FactoryTalk Optix First steps

Contents

1. Learning Ressources 4
2. Using web Help 5
3. Using integrated help 5
4. Configuring Tags from PLC access 6
5. Displaying a PLC Tag 10
6. Web access 14
7. Navigation Panels 16
8. PopUp or DialogBox Windows 22
   8.1. PopUp / DialogBox from Button 22
   8.2. PopUp / DialogBox from variable 28
   8.3. PopUp / DialogBox from PLC value 33
   8.4. Dragable DialogBox 33
9. Reusable Objects. Working with Alias 38
10. Communications with Micro8XX PLC's 47
11. First steps with OptixPanel 52
   11.1. OptixPanel Widgets 58
   11.2. Adding OptixPanel Widgets to a standard application 65
12. First steps with ASEM6300B PC and runtime 81
13. First steps with OptixEdge 86
   13.1. Wizard 86
   13.2. Device Manager 87
   13.3. Deploy your first Optix Application 87
   13.4. Testing PLC Tag access 89
   13.5. Portainer 89
   13.6. How to deploy a Softing EdgeConnector on OptixEdge 93
14. Users in Optix 96
   14.1. Creating users and Groups manually 96
   14.2. Create a Login form 98
   14.3. Writing the logged user to the PLC 100
   14.4. Creating Groups in design time 104
   14.5. Creating Users in Design time 111

14.6. User batch creation in design time (100 users) ... 118
14.7. Manage user rights to Panels in a Native way ... 118
14.8. Making Login Form invisible when a user is Logged in ... 126
14.9. Make Panels visible if user belongs to one of a set of Roles ... 132
14.10. User management with landing panel ... 139
14.11. Manage user rights to Panels on a NavigationPanel in C# ... 140
14.12. Registering logged user on a database ... 143
14.12.1. Adding the Logged user to a variable ... 143
14.13. Alphabetical order on the Users ComboBox ... 150
15. Datalogger and Writing to an SQL database ... 151
15.1. Datalogger. Working with an embedded database ... 151
15.2. Datalogger Trigger on Tag Transition ... 152
15.3. Let's write to an external SQL database from Optix ... 152
16. OPC UA Client ... 173
17. Creating a OPC UA to SQL converter ... 180
18. Configuring an application as an MQTT client ... 186
19. PLC data and MQTT ... 196
19.1. Asynchronous publish ... 215
20. Subscribing to an MQTT broker with user and password. TTN example ... 218
21. Creating an Http REST API client ... 223
21.1. GET request ... 223
21.2. Installing InfluxDB on windows ... 234
21.3. Inject your first data with node-red ... 236
21.4. Injecting to InfluxDB from command line ... 238
21.5. POST request to local InfluxDB ... 241
21.6. POST request to remote InfluxDB ... 244
22. Simpler REST API ... 245
22.1. Simple InfluxDB http POST request ... 253
23. Local External InfluxDB in FTOptix v1.4 ... 255
24. Cloud External InfluxDB in FTOptix v1.4 ... 263
25. Using Github to store and share your repositories on the cloud ... 263
26. Pushing your local Optix projects to Github ... 274
26.1. Using Optix ... 274
26.2. Using Git ... 292

26.3. Remote repository creation with FT Optix version 1.3 302
27. Sending Telegram messages 306
28. Creating, importing C# .Net libraries to your environment 307
28.1. Design time libraries. Importing Modbus Tags from a CSV File 307
28.2. Runtime libraries 316
28.3. How to use .NET third party libraries in our projects 321
28.3.1. Installing Nuget packages in Visual Studio Code 327
28.3.2. With Visual Studio 2022 community edition 334
28.3.3. Complete your MQTTnet application 339
29. NetLogic and C# tutorial 342
30. Understanding compilation of NetLogic code 343
31. Understanding Classes (Objects), methods and data 346
32. Creating a Method 348
33. Linking events with Methods without input parameters 352
34. Linking events with Methods with input parameters 355
35. Asynchronous tasks, creating callback functions 357
36. Periodic Tasks 362
36.1. Native Periodic Tasks 362
36.2. Netlogic Periodic Tasks 366
37. Log info to emulator output 367
38. Accessing variables 367
39. Accessing objects 374
40. Using Owner 381
40.1. Accessing object using owner 381
40.2. Detecting change on object using Owner 384
41. Accessing PLC Tags in NetLogic 386
41.1. Method0 387
41.2. Method1 389
41.3. Method 2 392
41.4. Writing to an array of PLC values 395
42. Object events 398
43. Open a cmd terminal to run a command or program 404
44. Socket TCP using System.Net.Sockets without external libraries 409
45. How to define NetLogic Parameters 442

46. Using third party dll libraries in FactoryTalk Optix 445
46.1. Using third party libraries in Visual Studio 2022 windows Forms Framework 445
46.2. Thirt party libraries (dll) Socket Server application with Optix 461
46.3. Thirt party libraries (dll) Socket Client application with Optix 472
46.3.1. Installing external references (dll libraries) into Visual Studio project 2022 477
46.3.2. Installing external references (dll libraries) into Visual Studio Code project 485
47. Using NuGet libraries (HiveMQ .Net example) 487
48. Modbus communications 499
48.1. Modbus server with FTOptix 499
48.2. Native Modbus TCP client with FactoryTalk Optix 503
48.3. DotNet ModbusTCP client with C# and FactoryTalk Optix 504
48.4. Modbus TCP to OPC UA converter 514
48.5. Modbus TCP to EtherNet/IP Gateway 514
48.6. Modbus to MQTT data aggregator 516
49. Communicating Optix with HARDY 6200 weight processor 517
49.1. TCP/UDP communications 517
49.2. Node-RED 520
49.3. TCP/IP with FT Optix 523
49.4. Modbus TCP 527
50. Generating a pdf file from C# 527
50.1. Using Foxit Pdf 529
51. Using BoilerDemo 531

First steps with FT Optix

1. Learning Ressources

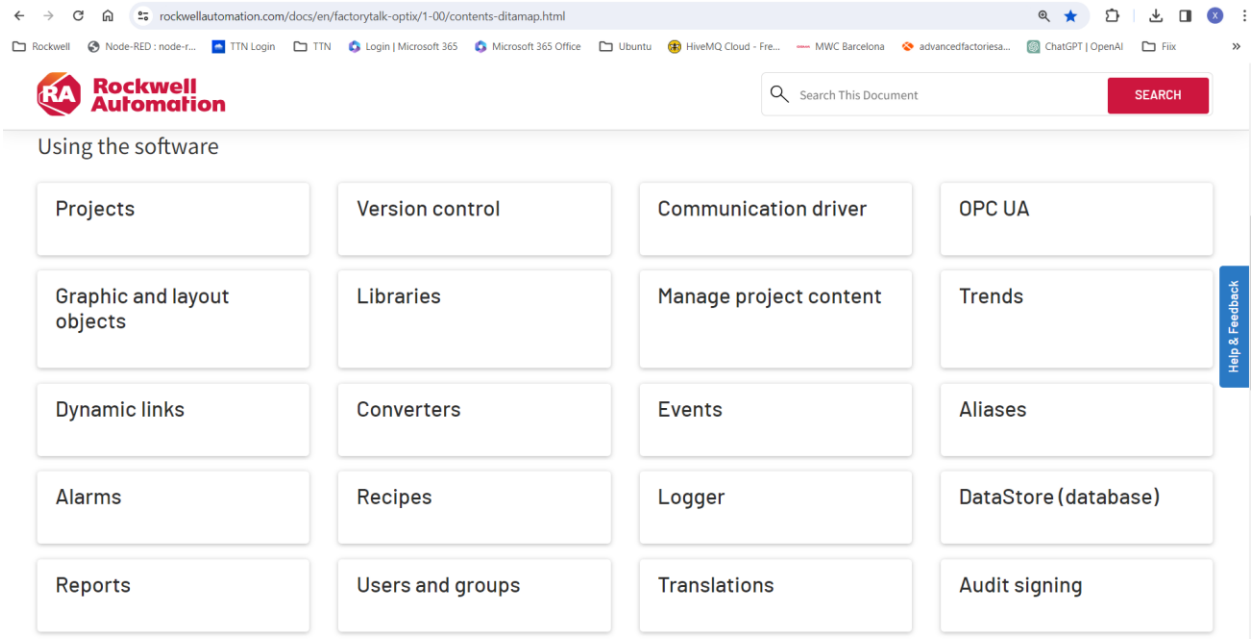
Blogs
https://wawelk.org/optix/
https://github.com/massimovar/LearningFTOptix/tree/main

Video Channels
ASEM
https://www.youtube.com/watch?v=G_PphV37cbQ&list=PLdrWYH7B3F3566vPxCiw3HmPZ56HUIVkN
Rockwell Automation

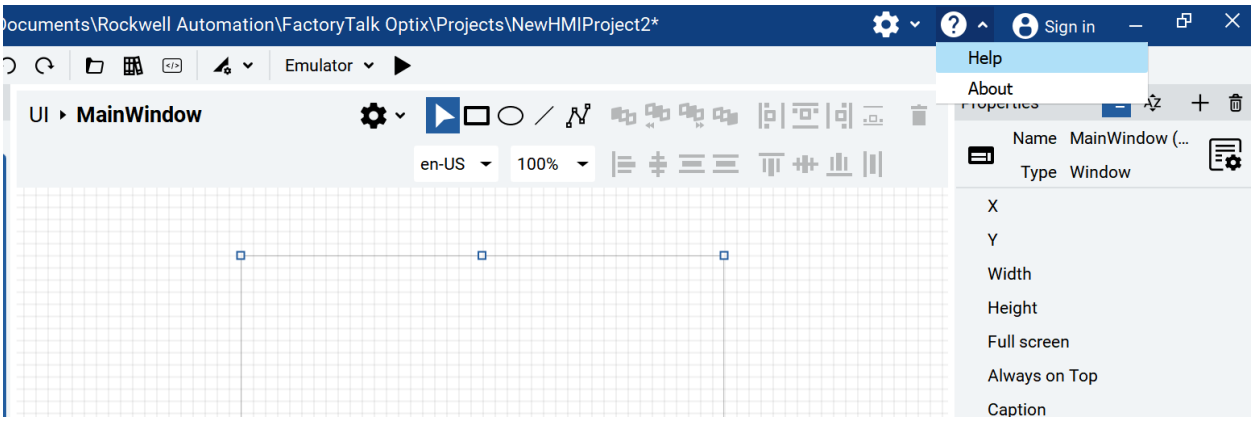
https://www.youtube.com/watch?v=15hHpvhzwzw&list=PL3K_BigUXJ1M1-JpRiwIIhzJUbhwtK3yy
Xavier Florensa
https://www.youtube.com/@xavierflorensa

2. Using web Help

https://www.rockwellautomation.com/docs/en/factorytalk-optix/1-00/contents-ditamap.html

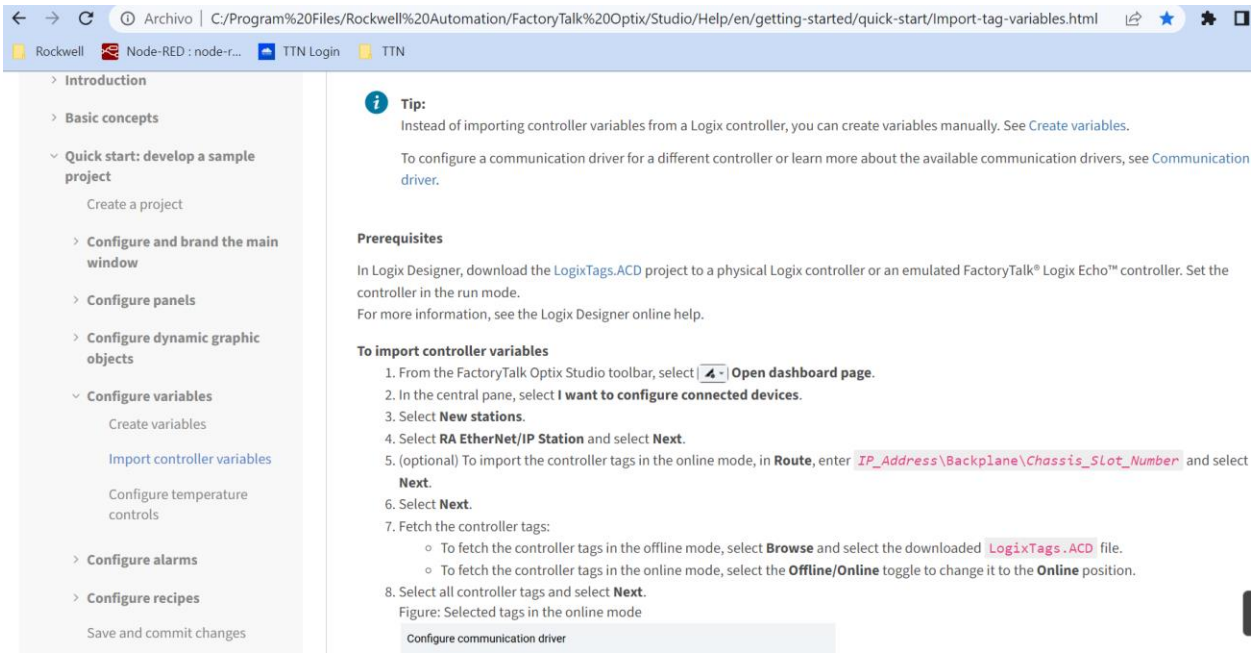


3. Using integrated help

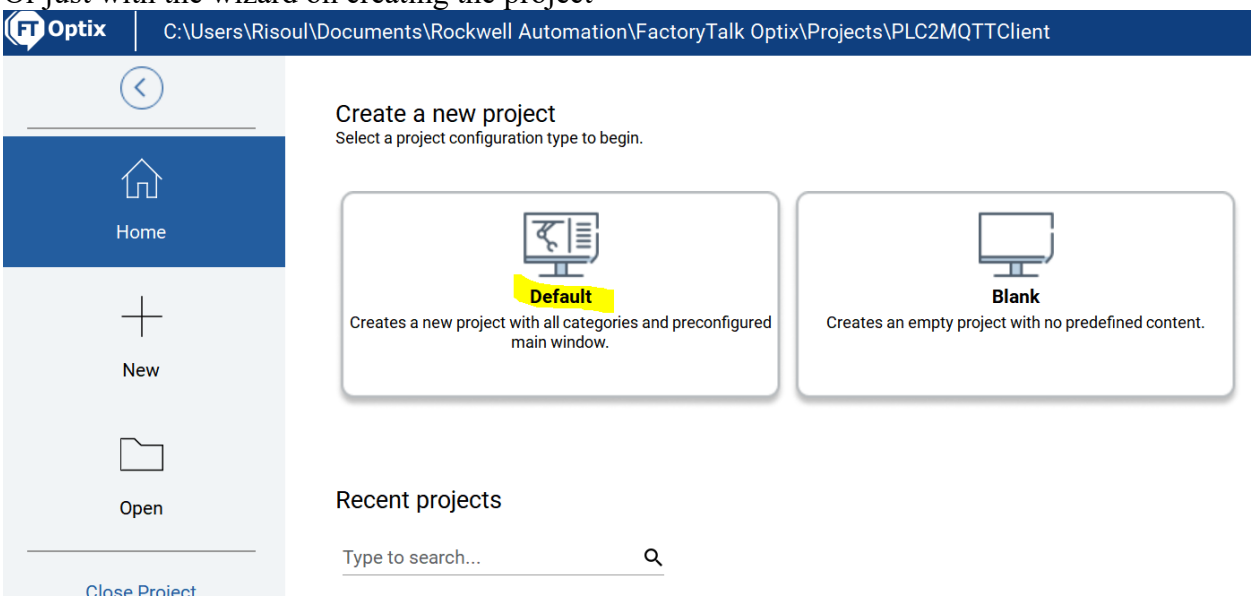


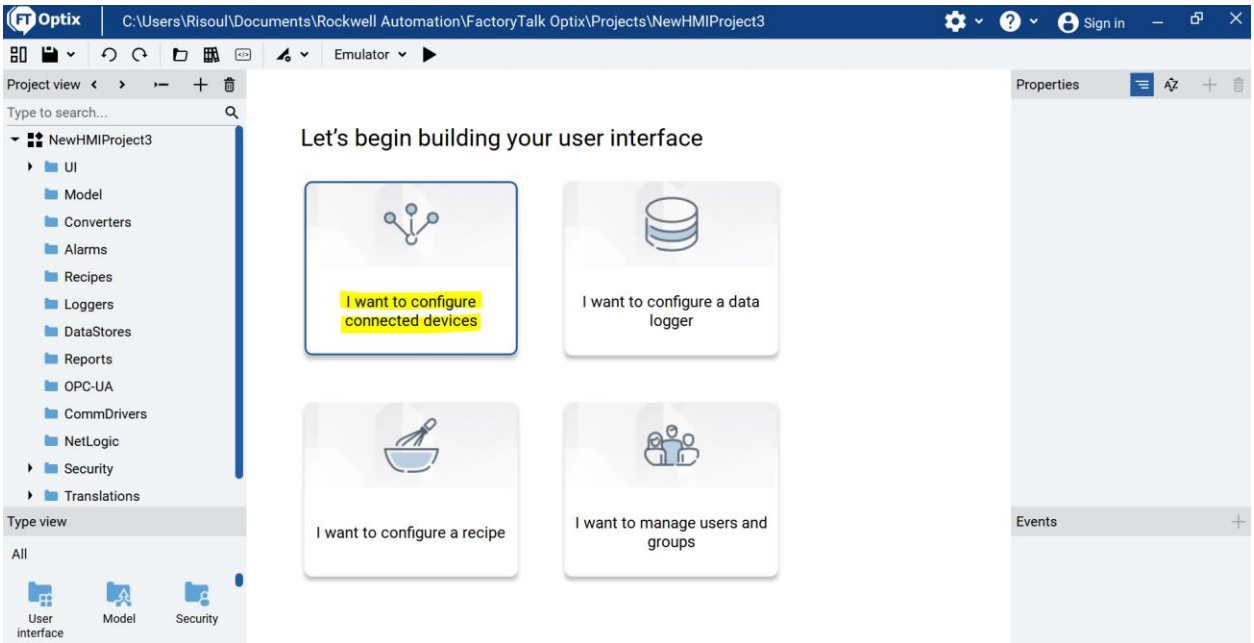
4. Configuring Tags from PLC access

You can do as on the manual
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/getting-started/quick-start/configure-variables/import-tag-variables.html

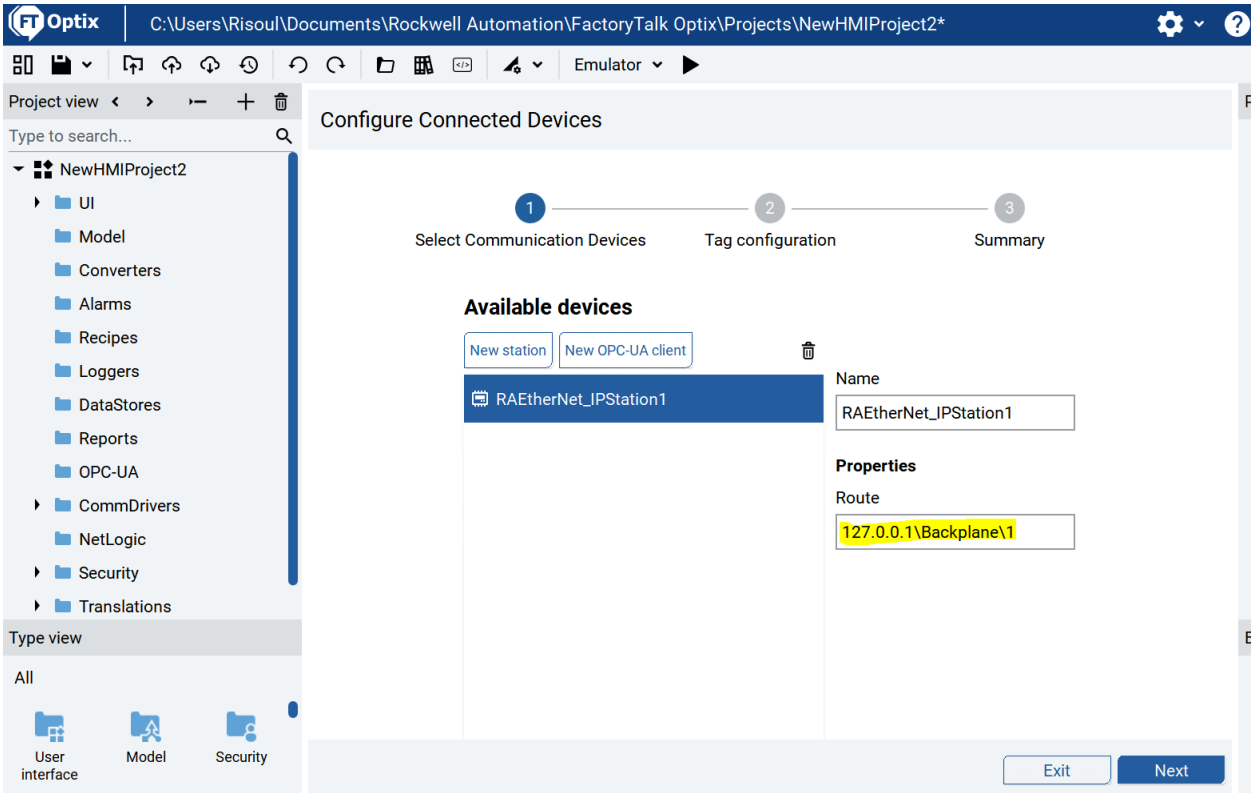


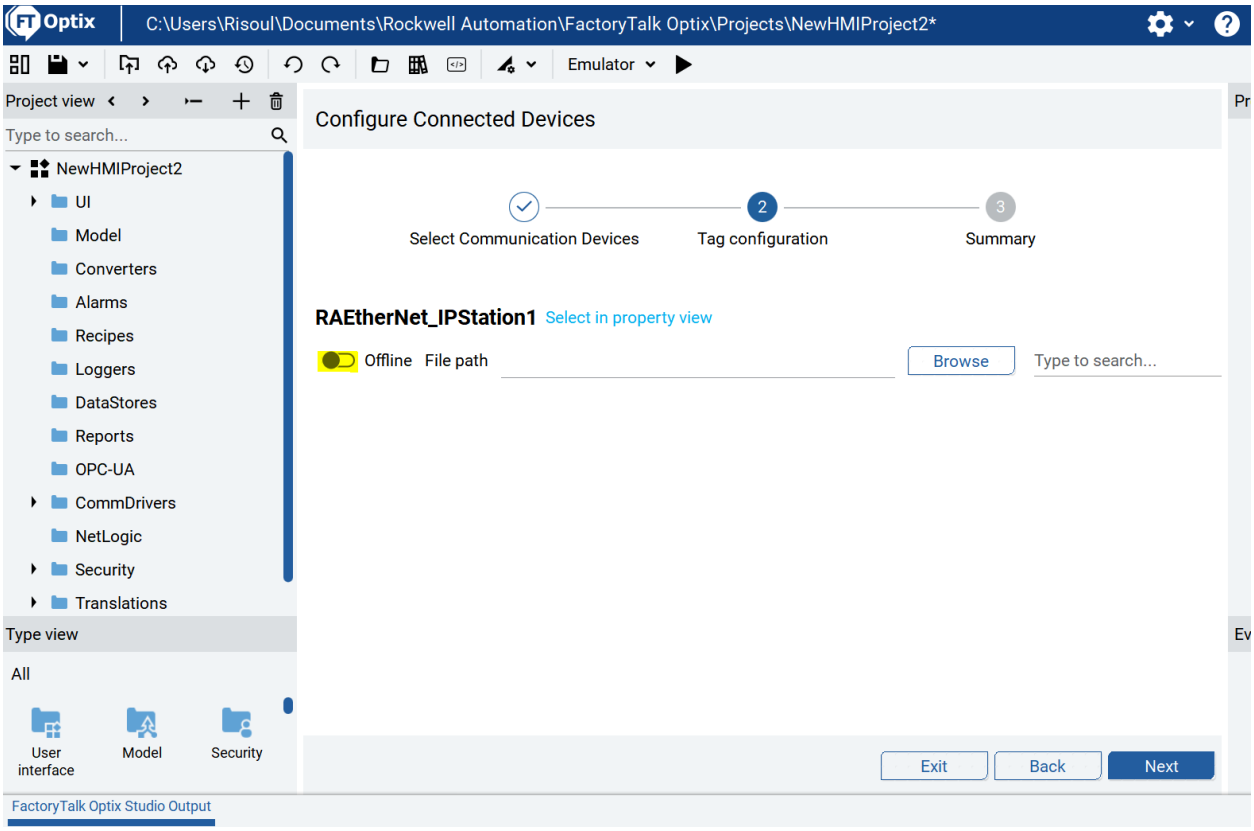
Or just with the wizard on creating the project

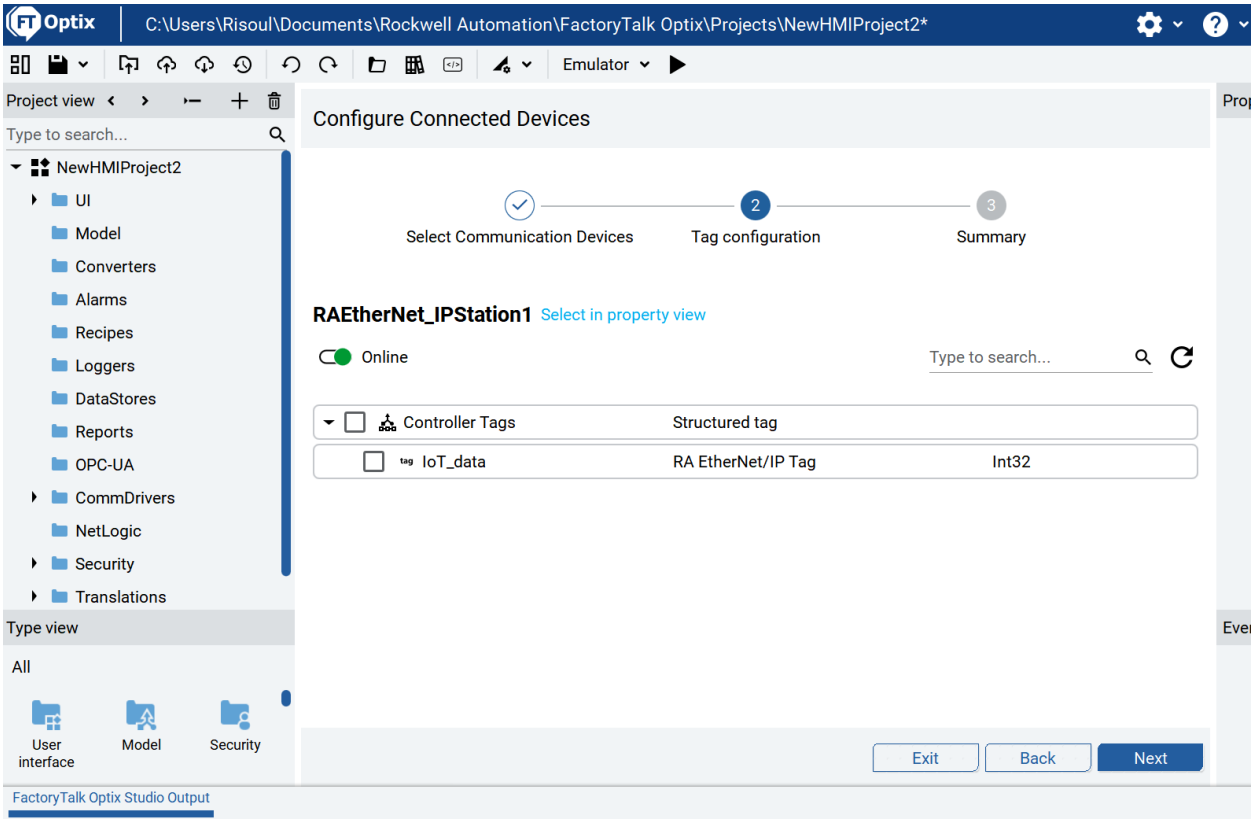


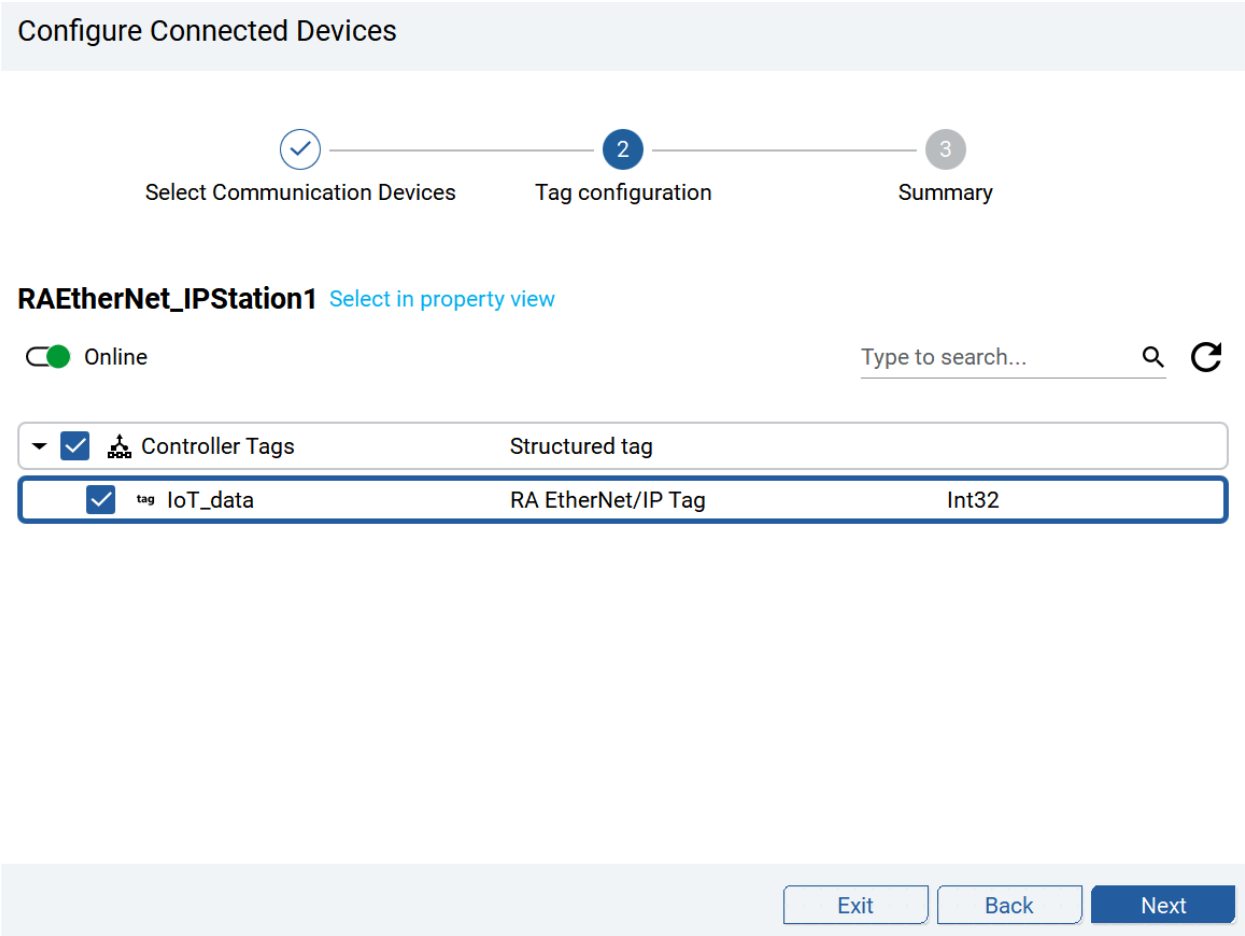


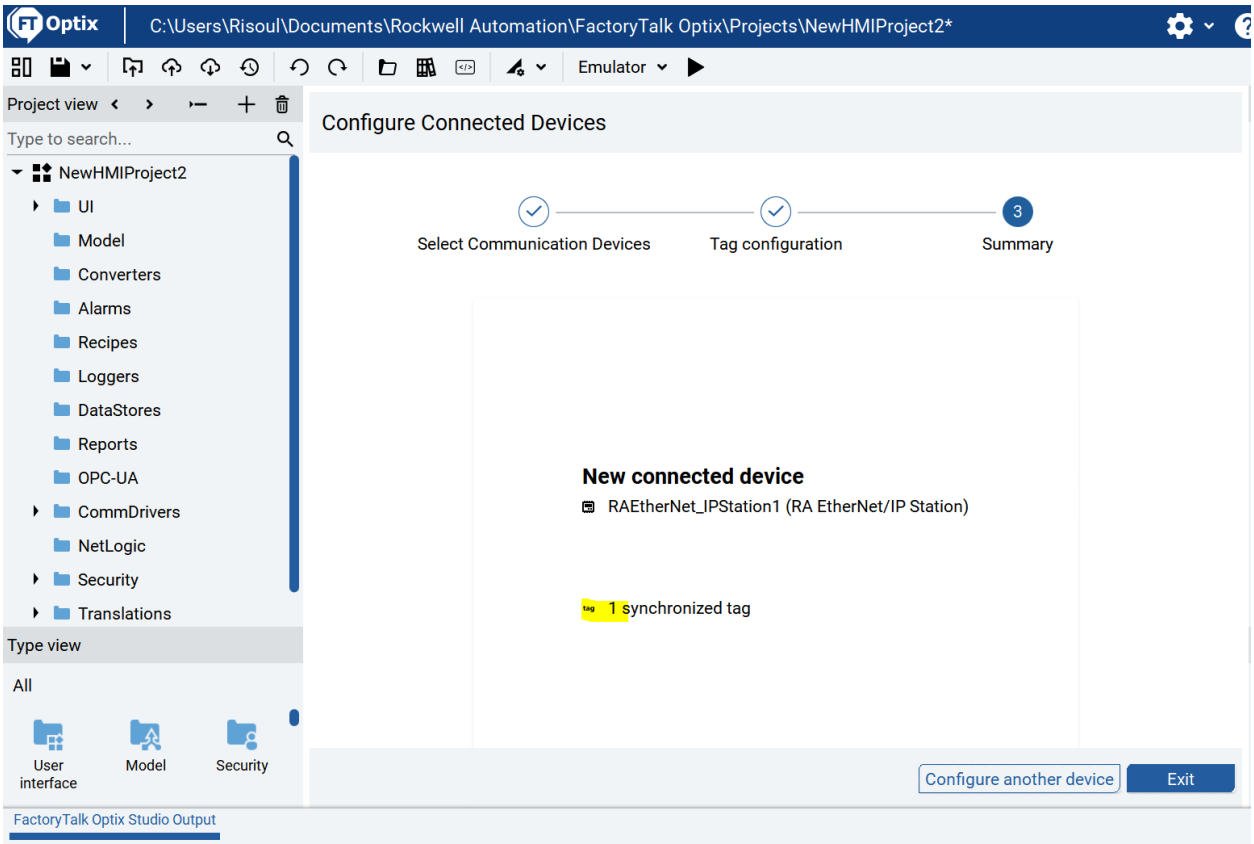
On both cases you will arrive to this point




Optix
C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\NewHMIProject2*
Emulator
Project view
Type to search...
NewHMIProject2
UI
Model
Converters
Alarms
Recipes
Loggers
DataStores
Reports
OPC-UA
CommDrivers
NetLogic
Security
Translations
Type view
All
User interface
Model
Security
Configure Connected Devices
Select Communication Devices
Tag configuration
Summary
RAEtherNet_IPStation1
Select in property view
Offline
File path
Browse
Type to search...
Exit
Back
Next
FactoryTalk Optix Studio Output

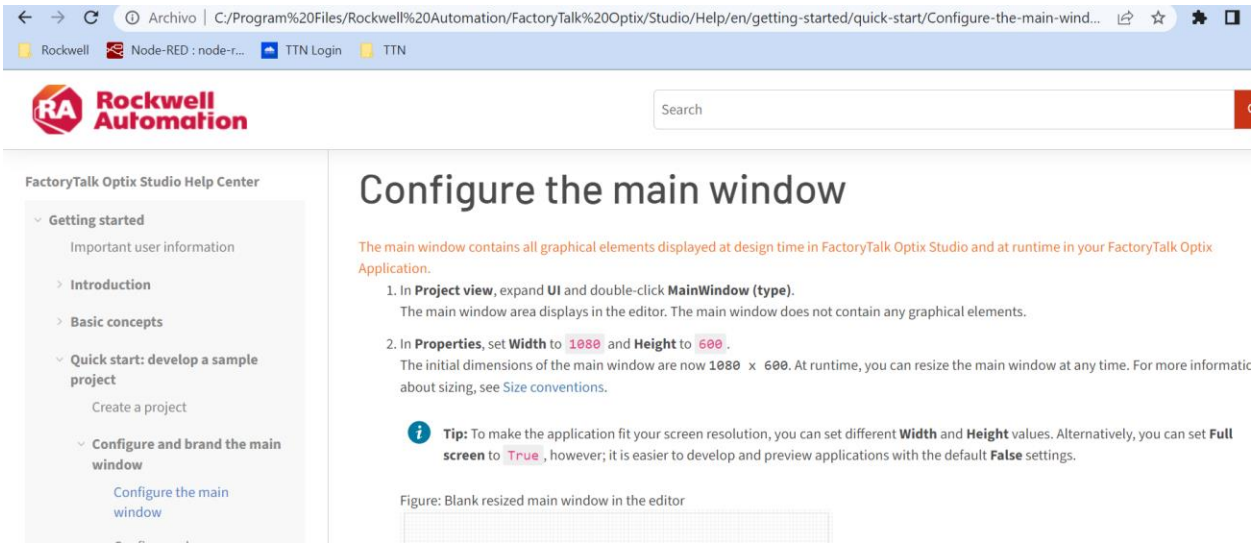

Optix
C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\NewHMIProject2*
Emulator
Project view
Type to search...
NewHMIProject2
UI
Model
Converters
Alarms
Recipes
Loggers
DataStores
Reports
OPC-UA
CommDrivers
NetLogic
Security
Translations
Type view
All
User interface
Model
Security
Configure Connected Devices
Select Communication Devices
Tag configuration
Summary
RAEtherNet_IPStation1
Select in property view
Online
Type to search...
Controller Tags
Structured tag
IoT_data
RA EtherNet/IP Tag
Int32
Exit
Back
Next
FactoryTalk Optix Studio Output


Configure Connected Devices
2
3
Select Communication Devices
Tag configuration
Summary
RAEtherNet_IPStation1
Select in property view
Online
Type to search...
Controller Tags
Structured tag
IoT_data
RA EtherNet/IP Tag
Int32
Exit
Back
Next



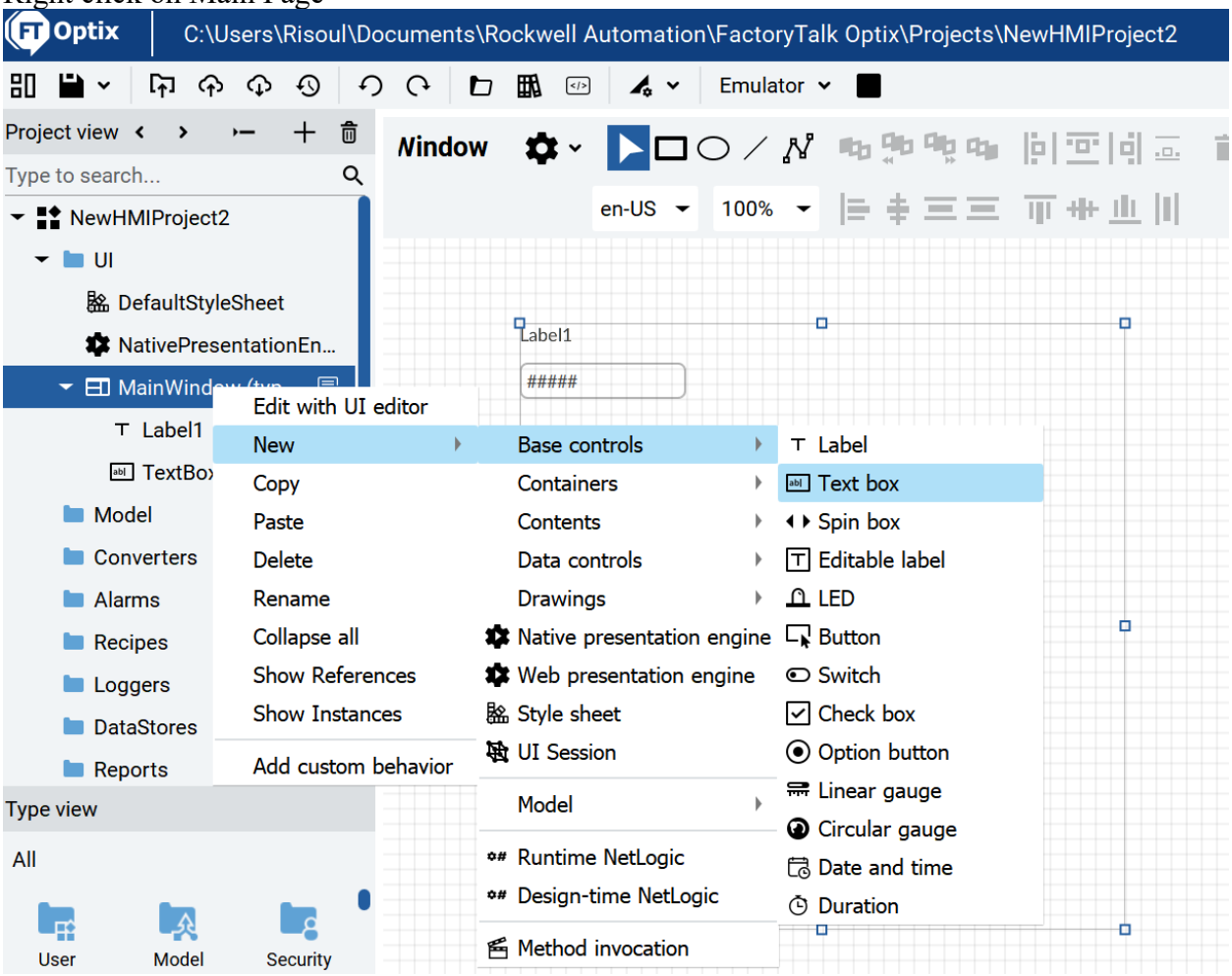
5. Displaying a PLC Tag

https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/getting-started/quick-start/configure-and-brand-the-main-window/configure-the-main-window.html

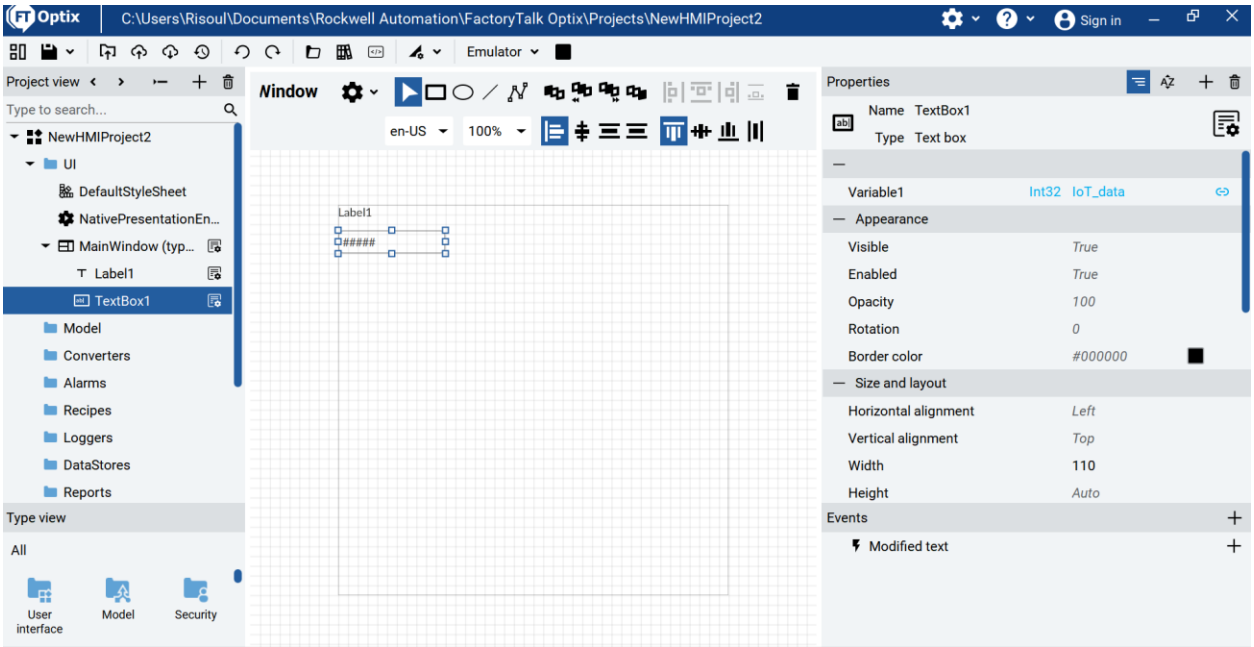


Let’s prepare the page with some controls

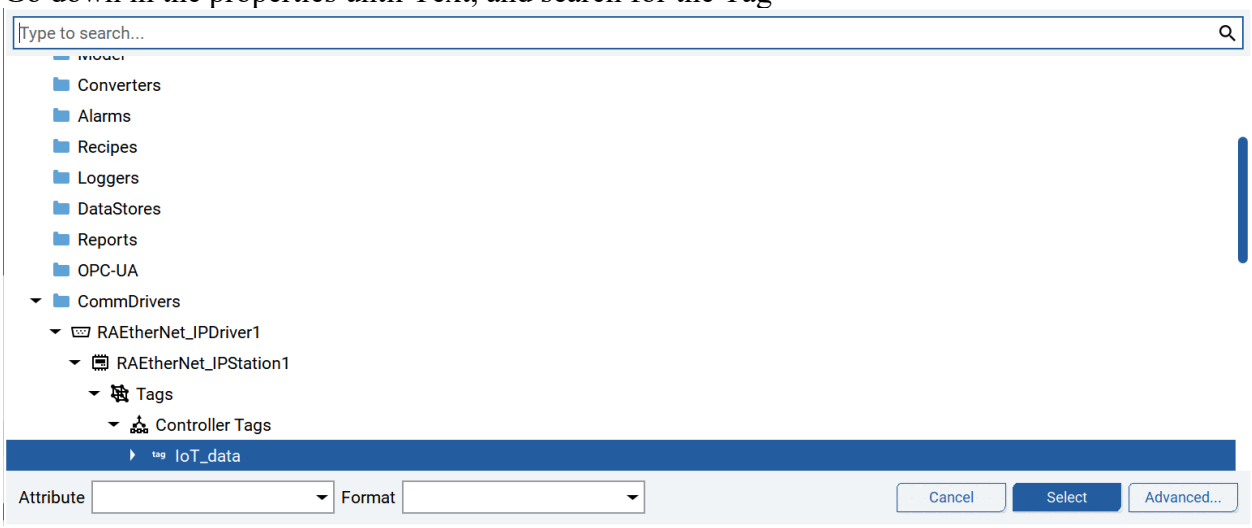
Right click on Main Page



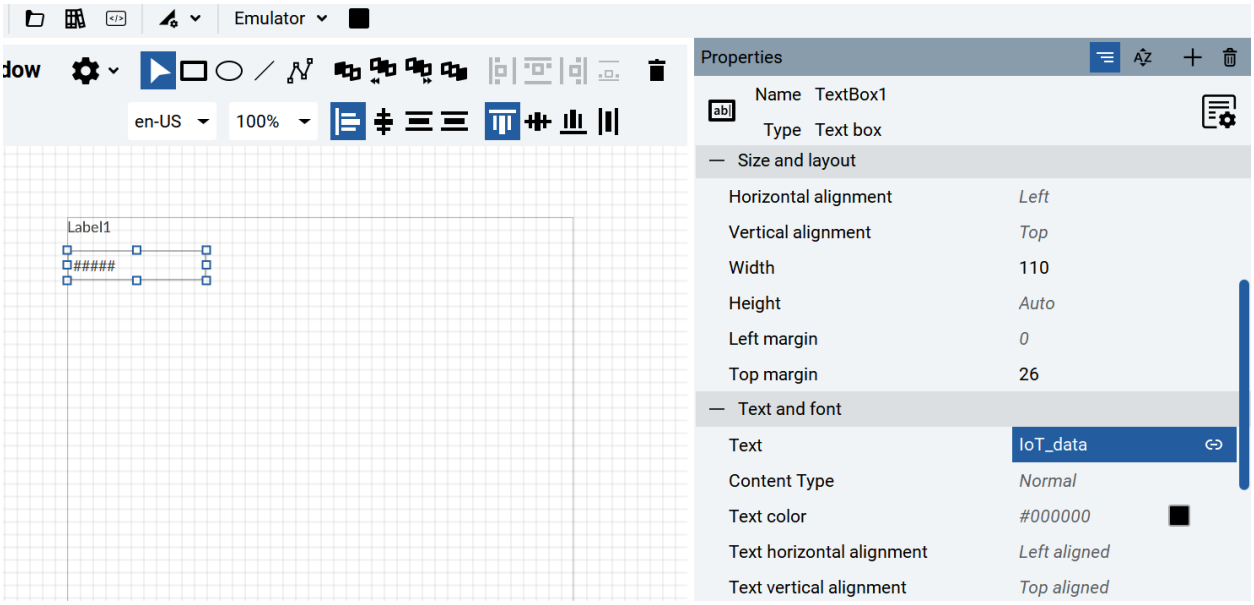
Double click on the object on the page



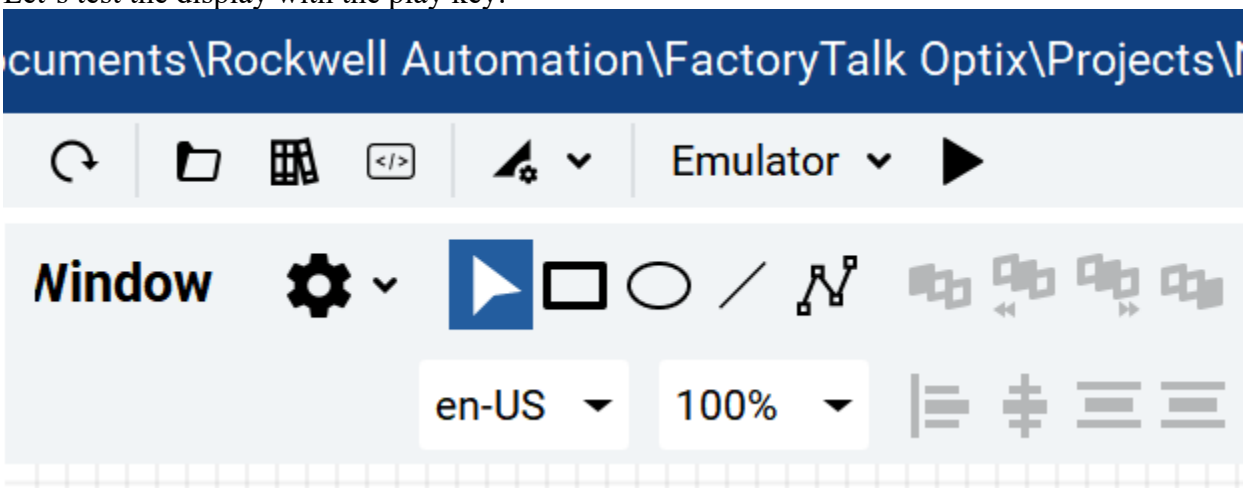
Go down in the properties until Text, and search for the Tag



Select, you will see this

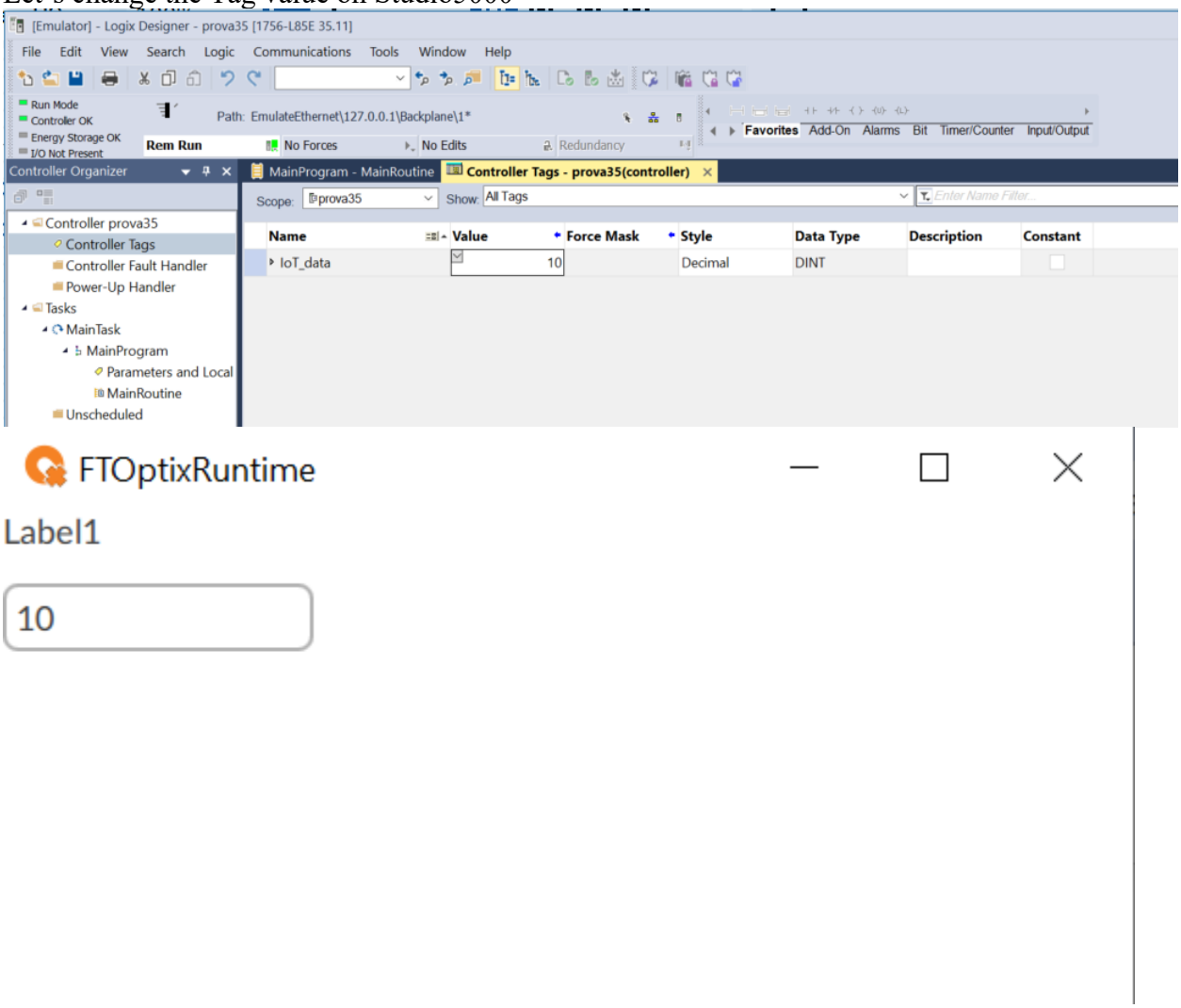


Let’s test the display with the play key:



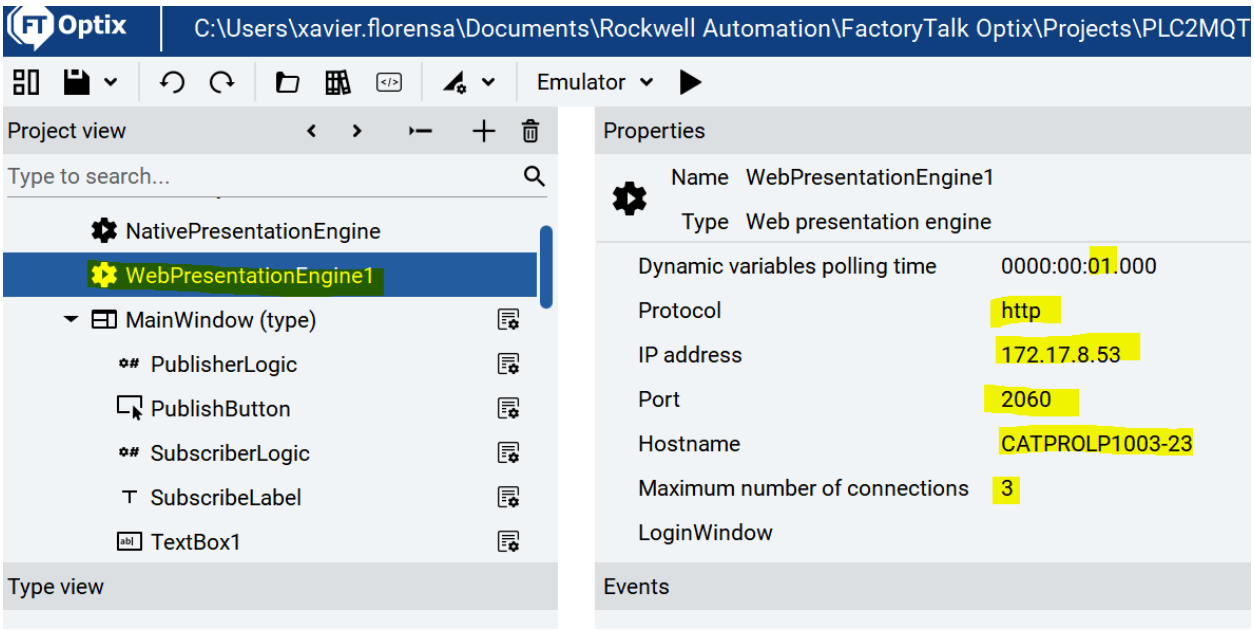
You will see this

Let’s change the Tag value on Studio5000

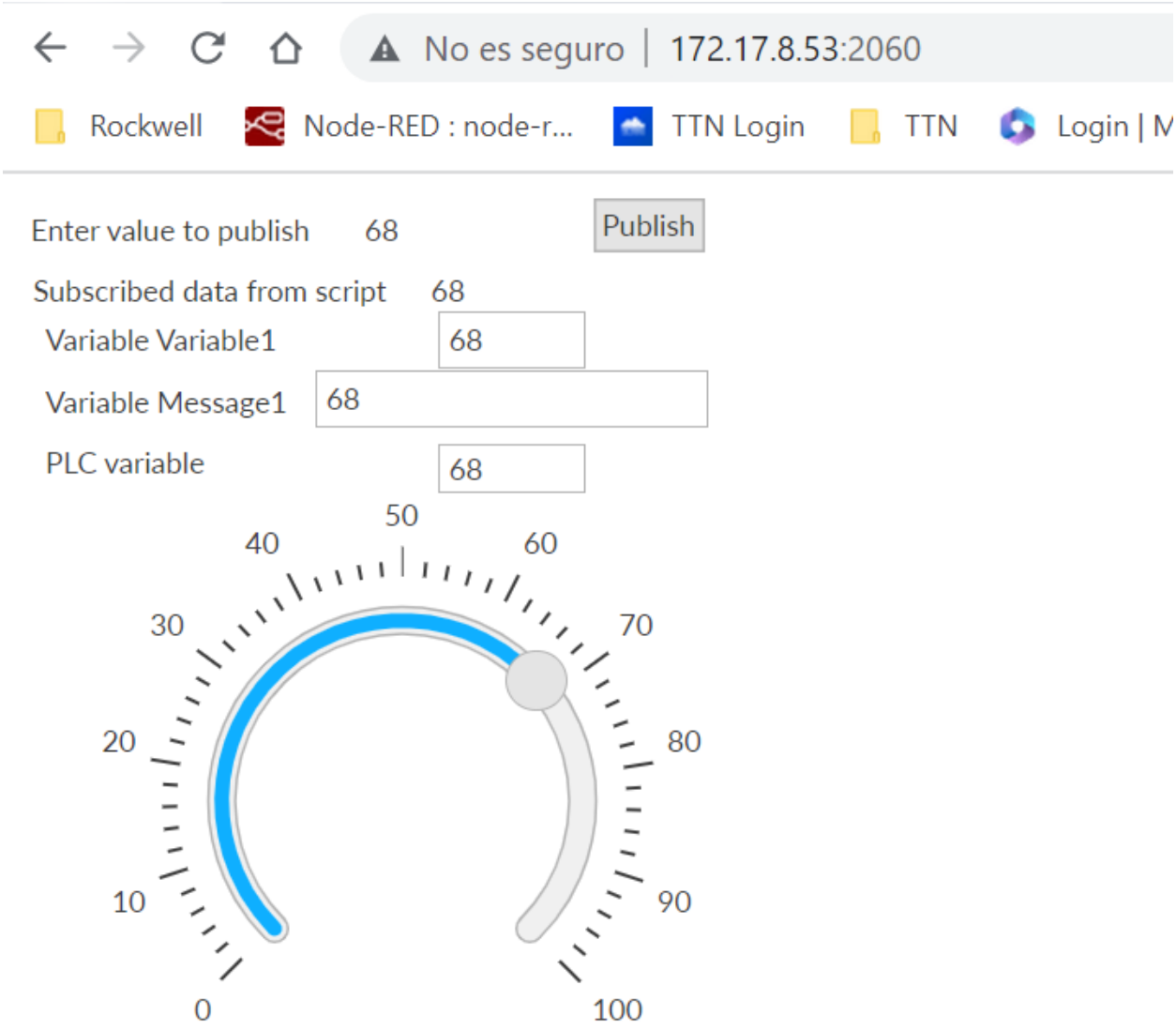


6. Web access

Do add a WebPresentationEngine (as new object), place before the MainWindow (type) and set it up with the IP address you have on your PC. Choose a port, for instance 2060, but any port may work.



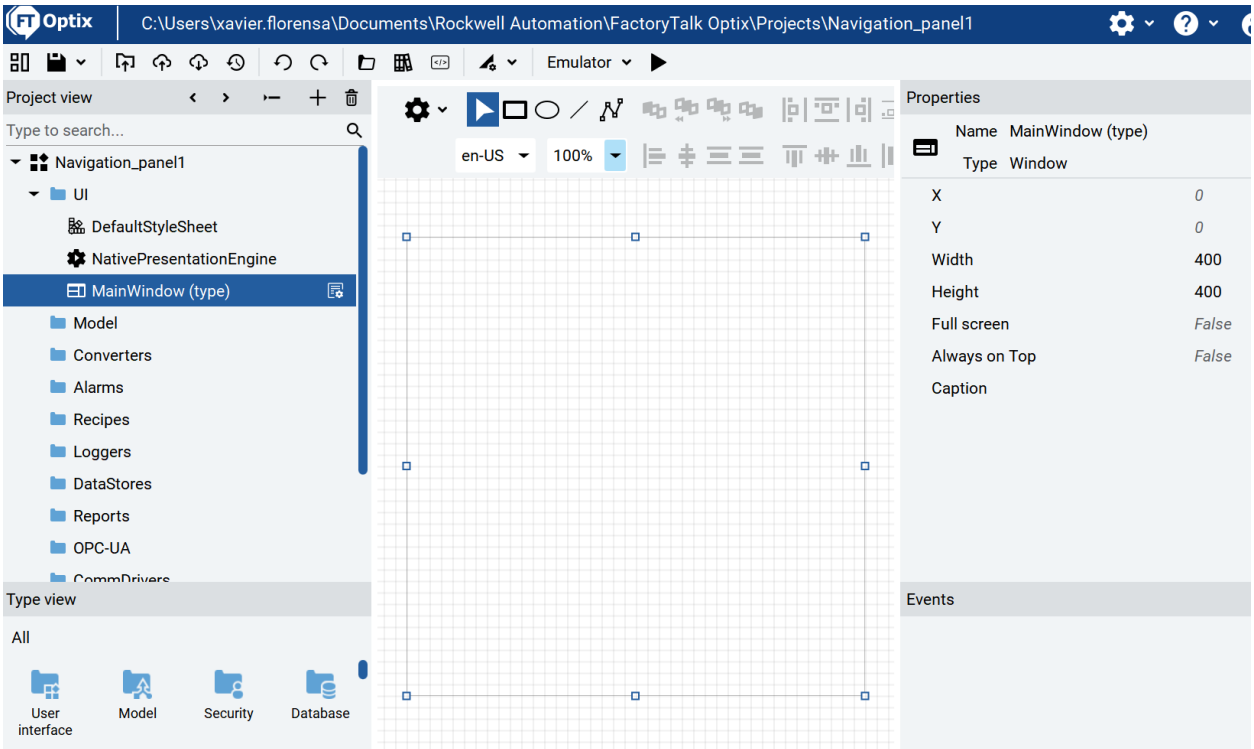
Open a explorer with the IP address:port
That's all



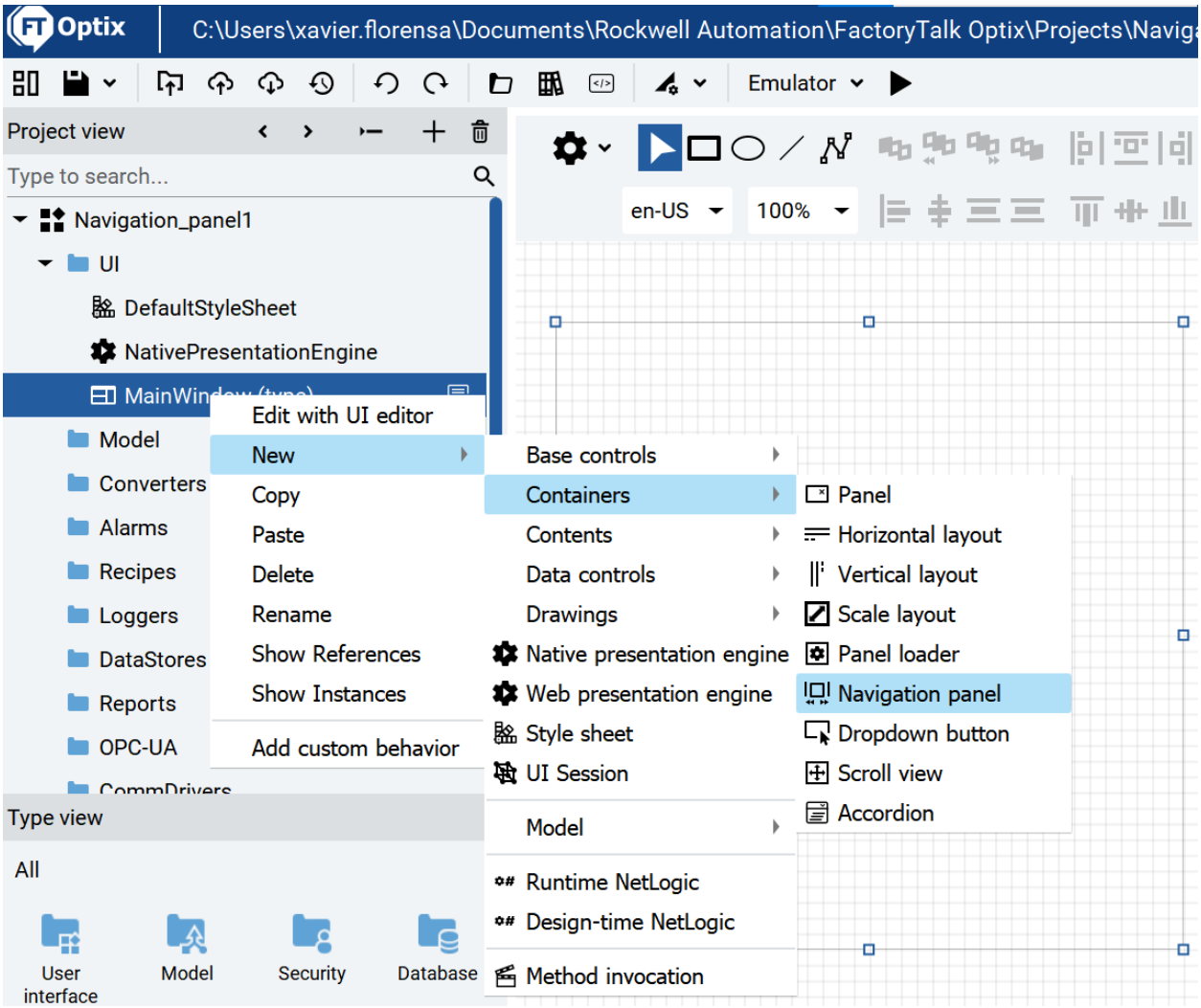
On Chrome works fine
By default you can only monitor, not control like the value of the slider.
On Apple Safari, there were some differences on the display.
You can see a demo on web access on this video
https://www.youtube.com/watch?v=78_cwp0iSJA

7. Navigation Panels

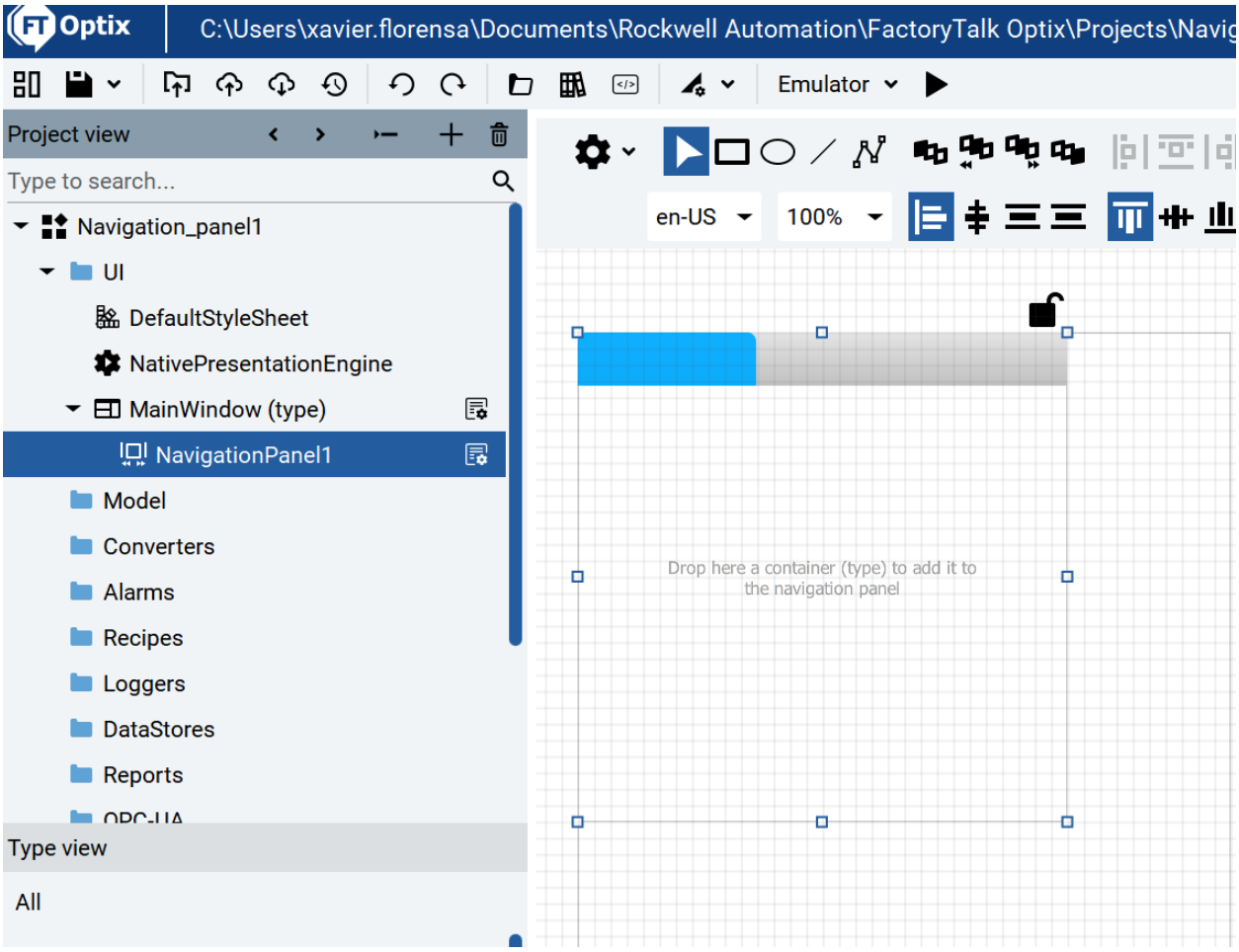
Start with this if you want to have different windows or panels visible on your project.
Create a blank new project
Go to Main window



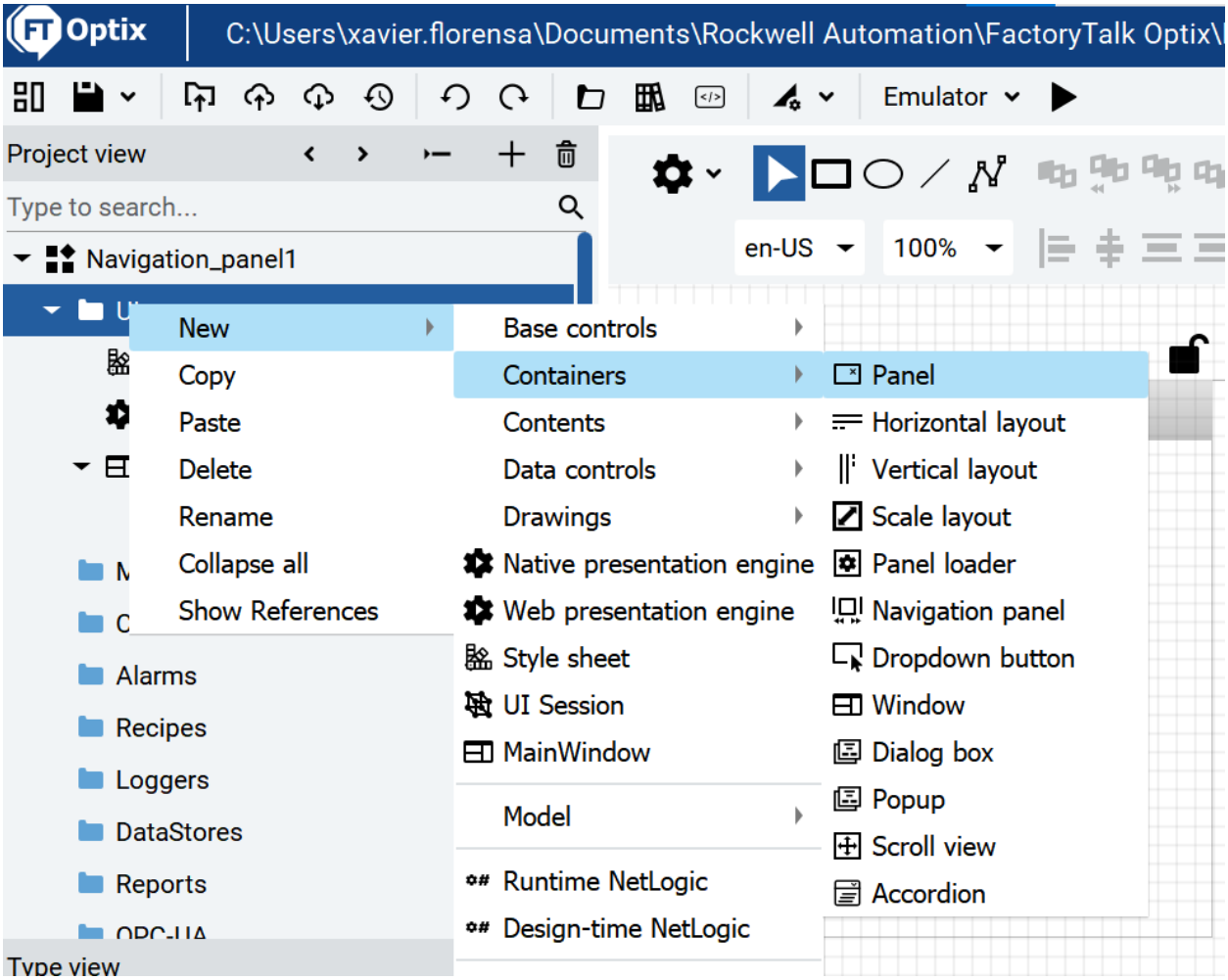
Create a Navigation Pannel with right button click



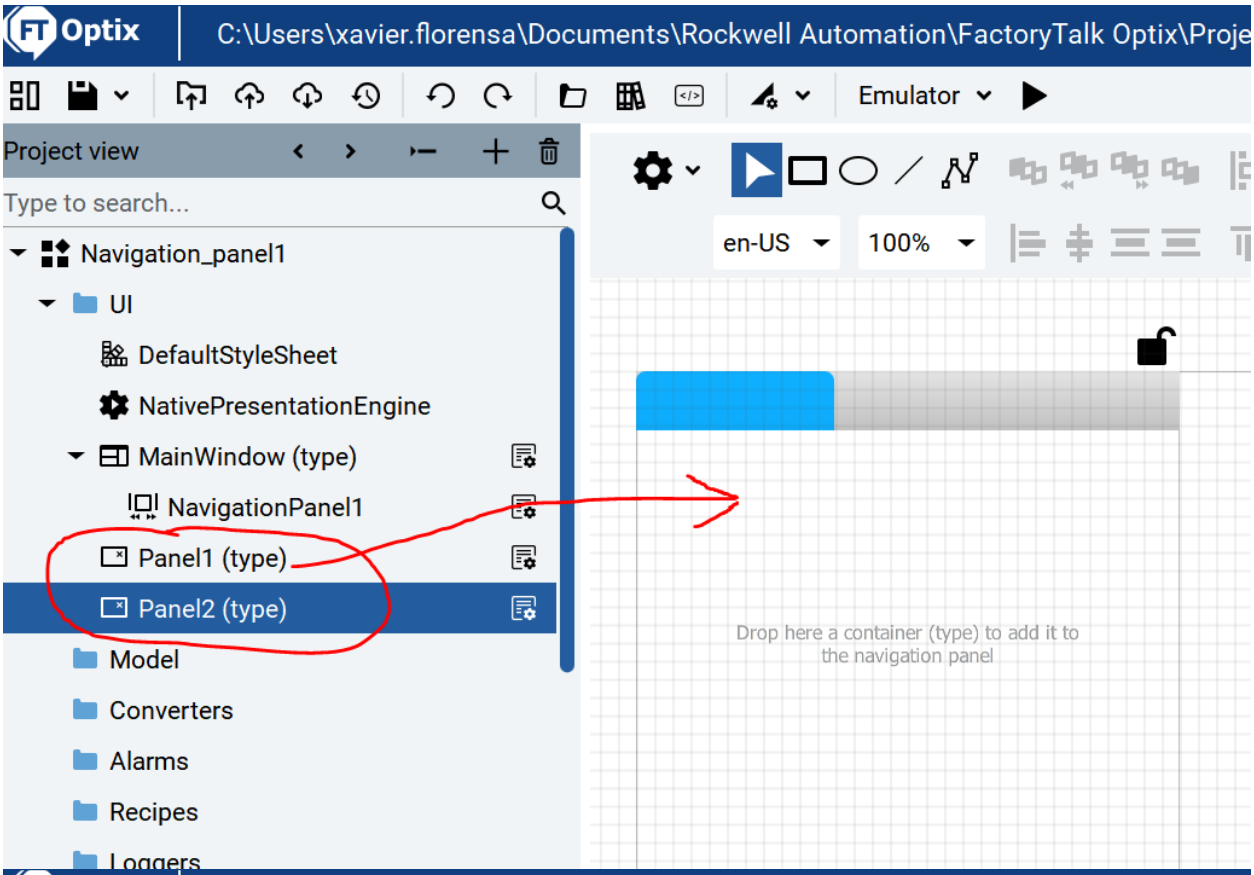
You will see this

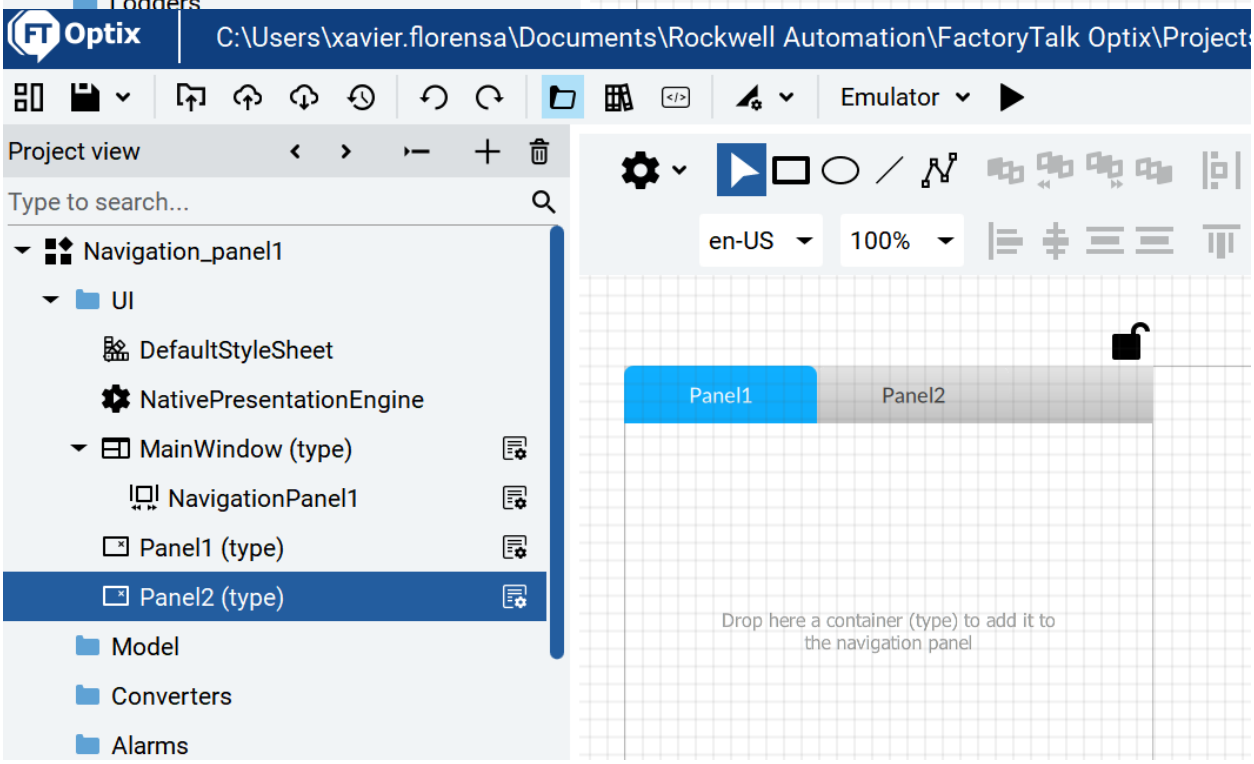


Then create two (or more) new panels with the right mouse button.

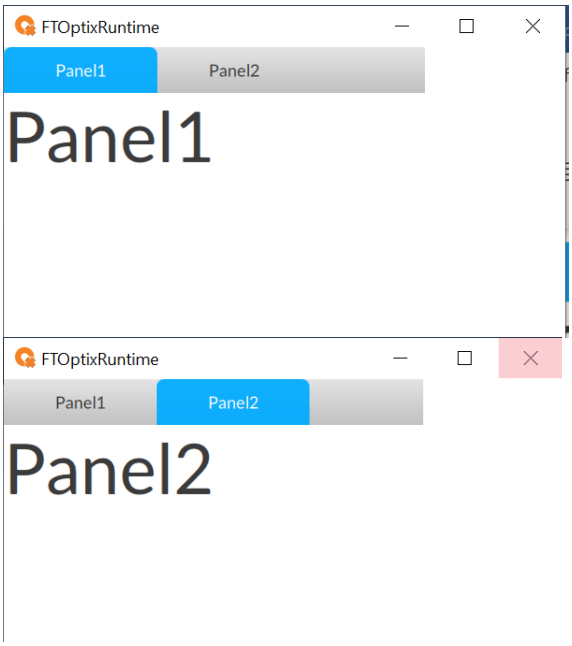


Then Drag and drop the two panels into the navigation panel





You can add labels to each panel to see on which panel you are



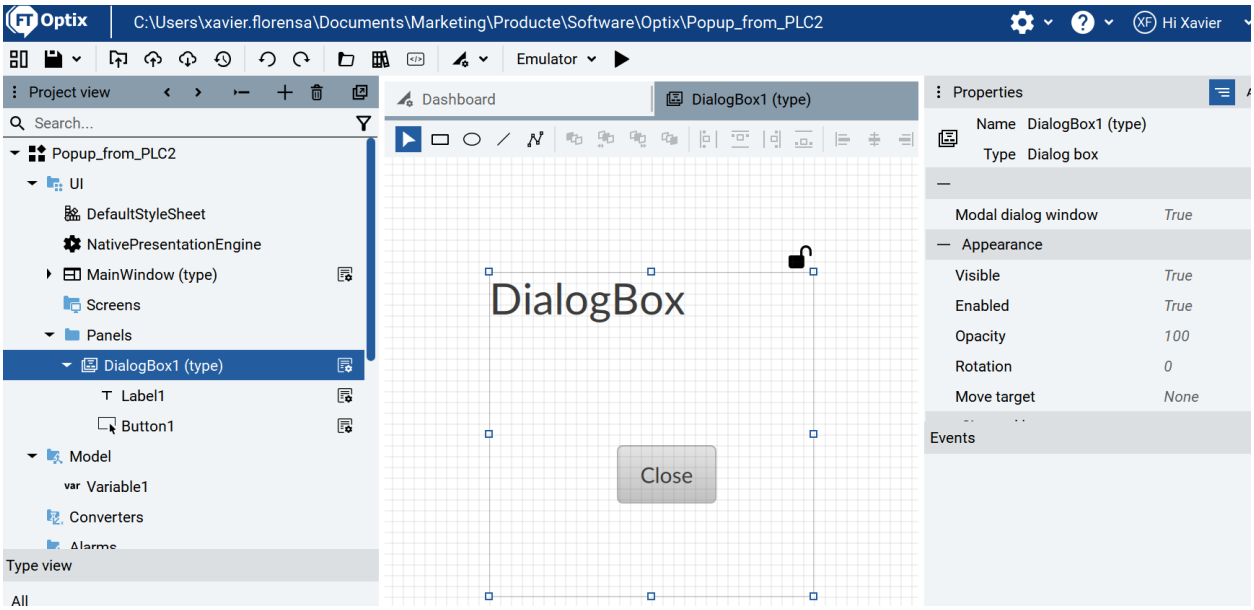
That's all
If you introduce a remote web navigation as on previous chapter, you will be able to have two operators working with different panels each.
You can see a demo on Navigation panels and web access on this video
https://www.youtube.com/watch?v=78_cwp0iSJA

8. PopUp or DialogBox Windows

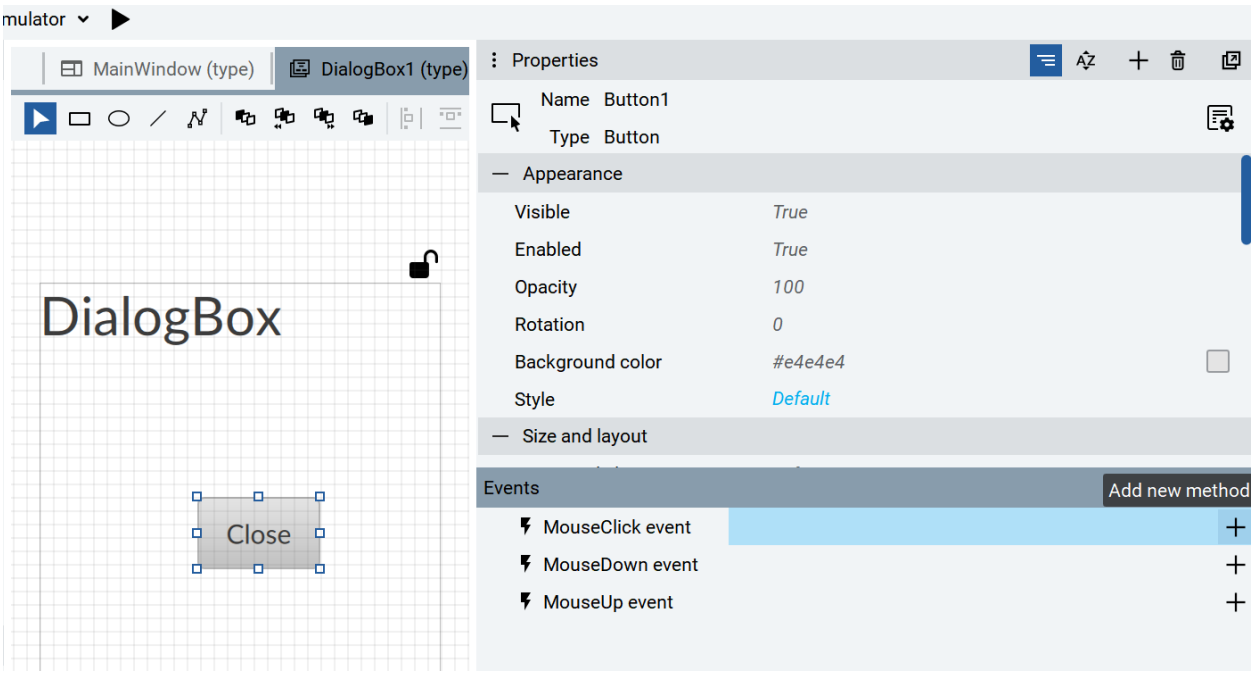
The PopUp Window is called DialogBox in Optix.
It's easy to create a buton to open a DialogBox.

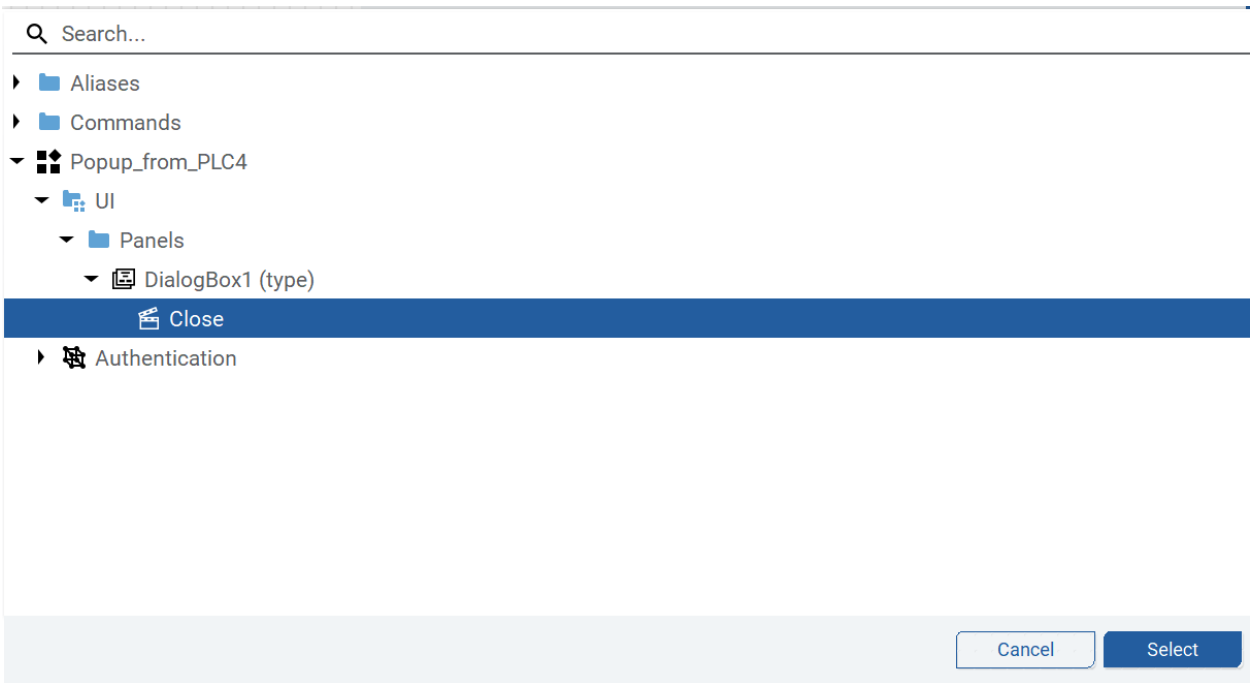
8.1. PopUp / DialogBox from Button

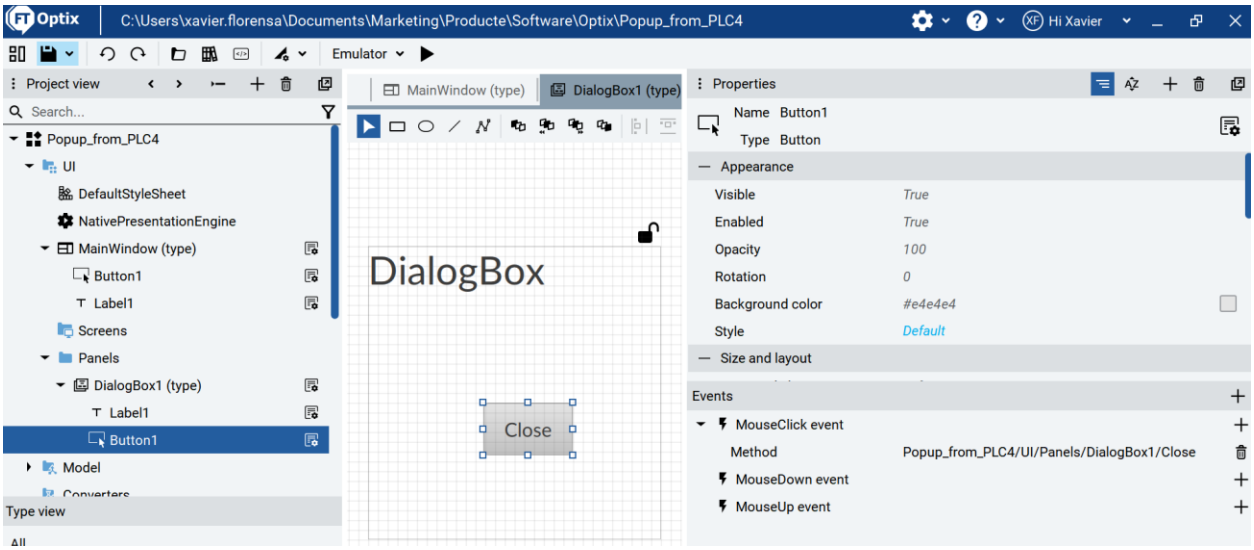
Let's create a new DialogBox



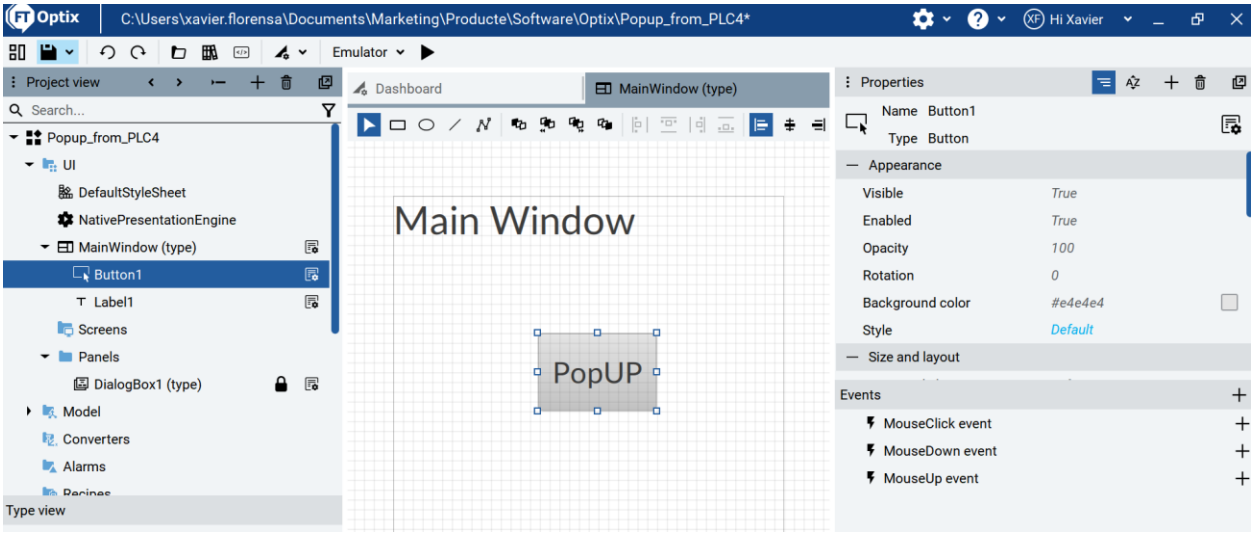
Add also a close Button



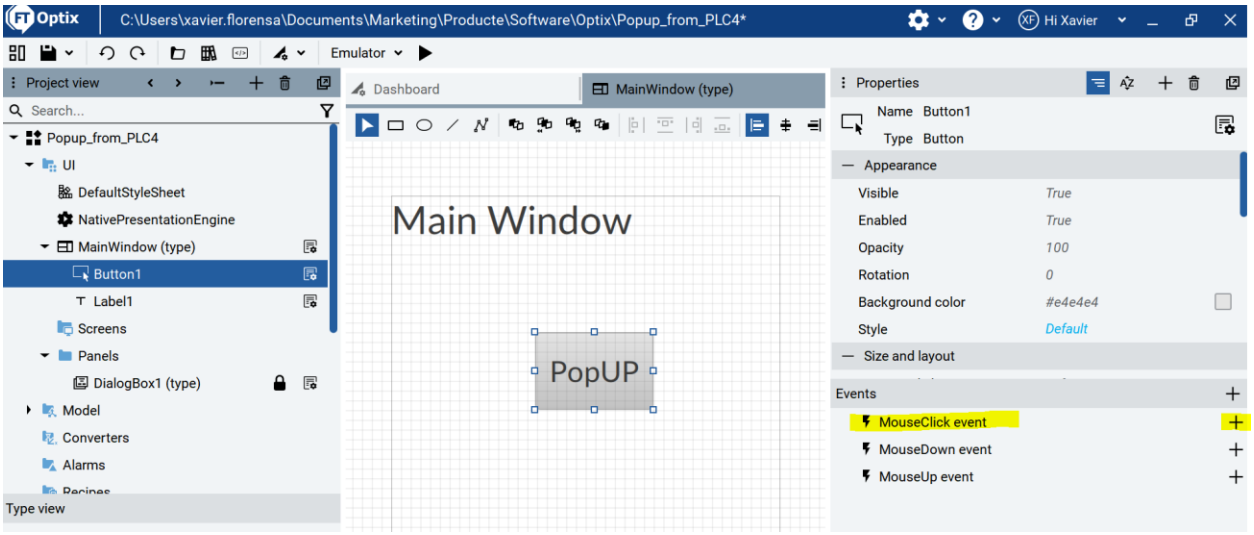




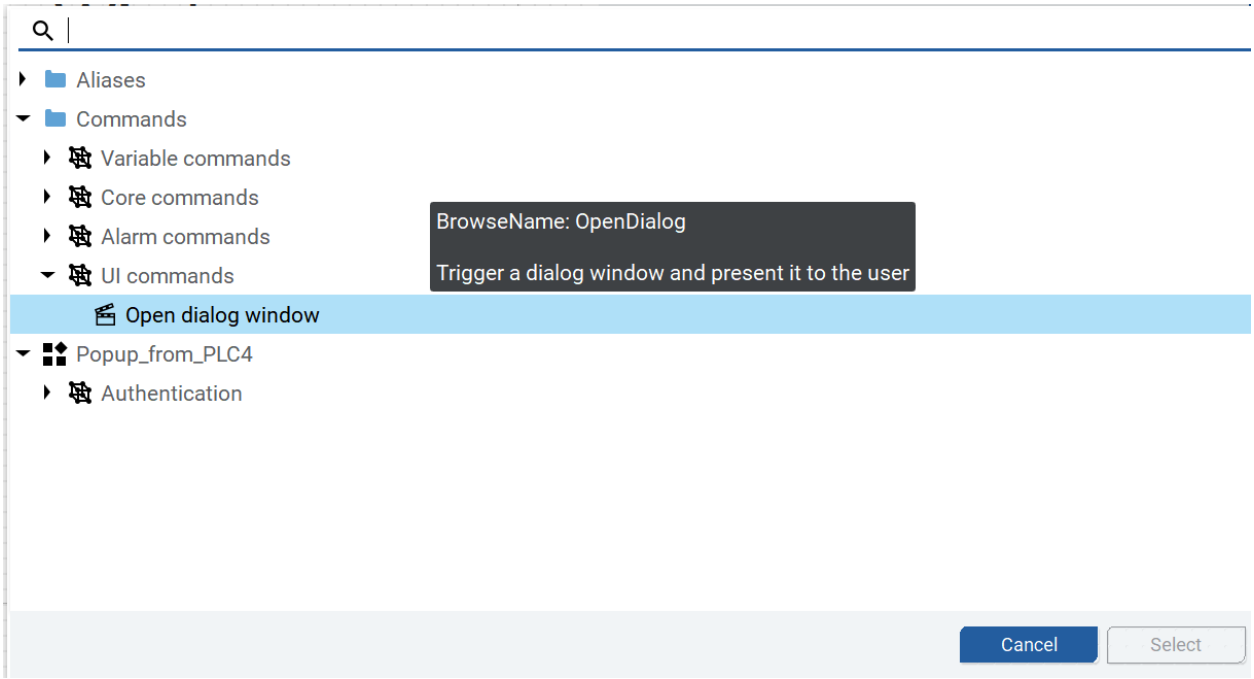
Now create a Button for instance on MainWindow



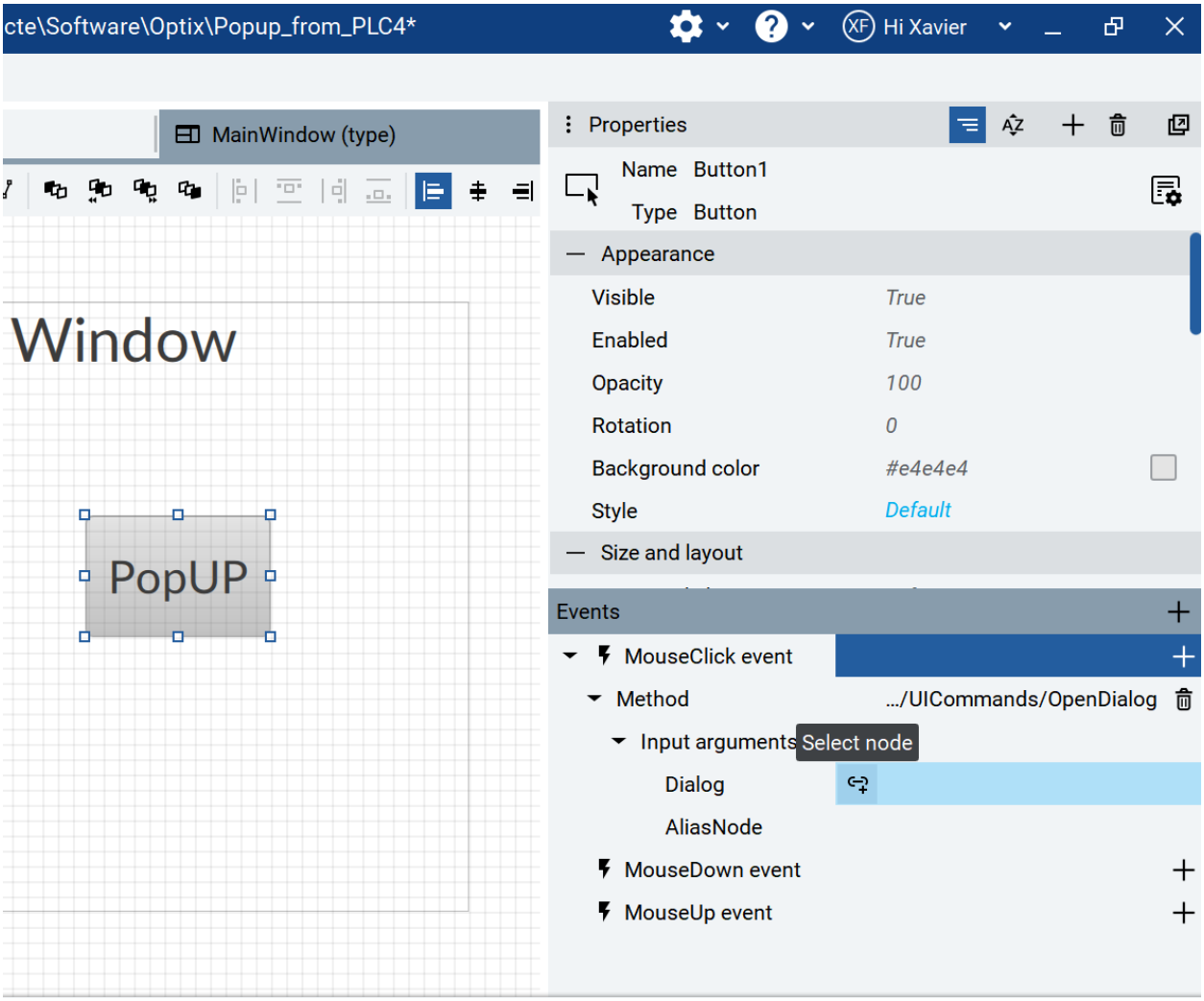
Create a new MouseClick event with +



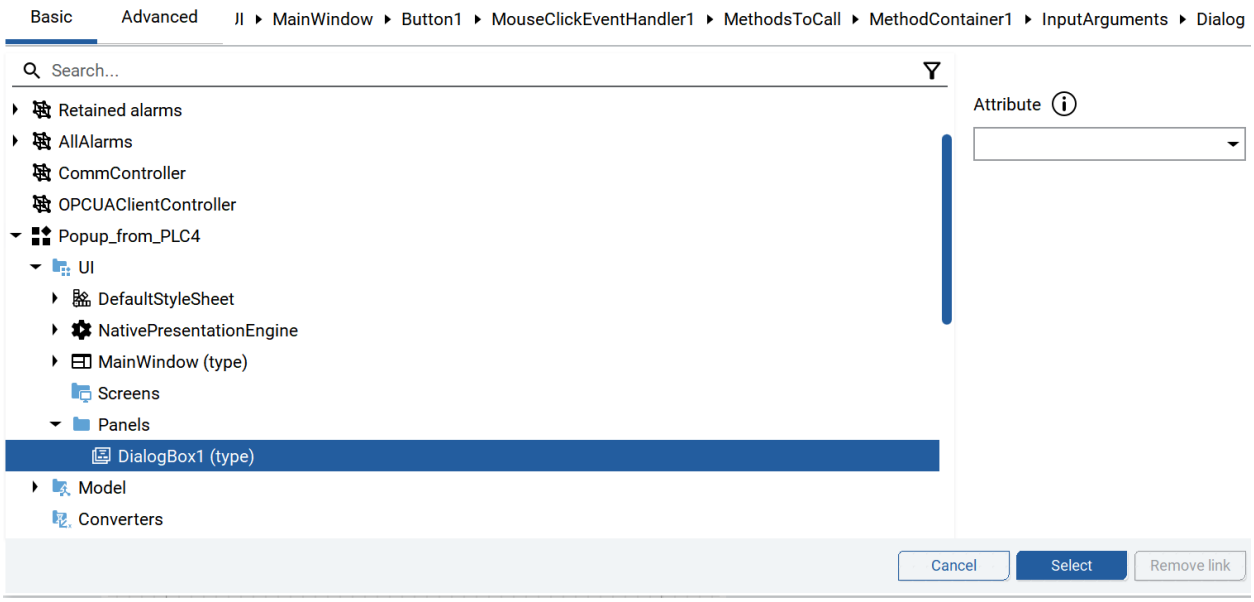
Navigate to

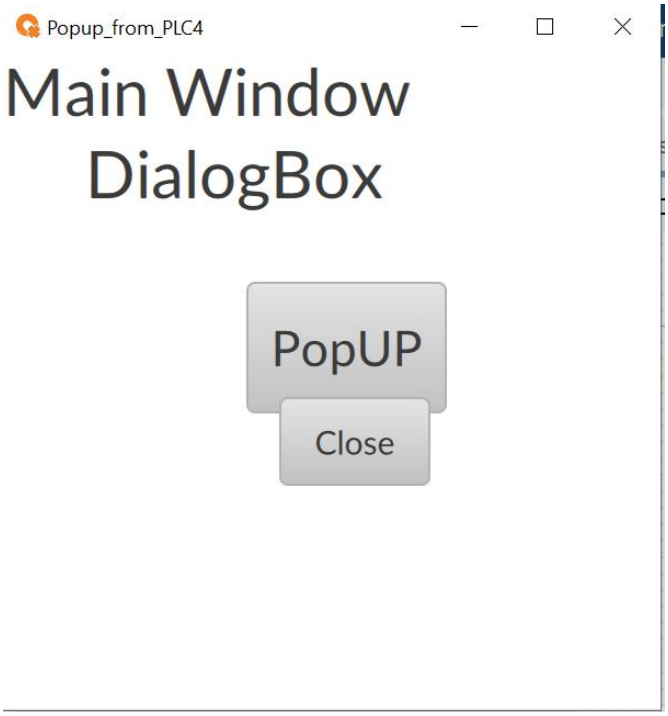


And select the DialogBox


cte\Software\Optix\Popup_from_PLC4*
Hi Xavier
MainWindow (type)
Window
PopUP
Properties
Name Button1
Type Button
Appearance
Visible True
Enabled True
Opacity 100
Rotation 0
Background color #e4e4e4
Style Default
Size and layout
Events
MouseClick event
Method .../UICommands/OpenDialog
Input arguments
Select node
Dialog
AliasNode
MouseDown event
MouseUp event

Select


Basic
Advanced
UI ▸ MainWindow ▸ Button1 ▸ MouseClickEventHandler1 ▸ MethodsToCall ▸ MethodContainer1 ▸ InputArguments ▸ Dialog
Search...
Retained alarms
AllAlarms
CommController
OPCUAClientController
Popup_from_PLC4
UI
DefaultStyleSheet
NativePresentationEngine
MainWindow (type)
Screens
Panels
DialogBox1 (type)
Model
Converters
Attribute
Cancel
Select
Remove link



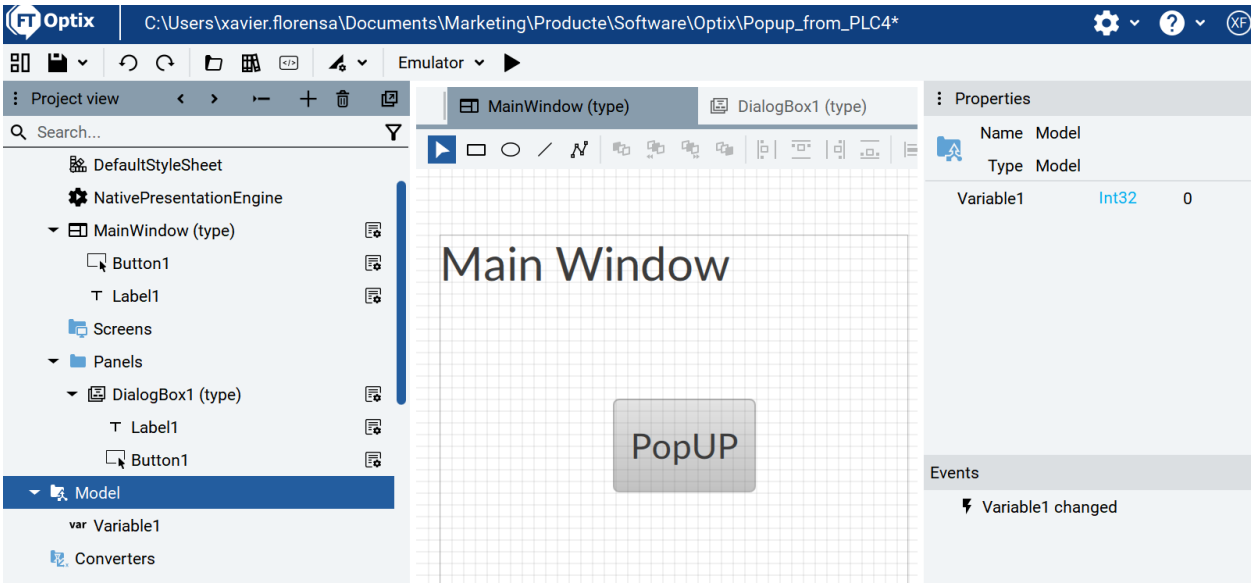
You can add a colored rectangle to overlap and do not see what's behind the Dialog Box

8.2. PopUp / DialogBox from variable

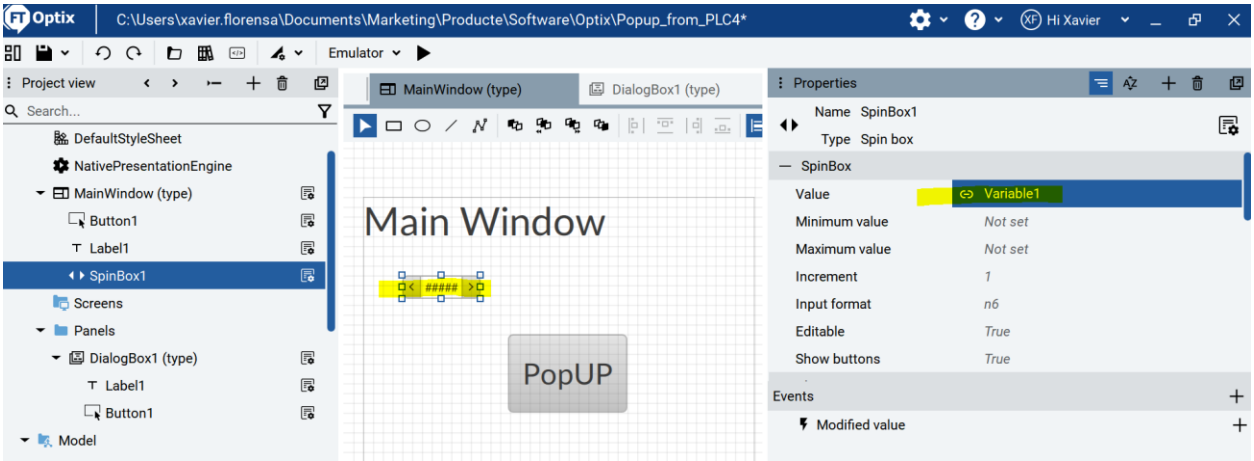
You can get the code from here

https://github.com/xavierflorensa/Popup_from_PLC2

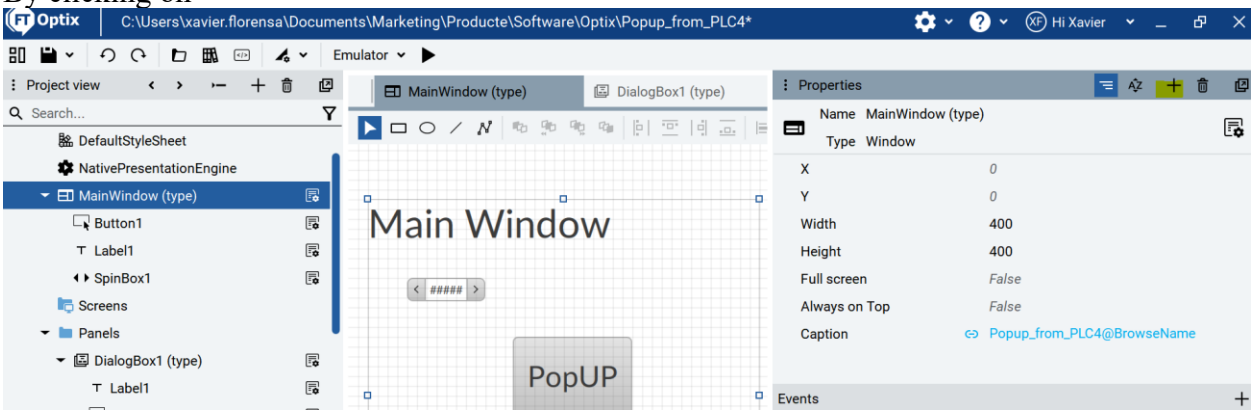
Create a new variable on Model, this will be used to trigger the DialogBox if changes

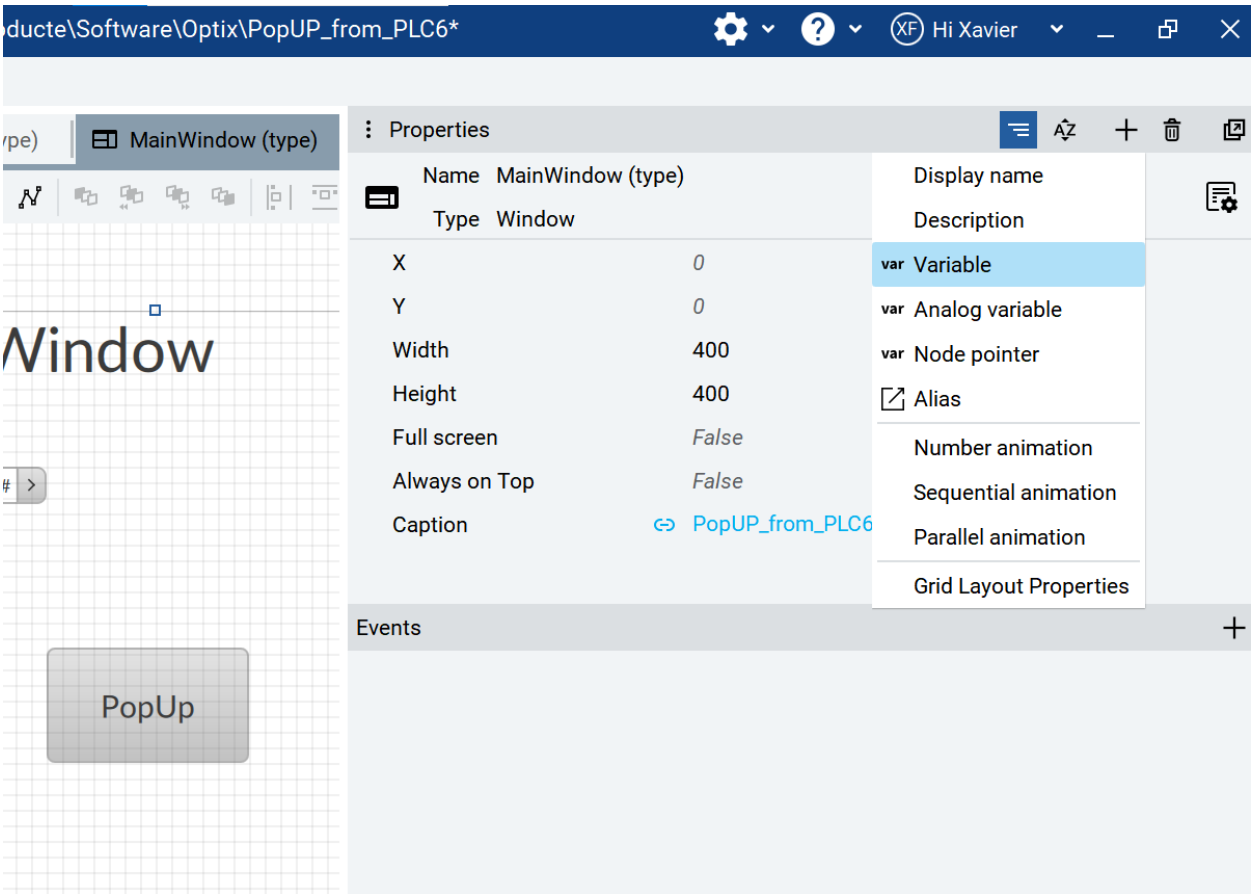


Let’s create a SpinBox to be vinculated with this variable

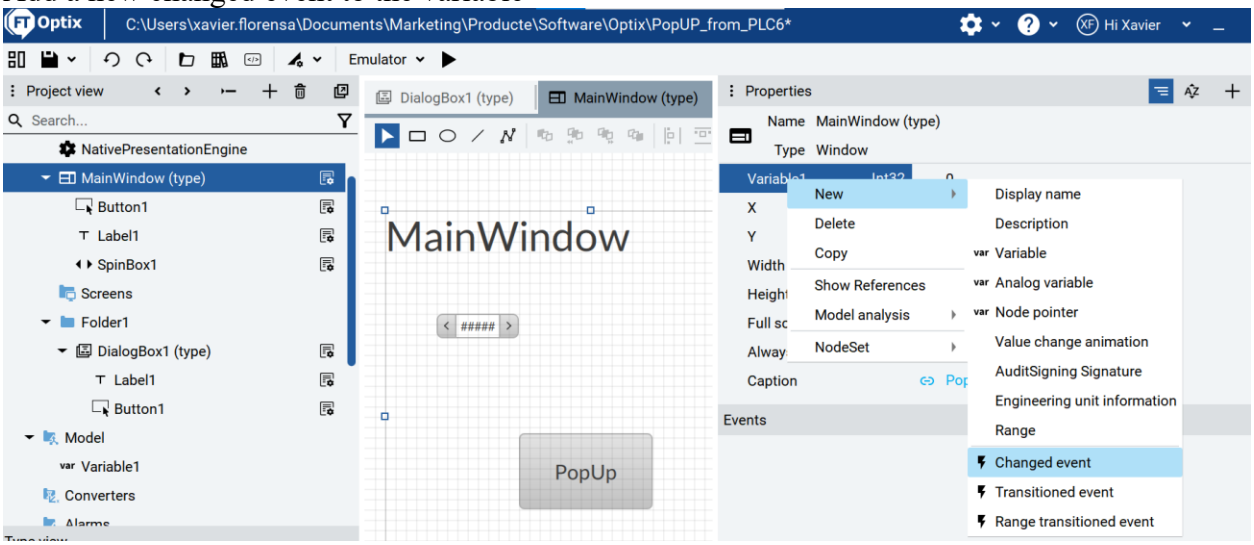


Now let’s create a new variable on the Main Window (Not on Model)
By clicking on +

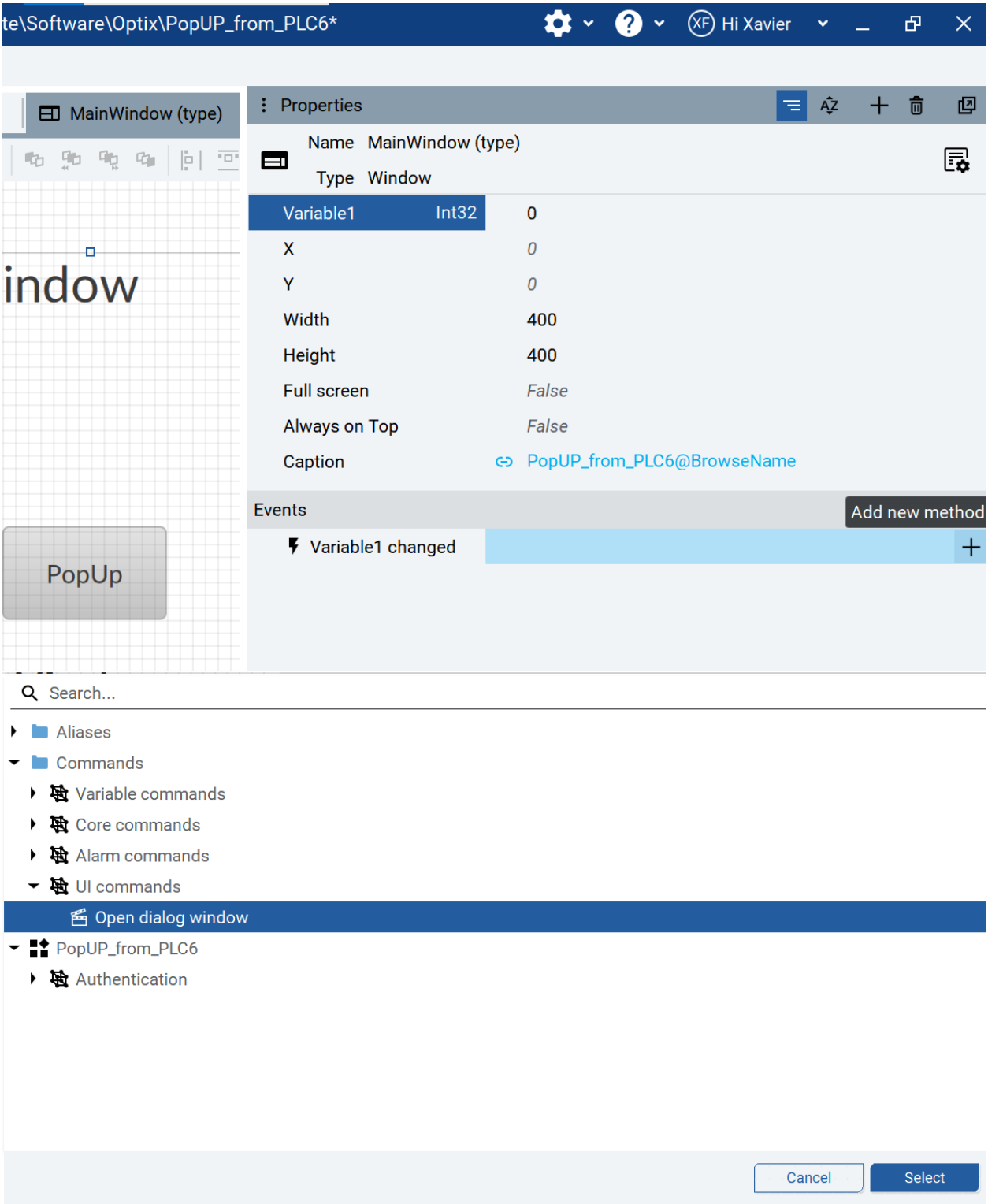




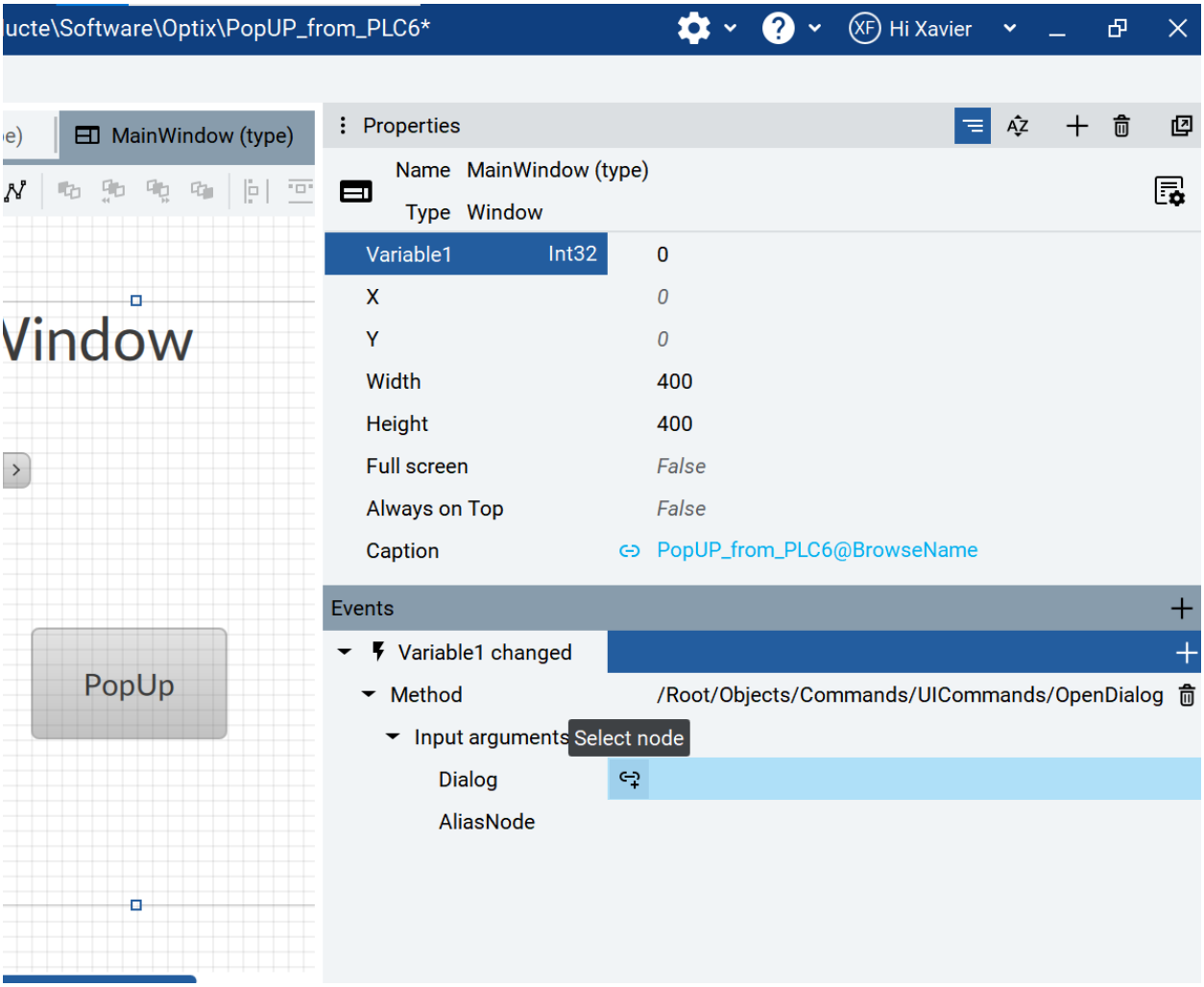
Add a new changed event to the variable

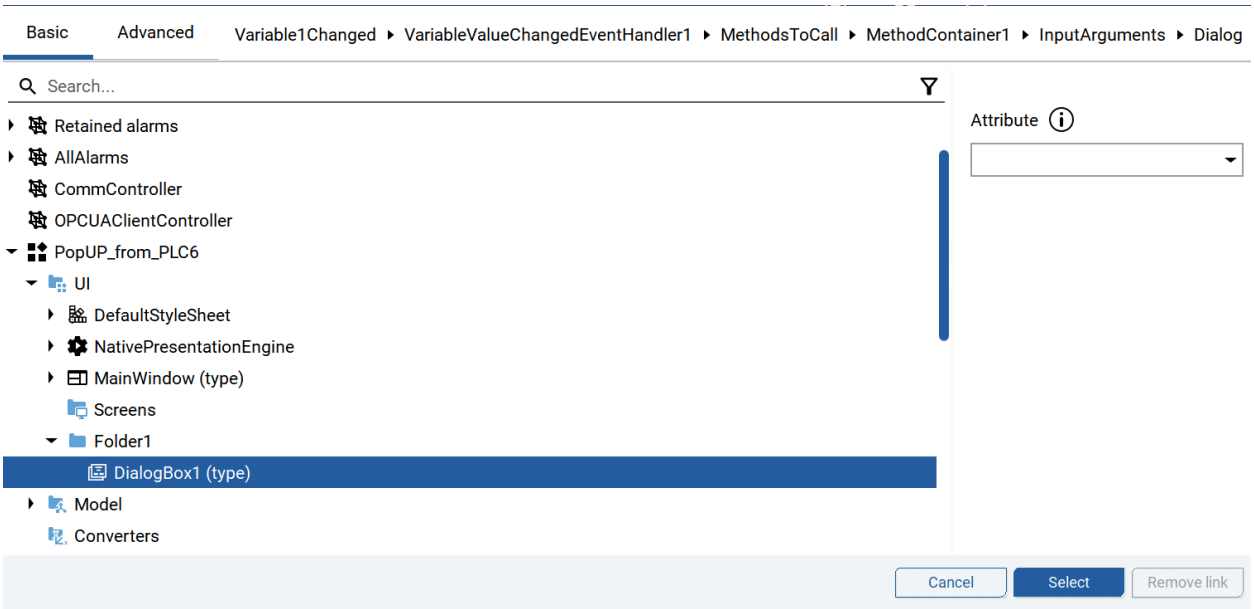


Select the method

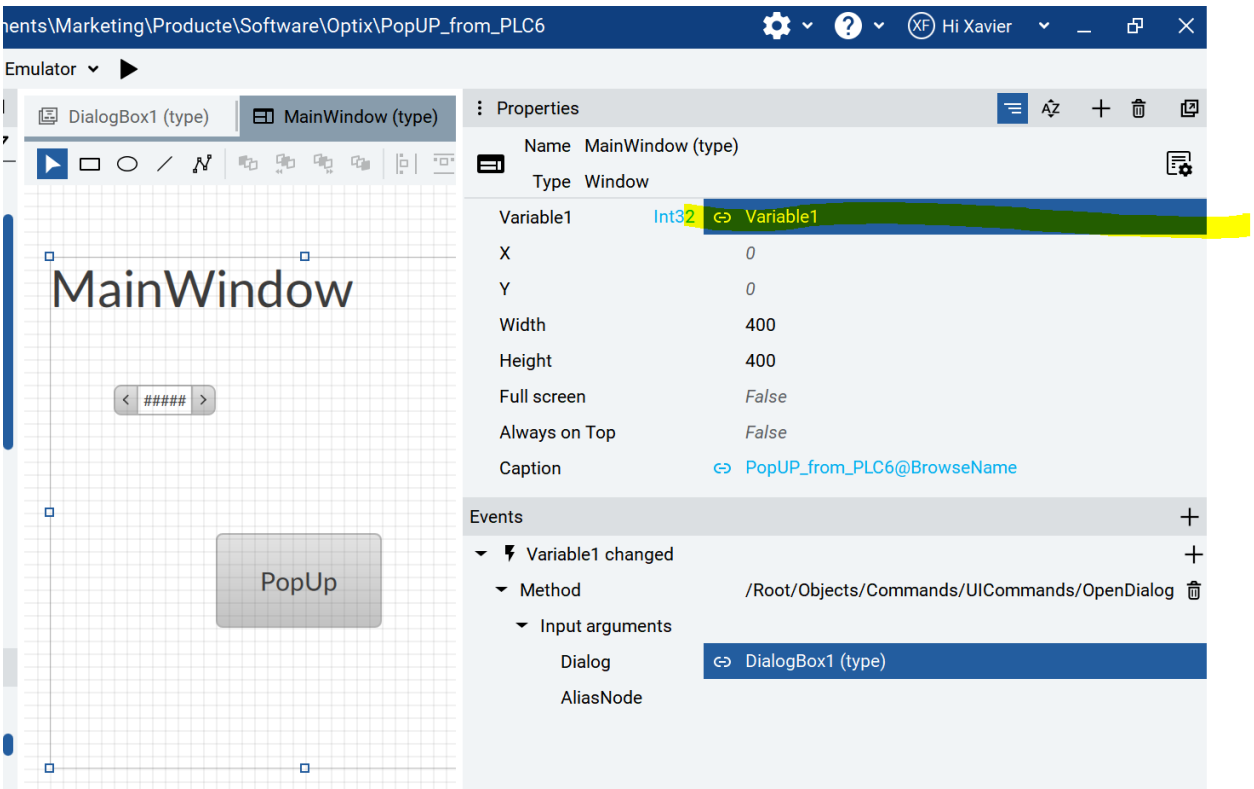


and the Dialog to open


ucte\Software\Optix\PopUP_from_PLC6*
Hi Xavier
MainWindow (type)
Vindow
PopUp
Properties
Name MainWindow (type)
Type Window
Variable1 Int32 0
X 0
Y 0
Width 400
Height 400
Full screen False
Always on Top False
Caption PopUP_from_PLC6@BrowseName
Events
Variable1 changed
Method /Root/Objects/Commands/UICommands/OpenDialog
Input arguments
Select node
Dialog
AliasNode


Basic
Advanced
Variable1Changed ▸ VariableValueChangedEventHandler1 ▸ MethodsToCall ▸ MethodContainer1 ▸ InputArguments ▸ Dialog
Search...
Retained alarms
AllAlarms
CommController
OPCUAClientController
PopUP_from_PLC6
UI
DefaultStyleSheet
NativePresentationEngine
MainWindow (type)
Screens
Folder1
DialogBox1 (type)
Model
Converters
Attribute
Cancel
Select
Remove link

Finally vinculate Variable1 from MainWindow to the variable from Model



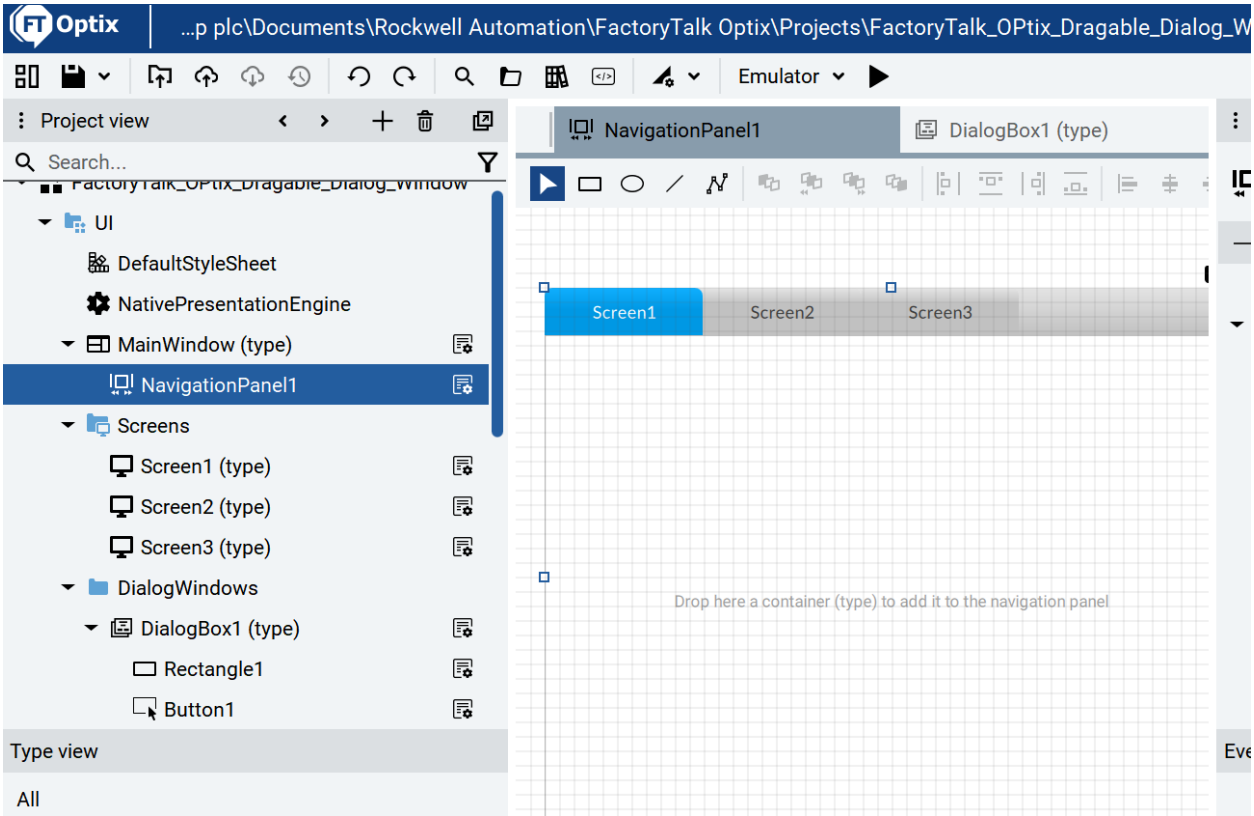
That's all

8.3. PopUp / DialogBox from PLC value

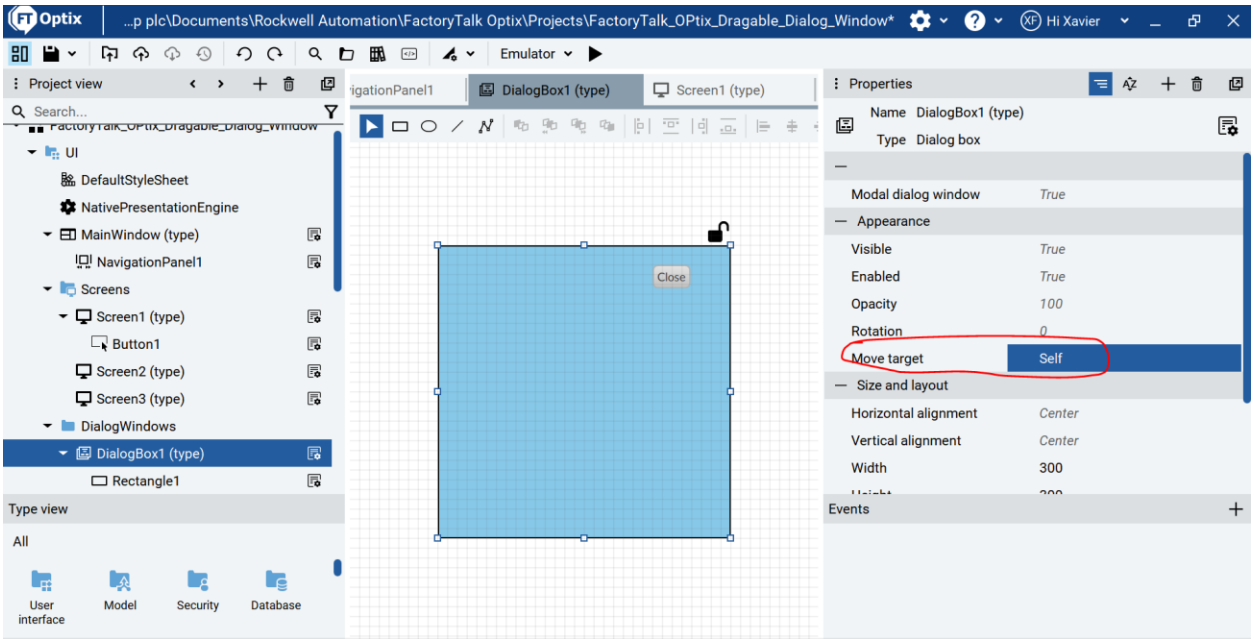
You have an example here
Only one DialogBox
https://github.com/xavierflorensa/FTOptix_Popup_from_PLC_CSHARP
Multiple Dialogboxes
https://github.com/FactoryTalk-Optix/DialogByPlcValue.git

8.4. Dragable DialogBox

Create a Navigation Panel with some Screens

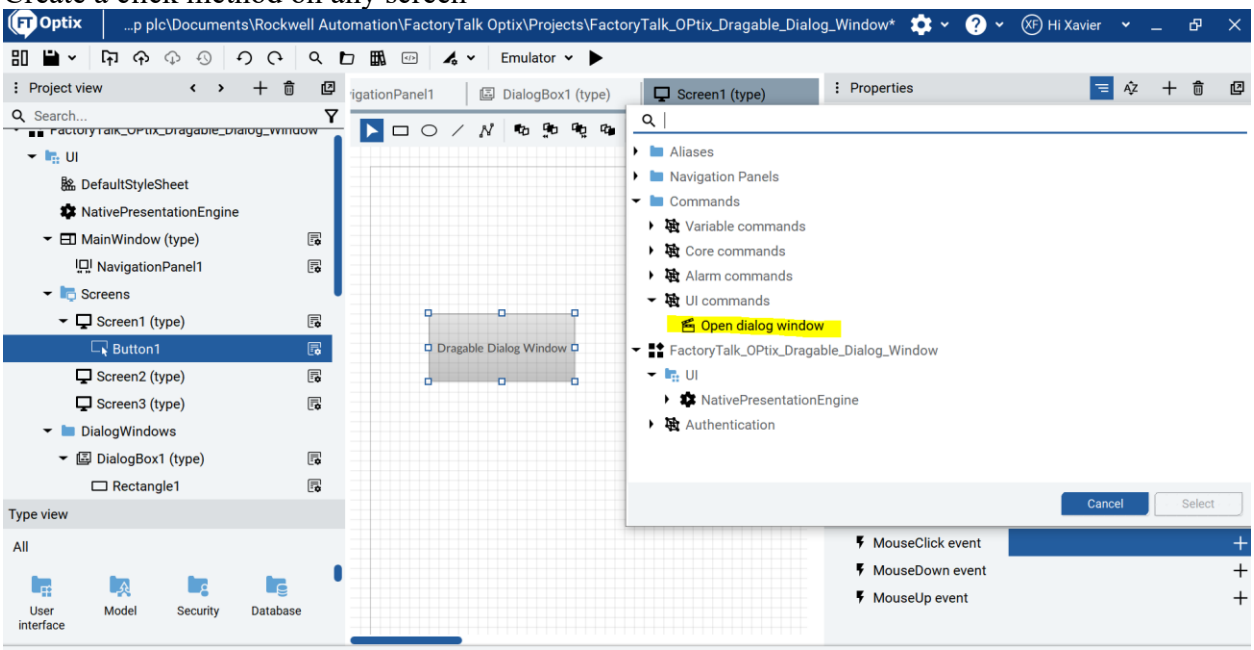


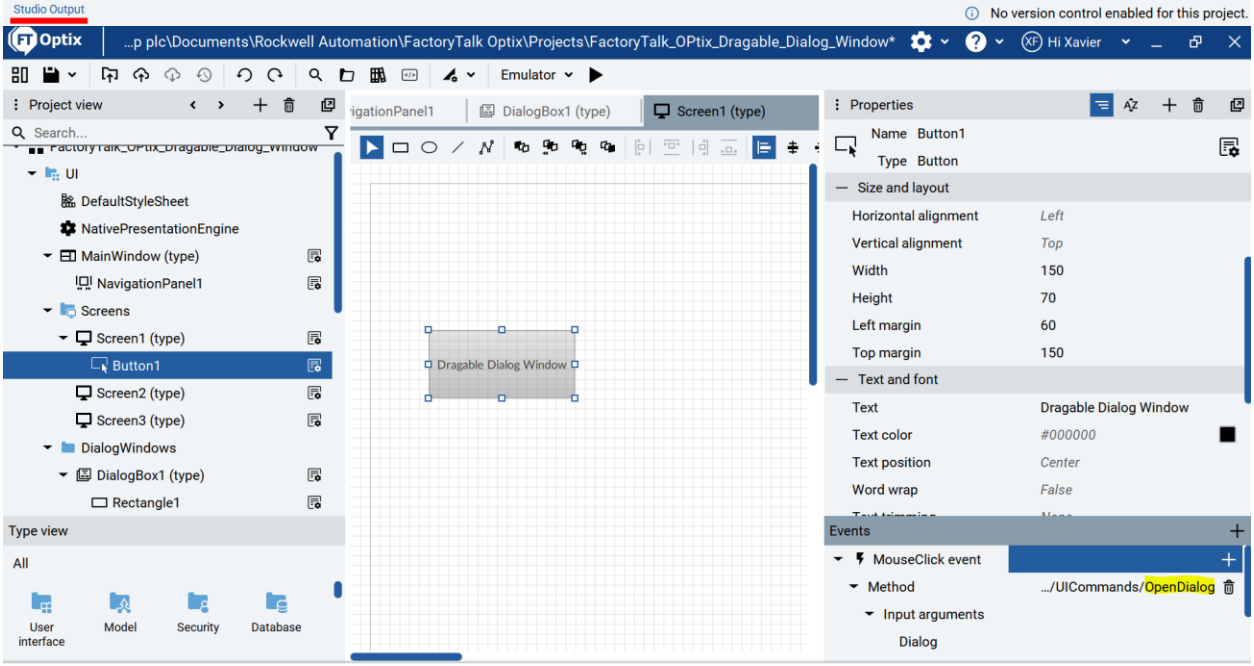
Create a Folder ans a DialogBox with a colored rectangle to indentify and a close button
On Move Target, set to “Self”.



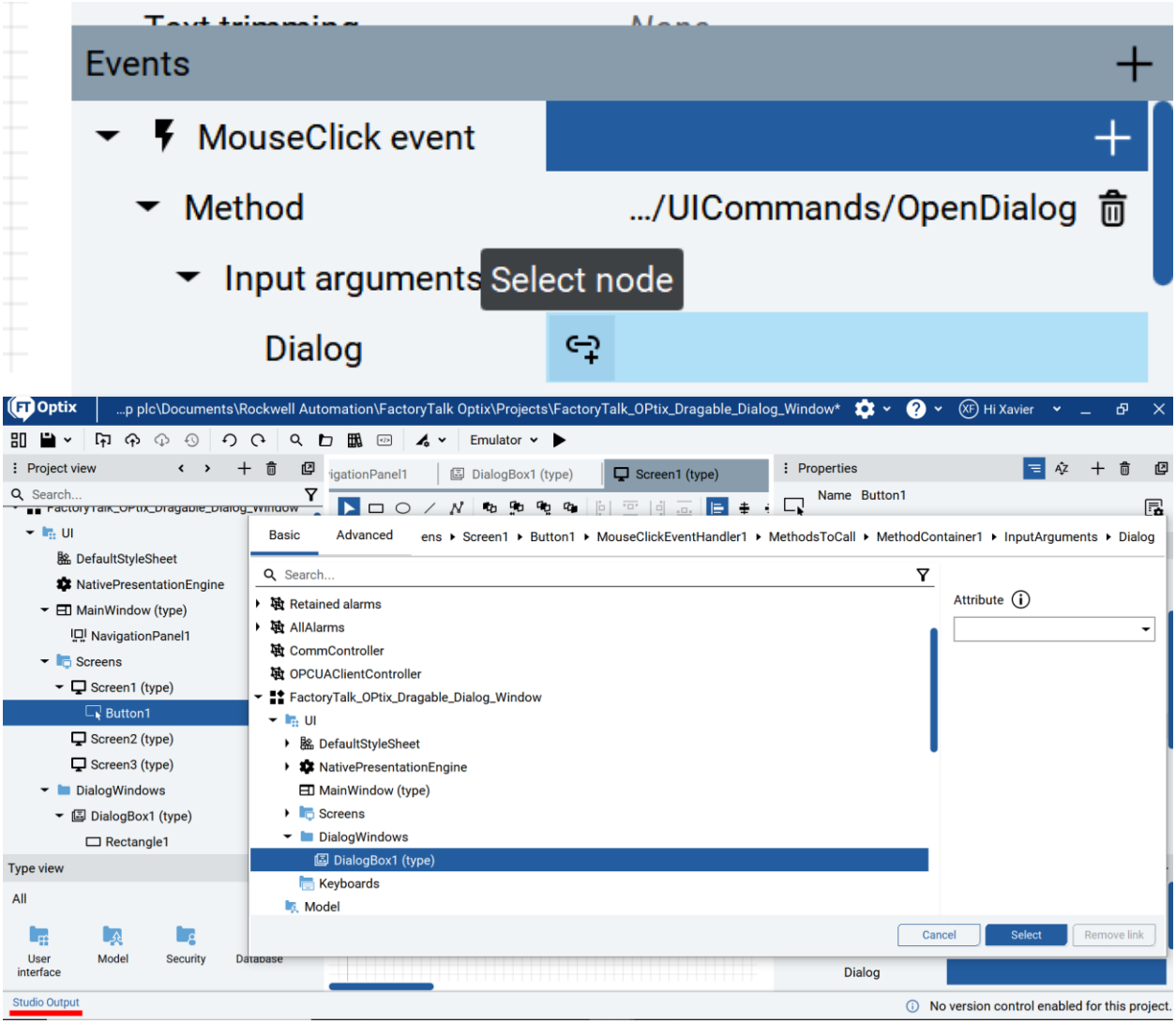
Create a Button on one of the screens

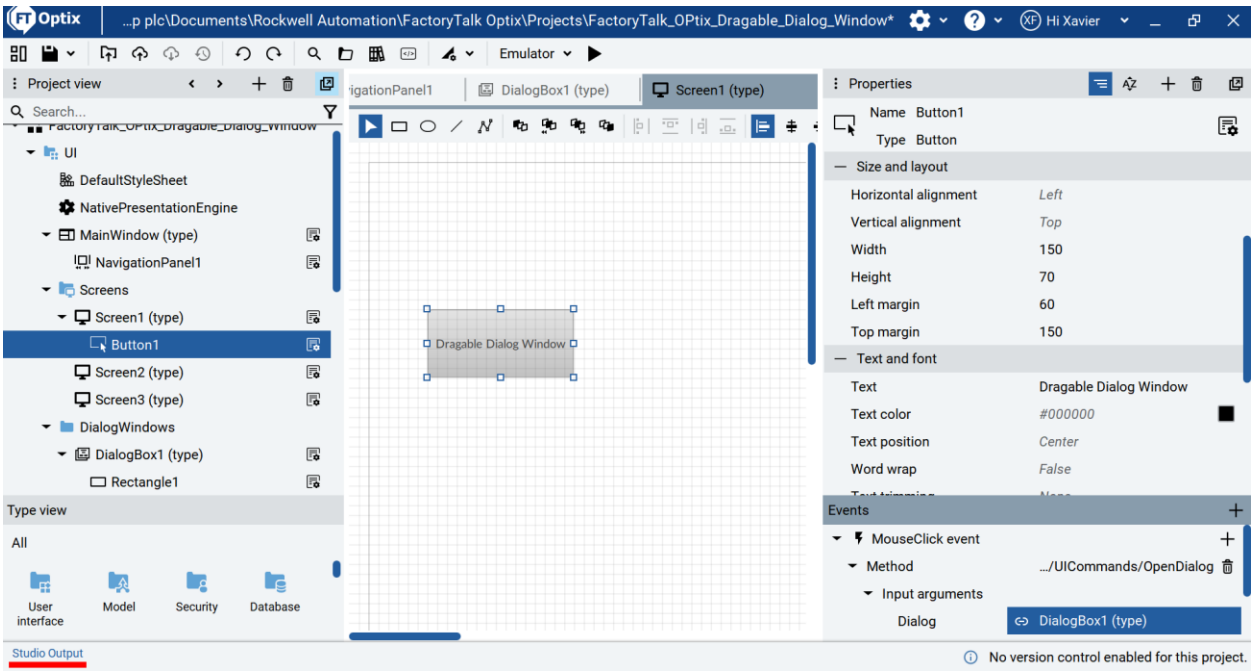
Create a click method on any screen



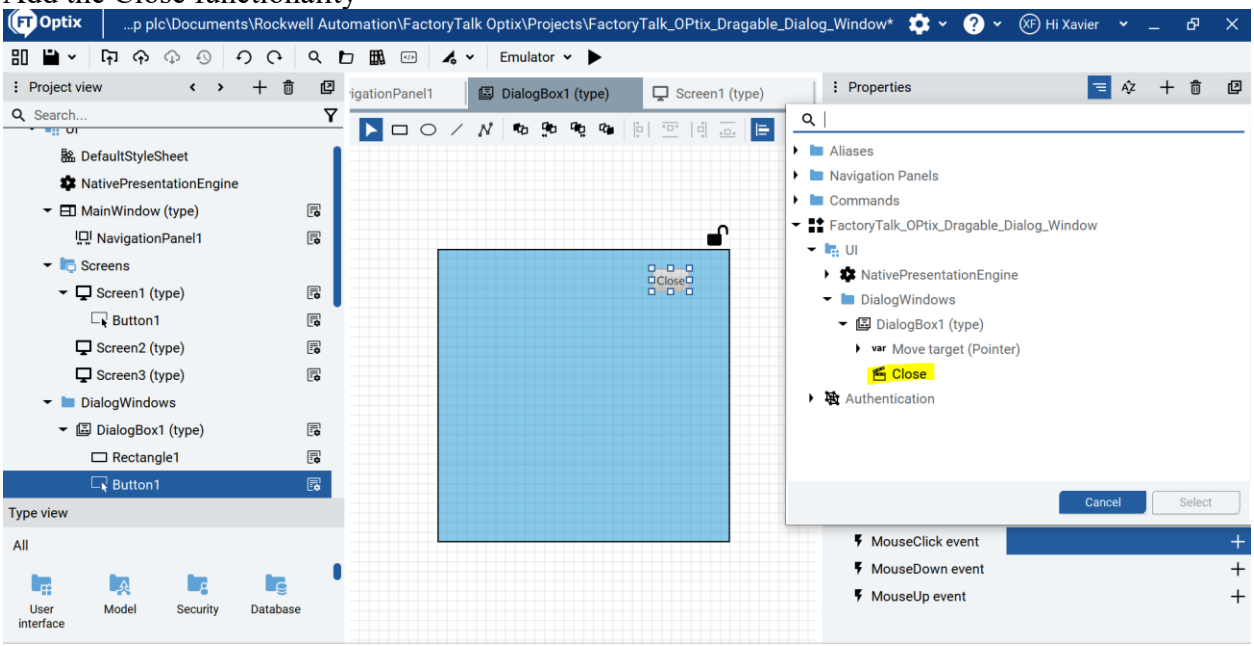


Select which DialogBox

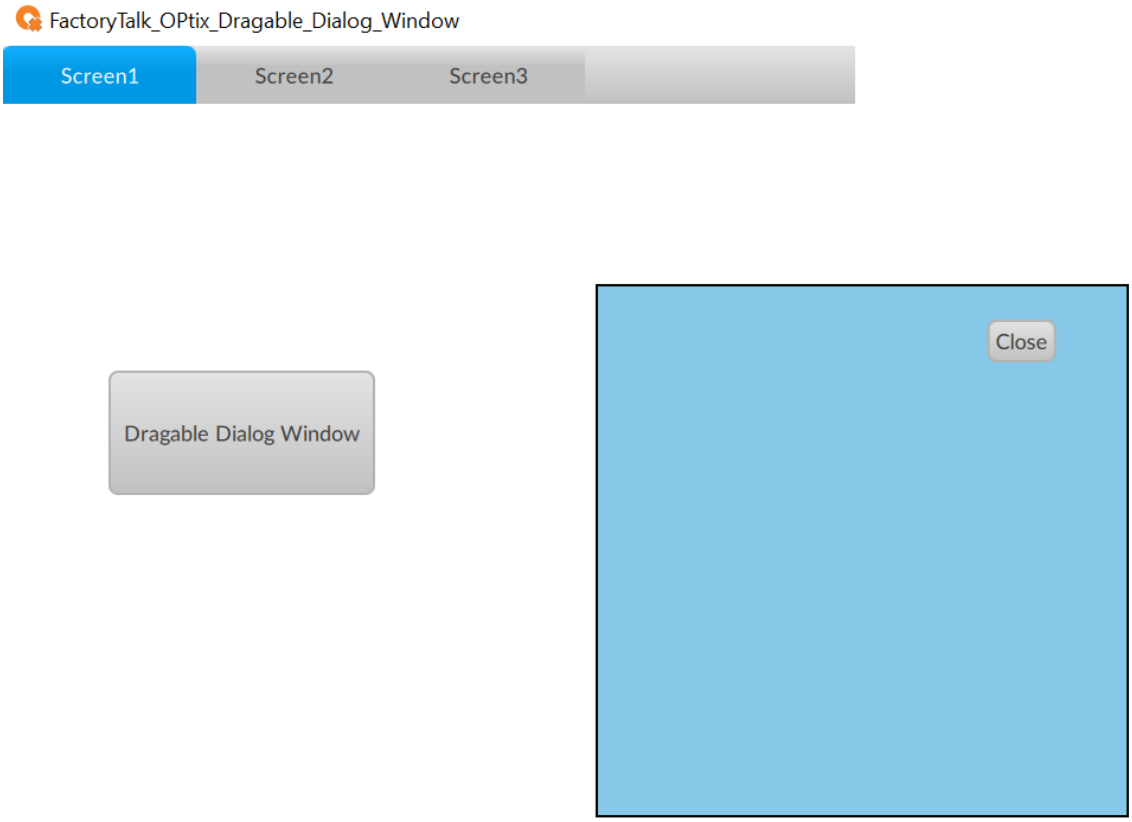

Events
MouseClick event
Method
.../UICommands/OpenDialog
Input arguments
Select node
Dialog
Optix
...p plc\Documents\Rockwell Automation\FactoryTalk Optix\Projects\FactoryTalk_OPtix_Dragable_Dialog_Window*
Hi Xavier
Emulator
Project view
Search...
DialogBox1 (type)
Screen1 (type)
Properties
Name Button1
UI
DefaultStyleSheet
NativePresentationEngine
MainWindow (type)
NavigationPanel1
Screens
Screen1 (type)
Button1
Screen2 (type)
Screen3 (type)
DialogWindows
DialogBox1 (type)
Rectangle1
Type view
All
User interface
Model
Security
Database
Basic
Advanced
Screen1 ▸ Button1 ▸ MouseClickEventHandler1 ▸ MethodsToCall ▸ MethodContainer1 ▸ InputArguments ▸ Dialog
Retained alarms
AllAlarms
CommController
OPCUAClientController
FactoryTalk_OPtix_Dragable_Dialog_Window
Keyboards
Model
Attribute
Cancel
Select
Remove link
Dialog
Studio Output
No version control enabled for this project.



Add the Close functionality



Test the project
Voilà



9. Reusable Objects. Working with Alias

You can learn to work with Alias in those series of videos

FactoryTalk Optix Reusable Graphics Part 1 - Create Reusable Graphics

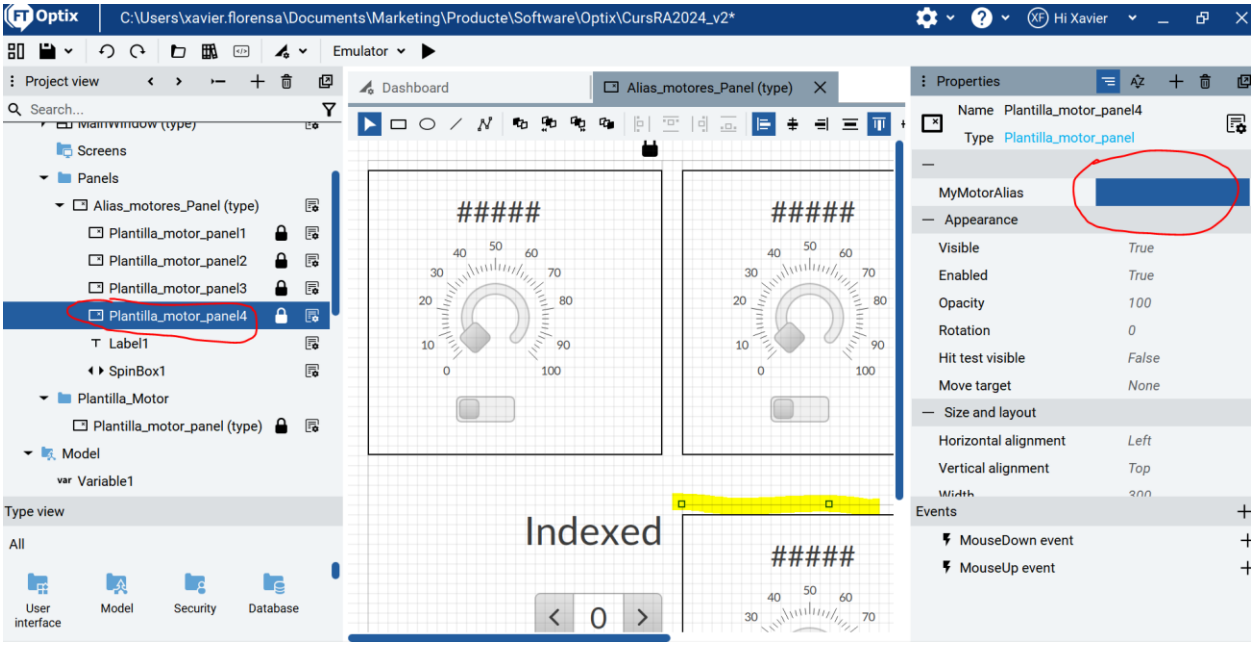
https://www.youtube.com/watch?v=nVEAtHQeFu4&list=PLctxwem-UjOVrNpq7SqswHjy0mYjqHwqj&index=7

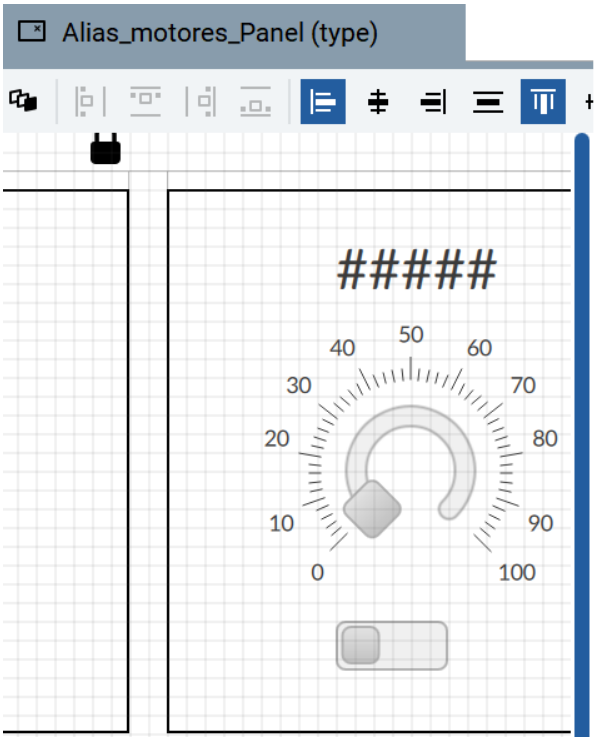
FactoryTalk Optix Reusable Graphics Part 2 - Object Model Alias

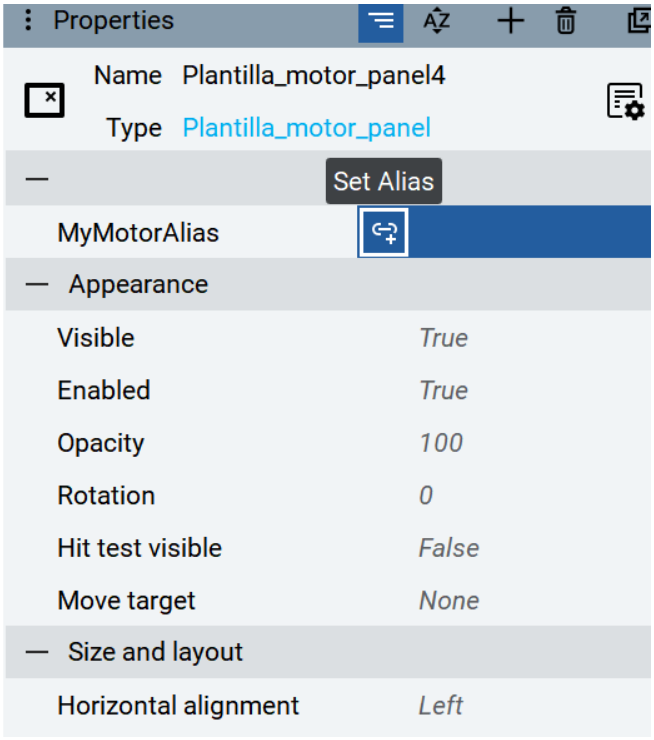
https://www.youtube.com/watch?v=0O_royUATME&list=PLctxwem-UjOVrNpq7SqswHjy0mYjqHwqj&index=8

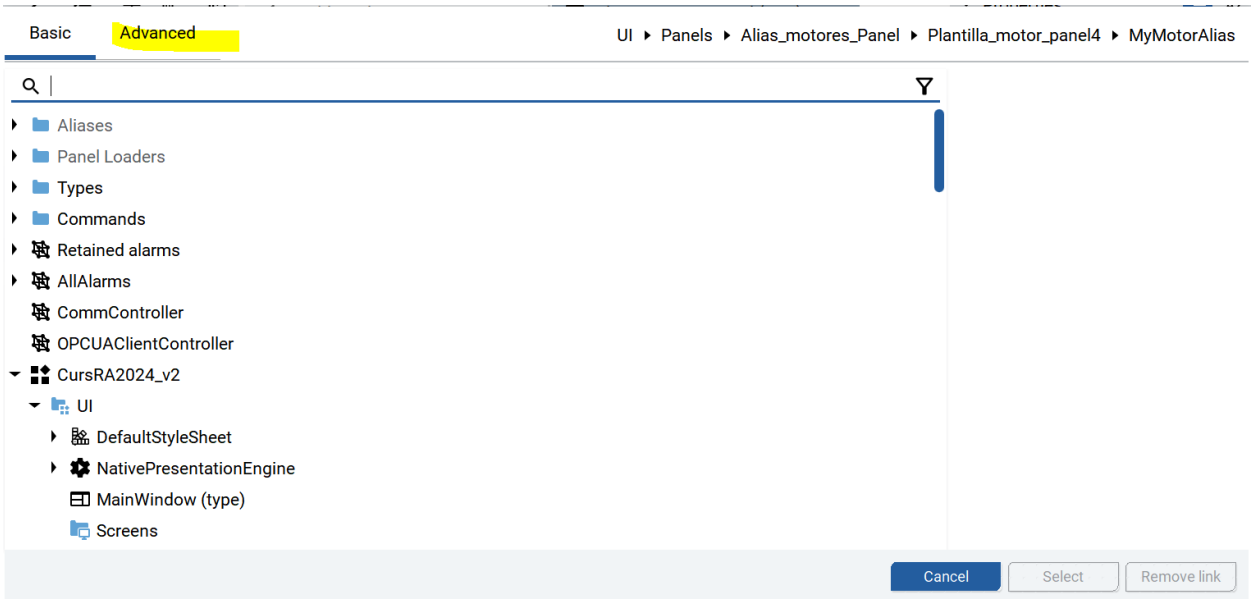
With this you can use several Widgets instances pointing to different UDT's

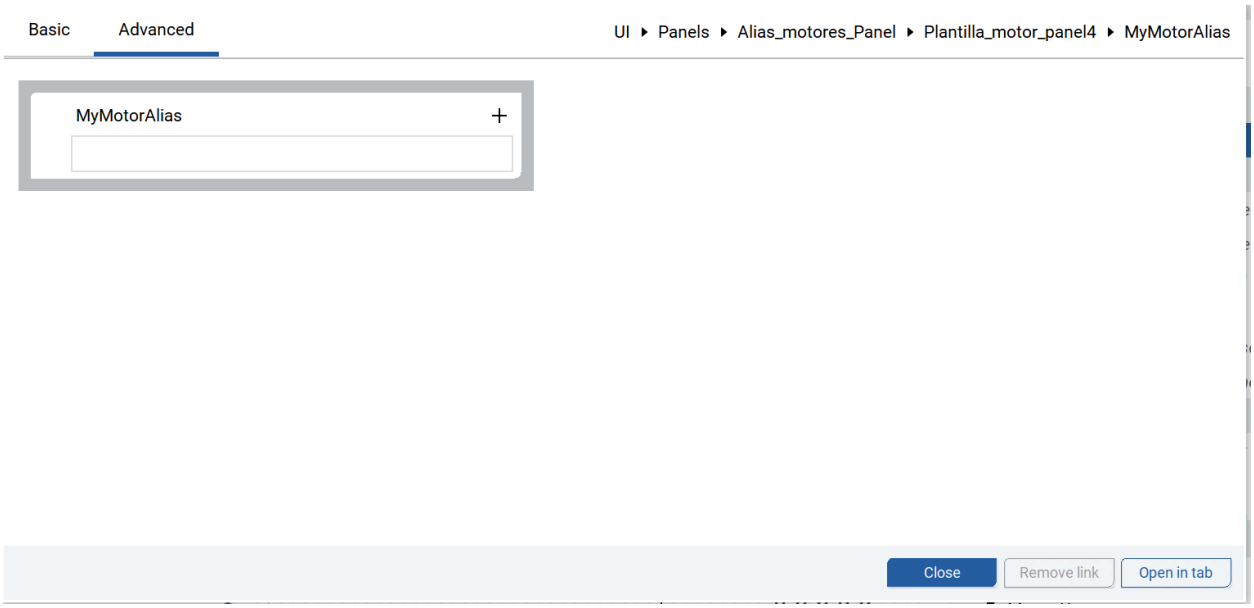
But what if you have lots of motors
On the same project defined on the videos you can create an indexed widget
But instead of pointing directly to the UDT instances let's point this way

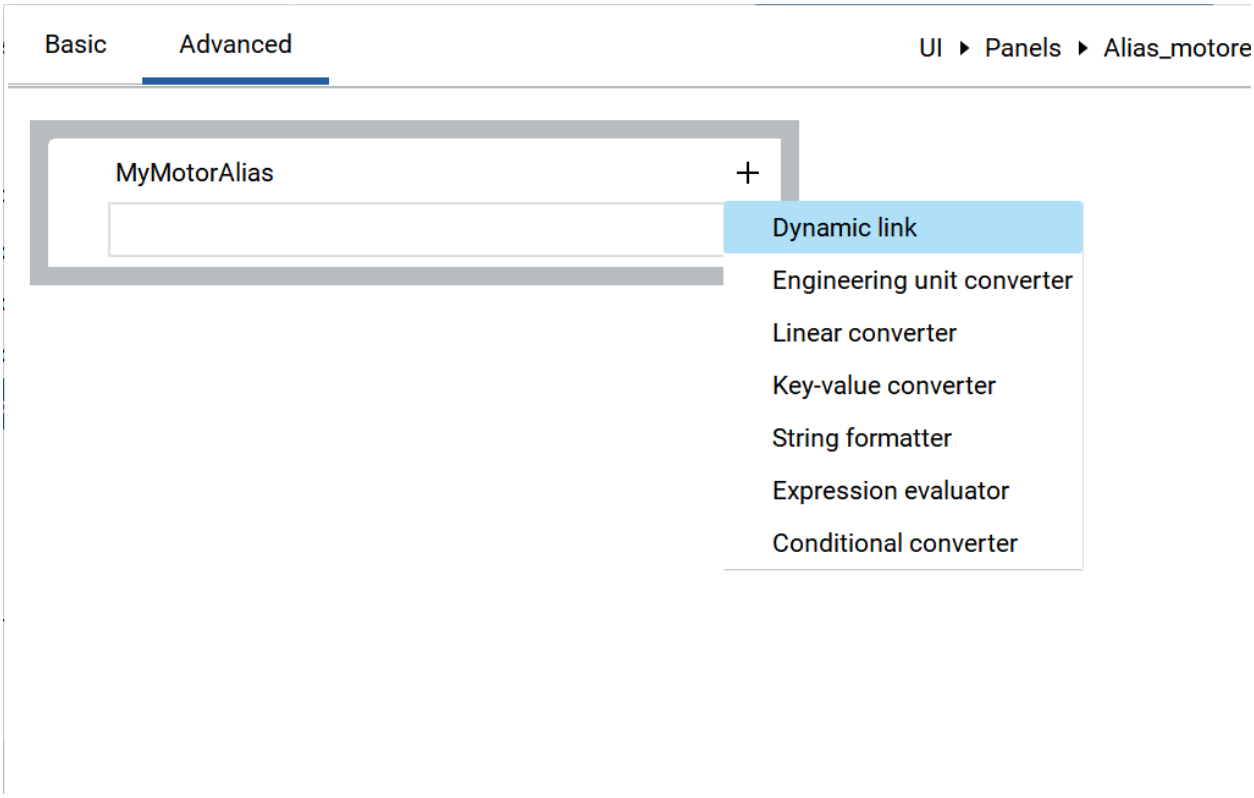


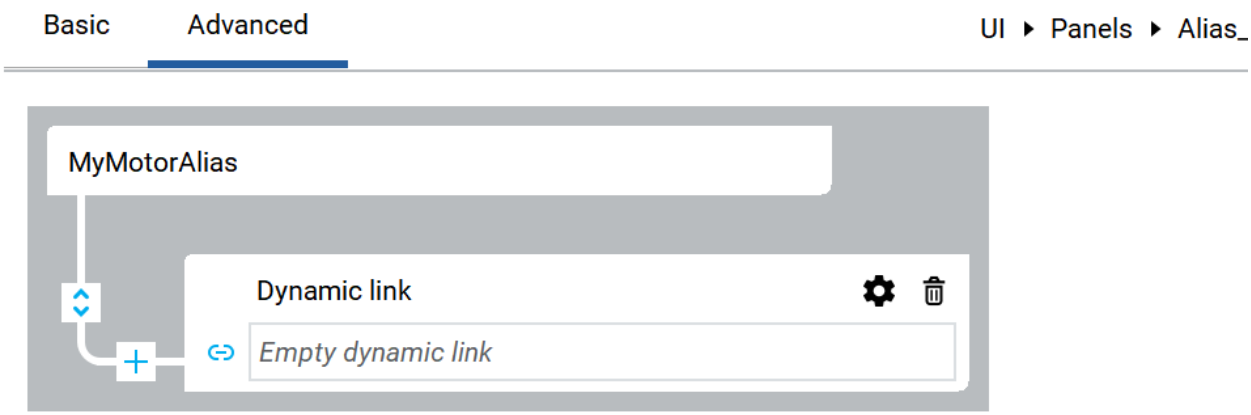


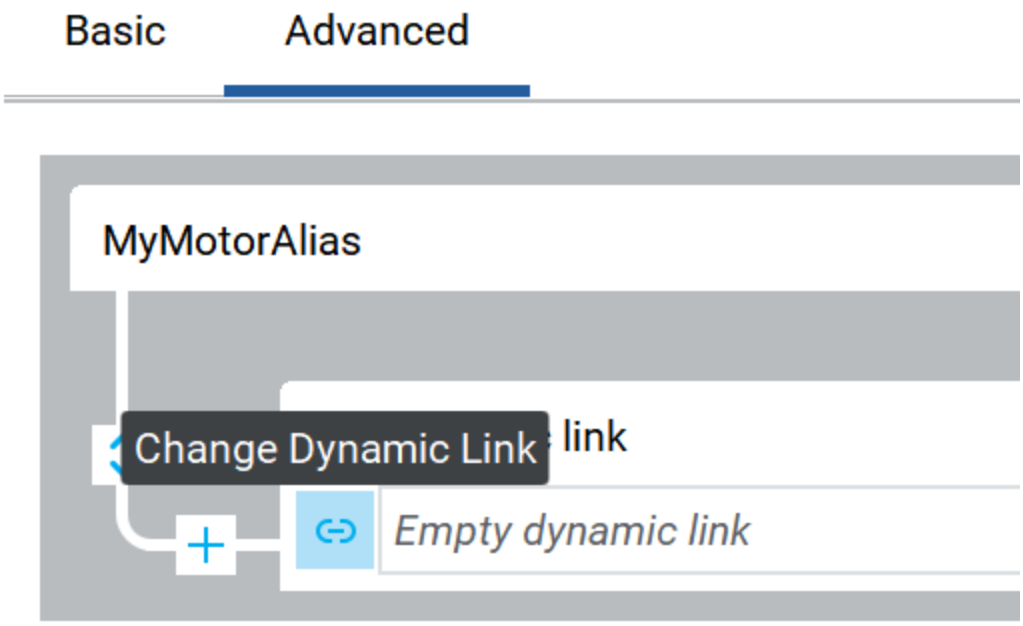


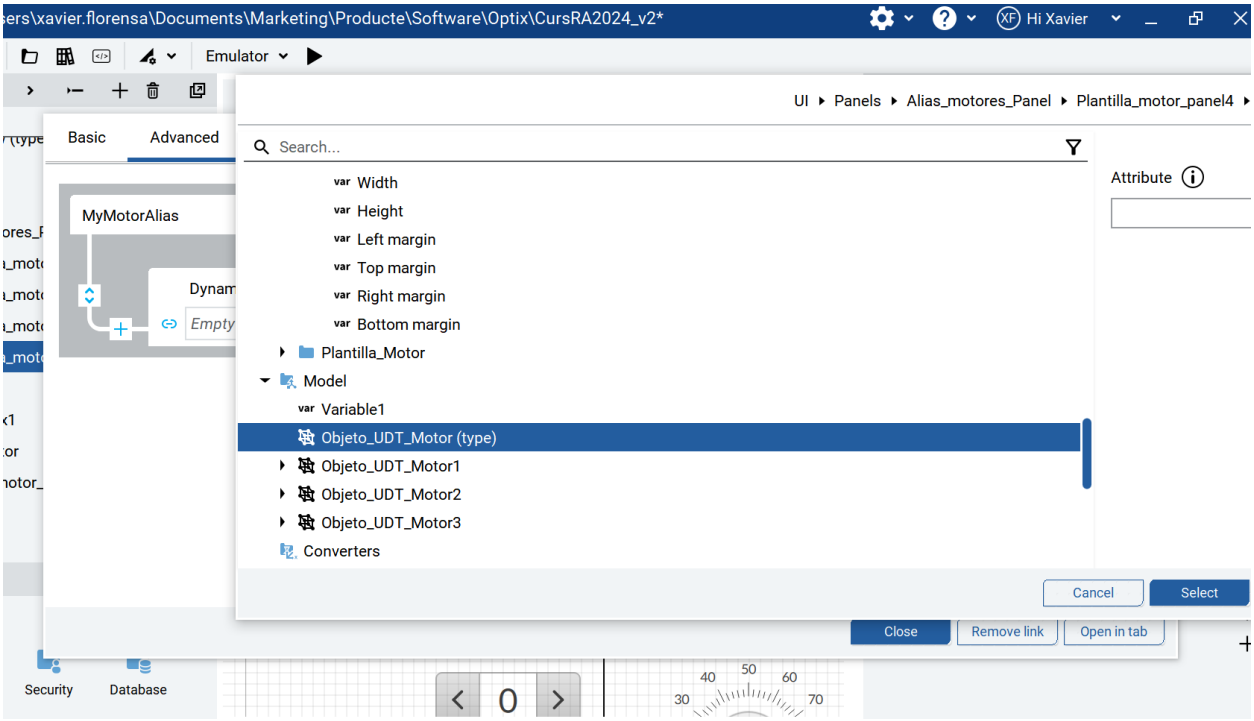

Basic
Advanced
UI ▸ Panels ▸ Alias_motores_Panel ▸ Plantilla_motor_panel4 ▸ MyMotorAlias
Aliases
Panel Loaders
Types
Commands
Retained alarms
AllAlarms
CommController
OPCUAClientController
CursRA2024_v2
UI
DefaultStyleSheet
NativePresentationEngine
MainWindow (type)
Screens
Cancel
Select
Remove link


Basic
Advanced
UI ▸ Panels ▸ Alias_motores_Panel ▸ Plantilla_motor_panel4 ▸ MyMotorAlias
MyMotorAlias
Close
Remove link
Open in tab

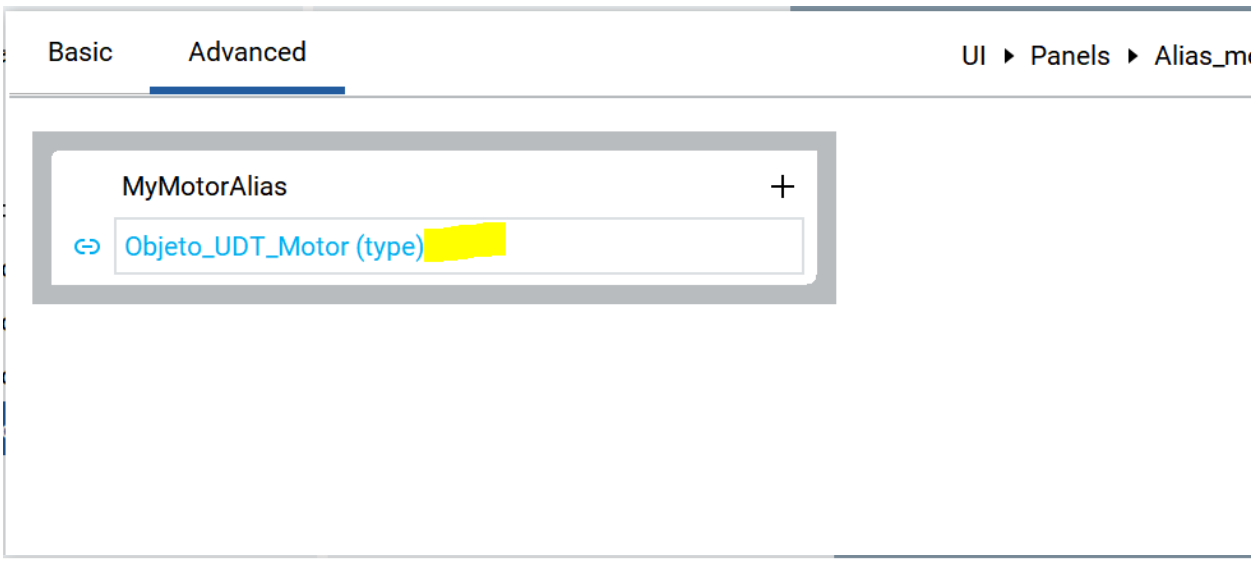

Basic
Advanced
UI ▸ Panels ▸ Alias_motore
MyMotorAlias
Dynamic link
Engineering unit converter
Linear converter
Key-value converter
String formatter
Expression evaluator
Conditional converter

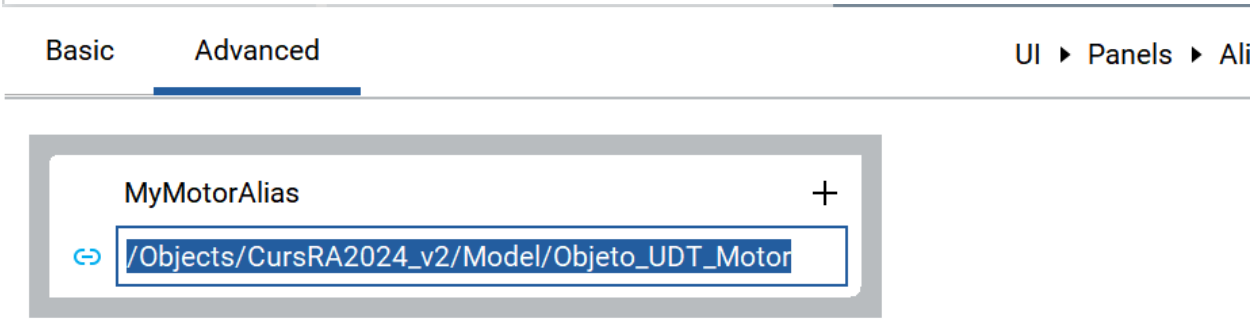

Basic
Advanced
UI ▸ Panels ▸ Alias_
MyMotorAlias
Dynamic link
Empty dynamic link

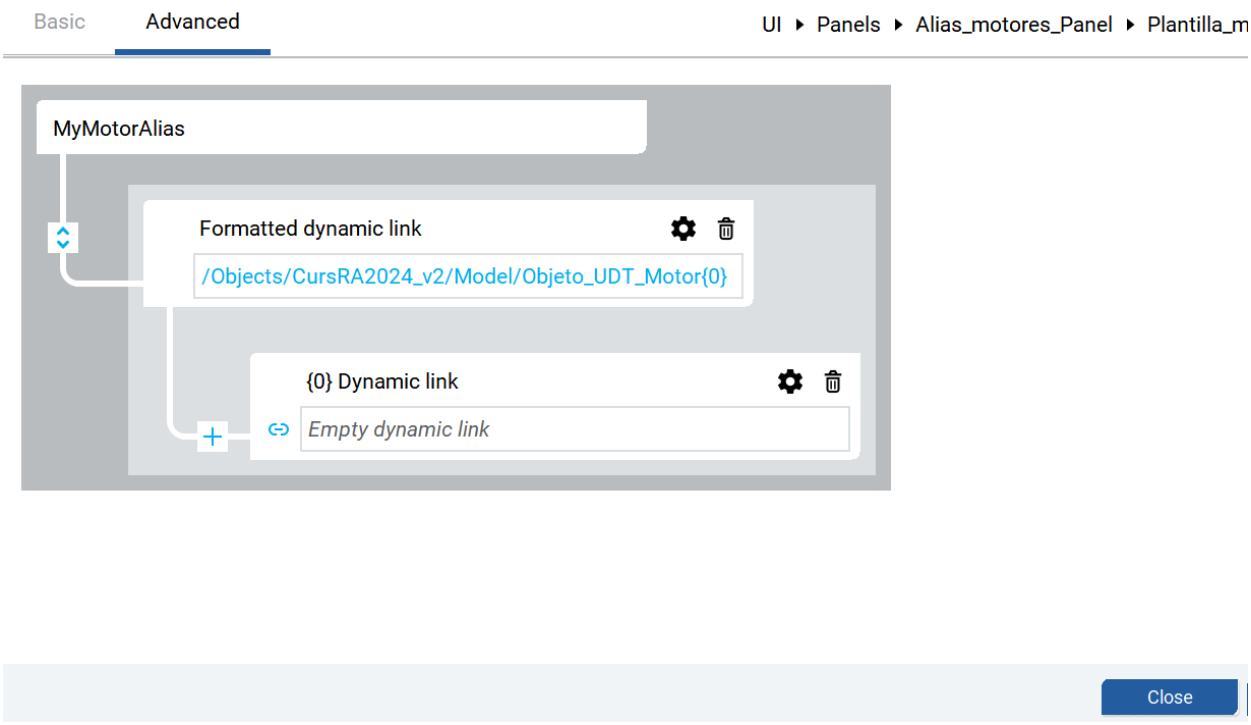




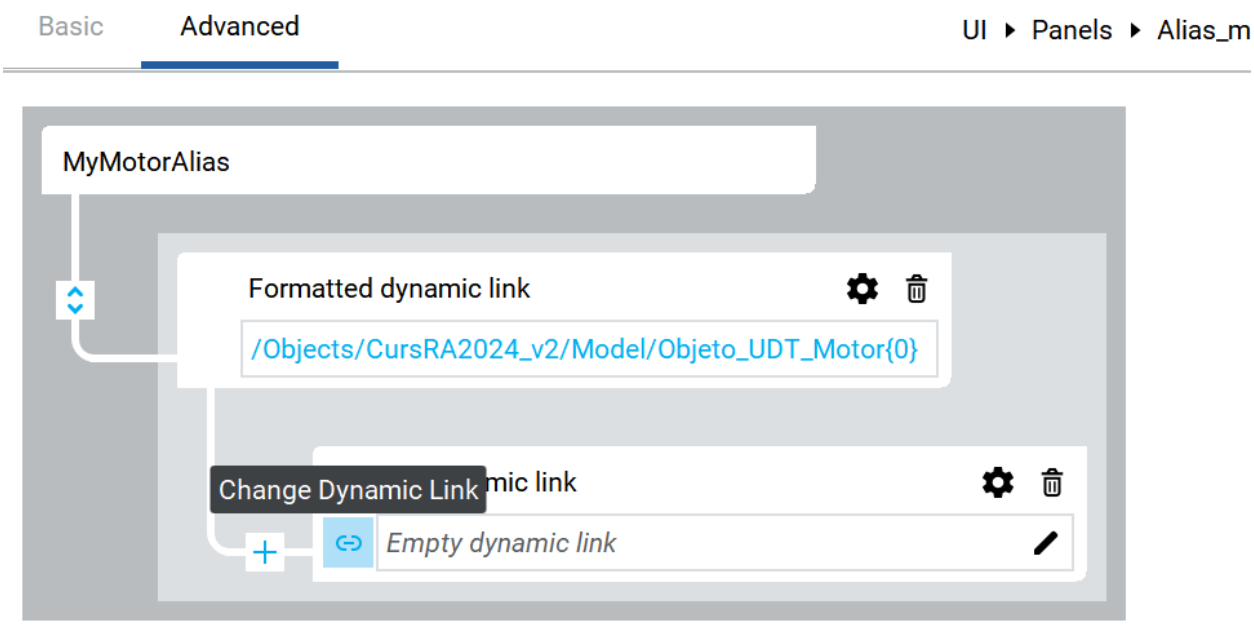
Add {0} at the end and click enter

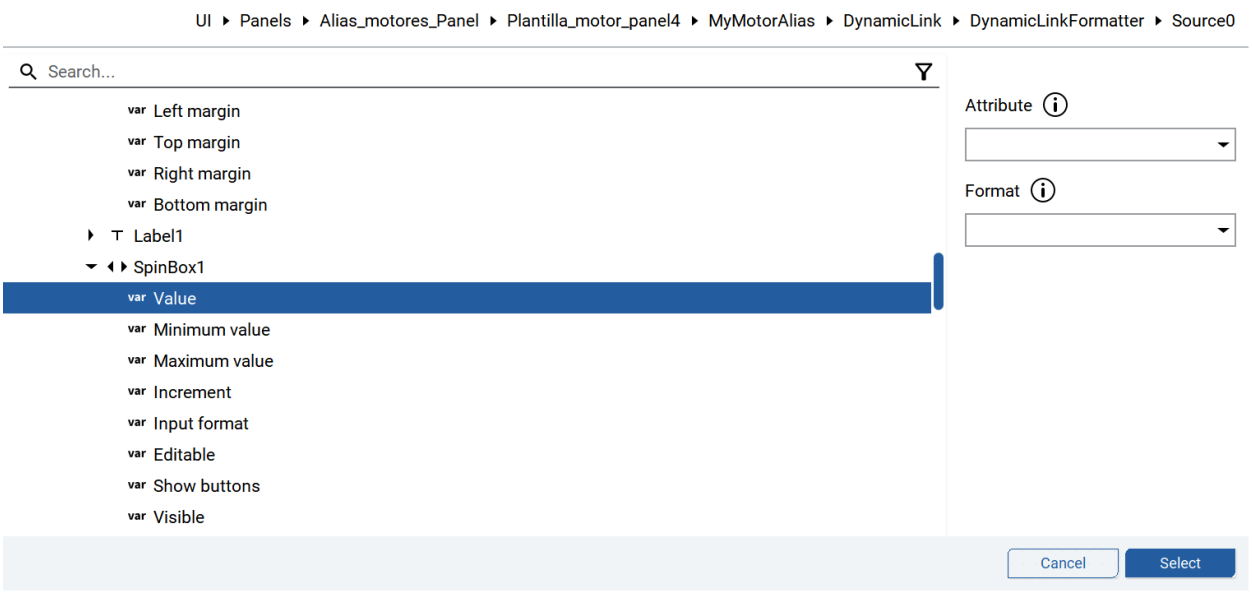

Basic
Advanced
UI ▸ Panels ▸ Alias_m
MyMotorAlias
Objeto_UDT_Motor (type)

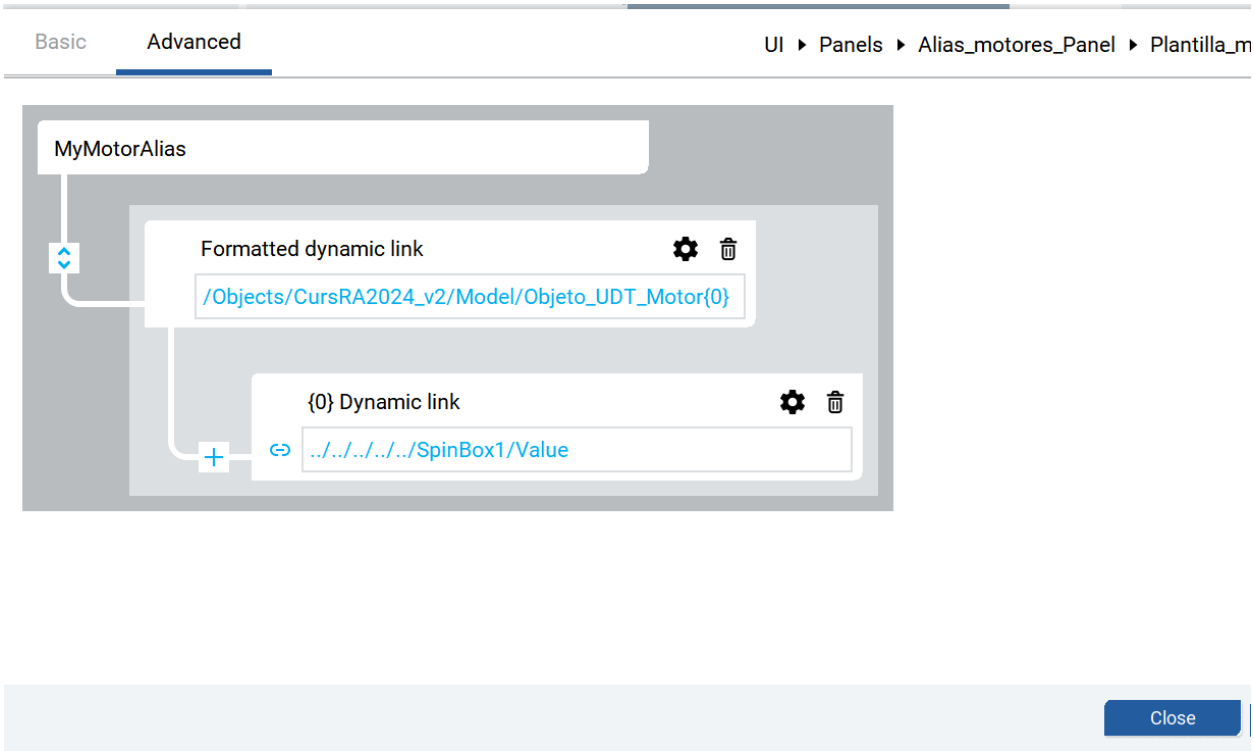

Basic
Advanced
UI ▸ Panels ▸ Ali
MyMotorAlias
/Objects/CursRA2024_v2/Model/Objeto_UDT_Motor

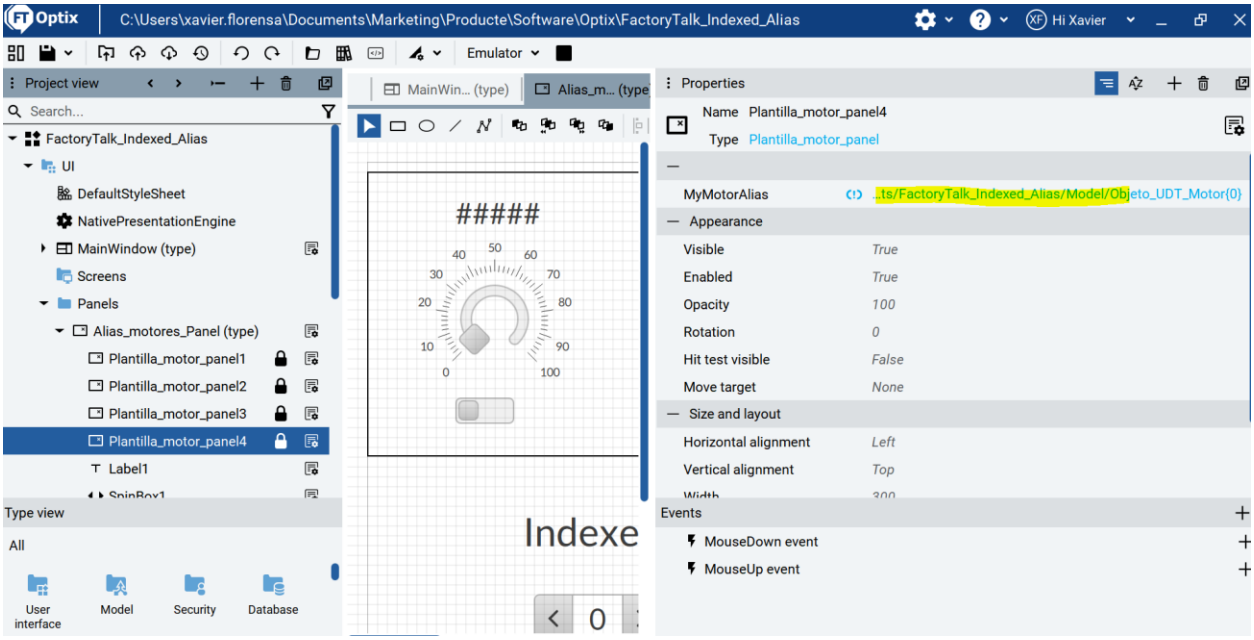


Select the dynamic link



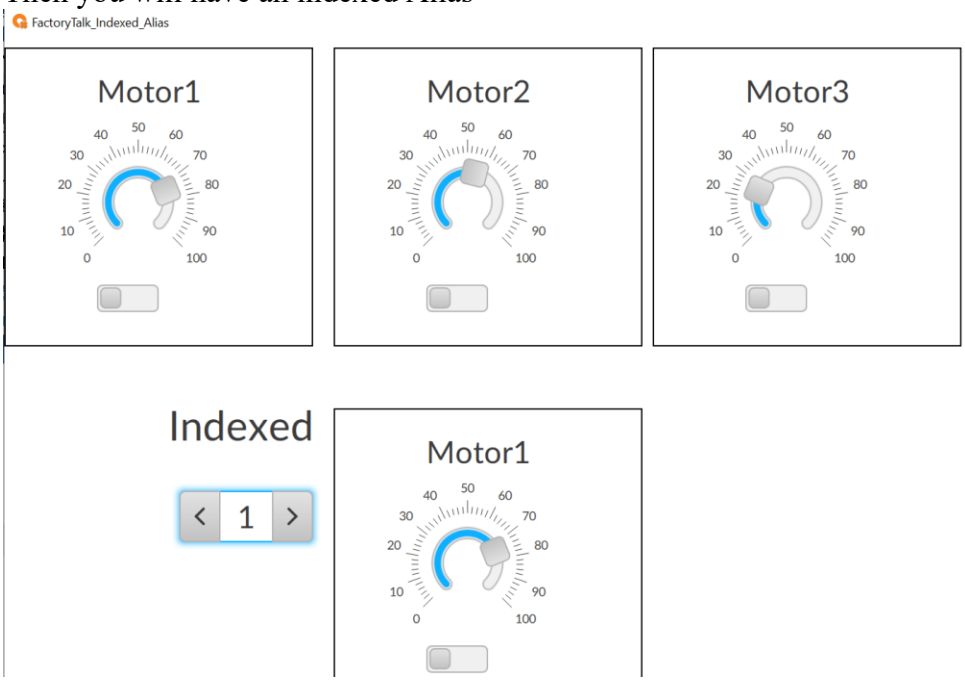

UI ▸ Panels ▸ Alias_motores_Panel ▸ Plantilla_motor_panel4 ▸ MyMotorAlias ▸ DynamicLink ▸ DynamicLinkFormatter ▸ Source0
Search...
var Left margin
var Top margin
var Right margin
var Bottom margin
Label1
SpinBox1
var Value
var Minimum value
var Maximum value
var Increment
var Input format
var Editable
var Show buttons
var Visible
Attribute
Format
Cancel
Select

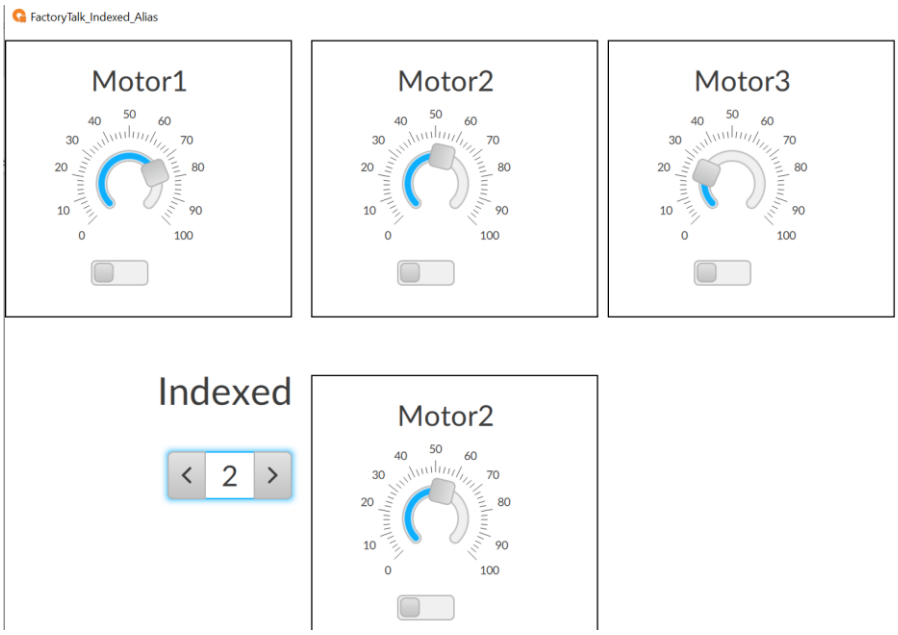

Basic
Advanced
UI ▸ Panels ▸ Alias_motores_Panel ▸ Plantilla_m
MyMotorAlias
Formatted dynamic link
/Objects/CursRA2024_v2/Model/Objeto_UDT_Motor{0}
{0} Dynamic link
../../../../../SpinBox1/Value
Close

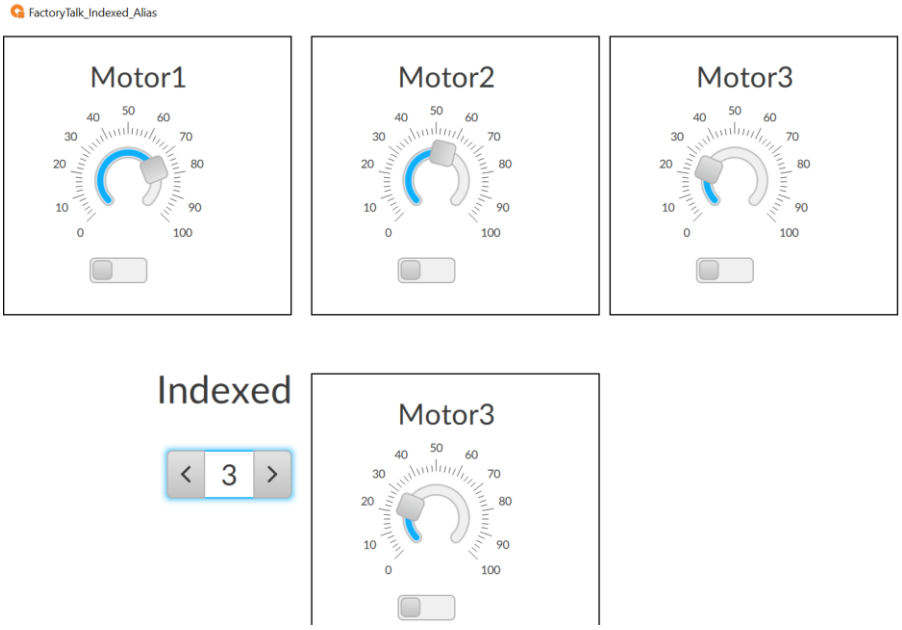


Voilà

Then you will have an indexed Alias







You can find the code here

https://github.com/xavierflorensa/FactoryTalk_Indexed_Alias.git

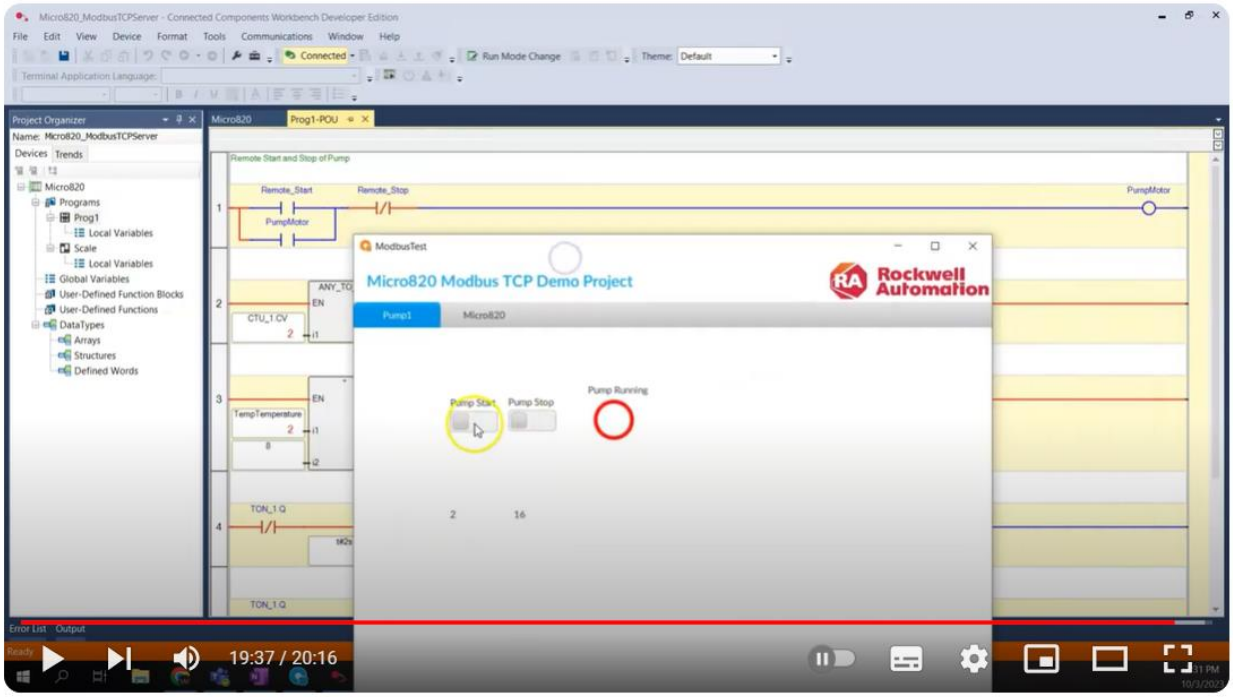
10. Communications with Micro8XX PLC's

From Optix version 1.03 and so on, the driver is enabled if you activate the features option on your FT Optix configuration, and restart FTOptix. So if you have this version, you can skip this section.
You can see it on this video
https://www.youtube.com/watch?v=UKUBQ2-sX24

Before version 1.03

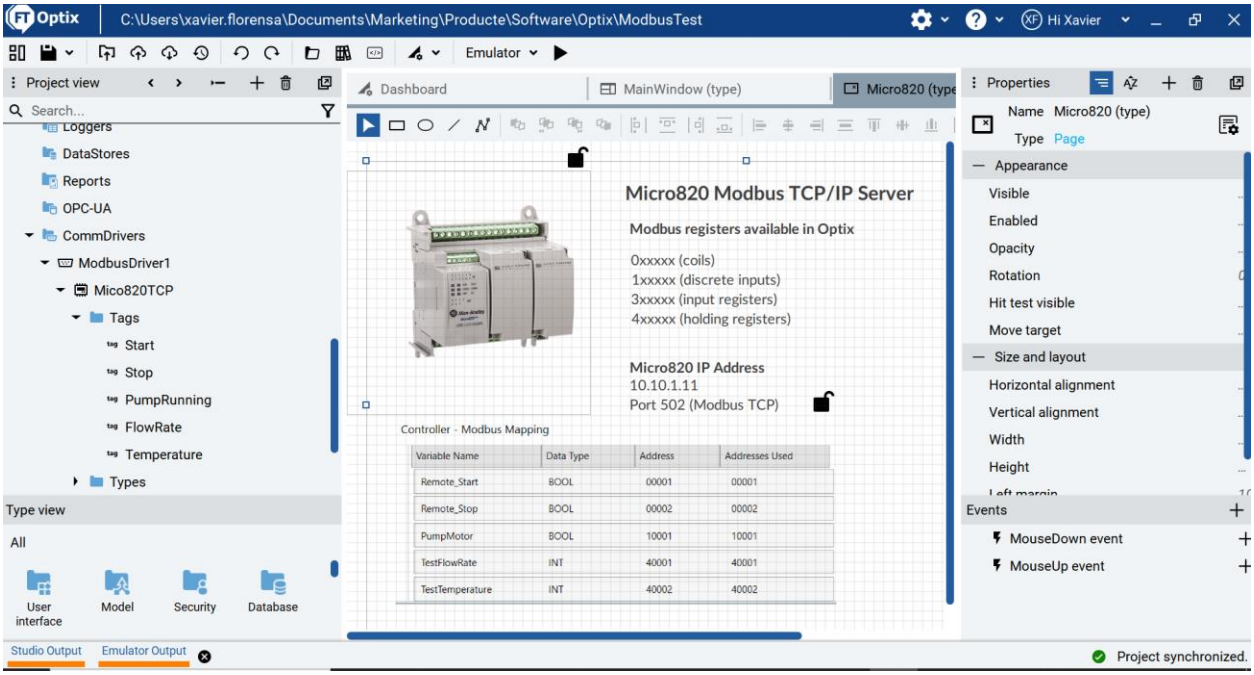
You can follow this video from Wayne Welk
https://www.youtube.com/watch?v=k4Q0C0uCeG0&t=848s

And you can download the code from here
https://github.com/wawelk/ModbusTest



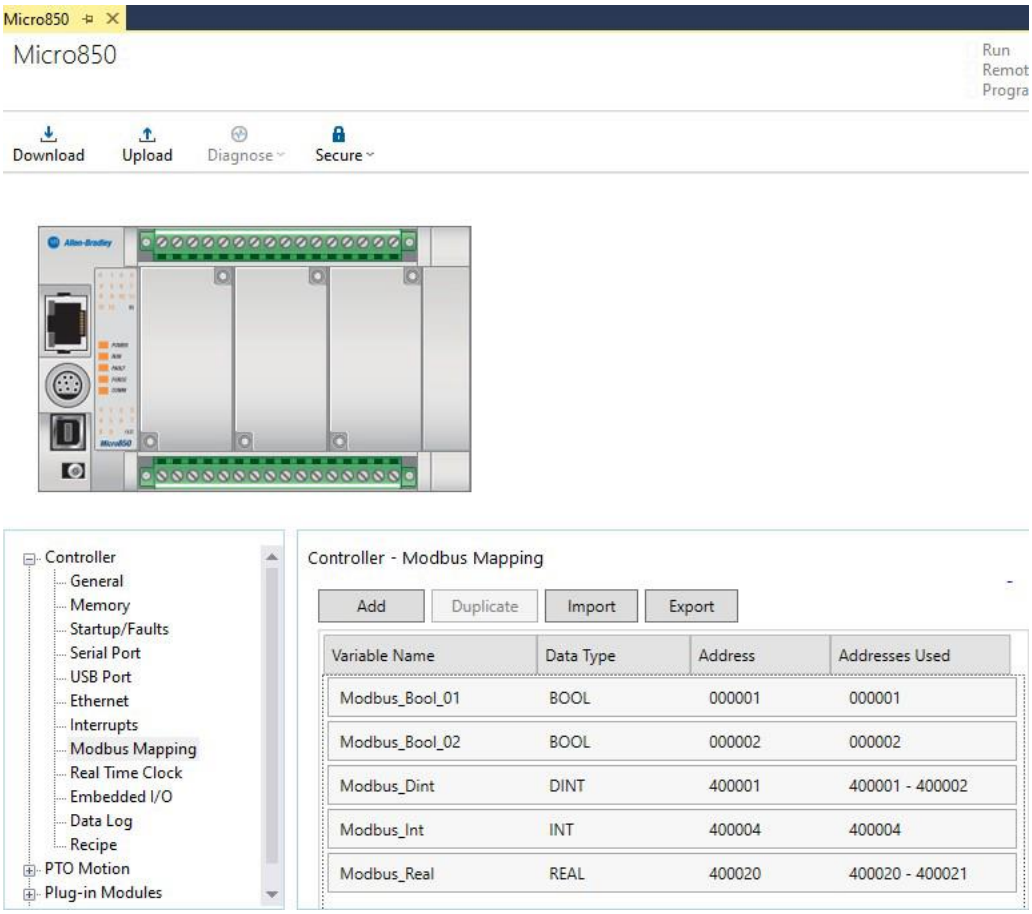
Configuring FactoryTalk Optix Modbus TCP Communications with a Micro820 PLC

Wayne Welk
253 suscriptores
Suscrito
6
Compartir
Descargar



Document ID QA64805

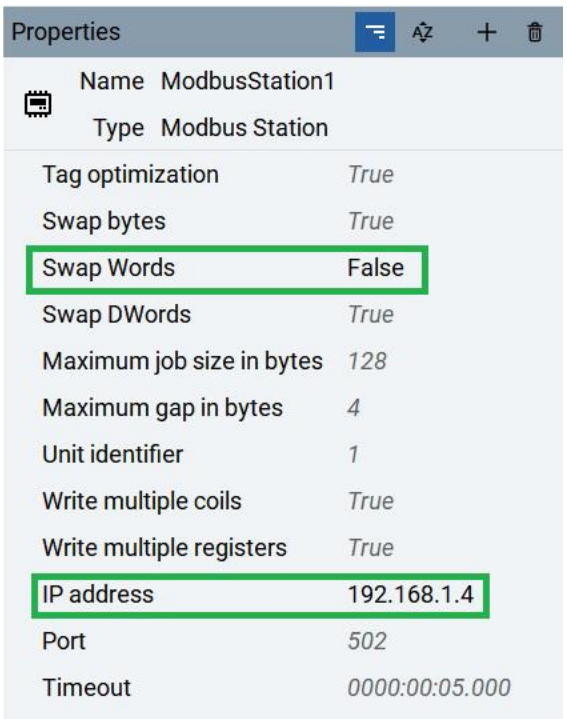
Published Date 05/11/2023
FactoryTalk Optix: Modbus TCPcommunication with Micro850
Micro800 is not compatible with RA EtherNet/IP Driver. But FactoryTalk Optix has ModbusTCP Driver that can be used to communicate with Micro800 controllers over Ethernet/IP.To configure Modbus TCP on the Micro800 controller:
In Project Organizer or Controller Organizer, double-click the controller to open thecontroller workspace.
Check if Modbus TCP is enabled in the controller properties, like in thistechnote: QA14133 - How to configure the Ethernet port for Modbus TCP in theMicro850
In the Controller tree, click Modbus Mapping --> Add to add tags that will be sharedwith FactoryTalk Optix.



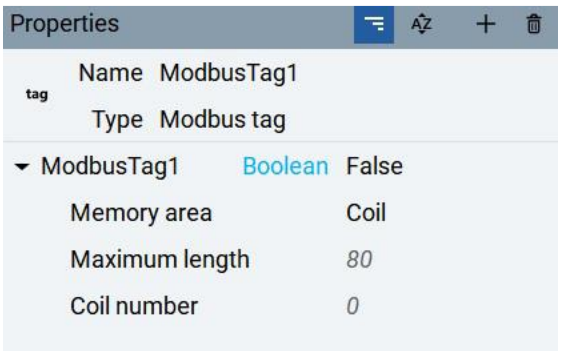
Note: See this technote for Micro800 Modbus mapping limitations: QA49552 -Micro800: Modbus mapping limitation

Important: A valid numeric address is five or six characters in length with the firstdigit representing the register type.0xxxxx (coils)
1xxxxx (discrete inputs)
3xxxxx (input registers)
4xxxxx (holding registers)

To configure Modbus TCP in the FactoryTalk Optix:
Navigate to Project view and right-click CommDrivers. Select New to get list ofsupported communication protocols. Select Modbus Driver. The driver will be addedand named automatically within your Project view.
Right click Modbus Driver1 --> New --> Modbus Station toopen Modbus Station1 properties.
Change Swap Words to False and in the IP address field set the IP address of thecontroller.



Expand Modbus Station1, right-click Tags --> New --> Modbus Tag to add and configure tags mapped in the controller.Click on the blue tag data type to match it with the controllers mapped tag.Important: Some tag data types are different in Connected ComponentWorkbench and FactoryTalk Optix. For example INT is Int16, DINT is Int32, REAL isFloat.
Memory area - select the correct register type of the mapped tag.
Coil number - set address of the mapped tag.Important: Connected Component Workbench is starting addressing at 1,while FactoryTalk Optix is starting addressing at 0. For example if tag is mappedin Connected Component Workbench to 400001, in FactoryTalk Optix you need to enter Holding Register in the Memory area, and set Register number to 0

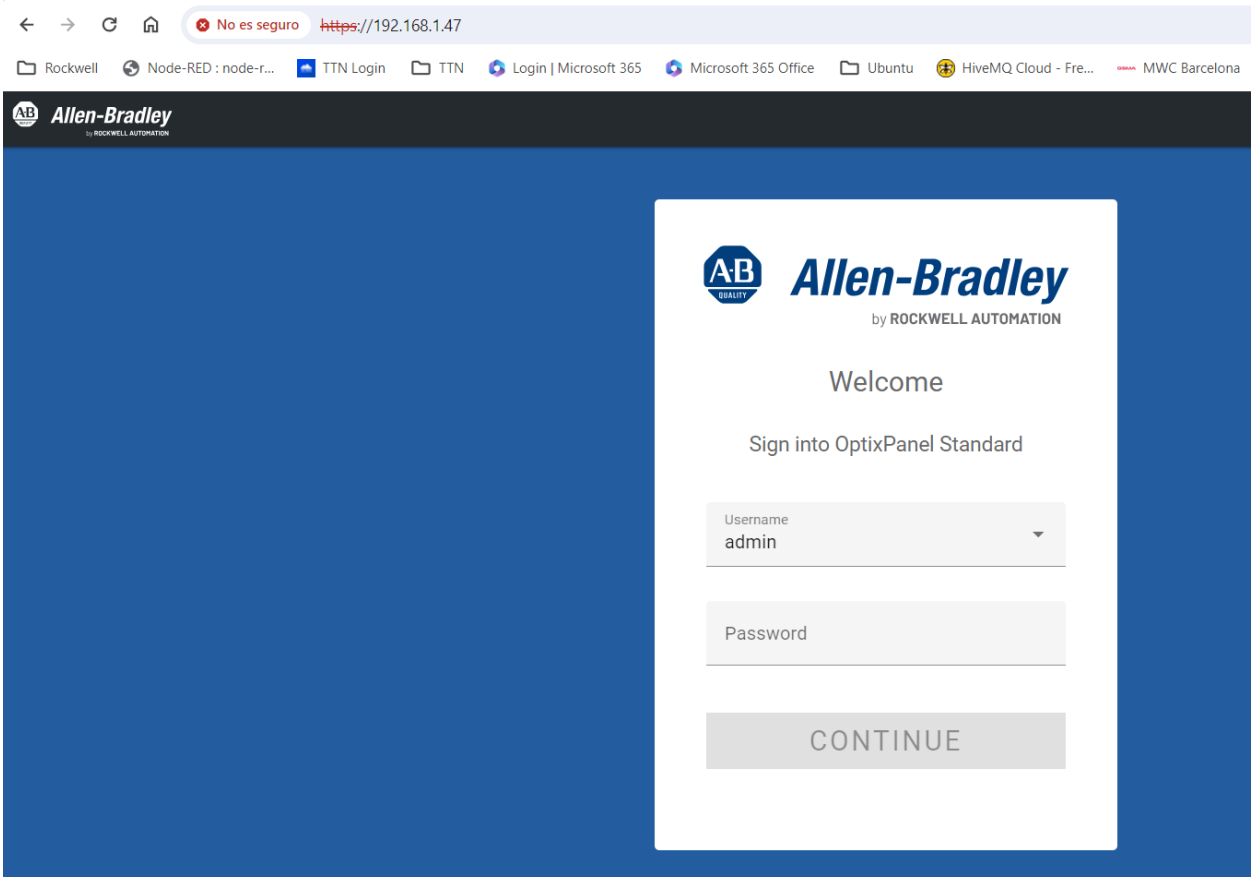


Red X wireframe on the object using modbus tag can be caused by:
FactoryTalk Optix has not established communication with the controller,
Tag data type in FactoryTalk Optix do not match the tag data type in the controller,Important: Some tag data types are different in Connected Component Workbenchand FactoryTalk Optix. For example Int16 is INT, Int32 is DINT, Float is REAL.
FactoryTalk Optix is checking different register type or different address than it ismapped to,Important: Connected Component Workbench is starting addressing at 1,while FactoryTalk Optix is starting addressing at 0. For example if tag is mappedin Connected Component Workbench to 400001, in FactoryTalk Optix you need toenter Holding Register in the Memory area, and set Register number to 0.

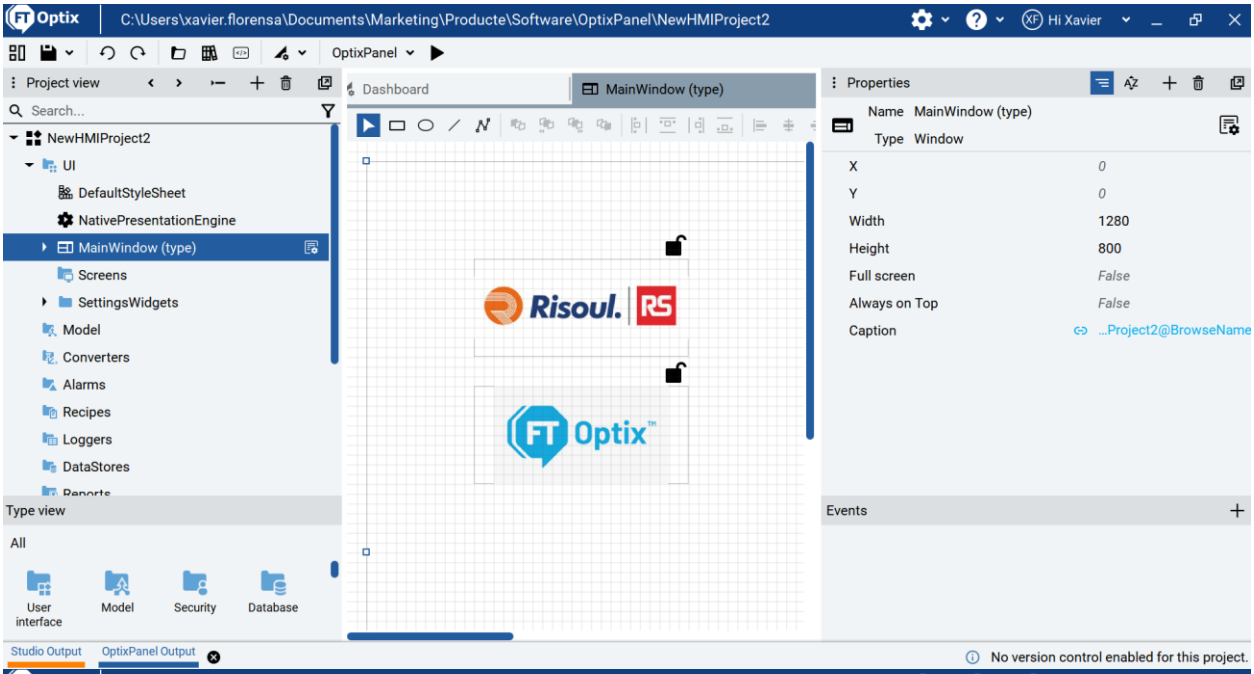
11. First steps with OptixPanel

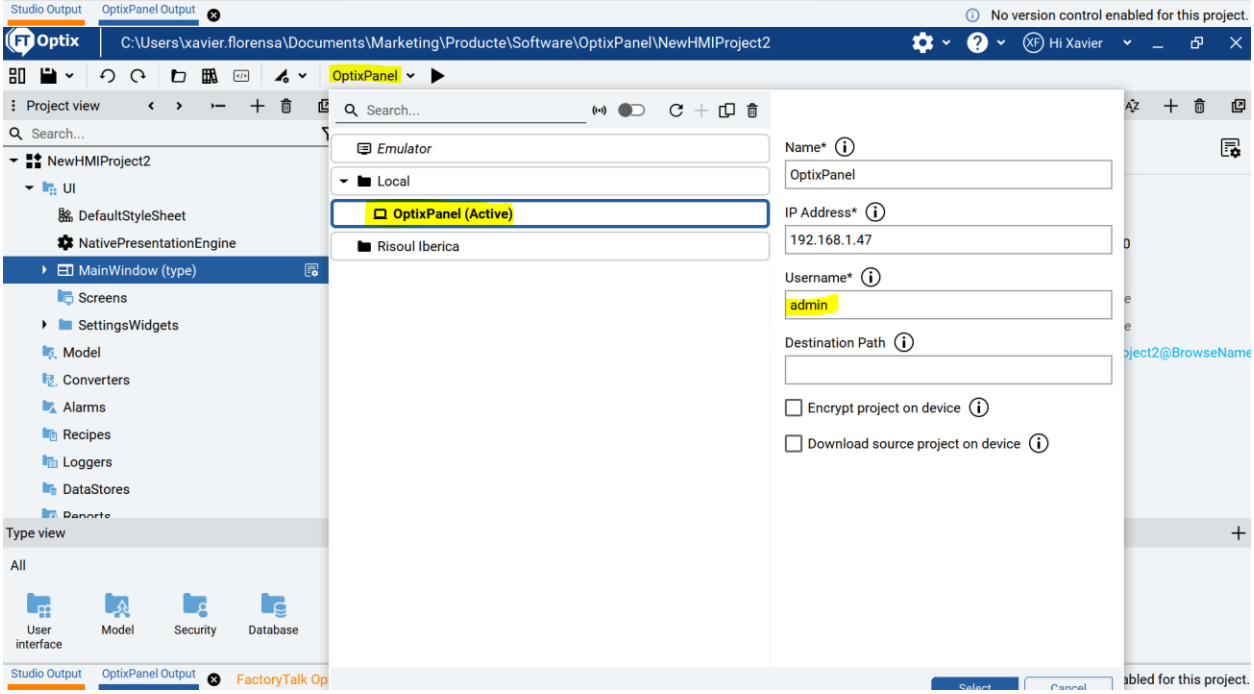
You have a video with an introduction to Remote Access on FT Optix Panel here
https://www.youtube.com/watch?v=kDGCZ4npzLE&t=7s

These are the unit we are testing
Standard 12.1 inches
2800S-121FM-N1S
Ethernet Ports
WAN Eth 2 MAC 5C:88:16:FC:32:71 DHCP
LAN Eth1 MAC 5C:88:16:FC:32:70 Fixed IP 192.168.1.101/255.255.255.0
Accessing configuration

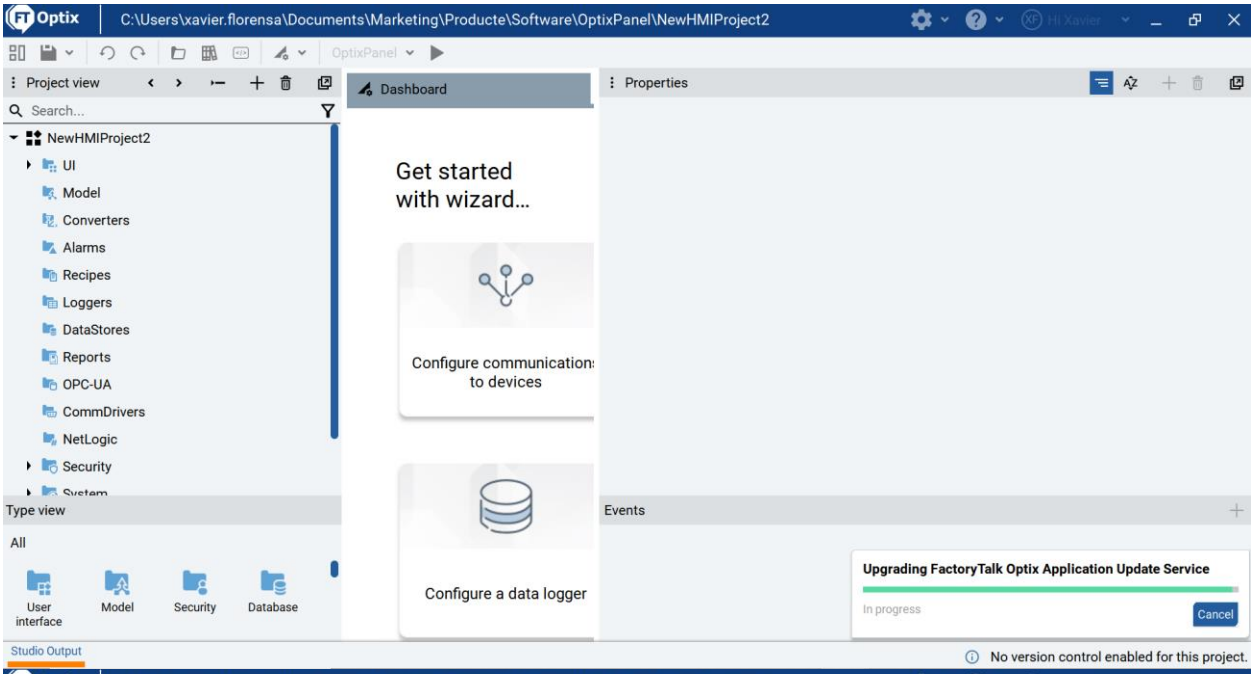


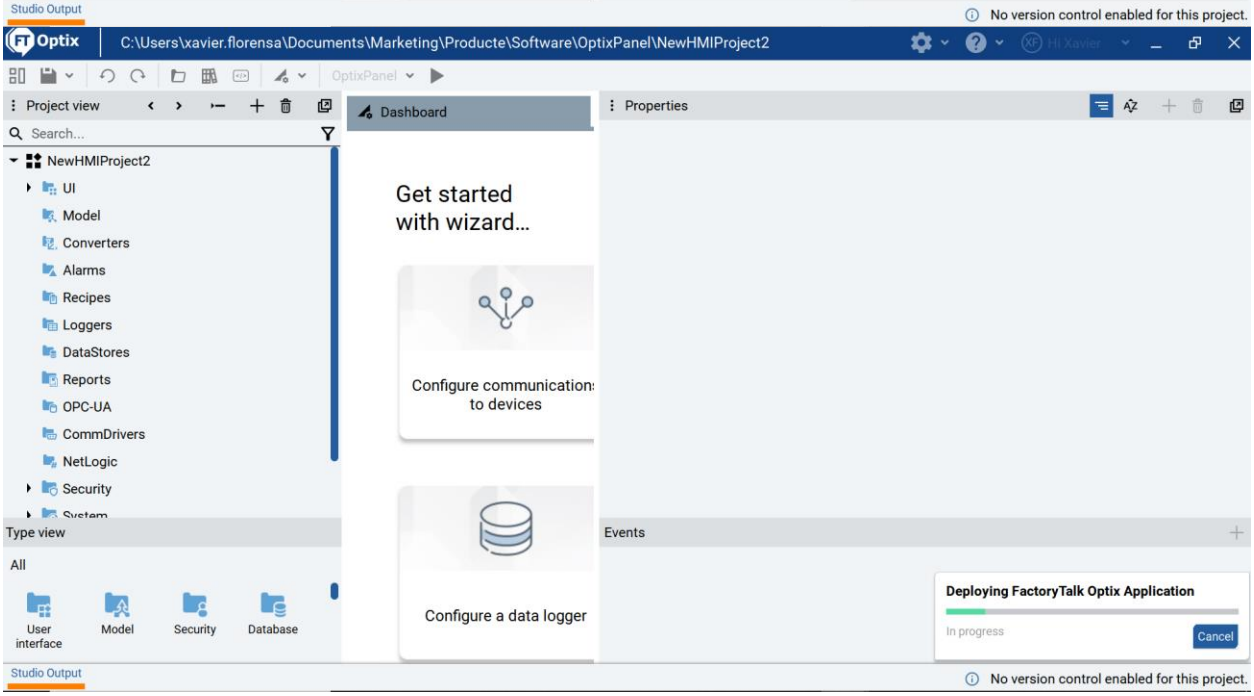
You have to introduce a password
First time is
User:admin
Password:admin
Let's change to Risoul10
A message with:
"No FactoryTalk Optix application found, please upload one using FactoryTalk Optix Studio"
Let's create a new application and select the destination different from Emulator but OptixPanel



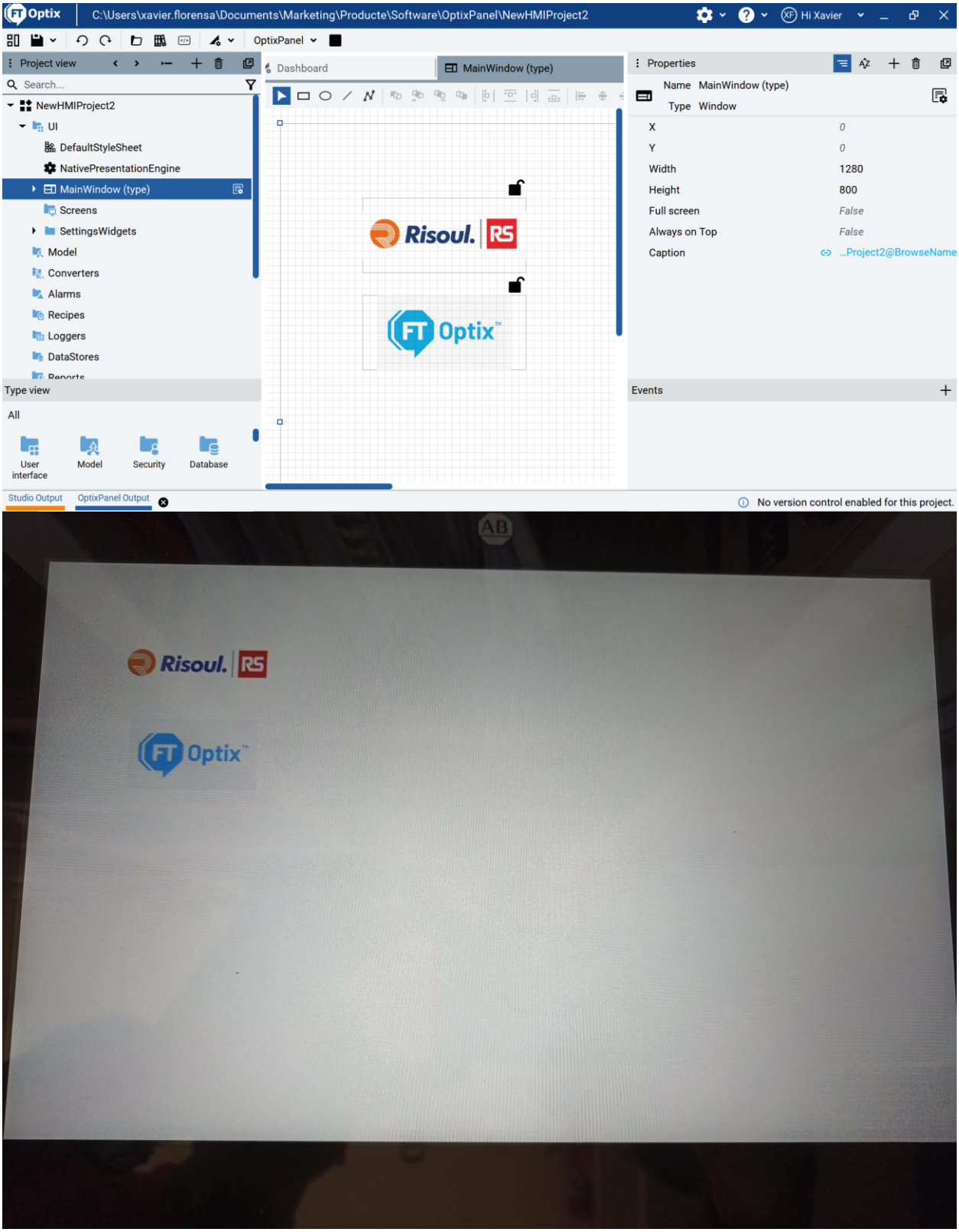


And test with the play button on the Panel
Admin
Password: Risoul10

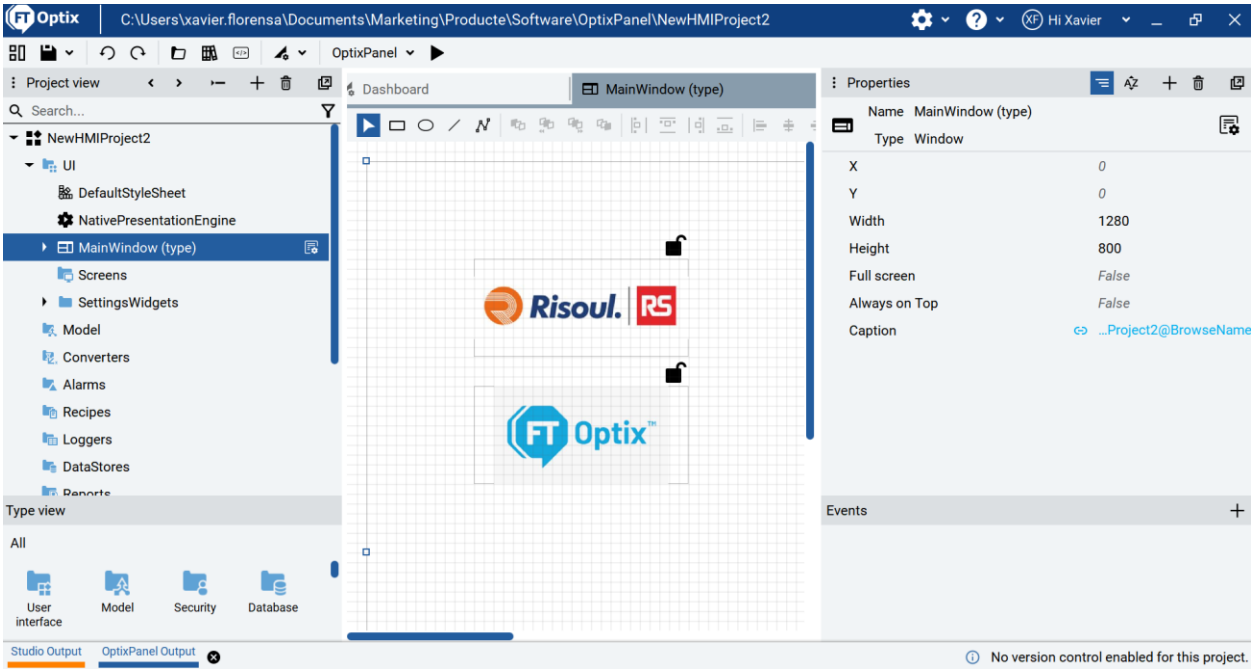

Optix
C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\OptixPanel\NewHMIProject2
Project view
Search...
NewHMIProject2
UI
Model
Converters
Alarms
Recipes
Loggers
DataStores
Reports
OPC-UA
CommDrivers
NetLogic
Security
Type view
All
User interface
Model
Security
Database
Dashboard
Get started with wizard...
Configure communications to devices
Configure a data logger
Properties
Events
Upgrading FactoryTalk Optix Application Update Service
In progress
Cancel
Studio Output
No version control enabled for this project.


Optix
C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\OptixPanel\NewHMIProject2
Project view
Search...
NewHMIProject2
UI
Model
Converters
Alarms
Recipes
Loggers
DataStores
Reports
OPC-UA
CommDrivers
NetLogic
Security
Type view
All
User interface
Model
Security
Database
Dashboard
Get started with wizard...
Configure communications to devices
Configure a data logger
Properties
Events
Deploying FactoryTalk Optix Application
In progress
Cancel
Studio Output
No version control enabled for this project.

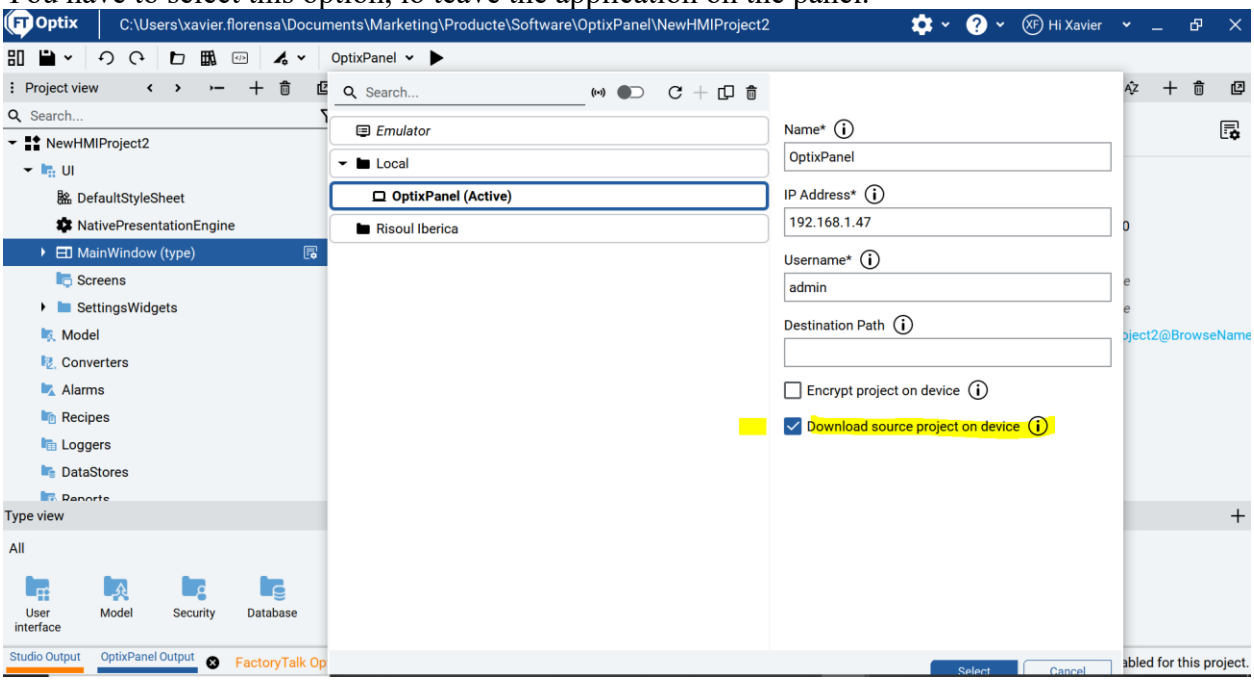
Success



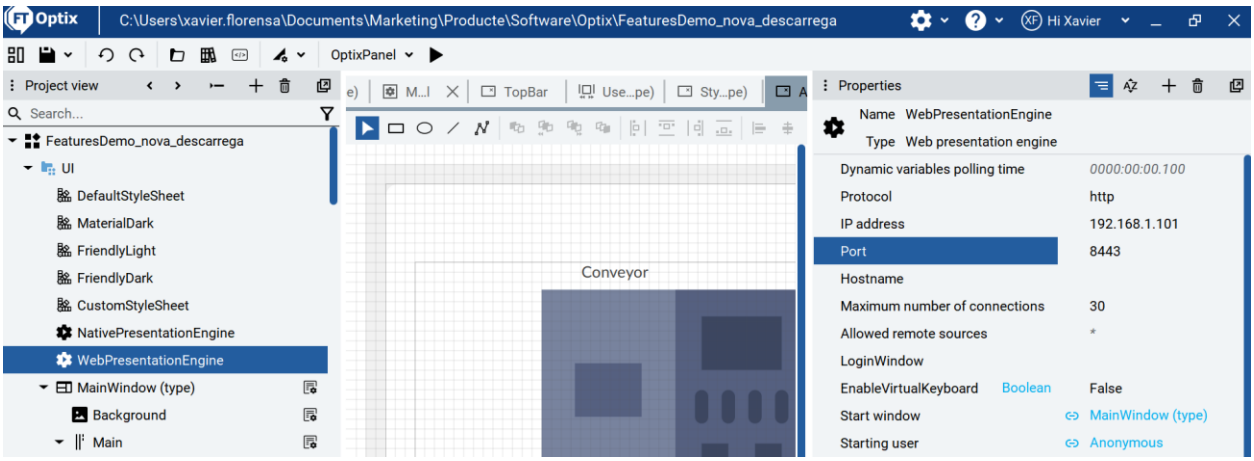
But when you hit the stop button, then the screen becomes full white.



You have to select this option, lo teave the application on the panel.

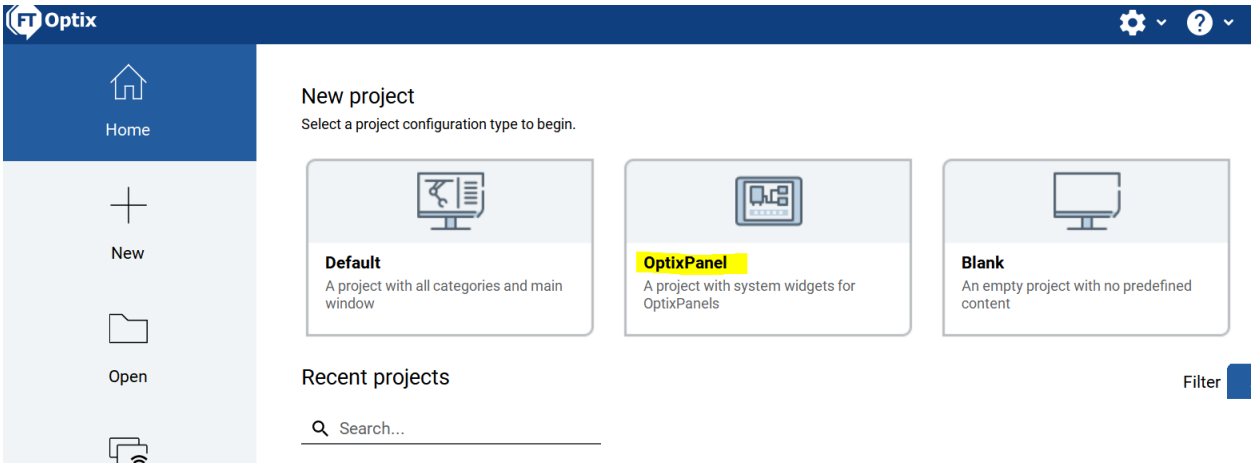


But still when clicking stop the screen becomes full white.
But if you power OptixPanel again it will open with the new application. After 10 seconds.
That's all.
You can use web access from your Laptop
Just plug your Laptop with a patchcord to Eth1 (LAN)
Set your IP laptop to 192.168.1.10 for instance

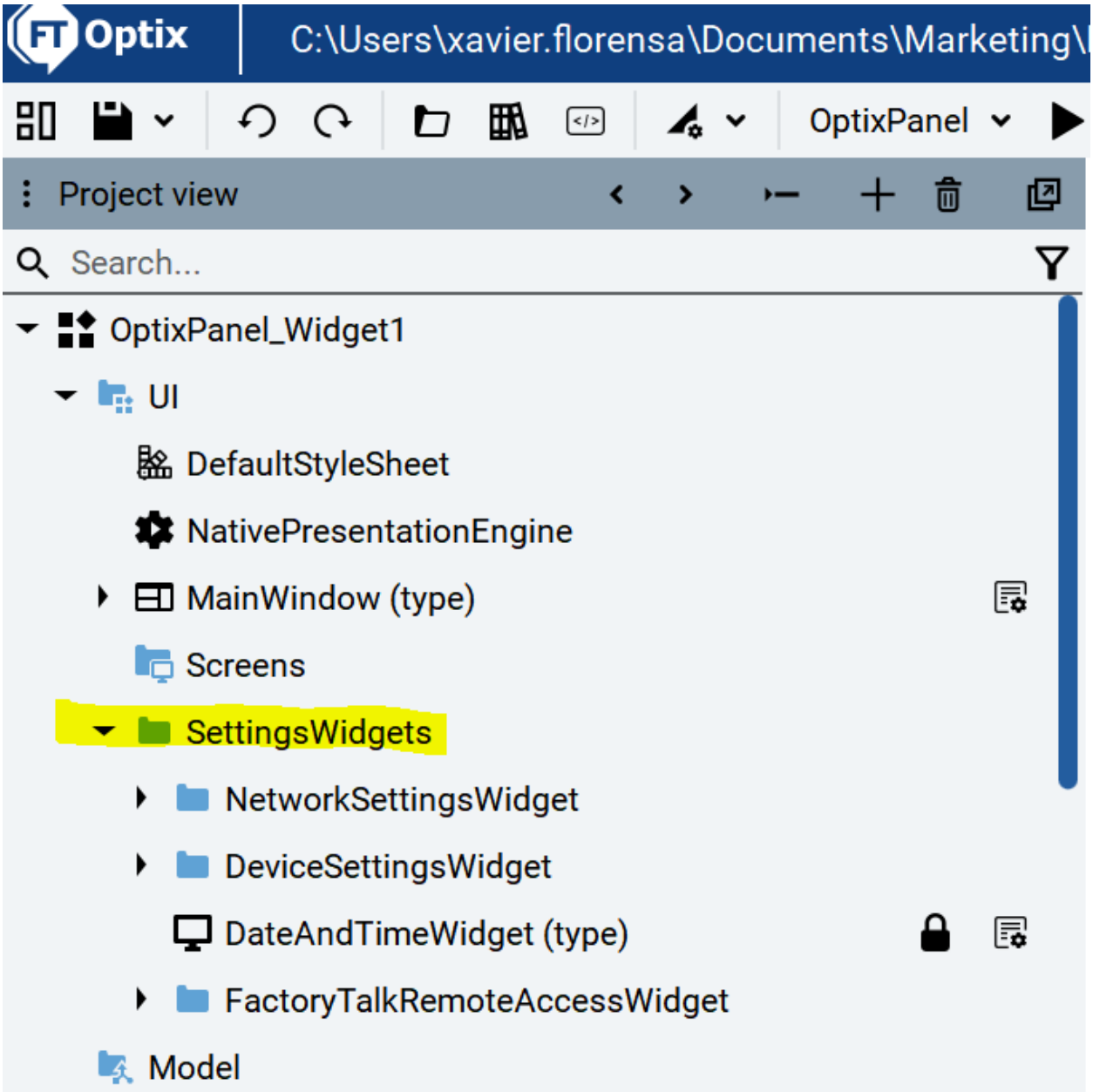


11.1. OptixPanel Widgets

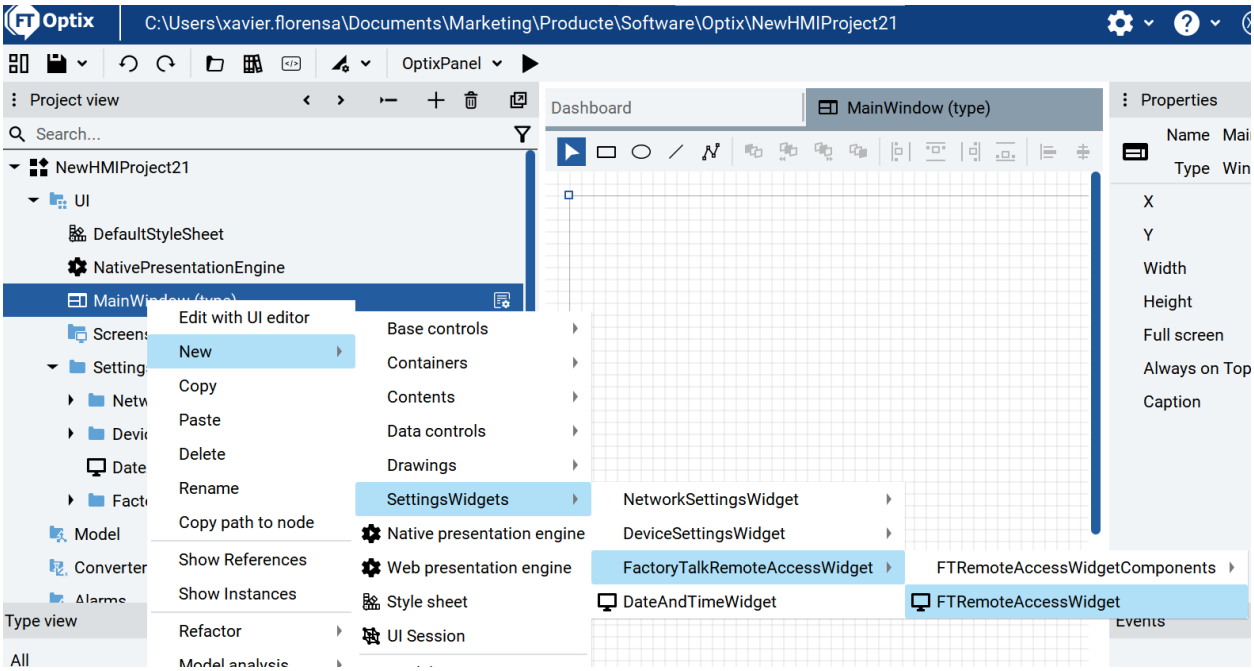
You have to create the new project as an OptixPanel default



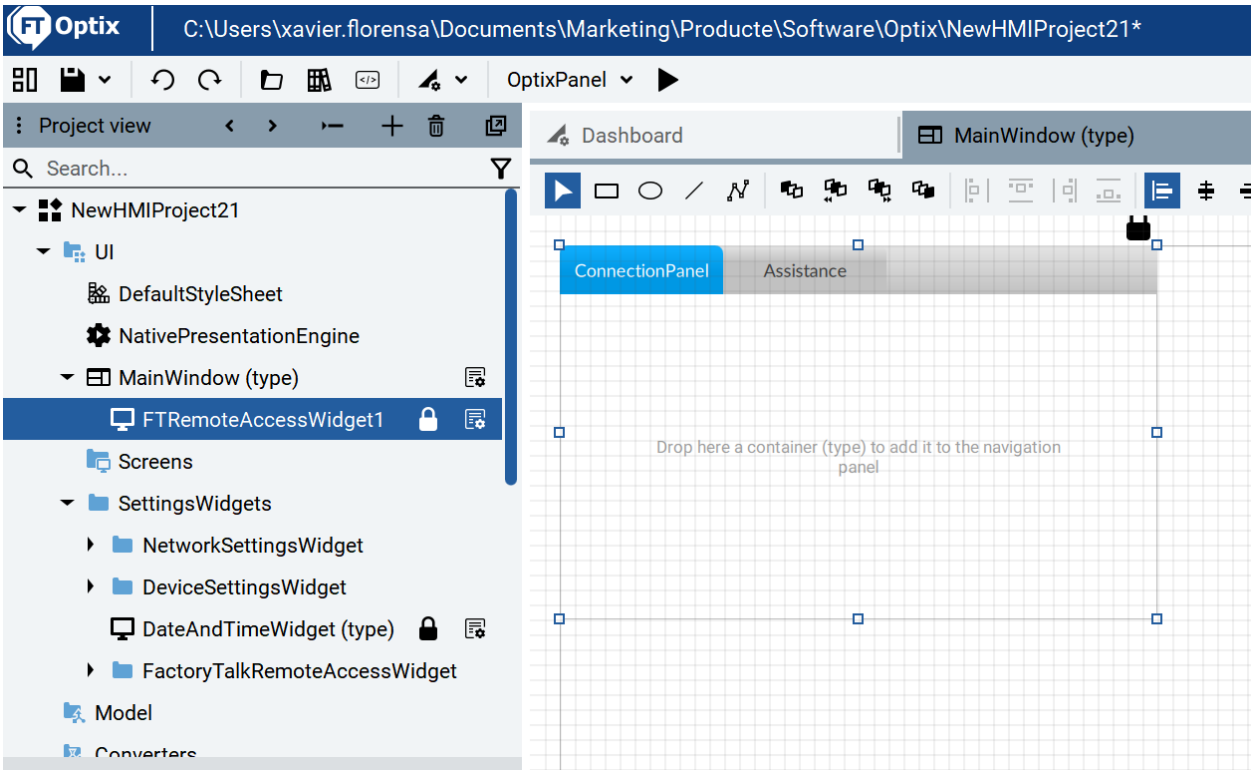
This will populate with the needed libraries



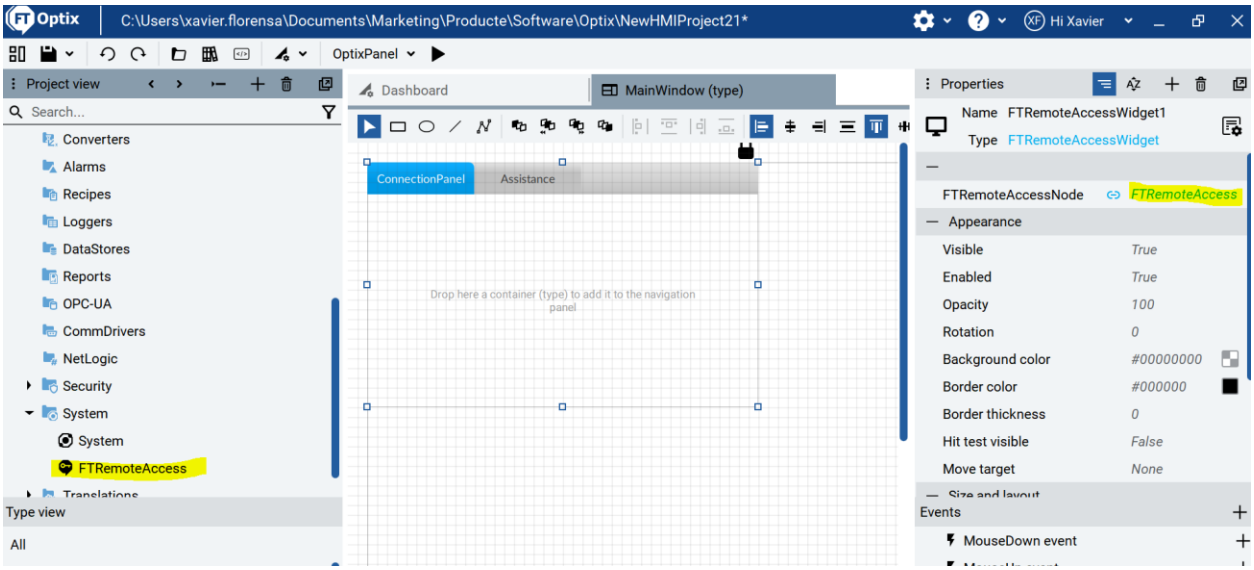
Then add a new instance of any of these widgets.
For instance Remote Access



You will see this



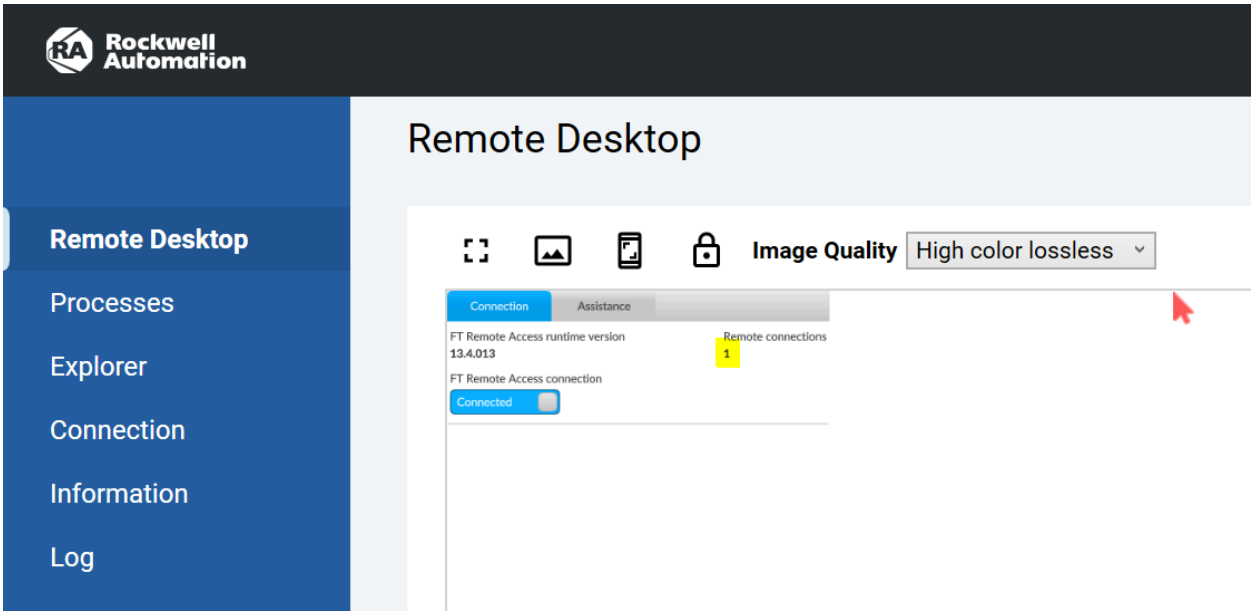
Be aware that you have a property already selected. You do not have to select this.



