



# Práctica

# MuleSoft Trainee

Torre Esmeralda II, 10, Perif. Blvd. Manuel Ávila Camacho 36,  
Lomas de Chapultepec, 11000 Ciudad de México, CDMX.

Información Confidencial

## Contenido

Resumen .....	3
Resultados .....	4
¿Cómo probar? .....	5
En Postman .....	5
En Windows PowerShell usando CURL .....	7
Desarrollo de Práctica .....	8
Archivo de implementación .....	9
Path .....	10
Conexión con la Base de datos .....	10
global.xml y env.properties (local, dev) .....	12
Configuration properties .....	16
Global Property .....	17
Secure Properties .....	18
Databaseconfig en xml con secure:: .....	19
¿Cómo le indico a la etiqueta de configuración de base de datos que valores como db.host, db.database vienen del local.properties y el db.user con db.password vienen de local.secure.properties? .....	20
Especificación de API .....	21
Cloudfoundry .....	22
EXTRA 1: Crear un API en API Manager .....	24
Abrir API Manager .....	24
Selección de Runtime: ¿Flex Gateway o Mule Gateway? .....	25
Selección del tipo de API (Asset Type) a gestionar .....	26
Downstream .....	26
Upstream .....	27
API Registrada exitosamente .....	27
EXTRA 2: API Autodiscovery .....	28
Obtener Client ID y Client Secret .....	29
Explorando Logs .....	31
Status Active .....	32
EXTRA 3: Aplicar la política de client-id Enforcement para proteger nuestra API. ....	33
Agregar Policy .....	33

¿HTTP Basic Authentication Header o Custom Expression? .....	33
Imagen del resultado obtenido de toda la práctica .....	36
ADICIONAL EXTRA: JWT VALIDATION .....	36
JWT Origin.....	37
JWT Signing Method .....	38
JWT Signing Key Length.....	38
JWT Key origin y JWT Key .....	39
Expiration Claim Mandatory y Validate Custom Claim.....	39
Resultados del punto adicional .....	41

## Desarrollo de API para Exponer Información de Clientes – Práctica MuleSoft

### Resumen

Como parte del entrenamiento en MuleSoft, desarrollé una solución que permite exponer datos de clientes almacenados en una base de datos MySQL mediante una API. Esta solución tiene como objetivo facilitar al equipo de marketing el acceso a datos.

#### *Requisitos Implementados*

- **Ruta de exposición:** La API expone los datos de clientes a través de la ruta `/api/v1/sps/customers`.
- **Gestión de entornos:** Se crearon dos archivos de propiedades, uno para entorno local y otro para desarrollo (dev).
- **Organización del proyecto:** La solución está estructurada en dos archivos principales: uno para la implementación lógica y otro para componentes globales.
- **Seguridad:** Las credenciales de conexión a la base de datos están protegidas utilizando *Secure Properties*.
- **Diseño de la API:** Se construyó la especificación de la API (RAML).
- **Despliegue:** La aplicación fue desplegada correctamente en CloudHub.

#### *Extras Implementados*

1. **Publicación de la API en API Manager.**
2. **Configuración de API Autodiscovery** en la aplicación para su administración desde API Manager.
3. **Aplicación de la política de seguridad *Client ID Enforcement*** para controlar el acceso a la API.

#### *Exploración Adicional*

Además de los puntos requeridos y los extras, realicé una exploración adicional sobre autenticación mediante **JWT Validation**, configurando esta política en API Manager para comprender cómo implementar en Mulesoft otras formas de control de acceso y seguridad.

## Resultados

Nota: Esta sección fue hecha con la finalidad de demostrar el funcionamiento. Más adelante, en el apartado de Desarrollo de Práctica, estará descrito el procedimiento, análisis y aprendizaje.

*Api desplegado en cloudfury:*

- <https://sps-customers-e96hyd.5sc6y6-2.usa-e2.cloudfury.io/api/v1/sps/customers>

*Parámetros requeridos por el Client ID Enforcement:*

- client\_id: cbca3d1cce834f2abf45364d216862fa
- client\_secret: 40EC2EA4Fb6e40c2a0cd553393aa8e89

*Especificación del JWT:*

Para poder generar un token válido, es imprescindible que este contenga, key "role" con el valor "marketing", ya que la validación configurada en API Manager – Policy – JWT Validation permite el acceso al recurso (GET para obtener clientes) únicamente a aquellos usuarios cuyo rol sea de marketing. Además, es fundamental que el valor de expiración (exp) del token no haya caducado, ya que también se valida este parámetro.

Para facilitar la prueba, le sugiero copiar y pegar el token resaltado en amarillo, el cual cumple con todos los requisitos mencionados y con las especificaciones del JWT.

### Header: Algorithm & Token Type

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

### PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "PriscilaM",
  "role": "marketing",
  "exp": 4102444800
}
```

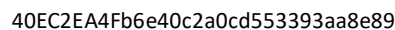
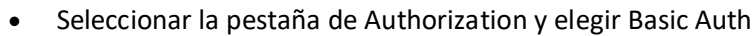
### SIGN JWT: SECRET

a-string-secret-at-least-256-bits-long

### TOKEN

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJkMjM0NTY3ODkwIiwibmFtZSI6IkJyaXNjaWxhTSIsInJvbGUiOiJtYXJrZXRpbmciLCJleHAiOiJxMDI0NDQ4MDB9.F1rU9Ghm2JXgDru3YfsG5V-HSeTNOiTDi_R7TwfxdpU
```

- Copiar la URL desplegada y seleccionar el método GET



- eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJsMjM0NTY3ODkwIiwibmFtZSI6IiBjaXNjaWxhTSIsInJvbGU6IjE0Jm9udG4iLCJleHAiOiJxMDI0NDQ4MDB9.F1rU9Ghm2JXgDru3YfsG5V-HSeTNOiTdi\_R7TwxdpU

Respuesta:

The screenshot shows a REST client interface with the following details:

- URL:** `https://sps-customers-e96hyd.5sc6y6-2.usa-e2.cloudhub.io/apl/v1/sps/customers`
- Method:** GET
- Headers:** One header is visible: `x-access-token` with a long alphanumeric value.
- Status:** 200 OK, 962 ms, 1.06 KB
- Body:** A JSON array of customer objects.

The JSON response is as follows:

```
[
  {
    "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": "Axi",

```

## En Windows PowerShell usando CURL

- Abrir powershell y escribir lo siguiente en consola:  

```
curl.exe -u cbca3d1cce834f2abf45364d216862fa:40EC2EA4Fb6e40c2a0cd553393aa8e89 -H "x-access-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IiByaXNjaWxhTSIsInJvbmGUiOiJtYXJrZXRpbnmcilCJleHAiOiQxMDI0NDQ4MDB9.F1rU9Ghm2JXgDru3YfsG5V-HSeTNOiTdi_R7TwfxdpU" -X GET "https://sps-customers-e96hyd.5sc6y6-2.usa-e2.cloudhub.io/api/v1/sps/customers"
```

*Respuesta:*

```
PS C:\Users\martc> curl.exe -u cbca3d1cce834f2abf45364d216862fa:40EC2EA4Fb6e40c2a0cd553393aa8e89
-H "x-access-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IiByaXNjaWxhTSIsInJvbmGUiOiJtYXJrZXRpbnmcilCJleHAiOiQxMDI0NDQ4MDB9.F1rU9Ghm2JXgDru3YfsG5V-HSeTNOiTdi_R7TwfxdpU" -X GET "https://sps-customers-e96hyd.5sc6y6-2.usa-e2.cloudhub.io/api/v1/sps/customer
s"
[
  {
    "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",
    "Company": "NPR"
  }
]
```



## Desarrollo de Práctica

*"Cada vez que hacía un paso técnico, me detenía a preguntarme: ¿qué significa esto? ¿Por qué lo estoy haciendo así?"*

Esa curiosidad fue clave para realmente **entender** cómo funciona MuleSoft.

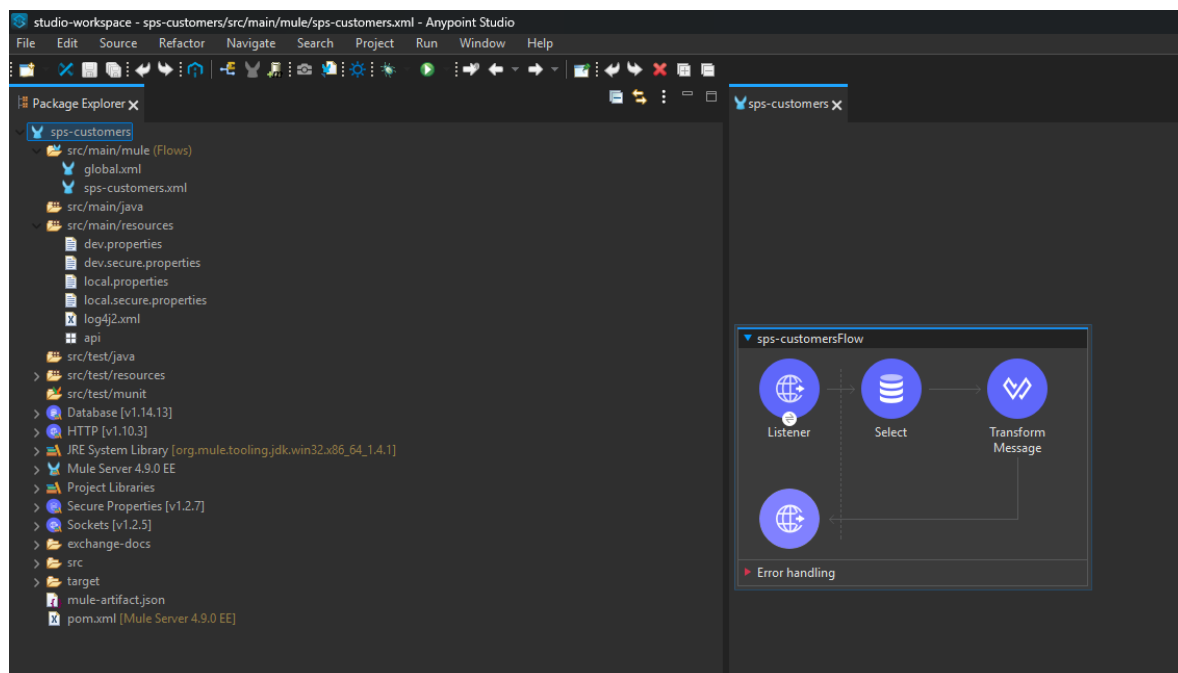
Tuve la oportunidad de revisar el siguiente enlace:

<https://developer.mulesoft.com/tutorials-and-howtos/getting-started/hello-mule/>,

en el cual exploré las distintas secciones del entorno de desarrollo (IDE).

Particularmente, en la sección donde se muestra la estructura del proyecto (archivos y carpetas), observé que es posible crear directamente los archivos necesarios. Sin embargo, opté por iniciar utilizando la paleta de herramientas, con el objetivo de familiarizarme mejor con el entorno de desarrollo.

Una vez logrado esto, procedí a organizar y modularizar el proyecto a través de los archivos `global.xml`, `local.properties`, `dev.properties`, `local.secure.properties` y `dev.secure.properties`, etc.



## Archivo de implementación

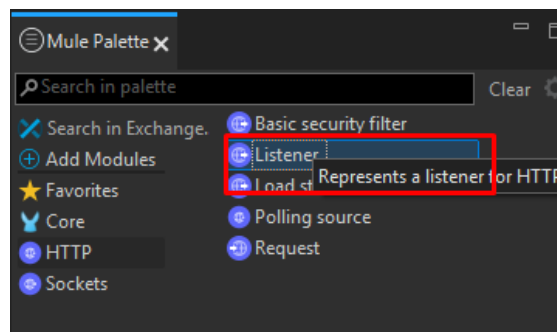
Hice un New Mule Project, nombrándolo sps-customers



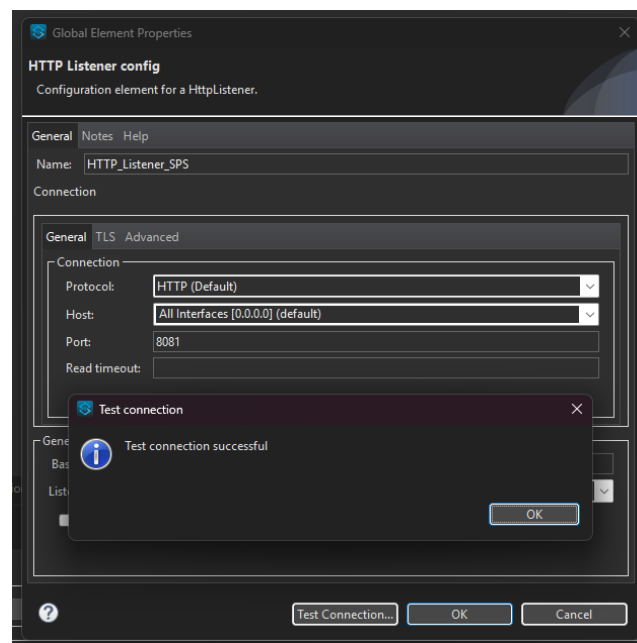
Después de crear el proyecto en MuleSoft, me dirigí a la paleta de componentes y seleccioné el HTTP Listener.

*¿Por qué iniciar con este componente?*

Porque el Listener permite exponer una API al recibir solicitudes HTTP. Es decir, actúa como el punto de entrada al flujo, lo cual es esencial para comenzar a construir y probar la lógica de la aplicación desde un endpoint accesible.



Seleccioné el puerto **8081** como puerto local del Listener, ya que es comúnmente utilizado para pruebas en desarrollo. Realicé una prueba de conexión (Test Connection) para asegurarme de que ningún otro proceso en mi computadora estuviera ocupando dicho puerto

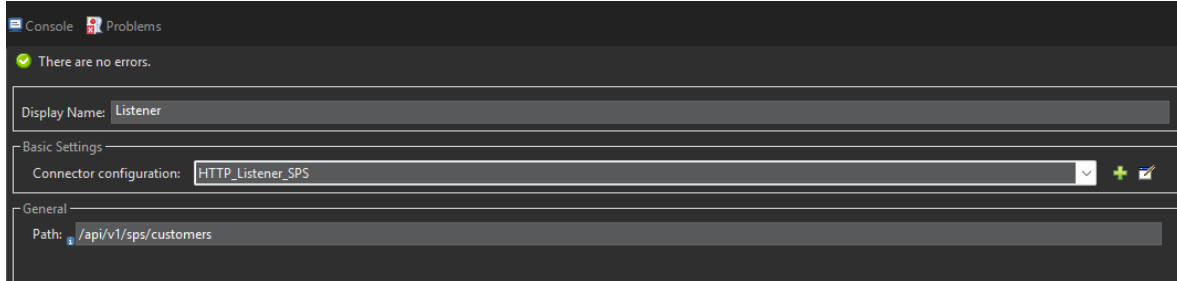


## Path

En la configuración del **HTTP Listener**, se define la **ruta base** de la API. En este caso, establecí la siguiente:

/api/v1/sps/customers

Esta ruta representa el punto al que se dirigirán las solicitudes HTTP entrantes, y responde a una convención común en el diseño de APIs RESTful, donde se incluye la versión (v1) y el recurso (customers) dentro de un contexto más amplio (sps).

