Notas de la práctica

# Fecha: 7 de Mayo de 2023

## Práctica Mulesoft

Hoy en día, se está desarrollando una aplicación móvil que muestra información básica del universo de Starwars, como las películas existentes, los personajes principales de todas las películas o las naves espaciales.  
  
En esta ocasión, se espera tener la información de los personajes de las películas en un formato CSV para que la aplicación muestre estos datos en una tabla tipo excel y el usuario final pueda comparar las características de los personajes.  
Se sabe que el origen de estos datos es un API(Swapi) la cual devuelve la información en un formato JSON. En este sentido, el implementador deberá diseñar(con RAML o Swagger) una especificación de API y desarrollar un proyecto Mule que permita recuperar la información de todos los personajes de Starwars que la Swapi provee y entregarlos en un formato CSV, ejemplo:

name,height,mass,hair\_color,skin\_color,eye\_color,bith\_year,gender  
Luke Skywalker,172,77,blond,fair,blue,19BBY,male

C-3PO,167,75,n/a,gold,yellow,112BBY,n/a

Como segundo requisito, se deberá proveer la funcionalidad de que a través de un Query parameter se podrá filtrar el resultado recibido en el CSV basándose en el género del personaje, ejemplo:  
gender=female  
  
name,height,mass,hair\_color,skin\_color,eye\_color,bith\_year,gender

Leia Organa,150,49,brown,light,brown,19BBY,female

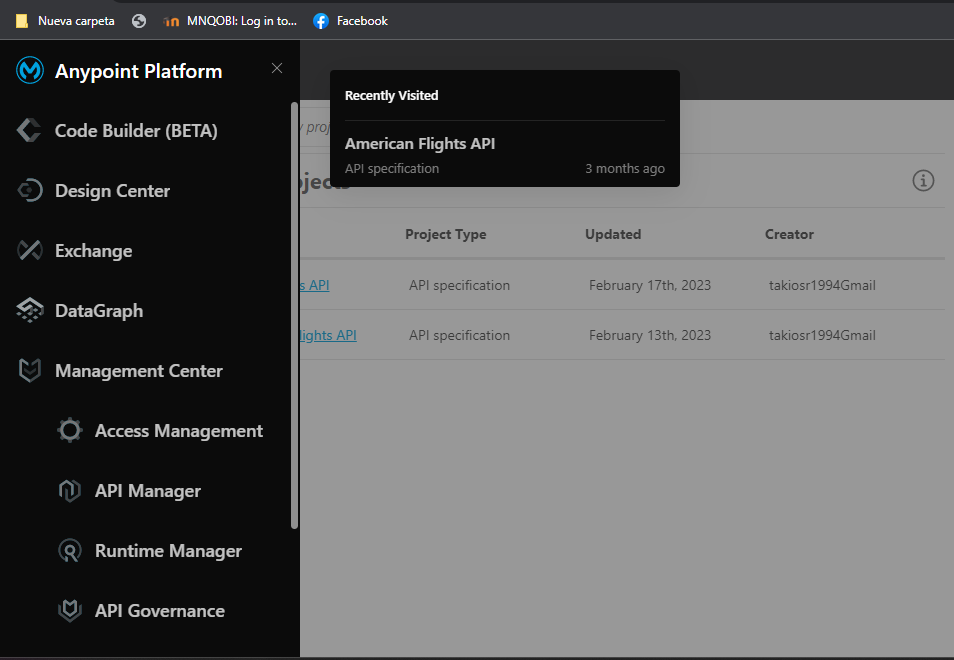
Beru Whitesun lars,165,75,brown,light,blue,47BBY,female

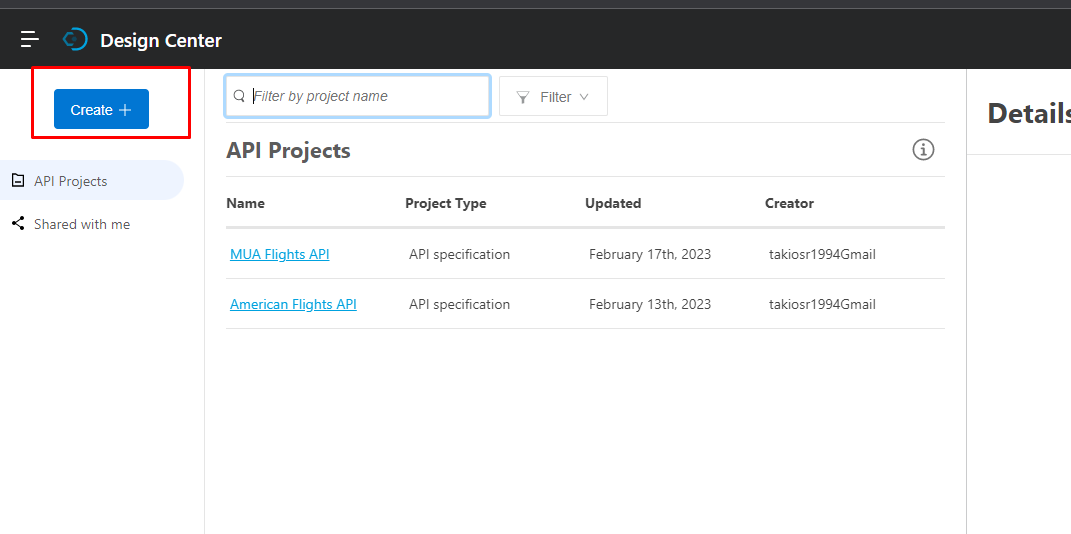
## Consideraciones:

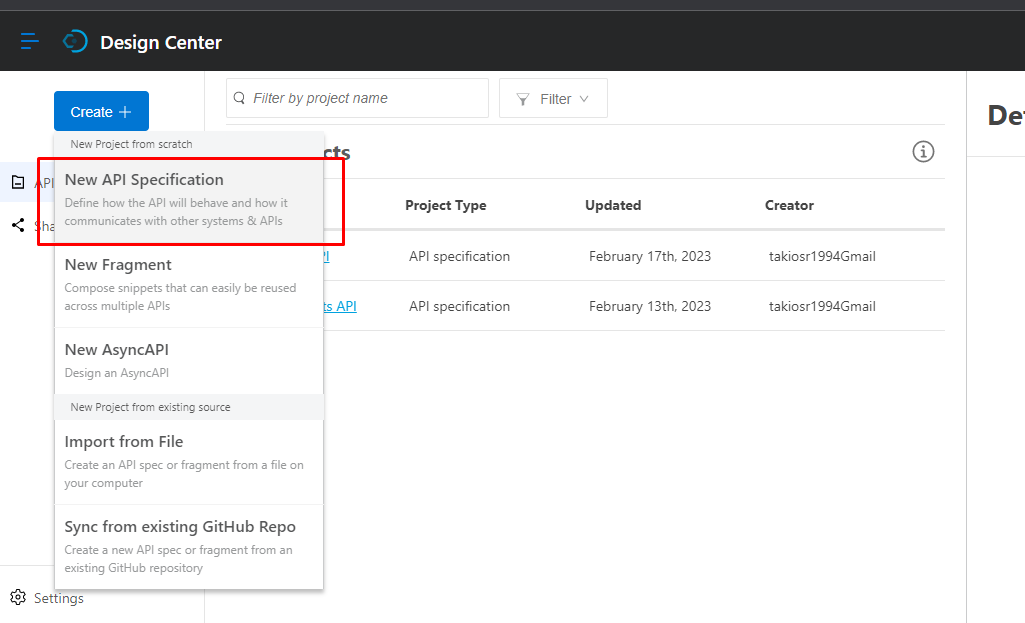
* Se evaluará la versatilidad con la que se diseñe la especificación de API, donde se espera que las buenas prácticas en diseño de APIs sean aplicadas.
* Se evaluará cómo se utilizan los conectores, así como la lógica implementada para resolver el problema planteado.
* El implementador deberá contar con un repositorio público para compartir su proyecto Material.
* <https://swapi.dev/> API de donde se obtendrá la información de los personajes, es de uso público.

