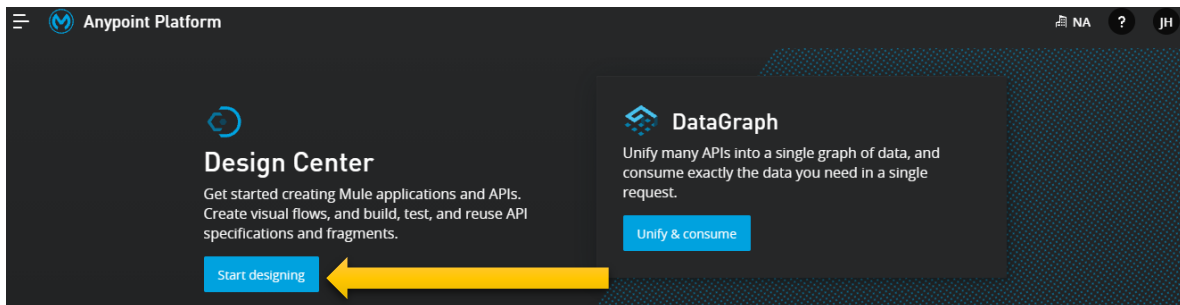
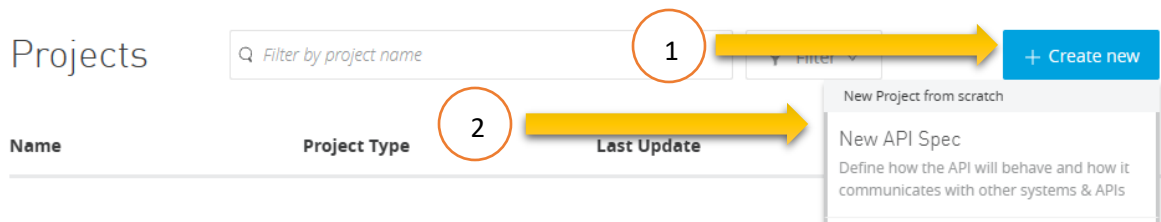


Despues de crear tu cuenta en Mulesoft nos desplegara elsiguiente menu donde debemos hacer click en “start designing”:



Creamos un nuevo proyecto y seleccionamos la API Spec:



Establecemos las especificaciones:

The screenshot shows the 'New API Spec' form in MuleSoft. The form includes fields for 'API Title' (Practica MuleSoft Trainee), 'How do you want to draft the API Spec?' (I'm comfortable designing it on my own), and 'Specification Language' (RAML 1.0). The 'Guide me through it' option is selected. The 'Create API Spec' button is highlighted. The form has a light gray background with a white border.

Se llenan los parámetros, el nombre, los protocolos, la URI y alguna descripción:

The screenshot shows a configuration form for an API. At the top, the title 'Practica MuleSoft Trainee' and version '1.0.0' are displayed. Below this, the 'Protocols' section has buttons for 'HTTP' and 'HTTPS', and a dropdown arrow. The 'Media type' section has a dropdown menu showing 'application/json'. The 'Base URI' field contains 'api.samplebasseuri.com', and there is a button to 'Add Base URI Parameter'. The 'Description' section features a rich text editor with 'Practica: MuleSoft Trainee' entered. Below the description is a 'Secured By' dropdown menu. At the bottom, there is a 'Documentation (0)' section with an 'Add Documentation' button.

En la parte de RAML obtenemos los siguientes lineamientos:

```
##RAML 1.0
title: Practica MuleSoft Trainee
baseUri: api.samplebasseuri.com
description: "Practica: MuleSoft Trainee"
mediaType:
  - application/json
version: 1.0.0
protocols:
  - HTTP
  - HTTPS
```

Y esto en la parte de OAS:

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "Practica MuleSoft Trainee",
    "description": "Practica: MuleSoft Trainee",
    "version": "1.0.0"
  },
  "servers": [
    {
      "url": "api.samplebasseuri.com"
    }
  ],
  "paths": {}
}
```

Creamos el DataType **adress** y **contact**:

Data types

Name	Type	Union?
address	Object	<input type="checkbox"/>

Description

B *I* » </> |≡ |≡ H Markdown Visual

Adress data type

Data types

Name	Type	Union?
contact	Object	<input type="checkbox"/>

Description

B *I* » </> |≡ |≡ H Markdown Visual

Adress data type

Se agregan propiedades:

Property Name	Type
firstName	S... <input type="checkbox"/> Required?
lastName	S... <input checked="" type="checkbox"/> Required?
phone	S... <input type="checkbox"/> Required?
email	S... <input type="checkbox"/> Required?
deliveryAddress	a... <input type="checkbox"/> Required?
postalAddress	a... <input type="checkbox"/> Required?

Add Property

Property Name	Type
street	S... <input type="checkbox"/> Required?
city	S... <input type="checkbox"/> Required?
postalCode	S... <input type="checkbox"/> Required?
state	S... <input type="checkbox"/> Required?
country	S... <input type="checkbox"/> Required?

Add Property

1.- El campo de apellido es obligatorio para este objeto

2.- La dirección de envío y el código postal son de tipo "address"

Y copiamos los edits de las instrucciones:

```
"street": "44 Shirley Ave.",  
"city": "West Chicago",  
"postalCode": "60185",  
"state": "IL",  
"country": "USA"
```

```
{  
  "firstName": "Danny",  
  "lastName": "Brookshire",  
  "phone": "123-412-3412",  
  "email": "danny.brookshire@example.com",  
}
```

```

"deliveryAddress": {
  "street": "44 Shirley Ave.",
  "city": "West Chicago",
  "postalCode": "60185",
  "state": "IL",
  "country": "USA"
}

