



Mulesoft Jr / Mid Developer

Technical practice

Jesús Sánchez Nájera

Content

Content	1
API specification design	2
API specification in RAML format	6
Create a mule project	7
Test API on localhost	24
Deploy to CloudHub (Anypoint platform)	28

API specification design

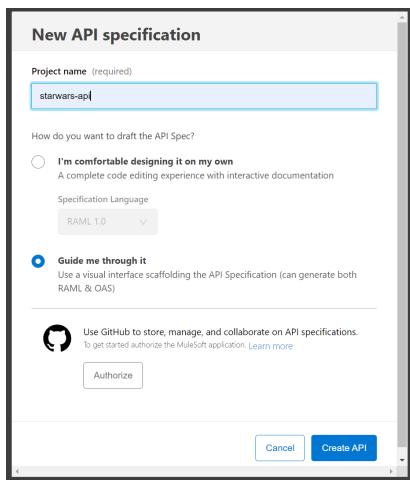
The first step is to create an account in Anypoint Platform.

Once the account is created and we are on the main screen, we will look for the option "DESIGN CENTER" in the left side menu.

On the next page go to “Create +”

Click on “New API Specification”

Create API with the next values



The next step is start adding the properties of our api

```

{
  "title": "Star Wars Characters",
  "version": "1.0",
  "protocols": [
    "HTTP",
    "HTTPS"
  ],
  "mediaType": "application/json",
  "baseUri": "https://swapi.dev/api/"
}
  
```

Then we will aggregate a Data Types

In this case we will add the values of the response that we expect in the api based on the response that the SWAPI offers us

The screenshot shows the MuleSoft Anypoint Studio Design Center interface. The top part displays the RAML code for the /people endpoint:

```

{
  "count": 82,
  "next": "https://swapi.dev/api/people/?page=2",
  "previous": null,
  "results": [
    {
      "name": "Luke Skywalker",
      "height": "172",
      "mass": "77",
      "hair_color": "blond",
      "skin_color": "fair",
      "eye_color": "blue",
      "birth_year": "19BBY",
      "gender": "male",
      "homeworld": "https://swapi.dev/api/planets/1/",
      "films": [
        "https://swapi.dev/api/films/1/",
        "https://swapi.dev/api/films/2/"
      ]
    }
  ]
}
  
```

The bottom part shows the 'starwars-api' Data Types interface. A 'People' data type is being defined with the following properties:

- Property Name: homeworld, Type: String
- Property Name: films, Type: Array
- Property Name: species, Type: Array
- Property Name: vehicles, Type: Array
- Property Name: starships, Type: Array
- Property Name: created, Type: String
- Property Name: edited, Type: String

The right side of the interface shows the generated RAML code for the 'People' data type.

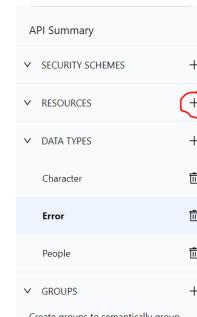
And aggregate a datatype from the possible error responses

The screenshot shows the MuleSoft Anypoint Studio Design Center interface. A new 'Error' data type is being defined, which inherits properties from the 'People' data type:

- Property Name: code, Type: String
- Property Name: message, Type: String

The right side of the interface shows the generated RAML code for the 'Error' data type, which includes the properties inherited from 'People': count, next, previous, and result.

Once we finish adding the datatypes and properties we will start with the resources



In this case, we will add the code OK-200 and assign the datatype "people" as the expected response.

We continue adding the rest of the possible error response codes

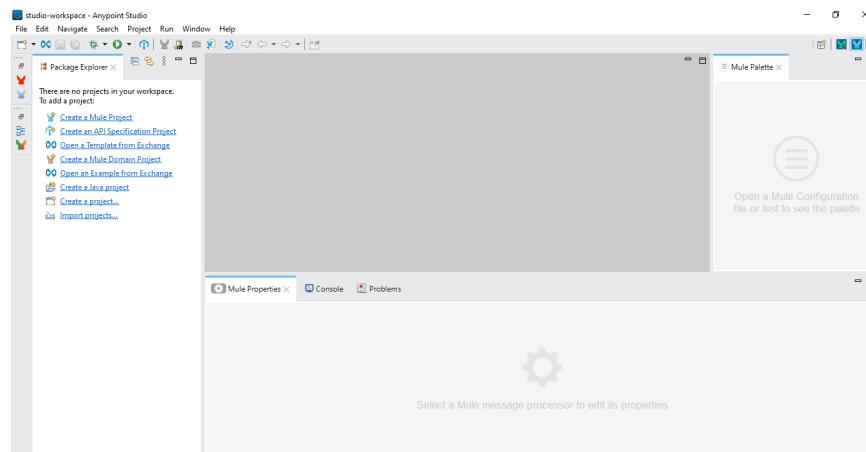
Done!, we already have our api specification in RAML format.

API specification in RAML format

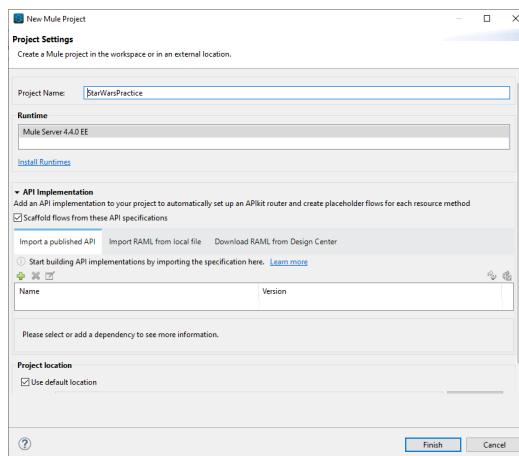
```
%%RAML 1.0
title: Star Wars Characters
baseUri: https://swapi.dev/api/
mediaType:
  - application/json
version: "1.0"
protocols:
  - HTTP
  - HTTPS
types:
  Character:
    properties:
      name:
        type: string
      height:
        type: string
      mass:
        type: string
      hair_color:
        type: string
      skin_color:
        type: string
      eye_color:
        type: string
      birth_year:
        type: string
      gender:
        type: string
      homeworld:
        type: string
    films:
      items:
        type: string
    species?:
      items:
        type: string
    vehicles?:
      items:
        type: string
    starships?:
      items:
        type: string
    created:
      type: string
    edited:
      type: string
    url:
      type: string
  People:
    properties:
      count:
        type: number
        format: int
      next?:
        type: string
      previous?:
        type: string
      result:
        items:
          type: Character
  Error:
    properties:
      code:
        type: string
      message:
        type: string
/people:
  get:
    responses:
      "200":
        body:
          application/json:
            type: People
      "500":
        body:
          application/json:
            type: Error
      "400":
        body:
          application/json:
            type: Error
      "401":
        body:
          application/json:
            type: Error
      "403":
        body:
          application/json:
            type: Error
      "404":
        body:
          application/json:
            type: Error
```

Create a Mule Project

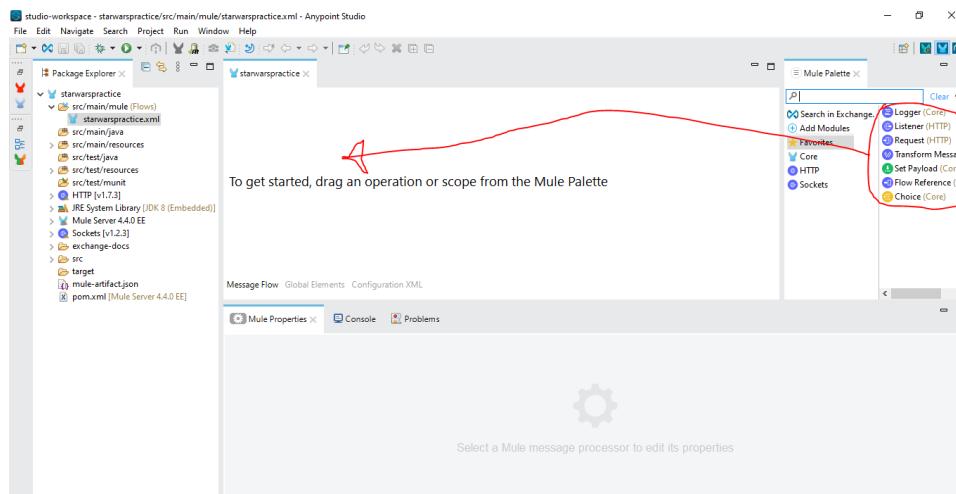
After download and install anypoint studio we are going to create a new mule project



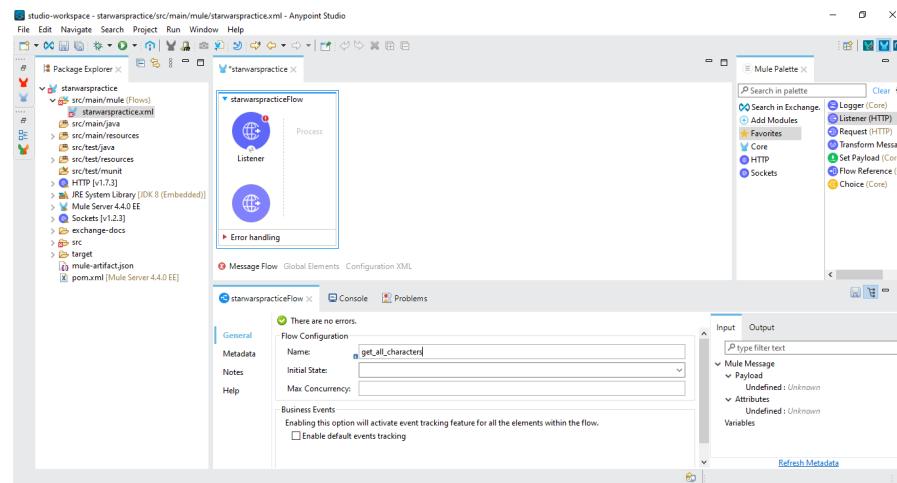
We can name the project whatever we want



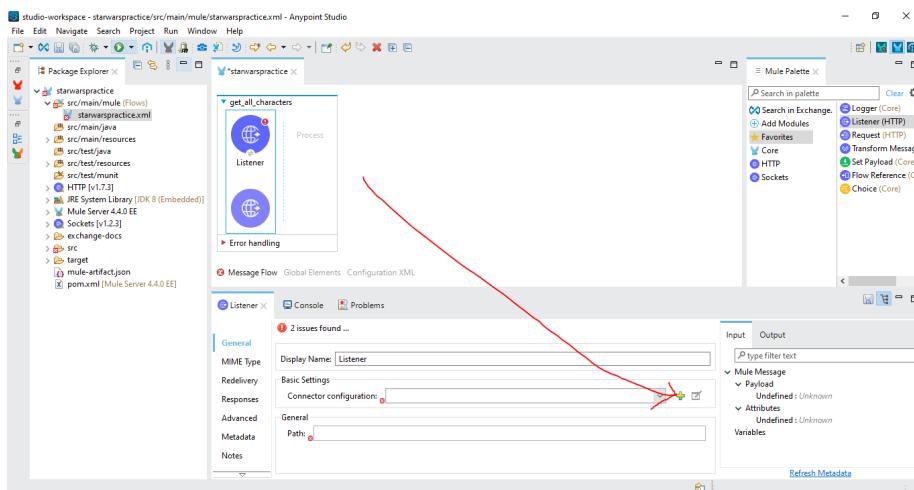
The next step is to add the components by dragging and dropping in the workspace according to what we need



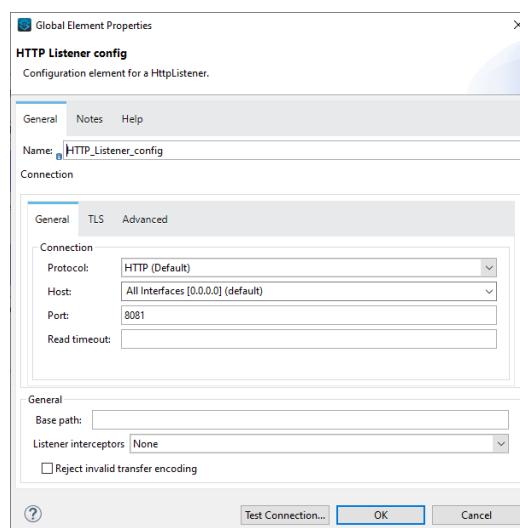
The first component we will use is a "listener".



