Daniel Gonzalez Garcia
September 2024
danielcabilan@hotmail.com

Introduction:

The Customer Data Retrieval API enables programmatic access to the company's customer database such as name, lastname, email, and the company they are associated with, which can be used for both internal and external use.

The Customer Data Retrieval API delivers information in JSON format ready for deserialization and quick understanding, which enables it to be used in multiple programming languages.

Authorization:

Authorization is required for the use of the API, this is implemented by using a username and password
All requests to the API must include a "Basic Auth" authorization header with the following data:

Username: 5161aa31f9b94a7b82b5e4d28bbdbd65
Password: eB07357Fb01E4b9594C91748CFe0F062

Customer:

-The customer resource

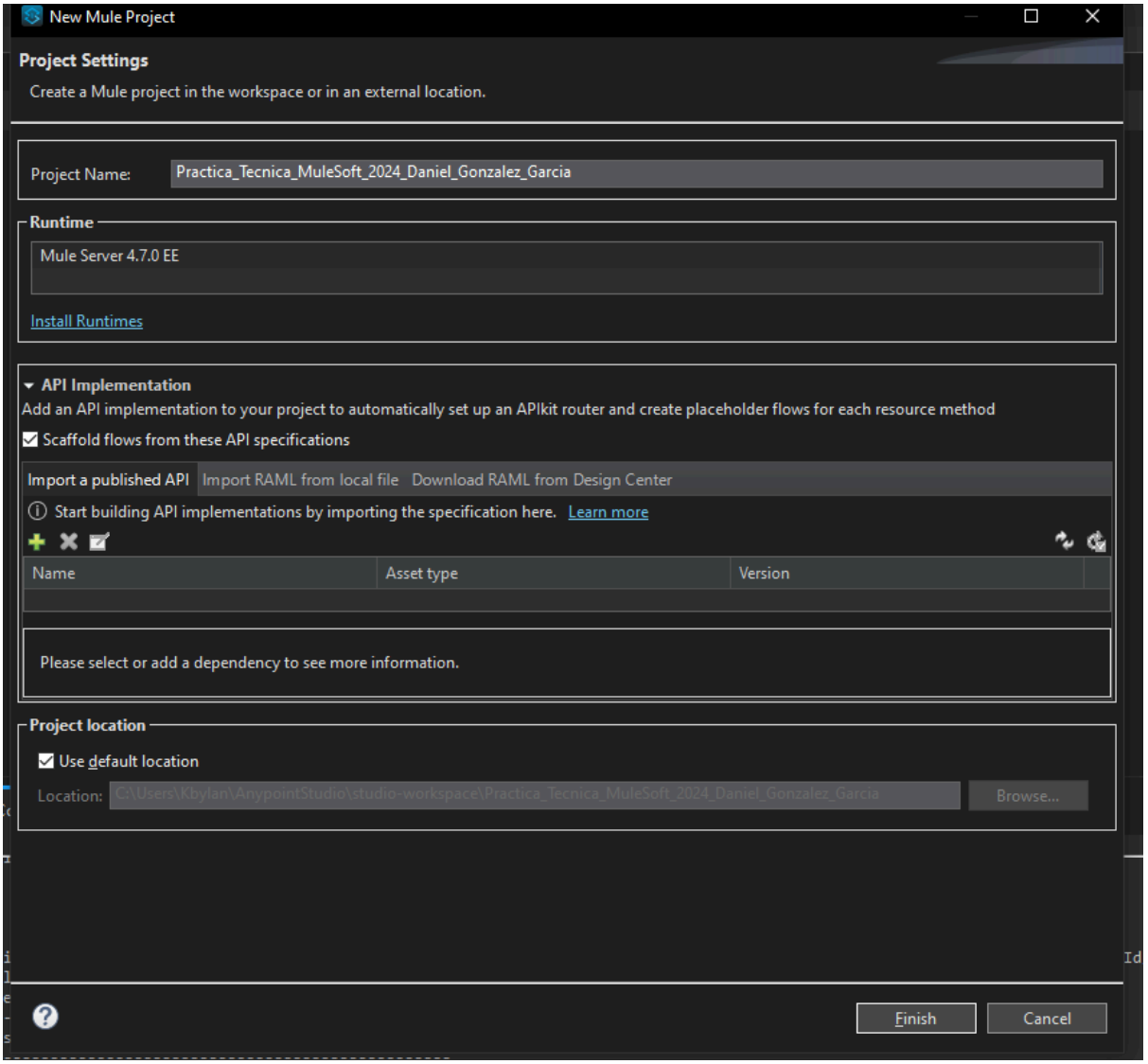
The "customer" resource is a JSON formatted version of a customer's data, the /customer endpoint responds with this format

Get Customer Data:

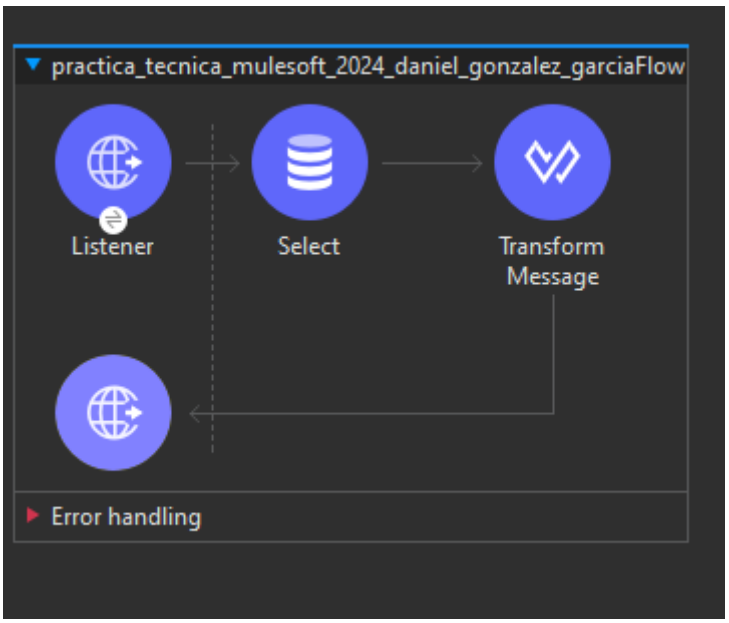
GET https://practica-tecnica-daniel-gonzalez-garcia-24-f49kwq.5sc6y6-2.usa-e2.cloudhub.io/api/v1/sps/customers
Response: Array of Customer

MuleSoft Technical Practice Trainee

We will start the process by creating the project within Anypoint Studio with the following configuration

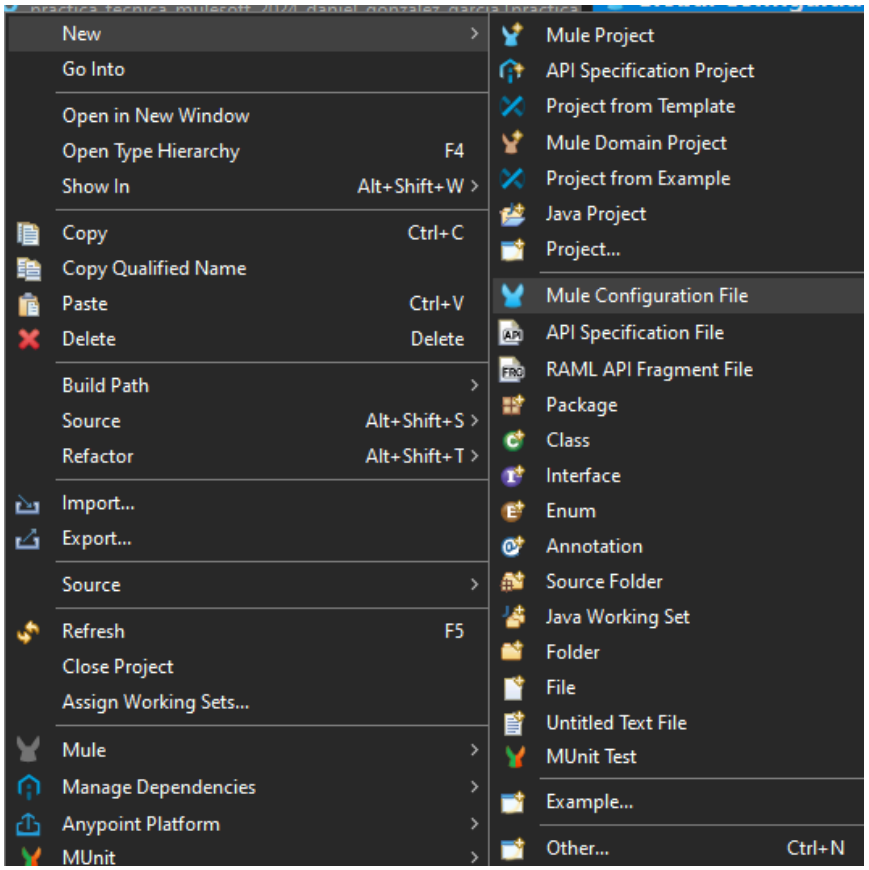


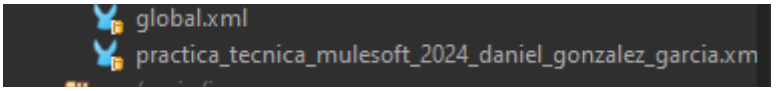
Once created, we will add three modules: "Listener", "Select", "Transform Message"



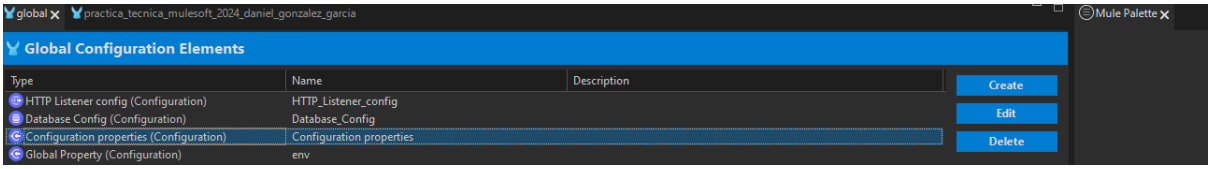
While "Listener" and "Transform Message" are in the default options in Mule Palette, "Select" is a module found inside "Database" just search for it in "Add Modules"

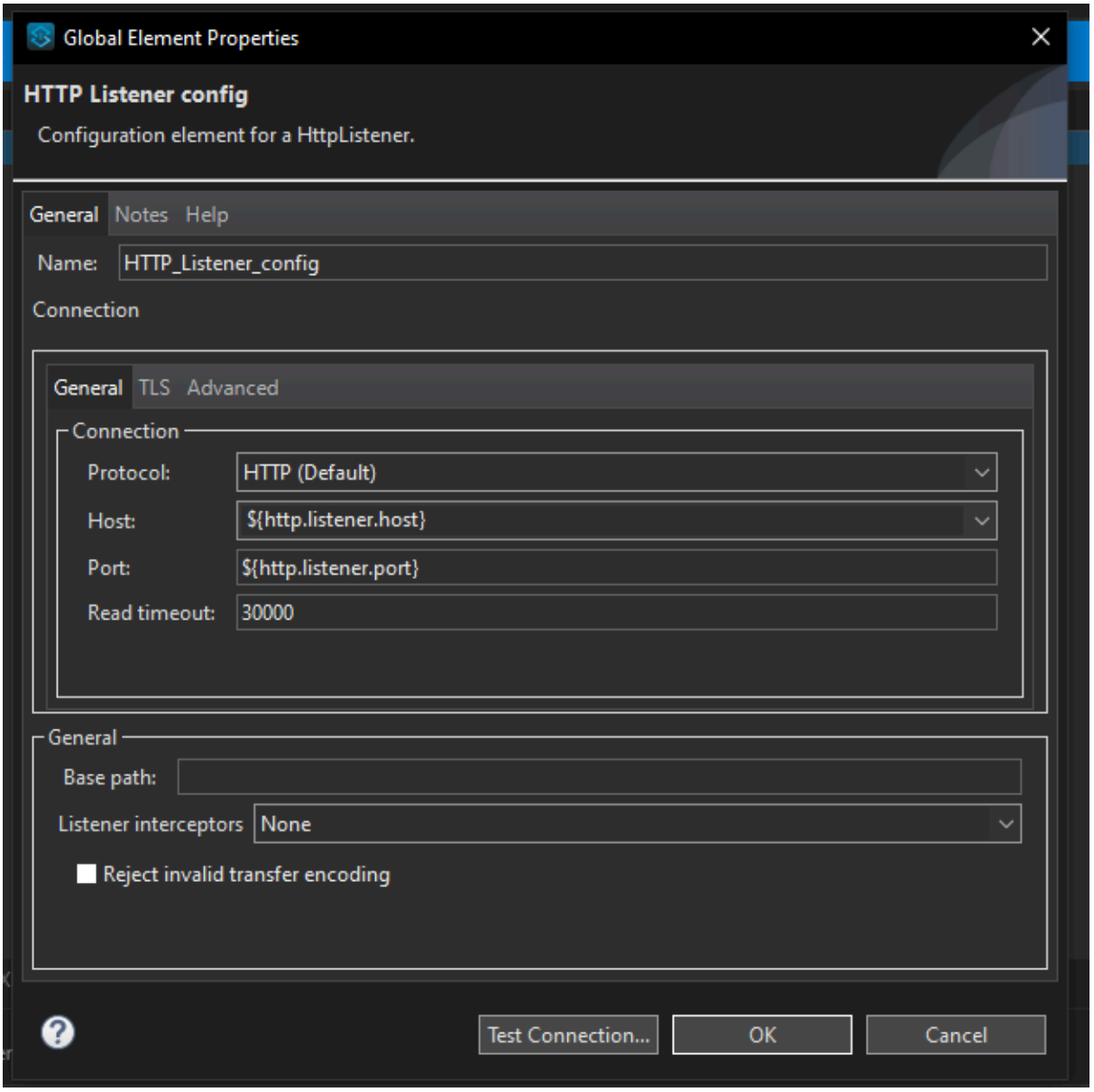
Now we will add a new configuration xml file that will serve as our global configuration file. with the name “global.xml”

