



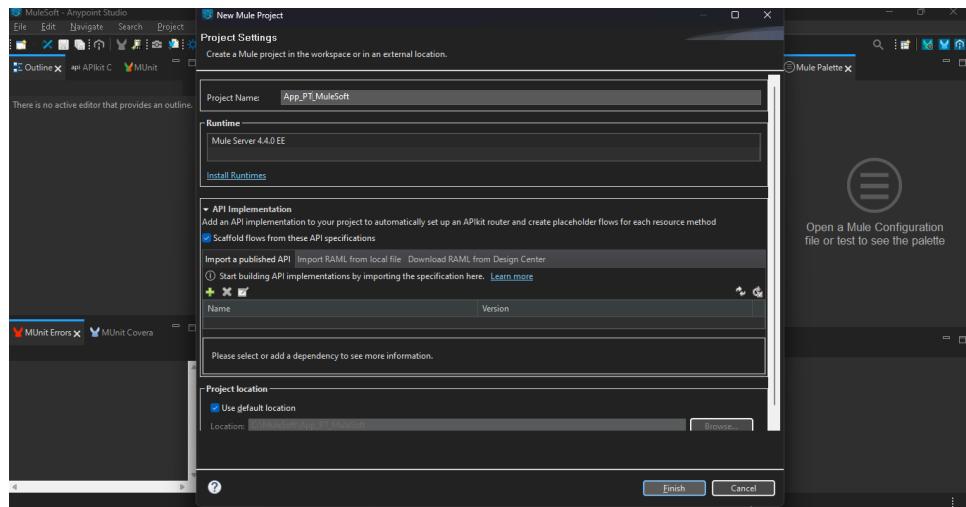
Práctica MuleSoft Trainee

Ulices Salvador Aguila Contreras

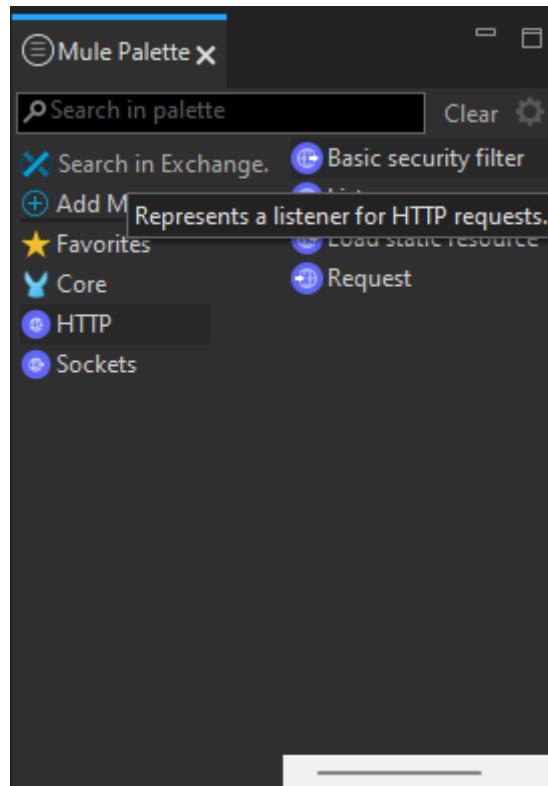
Build your first Hello Mule application.

Creación, prueba y despliegue de Aplicación Mule

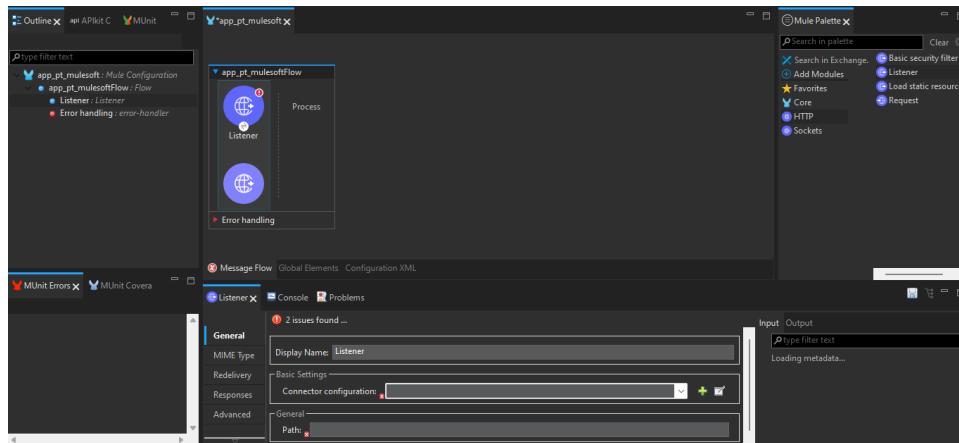
En Anypoint Studio abre el asistente **New Mule Project**. En el campo **Project Name** se ingresa el valor App_PT_MuleSoft luego Se hace clic en **Finish**.



Posteriormente a la creación del proyecto, se colocan los componentes necesarios para la aplicación, por lo que se toman de la sección de componentes (Mule Palette) y se arrastran al proyecto.



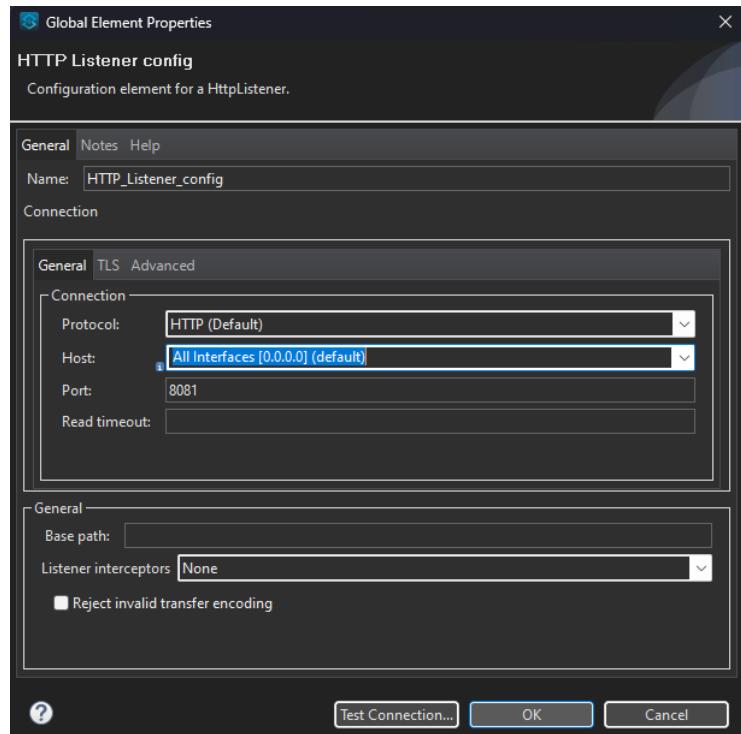
Para este proyecto se usa un oyente(listener), el cual es el punto final HTTP que escucha una request HTTP para llegar a la URL que se defina. Cuando el Listener recibe una request HTTP, el contenido dentro del flujo se ejecutará en el orden que se defina.



Para el oyente primeramente se asignan las configuraciones respectivas.



En las configuraciones se definen el nombre, puerto, el protocolo y el host para el proyecto, en este caso, se usa la configuración por defecto.

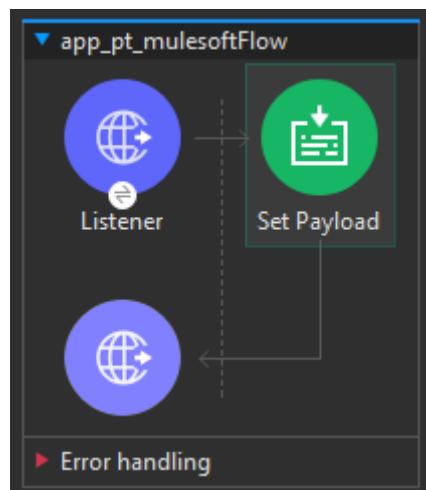


La ruta que se defina representará el punto final que ejecutará el flujo cuando se realice una request HTTP a su Listener HTTP.

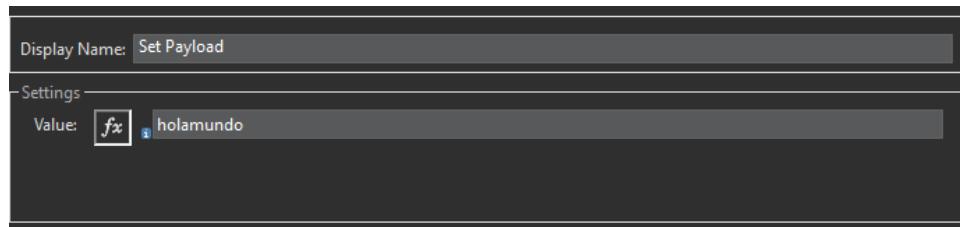
Posterior a realizar la configuración del conector, se asigna una Ruta(path), la cual sera tomado como base para realizar las peticiones respectivas.



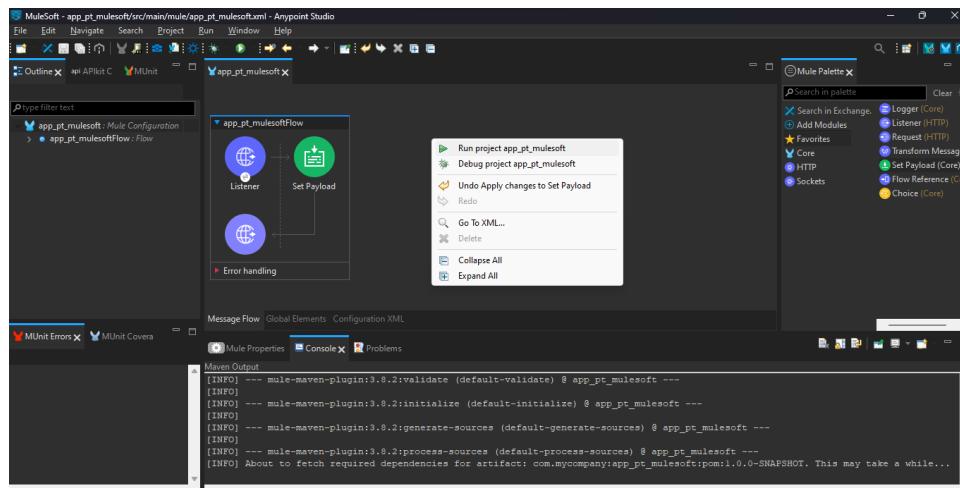
Realizado los pasos anteriores, el oyente ya está listo. Dicho lo anterior, se asignan la carga util (set payload).



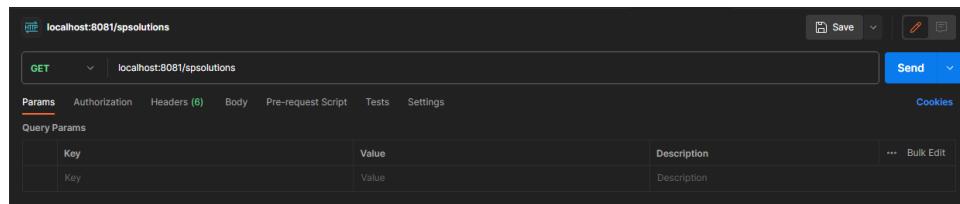
En la carga util la cual es el valor que se muestra al realizar la peticon HTTP, se establece los valores que se muestran a continuación.



Establecido los valores, se guardara los cambios para establecer y mostrar los valores antes asignados. Al guardar los datos se podrá ejecutar el proyecto para mostrar los datos en local.



Para realizar el Test de la aplicación, se usará Postman para realizar las peticiones, usando el Path generado para el proyecto.

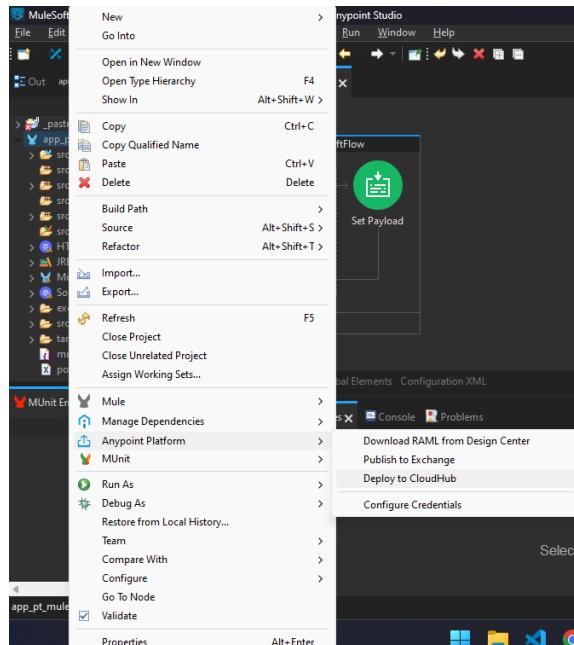


Al ejecutar la petición en Postman. Este devuelve un status 200 OK, el cual demuestra que la petición se realizó de manera correcta, además de que esta devuelve los valores antes asignados.

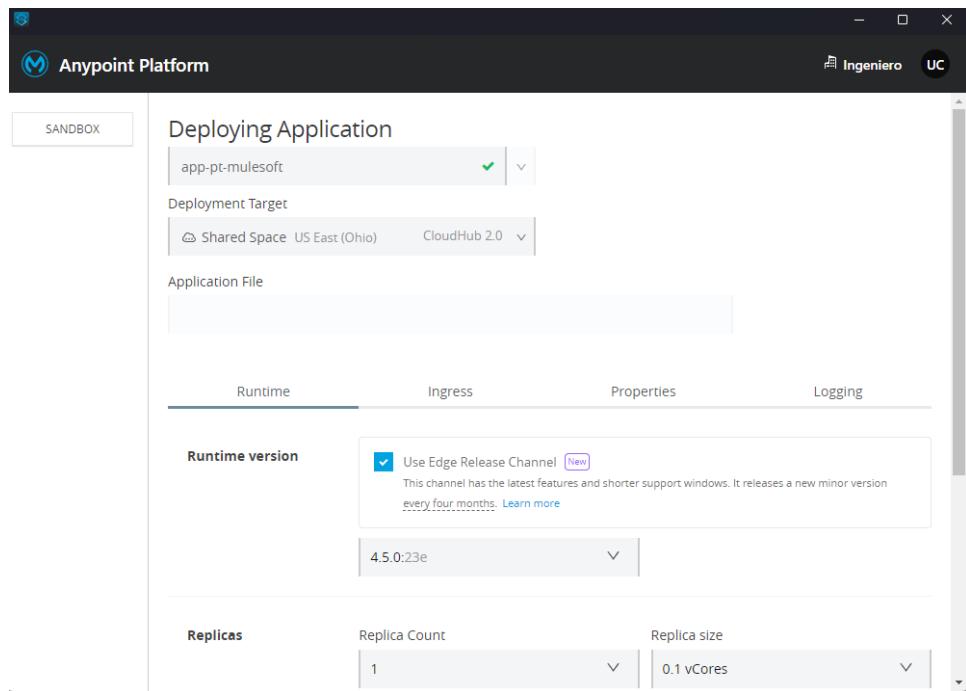
The screenshot shows a Postman interface with a single GET request to 'localhost:8081/spsolutions'. The 'Params' tab is selected, showing a 'Query Params' table with one entry: 'Key' with value 'holamundo'. The 'Body' tab shows the response content: '1 holamundo'. The status bar at the bottom indicates a 200 OK status, 13 ms time, and 84 B size.

Realizados las pruebas anteriores, para verificar que las peticiones se realicen de manera correcta, además de verificar que no haya errores al ejecutar la aplicación, se procede a realizar el despliegue de la aplicación.

Para desplegar el proyecto en el apartado de WorkSpace se selecciona el archivo del proyecto y se dara click derecho para buscar la opcion **Anypoint Platform** y se selecciona **Deploy to CloudHub**.



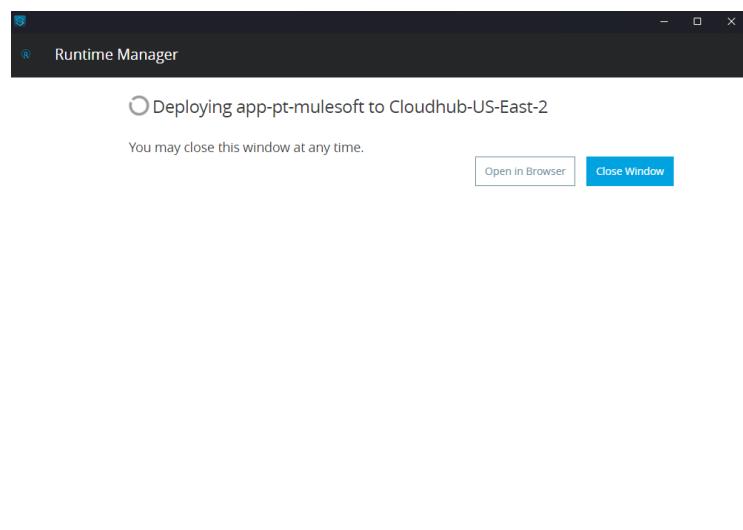
Posteriormente se muestra una venta emergente, en donde se deberá colocar el nombre del proyecto para desplegarlo en CloudHub.



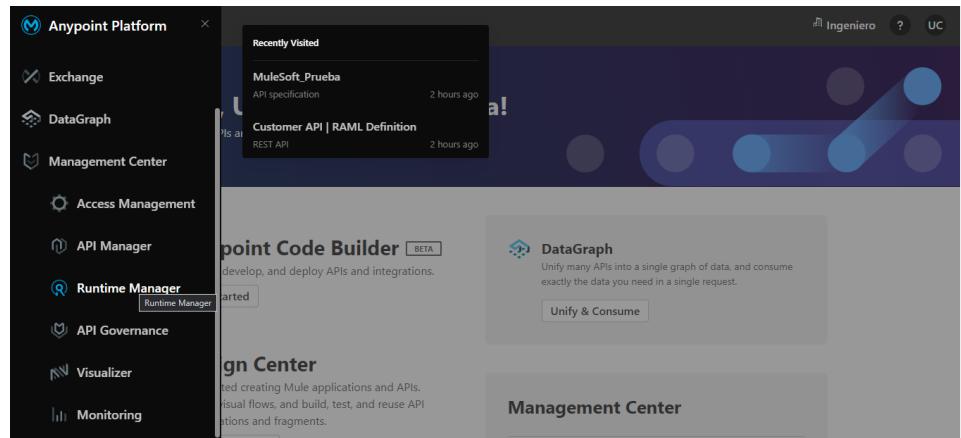
Al realizar el despliegue, MuleSoft prepara el proyecto.



Al acabar el proceso, se mostrará un aviso para notificar que el proceso ha terminado y se selecciona **Close Window**



Para visualizar el proyecto desplegado, se tendrá que visitar la página oficial de MuleSoft, en la pestaña de **Anypoint Platform > Runtime Manager**



En la plataforma, se muestra dos opciones, se selecciona **SandBox**

Al ingresar se observa el proyecto desplegado.

Name	Target Name	Target Type	Status	Runtime Version	Date Modified
app-pt-mulesoft	Shared Space	CloudHub 2.0	Running	4.5.0.23e	2023-11-01 16:58:33

Dentro del proyecto se visualiza las configuraciones respectivas y el Endpoint para ser consumido.

app-pt-mulesoft

Application status: Running
Configuration: 2e162e
Last updated: 2023-11-02 4:32:52PM
Replicas: 1 / 1 started

Public Endpoint: <https://app-pt.mulesoft.qafk5.Scfy6-1.usa-e2.cloudhub.us>
Target name: Cloudhub-US-East-2
Target type: Shared Space

Configuration 2e162e

Application File: app_pt_mulesoft.jar
Version 1.0.5

Deployment Target: Ingress Properties Logging

Runtime version: 4.5.0.23e
Use Edge Release Channel:

Para verificar que todo sea correcto, se hace uso de EndPoint generado seguido de “/spsolutions”.

The screenshot shows the Postman interface with the following details:

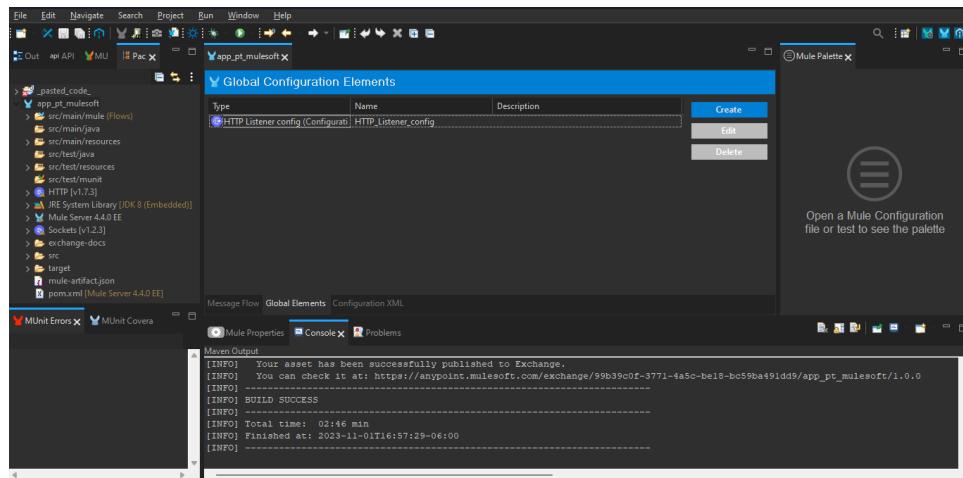
- Request URL:** https://app-pt-mulesoft-qafxic.5sc6y6-1.usa-e2.cloudhub.io/spsolutions
- Method:** GET
- Headers:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:** Key: Value
- Body:** Status: 200 OK, Time: 761 ms, Size: 164 B, Save as example
- Response Body:** 1 holamundo

How to set up your global elements and properties files in Anypoint Studio.

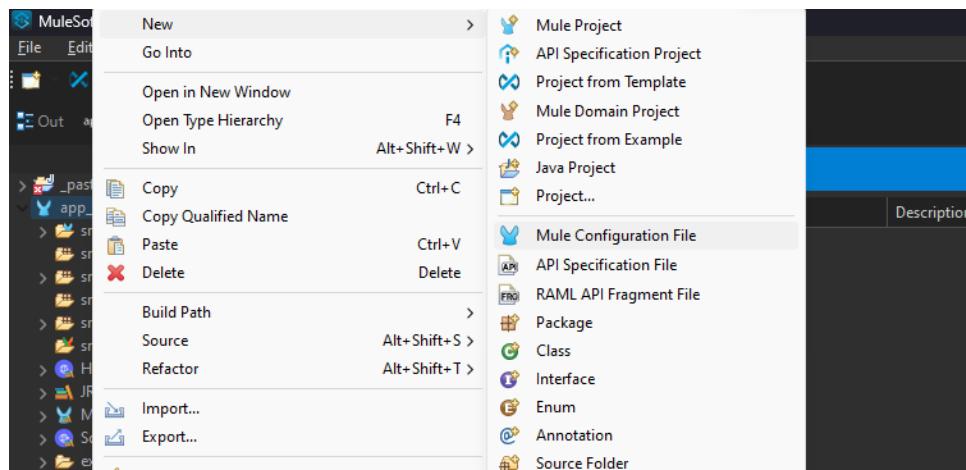
Utilizar archivo de propiedades para mantener y referenciar datos sensibles.

Es una buena práctica crear un nuevo archivo **global.xml** para mantener todos los elementos globales almacenados allí. Esto es útil cuando tenemos más archivos XML y no queremos buscar nuestros elementos globales en cada archivo.

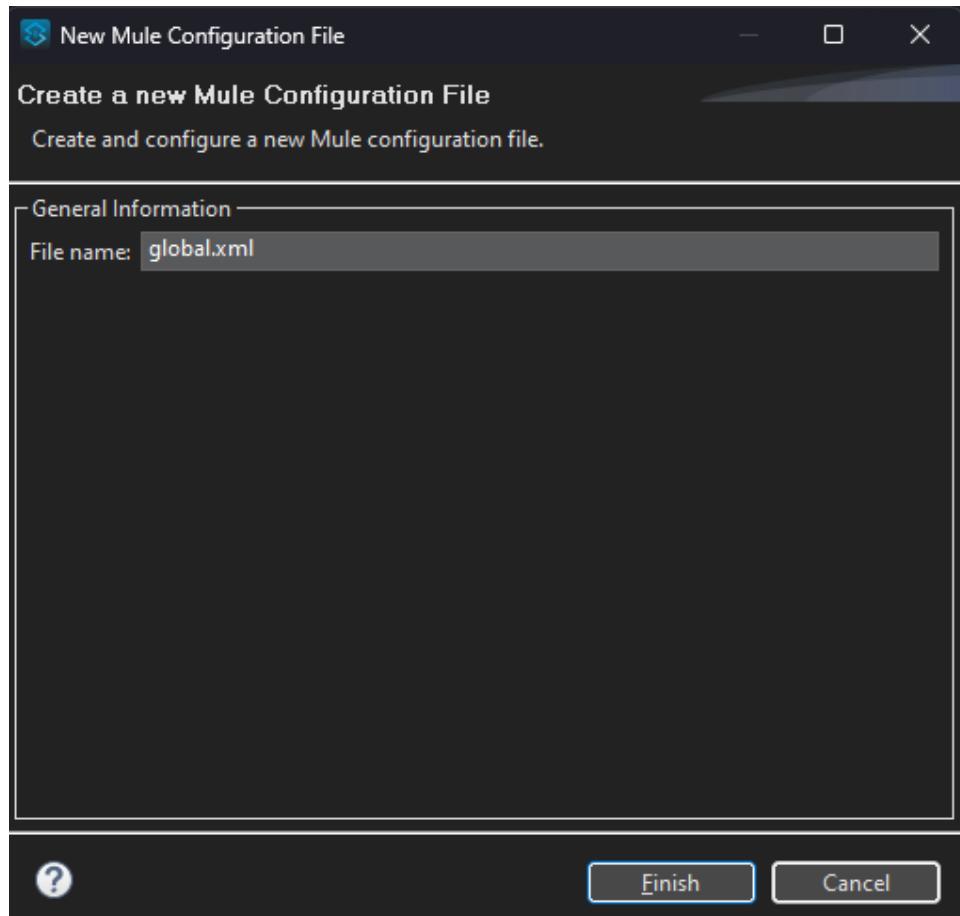
Para ver los archivos **global.xml**, se podrá establecer en la pestaña Global Elements, el cual muestra la ventana de configuración de elementos globales.



