Technical Documentation

This document describes how to configure the entire Datawarehouse on a local machine.

Required Software

The following list of software need to be installed before proceeding with the installation of the solution:

1. Git v. 2.44.0
2. SQL Server (install and configure a local instance) v. 2022 (16.0.1000.6).
   ⚠ Important: choose SQL Server Developer version and install *Integration Services*
3. SQL Server Management Studio (SSMS) v. 2014 (12.0.2000.8).
4. Visual Studio v. 16.11.34
5. Data Tools Integration Services for Visual Studio 2019
6. Draw IO v24.1.0

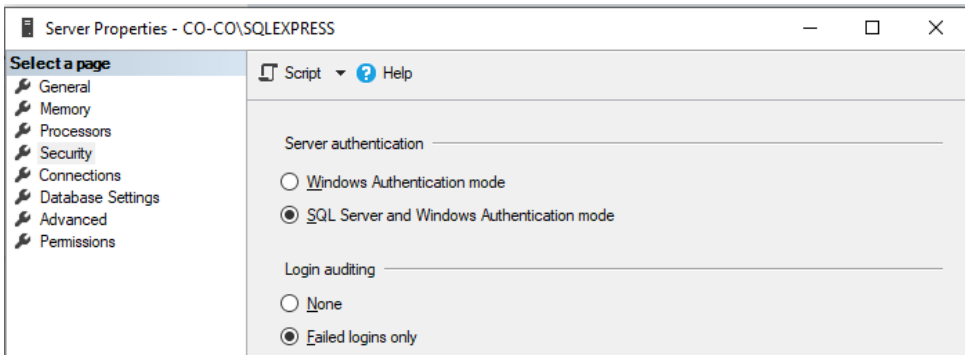
To install SSIS, we can do it via executables or if it doesn't work, directly on Visual Studio by using extensions. It depends on the Visual Studio version.

Installation procedure

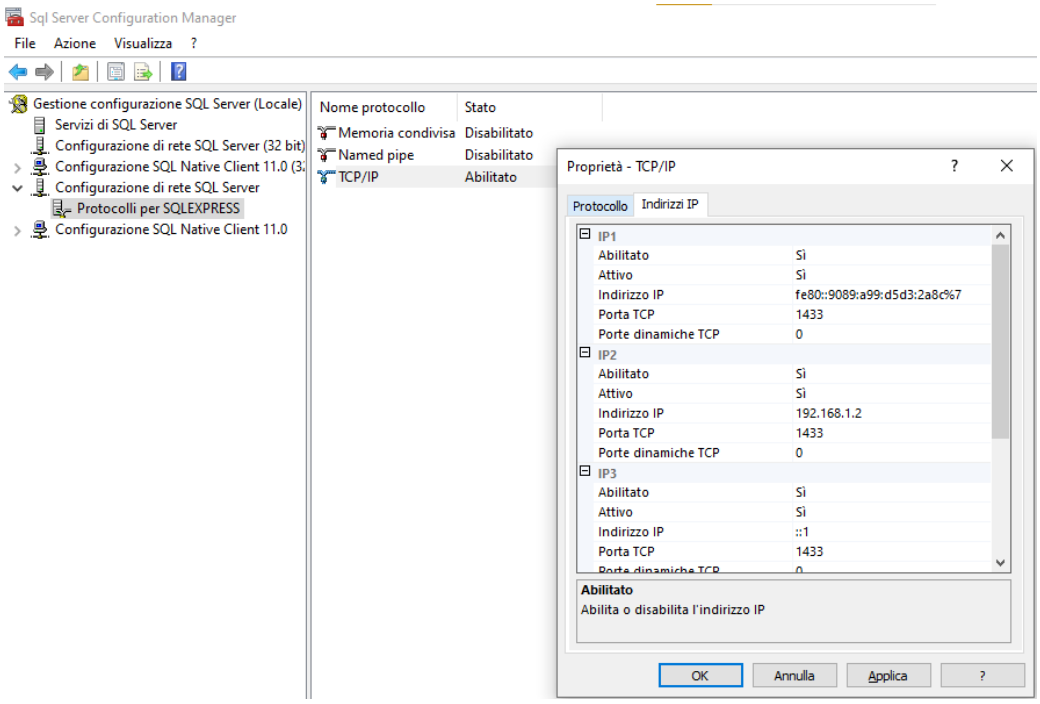
- Clone the public repository with the following instruction:

  git clone https://github.com/myDelevop/ForeignExchangeRateDWH.git

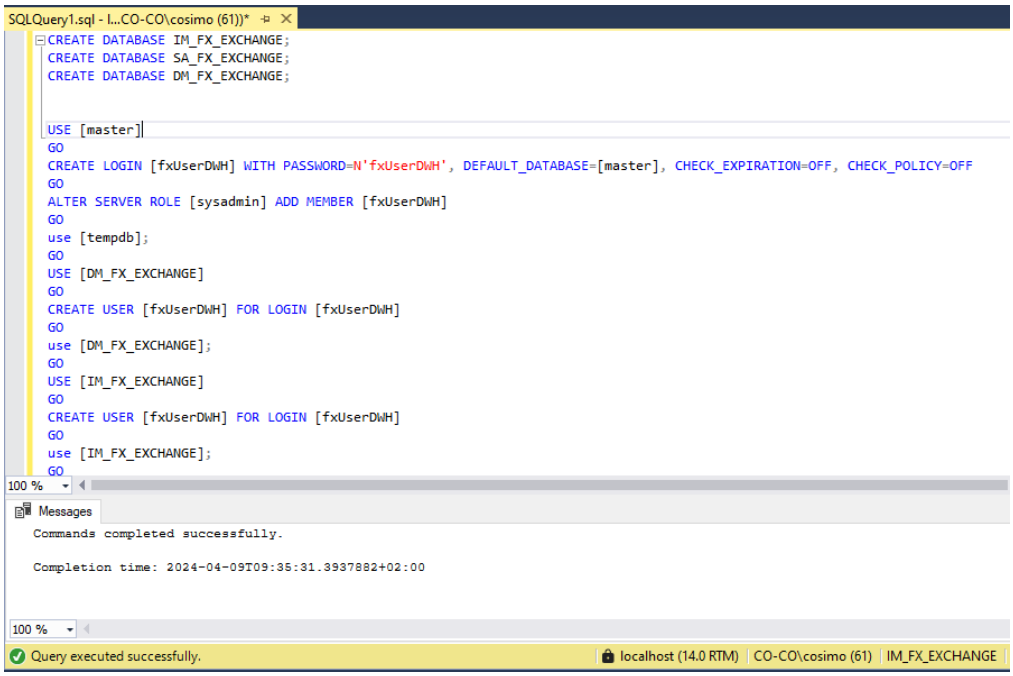
- *Open SSMS and Connect to the local installed instance of DB. Then, right click on DB name, go to Properties, go to Security tab and choose "SQL Server and Windows authentication mode"*