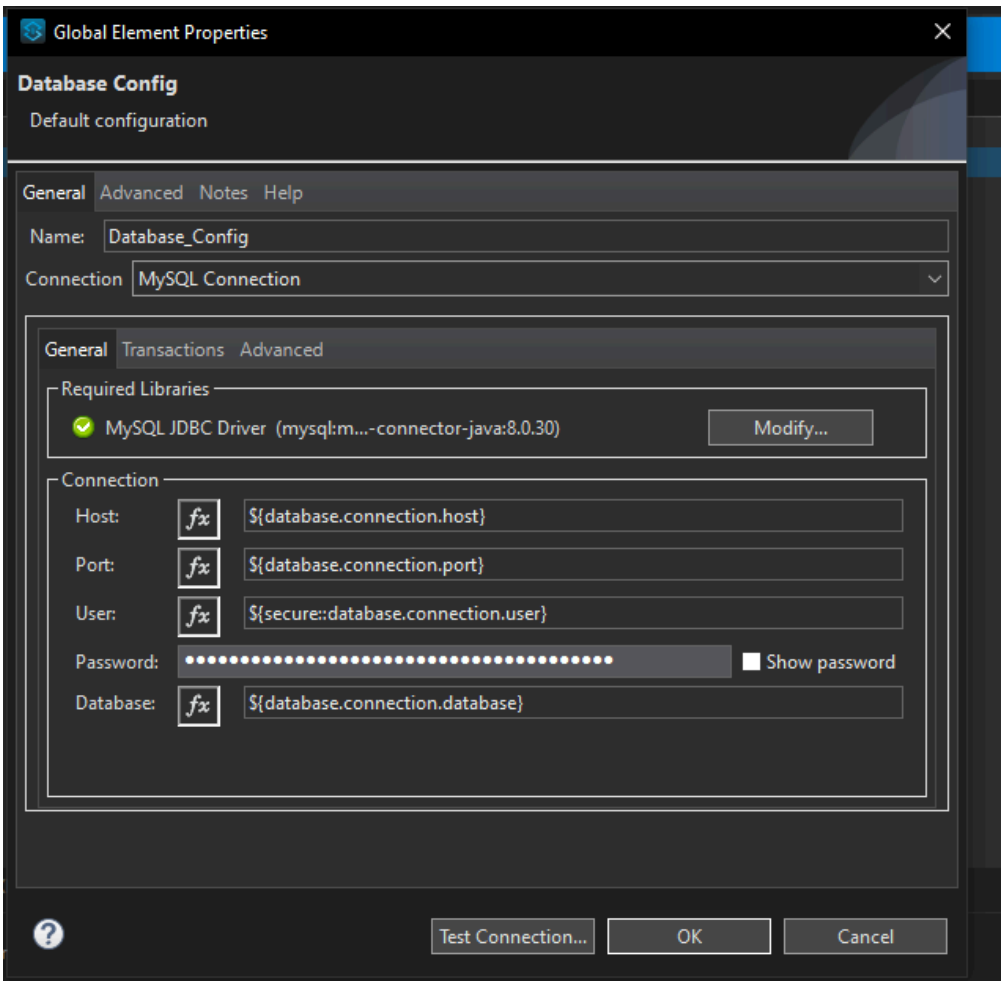


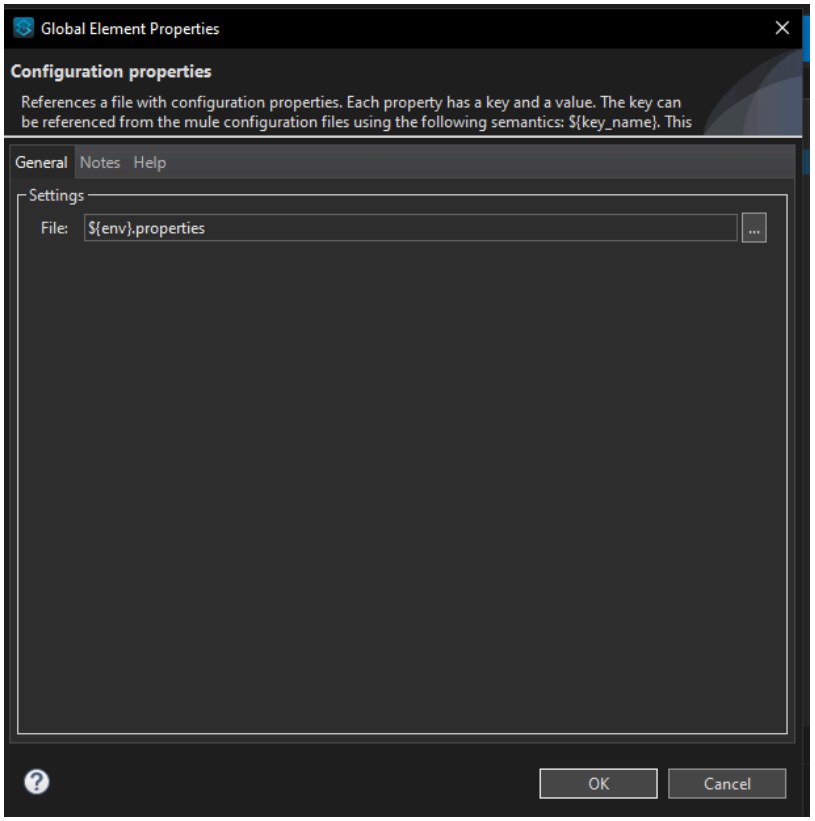


Here we will add three types of elements: "Http Listener config" "DataBase config" “Configuration Properties” and "GlobalProperty" with the following settings

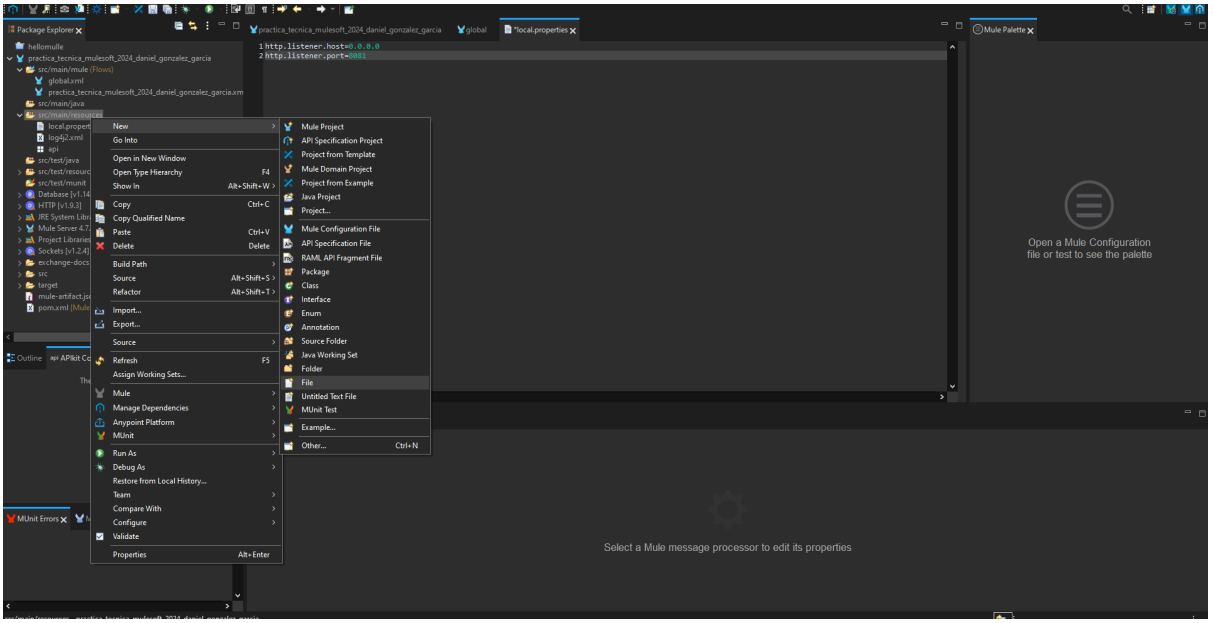





Global Element Properties
Database Config
Default configuration
General
Advanced
Notes
Help
Name:
Database_Config
Connection
MySQL Connection
General
Transactions
Advanced
Required Libraries
MySQL JDBC Driver (mysql:m...-connector-java:8.0.30)
Modify...
Connection
Host:
${database.connection.host}
Port:
${database.connection.port}
User:
${secure::database.connection.user}
Password:
Show password
Database:
${database.connection.database}
Test Connection...
OK
Cancel


Global Element Properties
Configuration properties
References a file with configuration properties. Each property has a key and a value. The key can be referenced from the mule configuration files using the following semantics: ${key_name}. This
General
Notes
Help
Settings
File:
${env}.properties
OK
Cancel

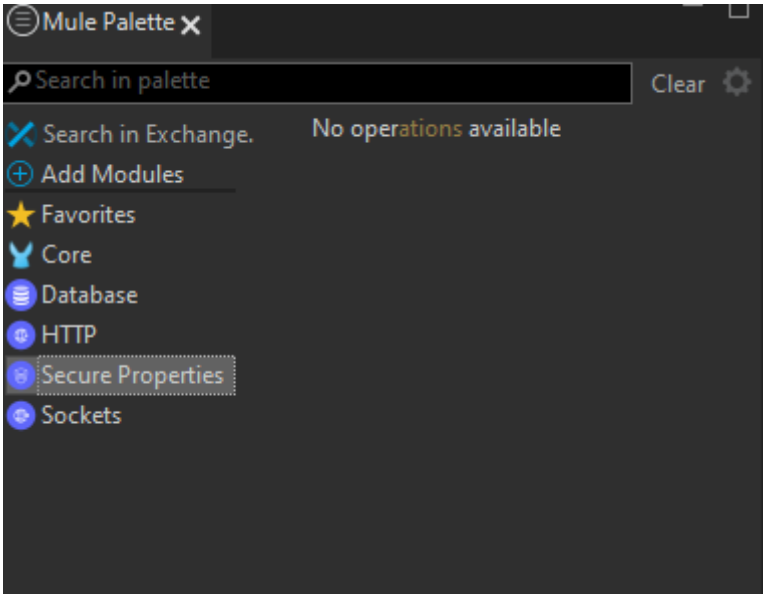
To store the configuration variables we will create the files "local.properties" and "dev.properties"

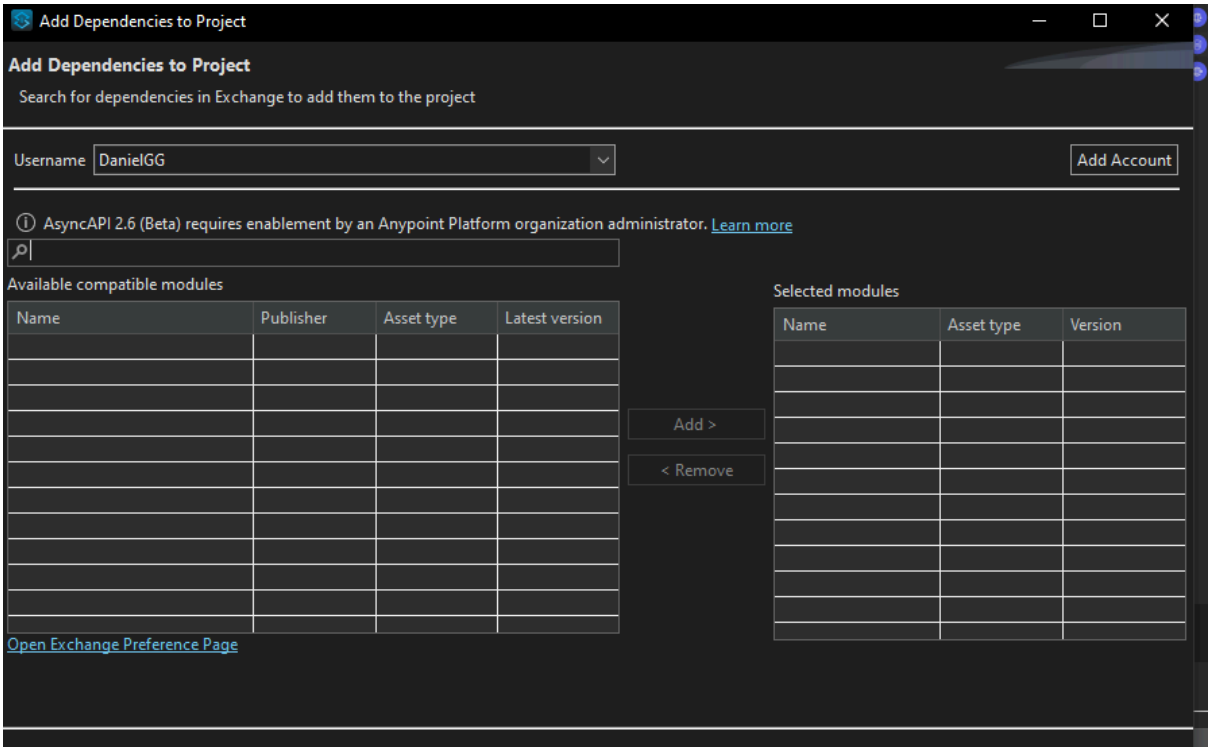


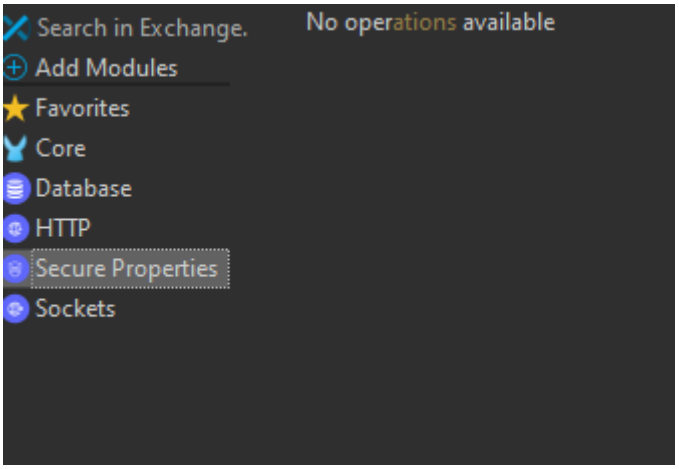
Where we will add the following data:

http.listener.host=0.0.0.0
http.listener.port=8081
database.connection.host=mudb.learn.mulesoft.com
database.connection.port=3306
database.connection.database=training

