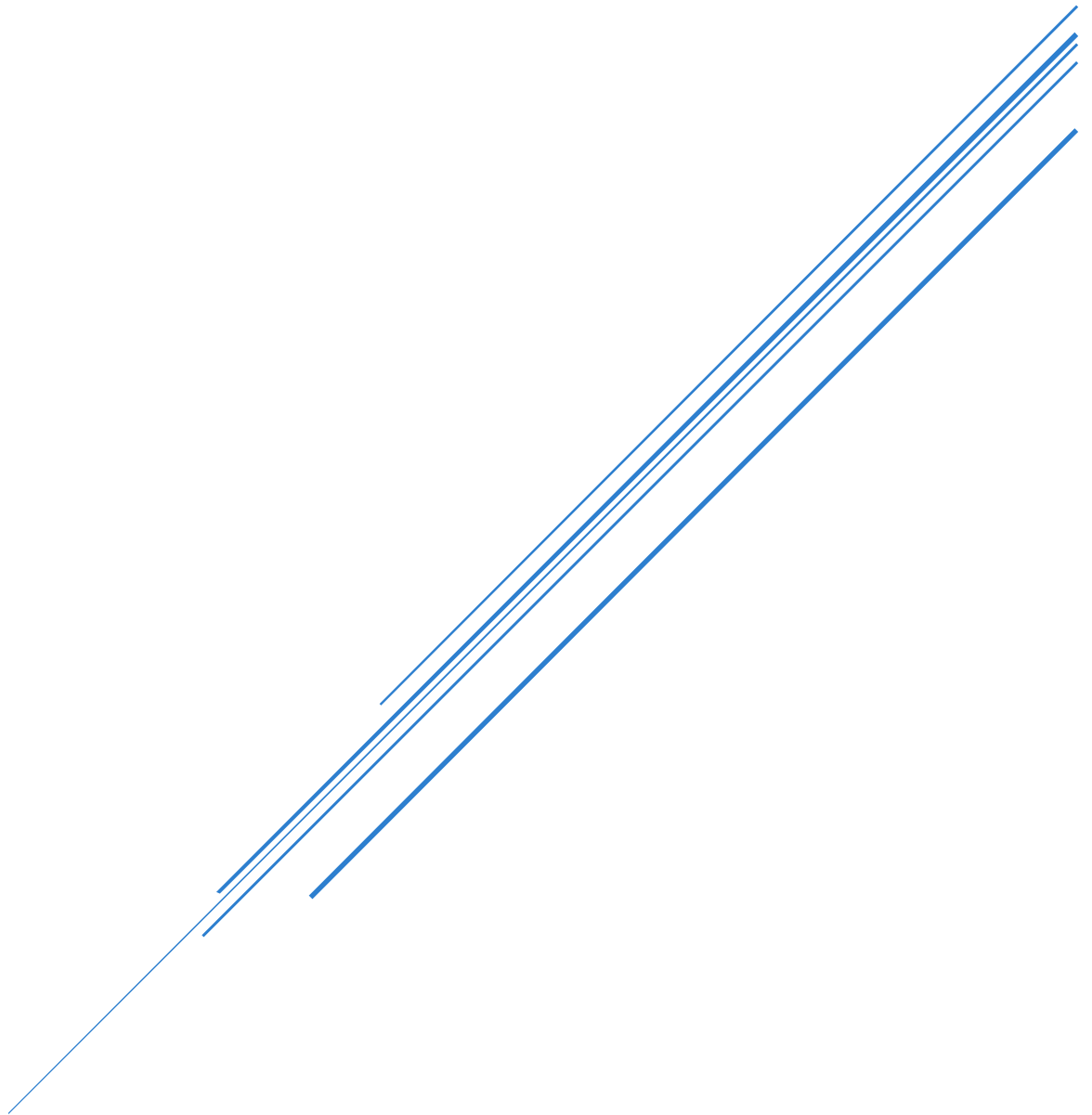


PRÁCTICA – MULESOFT TRAINEE

Alan Eduardo Gamboa Del Angel



Documentación
V.1.0.0

Contenido

Introducción	2
Requisitos.....	2
Implementación.....	3
1.1 Crear un proyecto nuevo de Mulesoft	3
1.2 Asignar un Project Name.....	4
1.3 Creación de archivo de configuración global	4
1.4 Creacion de archivos properties	5
1.6 Configuración del archivo global.xml.....	9
1.7 Creación de la conexión a la base de datos	10
1.8 Agregando seguridad a los parámetros de la base de datos.....	18
1.9 Agregando un Logger.....	25
2.1- Despliegue de la aplicación.....	27
3.- Especificación de la API.....	32
3.3 Creación de Endpoints.....	35
3.4 Publicando la especificación en Exchange	39
4.- Creación de una API en API Manager	41
5. Configuración de API Autodiscovery	45
5.2 Client IDy Client Secret	47
5.3 Redespliegue de nuestra aplicación	49
6.1- Aplicación de política Client ID Enforcement.....	51
6.2 Obtención de credenciales.....	54
Recursos.....	56

Introducción

El presente documento describe el desarrollo detallado de una API en MuleSoft cuyo objetivo es exponer la información de los clientes almacenada en una base de datos MySQL. También se explica el desarrollo e implementaciones utilizando distintas herramientas ofrecidas por Anypoint Platform.

Requisitos

Herramientas Necesarias

- Anypoint Studio 7

Para el desarrollo de esta práctica será necesario descargar Anypoint Studio 7. Se puede descargar del siguiente enlace(es necesario crear una cuenta en Anypoint

Platform para poder descargar el IDE y hacer uso de los servicios ofrecidos por dicha plataforma):

[Download Anypoint Studio & Mule | MuleSoft](#)

- Postman

Para poder realizar las pruebas del funcionamiento de nuestra API se utilizará la herramienta Postman que puede ser descargada mediante el siguiente enlace:

[Download Postman | Get Started for Free](#)

Implementación

1.- Configuración del proyecto en Anypoint Studio

1.1 Crear un proyecto nuevo de Mulesoft

Para crear un nuevo proyecto de Mulesoft seguiremos el siguiente flujo: File -> New -> Mule Project

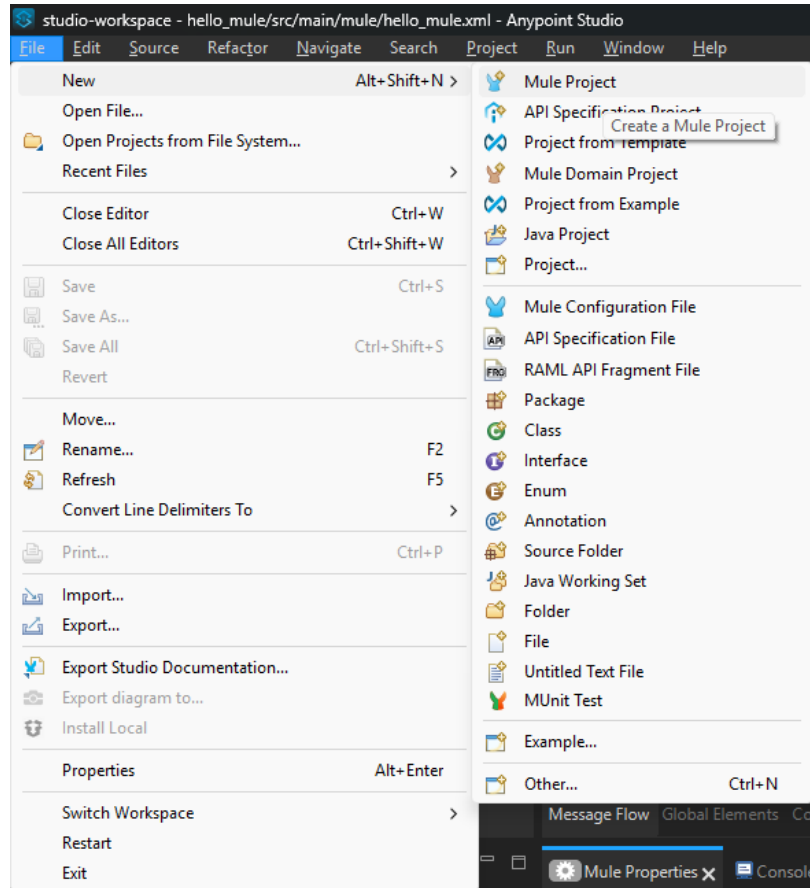


Ilustración 1. Crear nuevo proyecto

1.2 Asignar un Project Name

Para este caso asignaremos el Project Name como “Customers_API” y daremos click en “Finish”

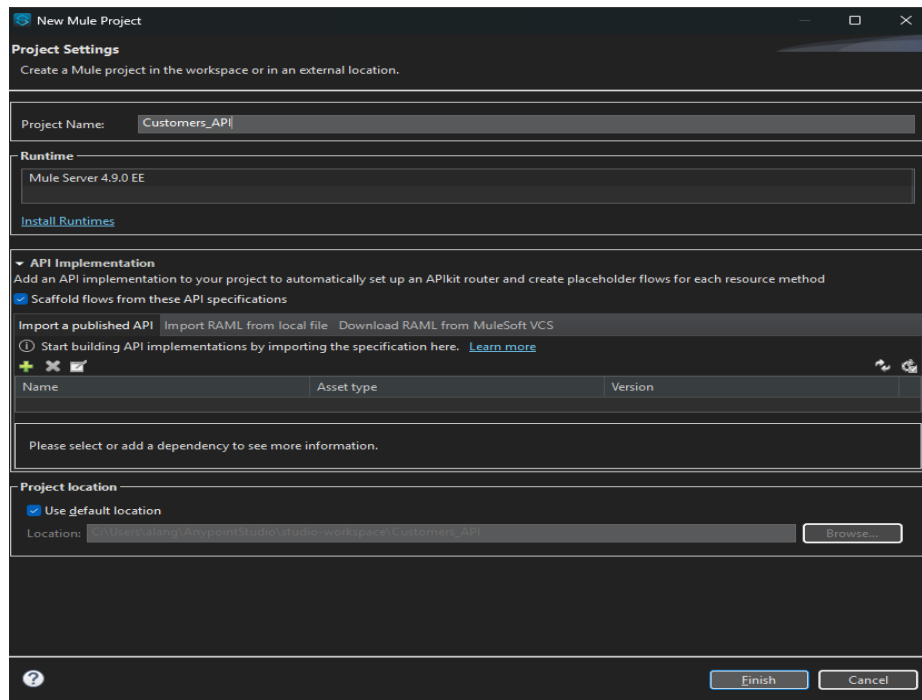


Ilustración 2. Project Settings

1.3 Creación de archivo de configuración global

Dentro del directorio **src/main/mule**, agregaremos un archivo de configuración global que contendrá las propiedades del proyecto dando click izquierdo dentro de en New -> Mule Configuration File.

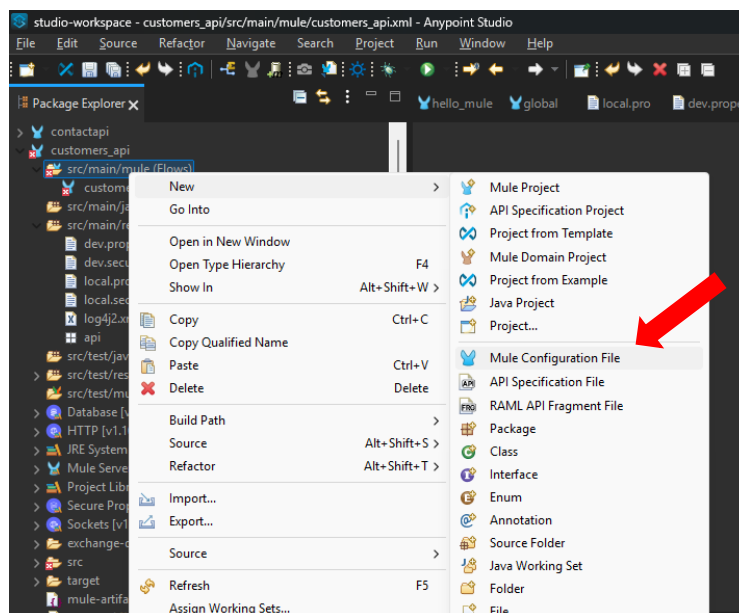


Ilustración 3. Creación de archivo global.xml

Dicho archivo será llamado “gobal”. Esto se hará con la intención de que el archivo **global.xml** contenga todas las propiedades de configuración del proyecto, mientras que el archivo **customers_api.xml** sólo contenga el flujo de nuestra aplicación.

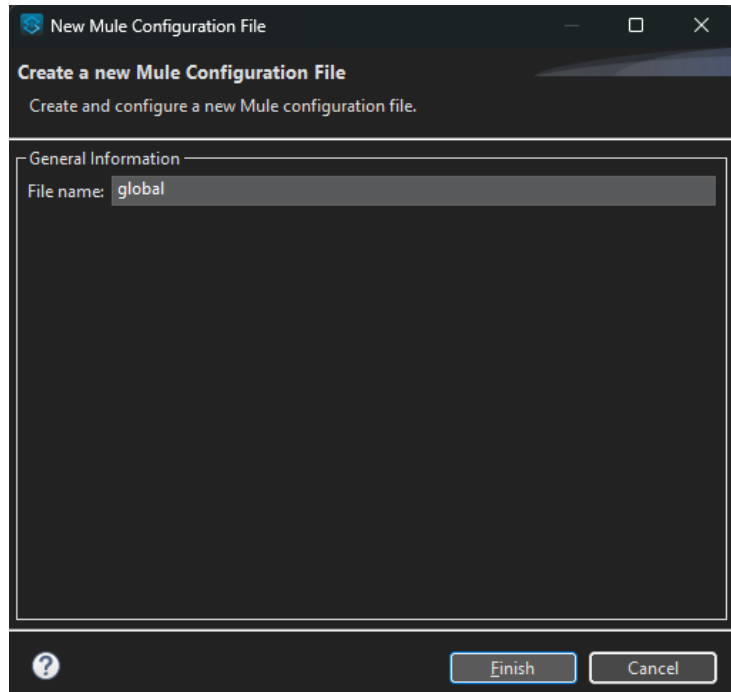


Ilustración 4. New Mule Configuration File

1.4 Creacion de archivos properties

Dentro del directorio **src/main/resources** crearemos 4 archivos dando click derecho en **New -> File**

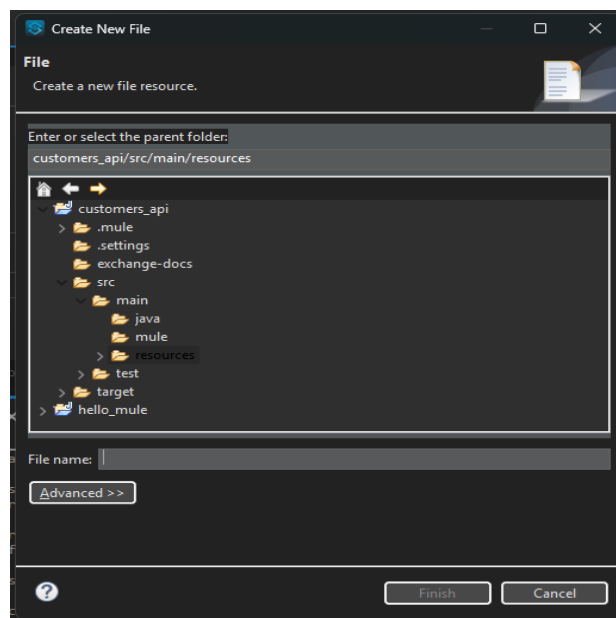


Ilustración 5. Directorio src/main/resources

Los cuatro archivos creados serán los siguientes:

- local.properties
- dev.properties
- local.secure.properties
- dev.secure.properties

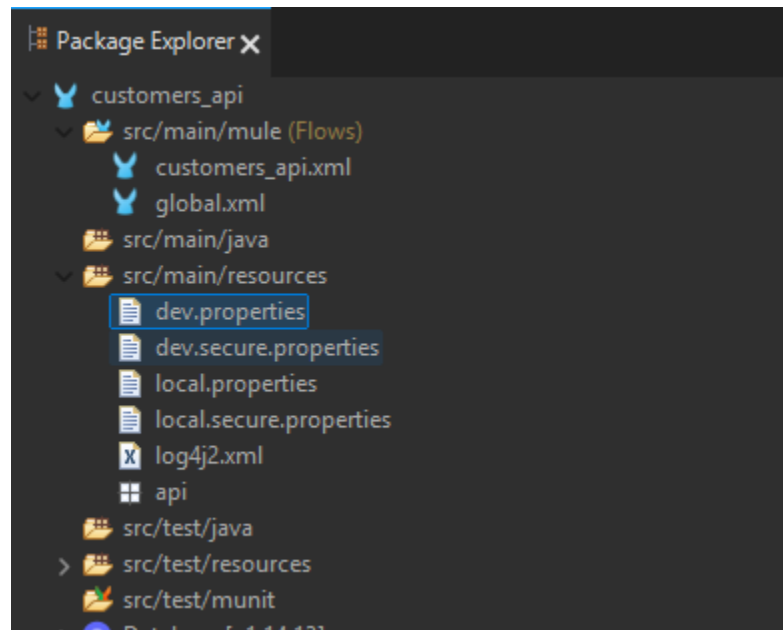


Ilustración 6. Estructura dentro de src/main/resources

En secciones posteriores explicaremos la utilidad de los archivos **local.secure.properties** y **dev.secure.properties**. Por el momento, nos enfocaremos en los archivos *local.properties* y *dev.properties*.

En cada uno de ellos agregaremos las siguientes propiedades:

```
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
```

Ilustración 7. Archivo local.properties

```
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
```

Ilustración 8. Archivo dev.properties

1.5 Creación un HTTP Listener

Dentro de nuestro archivo *customers_api.xml* del lado derecho de nuestro Anypoint Studio podremos observar una sección llamada **Mule Palette**. Dentro de esta sección podremos encontrar diversos componentes para desarrollar nuestros flujos.

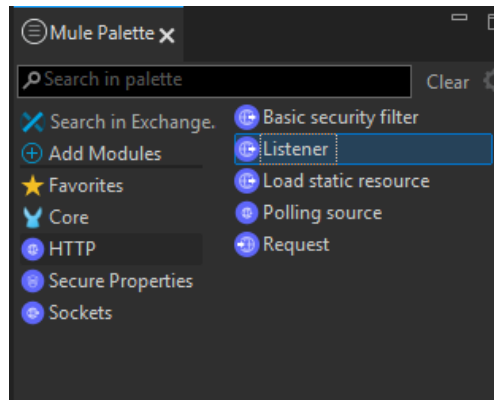


Ilustración 9. Mule Palette

En este caso utilizaremos el componente *Listener* de la Sección **HTTP**. Daremos click en el componente de Listener y lo arrastraremos y soltaremos en la vista de Message Flow

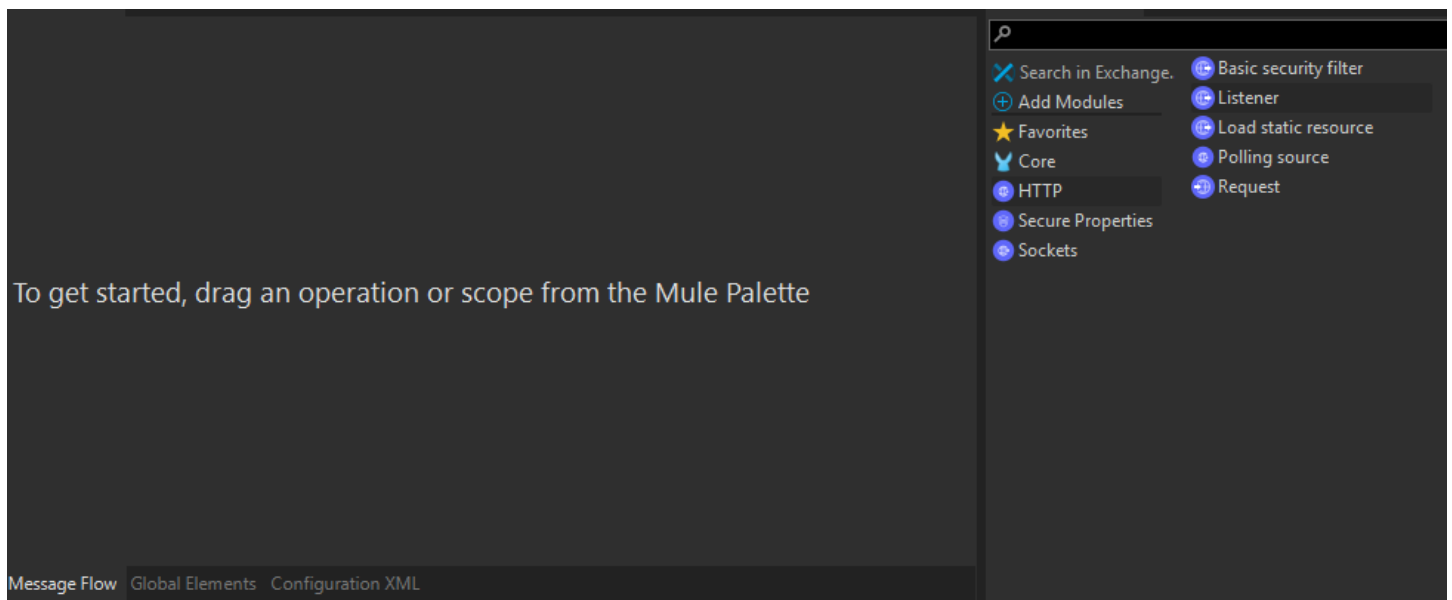


Ilustración 10. HTTP Listener

Lo cual nos dará como resultado la siguiente vista:

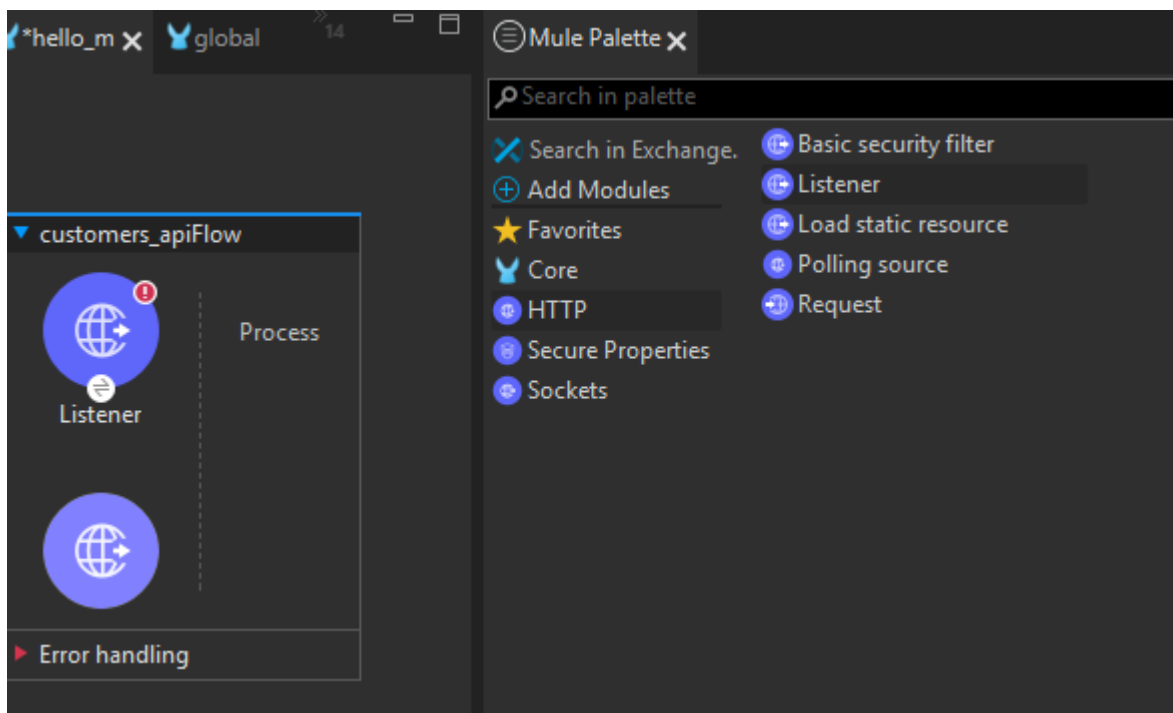


Ilustración 11. Message Flow

Damos click en el Listener creado, y en el cuadro inferior en la sección de General en la propiedad de path colocaremos la siguiente ruta **/api/v1/sps/customers**

2.