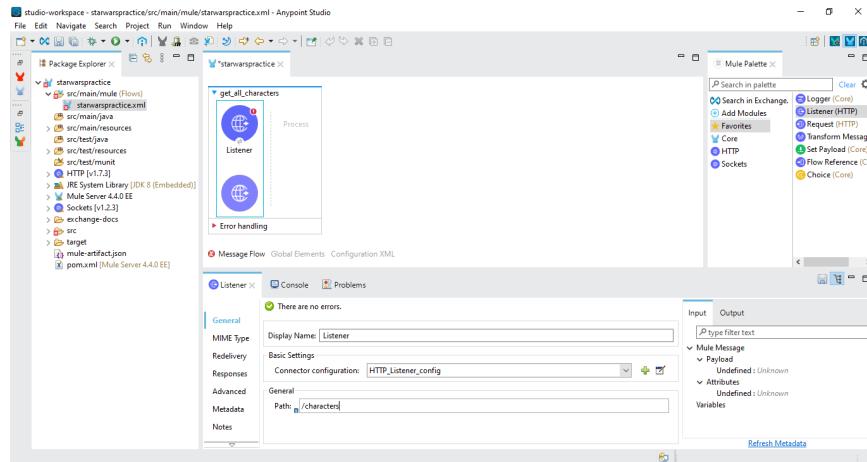
Click on “+” button.



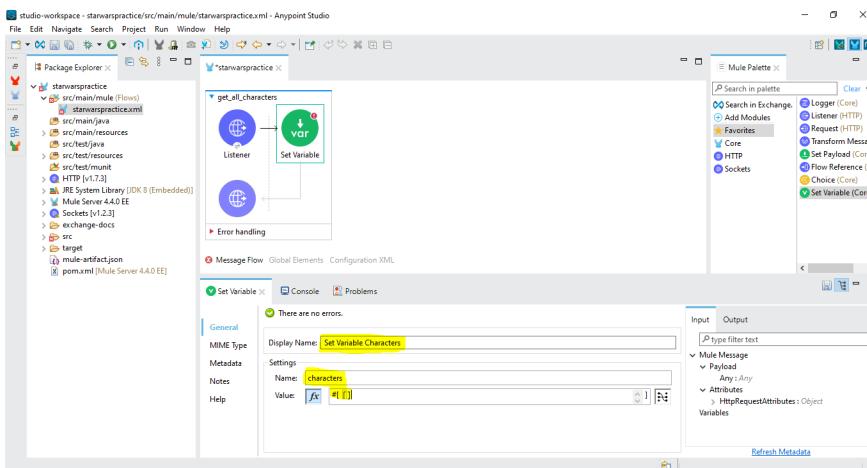
Click the ok button with the default properties.



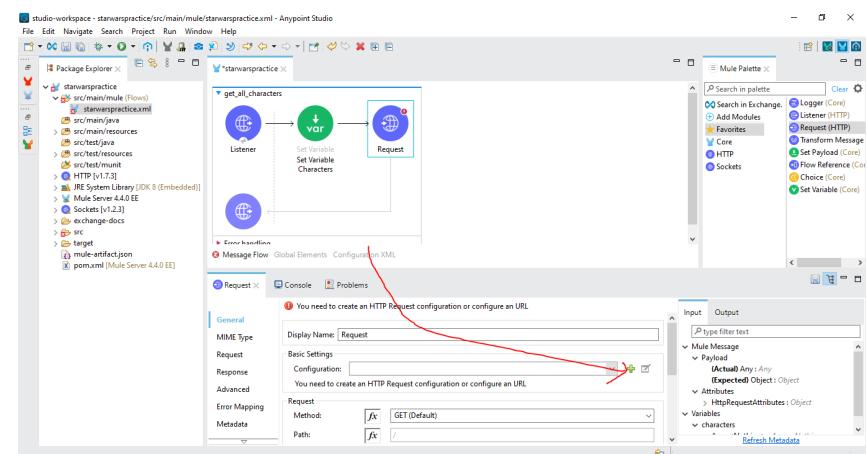
Agregamos el Path "/characters".



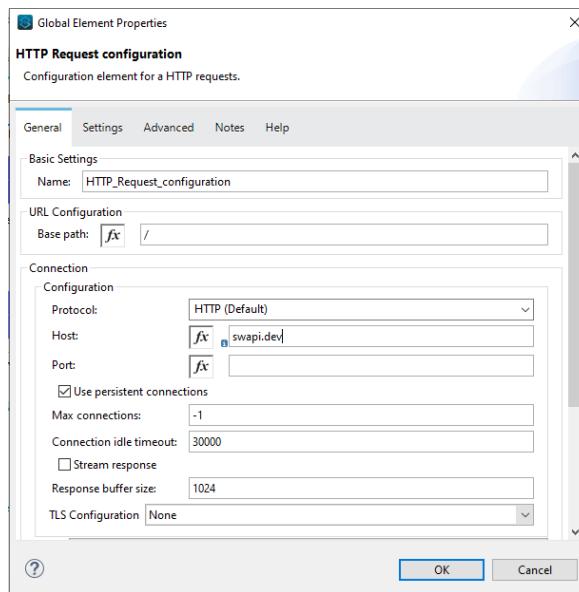
The next component we will use is "set variable" and we will initialize a variable of type array called characters.



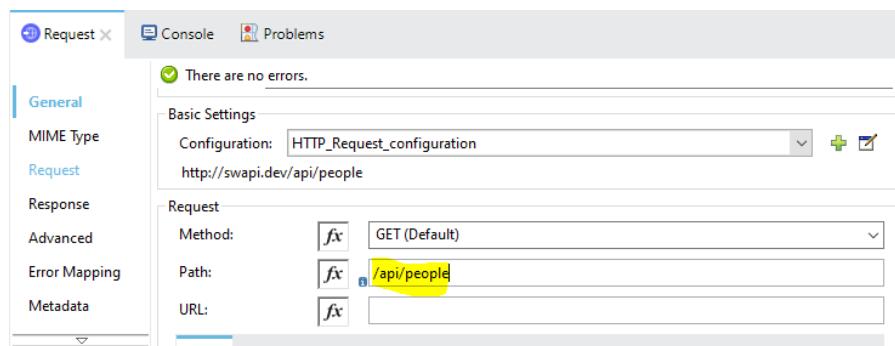
The next component we will use is "request". We will click on the "+" button to add the properties.



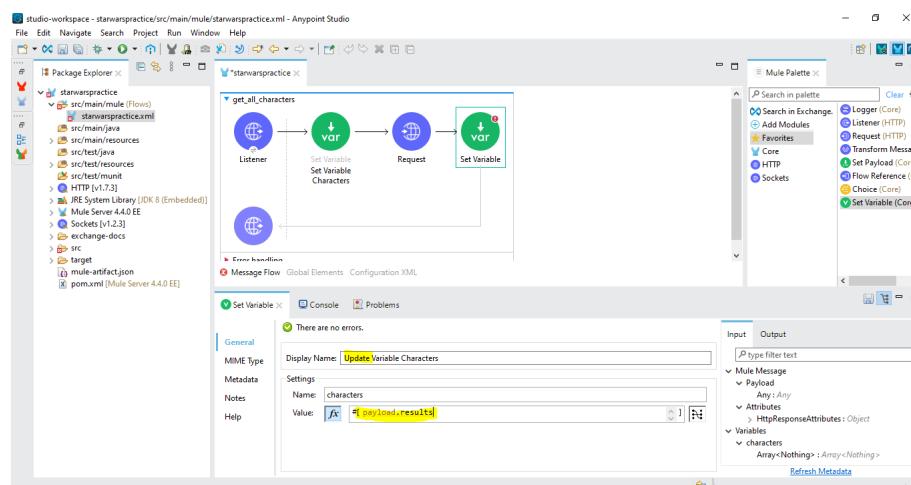
We will add "swapi.dev" in the "Host" property.



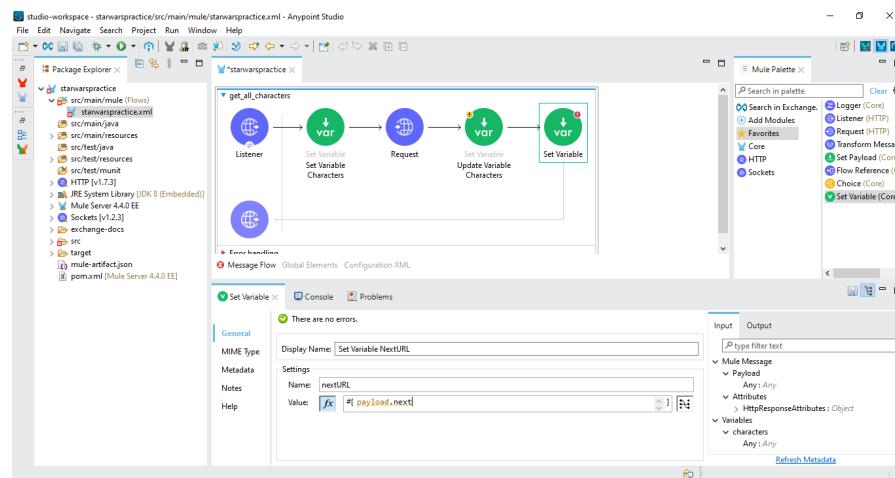
In the request properties we will add "/api/people/" in the "Path" property.



