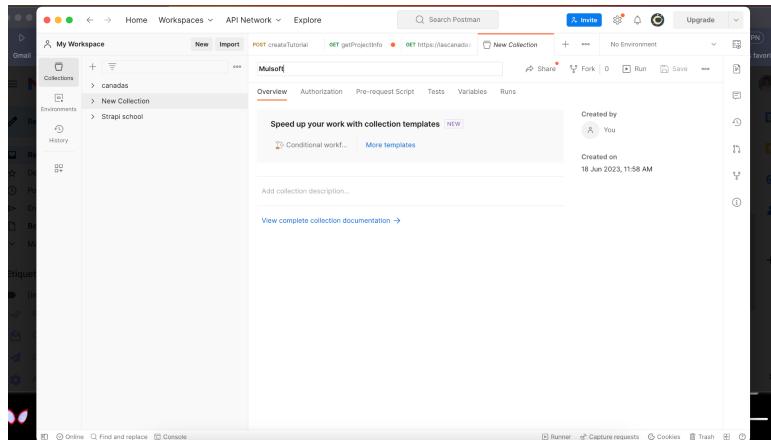


Prácticas con MuleSoft para SP Solutions

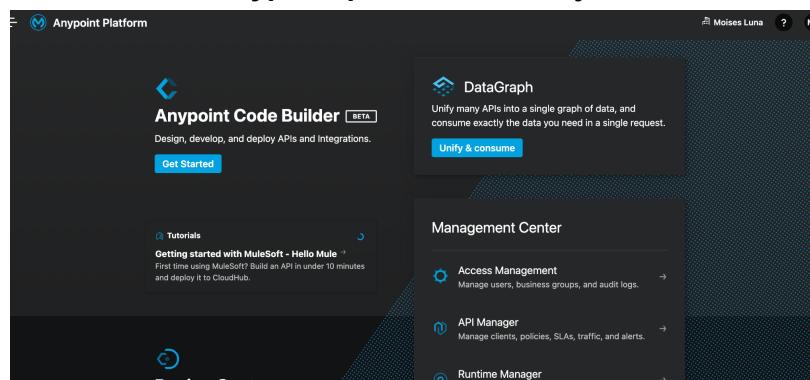
A continuación listo la serie de pasos que realicé durante la práctica trabajando con la suite de herramientas de integración de MuleSoft.

Build your first Hello Mule Application

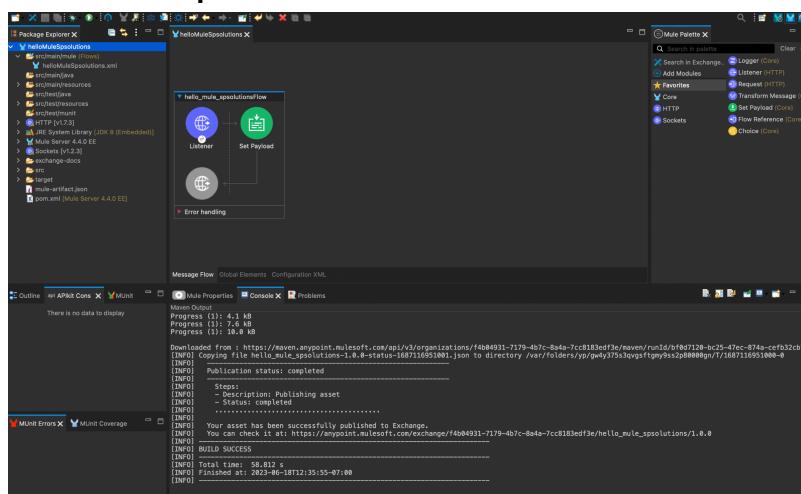
1. Crear una nueva colección de API's en postman para el ejercicio.



2. Cree cuenta en anypoint para realizar el ejercicio



3. Creación de aplicación Hello World en MuleSoft



4. Despliegue de la aplicación en CloudHub

The screenshot shows the Runtime Manager interface for the application 'hellomulemoilu'. The application is listed as 'Running' with a green status indicator. Key details shown include:

- Type: Application
- Status: Running
- URL: <https://hellomulemoilu-zzs5x3.5sc6y6-4.usa-e2.cloudhub.io>
- Runtime: 4.4.0
- Replicas: -
- Last updated: 46 seconds ago

Below the summary, there are three monitoring charts: 'Mule Messages', 'Average Response Time', and 'Errors', all showing 'No data points'.

5. Uso de la aplicación en PostMan llamando al endpoint que se desplegó

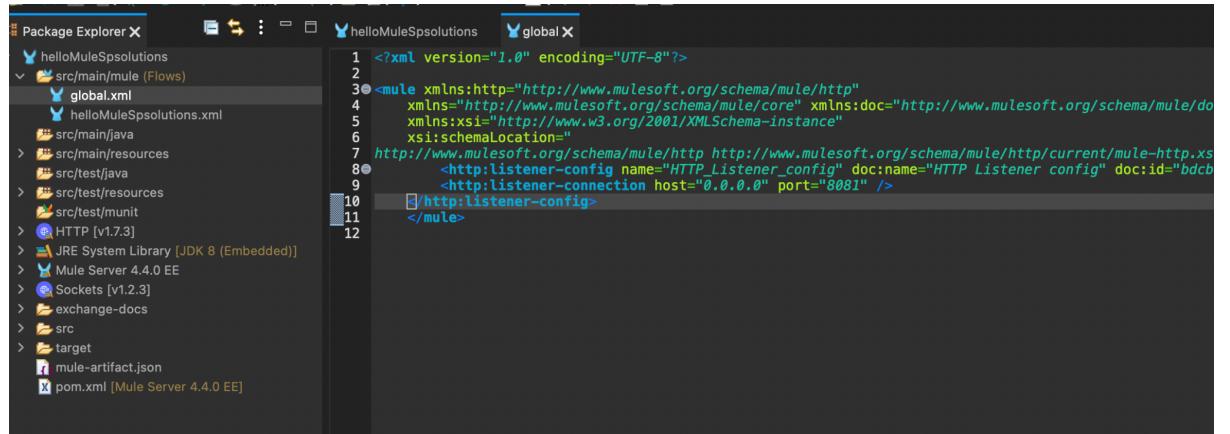
The screenshot shows the Postman interface with a successful API call to the deployed application. The request details are as follows:

- Method: GET
- URL: <https://hellomulemoilu-zzs5x3.5sc6y6-4.usa-e2.cloudhub.io/hellomule>
- Response status: 200 OK
- Time: 455 ms
- Size: 166 B

The response body contains the message "Hello Mule".