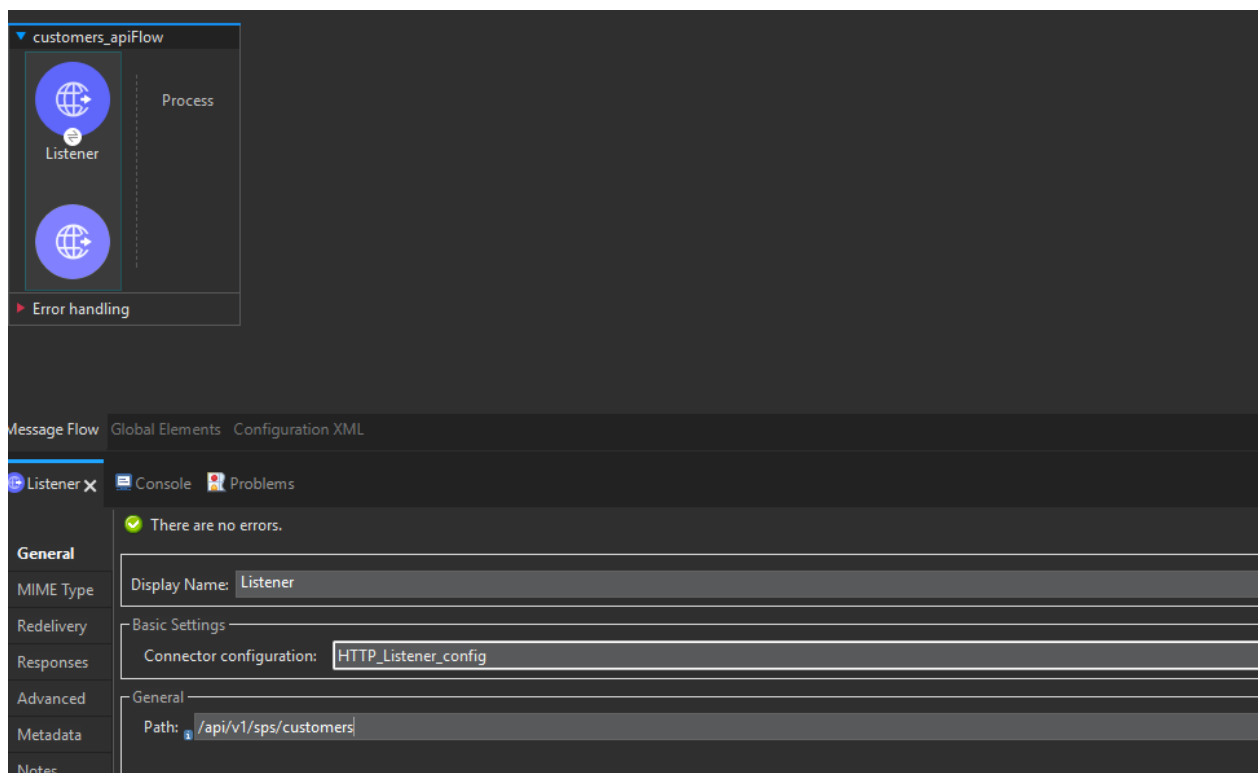
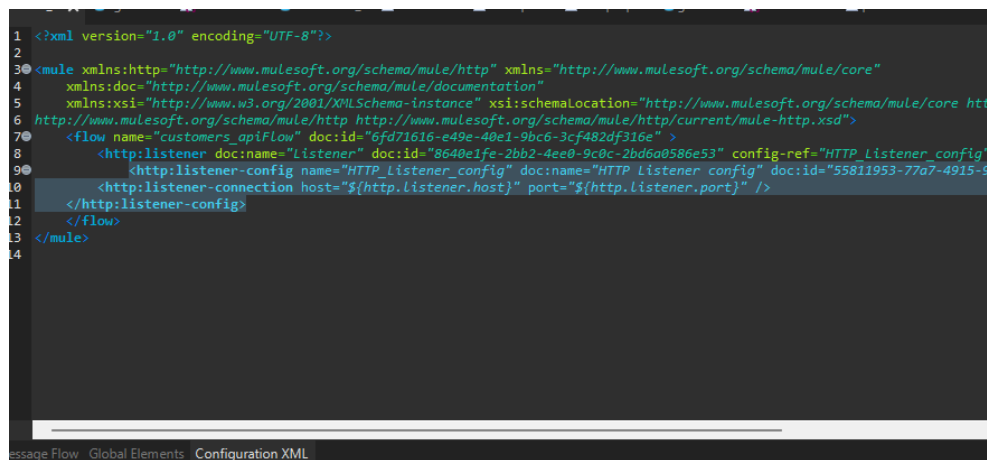


Ilustración 12. Listener Path

1.6 Configuración del archivo global.xml

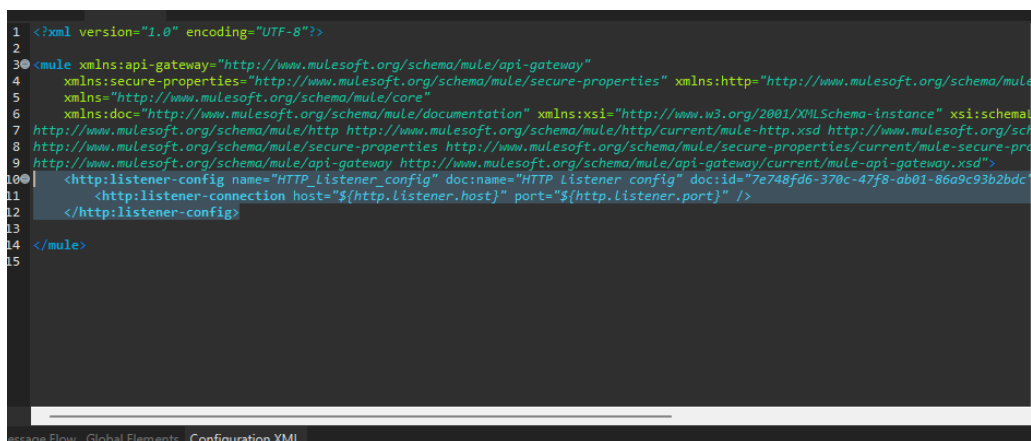
Dentro del archivo customers_api.xml en la vista ConfigurationXML, vamos a transferir la configuración nuestro HTTP Listener hacía el archivo global.xml.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns="http://www.mulesoft.org/schema/mule/core"
4     xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
5     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd">
6     <flow name="customers_apiFlow" doc:id="6fd71616-e49e-40e1-9bc6-3cf482df316e">
7         <http:listener doc:name="Listener" doc:id="8640e1fe-2bb2-4ee0-9c0c-2bd6a0586e53" config-ref="HTTP Listener config">
8             <http:listener-config name="HTTP Listener config" doc:name="HTTP Listener config" doc:id="55811953-77a7-4915-9" />
9             <http:listener-connection host="${http.listener.host}" port="${http.listener.port}" />
10        </http:listener-config>
11    </flow>
12</mule>
```

Ilustración 13. Vista XML de customers_api.xml

Cortamos la sección mostrada en la Ilustración 13 y lo pegamos dentro del archivo global.xml dentro de las etiquetas <mule> </mule>



```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:api-gateway="http://www.mulesoft.org/schema/mule/api-gateway"
4     xmlns:secure-properties="http://www.mulesoft.org/schema/mule/secure-properties" xmlns="http://www.mulesoft.org/schema/mule/core"
5     xmlns:doc="http://www.mulesoft.org/schema/mule/documentation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd http://www.mulesoft.org/schema/mule/secure-properties http://www.mulesoft.org/schema/mule/secure-properties/current/mule-secure-properties.xsd http://www.mulesoft.org/schema/mule/api-gateway http://www.mulesoft.org/schema/mule/api-gateway/current/mule-api-gateway.xsd">
6     <http:listener-config name="HTTP Listener config" doc:name="HTTP Listener config" doc:id="7e748fd6-370c-47f8-ab01-86a9c93b2bdc" />
7     <http:listener-connection host="${http.listener.host}" port="${http.listener.port}" />
8 </mule>
```

Ilustración 14. Archivo global.xml

Para comprobar que no haya errores, dentro del mismo archivo global.xml cambiamos a la vista de Global Elements como se muestra en la siguiente figura.

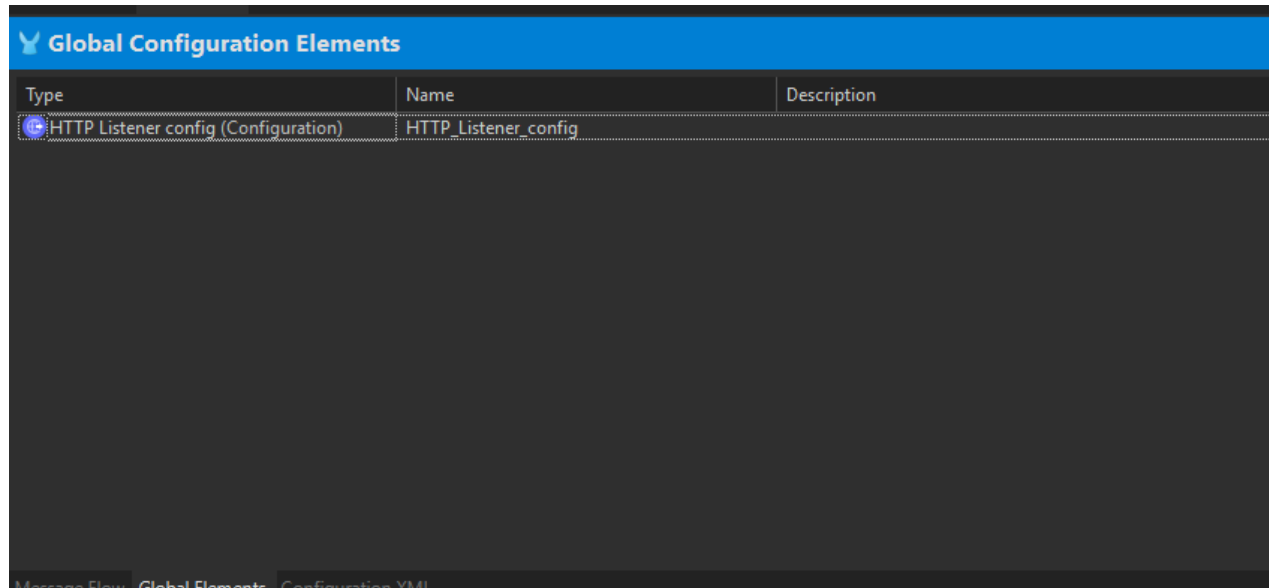


Ilustración 15. Global Configuration Elements

1.7 Creación de la conexión a la base de datos

Dentro de nuestro archivo customers_api.xml, utilizando Mule Palette, daremos click en la opción *Add Modules* y posteriormente agregaremos el módulo de *Database* arrastrándolo hacia la columna del lado izquierdo.

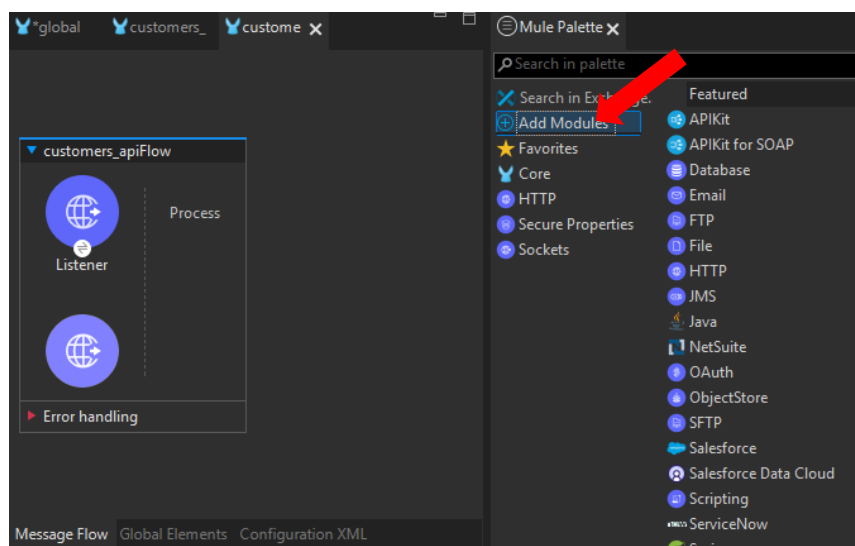


Ilustración 16. Modulo Database

Dado que estaremos haciendo una consulta hacia una base de datos, utilizaremos el modulo Select y lo arrastraremos al diagrama de customers_apiFlow

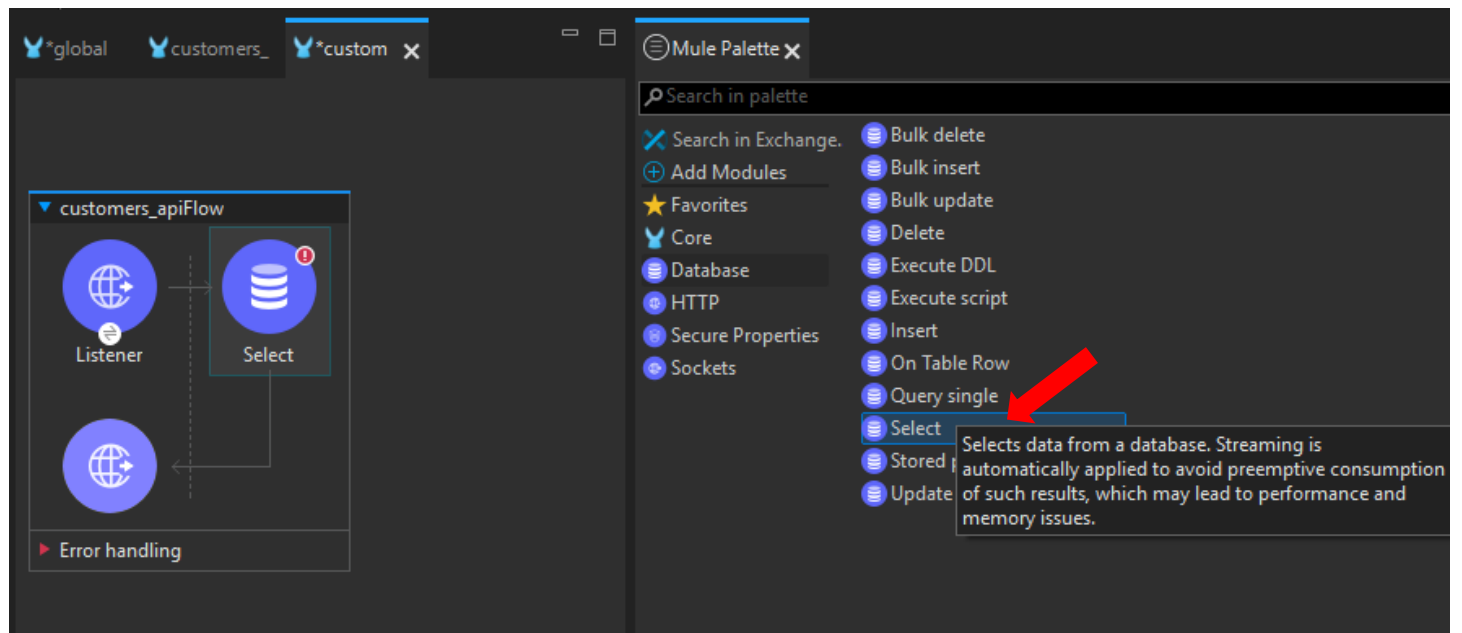


Ilustración 17. Módulo Select

Agregaremos los parámetros para configurar la conexión a la base de datos.



Ilustración 18. Basic Settings

De ser necesario, agregaremos las librerías recomendadas para MySQL Connection.

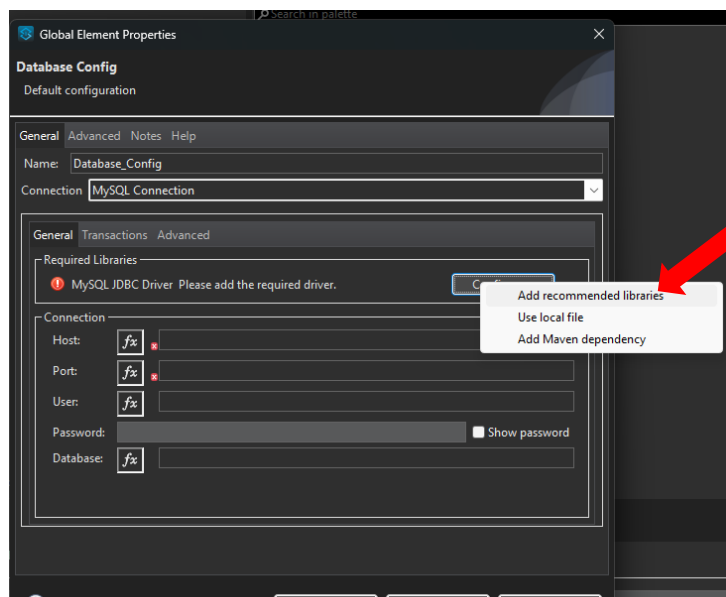


Ilustración 19. Database Connection

Agregaremos los parámetros que se muestran en la imagen en la configuración del módulo.

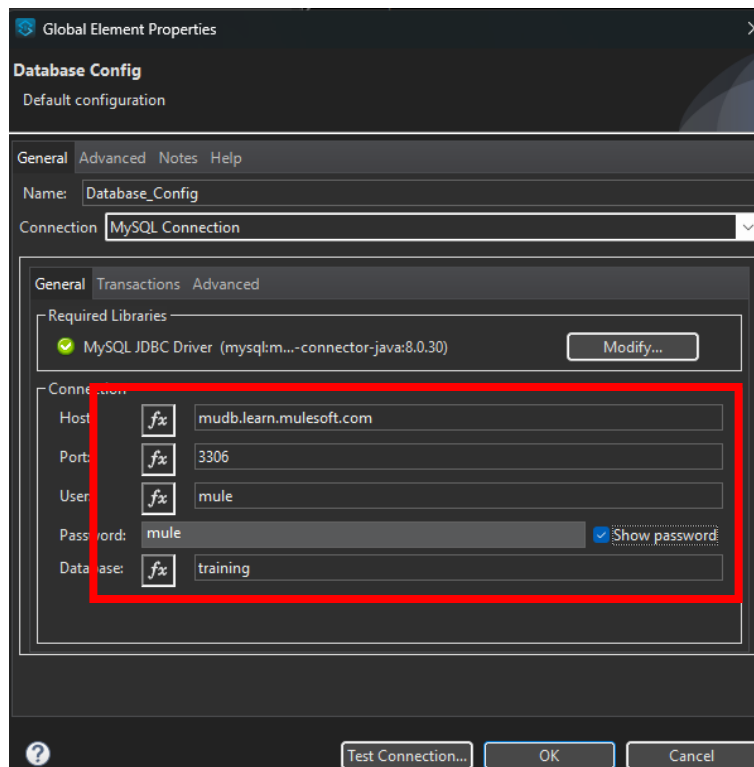


Ilustración 20. Database Config

De la misma manera que hicimos con las propiedades de HTTP Listener, vamos a cortar las propiedades del archivo de customers_api.xml al global.xml, quedando de la siguiente manera:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:db="http://www.mulesoft.org/schema/mule/db" xmlns:http="http://www.mulesoft.org/schema/mule/http"
4   xmlns="http://www.mulesoft.org/schema/mule/core"
5   xmlns:doc="http://www.mulesoft.org/schema/mule/documentation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.mulesoft.org
6   http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
7   http://www.mulesoft.org/schema/mule/db http://www.mulesoft.org/schema/mule/db/current/mule-db.xsd">
8   <db:config name="Database_Config" doc:name="Database Config" doc:id="ac17e2bc-b27b-4c3b-8c50-1c9c005e6a31" >
9     <db:mysql-connection host="mudb.learn.mulesoft.com" port="3306" user="mule" password="mule" database="training" />
10  </db:config>
11  <flow name="customers_apiFlow" doc:id="6fd71616-e49e-40e1-9bc6-3cf482df316e" >
12    <http:listener doc:name="listener" doc:id="8640e1fe-2bb2-4ee0-9c0c-2bd6a0586e53" config-ref="HTTP_Listener_config" path="/api/v1/sps/customers"/>
13    <db:select doc:name="Select" doc:id="a140e6e5-216c-4e87-94a0-6488d6ce043f" config-ref="Database_Config">
14      <db:sql >SELECT * FROM customers]]&gt;&lt;/db:sql&gt;
15    &lt;/db:select&gt;
16  &lt;/flow&gt;
17 &lt;/mule&gt;
18</pre></div><div data-bbox="356 354 579 369" data-label="Caption"><p>Ilustración 21. customers_api.xml</p></div><div data-bbox="107 484 832 691" data-label="Code-Block"><img alt="Screenshot of the global.xml file in an IDE. The file contains XML configuration for a Mule application. Line 1 shows the XML declaration. Line 2 shows the root &lt;mule&gt; element with various xmlns attributes. Line 3 shows a &lt;http:listener&gt; element with attributes doc:name='HTTP Listener', doc:id='7e748fd6-370c-47f8-ab01-86a9c93b2bdc', and config-ref='HTTP_Listener_config'. Line 4 shows a &lt;http:listener-connection&gt; element with attributes host='http://localhost:8080' and port='8080'. Line 5 shows a &lt;db:config&gt; element with attributes name='Database_Config', doc:name='Database Config', and doc:id='5f19aa7f-46f1-4894-be0e-1f8aff282853'. Line 6 shows a &lt;db:mysql-connection&gt; element with attributes host='mudb.learn.mulesoft.com', port='3306', user='mule', password='mule', and database='training'. The application ends with &lt;/mule&gt;."/><pre>1 &lt;?xml version="1.0" encoding="UTF-8"?&gt;
2
3 &lt;mule xmlns:db="http://www.mulesoft.org/schema/mule/db"
4   xmlns:api-gateway="http://www.mulesoft.org/schema/mule/api-gateway"
5   xmlns:secure-properties="http://www.mulesoft.org/schema/mule/secure-properties" xmlns:http="http://www.mulesoft.org/schema/mule/http"
6   xmlns="http://www.mulesoft.org/schema/mule/core"
7   xmlns:doc="http://www.mulesoft.org/schema/mule/documentation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
8   http://www.mulesoft.org/schema/mule/db http://www.mulesoft.org/schema/mule/db/current/mule-db.xsd http://www.mulesoft.org/schema/mule/core http://www.mules
9   http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd http://www.mulesoft.org/schema/mule/secure-properties http://www.mulesoft.org/schema/mule/secure-properties/current/mule-secure-properties.xsd
10  http://www.mulesoft.org/schema/mule/api-gateway http://www.mulesoft.org/schema/mule/api-gateway/current/mule-api-gateway.xsd"&gt;
11  &lt;http:listener doc:name="HTTP Listener" doc:id="7e748fd6-370c-47f8-ab01-86a9c93b2bdc" &gt;
12    &lt;http:listener-connection host="http://localhost:8080" port="8080" /&gt;
13  &lt;/http:listener&gt;
14  &lt;db:config name="Database_Config" doc:name="Database Config" doc:id="5f19aa7f-46f1-4894-be0e-1f8aff282853" &gt;
15    &lt;db:mysql-connection host="mudb.learn.mulesoft.com" port="3306" user="mule" password="mule" database="training" /&gt;
16  &lt;/db:config&gt;
17 &lt;/mule&gt;
18</pre></div><div data-bbox="360 698 577 713" data-label="Caption"><p>Ilustración 22. Archivo global.xml</p></div><div data-bbox="921 919 953 936" data-label="Page-Footer"><p>14</p></div>
```