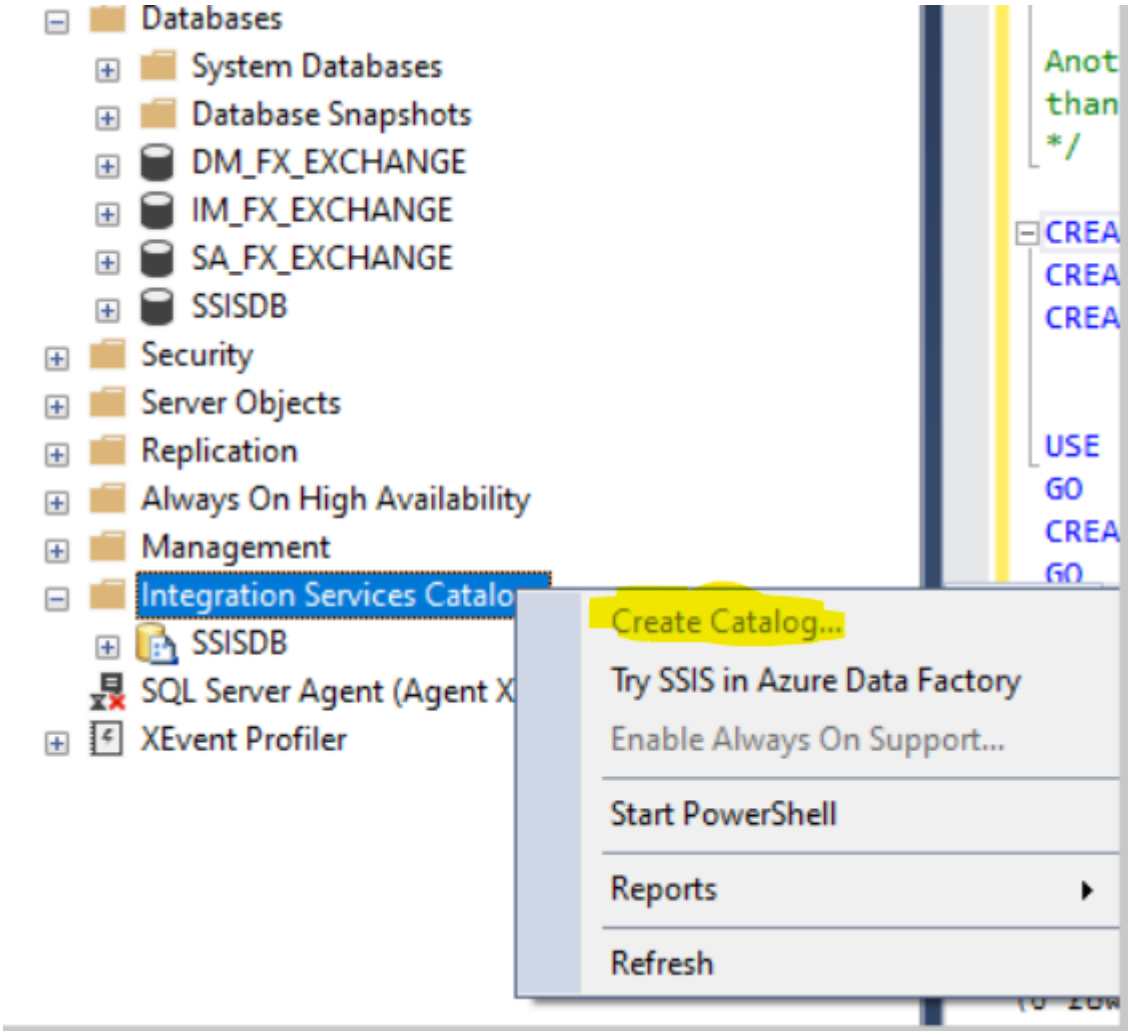
- Then, open SQL Server Configuration Manager, open protocols and go to properties on TCP/IP protocol. Set Enabled Yes and TCP port equals to the default SQL Server Port: 1433



- Restart the database service this will allow us to connect by using 127.0.0.1 (localhost)
- Run the script in *README.md* on the localhost (or servername) instance in the SQL Server DB. This allows us to create the databases in SQL Server that will contain our Datawarehouse



- Create SSISDB by creating a SSIS Catalog: right click on *Integration Services Catalog, Create Catalog*... In my case is obscure because I have already created it:



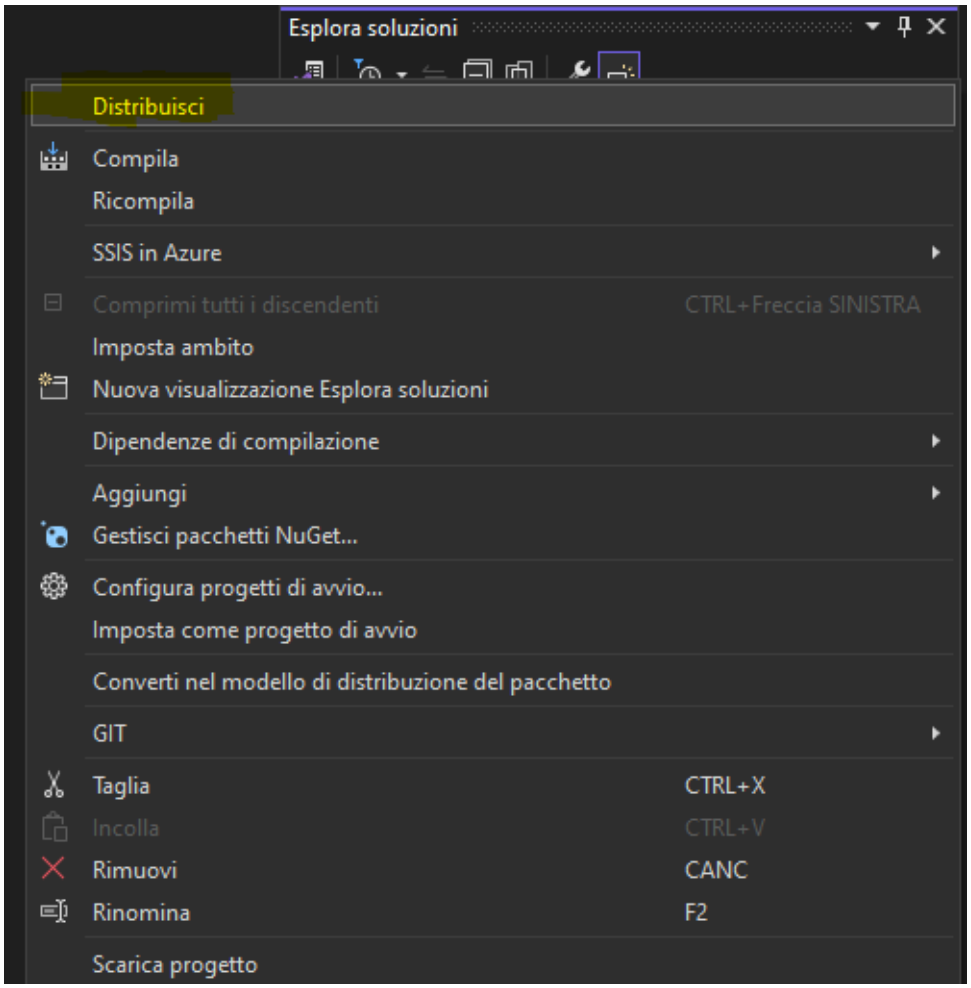
- Now it is possible to open the file *ForeingExchangeRate.sln* with Visual Studio and run the solution.