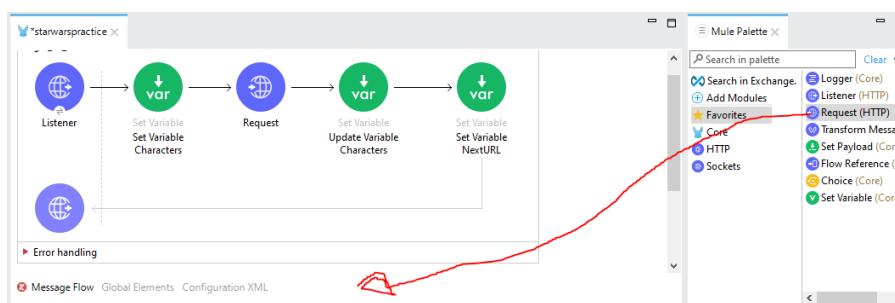
The next component is a "set variable" to assign the characters obtained from the previous request in the "characters" variable.



The next component is a "set variable" to assign the nextUrl obtained from the previous request, this url will give us the information of the characters of the next page of the SWAPI.

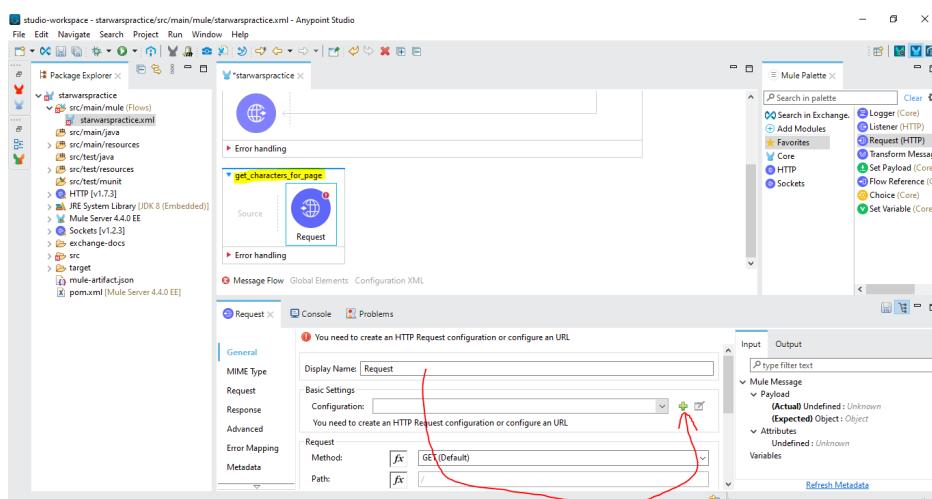


The next step is to drag and drop a "request" component outside of the flow we're working on to create another flow that will serve as a recursive function.

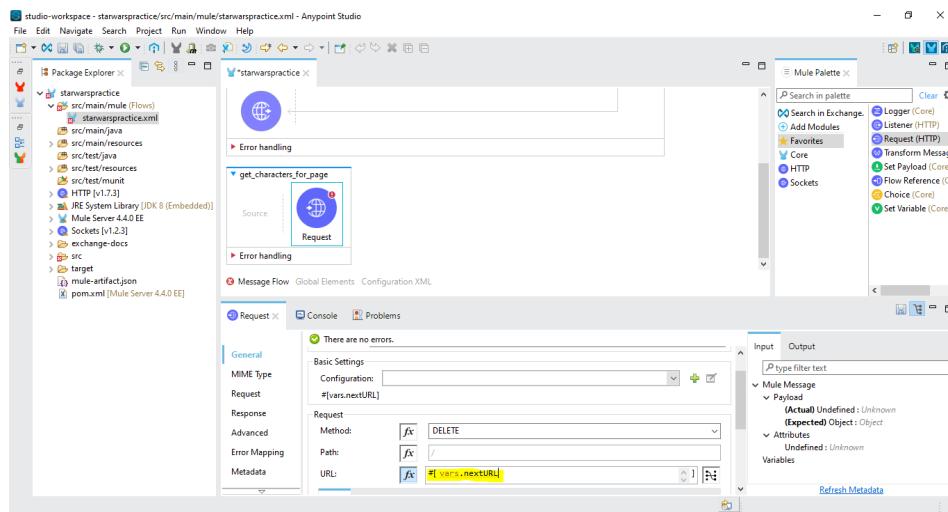


This recursive function will help us to navigate through the SWAPI character pages.

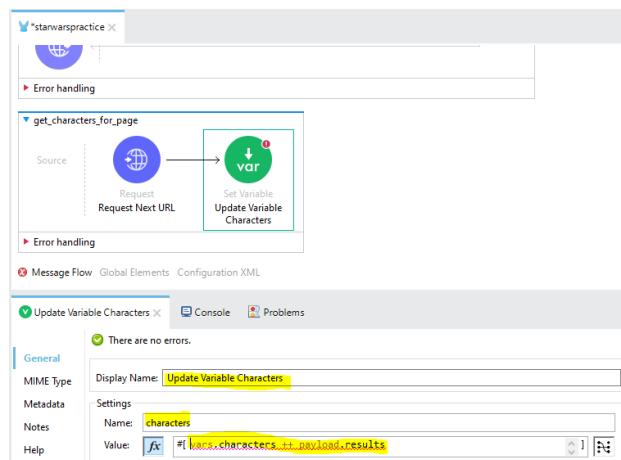
In the properties of the "request" component we will click on the "+" button.



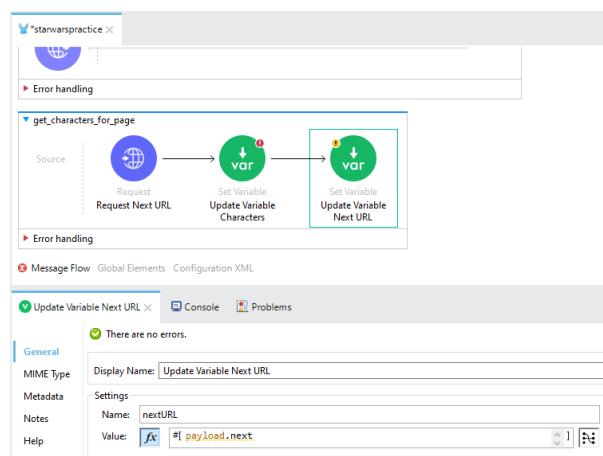
In the URL property we will assign the value of the "nextURL" variable.



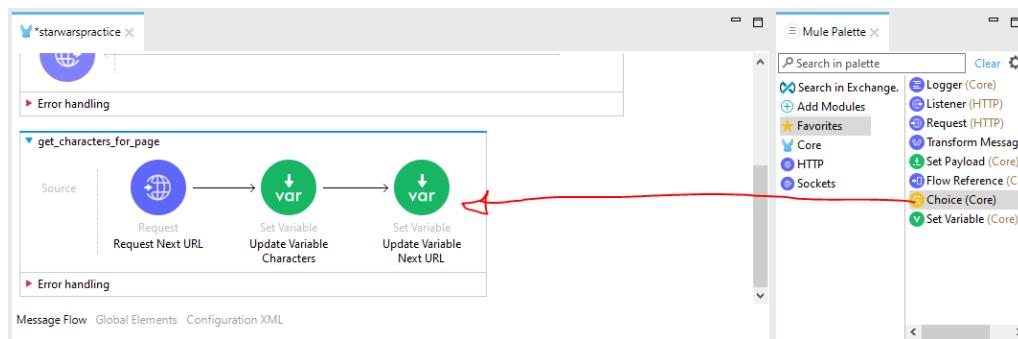
We add a "Set Variable" component to add the information obtained from this new request to the "characters" variable.



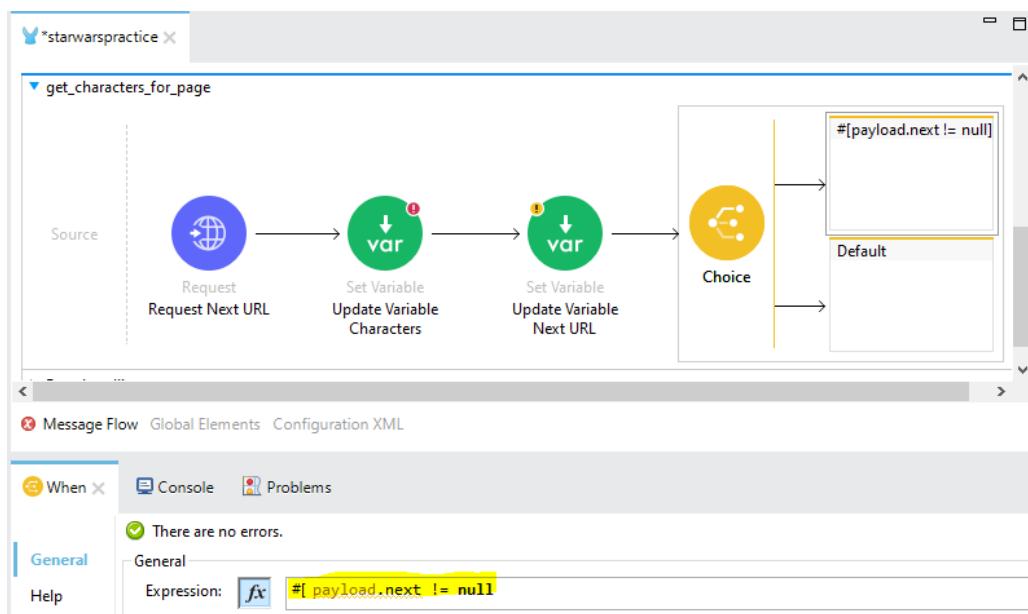
We add another "Set Variable" component to add to the "nextURL" variable the url of the next page of the SWAPI characters obtained from the new request.



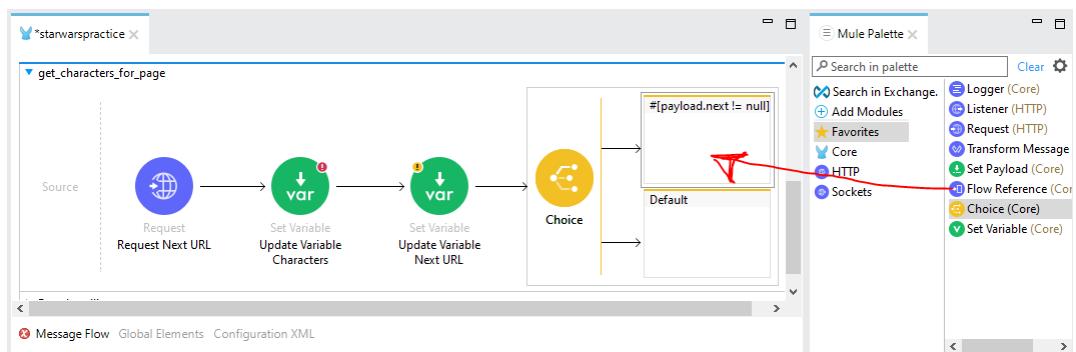
The next component we will add is "Choice".