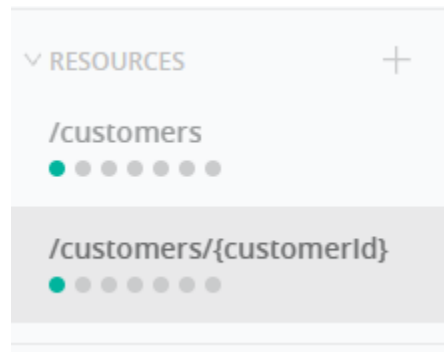
```

```

value:
  firstName: Danny
  lastName: Brookshire
  phone: 123-412-3412
  email: danny.brookshire@example.com
  deliveryAddress:
    street: 44 Shirley Ave.
    city: West Chicago
    postalCode: "60185"
    state: IL
    country: USA
  postalAddress:
    street: 44 Shirley Ave.
    city: West Chicago
    postalCode: "60185"
    state: IL
    country: USA

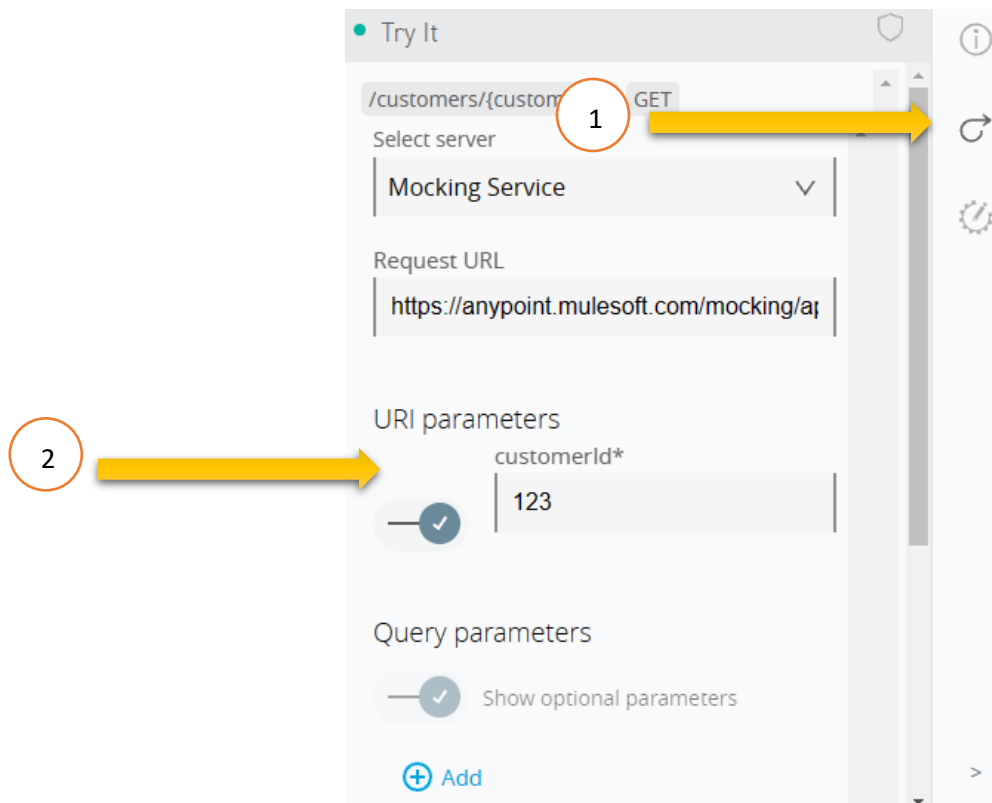
```

Después creamos dos recursos con solamente “get”:

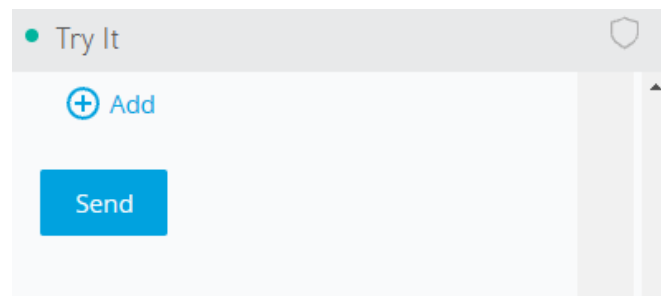


Con esto creamos dos get endpoints para la API de cliente el primero devolverá una matriz para todos los clientes, y la segunda devolverá un solo objeto de contacto que dará la URL de la solicitud.

Después para probarla podemos hacer click en el botón de “Try It” (1) donde nos dará diferentes parámetros y en los parámetros de URI ponemos un ejemplo como “123” (2), de la siguiente manera.



Después le damos en enviar:



Y nos desplegara esto:

200 OK

Time: 678 ms