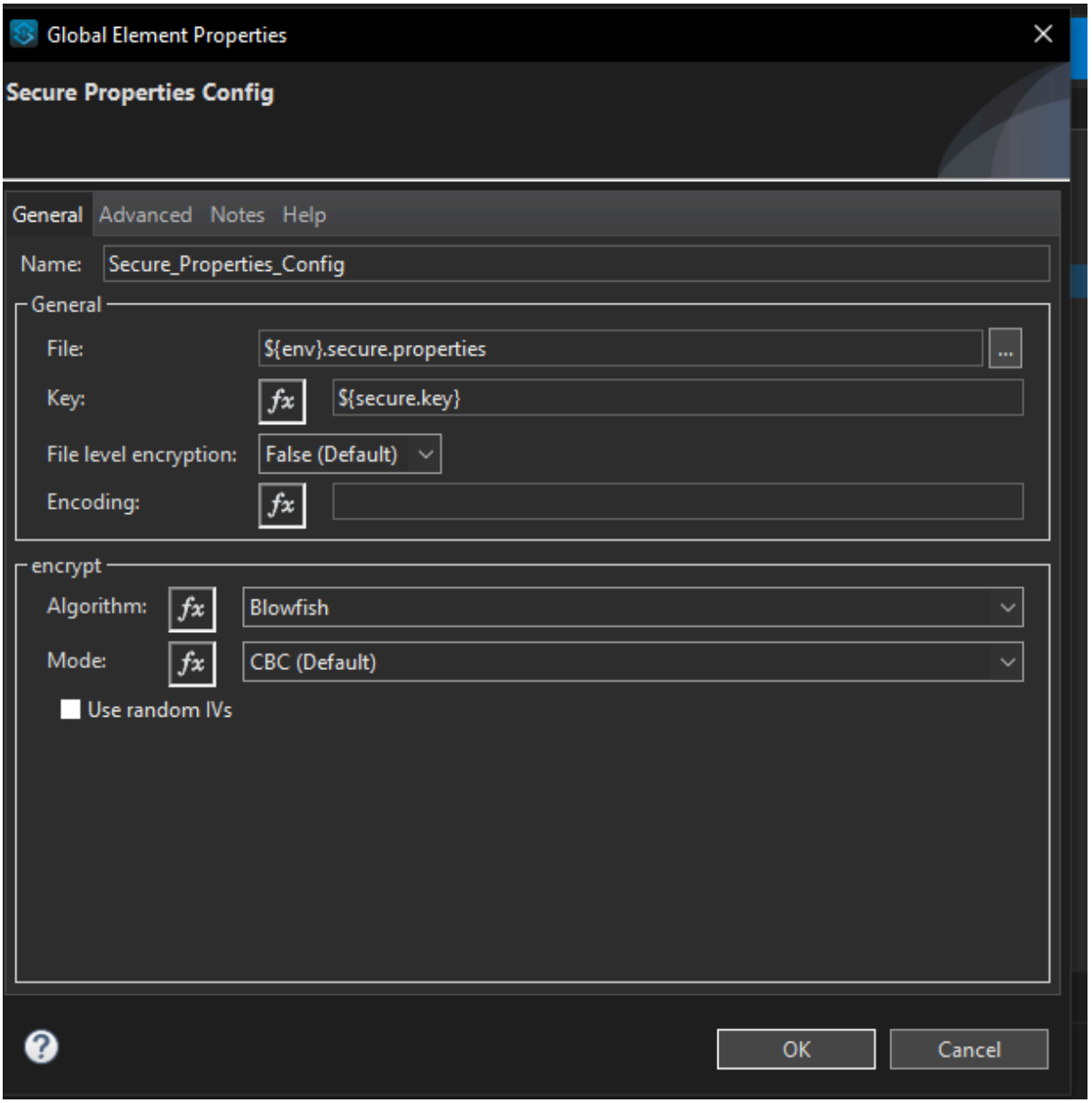
We can modify this data via the .properties files, but we need secure files to contain sensitive data, so we will use "Secure Properties" to encrypt and decrypt them, so we will use the "Search in exchange" option to add it to our project.



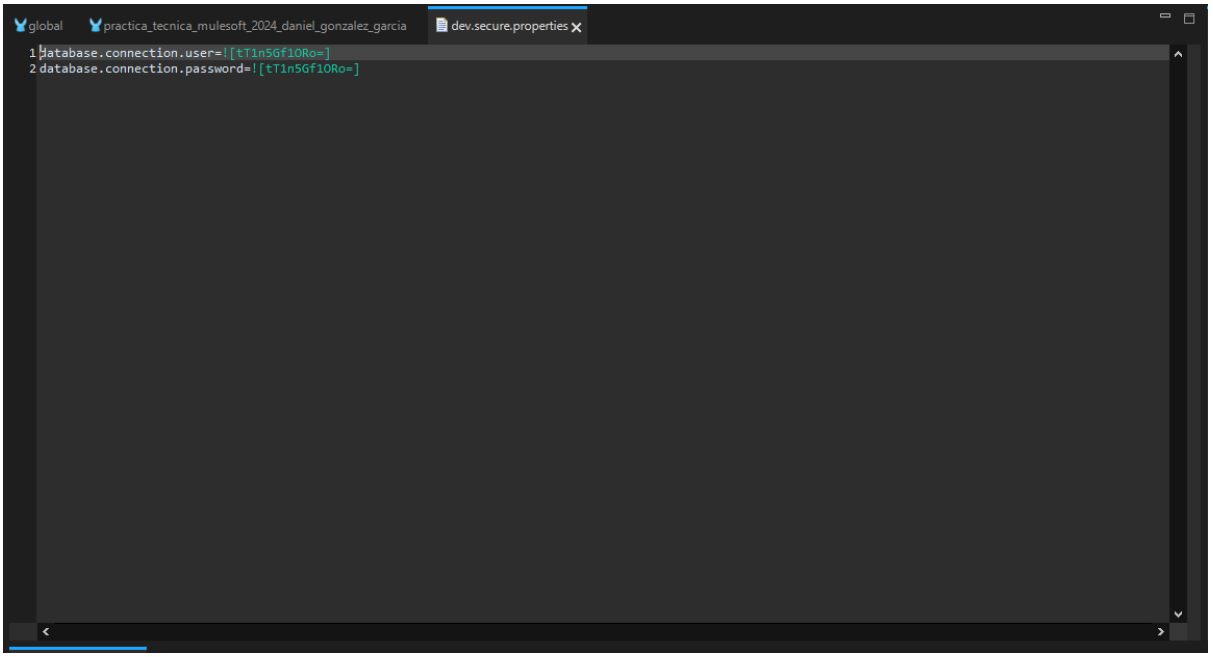




Now we will have available in our global document a new configuration element "Secure properties Config", we add it with the following configuration



And just like before we will now create two new files, "dev.secure.properties" and "local.secure.properties" with the following data

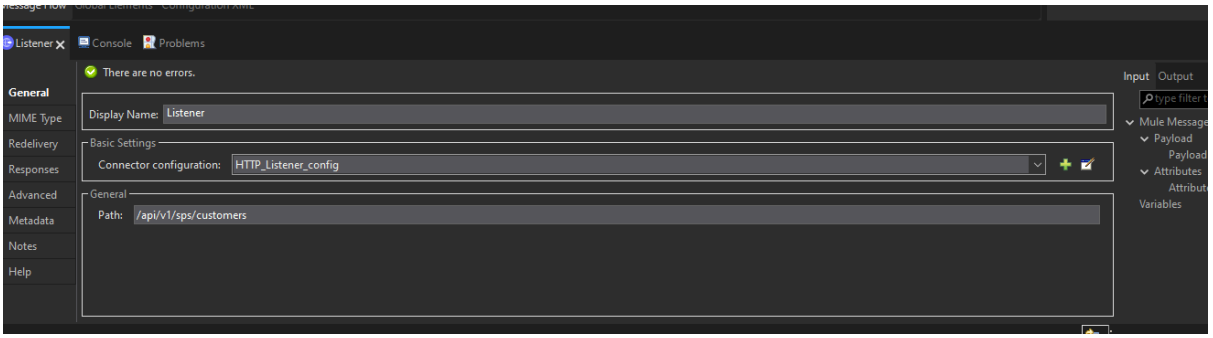


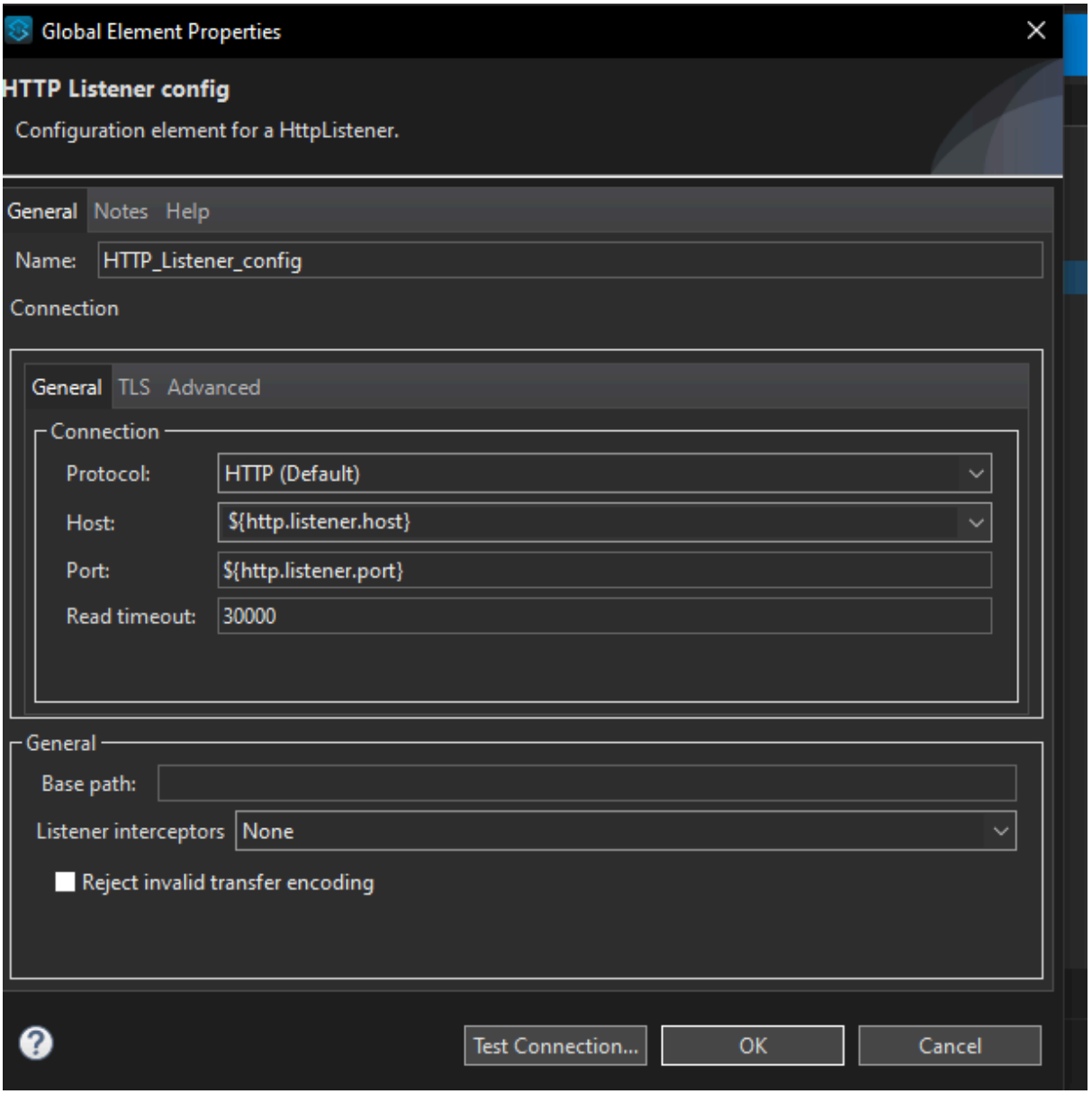
This data must be encrypted, and is obtained using the "secure-properties-tool.jar" file by opening a terminal in the path where the file is located and using the following command

```
C:\Users\Kbylan\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blo
wfish CBC PruebaTecnica2024 "mule"
tT1n5Gf1ORo=
```

It should be noted that: "PruebaTecnica2024" is the key to encrypt and decrypt and that "mule" is the value to be encrypted.
Let us also remember that for encrypted data it is necessary to add them with a symbol "!" and between "[ ]"
We can now modify the "Mule properties" of our modules from the beginning of this documentation

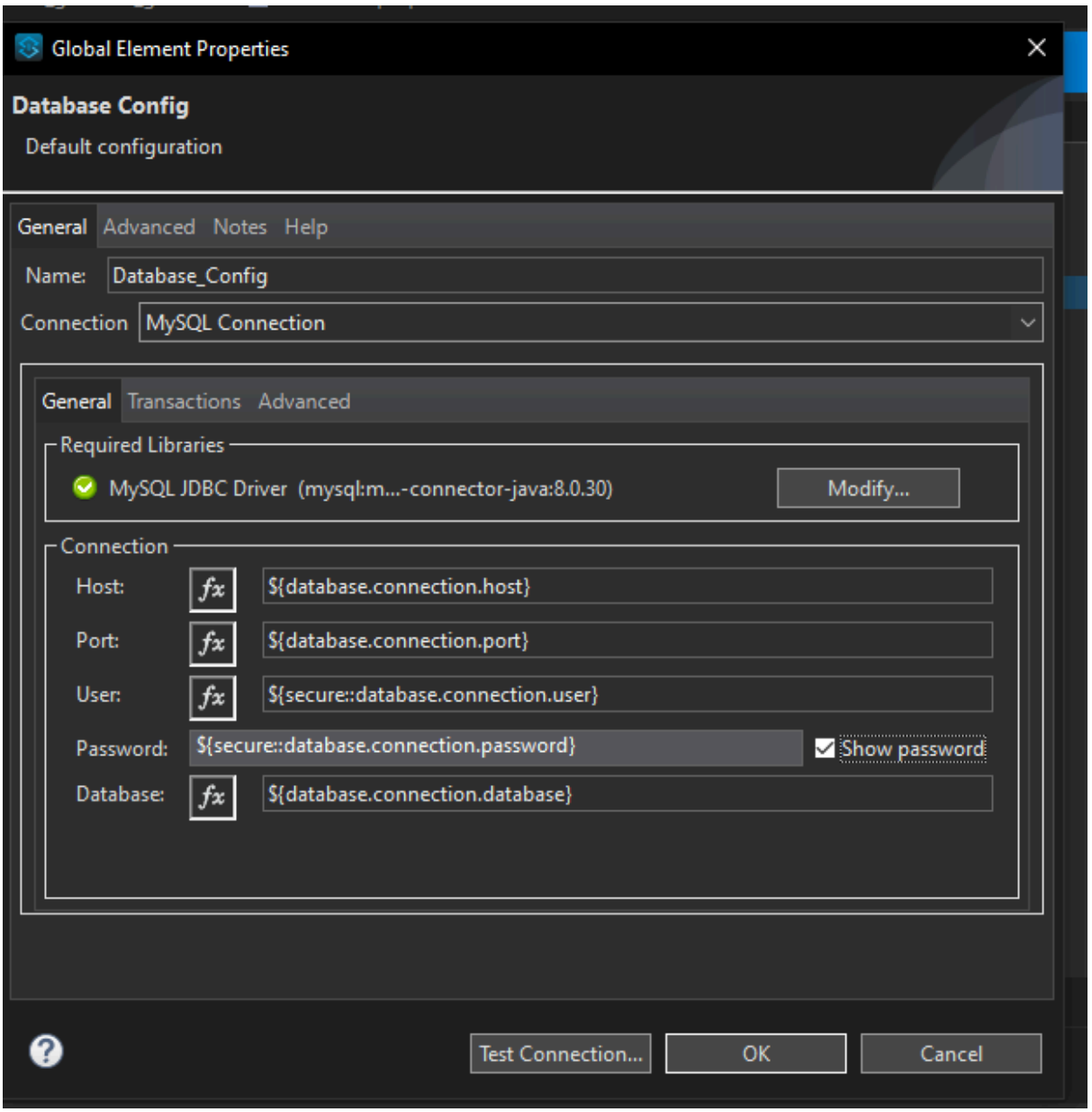
Listener:




Global Element Properties
HTTP Listener config
Configuration element for a HttpListener.
General
Notes
Help
Name:
HTTP_Listener_config
Connection
General
TLS
Advanced
Connection
Protocol:
HTTP (Default)
Host:
${http.listener.host}
Port:
${http.listener.port}
Read timeout:
30000
General
Base path:
Listener interceptors
None
Reject invalid transfer encoding
Test Connection...
OK
Cancel

Select

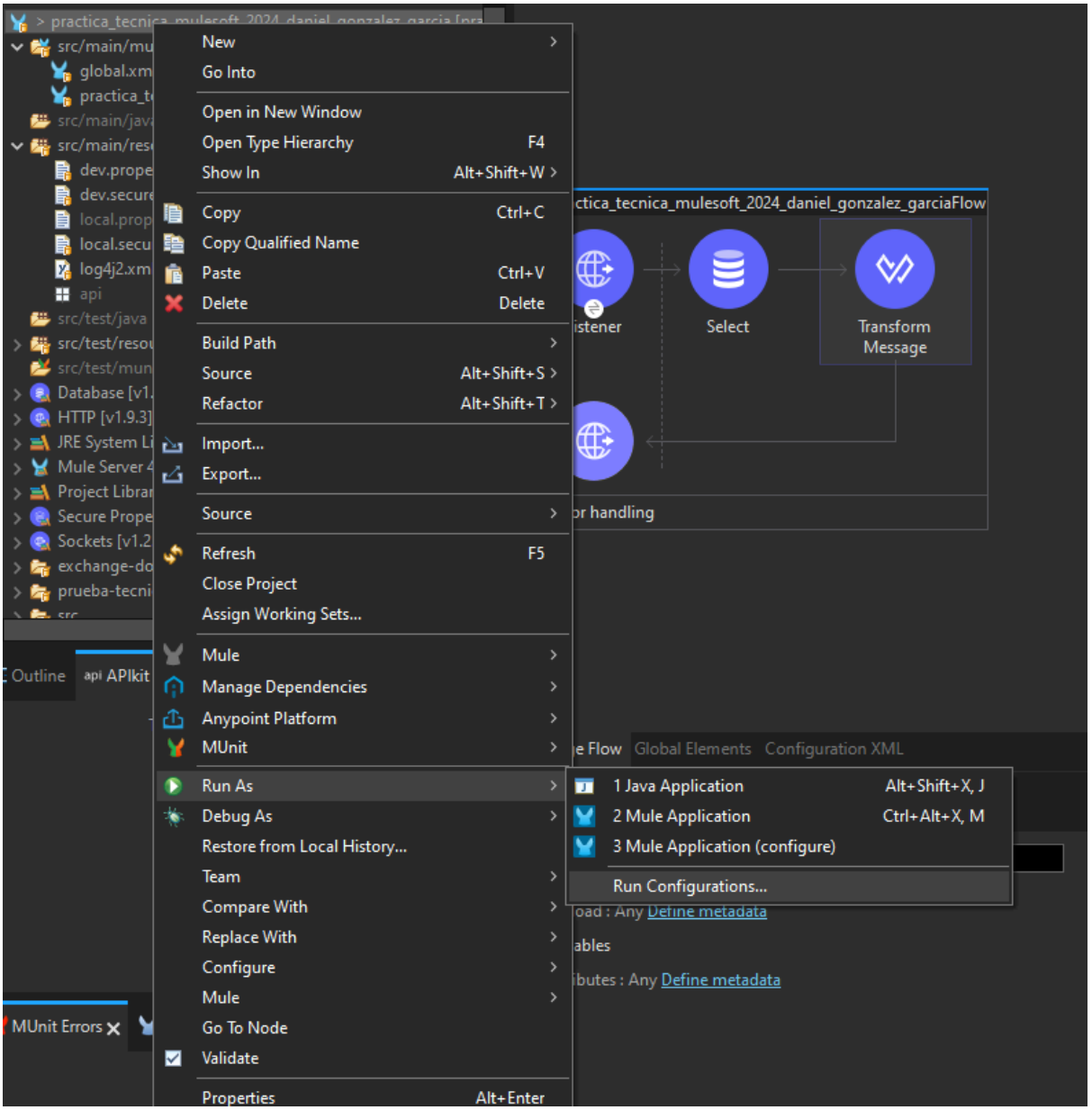

Message Flow
Global Elements
Configuration XML
Select
Console
Problems
There are no errors.
General
Display Name:
Select
Advanced
Error Mapping
Metadata
Notes
Help
Basic Settings
Connector configuration:
Database_Config
Query
SQL Query Text:
SELECT * FROM customers;
Input Parameters:
#[{'name': "Max", 'nickname': "The Mule", 'company': "MuleSoft"}]

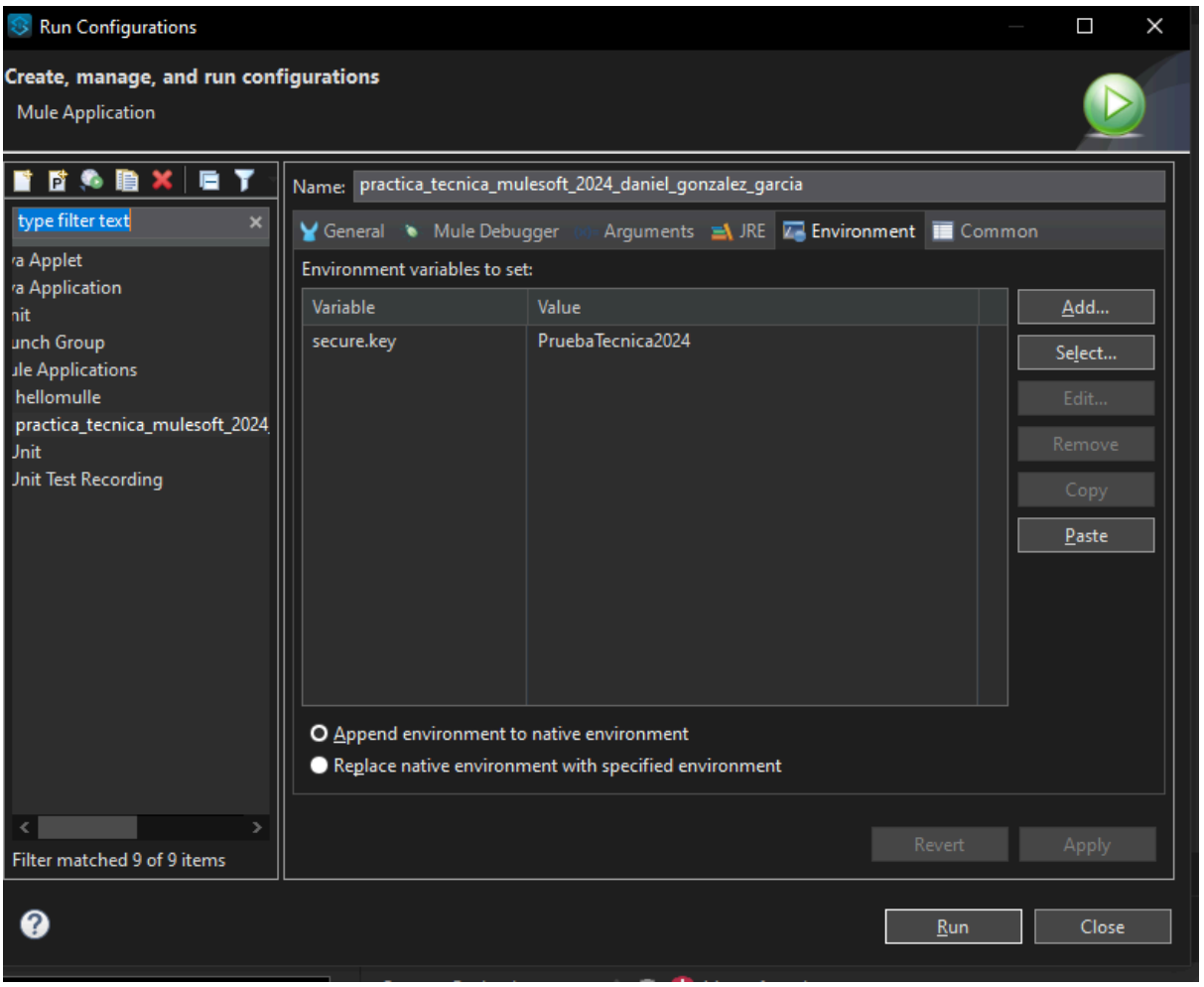


It is important to note that when we use a "Secure Property" we have to add "secure::" as in the case of "user" and "password"

Transform Message:

Finally we have to add the last variable that we are missing "secure.key" but we must add it not in the code but as an environment variable when running the program

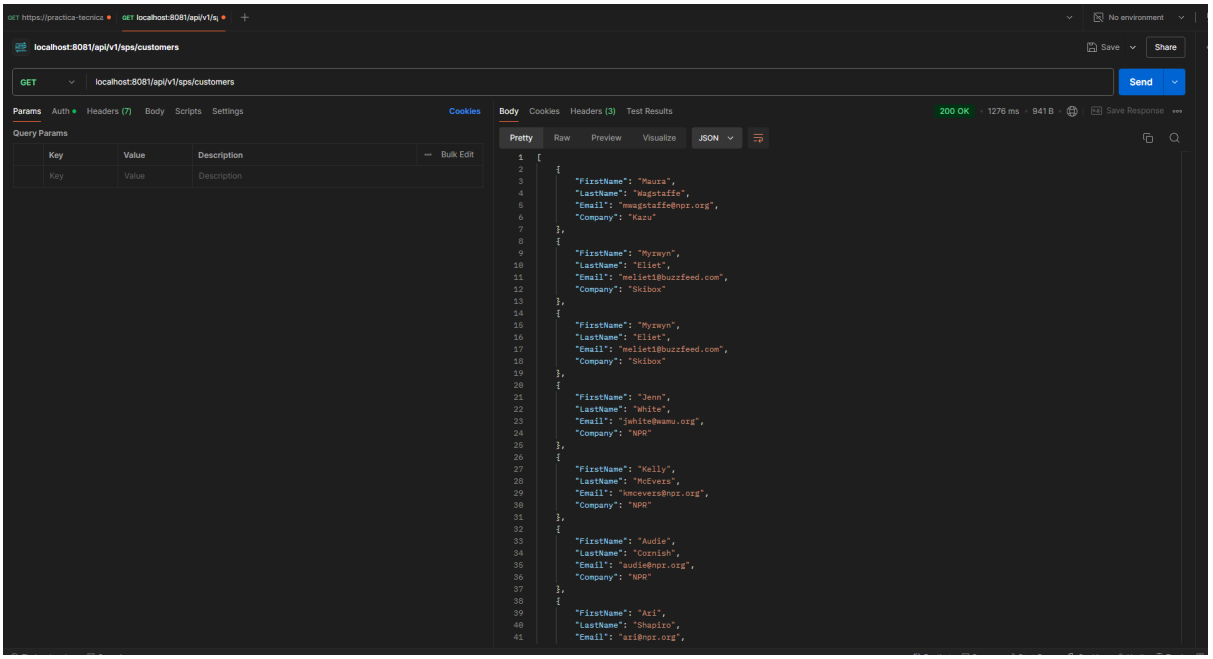

New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5
Close Project
Assign Working Sets...
Mule
Manage Dependencies
Anypoint Platform
MUnit
Run As
Debug As
Restore from Local History...
Team
Compare With
Replace With
Configure
Mule
Go To Node
Validate
Properties Alt+Enter
1 Java Application Alt+Shift+X, J
2 Mule Application Ctrl+Alt+X, M
3 Mule Application (configure)
Run Configurations...
Select
Transform Message
Global Elements
Configuration XML
Outline
MUnit Errors



