

Documentación de la Práctica

Mulesoft Trainee

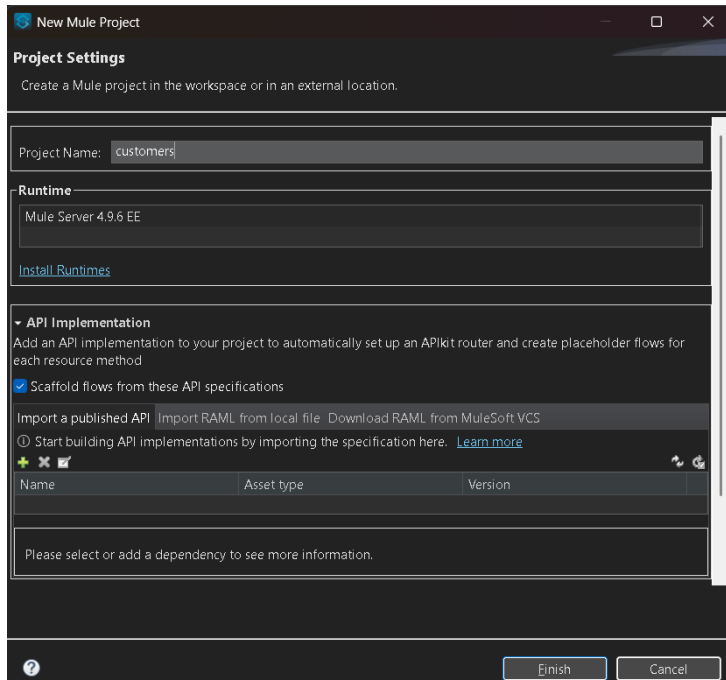
Proyecto Creado en AnypointStudio
para realizar la consuta a una base de
datos mediante una API

Realizado por Julio César Méndez
Torres

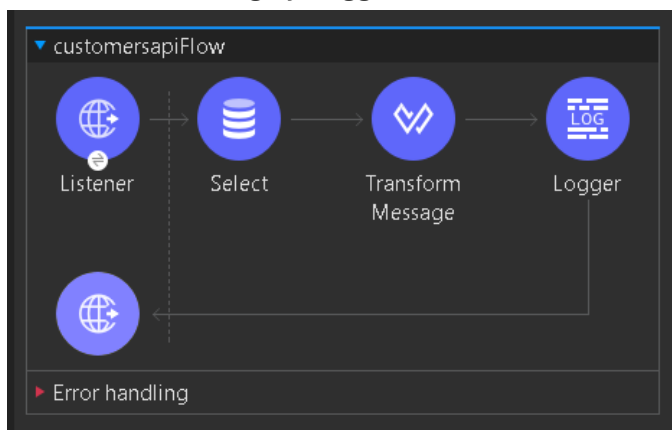
Indice

1. Creación del proyecto.....	1
2. Curtomers.xml.....	2
3. Creación de archivos local y dev.....	5
4. Global.xml.....	6
5. Especificación API.....	7
6. Desplegar el proyecto en Cloudhub.....	8
7. Api Manager.....	9

1. Se crea un proyecto en el IDE AnypointStudio con el nombre “customers”



2. En customers.xml se crea un canvas con los componentes Listener, Select, Transform Message y Logger



En Listener se asigna la ruta /api/v1/sps/customers donde se verán las peticiones HTTP

Display Name: Listener

Basic Settings

Connector configuration: HTTP_Listener_config

General

Path: /api/v1/sps/customers

En Select se colocan los parámetros para realizar la conexión a la base de datos y también se añade el query para hacer la consulta a la tabla customers

Global Element Properties

Database Config

Default configuration

General Advanced Notes Help

Name: Database_Config

Connection: MySQL Connection

General Transactions Advanced

Required Libraries

MySQL JDBC Driver (mysql:m...-connector-java: Modify...