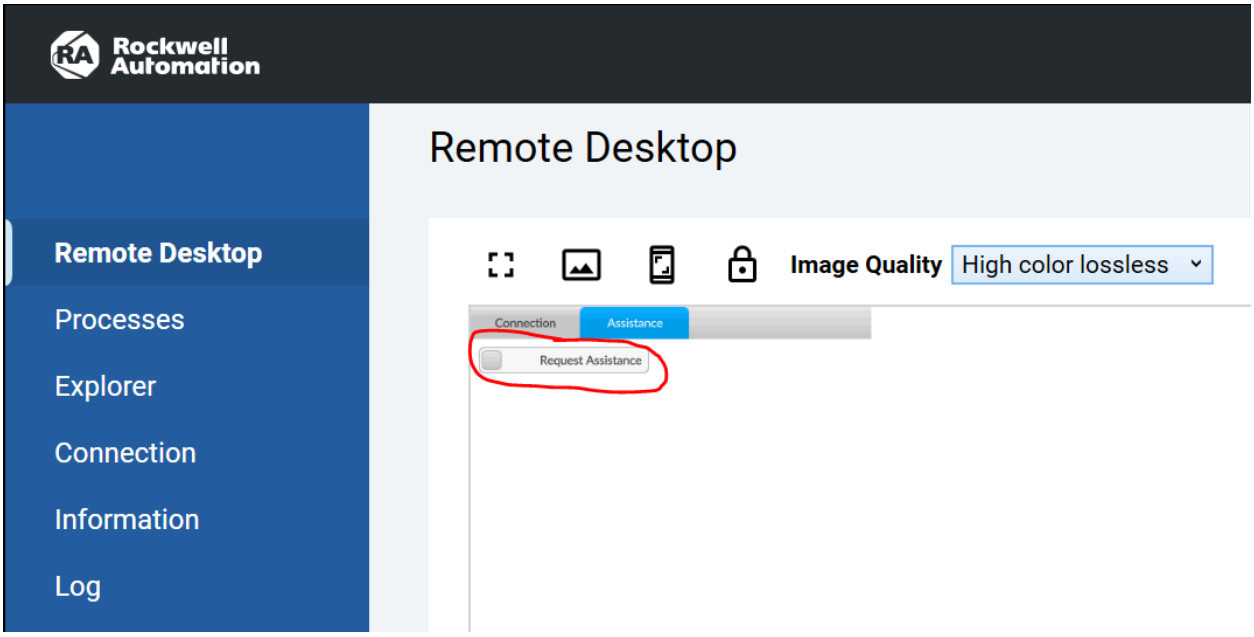
If you are on a Emulator default project, you will not see this property filled up and no system folder.

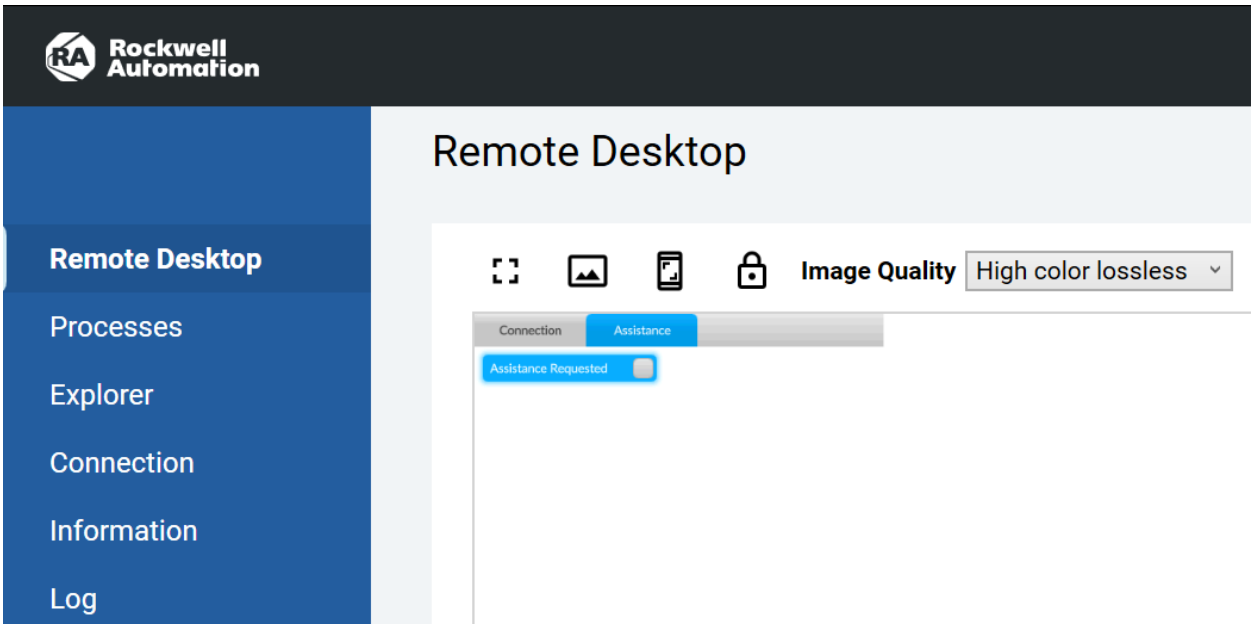
You can download the application to the Optix Panel.

Then, after you enter to the interactive desktop on FT Hub / FT Remote Access, you will see a new connectrion on the machine side

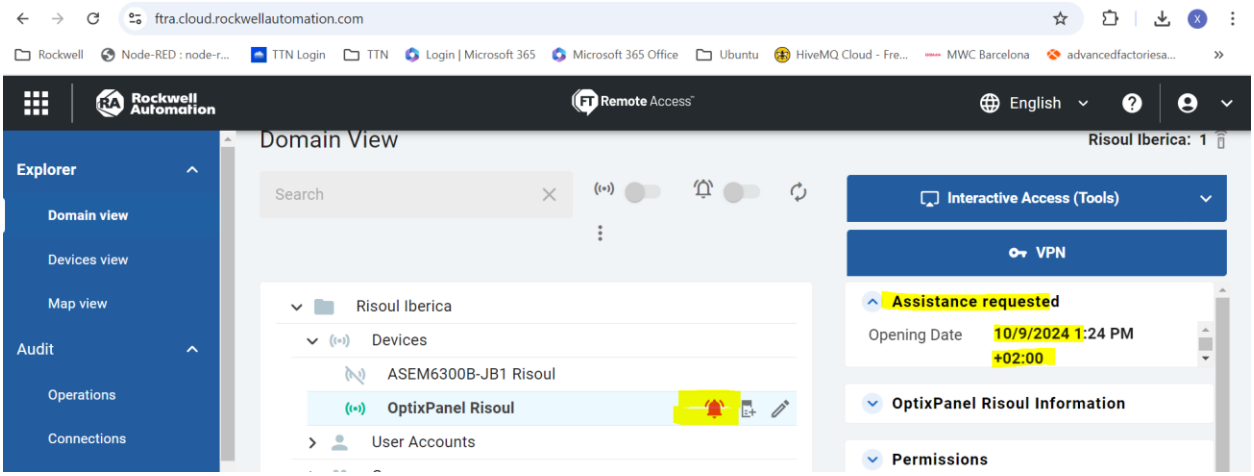


And you can request assistance by pushing the button

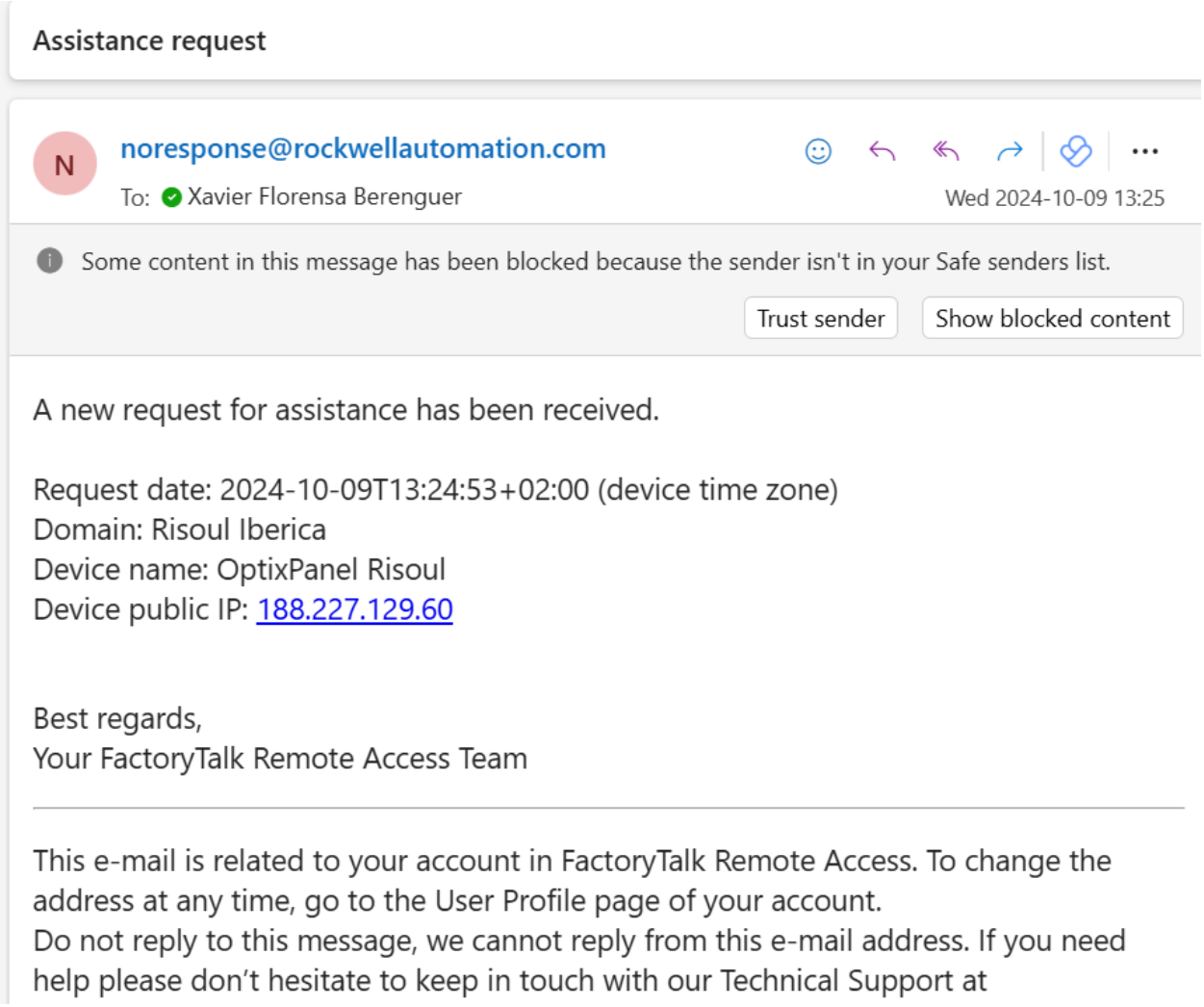




You will see the red sign on the dashboard



You will also receive an e-mail if properly configured



Assistance request

N noresponse@rockwellautomation.com

To: Xavier Florensa Berenguer

Wed 2024-10-09 13:25

Some content in this message has been blocked because the sender isn't in your Safe senders list.

Trust sender | Show blocked content

A new request for assistance has been received.

Request date: 2024-10-09T13:24:53+02:00 (device time zone)
Domain: Risoul Iberica
Device name: OptixPanel Risoul
Device public IP: 188.227.129.60

Best regards,
Your FactoryTalk Remote Access Team

This e-mail is related to your account in FactoryTalk Remote Access. To change the address at any time, go to the User Profile page of your account.
Do not reply to this message, we cannot reply from this e-mail address. If you need help please don't hesitate to keep in touch with our Technical Support at

Or even with user contact details

Assistance request

N noresponse@rockwellautomation.com

To: Xavier Florensa Berenguer

Wed 2024-10-09 13:40

Some content in this message has been blocked because the sender isn't in your Safe senders list.

Trust sender | Show blocked content

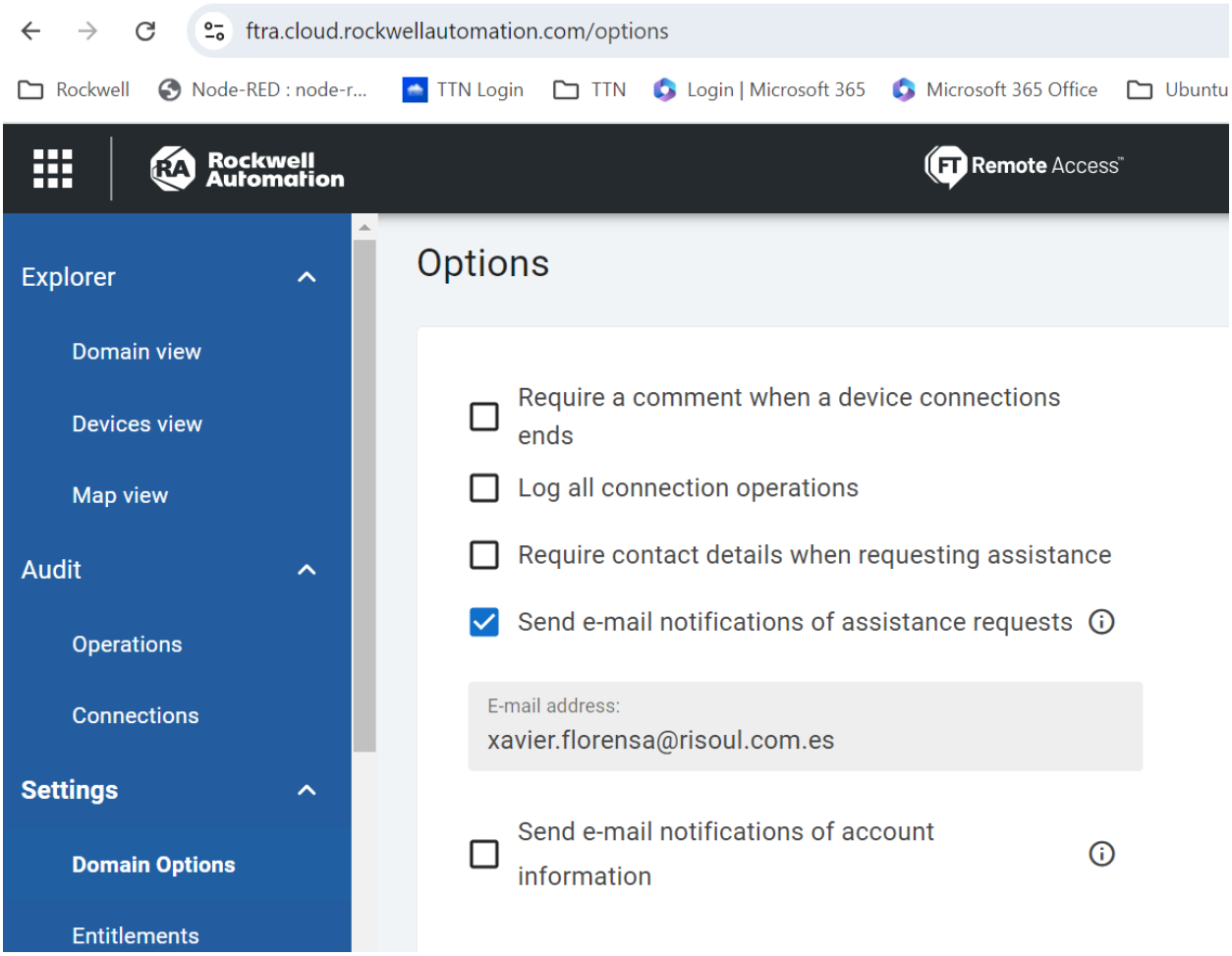
A new request for assistance has been received.

Request date: 2024-10-09T13:40:09+02:00 (device time zone)
Domain: Risoul Iberica
Device name: OptixPanel Risoul
Device public IP: 188.227.129.60

Requester name: Xavier
Contact information: Risoul
Problem description:
Sofware

Best regards,
Your FactoryTalk Remote Access Team

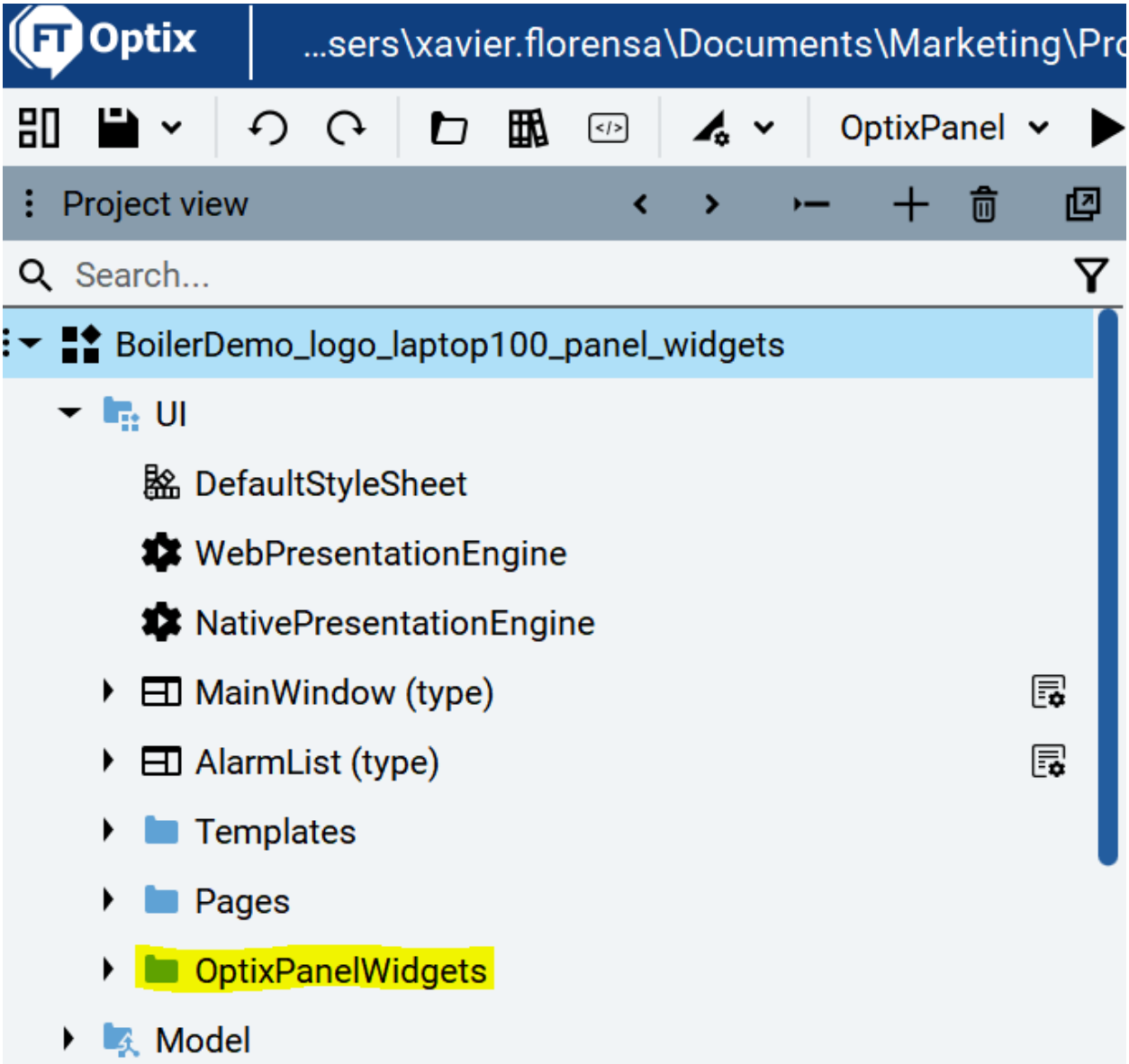
How to configure e-mail sending



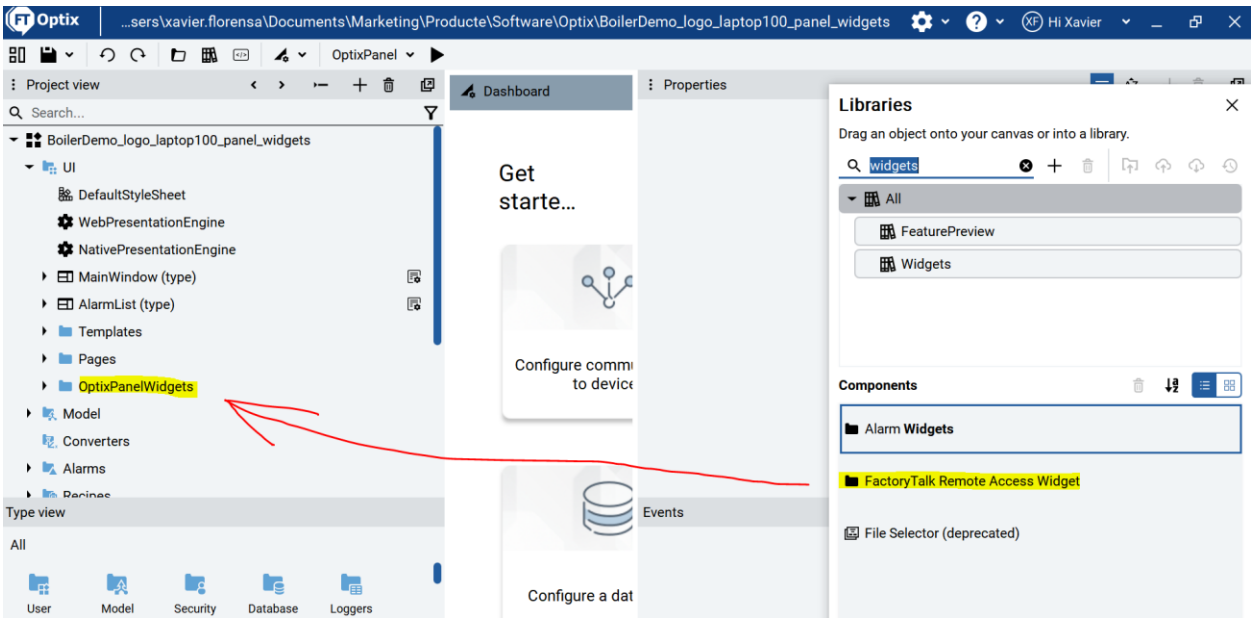
11.2. Adding OptixPanel Widgets to a standard application

If you start your application as default, then even thought you install the OptixPanel Widgets libraries, those will not work on runtime.

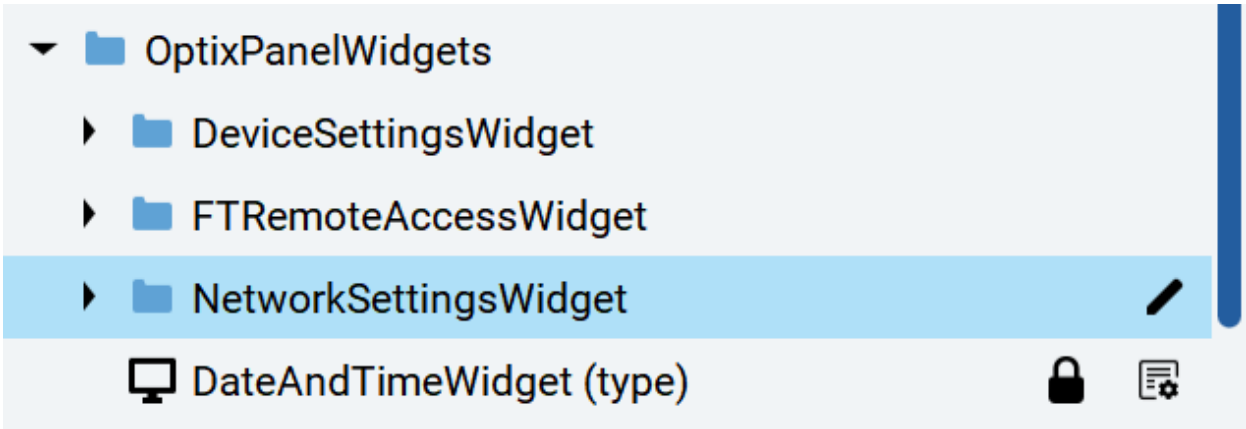
But you can do it this way
Create a new folder and name it OptisPanelWidgets for instance.



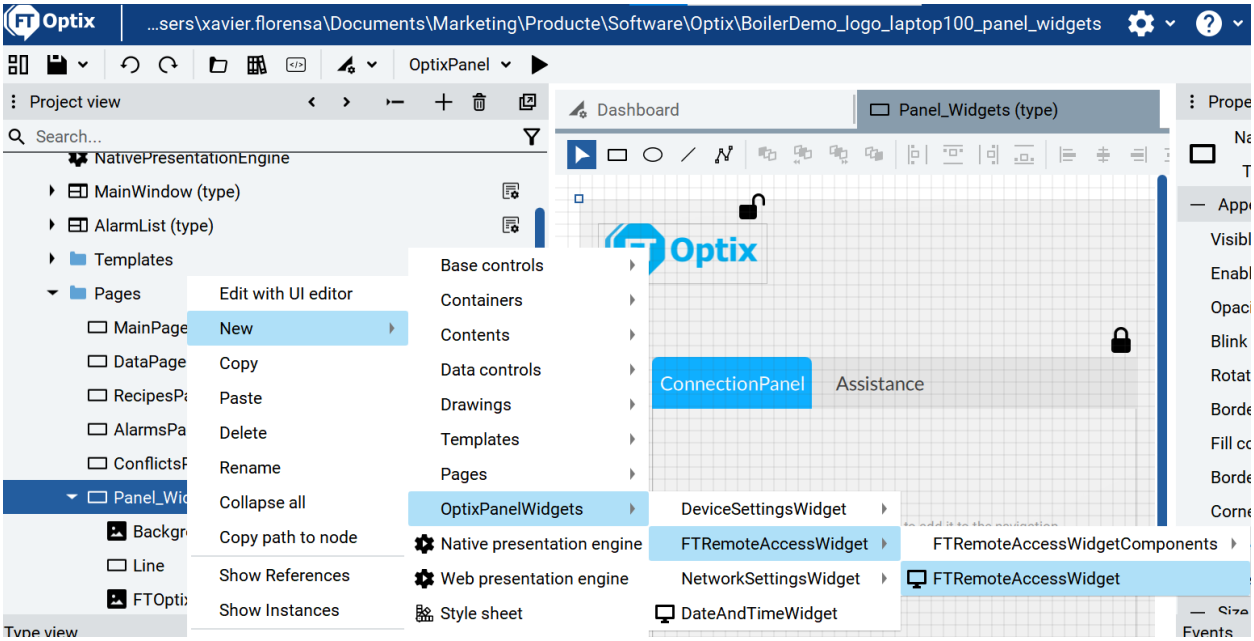
Look for Widgets on the library
Drag and drop to the new folder



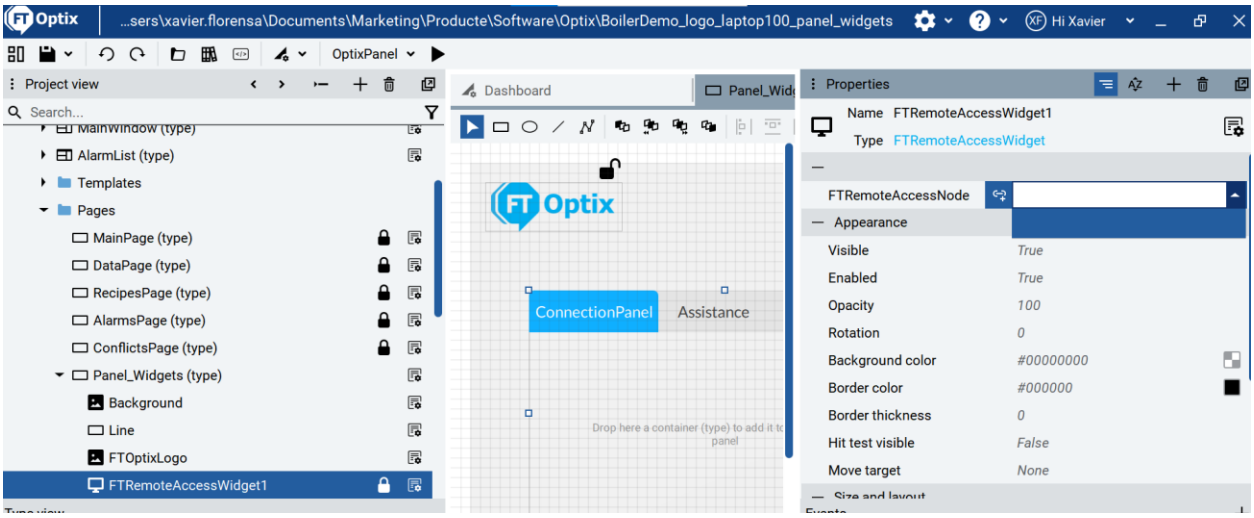
Now you will see all the widgets on your folder



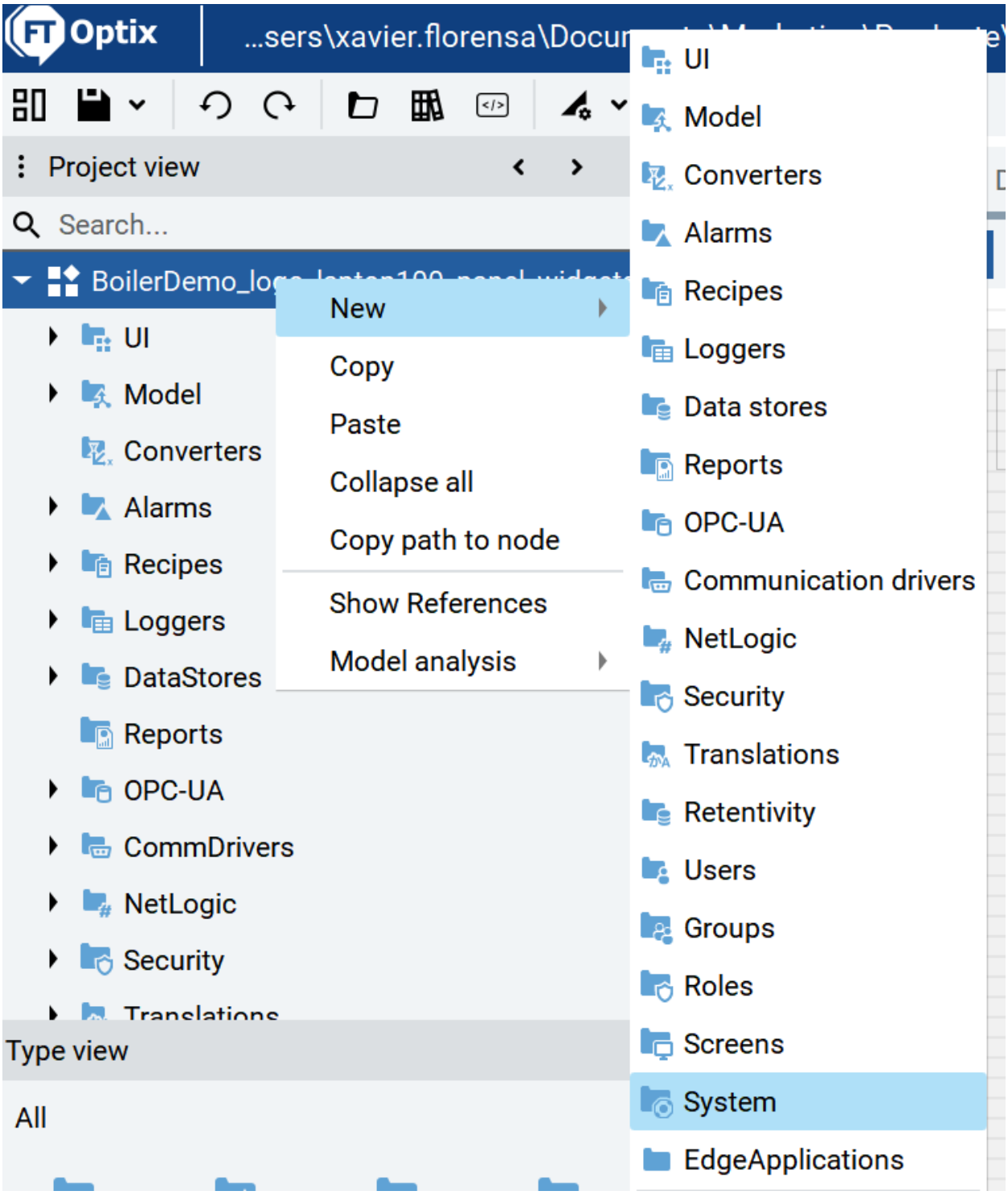
You can add a widget instance to your main page or to your panels list



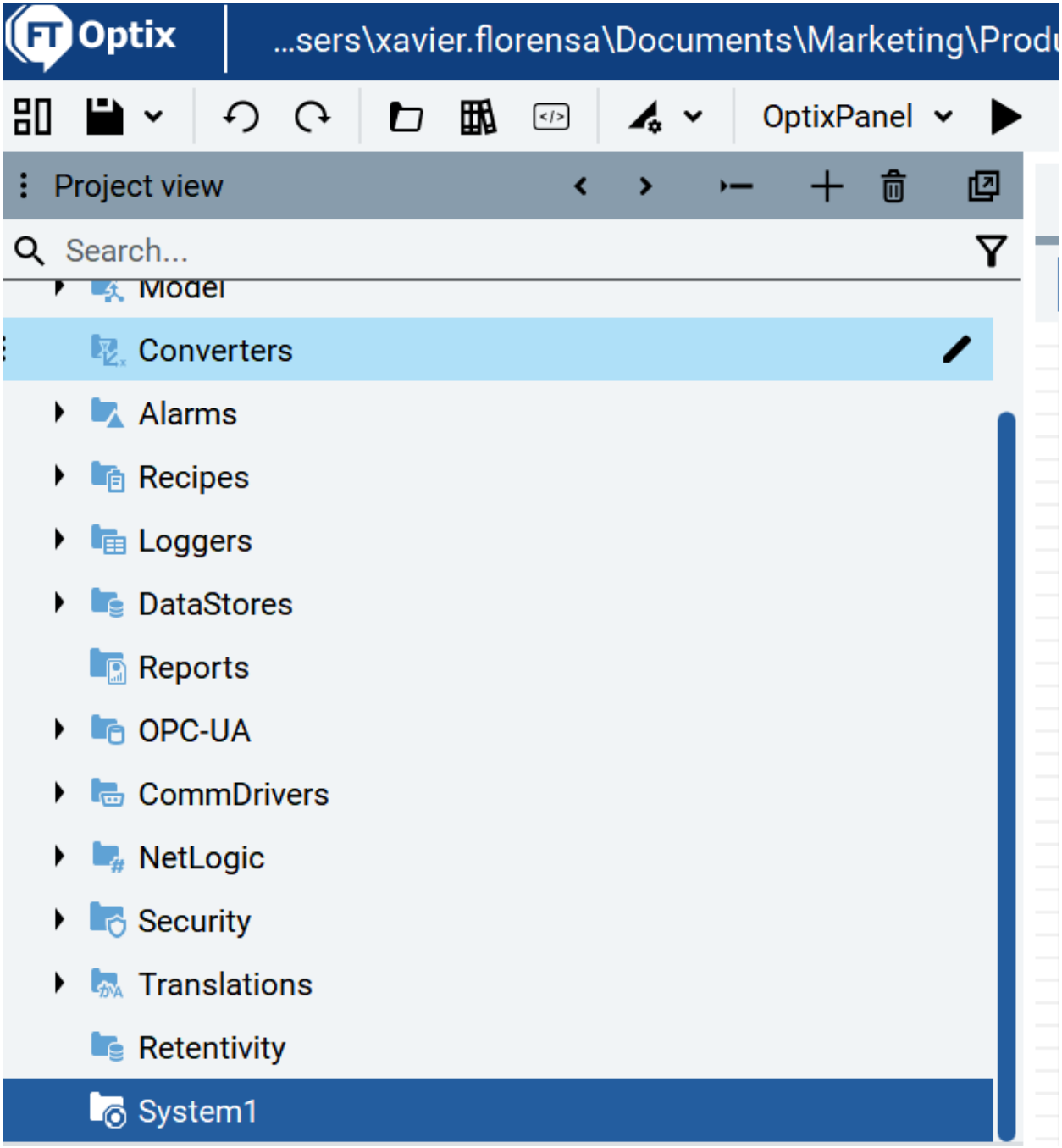
But this will not work since it has no reference or node to point to



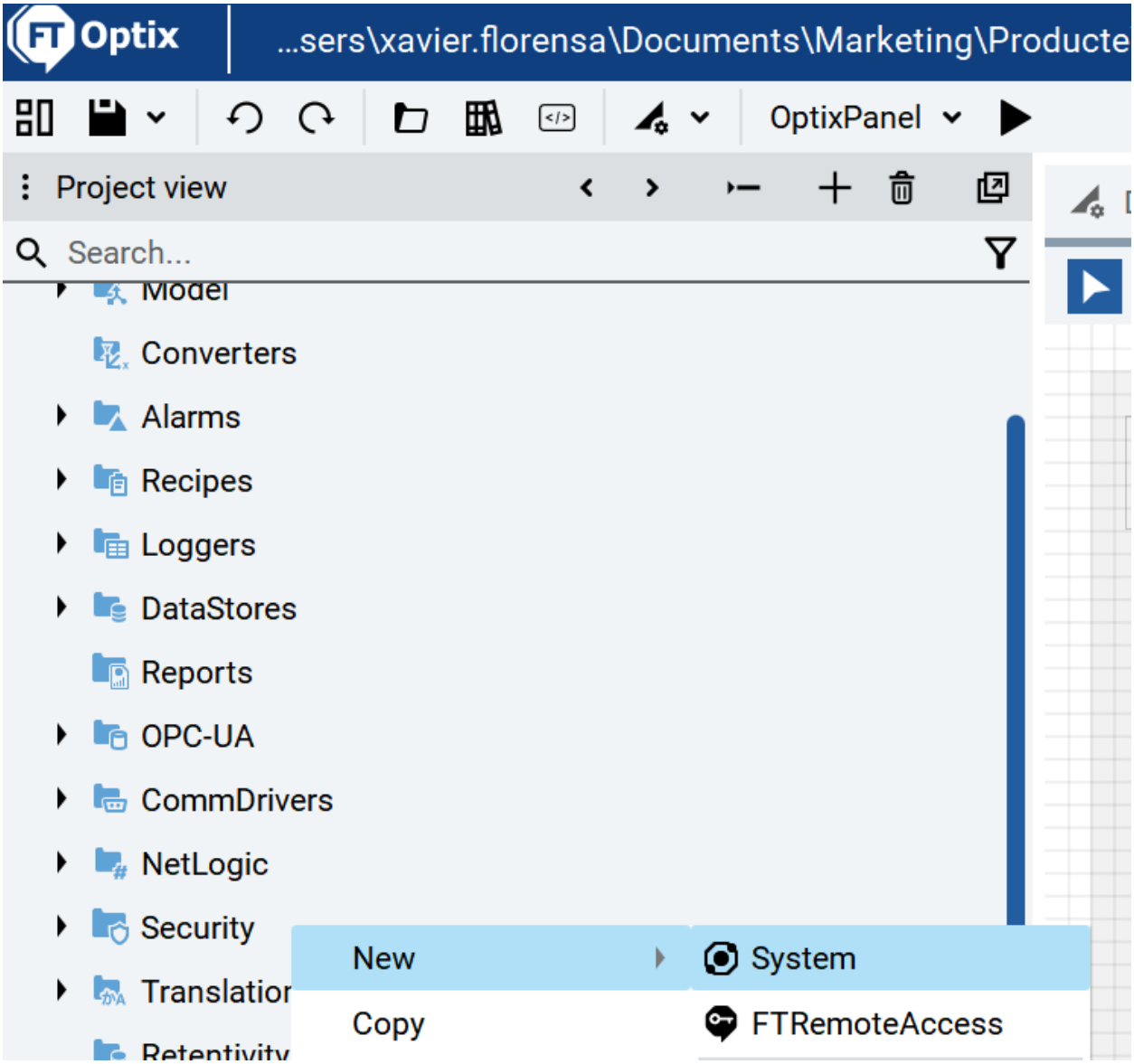
Now this is how you can solve this.
Add a new System object in your project



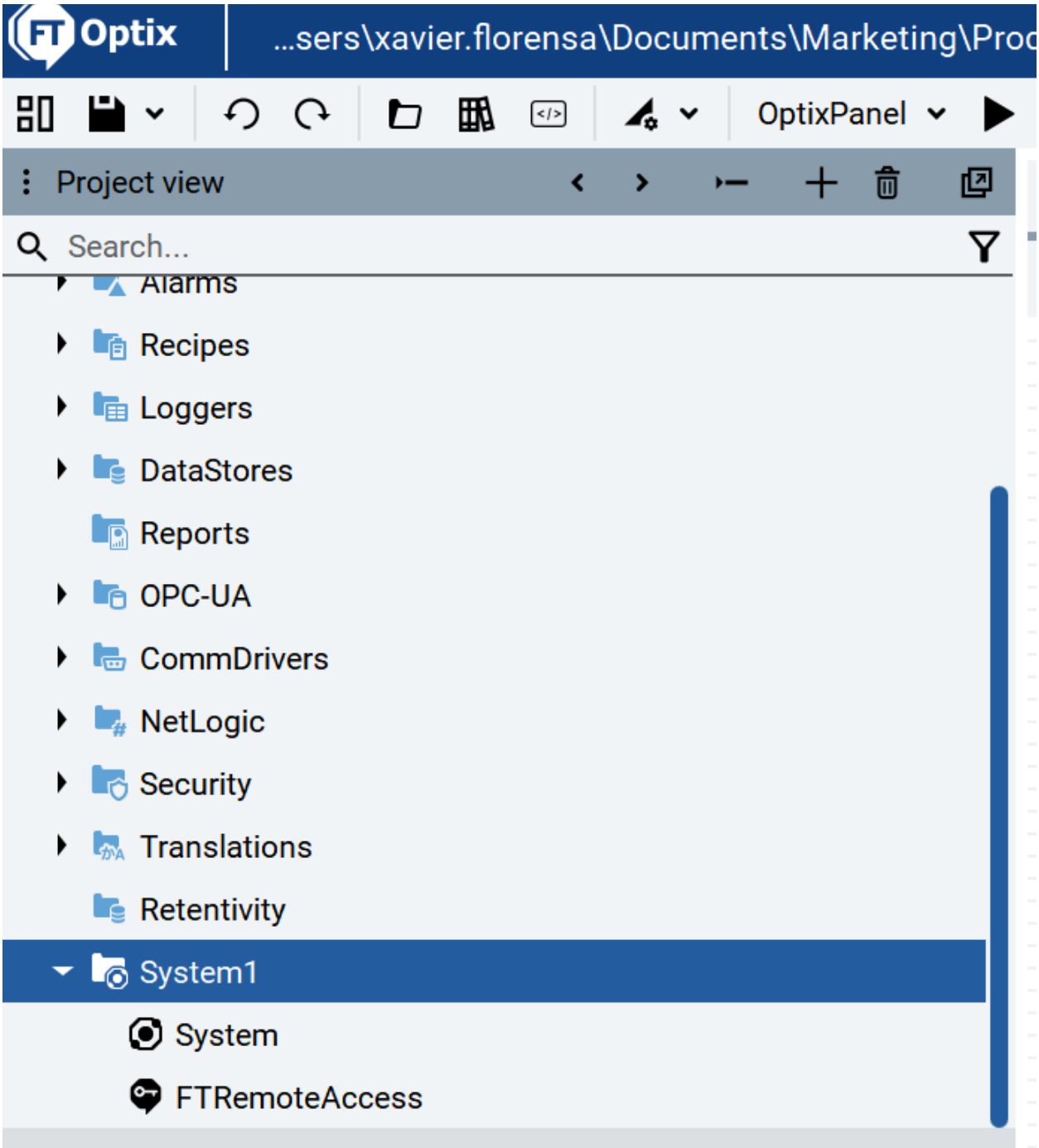
You will see this empty system



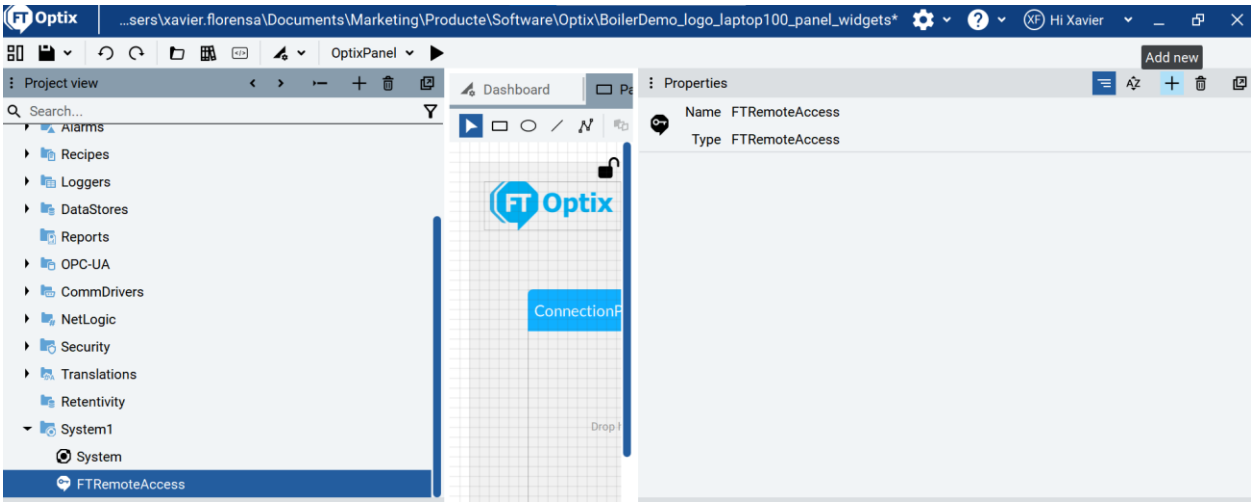
Add two net items to your new System object
A System and a Remote Acess

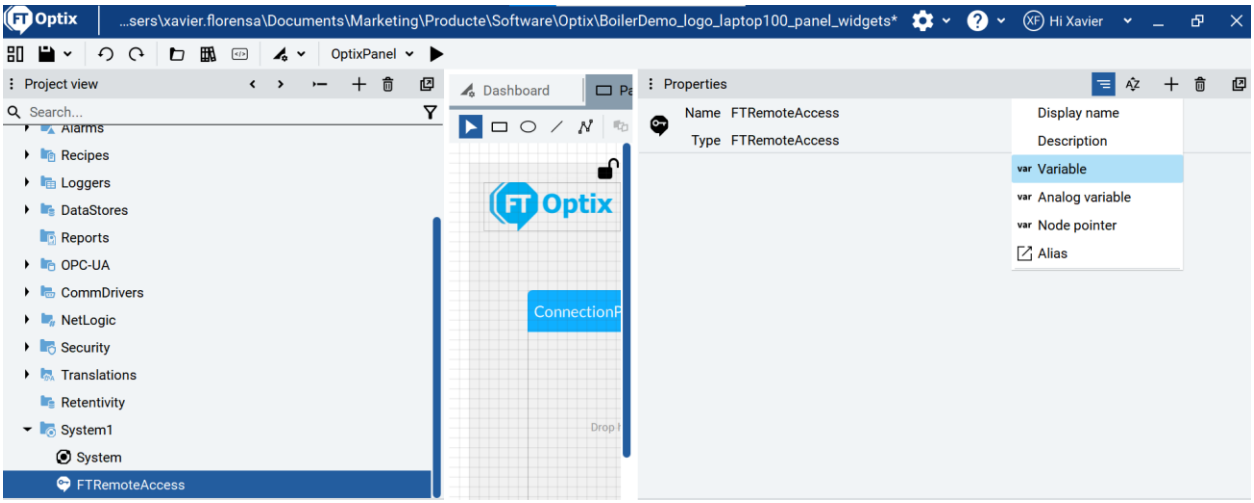


Now you will see your project tree this way

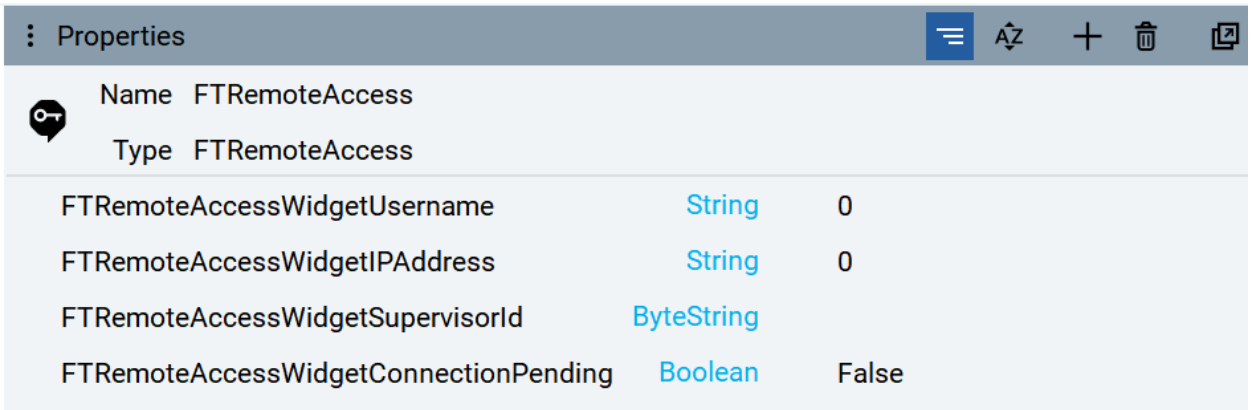


Click on FTRemoteAcess and create new variables

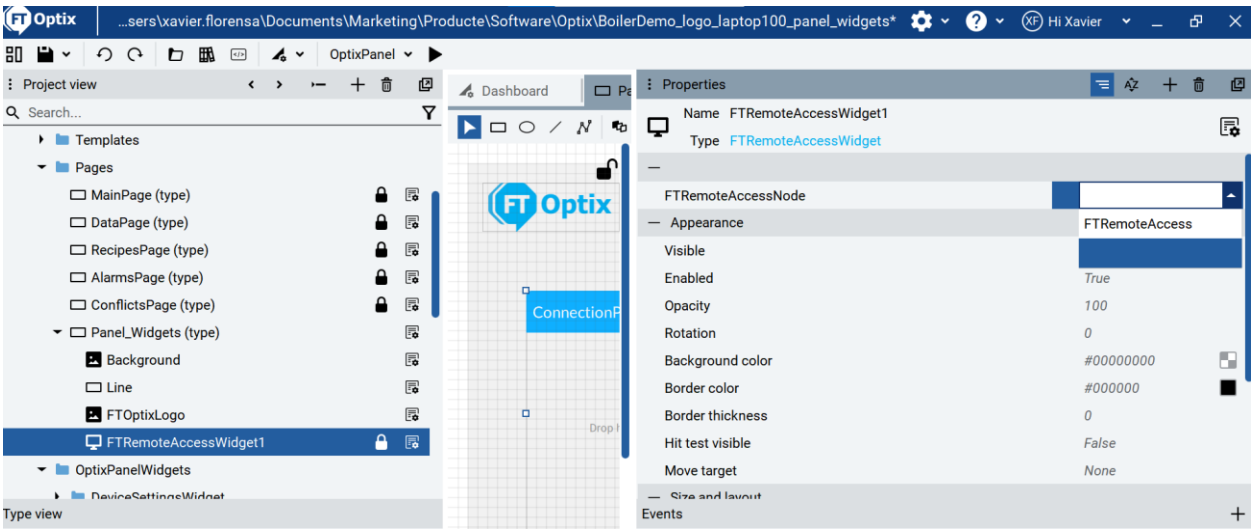


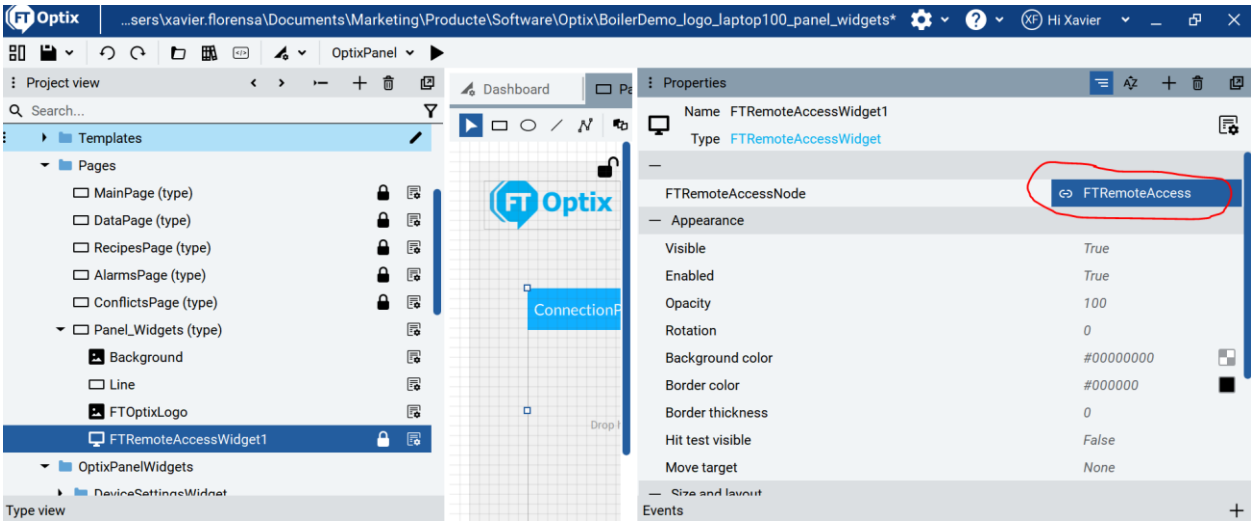


Create such new variables on Properties

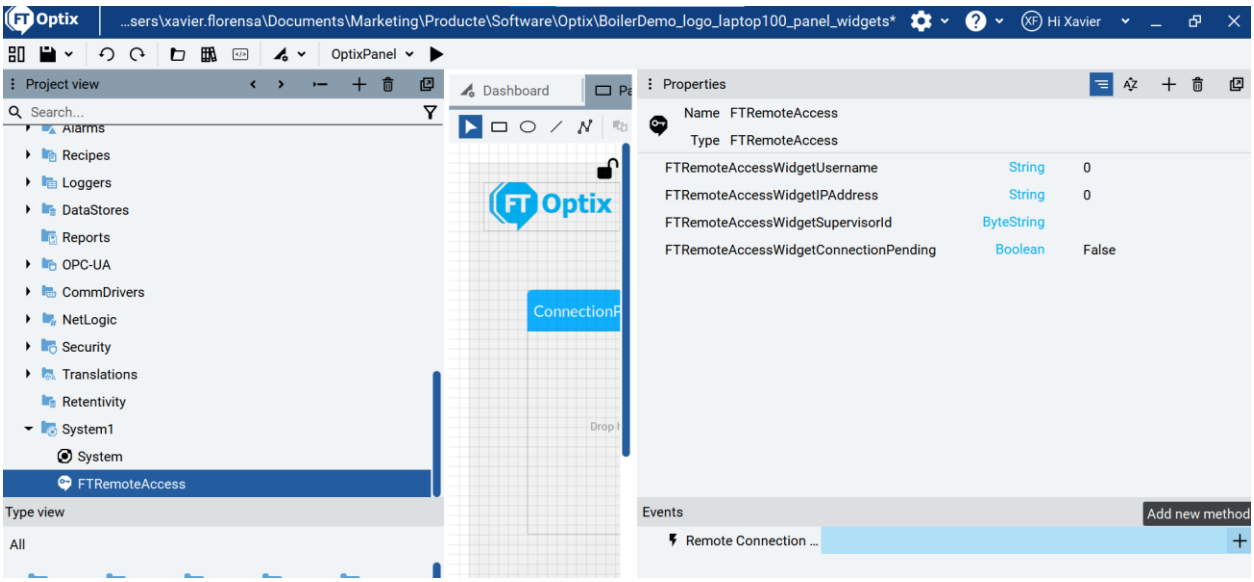


Now you can go to your Widget instance, click on it and add the missing node that you will find unfolding node property





Now let's go to the System Object / Remote Access properties and add some methods on remote connection event



Add some new “Variable set” commands methods this way (see next paragraph for a step by step guide)

Events

Remote Connection ...

Method /Root/Objects/Commands/VariableCommands/Set

Input arguments

VariableToModify FTRemoteAccessWidgetUsername

Value String ../../../../EventArguments/Username

ArrayIndex 0

Method /Root/Objects/Commands/VariableCommands/Set

Input arguments

VariableToModify FTRemoteAccessWidgetIPAddress

Value String ../../../../EventArguments/IpAddress

ArrayIndex 0

Method /Root/Objects/Commands/VariableCommands/Set

Input arguments

VariableToModify FTRemoteAccessWidgetSupervisorId

Value ByteString ../../../../EventArguments/SupervisorId

ArrayIndex 0

Method /Root/Objects/Commands/VariableCommands/Set

Input arguments

VariableToModify FTRemoteAccessWidgetConnectionPending

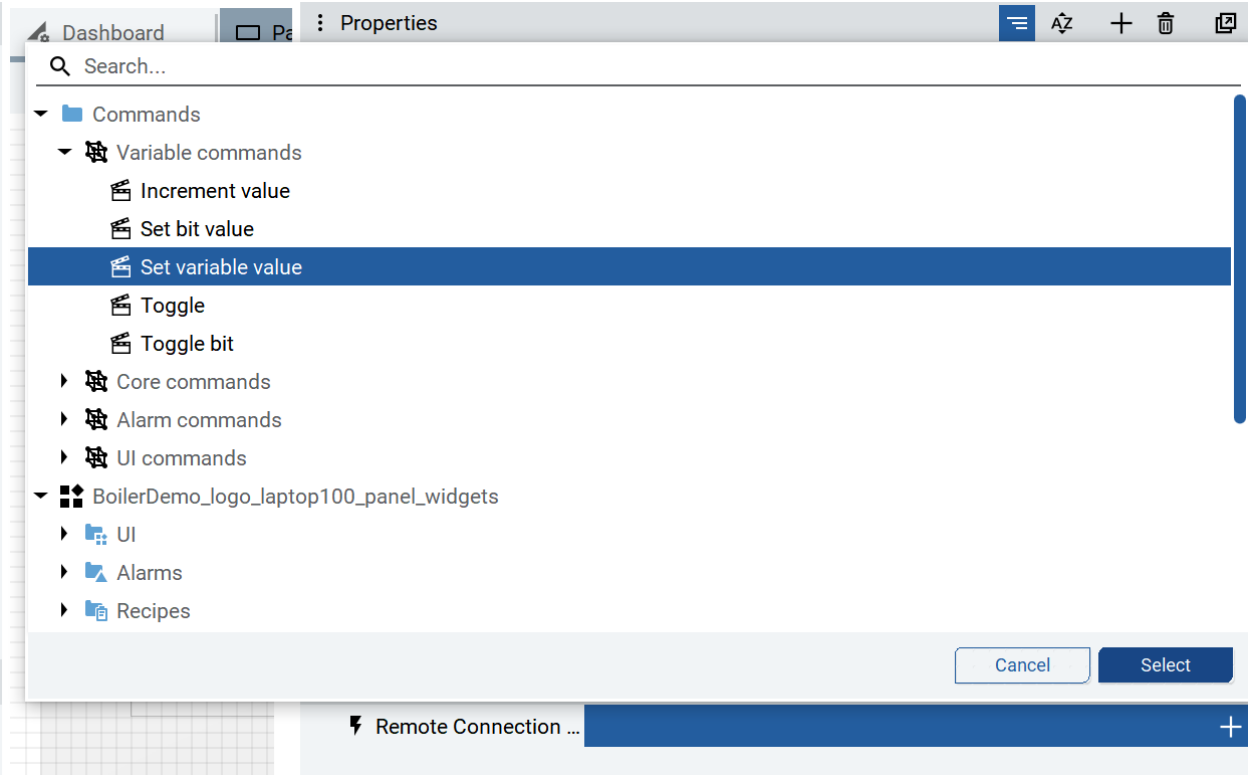
Value Boolean True

Start this way

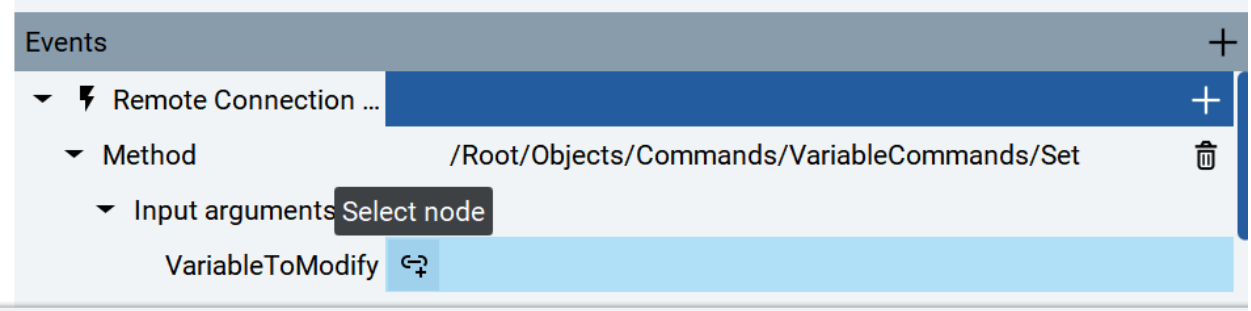
Events

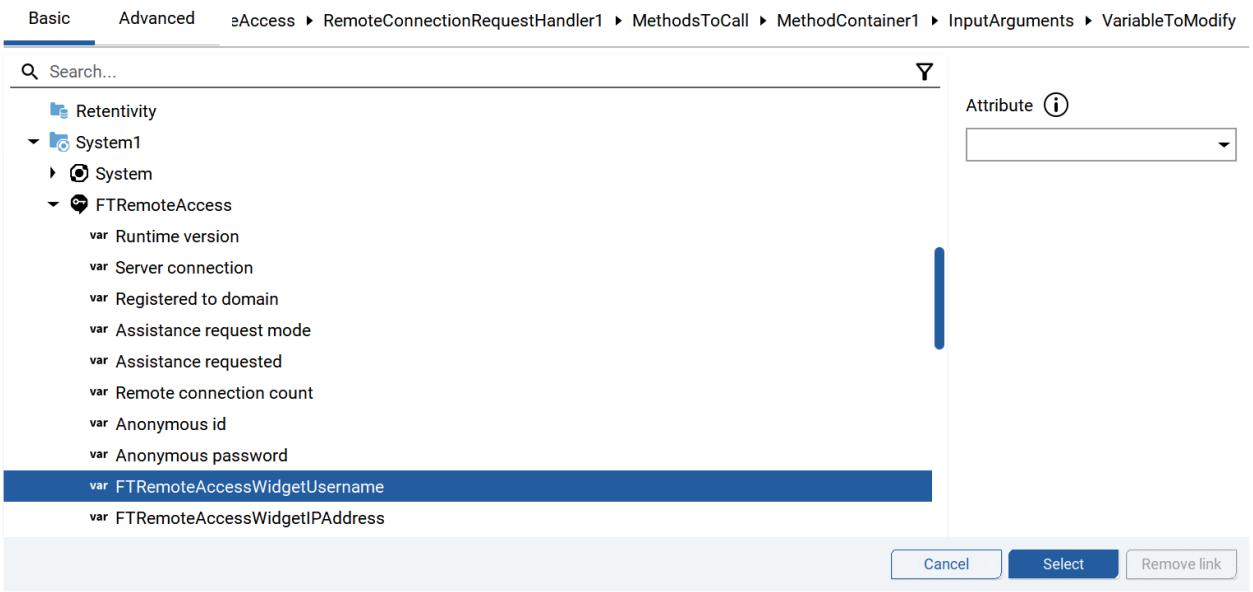
Add new method

Remote Connection ...

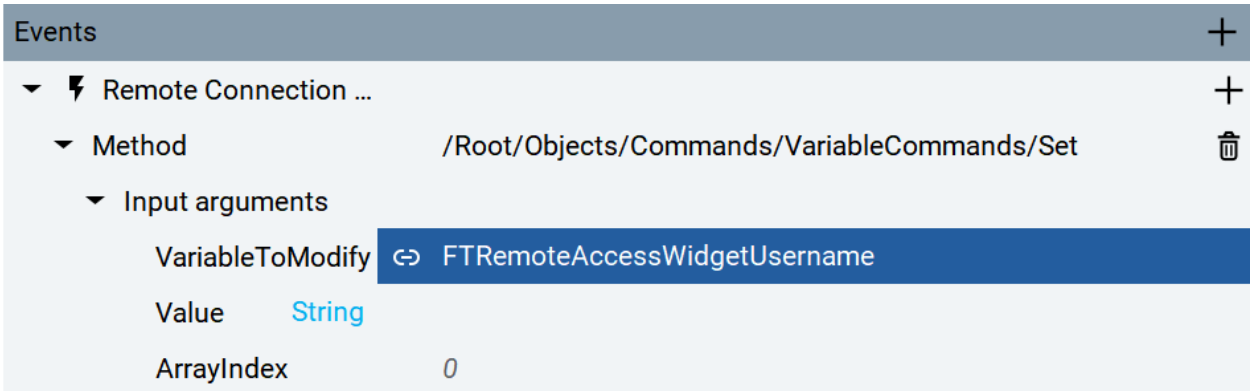


You will see this new fields, then select the variable to modify

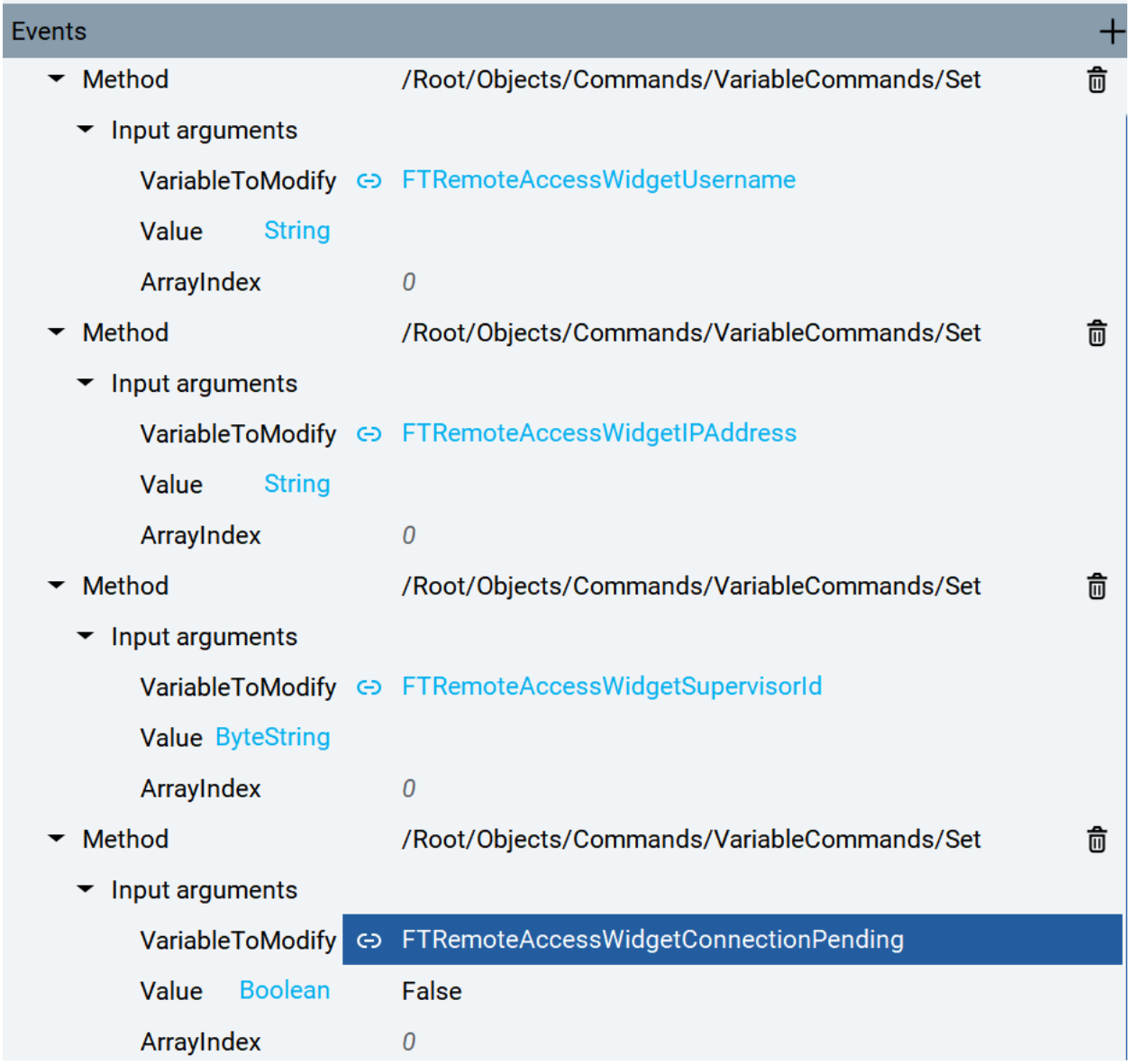




You will see this



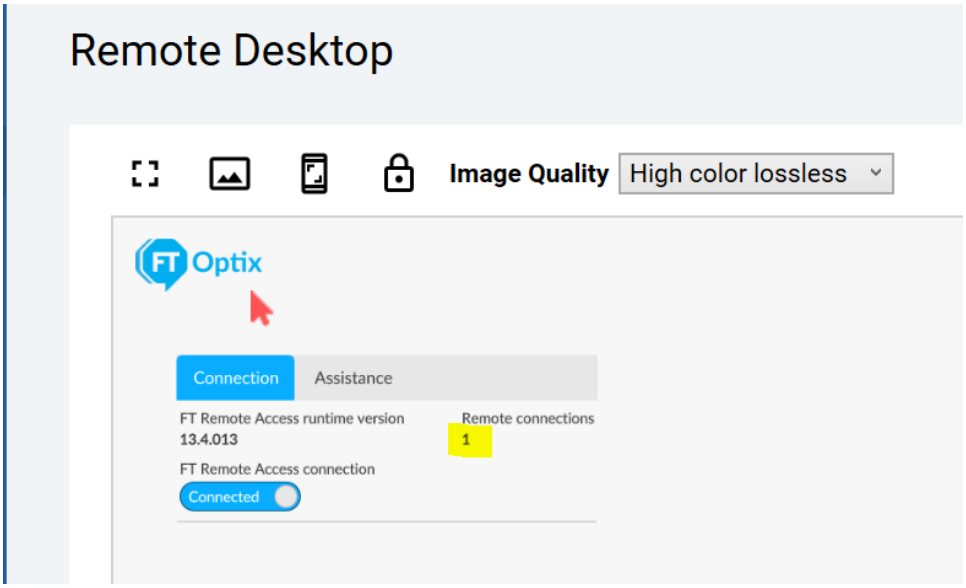
Complete until you have all these methods populated



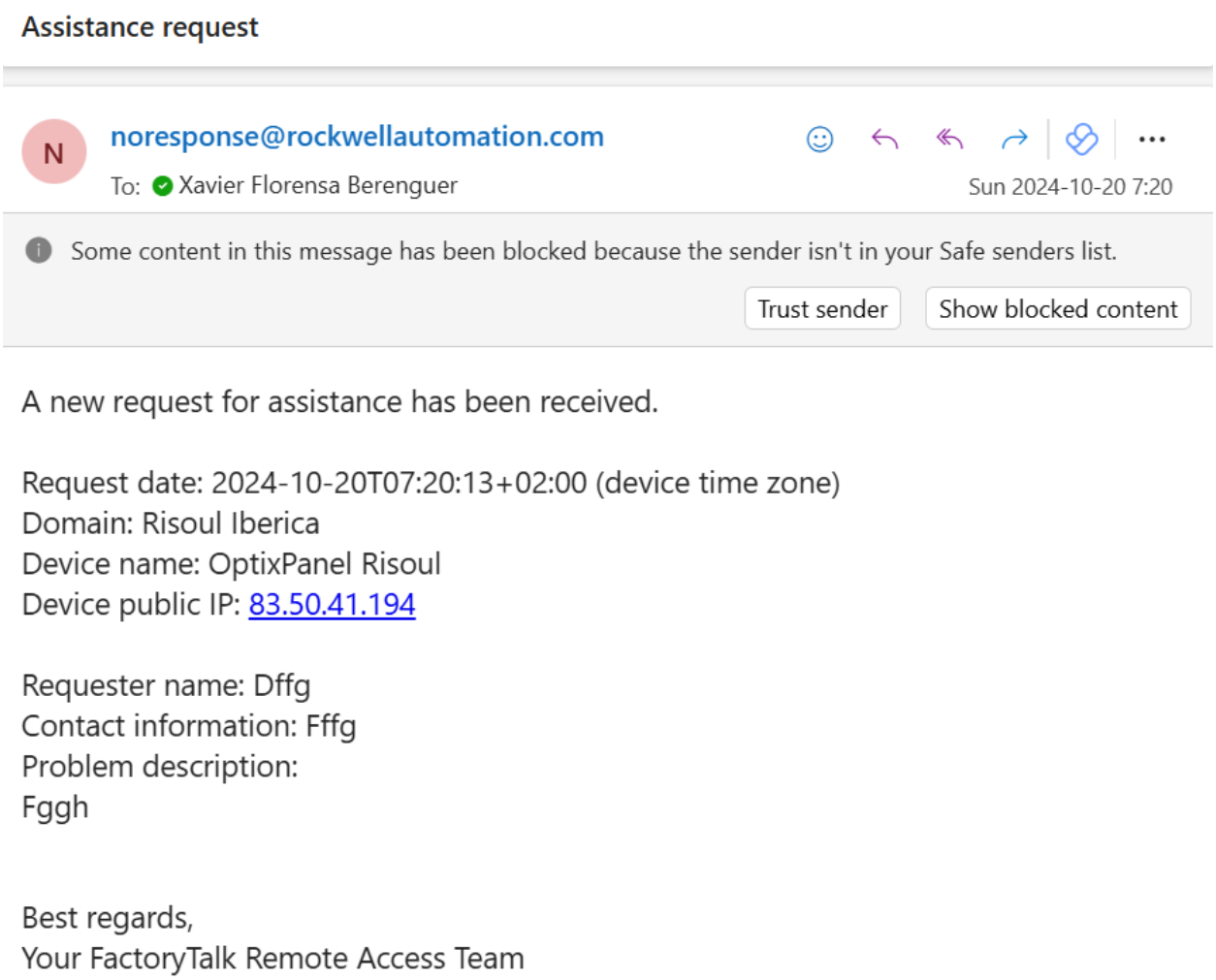
Start the new application on OptixPanel

Open FactoryTalk Hub / Remote Access and start a new interactive remote desktop

Voilà



And you can add for remote assistance



You can find the code here
https://github.com/xavierflorensa/FactoryTalkOptix_BoilerDemo_with_Panel_Widgets.git

Here you have a video with OptixPanel and Remote Access
https://www.youtube.com/watch?v=kDGCZ4npzLE&t=7s

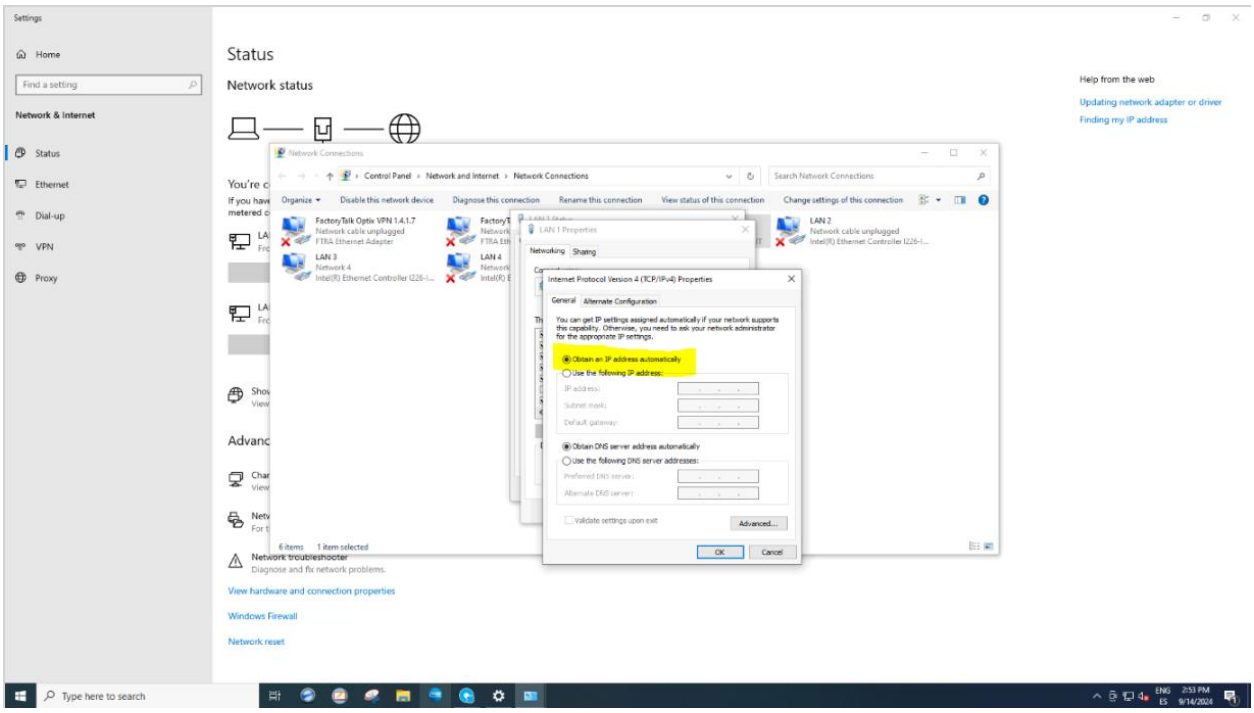
12. First steps with ASEM6300B PC and runtime

Here you have a video with ASEM6300 and Remote Access
https://www.youtube.com/watch?v=kDGCZ4npzLE&t=7s

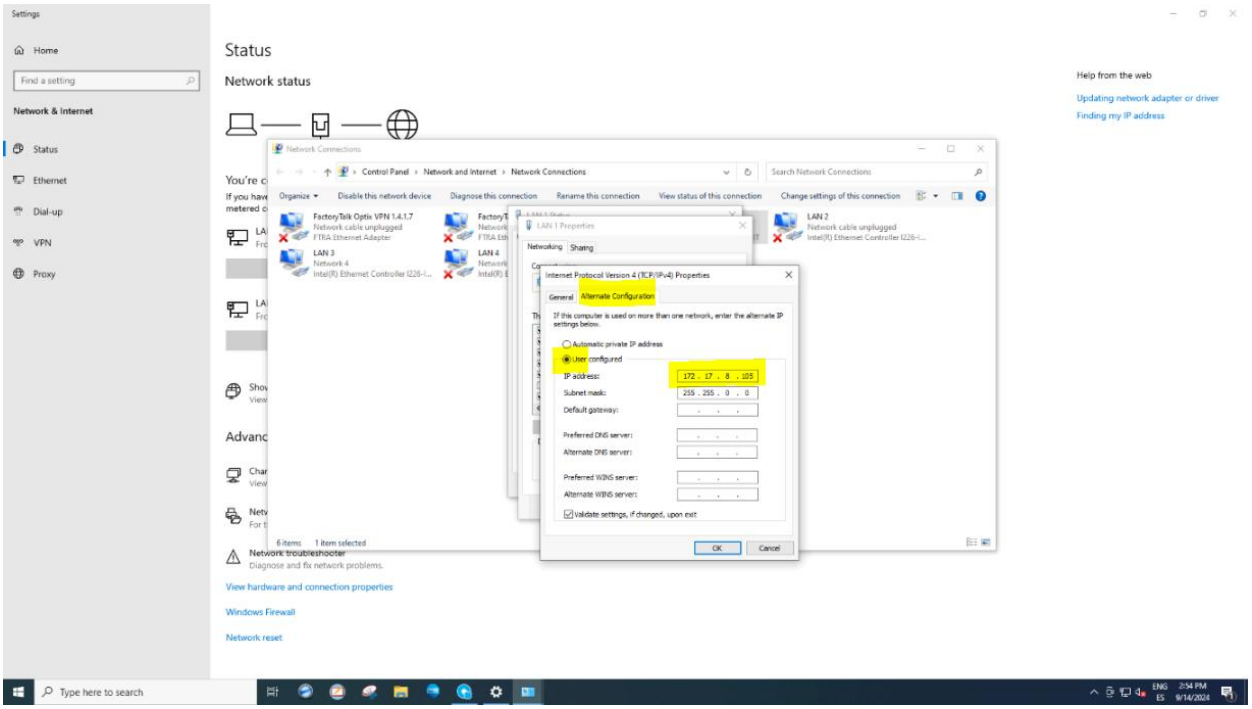
Install FactoryTalk Remote Access Tools on your ASEM6300B computer.
Give a fixed known IP address to your ASEM6300B computer.

It is recommended to have a DHCP IP address (to be able to connect to the internet to any router everywhere) together with a fixed IP (to connect directly with your Laptop in range for instance 172.17.8.103)
So Laptop fixed IP address 172.168.18.3

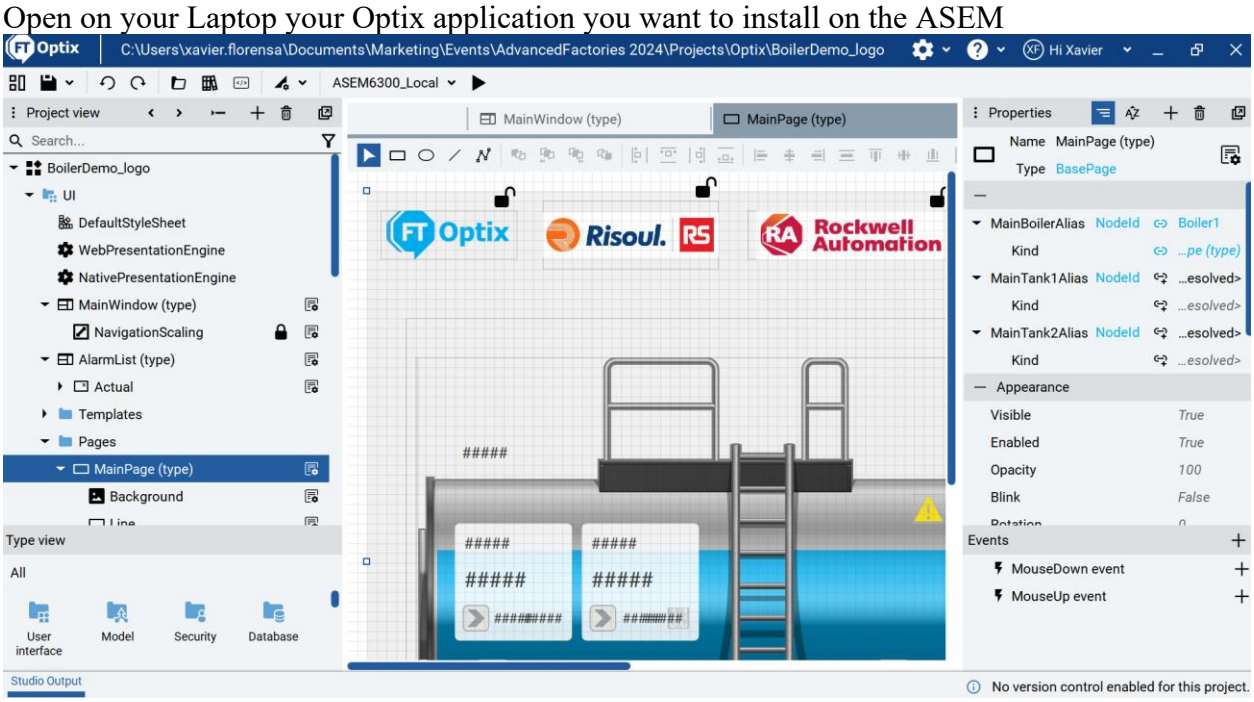
Set ASEM network adapter settings this way



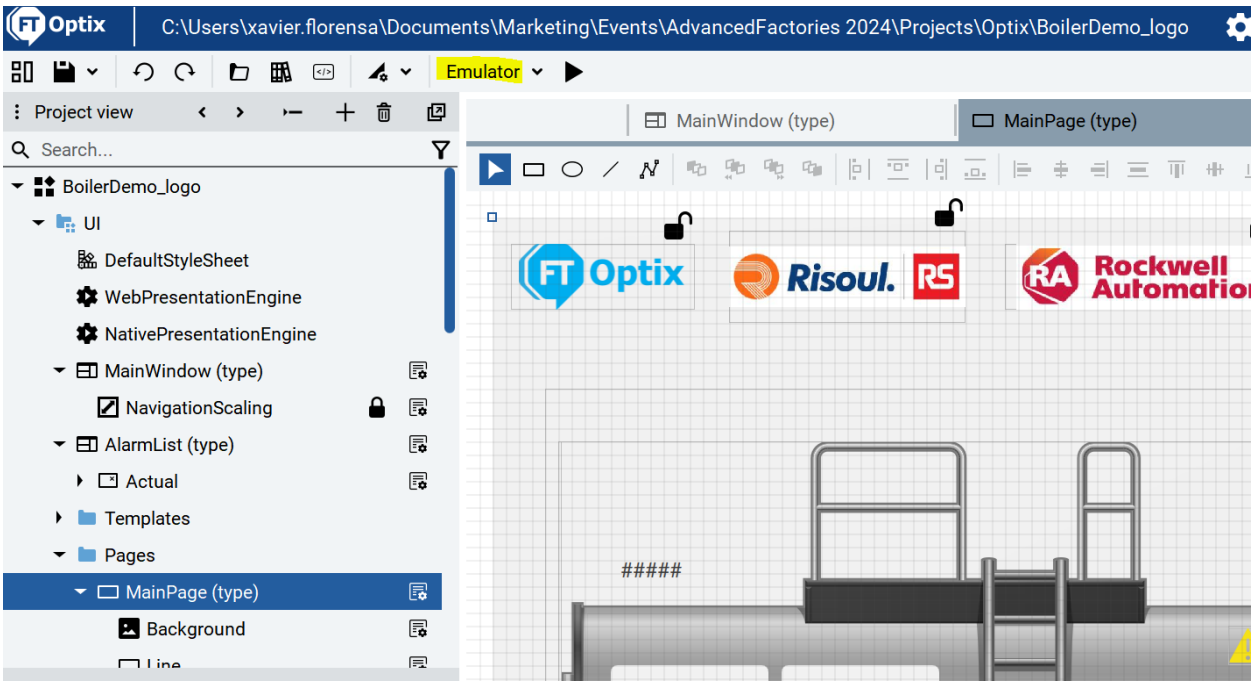
Then add a fixed alternate IP on ASEM 172.17.8.105



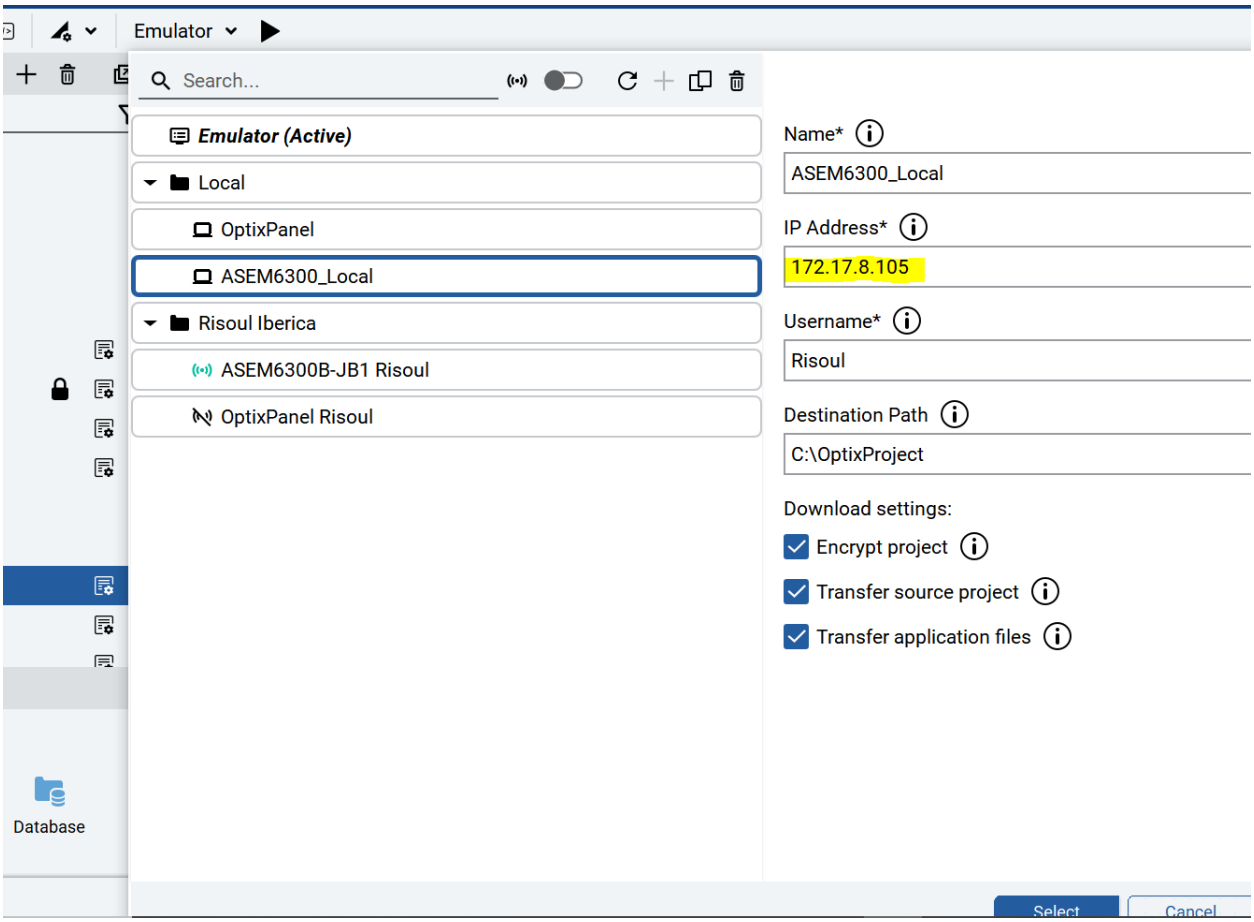
Open on your Laptop your Optix application you want to install on the ASEM



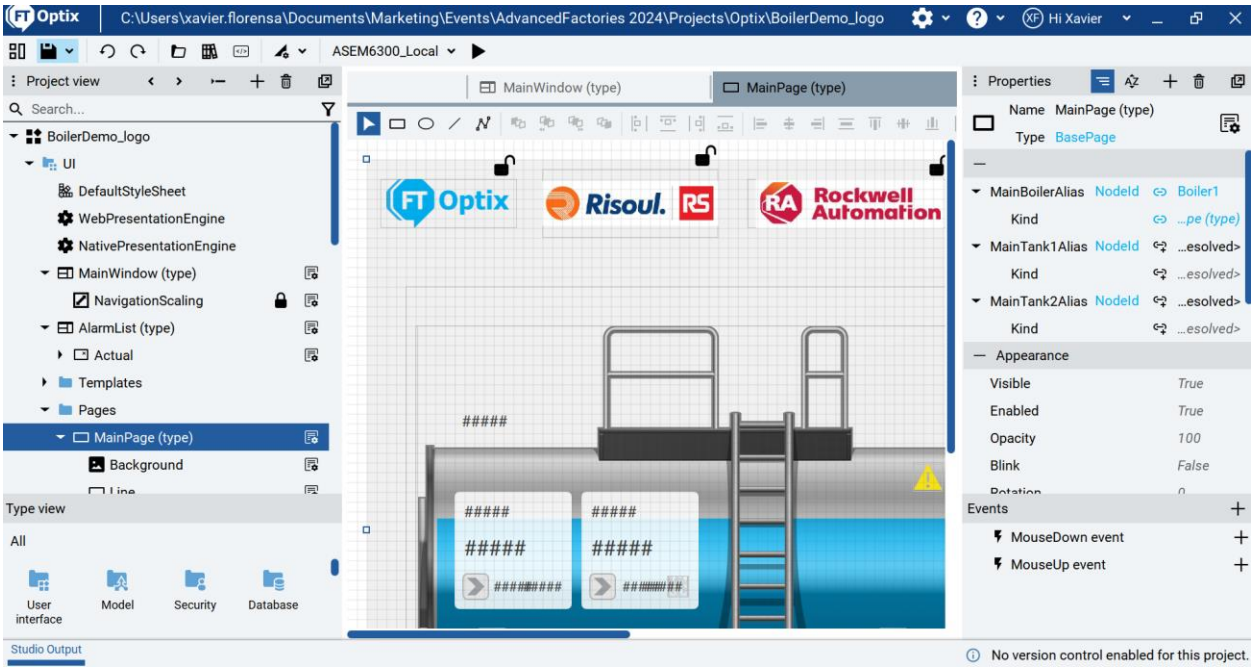
Click on the emulator (destination) pane

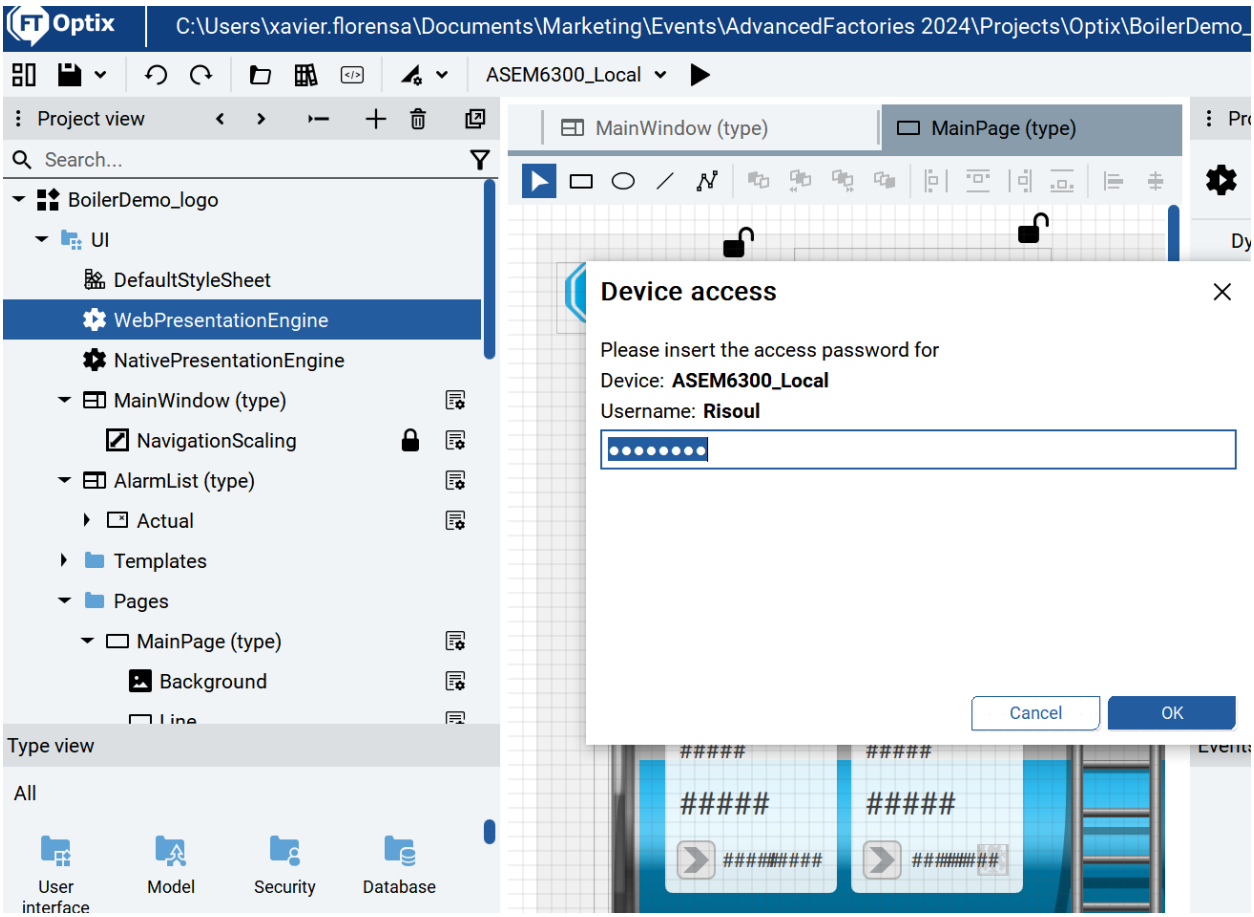


Add a new device this way
Giving the ASEM6300B fix IP address
Give the username
And a destination Directory (you have created previously on your ASEM) like for instance C:\OptixProject
Click on select



Then plug a patchcord between the ASEM6300 and the Laptop (both in same subnet)
Click on play

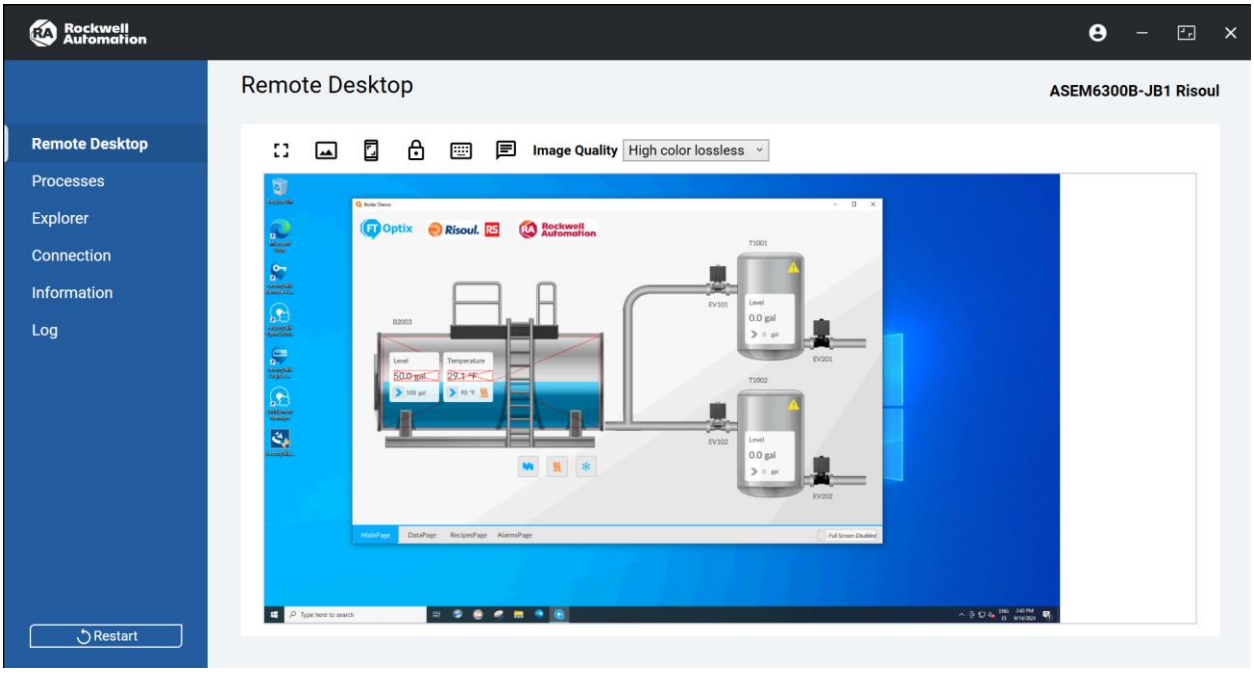




You will see a loading prompt

and the play symbol again.
But your application is already running on the ASEM

Voilà

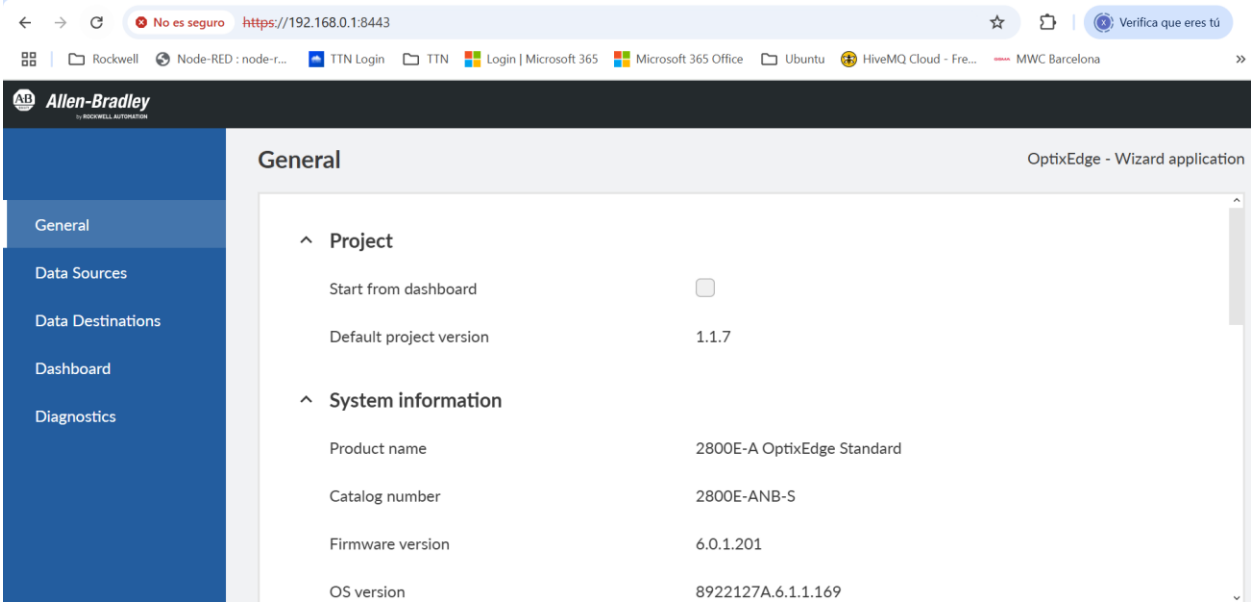


If there is a web access on the Optix application, you can open it from a web explorer provided you know the port

13. First steps with OptixEdge

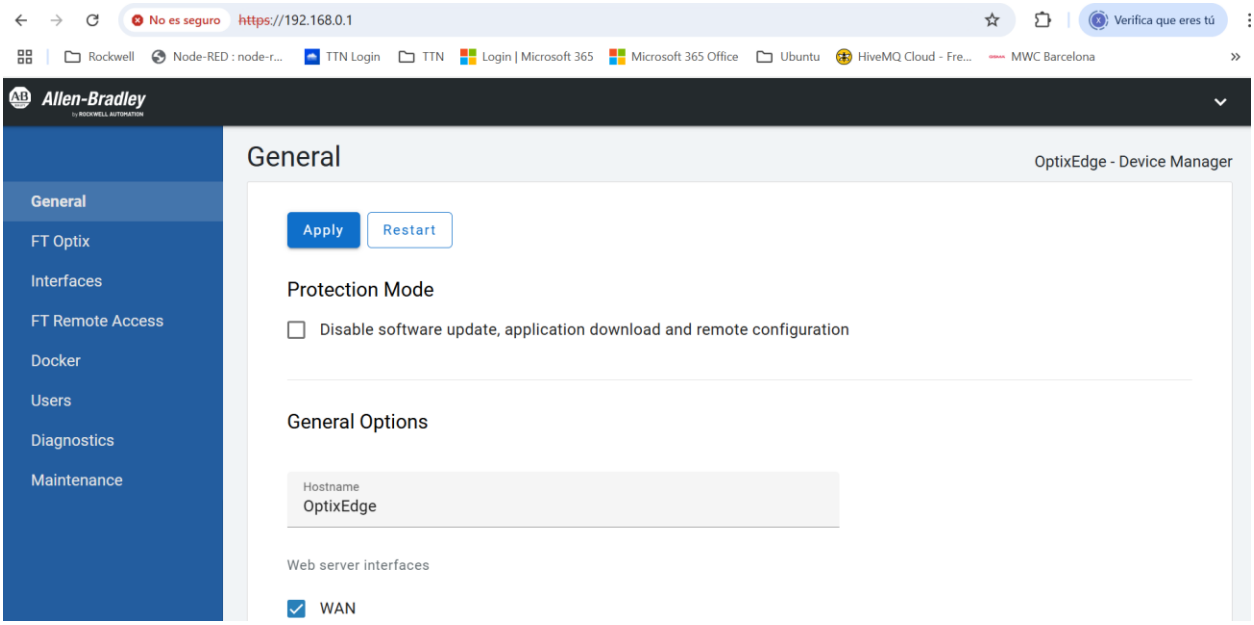
13.1. Wizard

Use port 2 with preassigned IP address 192.168.0.1
Open explorer to https:// 192.168.0.1:8443



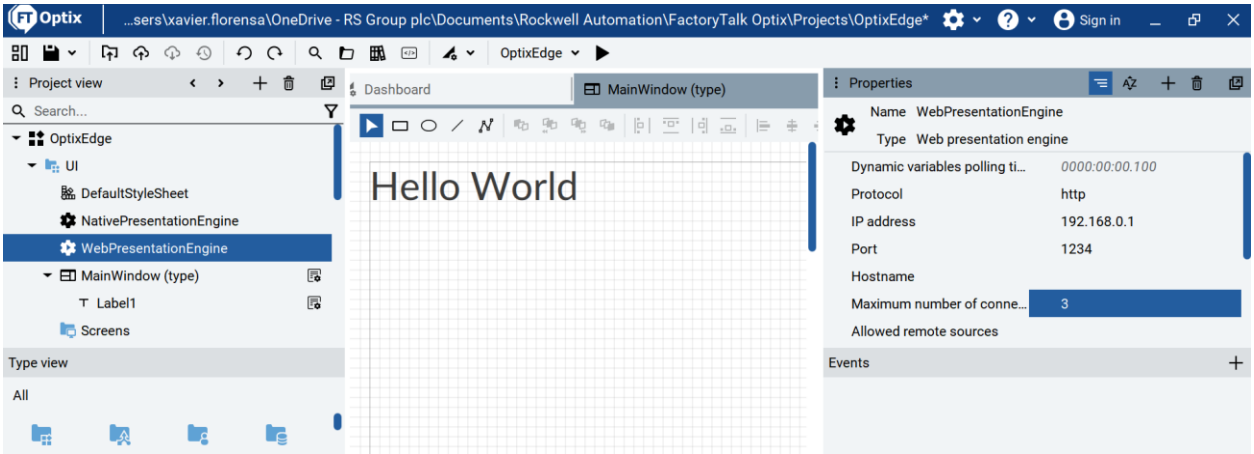
13.2. Device Manager

Open in browser
192.168.0.1
User admin
Password Risoul10

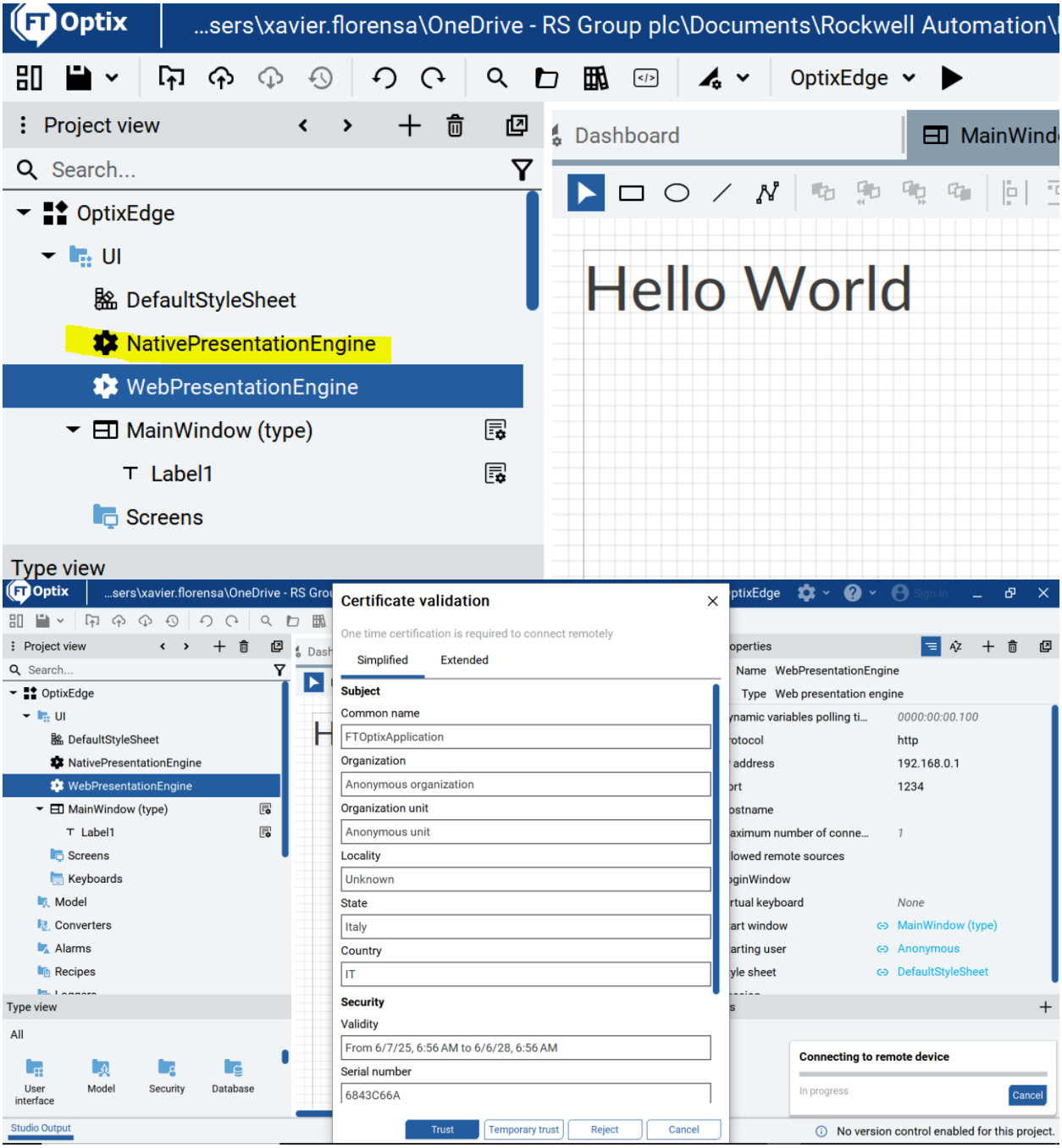


13.3. Deploy your first Optix Application

Add a web interface
And some content on MainWindow



You can delete native presentation output since this is not supported on Optix Edge



Open an explorer
Success

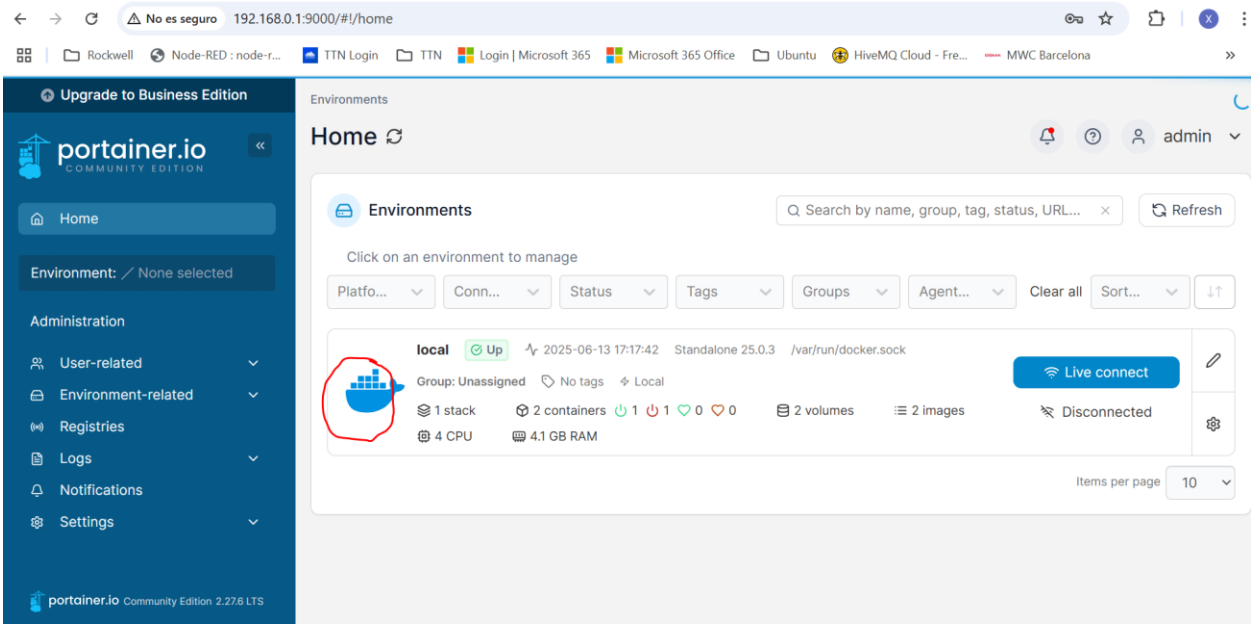
Close OptixEdge output to leave the application running and running on startup

13.4. Testing PLC Tag access

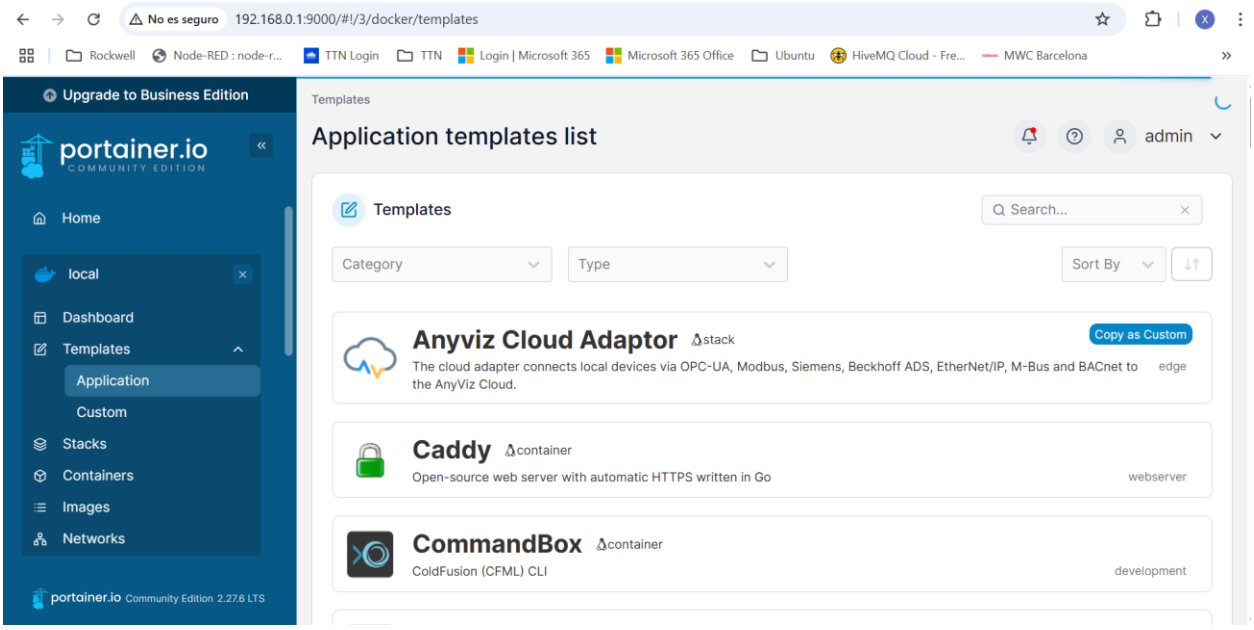
Success

13.5. Portainer

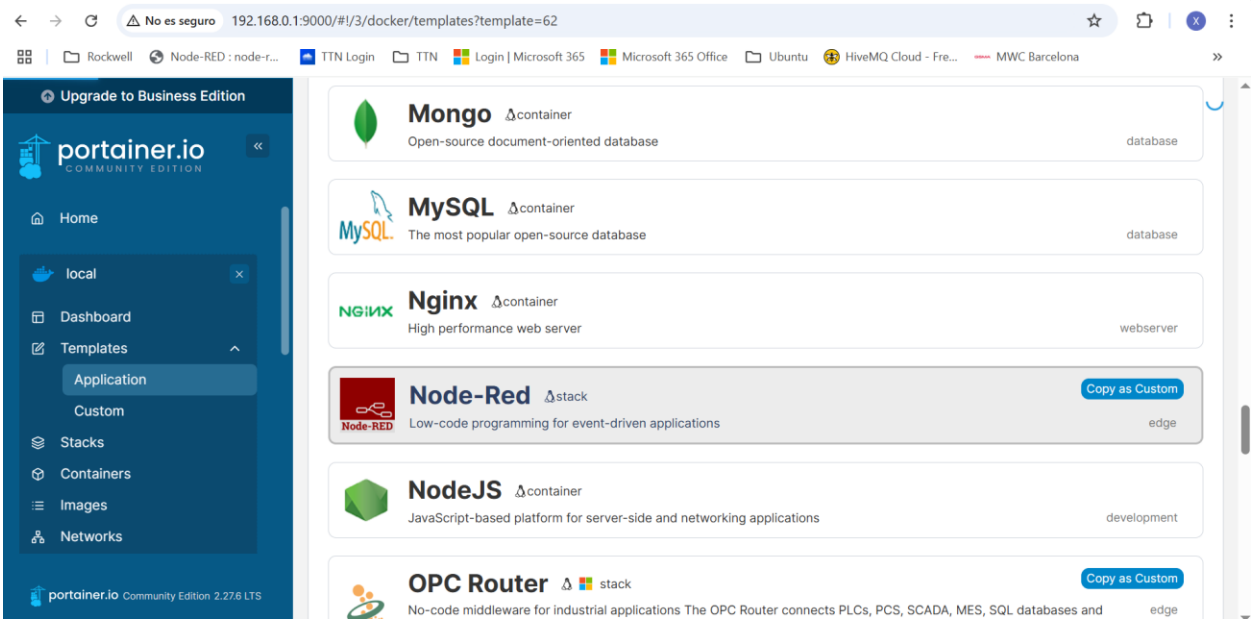
First enable Docker on OptixEdge configuration on 192.168.0.1
Then access portainer on your web browser 192.168.0.1:9000
Click on Docker



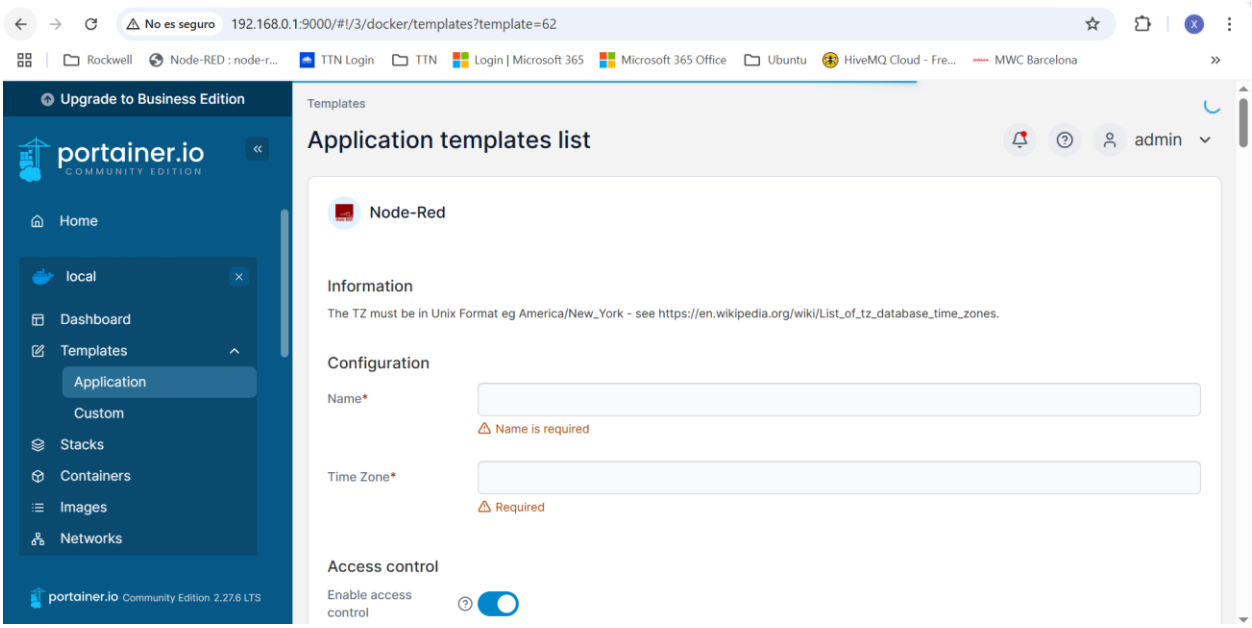
Go to Application templates
Look for your favorite application, for instance node-red



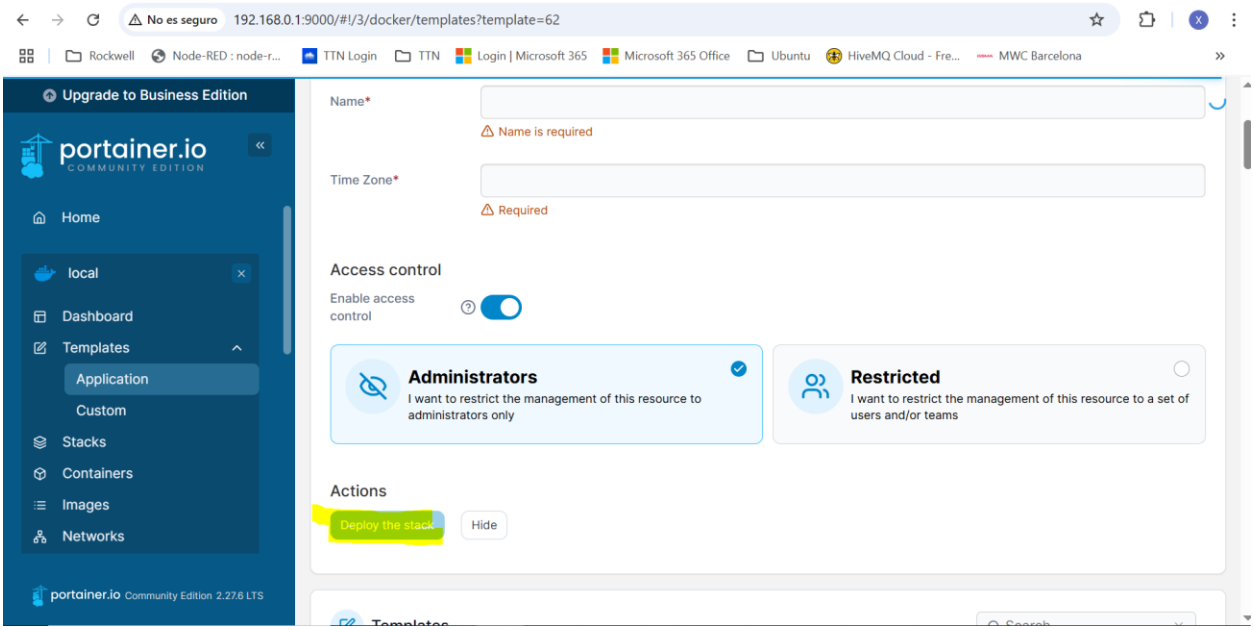
Select it



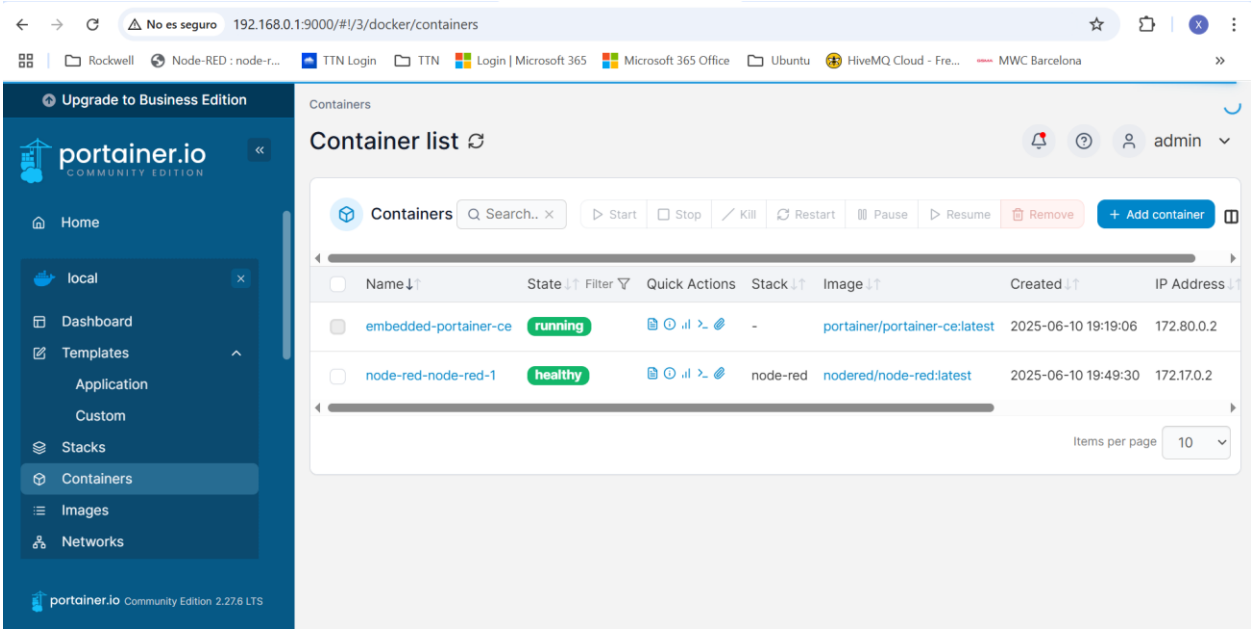
Go up and give a name



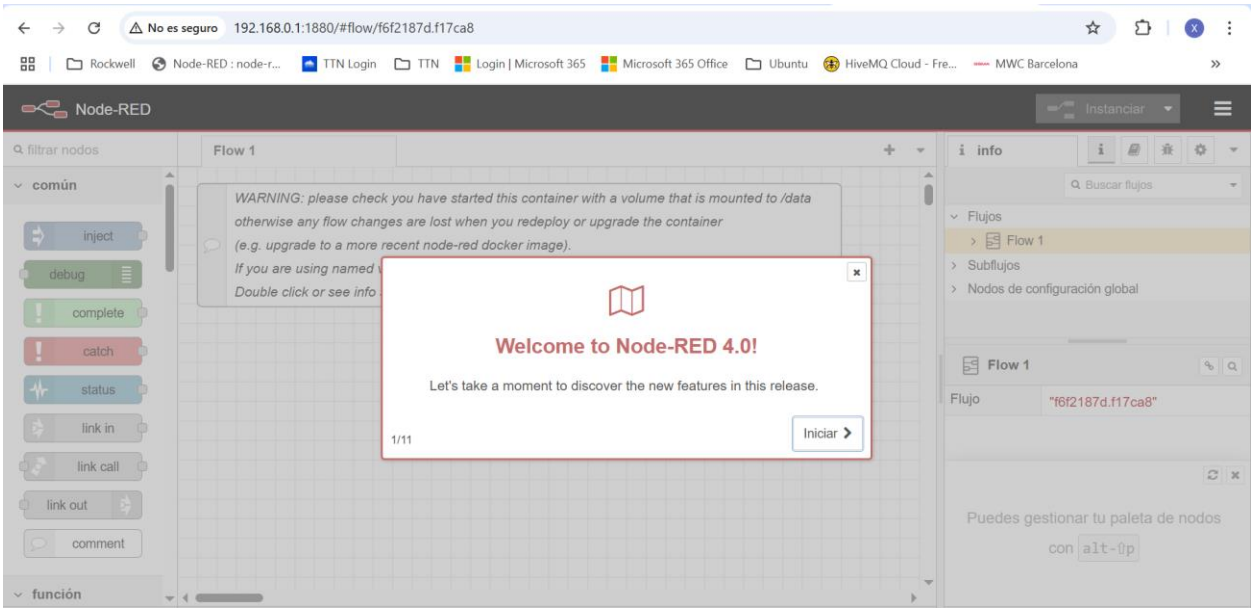
Deploy the stack a little bit downwards

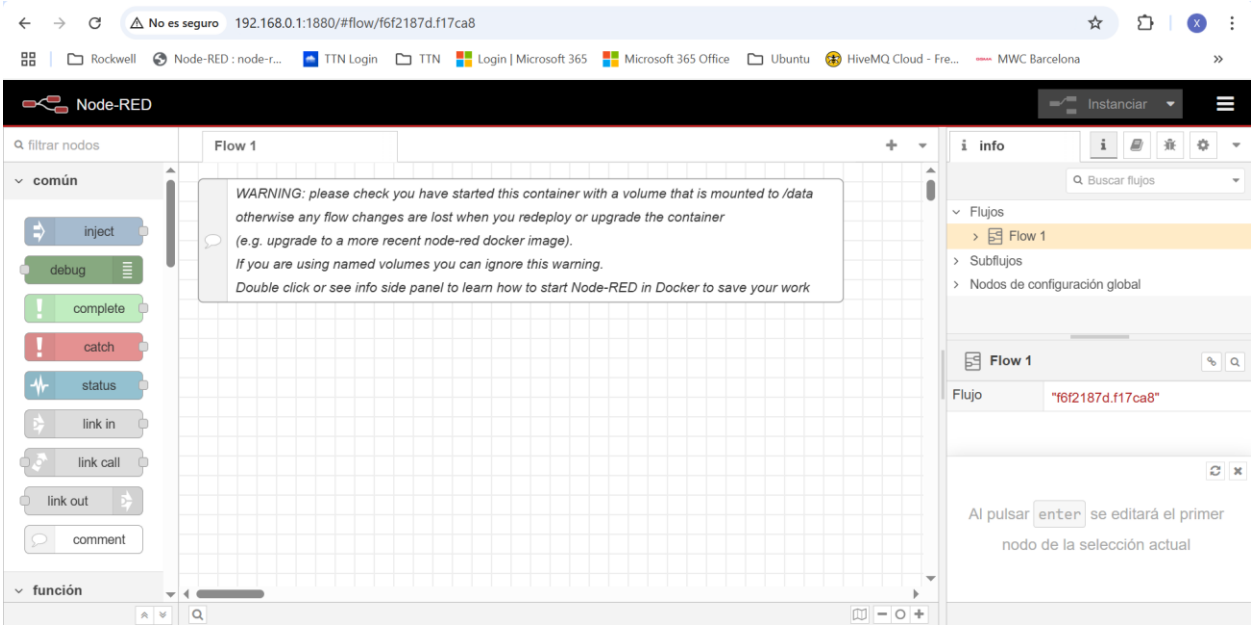


Look at the container list



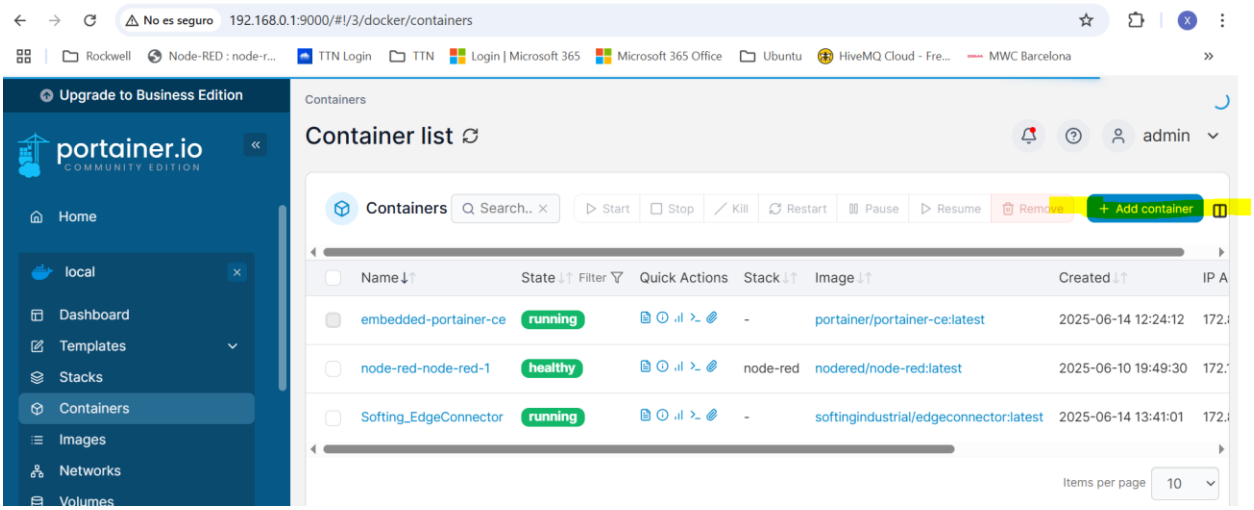
Go to a web browser and open the OptixEdge address and port 1880
voila



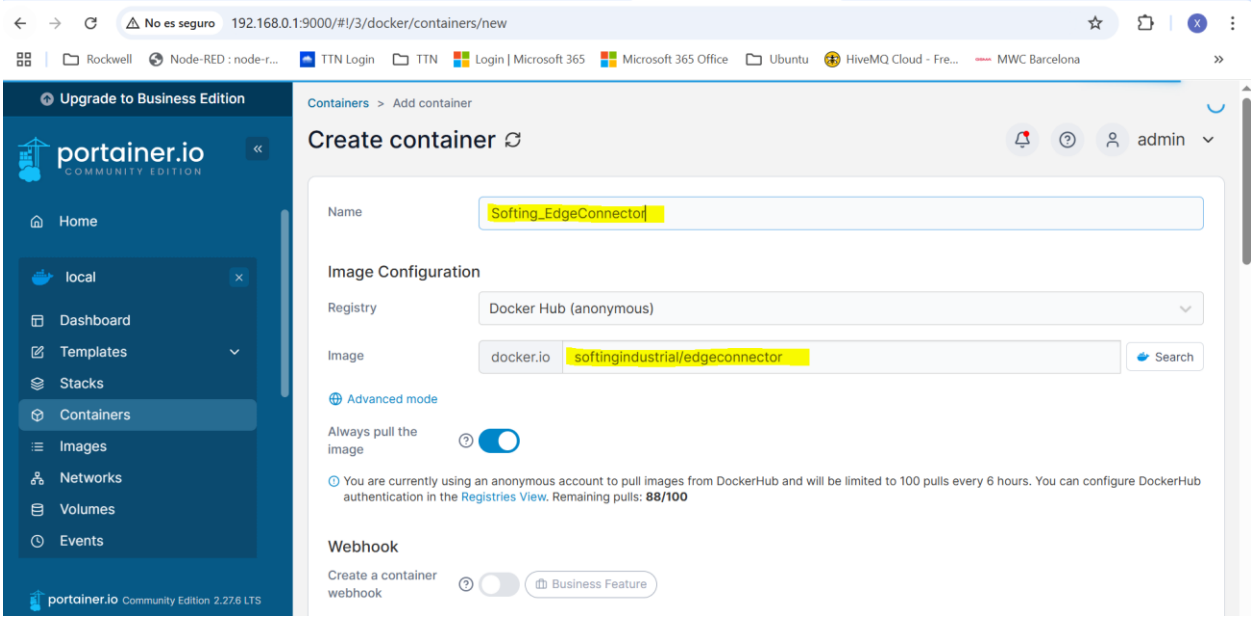


13.6. How to deploy a Softing EdgeConnector on OptixEdge

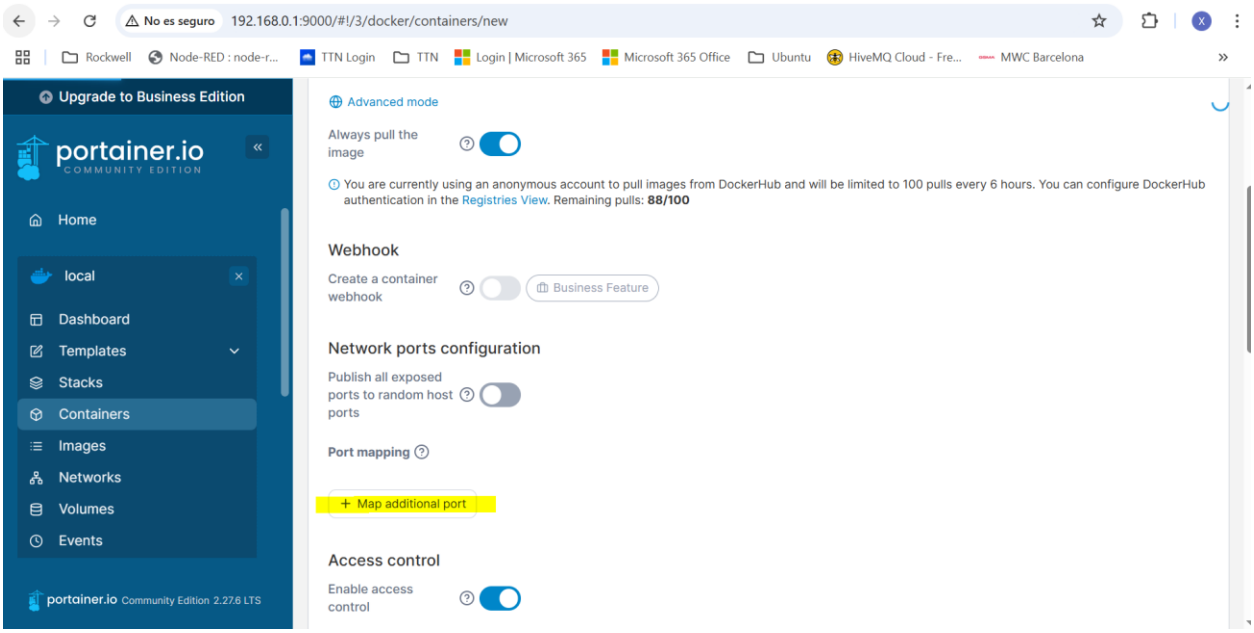
Add container



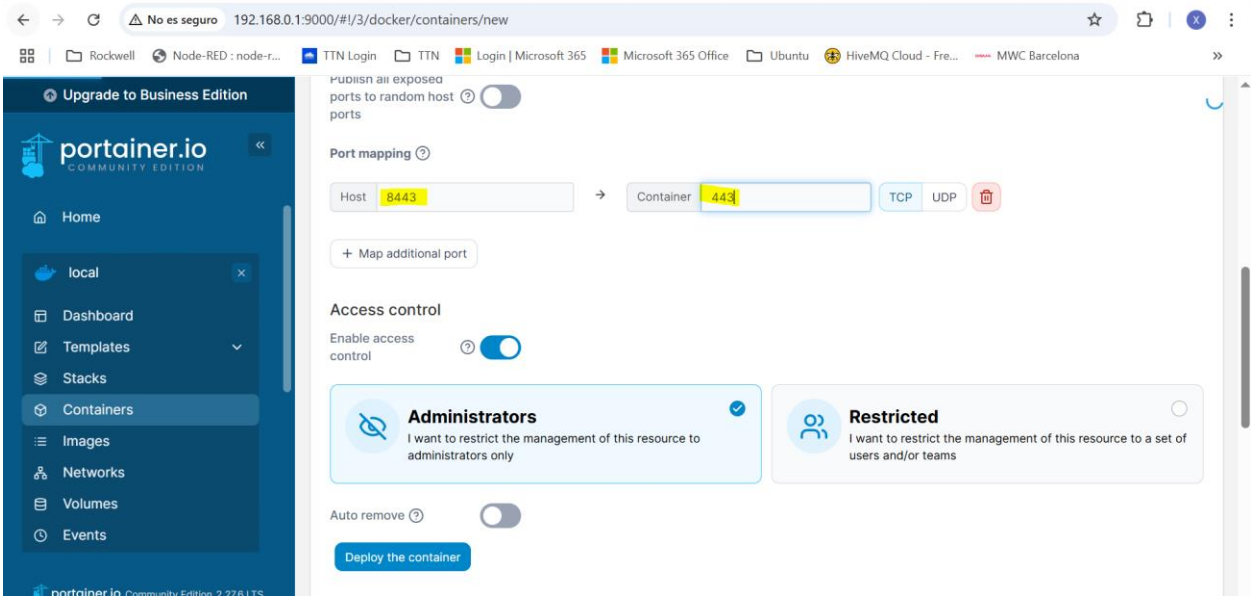
Give a name and use this url softingindustrial/edgeconnector



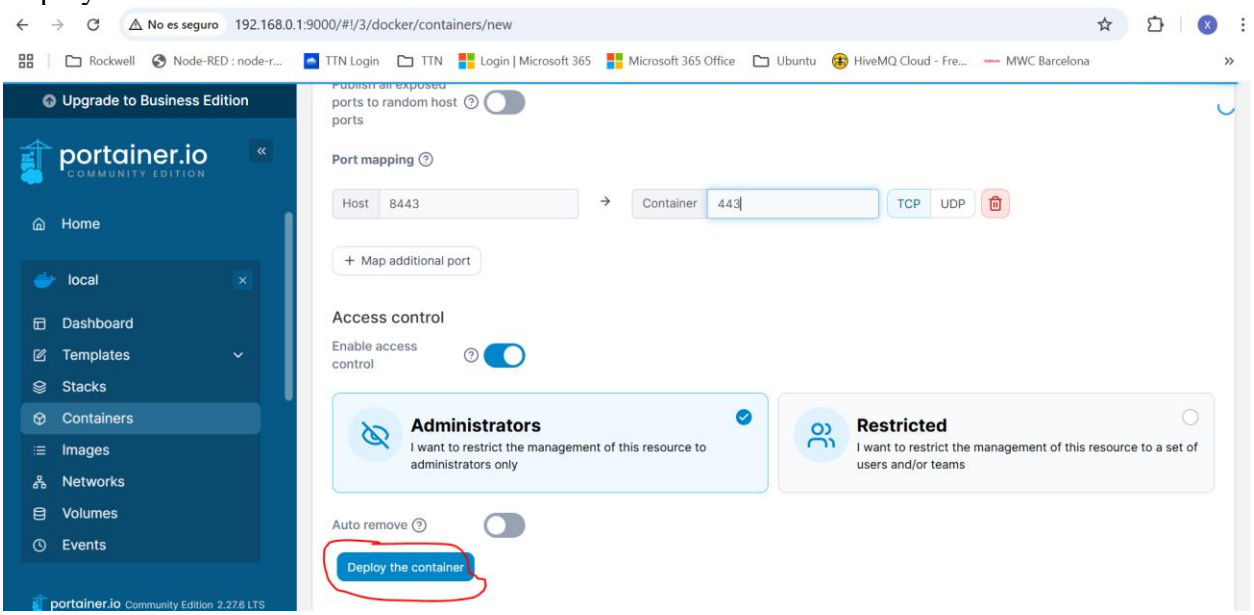
Click on add additional ports



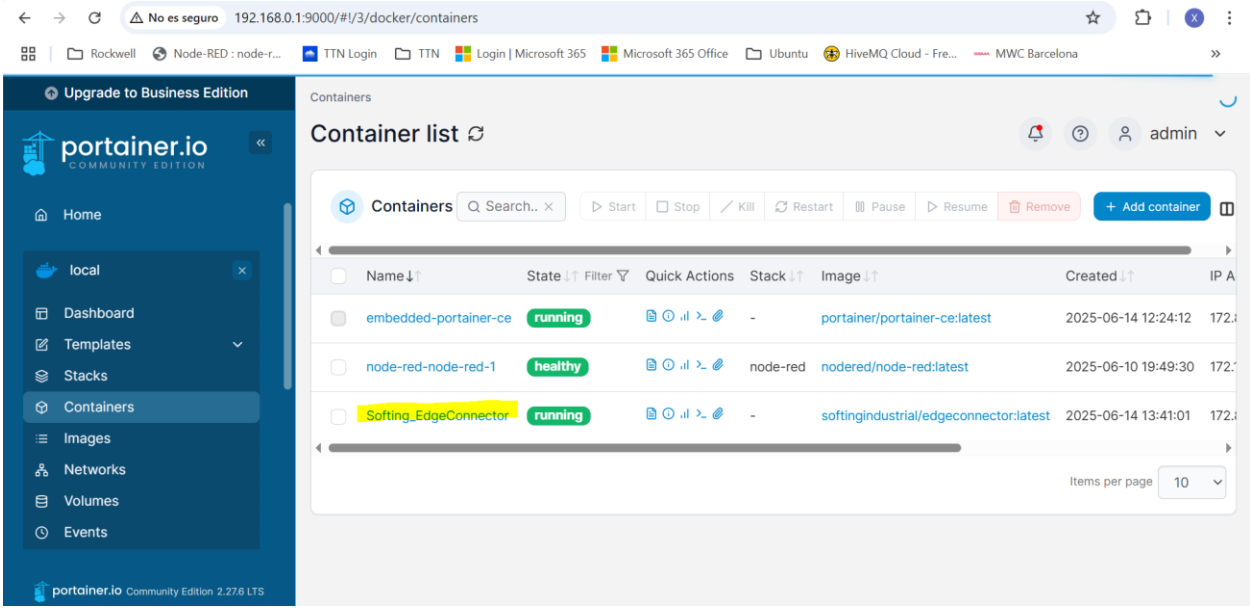
Us this port configuration (external port 443 is already used by Device manager so we should use another for instance 8443, and internal port is 443



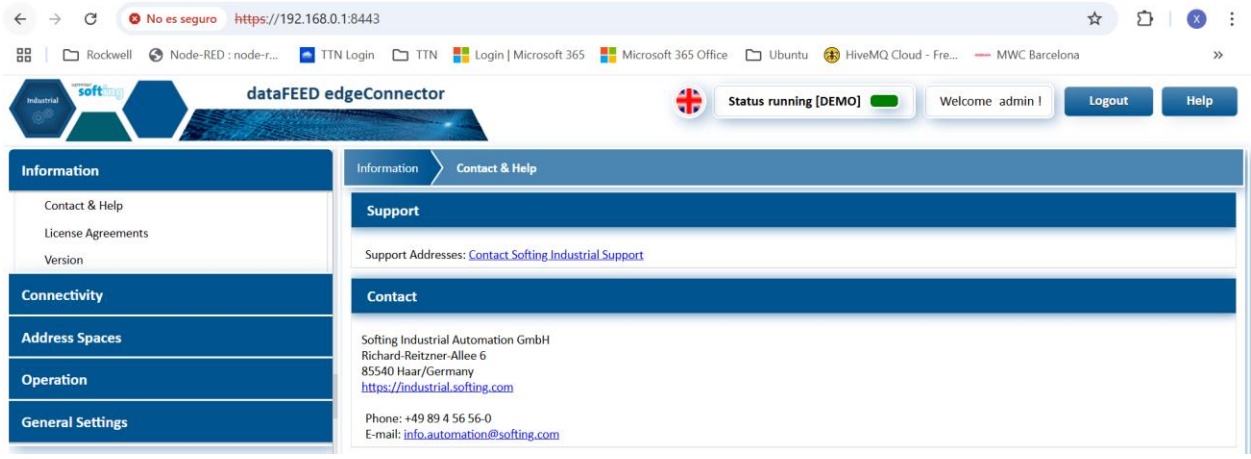
Deploy the container



We have it running



Open a web explorer on this address 192.168.0.1:8443
voila

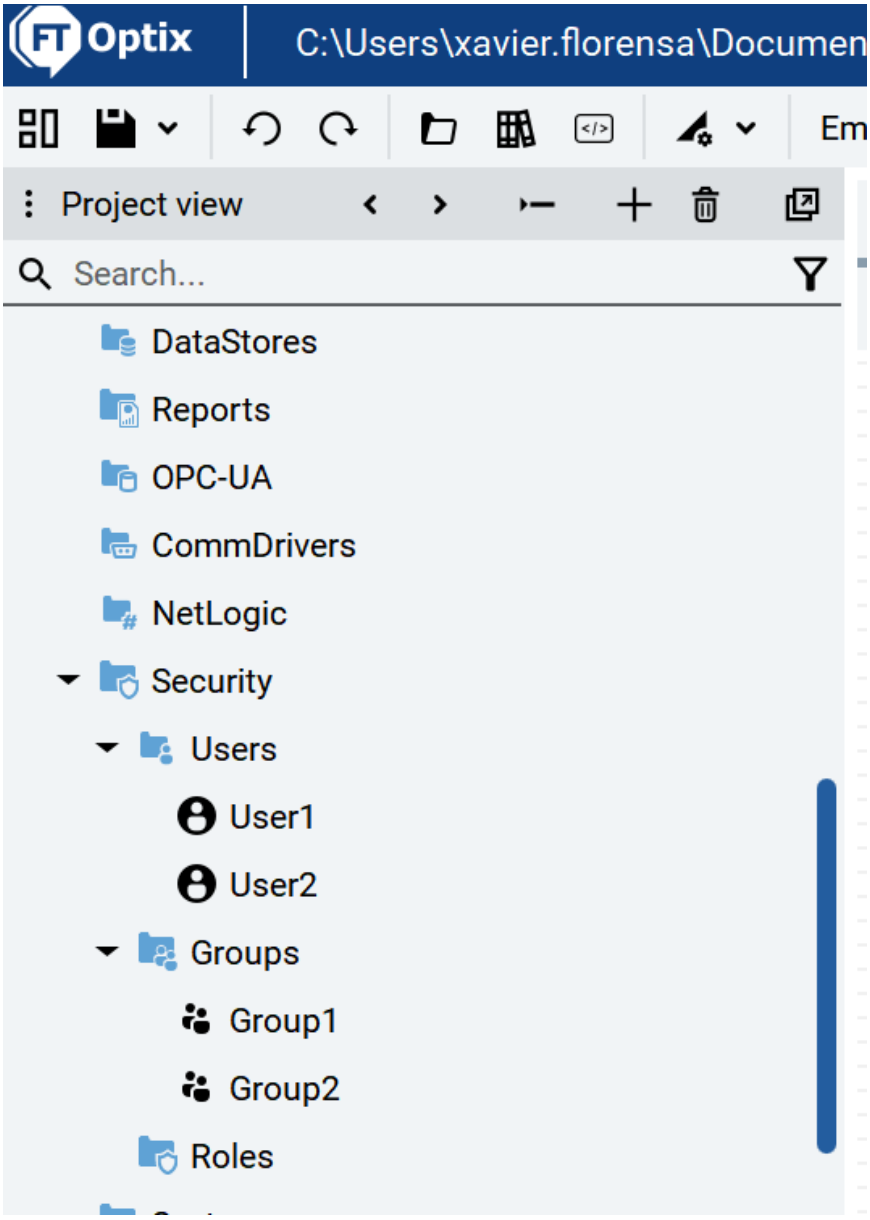


As you can see in this video
https://youtu.be/qTyxJ3Ixlng

14. Users in Optix

14.1. Creating users and Groups manually

Create users on Security Folder
Create Groups on Security Folder


Optix
C:\Users\xavier.florensa\Documen
Em
Project view
Search...
DataStores
Reports
OPC-UA
CommDrivers
NetLogic
Security
Users
User1
User2
Groups
Group1
Group2
Roles

14.2. Create a Login form

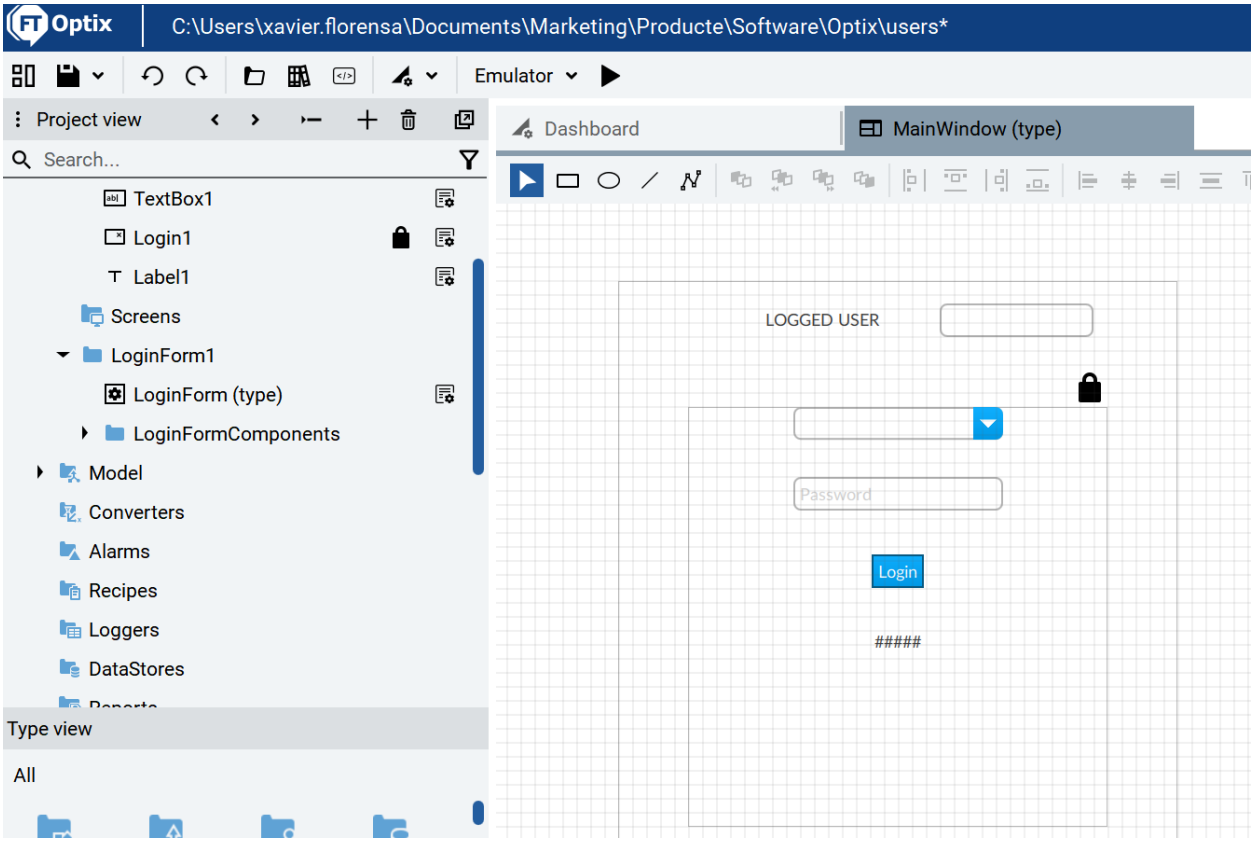
Create a login form

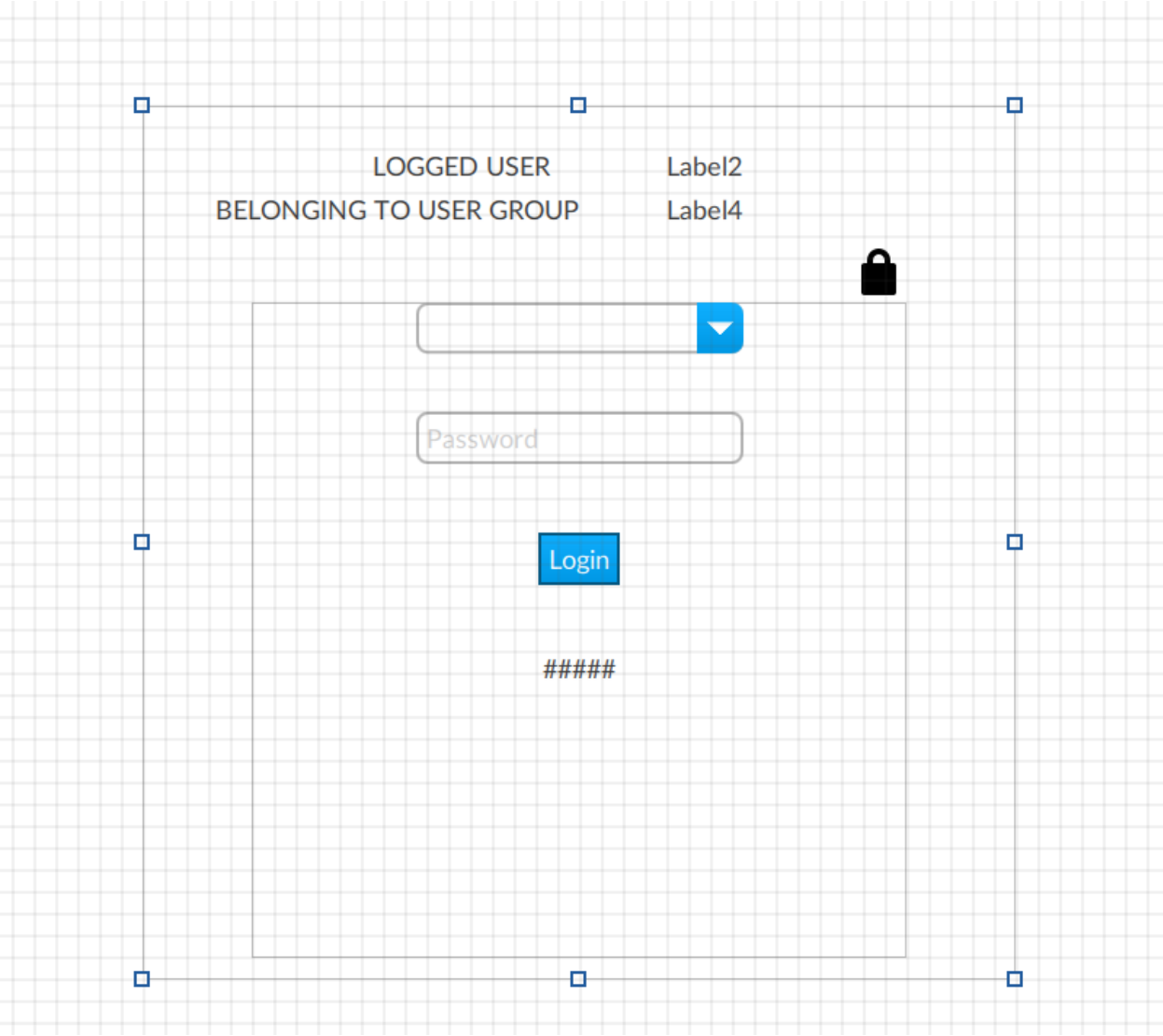
Create an interface that allows for logging in.

1. Add a preconfigured widget (type):
   a. In the toolbar, select **Template Libraries**.
   b. In **Libraries**, search for `Login form`.
   c. Under **Components**, drag **Login form** onto **UI** in **Project view**.
2. In **Project view**, right-click **MainWindow (type)** and select **New** > **LoginForm1** > **LoginFormComponents** > **Login**.
3. In **Properties**, create a dynamic link between **Users** and ***ProjectName*** > **Security** > **Users**.

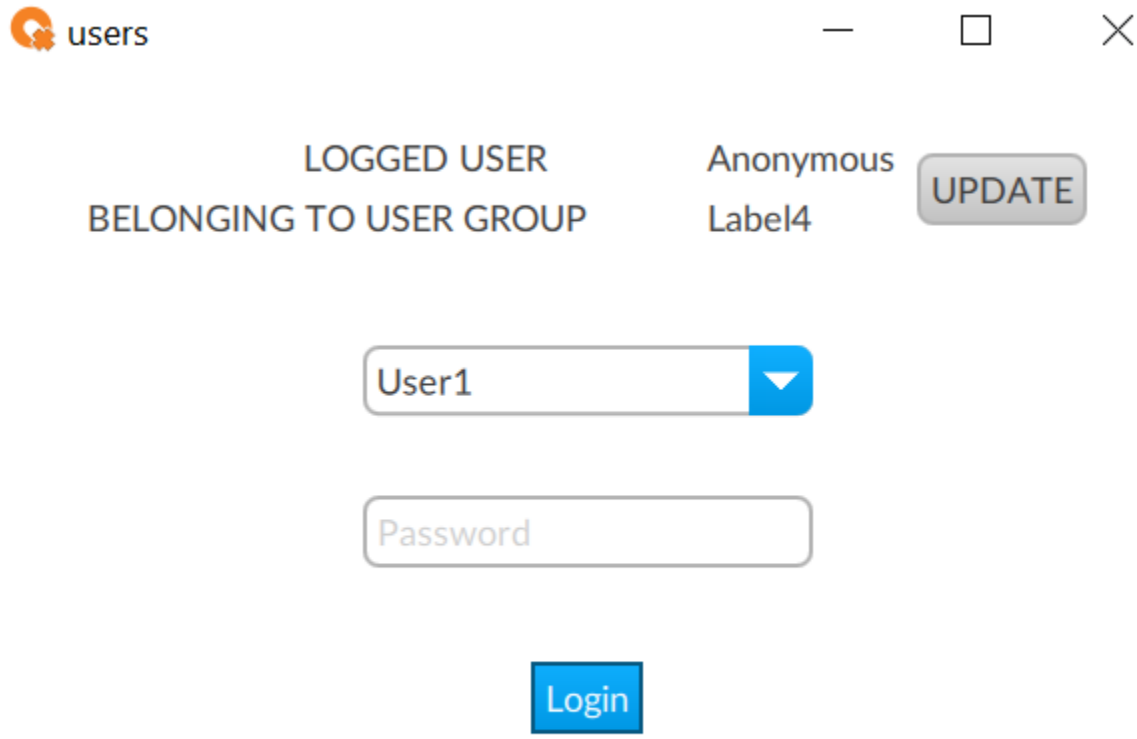
For more information about dynamic links, see Create dynamic links.

https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/application-examples/translations-tutorial/develop-a-multilingual-login-form/create-a-login-form.html

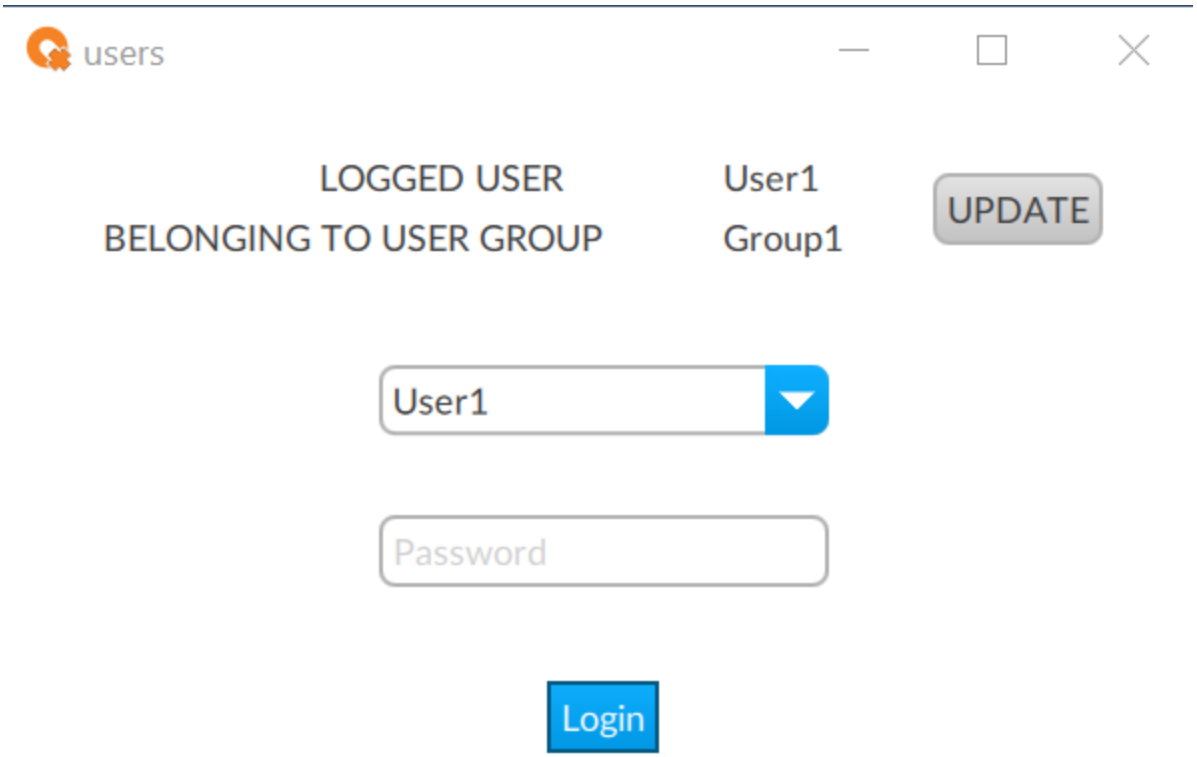



LOGGED USER
Label2
BELONGING TO USER GROUP
Label4
Password
Login
#####

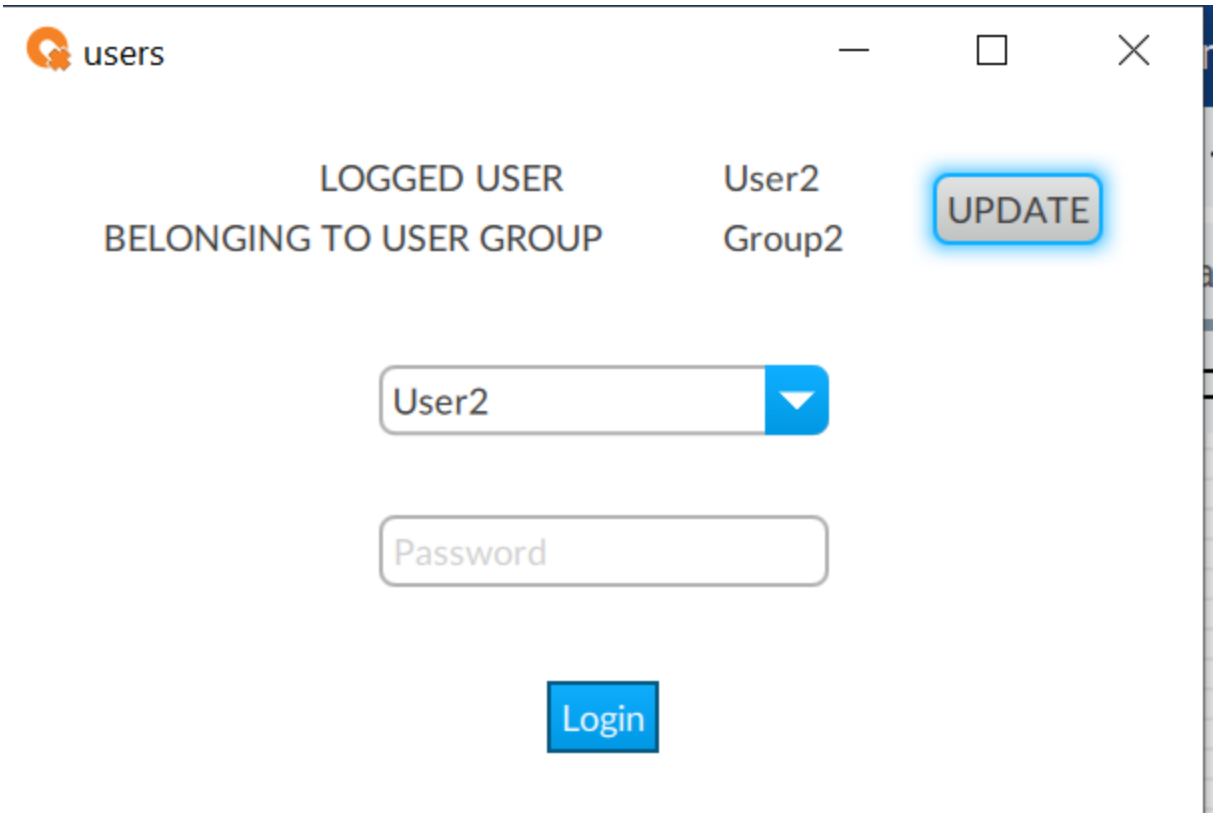
14.3. Writing the logged user to the PLC



Login User1


users
LOGGED USER
User1
BELONGING TO USER GROUP
Group1
UPDATE
User1
Password
Login

Login user 2



It works!
This is the code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.WebUI;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.Core;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Mostrar el usuari logat
        var usuario = Session.User;
```

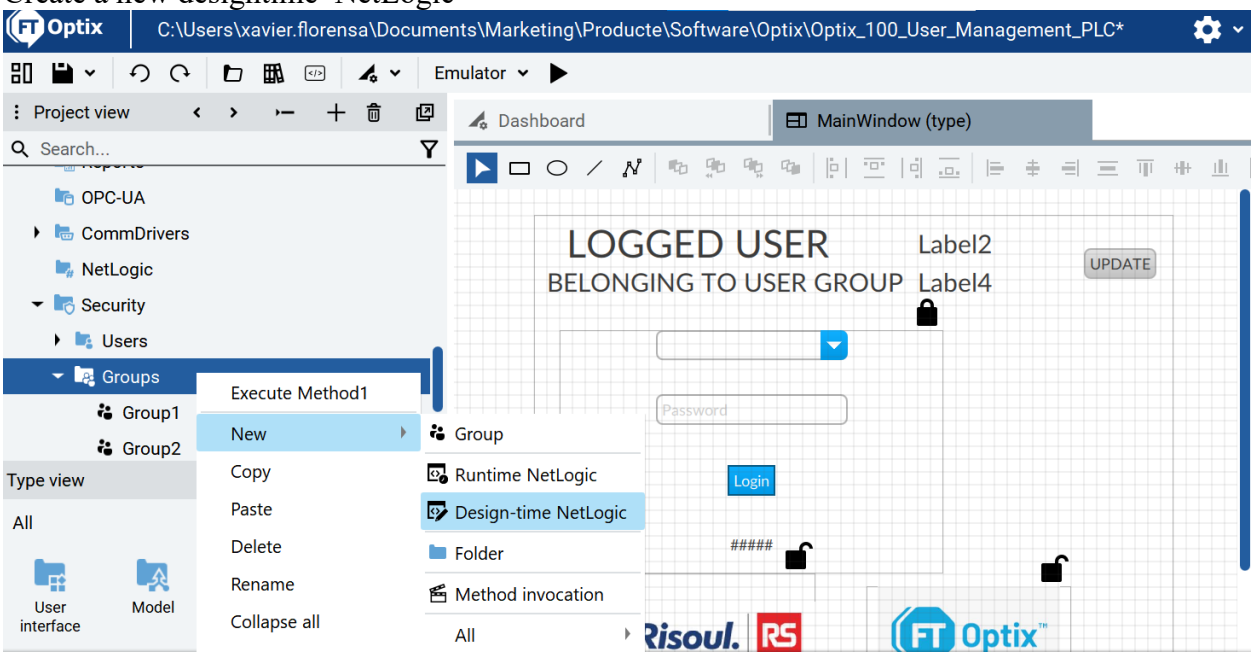
```
        var user = Session.User.BrowseName;
        var mylabel = Project.Current.Get<Label>("UI/MainWindow/Label2");
        mylabel.Text = user.ToString();
        //Mostra el grup del usuari
        var userGroups =
usuario.Refs.GetObjects(FTOptix.Core.ReferenceTypes.HasGroup, false);
        foreach (var group in userGroups)
        {
            Log.Info(group.BrowseName);
            Project.Current.GetVariable("Model/Locales/UserGroups").Value =
group.BrowseName;
        }
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
    [ExportMethod]

    public void Update()
    {
       Actualitza();
       Log.Info("Updating: ");
    }

     private void Actualitza()
    {
       // Mostrar el usuari logat
       var usuario = Session.User;
       var user = Session.User.BrowseName;
       var mylabel = Project.Current.Get<Label>("UI/MainWindow/Label2");
       mylabel.Text = user.ToString();
       //Mostra el grup del usuari
       var userGroups =
usuario.Refs.GetObjects(FTOptix.Core.ReferenceTypes.HasGroup, false);
       foreach (var group in userGroups)
       {
            Log.Info(group.BrowseName);
            var mylabel_group =
Project.Current.Get<Label>("UI/MainWindow/Label4");
            mylabel_group.Text = group.BrowseName.ToString();
            Project.Current.GetVariable("Model/Locales/UserGroups").Value =
group.BrowseName;
       }
```

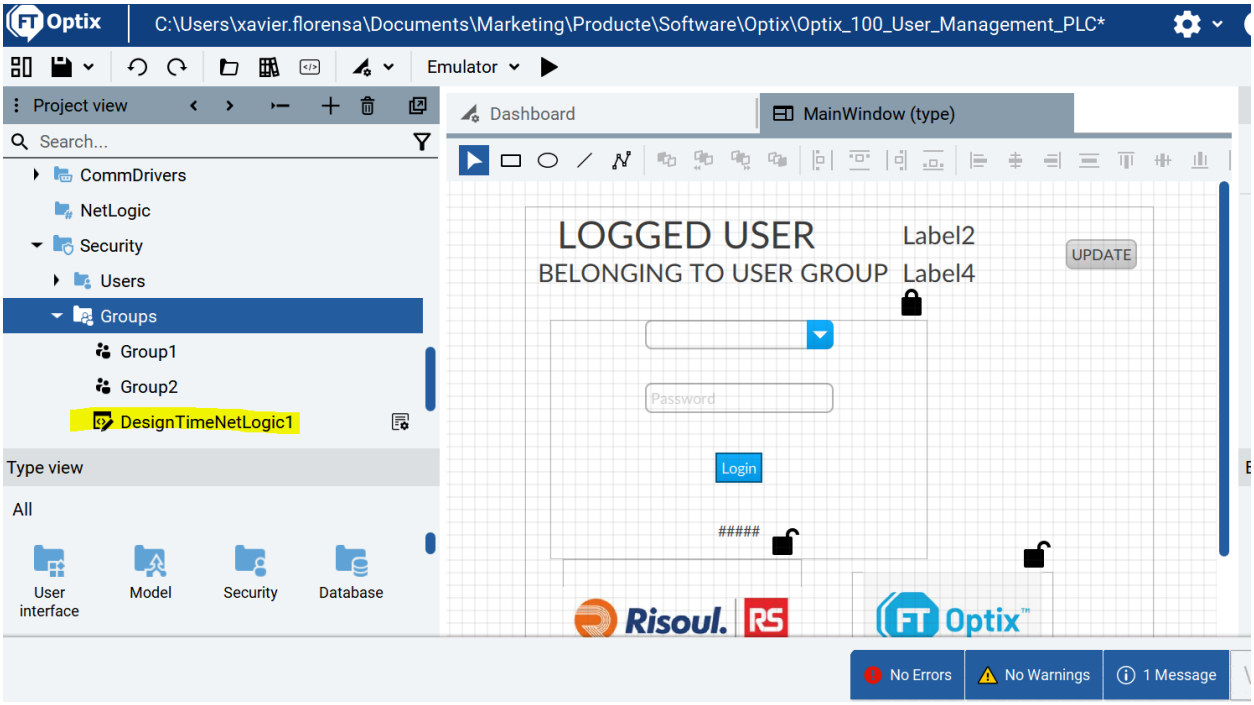
```
    }

}
```

As you can see on this video
https://www.youtube.com/watch?v=3D85iqPMuCo&t=9s
You can find the code here
https://github.com/xavierflorensa/Optix_User_Management_PLC

14.4. Creating Groups in design time

Create a new designtime NetLogic





Open the code

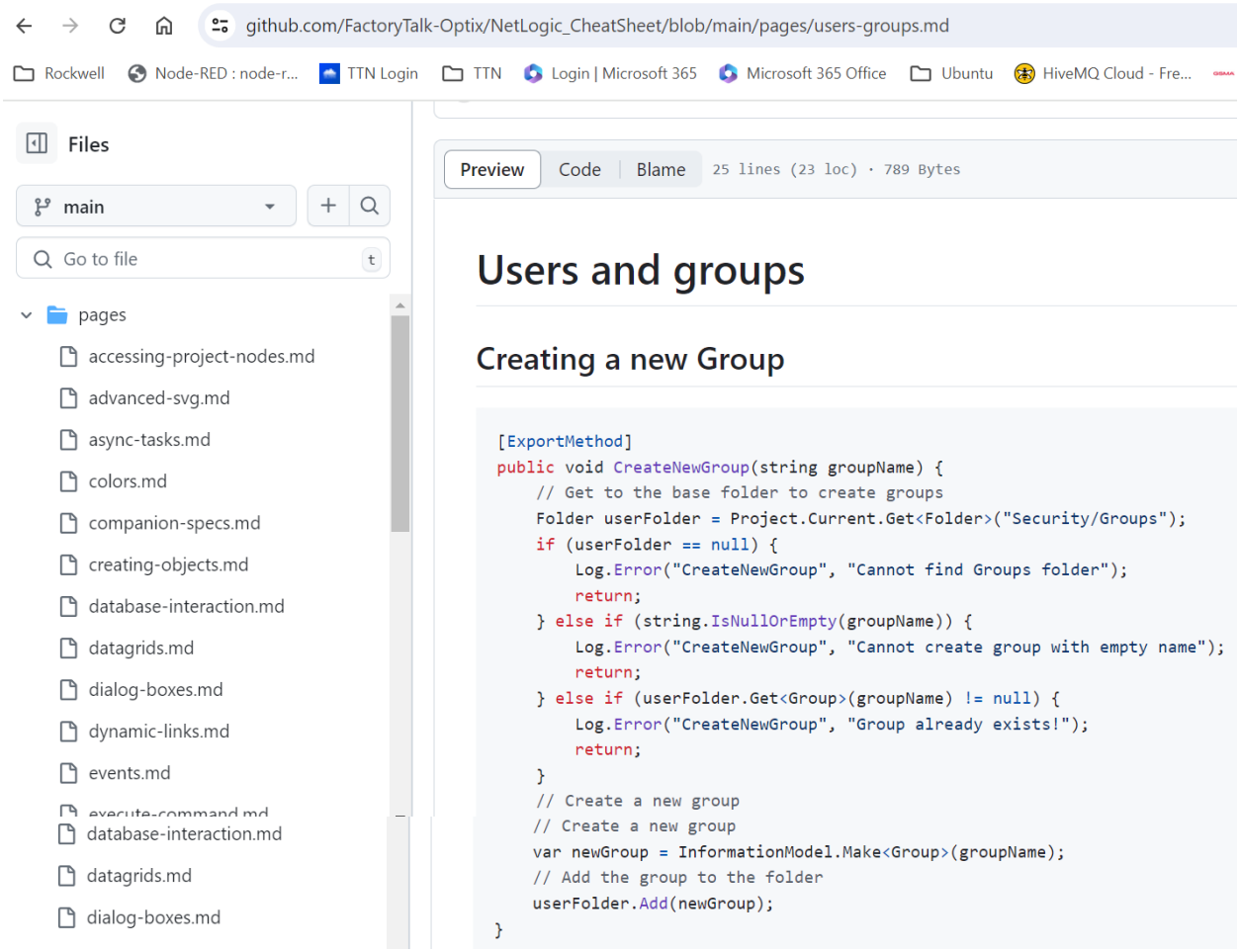
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic1 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method

    }
}
```

Copy this code, as a simple way to create a new Group with gibe name: “Nou_Grup”

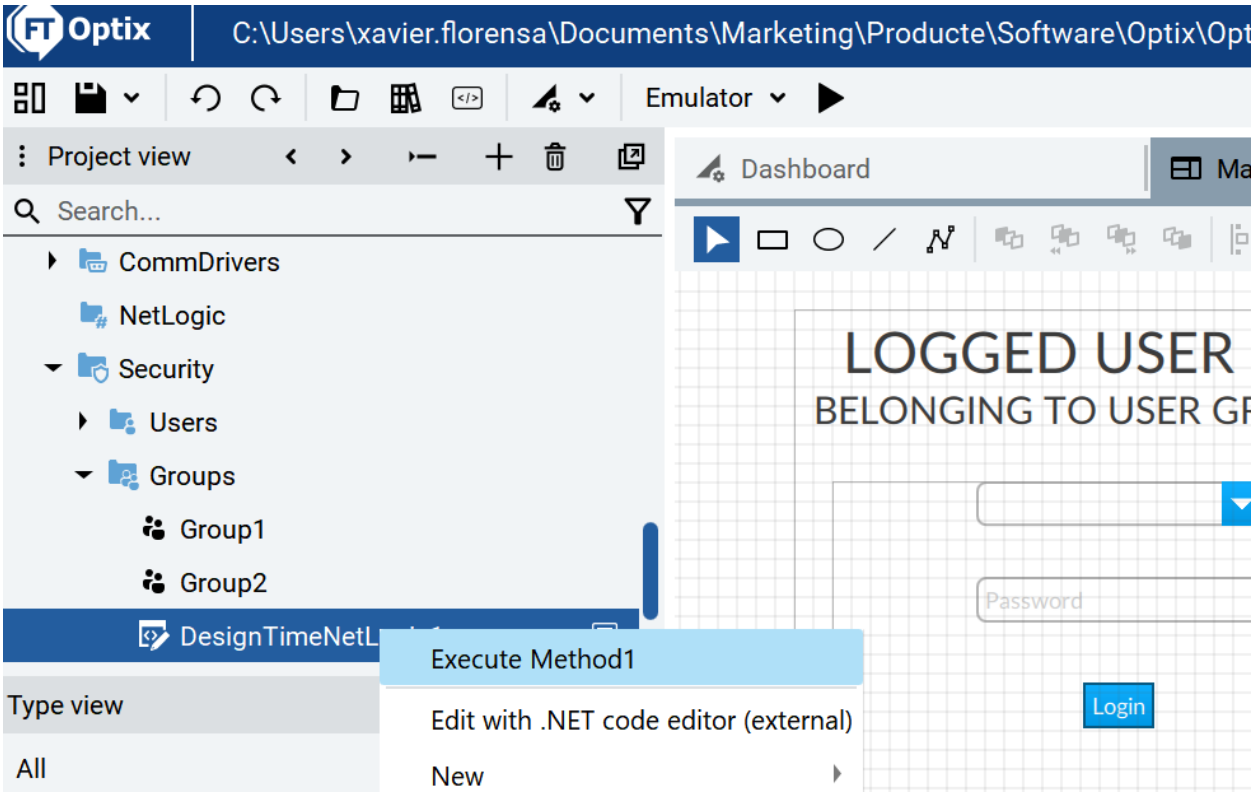
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic1 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
        Folder userFolder = Project.Current.Get<Folder>("Security/Groups");
    if (userFolder == null) {
        Log.Error("CreateNewGroup", "Cannot find Groups folder");
        return;
    }
    /*else if (string.IsNullOrEmpty(groupName)) {
        Log.Error("CreateNewGroup", "Cannot create group with empty name");
        return;
    } else if (userFolder.Get<Group>(groupName) != null) {
        Log.Error("CreateNewGroup", "Group already exists!");
        return;
    }
    */
    // Create a new group
    var newGroup = InformationModel.Make<Group>("Nou_Grup");
    // Add the group to the folder
    userFolder.Add(newGroup);
    }
}
```

This is done using following RA recommendations:
https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/users-groups.md

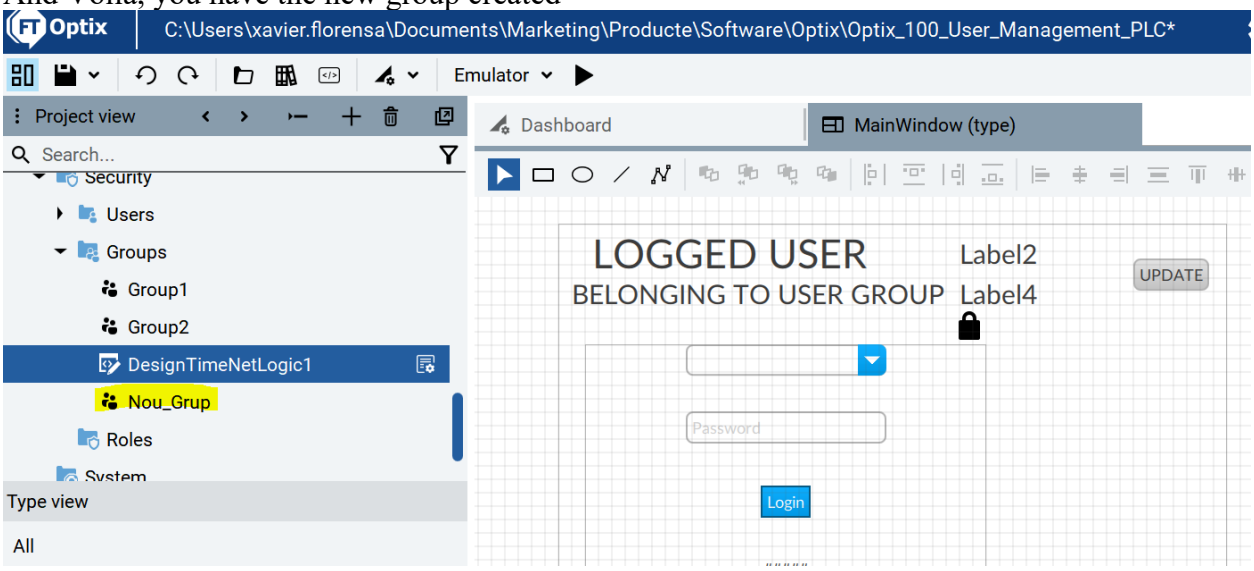
We are using only the inside part of the method



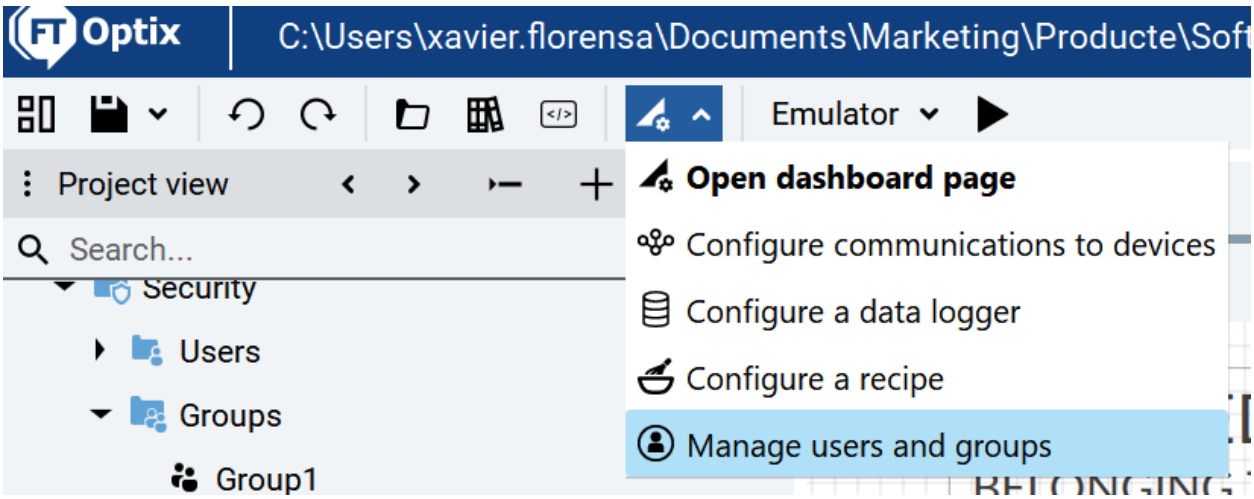
Now you are ready to execute the method this way in design time:



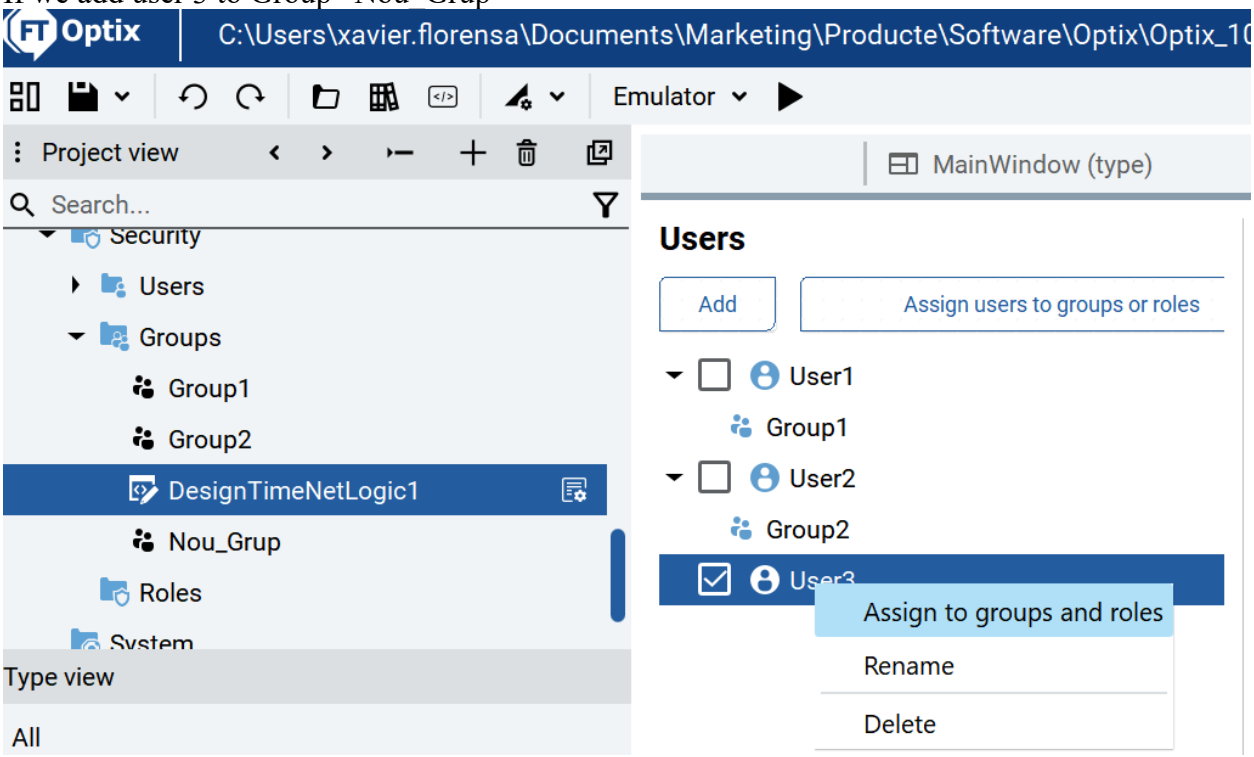
And Voilà, you have the new group created

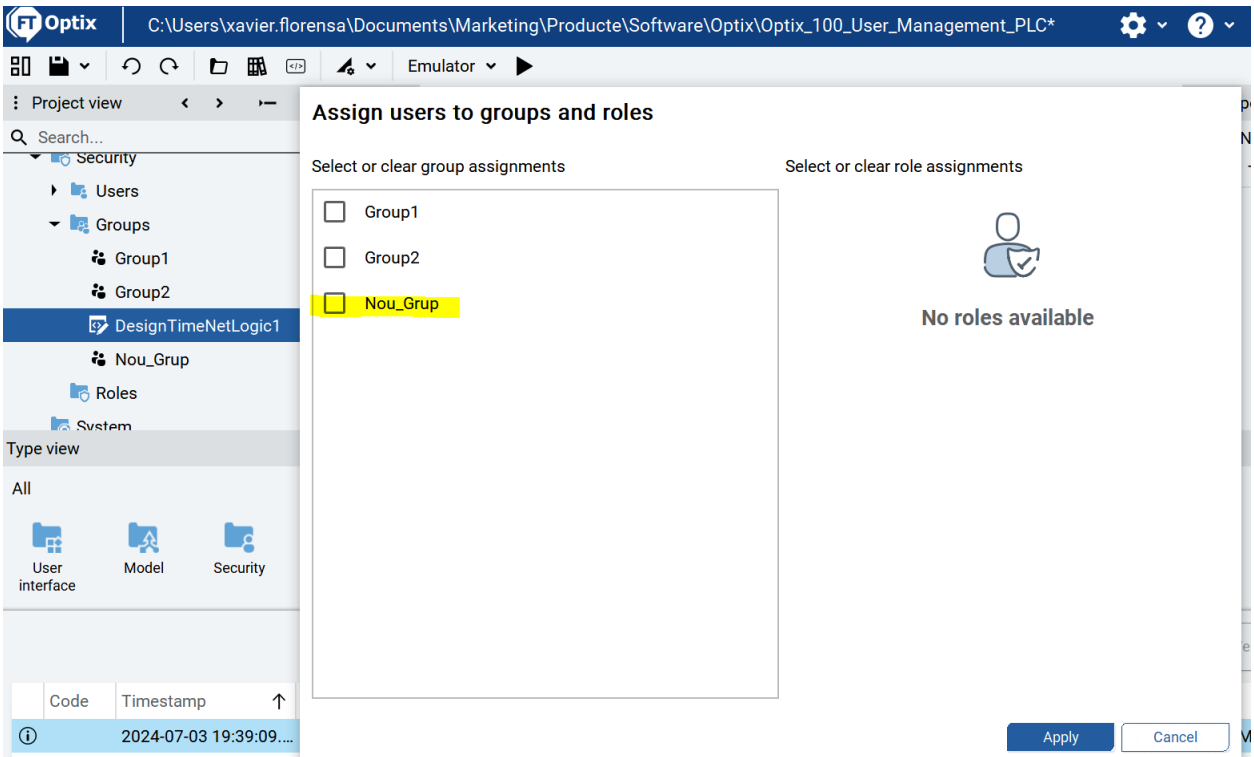


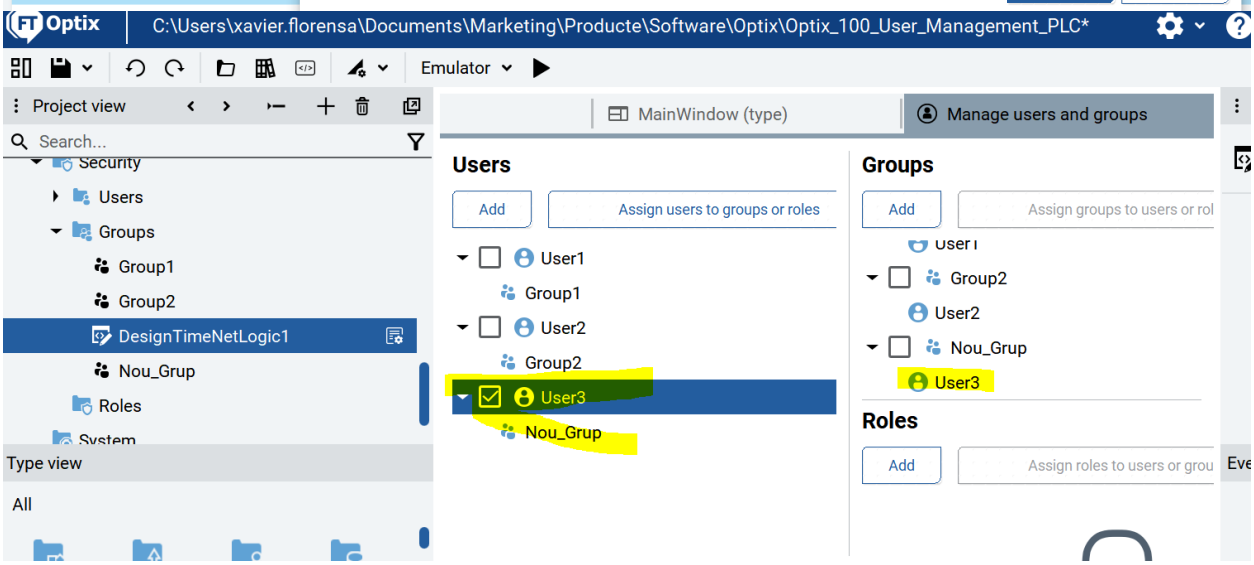
And you have this new group available to add users



If we add user 3 to Group "Nou_Grup"

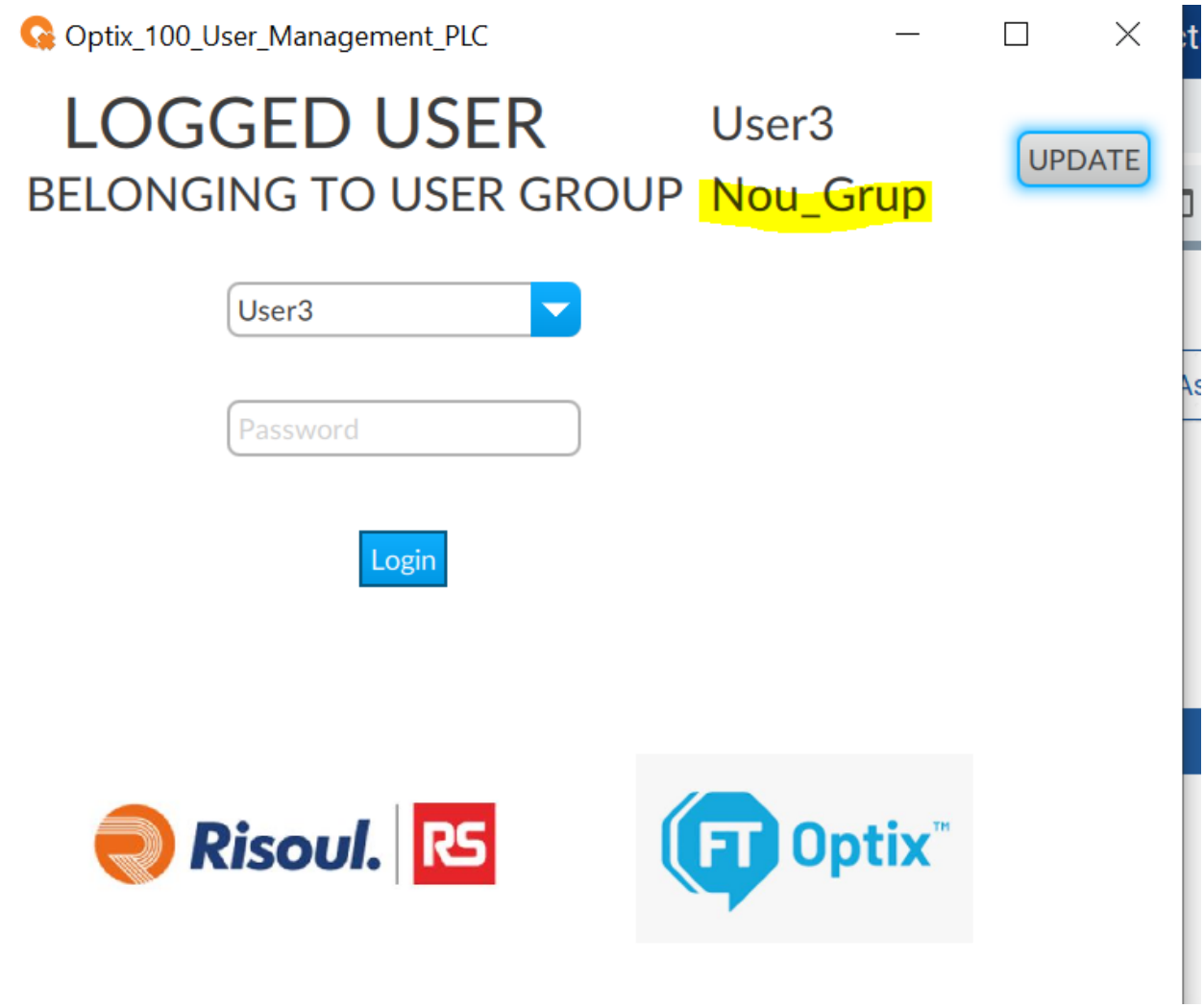






If we execute the project, we will be able to log as user 3

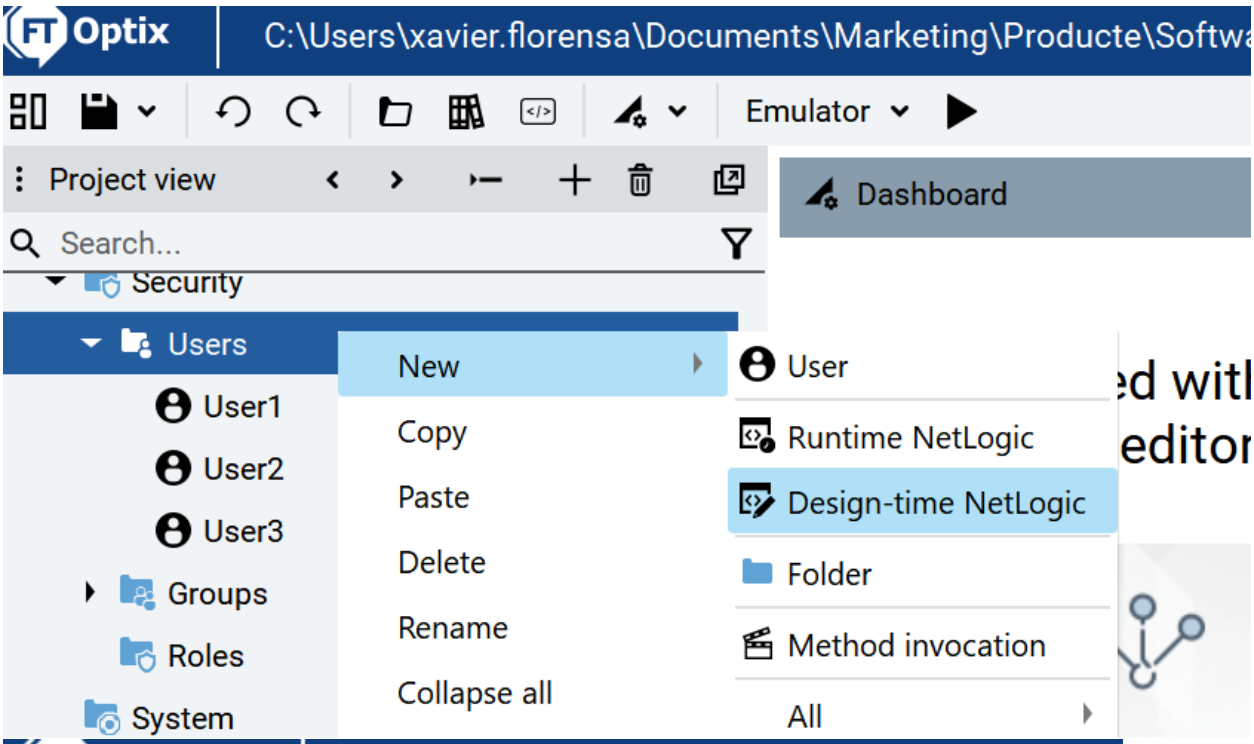
First log in and then click on UPDATE

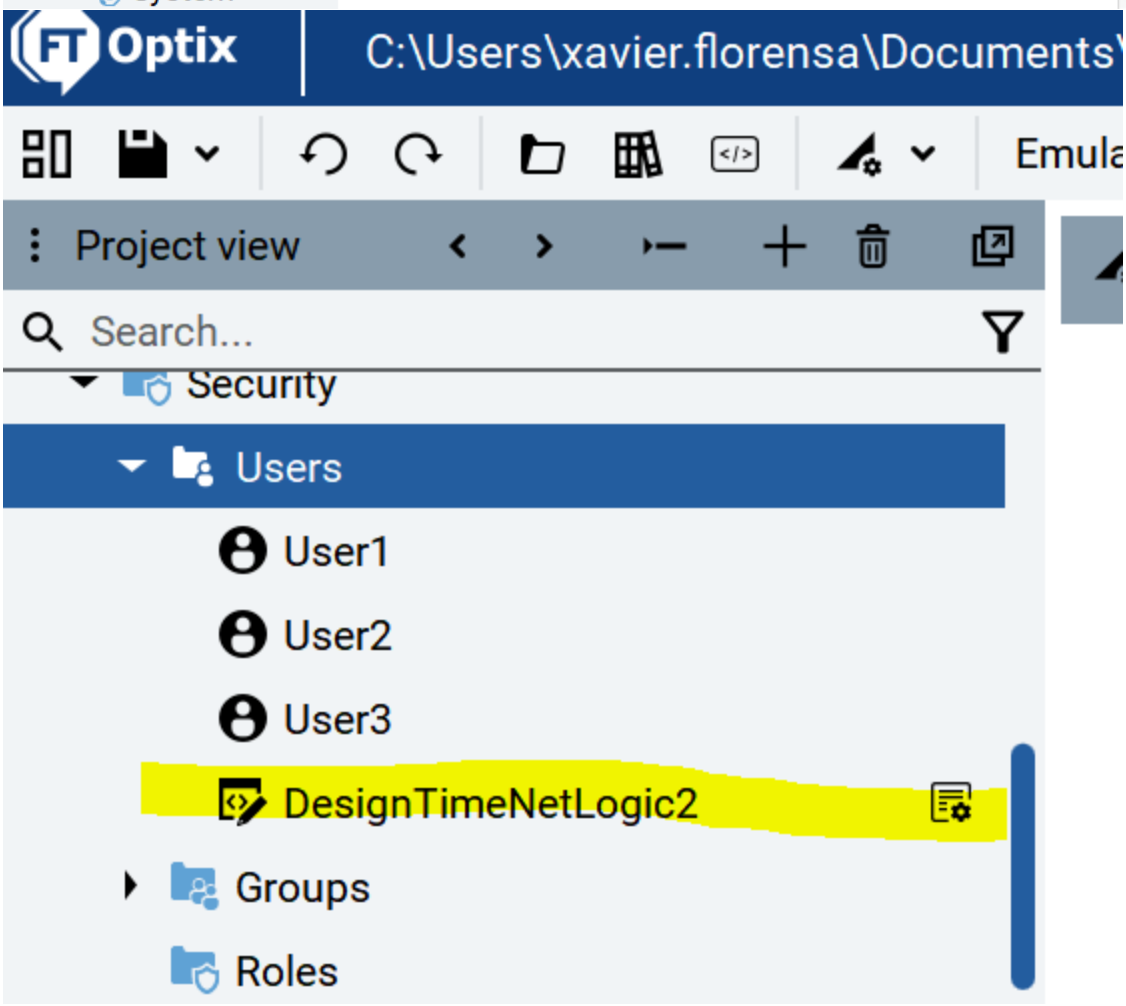


14.5. Creating Users in Design time

Let's try with this code

Let's create a new Design Time Netlogic


Optix
C:\Users\xavier.florensa\Documents\Marketing\Producte\Softwa
Emulator
Project view
Dashboard
Search...
Security
Users
User1
User2
User3
Groups
Roles
System
New
Copy
Paste
Delete
Rename
Collapse all
User
Runtime NetLogic
Design-time NetLogic
Folder
Method invocation
All


Optix
C:\Users\xavier.florensa\Documents\
Project view
Search...
Security
Users
User1
User2
User3
DesignTimeNetLogic2
Groups
Roles

Open the code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic2 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
    }
}
```

Try with this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic2 : BaseNetLogic
{
    [ExportMethod]
    public void Method1()
```

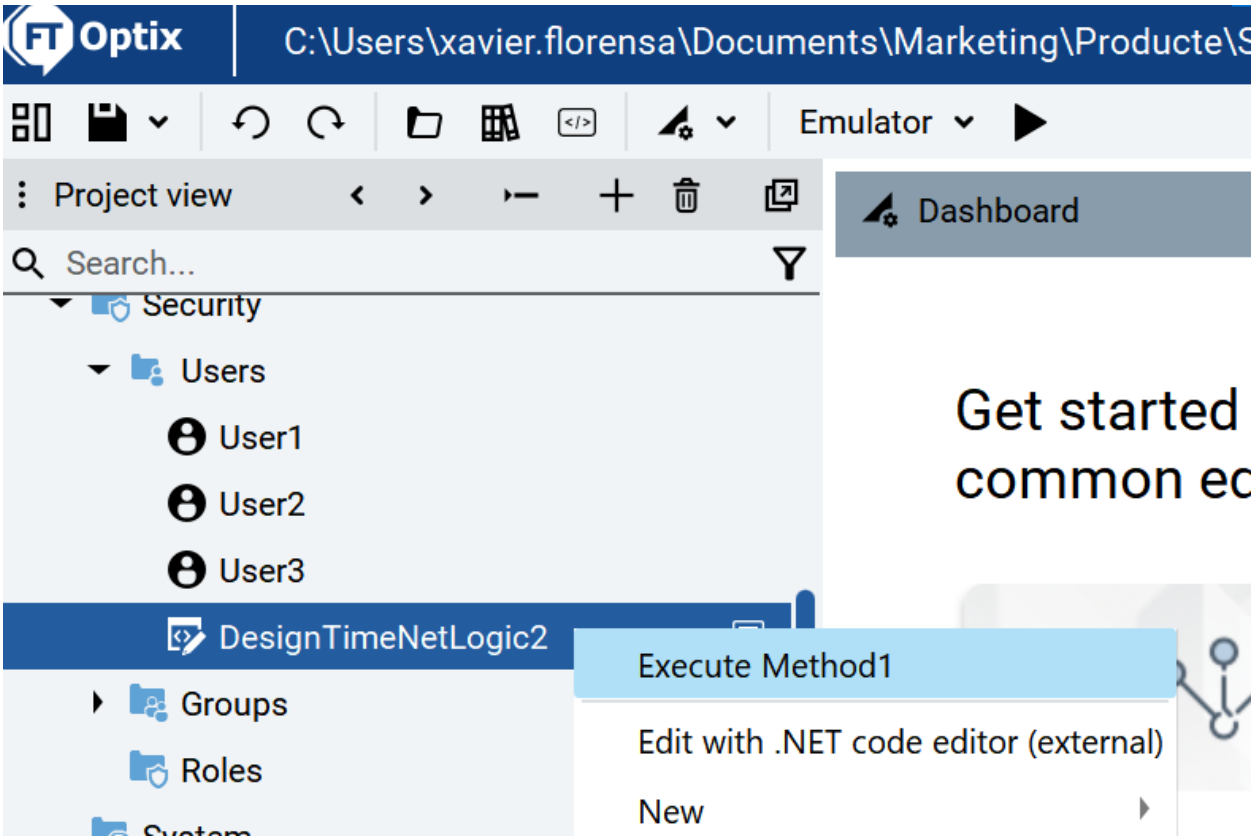
```
    {
        // Insert code to be executed by the method
                // Insert code to be executed by the method
        Folder userFolder = Project.Current.Get<Folder>("Security/Users");
    if (userFolder == null) {
        Log.Error("CreateNewUser", "Cannot find Users folder");
        return;
    }
    /*else if (string.IsNullOrEmpty(groupName)) {
        Log.Error("CreateNewGroup", "Cannot create group with empty name");
        return;
    } else if (userFolder.Get<Group>(groupName) != null) {
        Log.Error("CreateNewGroup", "Group already exists!");
        return;
    }
    */
    // Create a new group
    //var newGroup = InformationModel.Make<Group>("Nou_Grup");
    //var newUser = InformationModel.Create<User>("Xavier");
    var newUser = InformationModel.Make<User>("Xavier");
    // Add the group to the folder
    //userFolder.Add(newGroup);
    userFolder.Add(newUser);

    }
}
```

Or even more clean with unnecessary lines

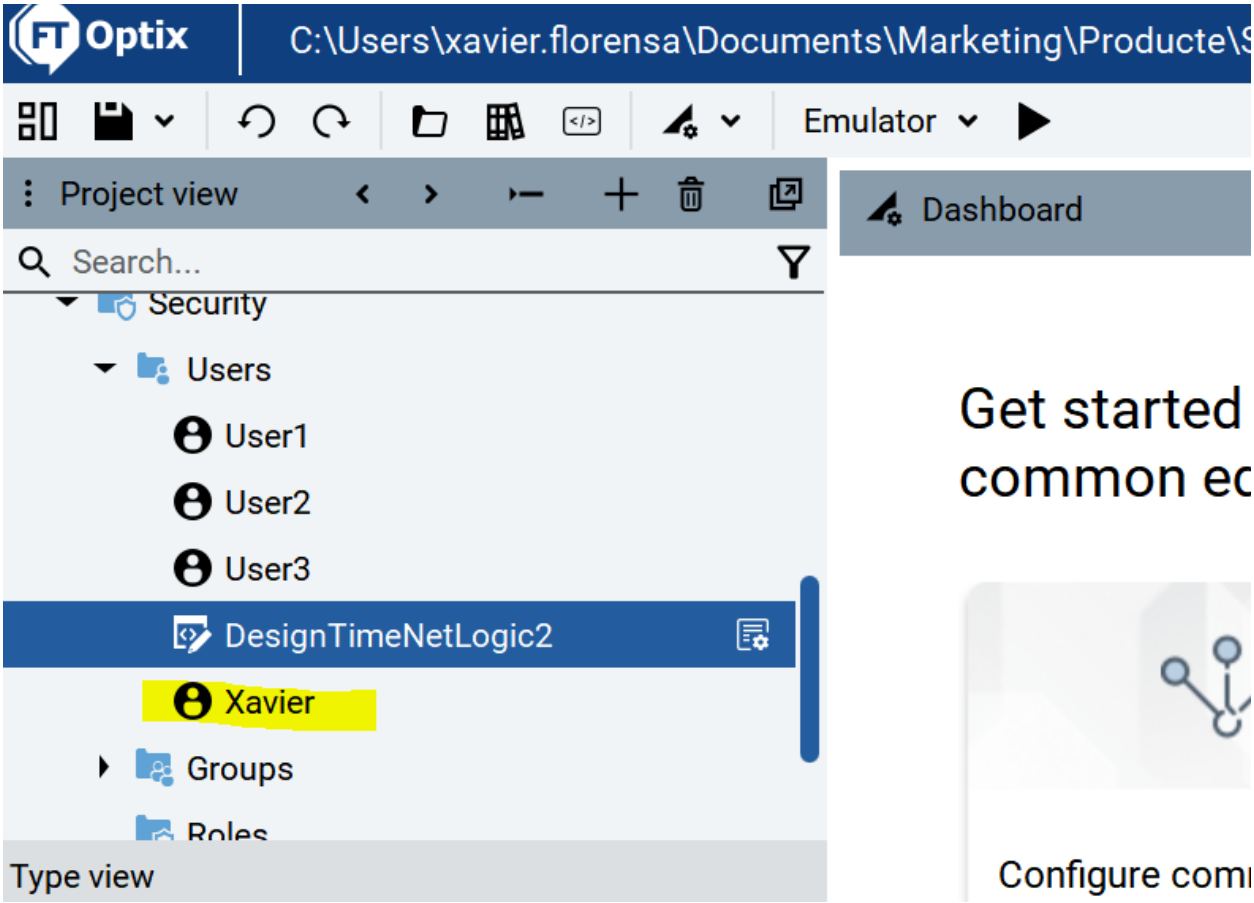
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.RAEtherNetIP;
using FTOptix.NativeUI;
using FTOptix.NetLogic;
using FTOptix.CoreBase;
using FTOptix.Retentivity;
using FTOptix.CommunicationDriver;
using FTOptix.Core;
#endregion

public class DesignTimeNetLogic2 : BaseNetLogic
```

```
{
    [ExportMethod]
    public void Method1()
    {
        // Insert code to be executed by the method
                // Insert code to be executed by the method
        Folder userFolder = Project.Current.Get<Folder>("Security/Users");
    if (userFolder == null) {
        Log.Error("CreateNewUser", "Cannot find Users folder");
        return;
    }
    /*else if (string.IsNullOrEmpty(groupName)) {
        Log.Error("CreateNewGroup", "Cannot create group with empty name");
        return;
    } else if (userFolder.Get<Group>(groupName) != null) {
        Log.Error("CreateNewGroup", "Group already exists!");
        return;
    }
    */
    // Create a new user
    var newUser = InformationModel.Make<User>("Xavier");
    // Add the user to the folder
    userFolder.Add(newUser);

    }
}
```

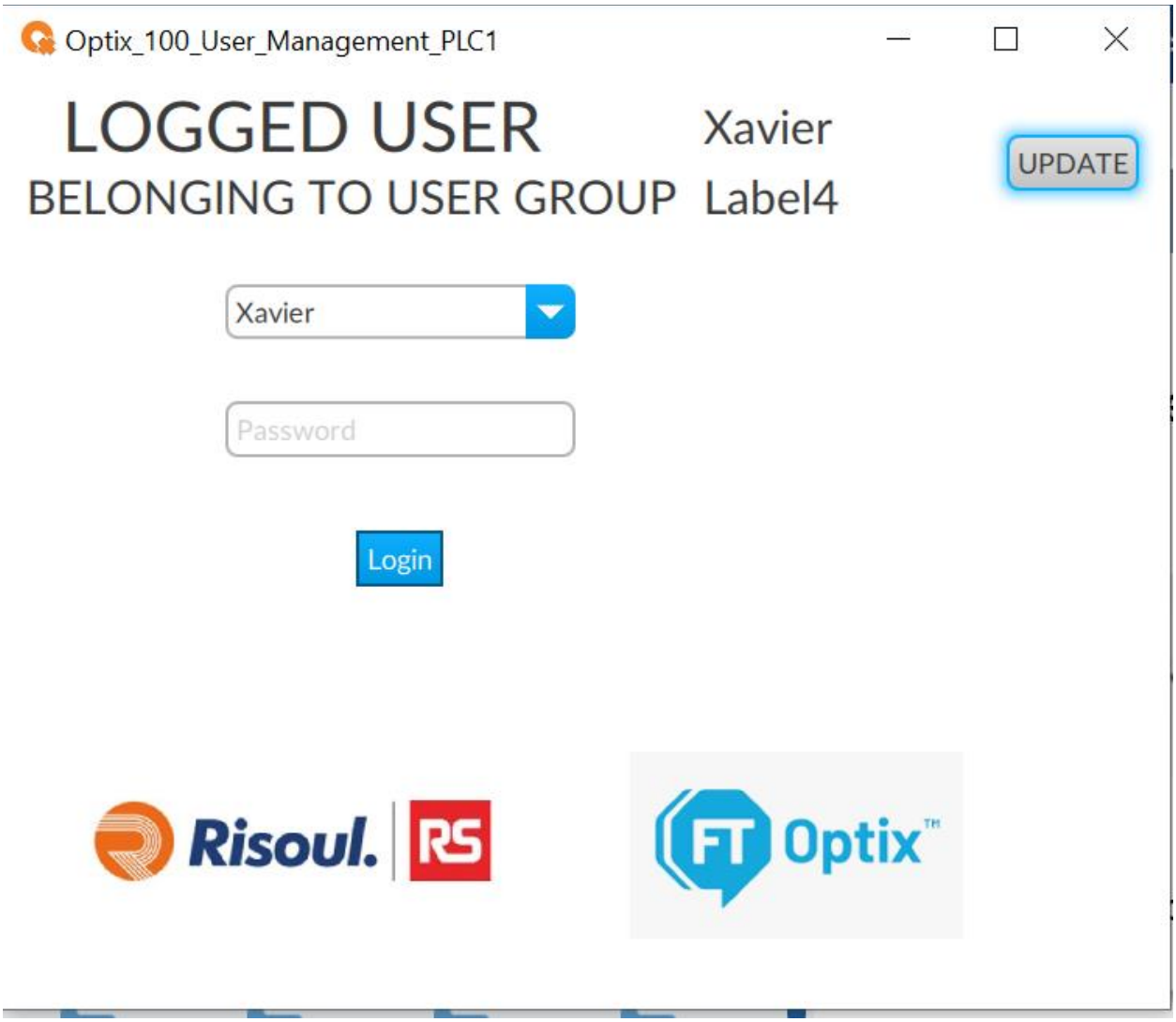
Now execute the design time script


Optix
C:\Users\xavier.florensa\Documents\Marketing\Producte\S
Emulator
Project view
Dashboard
Search...
Security
Users
User1
User2
User3
DesignTimeNetLogic2
Groups
Roles
Execute Method1
Edit with .NET code editor (external)
New
Get started
common ec

Voilà



Let’s test the application
We did not assign Xavier user to any group



14.6. User batch creation in design time (100 users)

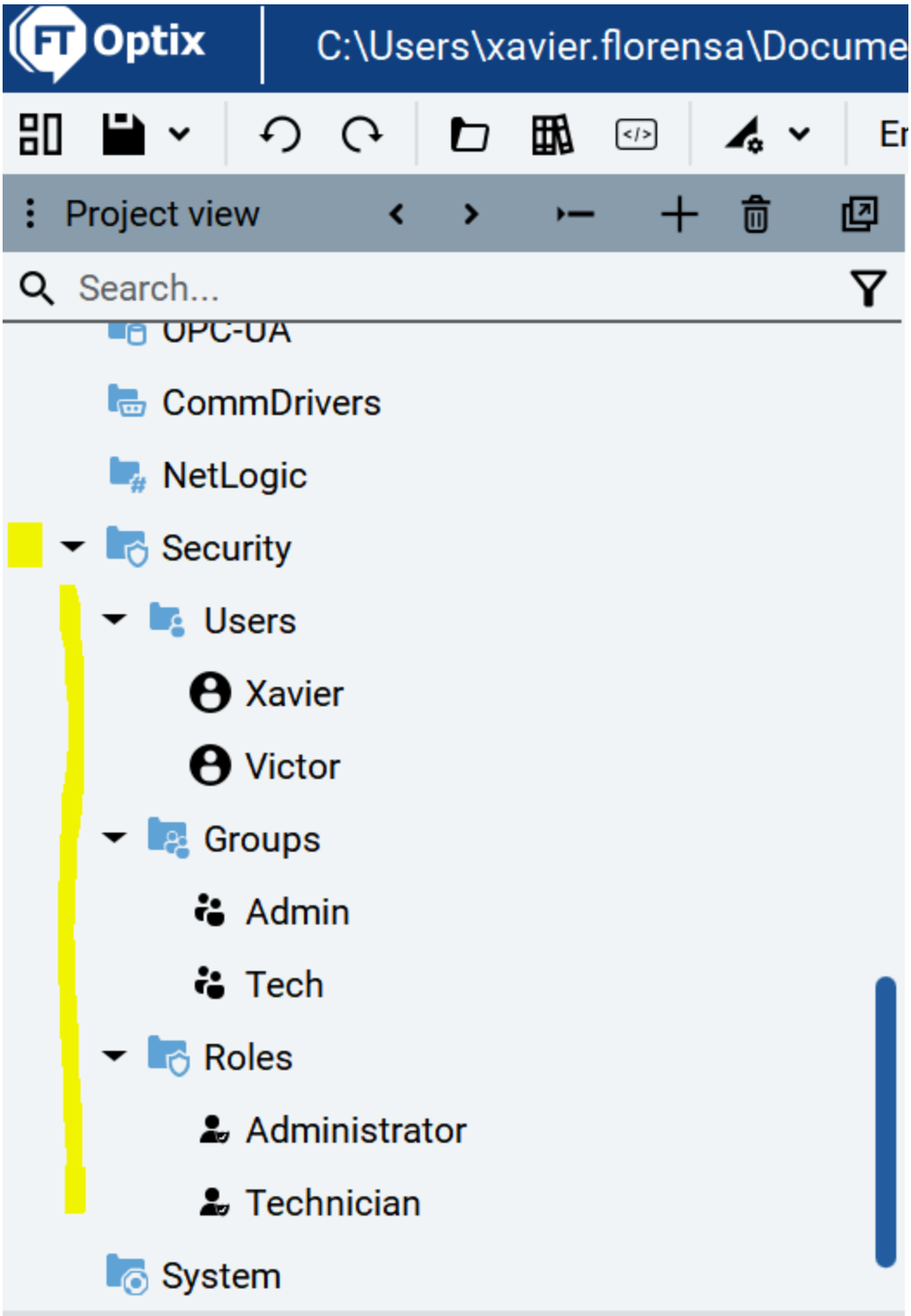
Now we can create 100 users as a batch task

As you can see in this video
https://youtu.be/wpdAVwmzsEs

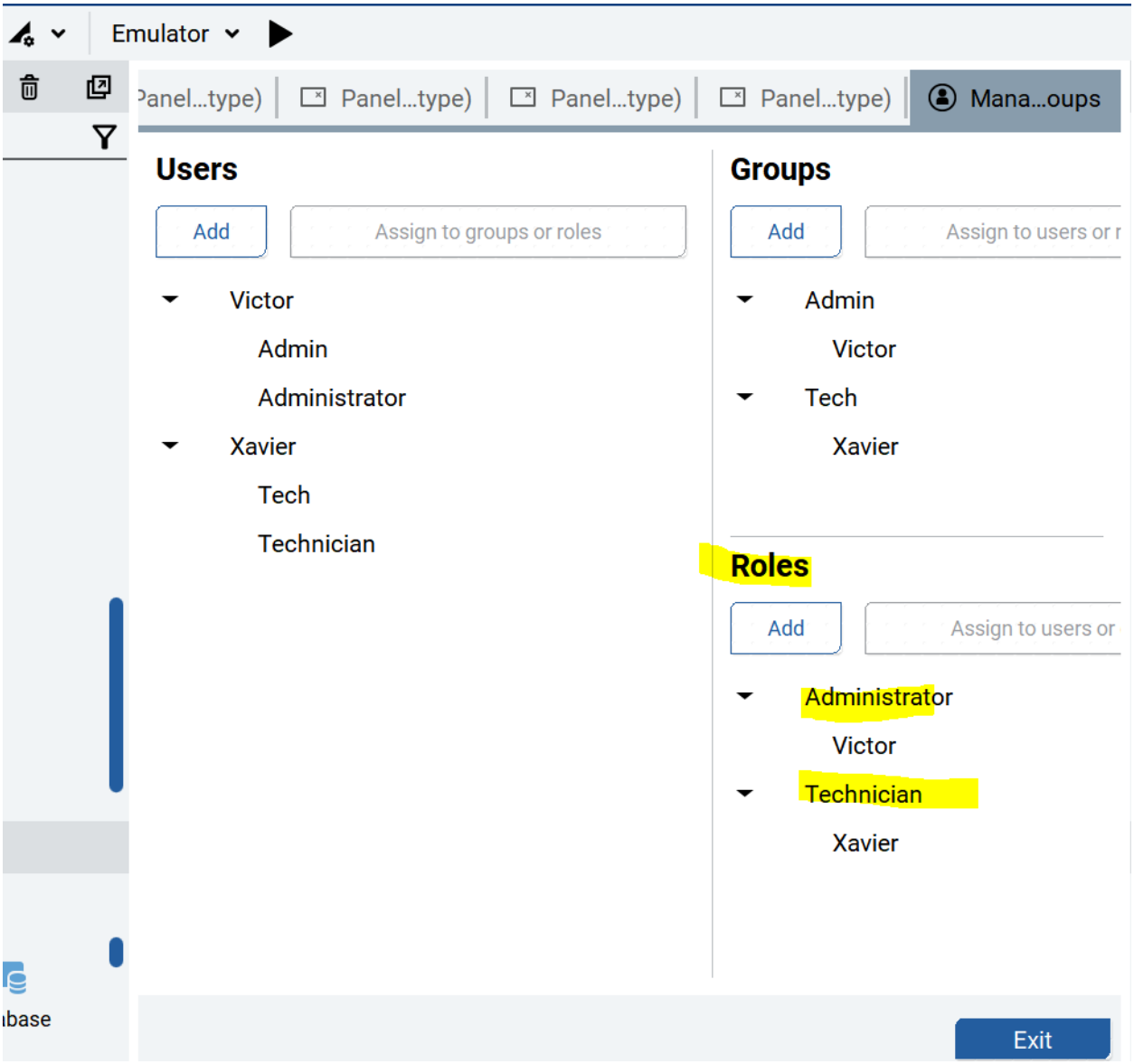
The code is located here
https://github.com/xavierflorensa/FTOptix_100_User_Management_PLC

14.7. Manage user rights to Panels in a Native way

Create your Users, Groups and Roles.