How to set up your global elements and properties

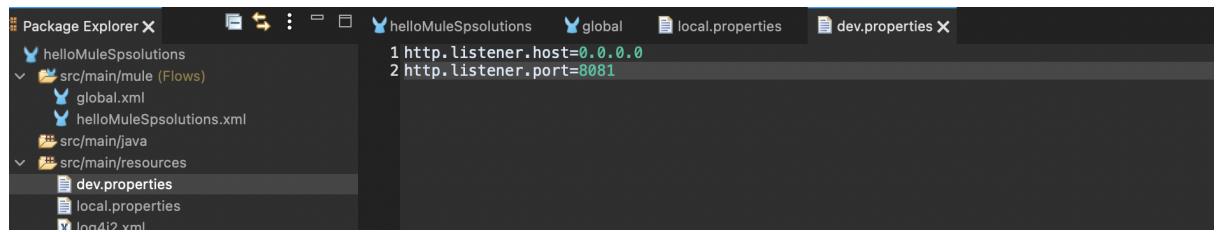
1. Cree nuevo archivo global



```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns:http="http://www.mulesoft.org/schema/mule/http"
      xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:doc="http://www.mulesoft.org/schema/mule/doc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd"
      doc:id="bdcb">
    <http:listener-config name="HTTP_Listener_config" doc:name="HTTP Listener config" doc:id="bdcb">
        <http:listener-connection host="0.0.0.0" port="8081" />
    </http:listener-config>

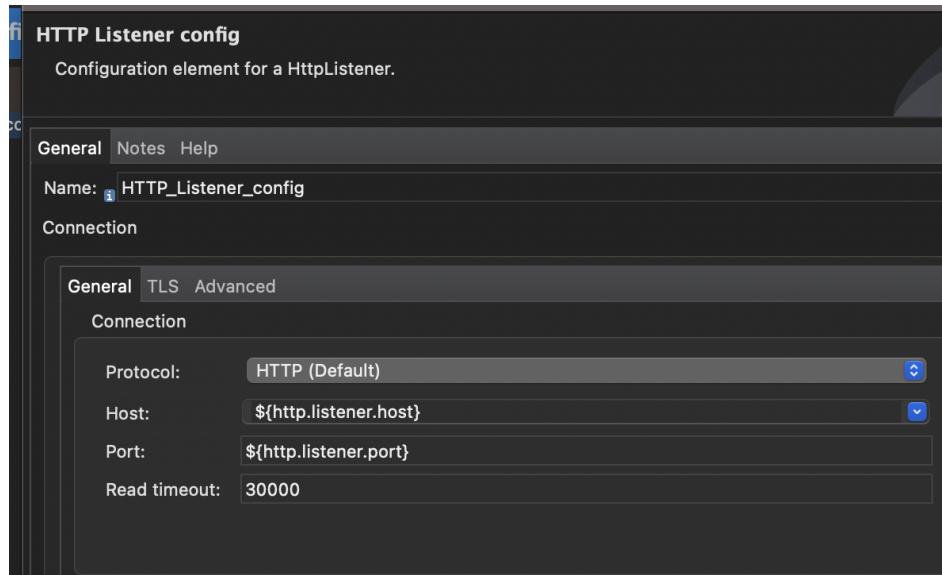
```

2. Cree archivos de configuración local y dev para evitar que esté hardcoded.

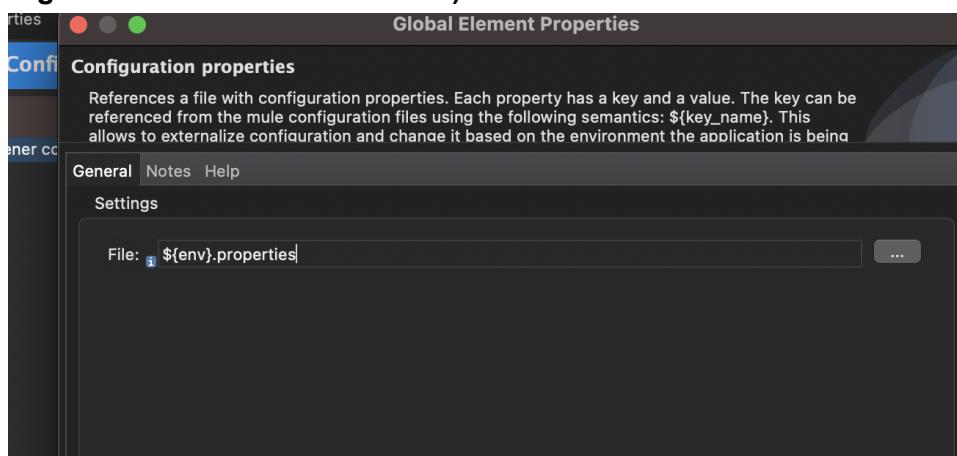


```
http.listener.host=0.0.0.0
http.listener.port=8081
```

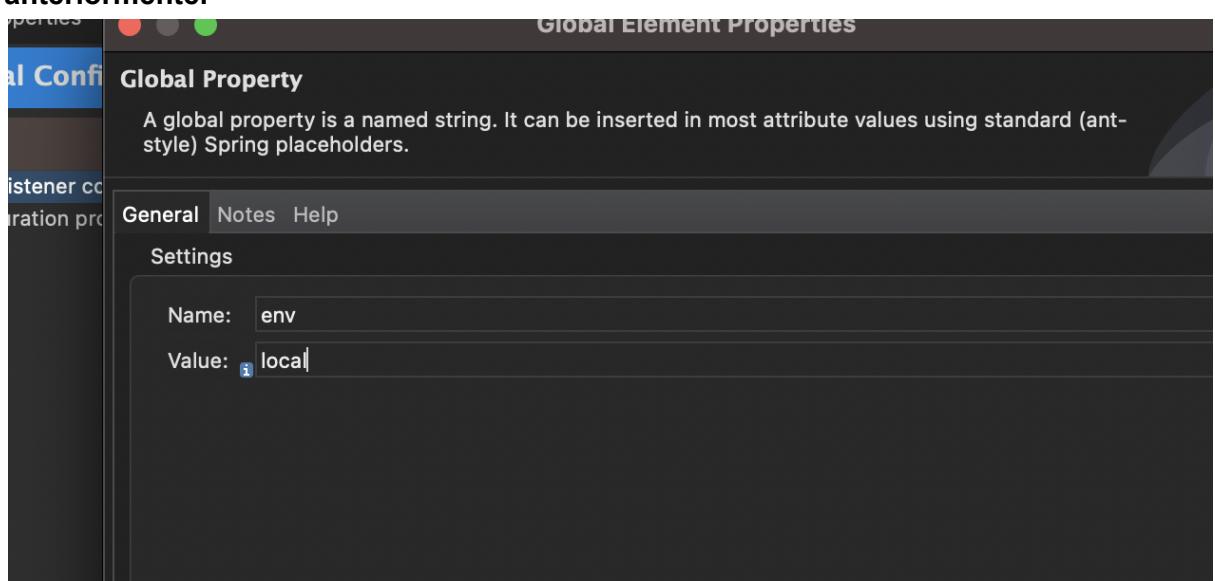
3. Agregué en la configuración el nombre de las variables creadas



4. Cree la configuración de las propiedades (esto sirve para seleccionar una según el ambiente de desarrollo)



5. Seleccioné las propiedades globales para el ambiente en base a lo creado anteriormente.



6. Verificando que todo este correcto tras estas configs.

A screenshot of the Mule Studio interface. At the top, there's a toolbar with a GET icon and the URL "http://localhost:8081/hellomule". Below the toolbar, there's a main panel with a "Send" button and a "Body" tab showing the response "Hello Mule". On the left, there's a sidebar with sections for "Params" and "Query Params", both currently empty.

