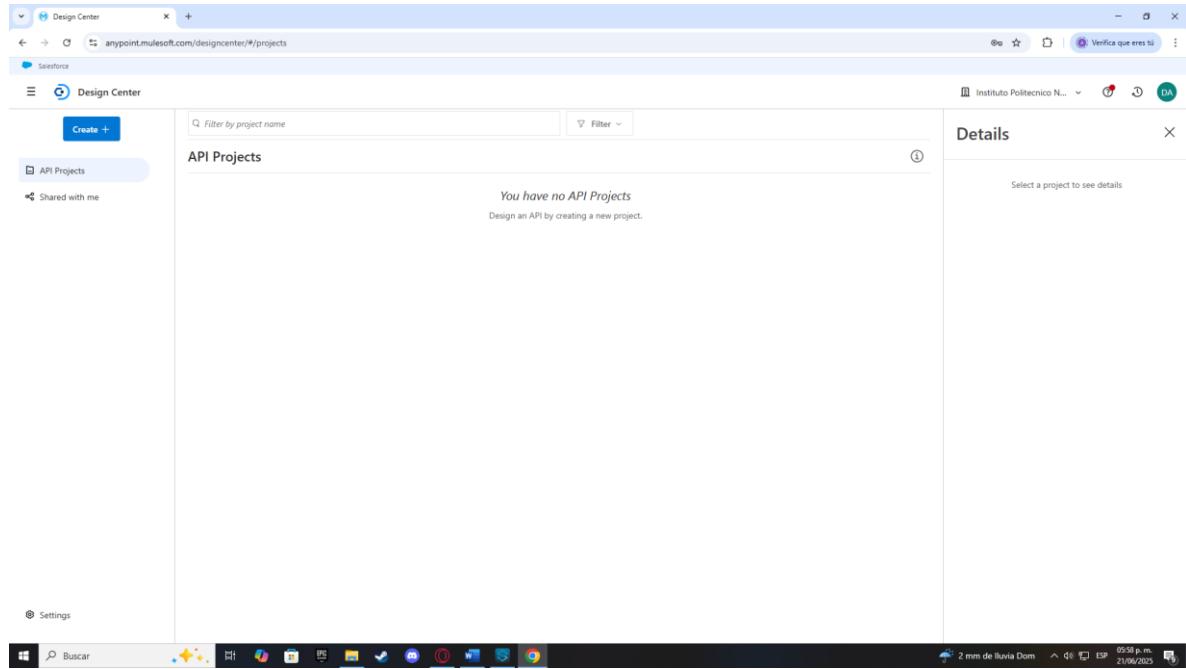
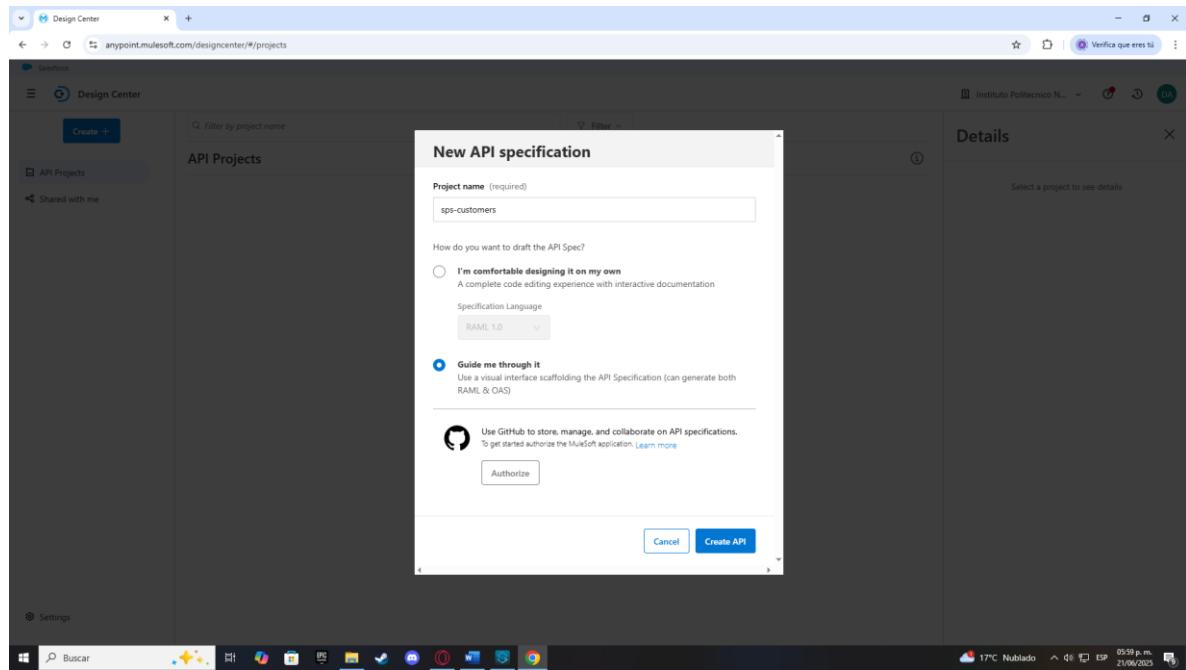


Ingreso al Portal Design Center para definir el diseño de la API a realizar, le doy clic en create y luego en NEW API specification



Le damos un nombre, en mi caso sps-customers y la opción de guía para ir armando la API mediante un formulario y le damos a create API



Y nos lleva a la siguiente pantalla, ya creado el proyecto para diseñar

The screenshot shows the Anypoint API Designer interface. On the left, there's a sidebar with sections for SECURITY SCHEMES, RESOURCES, DATA TYPES, and GROUPS. The main area is titled 'API Summary' for the 'sps-customers' API. It includes fields for Title ('New API'), Version, Protocols ('Select...'), Media type ('Select...'), Base URI (''), Description (with a rich text editor), Secured By ('Select...'), and Documentation ('Add Documentation'). On the right, there are buttons for 'Edit RAML' and 'Download'. The status bar at the bottom shows the RAML version as 'RAML 1.0 title: New API'.

Ahora en la entrevista nos piden una función get en la ruta o en el endpoint /api/v1/sps/customers el cual debe retornar todos los clientes guardados de la base de datos. Por lo que llenare el formulario de la siguiente manera:

Title : spi-customers,

Version: 1.0.0, la uri solicitada indica que estamos en la versión 1 del desarrollo de la API

Protocolos: Http

Media Type: Seleccione application/json por si en algun momento se quiere realizar operaciones del tipo create con post, los parametros se enviaran en formato json, ademas el retorno de los registros de los clientes se hara con json.

Base URI : spi-solutions/api, spi-solutions como dominio y la ruta /api para definir que todos los endpoints que le preceden corresponden a la API que estamos desarrollando

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

- Title:** spi-customers
- Version:** 1.0.0
- Protocols:** HTTP, HTTPS
- Media type:** application/json
- Base URI:** spis-solutions/api/v1/spis
- Description:** API para realizar operaciones de lectura en una base de datos
- RAML:** RAML 1.0 specification is visible on the right.

Una vez definido la parte fundamental o basica de la api procederemos a crear resources o endpoints, donde casaremos los recursos como por ejemplo los clientes guardados en la base de datos. Para esto me voy a a la parte izquierda del apartado resources y doy clic en el boton de +.

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

- Resource path:** /new-resource
- Operations:** GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD
- Summary Tab:** Selected
- RAML:** RAML 1.0 specification is visible on the right, showing the addition of a new resource: /new-resource.

Ahí nos piden llenar un formulario donde definimos las características del endpoint

Resorce Path: como se solicito será /v1/spis/customers debido a que la parte de /api la definimos como dominio base. Se hara una operación Get, por que solo estamos obteniendo o leyendo registros guradados previamente en la base de datos.

Summary

Name: GET CUSTOMERS, nombre descriptivo de lo que hara el endpoint

Description : Obtenemos todos los clientes registrados en la base de datos

The screenshot shows the Anypoint API designer interface. On the left, there's a sidebar with sections like API Summary, SECURITY SCHEMES, RESOURCES, DATA TYPES, and GROUPS. Under RESOURCES, a new resource named 'spis-customers' is being created with the URI path '/v1/spis/customers'. The main workspace shows the 'Summary' tab for the 'GET CUSTOMERS' operation, which has a description: 'Obtenemos todos los clientes registrados en la base de datos'. To the right, the RAML and OAS tabs are visible, showing the generated API specification. The top right corner includes buttons for Share, Publish, and a DA icon.

Nos dirigiremos al apartado Responses donde, se define que respuestas puede dar un endpoint, codigos de estado http, el cuerpo de la respuesta y demas.

Status : 200 ok, en caso de que no haya problema se retorna los registros,

Descripcion: En mi caso decidí comentar que campos son los que se deben retornar, ya que es mala practica retornar el id o la contraseña de un usuario.

Bodies: application/json, que tipo de contenido retornara, en este caso un json.

The screenshot shows the API designer interface for a project named 'sps-customers'. On the left, the sidebar includes sections for API Summary, SECURITY SCHEMES, RESOURCES, DATA TYPES, and GROUPS. The main area displays a 'Responses (1)' tab for a GET method. The response status is set to 200 - OK. The 'Description' field contains a Markdown block with the following content:

```

Regresa los clientes en formato json todos los clientes en la base de datos, con ciertos campos definidos por la lógica del negocio.
{
  "nombre": "nombre x",
  "apellido": "apellido x",
  "correo": "domingo@email.com"
}

```

The 'Bodies (1)' section shows a media type of application/json and a type of Object.

On the right, the RAML and OAS tabs are visible, along with a detailed description of the API endpoint.

Añadimos otra respuesta, por si tenemos un fallo interno en el servidor o en la API como una desconexión a la base de datos

Status: 500, en métodos http los códigos de error 5XX informan sobre un problema ocurrido en la parte del servidor.

Description: Si falla la conexión de base de datos, notificar al usuario con un error 500

The screenshot shows the same API designer interface for the 'sps-customers' project. The 'Responses (2)' tab is selected, showing two entries: a 200 status and a 500 status. The 500 status is defined with the following description:

Si falla la conexión de base de datos, notificar al usuario con un error 500

The 'Bodies (0)' section is empty.

On the right, the RAML and OAS tabs are visible, along with a detailed description of the API endpoint, including the addition of the 500 error response.

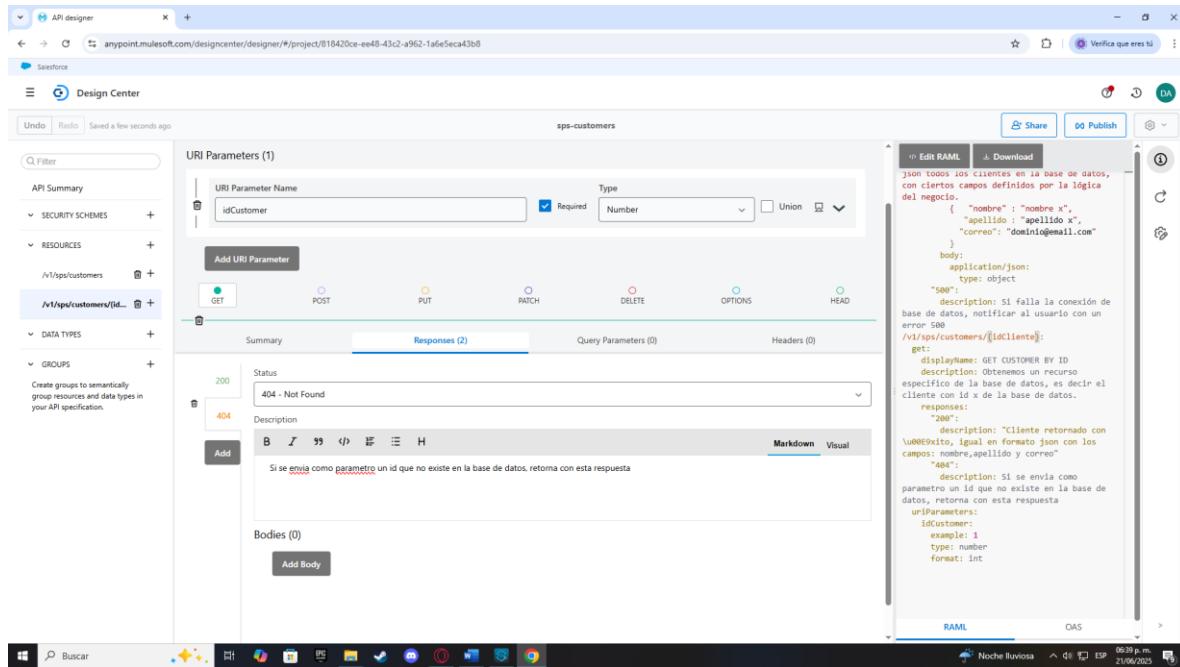
De igual forma lo hacemos para el endpoint `/v1/customers{idCliente}` donde le pasamos como path param el id del cliente

The screenshot shows the Anypoint API Designer interface. On the left, the sidebar lists resources under the path `/v1/sp/s/customers/{idCliente}`. The main panel shows the configuration for this endpoint. The "Resource path" is set to `/v1/sp/s/customers/{idCliente}`. Under "URI Parameters (1)", there is a parameter named `idCustomer` of type Number, marked as Required. Below this, the "Responses" tab is selected, showing a single response entry for status code 200. The description for this response is: "Obtenemos un recurso específico de la base de datos, es decir el cliente con id x de la base de datos." To the right, the "Edit RAML" section displays the corresponding RAML code for this endpoint.

Agregamos una respuesta si es que el id existe en la base de datos, retornará un código 200 y en formato json el nombre, apellido y el correo.

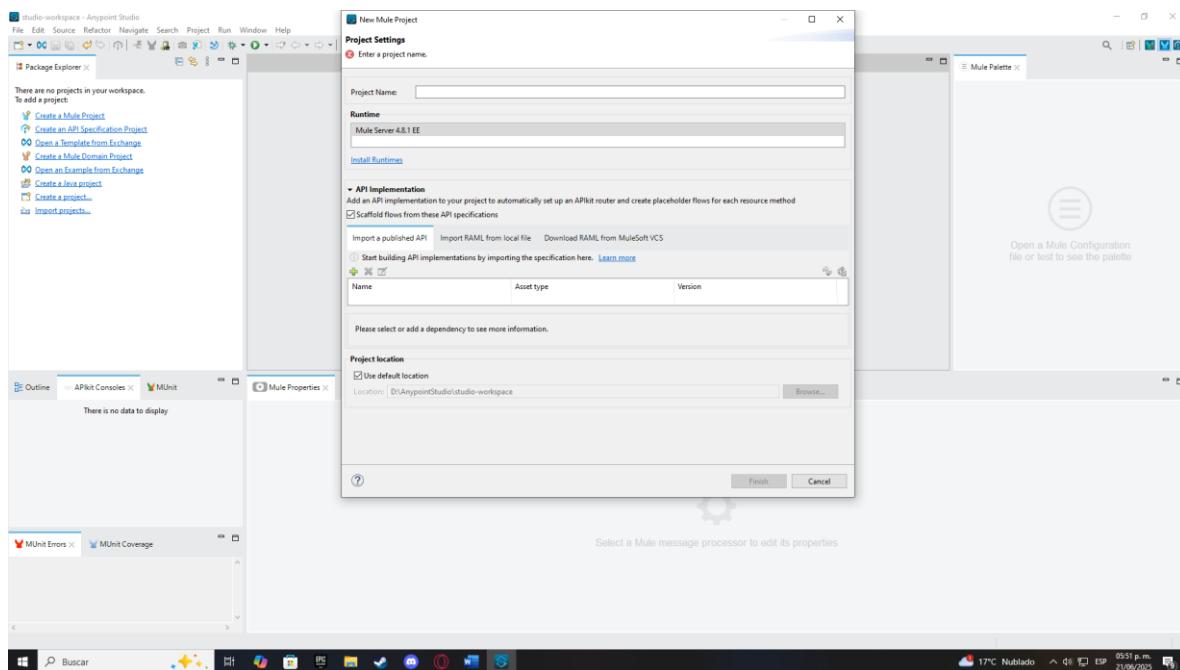
This screenshot shows the same API Designer interface after adding a response. The "Responses (1)" tab is now active, showing a new entry for status code 200 with the description: "Cliente retornado con éxito, igual en formato json con los campos: nombre, apellido y correo". The RAML code on the right side of the interface has been updated to reflect this new response.