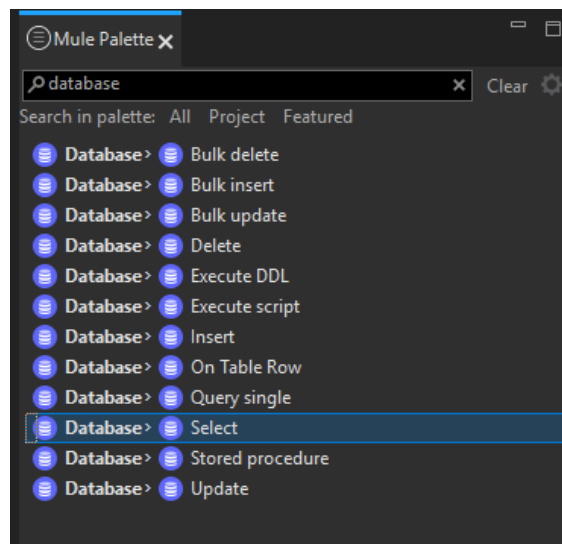


## Conexión con la Base de datos

Para realizar la conexión con una base de datos MySQL, al principio no tenía claro en qué parte del IDE debía configurarse. Por ello, decidí buscar un nuevo recurso visual que me ayudara a orientarme dentro del entorno, y encontré útil el siguiente video:

<https://www.youtube.com/watch?v=1eT-Gdi2Znw>

A partir de ello, identifiqué que la paleta de componentes incluye una sección llamada Database, que proporciona operaciones como Select, entre otras.



Al incorporar esta operación en el flujo, el IDE solicita configurar una Database Configuration (DB Config). Esta configuración me permitió establecer los parámetros necesarios para conectarse a MySQL: el host, el puerto, el usuario, la contraseña y el nombre de la base de datos.

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:mysql-connector-java:8.0.30) Modify...

Connection

Host: fx mudb.learn.mulesoft.com

Port: fx 3306

User: fx mule

Password: •••• ☐ Show password

Database: fx training

? Test Connection... OK Cancel

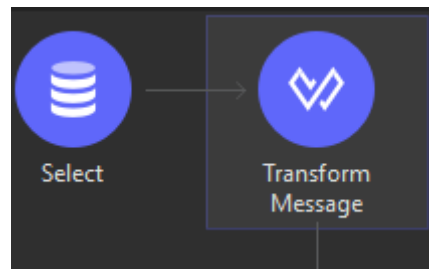
Así mismo, hacer la consulta.

```
Query
SQL Query Text:
SELECT * FROM customers;
```

Dentro de ese mismo recurso, observé que también era necesario utilizar el componente Transform Message en MuleSoft.

*¿Cuál es el propósito de este componente?*

El Transform Message permite mapear, transformar y estructurar los datos que fluyen entre componentes, ya sea desde una base de datos, una solicitud HTTP o cualquier otra fuente. En este caso, su uso fue fundamental para convertir la respuesta obtenida desde MySQL en una estructura adecuada para la respuesta de la API, como un objeto JSON limpio y bien definido.



```
Output Payload
1 %dw 2.0
2 output application/json
3 ---
4 payload
```

### [global.xml y env.properties \(local, dev\)](#)

Una vez que la primera parte del flujo funcionó correctamente, procedí a realizar una mejora importante: separar los datos sensibles del flujo principal y moverlos a una configuración externa en el archivo global.xml.

*¿Qué debe contener este archivo?*

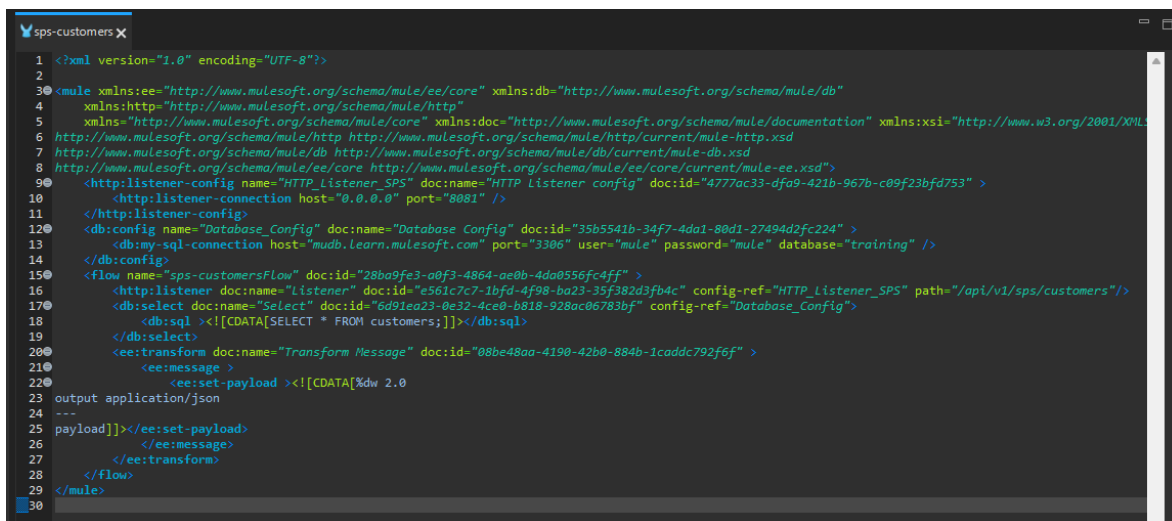
El archivo global.xml se utiliza para definir configuraciones compartidas que serán reutilizadas en distintos puntos del proyecto. En el contexto de esta aplicación, incluye elementos como:

- La configuración del HTTP Listener.

- La configuración de la base de datos (DB Config), que hace referencia a propiedades externas para evitar exponer credenciales directamente en el flujo.
- La declaración de variables o propiedades globales que pueden ser referenciadas mediante el lenguaje \${property.name}.
- Referencias a archivos de propiedades (.properties y .secure.properties) que contienen datos sensibles como usuarios, contraseñas, URLs, etc.

Esta práctica no solo mejora la seguridad del proyecto, sino que también favorece la modularidad, la reutilización de configuraciones

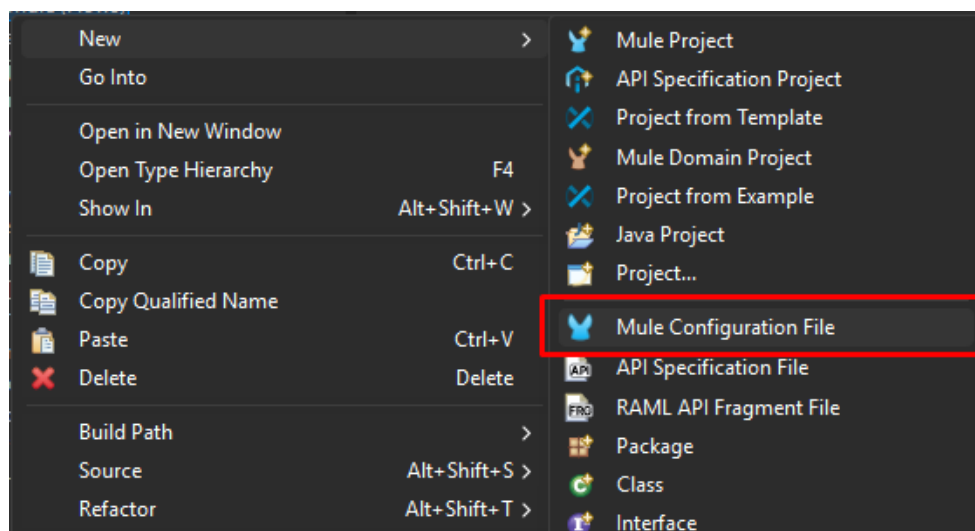
Observemos el contenido XML del flujo principal para separarlo:



```

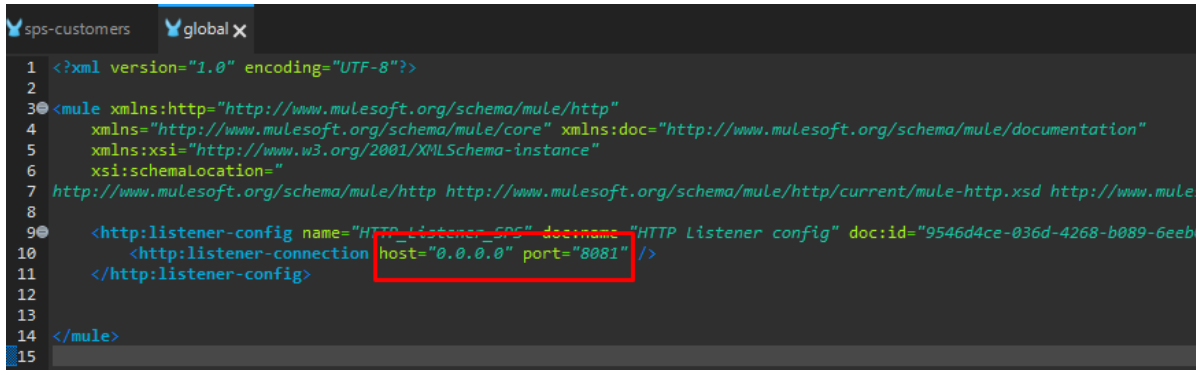
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core" xmlns:db="http://www.mulesoft.org/schema/mule/db"
4     xmlns:http="http://www.mulesoft.org/schema/mule/http"
5     xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/documentation" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
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     http://www.mulesoft.org/schema/mule/ee/core http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd">
9     <http-listener-config name="HTTP Listener SPS" doc:name="HTTP Listener config" doc:id="4777ac33-dfa9-421b-967b-c09f23bfd753">
10         <http-listener-connection host="0.0.0.0" port="8081" />
11     </http-listener-config>
12     <db:config name="Database Config" doc:name="Database Config" doc:id="35b5541b-34f7-4da1-80d1-27494d2fc224">
13         <db:mysql-connection host="mudb.learn.mulesoft.com" port="3306" user="mule" password="mule" database="training" />
14     </db:config>
15     <flow name="sps-customersFlow" doc:id="28ba9fe3-a0f3-4864-ae0b-4da8556fc4ff">
16         <http-listener doc:name="Listener" doc:id="e561c7c7-1bfd-4f98-ba23-35f382d3fb4c" config-ref="HTTP Listener SPS" path="/api/v1/sps/customers"/>
17         <db:select doc:name="Select" doc:id="6d91ea23-0e32-4ce0-b818-928ac06783bf" config-ref="Database Config">
18             <db:sql><![CDATA[SELECT * FROM customers;]]></db:sql>
19         </db:select>
20         <ee:transform doc:name="Transform Message" doc:id="08be48aa-4190-42b0-884b-1caddc792f6f">
21             <ee:message>
22                 <ee:set-payload><![CDATA[%dw 2.0
23 output application/json
24 ---
25 payload]]></ee:set-payload>
26             </ee:message>
27         </ee:transform>
28     </flow>
29 </mule>
30
  
```

Creemos un archivo MULE:



Dentro de la etiqueta `<mule></mule>`, incluí la primera configuración global: la del **HTTP Listener**.

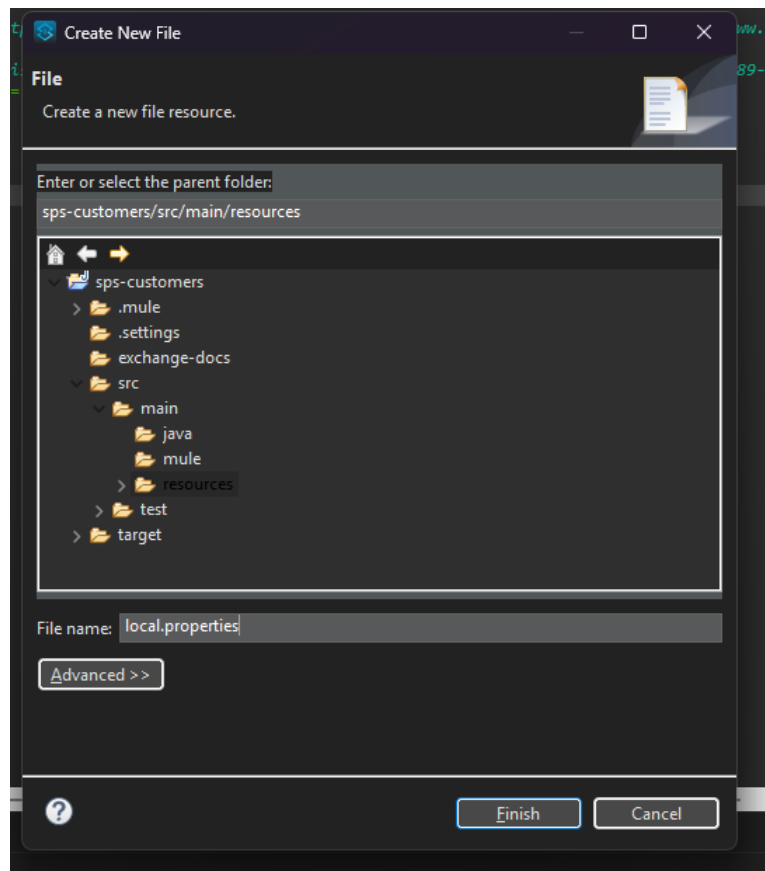
En esta etapa inicial, la configuración aún contiene valores expuestos directamente, como el host y el port.