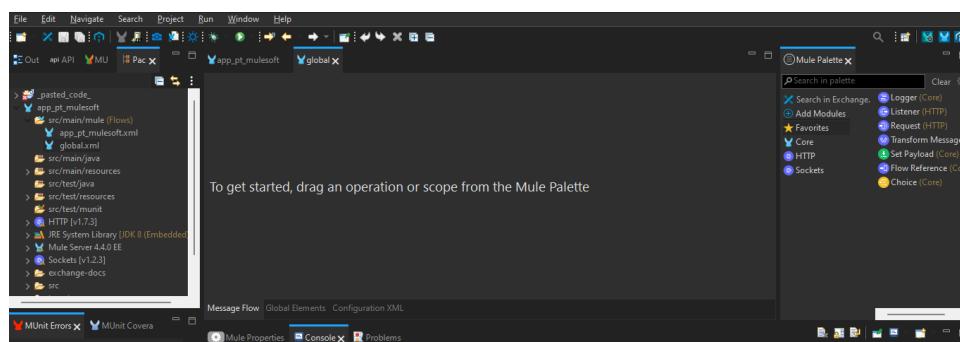
Para crear un nuevo elemento, se hace clic derecho en la carpeta **src/main/mule**. Seleccionar **Mule Configuration File**.



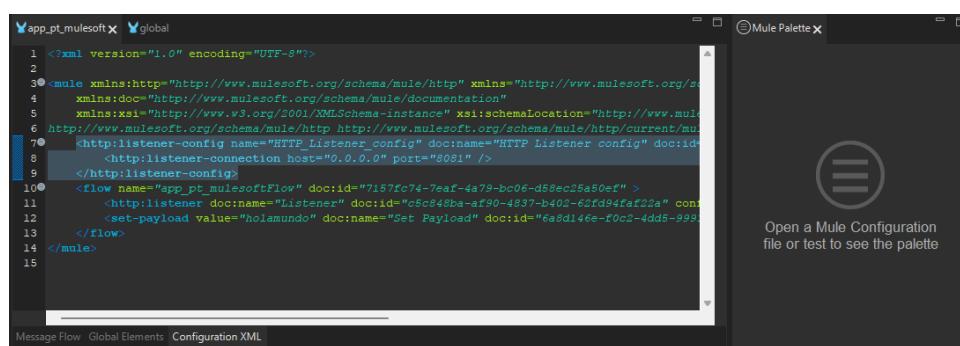
Se agrega el nombre **global.xml** y se hace clic en Finish.



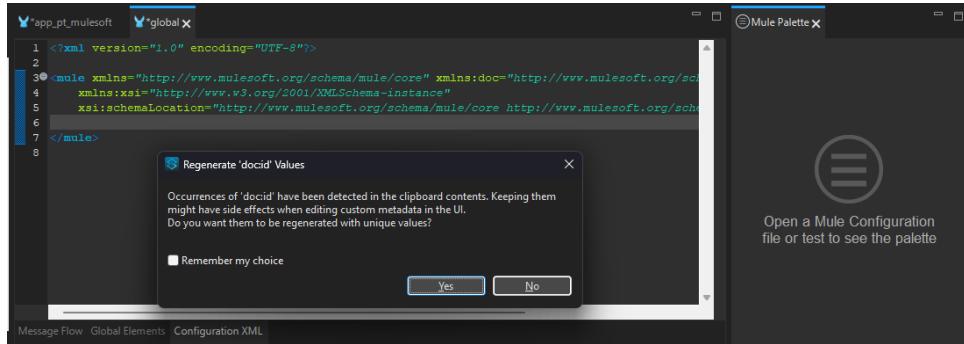
De regreso en el archivo **app_pt_mulesoft.xml** y se realiza el cambio a la vista XML de configuración. Localiza el `http:listener-config` código y se corta todo lo que esté entre las etiquetas.



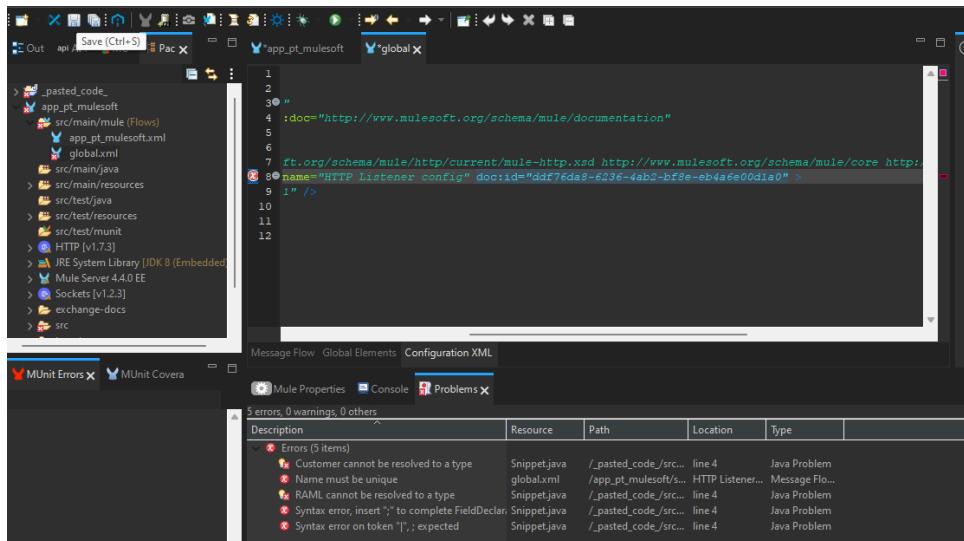
Las etiquetas que se cortan tienen la estructura `<http:listener-config>`



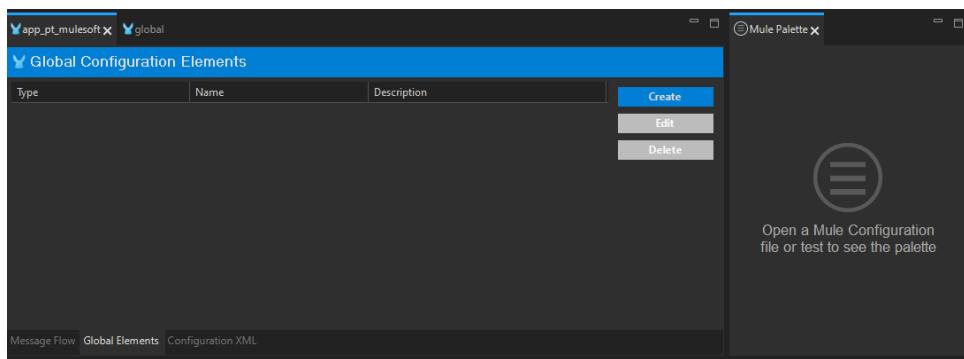
Posteriormente se pegan al archivo **global.xml** y se cambia a la vista **Configuration XML**. El código se pega dentro de las etiquetas mule.



Al pegar el código se observa cómo puede recibir un error en este punto. Esto se debe a que el archivo **app_pt_mulesoft.xml** no se ha guardado y Studio cree que ahora tiene dos elementos globales con el mismo nombre. Para solucionar este error solo se deberá dar en Guardar y el error desaparecerá.

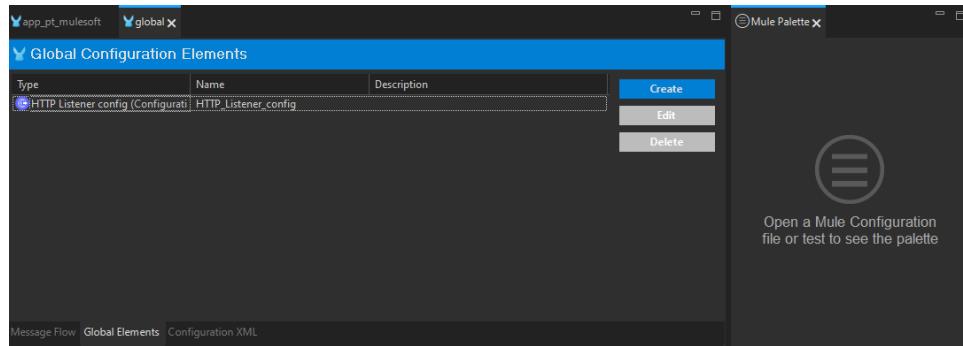


Al quitar elementos del archivo **app_pt_mulesoft.xml** estos ya no se visualizan en los Global Elements para ambos archivos, ahora se visualiza que **HTTP_Listener_config** ya no está presente en el archivo **app_pt_mulesoft.xml** y se movió al archivo **global.xml**.

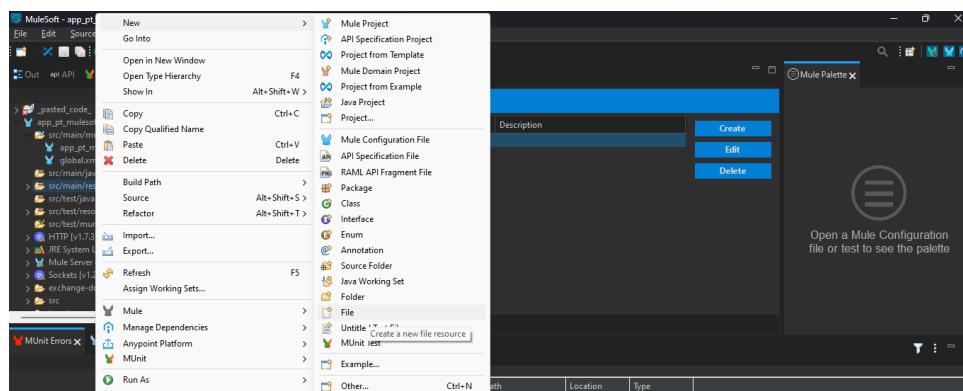


Hecho lo anterior, se deberá externalizar los valores codificados en propiedades.

En el archivo **global.xml**, se selecciona la vista Global Elements. Se selecciona el elemento **HTTP_Listener_config** y posteriormente se da clic en el botón **Edit** a la derecha o se da doble clic en él. Los valores predeterminados para Host y port, estos son valores codificados.



Para evitar tener valores codificados en el código, es una buena práctica externalizarlos en archivos de propiedades. Se hace clic derecho en la carpeta **src/main/resources** y se selecciona **New > File**.

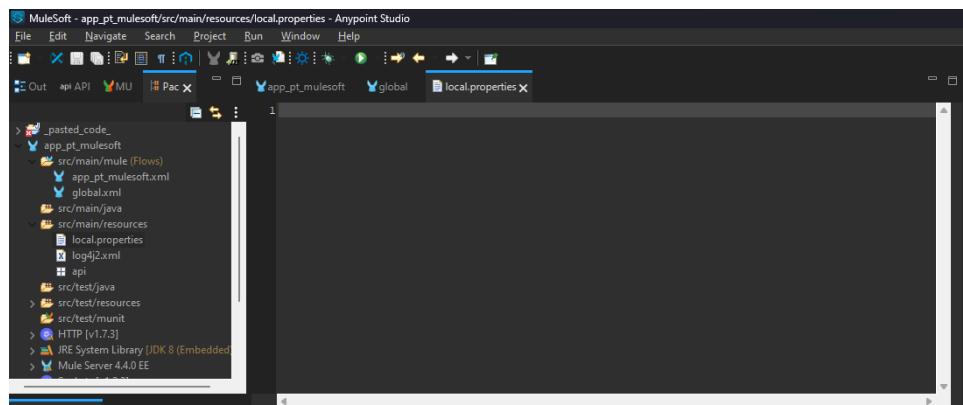
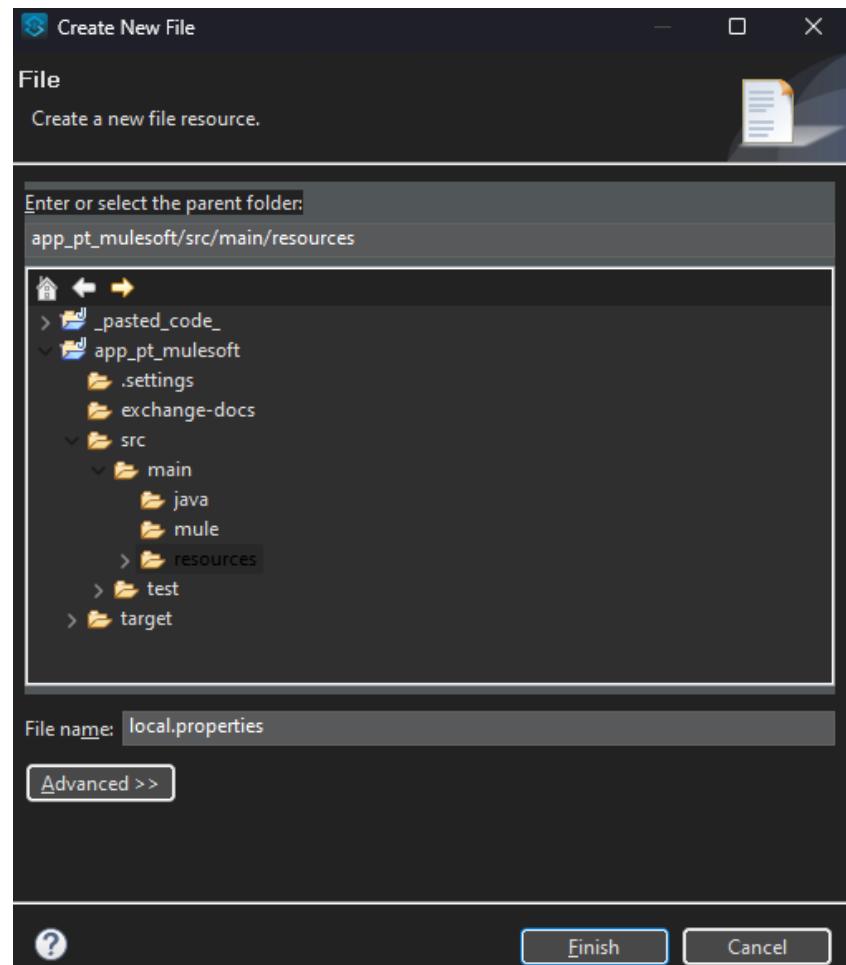


Hay dos tipos de archivos de propiedades admitidos:

- **.properties**
- **.yaml**

La principal diferencia es que los archivos **.properties** son más fáciles de copiar y pegar, pero las propiedades **.yaml** son más limpias de ver.

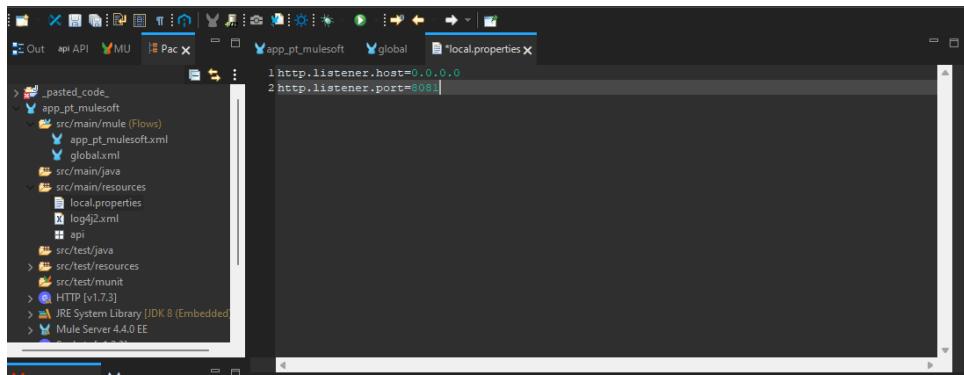
Para este proyecto de usa **.properties** por lo que se crea un archivo **local.properties** en **src/main/resources**.



Dentro de este nuevo archivo, se colocan las siguientes propiedades y se guardan los cambios.

```
http.listener.host=0.0.0.0
```

```
http.listener.port=8081
```

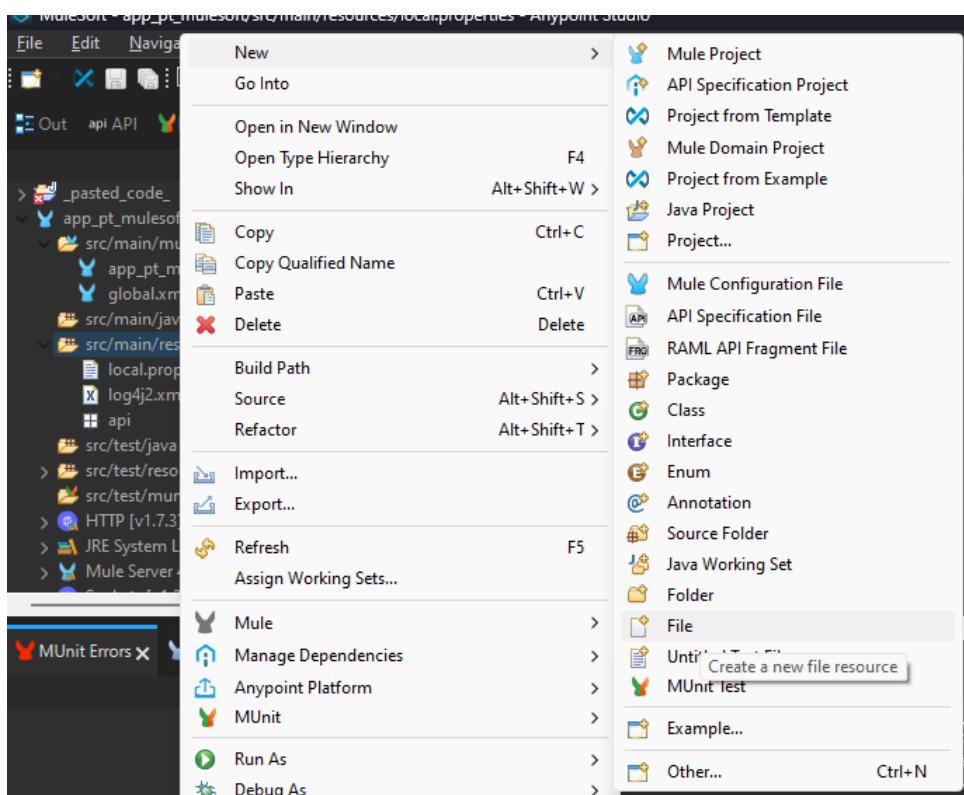


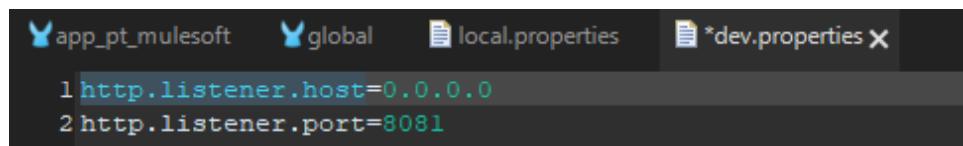
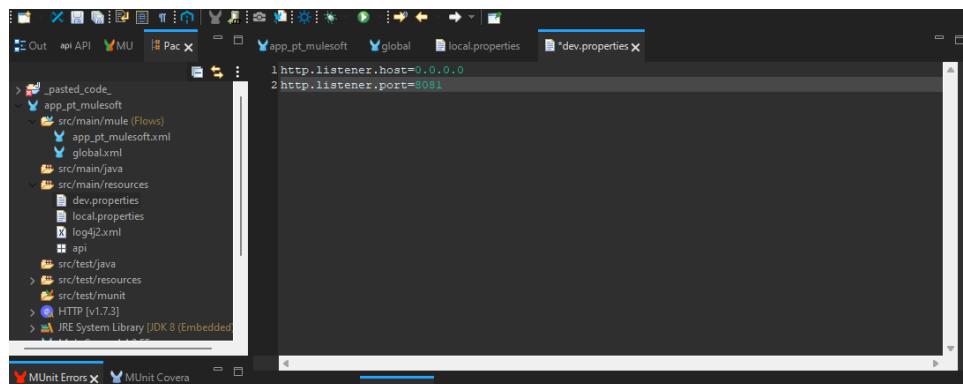
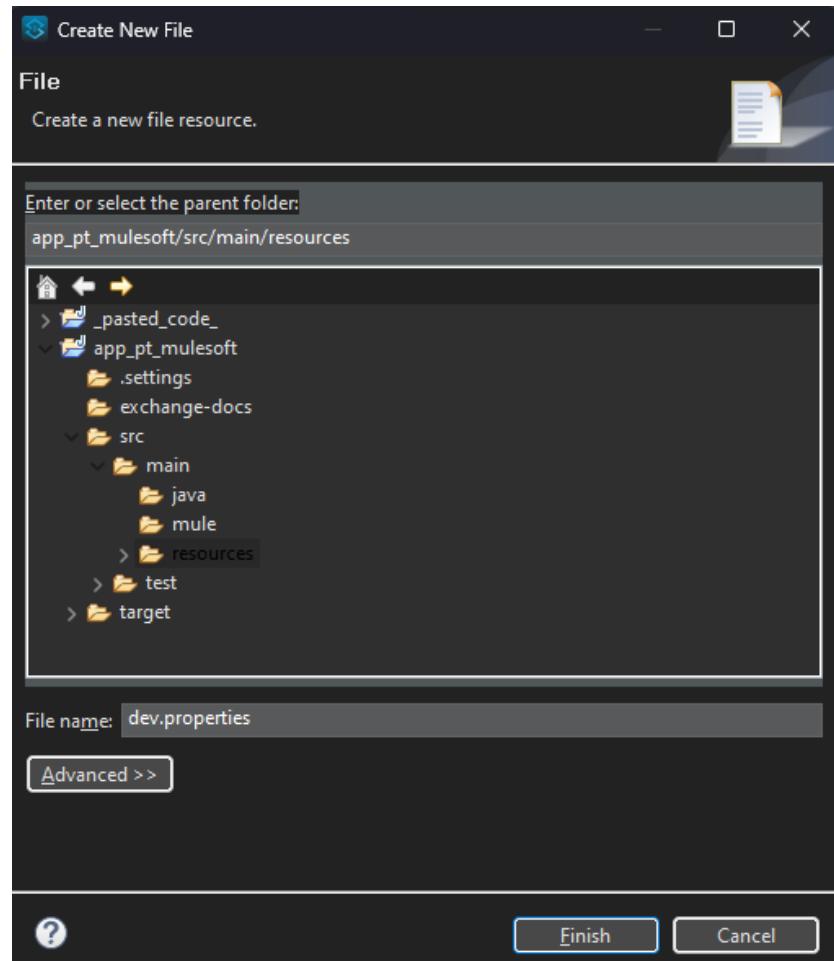
The screenshot shows the Anypoint Studio interface with the file "local.properties" open in the center. The code editor displays the following properties:

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

The left sidebar shows the project structure with folders like "src/main/mule", "src/main/java", and "src/main/resources".

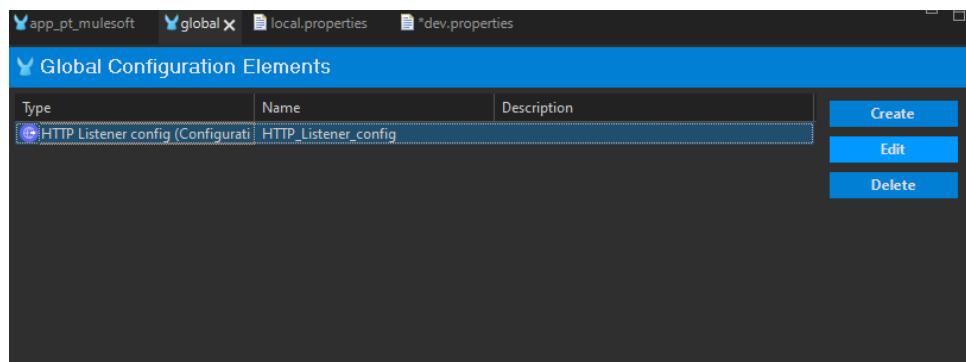
Se repetirán los mismos pasos para un archivo **dev.properties**. Esto ayuda a separar las propiedades por entorno. Cuando se ejecute la aplicación Mule de manera local, se usará el archivo local.properties. Cuando se implemente la aplicación en CloudHub, se usará el archivo dev.properties. En este caso, ambas propiedades tienen los mismos valores.