Agregamos una nueva respuesta en dado caso de que no exista el id pasado como parametro, lanzara un codgio 404 Not Found, los codigos 4XX indican un error de parte del cliente

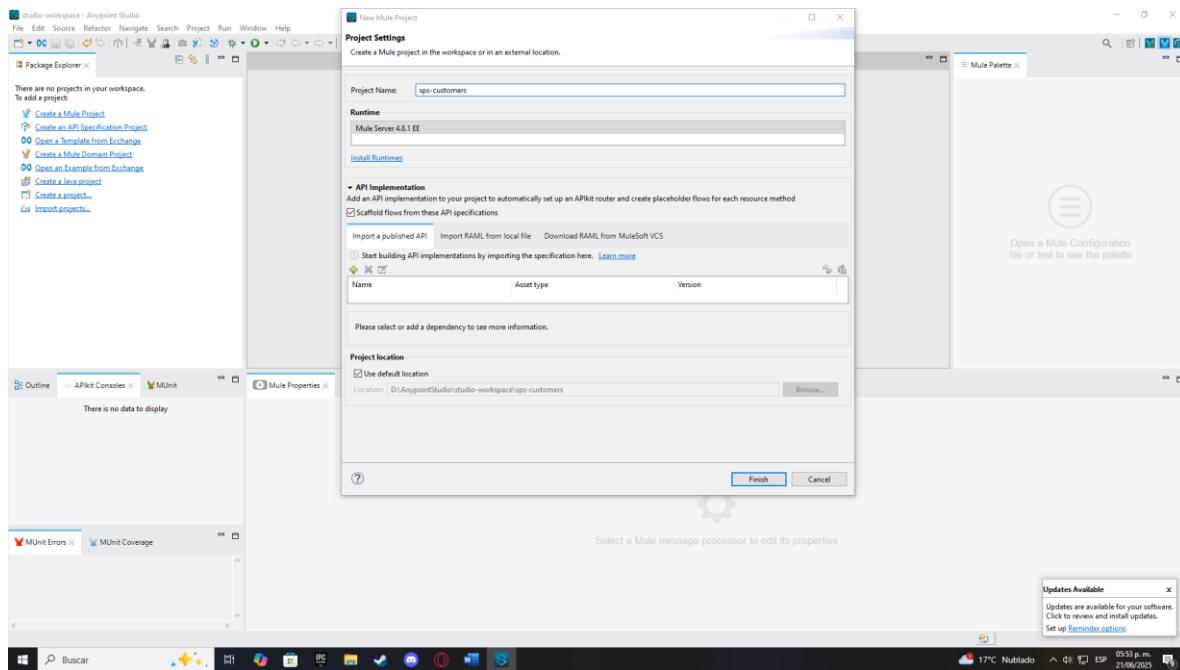


Ya definido nuestra API pasaremos a AnyPoint Studio para desarrollarla.

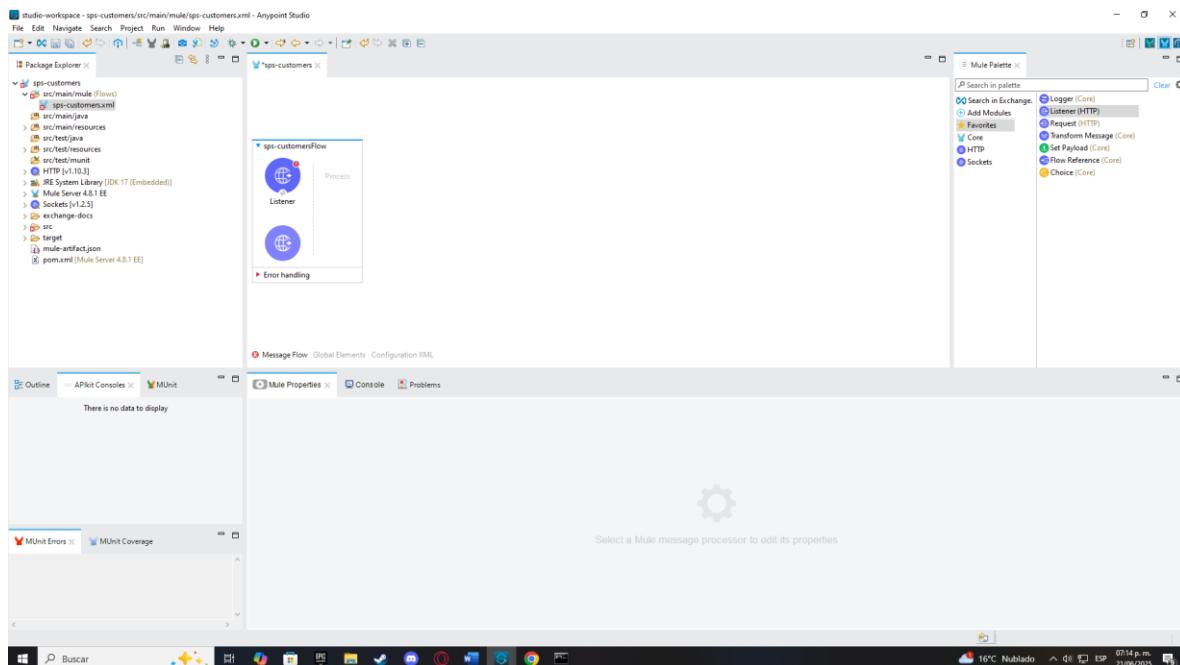
1.-Ingreso al programa Anypoint Studio y le doy clic en crear un nuevo “Mule Project”



2.-En mi caso se llamará el proyecto sps-customers.

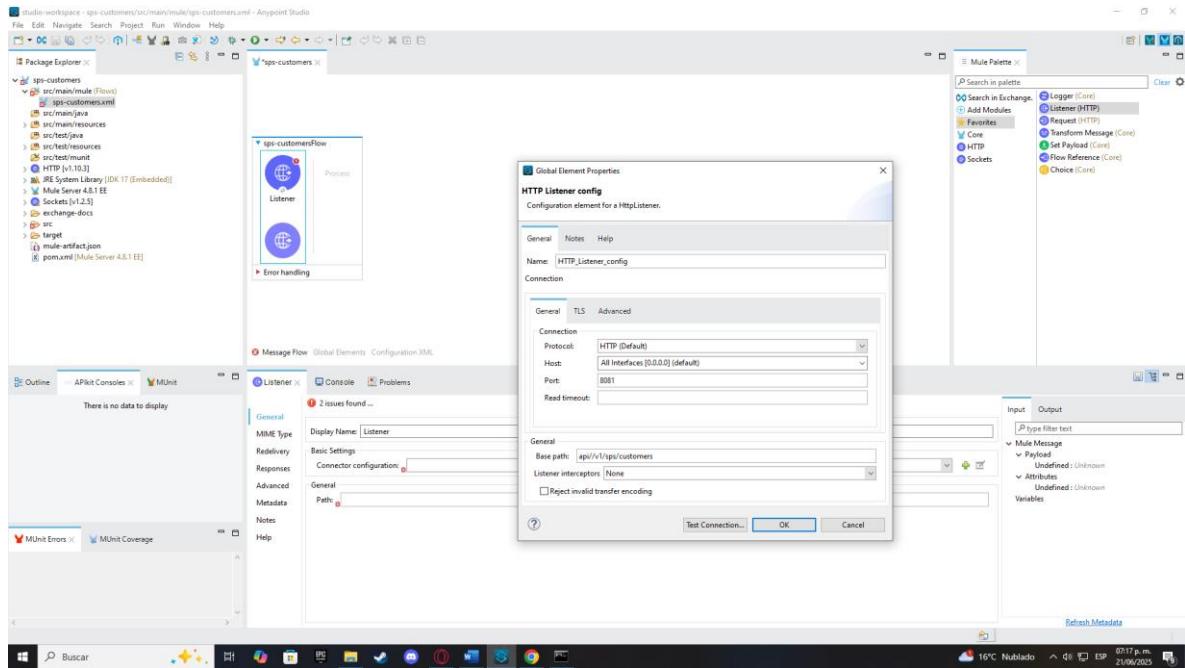


Agregaremos un HTTP Listener para que pueda recibir peticiones realizadas por un cliente.

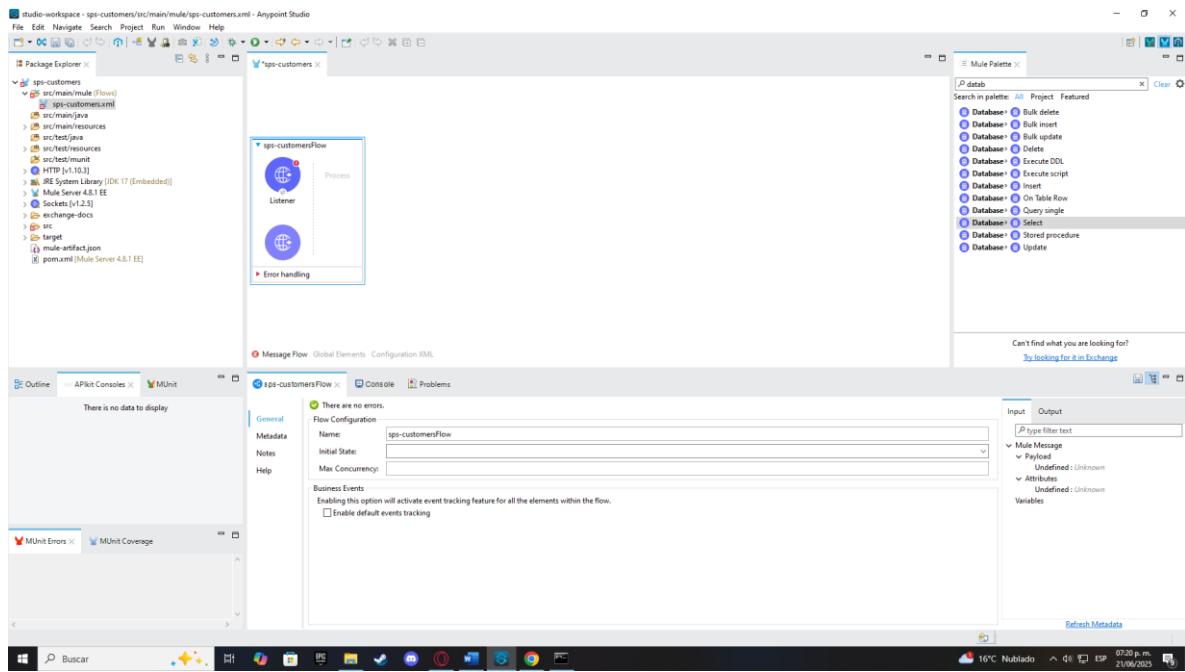


Le damos doble clic al elemento y luego al botón + de color verde.

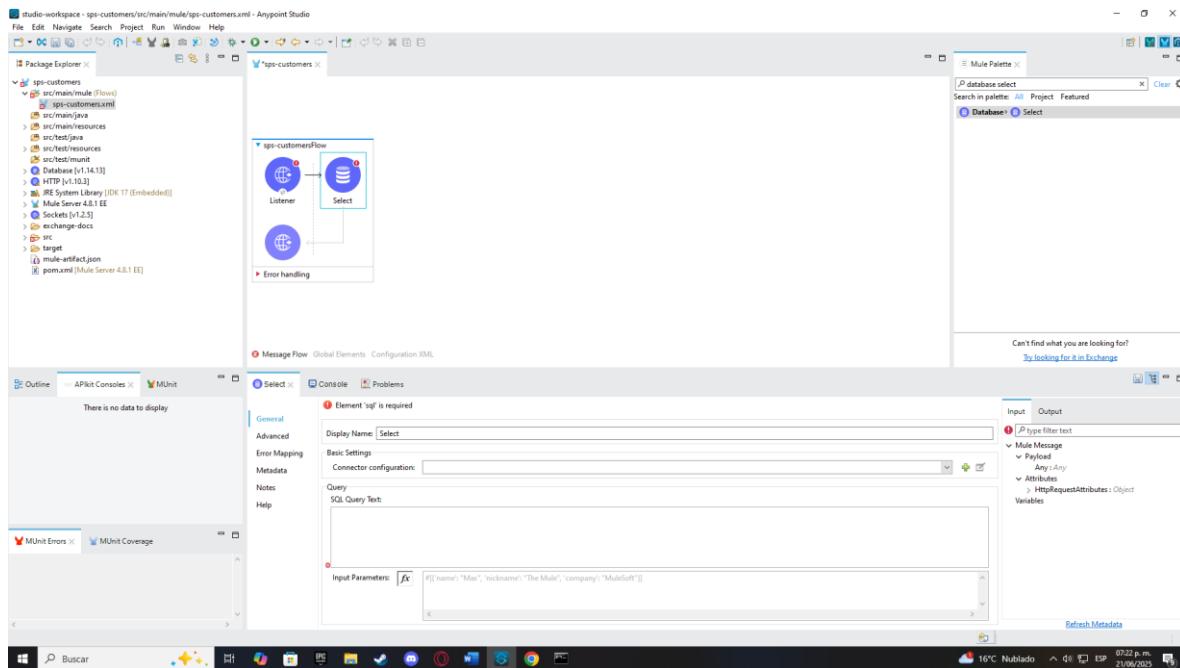
El campo que nos interesa es el base path, el cual es el endpoint o la ruta, en nuestro caso: api/v1/sps/customers



Ahora debemos definir el “flujo” del listener es decir, las operaciones que va a realizar, en este caso ir a la base de datos e intentar hacer una operación select de los customers. Para esto vamos a la paleta y buscamos el componente Database Select



Una vez que lo encontramos lo seleccionamos y lo enviamos al listener en la parte del procces o flujo



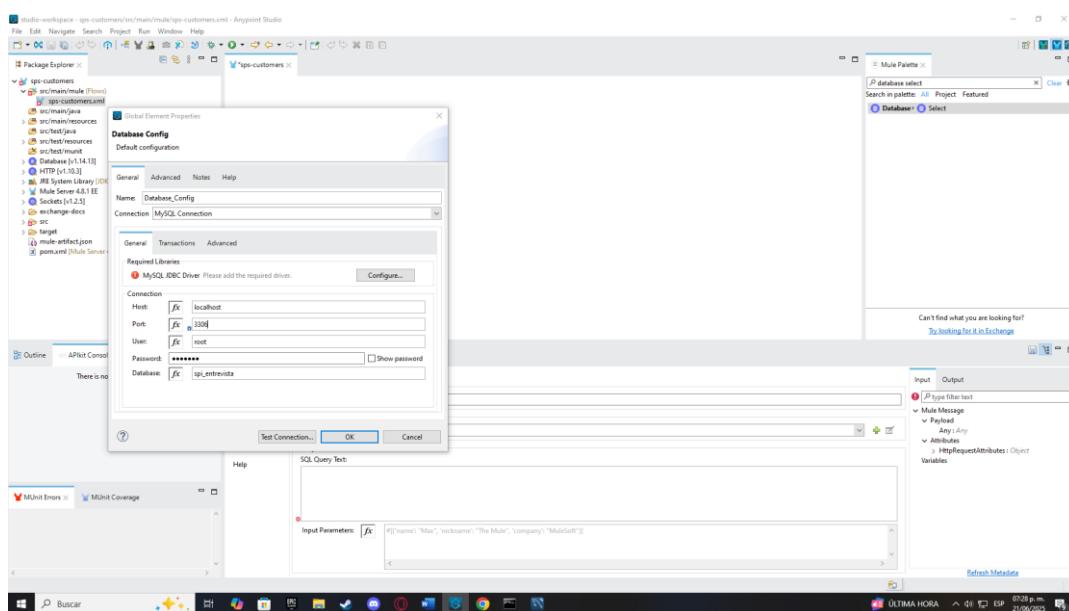
Nos metemos a la configuración y en tipo de conexión ponemos MySQL Connection nos pedirá los datos (que por el momento setearemos) como