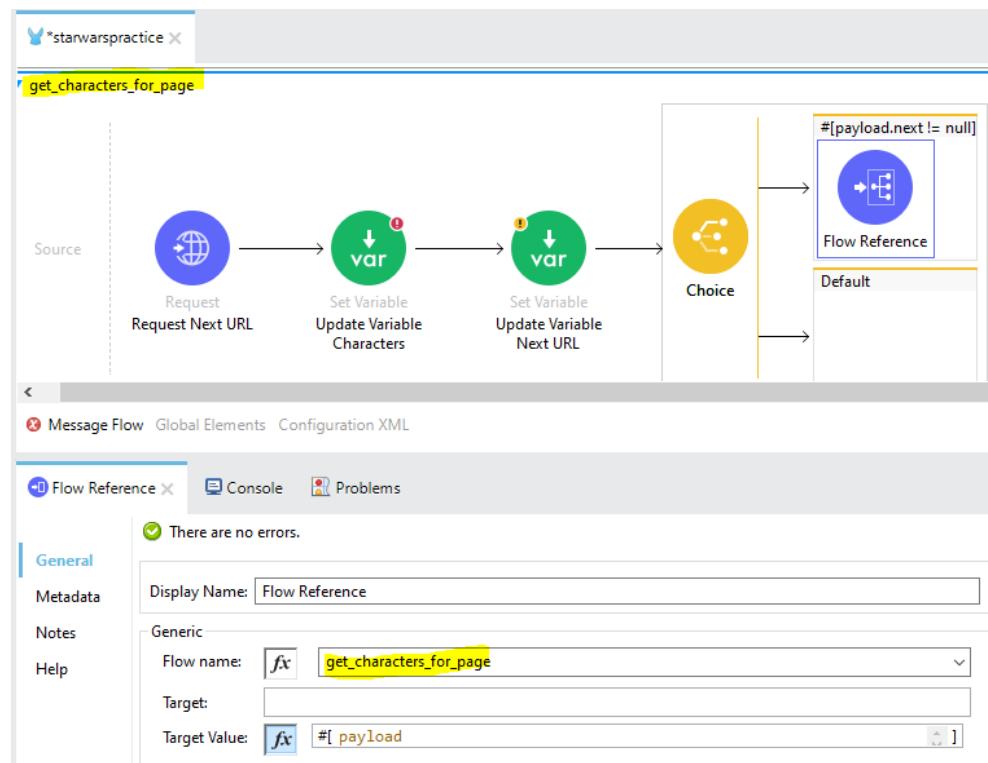
This component will make a validation. If the "nextURL" value of the previous request exists, the flow will continue.



In the case of continuing we will add the "Flow Reference" component.

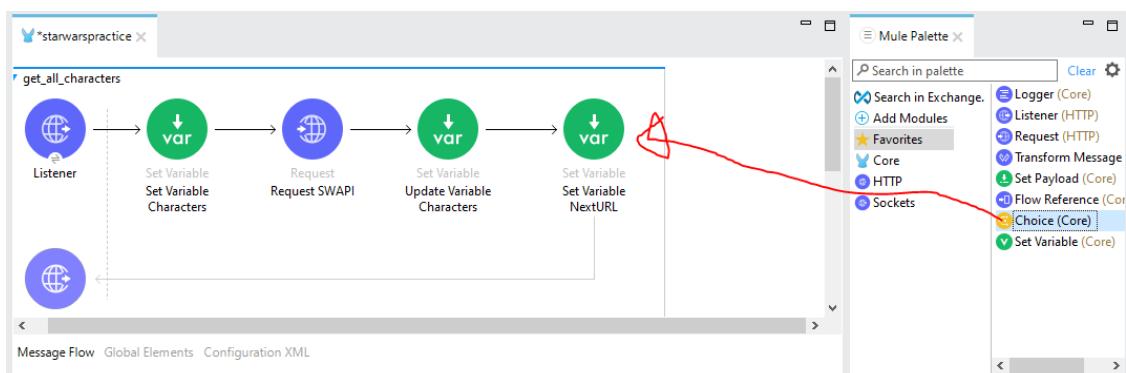


In the properties of this component we are going to assign the same flow as the objective to apply the recursion.

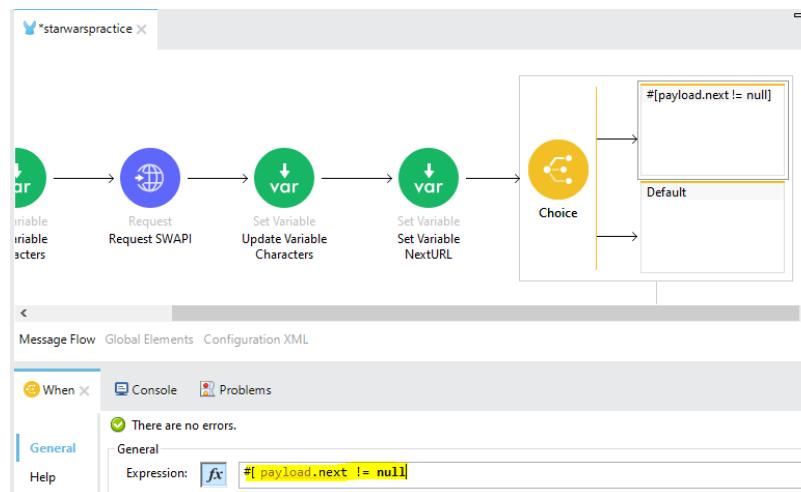


At this point it's important to understand the recursion of the flow, it will consume itself by changing the url of the request to the next one until the nextURL no longer exists.

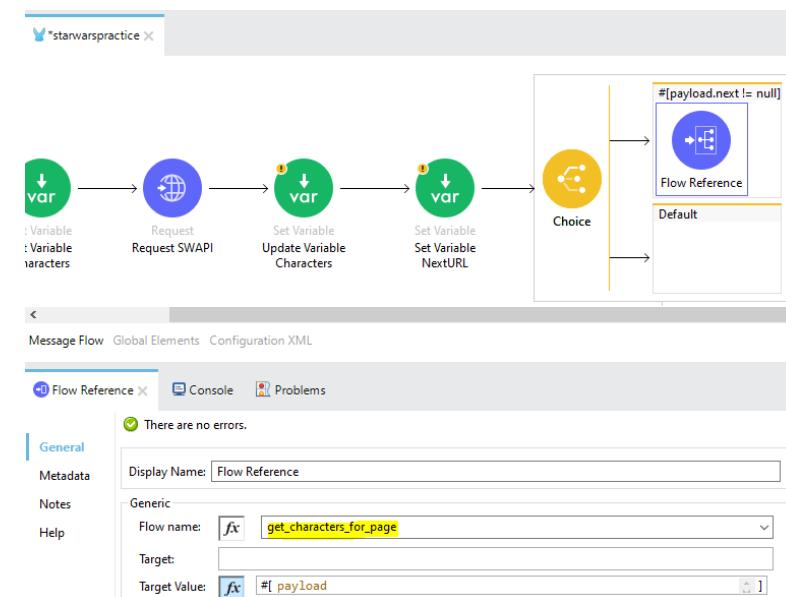
Going back to the first flow we are going to add a "choice" component.



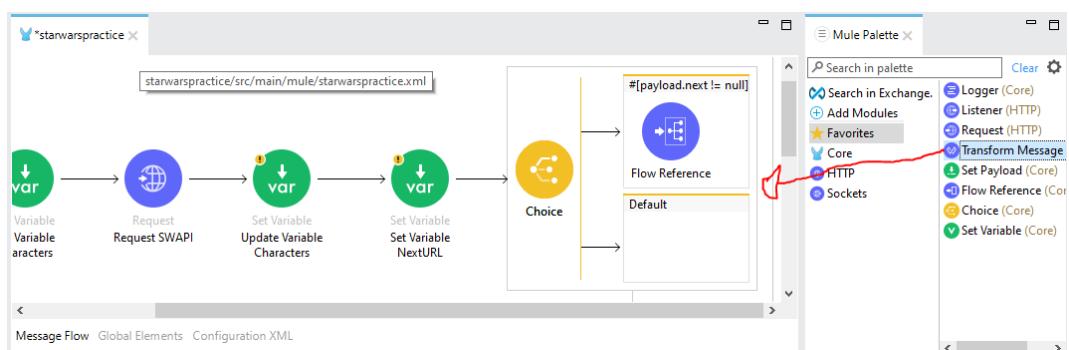
Once again, the choice component will perform a validation in which it will continue with the flow as long as the "nextURL" variable of the previous request is not null.



Here we will use a "*Flow Reference*" component to invoke our second created flow which we assimilate as a recursive function.

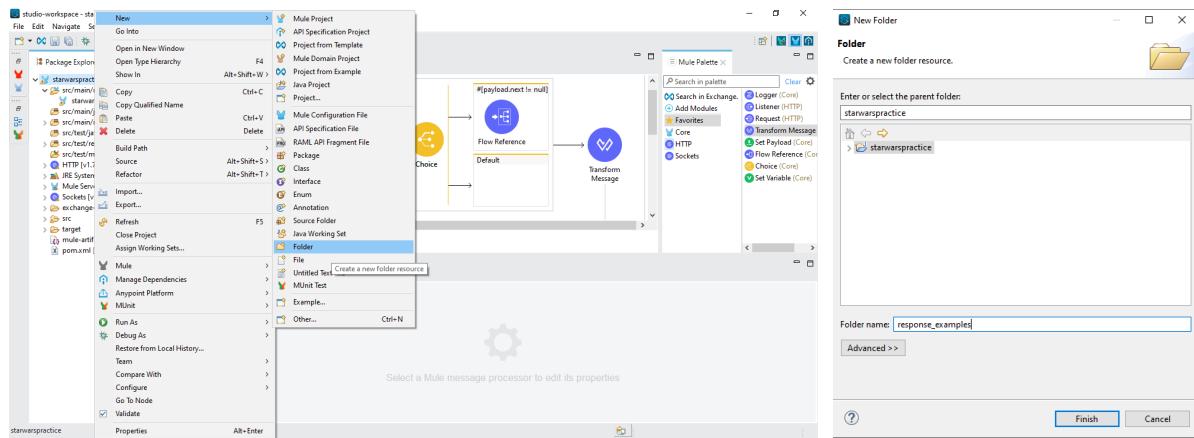


Finally we will use a "*Transform Message*" component.