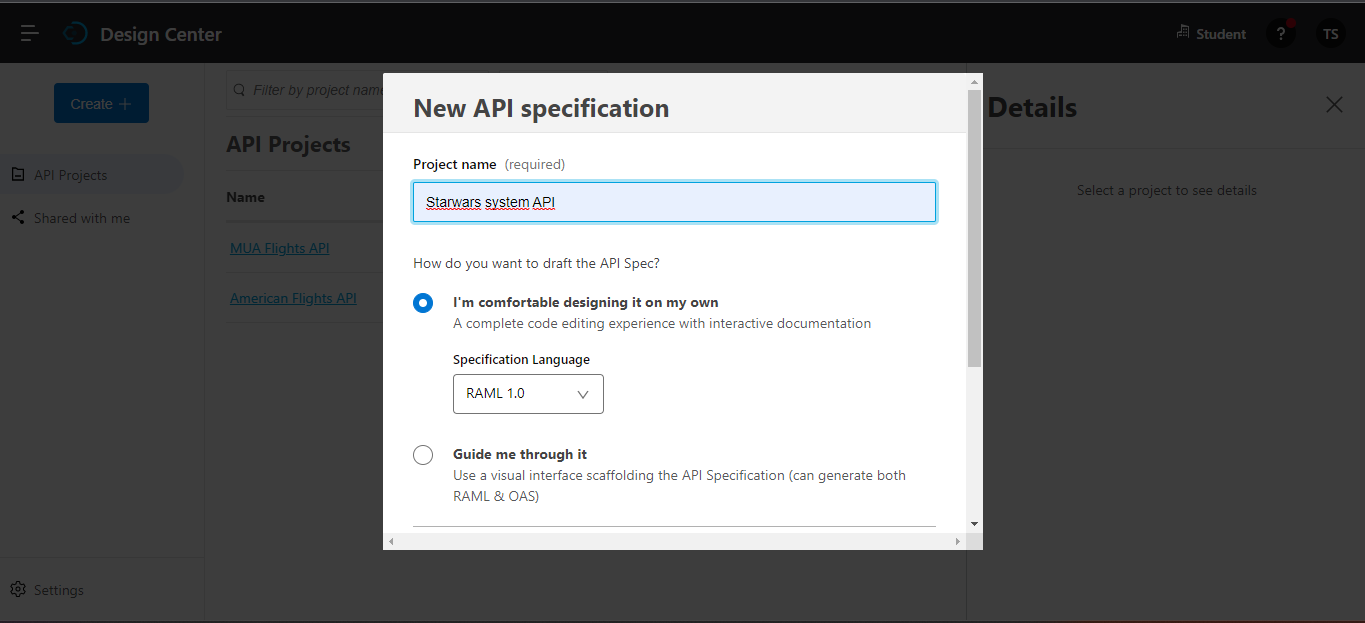
## Desarrollo

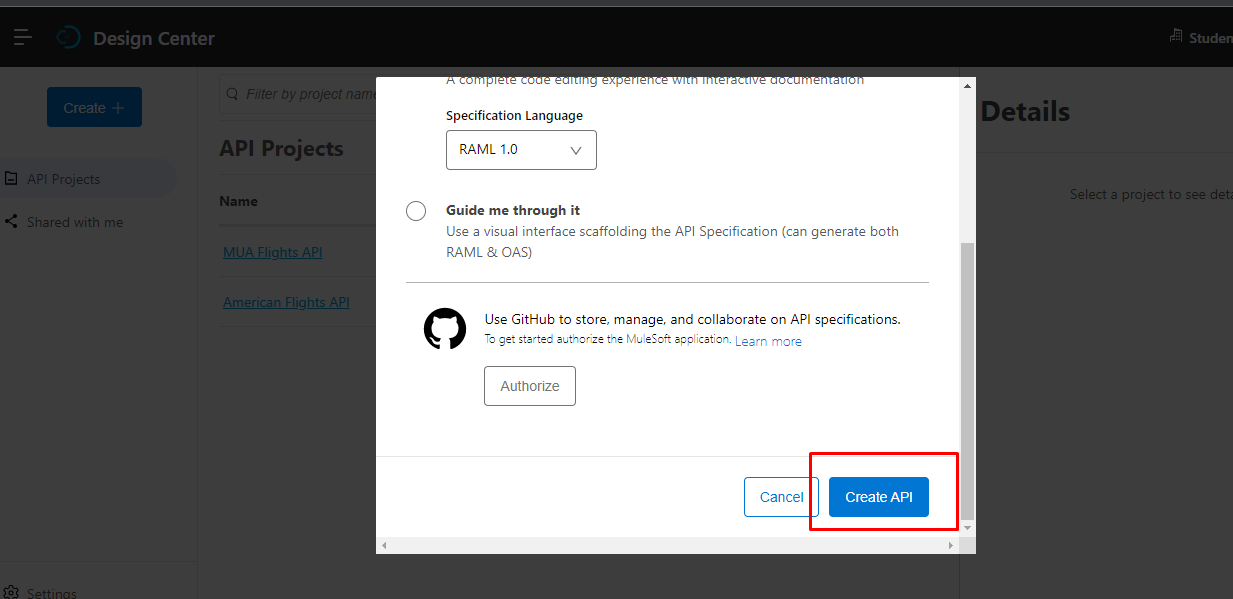
* Como primer paso nos dirigimos al Design center de Anypoint Platform.