- Place the input file rates_sample.csv under the following path: *C:\rates_sample.csv*.

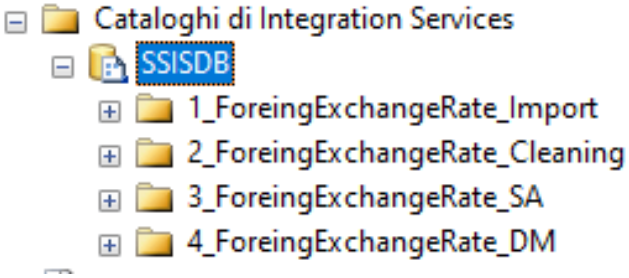
Project Deployment

1 – Rebuild each of the following projects (right click on the project name and click on "*Rebuild*" option:

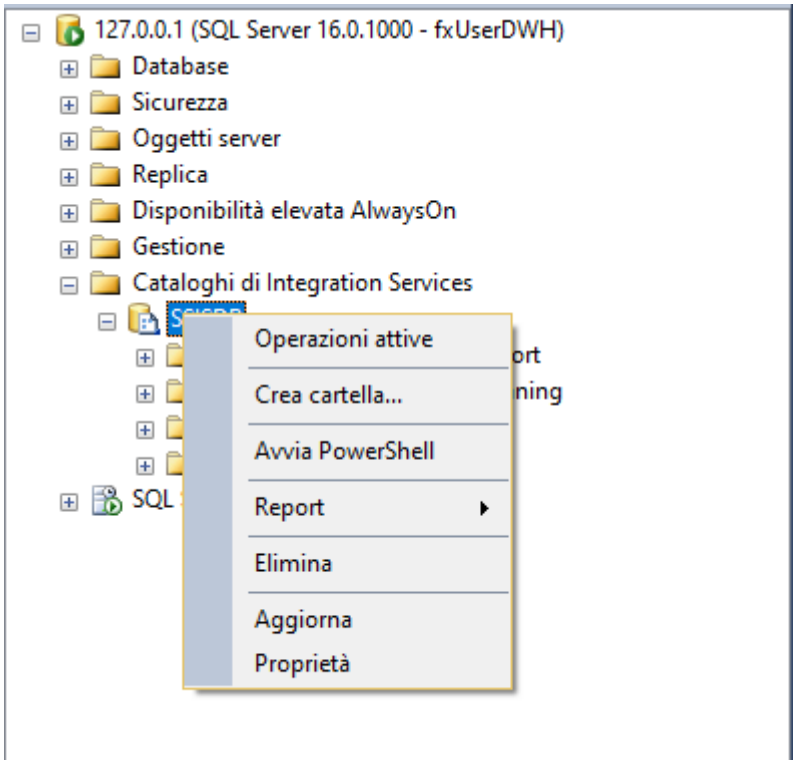
- *1_ForeignExchangeRate_Import*
- *2_ForeignExchangeRate_Cleaning*
- *3_ForeignExchangeRate_SA*
- *4_ForeignExchangeRate_DM*



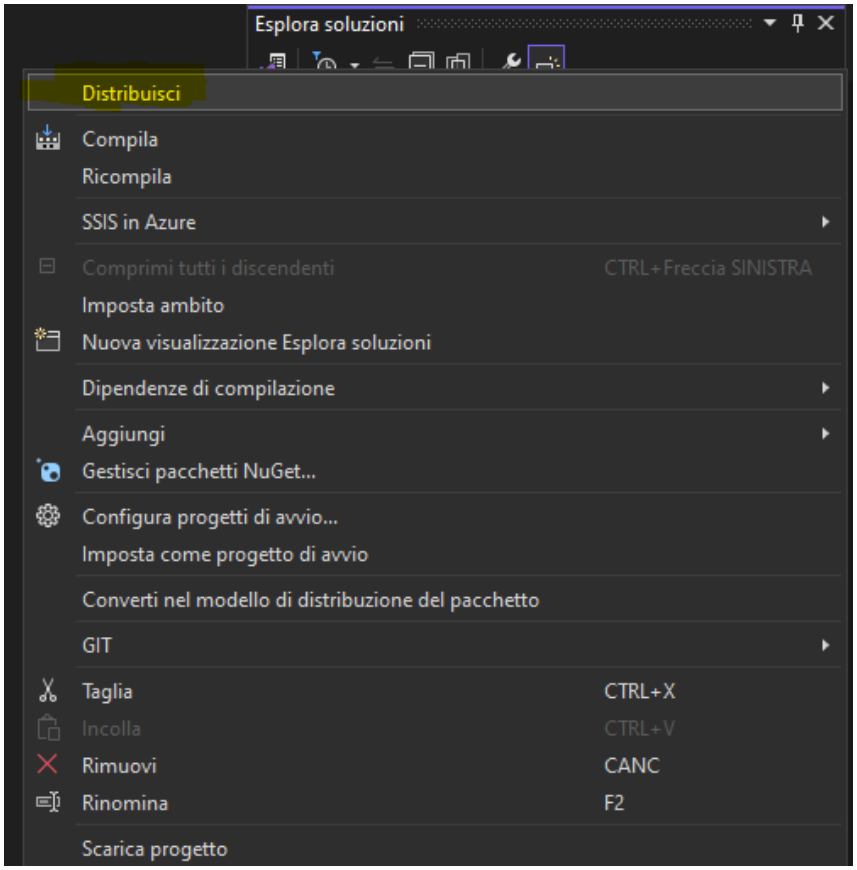
2. The first time we need first to create the following folders in the SSIS Catalog:



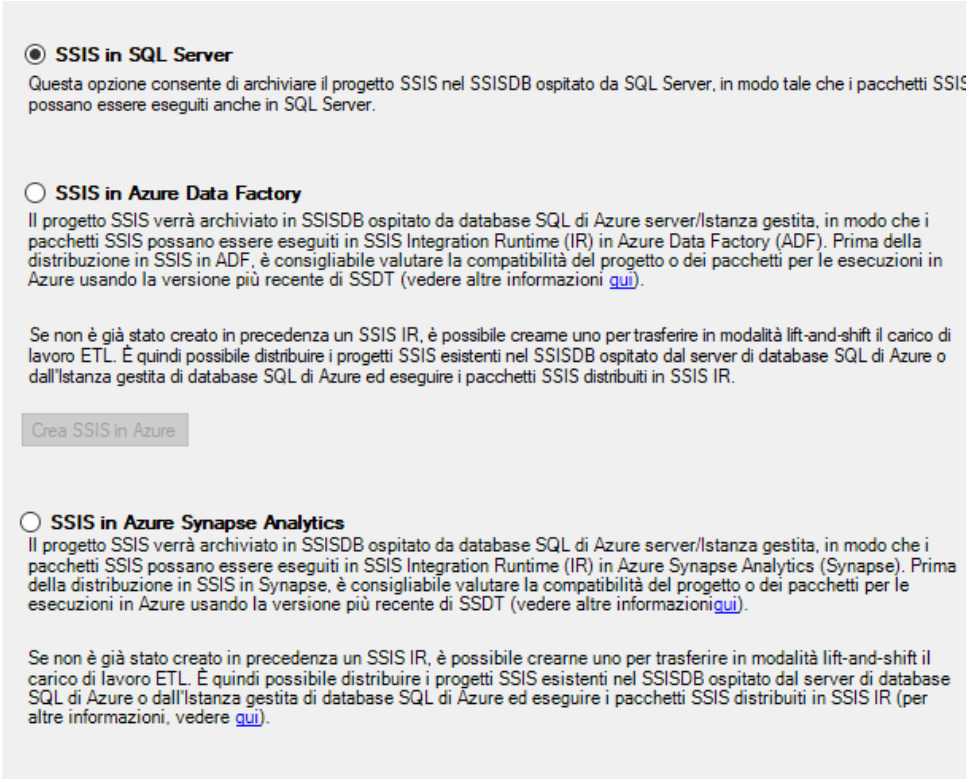
To create a folder, click on *Integration Services Catalog*, right click on *SSISDB* then click on *Create Folder*:



3. Once we built each project and we have created the folders, for each project do the following steps (I'll show only of the Import project, do the same for Cleaning, SA and DM):
   - Right click on the project, then click on *Release*:

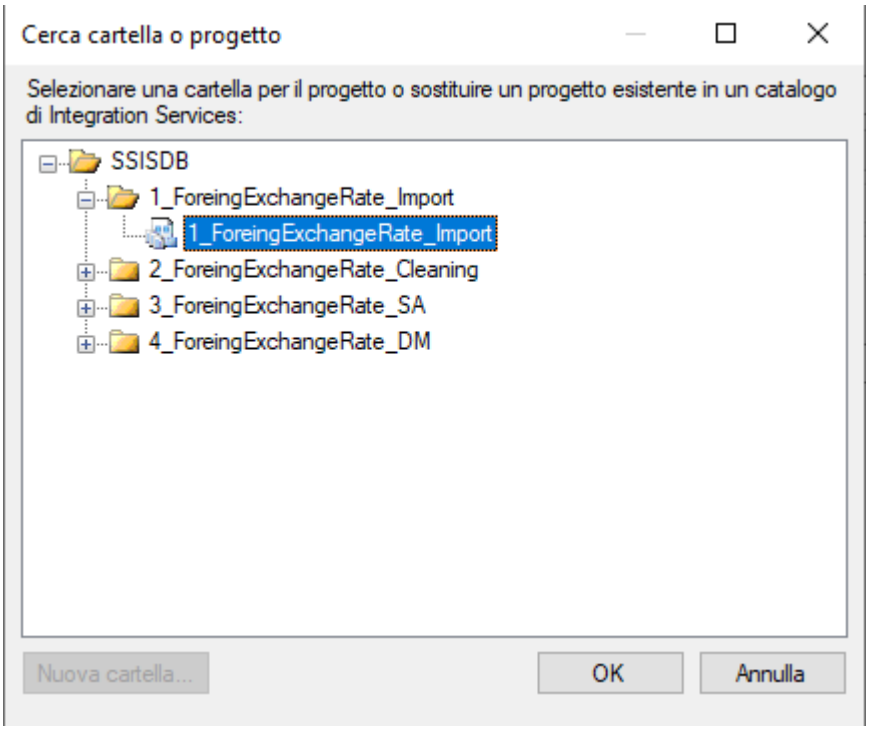


- Choose: *SSIS in SQL Server*



- Connect to localhost (or servername) by using *Windows Authentication*:

- Click on *Browse*...
- Select the project that we are going to release (Import in our case):