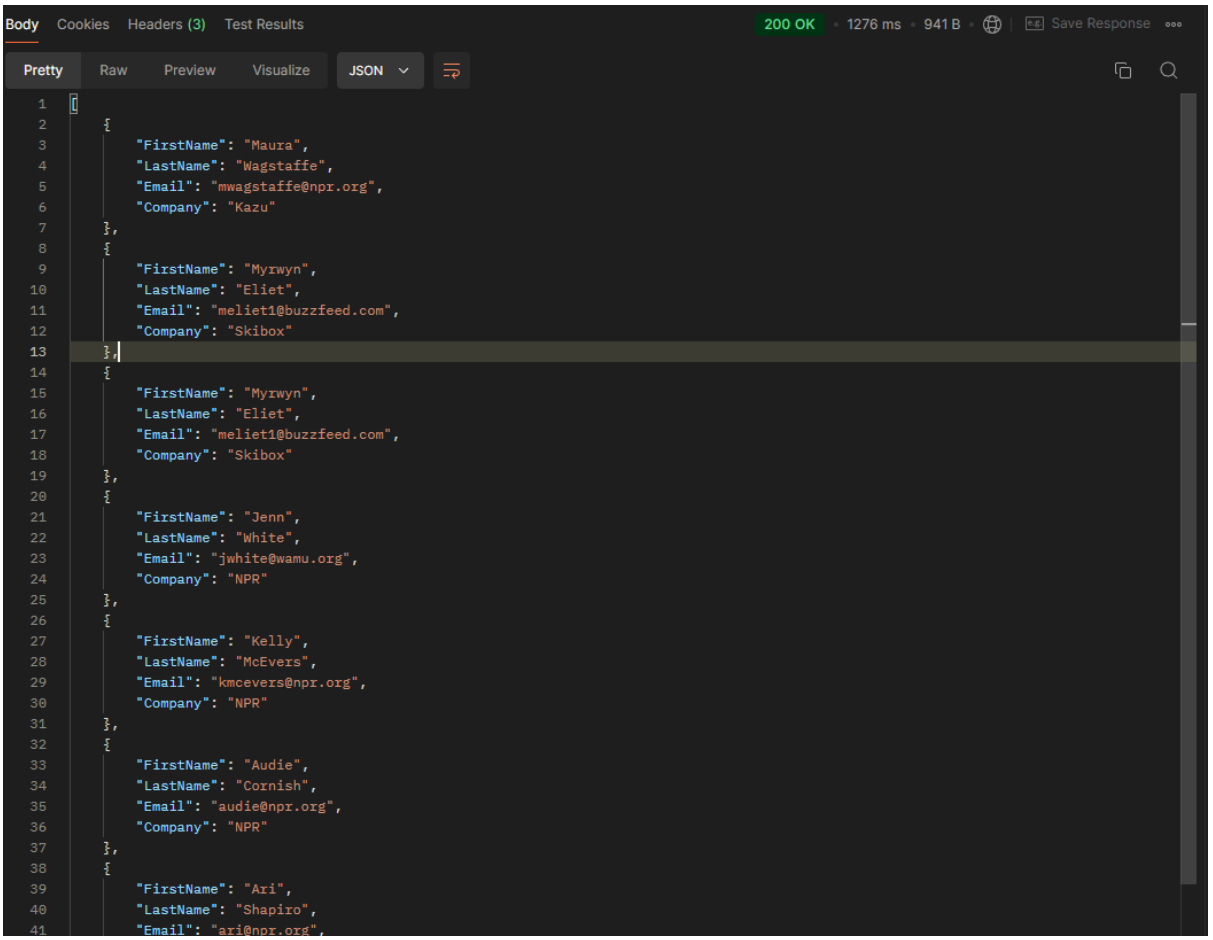
Remember that the value of the variable must be the key that we use to encrypt our values.

Once done, we run the program and we will verify that everything is correct with the help of "Postman"

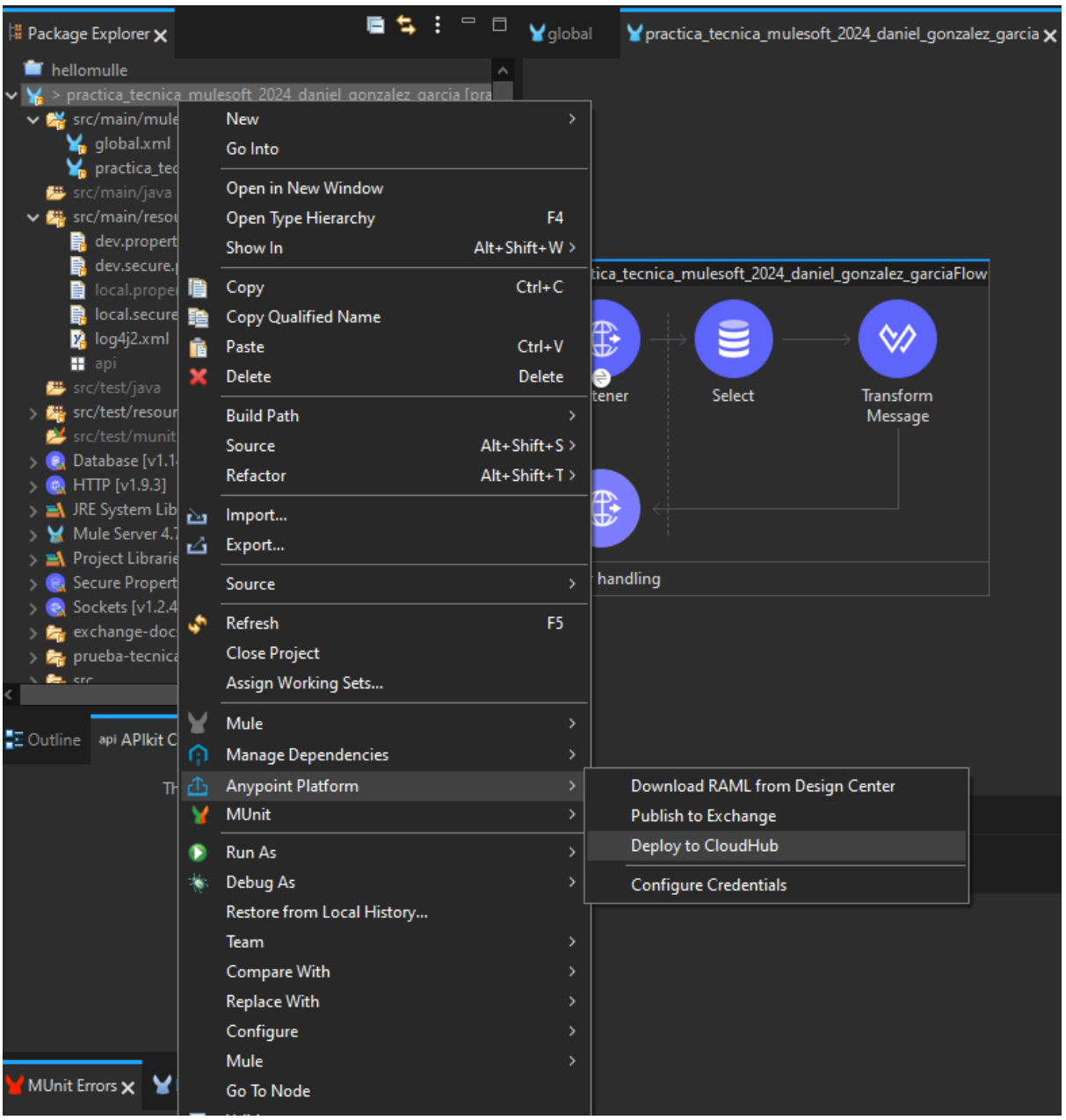
Now we just need to make a get request to the endpoint that we configured in our project with our listener



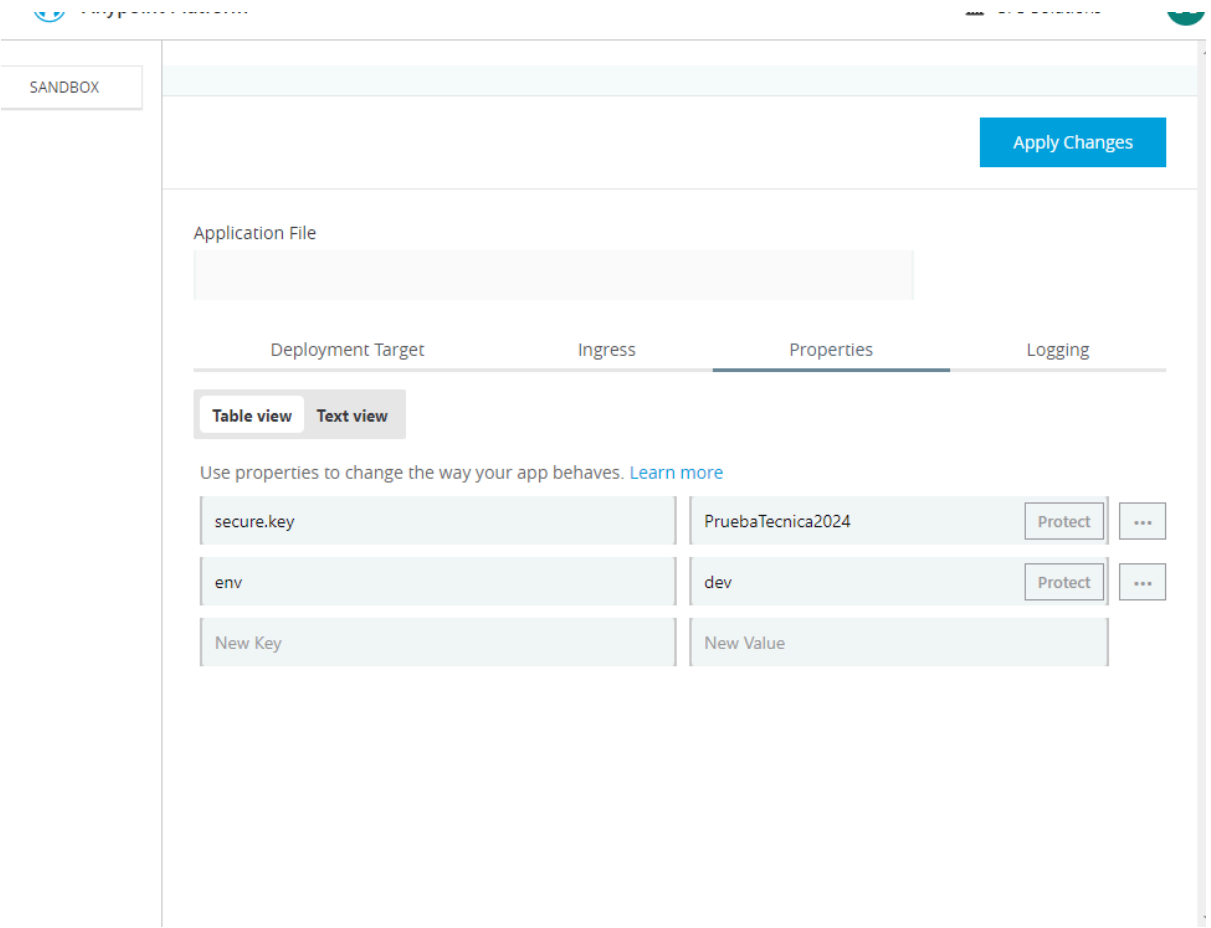
As a response we received the code "200" and a json file with the data extracted from the database



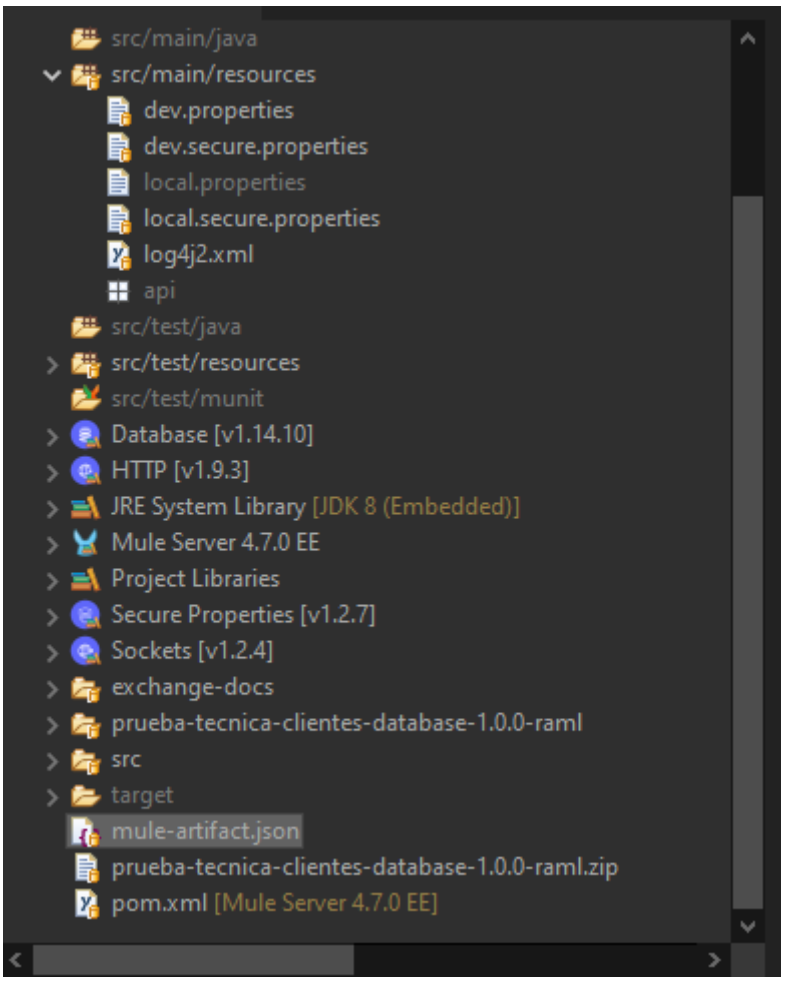
Now we just need to deploy it in "cloudhub" for this we only need a valid account and click on the following option



And before deploying we must add the following environment variables in the properties tab



If you want to hide the property values in "AnyPoint Platform" you just have to modify the following file

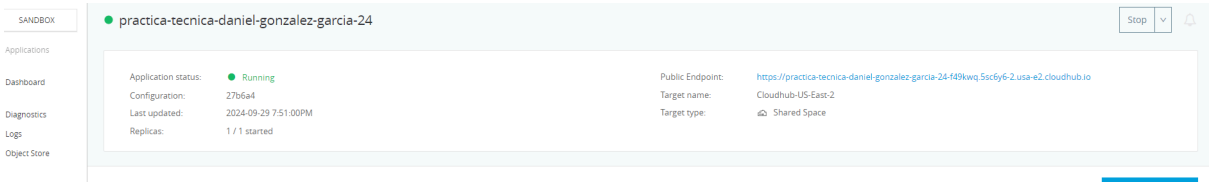


Adding the following line

"secureProperties": ["secure.key"]

Now we can test the deployed version with "Postman"

In "Runtime Manager" of "AnyPoint Management" we must obtain this data from the application that we deploy



And we make a request to that address with the configured endpoint

As a response we will also get the same result as with the local version

GET https://practica-tecnica-daniel-gonzalez-garcia-24-f49kwq.5sc6y6-2.usa-e2.cloudhub.io/api/v1/sps/customers Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

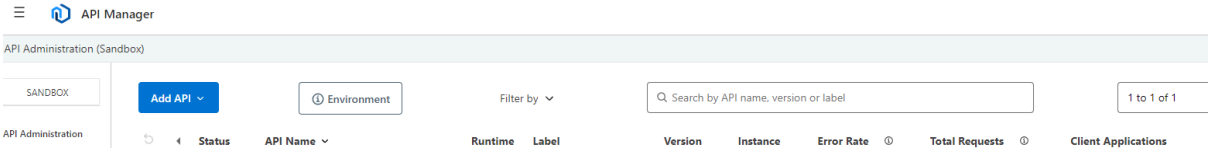
Body Cookies Headers (6) Test Results 200 OK 818 ms 1.06 KB Save Response