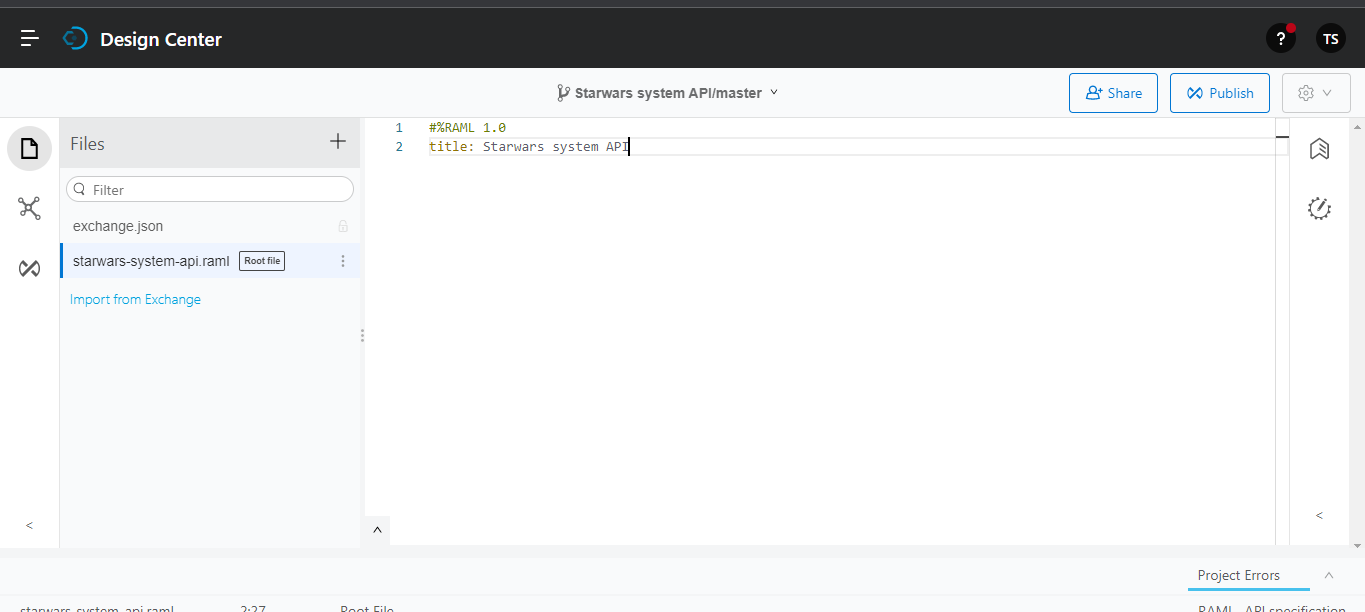


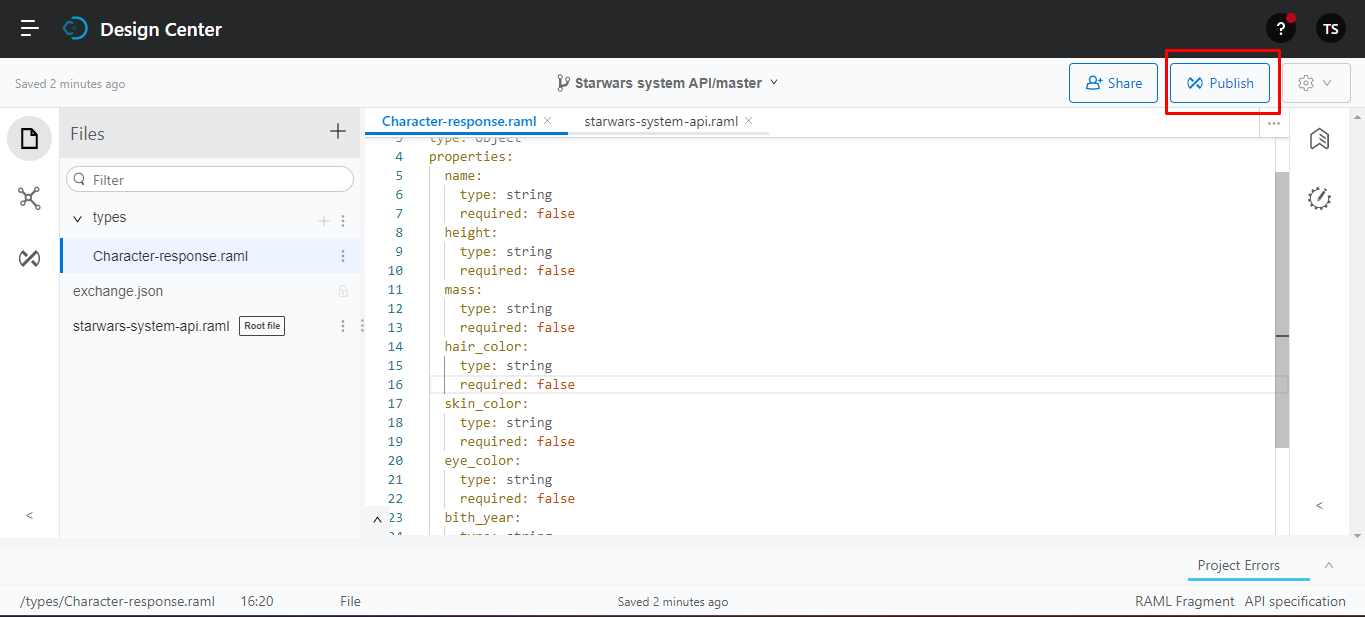
Después damos clic en la opción de crear   


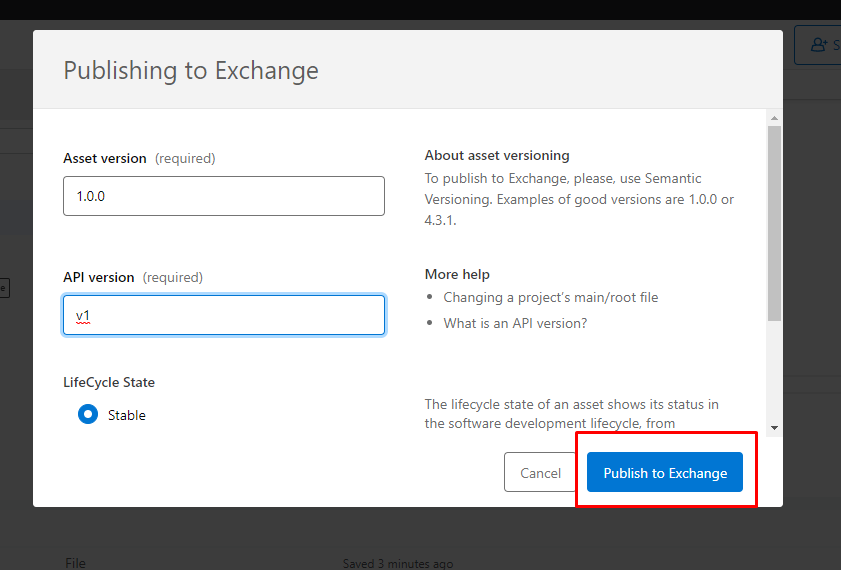
Seleccionamos la opción “New API Specification”  