Host : localhost

Port: 3306

User : root

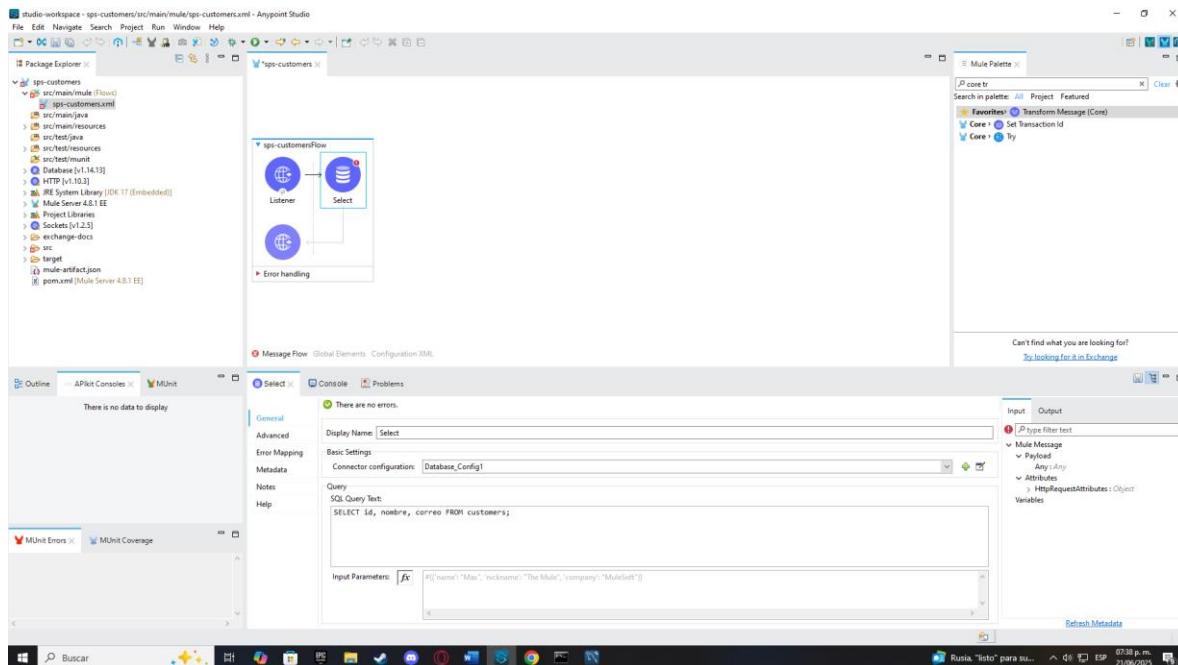
Password : contraseñaLocal



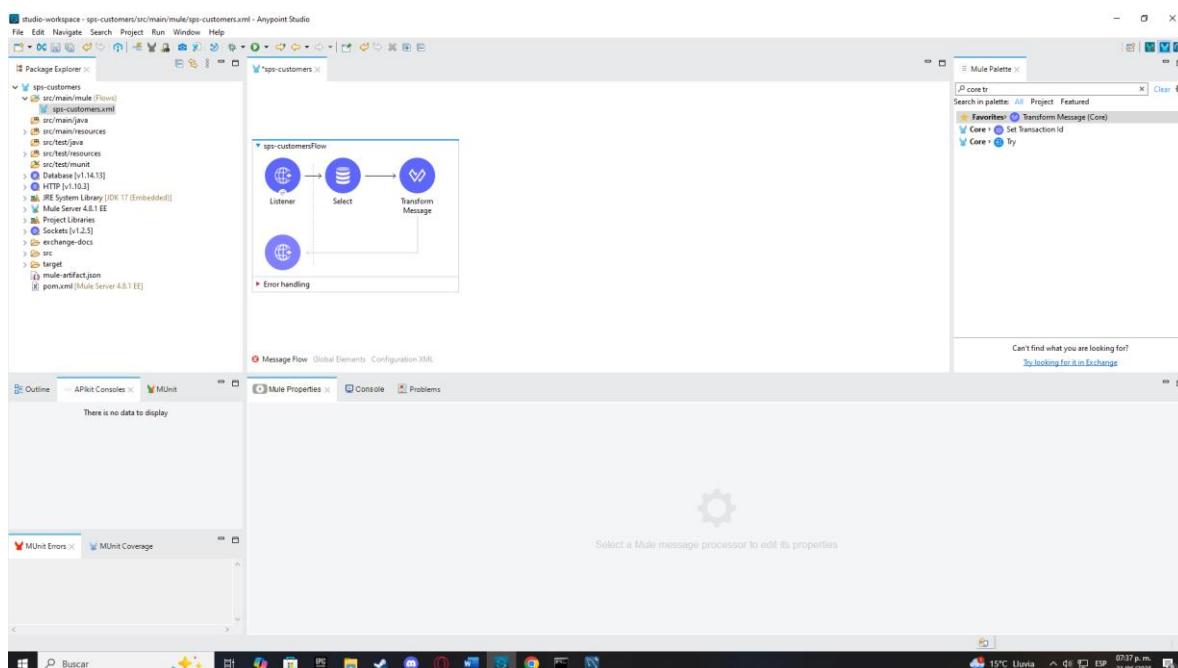
Una vez configurada la conexión en el campo Query, colocamos la query o la operación a realizar, en este caso un select que traiga los campos id,nombre,correo de la tabla customers.

Query : SELECT nombre,apellido,correo from costumer

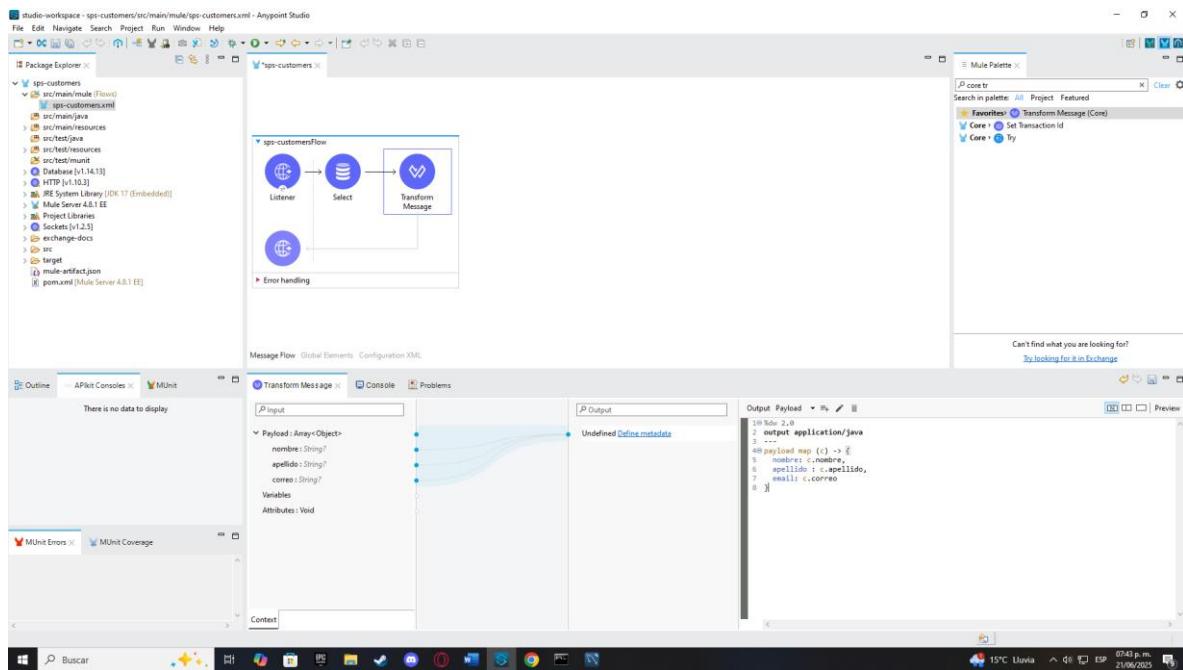
El campo de Input Parameters lo dejamos vacio por que no lo necesitamos.



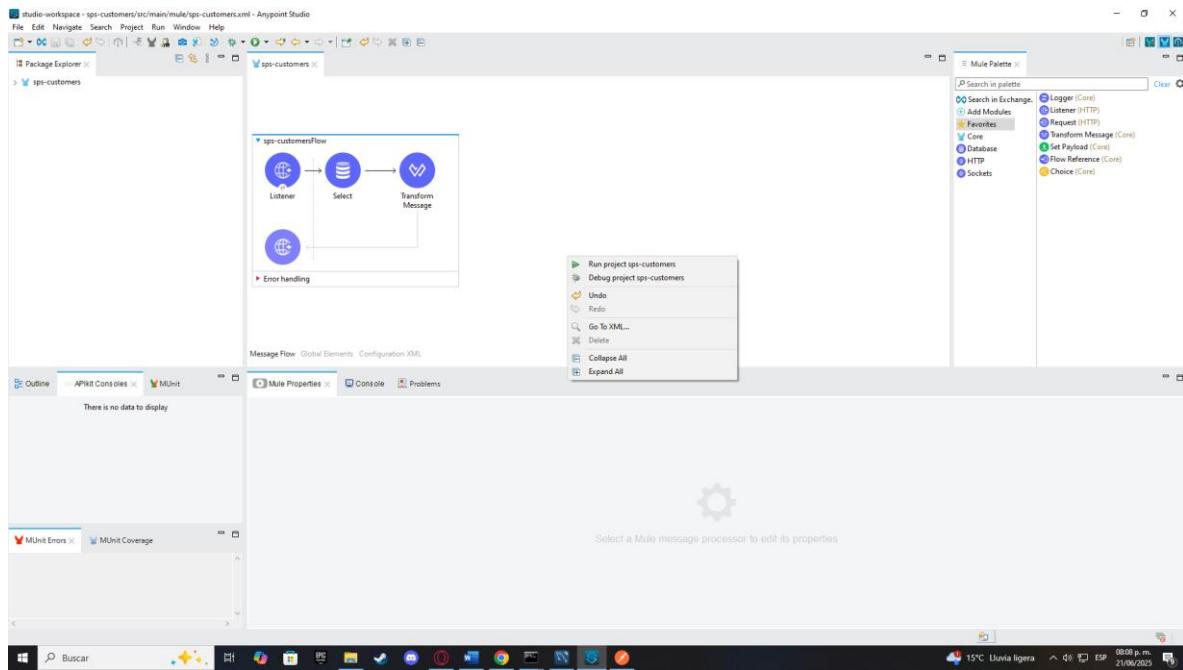
Ahora necesitamos un elemento del tipo Transform Message, para retornar como json la respuesta. Para esto lo buscamos en la paleta y lo arrastramos después de realizar el select.



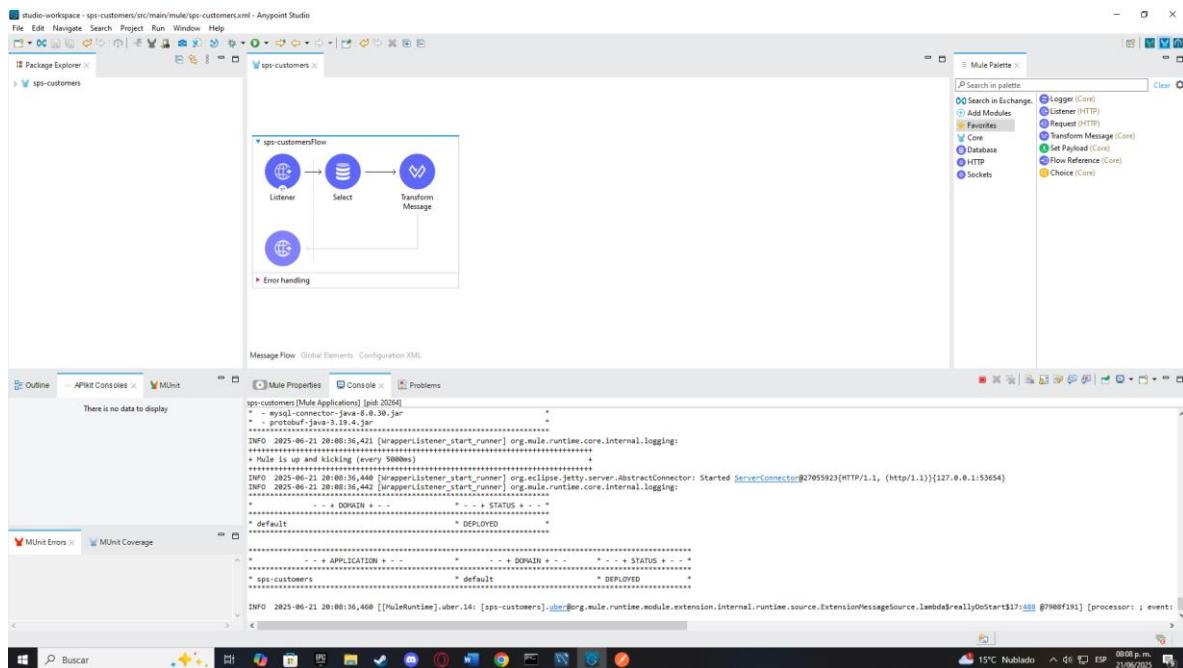
Al darle clic, nos aparece el recuador de Output



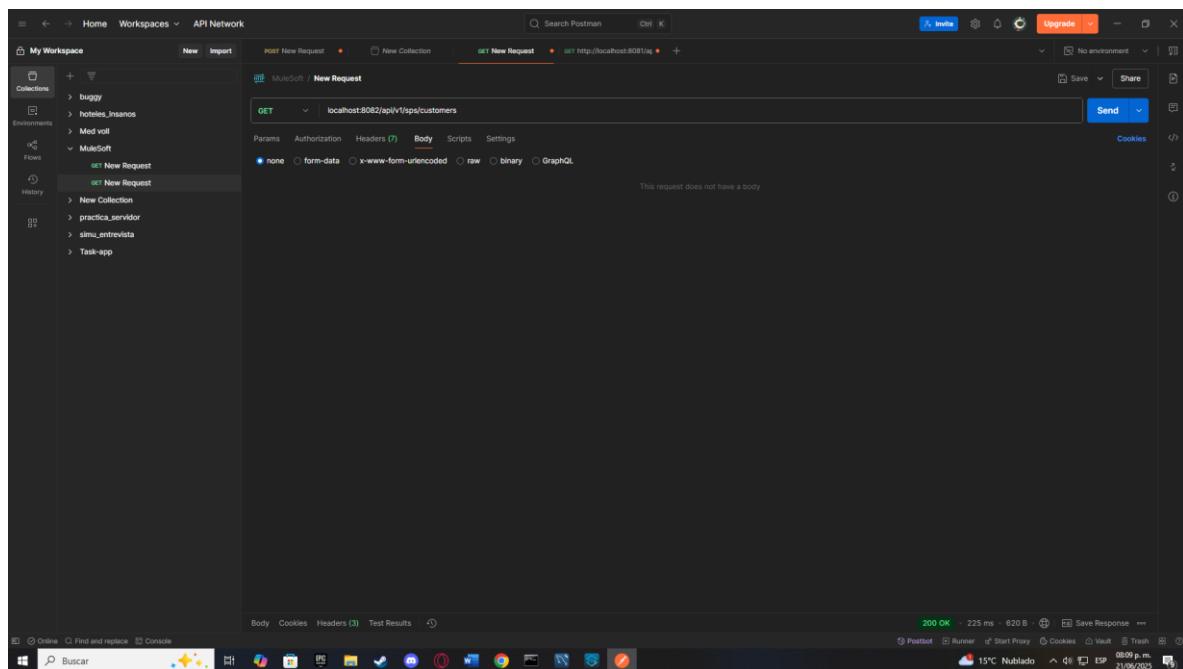
Ahora probaremos el endpoint para esto primero corremos el proyecto



Nos aparece un status de deployed correcto.



Ahora abrimos nuestro cliente de API, en mi caso postman y probamos el endpoint con el verbo get.



Y nos retorna en forma de json los clientes que existen en la base de datos.

The screenshot shows the Postman interface with a successful API call. The URL is `localhost:8082/api/v1/sps/customers`. The response body is a JSON array containing six customer records:

```

[{"id": 1, "name": "Juan", "apellido": "Perez", "email": "juan.perez@example.com"}, {"id": 2, "name": "Maria", "apellido": "Gomez", "email": "maria.gomez@example.com"}, {"id": 3, "name": "Carlos", "apellido": "Rodriguez", "email": "carlos.rodriguez@example.com"}, {"id": 4, "name": "Ana", "apellido": "Lopez", "email": "ana.lopez@example.com"}, {"id": 5, "name": "Luis", "apellido": "Martinez", "email": "luis.martinez@example.com"}]

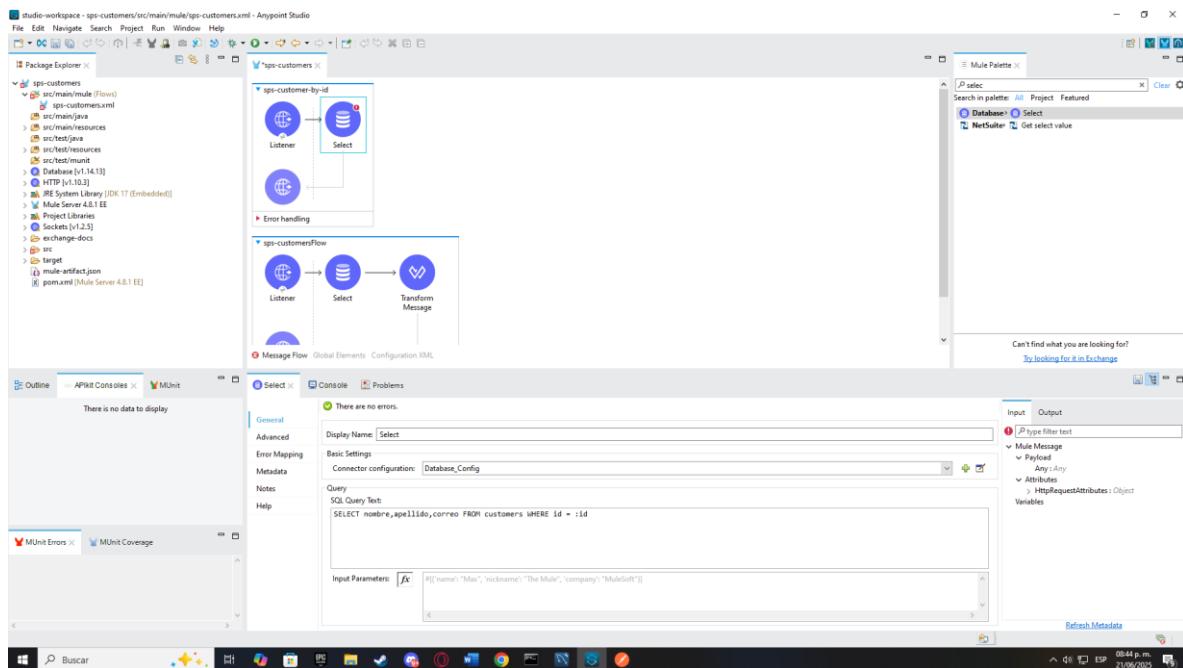
```

Ahora realizare la búsqueda por cliente a través de su id mediante un path param, para eso agregamos un nuevo listener copiando la ruta establecida, pero con la diferencia de que le agregaremos el parámetro `{idCustomer}` por lo que quedaría así:
`api/v1/sps/customers/{idCustomer}`. Reutilizamos la configuración del listener, lo único que cambia es la ruta.