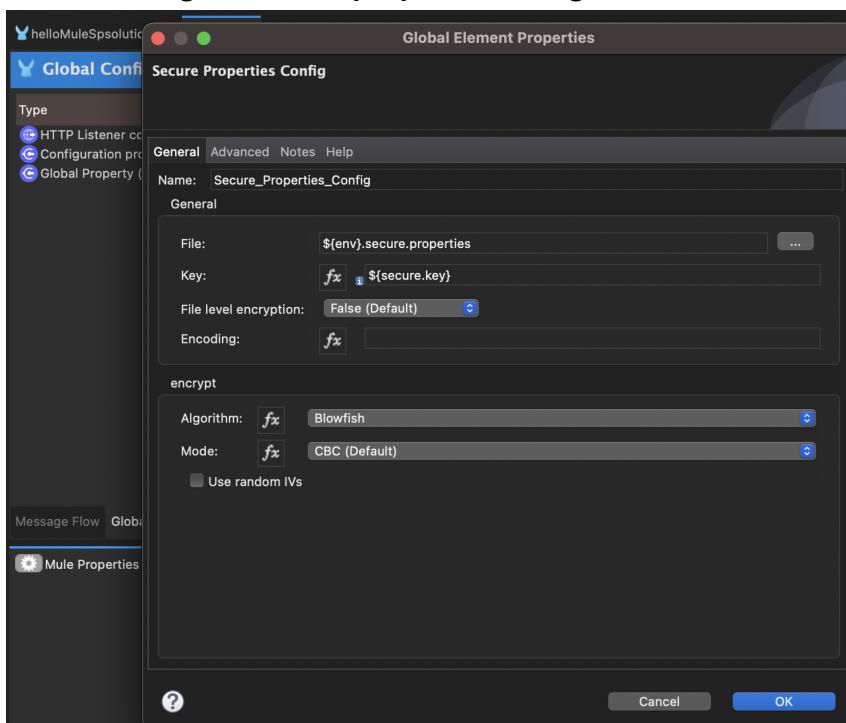
How to secure properties before deployment

1. Cree archivos de propiedades seguras

The screenshot shows a file editor with two tabs: "local.secure.properties" and "dev.secure.properties". The "local.secure.properties" tab contains the following content:

```
1 example.username=myUsernameDev
2 example.password=myPasswordDev
```

2. Cree la configuración de propiedades seguras



3. Cree un script que va a leer las credenciales usando el lenguaje de programación de mulesoft

The screenshot shows the Mule Studio interface with a Java script component. The code in the script is:

```
1@output application/java
2 ---
3"Username: " ++ Mule::p("secure::example.username")
4 ++ " --" ++
5 "Password: " ++ Mule::p("secure::example.password")
```

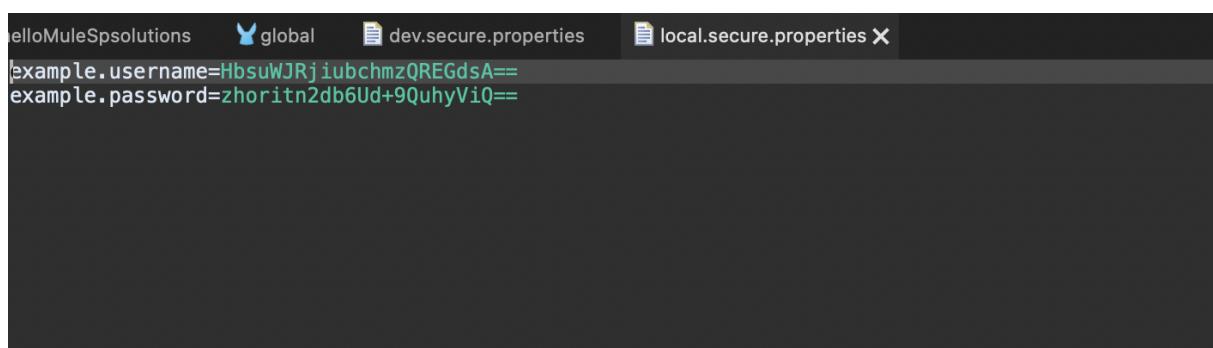
4. Se encriptan y pegan las credenciales encriptadas

```
["myPasswordLocal"

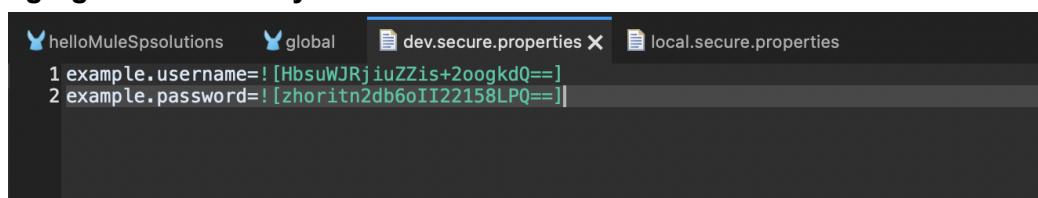
zhoritn2db6Ud+9QuhyViQ==
→ mulesoft java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool \
string \
encrypt \
Blowfish \
CBC \
MyMuleSoftKey \
["myUsernameDev"

HbsuWJRjiuZZis+2oogkdQ==
→ mulesoft java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool \
string \
encrypt \
Blowfish \
CBC \
MyMuleSoftKey \
["myPasswordDev"

zhoritn2db6oII22158LPQ==
→ mulesoft ]
```



5. Agregar la secure.key en environment



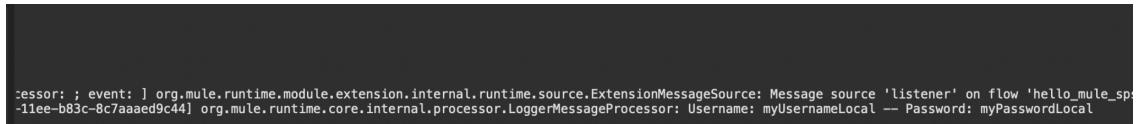
6. Correr y probar que la aplicación siga funcionando