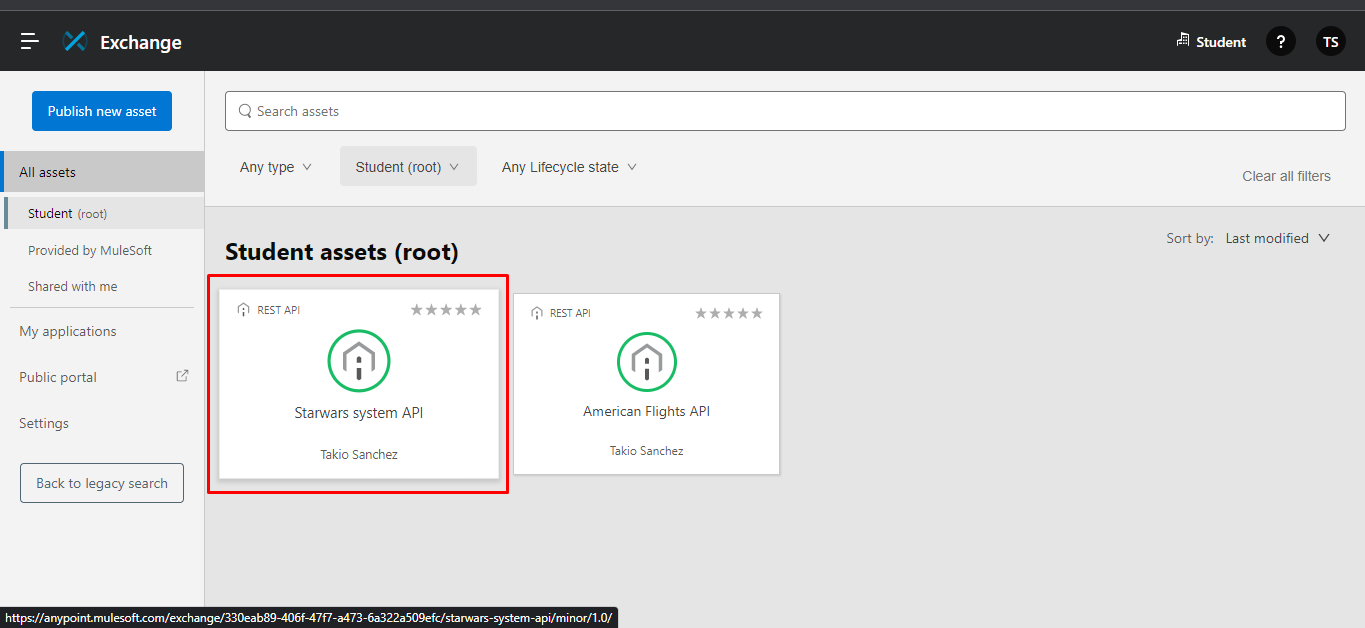
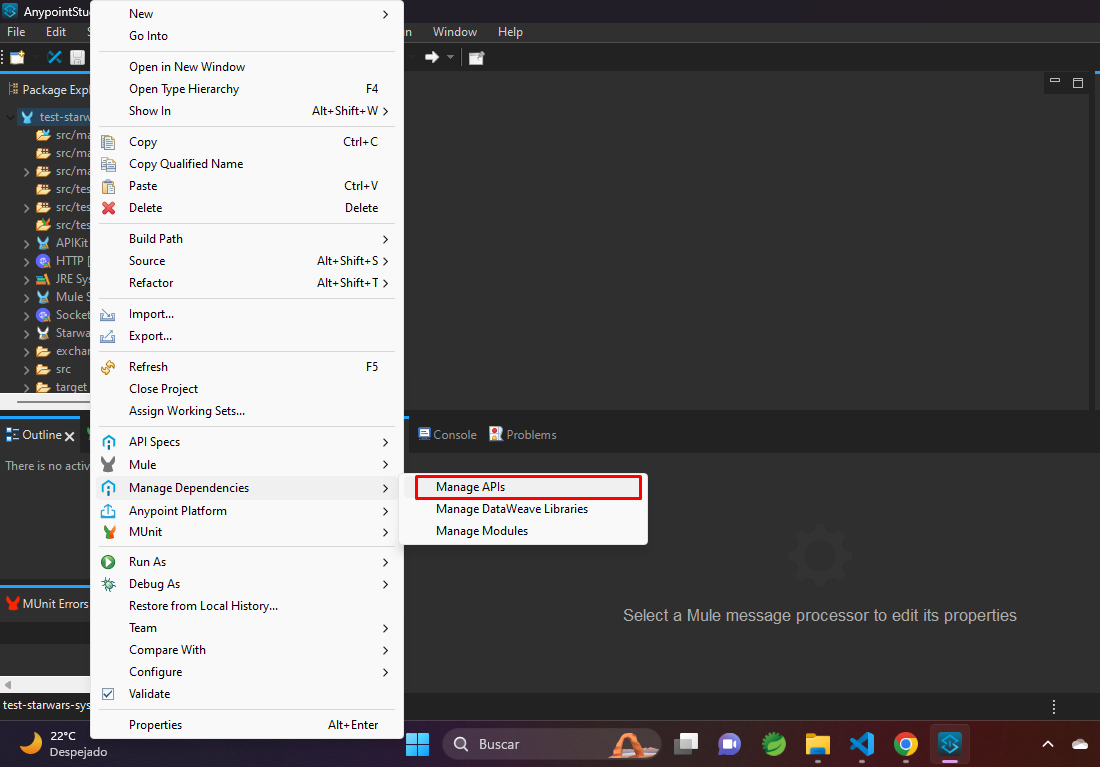
Nombramos nuestra especificación en nuestro caso será “Starwars system API”  


Y le damos clic en el botón “Create API”  


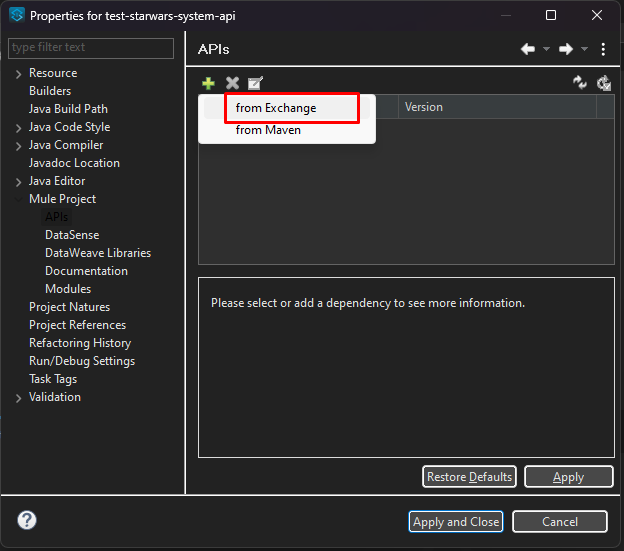
Nos aparece la siguiente ventana donde realizamos la especificación que necesitamos.  


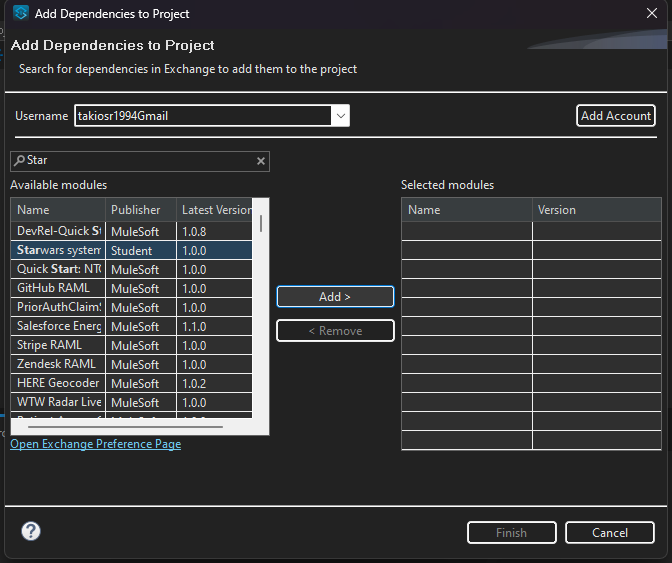
Una vez realizada la especificación, damos clic en el botón Publish  


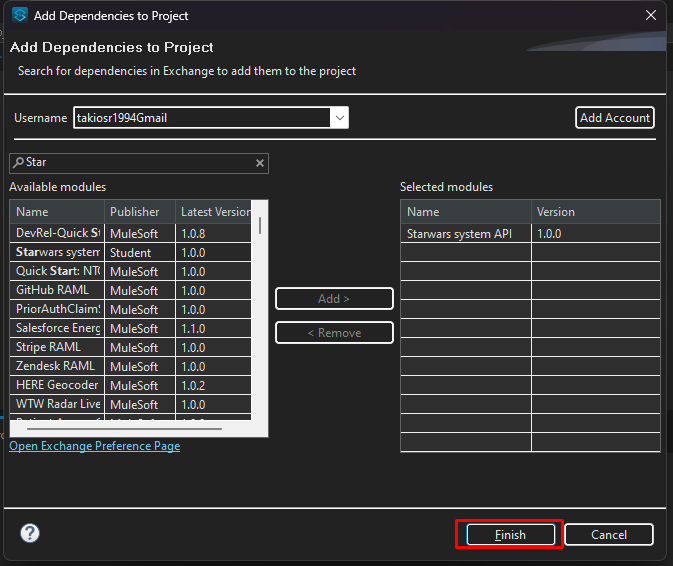
Dejamos la opciones como están y damos clic en el botón Publish to Exchange   