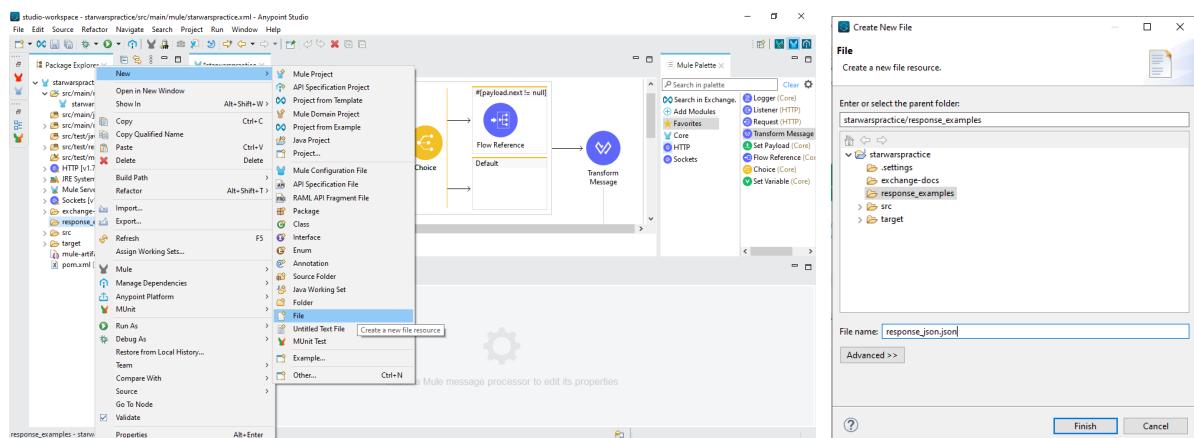


In the "*Transform message*" component we can format the response that our project will give. To give the response in the requested format it is necessary to import two files as an example. The first is a .json file which is the format of the response provided by SWAPI. The second is a .csv file that is the format of the response that our project will give.

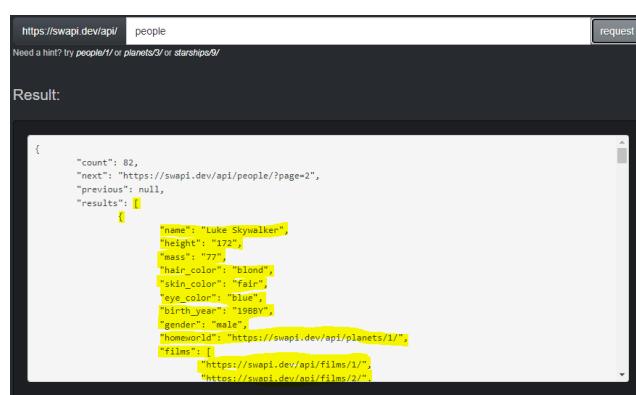
We will create a new folder called "*response_examples*".



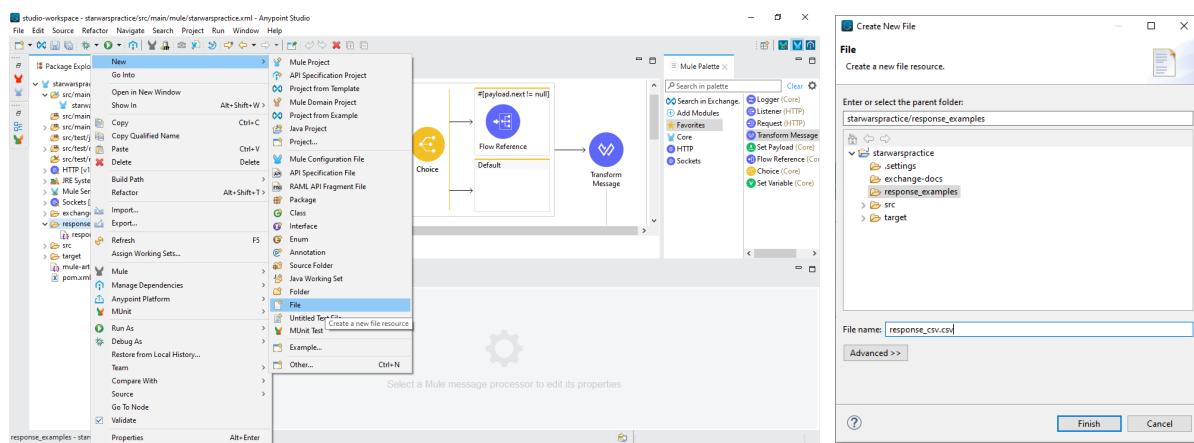
Inside the folder we will create a file called "*response_json.json*".



This file will contain the json that SWAPI gives us.



In the same folder we will create a file called “`response_csv.csv`”.



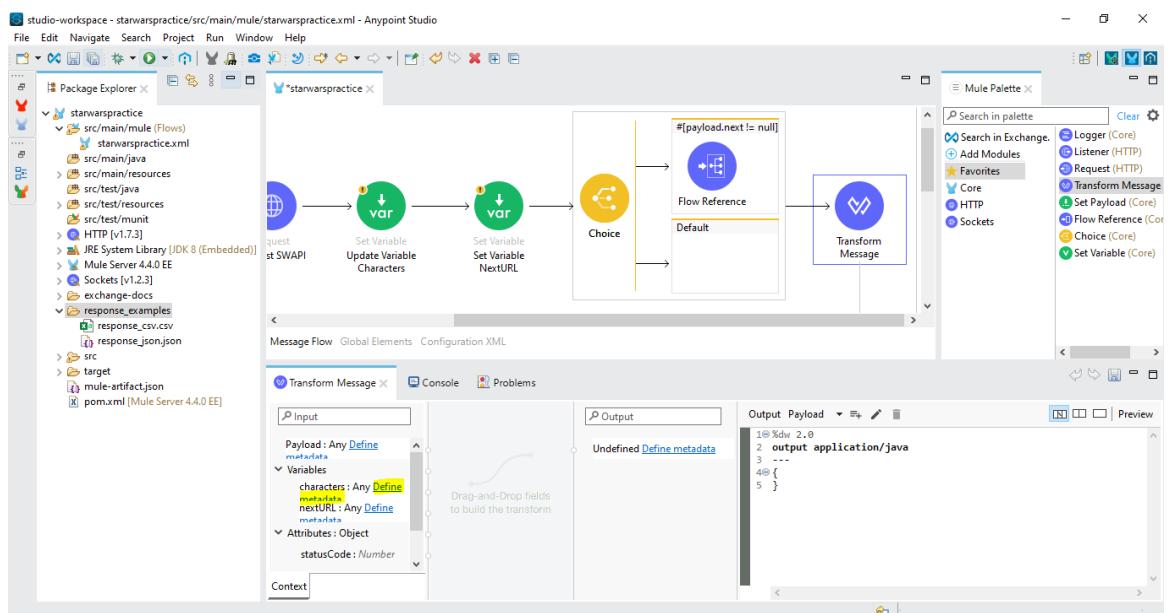
This file will contain the required format.

***response_csv: Bloc de notas**

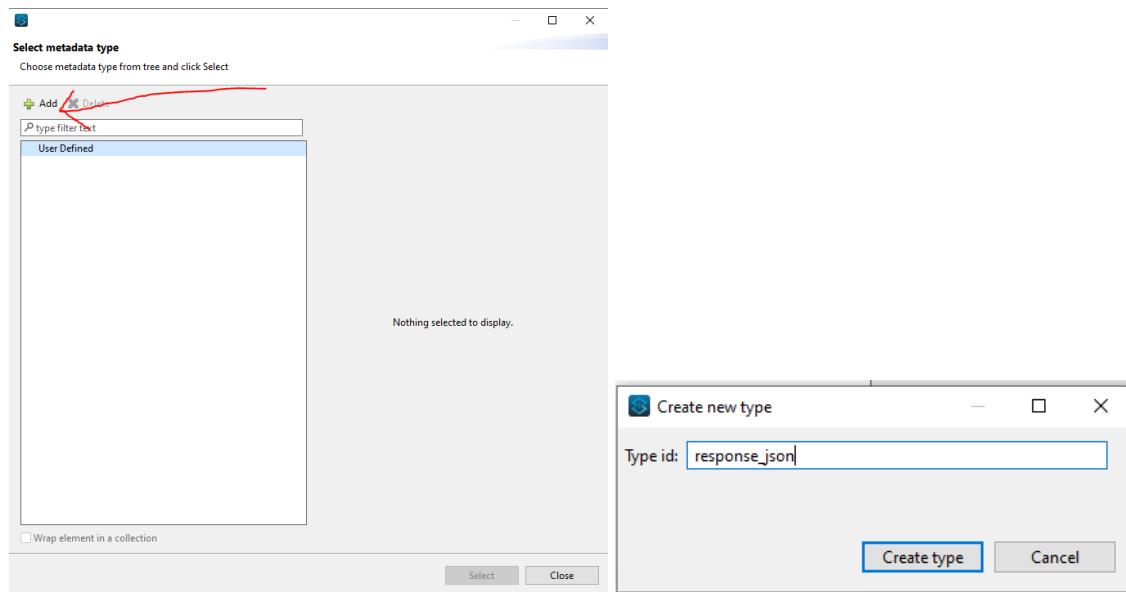
Archivo Edición Formato Ver Ayuda

```
name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
Luke Skywalker,172,77,blond,fair,blue,19BBY,male
C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
```

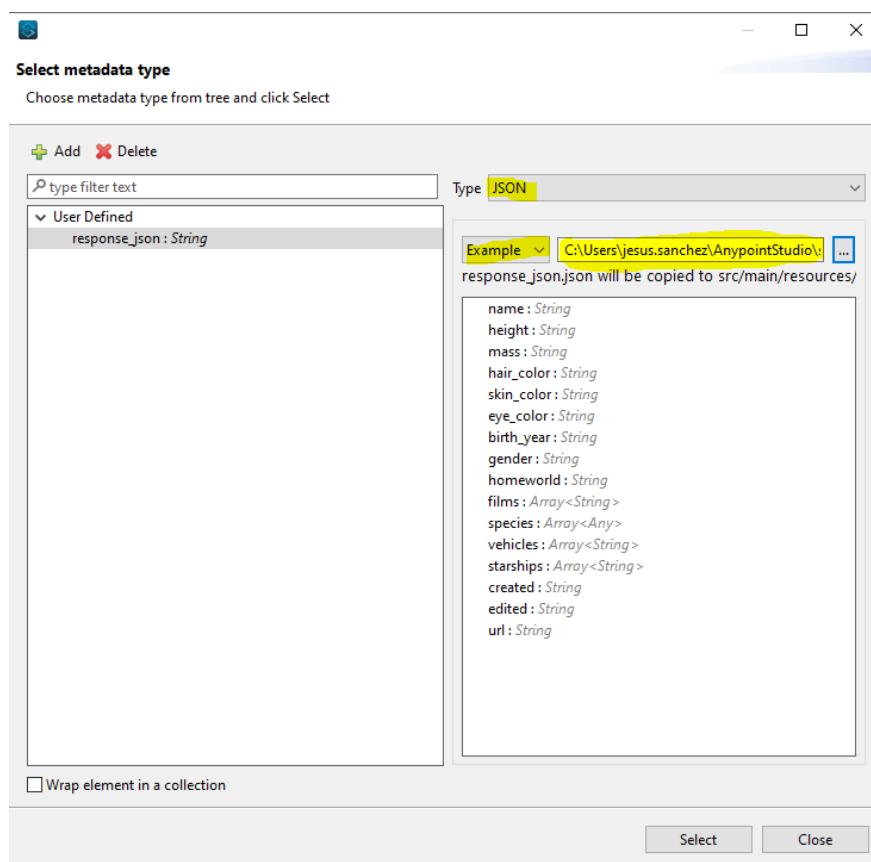
The next step is to map the response from the “*Transform Message*” component. We define our “*Characters*” variable with the json format that we added as an example. Click on “*Define metadata*” on variable “*Characters*”