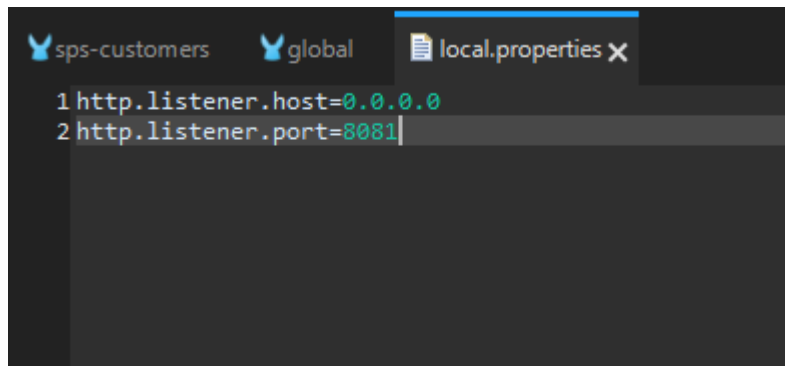
```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:http="http://www.mulesoft.org/schema/mule/http"
4     xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
5     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6     xsi:schemaLocation="
7     http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd http://www.mule
8
9     <http:listener-config name="HTTP Listener_SPS" doc:name="HTTP Listener config" doc:id="9546d4ce-036d-4268-b089-6eeb
10     <http:listener-connection host="0.0.0.0" port="8081" />
11 </http:listener-config>
12
13
14 </mule>
15
```

local.properties

A continuación, creé un archivo llamado `local.properties` dentro de la carpeta `src/main/resources` que es el directorio estándar donde proyectos basados en Maven buscan archivos de configuración y recursos.

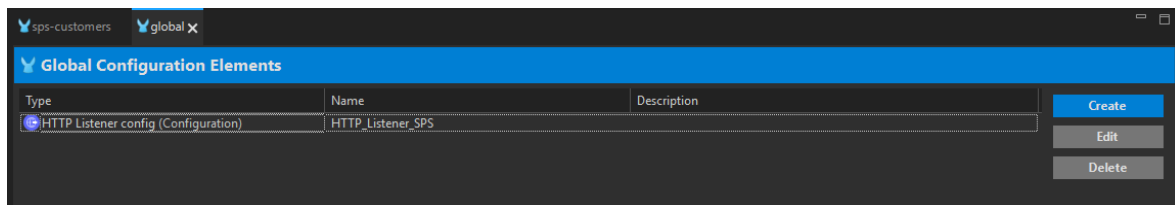


Y puse el host y puerto que es lo que deseo esconder.

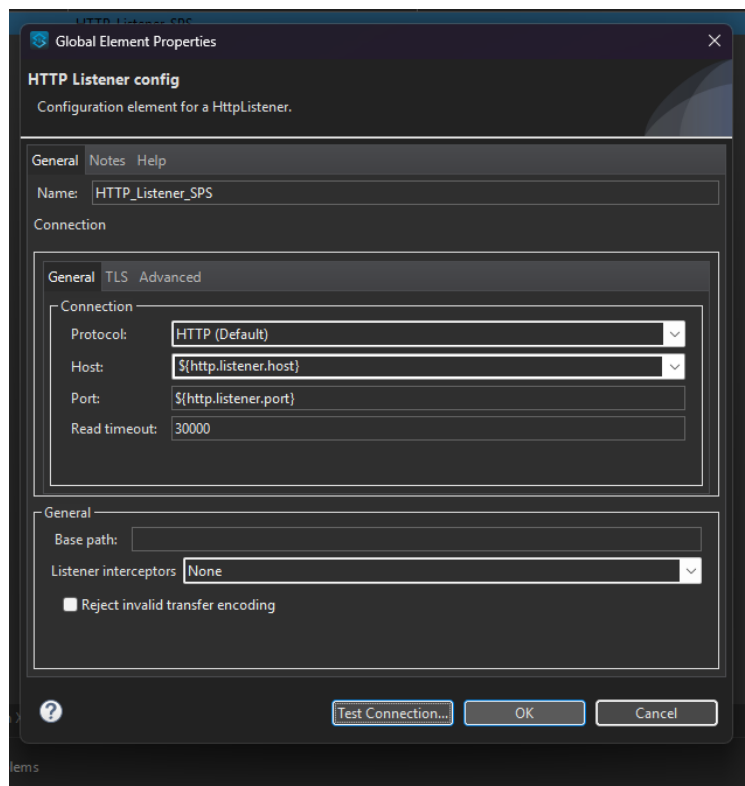


```
sps-customers  global  local.properties x
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
```

De manera visual, dentro del **Global Elements** del proyecto (accesible desde el panel inferior del IDE), es posible seleccionar la configuración del **HTTP Listener** y editar sus propiedades. Esta interfaz gráfica permite modificar valores como el `host` y el `port` sin necesidad de editar directamente el archivo XML.

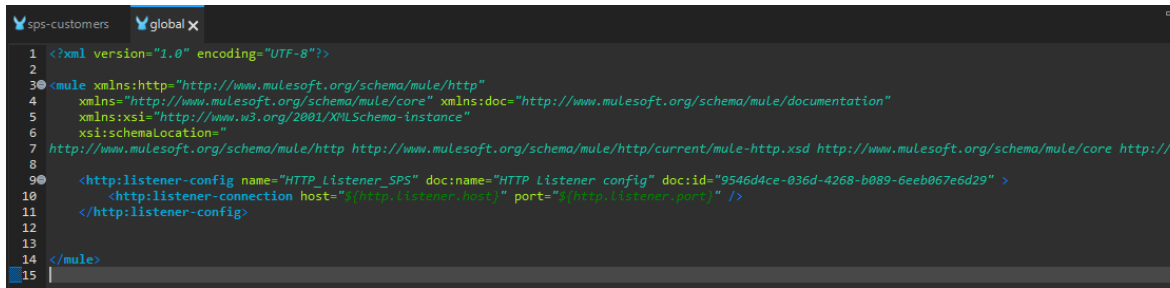


Luego, en el editor visual, reemplacé el valor del campo Host por `${http.listener.host}` y el de Port por `${http.listener.port}`.





Viéndose así en el XML

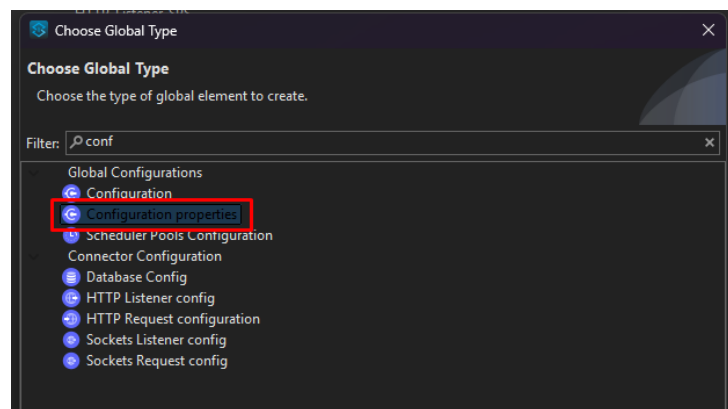


```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:http="http://www.mulesoft.org/schema/mule/http"
4     xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
5     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6     xsi:schemaLocation="
7     http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd http://www.mulesoft.org/schema/mule/core http://
8
9 <http:listener-config name="HTTP_Listener_SPS" doc:name="HTTP Listener config" doc:id="9546d4ce-036d-4268-b089-6eeb067e6d29" >
10   <http:listener-connection host="${http:listener-host}" port="${http:listener-port}" />
11 </http:listener-config>
12
13
14 </mule>
15

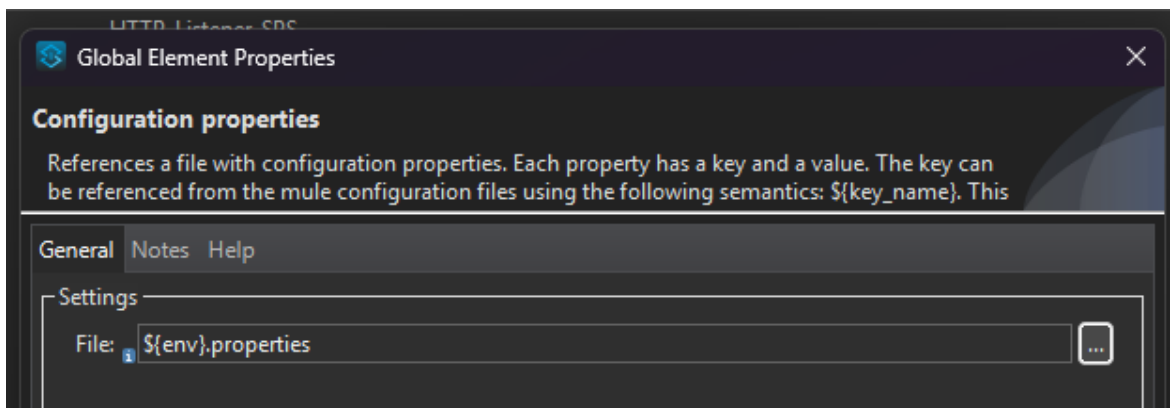
```

## Configuration properties



Para que el proyecto pueda reconocer y utilizar las variables definidas en los archivos .properties, es necesario añadir el componente Configuration Properties al archivo global.xml.

Este componente se encarga de leer los archivos de propiedades externos y cargar sus valores en tiempo de ejecución, de modo que puedan ser utilizados a lo largo de toda la aplicación mediante la sintaxis `${propiedad}`.



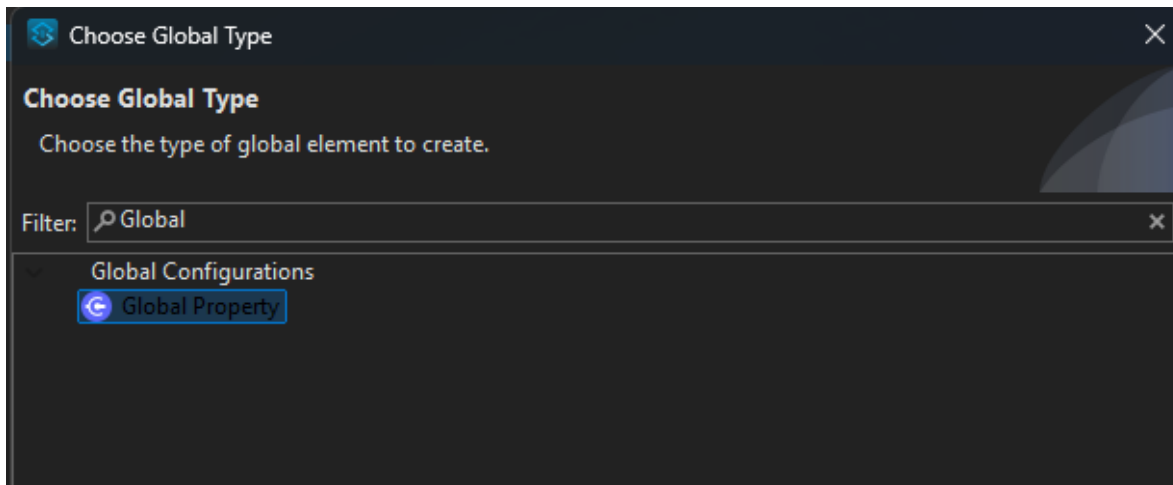
## Global Property

Desde el IDE de MuleSoft, accedí a la sección de **Global Elements** y, mediante la interfaz gráfica, añadí una nueva **Global Property**.

Definí la propiedad con:

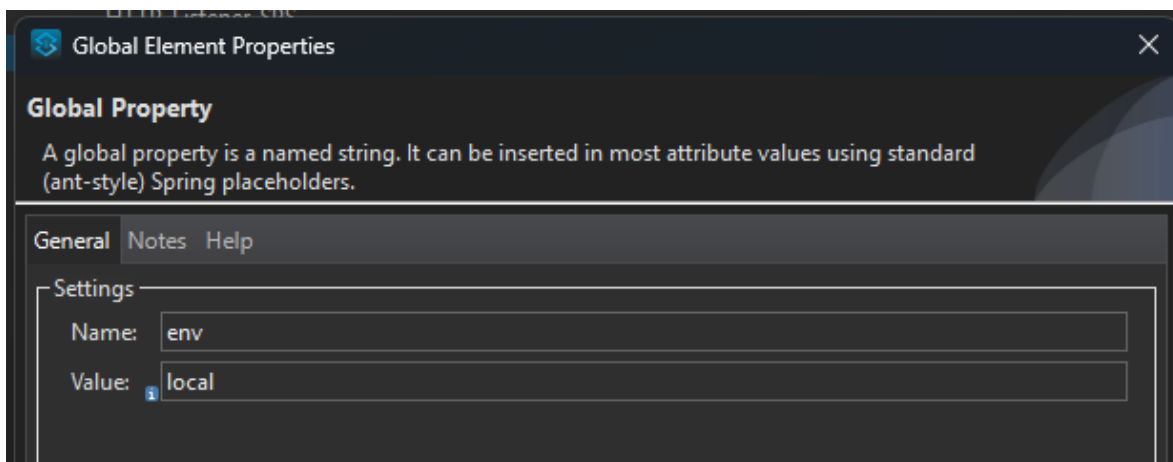
- **Name:** env
- **Value:** local

Esto genera internamente la siguiente declaración en el archivo global.xml:



*¿Por qué hacer esto?*

El objetivo fue contar con una referencia clara al entorno de ejecución actual. Esta propiedad puede utilizarse en diferentes puntos del flujo



Así es como va el archivo global.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:http="http://www.mulesoft.org/schema/mule/http"
4     xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
5     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6     xsi:schemaLocation="
7     http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd http://www.mulesoft.org/schema/mule/core http://
8
9 <http:listener-config name="HTTP_Listener_SPS" doc:name="HTTP listener config" doc:id="9546d4ce-036d-4268-b089-6eeb067e6d29" >
10 <http:listener-connection host="0.0.0.0" port="8080" >
11 </http:listener-connection>
12 </http:listener-config>
13 <configuration-properties doc:name="Configuration properties" doc:id="2d8b31f5-46a8-45eb-82bc-e2f01cf72aa8" file="global.properties" />
14 <global-property doc:name="Global Property" doc:id="14bd61c7-4473-4f30-9ffe-9dfbb1d9fde3" name="env" value="local" />
15
16
17
18 </mule>
19

```

## Secure Properties

Para proteger información sensible como el usuario y la contraseña de la base de datos, creé un archivo adicional llamado local.secure.properties, ubicado también en la carpeta src/main/resources. (También creé dev.secure.properties)

```

1 db.user=mule
2 db.password=mule

```

## Cifrado con Blowfish en modo CBC

Ejecuté el siguiente comando desde la terminal, utilizando el algoritmo **Blowfish** en modo **CBC**, una clave secreta personalizada (MyMuleSoftKey) y la cadena a cifrar.

