-02-03 03:09:57.007
3	3	Ferreñafe	PE	21.90	83	1009	4.12	clear sky	2025-02-03 03:09:57.007

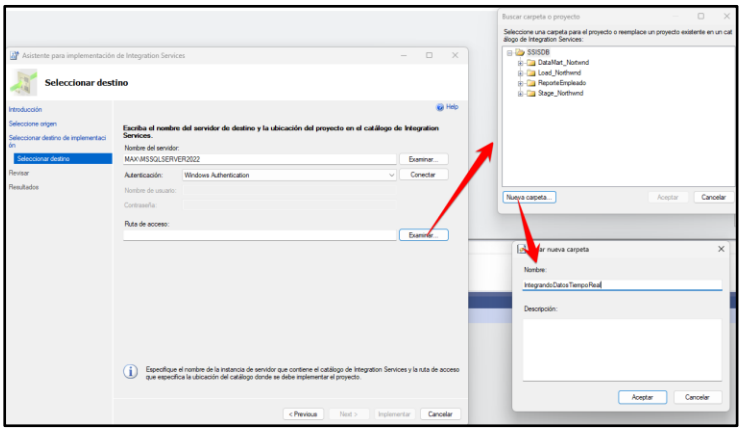
Paso 10: Implementando el proyecto de Integration Services.

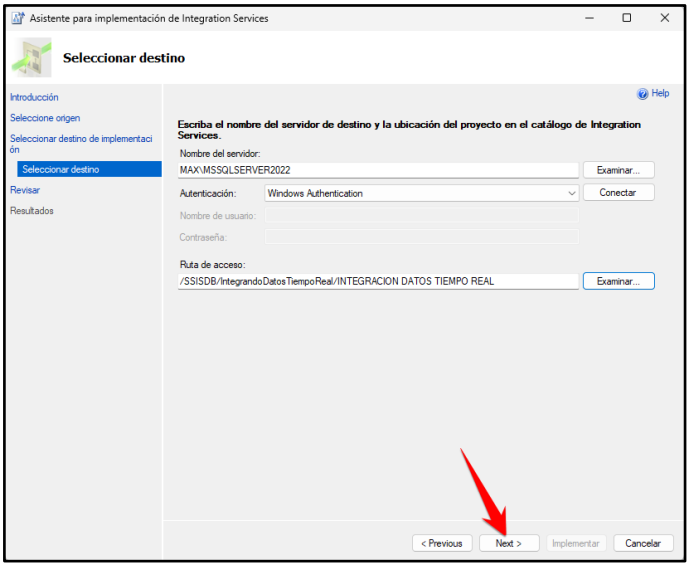
Una vez tenemos el proyecto, empezaré con el despliegue para que aparezca en SQL Server y poder crear una Tarea Programada en el Agente de SQL Server, para ello lo implementé:

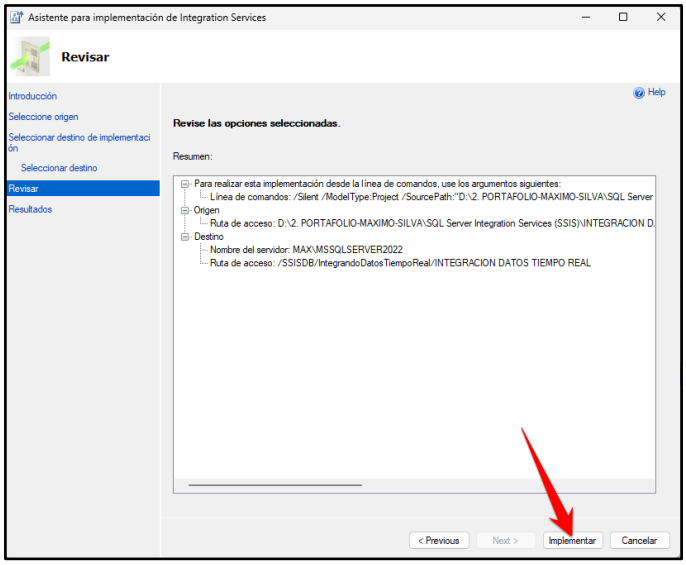

Explorador de soluciones
Buscar en Explorador de soluciones (Ctrl+ ')
Solución "INTEGRACION DATOS TIEMPO REAL" (1 de 1 p
INTEGRACION DATOS TIEMPO REAL
Implementar
Compilar
Recompilar
SSIS en Azure
Contraer todos los descendientes Ctrl+Flecha izquierda
Limitar el ámbito a esto
Nueva vista de Explorador de soluciones
Agregar
Administrar paquetes NuGet...
Configurar proyectos de inicio...
Establecer como proyecto de inicio
Depurar
Convertir al modelo de implementación de paquetes
Cortar Ctrl+X
Pegar Ctrl+V
Quitar Supr
Cambiar nombre F2
Descargar el proyecto
Cargar dependencias directas
Cargar todo el árbol de dependencias
Abrir en terminal
Propiedades Alt+Entrar

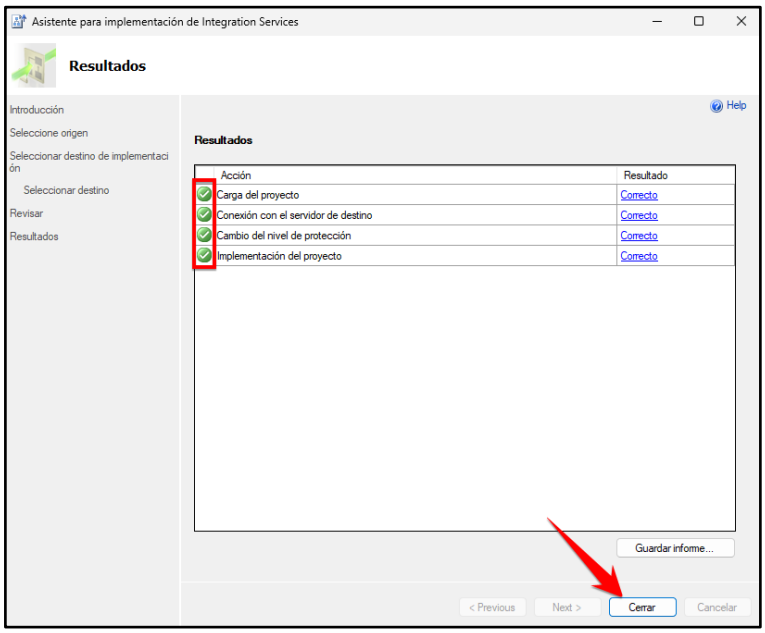

Asistente para implementación de Integration Services
Introducción
Introducción
Seleccione origen
Seleccionar destino de implementación
Seleccionar destino
Revisar
Resultados
Help
Implemente paquetes o un proyecto de Integration Services.
Este asistente implementa paquetes o un proyecto de Integration Services en un catálogo de Integration Services en una instancia de SQL Server o en un Azure-SSIS Integration Runtime en Azure Data Factory.
Existen cinco pasos para completar este asistente:
1. Seleccione el modelo de implementación: implementación de proyecto o implementación de paquete.
2. Seleccione el proyecto o los paquetes de Integration Services que desea implementar.
3. Seleccione el destino.
4. Revise las opciones seleccionadas.
5. Implemente el proyecto o los paquetes.
Haga clic en Siguiente para continuar.
No volver a mostrar esta página
< Previous
Next >
Implementar
Cancelar