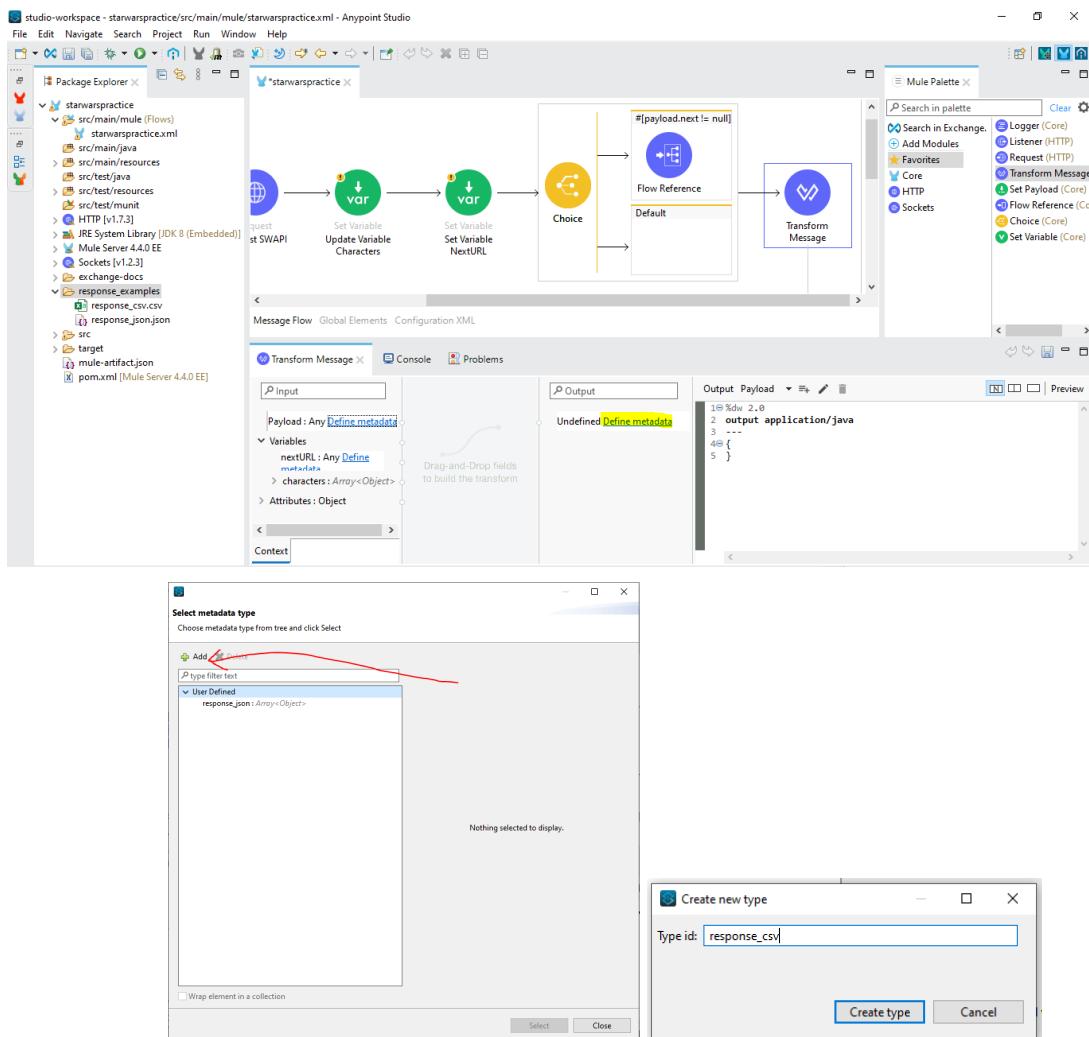
Follow the next steps.



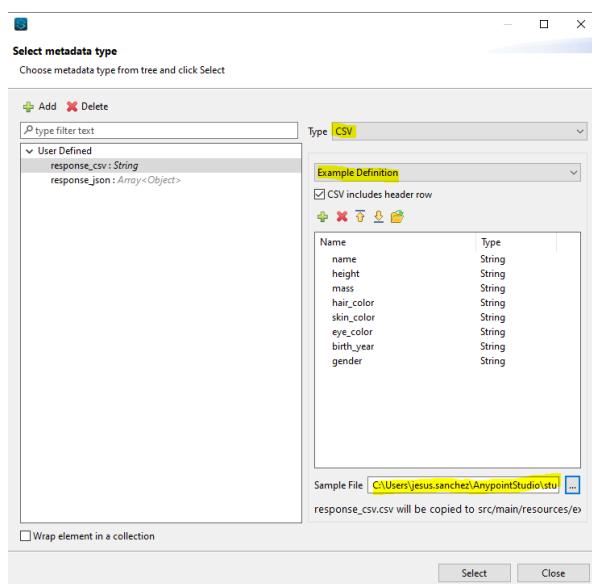
We choose the following options.



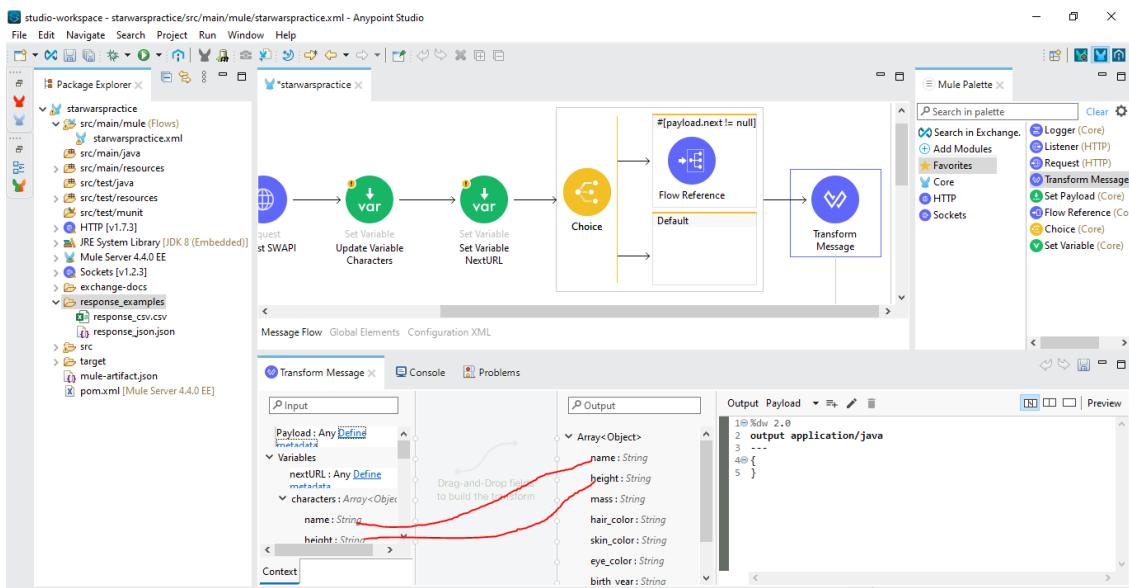
We apply the same steps for output but with the csv format



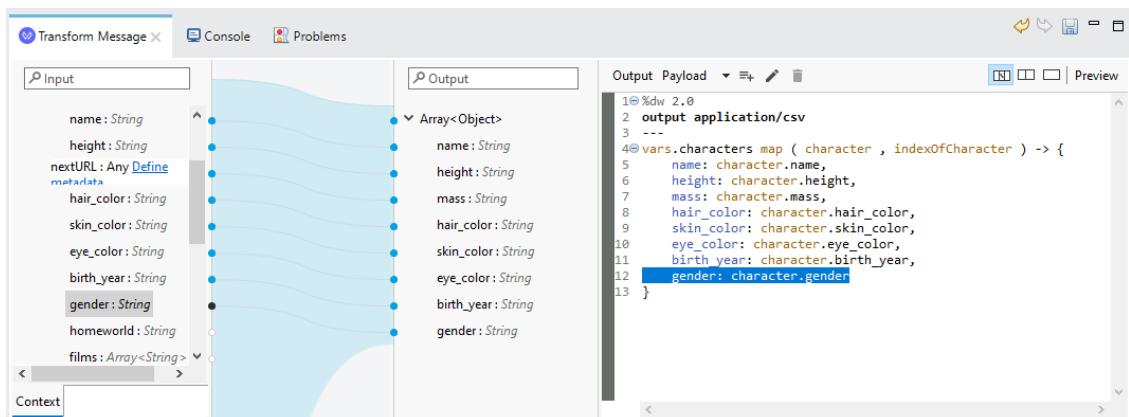
We choose the following options.



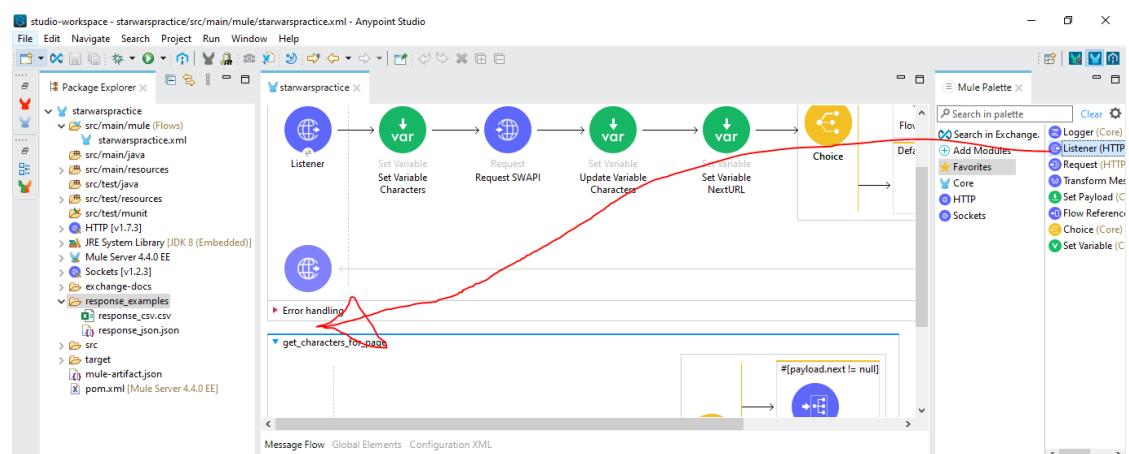
The next step is to map the variables, from the "Characters" variable to the output object.