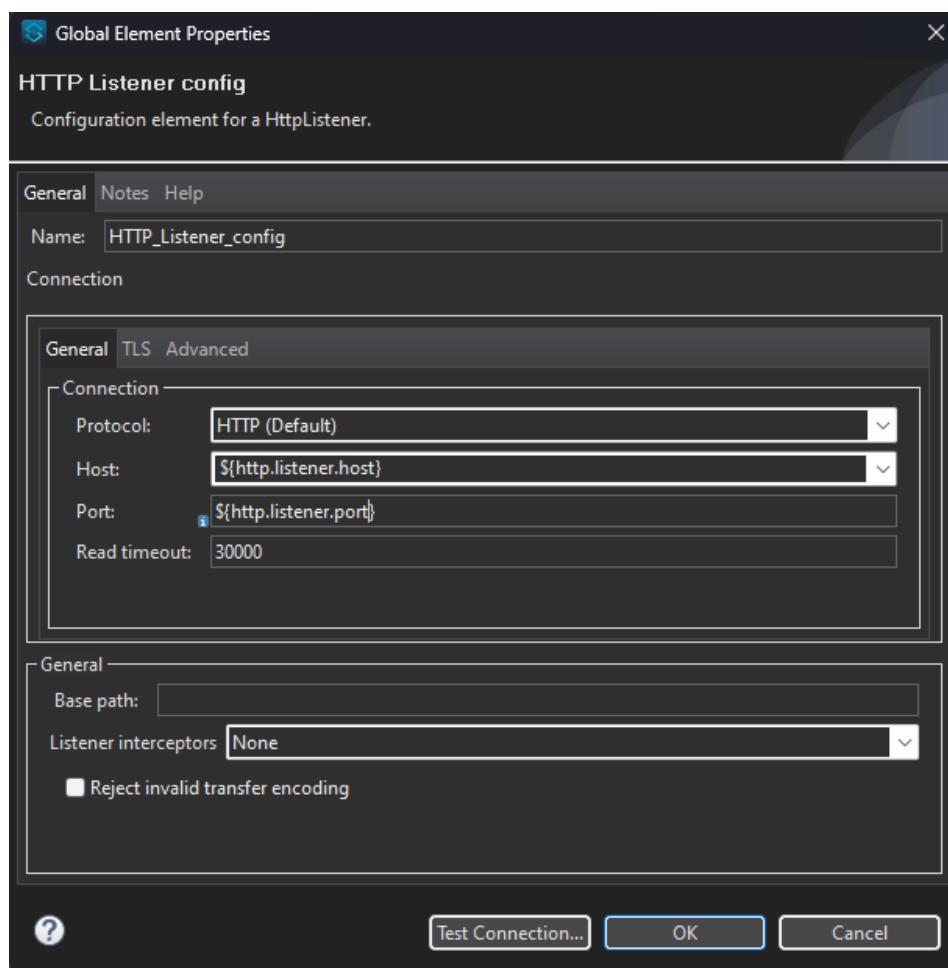


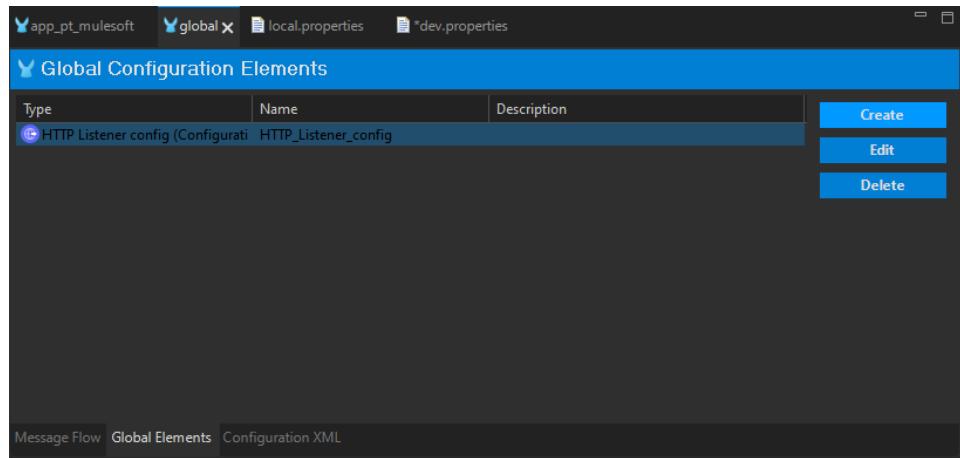
Al volver archivo **global.xml** y en la vista **Global Elements**. Se selecciona el elemento **HTTP_Listener_config** y se hace clic en **Edit**. Para hacer referencia a una propiedad desde cualquier configuración de conector, se debe utilizar la siguiente sintaxis:

`${your.property.name}`



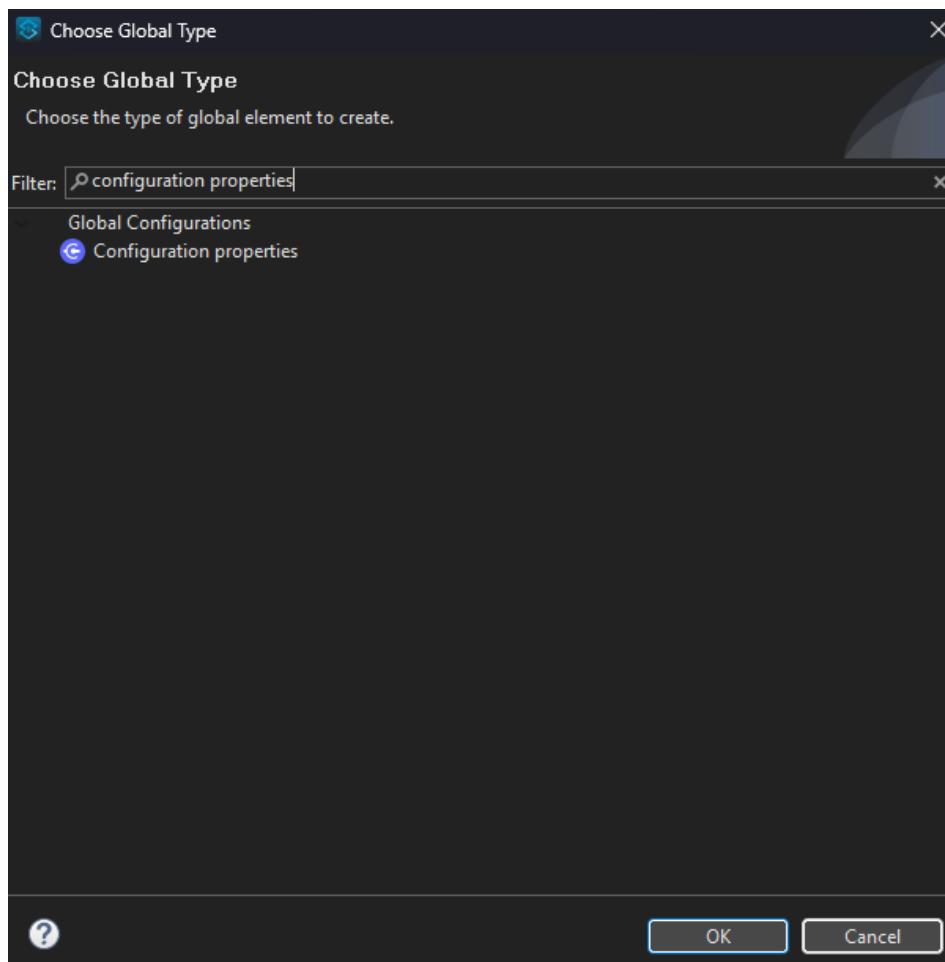
Se Reemplaza cada valor codificado con la propiedad correspondiente, utilizando la sintaxis anterior. La siguiente configuración se establece de la siguiente manera:





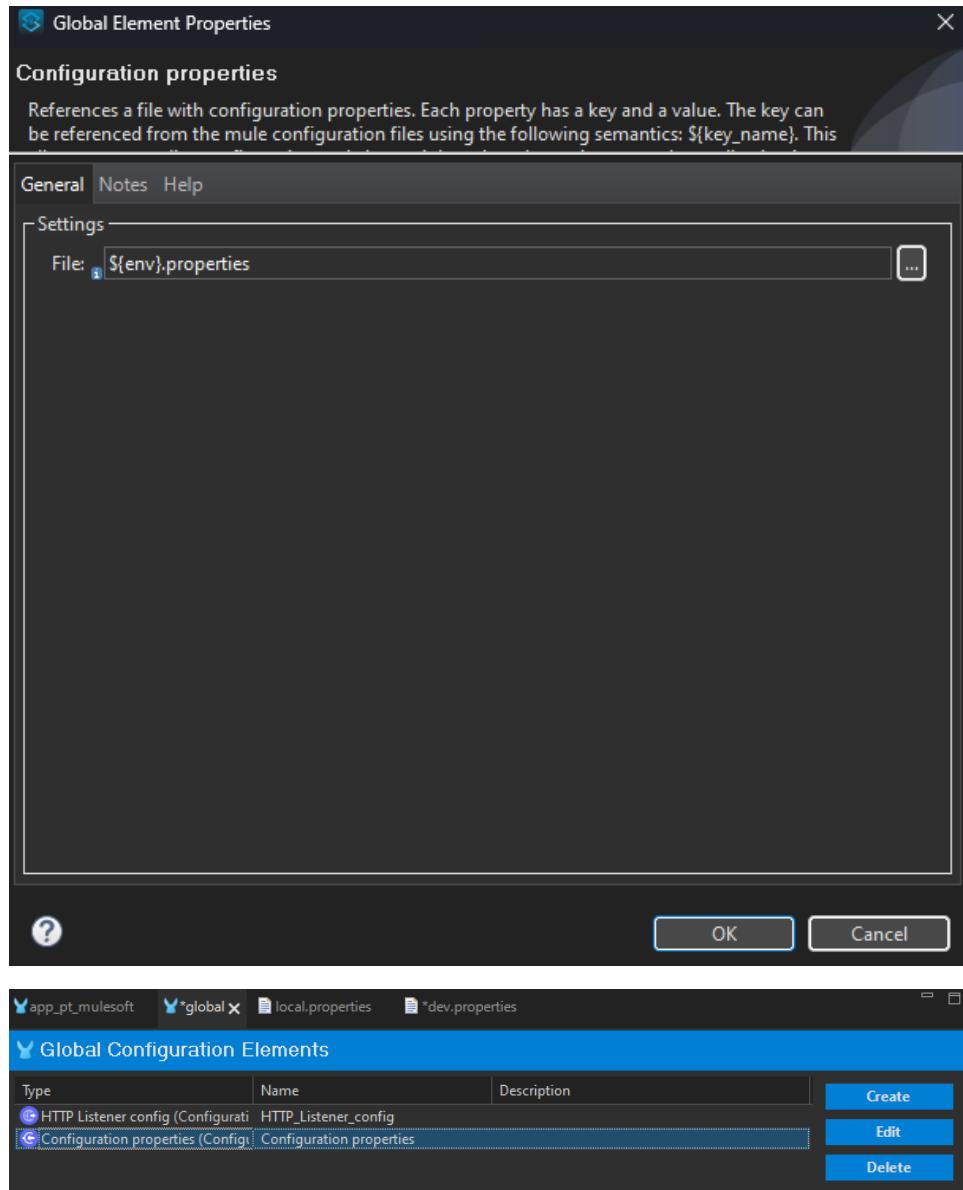
Hecho lo anterior, se realiza la configuración del elemento global de propiedades.

Agregar el archivo de propiedades no es suficiente para que la aplicación Mule sepa dónde buscarlos. Si se intenta ejecutar la aplicación sin esta configuración, arrojará un error indicando que no pudo encontrar sus propiedades. Por lo que en el archivo **global.xml**, En la vista **Global Elements**. Se da clic en **Create** para agregar un nuevo elemento global. En el buscador se escribe **Configuration properties**. Y se selecciona el elemento y se da clic en **Ok**.



Como no se hace uso de archivos de propiedades estáticas, se debe poder cambiar entre archivos según el entorno. Para realizar este paso se usa **`{}$intaxis`** de propiedades:

`{}${env}.properties`



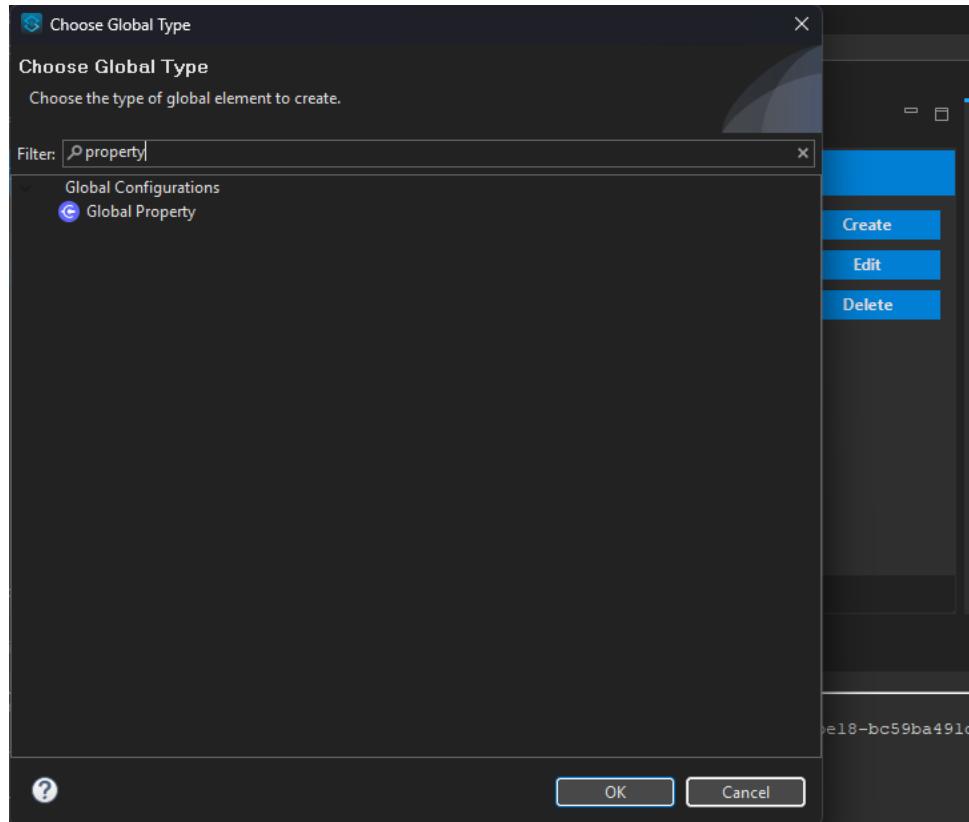
Se debe crear dos elementos globales en el archivo **global.xml**. En las que se agrega la propiedad **env** a la que se acaba de hacer referencia en el paso anterior.

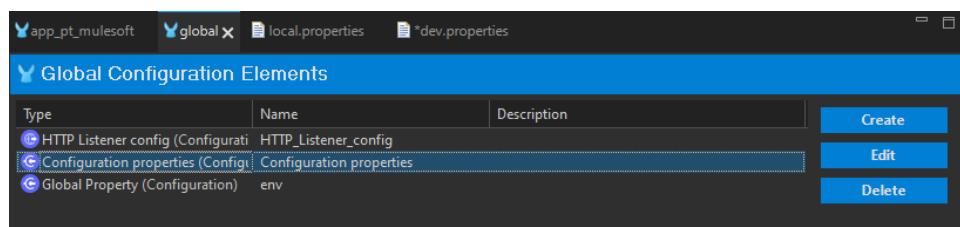
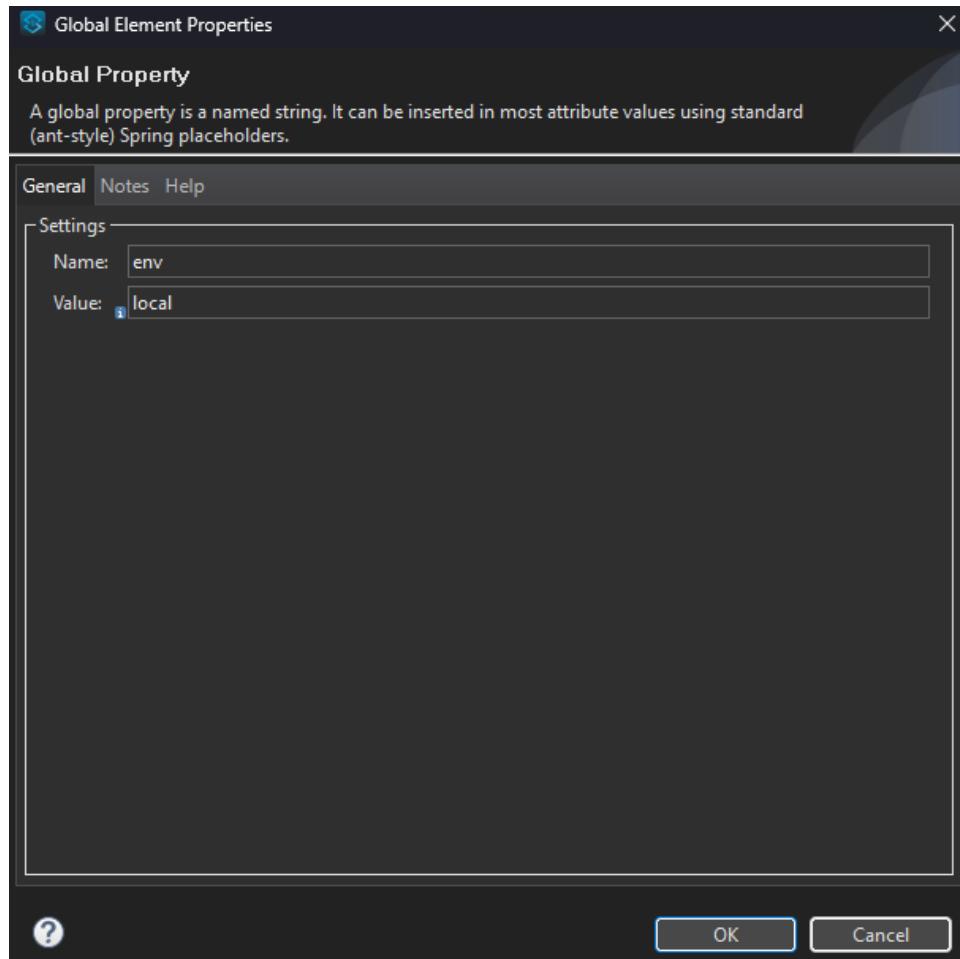
Hay dos formas de hacerlo:

- 1) agregarlo a las variables de entorno en tiempo de ejecución
- 2) crear una propiedad global en los elementos de configuración global.

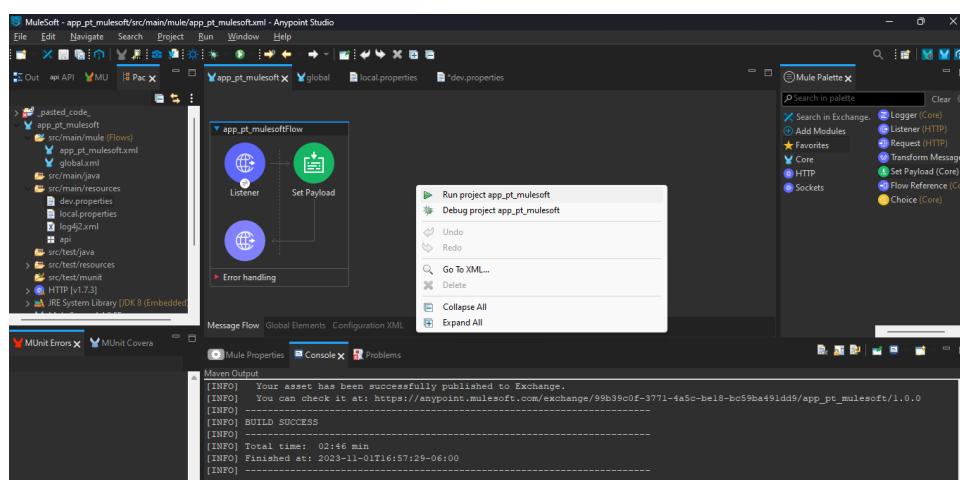
Para este caso, se usa la primera forma. Como esta no es una propiedad segura, se agrega a los elementos de configuración global.

Se clic en **Create** y se selecciona **Global properties**. El nombre debe ser **env** y el valor **local**.





Posteriormente al crear la configuración de los elementos globales, se verifica que Mule funcione correctamente, por lo que primeramente se guardan todos los archivos y posteriormente se ejecuta el proyecto.



También al desplegar, se verifica que no haya errores, por lo que en Postman se hace la respectiva ejecución del path y se espera la respuesta.

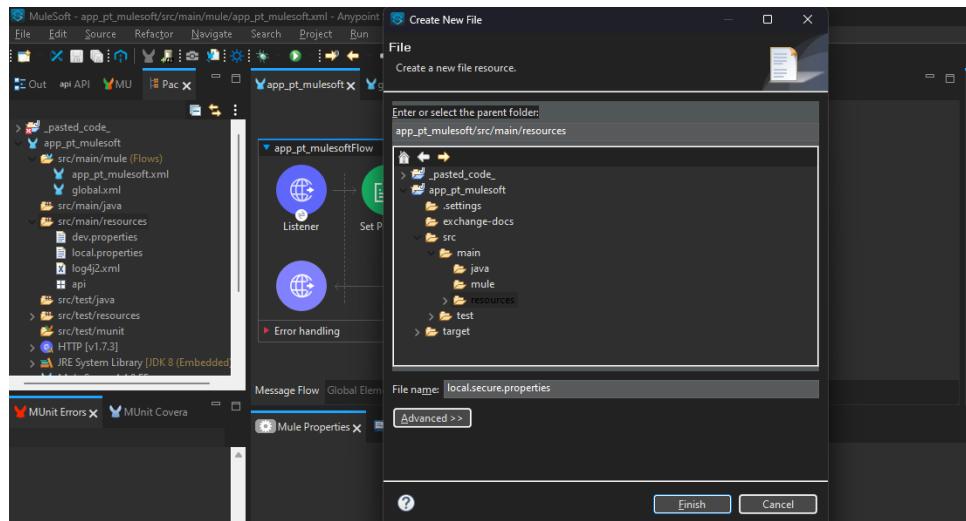
The screenshot shows the Postman application interface. At the top, there are two tabs: "GET https://app-pt-mulesol" and "GET localhost:8081/spsolutions". The second tab is active. Below the tabs, the URL "localhost:8081/spsolutions" is entered. The main workspace shows a table for "Query Params" with one row: "Key" (Value) and "Description". Below this, there are tabs for "Body", "Cookies", "Headers (2)", and "Test Results". The "Body" tab is selected, showing the response body: "1 holamundo". At the bottom right, status information is displayed: "Status: 200 OK", "Time: 364 ms", and "Size: 84 B".

How to secure properties before deployment in Anypoint Studio.

Crear archivo de propiedades seguros para proteger datos sensibles que representan un riesgo.

Para tener una mejor organización de las propiedades, es una buena práctica separar las propiedades seguras en archivos por entorno. Se puede crear un archivo **local.secure.properties** para su configuración local, un **dev.secure.properties** para su entorno de desarrollo, etc.

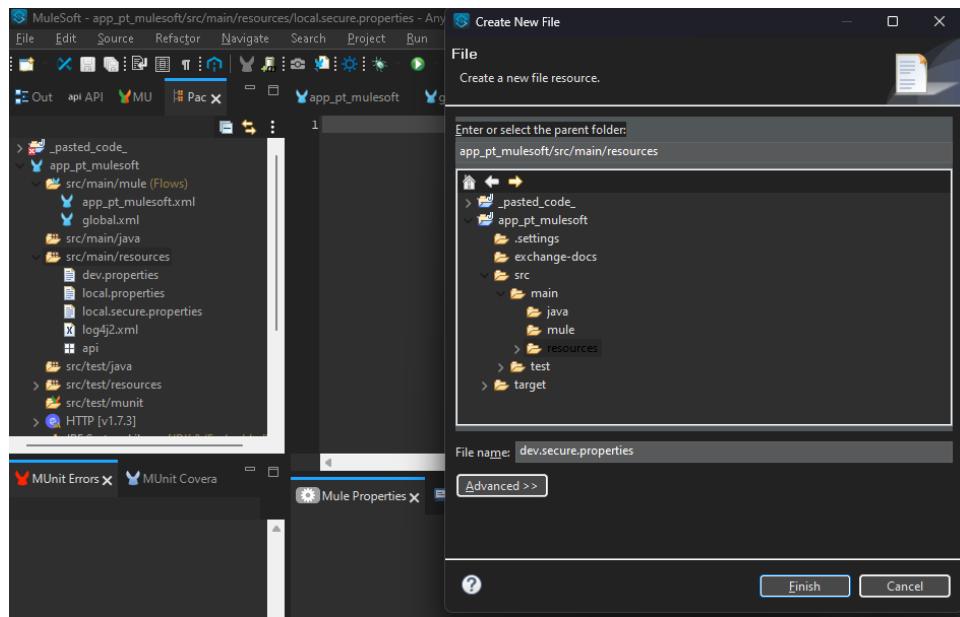
Para crear un archivo **local.secure.properties**, Se hace clic derecho en **src/main/resources** y se da clic en **New > File** y asigna un nombre al archivo **local.secure.properties**.



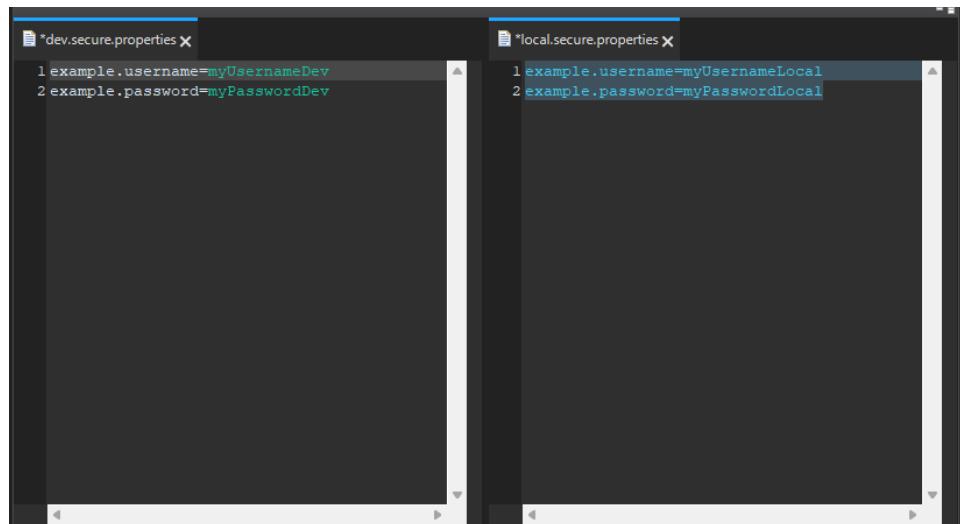
Creado correctamente el archivo, se hace doble clic en el archivo **local.secure.properties** para agregar una nueva entrada. En el archivo de agrega las siguientes propiedades:

`example.username=myUsernameLocal`

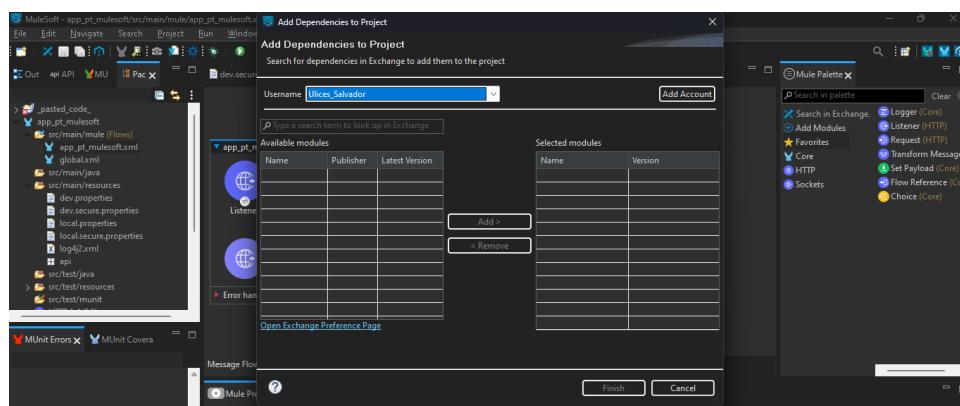
`example.password=myPasswordLocal`

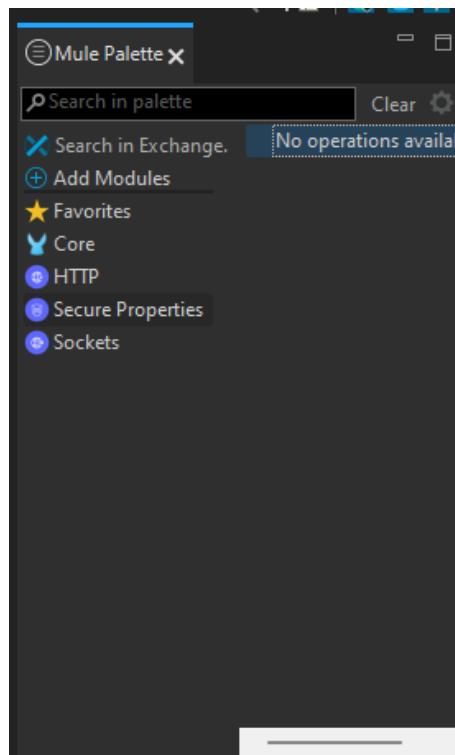
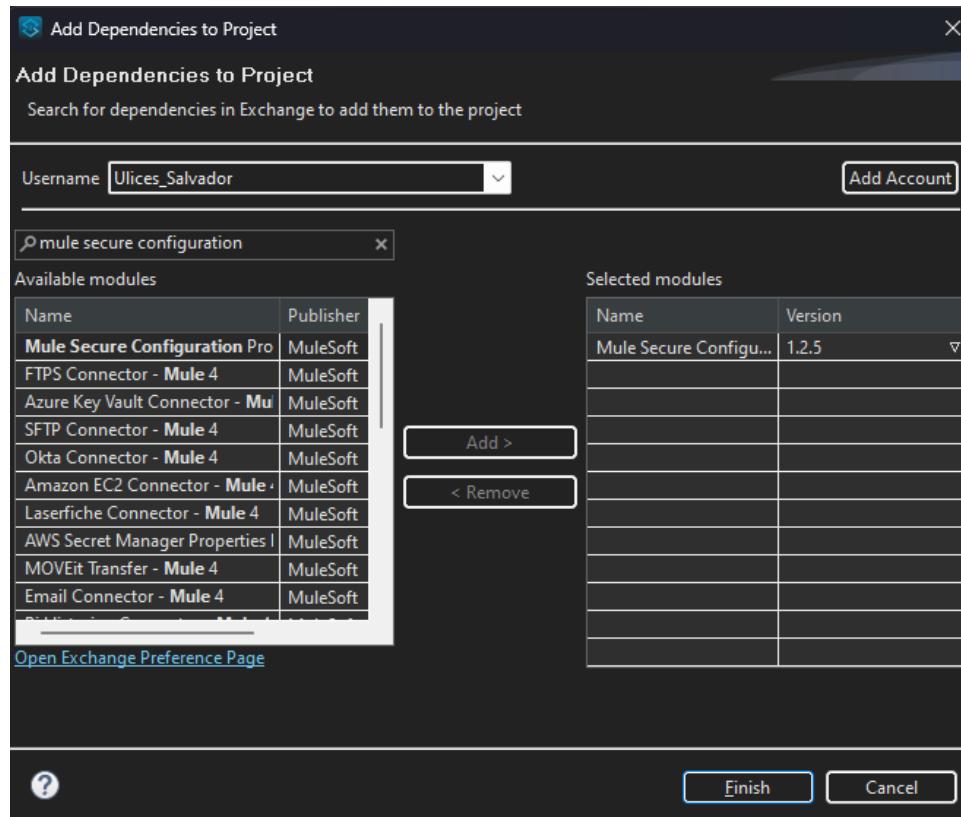


Después de haber ingresado las credenciales privadas en el archivo **local.secure.properties**, se repita los mismos pasos para un archivo **dev.secure.properties**.