Pretty Raw Preview Visualize JSON

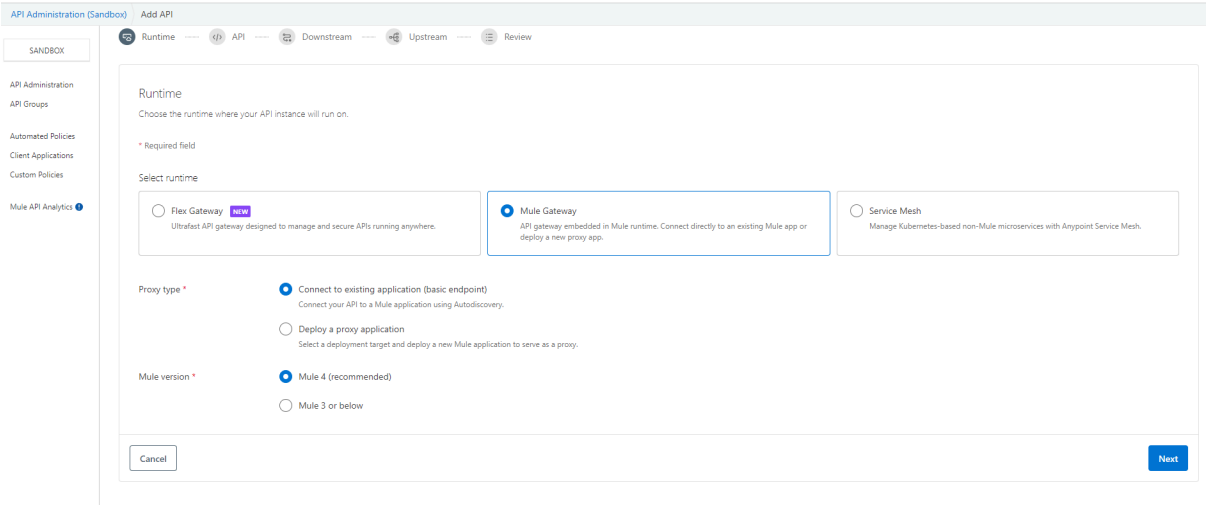

```
[
    {
        "FirstName": "Maura",
        "LastName": "Wagstaffe",
        "Email": "mwagstaffe@npr.org",
        "Company": "Kazu"
    },
    {
        "FirstName": "Myrwyn",
        "LastName": "Eliet",
        "Email": "meliet1@buzzfeed.com",
        "Company": "Skibox"
    },
    {
        "FirstName": "Myrwyn",
        "LastName": "Eliet",
        "Email": "meliet1@buzzfeed.com",
        "Company": "Skibox"
    },
    {
        "FirstName": "Jenn",
        "LastName": "White",
        "Email": "jwhite@wamu.org",
        "Company": "NPR"
    },
    {
        "FirstName": "Kelly",
        "LastName": "McEvers",
        "Email": "kmcevers@npr.org",
        "Company": "NPR"
    },
    {
        "FirstName": "Audie",
        "LastName": "Cornish",
        "Email": "audie@npr.org",
        "Company": "NPR"
    },
    {
        "FirstName": "Ari",
        "LastName": "Shapiro",
        "Email": "ari@npr.org",
```

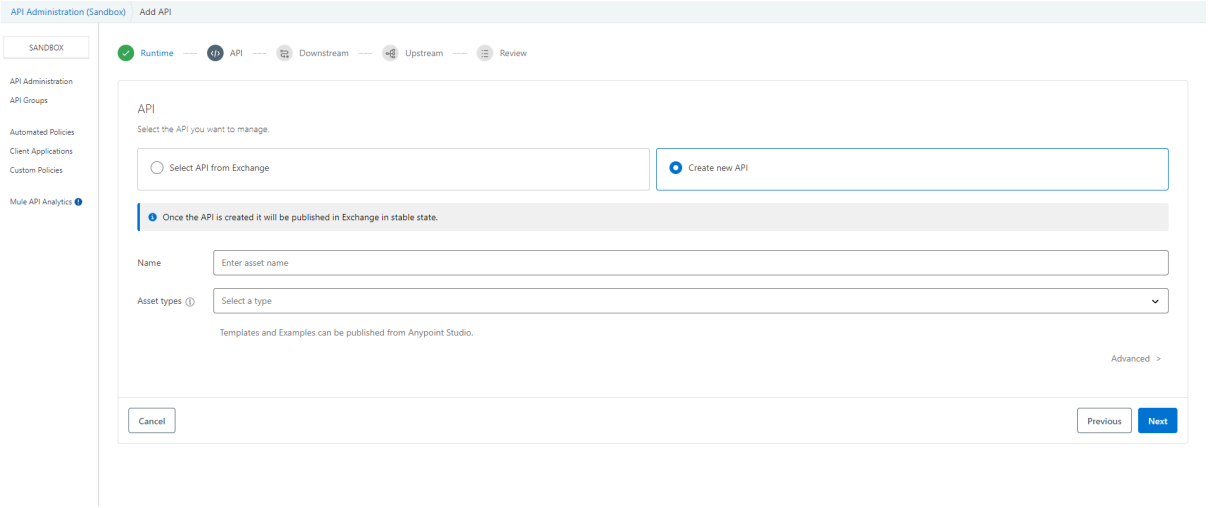
Creating API in "API Manager"

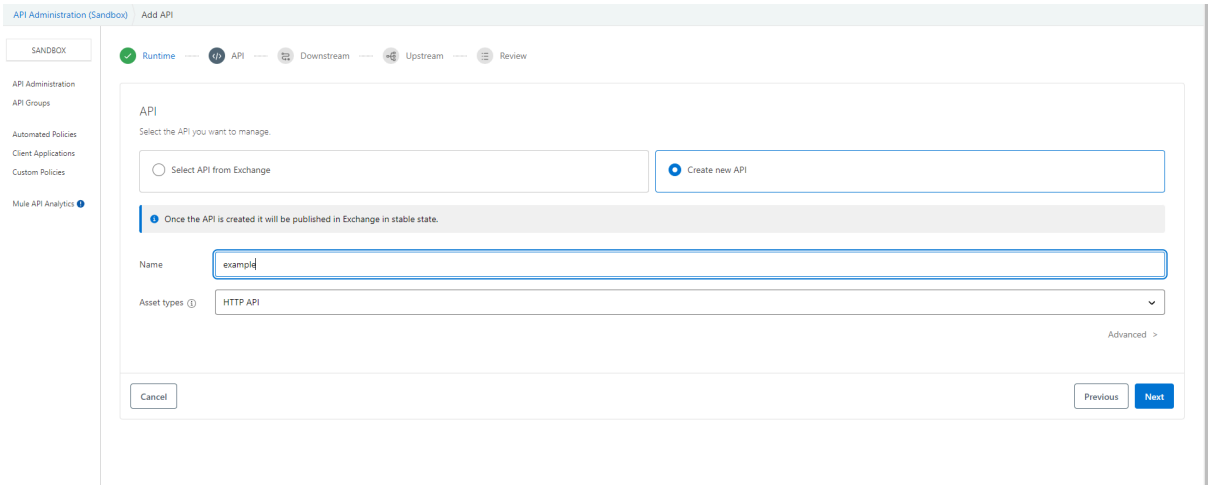
Once logged into AnyPoint Platform, we will enter API Manager and click on "Add API" and “Add New API”