Llamada al API

The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8081/hellomule`. The response body is displayed as:

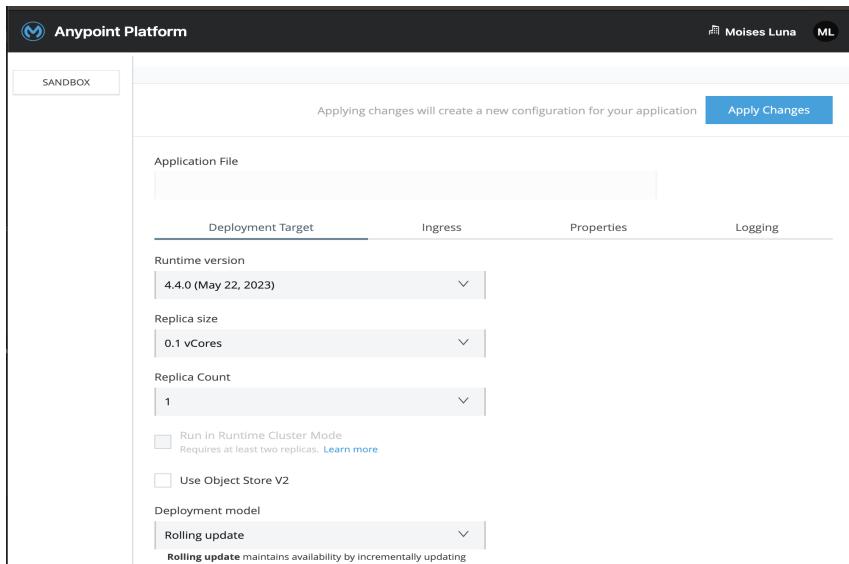
```
1 Hello Mule
```

Logs después de la llamada

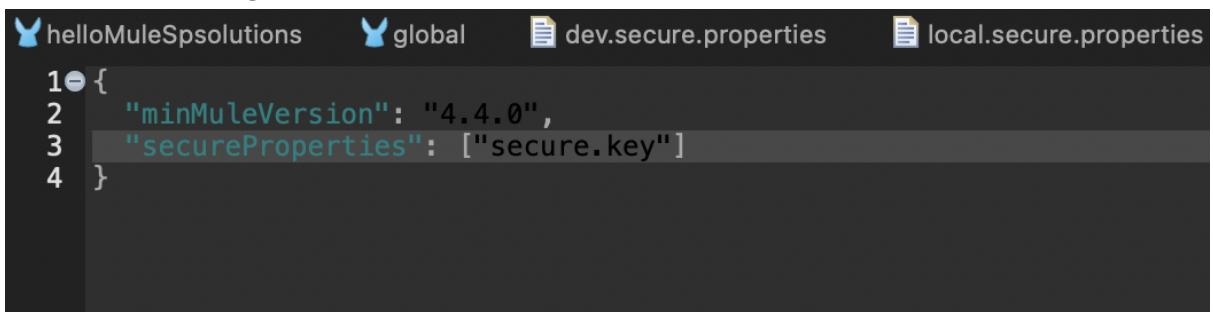


```
cessor: ; event: ] org.mule.runtime.module.extension.internal.runtime.source.ExtensionMessageSource: Message source 'listener' on flow 'hello_mule_sp-11ee-b83c-8c7aaaed9c44] org.mule.runtime.core.internal.processor.LoggerMessageProcessor: Username: myUsernameLocal -- Password: myPasswordLocal
```

7. Despliegué de aplicación

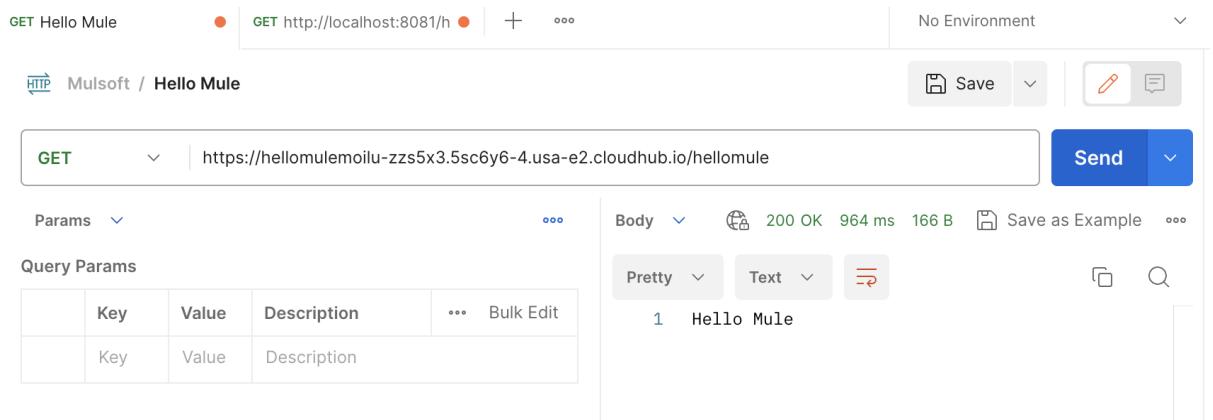


8. Escondiendo propiedades secretas agregando al mule-artifact.json(después de eso se despliega otra vez la aplicación)



```
1 - {  
2   "minMuleVersion": "4.4.0",  
3   "secureProperties": ["secure.key"]  
4 }
```

9. Verificando que la aplicación siga funcionando



GET Hello Mule GET http://localhost:8081/h No Environment

HTTP Mulsoft / Hello Mule Save Send

GET https://hellomulemoilu-zzs5x3.5sc6y6-4.usa-e2.cloudhub.io/hellomule

Params Body

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Pretty Text Bulk Edit

1 Hello Mule

10. Todo correcto en los logs de la app

● hellomulemoilu

The screenshot shows a log viewer interface with the following details:

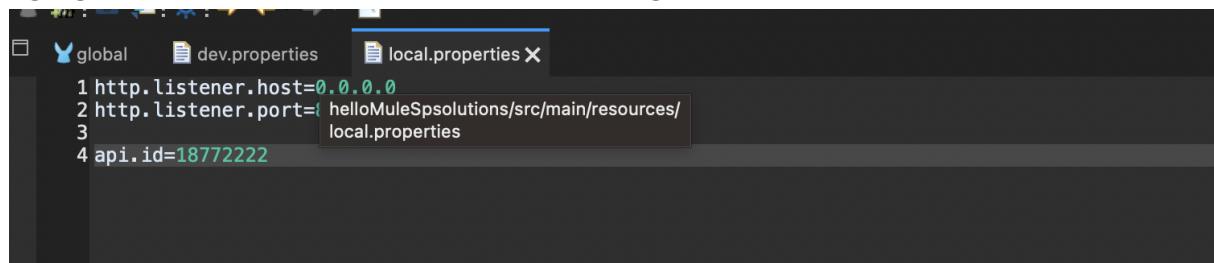
- Config:** d03ecd (Last Successful)
- Search:** Search bar with placeholder "Search".
- Time range:** Time range dropdown.
- Log Levels:** Log Levels (5/5) dropdown.
- Last deployment:** Last deployed 3 minutes ago - 2023-06-19 08:12 PDT.
- Logs:** A list of log entries with columns for level, timestamp, message, and source.
 - Info: a minute ago - 2023-06-19 08:13:29.026 PDT - CollectorAgentBase \$WrapperListener_start_runner - Registering listeners for application hellomulemoilu
 - Info: a minute ago - 2023-06-19 08:13:29.129 PDT - CollectorAgentBase \$WrapperListener_start_runner - Fast header injection enabled for app: hellomulemoilu
 - Info: a minute ago - 2023-06-19 08:13:29.228 PDT - ApplicationDeploymentListener \$WrapperListener_start_runner - Anypoint monitoring custom file appender disabled.
 - Warn: a minute ago - 2023-06-19 08:13:29.231 PDT - ApplicationDeploymentListener \$WrapperListener_start_runner - creating custom anypoint appender
 - Info: a minute ago - 2023-06-19 08:13:29.231 PDT - ApplicationDeploymentListener \$WrapperListener_start_runner - Disable console logging
 - Info: a minute ago - 2023-06-19 08:13:29.424 PDT - AbstractConnector \$WrapperListener_start_runner - Started ServerConnector@f6fd391{HTTP/1.1, (http/1.1)}{0.0.0.0:7777}
 - Info: a minute ago - 2023-06-19 08:13:30.246 PDT - ExtensionMessageSource \${!MuleRuntime}.uber.02: [hellomulemoilu].uber!org.mule.runtime.module.extension.internal.runtime.source.ExtensionMessageSource.lambda\$null\$17:435 @1c77e126 - Message source 'listener' on flow 'hello_mule_spssolutionsFlow' successfully started
 - Info: a minute ago - 2023-06-19 08:13:49.538 PDT - LoggerMessageProcessor - 6b0de4d5-b053-4d81-9de3-a0f8817bccee \${!MuleRuntime}.uber.02: [hellomulemoilu].hello_mule_spssolutionsFlow.CPU_LITE #6cc570c6 - Username: myUsernameDev -- Password: myPasswordDev

How to set up API autodiscovery

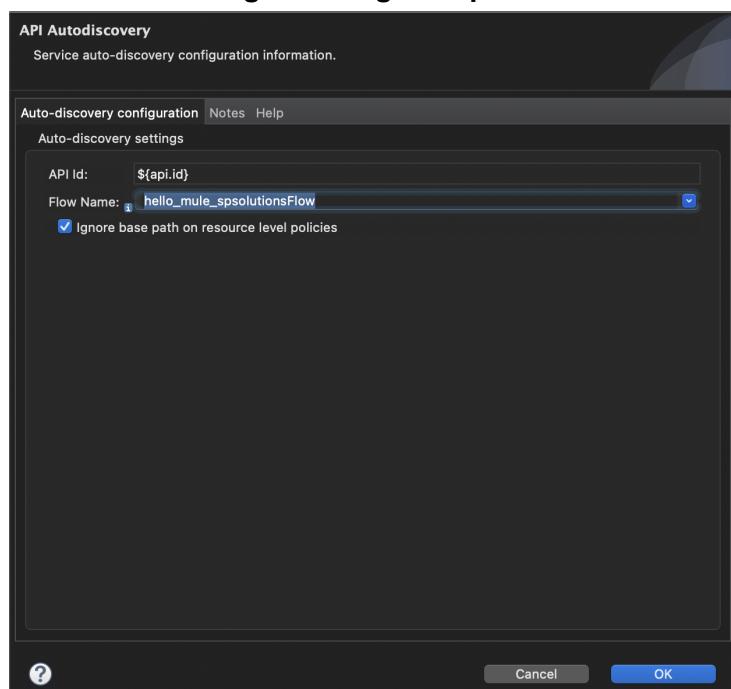
1. Creación de nueva API en AnyPoint

The screenshot shows the PI Administration (Sandbox) interface. On the left, there's a sidebar with options like SANDBOX, PI Administration, PI Groups, Automated Policies, Client Applications, Custom Policies, and Mule API Analytics. The main area is titled 'API' and has a sub-instruction 'Select the API you want to manage.' Below it are two buttons: 'Select API from Exchange' (radio button) and 'Create new API' (radio button, which is selected). A note below says 'Once the API is created it will be published in Exchange in stable state.' The 'Name' field contains 'hellomoluapi' and the 'Asset types' dropdown is set to 'HTTP API'. At the bottom right, there's an 'Advanced' link.