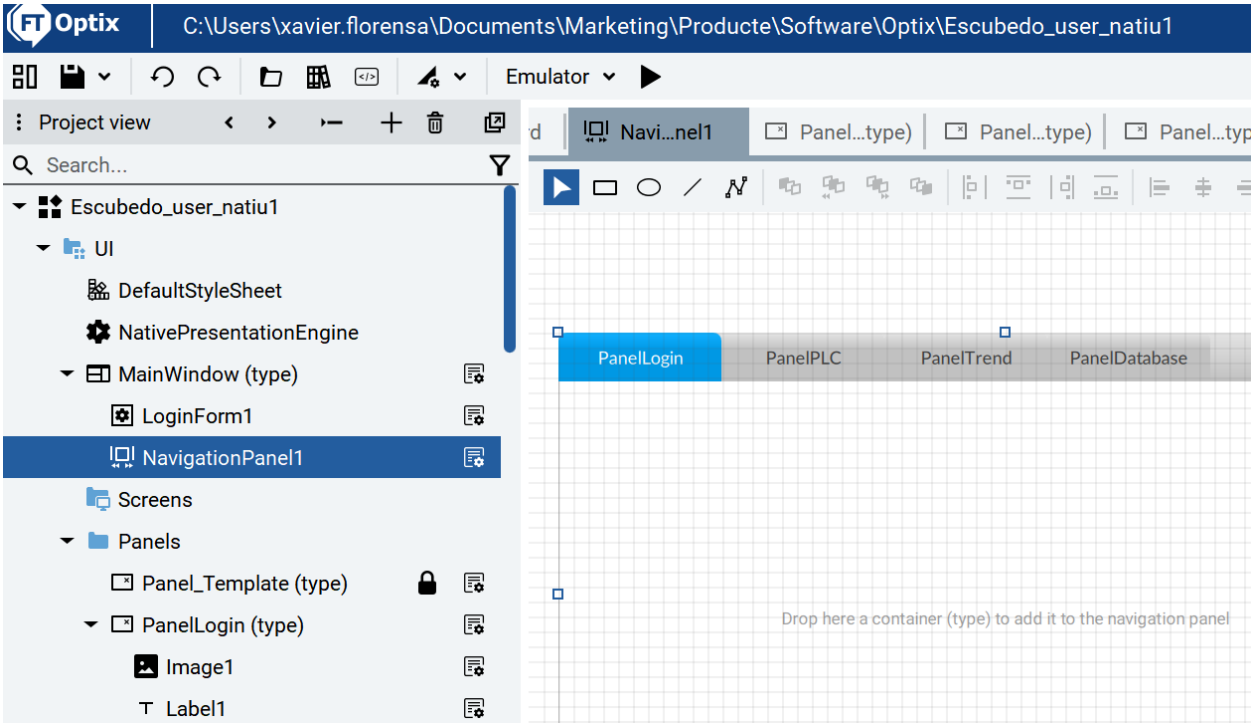


Configure different roles to different users

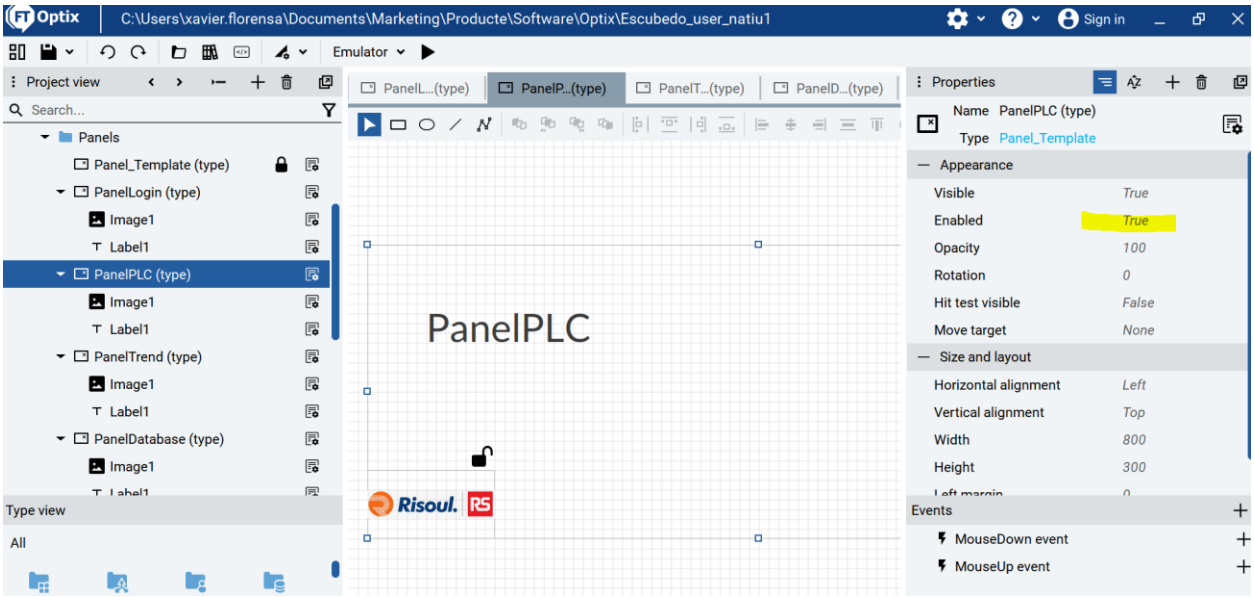


Insert your Login widget as explained early on this chapter

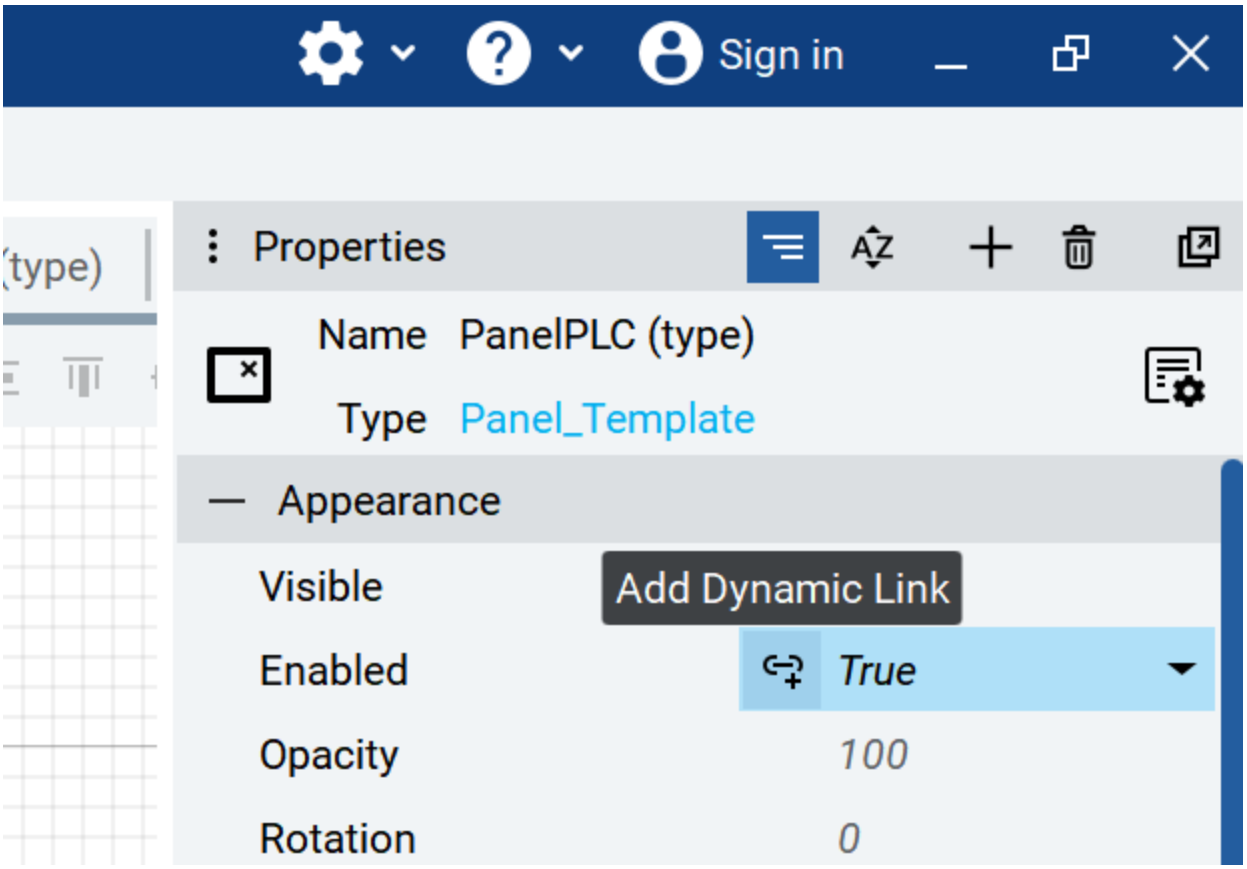
Create your Navigation Panel and Panels



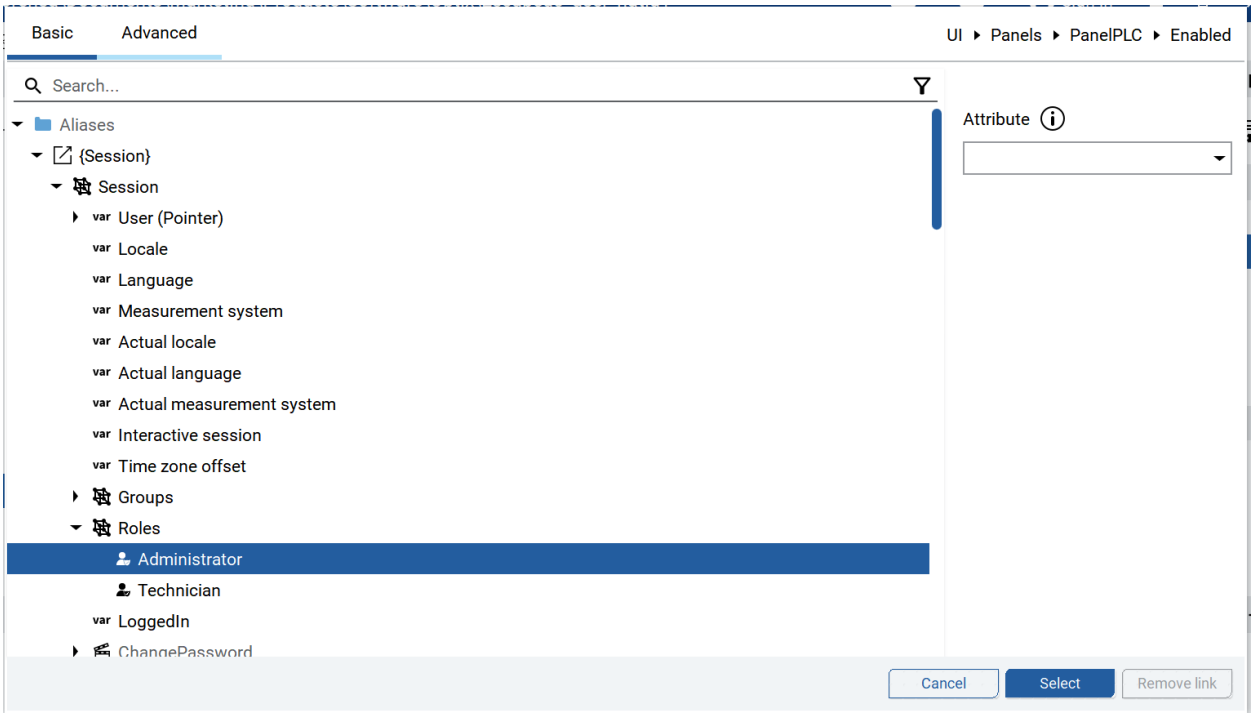
Select your Panel (Do this for all Panels) and setup the dynamic link on Enabled

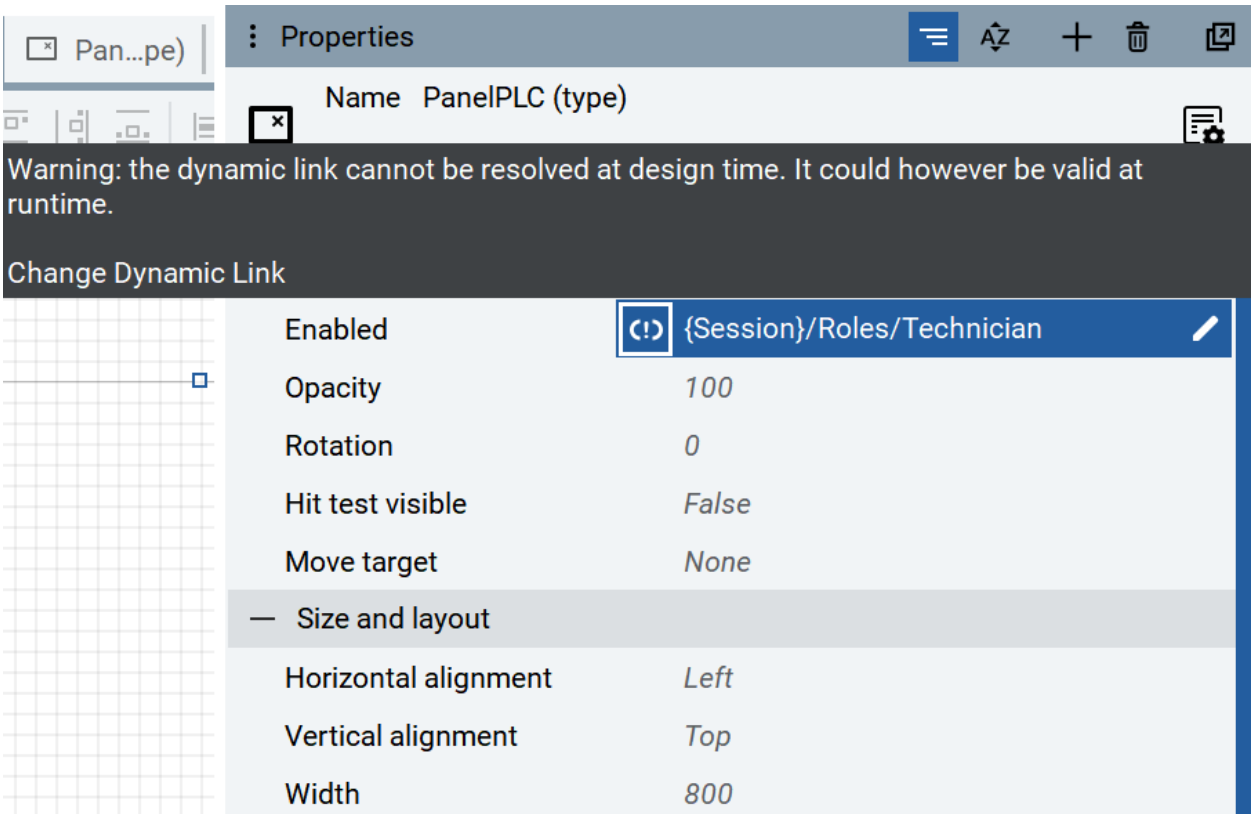


Add a dynamic link to Enabled



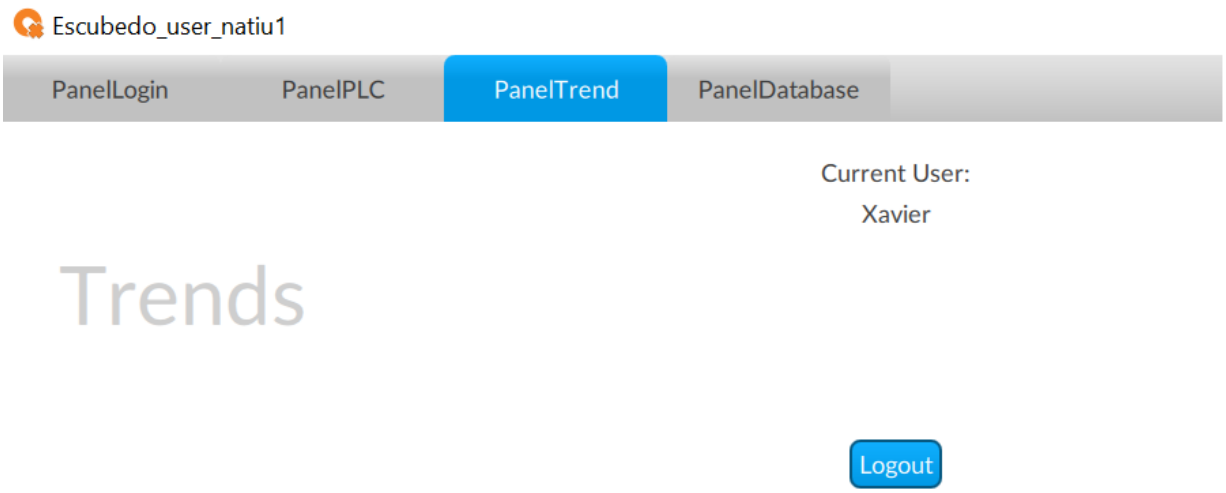
Navigate until
Select the desired Role under Aliases / Session / Session / Roles



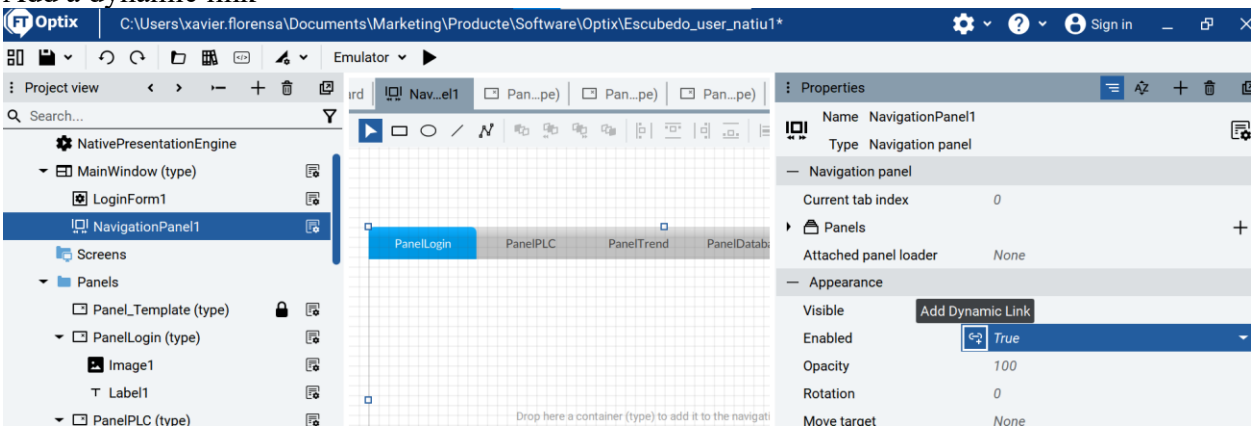


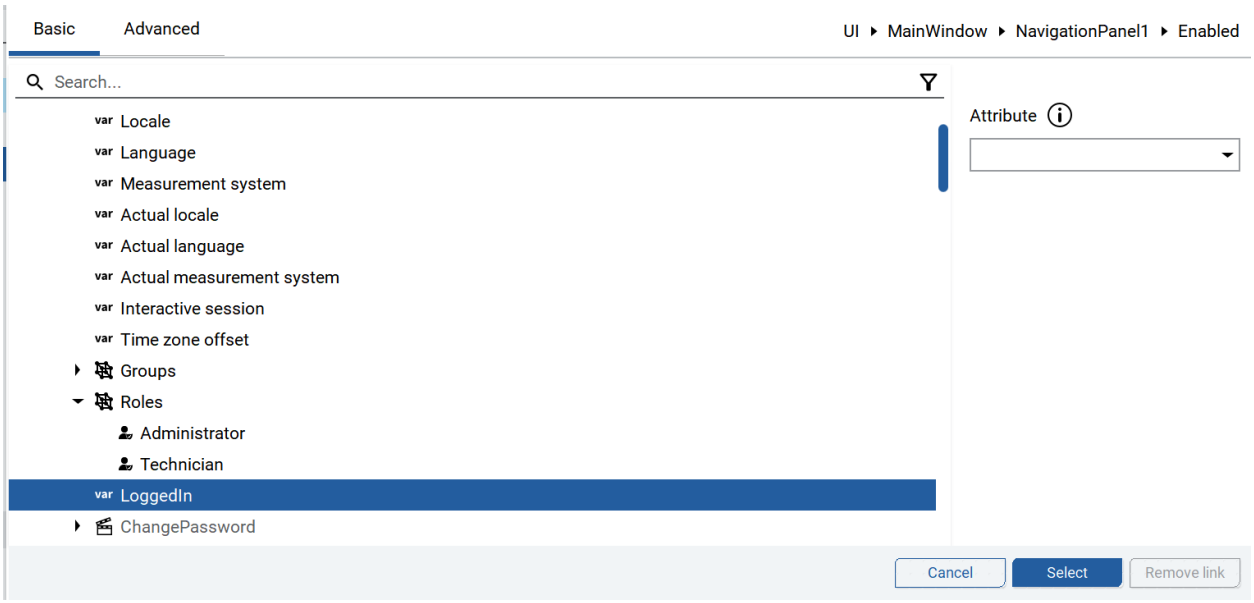
That's all

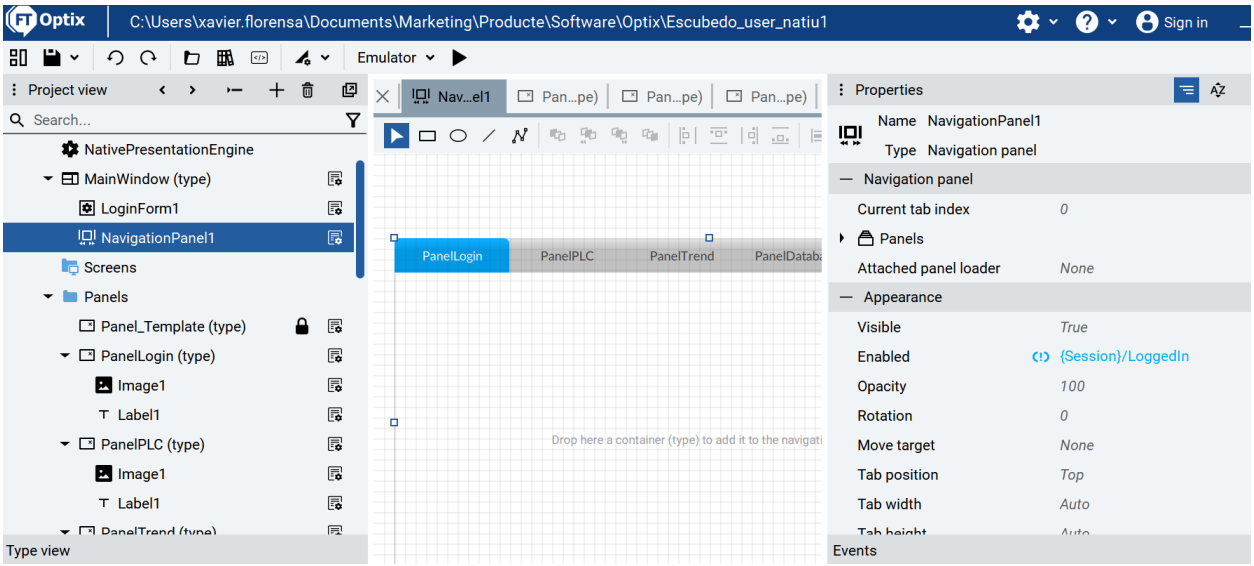
When you execute and log in you will be the Panel disabled, as you can see on the picture with difuminated intensity on the contents



You can make the pannels not enabled before any user is logged this way
Add a dynamic link

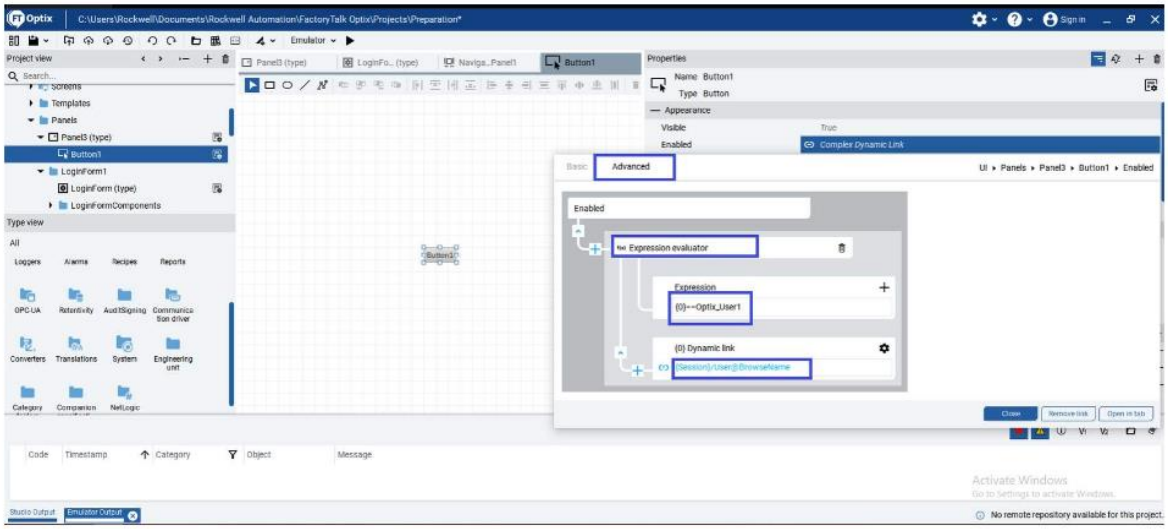






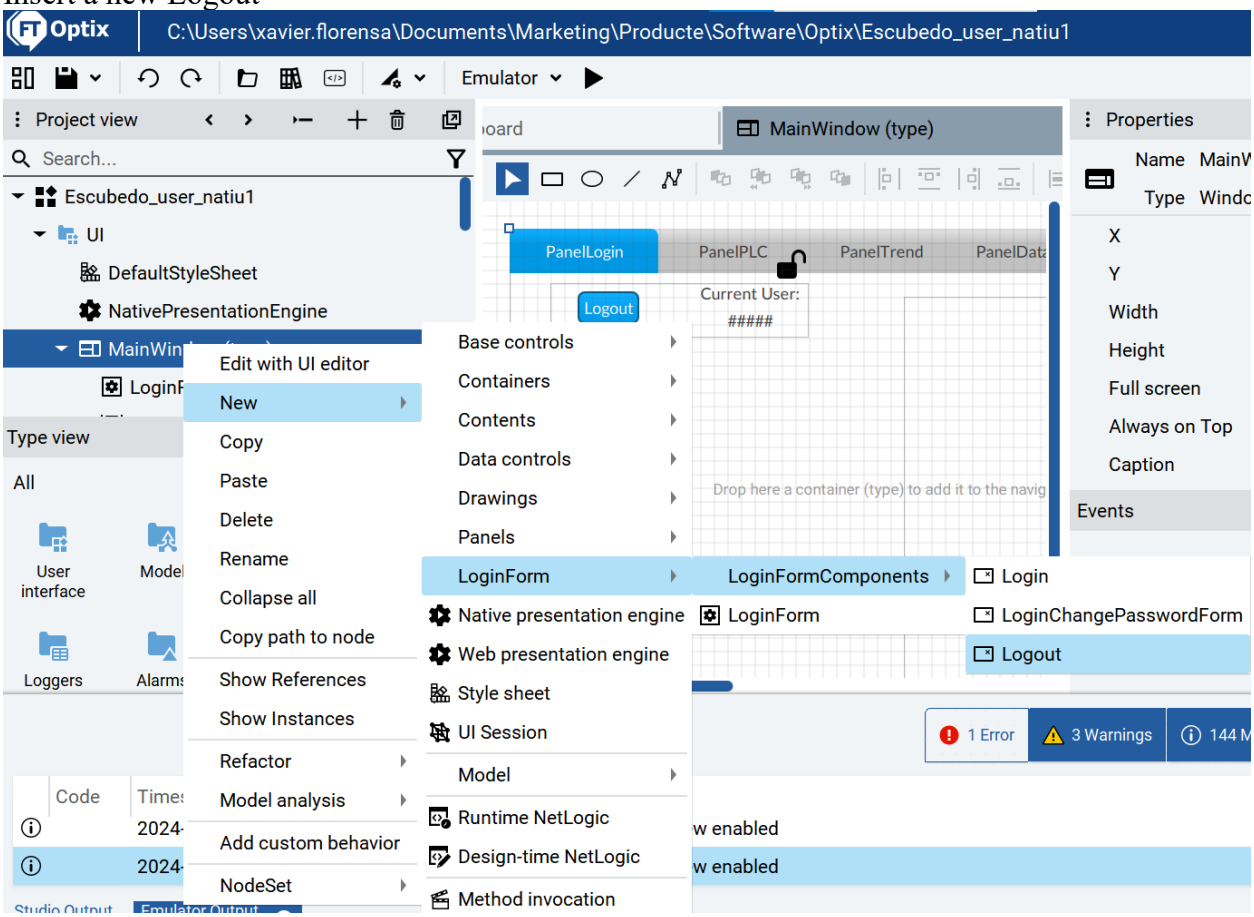
That's all
You can find the code here
https://github.com/xavierflorensa/FTOptix_user_management_panels_native

You can also use advanced dynamic link

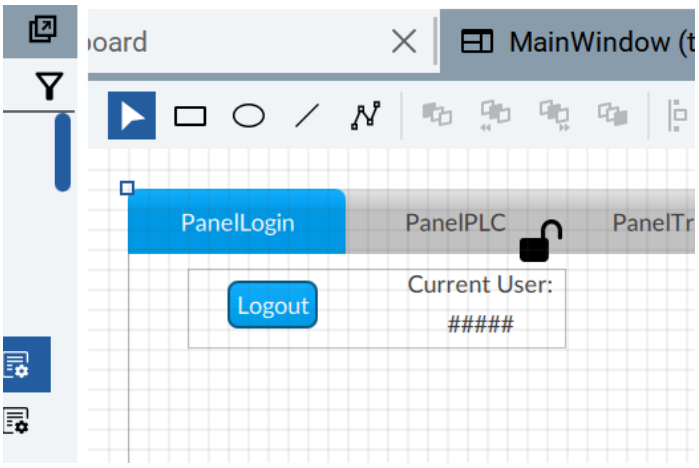


14.8. Making Login Form invisible when a user is Logged in

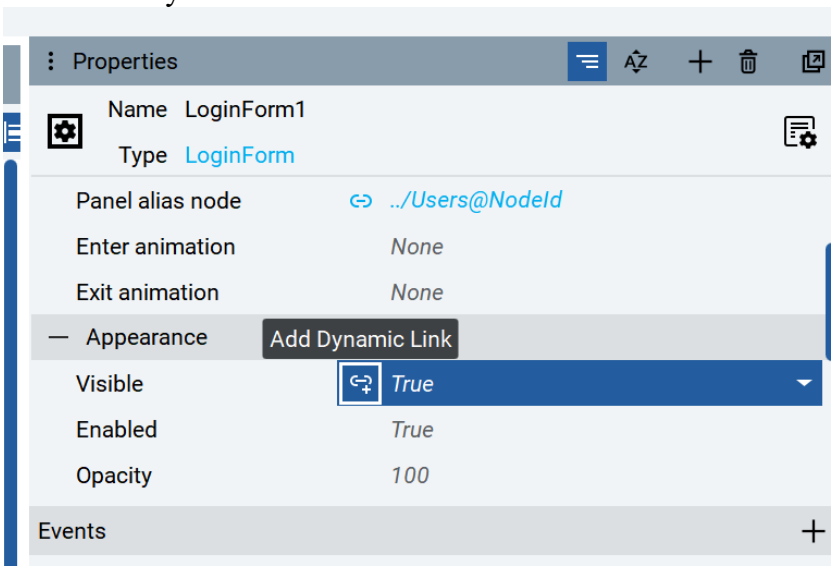
First of all you have to grant you have a logout button to perform a Logout
Insert a new Logout



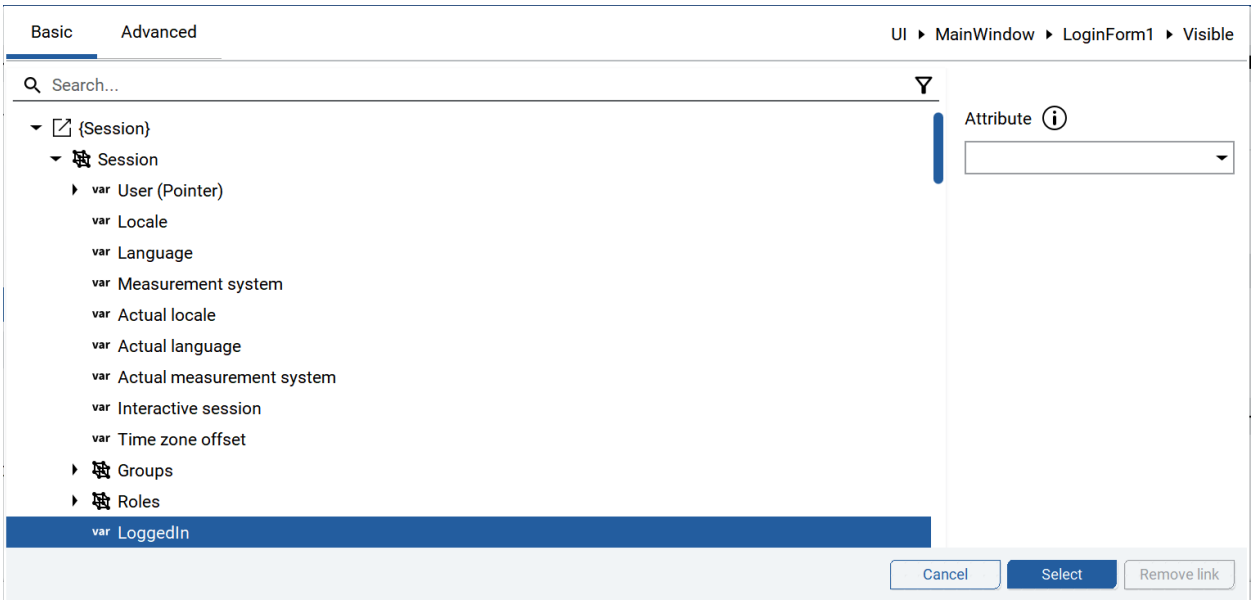
Make it little
Clicking on the padlock to be able to modify it



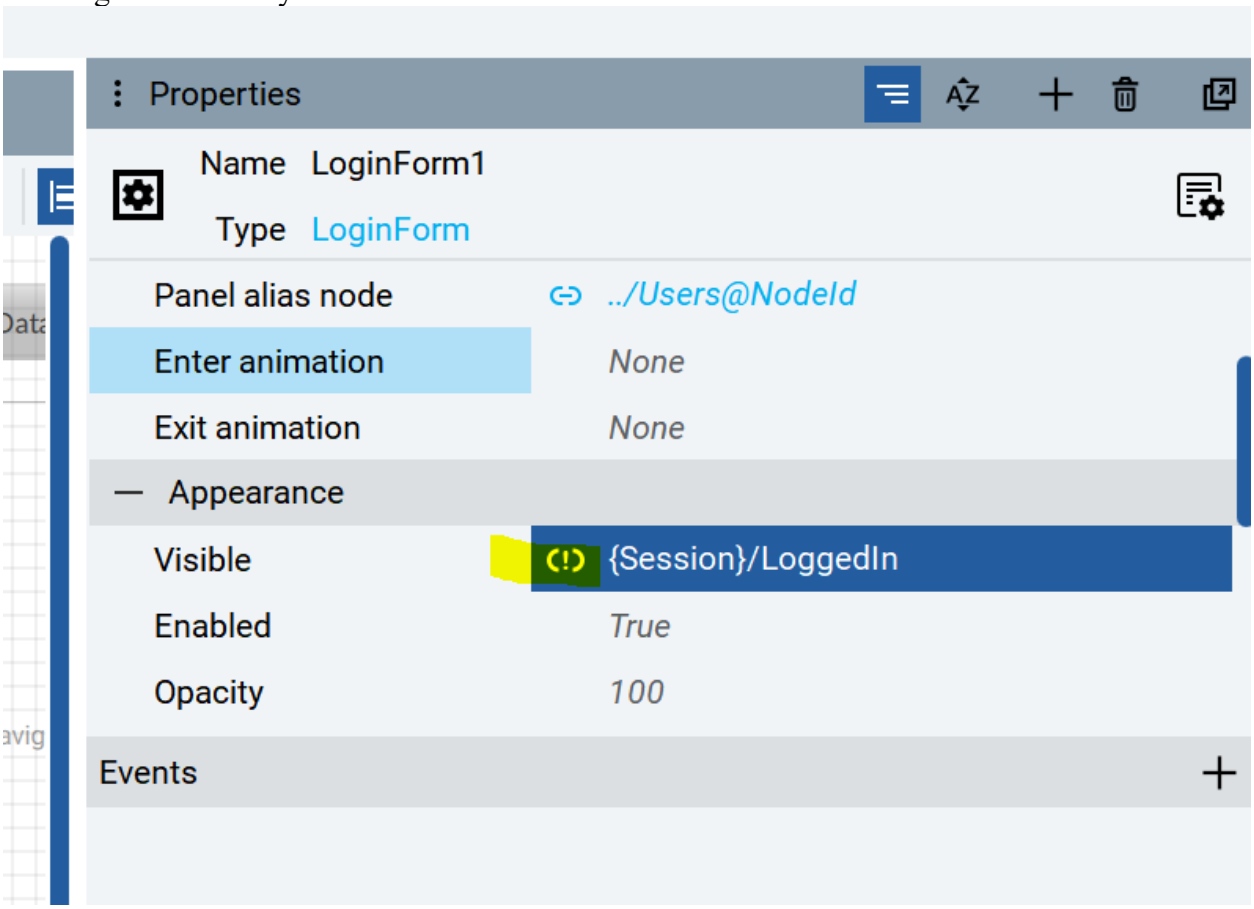
Then apply to the Login visible property an advanced condition like
The equivalent to Visible = NOT (LoggedIn)
Add a new dynamic link



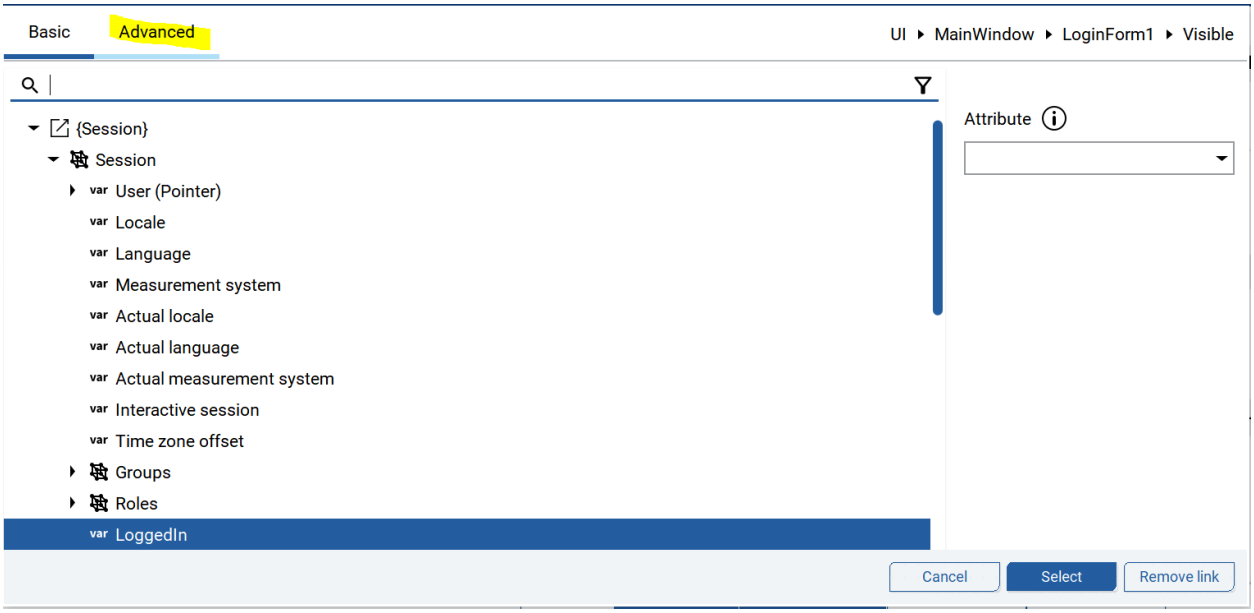
Place the cursor over LoggeId and select



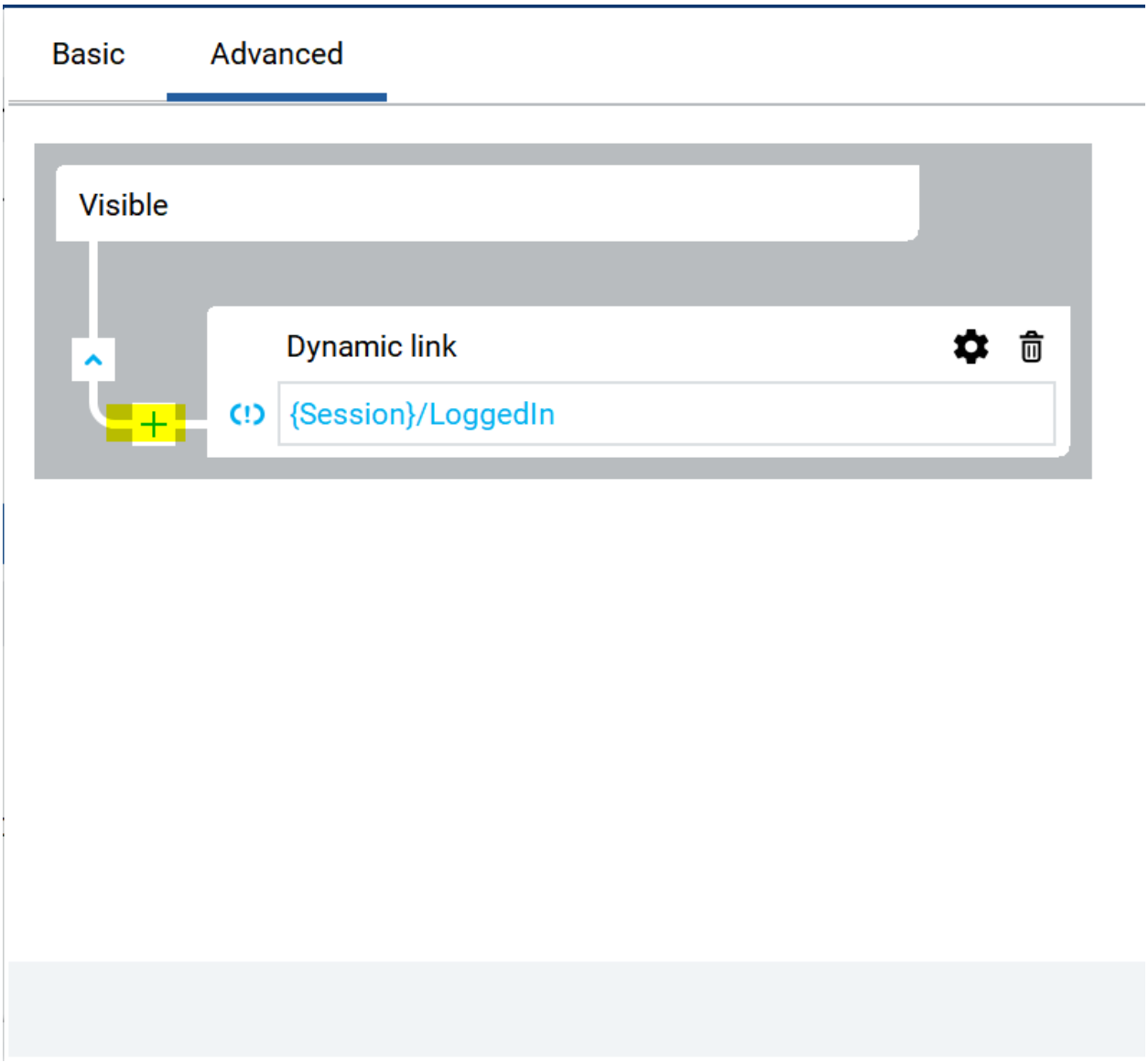
Enter again into the dynamic link to edit

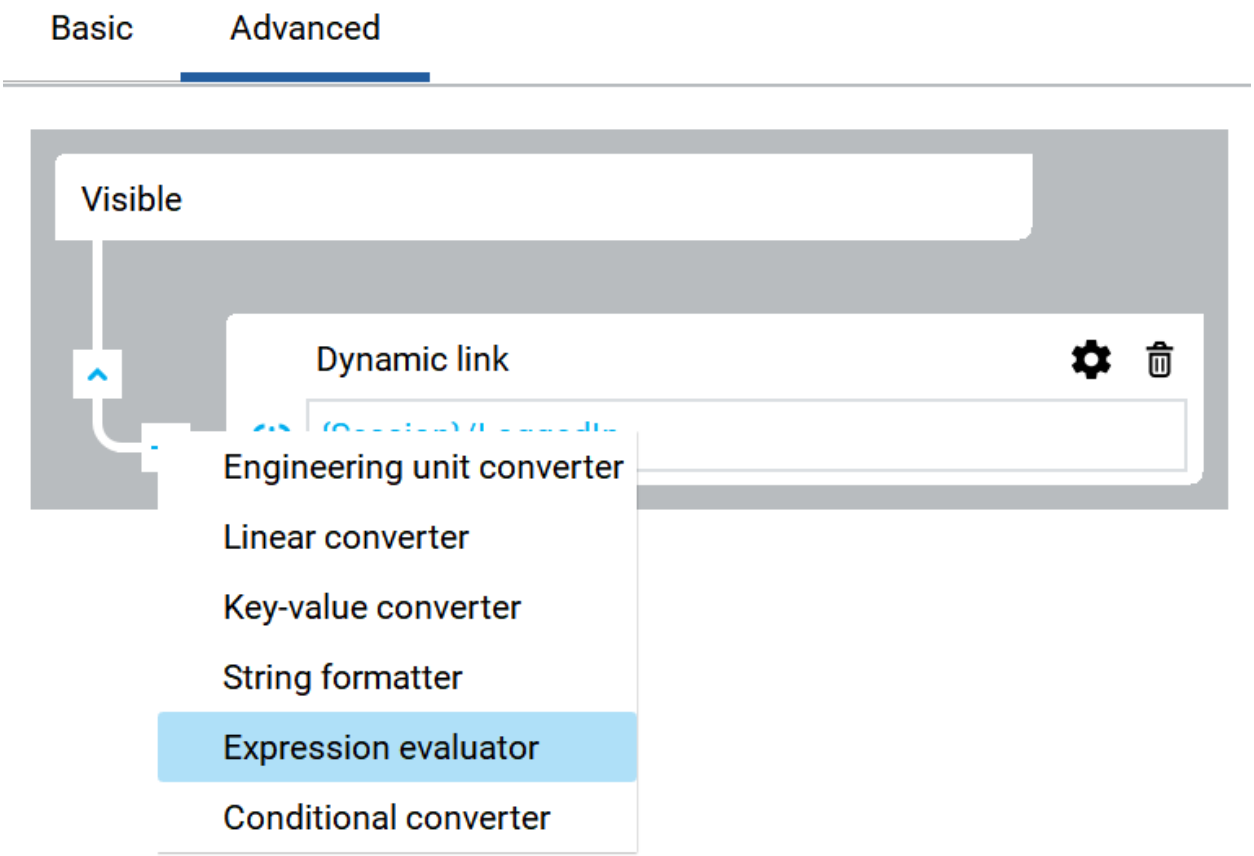


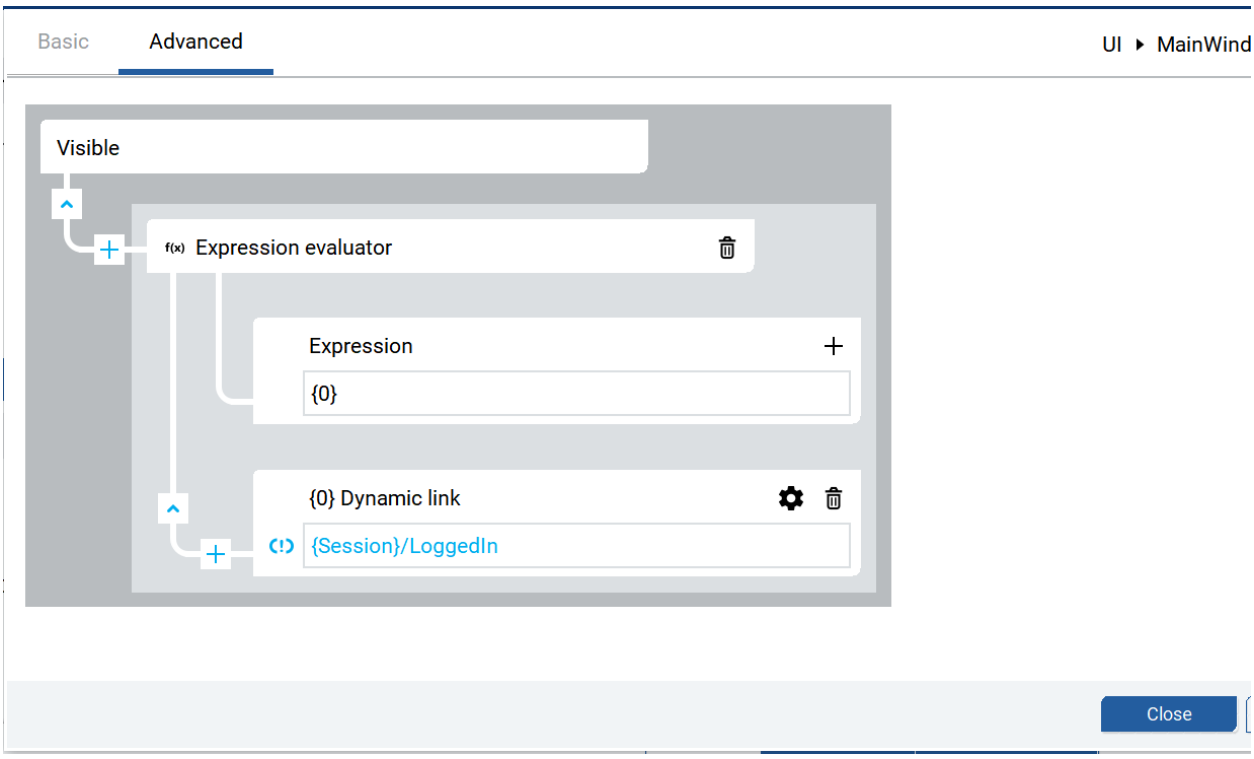
Go to Advanced



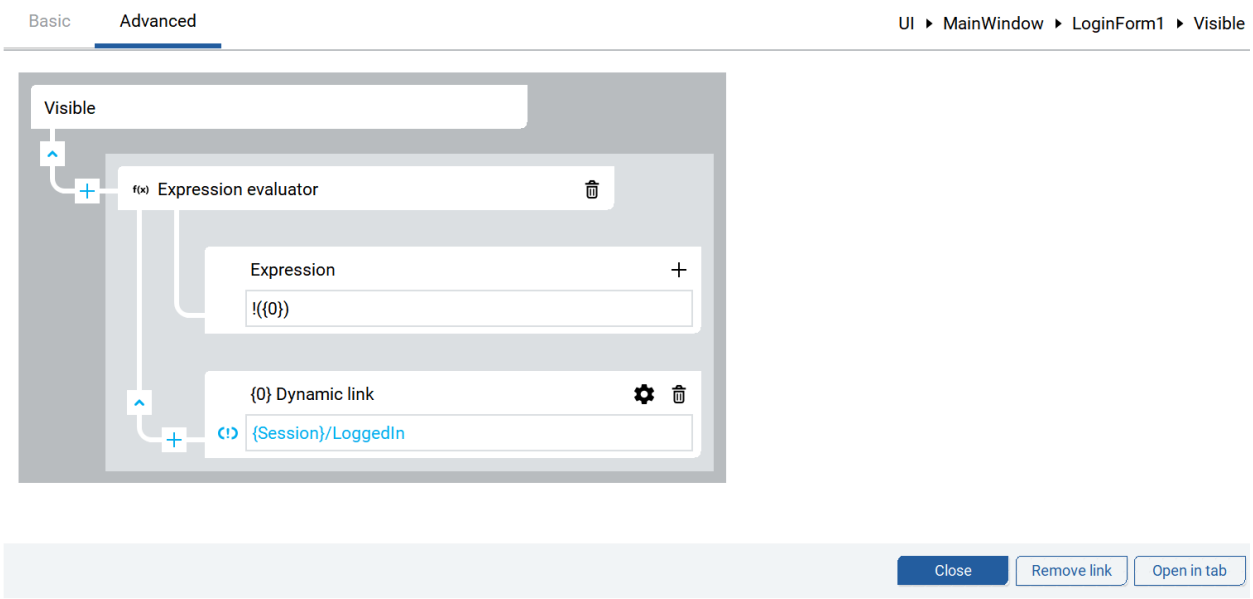
Add a Expression evaluator


Basic
Advanced
Visible
Dynamic link
{Session}/LoggedIn


Basic
Advanced
Visible
Dynamic link
Engineering unit converter
Linear converter
Key-value converter
String formatter
Expression evaluator
Conditional converter



Edit the expression this way !({0})



That's all

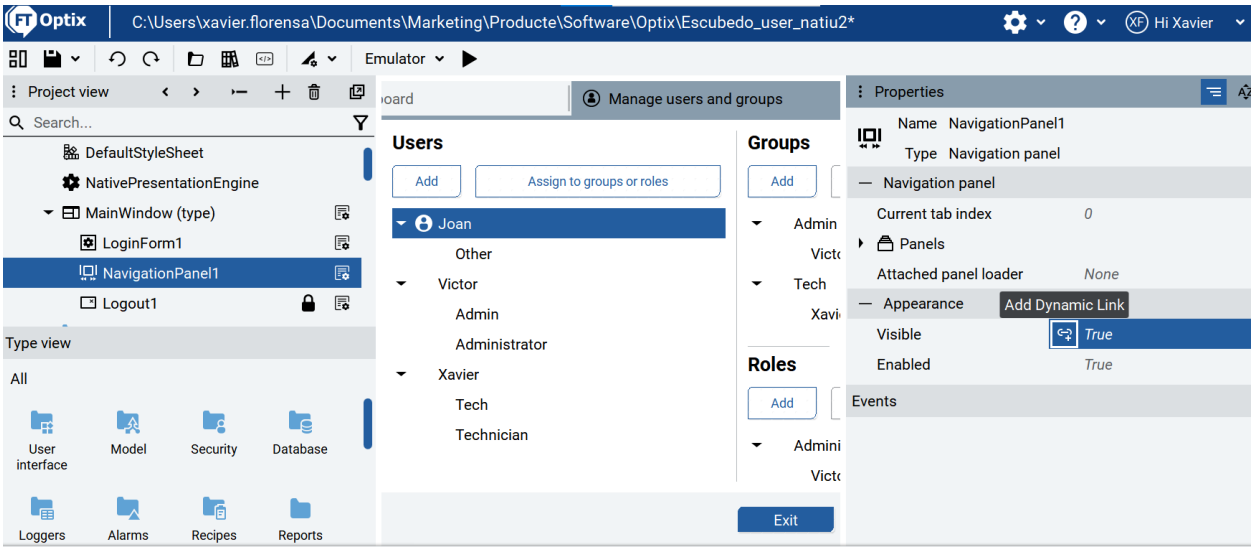
14.9. Make Panels visible if user belongs to one of a set of Roles

Imagine you want to exclude a user is he does not belong to any of a setr of roles.

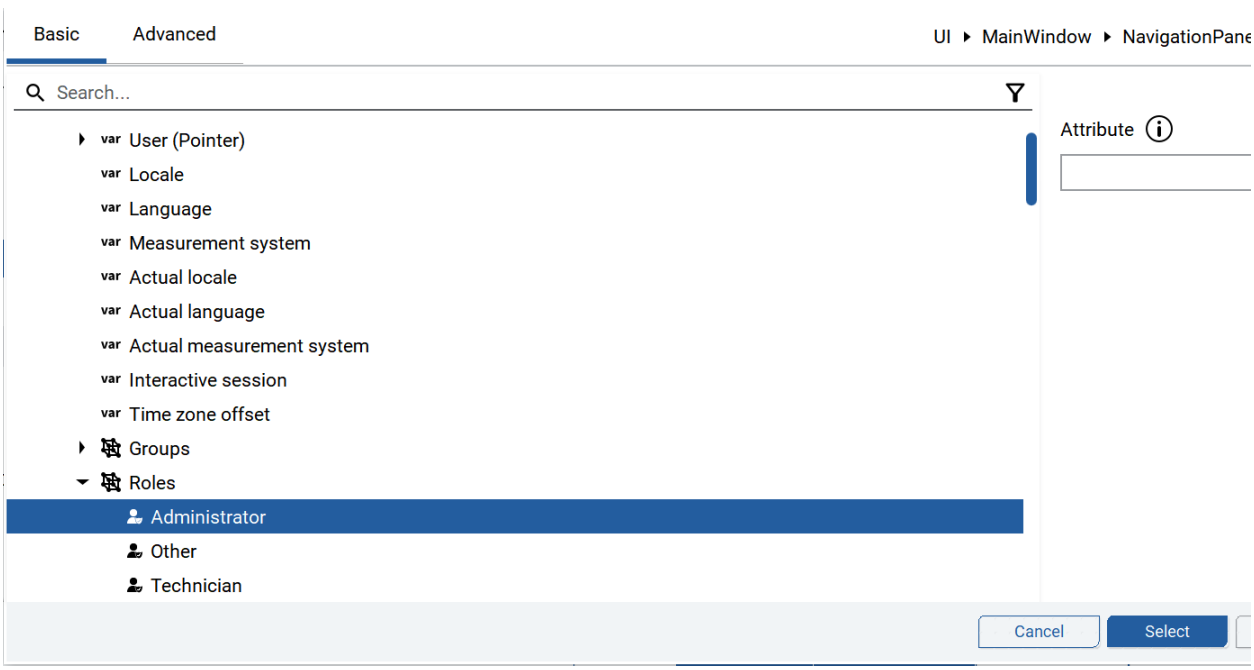
User Joan has Role Other, which is not Administrator or Technician Role.

Then you can do this way

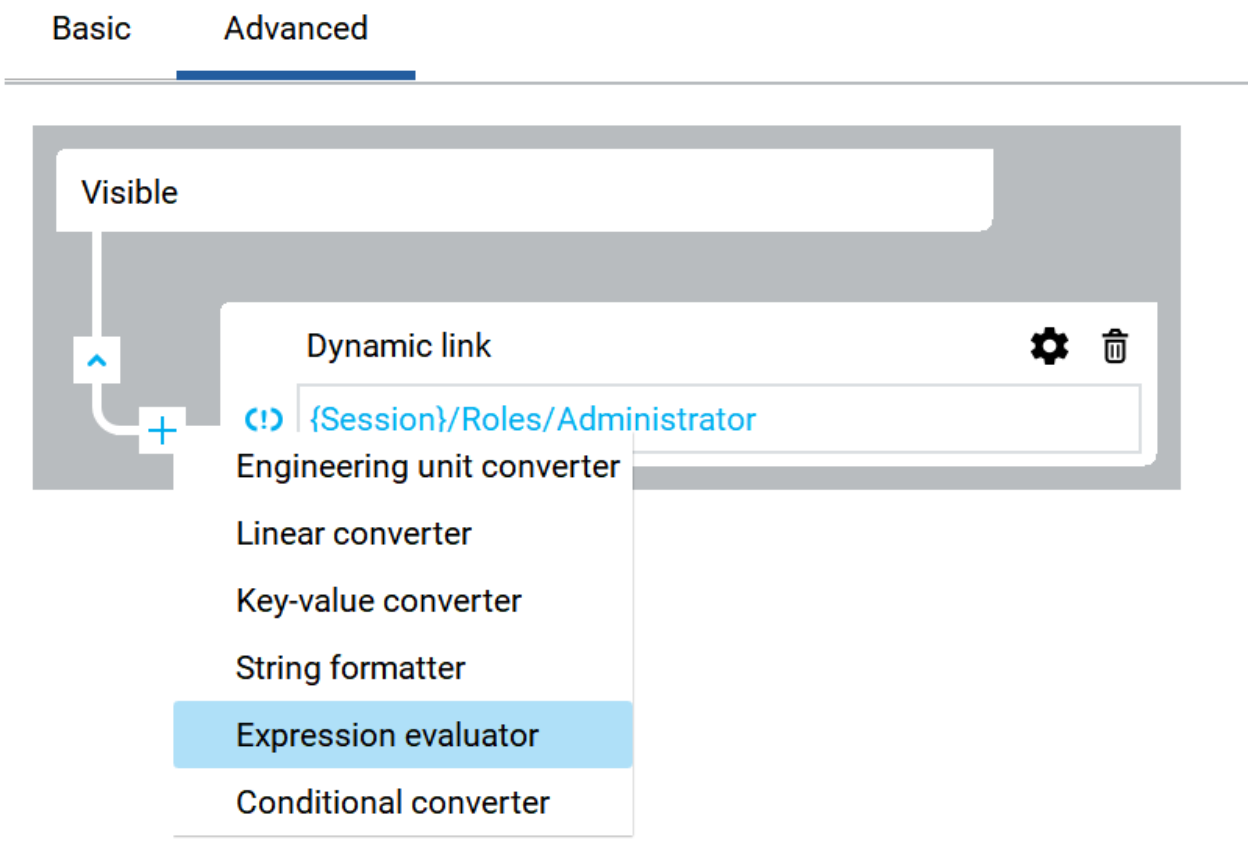
Add a new dynamic link to Navigation Panel property

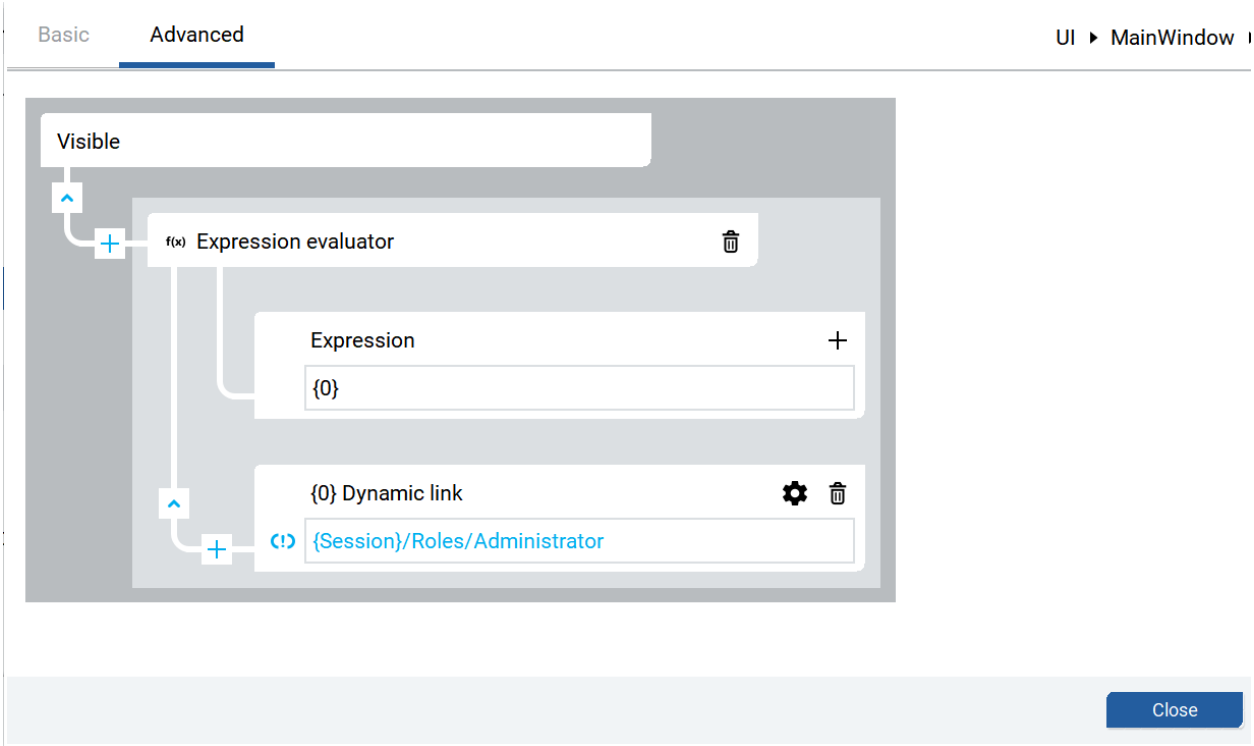


Highlight Administrator and lick on Select



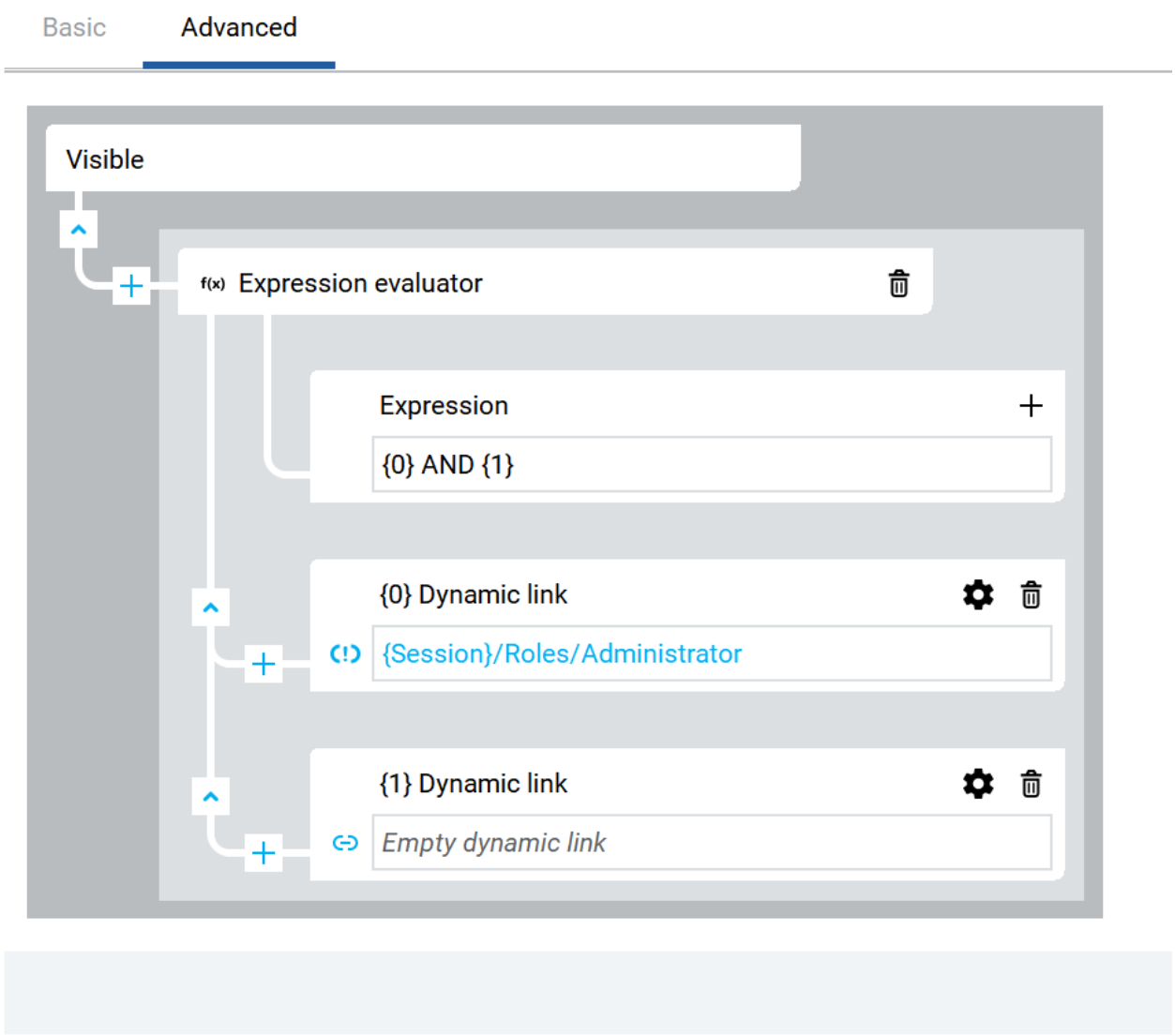
Open the dynamic link again to edit it on Advanced
Add an expression evaluator


Basic
Advanced
Visible
Dynamic link
{Session}/Roles/Administrator
Engineering unit converter
Linear converter
Key-value converter
String formatter
Expression evaluator
Conditional converter

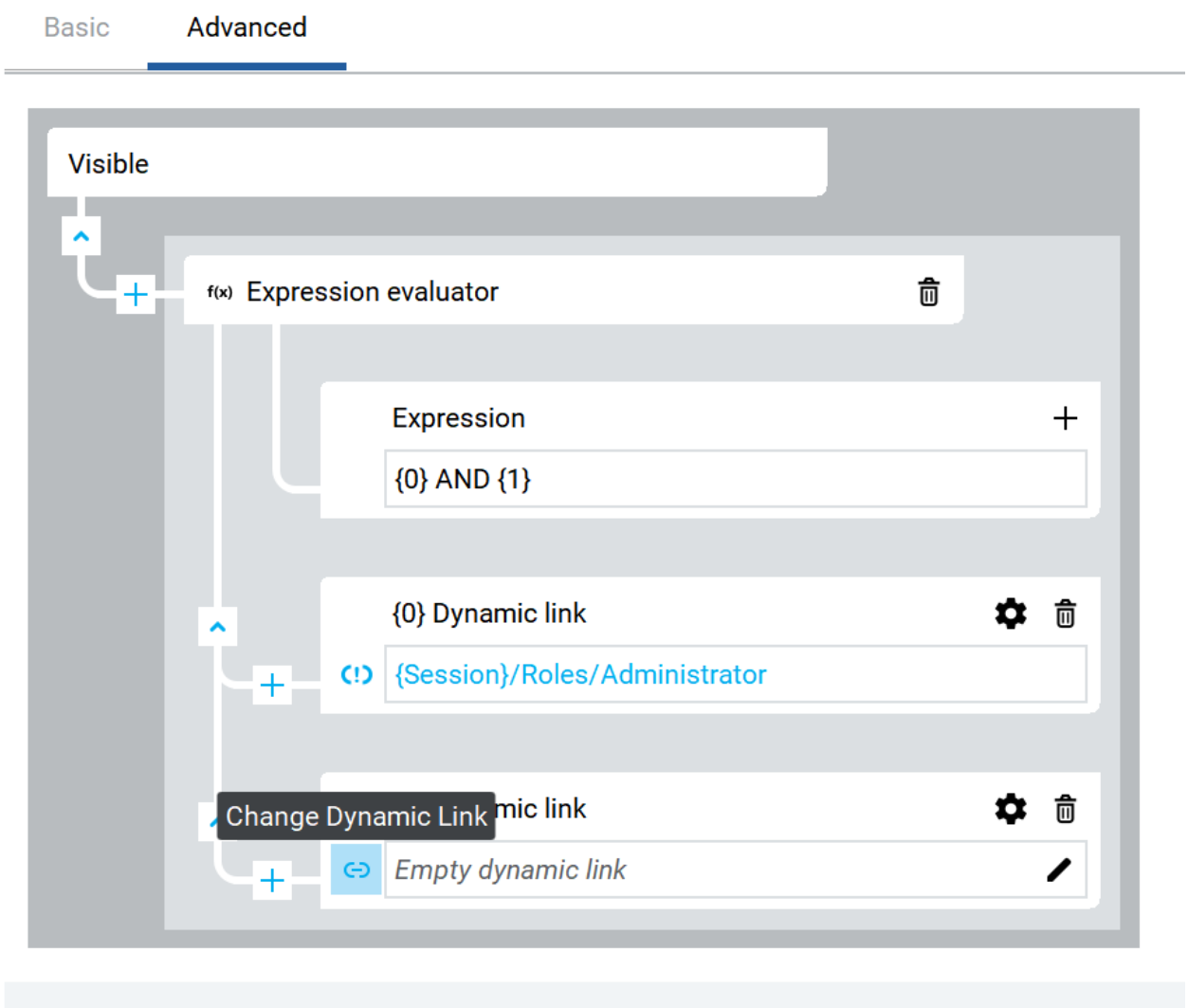


Write {0} AND {1} on Expression

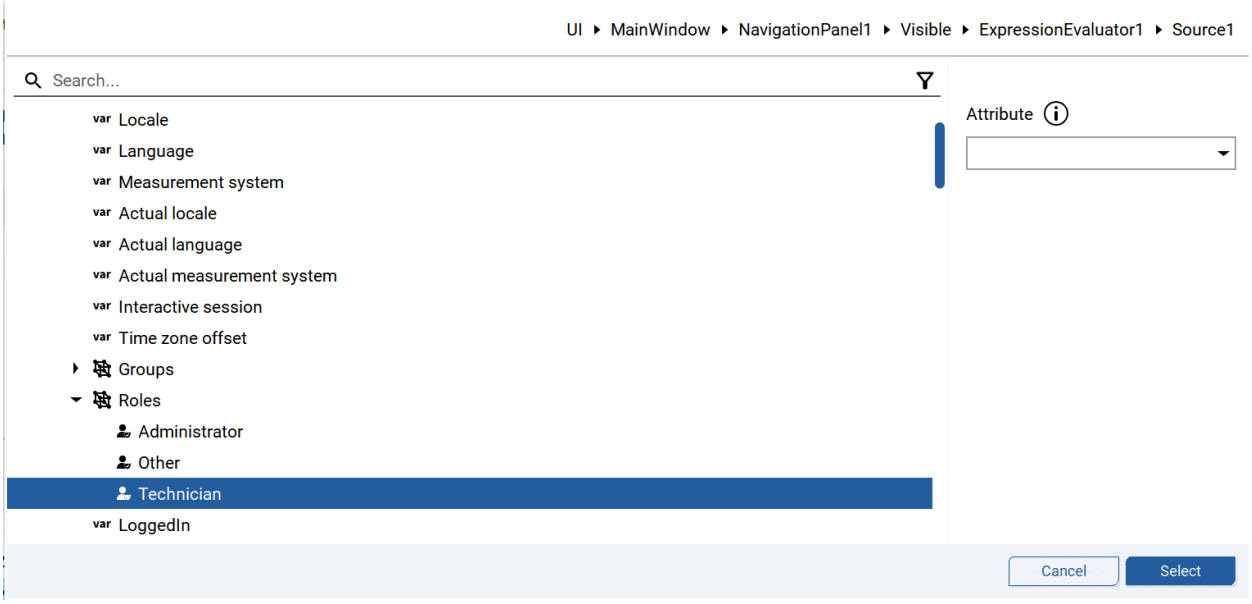
A new link will appear

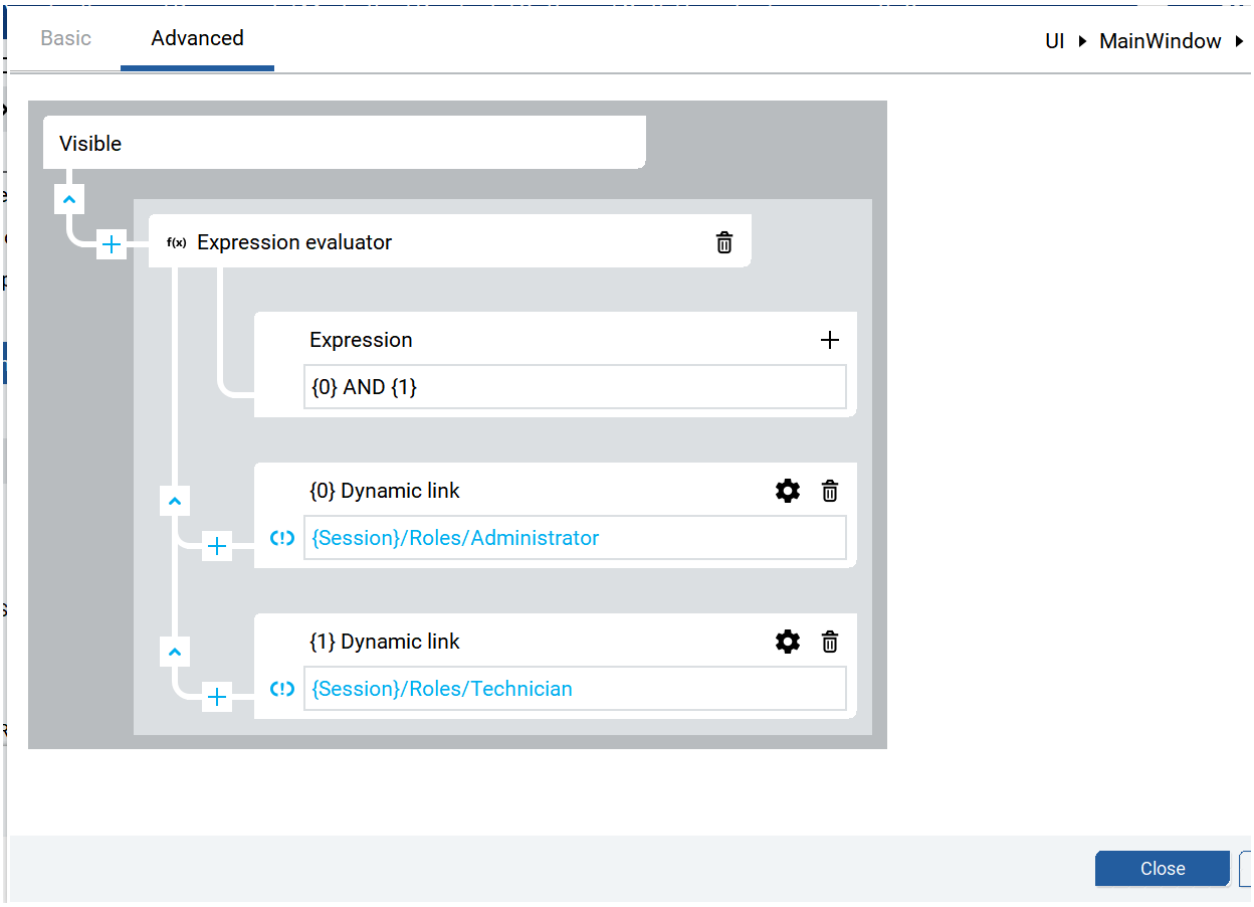


Select the “Technician” role clicking on the dynamic link



Point to technician role, and select


UI ▸ MainWindow ▸ NavigationPanel1 ▸ Visible ▸ ExpressionEvaluator1 ▸ Source1
Search...
Attribute
var Locale
var Language
var Measurement system
var Actual locale
var Actual language
var Actual measurement system
var Interactive session
var Time zone offset
Groups
Roles
Administrator
Other
Technician
var LoggedIn
Cancel
Select

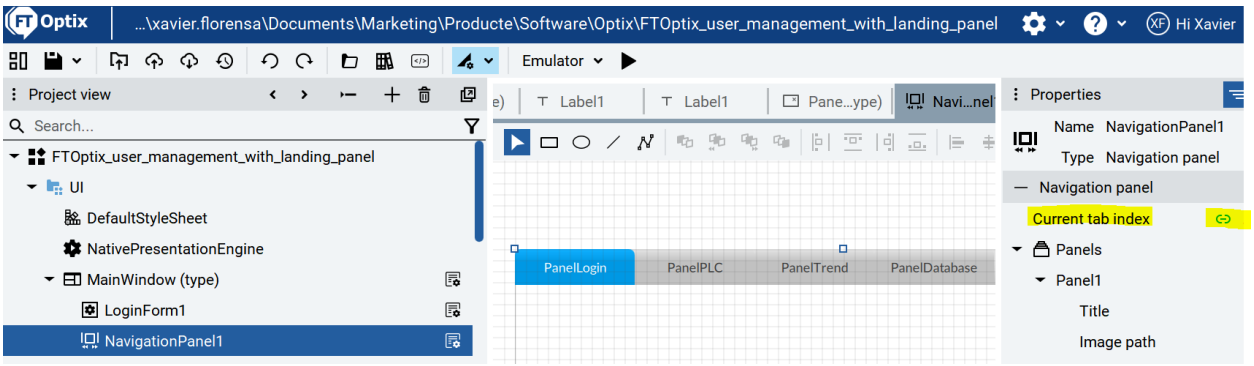

Basic
Advanced
UI ▸ MainWindow ▸
Visible
Expression evaluator
Expression
{0} AND {1}
{0} Dynamic link
{Session}/Roles/Administrator
{1} Dynamic link
{Session}/Roles/Technician
Close

That’s all

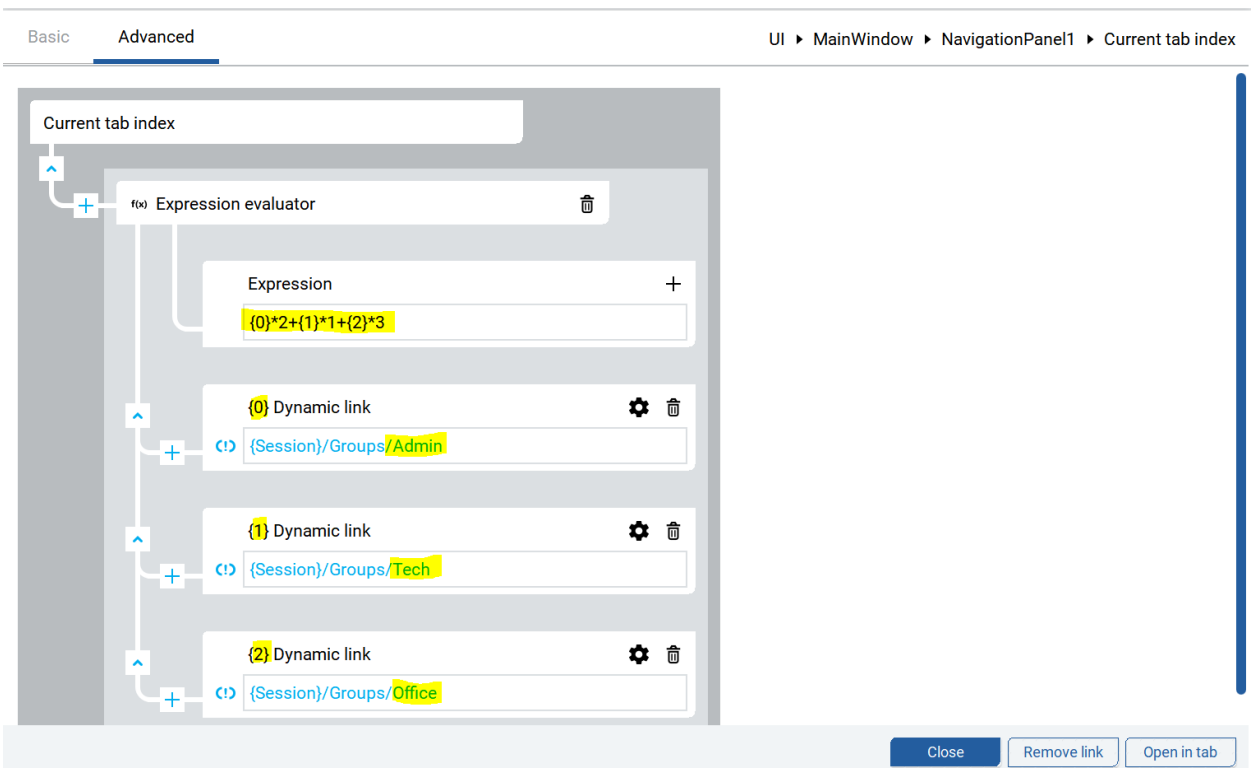
14.10. User management with landing panel

We want that after logging in, depending on the group, the first panel to open is different.

You have an index (0,1,2..) for that purpose on the Navigation Panel properties.
By default is 0, corresponding to Panel1



Let's introduce a complex dynamic link there with expression evaluator
With an artimetic addition wich result will be a number (1, 2 or 3).
So Admin group will open Panel number 2
Tech group will open Panel Number 1
And finally Office group will open Panel number 3



You can see the final result here

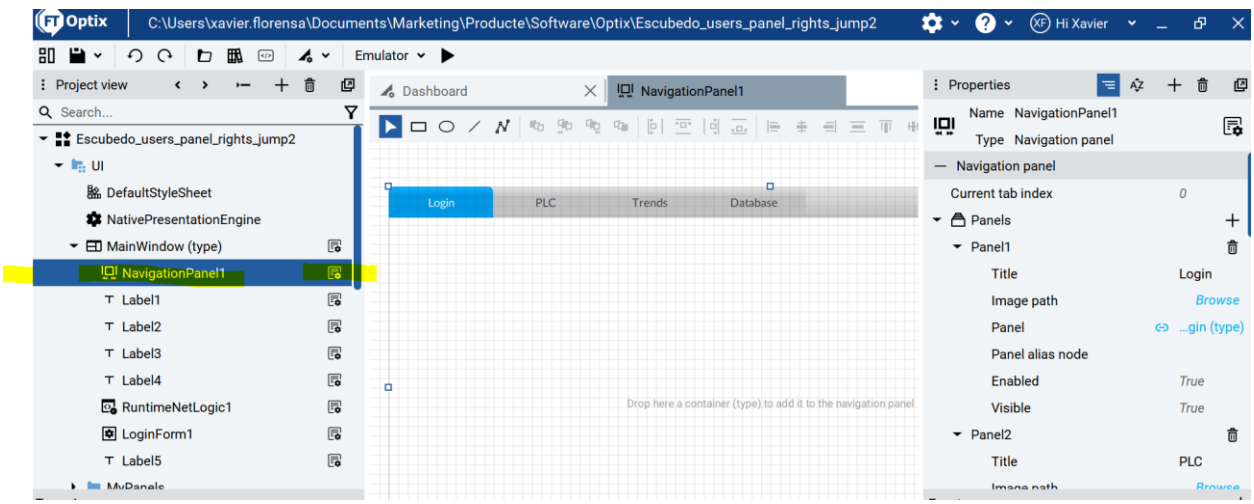
https://youtu.be/uUqUdoh2VVc

You can find the code here
https://github.com/xavierflorensa/FTOptix_user_management_with_landing_panel.git

14.11. Manage user rights to Panels on a NavigationPanel in C#

You can see the results on this video
https://youtu.be/bp9QS3-gpKI

The key to do this is to make each Panel Enabled or disabled, depending on the user group that is Logged.
This is easy since we already know how to be aware of which user group belongs to the logged user from previous chapters.
First of all you have to have access to the Navigation Panel in C# like this




```
var navigationPanel =
Project.Current.Get<NavigationPanel>("UI/MainWindow/NavigationPanel1");
```

Then you can select which Panel is enabled and which is disabled, for instance like this

```
            var panelPC = navigationPanel.Panels[1];
            var panelTrends = navigationPanel.Panels[2];
            var panelDatabase = navigationPanel.Panels[3];
            if (group.BrowseName.ToString()=="Technician")
            {
                navigationPanel.Enabled = true;
                panelPC.Enabled = false;
                panelTrends.Enabled = true;
```

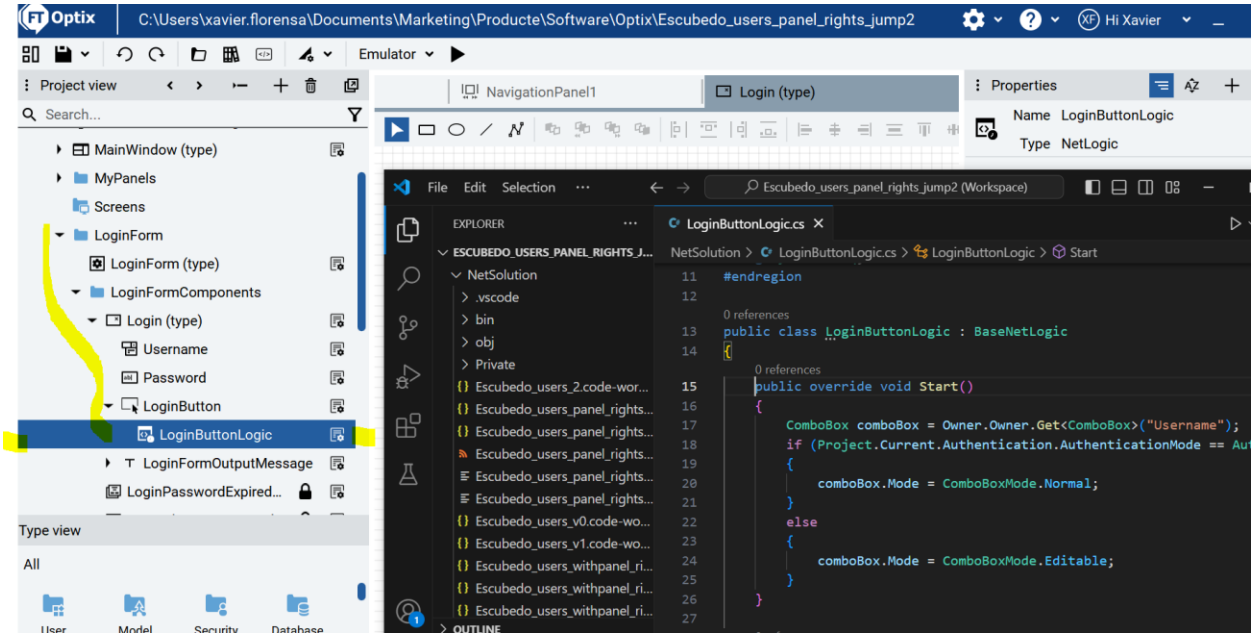
```
            panelDatabase.Enabled = true;

        }
        else
        {
            if (group.BrowseName.ToString()=="Administrator")
            {
            navigationPanel.Enabled = true;
            panelPC.Enabled = true;
            panelTrends.Enabled = false;
            panelDatabase.Enabled = true;
            }
```

Be aware that we also block the enabling of the whole NavigationPanel before any user is logged, and, once logout, when the user corresponds to Anonyous, to force a login as a previous step to see and use the panels on runtime.

You can do this this way

```
            navigationPanel.Enabled = true;
```

Another request from customers is that just after clicking the Login button, all these changes take effect. So let's introduce our code on the Login Form button script, located here:



So we can introduce this code after the logic on the click button export method

```
[ExportMethod]
0 references
public void PerformLogin(string username, string password)
{
```

Like this

```
        //Start Custom
behaviour*********************************************************************
         // Mostrar el usuari logat
       var usuario = Session.User;
       var usuari = Session.User.BrowseName;
       var mylabel = Project.Current.Get<Label>("UI/MainWindow/Label2");
       mylabel.Text = usuari.ToString();

       var navigationPanel =
Project.Current.Get<NavigationPanel>("UI/MainWindow/NavigationPanel1");
       //Mostra el grup del usuari
       var userGroups =
usuario.Refs.GetObjects(FTOptix.Core.ReferenceTypes.HasGroup, false);
       if (usuari== "Anonymous")
       {
        navigationPanel.Enabled = false;
       }
       foreach (var group in userGroups)
       {
            Log.Info("User Group: ",group.BrowseName);
            var mylabel_group =
Project.Current.Get<Label>("UI/MainWindow/Label4");
            mylabel_group.Text = group.BrowseName.ToString();
            //Make Navigation Panel visible if rigt usergroup is logged
            var  panelPC = navigationPanel.Panels[1];
            var panelTrends = navigationPanel.Panels[2];
            var panelDatabase = navigationPanel.Panels[3];
            if (group.BrowseName.ToString()=="Technician")
            {
                navigationPanel.Enabled = true;
                panelPC.Enabled = false;
                panelTrends.Enabled = true;
                panelDatabase.Enabled = true;

            }
            else
```

```
            {
                if (group.BrowseName.ToString()=="Administrator")
                {
                navigationPanel.Enabled = true;
                panelPC.Enabled = true;
                panelTrends.Enabled = false;
                panelDatabase.Enabled = true;
                }
            else
                {
                navigationPanel.Enabled = false;
                }

            }

        }

         //End custom behaviour*********************************************
```

You can find the code here

https://github.com/xavierflorensa/FTOptix_user_management_panels

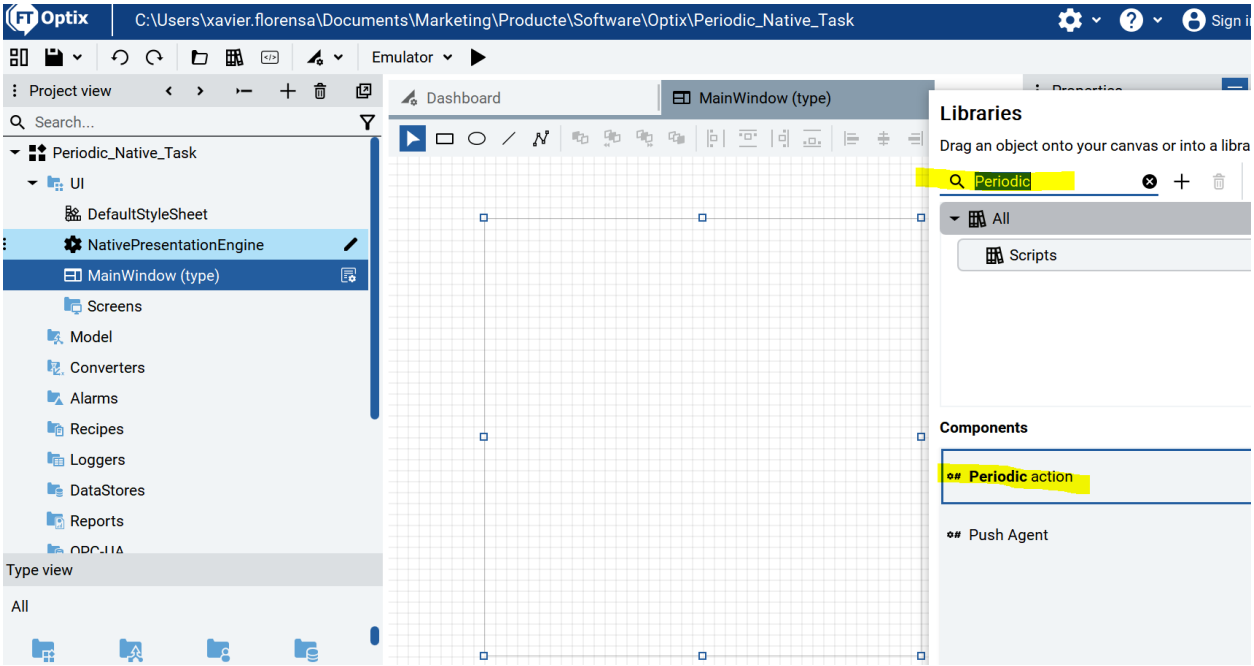
And you can see the results on this video
https://youtu.be/bp9QS3-gpKI

14.12. Registering logged user on a database

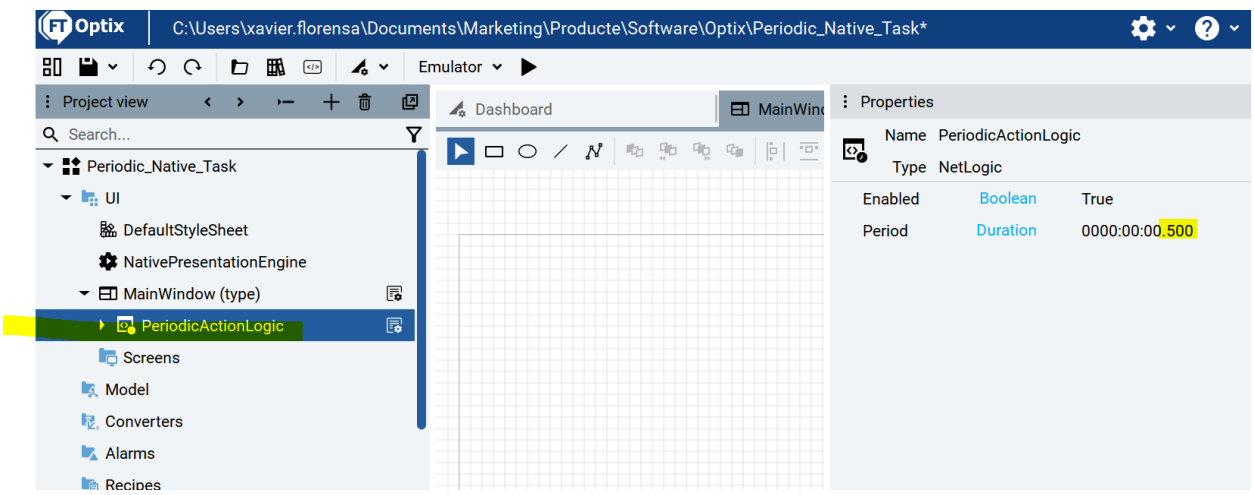
First you have to store the logged user on a Variable and then use this variable for the datalogger, the drag and drop the Datalogger to a datagrid

14.12.1. Adding the Logged user to a variable

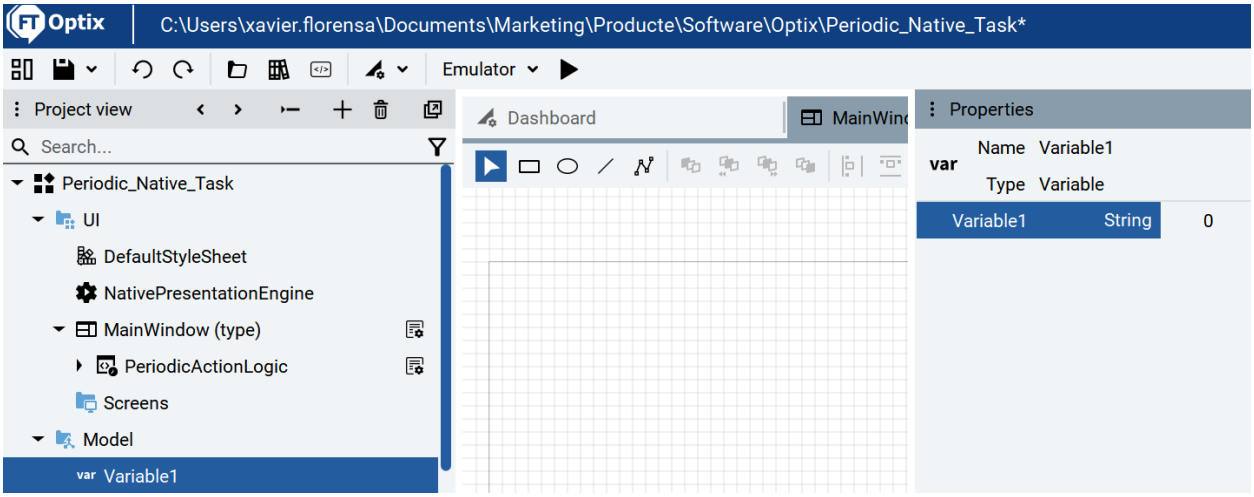
as explained in the Periodic tasks section:
You have a library to do so
Let's create a blank project
Open library tool and search for Periodic



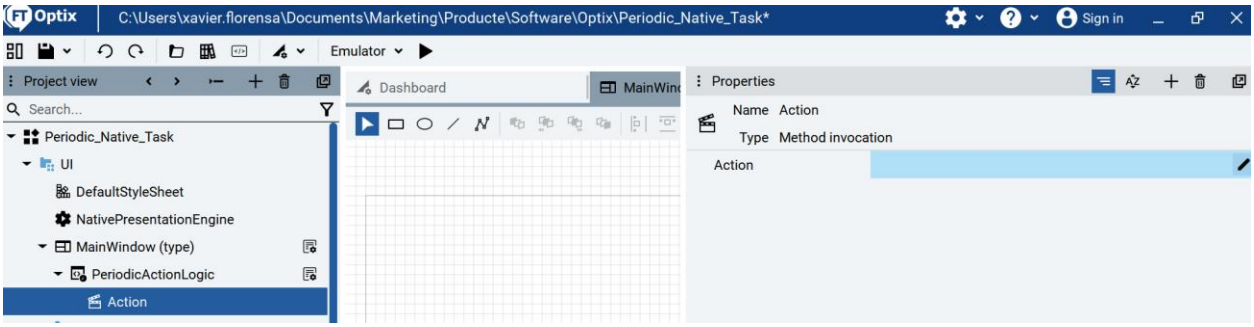
Drag and drop the "Periodic action" component to the desired scope, for instance, Main Window
A new PeriodicActionLogic script will appear under Mainwindow (but you do not need to go to C#)
You have the period, for instance each 500 mseconds



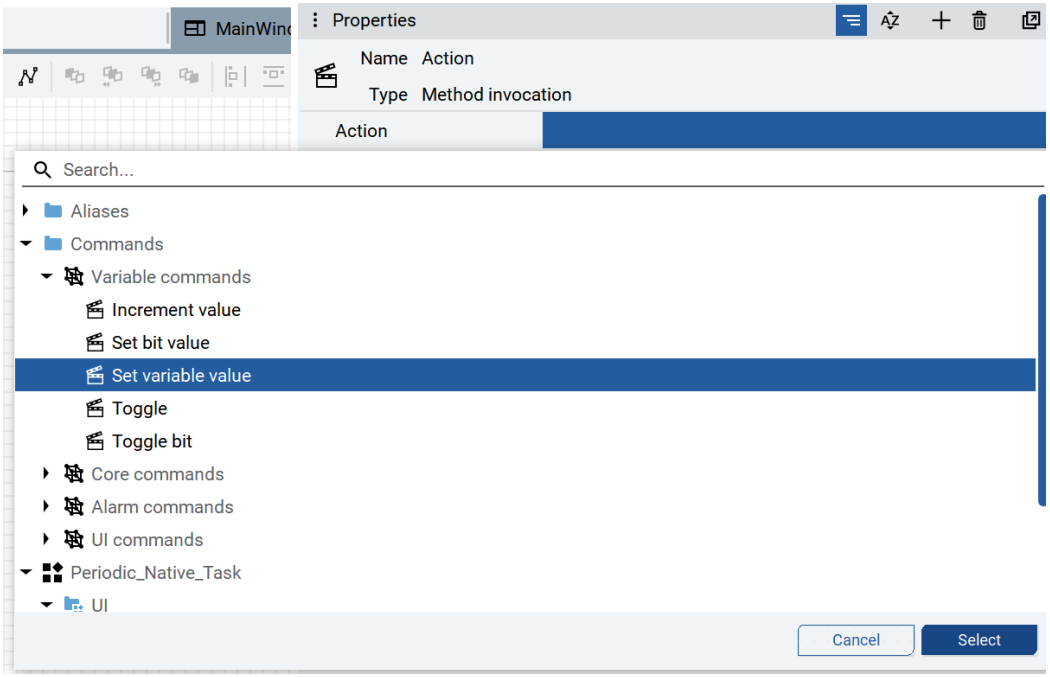
Then you can add an action, for instance modify a variable value
Add a variable to Model

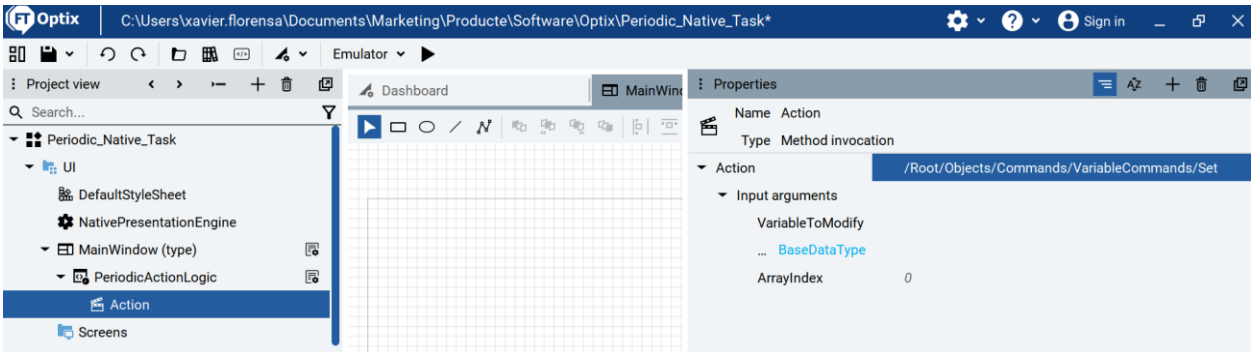


Then add an action to the Periodic
Click on ACtion
Click on the pen

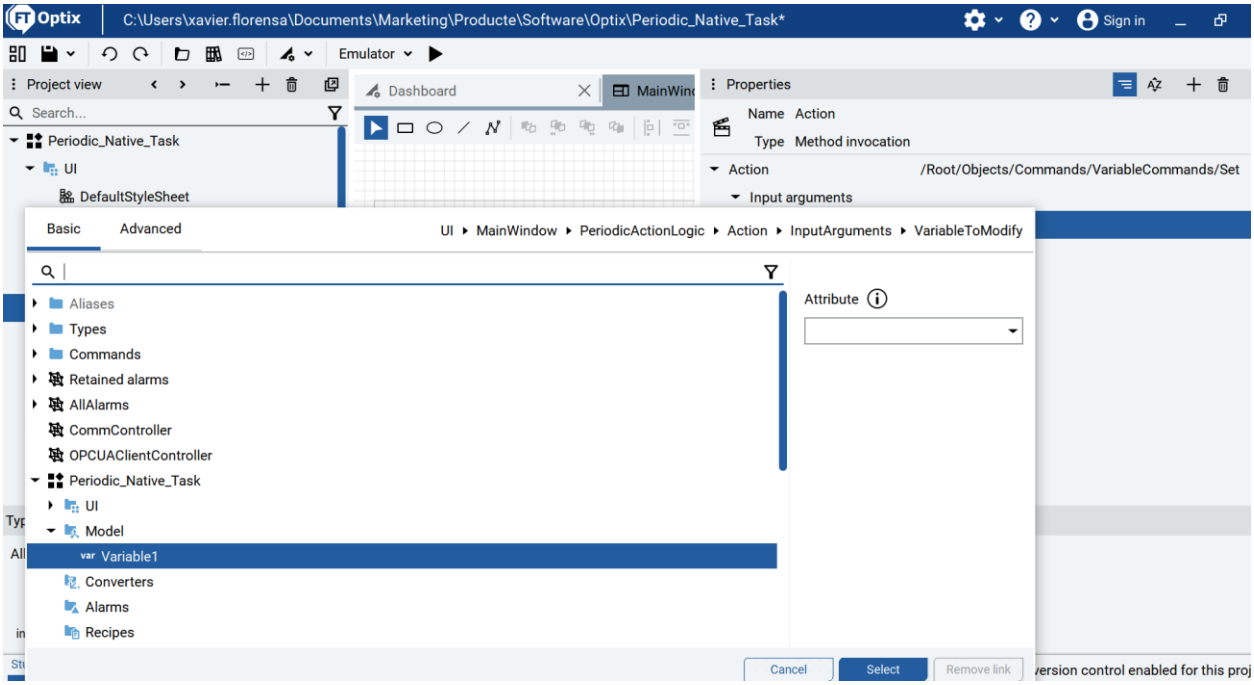


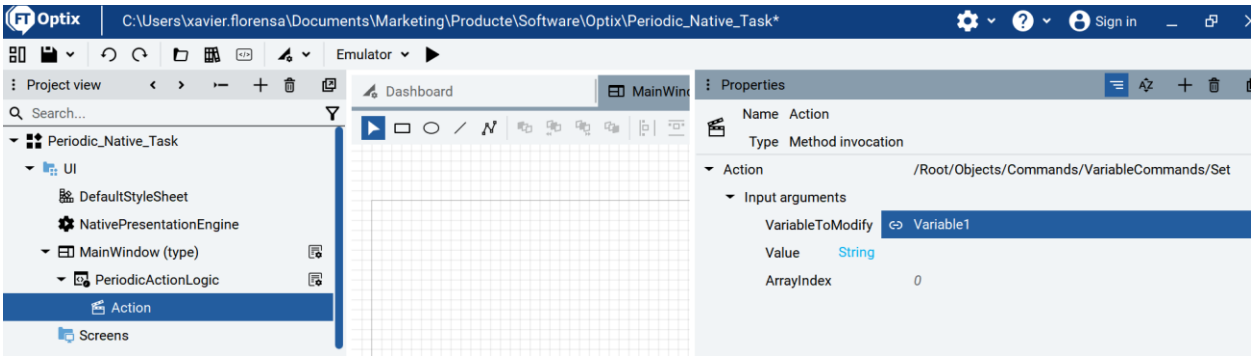
Add a commnad



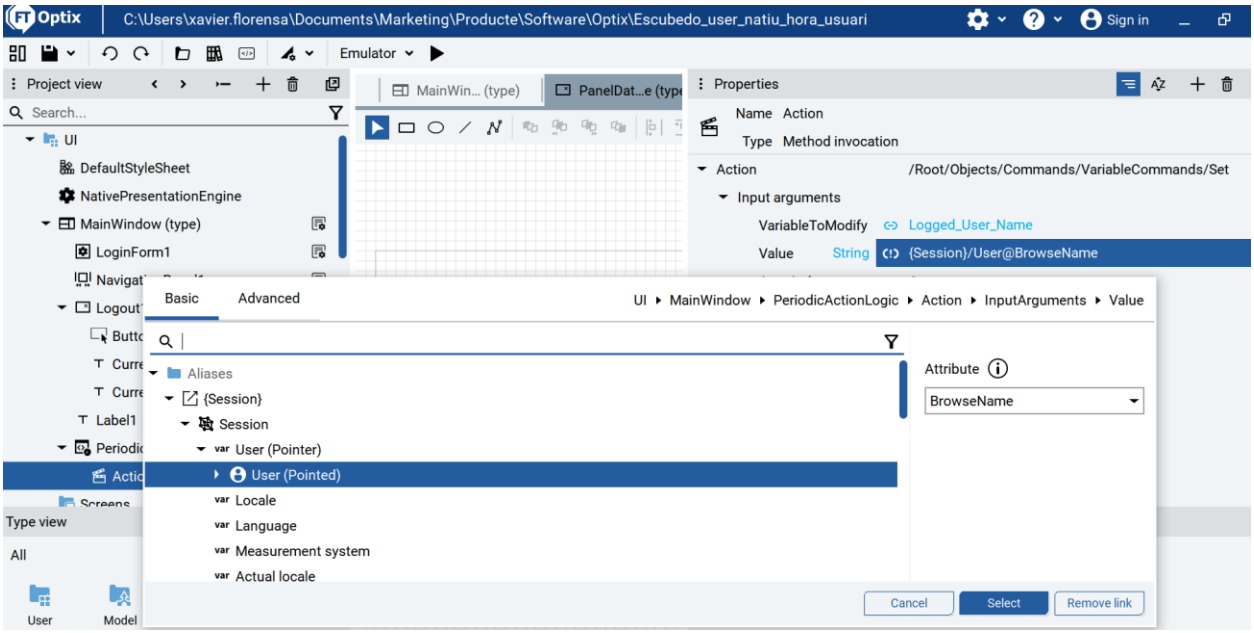


Set VariableToModify

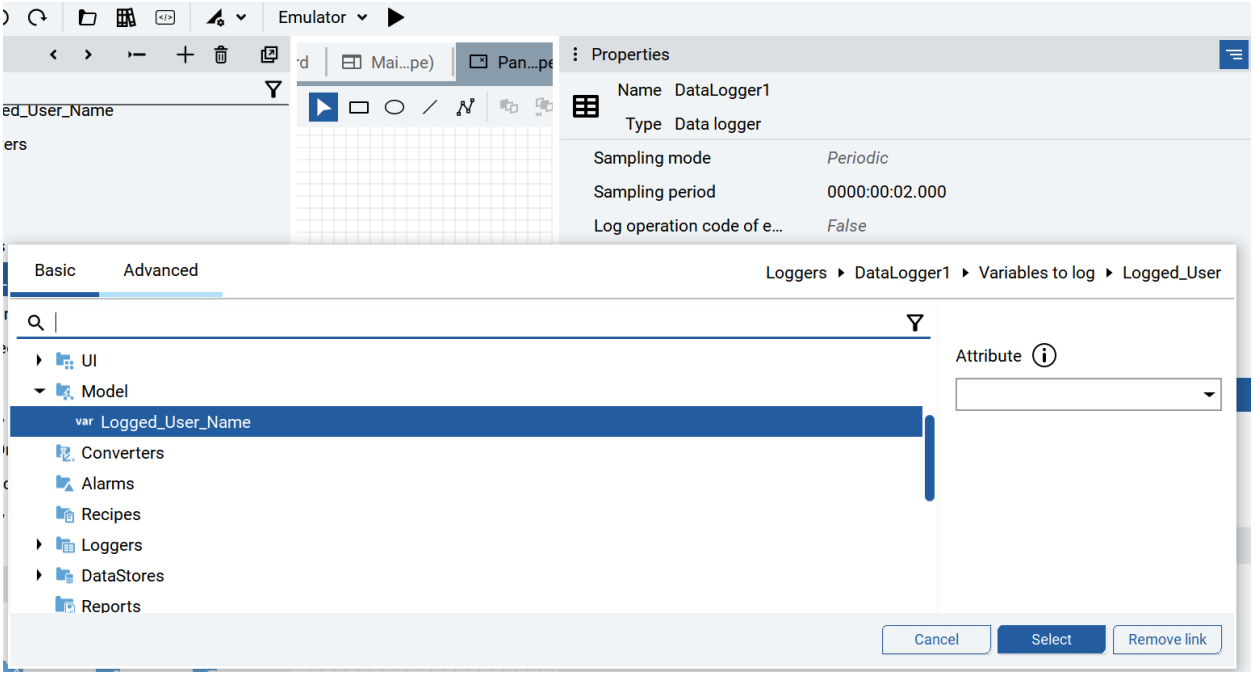


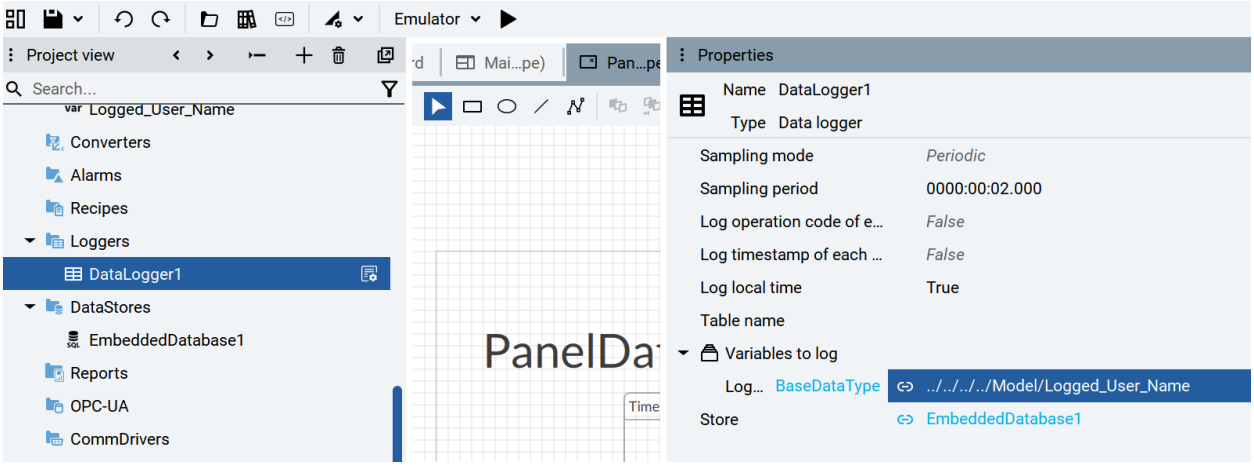


Select the value, for instance, logged user
Click on the dynamic link on Value



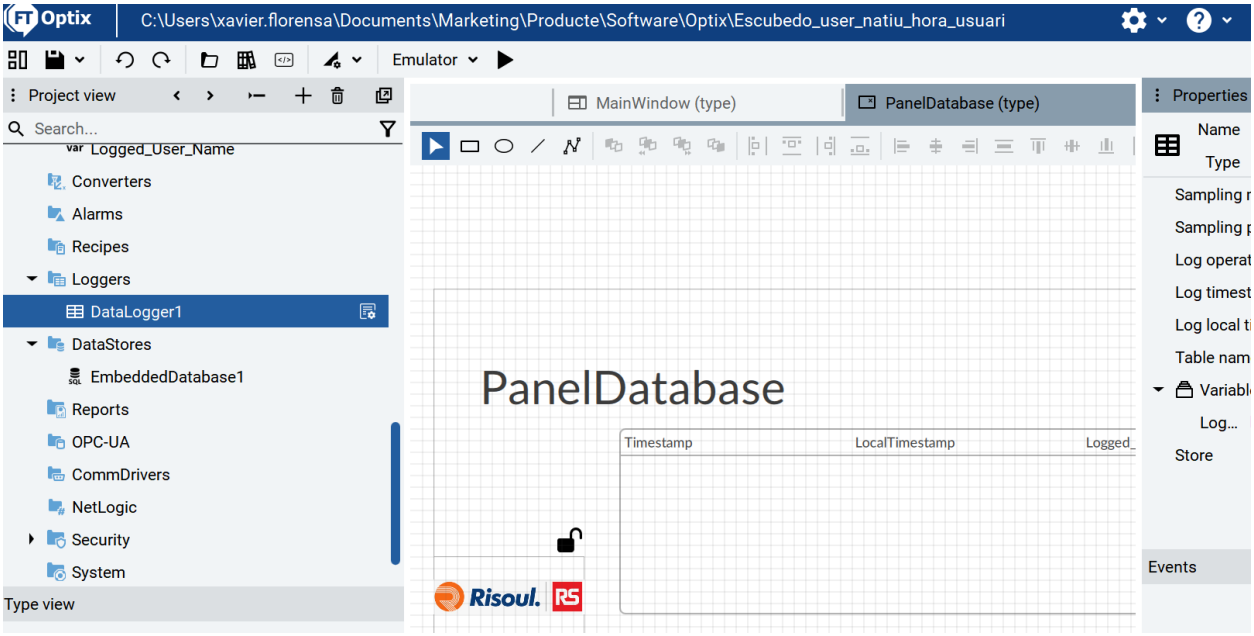
Then adjust the Variable to log on the Datalogger



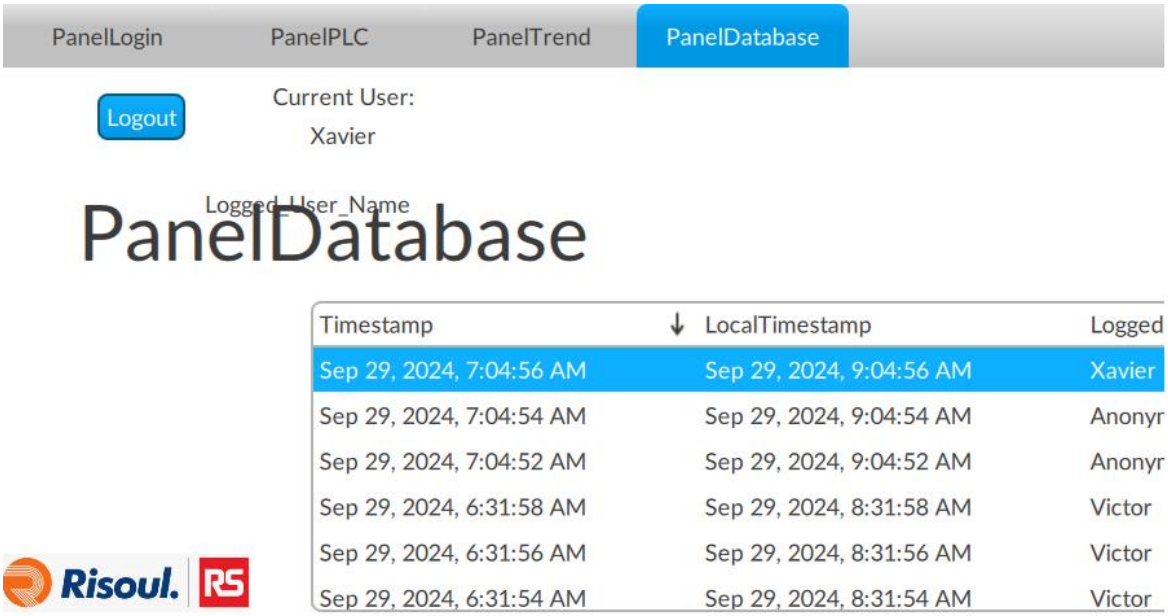


This will record the user each 2 seconds but you can do on each variable change, on Sampling mode.

Finally, drag and drop the Dataloger to the new datagrid on a desired panel.



That's all



You can find the code here

https://github.com/xavierflorensa/FTOptix_logged_user_record.git

This is also explained in this video from RA Engage forum

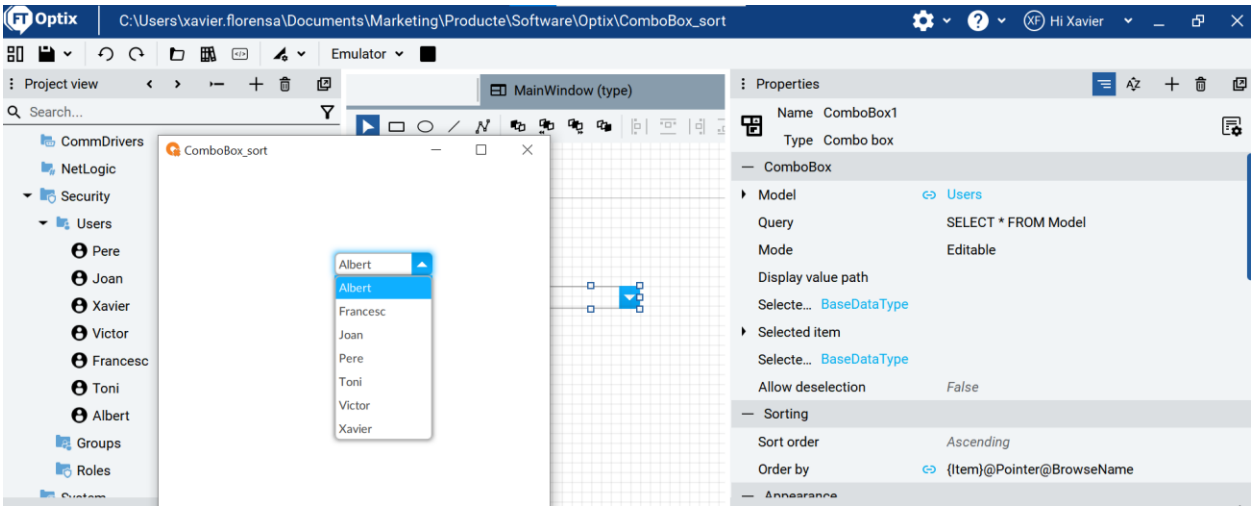
https://engage.rockwellautomation.com/HigherLogic/System/DownloadDocumentFile.ashx?DocumentFileKey=2d24f65e-70e0-437b-8e77-0190c6e4880c

14.13. Alphabetical order on the Users ComboBox

Normally the users will appear on the User folder and on the ComboBox in the order they were created.
If you want to have them in Alphabetical order, you can use the Alias called {item} and point to {item}@BrowseName

https://engage.rockwellautomation.com/discussion/username-in-alphabetical-order-on-combobox-when-login?ReturnUrl=%2fcommunities%2fcommunity-home%2fdigestviewer%3fListKey%3d13f227ed-c9e5-44bf-8301-493f3e399158#bm28484fce-d8c9-4e18-8b19-3a1529568deb

This way



You can find the code here

https://github.com/xavierflorensa/FactoryTalk_Optix_User_ComboBox_Alphabetical_Sort.git

15. Datalogger and Writing to an SQL database

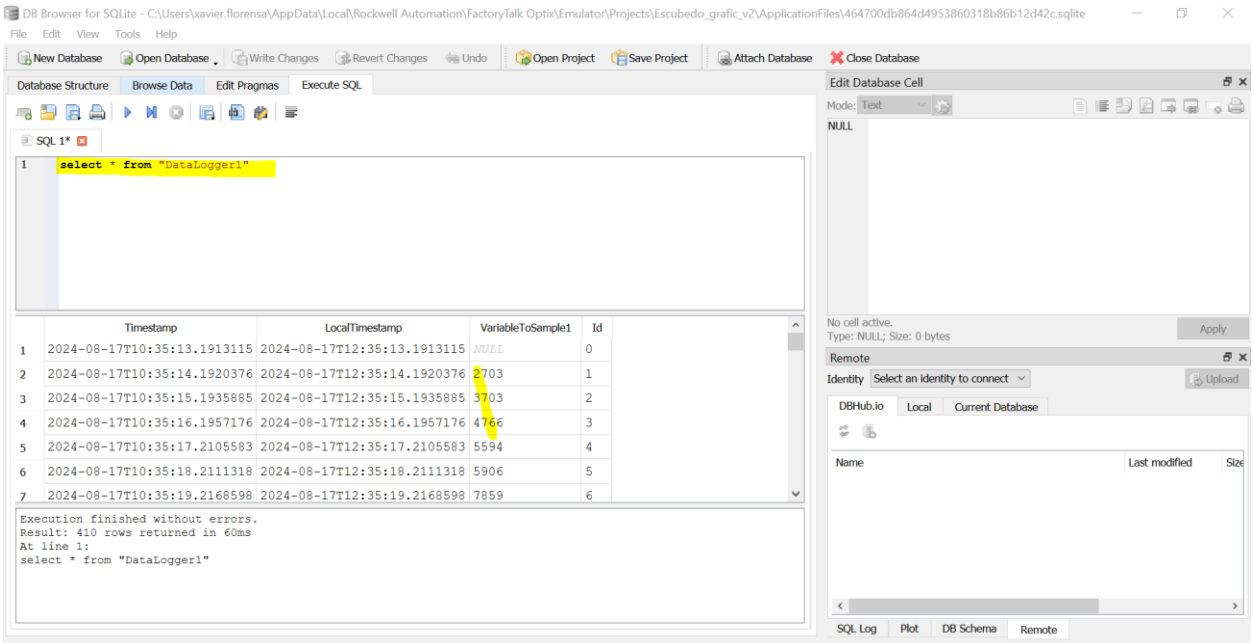
15.1. Datalogger. Working with an embedded database

You have the step by step process on this video
https://www.youtube.com/watch?v=Kiu5fdDzC7k&list=PL3K_BigUXJ1M1-JpRiwIIhzJUbhwtK3yy&index=7

What if you want to access the embedded database from other applications.
Each project stores the database with a number (or name) and extension *.sqlite on this folder

C:\Users\<yourName>\AppData\Local\Rockwell Automation\FactoryTalk Optix\Emulator\Projects\<ProjectName>\ApplicationFiles

You can open the database for instance with SQLite Browser
sqlitebrowser.org



15.2. Datalogger Trigger on Tag Transition

Imagine you want to store data on each machine cycle, or on each good part is produced.
You can do it this way

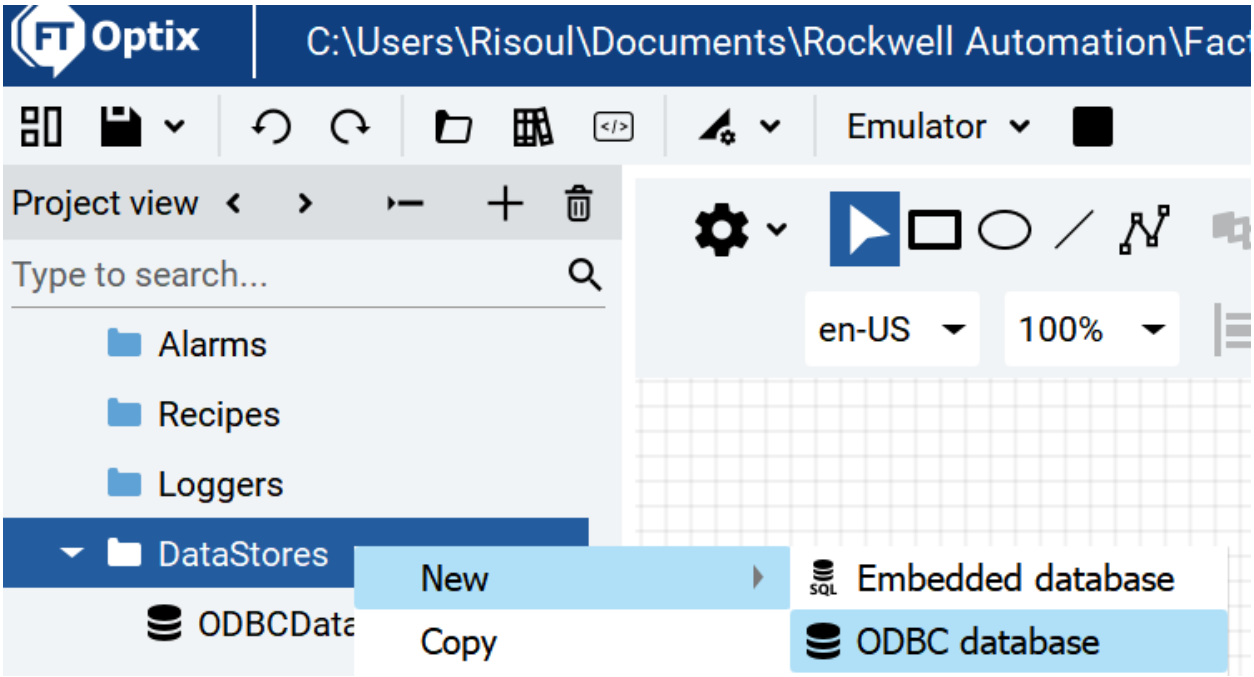
https://www.youtube.com/watch?v=IhsMH06kN0w

https://github.com/xavierflorensa/FactoryTalk_Optix_Datalog_Trigger_PLC

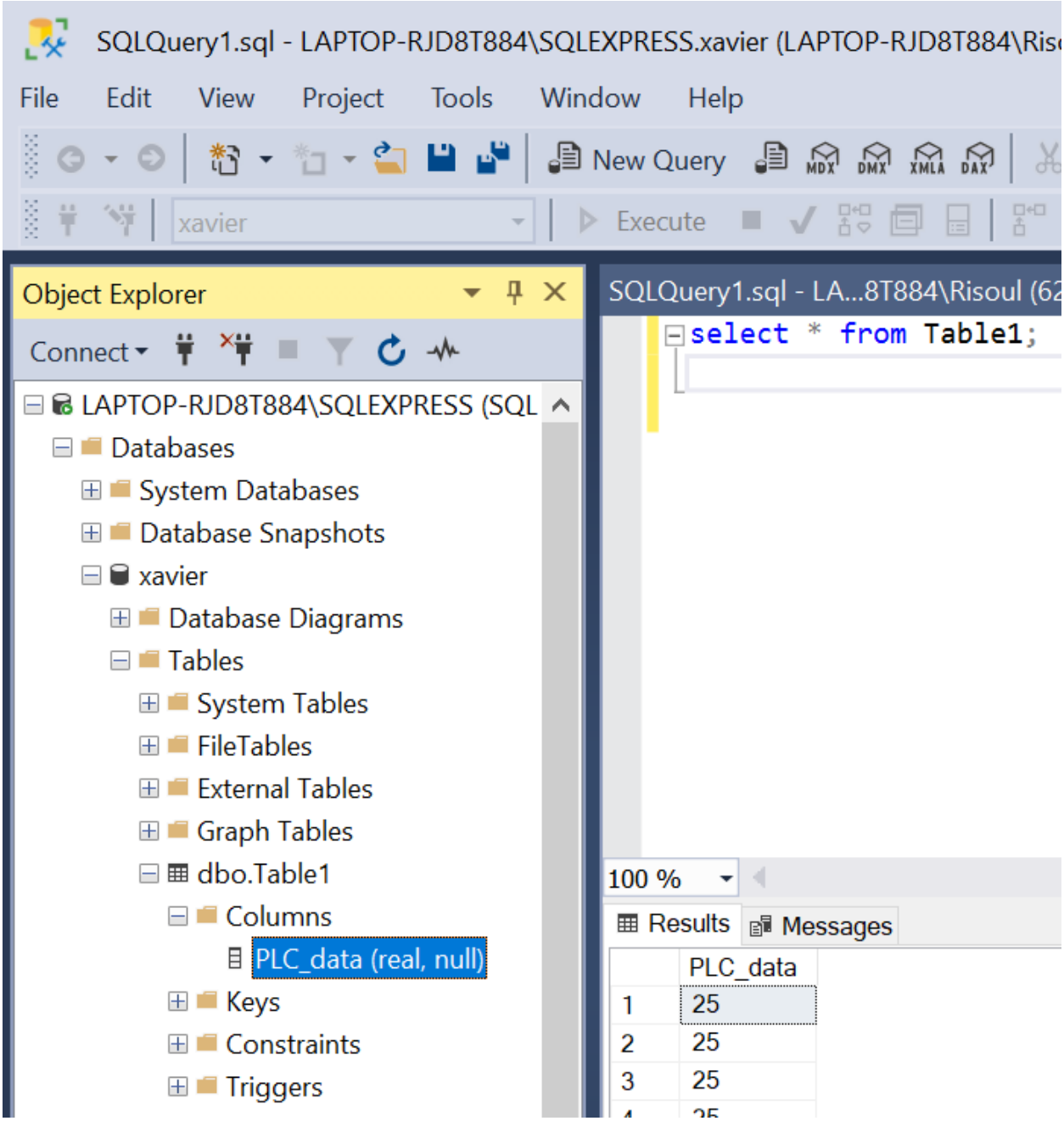
15.3. Let's write to an external SQL database from Optix

First just read from a Database
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/using-the-software/datastore-database/create-a-database.html

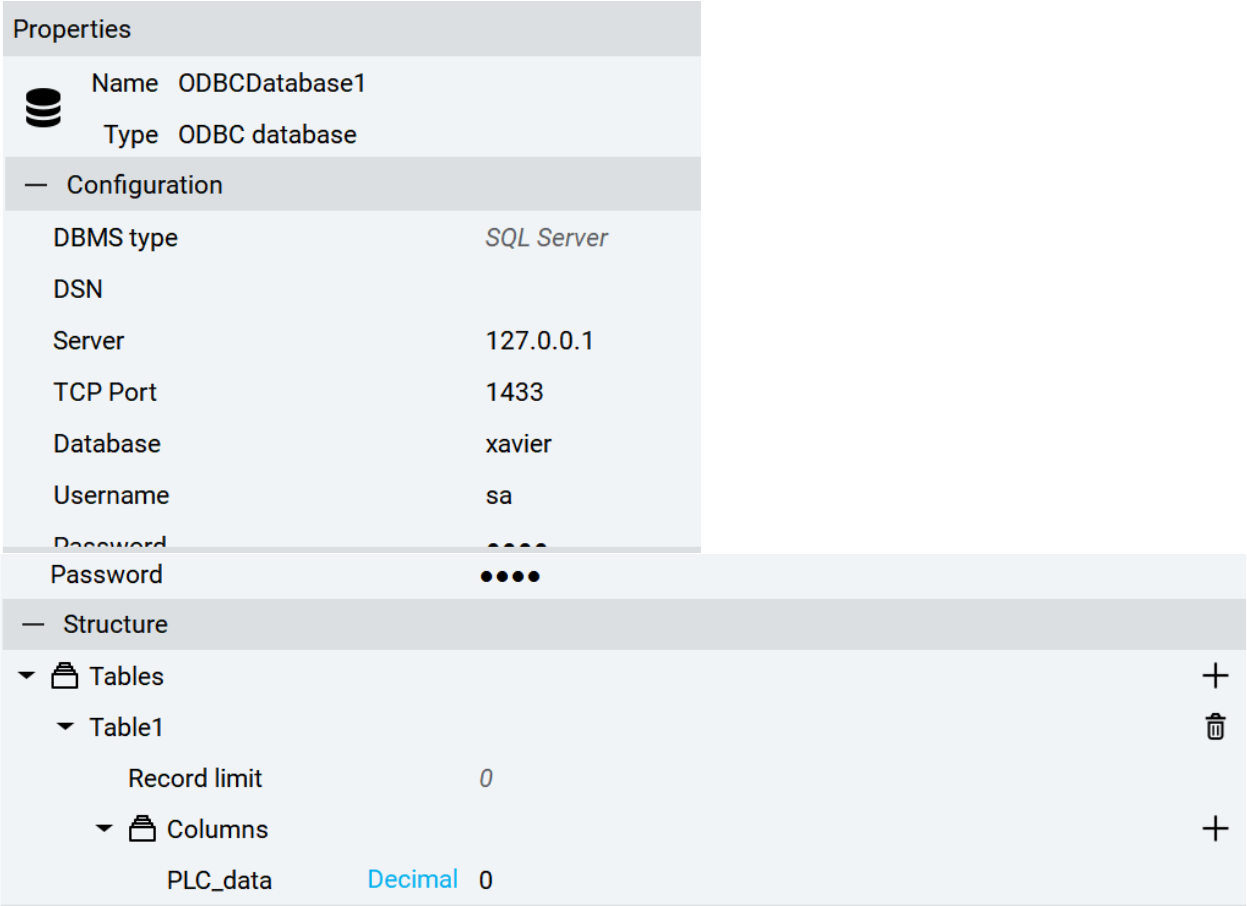
Add a Database object



Be sure to have an SQL working database, with user, password, access thru TCP/IP, etc.



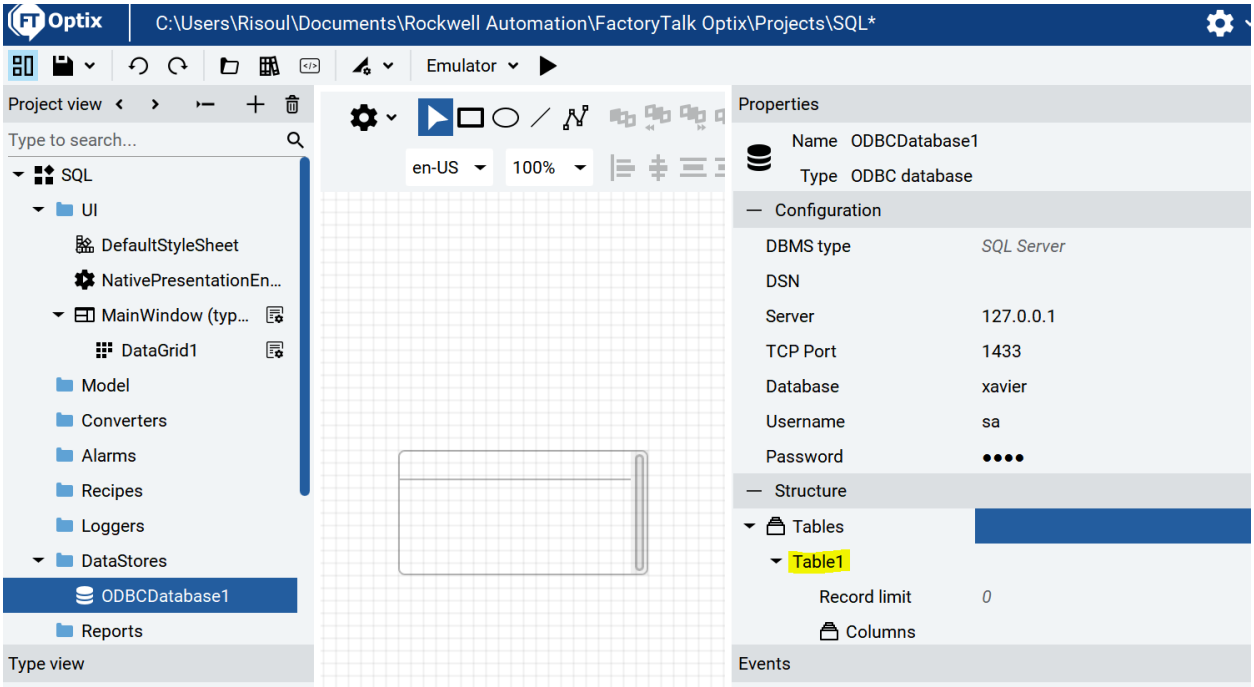
Complete database properties click on Table + and Column + to match our Database



If you do not add a column here you will have errors when drag and drop to the Table1 to the data grid (Unable to insert a empty table to the datagrid)

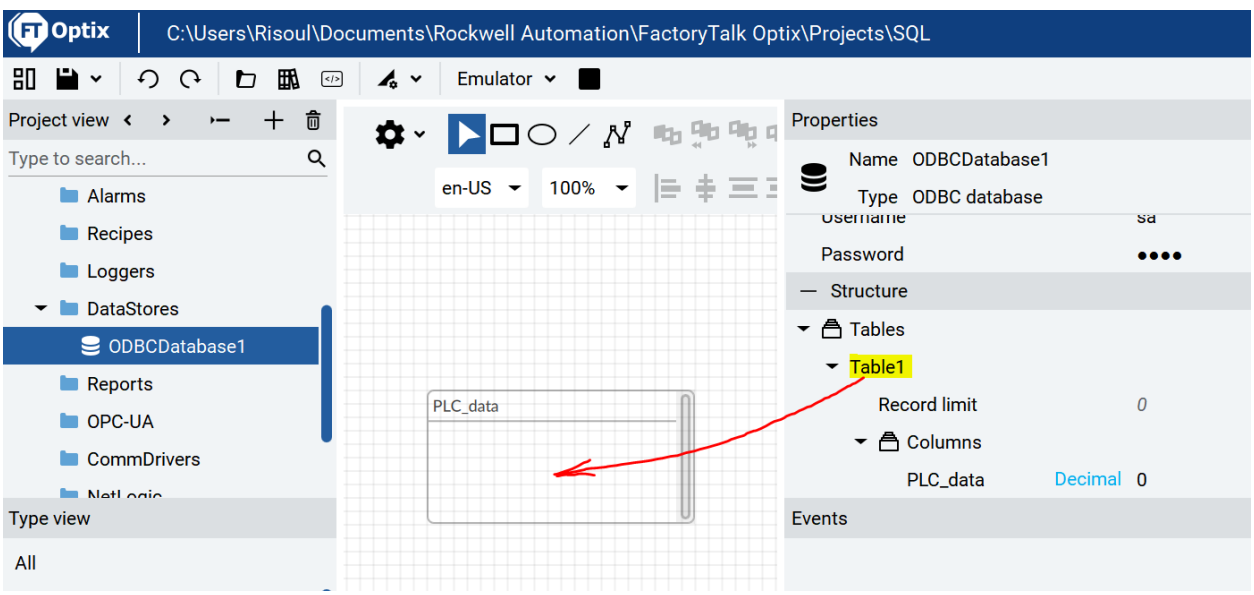
Display Database data
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/using-the-software/datastore-database/display-database-table-data.html

Just click on the + on Tables to create a new table (even thought it already exists on the MS database)

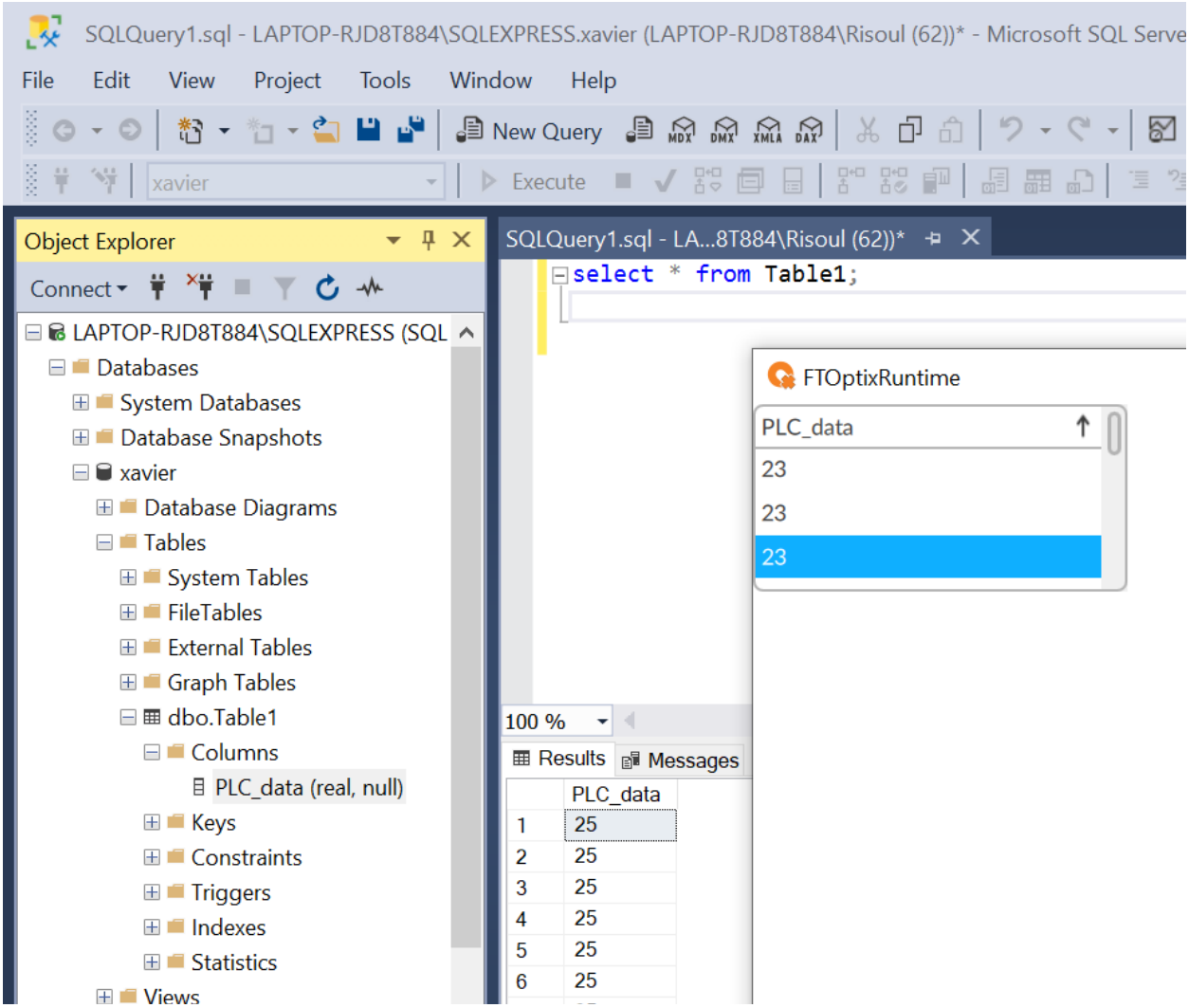


Do not forget to add a column minimum

Drag and Drop



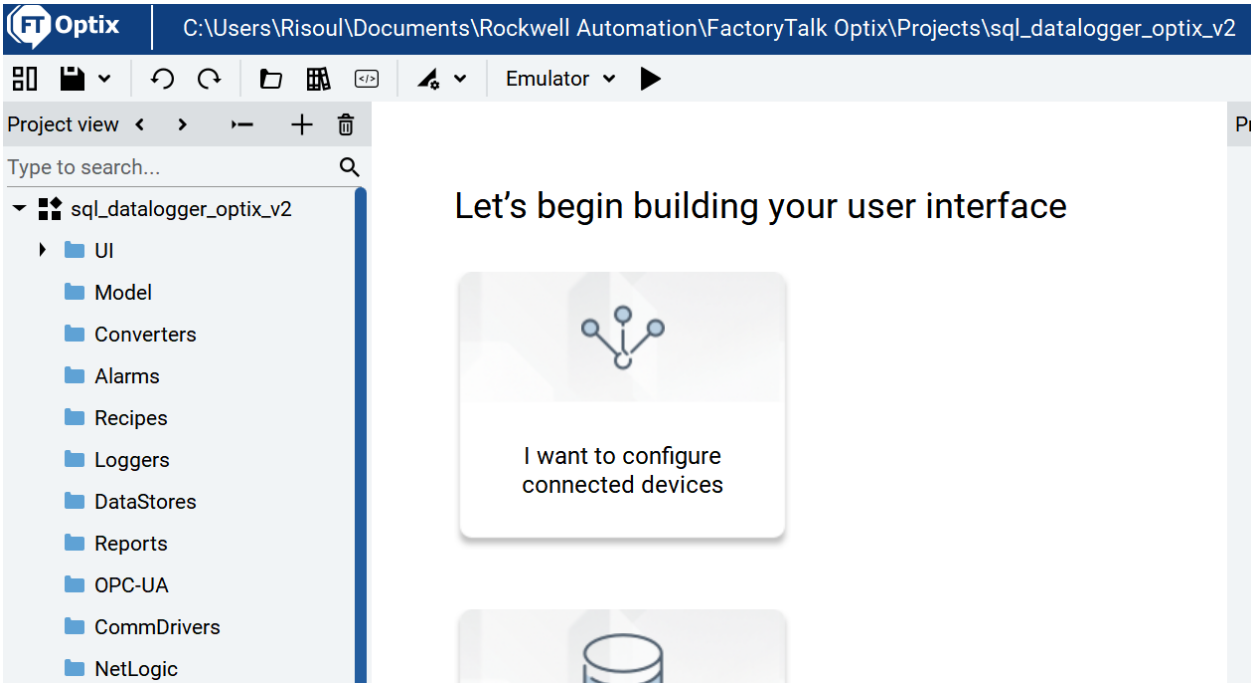
It works!!



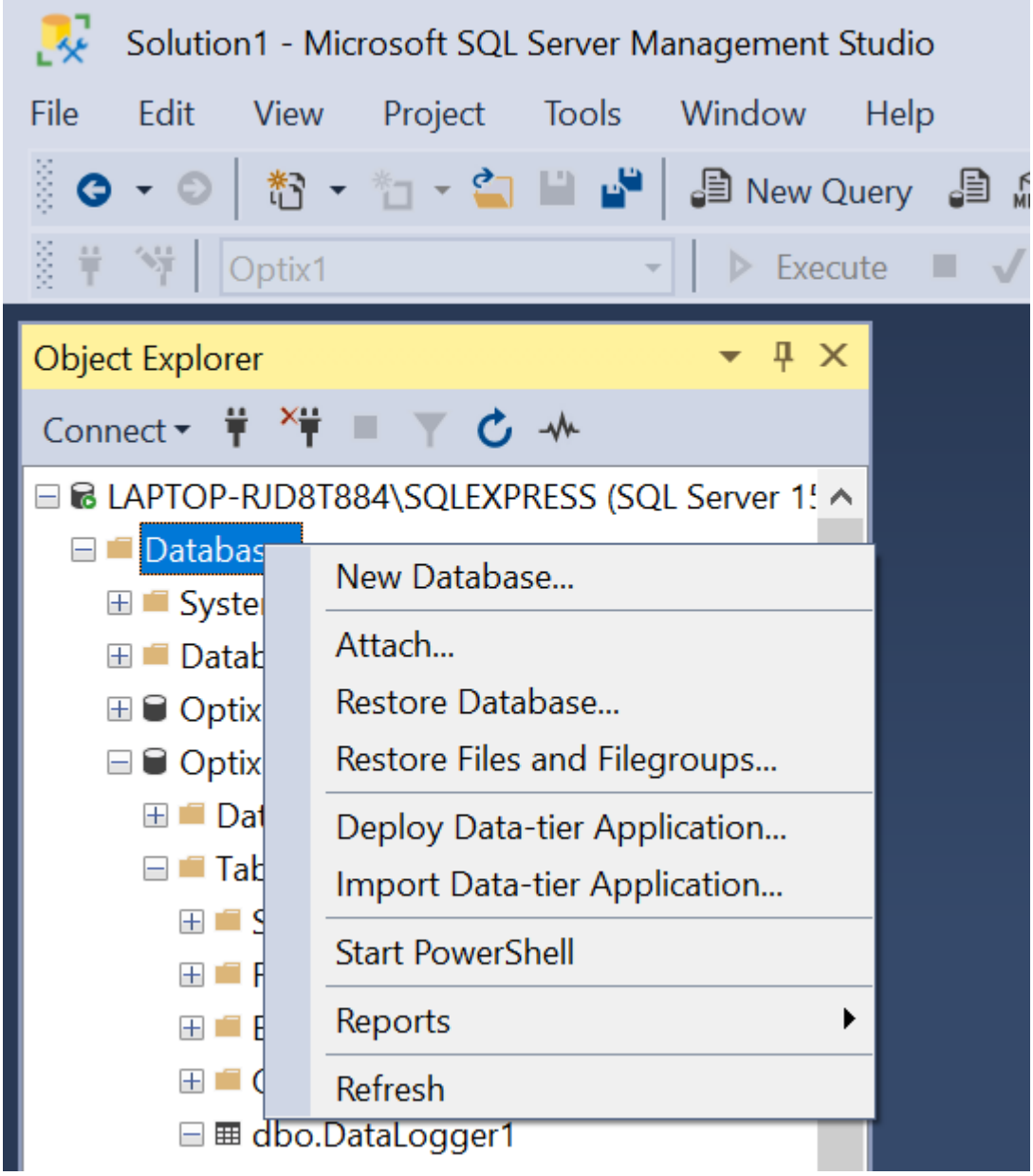
Now let's try to write a variable on the database
Using this example
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/application-examples/logger-tutorial/develop-a-data-logger-with-odbc-data-store.html

Download a sample project:
https://www.rockwellautomation.com/content/dam/dita/en/factorytalk-optix/1-00/downloads/DataLoggerODBC.zip

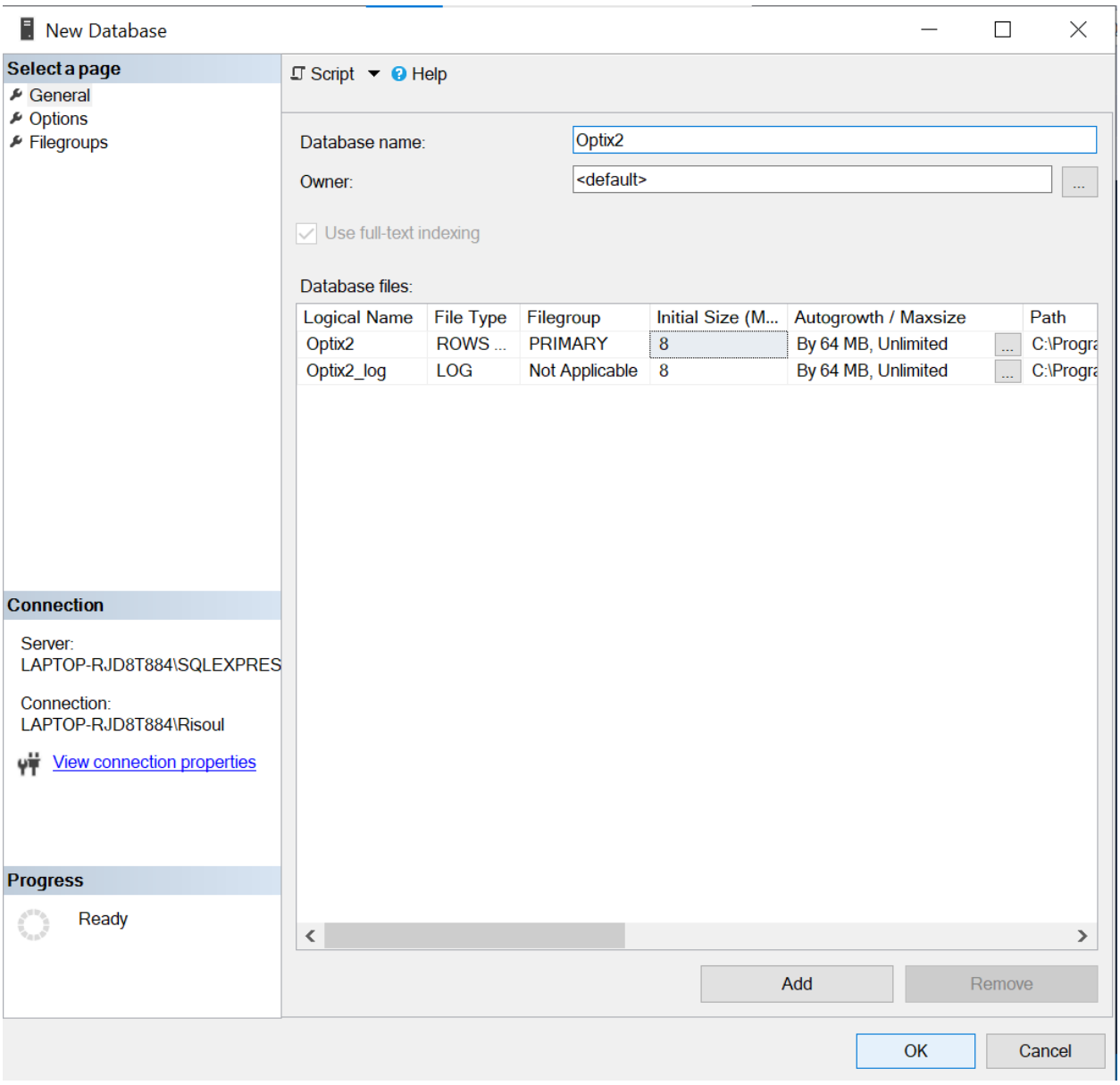
To see how it is made.
But let's start creating a complete new project
For instance sql_datalogger_optix_v2



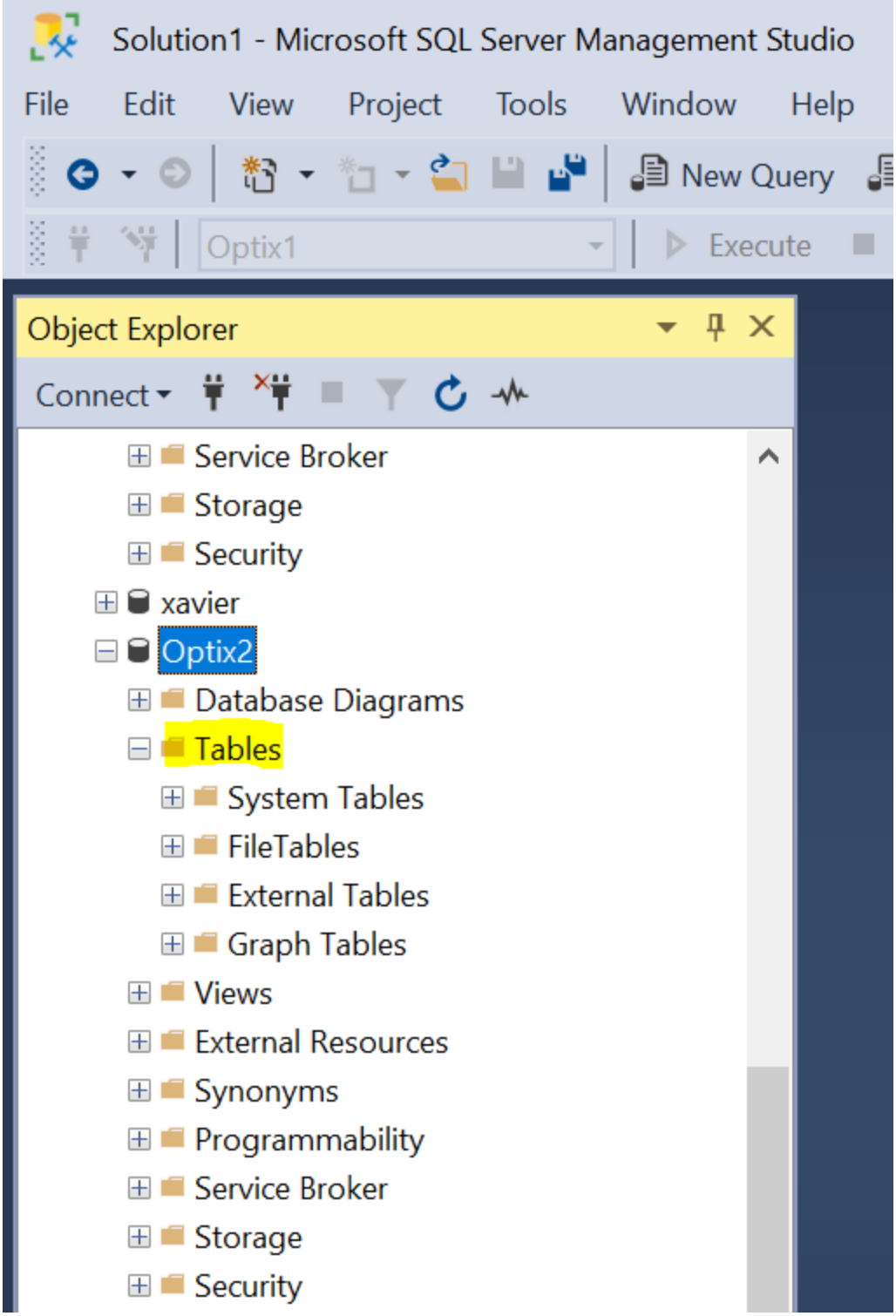
But first let's create a database without a Table (Optix and The Datalogger function will create the Table)
For instance let's create a new database Optix2



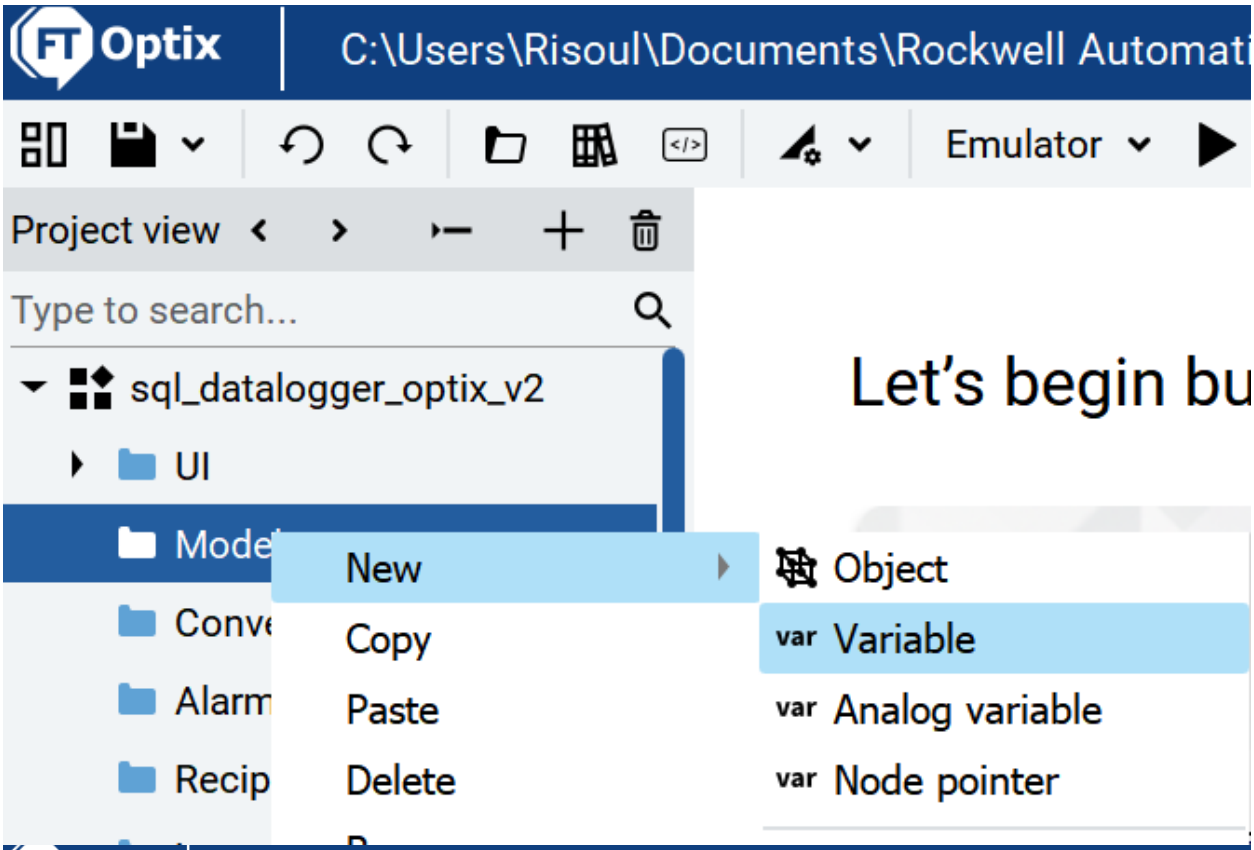
Just give a name, that's all

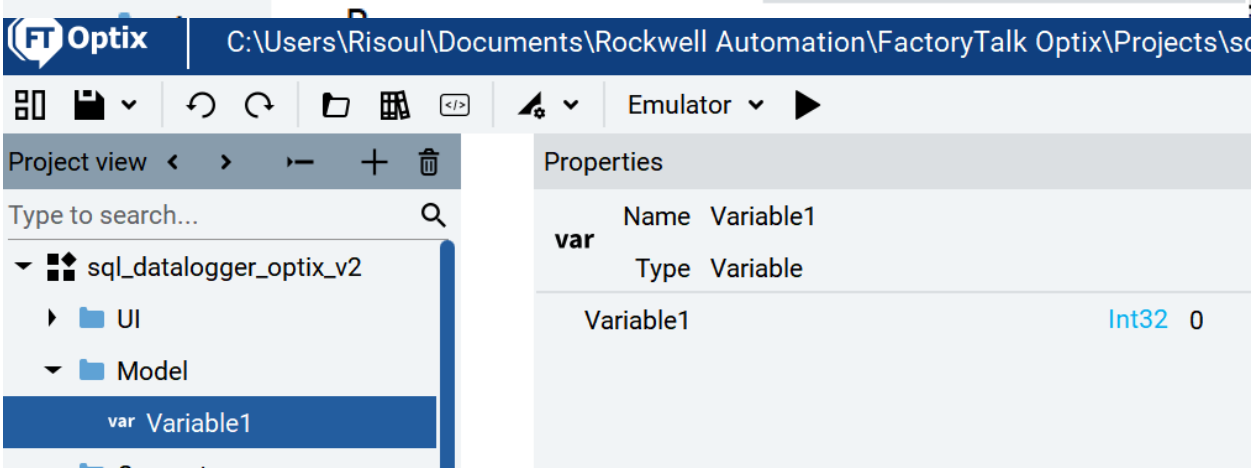


There are no user tables

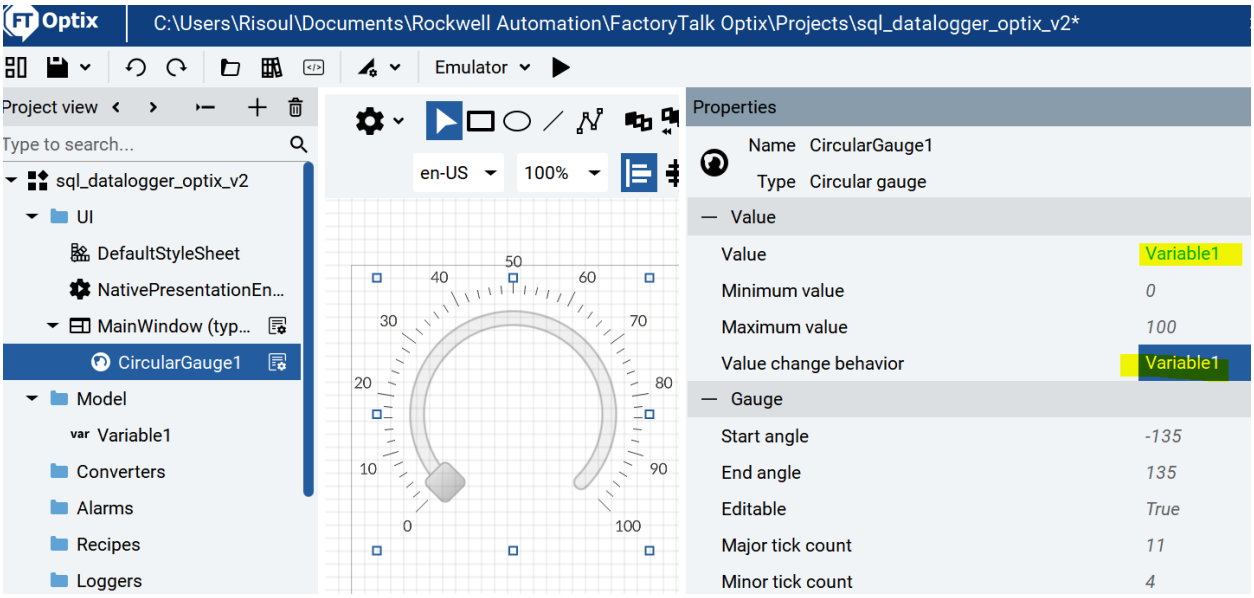


Now Let's go to Optix
Add a variable to work with

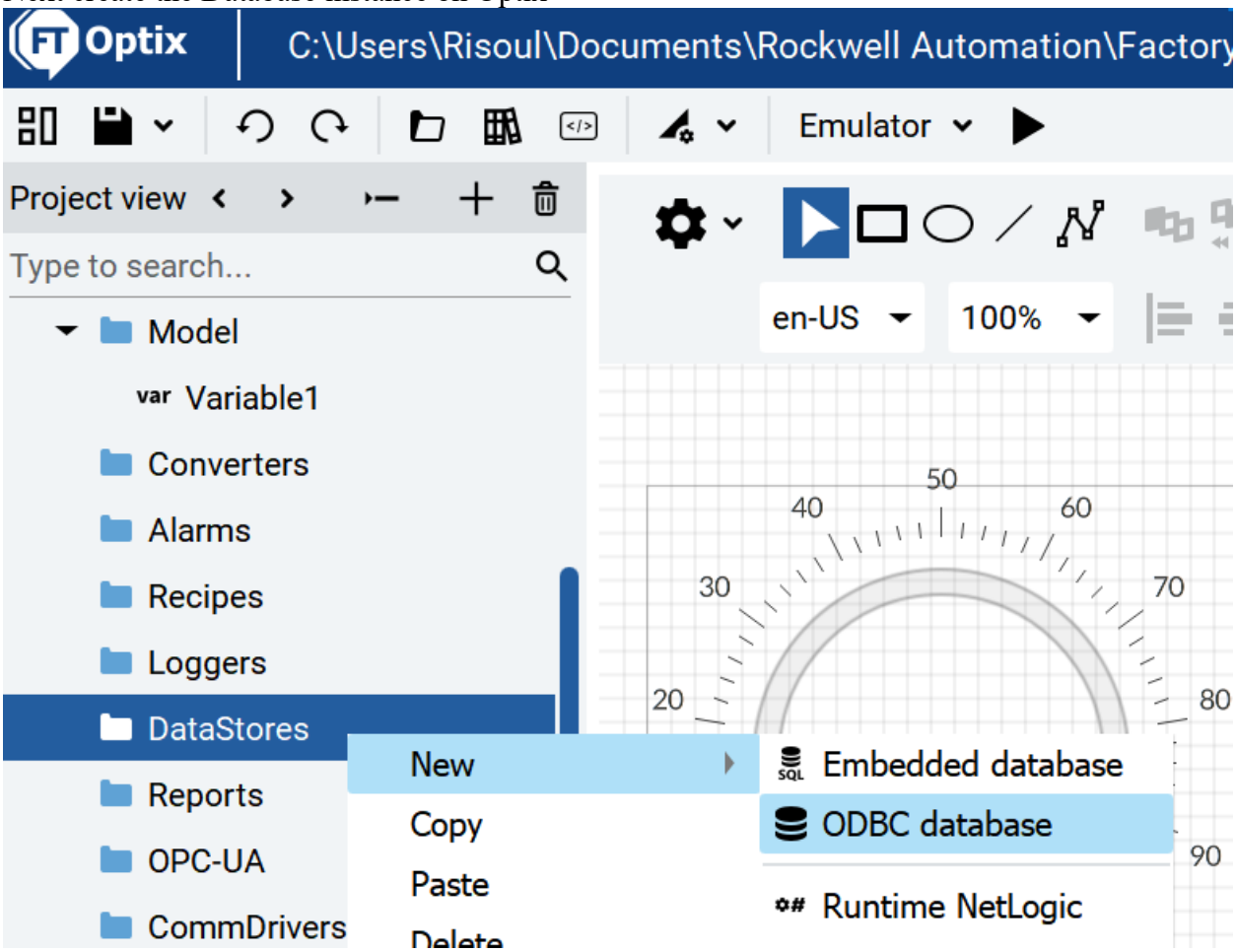




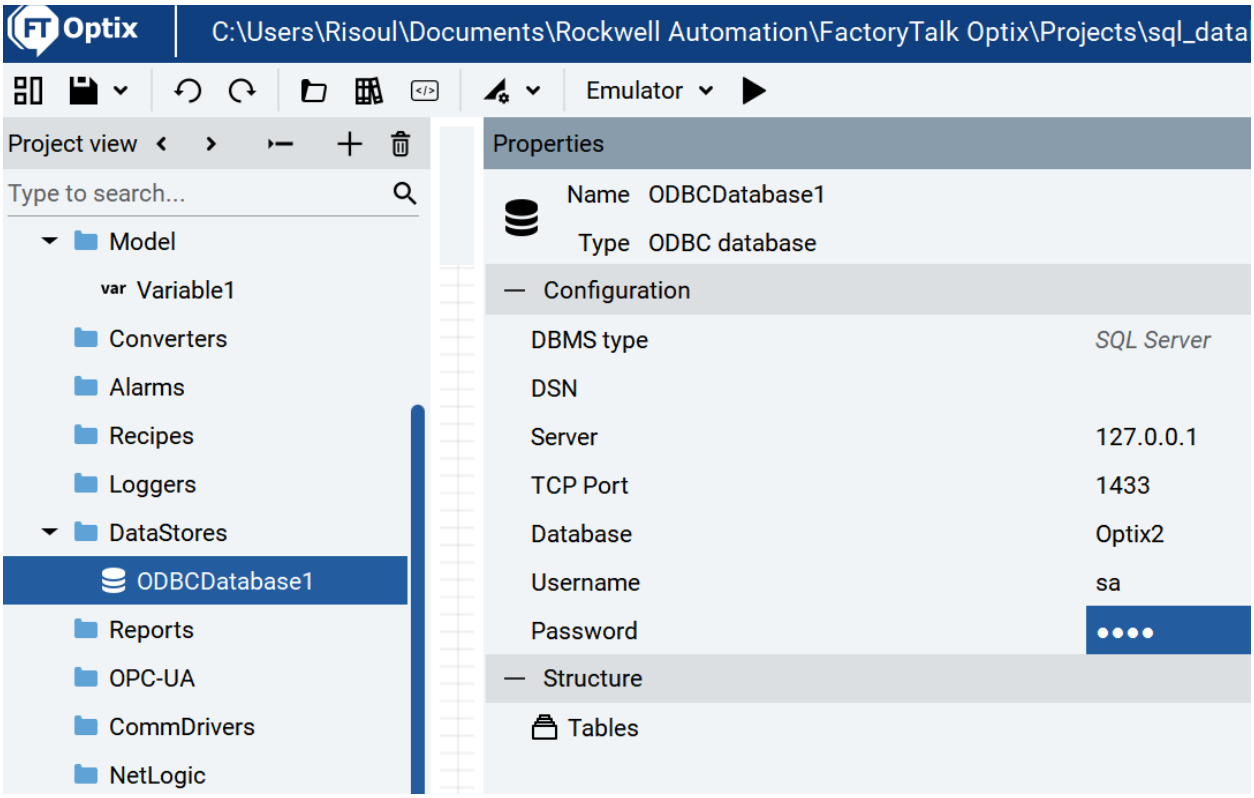
Add a Slider to enter values to such variable



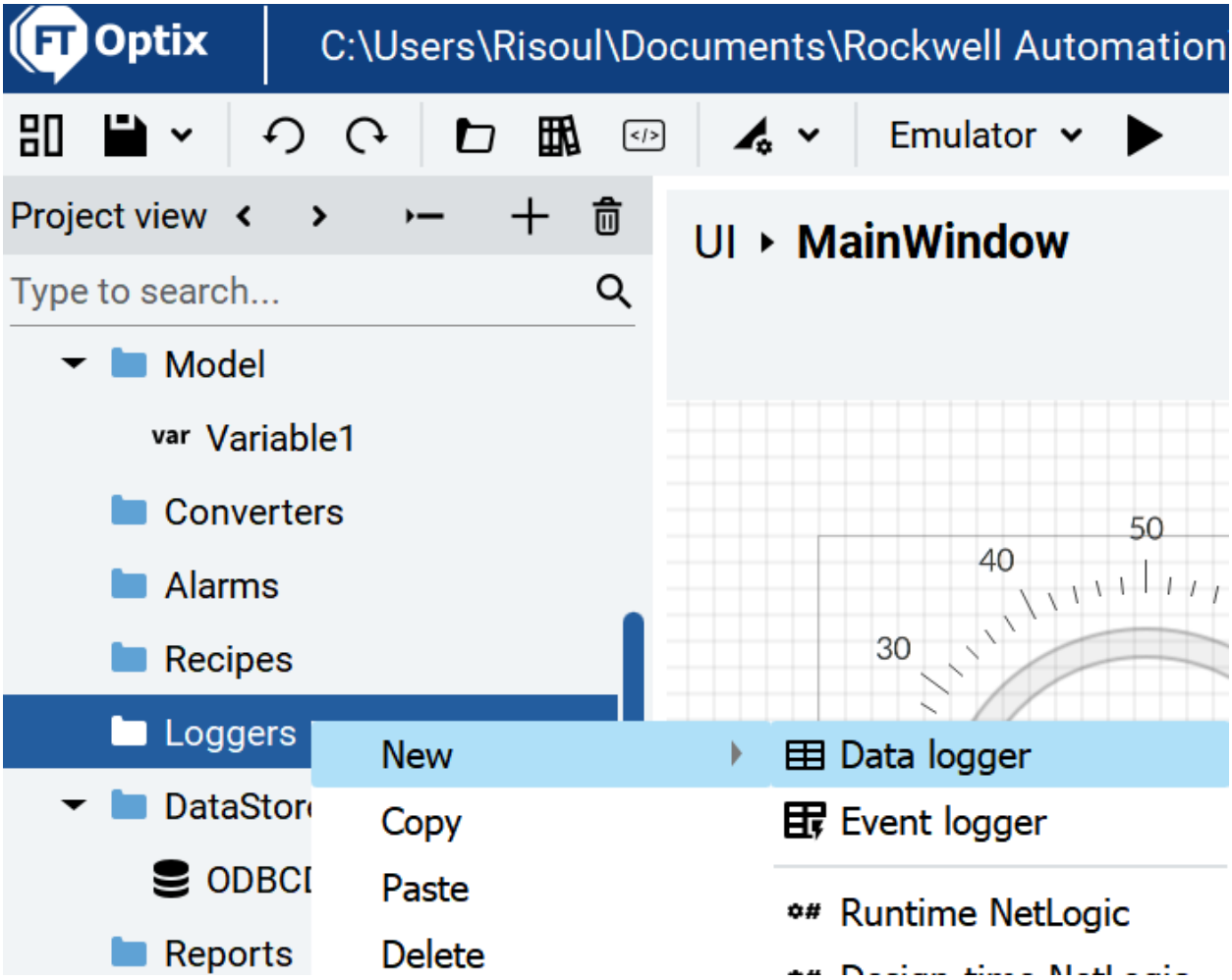
Next create the Database instance on Optix



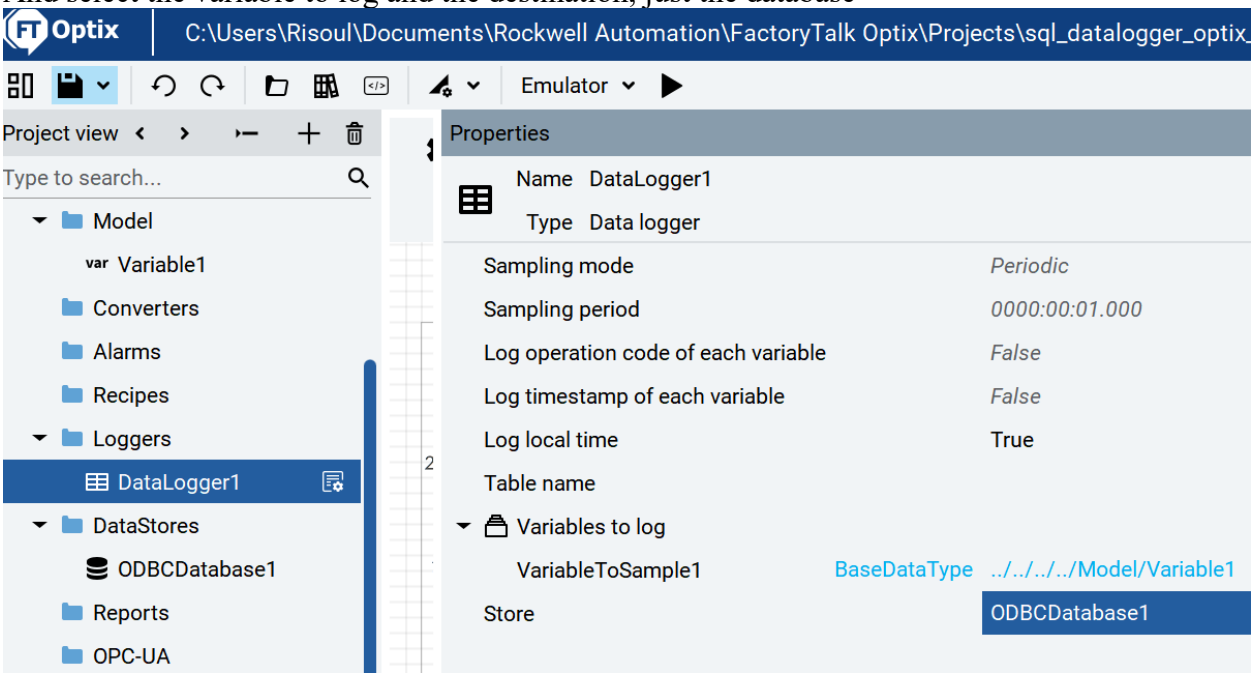
Point to the existing database, but do not create any Table on Optix



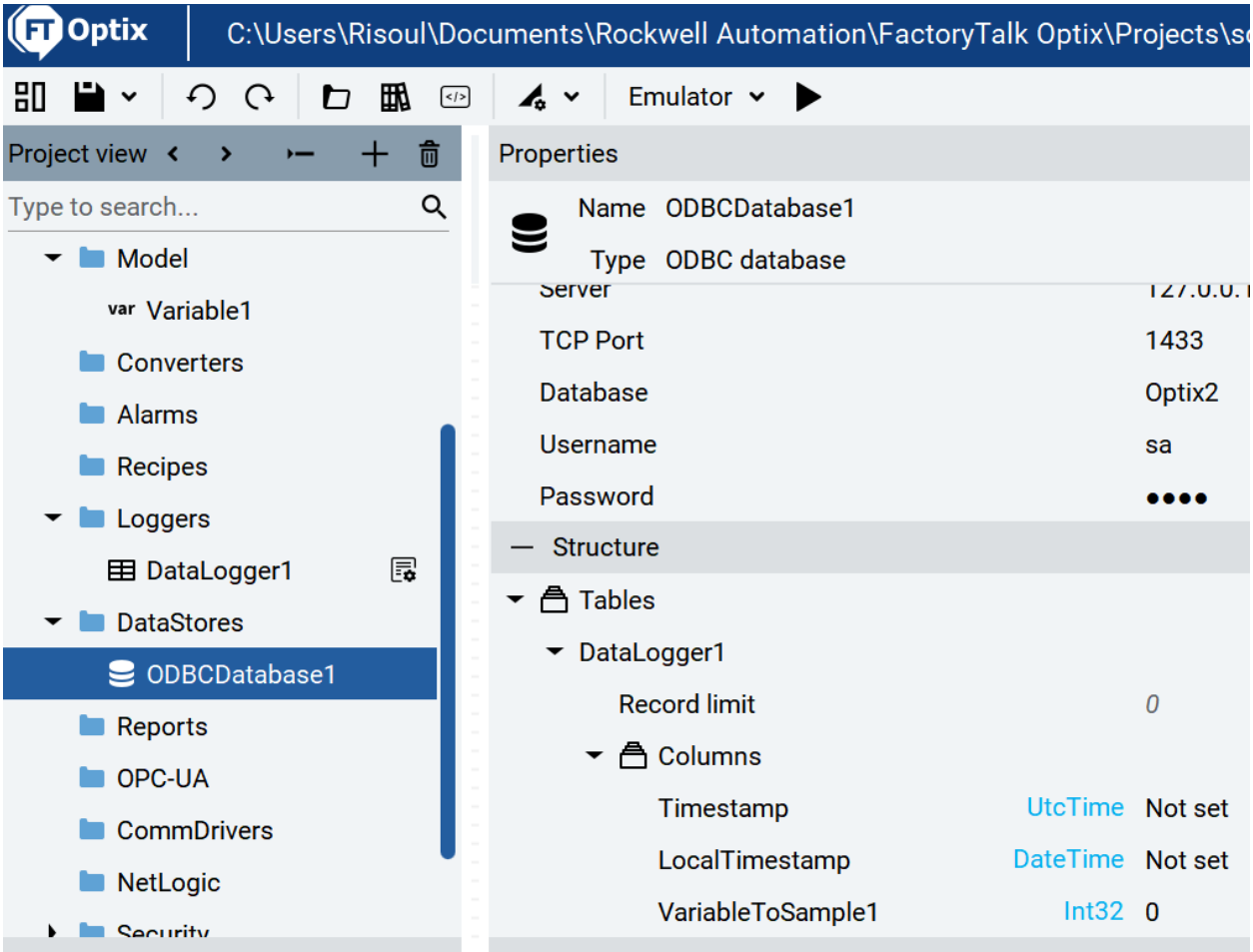
Then create a new DataLogger



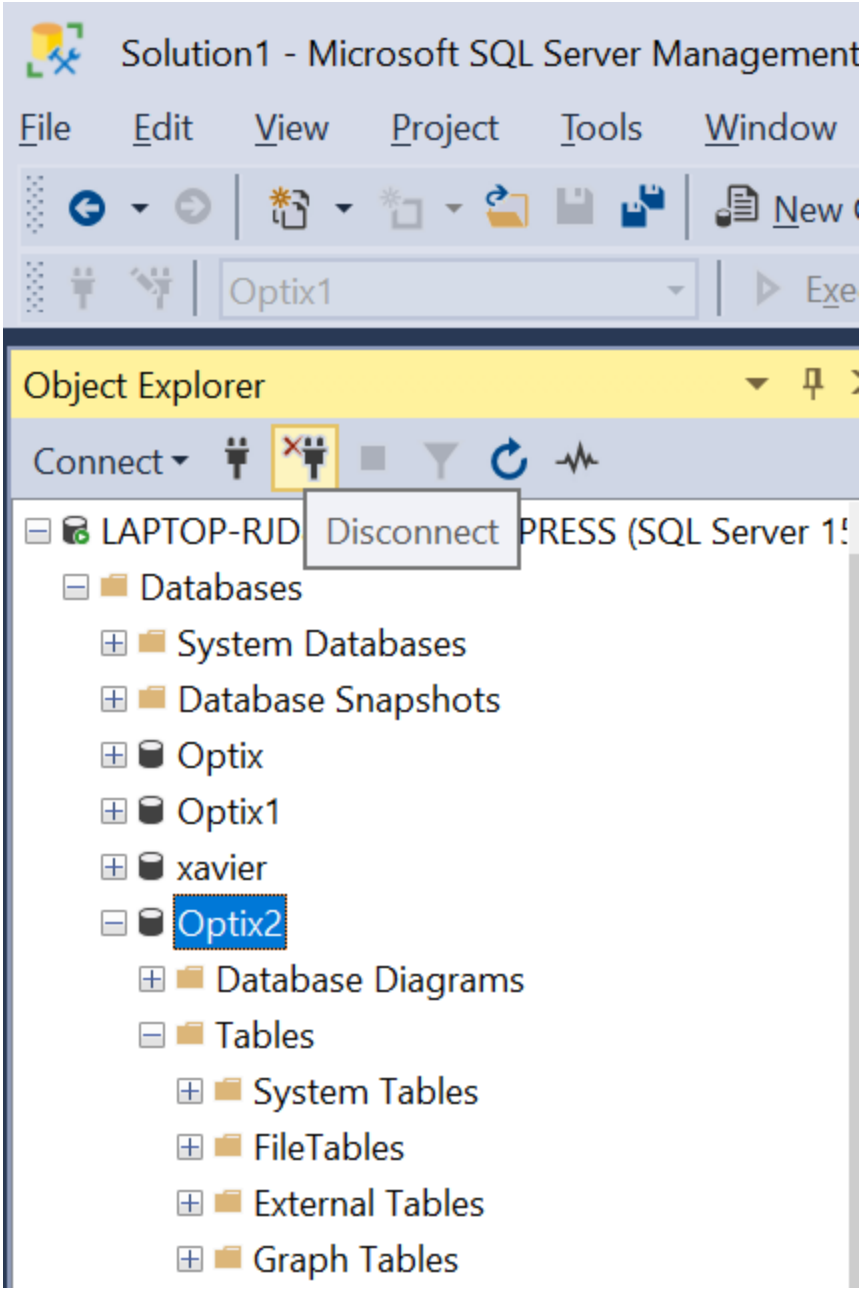
And select the variable to log and the destination, just the database

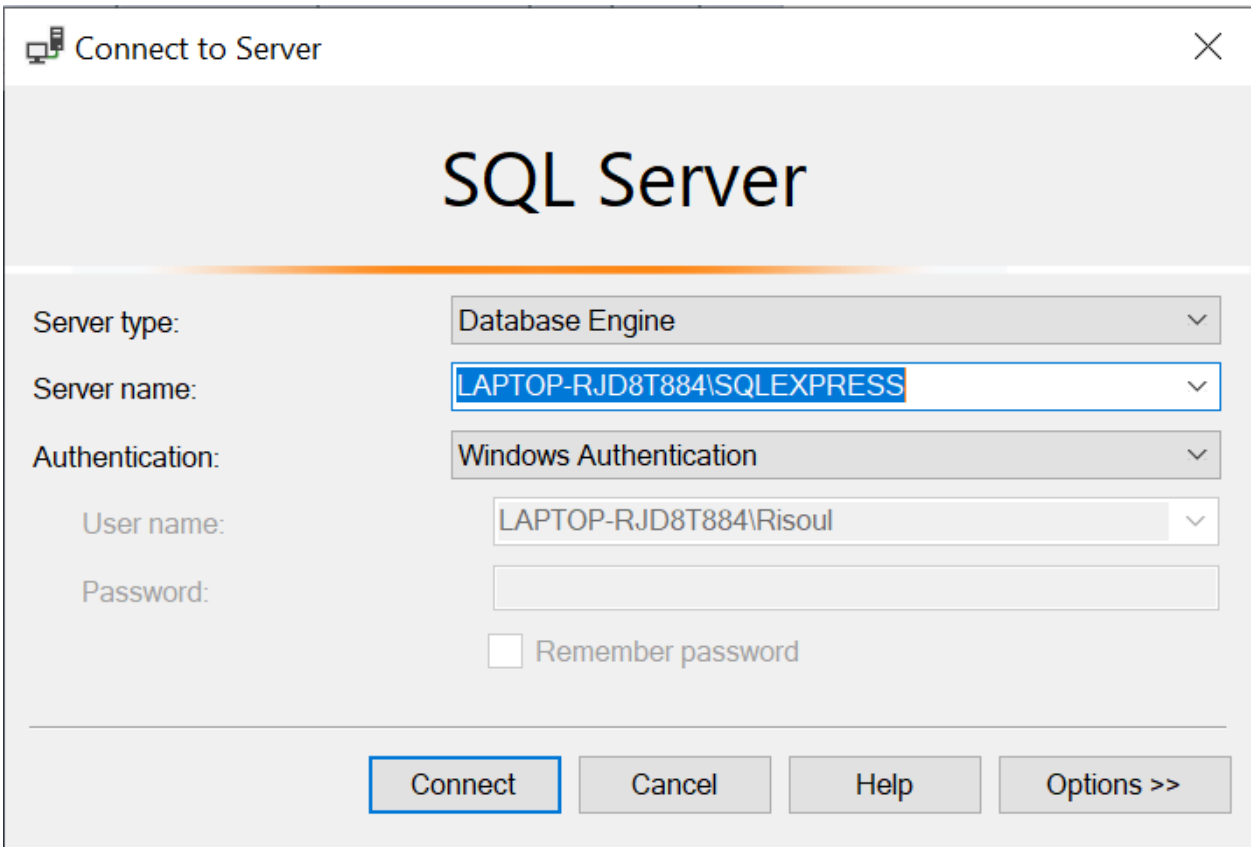


If we now open the ODBC object, surprise, a new Table an columns have been created for us

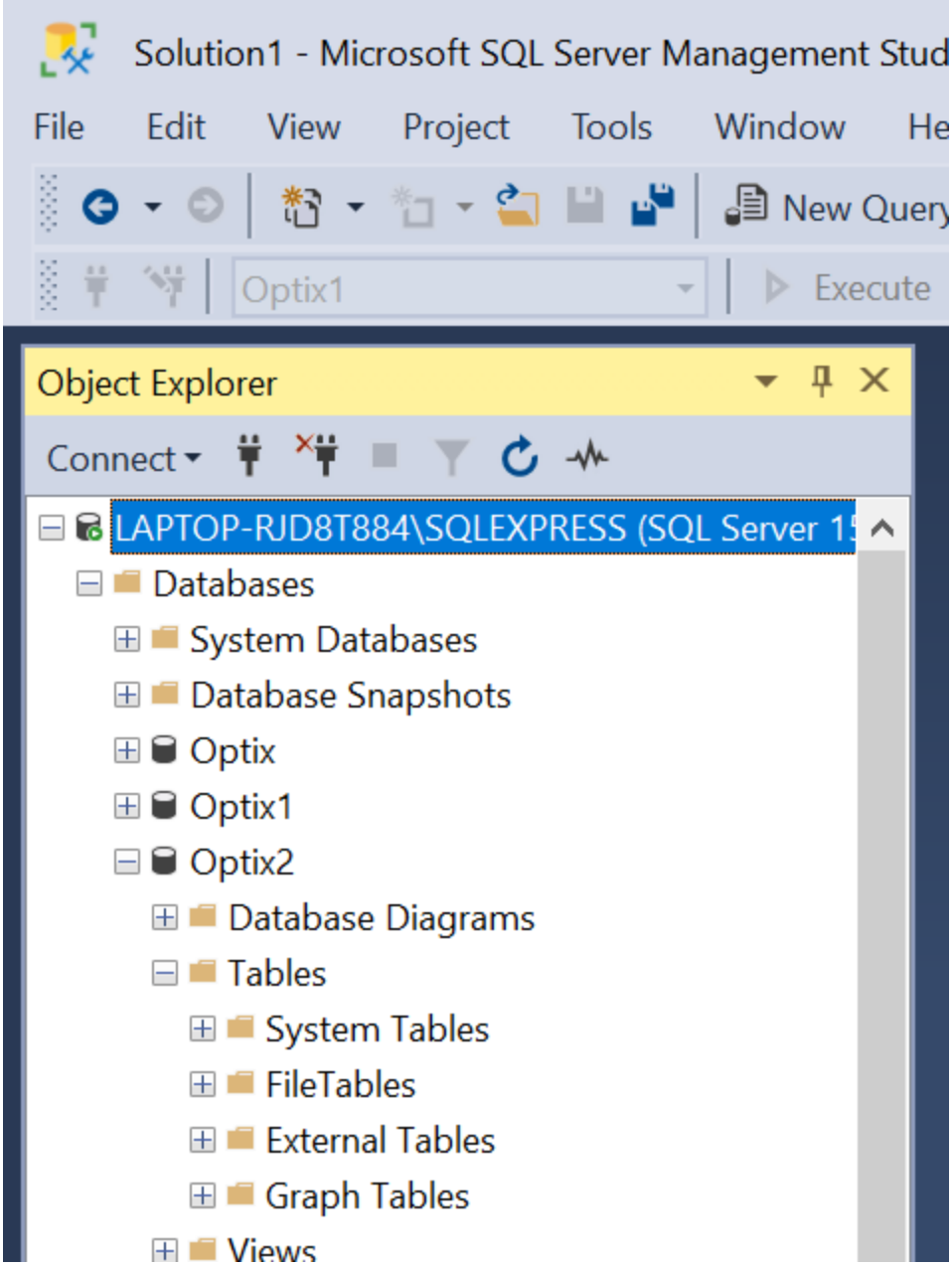


If we take a look at the database:
You have to disconnect and connect again to see the changes

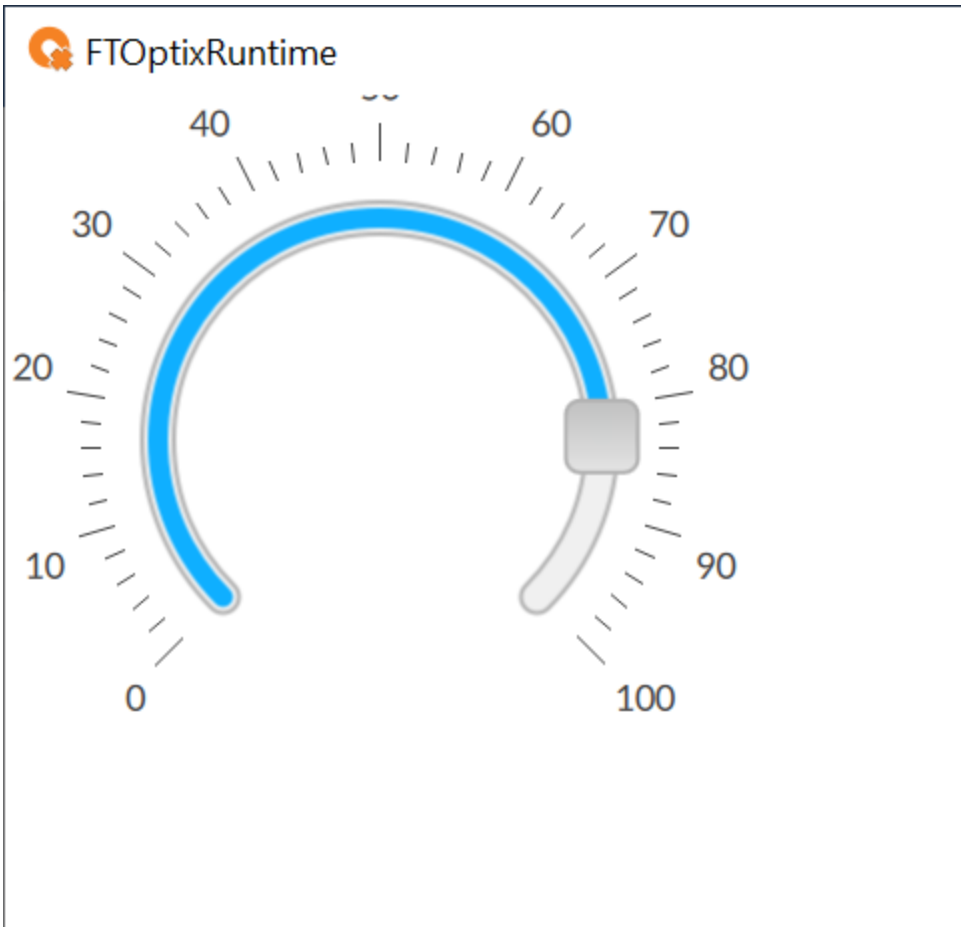

Solution1 - Microsoft SQL Server Management
File
Edit
View
Project
Tools
Window
New
Optix1
Exe
Object Explorer
Connect
Disconnect
LAPTOP-RJD
PRESS (SQL Server 1
Databases
System Databases
Database Snapshots
Optix
Optix1
xavier
Optix2
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables



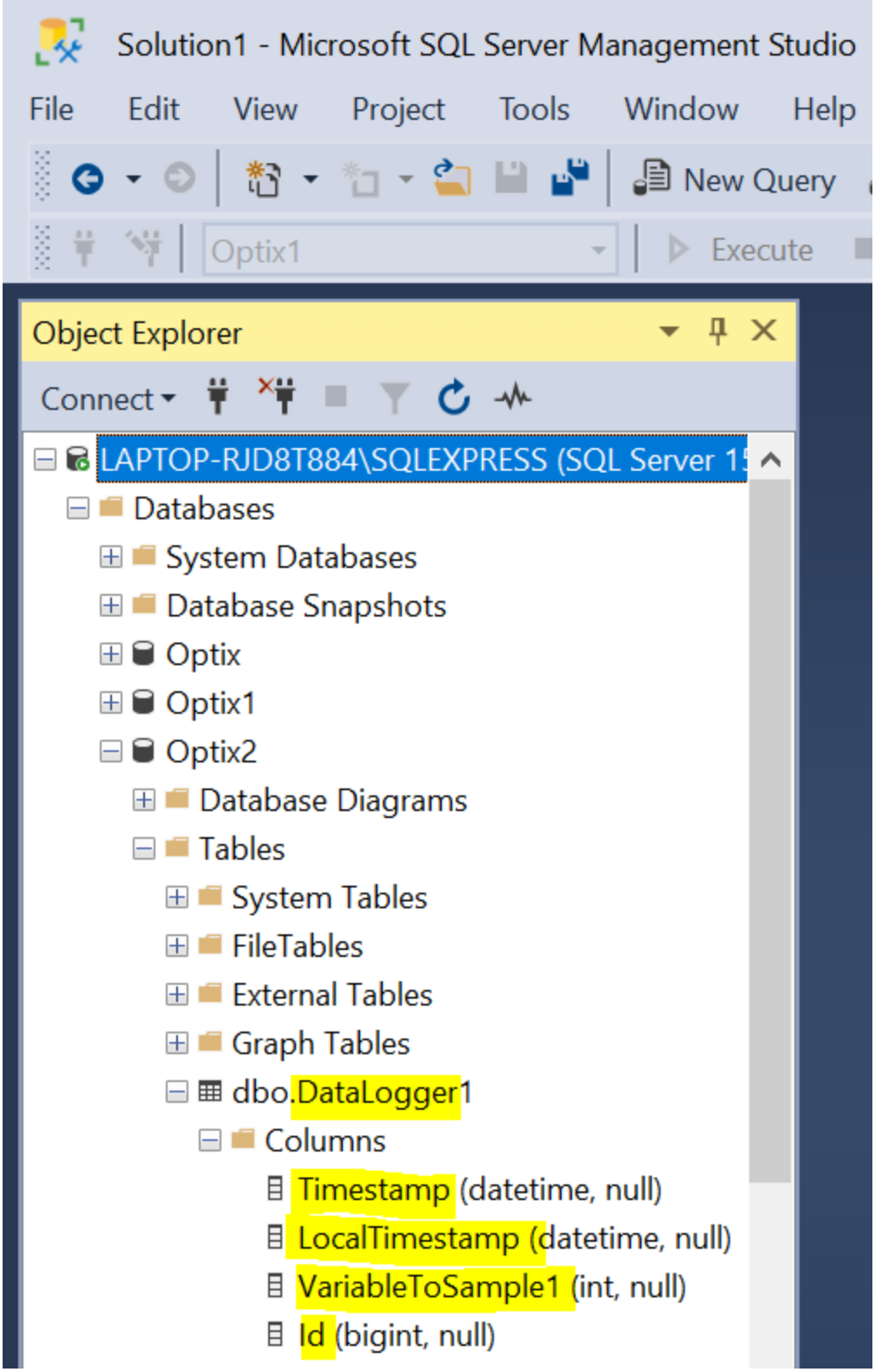
Still nothing



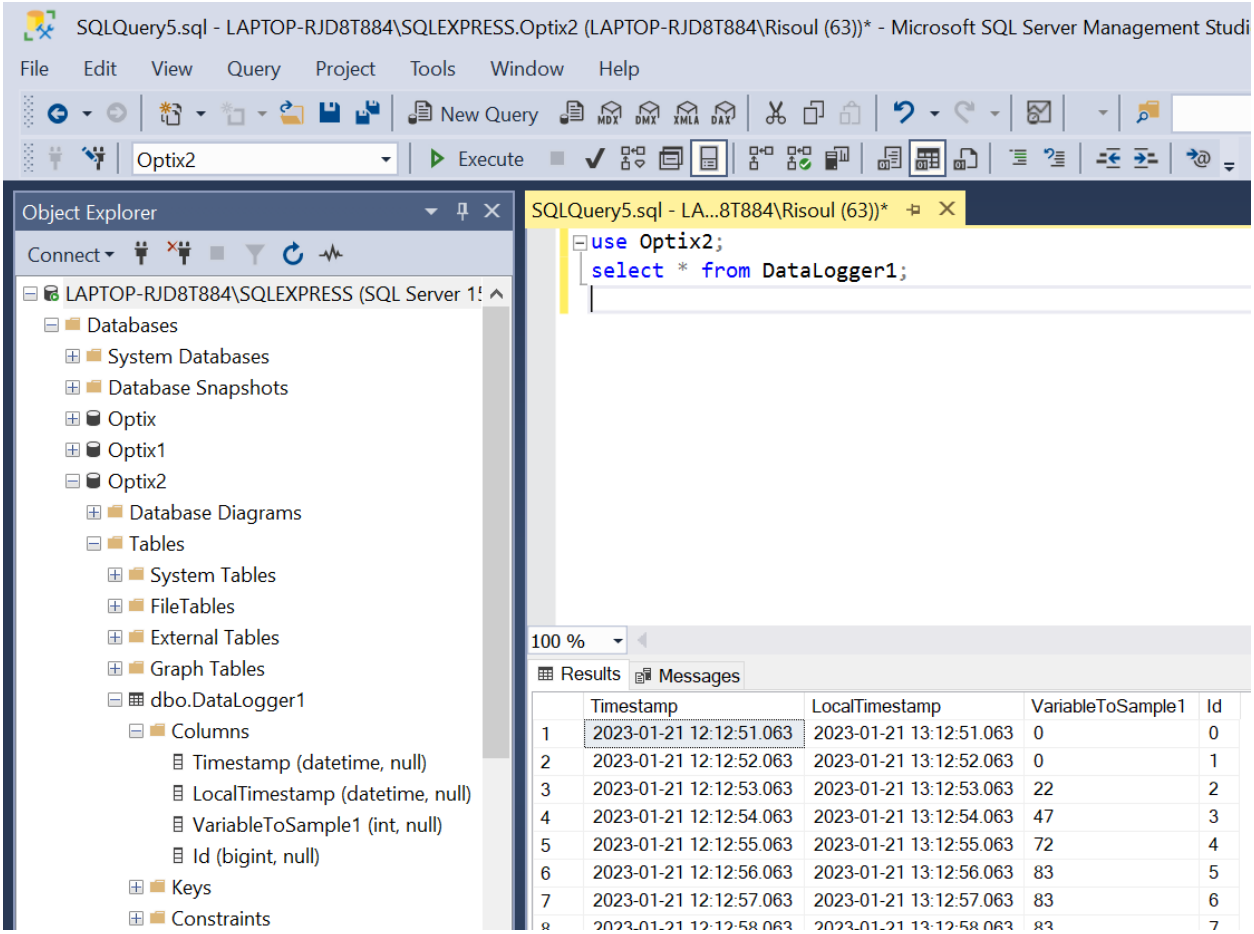
Until you run the Optix Runtime



Give some values to the gauge dropping the index
Then go to the Database, and …voilà

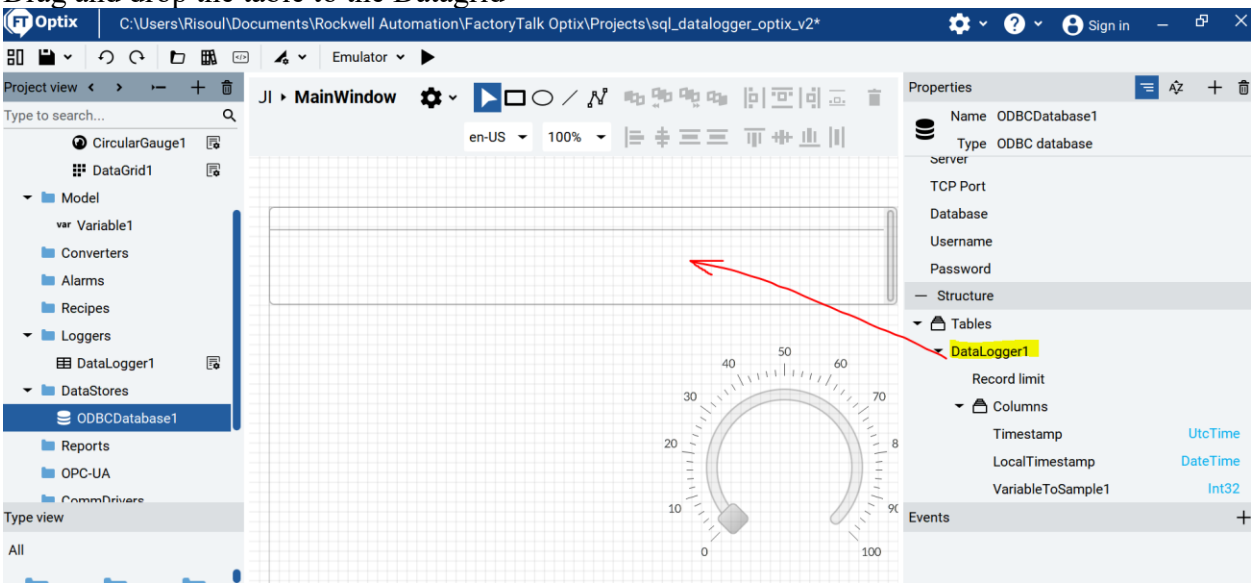


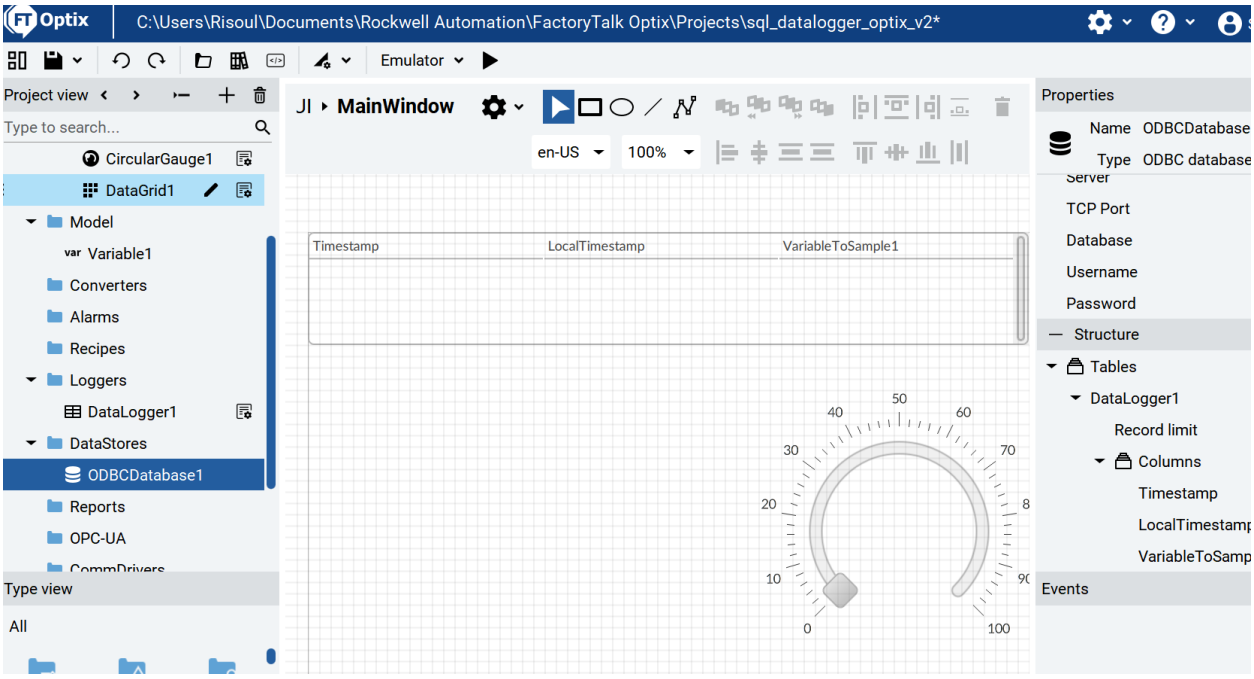
Now let's look at the data, this is working fine



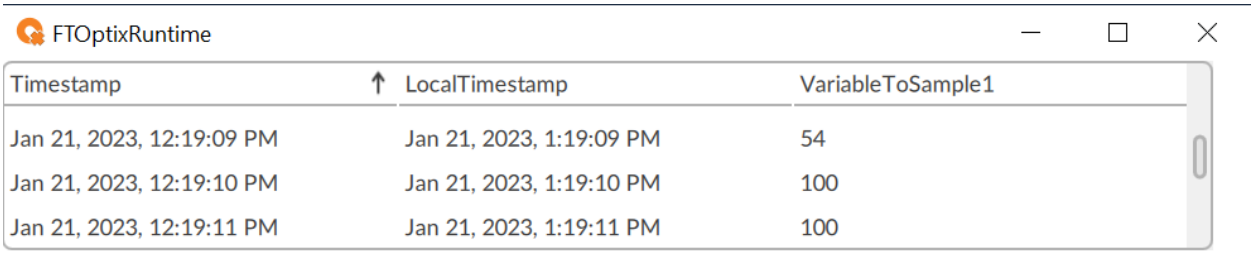
Now let's add a Datagrid on the Optix Project

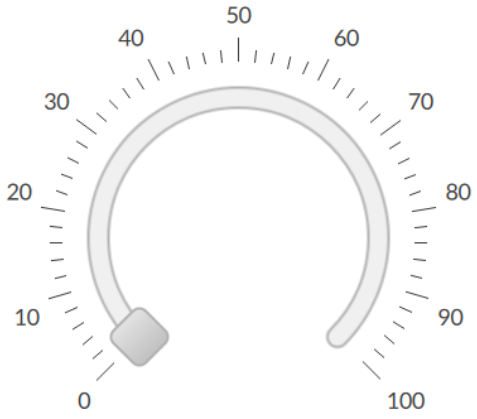
Drag and drop the table to the Datagrid





Let's test it. It works! That's all

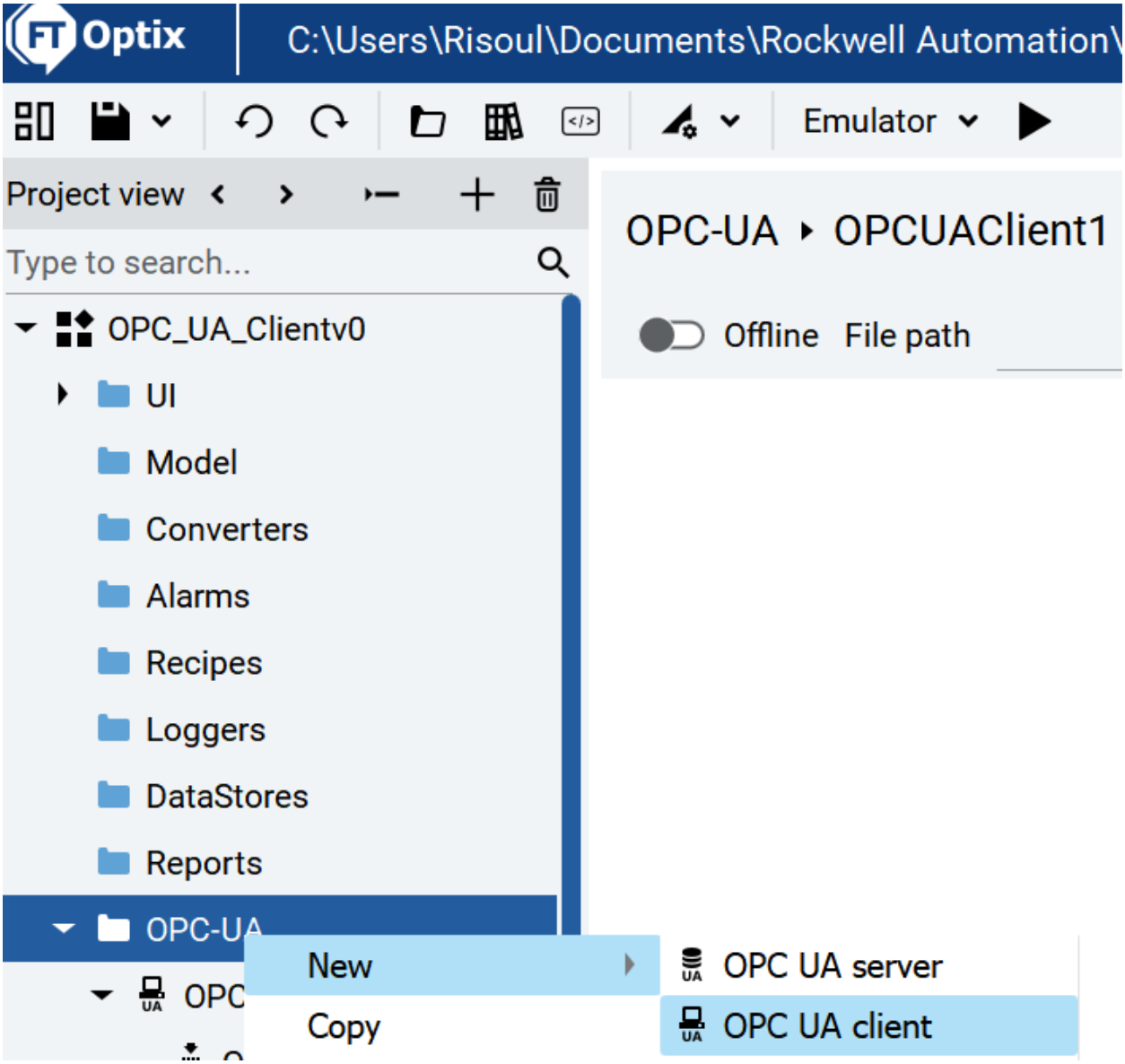




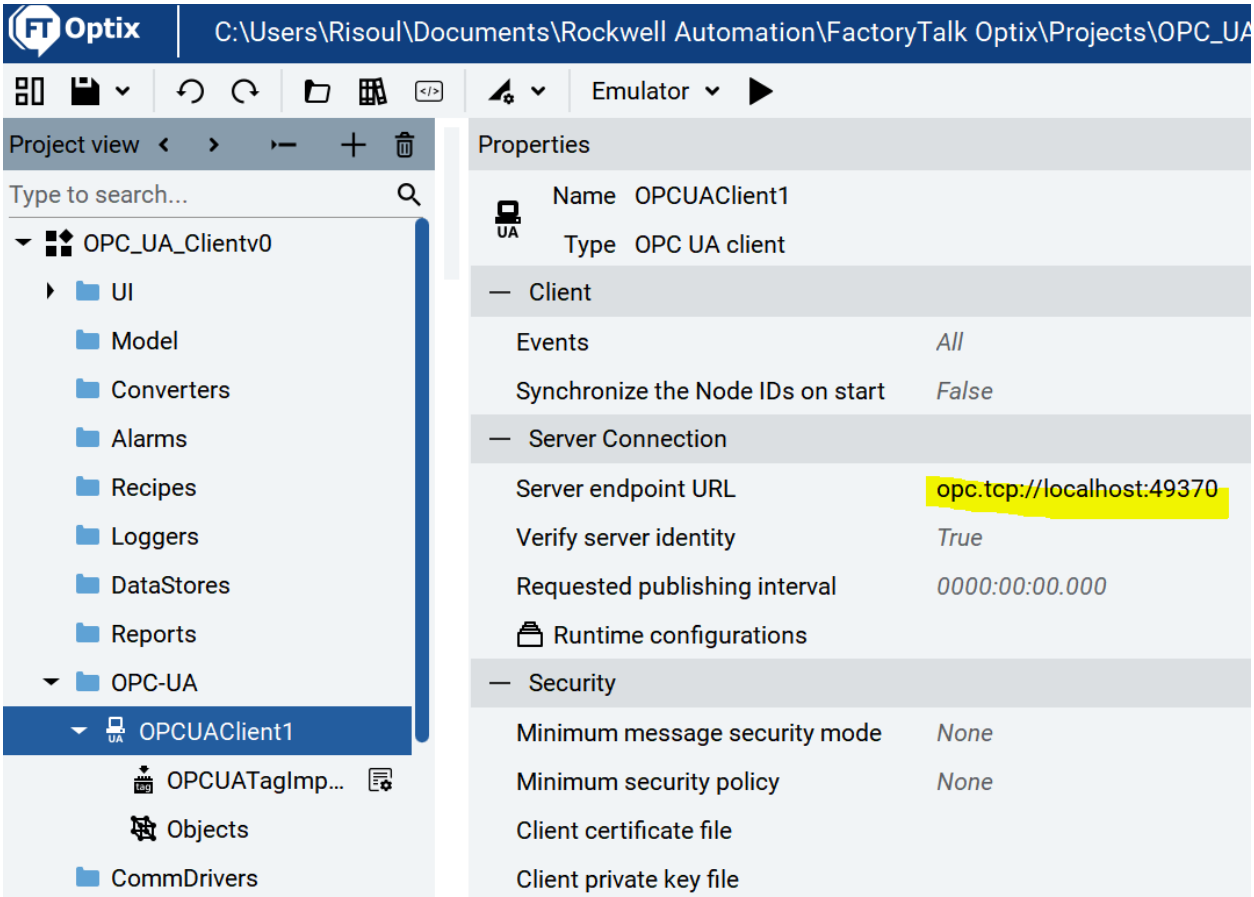
16. OPC UA Client

Let's create a new OPC Client

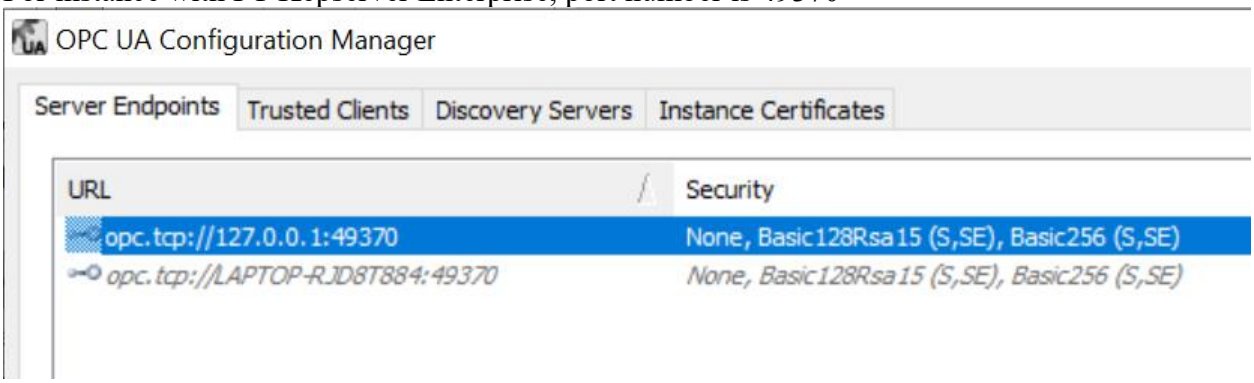
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/using-the-software/opc-ua/manage-the-opc-ua-client/add-an-opc-ua-client-object.html



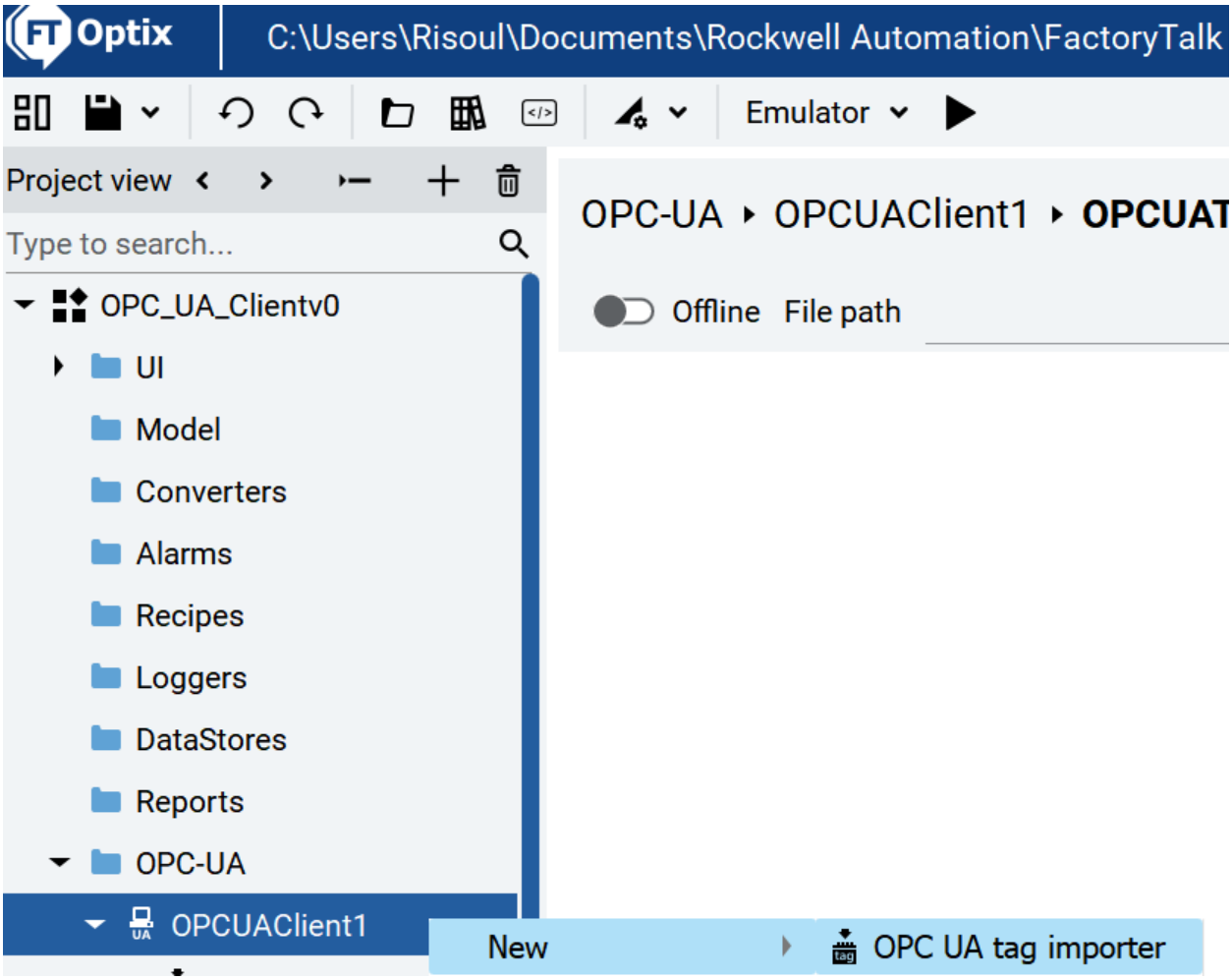
Change the address and port of the OPC UA server



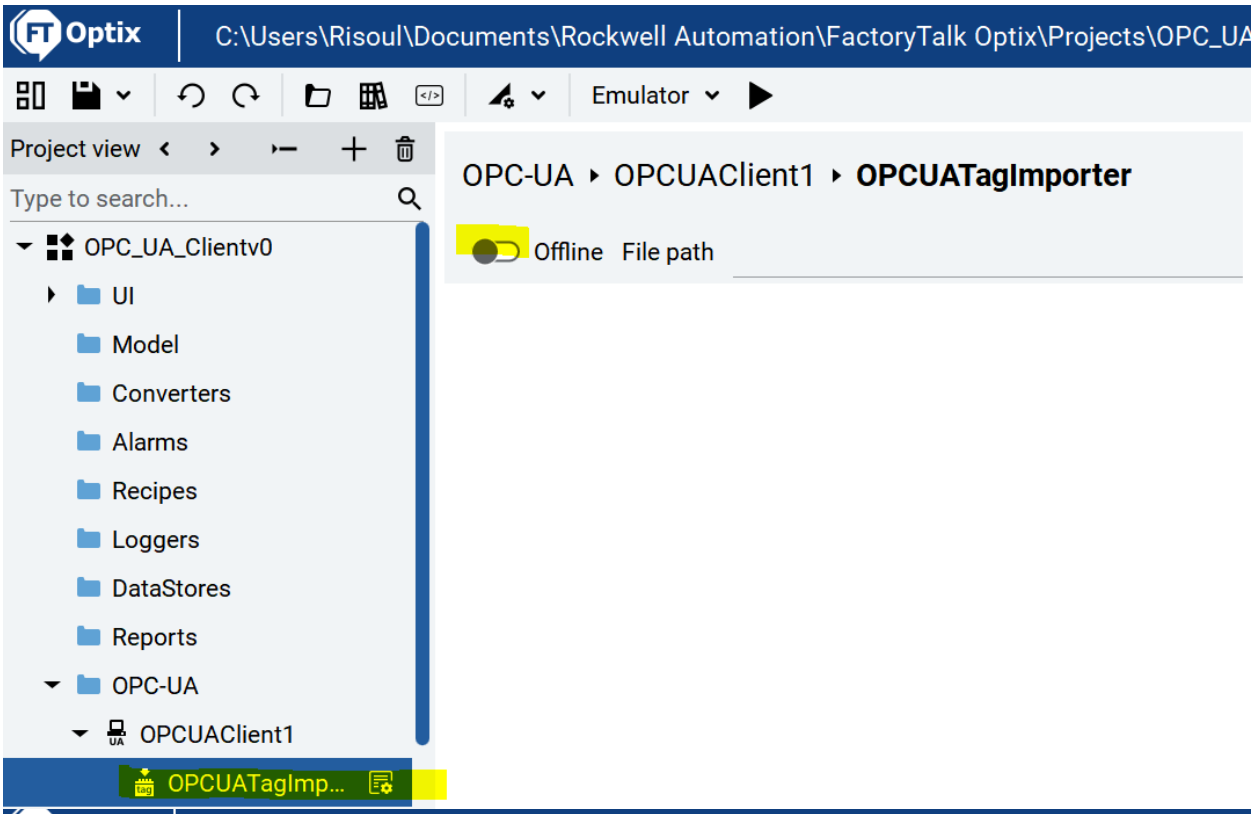
For instance with FT Kepserver Enterprise, port number is 49370

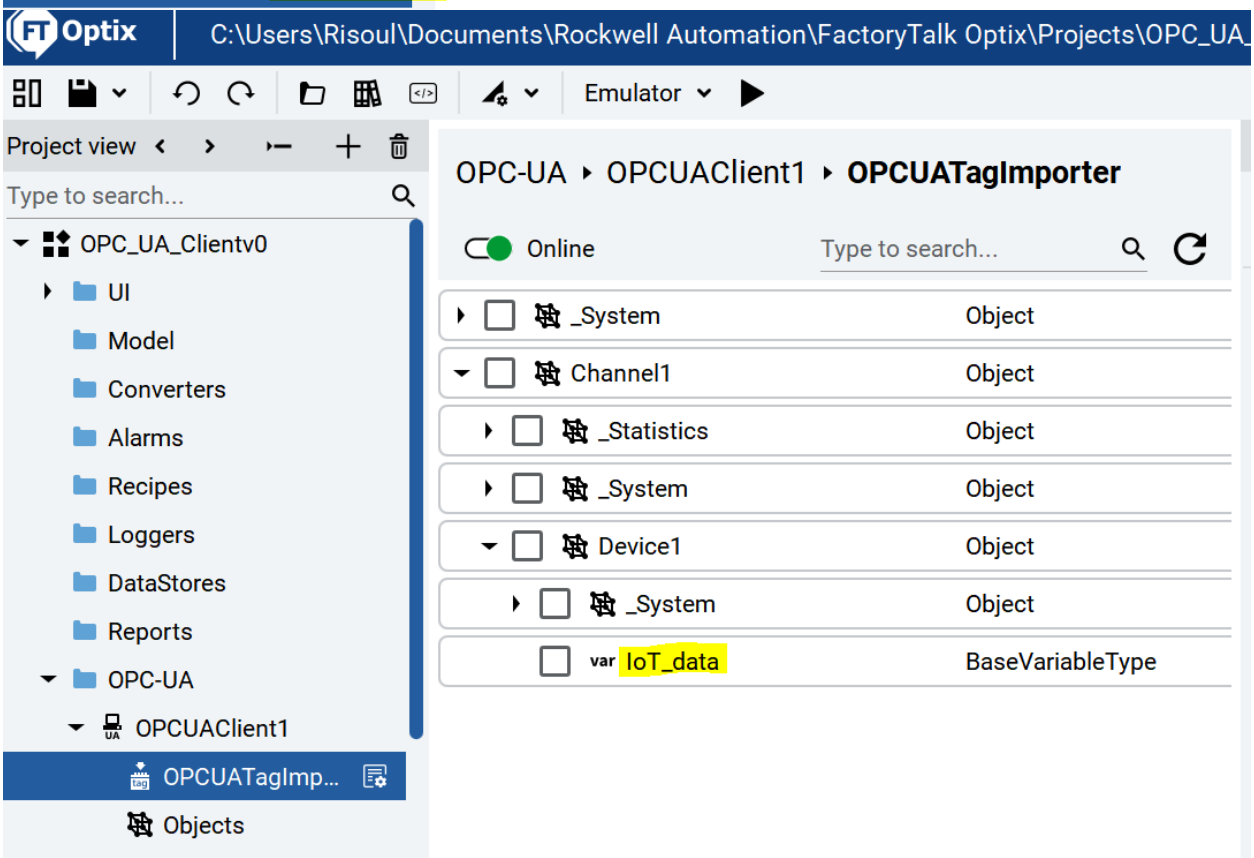


Right click to add a new OPCUA Tag Importer
Then double click on it

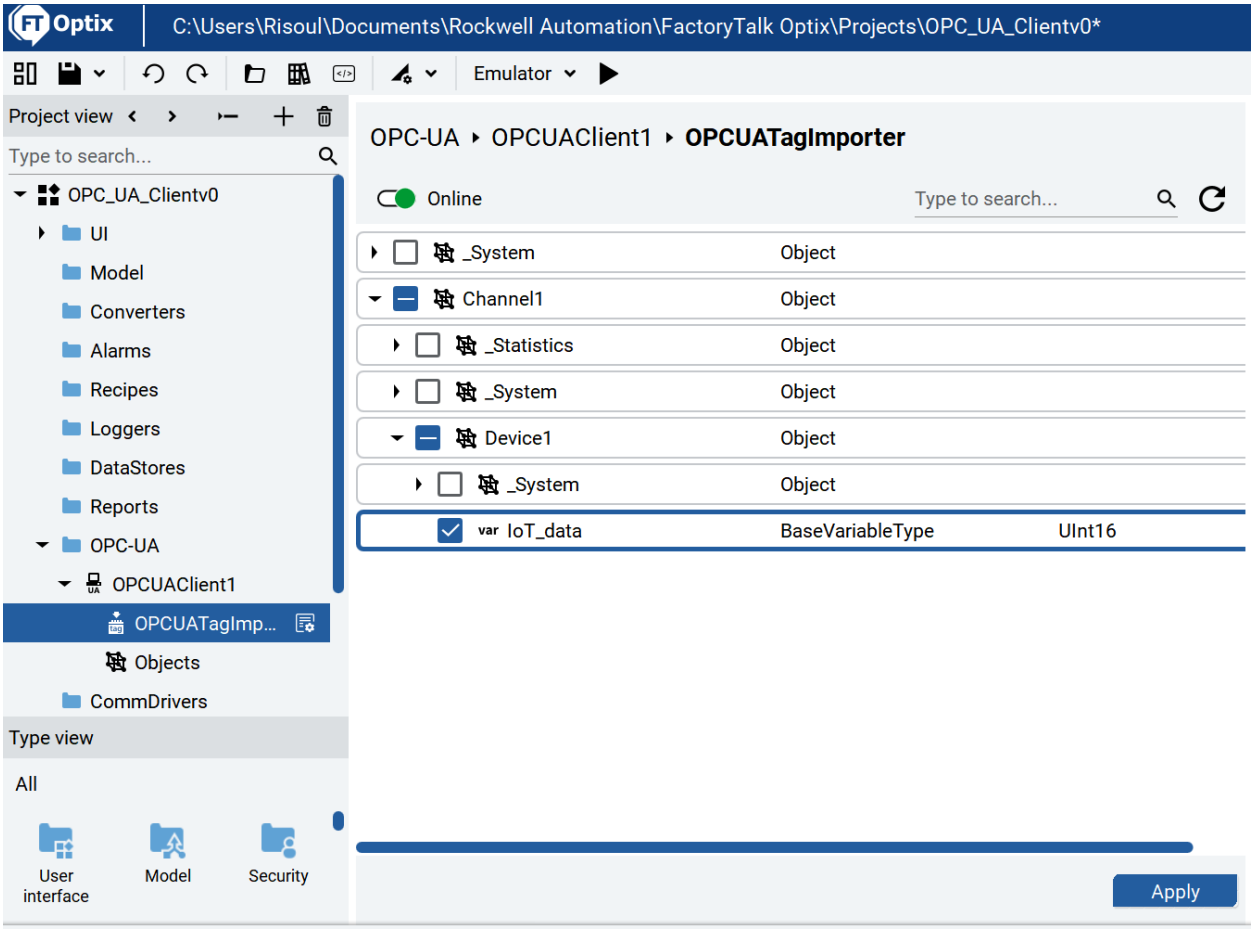


Click on the Button to go online with the OPC server

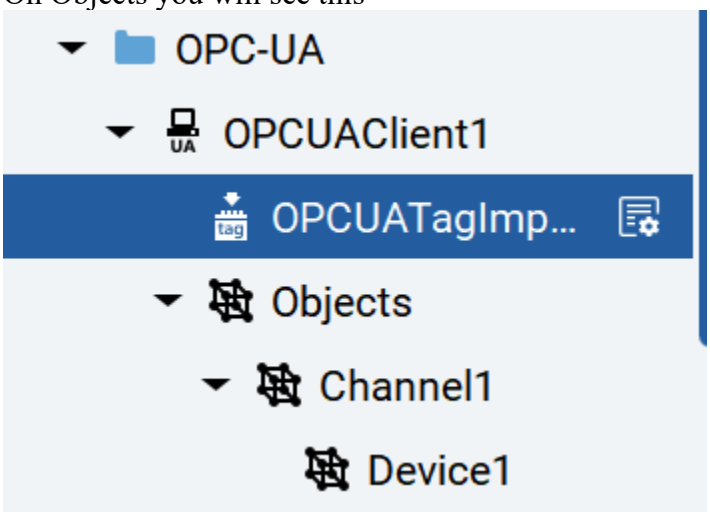




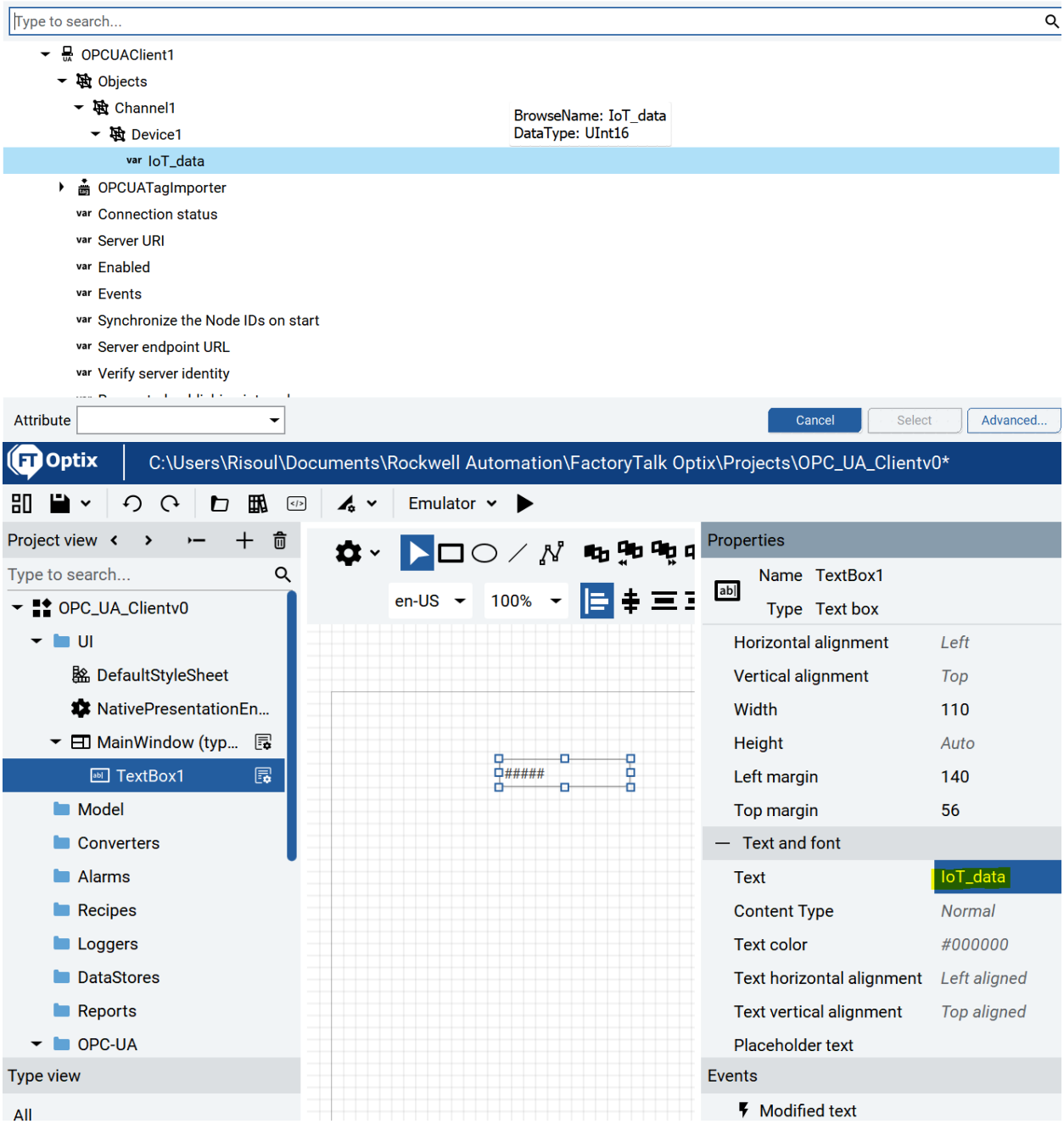
Click on apply



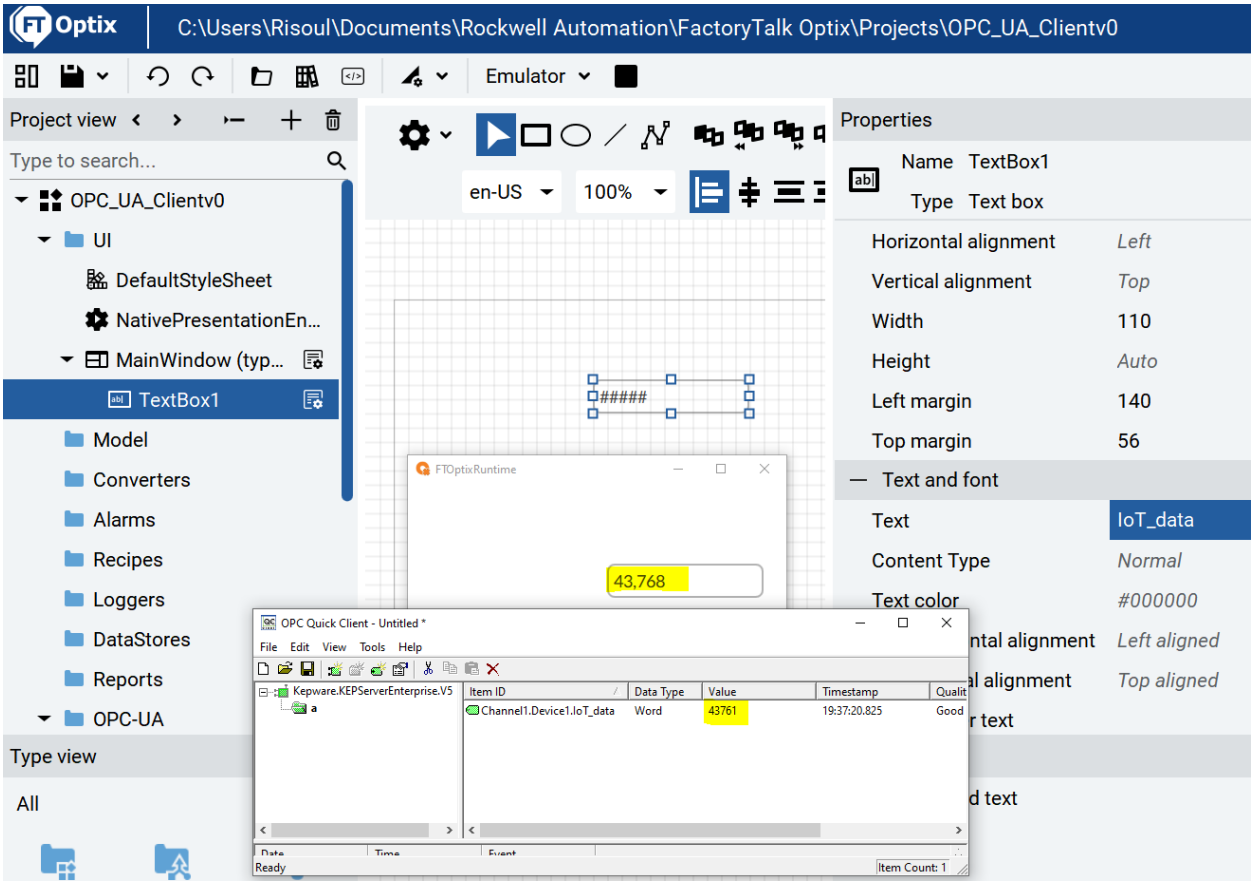
On Objects you will see this



Now let's create a texbox that points to the data


Type to search...
OPCUAClient1
Objects
Channel1
Device1
IoT_data
BrowseName: IoT_data
DataType: UInt16
OPCUATagImporter
Connection status
Server URI
Enabled
Events
Synchronize the Node IDs on start
Server endpoint URL
Verify server identity
Attribute
Cancel
Select
Advanced...
Optix
C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\OPC_UA_Clientv0*
Emulator
Project view
Type to search...
OPC_UA_Clientv0
UI
DefaultStyleSheet
NativePresentationEn...
MainWindow (typ...
TextBox1
Model
Converters
Alarms
Recipes
Loggers
DataStores
Reports
OPC-UA
Type view
All
en-US
100%
#####
Properties
Name TextBox1
Type Text box
Horizontal alignment Left
Vertical alignment Top
Width 110
Height Auto
Left margin 140
Top margin 56
Text and font
Text IoT_data
Content Type Normal
Text color #000000
Text horizontal alignment Left aligned
Text vertical alignment Top aligned
Placeholder text
Events
Modified text

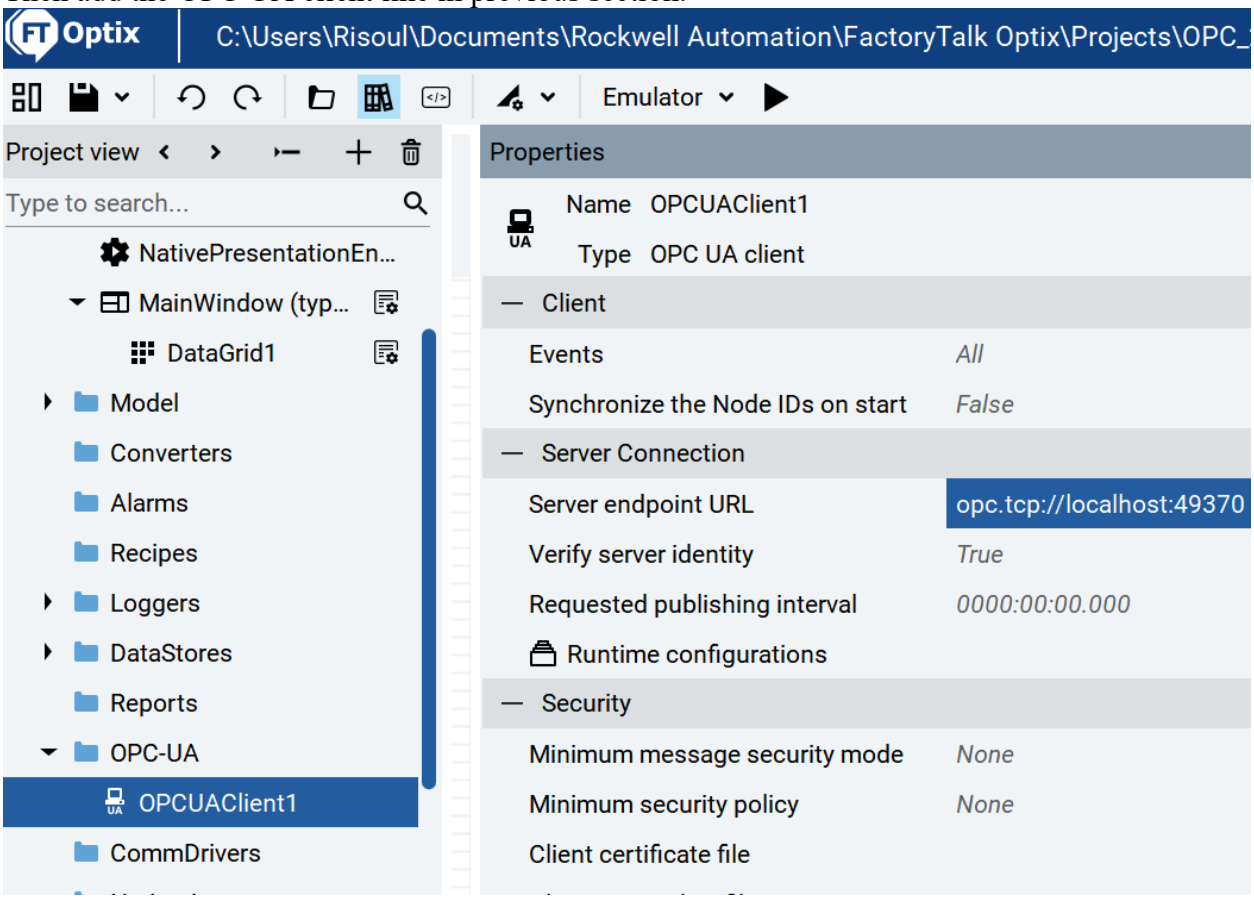
That's all



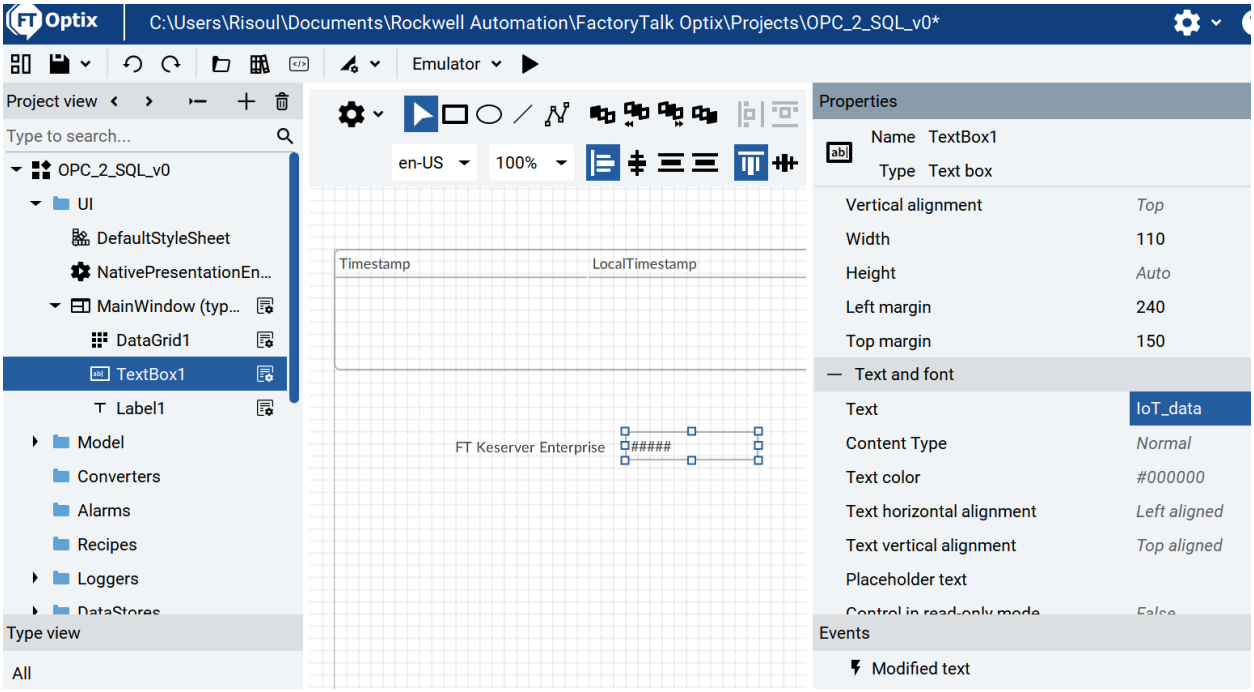
Let's add a label

17. Creating a OPC UA to SQL converter

You can see the results here
https://youtu.be/Bcs8rT0ytUQ
Take last SQL project and delete the gauge.
Then add the OPC UA client like in previous section.

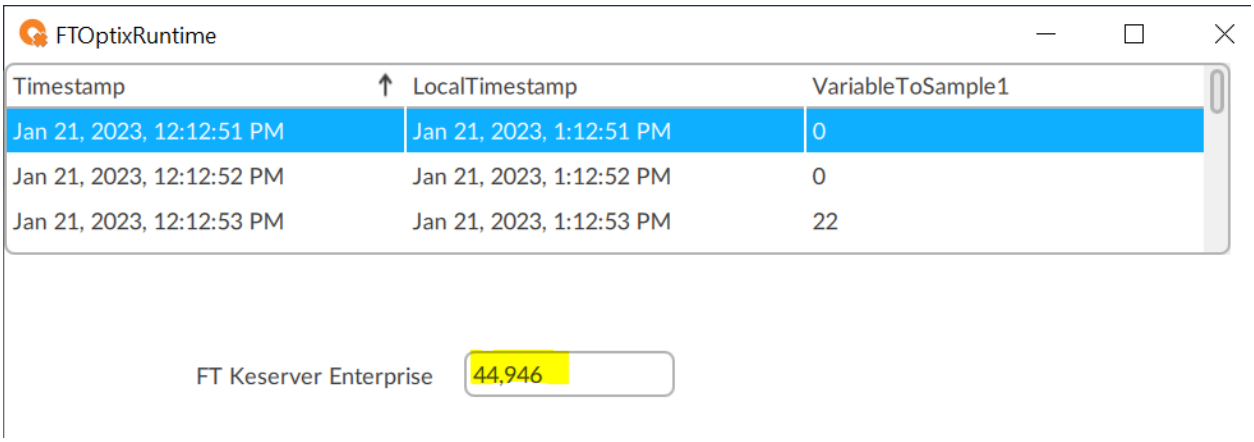


Now we have access to IoT_data Tag

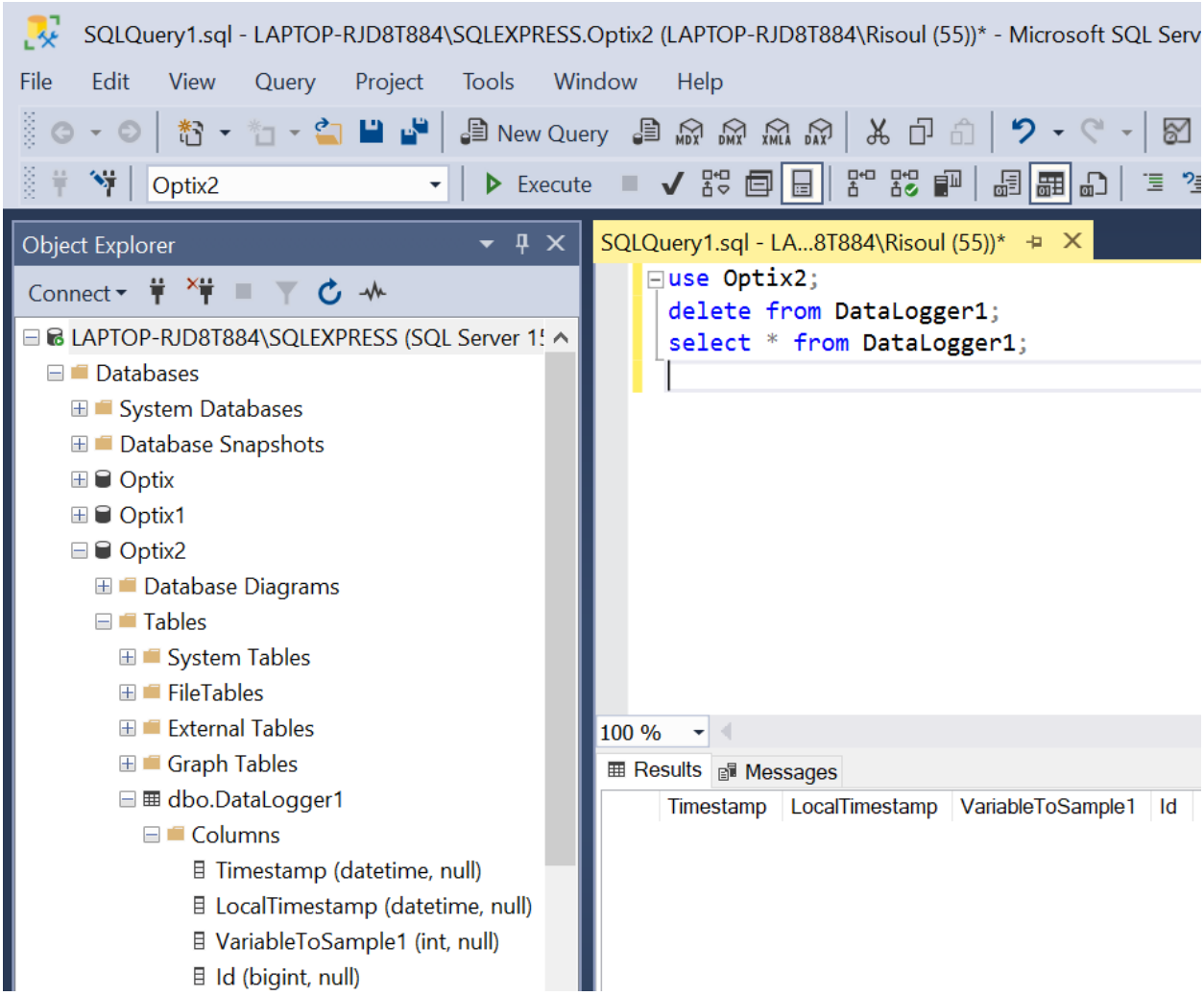


First of all test that we are reading OPC UA data

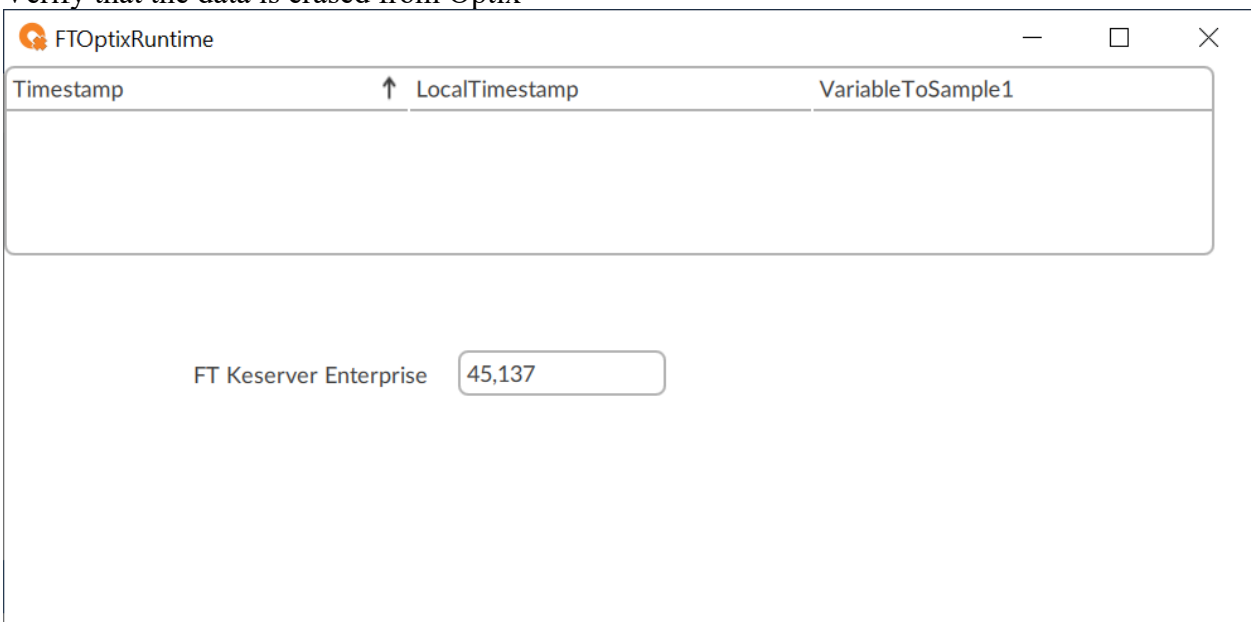
Yes



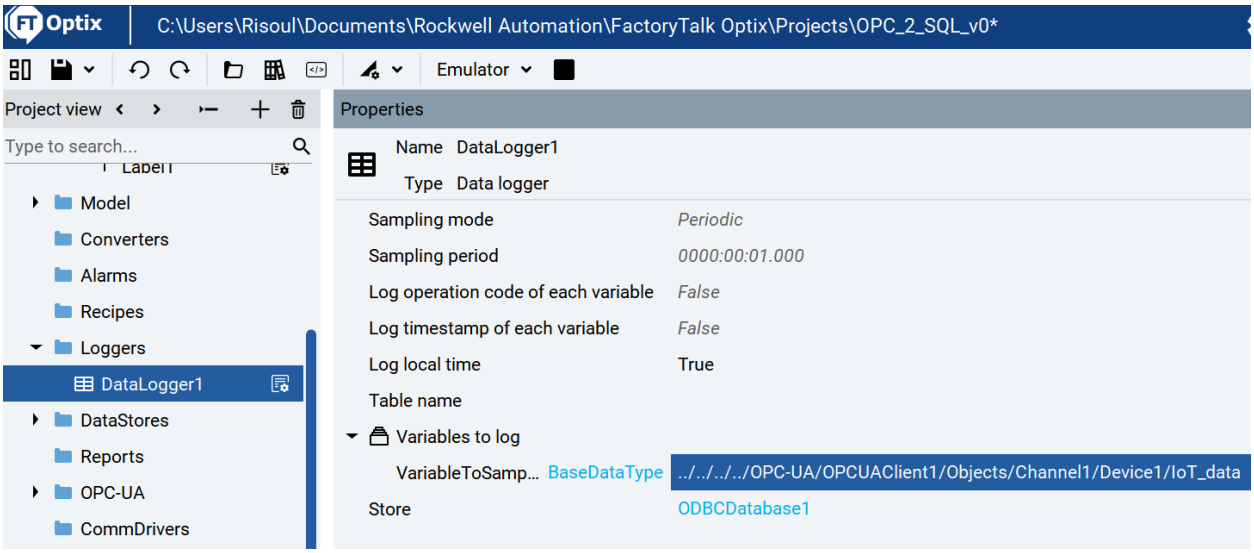
Then let's erase the data on the database from previous chapter.



Verify that the data is erased from Optix

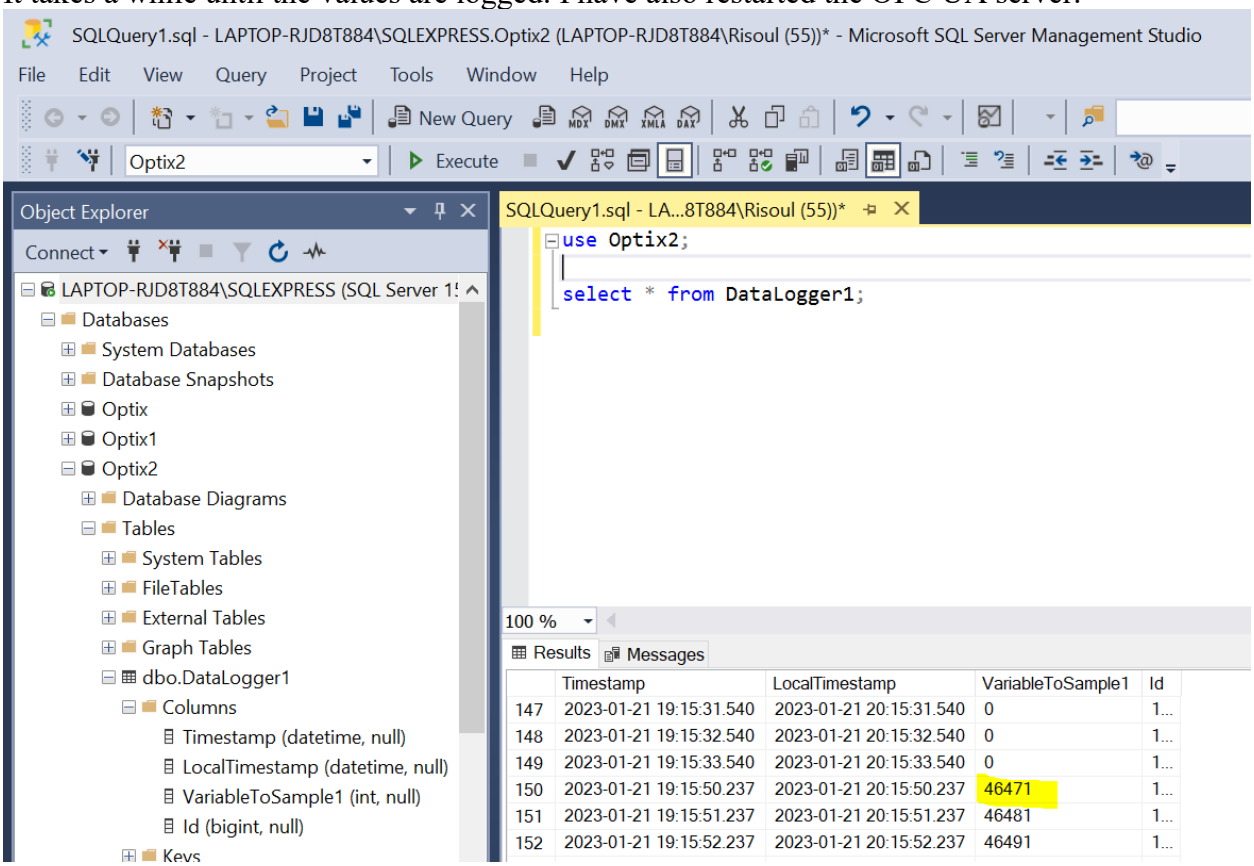


Now just change the source of data on the datalogger Object



Let's test The Optix Runtime

It takes a while until the values are logged. I have also restarted the OPC UA server.

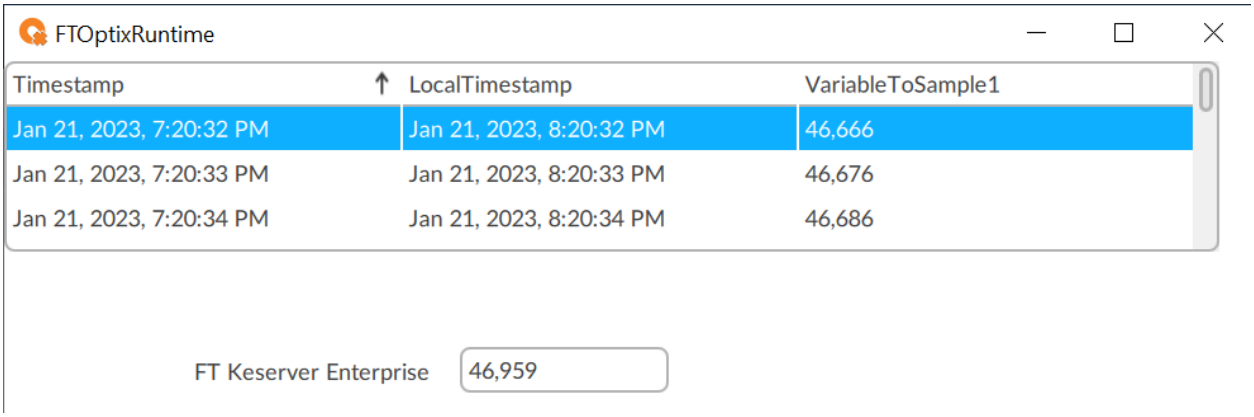


Clear the Table again

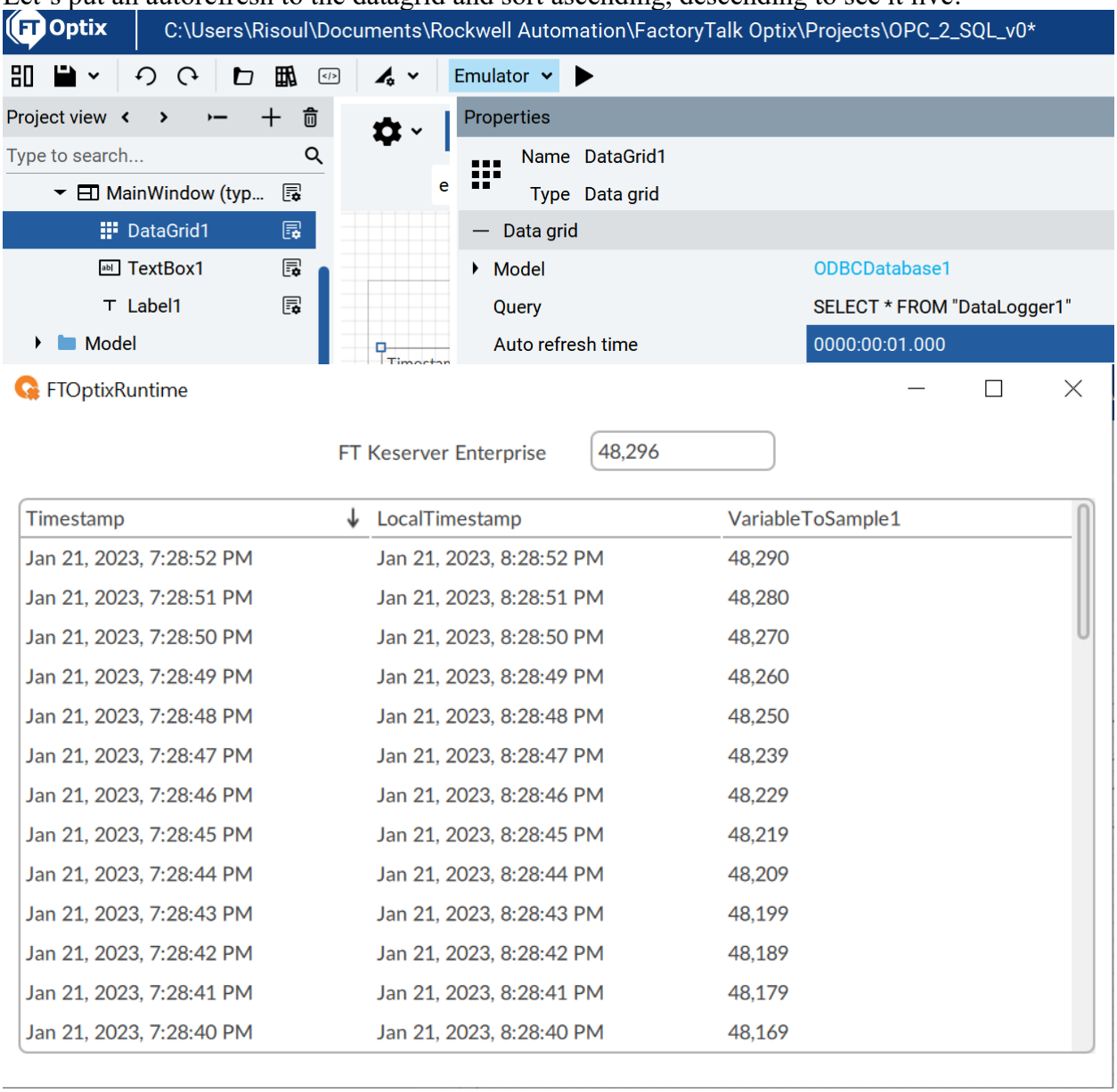
First time we see nothing on the Optix runtime datagrid window

Let's close the runtime and open it again

Now it works



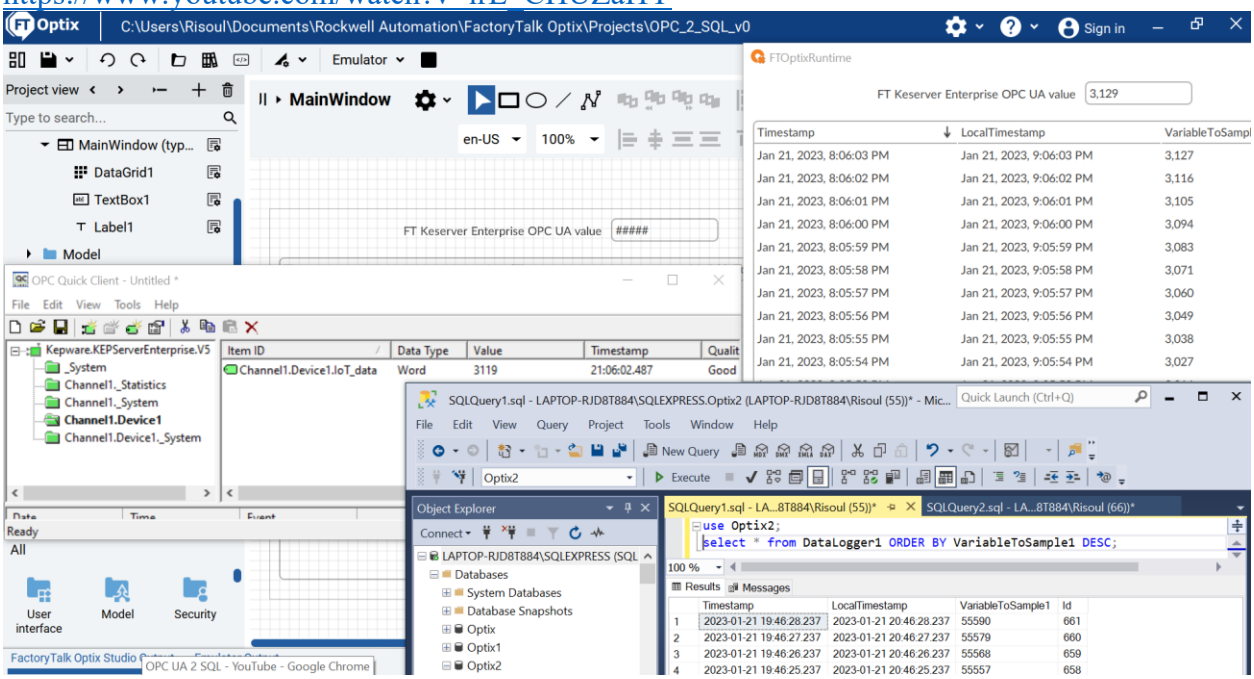
Let's put an autorefresh to the datagrid and sort ascending, descending to see it live!