Connection

Host: \$(db.host)

Port: \$(db.port)

User: mule

Password: Show password

Database: \$(db.name)

Test Connection... OK Cancel

Display Name: Select

Basic Settings

Connector configuration: Database_Config

Query

SQL Query Text

SELECT * FROM customers;

Y en metadata se creo un ejemplo de un JSON con las mismas columnas de la tabla de la base de datos que esperamos obtener

+ Add - Delete

Type: JSON

User Defined

base : Array<Object>

Example: examples/json-BD.json

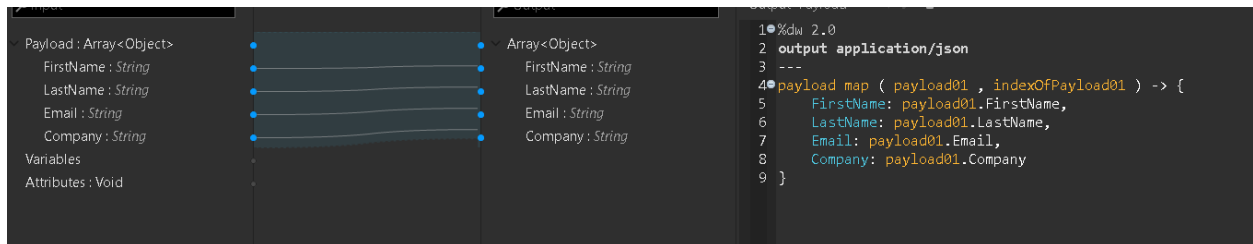
FirstName : String

LastName : String

Email : String

Company : String

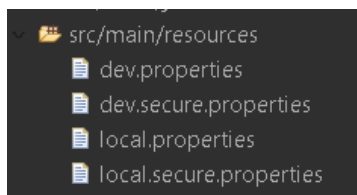
En Transform Message, la información obtenida de la base de datos se convierte a tipo JSON siguiendo el mismo ejemplo que creamos anteriormente



Y se crea un Logger con el mensaje “Información recibida”, que se mostrara en la consola si todo sale bien



3. En src/main/resources se crean 4 archivos



local.properties y dev properties donde crearemos variables de los parámetros de conexión de la base de datos, donde serán colocados en el Selct de nuestro canvas

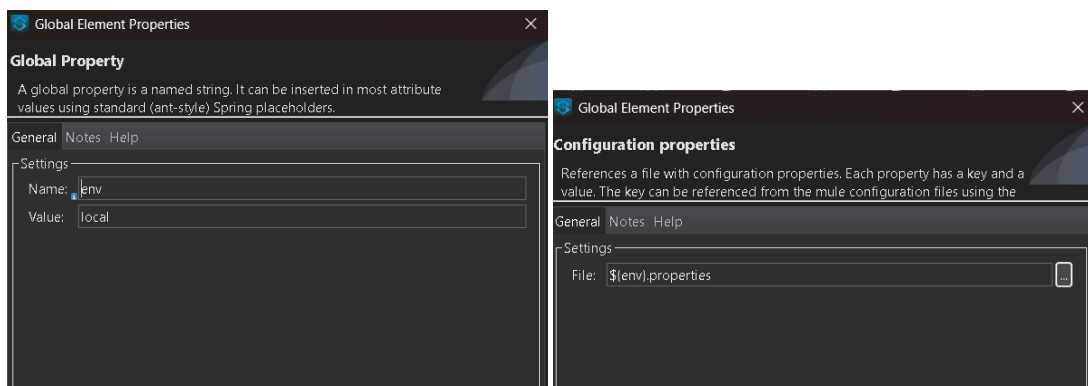
```
1 db.host=mudb.learn.mulesoft.com
2 db.port=3306
3 db.name=training
4 db.user=${secure::db.user}
5 db.password=${secure::db.password}
6
```

dev.secure.properties y local.secure.properties es donde crearemos las variables user y password cifradas previamente con secure-properties-tool.jar que ejecutaremos en la terminal