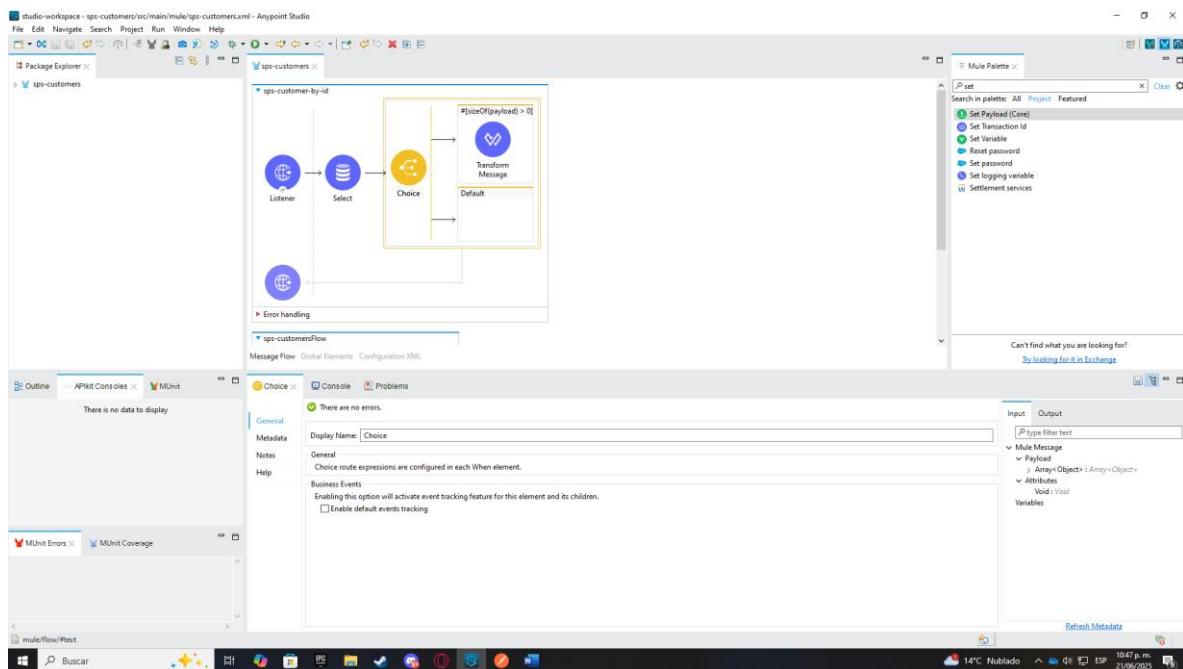
The screenshot shows the Anypoint Studio interface with the project `sps-customers` open. In the center, the `sps-customerFlow` is displayed with three components: Listener, Select, and Transform Message. On the right, the `Mule Palette` shows various components like Logger, Add Modules, Request (HTTP), and Transform Message (Core). Below the palette, the `Listener` configuration is shown with the path `/api/v1/sps/customers/{idCustomer}`.

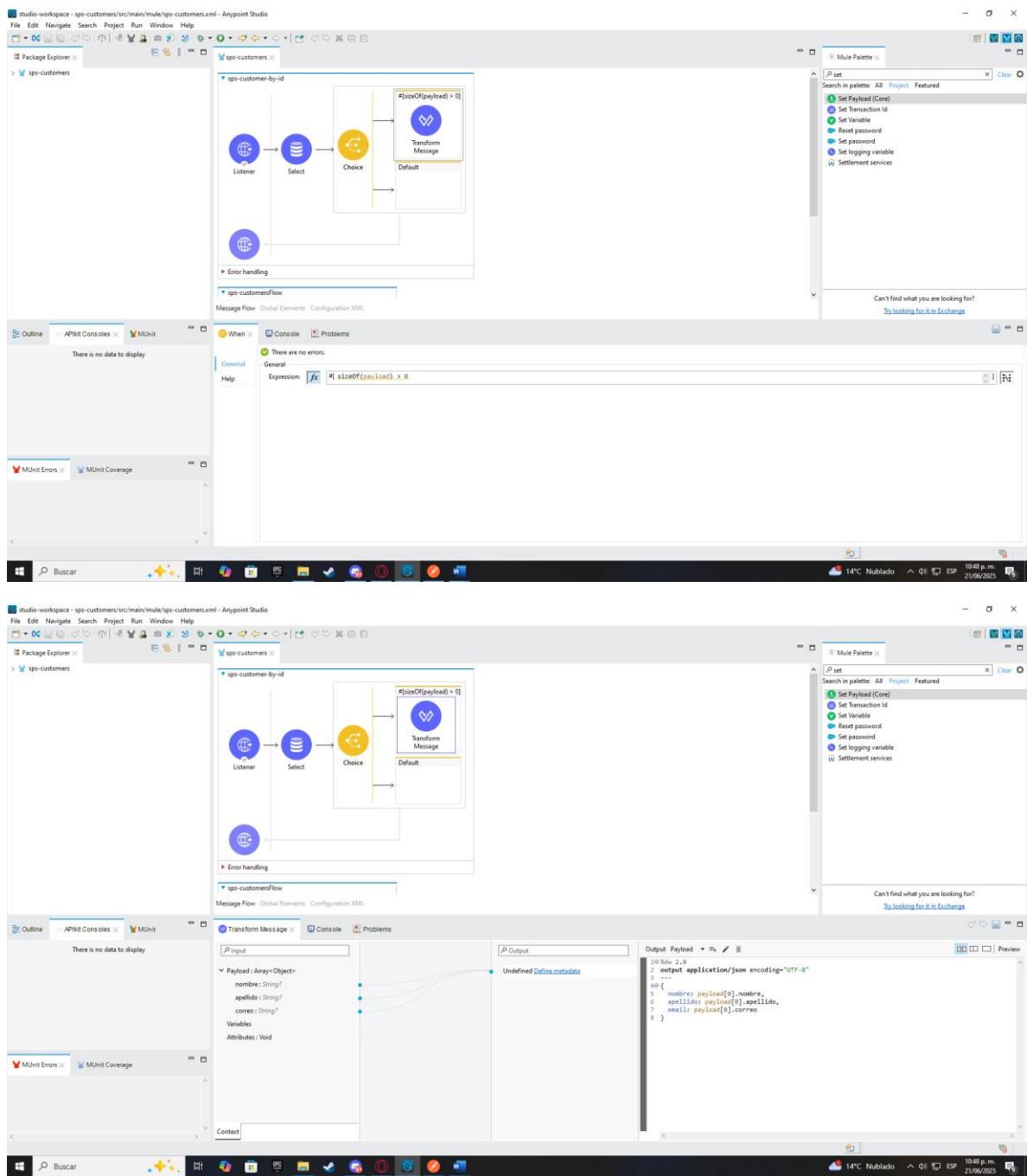
De igual forma arrastramos un elemento select y compartimos la configuración, sin embargo le cambiaremos la query por la siguiente:

SELECT nombre,apellido,correo FROM customers WHERE id = :id

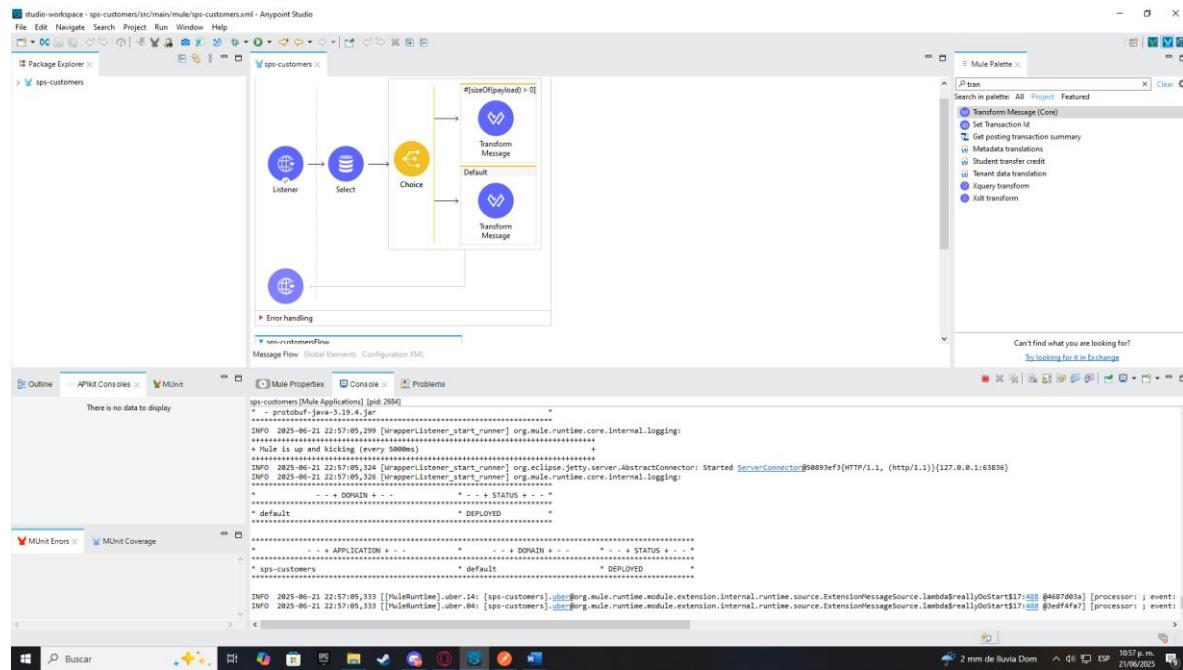


Ahora elegimos un componente llamado choice, en donde si se retorna un registro en la base de datos pasara por un transform message el cual retornara la informacion del customer.





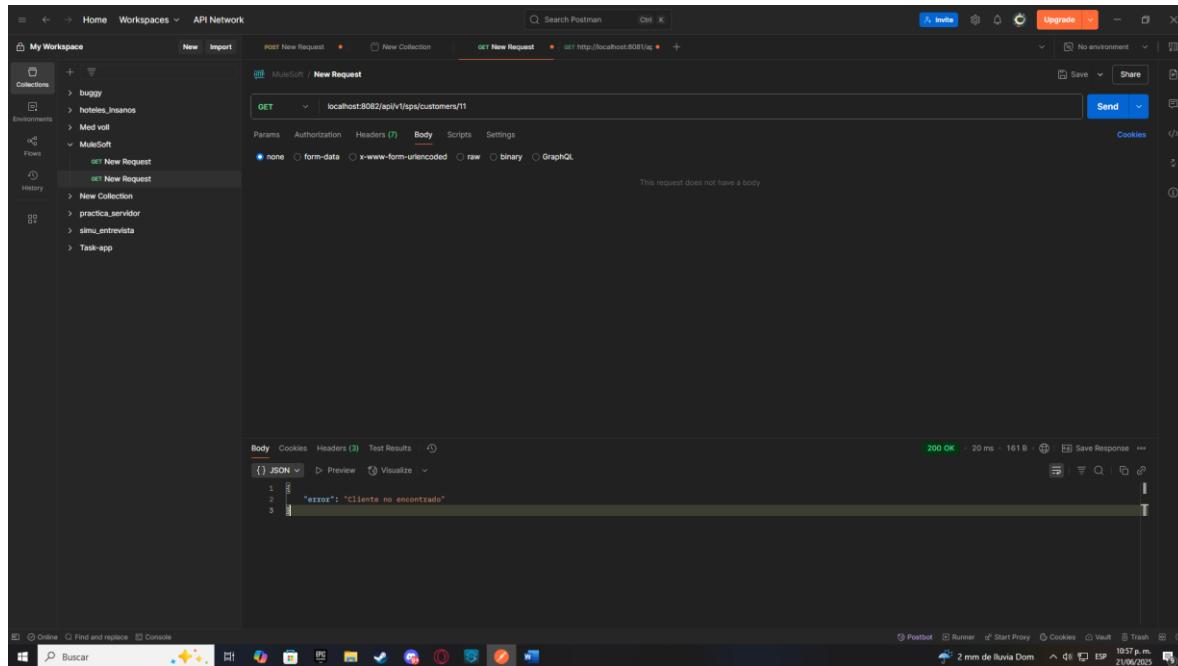
Y para el default, retornaremos un mensaje “Cliente no encontrado”



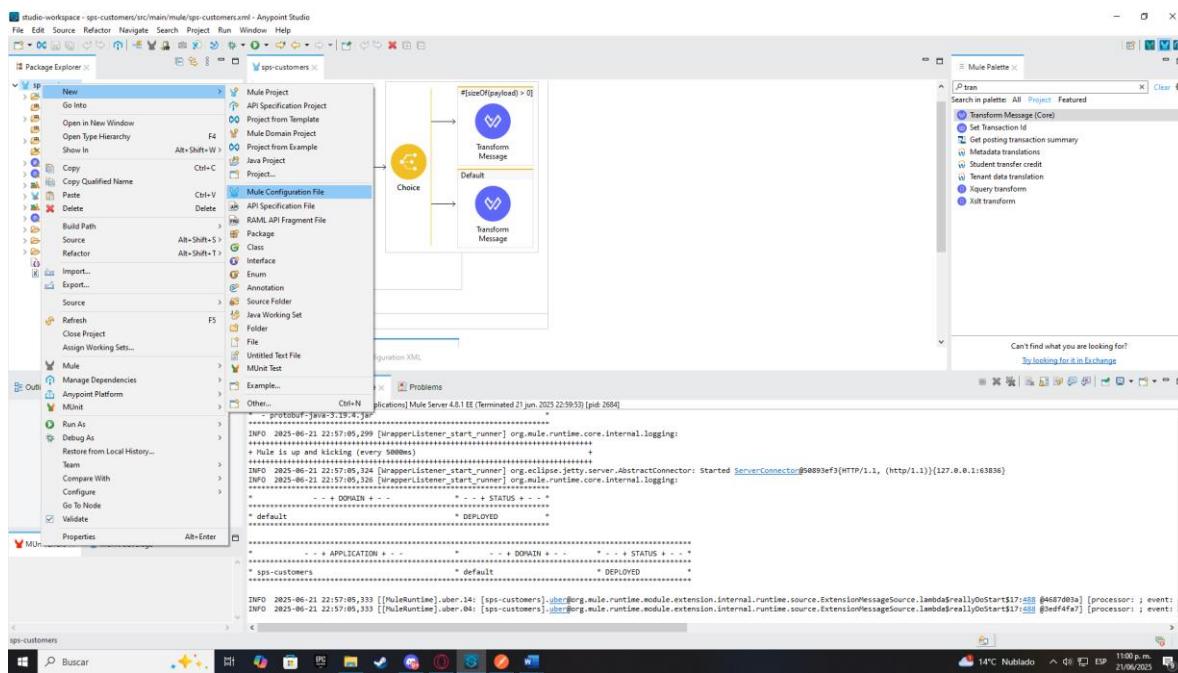
Ahora lo probaremos y efectivamente lanza la información del usuario con el id.