2. Agregando el app ID a la aplicación para configurar las policies(?)



3. Cambiando configuración global para el API autodiscovery



4. Copiando ID de cliente y secret de AnyPoint Platform

Edit environment

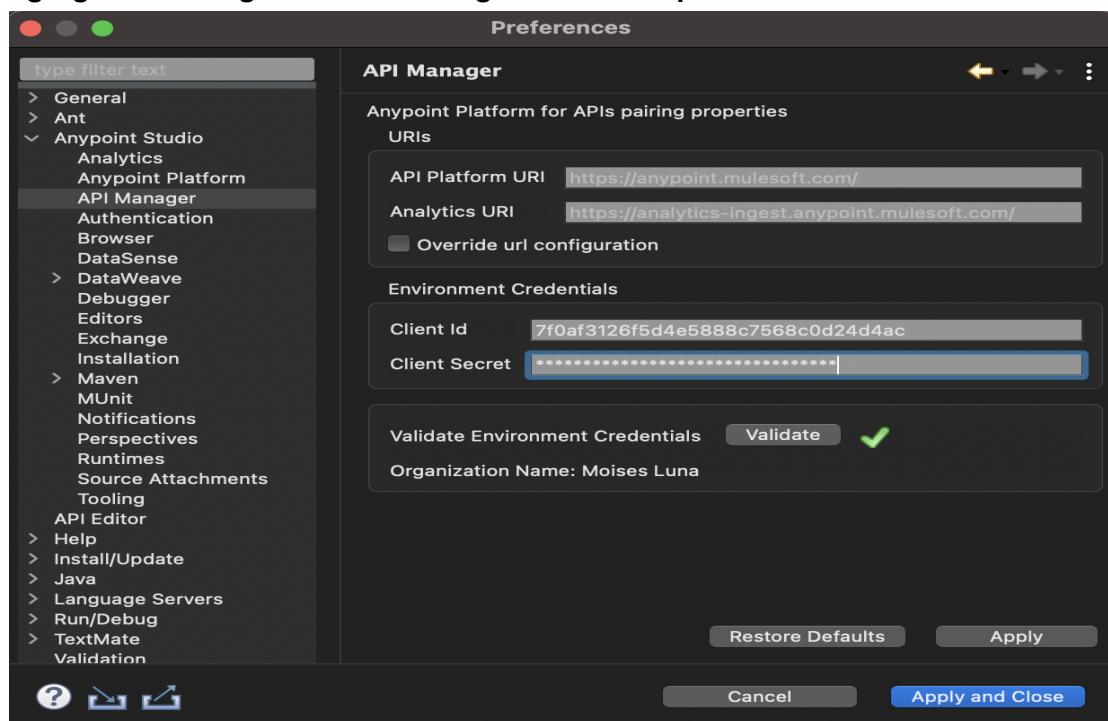
Name
Sandbox

Client ID
7f0af3126f5d4e5888c7568c0d24d4ac

Client Secret
.....
[Show](#)

[Delete Environment](#) [Cancel](#) [Update](#)

5. Agregando configuración de la organización de platform



6. Escondiendo ID y secret para que no sean visibles en la plataforma

```
al mule-artifact.json X
```

```
{  
    "minMuleVersion": "4.4.0",  
    "secureProperties": [  
        "secure.key",  
        "anypoint.platform.client_id",  
        "anypoint.platform.client_secret"  
    ]  
}
```

7. Despues de correr nuevamente la aplicación, reviso en postman que todo siga corriendo bien y se despliega otra vez a CloudHub

The screenshot shows the Anypoint Platform application configuration interface. At the top, there's a navigation bar with 'API Network' and 'Explore' tabs, a search bar, and various icons for invite, settings, notifications, and upgrade.

The main area displays a deployment configuration for a 'Hello Mule' application. It includes a table for 'Query Params' with columns for Key, Value, and Description. The 'Body' tab shows a successful response with status 200 OK, 534 ms, and 86 B. The response content is 'Hello Mule'.

Below the main configuration, there's a warning message: '⚠ 1 or more replicas in unexpected state: [Kubernetes] Container "init" - PodInitializing'. A note says 'Applying changes will create a new configuration for your application' with a 'Apply Changes' button.

The bottom section is titled 'Application File' and contains tabs for 'Deployment Target', 'Ingress', 'Properties', and 'Logging'. The 'Properties' tab is selected, showing a table view of properties. The properties listed are:

Key	Value	Type	Actions
env	dev	String	Protect ...
secure.key	MyMuleSoftKey	String	Protect ...
anypoint.platform.client_id	7f0af3126f5d4e5888c7568c0d24d4ac	String	Protect ...
anypoint.platform.client_secret	440b6fa1b3a0c4175e751a50d15055e4	String	Protect ...

8. Una vez que se despliega la aplicación se revisa en PostMan que todo esté bien.

The screenshot shows the deployment status of the application 'hellomulemoilu'. It indicates that the configuration 'ac18a8' is being applied, with 0 out of 1 replicas updated. The application status is 'Running' (green dot). Configuration details include ID 'd8e22f', last update '2023-06-19 9:10:56AM', and 1 replica started. Public endpoint is <https://hellomulemoilu-zzs5x3.5sc6y6-4.usa-e2.cloudhub.us>. Target name is 'Cloudhub-US-East-2' and target type is 'Shared Space'. A button at the bottom right says 'Apply changes'.

9. Finalmente se ve en API manager que el API se encuentra activa

The screenshot shows the API Manager interface with a single active API listed. The table has columns: Status, API Name, Runtime, Label, Version, Instance, Error Rate, Total Requests, and Client Applications. The API 'hellomoiluapi' is listed as Active, running on Mule 4, version v1, instance 18772222, with no error rate or requests data available.

Status	API Name	Runtime	Label	Version	Instance	Error Rate	Total Requests	Client Applications
Active	hellomoiluapi	Mule 4	-	v1	18772222	No data	No data	0

How to set up Client ID enforcement policy

1. Agregando nueva policy

The screenshot shows the 'APIs / hellomoiluapi / Policies / Add a policy' page. A search bar at the top contains the text 'Client'. Below it, a message says '1 results found'. A single result is listed: 'Client ID Enforcement' with a 'COMPLIANCE' badge. The description states: 'All calls to the API must include a client ID and client secret for an application that is register...'. A 'Learn more' link is present.

2. Configurando el Client enforcement policy

The screenshot shows the 'APIs / hellomoiluapi / Policies / Configure Client ID Enforcement policy' page. It includes sections for 'Credentials origin' (set to 'HTTP Basic Authentication Header'), 'Advanced options' (expanded to show policy version, methods, and resources), 'Policy version' (set to '1.3.2 (latest)'), 'Method & resource conditions' (set to 'Apply configuration to all API method & resources'), and a bottom row with 'Previous' and 'Apply' buttons.

3. Verifico que la configuración este correcta con postman. El error de autorizacion indica que sí

The screenshot shows a Postman request to 'https://hellomulemoilu-zzs5x3.5sc6y6-4.usa-e2.cloudhub.io/hellomule'. The method is 'GET'. The response status is 401 Unauthorized, with a duration of 1559 ms and a size of 296 B. The body of the response is JSON, showing the error message: "error": "Authentication denied."

4. Ir a la sección Exchange para obtener los valores necesarios de autenticación del API para que sea accesible desde PostMan

The screenshot shows the MuleSoft Exchange interface. At the top, there's a navigation bar with 'Exchange' and user information 'Moises Luna'. Below it is a card for the 'hellomoiluapi' API, which is an 'HTTP API' published by Moises Luna 3 hours ago. The card includes tabs for 'PAGES' (selected), 'Home', and 'OTHER DETAILS'. On the right side of the card, there are buttons for 'Share', 'Request access', 'Add version', and more. Below the card, there are sections for 'Moises Luna published 3 hours ago', 'Manage versions' (v1 Private), '1.0.x', 'Latest 1.0.0' (Stable), and a 'Tags' section.

5. Extraer llaves de acceso y agregarlas a PostMan

The screenshot shows the PostMan 'Request API access' dialog. It displays a message: 'Your request has been received and approved.' It instructs the user to use the provided Client ID and Client Secret to access the API instance. The Client ID is listed as '056dc41c2cd44a3b8c182524ea5a2231' and the Client Secret is a long, randomly generated string. Below this, there's a list of requests: 'GET Hello Mule' and 'GET http://localhost:8081/h'. The 'GET Hello Mule' request is highlighted with a red dot. To the right, there are buttons for 'No Environment', 'Save' (with a file icon), and 'Copy' (with a clipboard icon). The main area shows an API request configuration for a 'GET' method to 'https://hellomulemoilu-zzs5x3.5sc6y6-4.usa-e2.cloudhub.io/hellomule'. The 'Authorization' dropdown is set to 'Basic Auth'. The response body is shown as '200 OK' with a size of '820 ms' and '166 B'. The body content is 'Hello Mule'. A note at the bottom left says: 'The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)'.