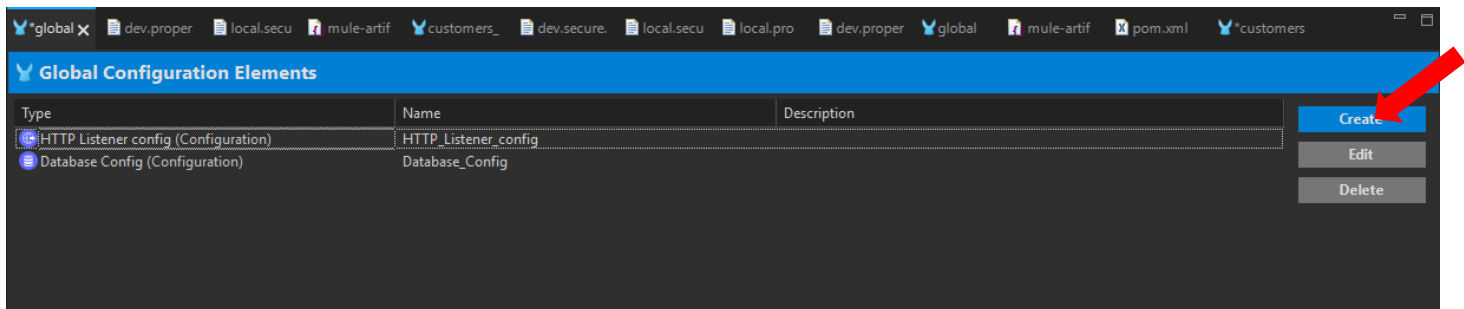


Ilustración 23. Global Configuration Elements

Finalmente, agregaremos el query que realizará nuestro componente de SELECT, en este caso será `SELECT * FROM Customers;`

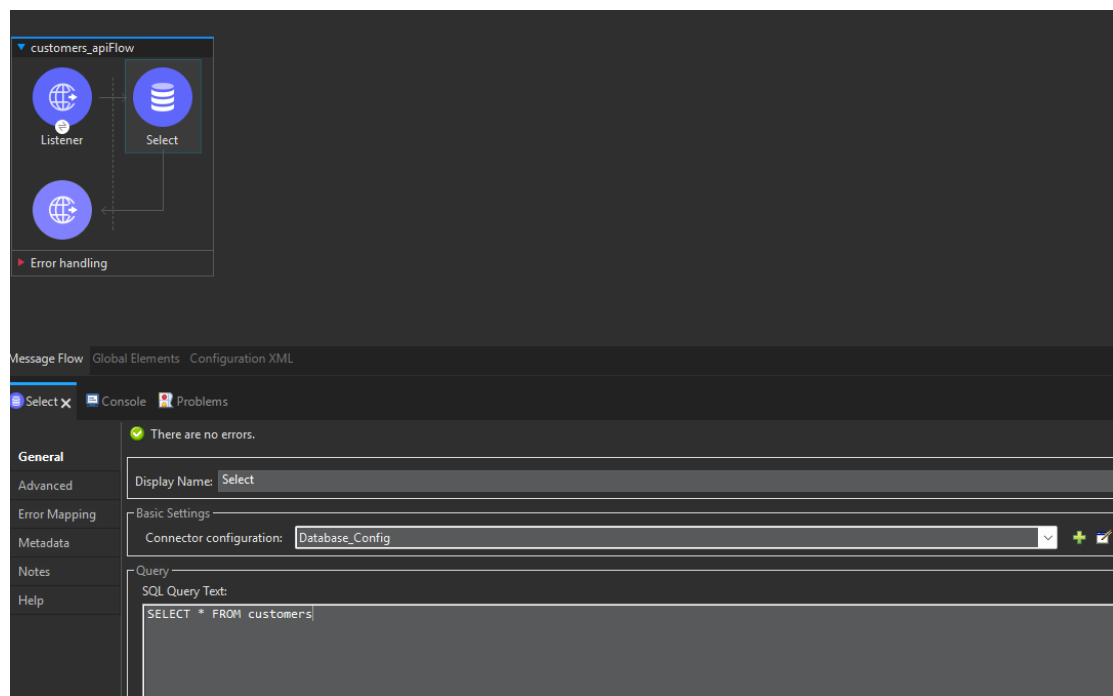


Ilustración 23. SELECT Query

Posteriormente agregaremos un módulo del tipo Transform Message, esto será para que la información que sea recibida se visualice con formato JSON. Arrastraremos dicho módulo a nuestro diagrama en la ventana de Message Flow.

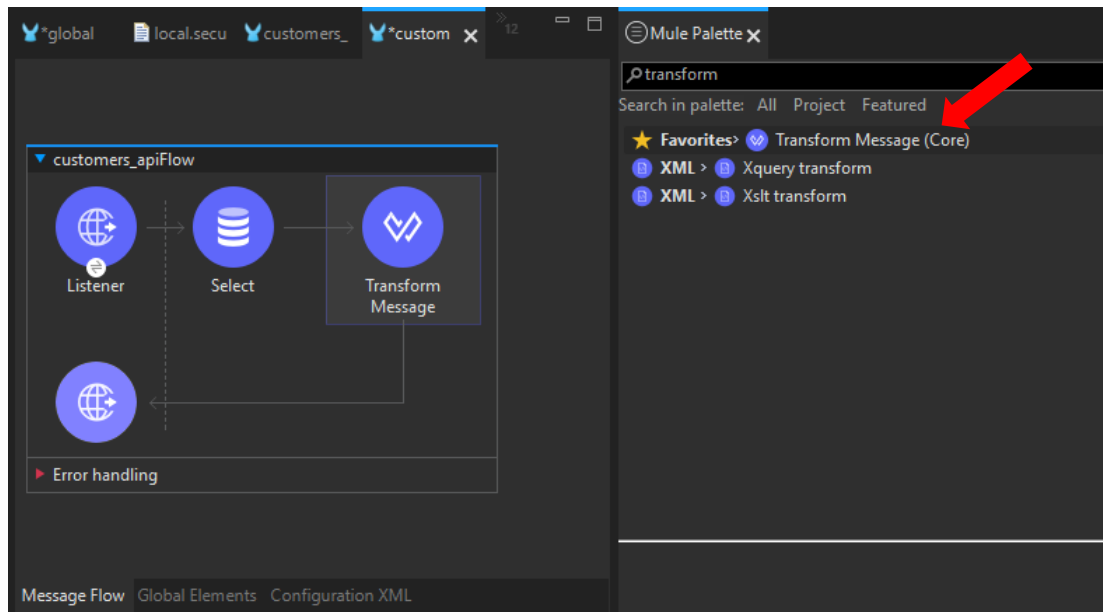


Ilustración 24. Transform Message

Posteriormente, daremos click en el módulo que acabamos de agregar, y utilizando el lenguaje de DataWave agregaremos el siguiente código con los campos que serán recibidos como se muestra a continuación.

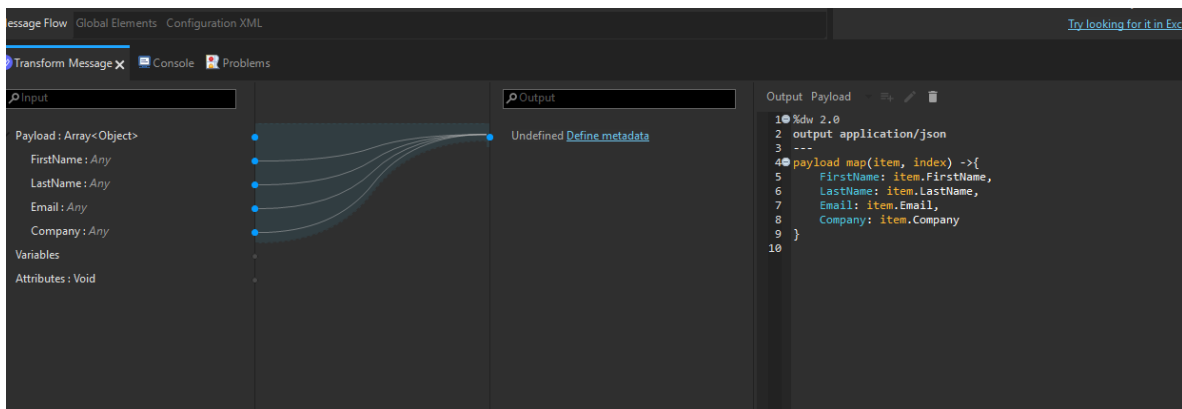


Ilustración 25. Vista de código y gráfica de Transform message

Finalmente haremos una solicitud de prueba en postman para verificar que la API esté funcionando correctamente.

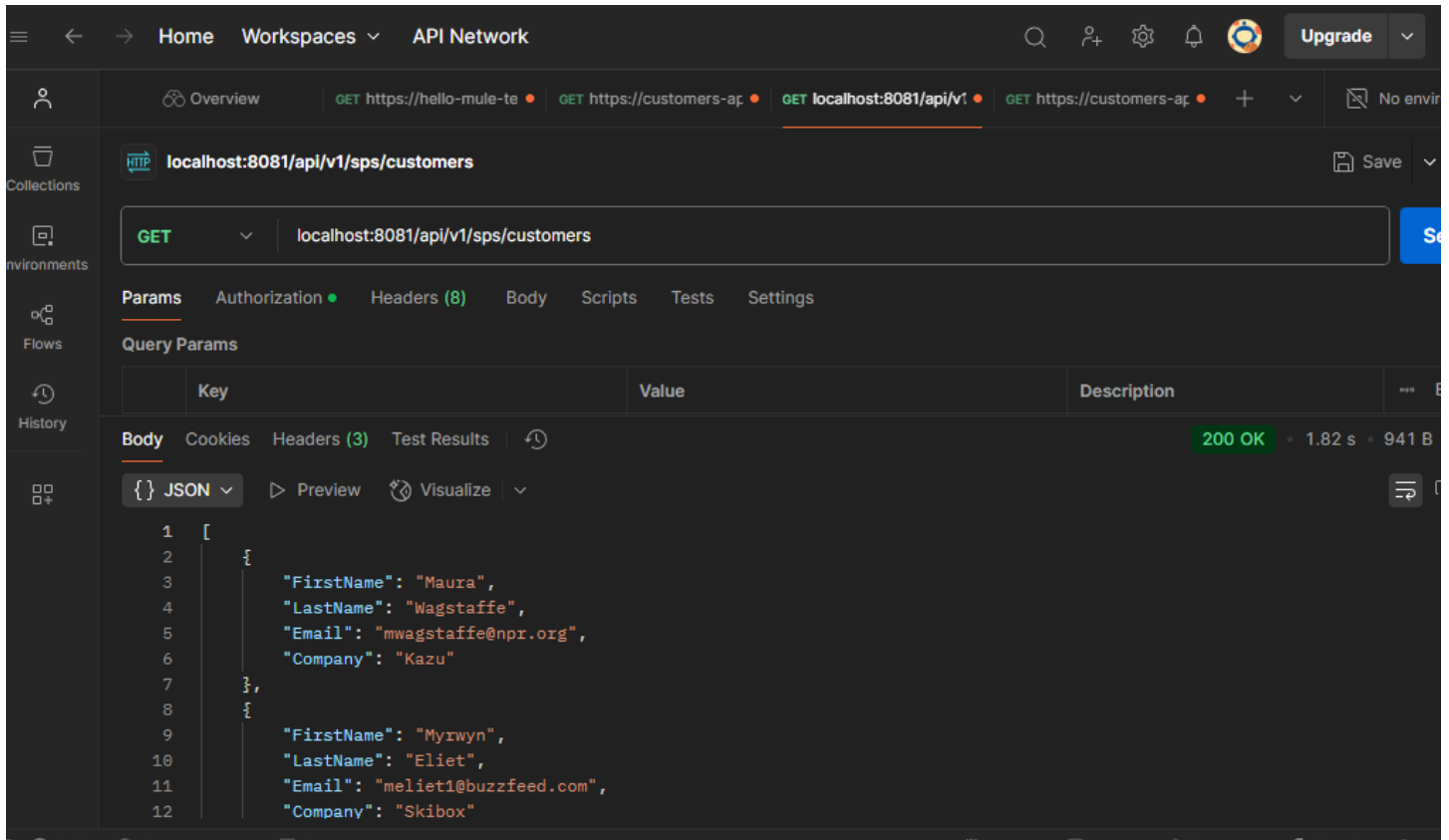


Ilustración 26. GET Response

1.8 Agregando seguridad a los parámetros de la base de datos

Para seguir las buenas prácticas, se agregarán medidas de seguridad y encriptación de parámetros sensibles. Para esto necesitamos agregar el módulo de Secure Properties. Primero agregaremos una propiedad global.

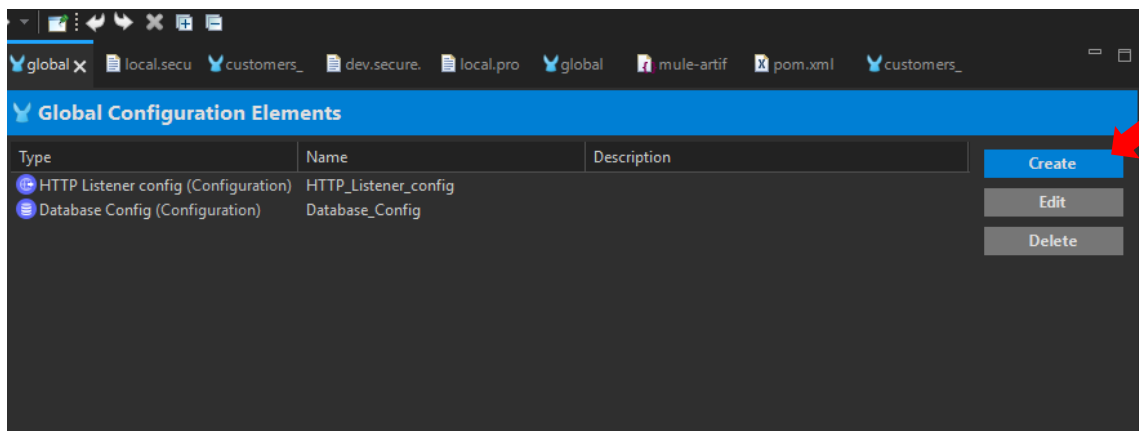


Ilustración 27. Global Elements

La configuración que agregaremos será del tipo Global Property.

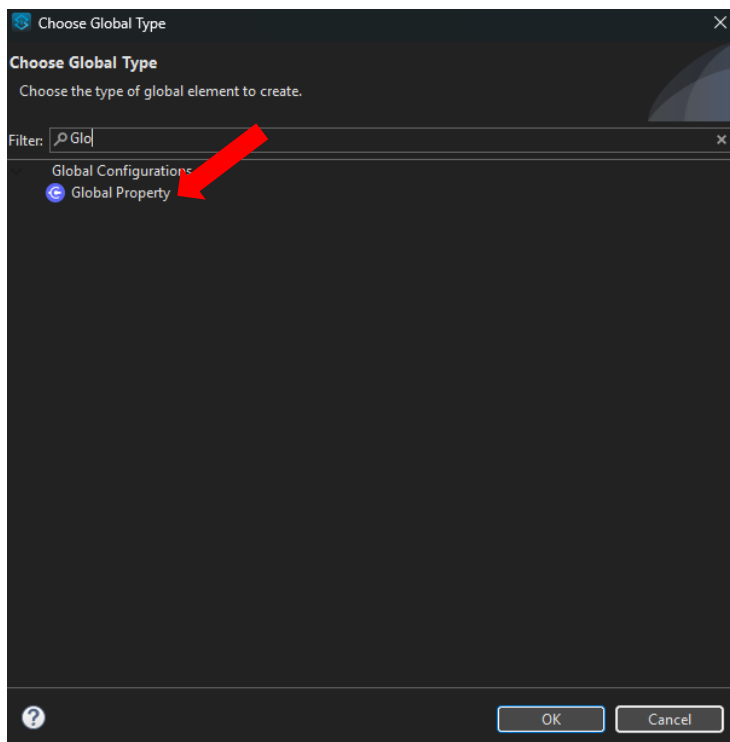


Ilustración 28. Global Properties

La propiedad tendrá como nombre env y como valor local.

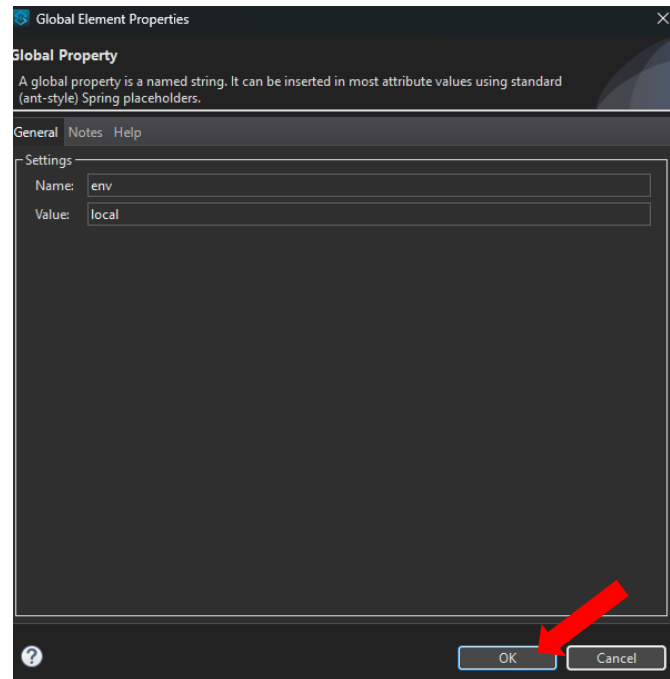


Ilustración 29. Global Property

A su vez, agregaremos una propiedad de configuración del tipo `${env}.properties`, esto permitirá a la API obtener las propiedades sin importar si es ambiente local o dev.

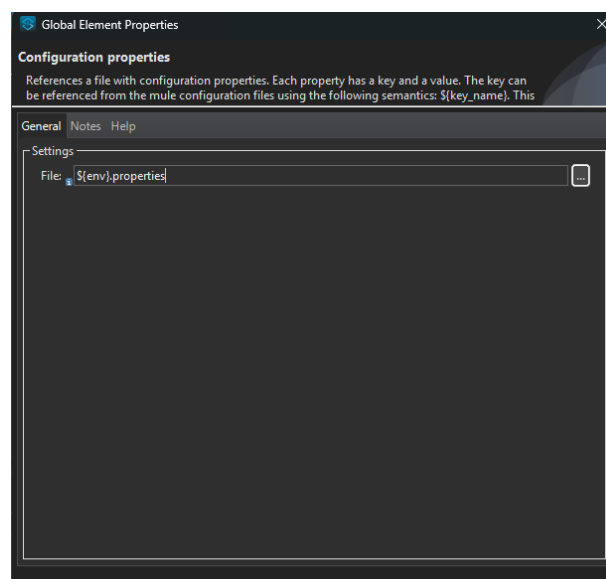


Ilustración 30. Configuration properties

Posteriormente utilizando Mule Palette agregaremos Secure Properties.