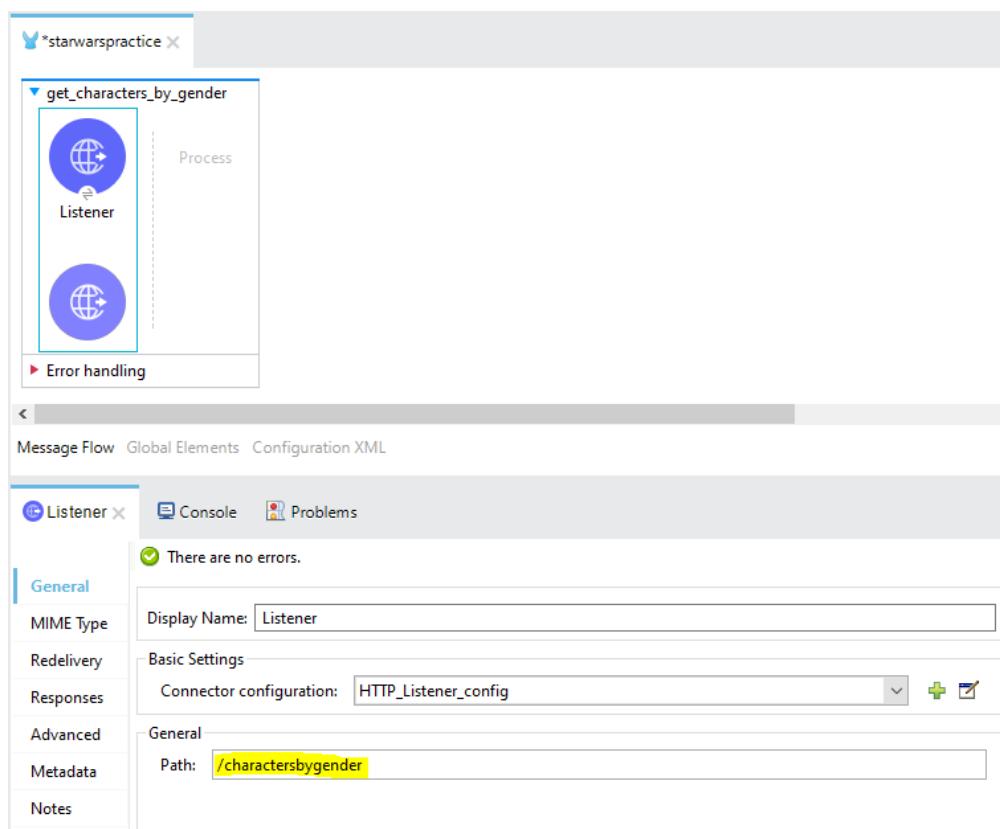
At the end of the mapping we will obtain the following code.



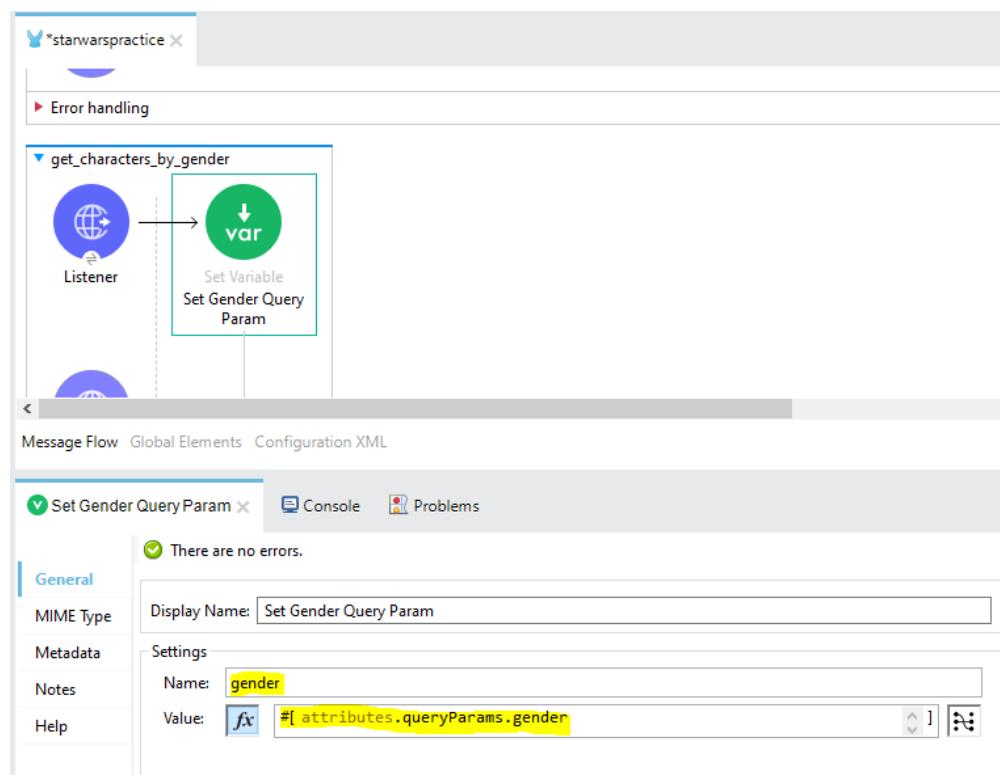
To add the functionality of filtering the response through a QueryParameter we need to add another flow starting with a "Listener" component.



In the Path property we add “/charactersbygender”.



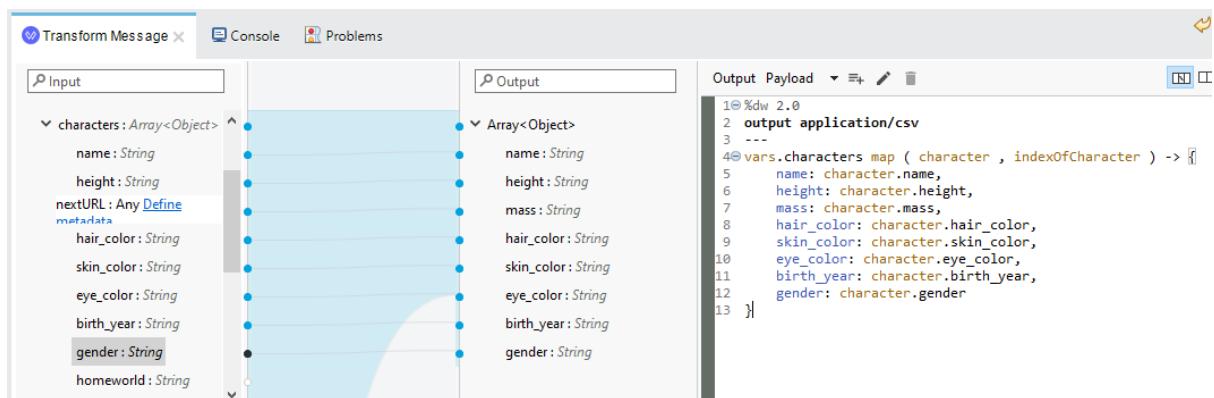
The next component is a "Set Variable" in which we are going to assign the value of the Query Parameter we receive.



The following components that are added are the same as the first flow with the same properties.



The only thing that will change is the mapping in the "Transform message" component.



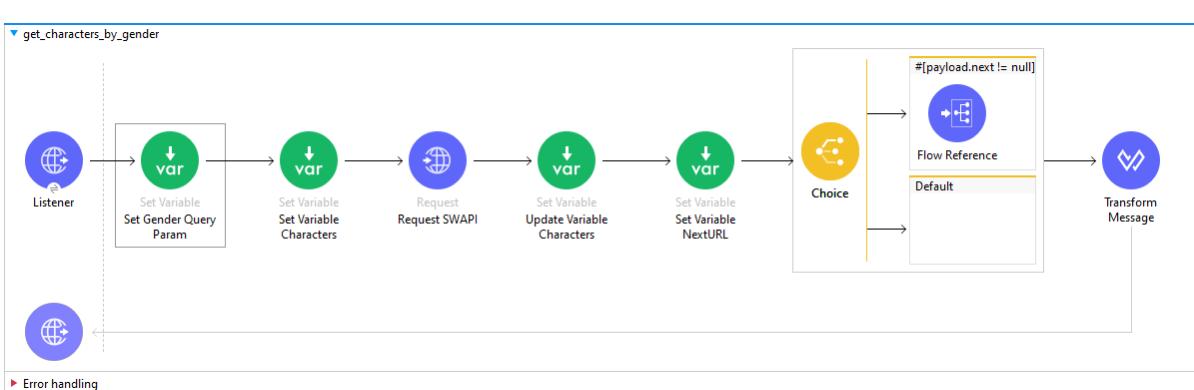
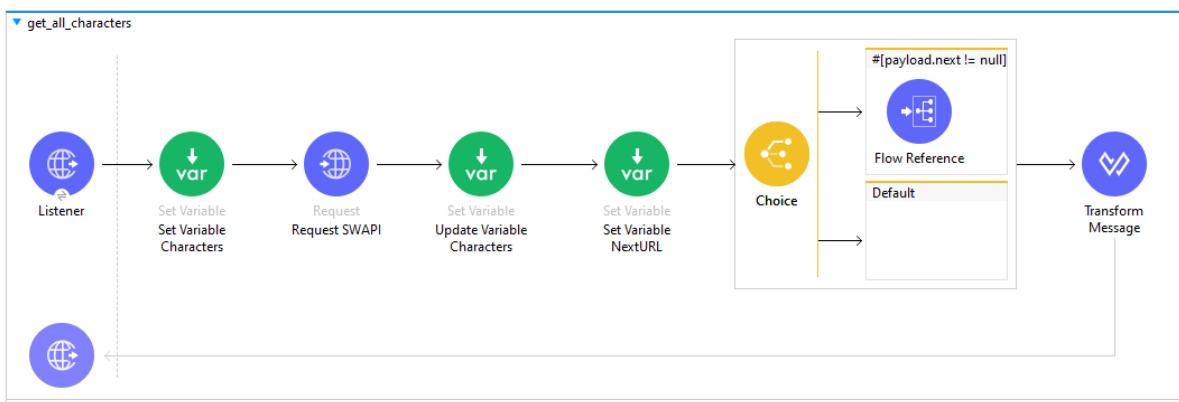
In the code obtained after mapping we are going to add the following filter.

```

1@%dw 2.0
2  output application/csv
3  ---
4@vars.characters filter ($.gender == vars.gender) map ( character , indexOfCharacter ) -> {
5    name: character.name,
6    height: character.height,
7    mass: character.mass,
8    hair_color: character.hair_color,
9    skin_color: character.skin_color,
10   eye_color: character.eye_color,
11   birth_year: character.birth_year,
12   gender: character.gender
13 }
```