Best practices to design your first API specification

Este tutorial explica que lo que son las especificaciones. En resumen es la planeación de los estándares que debe seguir un API que vas a crear.

Algunos de estos estándares son:

- Nombrar correctamente los recursos que va a devolver el API. Usar adjetivos en vez de verbos(/contacts)
- Definir el nombre y url base correctamente para que defina la industria correcta.
- Tener claro que métodos HTTP va a soportar.
- Mantener los query parameters en snake_case.
- Definir si el API tendrá funcionalidades de ordenamiento, filtración y paginación.

Build your first API specification

1. Creación de nueva especificación en el Design Center siguiendo la opción Guide Me

The screenshot shows the 'Design Center' interface. On the left, there's a sidebar with buttons for 'Redo' (highlighted), 'Share', 'Edit RAML', and 'Download'. The main area is titled 'API Summary' for the 'NTO Moilu Customer API'. It includes fields for 'Title' (set to 'NTO Moilu Customer API'), 'Version' (set to '1.0.0'), 'Protocols' (with dropdowns for 'Select...'), 'Media type' (with dropdowns for 'Select...'), 'Base URI' (an input field), and 'Base URI Parameters (0)'. Below these are sections for 'Description' (with a rich text editor toolbar and 'Markdown' tab selected) and 'Searched By'. On the far left, there are collapsed sections for 'SECURITY SCHEMES', 'RESOURCES', 'DATA TYPES', and 'GROUPS'.

2. Agregado de data types al especificación del API

The screenshot shows the API specification editor for the 'NTO Moilu Customer API'. The left sidebar shows collapsed sections for 'SECURITY SCHEMES', 'RESOURCES', 'DATA TYPES', and 'GROUPS'. Under 'DATA TYPES', the 'address' type is selected and expanded. The right panel shows the 'address' data type definition with a 'Name' field ('address') and a 'Type' field ('Object'). Below this, the 'Description' section contains the text 'Address data type'. At the bottom, there are three property definitions: 'street' (String, Required), 'city' (String, Required), and 'postalCode' (String, Required). Each property has a delete icon to its left.

Undo | Redo | Saved a few seconds ago | NTO Moilu Customer API

Filter

API Summary

- ✓ SECURITY SCHEMES +
- ✓ RESOURCES +
- ✓ DATA TYPES +
- address Delete
- contact** Delete
- ✓ GROUPS +

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

Properties

- Property Name** email **Type** String Required Union ↗ ↘
- Property Name** deliveryAddress **Type** address Required Union ↗ ↘
- Property Name** postalAddress **Type** address Required Union ↗ ↘

Add Property

Example

```
{
  "firstName": "Example",
  "lastName": "Example",
  "phone": "Example",
  "email": "Example",
  "deliveryAddress": {
    "street": "Example",
    "city": "Example",
    ...
  }
}
```

Inherited Edit

3. Agregando un recurso nuevo a la especificación

/customers

● GET ○ POST ○ PUT ○ PATCH ○ DELETE ○ OPTIONS ○ HEAD

Delete

Responses (1)

Delete	200	Status
	200 - OK	▼
Add	Description	
	B I » </> ≡ ≡ H Markdown Visual	

Bodies (1)

Delete	Media Type	Type
	application/json	▼
		Array ▼
		<input type="checkbox"/> Union ↗ ↘

Bodies (1)

Media Type	Type
application/json	Array

Items

Items Type
contact

Add Property

Example

```
{
  "firstName": "Example",
  "lastName": "Example",
  "phone": "Example",
  "email": "Example",
  "deliveryAddress": {
    "street": "Example Street",
    "city": "Example City",
    "state": "Example State",
    "zip": "Example Zip"
  }
}
```

Inherited Edit

ago NTO Moilu Customer API

Resource path

/customers/{customerId} URI Parameters (1) ▾

URI Parameters (1)

URI Parameter Name	Type
customerId	String

Add URI Parameter

GET POST PUT PATCH DELETE OPTIONS HEAD

Summary Responses (0) Query Parameters (0) Headers (0)

Name Secured By

Get customer by id Select...

The screenshot shows the 'Responses (1)' tab selected in the API configuration interface. A single 200 status code entry is present with the description '200 - OK'. Below it, there's a rich text editor toolbar with 'Markdown' selected. Under 'Bodies (1)', a media type is set to 'Default (application/json)' and the type is 'contact'. There are also options for 'Union' and a dropdown menu.

4. Poniendo a prueba el API con el mocking service

The screenshot shows the 'Try It' interface for the '/customers' endpoint. The method is set to 'GET'. The 'Request URL' is 'https://anypoint.mulesoft.com/mockng/api/v1/so'. The response is a 200 OK status with a JSON payload representing a customer record. The JSON output is as follows:

```

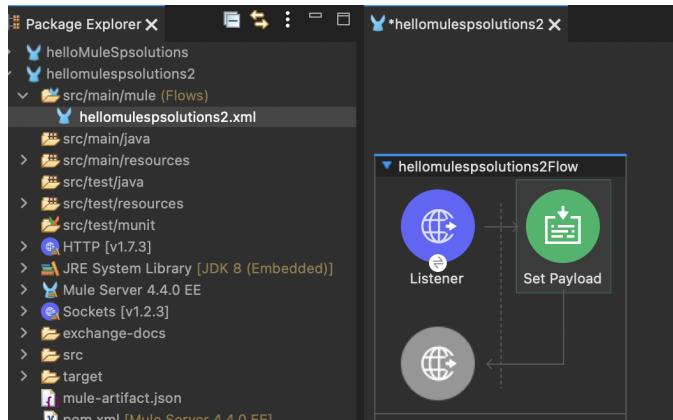
1  [
2   {
3     "firstName": "Example",
4     "lastName": "Example",
5     "phone": "Example",
6     "email": "Example",
7     "deliveryAddress": {
8       "street": "Example",
9       "city": "Example",
10      "postalCode": "Example",
11      "state": "Example",
12      "country": "Example"
13    },
14    "postalAddress": {
15      "street": "Example",
16      "city": "Example",
17      "postalCode": "Example"
18    }
19  ]
  