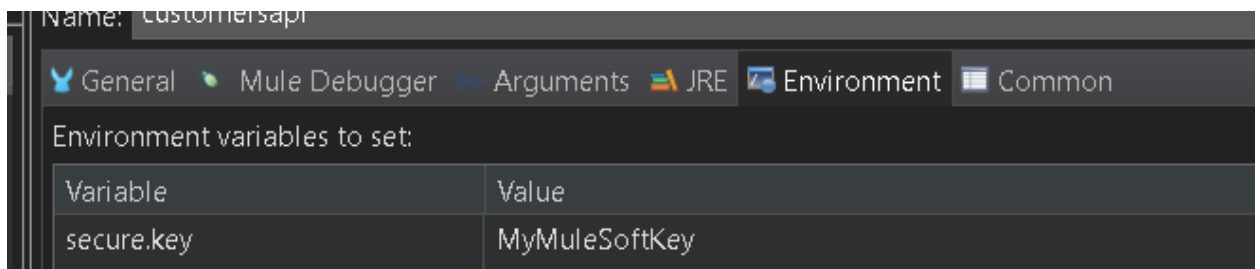
```
1 db.user=[hn5PdGLGVPw=]
2 db.password=[hn5PdGLGVPw=]
```

```
julio@Julio_Mendez MINGW64 ~/Downloads
$ java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool \
string \
encrypt \
Blowfish \
CBC \
MyMuleSoftKey \
"mule" \
hn5PdGLGVPw=
```

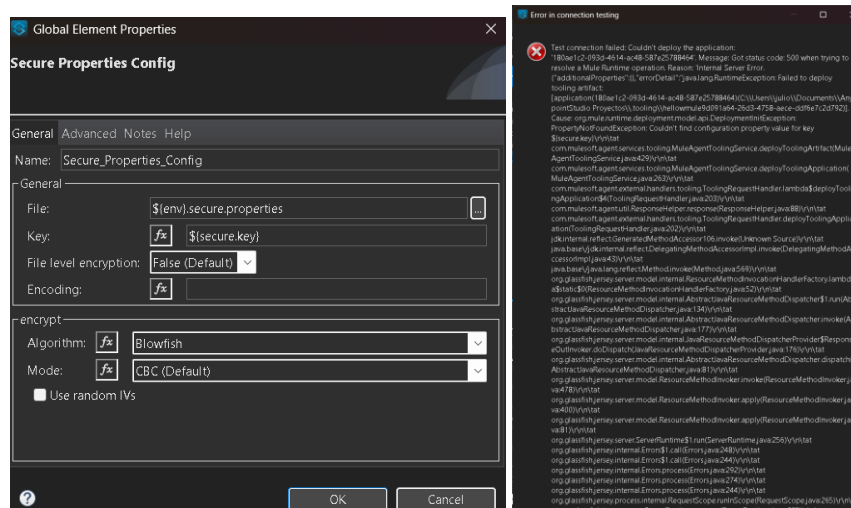
4. Se crea el archivo global.xml donde estará env para poder cambiar entre local o dev de acuerdo a su valor



En Run configurations crearemos la variable secure.key con el valor MyMuleSoftKey que servira como llave para descriptar las variables user y password



Y se coloca de igual manera en global.xml, aunque yo tuve que quitarlo debido a un error que no me dejaba hacer la conexión con la base de datos que no fui capaz de solucionar



5. Se crea la especificación de la API customers-api.raml

De nombre Customers API, con la ruta /api/v1/sps y con el mismo formato de la tabla customers, siendo un GET al ser solo de consulta

```
1 ##RAML 1.0
2 title: Customers API
3 version: v1
4 baseUri: /api/v1/sps
5
6 /customers:
7   get:
8     description: Returns all customers
9     responses:
10      200:
11        body:
12          application/json:
13            example: |
14              [
15                {
16
17                  "FirstName": "Maura",
18                  "LastName": "Wagstaffe",
19                  "Email": "mwagstaffe@npr.org",
20                  "Company": "Kazu"
21                }
22              ]
```