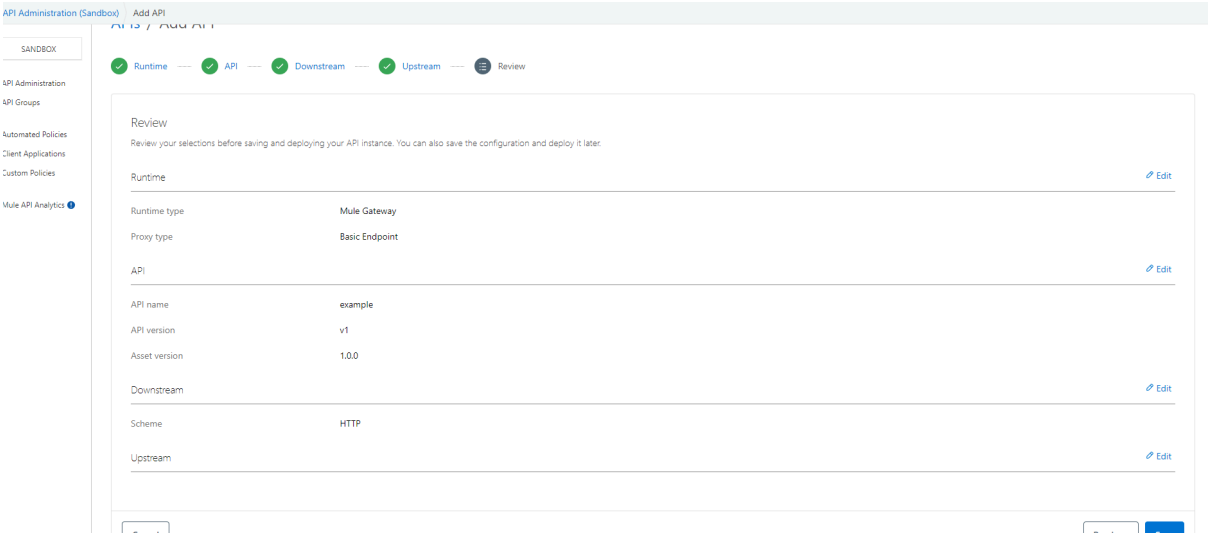


Then we will select the following option

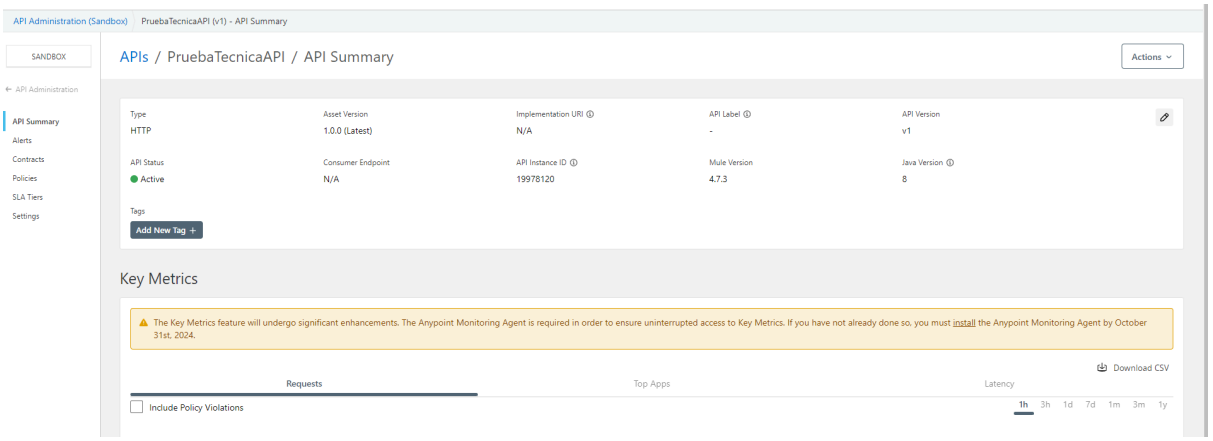




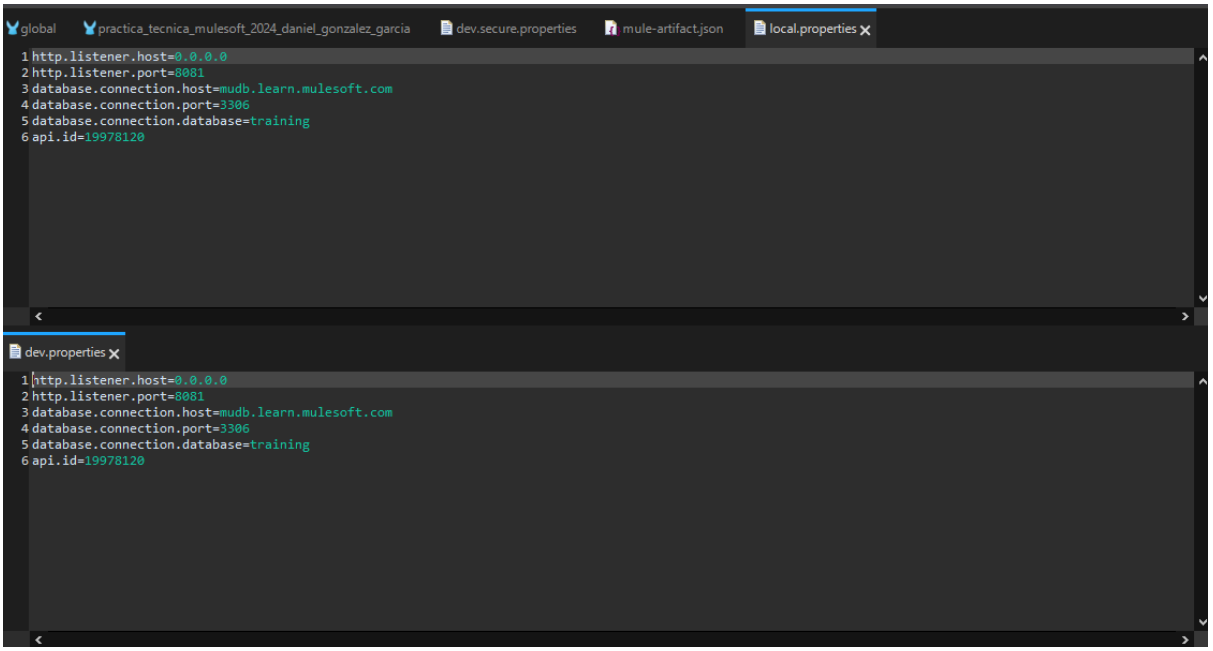




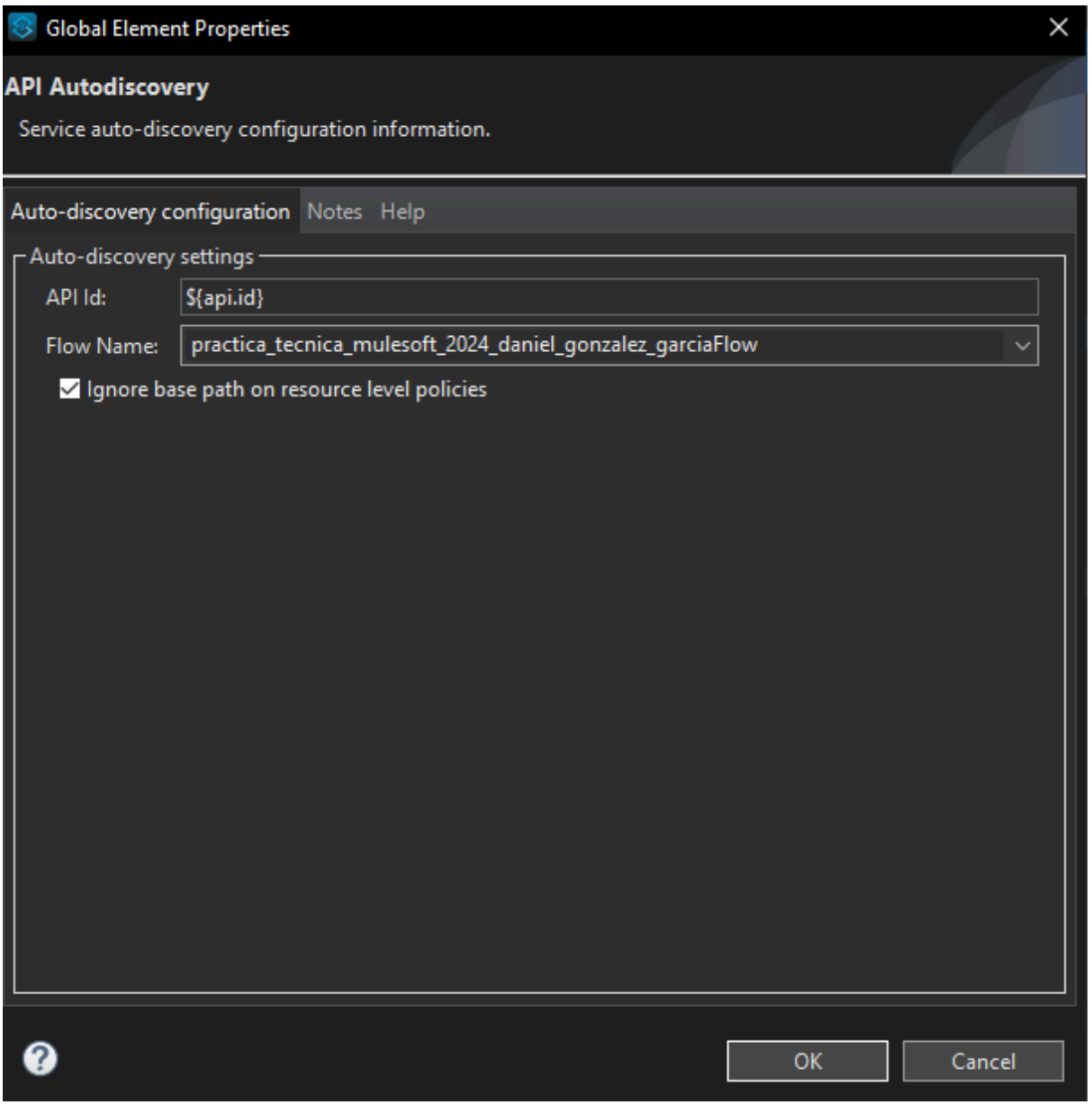
Which leaves us with the following screen



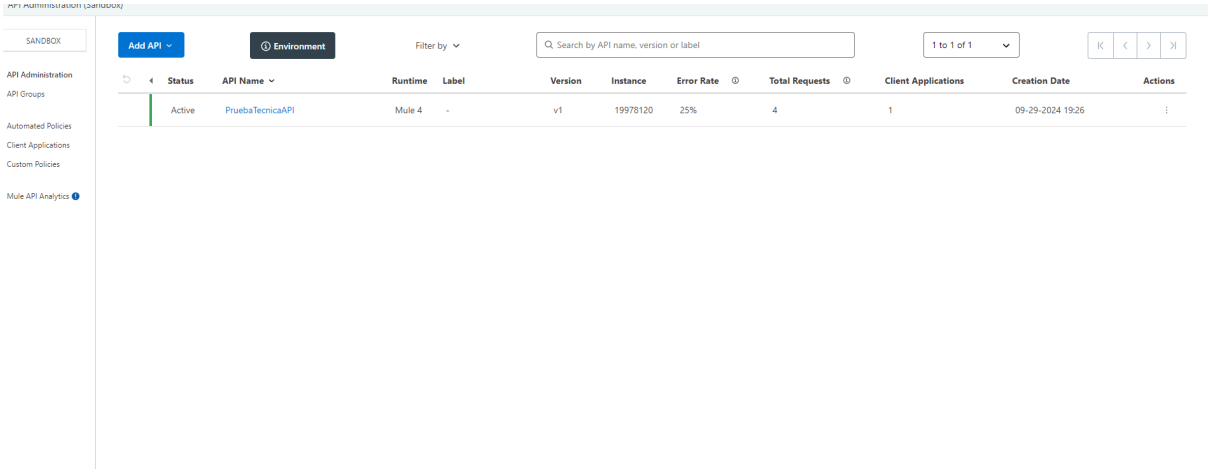
The data we need for now is the "API instance ID"
Now in our "dev.properties" and "local.properties" files we must add this id and they would look like this:



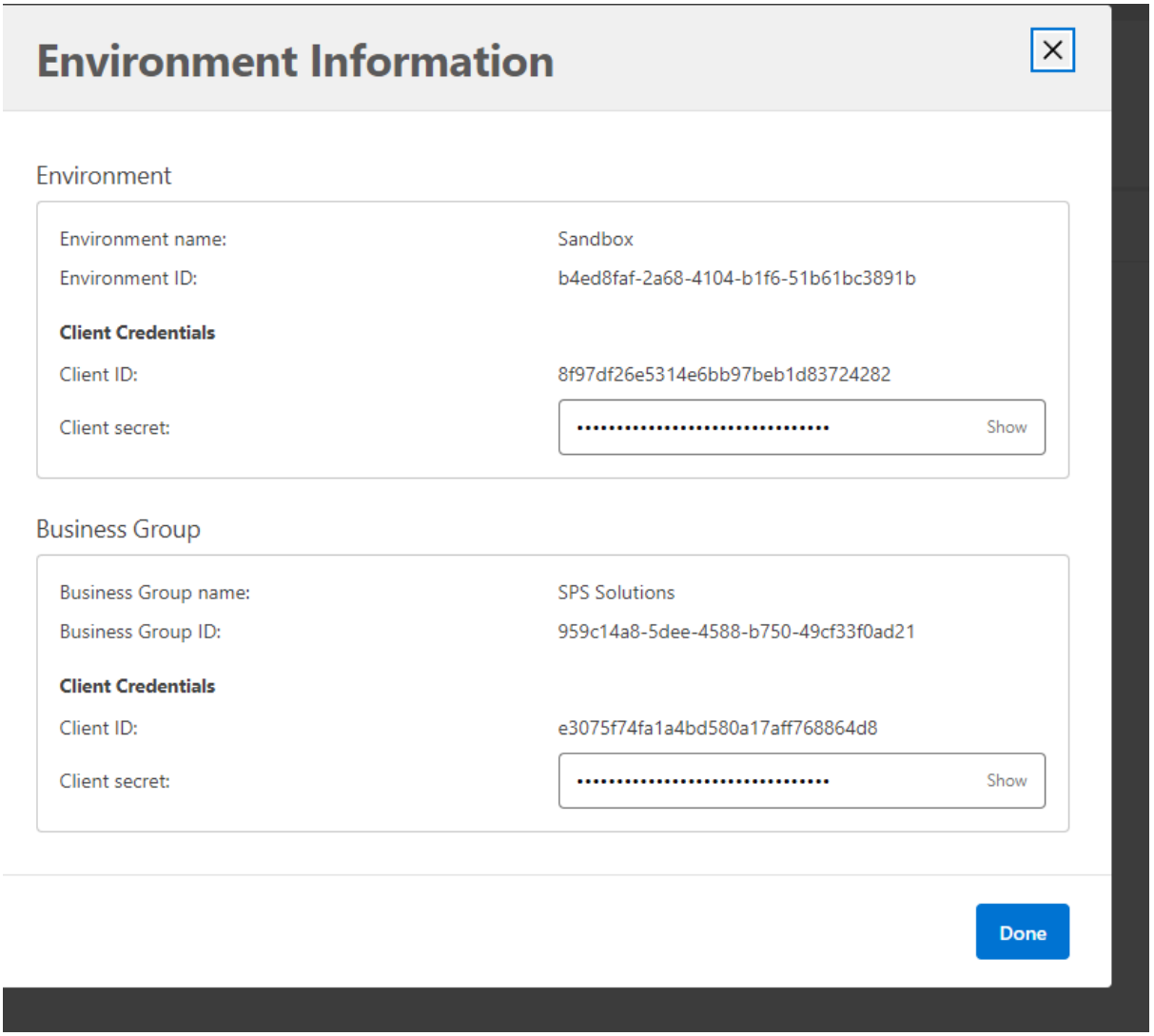
In our global file we will add a new configuration element "API Autodiscovery"



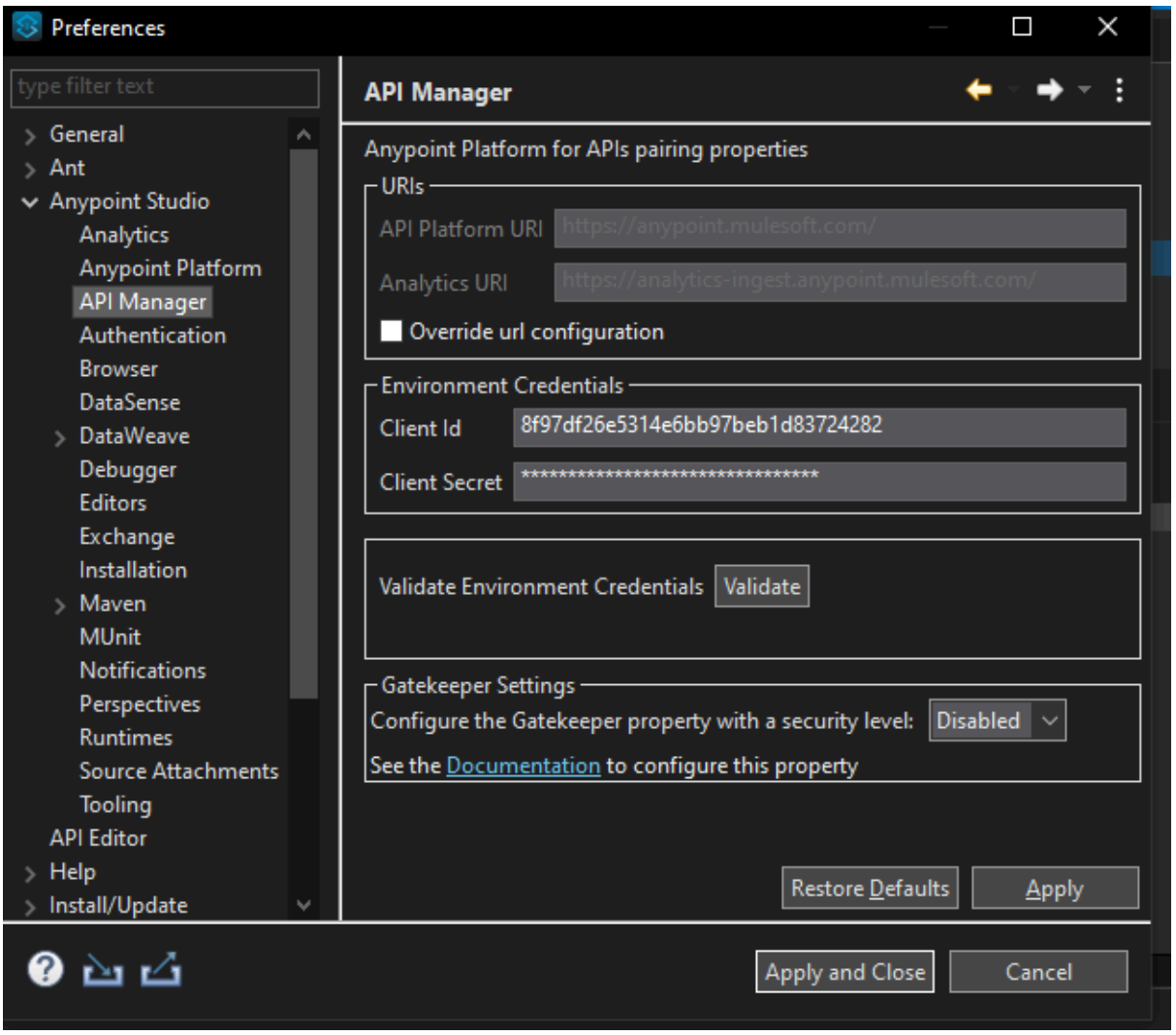
Now on the API Manager selection screen we must enter the following window