```

5. Especificación en Exchange

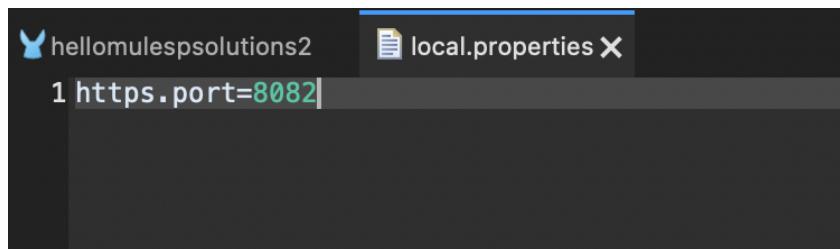
<https://anypoint.mulesoft.com/exchange/f4b04931-7179-4b7c-8a4a-7cc8183edf3e/nto-moilu-customer-api/minor/1.0/>

How to configure an HTTPS endpoint in Anypoint Studio

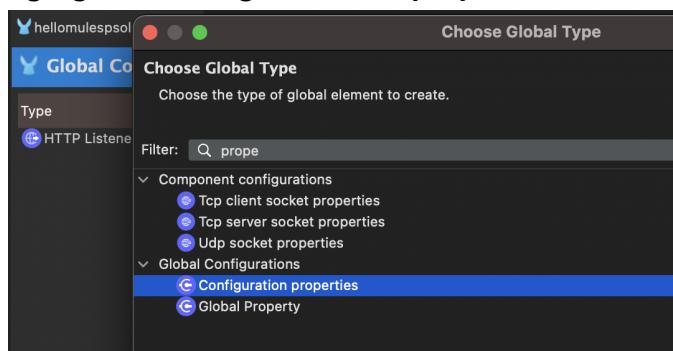
1. Creación de nuevo proyecto



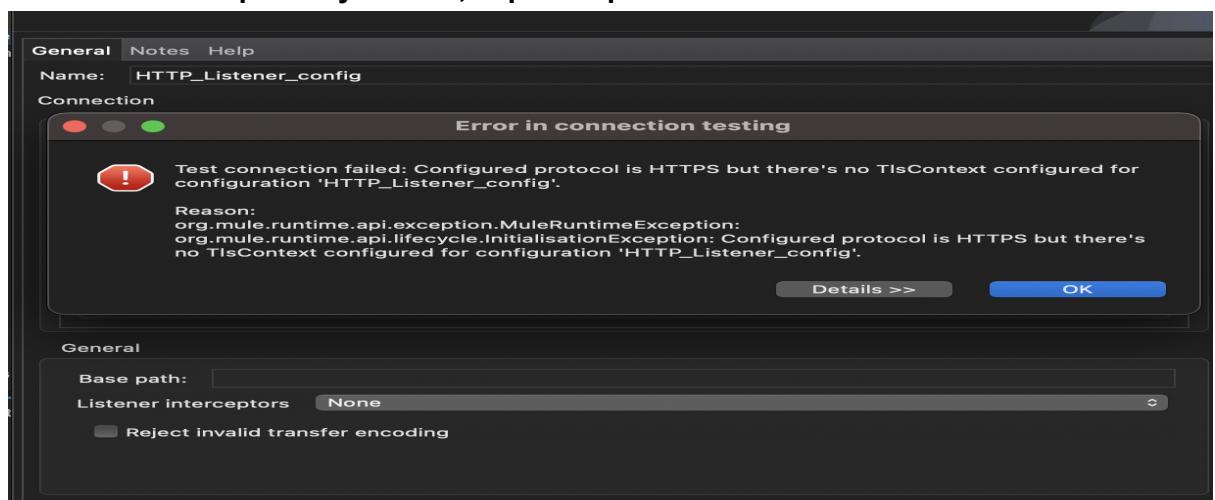
2. Agregado archivo de propiedades en recursos con configuración de puerto https



3. Agregando configuración de propiedades



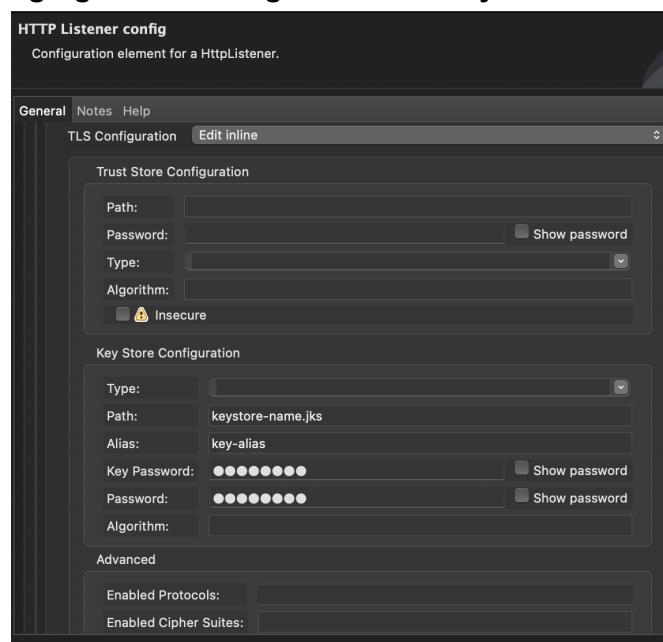
4. Se selecciona el puerto y HTTPS, la primer prueba de conexión debe fallar



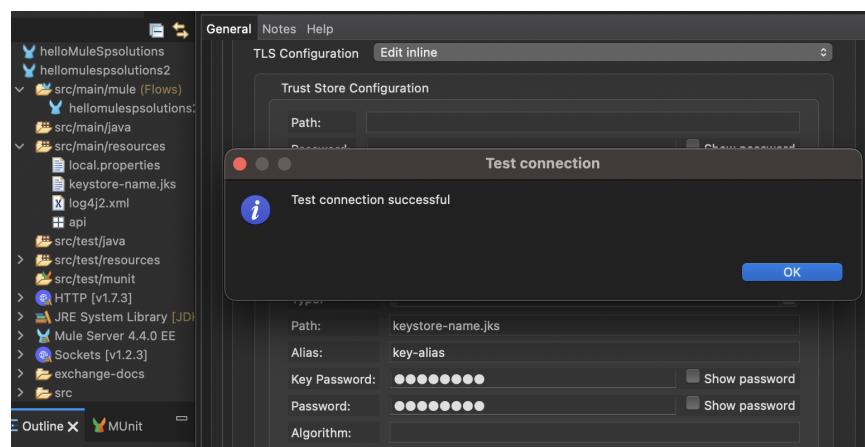
5. Se debe arreglar la configuración TLS para que no falle la conexión. Para esto primero se agrega un keystore

```
→ NTO-Moilu-Customer-API git:(main) ✘ ls  
NTO_Moilu_API.raml README.md keystore-name.jks  
→ NTO-Moilu-Customer-API git:(main) ✘
```

6. Agregado de configuración del keystore



7. Prueba de conexión



How to configure an HTTPS endpoint in Anypoint Studio

Pendiente