```
1  {
2    "firstName": "Danny",
3    "lastName": "Brookshire",
4    "phone": "123-412-3412",
5    "email": "danny.brookshire@
6    "deliveryAddress": {
7      "street": "44 Shirley Av
8    e.",
9      "city": "West Chicago",
10     "postalCode": "60185",
11     "state": "IL",
12     "country": "USA"
13   },
14   "postalAddress": {
15     "street": "44 Shirley Av
16   e.",
17     "city": "West Chicago",
18     "postalCode": "60185",
19     "state": "IL",
20     "country": "USA"
  }
```

Despues publicamos la API.

Publish ▾



Publish to share with your organization and business group, or make publicly available.

Last published 0 days ago by Jorge Alberto Ayala.

[View in Exchange](#)

Publish to Exchange

Escogemos una versión y seleccionamos “Publish to Exchange”:

Practica MuleSoft Trainee

Asset version (required)

1.0.1

1.0.0 published 0 days ago

API version (required)

1.0.0

Specified in root file

> More options

Asset versioning

To publish to Exchange, you must version assets using [SemVer](#). Examples of good versions are 1.0.0 or 4.3.1.

Additional help

- [Changing a project's main/root file](#)
- [What is an API version?](#)



Damos click en Exchange y nos dirige ala siguiente pagina

The screenshot shows the MuleSoft Exchange interface. On the left is a sidebar with navigation links: Assets list, PAGES, Home, SPECIFICATION, Summary, Endpoints, /customers, /(customerId), Types, OTHER DETAILS, and API Instances. The main content area displays the asset 'Practica MuleSoft Trainee' with version 1.0.0. It includes a star rating (5 stars), a 'Rate and review' link, and a description field. A 'Contact information' modal is open, showing fields for contact name and email. On the right, there are tabs for 'Share', 'Download', and 'Edit'. Below these, there are details about the asset: Type (REST API), Organization (NA), Published by (Jorge Alberto Ayala Hernandez), Published on (Oct 8, 2021), Visibility (Private), and a table of asset versions for 1.0.x. At the bottom, there is a cookie consent banner.

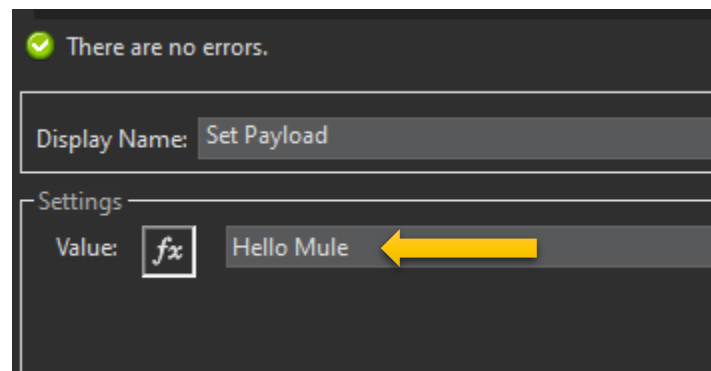
Con esto se crean las especificaciones de la API.

Parte 1:

Para iniciar con la primera API se necesita un Listener de HTTP el cual se encuentra en Mule Palette, para escuchar una solicitud HTTP y este flujo se ejecutará definiendo una ruta para este:



Y agregamos un set deploy que se ejecutara en el flujo de la petición a nuestra ruta con un mensaje que queramos:




Al ejecutar estos procesos y ser implementada nos aparece el siguiente mensaje en consola:

```
*****
*      - - + APPLICATION + - -      *      - - + DOMAIN + - -      *      - - + STATUS + - - *
*****
* hellomule                        * default                        * DEPLOYED                        *
*****
```

Al hacer el POST en nuestro REST nos aparecerá lo siguiente según nuestras especificaciones en la petición:



Body Cookies Headers (2) Test Results 🌐 Status: 200 OK Time: 1355 ms


Pretty Raw Preview Visualize Text 

1 Hello Mule


Sybimos la aplicación a CloudHub:

SANDBOX




Deploying Application


hellomu 

Deployment Target

CloudHub 

Application File


Runtime	Properties	Insight	Logging	Static IPs
Runtime version	Worker size		Workers	
4.4.0 	0.1 vCores 		1 	

 To use Monitoring and Visualizer with this version, you may need to enable the agent after deploying. [Learn how](#)

☒ Automatically restart application when not responding

☐ Persistent queues | ☐ Encrypt persistent queues

☒ Use Object Store v2

 Your current subscription allows only one worker per application

Cancel

Deploy Application

Podremos visualizarla en el buscador


 hellomu

Domain: hellomu.us-e2.cloudhub.io · Last Updated 2021-10-08 9:17:25PM · 1 micro worker, using 4.4.0

Mule messages

Y desde el REST mandarla a llamar con el POST:

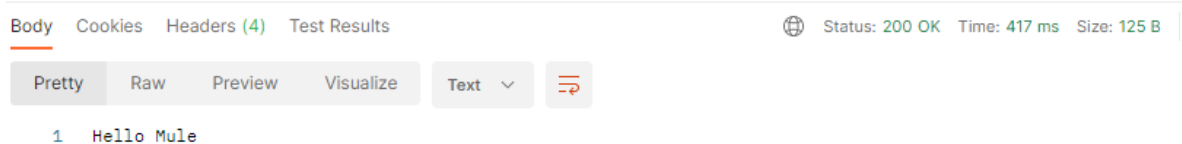
hellomu.us-e2.cloudhub.io/hellomule

POST 

hellomu.us-e2.cloudhub.io/hellomule

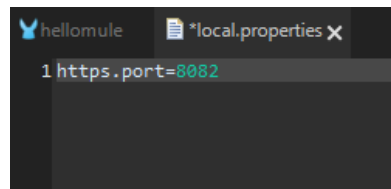
Params Authorization Headers (7) Body Pre-req

Query Params

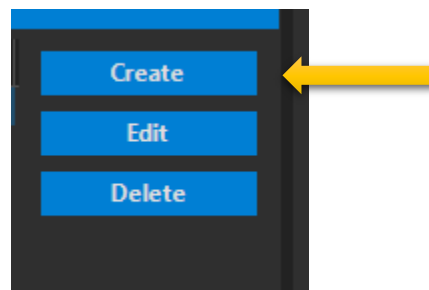


Parte 2:

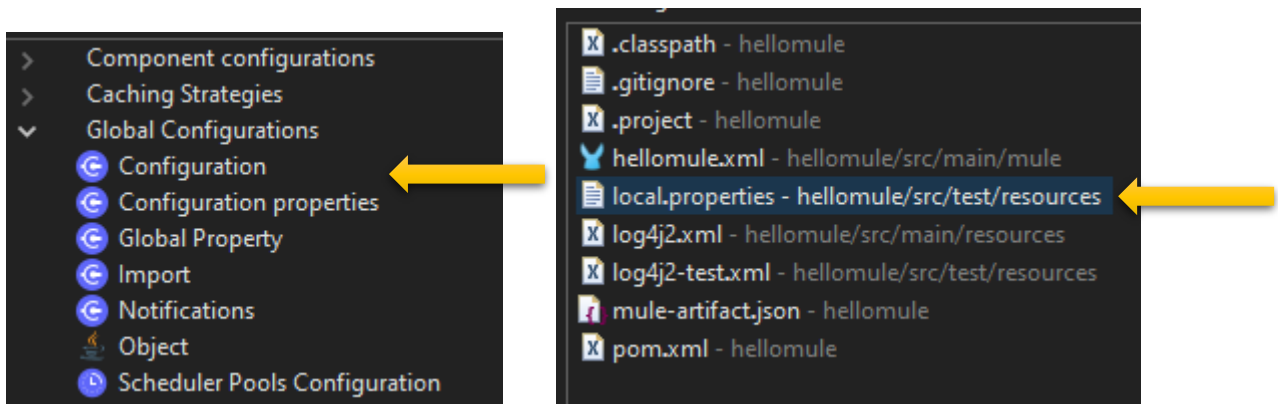
Para configurar en HTTPS debemos crear un archivo en recursos llamado "local.properties" y determinar el puerto https:



Damos click en "Create"



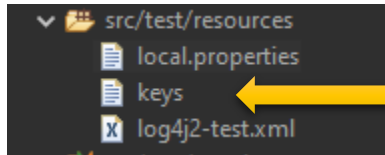
Escogemos la opción de "global configuration" y después "Configuration properties" y seleccionamos el archivo que creamos:



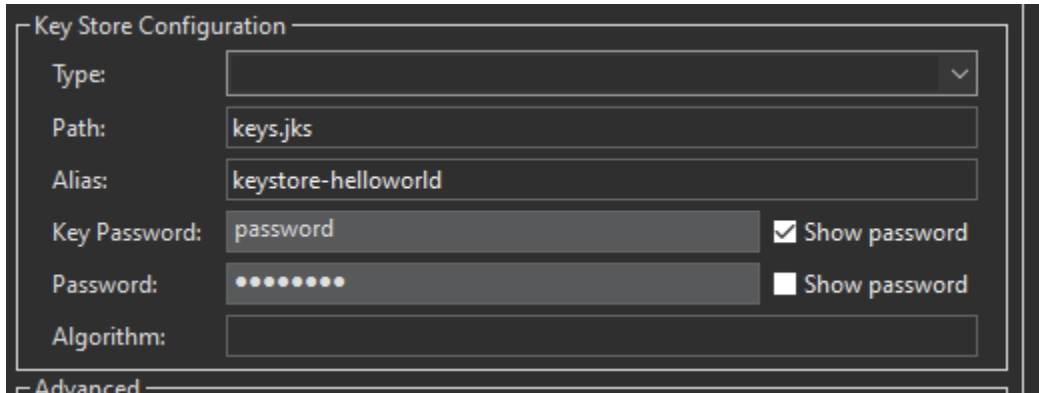
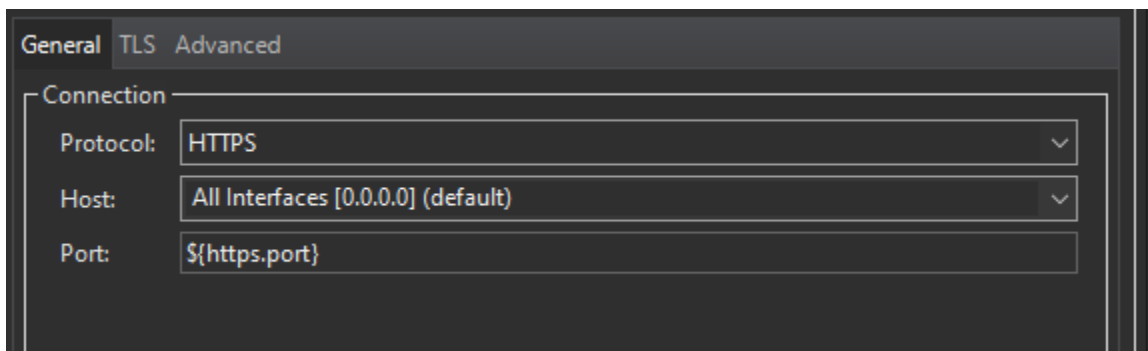
Generamos el JKS mediante la siguiente linea de comando:

```
C:\Program Files\Java\jdk1.8.0_111\bin>keytool -genkey -alias keystore-helloworld.jks -keystore "C:\Users\jorhe\OneDrive\Documentos\keys" -keyalg RSA -storetype JKS
```

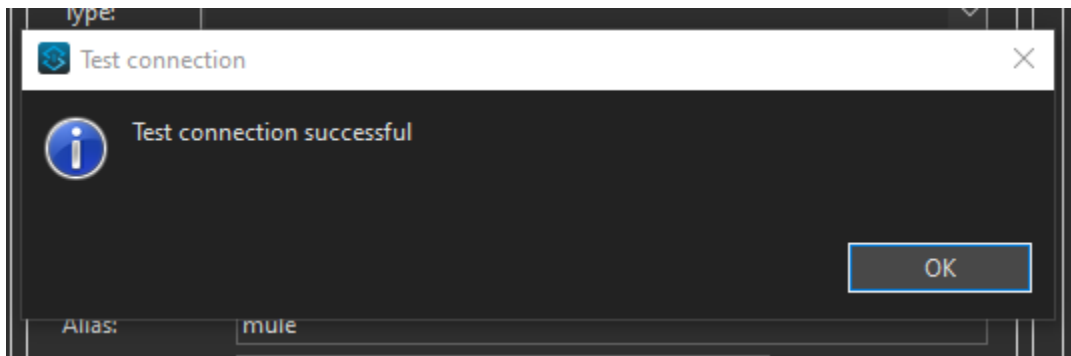
El archivo creado lo movemos a la carpeta de main/resources:



Y hacemos las siguientes configuraciones:



Al realizar el test nos debe aparecer como exitoso:

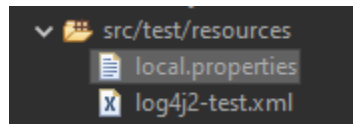


Al hacer el Post en nuestro REST obtenemos una respuesta de nuestra API:



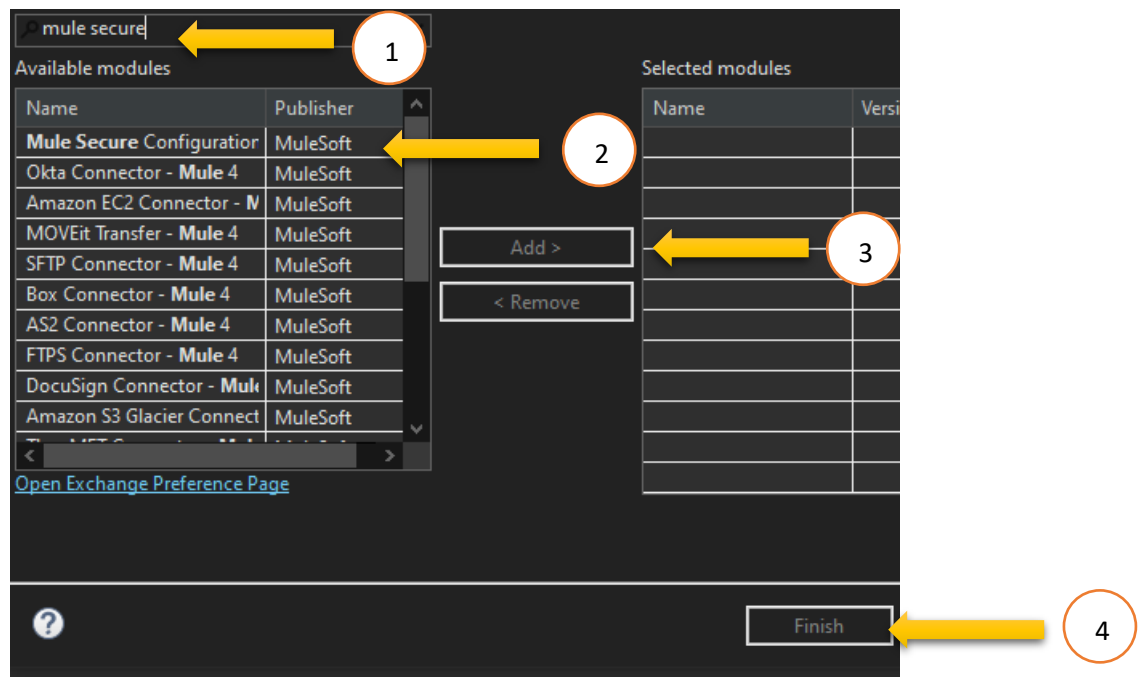
Parte 3:

Para crear las propiedades de seguridad, creamos un archivo "local.properties" con los siguientes parámetros:

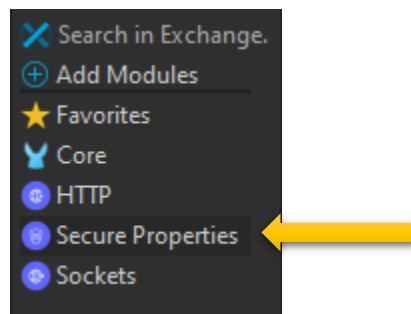