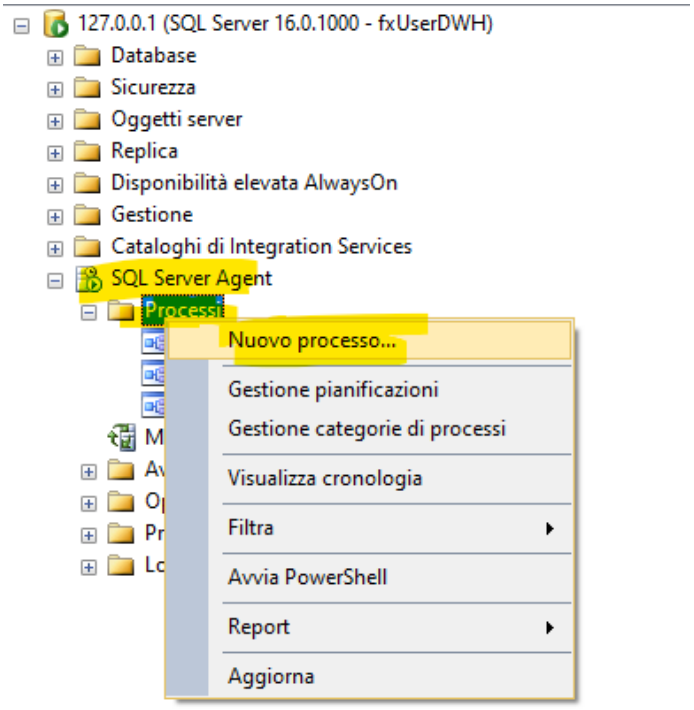


- Click twice on *Next*

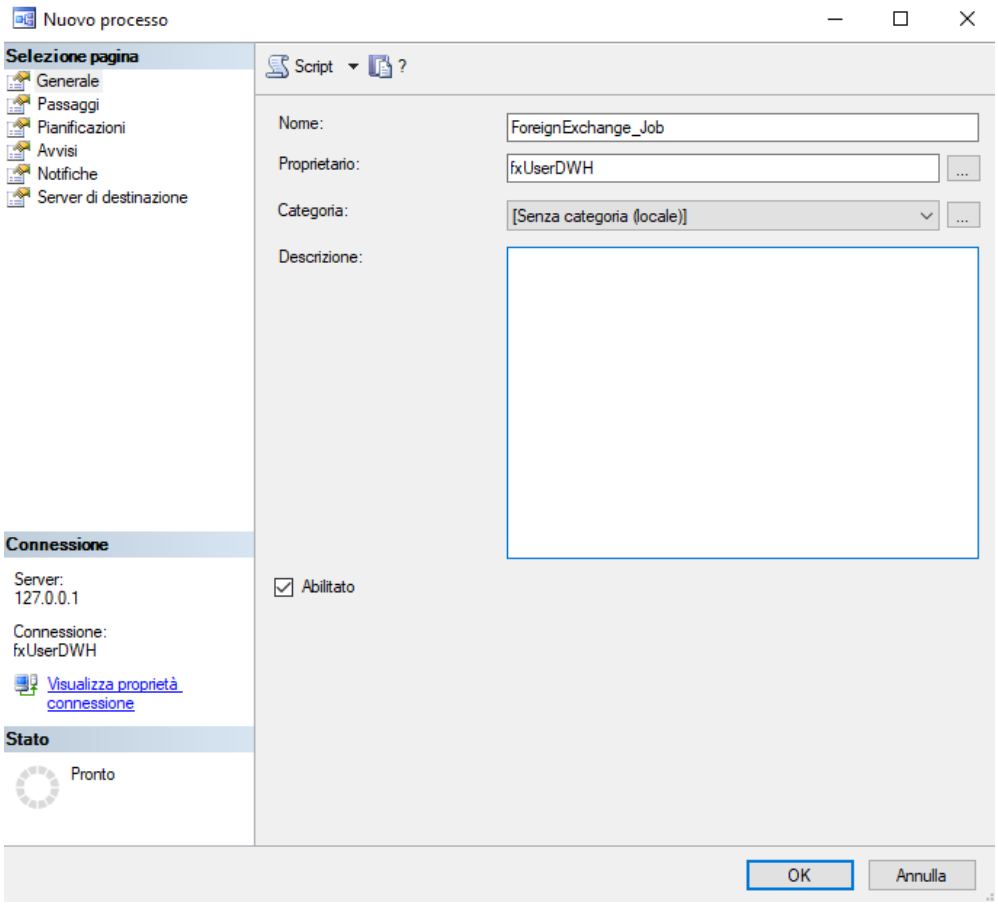
Schedule the job

This is an important step because one of the requirements ask to schedule the job every hour (or every minute). We can create a job from SSMS with the following steps:

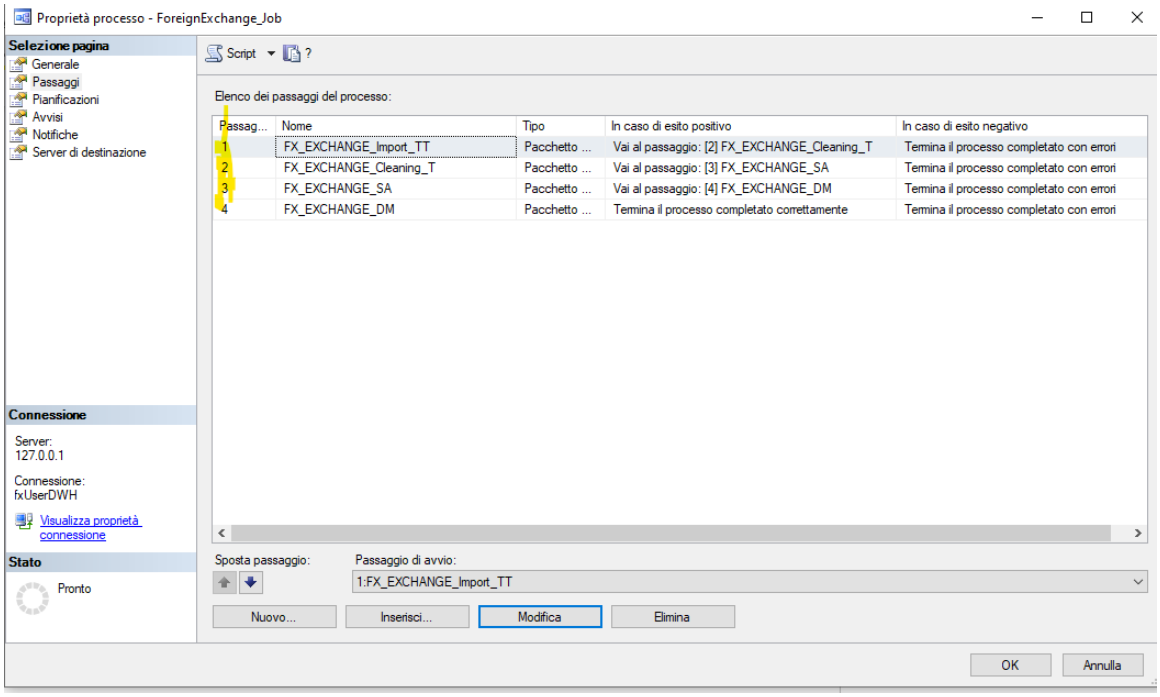
1. Create a new process: Click on *SQL Server Agent* then right click on the *Processes* folder, click on *New Process*:



2. Give a name to the job (for example, *ForeignExchange_Job*):

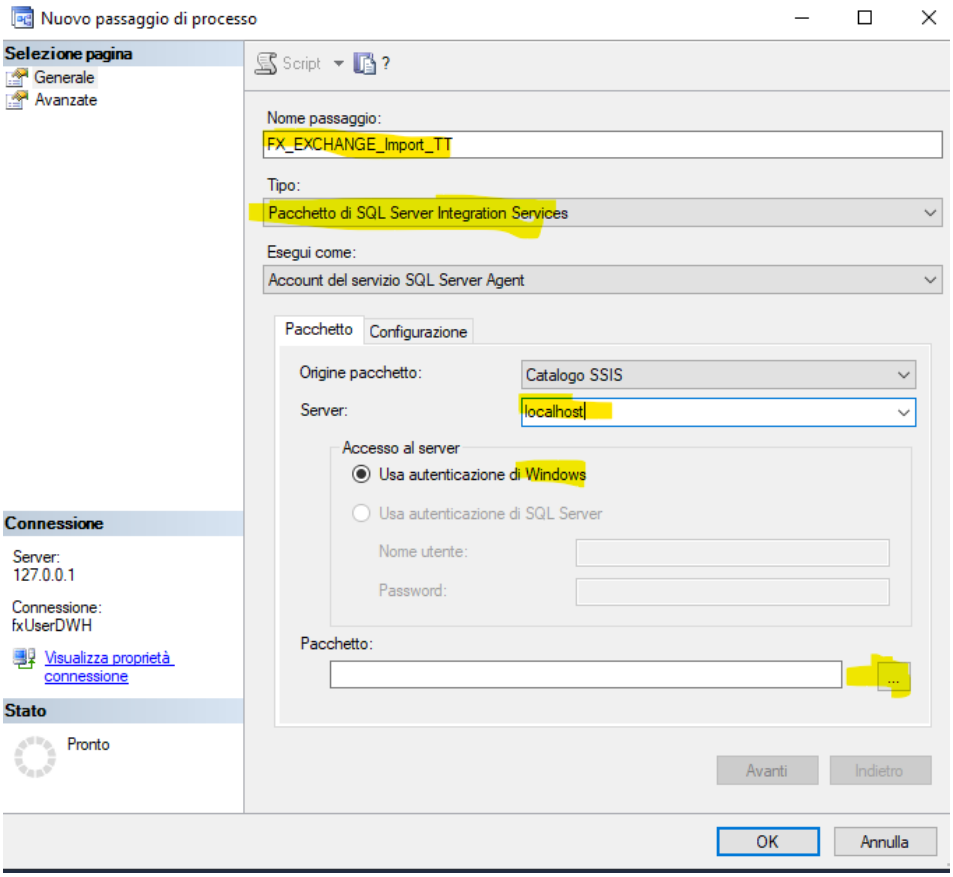


3. Create the following steps:

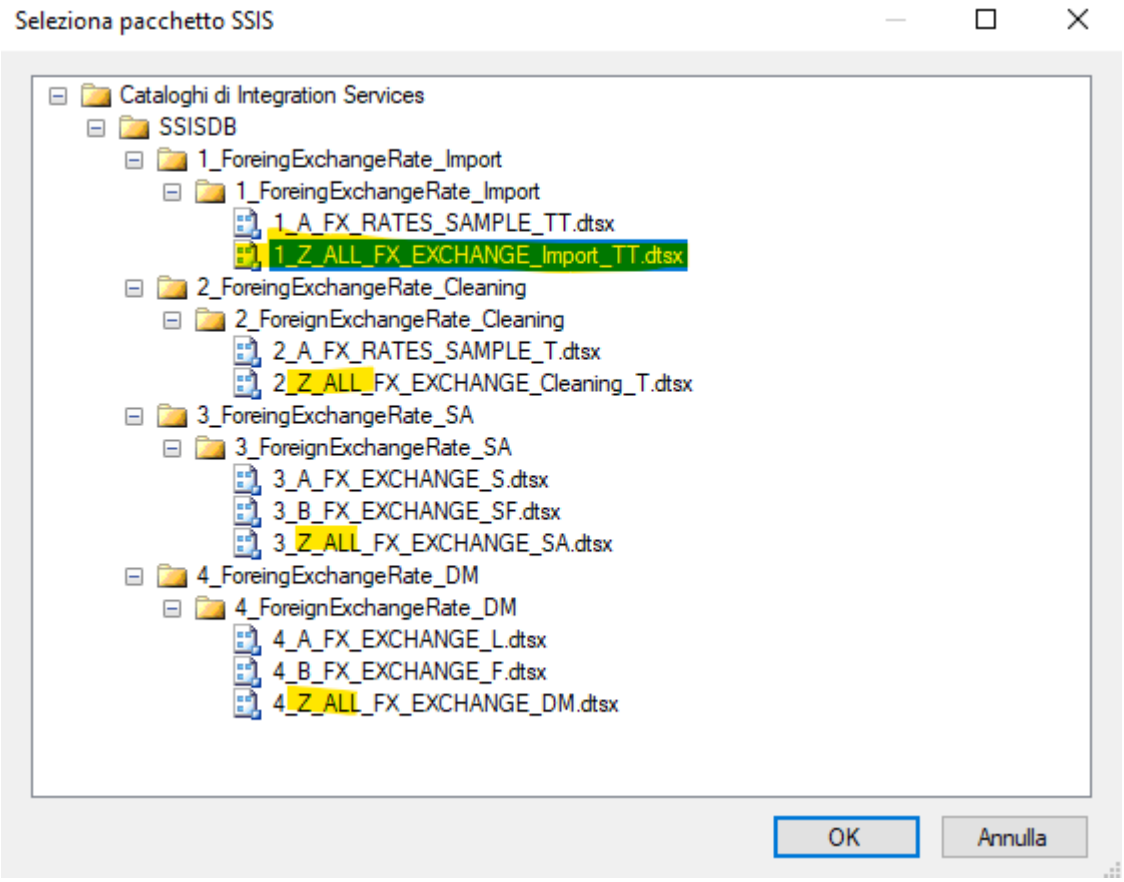


To create a step:

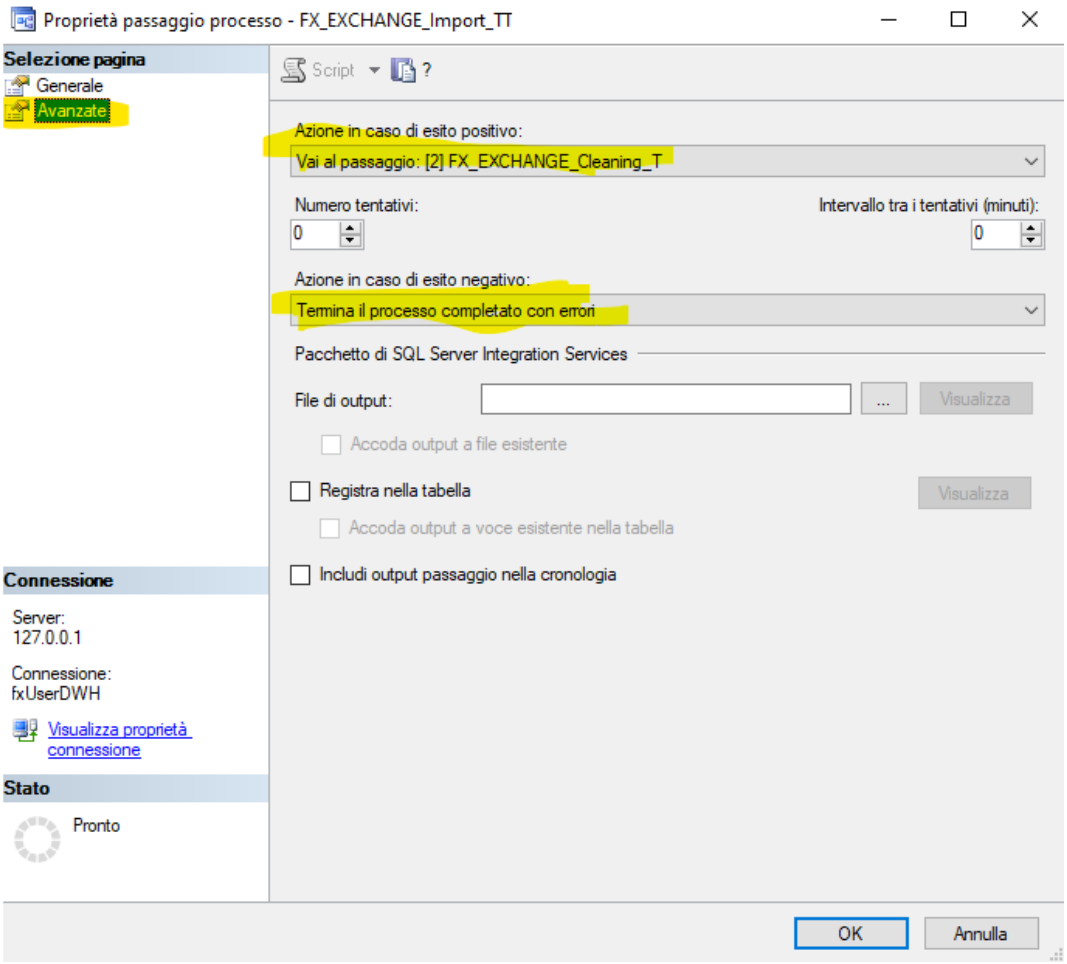
- Click on *New*
- Give a name to the step (in the guide we do only the first one that is *FX_EXCHANGE_Import_TT*), Select *SQL Server Integration Services* as Type, enter the localhost (or servername) as server name, verify that Windows Authentication is checked, click on the three dots