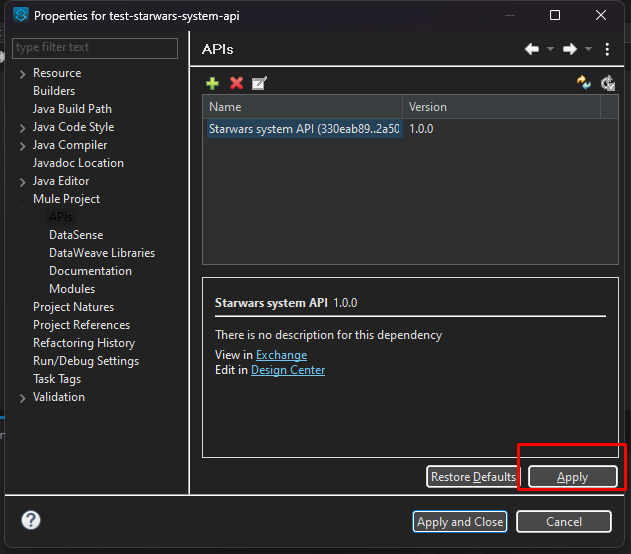
En el Exchange seleccionar la especificación que hemos creado  
  
  
Después nos vamos al Anypoint Studio, donde crearemos nuestra respectiva API, una vez creada le damos clic derecho para seleccionar la opción Manage Dependencies>Manage APIs  
  


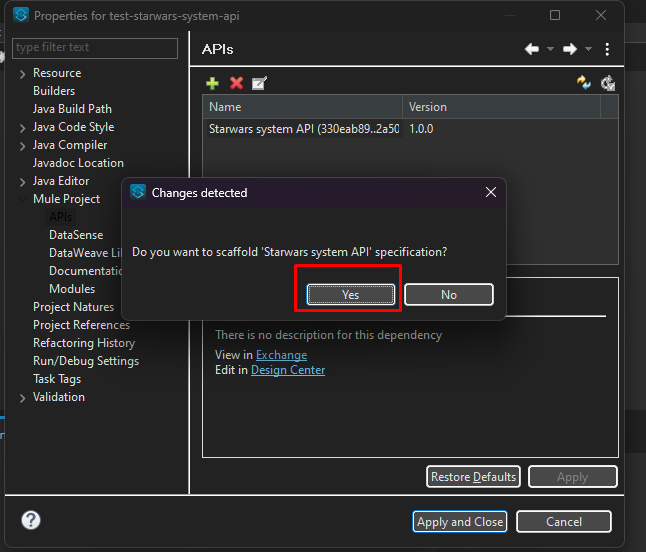
De la ventana seleccionamos en agregar botón + y seleccionamos from Exchange

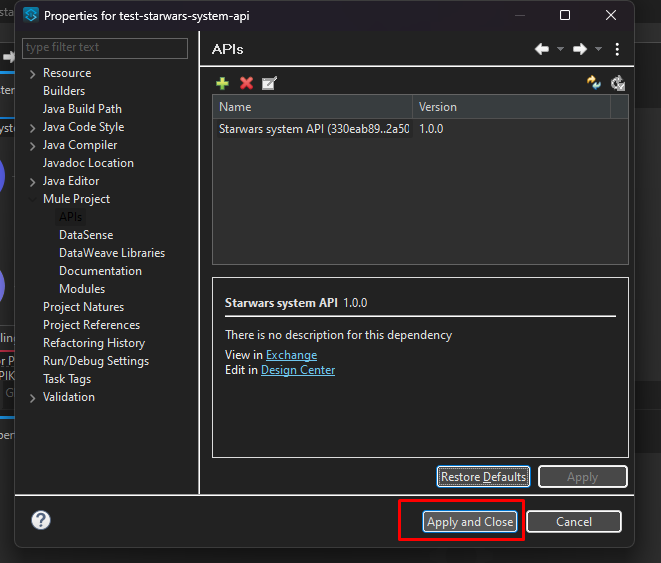
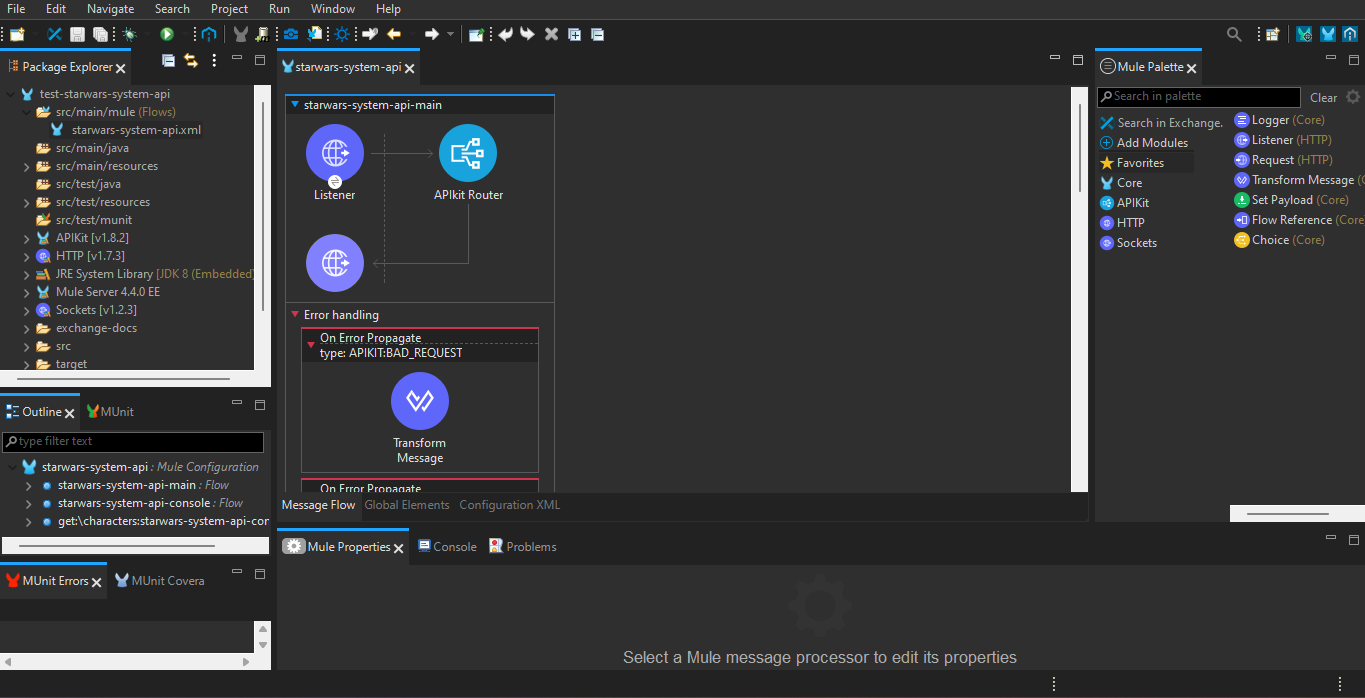