```

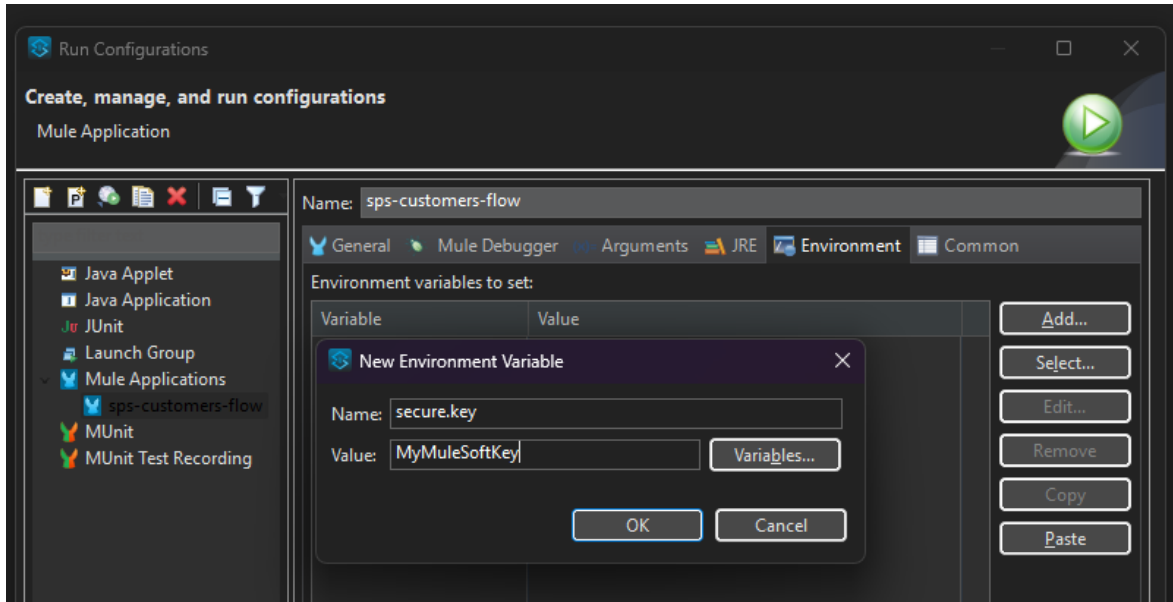
C:\Users\martc\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish CBC MyMuleSoftKey "mule"
hn5PdGLGVpw=

C:\Users\martc\Downloads>java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish CBC MyMuleSoftKey "mule"
hn5PdGLGVpw=

C:\Users\martc\Downloads>

```

Aunque MuleSoft permite definir directamente la clave de cifrado (encryptionKey) dentro del componente <secure-property-placeholder> en el archivo global.xml, **una práctica mucho más segura y profesional es pasar esta clave como parámetro externo** desde las **Run Configurations** del proyecto en Anypoint Studio.



#### Databaseconfig en xml con secure::

Para este punto, solo faltaba finalizar la configuración de la conexión a la base de datos. Ya había definido en el global.xml los dos componentes Configuration Properties necesarios, apuntando a los archivos local.properties y local.secure.properties, respectivamente.

El archivo local.secure.properties contenía las credenciales cifradas de acceso a la base de datos:

```
local.secure.properties
1 db.user=[hn5PdGLGVPw=]
2 db.password=[hn5PdGLGVPw=]
```

Dejando el resto de propiedades de la base de datos en local.properties

```
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
3
4 # data base
5 db.host=mudb.learn.mulesoft.com
6 db.port=3306
7 db.database=training
8
```

Para este punto, me pregunté algo importante.

Si bien, ya mi global.xml tiene especificados archivos de configuración que son los siguientes.

```
<configuration-properties doc:name="Configuration Properties" doc:id="2d8b31f5-46a8-45eb-82bc-e2f01cf72aa8" file="global.properties" />
<secure-properties:config name="Secure Properties Config" doc:name="Secure Properties Config" doc:id="79396c45-9642-4360-ad8a-49cc33300f5d" file="global.secure.properties" key="{secure.key}" >
  <secure-properties:encrypt algorithm="Blowfish" />
</secure-properties:config>
```

¿Cómo le indico a la etiqueta de configuración de base de datos que valores como db.host, db.database vienen del local.properties y el db.user con db.password vienen de local.secure.properties?