As you can see on this video

https://youtu.be/Bcs8rT0ytUQ

https://www.youtube.com/watch?v=irL_CHUZaHY



18. Configuring an application as an MQTT client

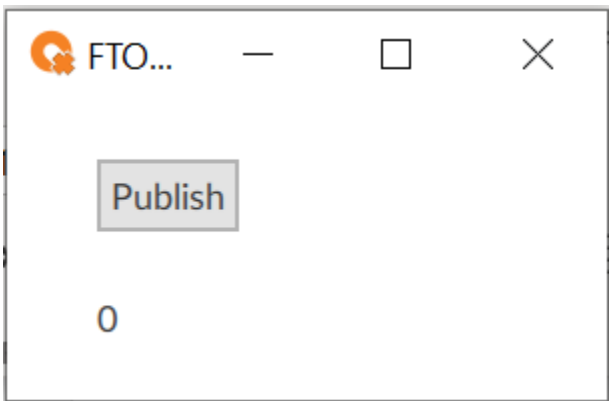
This example is using a cloud public broker like test.mosquitto.org
The Optix application is both subscribed and publishing to the same topic
Go to

https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/application-examples/netlogic-tutorial/configure-an-application-as-an-mqtt-client.html

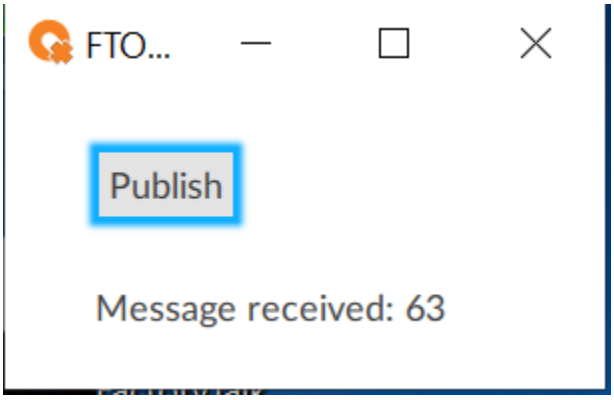
You can download the application from here
https://www.rockwellautomation.com/content/dam/dita/en/factorytalk-optix/1-00/downloads/MQTTClient.zip

Tip: You can download a sample project from:
https://github.com/xavierflorensa/MQTTClient_FTOptix_help

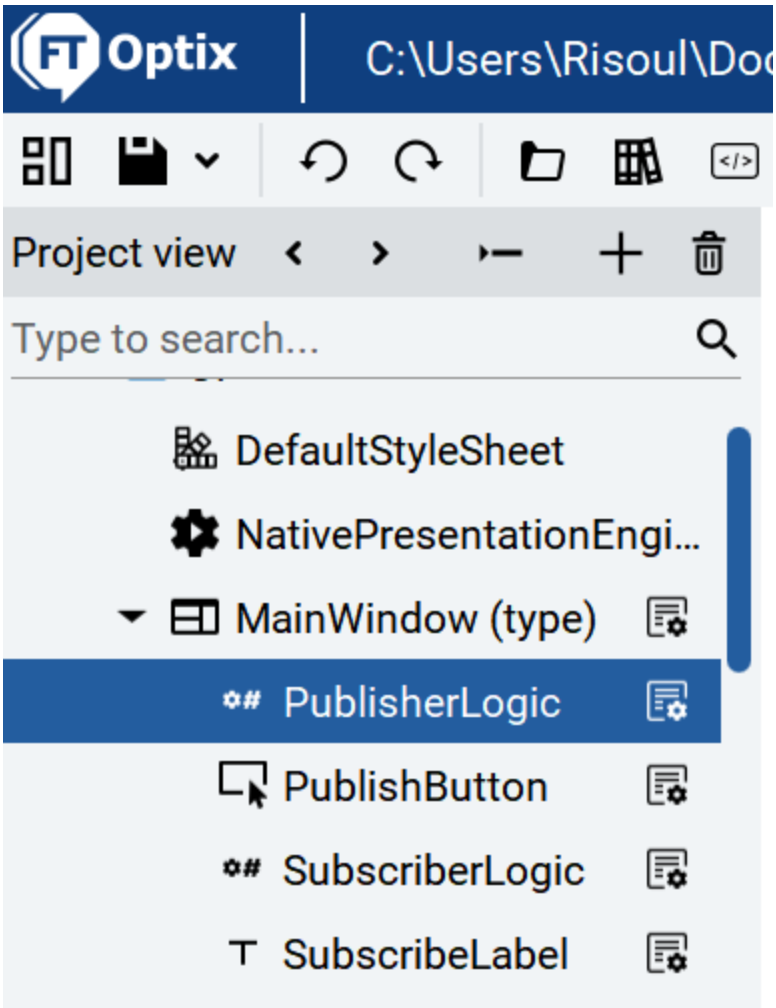
Let's try opening the example



Click on “Publish”
It works



But let’s look at the details
Like PublisherLogic
Doubleclick on PublisherLogic



Visual Studio code opens

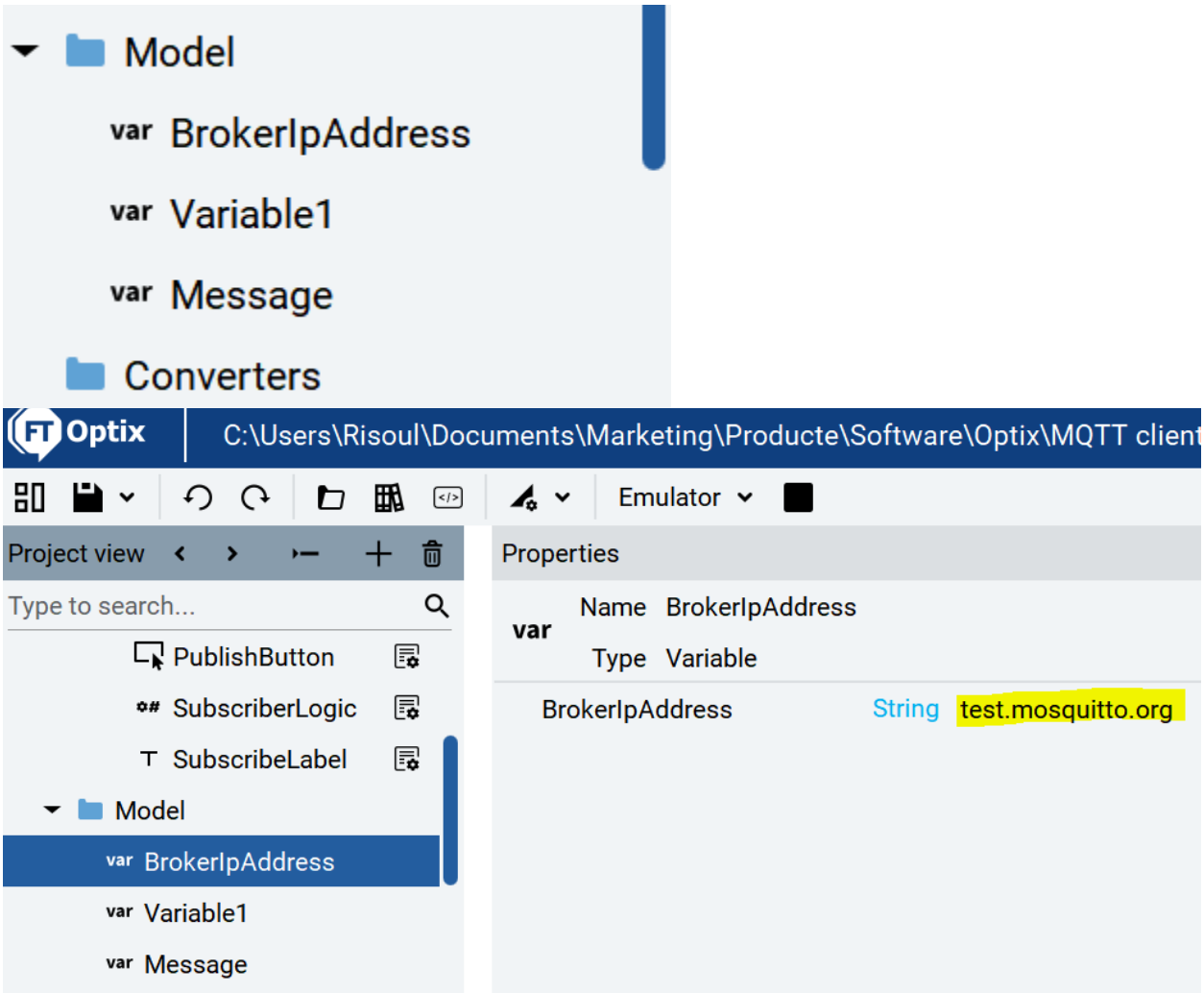
File Edit Selection View Go Run ··· MQTTClient (Workspace)

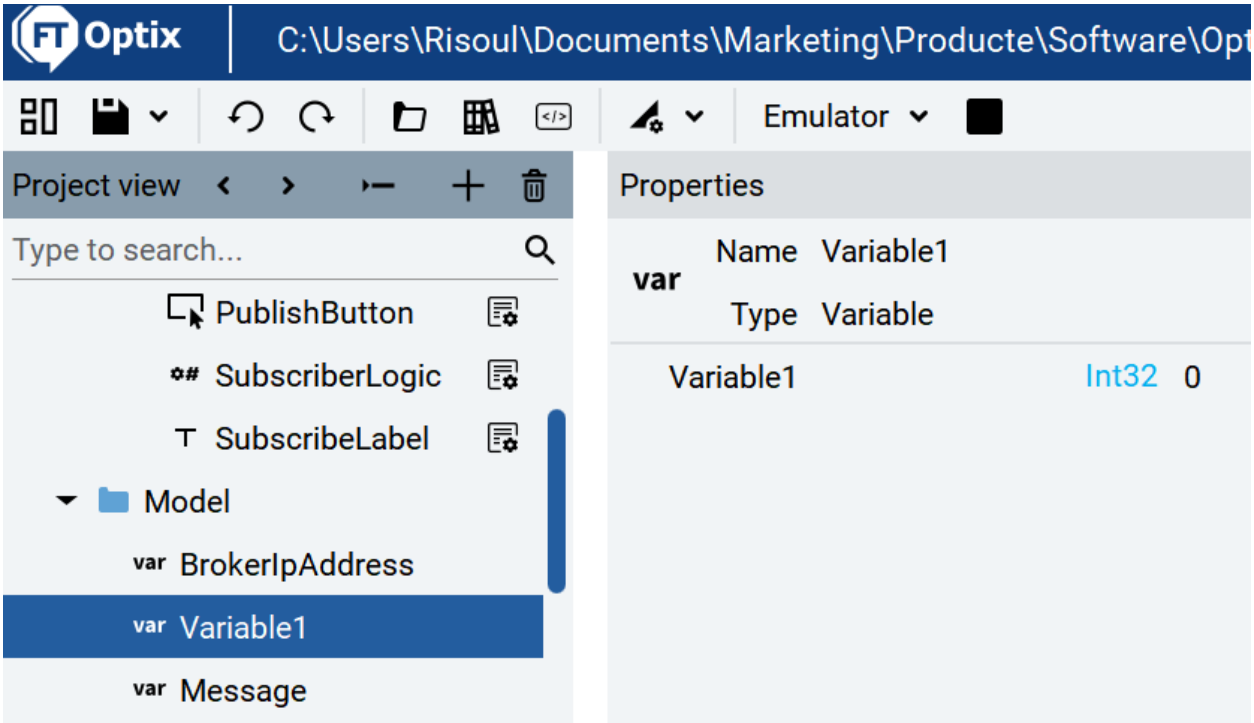
PublisherLogic.cs

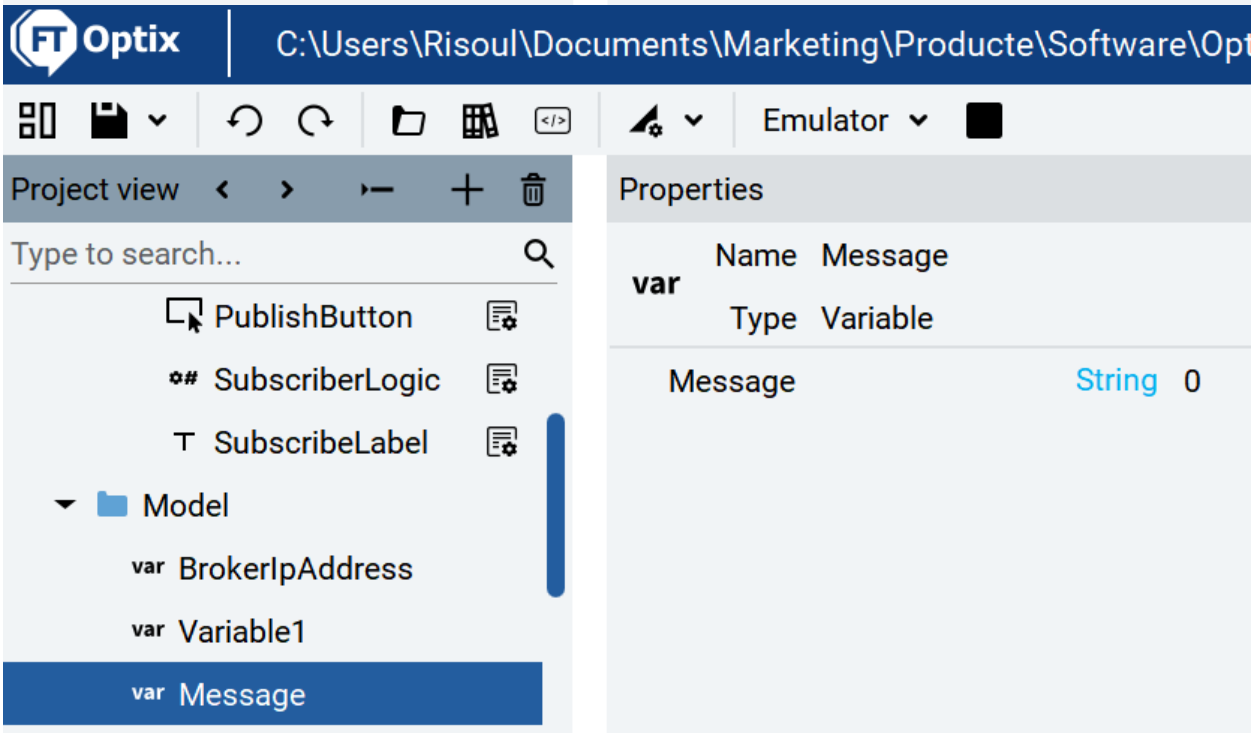
NetSolution > PublisherLogic.cs

```
#region StandardUsing
using System;
using FTOptix.CoreBase;
using FTOptix.HMIProject;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

public class PublisherLogic : BaseNetLogic
{
    public override void Start()
    {
        var brokerIpAddressVariable = Project.Current.GetVariable("Model/BrokerIpAddress");

        // Create a client connecting to the broker (default port is 1883)
        publishClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        publishClient.Connect("FTOptixPublishClient");
        // Assign a callback to be executed when a message is published to the broker
        publishClient.MqttMsgPublished += PublishClientMqttMsgPublished;
    }

    public override void Stop()
    {
        publishClient.Disconnect();
        publishClient.MqttMsgPublished -= PublishClientMqttMsgPublished;
    }

    private void PublishClientMqttMsgPublished(object sender, MqttMsgPublishedEventArgs e)
    {
        Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
    }

    [ExportMethod]
    public void PublishMessage()
    {
        var variable1 = Project.Current.GetVariable("Model/Variable1");
        variable1.Value = new Random().Next(0, 101);

        // Publish a message
        ushort msgId = publishClient.Publish("/my_topic", // topic
            System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString()), // message body
            MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
            false); // retained
    }

    private MqttClient publishClient;
}
```

Look at BrokerIpAddress under UI


Model
var BrokerIpAddress
var Variable1
var Message
Converters
Optix
C:\Users\Risoul\Documents\Marketing\Producte\Software\Optix\MQTT client
Emulator
Project view
Type to search...
PublishButton
SubscriberLogic
SubscribeLabel
Model
var BrokerIpAddress
var Variable1
var Message
Properties
var
Name BrokerIpAddress
Type Variable
BrokerIpAddress String test.mosquitto.org



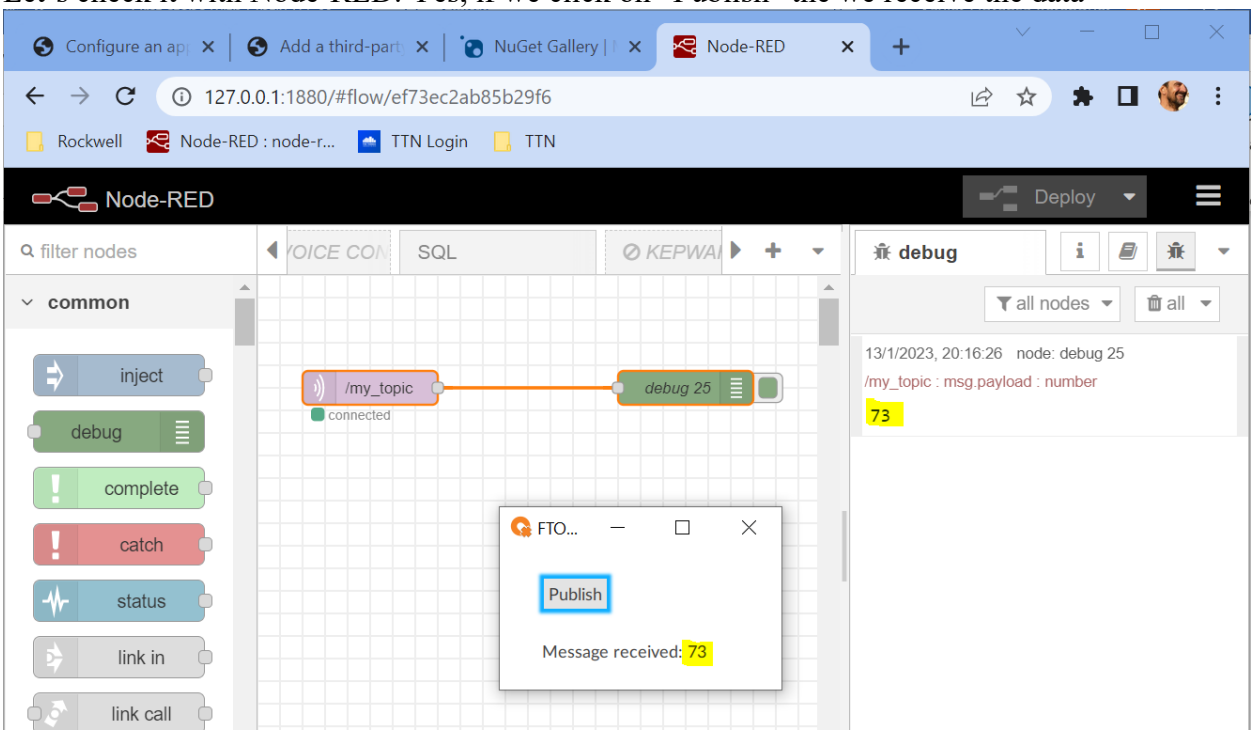


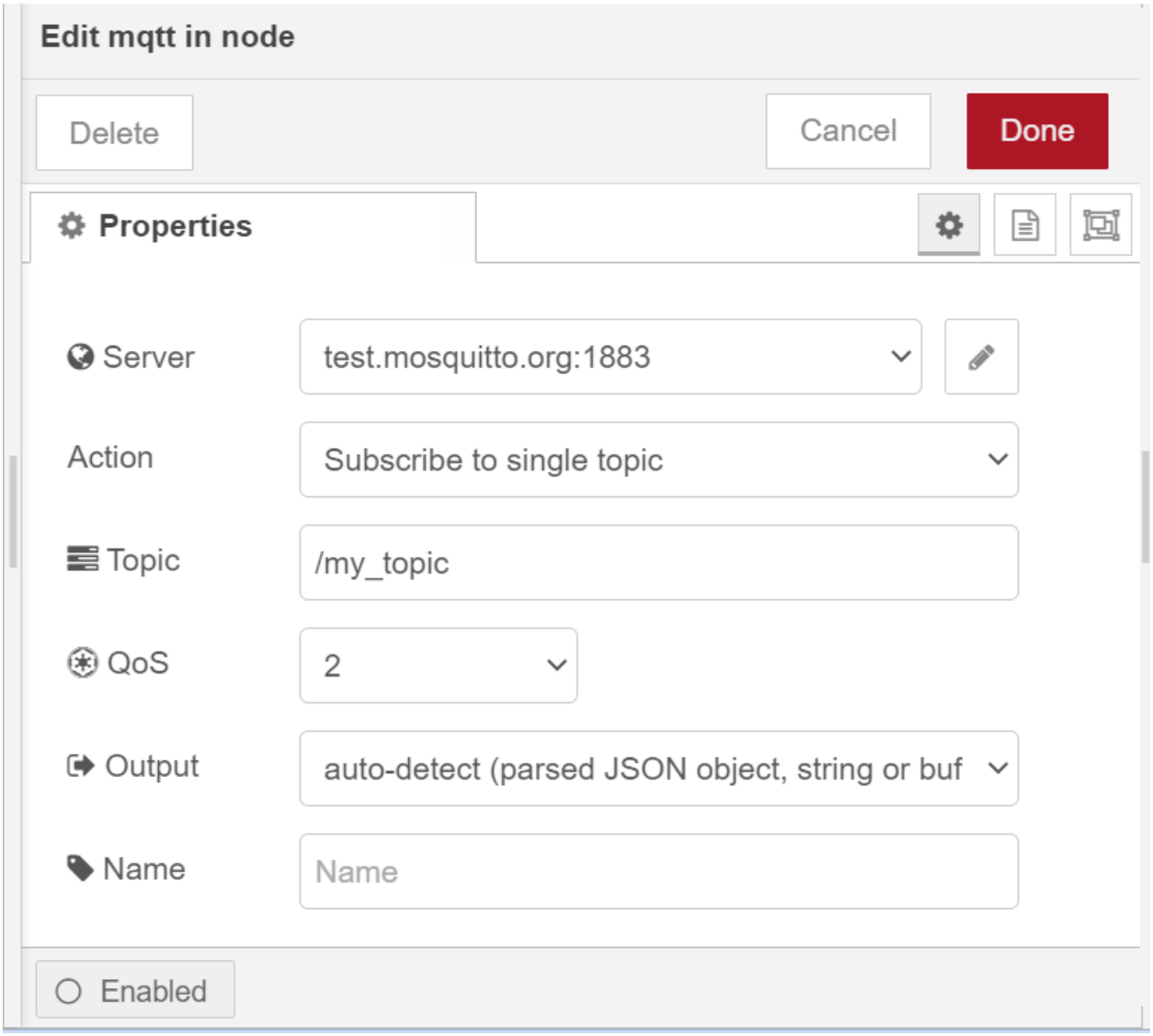
So we are publishing a random value to mosquito on the cloud

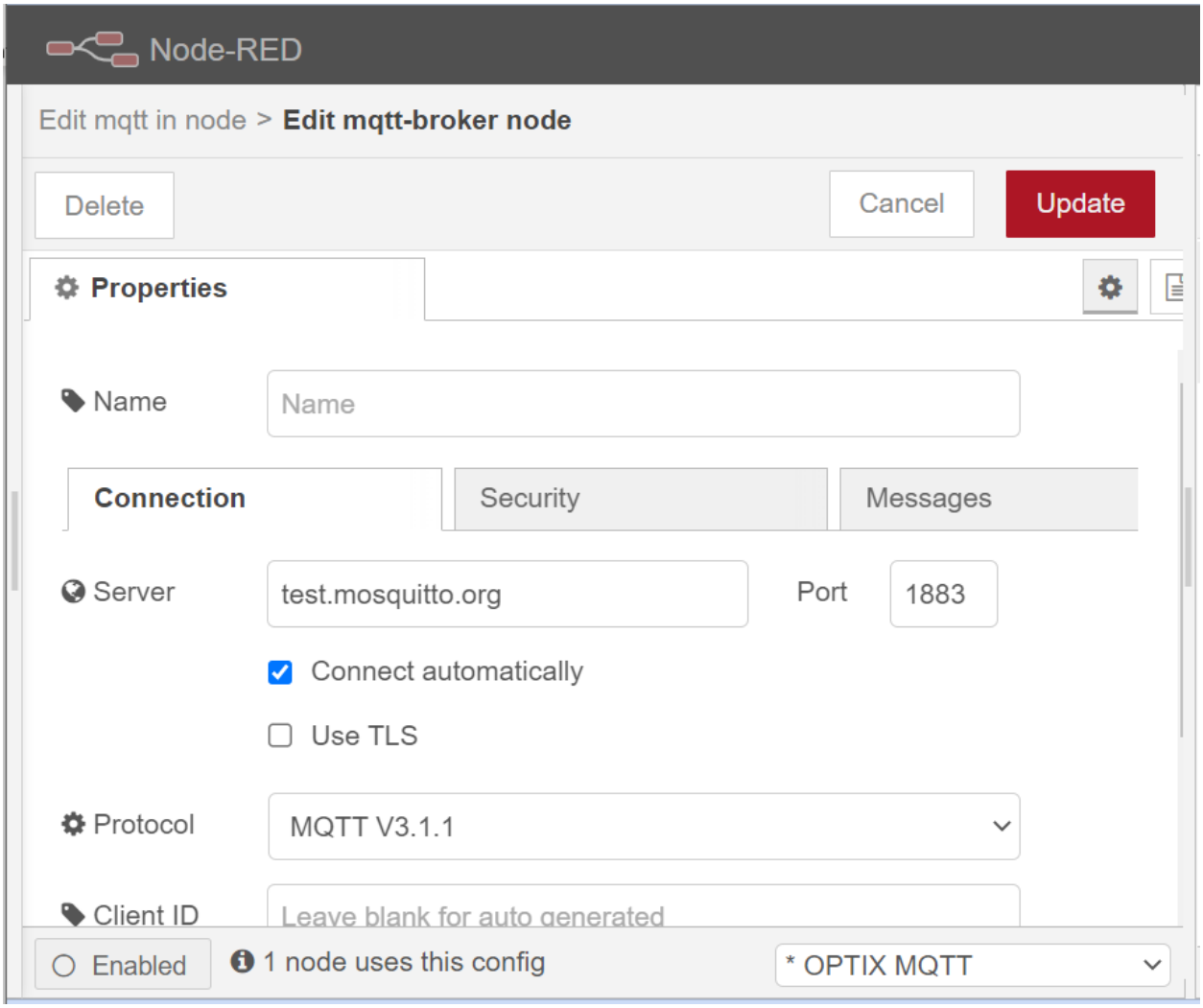
```
[ExportMethod]
public void PublishMessage()
{
    var variable1 = Project.Current.GetVariable("Model/Variable1");
    variable1.Value = new Random().Next(0, 101);

    // Publish a message
    ushort msgId = publishClient.Publish("/my_topic", // topic
        System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString()), // message body
        MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
        false); // retained
}
```

We are publishing to Topic /my_topic

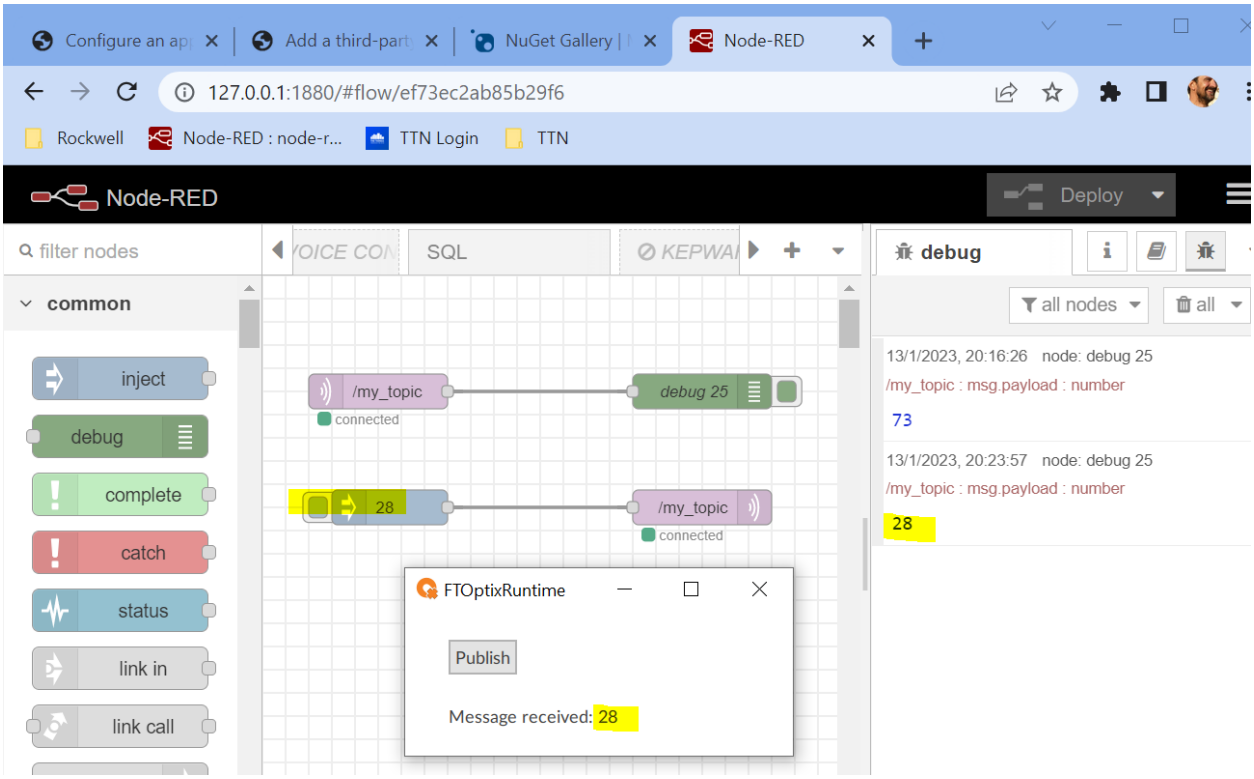
Let's check it with Node-RED. Yes, if we click on "Publish" the we receive the data

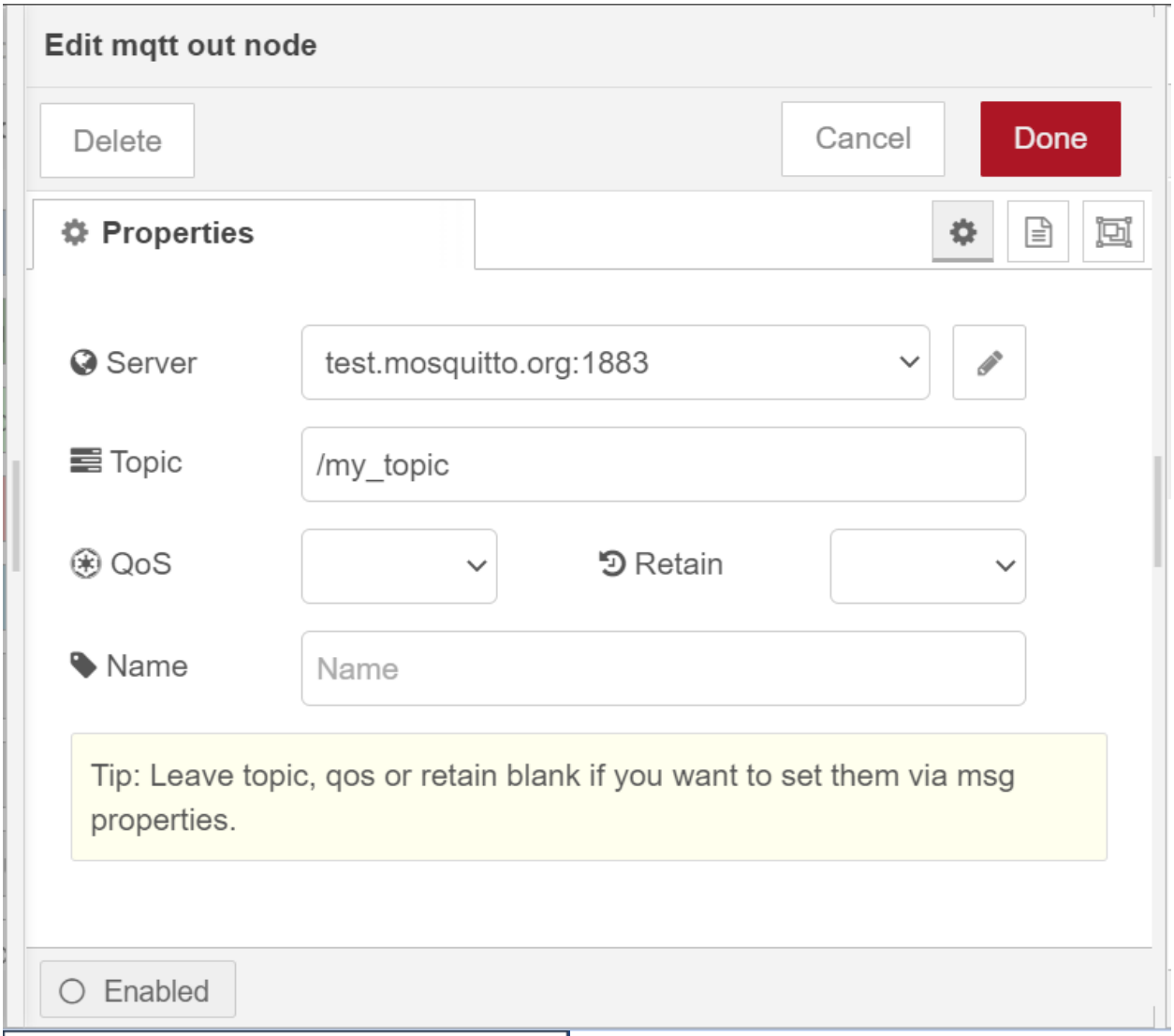


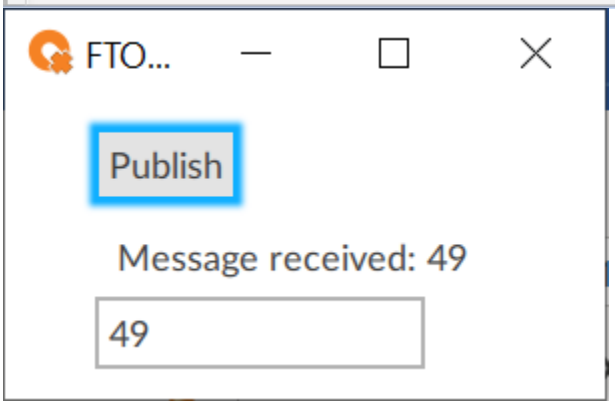

Edit mqtt in node
Delete
Cancel
Done
Properties
Server
test.mosquitto.org:1883
Action
Subscribe to single topic
Topic
/my_topic
QoS
2
Output
auto-detect (parsed JSON object, string or buf
Name
Name
Enabled



Let's test the subscription from Optix application
Yes, if we inject on Node-RED, the Optix Application gets the value


Configure an ap
Add a third-part
NuGet Gallery
Node-RED
127.0.0.1:1880/#flow/ef73ec2ab85b29f6
Rockwell
Node-RED : node-r...
TTN Login
TTN
Node-RED
Deploy
filter nodes
SQL
debug
common
all nodes
all
inject
debug
complete
catch
status
link in
link call
/my_topic
connected
debug 25
28
/my_topic
connected
13/1/2023, 20:16:26 node: debug 25
/my_topic : msg.payload : number
73
13/1/2023, 20:23:57 node: debug 25
/my_topic : msg.payload : number
28
FTOptixRuntime
Publish
Message received: 28



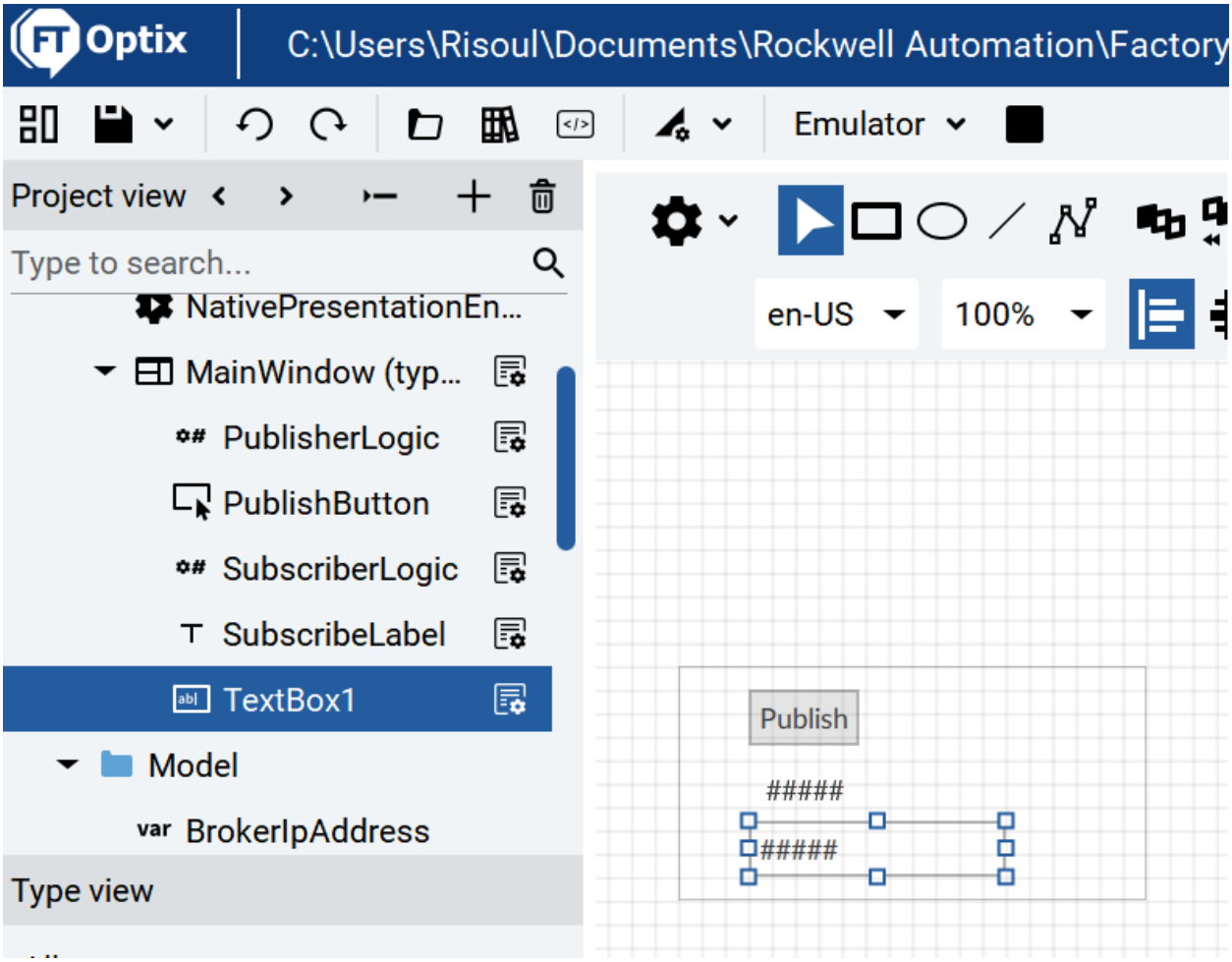


Next step is to use PLC data

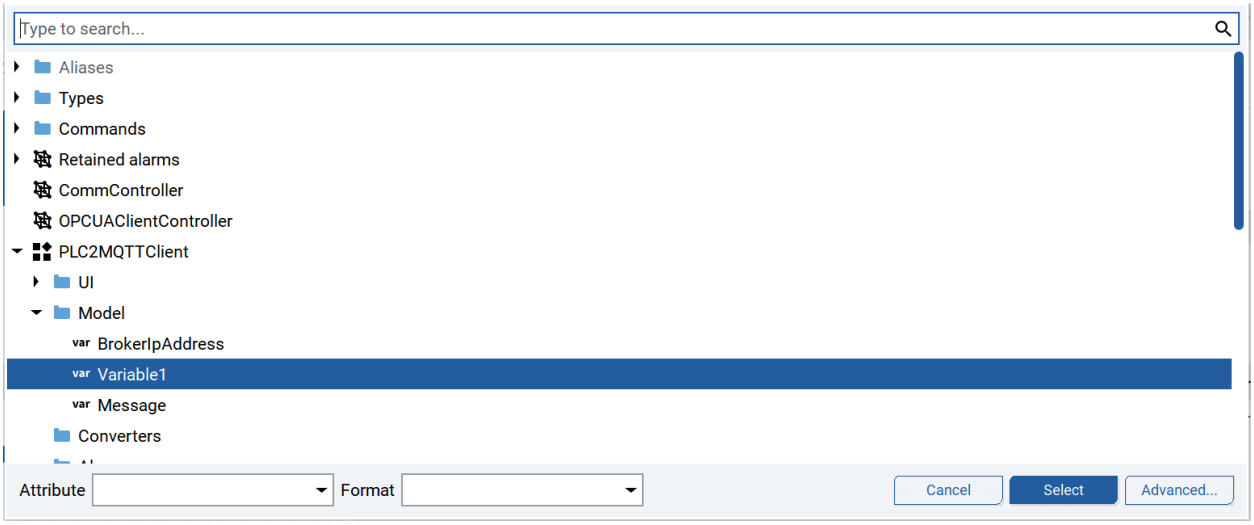
19. PLC data and MQTT

This is what we will do on this example

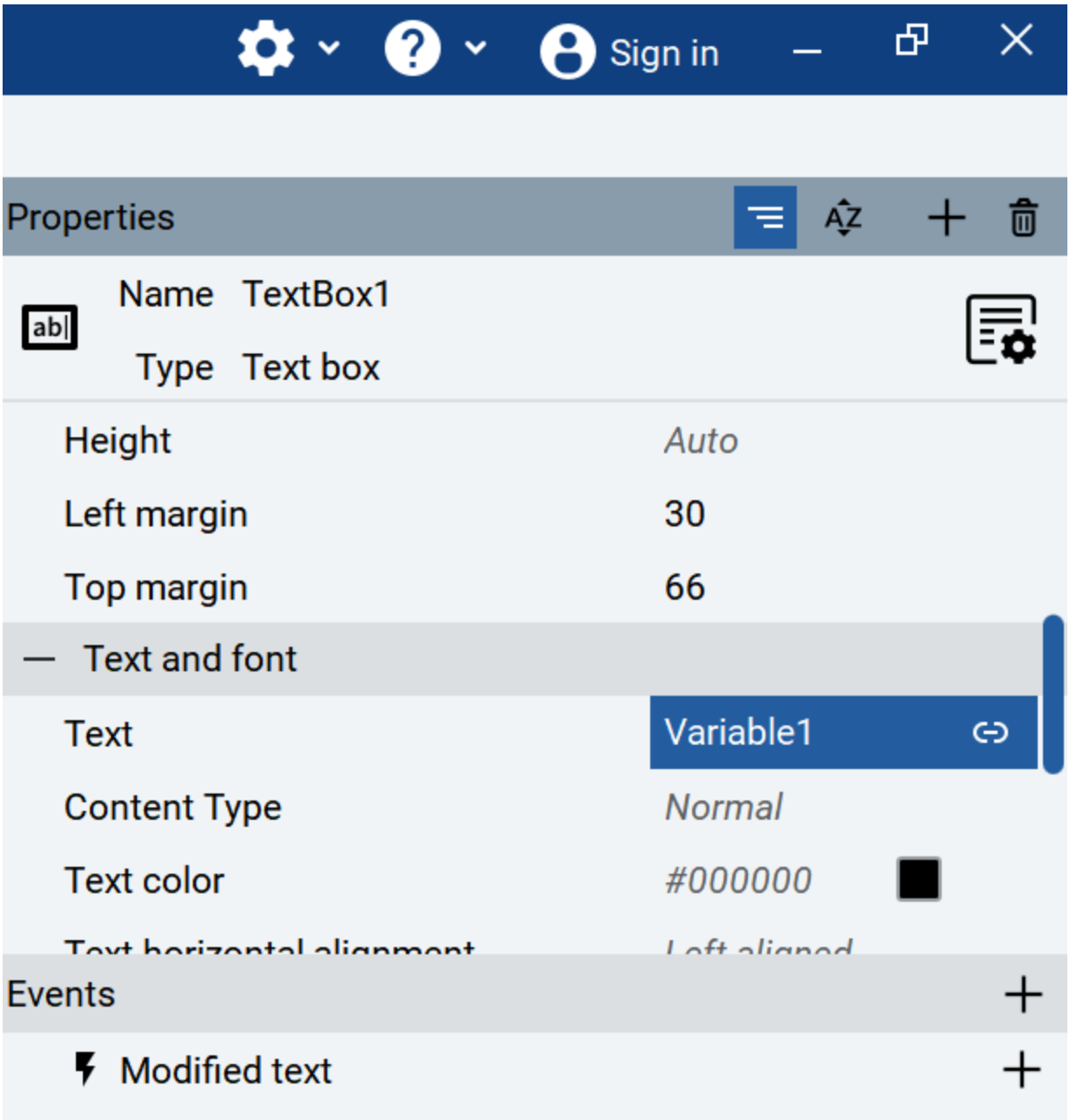
You an find the code here
https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git
We will create an EtherNet/IP driver on same MQTT sample project
Let's take a look on how to work in our project with the variables provided by MQTT
Reading the subscribed variable
Create a new Texbox



On properties point to Model.variable1

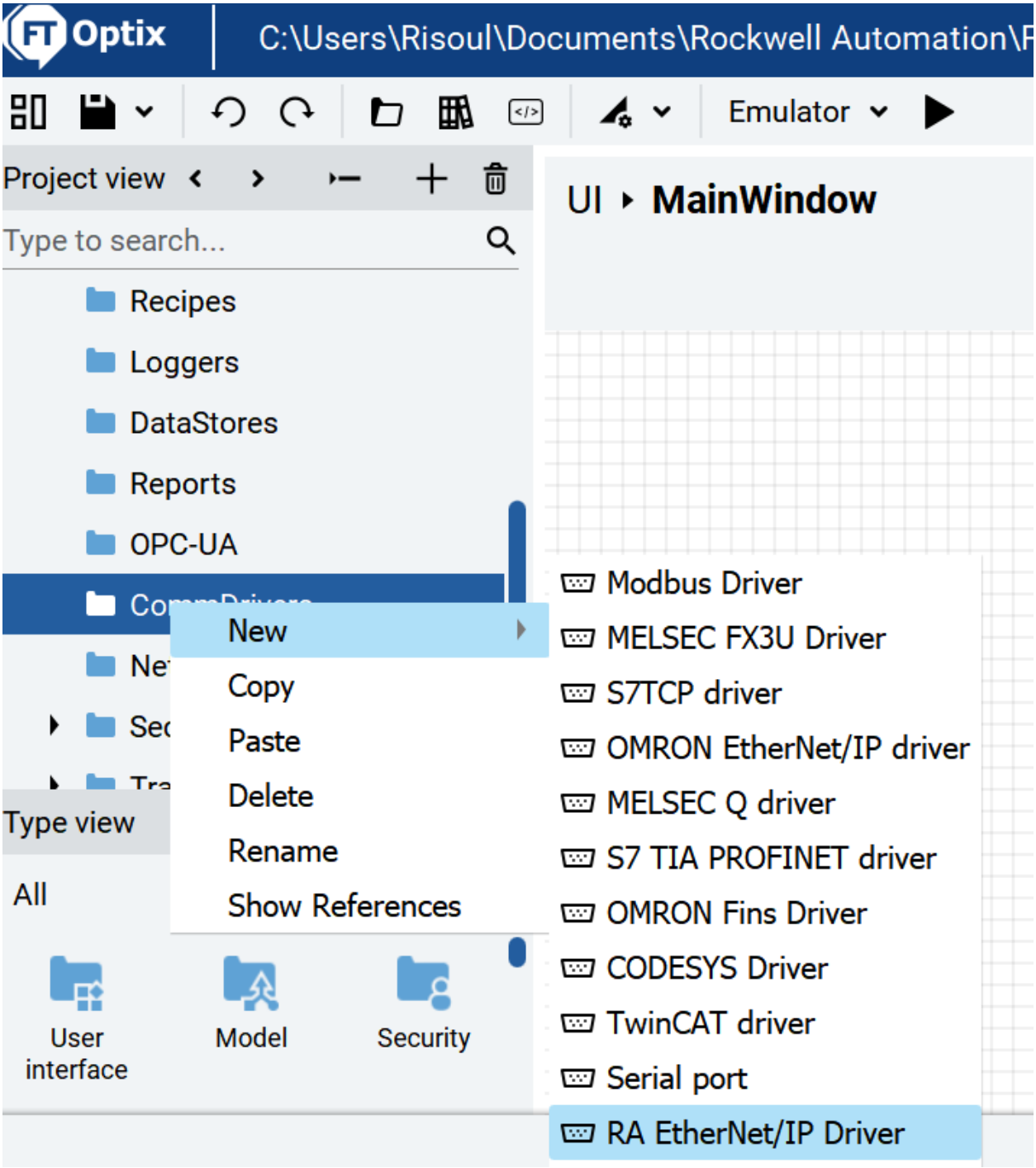

Type to search...
Aliases
Types
Commands
Retained alarms
CommController
OPCUAClientController
PLC2MQTTClient
UI
Model
var BrokerIpAddress
var Variable1
var Message
Converters
Attribute
Format
Cancel
Select
Advanced...

Like this

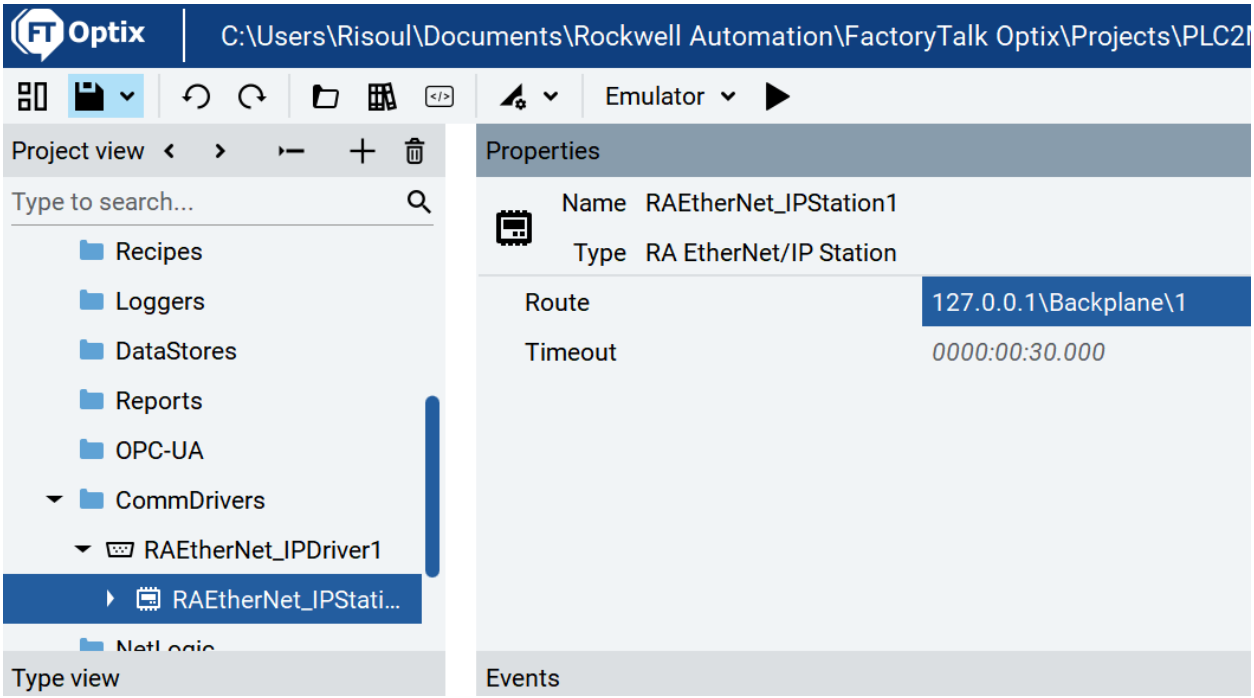


And on the other hand let's try to write to MQTT from PLC data
First let's get PLC data
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/getting-started/quick-start/configure-variables/import-tag-variables.html

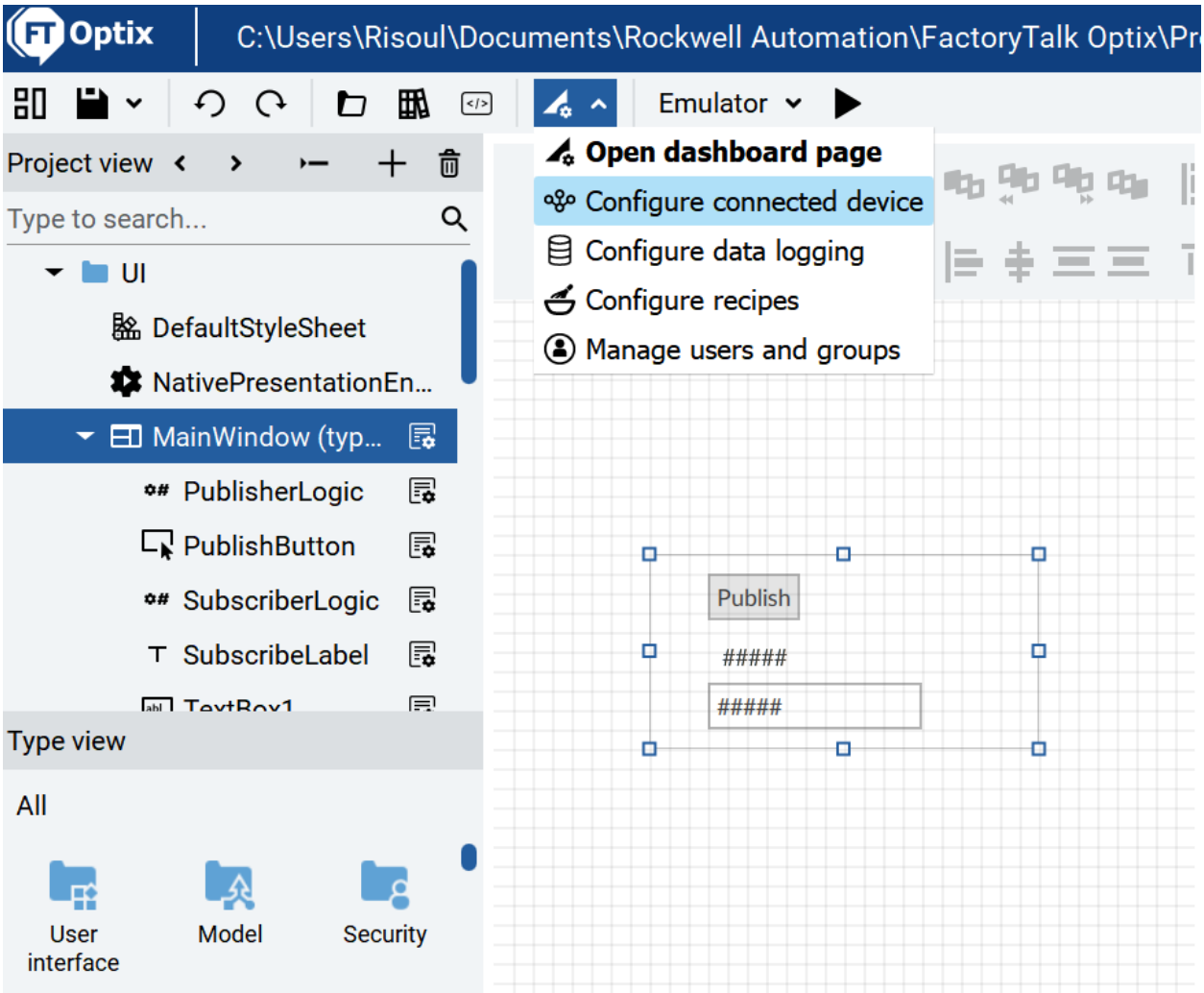
First of all create a communications driver

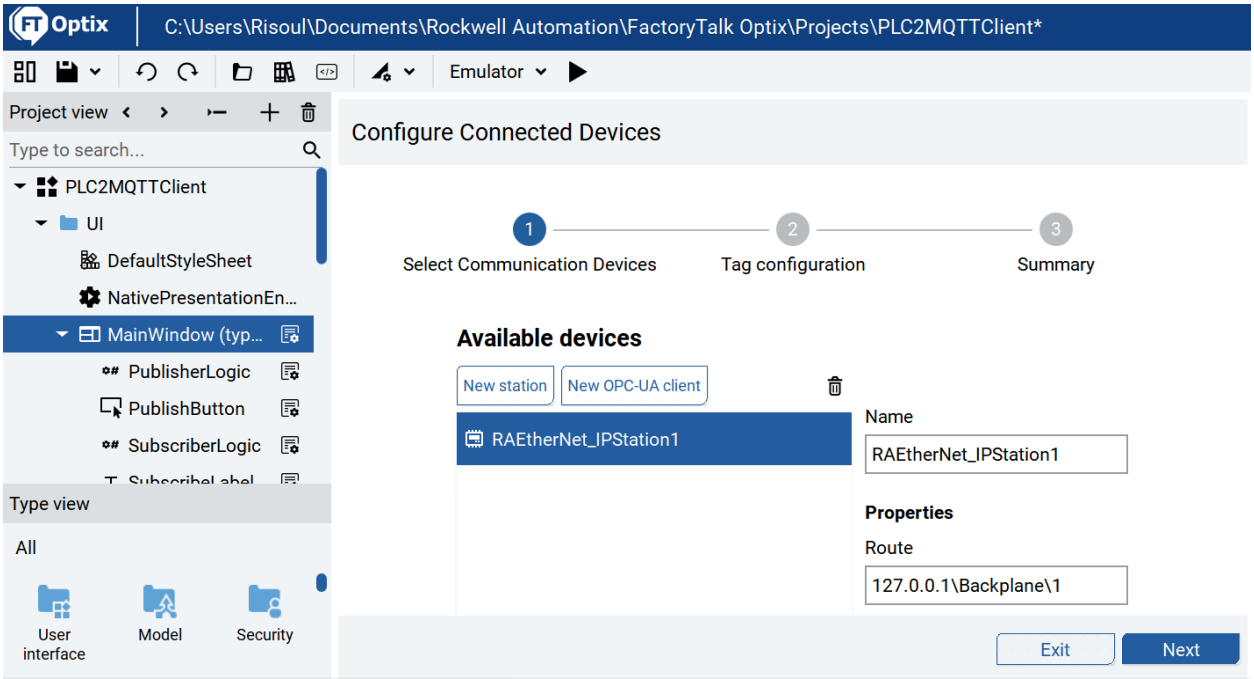


Add a new station

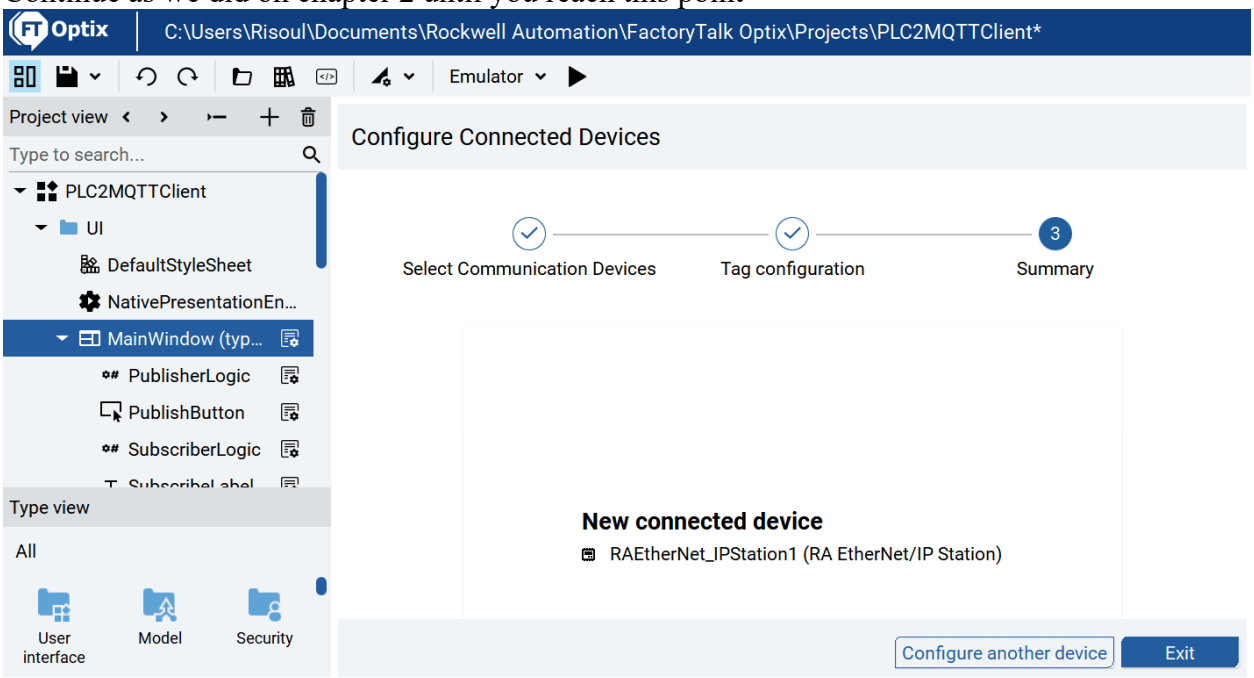


Go to configure connected device

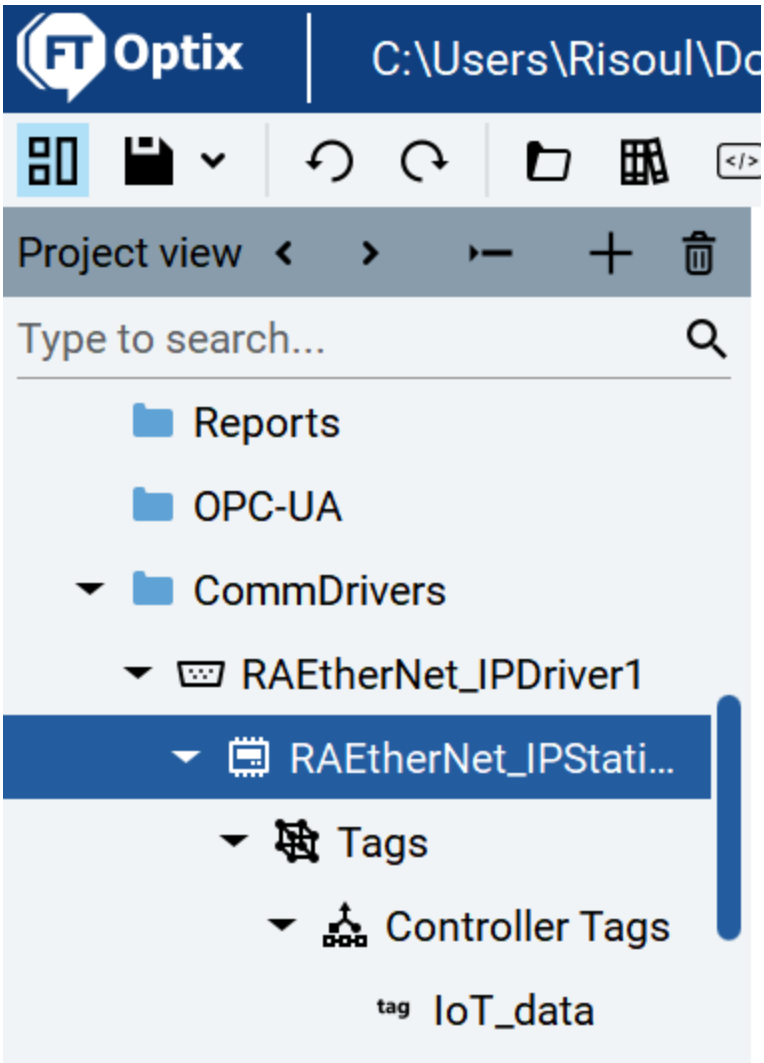

Optix
C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Pr
Emulator
Open dashboard page
Configure connected device
Configure data logging
Configure recipes
Manage users and groups
Project view
Type to search...
UI
DefaultStyleSheet
NativePresentationEn...
MainWindow (typ...
PublisherLogic
PublishButton
SubscriberLogic
SubscribeLabel
Type view
All
User interface
Model
Security
Publish
#####
#####



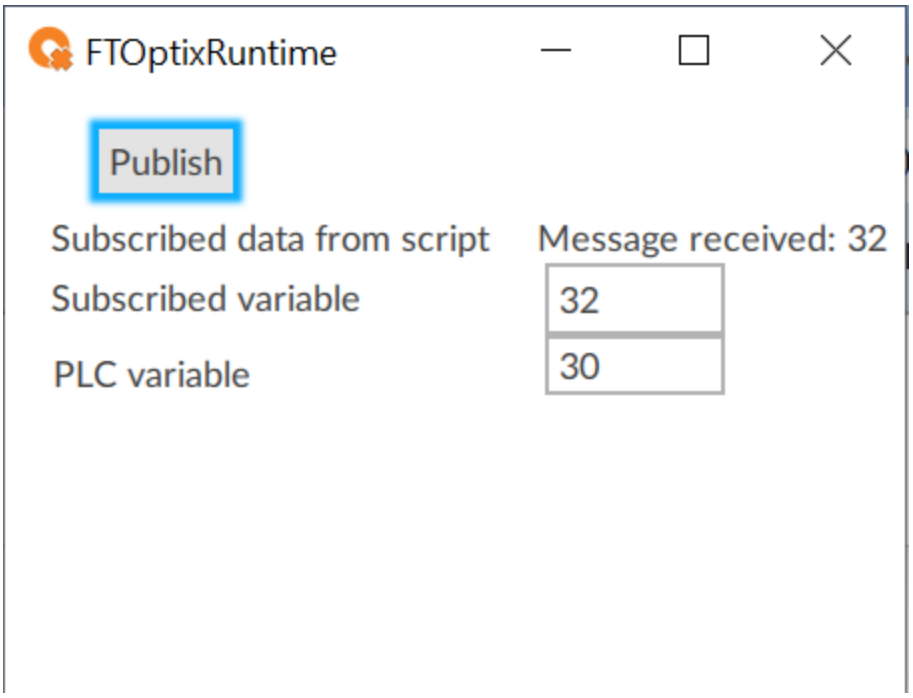
Continue as we did on chapter 2 until you reach this point



Verify that the Tag is there

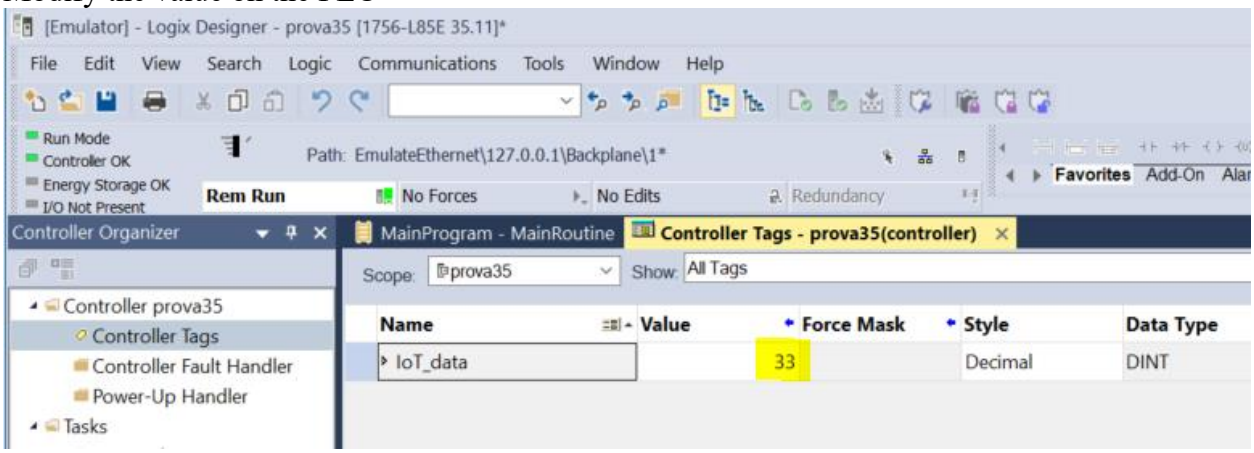


Now let's display the PLC data on a new text label

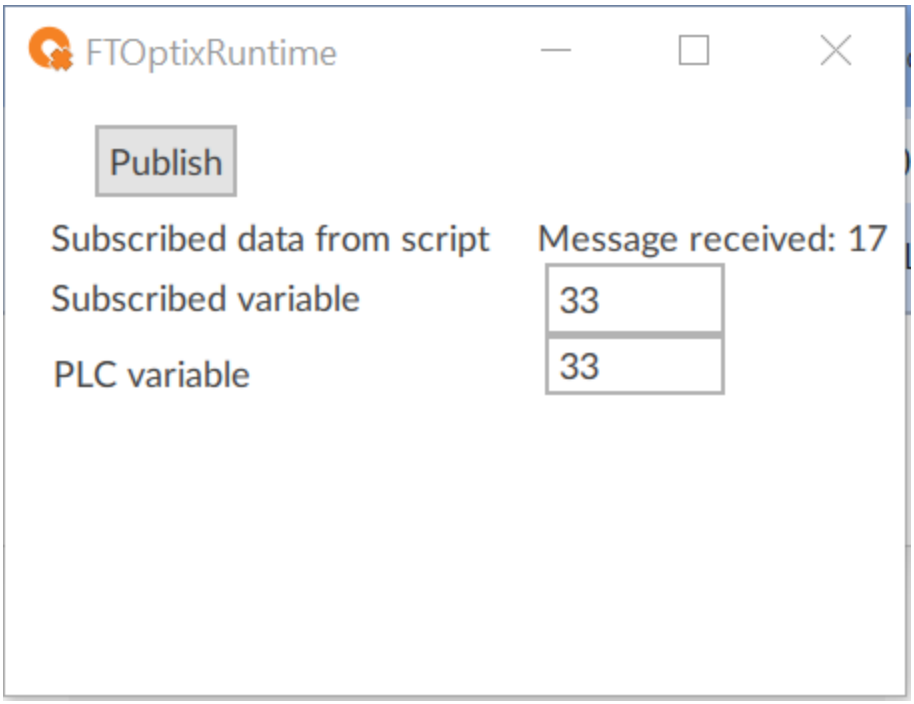


Now let's try to publish the data from PLC variable
We did a try that is publishing PLC data to Mosquitto, without writing any script, just with dynamic links:

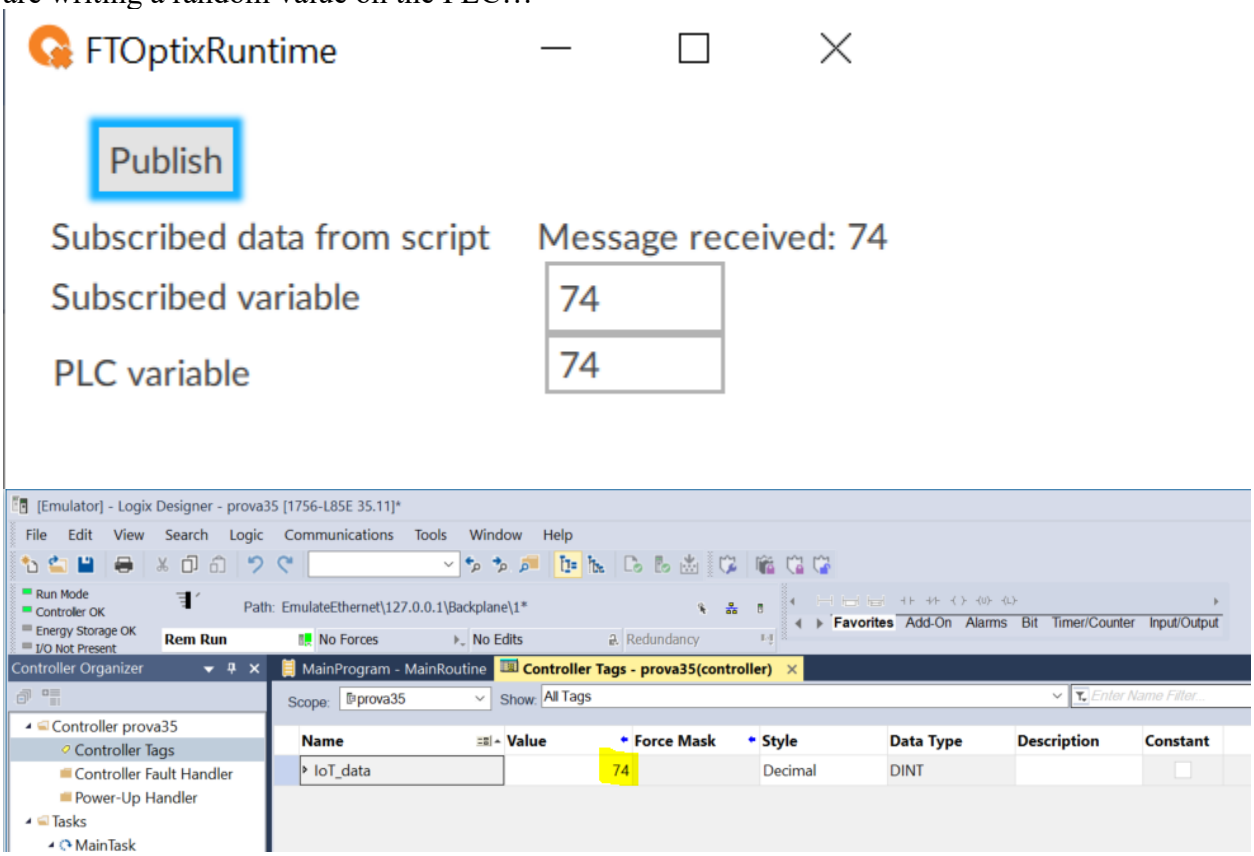
Modify the value on the PLC



Then look at the Optix application

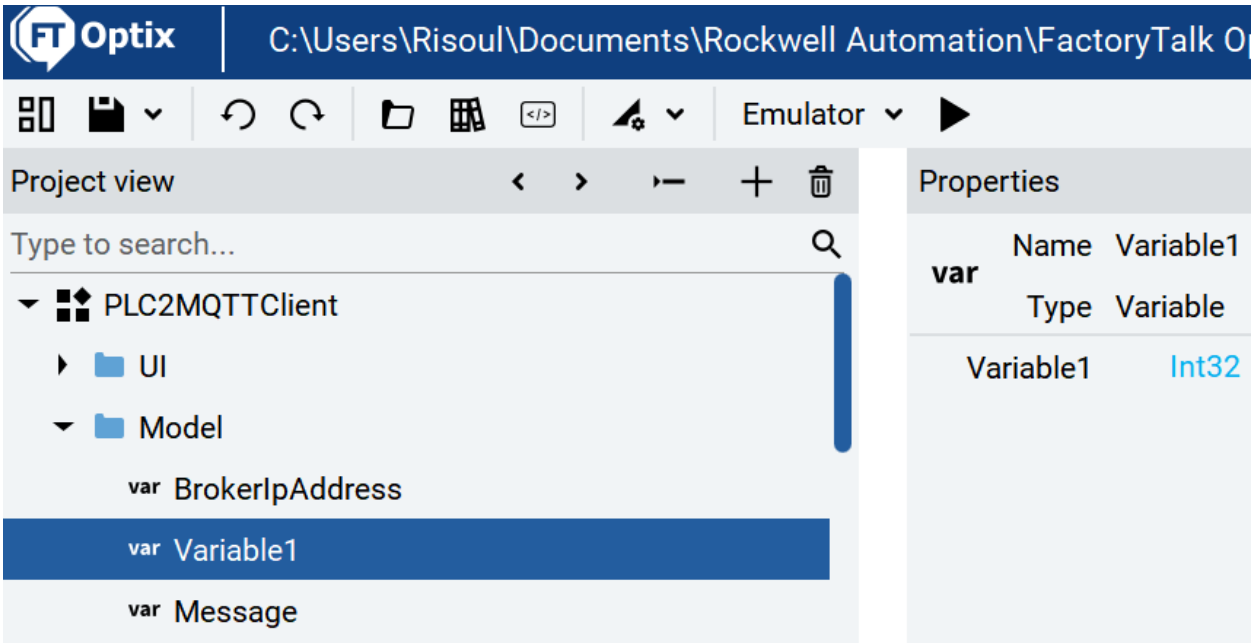


We wanted just to write on the PLC, but on the other hand if we click the button publish, then we are writing a random value on the PLC!!!

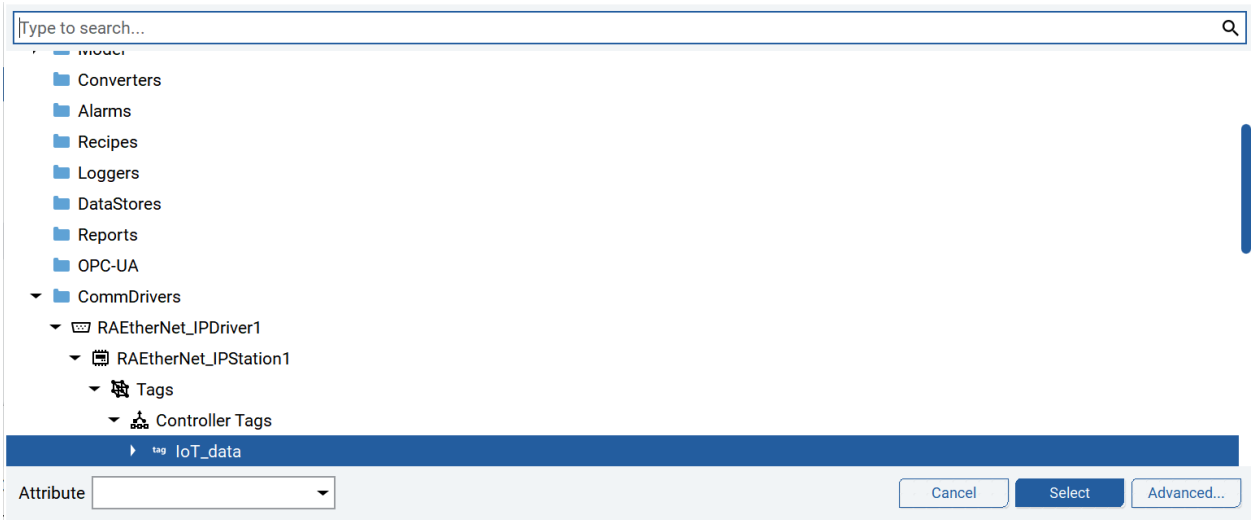


This is how to make this link

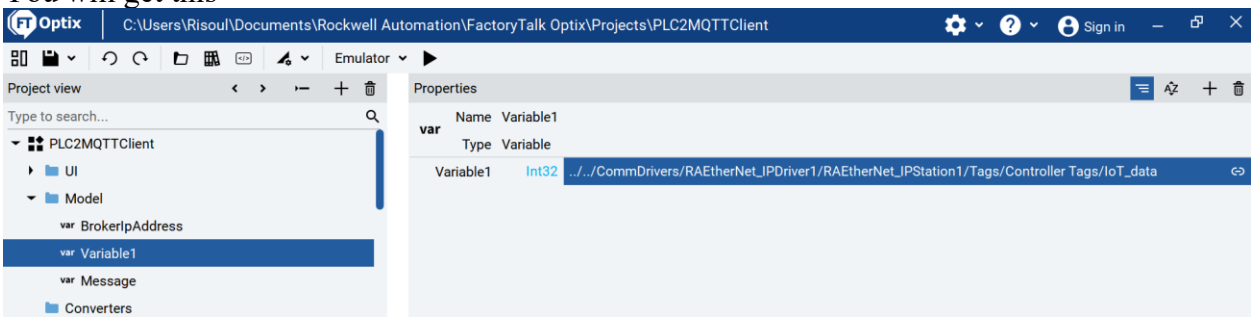
Just go to the Variable1 properties under model



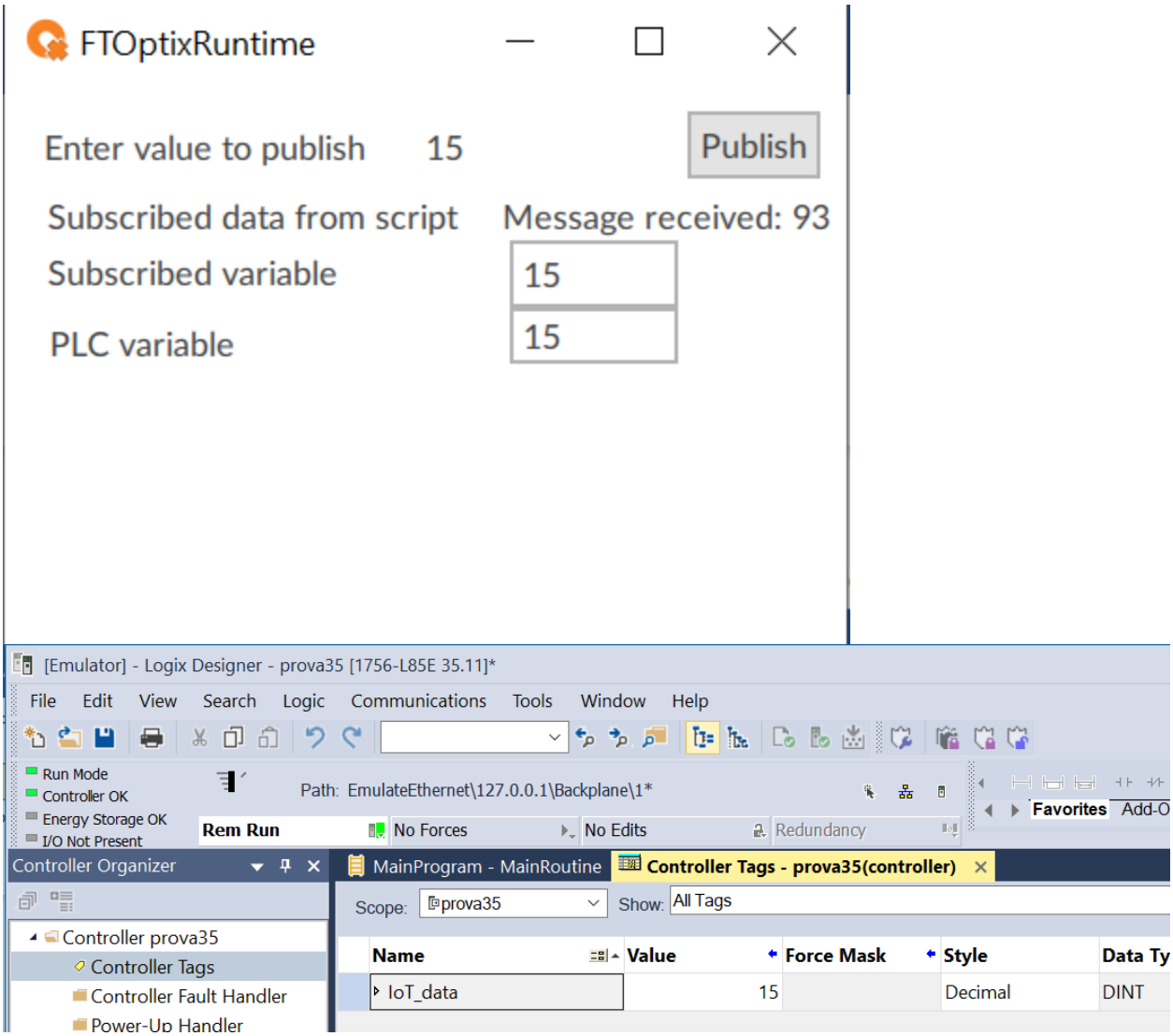
Then edit the link
Like this



You will get this

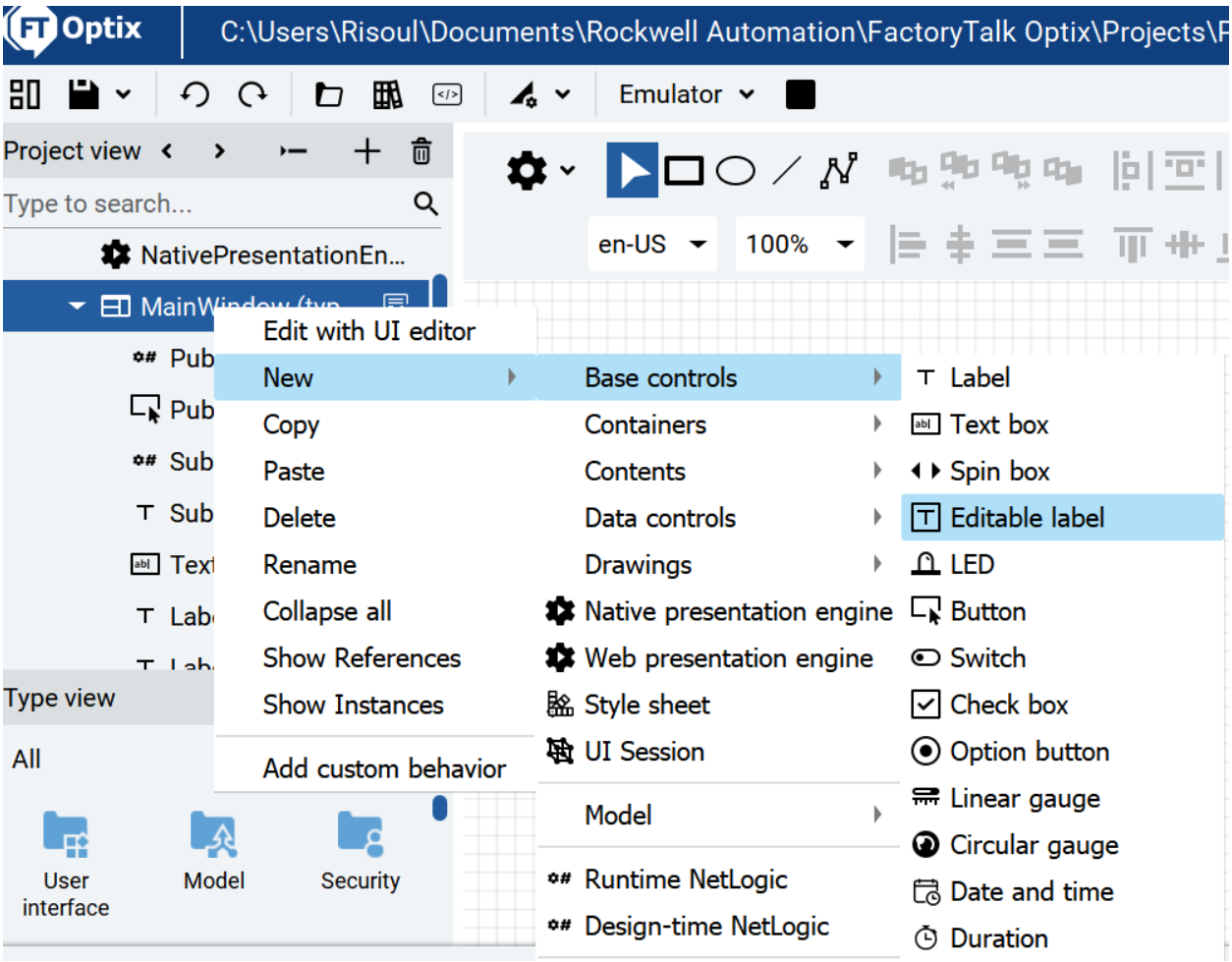


And that's all, you have the variables linked in both directions
But the problem when writing to the PLC is that the variable is random. Let's try to be our desired value entered by the operator on Optix.
Now we are able to write the desired value on the PLC as soon as we hit enter

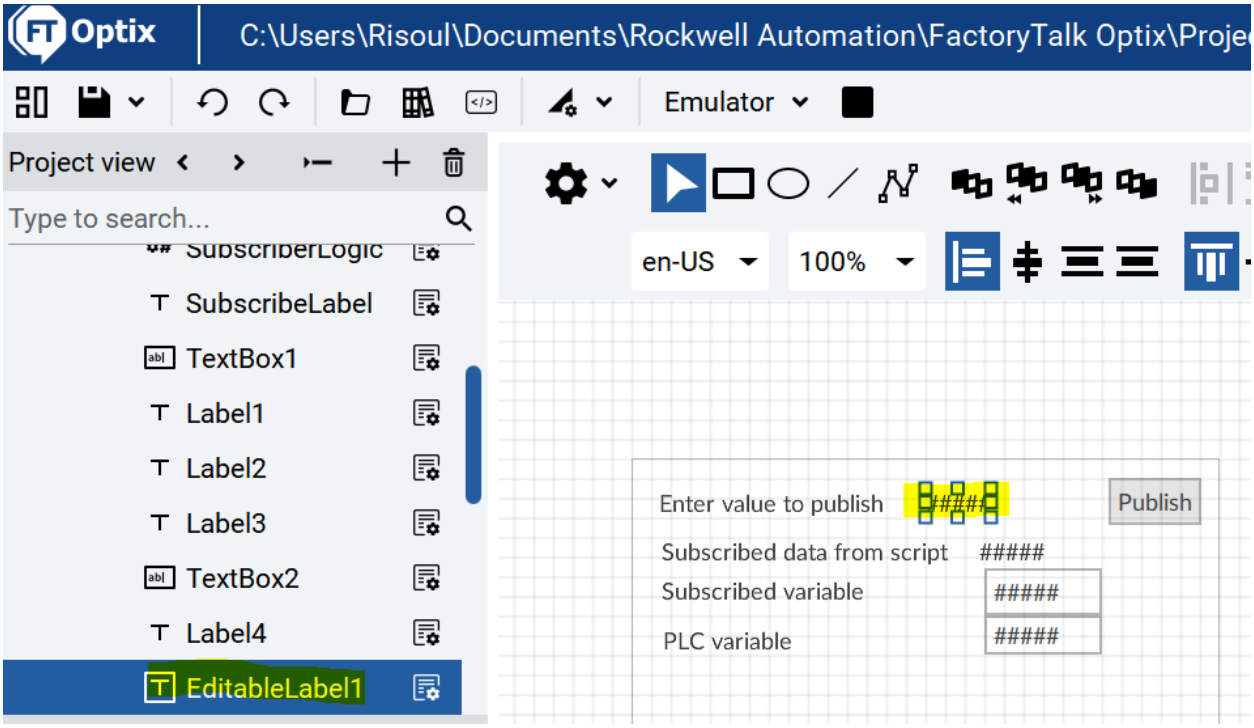


Let’s see how to do it

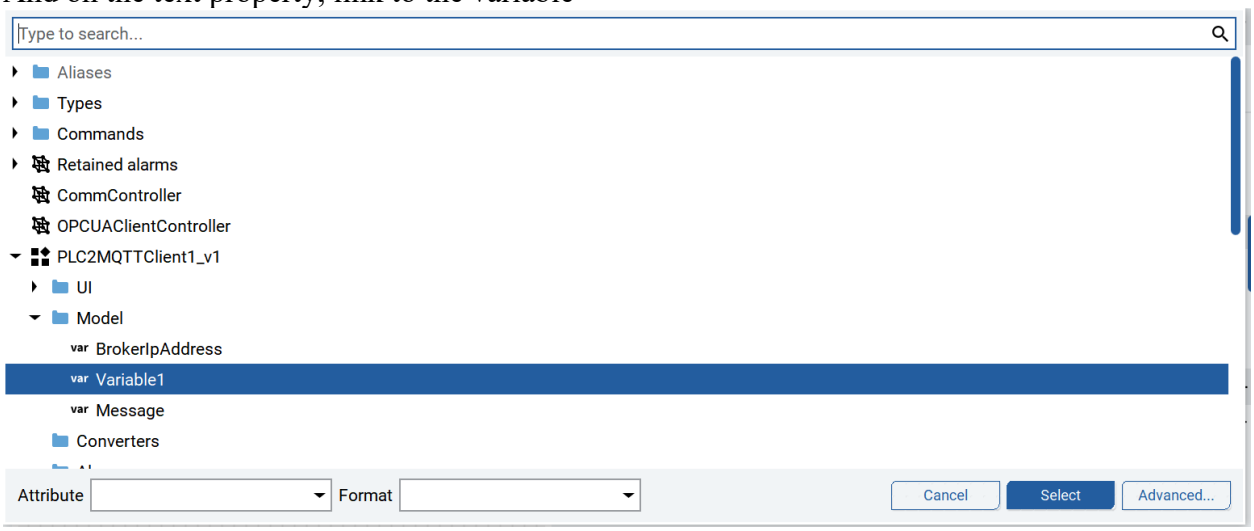
Let’s create an editable Label


Optix
C:\Users\Risoul\Documents\Rockwell Automation\FactoryTalk Optix\Projects\F
Emulator
Project view
Type to search...
NativePresentationEn...
en-US
100%
Edit with UI editor
New
Copy
Paste
Delete
Rename
Collapse all
Show References
Show Instances
Add custom behavior
Base controls
Containers
Contents
Data controls
Drawings
Native presentation engine
Web presentation engine
Style sheet
UI Session
Model
Runtime NetLogic
Design-time NetLogic
Label
Text box
Spin box
Editable label
LED
Button
Switch
Check box
Option button
Linear gauge
Circular gauge
Date and time
Duration
Type view
All
User interface
Model
Security

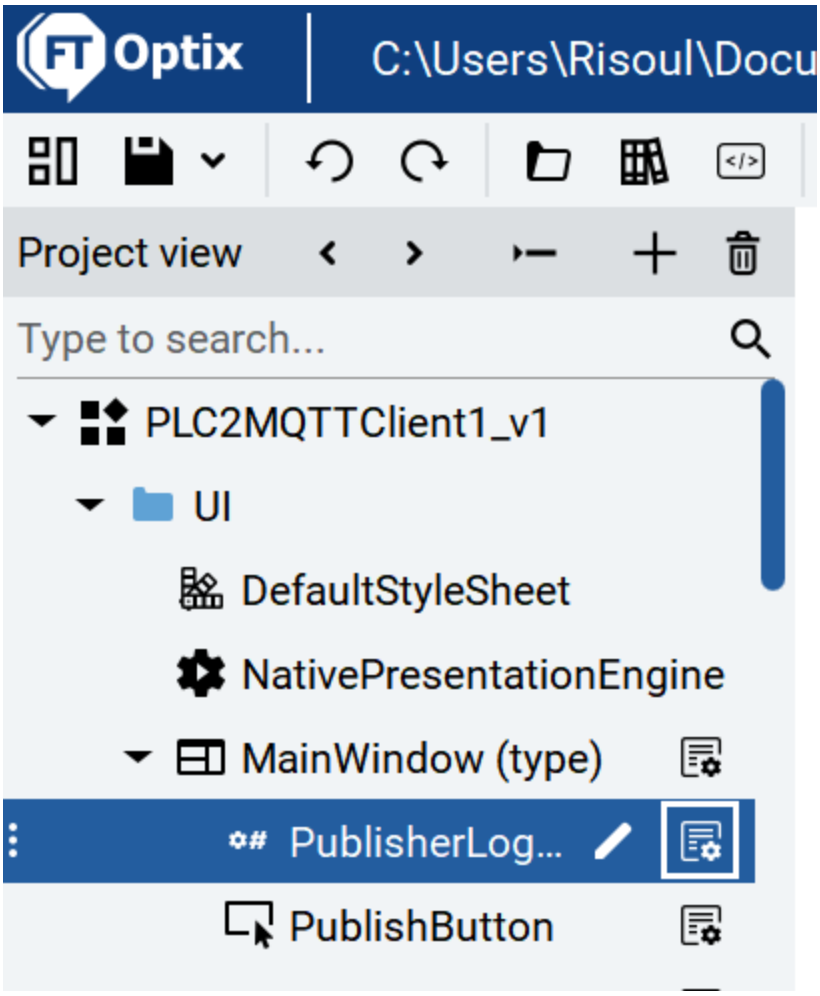
Like this



And on the text property, link to the variable



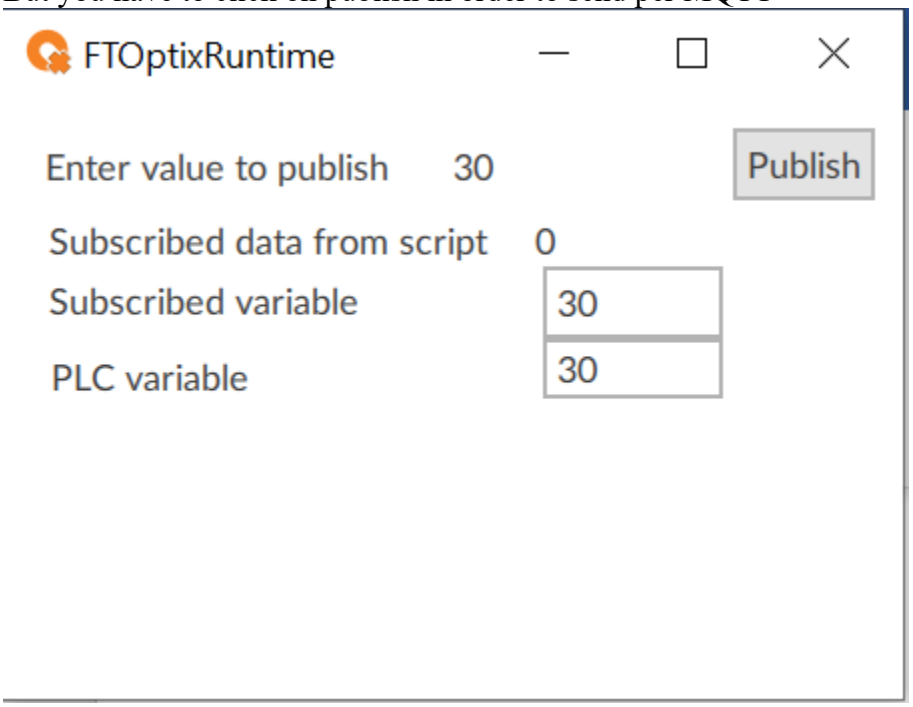
We are still writing a random variable with the publish button.
In order to cancel this we can try to comment this line on the code
Click on the code icon



And comment this line
Then save on Visual Studio Code

```
36      private void PublishClientMqttMsgPublished(object sender, MqttMsgPublishedEventArgs e)
37      {
38          Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
39      }
40
41      [ExportMethod]
42      public void PublishMessage()
43      {
44          var variable1 = Project.Current.GetVariable("Model/Variable1");
45
46          //variable1.Value = new Random().Next(0, 101);
47
48          // Publish a message
49          ushort msgId = publishClient.Publish("/my_topic", // topic
50              System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString()), // message body
51              MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
52              false); // retained
53      }
54
55      private MqttClient publishClient;
56  }
57
```

Now click on play and try
No more random publishing.
But you do not need to click the publish button to write on the PLC.
But you have to click on publish in order to send per MQTT



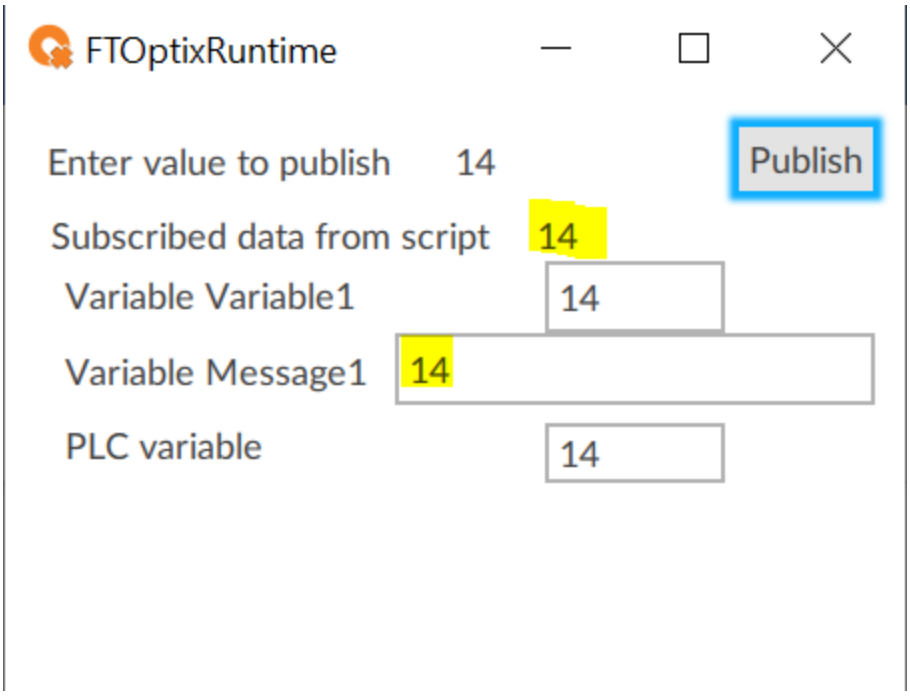
But we are not yet able to write on the PLC from an MQTT client like a mobile phone.
We have to do two steps
First of all, modify on the subscriber code this line

```
43        messageVariable.Value = "Message received: " + System.Text.Encoding.UTF8.GetString(e.Message);
```

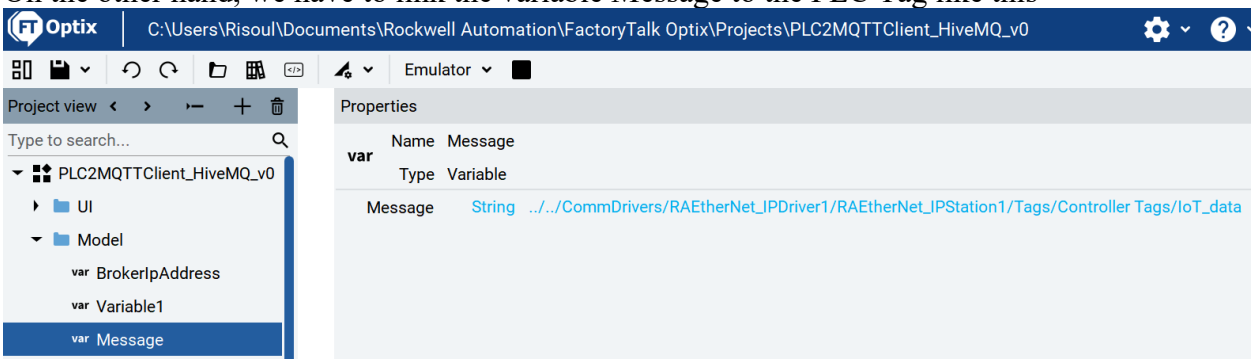
And leaving like this

```
44        messageVariable.Value = System.Text.Encoding.UTF8.GetString(e.Message);
```

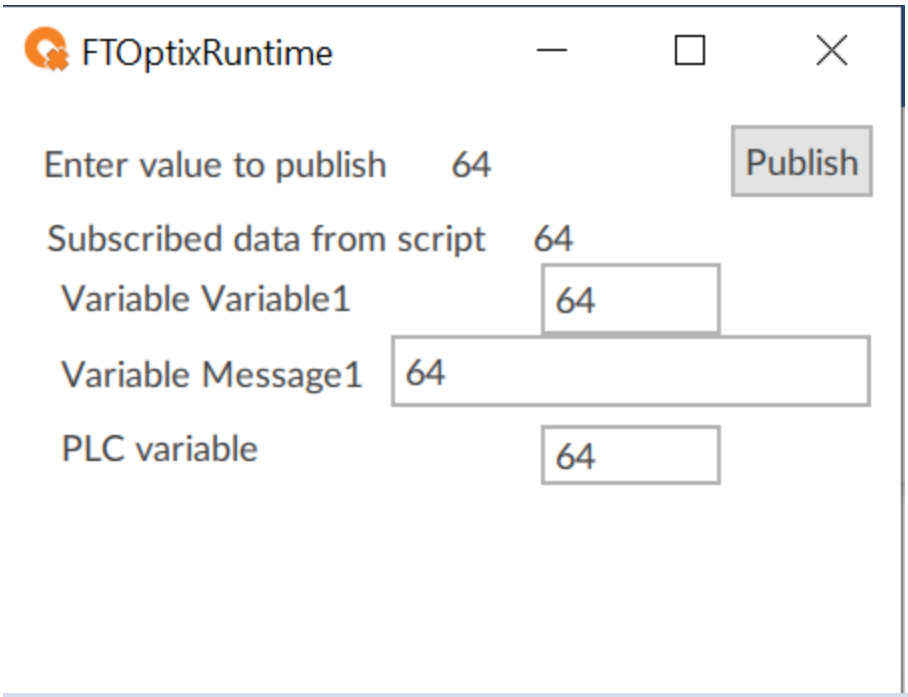
Then save on Visual Studio Code
Now we do not have the string like before

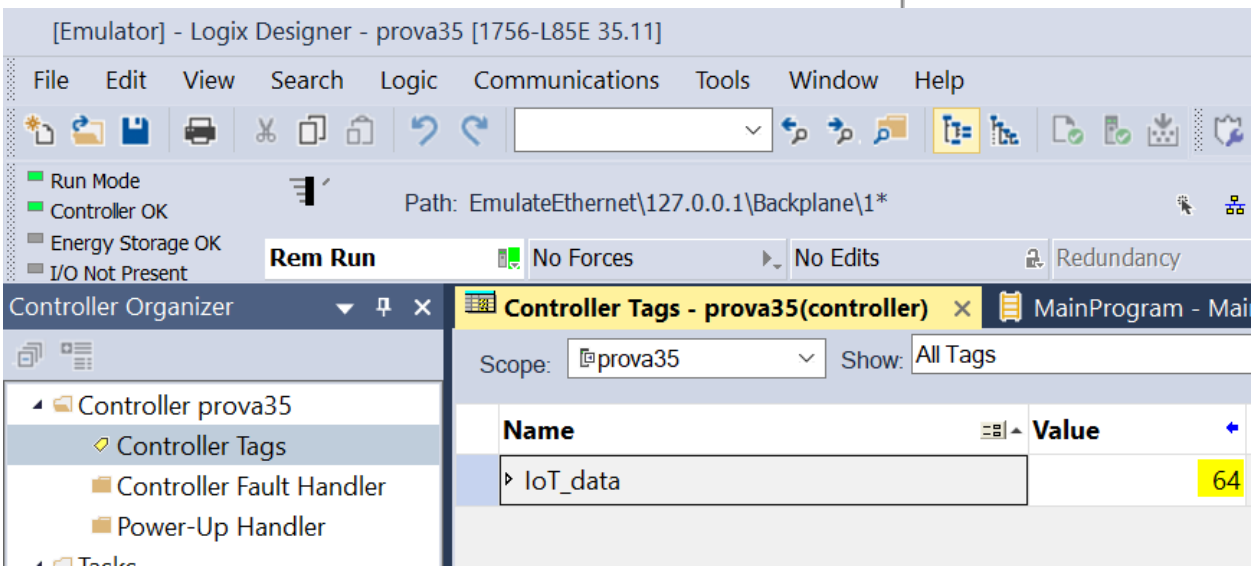


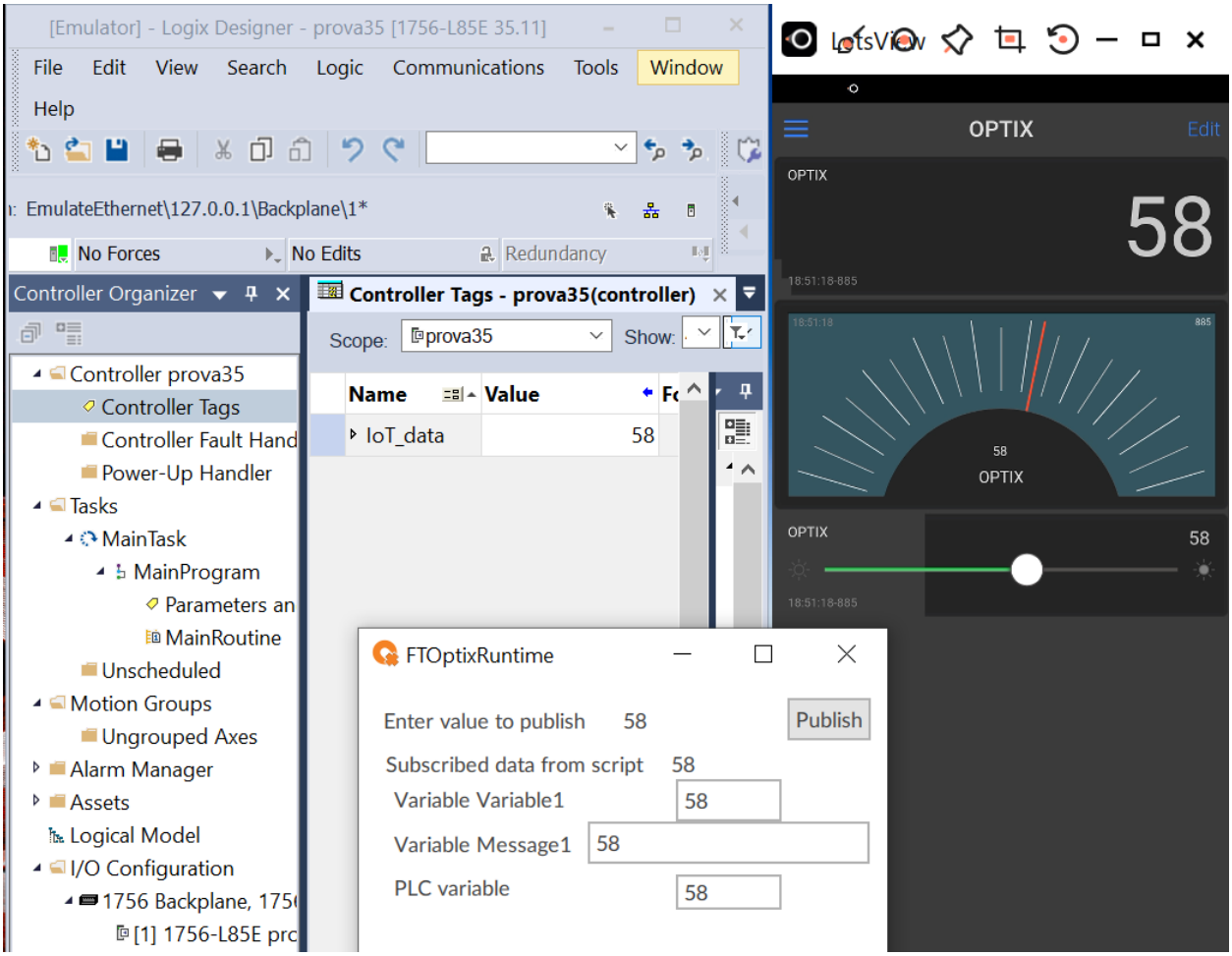
On the other hand, we have to link the variable Message to the PLC Tag like this



Now we are writing on the PLC from a MQTT client like a Mobile Phone


FTOptixRuntime
Enter value to publish 64
Publish
Subscribed data from script 64
Variable Variable1
64
Variable Message1
64
PLC variable
64


[Emulator] - Logix Designer - prova35 [1756-L85E 35.11]
File Edit View Search Logic Communications Tools Window Help
Run Mode
Controller OK
Energy Storage OK
I/O Not Present
Path: EmulateEthernet\127.0.0.1\Backplane\1*
Rem Run
No Forces
No Edits
Redundancy
Controller Organizer
Controller Tags - prova35(controller)
MainProgram - Mai
Scope: prova35
Show: All Tags
Controller prova35
Controller Tags
Controller Fault Handler
Power-Up Handler
Name
Value
IoT_data
64



As you can see on this video
https://youtu.be/u6EEDMmJJBU
You can find the code here
https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git

19.1. Asynchronous publish

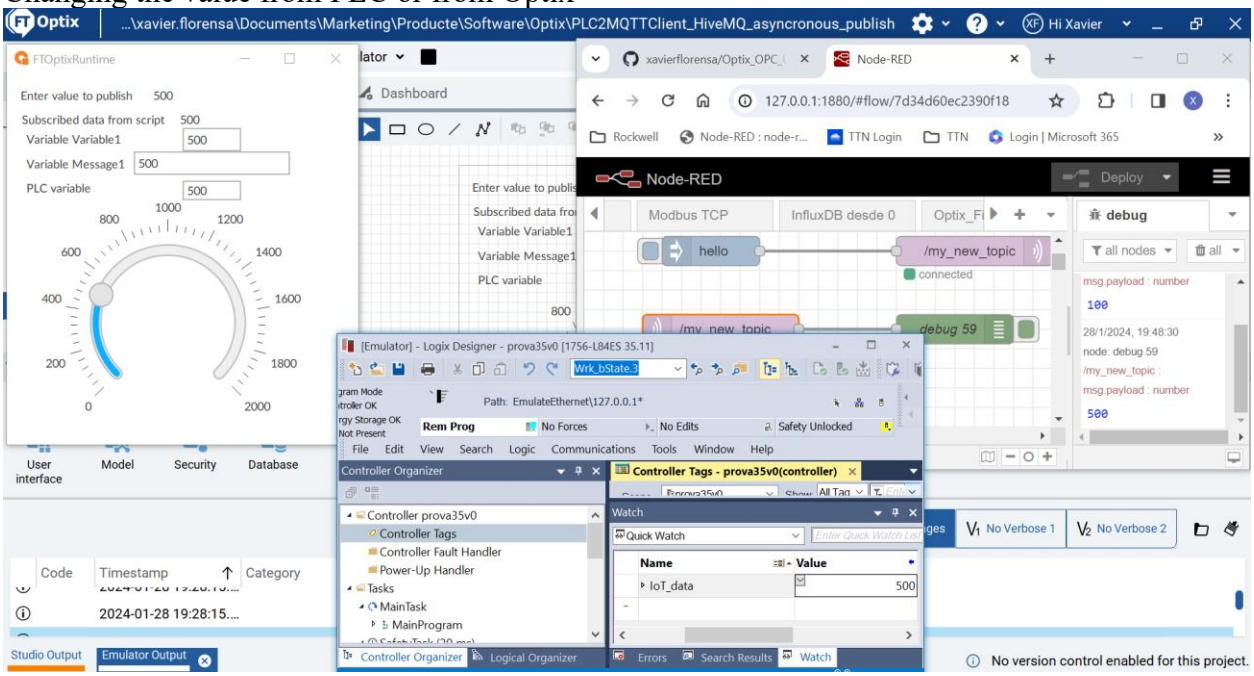
You may want to publish to MQTT broker each time you modify the PLC variable
You can see the result here
https://youtu.be/4C3tAHiFN1I
You can do it with a callback function that executes a publish when Variable1 is modified.
This way
On same project delete the publish button
Copy the publish method code and insert it on a publish function
Also change the topic name from /my_topic to /my_new_topic
So use this code on the publisher side

```
#region StandardUsing
using System;
using FTOptix.CoreBase;
using FTOptix.HMIProject;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
```

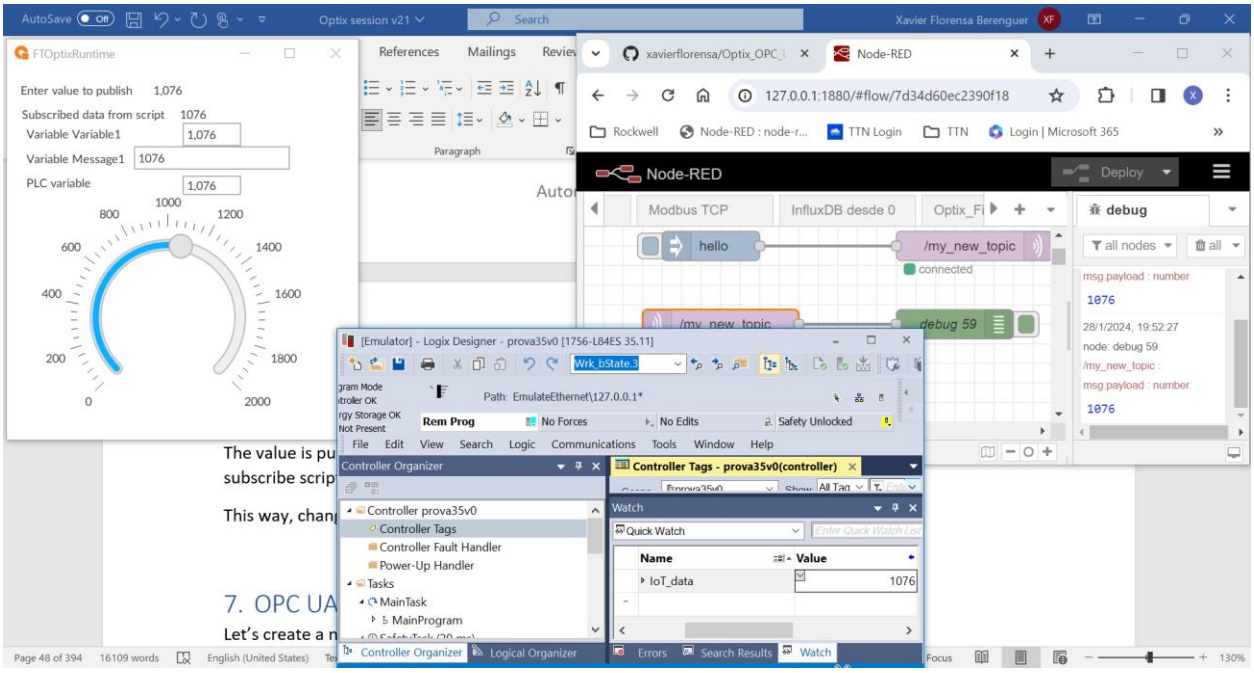
```
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;

public class PublisherLogic : BaseNetLogic
{
    private IUAVariable variable1;
    public override void Start()
    {
        var brokerIpAddressVariable =
Project.Current.GetVariable("Model/BrokerIpAddress");

        // Create a client connecting to the broker (default port is 1883)
        publishClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        publishClient.Connect("FTOptixPublishClient");
        // Assign a callback to be executed when a message is published to the
broker
        publishClient.MqttMsgPublished += PublishClientMqttMsgPublished;
        variable1 = Project.Current.GetVariable("Model/Variable1");
        //assing a callback function to be executed when the variable changes
        variable1.VariableChange += variable1_VariableChange;
    }

    public override void Stop()
    {
        publishClient.Disconnect();
        publishClient.MqttMsgPublished -= PublishClientMqttMsgPublished;
        variable1.VariableChange -= variable1_VariableChange;

    }

    private void PublishClientMqttMsgPublished(object sender,
MqttMsgPublishedEventArgs e)
    {
        Log.Info("Message " + e.MessageId + " - published = " + e.IsPublished);
    }

    private void variable1_VariableChange(object sender, VariableChangeEventArgs
e)
```

```
    {
        Log.Info("Variable has changed");
        var variable1 = Project.Current.GetVariable("Model/Variable1");
        //variable1.Value = new Random().Next(0, 101);

        // Publish a message
        ushort msgId = publishClient.Publish("/my_new_topic", // topic
            System.Text.Encoding.UTF8.GetBytes(((int)variable1.Value).ToString())
, // message body
            MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, // QoS level
            false); // retained
    }


    private MqttClient publishClient;
}
```

Test the application
Changing the value from PLC or from Optix



But what happens when you change the PLC value from your MQTT client as a mobile phone, The value is published twice, first time from mobile phone and second time from Optix, since Optix subscribe script will publish everything that is received thru MQTT
This way, change the value from mobile phone MQTT client app



This may be corrected from C#, but this will be done some other day…
You can find the code here
https://github.com/xavierflorensa/Optix_PLC_to_MQTT_asynchronous_publish
You can see the result here
https://youtu.be/4C3tAHiFN1I

20. Subscribing to an MQTT broker with user and password. TTN example

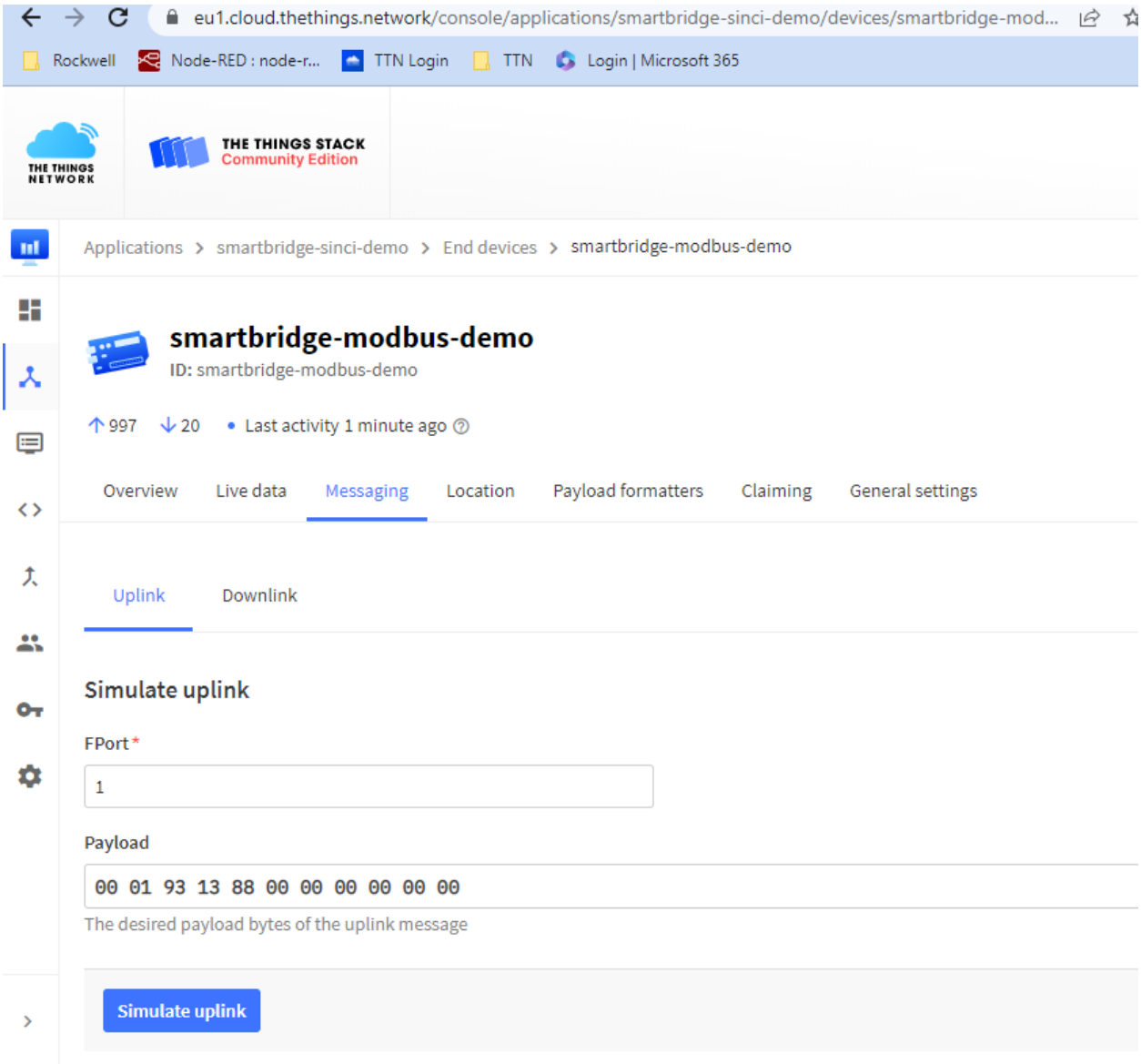
Just use the example from chapter 12
But change some lines on subcribelogic
From

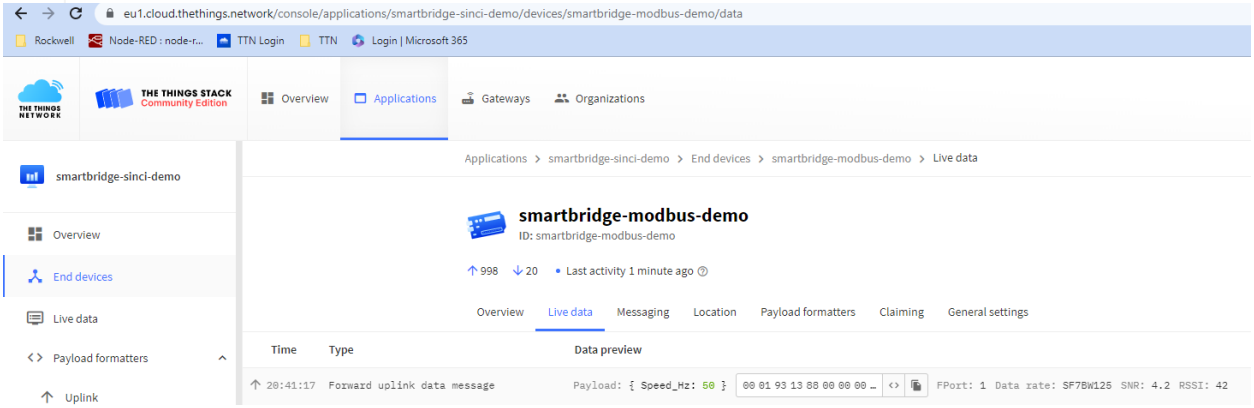
```
// Connect to the broker
        subscribeClient.Connect("FTOptixSubscribeClient");
```

to

```
// Connect to the broker
        subscribeClient.Connect("FTOptixSubscribeClient","username","your_passwor
d");
```

That's all
For example suscribing to TTN broker (eu1.cloud.thethings.network)
Then simulate an uplink


eu1.cloud.thethings.network/console/applications/smartbridge-sinci-demo/devices/smartbridge-mod...
Rockwell
Node-RED : node-r...
TTN Login
TTN
Login | Microsoft 365
THE THINGS NETWORK
THE THINGS STACK
Community Edition
Applications > smartbridge-sinci-demo > End devices > smartbridge-modbus-demo
smartbridge-modbus-demo
ID: smartbridge-modbus-demo
997
20
Last activity 1 minute ago
Overview
Live data
Messaging
Location
Payload formatters
Claiming
General settings
Uplink
Downlink
Simulate uplink
FPort *
1
Payload
00 01 93 13 88 00 00 00 00 00 00
The desired payload bytes of the uplink message
Simulate uplink


eu1.cloud.thethings.network/console/applications/smartbridge-sinci-demo/devices/smartbridge-modbus-demo/data
Overview
Applications
Gateways
Organizations
smartbridge-sinci-demo
Overview
End devices
Live data
Payload formatters
Uplink
Applications > smartbridge-sinci-demo > End devices > smartbridge-modbus-demo > Live data
smartbridge-modbus-demo
ID: smartbridge-modbus-demo
998
20
Last activity 1 minute ago
Overview
Live data
Messaging
Location
Payload formatters
Claiming
General settings
Time
Type
Data preview
20:41:17
Forward uplink data message
Payload: { Speed_Hz: 50 }
00 01 93 13 88 00 00 00 ...
FPort: 1 Data rate: SF7BW125 SNR: 4.2 RSSI: 42

This is how the subscriber logic looks like for this use case:

```
// Create a client connecting to the broker (default port is 1883)
        subscribeClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        subscribeClient.Connect("FTOptixSubscribeClient","smartbridge-sinci-
demo@ttn","this_is_my_TTN_tocken");
```

```
 // Subscribe to the "my_topic" topic with QoS 2
        ushort msgId = subscribeClient.Subscribe(new string[] { "v3/smartbridge-
sinci-demo@ttn/devices/smartbridge-modbus-demo/up" }, // topic
```

```
 public override void Stop()
    {
        subscribeClient.Unsubscribe(new string[] { "v3/smartbridge-sinci-
demo@ttn/devices/smartbridge-modbus-demo/up" });
        subscribeClient.Disconnect();
    }
```

```
 private void SubscribeClientMqttMsgPublishReceived(object sender,
MqttMsgPublishEventArgs e)
    {
        //messageVariable.Value = "Message received: " +
System.Text.Encoding.UTF8.GetString(e.Message);
        messageVariable.Value = "Message received: " +
System.Text.Encoding.Default.GetString(e.Message);

    }
```

But what if we want to get only the number after “Speed_Hz:” then we use the substring to cut and get only the number

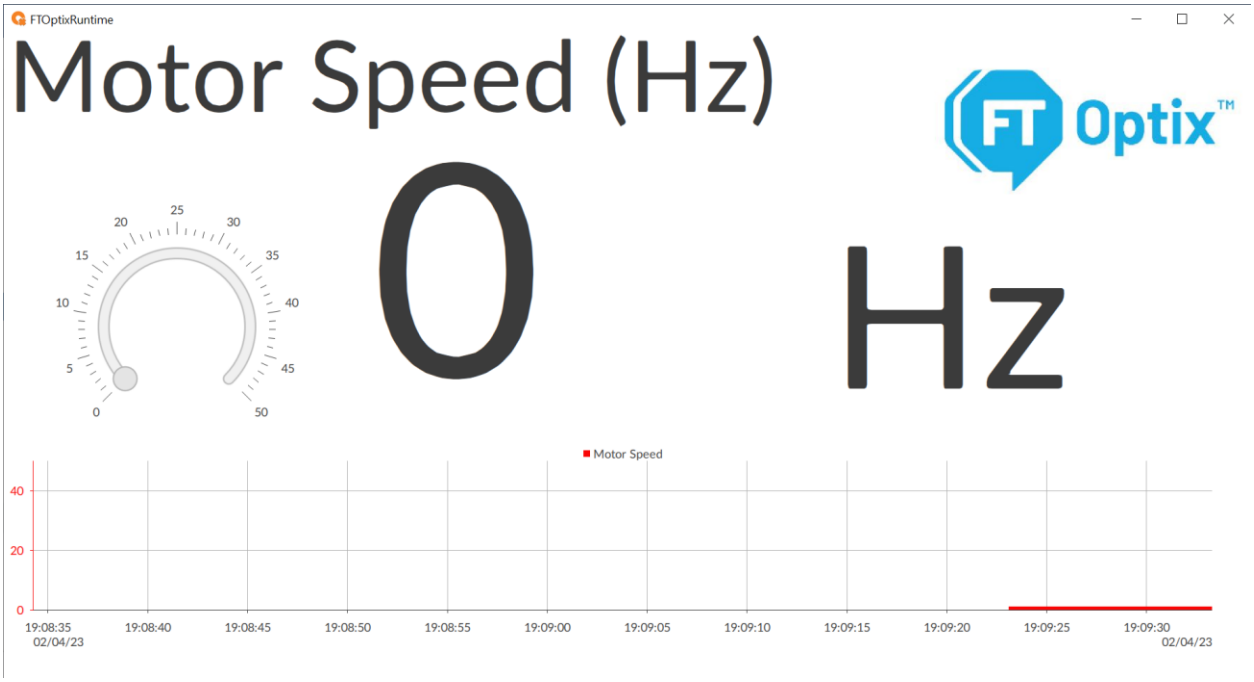
This is how the subscriber code looks like

```
#region StandardUsing
using System;
```

```
using FTOptix.CoreBase;
using FTOptix.HMIProject;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.OPCUAServer;
#endregion
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;

public class SubscriberLogic : BaseNetLogic
{
    public override void Start()
    {
        var brokerIpAddressVariable =
Project.Current.GetVariable("Model/BrokerIpAddress");

        // Create a client connecting to the broker (default port is 1883)
        subscribeClient = new MqttClient(brokerIpAddressVariable.Value);
        // Connect to the broker
        subscribeClient.Connect("FTOptixSubscribeClient","smartbridge-sinci-
demo@ttn","mytocken");
        // Assign a callback to be executed when a message is received from the
broker
        subscribeClient.MqttMsgPublishReceived +=
SubscribeClientMqttMsgPublishReceived;
        // Subscribe to the "my_topic" topic with QoS 2
        ushort msgId = subscribeClient.Subscribe(new string[] { "v3/smartbridge-
sinci-demo@ttn/devices/smartbridge-modbus-demo/up" }, // topic
            //new byte[] { MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE }); // QoS level
            new byte[] { MqttMsgBase.QOS_LEVEL_AT_MOST_ONCE }); // QoS level
        messageVariable = Project.Current.GetVariable("Model/Message");
    }

    public override void Stop()
    {
        subscribeClient.Unsubscribe(new string[] { "v3/smartbridge-sinci-
demo@ttn/devices/smartbridge-modbus-demo/up" });
        subscribeClient.Disconnect();
    }
```

```
    private void SubscribeClientMqttMsgPublishReceived(object sender,
MqttMsgPublishEventArgs e)
    {
        //messageVariable.Value = "Message received: " +
System.Text.Encoding.UTF8.GetString(e.Message);
        string toBeSearched = "Speed_Hz";
        string myString="Message received: " +
System.Text.Encoding.Default.GetString(e.Message);
        //string code = myString.Substring(myString.IndexOf(toBeSearched) +
toBeSearched.Length);
        //string code = myString.Substring(myString.IndexOf(toBeSearched)+10,2);
        string code = myString.Substring(myString.IndexOf(toBeSearched)+10,1);
        if (code=="0")
        {
          code="00";
        }
        else
        {
          code = myString.Substring(myString.IndexOf(toBeSearched)+10,2);
        }
        messageVariable.Value = code;

    }

    private MqttClient subscribeClient;
    private IUAVariable messageVariable;
}
```

And this is the runtime screenshot

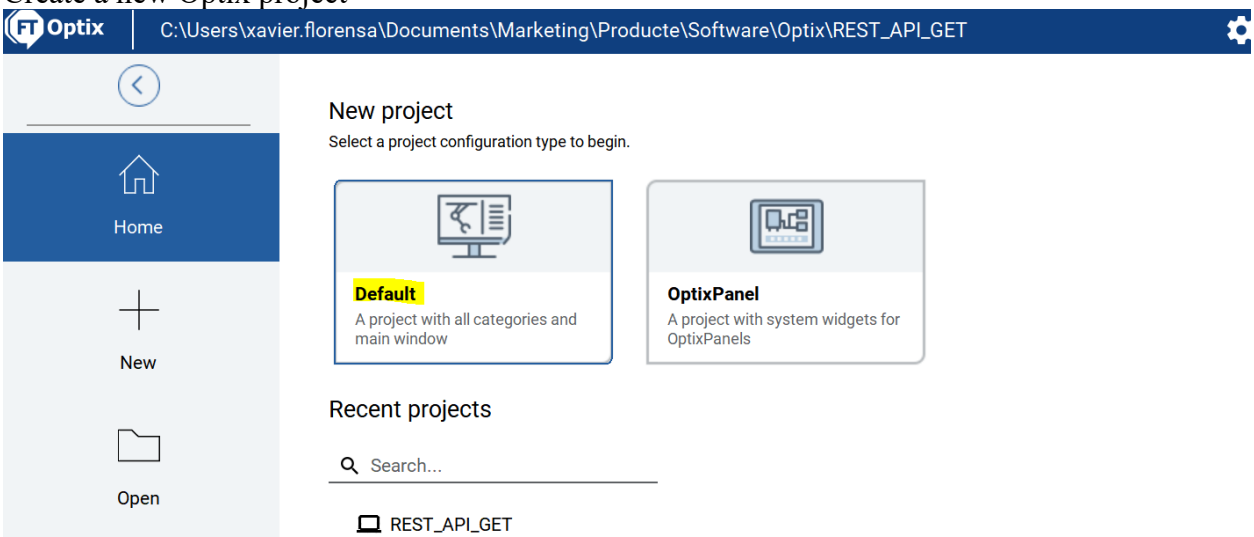


21. Creating an Http REST API client

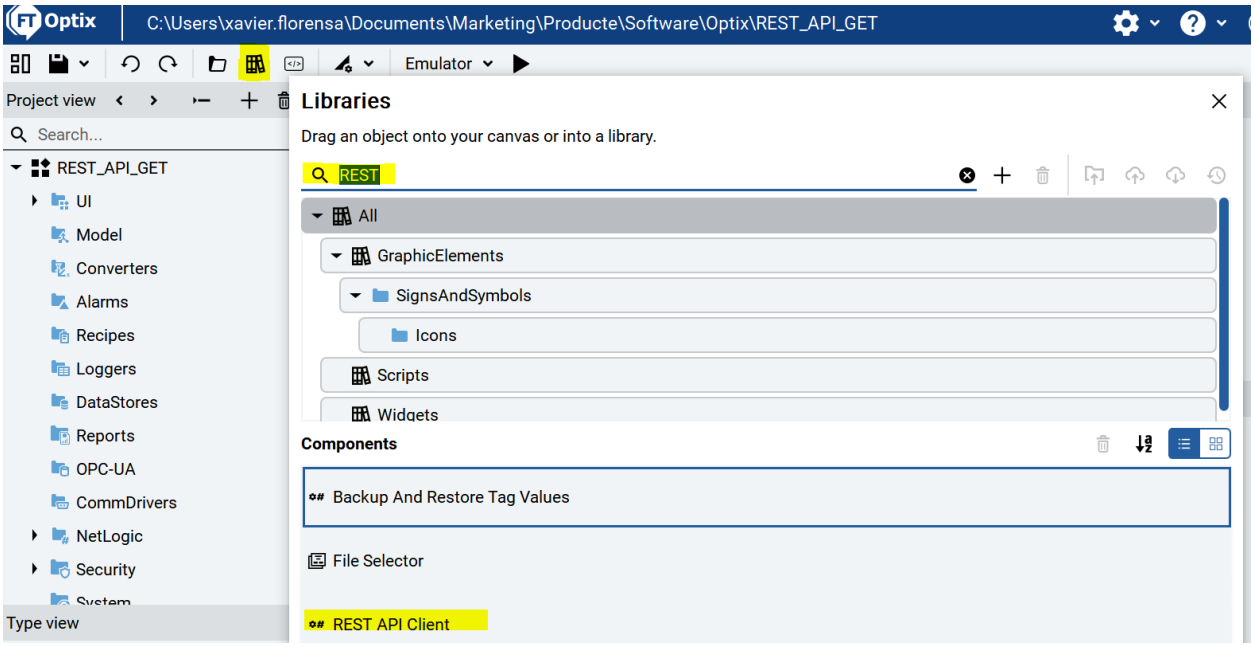
You do not need this if you want to store to an InfluxDB database and you are in version 1.4 of FT Optix. You can jump to chapter Local External InfluxDB in FTOptix v1.4

21.1. GET request

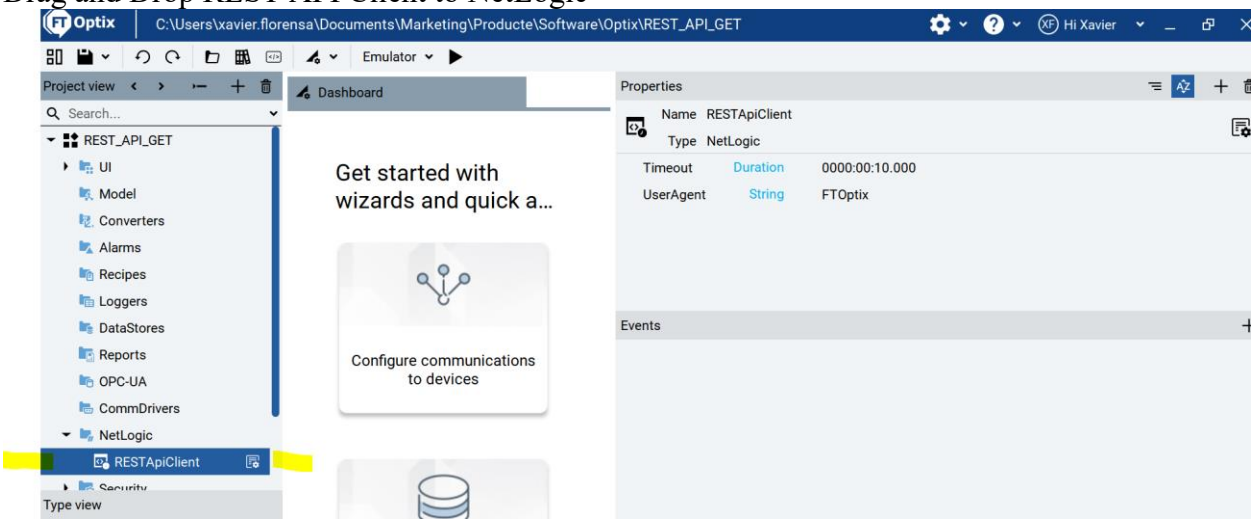
Create a new Optix project



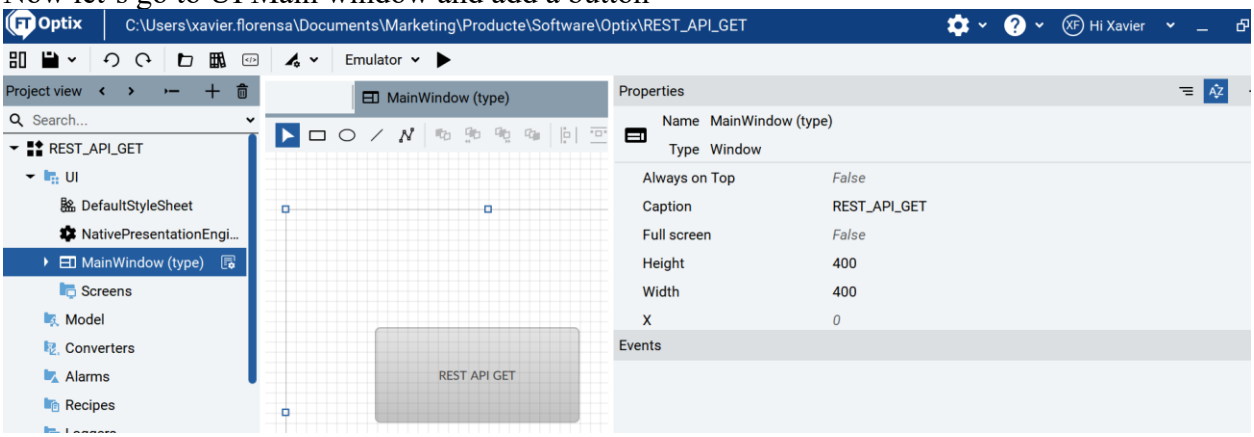
Go to libraries



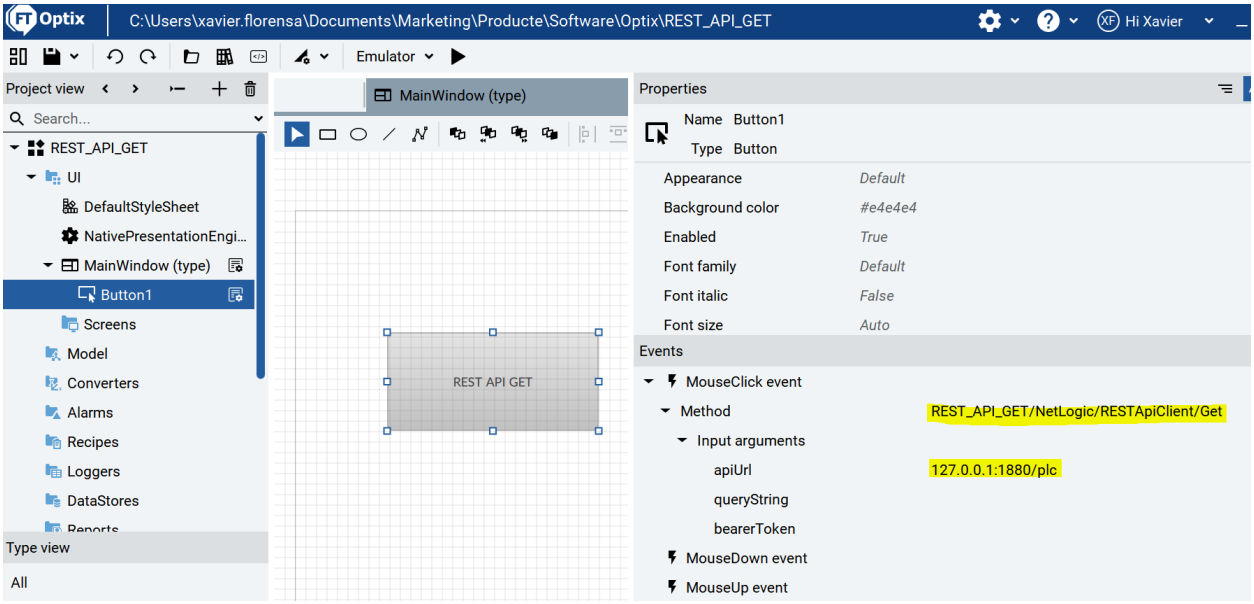
Drag and Drop REST API Client to NetLogic



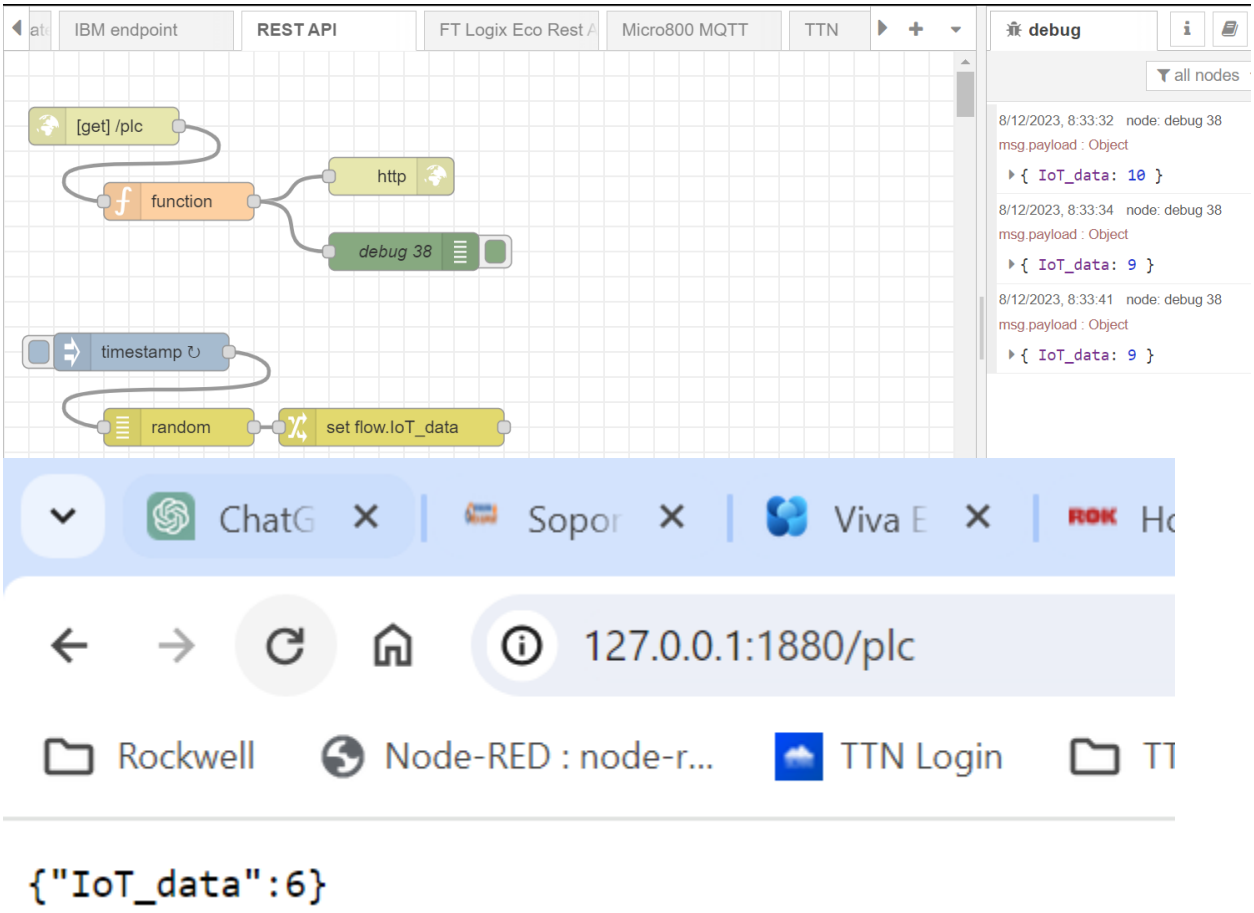
Now let's go to UI Main window and add a button

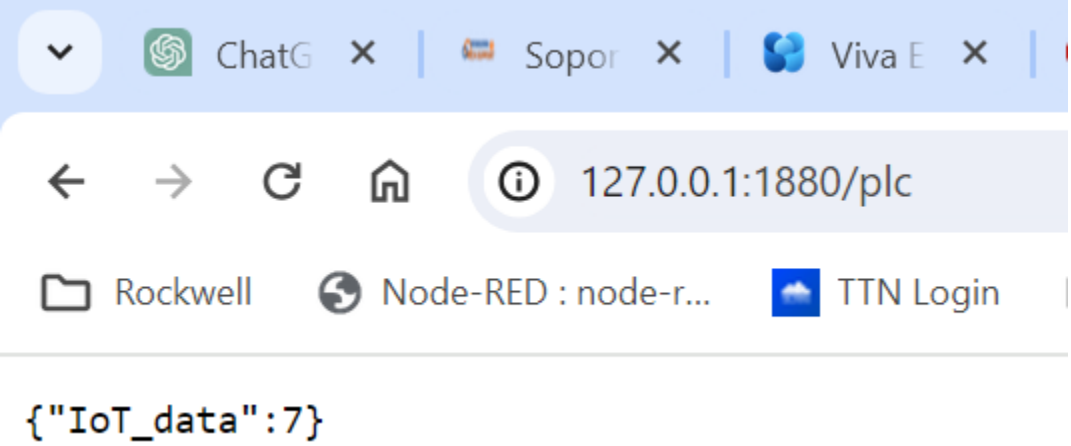


Click on the button to see the properties and add a Mouse Click event with the URL of the API

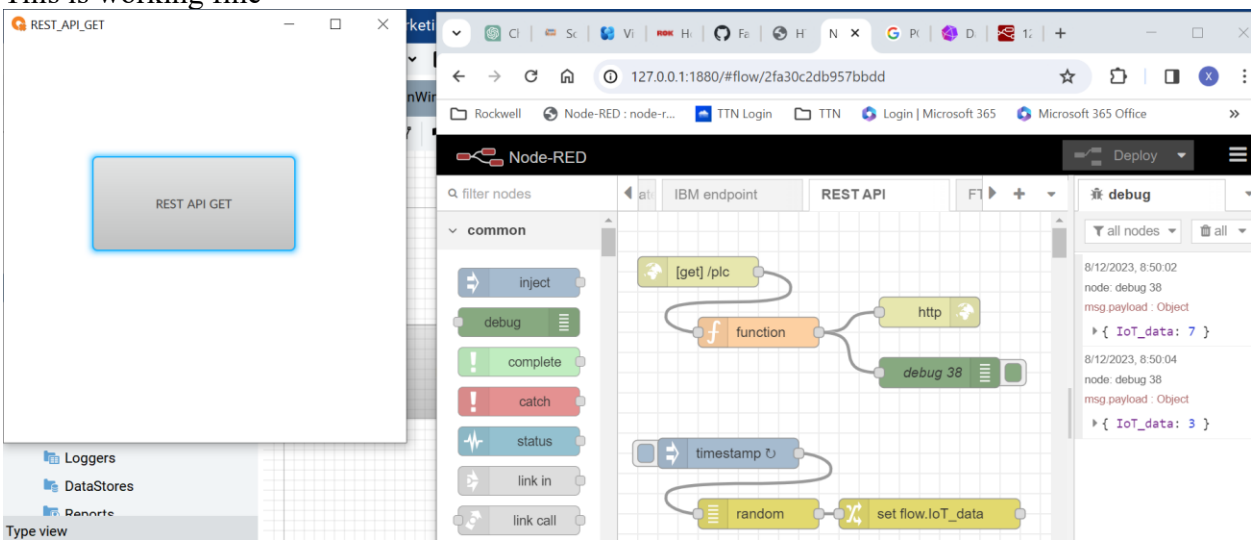


First of all we have to test these API URL, we are using node red here, to create an API that returns a random number between 1 and 10

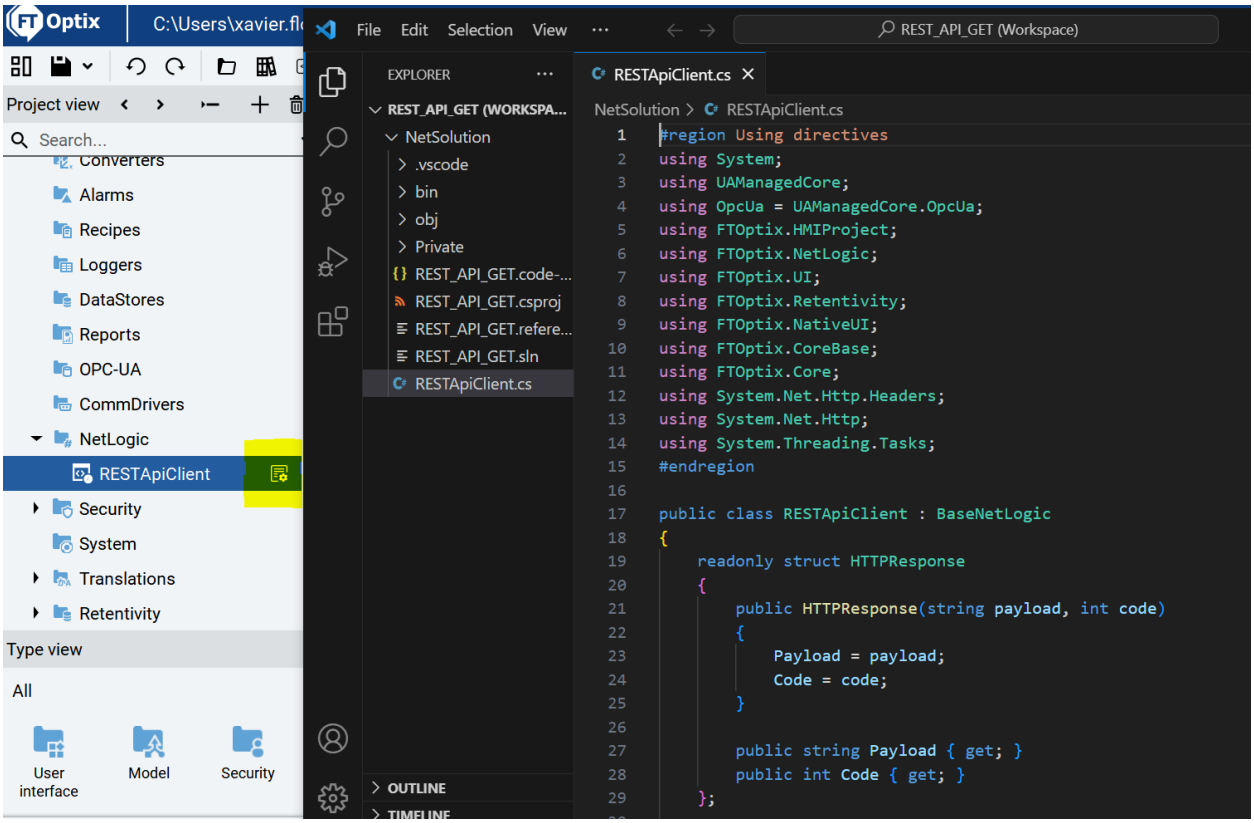




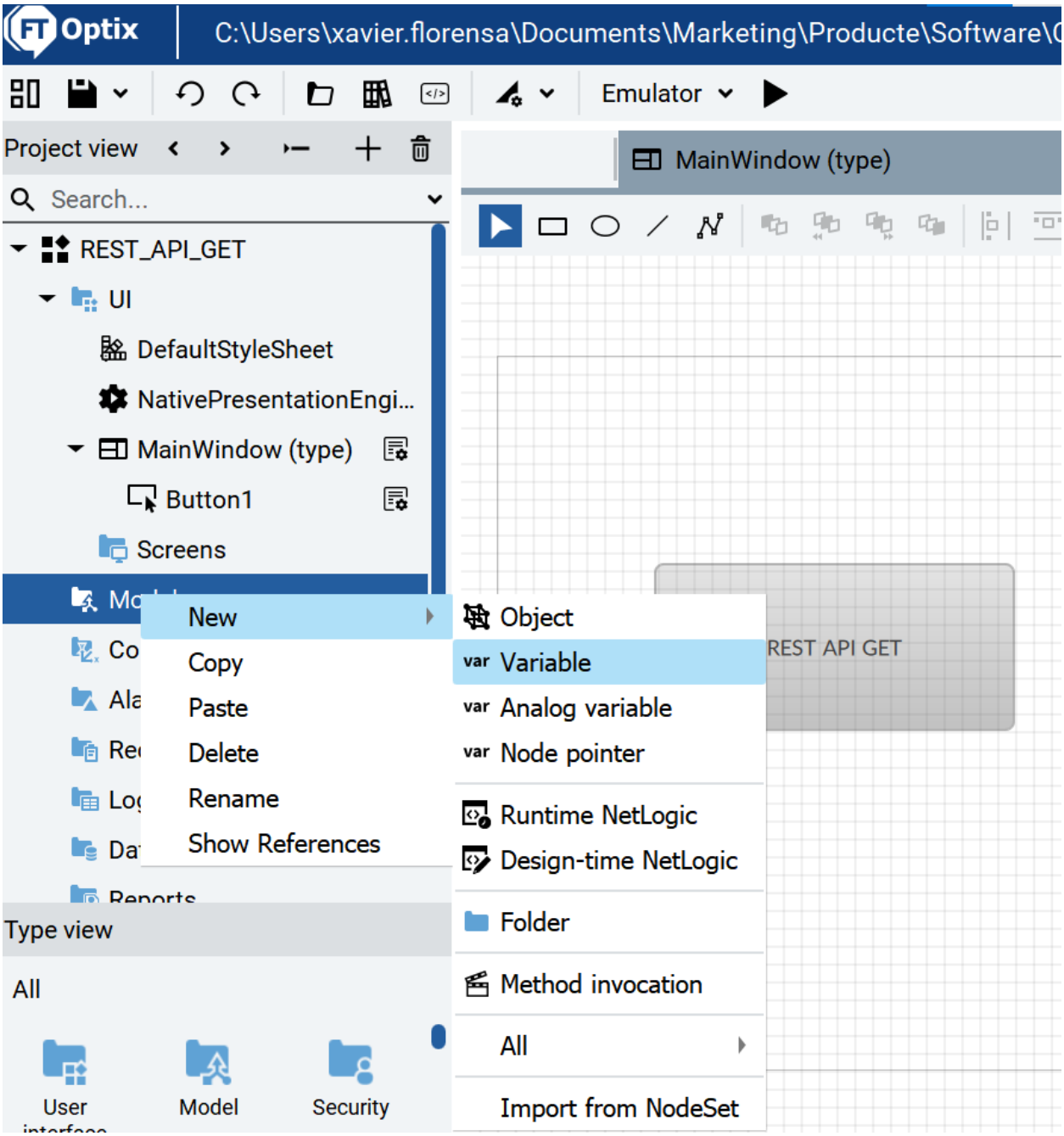
Now let's test the Optix runtime
This is working fine



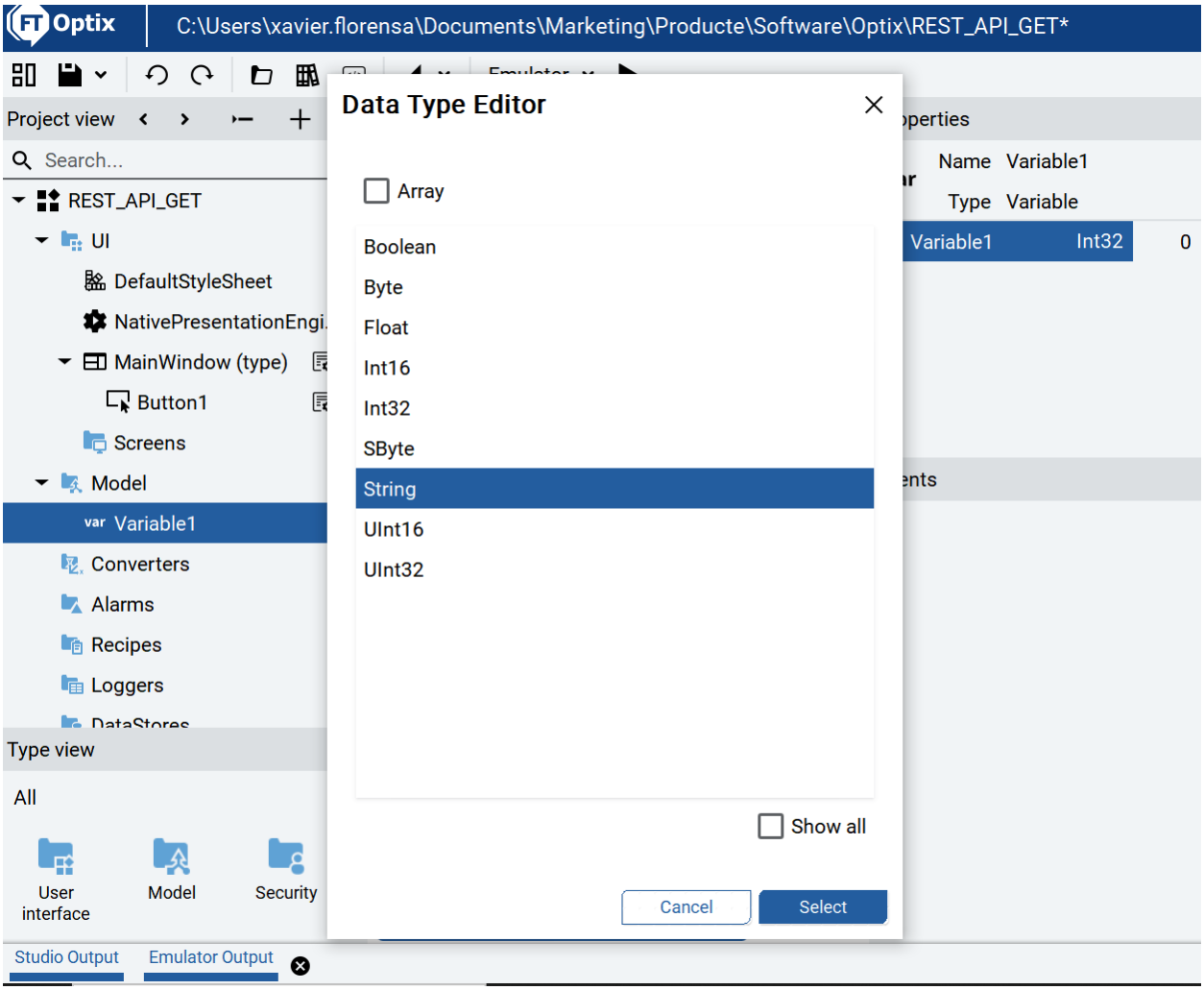
But we need to get the data on Optix
Let's open the C# script



Let’s create a new variable to store the response data



And change variable type



Now introduce the variable somehow on the script

```
var Variable1 = Project.Current.GetVariable("Model/Variable1");
```

So you can update the variable value like this

```
Variable1.Value = System.Text.Encoding.UTF8.GetString(Response);
```

Were response is the result of the Http request

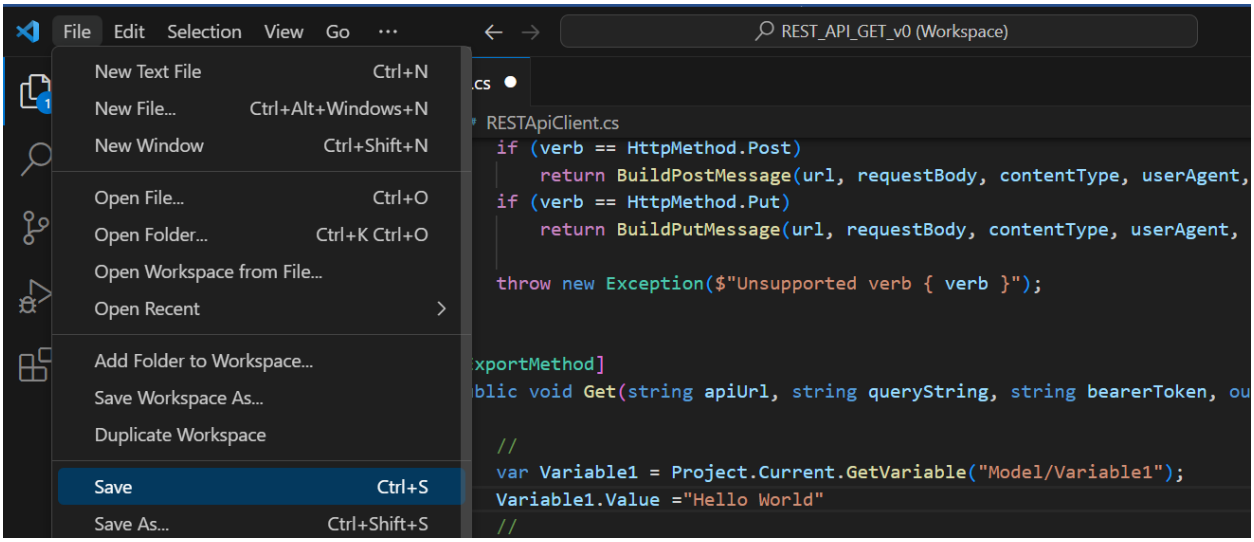
We can introduce our variable for instance here

```
[ExportMethod]
public void Get(string apiUrl, string queryString, string bearerToken, out string response, out int
{
    TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
    UriBuilder uriBuilder = new UriBuilder(apiUrl);
    uriBuilder.Query = queryString;

    var requestMessage = BuildMessage(HttpMethod.Get, uriBuilder.Uri, "", bearerToken, "");
    var requestTask = PerformRequest(requestMessage, timeout);
    var httpResponse = requestTask.Result;

    (response, code) = (httpResponse.Payload, httpResponse.Code);
}
```

Like this

```
[ExportMethod]
public void Get(string apiUrl, string queryString, string bearerToken, out string response, out int
{
    //
    var Variable1 = Project.Current.GetVariable("Model/Variable1");
    Variable1.Value ="Hello World"
    //
    TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
    UriBuilder uriBuilder = new UriBuilder(apiUrl);
    uriBuilder.Query = queryString;

    var requestMessage = BuildMessage(HttpMethod.Get, uriBuilder.Uri, "", bearerToken, "");
    var requestTask = PerformRequest(requestMessage, timeout);
    var httpResponse = requestTask.Result;

    (response, code) = (httpResponse.Payload, httpResponse.Code);
}
```

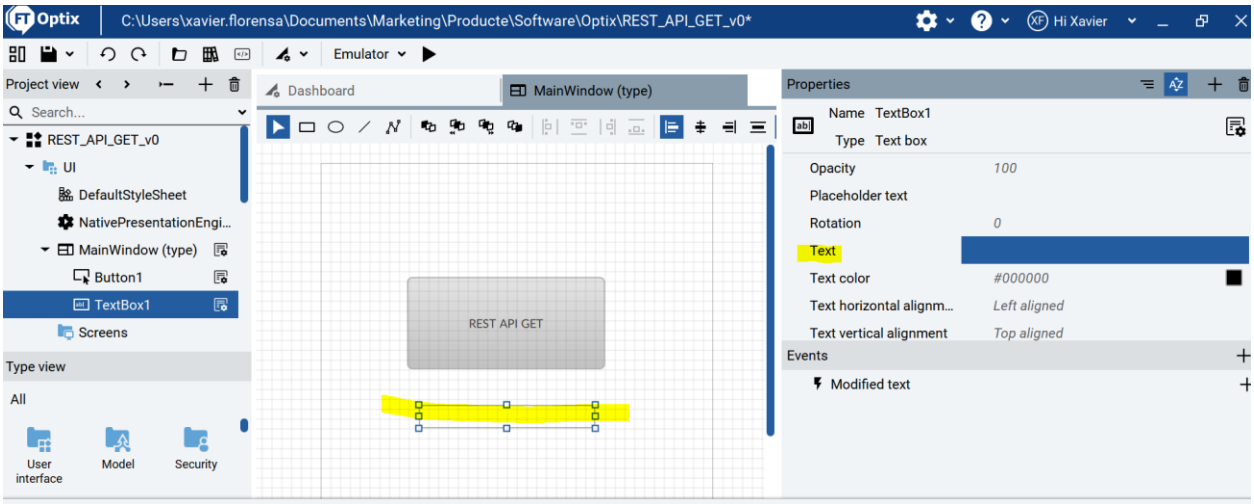
Save the file



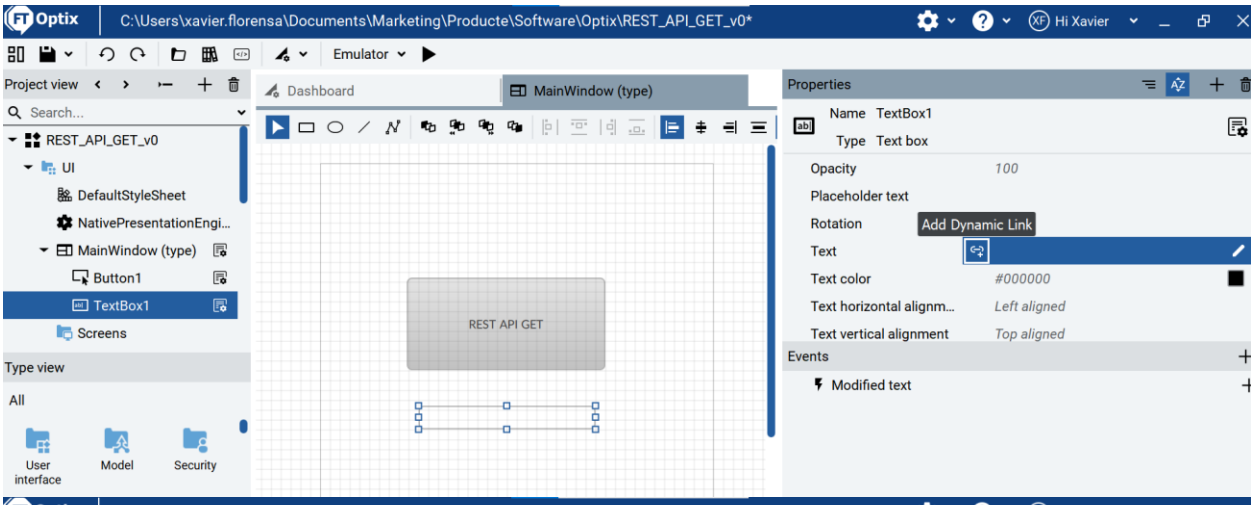
2023-12-08 09:17:07.807;NetHelper;222;Warning;16;User .NET solution failed to build: C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\Optix\REST_API_GET_v0\ProjectFiles\NetSolution\RESTApiClient.cs(152,39): error CS1002: ; expected [C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\Optix\REST_API_GET_v0\ProjectFiles\NetSolution\REST_API_GET_v0.csproj]

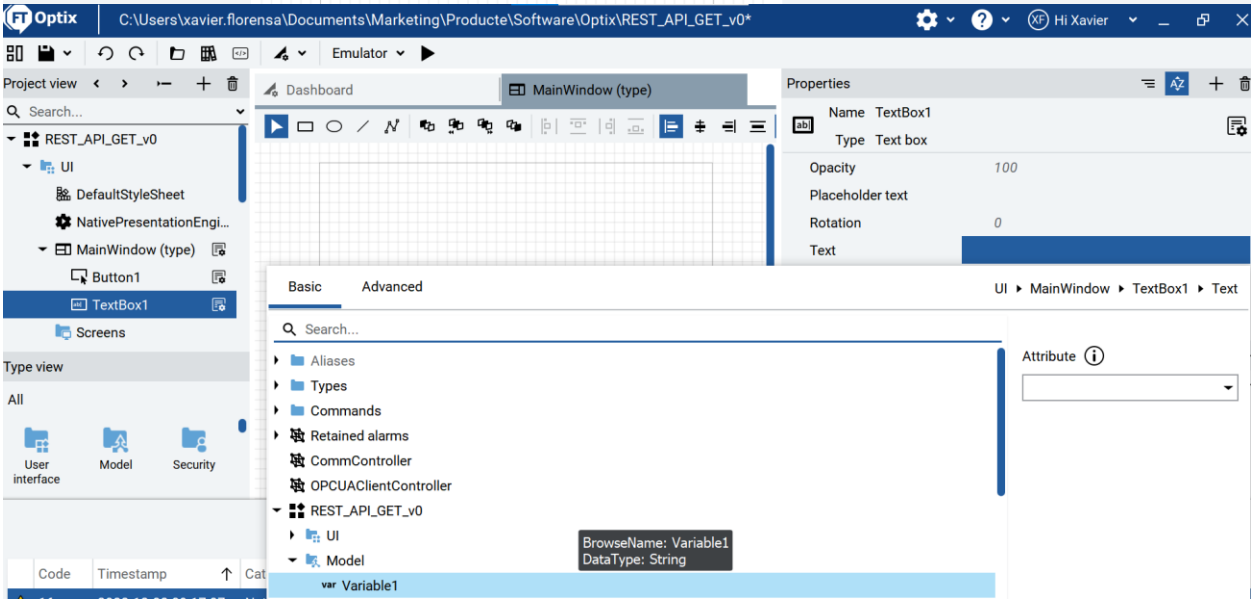
Let's correct this

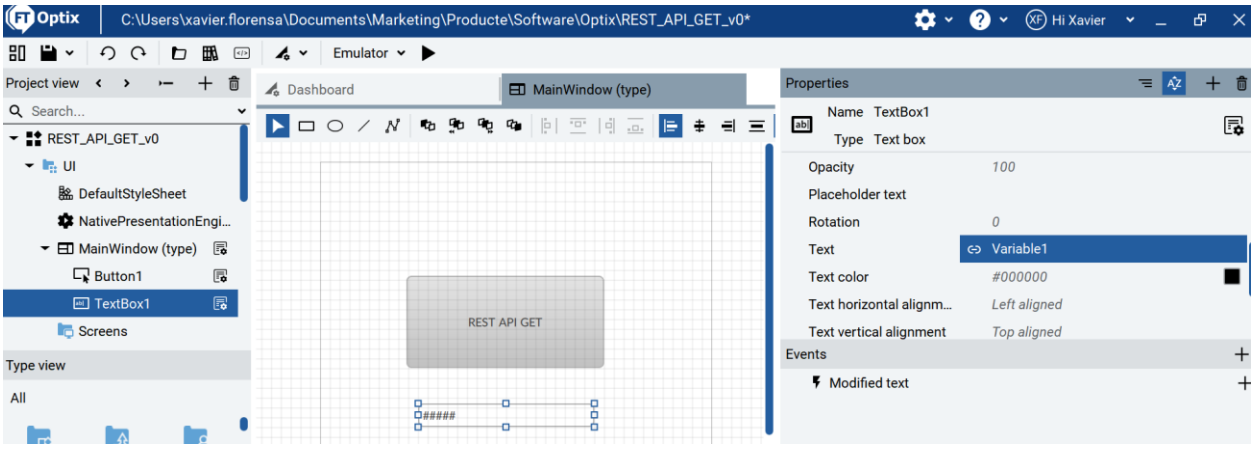
Now introduce a text box



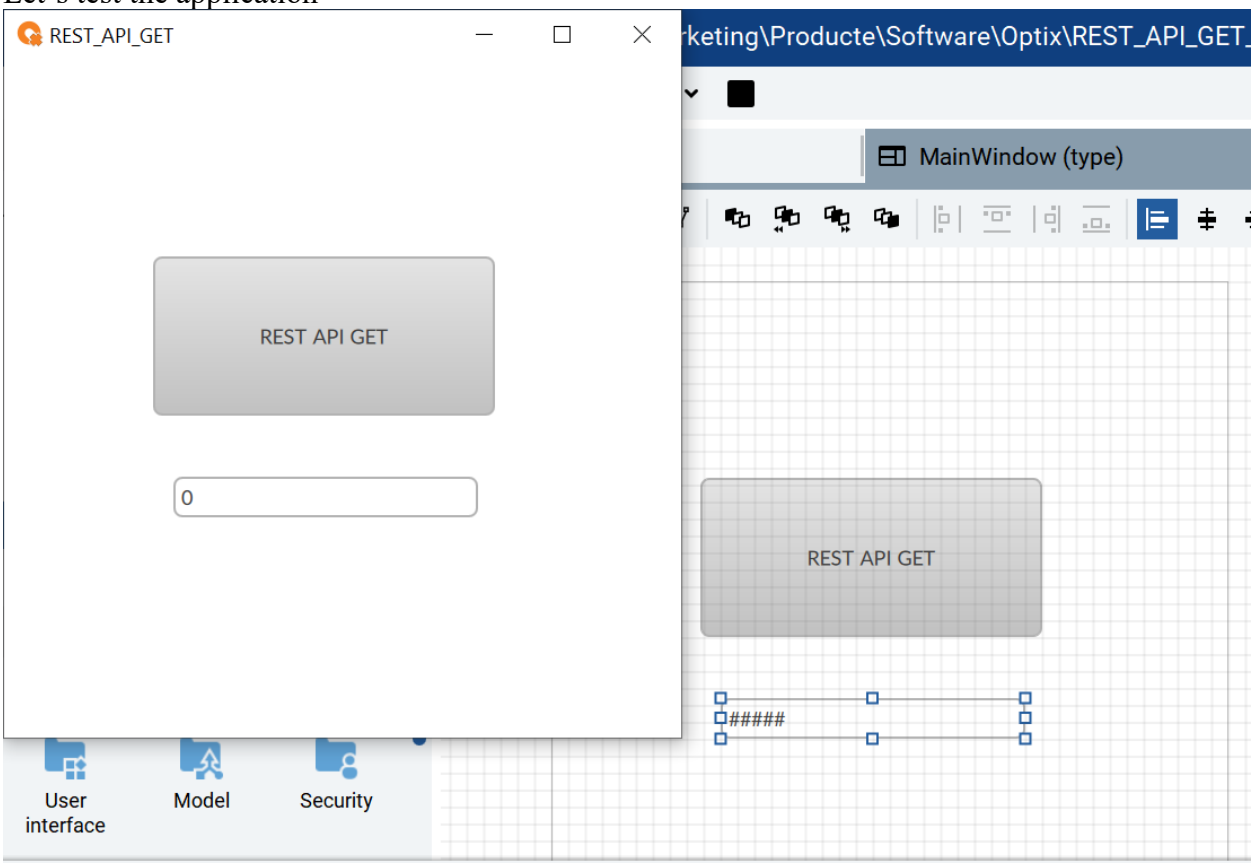
And link to Variable1



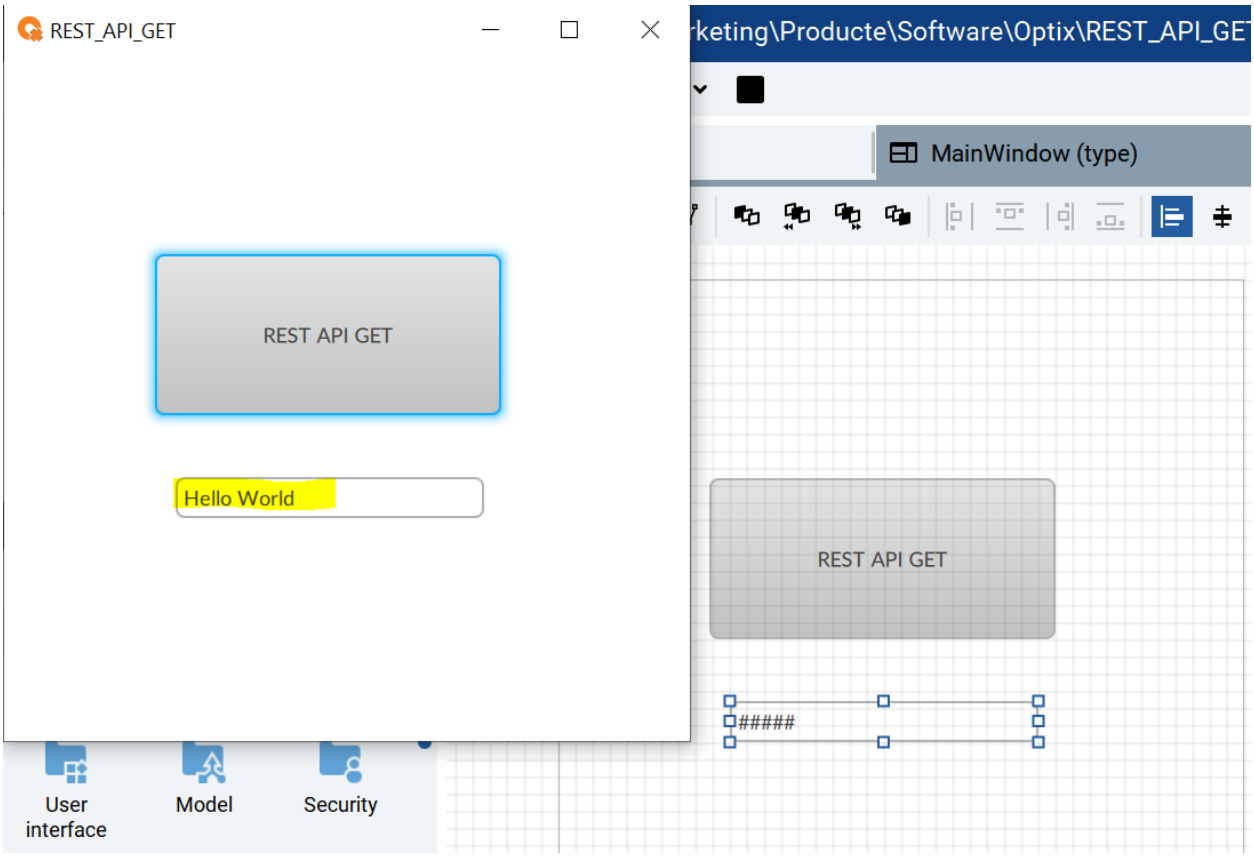




Let’s test the application



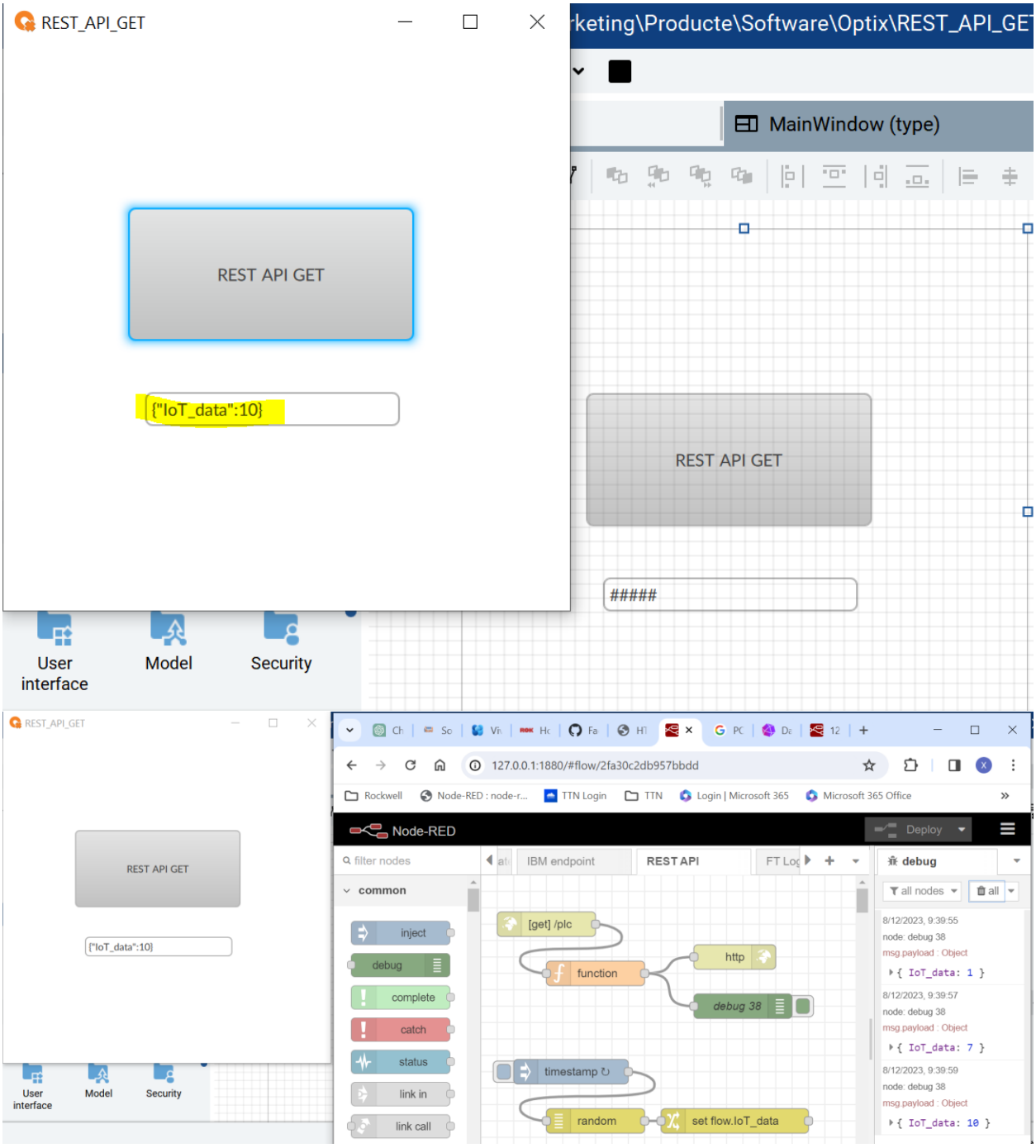
Voilà



Not let's try to get the http response on that Textbox

```
147     [ExportMethod]
148     public void Get(string apiUrl, string queryString, string bearerToken, out string
149     {
150         //
151         var Variable1 = Project.Current.GetVariable("Model/Variable1");
152         //Variable1.Value ="Hello World";
153         //
154         TimeSpan timeout = TimeSpan.FromMilliseconds(GetTimeout());
155         UriBuilder uriBuilder = new UriBuilder(apiUrl);
156         uriBuilder.Query = queryString;
157
158         var requestMessage = BuildMessage(HttpMethod.Get, uriBuilder.Uri, "", bearerTo
159         var requestTask = PerformRequest(requestMessage, timeout);
160         var httpResponse = requestTask.Result;
161         //
162         Variable1.Value =httpResponse.Payload;
163         //
164         (response, code) = (httpResponse.Payload, httpResponse.Code);
165     }
```

Voilà, it works



We have created a API Rest client to perform a GET Http request
As you can see on this video
https://youtu.be/-jf5SwSq8Us

21.2. Installing InfluxDB on windows

Using PowerShell in Administrator mode
Download with this command

```
wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.7.3-windows.zip -Use
BasicParsing -OutFile influxdb2-2.7.3-windows.zip

Expand-Archive .\influxdb2-2.7.3-windows.zip -DestinationPath 'C:\Program Files\I
nfluxData\influxdb\'
```

From this guide
https://portal.influxdata.com/downloads/

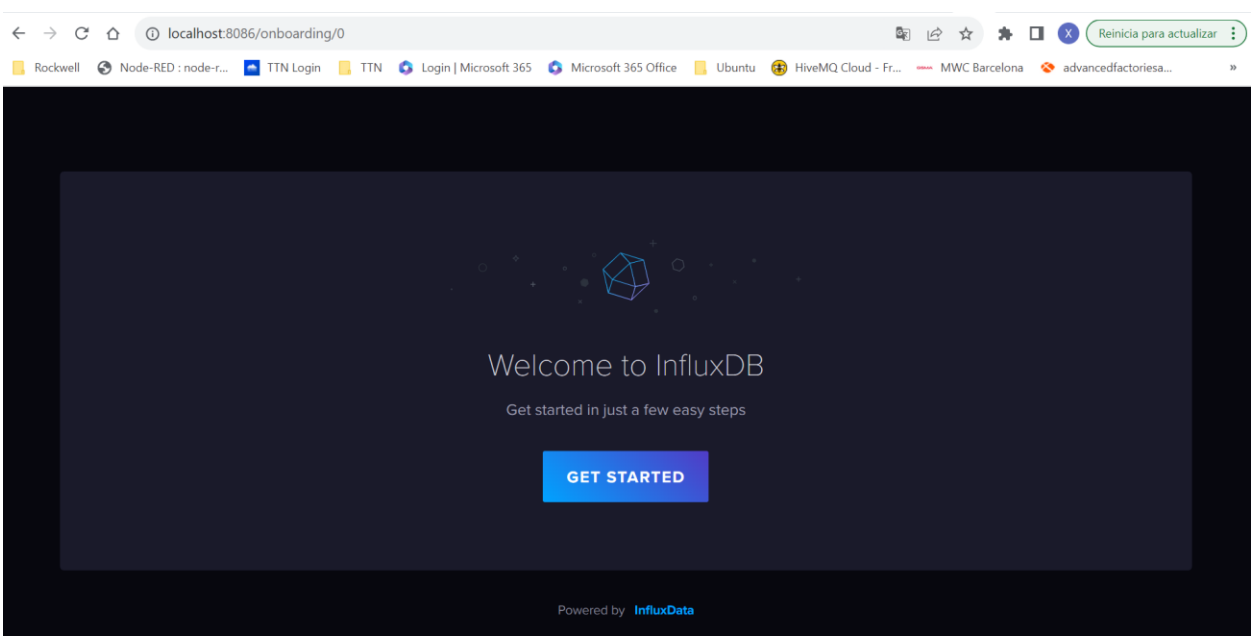
Start InfluxDB

In **Powershell**, navigate into `C:\Program Files\InfluxData\influxdb` and start InfluxDB by running the `influxd` daemon:

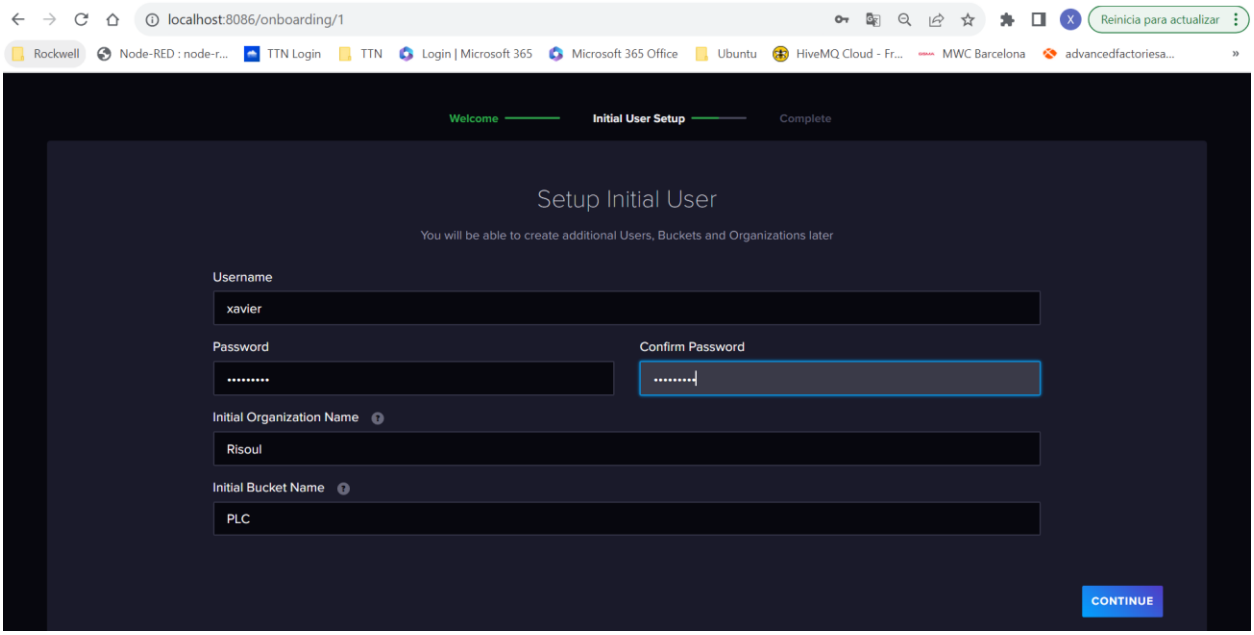
```
> cd -Path 'C:\Program Files\InfluxData\influxdb'
> ./influxd
```


```
> cd -Path 'C:\Program Files\InfluxData\influxdb'
> ./influxd
```

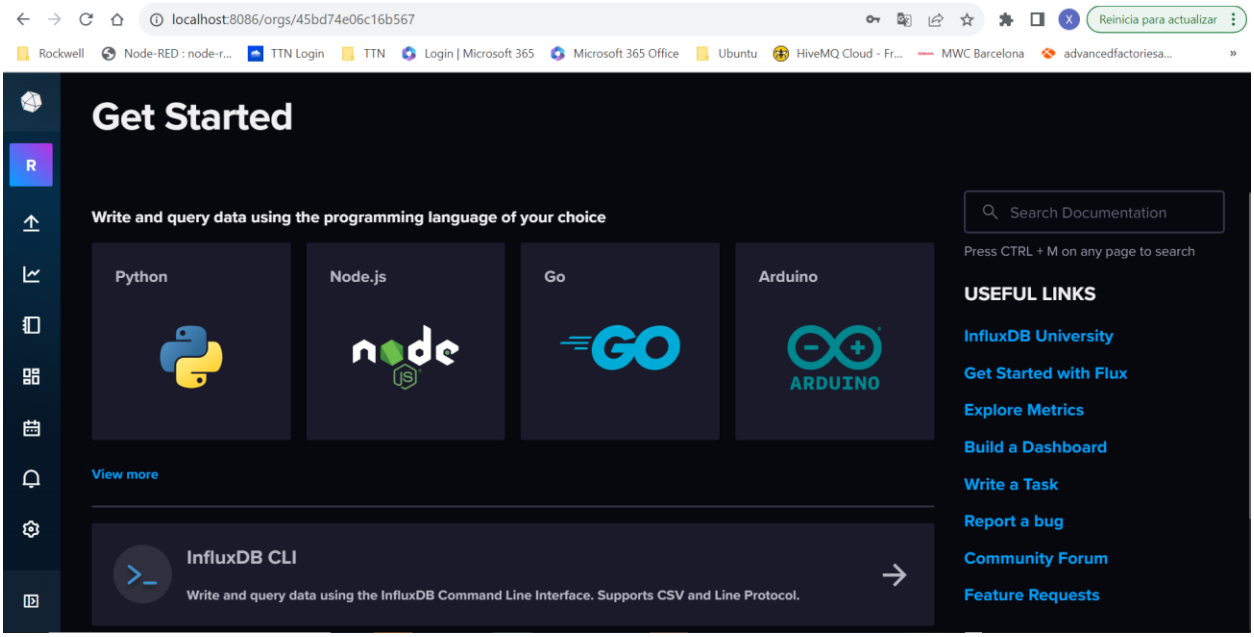

cd -Path 'C:\Program Files\InfluxData\influxdb'
./influxd
Open an explorer with localhost and port 8086
Username
xavier
Password



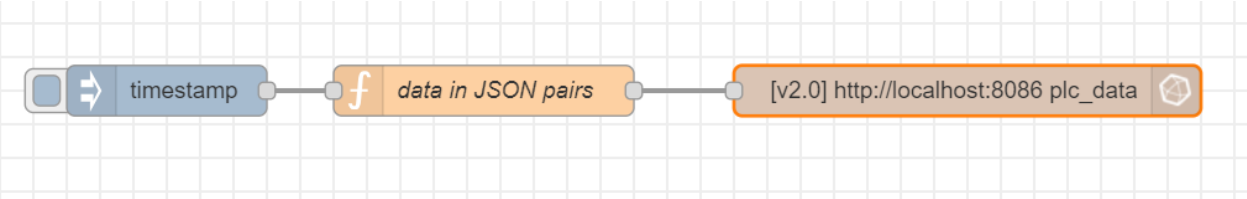
Start building your influxdb use cases

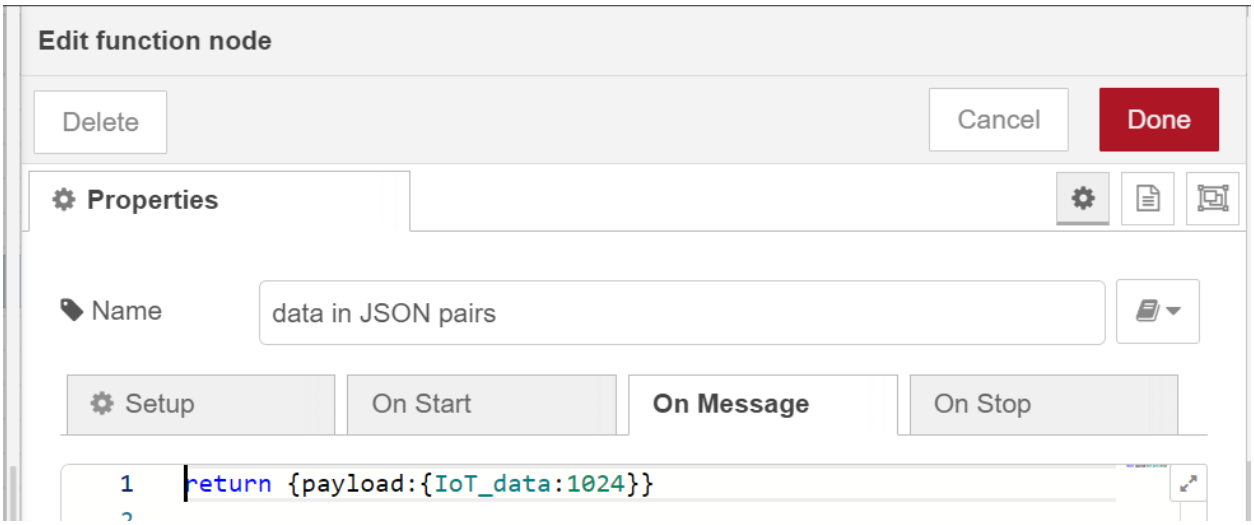


API tocken
MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-zGmMeLyGYKgHA==
Select Quickstart

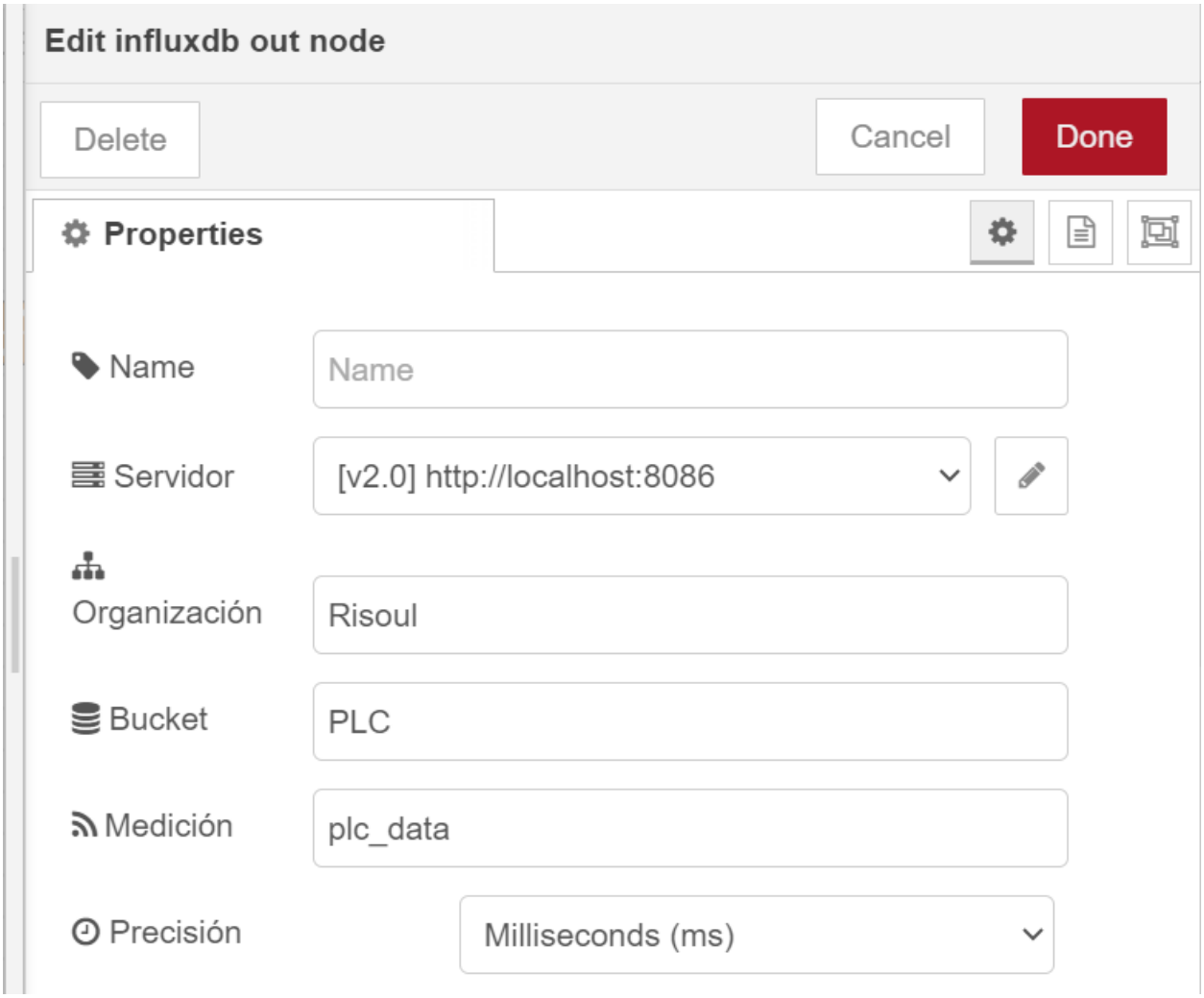


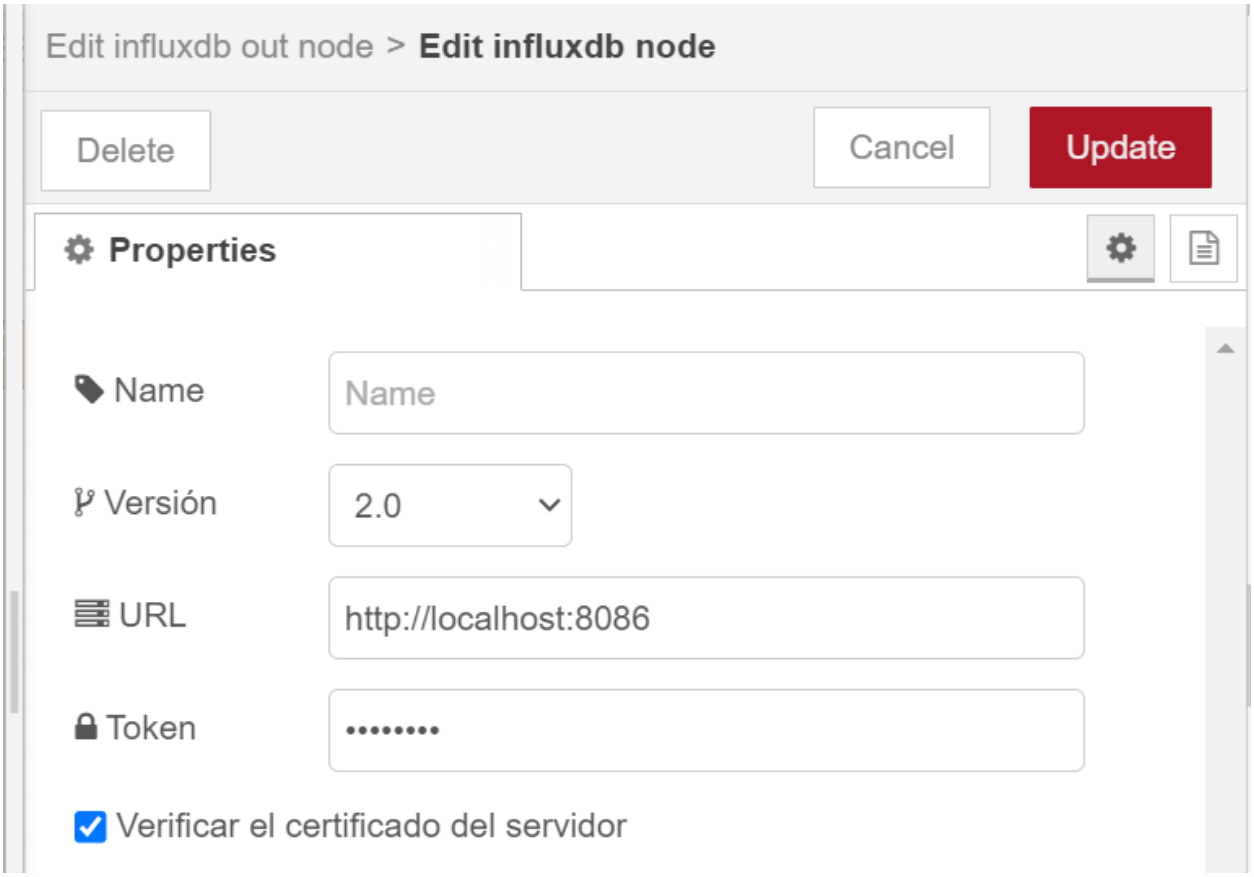
21.3. Inject your first data with node-red

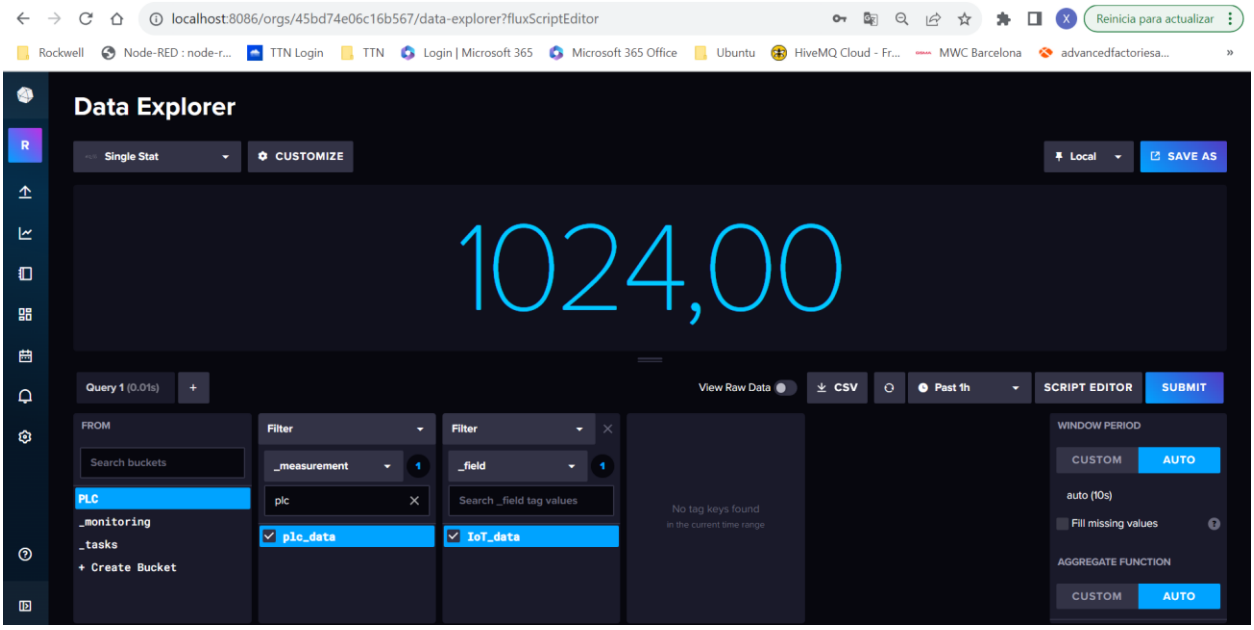




Let's try to inject to INfluxDB using an Http request





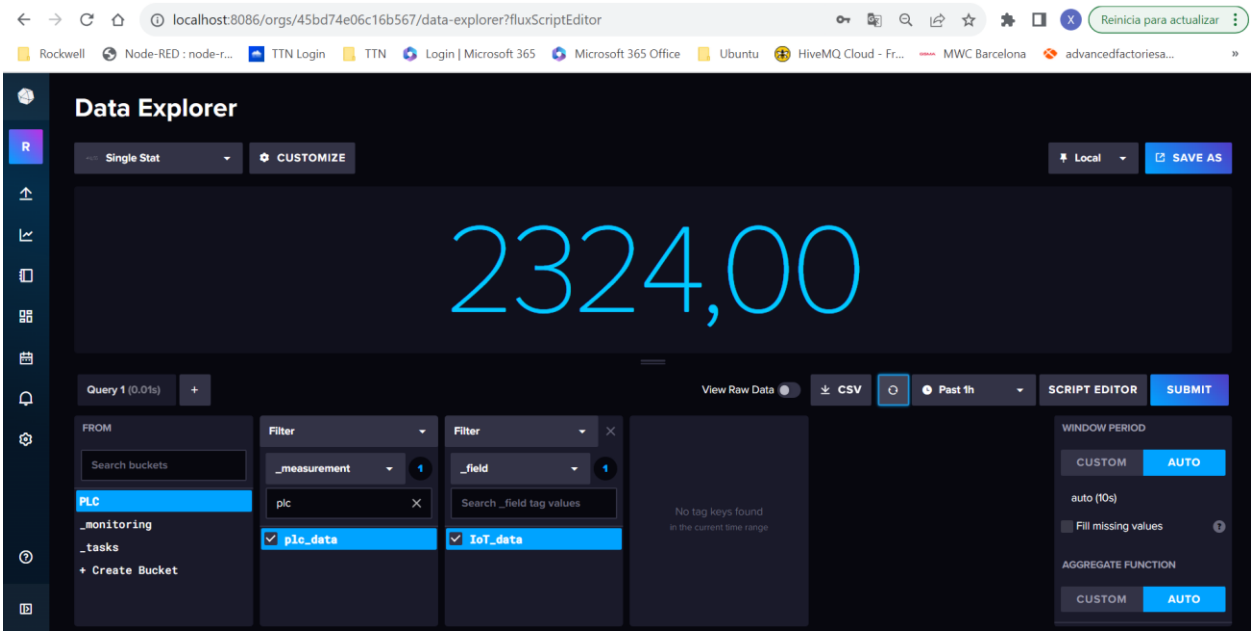


21.4. Injecting to InfluxDB from command line

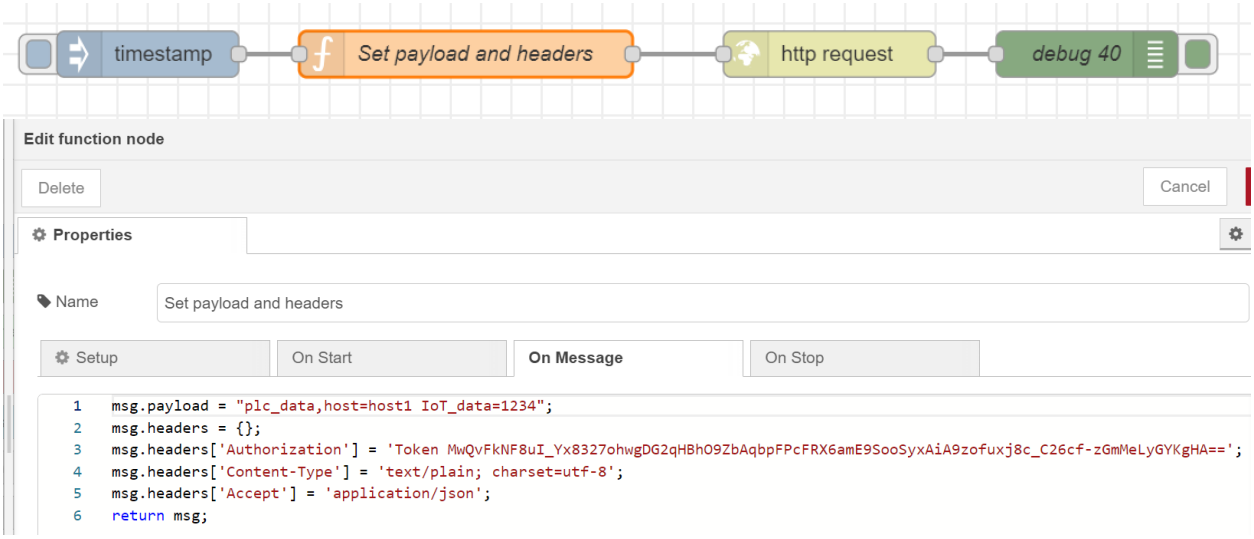
Now let's try to build the http request ourselves. From windows command line

curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=PLC&precision=s" --header "Authorization: Token

MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-zGmMeLyGYKgHA==" --data-raw "plc_data,host=host1 IoT_data=2324"

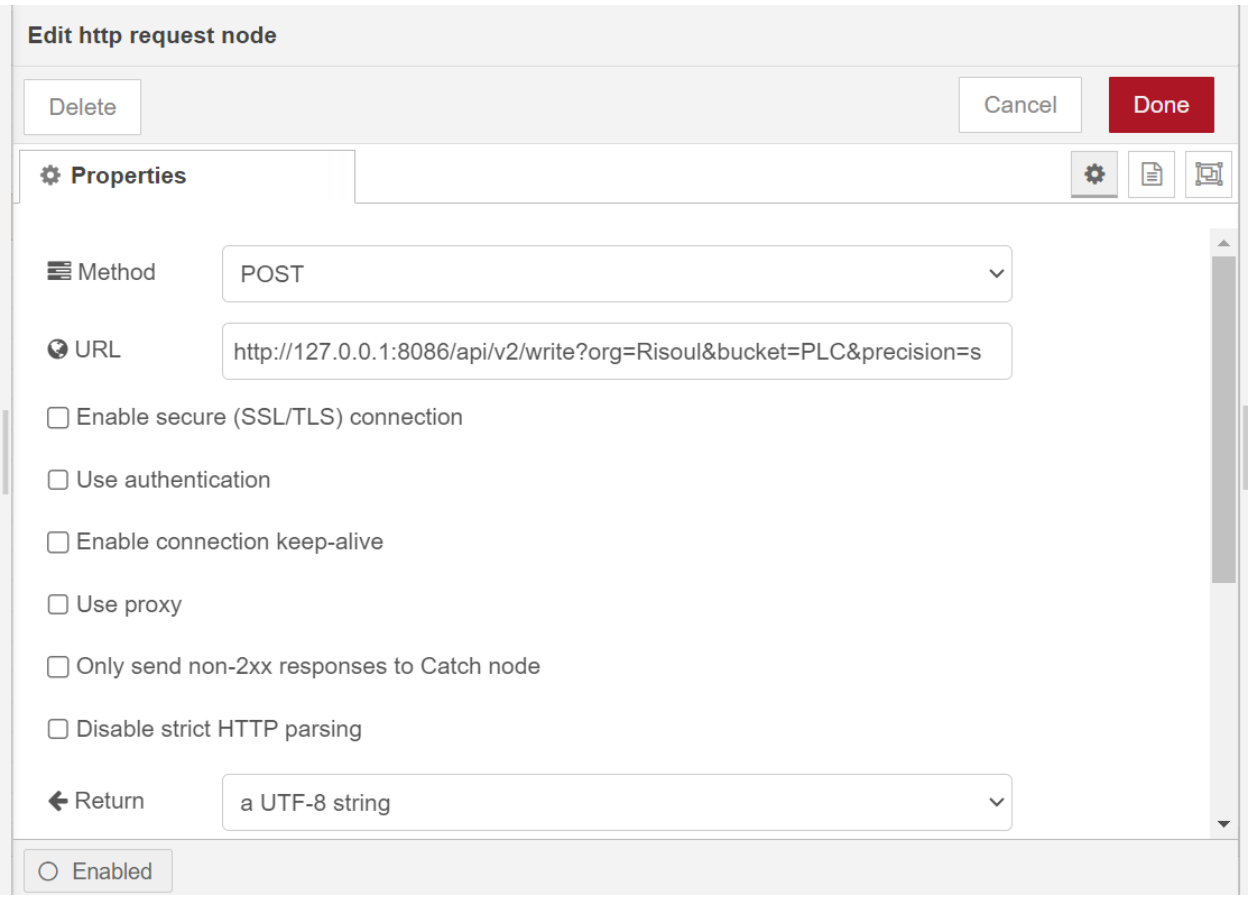


Let's do this from Node-RED

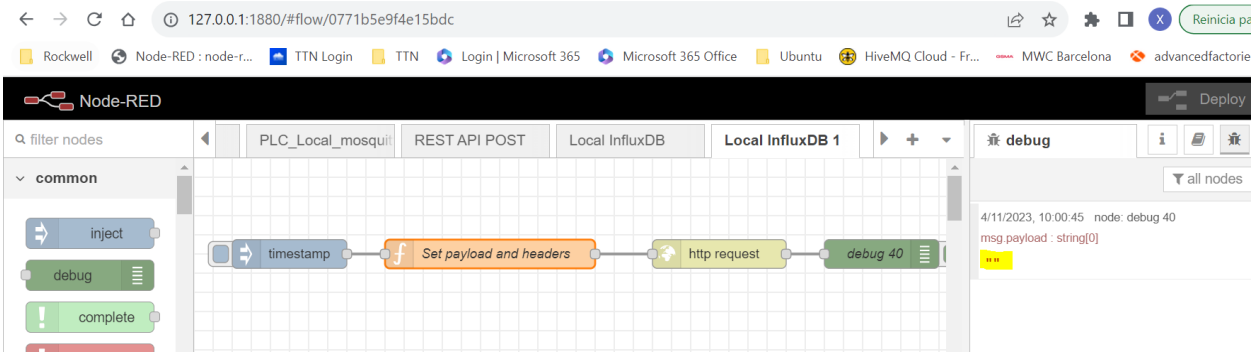


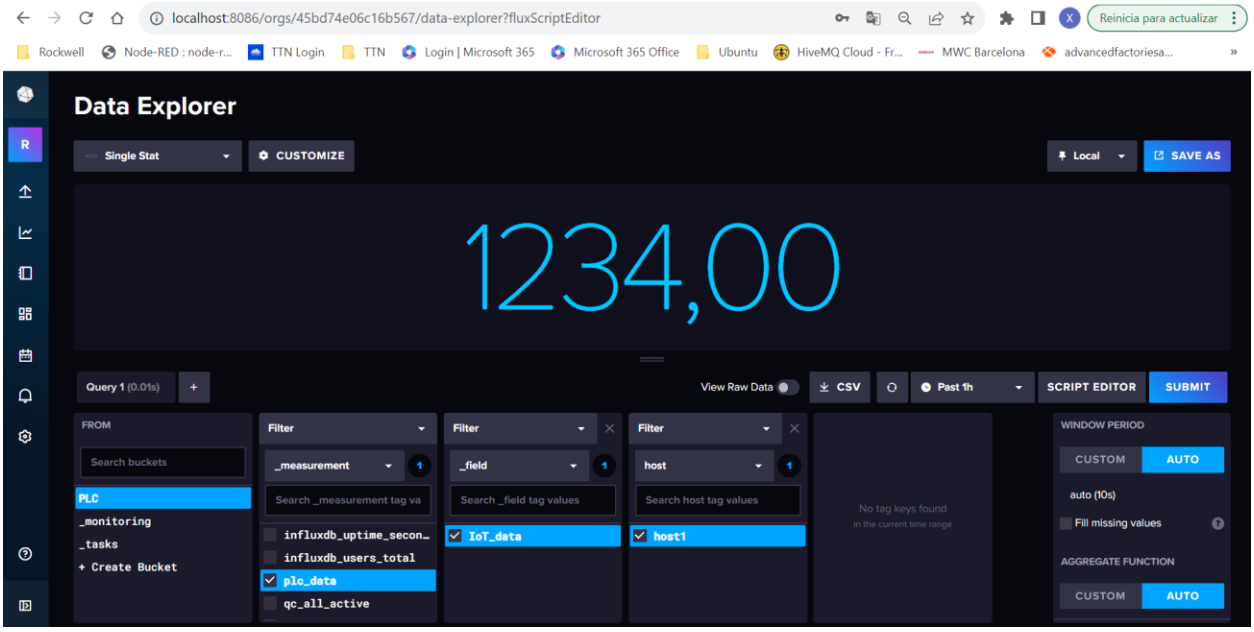

```
msg.payload = "plc_data,host=host1 IoT_data=1234";
msg.headers = {};
msg.headers['Authorization'] = 'Token
MwQvFkNF8uI_Yx8327ohwgDG2qHBhO9ZbAqbpFPcFRX6amE9SooSyxAiA9zofuxj8c_C26cf-
zGmMeLyGYKgHA==';
```

```
msg.headers['Content-Type'] = 'text/plain; charset=utf-8';
msg.headers['Accept'] = 'application/json';
return msg;
```



Success





Now, let's try to do this from Factory Talk Optix

21.5. POST request to local InfluxDB

First let's try the right http request

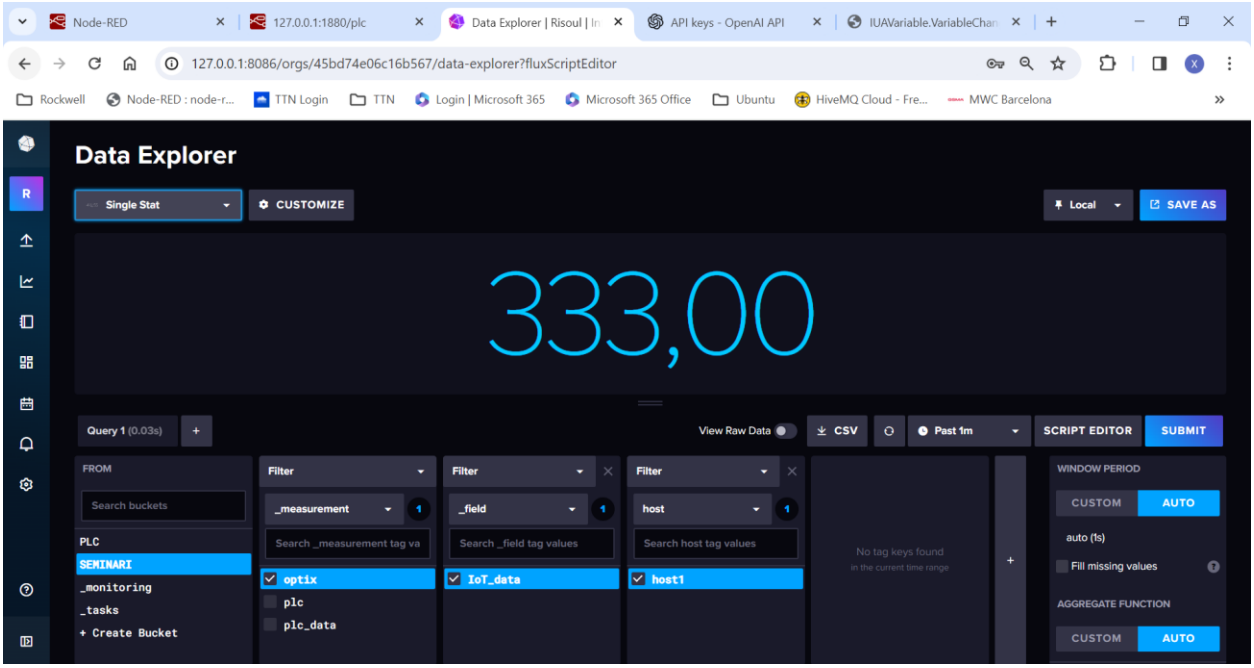
API token example

c4K4HCsdkgH_9Vv1zBDLJ2ay8QR9ORkYTIPTclHo7PI8--BPcuhEBQkb0sl5QCbAEVozZgzuA9vWk2iHnYxijg==

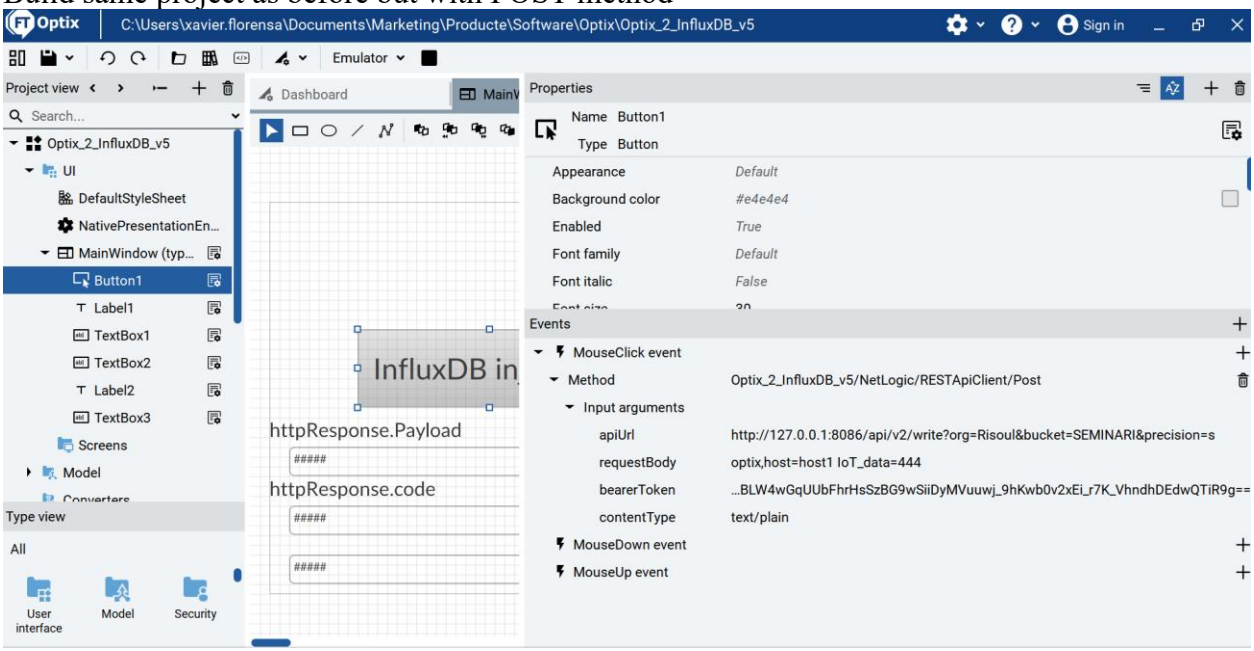
```
C:\Users\xavier.florensa>curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s" --header
 "Authorization: Token 3fIg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiR9g==" --data-
raw "optix,host=host1 IoT_data=333"

C:\Users\xavier.florensa>
```

curl -POST "http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s" --header "Authorization: Token 3fIg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiR9g==" --data-raw "optix,host=host1 IoT_data=333"

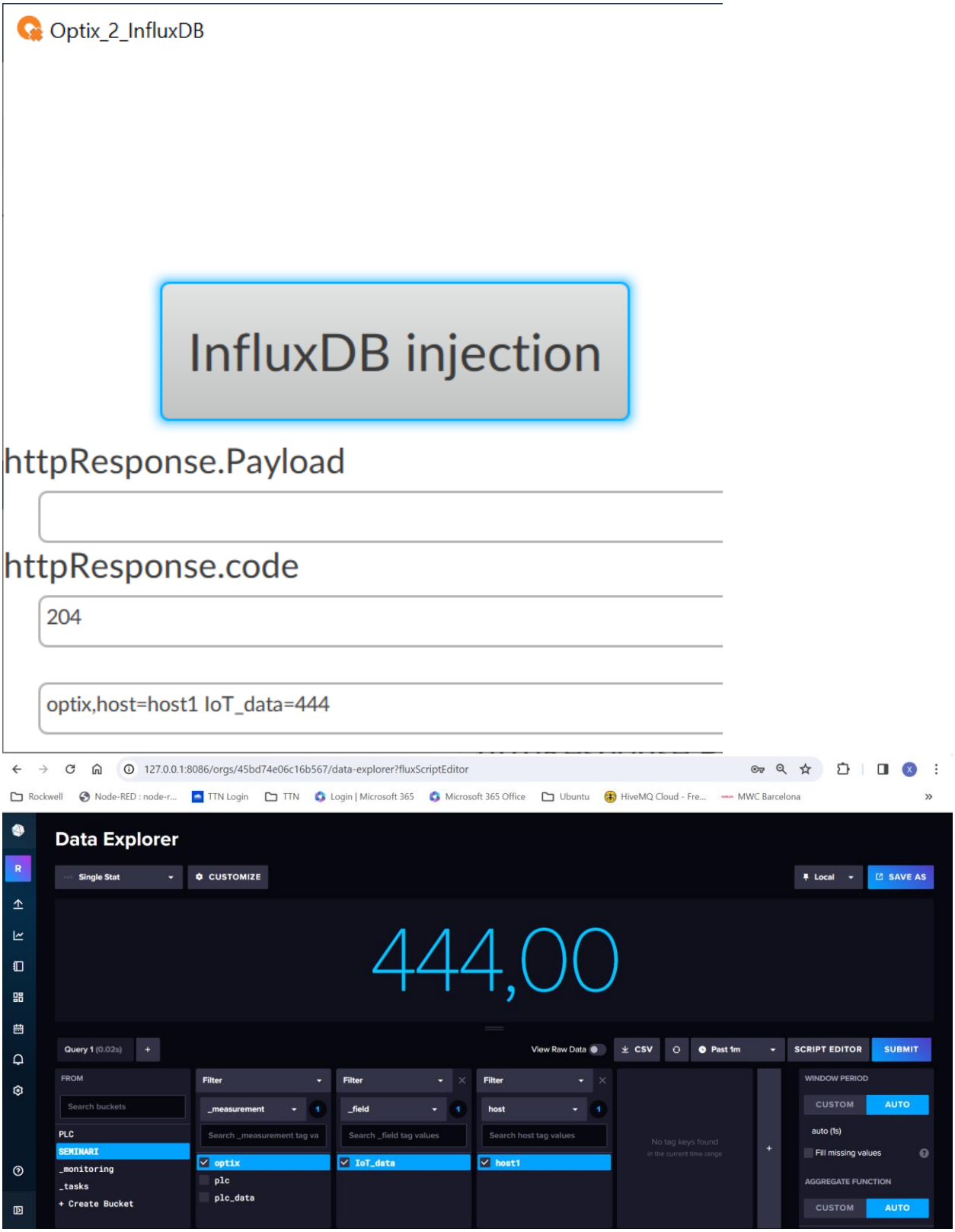
With this result



Build same project as before but with POST method



apiUrl
http://127.0.0.1:8086/api/v2/write?org=Risoul&bucket=SEMINARI&precision=s
Request Body
plc_data,host=host1 IoT_data=232
bearer Token
3fIg3FPeLRzm3nALax4VZdBLW4wGqUUbFhrHsSzBG9wSiiDyMVuuwj_9hKwb0v2xEi_r7K_VhndhDEdwQTiR9g==
Content type
text/plain



As you can see here
https://youtu.be/nqSctOoieOk

21.6. POST request to remote InfluxDB

You only have to change the URL address, organization, etc, and pay attention to this little change

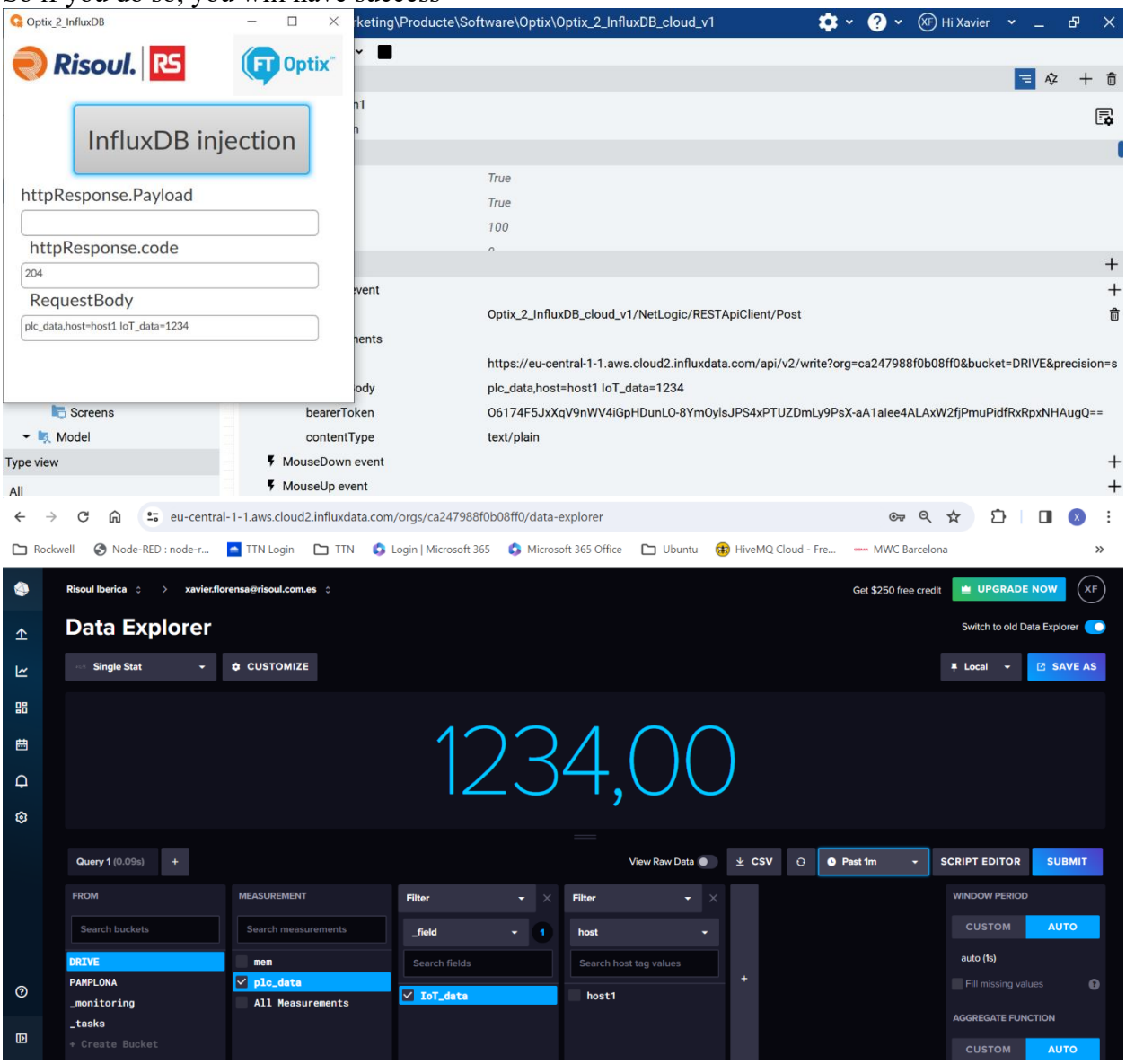
Local injection

```
91            request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearerToken);
```

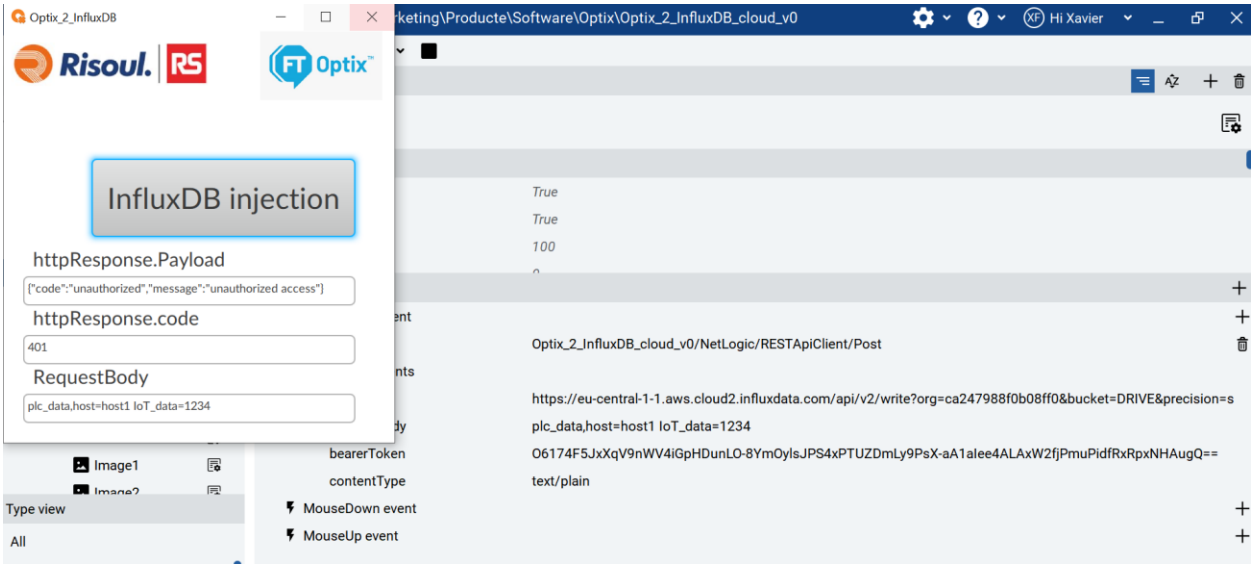
Remote injection

```
91            //request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", bearerToken);
92            request.Headers.Authorization = new AuthenticationHeaderValue("Token", bearerToken);
```

So if you do so, you will have success



But if you do not make the change from Bearer to Token, you will get this error



22. Simpler REST API

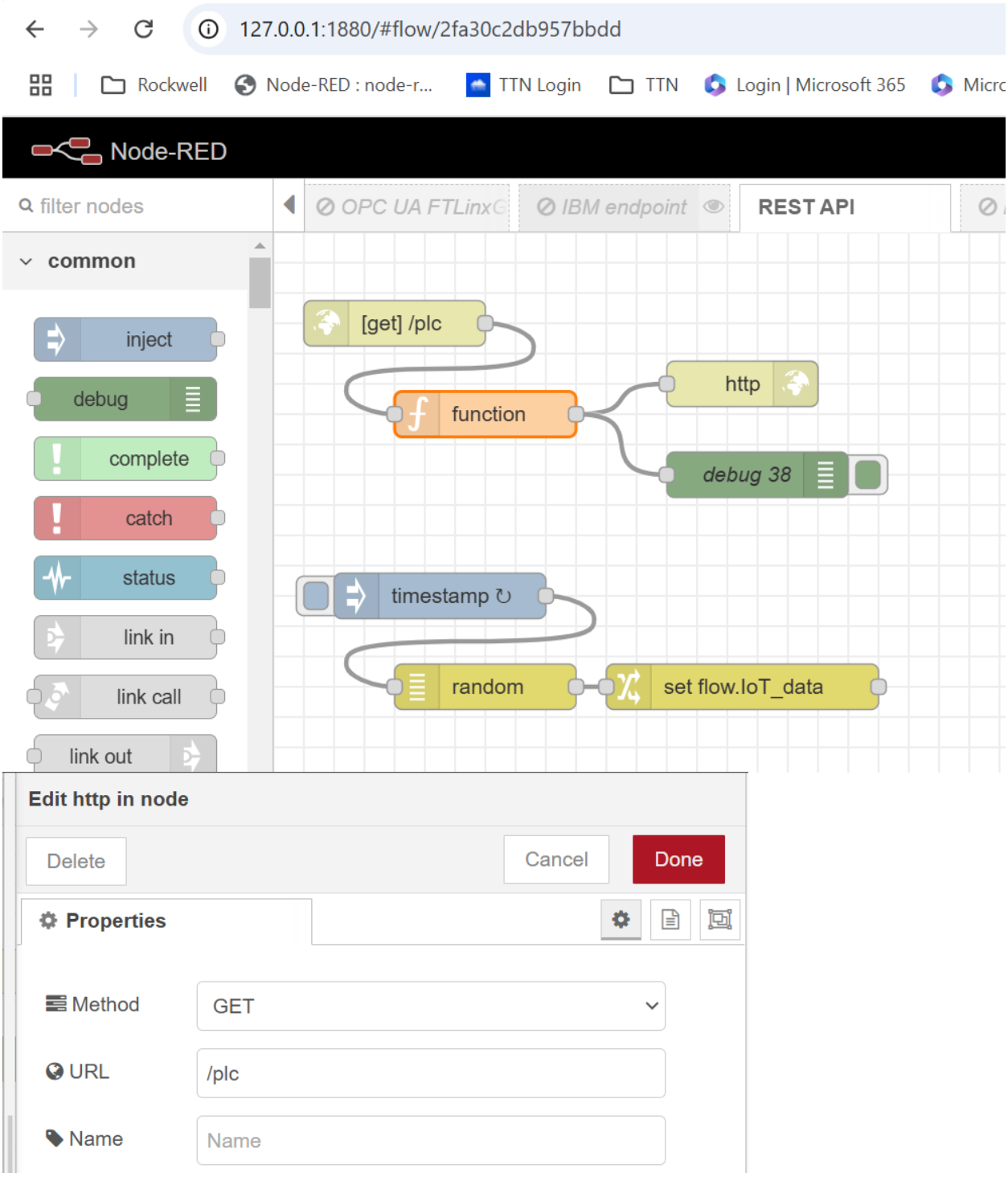
The library used for REST API is big and complicated.
You can use a simpler library as the one offered by .NET.