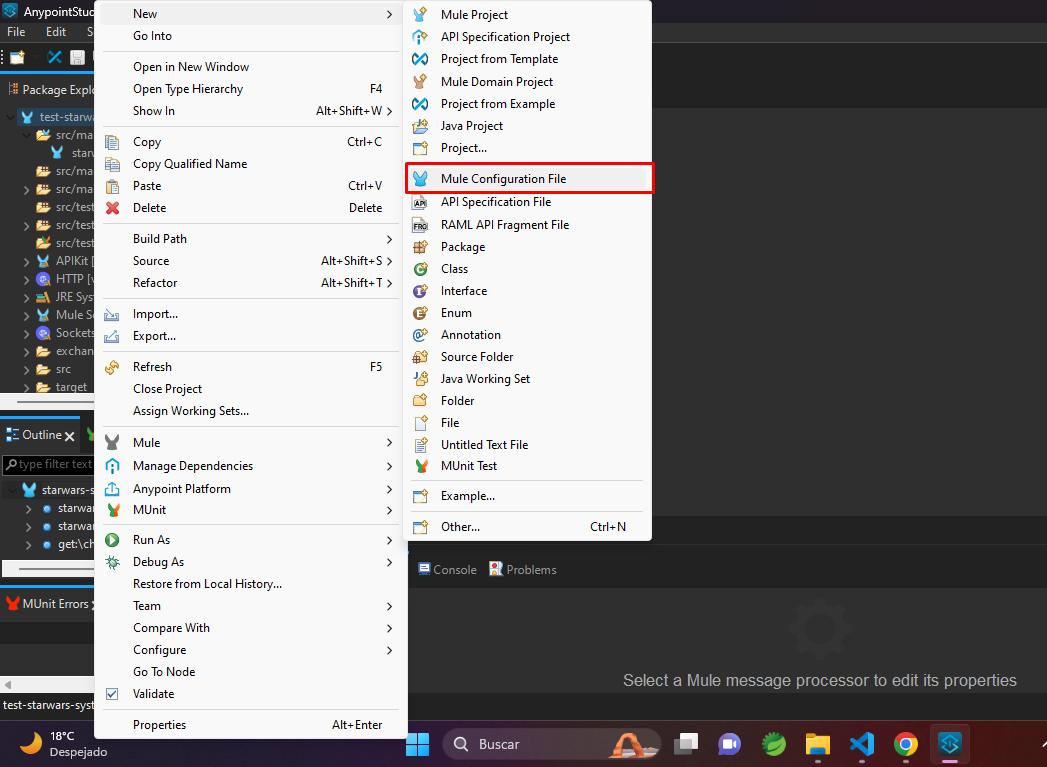
Seleccionamos la dependencia que queremos agregar, en este caso será la nueva especificación que agregamos al Exchange.  