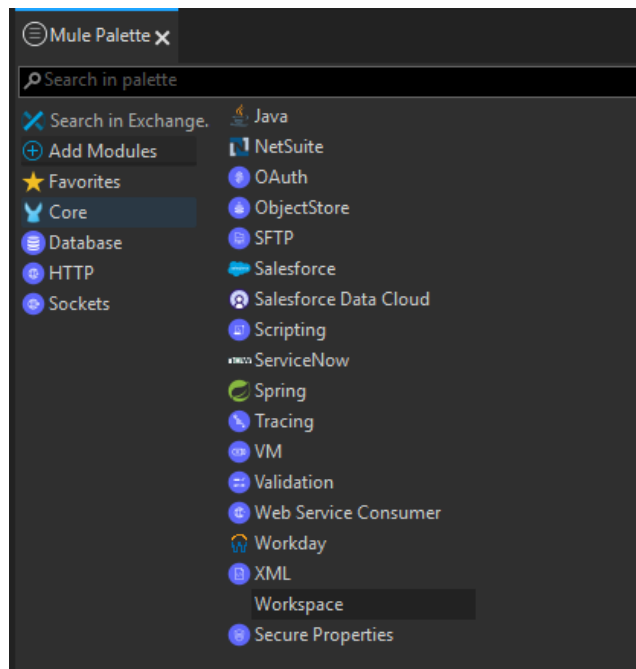


Ilustración 31. Secure Properties

Dentro de nuestro archivo global.xml, agregaremos una propiedad del tipo Secure Properties Config

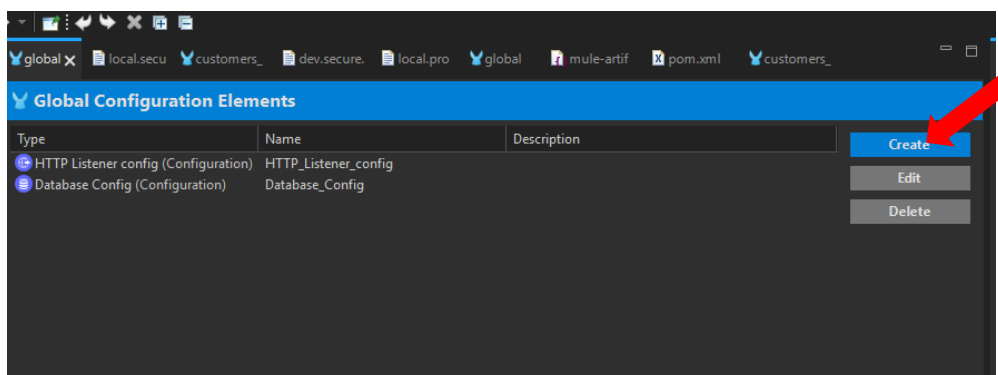


Ilustración 32. Global Configuration Elements

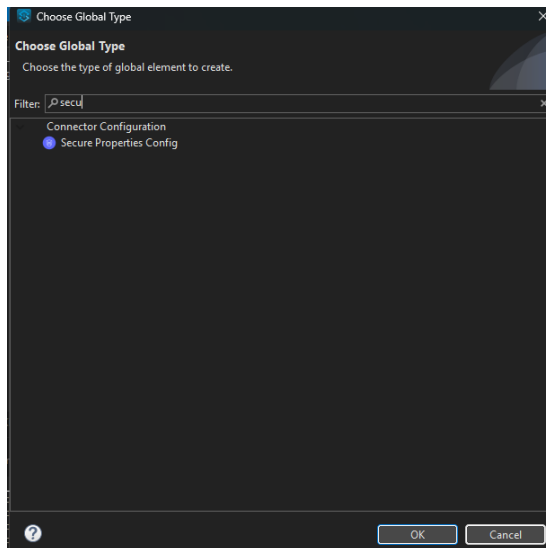


Ilustración 33. Secure Properties Config

Dentro de las secure properties agregaremos los valores que se muestran en la siguiente imagen y daremos click en Ok

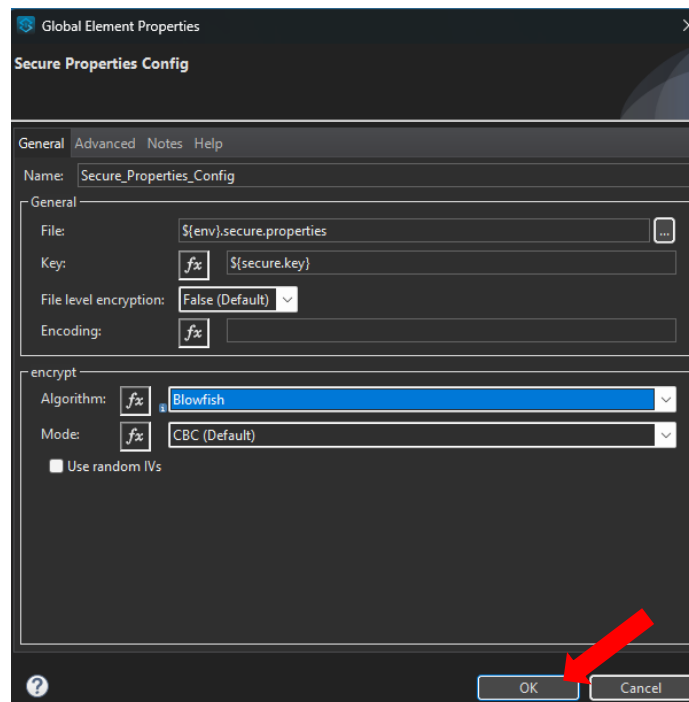


Ilustración 34. Valores del Secure Properties Config

Posteriormente, agregaremos configuraciones de ejecución, para esto daremos click derecho sobre nuestro proyecto y después **Run As -> Run Configurations**.

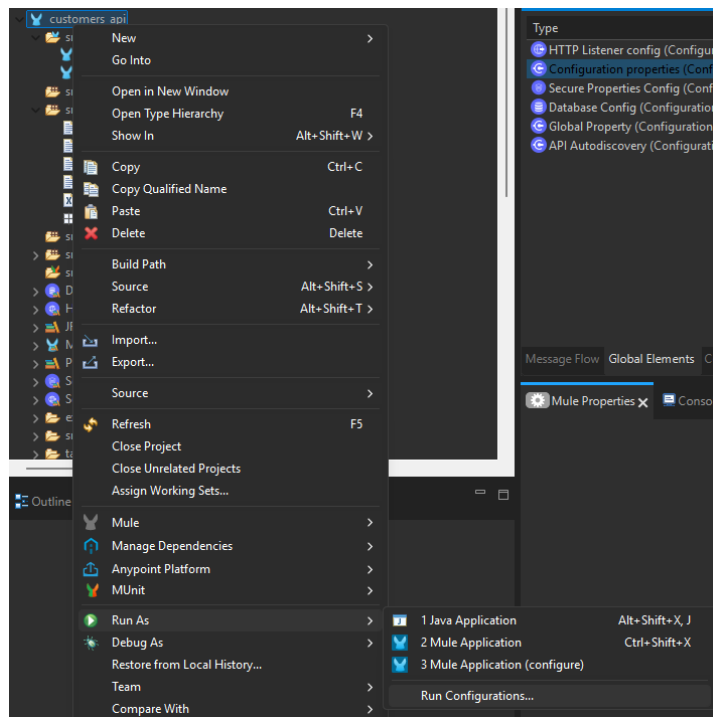


Ilustración 36. Run Configurations

Damos click en Add y agregamos una nueva variable con un nombre de secure.key y con valor de MyMuleSoftKey.

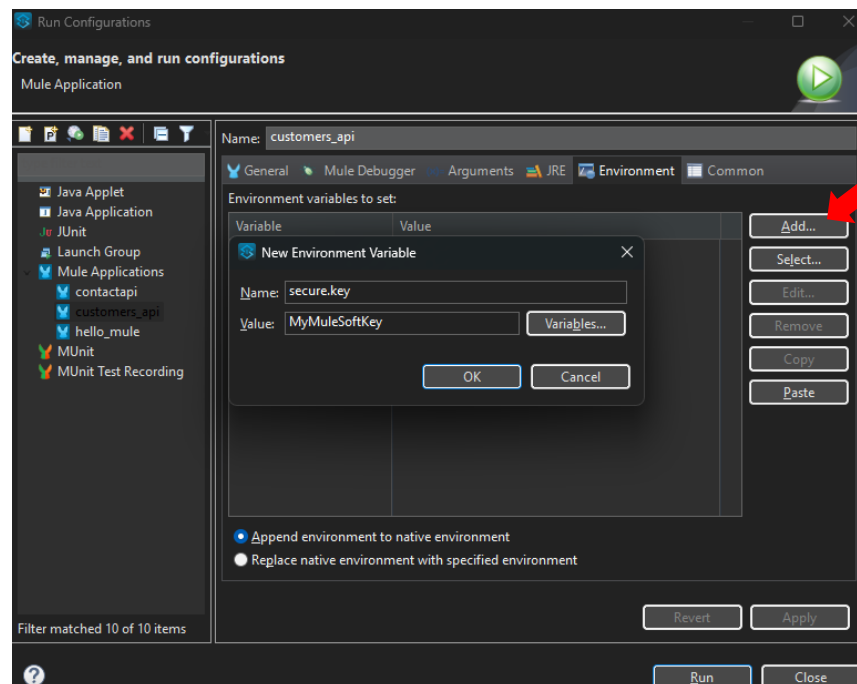


Ilustración 35. New Enviroment Variable

Con la ayuda de la documentación oficial de Mulesoft, vamos a descargar el archivo de secure properties tool del siguiente link:

[Secure Configuration Properties | MuleSoft Documentation](https://docs.mulesoft.com/mule-runtime/latest/secure-configuration-properties)



Ilustración 37. Secure Properties

Al darle click se descargará dicho archivo a nuestro equipo.

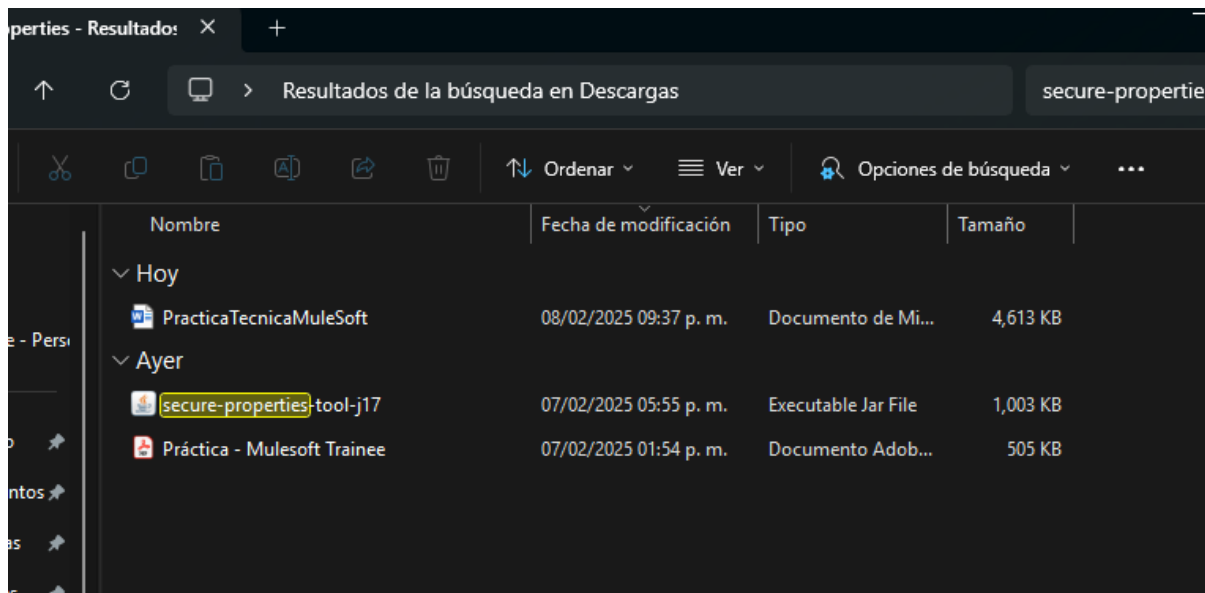
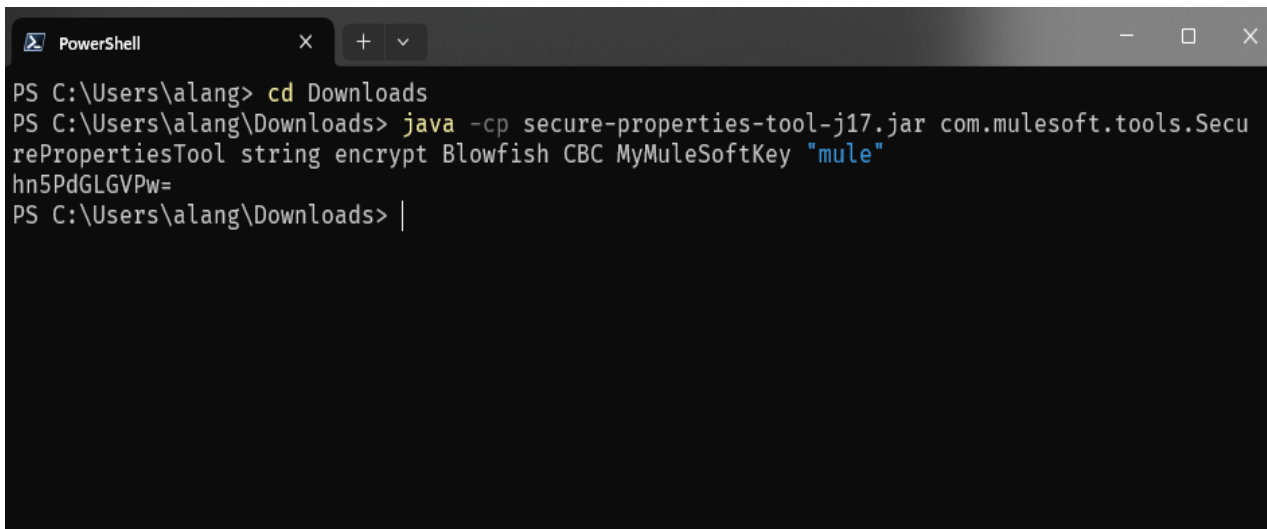


Ilustración 38. Secure-properties-tool-j17.jar

Después, abriendo una terminal en el directorio donde se encuentra el archivo *secure-properties-tool-j17.jar*, ejecutamos el siguiente comando:

```
java -cp secure-properties-tool-j17.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish CBC MyMuleSoftKey "mule"
```

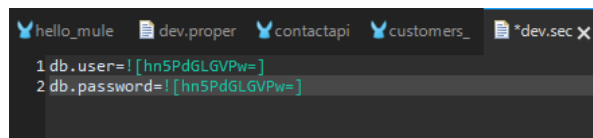
MyMuleSoftKey fue la *secure.key* que configuramos anteriormente, y “mule” es la palabra que se desea encriptar, la cual es el nombre y la contraseña de la base de datos.



```
PowerShell
PS C:\Users\alang> cd Downloads
PS C:\Users\alang\Downloads> java -cp secure-properties-tool-j17.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish CBC MyMuleSoftKey "mule"
hn5PdGLGVPw=
PS C:\Users\alang\Downloads> |
```

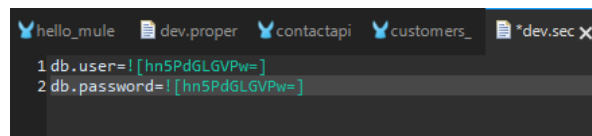
Ilustración 39. Comando de encriptación

Como resultado tenemos una cadena de caracteres que colocaremos en los archivos *local.secure.properties* y *dev.secure.properties*



```
hello_mule dev.proper contactapi customers_ *dev.sec x
1 db.user=! [hn5PdGLGVPw=]
2 db.password=! [hn5PdGLGVPw=]
```

Ilustración 41. *dev.secure.properties*



```
hello_mule dev.proper contactapi customers_ *dev.sec x
1 db.user=! [hn5PdGLGVPw=]
2 db.password=! [hn5PdGLGVPw=]
```

Ilustración 40. *local.secure.properties*

Posteriormente, en el archivo global.xml, sustituiremos los campos de db.user y db.password por \${secure::db.user} y \${secure::db.password} como se muestra en la siguiente imagen:

```
<db:config name="Database_Config" doc:name="Database Config" doc:id="8ff643ae-fdbe-4b91-a510-ac432feac467" >
  <db:mysql-connection host="mudb.learn.mulesoft.com" port="3306" user="${secure::db.user}" password="${secure::db.password}" database="training" />
</db:config>
```

Ilustración 42. Database config.xml

Esto le indica a MuleSoft que tome los parámetros encriptados de los archivos con terminación secure.properties.

1.9 Agregando un Logger

Finalmente se agregará un módulo de tipo Logger, esto con la intención de poder saber si los parámetros encriptados anteriormente son reconocidos en el ambiente de dev después de ser desplegados a CloudHub.

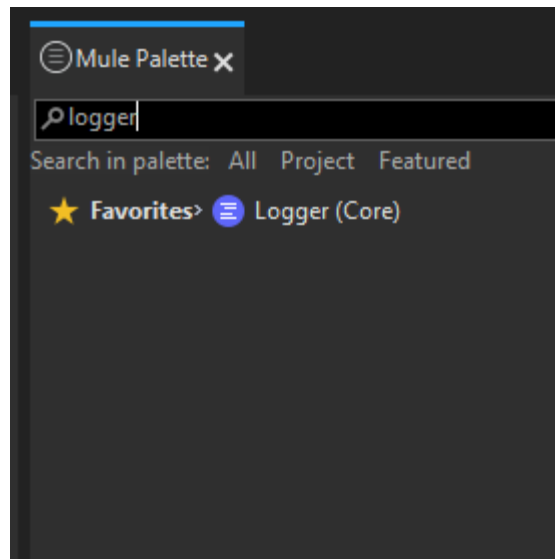


Ilustración 43. Logger

Para esto se usará nuevamente Mule Palette, dónde se realizará la búsqueda de Logger para posteriormente arrastrarlo hacia nuestro diagrama entre los componentes de HTTP Listener y Database Select. Como se muestra en la siguiente imagen

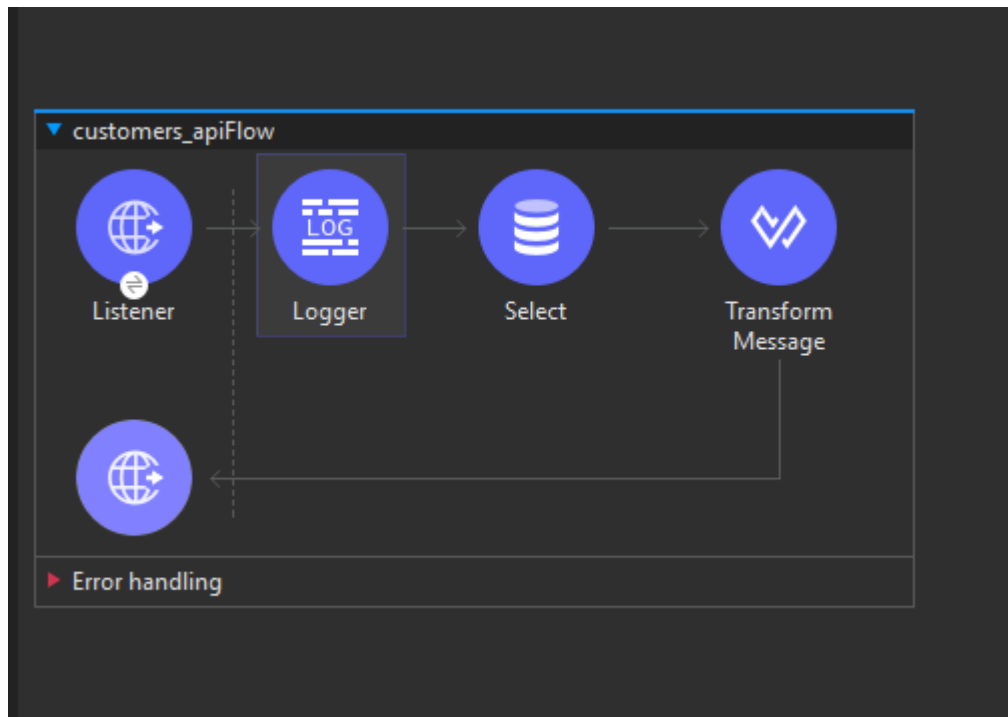


Ilustración 44. `customers_apiFlow`

Posteriormente, haciendo uso del lenguaje DataWave, designaremos el mensaje que queremos dando click en el botón de *Message*.

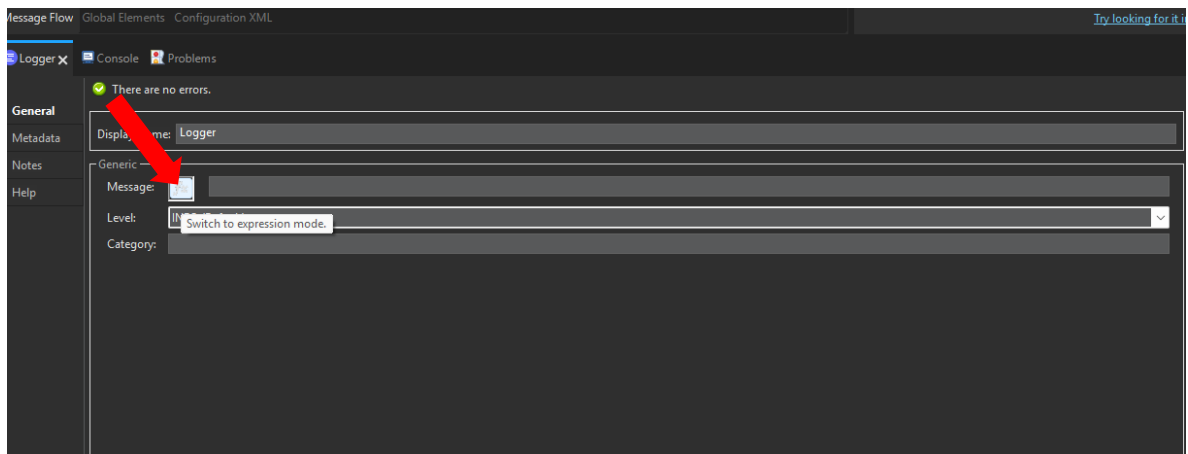


Ilustración 45. *Logger expression mode*

Esto abrirá una sección dónde se podrá escribir el siguiente código que permitirá saber si los parámetros encriptados son reconocidos:

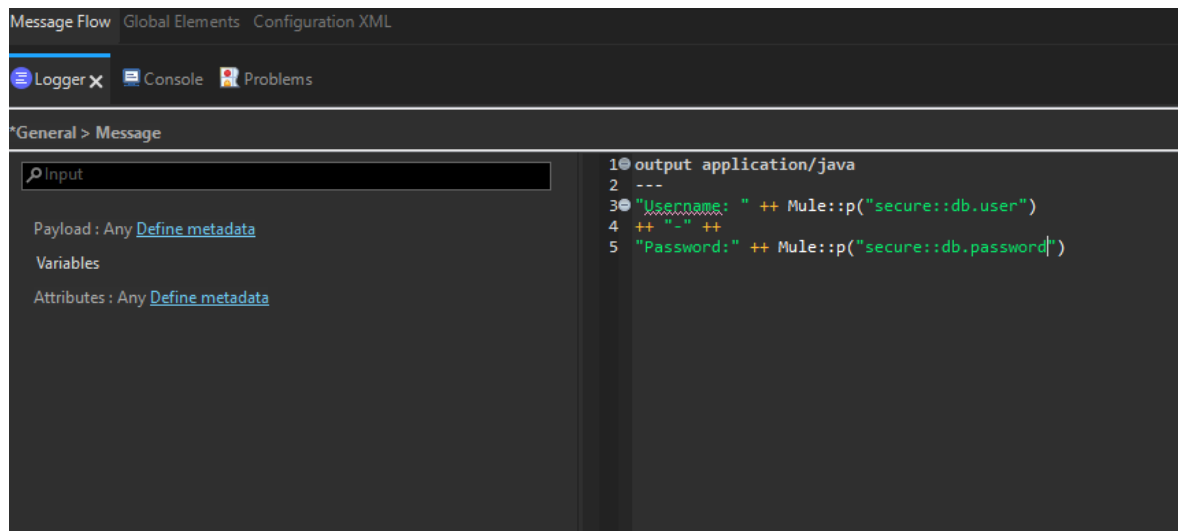


Ilustración 46. Código datawave del logger

2.1- Despliegue de la aplicación

Para poder desplegar la aplicación en CloudHub, daremos click derecho en el proyecto “customers_api”, y después **Anypoint Platform -> Deploy to CloudHub**

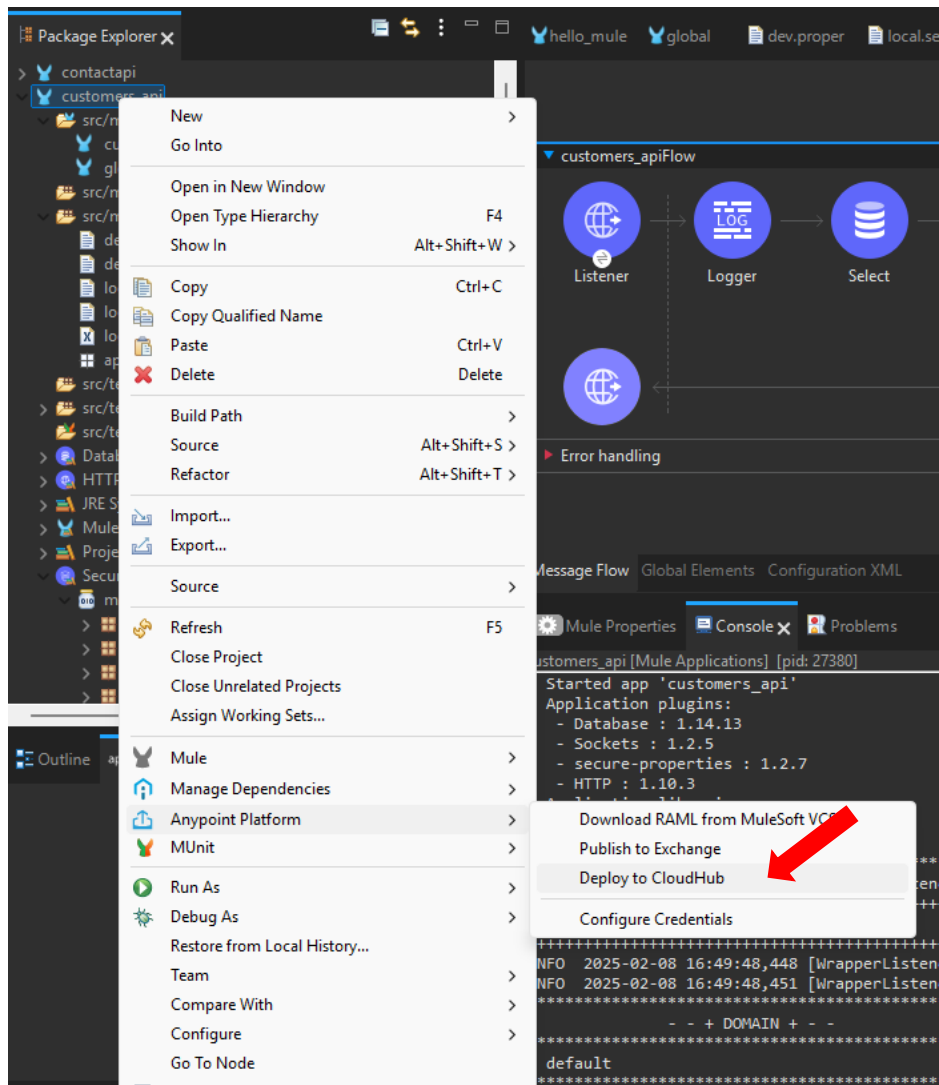


Ilustración 47. Deploy to CloudHub

Esto abrirá una ventana como la que se muestra a continuación, en la cual, en la ventana de *Properties*, se agregarán los parámetros de *secure.key* y *env*, con sus respectivos valores.