```
<db:config name="Database Config" doc:name="Database Config" doc:id="578d2319-d85e-48c2-aa47-bb438278f94b" >
  <db:mysql-connection host="{db.host}" port="{db.port}" user="{secure:db.user}" password="{secure:db.password}" database="{db.database}" />
</db:config>
```

Fue entonces cuando descubrí que MuleSoft utiliza el prefijo especial **secure::** para referenciar propiedades que provienen de archivos de propiedades seguras (secure properties).

*Acudí a la documentación para leer más sobre “secure::” en el siguiente recurso*

<https://docs.mulesoft.com/mule-runtime/latest/secure-configuration-properties>

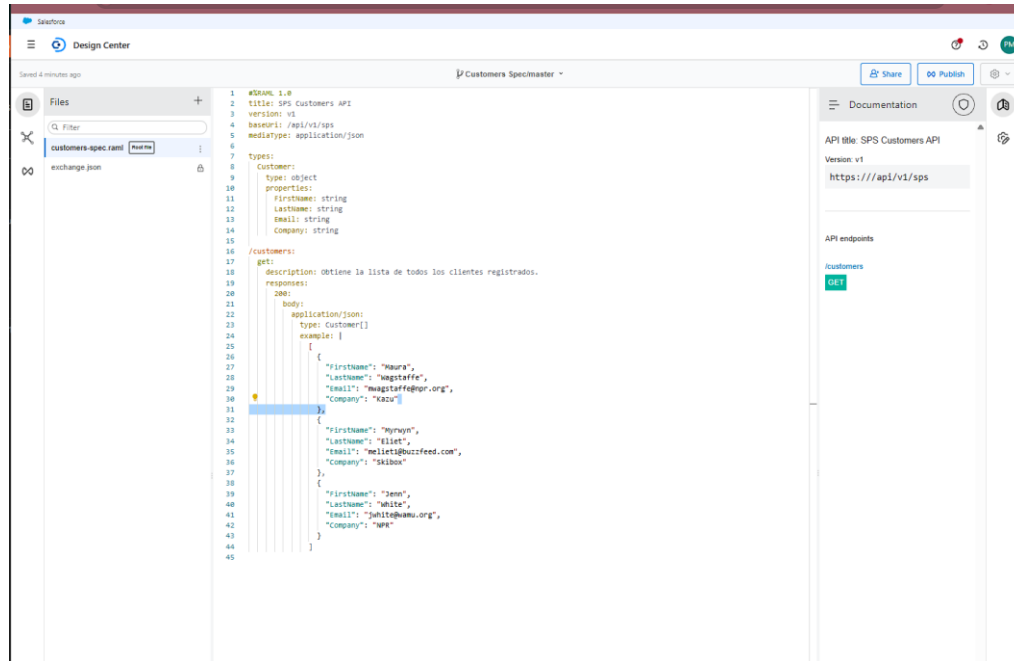
The screenshot shows the MuleSoft documentation on the left and the Salesforce Config interface on the right. The documentation page is titled "Secure Configuration Properties" and lists various security-related topics. The Salesforce Config interface shows the "Advanced" tab with fields for Username, Password, Security token, and Authorization URL. The Username field contains the expression `{secure:sfdc.username}` and the Password field contains `{secure:sfdc.password}`. A red box highlights the `secure::` prefix in the documentation text, stating that it is added before the property name definition to enable access to all values inside a secure properties file, even if the values are not encrypted. Another red box highlights the "Test Connection" button in the interface.

The `secure::` prefix is added before the property name definition, to enable access to all values inside a secure properties file, even if the values are not encrypted.

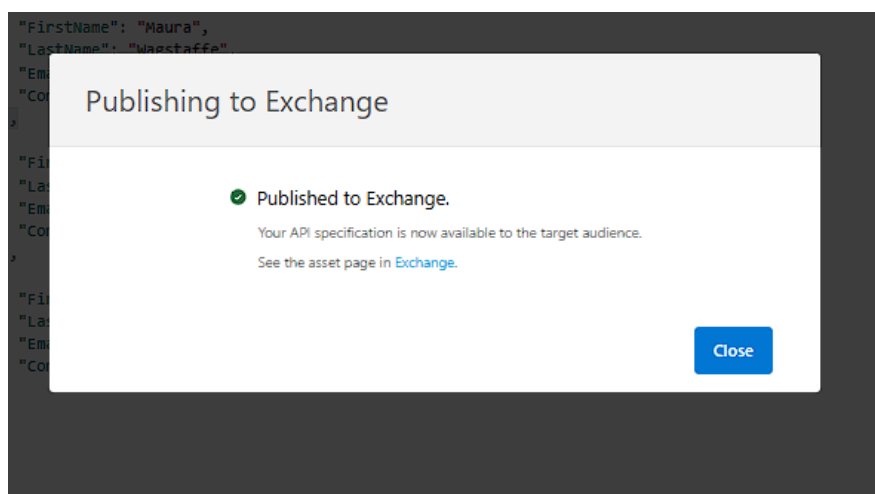
You can click **Test Connection** to verify that the connection is functioning as expected and assure that the secure properties are read successfully.

## Especificación de API

En el Design Center de MuleSoft diseñé la especificación de la API para exponer la información de los clientes en el endpoint `/api/v1/sps/customers`. Esta especificación define el contrato de la API, indicando qué datos devuelve, como nombre, apellido, correo electrónico y empresa.



La razón de crear esta especificación es asegurar que tanto el equipo de desarrollo como los consumidores de la API tengan una referencia clara y precisa de cómo debe comportarse el servicio. Finalmente, publiqué la especificación en Anypoint Exchange.



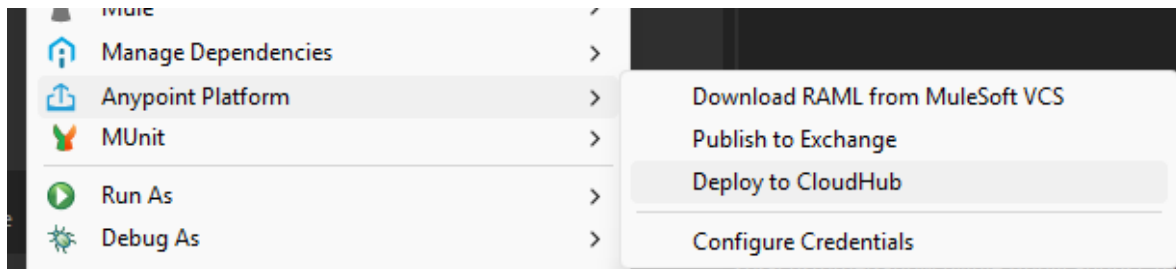
## Cloudhub

Una vez finalizado y probado el flujo localmente, procedí a realizar el despliegue de la aplicación en CloudHub, la plataforma de ejecución en la nube de MuleSoft.

Para ello, utilicé Anypoint Studio, que permite desplegar proyectos directamente a CloudHub mediante la integración con Anypoint Platform.

Desde el entorno de desarrollo, hice clic derecho sobre el nombre del proyecto en el panel de navegación y seleccioné la opción:

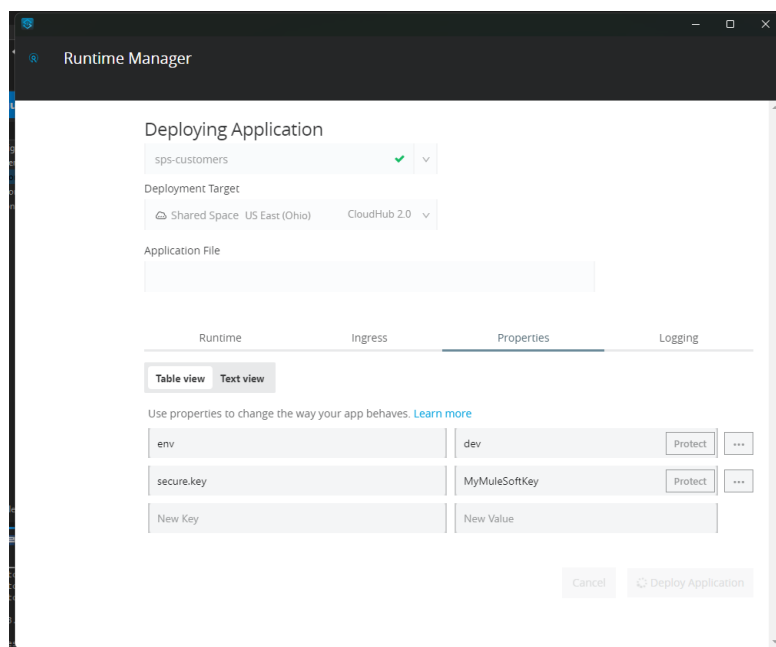
Anypoint Platform > Deploy to CloudHub



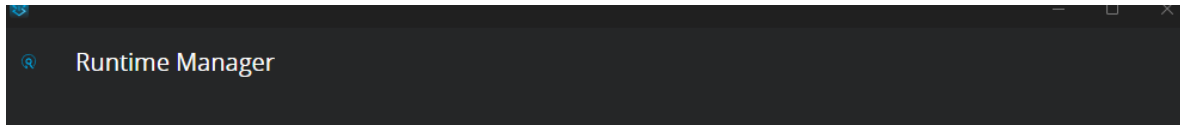
Esto abrió una ventana de configuración donde pude:

Definimos que para el despliegue usará el **env DEV**

Incluir los parámetros de entorno como la clave de descifrado (secure.key)



Tómo unos momentos



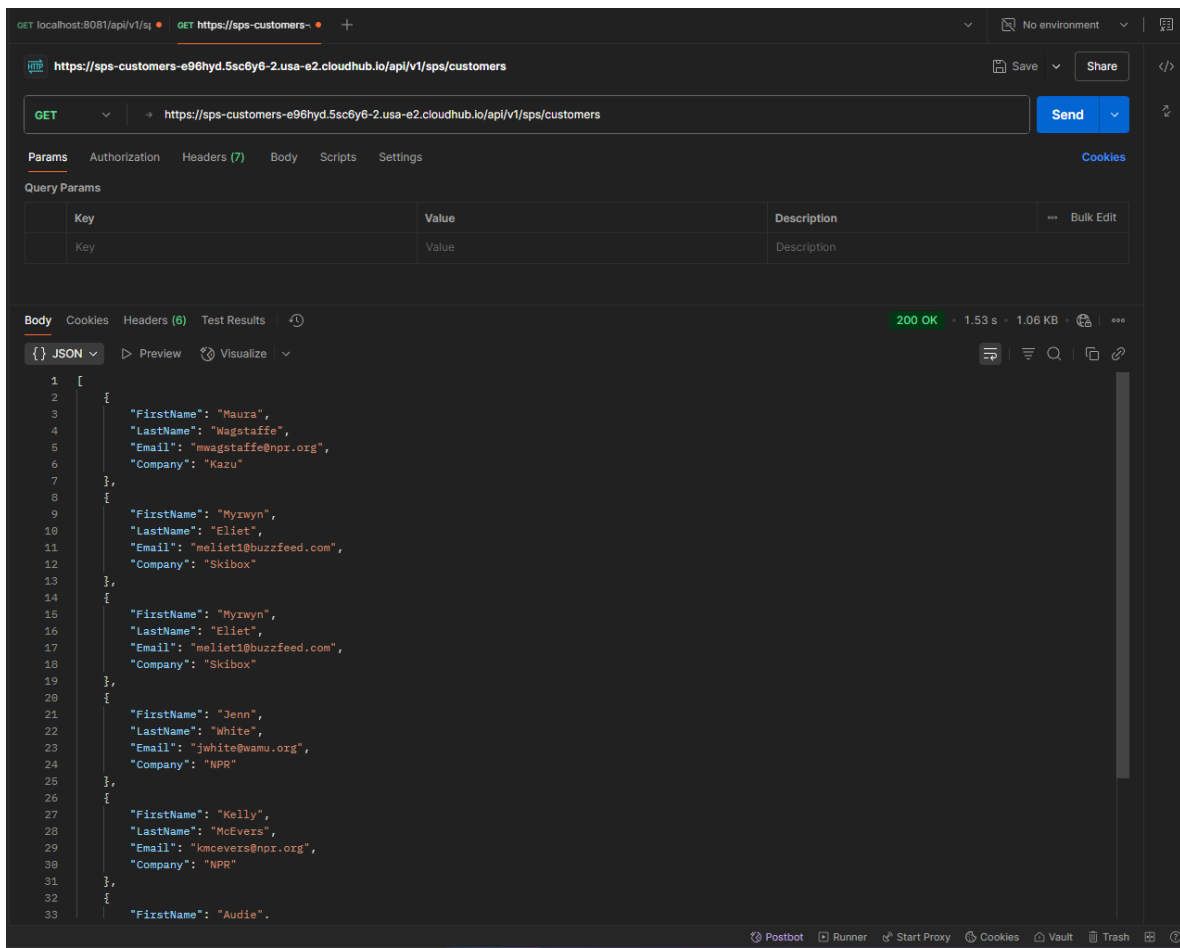
## Deploying sps-customers to Cloudhub-US-East-2

You may close this window at any time.

Open in Browser

Close Window

Una vez completado el despliegue en CloudHub, copié la URL pública generada para la aplicación y la probé en postman:

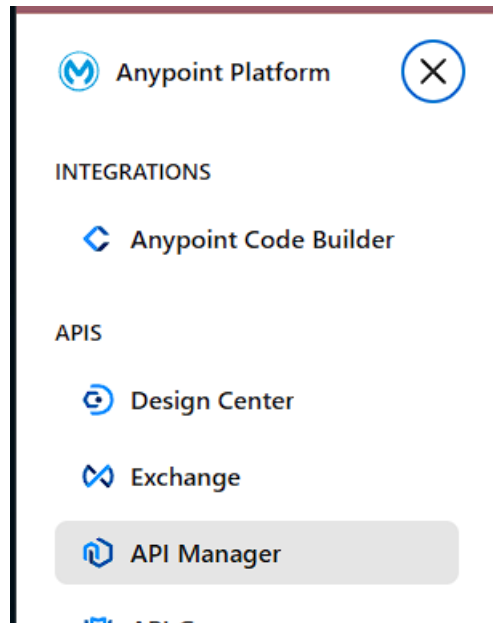




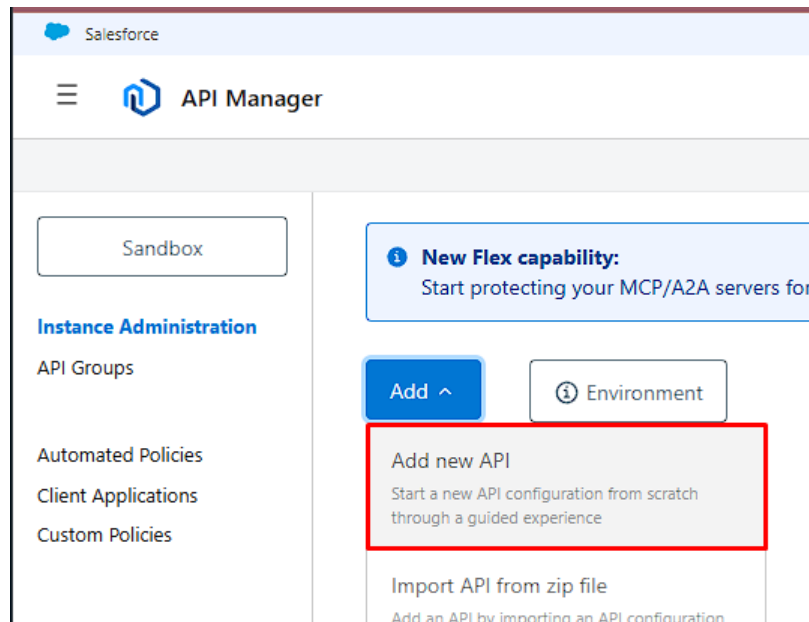
## EXTRA 1: Crear un API en API Manager

### Abrir API Manager

Inicié sesión en la plataforma y me dirigí a API Manager, el módulo responsable de gestionar el ciclo de vida completo de las APIs, incluyendo su registro, control de acceso, versionamiento, monitoreo, aplicación de políticas de seguridad y gobernanza.



Dentro del entorno **Sandbox** (entorno no productivo que se utiliza para pruebas, validaciones o configuraciones previas a producción), seleccioné la opción:



## Selección de Runtime: ¿Flex Gateway o Mule Gateway?

En esta etapa, MuleSoft me ofreció dos opciones:

- Flex Gateway
- Mule Gateway

Esto me llevó a formular algunas preguntas técnicas clave para elegir correctamente:

- ¿Cuál es la diferencia entre Flex Gateway y Mule Gateway?
- ¿En qué casos conviene usar Flex Gateway en lugar de Mule Gateway?

Busqué la documentación y encontré lo siguiente:

- Mule Gateway es ideal cuando la API ya está implementada en CloudHub, Runtime Fabric, o on-premise usando el runtime de Mule. Es el enfoque tradicional, estrechamente integrado con MuleSoft.
- Flex Gateway es un gateway más liviano, independiente del runtime de Mule, y pensado para escenarios de microservicios, Kubernetes, o infraestructura híbrida. Ofrece mayor flexibilidad y rendimiento para APIs implementadas fuera del ecosistema MuleSoft.

<https://docs.mulesoft.com/gateway-home/>

Dado que mi API estaba desplegada en CloudHub, seleccioné Mule Gateway, que es el más adecuado en este escenario por su integración nativa con CloudHub y el runtime Mule.

The screenshot shows the 'Add API' configuration page in the MuleSoft API Manager. The 'Runtime' section is active, and the 'Mule Gateway' option is selected. The 'Proxy type' section has 'Connect to existing application (basic endpoint)' selected. The 'Mule version' section has 'Mule 4 (recommended)' selected. The 'Next' button is visible at the bottom right.

## Selección del tipo de API (Asset Type) a gestionar

En la sección API, la plataforma me solicitó seleccionar el Asset type, es decir, el tipo de recurso que deseaba administrar desde API Manager.

The screenshot shows the Salesforce API Manager interface. The breadcrumb navigation indicates the path: API Administration (Sandbox) > Add API. The 'API' tab is selected in the top navigation bar. The main content area is titled 'Instance Administration / Add API'. It contains a form for adding a new API. The form has two main sections. The first section, 'Select the API you want to manage.', has a radio button for 'Select API from Exchange' and a 'Create new API' button, which is highlighted with a red box. Below this is a note: 'Once the API is created it will be published in Exchange in stable state.' The second section contains a 'Name' field with the value 'spscustomers' and an 'Asset types' dropdown menu with 'HTTP API' selected, both highlighted with a red box. At the bottom of the form are 'Cancel', 'Previous', and 'Next' buttons.

## Downstream

En la sección llamada Downstream, me pidió configurar algunos ajustes relacionados con el tráfico que llega a la API, es decir, cómo se reciben y manejan las solicitudes de los usuarios.

Por defecto, la opción llamada Client Provider estaba seleccionada como Anypoint y no me permitía cambiarla. Esto quiere decir que la plataforma Anypoint se encarga de controlar quién puede usar la API, gestionando la seguridad y el acceso.

## Instance Administration / Add API

Runtime API Downstream Upstream Review

## Downstream

Configure the API instance settings related to inbound traffic.

\* Required field

Client provider \*

Anypoint

Instance label ⓘ  
(Optional)

Recommended if you have multiple managed instances of the same API

Advanced options >

Cancel

Previous

Next

## Upstream

En cambio, en la sección Upstream no se me solicitó configurar nada obligatorio. Sin embargo, la plataforma indicaba que esta configuración define cómo fluye el tráfico hacia los servicios upstream

## Instance Administration / Add API

Runtime API Downstream Upstream Review

## Upstream

Define how the traffic flows to upstream services.

\* Required field

Upstream URL  
(Optional)

https://

Cancel

Previous

Next

## API Registrada exitosamente

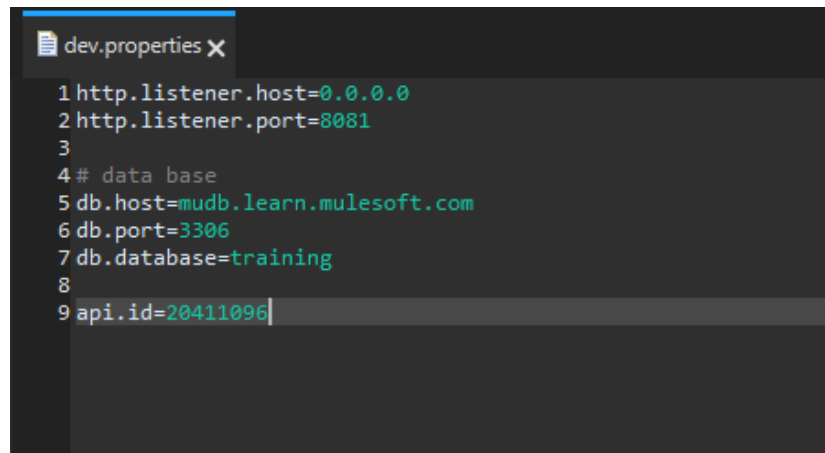
**New Flex capability:**  
Start protecting your MCP/A2A servers for Agent to API, Agent to Agent communications. Protect MCP server or A2A server

Status	Name	Runtime	Label	Version	Instance	Error Rate ⓘ	Total Requests ⓘ	Client Applications	Creation Date	Actions
Unregistered	spscustomersapi	Mule 4	-	v1	20411096	No data	No data	0	06-25-2025 07:01	

## EXTRA 2: API Autodiscovery

Esta es una funcionalidad de MuleSoft que permite que una aplicación Mule se registre automáticamente en API Manager al iniciar. Esto facilita que el runtime y la plataforma estén sincronizados, permitiendo aplicar políticas, monitorear tráfico y gestionar la API de forma centralizada sin necesidad de configurar manualmente detalles en el código.

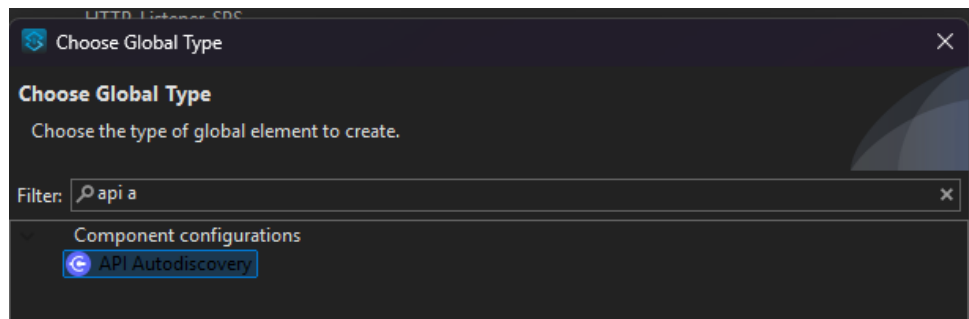
Para habilitar esta funcionalidad, es necesario identificar el api.id asignado a la API en API Manager, que es un identificador único de la instancia.

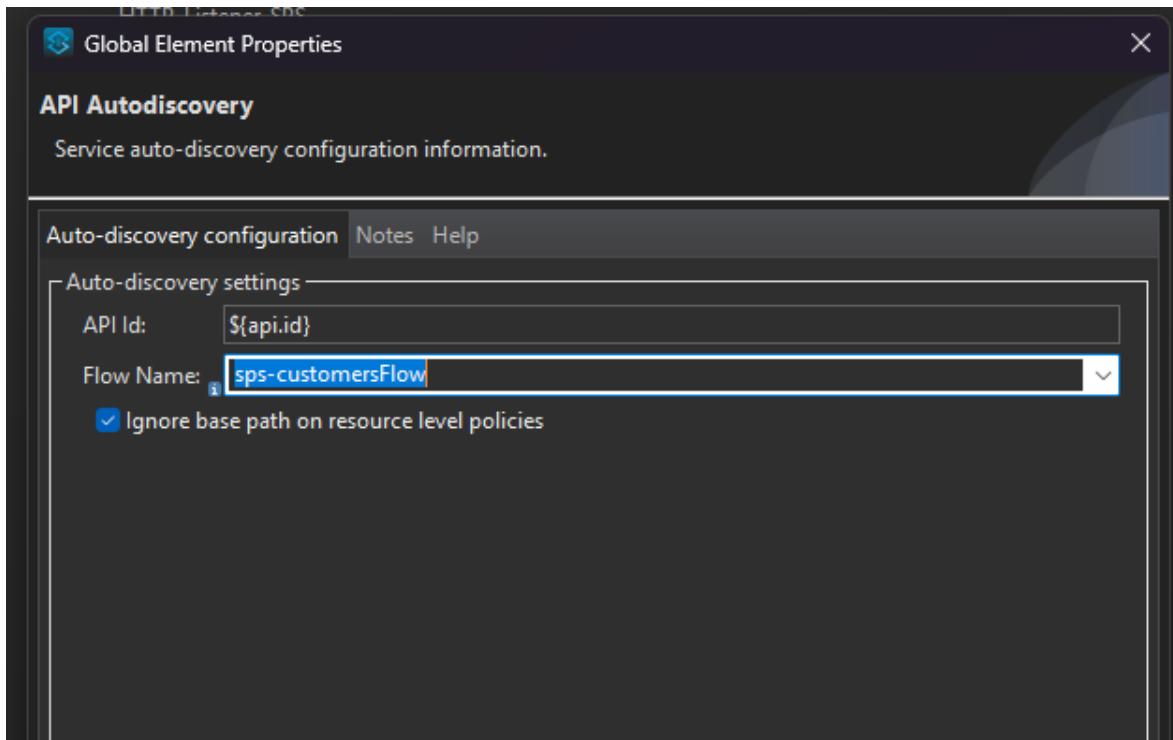