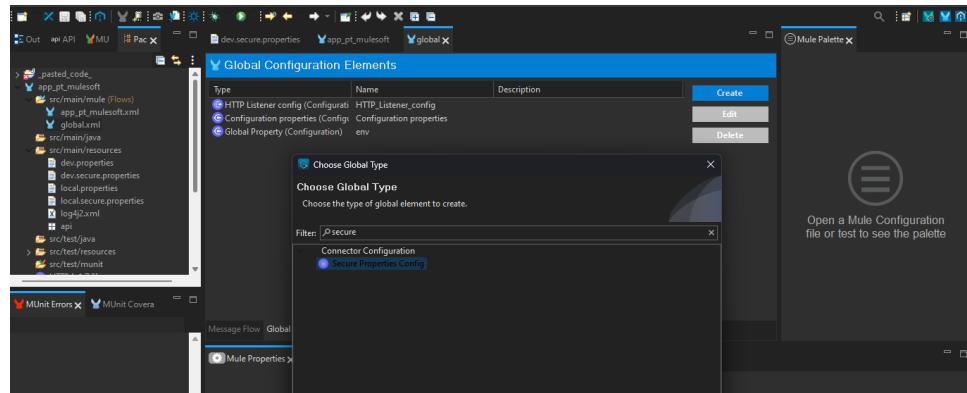
El siguiente paso es buscar en Exchange el módulo **Mule Secure Configuration**. En el archivo **global.xml**, se selecciona la opción **Search in Exchange** de la paleta Mule. Se busca el módulo y se hace clic en **Add >**



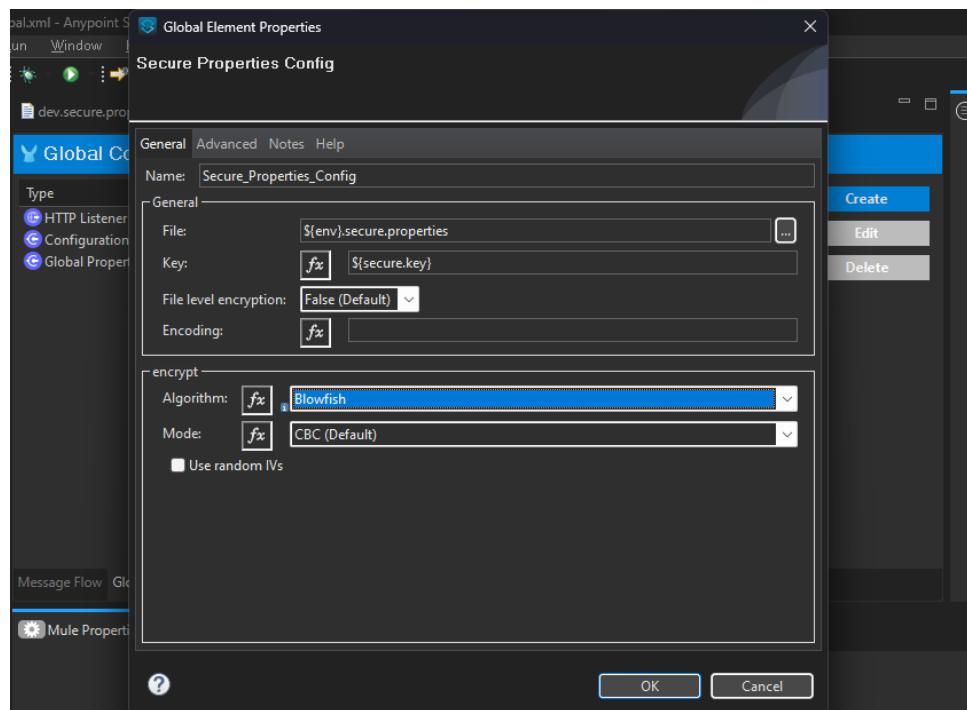


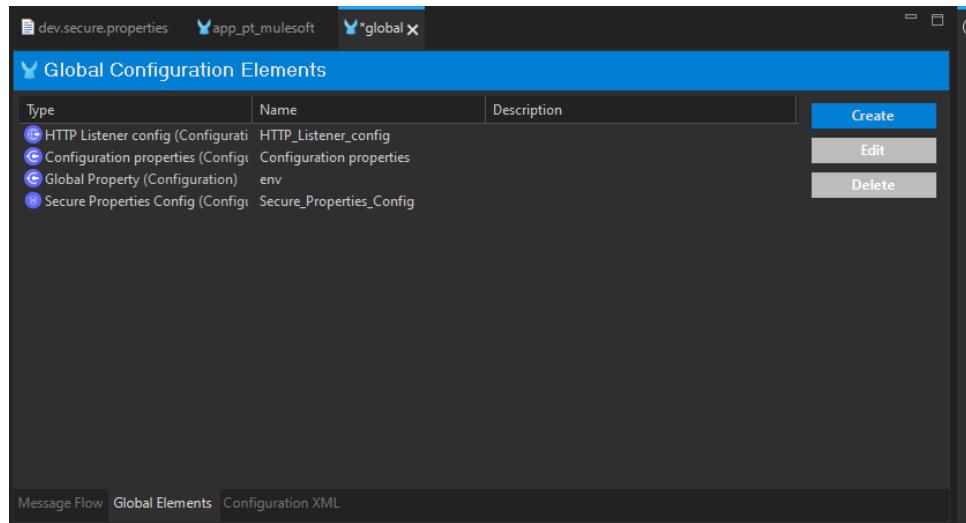
Hecho lo anterior se importa el módulo Secure properties, en la vista **Global Elements**. Se hace clic en el botón **Create** y se crea una **configuración de Secure Properties**. En el archivo, se escribe \${env}.secure.properties. Este es un archivo dinámico.

La aplicación Mule leerá primero la propiedad **env** y luego el archivo apropiado según ella. En este caso, **local.secure.properties**.

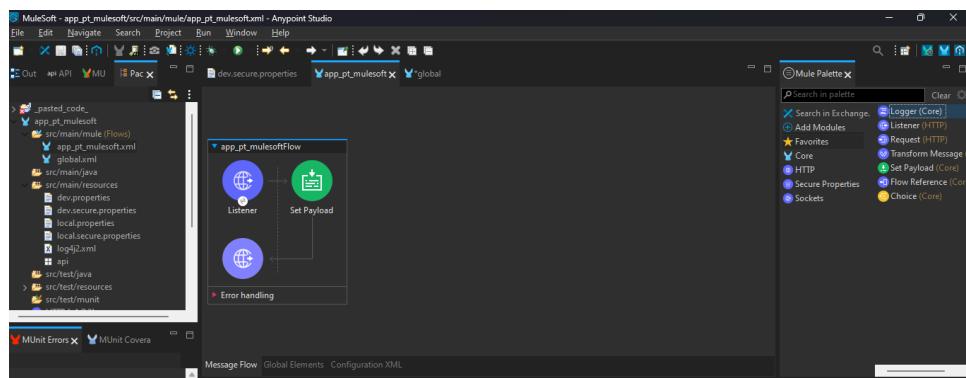


En Clave, se escribe `${secure.key}` y se selecciona **Blowfish** como Algoritmo.

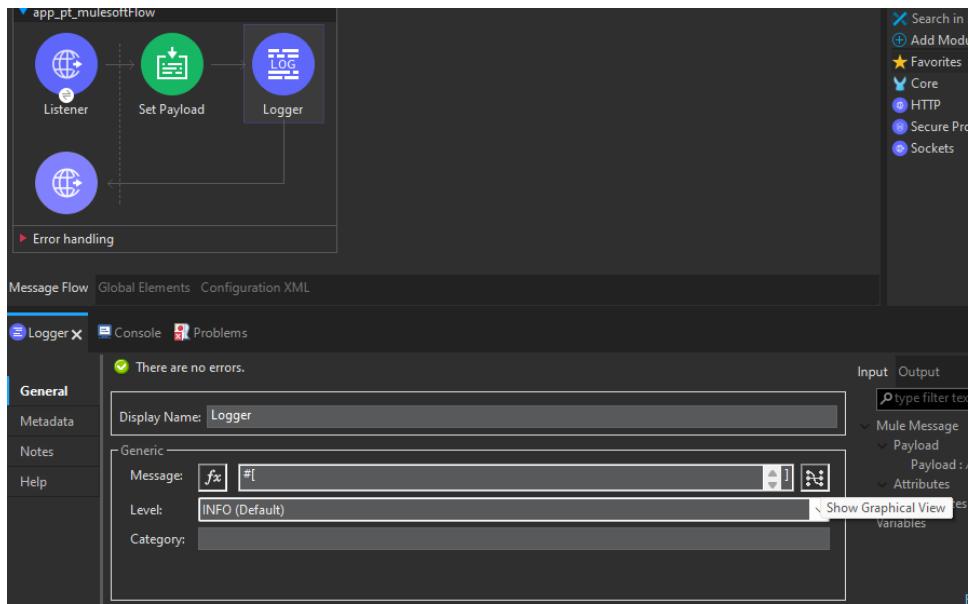




Hecho lo anterior, se habrá creado el elemento Secure Properties, por lo que se deberá guardar y posteriormente se creará un registro para leer las credenciales seguras. Para este paso se configura un componente de registro para enviar las credenciales a la consola. En el archivo **app_pt_mulesoft.xml** se agrega un componente **Logger** desde **Mule Palette**; se puede encontrar en el módulo **Core**.



En la configuración, se hace clic en el botón **fx** y luego en el botón **Show Graphical View** a la derecha. Esto abrirá una vista más grande para agregar el código DataWeave.



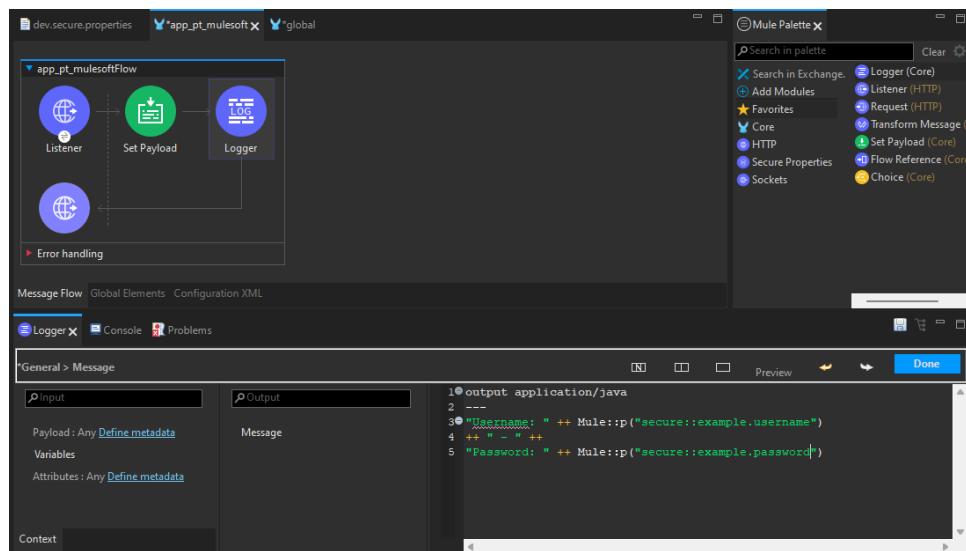
Se agrega el siguiente código para generar una Cadena con las dos propiedades seguras:

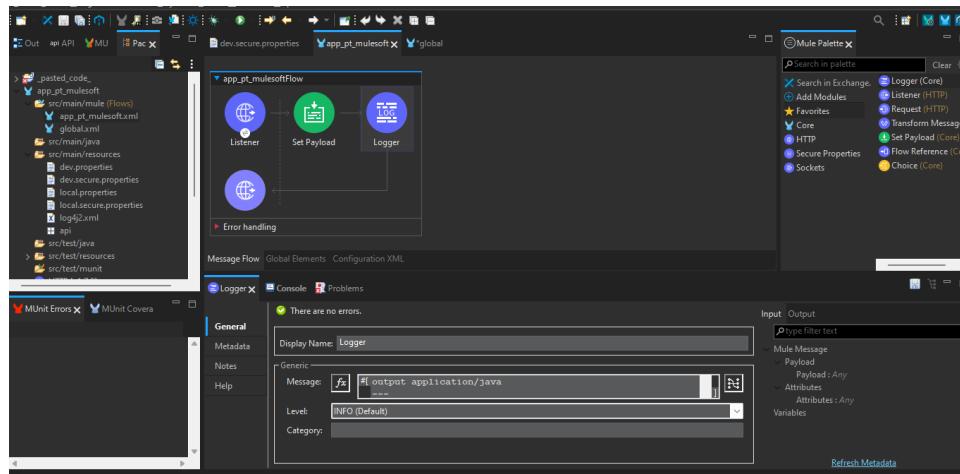
```
output application/java
```

```
"Username: " ++ Mule:::p("secure::example.username")
```

```
++ " - " ++
```

```
"Password: " ++ Mule:::p("secure::example.password")
```





Hecho lo anterior, ahora se debe cifrar los valores de los archivos de propiedades, por lo que en la página oficial de MuleSoft se puede encontrar algunos ejemplos para la configuración de los mismos.

Dentro de la página, se deberá descargar el archivo **Secure properties tool Jar File**.

Encrypt Properties Using the Secure Properties Tool

Use the Secure Properties Tool to encrypt or decrypt text strings, values inside a properties file, or all the contents of a properties file.

Ensure that you have the latest available version of this tool to use all the functions detailed in this article.

Download link: [Secure Properties Tool Jar file](#)
Latest release: 09/14/2022.

Note that the syntax has changed from earlier versions. To avoid issues, use the syntax and the JAR file provided in this article.

See [Parameters Reference](#) for a complete list and definition of all accepted parameters.

Hecho lo anterior, se debe copiar el siguiente código que nos proporciona MuleSoft, con el fin de lograr cifrar los valores respectivos a Username y Password de los entornos dev y env. **EL SIGUIENTE CÓDIGO DEBE EJECUTARSE EN UNA TERMINAL QUE ESTE UBICADA EN EL ARCHIVO JAR:**

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

The screenshot shows the Mule Runtime documentation for version 4.5.1. The page is titled "Secure Configuration Properties". It contains several sections: "Performance Tuning Guide", "Run Mule Runtime Engine On-Premises", "Mule Upgrade Tool", "Maven Support in Mule", "Security", and "Secure Configuration Properties". Under "Secure Configuration Properties", there are links to "Cryptography Module", "FIPS 140-2 Compliance Support", "Configure LDAP Provider for Spring Security", "Component Authorization Using Spring Security", "TLS Configuration", "OAuth Authorization Grant Types", and "Mule Secure Token Service". The main content area shows code snippets for encrypting strings using the `SecurePropertiesTool`:

```

<key> \
<value> \
--use-random-iv [optional]

```

Set `<method>` to `string` to configure the tool to process a text string.
Specify the other parameters to perform your desired operation.

For example, if you run:

```

java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool \
string \
encrypt \
Blowfish \
CBC \
mulesoft \
Some value to encrypt

```

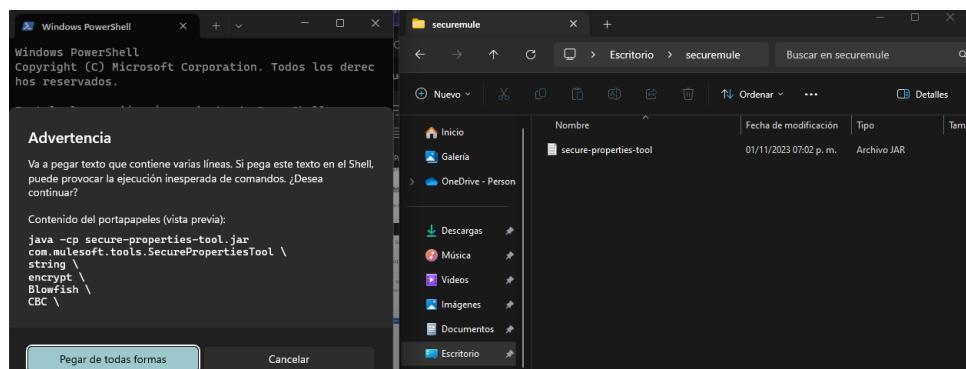
The tool returns:

```

8q5e1+jy0chD21V2uPThahmz6XsDuB6Z

```

Se Abre una terminal o línea de comandos en el directorio donde se encuentra este archivo Jar y ejecute el anterior código.



Esto devolverá el valor cifrado de **myUsernameLocal**

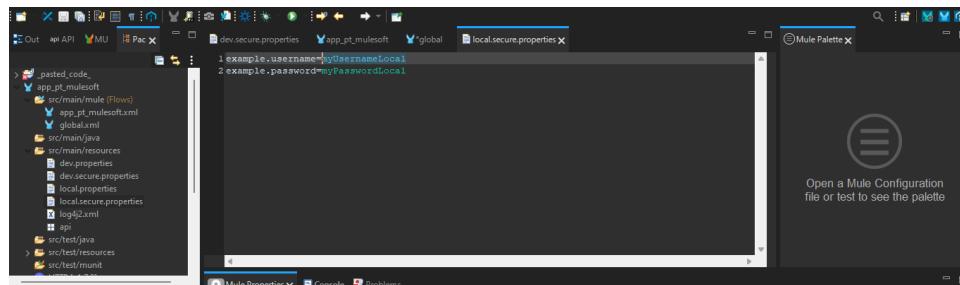
```

es-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfis
h CBC MyMuleSoftKey "myUsernameLocal"
HbsuWJRjiubchmzQREGdsA==

```

Después de obtener el valor cifrado, se debe agregar la siguiente sintaxis en los archivos de propiedades:

![encryptedValue]



Se repite para cada propiedad en local.secure.properties y dev.secure.properties. Estos se cifran de la siguiente manera y se asignan:

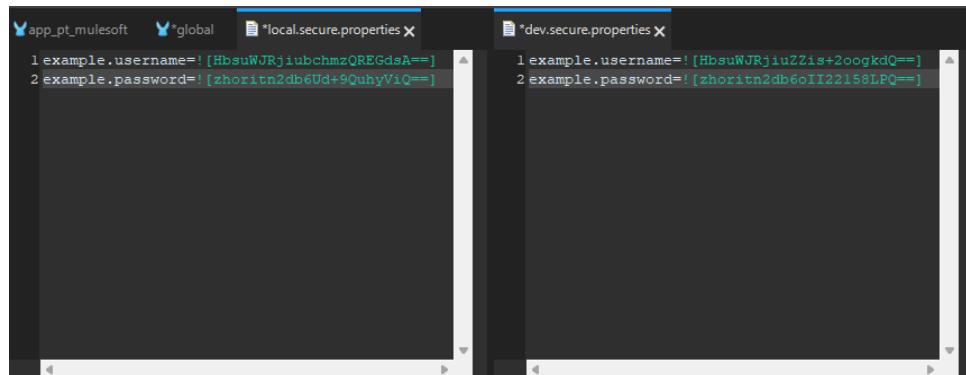
```

es-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish
h CBC MyMuleSoftKey "myPasswordLocal"
zhoritn2db6Ud+9QuhyViQ==

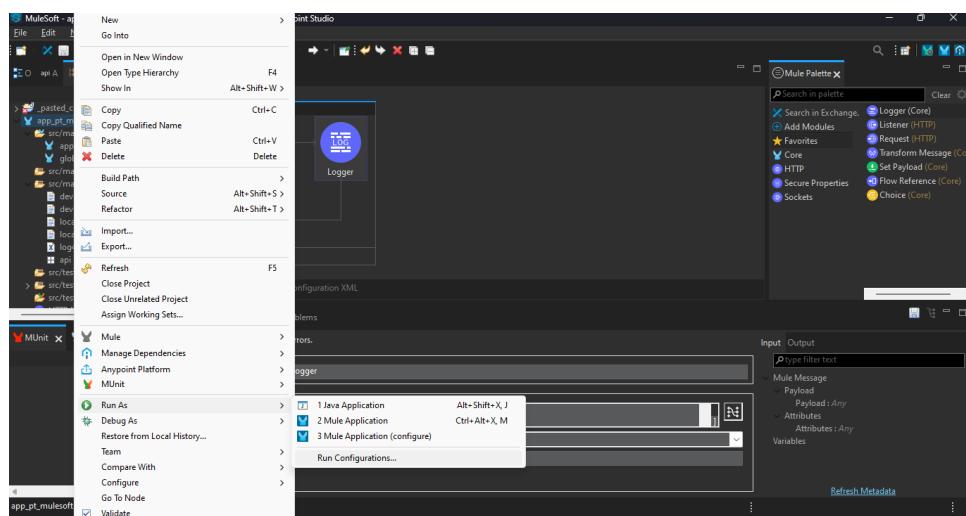
es-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish
h CBC MyMuleSoftKey "myUsernameDev"
HbsuWJRjiuZZis+2oogkdQ==

es-tool.jar com.mulesoft.tools.SecurePropertiesTool string encrypt Blowfish
h CBC MyMuleSoftKey "myPasswordDev"
zhoritn2db6oII22158LPQ==

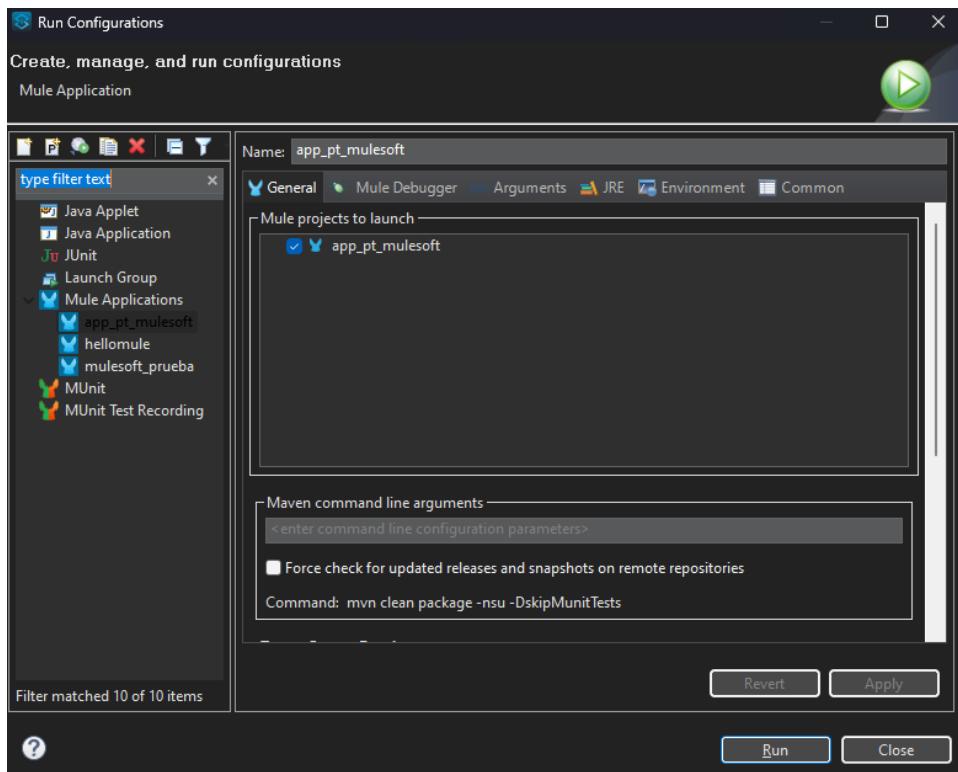
```