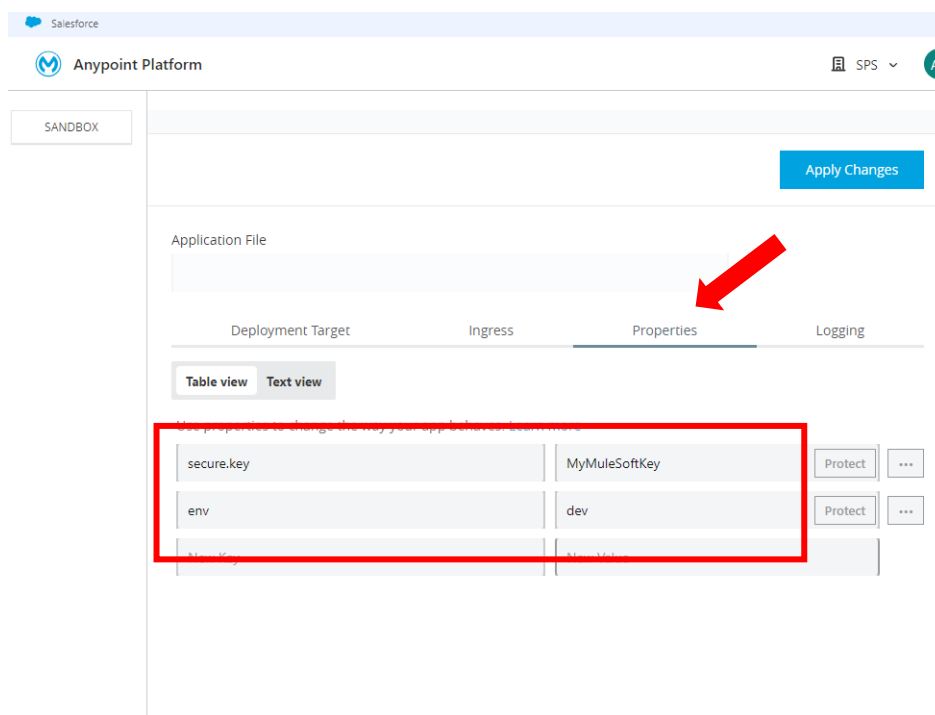


Ilustración 48. Propiedades de despliegue

Finalmente, se dará click en el botón *Apply Changes*

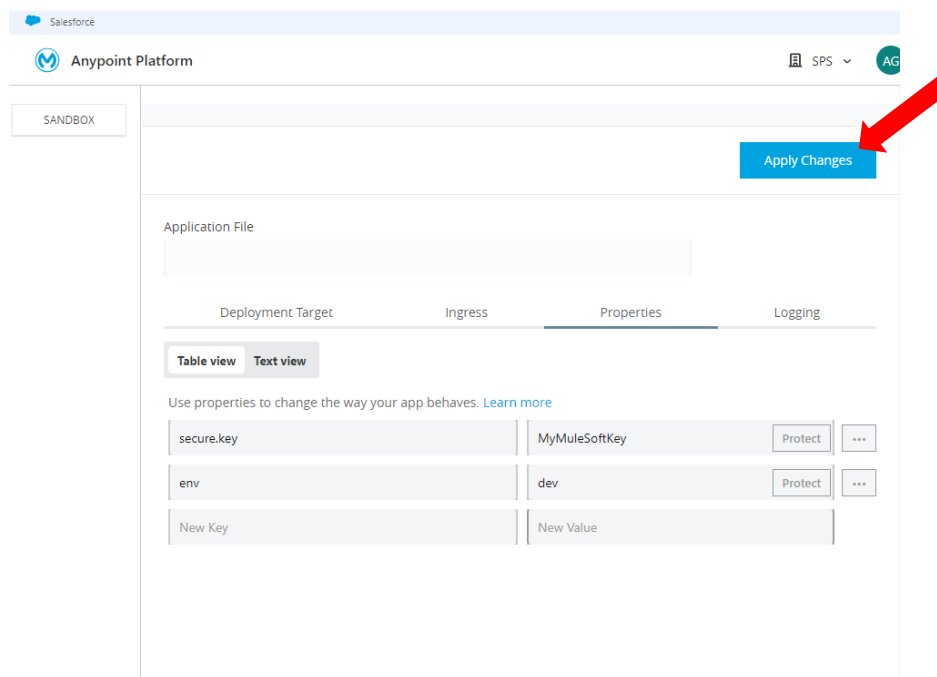


Ilustración 49. Apply Changes

Posteriormente se mostrará una ventana como en la siguiente imagen, lo cuál indica que la aplicación ya está siendo desplegada en CloudHub. El siguiente paso es dar click en *Open in Browser*.

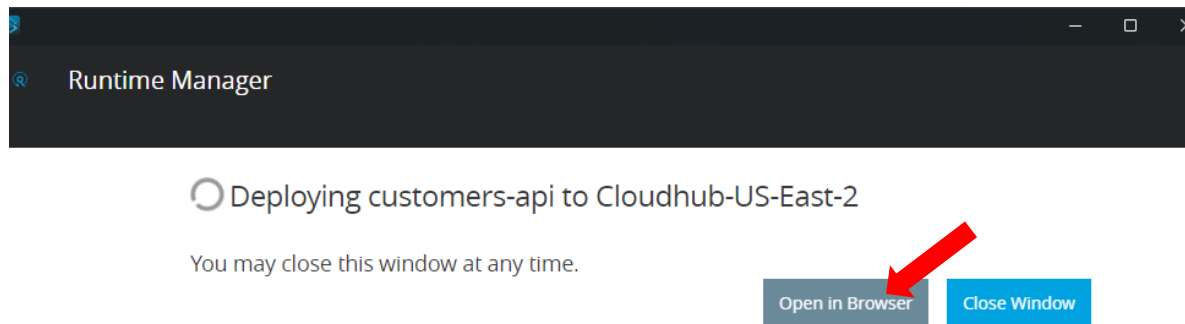


Ilustración 50. Abrir en Navegador

Esto abrirá una ventana de Runtime Manager dónde podremos confirmar el estado de despliegue de la aplicación, así como ver la url con la cuál podremos utilizar nuestra API.

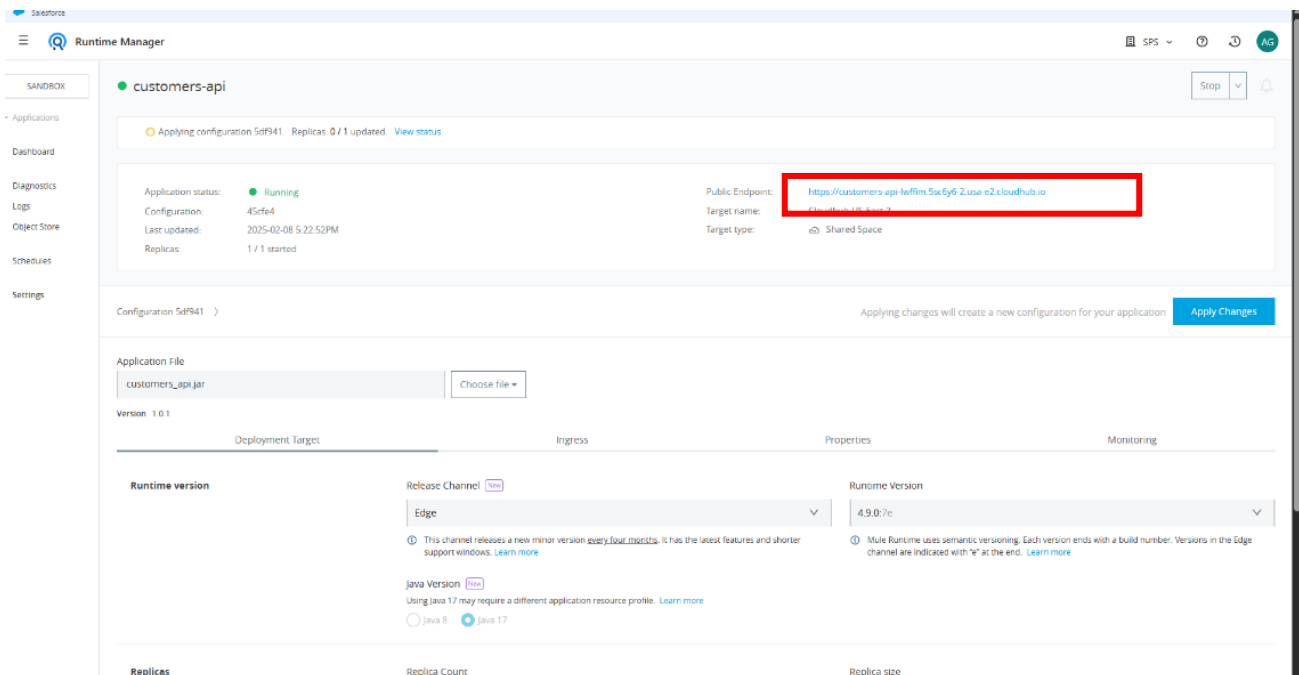


Ilustración 51. URL de CloudHub

Por último utilizando la herramienta de Postman, haremos una prueba utilizando dicha URL para verificar que todo funcione correctamente.

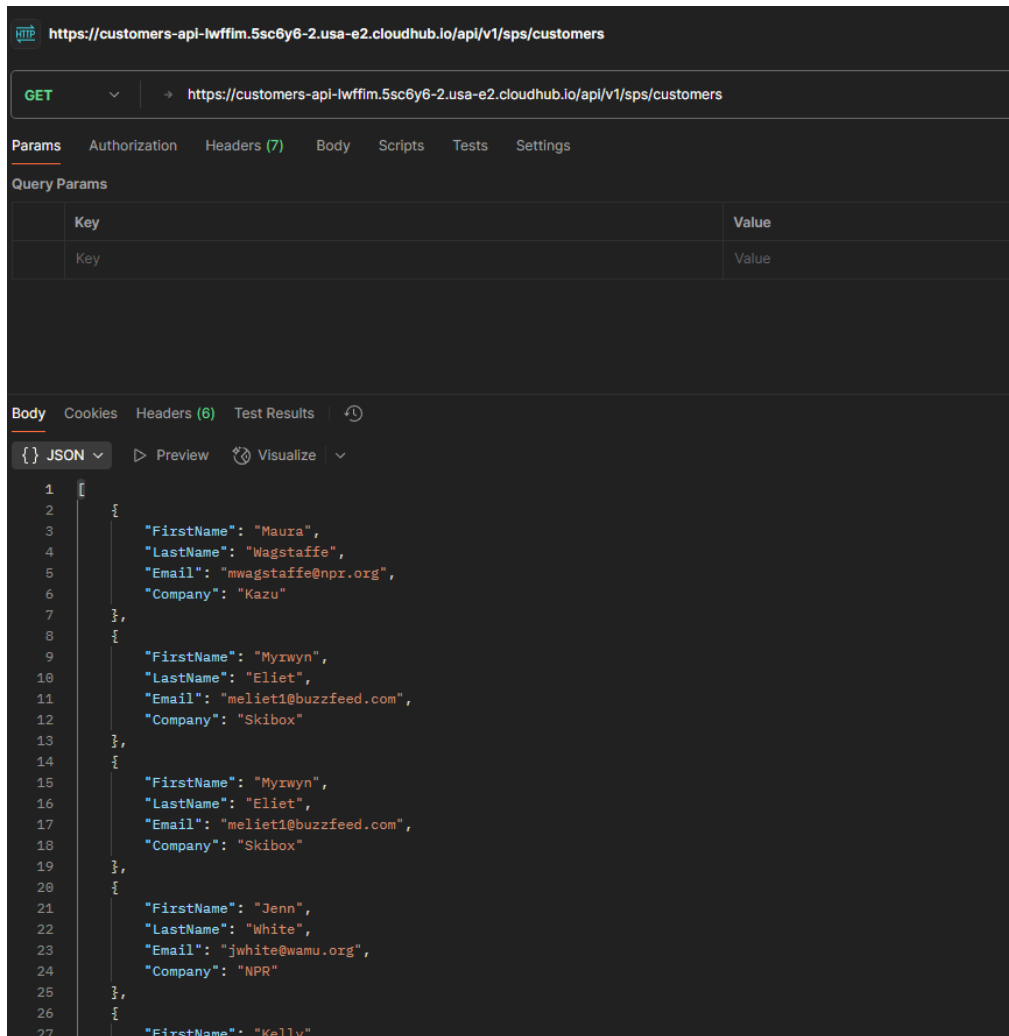


Ilustración 52. Petición de prueba

3.- Especificación de la API

Para poder realizar la especificación de nuestra API, se debe de iniciar sesión en la plataforma de AnyPoint [Anypoint Platform](#). Posteriormente daremos click en *Start Designing* en la sección de Design Center.

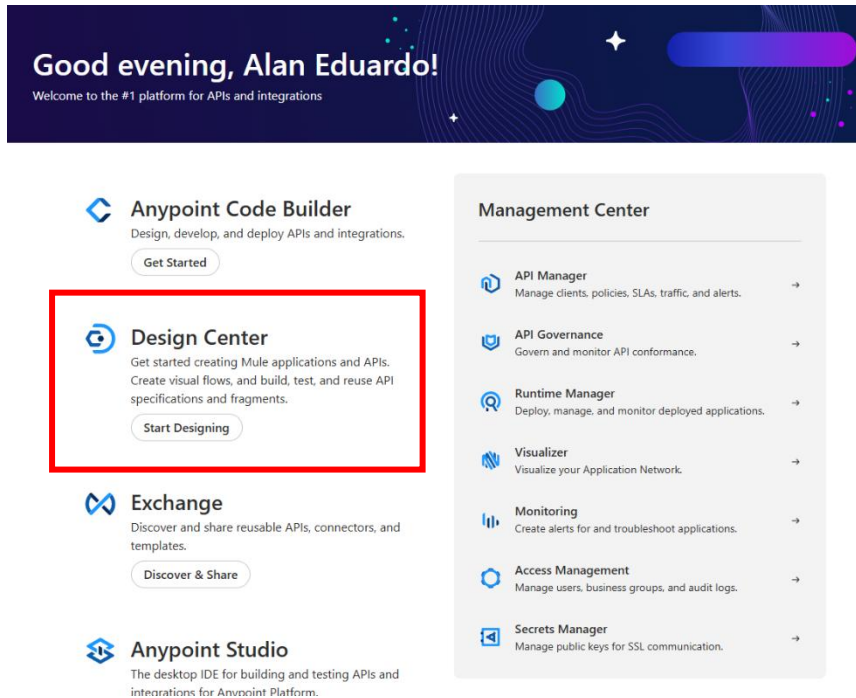


Ilustración 53. Design Center

Después daremos click en el botón de *Create* y después en *New API Specification*.

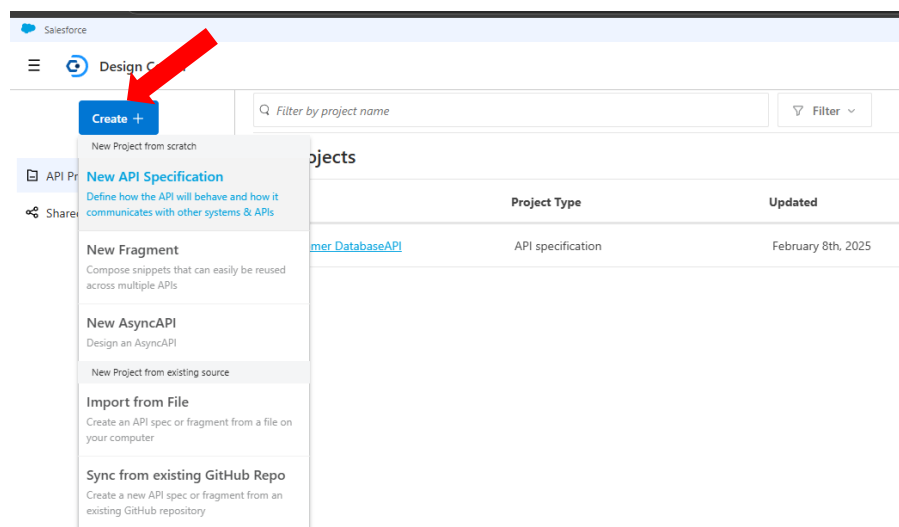
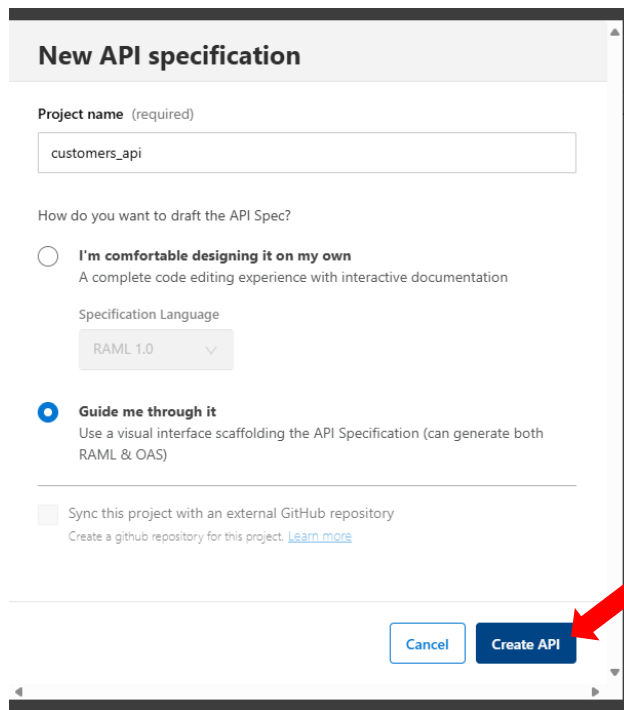


Ilustración 54. Create new api Specification

Se abrirá una ventana en dónde escribiremos el Project_name y seleccionaremos la opción de *Guide me through it* para finalmente dar click en Create API.



New API specification

Project name (required)
customers_api

How do you want to draft the API Spec?

☐ I'm comfortable designing it on my own
A complete code editing experience with interactive documentation

Specification Language
RAML 1.0

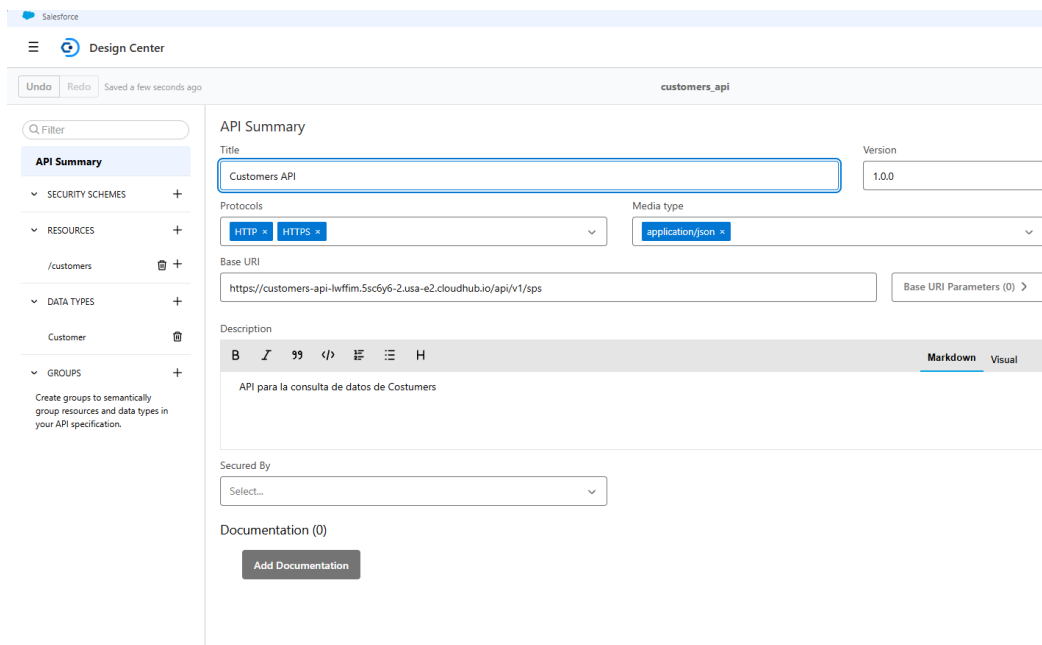
☒ **Guide me through it**
Use a visual interface scaffolding the API Specification (can generate both RAML & OAS)

☐ Sync this project with an external GitHub repository
Create a github repository for this project. [Learn more](#)

Cancel Create API

Ilustración 55. New API Specification

Esto abrirá una ventana en Design Center dónde podremos agregar información a nuestra API. En el apartado de versión colocaremos 1.0.0. Mientras que en Protocols agregaremos las tags de HTTP y HTTPS. En la sección de Media type seleccionaremos application/json. Y finalmente en el apartado de Base URL colocaremos la url que usamos anteriormente en postman para probar nuestra API.