Body	Cookies	Headers (3)	Test Results
<pre>{ "nombre": "Juan", "apellido": "Pérez", "email": "juan.perez@example.com" }</pre>			200 OK • 224 ms • 206 B
Save Response Copy Find Visualize Raw GraphQl			

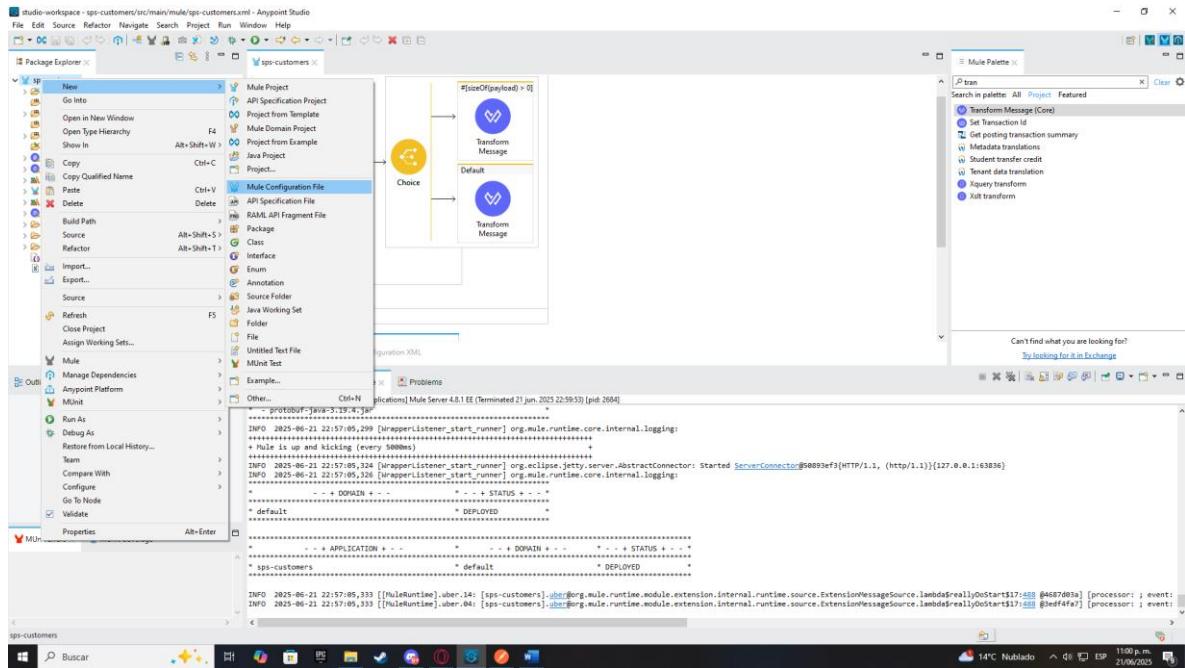
Y funciona también para un cliente que no existe en la base de datos



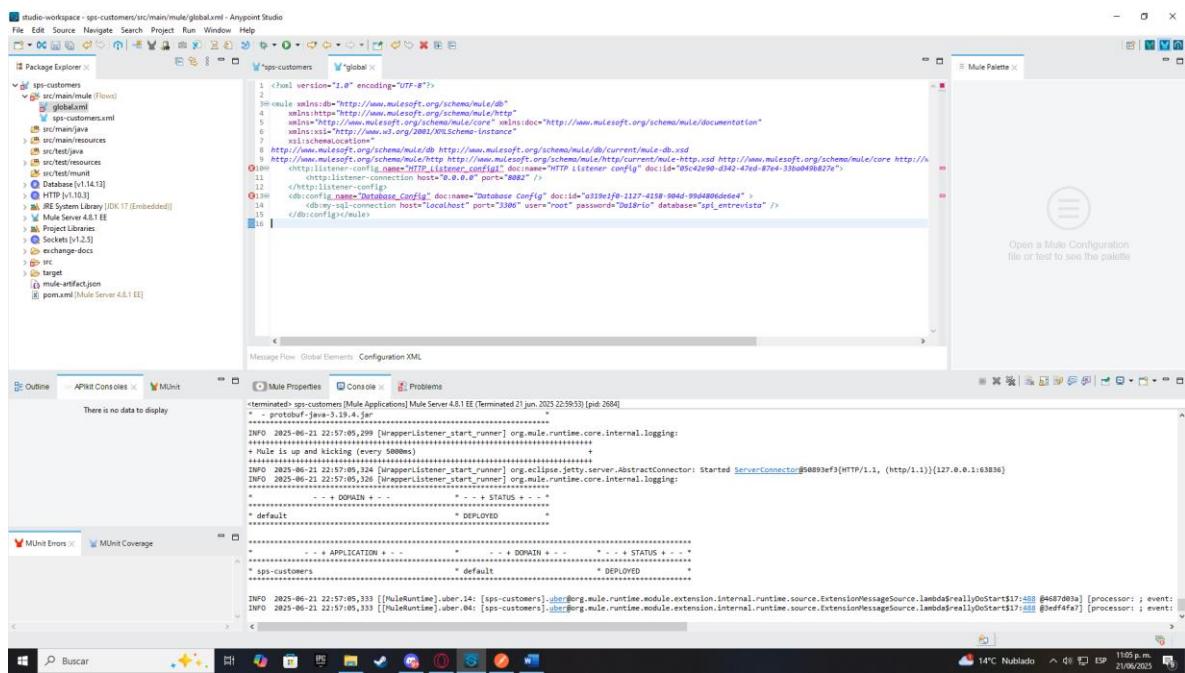
Ahora construiremos las propiedades tanto local para dev. Primero crearemos un archivo de configuración.



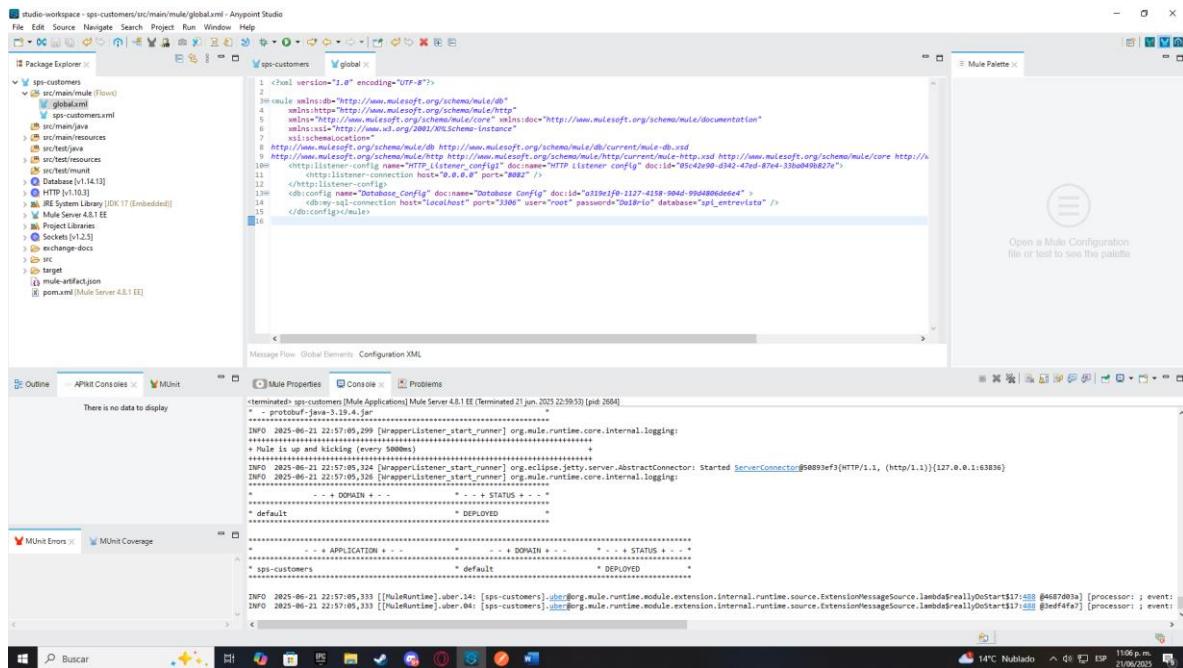
Agregaremos un archivo de configuracion



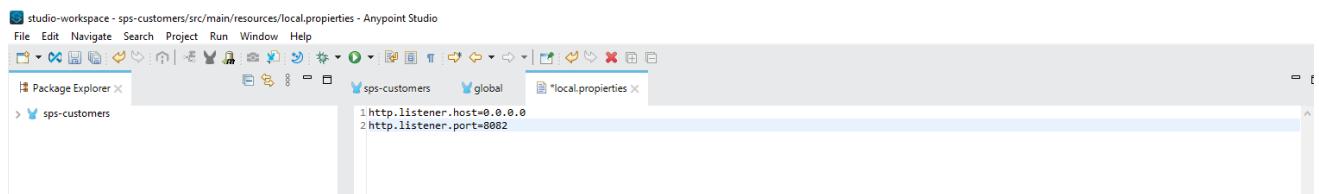
Coparemos los globales



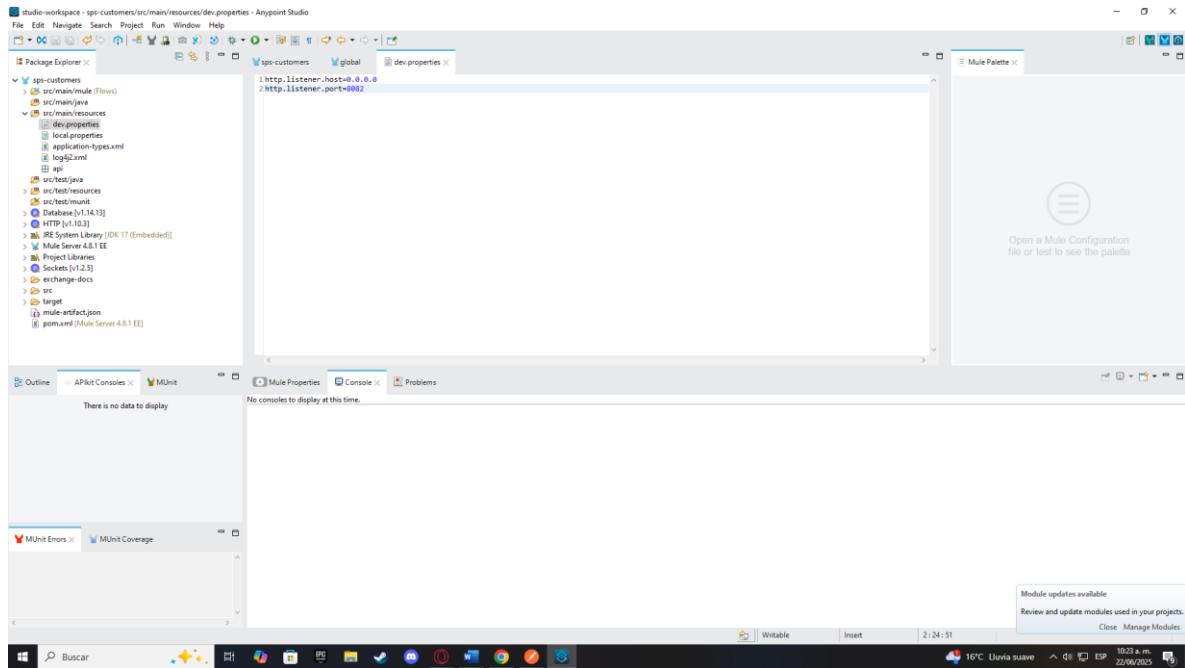
Tanto la configuración del http listener y de la conexión a la base de datos la pasaremos a la configuración global.



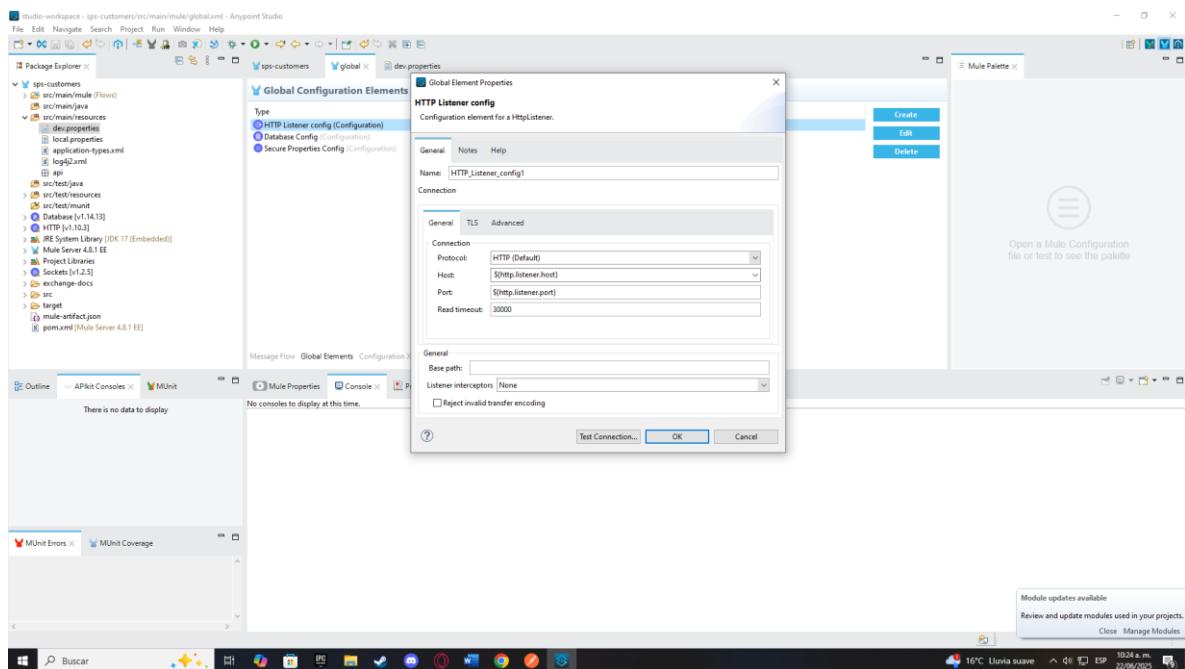
Creamos un archivo para configurar el host y puerto en el entorno local



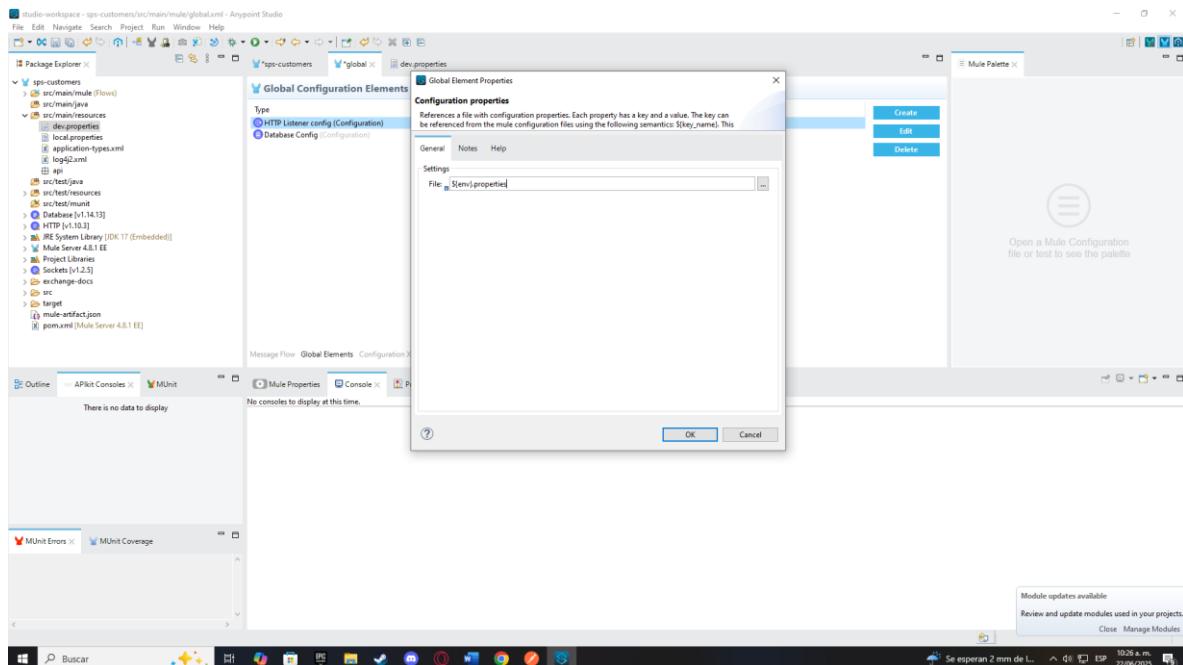
De igual forma lo hacemos para el archivo dev.properties



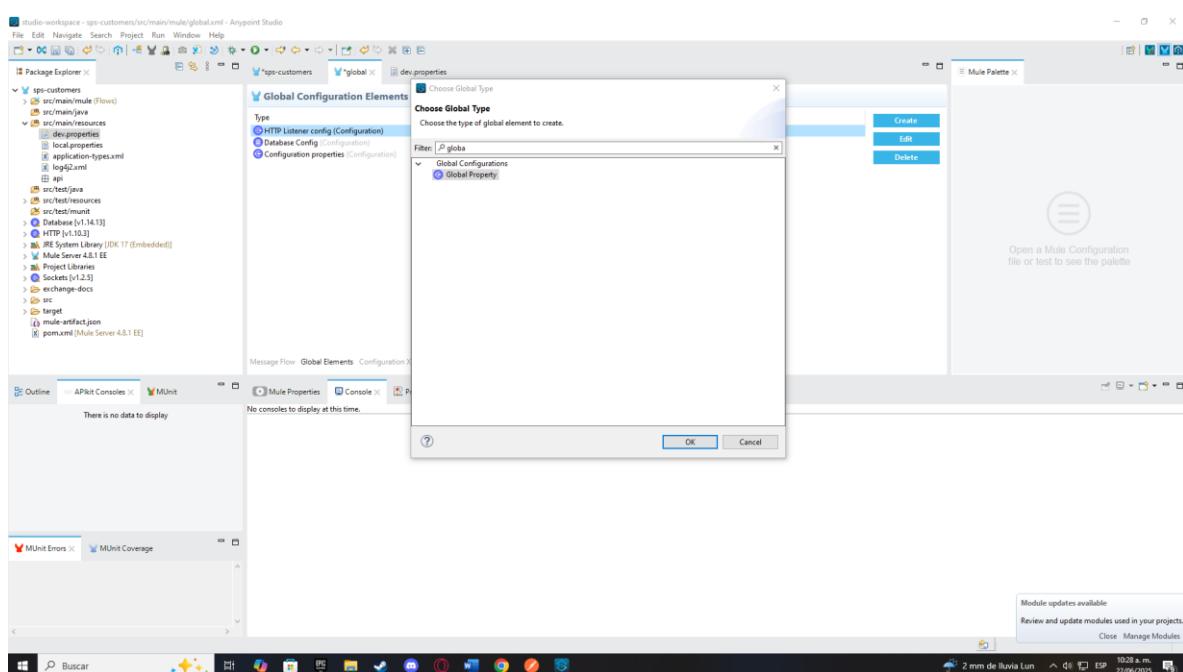
En la configuración del listener referenciamos los valores que colocamos previamente en los archivos de properties