6. Lo desplegaremos en Cloudhub, antes debemos de iniciar sesión en la plataforma para poder desplegarlo, y ya solo le indicaremos que suba nuestro proyecto

Applying changes will create a new configuration for your application Apply Changes

Application File

Deployment Target Ingress Properties Logging

Runtime version

Release Channel new Edge ▼

① This channel releases a new minor version every four months. It has the latest features and shorter support windows. [Learn more](#)

Runtime Version 4.9.6.See ▼

② Mule Runtime uses semantic versioning. Each version ends with a build number. Versions in the Edge channel are indicated with "ee" at the end. [Learn more](#)

Java Version new

Using Java 17 may require a different application resource profile. [Learn more](#)

☐ Java 8 ☒ Java 17

Replicas

Replica Count 1 ▼

Replica size 0.1 vCores ▼

Y en la página de Anypoint Runtime Manager podemos ver que ya está corriendo

customersapi Stop ▼ 🔔

Application status: Running

Configuration: 611c2a

Last updated: 2025-06-25 1:15:52AM

Replicas: 1/1 started

Public Endpoint: <https://customersapi-cxo7dt.5sc6y6-1.usa-e2.cloudhub.io>

Target name: Cloudhub-US-East-2

Target type: Shared Space

Configuration 611c2a → Applying changes will create a new configuration for your application Apply Changes

Application File

customersapi.jar Choose file

Version 1.0.0

Deployment Target Ingress Properties Monitoring

Con la siguiente URL <https://customersapi-cxo7dt.5sc6y6-1.usa-e2.cloudhub.io/api/v1/sps/customers> verificamos que está funcionando, trayendo en tipo JSON los datos de la tabla customers de la base de datos Training

```
[{"FirstName": "Maura", "LastName": "Wagstaffe", "Email": "mwagstaffe@npr.org", "Company": "Kazu"}, {"FirstName": "Myrwyn", "LastName": "Eliet", "Email": "meliet1@buzzfeed.com", "Company": "Skibox"}, {"FirstName": "Myrwyn", "LastName": "Eliet", "Email": "meliet1@buzzfeed.com", "Company": "Skibox"}, {"FirstName": "Jenn", "LastName": "White", "Email": "jwhite@wamu.org", "Company": "NPR"}, {"FirstName": "Kelly", "LastName": "McEvers", "Email": "kmcEvers@npr.org", "Company": "NPR"}, {"FirstName": "Audie", "LastName": "Cornish", "Email": "audie@npr.org", "Company": "NPR"}, {"FirstName": "Ari", "LastName": "Shapiro", "Email": "ari@npr.org", "Company": "NPR"}]
```


7. Api Manager

En la pagina de anypoint en la parte de Api Manager le damos a la opción
Add New API

Y añadimos el archivo raml

Con eso tendríamos creada la API, en base a la que hicimos en el proyecto

Once the API is created it will be published in Exchange in stable state.

Name: Customer-API

Asset types: REST API

Method: Upload a RAML

File upload: customers-api.raml

Main file: customers-api.raml

Advanced >

Instance Administration / Add API

Runtime API Downstream Upstream Review

Downstream

Configure the API instance settings related to inbound traffic.

* Required field

Client provider: Anypoint

Instance label (Optional):

Recommended if you have multiple managed instances of the same API

Advanced options >

Review

Review your selections before saving and deploying your API instance. You can also save the configuration and deploy it later.

Runtime

Runtime type: Mule Gateway

Proxy type: Basic Endpoint

API

API name: Customer-API

API version: v1

Asset version: 1.0.0

Downstream

Scheme: HTTP

API instance label: customers-api-dev

Upstream

Status	Name	Runtime	Label	Version	Instance	Error Rate	Total Requests	Client Applications	Creation Date
Unregistered	Customer-API	Mule 4	customers-api-dev	v1	20410740	No data	No data	0	06-25-2025