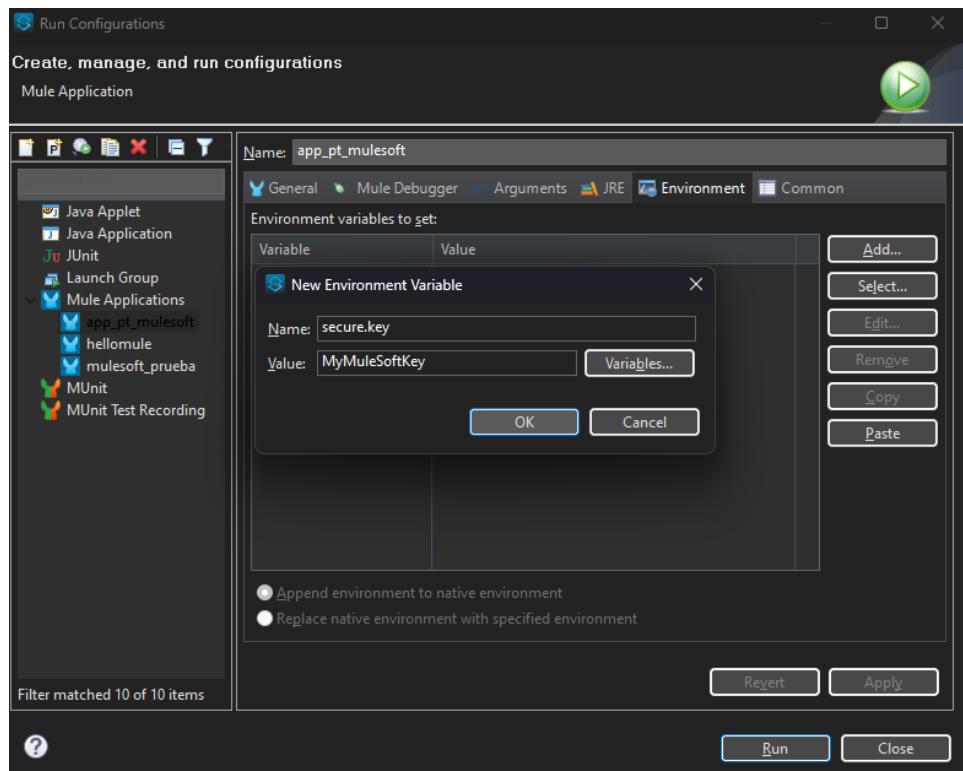
Hecho lo anterior, se hace la prueba de manera local, para que verificar que todo sea correcto, por lo que se hace clic derecho en su proyecto Mule y seleccione **Run as > Run Configurations**

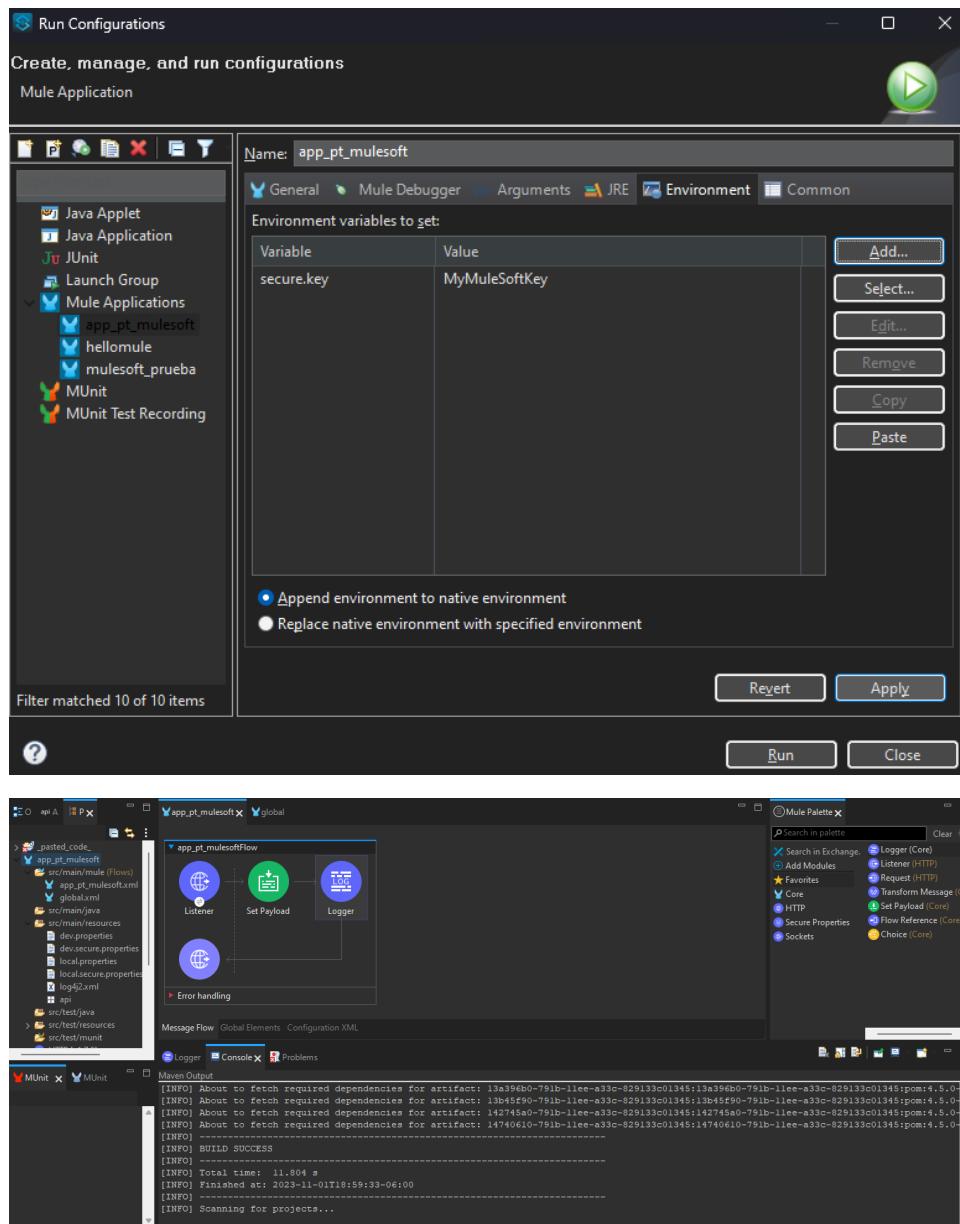


Muestra una venta en donde se dirige a la pestaña Enviroments y se hace clic en **Add**. Se introduce lo siguiente y se hace clic en **OK**:

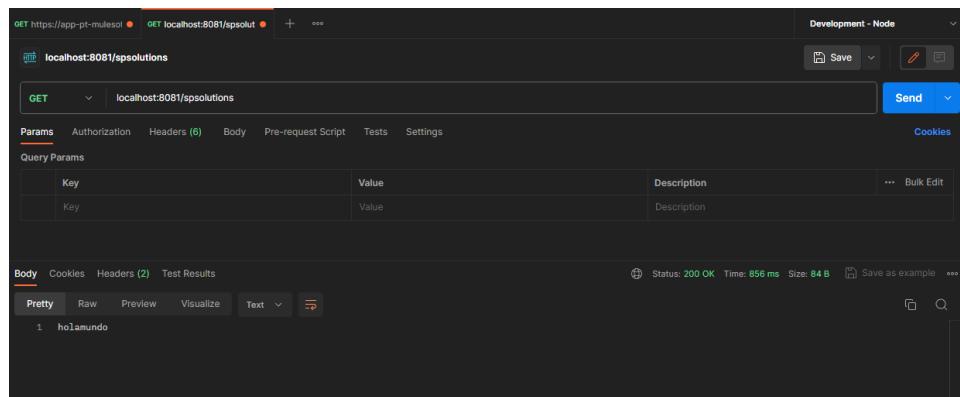


Una vez configurado, Se hace clic en **Apply** y **Run**. Esto enviará la propiedad **Secure.key** en tiempo de ejecución. Se hace esto para evitar violaciones de seguridad si alguien tiene acceso al código fuente de nuestra aplicación.





Desplegado el proyecto, es hora de realizar una petición, con el fin de verificar que la aplicación esté funcionando.

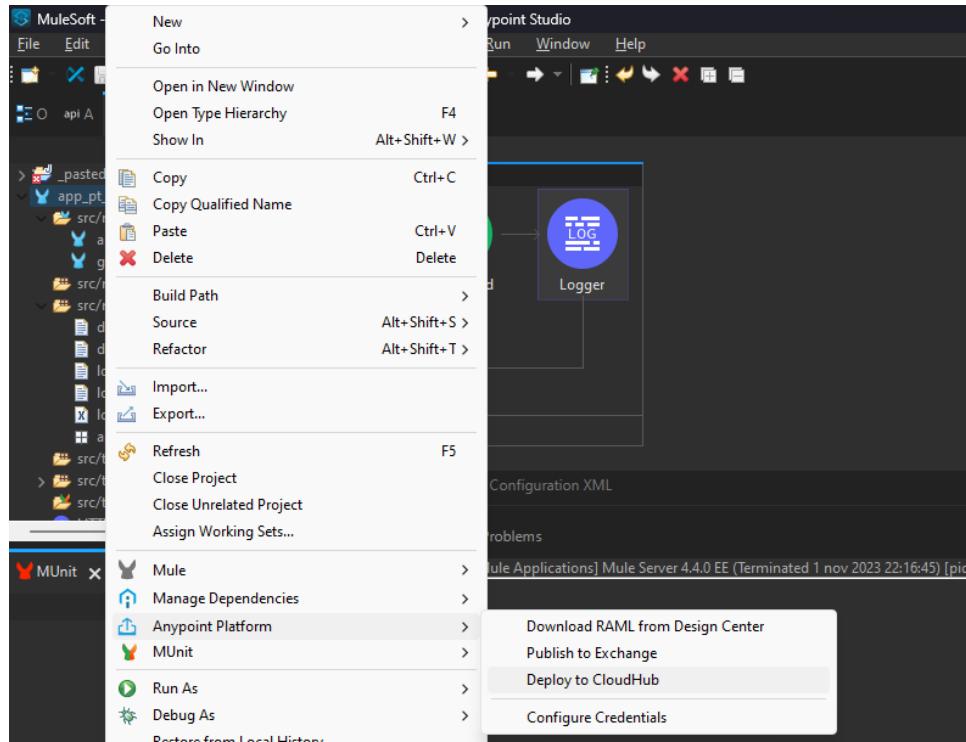


Después de verificar la petición, también se debe verificar que los valores cifrados se muestren por consola.

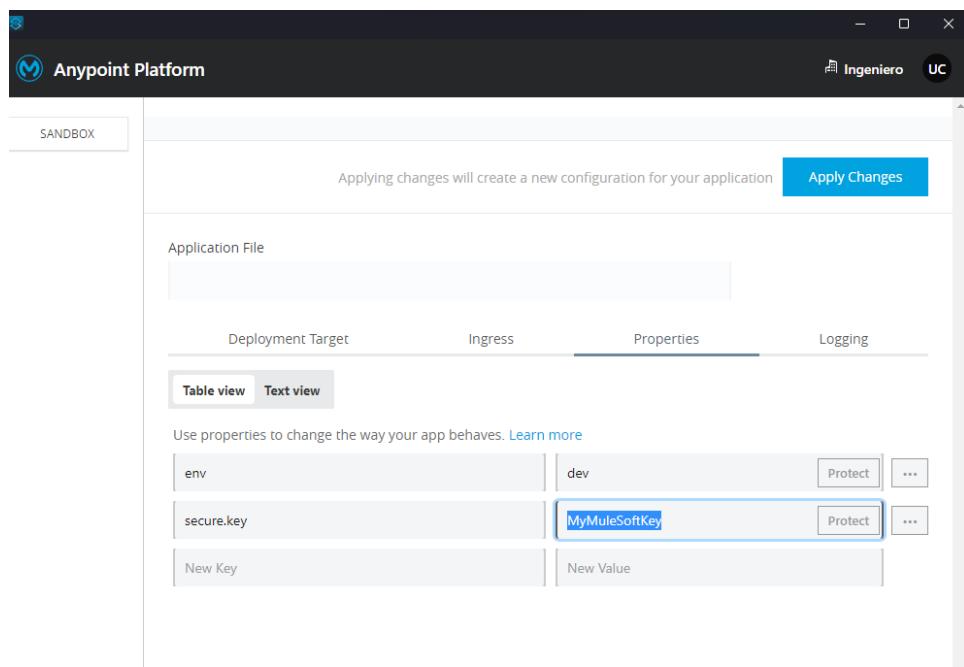
The screenshot shows the Anypoint Studio interface with the 'Logger' tab selected. In the bottom right corner of the main workspace, there is a small log entry window displaying the following message:

```
internal.runtime.source.ExtensionMessageSource: Message source 'listener' on flow 'app_pt_mulesoftFlow' successfully started
loggerMessageProcessor: Username: myUsernameLocal - Password: myPasswordLocal
```

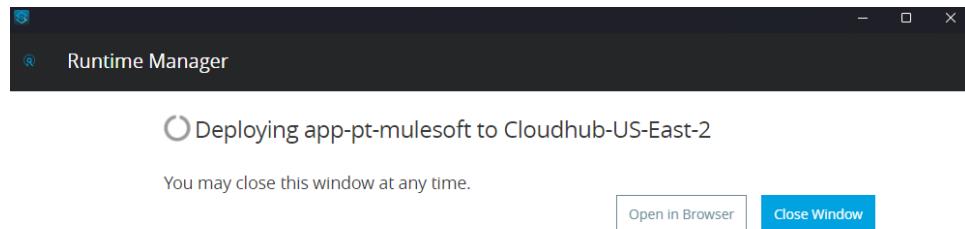
Hecho lo anterior, se debe desplegar en CloudHub.



Es importante que antes de implementar la aplicación en CloudHub, se agregue la **secure.key** y los valores **env** a las propiedades de implementación.



En la plataforma de MuleSoft se debe ir a la pestaña de **Runtime Manager**. Se seleccione el entorno **Sandbox** y se hace clic en la aplicación previamente implementada. Ahora, en la pestaña **Properties** se agregua las dos propiedades. Esta vez la propiedad **env** debe establecerse en **dev**. Posteriormente se hace clic en Aplicar cambios.



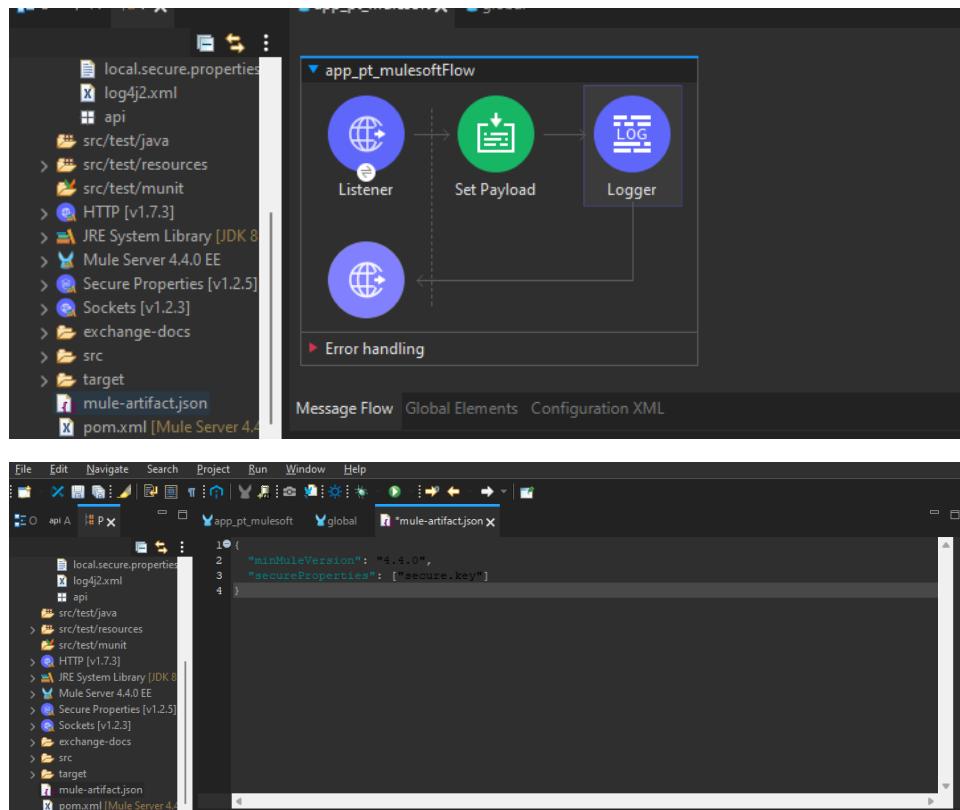
The screenshot shows the 'Runtime Manager' application list. The sidebar includes options like Applications, Servers, Flex Gateways, Alerts, VPCs, Private Spaces, and Load Balancers. The main area shows a table for 'All Applications (1)'. The table has columns: Name, Target Name, Target Type, Status, Runtime Version, and Date Modified. The single entry is 'app-pt-mulesoft' with 'Shared Space' as the target name, 'CloudHub 2.0' as the target type, 'Running' status, '4.5.0.23e' runtime version, and the date '2023-11-01 22:23:58'.

Se observa cómo el valor de Secure.key no está oculto actualmente.

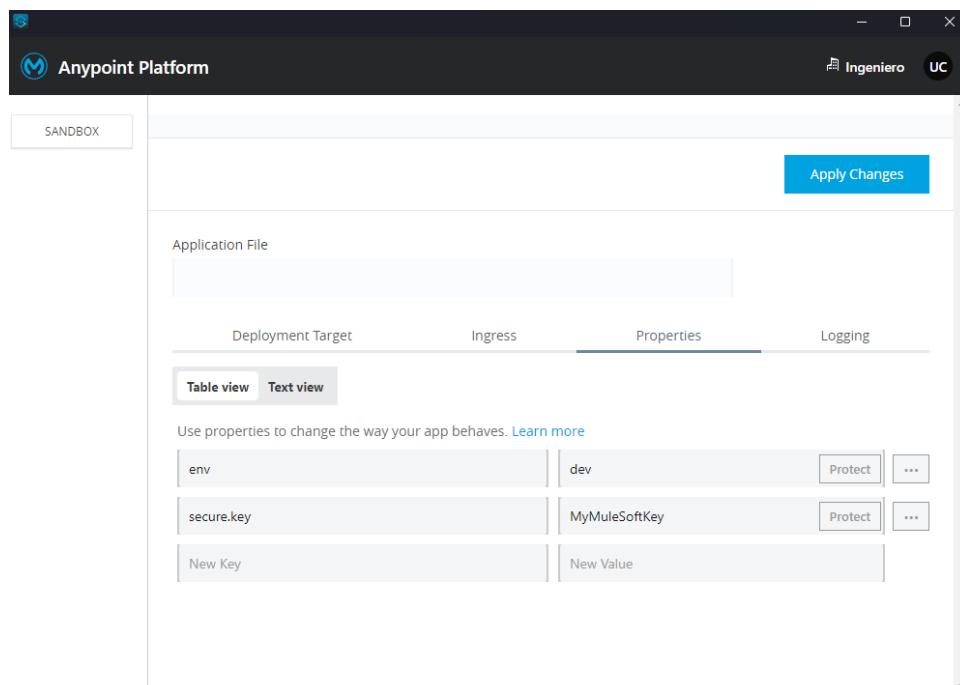
The screenshot shows the 'Runtime Manager' properties configuration window for the application 'app-pt-mulesoft'. The left sidebar lists 'Applications', 'Dashboard', 'Diagnostics', 'Logs', 'Object Store', 'Schedules', and 'Settings'. The main area shows the 'Properties' tab for 'Configuration 0944d8'. It lists properties: 'env' (value 'dev'), 'secure.key' (value 'MyMuleSoftKey'), and 'New Key' (value 'New Value').

Para ocultarlo, se va al archivo **mule-artifact.json** en Anypoint Studio y se agrega la siguiente línea antes de los corchetes de cierre.

```
"secureProperties": ["secure.key"]
```



Hecho lo anterior se guarda todos sus archivos e implemente la aplicación Mule en CloudHub seleccionando **Anypoint Platform > CloudHub**.



Ahora se espera a que los cambios se apliquen desde la pagina de MuleSoft.

The screenshot shows the Runtime Manager interface. On the left, there's a sidebar with options like SANDBOX, Applications, Servers, Flex Gateways (NEW), Alerts, VPCs, Private Spaces (NEW), and Load Balancers. The main area has a search bar for 'Search Applications' and a table titled 'All Applications (1)'. The table columns are Name, Target Name, Target Type, Status, Runtime Version, and Date Modified. One row is listed: 'app-pt-mulesoft' with 'Shared Space' as the target name, 'CloudHub 2.0' as the target type, 'Running' status, '4.5.0.23e' runtime version, and a modified date of '2023-11-01 22:31:23'.

Aplicados los cambios, se muestra que la secure.key ahora se encuentra oculta.

This screenshot shows the 'Properties' tab of an application configuration. It has tabs for 'Runtime' and 'Properties', with 'Properties' selected. Below are two tabs: 'Table view' (selected) and 'Text view'. The table view shows three properties: 'env' with value 'dev', 'secure.key' with value '*****', and 'key' with value 'value'.

Posteriormente, se debe verificar que la aplicación este funcionando de manera correcta, por lo que en Postman, se realiza una petición y se espera la respuesta de estatus 200

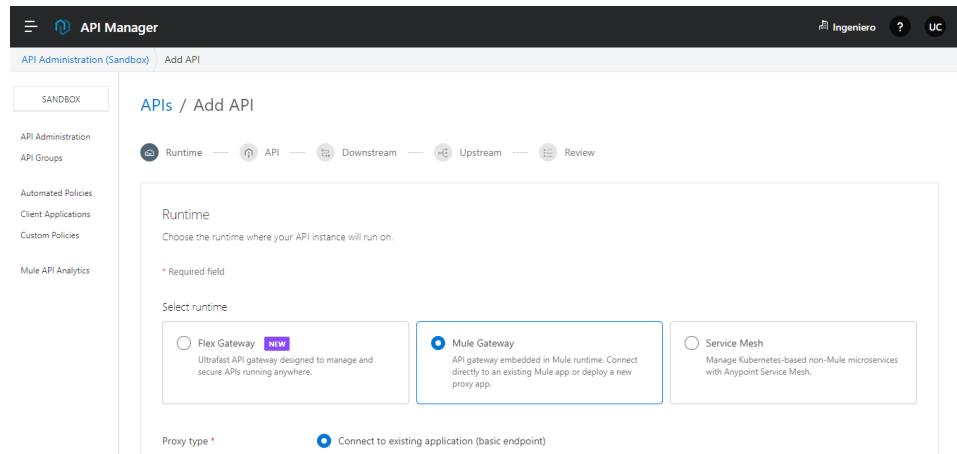
The screenshot shows a Postman request for 'https://app-pt-mulesoft-qafrxc.5sc8y6-2.usa-e2.cloudhub.io/spsolutions'. The method is 'GET'. In the 'Params' section, there is a 'Query Params' table with one entry: 'Key' and 'Value' both are 'holamundo'. The 'Body' tab is selected, showing the response body: 'holamundo'. The status bar at the bottom indicates 'Status: 200 OK'.

How to set up API Autodiscovery in Anypoint Studio. Crear una API en API Manager para permitir aplicar politicas.

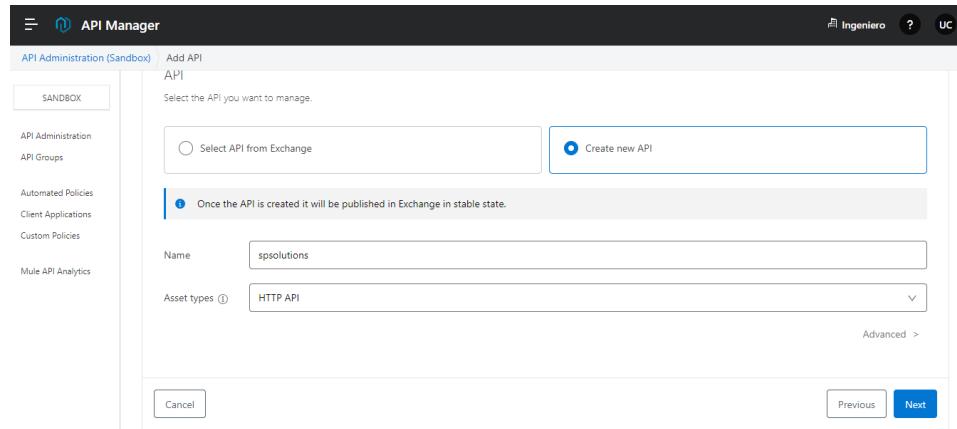
El primer paso para administrar una API y habilitar políticas de seguridad es crear una nueva API en API Manager.

Se selecciona **API Administration > Add New API**.

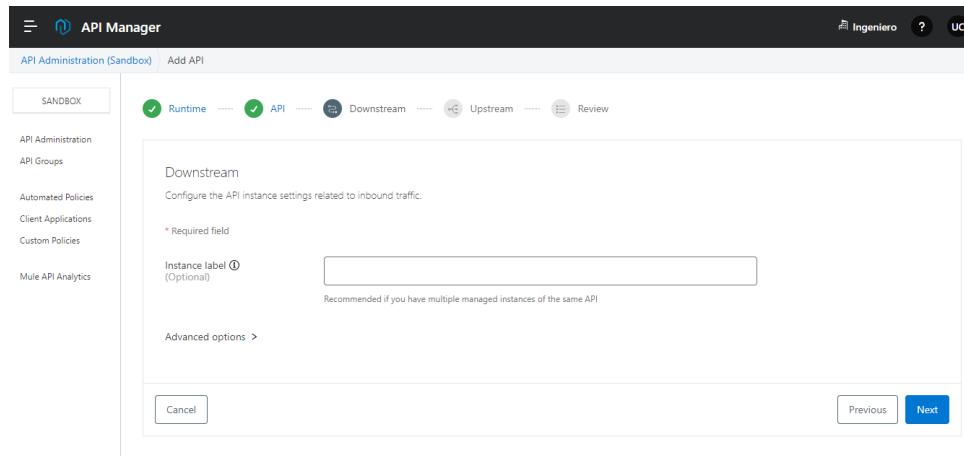
Posteriormente se selecciona la opcion **Mule Gateway** y se da en siguiente.



Se selecciona la opción **Create new API > Next**



En las siguientes opciones son opcionales, por lo que se da en **Next**



Al crear la nueva API, se le otorgará un **ID de API de detección automática de API**. Se ingresa el ID de API de detección automática en Anypoint Studio para vincular la aplicación mula con la API en API Manager. Por ahora, simplemente se toma nota del **API Instance ID**.

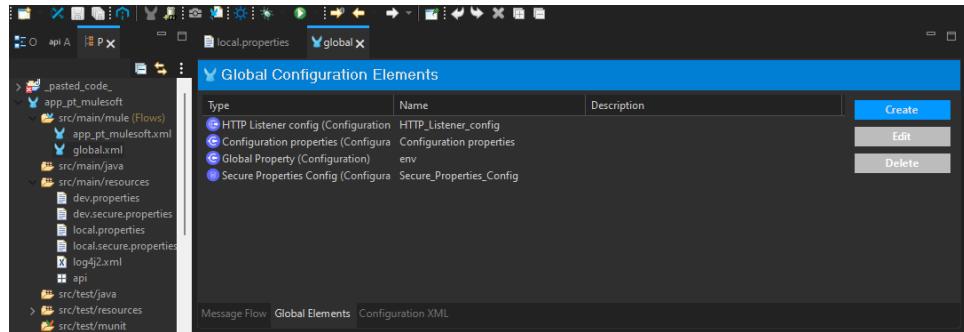
Ahora se deberá crear una nueva propiedad **api.id** por entorno dev – local.