```
1 salesforceUsername=*****
2 salesforcePassword=*****
```

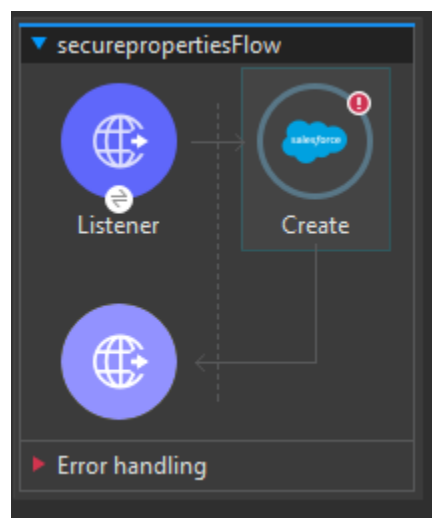
Posteriormente en searching Exchange buscamos mule secure y lo añadimos:



Nos aparece la siguiente opción:



Añadimos un elemento “create” de “Salesforce” a nuestro proceso:



Hacemos la configuración del create:

The screenshot shows the 'Salesforce_Config' configuration window. The 'General' tab is selected. The 'Name' field is 'Salesforce_Config'. The 'Connection' is 'Basic Authentication'. Below this, there is a sub-section with tabs: 'General', 'Connection Pool Config', 'Security', and 'Advanced'. The 'General' sub-tab is selected. It contains the following fields:

- Username: `fx` `${secure::salesforceUsername}`
- Password: `fx` `${secure::salesforcePassword}` ☒ Show password
- Security token: `fx`
- Authorization URL: `fx` `https://login.salesforce.com/services/Soap/u/52.0`

Creamos las siguientes propiedades:

The screenshot shows the 'Global Property' configuration window. It contains a description: 'A global property is a named string. It can be inserted in most attribute value (ant-style) Spring placeholders.' Below this, there is a sub-section with tabs: 'General', 'Notes', and 'Help'. The 'General' sub-tab is selected. It contains the following fields:

- Name: `secure.key`
- Value: `MuleSoft`

The screenshot shows the 'Secure Properties Config' configuration window. The 'General' tab is selected. The 'Name' field is 'Secure_Properties_Config'. Below this, there is a sub-section with tabs: 'General', 'Advanced', 'Notes', and 'Help'. The 'General' sub-tab is selected. It contains the following fields:

- File: `local.properties`
- Key: `fx` `${secure.key}`
- File level encryption: `False (Default)`
- Encoding: `fx`

Below the 'General' sub-section, there is an 'encrypt' sub-section with the following fields:

- Algorithm: `fx` `Blowfish`
- Mode: `fx` `CBC (Default)`
- ☐ Use random IVs

En los settings de créate hacemos las siguientes configuraciones:

The screenshot shows the 'Salesforce_Config' settings window. The 'General' tab is selected. The 'Name' field is 'Salesforce_Config'. The 'Connection' is 'Basic Authentication'. Below this, there is a sub-section with tabs: 'General', 'Connection Pool Config', 'Security', and 'Advanced'. The 'General' sub-tab is active. It contains the following fields:

- Username:** A field with a 'fx' icon and the value `${secure::salesforceUsername}`.
- Password:** A field with a 'fx' icon and the value `${secure::salesforcePassword}`. To its right is a checkbox labeled 'Show password' which is checked.
- Security token:** A field with a 'fx' icon and an empty value.
- Authorization URL:** A field with a 'fx' icon and the value `https://login.salesforce.com/services/Soap/u/52.0`.

Encriptamos la contraseña y el nombre de usuario:

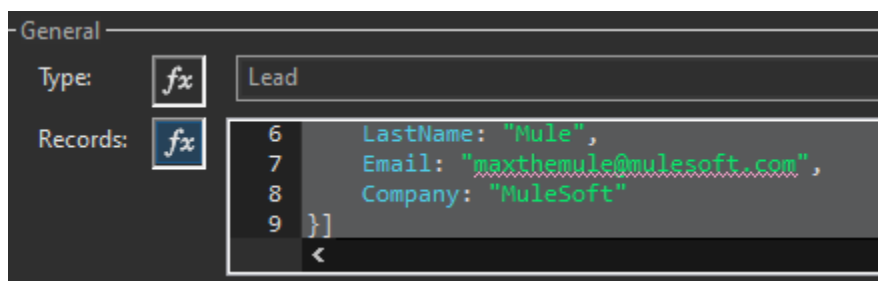
The screenshot shows a dialog box titled 'Add a new property' with a close button (X) in the top right corner. The dialog contains the following elements:

- A text prompt: 'Type the key and the value of the property to create.'
- A 'Key:' field with the value `salesforceUsername`.
- A 'Value:' field with the value `jorheay@gmail.com`.
- An 'Encrypt' button.

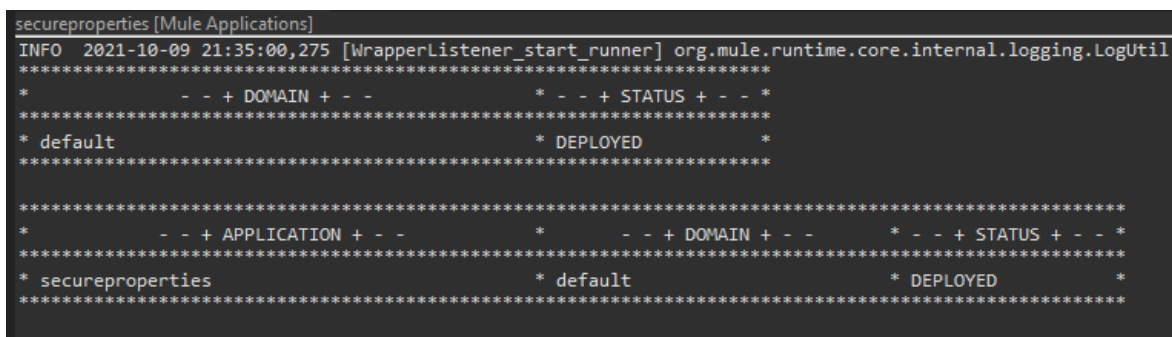
At the bottom of the dialog, there are 'OK' and 'Cancel' buttons. In the background, a table editor is visible with the following data:

Name	Value
salesforcePassword	password
salesforceUsername	jorheay@gmail.com

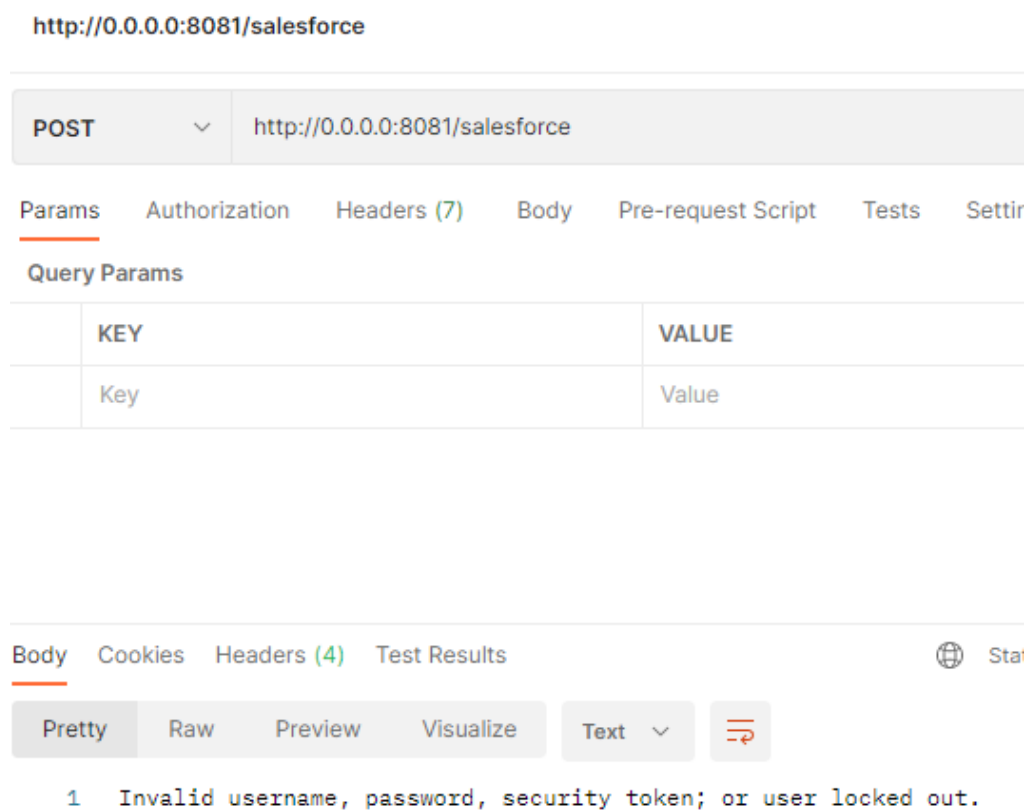
Completamos el create con el código de datawave:



Al ejecutar nos aparece el mensaje de éxito:

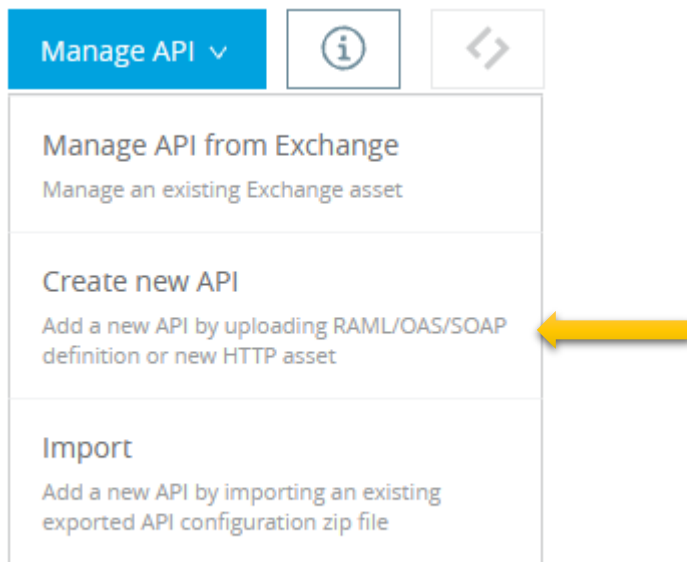


Respuesta de nuestro REST:



Parte 4:

Nos dirigimos a API administration y creamos una nueva API:



Configuramos nombre y tipo:

Creating an asset

Name Nueva API

Asset types ⓘ HTTP API

Templates and Example

Managing type: ☒ Basic Endpoint ☐ Endpoint with Proxy

Application type:

- ☒ Mule application
Running on Hybrid or CloudHub
- ☐ Non-Mule application managed by **Anypoint Service Mesh** New
Manage Kubernetes-based non-Mule microservices with Anypoint Service Mesh [Learn more about Anypoint Service Mesh](#)

Mule version: ☒ Mule 4