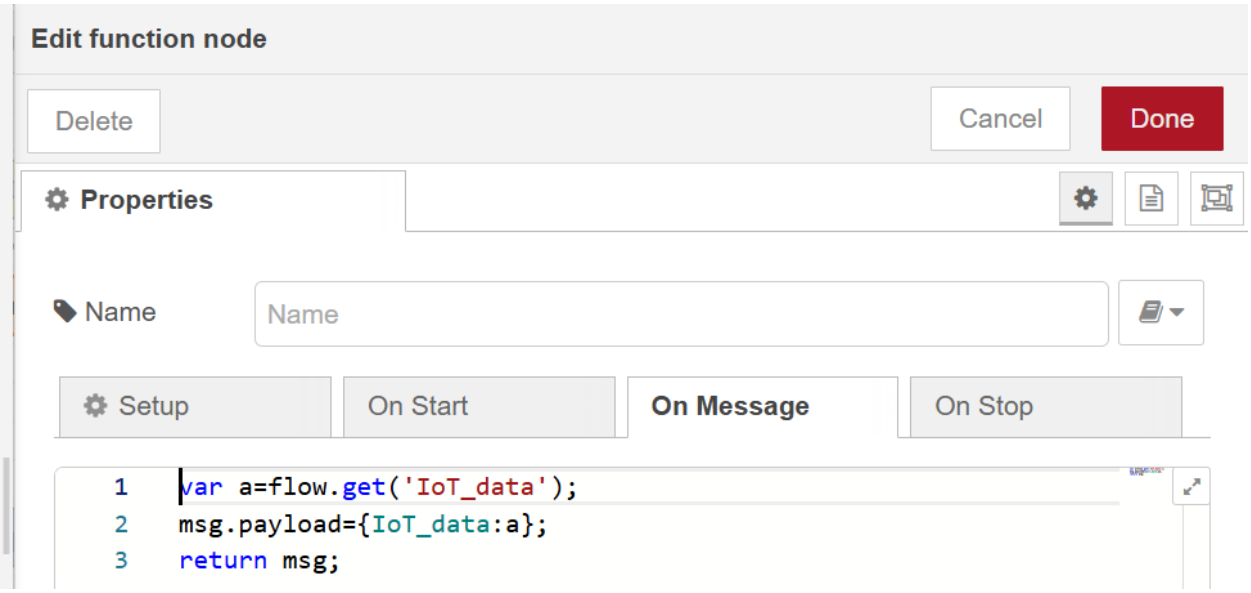
You can download the code from here
https://github.com/xavierflorensa/FTOptix_APIGET_Node_RED.git

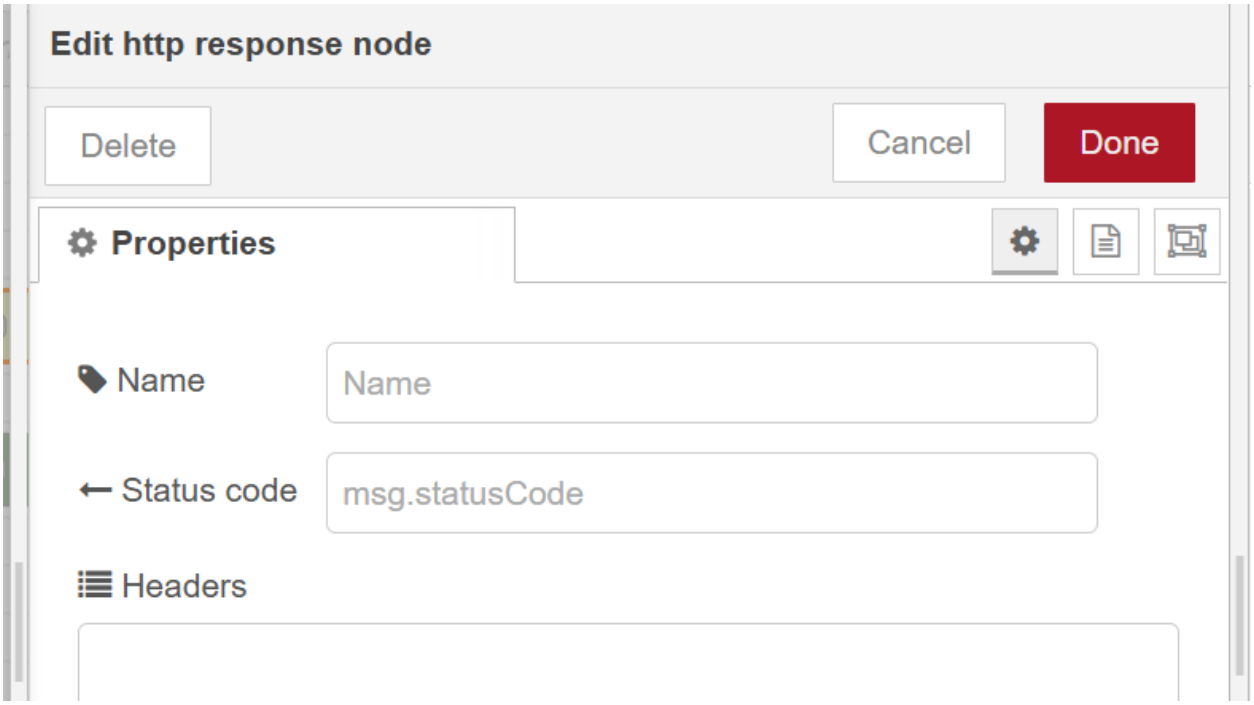
Just use this directive on your Netlogic

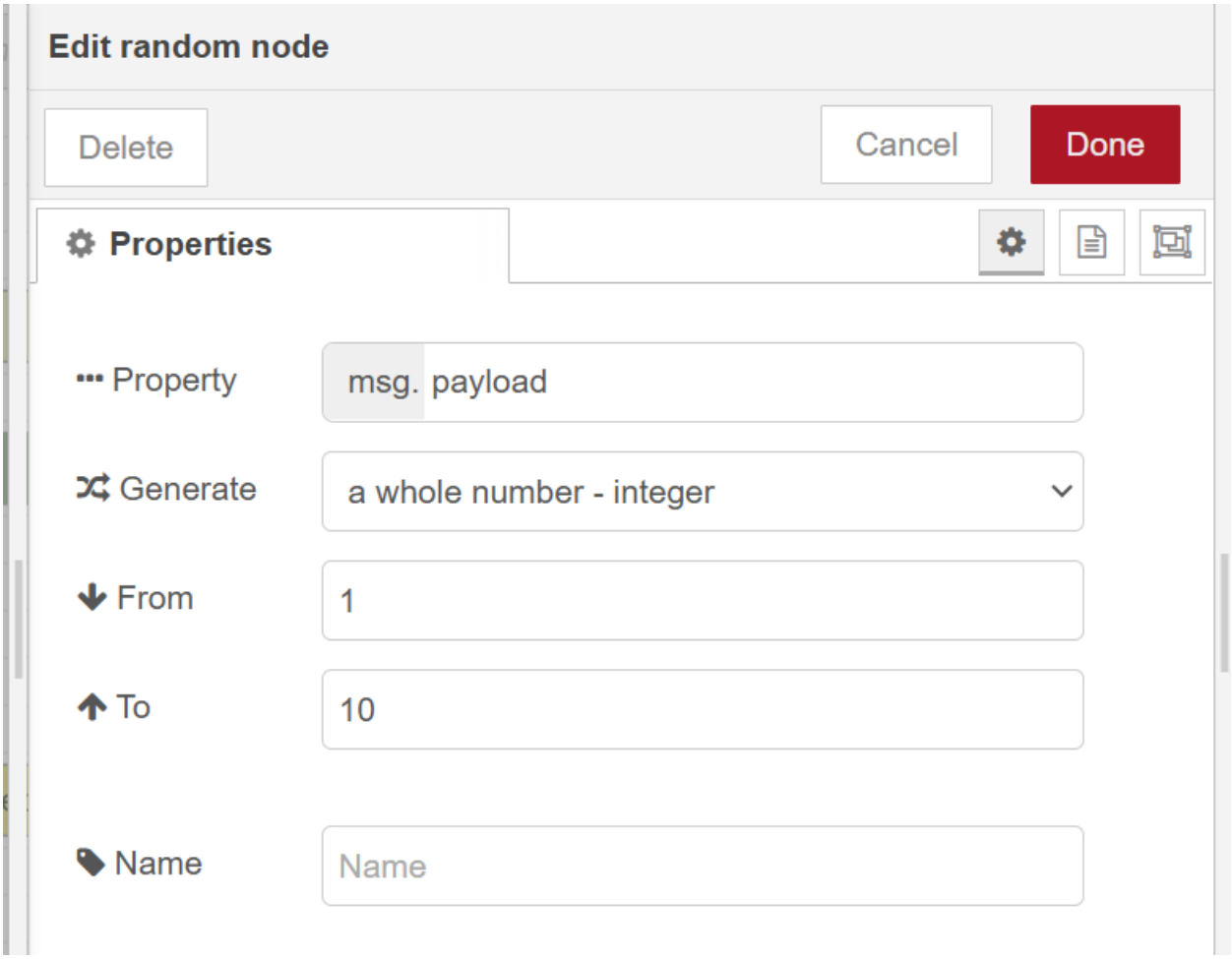
```
using System.Net.Http;
```

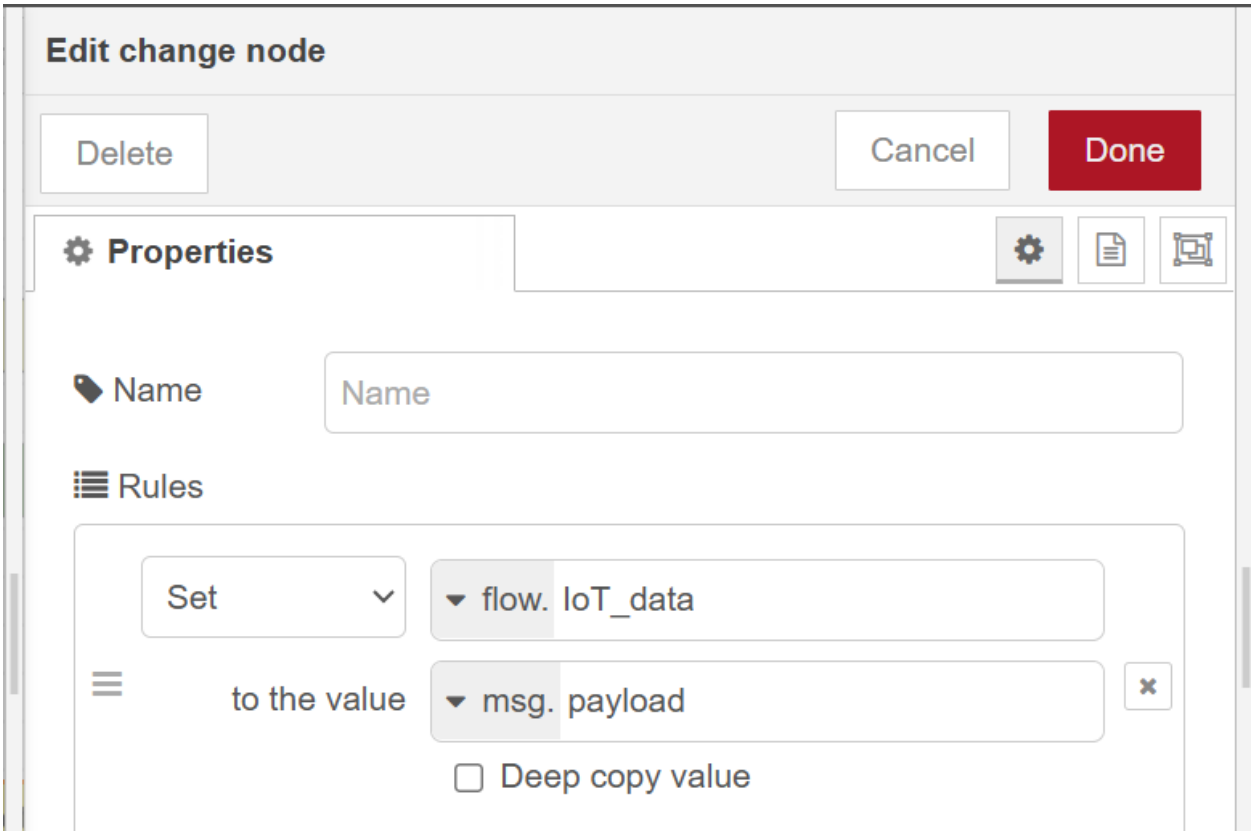
And let's try with the simplest REST API server we have seen in the GET chapter with Node-RED


127.0.0.1:1880/#flow/2fa30c2db957bbdd
Rockwell
Node-RED : node-r...
TTN Login
TTN
Login | Microsoft 365
Node-RED
filter nodes
OPC UA FTLinxC
IBM endpoint
REST API
common
inject
debug
complete
catch
status
link in
link call
link out
[get] /plc
function
http
debug 38
timestamp
random
set flow.IoT_data
Edit http in node
Delete
Cancel
Done
Properties
Method
GET
URL
/plc
Name
Name


Edit function node
Delete
Cancel
Done
Properties
Name
Name
Setup
On Start
On Message
On Stop
var a=flow.get('IoT_data');
msg.payload={IoT_data:a};
return msg;

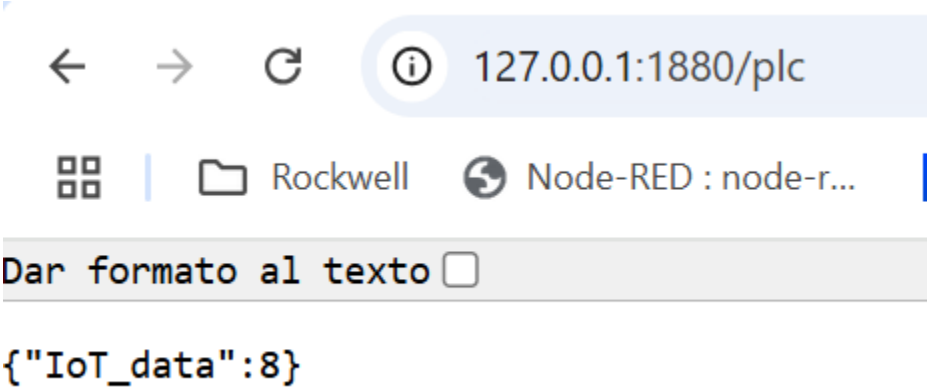

Edit http response node
Delete
Cancel
Done
Properties
Name
Name
Status code
msg.statusCode
Headers


Edit random node
Delete
Cancel
Done
Properties
Property
msg. payload
Generate
a whole number - integer
From
1
To
10
Name
Name

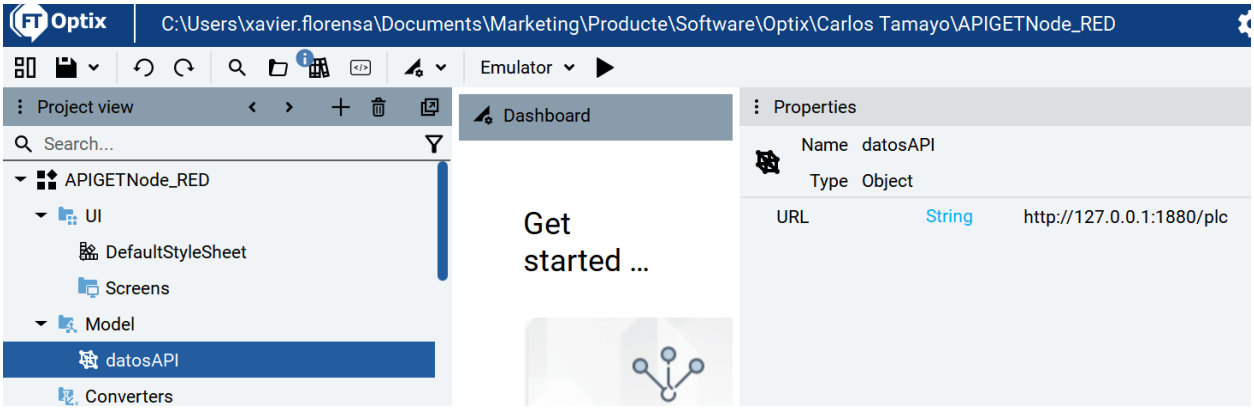


And executing the http request

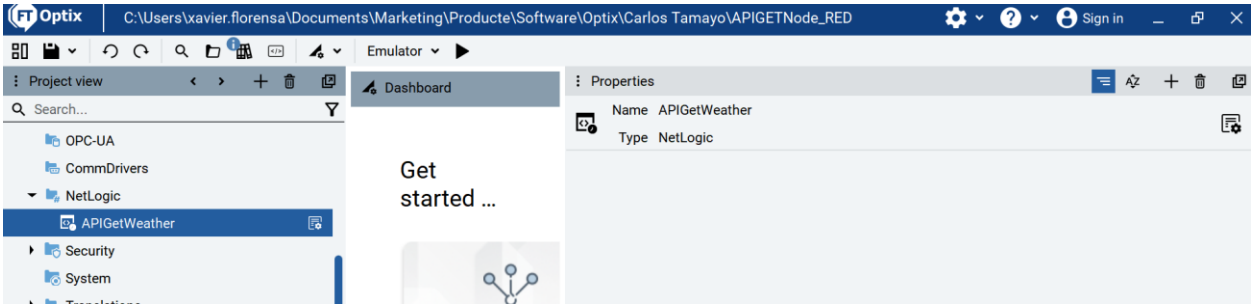
That returns a random number



Let's create this FT Optix application with no graphics



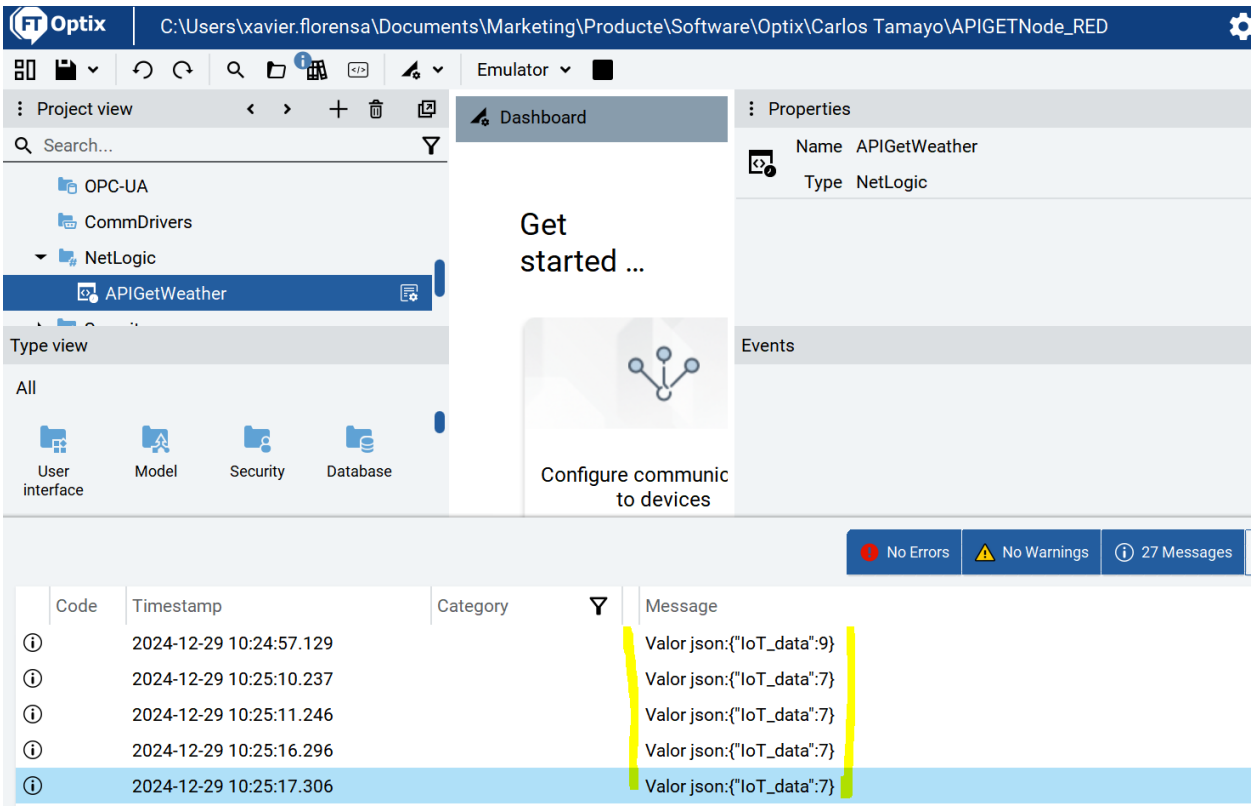
And create this Netlogic



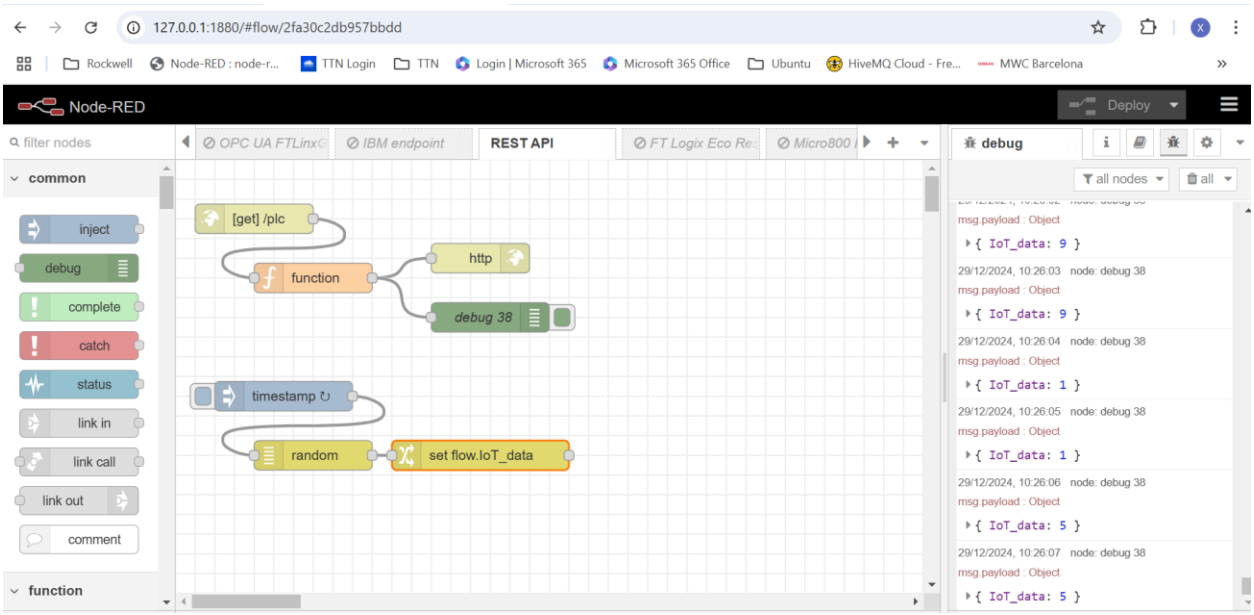

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using System.IO;
using System.Net.Http;
#endregion

public class APIGetWeather : BaseNetLogic
{
    public override void Start()
    {
        myPeriodictask = new PeriodicTask(GetWeather, 1000, LogicObject);
        myPeriodictask.Start();
    }

    public override void Stop()
```

```
    {
        // Insert code to be executed when the user-defined logic is stopped
        myPeriodictask?.Dispose();
    }

    private void GetWeather()
    {
        URL = Project.Current.GetVariable("Model/datosAPI/URL");

        HttpClient httpClient = new();
        var result = httpClient.GetAsync(URL.Value).Result;

        if (result.IsSuccessStatusCode)
        {
            StreamReader reader = new
StreamReader(result.Content.ReadAsStream());
            string json = reader.ReadToEnd();
            Log.Info("Valor json:" + json);
        }

        else
        {
            Log.Info("Failed to retrieve data. Status code: " +
result.StatusCode);
        }
    }

    private PeriodicTask myPeriodictask;
    private IUAVariable URL; //datos API
}
```

Let’s execute the code
This will perform an Http request each second
Do not wait for the runtime window to open. This will not open since we are not using native representation.



You can also see the inputs on Node-red



You can download the code from here
https://github.com/xavierflorensa/FTOptix_APIGET_Node_RED.git

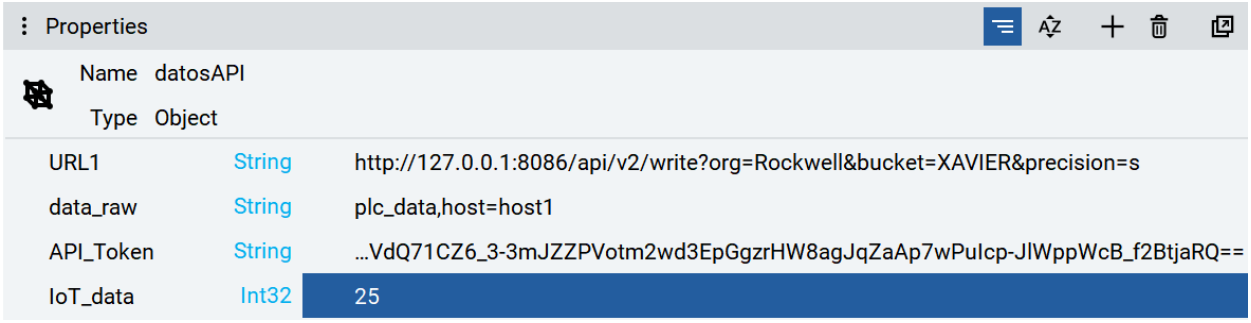
Now let's try to inject to InfluxDB this way

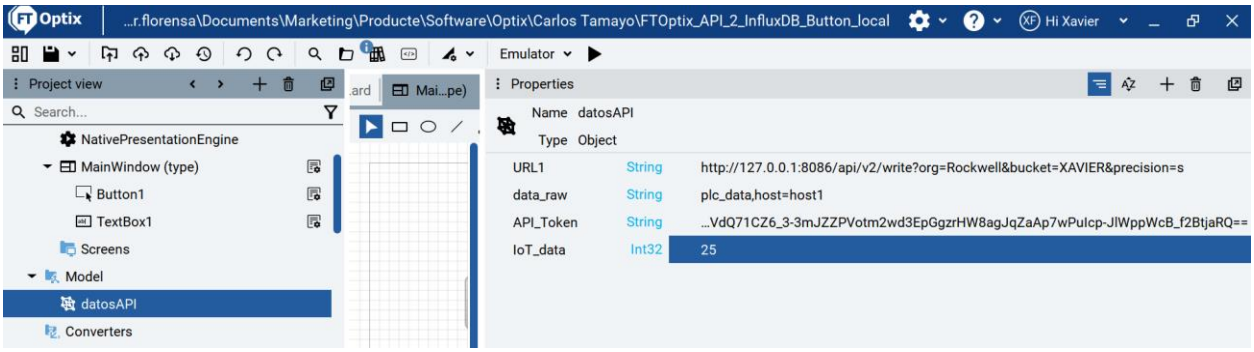
22.1. Simple InfluxDB http POST request

You can find the code here

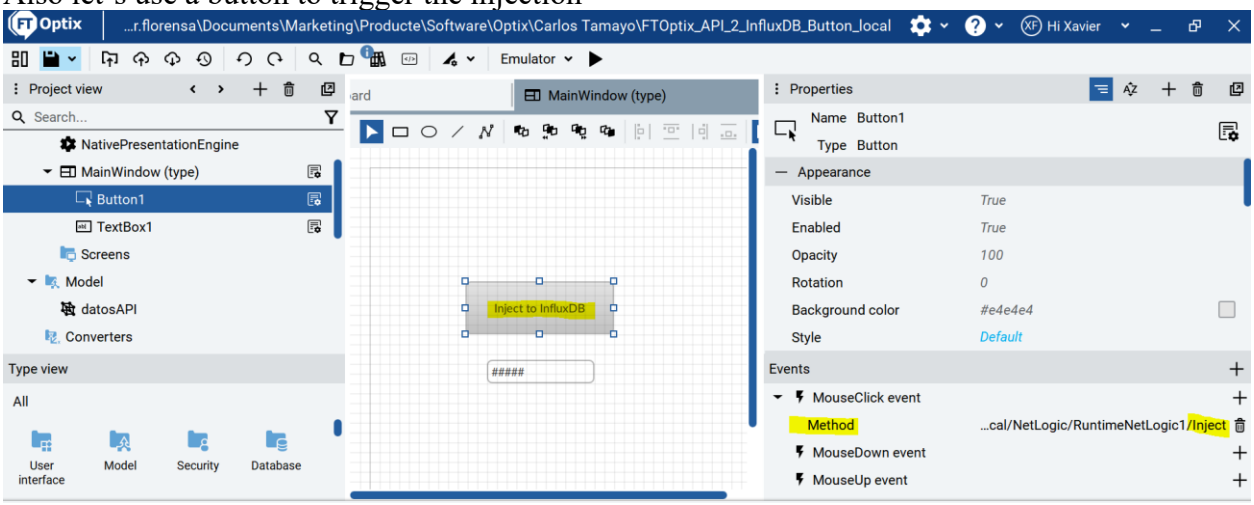
https://github.com/xavierflorensa/FTOptix_API_2_InfluxDB_Button_local.git

Modify the API data this way





Also let's use a button to trigger the injection



And use this code

```
#region Using directives
```

```
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using System.Net.Http;
using System.IO;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

    [ExportMethod]

    public void Inject()
    {
       GetWeather();
       Log.Info("Injecting: ");
    }

     private void GetWeather()
    {
        //URL = Project.Current.GetVariable("Model/datosAPI/URL");
        URL1 = Project.Current.GetVariable("Model/datosAPI/URL1");
        data_raw = Project.Current.GetVariable("Model/datosAPI/data_raw");
        API_Token = Project.Current.GetVariable("Model/datosAPI/API_Token");
        IoT_data = Project.Current.GetVariable("Model/datosAPI/IoT_data");
        HttpClient httpClient = new();

        var result3 = httpClient.DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue("Token",API_Token.Value);
```

```
        var content = new System.Net.Http.StringContent(data_raw.Value+"
IoT_data="+IoT_data.Value, System.Text.Encoding.UTF8);
        //var result = httpClient.GetAsync(URL.Value).Result;
        var result1 = httpClient.PostAsync(URL1.Value,content).Result;

        if (result1.IsSuccessStatusCode)
        {
            StreamReader reader = new
StreamReader(result1.Content.ReadAsStream());
            string json = reader.ReadToEnd();
            Log.Info("Valor json:" + json);
        }

        else
        {
            Log.Info("Failed to retrieve data. Status code: " +
result1.StatusCode);
        }

    }
    private IUAVariable URL1; //datos API
    private IUAVariable API_Token; //datos API

    private IUAVariable data_raw; //datos API
    private IUAVariable IoT_data; //datos API }
}
```

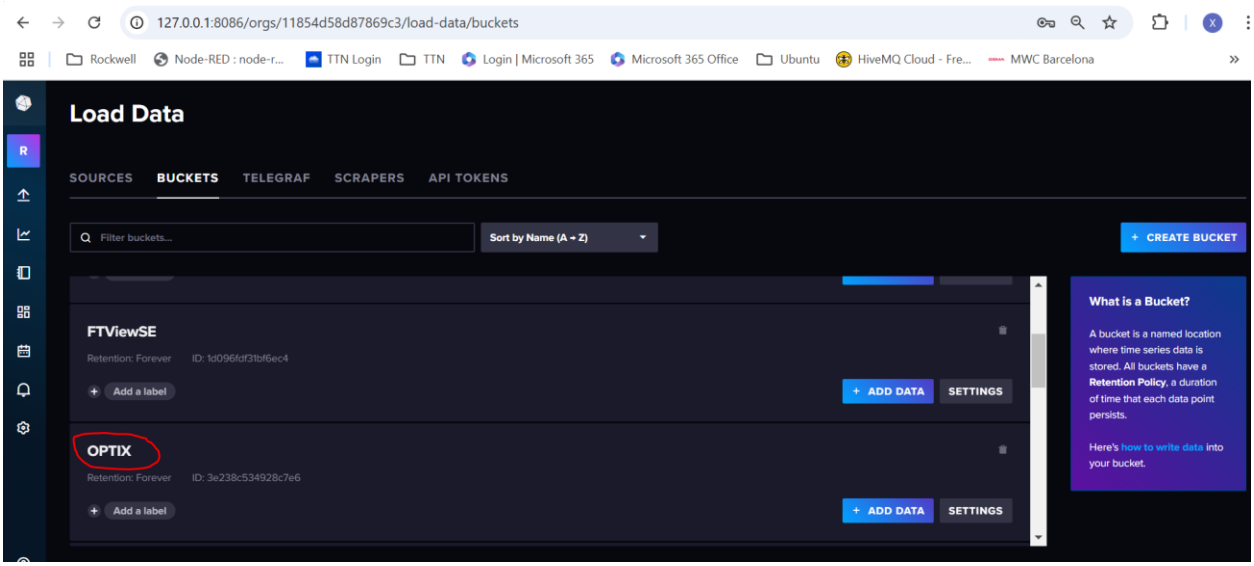
You can find the code here

https://github.com/xavierflorensa/FTOptix_API_2_InfluxDB_Button_local.git

23. Local External InfluxDB in FTOptix v1.4

Starting from FTOptix v1.4 you have already external InfluxDB injection embedded.
Starting from FTOptix v1.5 you can also use an embedded InfluxDB database (so you do not need to install it)

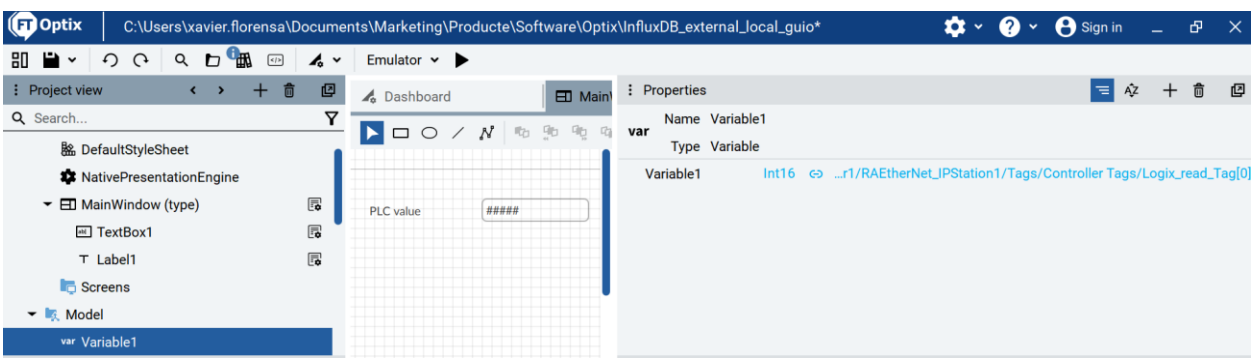
Let's try an external database.
Enter in your InfluxDB and create a Bucket (OPTIX for instance)



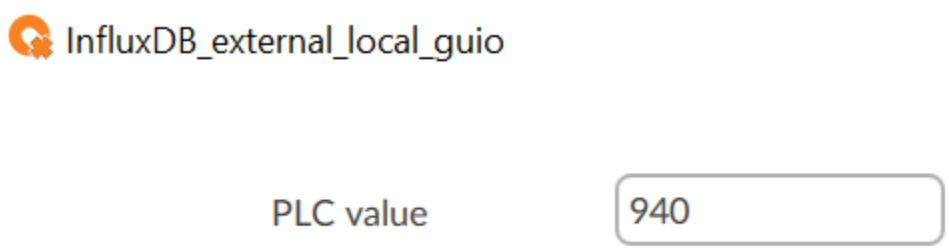
Create a new token and save it

Note the URL were you will find the organization

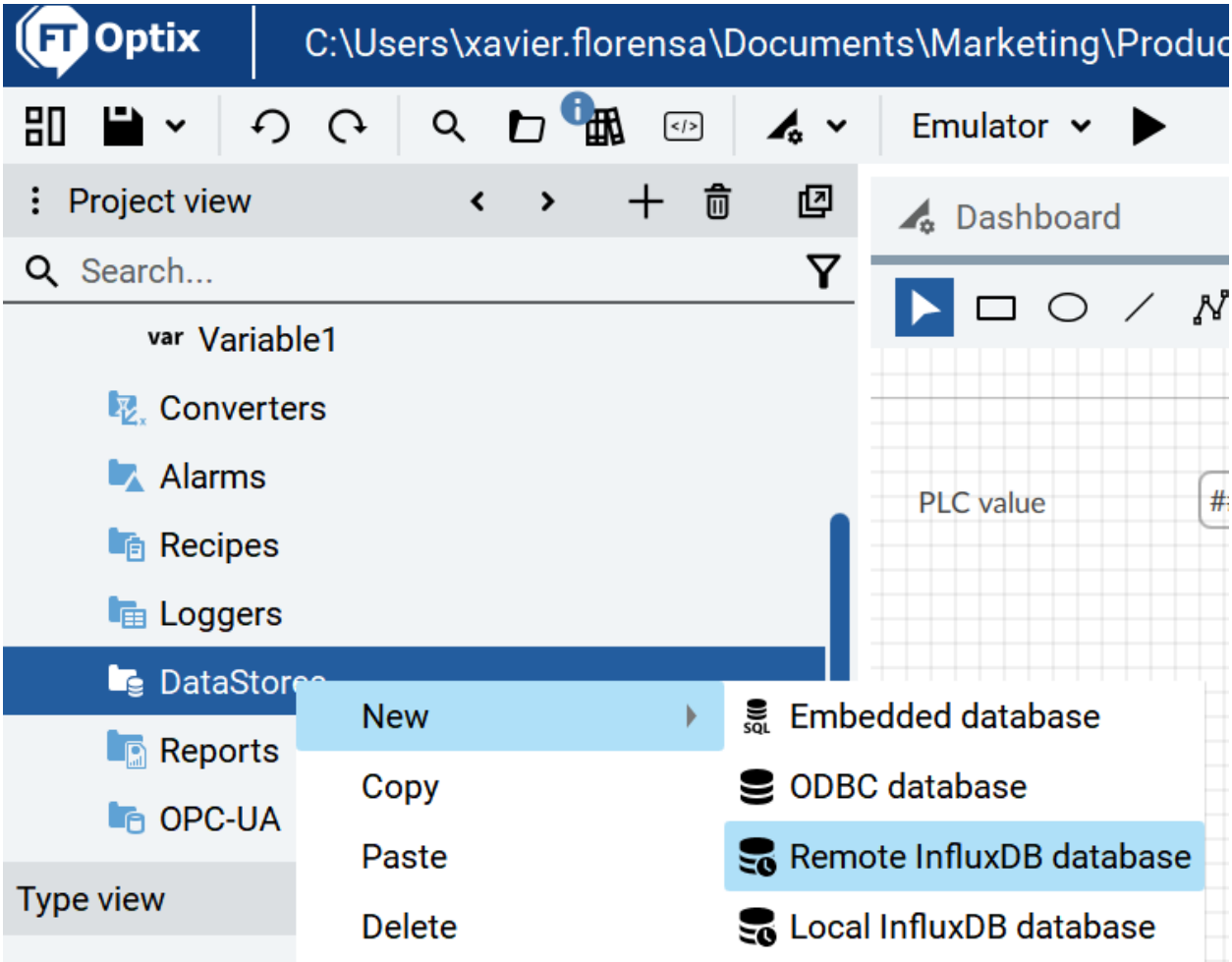
Create an Optix application with some live data, for instance a PLC Tag



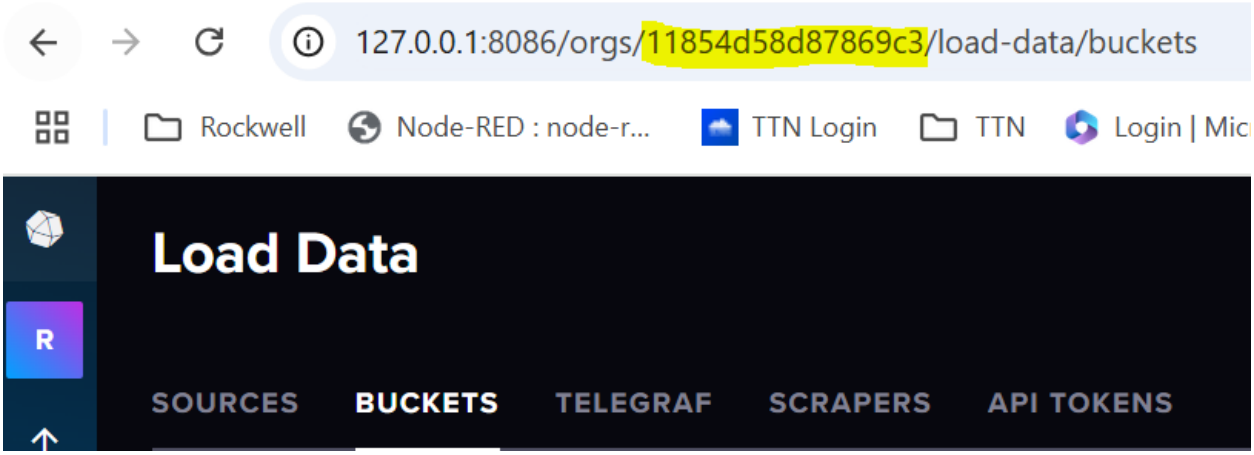
Let's test the Tag is changing live



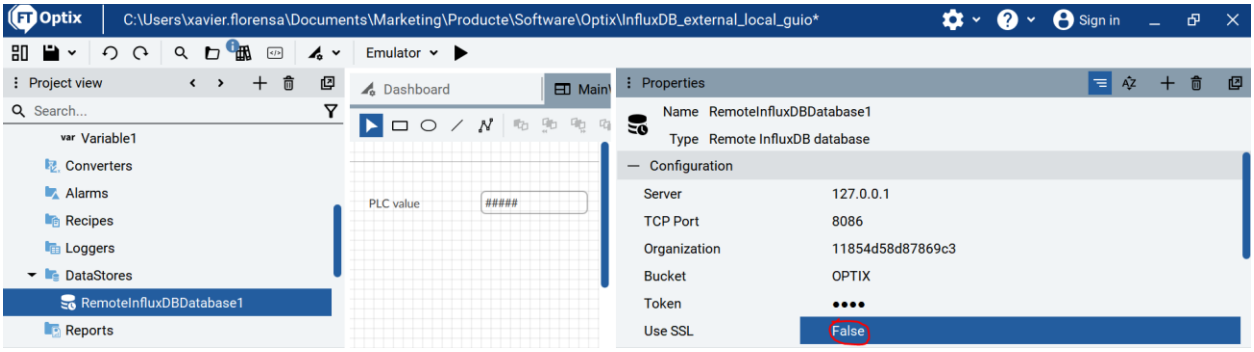
Now let's create the database (Do not create any table or measurement)



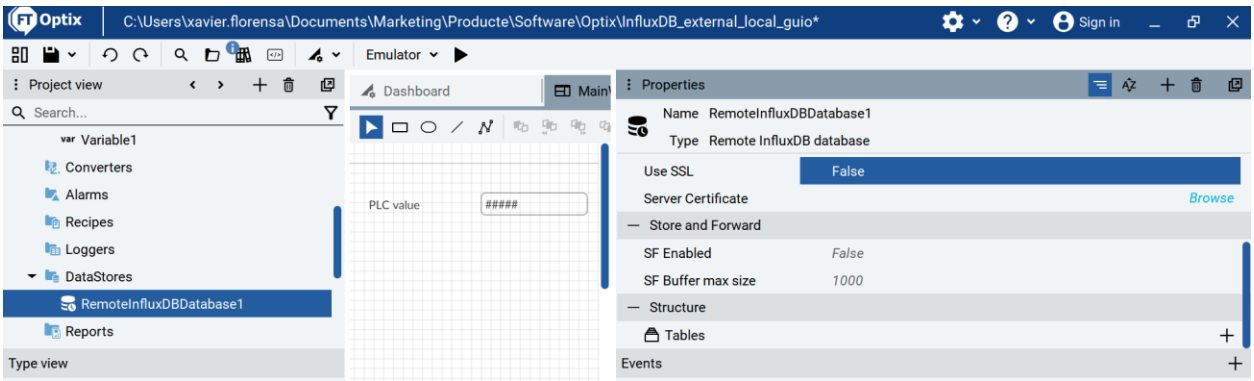
Use this organization you will see on the URL



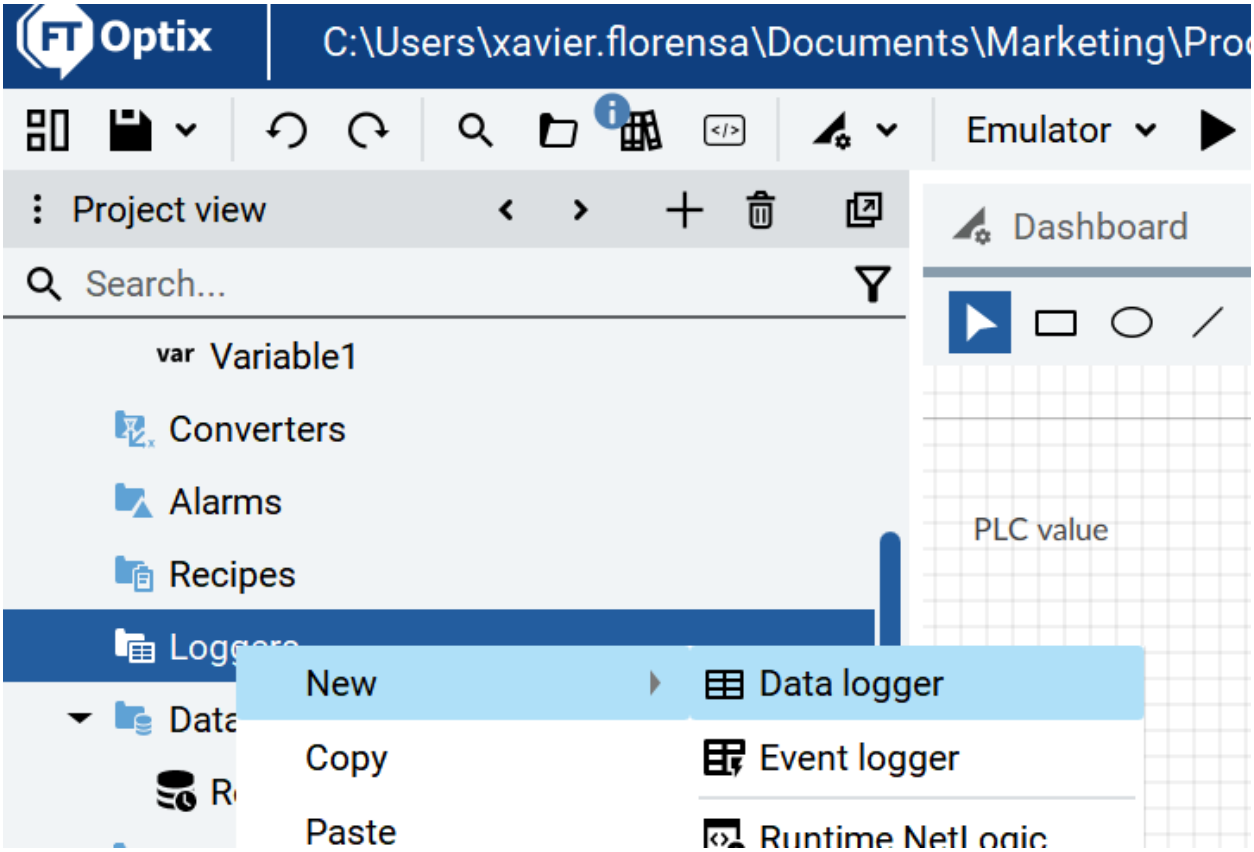
We will not use SSL this time



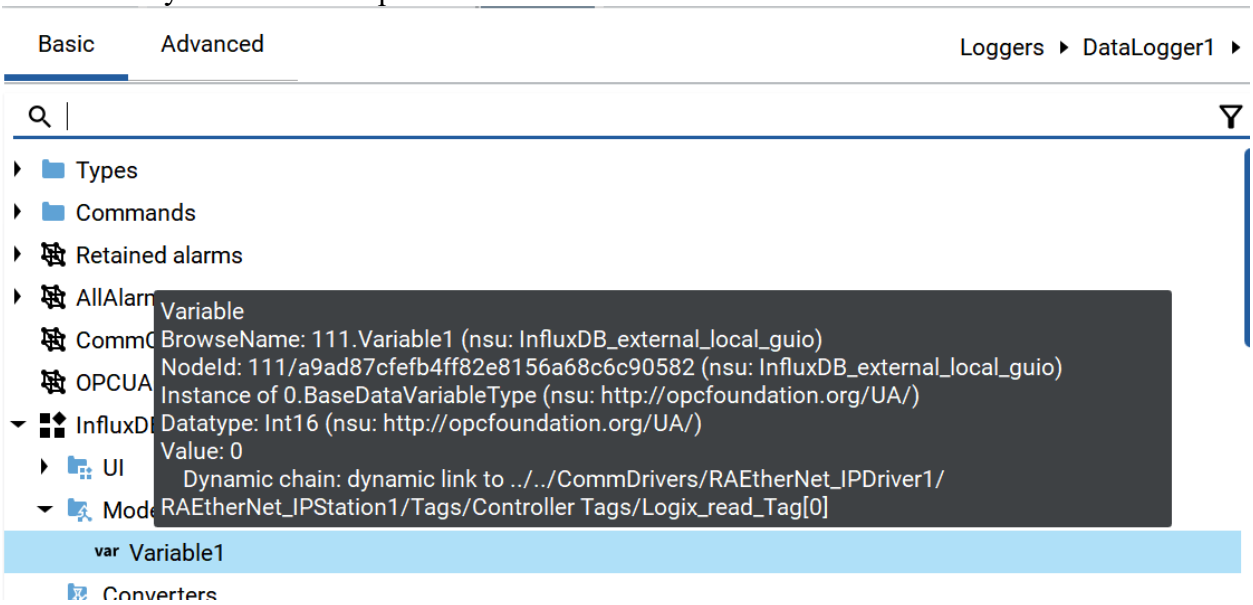
Take a look at Database structure (do not change anything here) it will be created from Datalogger section for you



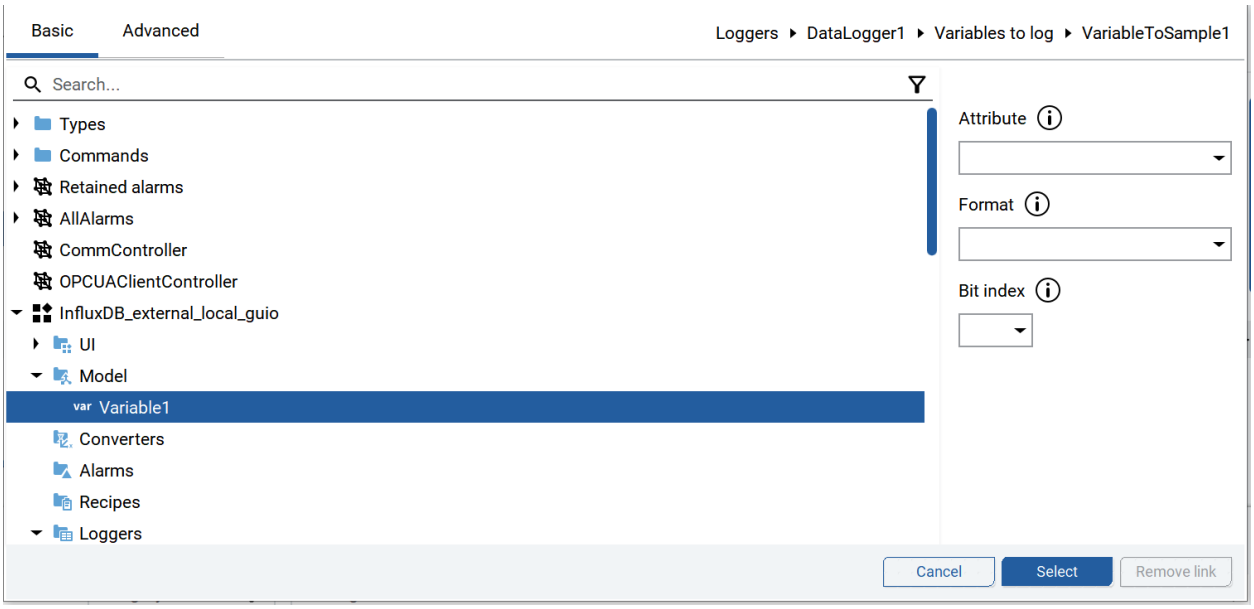
Now create a Datalogger



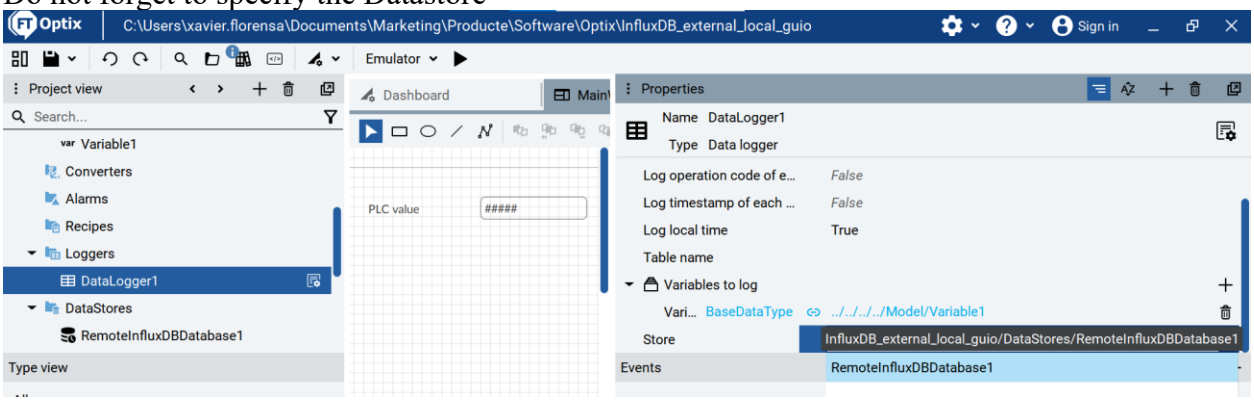
Click + on variable to sample
Now add a dynamic link and point to the desired variable

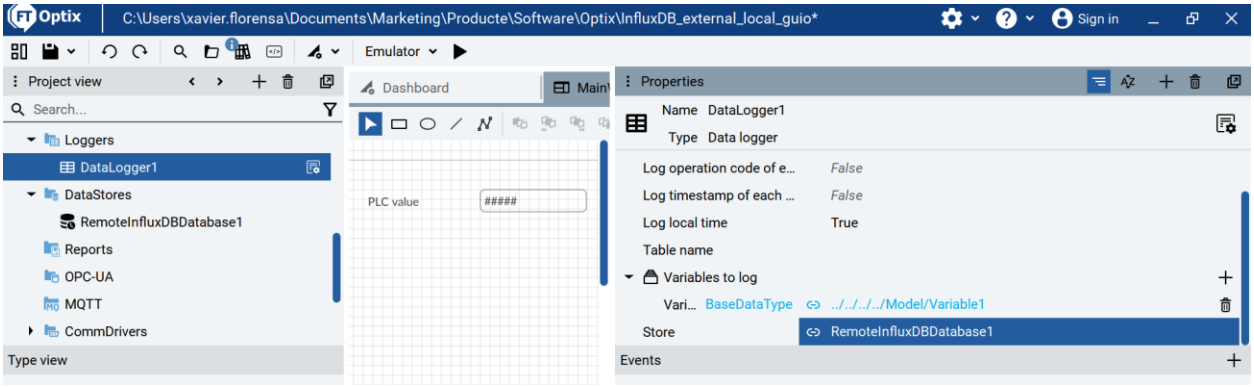


Select it

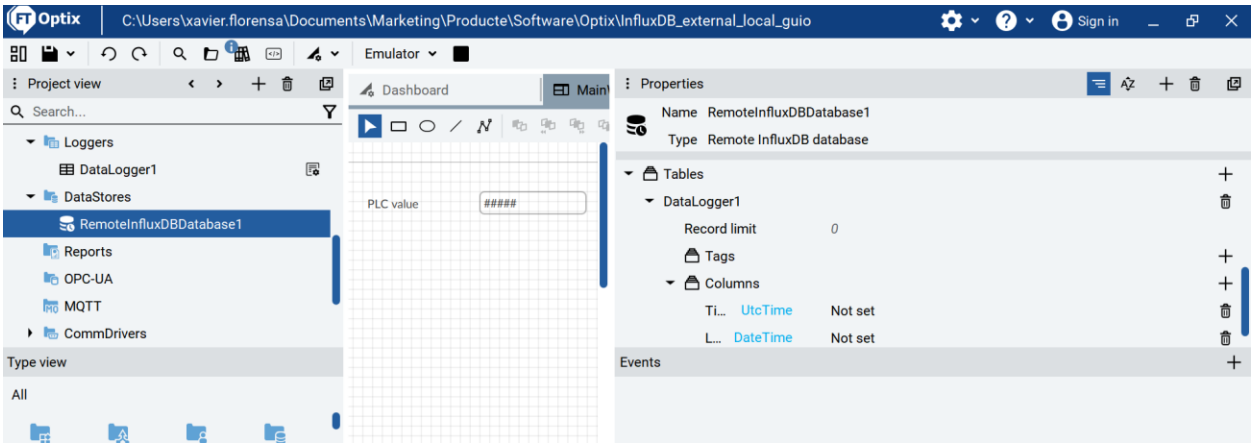


Do not forget to specify the Datastore

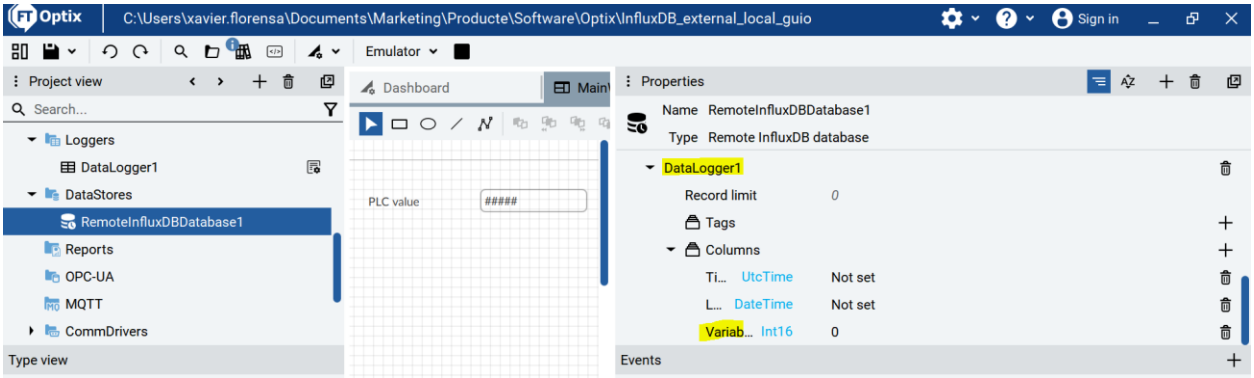




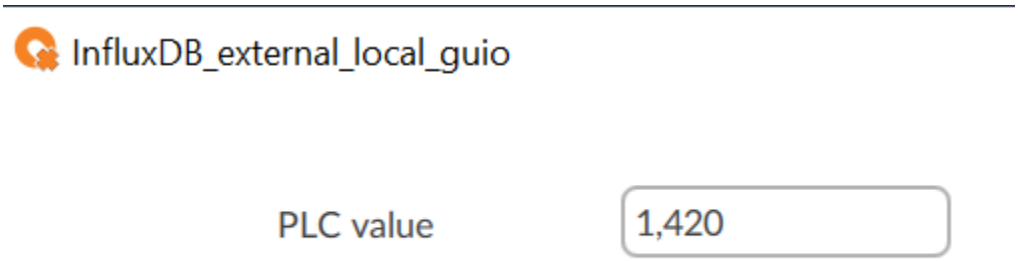
Now take a look at your Datastore configuration. The Data strructure has been created for you.
A new table (measurement) called Datalogger has been add for you

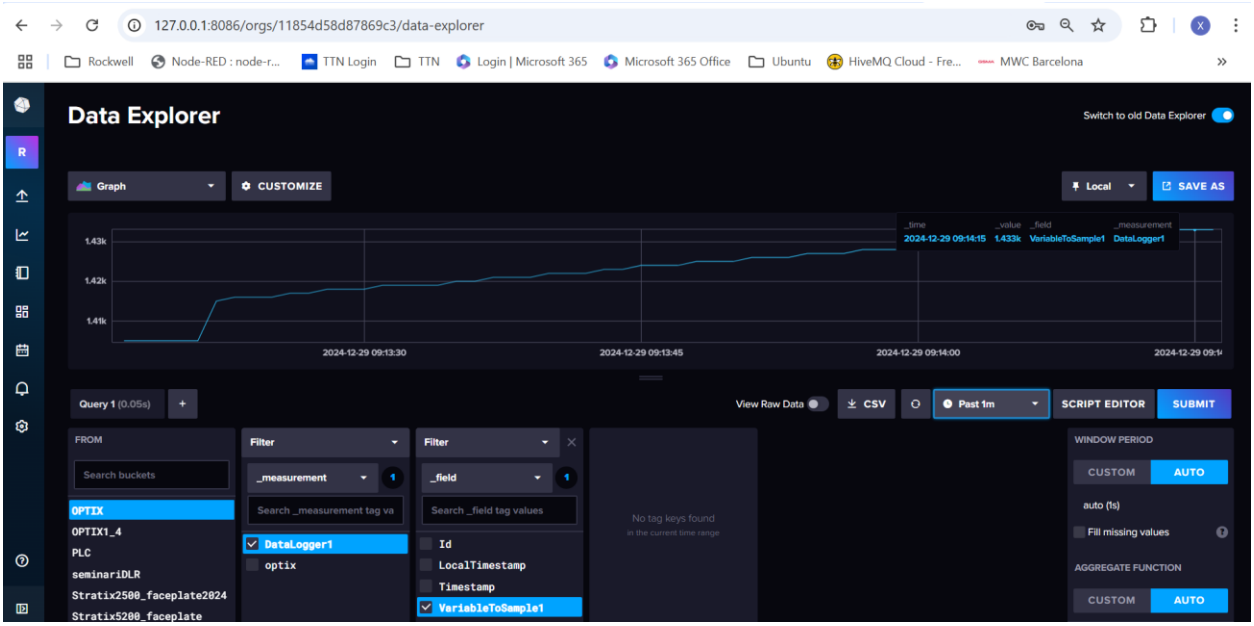


Together with variable to sample column

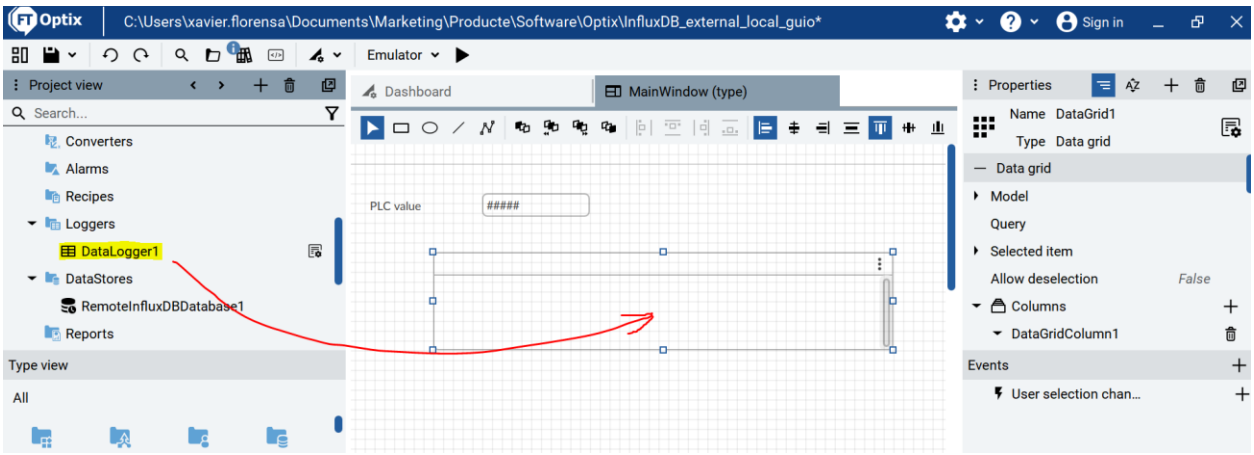


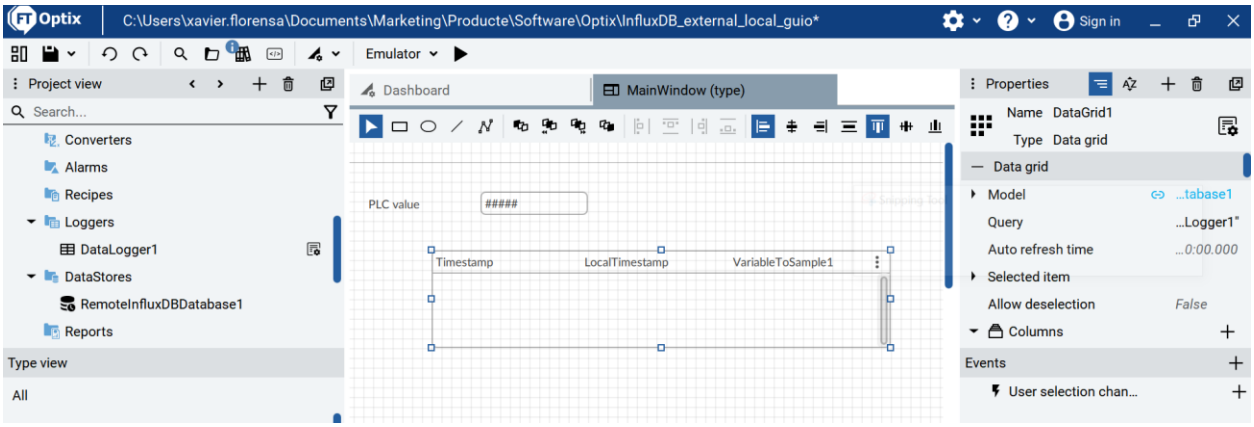
Just test the application now



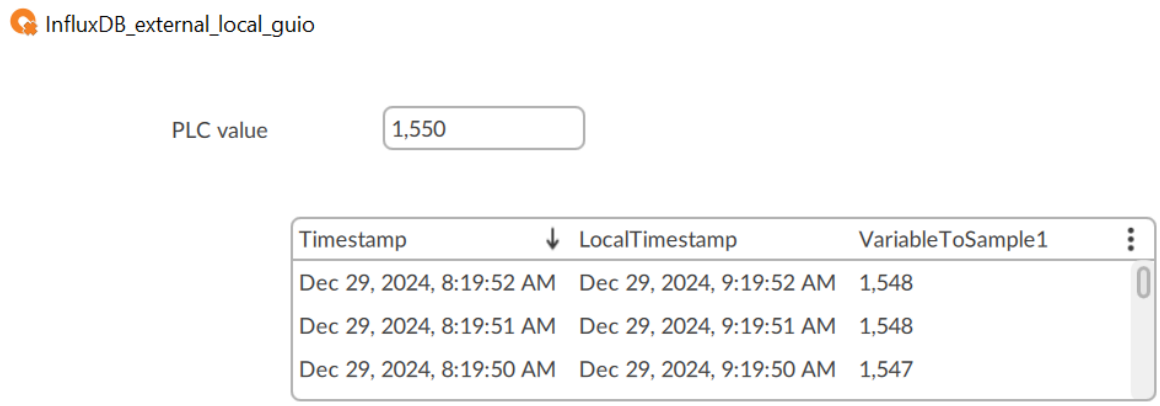


You can also insert a Data grid and drag and drop your dalalogger





Test it



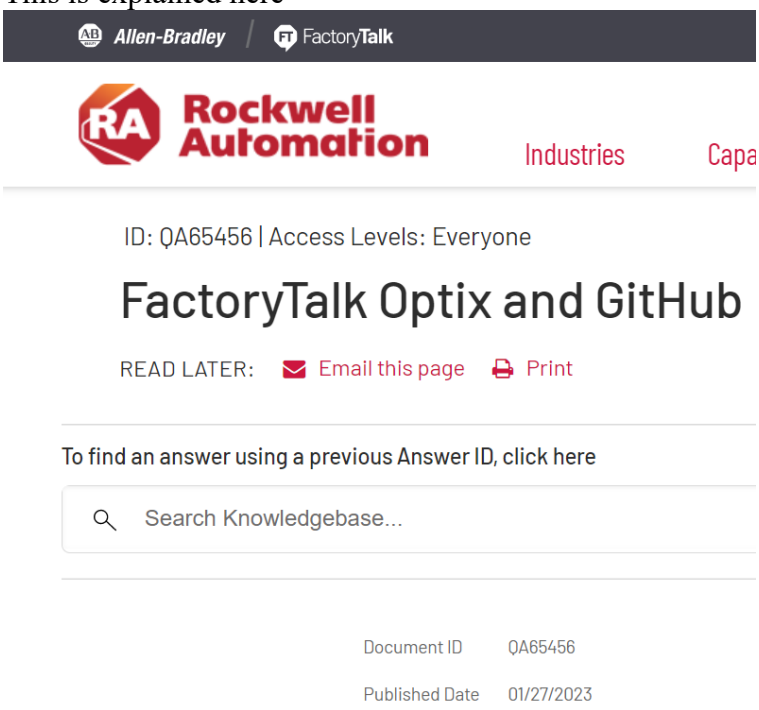
24. Cloud External InfluxDB in FTOptix v1.4

Pending to solve this issue

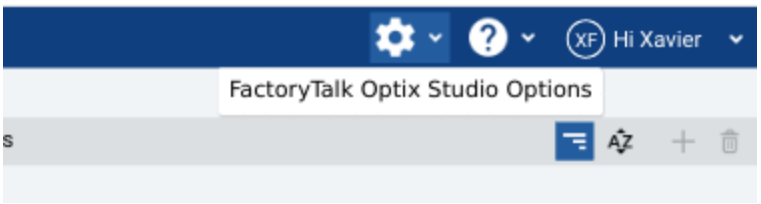
25. Using Github to store and share your repositories on the cloud

Around 100 million developers across the globe use GitHub. The majority of them are based in the US, India, and China. In 2022, there were over 413 million open-source contributions on the platform. Over 90 percent of Fortune 100 companies use GitHub.2 d'oct. 2023

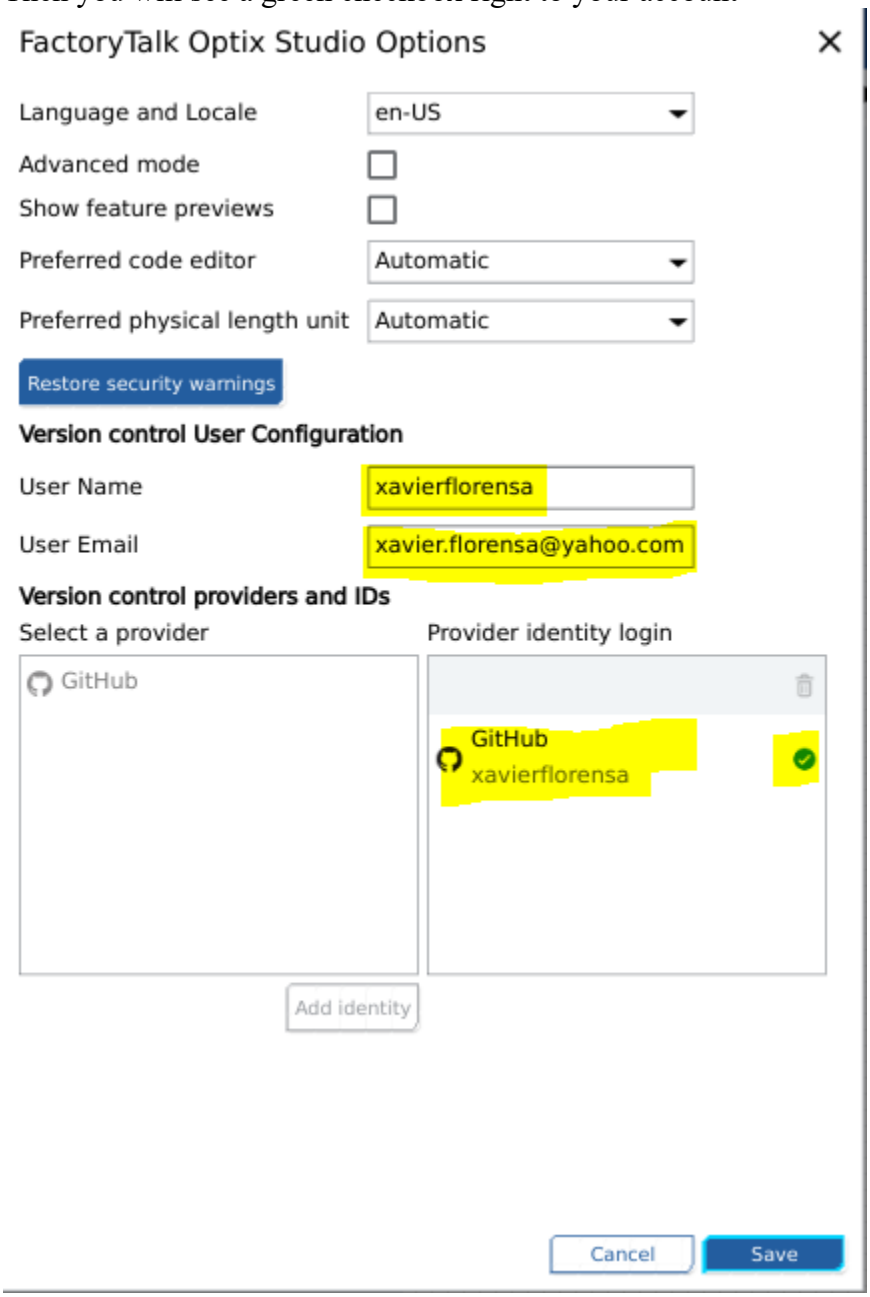
You need the Professional edition of FactoryTalk Optix and a Pro license
The Pro license runs only on the cloud
This is explained here



https://rockwellautomation.custhelp.com/app/answers/answer_view/a_id/1138150/loc/en_US#__highlight
First of all modify your Optix Pro configuration on the toothweel



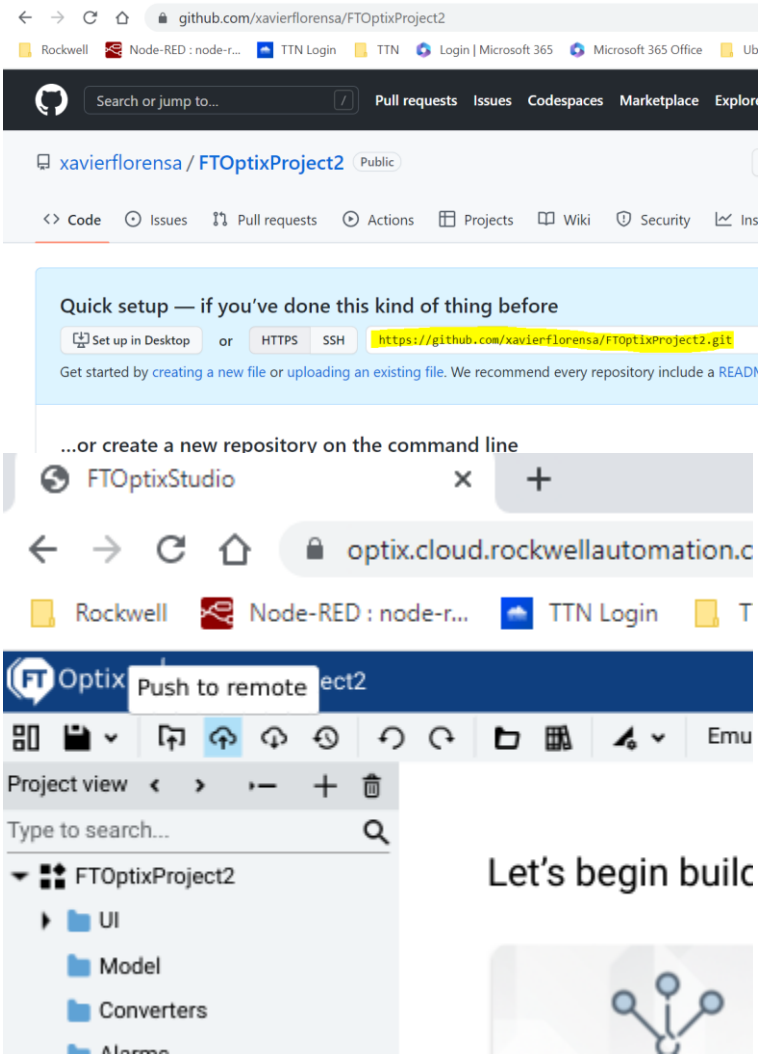
Use your github user name and the associated email
You should authorize ASEM to access your github account when prompted
Then you will see a green checkbox right to your account

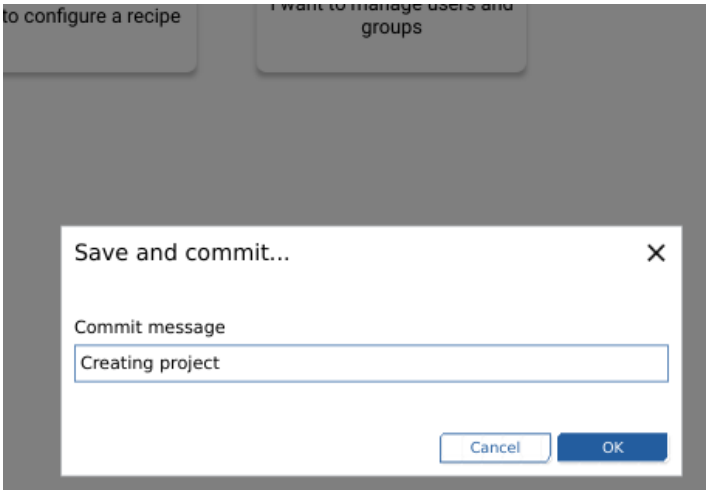


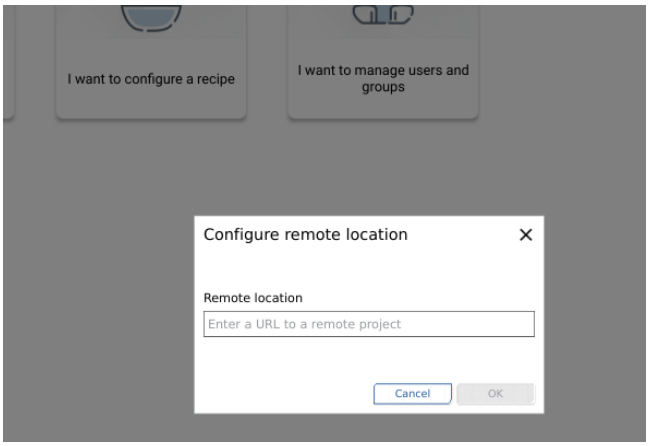
Click on save.
Create a project with a given name on your local instance of FT Optix

Check version control if you want that project to be committed to Github.
Go to Github
Create a new project on Github with the same name you use on Optix on Github
That's all.

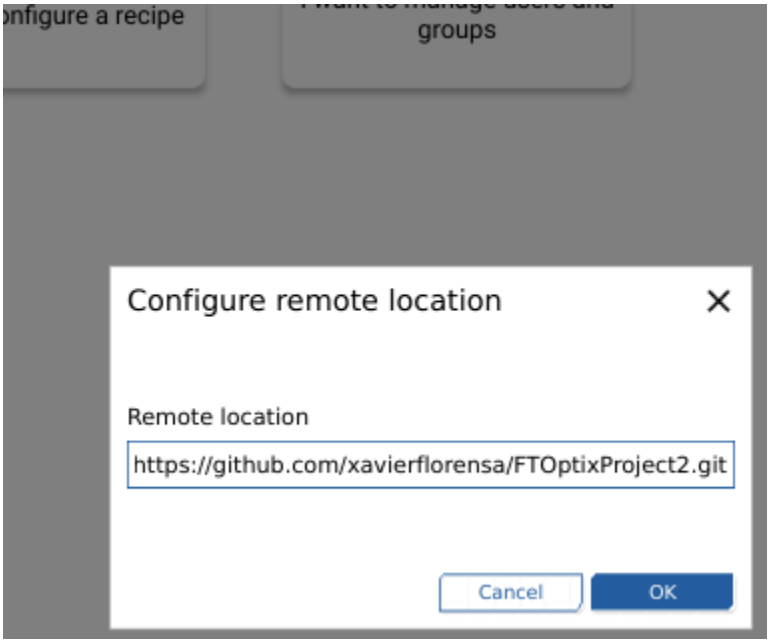
When you save and commit you will be prompted to give a comment and a destination url.
The destination url is this one



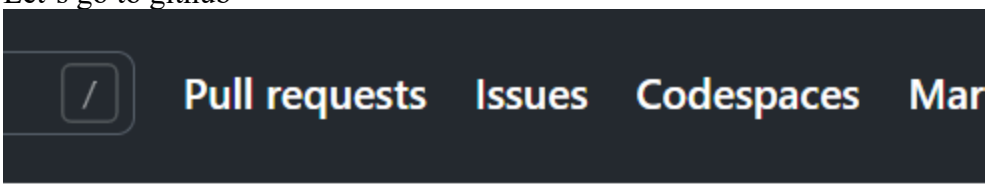


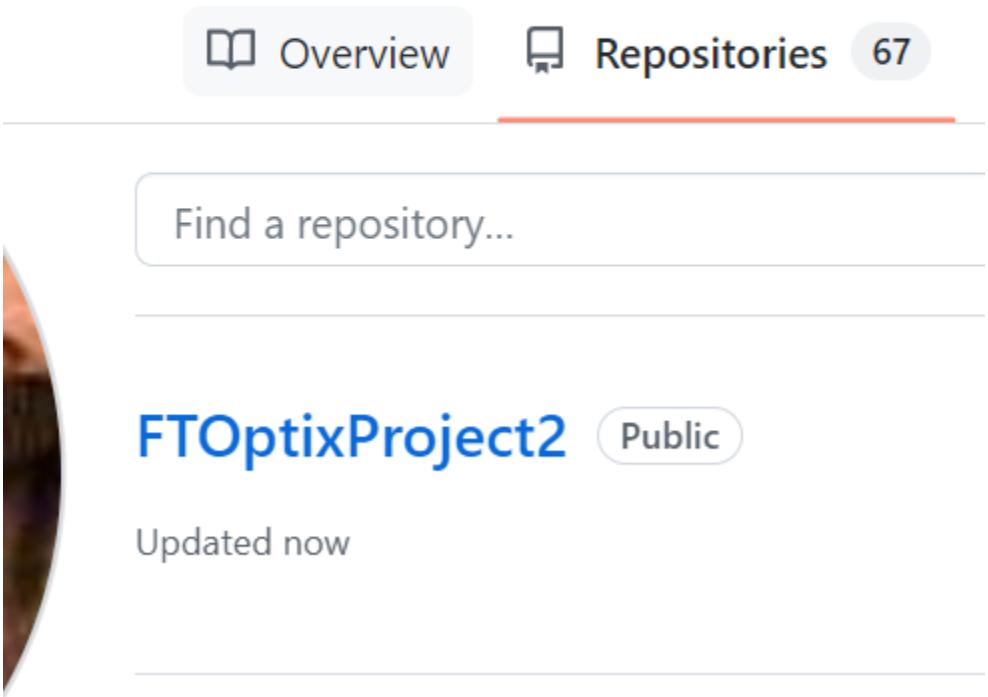


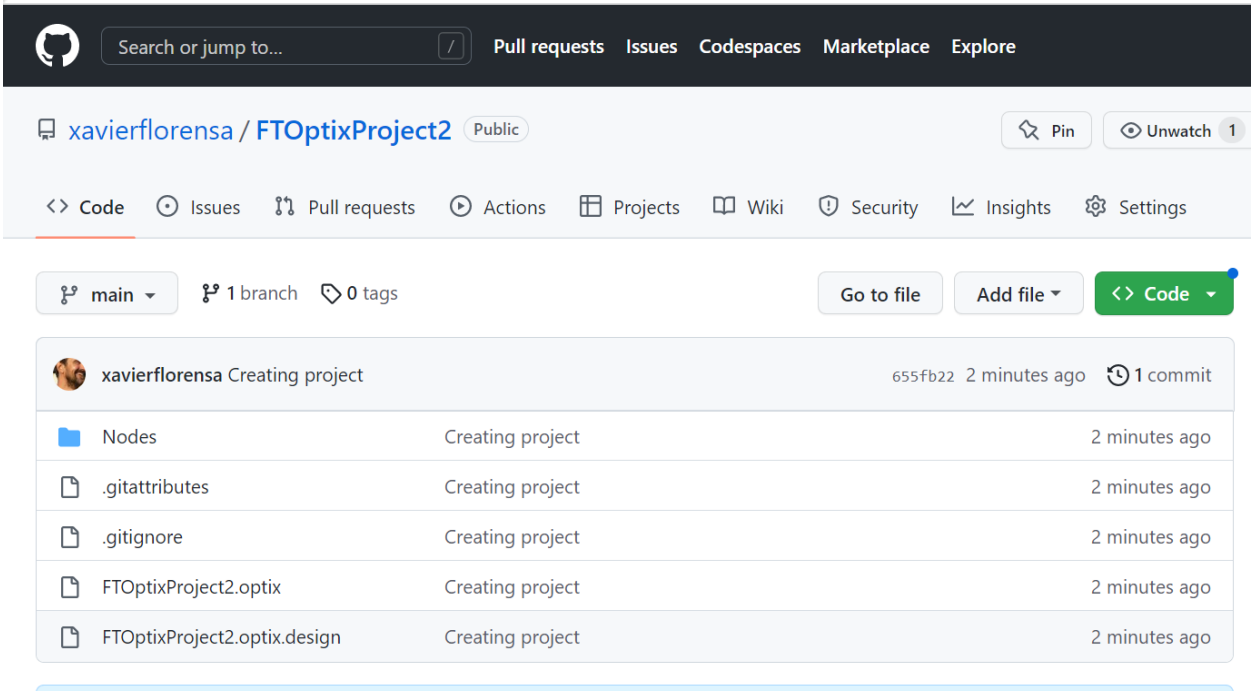
https://github.com/xavierflorensa/FTOptixProject2.git

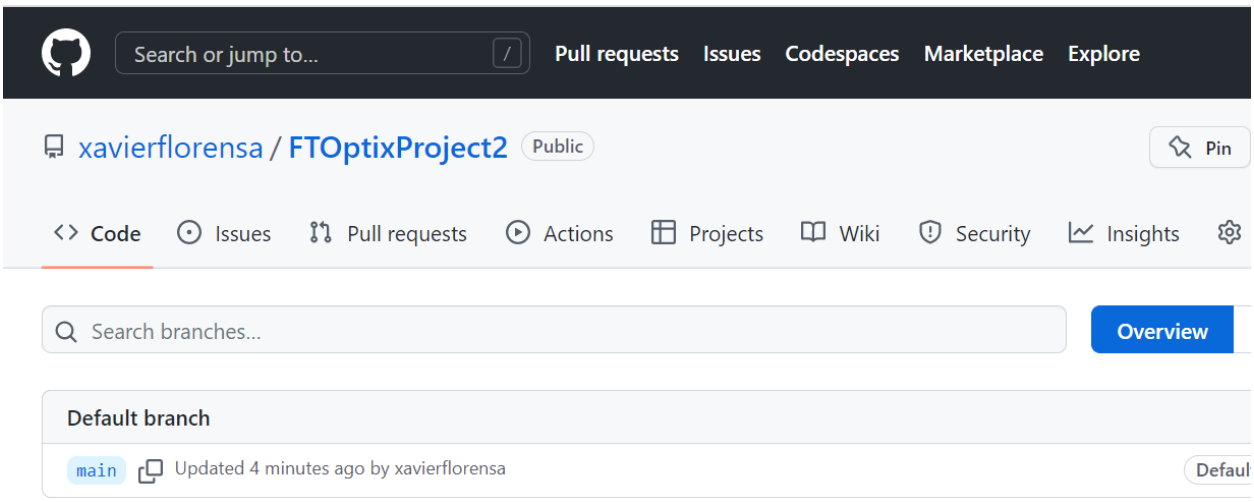


Let’s go to github

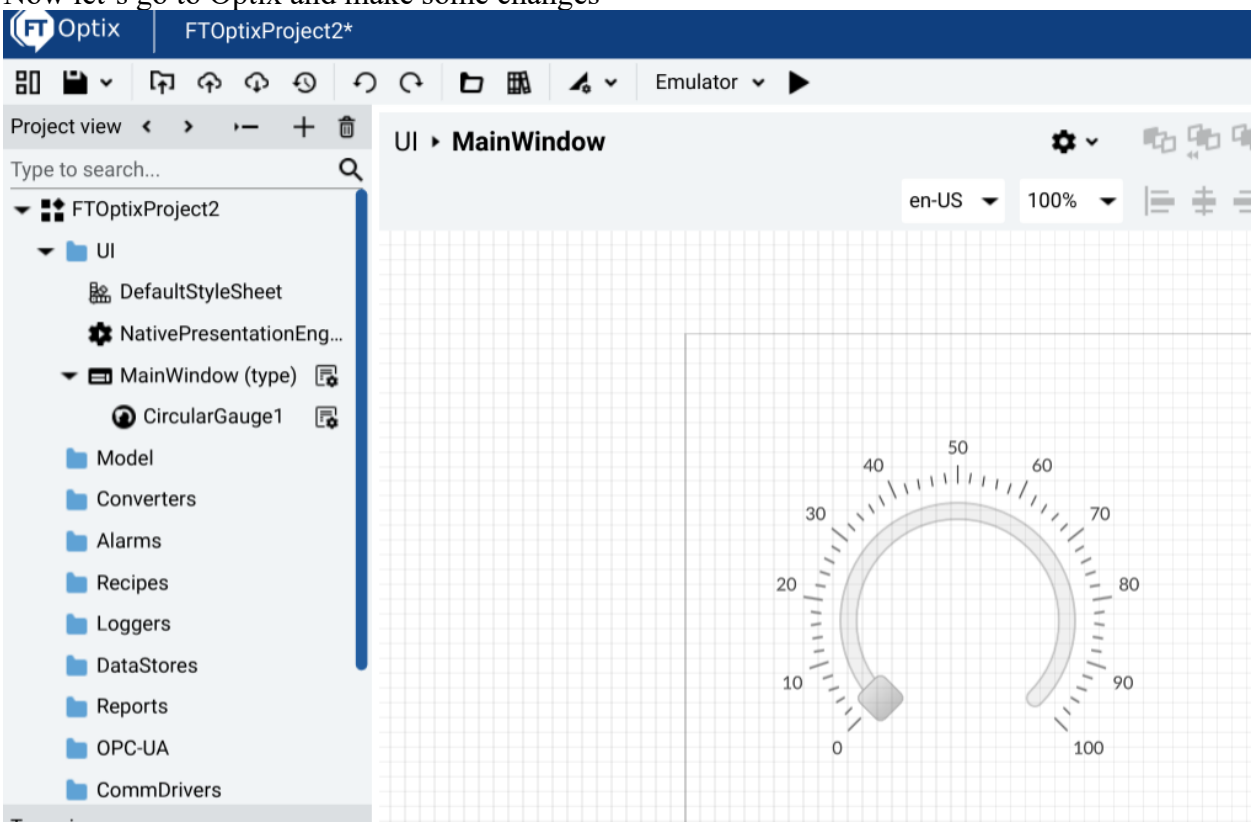




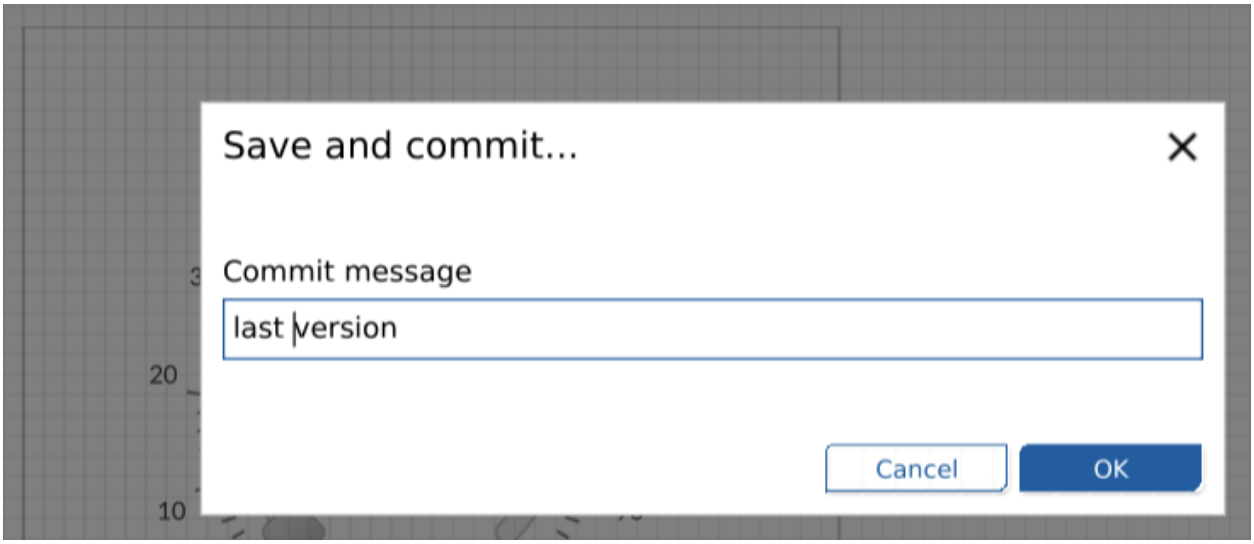




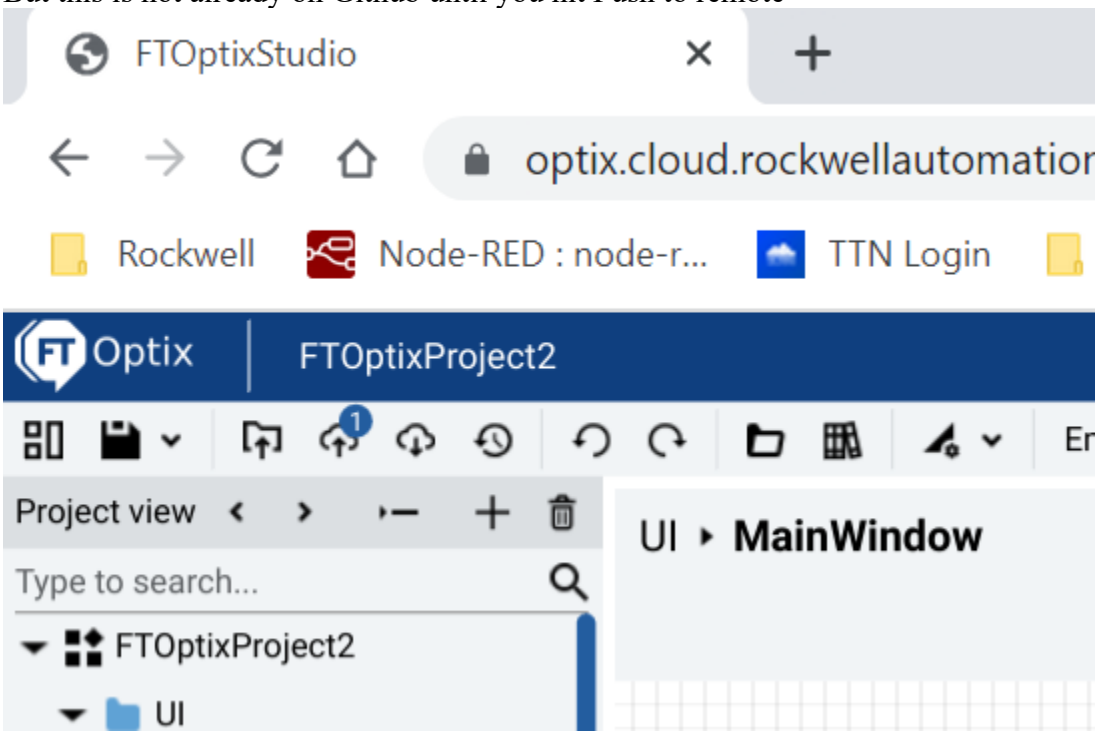
Now let's go to Optix and make some changes



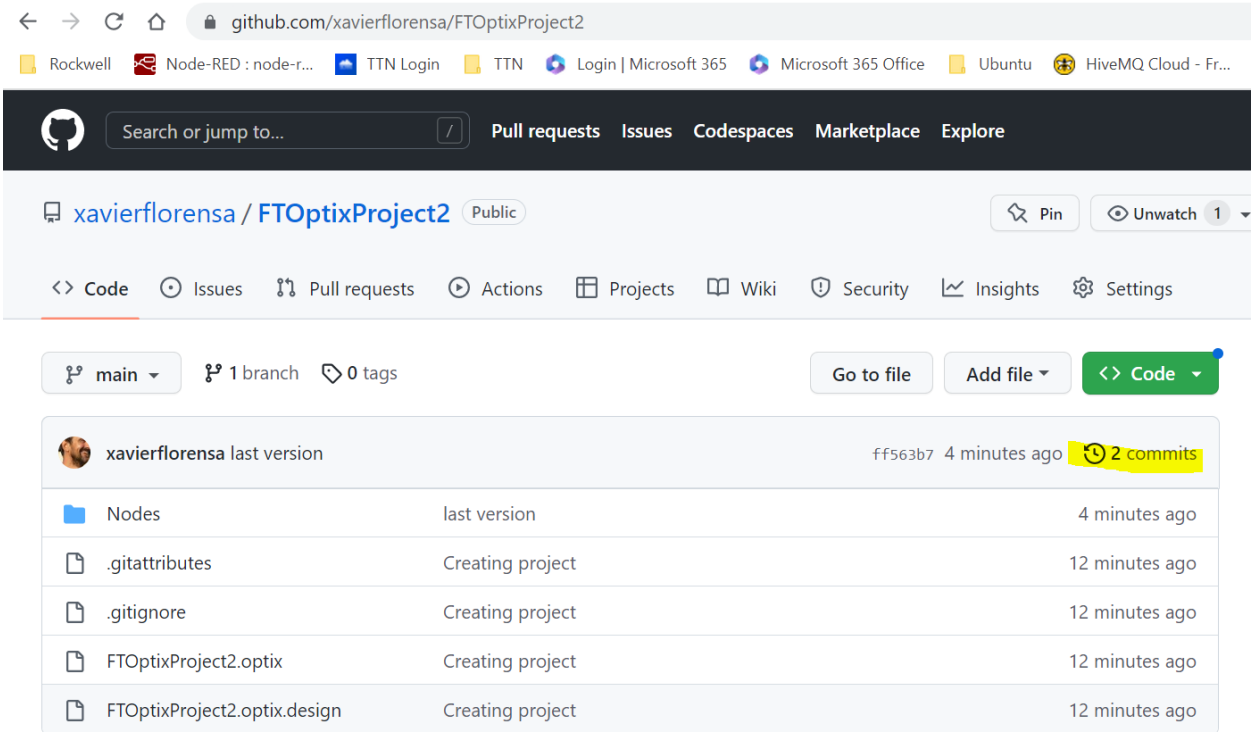
Now save and commit



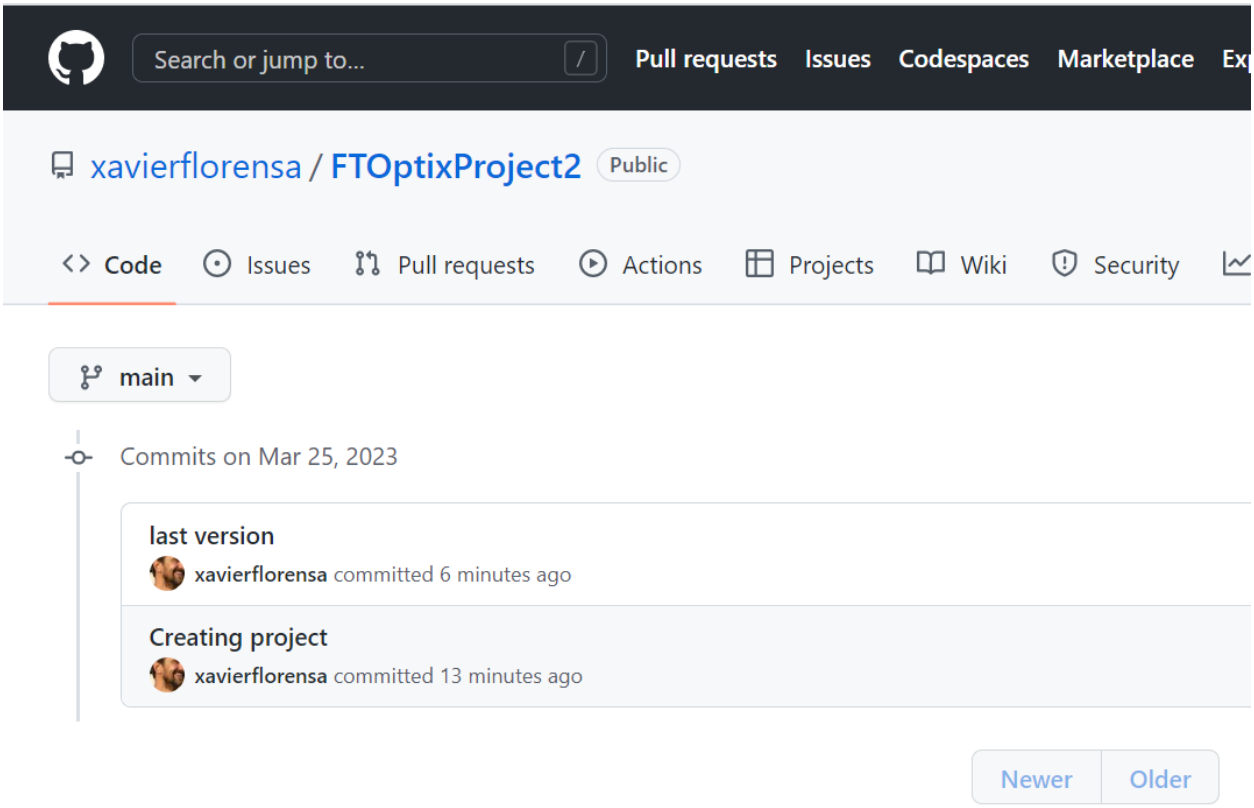
But this is not already on Github until you hit Push to remote



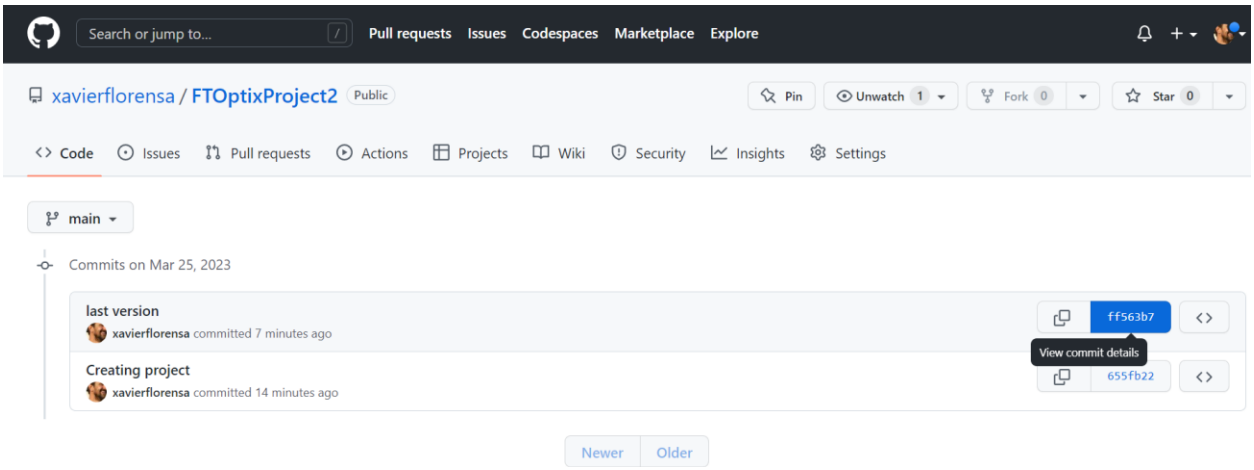
Now we have two commits. If you click on 2 commits:



You will see this



That's all, so you have a record on all the changes and who made it
You can view commit details



And you can see additional comments

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

xavierflorensa / FTOptixProject2 Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

last version

main

xavierflorensa committed 8 minutes ago

Showing 1 changed file with 37 additions and 0 deletions.

37 Nodes/UI/UI.yaml

```
@@ -82,3 +82,40 @@ Children:
82  82      DataType: Size
83  83      ModellingRule: Optional
84  84      Value: 400.0
    85  +   - Name: CircularGauge1
    86  +     Type: CircularGauge
    87  +     Children:
    88  +     - Name: Value
    89  +       Type: BaseDataVariableType
    90  +       DataType: Float
    91  +       Value: 0.0
    92  +     - Name: MinValue
    93  +       Type: BaseDataVariableType
    94  +       DataType: Float
    95  +       Value: 0.0
    96  +     - Name: MaxValue
    97  +       Type: BaseDataVariableType
    98  +       DataType: Float
    99  +       Value: 100.0
    100 +     - Name: WarningZones
    101 +       Type: BaseObjectType
    102 +     - Name: Width
    103 +       Type: BaseVariableType
    104 +       DataType: Size
    105 +       ModellingRule: Optional
    106 +       Value: 210.0
    107 +     - Name: Height
    108 +       Type: BaseVariableType
    109 +       DataType: Size
    110 +       ModellingRule: Optional
    111 +       Value: 210.0
    112 +     - Name: TopMargin
    113 +       Type: BaseVariableType
    114 +       DataType: Size
    115 +       ModellingRule: Optional
    116 +       Value: 90.0
    117 +     - Name: LeftMargin
    118 +       Type: BaseVariableType
    119 +       DataType: Size
    120 +       ModellingRule: Optional
    121 +       Value: 80.0
```

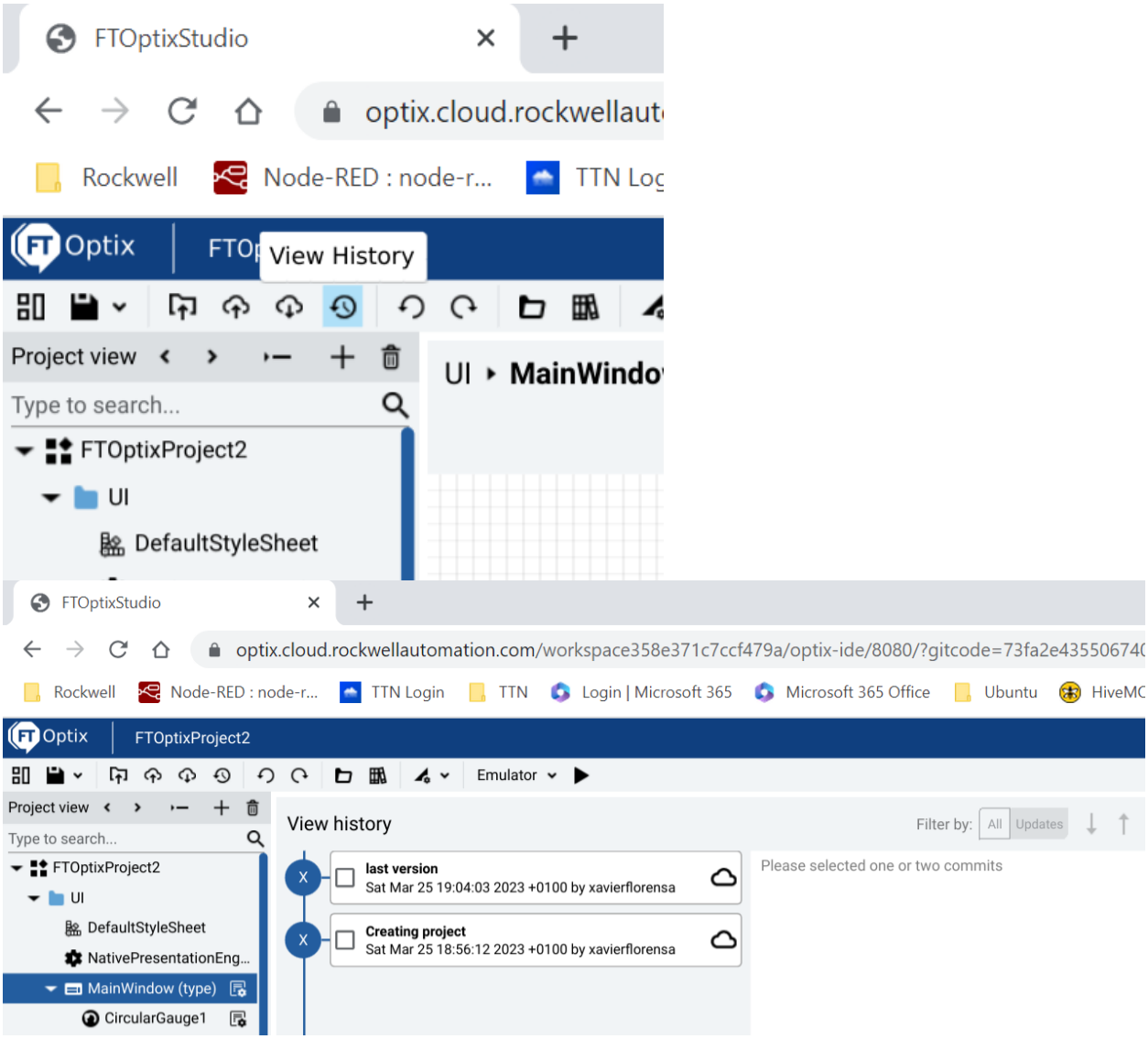
0 comments on commit ff563b7

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

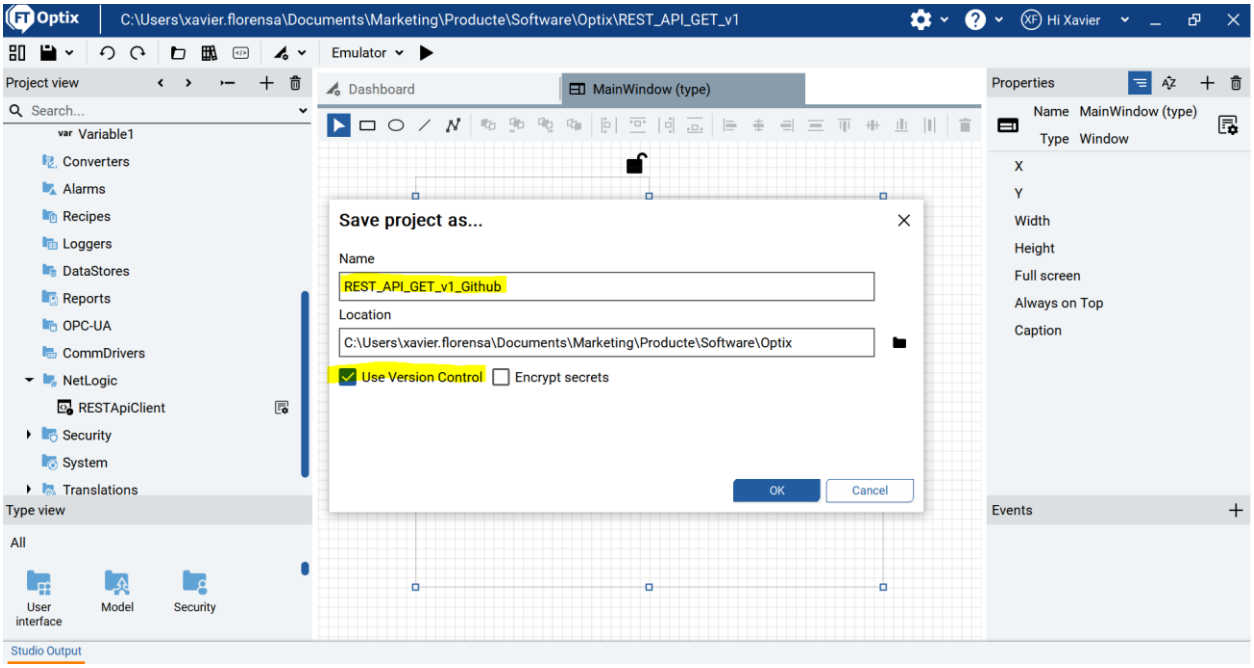
You can also view history on Optix



26. Pushing your local Optix projects to Github

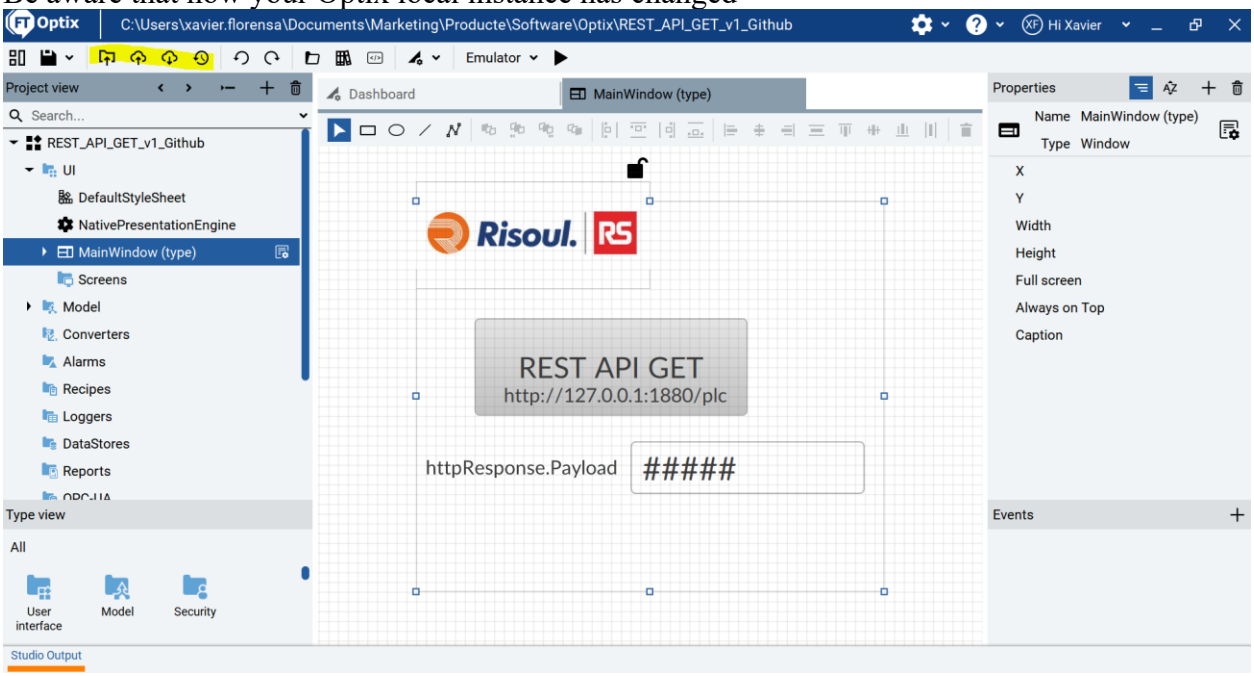
26.1. Using Optix

Probably you have started with Optix locally and created some projects that you want to store later on on Github without having to create them again.
Save your project with a different name and copy the project name
For instance
REST_API_GET_v1_Github
Mark “Use version control”



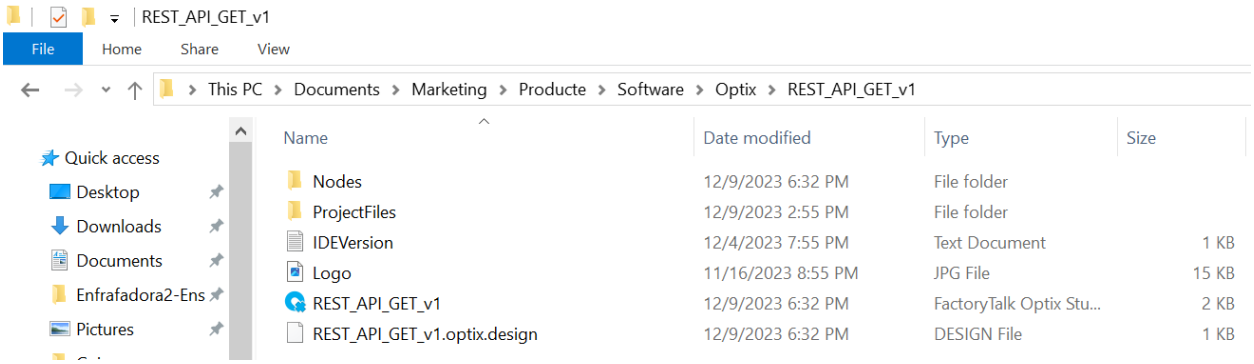
Click OK

Be aware that now your Optix local instance has changed

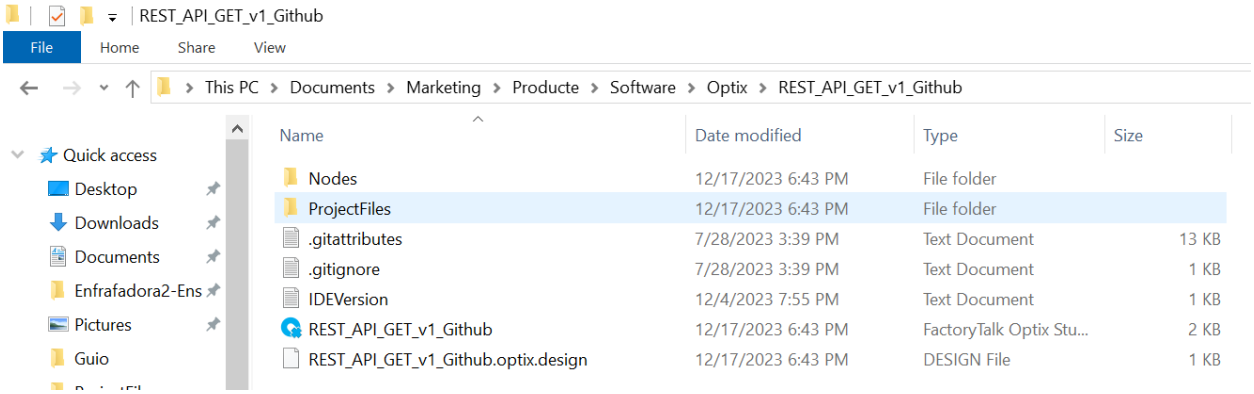


Also the files and directory has changed

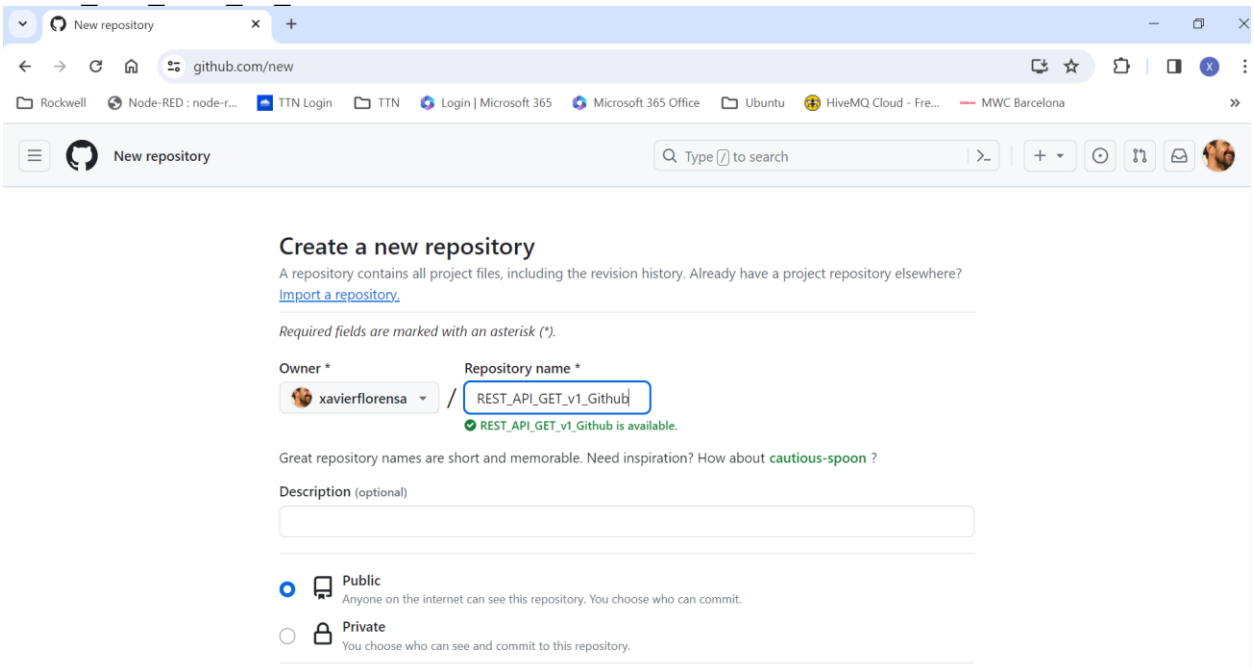
Before



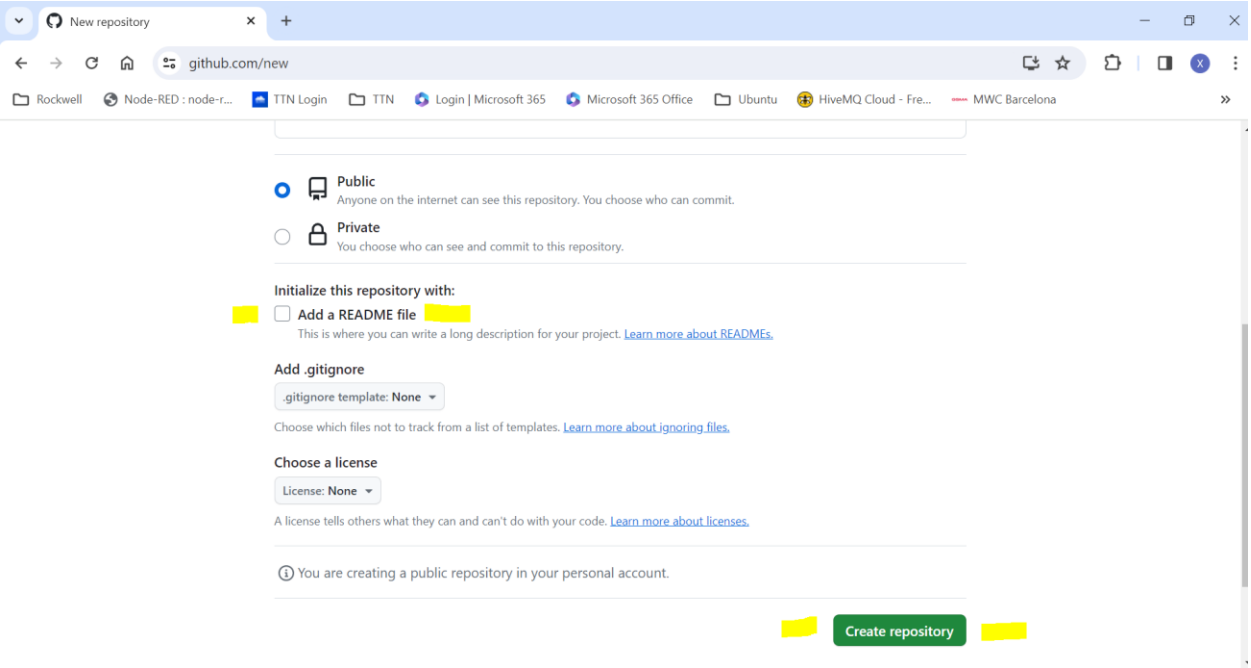
After



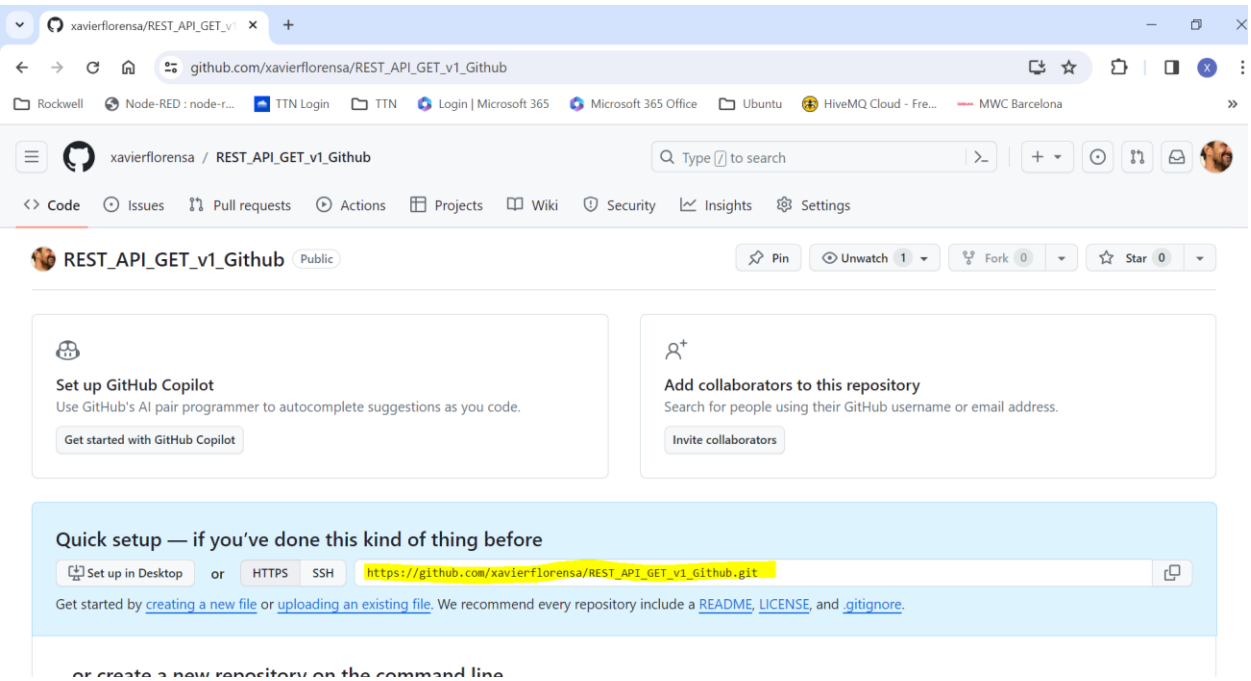
Now let's create a repository with same name on Github
So
REST_API_GET_v1_Github



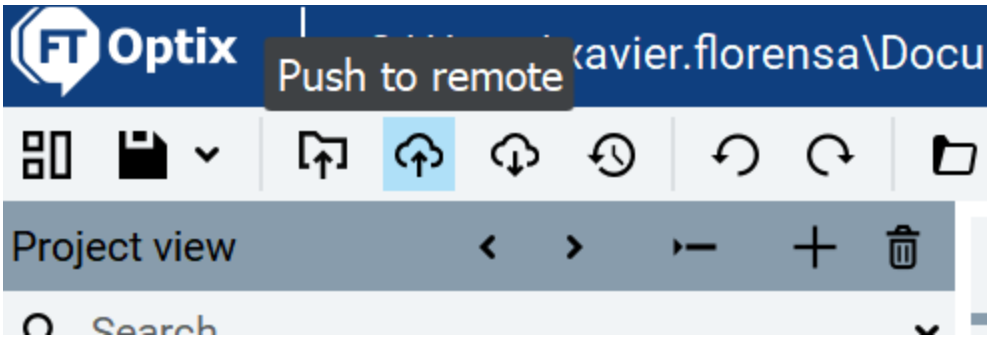
Do not check create a readme file



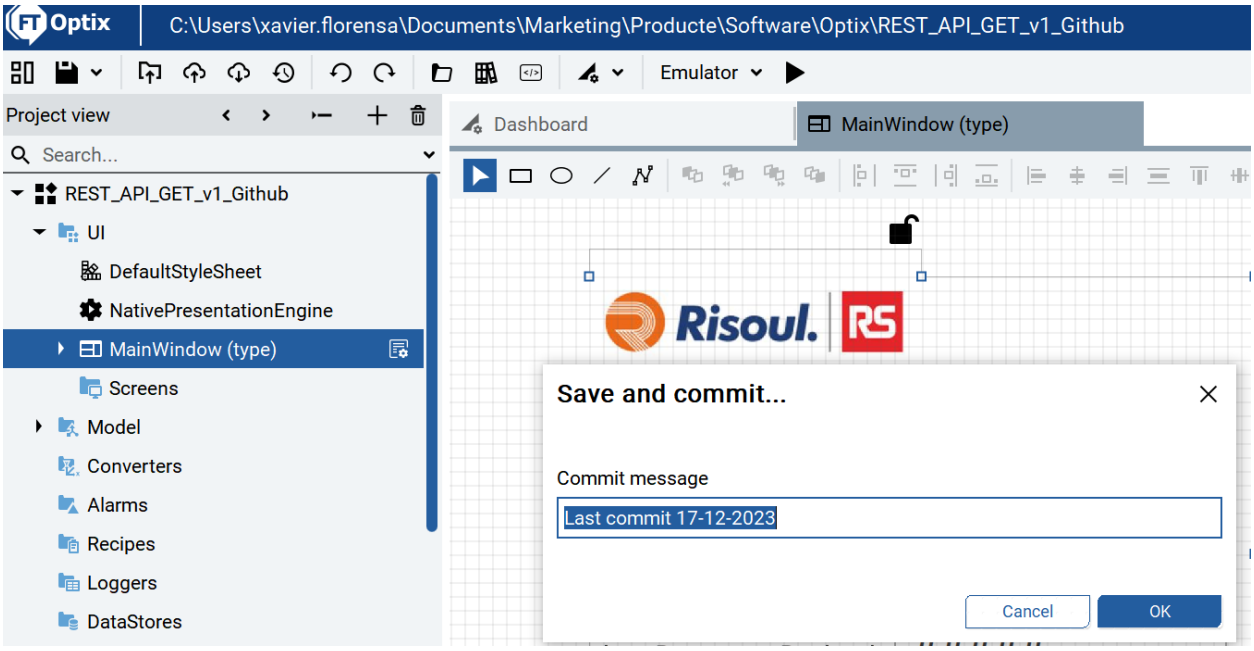
Click on create



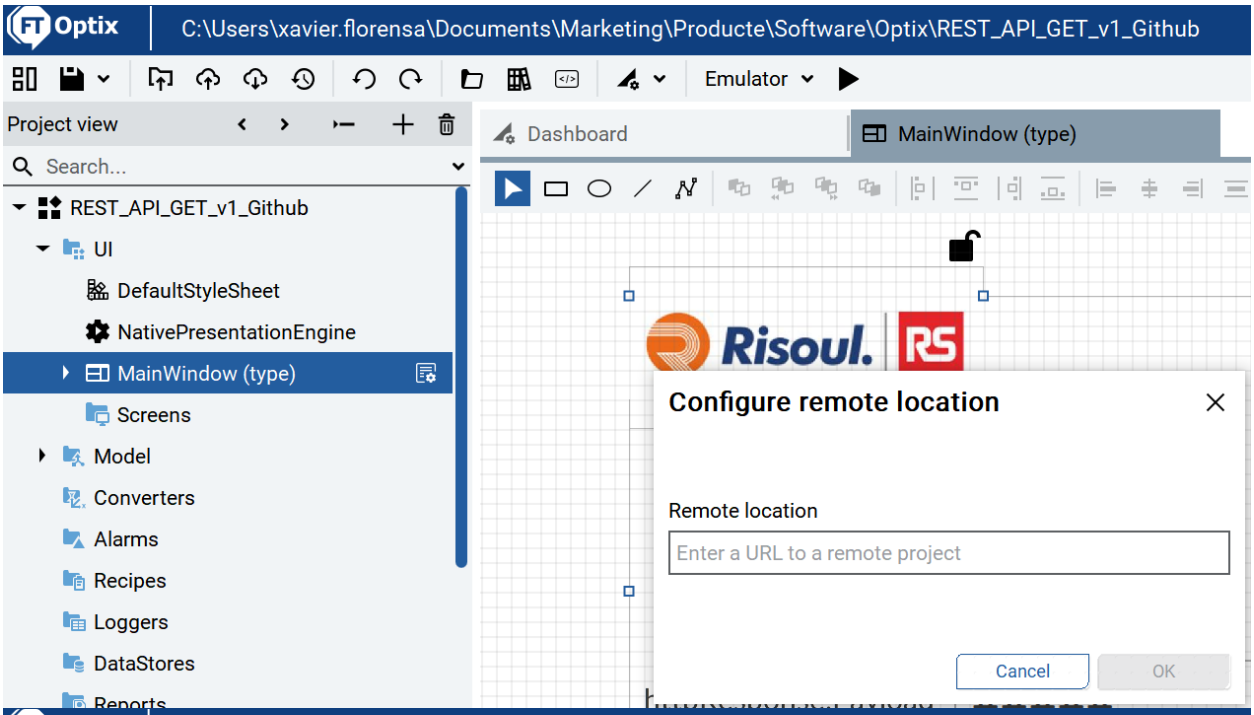
Copy this address
https://github.com/xavierflorensa/REST_API_GET_v1_Github.git
Your new repository is empty.
Now go to Optix Studio local
And hit on “Push to remote”

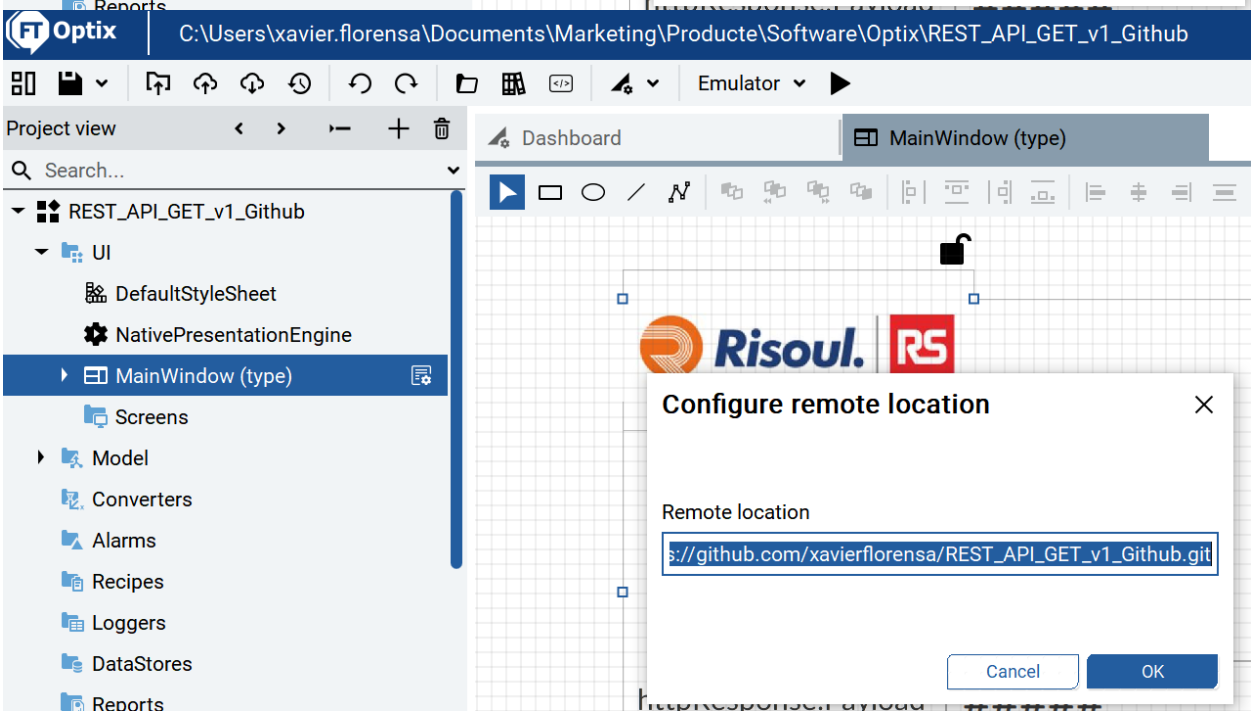


Give a comment if needed

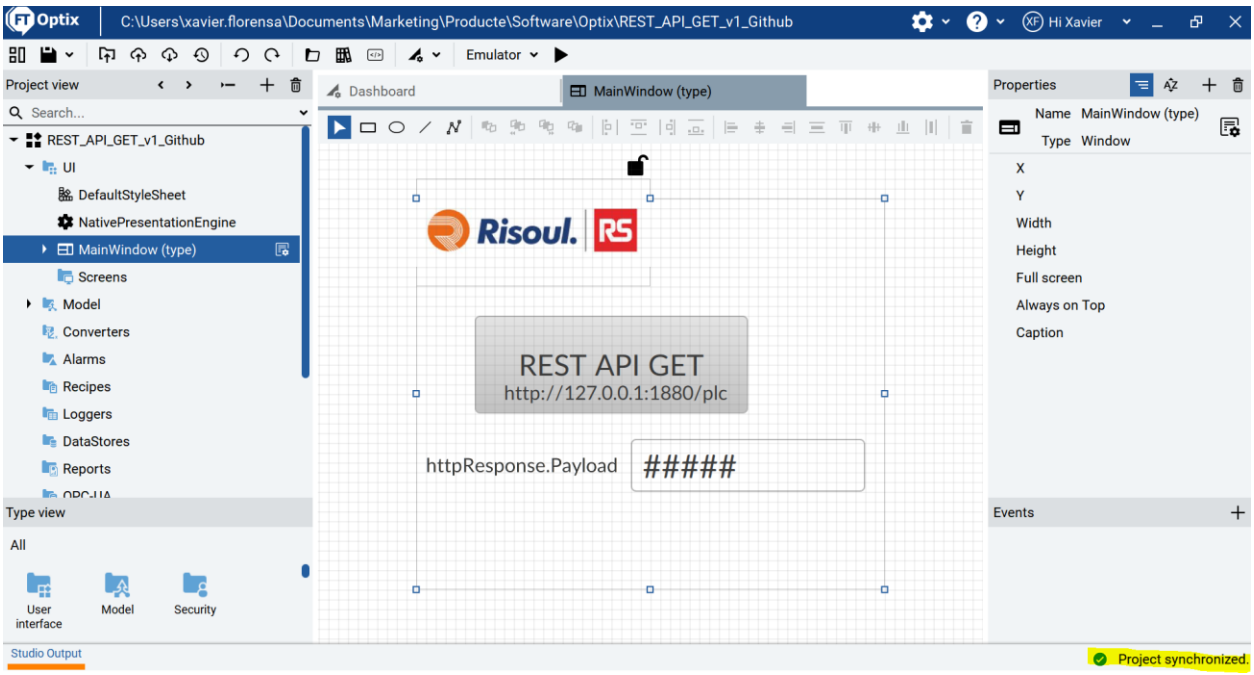


Give here the copied url address

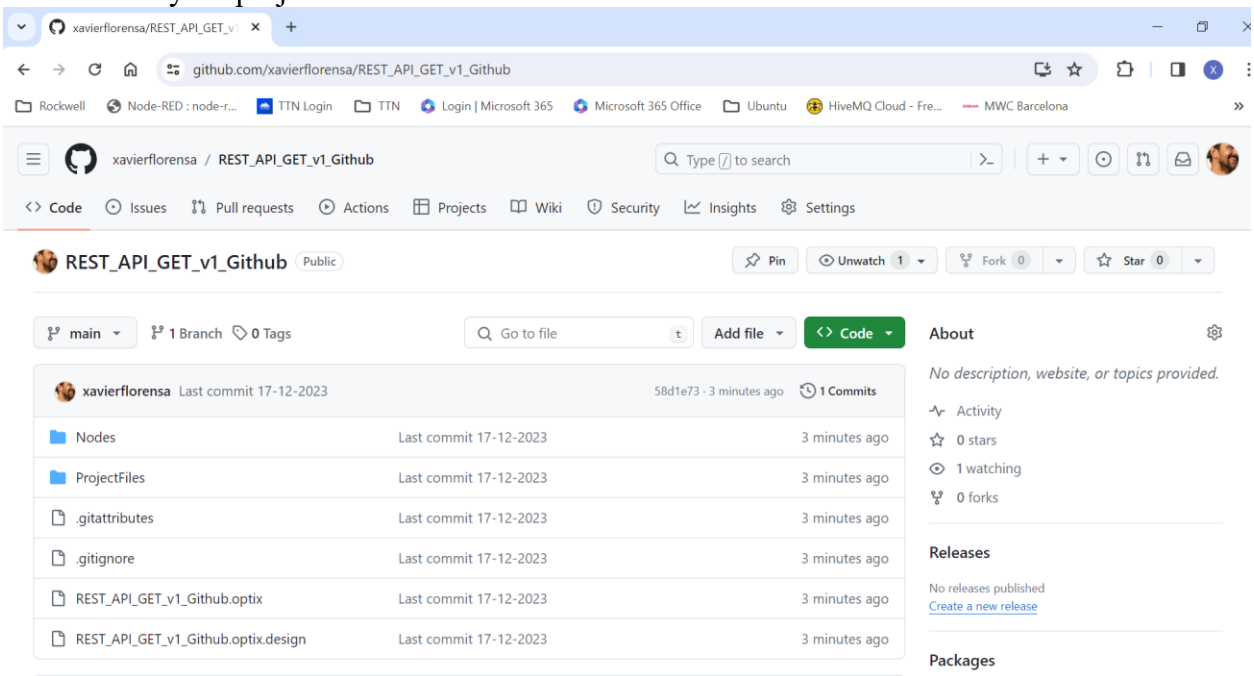




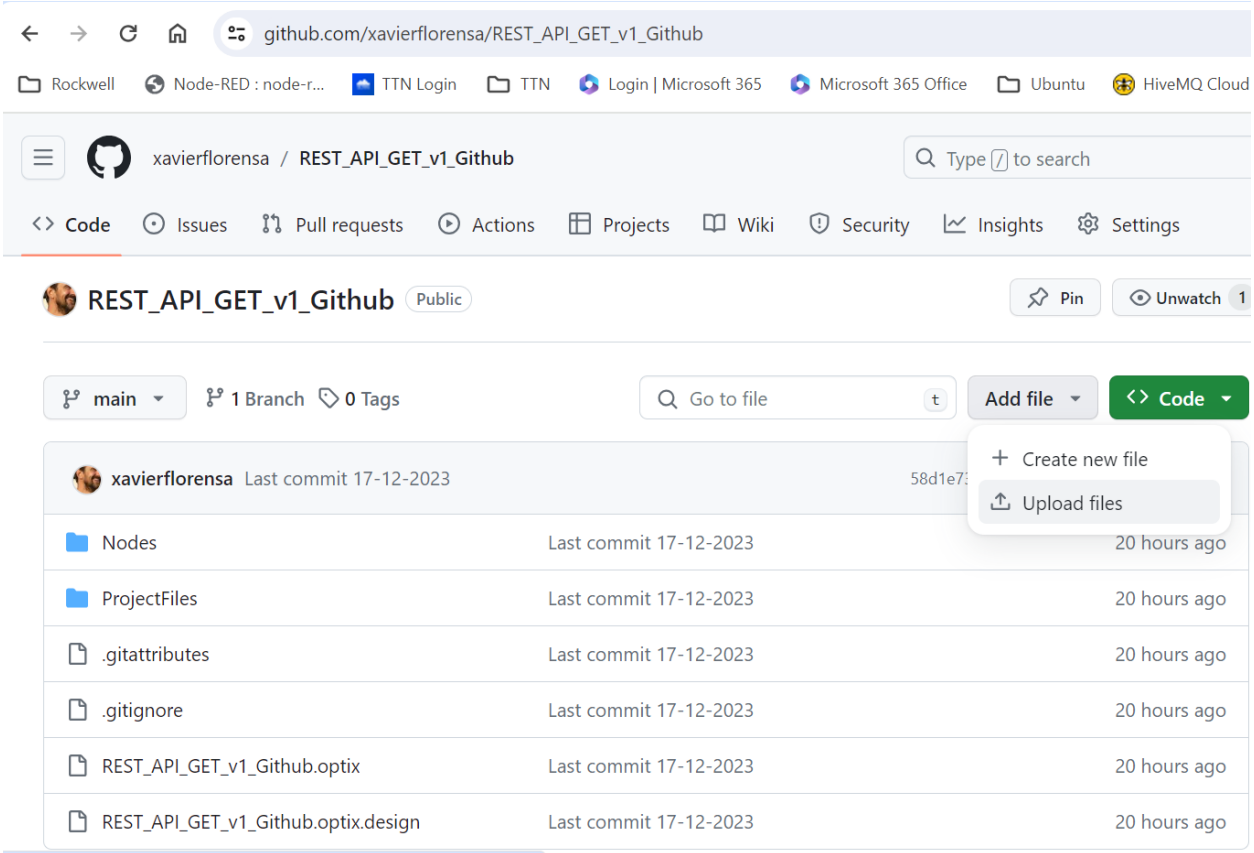
You get no errors and project is synchronized



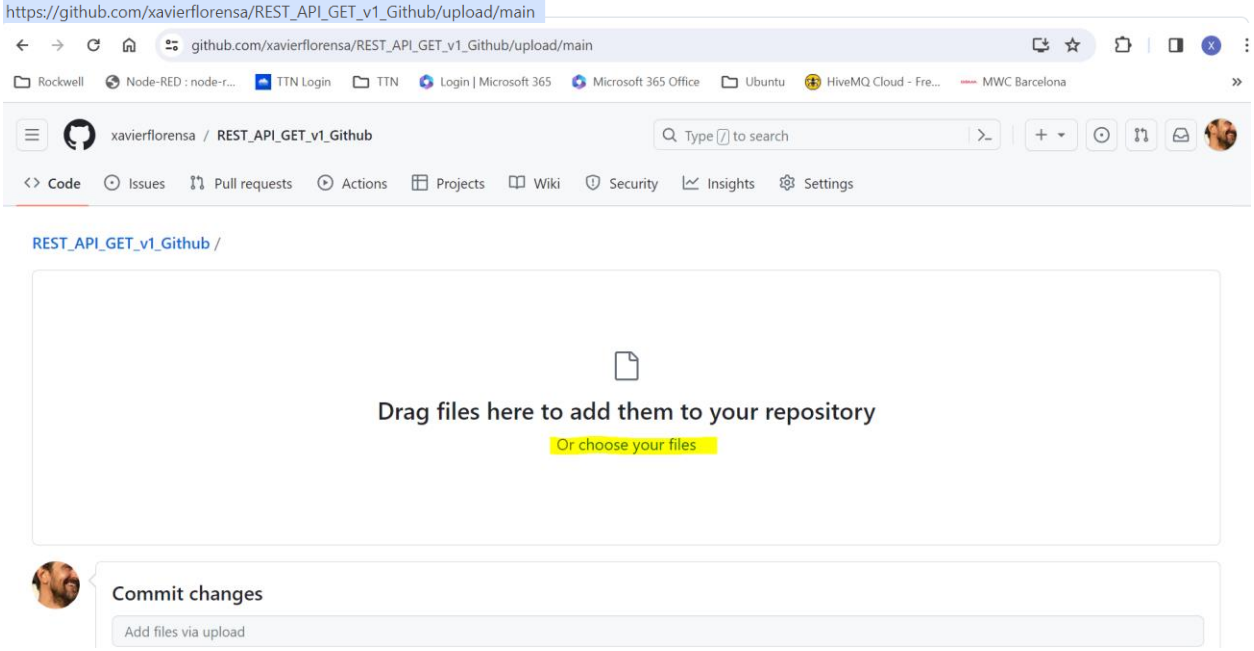
Now look at your project in Github

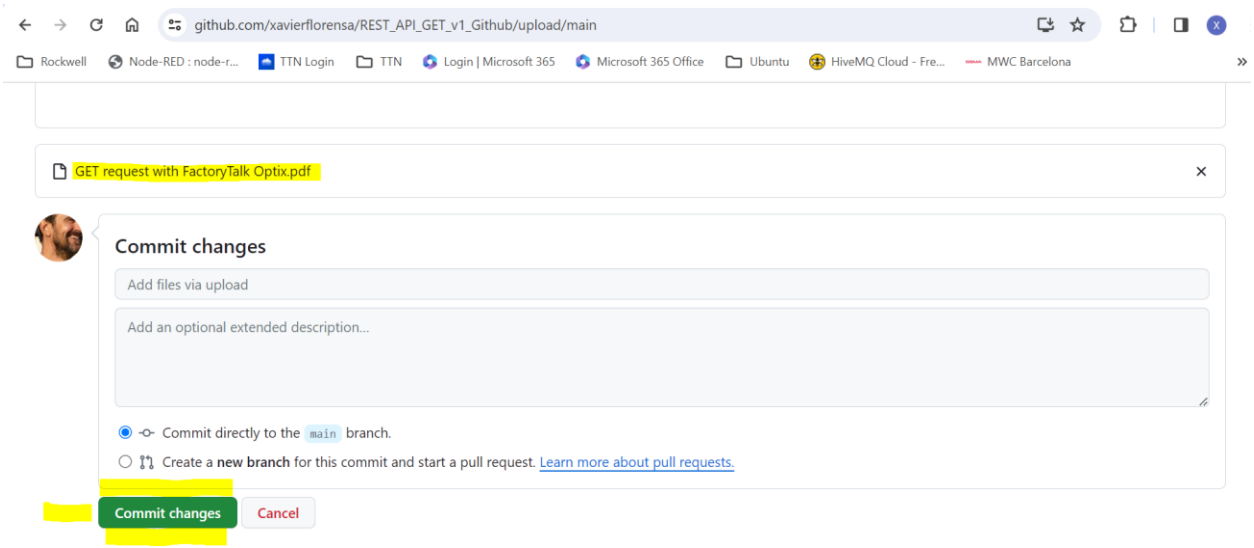


Your project is now public (or private) and you can share now this project with others.
Next step will be how to include a document with info since readme file is only text.
Let's try to do so from Github.
Add file / upload file

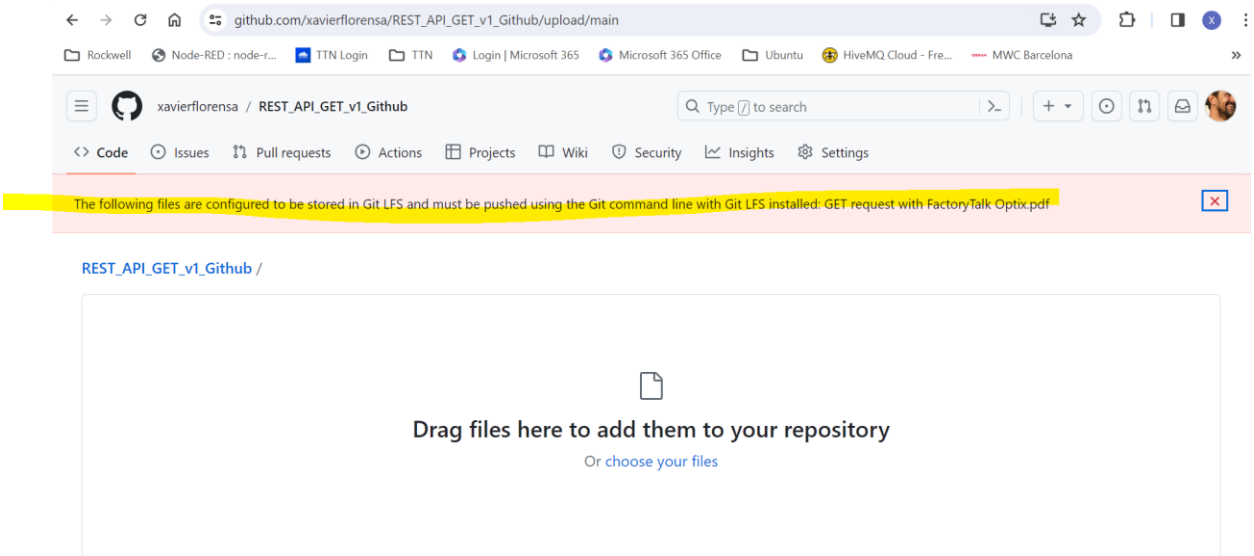


https://github.com/xavierflorensa/REST_API_GET_v1_Github/upload/main

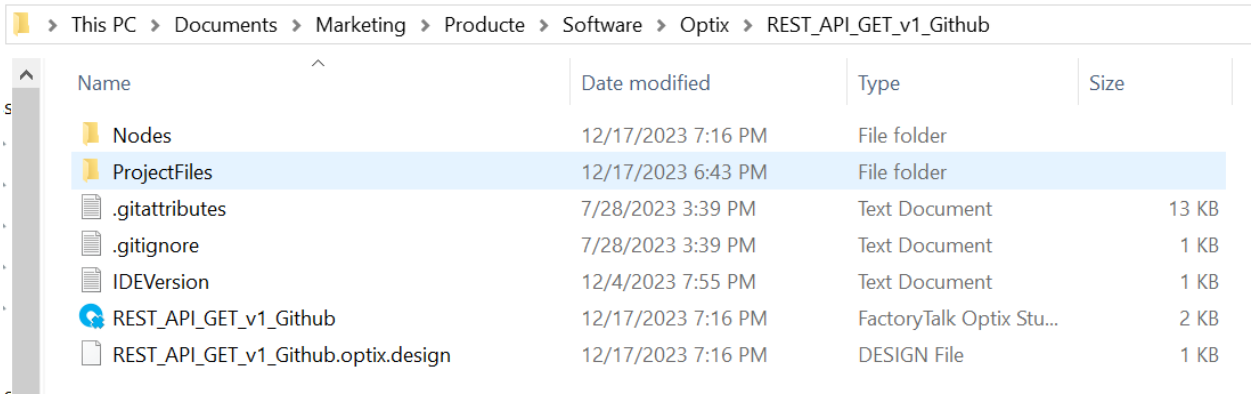




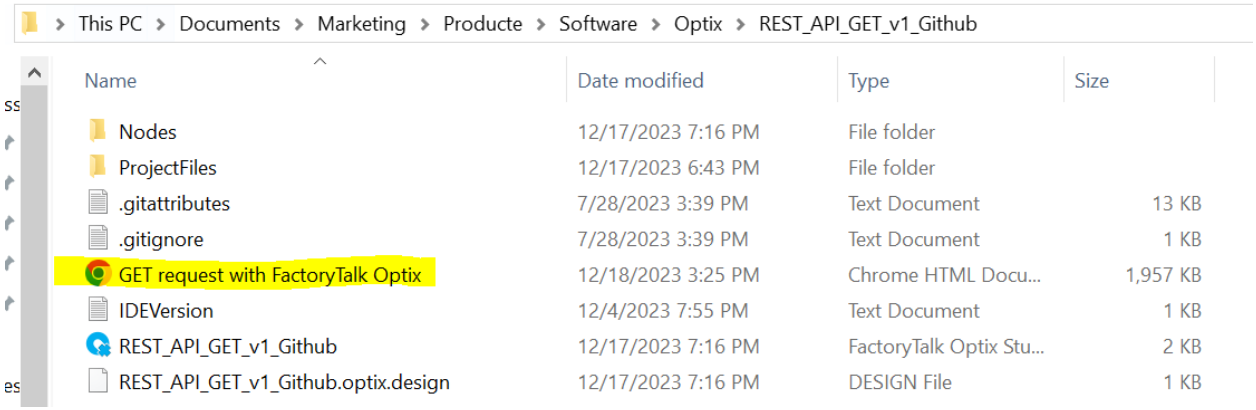
You can not do that this way



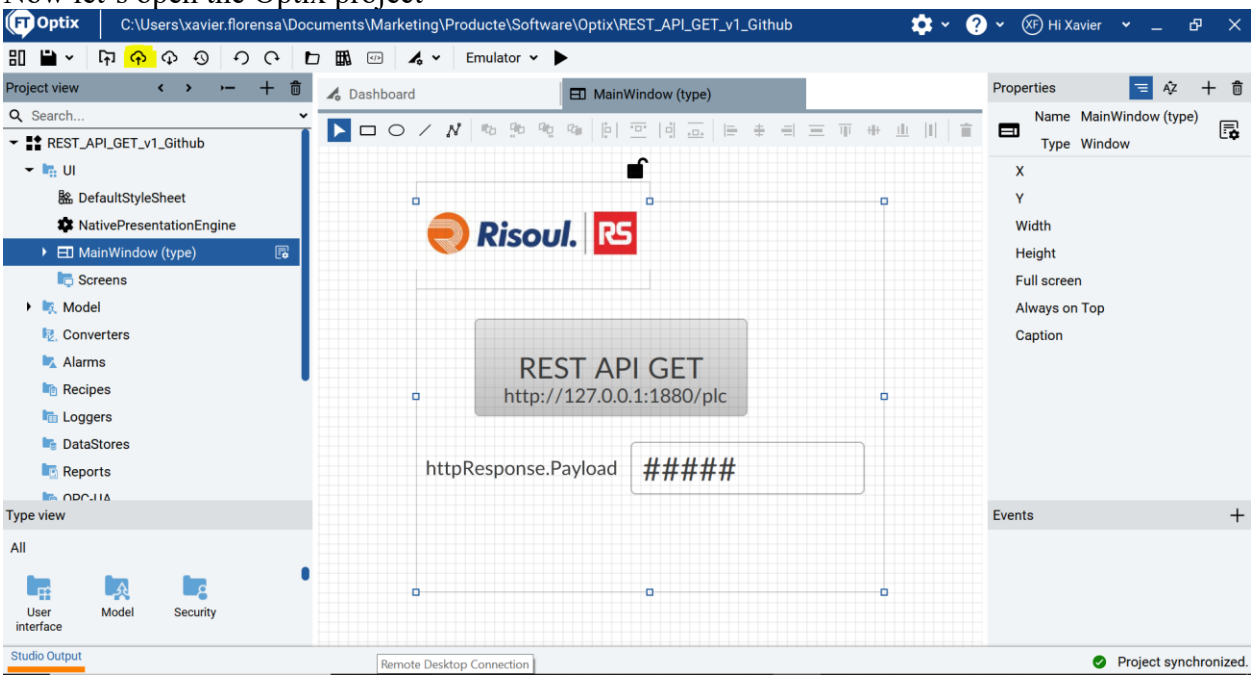
So you have to do it with GIT.
Unless you go back to your local Optix project directory and add a file there.
Like this



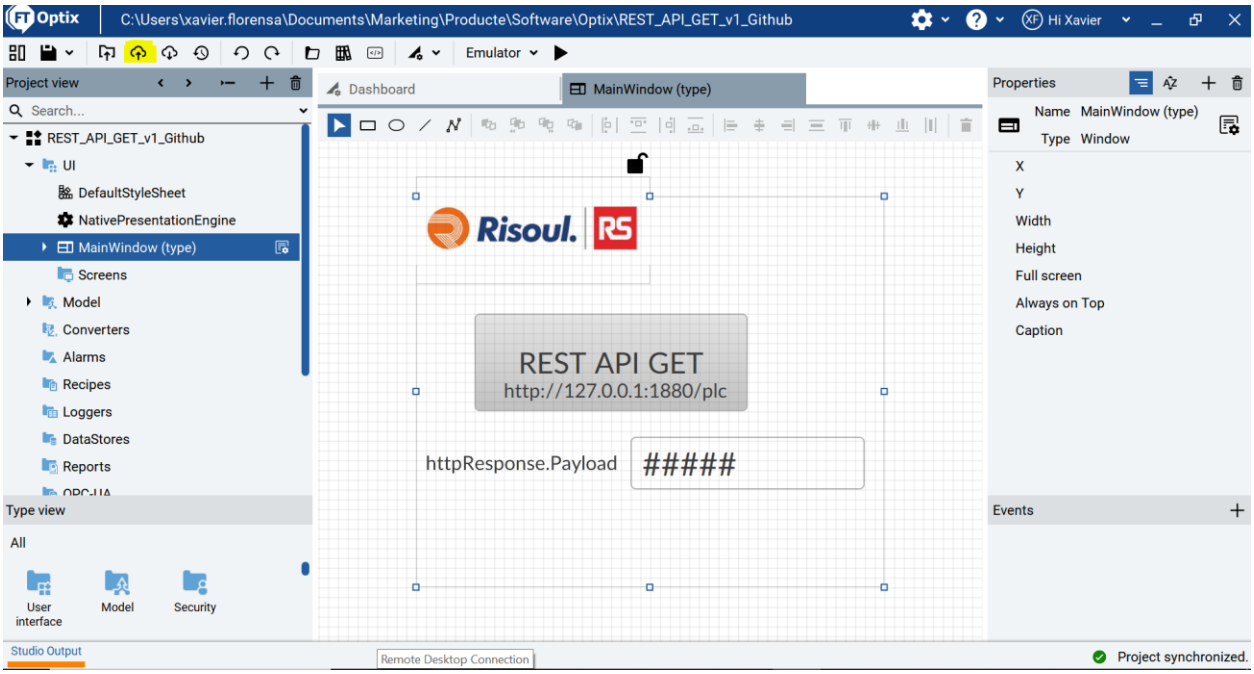
I add the documentation file I have forgot



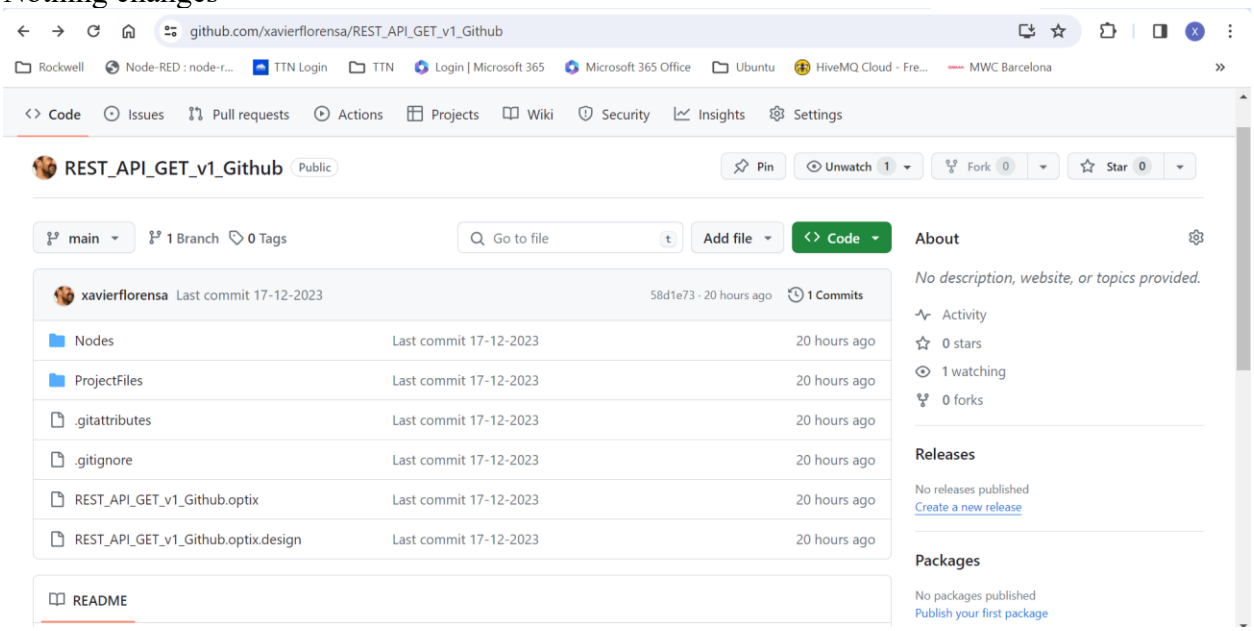
Now let's open the Optix project



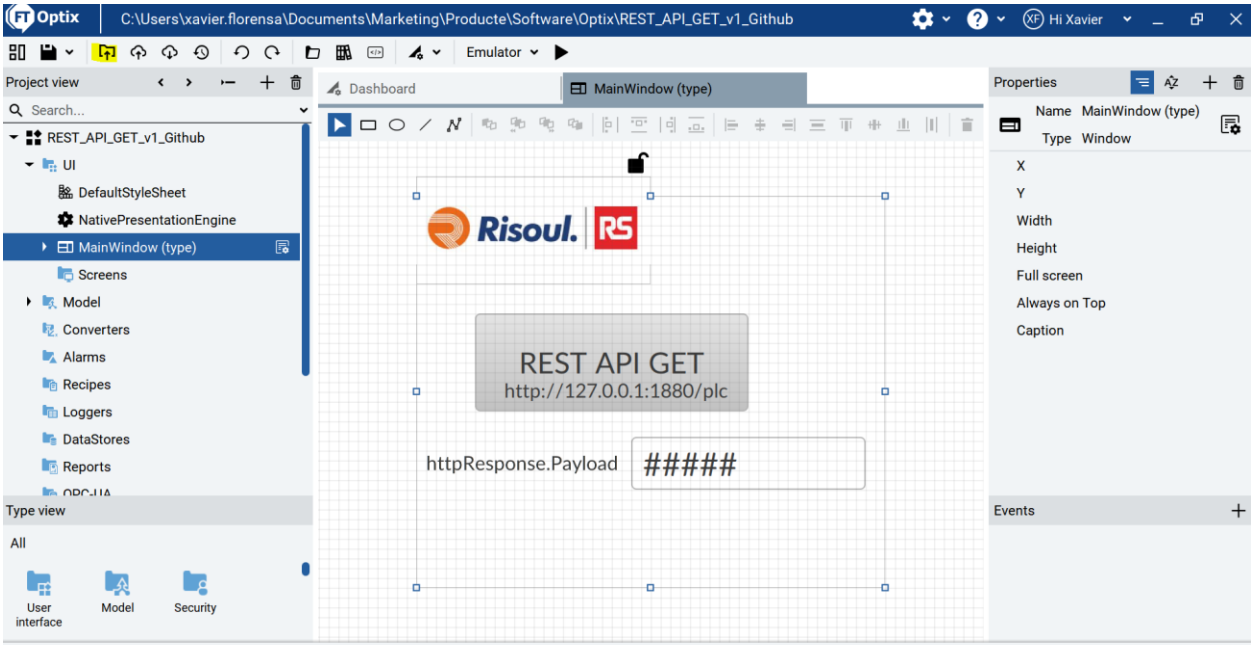
Click on push

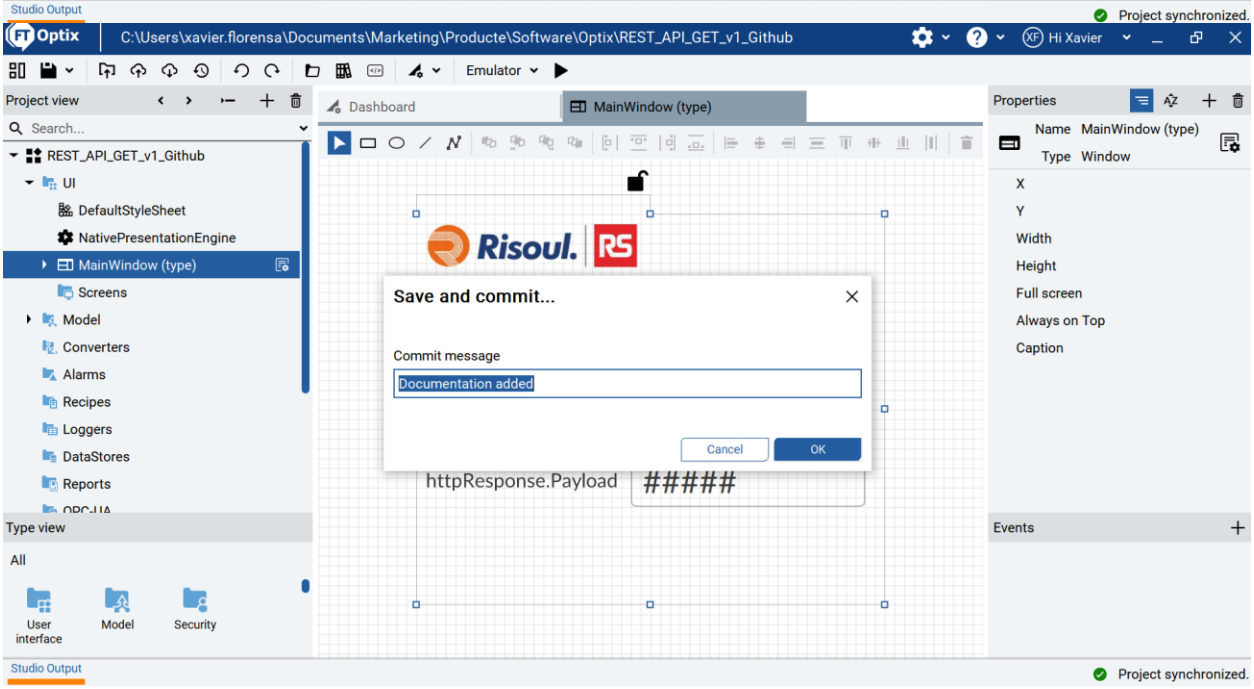


Nothing changes

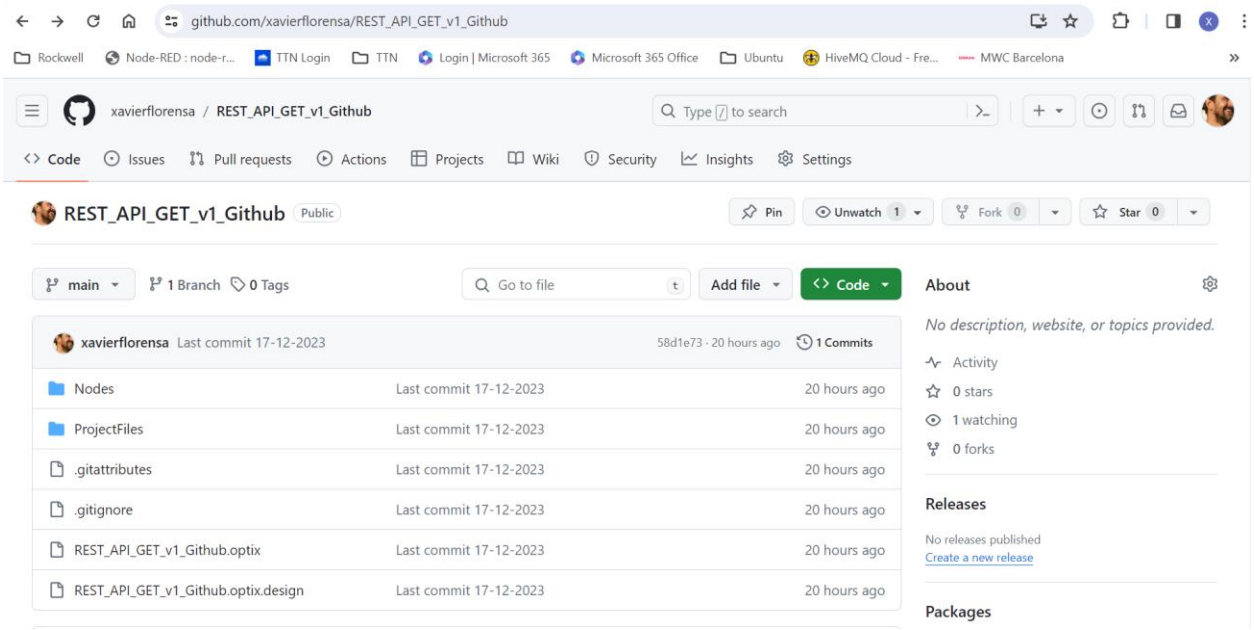


Let’s try with save and commit

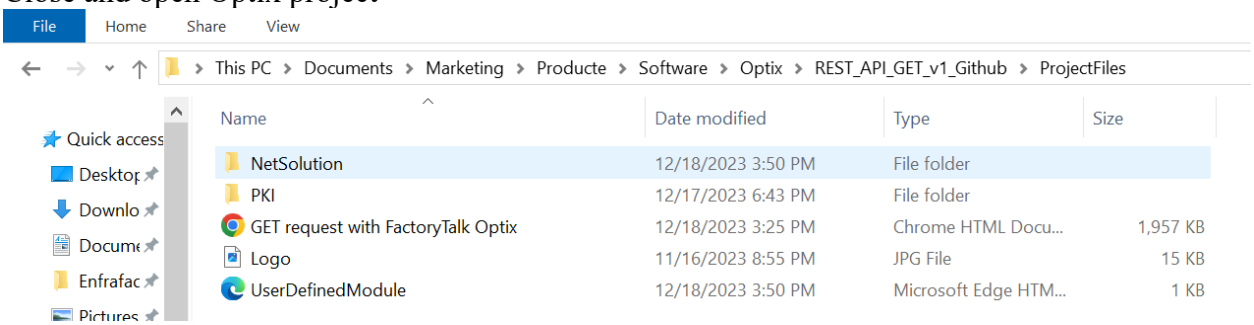




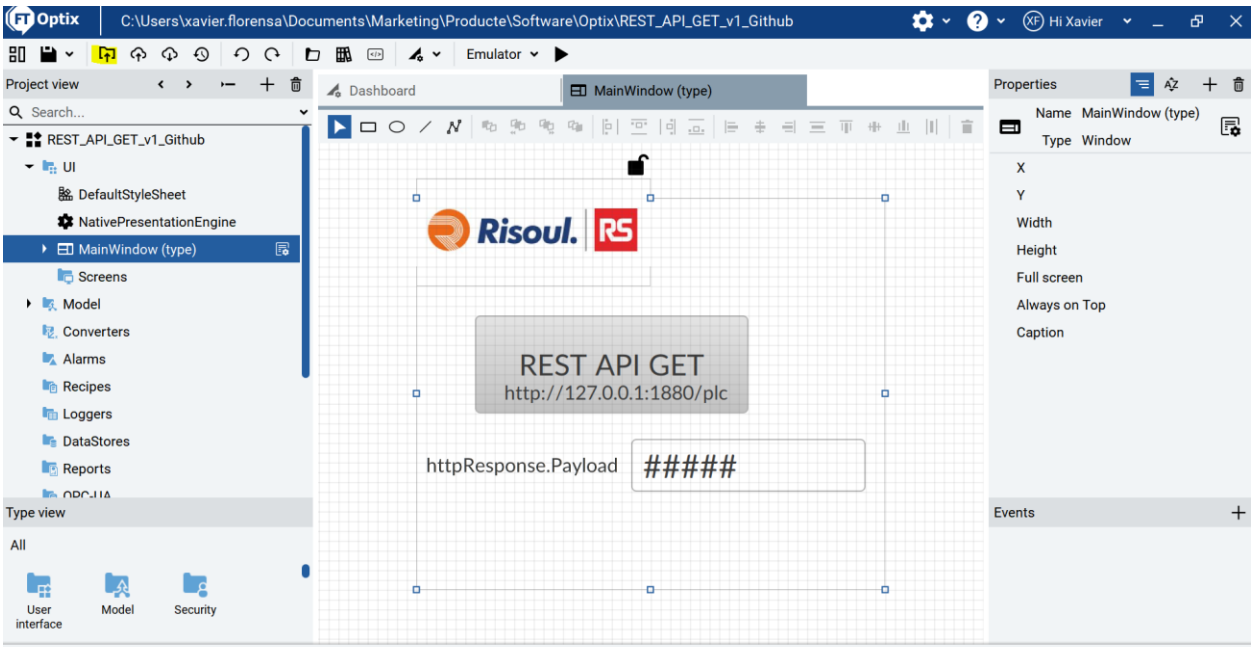
Nothing changes

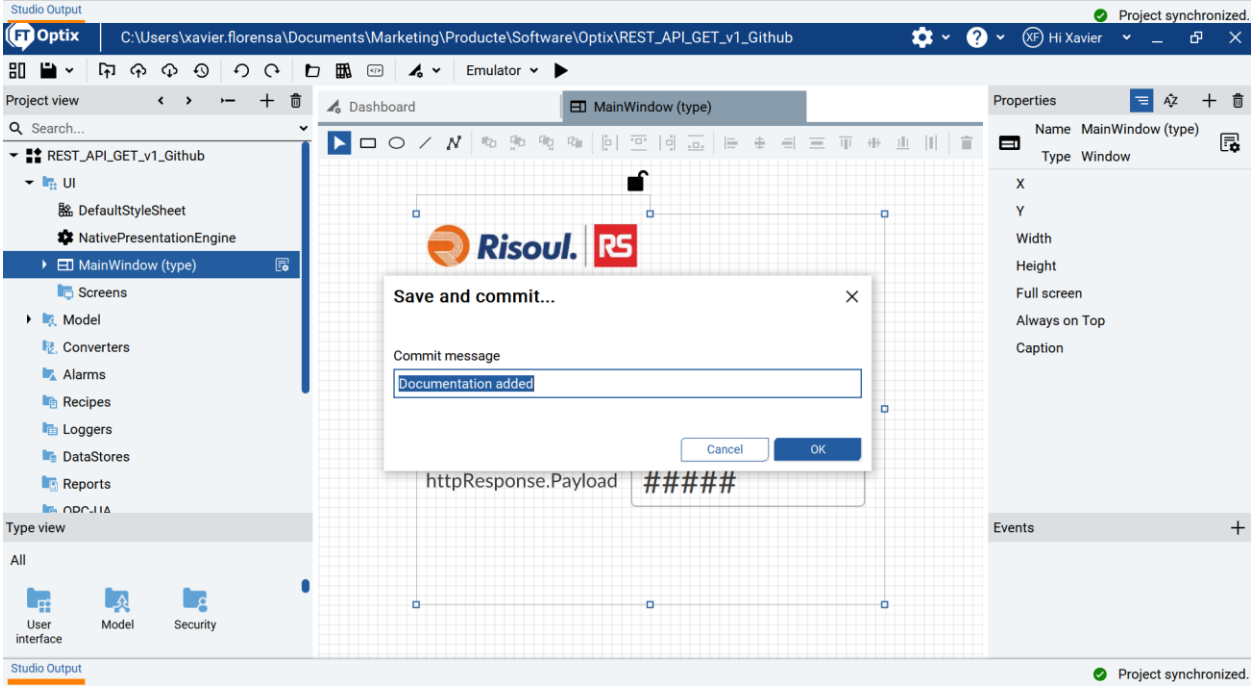


Let’s try to copy the file to Project files directory on our local directory
Close and open Optix project

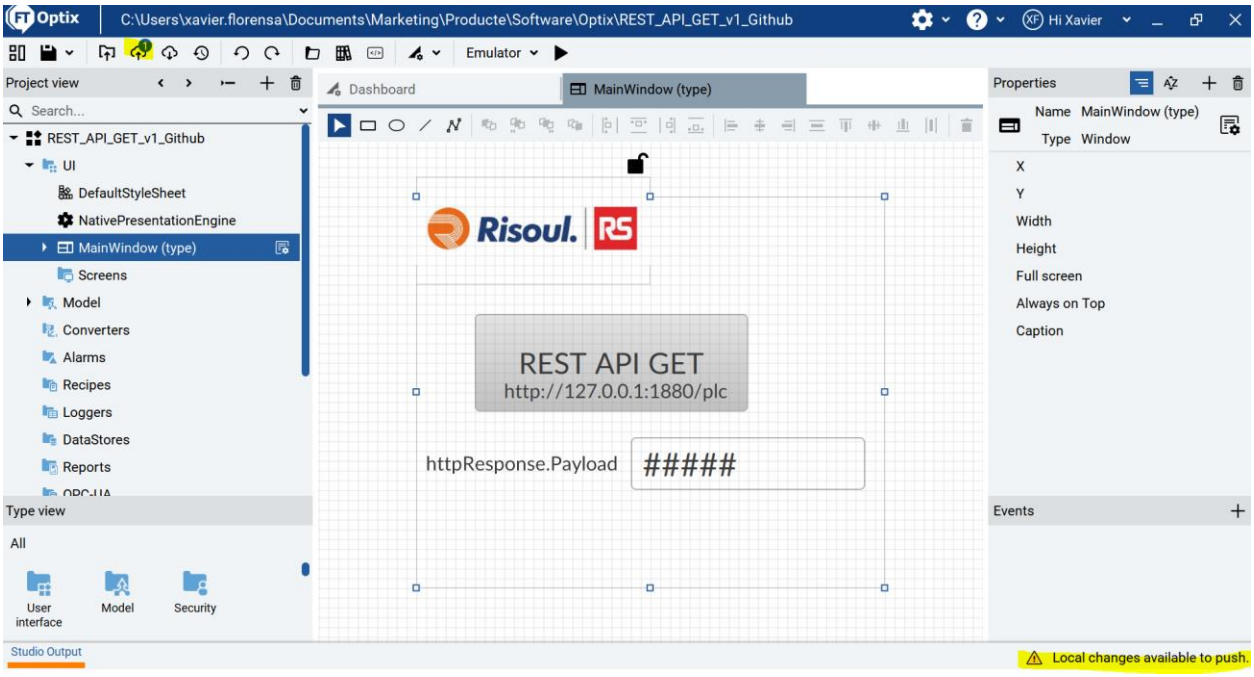


Now let’s repeat
Save and commit

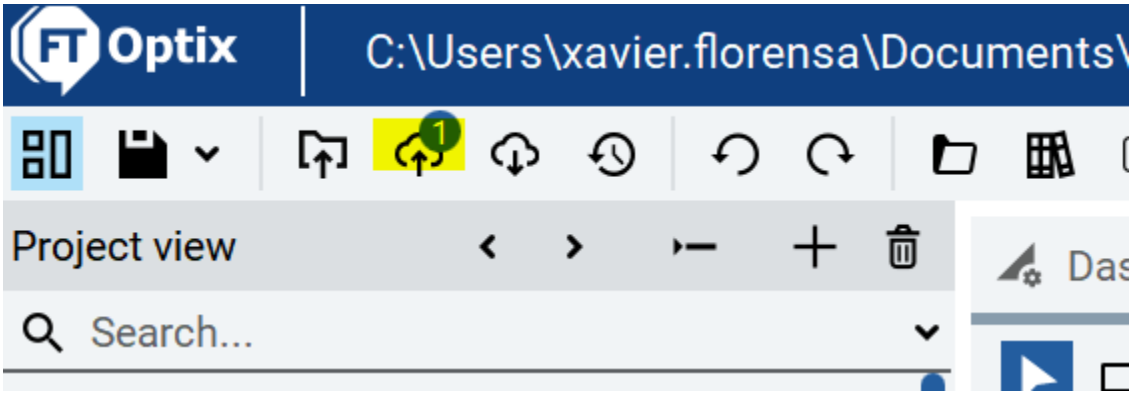




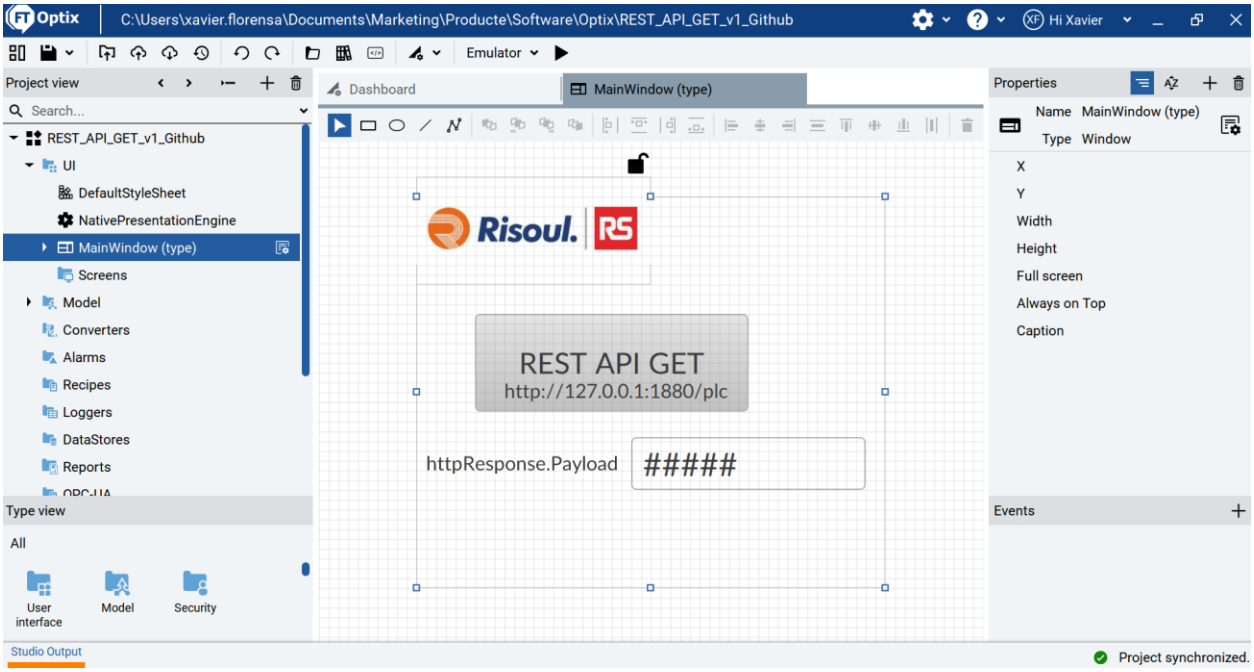
Something has changed now



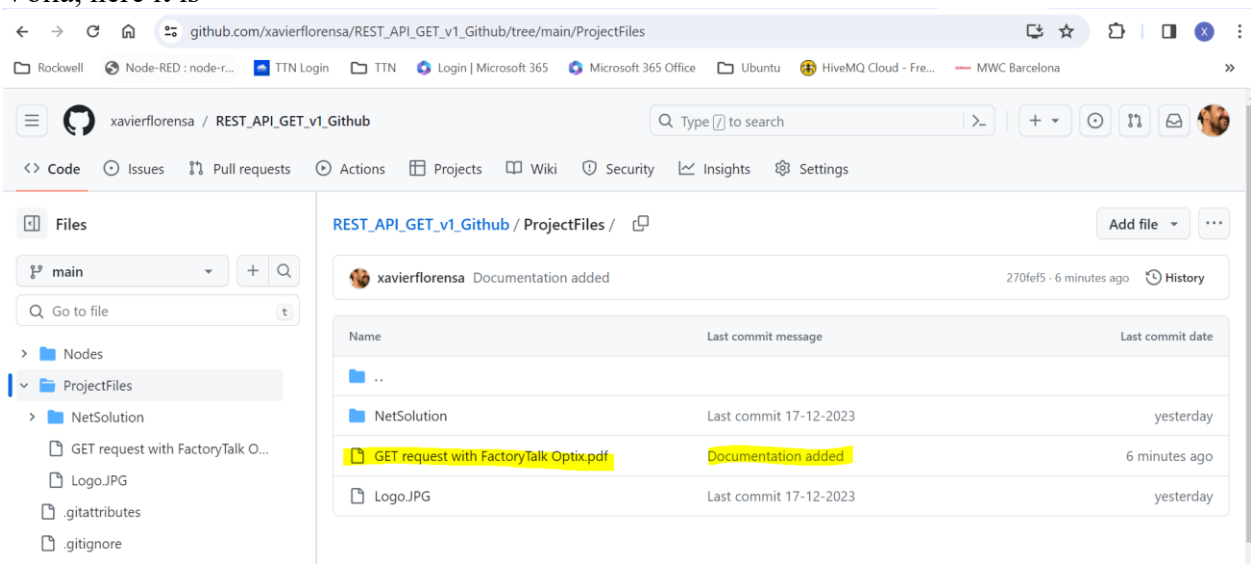
If we click on Push



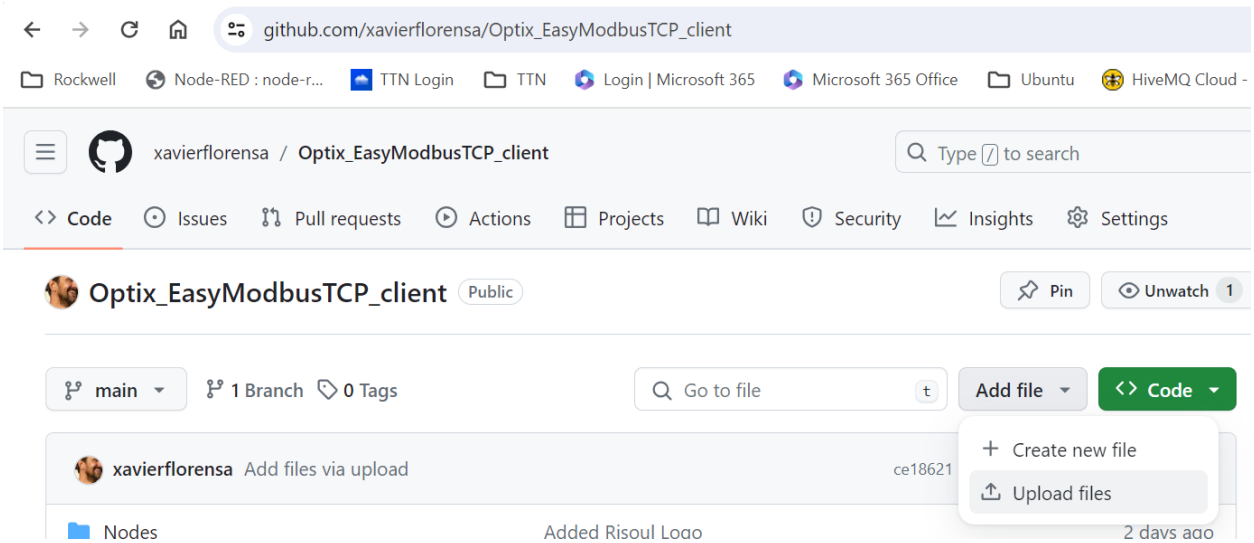
Now project is synchronized



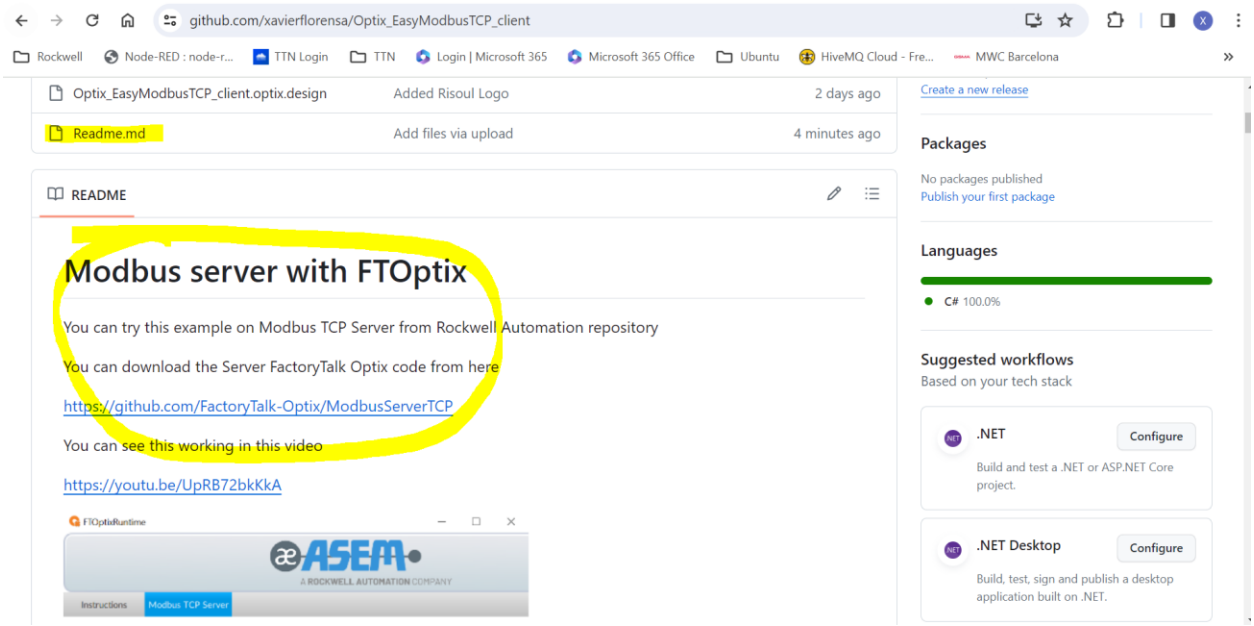
And if we open our Github project
Voilà, here it is



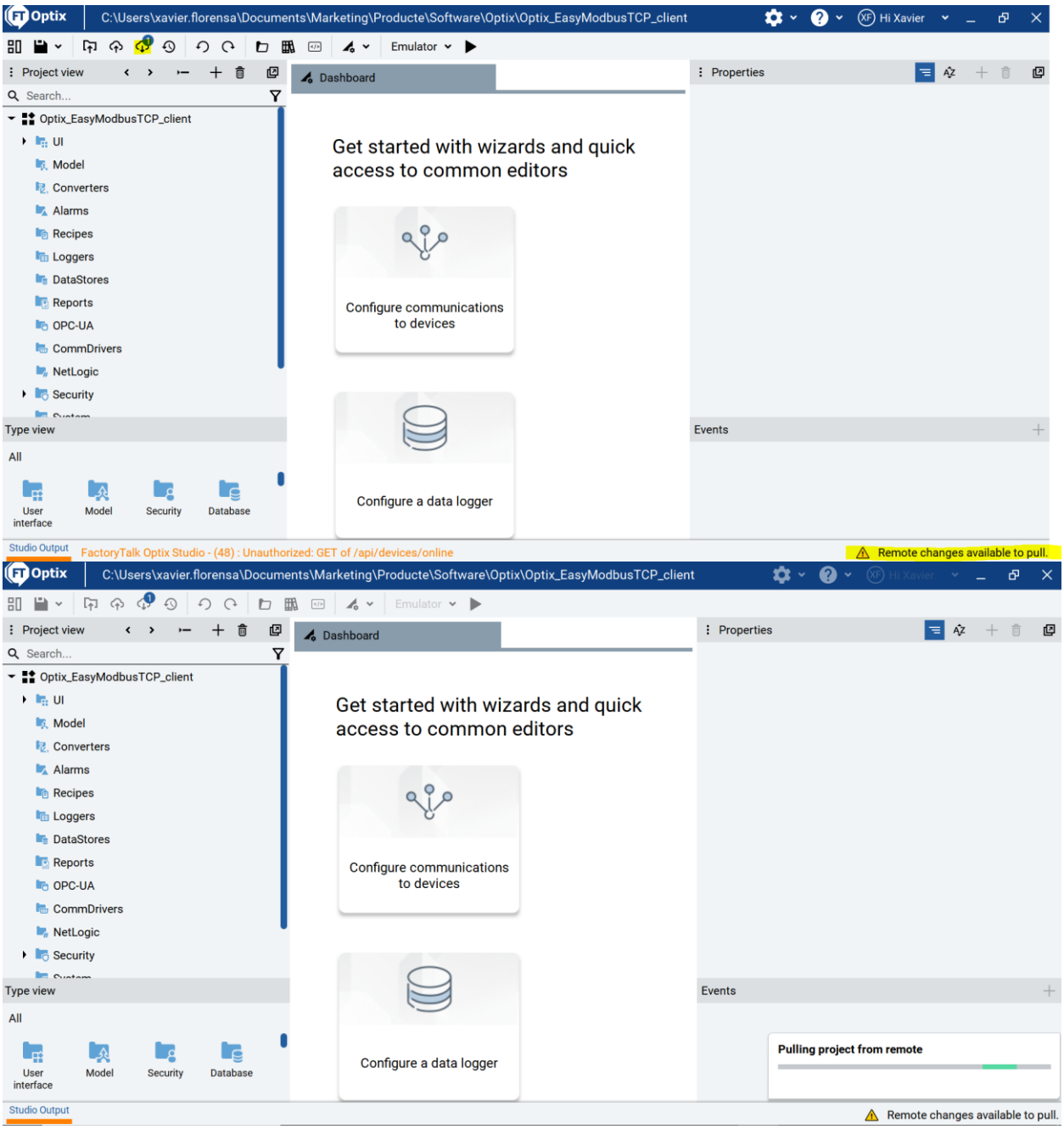
You can add a Readme.md file after the project is created.
Upload files



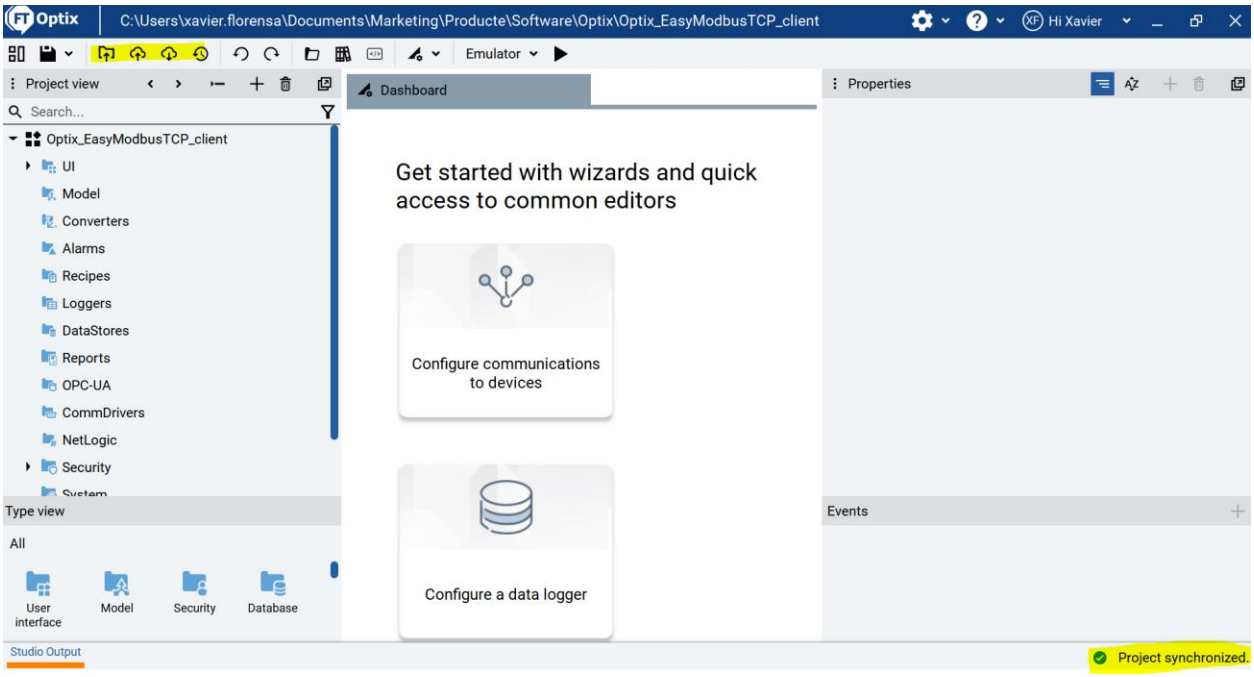
You have it



But your local project will not be syncronized
Then when you open the project locally on your computer
You will see this

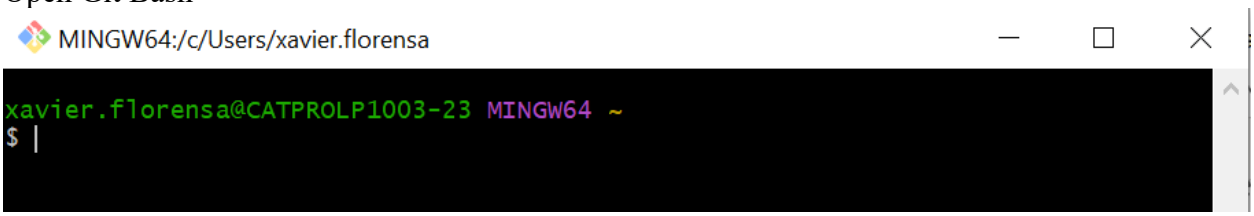


Now your project is synchronized



26.2. Using Git

First install Git on your machine
This is explained here
https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github
Locate the directory with your Optix project
Take a look at the path, you can not have spaces in between.
C:\Users\xavier.florensa\Documents\Rockwell Automation\FactoryTalk Optix\Projects
If so copy the directory on a shorter path like this
C:\Users\xavier.florensa\Documents\Optix
Open Git Bash



Navigate to the root directory of your project

MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd Documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix
$ cd PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
$ |
```

Inicialize a Git repository locally

```
git init -b main
```

MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd Documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix
$ cd PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
$ git init -b main
Initialized empty Git repository in C:/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git/.git/

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

Add the files to your local new repository

```
git add .
be aware that (main) in blue color is the branch you are working now
```

MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd Documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents
$ cd Optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix
$ cd PLC2MQTTClient_HiveMQ_v1_Git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git
$ git init -b main
Initialized empty Git repository in C:/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git/.git/

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git add .

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

Commit the files you have staged in your local repository

```
git commit -m "First commit"
```

MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git commit -m "First commit"
[main (root-commit) 8ca9b56] First commit
 Committer: Xavier Florensa Berenguer <xavier.florensa@soporte.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 65 files changed, 24341 insertions(+)
 create mode 100644 DesignTimeNodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/TagImporter.yaml
 create mode 100644 IDEVersion.txt
 create mode 100644 Nodes/Alarms/Alarms.yaml
 create mode 100644 Nodes/CommDrivers/CommDrivers.yaml
 create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPDriver1.yaml
 create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/DataTypes/DataTypes.yaml
 create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/RAEtherNet_IPStation1.yaml
 create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/Tags.yaml
 create mode 100644 Nodes/CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/VariableTypes/VariableTypes.yaml
 create mode 100644 Nodes/Converters/Converters.yaml
 create mode 100644 Nodes/DataStores/DataStores.yaml
 create mode 100644 Nodes/Loggers/Loggers.yaml
 create mode 100644 Nodes/Model/Model.yaml
 create mode 100644 Nodes/NetLogic/NetLogic.yaml
 create mode 100644 Nodes/OPC-UA/OPC-UA.yaml
 create mode 100644 Nodes/PLC2MQTTClient_HiveMQ_v1_Git.yaml
 create mode 100644 Nodes/Recipes/Recipes.yaml
 create mode 100644 Nodes/Reports/Reports.yaml
 create mode 100644 Nodes/Retentivity/Retentivity.yaml
 create mode 100644 Nodes/Security/Groups/Groups.yaml
 create mode 100644 Nodes/Security/Security.yaml
 create mode 100644 Nodes/Security/Users/Users.yaml
 create mode 100644 Nodes/Translations/Translations.yaml
 create mode 100644 Nodes/UI/UI.yaml
 create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix
 create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix.design
 create mode 100644 ProjectFiles/NetSolution/.vscode/launch.json
 create mode 100644 ProjectFiles/NetSolution/MQTTClient.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient1_v1.code-workspace
```

MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

```
 create mode 100644 Nodes/Security/Users/Users.yaml
 create mode 100644 Nodes/Translations/Translations.yaml
 create mode 100644 Nodes/UI/UI.yaml
 create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix
 create mode 100644 PLC2MQTTClient_HiveMQ_v1_Git.optix.design
 create mode 100644 ProjectFiles/NetSolution/.vscode/launch.json
 create mode 100644 ProjectFiles/NetSolution/MQTTClient.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient1_v1.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v0.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.csproj
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.references
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.sln
 create mode 100644 ProjectFiles/NetSolution/Private/TypeConstants.cs
 create mode 100644 ProjectFiles/NetSolution/Private/UITypeDefinitions.cs
 create mode 100644 ProjectFiles/NetSolution/PublisherLogic.cs
 create mode 100644 ProjectFiles/NetSolution/SubscriberLogic.cs
 create mode 100644 ProjectFiles/NetSolution/bin/M2Mqtt.Net.dll
 create mode 100644 ProjectFiles/NetSolution/bin/PLC2MQTTClient_HiveMQ_v1_Git.deps.json
 create mode 100644 ProjectFiles/NetSolution/bin/PLC2MQTTClient_HiveMQ_v1_Git.dll
 create mode 100644 ProjectFiles/NetSolution/bin/PLC2MQTTClient_HiveMQ_v1_Git.pdb
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/.NETCoreApp,Version=v6.0.AssemblyAttributes.cs
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.AssemblyInfo.cs
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.AssemblyInfoInputs.cache
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.GeneratedMSBuildEditorConfig.editorconfig
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.assets.cache
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.AssemblyReference.cache
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.CopyComplete
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.CoreCompileInputs.cache
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.csproj.FileListAbsolute.txt
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.dll
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/PLC2MQTTClient_HiveMQ_v1_Git.pdb
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/ref/PLC2MQTTClient_HiveMQ_v1_Git.dll
 create mode 100644 ProjectFiles/NetSolution/obj/Debug/refint/PLC2MQTTClient_HiveMQ_v1_Git.dll
 create mode 100644 ProjectFiles/NetSolution/obj/PLC2MQTTClient_HiveMQ_v1_Git.csproj.nuget.dgspec.json
 create mode 100644 ProjectFiles/NetSolution/obj/PLC2MQTTClient_HiveMQ_v1_Git.csproj.nuget.g.props
 create mode 100644 ProjectFiles/NetSolution/obj/PLC2MQTTClient_HiveMQ_v1_Git.csproj.nuget.g.targets
 create mode 100644 ProjectFiles/NetSolution/obj/project.assets.json
 create mode 100644 ProjectFiles/NetSolution/obj/project.nuget.cache
 create mode 100644 ProjectFiles/UserDefinedModule.xml
 create mode 100644 ProjectFiles/prova35v0.ACD
 create mode 100644 prova35v0.ACD

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

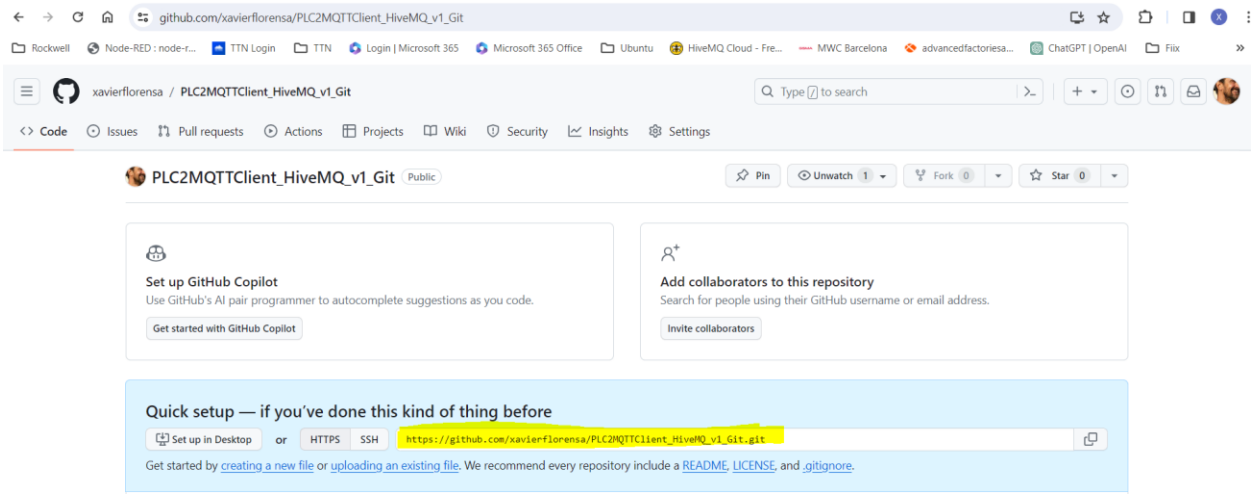
If you want, you can make a log

If you look at HEAD, it points to the local branch (main) where we are now, and you have also the remote branch (origin/main). Origin is the short name of the remote repository

MINGW64:/c/Users/xavier.florensa/documents/optix/plc2mqttclient_hivemq_v1_git

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~
$ cd documents

xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents
$ cd optix

xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix
$ cd plc2mqttclient_hivemq_v1_git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_
v1_git (main)
$ git log
commit 8ca9b565b9b143c310ddbf6b92ba5e4d161d30f2 (HEAD -> main, origin/main)
Author: Xavier Florensa Berenguer <xavier.florensa@soporte.com>
Date:   Sun Dec 17 19:42:22 2023 +0100

    First commit

xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_
v1_git (main)
$ |
```

Create a new repository with the same name (or similar) on Github
Copy this url
https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git



Type this on Git bash while on the project repository (origin is the shortname for the repository and may be whatever you want)

```
git remote add origin REMOTE-URL
```

so in this case

```
git remote add origin
https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git
```

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git remote add origin https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

You can skip this step, the branch on the remote repository will be created anyway with the git push, you will do later.

```
git branch -M main
```

```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git branch -M main

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

You have two branches now, the local and the remote

```
MINGW64:/c/Users/xavier.florensa/documents/optix/plc2mqttclient_hivemq_v1_git
xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ git branch --all
* main
  remotes/origin/main

xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ |
```

You can list the remote repository connected to the local repository

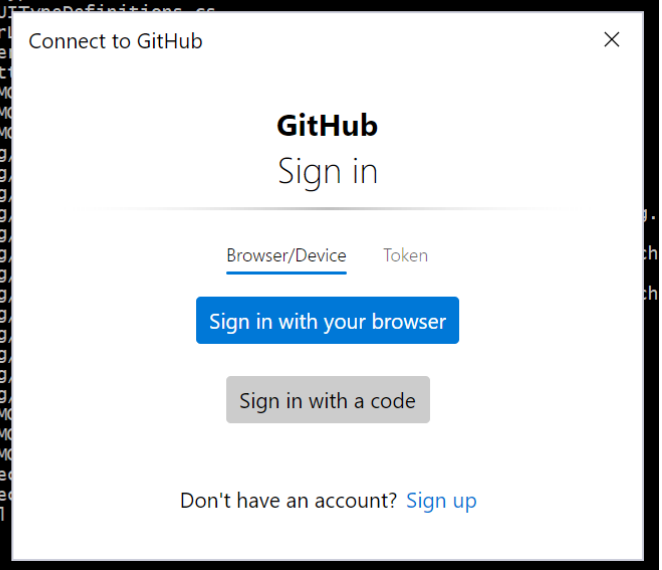
MINGW64:/c/Users/xavier.florensa/documents/optix/plc2mqttclient_hivemq_v1_git

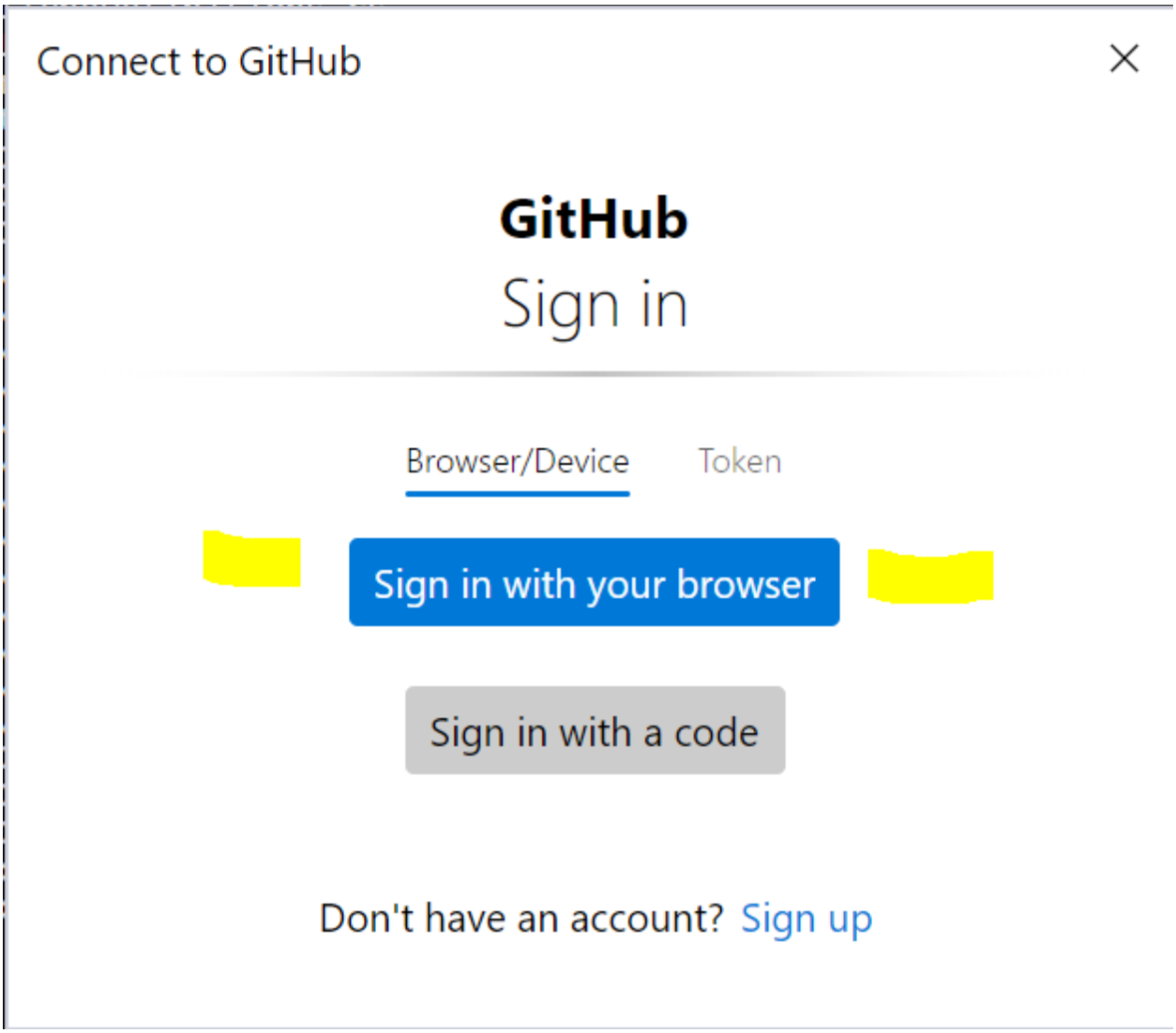
```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_
v1_git (main)
$ git remote -v
origin  https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git (fetch)
origin  https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git (push)

xavier.florensa@CATPROLP1003-23 MINGW64 ~/documents/optix/plc2mqttclient_hivemq_v1_git (main)
$ |
```

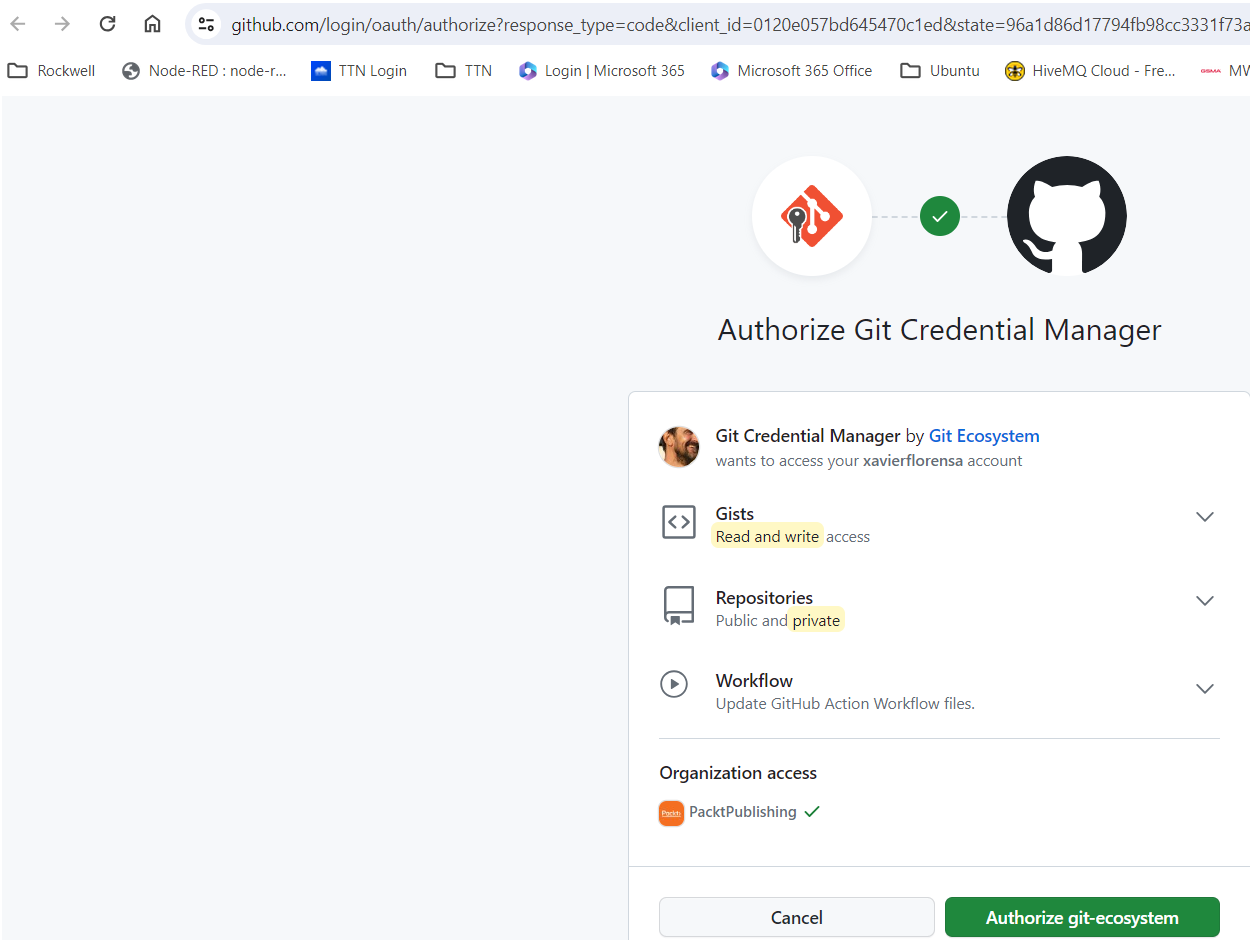
```
git push -u origin main
You need to autenthicate
```

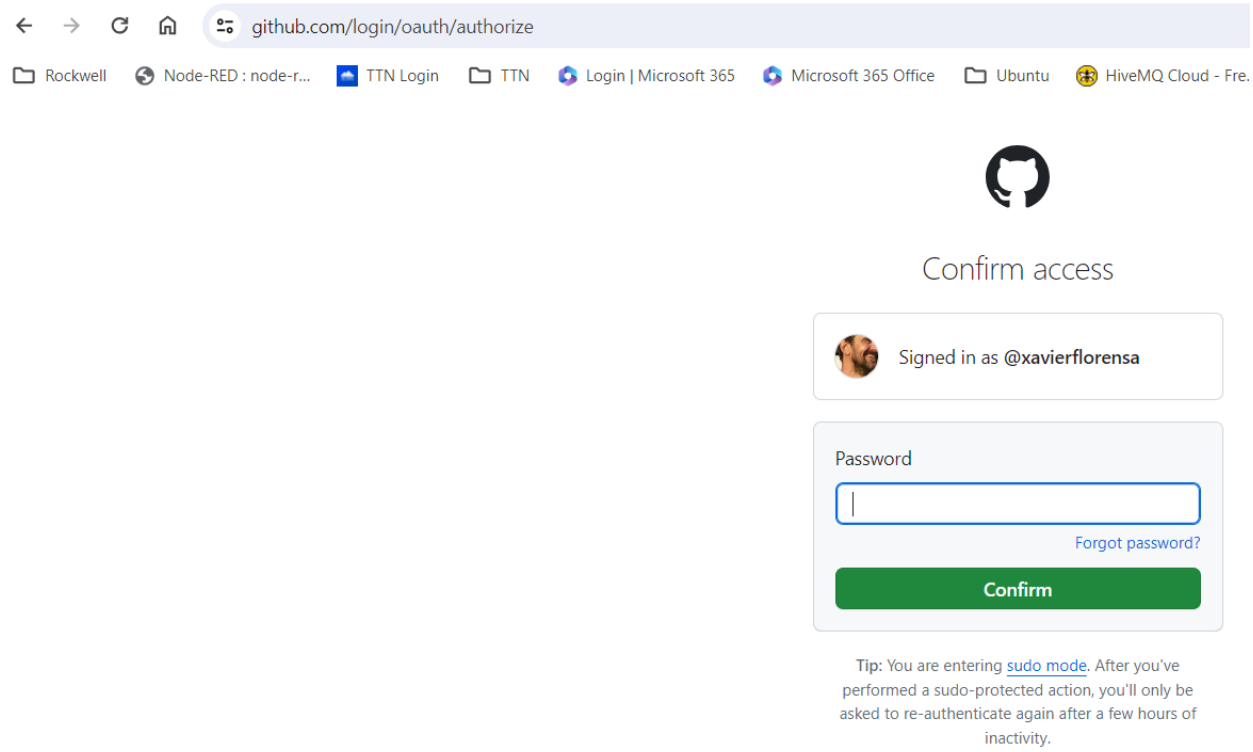
MINGW64:/c/Users/xavier.florensa/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git

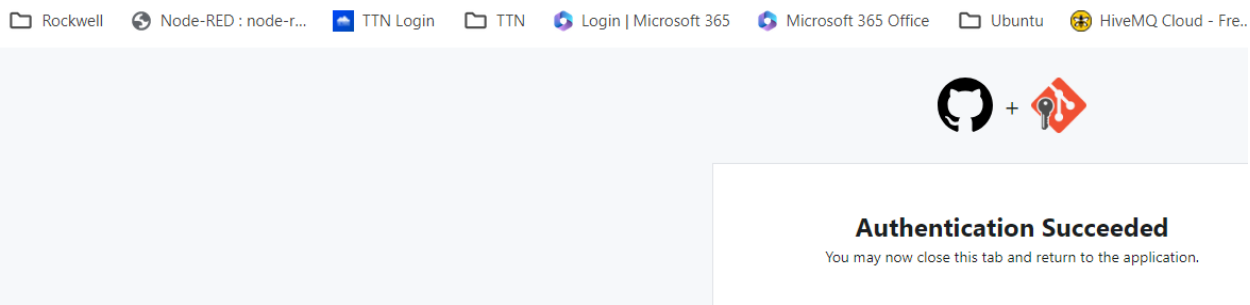
```
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient1_v1.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v0.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1.code-workspace
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.csproj
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.references
 create mode 100644 ProjectFiles/NetSolution/PLC2MQTTClient_HiveMQ_v1_Git.sln
 create mode 100644 ProjectFiles/NetSolution/Private/TypeConstants.cs
 create mode 100644 ProjectFiles/NetSolution/Private/UI
 create mode 100644 ProjectFiles/NetSolution/Publisher
 create mode 100644 ProjectFiles/NetSolution/Subscribe
 create mode 100644 ProjectFiles/NetSolution/bin/M2Mqt
 create mode 100644 ProjectFiles/NetSolution/bin/PLC2M
 create mode 100644 ProjectFiles/NetSolution/bin/PLC2M
 create mode 100644 ProjectFiles/NetSolution/bin/PLC2M
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/Debug
 create mode 100644 ProjectFiles/NetSolution/obj/PLC2M
 create mode 100644 ProjectFiles/NetSolution/obj/PLC2M
 create mode 100644 ProjectFiles/NetSolution/obj/PLC2M
 create mode 100644 ProjectFiles/NetSolution/obj/proje
 create mode 100644 ProjectFiles/NetSolution/obj/proje
 create mode 100644 ProjectFiles/UserDefinedModule.xml
 create mode 100644 ProjectFiles/prova35v0.ACD
 create mode 100644 prova35v0.ACD

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git remote add origin https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git branch -M main

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git push -u origin main
```



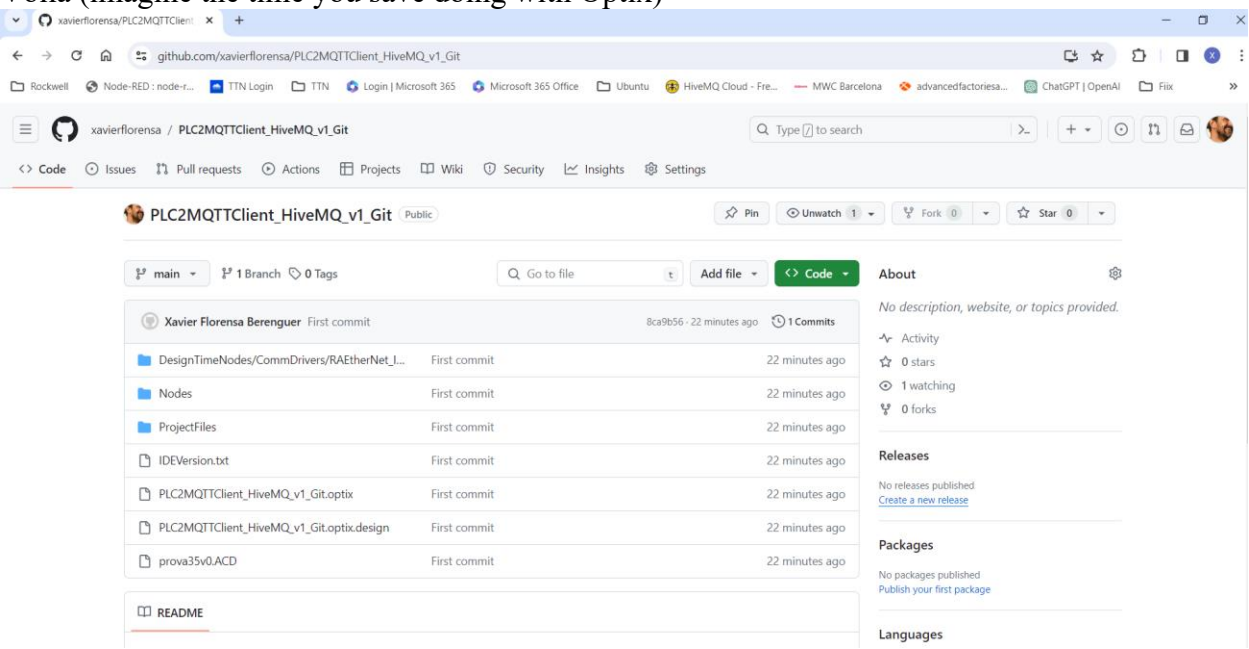

Connect to GitHub
GitHub
Sign in
Browser/Device
Token
Sign in with your browser
Sign in with a code
Don't have an account? Sign up


github.com/login/oauth/authorize?response_type=code&client_id=0120e057bd645470c1ed&state=96a1d86d17794fb98cc3331f73a
Rockwell
Node-RED : node-r...
TTN Login
TTN
Login | Microsoft 365
Microsoft 365 Office
Ubuntu
HiveMQ Cloud - Fre...
Authorize Git Credential Manager
Git Credential Manager by Git Ecosystem
wants to access your xavierflorensa account
Gists
Read and write access
Repositories
Public and private
Workflow
Update GitHub Action Workflow files.
Organization access
PacktPublishing
Cancel
Authorize git-ecosystem


github.com/login/oauth/authorize
Rockwell
Node-RED : node-r...
TTN Login
TTN
Login | Microsoft 365
Microsoft 365 Office
Ubuntu
HiveMQ Cloud - Fre.
Confirm access
Signed in as @xavierflorensa
Password
Forgot password?
Confirm
Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

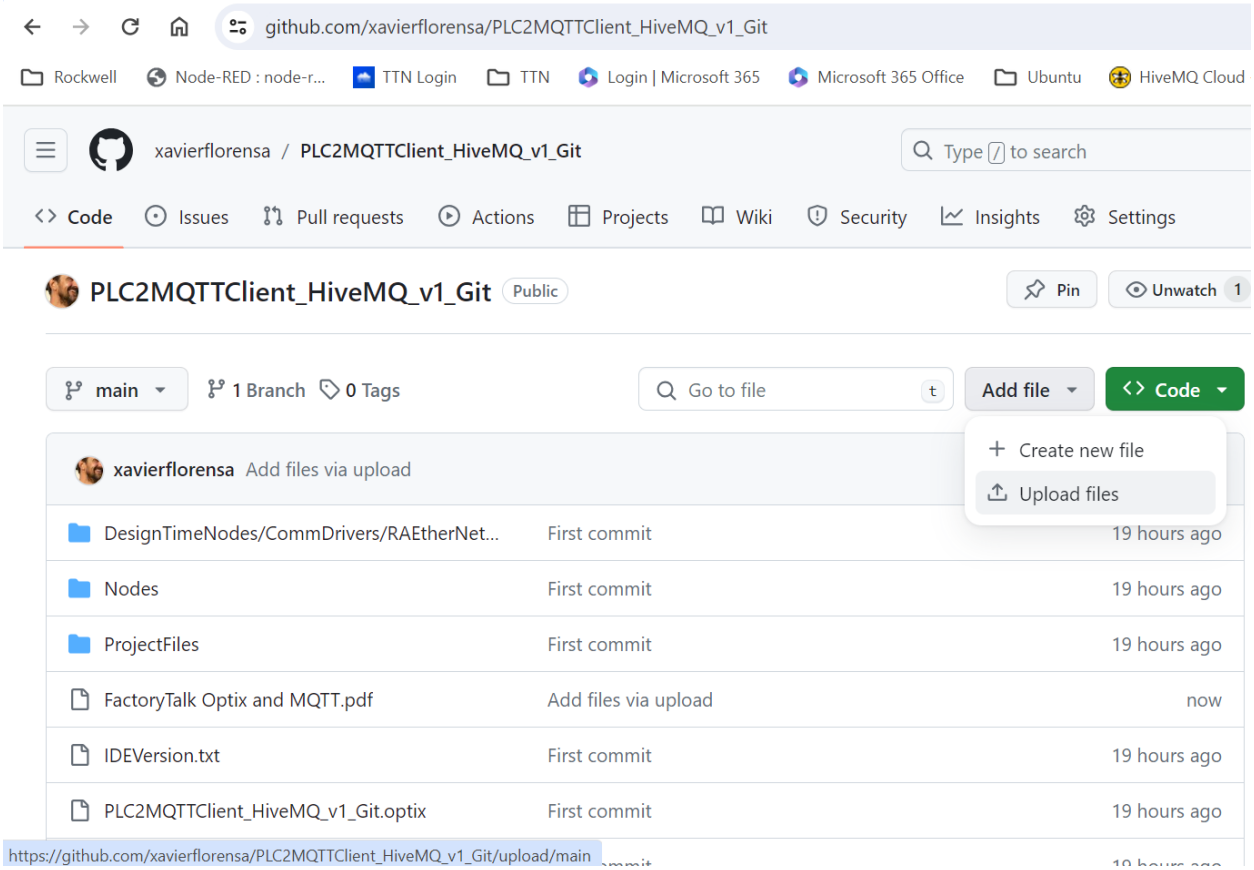


Now take a look at the Github repository
Voilà (imagine the time you save doing with Optix)

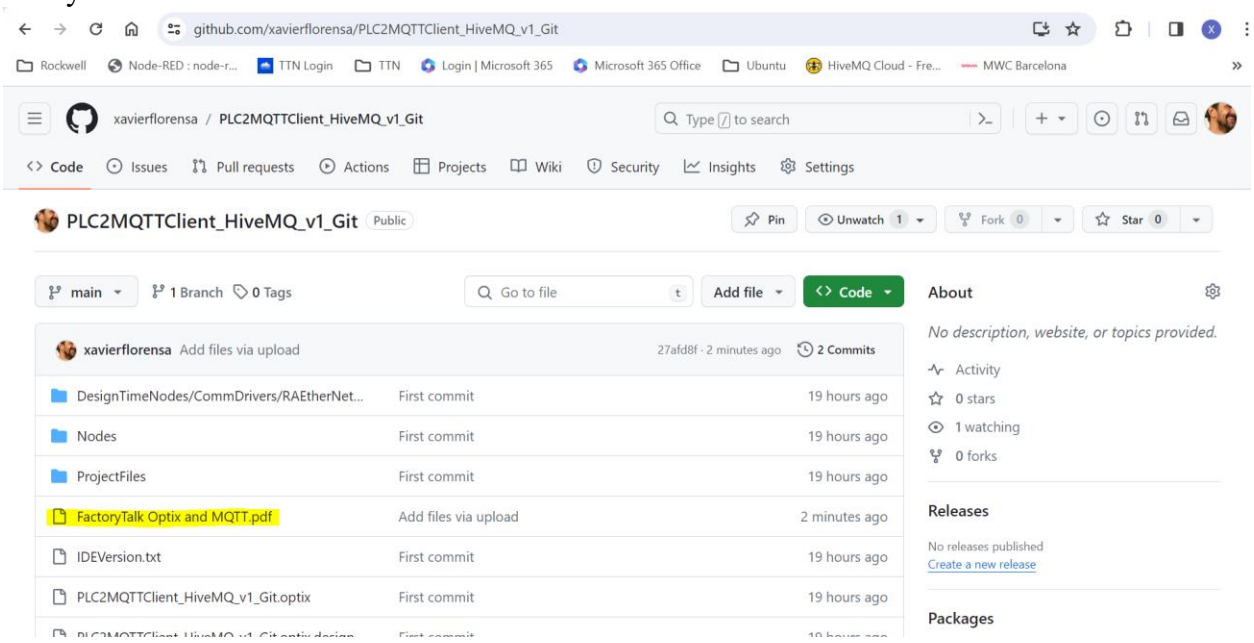



```
xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git branch -M main

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 93, done.
Counting objects: 100% (93/93), done.
Delta compression using up to 12 threads
Compressing objects: 100% (56/56), done.
Writing objects: 100% (93/93), 3.97 MiB | 1.84 MiB/s, done.
Total 93 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/xavierflorensa/PLC2MQTTClient_HiveMQ_v1_Git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

xavier.florensa@CATPROLP1003-23 MINGW64 ~/Documents/Optix/PLC2MQTTClient_HiveMQ_v1_Git (main)
$ |
```

Any file you have on the repository will be uploaded to Github, for instance a user manual, etc.
What if you have forgotten to add a file, no problem you can add from Github directly.
Just Add file / Upload file



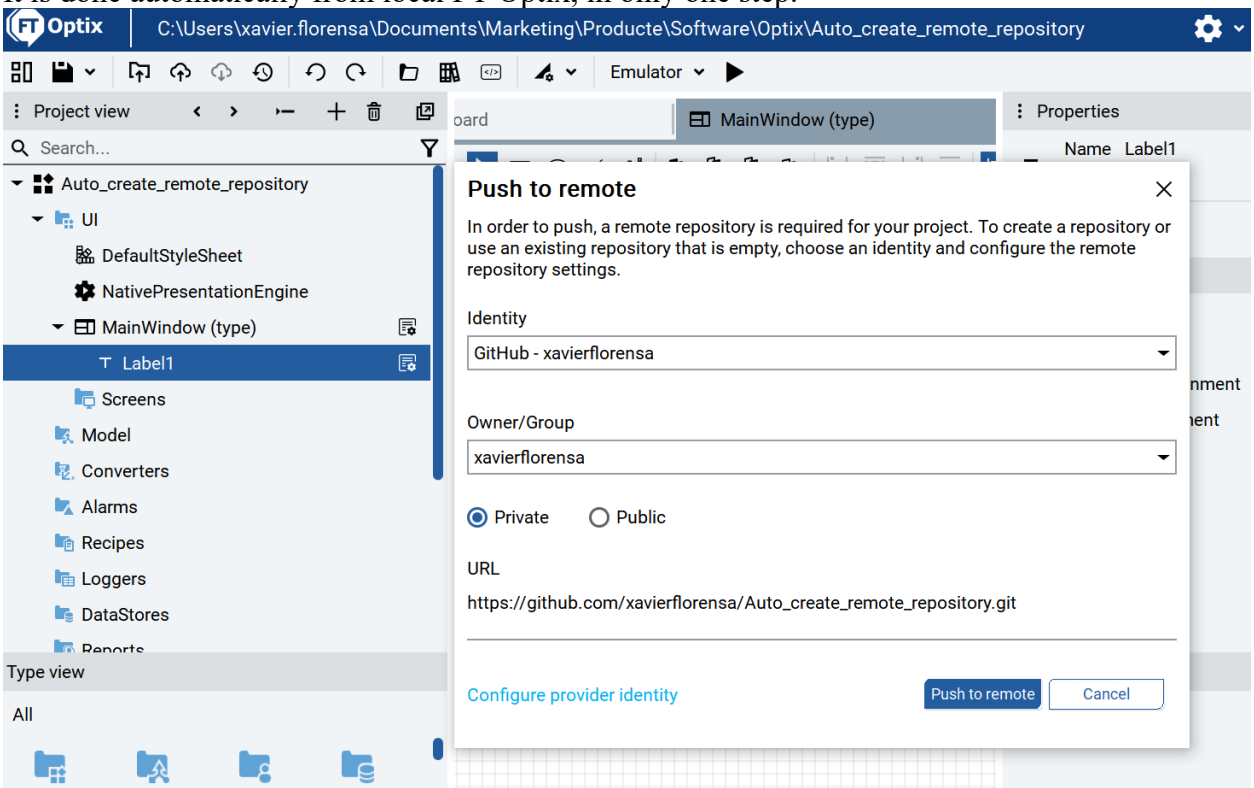
And you have it there

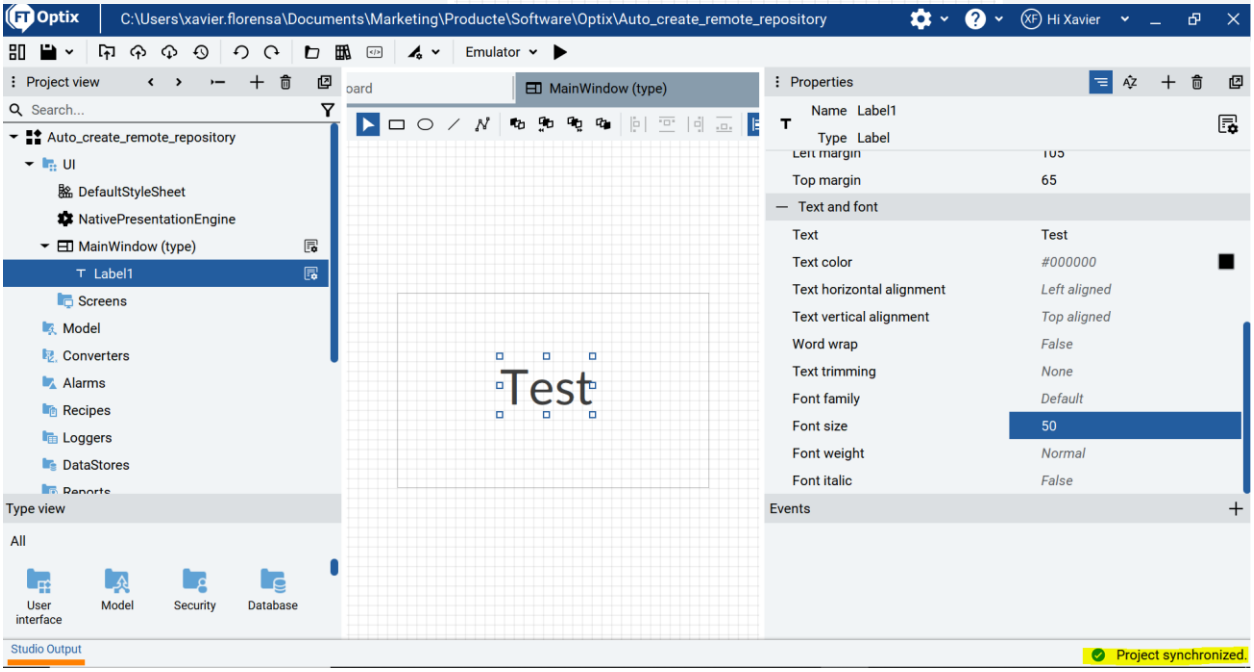


This is not possible if you have used Optix to push the repository

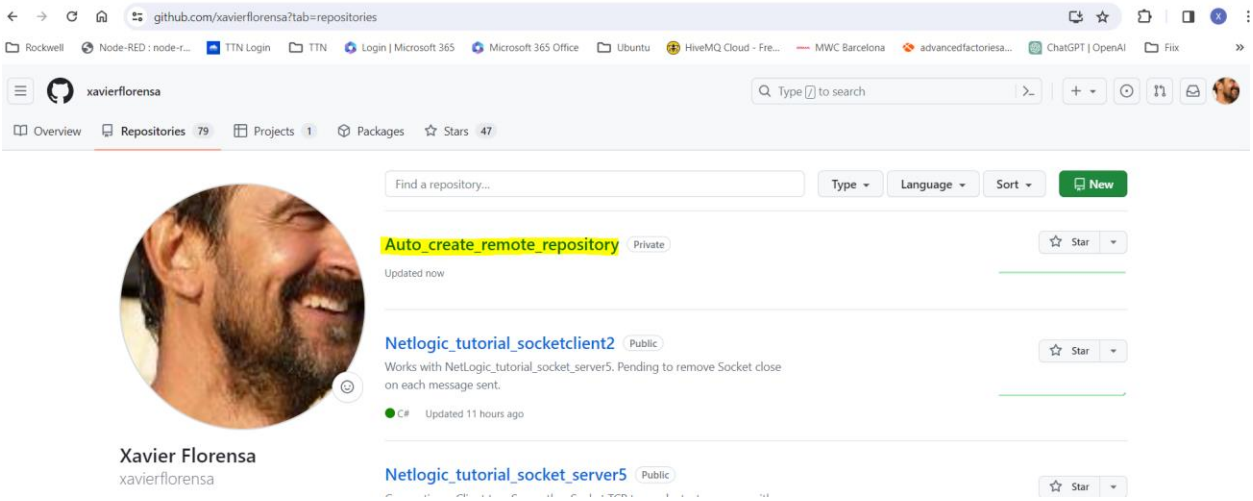
26.3. Remote repository creation with FT Optix version 1.3

Now you do not need to create the remote repository before performing a Remote Push.
It is done automatically from local FT Optix, in only one step.

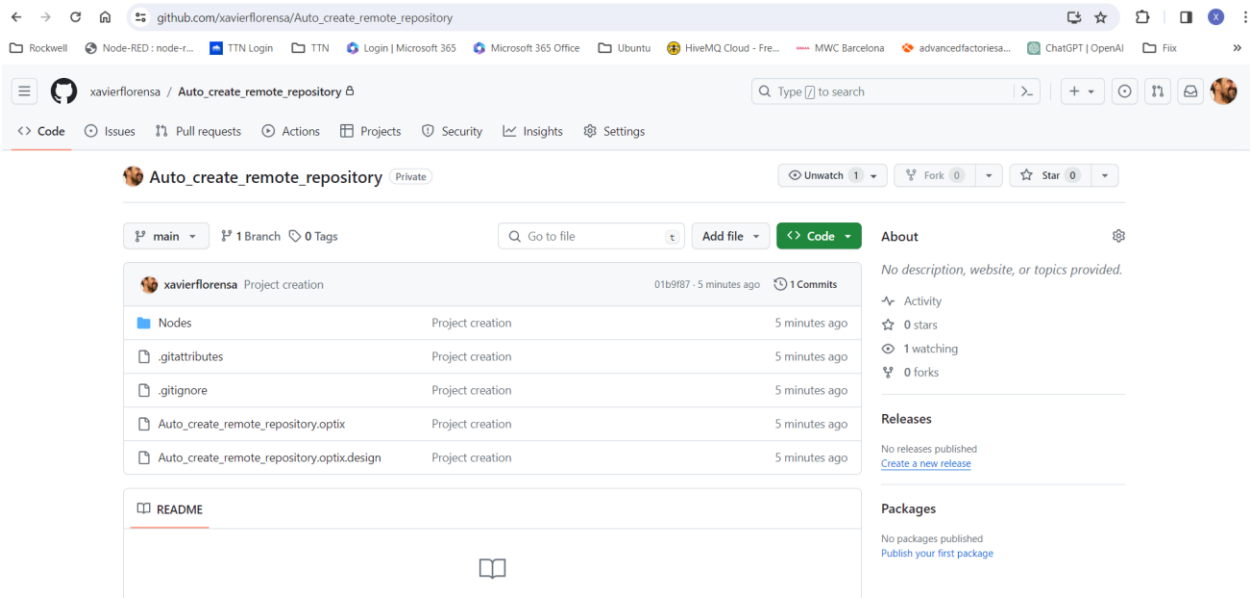




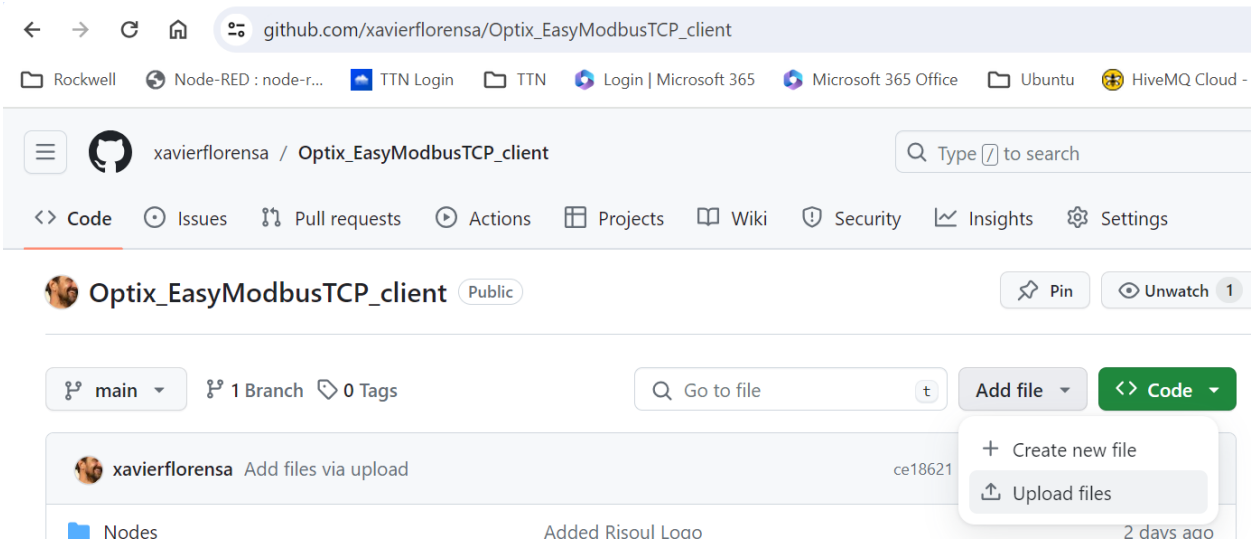
You can open your Github account



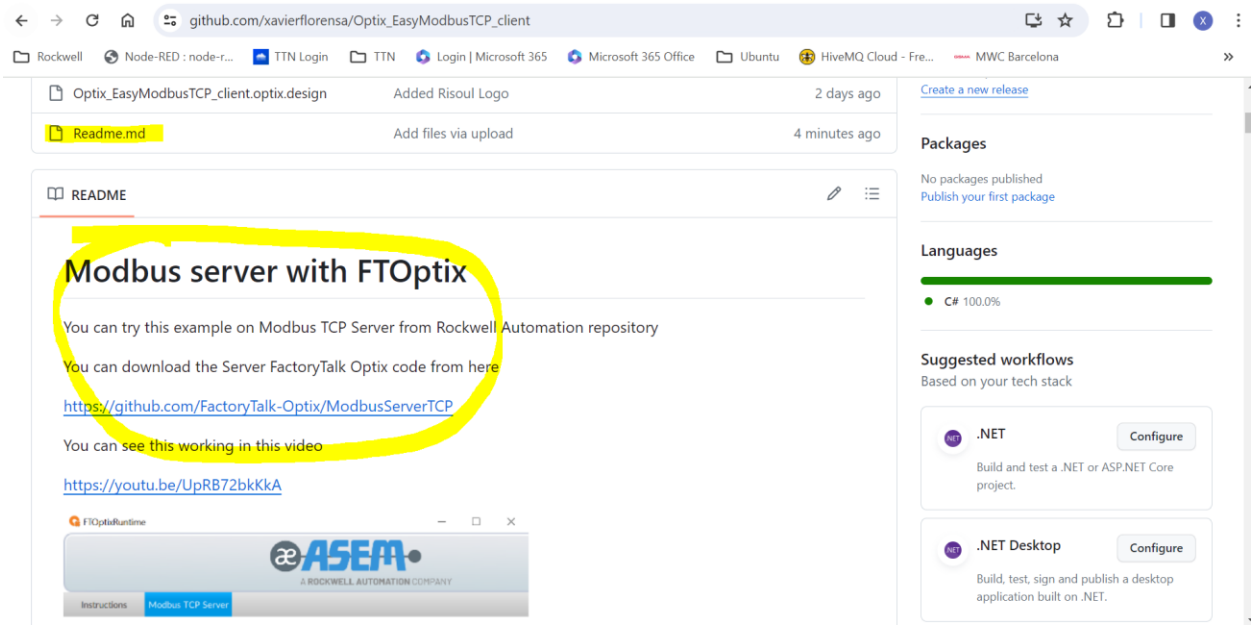
A new repository appeared



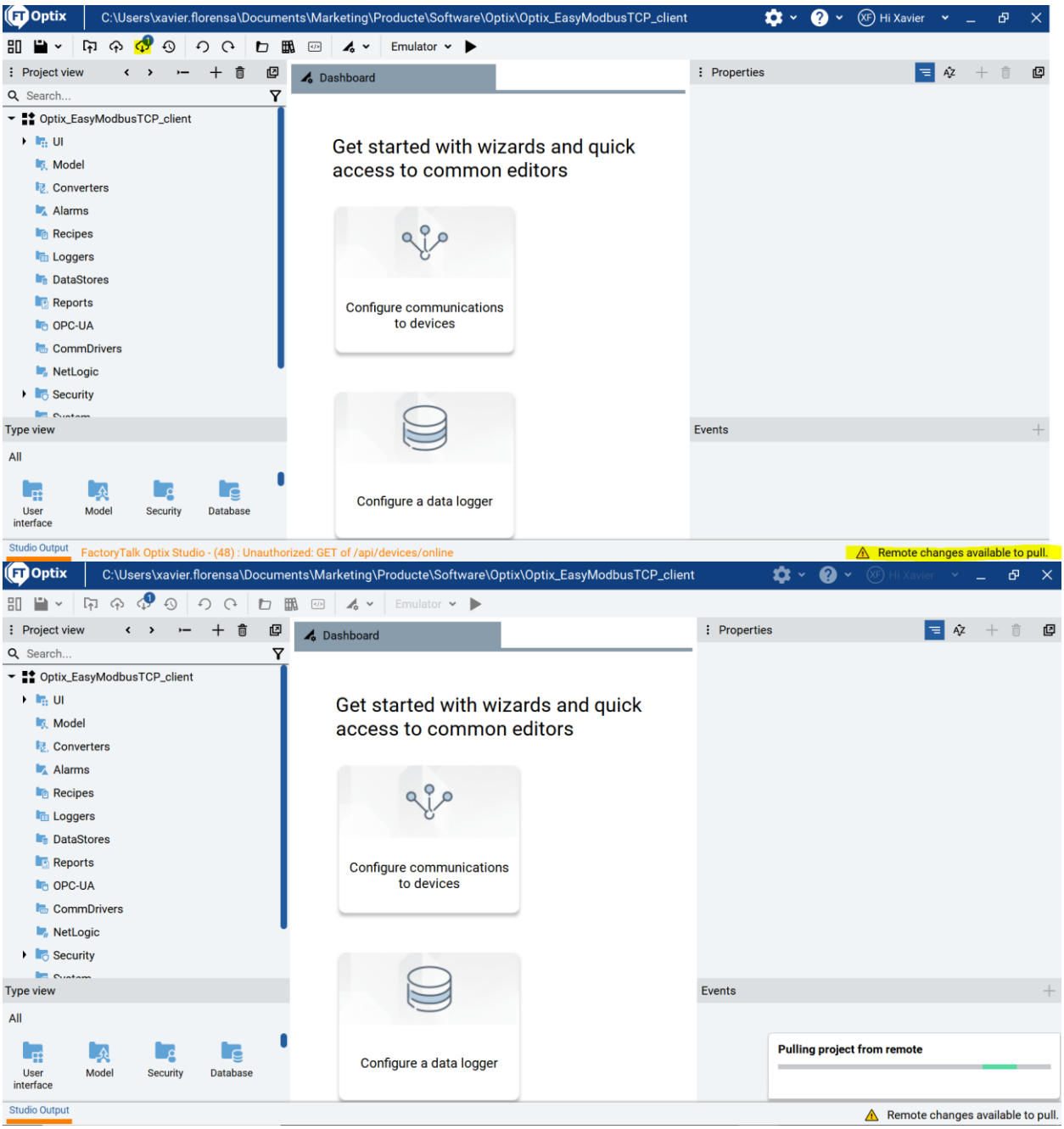
You can add a Readme.md file after the project is created.
Upload files



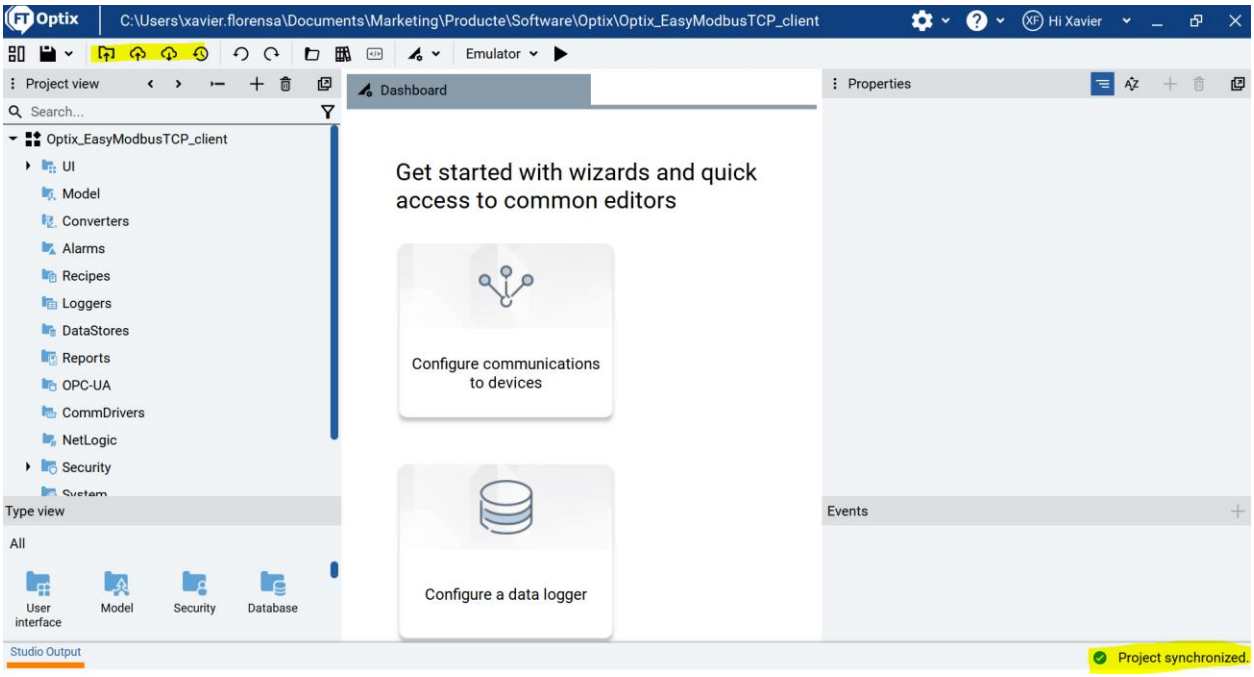
You have it



But your local project will not be syncronized
Then when you open the project locally on your computer
You will see this

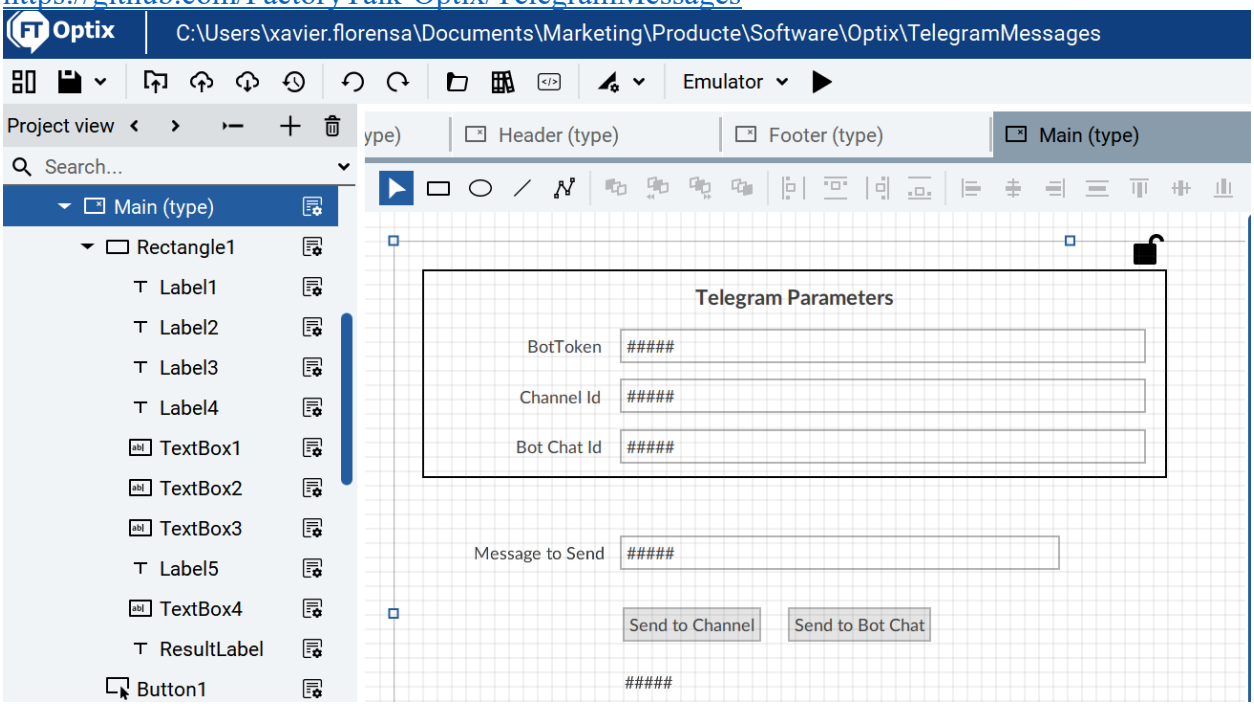


Now your project is synchronized



27. Sending Telegram messages

You can use the example from this repository
https://github.com/FactoryTalk-Optix/TelegramMessages

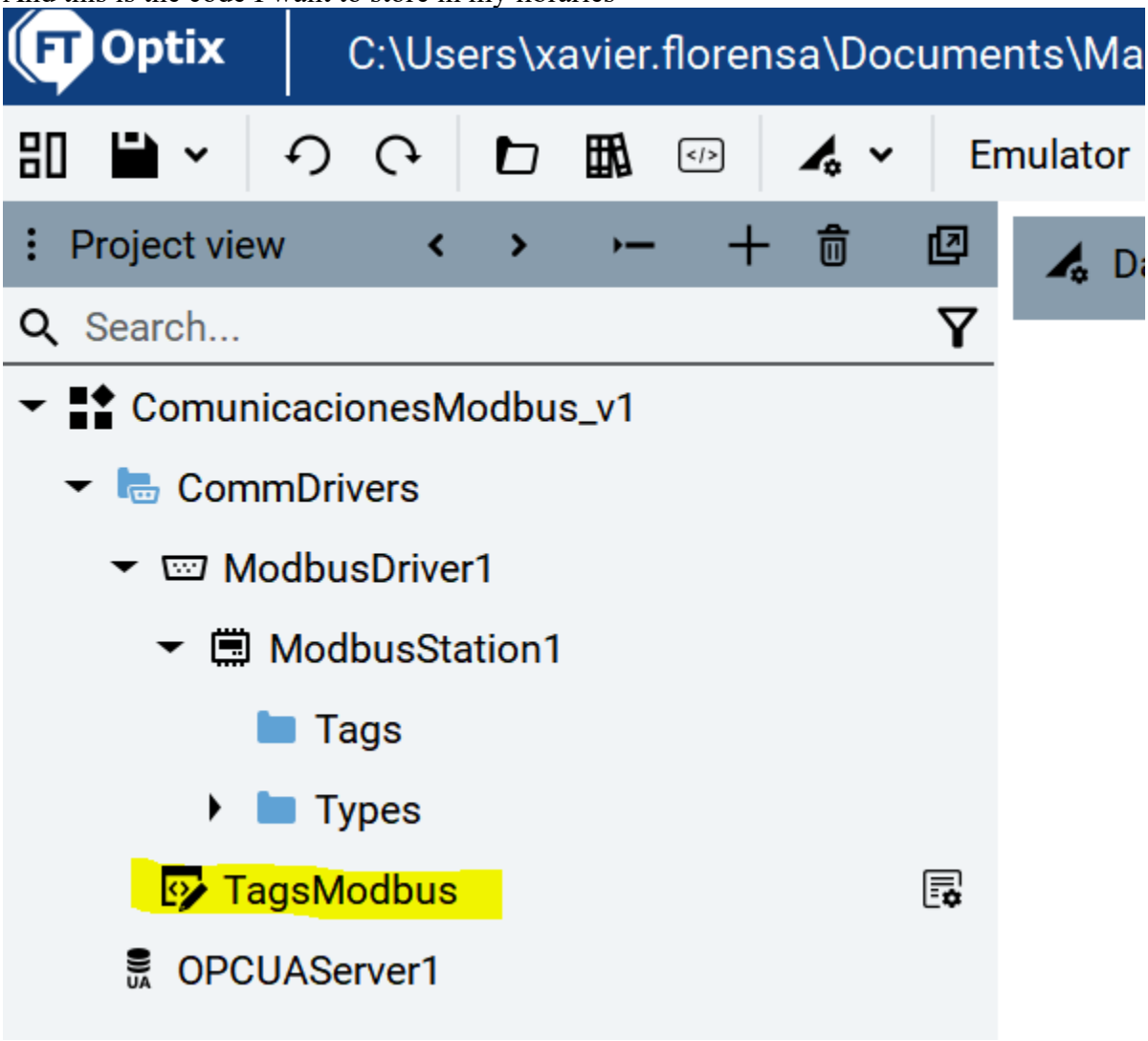


28. Creating, importing C# .Net libraries to your environment

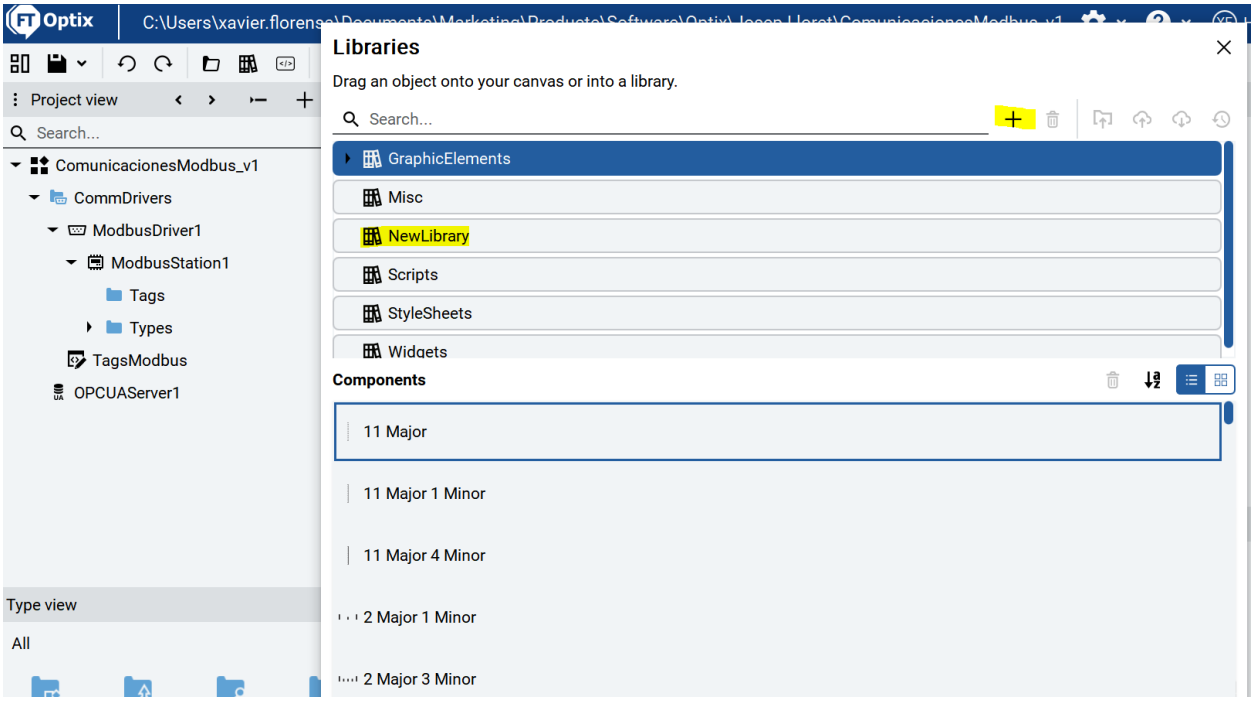
28.1. Design time libraries. Importing Modbus Tags from a CSV File.

You can find the code here
https://github.com/xavierflorensa/FactoryTalk_Optix_ModbusTagsImporter

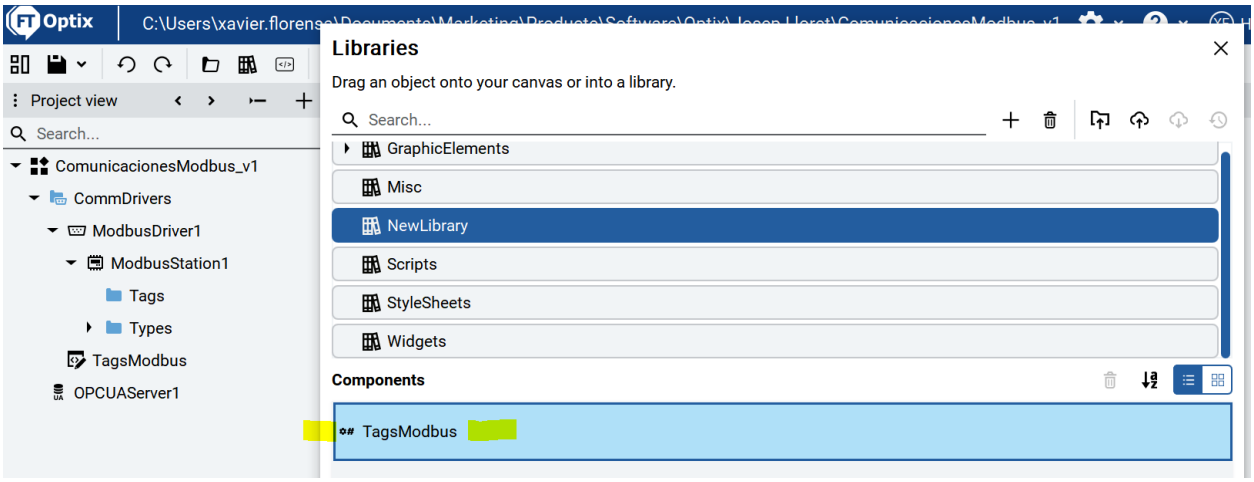
Let's import one library to our environment.
For instance this useful Modbus variable generator described on this video
https://www.youtube.com/watch?v=WLTmArGiwG8
From an example code I can take the .net code and store in my libraries collection to be used later
Let's open the example program
And this is the code I want to store in my libraries



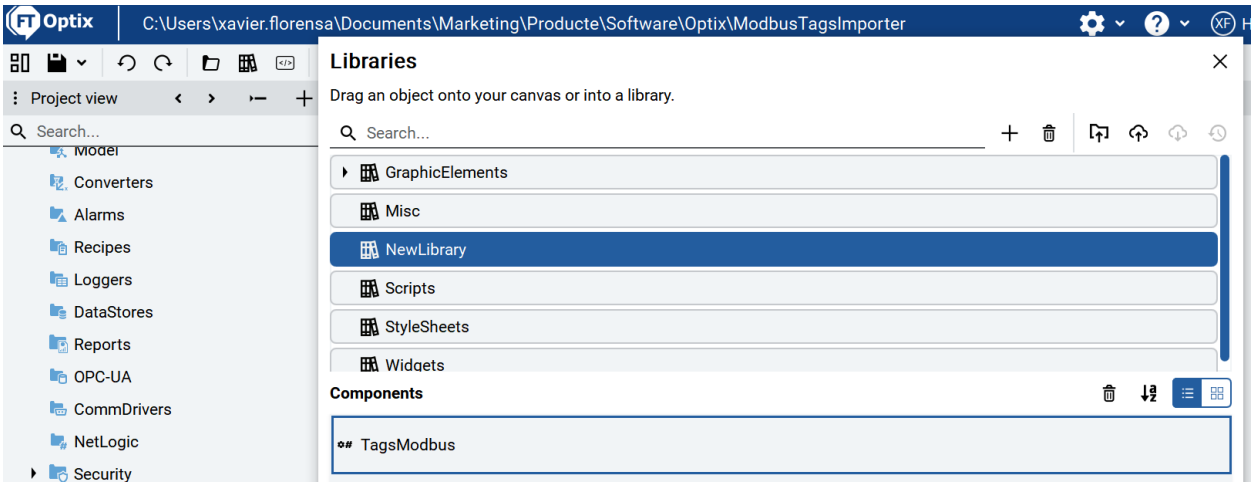
Let's open libraries and create a new library



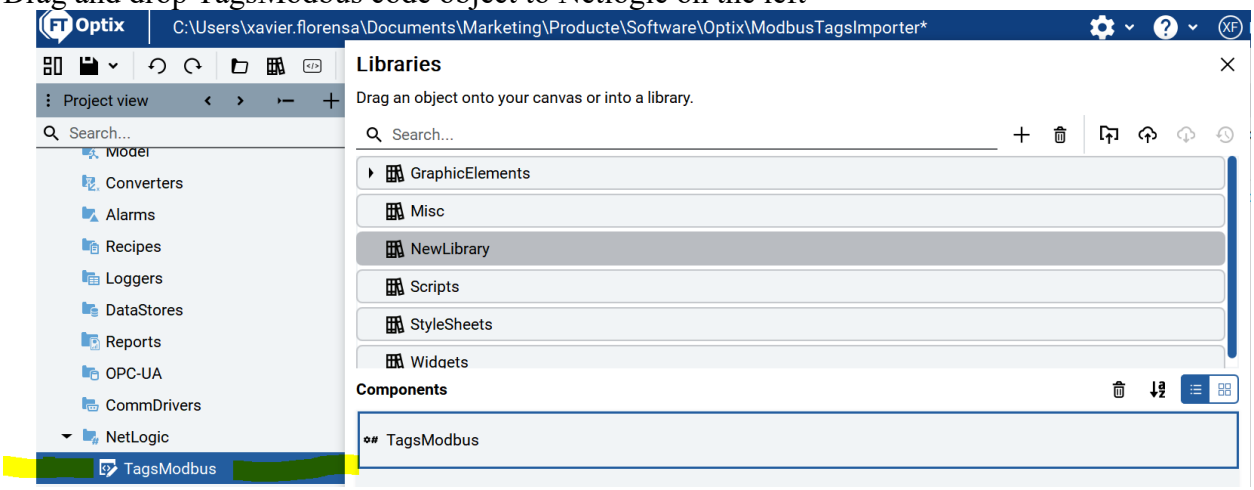
Then drag and drop the code object to the library
We have it



Now let's try to use on a new project
Create a new project



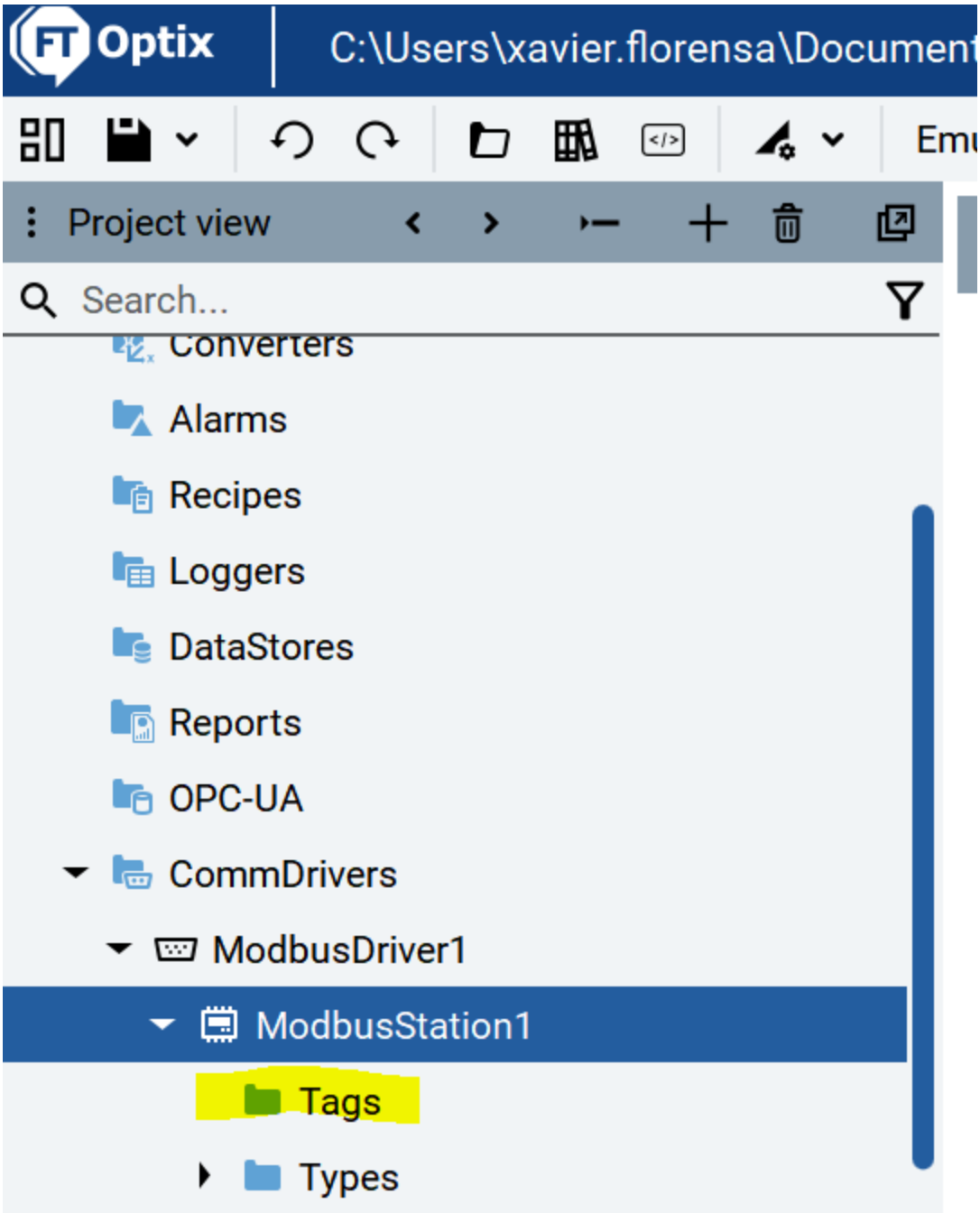
Drag and drop TagsModbus code object to Netlogic on the left



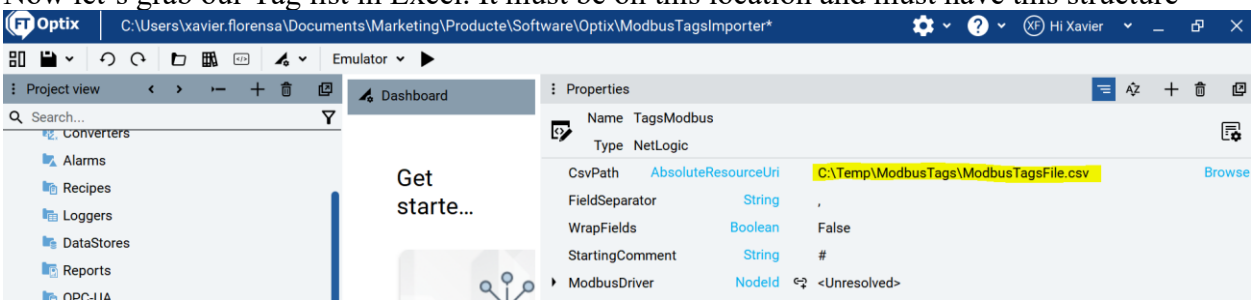
Now let's test it

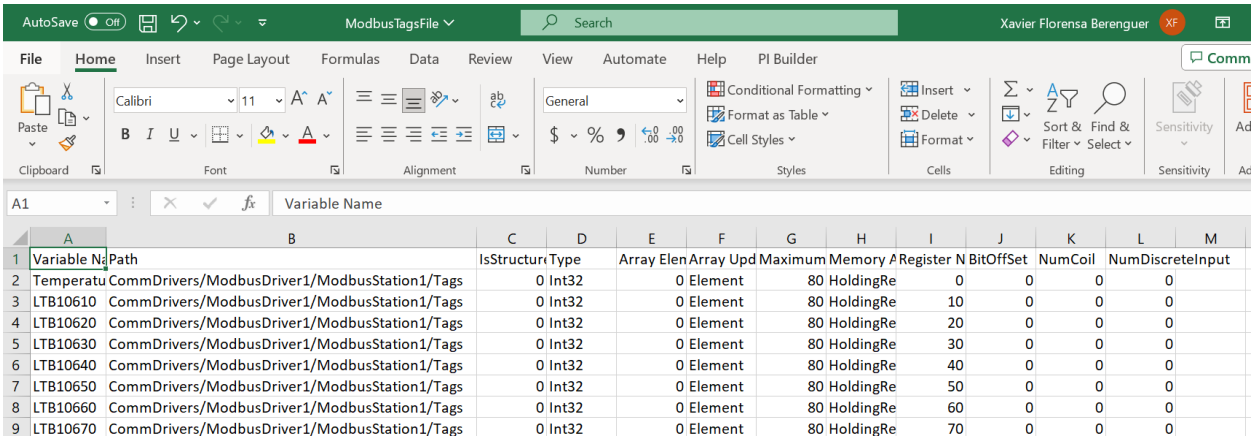
First of all let's create a Modbus driver with default names

The Tags folder is empty



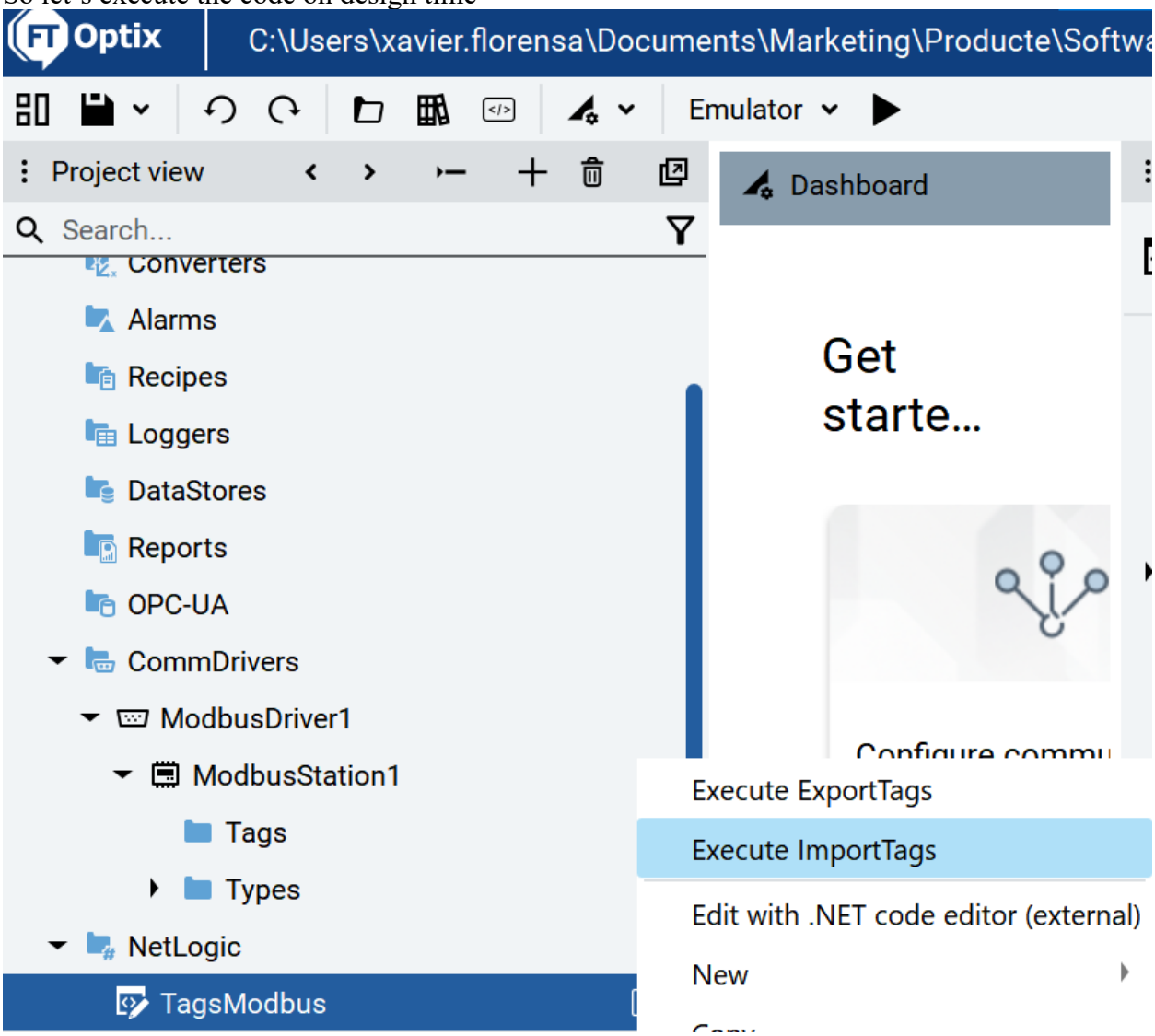
Now let's grab our Tag list in Excel. It must be on this location and must have this structure



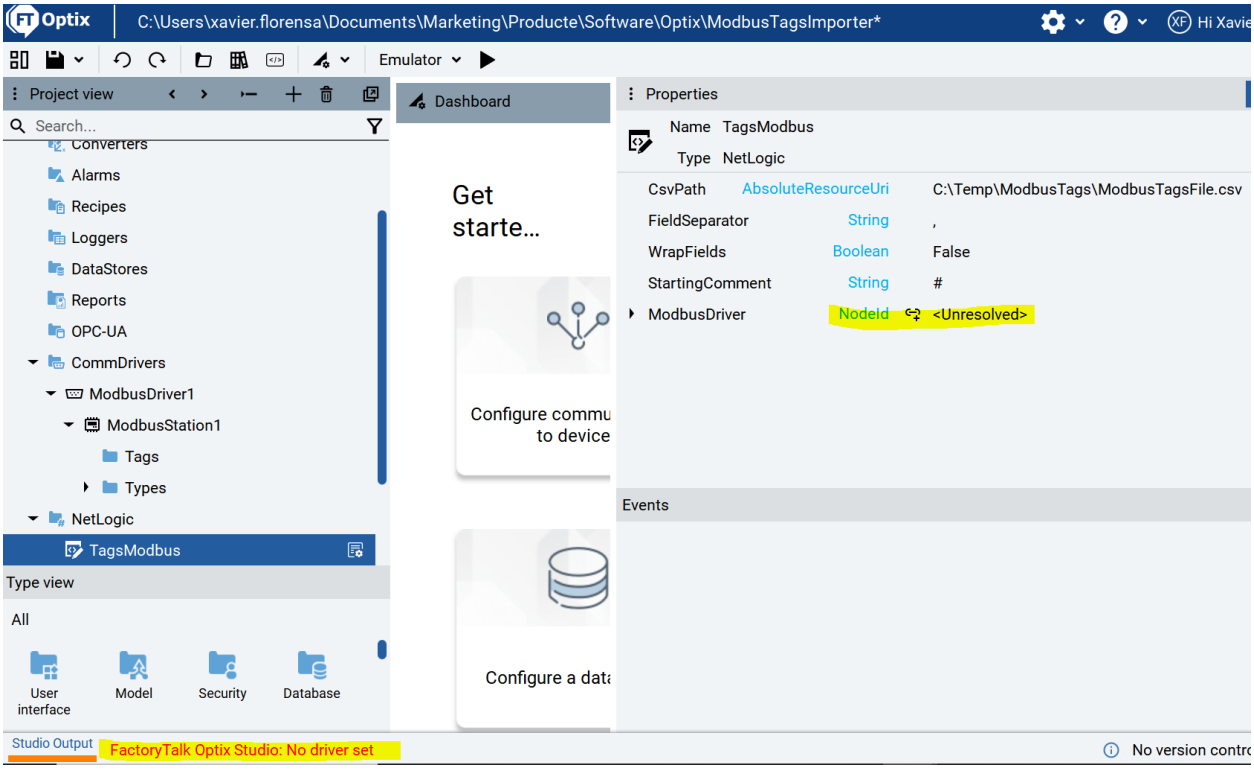


Variable Na	Path	IsStructur	Type	Array Elen	Array Upd	Maximum	Memory A	Register N	BitOffSet	NumCoil	NumDiscreteInput
Temperatu	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	0	0	0	0
LTB10610	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	10	0	0	0
LTB10620	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	20	0	0	0
LTB10630	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	30	0	0	0
LTB10640	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	40	0	0	0
LTB10650	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	50	0	0	0
LTB10660	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	60	0	0	0
LTB10670	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	0	Element	80	HoldingRe	70	0	0	0

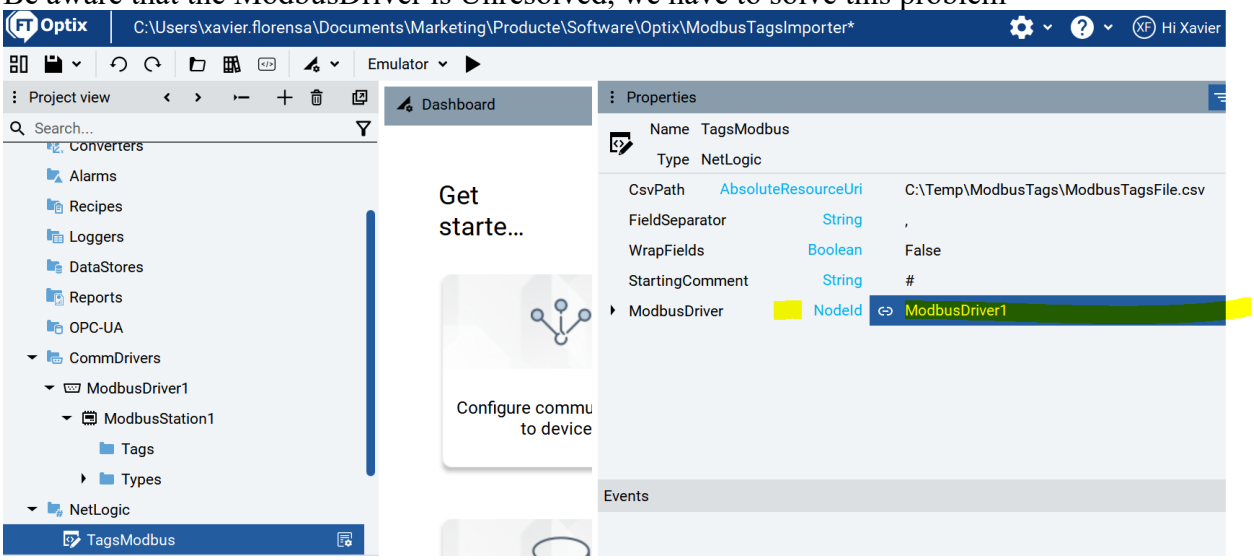
So let's execute the code on design time



We get an error message



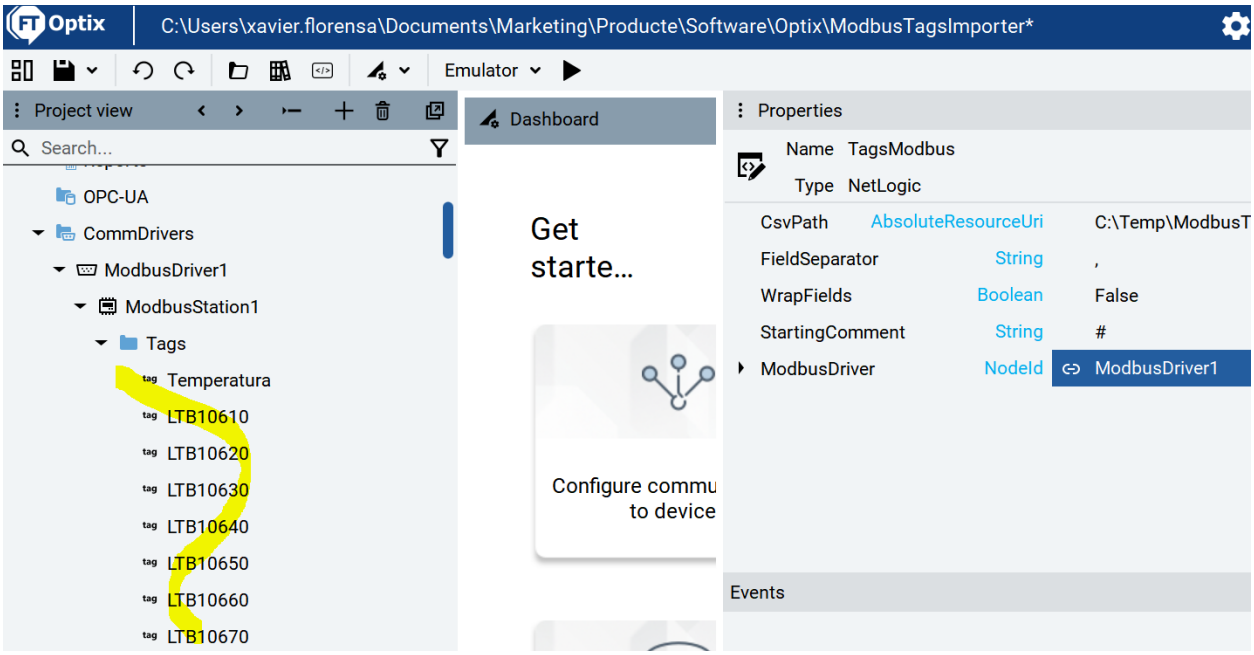
Be aware that the ModbusDriver is Unresolved, we have to solve this problem



Now let's try to execute again

Be sure to close the excel file. If you have the file open, Optix will complain and will not do the task

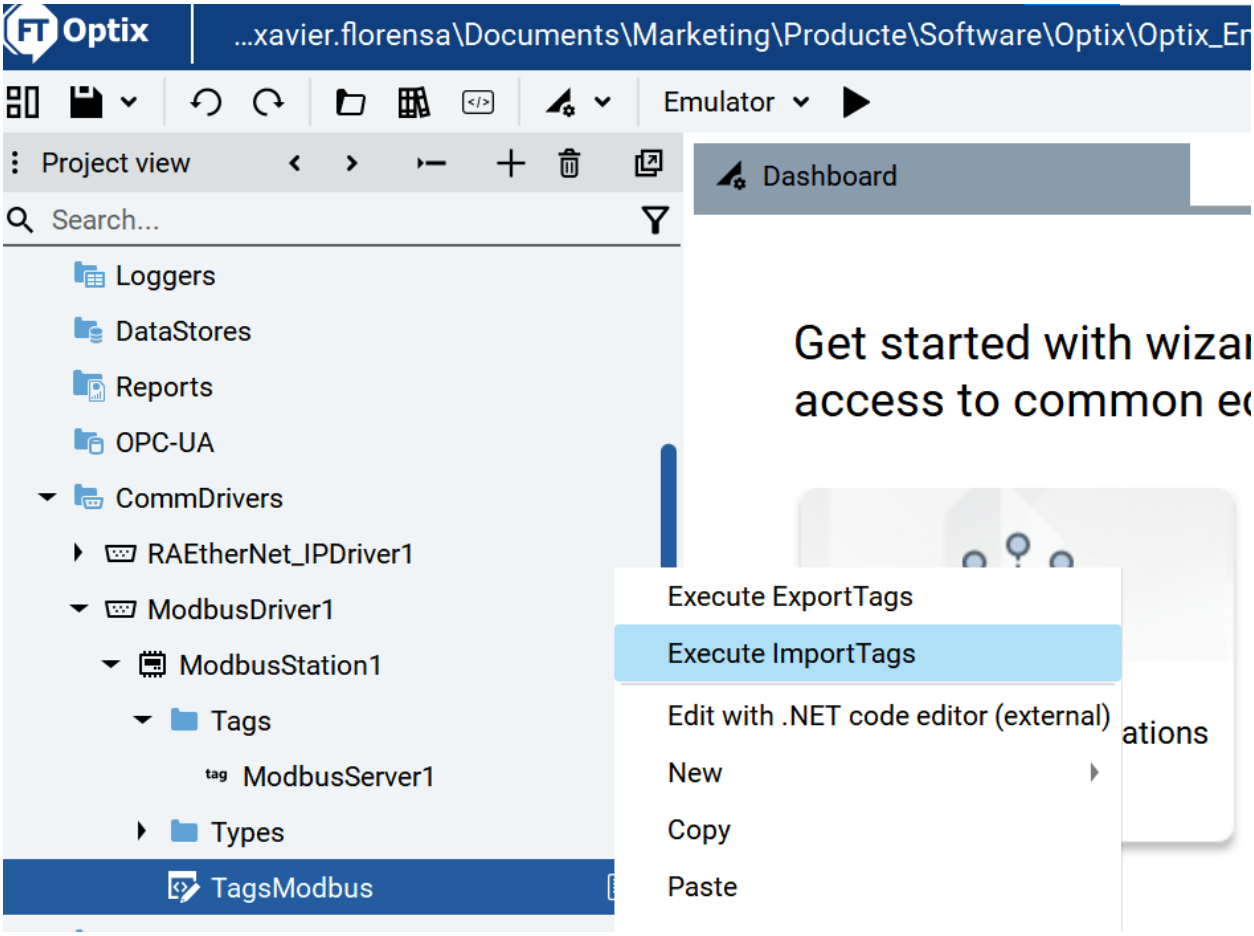
Voilà



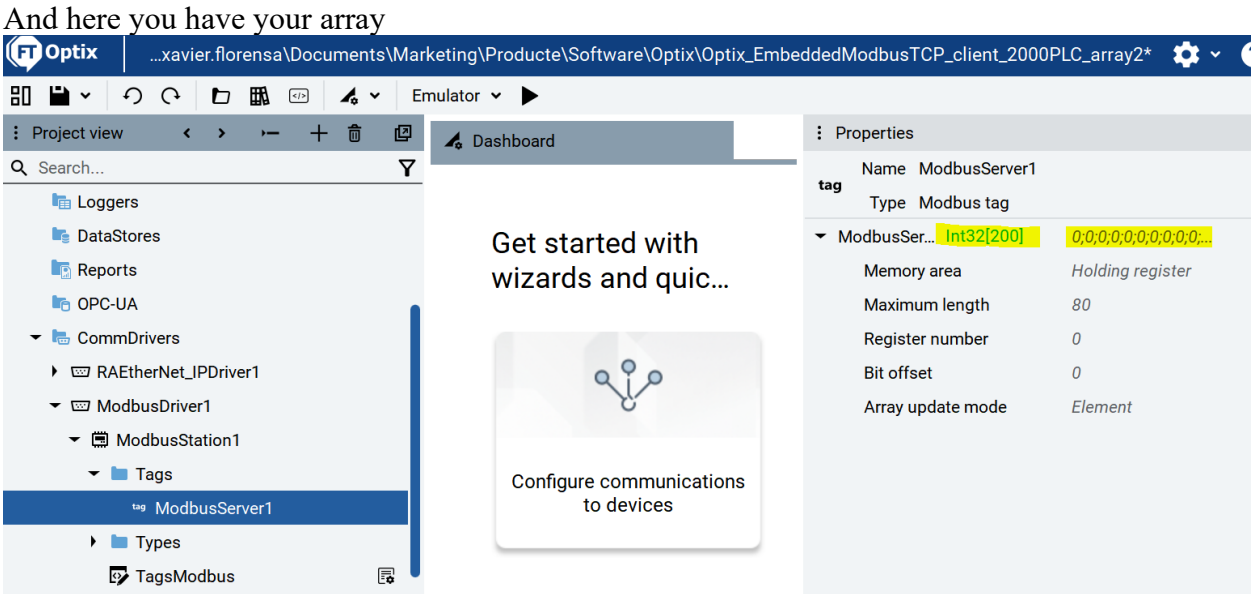
You can even drag and drop the code into Modbus driver instead of into Netlogic
Imagine you want to import an array of Tags
Then you have to set up your CSV file this way:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Variable Name	Path	IsStructure	Type	Array Elements	Array Update Mode	Maximum	Memory Area	Register N	BitOffSet	NumCoil	NumDiscreteInput	
2	ModbusServer1	CommDrivers/ModbusDriver1/ModbusStation1/Tags	0	Int32	200	Element	80	HoldingRegister	0	0	0	0	

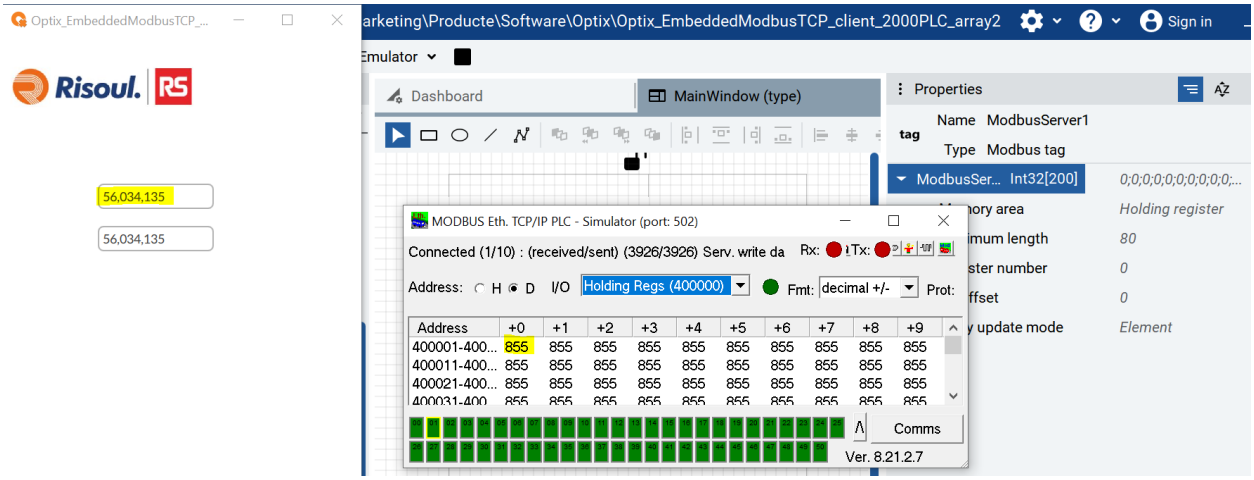
Then execute the ImportTags



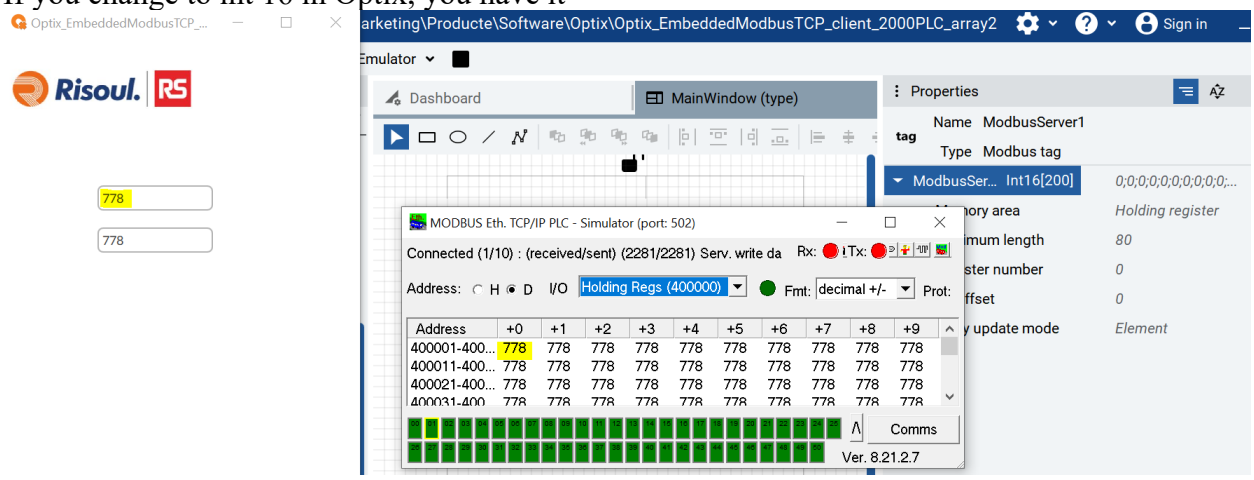
And here you have your array



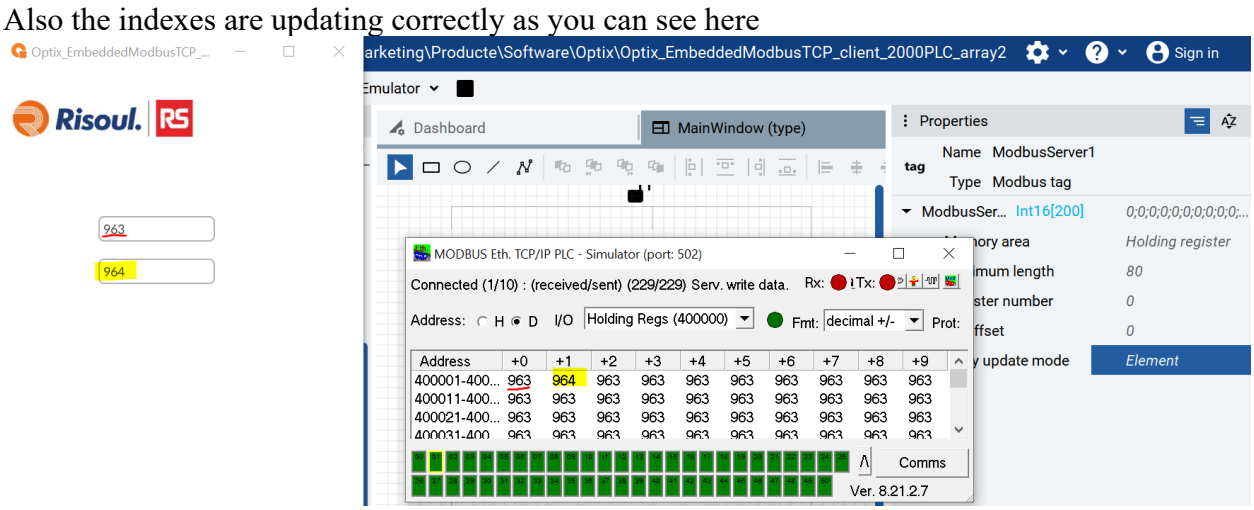
If you try to execute the project you will not see the same number on the Modbus Device and in Optix, since the Modbus device is Int16 and in Optix we have selected Int32.