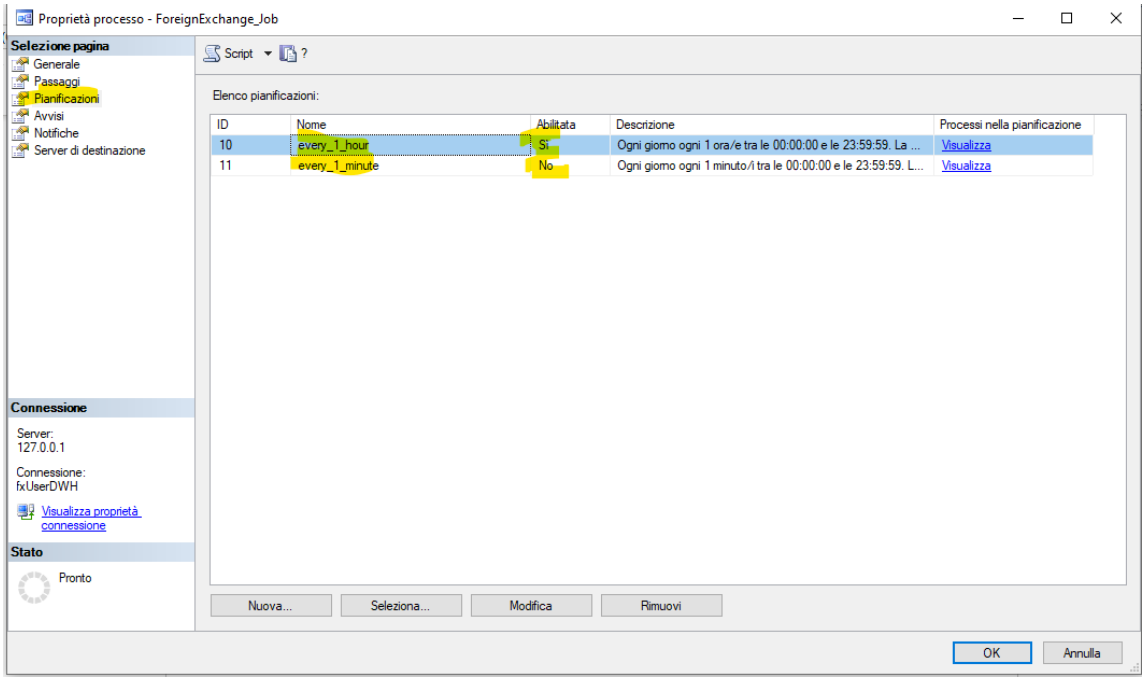
- Once you clicked on the three dots, select the right package. In this case, *1_Z_**ALL**_FX_EXCHANGE_Import_TT.dtsx*. Note! always choose for each project the package with the "*ALL*" suffix, this is responsible for calling all the others:



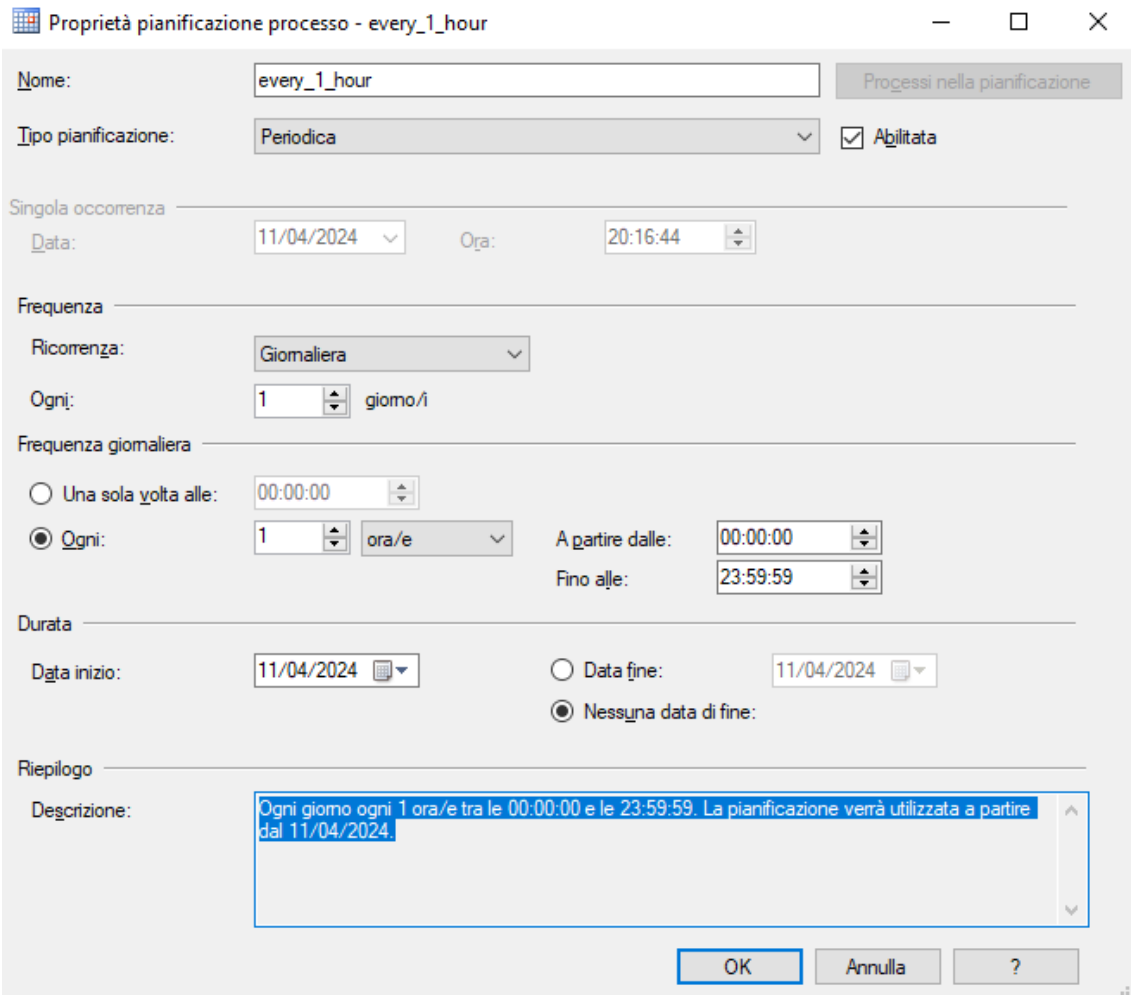
- Go to *Advanced Properties* and set the action in case of positive or negative execution as follows (see point 3):