Design Center

customers_api

Undo Redo Saved a few seconds ago

API Summary

SECURITY SCHEMES +

RESOURCES +

/customers +

DATA TYPES +

Customer +

GROUPS +

Create groups to semantically group resources and data types in your API specification.

API Summary

Title: Customers API Version: 1.0.0

Protocols: HTTP HTTPS Media type: application/json

Base URI: https://customers-api-lwffim.5sc6y6-2usa-e2.cloudhub.io/api/v1/sps Base URI Parameters (0)

Description

API para la consulta de datos de Costumers

Secured By: Select...

Documentation (0)

Add Documentation

Ilustración 56. API Summary

3.1 Creación de Data Types

El siguiente paso es agregar un Datatype de tipo Customer que contendrá la estructura con los parámetros que van a ser recibidos. Para esto damos click en la sección de Data Types. Y le asignaremos el Nombre de Customer y el Type de Object. Después, agregaremos los parámetros basándonos en la respuesta de Postman dando click en el botón de Add Property.

The screenshot shows the 'Design Center' interface for 'customers_api'. On the left, a sidebar lists 'API Summary', 'SECURITY SCHEMES', 'RESOURCES', 'DATA TYPES', and 'GROUPS'. The 'DATA TYPES' section is expanded, showing a 'Customer' type. The main area is titled 'Data types' and contains a form with 'Name' (Customer) and 'Type' (Object). Below this is a 'Description' field with a rich text editor. A red arrow points to the 'Add Property' button. At the bottom, there is an 'Example' section with the text 'No inherited example...'.

Ilustración 57. Data Types

Las propiedades que serán agregadas serán *FirstName*, *LastName*, *Email* y *Company*, todas de un Type *String*.

This screenshot shows the 'Add Property' dialog box within the 'Data types' section. It lists four properties to be added to the 'Customer' type: 'FirstName', 'LastName', 'Email', and 'Company'. Each property has a 'Property Name' field, a 'Required' checkbox, and a 'Type' dropdown set to 'String'. The 'Company' property is highlighted with a blue border, and its 'Required' checkbox is checked. The 'Add Property' button is at the bottom of the list.

Ilustración 58. Data Type Properties

Finalmente agregaremos información de ejemplo de cómo sería la respuesta del Data Type que acabamos de crear. Daremos click en el botón de Edit y agregaremos un registro desde la respuesta que obtuvimos previamente de Postman

The screenshot displays the Swagger Editor interface for a project named 'customers.api'. On the left, the 'API Summary' sidebar shows a tree view with 'SECURITY SCHEMES', 'RESOURCES', 'DATA TYPES', and 'GROUPS'. The 'DATA TYPES' section is expanded, showing a 'Customer' data type. The main editor area is titled 'Data Type de Customer' and contains a table of properties:

Property Name	Type	Required	Union
FirstName	String	<input type="checkbox"/>	<input type="checkbox"/>
LastName	String	<input type="checkbox"/>	<input type="checkbox"/>
Email	String	<input type="checkbox"/>	<input type="checkbox"/>
Company	String	<input type="checkbox"/>	<input type="checkbox"/>

Below the table is an 'Add Property' button. Underneath, the 'Example' section is highlighted with a red box and contains the following JSON:

```
{  "FirstName": "Jenn",  "LastName": "White",  "Email": "jwhite@amu.org",  "Company": "NPR"}
```

To the right of the example is an 'Inherited' button and an 'Edit' button, which is pointed to by a red arrow. On the far right, the 'Edit RAML' tab is active, showing the RAML code for the data type.

Ilustración 59. Información de ejemplo

3.3 Creación de Endpoints

En la sección de RESOURCES, agregaremos el endpoint solicitado que nos permita obtener la información de los customers. Dicha ruta será **/api/v1/sps/customers**. y será del tipo GET. También se podrá agregar el nombre de la ruta así como una breve descripción de la misma.

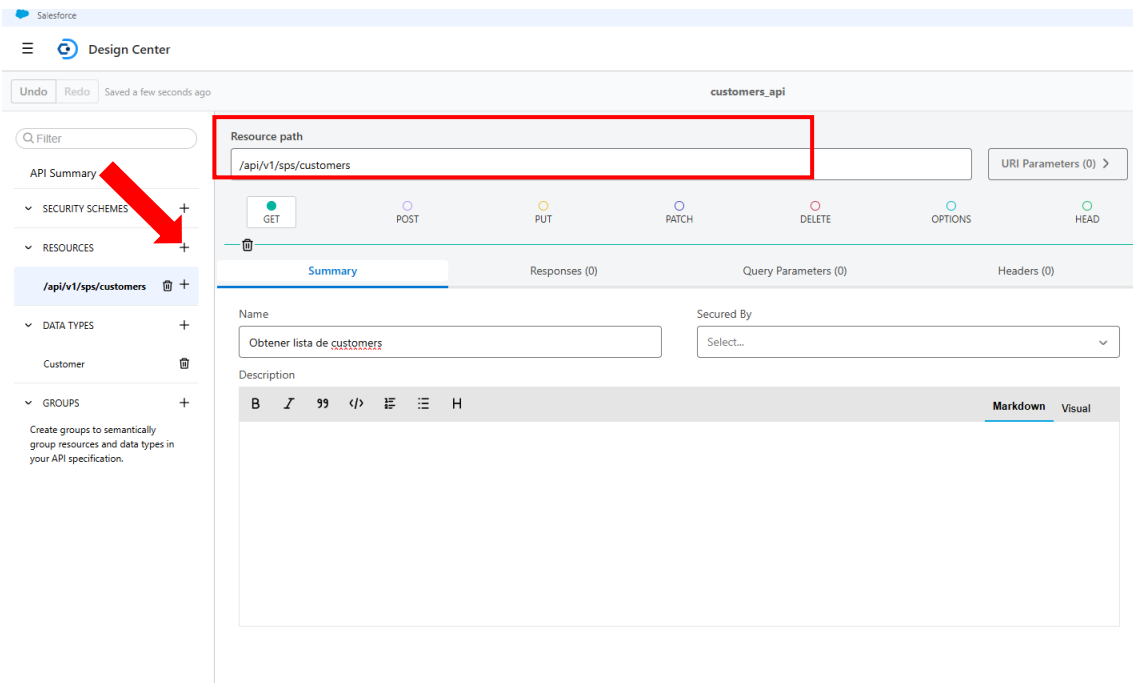


Ilustración 60. Resource path

Posteriormente, en la sección de Responses, agregaremos un tipo de respuesta.

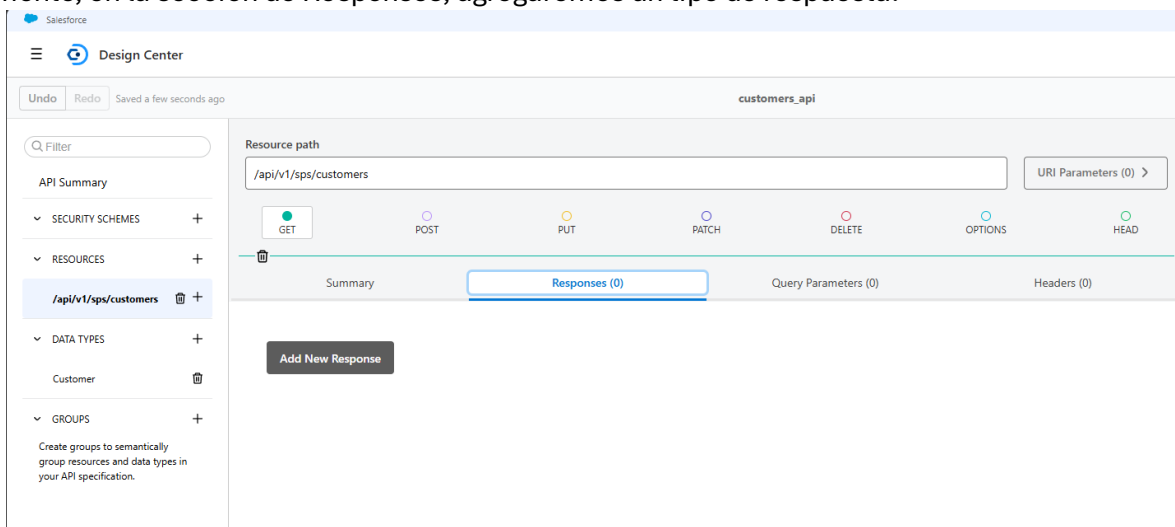


Ilustración 61. New response

Asignaremos el bodie como application/json y en Type Customer.

The screenshot shows an API client interface with the following components:

- Resource path:** /customers
- URI Parameters (0):** >
- Methods:** GET (selected), POST, PUT, PATCH, DELETE, OPTIONS, HEAD
- Summary:** Responses (1), Query Parameters (0), Headers (0)
- Status:** 200 - OK (indicated by a red arrow)
- Description:** A text area with a rich text editor toolbar (B, I, U, </>, etc.) and tabs for Markdown and Visual.
- Bodies (1):**
 - Media Type:** Default (application/json)
 - Type:** A dropdown menu is open, showing options: Datetime-Only, Date-Only, File, Array, Nil, and Customer (highlighted in blue).
 - Buttons:** Add Body

Ilustración 62.Add body

3.3 Probando la Especificación

Para comprobar que todo funcione correctamente, del lado derecho daremos click en el botón de Try It. Esto abrirá una ventana dónde podremos probar el endpoint creado previamente, y en dado caso que sea necesario también se podrán agregar Query parameters.

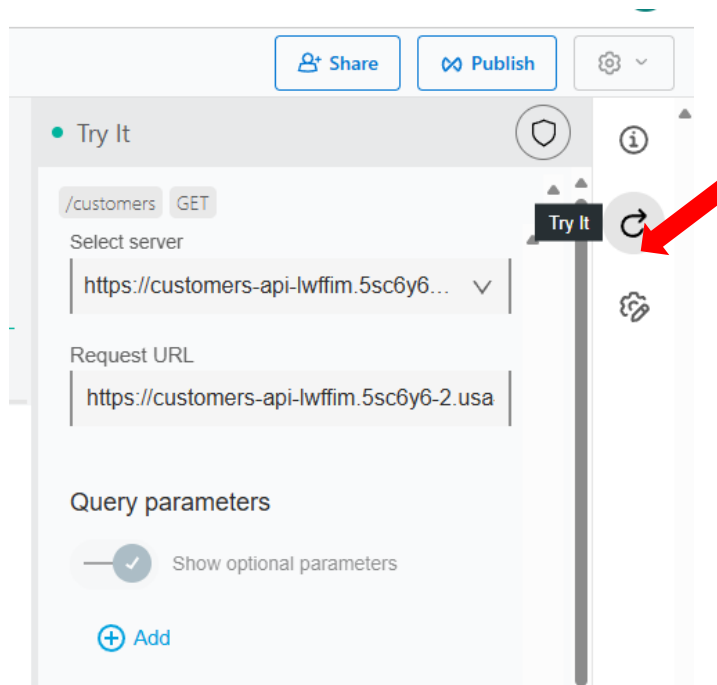


Ilustración 63. Try It

Para realizar una petición de prueba simplemente damos click en el botón de Send, y si todo funciona correctamente deberíamos obtener un código 200 así como la información solicitada.

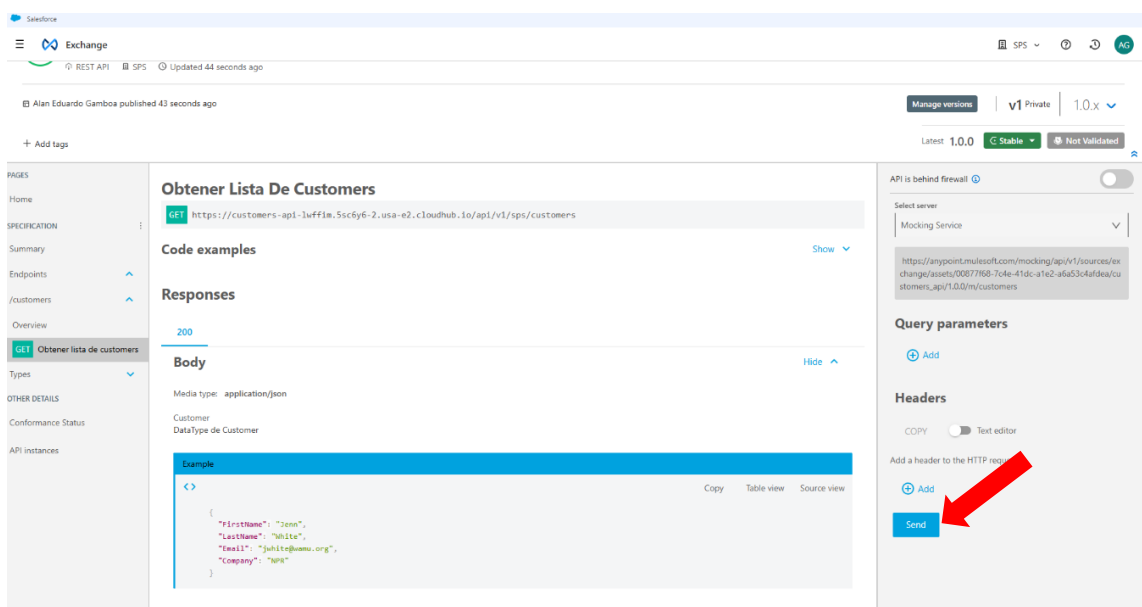


Ilustración 64. Send Request

3.4 Publicando la especificación en Exchange

Para publicar nuestra especificación en Exchange daremos click en el botón de Publish

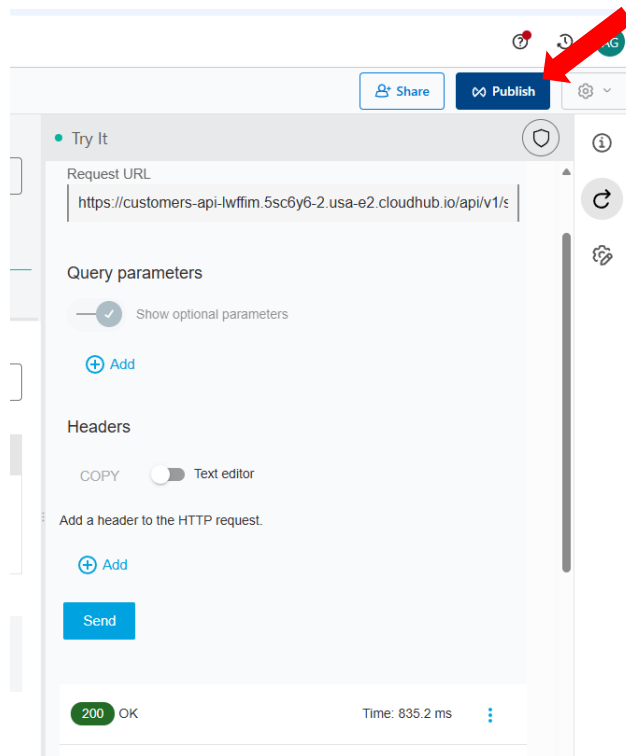


Ilustración 65. Publish

Esto abrirá una ventana como la que se muestra a continuación, dónde se asignará un **Asset Version** de 1.0.0 y una **API Version** de v1, además de asignar un **LifeCycle State** de Stable. Finalmente daremos click en *Publish to Exchange*.

A screenshot of a 'Publishing to Exchange' dialog box. It contains three main sections: 'Asset version' with a text input field containing '1.0.0' and a note '1.0.1 published 0 days ago'; 'API version' with a text input field containing 'v1'; and 'LifeCycle State' with three radio buttons: 'Stable' (selected), 'Development', and 'Advanced options'. To the right of these sections is a 'More help' section with links to 'Changing a project's main/root file' and 'What is an API version?'. At the bottom right, there is a red arrow pointing to a button labeled 'Publish to Exchange'.

Ilustración 66. Publish to Exchange

Cuando la especificación haya sido publicada en Exchange, se podrá observar un cuadro de diálogo como el que se muestra a continuación.

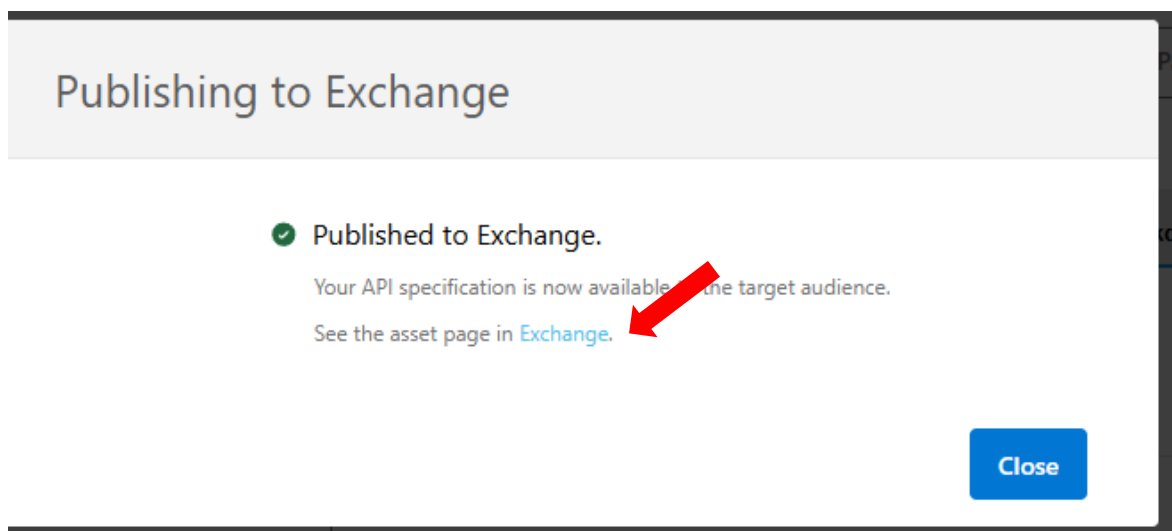


Ilustración 67. Exchange

Opcionalmente, se puede dar click en el texto de Exchange el cuál abrirá la página de detalles de la especificación ya publicada.

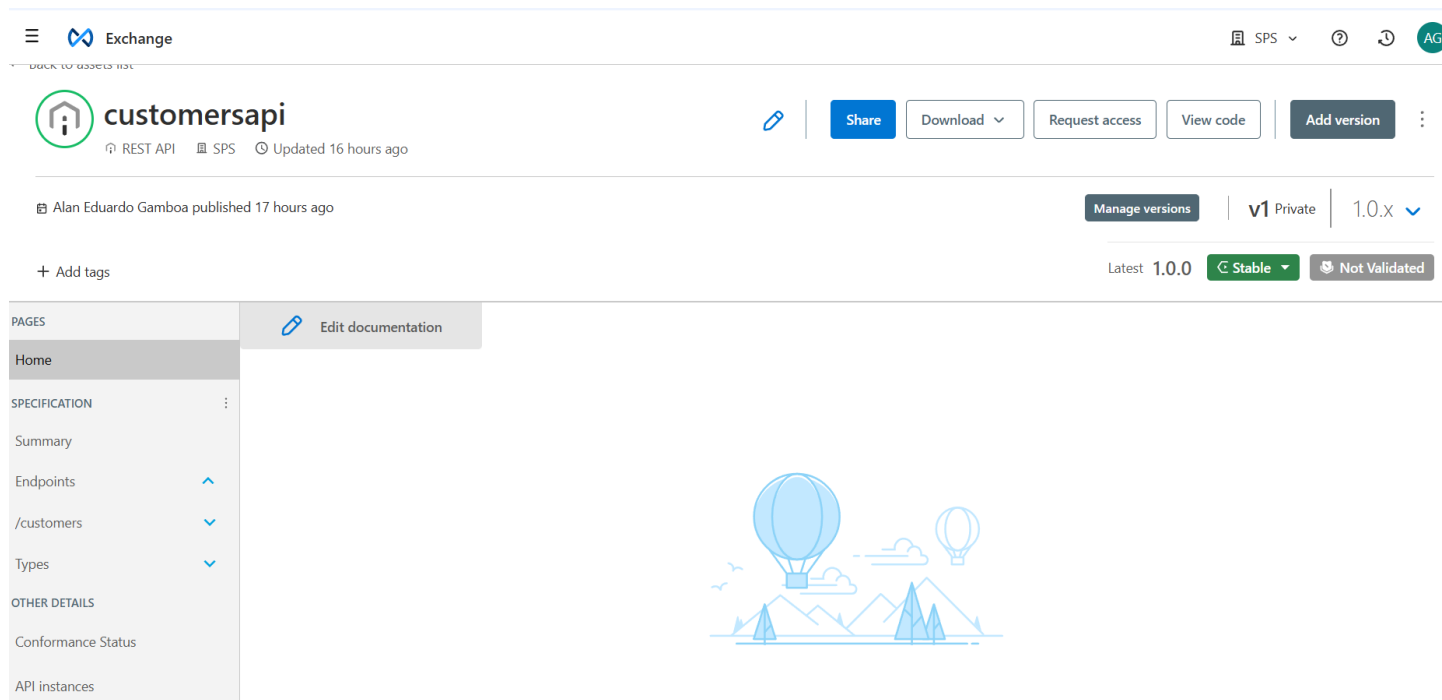


Ilustración 68. Exchange Summary

4.- Creación de una API en API Manager

Para la creación de una API ingresaremos nuevamente a Anypoint Platform y después a la sección de API Manager

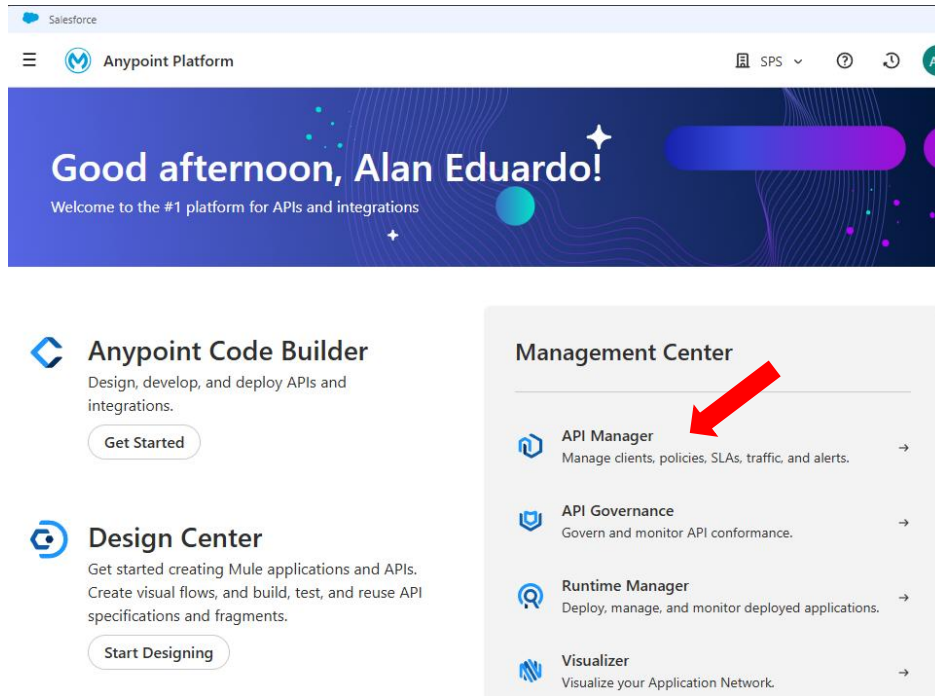


Ilustración 69. API Manager

Después daremos click en al botón de *Add API* y luego en *Add new API*

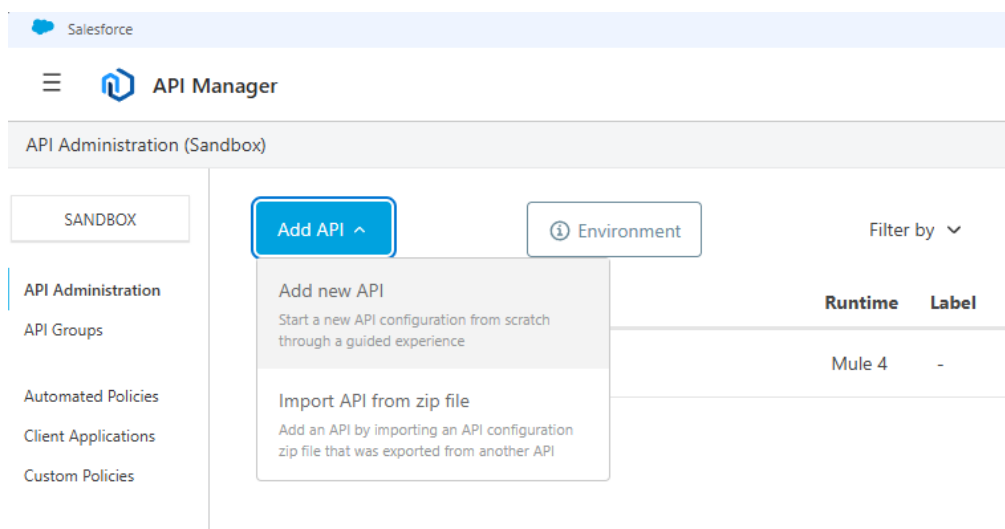


Ilustración 70. Add New API

Seleccionamos la opción de Mule Gateway, y damos click en Next

The screenshot shows the 'Add API' wizard in the Mule API Manager, specifically the 'Runtime' step. The left sidebar contains navigation links: 'Sandbox', 'API Administration', 'API Groups', 'Automated Policies', 'Client Applications', and 'Custom Policies'. The main content area has a progress bar with steps: Runtime, API, Downstream, Upstream, and Review. The 'Runtime' section is active, with the instruction 'Choose the runtime where your API instance will run on.' Below this, there is a 'Select runtime' section with two options: 'Flex Gateway' (unselected) and 'Mule Gateway' (selected). The 'Mule Gateway' option is highlighted with a blue border and a blue radio button. Below the runtime selection, there is a 'Proxy type' section with two options: 'Connect to existing application (basic endpoint)' (selected) and 'Deploy a proxy application' (unselected). Below that, there is a 'Mule version' section with two options: 'Mule 4 (recommended)' (selected) and 'Mule 3 or below' (unselected). At the bottom right, there is a blue 'Next' button, which is pointed to by a red arrow.