Recommended

Al continuar nos dará un API ID

Nueva API v1

API Status: ● Unregistered

Asset Version: 1.0.0

Latest ⓘ

Type: HTTP

[⊕ Add consumer endpoint](#)

API Instance ⓘ

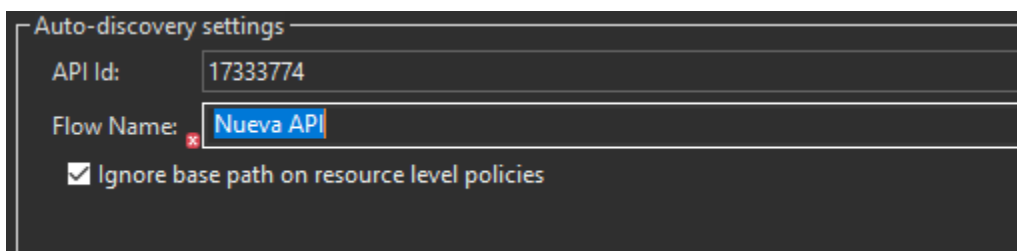
ID: 17333774

Label: [⊕ Add a label](#)

Autodiscovery ⓘ

API ID: 17333774

En AnyPoint studio creamos un API autodiscovery y lo llenamos con el ID de nuestra API y el nombre:



Auto-discovery settings

API Id: 17333774

Flow Name: ✖ Nueva API

☒ Ignore base path on resource level policies

Obtenemos nuestro client ID para subirla ligada al CloudHub

Edit environment

Name

Sandbox

Client ID

d781570b0d1a4f9e8a5ecd29b29b975c

Client Secret

765BC371AbC44C90a78D3241DA37c007

[Hide](#)

Delete Environment

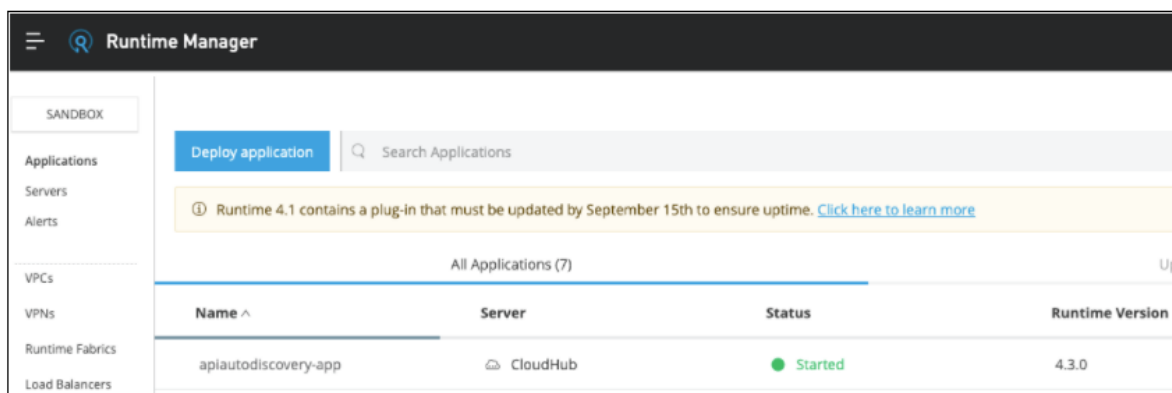
Cancel

Update

Y hacemos el Deploy de la aplicación:



Y esta se ve reflejada en la plataforma:



Parte 5:

Creamos el API:

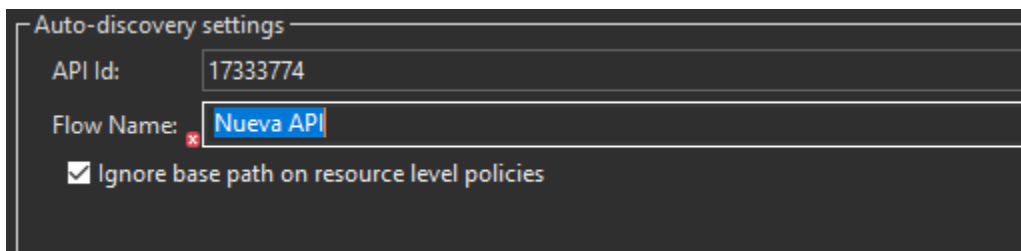


Configuramos la autenticación con el snippet que nos proporciona Anypoint:



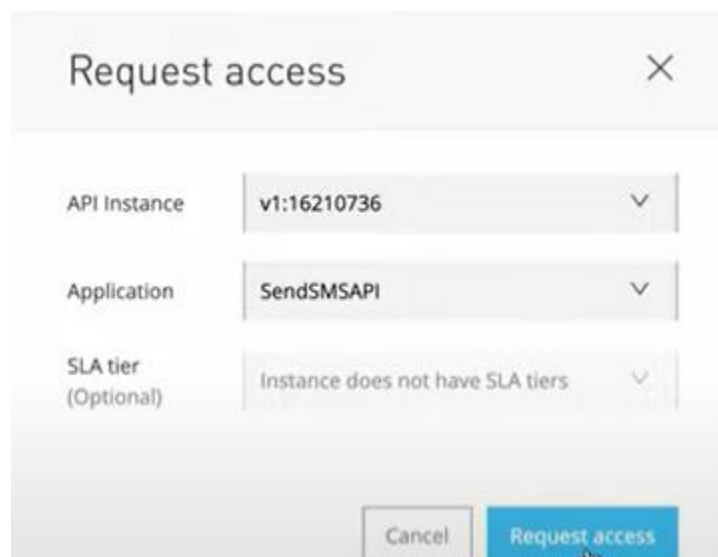
Donde tenemos dos String que son obligatorios que son el Client_ID y el client_secret que son requeridos para acceder a la API

En nuestro AnyPoint studio creamos un autodiscovery e introducimos el ID y el nombre de la API:



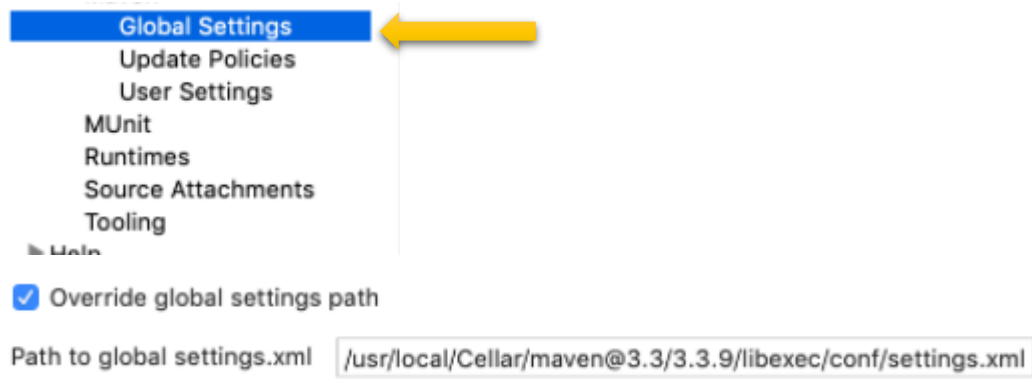
Y hacemos el Deploy en Cloud Hub en propiedades ponemos el Client_ID y el client_secret para poder hacer uso de la API

Asi podemos hacer el manejo de quien usa nuestra API y controlar o restringir u acceso:

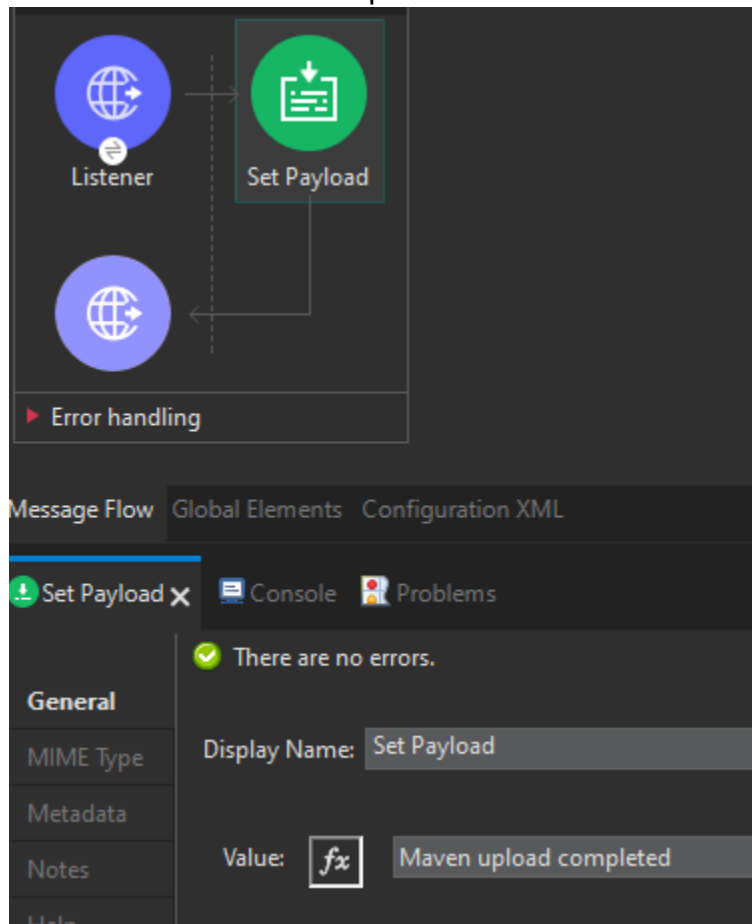


Parte 6:

Configuramos nuestro Anypointstudio con Maven:



Creamos un proyecto con un listener y una ejecución que diga “Maven upload completed”



Vamos a terminal y ejecutamos los comandos siguientes:

```
C:\Users\jorhe\AnypointStudio\studio-workspace>maven-tutorial
```

```
>mvn clean package deploy -DmuleDeploy
```

Cuando el API se encuentre en nuestra CloudHub la podemos ver en el manager:

Load Balancers

maven-tutorial

 CloudHub

Y si accedemos a la URL obtenemos salida:

```
"Maven upload completed"
```