```
dev.properties X
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
3
4 # data base
5 db.host=mudb.learn.mulesoft.com
6 db.port=3306
7 db.database=training
8
9 api.id=20411096
```

Dentro del archivo de configuración global.xml, accedí a la sección de Global Elements para agregar el componente de API Autodiscovery.

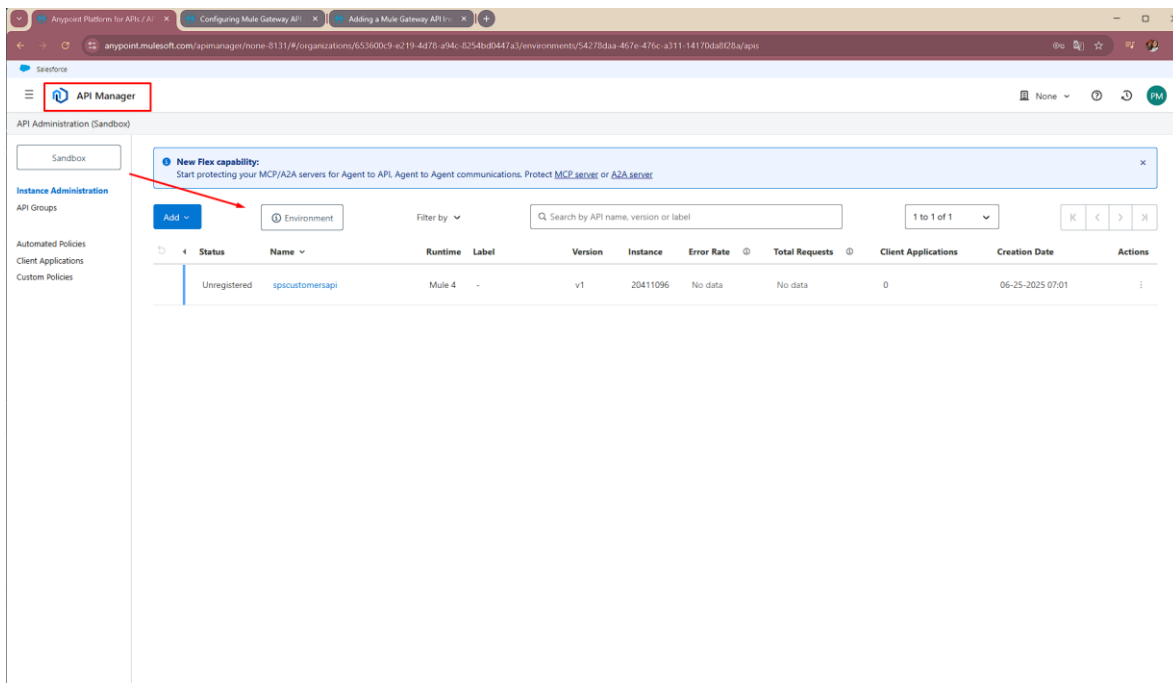
Esto implica crear una instancia global que conecta el flujo Mule con API Manager usando el api.id previamente configurado en el archivo dev.properties.





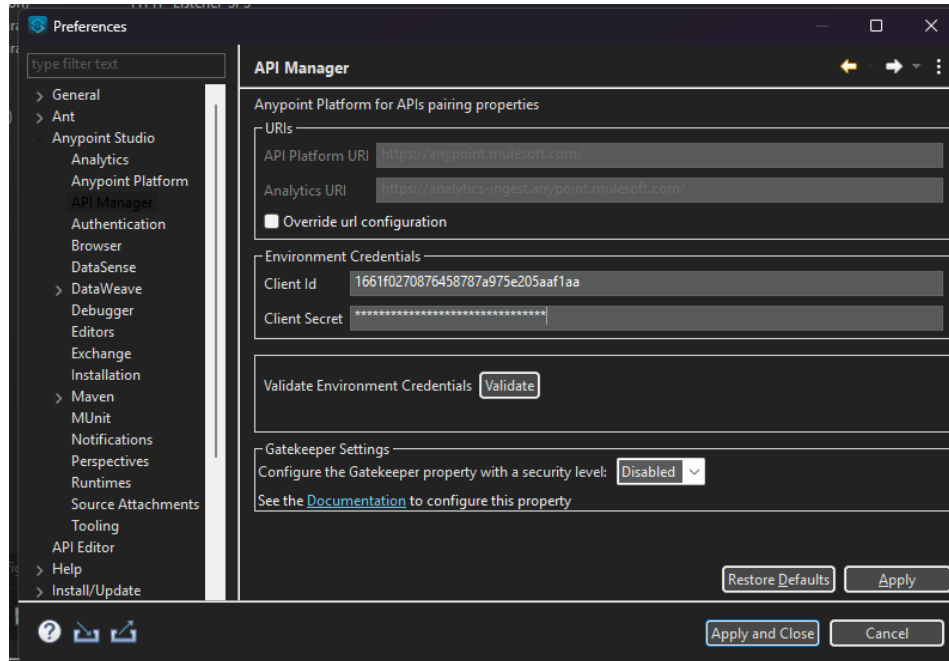
### Obtener Client ID y Client Secret

Una vez habilitado el API Autodiscovery, el siguiente paso fue configurar el acceso seguro a la API mediante credenciales. Para ello, fue necesario obtener el Client ID y el Client Secret asignados automáticamente por Anypoint Platform al crear la instancia de la API en el entorno Sandbox.

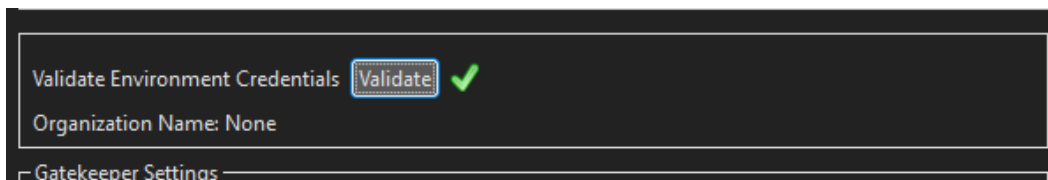


Luego, accedí a las preferencias del proyecto en Anypoint Studio, específicamente a la sección de API Manager, donde es posible vincular localmente la configuración de Autodiscovery.

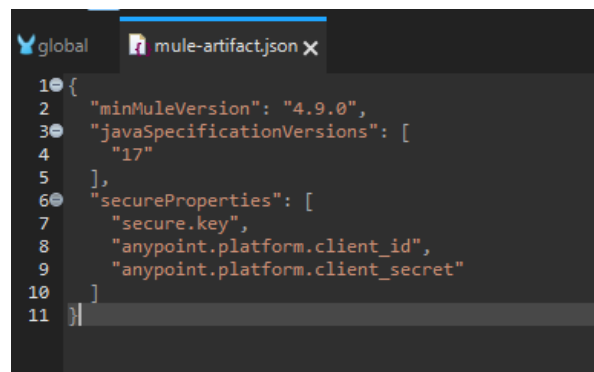
Allí, agregué lo anterior descrito, para que la aplicación se registre correctamente en API Manager y pueda ser gestionada desde la plataforma:



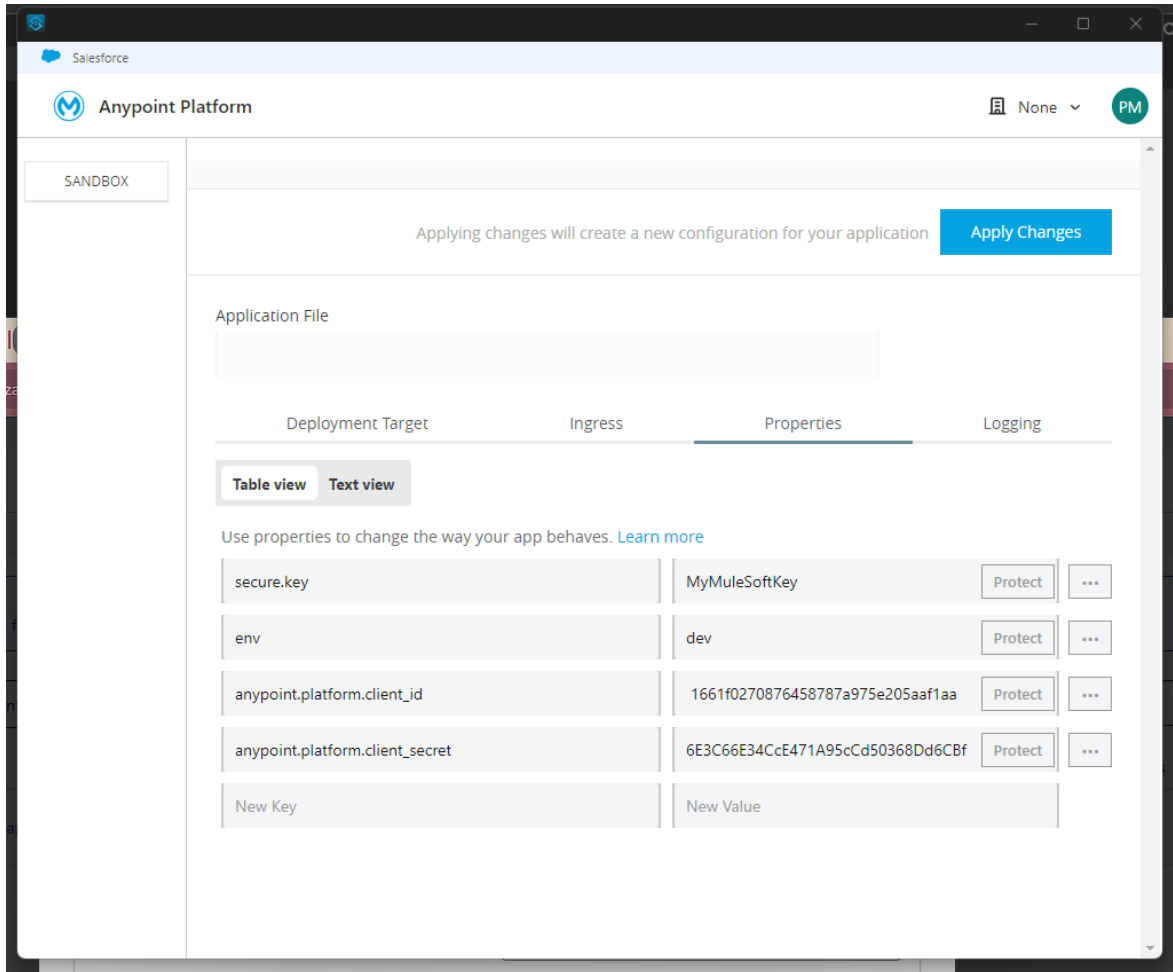
Podemos validar que la conexión sea exitosa.



Después accedí al archivo `mule-artifact.json`, ubicado en la raíz del proyecto Mule, y sirve para definir la configuración general del despliegue y permitir que el runtime de Mule pueda autenticarse contra Anypoint Platform.



Para poder realizar el despliegue de la aplicación Mule en CloudHub, es necesario incluir las propiedades `anypoint.platform.client_id` y `anypoint.platform.client_secret`. Para que se autentique el proyecto de Mule con Anypoint Platform.



### Explorando Logs

Cuando terminé de agregar las propiedades necesarias y realicé el despliegue, noté que el proceso estaba tardando más de lo habitual. Ante esto, decidí revisar los logs de la aplicación para entender qué estaba ocurriendo.

Al revisar la consola, encontré un mensaje de error que indicaba un problema de autenticación con Anypoint Platform. Esto me llevó a descubrir que había ingresado de forma incorrecta el valor del Client Secret.

Este incidente se convirtió en un aprendizaje muy valioso, ya que me permitió comprobar la importancia de validar los mensajes del log para detectar errores rápidamente y entender cómo



responder ante fallos durante el despliegue.

● sps-customers

Config 8e533b (Desired)  Time range Log Levels (5/5)

Last deployed 7 minutes ago - 2025-06-25 13:40 CST

— \$main - Two modules with namespace 'Batch' were found in a binding context. Set log level to DEBUG for details

Info 6 minutes ago - 2025-06-25 13:42:08.560 CST - TransactionJournal \$main - Using files for tx logs /opt/mule/.mule/analytics-policy-20411096 @ sps-customers-sps-customersFlow/queue-xa-tx-log/tx1.log and /opt/mule/.mule/analytics-policy-20411096 @ sps-customers-sps-customersFlow/queue-xa-tx-log/tx2.log Replica l8x6m

Info 6 minutes ago - 2025-06-25 13:42:08.560 CST - TransactionJournal \$main - Using files for tx logs /opt/mule/.mule/analytics-policy-20411096 @ sps-customers-sps-customersFlow/queue-tx-log/tx1.log and /opt/mule/.mule/analytics-policy-20411096 @ sps-customers-sps-customersFlow/queue-tx-log/tx2.log Replica l8x6m

Info 6 minutes ago - 2025-06-25 13:42:08.656 CST - AbstractLifecycleManager \$main - Initialising Bean: proxy-policy-7aef0aa0-51fc-11f0-a2e0-2e73467132ee Replica l8x6m

Info 6 minutes ago - 2025-06-25 13:42:08.659 CST - QueueXaResourceManager \$main - Starting ResourceManager Replica l8x6m

Info 6 minutes ago - 2025-06-25 13:42:08.659 CST - QueueXaResourceManager \$main - Started ResourceManager Replica l8x6m

Info 6 minutes ago - 2025-06-25 13:42:08.661 CST - AbstractLifecycleManager \$main - Starting Bean: proxy-policy-7aef0aa0-51fc-11f0-a2e0-2e73467132ee Replica l8x6m

Error 5 minutes ago - 2025-06-25 13:42:17.400 CST - ApiPlatformClientProvider \$gw-api-platform-connection-retry.01 - Client ID and Client Secret could not be validated against API Manager. Replica l8x6m

Error 5 minutes ago - 2025-06-25 13:42:17.400 CST - ApiPlatformClientProvider \$gw-api-platform-connection-retry.01 - Failed to connect with API Manager. This request will be retried after some backoff time. Reason: com.mulesoft.mule.runtime.gw.client.exception.UnauthorizedException: Authorization request to Anypoint Platform was not successful, client\_id and/or client\_secret may be wrong. Status code: 401 Server Payload: Unauthorized Replica l8x6m

Error 4 minutes ago - 2025-06-25 13:43:29.463 CST - ApiPlatformClientProvider \$gw-api-platform-connection-retry.01 - Client ID and Client Secret could not be validated against API Manager. Replica l8x6m

Error 4 minutes ago - 2025-06-25 13:43:29.464 CST - ApiPlatformClientProvider \$gw-api-platform-connection-retry.01 - Failed to connect with API Manager. This request will be retried after some backoff time. Reason: com.mulesoft.mule.runtime.gw.client.exception.UnauthorizedException: Authorization request to Anypoint Platform was not successful, client\_id and/or client\_secret may be wrong. Status code: 401 Server Payload: Unauthorized Replica l8x6m

## Status Active

El Estado Activo indica que el **API Autodiscovery** está funcionando correctamente, y que la API ya puede recibir políticas, ser monitoreada, y estar disponible para su consumo bajo control.

Anyoint Management Center Anyoint Platform for APIs / API

anypoint.mulesoft.com/apimanager/none-8131/#/organizations/653600c9-e219-4d78-a94c-8254bd0447a3/environments/54278daa-467e-476c-a311-14170da8f28a/apis

Salesforce

API Manager

API Administration (Sandbox)

Sandbox

Instance Administration

API Groups

Automated Policies

Client Applications

Custom Policies

**New Flex capability:**  
Start protecting your MCP/A2A servers for Agent to API, Agent to Agent communications. Protect [MCP server](#) or [A2A server](#)

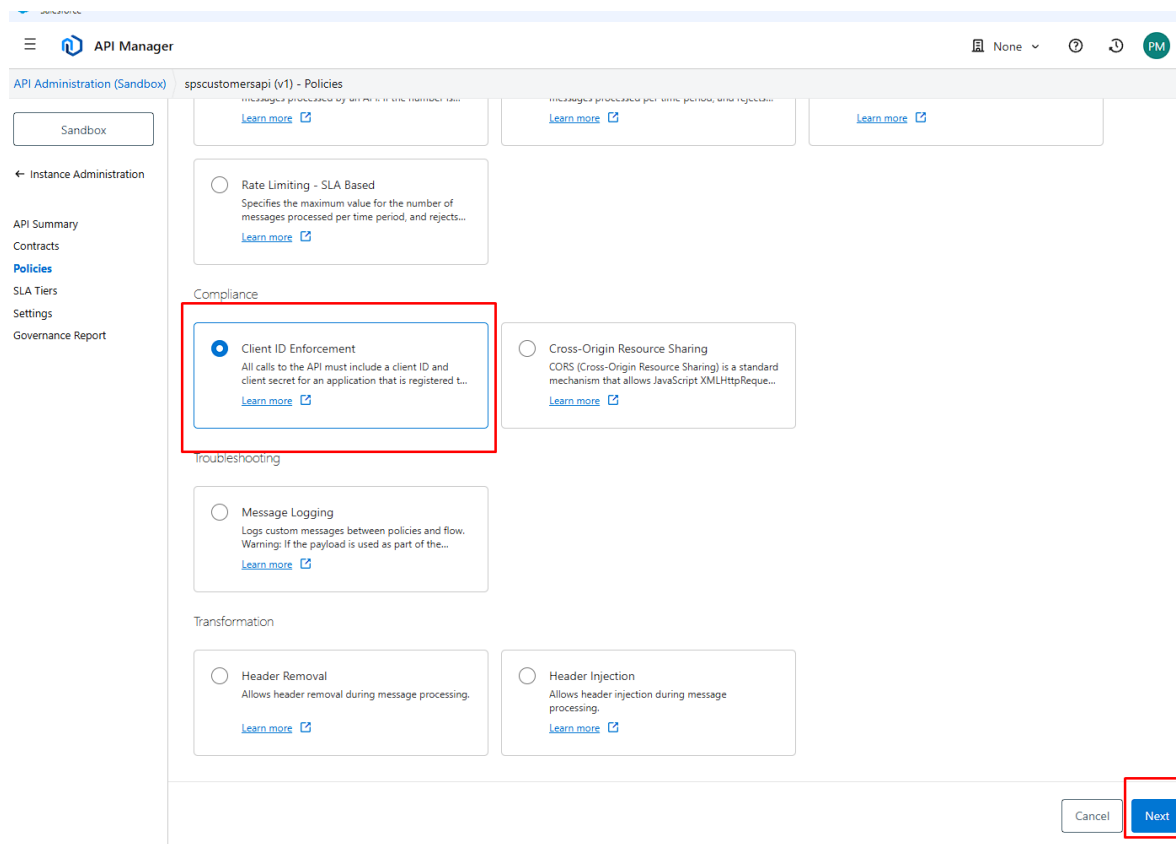
Add Environment Filter by Search by API name, version or label 1 to 1 of 1

Status	Name	Runtime	Label	Version	Instance	Error Rate	Total Requests	Client Applications	Cr
Active	spscustomersapi	Mule 4	-	v1	20411096	No data	No data	0	0

## EXTRA 3: Aplicar la política de client-id Enforcement para proteger nuestra API.

La política Client ID Enforcement es una medida de seguridad que se aplica a una API para asegurarse de que solo puedan acceder a ella los usuarios autorizados. Cuando esta política está activa, la API exige que cada solicitud incluya un Client ID y un Client Secret, que son como una llave y contraseña que identifican a cada aplicación que quiere consumir la API. Si alguien intenta usar la API sin estas credenciales, o con unas incorrectas, la plataforma rechaza la solicitud automáticamente.

### Agregar Policy



### ¿HTTP Basic Authentication Header o Custom Expression?

Al configurar la política Client ID Enforcement, se presentaron dos opciones principales para definir el origen de las credenciales:

Credential Origin, donde debía elegir entre:

- HTTP Basic Authentication Header
- Custom Expression

La opción HTTP Basic Authentication Header es la más común y recomendada cuando las credenciales Client ID y Client Secret se envían directamente en el encabezado de la solicitud usando el estándar de autenticación básica HTTP. Esto facilita la validación automática de las credenciales sin necesidad de configuraciones adicionales.

Por otro lado, la opción Custom Expression permite definir una expresión personalizada para extraer las credenciales de otra parte de la solicitud, como puede ser un parámetro en la URL, un campo en el cuerpo, o un encabezado diferente. Esta opción brinda mayor flexibilidad, pero requiere conocimientos más avanzados para configurar correctamente la expresión y asegurar que las credenciales sean extraídas de manera segura.

Elegí la opción **HTTP Basic Authentication Header** porque es un estándar ampliamente utilizado para enviar credenciales de forma segura y sencilla. Esta opción permite que el Client ID y el Client Secret se envíen en el encabezado de la solicitud utilizando un método que es compatible con la mayoría de las herramientas de prueba como Postman.

Instance [spscustomersapi](#) / Policies / Configure Client ID Enforcement policy

Administration /

Credentials origin  
Origin of the Client ID and Client Secret credentials.

☒ HTTP Basic Authentication Header

☐ Custom Expression

---

Advanced options ▾  
Configure policy version, methods and resources

Policy version \*

1.3.2 (latest) ▾ ⓘ

Method & resource conditions

☒ Apply configuration to all API method & resources

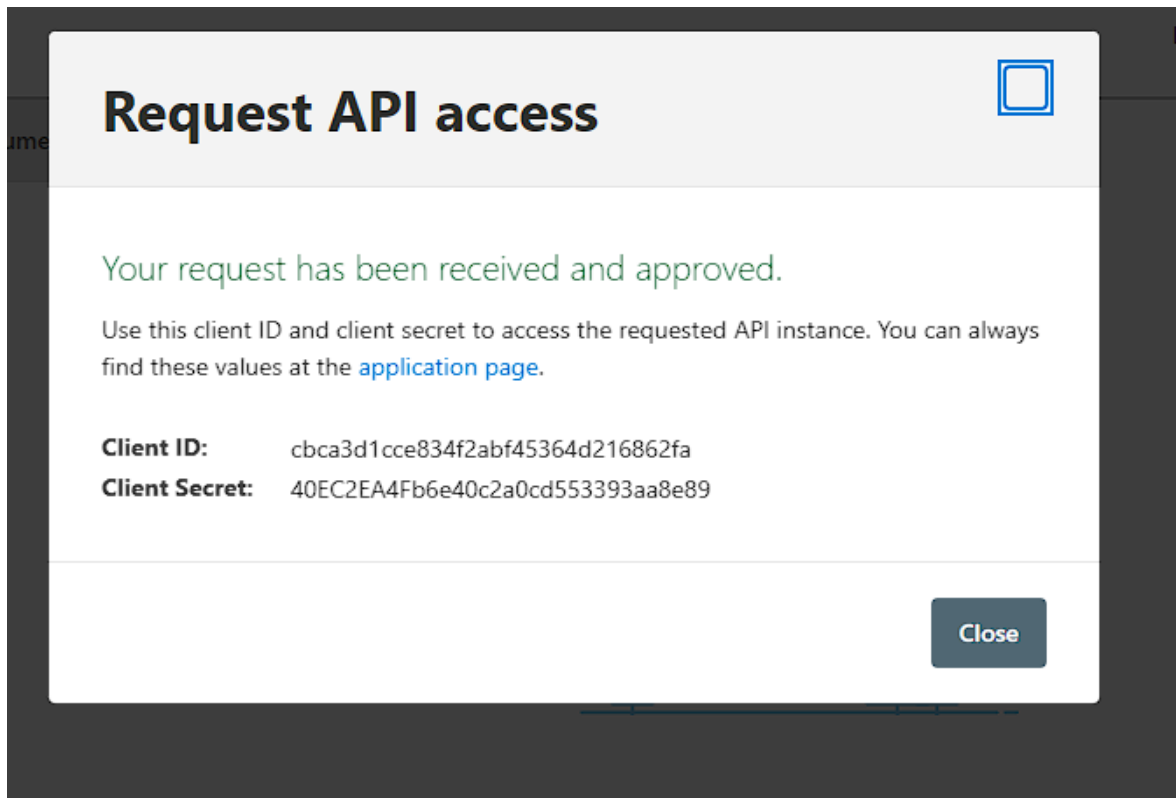
☐ Apply configuration to specific API method & resources

Previous Apply

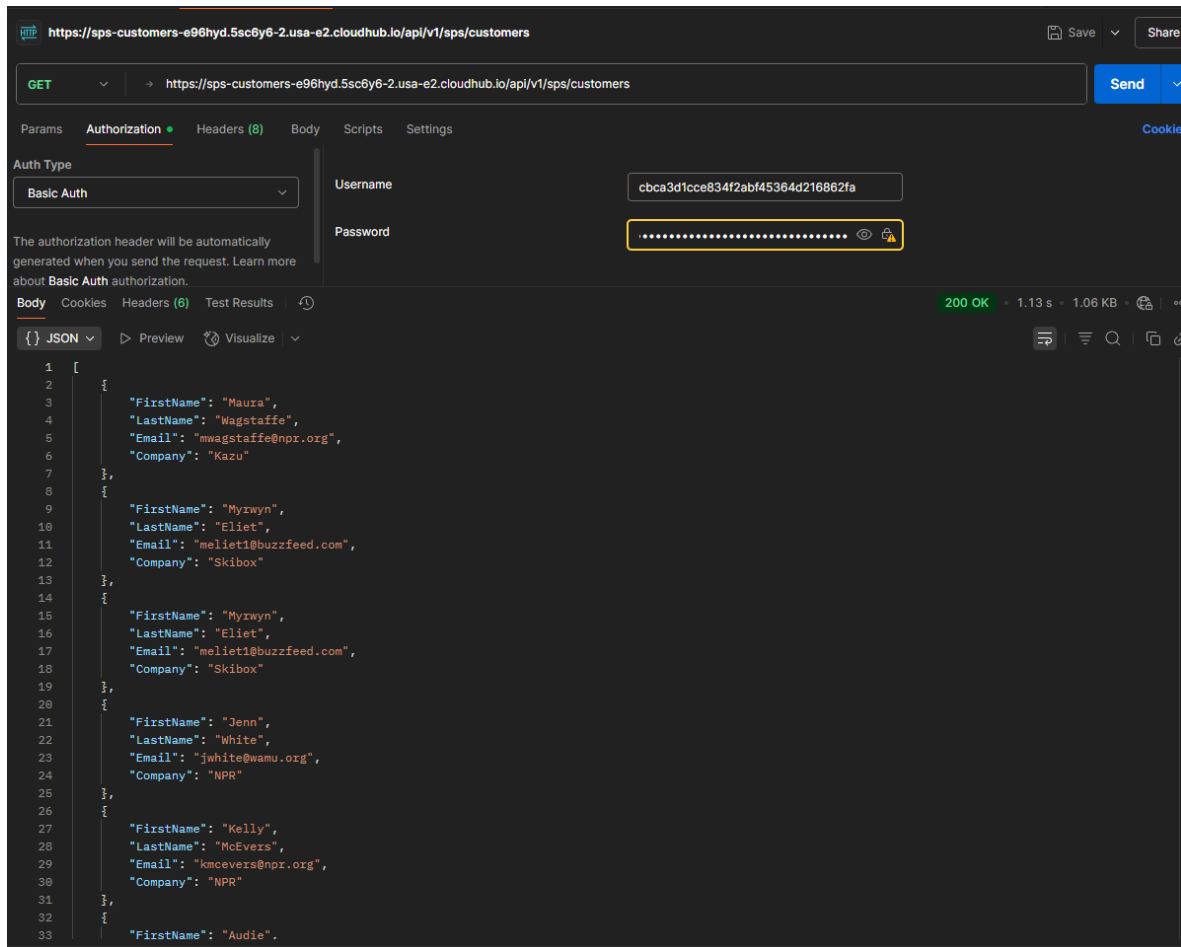
En la configuración de la política también apareció la opción "Method and Resource Condition", que permite decidir si la política se aplica a:

- Todos los métodos y recursos de la API, es decir, a todas las rutas y tipos de solicitud (GET, POST, etc.).
- Métodos y recursos específicos, permitiendo una aplicación más granular de la política solo en ciertas rutas o tipos de métodos.

Finalmente nos regresa:



## Imagen del resultado obtenido de toda la práctica



## ADICIONAL EXTRA: JWT VALIDATION

Por curiosidad y con el fin de profundizar en las posibilidades de seguridad que ofrece MuleSoft, decidí explorar cómo implementar otras validaciones, como la autenticación mediante JWT (JSON Web Token).

Partiendo del planteamiento del proyecto SPS, donde el equipo de marketing necesita acceder a la información de nuestros clientes para generar campañas personalizadas y mejorar la experiencia del usuario, consideré importante que solo ciertos roles específicos puedan acceder a esos datos sensibles.

Por ello, decidí incluir un token JWT que contuviera una claim de rol, de modo que únicamente los usuarios con roles de admin o marketing pudieran realizar la consulta. Además, implementé la validación del tiempo de vida del token para asegurar que solo los tokens vigentes fueran aceptados.

Este paso adicional me permitió entender mejor cómo proteger datos sensibles de acuerdo con los perfiles de acceso, garantizando seguridad y cumplimiento con las necesidades del negocio.

Instance Administration / spscustomersapi / Policies / Add a policy

Search by policy name

Browse by category

ALL CATEGORIES (19) SECURITY (10) QUALITY OF SERVICE (4) COMPLIANCE (2) TROUBLESHOOTING (1) TRANSFORMATION (2)

Security

☒ **JWT Validation**  
All calls to the API must include a Json Web Token (JWT) to use the API. This policy will require...  
[Learn more](#)

☐ **Basic Authentication - LDAP**  
Enforces HTTP Basic authentication against the specified LDAP server. Organizations often use...  
[Learn more](#)

☐ **XML Threat Protection**  
Protects against malicious XML in API requests.  
[Learn more](#)

☐ **IP Allowlist**  
Limits all service calls to a defined set of IP addresses.  
[Learn more](#)

☐ **Basic Authentication - Simple**  
Enforces HTTP Basic authentication according to the details configured in the policy.  
[Learn more](#)

☐ **IP Blocklist**  
Denies service calls from a defined set of IP addresses.  
[Learn more](#)

☐ **OAuth 2.0 Access Token Enforcement using Mule OAuth Provider**  
Enforces use of an OAuth 2.0 access token issued through an Mule OAuth provider. This policy will require...  
[Learn more](#)

☐ **Json Threat Protection**  
Protects against malicious JSON in API requests.  
[Learn more](#)

☐ **Detokenization**  
**You need permissions to apply this policy. [Learn more](#)**

☐ **Tokenization**  
**You need permissions to apply this policy. [Learn more](#)**

Quality of service

Cancel Next

## JWT Origin

Al configurar la política de JWT Validation, se me solicitó especificar el origen del JWT (JWT Origin). Elegí la opción de Custom Expression debido a que el estándar HTTP no permite enviar múltiples encabezados con el mismo nombre, como por ejemplo dos encabezados Authorization: Bearer en una misma solicitud.

Dado que la política Client ID Enforcement ya utiliza el encabezado Authorization para validar el Client ID y Client Secret, no es posible utilizar el mismo encabezado para el token JWT sin generar conflicto.

Por esta razón, configuré la política JWT para que lea el token desde una ubicación diferente dentro de la solicitud mediante una expresión personalizada, lo cual garantiza que ambas políticas puedan coexistir sin interferencias.

### JWT origin

Origin of the JWT.

- ☐ HTTP Bearer Authentication Header
- ☒ Custom Expression

### Token Expression

DataWeave Expression to be used to extract the JWT from API requests

`#[attributes.headers['x-access-token']]`

## JWT Signing Method

En la configuración de la política JWT Validation, se debe seleccionar el método de firma (Signing Method) que se utilizará para validar la integridad y autenticidad del token.

Elegí el método HMAC (Hash-based Message Authentication Code) porque es un algoritmo ampliamente utilizado que permite verificar que el token JWT no haya sido alterado, utilizando una clave secreta compartida entre el emisor y el receptor.

Este método es eficiente y seguro para escenarios donde ambas partes pueden compartir una clave secreta, como en muchas arquitecturas de APIs internas o cuando se controla completamente el sistema que emite y consume los tokens.

### JWT Signing Method

Specifies the method to be used by the policy to decode the JWT.

HMAC

## JWT Signing Key Length

Este se refiere al tamaño de la clave secreta usada para firmar y verificar el token JWT.

Una clave más larga suele ser más segura porque es más difícil de adivinar o romper mediante ataques. Por eso, es importante usar una longitud adecuada para garantizar la protección del token y evitar vulnerabilidades.

### JWT Signing Key Length

Specifies the length of the key to be in the signing method for HMAC, or the SHA algorithm used for RSA or ES. Ignore this field if the JWT Signing Method was set to None.

☒ 256

☐ 384

☐ 512

## JWT Decoder

## JWT Encoder

Fill in the fields below to generate a signed JWT.

HEADER: ALGORITHM & TOKEN TYPE

CLEAR

Valid header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

## JWT Key origin y JWT Key

La JWT Key es la clave secreta que se utiliza para firmar y verificar el token JWT, y debe coincidir con la que se generó al crear el token.

Por ejemplo, generé el token en una herramienta como [jwt.io](https://jwt.io), la clave que uses para firmar el token es la misma que debes configurar en la política de validación en MuleSoft para asegurar que la firma sea válida.

### JWT Key origin

Origin of the JWT Key. The JWKS option is only supported if the JWT Signing Method was set to RSA or ES. Ignore this field if the JWT Signing Method was set to None.

- ☒ Text
- ☐ JWKS

### JWT Key

The shared secret in case the JWT Signing Method is set to HMAC. Include the public PEM key without -----BEGIN PUBLIC KEY----- and -----END PUBLIC KEY----- for RSA or ES signing. Ignore this field if the JWT Signing Method was set to None.

.....


Show

## Expiration Claim Mandatory y Validate Custom Claim

- ☒ **Skip Client Id Validation**  
Skips client application's API contract validation.
- ☐ **Validate Audience Claim**  
The JWT will be valid only if the aud claim contains at least one audiences value defined here.
- ☒ **Expiration Claim Mandatory**  
If a claim is marked as mandatory, and this claim is not present in the incoming JWT, the request will fail.
- ☐ **Not Before Claim Mandatory**  
If a claim is marked as mandatory, and this claim is not present in the incoming JWT, the request will fail.
- ☒ **Validate Custom Claim**  
The JWT will be valid only if all DataWeave expressions defined here are valid.

### Mandatory Custom Claim Validations (Optional)

Specify the Claim Name and the literal to validate the value of a claim. E.g foo : fooValue. If more complex validations must be made or the claim value is an array or an object, provide Claim Name and DataWeave expression to validate the value of a claim. E.g. foo : #[vars.claimSet.foo == 'fooValue']. If a claim is marked as mandatory and this claim is not present in the incoming jwt, the request will fail.

^ #1 

Key

role

Value

#[vars.claimSet.role == "marketing" or vars.claimSet.role == "admin"]



## 1. Expiration Claim Mandatory

Esta opción obliga a que el token JWT incluya una claim de expiración (exp). Con esto, se asegura que el token tenga una vida útil definida y que no pueda ser usado indefinidamente, lo cual es clave para mantener la seguridad del acceso.

## 2. Validate Custom Claim

Esta opción permite validar una o más claims personalizadas dentro del token. En mi caso, configuré esta opción para validar que el rol del usuario fuera adecuado. Específicamente, verifiqué que la claim role tuviera un valor igual a "admin" o "marketing", ya que solo esos perfiles están autorizados a consultar la información sensible de clientes.

Por lo tanto, nuestro token debe contener dichas llaves y valores respectivamente.

<https://jwt.io/>

The image shows the 'JWT Encoder' section of the 'JWT Decoder / JWT Encoder' tool. The interface is dark-themed and contains several input fields and a 'Generate example' button.

**JWT Encoder**

Fill in the fields below to generate a signed JWT.

**HEADER: ALGORITHM & TOKEN TYPE** (CLEAR)

Valid header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

**PAYLOAD: DATA** (CLEAR)

Valid payload