Ilustración 71. API properties

Después seleccionamos la opción de *Select API from Exchange* y seleccionamos la api que hemos publicado previamente desde Exchange

The screenshot shows the 'Add API' wizard in the Mule API Manager, specifically the 'API' step. The left sidebar is the same as in the previous screenshot. The main content area has a progress bar with steps: Runtime, API, Downstream, Upstream, and Review. The 'API' step is active, with the instruction 'Select the API you want to manage.' Below this, there are two options: 'Select API from Exchange' (selected) and 'Create new API' (unselected). Below these options, there is a 'Select API' section with a search bar labeled 'Q Search APIs'. Below the search bar, there is a list of APIs. The first API, 'customers_api', is highlighted with a red box. It has a blue radio button and the text 'Published to Exchange: February 8, 2025'. To the right of this entry is a link 'View in Exchange'. Below this entry, there is another API, 'hellomuleapi', which is not highlighted.

Ilustración 72. Customer_API

En la sección de Downstream y Upstream dejaremos las opciones por default.

The screenshot shows the 'Add API' page in the Salesforce API Manager. The breadcrumb trail is 'APIs / Add API'. The left sidebar contains a 'SANDBOX' tab and a list of links: 'API Administration (Sandbox)', 'API Groups', 'Automated Policies', 'Client Applications', and 'Custom Policies'. The main content area has a progress bar with five steps: 'Runtime' (checked), 'API' (checked), 'Downstream' (checked), 'Upstream' (active), and 'Review' (disabled). Below the progress bar, the 'Upstream' section is titled 'Define how the traffic flows to upstream services.' It includes a required field 'Upstream URL (Optional)' with the value 'https/'. At the bottom, there are three buttons: 'Cancel', 'Previous', and 'Next'.

Ilustración 73. Upstream

Finalmente daremos click en Save

The screenshot shows the 'Review' tab of the 'Add API' page. The progress bar now shows 'Upstream' as completed and 'Review' as the active step. The main content area displays a summary of the configuration. It includes a 'Runtime' section with 'Runtime type' as 'Mule Gateway' and 'Proxy type' as 'Basic Endpoint'. Below this is an 'API' section with 'API name' as 'customers_api', 'API version' as 'v1', 'Asset version' as '1.0.0', and 'Conformance status' as 'Not Validated'. There are 'Edit' links for each of these sections. The 'Downstream' section shows 'Scheme' as 'HTTP'. At the bottom, there are three buttons: 'Cancel', 'Previous', and 'Save'. A large red arrow points to the 'Save' button.

Ilustración 74. Summary

Cuando la API haya sido creada, necesitaremos guardar en algún lugar accesible nuestro *API Instance ID*.

The screenshot shows the Salesforce API Manager interface. The top navigation bar includes the Salesforce logo and 'API Manager'. Below it, the breadcrumb trail is 'API Administration (Sandbox) > customers_api (v1) - API Summary'. The left sidebar contains a 'SANDBOX' tab and a list of navigation items: 'API Administration', 'API Summary', 'Contracts', 'Policies', 'SLA Tiers', 'Settings', and 'Governance Report'. The main content area is titled 'APIs / customers_api / API Summary'. A blue banner at the top of the main content area contains an information icon and the text: 'To complete the registration process, you need to connect this API to your Mule application using Autodiscovery. [Learn more](#)'. Below this banner is a table with the following data:

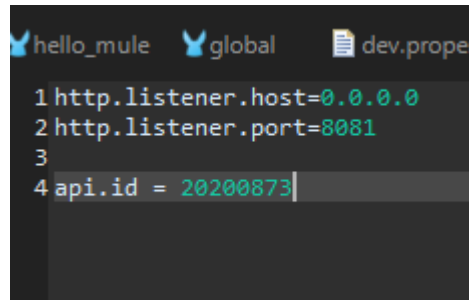
Type	Asset Version	Implementation URI ⓘ	API Label ⓘ
RAML/OAS	1.0.0 (Latest)	N/A	-
API Status	Consumer Endpoint	API Instance ID ⓘ	Instance Conformance ⓘ
● Unregistered	N/A	20200873	Not Validated

Below the table, there is a 'Tags' section with an 'Add New Tag +' button. At the bottom of the main content area, there is a 'Key Metrics' section.

Ilustración 75. API instance ID

5. Configuración de API Autodiscovery

Para configurar nuestro proyecto en Anypoint Studio en API Autodiscovery agregaremos nuestro *api instance id* como una propiedad en los archivos de dev.properties y local.properties.



```
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
3
4 api.id = 20200873
```

Ilustración 76. local.properties

Volviendo a nuestro archivo global.xml, agregaremos una nueva propiedad de tipo API Autodiscovery.

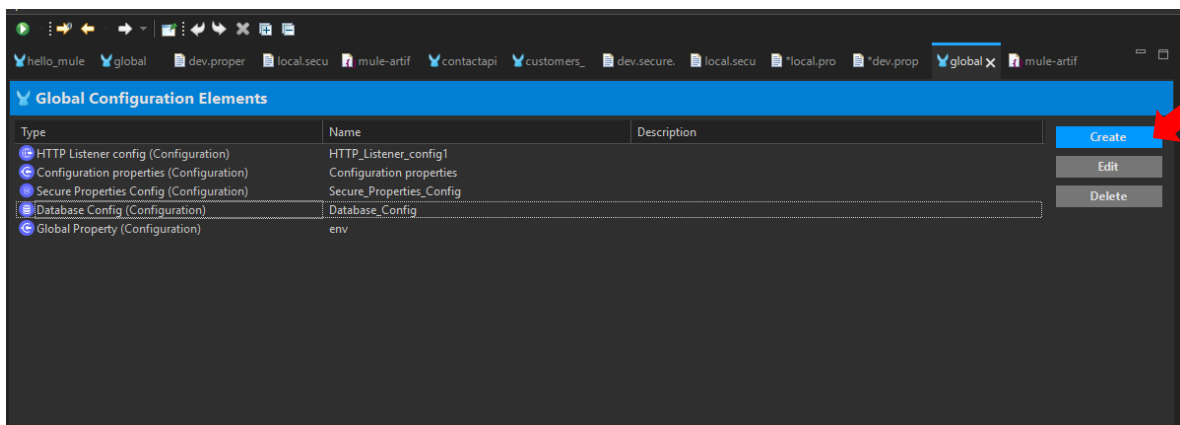


Ilustración 77. Global XML

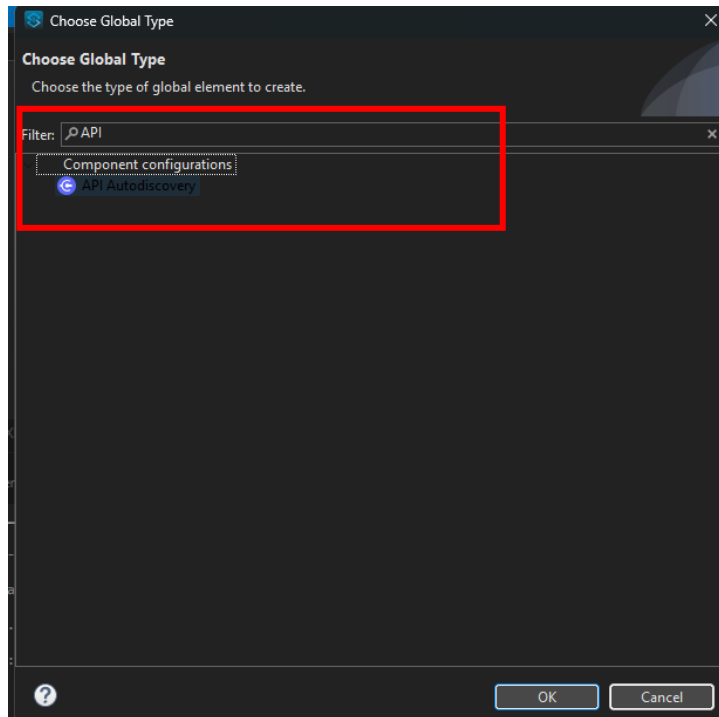


Ilustración 78. API Autodiscovery

Dentro de dicha propiedad, colocaremos en la sección de API id, el nombre agregado en nuestros archivos de local.properties y dev.properties, en este caso el nombre fue `${api.id}`. Así mismo en la sección de Flow Name escribiremos el nombre del flujo con el que hemos estado trabajado hasta el momento. En este caso es `customers_apiFlow`

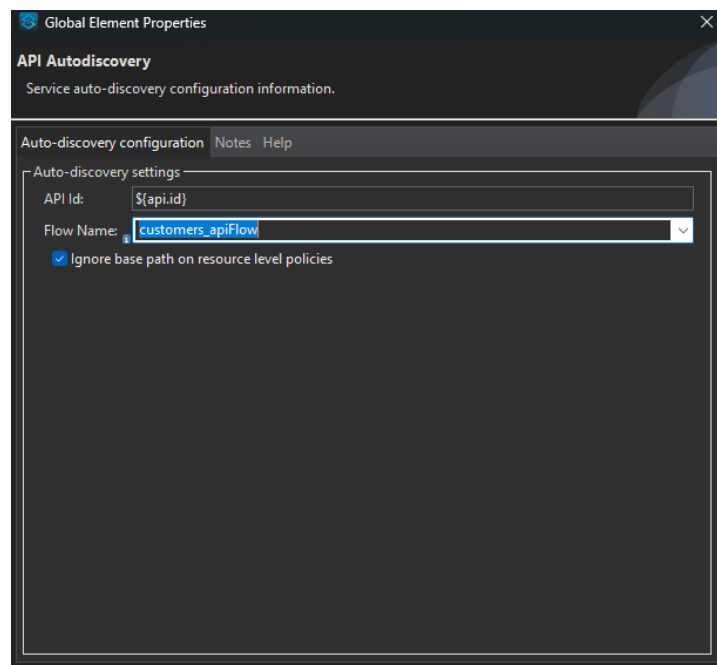


Ilustración 79. Global Element Properties

5.2 Client ID y Client Secret

El siguiente paso es ir a nuestro API Manager en Anypoint Platform y dar click en el botón de Environment

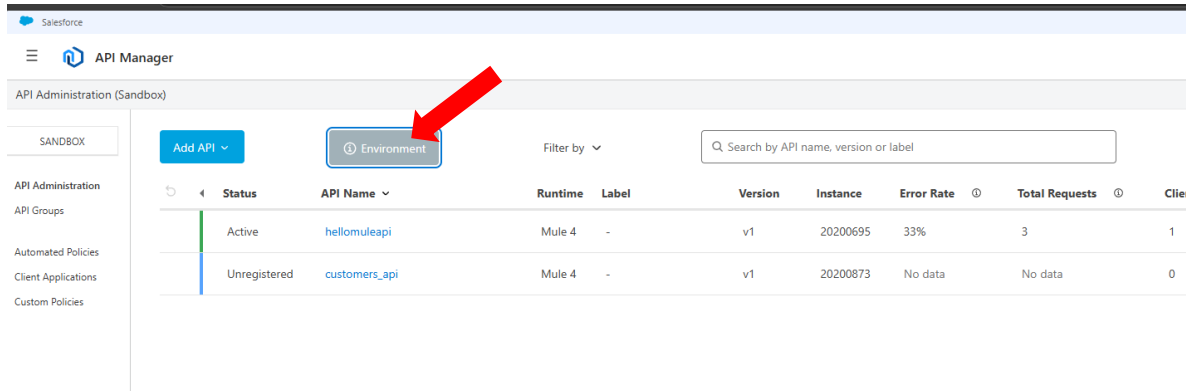


Ilustración 80. Environment

Se abrirá una nueva ventana, de dónde obtendremos las credenciales de *Client ID* y *Client secret*. Las cuales utilizaremos para enlazar nuestro proyecto local con la API de API Manager.

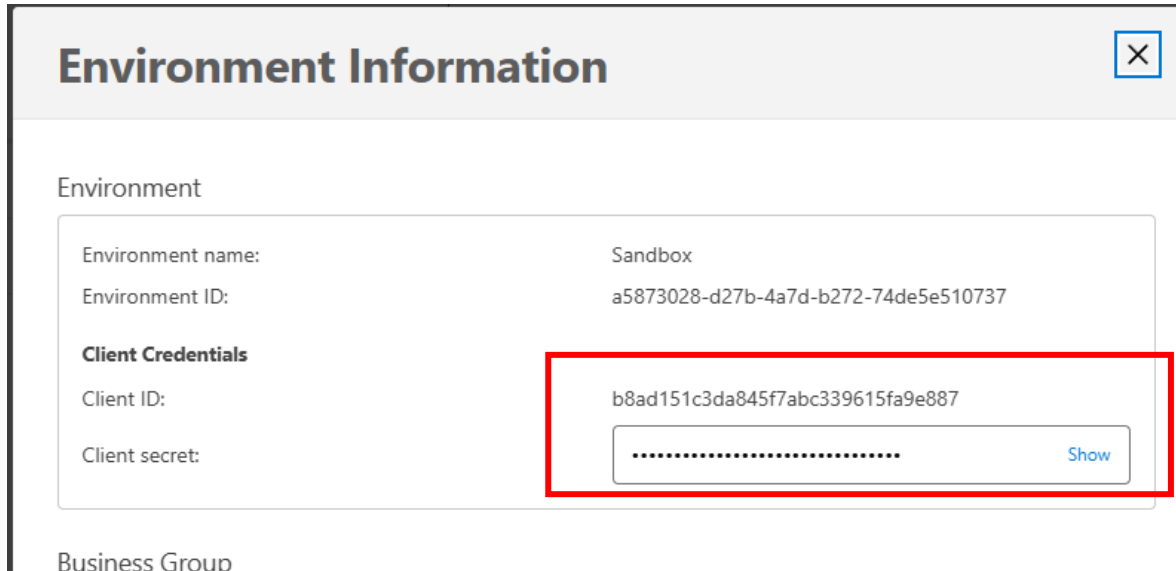


Ilustración 81. Client Credentials

Dentro de nuestro Anypoint Studio, daremos click en **Window -> Preferences**.

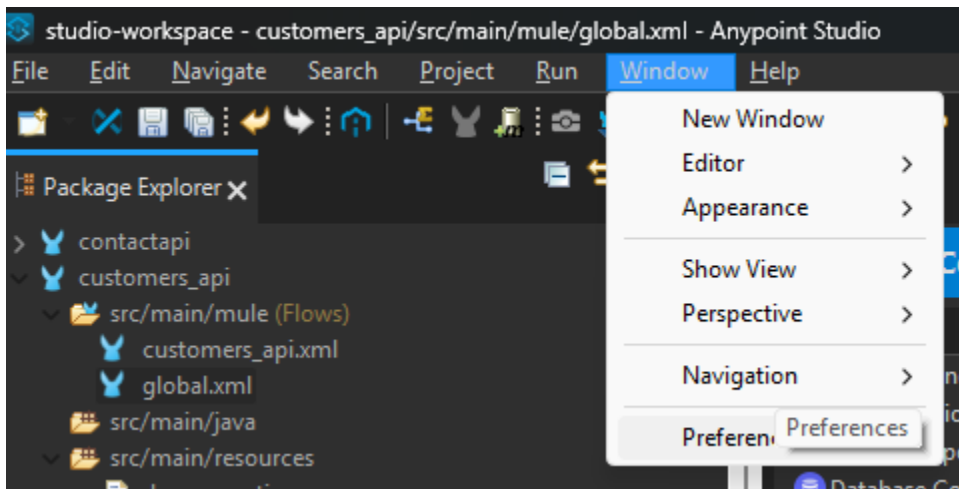


Ilustración 83. Preferences

Después dentro de la ventana de Preferences, en la sección de API Manager, agregaremos nuestro *client id* y Client Secret. Para finalmente dar click en el botón de Validate. Deberíamos obtener una confirmación, así como el nombre de nuestra organización, en este caso SPS. Finalmente daremos click en Apply, y Apply and Close

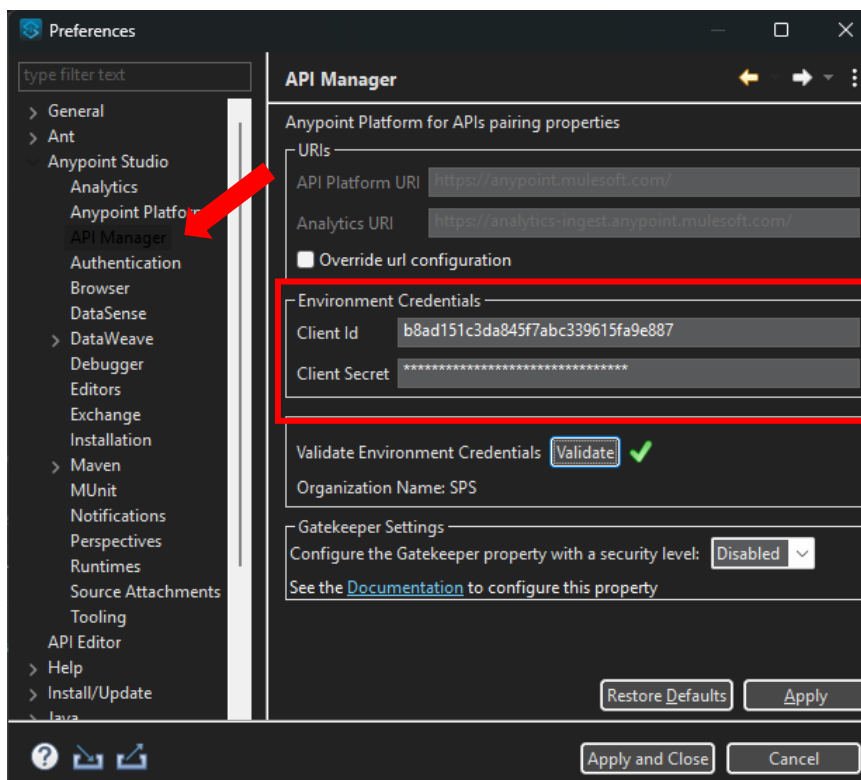


Ilustración 82. Enviroment Credentials

5.3 Redespliegue de nuestra aplicación

Para finalizar con el proceso de enlace de nuestro proyecto local con API Manager, necesitaremos volver a desplegar nuestro proyecto agregando las propiedades de `anypoint.platform.client_id` y `anypoint.platform.client_secret`

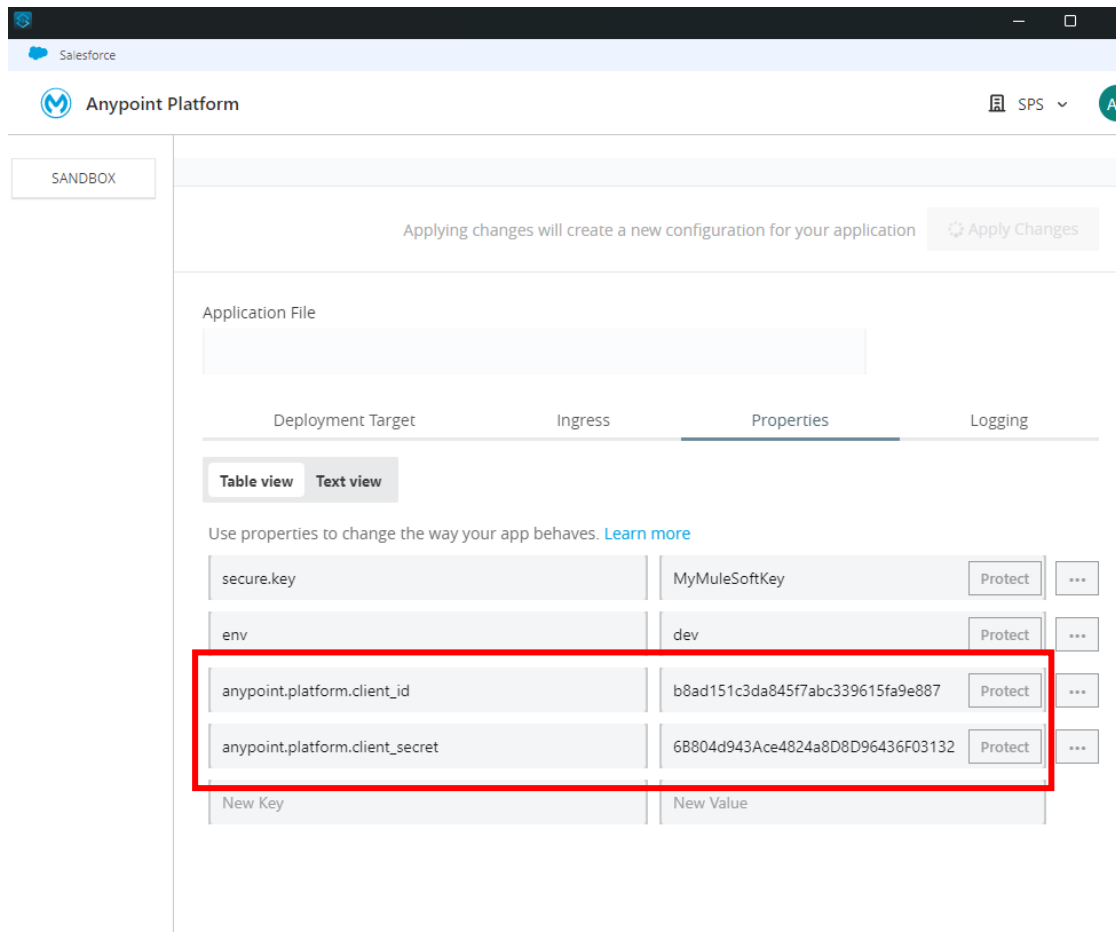


Ilustración 84. Anypoint platform properties

Posteriormente al ingresar al API Manager podremos verificar que nuestro API local fue enlazada correctamente en la sección de API Status con el estado de Active

The screenshot displays the Salesforce API Manager interface. The top navigation bar includes the Salesforce logo and the 'API Manager' title. Below this, a breadcrumb trail shows 'API Administration (Sandbox)' and 'customers_api (v1) - API Summary'. The left sidebar contains a menu with options: 'SANDBOX', 'API Administration', 'API Summary', 'Contracts', 'Policies', 'SLA Tiers', 'Settings', and 'Governance Report'. The main content area is titled 'APIs / customers_api / API Summary' and contains a table with the following data:

Type	Asset Version	Implementation URI ⓘ	API Label ⓘ	API Version
RAML/OAS	1.0.0 (Latest)	N/A	-	v1
API Status	Consumer Endpoint	API Instance ID ⓘ	Mule Version	Java Version ⓘ
● Active	N/A	20200873	4.9.0	17

Below the table, there is a section for 'Instance Conformance ⓘ' with the status 'Not Validated'. At the bottom, there is a 'Tags' section with an 'Add New Tag +' button.