Asistente para implementación de Integration Services
Seleccionar destino de implementación
Introducción
Seleccione origen
Seleccionar destino de implementación
Seleccionar destino
Revisar
Resultados
Help
SSIS en SQL Server
Esta operación almacenará el proyecto SSIS en una base de datos SSIS hospedada en SQL Server, de modo que sus paquetes SSIS también podrán ejecutarse en SQL Server.
SSIS en Azure Data Factory
Esta operación almacenará el proyecto SSIS en una base de datos SSIS hospedada en el servidor o la instancia administrada de Azure SQL Database, de modo que sus paquetes SSIS podrán ejecutarse en el entorno de ejecución de integración (IR) de SSIS de Azure Data Factory (ADF). Antes de la implementación en SSIS en ADF, le recomendamos que evalúe la compatibilidad del proyecto o los paquetes para las ejecuciones en Azure con la versión más reciente de SSDT (consulte más información aquí).
Si nunca ha creado un entorno SSIS IR, puede crear uno para migrar su carga de trabajo de ETL mediante lift-and-shift. A continuación, puede implementar los proyectos SSIS existentes en una base de datos SSIS hospedada en su servidor o instancia administrada de Azure SQL Database y ejecutar los paquetes de SSIS implementados en su entorno SSIS IR.
Crear entorno SSIS en Azure
SSIS en Azure Synapse Analytics
Esto almacenará su proyecto SSIS en SSISDB hospedado en el servidor de Azure SQL Database/Managed Instance, para que sus paquetes SSIS puedan ejecutarse en SSIS Integration Runtime (IR) en Azure Synapse Analytics (Synapse). Antes de la implementación en SSIS en Synapse, le recomendamos que evalúe la compatibilidad de su proyecto/paquetes para ejecuciones en Azure utilizando la última SSDT (ver más información aquí).
Si nunca ha creado un entorno SSIS IR, puede crear uno para migrar su carga de trabajo de ETL mediante lift-and-shift. A continuación, puede implementar los proyectos SSIS existentes en una base de datos SSIS hospedada en su servidor o instancia administrada de Azure SQL Database y ejecutar los paquetes de SSIS implementados en su entorno SSIS IR (más información aquí).
< Previous
Next >
Implementar
Cancelar

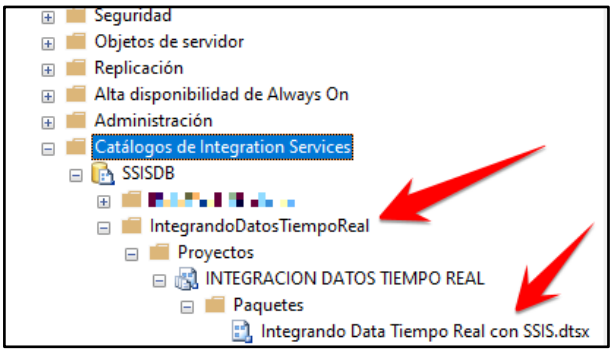

Seleccionar destino
Escriba el nombre del servidor de destino y la ubicación del proyecto en el catálogo de Integration Services.
Nombre del servidor:
MAX\MSSQLSERVER2022
Autenticación:
Windows Authentication
Nombre de usuario:
Contraseña:
Ruta de acceso:
Examinar...
Conectar
Seleccionar una carpeta para el proyecto o reemplace un proyecto existente en un catálogo de Integration Services:
SSISDB
DataMart_Nothwind
Load_Nothwind
ReporteEmpleado
Stage_Nothwind
Nueva carpeta...
Aceptar
Cancelar
Nombre:
IntegrandoDatosTiempoReal
Descripción:
Especifique el nombre de la instancia de servidor que contiene el catálogo de Integration Services y la ruta de acceso que especifica la ubicación del catálogo donde se debe implementar el proyecto.


Asistente para implementación de Integration Services
Seleccionar destino
Introducción
Seleccione origen
Seleccionar destino de implementación
Seleccionar destino
Revisar
Resultados
Help
Escriba el nombre del servidor de destino y la ubicación del proyecto en el catálogo de Integration Services.
Nombre del servidor:
MAX\MSSQLSERVER2022
Examinar...
Autenticación:
Windows Authentication
Conectar
Nombre de usuario:
Contraseña:
Ruta de acceso:
/SSISDB/IntegrandoDatosTiempoReal/INTEGRACION DATOS TIEMPO REAL
Examinar...
< Previous
Next >
Implementar
Cancelar


Asistente para implementación de Integration Services
Revisar
Introducción
Seleccione origen
Seleccionar destino de implementación
Seleccionar destino
Revisar
Resultados
Help
Revise las opciones seleccionadas.
Resumen:
Para realizar esta implementación desde la línea de comandos, use los argumentos siguientes:
Línea de comandos: /Silent /ModelType:Project /SourcePath:"D:\2. PORTAFOLIO-MAXIMO-SILVA\SQL Server
Origen
Ruta de acceso: D:\2. PORTAFOLIO-MAXIMO-SILVA\SQL Server Integration Services (SSIS)\INTEGRACION D
Destino
Nombre del servidor: MAX\MSSQLSERVER2022
Ruta de acceso: /SSISDB/IntegrandoDatosTiempoReal/INTEGRACION DATOS TIEMPO REAL
< Previous
Next >
Implementar
Cancelar