```
{
  "sub": "1234567890",
  "name": "PriscilaM",
  "role": "marketing",
  "exp": 4102444800
}
```

**SIGN JWT: SECRET** (CLEAR)

Valid secret

a-string-secret-at-least-256-bits-long

Encoding Format: UTF-8

**JSON WEB TOKEN** (COPY)

Generate example

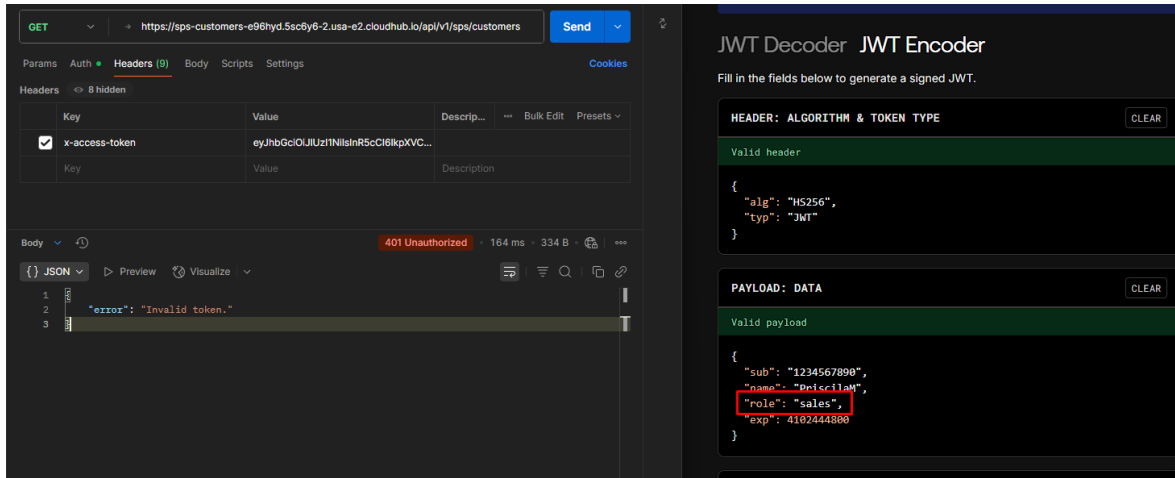
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IjB5aXNjaWxhTSIsInVubGVuYXNpdCI6IjE4IiwiaWF0IjoxMjM0NTY3ODkwMDI0NDQ4MDh9.F1rU9Ghm2JXgDru3YfsG5V-HSeTNOiTDi_R7TwxpU
```