Ilustración 85. API Status

6.1- Aplicación de política Client ID Enforcement.

Hasta el momento, la API desarrollada responde a cualquier petición siempre y cuando se tenga acceso a la url de la misma. Esto no siempre es conveniente ya podría ser vulnerable a ataques de denegación de servicio, por lo que con ayuda de API Manager, agregaremos políticas que refuercen la seguridad de nuestra API.

Dentro de la sección de API Summary en API Manager, daremos click a la opción de *Policies*

The screenshot shows the Salesforce API Manager interface. At the top, there's a header with the Salesforce logo and 'API Manager'. Below that, a breadcrumb trail shows 'API Administration (Sandbox)' and 'customers_api (v1) - API Summary'. The main content area is titled 'APIs / customers_api / API Summary'. On the left, a sidebar lists navigation options: API Summary, Contracts, Policies, SLA Tiers, Settings, and Governance Report. The main content area displays the following information:

Type	Asset Version	Implementation URI ⓘ
RAML/OAS	1.0.0 (Latest)	N/A

API Status	Consumer Endpoint	API Instance ID ⓘ
● Active	N/A	20200873

Instance Conformance ⓘ
Not Validated

Tags
Add New Tag +

Key Metrics

Ilustración 86. API Summary

Después daremos click en Add Policy

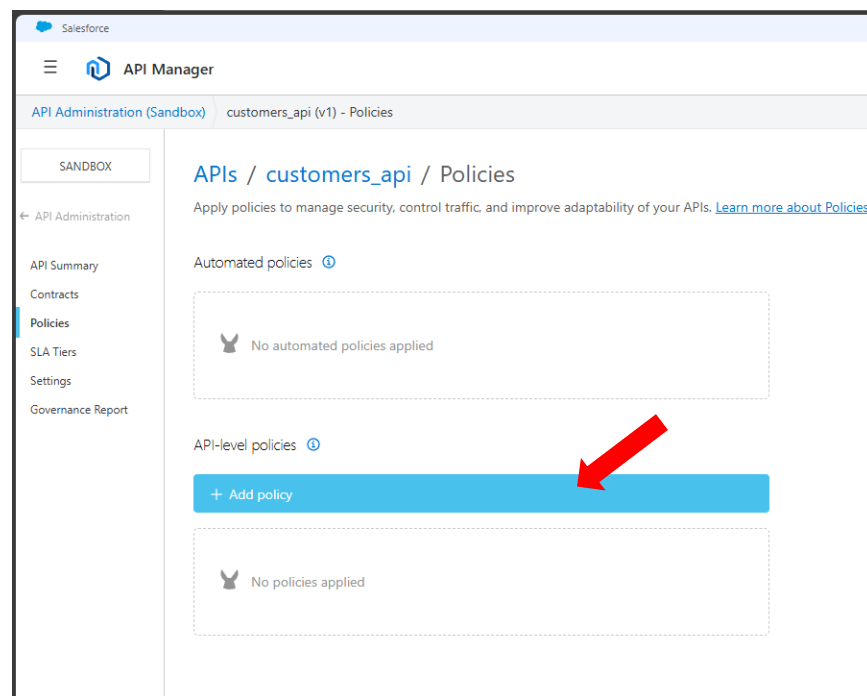


Ilustración 87. Add Policy

Utilizando la barra de búsqueda, seleccionaremos la política de *Client ID Enforcement*

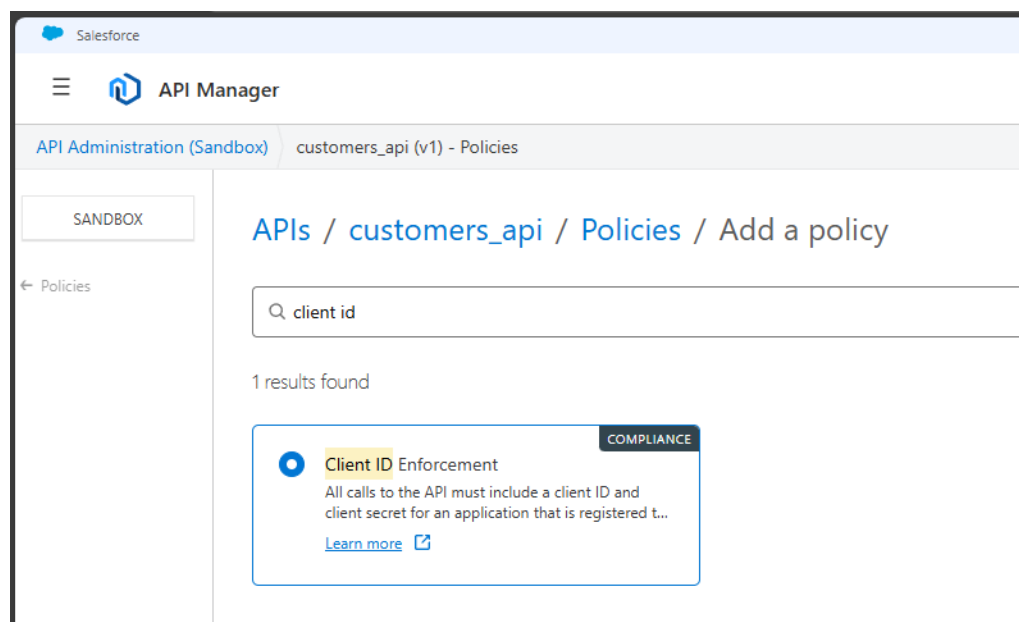


Ilustración 88. Client ID Enforcement

Seleccionaremos las opciones de *HTTP Basic Authentication Header* y *Apply configuration to all API method and resources*. Para finalmente dar click en *Apply*

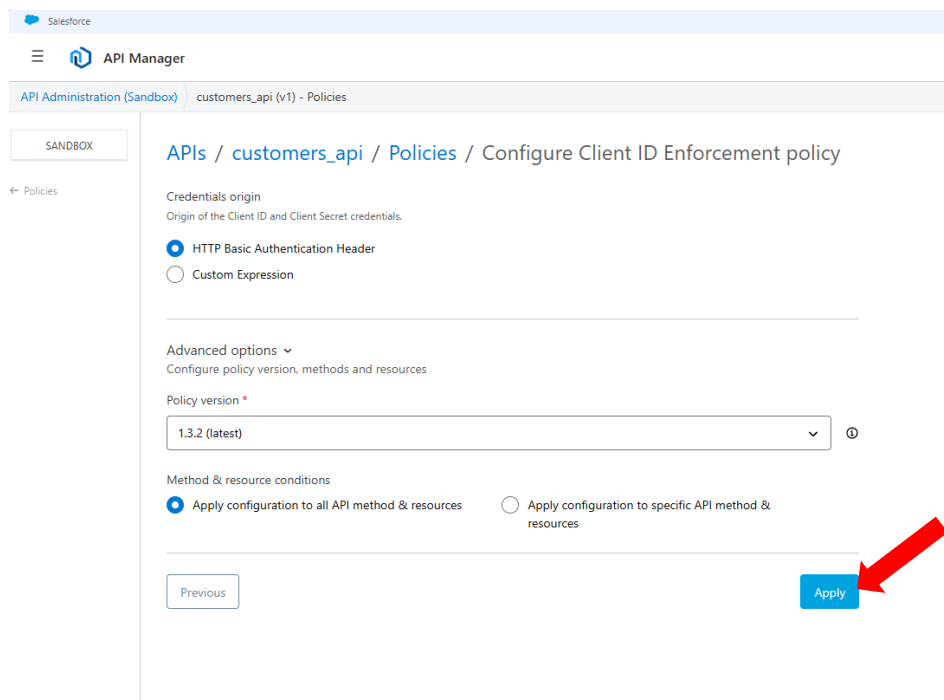


Ilustración 89. Configure Client ID Enforcement policy

Para comprobar que la política se haya aplicado correctamente, utilizaremos Postman para enviar una petición, deberíamos de obtener un error 401 Unauthorized. Esto quiere decir que la política se aplicó correctamente a nuestra API y que ya no es suficiente contar con al url para poder realizar peticiones.

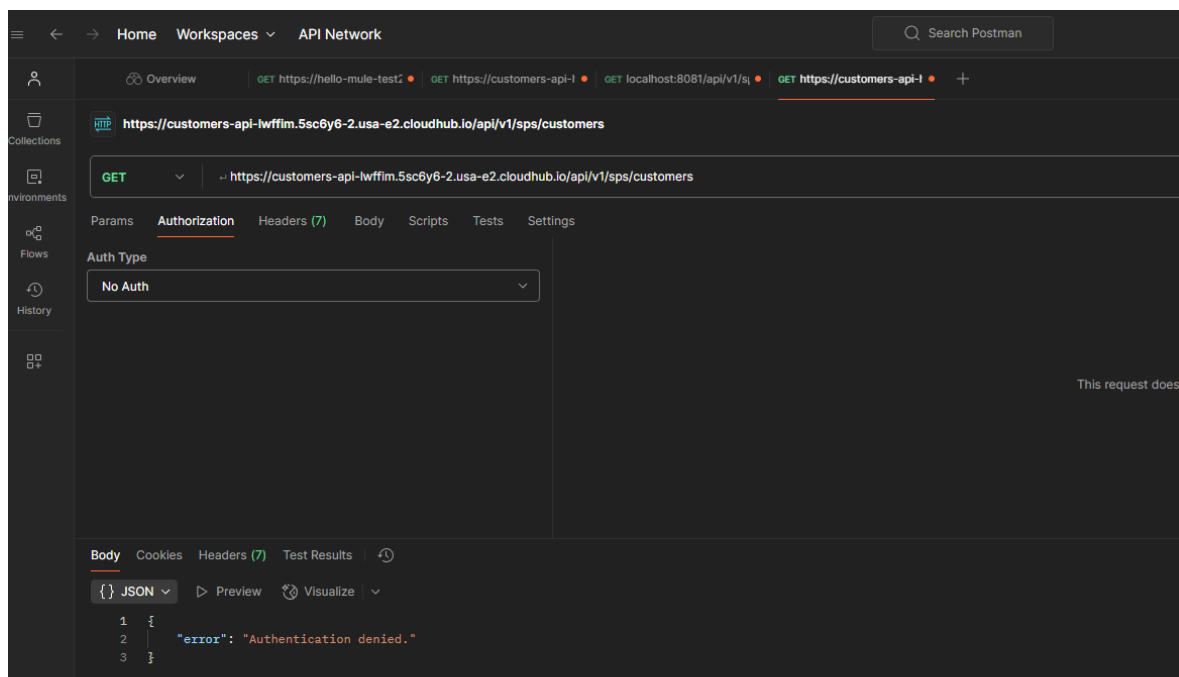


Ilustración 90. Error 401. Unauthorized

6.2 Obtención de credenciales

Con la política aplicada, necesitaremos de credenciales para poder realizar peticiones a nuestra API, para esto, nos dirigiremos a Exchange en Anypoint Platform, y daremos click en el botón de Request Access.

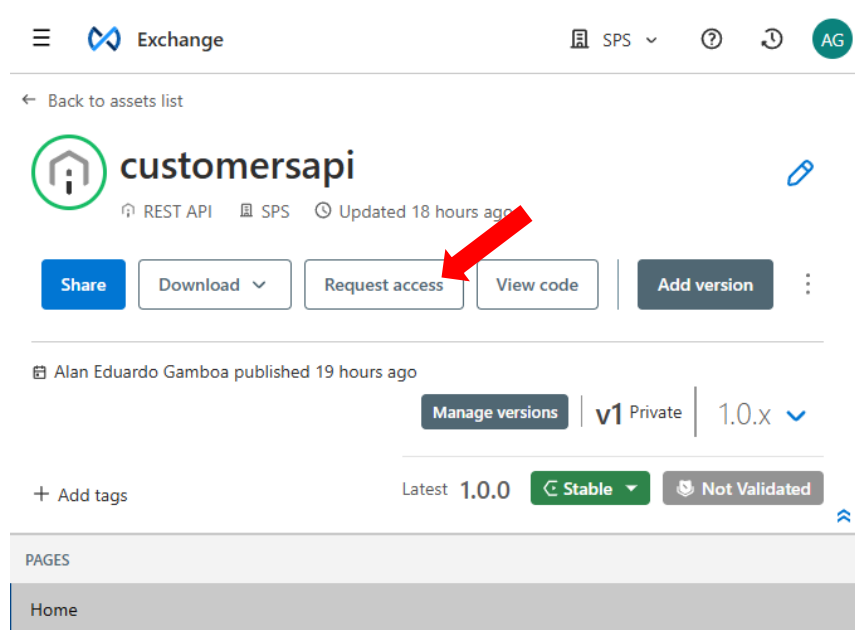


Ilustración 91. Request access

Esto abrirá una ventana dónde seleccionaremos la instancia de la api y nuestra aplicación, en este caso customersapp. Posteriormente daremos click en Request access

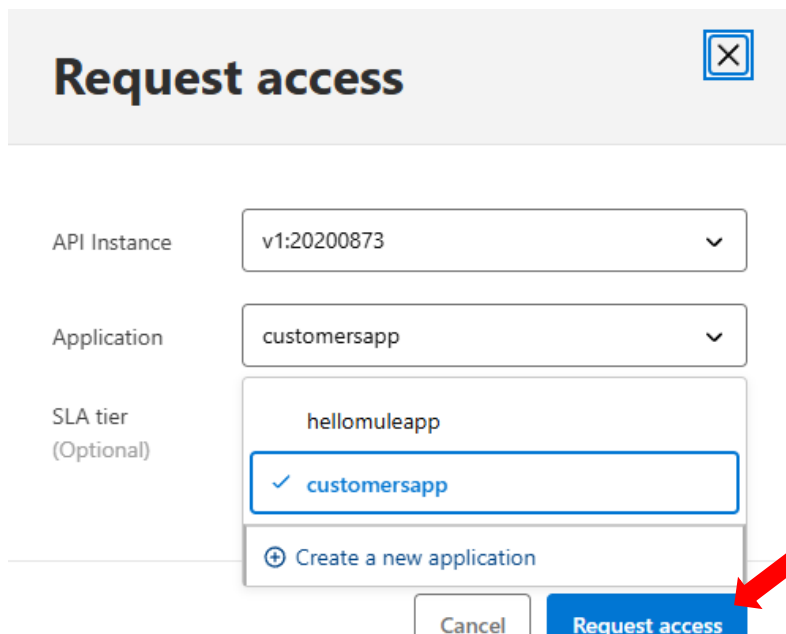


Ilustración 92. SLA tier

Con esto obtendremos una client id y client secret que utilizaremos para poder enviar peticiones a nuestra API

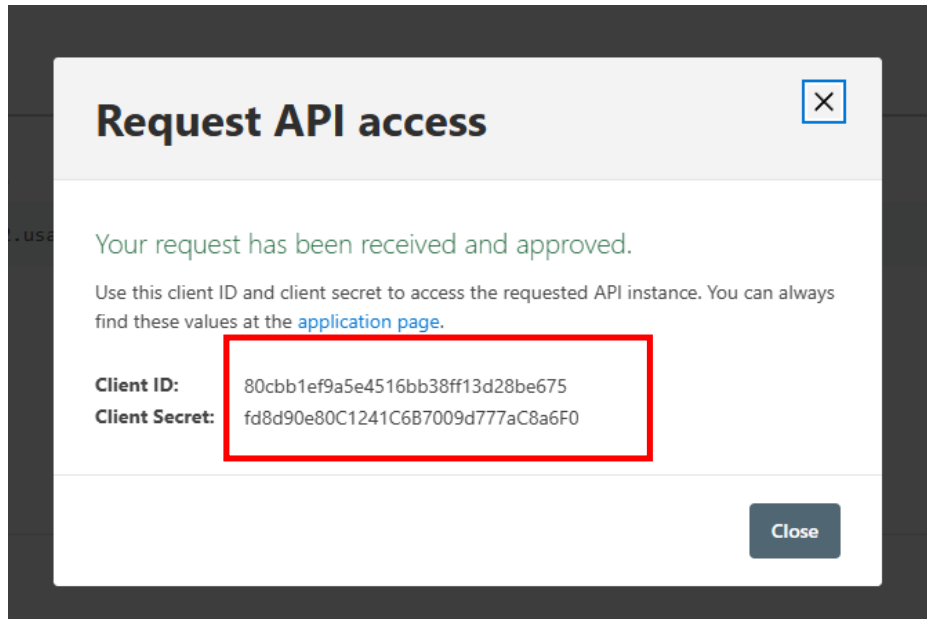


Ilustración 93. Request API ACCESS

Finalmente en Postman, en la sección de Authorization seleccionaremos una autenticación del tipo Basic Auth, del lado derecho en los parámetros username y password, agregaremos las llaves previamente obtenidas. Al momento de realizar la petición, obtendremos un 200-OK, lo cuál significa que la política y la autenticación funcionan correctamente.

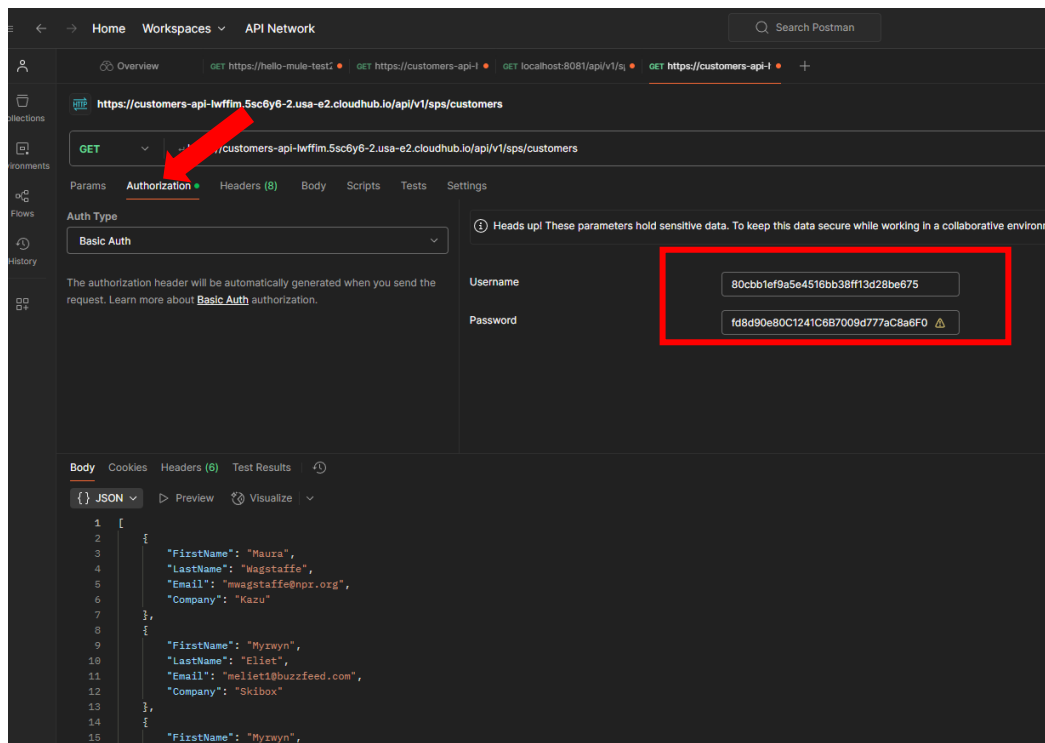


Ilustración 94. Request de prueba

Recursos

- <https://developer.mulesoft.com/tutorials-and-howtos/getting-started/hello-mule/>
- How to set up your global elements and properties files in Anypoint Studio. Utiliza archivo de propiedades para mantener y referenciar datos sensibles por separado al código generado.
<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/global-elements-properties-files/>
- How to secure properties before deployment in Anypoint Studio. Crea archivo de propiedades seguros para proteger datos sensibles que representan un riesgo mantenerlos en claro.
<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/how-to-secure-properties-before-deployment>
- Best practices to design your first API Specification. Conoce sobre el enfoque de first API Specification.
<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/best-practices-first-api-spec/>
- Build your first API Specification with API Designer. Construye un API Specification en API Designer.
<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/build-api-specification-api-designer/>
- How to set up API Autodiscovery in Anypoint Studio. Crear un API en API Manager para permitir aplicar políticas.
<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/how-to-setup-api-autodiscovery-anypoint-studio>
- API design best practices and applying client ID enforcement. Aplicar Client ID enforcement policy a API en API Manager.
<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/client-id-policy/>