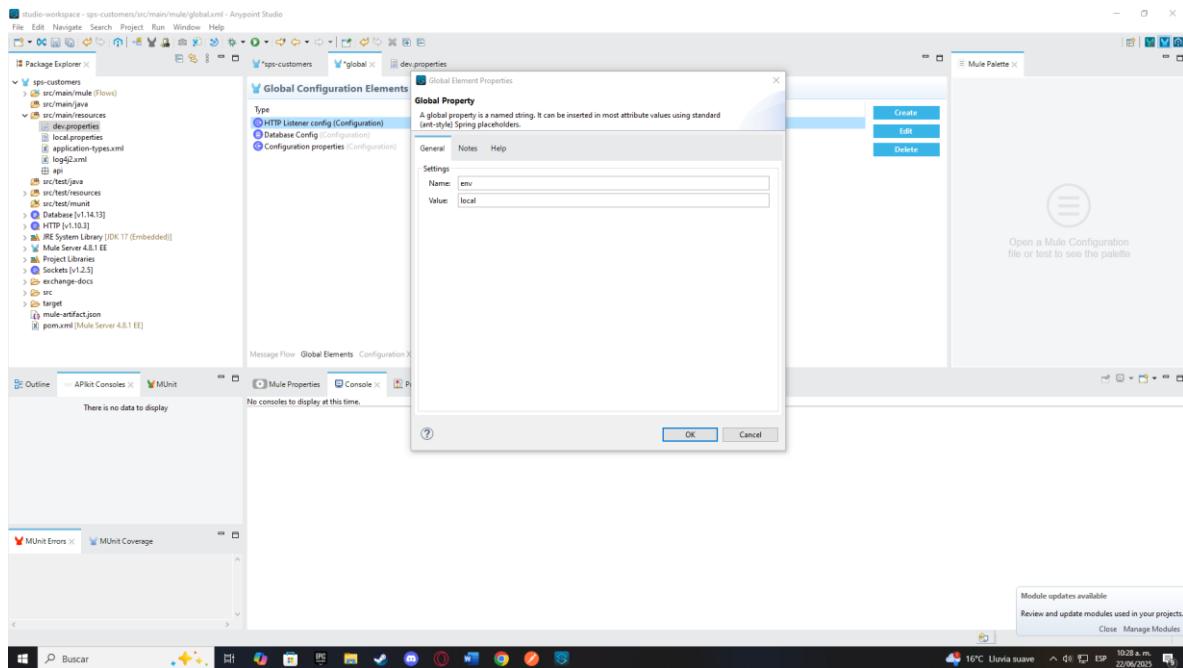
Luego seleccionamos, la configuración global de listener y le damos en create, para configura el archivo de propiedades



De igual forma creamos un global property



Y seteamos los valores env y local



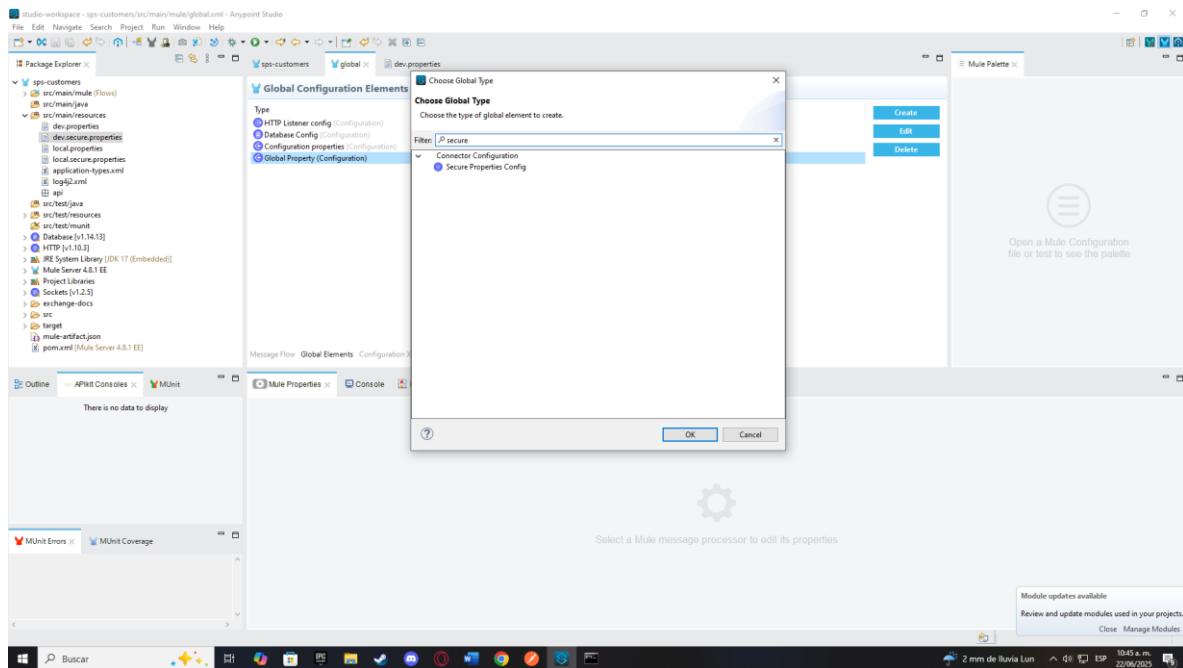
La api sigue corriendo sin ningún problema

```

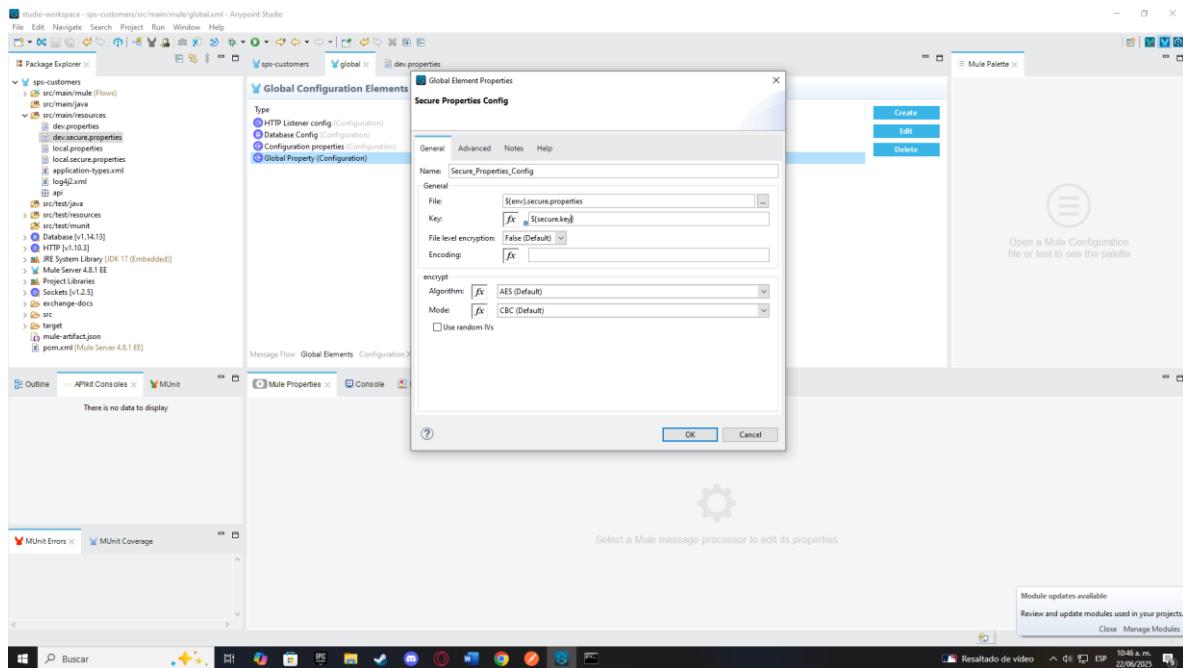
[{"id": 1, "name": "Juan", "apellido": "Perez", "email": "juan.perez@example.com"}, {"id": 2, "name": "Maria", "apellido": "Gomez", "email": "maria.gomez@example.com"}, {"id": 3, "name": "Carlos", "apellido": null, "email": null}
]

```

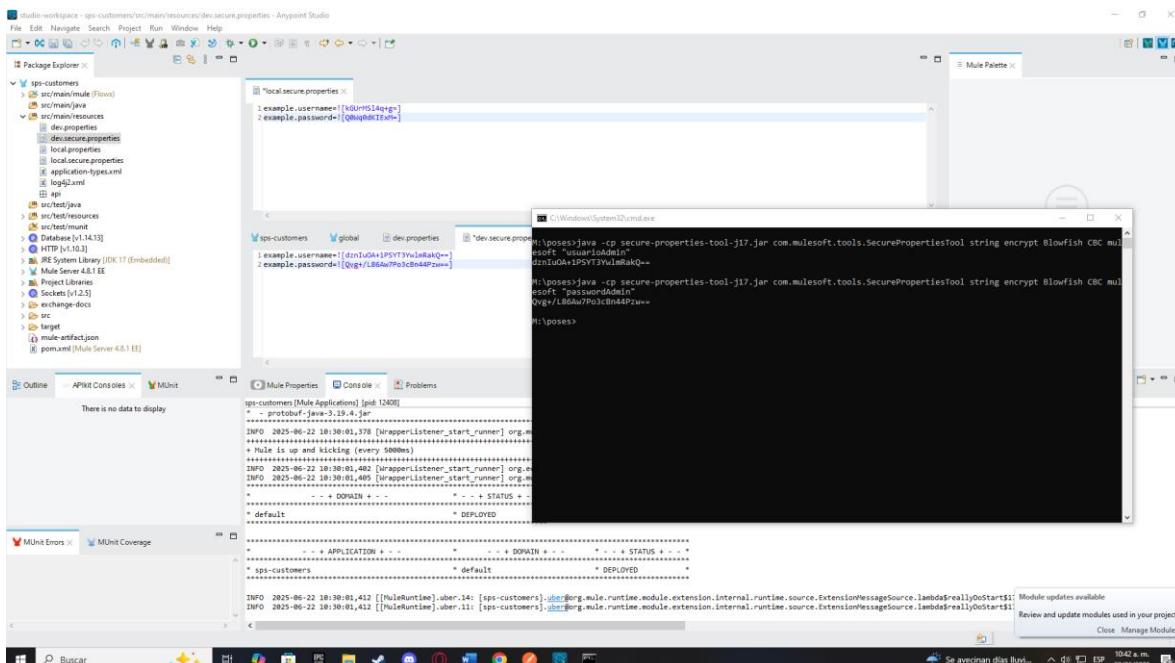
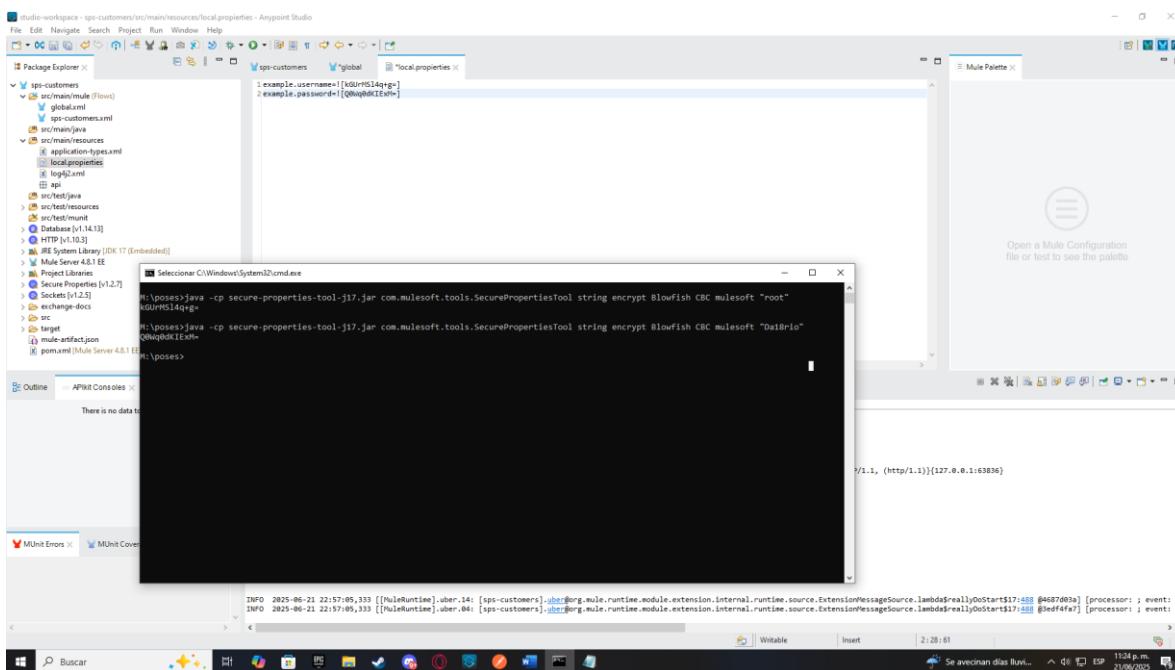
Buscamos la opción de Secure Properties Config



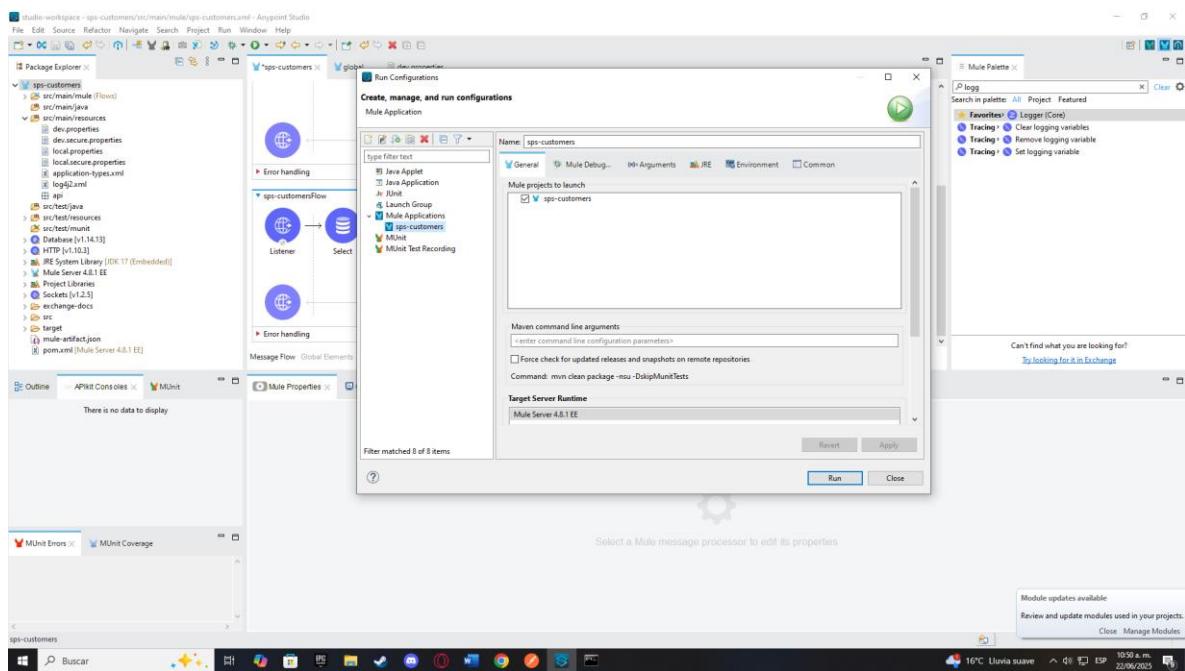
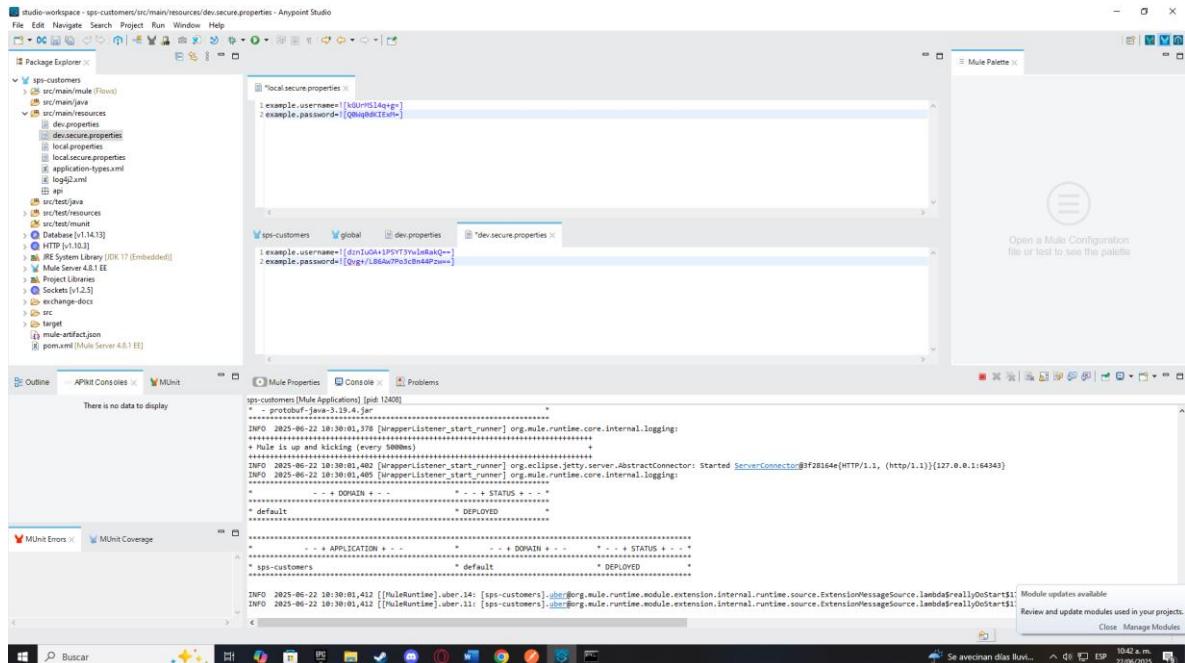
Establecemos el archivo y la llave a la que va a referenciar, junto con el algoritmo de encriptado.