Siempre es una buena práctica mantener las propiedades en archivos externos en lugar de codificar los valores en los componentes. En el proyecto Anypoint Studio, se abre los archivos **local.properties** y **dev.properties** en **src/main/resources**. Se agrega una nueva propiedad **api.id** en ambos archivos con el valor de **API Instance ID** que se obtuvo del **API Administration**.

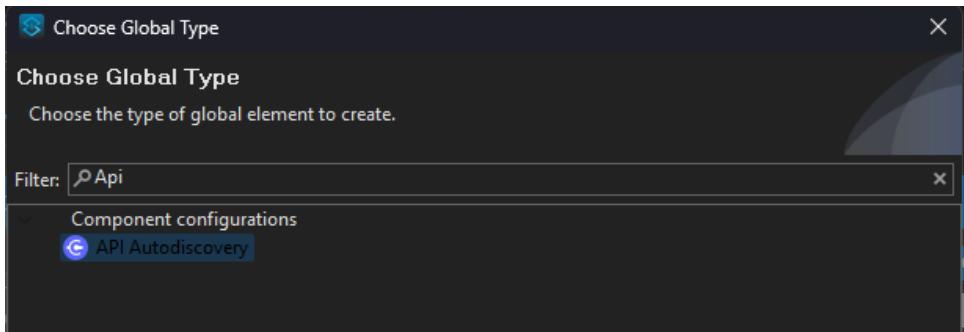
```
*local.properties*
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
3
4 api.id=19084083

app_pt_mulesoft  global  *dev.properties*
1 http.listener.host=0.0.0.0
2 http.listener.port=8081
3
4 api.id=19084083
```

Posteriormente en **global.xml** y se cambia a la vista **Global Elements**. Y se hace clic en el botón **Create**.



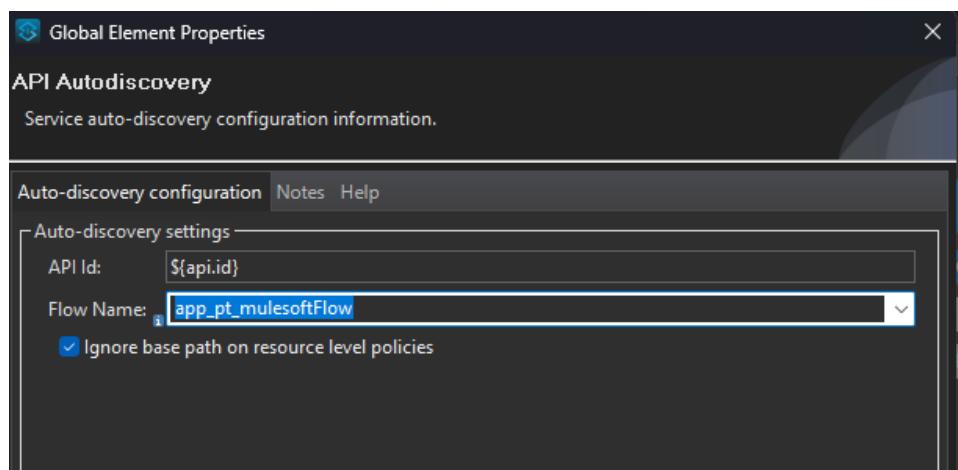
Posteriormente se selecciona **API Autodiscovery**.



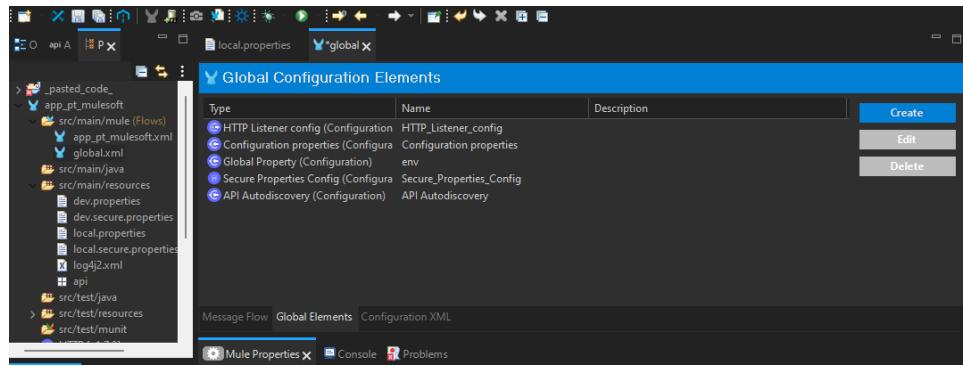
Para vincular el campo ID de Api a su nueva propiedad, simplemente se hace referencia a él usando la siguiente sintaxis:

`${api.id}.`

Para el nombre del flujo, puede seleccionar **app_pt_mulesoftFlow**.



Al realizar lo ante se crea un nuevo campo global en el archivo **Global.xml**



Antes de ejecutar la aplicación, se necesita agregar dos valores más para poder conectarnos a nuestra API Administration. Se deberá regresar a Anypoint Platform y en la opción **Access Management**. Dentro se tendrá que seleccionar **Business Groups** y se selecciona el entorno **Ingeniero**.

Name	Environments	Total vCores
Ingeniero	2	2

Posteriormente se abrirá una ventana con diferentes opciones, en la cual se selecciona **Environments > Sandbox**

Name	Type	Default client provider
Design	Design	Anypoint
Sandbox	Sandbox	Anypoint

<https://anypoint.mulesoft.com/accounts/businessGroups/99b39c0f-3771-4a5c-be18-bc59fe491dd9/environments/00c6caa0-07d2-4f22-b724-8e0764f1d1e6/edit>

Al ingresar se mostrarán los valores de:

Client ID: 598e96fe15724e2687953931cdec4c6d

Client Secret: E762D3a01c6147478656fCD0897Aad37

Necesarios para establecer la conexión

Edit environment

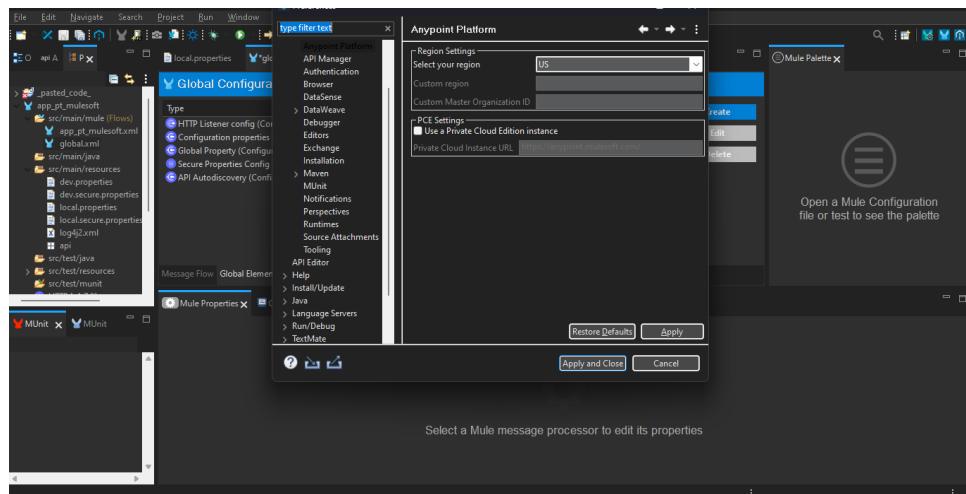
Name
Sandbox

Client ID
598e96fe15724e2687953931cdec4c6d

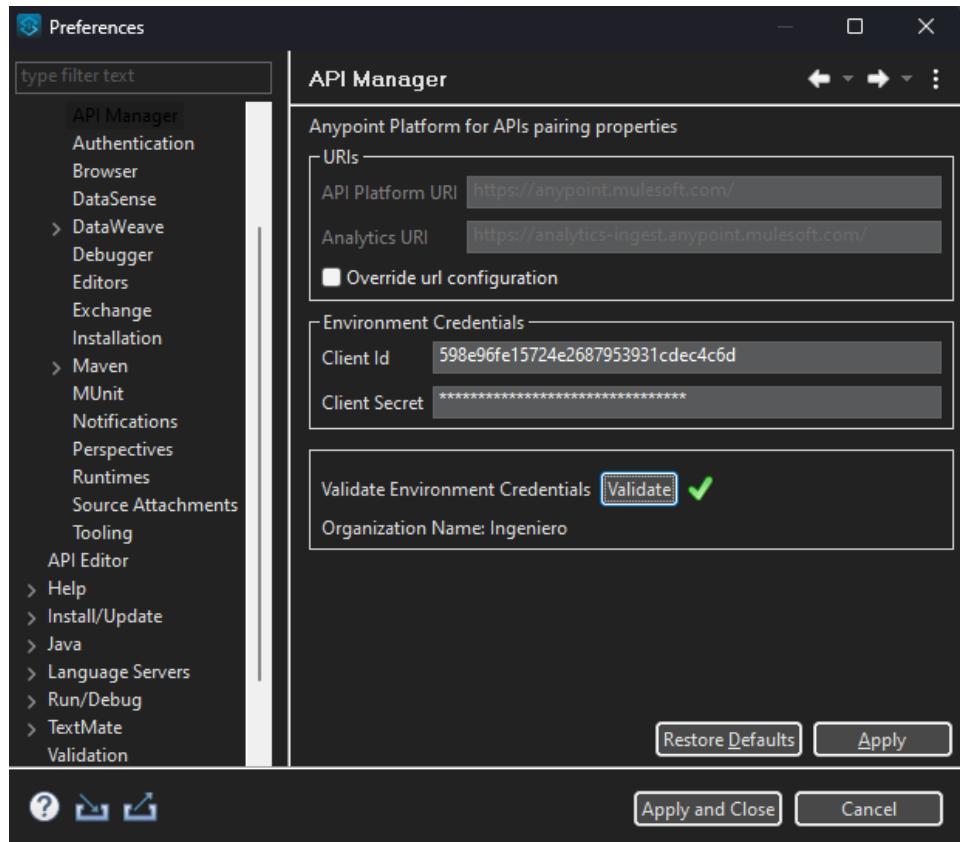
Client Secret
E762D3a01c6147478656fCD0897Aad37 Hide

Delete Environment Cancel Update

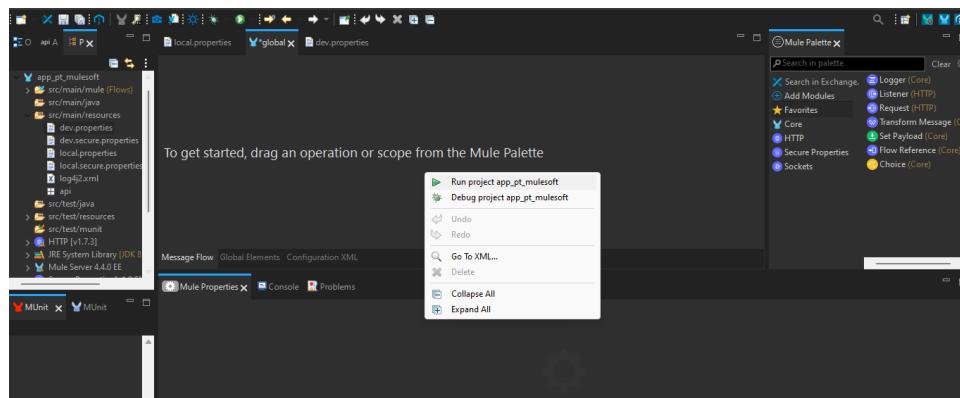
Para establecer la conexión se accedio a las preferencias o configuración de Anypoint Studio.



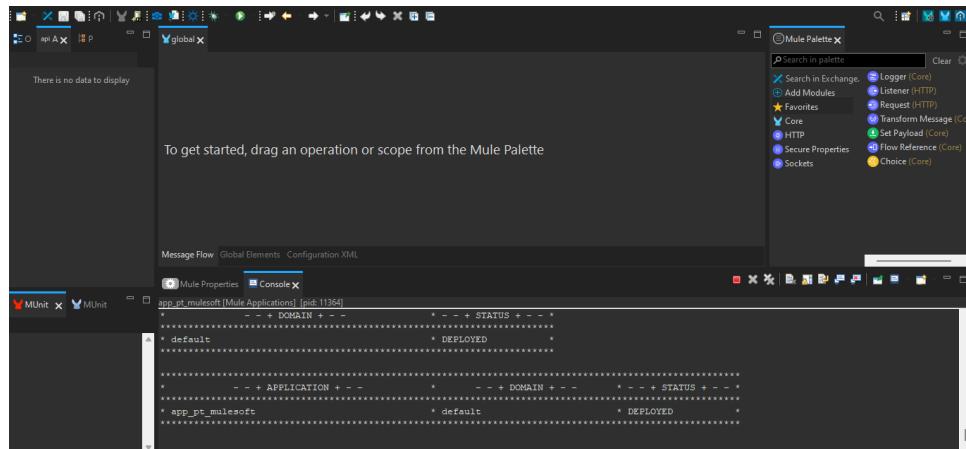
Se abrirá una ventana en donde se selecciona **Anypoint Studio > API Administration**. En la plataforma seleccionada se pegará las credenciales de Access Management (client Id y Client secret) y se hace clic en **Validate**. El nombre de la organización debe coincidir con el que tiene en la esquina superior derecha la cuenta de Anypoint Platform.



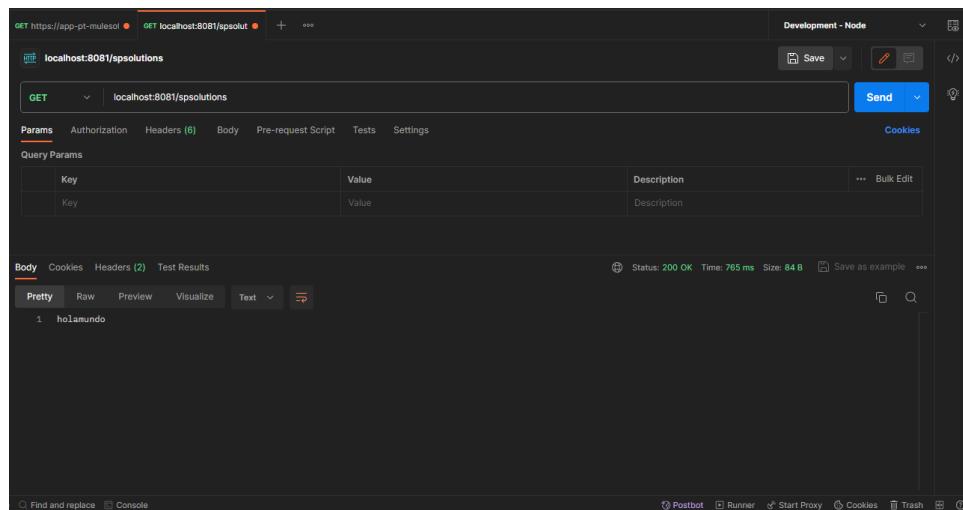
Una vez configurado, Se hace clic en **Apply and Close** y se ejecute la aplicación Mule.



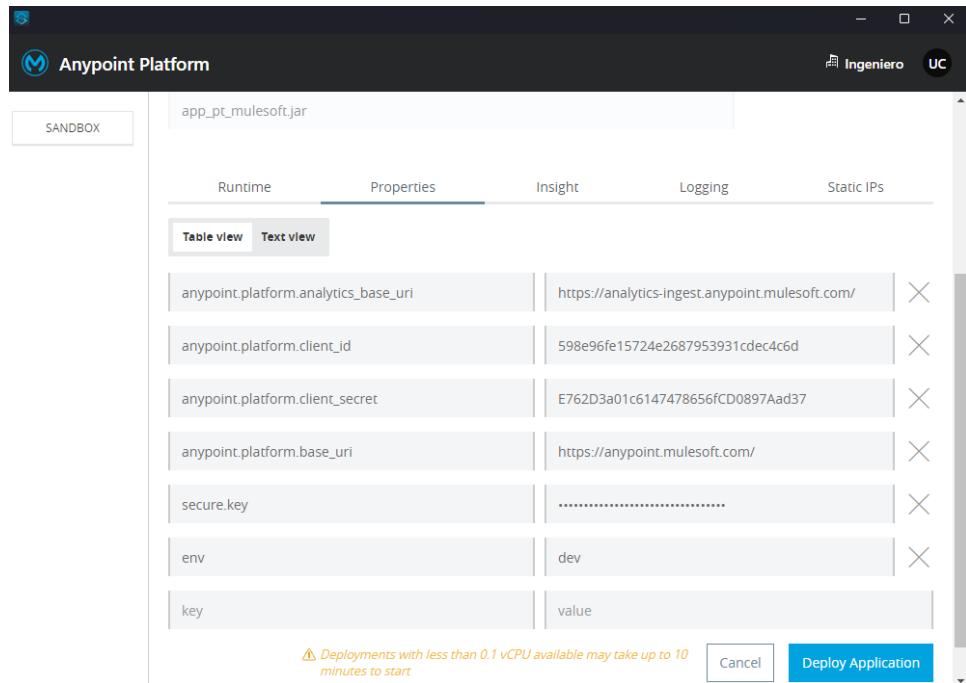
Se verifica que no haya errores en la Consola y que el Estado de la aplicación sea **DEPLOY**.



Para verificar que la aplicación siga funcionando correctamente se hace una petición en local en el programa Postman.



Verificado lo anterior, se implementa en CloudHub con las credenciales de entorno.



En Anypoint Studio, Se abre el archivo **mule-artifact.json** para agregar estas dos propiedades a la lista de propiedades seguras.

```
{
    "minMuleVersion": "4.3.0",
    "secureProperties": [
        "secure.key",
        "anypoint.platform.client_id",
        "anypoint.platform.client_secret"
    ]
}
```

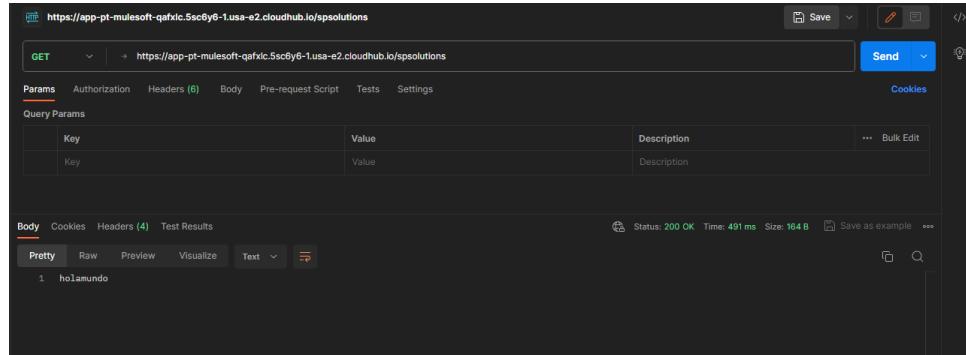
Esto ocultará ambas propiedades en nuestro Runtime Manager por razones de seguridad.

```
global *mule-artifact.json
1 {
2     "minMuleVersion": "4.4.0",
3     "secureProperties": [
4         "secure.key",
5         "anypoint.platform.client_id",
6         "anypoint.platform.client_secret"
7     ]
8 }
```

Implementado las propiedades necesarias, se hace el despliegue de la aplicación, y se guardan los cambios.

app-pt-mulesoft1 CloudHub CloudHub Updating

Por último, se realiza una prueba para verificar que el despliegue se haya hecho de manera correcta y la aplicación siga funcionando.



The screenshot shows the MuleSoft API Manager interface. On the left, there is a sidebar with options like API Summary, Alerts, Contracts, Policies, SLA Tiers, and Settings. The main area is titled "APIs / sspssolutions / API Summary". It displays the following details:

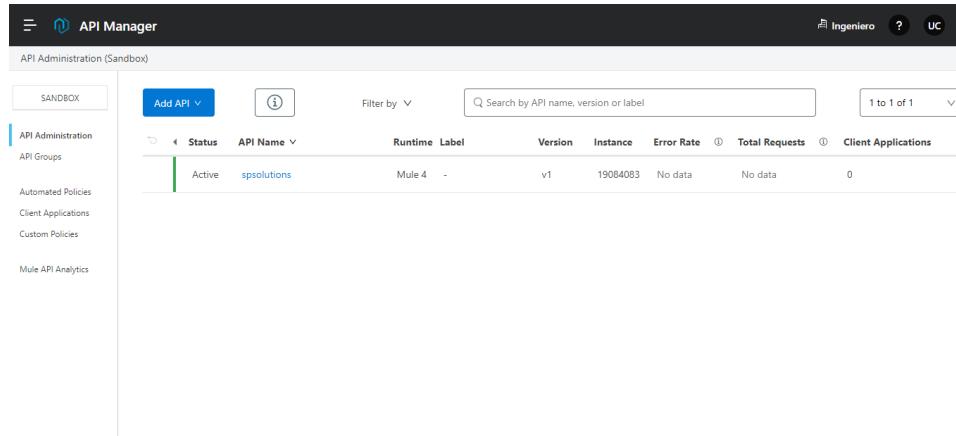
Type	Asset Version	Implementation URI
HTTP	1.0.0 (Latest)	N/A
API Label	API Version	API Status
-	v1	Active
Consumer Endpoint	API Instance ID	Mule Version
N/A	19084083	4.5.0

Below this, there is a "Key Metrics" section.

How to apply Client ID enforcement policy to your Mule app in API Manager.

Aplicar Client ID Enforcement policy a API Manager

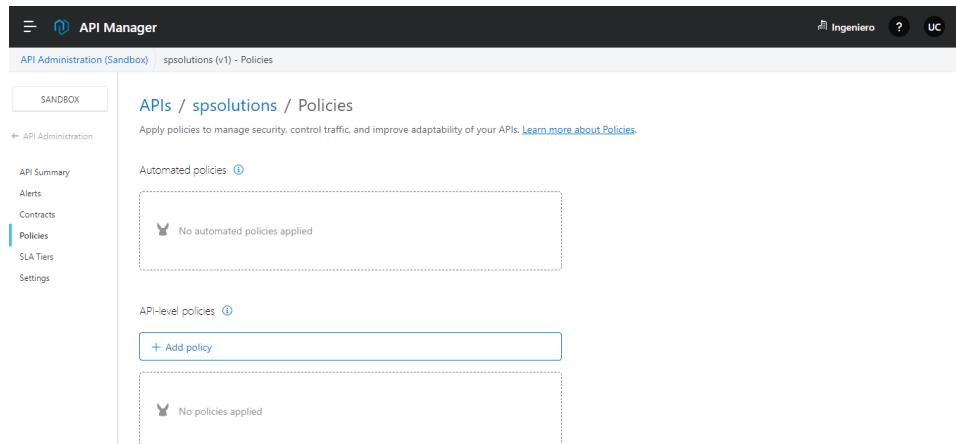
En la plataforma API Manager se Selecciona la API a la que desea aplicar la política. En este caso, la API **spsolutions**. A continuación, se hace clic en **Policies**



The screenshot shows the API Manager interface with the following details:

- Sandbox** tab is selected.
- Add API** button is visible.
- API Administration (Sandbox)** section is open.
- API Name** filter is set to **solutions**.
- Status**: Active.
- Runtime**: Mule 4.
- Label**: -
- Version**: v1.
- Instance**: 19084083.
- Error Rate**: No data.
- Total Requests**: No data.
- Client Applications**: 0.

Dentro de Policies se muestra una opción para agregar políticas respectivas para la API, se da clic en **Add Policy**.



The screenshot shows the Policies page for the 'solutions' API:

- APIs / solutions / Policies**
- Message: "Apply policies to manage security, control traffic, and improve adaptability of your APIs. [Learn more about Policies](#)".
- Automated policies**: "No automated policies applied".
- API-level policies**: "+ Add policy". "No policies applied".

Mostrará ventana que muestra diferentes opciones para agregar políticas de API, en este caso se da clic en **Client ID Enforcement**.

En la siguiente pantalla, se puede seleccionar cómo se desea que la API reciba las credenciales de Client ID y Client Secret. Se selecciona Expresión personalizada, puede cambiar el nombre de los campos, si serán encabezados, parámetros de consulta o incluso la carga útil de la solicitud. En este caso, se selecciona **HTTP Basic Authentication Header** y Se aplica la configuración **Apply configuration to all API**. Y se da clic en **Apply**.

Hecho lo anterior, las configuraciones aplicadas se mostrarán en la ventana de Policies.

Para verificar que las políticas se hayan aplicado de manera correcta, se usa Postman, al realizar una petición se observa que un mensaje de error en la petición:

```
{
  "error": "Authentication denied"
}
```

The screenshot shows a Postman interface with a GET request to the URL `app-pt-mulesoft1us-e2.cloudhub.io/spsolutions`. The response status is 401 Unauthorized. The response body is a JSON object with one key, "error": "Authentication denied.".

Esto significa que nuestra política de aplicación de ID de cliente recientemente agregada está funcionando.

Para poder realizar las peticiones se debe generar las credenciales de Client Id y Client Secret, para realizarlo se regresa a AnyPoint Studio en la pestaña de **Exchange**, Se observa que ya se tiene una API HTTP publicada en los activos de su organización.

The screenshot shows the AnyPoint Studio Exchange interface. In the left sidebar, under 'Ingeniero (root)', there is an 'HTTP API' named 'spsolutions'. The main area displays the details of this API, including its name, rating, and author.

Se da clic en HTTP API `spsolutions`. Esto abrirá la vista previa de la API. Se hace clic en **Request Access**.

The screenshot shows the 'spsolutions' API instance page. It includes options like 'Share', 'Request access', 'Add version', and 'Manage versions'. Below these, there are sections for 'PAGES' (Home) and 'OTHER DETAILS'.

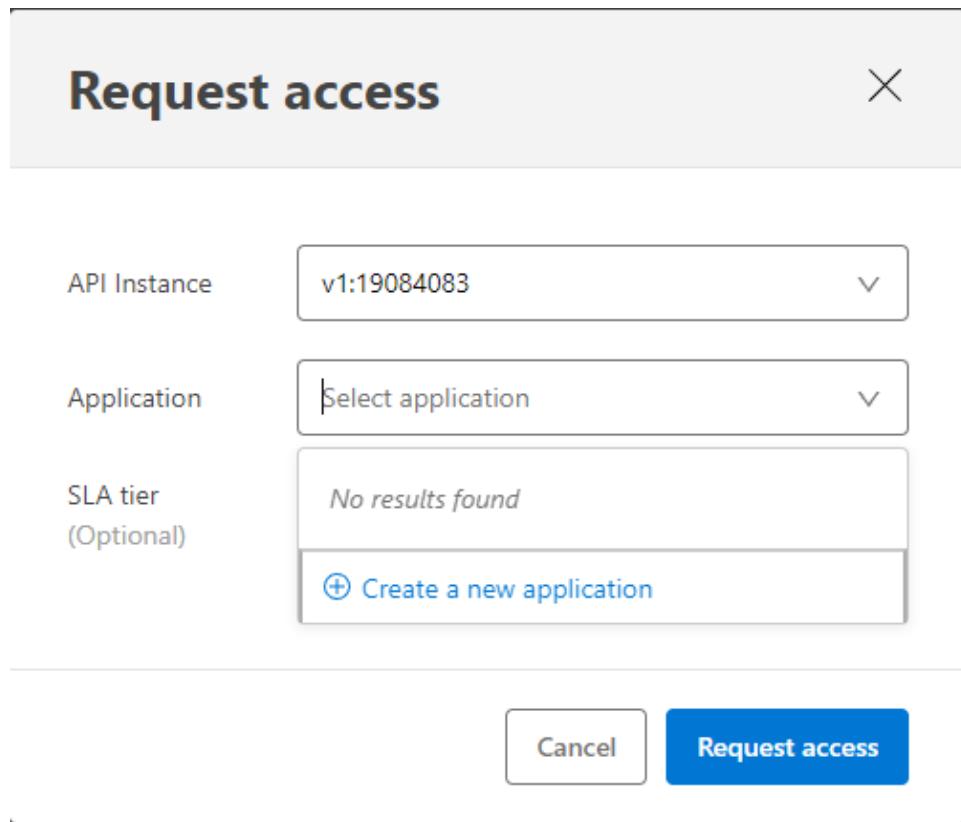
Seleccione la API Instance de la aplicación, después se selecciona la aplicación, en este caso no se muestra ninguna creada, pero da la opción de crear una nueva, por lo que selecciona **Create new application**.