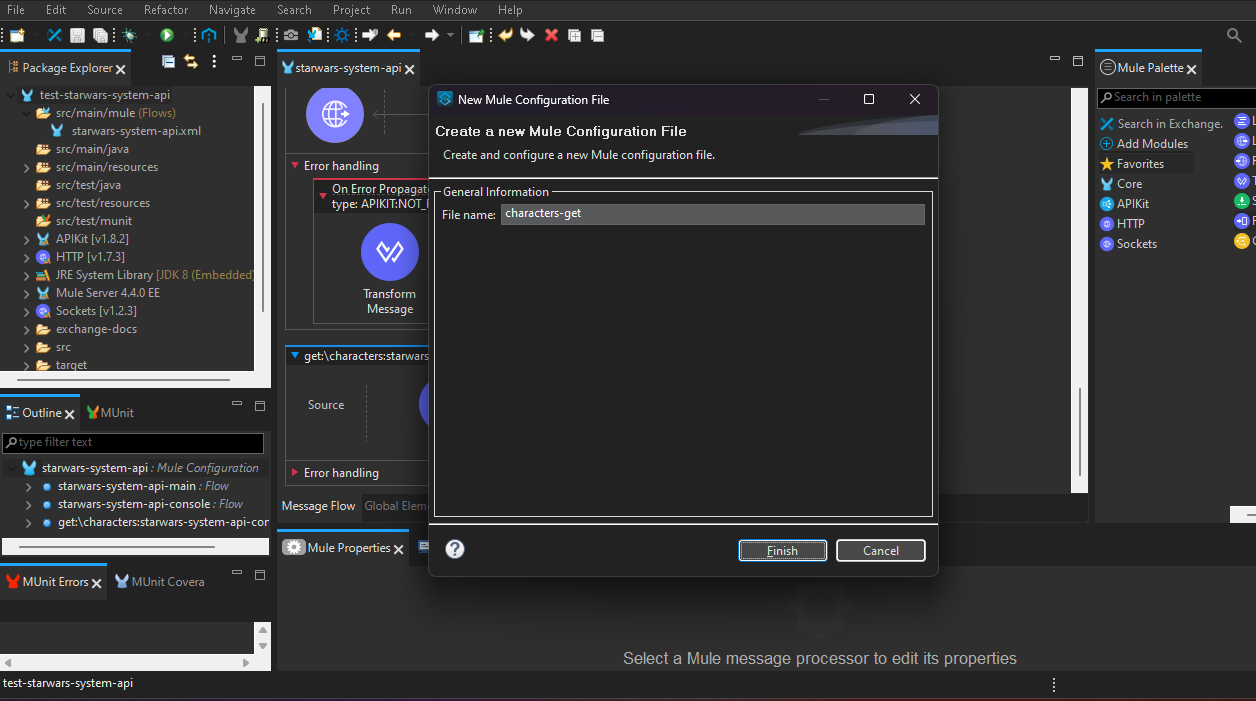
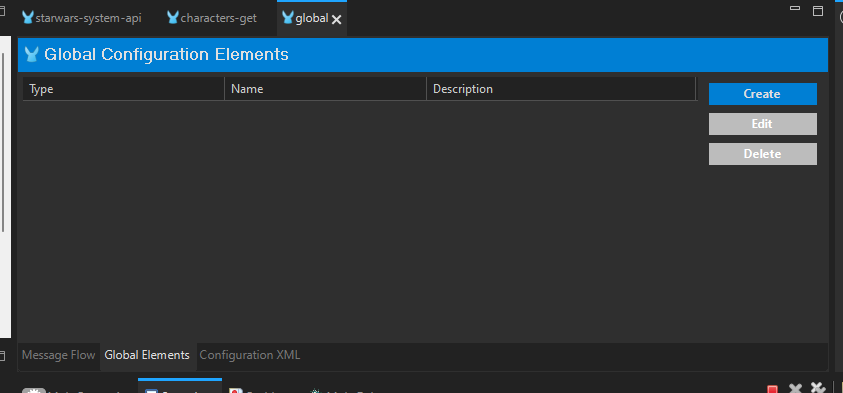
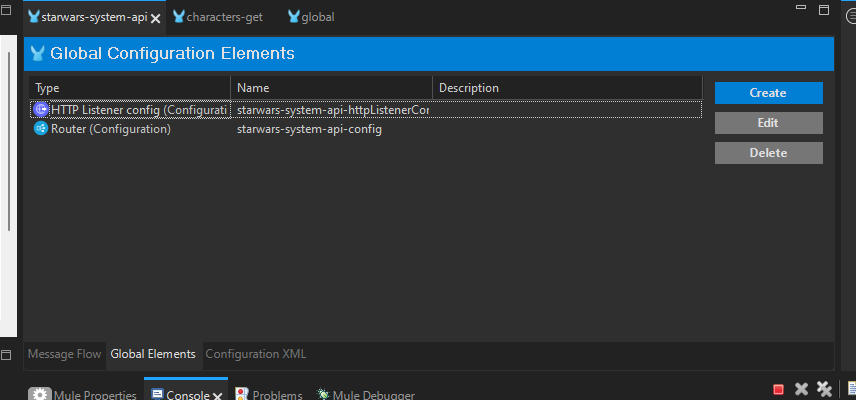
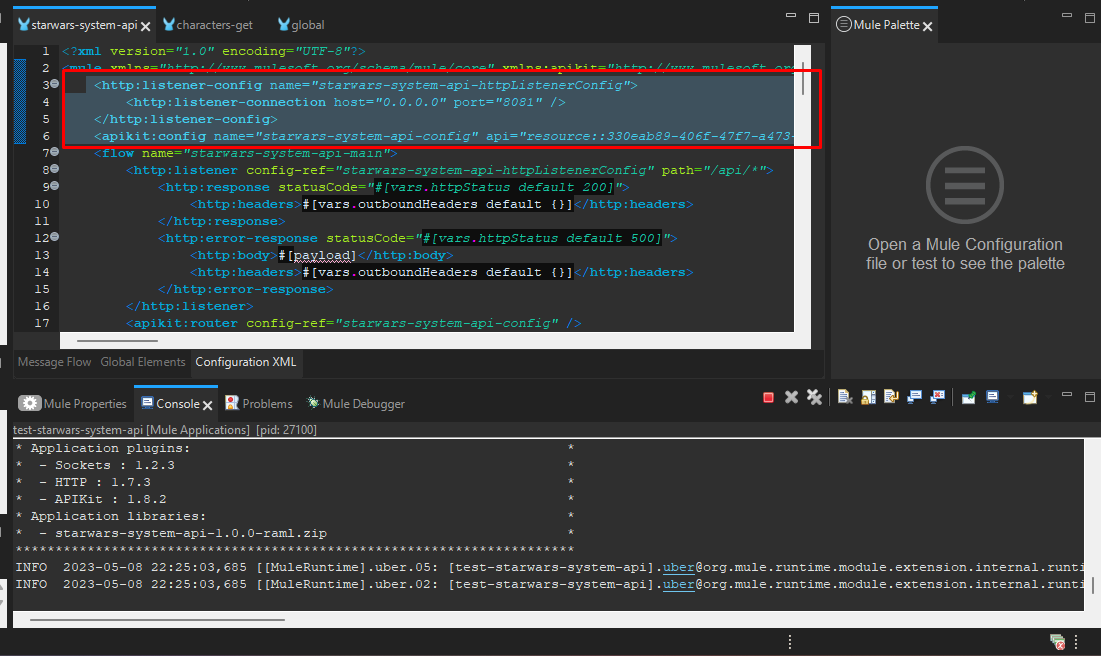
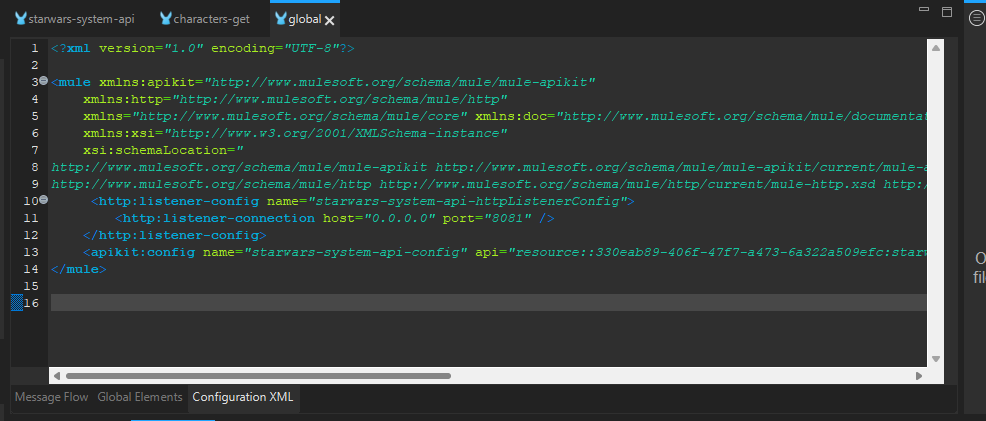
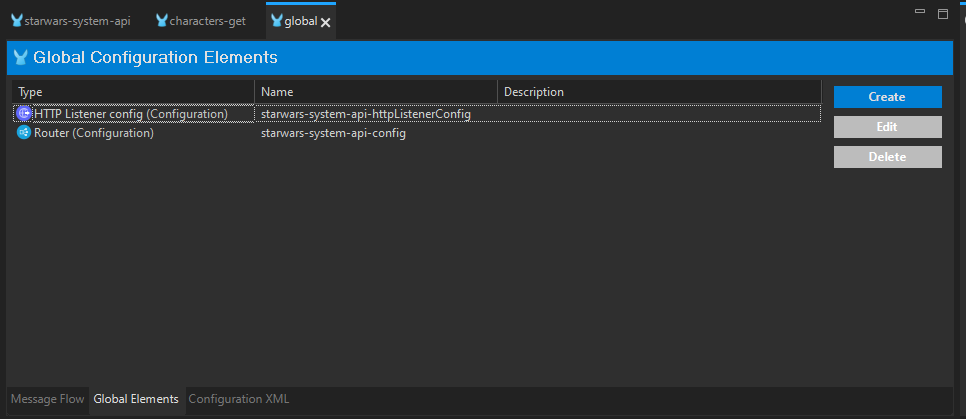
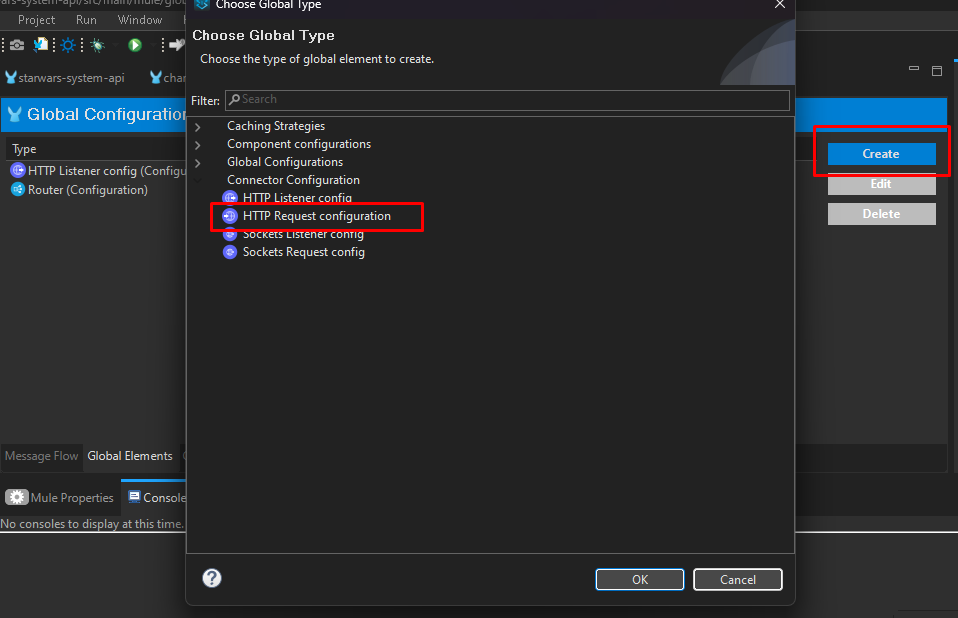
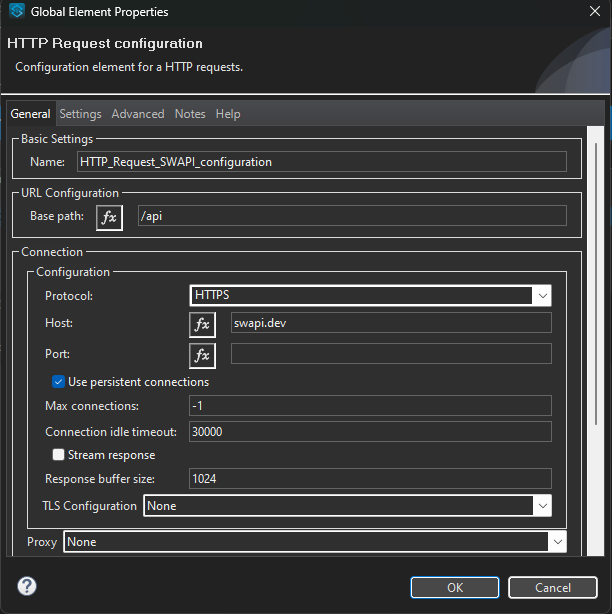
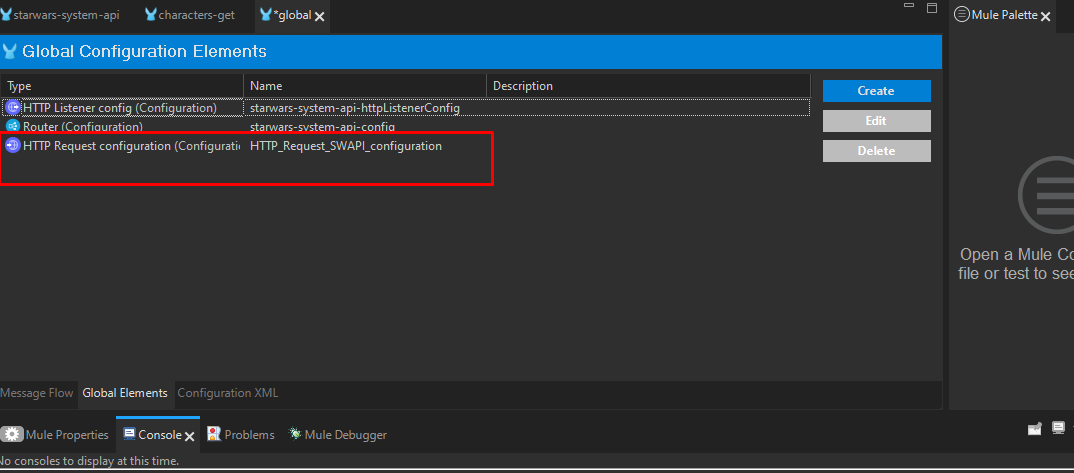
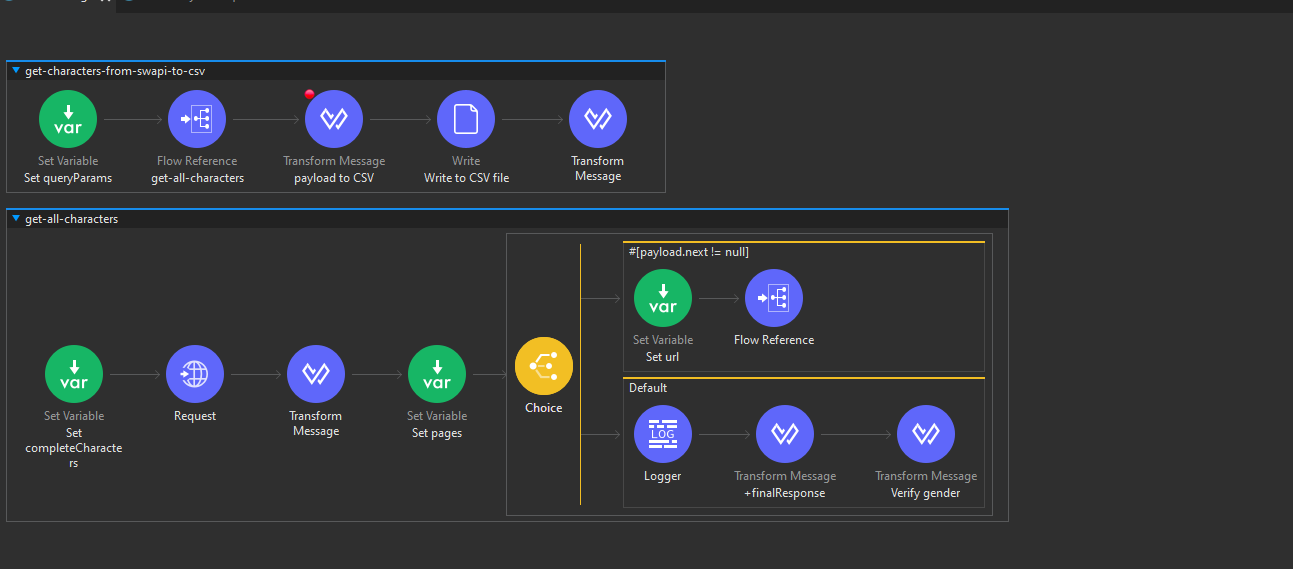
Una vez agregada la dependencia, damos clic en el botón Finish.  