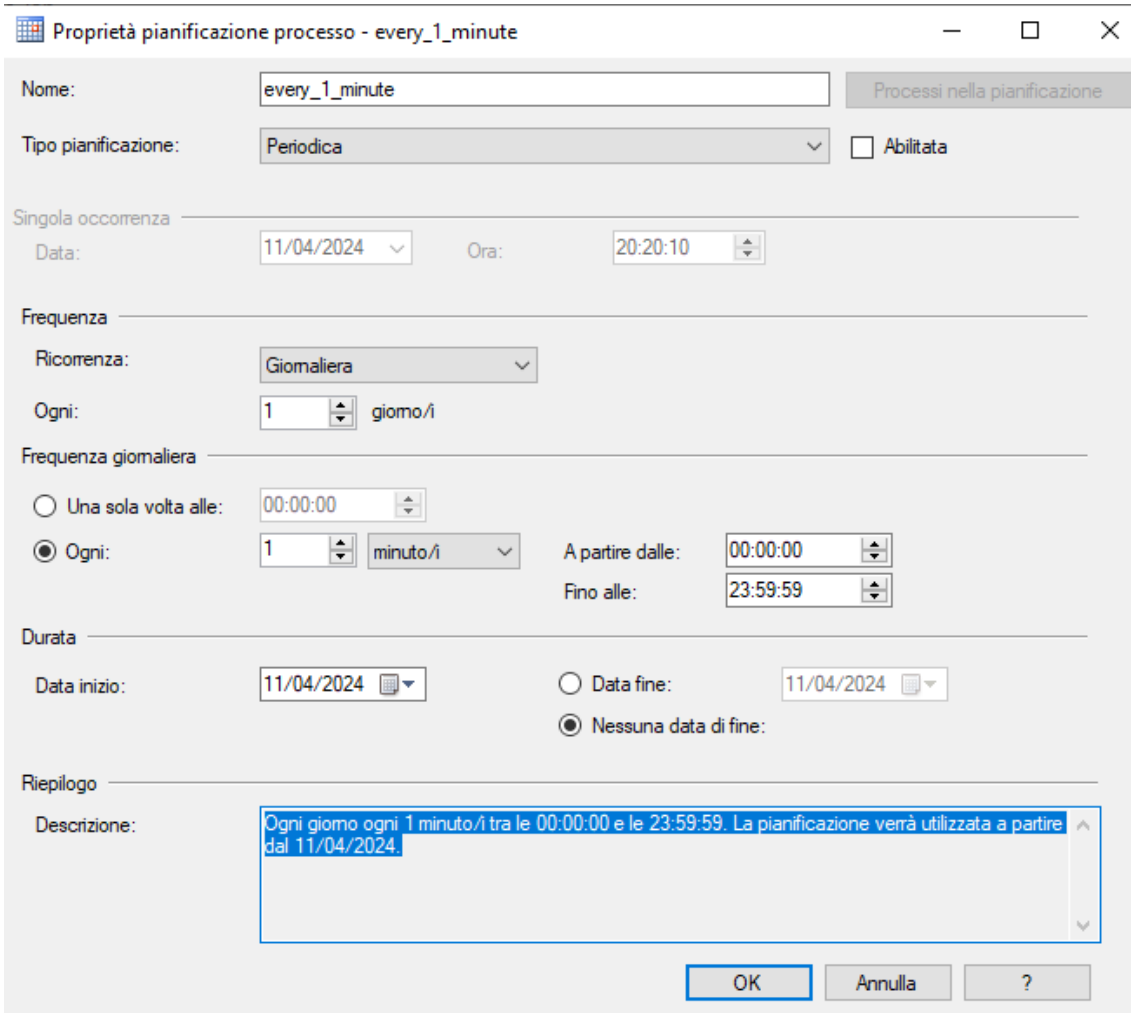
4. In the schedule tab, create the two schedules with name *every_1_hour* (enabled) and *every_1_minute* (disabled). It is easy to switch from 1 hour schedule to 1 minute schedule just by modifying to schedules and disable *every_1_hour* and enable *every_1_minute*.



To create every one hour set as follows: name *every_1_hour*, periodic, enabled, daily recurring every 1 day, every hour from 00:00:00 to 23:59:59



To create every one hour set as follows: name *every_1_minute*, periodic, disabled, daily recurring every 1 day, every minute from 00:00:00 to 23:59:59



Everything is ready now once the job has been executed at least once, run the query in the *output.sql* file to see the output in the following format:

For the LED screen to display information, it must be fed with input in the below format:

```
ccy_couple,rate,change
"EUR/USD",1.08081,"-0.208%"
```

Requirements

REQUIREMENT ID	REQUREMENT NAME	Satisfied	DESCRIPTION
REQ_001	DisplayRate	✅	Display for each currency pair the FX exchange rate
REQ_002	DisplayChangePercentage	✅	Display for each currency pair the percentage of the change compared to yesterday's rate at 5PM New York Time
REQ_003	HighFrequency	✅	Rates are received in high frequency, milliseconds

REQ_004	OutputFormat	✅	The output must have the following format:
			`ccy_couple,rate,change`
			`"EUR/USD",1.08081,"-0.208%"`

REQ_005	JobSchedule	✅	Schedule a job which runs every 1 hour and have the possibility to change the schedule to run every 1 minute instead
REQ_006	ActiveRates	✅	The job should consider only "*active*" rates. What "*active*" means will be specified in the business logic during the design phase (for currency pairs that don't have an "*active*" rate, no output should be produced)
REQ_007	GenericSolution	✅	In the example file there are 5 currency pairs but the job should work also if we assume 300 currency couples (or even more)
REQ_008	RealTimeStreaming	⚠ Not yet	It's an optional requirement. Instead of a batch, we would like to change the process into a streaming one and display data in real time