If you change to int 16 in Optix, you have it



Also the indexes are updating correctly as you can see here

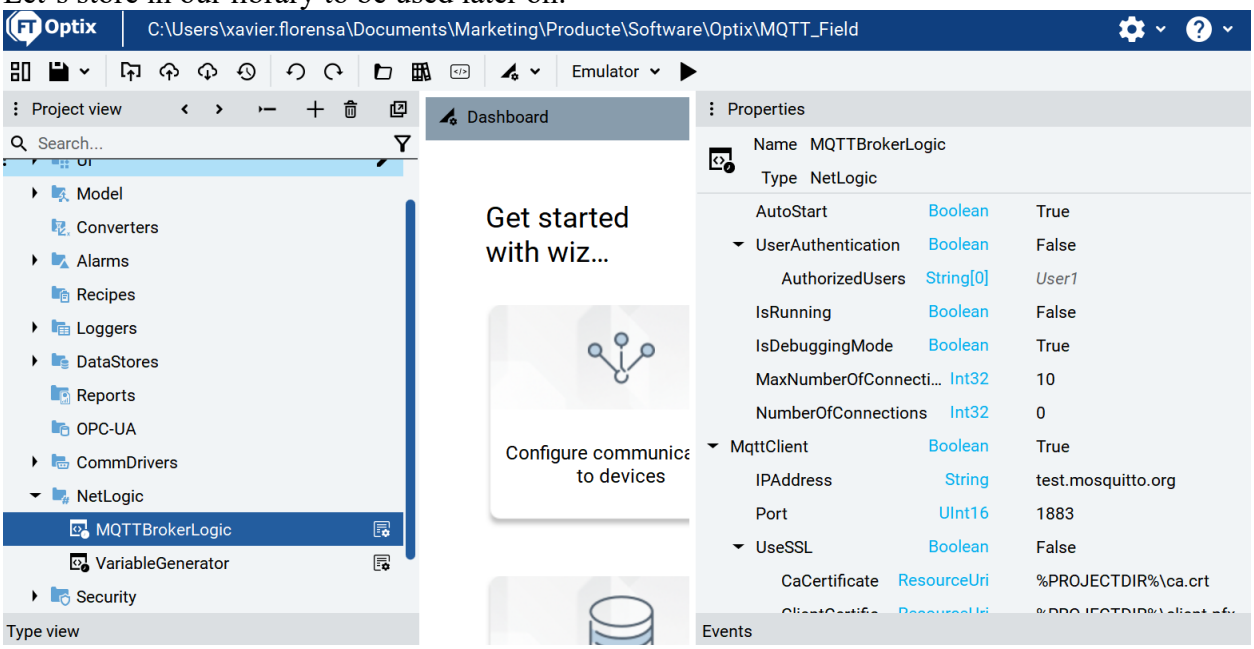


You can find the code here
https://github.com/xavierflorensa/FactoryTalk_Optix_ModbusTagsImporter

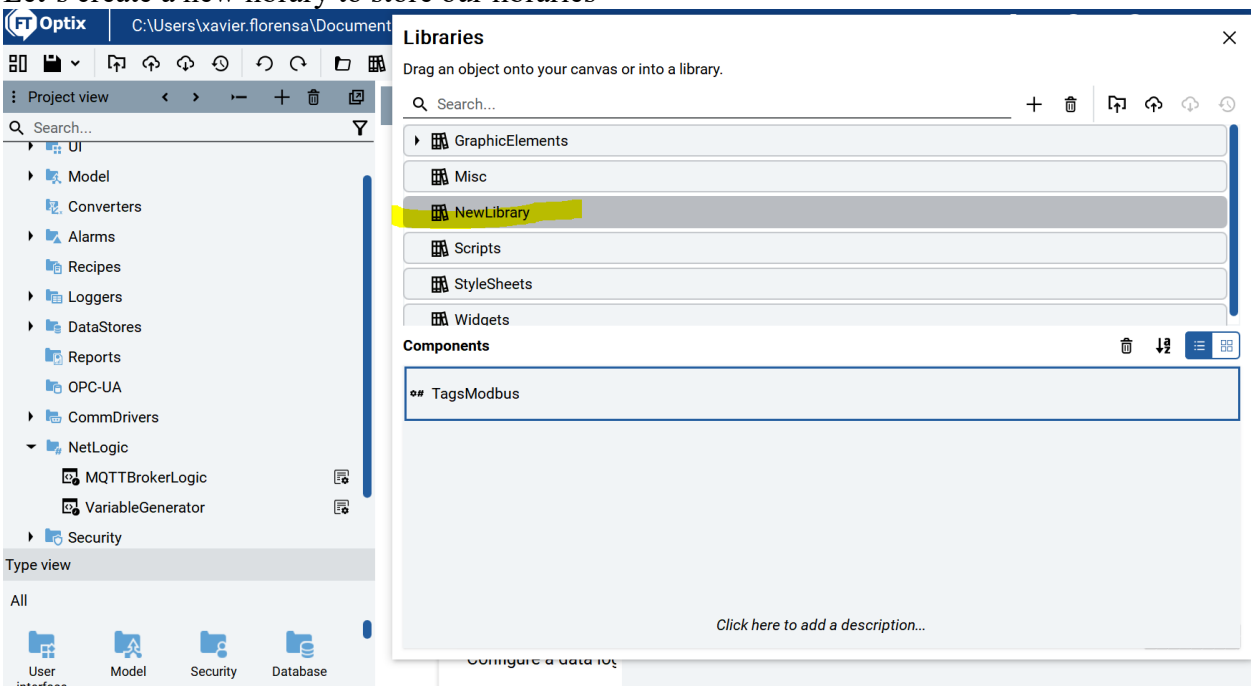
28.2. Runtime libraries

I realized that on the MQTT examples on the Rockwell automation repository they use the same code

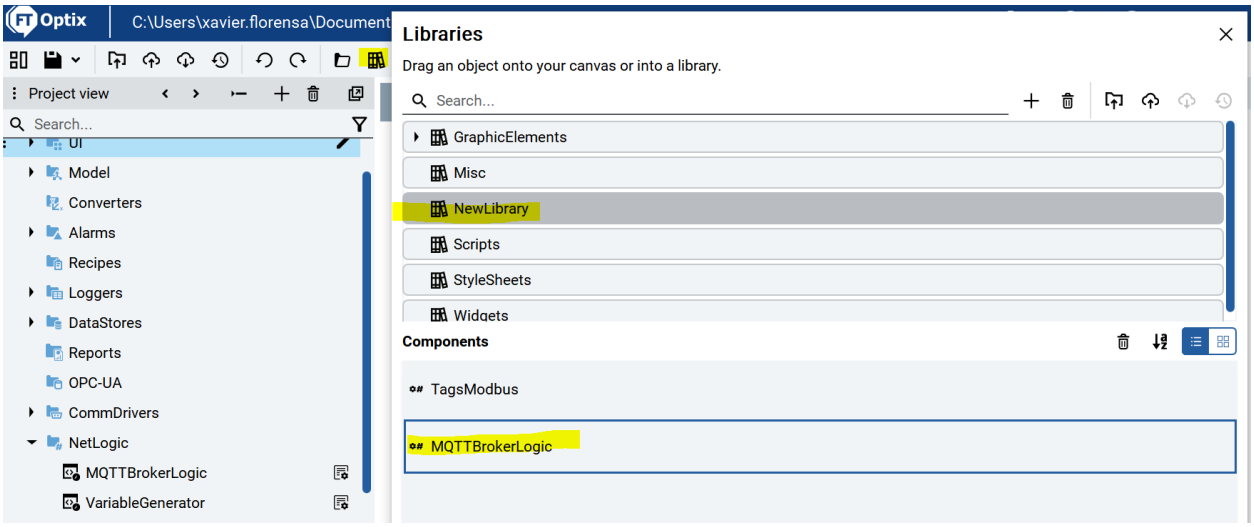
Let's store in our library to be used later on.



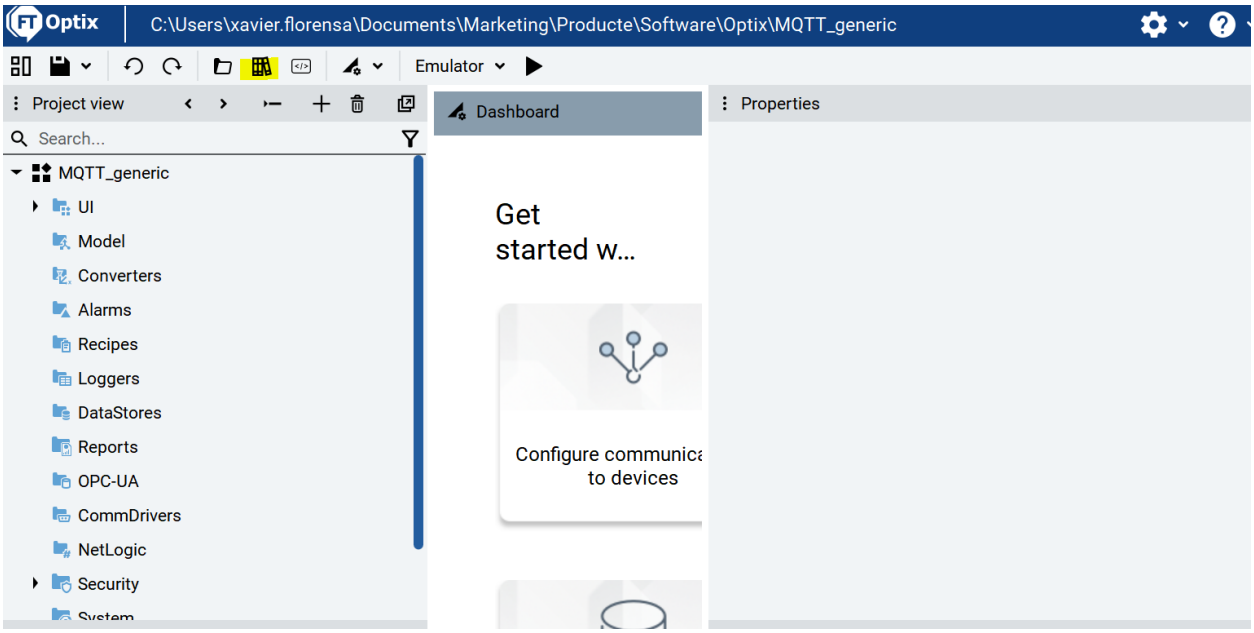
Let's create a new library to store our libraries



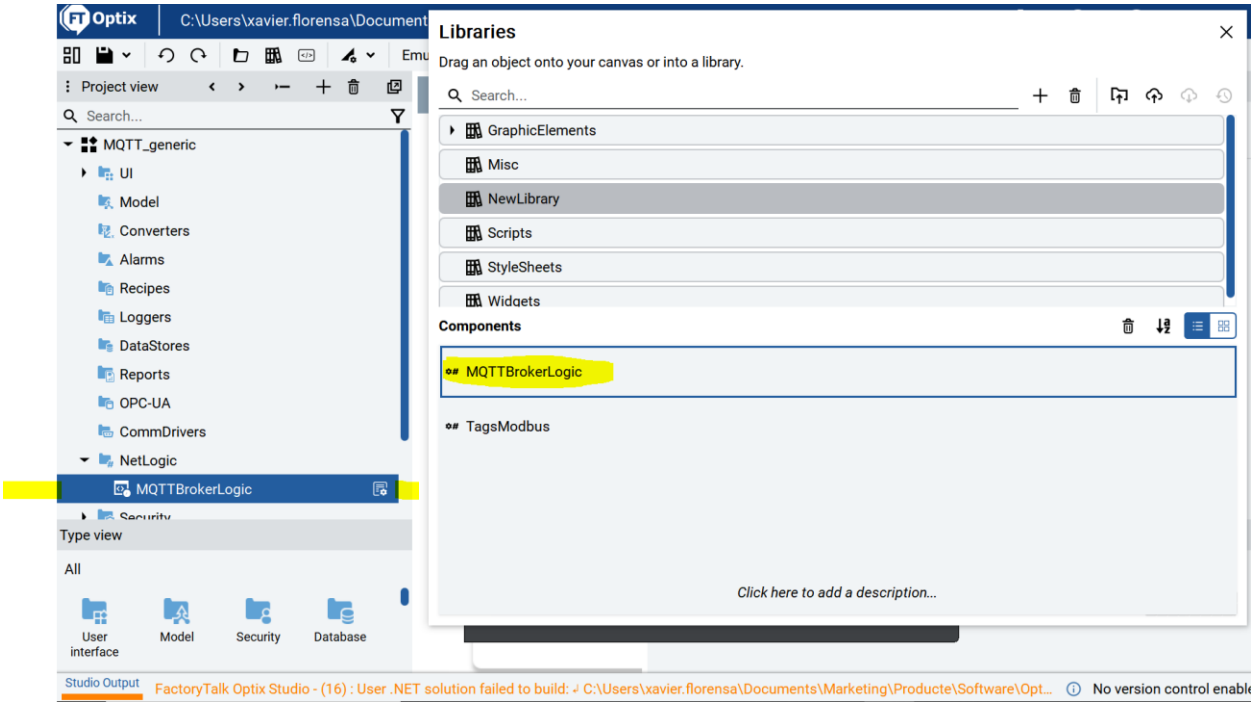
Drag and drop the library



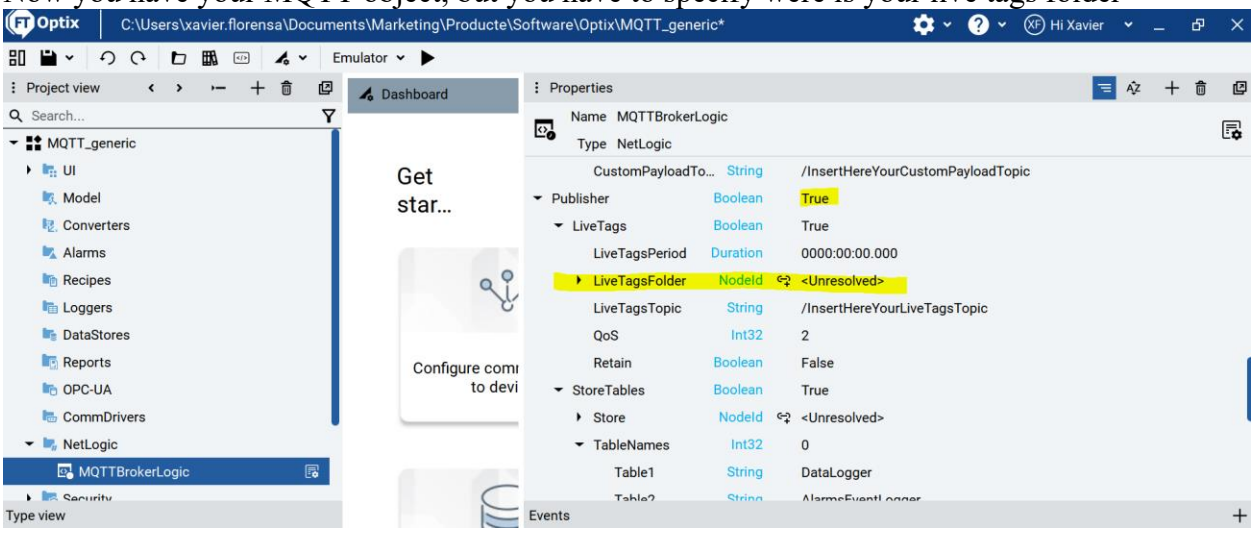
You can close the project and open a new one
Go to libraries.



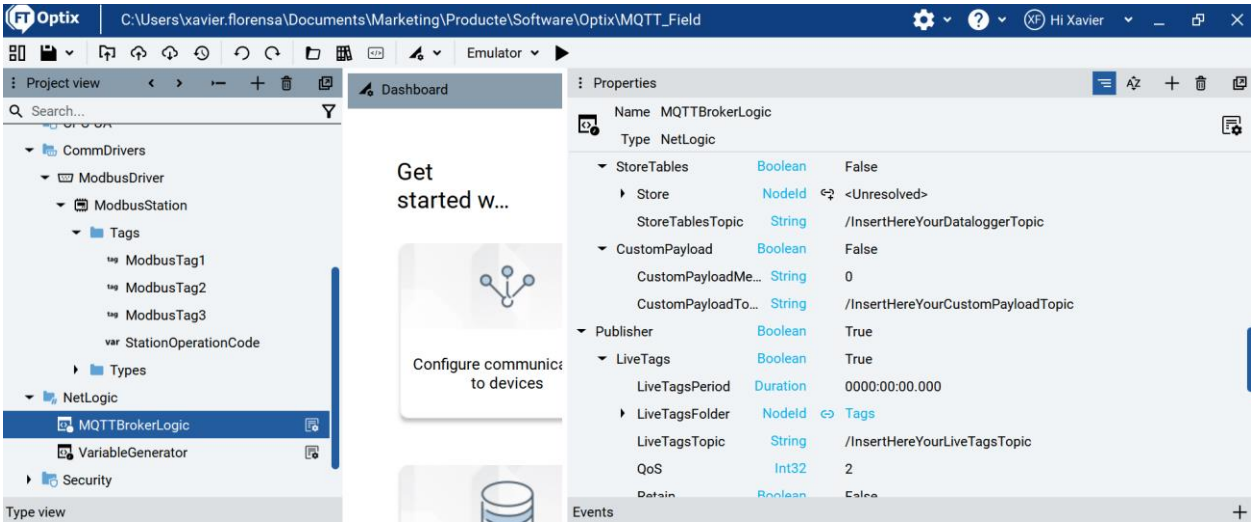
Drag and drop the library to the left over NetLogic



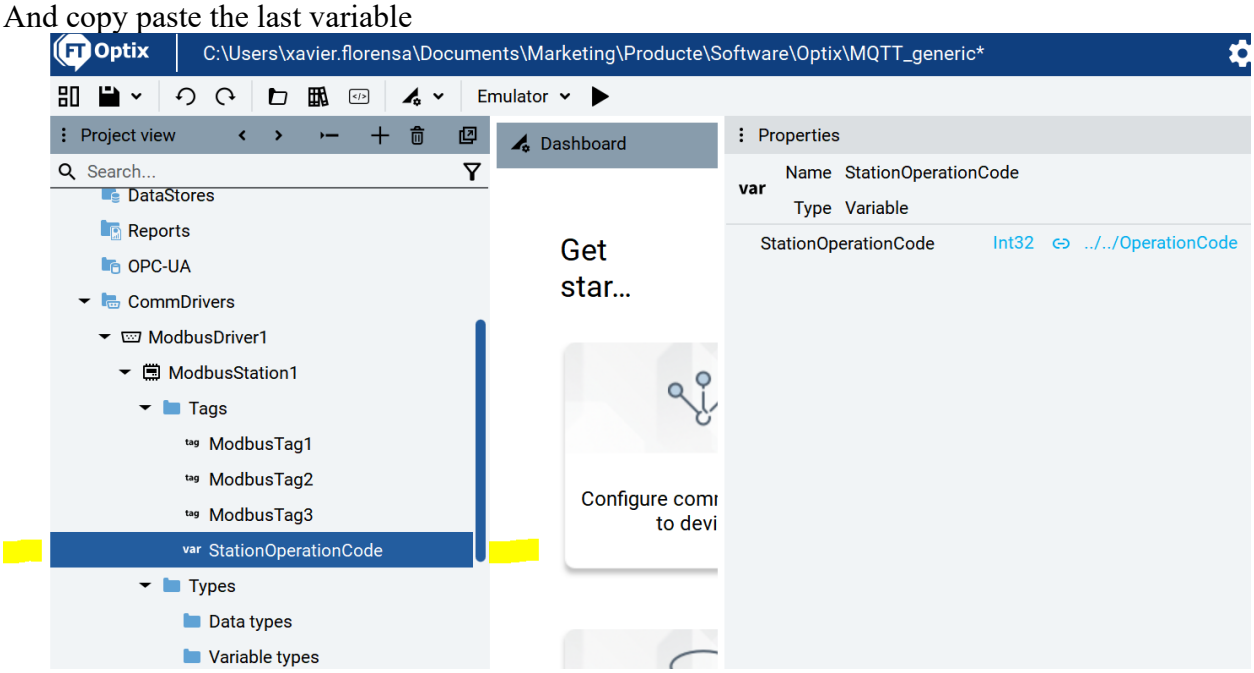
Now you have your MQTT object, but you have to specify were is your live tags folder



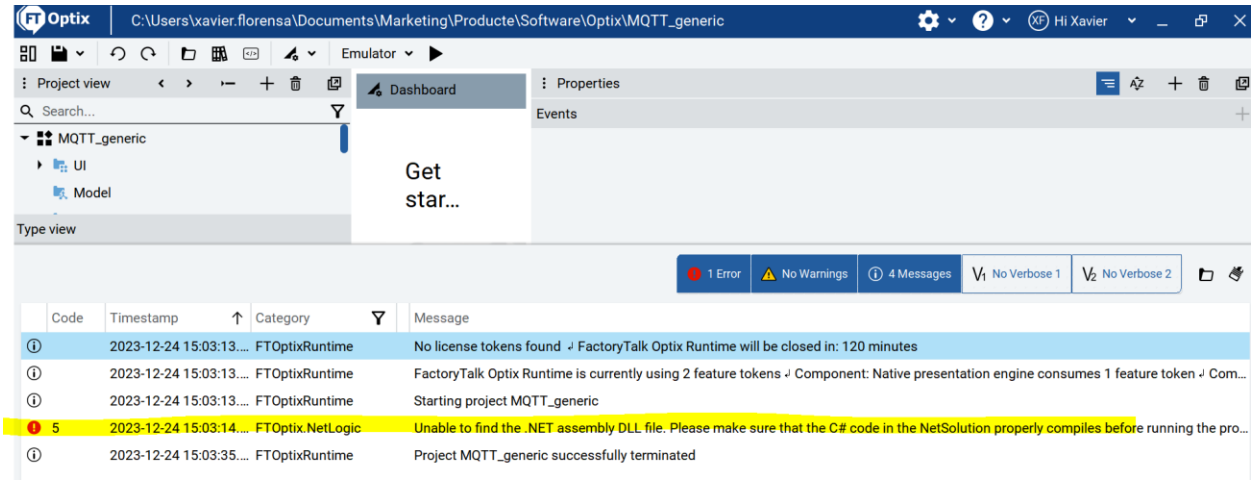
You have to create something like this



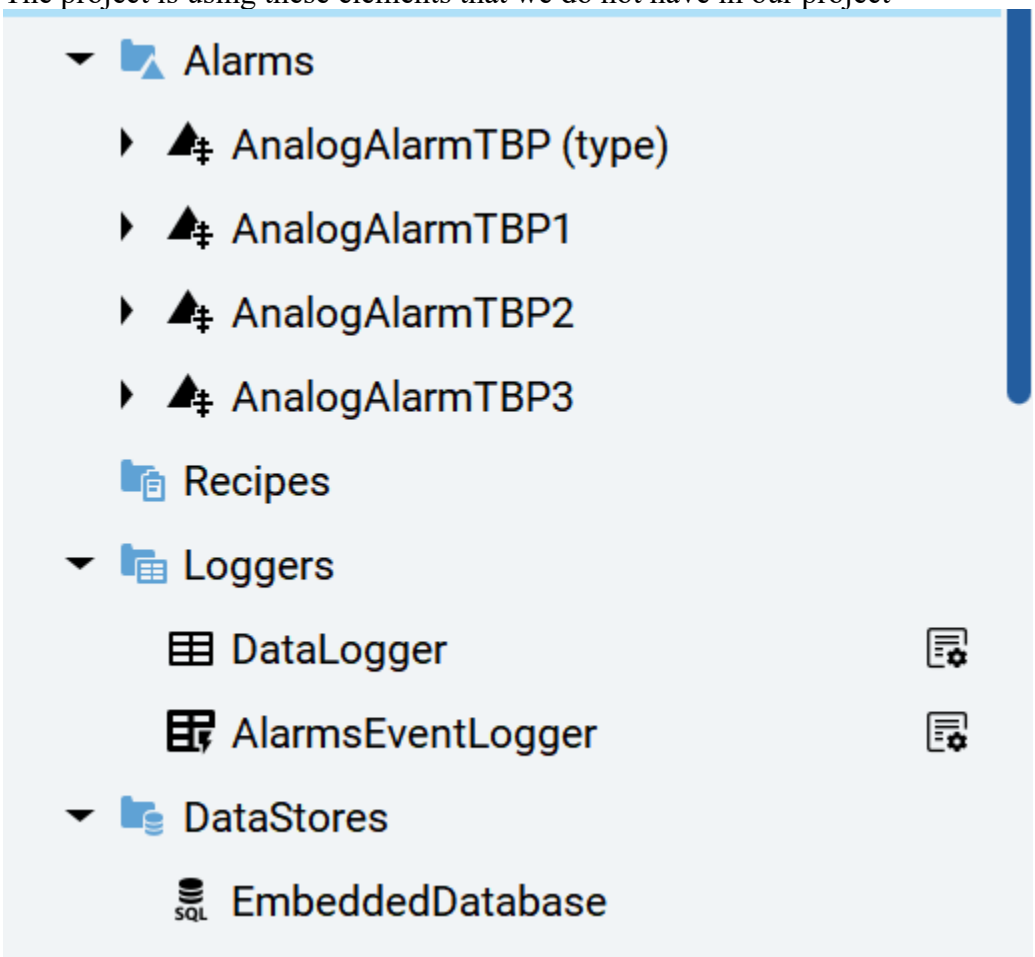
So let's adapt our application to the code accordingly
And copy paste the last variable

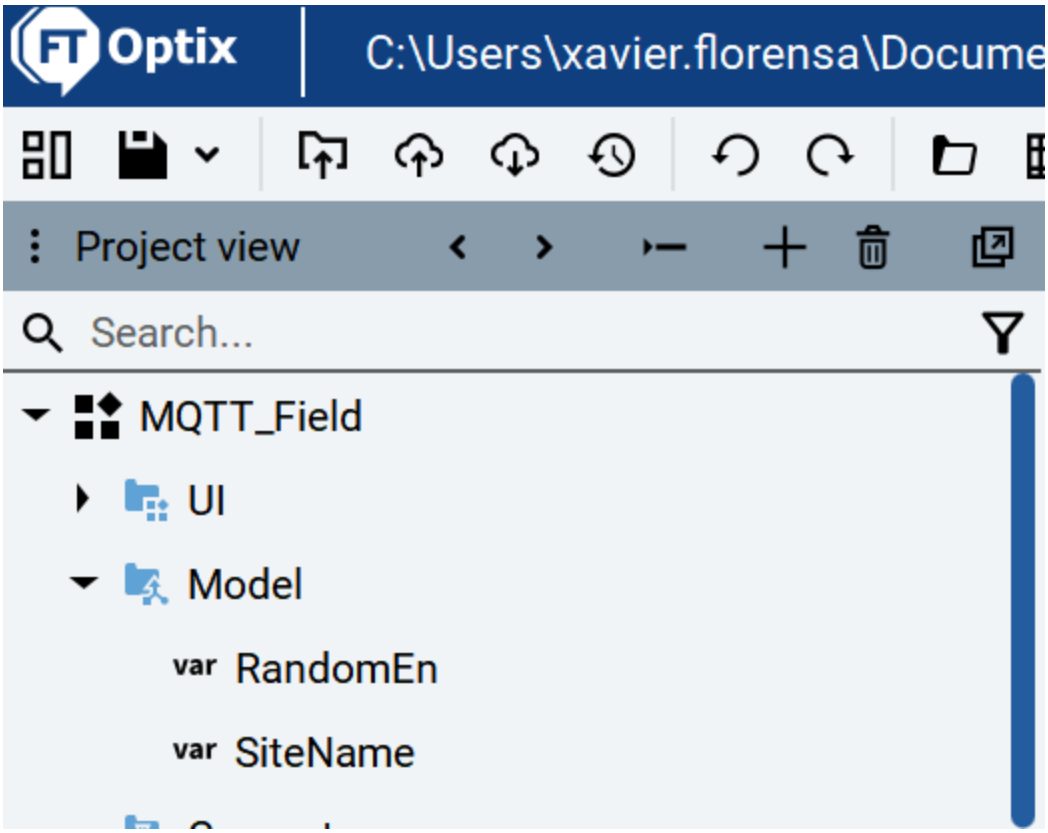


Let's run the code with a Modbus variable simulator



The project is using these elements that we do not have in our project





So sometimes it's not easy to reuse some code, especially if there is no simple application to extract the code from.
In this case the code has 2800 lines.
Let's use a simpler code with 50 lines

28.3. How to use .NET third party libraries in our projects

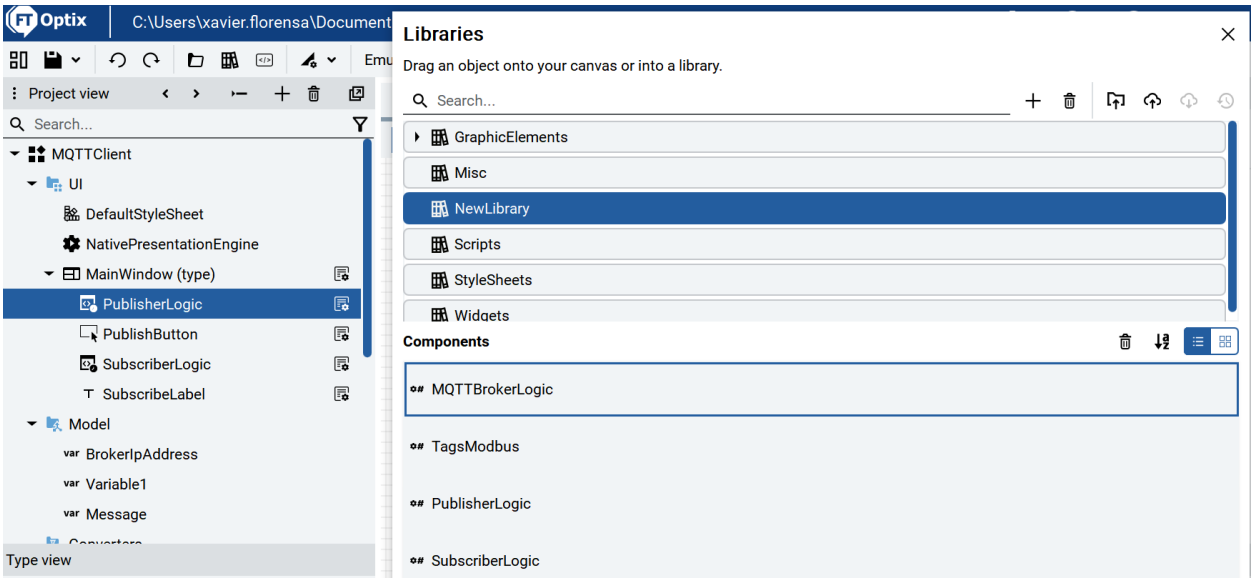
Let's use a simpler code with 50 lines
MQTT client you can download from here as we saw on chapter [Configuring an application as an MQTT client](file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html)
file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html

Tip: You can download a sample project from: MQTTClient.zip

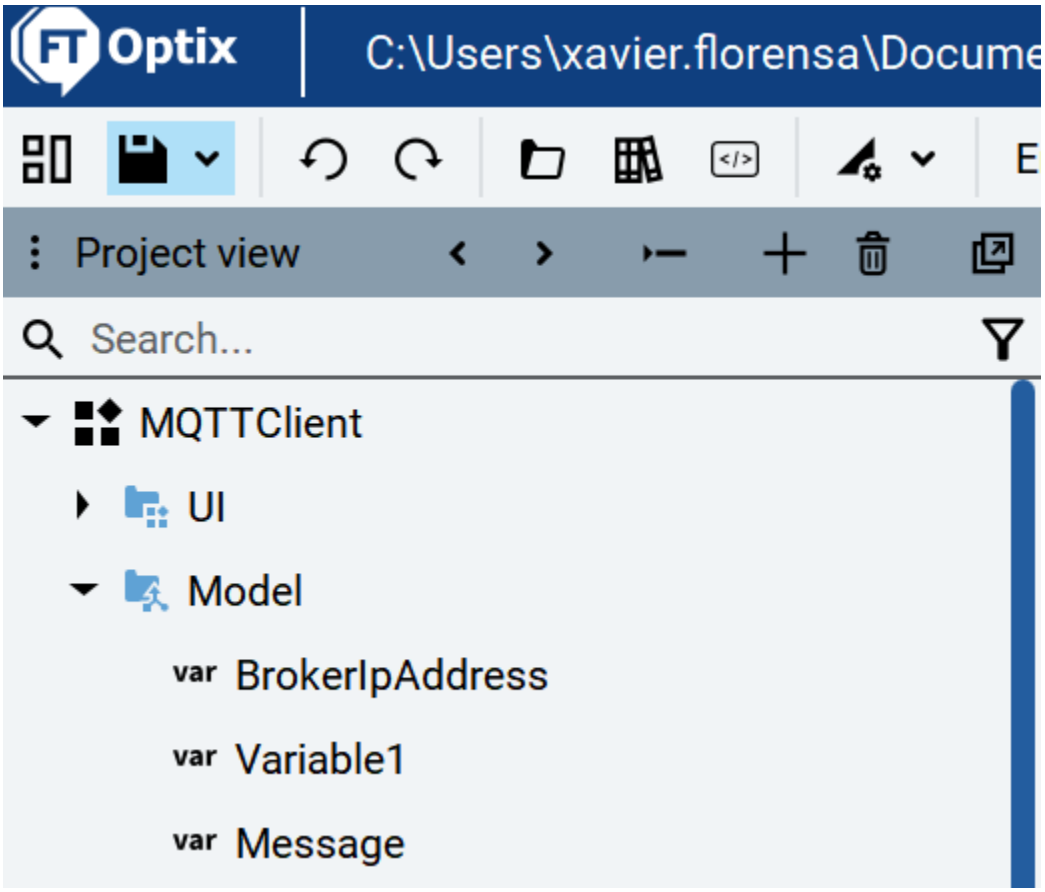
You can download the code from here

https://github.com/xavierflorensa/MQTTClient_FTOptix_help

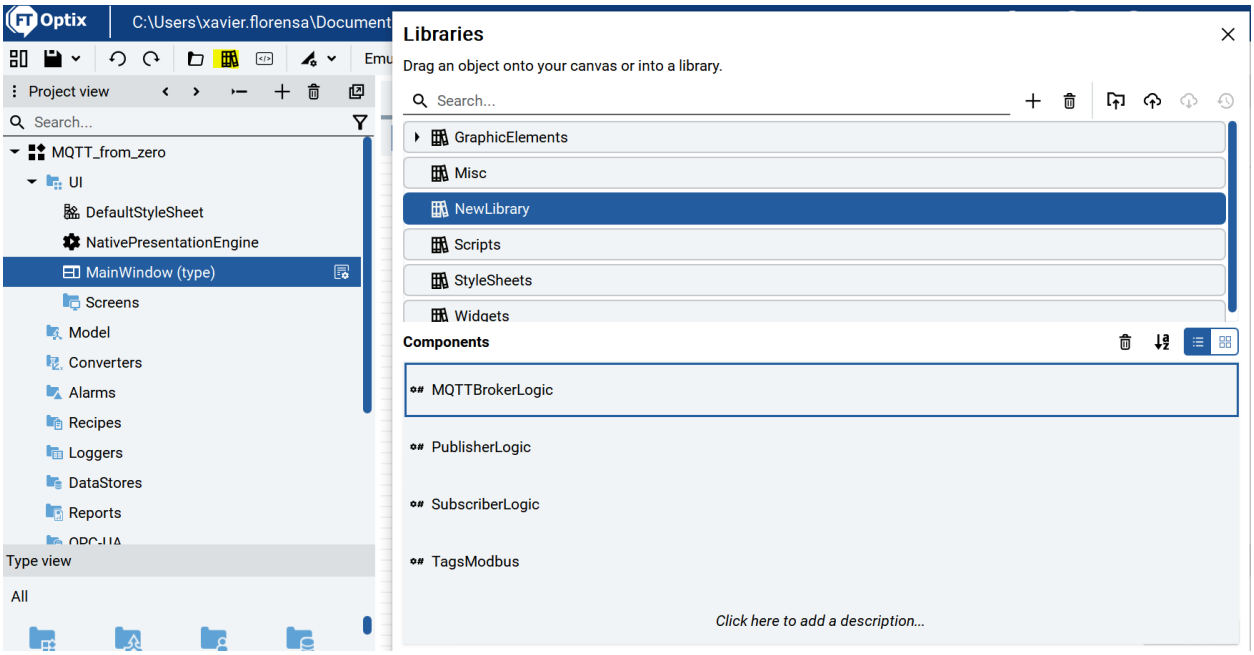
First of all, store the code objects in my libraries with drag and drop.



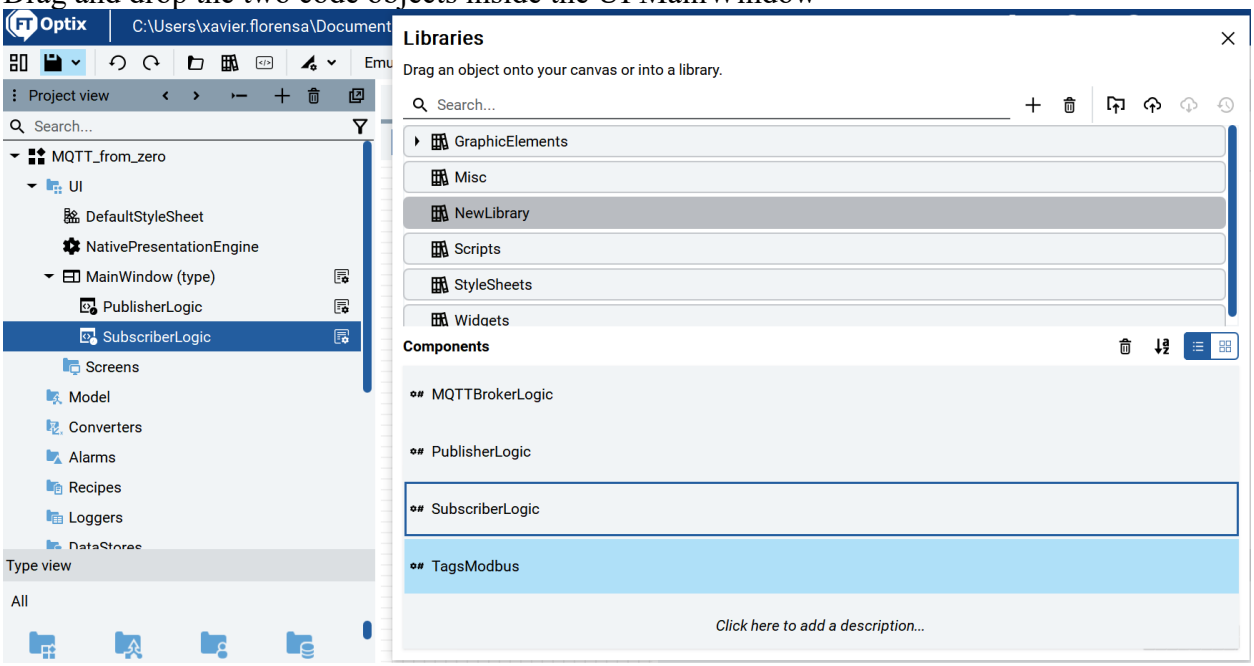
Be aware that this code works with these variables



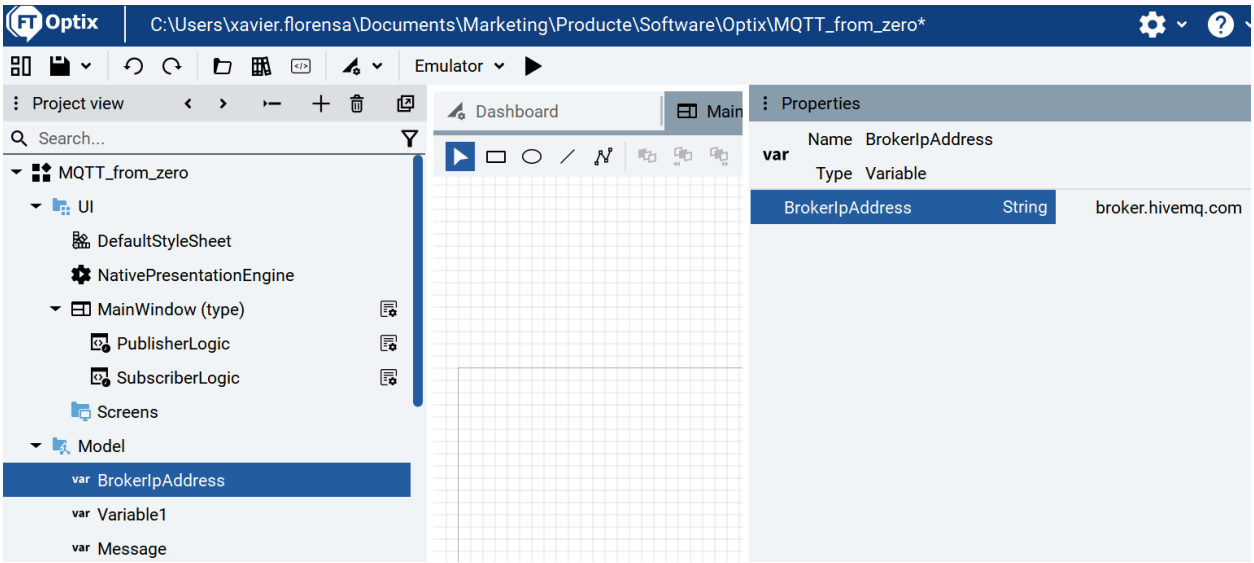
Close the project
Now create a new project and define the variables
Open libraries



Drag and drop the two code objects inside the UI MainWindow



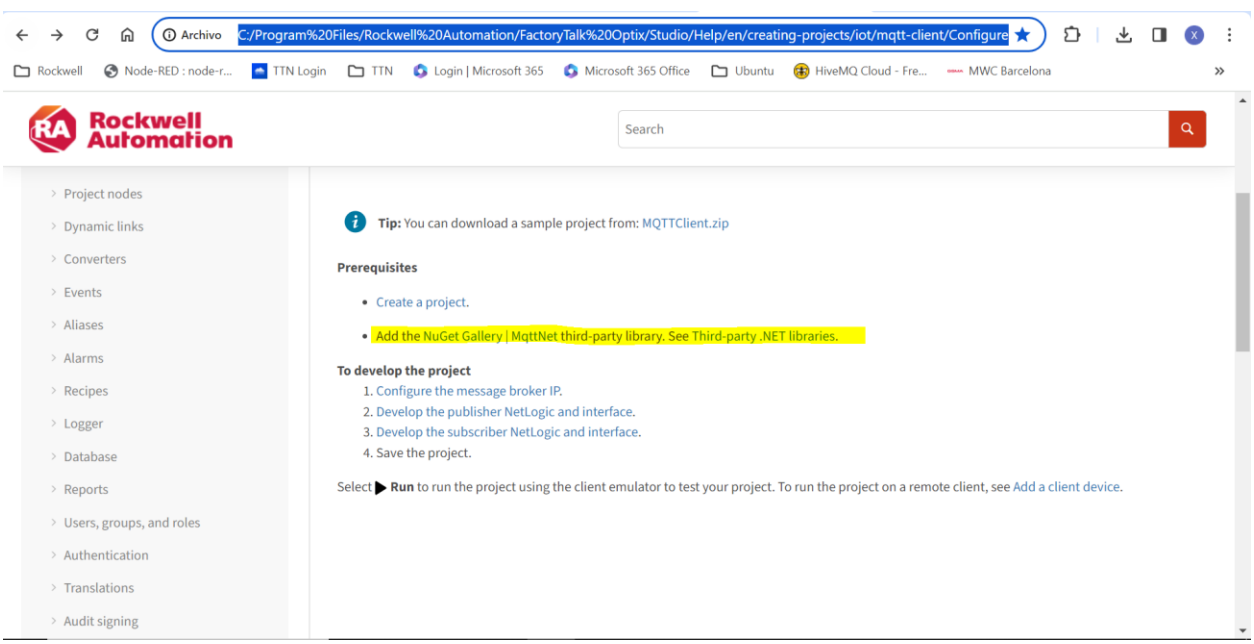
Create the variables and initialize the IP address



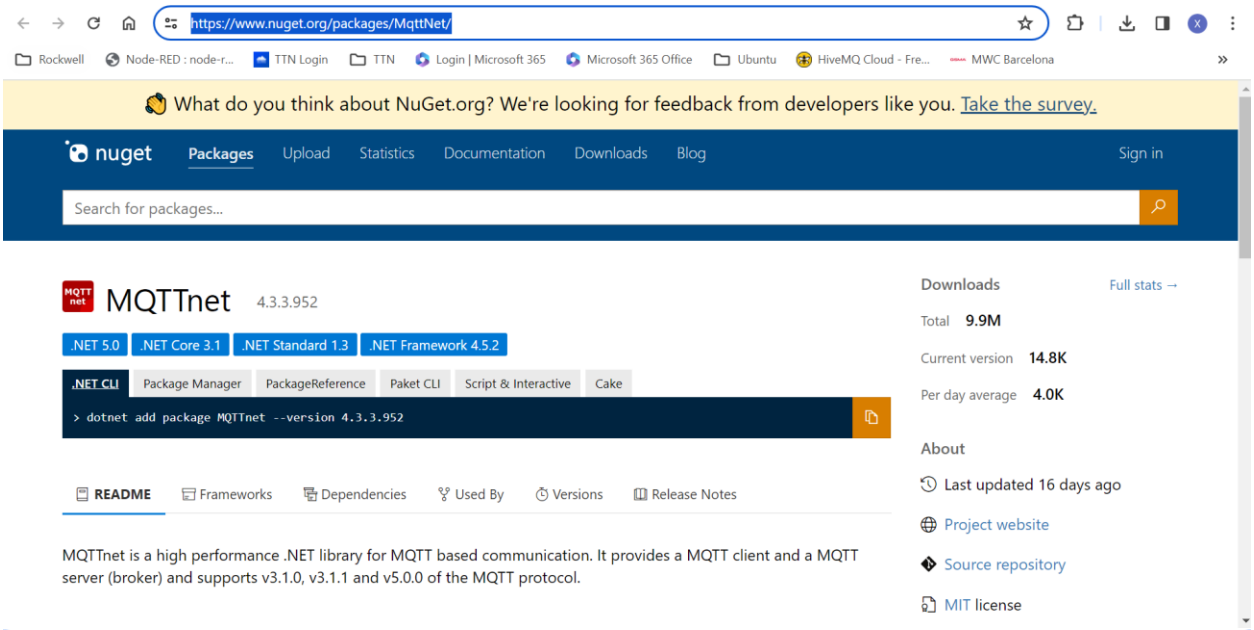
Now let's put a box to show the subscribed message
We see that the topic is
/my_topic
As we can see here

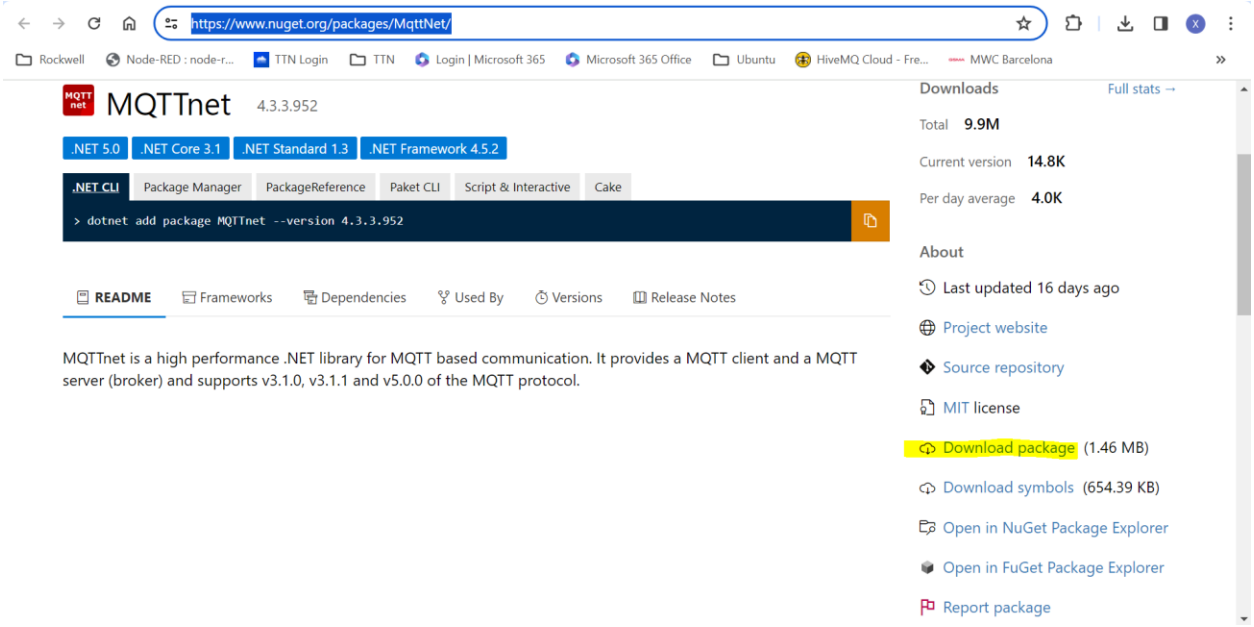
```
ushort msgId = subscribeClient.Subscribe(new string[] { "/my_topic" }, // topic
```

But this is not working since we did not installed the right libraries.
Following the guide we have to install a library
file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html

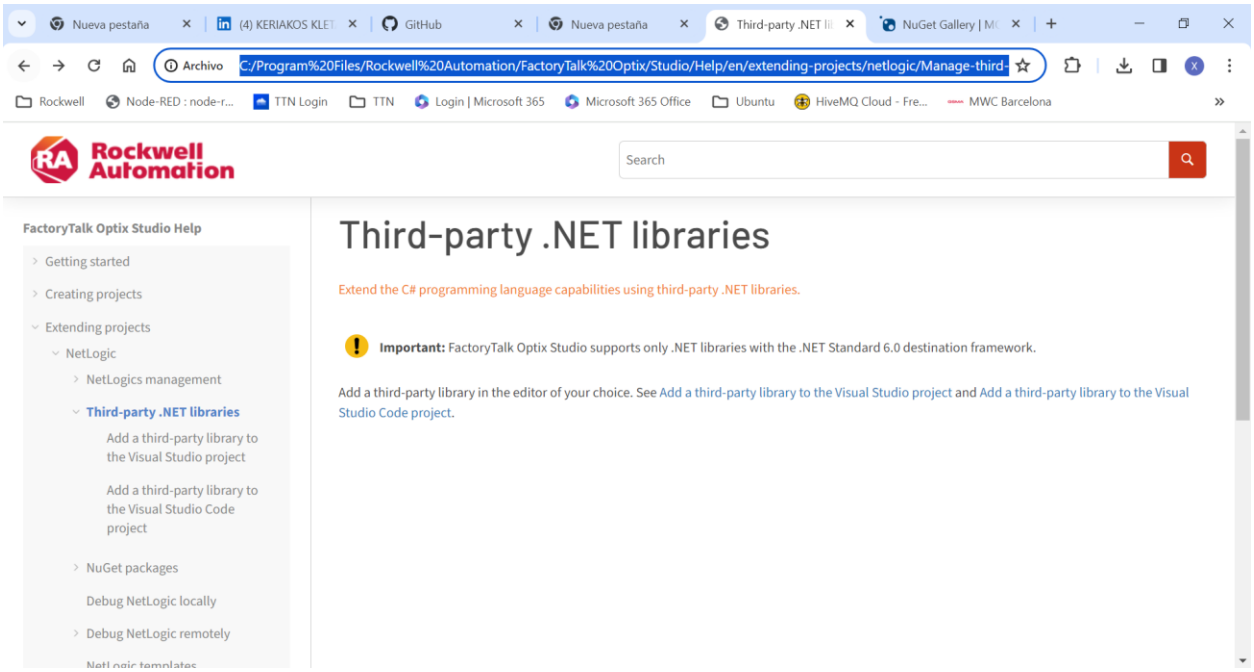


https://www.nuget.org/packages/MqttNet/

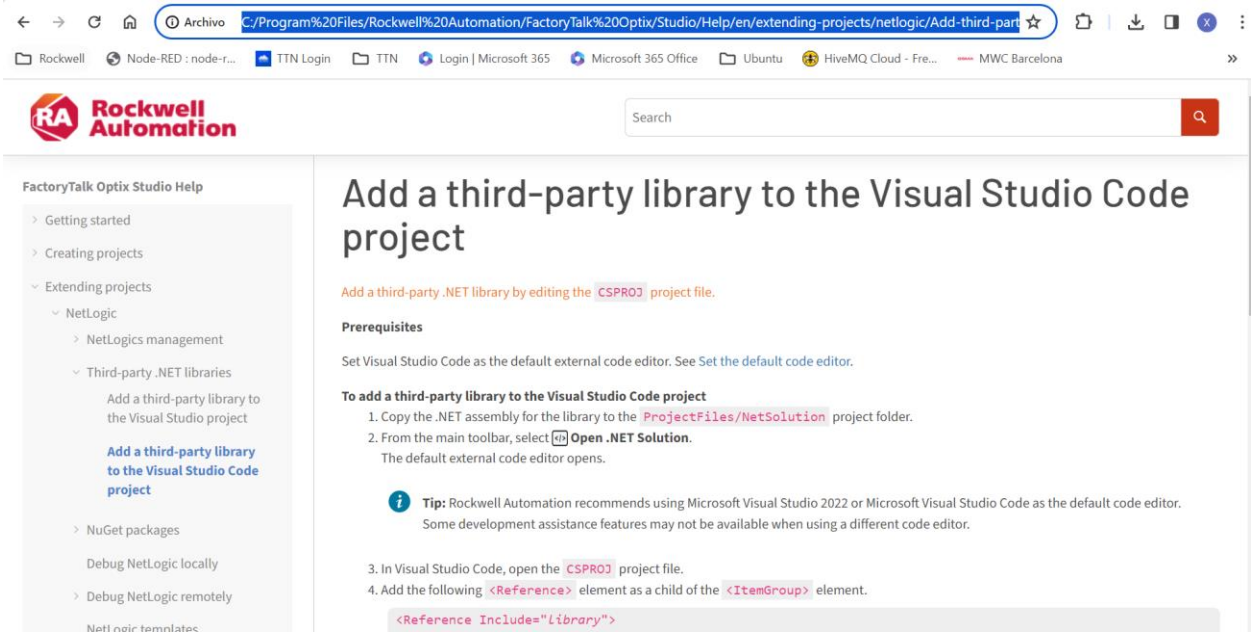




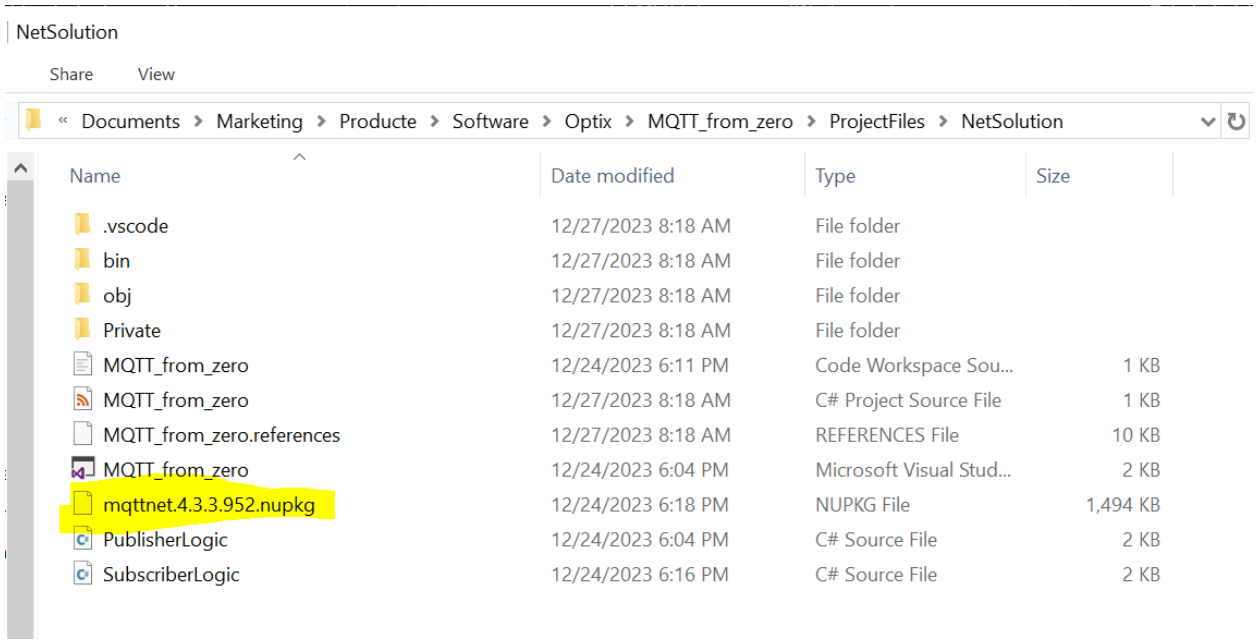
File:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Manage-third-party-dot-net-libraries.html



file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Add-third-party-library-the-vs-code-project.html



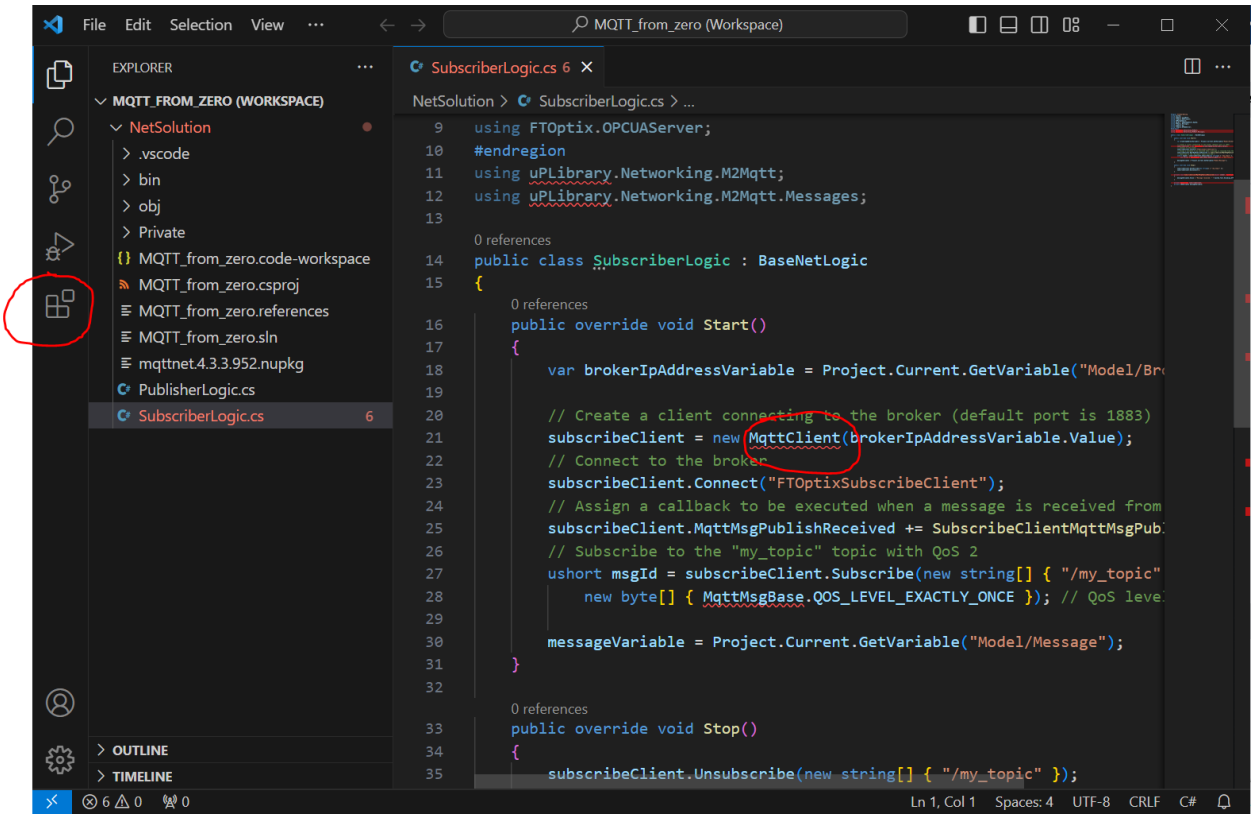
Copy the downloaded file to the directory



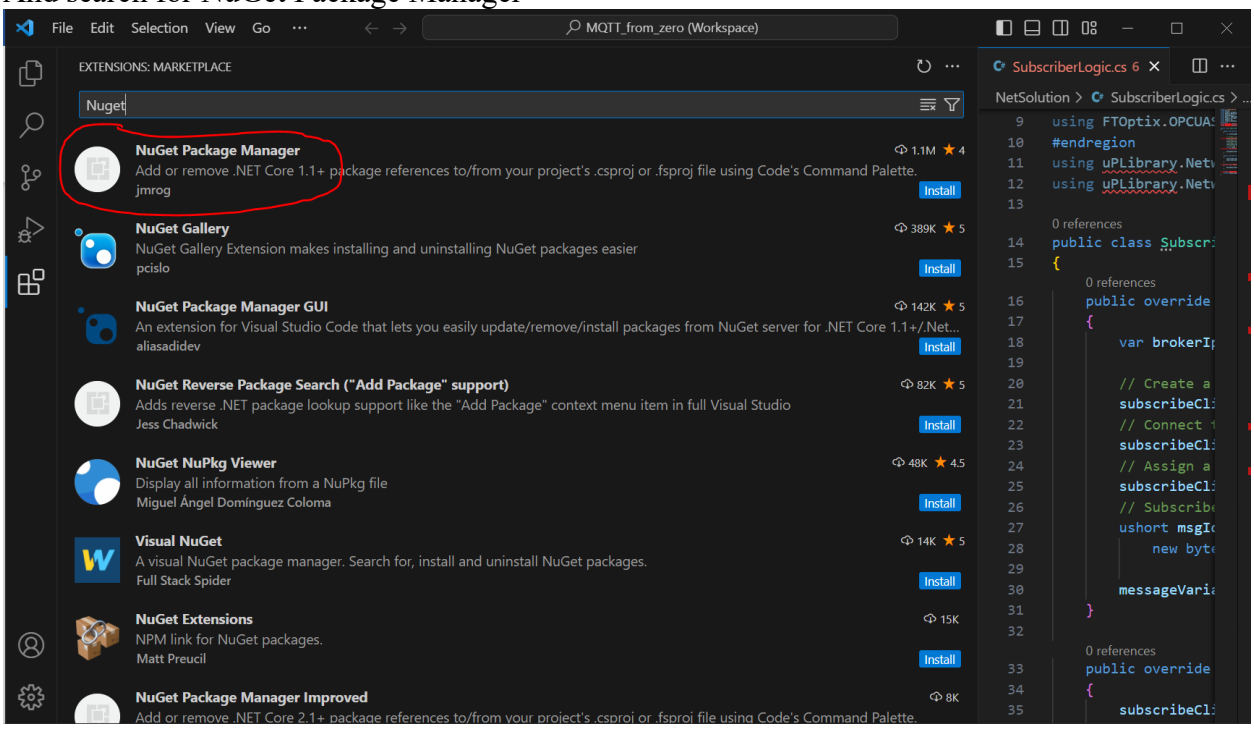
28.3.1. Installing Nuget packages in Visual Studio Code

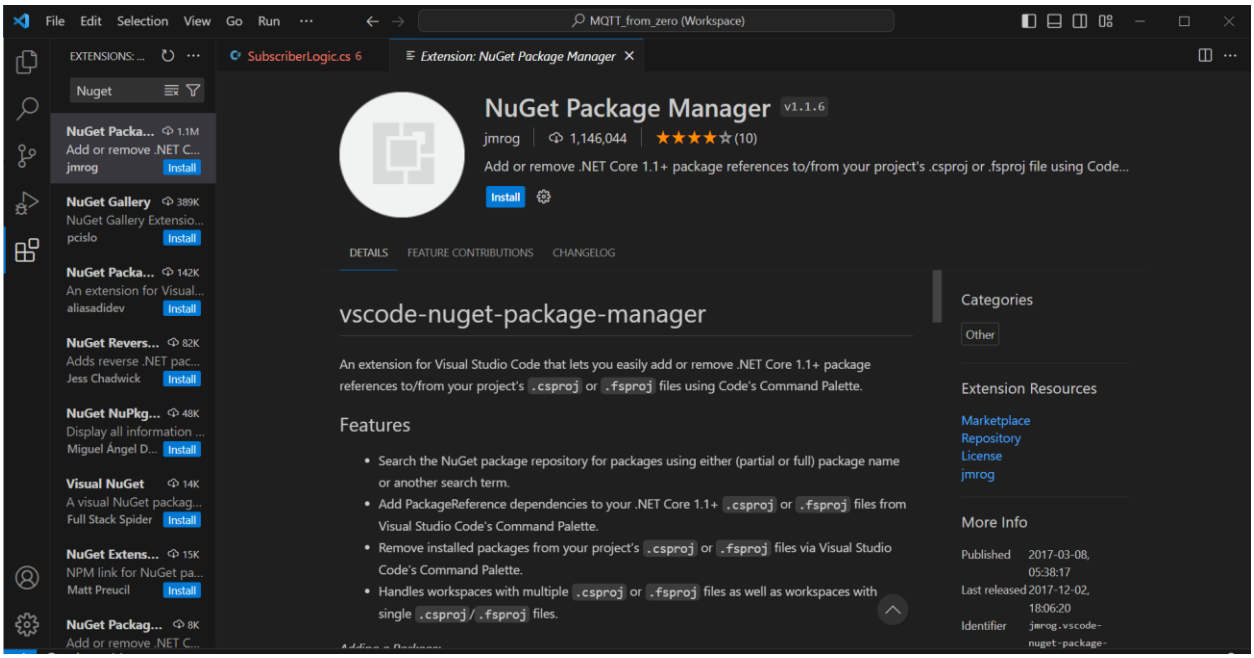
It is much easier to install libraries in Visual Studio Code

https://www.youtube.com/watch?v=Qy-vwB_TEW4
Open for instance, subscriberlogic.cs on VS code
Be aware that VScode complaints with MqttClient
So let's go to Extensions (the lowest icon on the left)

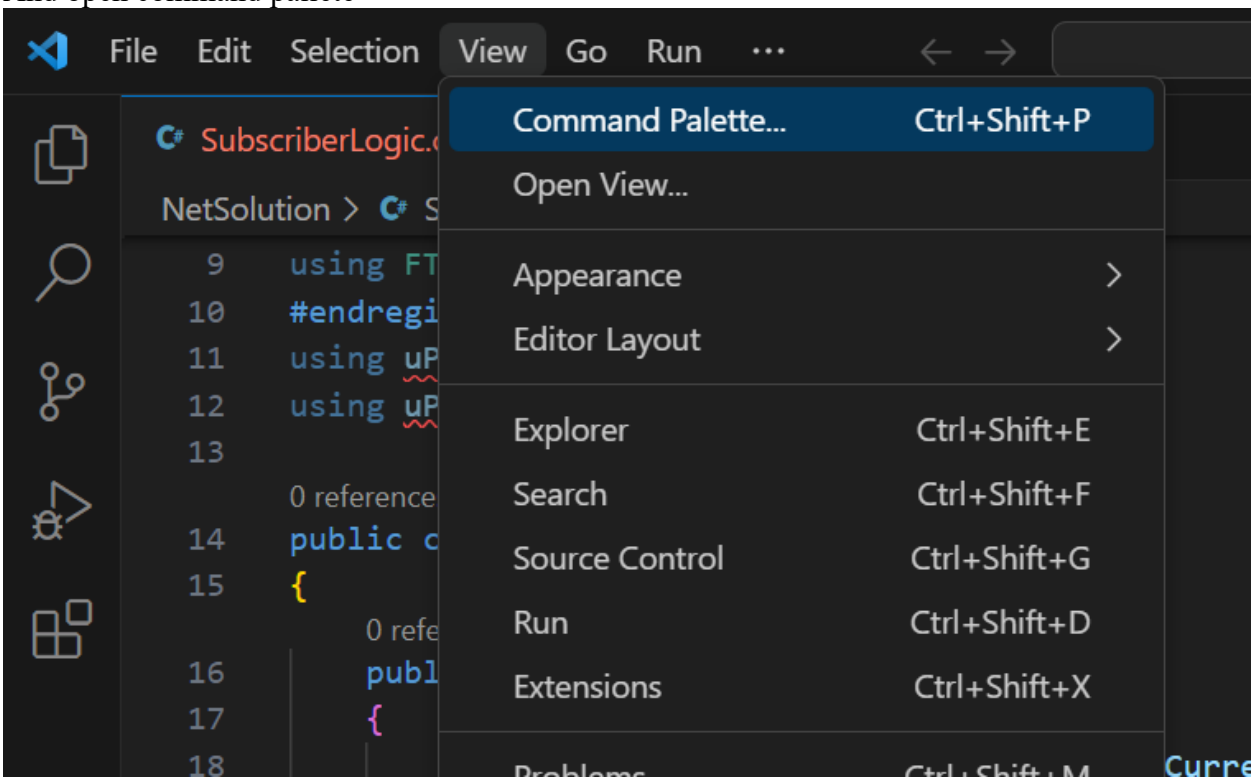


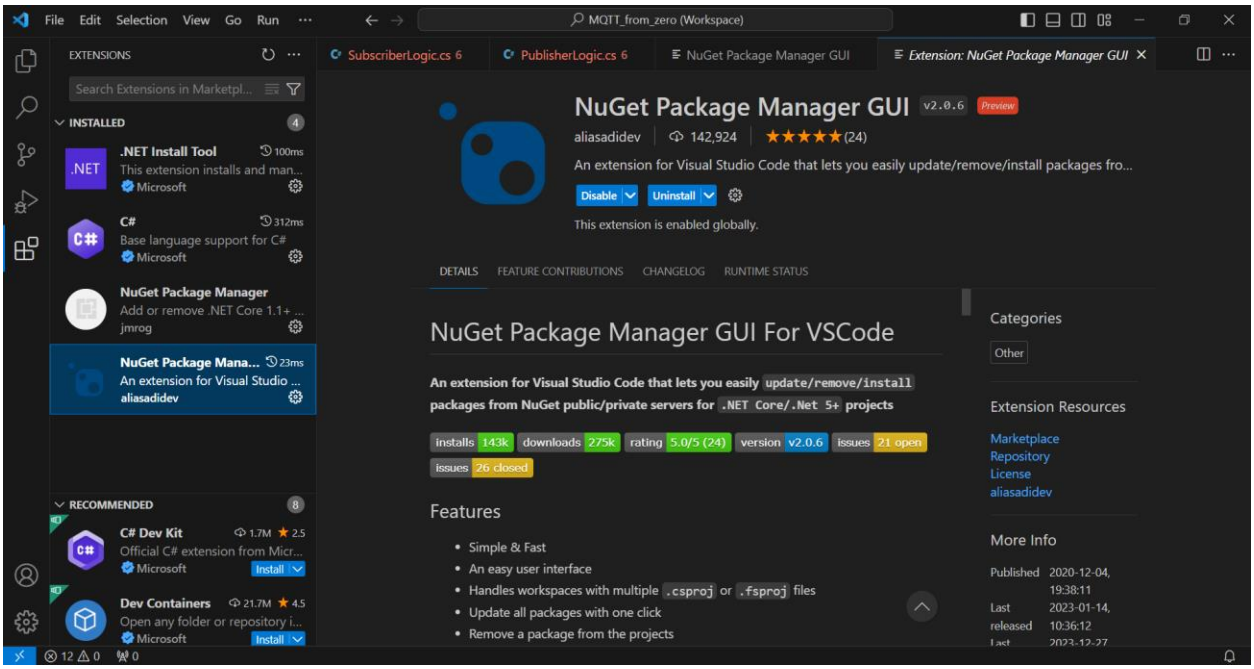
And search for NuGet Package Manager





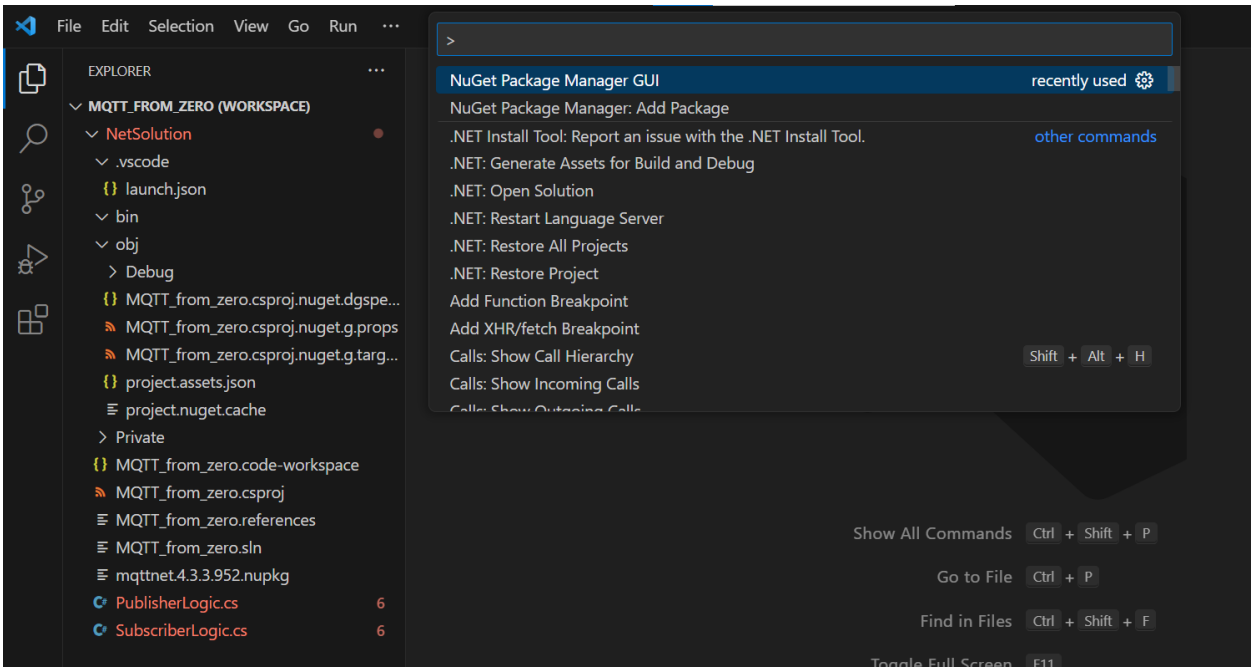
Now close clicking again on the Extensions lowest left icon,
And open command pallete



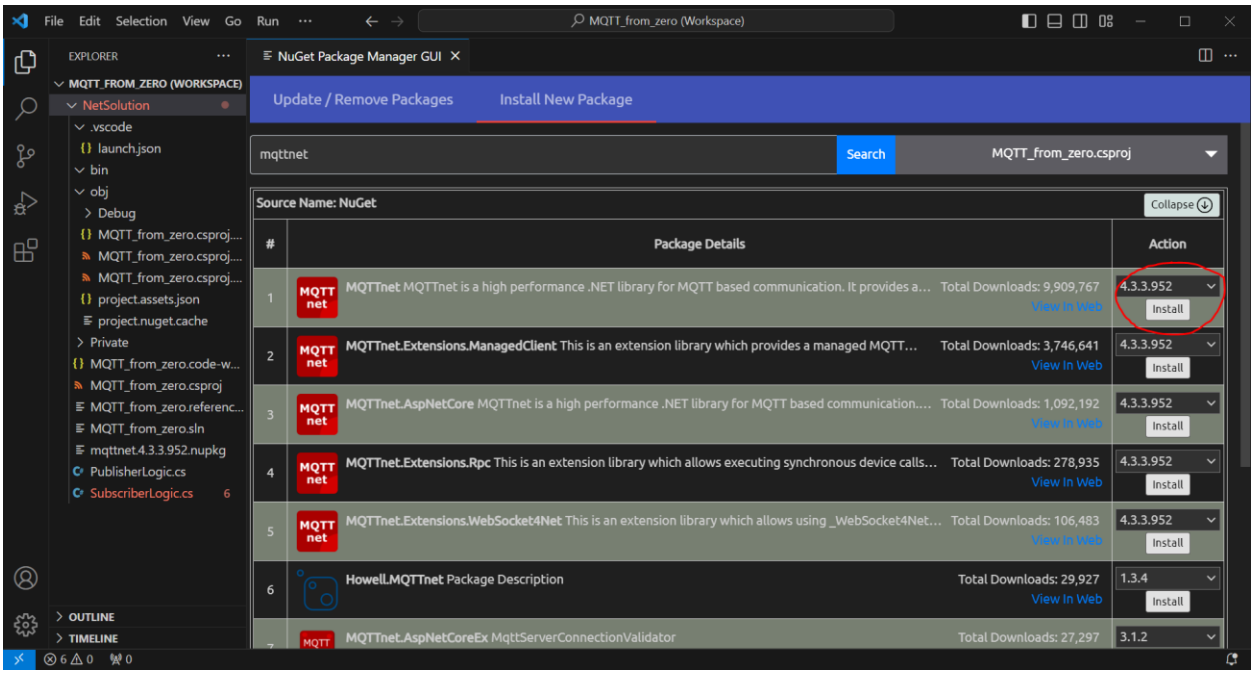


Install such package manager if you did do this step in the past

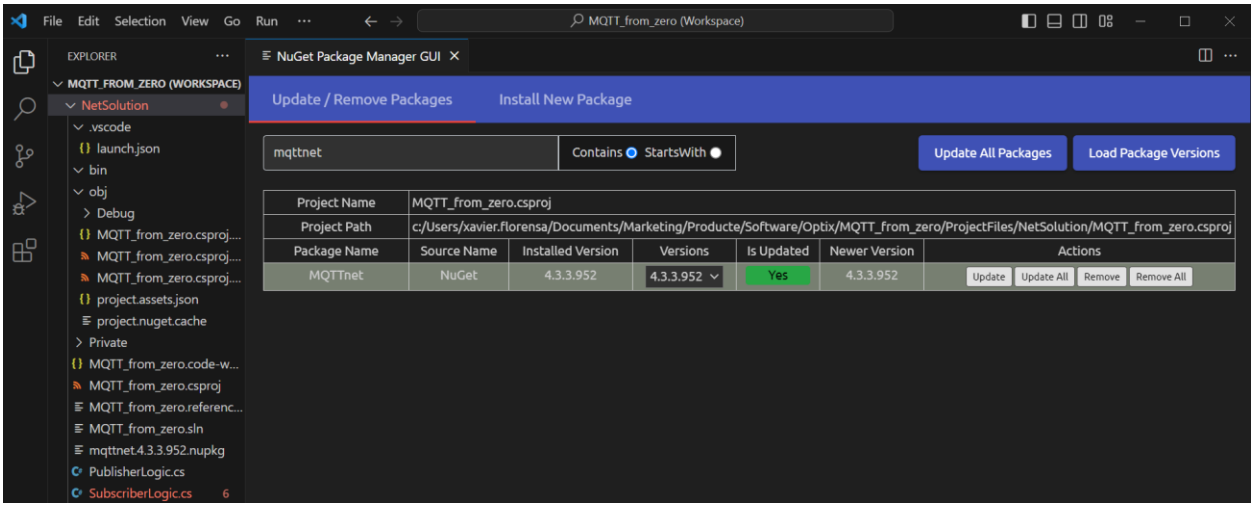
Go to View / Command Pallete



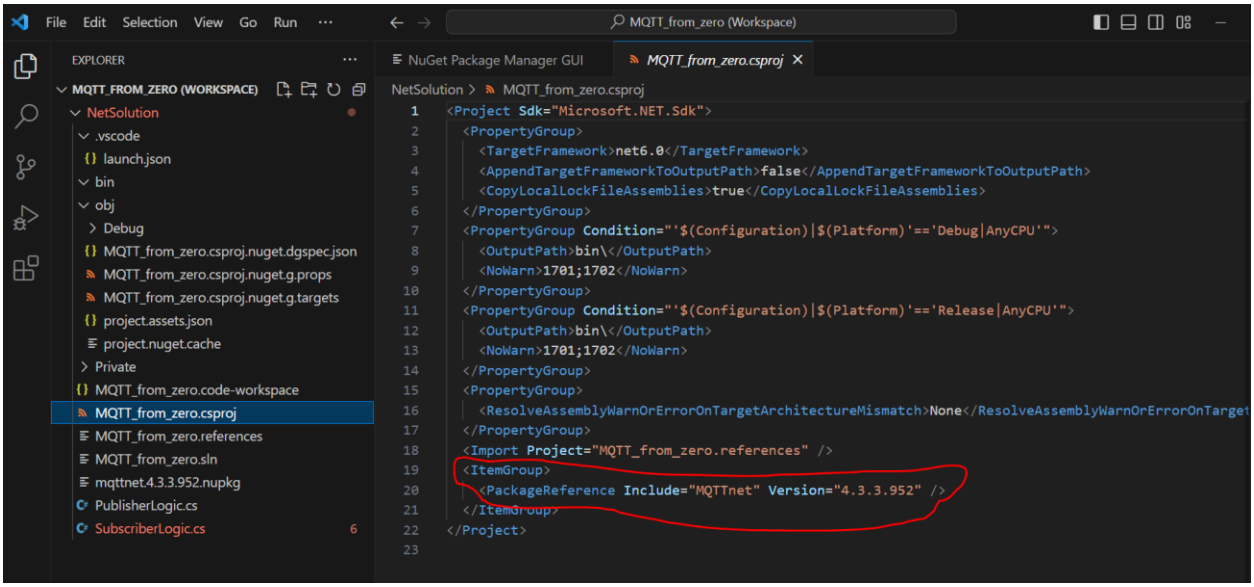
Search for the library on the server and click on install



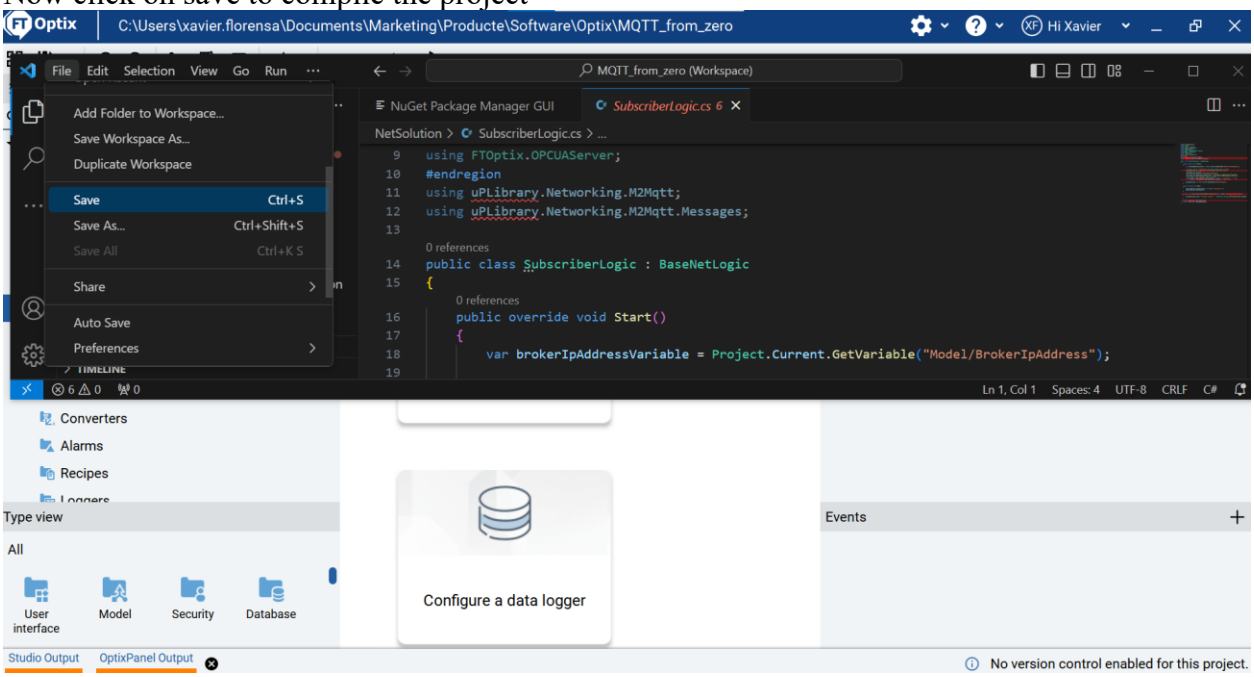
Then go to Upload/Remove Packages
Proceed with Upload and Update



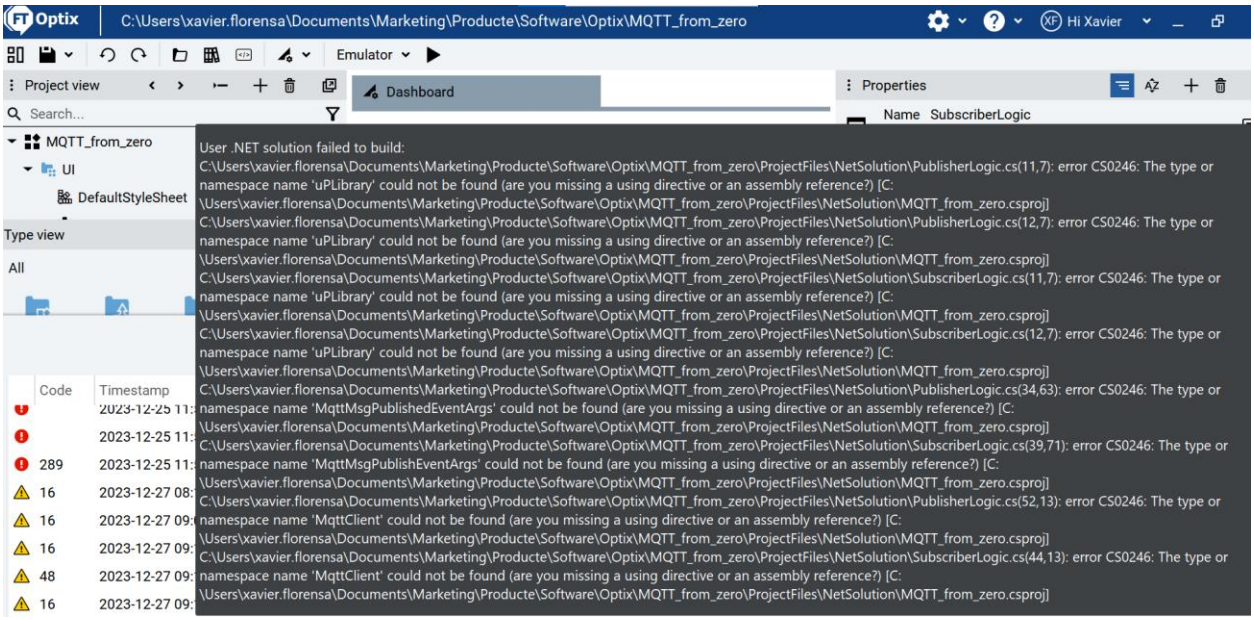
If we take a look at *.csproj



We have it installed as it was recommending on the web help
C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/Add-third-party-library-the-vs-code-project.html
Now click on save to compile the project



We still have errors,
This is due to the fact that we still need the library M2MQtt.dll



Let’s take a look at the example you can download from Help pages

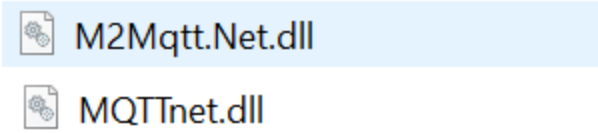
Figure: Application example

Tip: You can download a sample project from: MQTTClient.zip

C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/creating-projects/iot/mqtt-client/Configure-an-application-as-an-mqtt-client.html

We see that on that example there are two dll files.
To advance in this issue just copy these two DLLs on bin directory (as we see on the examples that you have these dlls that we are missing

ProjectFiles > NetSolution > bin



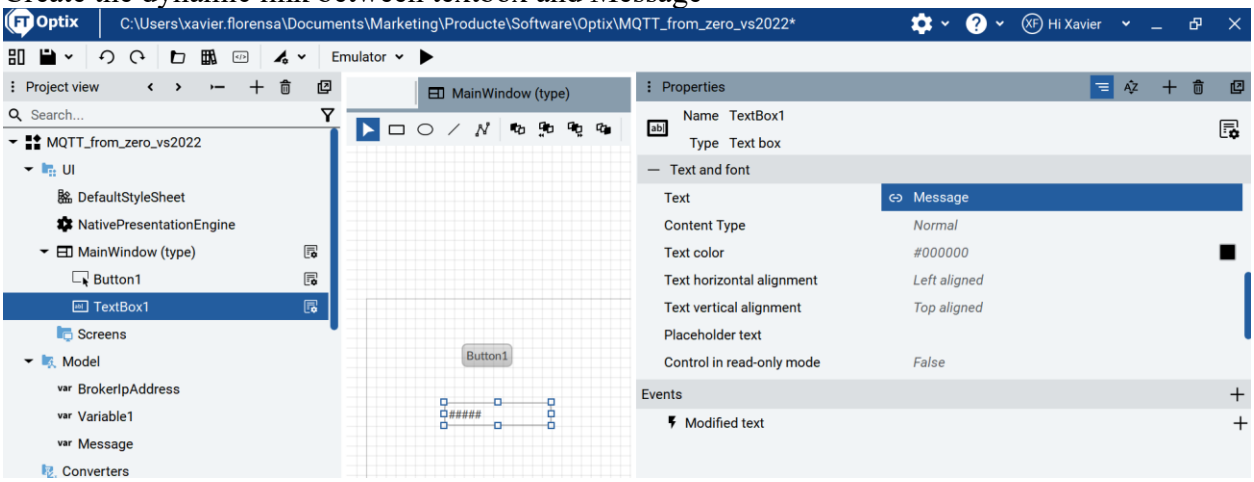
You also have to install the M2Mqtt NuGet package
Proceed as before when installing the library MQTTnet, install, load and update
Then you will have no errors

Not it works

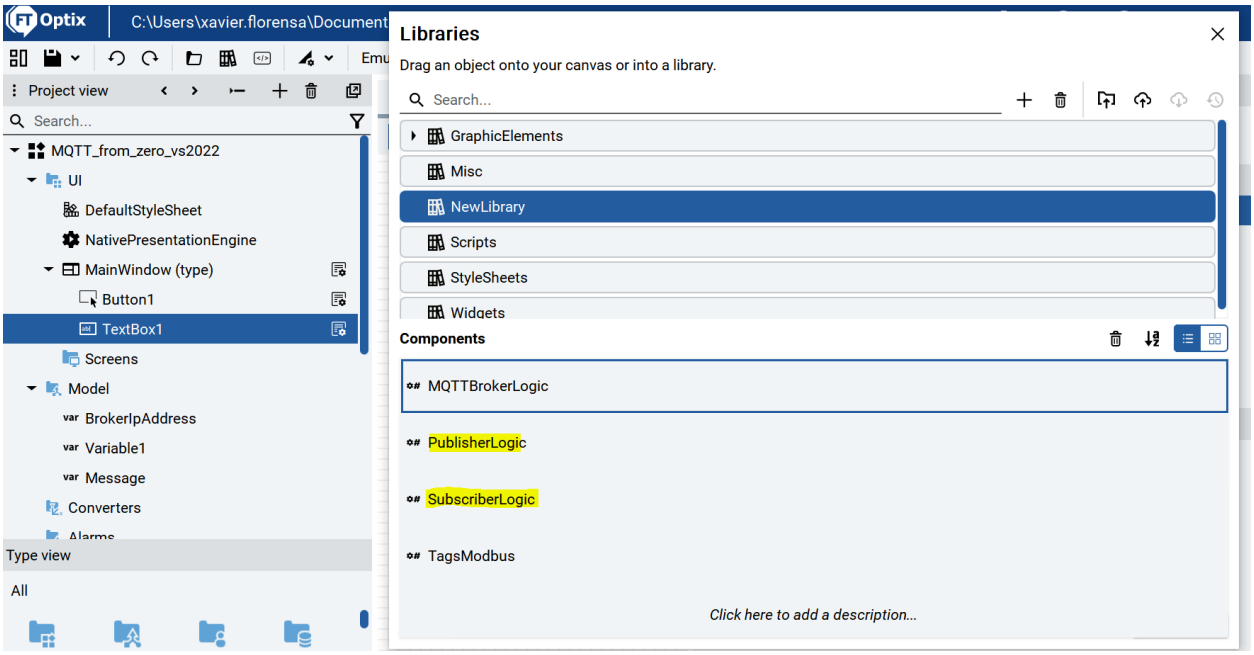
You can also use Visual Studio 2022 to install the .net packages

28.3.2. With Visual Studio 2022 community edition

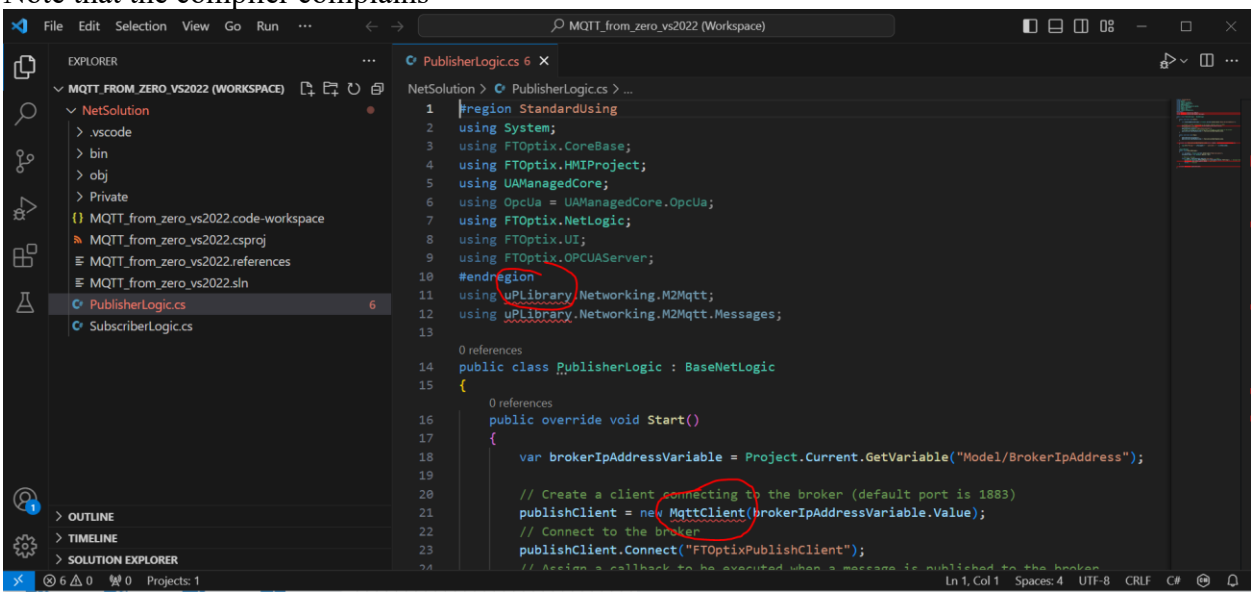
Let's try to install the two Nuget packages with Visual Studio 2022, and then once installed, let's move back to Visual Studio Code.
Create an application with FT Optix, with the three variables and the button and label.
Create the dynamic link between textbox and Message



Drag and drop to MainWindow the two libraries that you stored in previous chapters.
PublisherLogic and SubscriberLogic

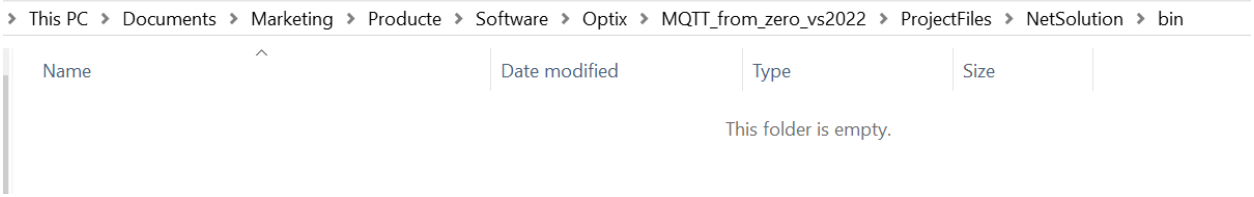


Open PublisherLogic
Note that the compiler complains

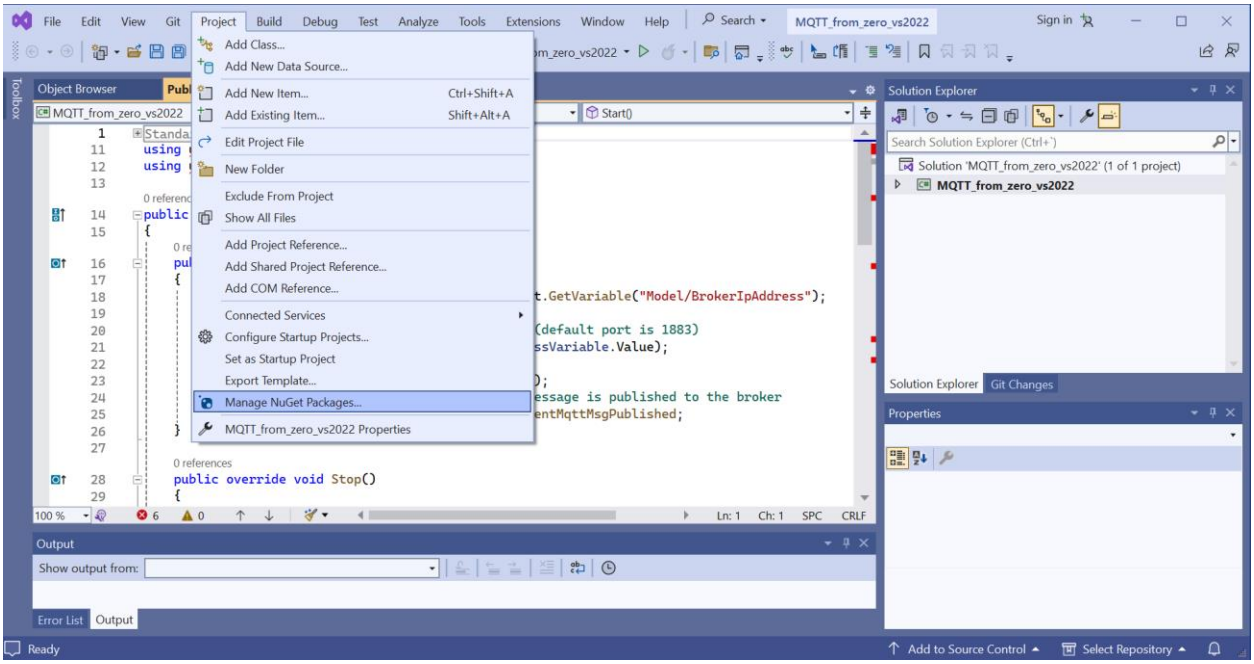


Close VS Code

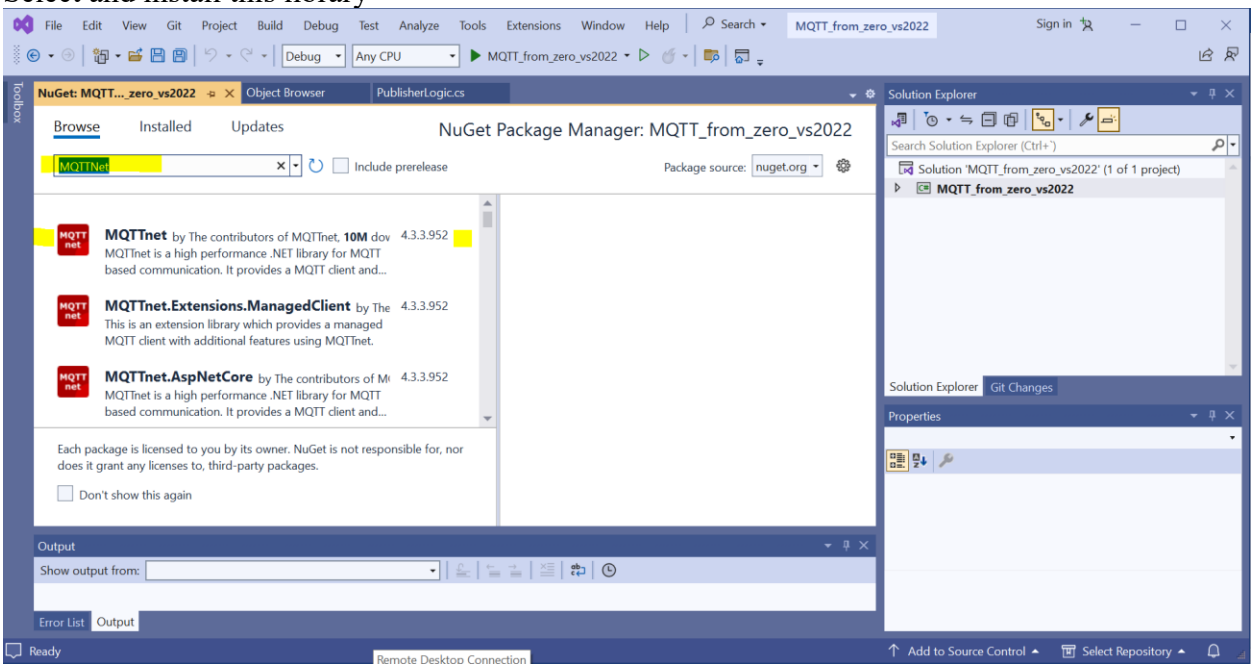
Now change the editor to Visual Studio 2022, save close and open FTOptix again to admit chages.
Look at this directory, there is nothing since compiler has not succeed building the application.

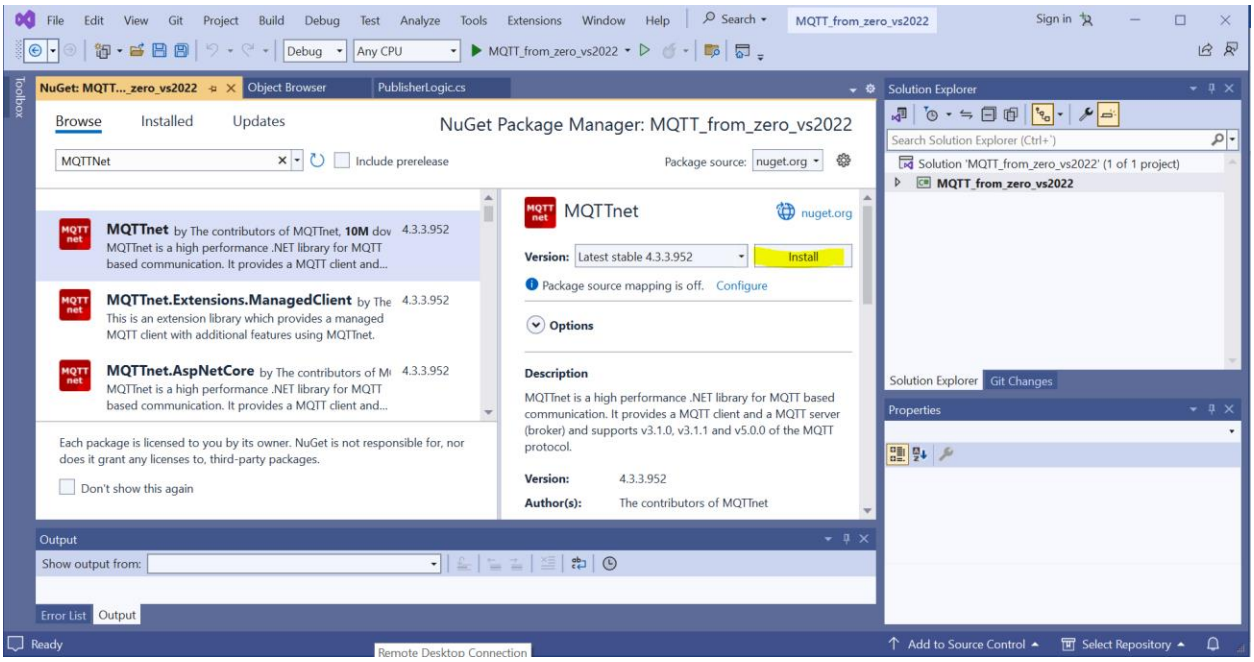


Click on Publisherlogic. Go to Project/Manage NuGet Packages.

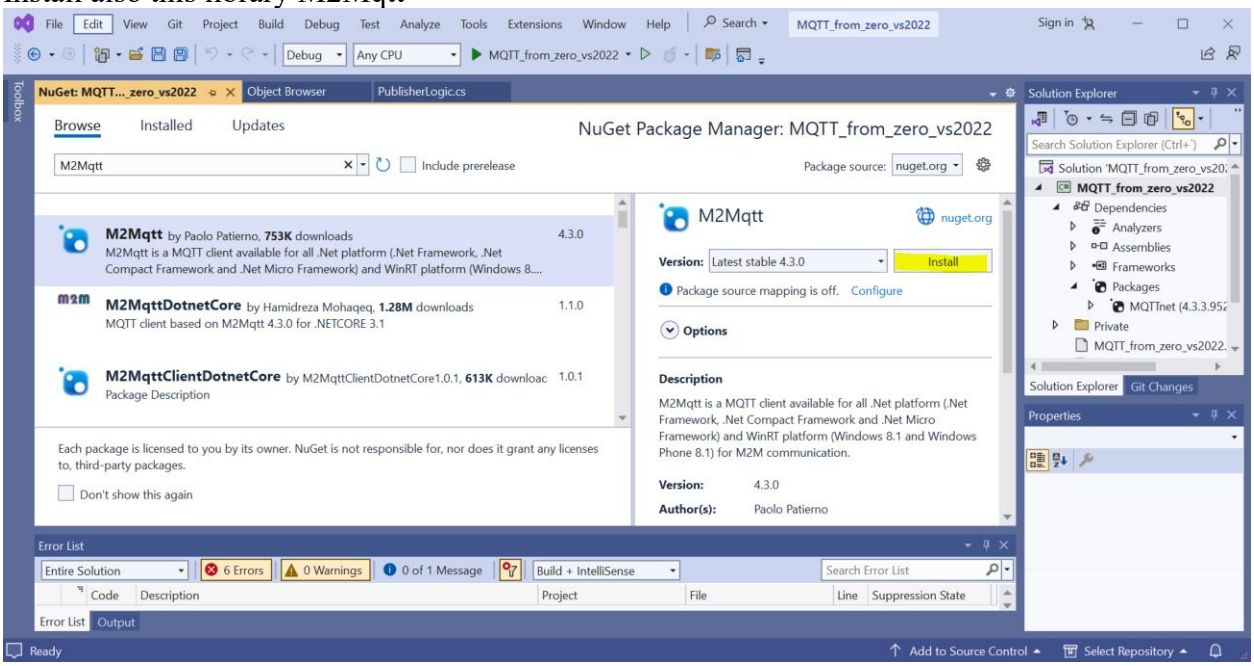


Select and install this library

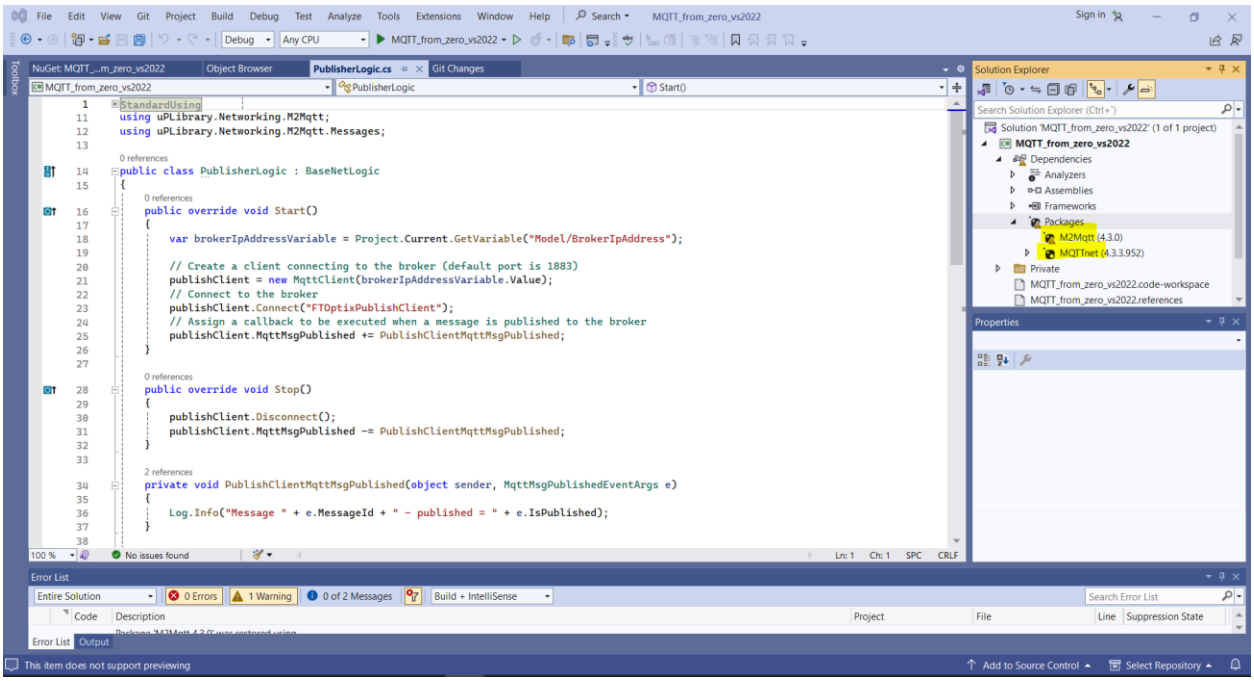




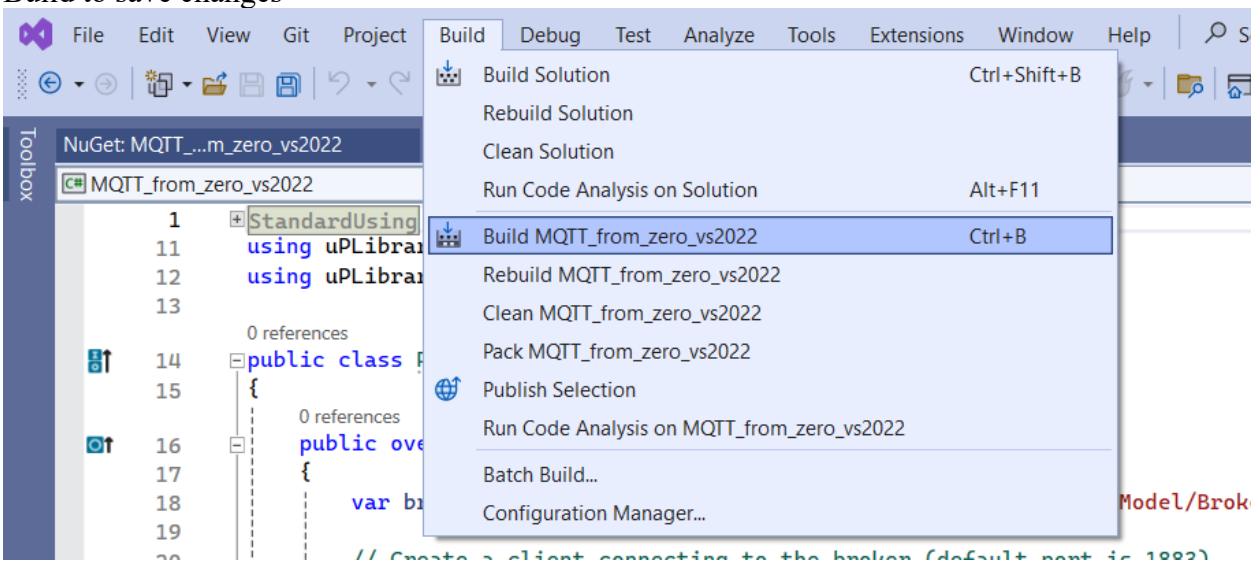
Install also this library M2Mqtt



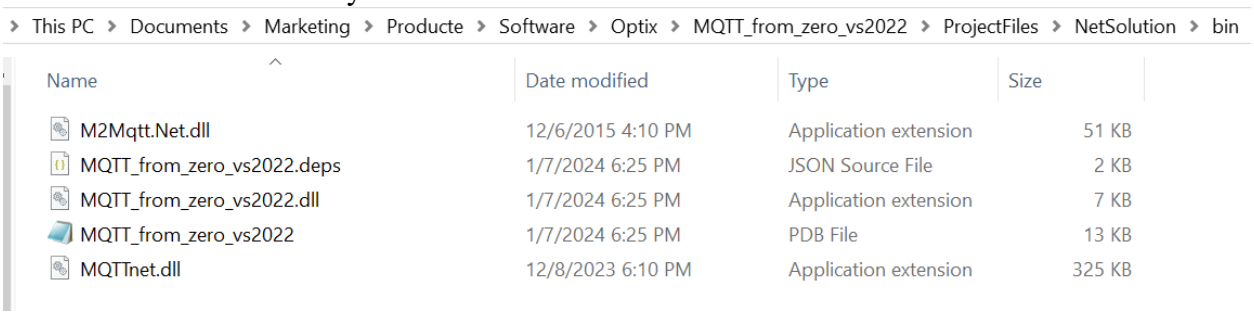
Now we have the two packages



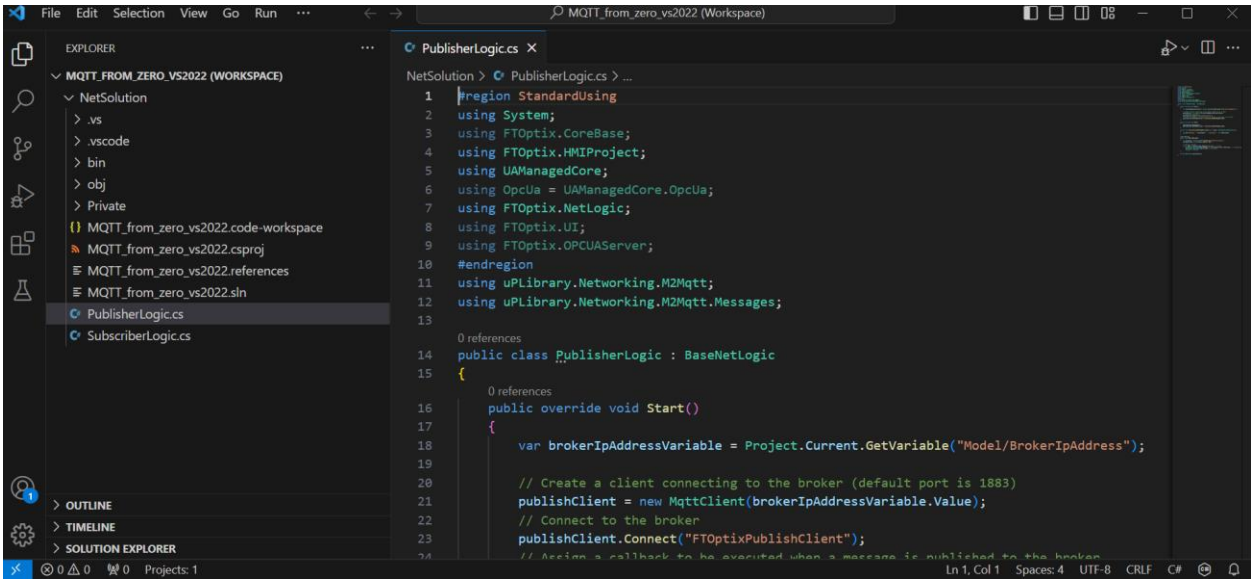
Be aware that there are no errors on the code like before.
Build to save changes



Close VS 2022
Change the editor on FT Optix to VS Code
Take a look at this directory

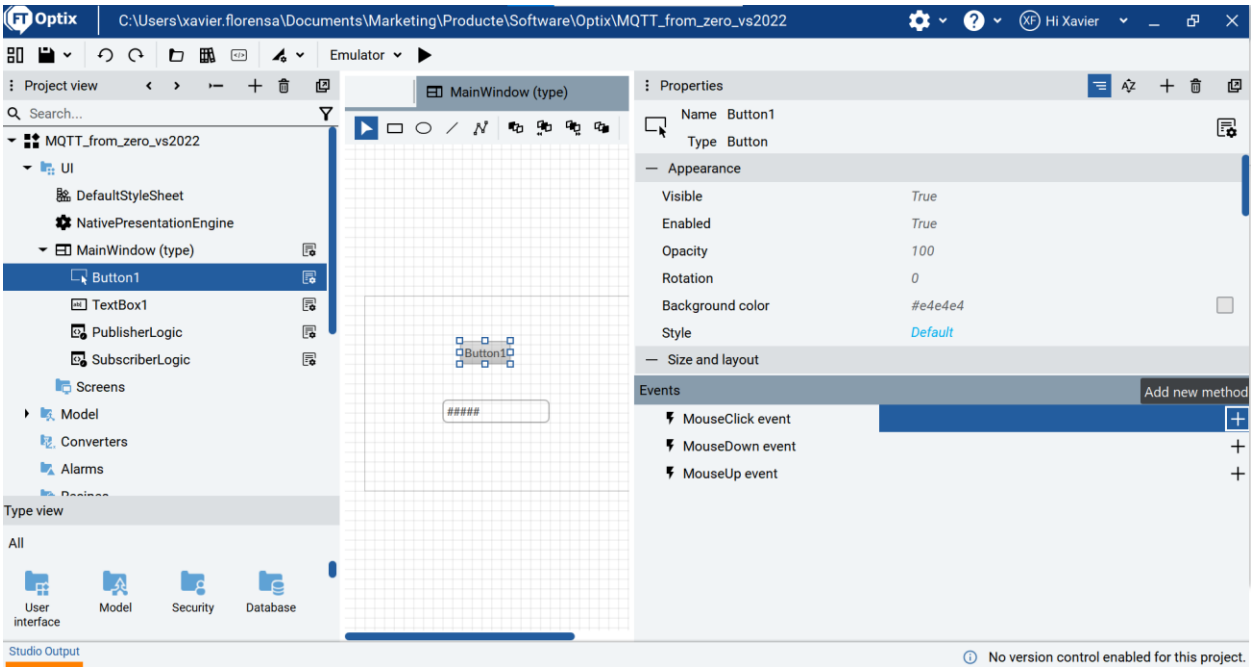


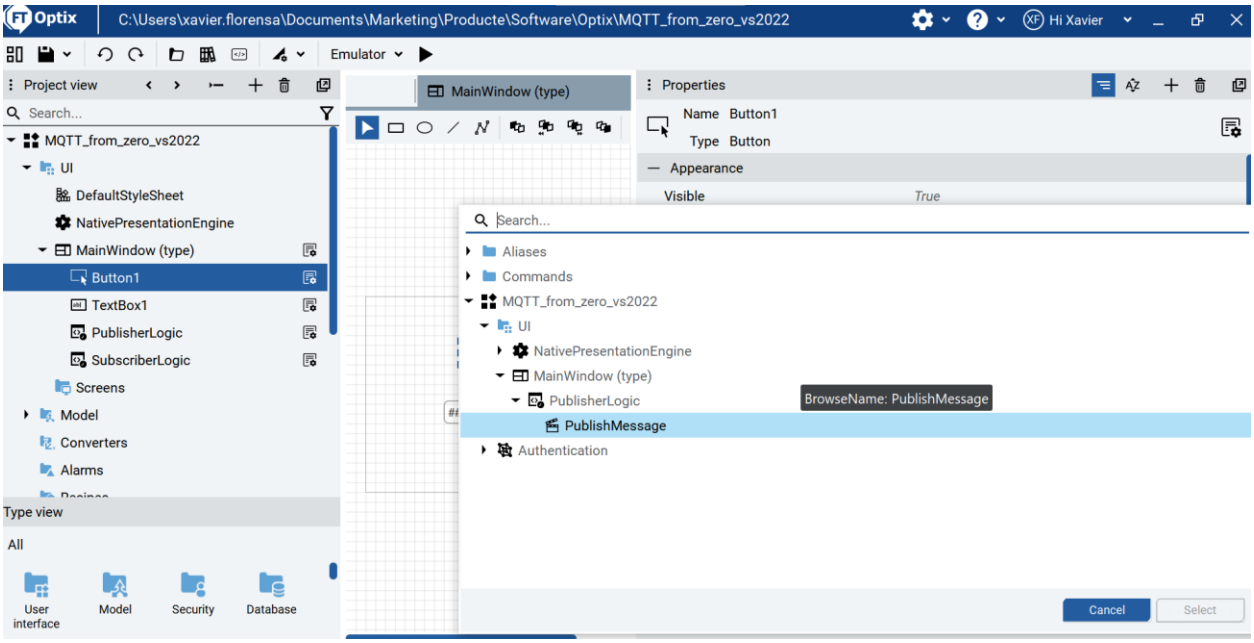
Open FT Optix
Open Publisherlogic
We have no errors now



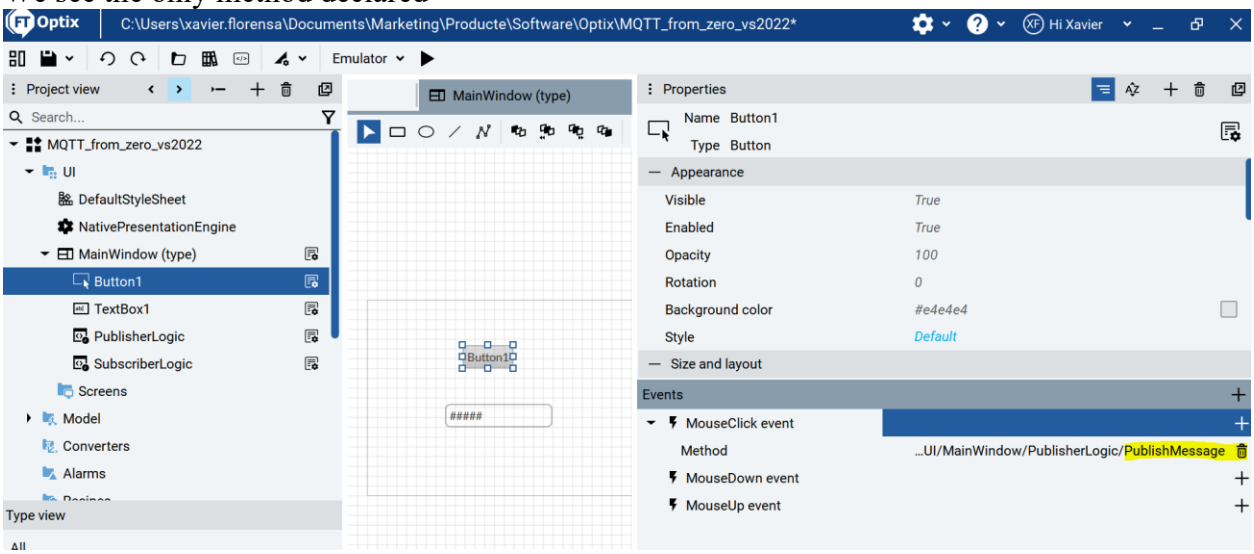
28.3.3. Complete your MQTTnet application

Let’s build the mouseclick action

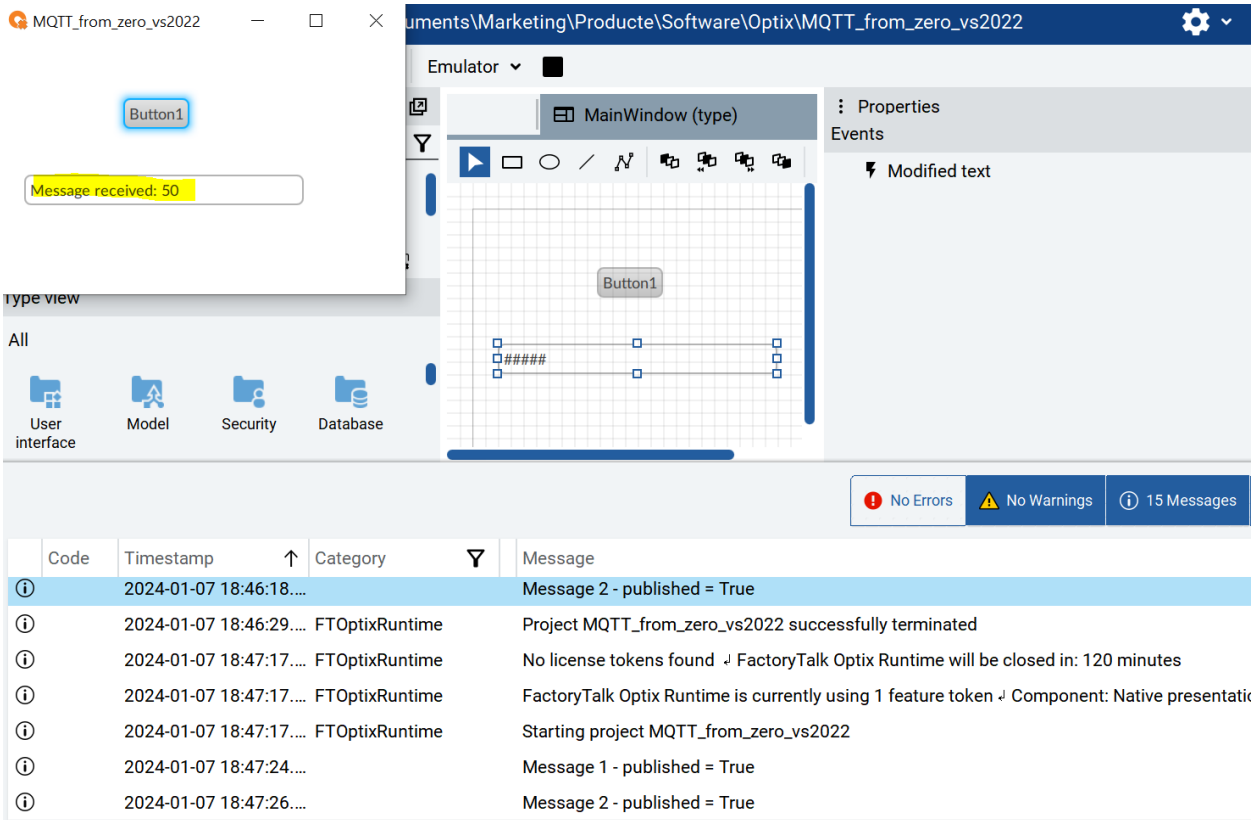




We see the only method declared



We can test the program
It works properly
Each time we click on the button, a random number is published and automatically received on the textbox thanks to the subscribe logic.



Not only we were able to know how to create the application from scratch, but also we see how to set up a callback function on the subscriberlogic to asynchronous run when a message appears. We can use this feature with our Socket examples or on our next MQTT applications.

```
// Assign a callback to be executed when a message is received from the broker
subscribeClient.MqttMsgPublishReceived += SubscribeClientMqttMsgPublishReceived;
```

```
1 reference
private void SubscribeClientMqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
{
    messageVariable.Value = "Message received: " + System.Text.Encoding.UTF8.GetString(e.Message);
}
```

29. NetLogic and C# tutorial

Take a look at these videos
Basics
https://www.youtube.com/watch?v=ZCLcgSGX1YU

Design time Netlogic
https://www.youtube.com/watch?v=DMC5OB__D_c

Runtime Netlogic
https://www.youtube.com/watch?v=EEExU0MsbtI&t=2076s

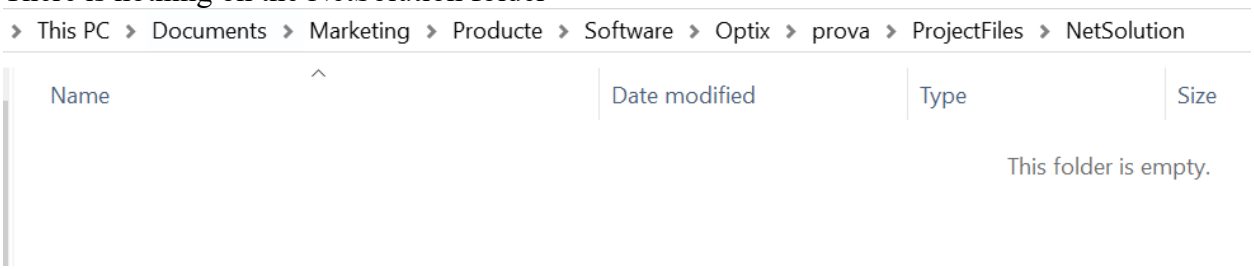
https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet
here you will find these concepts explained

README MIT license

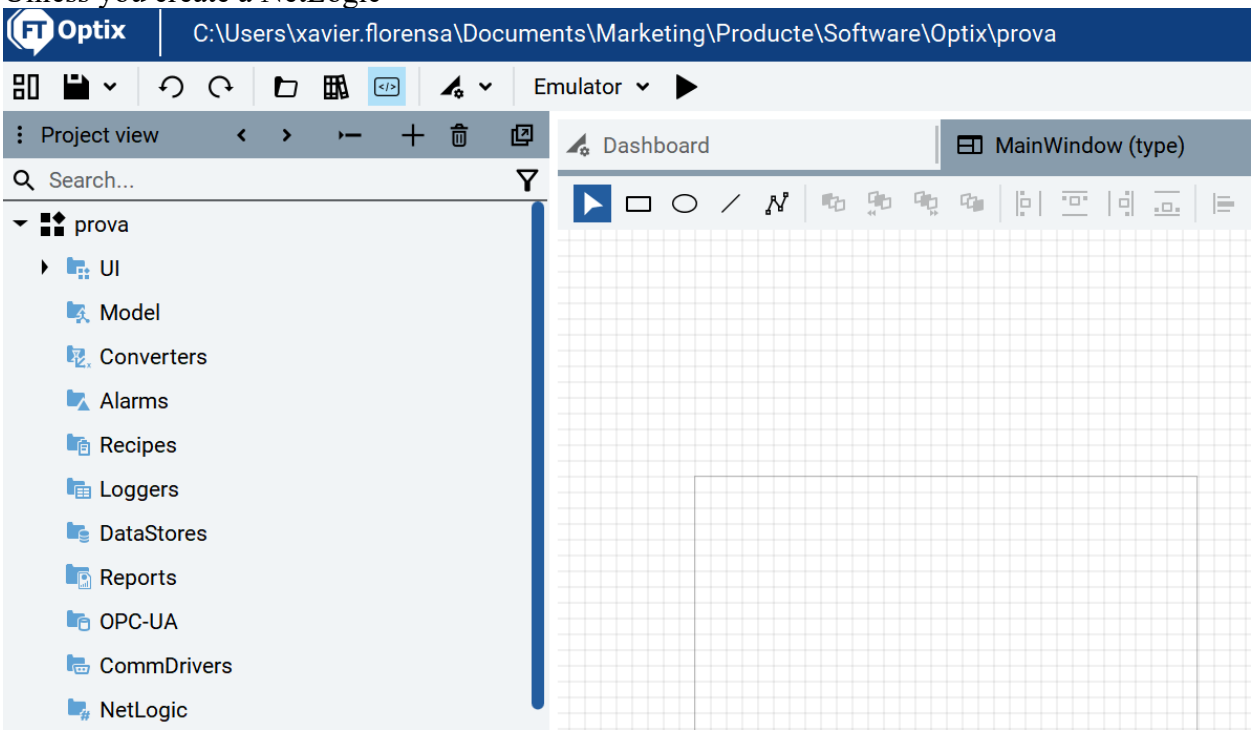
- General FT Optix good practices
- NetLogic overview
- Log output
- Accessing project nodes
- Managing aliases
- Objects, types and instances
- Asynchronous tasks
- Database interaction
- Random generation
- Resource URI
- Variables formatting
- RegEx
- DialogBox
- Variables Interaction
- Execute commands
- Sessions
- Translations
- Users and groups
- Mouse handling
- Dynamic Links
- Colors

30. Understanding compilation of NetLogic code

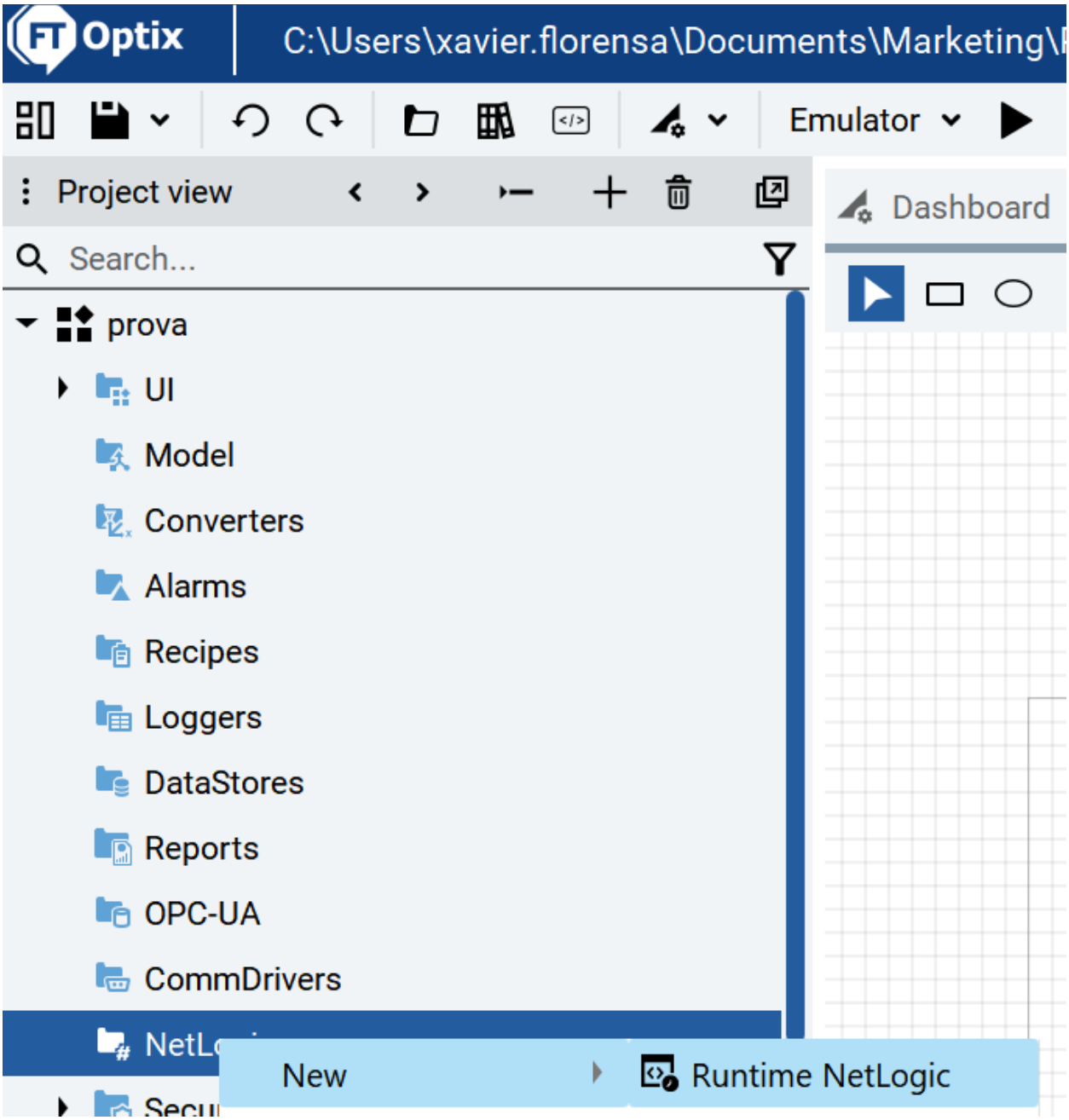
After you create a new standard project, before creating a Netlogic script and even compiling, you will have nothing on folder Projects/Netlogic/bin
Let's create a project in FT Optix Studio called prova
There is nothing on the NetSolution folder



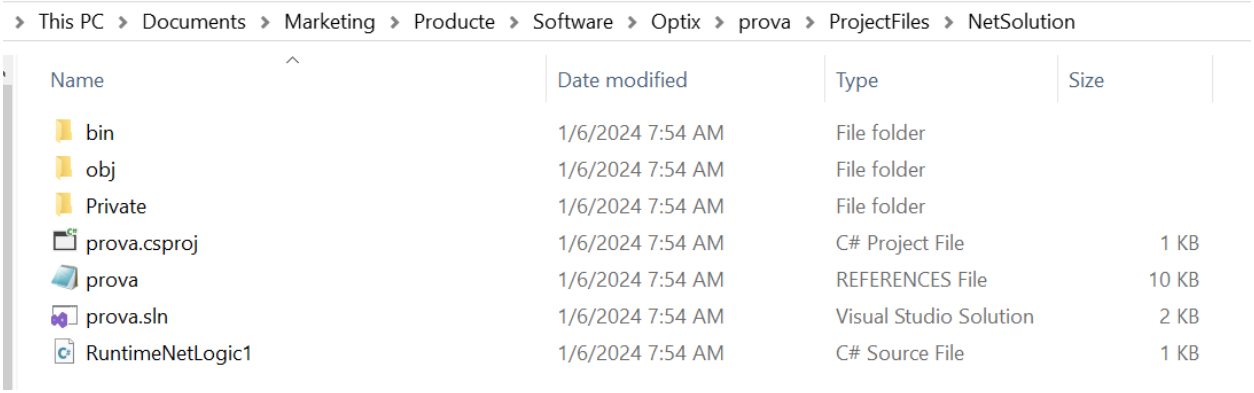
Unless you create a NetLogic



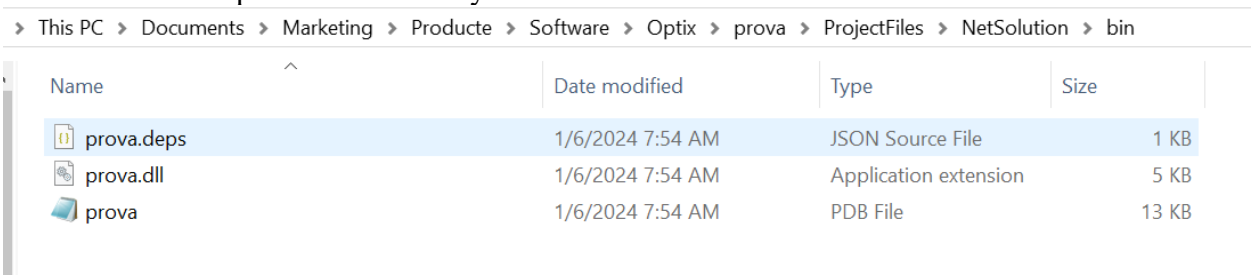
Now let's create a new Netlogic script



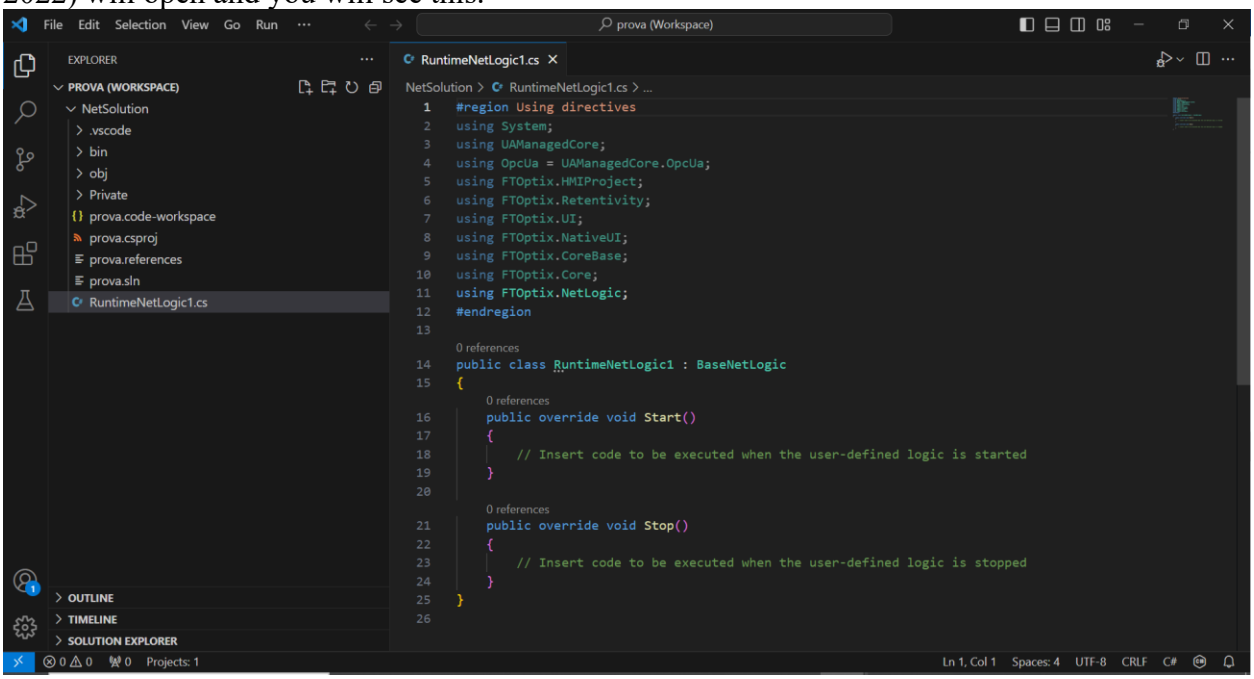
Look how is this populated
You will see the file RuntimeNetLogic1 where you have the script.



If you look on bin, it is also populated, since creating a new Netlogic also means compiling it. These are the compiled files in binary.



When you make changes on the code, you need to compile it, before executing, this is done with save on Visual Studio Code, or with Build in Visual Studio 2022.
If you click on FT Optix Studio on RunTimeNetLogic1 the Visual Studio code (or Visual Studio 2022) will open and you will see this.



31. Understanding Classes (Objects), methods and data

When you write an object or a class, (classes are marked in green color)

For ejample

```
public override void Start()
{
    // Insert code to be executed when
    Log.
}
```

Debug
Equals
Error
Info
Node
ReferenceEquals
Verbose1
Verbose2
Warning

Or for ejample (an instance of a class is marked in sky blue)

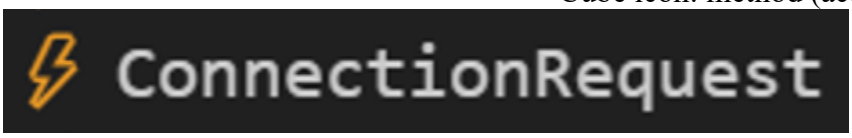
```
}

0 references
public void Star
//public overrid
{
    winsock_Ear.
```

BufferSize
Close
Connect
Connected
ConnectionRequest
Container
DataArrival
Disconnected
Dispose
Disposed

then you will write . and you will see (thank you to IntelliSense) a collection of

BufferSize Tool icon: read write access data

Cube icon: method (action to perform a task)



Blizard: event

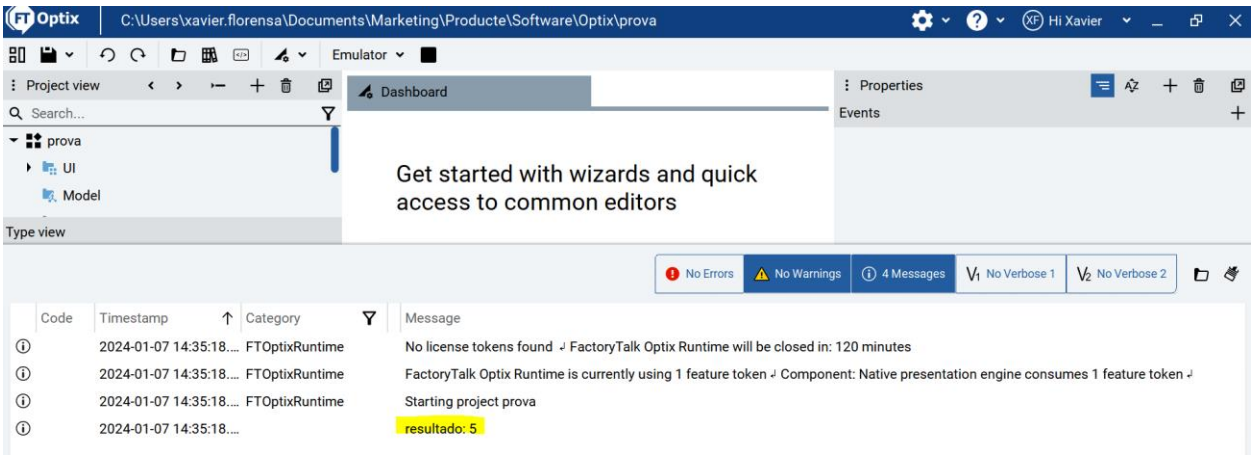
32. Creating a Method

Imagine we want to create a new Method to add 2+3=5
Create a new Optix project, and add a new Runtime Netogic
You have to use the return statement
Copy and paste this code (or complete with what you have on the code)

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public int metodo_suma(int primero, int segundo)
    {
        return primero + segundo;
    }
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        int resultado = metodo_suma (2,3);
        Log.Info("resultado: "+resultado);
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
```

```
}
```

Save and run emulator



We could use this method after executing the method of pushing a button.
Let's do this
Add this sentence in the same space

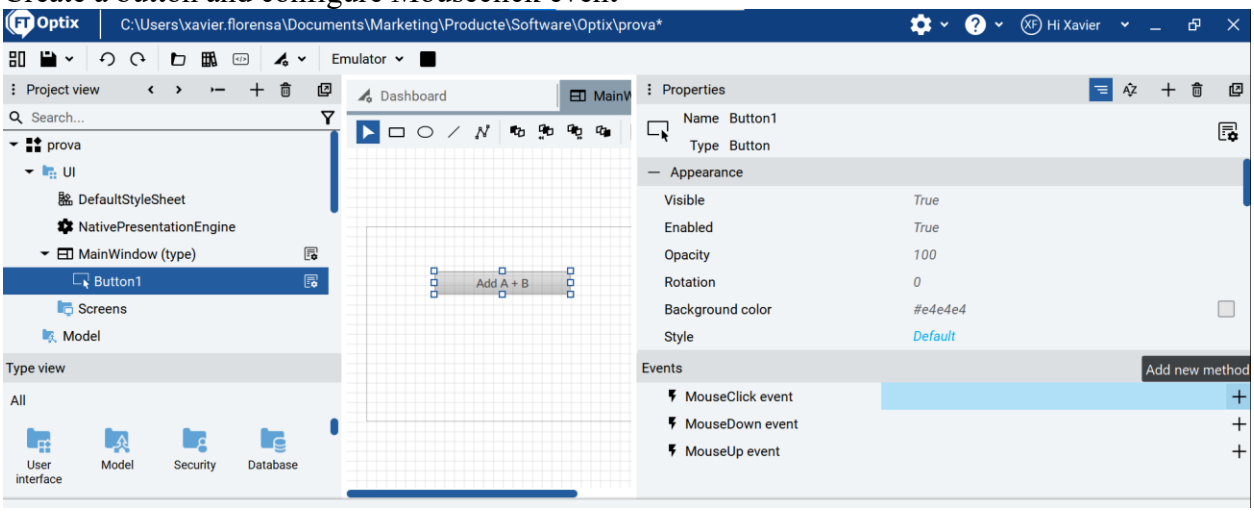
```
 [ExportMethod]

    public void Add(int sum1, int sum2)
    {
       Log.Info("Resultado tras pulsar botón: "+metodo_suma(sum1,sum2));
    }
```

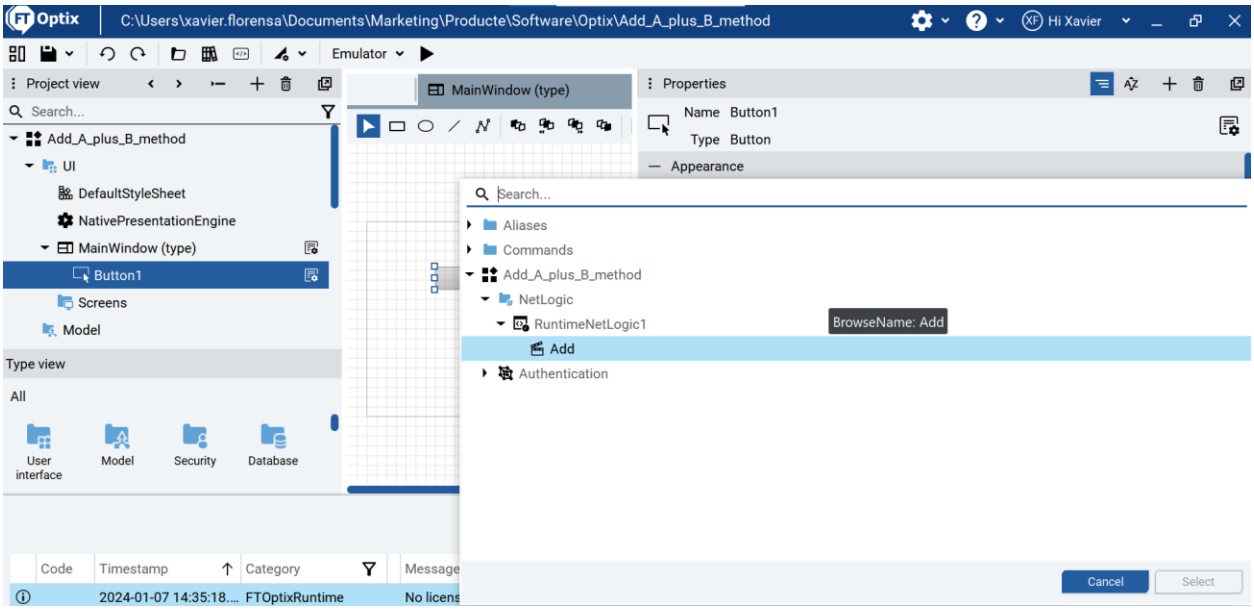
So the complete code will be like this

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public int metodo_suma(int primero, int segundo)
    {
        return primero + segundo;
```

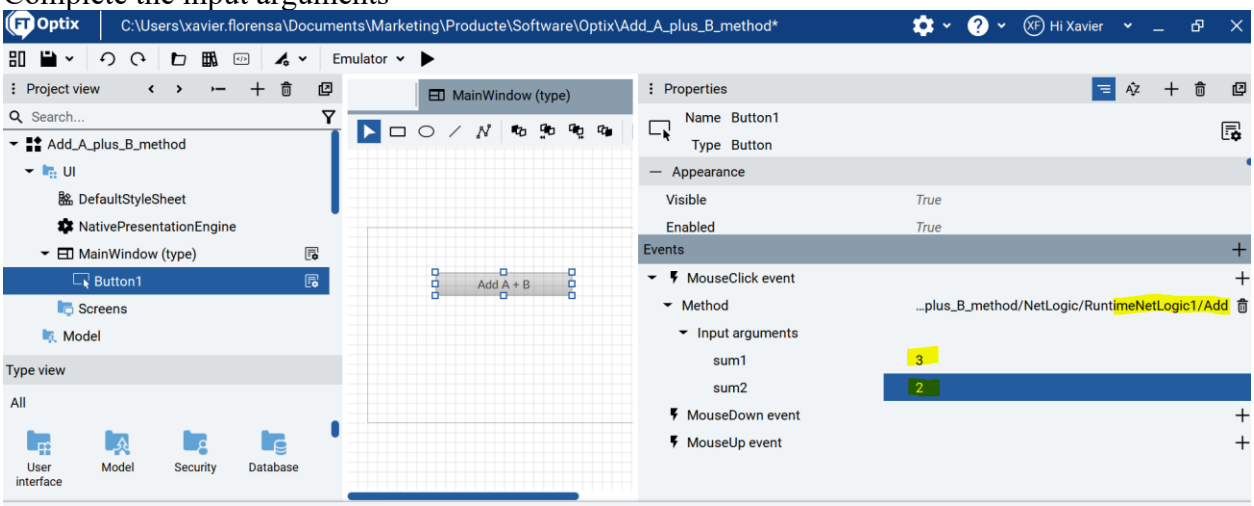
```
    }
    [ExportMethod]

    public void Add(int sum1, int sum2)
    {
       Log.Info("Resultado tras pulsar botón: "+metodo_suma(sum1,sum2));
    }

    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        //int resultado = metodo_suma (2,3);
        //Log.Info("resultado: "+resultado);
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Save on VS Code
You have to configure this button click action on your Optix environment
Create a button and configure Mouseclick event

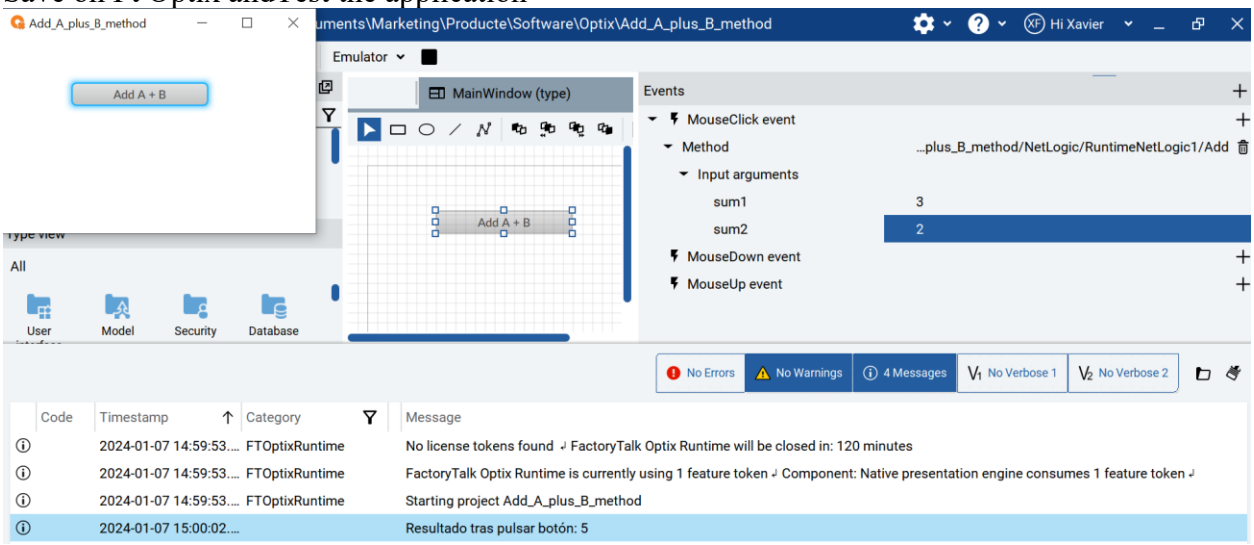




Complete the input arguments



Save on Ft Optix andTest the application



33. Linking events with Methods without input parameters

Create a new project and on Runtime Netlogic copy and paste this

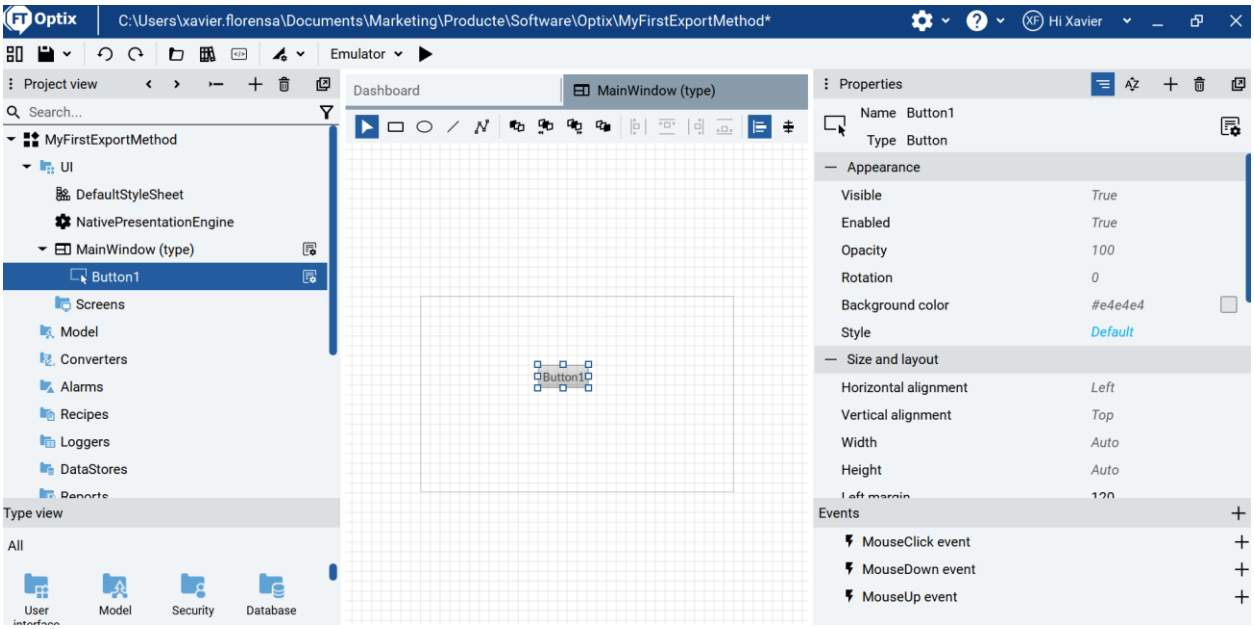
```
 [ExportMethod]

    public void MyNewMethod()
    {

    }
```

Populate with the task you want to execute
For instance

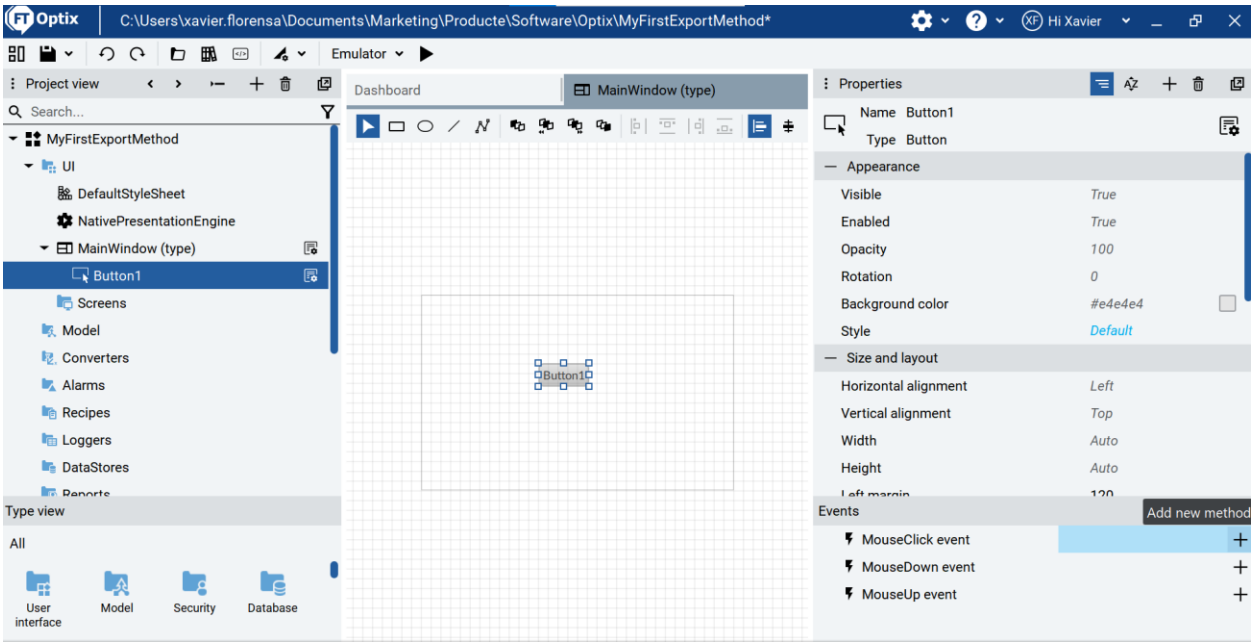
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    [ExportMethod]

    public void MyNewMethod()
    {
        Log.Info("Hello World, a button has been pressed");
    }

    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

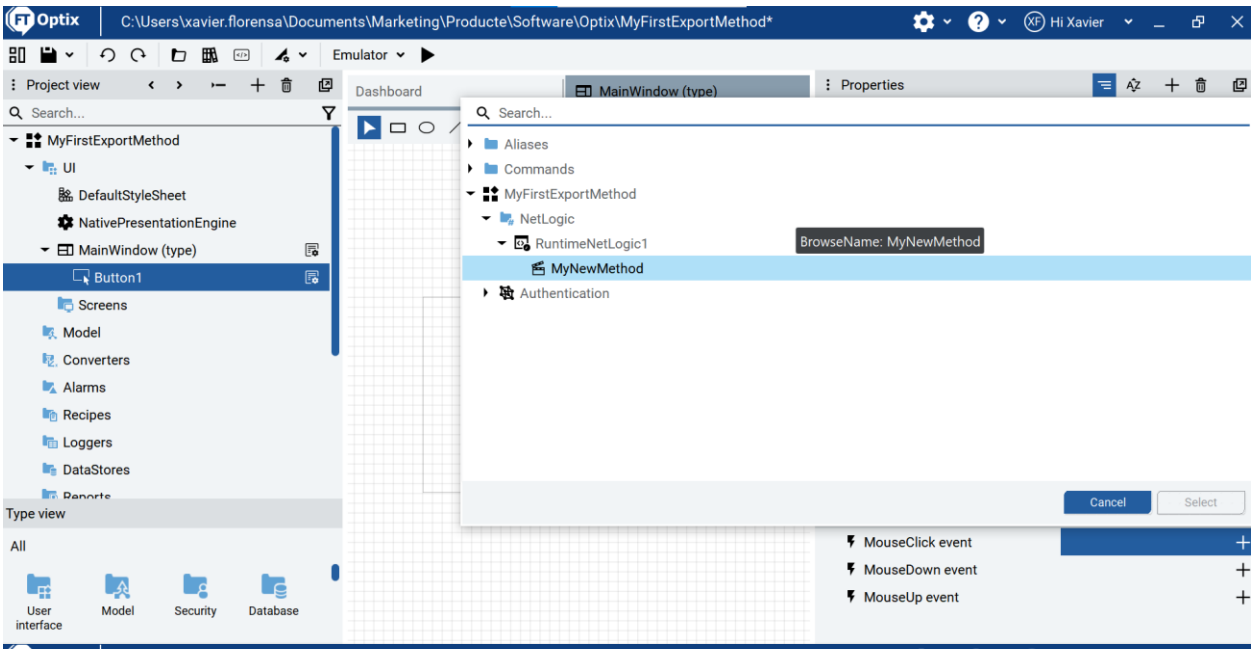
Next, save the code (in order for the complier to build the new Method)

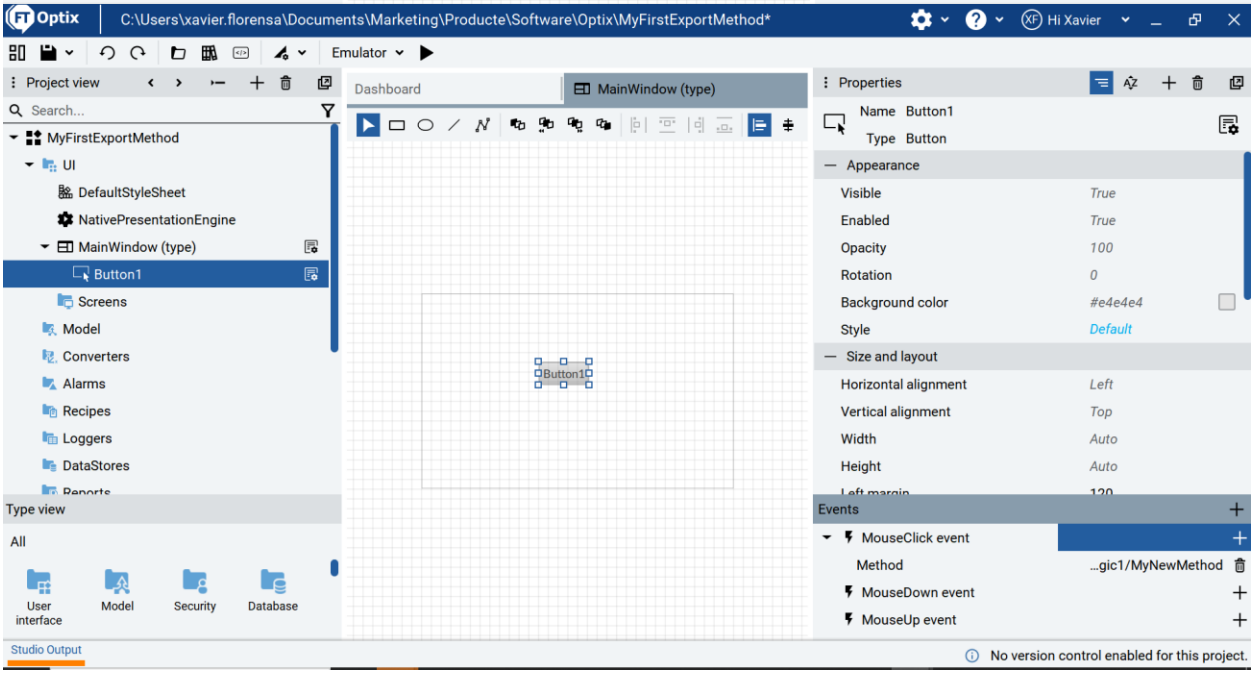
Add a new button



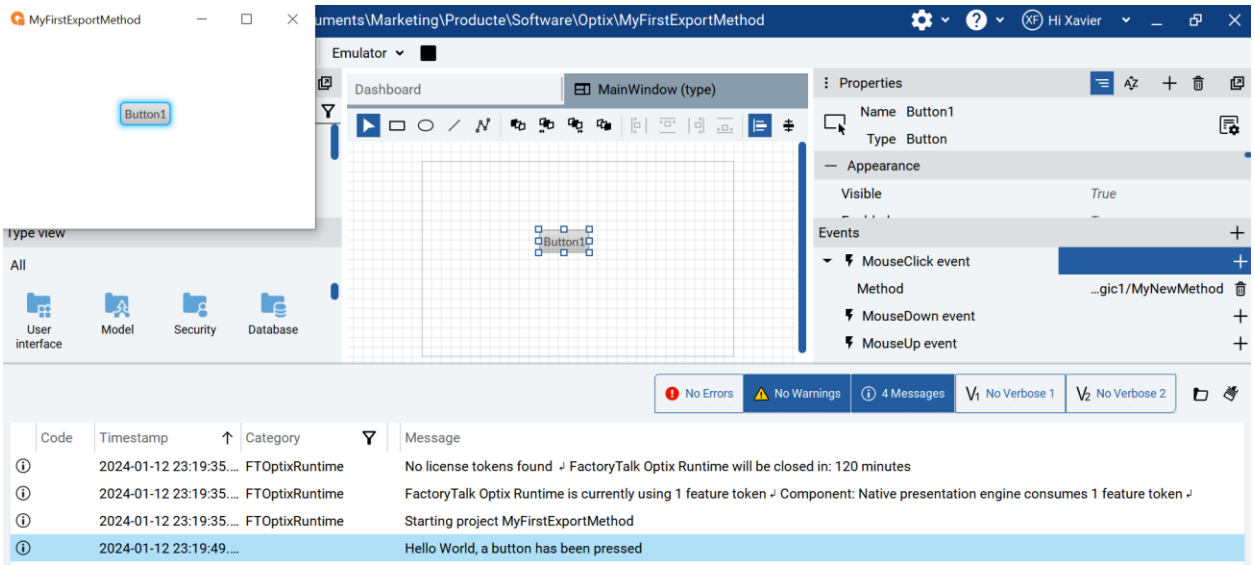
Add a new MouseClick event







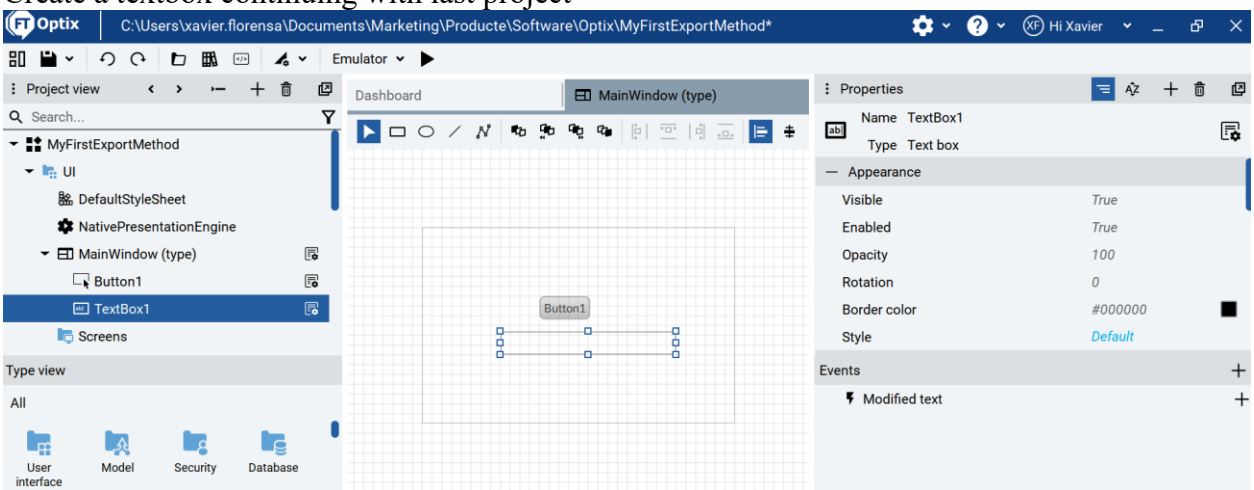
Now execute the project



Next you would like to write the text that you will output

34. Linking events with Methods with input parameters

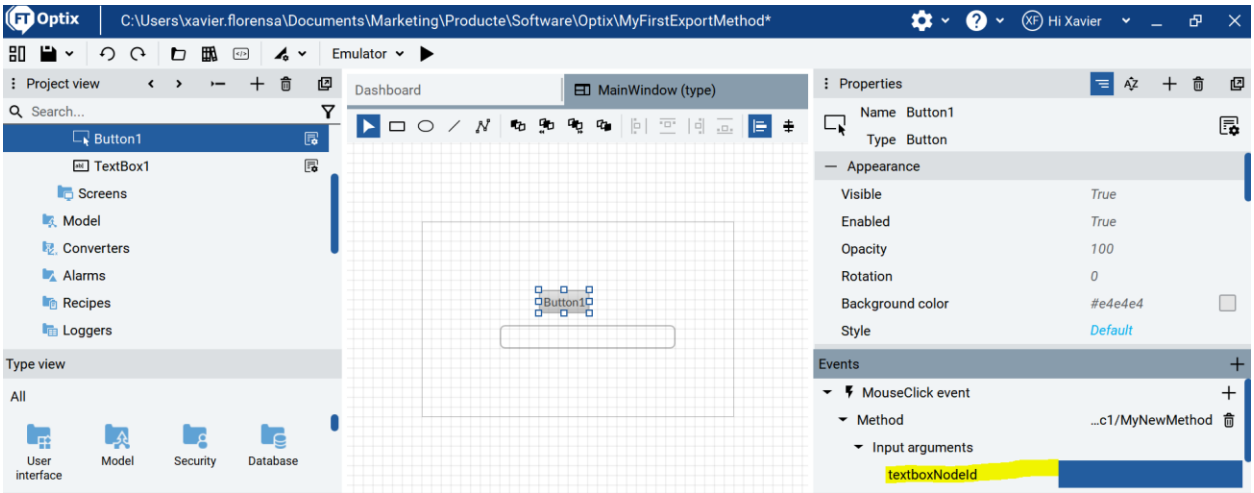
Create a textbox continuing with last project



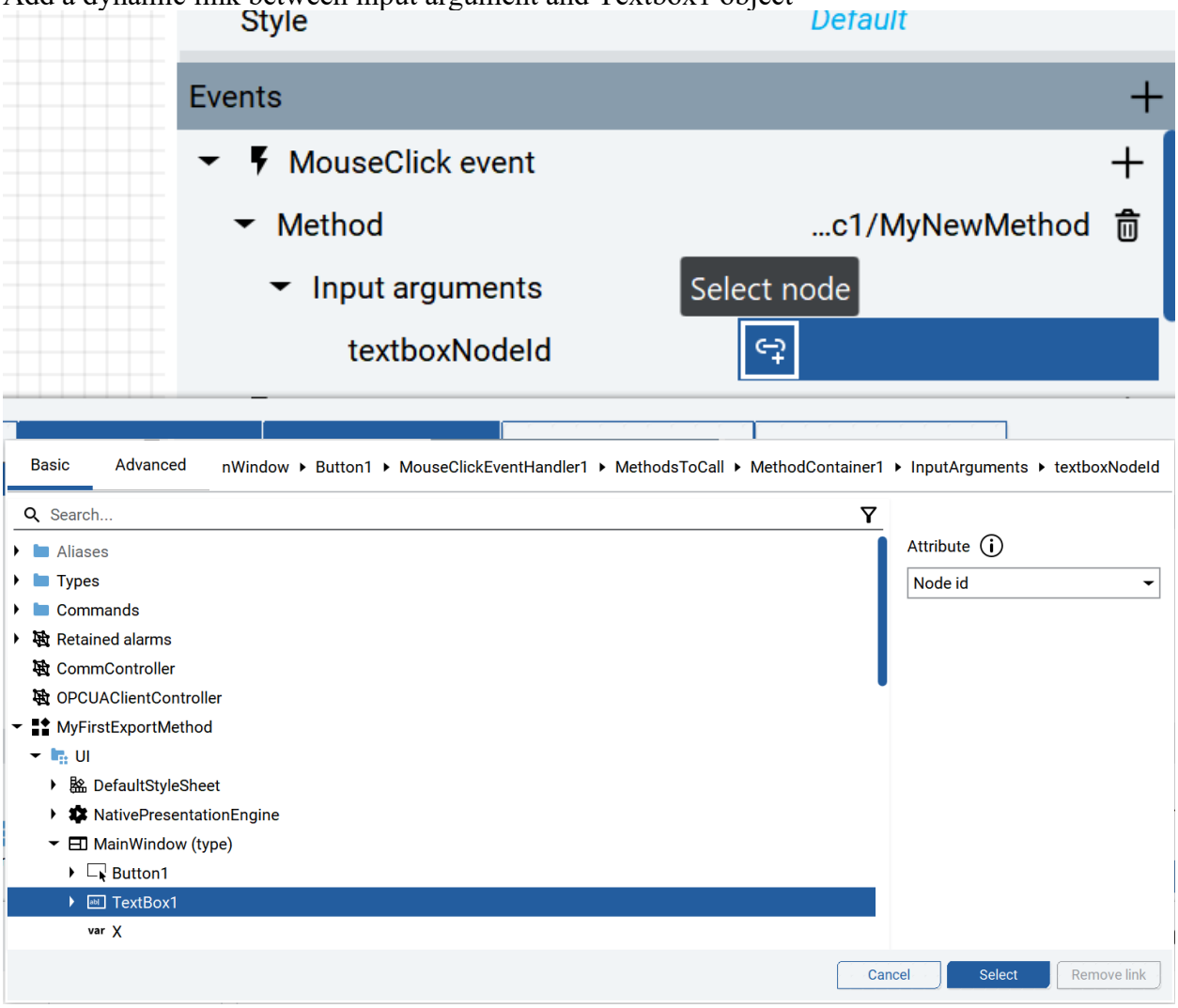
Add this code to the Netlogic code

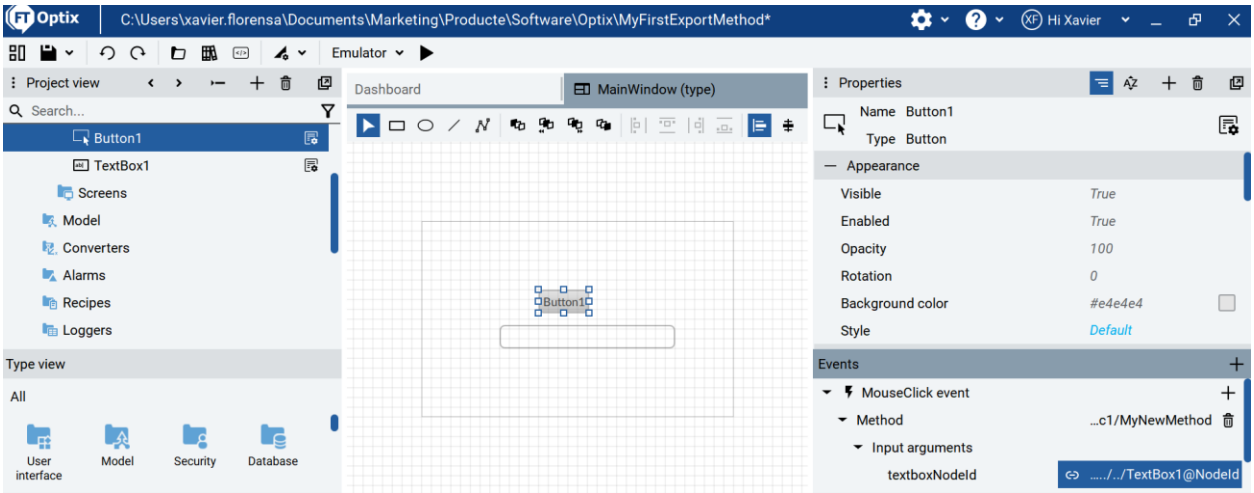
```
[ExportMethod]

    public void MyNewMethod(NodeId textboxNodeId)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        string messagetosend = textbox.Text;
        Log.Info(messagetosend+"Hello World, a button has been pressed");
    }
```

Save the code
Now when you add a mouseclickbutton you will find an inputparameter



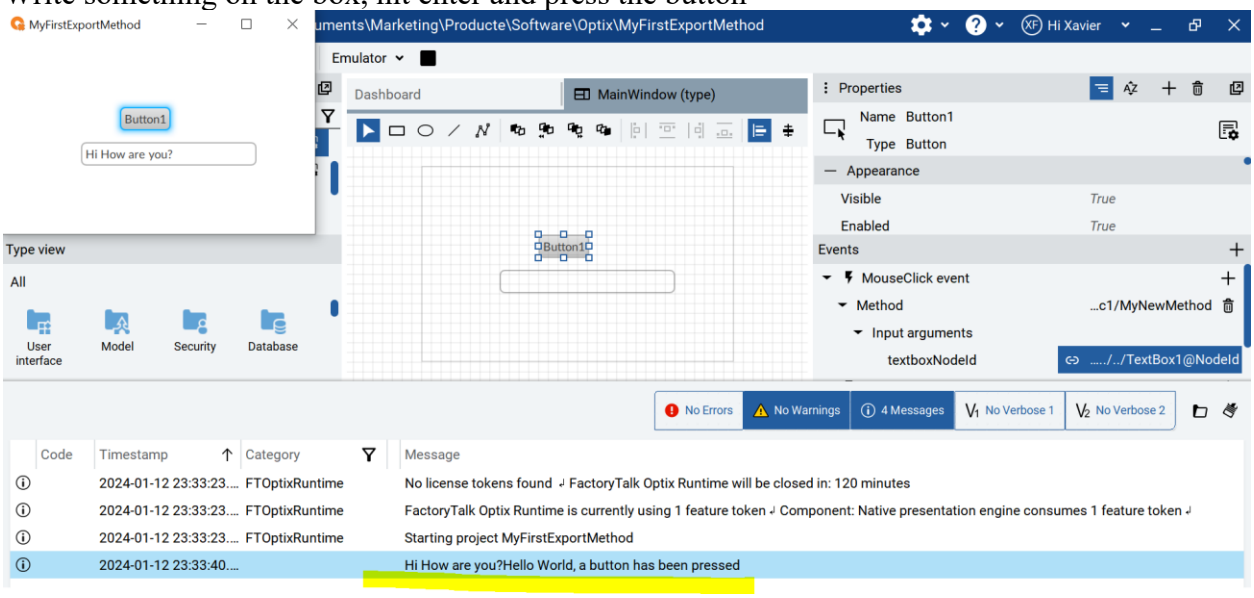
Add a dynamic link between input argument and Textbox1 object





Execute the code

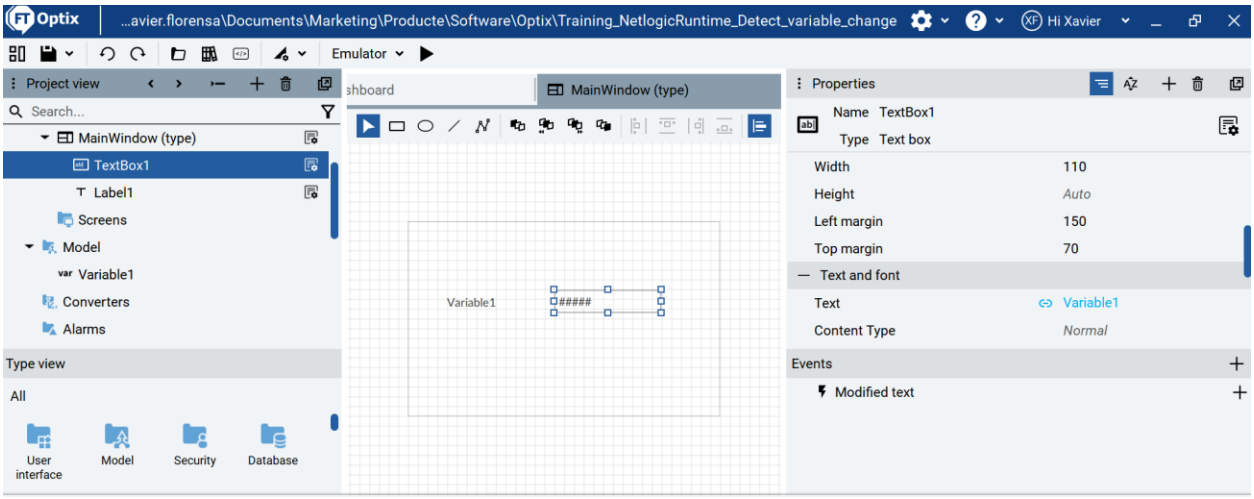
Write something on the box, hit enter and press the button



35. Asynchronous tasks, creating callback functions

Let's create a function to detect a variable change, without polling. This is called a callback function.

Create a new project with a textbox and a variable

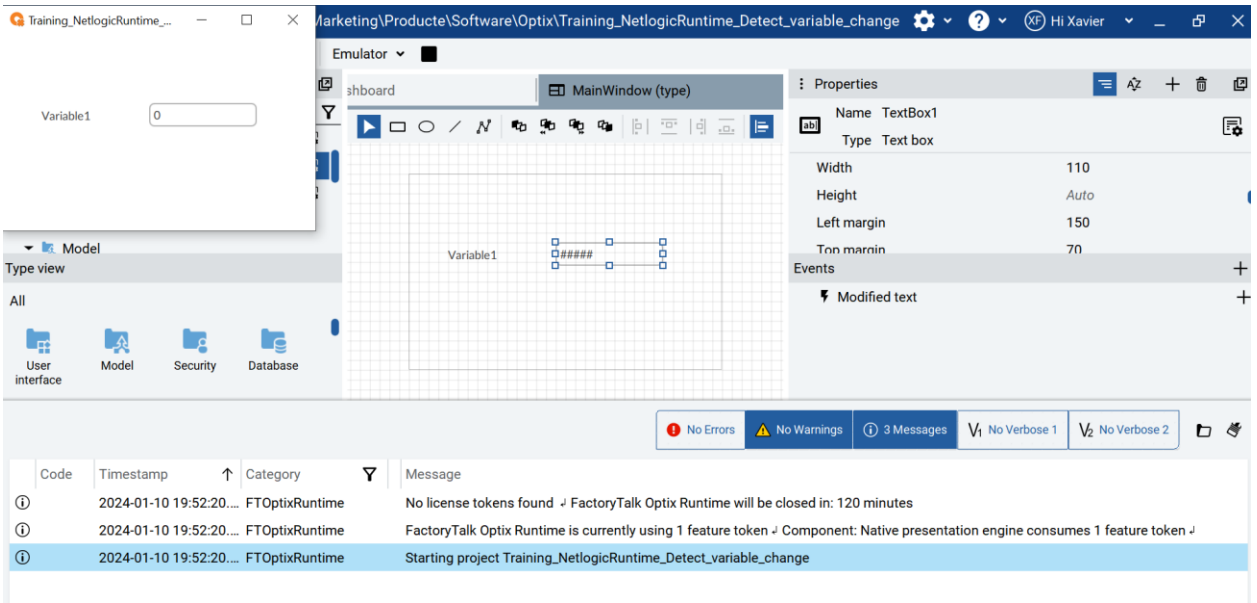


Create this Netlogic

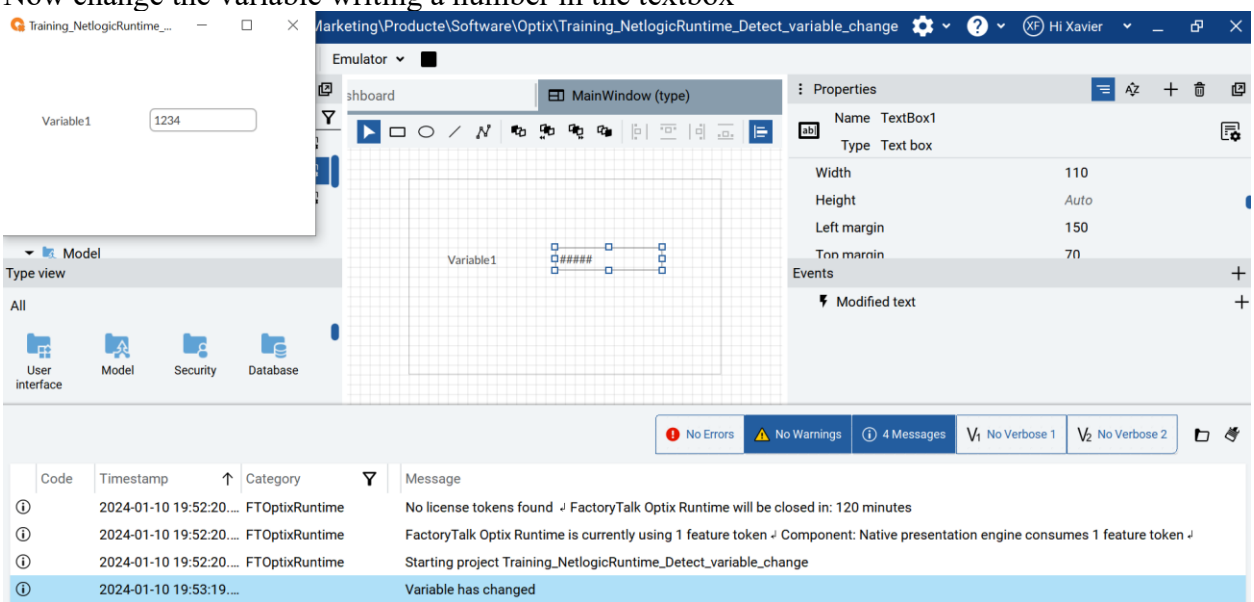
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    private IUAVariable variable1;
    public override void Start()
    {
        variable1 = Project.Current.GetVariable("Model/Variable1");
        //assing a callback function to be executed when the variable changes
        variable1.VariableChange += variable1_VariableChange;
    }

    public override void Stop()
    {
    }
    private void variable1_VariableChange(object sender, VariableChangeEventArgs
e)
    {
        Log.Info("Variable has changed");
    }
```

```
}
```

Let’s execute the code



Now change the variable writing a number in the textbox



You can access other methods, events and data for the variable,
Take a look, if you write the instance, followed by . you will see all:

```
variable1.
                    ★ VariableChange
                    ★ SetConverter
erences             ★ Children
lic overrid         ★ ValueRank
                    ★ FindVariable
                      ActualArrayDimensions
erence                ActualDataType
vate void v           Add
                      ArrayDimensions
 Log.Info("           BrowseName
                      Children
                      ChildrenRemoteRead
```

```
variable1.
                      ChildrenRemoteWrite
                      DataType
rences                DataValue
ic overrid            Delete
                      Description
                      DisplayName
rence                 Equals
ate void v            Find
                      Find<>
Log.Info("            FindByType<>
                      FindNodesByType<>
                      FindObject
```

```
variable1.
erences
lic overrid
erence
vate void v
Log.Info("
```

- FindVariable
- Get
- Get<>
- GetAlias
- GetByType<>
- GetHashCode
- GetNodesByType<>
- GetObject
- GetPanelLoader
- GetType
- GetVariable
- IsInstanceOf

```
variable1.
erences
lic overrid
erence
vate void v
Log.Info("
```

- IsInstanceOf
- IsValid
- MoveDown
- MoveUp
- NodeClass
- NodeId
- Owner
- Prototype
- QualifiedBrowseName
- Quality
- RemoteRead
- RemoteWrite

```
variable1.
                Remove
                ResetDynamicLink
erences         ServerTimestamp
lic overrid     SetAlias
                SetConverter
                SetDynamicLink
erence          SourceTimestamp
vate void v     Start
                StatusCode
 Log.Info("     Stop
                ToString
                Value
```

```
ValueRank
VariableChange
VariableType
```

So wich input argument must be written on the callback function to make it work properly?
For instance what to write here?

```
private void variable1_VariableChange(object sender, "¿What to write here?" e)
```

Just hover your **mouse over** the `variable1.VariableChange` event and you will have the solution:

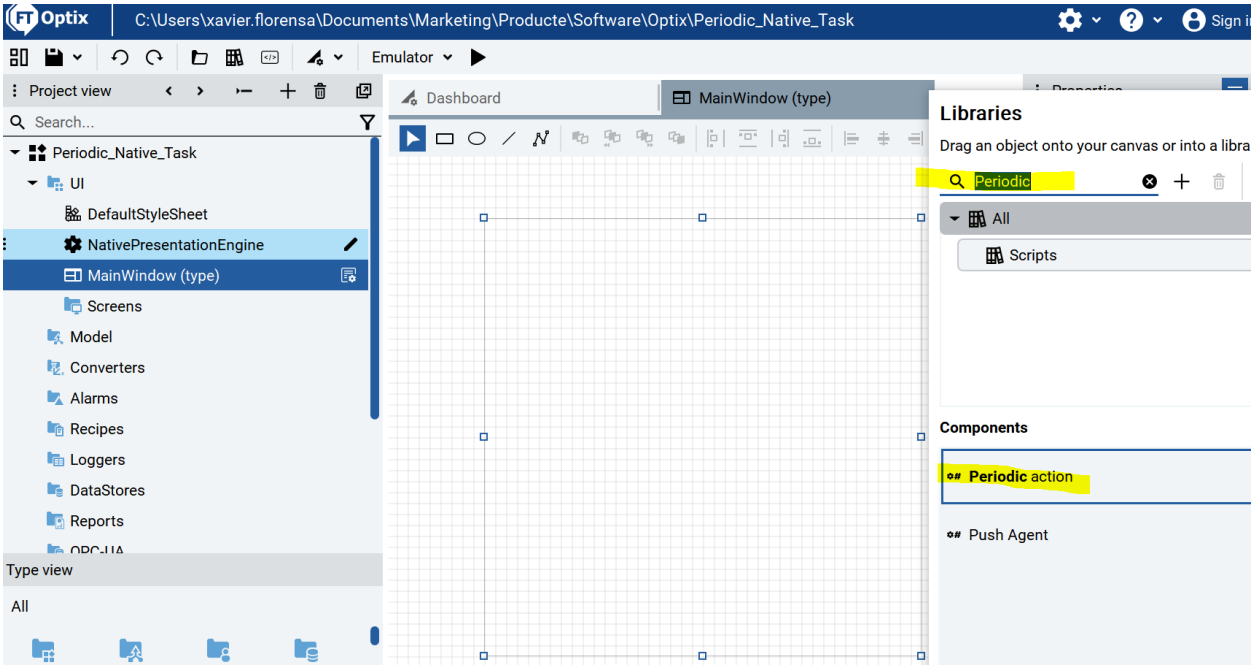
```
//assing a  EventHandler<VariableChangeEventArgs> IUAVariable.VariableChange
variable1.VariableChange += variable1_VariableChange;
```

That's all.

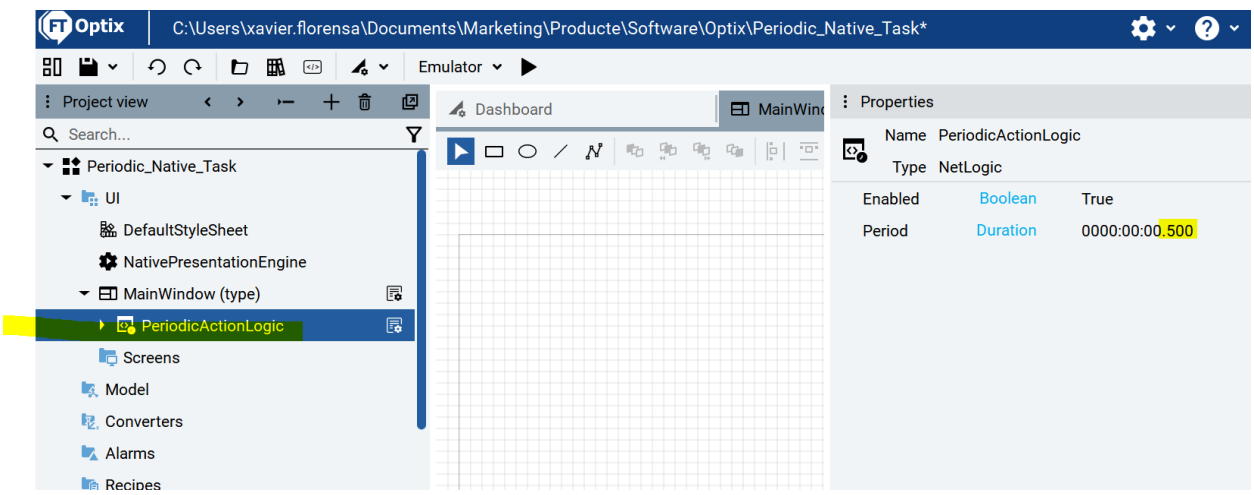
36. Periodic Tasks

36.1. Native Periodic Tasks

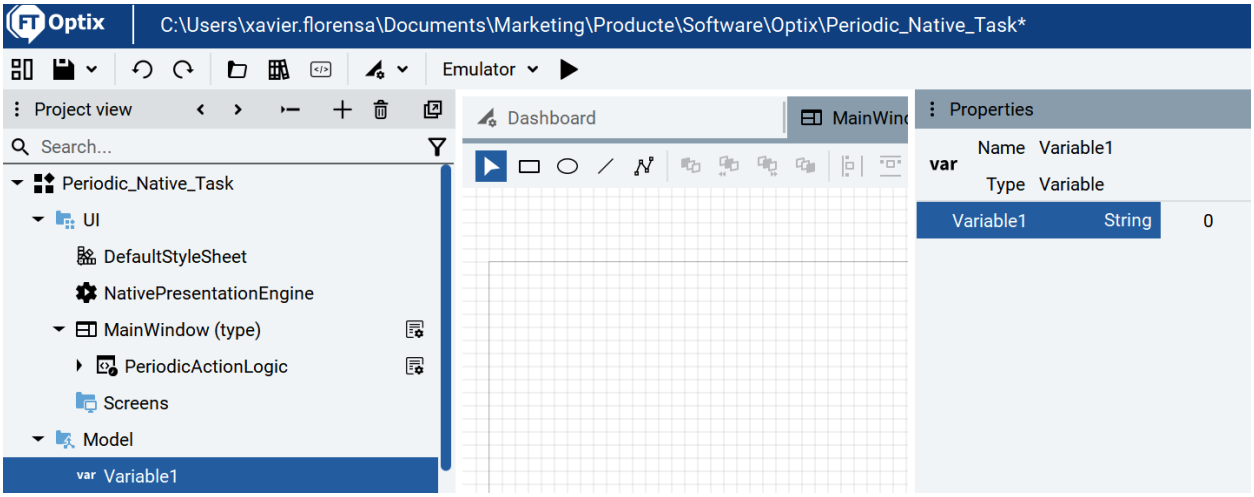
You have a library to do so
Let's create a blank project
Open library tool and search for Periodic



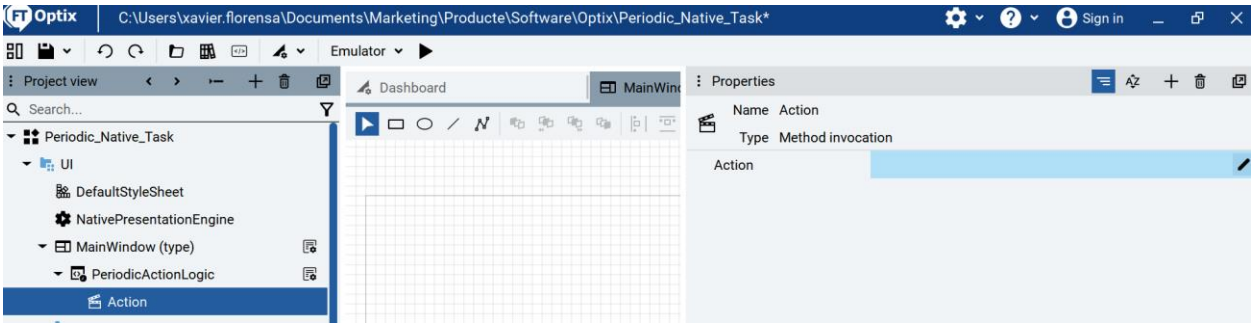
Drag and drop the "Periodic action" component to the desired scope, for instance, Main Window
A new PeriodicActionLogic script will appear under Mainwindow (but you do not need to go to C#)
You have the period, for instance each 500 mseconds



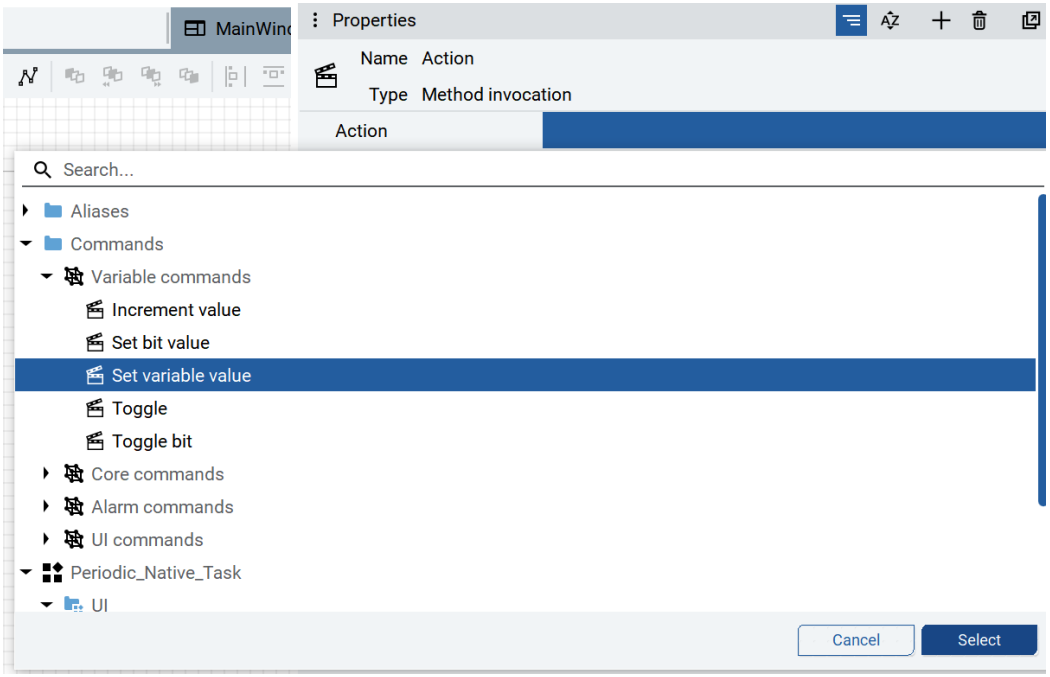
Then you can add an action, for instance modify a variable value
Add a variable to Model

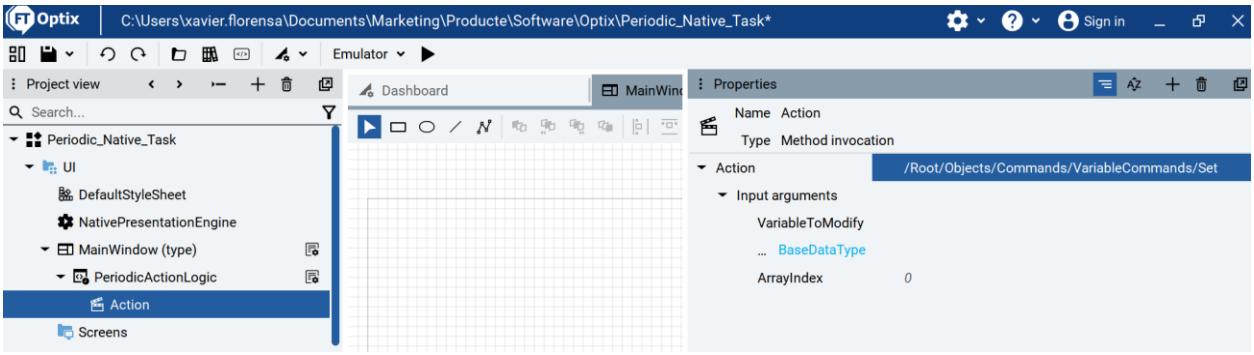


Then add an action to the Periodic
Click on ACtion
Click on the pen

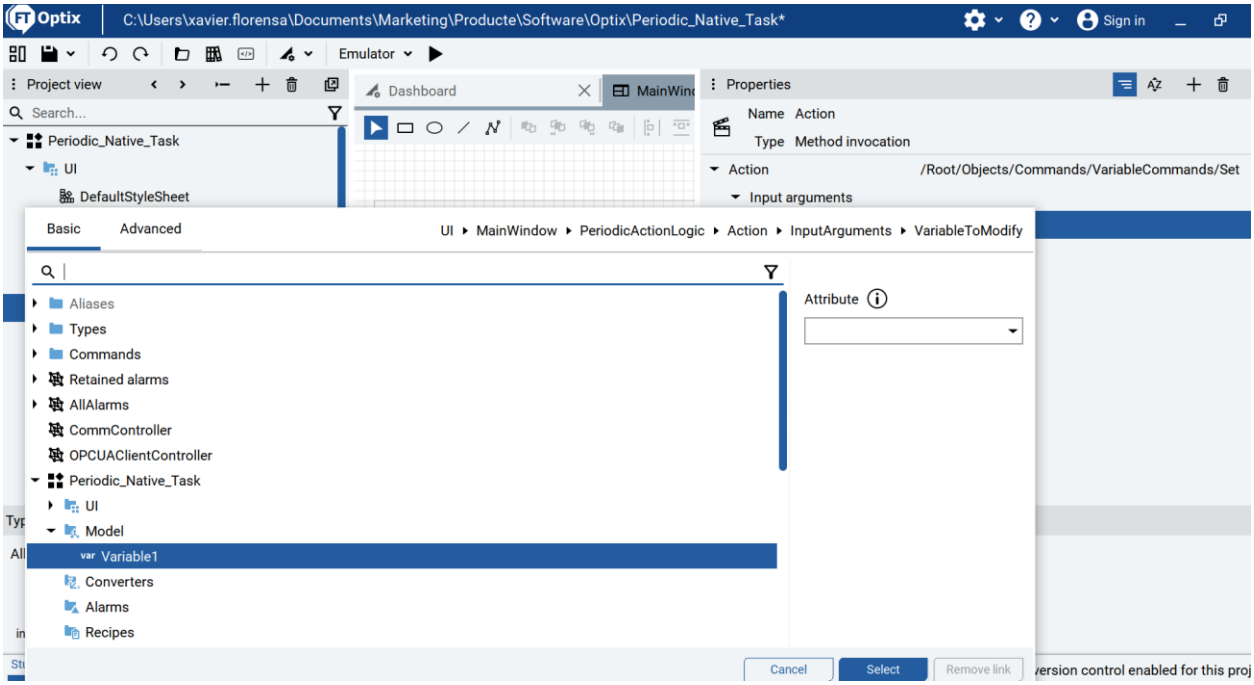


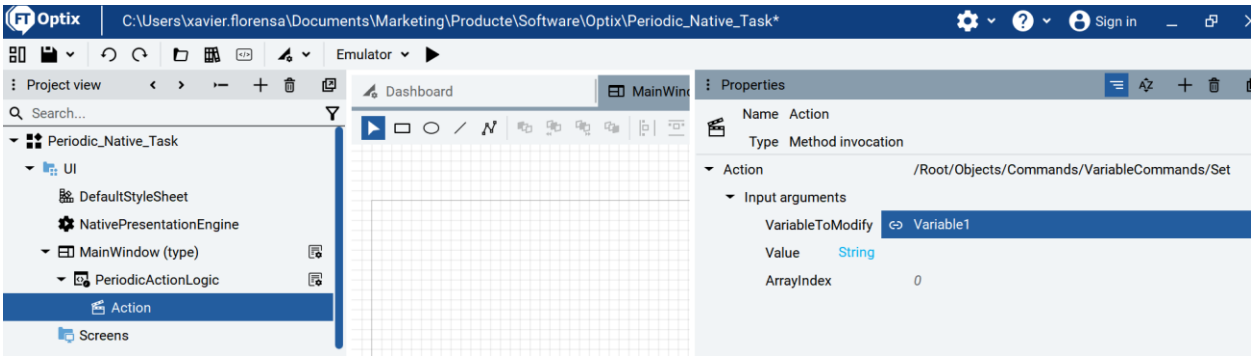
Add a commnad





Set VariableToModify





Select the value, for instance, logged user (you have this on users section) or whatever you want.

36.2. Netlogic Periodic Tasks

Let's assume we want a function called "my_Function" task to be executed each 1000 mseconds

We have to use these directives on our NetLogic

```
public class RuntimeNetLogic1 : BaseNetLogic
{
    PeriodicTask myTask1;

    public override void Start()
    {
        myTask1 = new PeriodicTask(my_Function, 1000, LogicObject);
        myTask1.Start();
 }
```

```
public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
        myTask1.Dispose();
    }
```

And write your code inside this function after the Stop() function

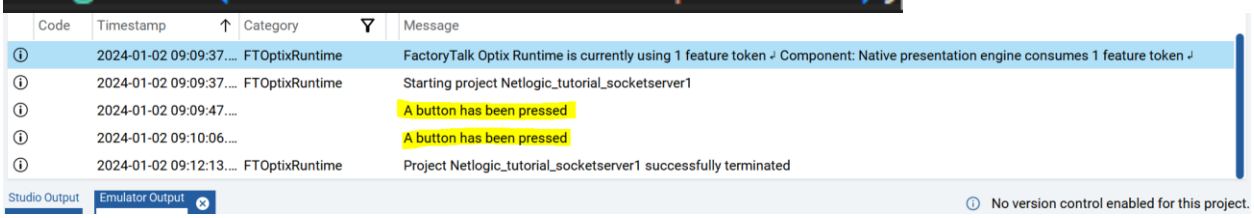
```
private void my_Function()
    {
    }
```

You can even use input and output parameters

37. Log info to emulator output

Just insert this sentence on your code

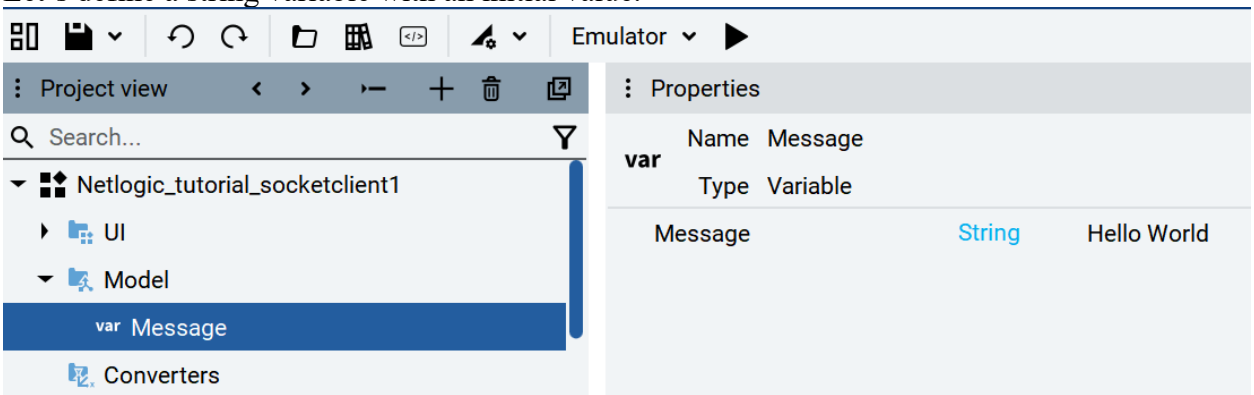
```
Log.Info("A button has been pressed");
```



https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/log-output.md

38. Accessing variables

Let's define a string variable with an initial value.



The simplest way to access variables in our program is once declared the variable "Message", with this sentence

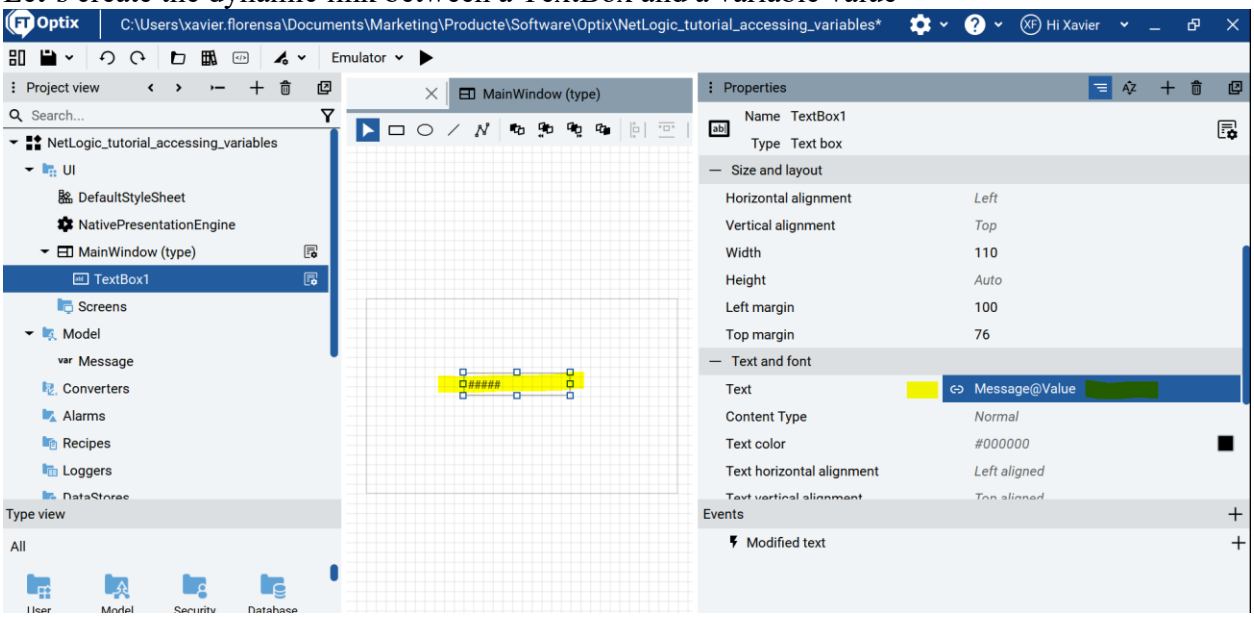
```
var missatge = Project.Current.GetVariable("Model/Message");
```

So we are coping the whole object "Message" variable with all attributes to the Netlogic variable "missatge"
But be careful, and aware that missatge is not a string type, it is an object.

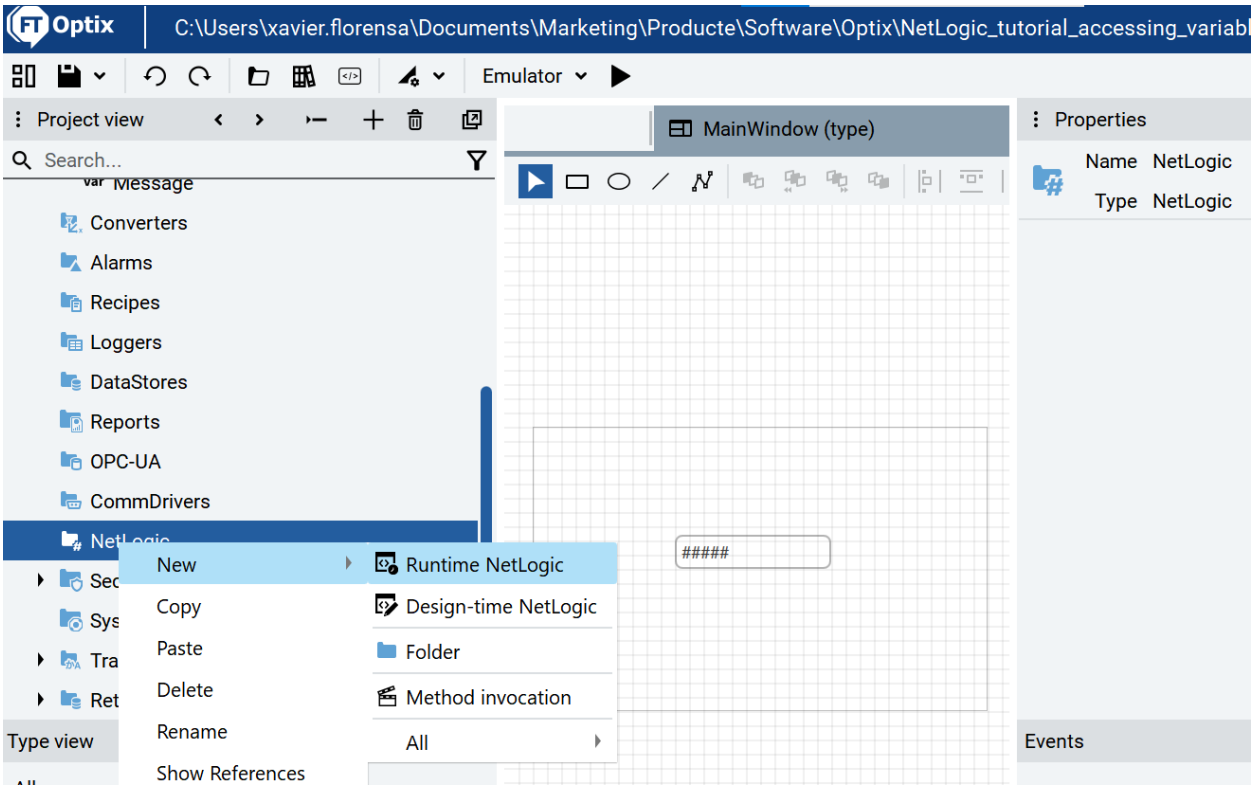
If you want to refer to the value, you have to do this way

```
missatge.Value;
```

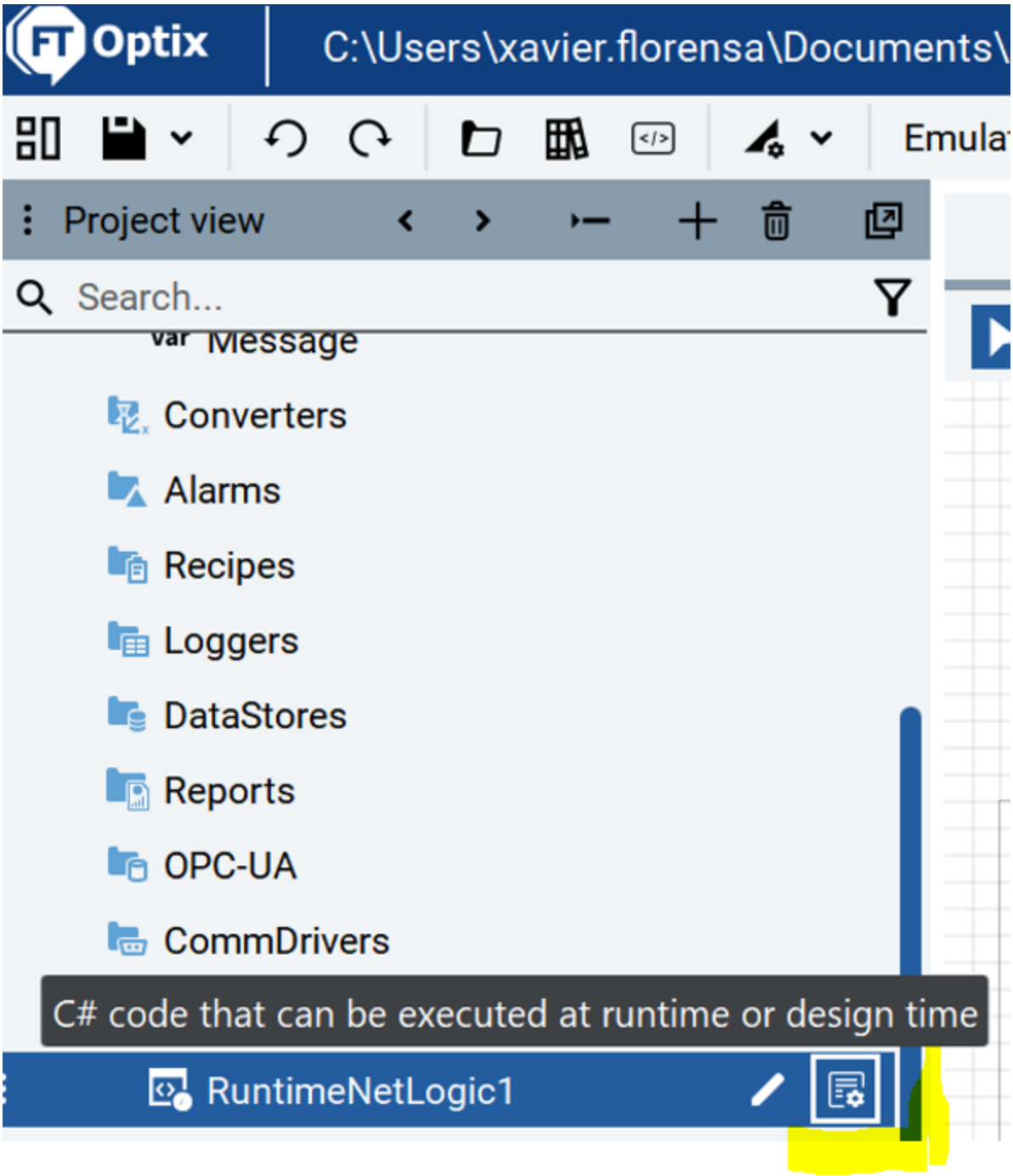
So if you want to copy the value to a string variable you do this way
String myNewMessage = missatge.Value
Or if you want to assign a new value in the code, you do this way
missatge.Value = “Hello, how are you?”
You can output the new code value to a User Interface Textbox with a dynamic link to the variable “Message”
Let’s try it with this project
Let’s create the dynamic link between a TextBox and a variable value



Now let’s create some Netlogic



Click on the icon with textlines and gearbox



The NetLogic code will open

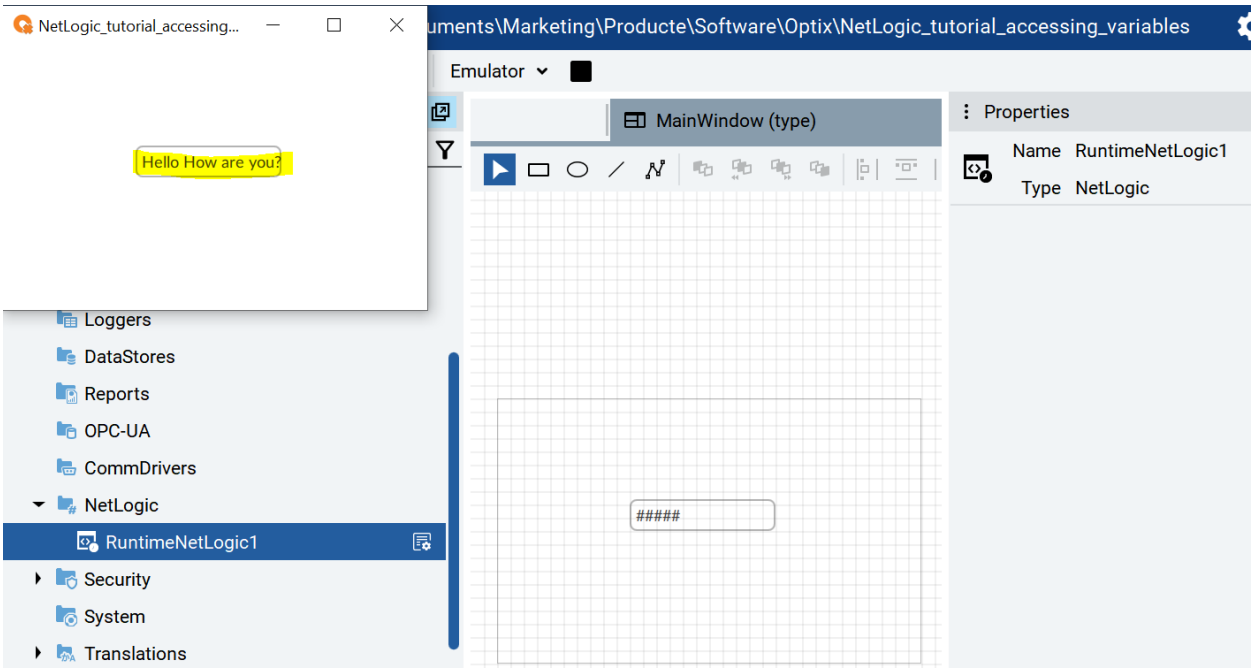
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
```

```
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Introduce such code on the Start() routine

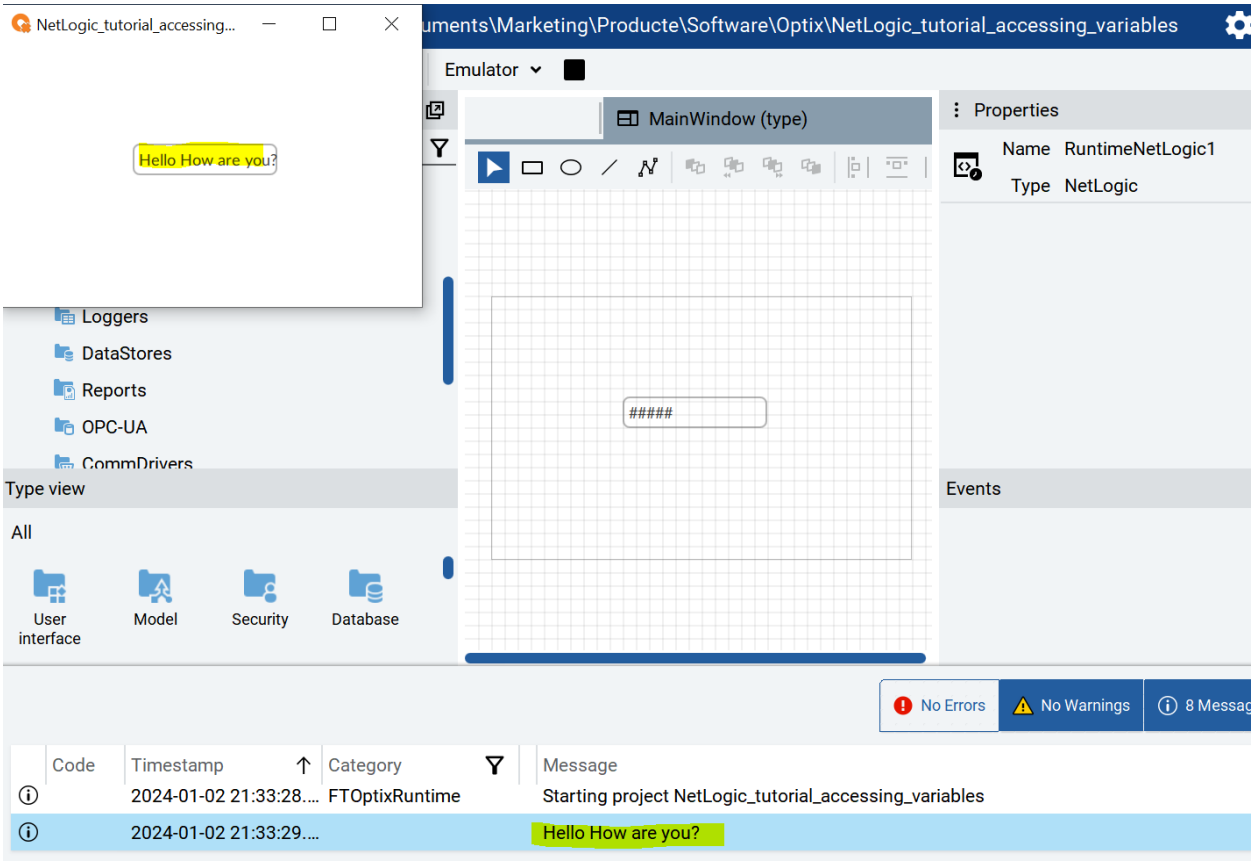
```
public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var missatge = Project.Current.GetVariable("Model/Message");
        missatge.Value="Hello How are you?";
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Save the code and test the emulator



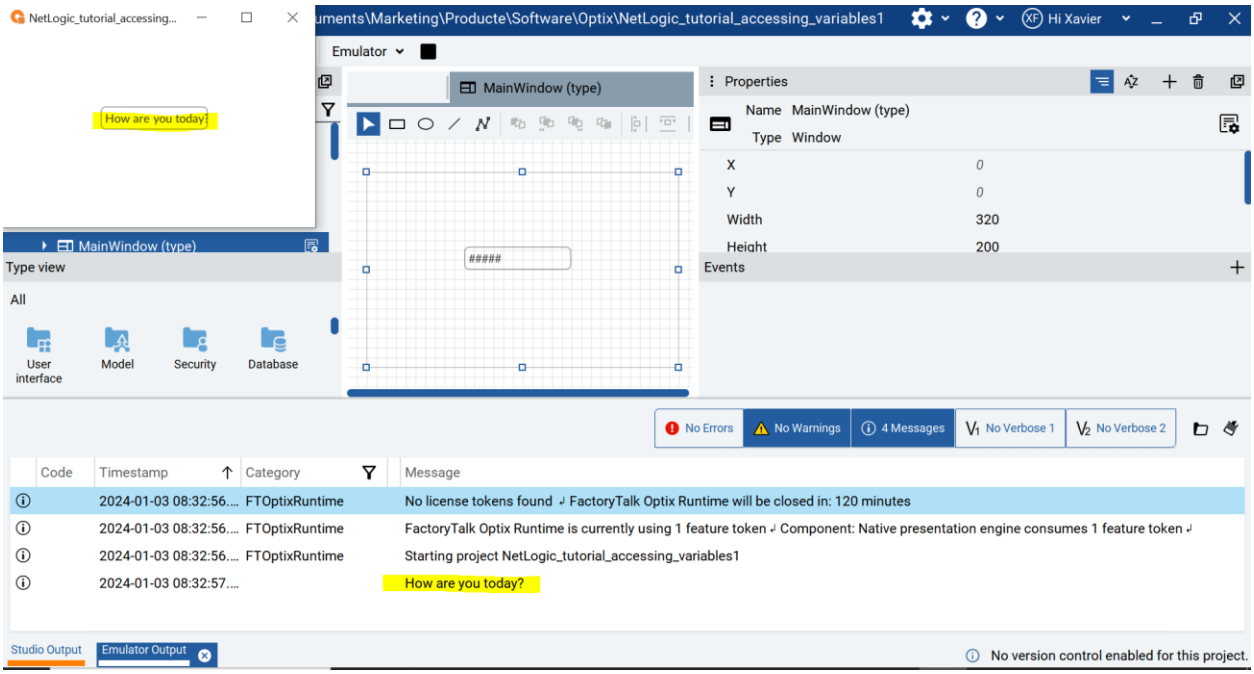
Now imagine you want to display the value on the Emulator Output. As seen on previous chapter Introduce this line on NetLogic

```
public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var missatge = Project.Current.GetVariable("Model/Message");
        missatge.Value="Hello How are you?";
        Log.Info(missatge.Value);
     }
```



You can even access the UI variable without using another variable in the code, this way

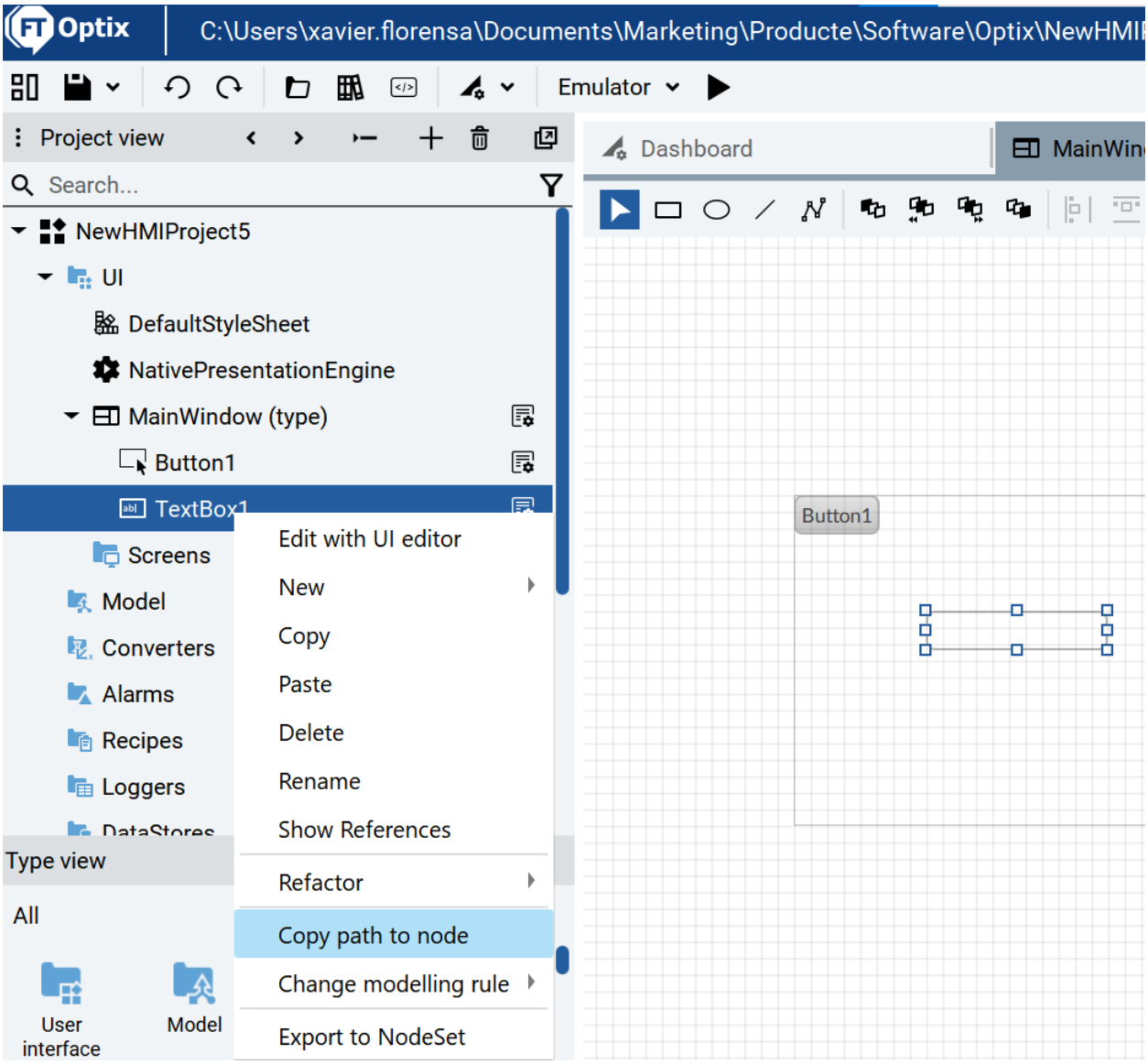
```
public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started

        Project.Current.GetVariable("Model/Message").Value = "How are you
today?";

        Log.Info(Project.Current.GetVariable("Model/Message").Value);
     }
```



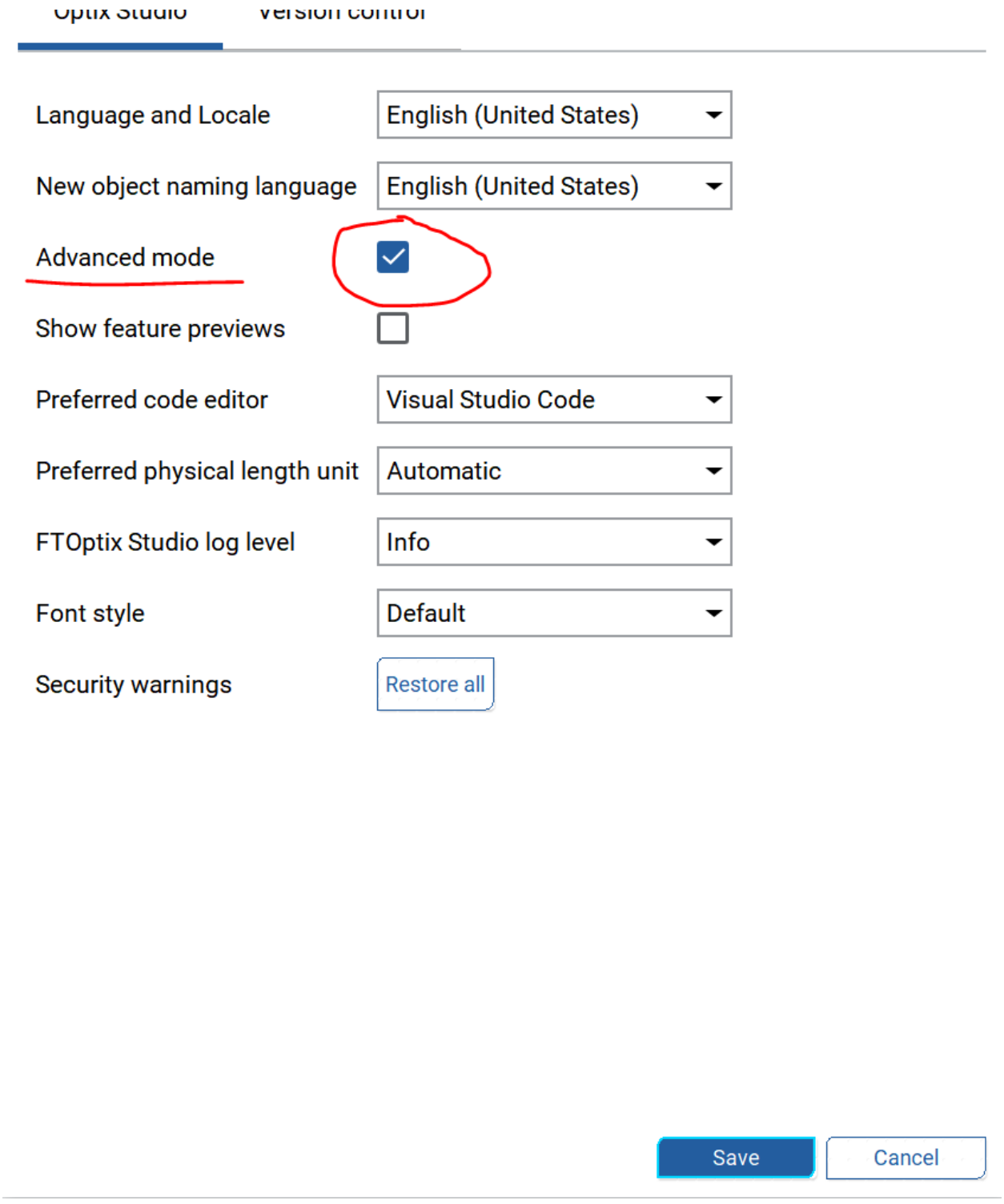
39. Accessing objects

You can have access to project objects like labels, texboxes, etc from your code.
As explained here
file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/netlogic/label-text/Develop-a-button-that-changes-label-text.html

The path to the NodeId of any object is
file:///C:/Program%20Files/Rockwell%20Automation/FactoryTalk%20Optix/Studio/Help/en/extending-projects/api/log/Log-Node.html?hl=path%2Cnodeid
To see the path to node you have to right click over the object and click on Copy Path to node



NewHMIProject5/UI/MainWindow/TextBox1
But you will not see this unless you select this checkbox on Optix configuration


Optix Studio
Version control
Language and Locale
English (United States)
New object naming language
English (United States)
Advanced mode
Show feature previews
Preferred code editor
Visual Studio Code
Preferred physical length unit
Automatic
FTOptix Studio log level
Info
Font style
Default
Security warnings
Restore all
Save
Cancel

Accessing project nodes

In order to manipulate a project element, you first need to access it, this can be either done by getting its NodeId or its node object

- Examples:
  - `var myNodeId = Project.Current.Get( [path/to/node] ).NodeId;`
  - `var myObject = Project.Current.Get( [path/to/node] );`
  - Where `path/to/node` can be obtained by right clicking any element of the IDE and then clicking `Copy path to node`, here you need to remember to remove the project name from the pasted element
    - Example:
      - `NewHMIProject25/Model/MyCustomMotor` --> `Model/MyCustomMotor`

The `Get` method also accepts an input type that will be passed to the variable type

- Example:
  - `var myButton = … .Get<Button>("path/to/button");` -> Will create a *myButton* variable of Type `FTOptix.UI.Button` pointing to the desired element
- After the object has been casted to the proper path, you can access its properties using IntelliSense
  - Example:
    - `myButton.Text = Hello;`

Log.Node(node, verbose)

Returns a string that contains the path of the node passed in the argument. The second argument is optional and inserts the `NodeID` and object type in the returned string.

```
static string Node(IUANode node, bool verbose);
```

Arguments

`node` (IUANode)

The node for which the path is to be generated in a string format.

`verbose` (bool)

`false` (default)

Does not insert additional information in the string returned.

`true`

Inserts the `NodeID` and object type in the string returned.

Returns

`string`

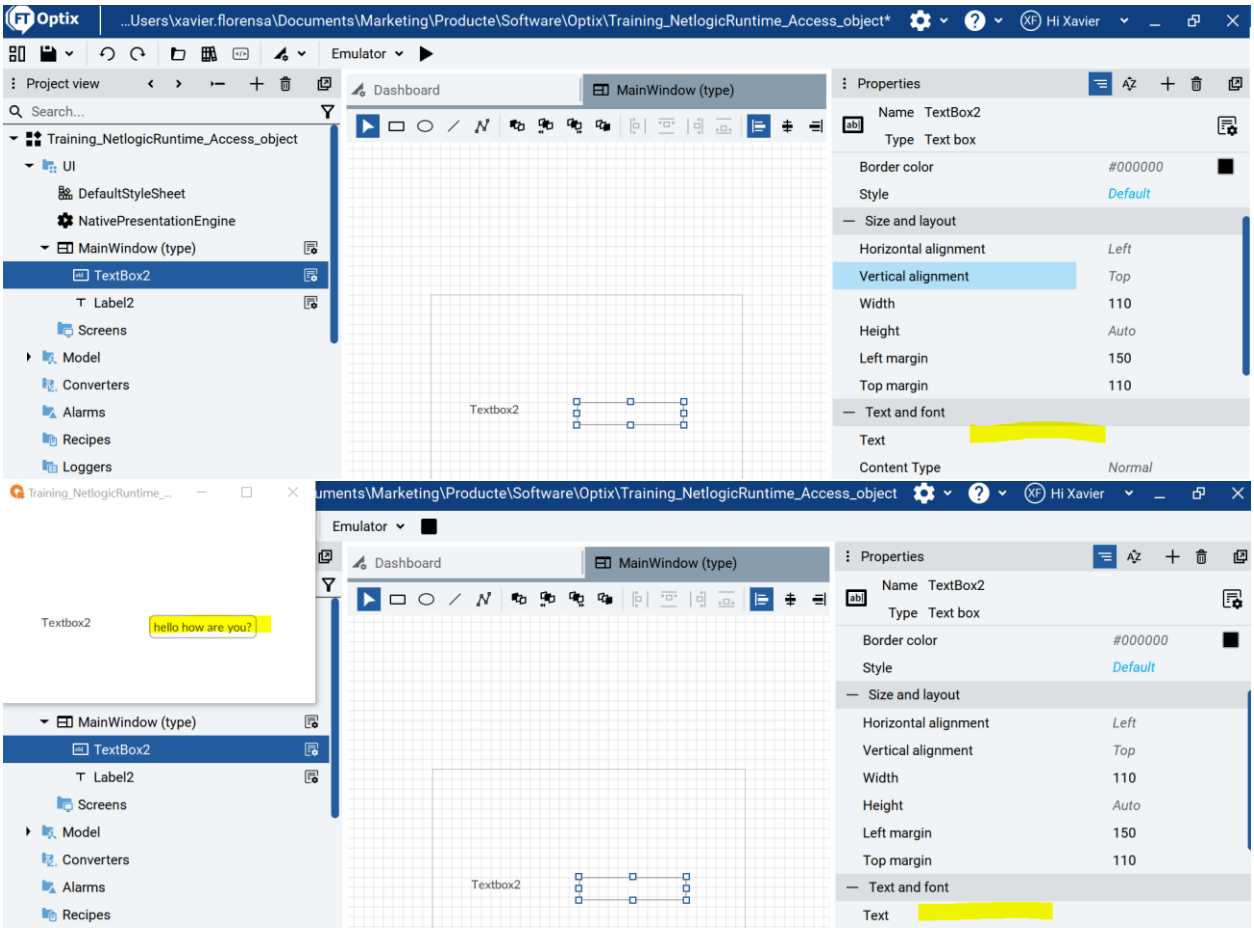
The path of the node passed as the argument. Based on the `verbose` argument value, it can also contain the `NodeId` and the object type.

Examples

The following example shows an API that generates a message consisting of the "Error on node " string and the *NetLogic path*, returned as a string by the `Log.Node` API:

```
Log.Info("Error on node " + Log.Node(LogicObject);
```

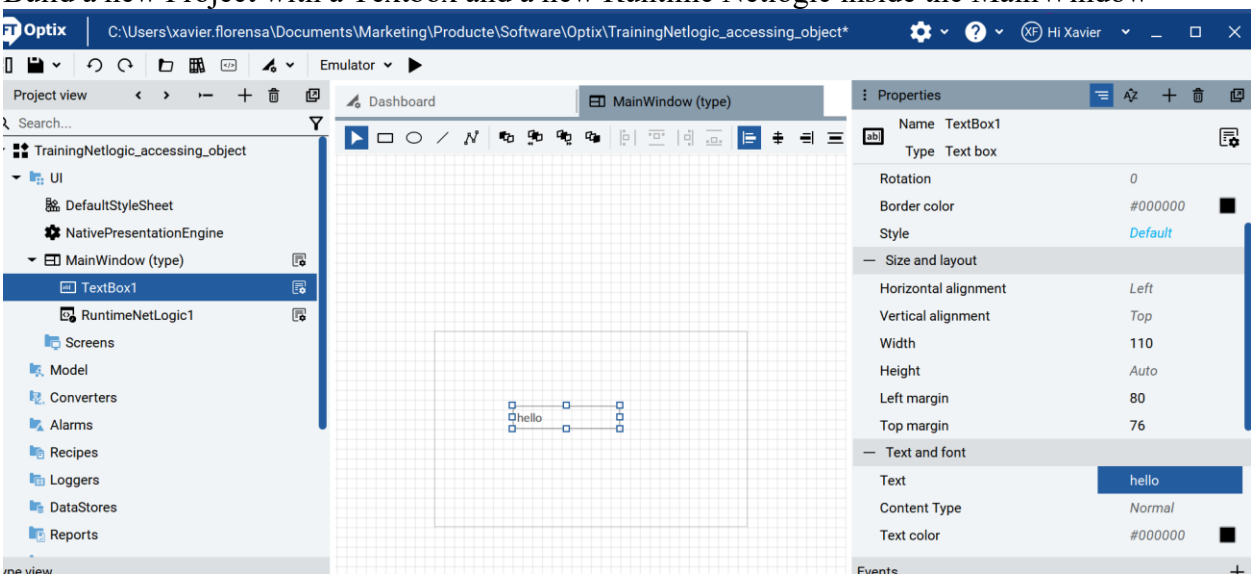
Let' define a textbox on UI




```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
      var myTextbox = Project.Current.Get<TextBox>("UI/MainWindow/TextBox2");
      myTextbox.Text="hello how are you?";
    }
```

```
    public override void Stop()
    {
    }
}
```

You can also access an object this way
Build a new Project with a Textbox and a new Runtime Netlogic inside the MainWindow

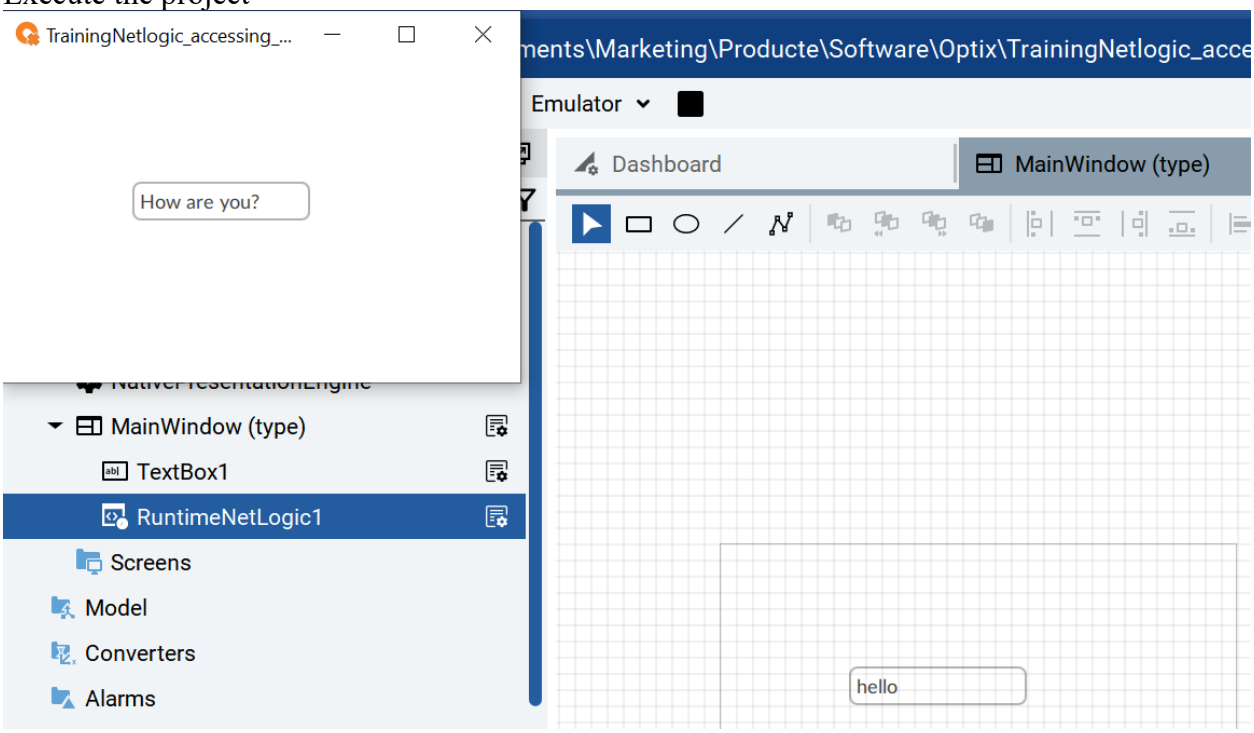


Open code and insert this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
         var textbox1 = Owner.Get<TextBox>("TextBox1");
         textbox1.Text="How are you?";
    }
```

```
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Execute the project



Detecting a change on an object

IUAObject.UAEvent

This event occurs when the project object to which the `IUAObject` C# object refers generates any OPC UA event.