Share feedback | Report issue

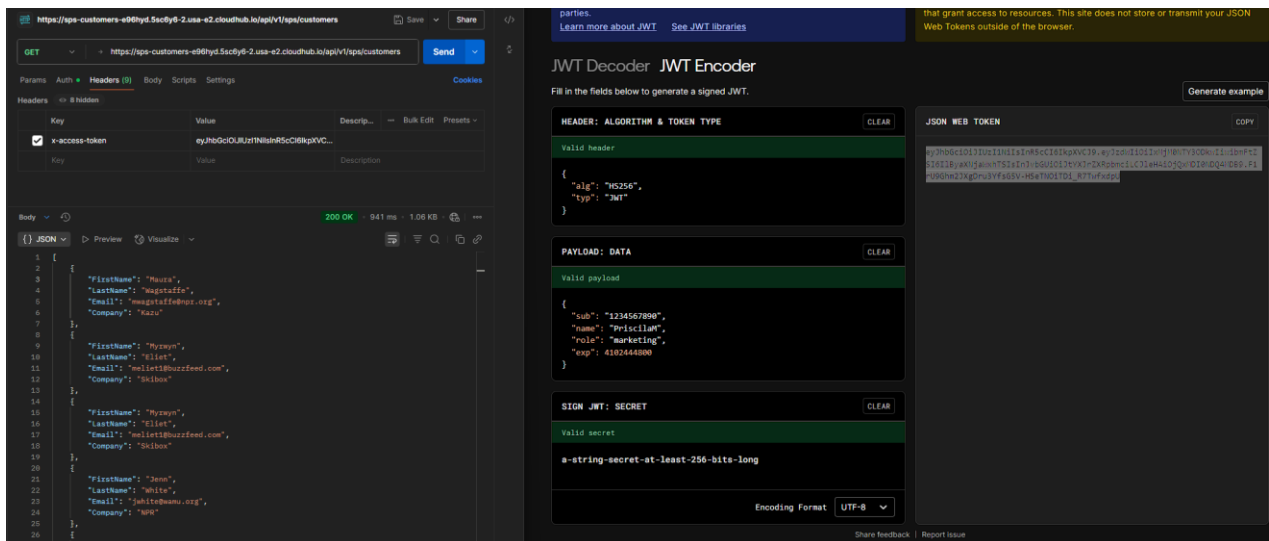
## Resultados del punto adicional

Una vez definida la validación de la claim personalizada role, configuré la política para permitir el acceso únicamente a los usuarios cuyo rol fuera "marketing" o "admin".

Al probar con un token que contenía un valor distinto, como por ejemplo "sales", la API respondió con un código de estado 401 (Unauthorized), lo cual indica que el acceso fue correctamente denegado.



Sin embargo, si el token cumple con todos los parámetros definidos en la configuración del JWT — es decir, si no ha expirado y contiene un role válido como "marketing" o "admin" —, la API responde correctamente con un código 200 OK, lo que confirma que el acceso ha sido autorizado.



Este enfoque refuerza significativamente la seguridad de la API, ya que combina dos mecanismos complementarios: por un lado, Client ID Enforcement asegura que solo aplicaciones registradas y autorizadas puedan consumir la API, y por otro, JWT Validation permite controlar el acceso a nivel de usuario, validando tanto la vigencia del token como los roles permitidos.