Request access

X

API Instance	v1:19084083
Application	Select application
SLA tier (Optional)	No results found
	+ Create a new application

[Cancel](#) [Request access](#)

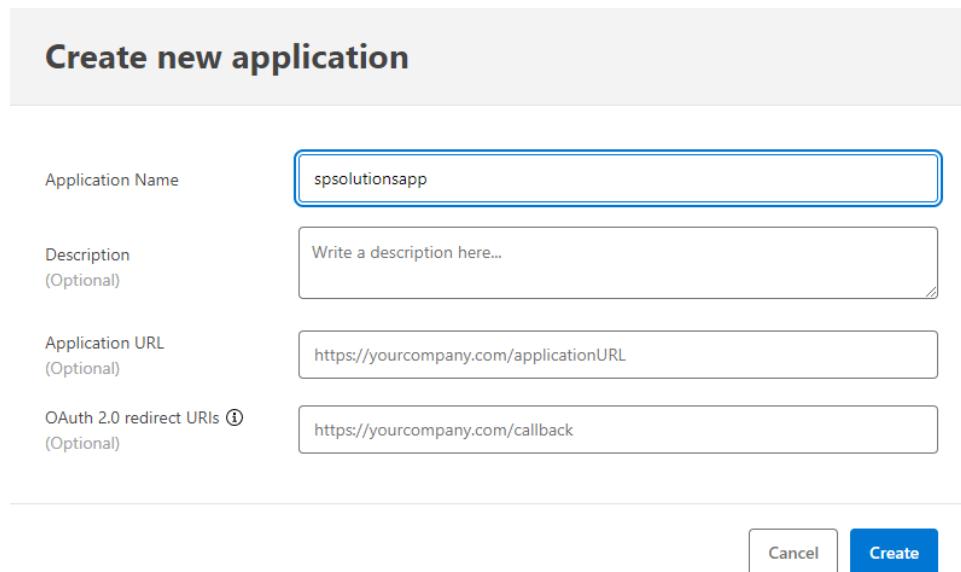


Muestra una venta con la descripción de la aplicación, pero en esta caso solo se colocará el nombre.

Create new application

Application Name	spsolutionsapp
Description (Optional)	Write a description here...
Application URL (Optional)	https://yourcompany.com/applicationURL
OAuth 2.0 redirect URIs <small>①</small> (Optional)	https://yourcompany.com/callback

[Cancel](#) [Create](#)



Realizados los pasos anteriores, solo queda crear el **Request access**.

Request access

X

API Instance	v1:19084083
Application	spsolutionsapp
SLA tier (Optional)	Instance does not have SLA tiers

[Cancel](#) [Request access](#)

Ahora se muestra el Client ID y Client Secret. También se verá un nuevo enlace donde podrá recuperar las credenciales en cualquier momento que se necesite.

Request API access

X

Your request has been received and approved.

Use this client ID and client secret to access the requested API instance. You can always find these values at the [application page](#).

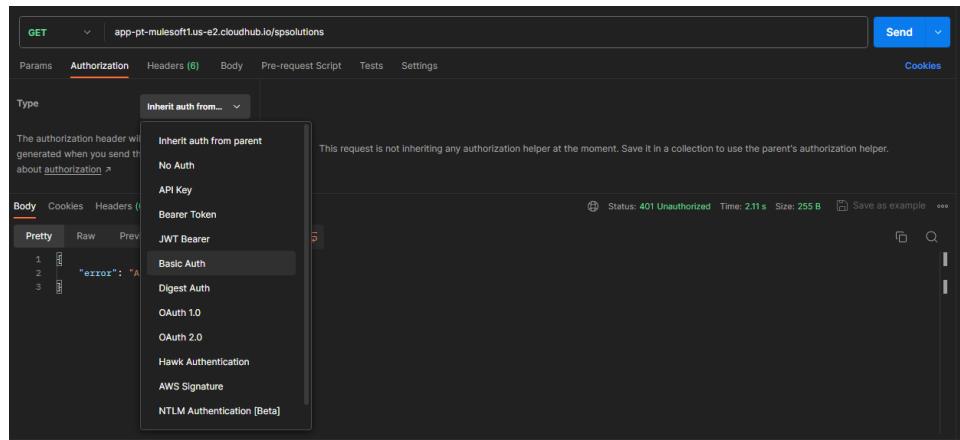
Client ID: 2e390a2b982f4419ab5dd5950c6ffe4d
Client Secret: 0210C892572b4122A5F993DB6bA53828

[Close](#)

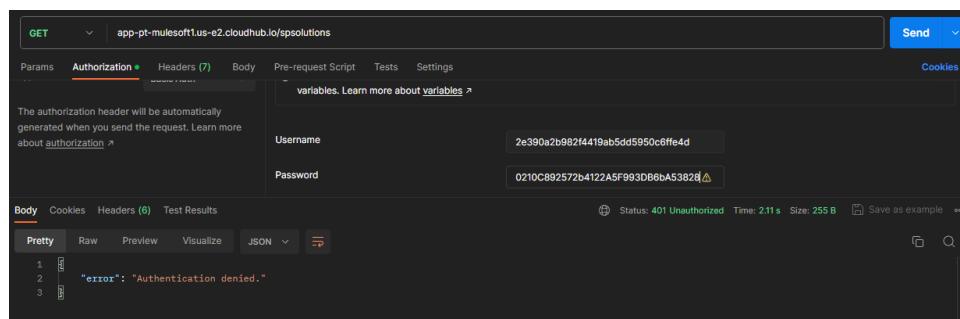
Client ID: 2e390a2b982f4419ab5dd5950c6ffe4d

Client Secret: 0210C892572b4122A5F993DB6bA53828

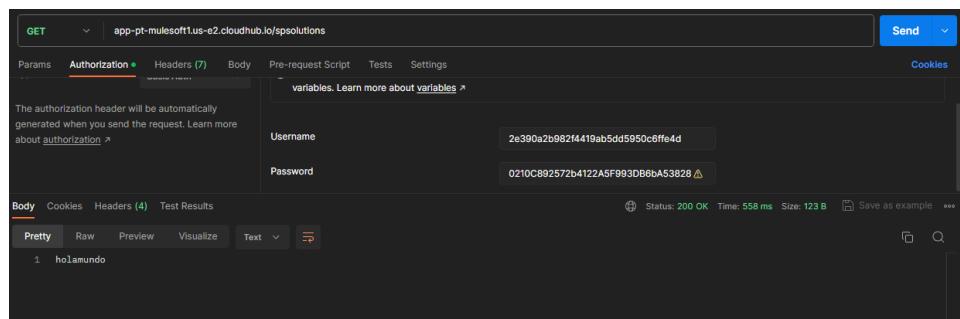
Para verificar que las credenciales son correctas, se hará uso de Postman, dentro del programa se dirige la pestaña de **Authorization**, y se selecciona **Basic Auth**.



Esto habilitara las opciones para colocar el Client ID y el Client Secret



Colocado las credenciales, se hace la verificacion realizando una peticion al Endpoint, el cual regresa un estatus 200 y la respuesta.



app-pt-mulesoft1.us-e2.cloudhub.io/spsolutions

localhost:8081/spsolutions

Client ID: 2e390a2b982f4419ab5dd5950c6ffe4d

Client Secret: 0210C892572b4122A5F993DB6bA53828

API Manager

API Administration (Sandbox) spsolutions (v1) - Policies

SANDBOX

← API Administration

API Summary
Alerts
Contracts
Policies
SLA Tiers
Settings

APIs / spsolutions / Policies

Apply policies to manage security, control traffic, and improve adaptability of your APIs. [Learn more about Policies](#).

Automated policies ⓘ

No automated policies applied

API-level policies ⓘ

+ Add policy

Client ID enforcement

Methods: All API methods Resource: All API resources

COMPLIANT

⋮

Policy enabled

API specification snippet

Edit configuration

Remove policy

Exchange

Publish new asset

All assets

Ingeniero (root)

Provided by MuleSoft

Shared with me

My applications

Public portal

My applications

Settings

Back to legacy search

Search assets

Any type: Ingeniero (root) Any lifecycle state:

Ingeniero assets (root)

HTTP API spsolutions ★★★★★ Ulices Salvador Aguila Contreras

Exchange

Publish new asset

All assets

My applications

Public portal

Settings

Search

My applications

Name	Description
spsolutionsapp	

☰  Exchange

Ingeniero ? UC

← My applications list

[Edit](#) [⋮](#)

spsolutionsapp

Client ID: 2e390a2b982f4419ab5dd5950c6ffe4d [Show](#)

Client Secret: Show

Application URL: –
Redirect URIs: –

Grant Types: –

v
Sandbox

spsolutions **1.0.x**
v1:19084083



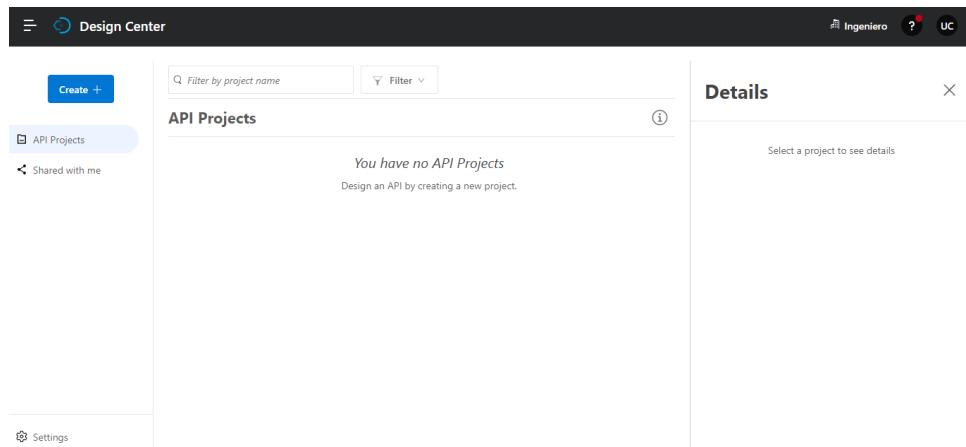
Application dashboard

Client ID: 2e390a2b982f4419ab5dd5950c6ffe4d

Client Secret: 0210C892572b4122A5F993DB6bA53828

Build your first API Specification with API Designer. Construcción de API Specification en API Designer

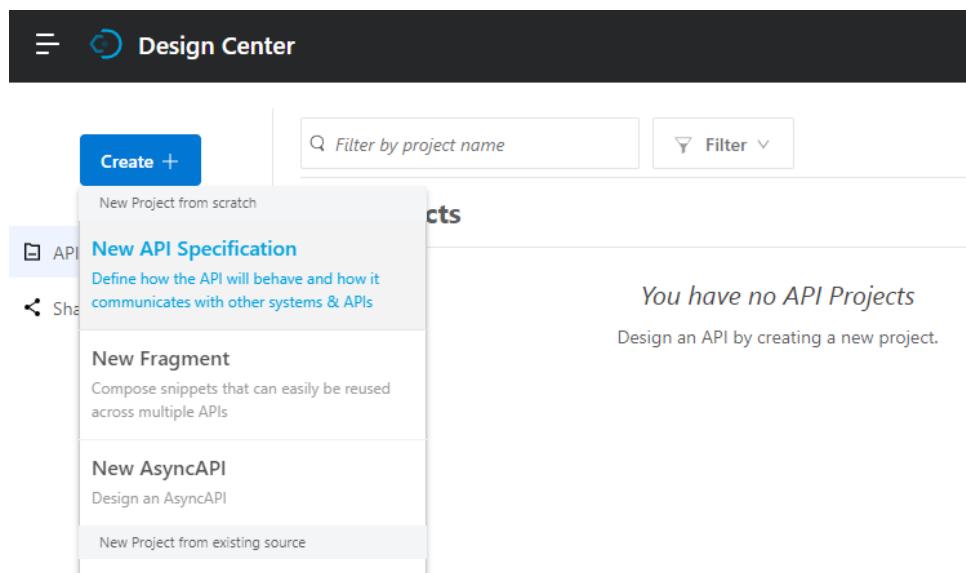
Antes de comenzar con la creación de la API se debe establecer la pagina de Anypoint Studio de MuleSoft, en ella se dirige a la pestaña **Desing Center**.



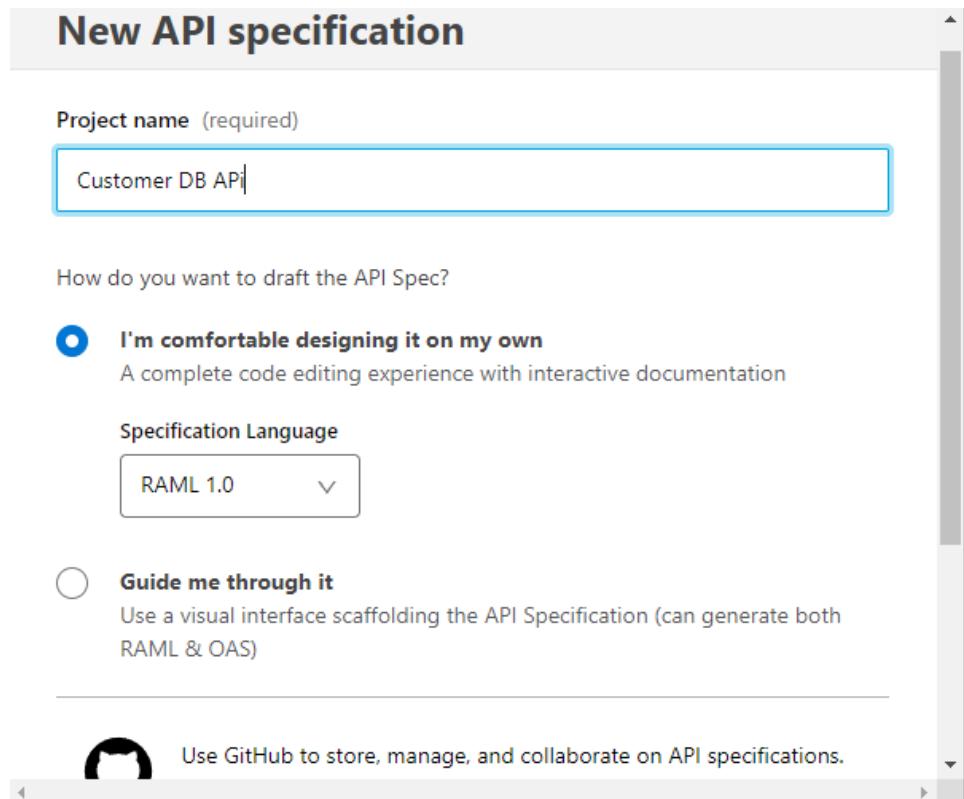
Problema:

Danny trabaja para Northern Trail Outfitters (NTO), un minorista de ropa deportiva, equipamiento y productos nutricionales. NTO tiene una base de datos de clientes antigua que requiere controladores y permisos especiales para acceder. Danny quiere que estos datos sean ampliamente accesibles y fácilmente en toda NTO. Para que esto sea posible, Danny decide crear una API REST sobre la base de datos.

Para solucionar el problema se debe crear una API que pueda recuperar contactos de la base de datos. Por lo que para crear la API la pestaña de Disign Center se hace clic en botón **Create > New API Especification**.



Se muestra una venta emergente la cual tiene algunas opciones diferentes que puede utilizar para configurar la especificación. Primero, debe dar un título a la API y, en este caso **Customer DB API**.



Se selecciona el lenguaje de especificación de API o el editor visual guiado. El editor actualmente es compatible con OAS y RAML, pero este proyecto, se usará el editor visual que desarrollará ambas especificaciones en tiempo real, por lo que se selecciona **Guide me through it**.

New API specification

Project name (required)

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

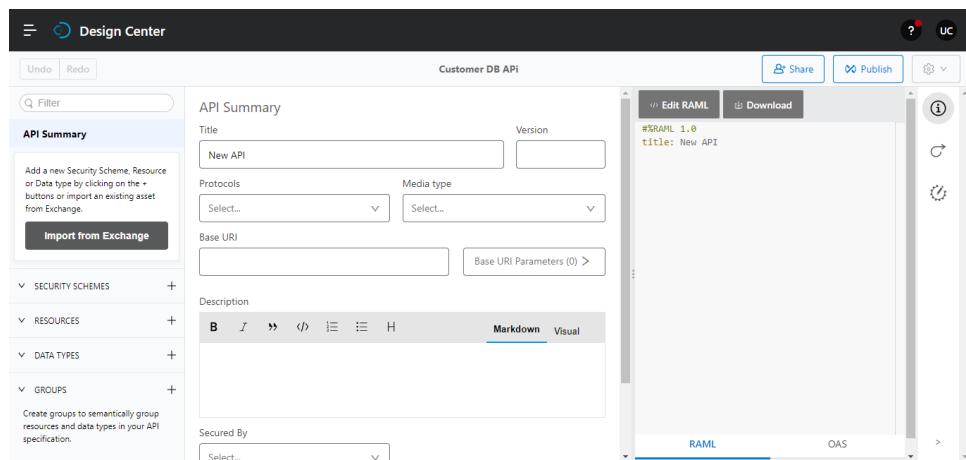
Specification Language

RAML 1.0

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

 Use GitHub to store, manage, and collaborate on API specifications.

Al crear la API mostrara la siguiente ventana emergente, en donde se visualiza los módulos dedicados a los distintos componentes de especificación que analizamos anteriormente. La primera página a la que se accede es el módulo **API Summary**, que es donde puede configurar el título de la especificación API, los métodos de comunicación (protocolo y tipo de medio), el URI base y la descripción. En este punto, se puede configurar manualmente su API según las instrucciones que se muestran a continuación.



```
#%RAML 1.0
title: New API
```

En parte derecha, se observa que el código RAML y OAS se genera automáticamente a medida que se actualiza la especificación en el editor visual.

En el panel izquierdo, se visualiza las opciones para configurar esquemas de seguridad, recursos y tipos de datos. Se agrega algunos de los **Data Types y recursos que necesitará la API**. Se hace clic en el ícono + al lado de la pestaña de Data Types a la izquierda y se hace clic en **New data type**.

En esta sección se agregan los tipos de datos con la siguiente información.

Posteriormente se da click en **Add Property** en donde se agregan las siguientes propiedades para la construcción de la API.

The screenshot shows the Design Center interface with the title "Customer DB API". On the left, there is a sidebar with sections like "API Summary", "SECURITY SCHEMES", "RESOURCES", "DATA TYPES", and "GROUPS". Under "DATA TYPES", there are five properties listed: "street", "city", "PostalCode", "state", and "country". Each property has fields for "Property Name", "Type" (set to "String"), "Required" (checkbox), and "Union" (checkbox). A "Direction" section is also present. At the bottom of the list is a "Add Property" button.

Al agregar propiedades estas se visualizan de manera simultánea en el editor de código.

The screenshot shows the RAML code editor with two buttons: "Edit RAML" and "Download". The code itself is a RAML schema for a "types" section. It includes a "Direction" section and properties for "street?", "city?", "PostalCode?", "state?", and "country?". Each property has an "example" field set to "Example" and a "type" field set to "string".

```
types:
  Direction:
    description: Data types direction
  properties:
    street?:
      example: Example
      type: string
    city?:
      example: Example
      type: string
    PostalCode?:
      example: Example
      type: string
    state?:
      example: Example
      type: string
    country?:
      example: Example
      type: string
```

Finalmente, en el campo de Example se llenará los campos con información para que esta se visualice.

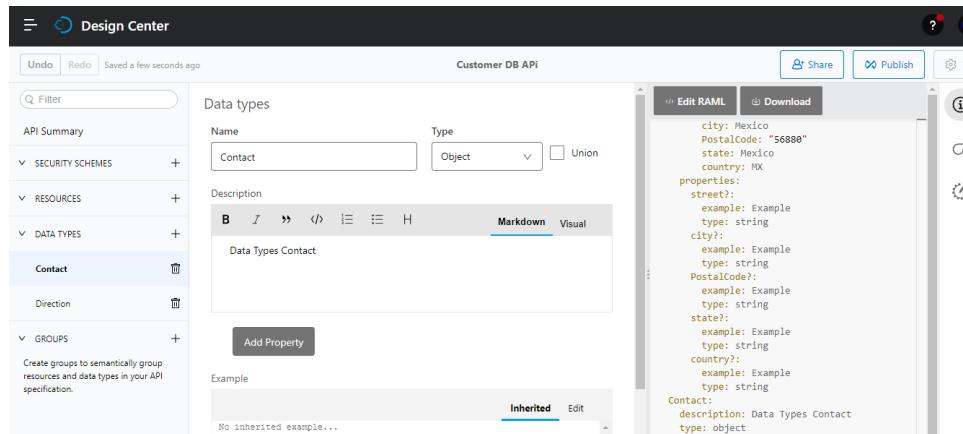
Example



The screenshot shows a code editor window with the title "Edit". It displays a JSON-like object definition:

```
{  
  "street": "Mexico, Mexico",  
  "city": "Mexico",  
  "PostalCode": "56880",  
  "state": "Mexico",  
  "country": "MX"  
}
```

Realizado lo anterior, se repetirá los mismos pasos, pero para las propiedades de Contact.

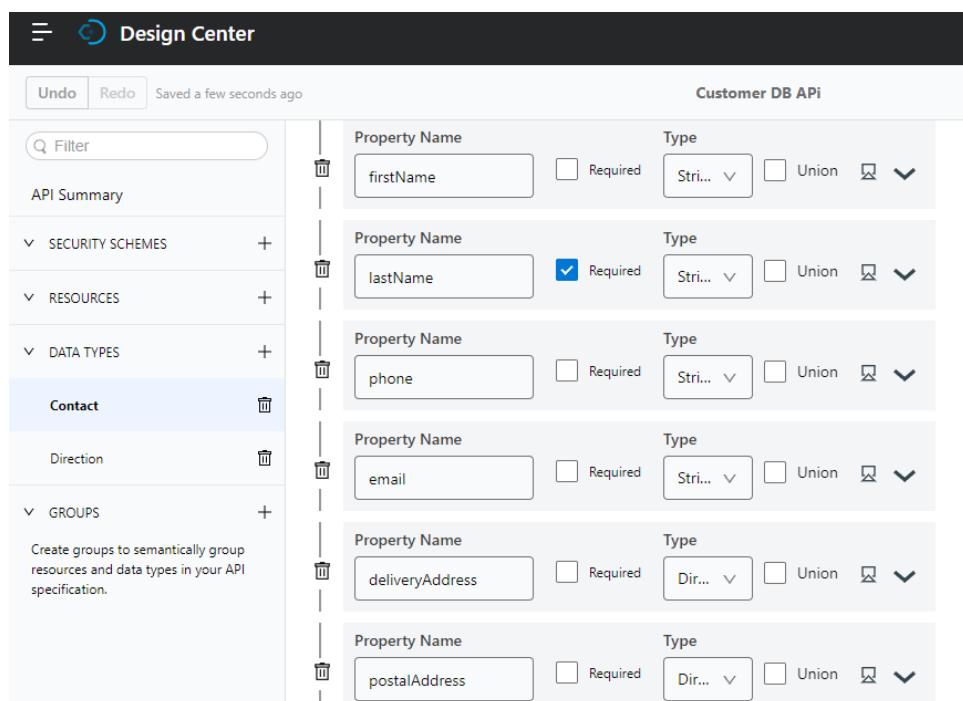


The screenshot shows the "Design Center" interface for a "Customer DB API". On the left, the sidebar shows "Data types" with "Contact" selected. In the main area, a "Data types Contact" card is displayed with the following details:

- Name: Contact
- Type: Object
- Description: Data Types Contact
- Properties:
 - street?: string
 - city?: string
 - PostalCode?: string
 - state?: string
 - country?: string
- Example:

```
city: Mexico  
PostalCode: "56880"  
state: Mexico  
country: MX  
properties:  
  street?:  
    example: Example  
    type: string  
  city?:  
    example: Example  
    type: string  
  PostalCode?:  
    example: Example  
    type: string  
  state?:  
    example: Example  
    type: string  
  country?:  
    example: Example  
    type: string
```

Contact:
 description: Data Types Contact
 type: object



The screenshot shows the "Design Center" interface for a "Customer DB API". On the left, the sidebar shows "Data types" with "Contact" selected. In the main area, the "Contact" data type is expanded to show its properties:

Property Name	Type
firstName	String
lastName	String
phone	String
email	String
deliveryAddress	Directory
postalAddress	Directory

Example

The screenshot shows a RAML editor interface. At the top, there are tabs for "Inherited" and "Edit". The "Edit" tab is selected, indicated by a blue underline. Below the tabs is a code editor window containing a JSON object:

```
{"firstName": "Ulises Salvador",  
 "lastName": "Aguila Contreras",  
 "phone": "5514991559",  
 "email": "Ulices13SAC@outlook.es",  
 "deliveryAddress": {  
   "street": "Mexico, Mexico",  
   "city": "Mexico"  
 } }
```

Below the code editor is a preview pane. It contains two buttons: "Edit RAML" and "Download". The "Edit RAML" button is highlighted with a dark grey background. The preview pane displays the following RAML code:

```
description: Data Types Contact  
example:  
  strict: true  
  value:  
    firstName: Ulises Salvador  
    lastName: Aguila Contreras  
    phone: "5514991559"  
    email: Ulices13SAC@outlook.es  
    deliveryAddress:  
      street: Mexico, Mexico  
      city: Mexico  
      PostalCode: "56880"  
      state: Mexico  
      country: MX  
    postalAddress:  
      street: Mexico, Mexico  
      city: Mexico  
      PostalCode: "56880"  
      state: Mexico  
      country: MX  
    properties:  
      firstName?:  
        example: Example
```

At the bottom of the preview pane, there are two tabs: "RAML" and "OAS". The "RAML" tab is selected, indicated by a blue underline.