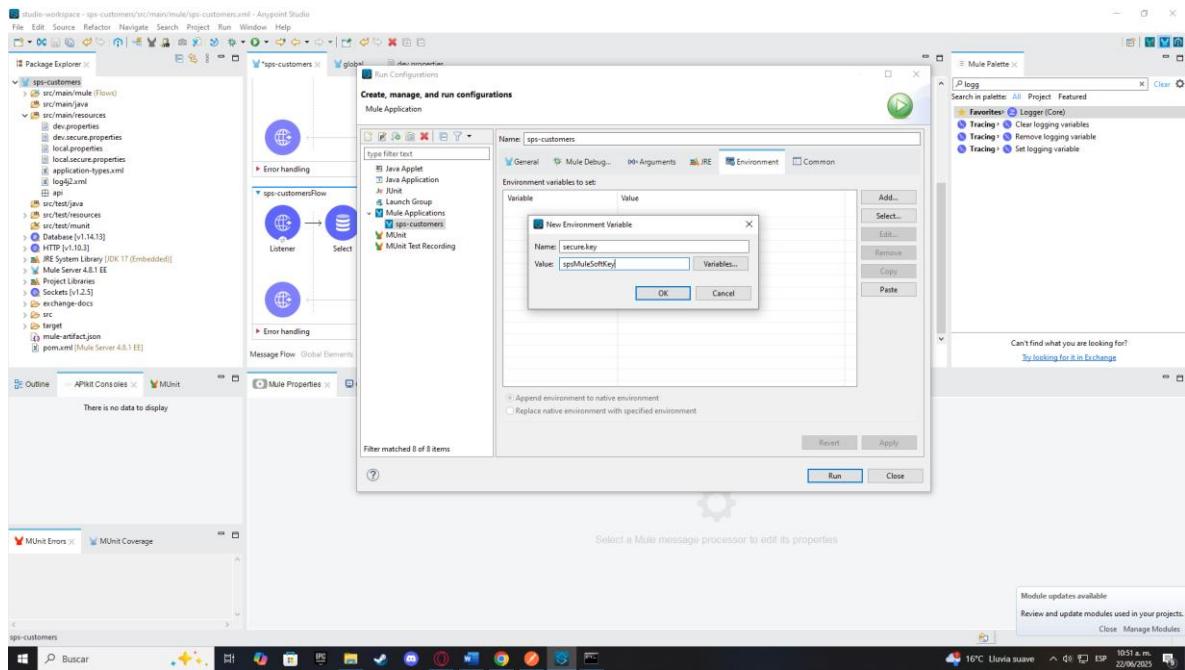
Ahora haremos contraseñas seguras (cifradas) para esto correré el jar enviado, el cual cifra las credenciales para no ponerlas como texto plano



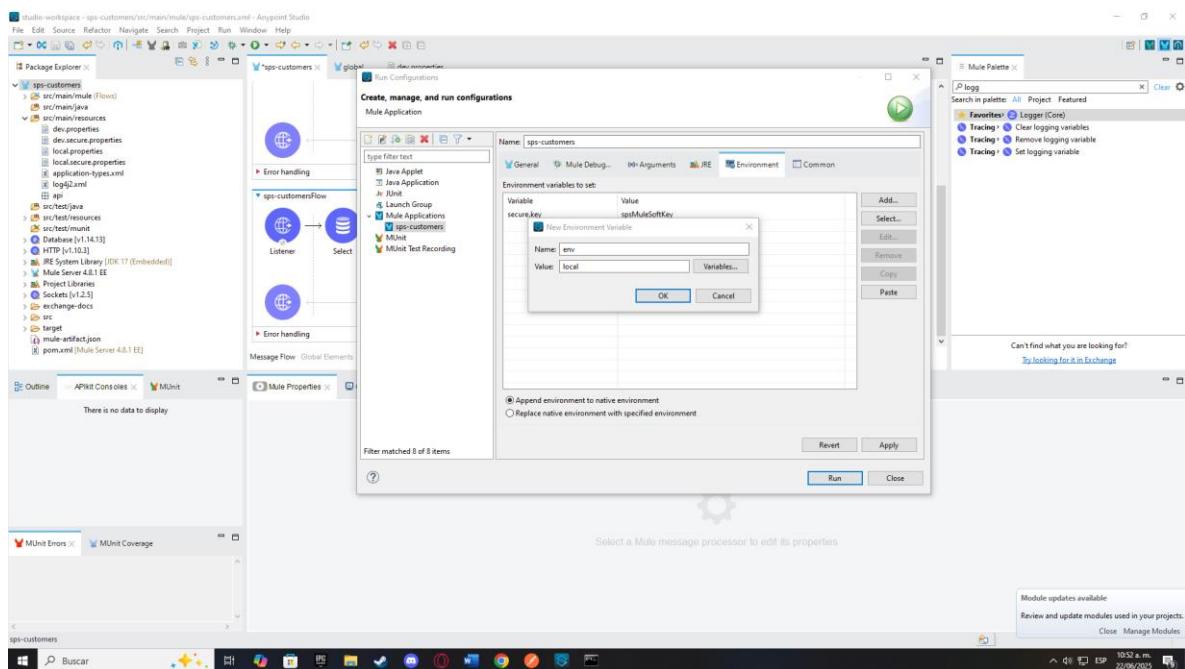
Copiamos y pegamos las credenciales encriptadas creadas por el jar en los archivos secure



Creamos una variable de entorno para proteger la api con una llave



Y creamos otra variable local para el entorno de desarrollo



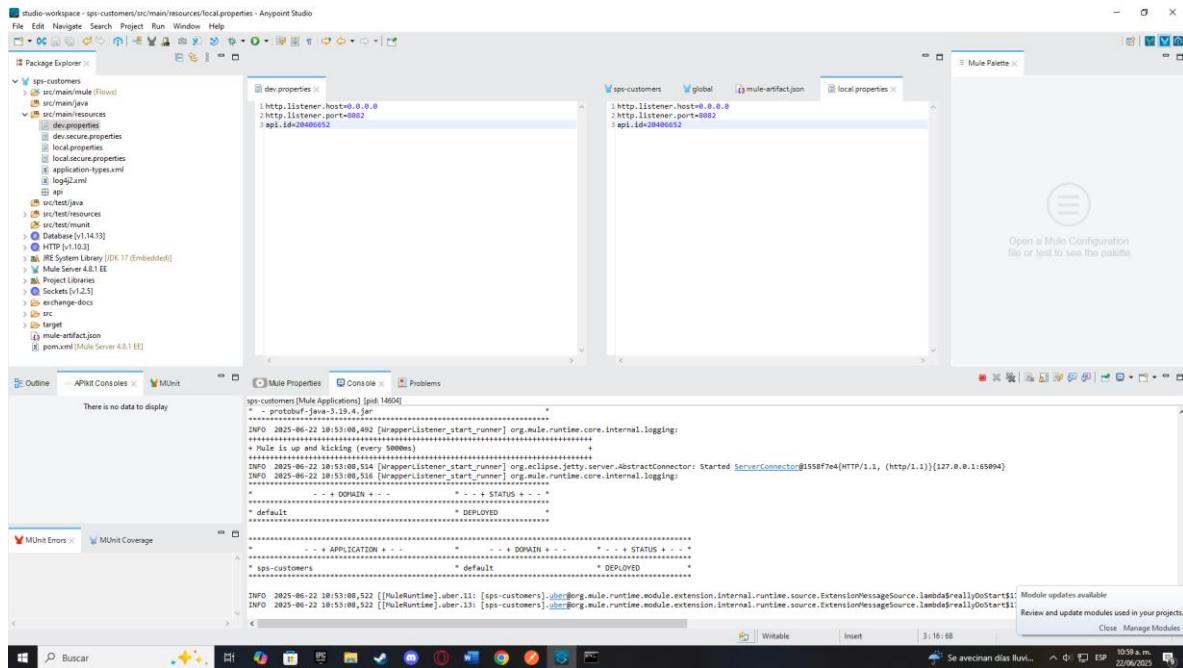
Agregaremos la API en el apartado API Manager

The screenshot shows the 'Instance Administration / Add API' screen. The 'Create new API' radio button is selected. The 'Name' field contains 'sps'. The 'Asset types' dropdown is set to 'HTTP API'. At the bottom right, there are 'Previous' and 'Next' buttons.

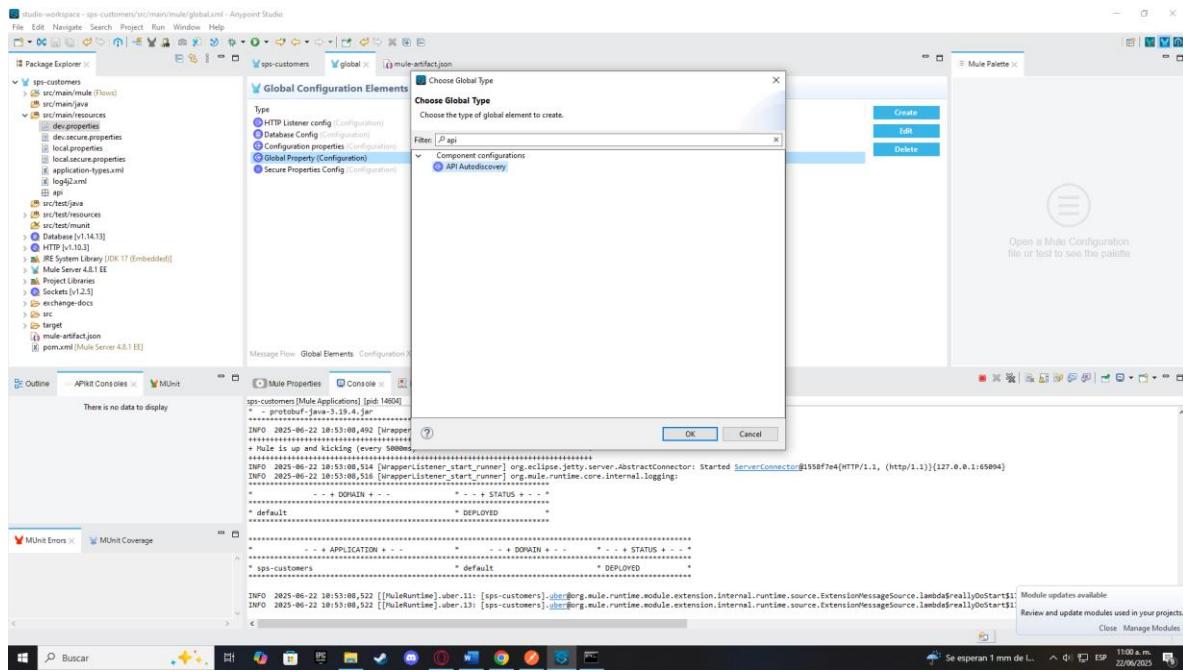
Revisaremos la creación de instancia y obtenemos el ID del Api

The screenshot shows the 'sps (v1) - API Summary' screen. It lists the API details: Type: HTTP, Asset Version: 1.0.0 (Latest), Implementation URI: N/A, API Label: -, API Version: v1. The API Status is Unregistered. In the 'Key Metrics' section, there are two charts: 'Total Requests over Time' and 'Total Policy Violations over Time'. A note at the top says: 'To complete the registration process, you need to connect this API to your Mule application using Autodiscovery. [Learn more](#)'.

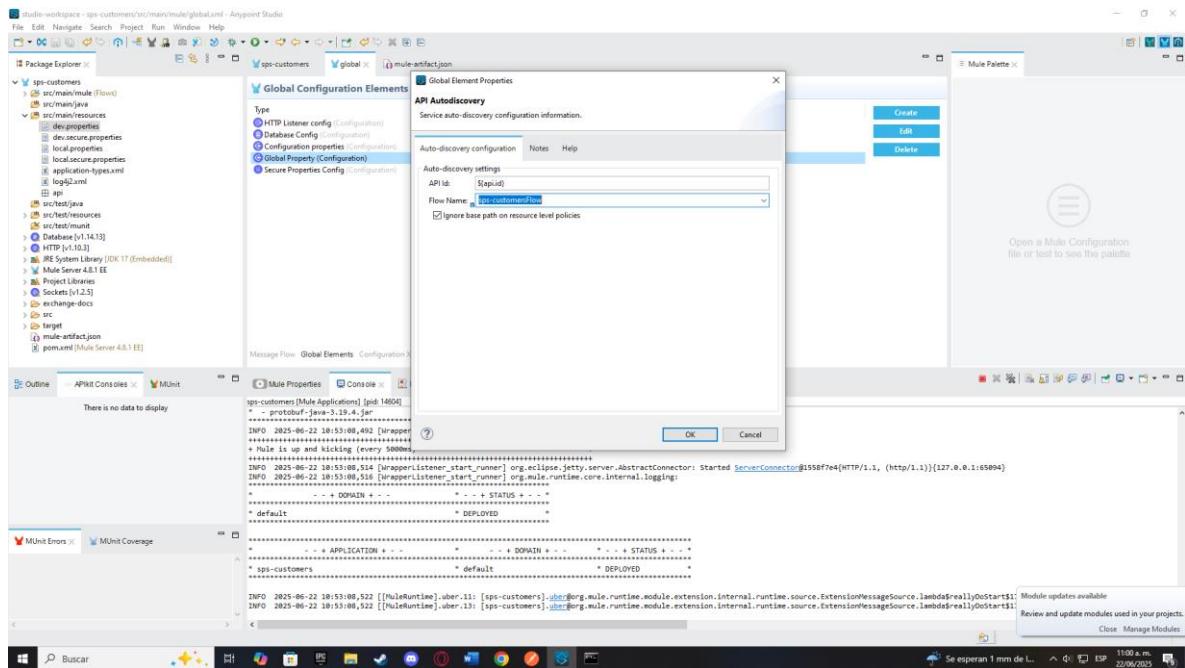
Nos vamos a los archivos de propiedades para copiarlo