Dar clic en el botón APPLY  


Dar clic en el botón Yes  


Damos clic en el botón Apply and Close  
  
  
Nos genera en automático el flujo principal con los endpoints que se definieron en la especificación.  


Aquí es donde tendremos que agregar un subflujo para la consulta de nuestros personajes de Starwars a través de la SWAPI, creando un nuevo archivo de configuración, para eso damos clic derecho en el proyecto y seleccionamos la opción New>Mule Configuration File  


Nombramos nuestro archivo y damos clic en el botón Finish  
  
  
Después creamos un “New Configuration File” al cual llamaremos global, ya estando en el archivo nos colocaremos en la pestaña “Global Elements”  
  
  
Ahí observamos que no existe ninguna configuración global.  
  
  
Ahora nos cambiamos nuevamente al archivo de configuración principal “starwars-system-api”, en la pestaña de “Global Elements”  
  
  
Para poder obtener la configuraciones, nos cambiaremos a la pestaña “Configuration XML” y seleccionamos dichas configuraciones copiandolas y eliminandolas del archivo XML  
  
  
una vez copiadas las configuraciones y eliminadas del archivo “starwars-system-api”, las pegamos en el archivo “global.xml”  
  
  
En la pestaña “Global Elements” podemos observar que ya se encuentran las configuraciones.  
  
  
Agregamos una configuración HTTP listener para el consumo de SWAPI, para eso damos clic en el botón “Create” y seleccionamos “Global Configuration>HTTP Request configuration” y damos clic en el botón “OK”  
  
  
  
Configuramos el request con los siguientes parámetros y damos clic en el botón “OK”  
  
  
Podemos observar que la configuración fue creada exitosamente.  
  
  
  
Regresamos al archivo de configuración donde estamos realizando el flujo “characters-get” y realizamos el flujo para recorrer todas las páginas de SWAPI para obtener todos los personajes.  


## get-characters-from-swapi-to-csv

* Colocamos un “Set Variable” al que nombramos “Set queryParams” donde obtenemos los queryParams de la petición y lo almacenamos en vars.queryParams.
* Colocamos un “Flow Reference” que haga referencia al flujo “get-all-characters”
* Colocamos un “Transform Message” que transforma el payload a formato CSV
* Colocamos un Write, que nos genera un archivo .csv con la información del payload.
* Colocamos un “Transform Message” que vuelve a convertir el payload a formato JSON para dar la respuesta al cliente.

## get-all-characters

* Colocamos un “Set Variable” donde generamos un arreglo vacío en caso que no exista la variable vars.completeCharacters.
* Después colocamos un HTTP Request, y lo asociamos a la configuración previamente realizada pero en en la propiedad URL le asignamos la url inicial en caso de que no exista la variable vars.url
* Colocamos un “Transform Message” en donde concatenamos a la variable vars.completeCharacters los elementos del arreglo que nos responde el “HTTP Request” en el objeto results
* Colocamos un “Set Variable” donde iniciamos la variable var.pages en 1 en caso de que no exista o la incrementa en 1 en caso de que si.
* colocamos un “Choice”
  + En el when colocamos la condición payload.next != null para ver si se tiene información del next significa que hay mas paginas por recorrer.
  + Colocamos un “Set Variable” donde guardamos en vars.url el valor del payload.next
  + Colocamos un “Flow Reference para volver a consumirse el mismo flujo “get-all-characters”
  + En el default un logger para imprimir el valor de vars.pages y vars.completeCharacters
  + Colocamos un “Transform Message” donde se realiza el mapeo para dejar un arreglo limpio con los campos solicitados “name, height, mass, hair\_color, skin\_color, eye\_color, birth\_year, gender”
  + Colocamos un Transform Message” donde filtramos por la variable vars.gender en caso de que exista la variable vars.queryParams.

## Conclusión

* Se utilizó recursividad para poder recorrer todas las páginas de SWAPI controlando las variables para cuando fuera la primera iteración y la última.