Despues de crear los Data Types de debe definir los recursos específicos, para realizar este paso, se selecciona el icono + que se encuentra en la pestaña de Resources.

La configuración dos recursos se realiza de la siguiente manera:

Posteriormente se configura las respuestas para el /customers resource:

- Se seleccione la pestaña **Responses** junto a Summary y se hace clic en **Add New Response**.
- Se asegura de que el código de status sea **200 OK** y luego se hace clic en **Add Body**.
- Se segúra de que el Media Type contenga **application/json** y se establece el tipo en **Array**.
- Expandir el cuerpo desde la flecha de la derecha (Details). Se seleccione el tipo de **Contact** que se acaba de crear.

Una vez más se realiza los pasos anteriores para definir los recursos específicos, pero para **/customers/{customerId}**

Para este caso, se debe agregar los parámetros ya que se requiere un parámetro URI, por lo que se asignan los siguientes valores.



Para la configuración de la response para el recurso /customers/{customerId} se realiza de la siguiente manera:

- Se seleccione la pestaña **Responses** junto a Summary y se hace clic en **Add New Response**.
- Se asegura de que el código de status sea **200 OK** y luego se hace clic en **Add Body**.
- Se asegura de que el Media Type contenga **application/json** y establece el tipo en **Contact**.



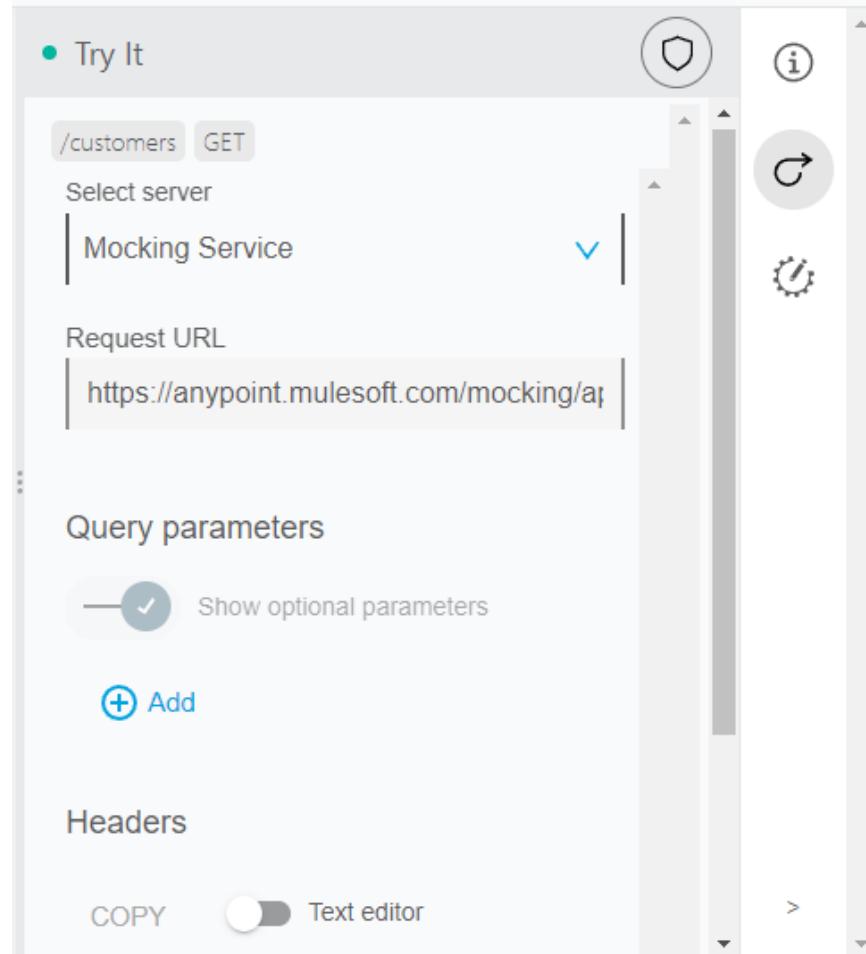
Realizado los pasos de Data Types y Resources se realiza la prueba de la API utilizando el servicio de simulación de API Designer. El servicio simulado proporciona un enlace de prueba a una API y devuelve las respuestas (tanto códigos de estado HTTP como cargas útiles de ejemplo) que están definidas en la especificación de API.

Para realizar las pruebas pertinentes el la pestaña de Resources se selecciona /customers en el panel de navegación izquierda.

Posteriormente en el panel, se dará clic en el botón **Try It**, que se encuentra en el extremo derecho de la pantalla.

The screenshot shows the API Designer interface with the RAML tab selected. The central area displays the RAML code for the '/customers/{customerId}' endpoint. On the right side, the 'Try It' button is highlighted with a red box. The 'Responses' tab is also highlighted in blue, indicating it's the active section.

En **Select Server**, se selecciona **Mocking Service**. Esto crea una solicitud de API ficticia y muestra la respuesta de ejemplo tal como aparecería si la API estuviera activa y devolviera datos.



Se selecciona el recurso `/customers/{customerId}` y se hace clic en **Send** para ver una respuesta similar para un solo cliente. Se observa cómo el primer punto `/customers` devuelve una matriz de objetos de contacto y el segundo punto `/customers/{customerId}` devuelve un único objeto de contacto.

The screenshot shows the 'Try It' section of the Anypoint Platform Mocking Service. The URL is set to `/customers` with a `GET` method. The response status is `200 OK` and the time taken is `526.6 ms`. The response body is a JSON array containing two objects, each representing a customer contact. The first object is for 'Ulises Salvador' and the second for 'Juan Perez'.

```
[{"id": 1, "name": "Ulises Salvador", "lastName": "Aguila Contreras", "phone": "5514991559", "email": "Ulises135AC@outlook.es", "deliveryAddress": [{"street": "Mexico, Mexico", "city": "Mexico", "PostalCode": "56880", "state": "Mexico", "country": "MX"}], {"id": 2, "name": "Juan Perez", "lastName": "Garcia", "phone": "5514991560", "email": "JuanPerez123@gmail.com", "deliveryAddress": [{"street": "Mexico, Mexico", "city": "Mexico", "PostalCode": "56880", "state": "Mexico", "country": "MX"}]}
```

```

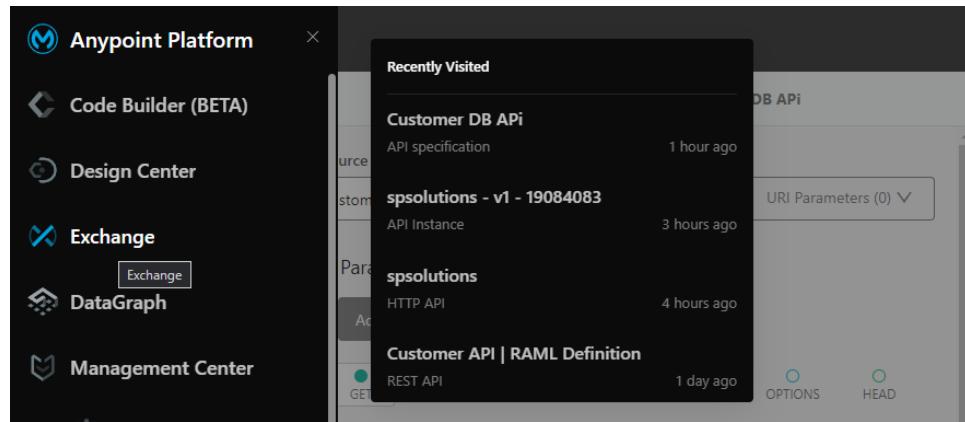
1  [
2    {
3      "firstName": "Ulises Salvador",
4      "lastName": "Aguila Contreras",
5      "phone": "5514991559",
6      "email": "Ulises135AC@outlook.es",
7      "deliveryAddress": {
8        "street": "Mexico, Mexico",
9        "city": "Mexico",
10       "PostalCode": "56880",
11       "state": "Mexico",
12       "country": "MX"
13     },
14     "postalAddress": {
15       "street": "Mexico, Mexico",
16       "city": "Mexico",
17       "PostalCode": "56880",
18       "state": "Mexico",
19       "country": "MX"
20   }

```

Para crear un enlace público que se pueda compartir con otros usuarios con fines de prueba, puede hacerse desde el botón **Mocking Service Configuration** a la derecha.

<https://anypoint.mulesoft.com/mockng/api/v1/links/7e7090a0-a0c4-4e33-96b8-9fa3e514e9b1>

Por último, se publica la API en Exchange para que otros usuarios puedan descargarla y trabajar con ella. Se selecciona el botón azul **Publish** en la parte superior derecha y seleccione **Publish to Exchange**.



The screenshot shows the Exchange interface under the "Ingeniero" user. The top navigation bar includes "Exchange", "Ingeniero", a help icon, and a user icon. The main area is titled "Ingeniero assets (root)" and displays two assets:

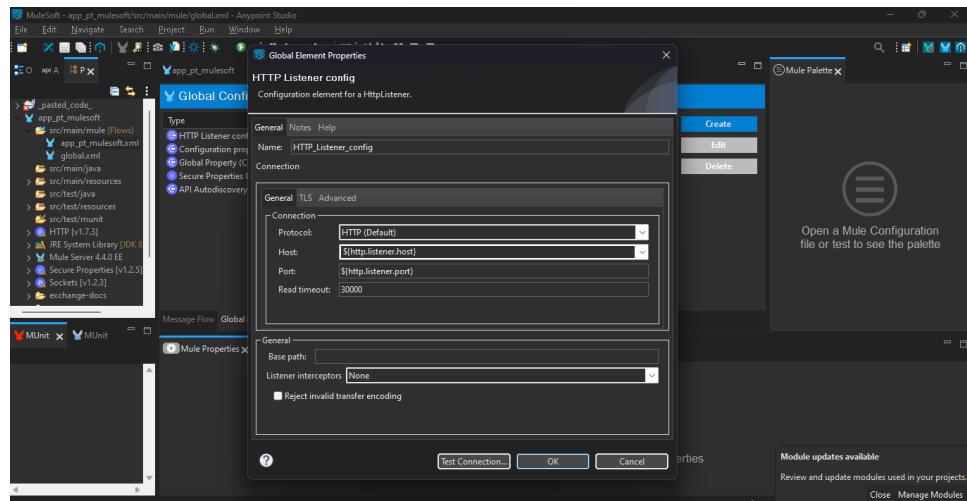
- Customer DB API (REST API, 5 stars, Ulises Salvador Aguila Contreras)
- spsolutions (HTTP API, 5 stars, Ulises Salvador Aguila Contreras)

On the left, there's a sidebar with options like "Publish new asset", "Search assets", "All assets", "Ingeniero (root)", "Provided by MuleSoft", "Shared with me", "My applications", "Public portal", and "Settings".

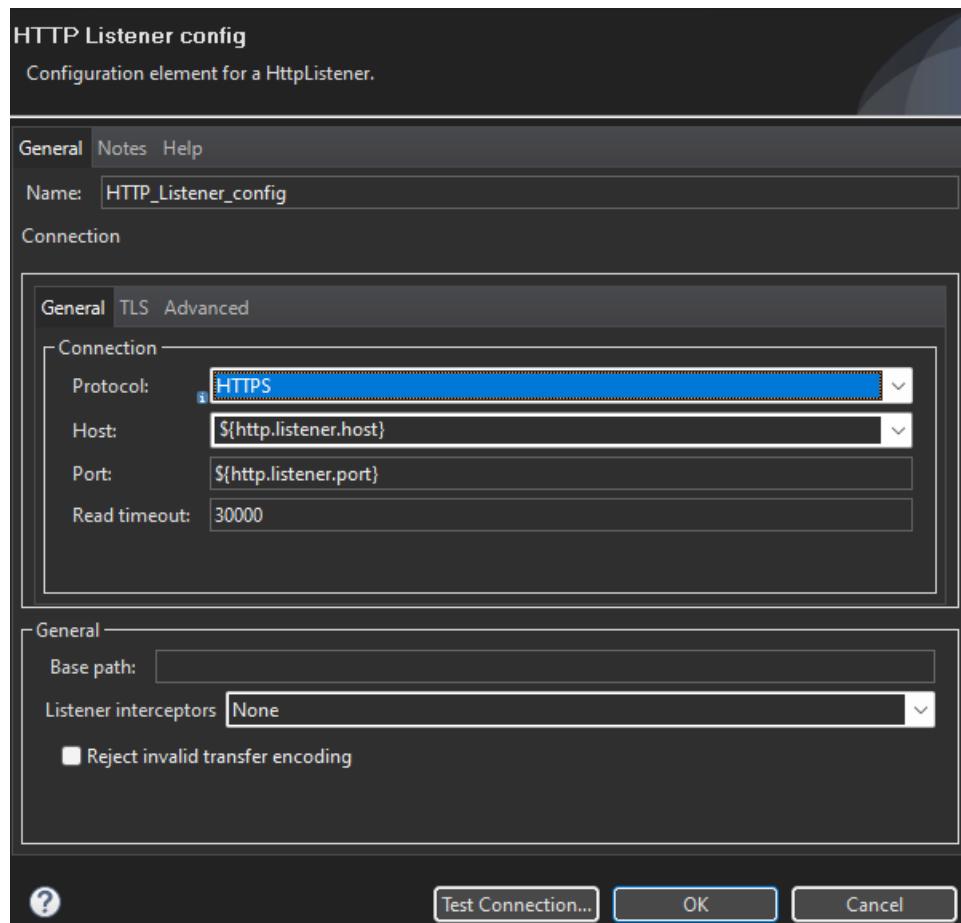
How to configure an HTTPS endpoint in Anypoint Studio. Habilitar el consumo de Mule Application por medio de HTTPS

Para realizar la configuración respectiva se hará uso del proyecto de app_pt_mulesoft.xml, en la cual se definió el port, host y la respuesta del Endpoint Local y para CloudHub.

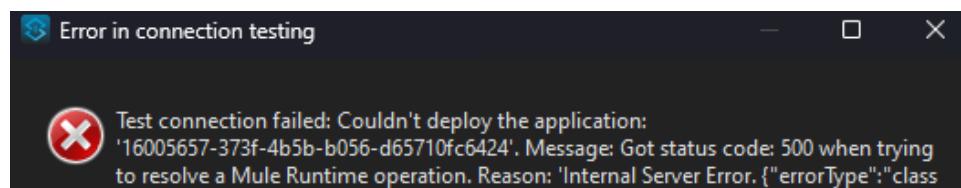
Como primer paso para la configuración se hará uso del Listener aplicado al proyecto, en el cual se define la configuración hecha anteriormente.



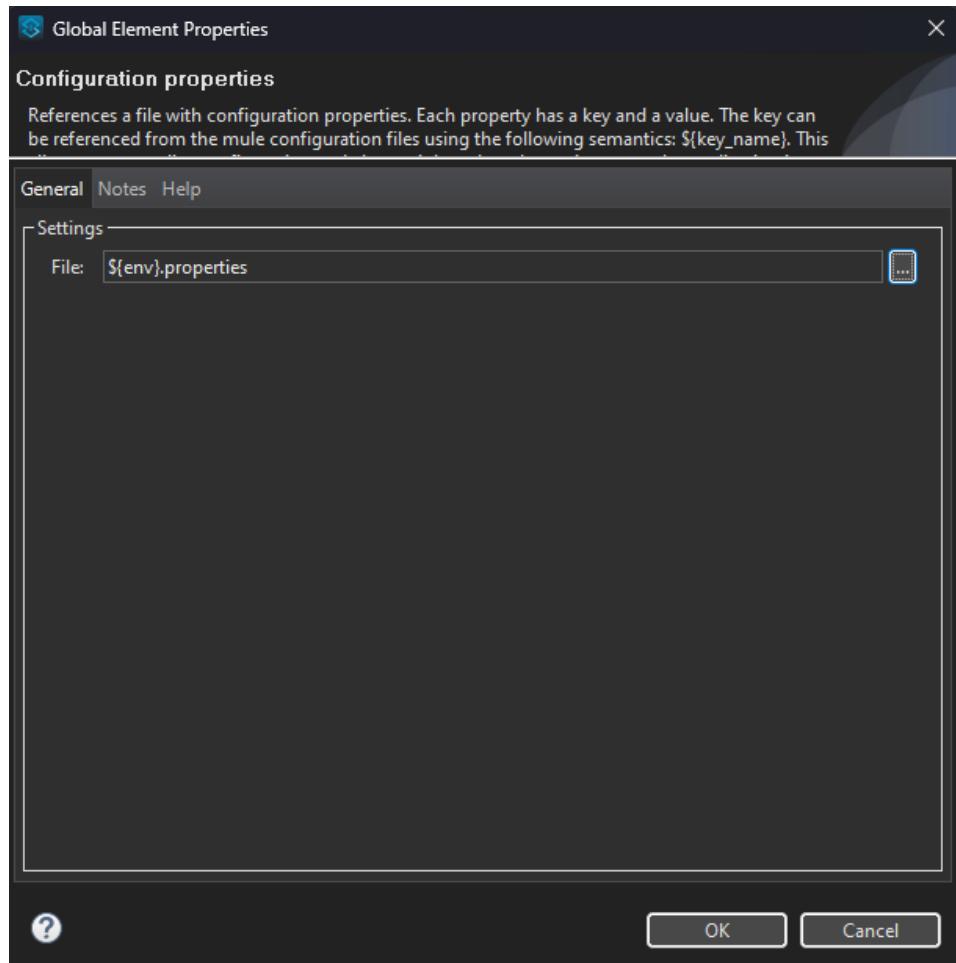
Se realiza la configuración para habilitar HTTPS en su host local y punto final de CloudHub.



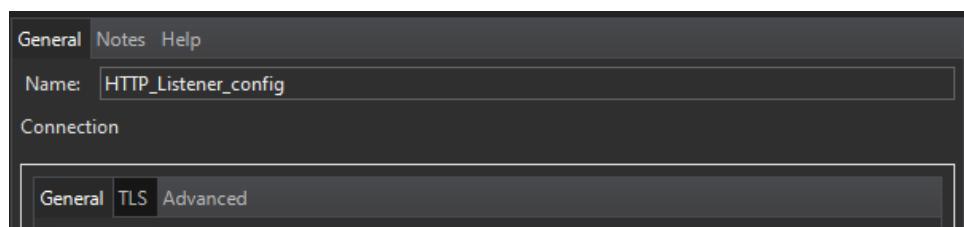
Posteriormente se realiza el testeo del enlace, por lo que se hace clic en **Test Connection**.



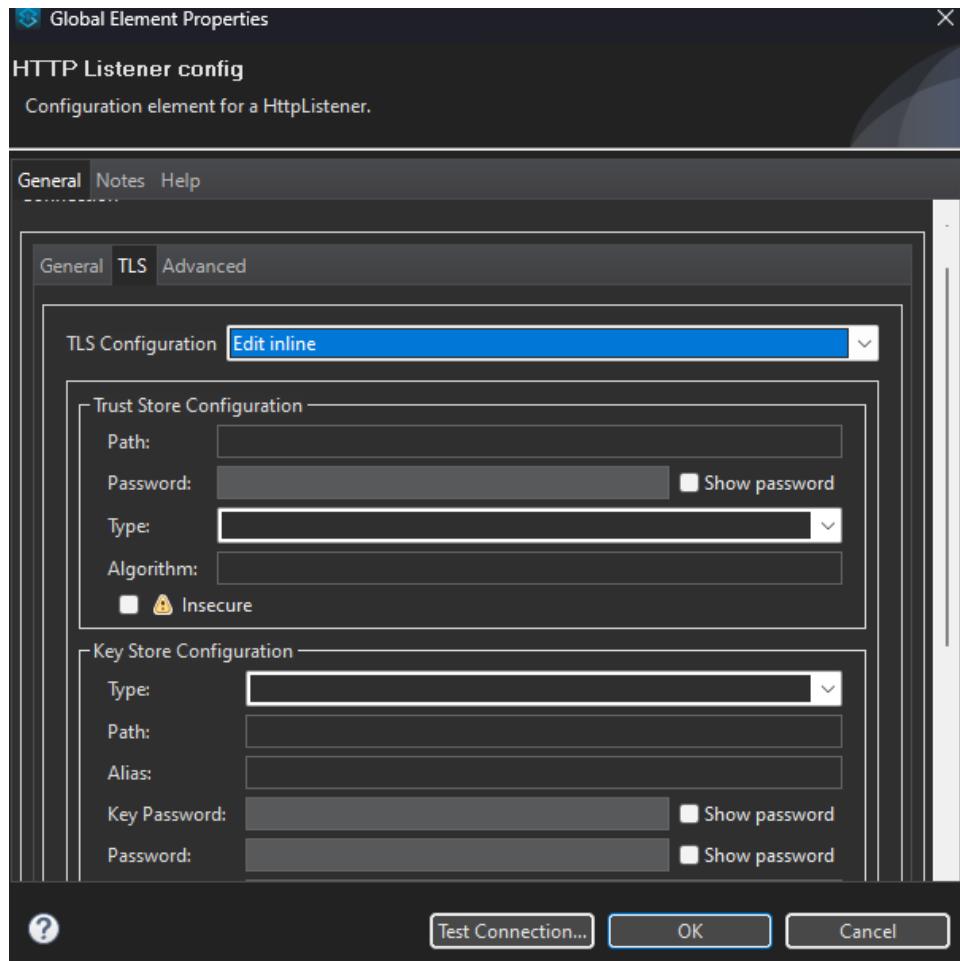
Como se muestra, la conexión no se realizó de manera correcta por lo que se debe realizar una configuración en **Configure properties**.



Para hacer la modificación se da clic en los tres puntos y se selecciona el archivo **local.properties**.



En su configuración de TLS se seleccionara la opción **Edit Line**, la cual desplegará una serie de campos que deberá llenarse.



Para crear el archivo con los valores necesarios para llenar los campos, en una terminal se hace uso del siguiente código:

```
keytool -genkey -alias key-alias -keystore keystore-name.jks -keyalg RSA -storetype JKS
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Ulide> keytool -genkey -alias key-alias -keystore keystore-name.jks -keyalg RSA -storetype JKS
Introduzca la contraseña del almacén de claves: |
```

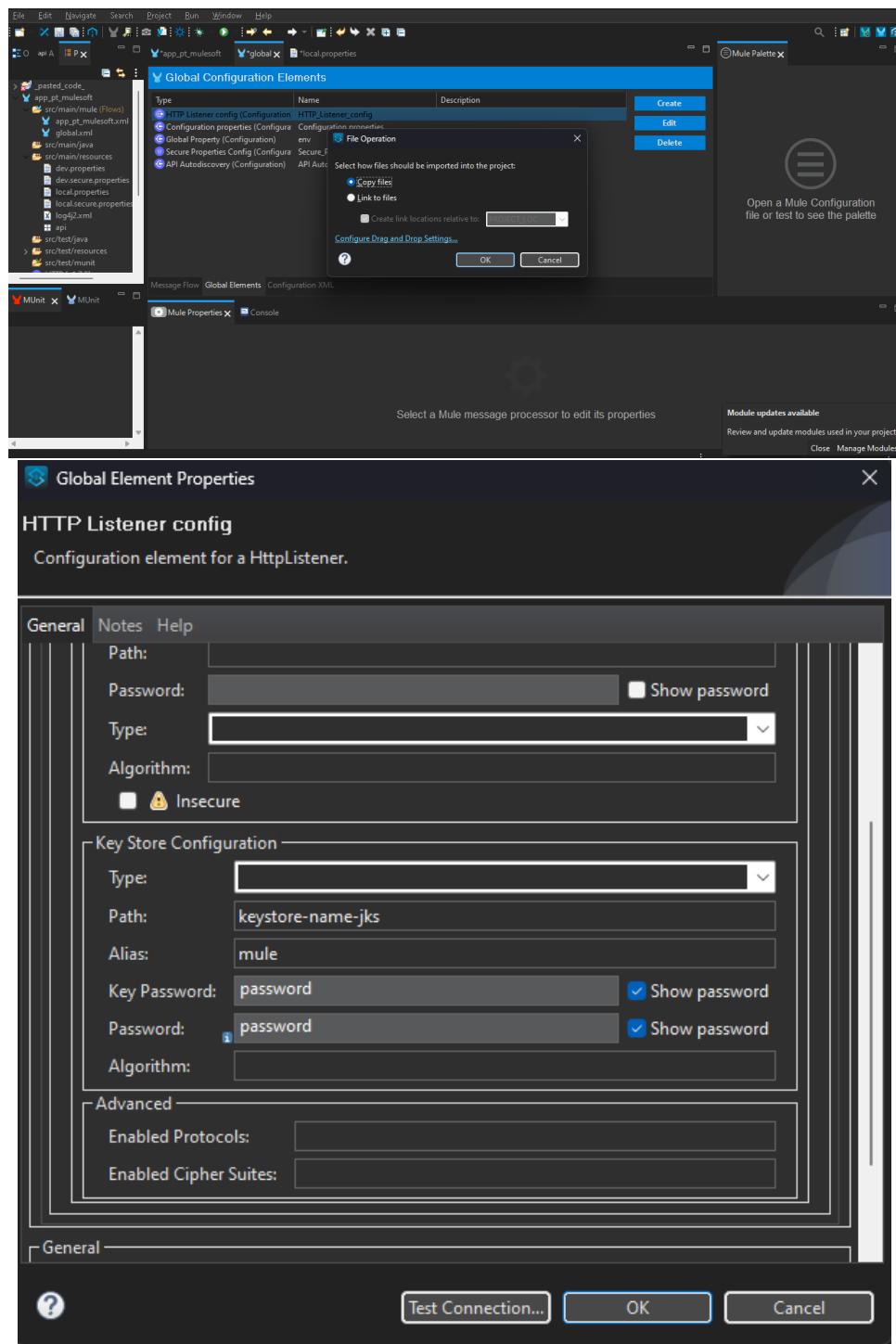
Al ejecutar el código, realizará una serie de campos que se deben llenar, para que de esa manera pueda generar el archivo.

```

¿Cuáles son su nombre y su apellido?
[unknown]: ulices
¿Cuál es el nombre de su unidad de organización?
[unknown]: spsolutions
¿Cuál es el nombre de su organización?
[unknown]: spsolutions
¿Cuál es el nombre de su ciudad o localidad?
[unknown]: mexico
¿Cuál es el nombre de su estado o provincia?
[unknown]: mexico
¿Cuál es el código de país de dos letras de la unidad?
[unknown]: mx
¿Es correcto CN=Ulises, OU=spsolutions, O=spsolutions, L=mexico, ST=mexico, C=mx?
[no]: 

```

Creado el archivo, este debe colocarse en la carpeta **src/main/resources**.



Durante la realización de esta practica, se presento el problema que al ejecutar el código en la terminal si me genera el archivo JDK, pero no me genera el **Alias**, Por lo que el enlace y el test configuration me marcan un conflicto.