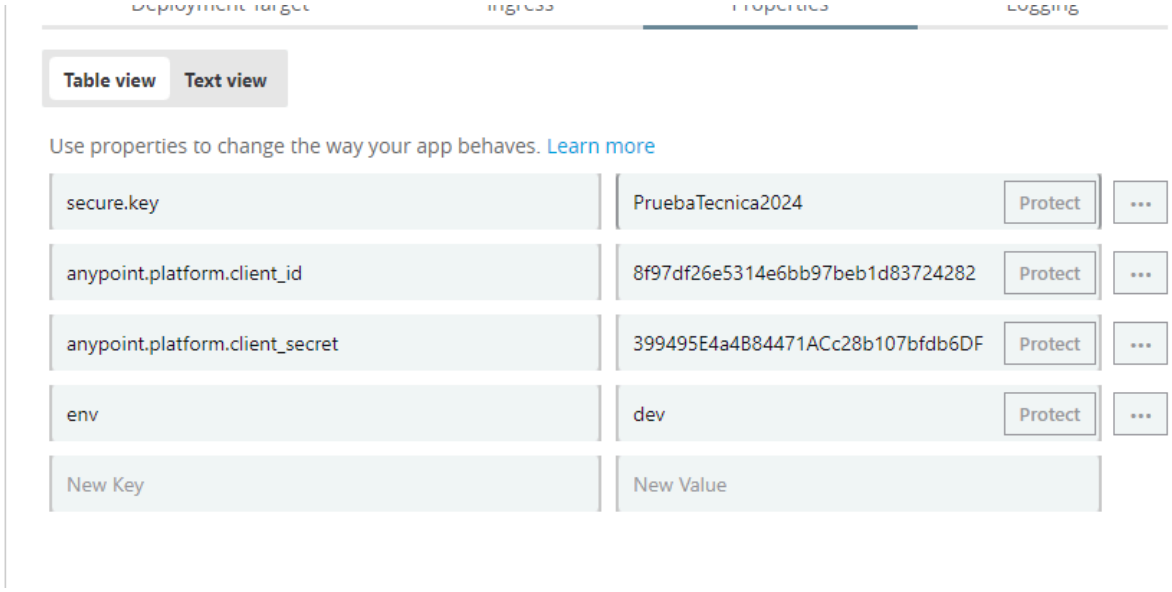
And get the data: "Client ID" and "Client Secret" from the "Sandbox" environment



and add them to AnyPoint Studio preferences

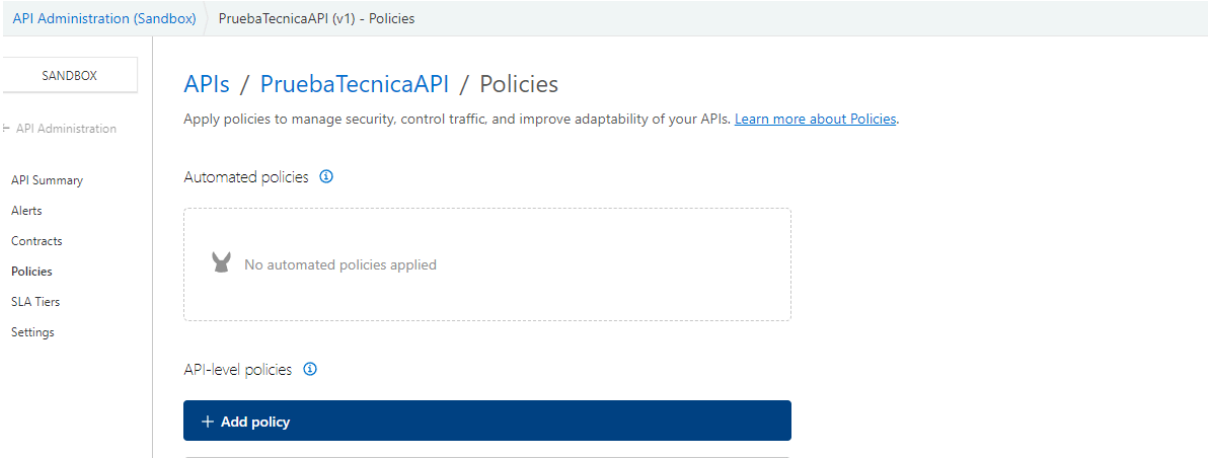


As with other variables, we must add them to the mule-artifact.json file and to the CloudHub variables.

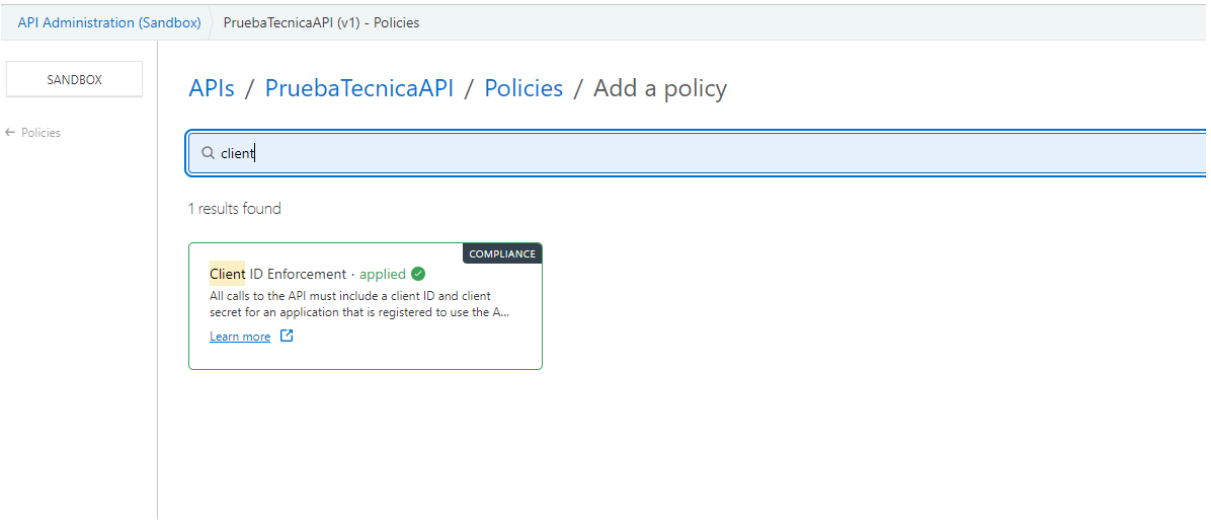


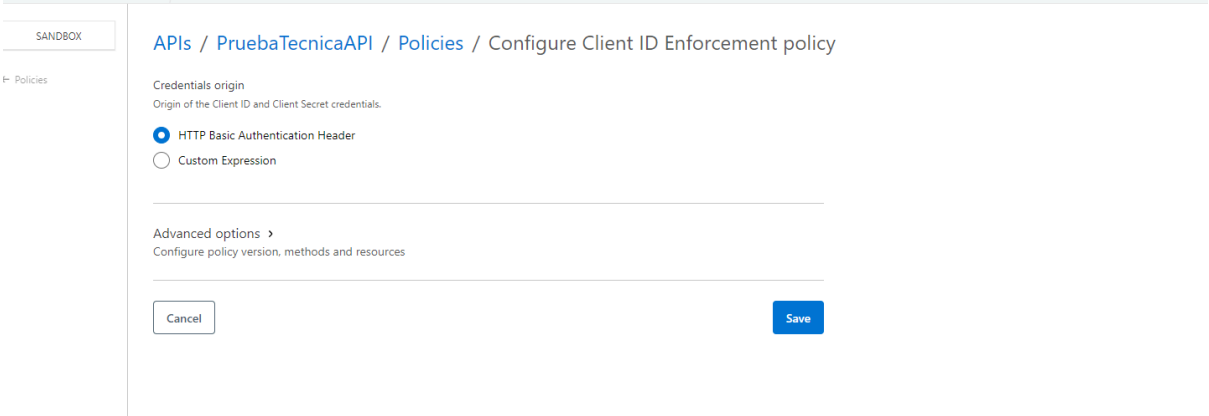
Applying Client ID enforcement policy in API Manager

In the newly created API in the "Policies" module we select the following option

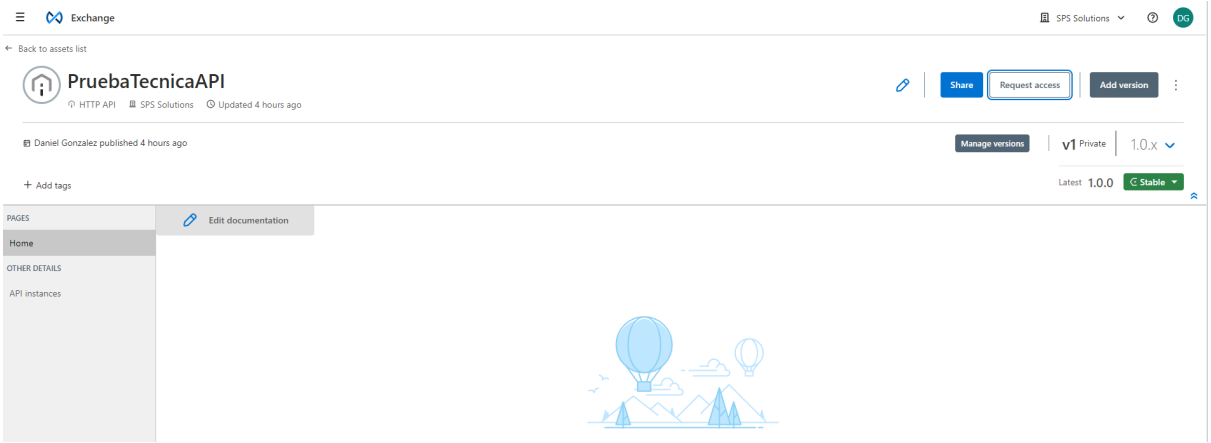


We look for "Client" and choose the option that appears.

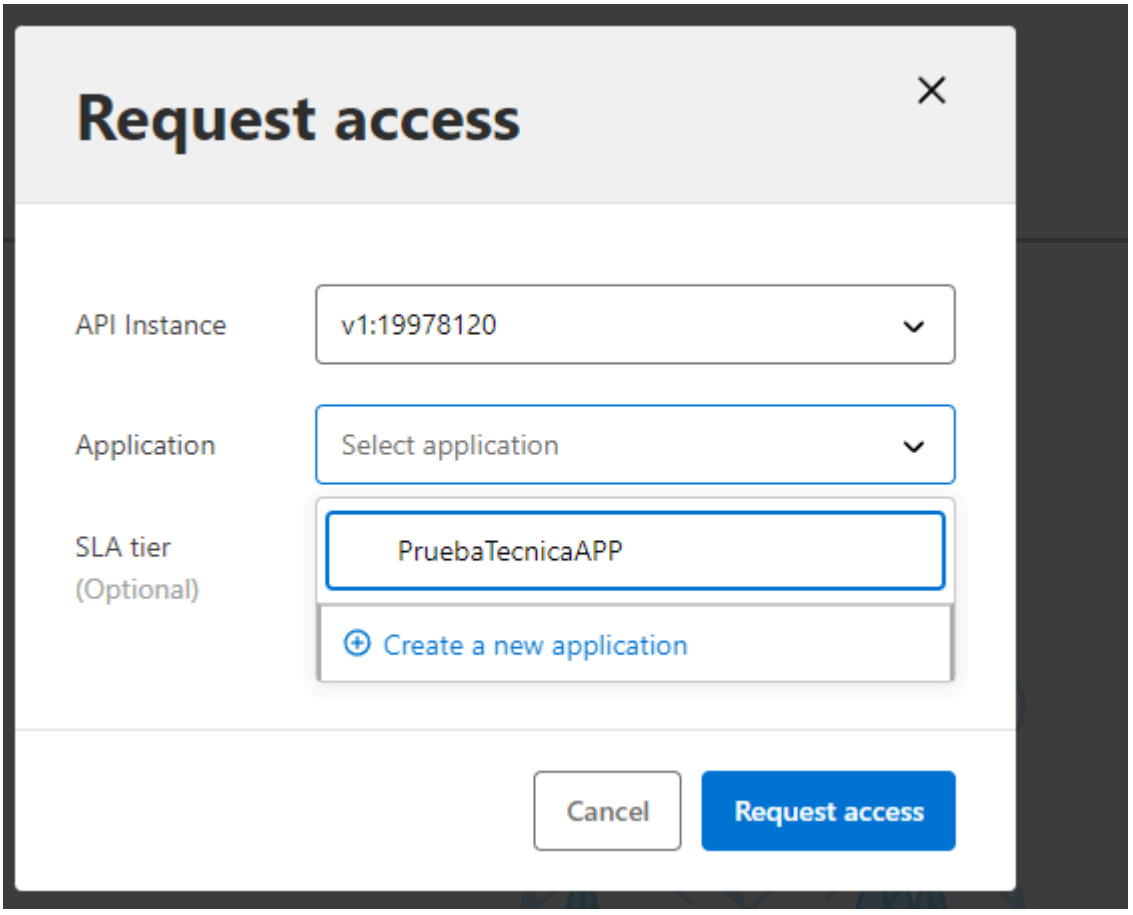




Now we will go to "Exchange" of AnyPoint Platform and select the newly created one, and click on "Request Access"



We select the "API Instance" and create a new application with a name that tells us who it belongs to.

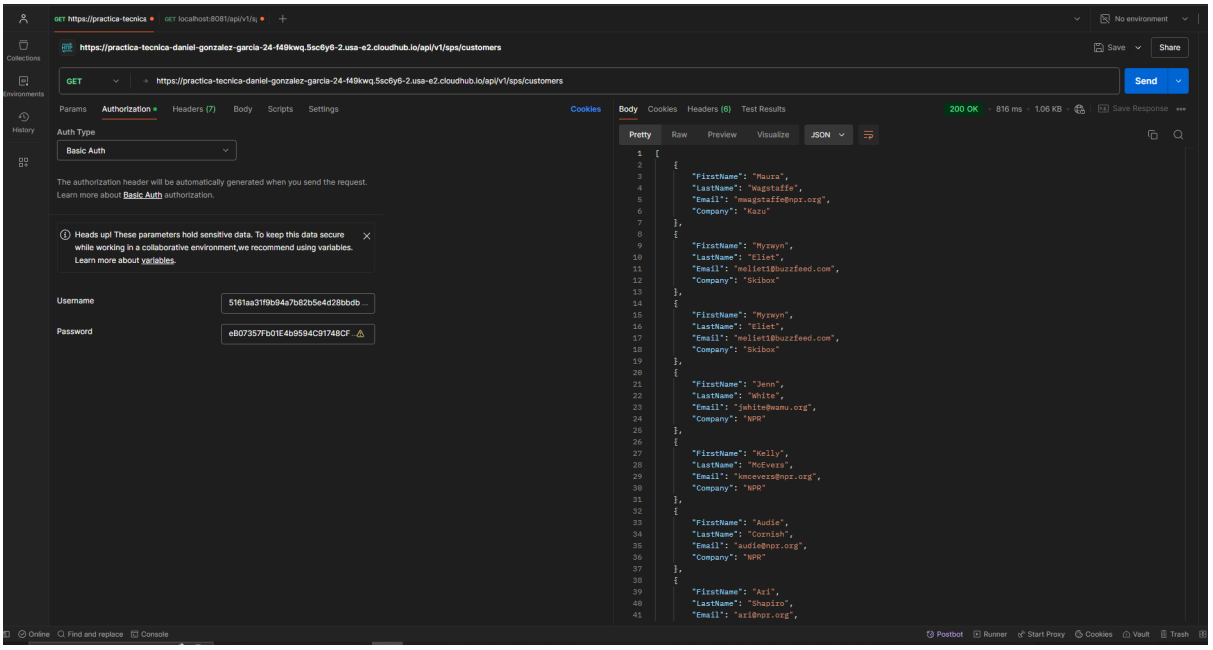


After this it will give us two very important pieces of data: ClientID and ClientSecret

For this specific application these are the data:

Client ID: 5161aa31f9b94a7b82b5e4d28bbdbd65
Client Secret: eB07357Fb01E4b9594C91748CFe0F062

With this the authentication is ready, now when making a request we must add this data as follows:



This must be done both locally and in the deployed version.