Ahora crearemos la configuración para el API Autodiscovery



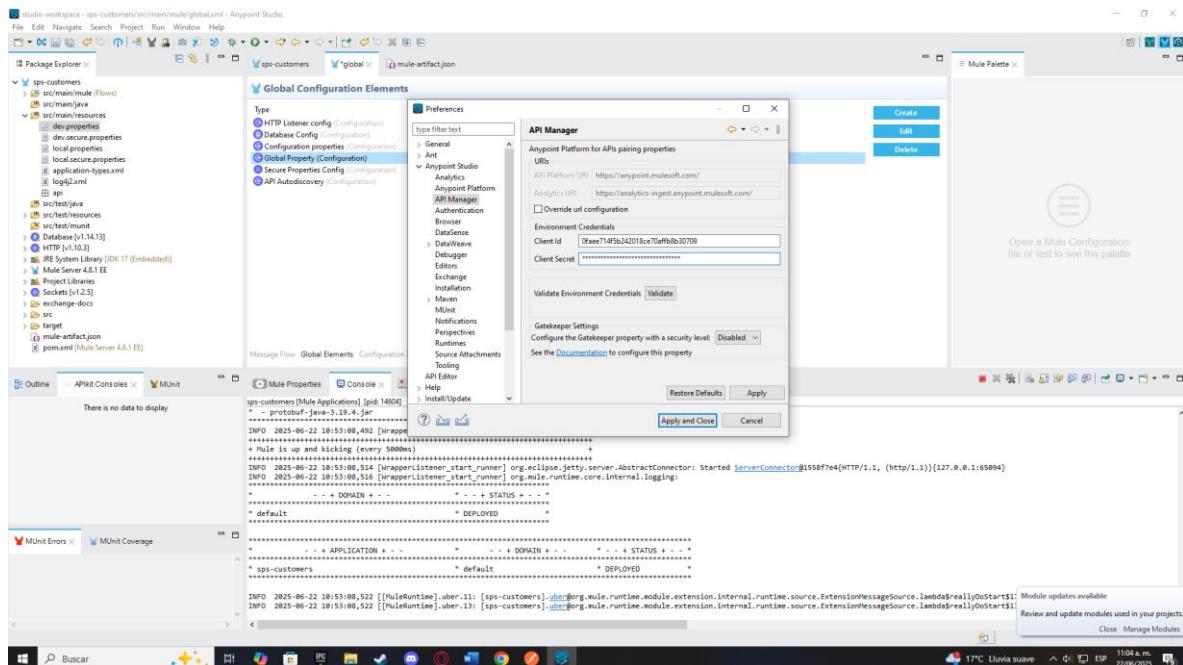
Para esto en un formulario se nos pedirá el API id de la instancia y el nombre del flujo



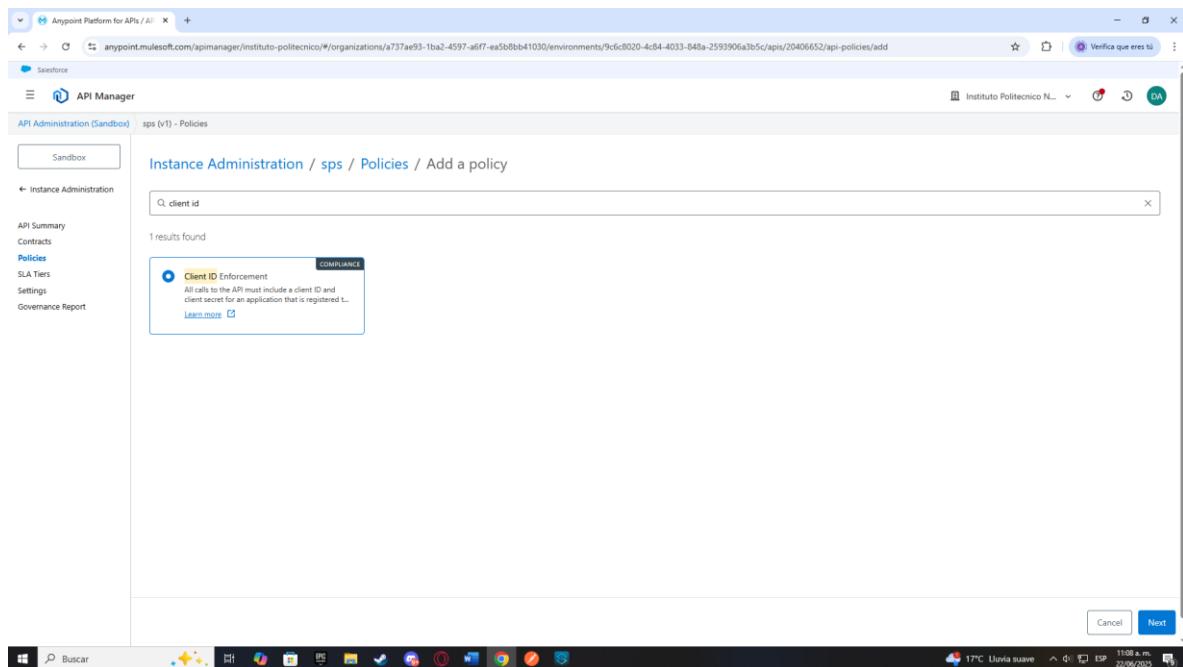
Nos otorgara las credenciales como el id y la llave secreta

Client ID	Client secret
0faee714f5b242018ce70affbb30709	67c440EEf3F4f04dD5e555482Fede1

Estas las tenemos que copiar en la configuración de AnyPoint Studio en el API Manager



Nos regresamos al portal del API Manager y buscamos las pólices para agregar una política



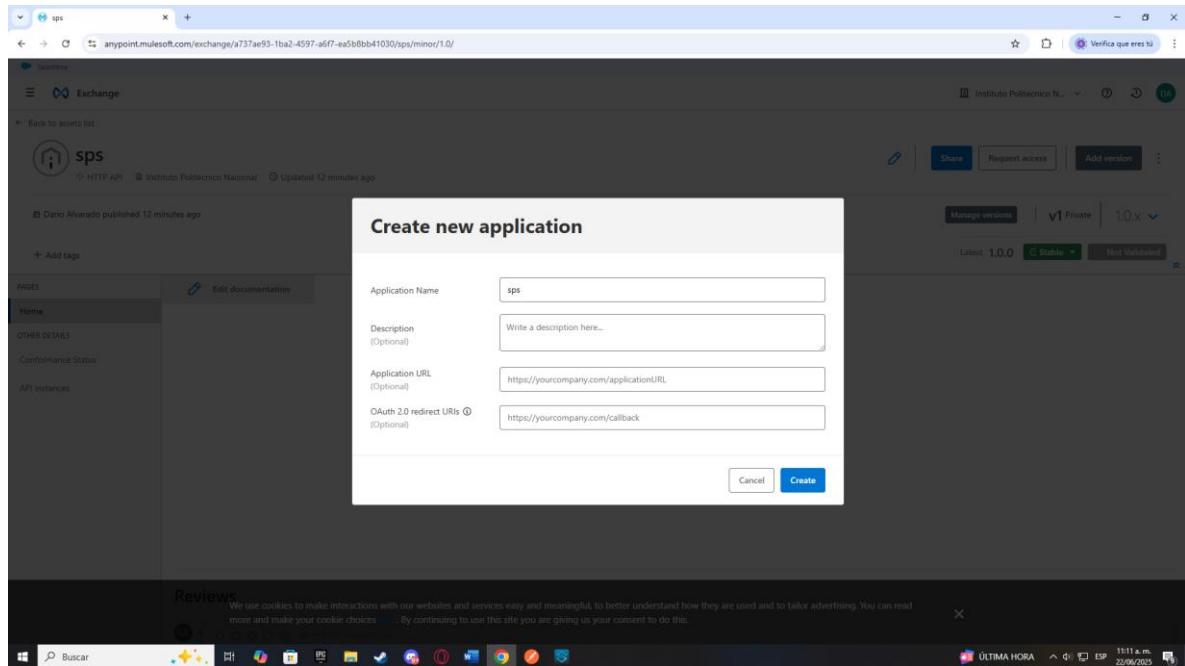
Y le colocamos una autenticación básica de HTTP, lo que hará seré bloquear la api en dado caso de que no se pase en el header de la petición el usuario y contraseña,

The screenshot shows the Anypoint Platform API Manager interface. The URL in the browser is anypoint.mulesoft.com/apimanager/instituto-politecnico/#/organizations/a737ae93-1ba2-4597-a6f7-ea5bb8b41030/environments/9cb68020-4c84-4033-848a-259390e3305c/apis/configure?groupId=60ef9520-2.... The page title is "Anypoint Platform for APIs / API Manager". The left sidebar shows "API Administration (Sandbox)" and "sps (v1) - Policies". The main content area is titled "Instance Administration / sps / Policies / Configure Client ID Enforcement policy". It includes sections for "Credentials origin" (HTTP Basic Authentication Header selected), "Advanced options" (Configure policy version, methods and resources), and buttons for "Previous" and "Apply".

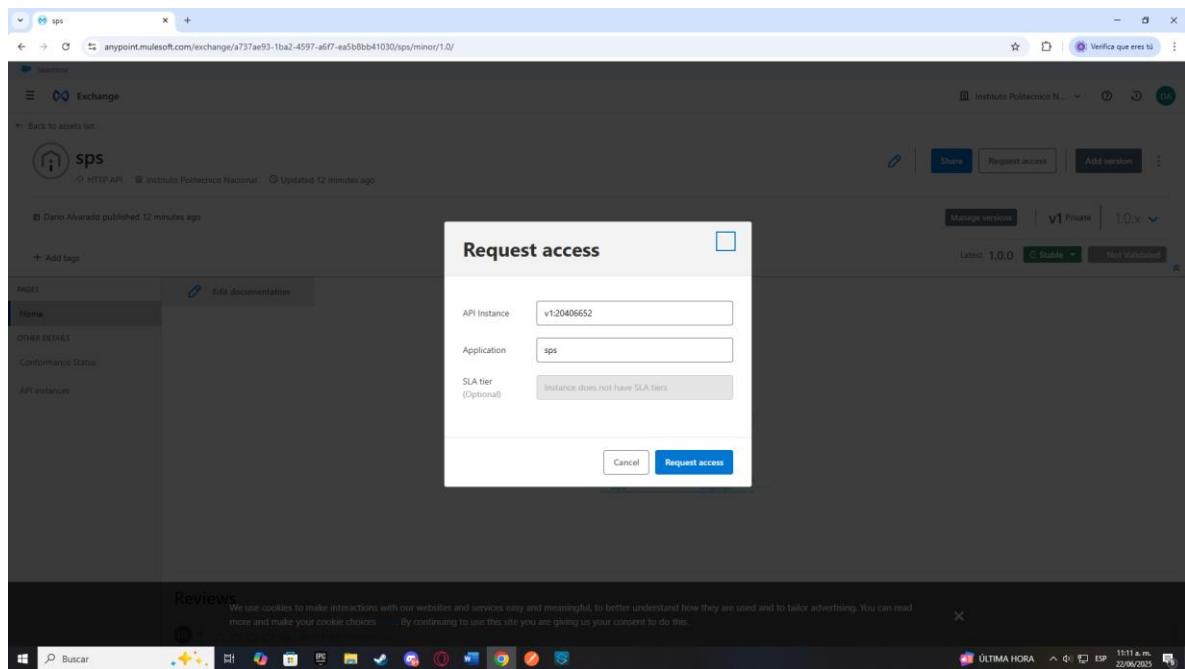
Al aplicar se nos dará un request acces

The screenshot shows the "Request access" dialog box overlaid on the API instance details page. The dialog has fields for "API Instance" (set to v120406652) and "Application" (with a dropdown placeholder "Select application"). Below these fields is a note: "SLA tier (Optional) Instance does not have SLA tiers". At the bottom of the dialog are "Cancel" and "Request access" buttons.

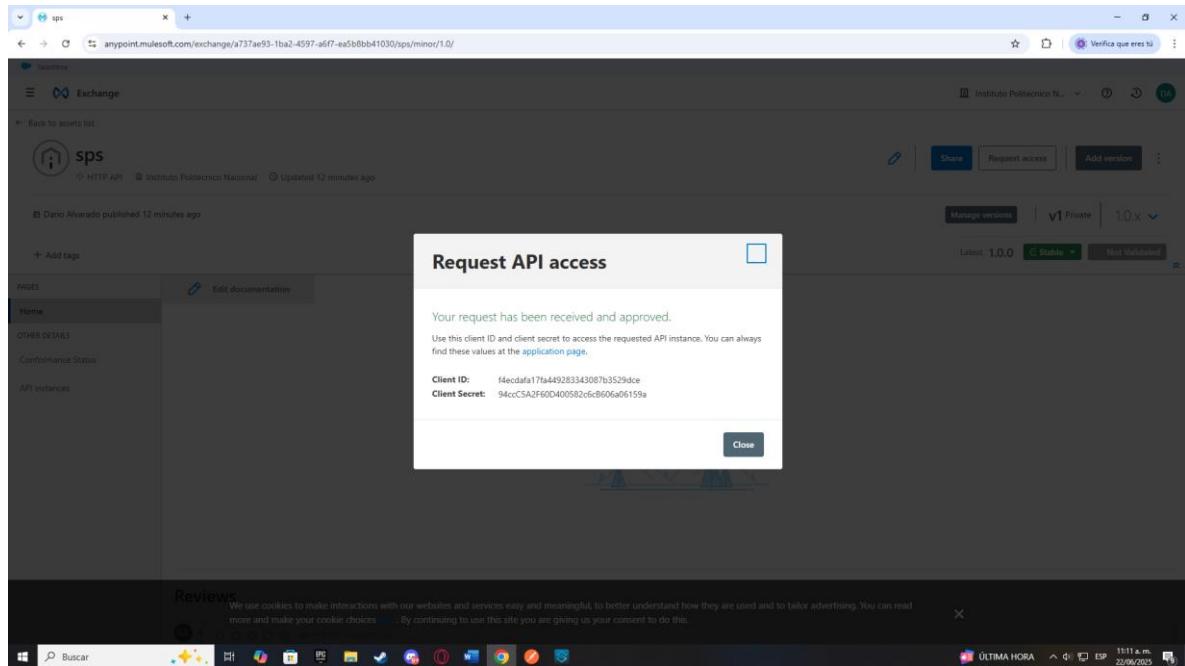
Y creare una nueva aplicación



Despues tenemos que seleccionar la instancia de la API



Y como resultado nos da el client id y el client secret para acceder al endpoint



//BASE DE DATOS

Cree la base de datos para la realización de la practica y así poder visualizar un resultado fiel.

A screenshot of MySQL Workbench. The left sidebar shows the 'Navigator' with a list of schemas, including 'ad_sistemas', 'advanced_fx_me', 'alumnos', 'db_cursos', 'beers', 'beyanaps', 'chat', 'complex_fx_me_d', 'complex_fx_me_p', 'crethos', 'crudapp', 'cursos', 'db_cursos', 'ecommerce', 'ejemploweb', 'empresitas', 'empresas', 'entrevistas', 'oficinas', 'pestorbiblioteca', 'pestortress', 'hotels', 'hotels_innos'. The main pane shows a SQL editor with the command 'create database sp1_entrevistas;'. Below the editor is an 'Output' pane showing the result: '1 row(s) affected'. The status bar at the bottom indicates '19:27:29' and 'Duration / Fetch 0.016 sec'.

Cree una tabla sencilla con un id,nombre,apellido, correo y password

MySQL Workbench

Local instance MySQL8000 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- ad_sistemas
- advanced_fn_me_...
- alumnos
- ap_tareas
- beers
- bitacoras
- cancil_vt
- chat
- complex_fn_me_d
- control_de_stock
- creditos
- crudapp
- curso
- db_tareas
- economica
- ecommerce
- empleados
- empresa
- examen
- gestionbiblioteca
- holamundo
- hotatatura
- hotels_innos
- Information
- Administration Schema

No object selected

SQL File T X

```
1 CREATE TABLE customer (
2   id INT AUTO_INCREMENT PRIMARY KEY,
3   nombre VARCHAR(100),
4   apellido VARCHAR(200),
5   correo VARCHAR(250),
6   password VARCHAR(250)
7 );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	19:27:29	create database sp_entrevista	1 row(s) affected	0.016 sec
2	19:33:11	use sp_entrevista	0 row(s) affected	0.000 sec

Object Info Session Buscar 16°C Nublado 07:35 p. m. 21/06/2023

E insertamos 5 registros en la base de datos

MySQL Workbench

Local instance MySQL8000 X

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- ad_sistemas
- advanced_fn_me_...
- alumnos
- ap_tareas
- beers
- bitacoras
- cancil_vt
- chat
- complex_fn_me_d
- control_de_stock
- creditos
- crudapp
- curso
- db_tareas
- economica
- ecommerce
- empleados
- empresa
- examen
- gestionbiblioteca
- holamundo
- hotatatura
- hotels_innos
- Information
- Administration Schema

No object selected

SQL File T X

```
1 CREATE TABLE customer (
2   id INT AUTO_INCREMENT PRIMARY KEY,
3   nombre VARCHAR(100),
4   apellido VARCHAR(200),
5   correo VARCHAR(250),
6   password VARCHAR(250)
7 );
```

```
8 INSERT INTO customer (nombre, apellido, correo, password) VALUES
9   ('Juan', 'Perez', 'juan.perez@example.com', 'password123'),
10  ('Maria', 'Gomez', 'maria.gomez@example.com', 'maria456'),
11  ('Carlos', 'Rodriguez', 'carlos.rodriguez@example.com', 'carlos789'),
12  ('Ana', 'Lopez', 'ana.lopez@example.com', 'ana321'),
13  ('Luis', 'Martinez', 'luis.martinez@example.com', 'luis654');
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	19:33:11	use sp_entrevista	0 row(s) affected	0.000 sec
3	19:35:14	CREATE TABLE customer (id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(100), apellido VARCHAR(200), correo VARCHAR(250), p...)	0 row(s) affected	0.002 sec

Object Info Session Buscar 16°C Nublado 07:35 p. m. 21/06/2023