```
event EventHandler<UAEventArgs> UAEvent;
```

Event handler

```
public delegate void UAEvent(object sender, UAEventArgs e);
```

Event handler arguments

`sender` **(object)**

A C# object that corresponds to the object of the project origin of the event.

`e` **(UAEventArgs)**

A C# object that contains the following properties:

`EventType` **(IUAObjectType)**

The node of the type of event generated.

`Arguments` **(UAEventArgumentList)**

A C# object that contains the arguments of the generated event.

Example

```
public override void Start()
{
   var button1 = Owner.Get<Button>("Button1");
   button1.UAEvent += Button1_UAEvent;
}

private void Button1_UAEvent(object sender, UAEventArgs e)
{
   var label1 = Owner.Get<Label>("Label1");
   var button1 = (Button)sender;
   label1.Text = "Event on " + button1.BrowseName + " of type " + e.EventType.BrowseName +
" , ";
}
```

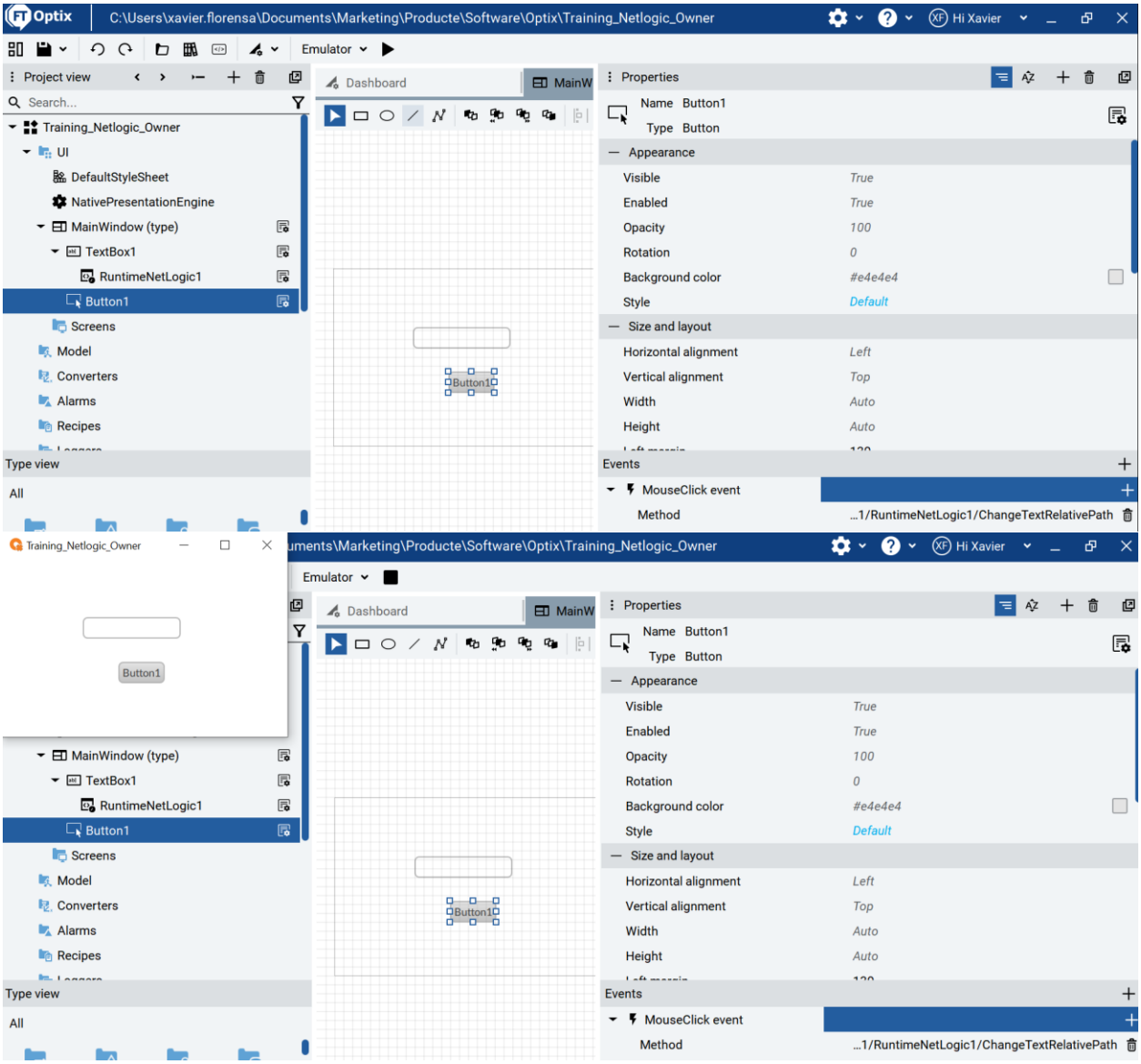
Create a project with a textbox

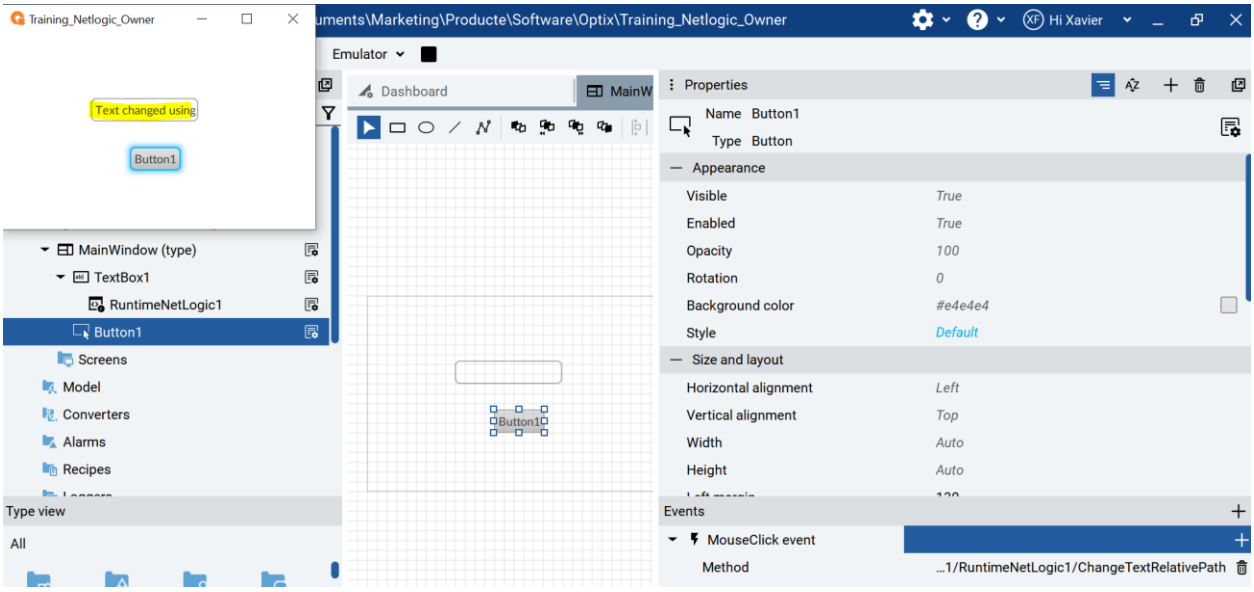
40. Using Owner

You can access object properties if you insert RuntimeNetlogic just inside an object

40.1. Accessing object using owner

You have to build the solution in order to find the routine


Optix
C:\Users\xavier.florensa\Documents\Marketing\Producte\Software\Optix\Training_Netlogic_Owner
Hi Xavier
Emulator
Project view
Search...
Training_Netlogic_Owner
UI
DefaultStyleSheet
NativePresentationEngine
MainWindow (type)
TextBox1
RuntimeNetLogic1
Button1
Screens
Model
Converters
Alarms
Recipes
Type view
All
Dashboard
MainW
Properties
Name Button1
Type Button
Appearance
Visible True
Enabled True
Opacity 100
Rotation 0
Background color #e4e4e4
Style Default
Size and layout
Horizontal alignment Left
Vertical alignment Top
Width Auto
Height Auto
Events
MouseClick event
Method ...1/RuntimeNetLogic1/ChangeTextRelativePath




```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

    [ExportMethod]
    public void ChangeTextRelativePath()
    {
     // WORKING AT RUNTIME!
    Owner.GetVariable("Text").Value = "Text changed using Owner";
```

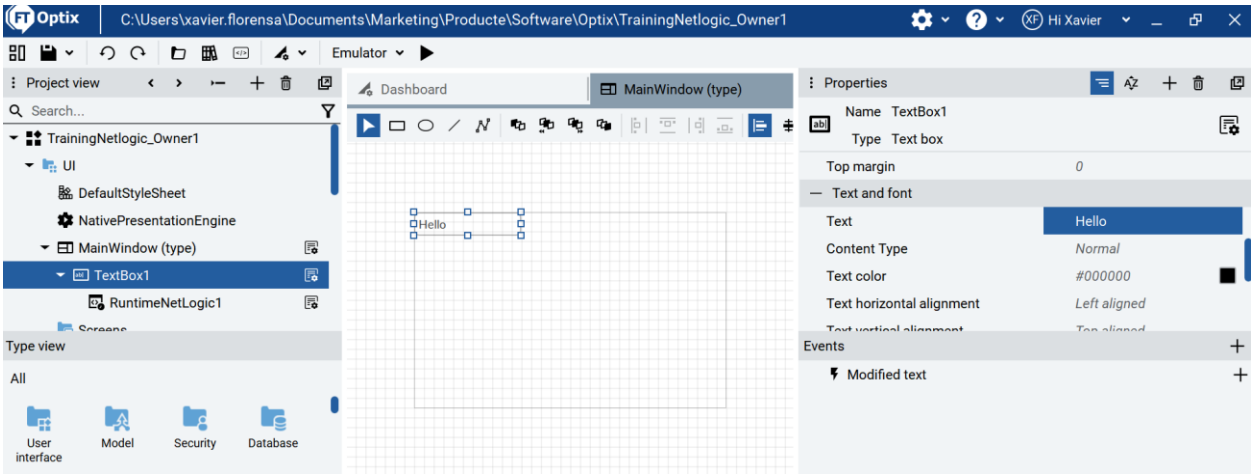
```
    }
}
```

You can also write on the Text field just after starting the program (without clicking on the button)

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    Owner.GetVariable("Text").Value = "Text changed using Owner";
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

    [ExportMethod]
    public void ChangeTextRelativePath()
    {
     // WORKING AT RUNTIME!
    //Owner.GetVariable("Text").Value = "Text changed using Owner";
    }
}
```

40.2. Detecting change on object using Owner

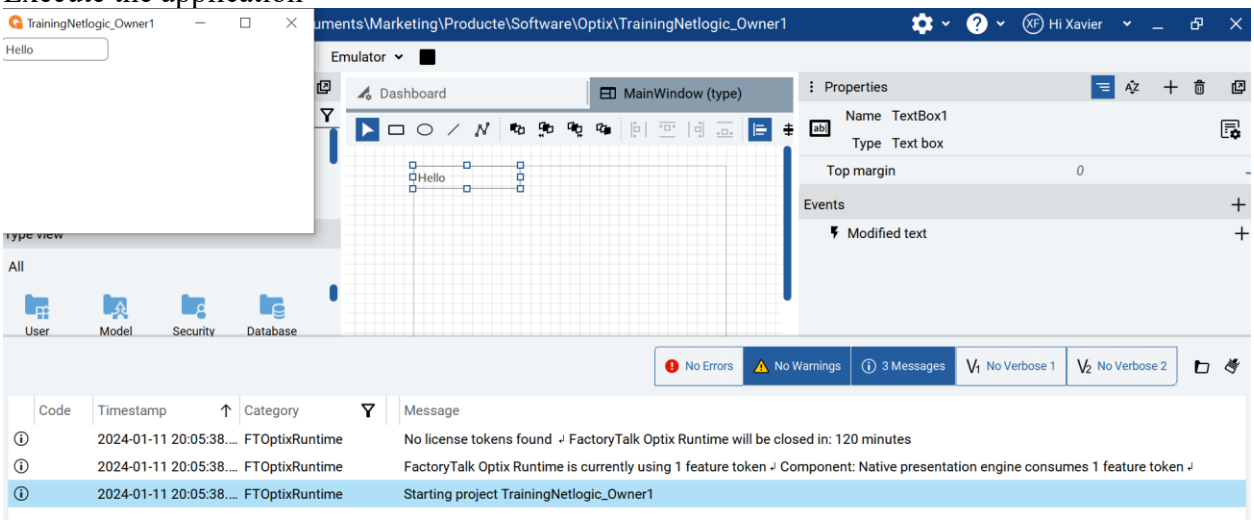
Create a Textbutton and insert a Runtime Netlogic on it



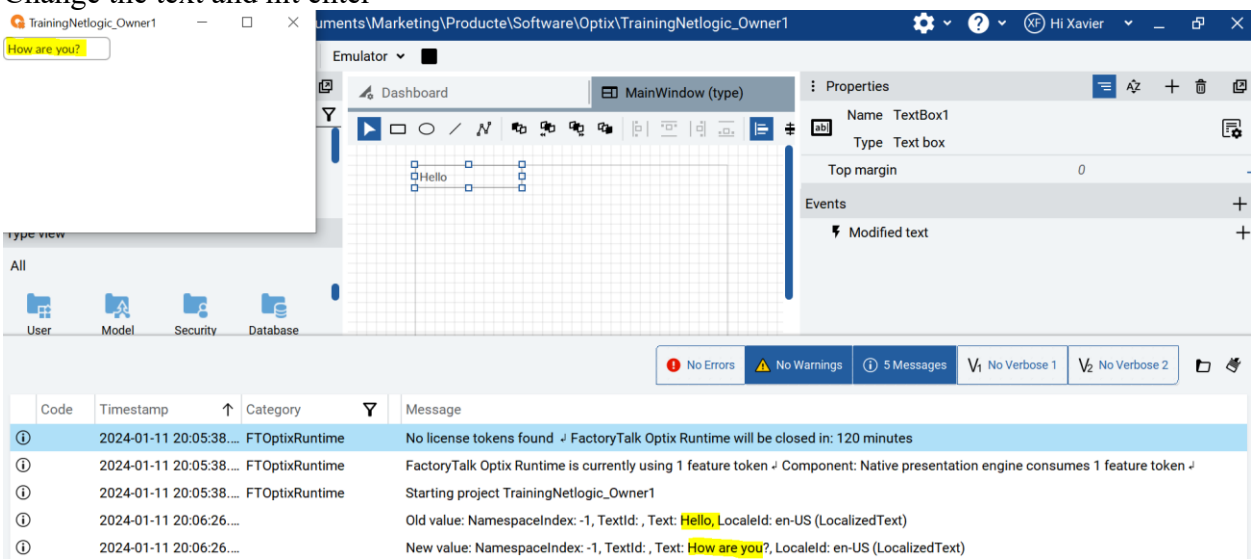

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    private IUAVariable myVar;
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        myVar = ((TextBox)Owner).TextVariable;
        myVar.VariableChange += MyVar_VariableChange;
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
     private void MyVar_VariableChange(object sender, VariableChangeEventArgs e)
    {
        // Log some information
        Log.Info("Old value: " + e.OldValue.ToString());
        Log.Info("New value: " + e.NewValue.ToString());
    }
```

```
}
```

Execute the application

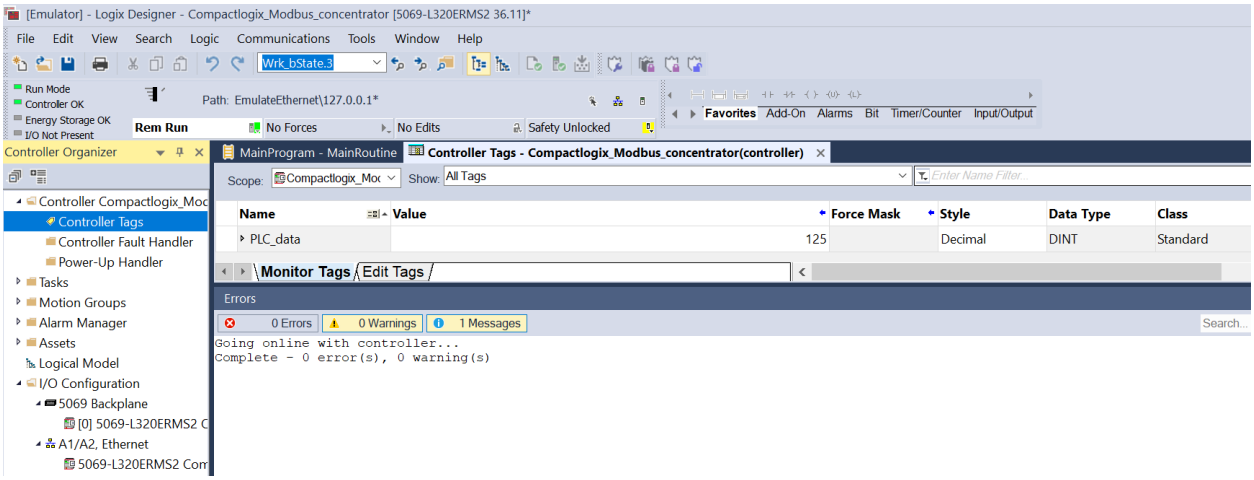


Change the text and hit enter

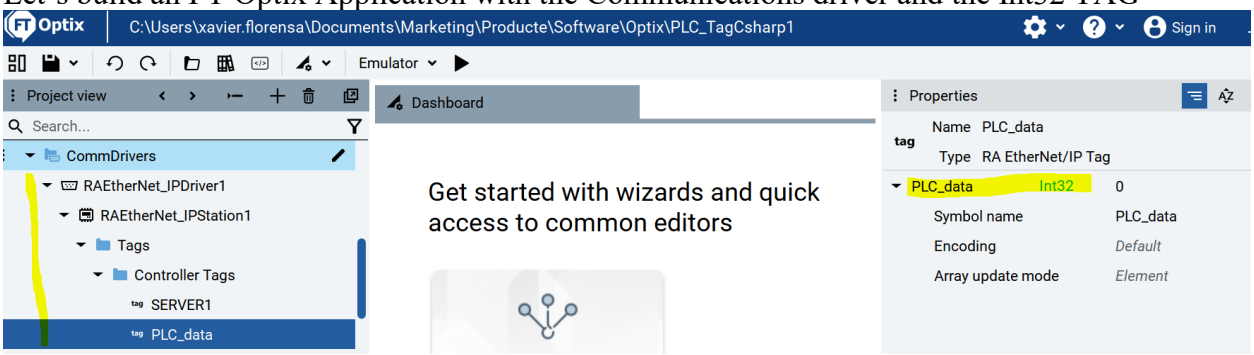


41. Accessing PLC Tags in NetLogic

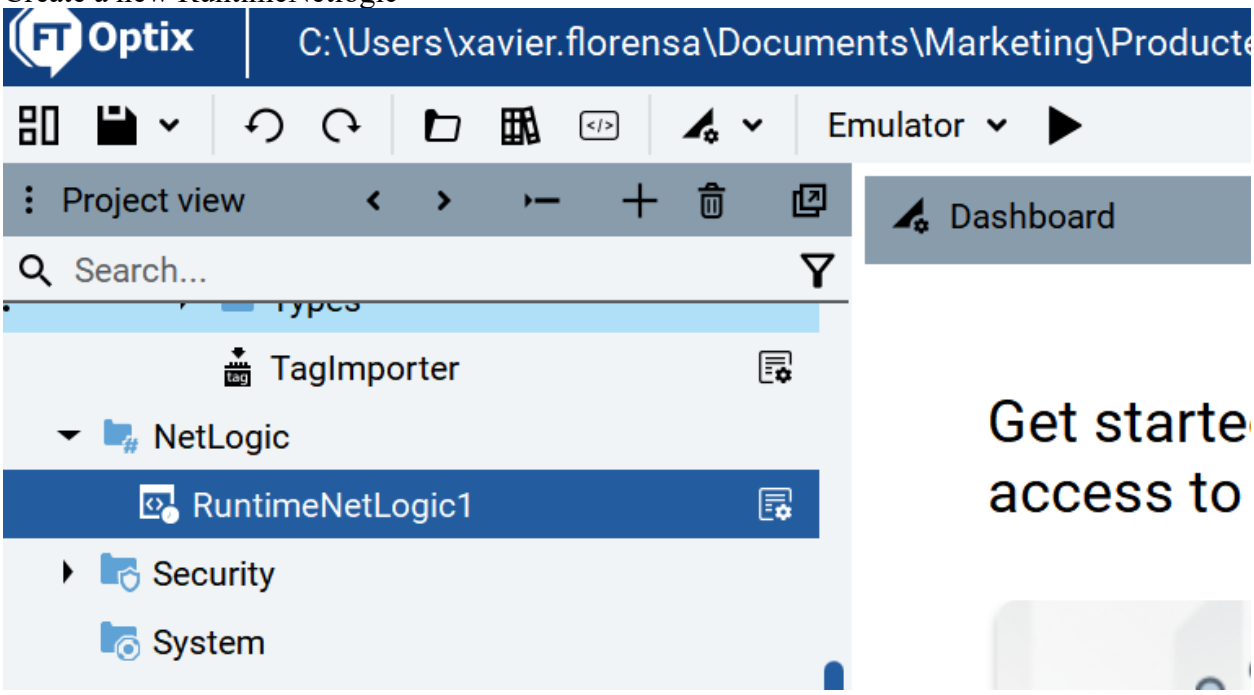
Let's build a PLC program with a DINT Tag, download and go online with our FactoryTalk Logix Echo emulator



Let's build an FT Optix Application with the Communications driver and the Int32 TAG

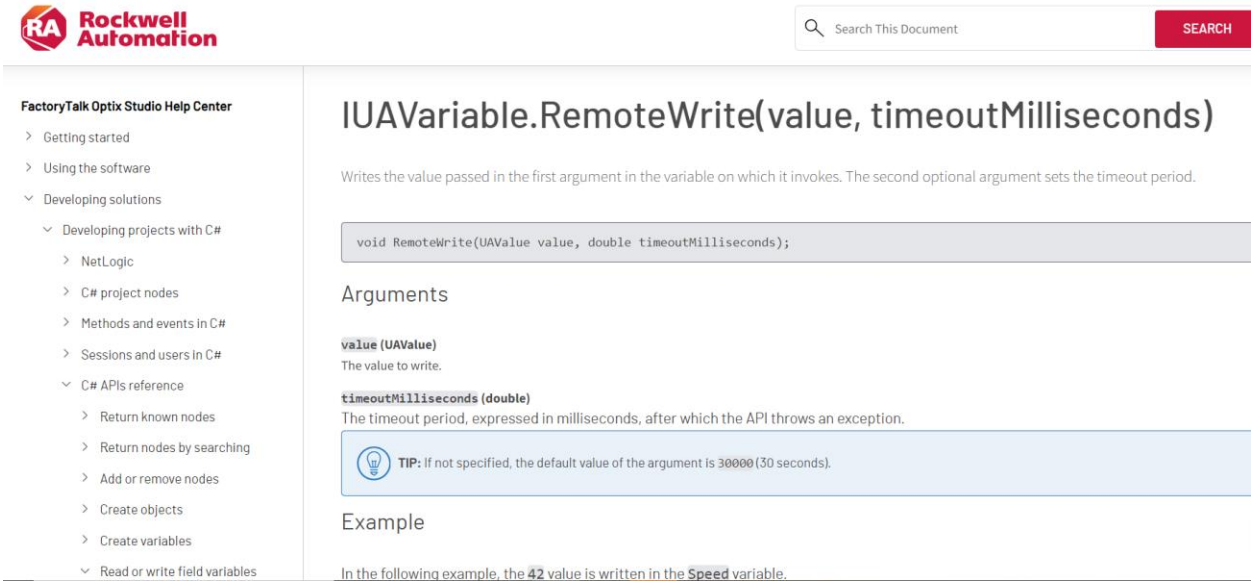


Create a new RuntimeNetlogic



41.1. Method0

This way
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/developing-projects-with-csharp/csharp-apis-reference/read-or-write-field-variables/iuavariable-remotewrite.html

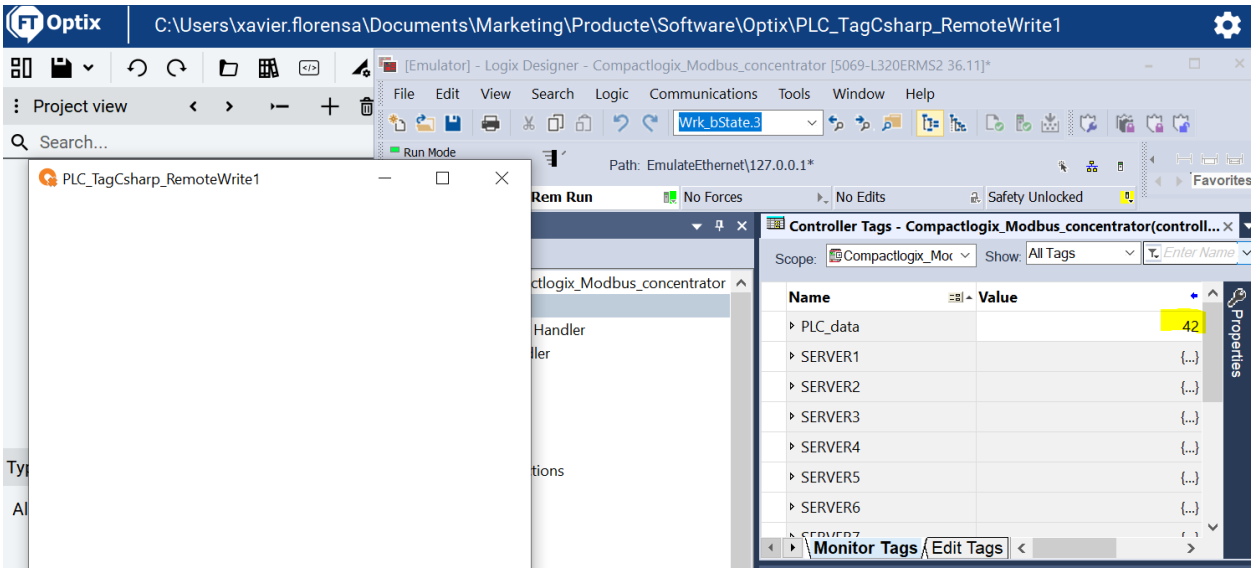


Use this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
using System.Collections.Generic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var currentSpeed =
Project.Current.GetVariable("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStatio
n1/Tags/Controller Tags/PLC_data");
        currentSpeed.RemoteWrite(42);
```

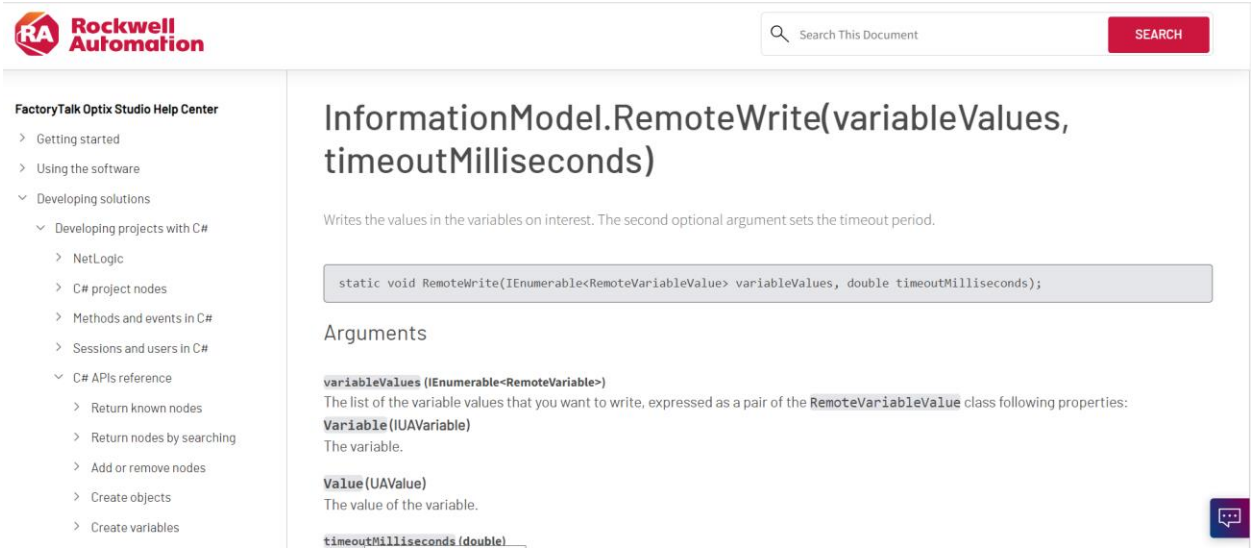
```
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Execute the application
Voilà



41.2. Method1

This way
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/developing-projects-with-csharp/csharp-apis-reference/read-or-write-field-variables/informationmodel-remotewrite.html



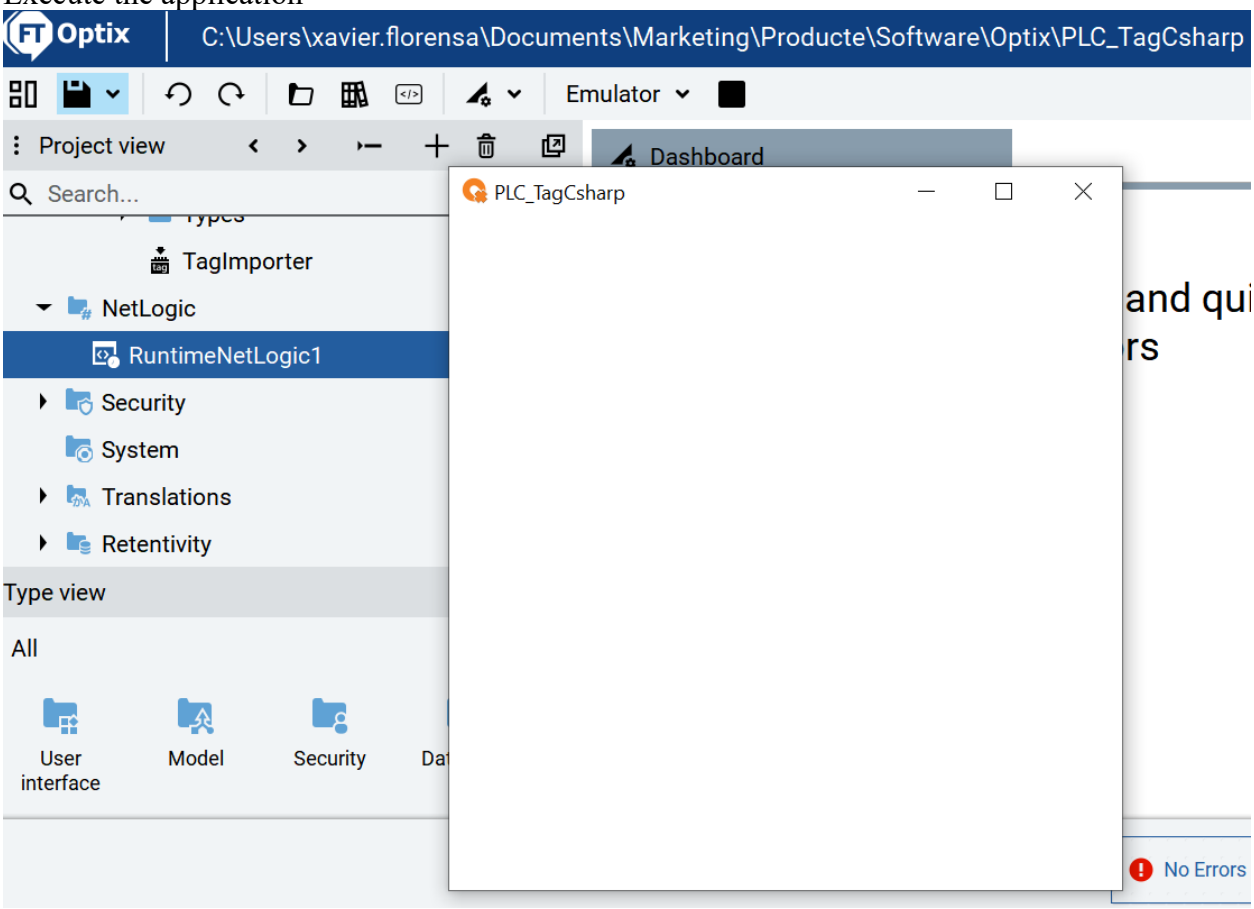
Use this code on the NetLogic
Do not forget to use the following directive

```
using System.Collections.Generic;
```

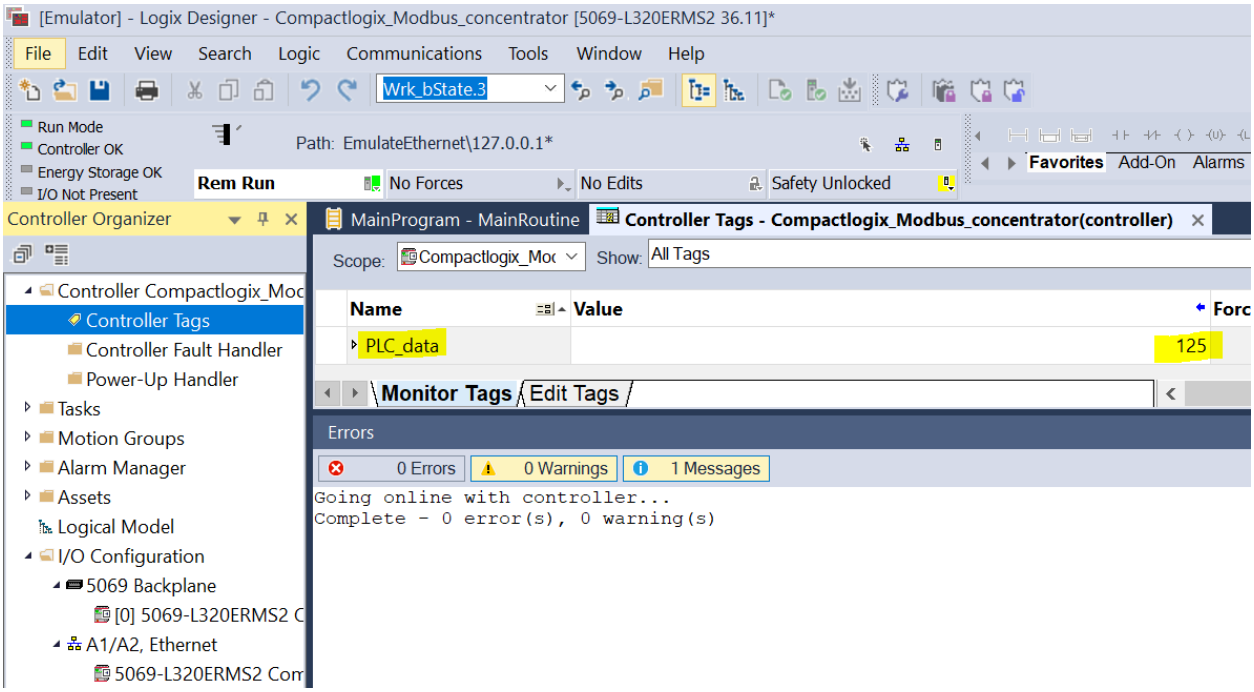
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
using System.Collections.Generic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started

        var tag1 =
Project.Current.Get<FTOptix.RAEtherNetIP.Tag>("CommDrivers/RAEtherNet_IPDriver1/R
AEtherNet_IPStation1/Tags/Controller Tags/PLC_data");
        var remoteVariableValues = new List<RemoteVariableValue>()
        {
```

```
        new RemoteVariableValue(tag1, 125)
        };
        InformationModel.RemoteWrite(remoteVariableValues);
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

}
```

Execute the application

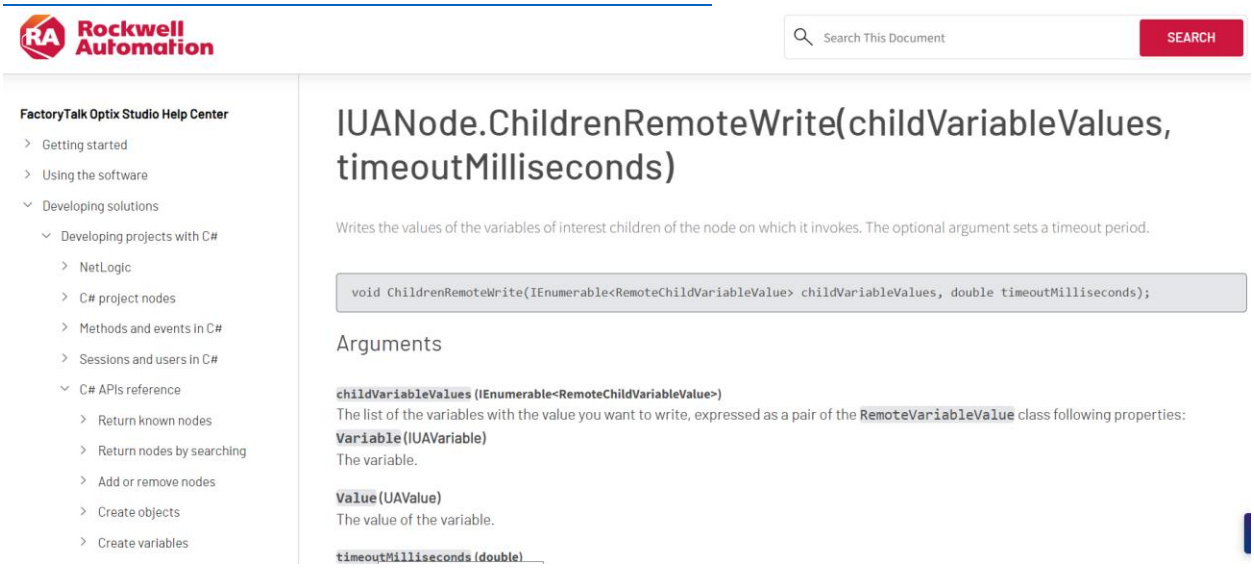


And voilà



41.3. Method 2

This way
https://www.rockwellautomation.com/en-us/docs/factorytalk-optix/1-00/contents-ditamap/developing-solutions/developing-projects-with-csharp/csharp-apis-reference/read-or-write-field-variables/iuanode-childrenremotewrite.html



Use this code on the NetLogic
Do not forget to use the following directive

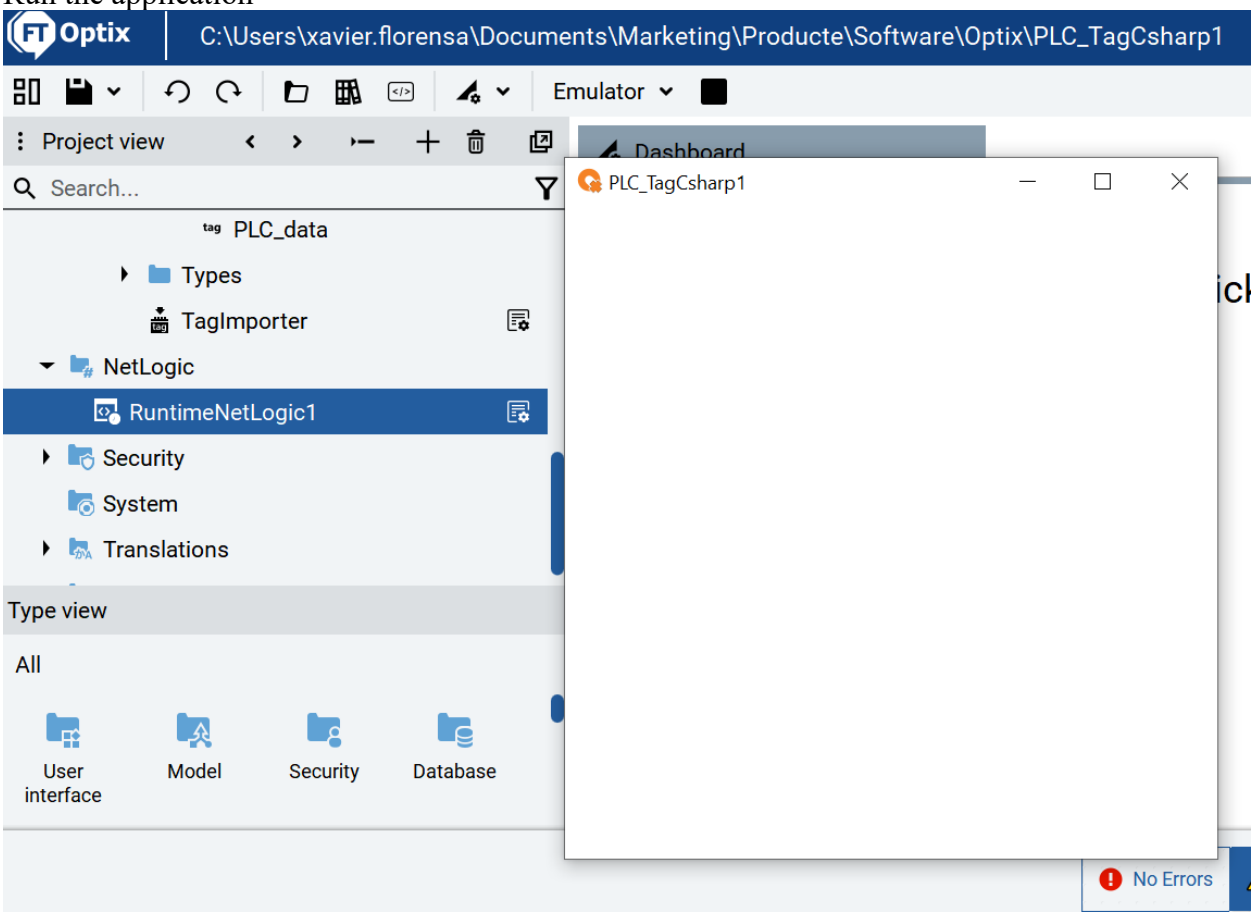
```
using System.Collections.Generic;
```

This is the code

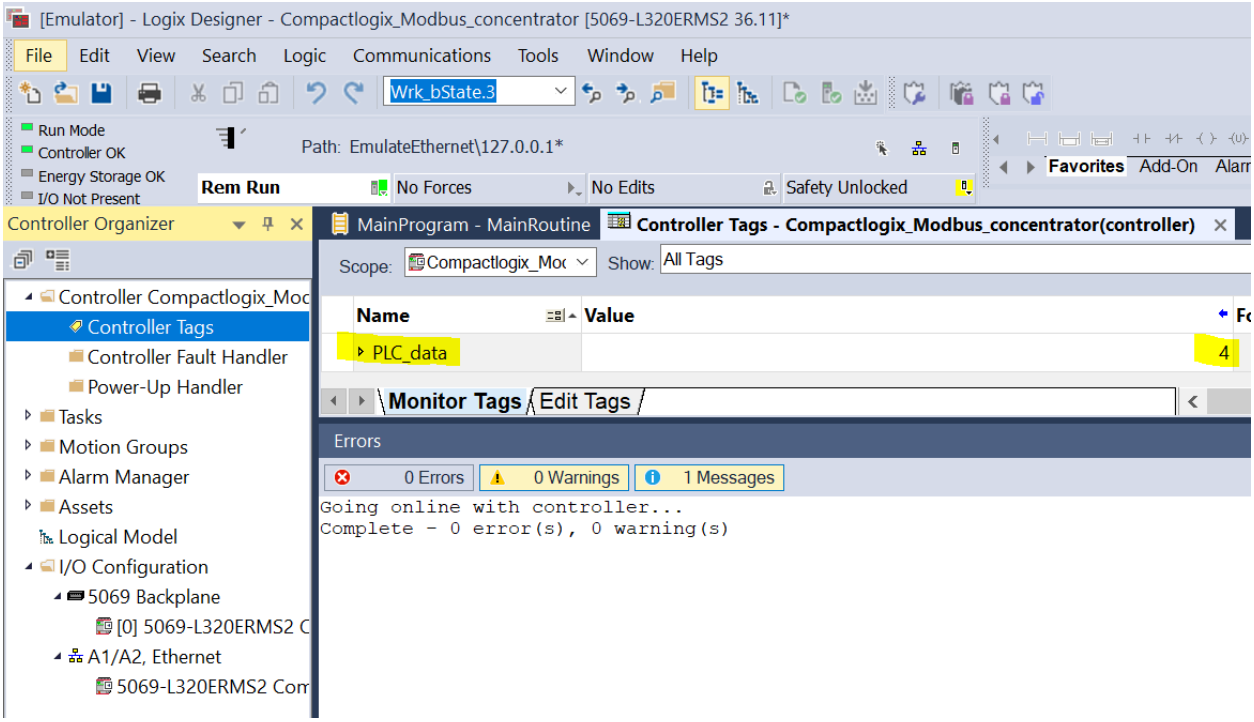
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
using System.Collections.Generic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var valuesToWrite = new List<RemoteChildVariableValue>()
        {
        new RemoteChildVariableValue("PLC_data", 4),


        };
        var myNode =
Project.Current.Get("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStation1/Tags/
Controller Tags");
        try
        {
            myNode.ChildrenRemoteWrite(valuesToWrite);
        }
        catch (Exception ex)
        {
            Log.Error("ChildrenRemoteWrite failed: " + ex.ToString());
        }



    }

    public override void Stop()
```

```
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Run the application



And voilà



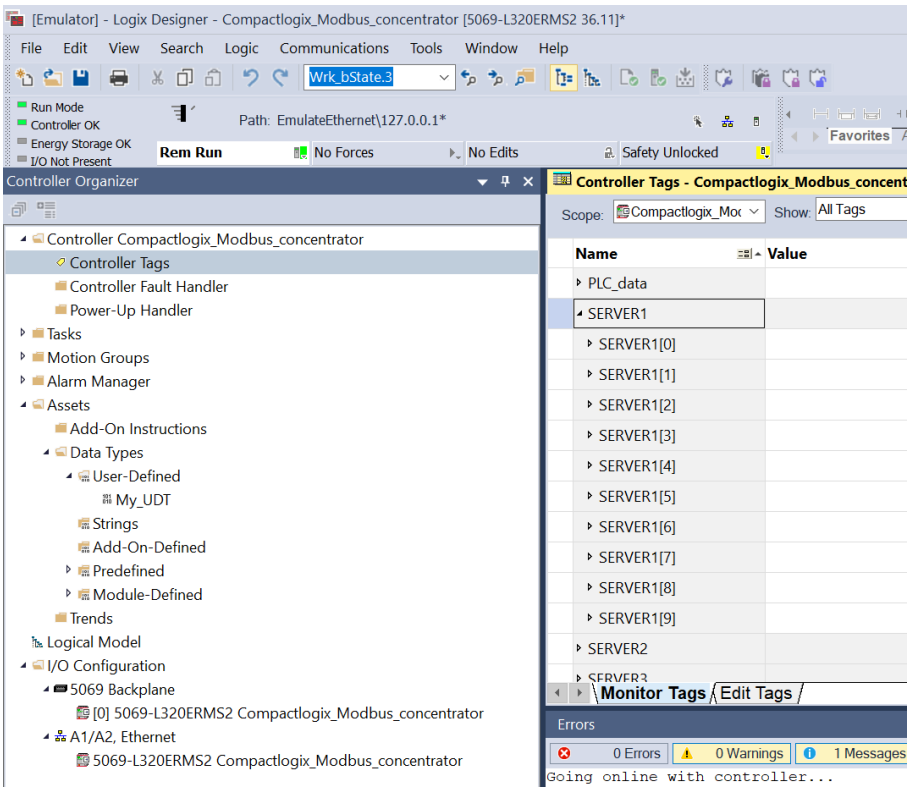
41.4. Writing to an array of PLC values

We will use this example, adapted to our needs.
https://github.com/FactoryTalk-Optix/NetLogic_CheatSheet/blob/main/pages/variables-interaction.md

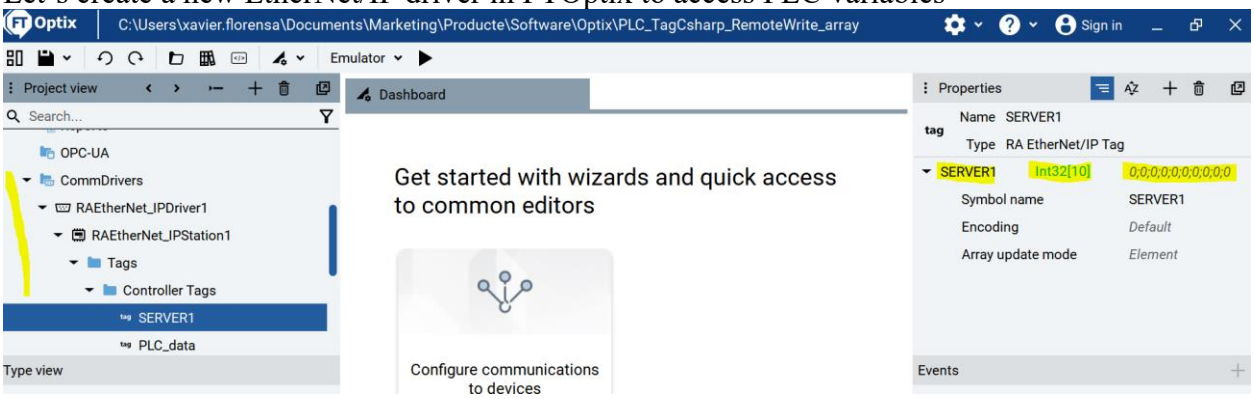
Access mono-dimensional arrays

```
// Get the model variable
var MyVar = Project.Current.GetVariable("Model/Variable");
// Create a C# variable to manipulate values
var tempVar = (float[])MyVar.Value.Value;
// Set the elements of the C# variable
tempVar[0] = (float)Project.Current.GetVariable("CommDrivers/EthernetIPDriver1/EthernetIPStation1/TagName").Value;
// Write the new value(s) to the model variable
MyVar.SetValue(tempVar);
```

Let’s Build a PLC program with an array of integers



Let’s create a new EtherNet/IP driver in FTOptix to access PLC variables



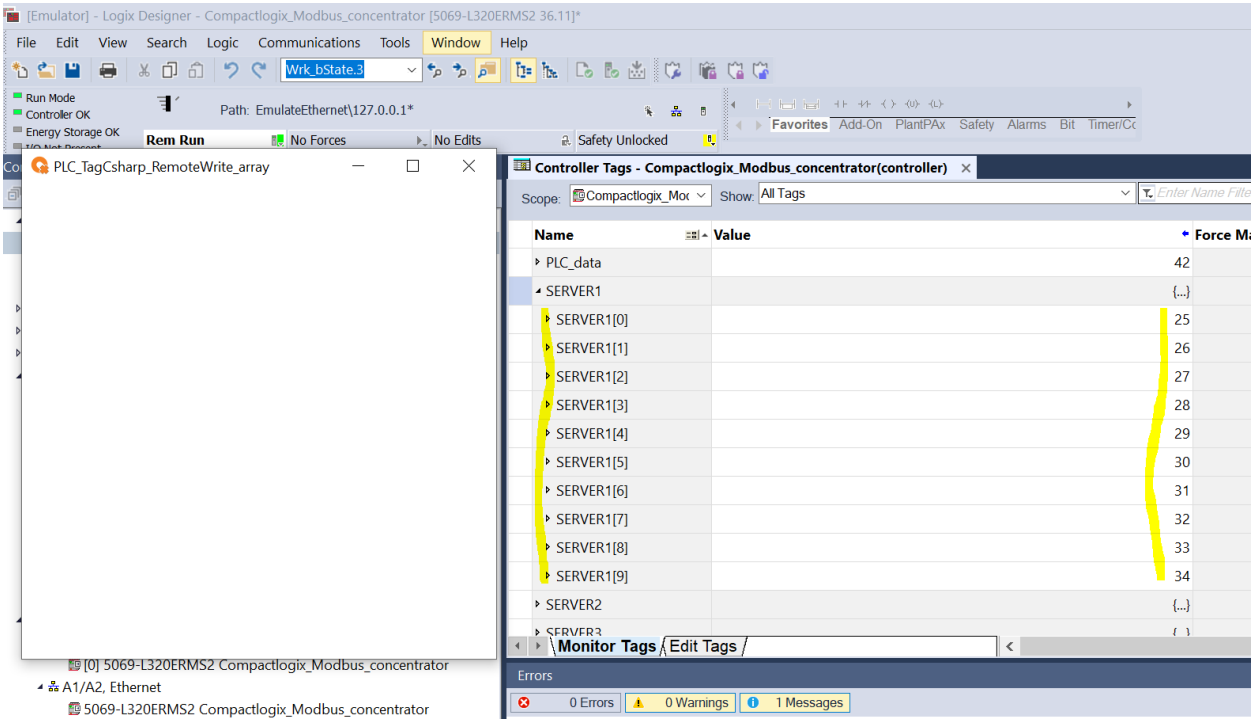
Now create a Runtime NetLogicl

And et’s use this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
```

```
using FTOptix.RAEtherNetIP;
using FTOptix.CommunicationDriver;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        Log.Info("Starting application Xavier");
        var currentSpeed =
Project.Current.GetVariable("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStatio
n1/Tags/Controller Tags/PLC_data");
        var SERVER1 =
Project.Current.GetVariable("CommDrivers/RAEtherNet_IPDriver1/RAEtherNet_IPStatio
n1/Tags/Controller Tags/SERVER1");

        var arraySERVER1 = (Int32[])SERVER1.Value.Value;
        arraySERVER1[0]=25;
        arraySERVER1[1]=26;
        arraySERVER1[2]=27;
        arraySERVER1[3]=28;
        arraySERVER1[4]=29;
        arraySERVER1[5]=30;
        arraySERVER1[6]=31;
        arraySERVER1[7]=32;
        arraySERVER1[8]=33;
        arraySERVER1[9]=34;
        SERVER1.SetValue(arraySERVER1);

        currentSpeed.RemoteWrite(42);

        Log.Info("SERVER1 INDEX 0 VALUE" + arraySERVER1[0]);
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

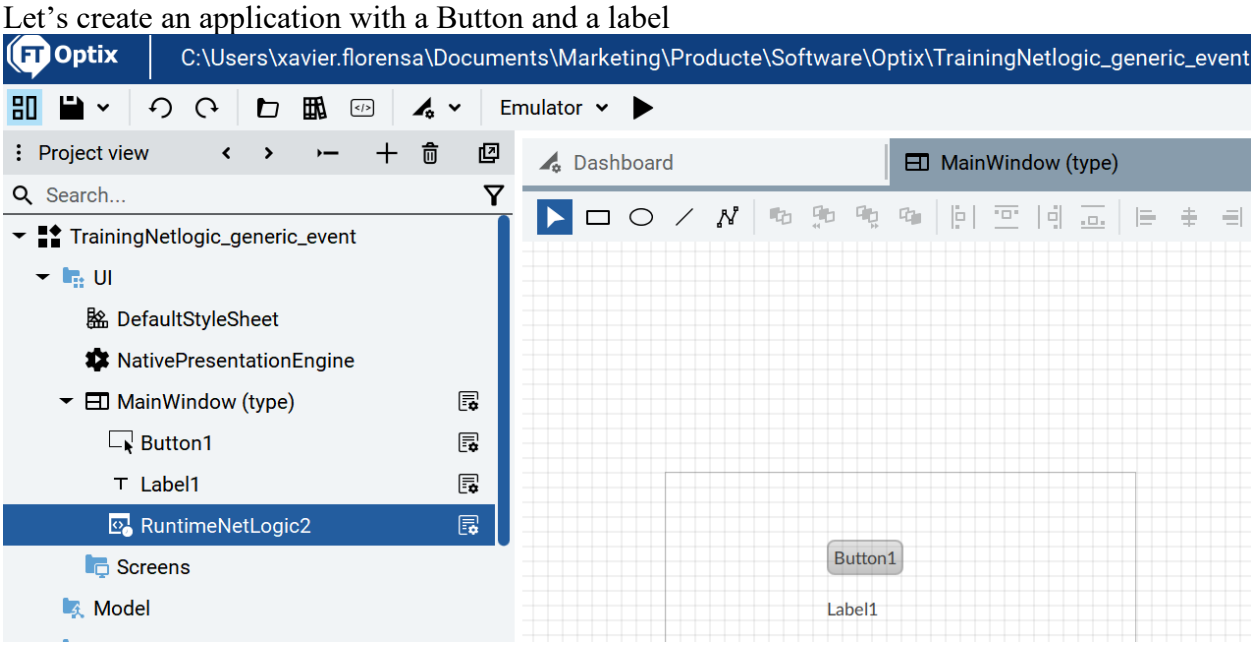
Run the application and …
voilà



42. Object events

You can also detect a change this way

Let's create an application with a Button and a label

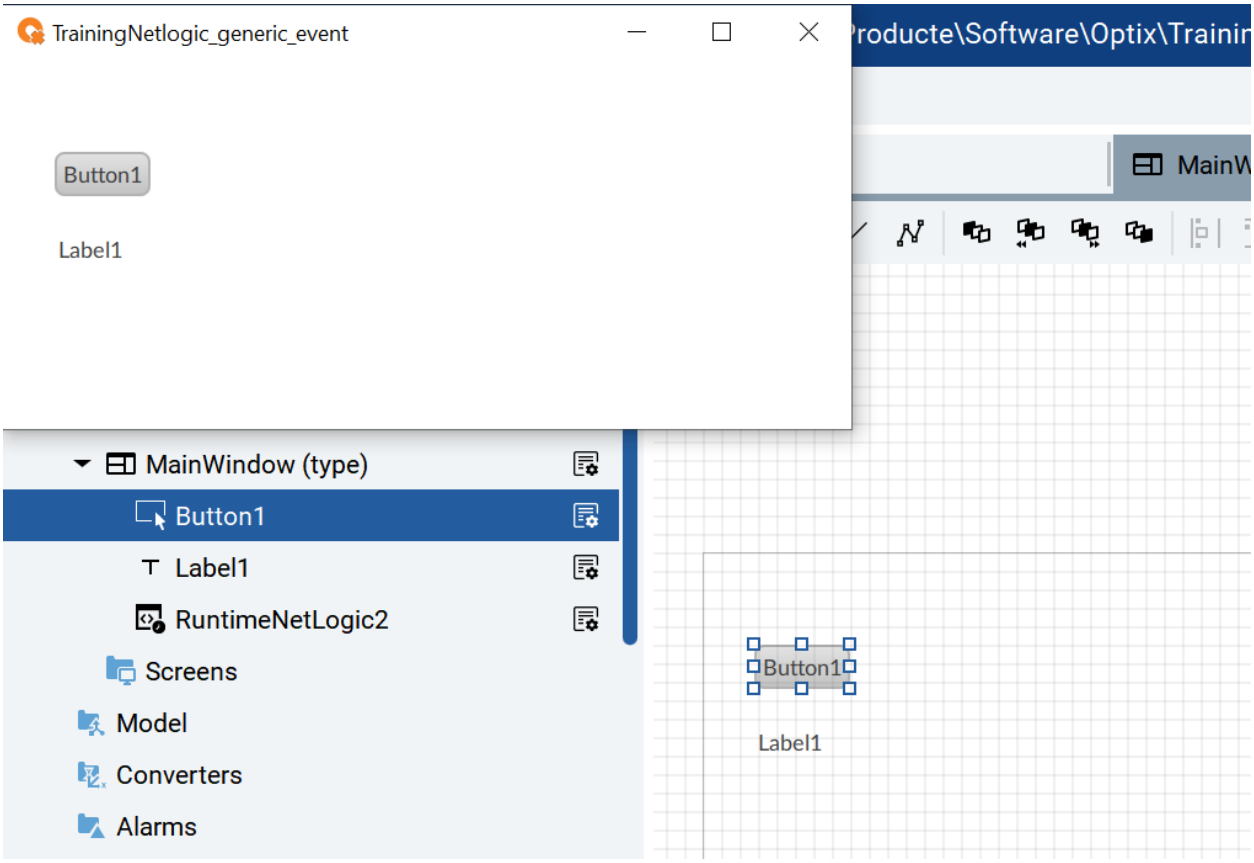


Create a new RuntimeNetLogic under MainWindow

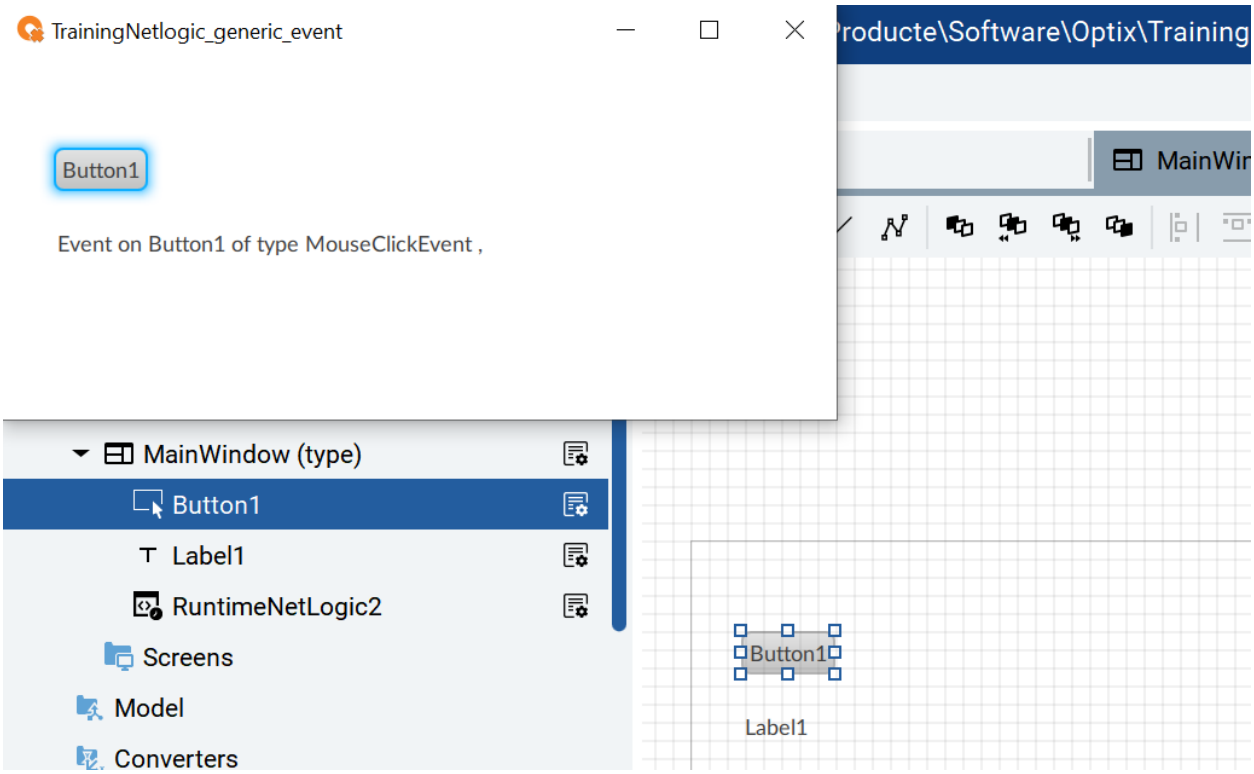
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
```

```
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic2 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    var button1 = Owner.Get<Button>("Button1");
    button1.UAEvent += Button1_UAEvent;
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
     private void Button1_UAEvent(object sender, UAEventArgs e)
    {
        var label1 = Owner.Get<Label>("Label1");
        var button1 = (Button)sender;
        label1.Text = "Event on " + button1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
    }
}
```

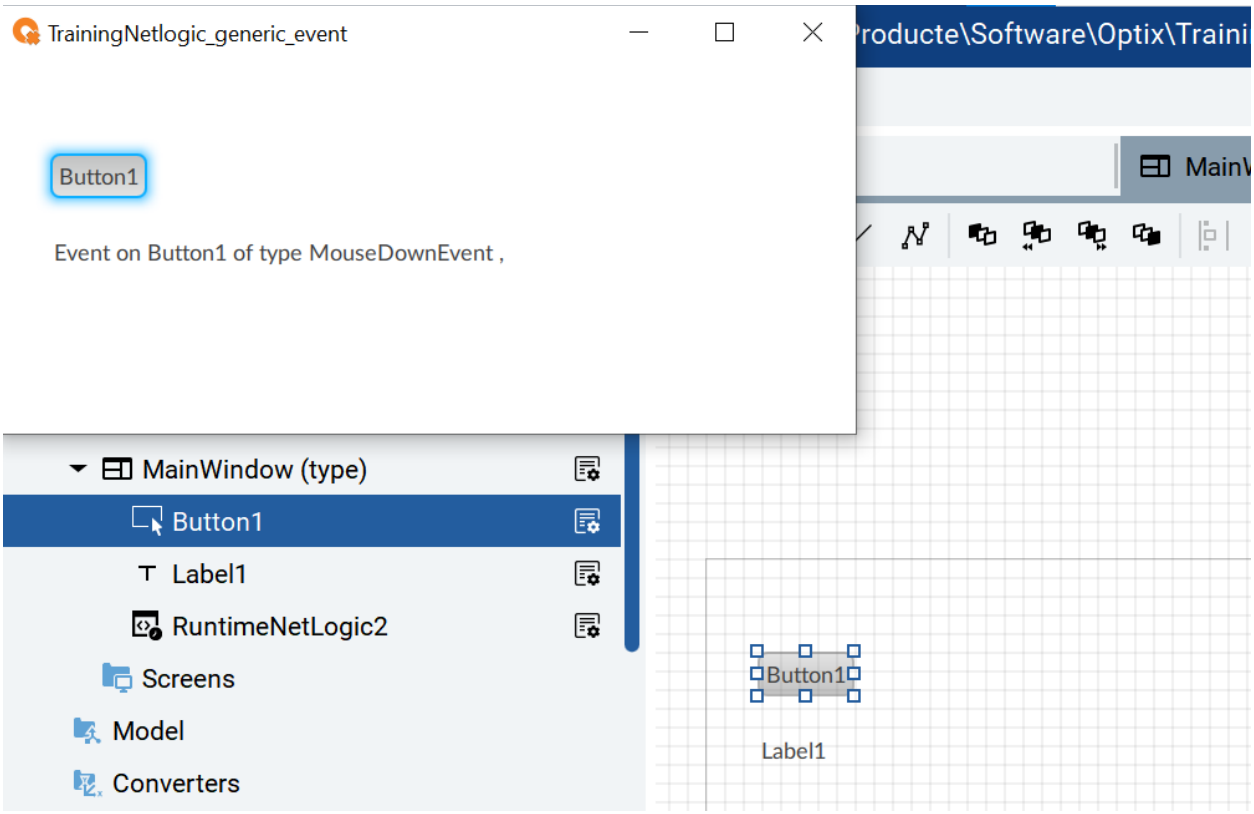
Execute the application
Click the button



Click on the button

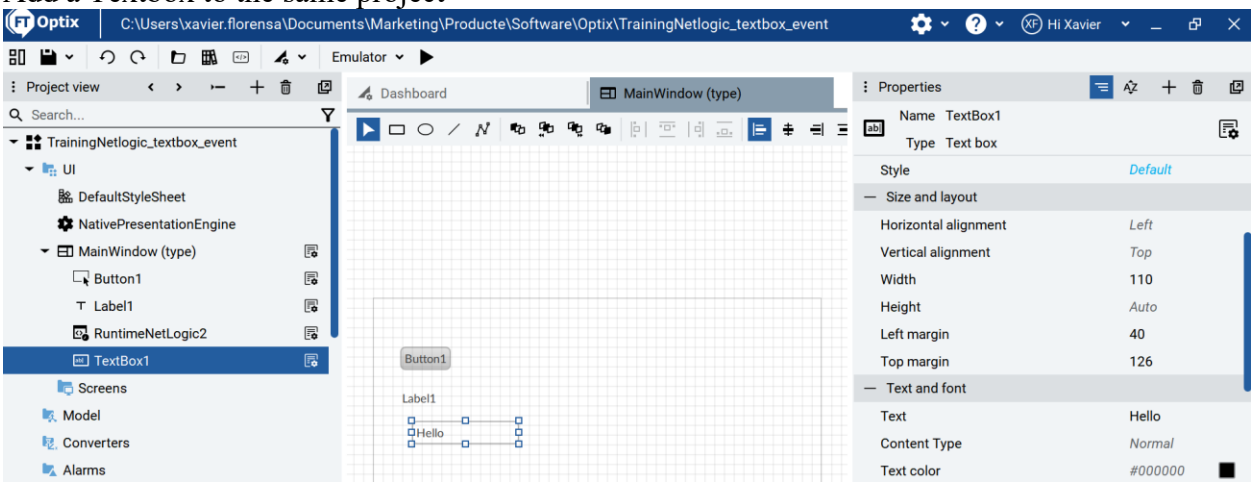


Hold pressing the button



Now let's try to detect a change in a Textbox

Add a Textbox to the same project

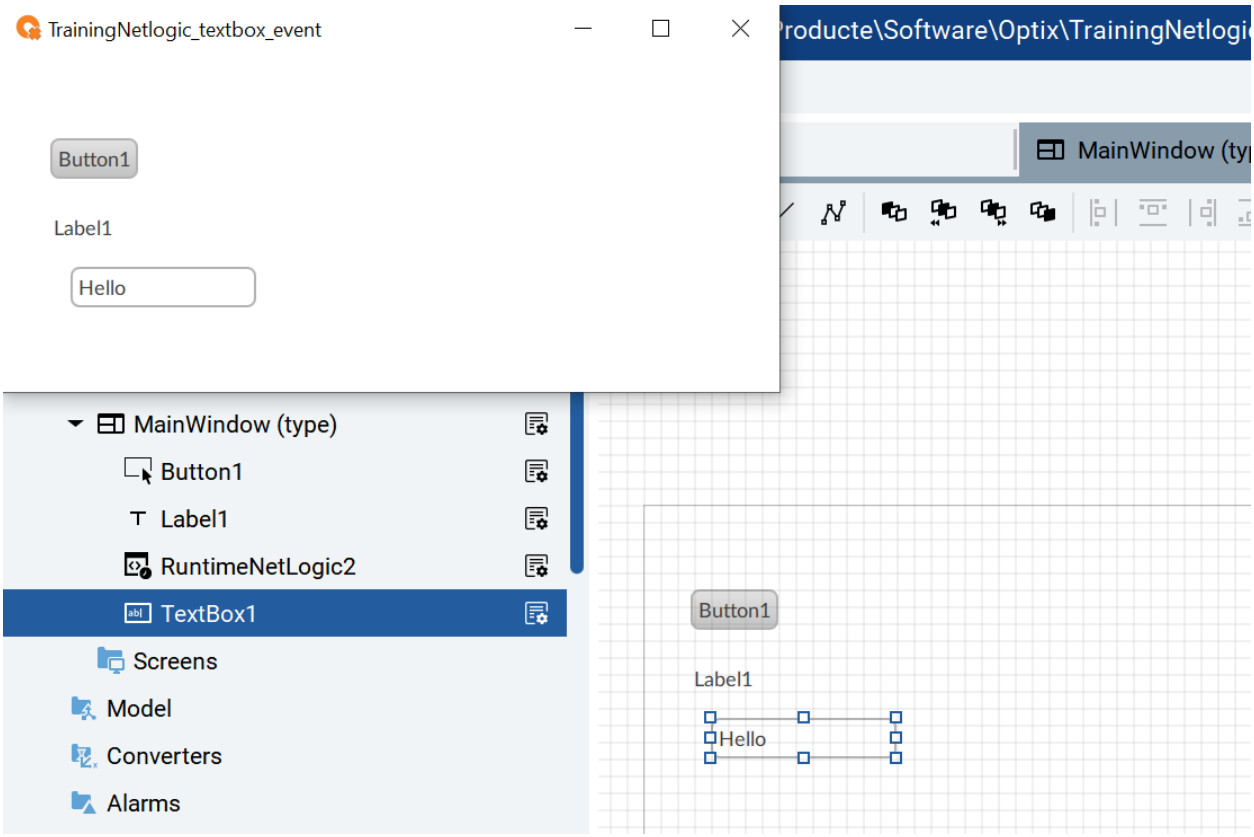


And use this code

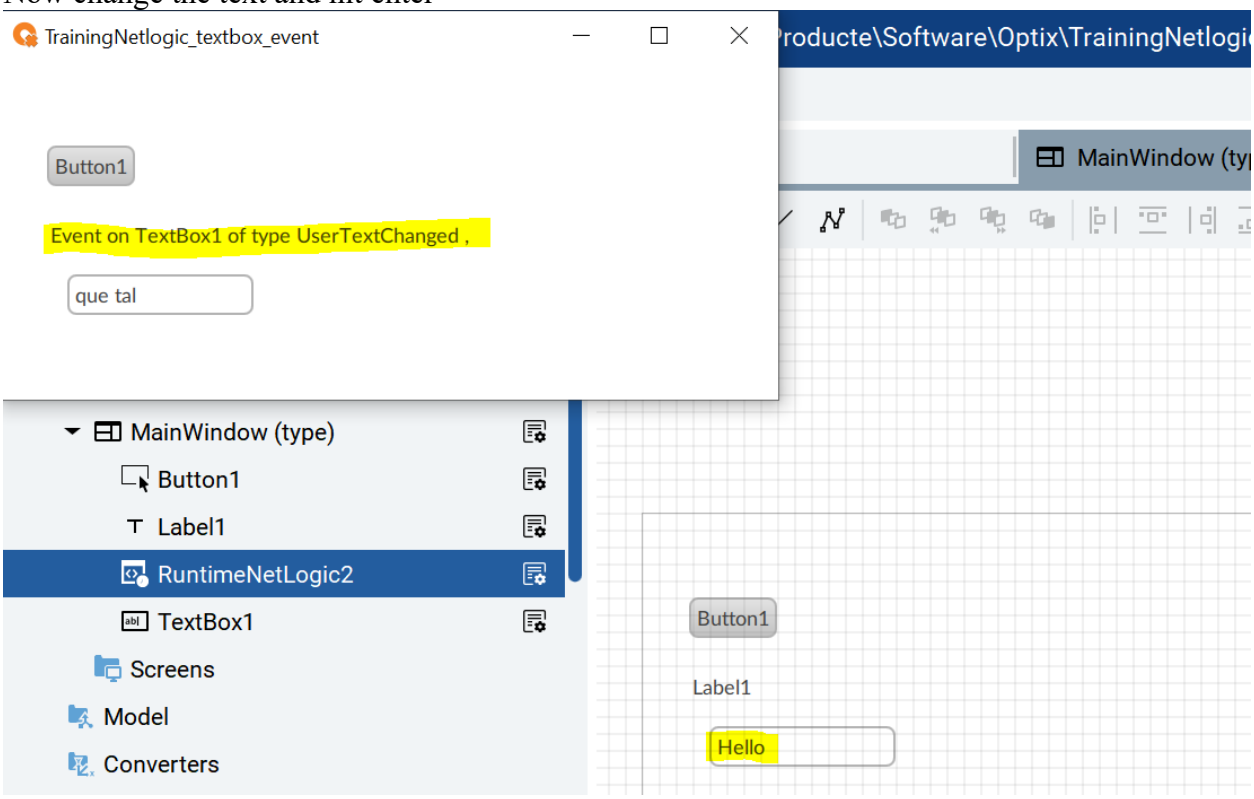
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
```

```
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic2 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    var button1 = Owner.Get<Button>("Button1");
    button1.UAEvent += Button1_UAEvent;
    var textbox1 = Owner.Get<TextBox>("TextBox1");
    textbox1.UAEvent += TextBox1_UAEvent;

    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
     private void Button1_UAEvent(object sender, UAEventArgs e)
    {
        var label1 = Owner.Get<Label>("Label1");
        var button1 = (Button)sender;
        label1.Text = "Event on " + button1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
    }
     private void TextBox1_UAEvent(object sender, UAEventArgs e)
    {
        var label1 = Owner.Get<Label>("Label1");
        var textbox1 = (TextBox)sender;
        label1.Text = "Event on " + textbox1.BrowseName + " of type " +
e.EventType.BrowseName + " , ";
    }

}
```

Run the project

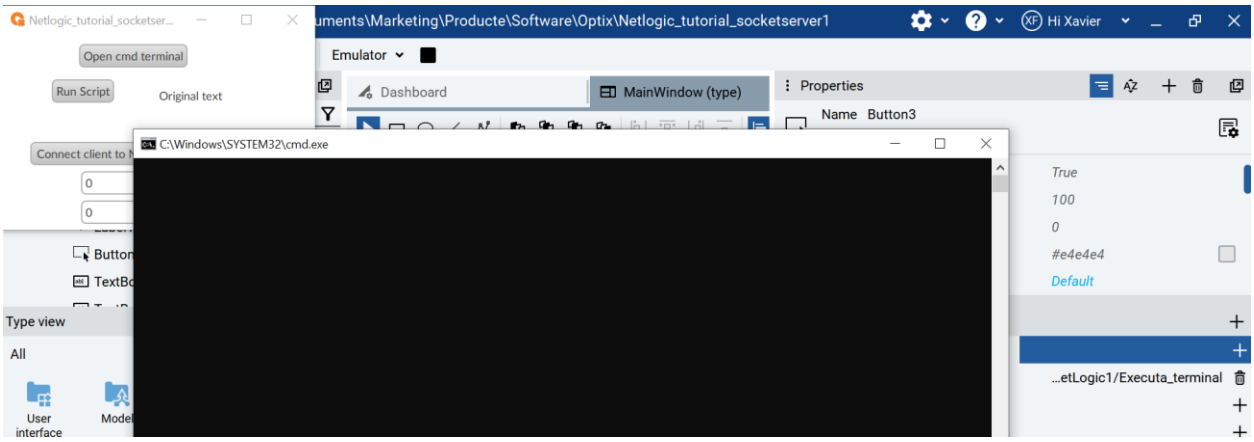


Now change the text and hit enter



43. Open a cmd terminal to run a command or program

After the click of a button

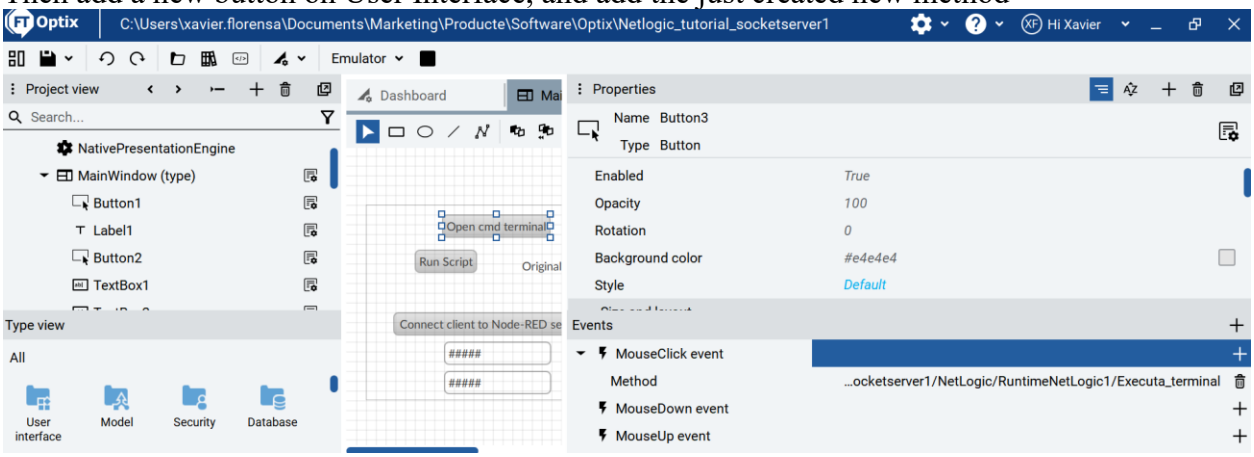


In Netlogic create a new Method

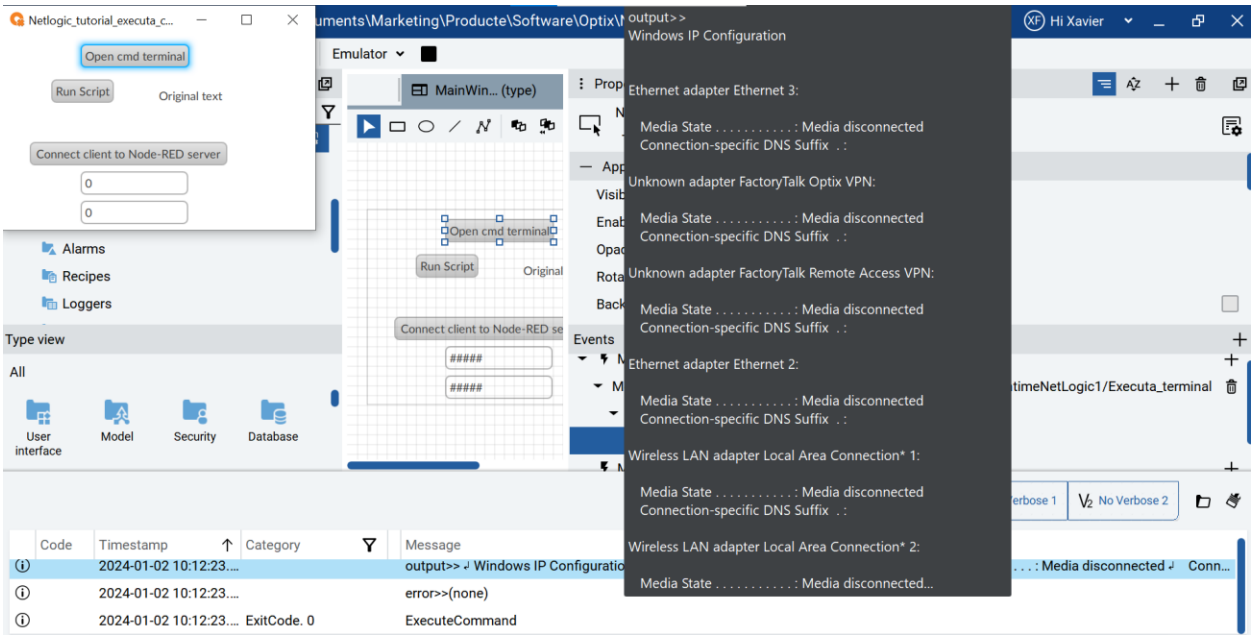
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.Text;
using System.Diagnostics;

#endregion

public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    //static void Executa_terminal(string command)
    public void Executa_terminal()
{
    int exitCode;
    ProcessStartInfo processInfo;
    Process process;
```

```
    //processInfo = new ProcessStartInfo("cmd.exe", "/c " + command);
    processInfo = new ProcessStartInfo("cmd.exe");
    processInfo.CreateNoWindow = false;
    processInfo.UseShellExecute = false;
    // *** Redirect the output ***
    processInfo.RedirectStandardError = true;
    processInfo.RedirectStandardOutput = true;

    process = Process.Start(processInfo);
    process.WaitForExit();

    // *** Read the streams ***
    // Warning: This approach can lead to deadlocks, see Edit #2
    string output = process.StandardOutput.ReadToEnd();
    string error = process.StandardError.ReadToEnd();

    exitCode = process.ExitCode;

    Log.Info("output>>" + (String.IsNullOrEmpty(output) ? "(none)" : output));
    Log.Info("error>>" + (String.IsNullOrEmpty(error) ? "(none)" : error));
    Log.Info("ExitCode: " + exitCode.ToString(), "ExecuteCommand");
    process.Close();
}
```

Then add a new button on User Interface, and add the just created new method



That's all.
Now we want to execute a command
Let's use the original code
But no way to keep the cmd window open,
Instead you get the result on the emulator output



Let’s try to open a new window,
changing

```
processInfo.CreateNoWindow = true;
```

To
false

But the window is closed inmediately
Let’s try with cmd arguments
Cmd /k ipconfig
Now the window remains open

```
C:\Windows\System32\cmd.exe

Ethernet adapter VMware Network Adapter VMnet8:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::4636:4deb:df47:49c3%5
   IPv4 Address. . . . . . . . . . . : 192.168.175.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.163
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter vEthernet (WSL):

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::817a:e205:4dba:fc20%85
   IPv4 Address. . . . . . . . . . . : 172.20.160.1
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . :

C:\Windows\system32>
```

Let’s try with this in Netlogic

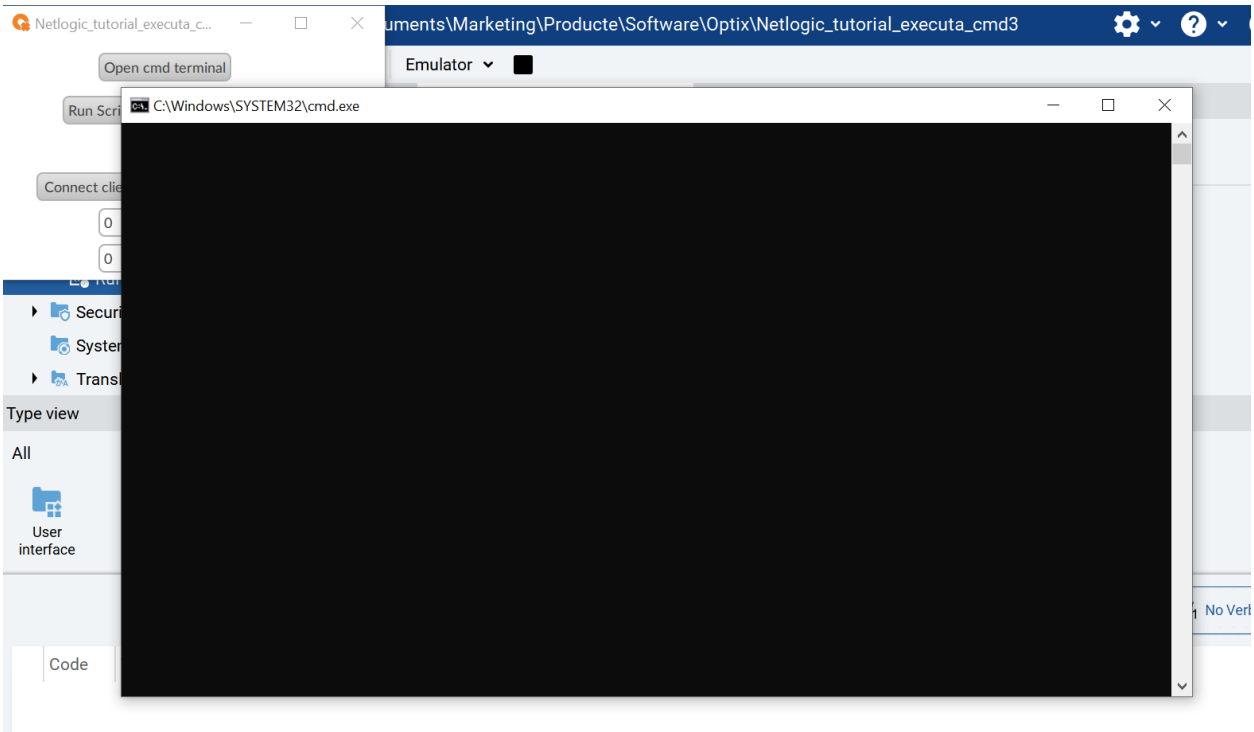
Changing this

```
    [ExportMethod]
    public void Executa_terminal(string command)
{
    int exitCode;
    ProcessStartInfo processInfo;
    Process process;

    processInfo = new ProcessStartInfo("cmd.exe", "/c " + command);
```

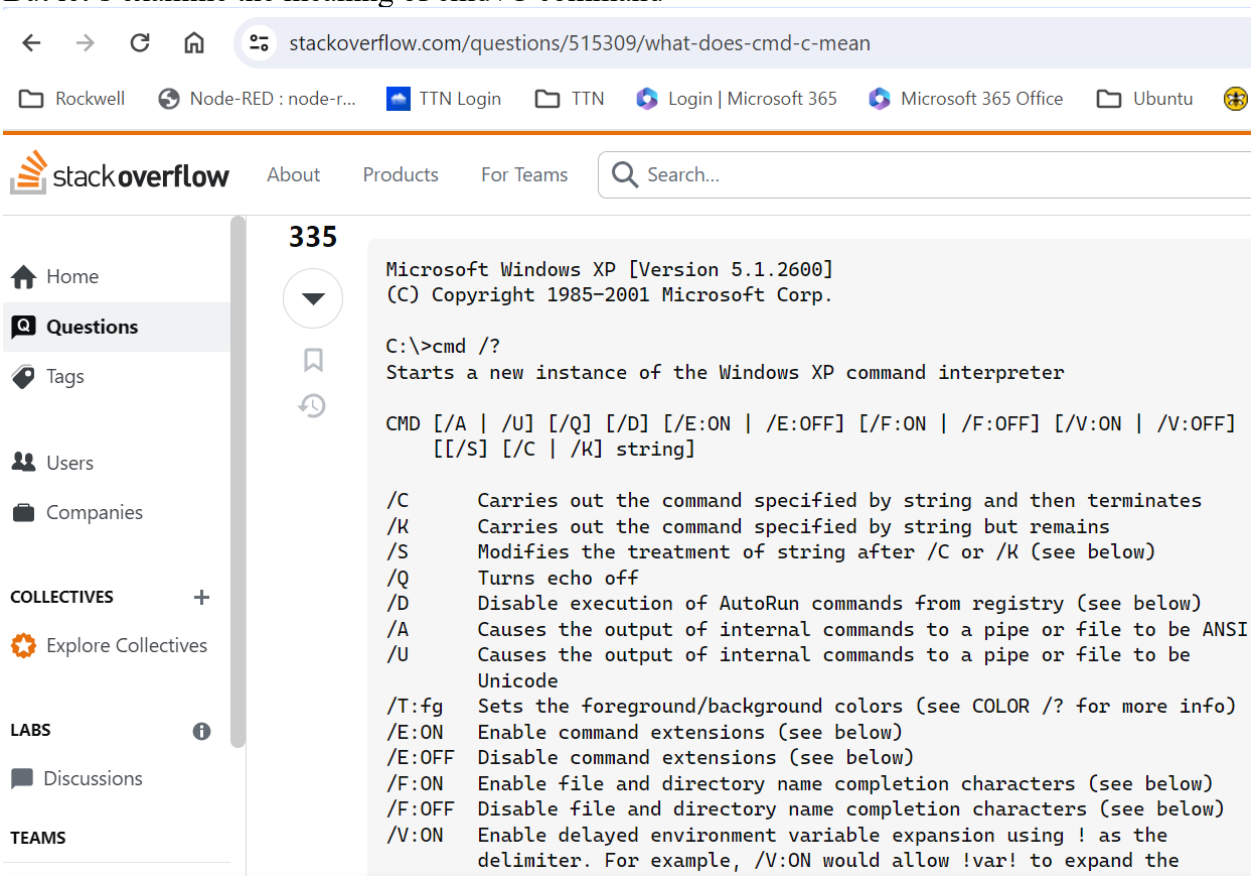
To this

```
    [ExportMethod]
    0 references
    public void Executa_terminal(string command)
{
    int exitCode;
    ProcessStartInfo processInfo;
    Process process;

    processInfo = new ProcessStartInfo("cmd.exe", "/k " + command);
```

But this time the terminal remains open with no output on it

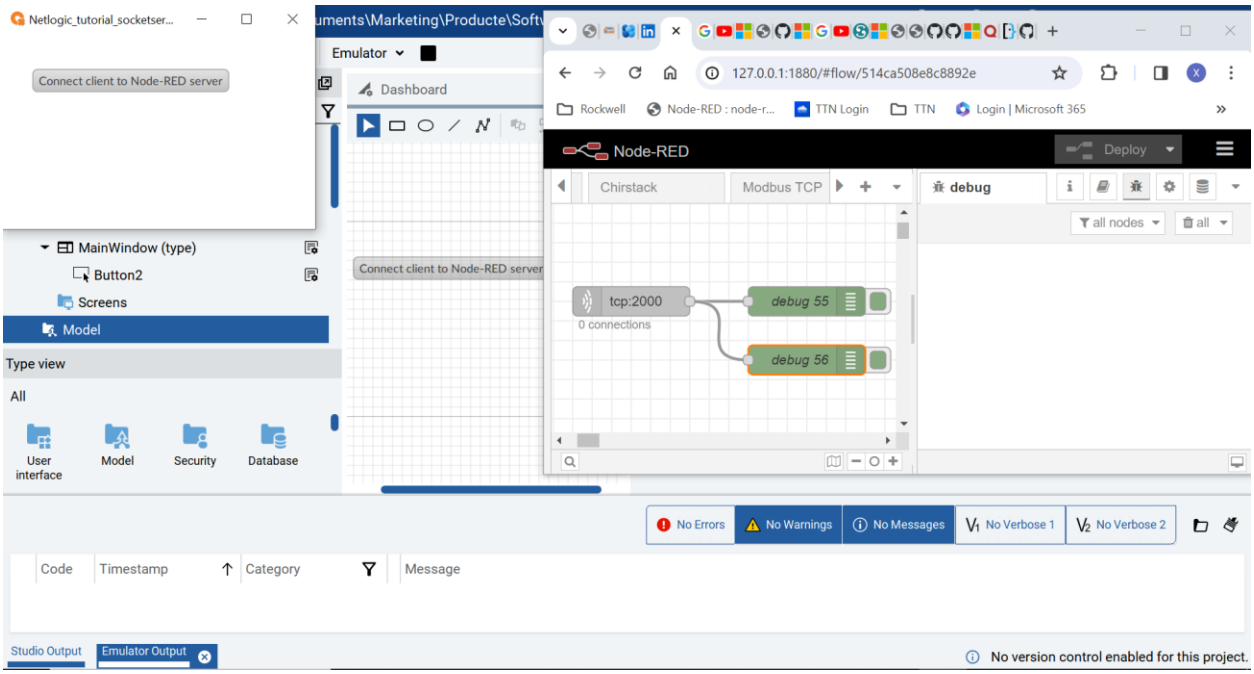


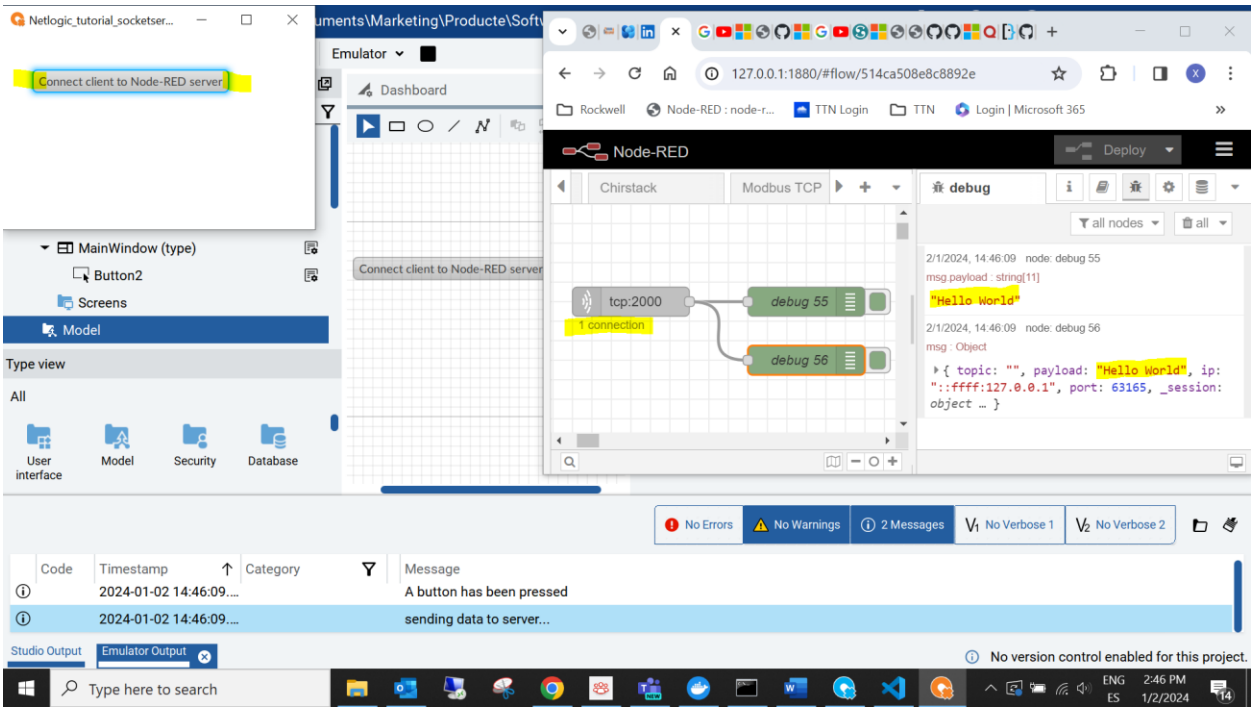
But let's examine the meaning of cmd /C command



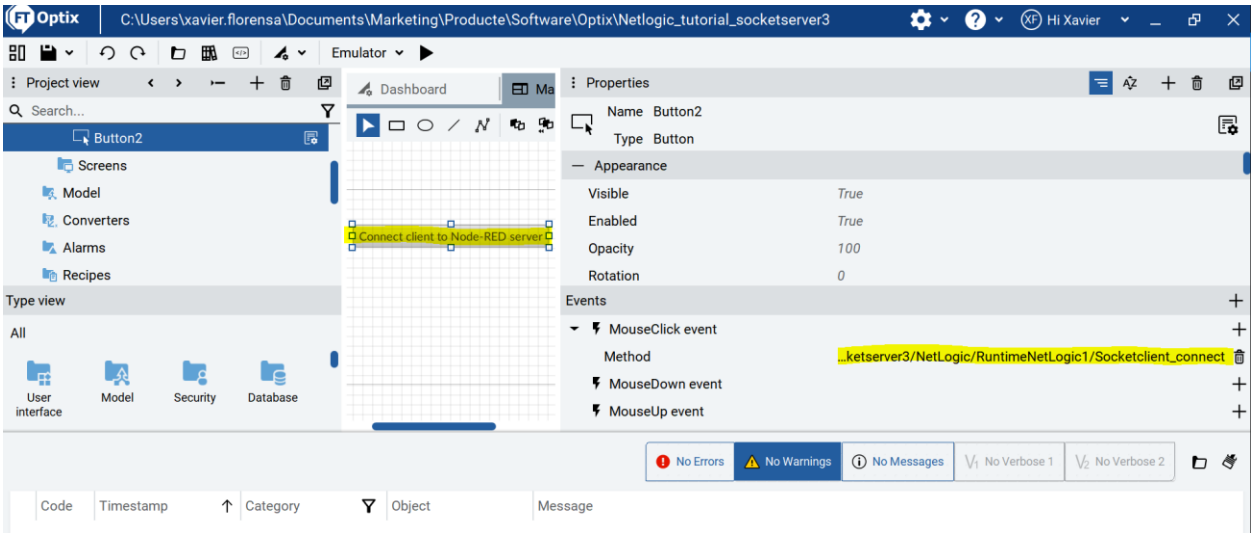
44. Socket TCP using System.Net.Sockets without external libraries

We have a socket server in Node-RED listening to port 2000.
Let's develop an application in FTOptix to connect to the server and to send a message "Hello World".





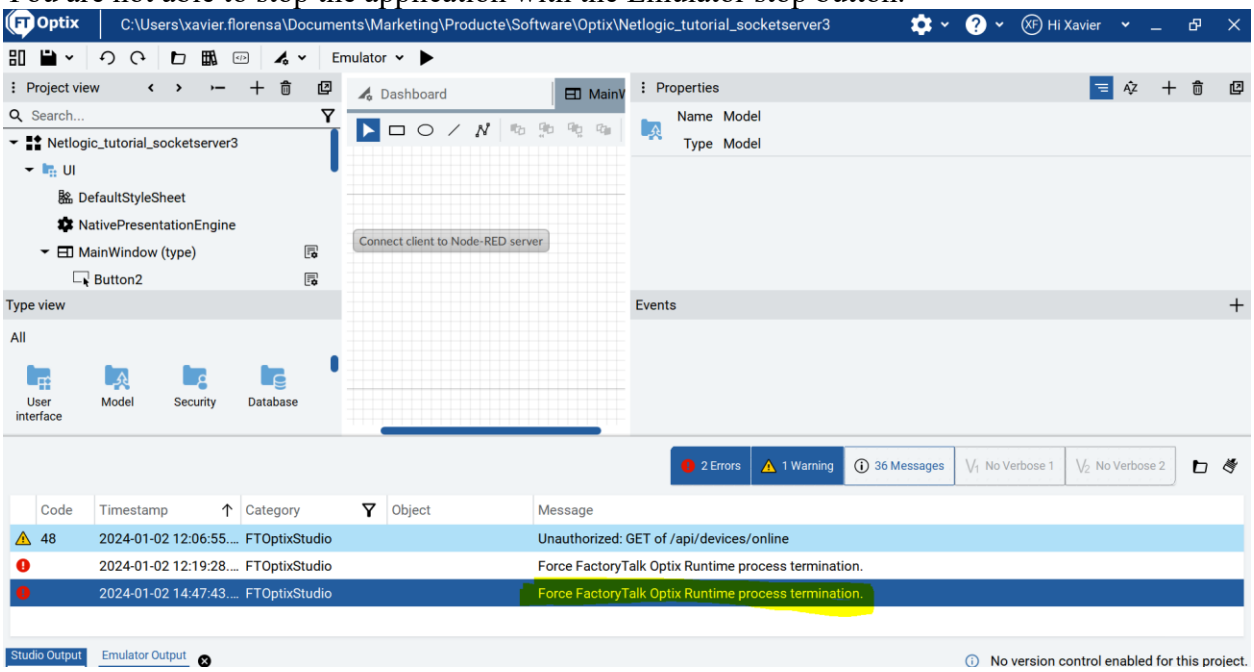
Using code pasted from this example
https://www.youtube.com/watch?v=g5yEWLJxNmI&t=11s
In this way inside Netlogic, using a Method




```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        string messageToSend = "Hello World";
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
```

```
        StreamReader sr = new StreamReader(stream);
        string response = sr.ReadLine();
        //Console.WriteLine(response);
        Log.Info(response);
        stream.Close();
        client.Close();
        //Console.ReadKey();
    }
}
```

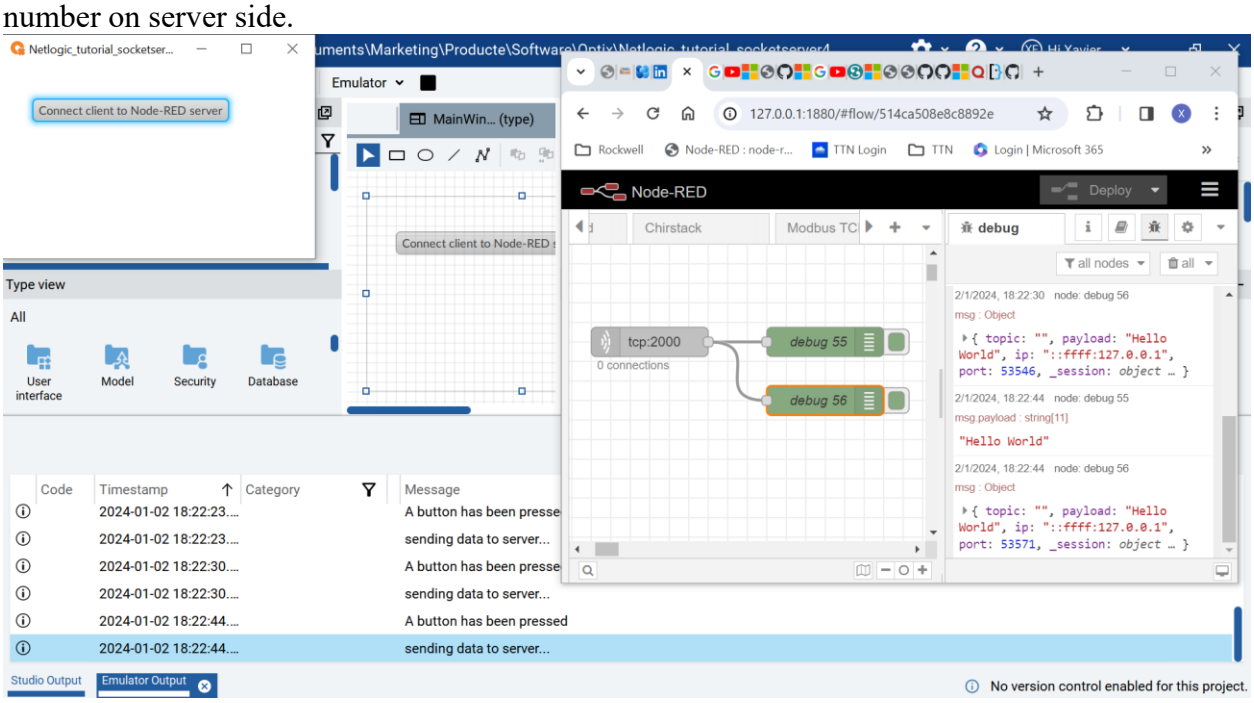
But there is a problem stopping this application
You are not able to stop the application with the Emulator stop button.



We can have better results if we comment the lines related to receiving data

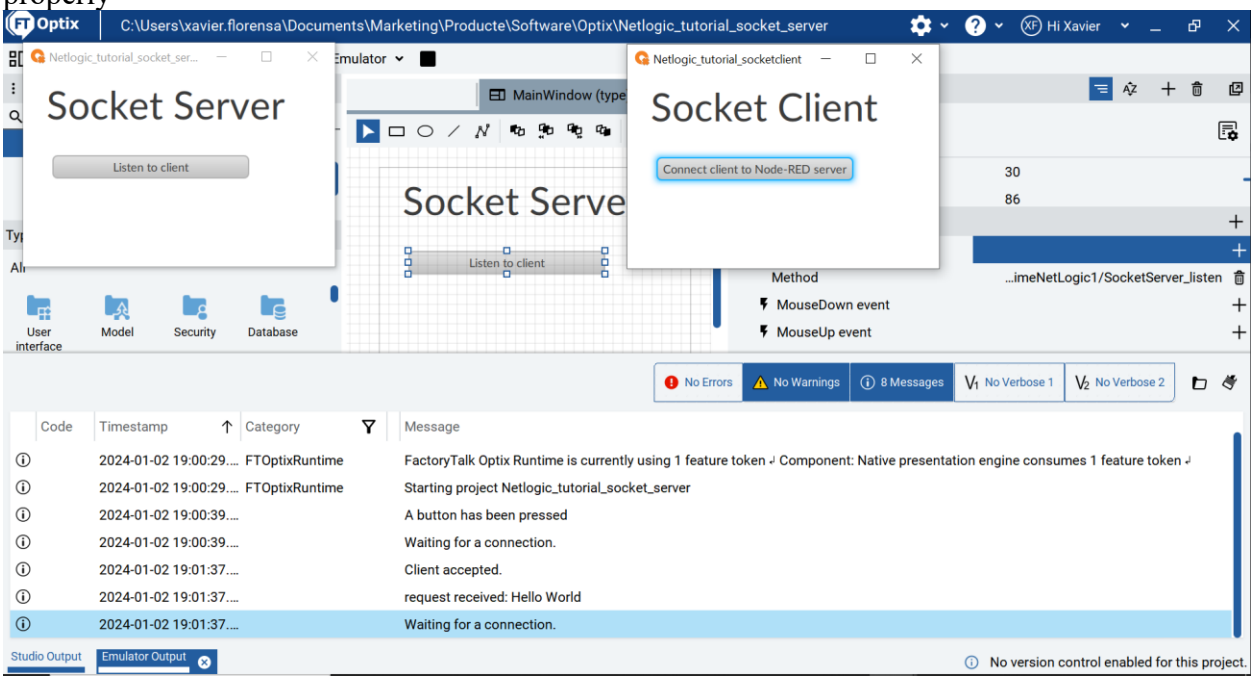
```
0 references
18  public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
19  {
20      [ExportMethod]
        0 references
21      public void Socketclient_connect()
22      {
23          Log.Info("A button has been pressed");
24          TcpClient client = new TcpClient("127.0.0.1", 2000);
25          string messageToSend = "Hello World";
26          int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
27          byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
28          NetworkStream stream = client.GetStream();
29          stream.Write(sendData, 0, sendData.Length);
30          //Console.WriteLine("sending data to server...");
31          Log.Info("sending data to server...");
32          //StreamReader sr = new StreamReader(stream);
33          //string response = sr.ReadLine();
34          //Console.WriteLine(response);
35          //Log.Info(response);
36          stream.Close();
37          client.Close();
38          //Console.ReadKey();
39      }
40  }
```

Now you can stop the program with the stop key
And you can execute several times the send button. But you will not see any connected client number on server side.

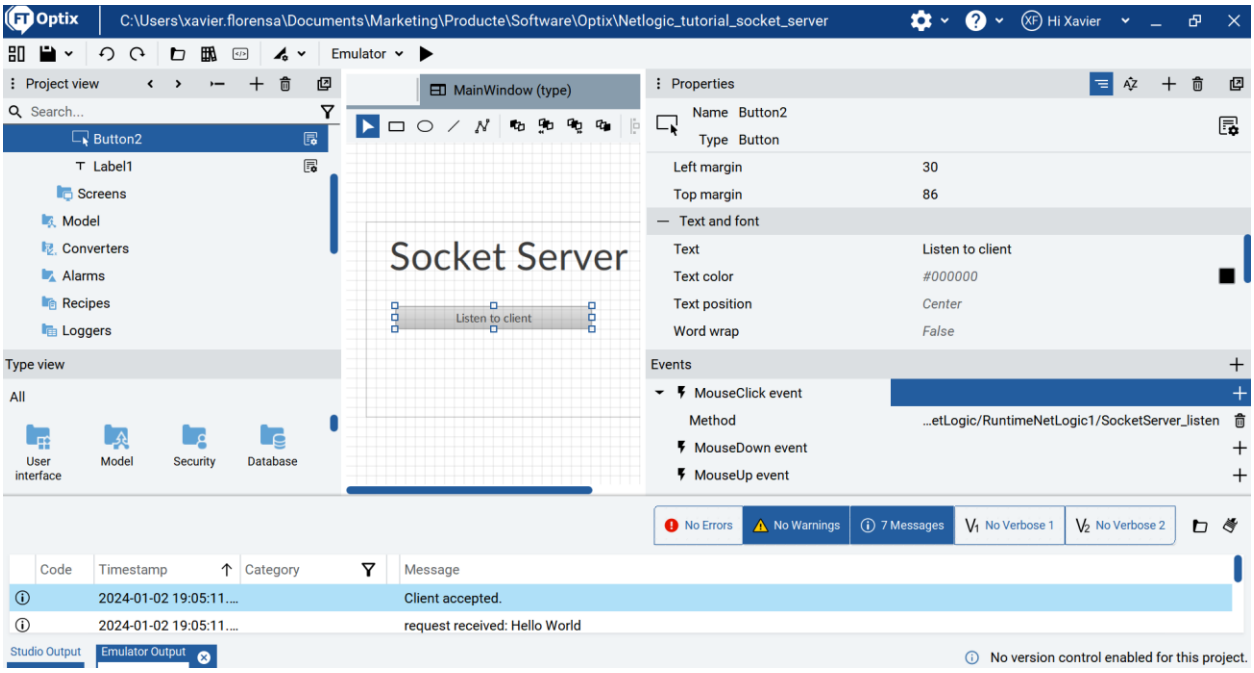


Now let's try to write an Optix application as a Socket server.

We managed to make a transaction of Hello World, but the server application does not stop properly



You can connect and send messages several times.
This is done with this code on the server side



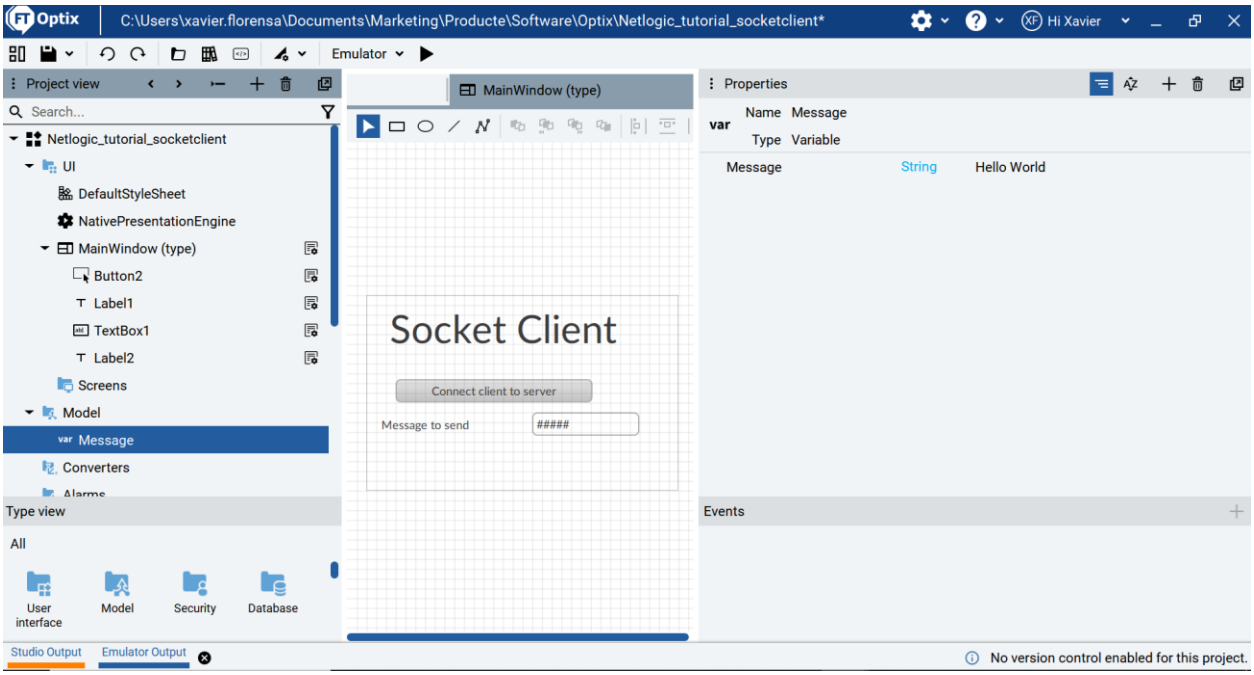
And this code on the server side Netlogic

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
```

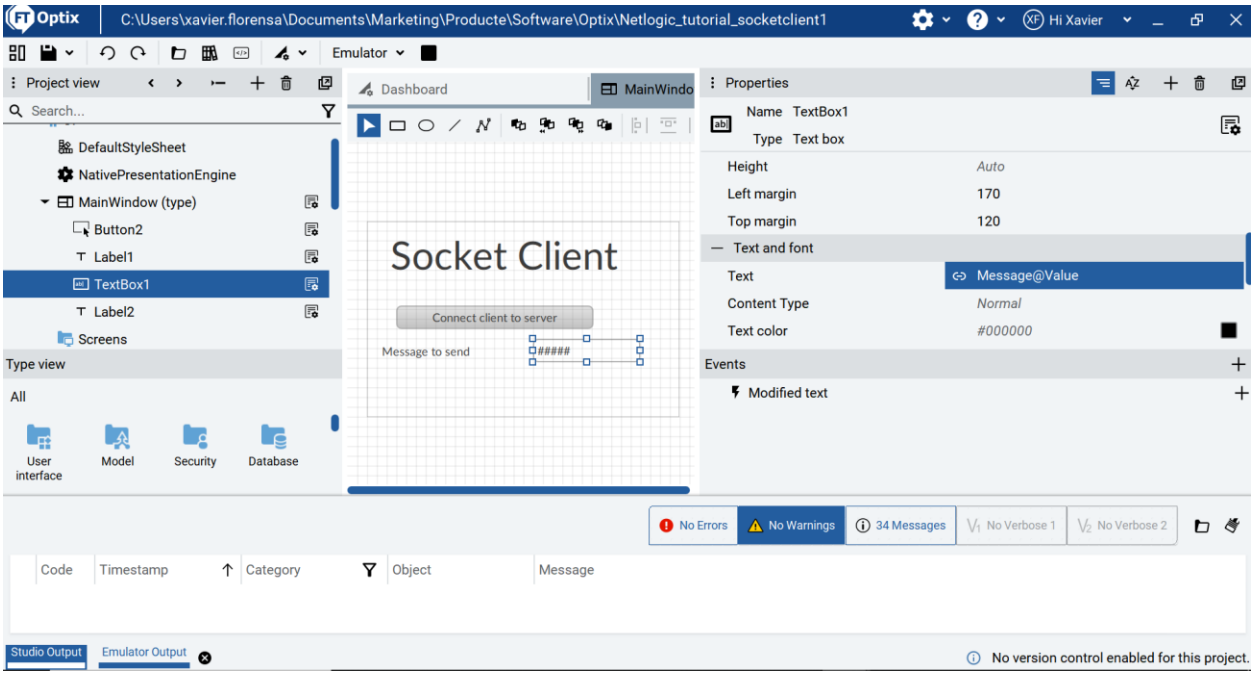
```
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen()
    {
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
        {
            //Console.WriteLine("Waiting for a connection.");
            Log.Info("Waiting for a connection.");
            TcpClient client = listener.AcceptTcpClient();
            //Console.WriteLine("Client accepted.");
            Log.Info("Client accepted.");
            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());
            try
            {
                byte[] buffer = new byte[1024];
                stream.Read(buffer, 0, buffer.Length);
                int recv = 0;
                foreach (byte b in buffer)
                {
                    if (b!=0)
                    {
                        recv++;
                    }
                }
                string request = Encoding.UTF8.GetString(buffer, 0, recv);
                //Console.WriteLine("request received: "+ request);
                Log.Info("request received: "+ request);
```

```
                sw.WriteLine("You rock!");
                sw.Flush();
            }
            catch(Exception e)
            {
                //Console.WriteLine("Something went wrong.");
                Log.Info("Something went wrong.");
                sw.WriteLine(e.ToString());
            }
        }
    }
}
```

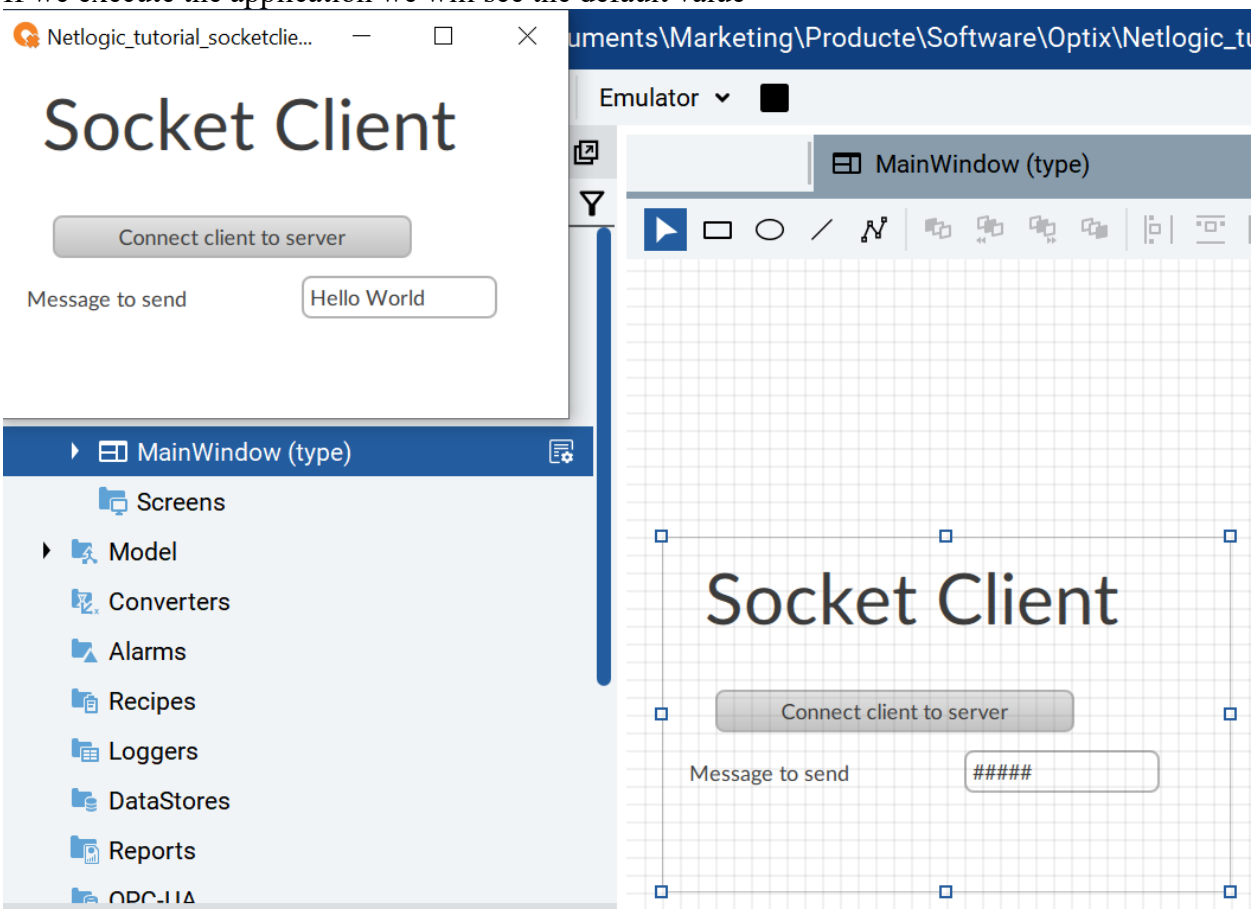
Now let' try to improve the code with some text boxes
First of all let's introduce the text message to send manually from a Text Box
Let's go to client and add a Message string variable.
We use a default variable value "Hello World"



And a Textbox with a dynamic link to variable Message



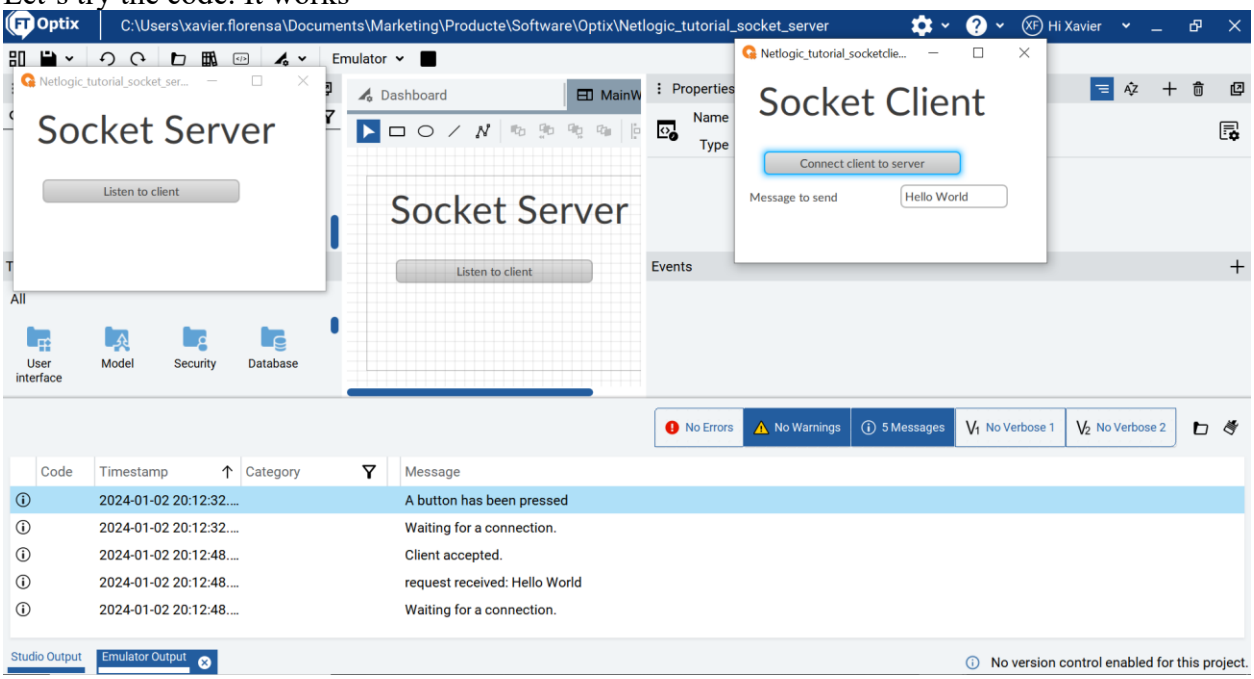
If we execute the application we will see the default value



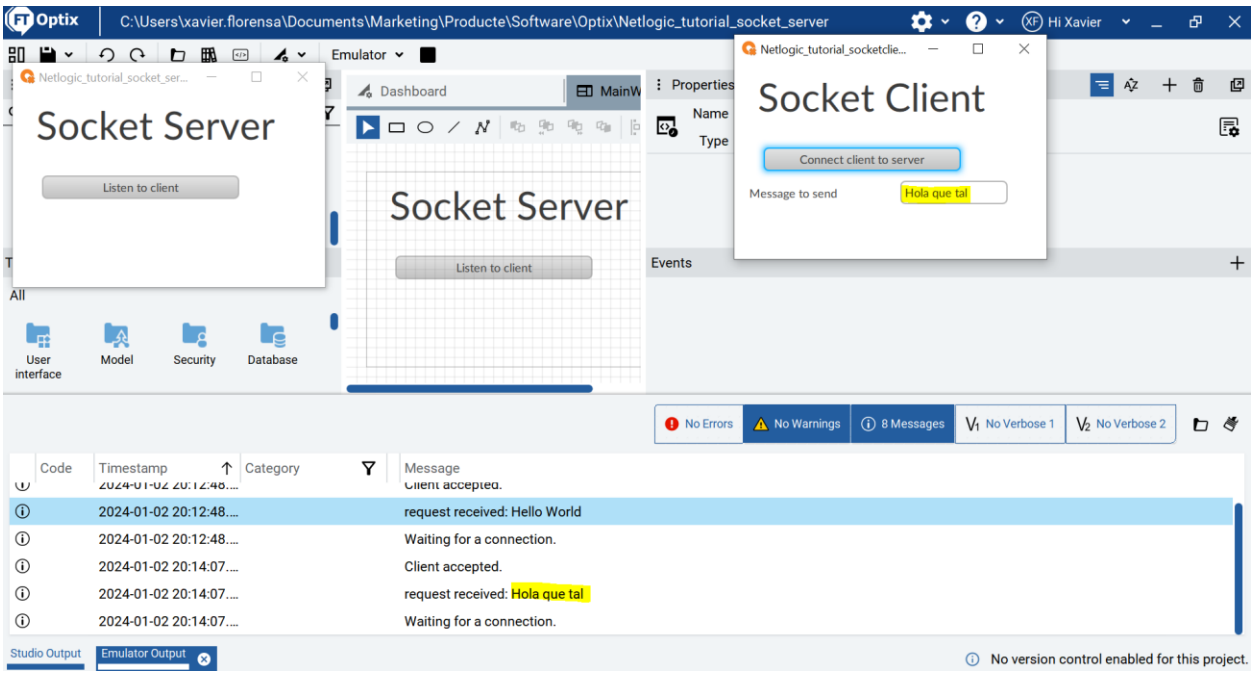
But we have to pass that string value to our code, by means of the UI variable Message

```
[ExportMethod]
0 references
public void Socketclient_connect()
{
    Log.Info("A button has been pressed");
    TcpClient client = new TcpClient("127.0.0.1", 2000);
    var missatge = Project.Current.GetVariable("Model/Message");
    //string messageToSend = "Hello World";
    string messageToSend = missatge.Value;
```

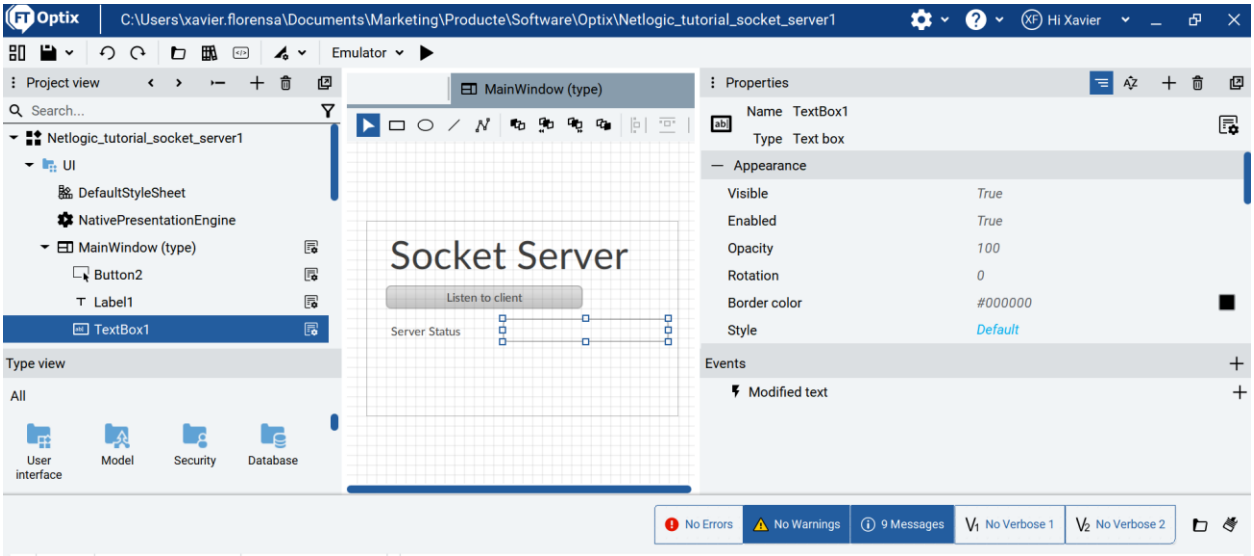
Let’s try the code. It works



And we can send different messages



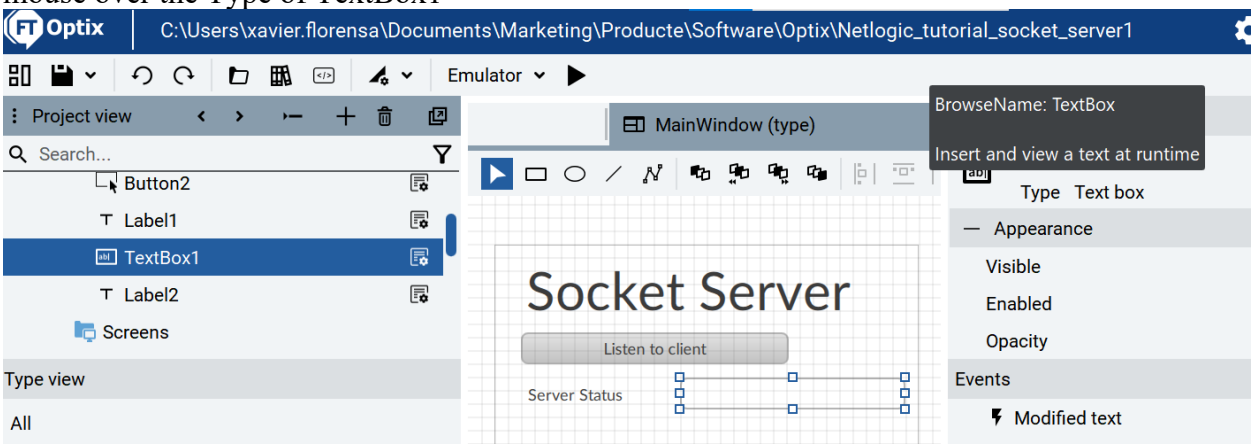
Let's improve the application with a status window on the server
Let's create a new Textbox



We will use the NodeId of the Textbox to be able to display text on it.
It's easy to use the Method we have already created to pass an argument as the Textbox NodeId
First of all we have to change the Method to require an input parameter, the NodeId.
This way

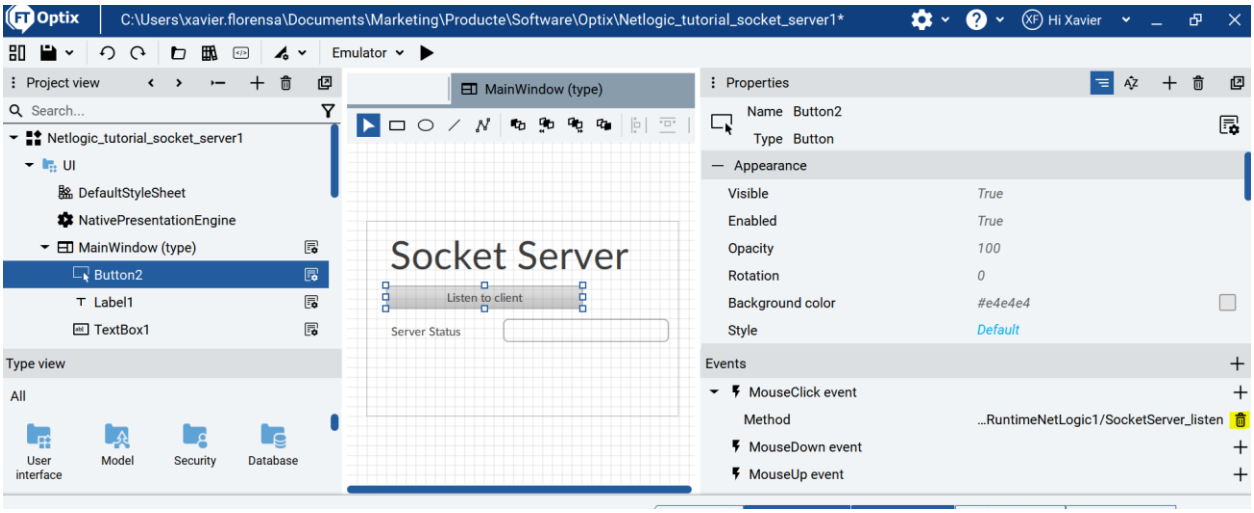
```
[ExportMethod]
0 references
public void SocketServer_listen(NodeId textboxNodeId)
{
    var textbox = InformationModel.Get<TextBox>(textboxNodeId);
    textbox.Text = "A button has been pressed";
    Log.Info("A button has been pressed");
    TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
    listener.Start();
    while (true)
    {
```

Use this name TextBox in brackets after Get, as is the name displayed here when you move the mouse over the Type of TextBox1

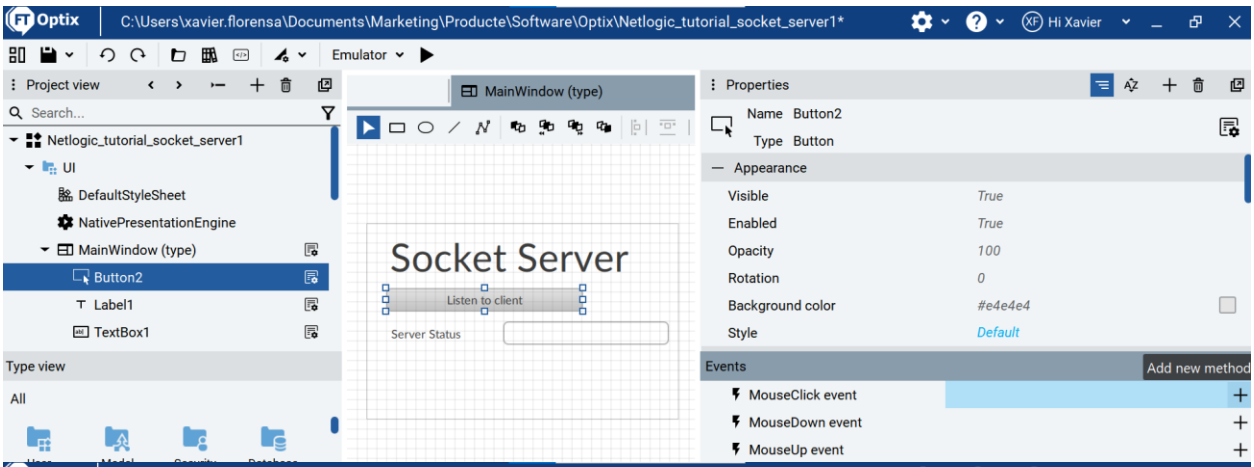


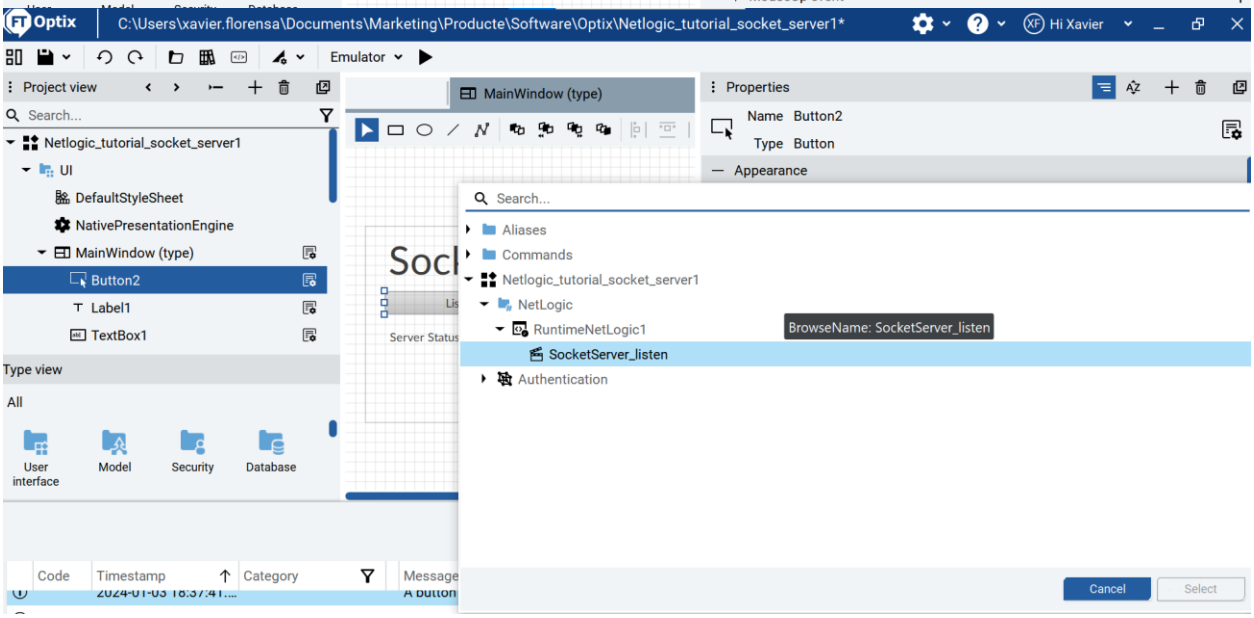
Now let's save the code and let's assign a dynamic link between the Method input parameter and TextBox1

We have to delete the method call after a button click

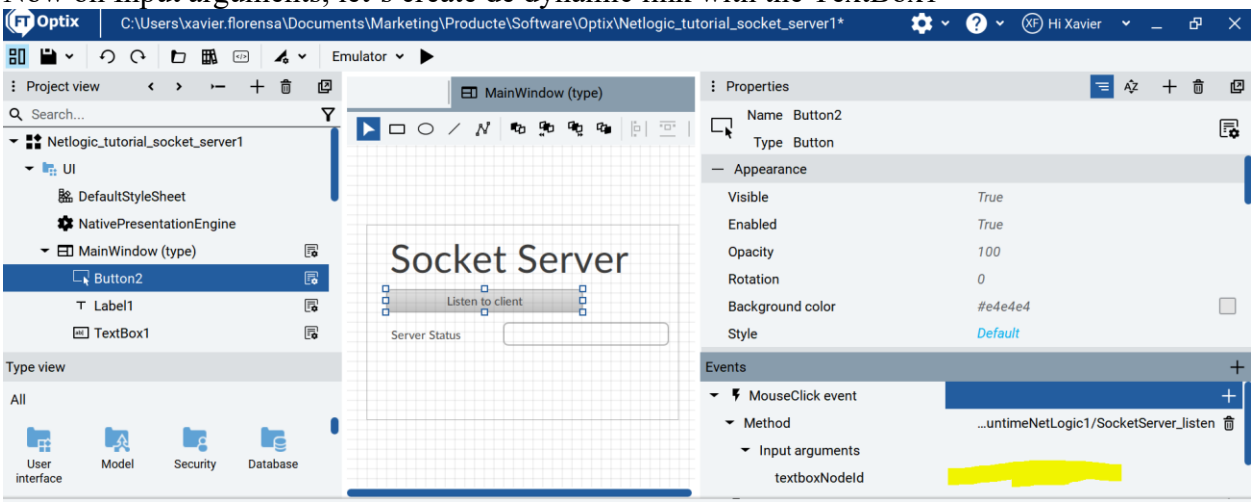


And create a new action on Button click

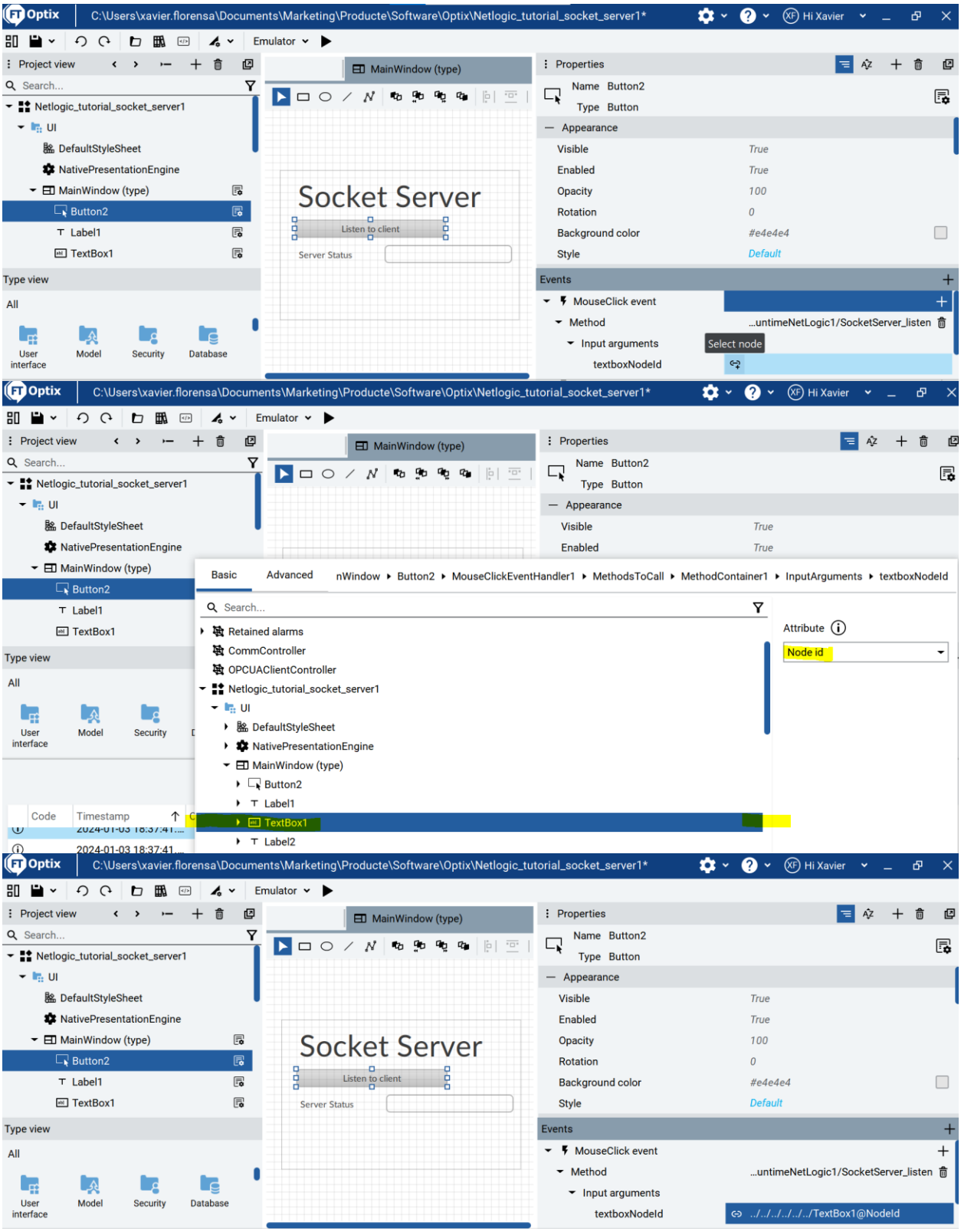




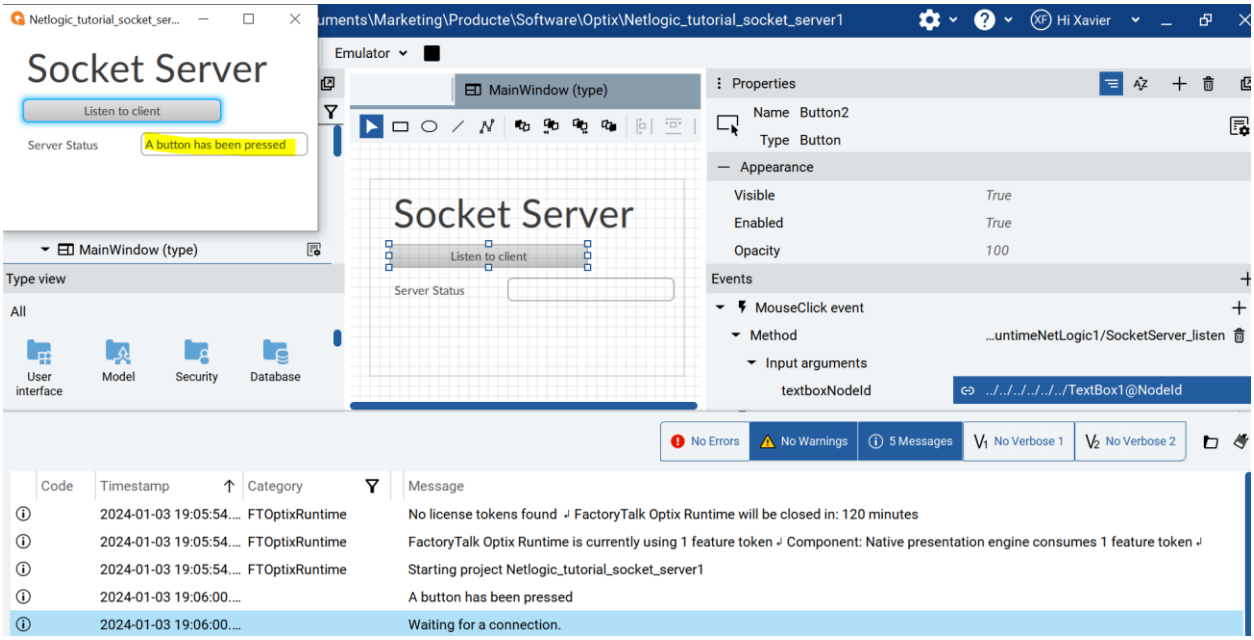
Now on Input arguments, let's create de dynamic link with the TextBox1



So select node



Let’s try it to see that “a button has been pressed” on the Textox on Emulator runtime window



This is working,

Now let's display further messages by modifying Netlogic this way, before each Log.Info sentence, let's use our TextBox.

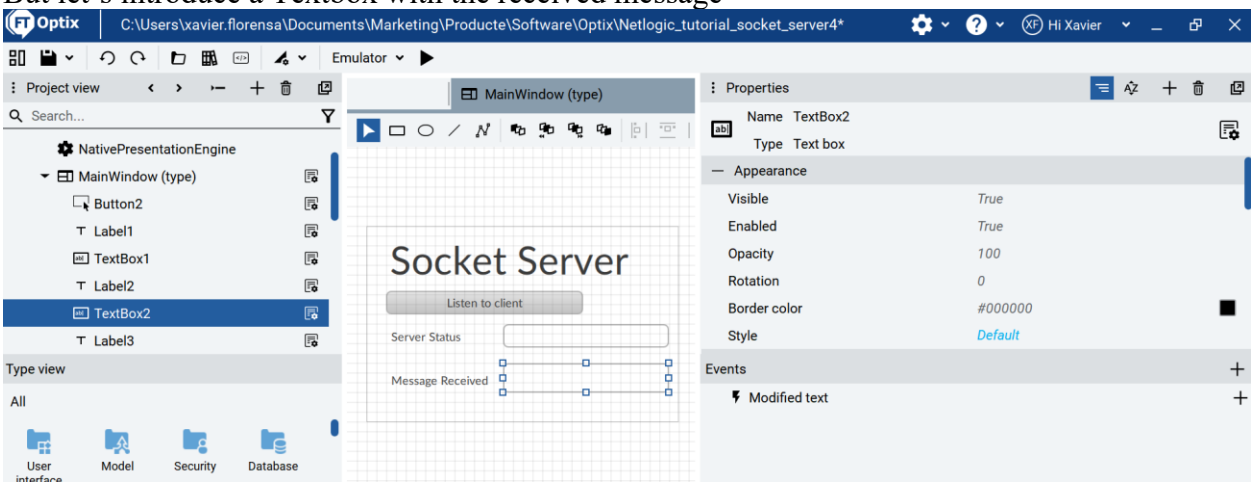
```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
        {
            //Console.WriteLine("Waiting for a connection.");
            textbox.Text = "Waiting for a connection.";
            Log.Info("Waiting for a connection.");
            TcpClient client = listener.AcceptTcpClient();
            //Console.WriteLine("Client accepted.");
            textbox.Text = "Client accepted.";
            Log.Info("Client accepted.");
            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());
            try
            {
```

Save the code and run the Emulator

This is working but you do not see the message Client accepted since it changes too fast to another waiting for connection.
So let's add a litte pause for 1 second at the end of the While (true) loop

```
    }
    System.Threading.Thread.Sleep(1000);
}
```

Since this code closes the connection after having received the message.
(this is another improvement to make to the code later on, to keep the socket connection open)
But let's introduce a Textbox with the received message



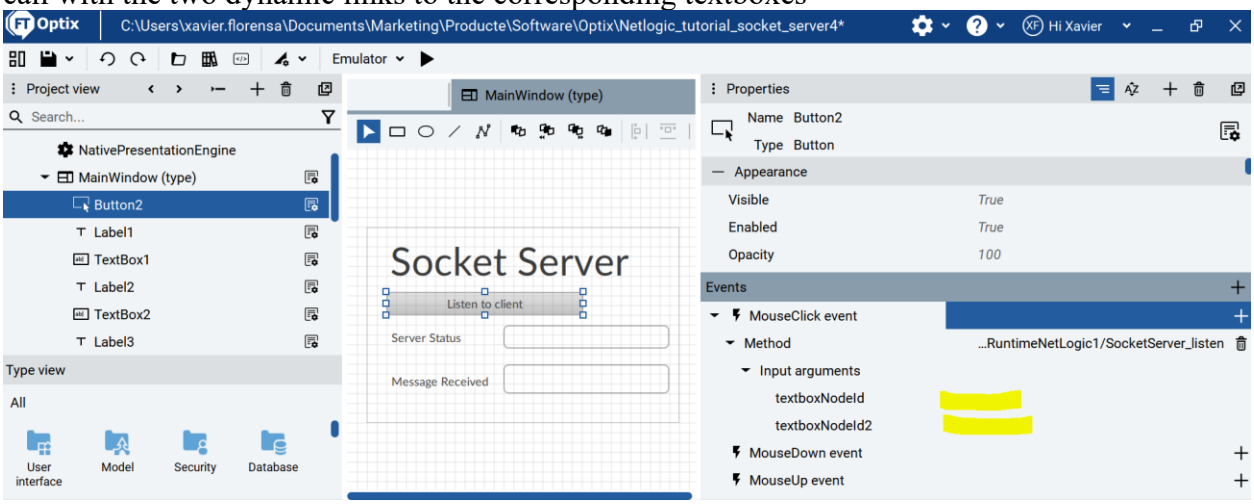
And modifying Netlogic
so let's change the Methos again with a second input parameter like this.

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId,NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
```

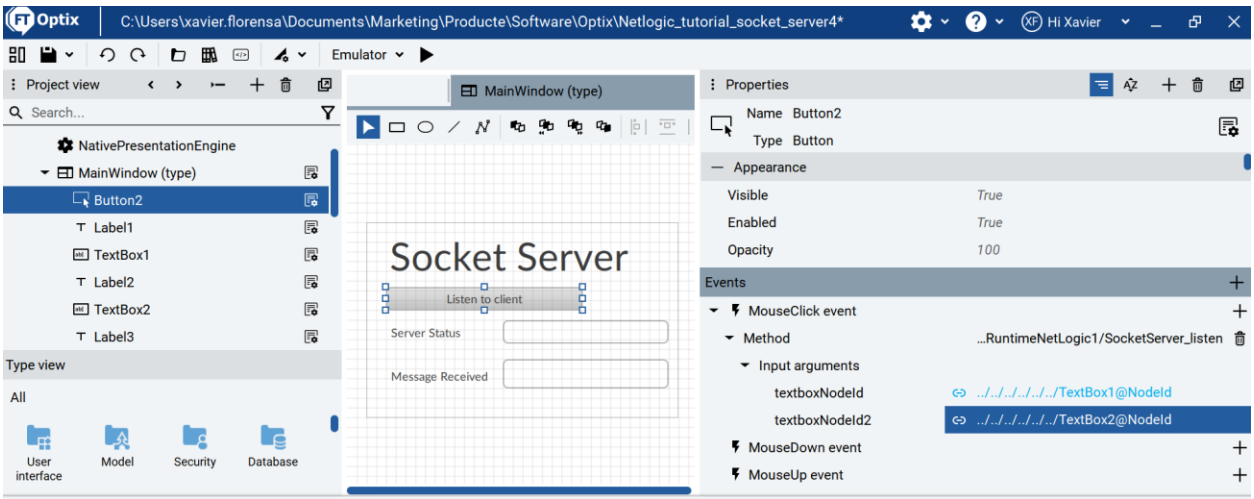
And this is the point where we will show the received message

```
string request = Encoding.UTF8.GetString(buffer, 0, recv);
//Console.WriteLine("request received: "+ request);
textbox2.Text = request;
Log.Info("request received: "+ request);
```

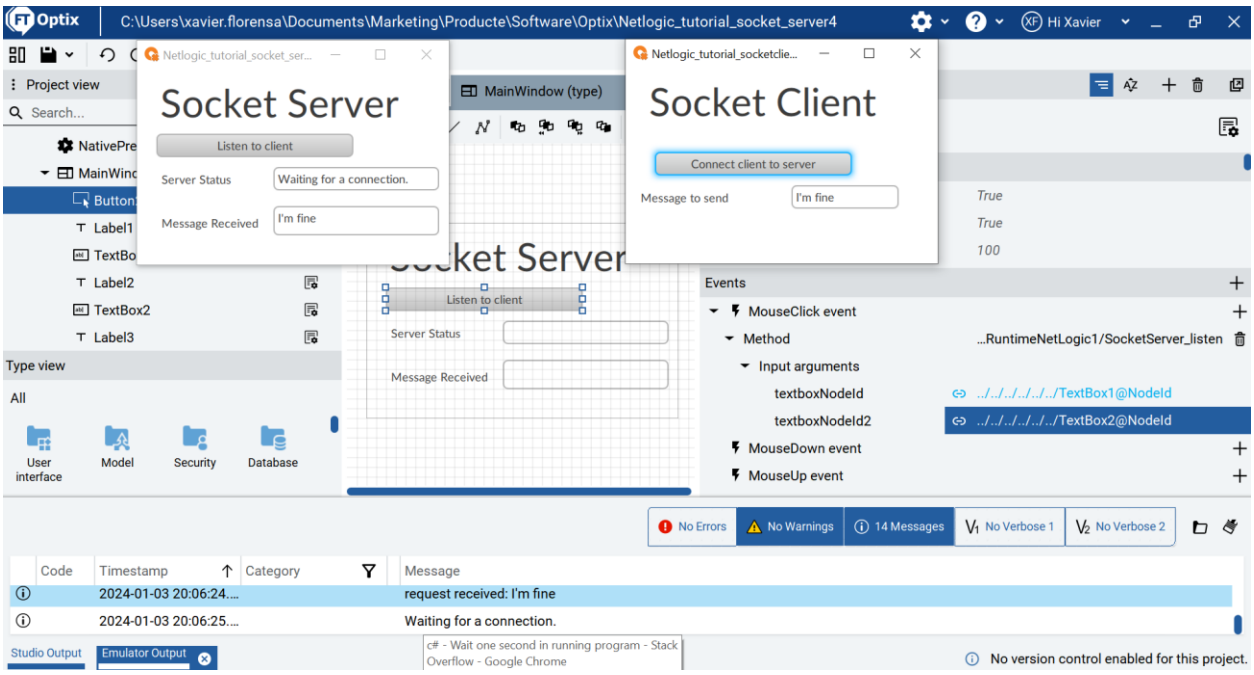
Let’s save the code, erase the Method call after a button click and generate again the button click call with the two dynamic links to the corresponding textboxes



Like this



Let’s run the application



It works!

This is the code for the server

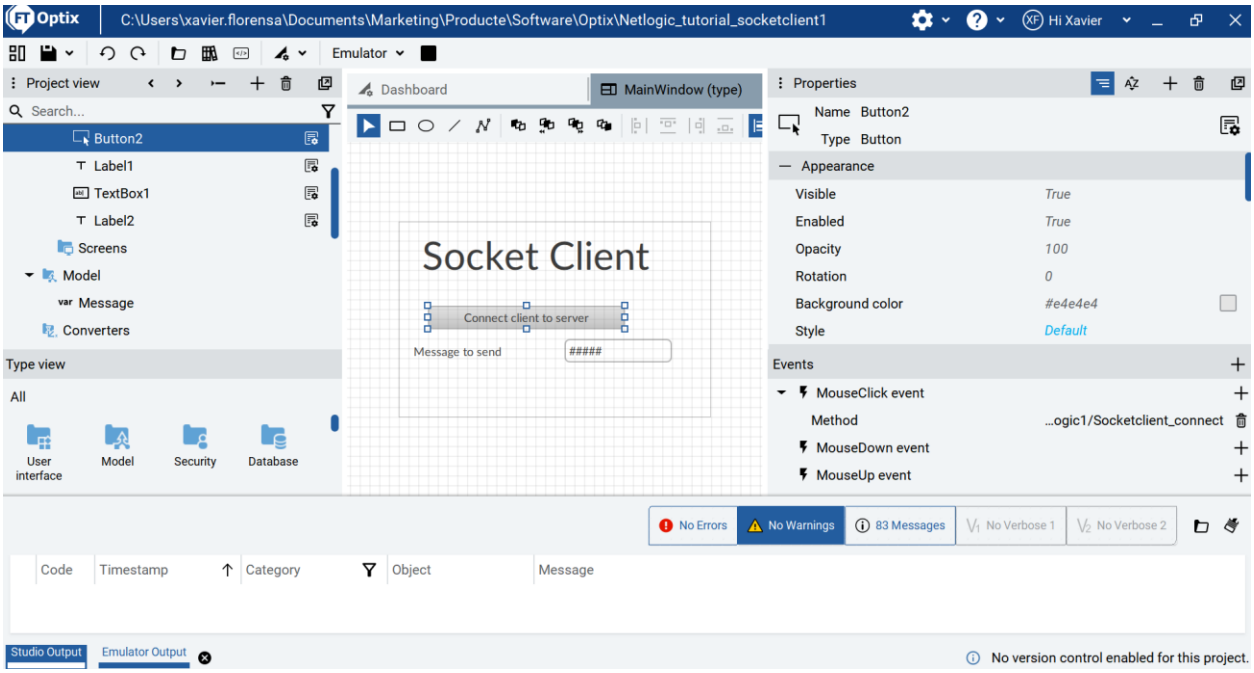
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId,NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
```

```
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        while (true)
        {
            //Console.WriteLine("Waiting for a connection.");
            textbox.Text = "Waiting for a connection.";
            Log.Info("Waiting for a connection.");
            TcpClient client = listener.AcceptTcpClient();
            //Console.WriteLine("Client accepted.");
            textbox.Text = "Client accepted.";
            Log.Info("Client accepted.");

            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());

            try
            {
                byte[] buffer = new byte[1024];
                stream.Read(buffer, 0, buffer.Length);
                int recv = 0;
                foreach (byte b in buffer)
                {
                    if (b!=0)
                    {
                        recv++;
                    }
                }
                string request = Encoding.UTF8.GetString(buffer, 0, recv);
                //Console.WriteLine("request received: "+ request);
                textbox2.Text = request;
                Log.Info("request received: "+ request);
                sw.WriteLine("You rock!");
                sw.Flush();
            }
            catch(Exception e)
            {
                //Console.WriteLine("Something went wrong.");
                Log.Info("Something went wrong.");
                sw.WriteLine(e.ToString());
            }
            System.Threading.Thread.Sleep(1000);
        }
    }
}
```

And this is the code for the client

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        string messageToSend = missatge.Value;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        stream.Close();
        client.Close();
        //Console.ReadKey();
    }
 }
```

[redacted]



As you can see on this video
https://youtu.be/6al-MB8CBQg
You can get the code here
https://github.com/xavierflorensa/Netlogic_tutorial_socket_server5.git
https://github.com/xavierflorensa/Netlogic_tutorial_socketclient2.git
Now let's improve the code
We do not want the socket connection to be closed after sending the message.
Let's comment the stream close and client close sentences at the end of the code

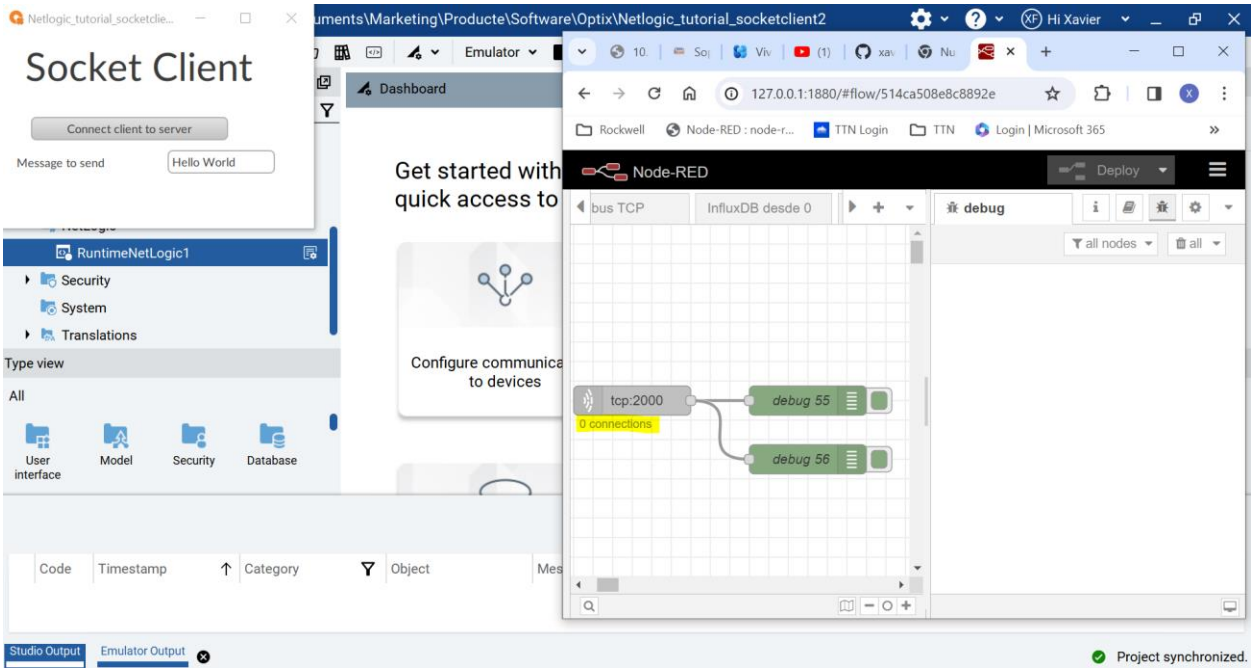
```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        string messageToSend = missatge.Value;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
```

```
        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        //stream.Close();
        //client.Close();
        //Console.ReadKey();
    }
 }
```

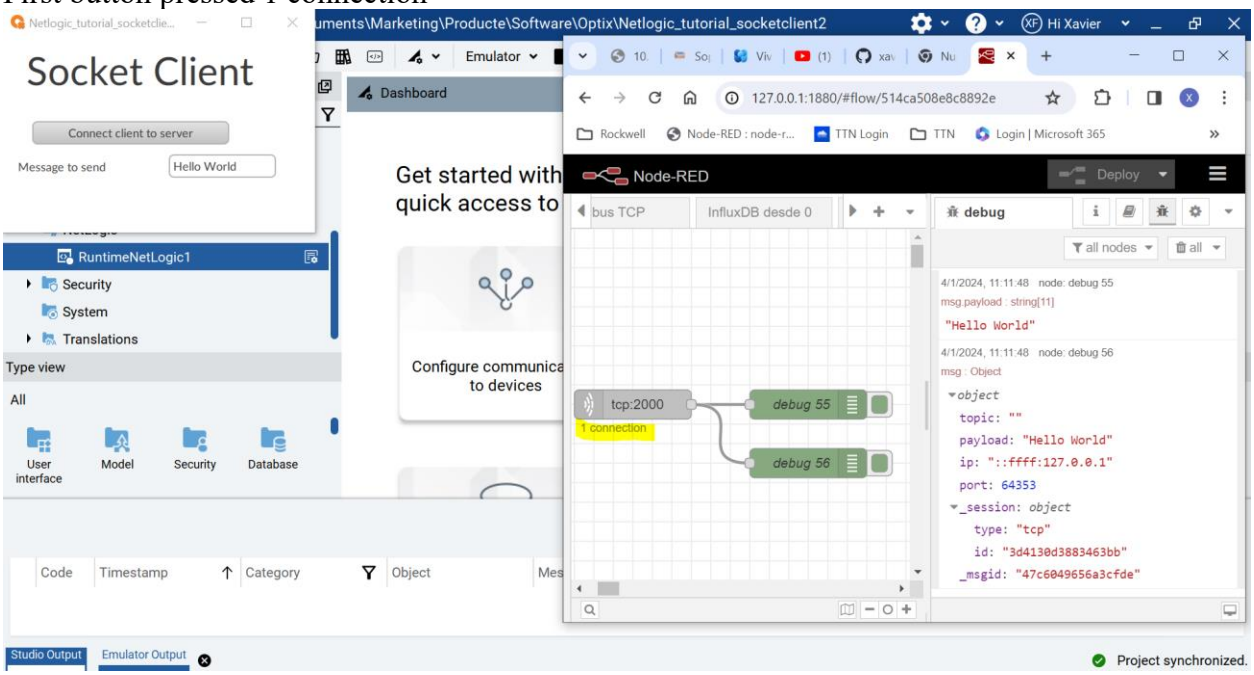
Now the connection is not closed, But each time we hit on the button a new socket is opened, and the message arrives on the new socket.

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void Socketclient_connect()
    {
        Log.Info("A button has been pressed");
        TcpClient client = new TcpClient("127.0.0.1", 2000);
        var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        string messageToSend = missatge.Value;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        //stream.Close();
        //client.Close();
        //Console.ReadKey();
    }
 }
```

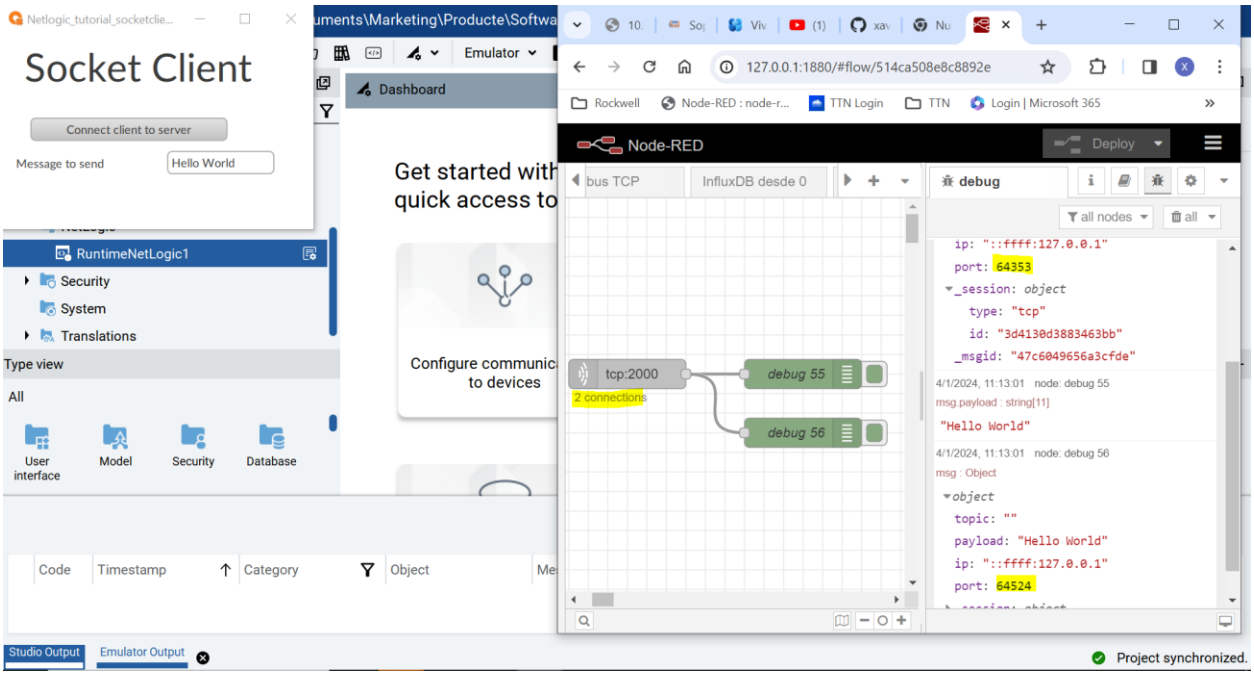
As we can monitor with Node-RED
Before clicking button
We have 0 connections



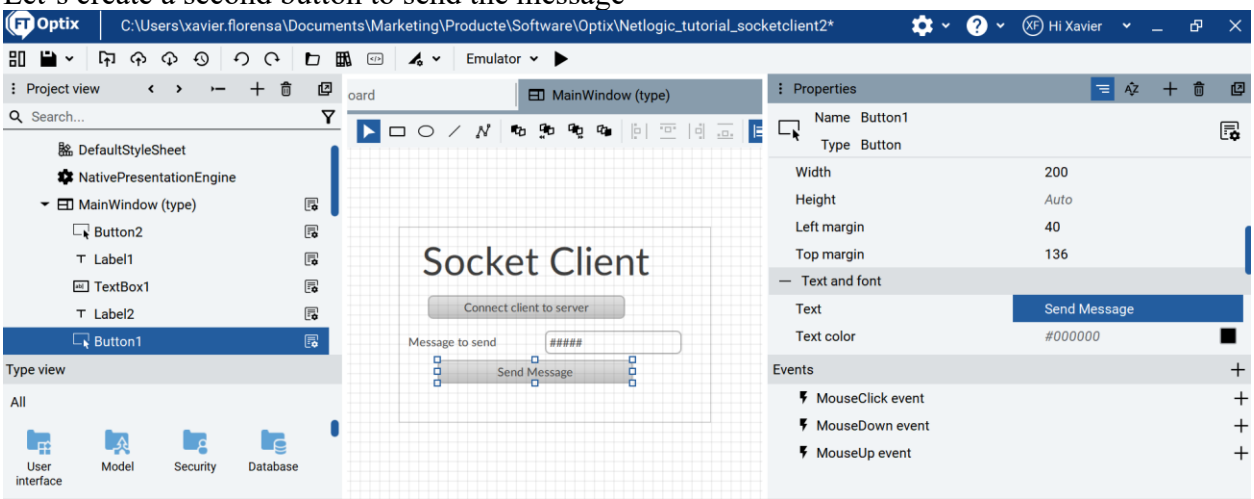
First button pressed 1 connection



Second button pressed, 2 connections and two different ports



We do not want to have 2 connections.
Let's try to separate the connection and the message sending, with a second button to send the message
Let's create a second button to send the message

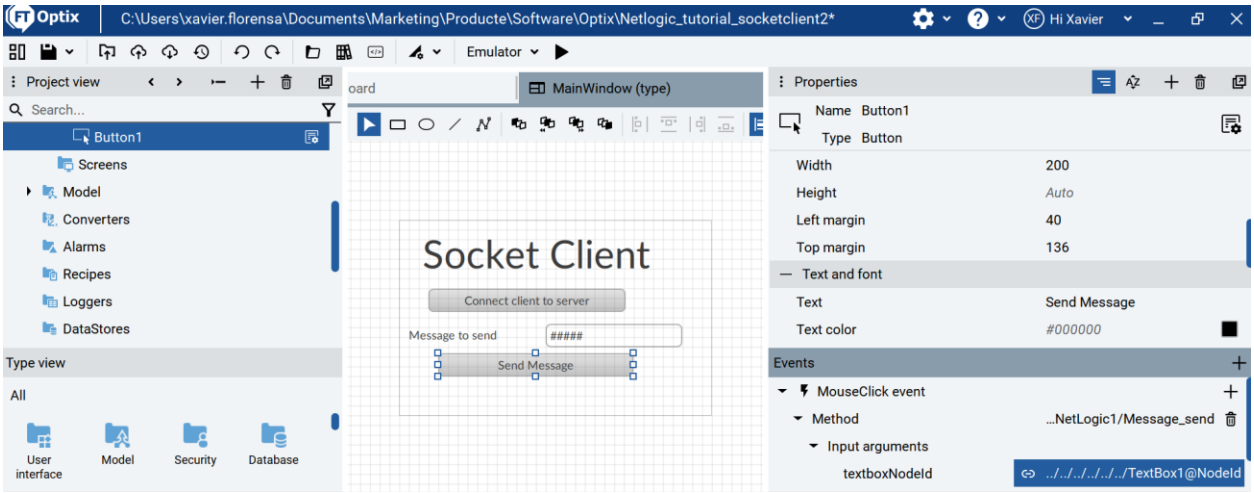


Let's create a new method with an input argument as we did on the server side.
And put the new client creation sentence outside of the first button, so the new client is created just when executing the application and the connect button is useless (we will improve this later on)

```
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    TcpClient client = new TcpClient("127.0.0.1", 2000);

    [ExportMethod]
    public void Socketclient_connect()
    {
```

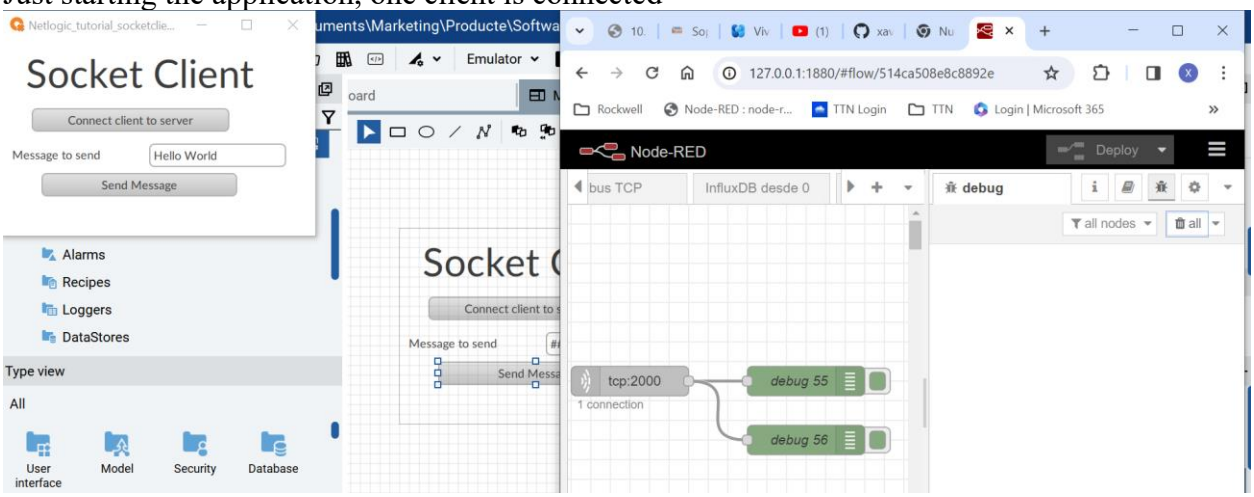
```
        Log.Info("A button has been pressed");
        //TcpClient client = new TcpClient("127.0.0.1", 2000);
        //var missatge = Project.Current.GetVariable("Model/Message");
        //string messageToSend = "Hello World";
        //string messageToSend = missatge.Value;
        //int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        //byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        //NetworkStream stream = client.GetStream();
        //stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        //Log.Info("sending data to server...");
        //StreamReader sr = new StreamReader(stream);
        //string response = sr.ReadLine();
        //Console.WriteLine(response);
        //Log.Info(response);
        //stream.Close();
        //client.Close();
        //Console.ReadKey();
    }
    public void Message_send(NodeId textboxNodeId)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        Log.Info("A button has been pressed");
        var missatge = textbox.Text;
        string messageToSend = missatge;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
    }
}
```

Save the code, erase the method action after pressing button 1 and create Method action after pushing button 2. Then create dynamic link between input parameter and textbox.

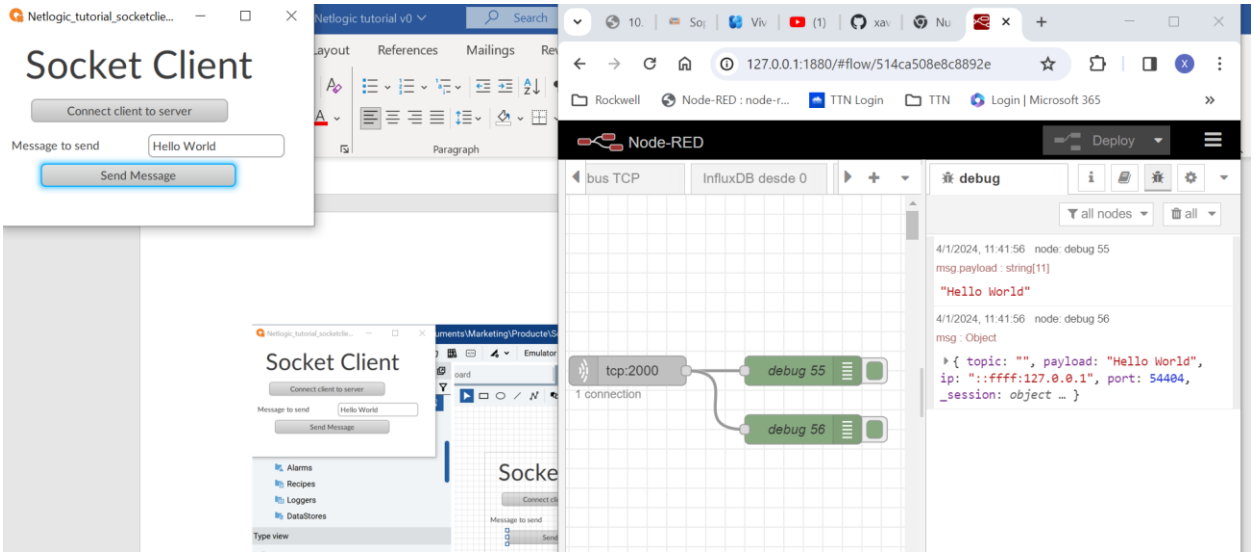


Test the application

Just starting the application, one client is connected



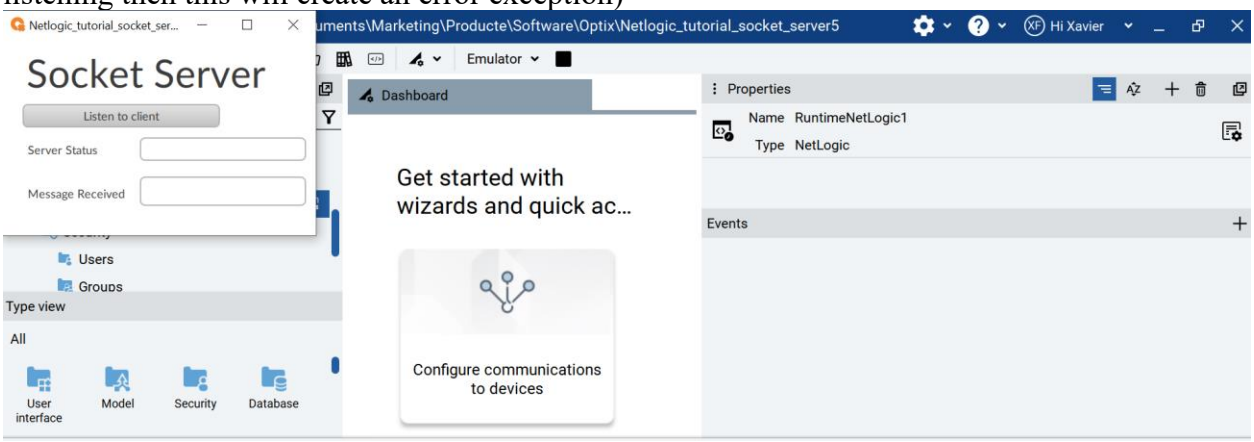
Send



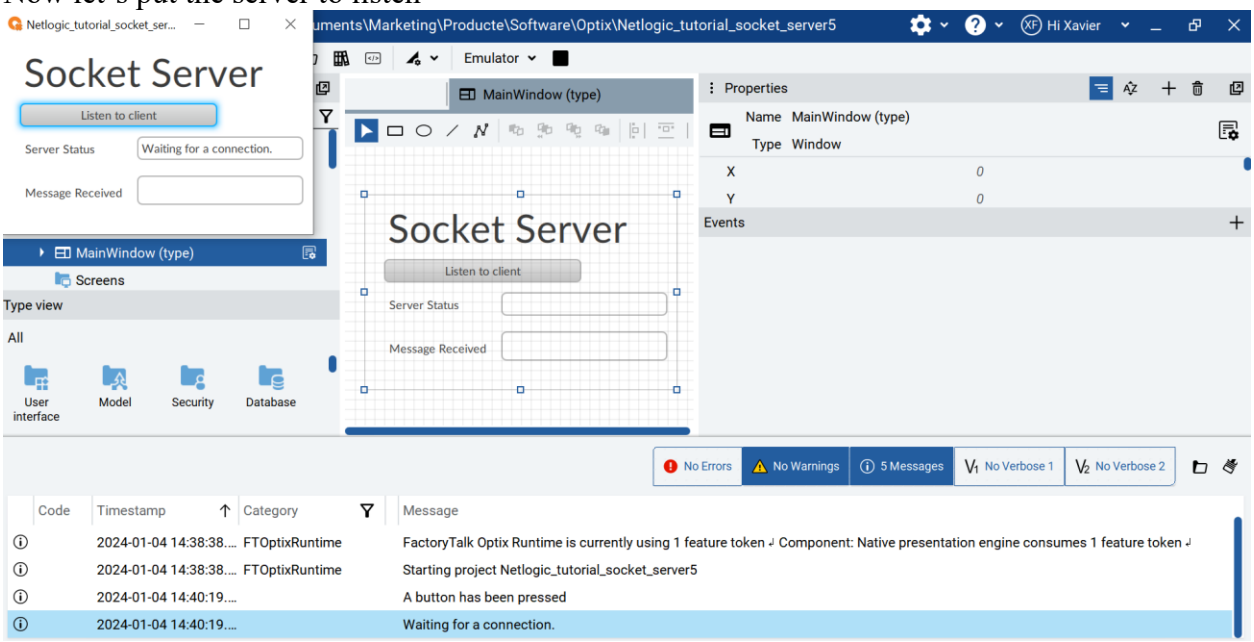
Second send, only one client is connected

Now let's try with our FactoryTalk Optix server

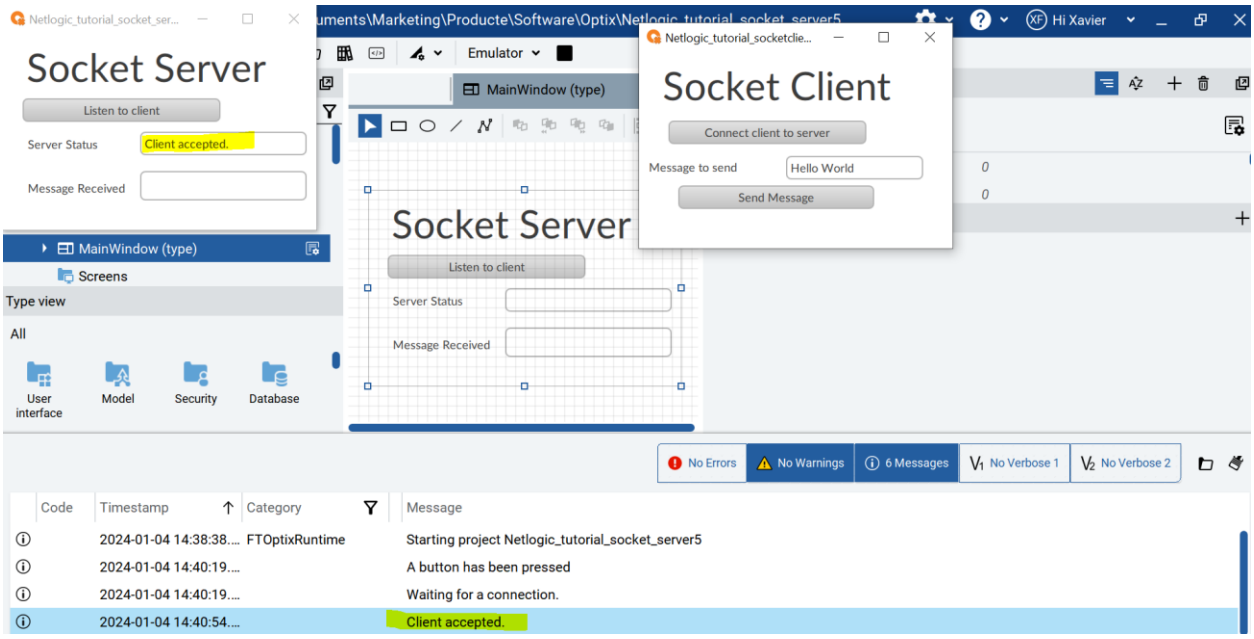
First we start only the server (since client makes connection just after starting, and if no server is listening then this will create an error exception)



Now let’s put the server to listen



Now let’s start the client



Now let's send the message

The socket is closed and there is no way to send more messages unless I stop and start the client

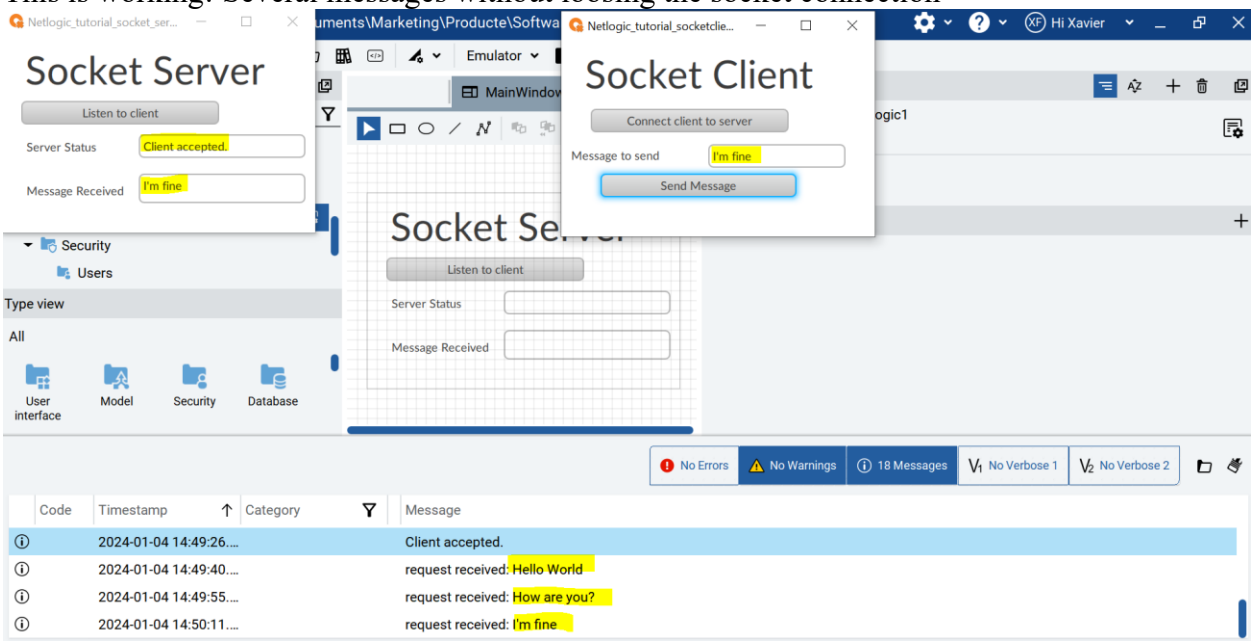
We should modify the server program in order not to close socket and wait for more connections.

We modify the server code this way

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId,NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
```

```
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        textbox.Text = "Waiting for a connection.";
        Log.Info("Waiting for a connection.");
        TcpClient client = listener.AcceptTcpClient();
        //Console.WriteLine("Client accepted.");
        textbox.Text = "Client accepted.";
        Log.Info("Client accepted.");

        while (true)
        {
            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());
            try
            {
                byte[] buffer = new byte[1024];
                stream.Read(buffer, 0, buffer.Length);
                int recv = 0;
                foreach (byte b in buffer)
                {
                    if (b!=0)
                    {
                        recv++;
                    }
                }
                string request = Encoding.UTF8.GetString(buffer, 0, recv);
                //Console.WriteLine("request received: "+ request);
                textbox2.Text = request;
                Log.Info("request received: "+ request);
                sw.WriteLine("You rock!");
                sw.Flush();
            }
            catch(Exception e)
            {
                //Console.WriteLine("Something went wrong.");
                Log.Info("Something went wrong.");
                sw.WriteLine(e.ToString());
            }
            System.Threading.Thread.Sleep(1000);
        }
    }
}
```

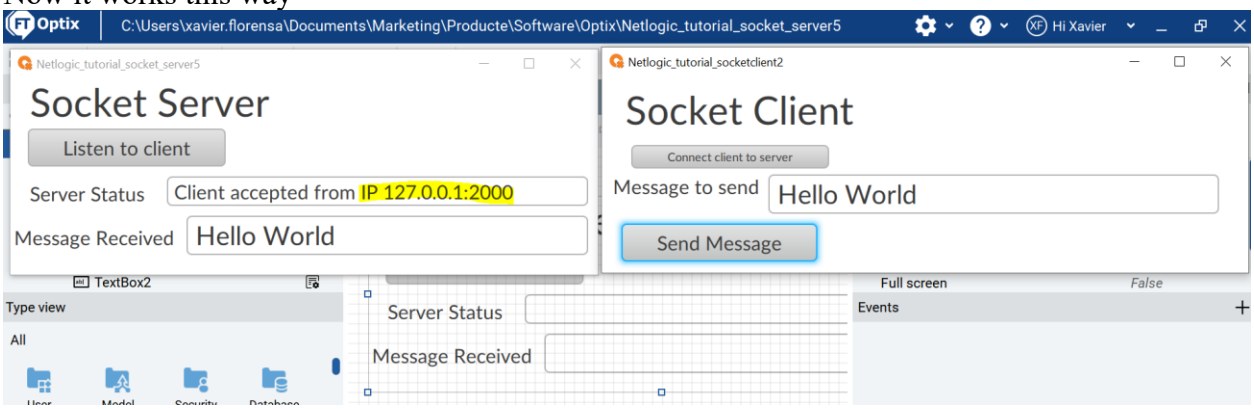
This is working! Several messages without loosing the socket connection



Still is the server difficult to stop (it hangs) so we still have to improve this.
Let's show the IP address and port from the connected client with this modification on the server address

```
[ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId,NodeId textboxNodeId2)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        textbox.Text = "Waiting for a connection.";
        Log.Info("Waiting for a connection.");
        TcpClient client = listener.AcceptTcpClient();
        //---------------------------------
        var clientIpLAN = client.Client.LocalEndPoint;
        Log.Info(clientIpLAN.ToString());
        //Console.WriteLine("Client accepted );
        textbox.Text = "Client accepted from IP " + clientIpLAN.ToString() ;
        //---------------------------------
        Log.Info("Client accepted.");

        while (true)
```

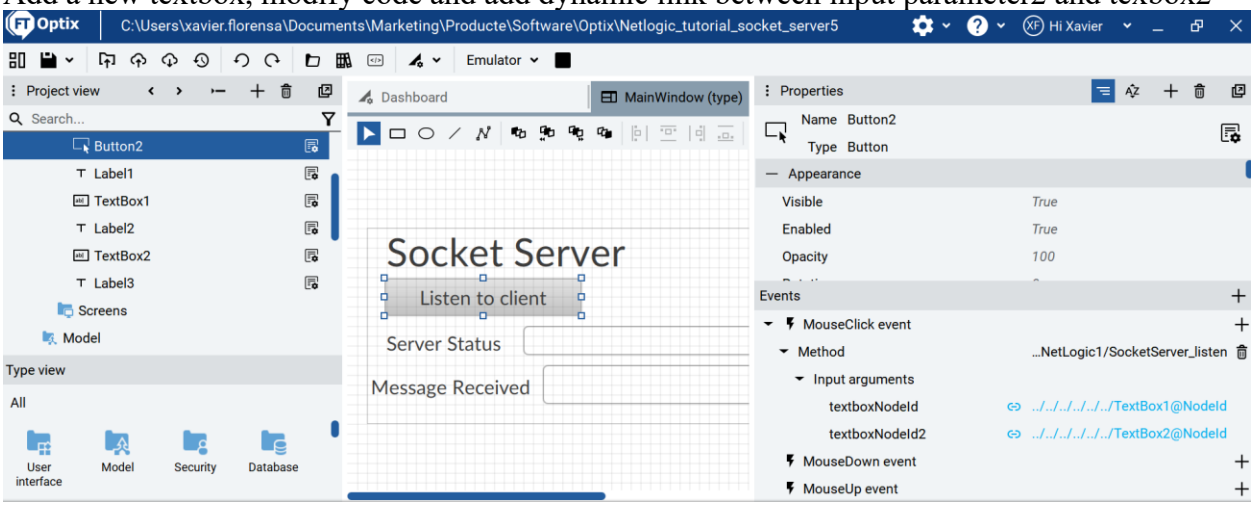
Now it works this way

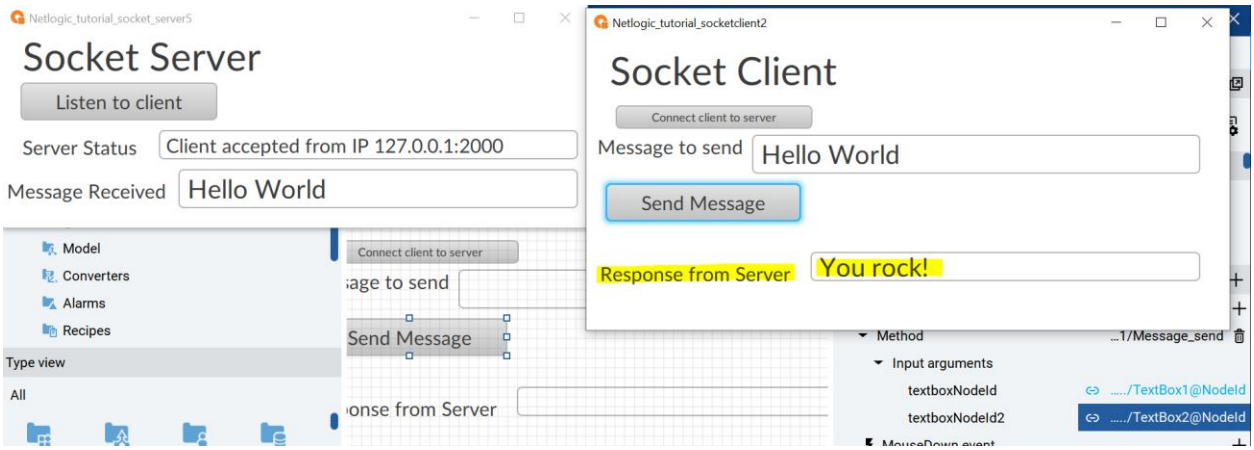


Now seems that if you close the client first and then you close the server, it stops inmediatelly. Even with an error. But this is OK since the socket is broken after you stop the client application. Next step will be to give a message on the client side that it is connected to…, and to make the connection after you hit a button.

To view responses from server on client side we have to modify the client code this way, and keep server as it was (server was sending message: "You rock!"

Add a new textbox, modify code and add dynamic link between input parameter2 and texbox2



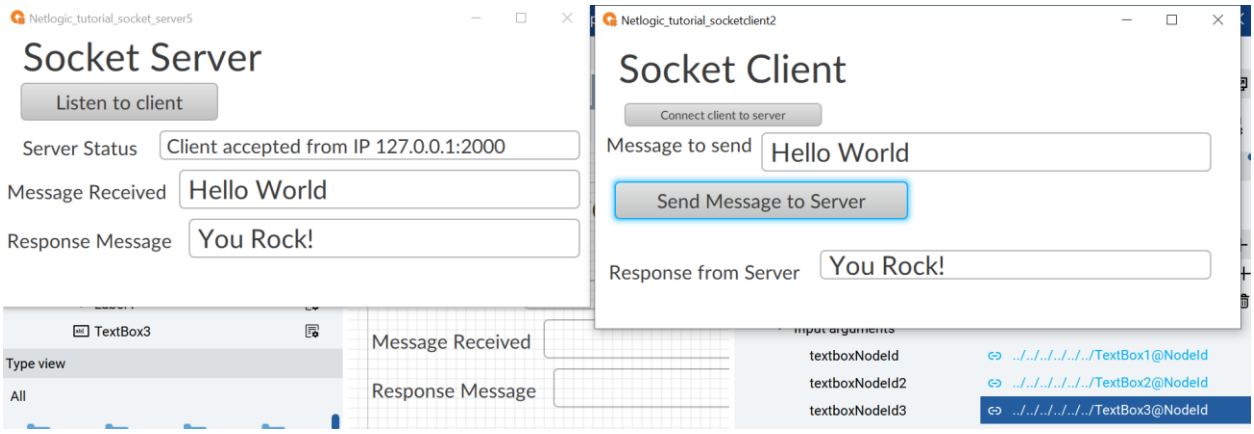


Client code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    TcpClient client = new TcpClient("127.0.0.1", 2000);

    [ExportMethod]

    public void Message_send(NodeId textboxNodeId,NodeId textboxNodeId2 )
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        Log.Info("A button has been pressed");
        textbox.FontSize = 40;
        var missatge = textbox.Text;
        string messageToSend = missatge;
        int byteCount = Encoding.ASCII.GetByteCount(messageToSend + 1);
        byte[] sendData = Encoding.ASCII.GetBytes(messageToSend);
        NetworkStream stream = client.GetStream();
        stream.Write(sendData, 0, sendData.Length);
        //Console.WriteLine("sending data to server...");
        Log.Info("sending data to server...");
        StreamReader sr = new StreamReader(stream);
        string response = sr.ReadLine();
        //Console.WriteLine(response);
        Log.Info(response);
        textbox2.FontSize = 40;
        textbox2.Text = response;
    }
```

```
}
```

Now we will add a textbox to send a response message to the client
On the server



Modify the server code this way

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.Core;
using FTOptix.CoreBase;
using FTOptix.NetLogic;
using System.Net.Sockets;
using System.Net;
using System.Threading;
using System.IO.Pipes;
using System.IO;
using System.Text;
using System.Diagnostics;
#endregion
public class RuntimeNetLogic1 : FTOptix.NetLogic.BaseNetLogic
{
    [ExportMethod]
    public void SocketServer_listen(NodeId textboxNodeId,NodeId
textboxNodeId2,NodeId textboxNodeId3)
    {
        var textbox = InformationModel.Get<TextBox>(textboxNodeId);
        var textbox2 = InformationModel.Get<TextBox>(textboxNodeId2);
        var textbox3 = InformationModel.Get<TextBox>(textboxNodeId3);
        textbox3.FontSize =40;
```

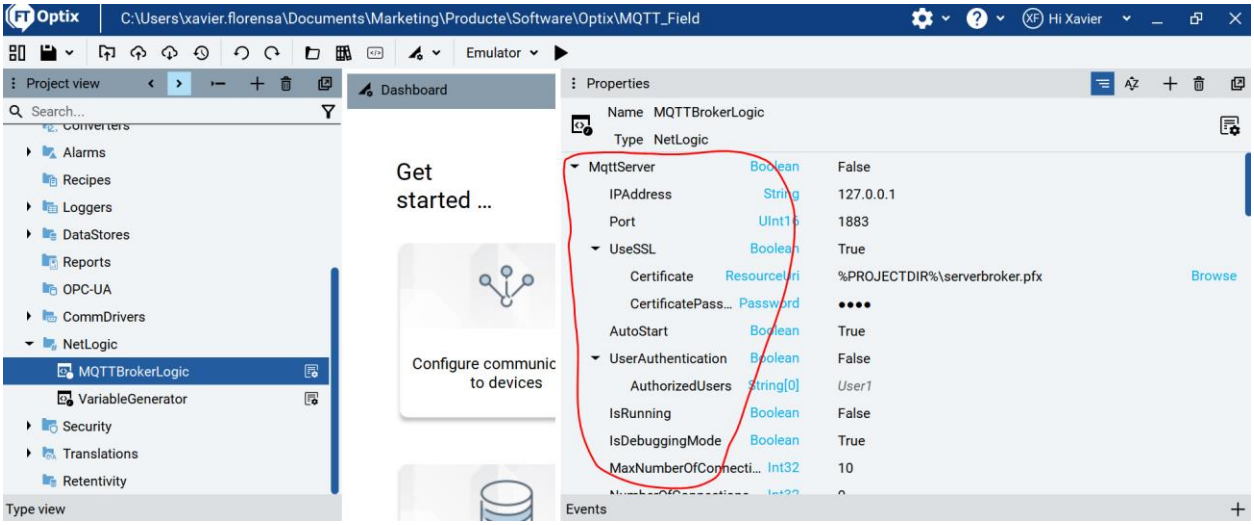
```
        textbox.Text = "A button has been pressed";
        Log.Info("A button has been pressed");
        TcpListener listener = new TcpListener(System.Net.IPAddress.Any, 2000);
        listener.Start();
        textbox.Text = "Waiting for a connection.";
        Log.Info("Waiting for a connection.");
        TcpClient client = listener.AcceptTcpClient();
        //----------------------------------
        var clientIpLAN = client.Client.LocalEndPoint;
        Log.Info(clientIpLAN.ToString());
        //Console.WriteLine("Client accepted );
        textbox.Text = "Client accepted from IP " + clientIpLAN.ToString() ;
        //----------------------------------
        Log.Info("Client accepted.");

        while (true)
        {
            NetworkStream stream = client.GetStream();
            StreamReader sr = new StreamReader(client.GetStream());
            StreamWriter sw = new StreamWriter(client.GetStream());
            try
            {
                byte[] buffer = new byte[1024];
                stream.Read(buffer, 0, buffer.Length);
                int recv = 0;
                foreach (byte b in buffer)
                {
                    if (b!=0)
                    {
                        recv++;
                    }
                }
                string request = Encoding.UTF8.GetString(buffer, 0, recv);
                //Console.WriteLine("request received: "+ request);
                textbox2.FontSize =40;
                textbox2.Text = request;
                Log.Info("request received: "+ request);
                //sw.WriteLine("You rock!");
                sw.WriteLine(textbox3.Text);
                sw.Flush();
            }
            catch(Exception e)
            {
                //Console.WriteLine("Something went wrong.");
                Log.Info("Something went wrong.");
```

```
                sw.WriteLine(e.ToString());
            }
            System.Threading.Thread.Sleep(1000);
        }
    }
}
```

Next step is to show a message on Client to tell that it is connected.
This can be made locally on the client, or as a response from the Server.
Let's do it as a local connection status.
Next Step will be selecting the target IP addresses of the client.

45. How to define NetLogic Parameters

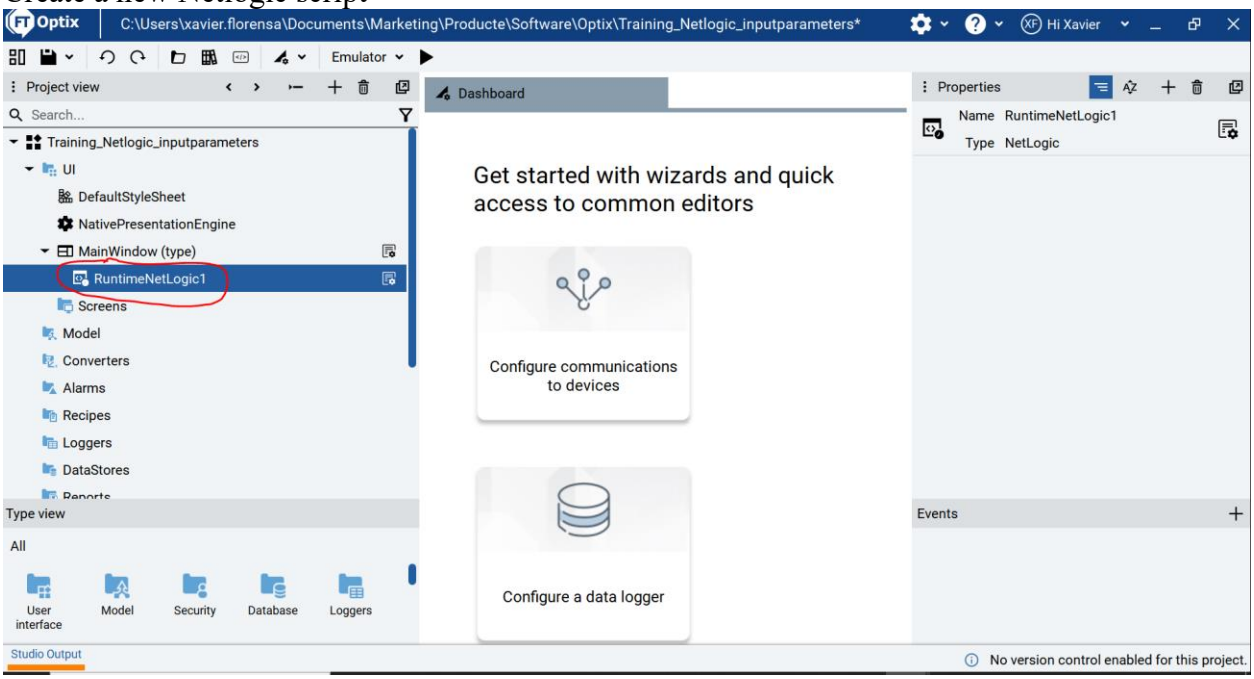
This is useful when we have a NetLogic script that needs some user customization and we do not want the user to enter in C# editor.
Take a look at some examples from Rockwell repository.
In some of them, when you click on Netlogic, you will see some parameters
For instance



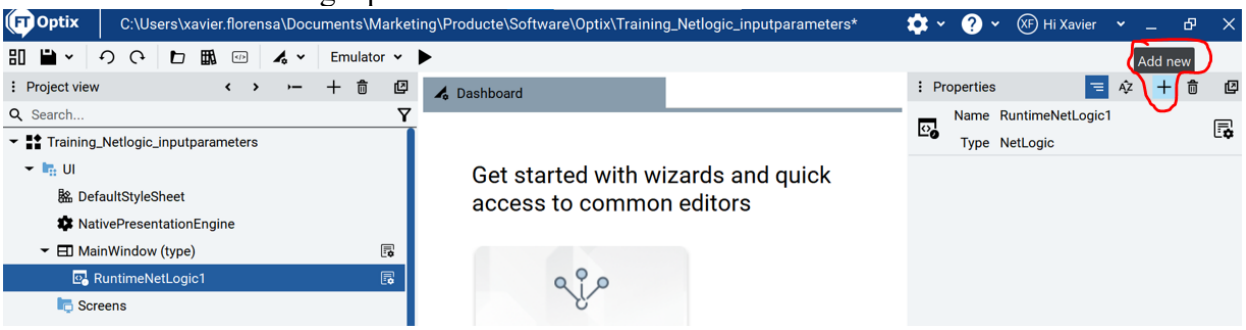
We see that it corresponds to a structure

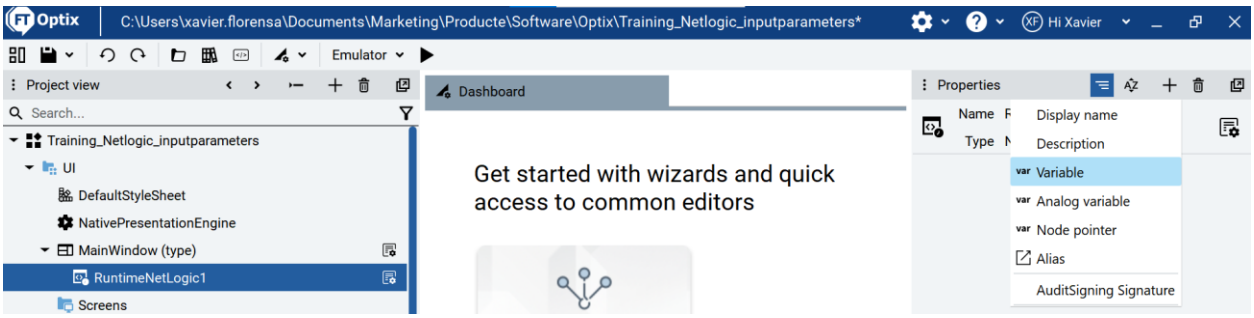
```
2416    private struct MQTTServerParameters
2417    {
            6 references
2418        public IPAddress ipAddress;
            4 references
2419        public ushort port;
            4 references
```

But let's see this with a simple example
Let's create a new project
Create a new Netlogic script

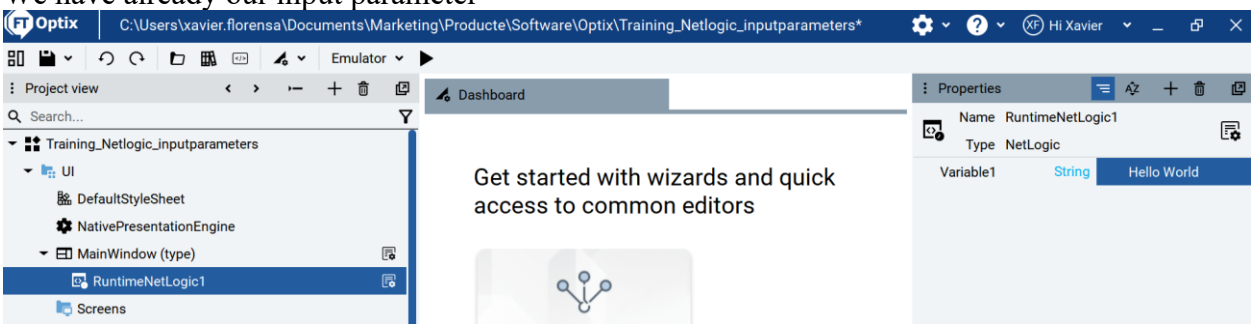


Let's create a New Netlogic parameter



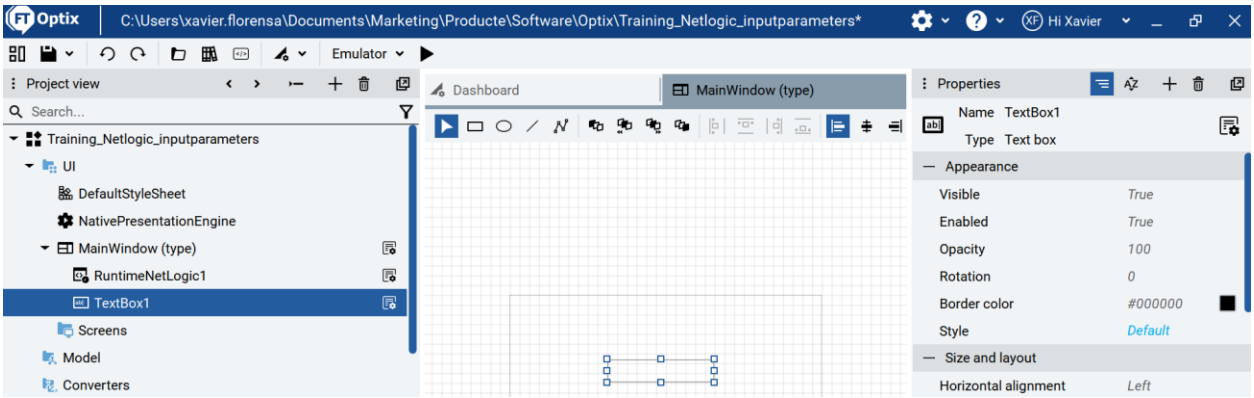


We have already our input parameter



Now let's go to access this value from C# and display display the value on a Textbox

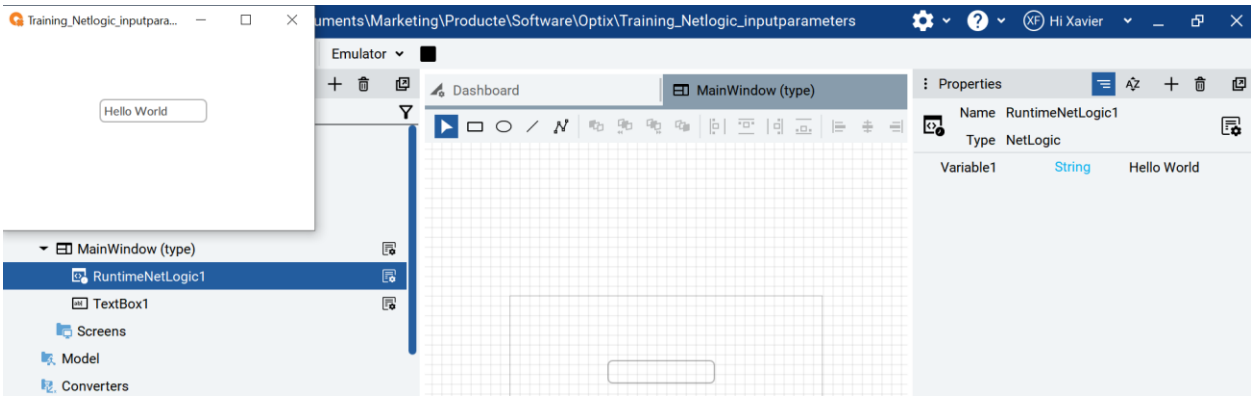
Create a new Textbox



Copy and paste this code on the Netlogic script

```
public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        var textbox1 = Owner.Get<TextBox>("TextBox1");
        var parameter = LogicObject.GetVariable("Variable1");
        textbox1.Text=parameter.Value;
    }
```

Test the application

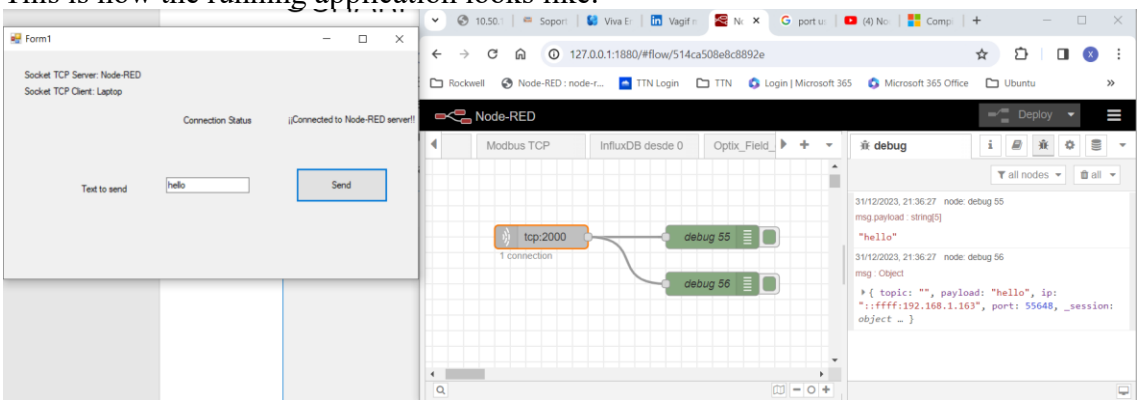


46. Using third party dll libraries in FactoryTalk Optix

You can see the result here
https://youtu.be/6E4xt02Ohzw
You can use third party libraries.
For instance Winsock_Orcas.dll
Even thought this library was created for Visual Basic, since it is compiled, it can be also used with C#
https://www.codeproject.com/Articles/21443/Winsock-Revamped
This DLL is tested in Visual Studio 2022. With this code in Visual Studio 2022

46.1. Using third party libraries in Visual Studio 2022 windows Forms Framework

On this chapter we will not use FactoryTalk Optix. This is just to demonstrate how to reuse other libraries that are already working in Visual Studio.
TCP/IP PC to Node-RED communications using this routine written in C# (C sharp) and the socket dll library called Winsock orcas
This is how the running application looks like:



About C# programming language
https://es.wikipedia.org/wiki/C_Sharp
About winsock
https://en.wikipedia.org/wiki/Winsock
About Winsock orcas dll library by Chris Kolkman

"Winsock Orcas is the result of refining a project that started when I graduated from VB6 to VB.NET 2003.
.NET no longer supported the old Winsock control that had been so easy to use in VB6. Instead they gave us something with much more power, but also much more complexity: the Socket.
It took me a bit of time to figure the socket out, but when I did I decided to create a wrapper that worked just like the old control I was familiar with - making sockets much easier.
The first version was wrought with bugs and wasn't thread-safe. When VS 2005 came out, and revealed even more functions with regards to the socket - I resolved to make a new version.
That was Winsock 2005. It was thread-safe (to a point), and fixed the major bugs of the previous version. It even had UDP support.
In April of 2007 I started work on Winsock 2007. Due to a project I was working on at the time, I was looking into Remoting to synchronize an object between server/client. I decided Remoting wasn't for my project (couldn't implement blacklist), thus a new version of Winsock was born.
Winsock 2007 enabled synchronizing of objects (via BinaryFormatter), making the Send/Get routines simpler. The thread-safe events have been improved, as has UDP support. IPv6 support was added making it much easier to use with Vista.
Winsock Orcas (version 4.0.0) was made just to keep this going. It had come to my attention that VS2008 had problems compiling the code for previous version, so I made this version. This version streamlines the code, making it simpler to read (mainly by removing the WinsockMonitor class), and also adds in some Generics support on the Get/Peek methods to do automatic conversion to the type you want (watch out, you could cause exceptions for casting to the wrong type).
"

Why to use this dll?:
to create your application easily thus forgetting about communications layer, letting the Visual Studio environment write the code for you.

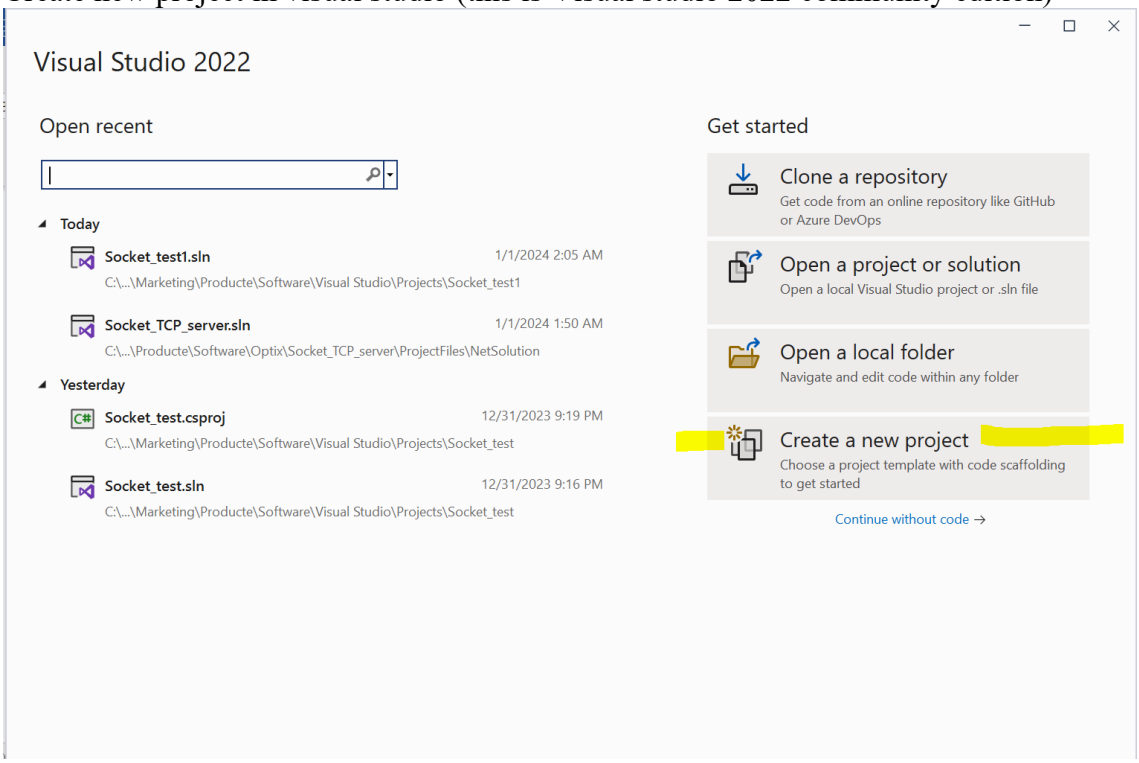
Customer normally uses Microsoft Visual Studio environment to edit, compile and run the application.
https://docs.microsoft.com/en-us/visualstudio/#pivot=features

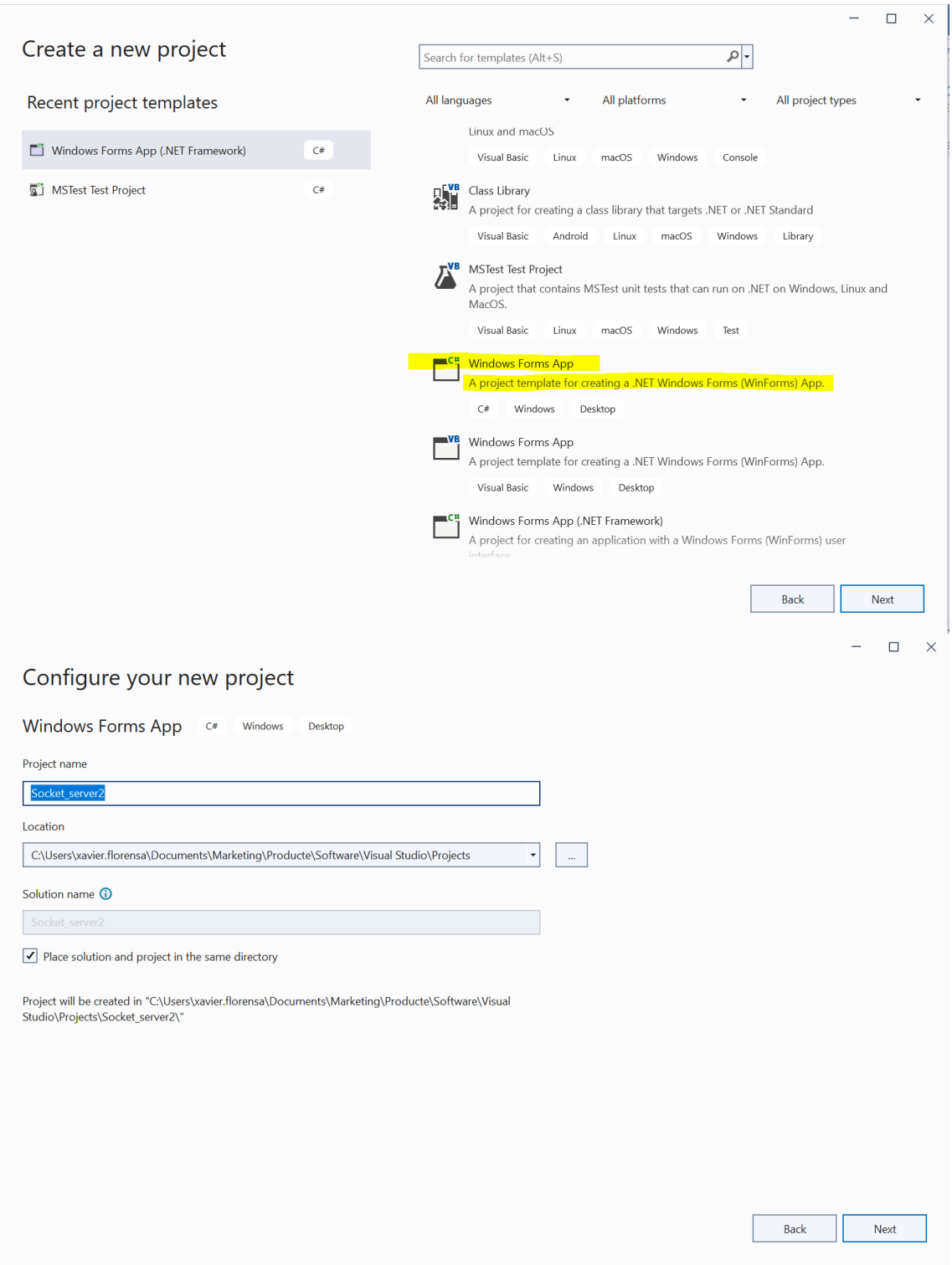
This is the code:
But do not copy and paste, you have to put the controls on the form and then edit little changes. See below.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Socket_test1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //winsock_Ear.Listen(2000);//This is to make the PC act as host
            winsock_Ear.Connect("192.168.1.163", 2000); //This is to make the PC act
as client

        }

        private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
        {
```

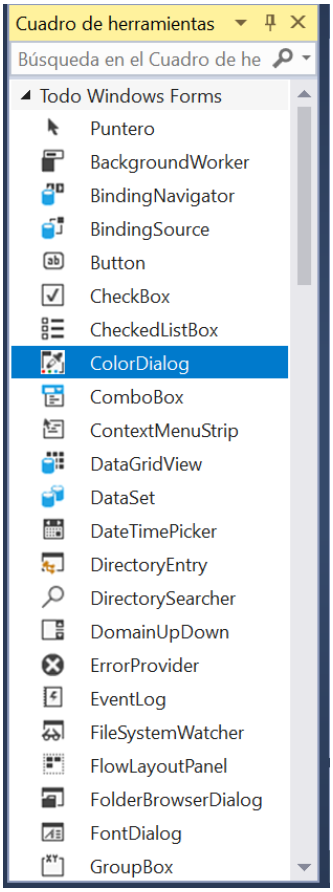
```
            label1.Text = "¡¡Connected to Node-RED server!!";
        }

        private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsockConnectionRequestEventArgs e)
        {
            winsock_Ear.Close();
            winsock_Ear.Accept(e.Client);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            //we get the text from the Textbox entered from the keyboard
            string text_to_send = this.textBox1.Text;
            winsock_Ear.Send(text_to_send);
        }
    }
}
```

How to proceed if you want to use this code for your own applications:
Copy the Winsock Orcas dll on a known location on your hard disk
Open Visual studio
Create new project in visual studio (this is Visual studio 2022 community edition)

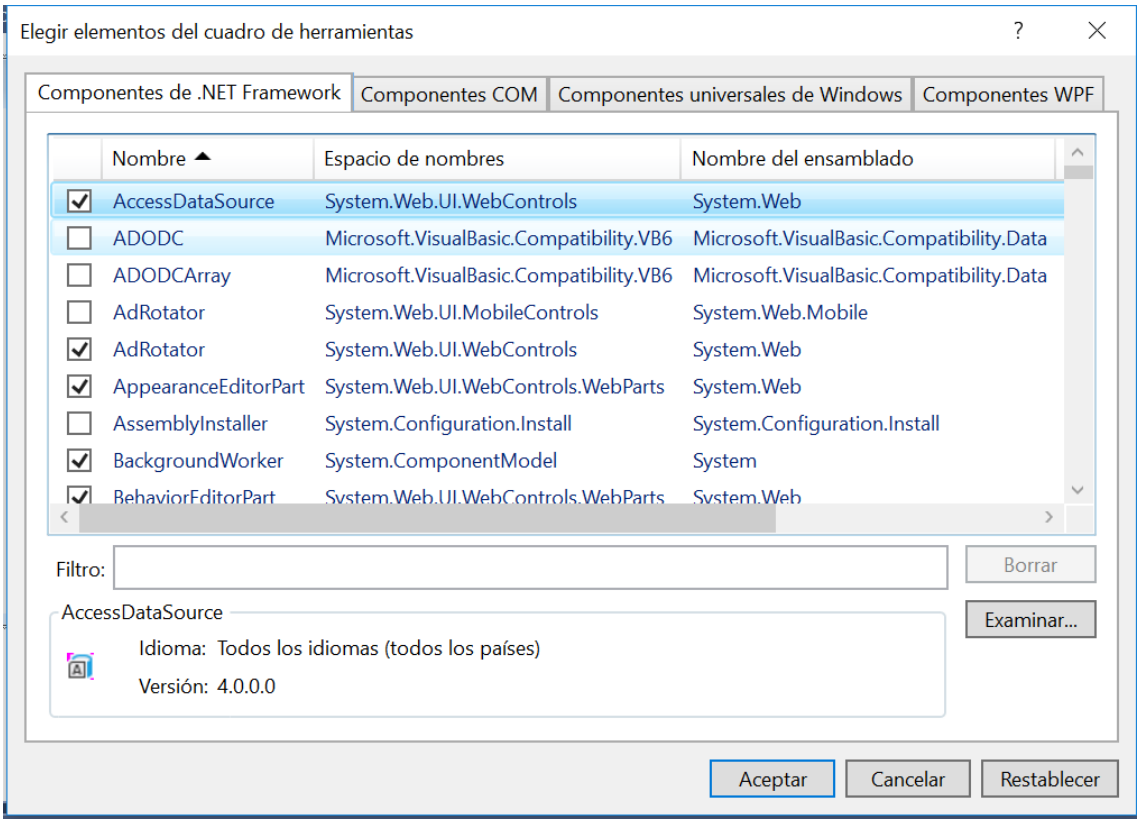




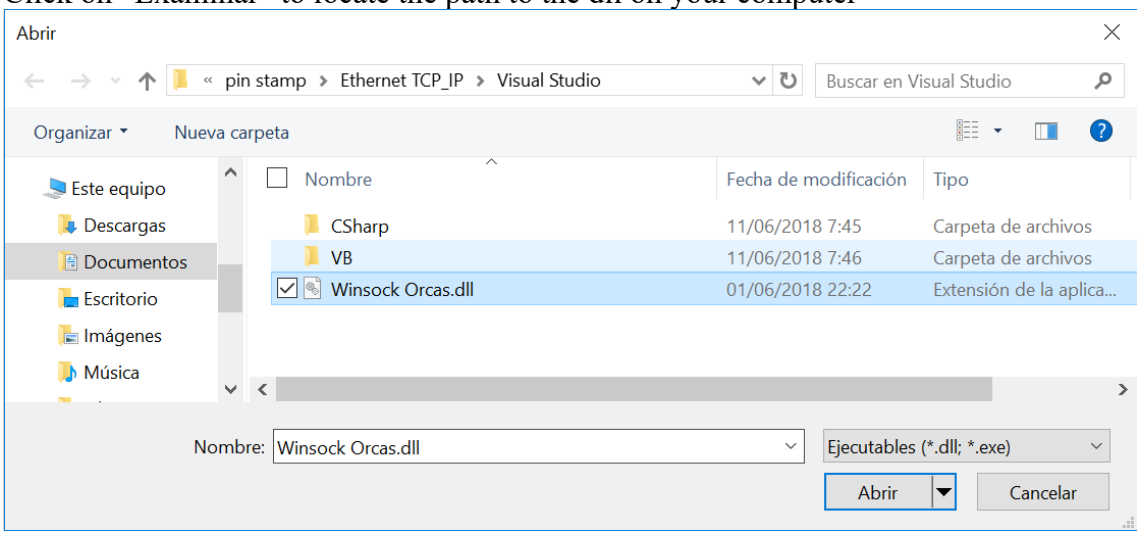
Once created,
then right click the mouse anywhere on your toolbox list to add the Winsock Orcas control for easy programming



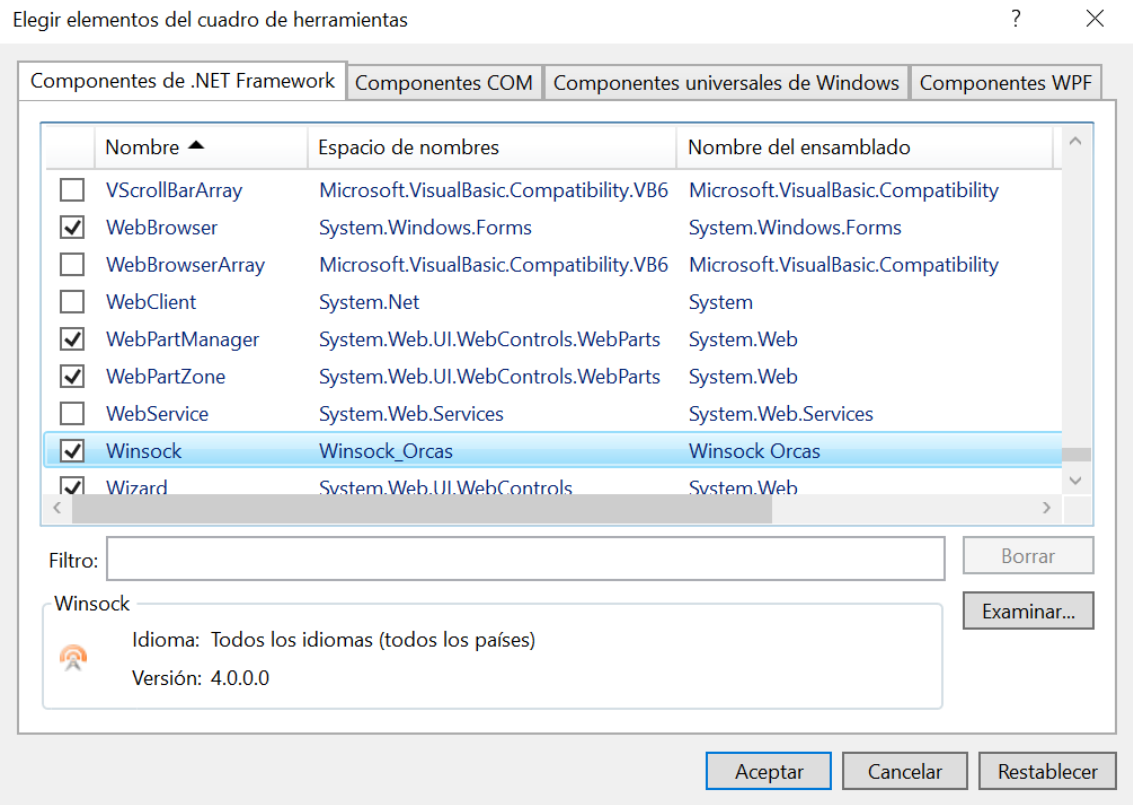
Then click on “Choose elements”



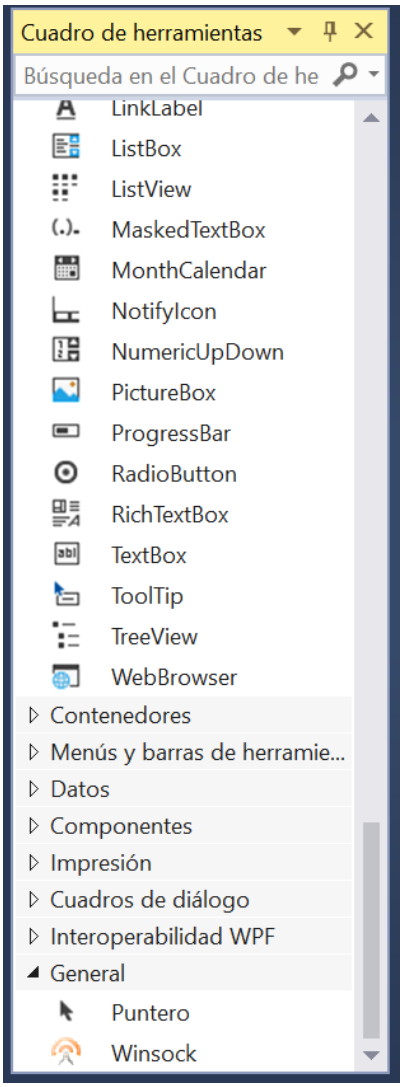
Click on “Examinar” to locate the path to the dll on your computer



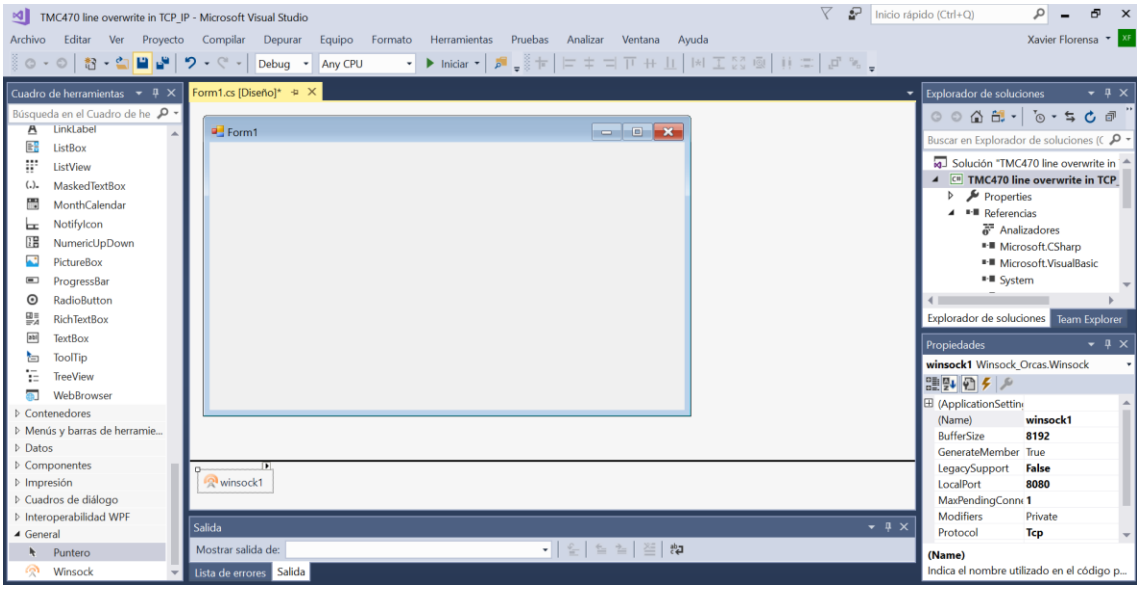
Open it and accept



You will find it now at the end of your toolbox list

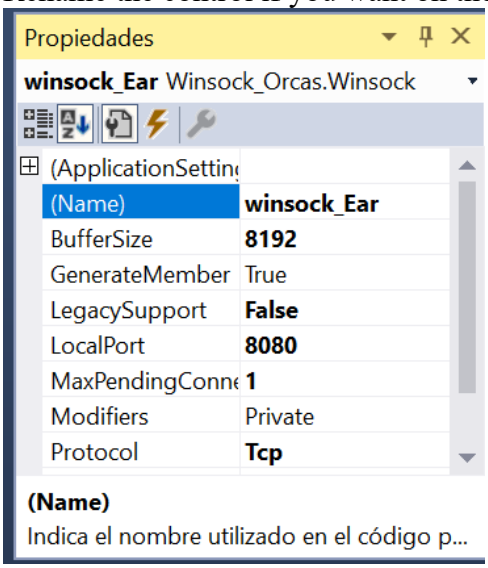


Click on the Winsock control icon and drag until your form
Then you will see it inserted on your form (just below)



Click on the arrow over the Winsock Orcas control on the form. Be sure to select the option enable Legagy (If not it will not be able to write data)

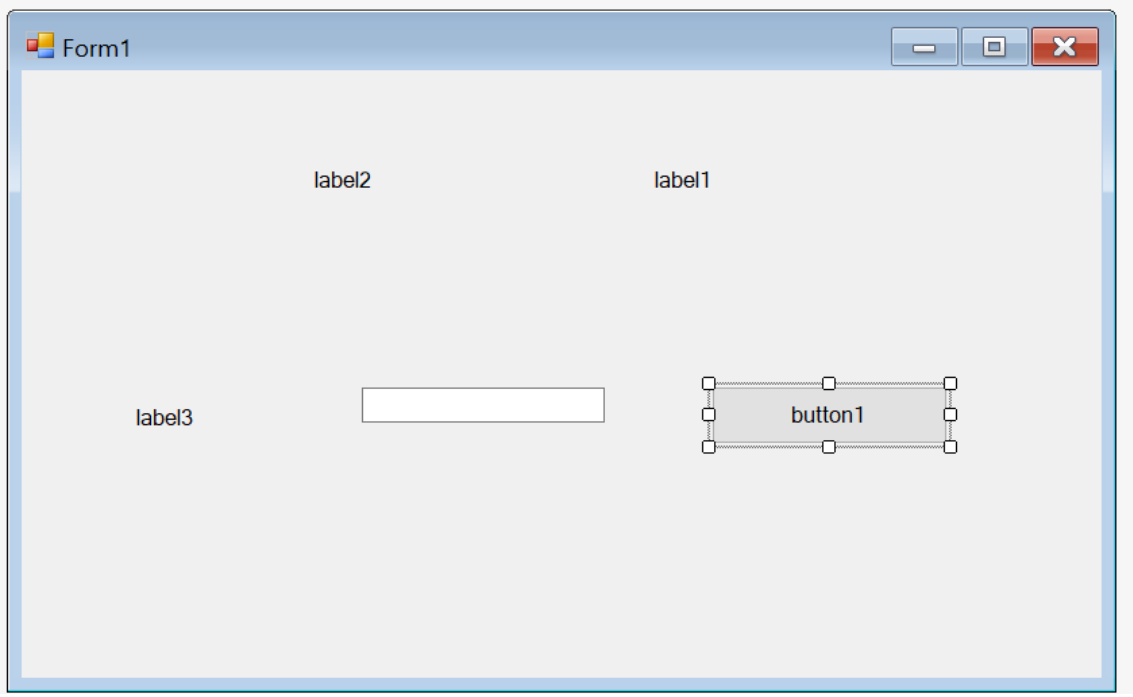
Rename the control if you want on the Properties (having selected the control)



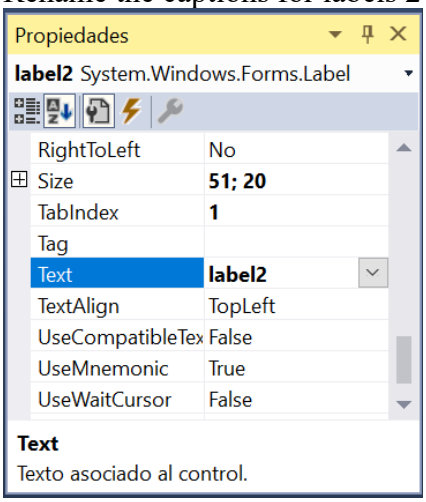
Double click on your form to see the generated code

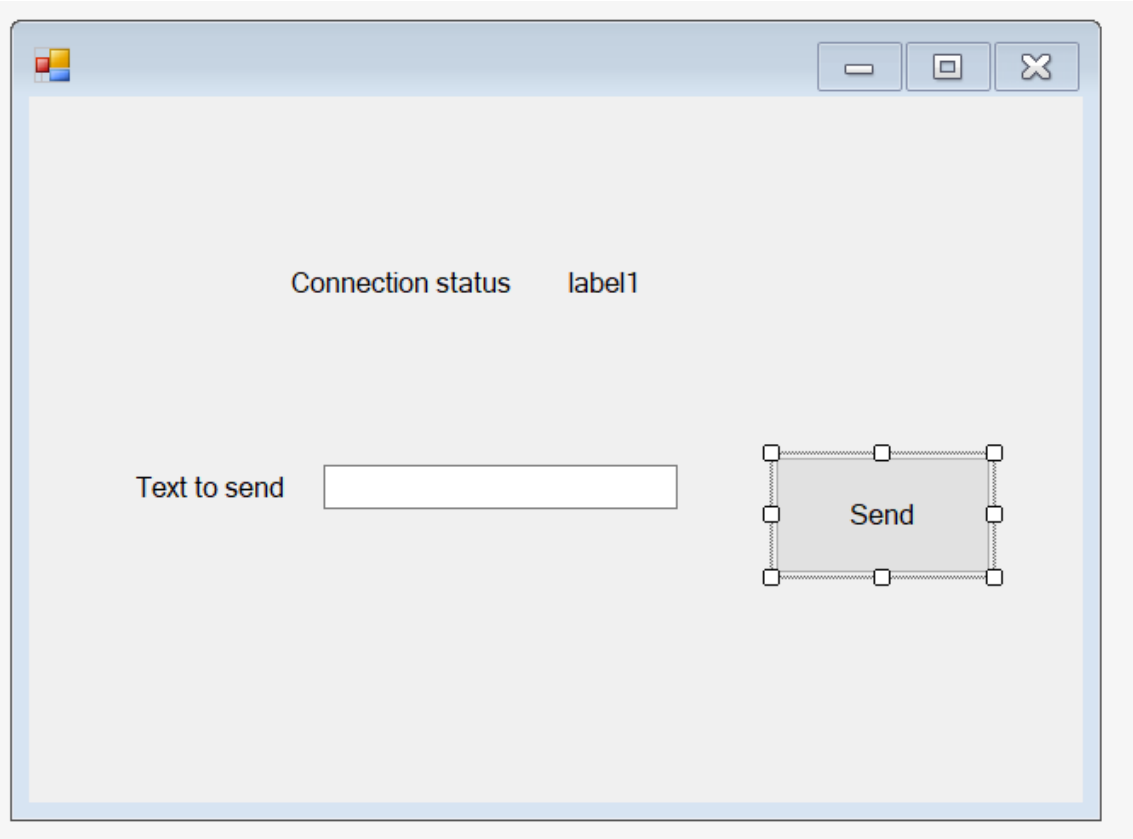
```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10
11  namespace TMC470_line_overwrite_in_TCP_IP
12  {
13      public partial class Form1 : Form
14      {
15          public Form1()
16          {
17              InitializeComponent();
18          }
19
20          private void Form1_Load(object sender, EventArgs e)
21          {
22              |
23          }
24      }
25  }
26
```

We move now to the form view
Then we start populating the form with the desired controls (click on the controls list and drag on the form)
In this case three labels, one text box and one button



Rename the captions for labels 2 and 3 and button as you want.





Now we will introduce three subroutines to get the connection
Click on the winsok_Ear control on your form and select “Connected” from the methods list
This will add this code for you (without typing anything)

```
private void winsock_Ear_Connected(object sender, Winsock_Orcas.WinsockConnectedEventArgs
e)
        {
        }
```

Then put the message that will appear when the TMC470 is connected, like this

```
private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
{
    label1.Text = "¡¡Connected to Node-RED server!!";
}
```

Complete the default subroutine

```
        private void Form1_Load(object sender, EventArgs e)
        {

        }
```

In this way, by typing (or copy / paste) this content
Be aware that the Laptop has this IP address per DHCP: 192.168.1.163

```
private void Form1_Load(object sender, EventArgs e)
{
```

```
    //winsock_Ear.Listen(2000);//This is to make the PC act as host
    winsock_Ear.Connect("192.168.1.163", 2000); //This is to make the PC act as
client connecting to port 2000 of server

}
```

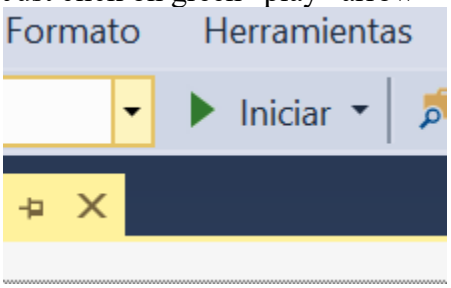
Then click on the Winsoc_Ear control on your form and select “ConnectionRequest”
This will add this code to your program without typing anything:

```
        private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionRequestEventArgs e)
        {

        }
```

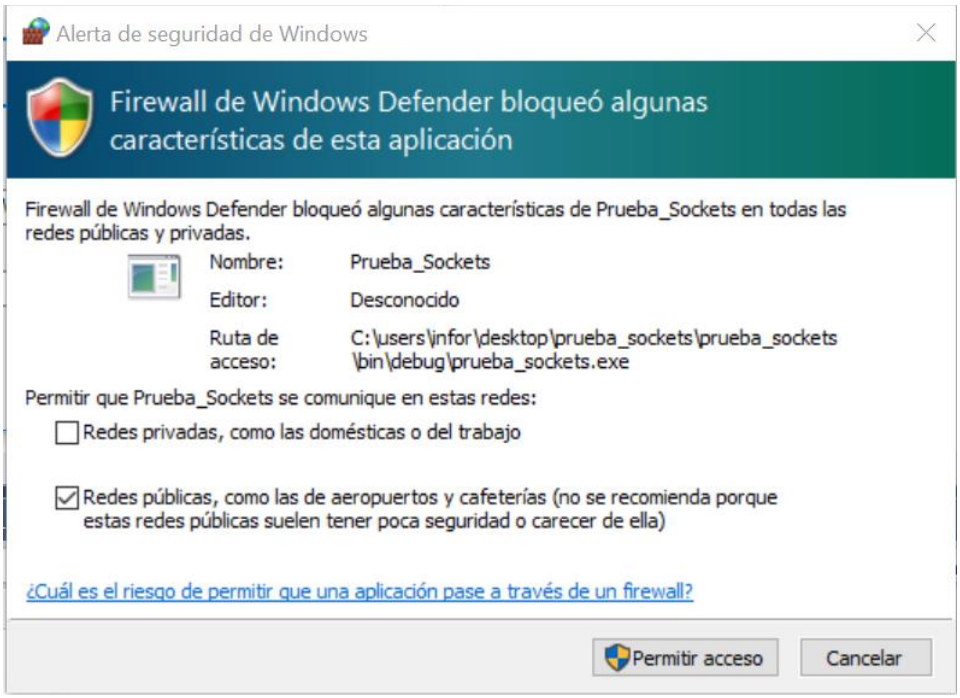
Then complete by typing this content

```
        private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionRequestEventArgs e)
        {
            winsock_Ear.Close();
            winsock_Ear.Accept(e.Client);
        }
```

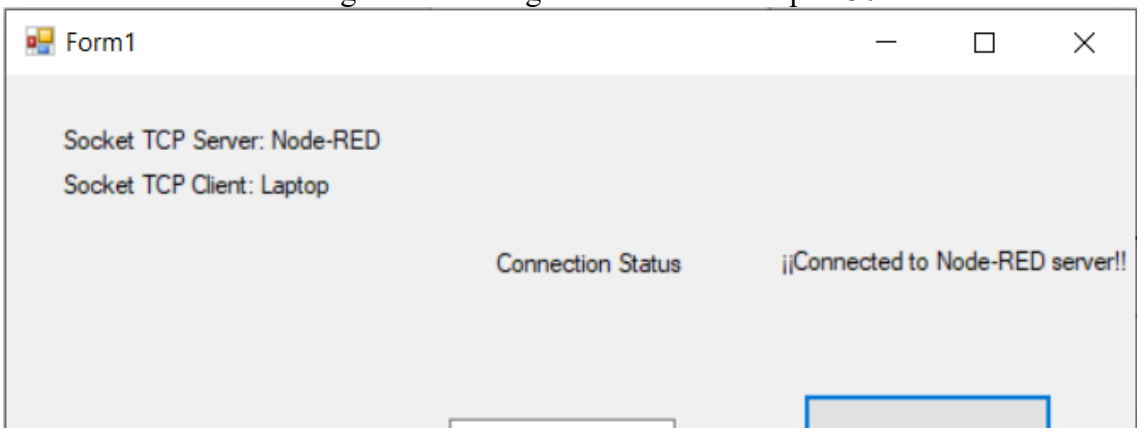
Now you can test the application to see if you get connection
Just click on green “play” arrow



First time you will see this message:



It's your firewall. Click on accept, otherwise you will not be able to test the application.
You should see this message after waiting between few and up to 30 seconds



Let's program the rest of the application in order to send data
Double click on the Button control on your form

This action will introduce this code without typing anything

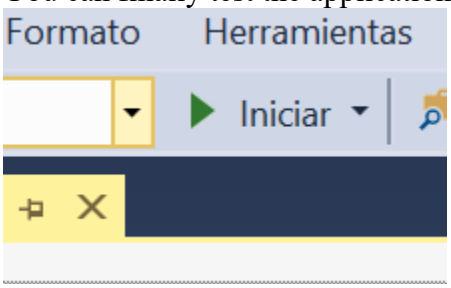
```
        private void button1_Click(object sender, EventArgs e)
        {

        }
```

Then type (or copy / paste) this content

```
        private void button1_Click(object sender, EventArgs e)
{
    //we get the text from the Textbox entered from the keyboard
    string text_to_send = this.textBox1.Text;
    winsock_Ear.Send(text_to_send);
```

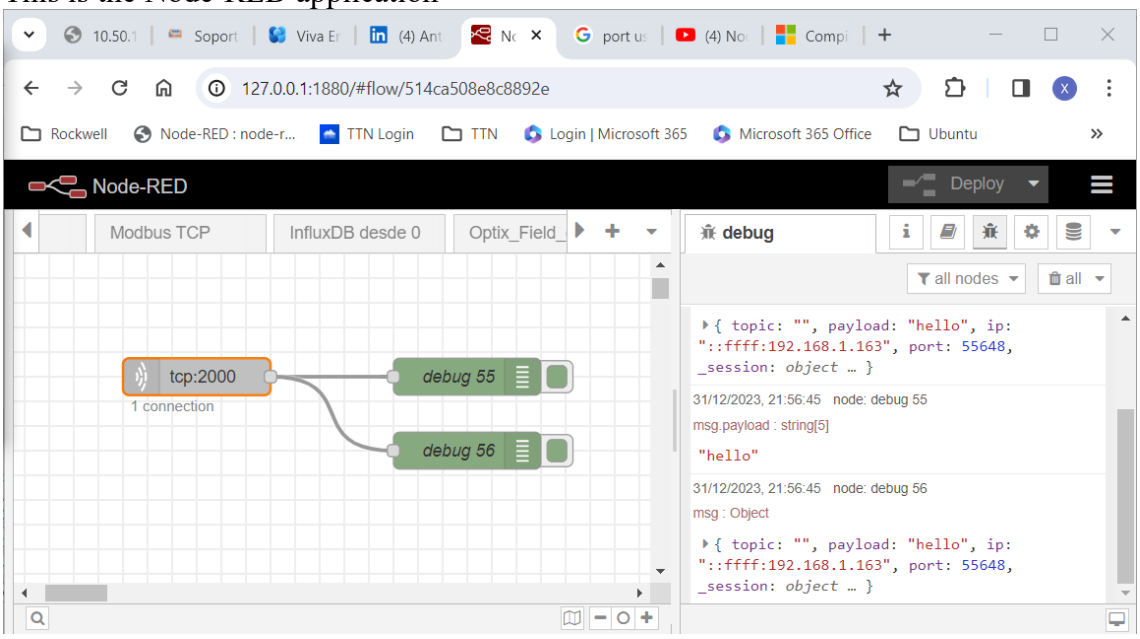
}

You can finally test the application



And that's all!