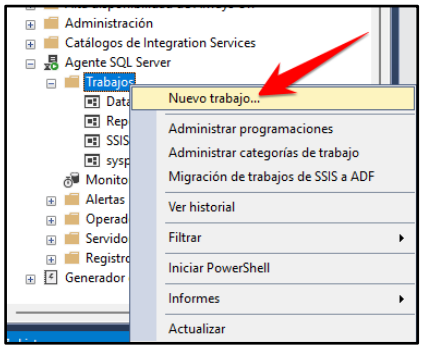


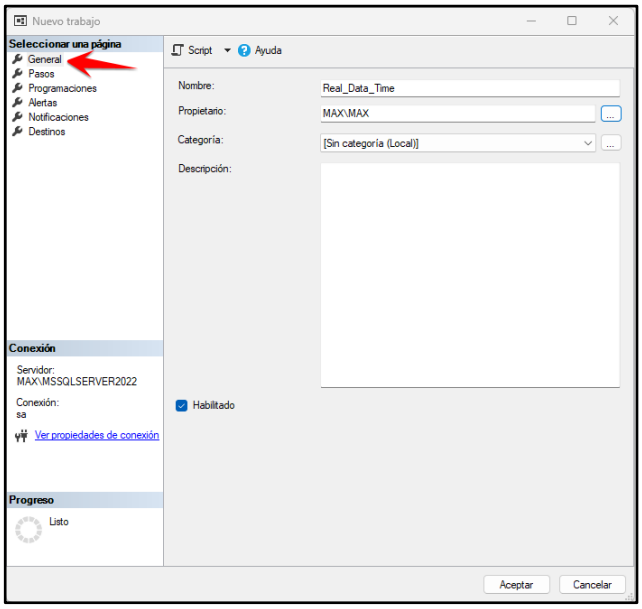
En SQL Server, en el apartado de Catálogos de Integration Services (Previamente se tuvo que crear un nuevo catálogo), ya aparece el proyecto:

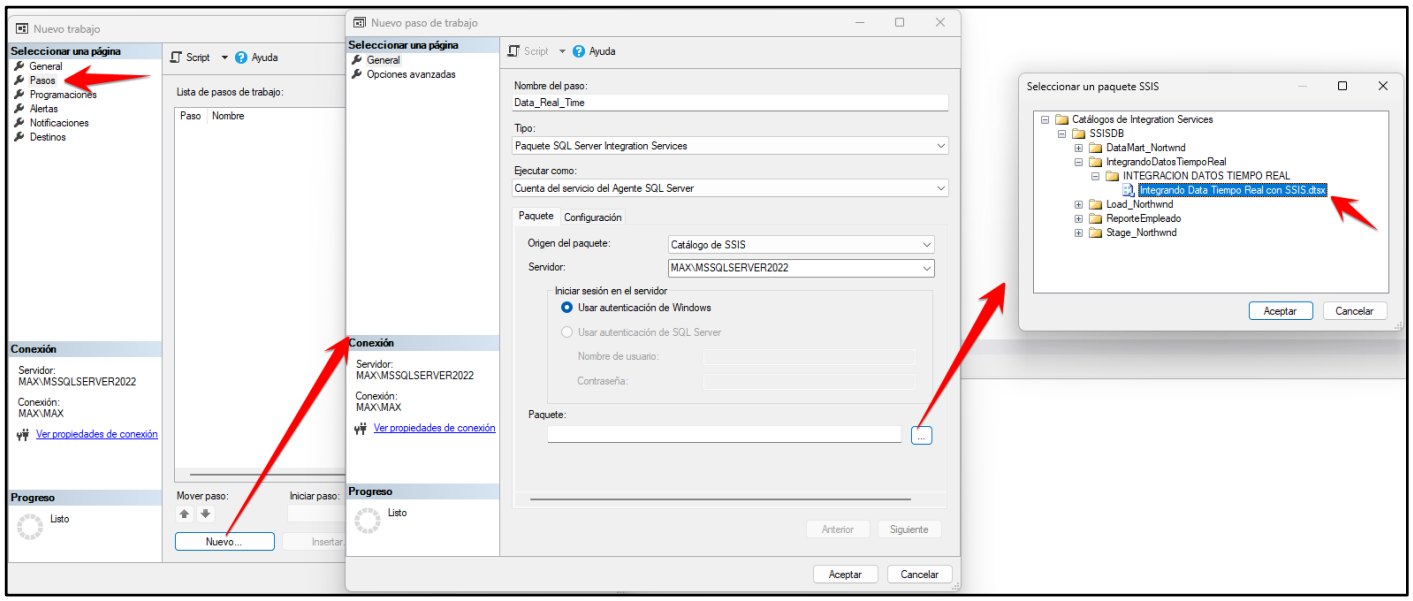


Paso 11: Creando una tarea programada automática con Agente SQL Server.

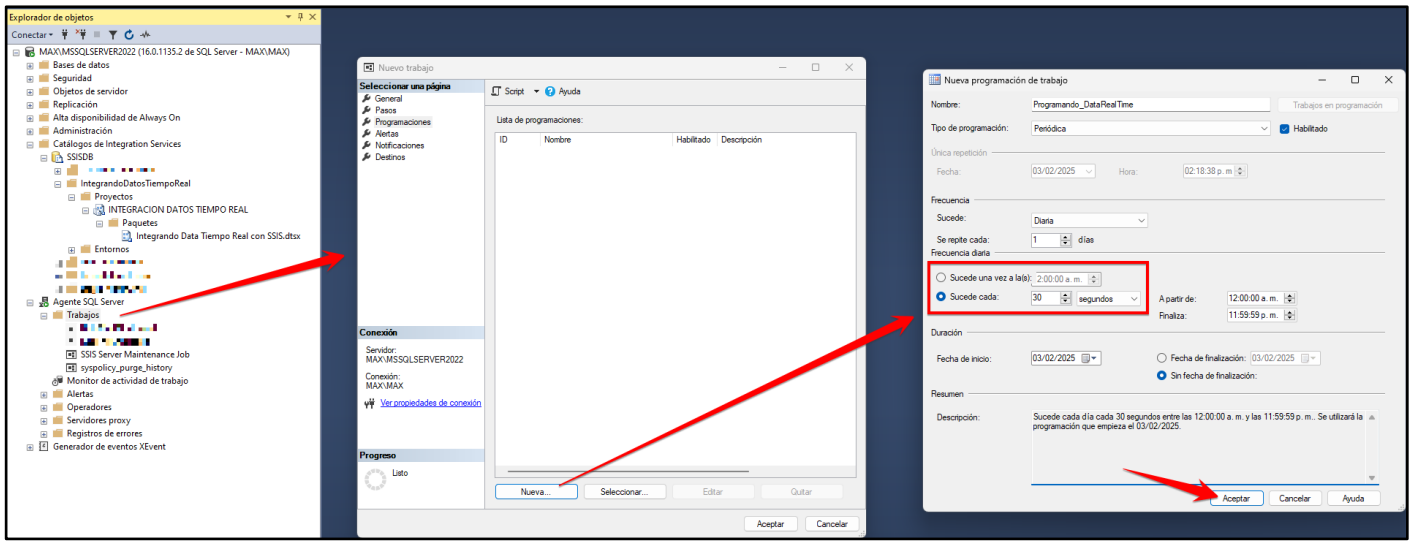
Cree un nuevo Job (trabajo) en SQL Server que ejecute el proyecto Implementado anteriormente:

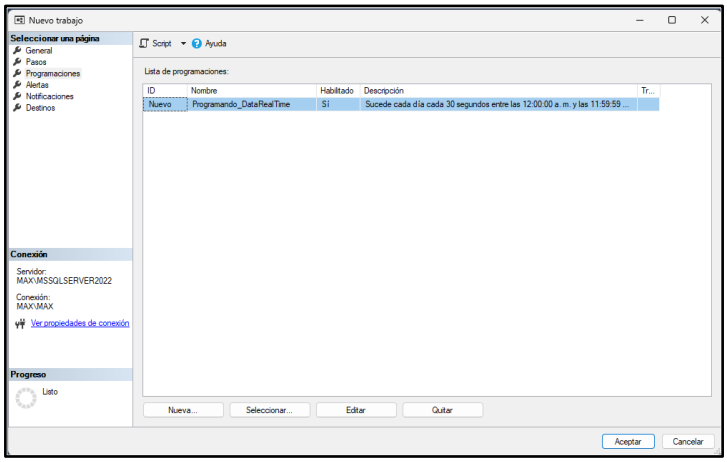




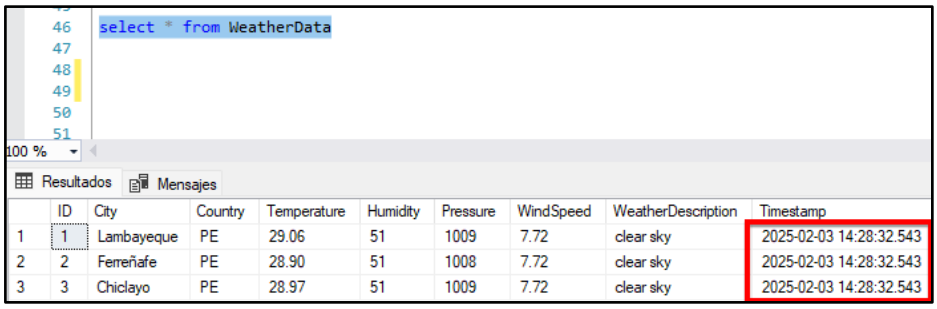


Programando para que se ejecute automáticamente cada 30 segundos:





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



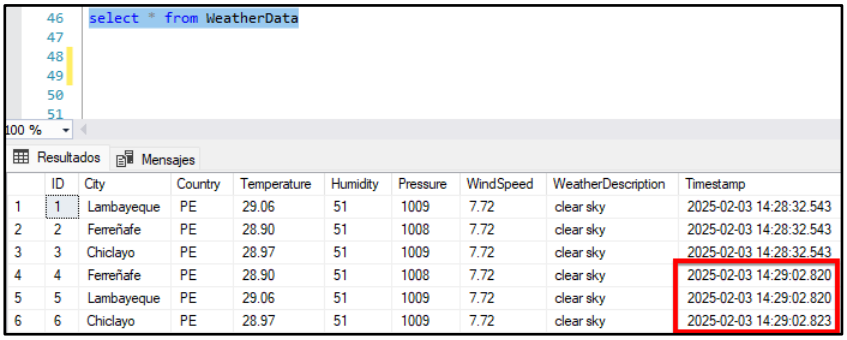
```
select * from WeatherData
```

100 %

Resultados | Mensajes

	ID	City	Country	Temperature	Humidity	Pressure	WindSpeed	WeatherDescription	Timestamp
1	1	Lambayeque	PE	29.06	51	1009	7.72	clear sky	2025-02-03 14:28:32.543
2	2	Ferreñafe	PE	28.90	51	1008	7.72	clear sky	2025-02-03 14:28:32.543
3	3	Chiclayo	PE	28.97	51	1009	7.72	clear sky	2025-02-03 14:28:32.543

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



```
select * from WeatherData
```

100 %

Resultados | Mensajes

	ID	City	Country	Temperature	Humidity	Pressure	WindSpeed	WeatherDescription	Timestamp
1	1	Lambayeque	PE	29.06	51	1009	7.72	clear sky	2025-02-03 14:28:32.543
2	2	Ferreñafe	PE	28.90	51	1008	7.72	clear sky	2025-02-03 14:28:32.543
3	3	Chiclayo	PE	28.97	51	1009	7.72	clear sky	2025-02-03 14:28:32.543
4	4	Ferreñafe	PE	28.90	51	1008	7.72	clear sky	2025-02-03 14:29:02.820
5	5	Lambayeque	PE	29.06	51	1009	7.72	clear sky	2025-02-03 14:29:02.820
6	6	Chiclayo	PE	28.97	51	1009	7.72	clear sky	2025-02-03 14:29:02.823

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.