This is the Node-RED application



Edit tcp in node

Delete | Cancel | Done

Properties

Type: Listen on port 2000

☐ Enable secure (SSL/TLS) connection

Output: stream of String payload(s)

delimited by (optional)

☐ re-attach delimiter

Topic: Topic

Name: Name

Edit debug node

Delete | Cancel | Done

Properties

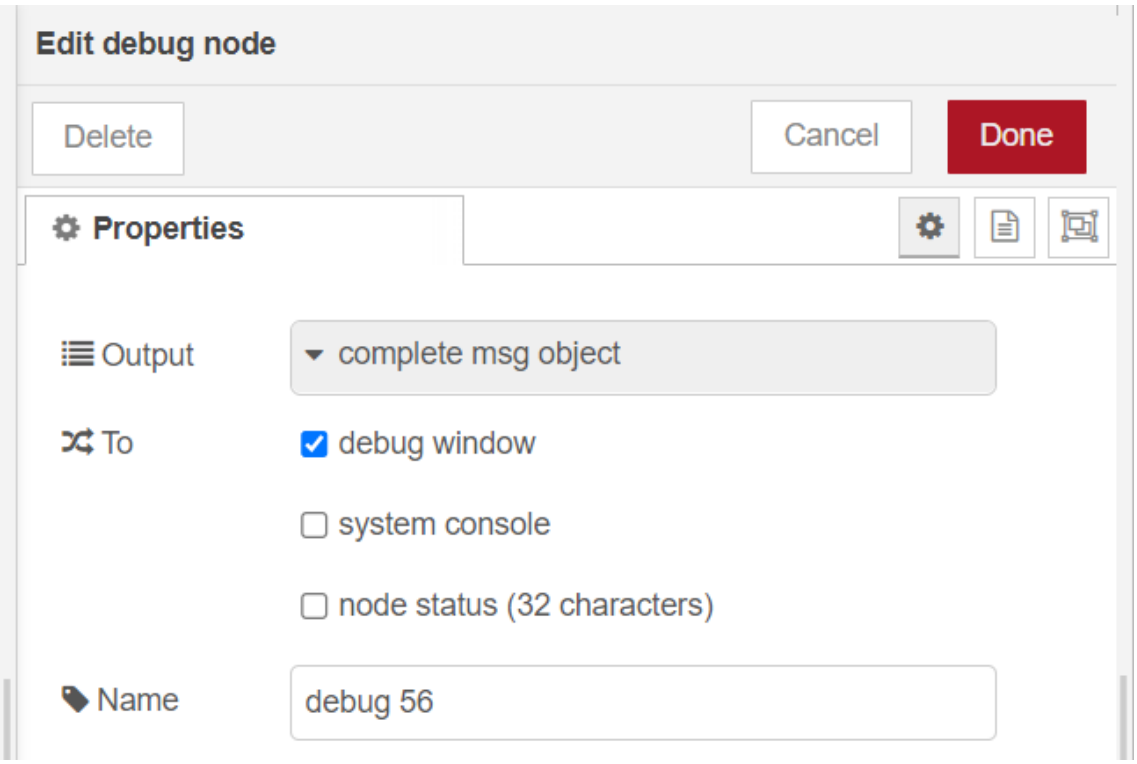
Output: msg. payload

To: ☑ debug window

☐ system console

☐ node status (32 characters)

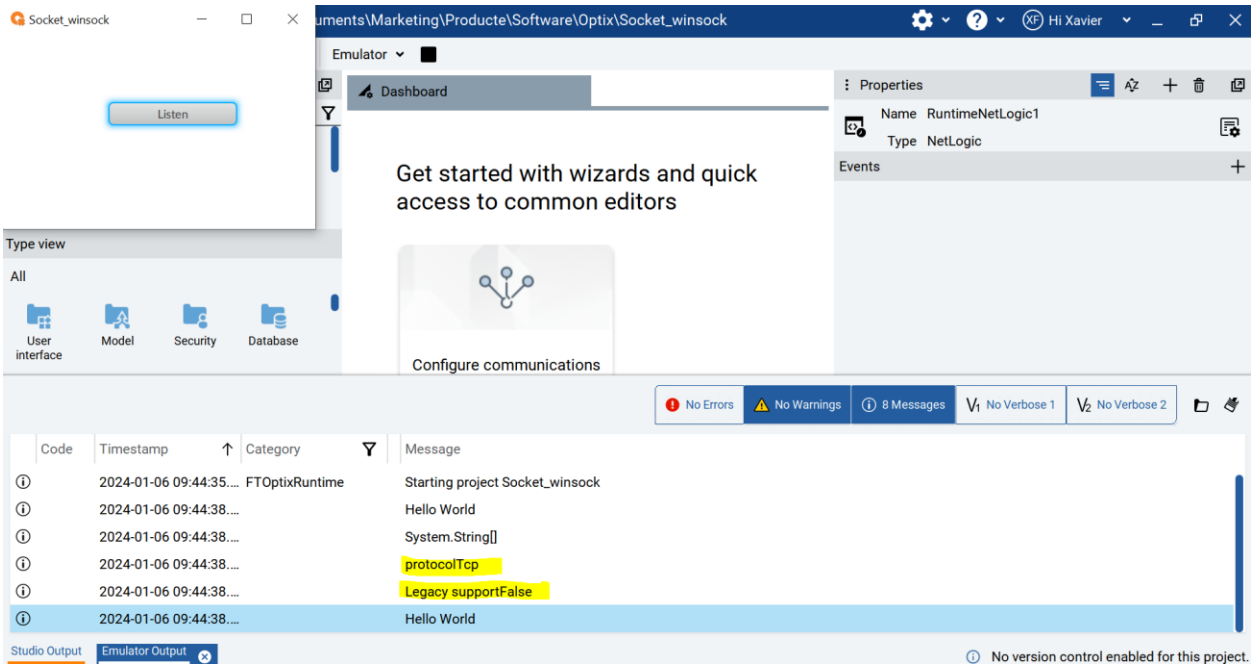
Name: debug 55



We have to learn how to use this library without Forms framework.
At least we know how to use in Forms frameworks, but FT Optix does not use Forms framework.

46.2. Thirt party libraries (dll) Socket Server application with Optix

You can see the result here
https://youtu.be/6E4xt02Ohzw
You can download the code from here
https://github.com/xavierflorensa/Optix_Winsock_Orcas_Server.git
We can see some data from class Winsock_Orcas
As you can see here, we have been able to see which one is the the Protocol used (TCP or UDP)



This is done with this code
At least we can have a dialog with the class to read, write data, when we hit the button.
Socket_winsock

```
[ExportMethod]

    public void Send()
    {
         Log.Info("Hello World");
         winsock_Ear.Listen(2000);//This is to make the PC act as host
                                  //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        Log.Info(winsock_Ear.LocalIP.ToString());
        Log.Info("protocol"+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support"+winsock_Ear.LegacySupport.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);

    }
```

This is the complete code

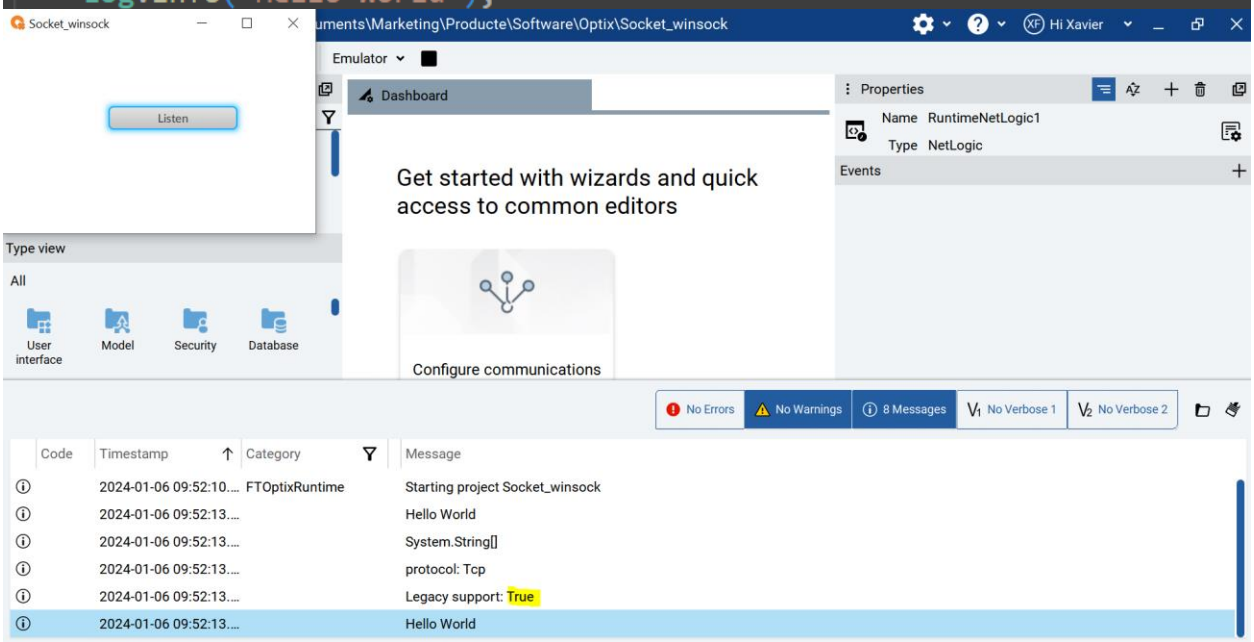
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
```

```
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{

    Winsock winsock_Ear = new Winsock();


    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
    {
        Log.Info("Connected to slave!!");

    }

    [ExportMethod]

    public void Send()
    {
         Log.Info("Hello World");
         winsock_Ear.Listen(2000);//This is to make the PC act as host
                                  //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        Log.Info(winsock_Ear.LocalIP.ToString());
        Log.Info("protocol"+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support"+winsock_Ear.LegacySupport.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);

    }
```

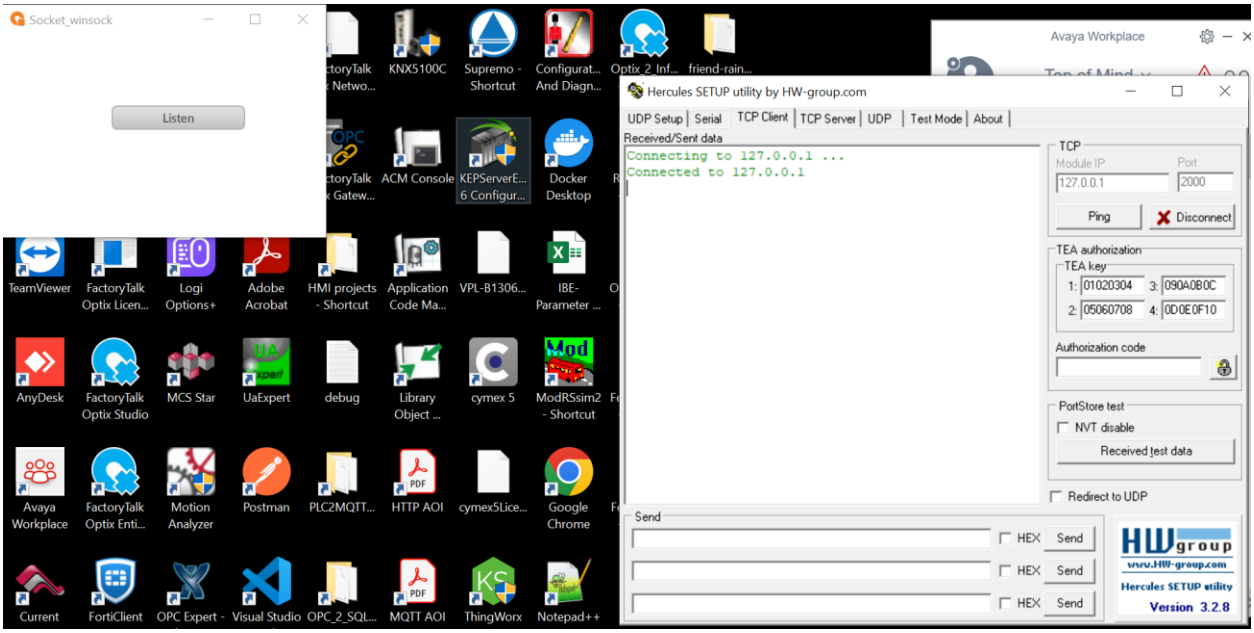
```
    public void Start(object sender, EventArgs e)
    //public override void Start(object sender, EventArgs e)
    {




       winsock_Ear.Listen(2000);//This is to make the PC act as host
                                //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client


        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    //public override void Stop()
    {
         Log.Info("Stopping");
        // Insert code to be executed when the user-defined logic is stopped
    }
    private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionRequestEventArgs e)
    {
        winsock_Ear.Close();
        winsock_Ear.Accept(e.Client);
        Log.Info("Connected to slave!!");

    }


    //private Winsock_Orcas.Winsock winsock_Ear;
    //public Winsock_Orcas.Winsock winsock_Ear;
}
```

If we change status of legacy support we see

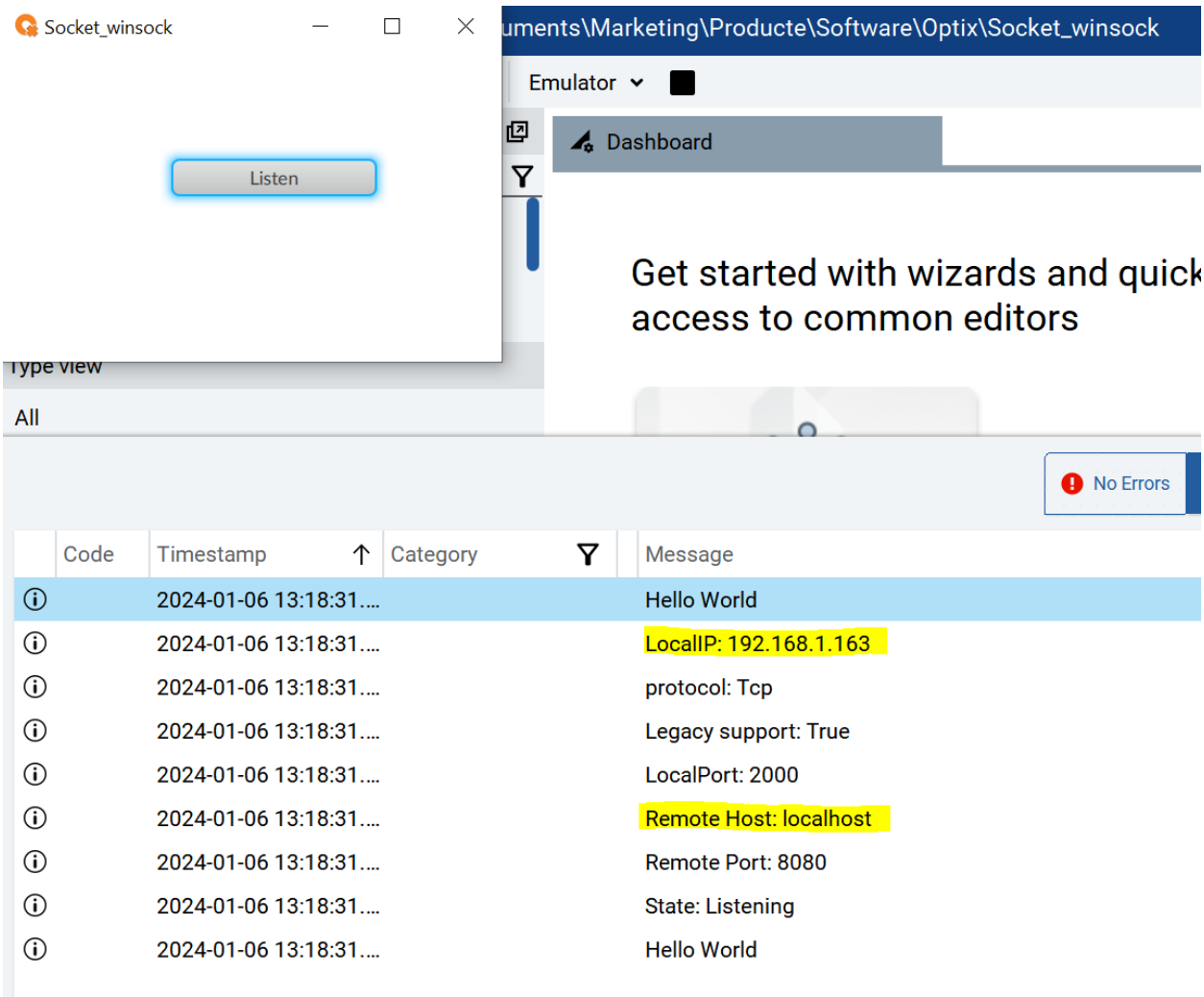
```
public void Send()
{
    Log.Info("Hello World");
    winsock_Ear.Listen(2000);//This is to make the PC act as host
                                //winsock_Ear.Connect("10.2.10.201", 2000);

    winsock_Ear.LegacySupport=true;
    Log.Info(winsock_Ear.LocalIP.ToString());
    Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
    Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
    Log.Info("Hello World");
```



We must check if the server is listening on port 2000.



We can even retrieve the IP from server host and from remote client



```
[ExportMethod]

    public void Send()
    {
         Log.Info("Hello World");
         winsock_Ear.Listen(2000);//This is to make the PC act as server host
                                  //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        winsock_Ear.LegacySupport=true;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);

    }
```

Let’s improve the code
Let’s try to put the code on the start program
This works accepting connections but not more else

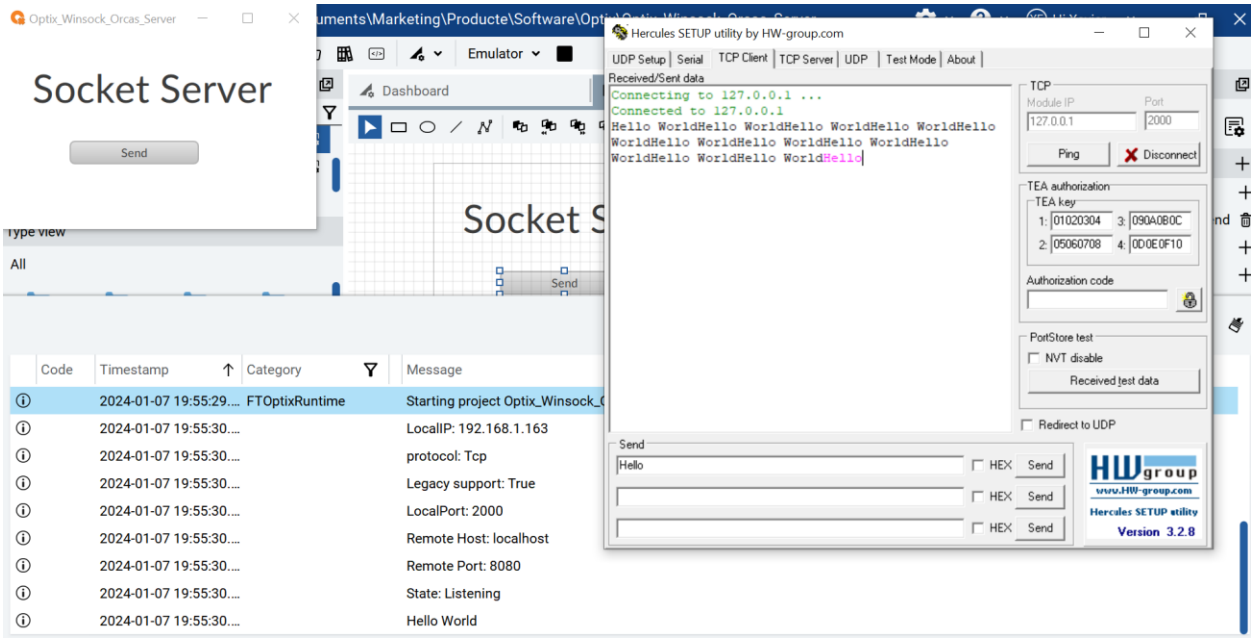
```
public class RuntimeNetLogic1 : BaseNetLogic
{

    //Winsock winsock_Ear = new Winsock();


    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
    {
        Log.Info("Connected to slave!!");

    }

    [ExportMethod]

    public void Send()
    {
        /*
```

```
         Log.Info("Hello World");
         winsock_Ear.Listen(2000);//This is to make the PC act as server host
                                  //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        winsock_Ear.LegacySupport=true;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        while (true)
        {
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
        System.Threading.Thread.Sleep(1000);
        }
        */
    }




    public override void Start()
    //public override void Start(object sender, EventArgs e)
    {
       Winsock winsock_Ear = new Winsock();
       winsock_Ear.LegacySupport=true;

       winsock_Ear.Listen(2000);//This is to make the PC act as host
                                  //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client

        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        /*
```

```
        while (true)
        {
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
        System.Threading.Thread.Sleep(1000);
        }
        */
    }

    public override void Stop()
    //public override void Stop()
    {
         Log.Info("Stopping");
        // Insert code to be executed when the user-defined logic is stopped
    }
    private void winsock_Ear_ConnectionRequest(object sender,
WinsockConnectionRequestEventArgs e)
    {
        //winsock_Ear.Close();
        //winsock_Ear.Accept(e.Client);
        Log.Info("Connected to slave!!");

    }


    //private Winsock_Orcas.Winsock winsock_Ear;
    //public Winsock_Orcas.Winsock winsock_Ear;
}
```

EventArgs e is a parameter called e that contains the event data, see the EventArgs MSDN page for more information.
Object Sender is a parameter called Sender that contains a reference to the control/object that raised the event.

You can download the code from here
https://github.com/xavierflorensa/Optix_Winsock_Orcas_Server.git
Let's deal with events,



Now we can send whatever messages we want from server
You have to change the application and add this sentence
To assign a callback that be executed after first connectionrequest

```
 public override void Start()
    {
       winsock_Ear.LegacySupport=true;

       winsock_Ear.Listen(2000);//This is to make the PC act as host
                                //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client
        // Assign a callback to be excuted when the client is connected
        //Winsock_Ear.Connected += winsock_Ear_Connected;
        // Assign a callback to be executed when a message is received from the
server
        winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
```

This is the complete code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
```

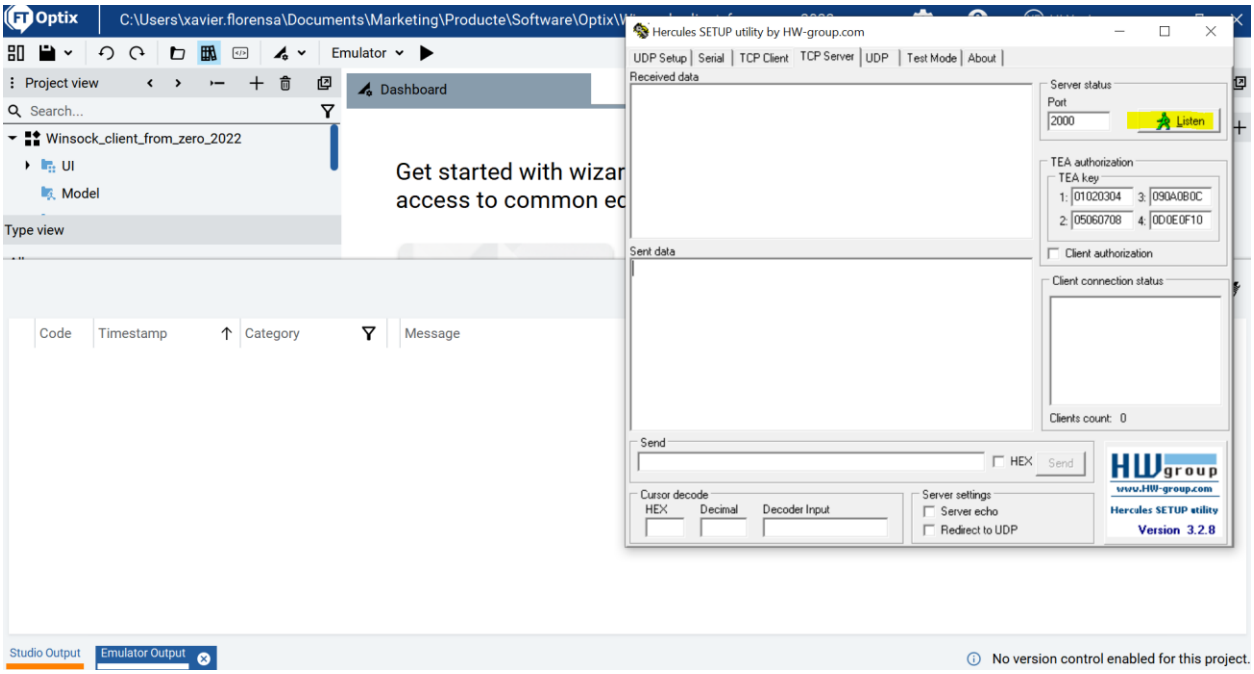
```
using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    Winsock winsock_Ear = new Winsock(); //opens a new socket

    [ExportMethod]

    public void Send()
    {
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Start()
    {
       winsock_Ear.LegacySupport=true;

       winsock_Ear.Listen(2000);//This is to make the PC act as host
                                //winsock_Ear.Connect("10.2.10.201", 2000);
//This is to make the PC act as client
        // Assign a callback to be excuted when the client is connected
        //Winsock_Ear.Connected += winsock_Ear_Connected;
        // Assign a callback to be executed when a message is received from the
server
        winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Stop()
    {
```

```
        Log.Info("Stopping");

    }

    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
    {
        Log.Info("Connected to slave!!");

    }
    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsockConnectionRequestEventArgs e)
        {
            winsock_Ear.Close();
            winsock_Ear.Accept(e.Client);
        }

    private void winsock_Ear_DataArrival(object sender,
Winsock_Orcas.WinsockDataArrivalEventArgs e)
        {
            string abRecibidos = winsock_Ear.Get<string>();
            Log.Info(abRecibidos);

        }
    private Winsock Winsock_Ear;
    //private Winsock_Orcas.Winsock winsock_Ear;
    //public Winsock_Orcas.Winsock winsock_Ear;
}
```

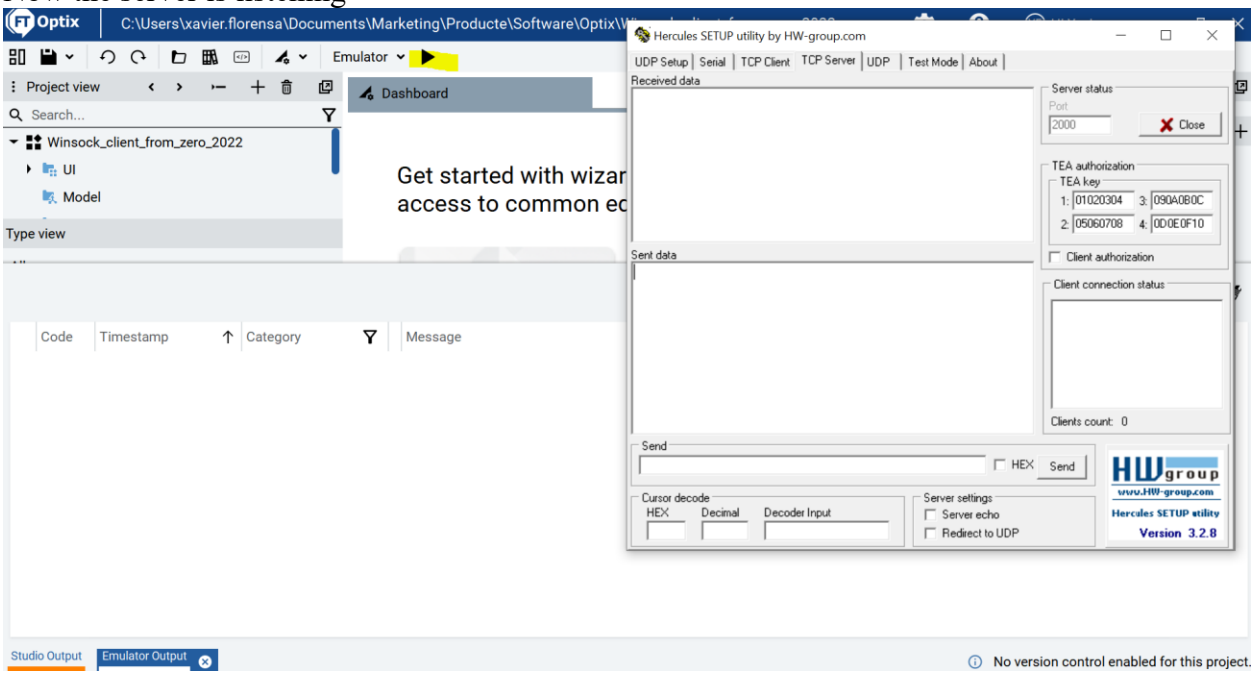
Let's try the client application with those callbacks

46.3. Thirt party libraries (dll) Socket Client application with Optix

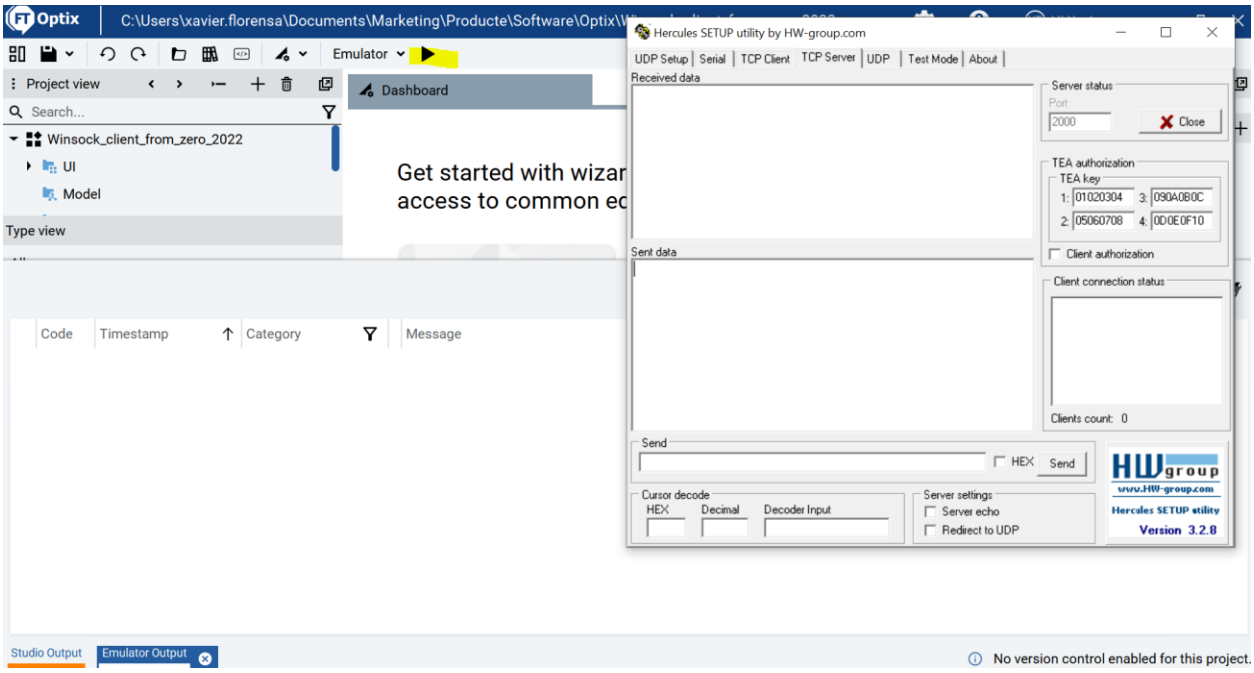
You can see the result here
https://youtu.be/6E4xt02Ohzw
You can download the project from here
https://github.com/xavierflorensa/Optix_Winsock_Orcas_Client.git
Test the program with a TCP server
First open the server and put it to listen on port 2000



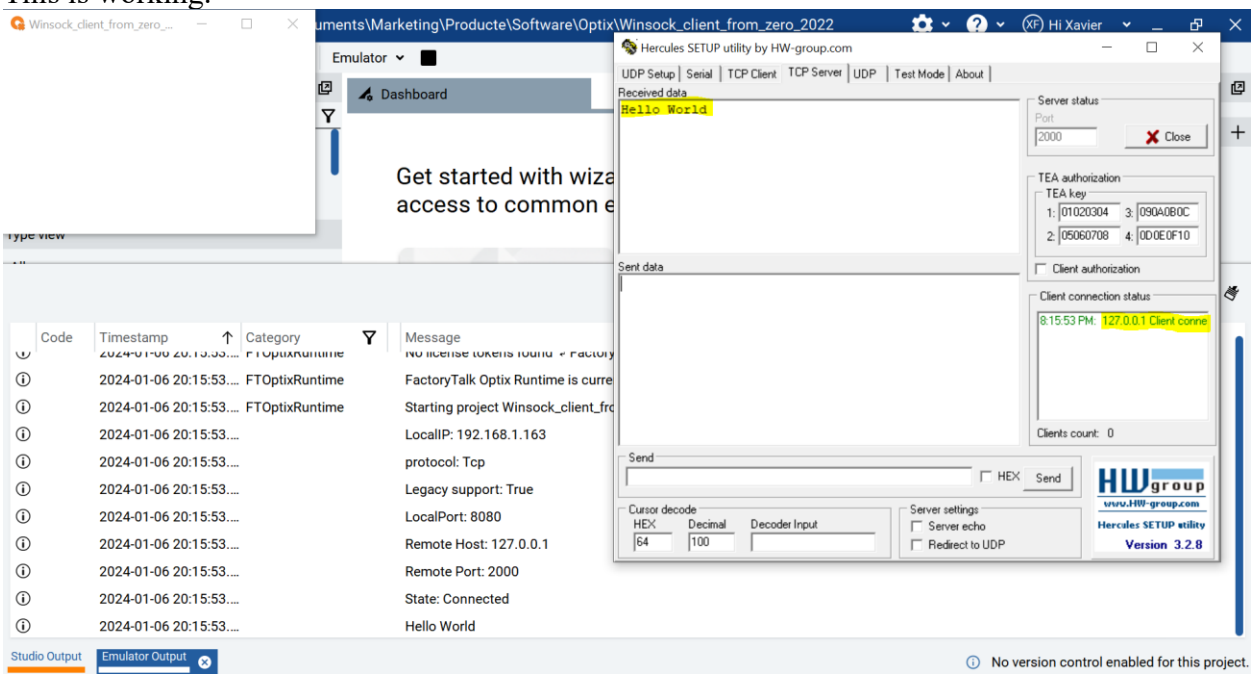
Now the server is listening



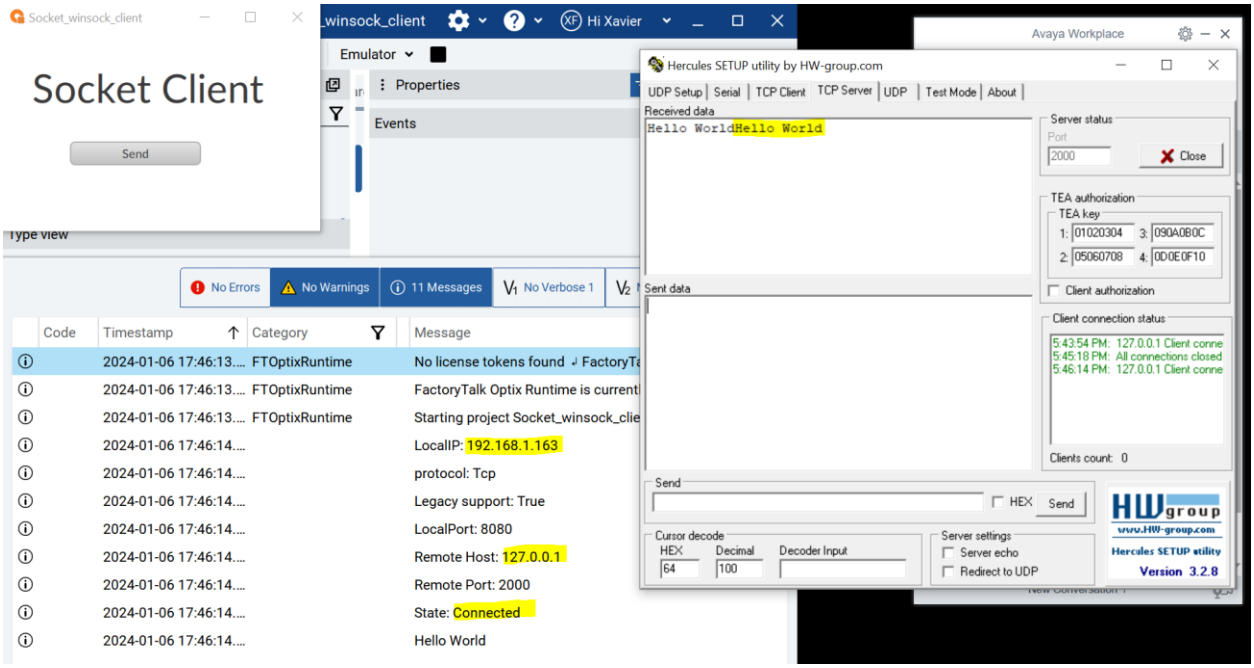
Next start the Optix project



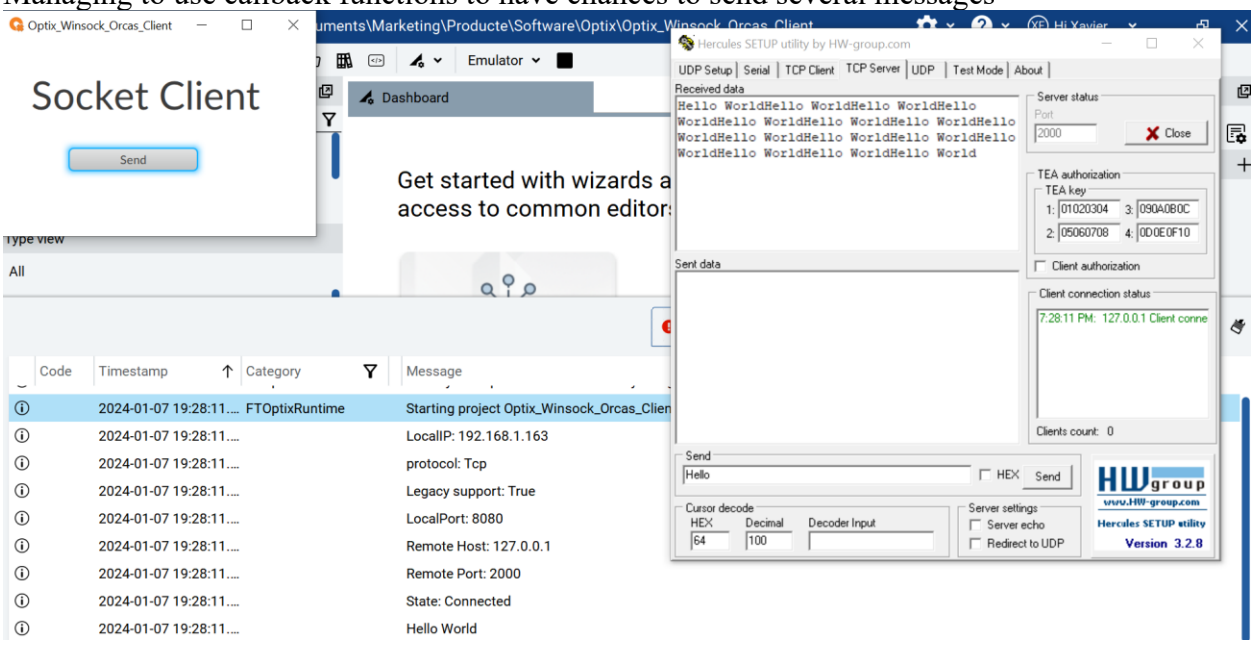
This is working.



But the connection is made and sending one message is done in one step, and then no chances to send more messages.
Pending to improve this.



Managing to use callback functions to have chances to send several messages



Using this code with the callback function

```
 //this is the callback function that will execute after the first connection
request
    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsockConnectionRequestEventArgs e)
        {
            winsock_Ear.Close();
            winsock_Ear.Accept(e.Client);
        }
```

This is the complete code

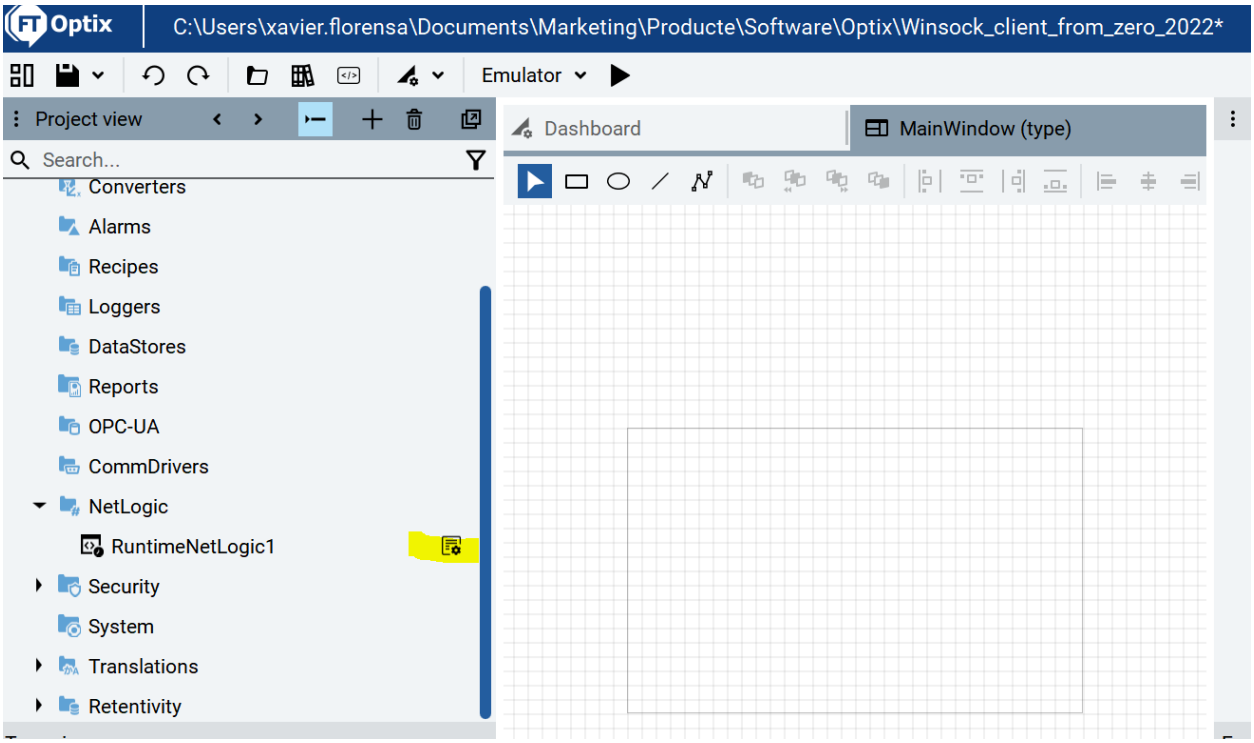
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    Winsock winsock_Ear = new Winsock(); //opens a new socket

    [ExportMethod]

    public void Send()
    {
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Start()
    {
       winsock_Ear.LegacySupport=true;
        //winsock_Ear.Listen(2000);//This is to make the PC act as host
        winsock_Ear.Connect("127.0.0.1", 2000); //This is to make the PC act as
client
        // Assign a callback to be excuted when the client is connected
        //Winsock_Ear.Connected += winsock_Ear_Connected;
        // Assign a callback to be executed when a message is received from the
server
        winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
```

```
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }

    public override void Stop()
    //public override void Stop()
    {
         Log.Info("Stopping");
    }
    //this is the callback function that will execute when a client is connected
    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
        {
            Log.Info("¡¡Conectado!!");
        }
    //this is the callback function that will execute after the first connection
request
    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsockConnectionRequestEventArgs e)
        {
            winsock_Ear.Close();
            winsock_Ear.Accept(e.Client);
        }
    private Winsock Winsock_Ear;
 }
```

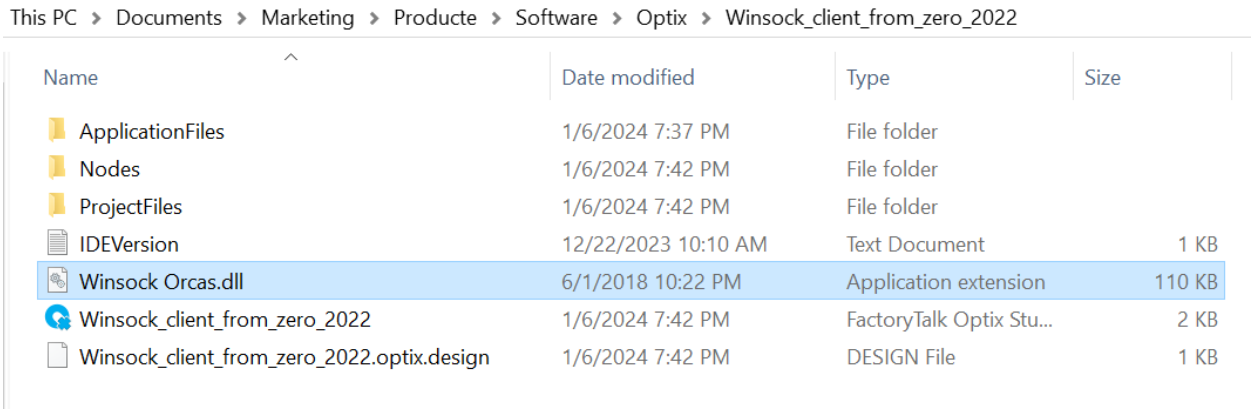
https://github.com/xavierflorensa/Optix_Winsock_Orcas_Client.git

46.3.1. Installing external references (dll libraries) into Visual Studio project 2022

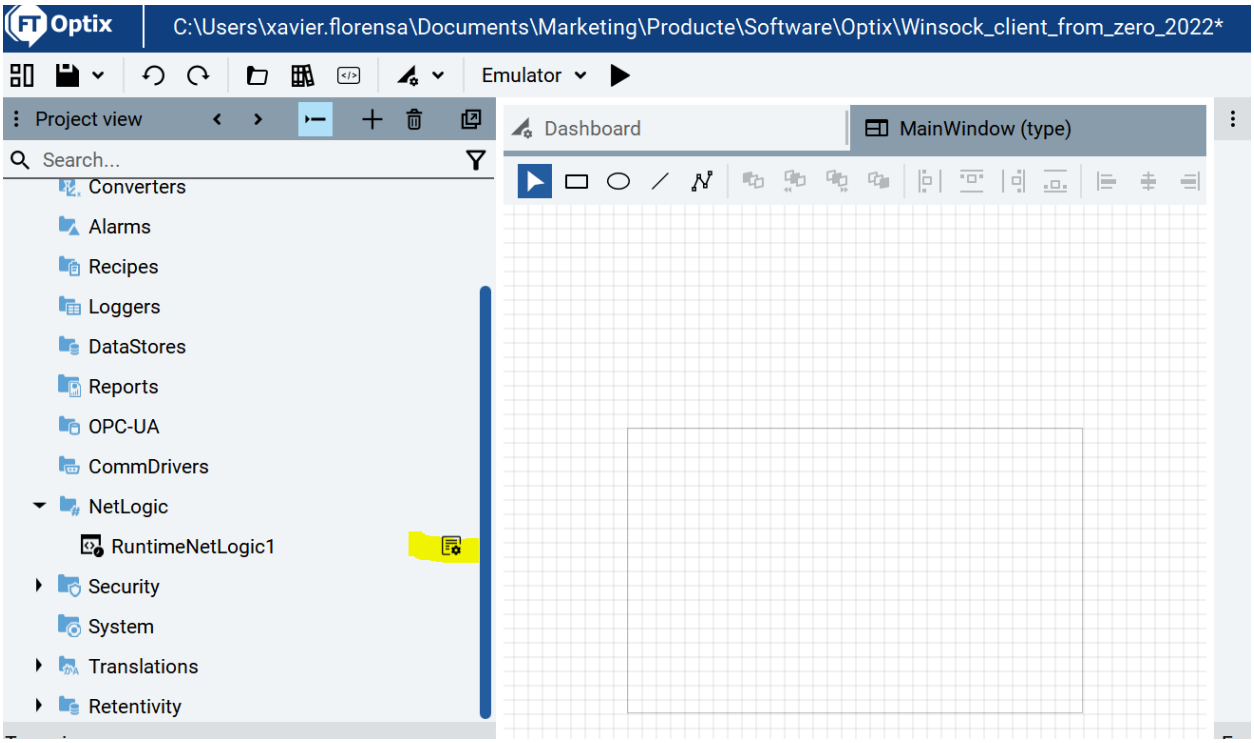
Let's start this application from zero
The easiest way to install external references to a project is using Visual Studio 2002, since it will do all the work for you. Then if you prefer you can continue with Visual Studio Code since it is much faster to open and close.
Start a new project in FT Optix Studio
Add a new NetLogic code



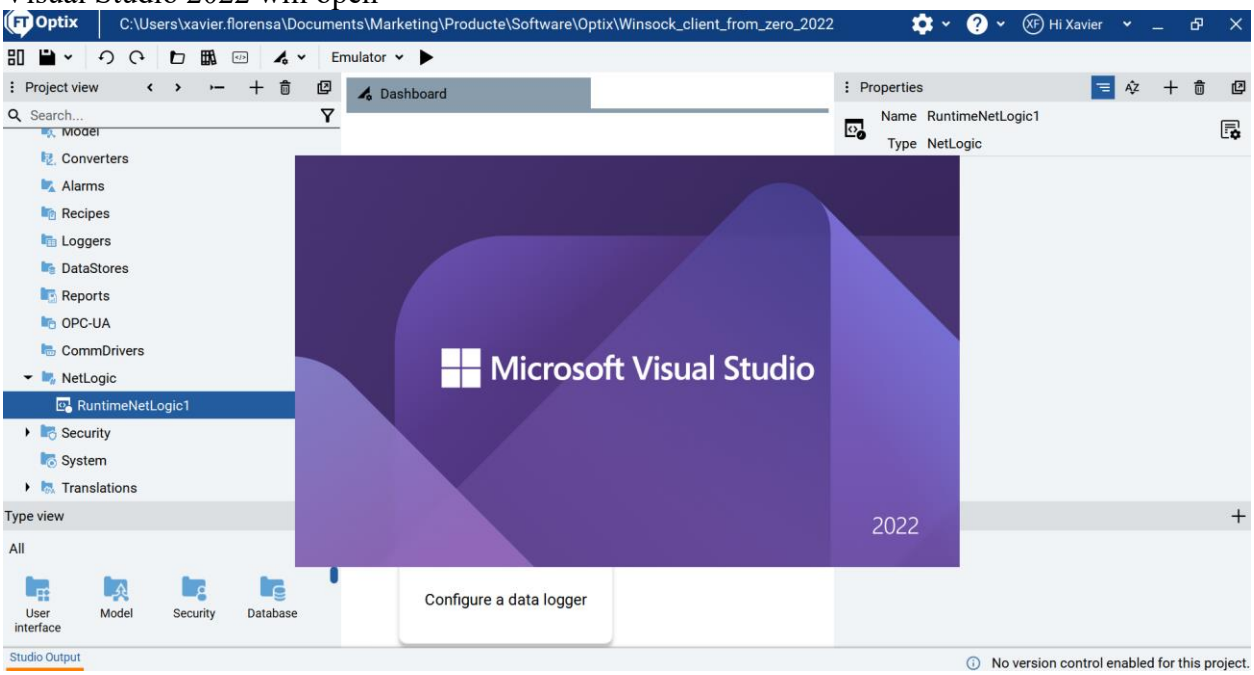
Save the project
Insert this dll on the root directory of the project
Winsock_Orcas.dll

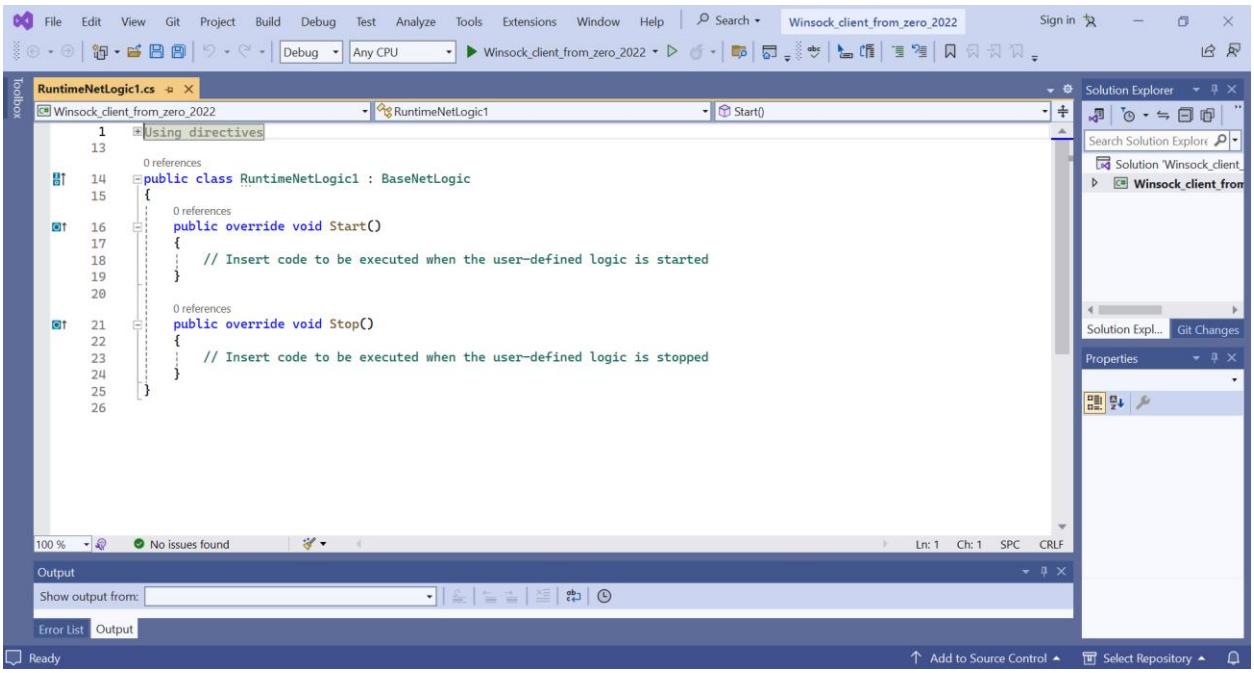


Select editor to Visual Studio 2022
Close and open FT Optix to have this change updated.
Click on the code

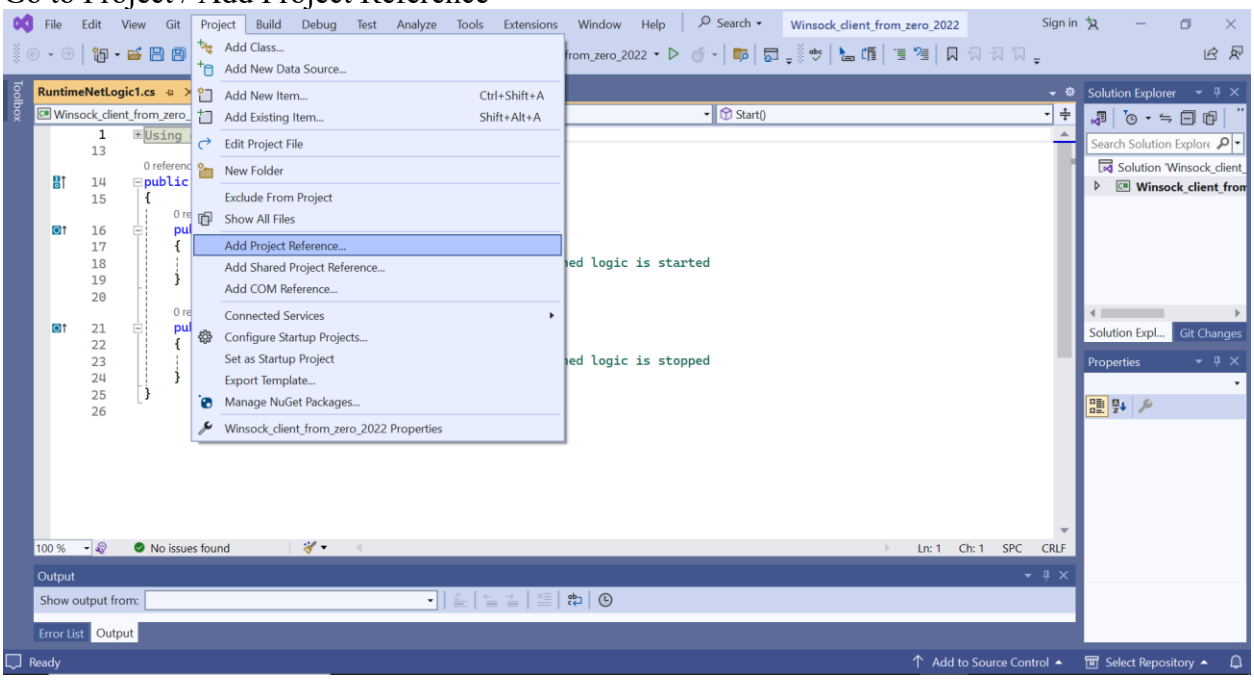


Visual Studio 2022 will open

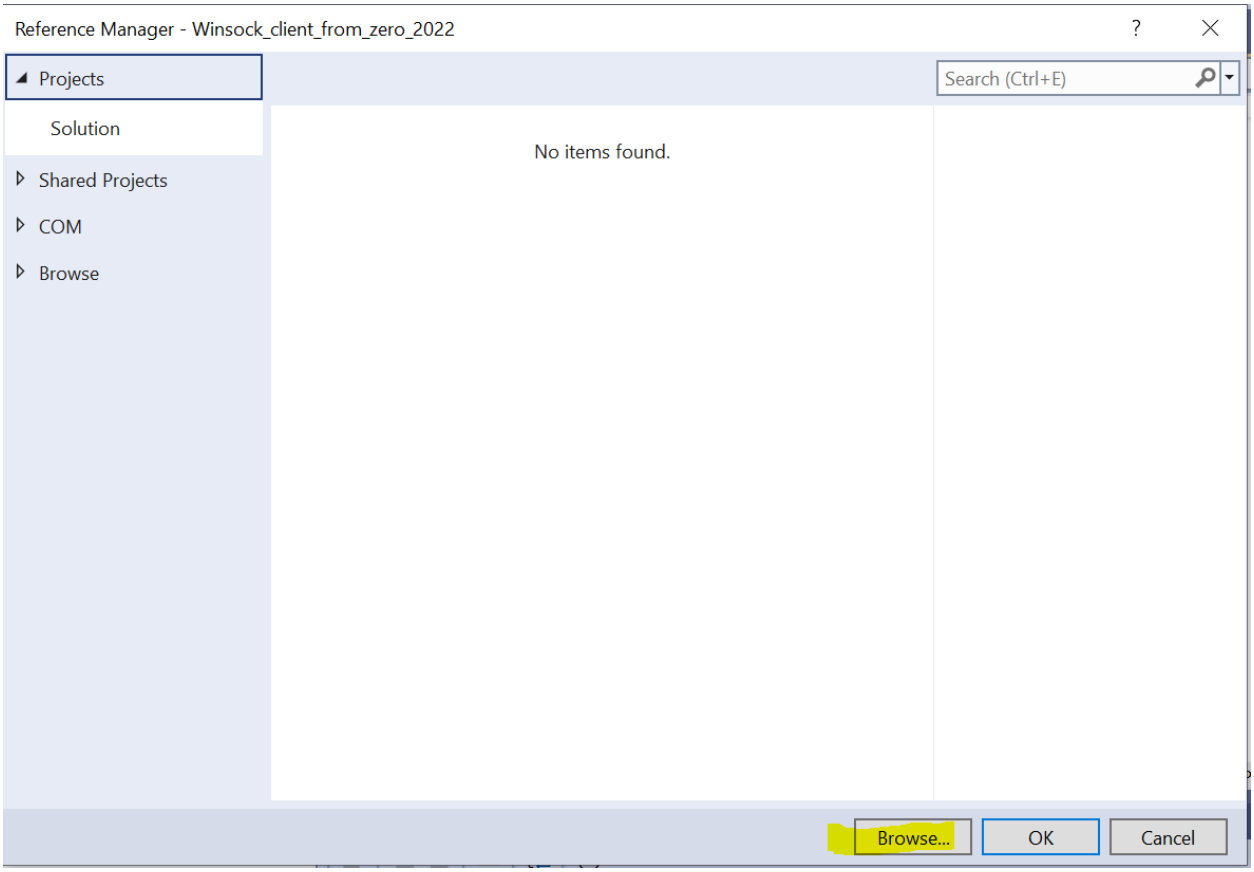




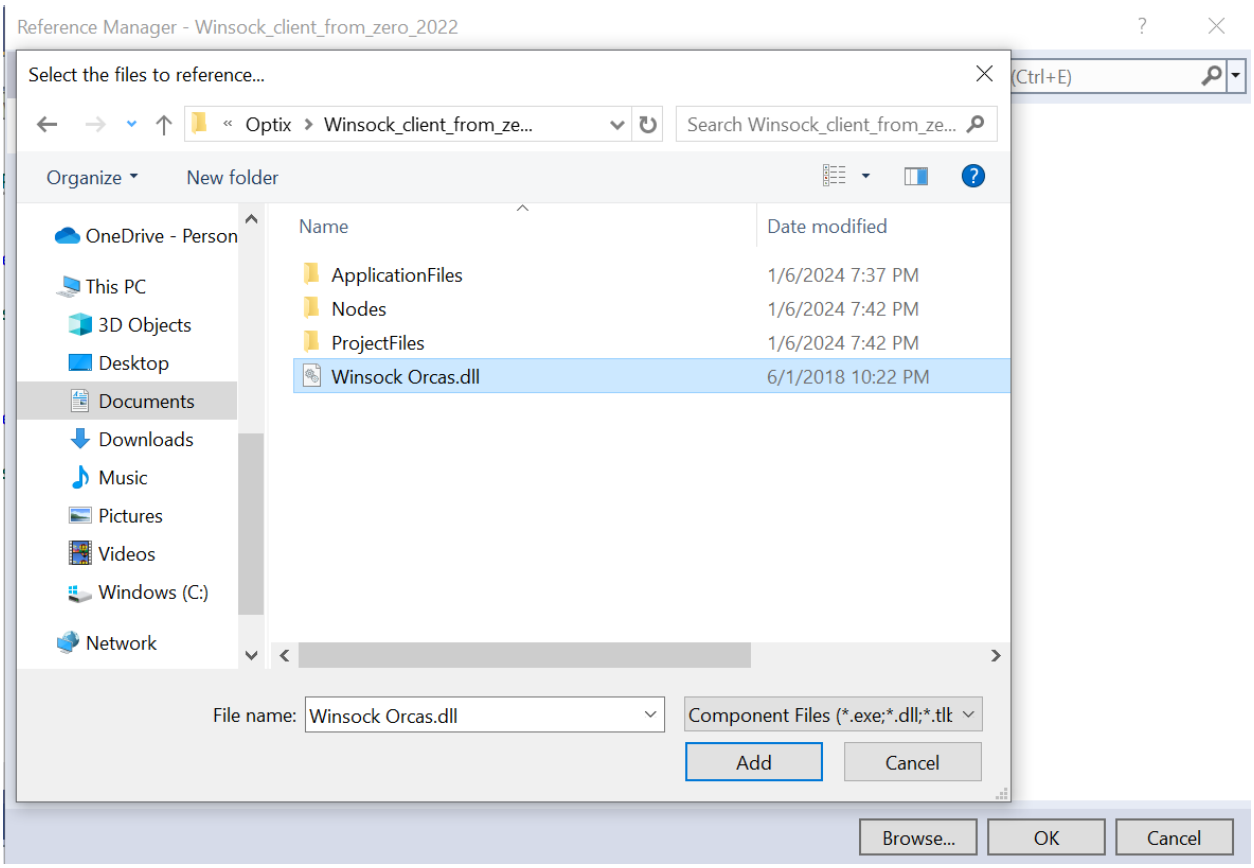
Go to Project / Add Project Reference



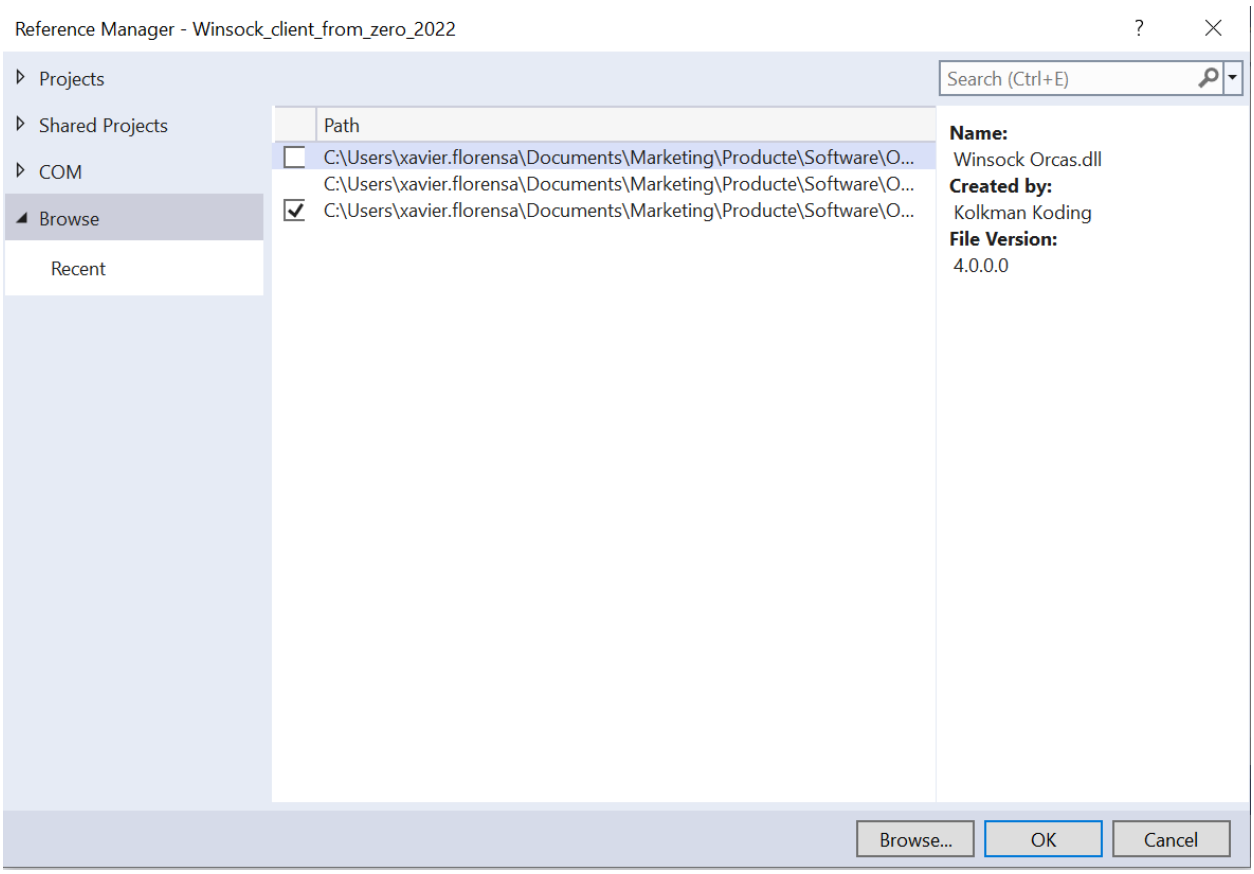
Click on Browse



Select the dll

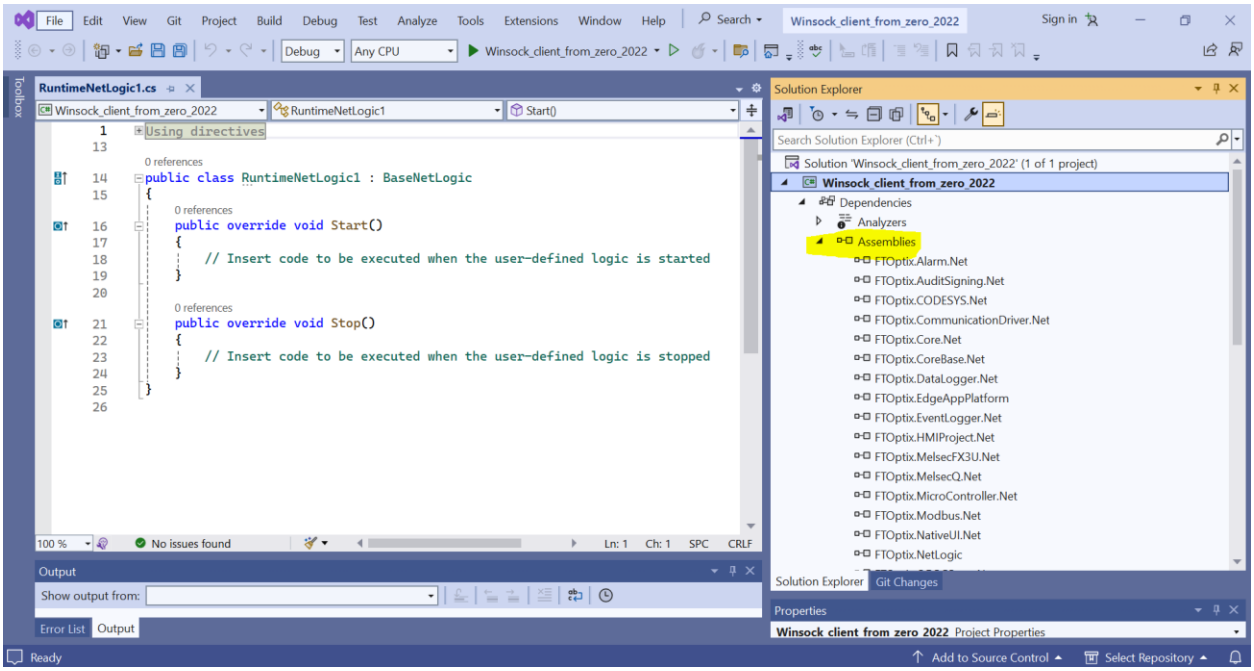


Verify that this is the right path

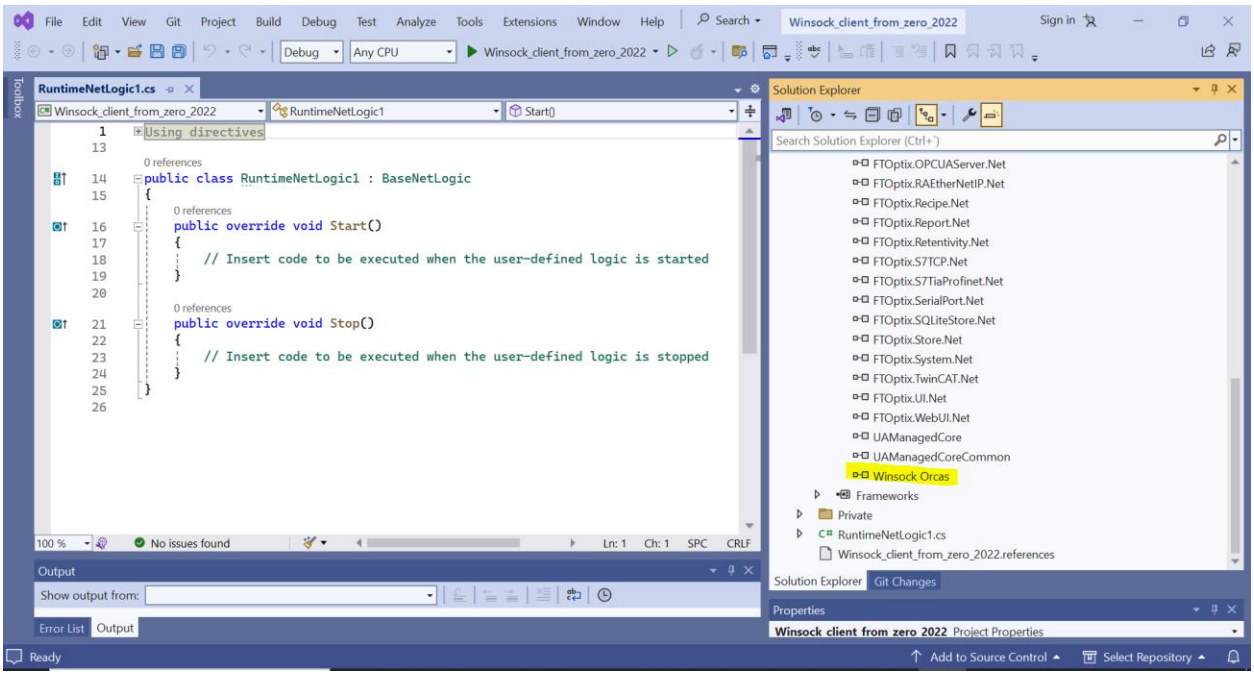


Click Ok

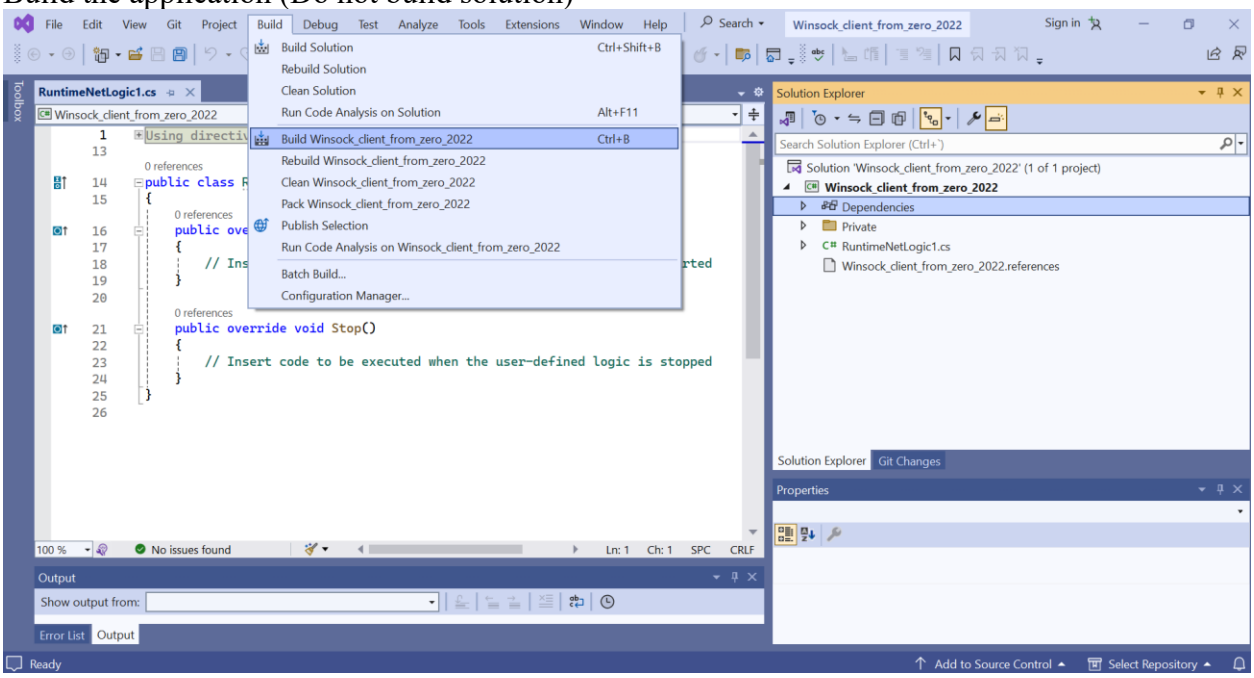
Unfold assemblies



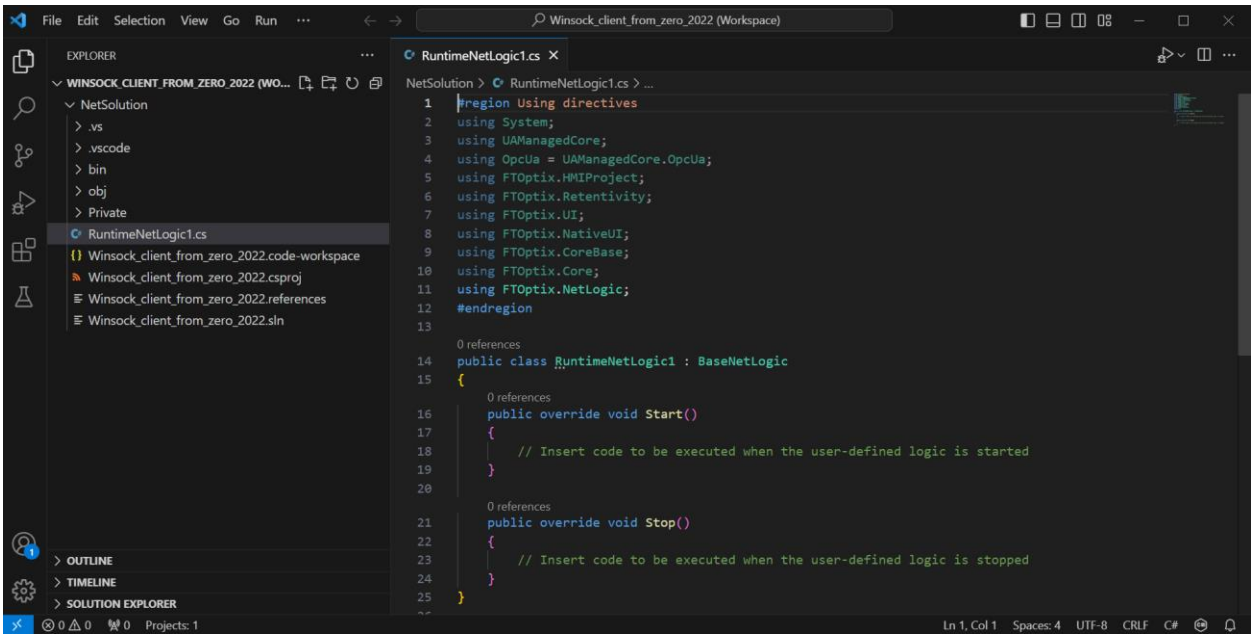
You have it installed



Build the application (Do not build solution)

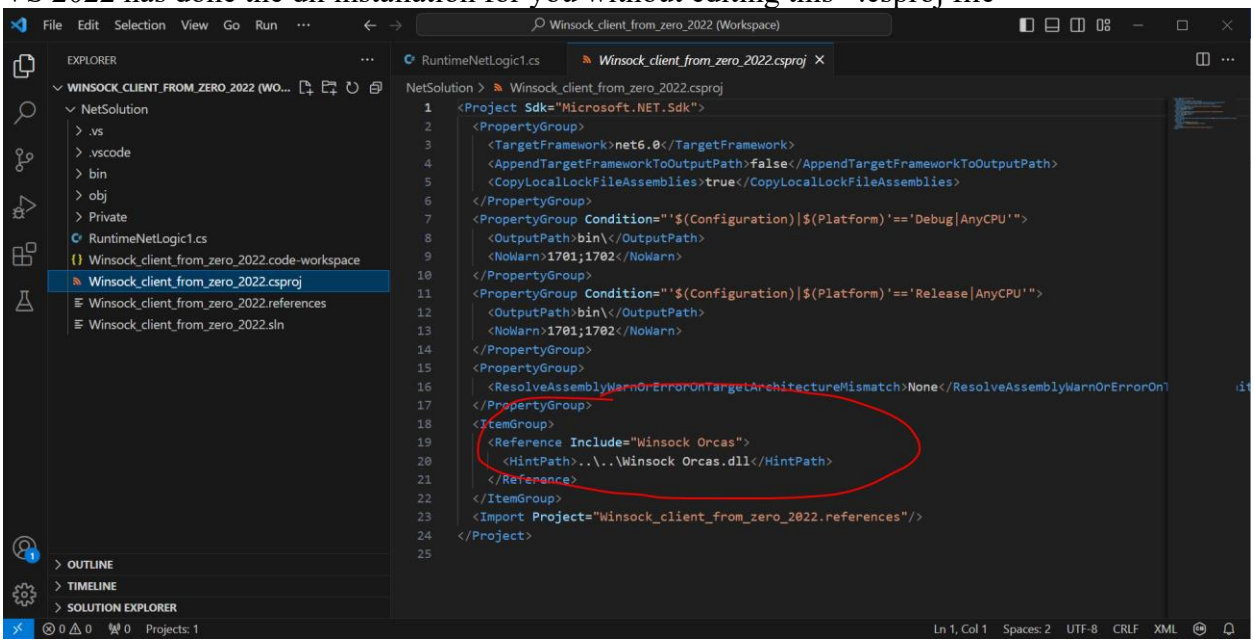


You can close Visual Studio 2022
Change the code editor to Visual Studio Code in FT Optix Studio
Save close and open it again
Open Netlogic



Open references file *.csproj

VS 2022 has done the dll installation for you without editing this *.csproj file

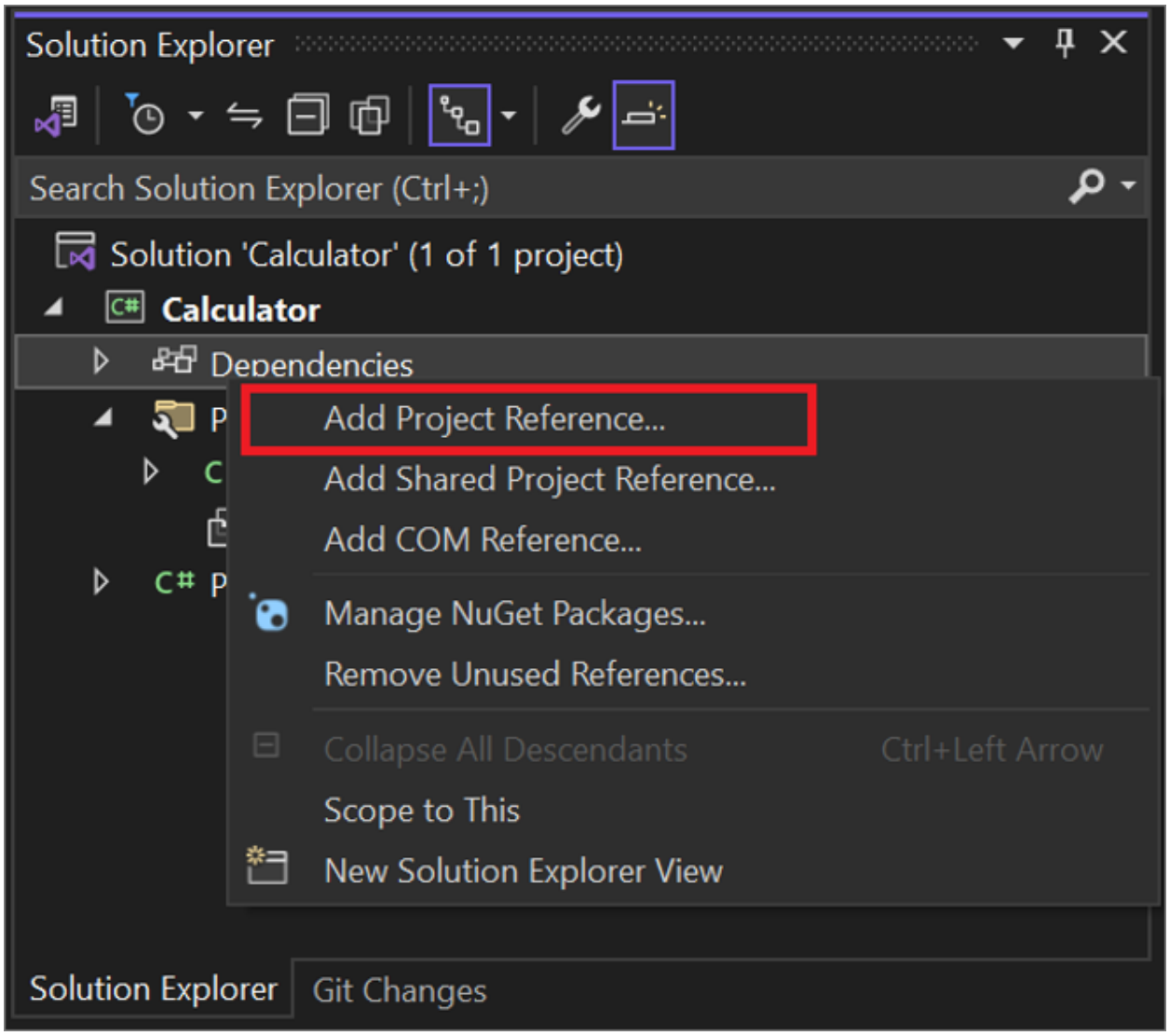


46.3.2. Installing external references (dll libraries) into Visual Studio Code project

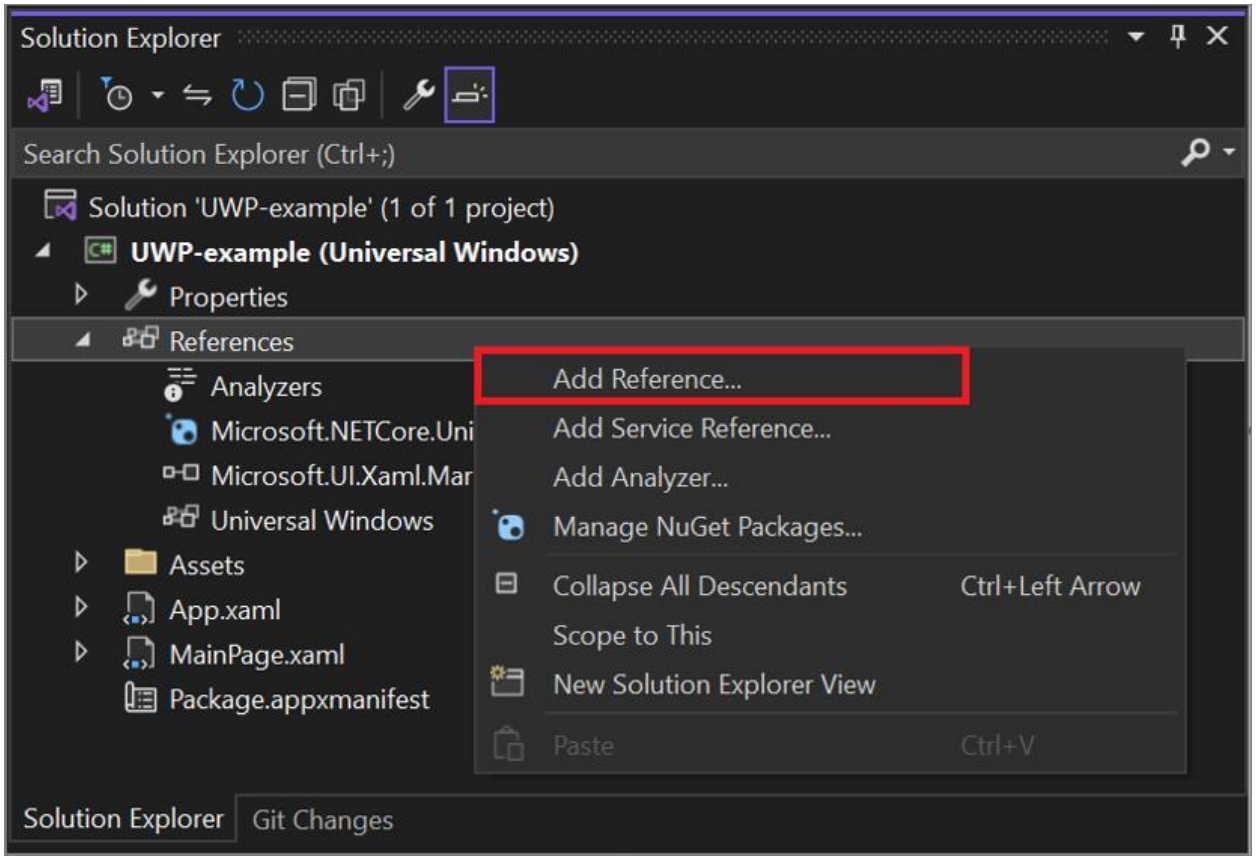
Pending to test this way

How you add a reference depends on the project type for the code you're working on:

- If you see a **Dependencies** node in **Solution Explorer**, you can use the right-click context menu to select **Add Project Reference**.



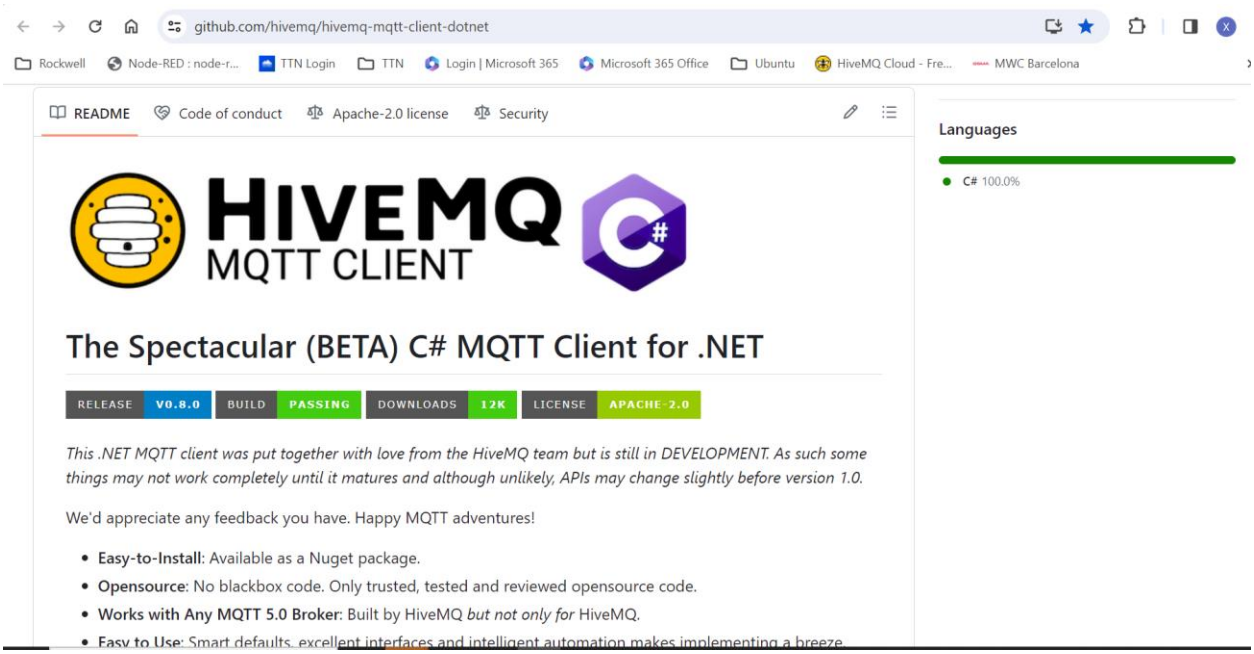
- If you see a **References** node in **Solution Explorer**, you can use the right-click context menu to select **Add Reference**.



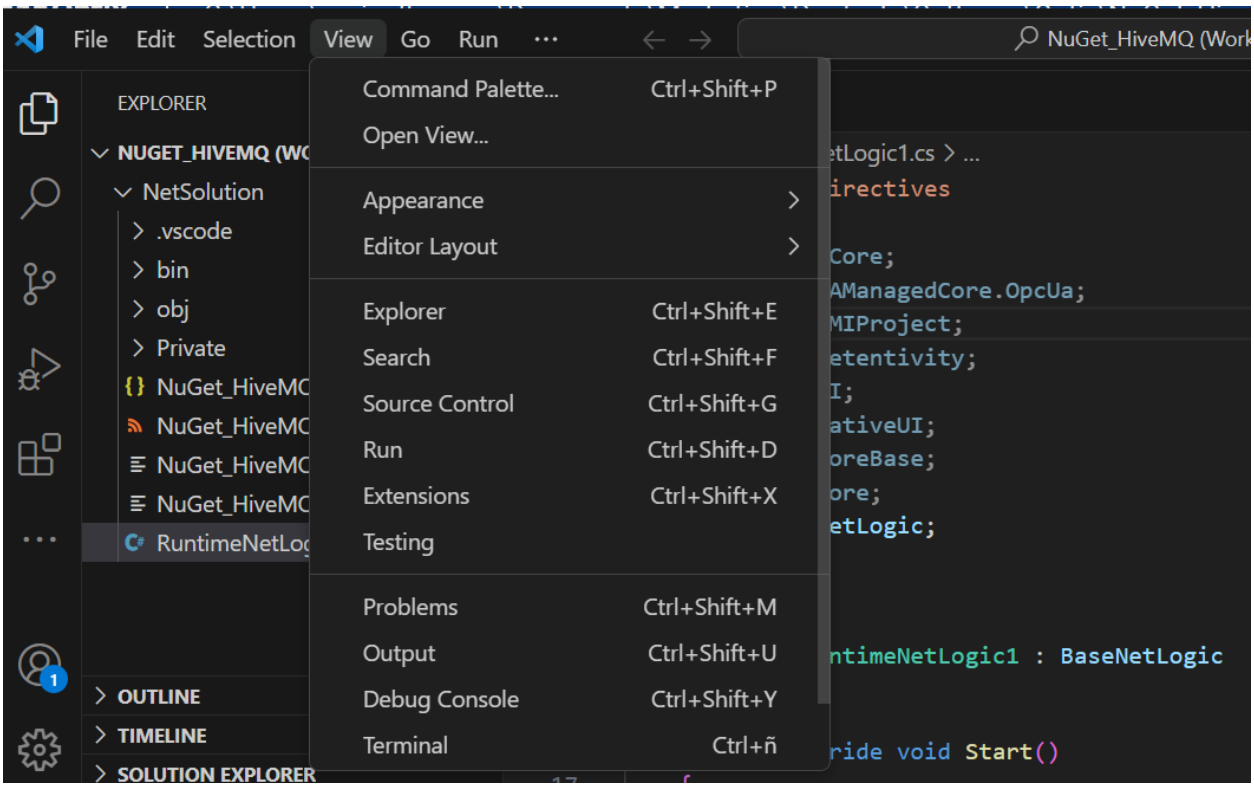
For more information, see [How to: Add or remove references](#).

47. Using NuGet libraries (HiveMQ .Net example)

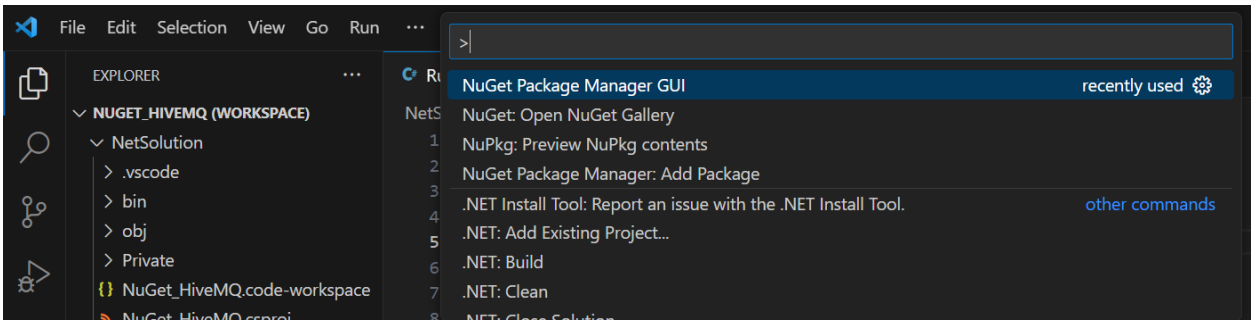
Let's try to use this fancy library,
https://github.com/hivemq/hivemq-mqtt-client-dotnet
You can see the result on this video
https://youtu.be/3g81mO2wqXQ



So create a new project and add a new Netlogic Runtime code
Open the code
Click on View/Command Pallete

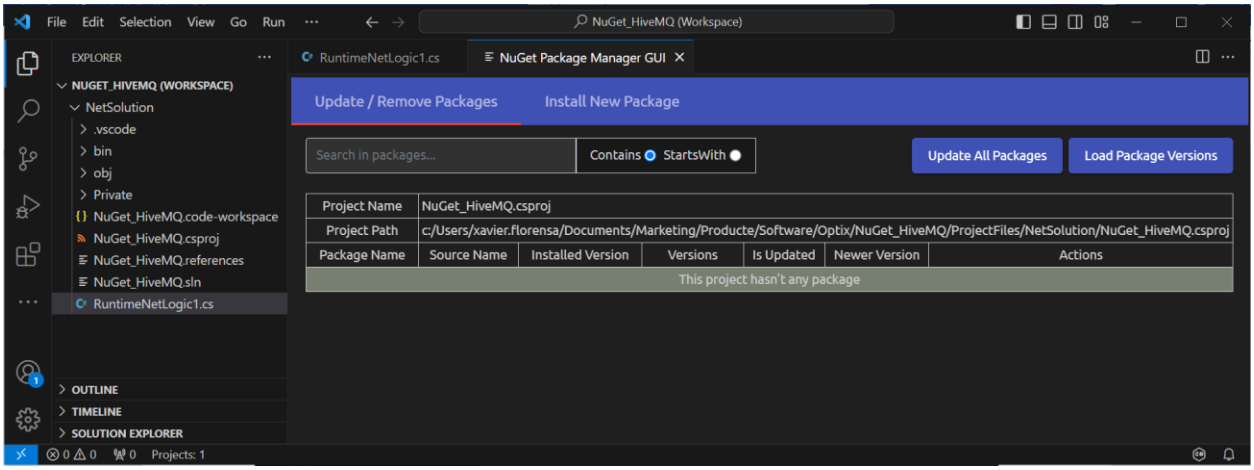


You will see this depending on what VScode extensions you have installed

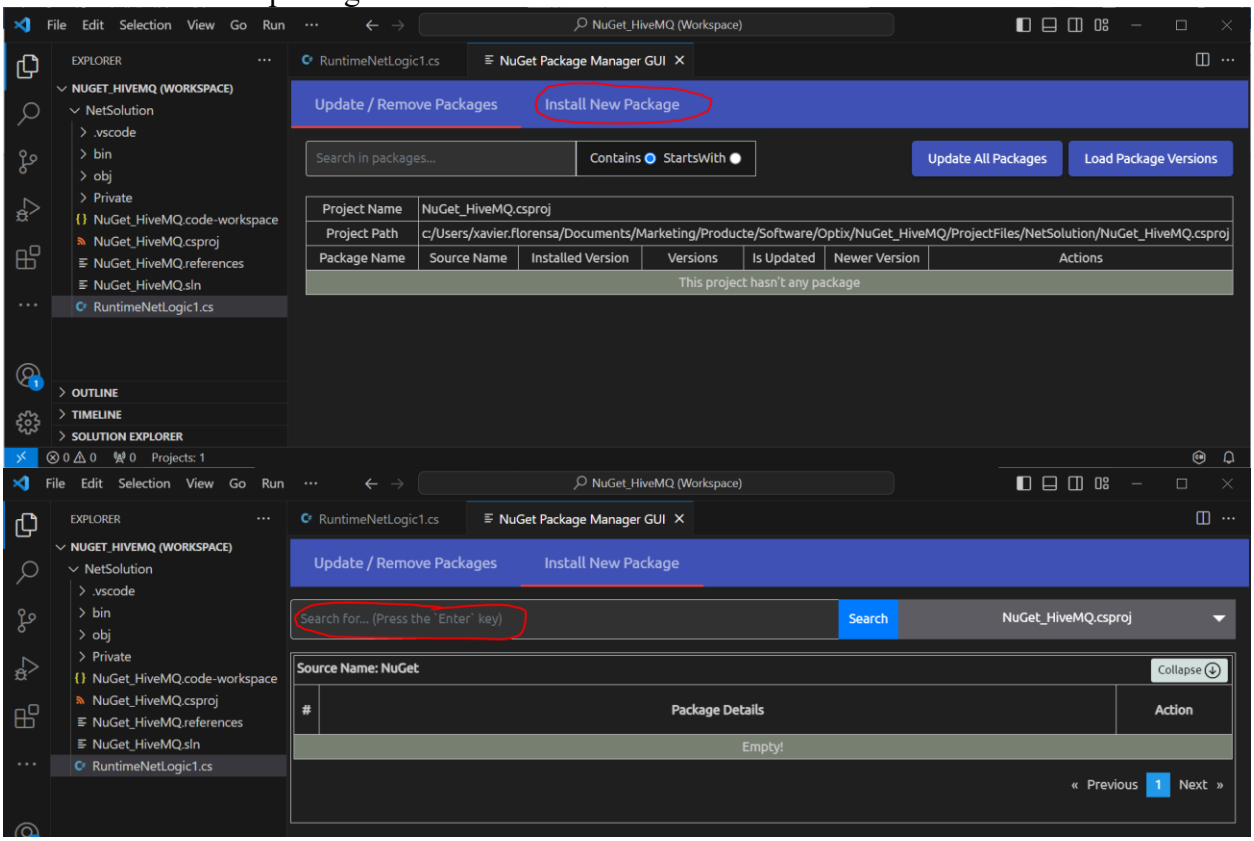


Select NuGet Package Manager GUI

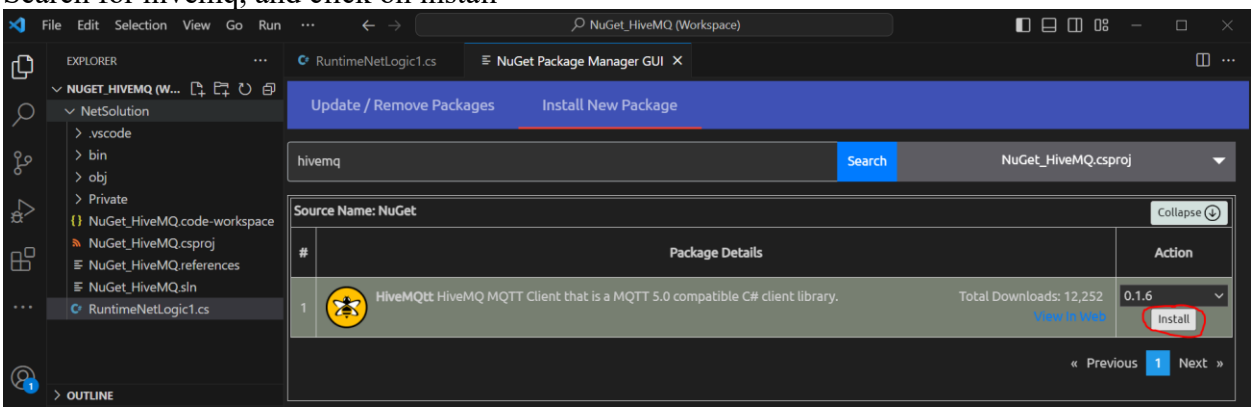
You will see this



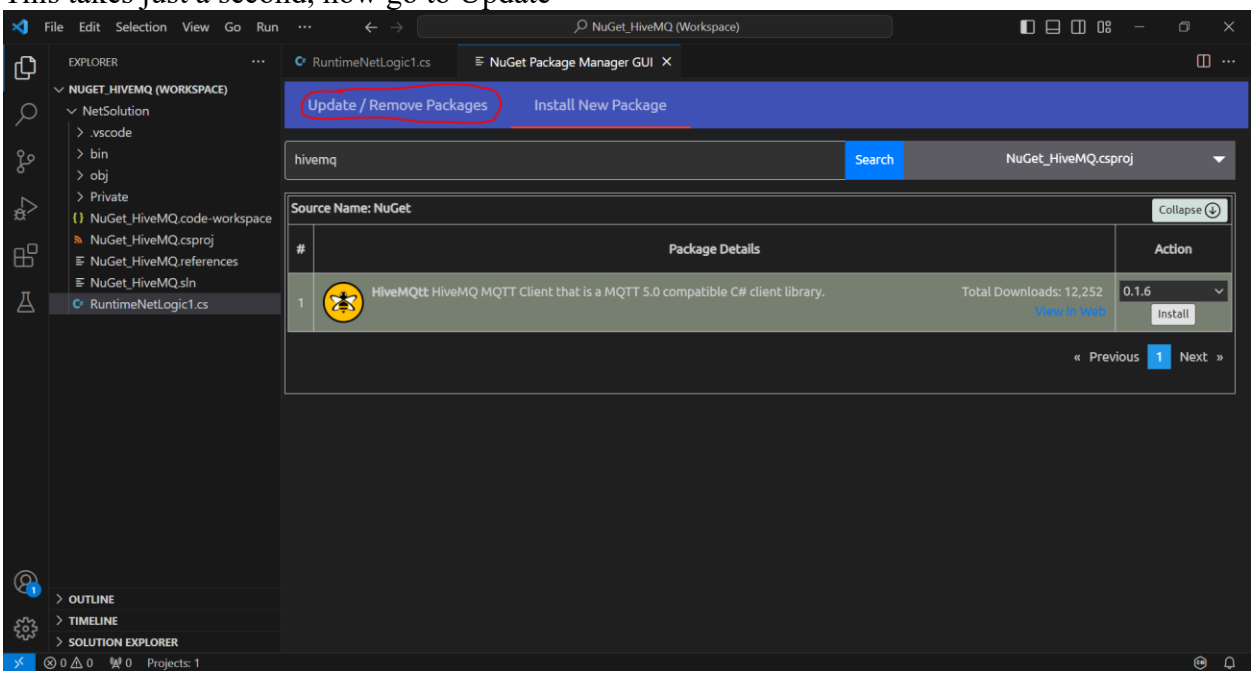
Click on Install new package



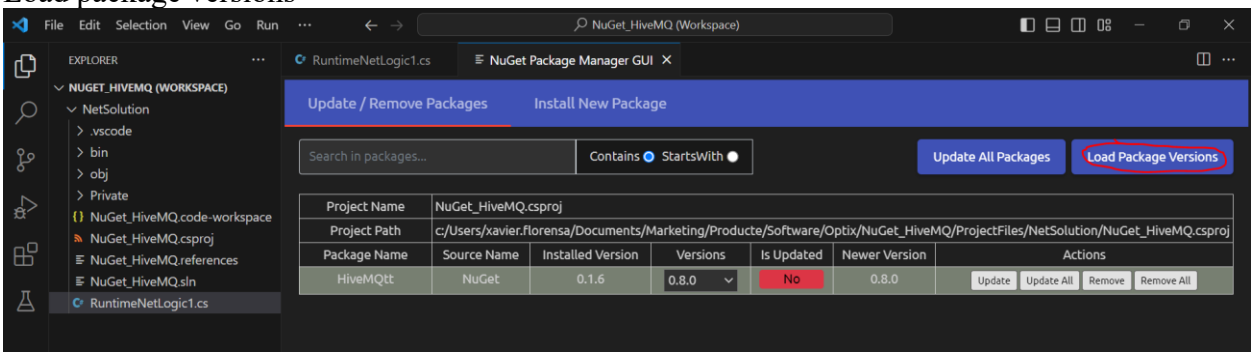
Search for hivemq, and click on install



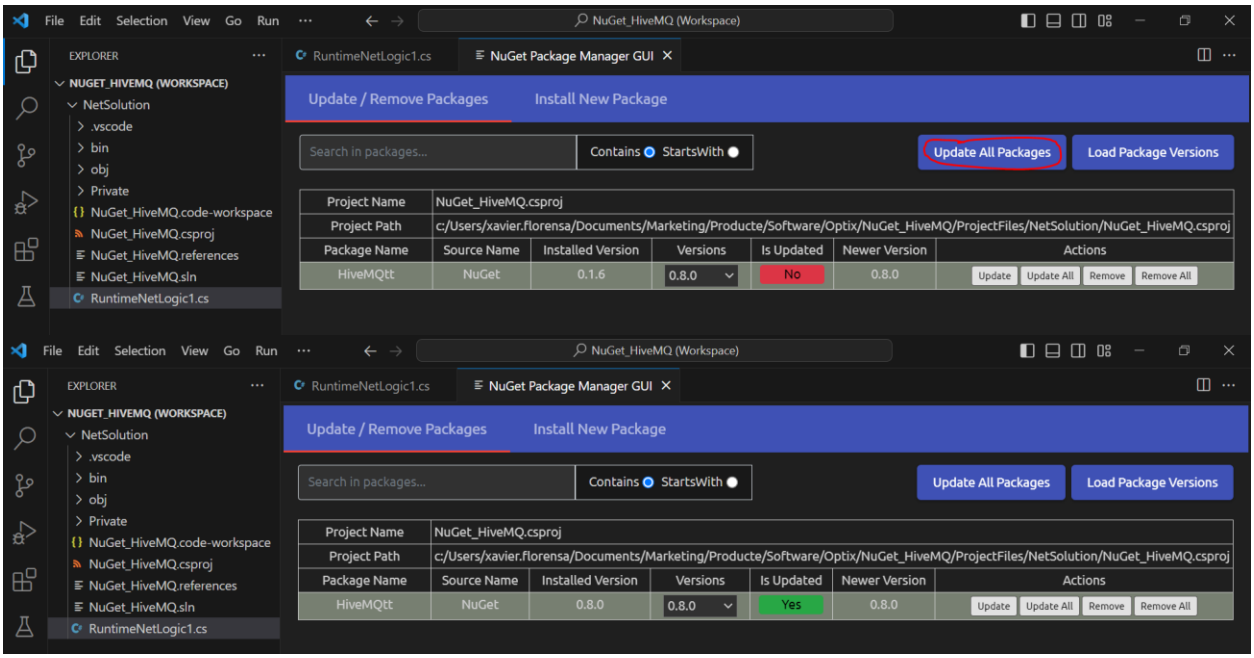
This takes just a second, now go to Update



Load package versions



Update all packages



You are done, take a look at *.csproj file

```
  </PropertyGroup>
  <Import Project="NuGet_HiveMQ.references" />
  <ItemGroup>
    <PackageReference Include="HiveMQtt" Version="0.8.0" />
  </ItemGroup>
</Project>
```

You are ready to use this library

Copy these two directives to your code

```
using HiveMQtt.Client;
using HiveMQtt.MQTT5.Types;
```

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using HiveMQtt.Client;
using HiveMQtt.MQTT5.Types;
#endregion
```

```
public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
    }
```

Now insert this code on Start()
Be aware that the procedure is public override Start()

```
// Setup Client options and instantiate
        var options = new
HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com").WithPort(1883).WithU
seTls(false).Build();
        var client = new HiveMQClient(options);
        // Connect to the MQTT broker
        var connectResult = await client.ConnectAsync().ConfigureAwait(false);
        // Publish a message
        var publishResult = await client.PublishAsync("topic1/example", "Hello
World");
```

See these errors

```
public class RuntimeNetLogic1 : BaseNetLogic
{
    0 references
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        // Setup Client options and instantiate
        var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com").WithPort(1883).WithUs
        var client = new HiveMQClient(options);
        // Connect to the MQTT broker
        var connectResult = await client.ConnectAsync().ConfigureAwait(false);
        // Publish a message
        var publishResult = await client.PublishAsync("topic1/example", "Hello World");
    }
```

You can correct by typing the sentences again like this

```
var connectResult =
```

You can copy and comment the line and start writing

```
// Connect to the MQTT broker
//var connectResult = await client.ConnectAsync().ConfigureAwait(false);
var connectResult = await
```

Write client and click on the box client

```
public override void Start()
{
    // Insert code to be executed when the user-defined logic is started
    // Setup Client options and instantiate
    var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync().ConfigureAwait(false);
    var connectResult = await client
    // Publish a message                    client
```

Then . and select ConnectAsync

```
0 references
public override void Start()
{
    // Insert code to be executed whe
    // Setup Client options and insta
    var options = new HiveMQClientOpt
    var client = new HiveMQClient(opt
    // Connect to the MQTT broker
    //var connectResult = await clien
    var connectResult = await client.
```

AfterUnsubscribe
BeforeConnect
BeforeDisconnect
BeforeSubscribe
BeforeUnsubscribe
ConnectAsync
DisconnectAsync
Dispose

. and select configureawait

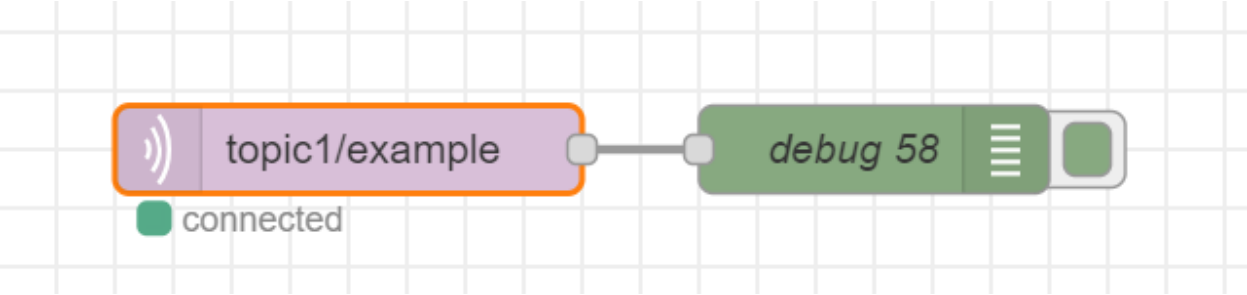
```
0 references
public override void Start()
{
    // Insert code to be executed when the user-defi
    // Setup Client options and instantiate
    var options = new HiveMQClientOptionsBuilder().W
    var client = new HiveMQClient(options);
    // Connect to the MQTT broker
    //var connectResult = await client.ConnectAsync(
    var connectResult = await client.ConnectAsync().
```

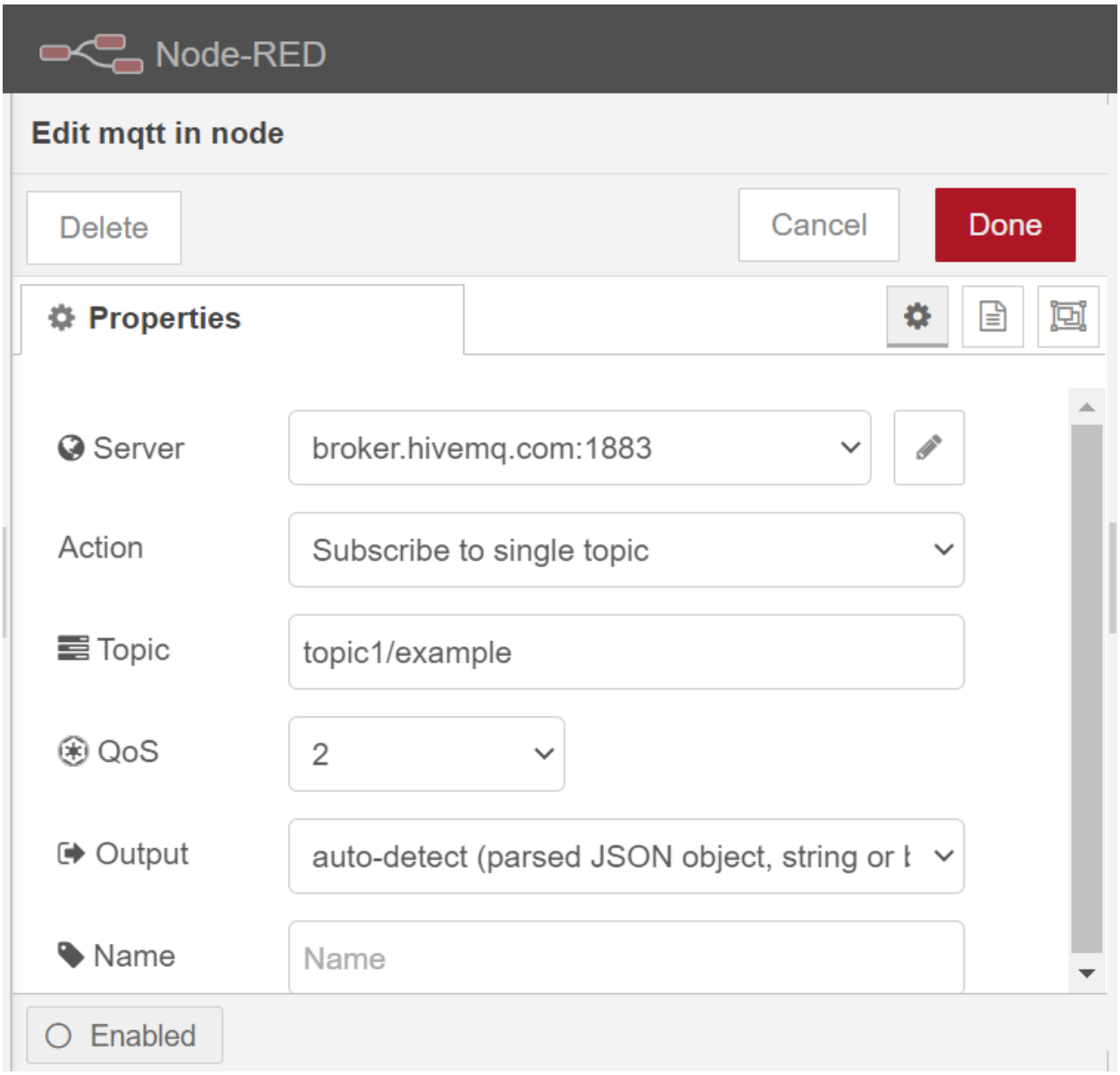
AsyncState
ConfigureAwait
ContinueWith
ContinueWith<>
CreationOptions
Dispose
Equals
Exception
GetAwaiter

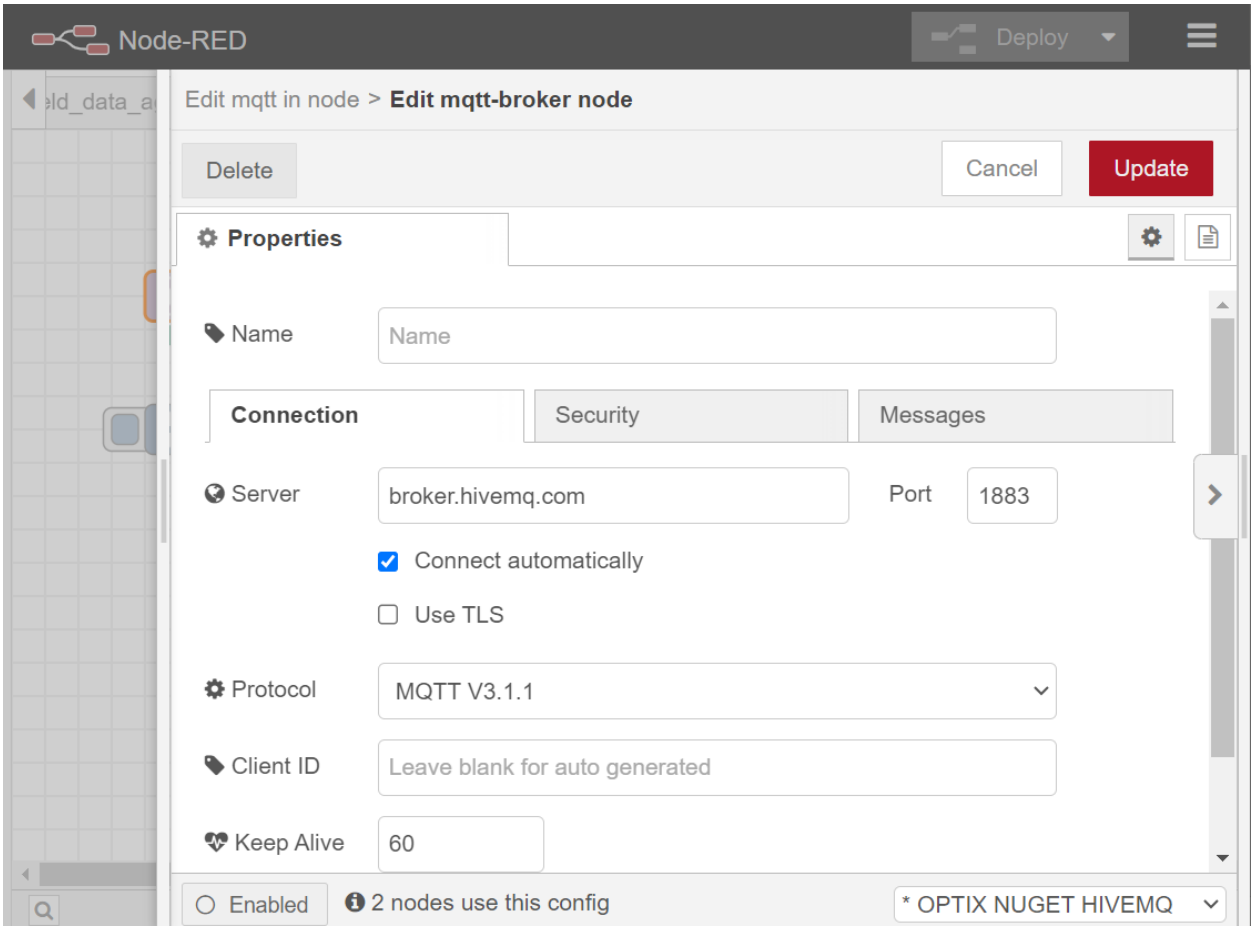
and so on until you get no errors, be aware of the change the compiler has made for you

```
public class RuntimeNetLogic1 : BaseNetLogic
{
    0 references
    public override async void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        // Setup Client options and instantiate
        var options = new HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com").With
        var client = new HiveMQClient(options);
        // Connect to the MQTT broker
        var connectResult = await client.ConnectAsync().ConfigureAwait(false);
            // Publish a message
        var publishResult = await client.PublishAsync("topic1/example", "Hello World");
```

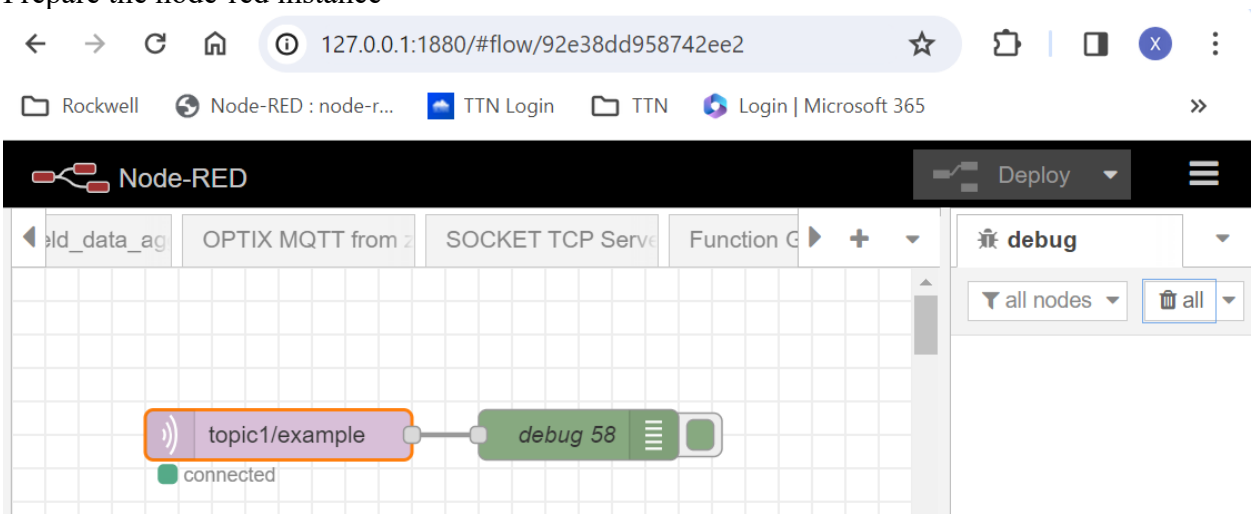
Then save and before executing open a node-red instance to verify the Optix project is working fine



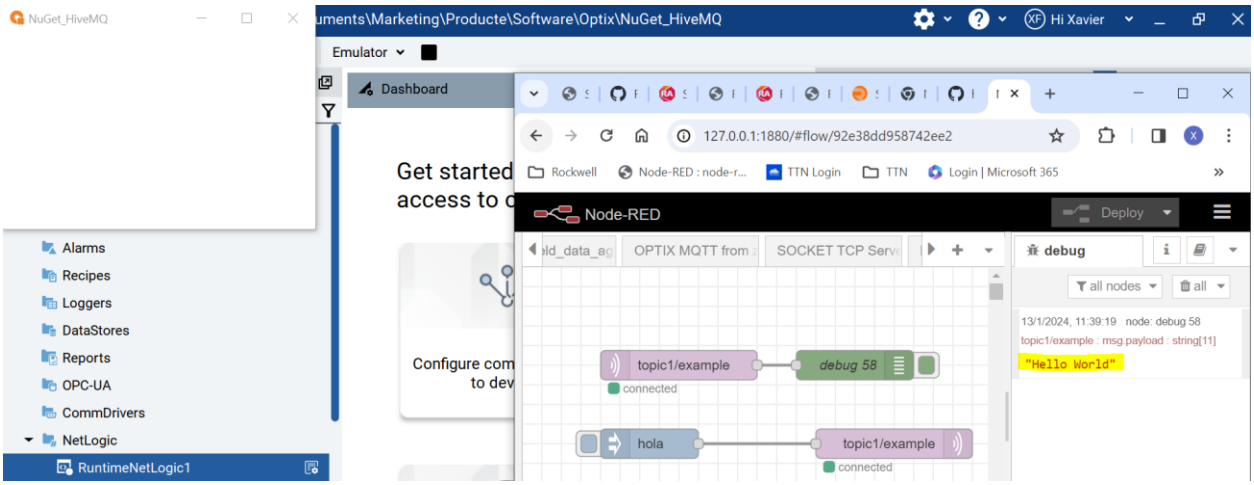

Node-RED
Edit mqtt in node
Delete
Cancel
Done
Properties
Server
broker.hivemq.com:1883
Action
Subscribe to single topic
Topic
topic1/example
QoS
2
Output
auto-detect (parsed JSON object, string or b
Name
Name
Enabled



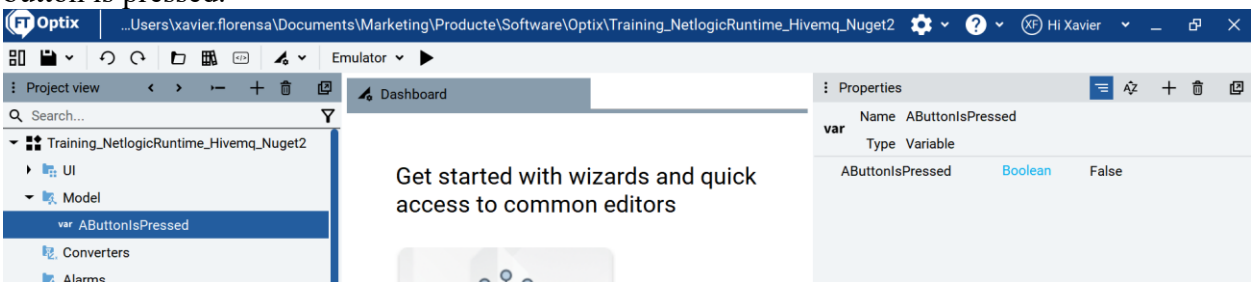
Prepare the node-red instance



Then execute the Optix application



That's all.
Here you have connection and publishing message in one shot.
The ideal would be, first connecting, and then asynchronously publish.
You can try with this code, but be careful with infinite loops that may cause uncontrolled program behaviour.
Let's create a variable to store a boolean value that will be set to true and then false when a button is pressed.



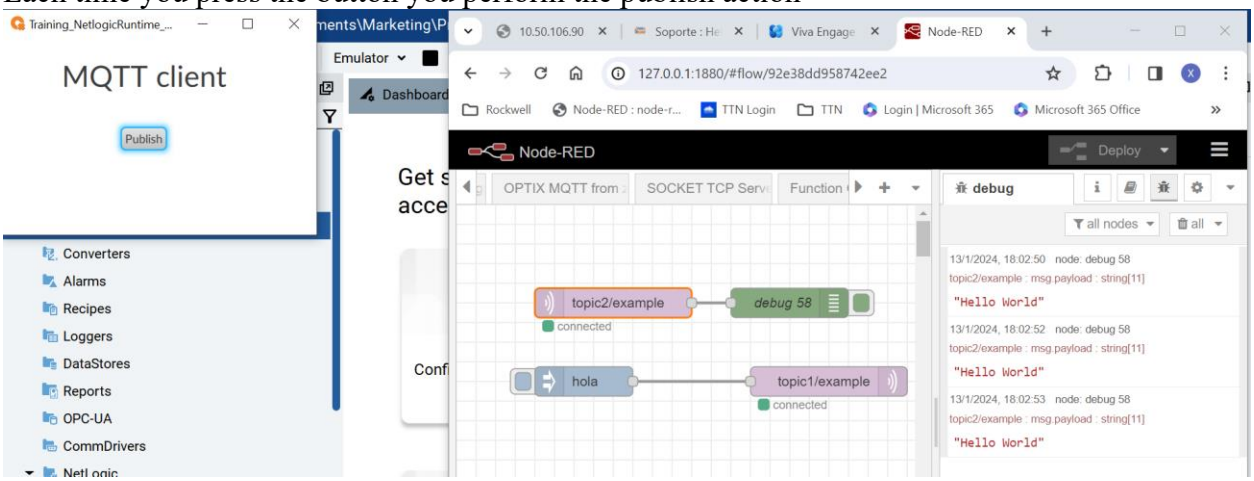
Then use this code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using HiveMQtt.Client;
using HiveMQtt.MQTT5.Types;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
```

```
    public override async void Start()
    {
        bool abuttonispressed=false;
        var Abuttonispressed =
Project.Current.GetVariable("Model/AButtonIsPressed");
        // Insert code to be executed when the user-defined logic is started
        // Setup Client options and instantiate
        var options = new
HiveMQClientOptionsBuilder().WithBroker("broker.hivemq.com")
            .WithPort(1883)
            .WithUseTls(false)
            .Build();

        var client = new HiveMQClient(options);
        // Connect to the MQTT broker
        var connectResult = await client.ConnectAsync().ConfigureAwait(false);
        //hold until a button is pressed
        while (true)
        {
            while (!abuttonispressed)
            {
            //holds until a button is pressed

            abuttonispressed = Abuttonispressed.Value;
            }
        // Publish a message
        var publishResult = await client.PublishAsync("topic2/example", "Hello
World");
        abuttonispressed=false;
        Abuttonispressed.Value=false;
        }
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
    [ExportMethod]

    public  void Publish()
    {
        var Abuttonispressed =
Project.Current.GetVariable("Model/AButtonIsPressed");
        Abuttonispressed.Value=true;
    }
```

```
}
```

Now it works
Each time you press the button you perform the publish action



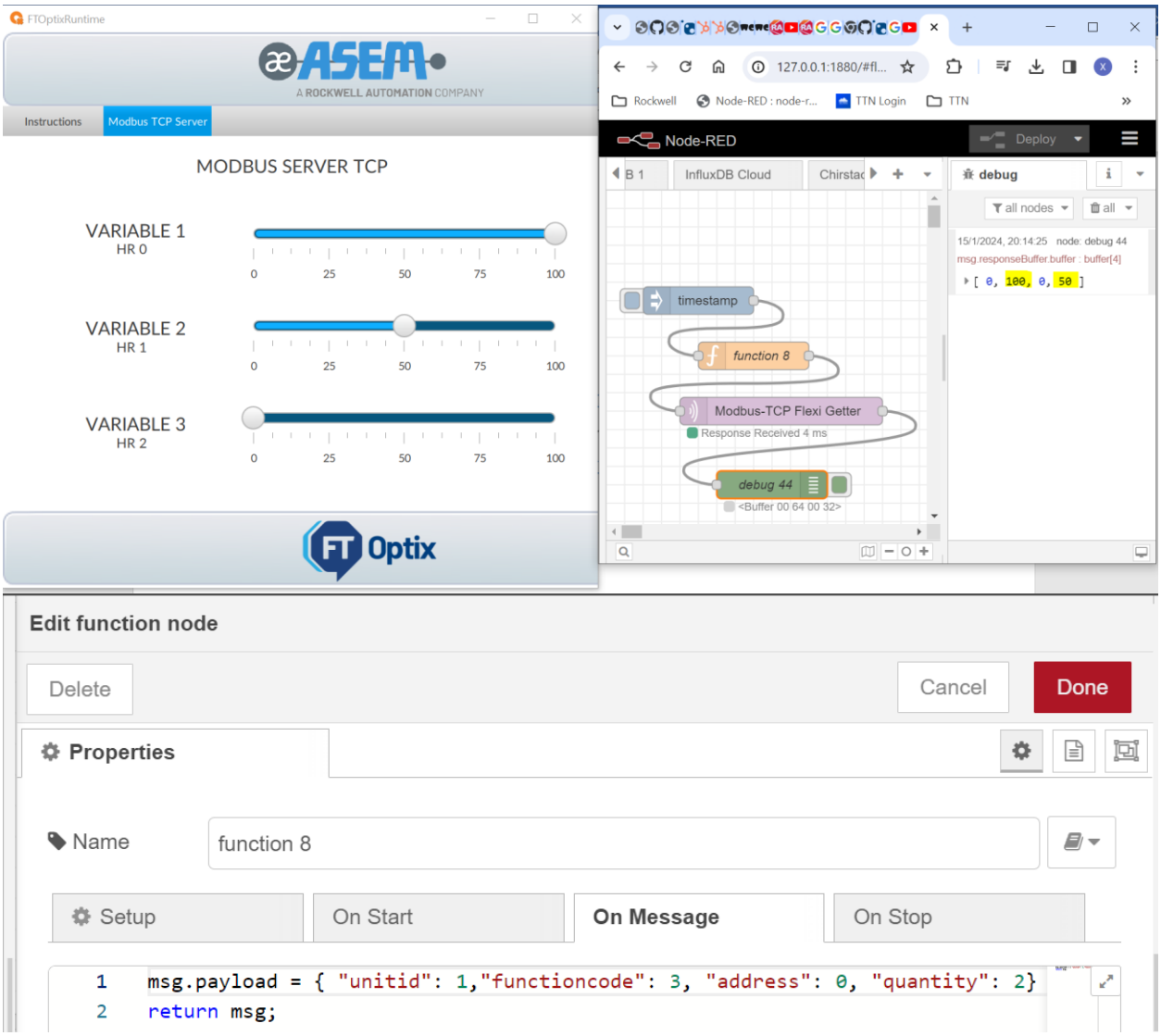
As you can see on this video
https://youtu.be/3g81mO2wqXQ

48. Modbus communications

48.1. Modbus server with FTOptix

You can try this example on Modbus TCP Server from Rockwell Automation repository
You can download the Server FactoryTalk Optix code from here
https://github.com/FactoryTalk-Optix/ModbusServerTCP
You can see this working in this video
https://youtu.be/UpRB72bkKkA

Let’s run the Optix application, and open a Modbus client


FTOptixRuntime
ASEM
A ROCKWELL AUTOMATION COMPANY
Instructions
Modbus TCP Server
MODBUS SERVER TCP
VARIABLE 1
HR 0
VARIABLE 2
HR 1
VARIABLE 3
HR 2
0
25
50
75
100
FT Optix
127.0.0.1:1880/#fl...
Rockwell
Node-RED : node-r...
TTN Login
TTN
Node-RED
Deploy
B 1
InfluxDB Cloud
Chirstac
debug
all nodes
all
15/1/2024, 20:14:25 node: debug 44
msg.responseBuffer.buffer : buffer[4]
[0, 100, 0, 50]
timestamp
function 8
Modbus-TCP Flexi Getter
Response Received 4 ms
debug 44
<Buffer 00 64 00 32>
Edit function node
Delete
Cancel
Done
Properties
Name
function 8
Setup
On Start
On Message
On Stop
msg.payload = { "unitid": 1,"functioncode": 3, "address": 0, "quantity": 2}
return msg;

Edit modbus-tcp-ip node

Delete | Cancel | Done

Properties

Name: Modbus-TCP Flexi Getter

IP: 127.0.0.1

Port: 502

Log Errors: ☐

Edit debug node

Delete | Cancel | Done

Properties

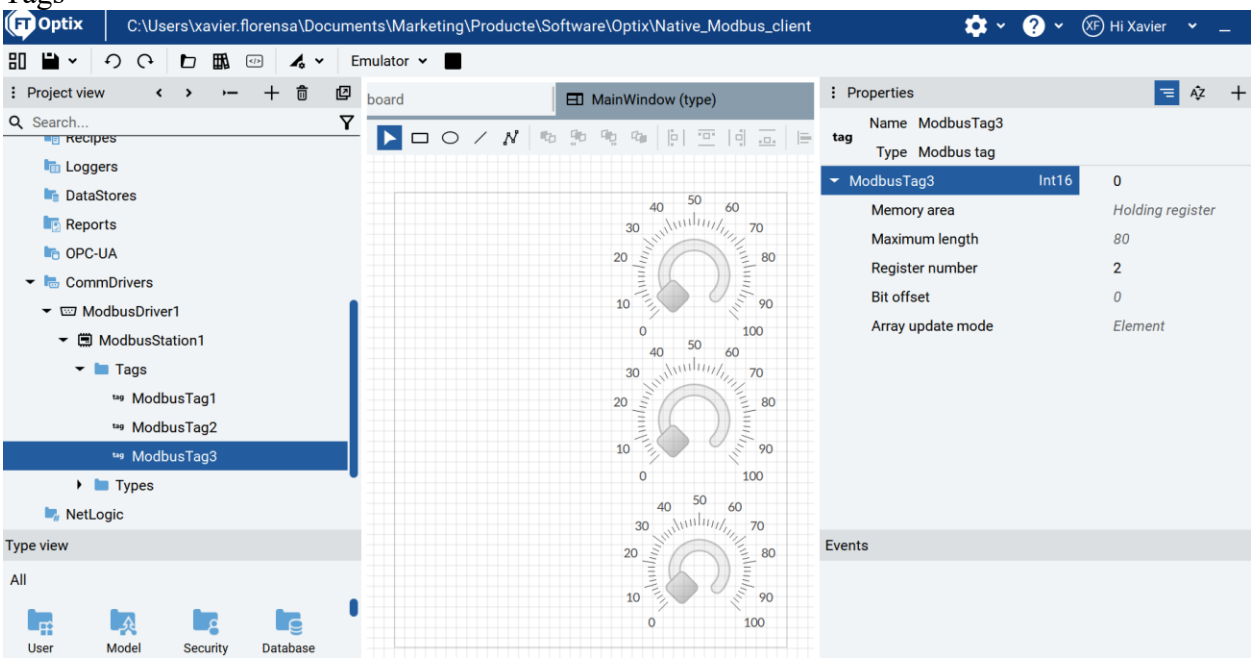
Output: msg. responseBuffer.buffer

To:
- ☑ debug window
- ☐ system console
- ☑ node status (32 characters)
- same as debug output

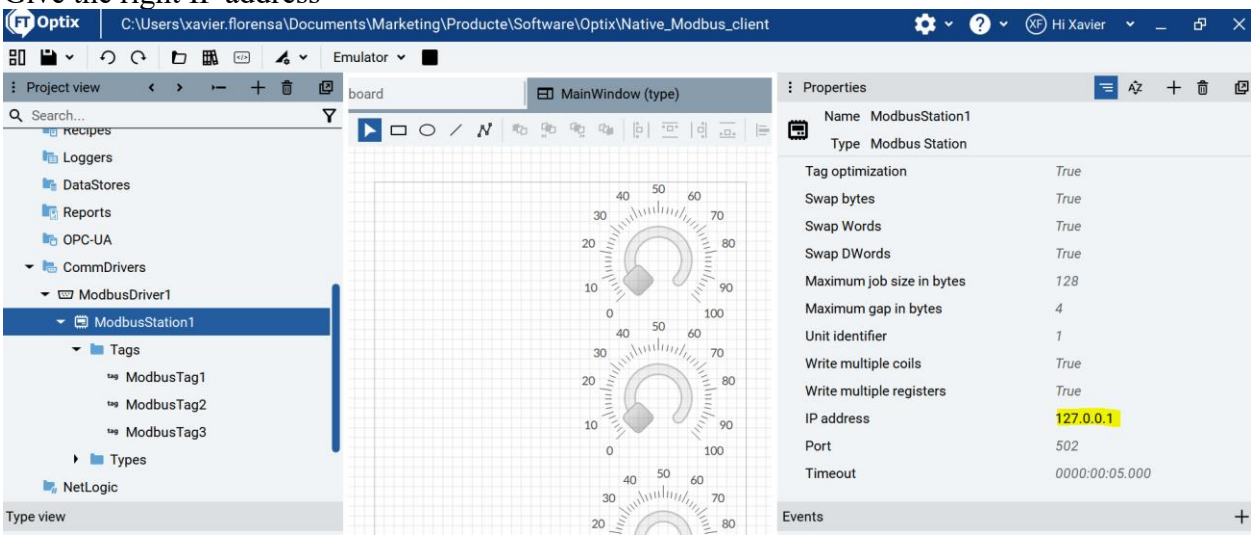
Name: debug 44

48.2. Native Modbus TCP client with FactoryTalk Optix

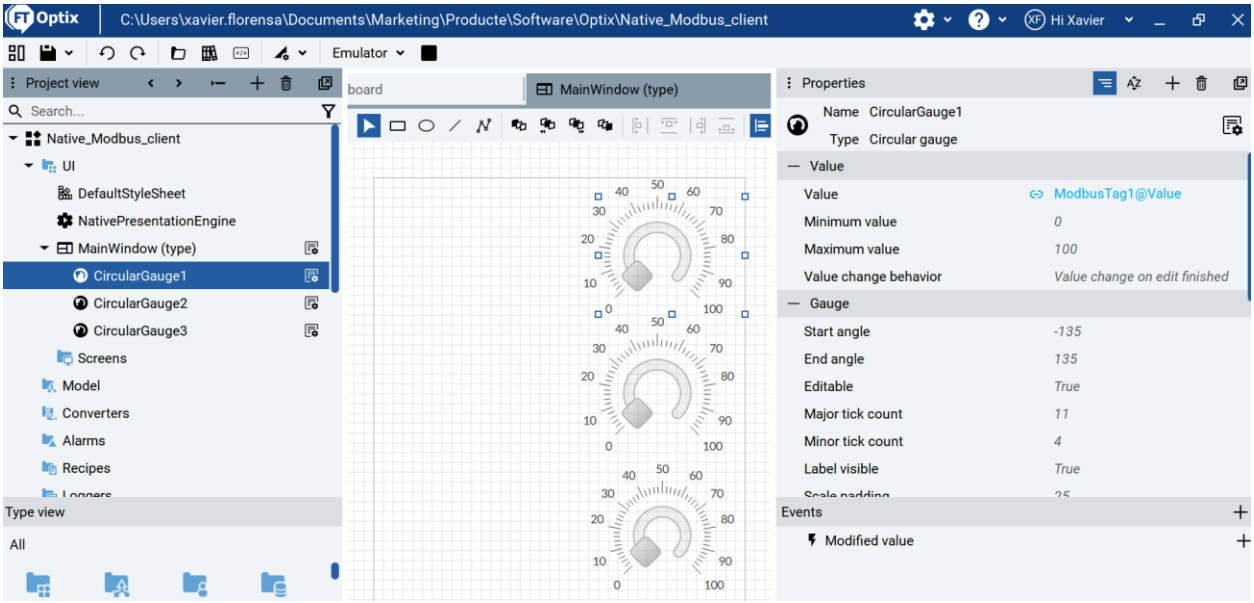
You can see it on this video
https://youtu.be/UpRB72bkKkA
Since FT OPtix has drivers for Modbus, let's use it to test the above server
Build and application and on Communication drivers add a Modbus driver, and declare three Tags



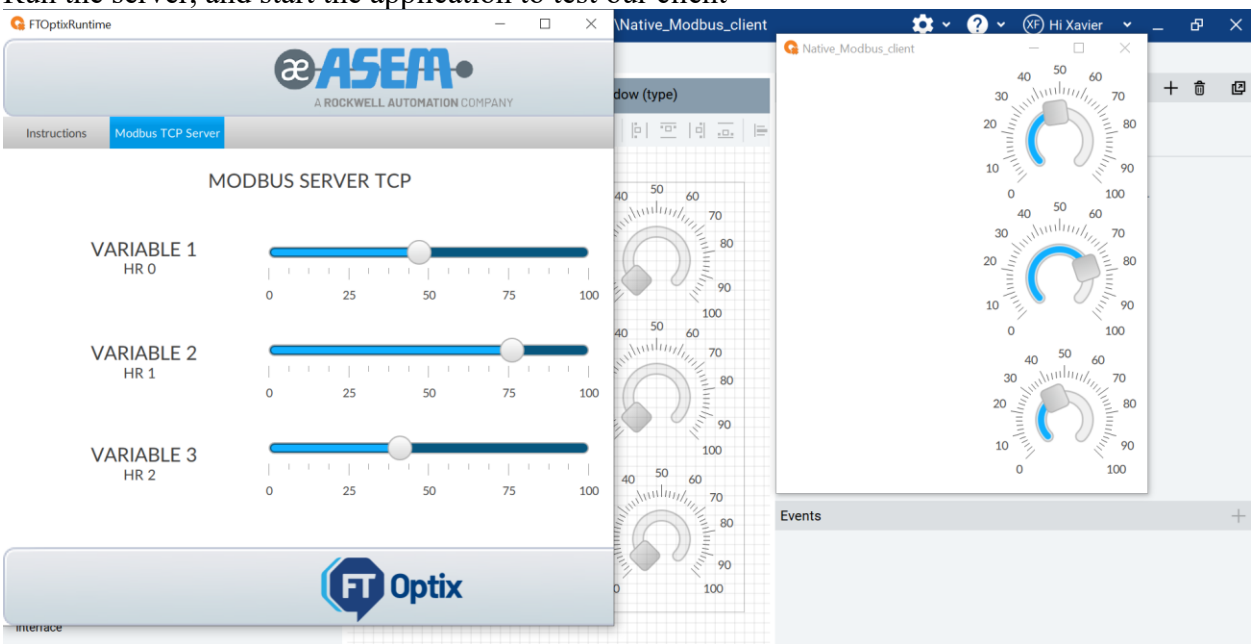
Give the right IP address



Make the corresponding dynamic links between Modbus variables and Values on circular gauges.



Run the server, and start the application to test our client



Here you can find the code
https://github.com/xavierflorensa/Optix_Native_Modbus_client

48.3. DotNet ModbusTCP client with C# and FactoryTalk Optix

Let's try to build a Modbus client application with FT Optix and the EasyModbus library
Here the dll library EasyModbusTCP is used
https://www.nuget.org/packages/EasyModbusTCP
You can see how to use this library (with a client)
https://www.youtube.com/watch?v=qcyJoDu7cok&t=726s
You can see the final result on this video
https://youtu.be/UpRB72bkKkA

Copy the two dll from the FTOptix example

This PC > Documents > Marketing > Producte > Software > Optix > ModbusServerTCP > ProjectFiles > NetSolution > bi

Name	Date modified	Type	Size
runtimes	1/15/2024 7:36 PM	File folder	
EasyModbus.dll	12/31/2020 2:44 PM	Application extension	71 KB
ModbusServerTCP.deps	1/15/2024 8:33 PM	JSON Source File	7 KB
ModbusServerTCP.dll	1/15/2024 8:33 PM	Application extension	7 KB
ModbusServerTCP	1/15/2024 8:33 PM	PDB File	15 KB
System.IO.Ports.dll	10/18/2022 6:29 PM	Application extension	37 KB

And locate anywhere in your new client project

This PC > Documents > Marketing > Producte > Software > Optix > ModbusClientTCP > ProjectFiles

Name	Date modified	Type	Size
NetSolution	1/15/2024 8:43 PM	File folder	
PKI	1/15/2024 8:31 PM	File folder	
EasyModbus.dll	12/31/2020 2:44 PM	Application extension	71 KB
System.IO.Ports.dll	10/18/2022 6:29 PM	Application extension	37 KB
UserDefinedModule	1/15/2024 8:42 PM	Microsoft Edge HTM...	1 KB

Create a new Runtime Netlogic

<u>Installing libraries with Visual Studio 2022</u>

change editor to Visual Studio 2022

Open code and go to

Project / Add references

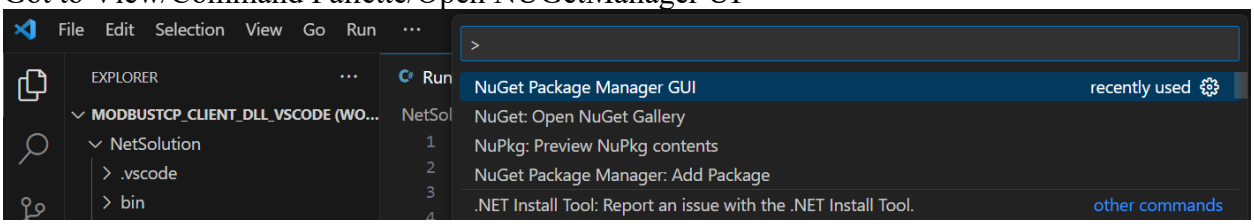
Browse and install the two dll one after the other.

Close Visual Studio.

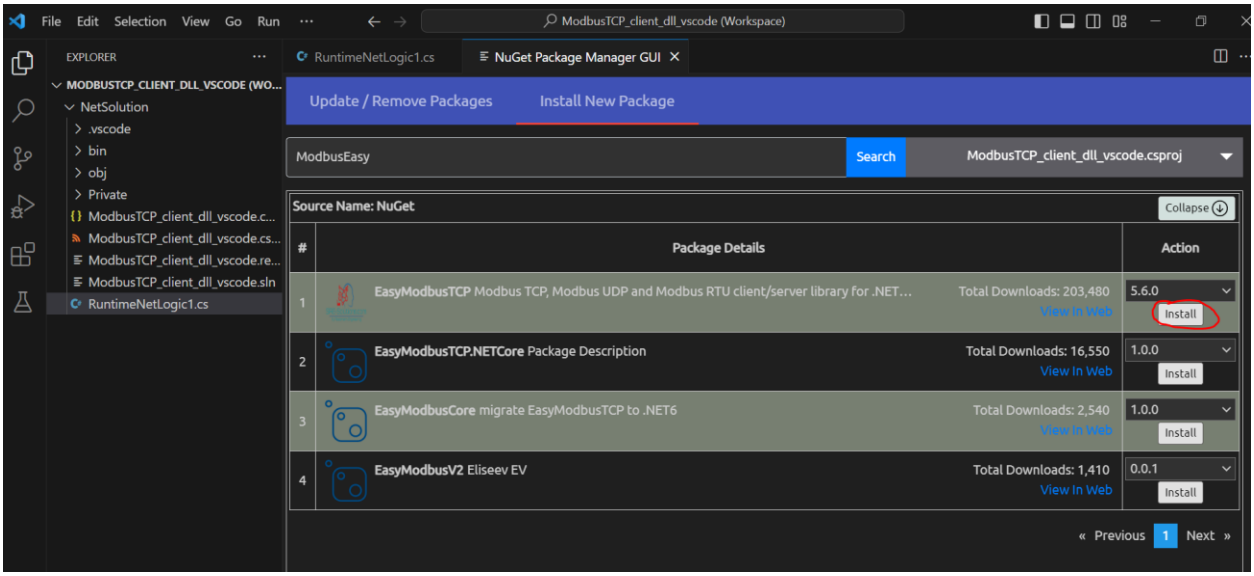
<u>Installing libraries with Visual Studio Code</u>

You can also do this on VScode, this way

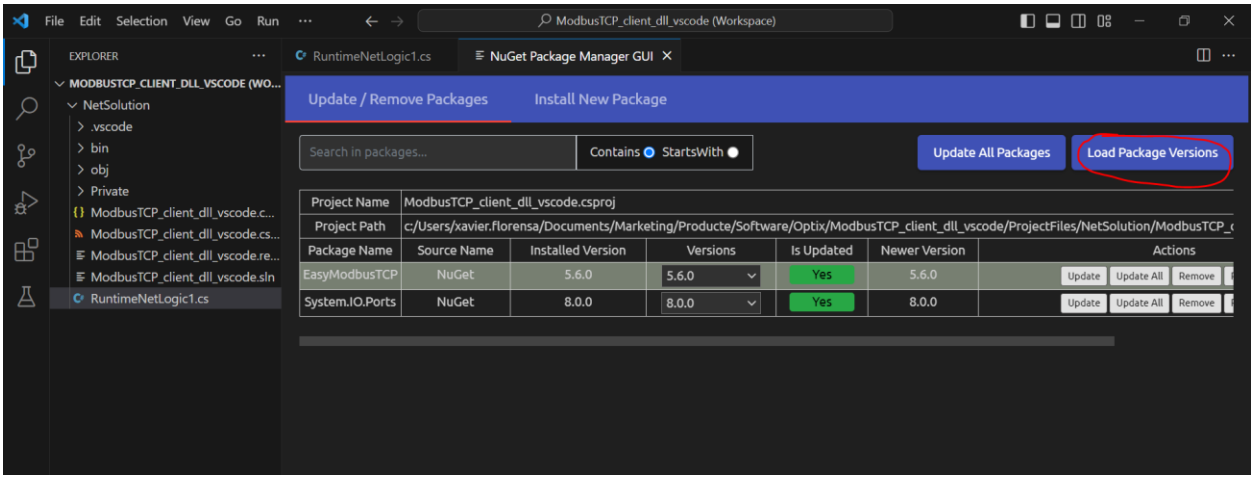
Got to View/Command Pallette/Open NUGetManager UI



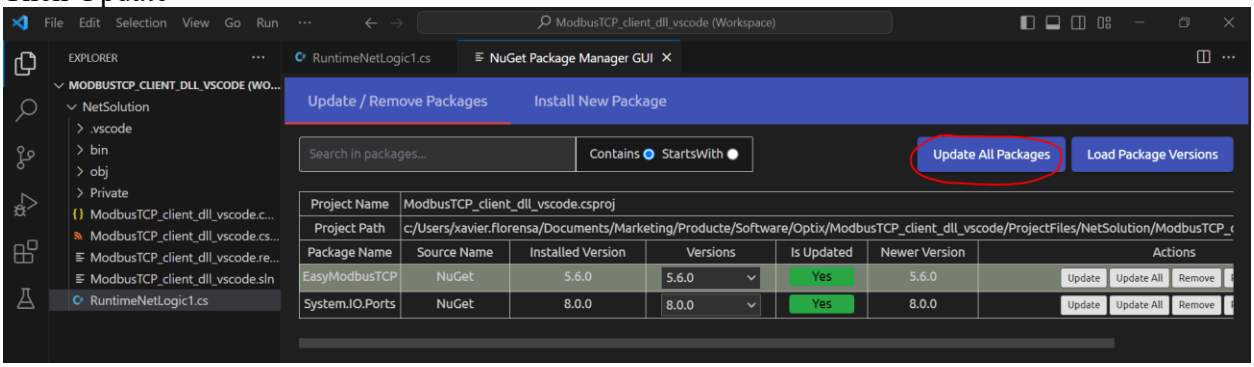
Search for ModbusEasy. Install



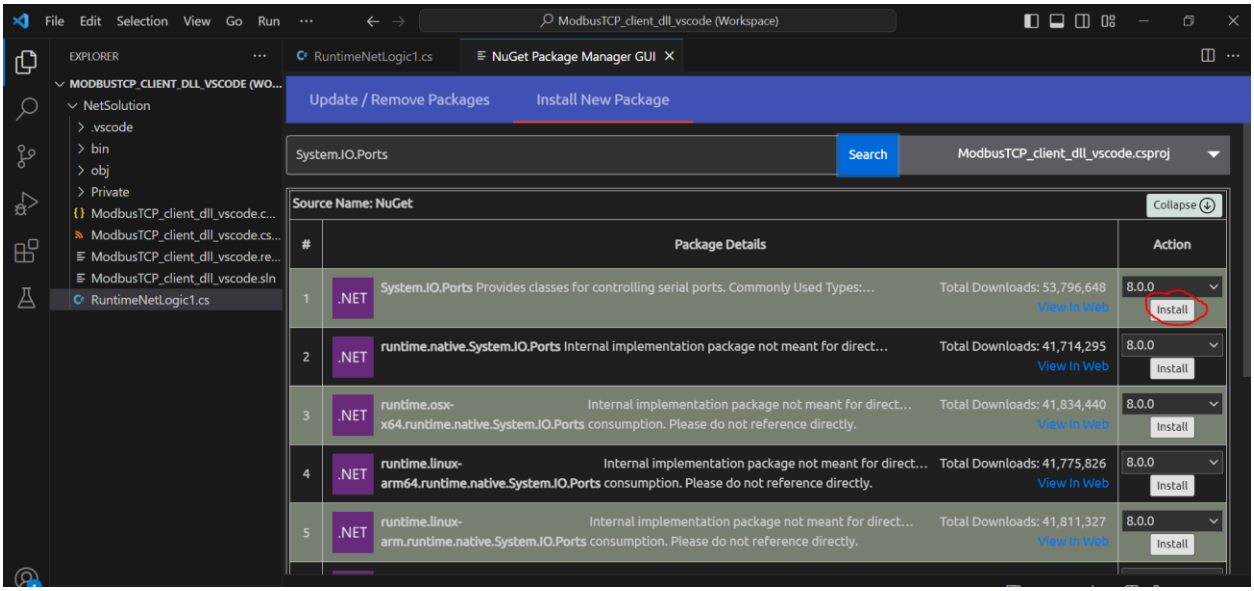
Then go to Update
Load



Then Update



Install also System.IO.Ports



Then Load and update
Open the code,
Look at the *.csproj file

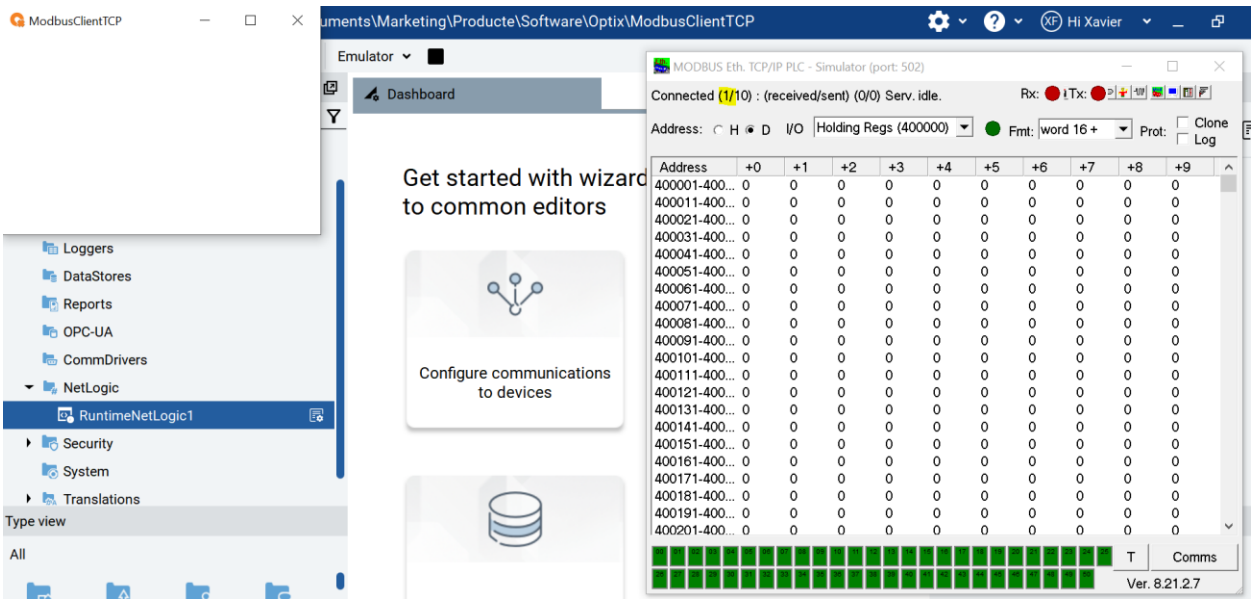
```
<Reference Include="EasyModbus">
  <HintPath>..\EasyModbus.dll</HintPath>
</Reference>
<Reference Include="System.IO.Ports">
  <HintPath>..\System.IO.Ports.dll</HintPath>
</Reference>
```

Now add the using directive
Using EasyModbus;
And this code inside the start procedure

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
```

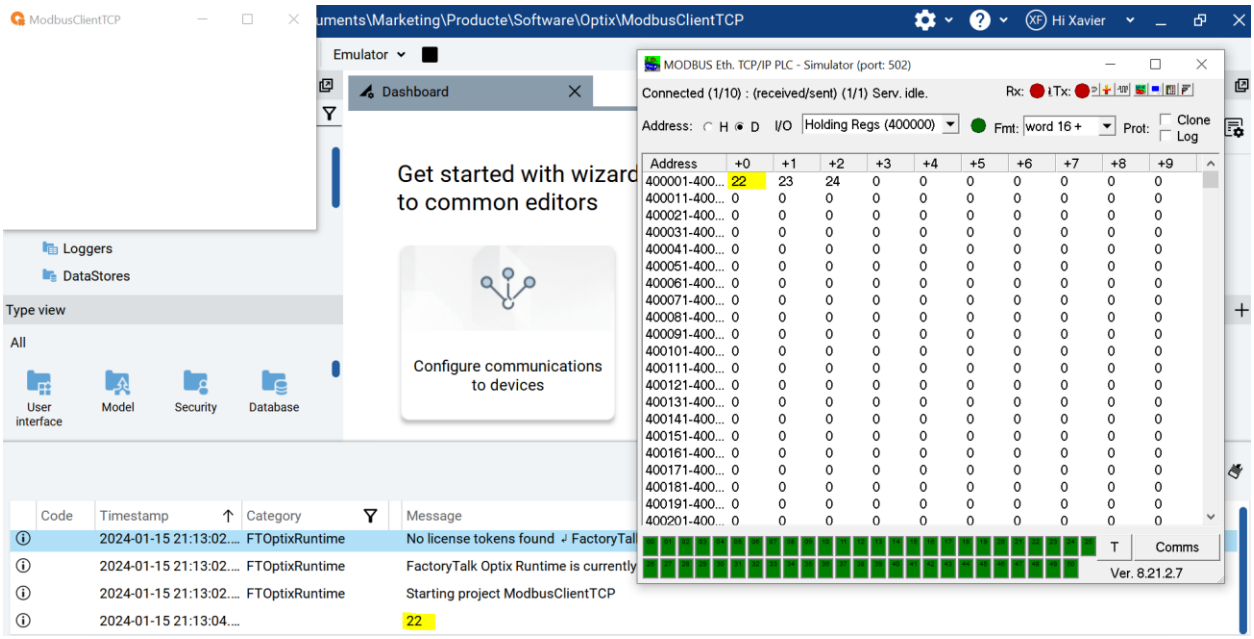
```
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Execute the project
Opening a Server, we are connected to the server



Now let's read a holding register
With this code

```
int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
Log.Info(readHoldingRegisters[0].ToString());
```



This is the code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        // Insert code to be executed when the user-defined logic is started
    }

    public override void Stop()
    {
```

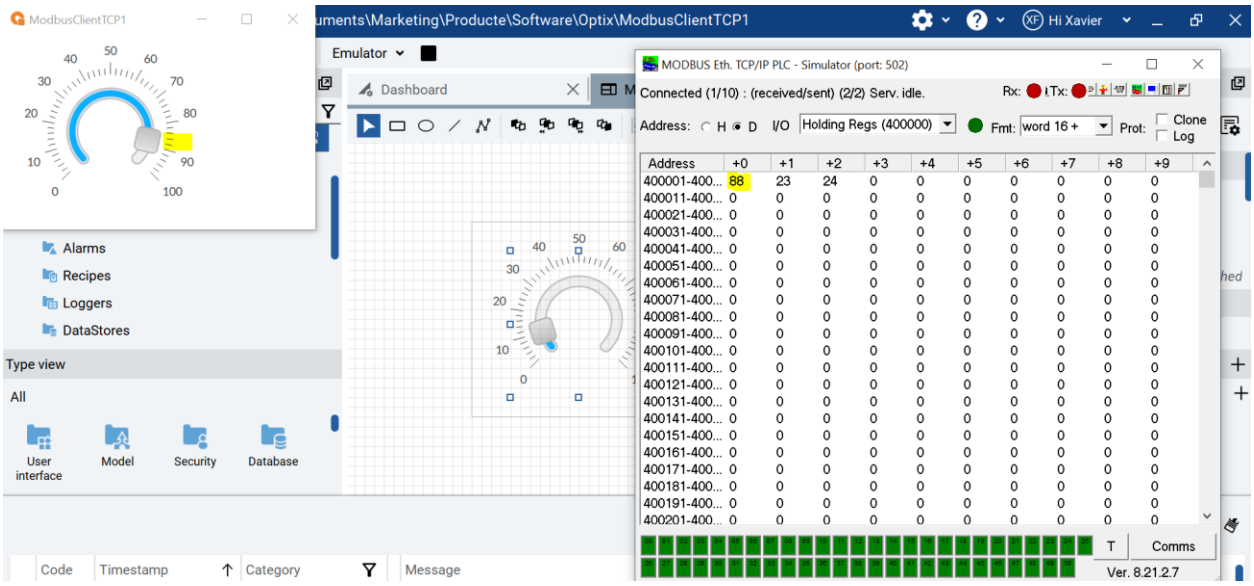
```
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```

Let's use a dashboard,
Add a new Gauge to your project and use this code

```
 var myGauge =
Project.Current.Get<CircularGauge>("UI/MainWindow/CircularGauge1");
        myGauge.Value=readHoldingRegisters[0];
```

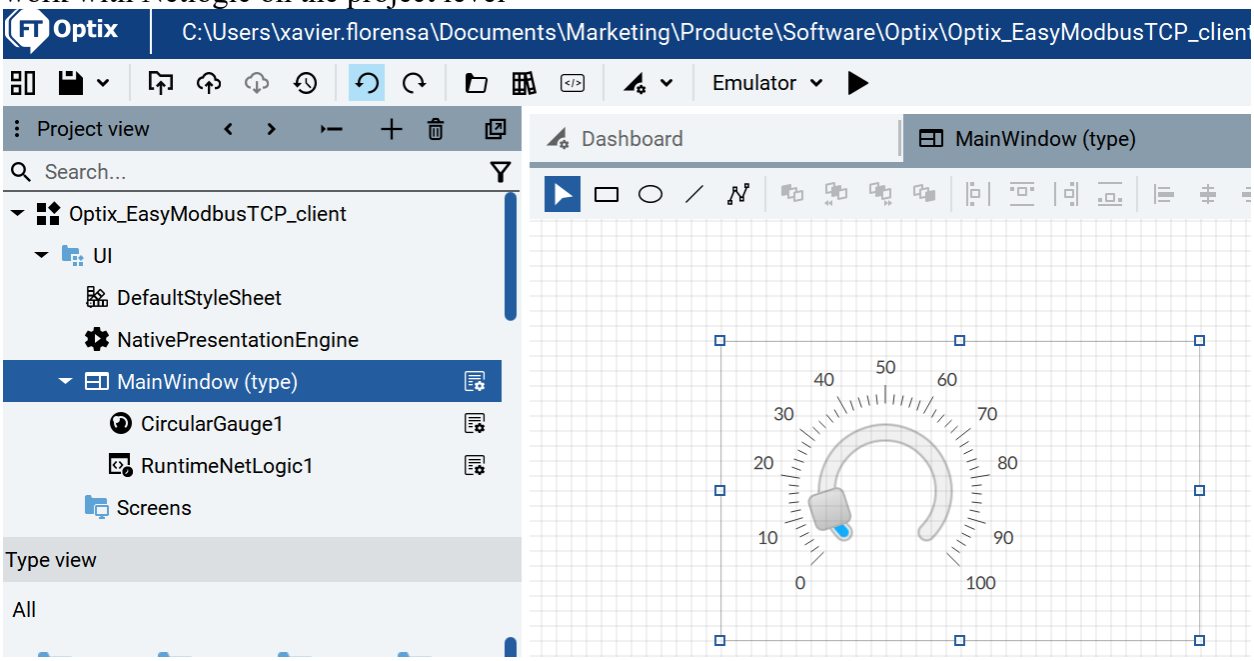
This is the whole code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    public override void Start()
    {
        ModbusClient myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        // Insert code to be executed when the user-defined logic is started
        var myGauge =
Project.Current.Get<CircularGauge>("UI/MainWindow/CircularGauge1");
        myGauge.Value=readHoldingRegisters[0];
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
    }
}
```



But we need a periodic task to update the value

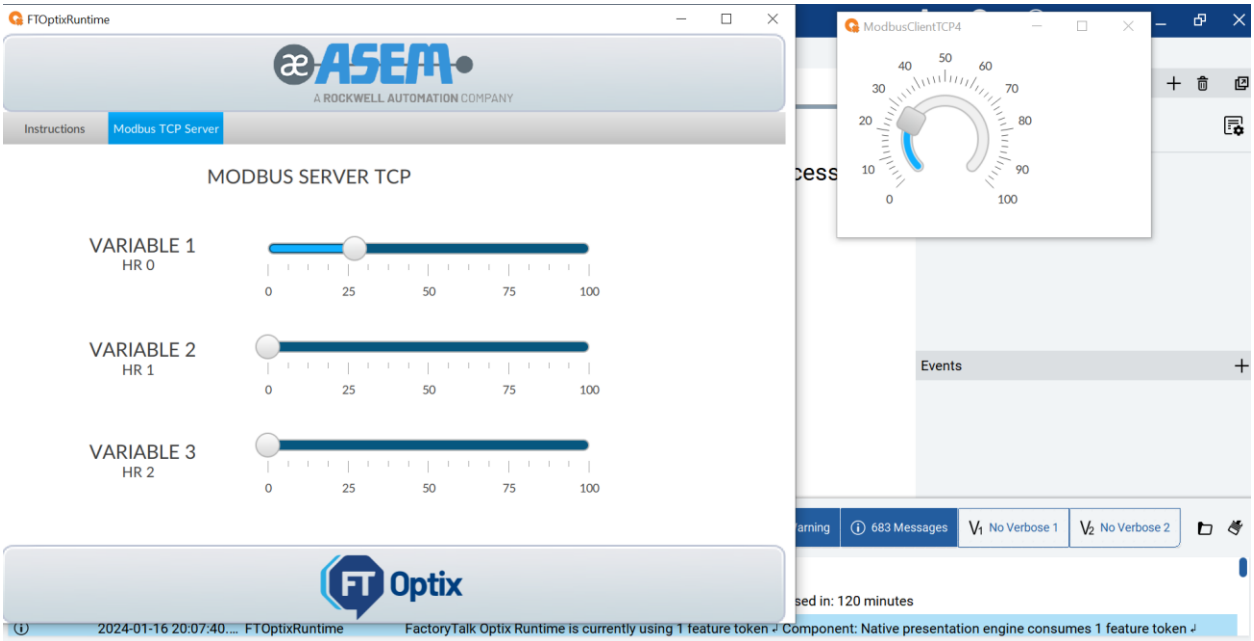
First of all, delete the former Netlogic and create a Netlogic under the Main Window (it will not work with Netlogic on the project level



Use this logic and test the project

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.NetLogic;
```

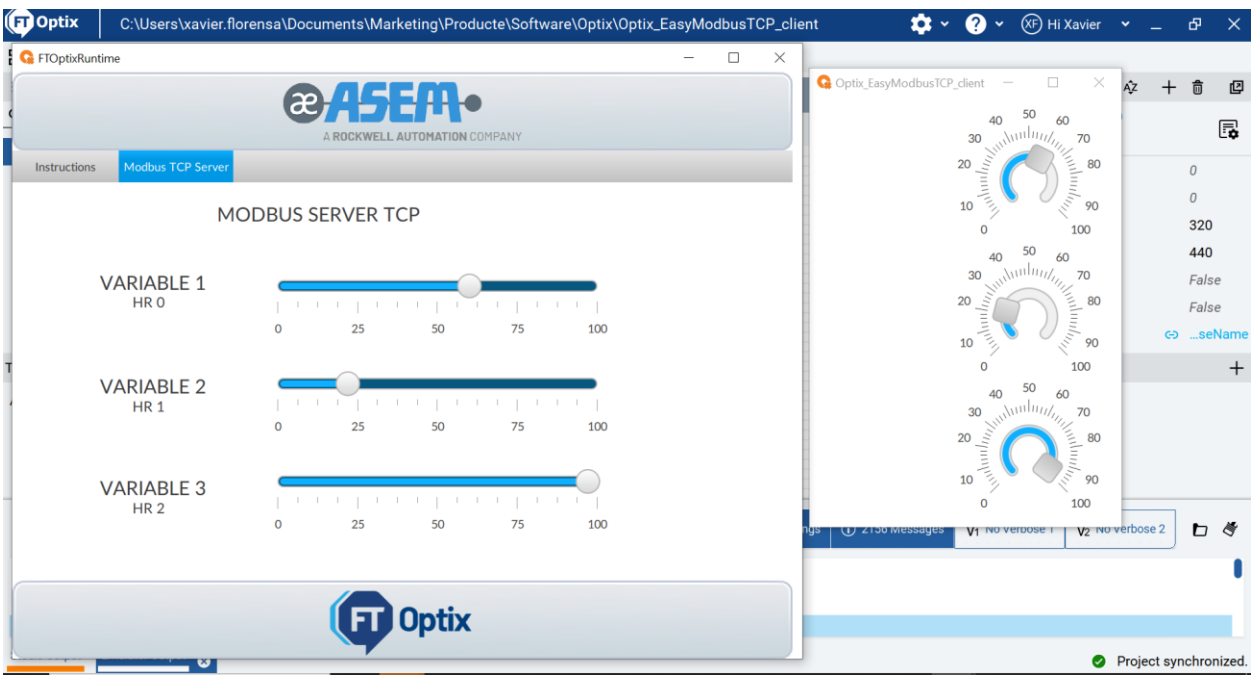
```
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
     PeriodicTask myTask;
    ModbusClient myModbusClient;
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        myTask = new PeriodicTask(MB_Server_Task, 100, LogicObject);
        myTask.Start();
    }

    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
        myTask.Dispose();
    }
    private void MB_Server_Task()
    {
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,2);
        Log.Info(readHoldingRegisters[0].ToString());
        Owner.Get<CircularGauge>("CircularGauge1").Value=readHoldingRegisters[0];
    }
}
```



Now let's modify the program to show three values

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.NetLogic;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using EasyModbus;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
     PeriodicTask myTask;
    ModbusClient myModbusClient;
    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        myModbusClient = new ModbusClient("127.0.0.1",502);
        myModbusClient.Connect();
        myTask = new PeriodicTask(MB_Server_Task, 100, LogicObject);
        myTask.Start();
    }
```

```
    public override void Stop()
    {
        // Insert code to be executed when the user-defined logic is stopped
        myTask.Dispose();
    }
    private void MB_Server_Task()
    {
        int[] readHoldingRegisters = myModbusClient.ReadHoldingRegisters(0,3);
        Owner.Get<CircularGauge>("CircularGauge1").Value=readHoldingRegisters[0];
        Owner.Get<CircularGauge>("CircularGauge2").Value=readHoldingRegisters[1];
        Owner.Get<CircularGauge>("CircularGauge3").Value=readHoldingRegisters[2];
    }
}
```

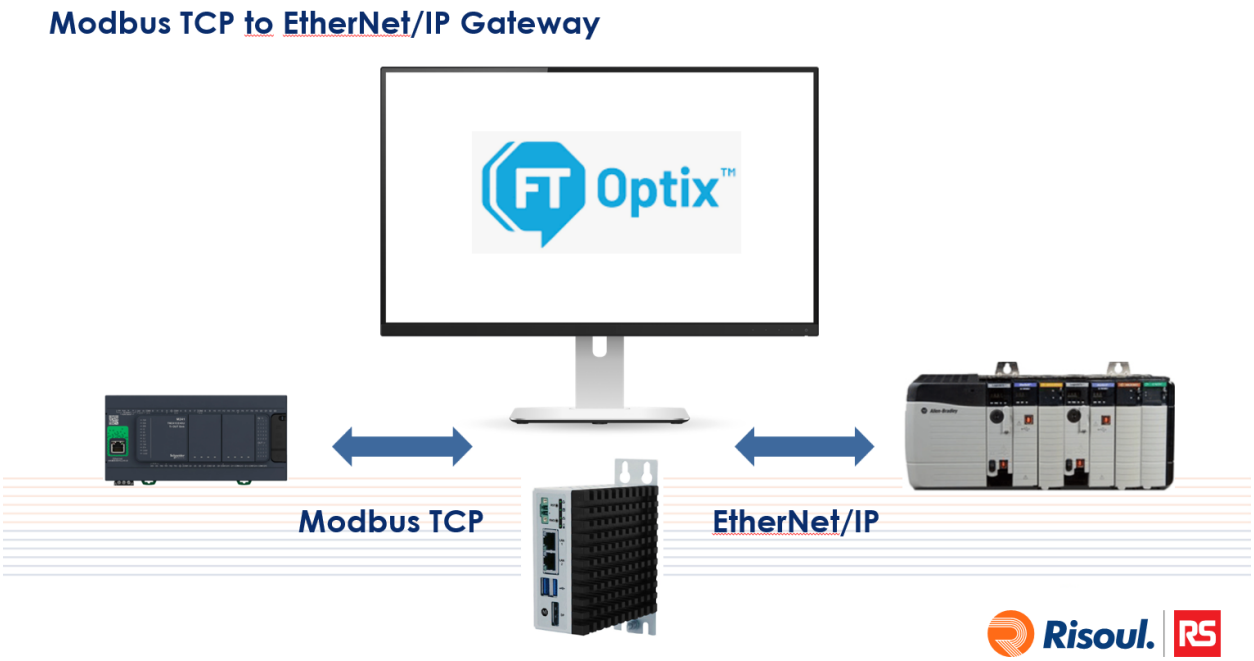


You have the code here
https://github.com/xavierflorensa/Optix_EasyModbusTCP_client

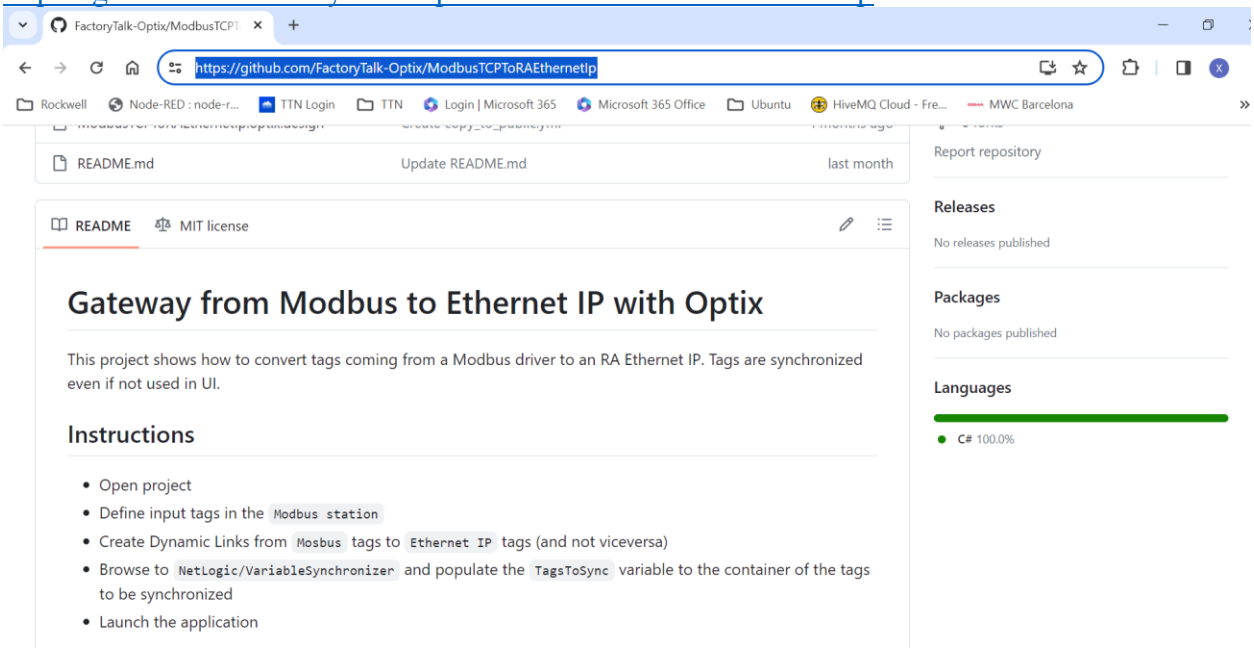
48.4. Modbus TCP to OPC UA converter

You can use Optix instead of Kepserver to do so. And in this case Optix is cheaper!
This can be used to read a Modbus PLC from FactoryTalk View
You can see the steps on this video
https://www.youtube.com/watch?v=WLTmArGiwG8
To be able to use the Tag importer take a look at chapter Design time libraries. Importing Modbus Tags from CSV file

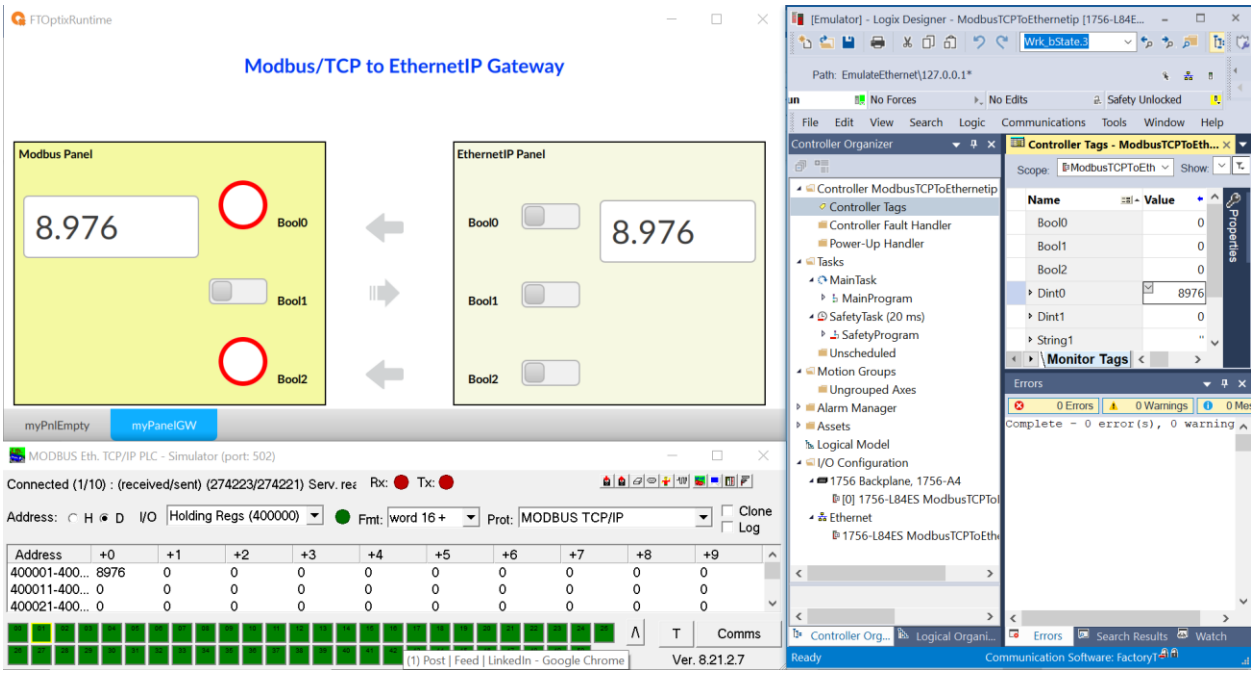
48.5. Modbus TCP to EtherNet/IP Gateway



You can use the following example from Github
https://github.com/FactoryTalk-Optix/ModbusTCPToRAEthernetIp

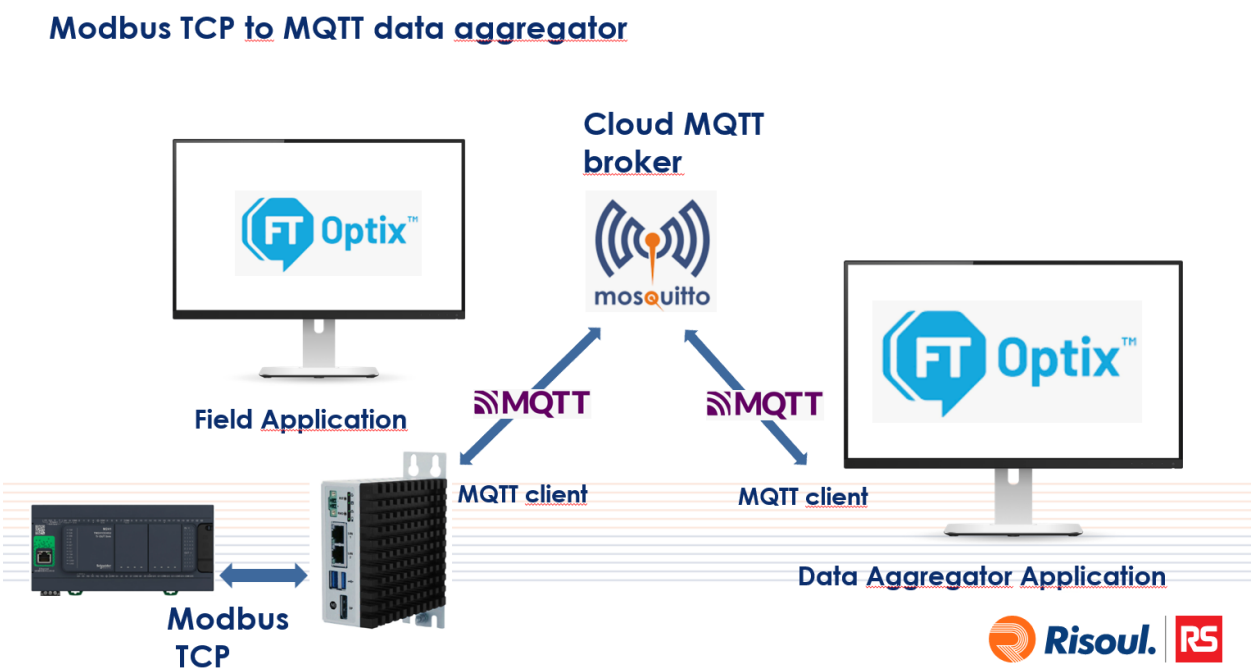


This is the result



As you can see on this video
https://youtu.be/UMlI0TQ1uGM

48.6. Modbus to MQTT data aggregator

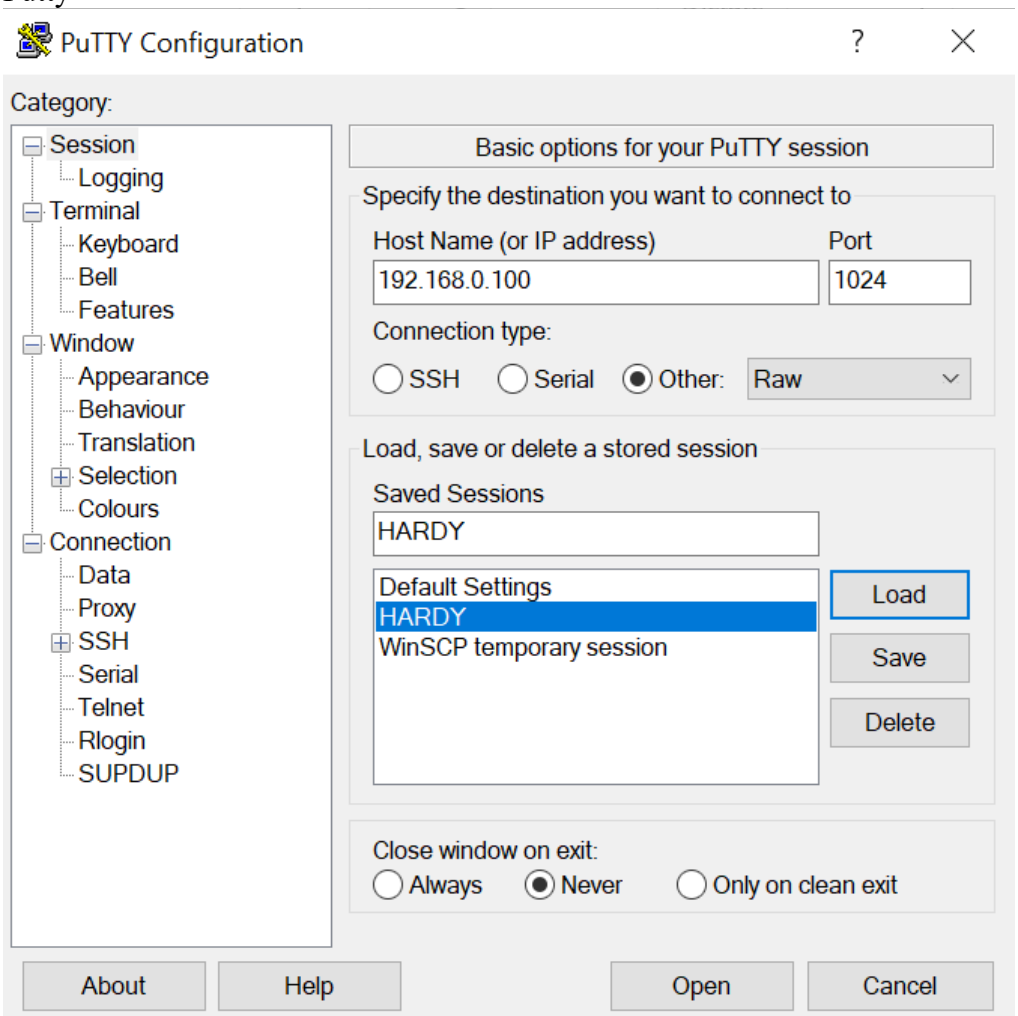


You can see the process here
https://youtu.be/PRgpzygpefA

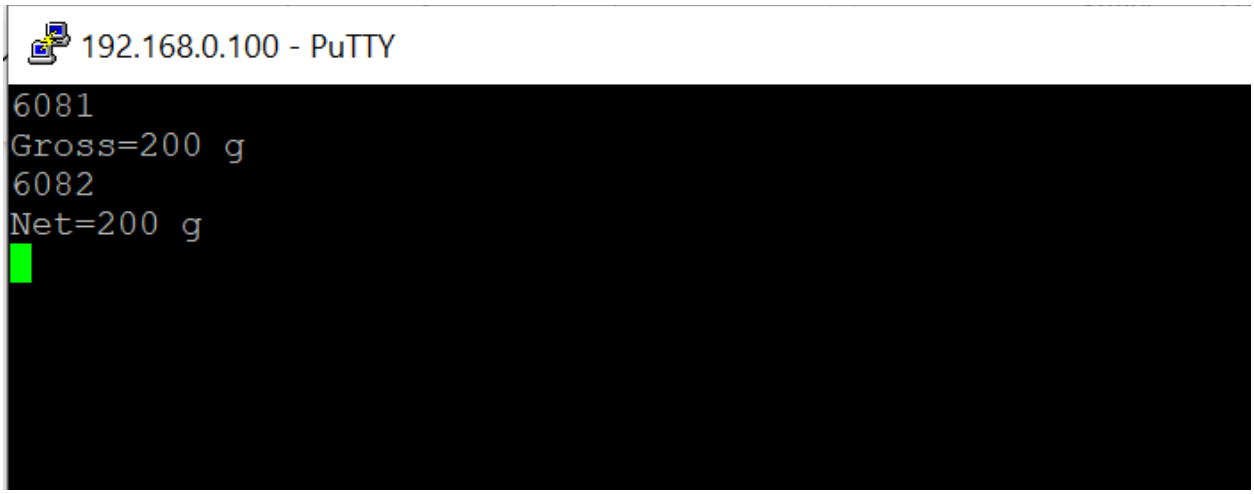
49. Communicating Optix with HARDY 6200 weight processor

49.1. TCP/UDP communications

You can use UDP communications in setup and port 1024
You must have only one program open (Putty or other)
Putty

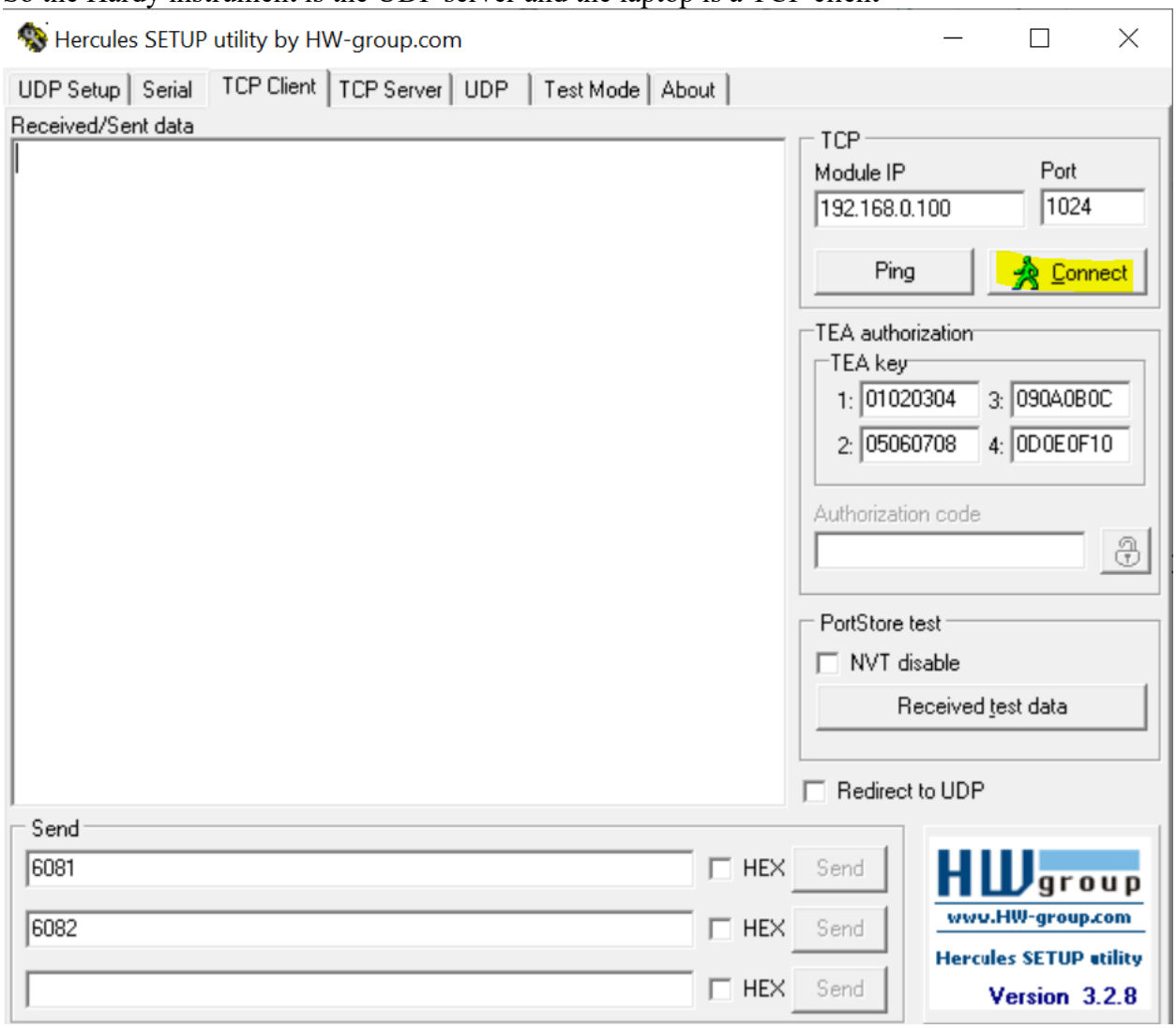


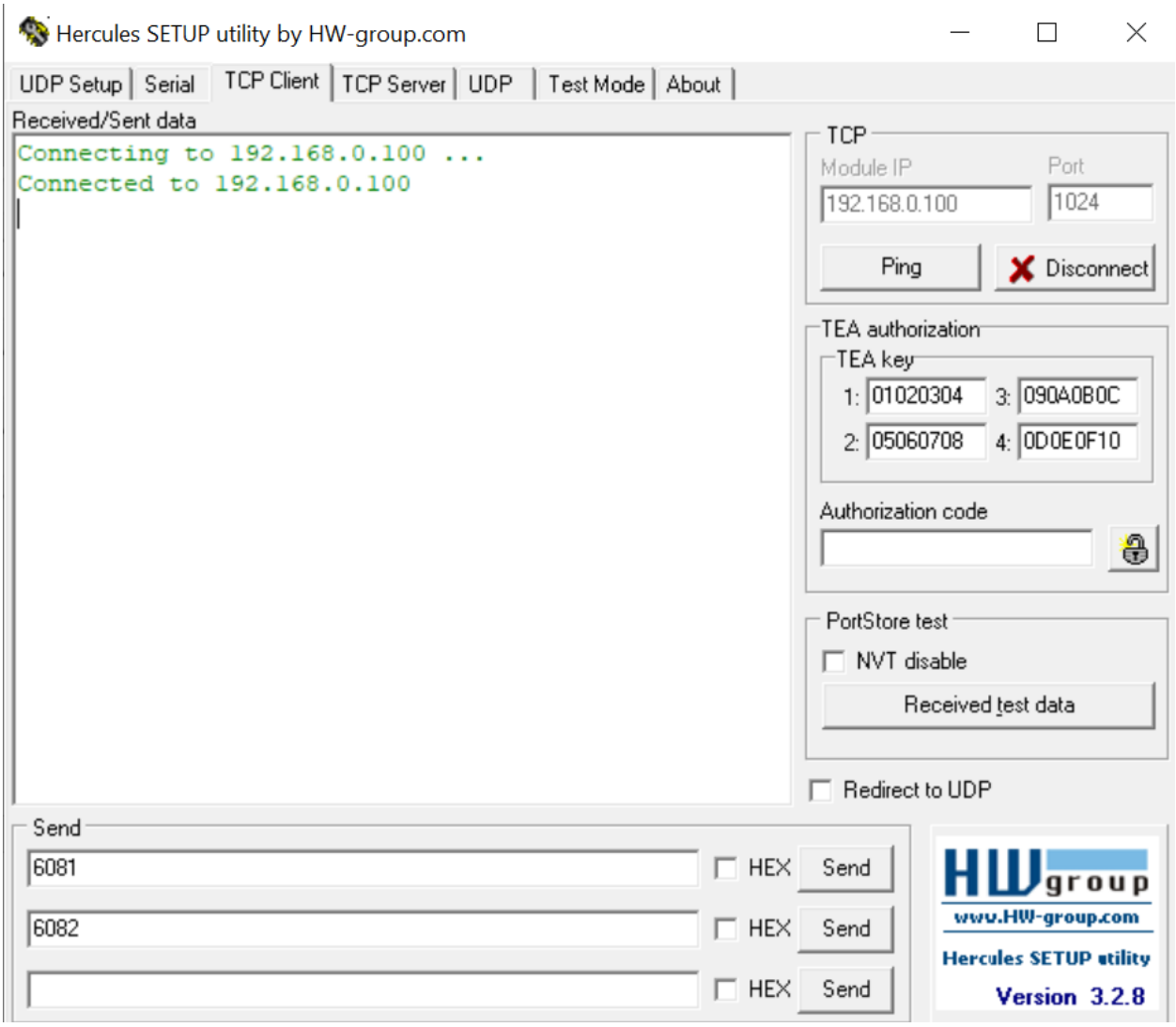
Send a command 6081



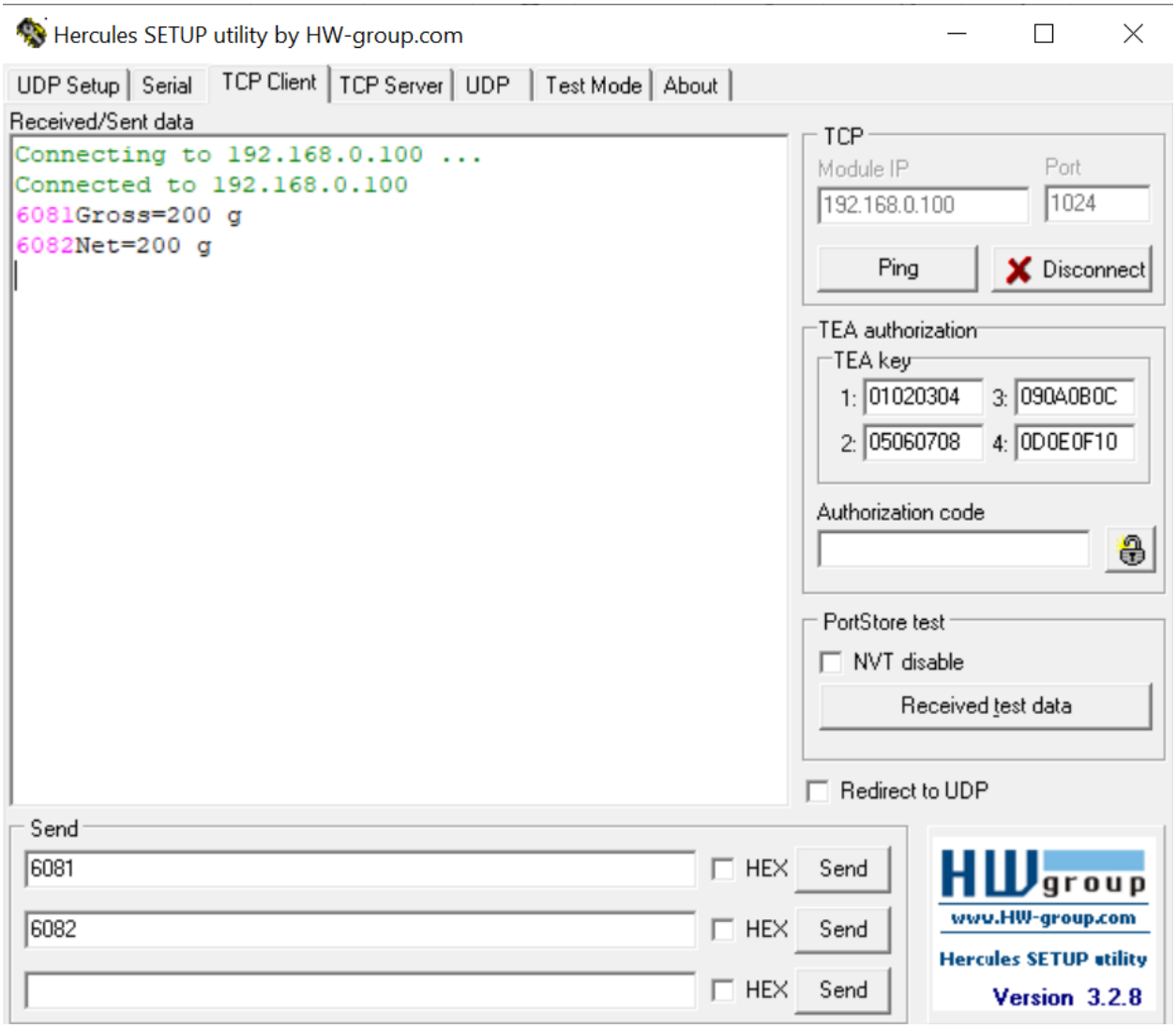
Hercules terminal

So the Hardy instrument is the UDP server and the laptop is a TCP client



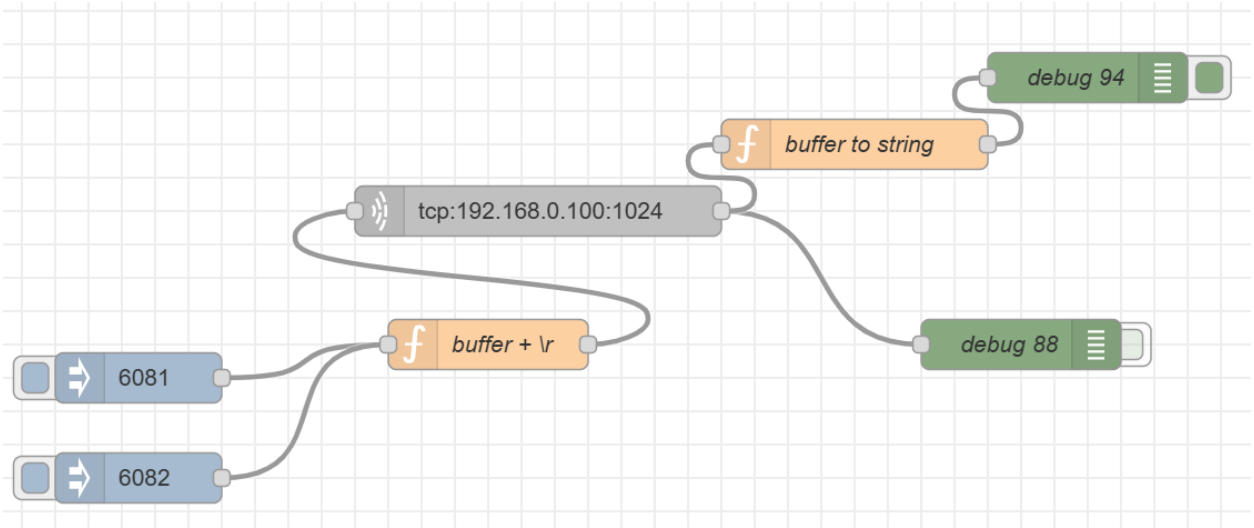


Send a command then hit enter
voilà



49.2. Node-RED

Let's use this flow



Edit inject node

Delete | Cancel | Done

Properties

Name: Name

msg. payload = 6081

msg. topic =

+ add | inject now

Inject once after 0.1 seconds, then

Repeat: none

Enabled

Edit function node

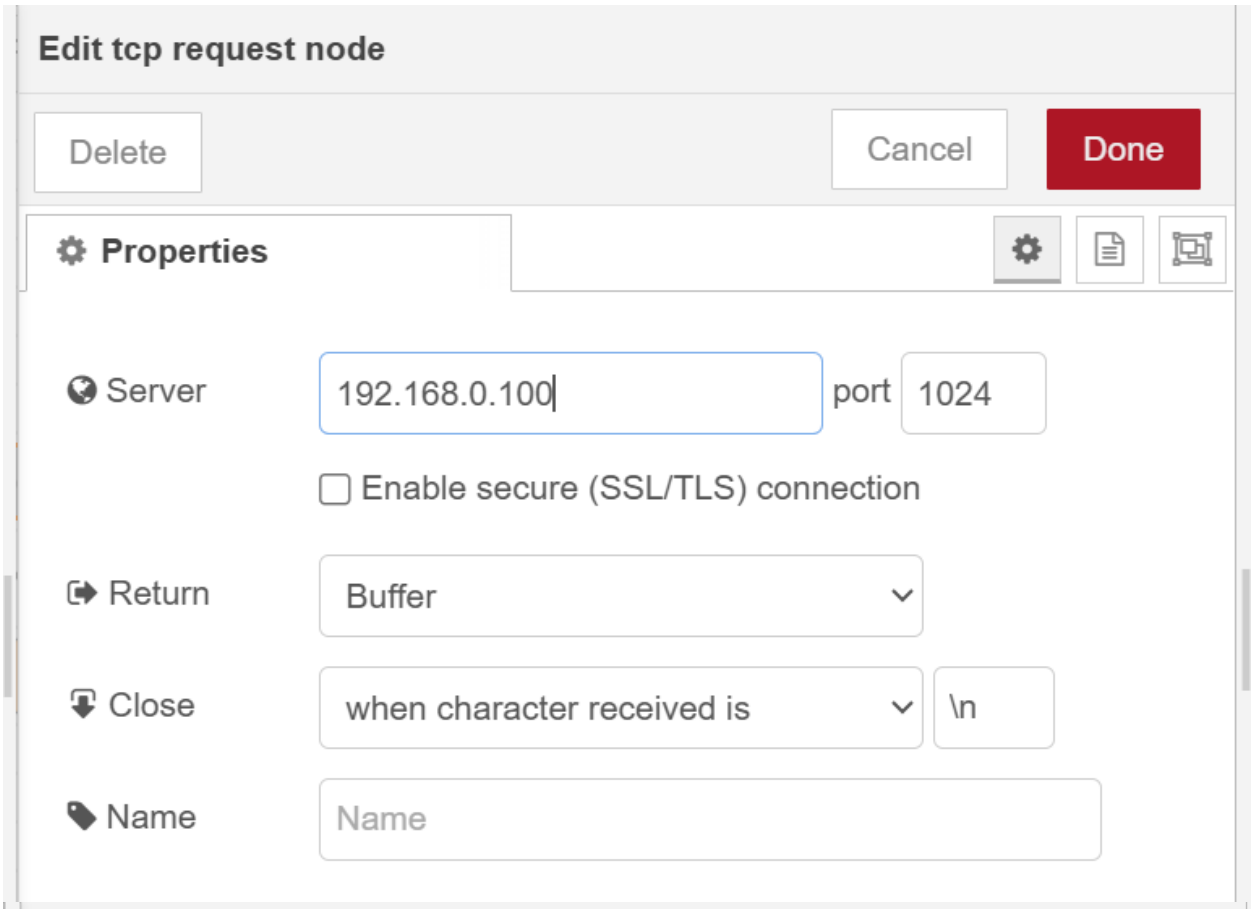
Delete | Cancel | Done

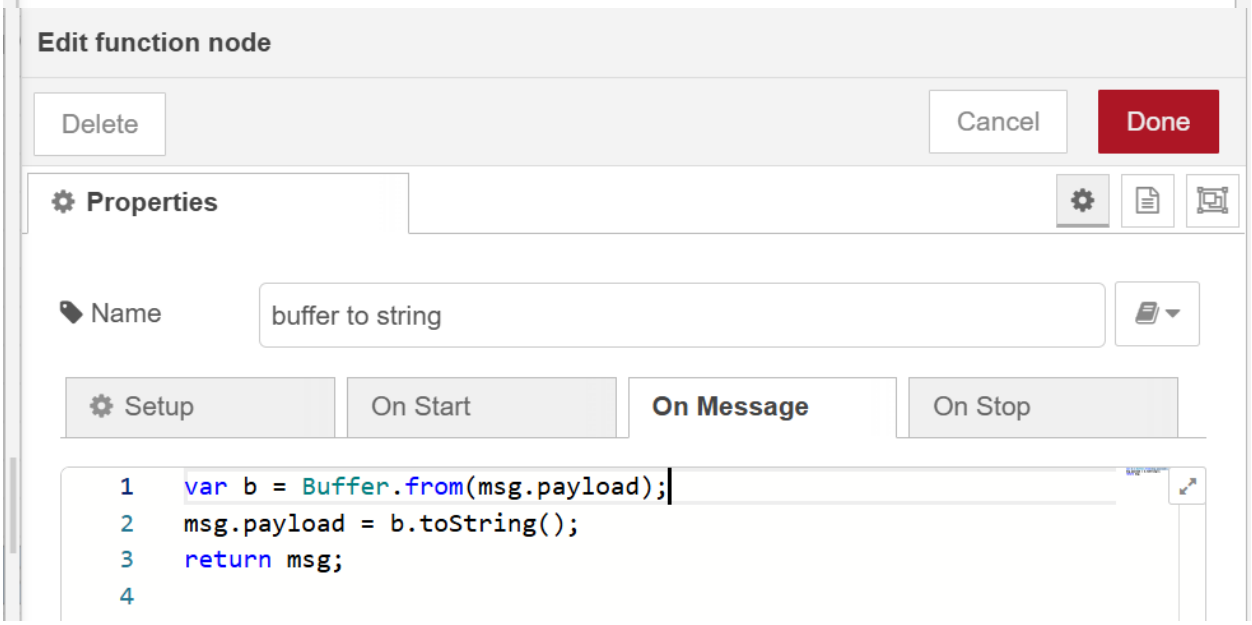
Properties

Name: buffer + \r

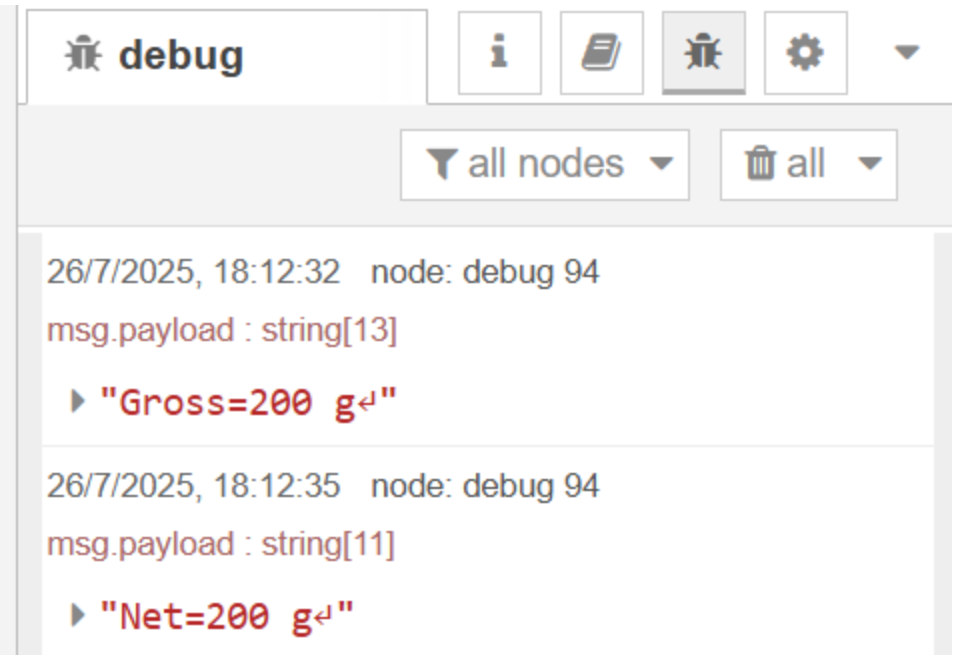
Setup | On Start | **On Message** | On Stop

```
msg.payload = Buffer.from(msg.payload + "\r");
return msg;
```


Edit tcp request node
Delete
Cancel
Done
Properties
Server
192.168.0.100
port
1024
Enable secure (SSL/TLS) connection
Return
Buffer
Close
when character received is
\n
Name
Name


Edit function node
Delete
Cancel
Done
Properties
Name
buffer to string
Setup
On Start
On Message
On Stop
var b = Buffer.from(msg.payload);
msg.payload = b.toString();
return msg;

Voila



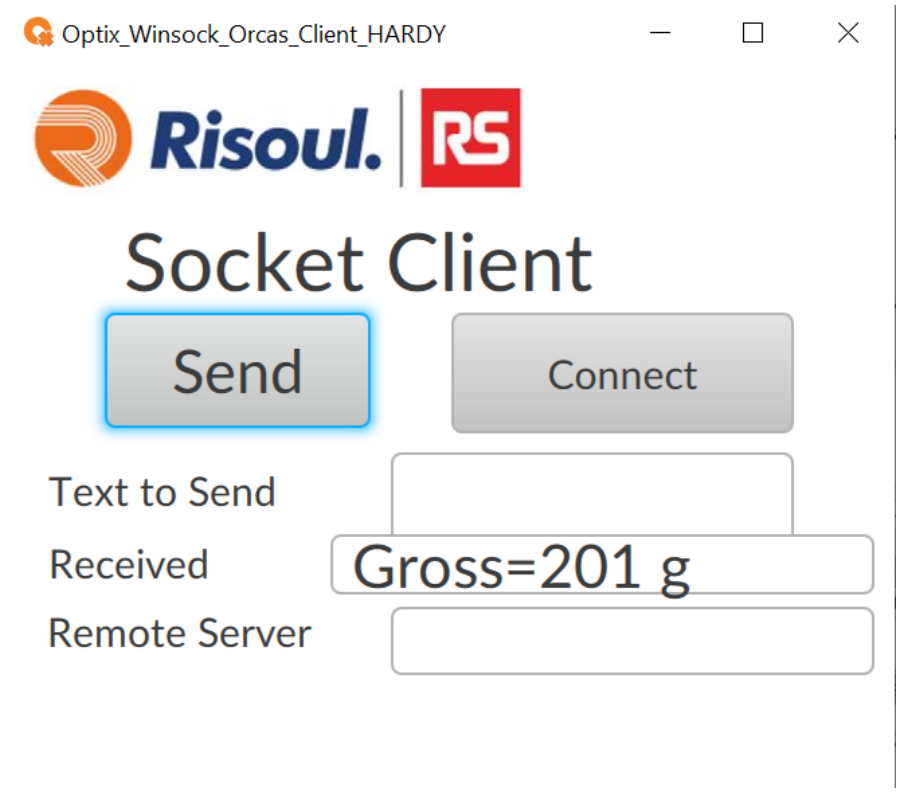
You can see the results in this video
https://www.youtube.com/watch?v=ozY86CZtpGE
You can download the code from here
https://github.com/xavierflorensa/HARDY_weight_reading_in_Node-RED_and_OptixEdge.git

49.3. TCP/IP with FT Optix

Let's introduce the carry return after 6081

```
[ExportMethod]

    public void Send(NodeId textboxNodeId)
    {
        //var text_to_send = InformationModel.Get<TextBox>(textboxNodeId);
        //var text_to_send = "6081";
        var text_to_send = "6081" + "\r";

        //winsock_Ear.Send(Int32.Parse(text_to_send.Text));
        winsock_Ear.Send(text_to_send);
    }
```

Voilà



This is the complete code

```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
using Winsock_Orcas;
using System.Reflection.Emit;
using System.Net.Sockets;
using System.Net.Security;
using System.Net.Http;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{
    Winsock winsock_Ear = new Winsock(); //opens a new socket
```

```
    [ExportMethod]

    public void Send(NodeId textboxNodeId)
    {
        //var text_to_send = InformationModel.Get<TextBox>(textboxNodeId);
        //var text_to_send = "6081";
        var text_to_send = "6081" + "\r";

        //winsock_Ear.Send(Int32.Parse(text_to_send.Text));
        winsock_Ear.Send(text_to_send);
    }

     [ExportMethod]

    public void Connect()
    {
        var remoteIP = Project.Current.GetVariable("Model/RemoteIP");
        winsock_Ear.Connect(remoteIP.Value, 1024);
    }

    public override void Start()
    {
       winsock_Ear.LegacySupport=true;
        //winsock_Ear.Listen(2000);//This is to make the PC act as host
        //winsock_Ear.Connect("127.0.0.1", 2000); //This is to make the PC act as
client
        // Assign a callback to be excuted when the client is connected
        //Winsock_Ear.Connected += winsock_Ear_Connected;
        // Assign a callback to be executed when a message is received from the
server
        winsock_Ear.ConnectionRequest += winsock_Ear_ConnectionRequest;
        // Assign a callback to be executed when data is available from server
        winsock_Ear.DataArrival += winsock_Ear_DataArrival;
        Log.Info("LocalIP: "+winsock_Ear.LocalIP[0]);
        Log.Info("protocol: "+winsock_Ear.Protocol.ToString());
        Log.Info("Legacy support: "+winsock_Ear.LegacySupport.ToString());
        Log.Info("LocalPort: "+winsock_Ear.LocalPort.ToString());
        Log.Info("Remote Host: "+winsock_Ear.RemoteHost.ToString());
        Log.Info("Remote Port: "+winsock_Ear.RemotePort.ToString());
        Log.Info("State: "+winsock_Ear.State.ToString());
        Log.Info("Hello World");
        string text_to_send = "Hello World";
        winsock_Ear.Send(text_to_send);
    }
```

```
    public override void Stop()
    //public override void Stop()
    {
         Log.Info("Stopping");
    }
    //this is the callback function that will execute when a client is connected
    private void winsock_Ear_Connected(object sender,
Winsock_Orcas.WinsockConnectedEventArgs e)
        {
            Log.Info("¡¡Conectado!!");
        }
    //this is the callback function that will execute after the first connection
request
    private void winsock_Ear_ConnectionRequest(object sender,
Winsock_Orcas.WinsockConnectionRequestEventArgs e)
        {
            winsock_Ear.Close();
            winsock_Ear.Accept(e.Client);
        }
    private void winsock_Ear_DataArrival(object sender,
Winsock_Orcas.WinsockDataArrivalEventArgs e)
        {
            string abRecibidos = winsock_Ear.Get<string>();
            var missatge = Project.Current.GetVariable("Model/Message");
            missatge.Value=abRecibidos;
            //missatge.Value=abRecibidos.ToString('ascii');
           Log.Info("recibido: "+ abRecibidos);
           //Log.Info("recibido: "+ abRecibidos.ToString('ascii'));
        }
    private Winsock Winsock_Ear;
 }
```

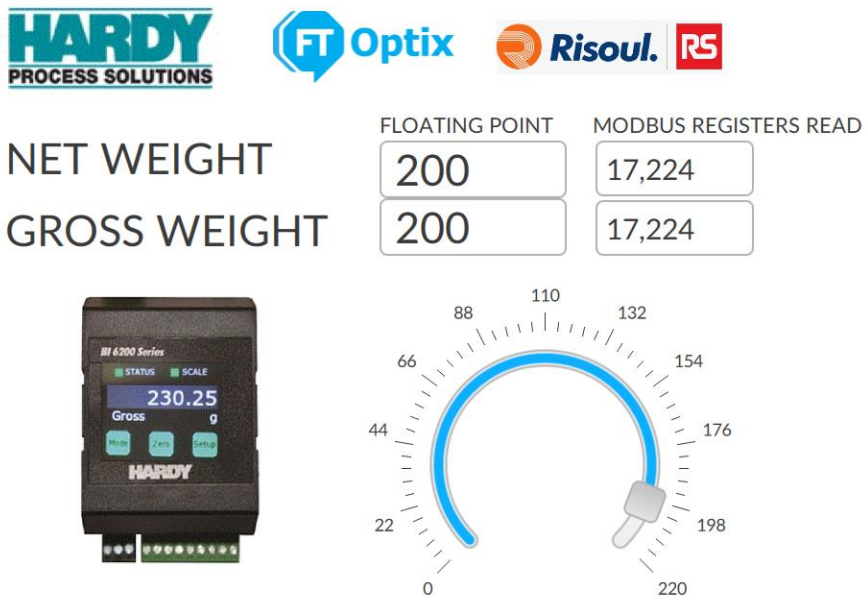
But this is sending always the constant string 6081

Let's send a variable string we get from keyboard input

You can see the results in this video
https://www.youtube.com/watch?v=YVblgQ-ySQ0
You can get the code here
https://github.com/xavierflorensa/FactoryTalk_Optix_TCP_Client_HARDY

49.4. Modbus TCP

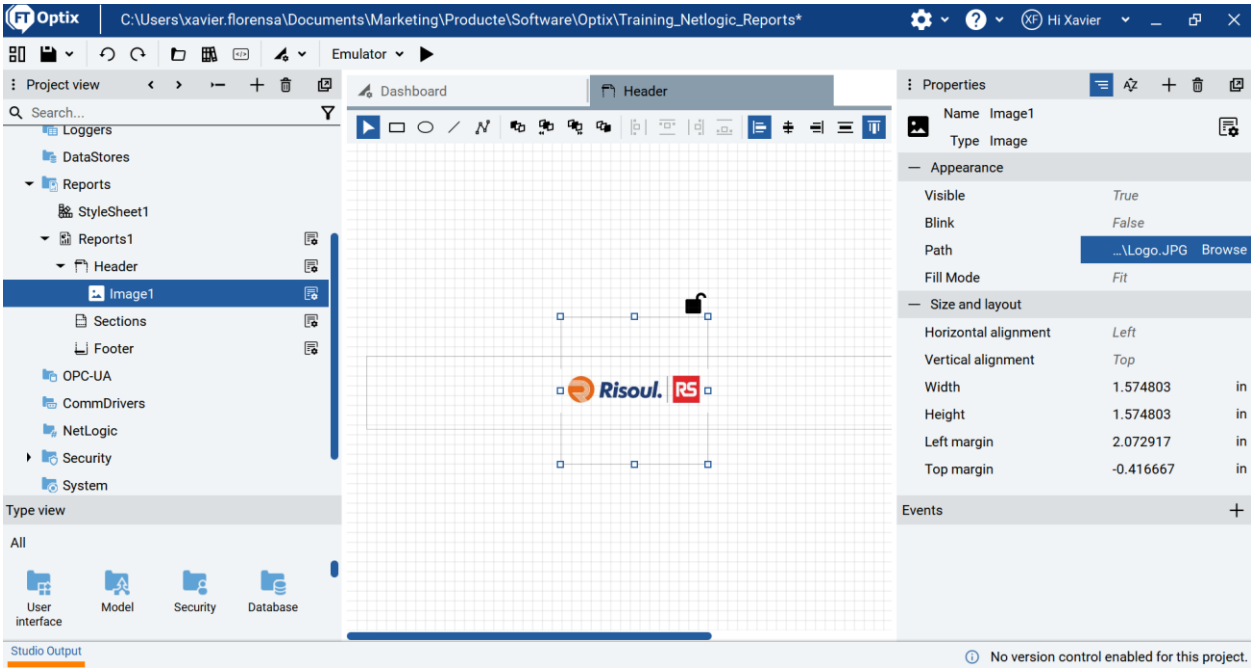
HARDY_Modbus14



You can see the results in this video
https://www.youtube.com/watch?v=4vOtRJDEYEE
You can find the code here
https://github.com/xavierflorensa/FactoryTalk_Optix_HARDY_6200_Modbus

50. Generating a pdf file from C#

Create a Report design with a Logo



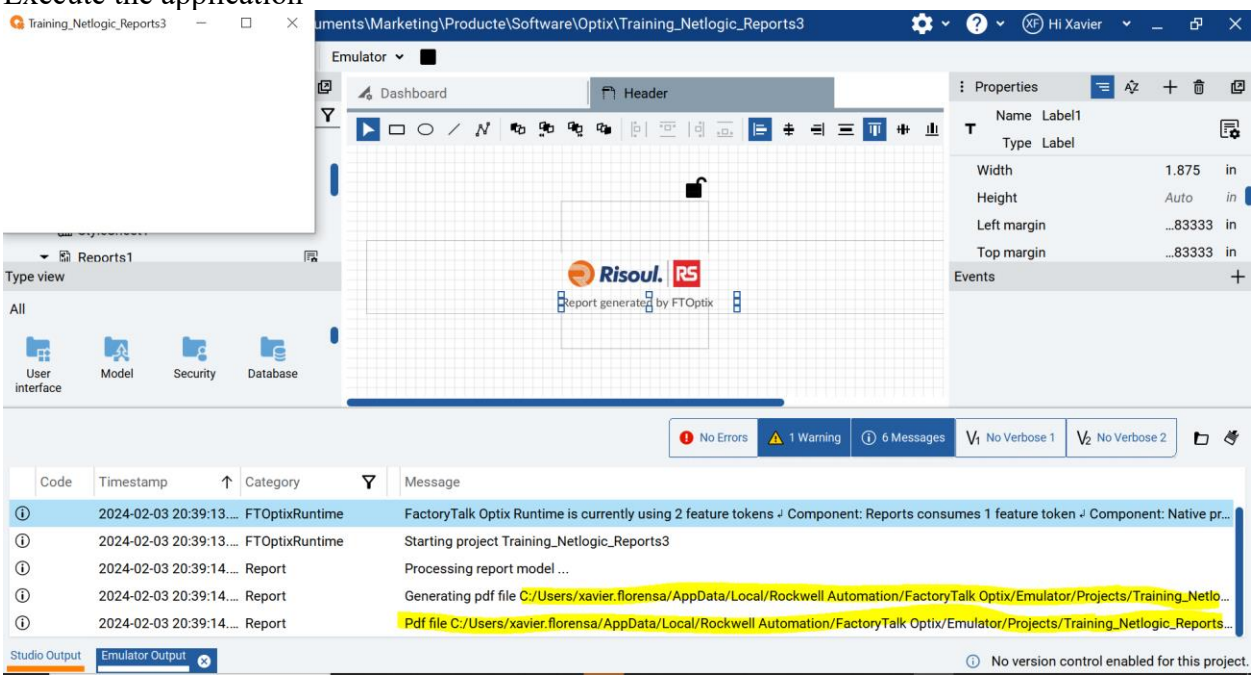
Add a Netlogic Script

Populate with this code where you will insert the name of the file to be generated

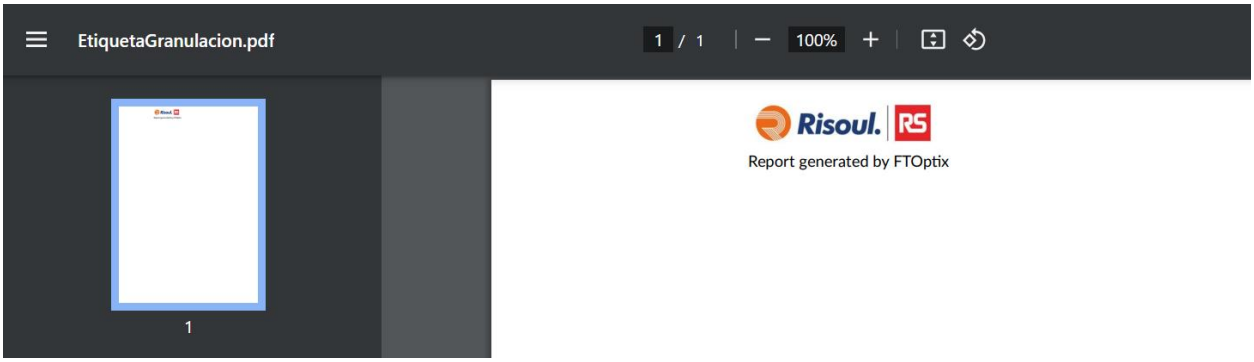
```
#region Using directives
using System;
using UAManagedCore;
using OpcUa = UAManagedCore.OpcUa;
using FTOptix.HMIProject;
using FTOptix.Retentivity;
using FTOptix.Report;
using FTOptix.UI;
using FTOptix.NativeUI;
using FTOptix.CoreBase;
using FTOptix.Core;
using FTOptix.NetLogic;
#endregion

public class RuntimeNetLogic1 : BaseNetLogic
{

    public override void Start()
    {
        // Insert code to be executed when the user-defined logic is started
        String[] Parametros = { "%APPLICATIONDIR%/EtiquetaGranulacion.pdf", "" };
        Project.Current.GetObject("Reports/Reports1").ExecuteMethod("GeneratePdf"
, Parametros);
    }

    public override void Stop()
```

```
    {
        // Insert code to be executed when the user-defined logic is stopped
    }

}
```

Execute the application



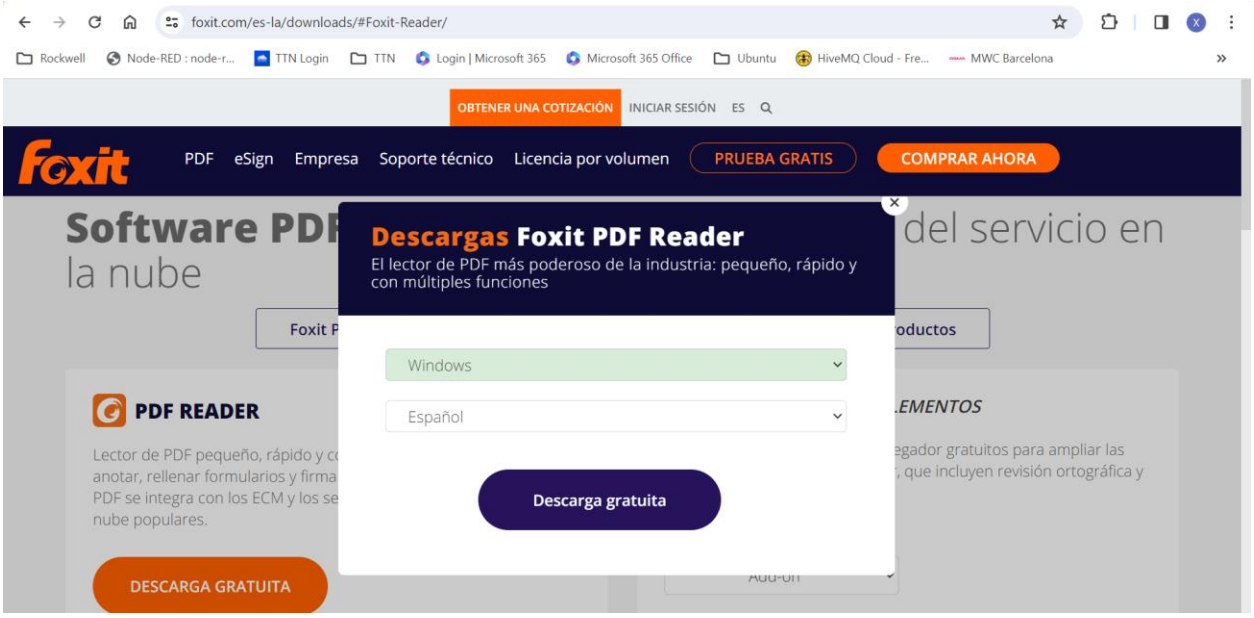
Look on the default application directory,
Be aware that this is not the directory where you store the application.
You have to look here with the right application name on the yellow market area
C:\Users\xavier.florensa\AppData\Local\Rockwell Automation\FactoryTalk Optix\Emulator\Projects\Training_Netlogic_Reports3\ApplicationFiles
And voilà



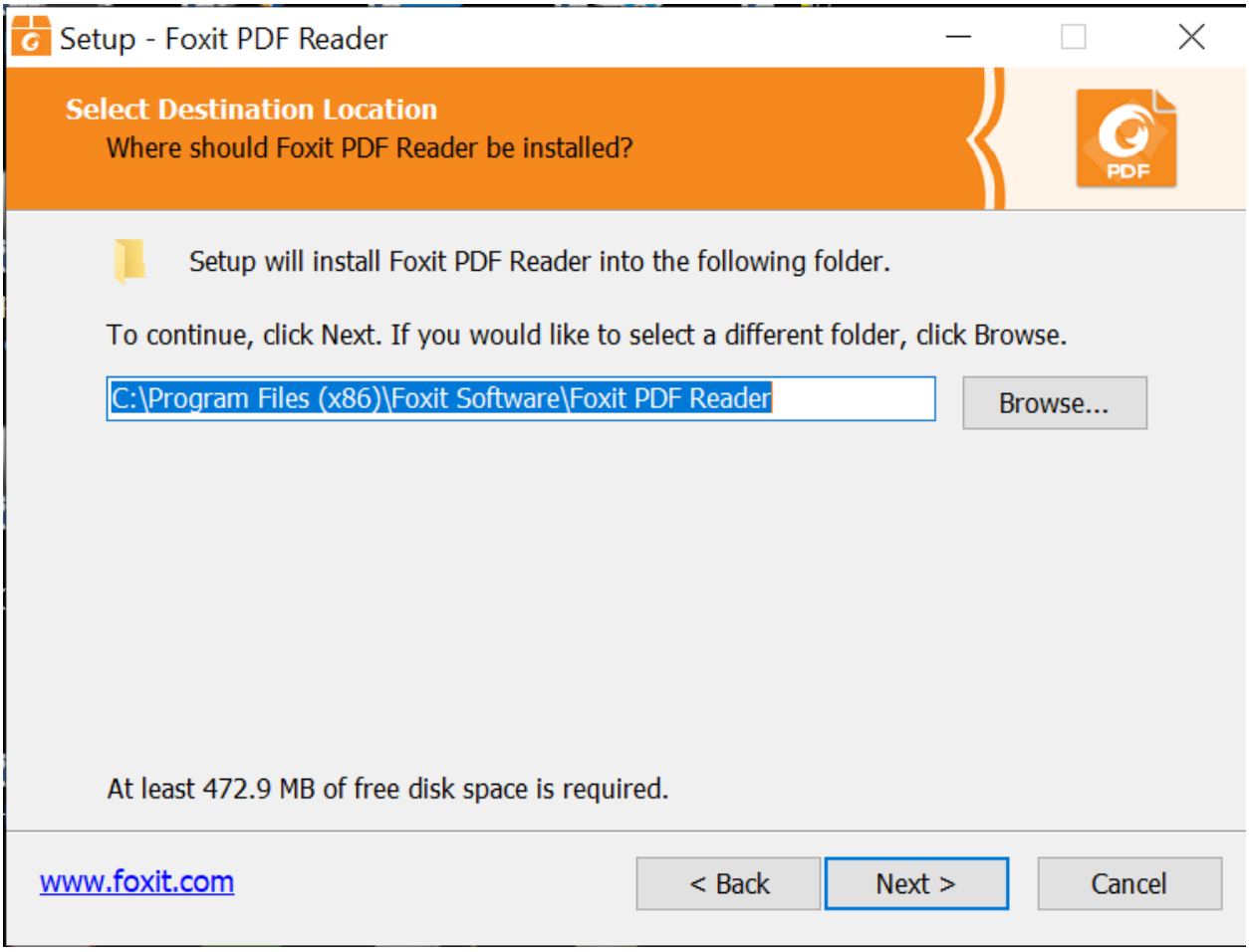
50.1. Using Foxit Pdf

Which advantages do we have with Foxit Pdf,

Pending to be cleared
Let's say that you will have a File explorer to place your pdf file, but I was not able to generate any populated pdf on that location.
Download this pdf editor
https://www.foxit.com/es-la/downloads/#Foxit-Reader/

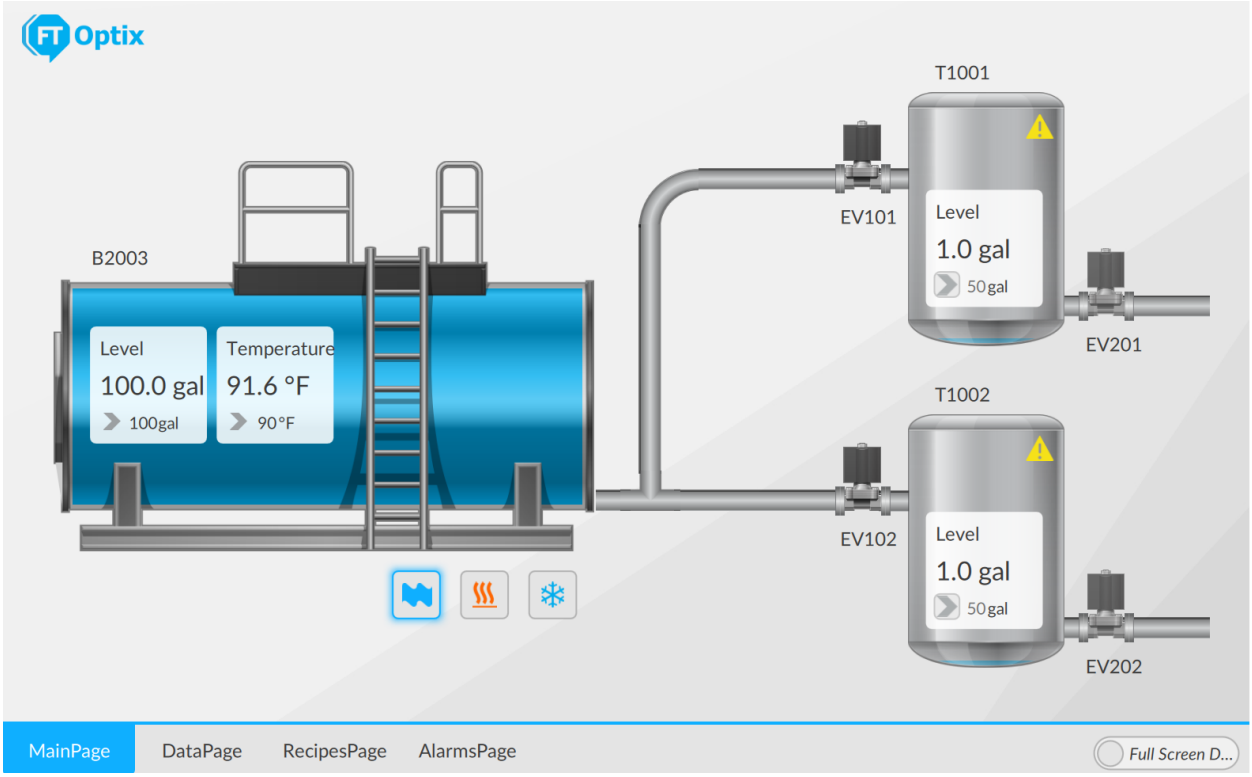


Use this default location



51. Using BoilerDemo

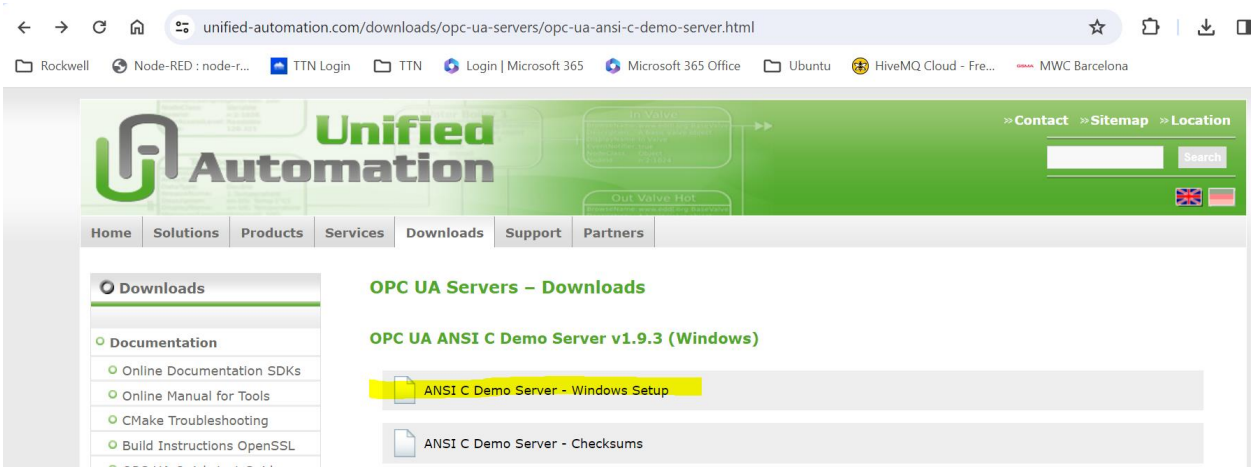
Probably you want to show this demo to your customers



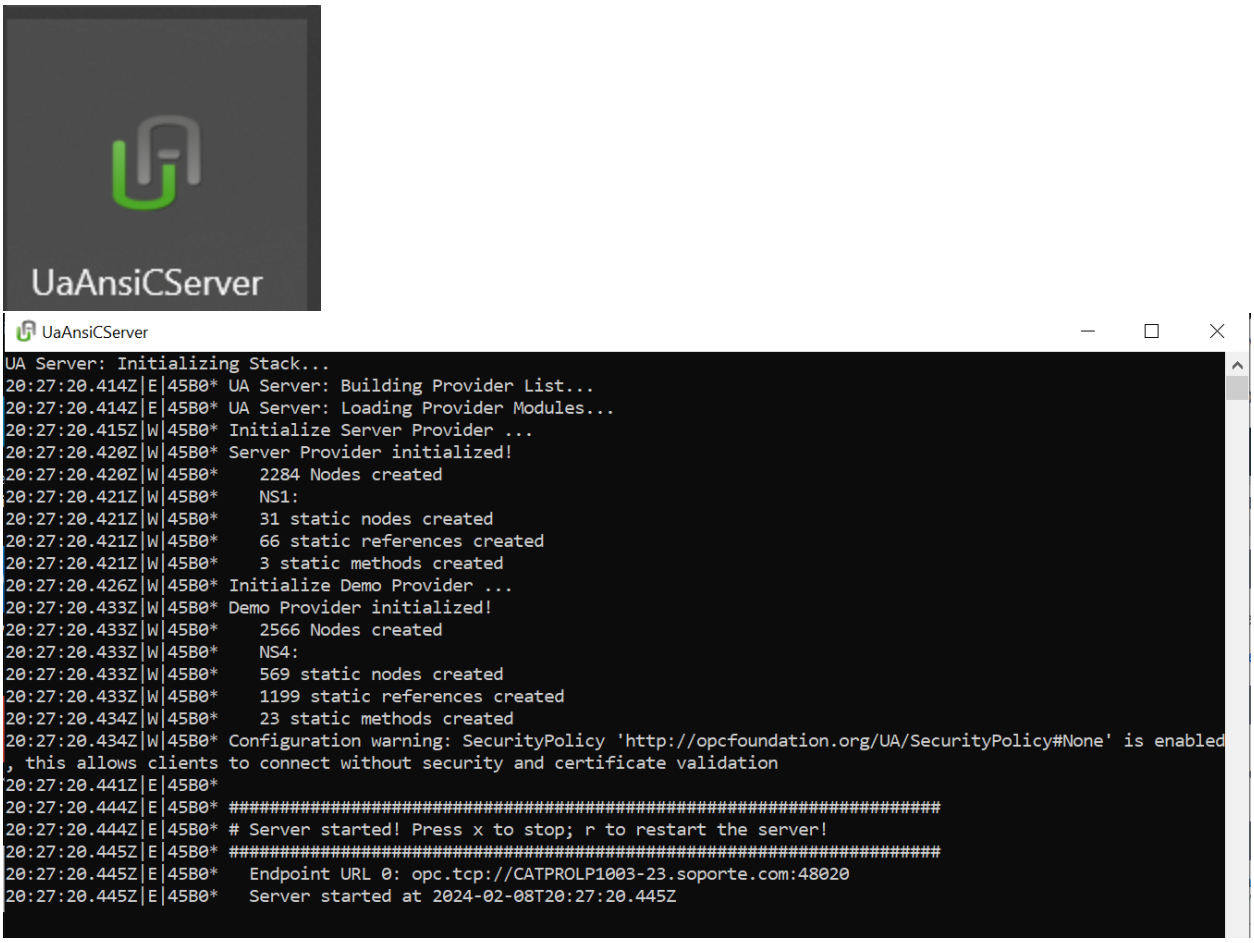
But apart from the Optix project you will need the following:

- A Studio5000 *.ACD file in order to execute the Tank1 and Tank2 on the PLC
- A OPC UA server example with predefined working variables on Boiler side.

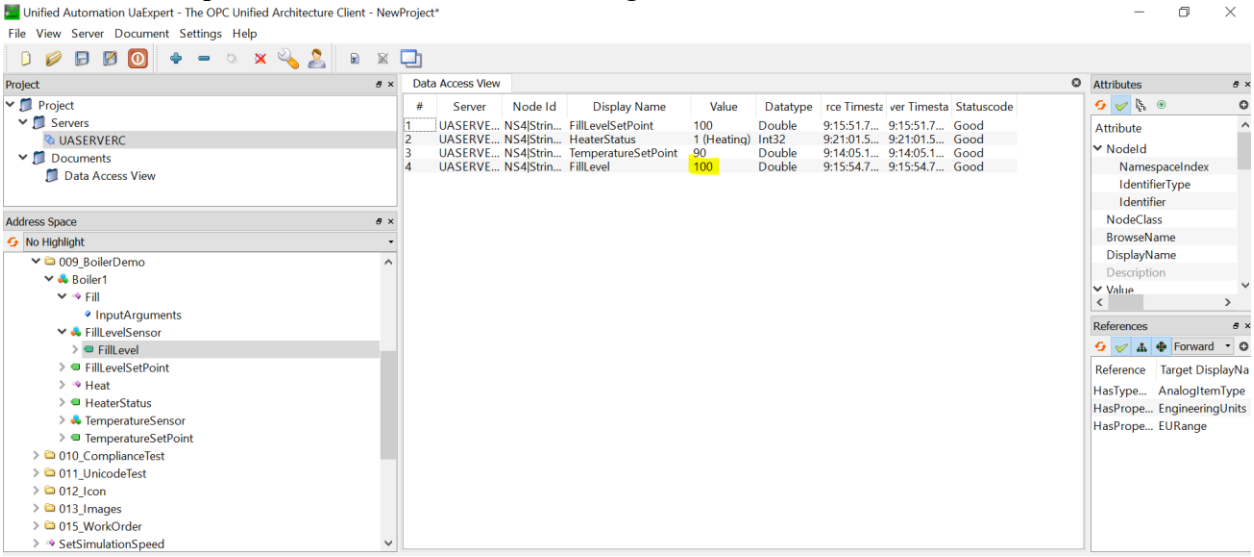
Install your open source OPC UA server
https://www.unified-automation.com/downloads/opc-ua-servers/opc-ua-ansi-c-demo-server.html

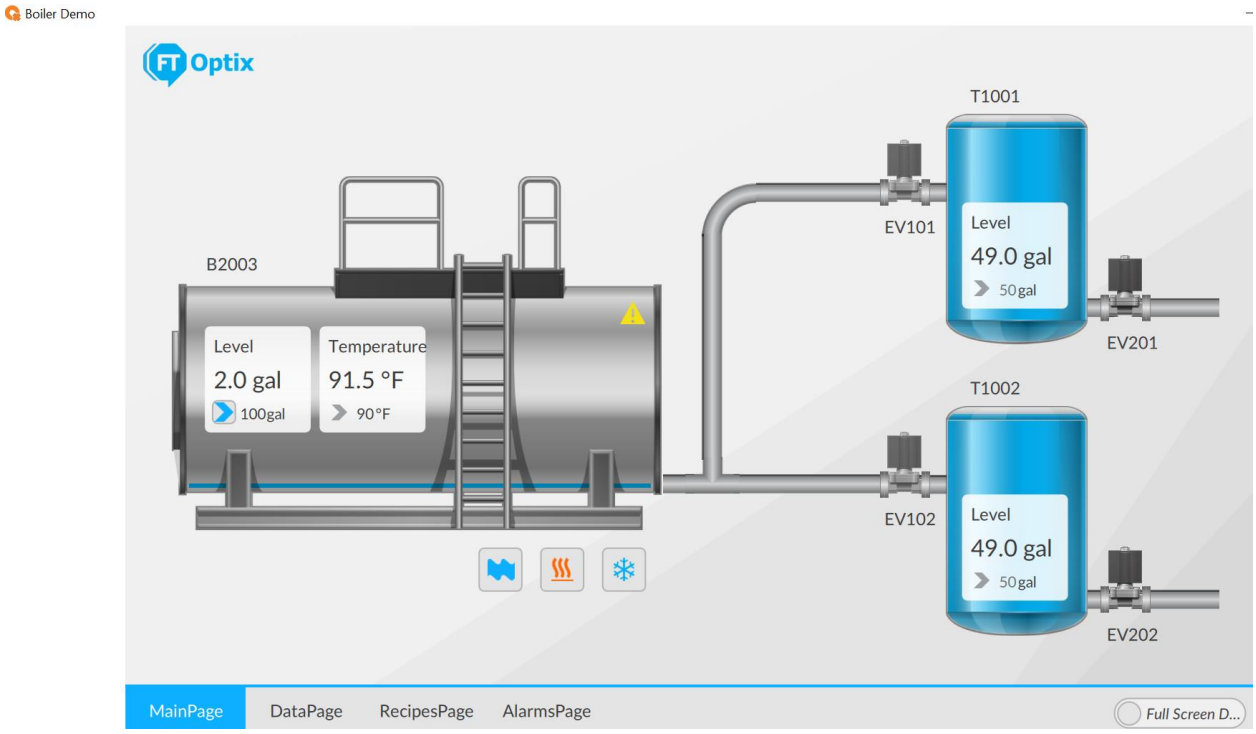


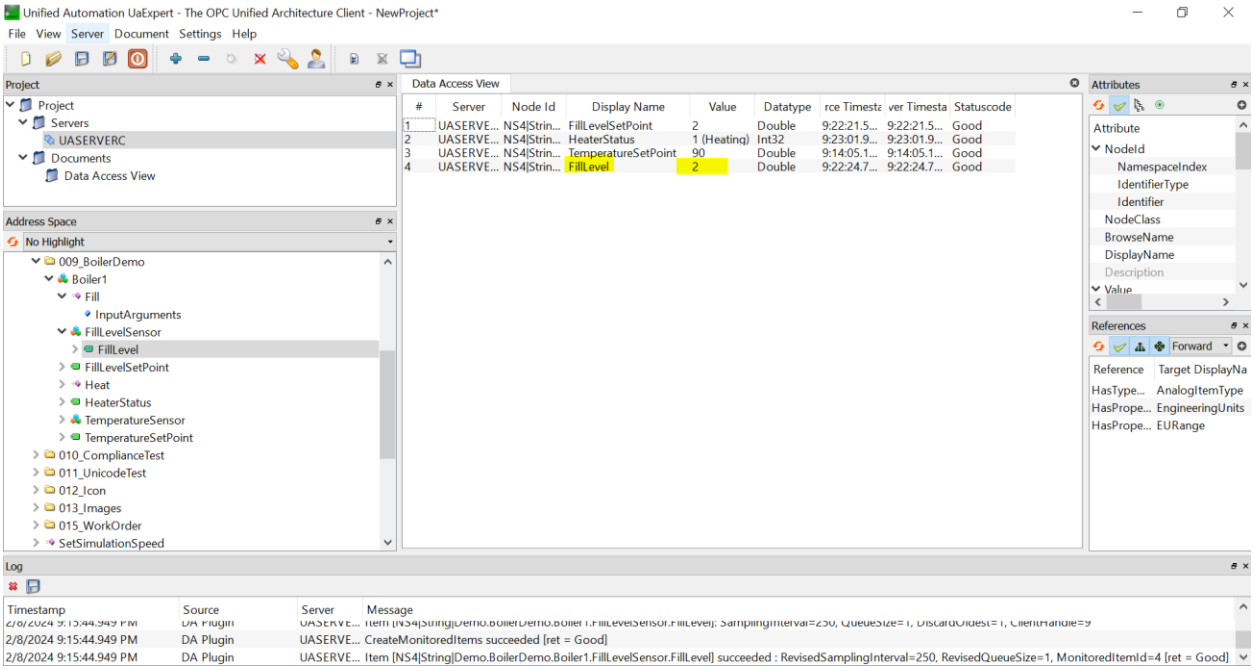
Let's start our server with this icon



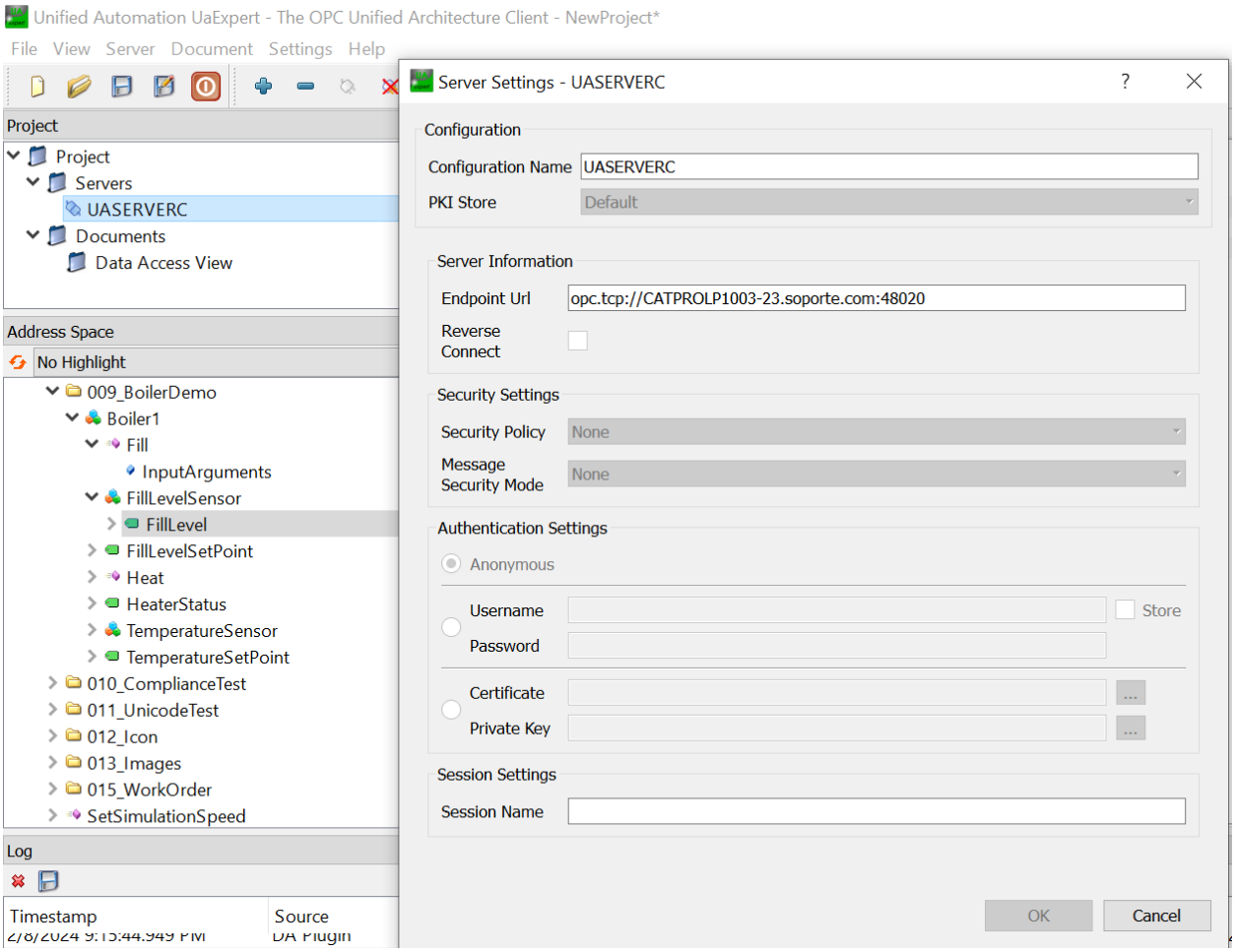
Let’s take a look at the OPC UA server used.
Let’s take a look from a OPC UA client view
Fill level corresponds to the real time Boiler widget will value







Let’s take a look at the Server configuration


Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject*
File View Server Document Settings Help
Project
Project
Servers
UASERVERC
Documents
Data Access View
Address Space
No Highlight
009_BoilerDemo
Boiler1
Fill
InputArguments
FillLevelSensor
FillLevel
FillLevelSetPoint
Heat
HeaterStatus
TemperatureSensor
TemperatureSetPoint
010_ComplianceTest
011_UnicodeTest
012_Icon
013_Images
015_WorkOrder
SetSimulationSpeed
Log
Timestamp
Source
Server Settings - UASERVERC
Configuration
Configuration Name
UASERVERC
PKI Store
Default
Server Information
Endpoint Url
opc.tcp://CATPROLP1003-23.soporte.com:48020
Reverse Connect
Security Settings
Security Policy
None
Message Security Mode
None
Authentication Settings
Anonymous
Username
Store
Password
Certificate
Private Key
Session Settings
Session Name
OK
Cancel

That’s all