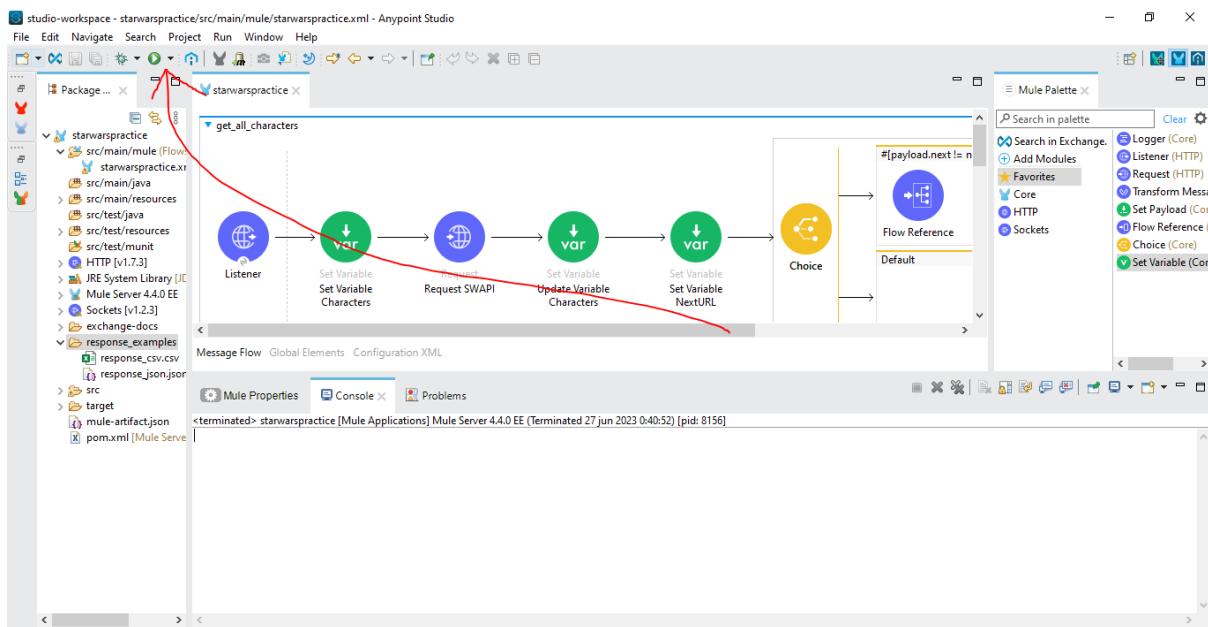
This filter is used to only give us as a response the elements of the csv that are equal to the Query Parameter that they sent us from the beginning.

Next we have a comparison of the two main flows.

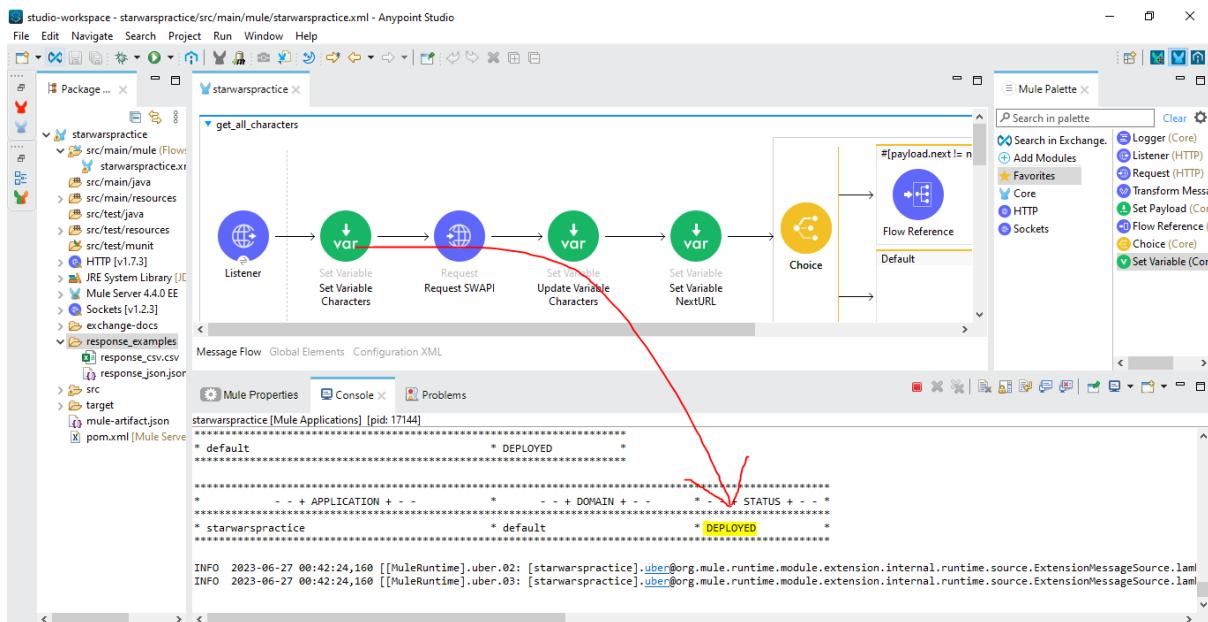


Test API on localhost

When we finish all the development of our project we will click on the "Run" button.



When we see the message "**DEPLOYED**" in the console, we will know that we can now make a request to our API



We can do the test in any REST Client, in this case we will use ARC (Advanced REST Client).

The endpoint that we must use to obtain all the characters is <http://localhost:8081/characters>

The screenshot shows the ARC interface with a 'GET' request to 'http://localhost:8081/characters'. The response section displays the message 'No response recorded'.

Response.

The screenshot shows the ARC interface displaying the response to the GET request. The status is 200 OK, and the response body is a list of Star Wars characters:

```
1 name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 Luke Skywalker,172,77,blond,fair,blue,19BBY,male
3 C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
4 R2-D2,96,32,n/a,white\, blue,red,33BBY,n/a
5 Darth Vader,202,136,none,white,yellow,41.9BBY,male
6 Leia Organa,150,49,brown,light,brown,19BBY,female
7 Owen Lars,178,120,brown\, grey,light,blue,52BBY,male
8 Beru Whitesun Lars,165,75,brown,light,blue,47BBY,female
9 R5-D4,97,32,n/a,white\, red,red,unknown,n/a
10 Biggs Darklighter,183,84,black,light,brown,24BBY,male
11 Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,57BBY,male
12 Anakin Skywalker,188,84,blond,fair,blue,41.9BBY,male
13 Wilehuff Tarkin,180,unknown,auburn\, grey,fair,blue,64BBY,male
14 Chewbacca,228,112,brown,unknown,blue,200BBY,male
15 Han Solo,180,80,brown,fair,brown,29BBY,male
16 Greedo,173,74,n/a,green,black,44BBY,male
```

The endpoint to get the characters filtered by gender is:
http://localhost:8081/charactersbygender?gender={gender}

gender = female

The screenshot shows the Advanced REST Client interface. In the top navigation bar, there are icons for History, API Client, and Environment (set to Default). Below the navigation is a toolbar with File, Edit, View, Window, Request, Workspace, and Help. A sidebar on the left has icons for History, Today, Favorites, Cloud, and Search, with 'TODAY' selected.

The main area shows a request configuration for a GET method to the URL <http://localhost:8081/charactersbygender?gender=female>. The request pane includes sections for HEADERS, AUTHORIZATION, ACTIONS, CONFIG, and CODE SNIPPETS. Below the request pane is a text editor labeled 'Text editor' with the placeholder 'Add a header to the HTTP request.' A 'COPY' button is available.

Under the request pane, there is a 'Response' section with a 'CLEAR' button. The status message 'No response recorded' is displayed, along with the instruction 'Send a request to see the response.'

Response:

The screenshot shows the response details. The status bar at the top indicates a 200 OK status, a response time of 17699 ms, and a size of 1007 Bytes. The response body is a JSON array of character objects, each with properties like name, height, mass, hair_color, skin_color, eye_color, birth_year, and gender. All 16 characters listed are female.

Index	Name	Height	Mass	Hair Color	Skin Color	Eye Color	Birth Year	Gender		
1	Leia Organa	150	49	brown	light	brown	19BBY	female		
2	Beru Whitesun Lars	165	75	brown	light	blue	47BBY	female		
3	Mon Mothma	150	unknown	auburn	fair	blue	48BBY	female		
4	Padmé Amidala	185	45	brown	light	brown	46BBY	female		
5	Shmi Skywalker	163	unknown	black	fair	brown	72BBY	female		
6	Ayla Secura	178	55	none	blue	hazel	49BBY	female		
7	Adi Gallia	184	50	none	dark	blue	unknown	female		
8	Cordé	157	unknown	brown	light	brown	unknown	female		
9	Luminara Unduli	170	56.2	black	yellow	blue	58BBY	female		
10	Barriiss Offee	166	50	black	yellow	blue	40BBY	female		
11	Dormé	165	unknown	brown	light	brown	unknown	female		
12	Zam Wesell	168	55	blonde	fair	green	yellow	yellow	unknown	female
13	Taun We	213	unknown	none	grey	black	unknown	female		
14	Jocasta Nu	167	unknown	white	fair	blue	unknown	female		
15	R4-P17	96	unknown	none	silver	red	red	blue	unknown	female
16										

gender = male

The screenshot shows a browser-like interface for making API requests. The URL bar contains "http://localhost:8081/charactersbygender?gender=ma...". The response tab is selected, showing a status of 200 OK. The response body is a JSON array of character objects, each with fields: name, height, mass, hair_color, skin_color, eye_color, birth_year, gender. The 'gender' field is highlighted in yellow for all 16 entries. The response body is as follows:

```

1 | name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 | Luke Skywalker,172,77,blond,fair,blue,19BBY,ma...
3 | Darth Vader,202,136,none,white,yellow,41.9BBY,ma...
4 | Owen Lars,178,120,brown, grey,light,blue,52BBY,ma...
5 | Biggs Darklighter,183,84,black,light,brown,24BBY,ma...
6 | Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,57BBY,ma...
7 | Anakin Skywalker,188,84,blond,fair,blue,41.9BBY,ma...
8 | Wilhuff Tarkin,180,unknown,auburn\, grey,fair,blue,64BBY,ma...
9 | Chewbacca,228,112,brown,unknown,blue,200BBY,ma...
10 | Han Solo,180,80,brown,fair,brown,29BBY,ma...
11 | Greedo,173,74,n/a,green,black,44BBY,ma...
12 | Wedge Antilles,170,77,brown,fair,hazel,21BBY,ma...
13 | Jek Tono Porkins,180,110,brown,fair,blue,unknown,ma...
14 | Yoda,66,17,white,green,brown,896BBY,ma...
15 | Palpatine,170,75,grey,pale,yellow,82BBY,ma...
16 | Boba Fett,183,78.2,black,fair,brown,31.5BBY,ma...

```

gender = n/a

The screenshot shows a browser-like interface for making API requests. The URL bar contains "http://localhost:8081/charactersbygender?gender=n/a". The response tab is selected, showing a status of 200 OK. The response body is a JSON array of character objects, each with fields: name, height, mass, hair_color, skin_color, eye_color, birth_year, gender. The 'gender' field is highlighted in yellow for all 4 entries. The response body is as follows:

```

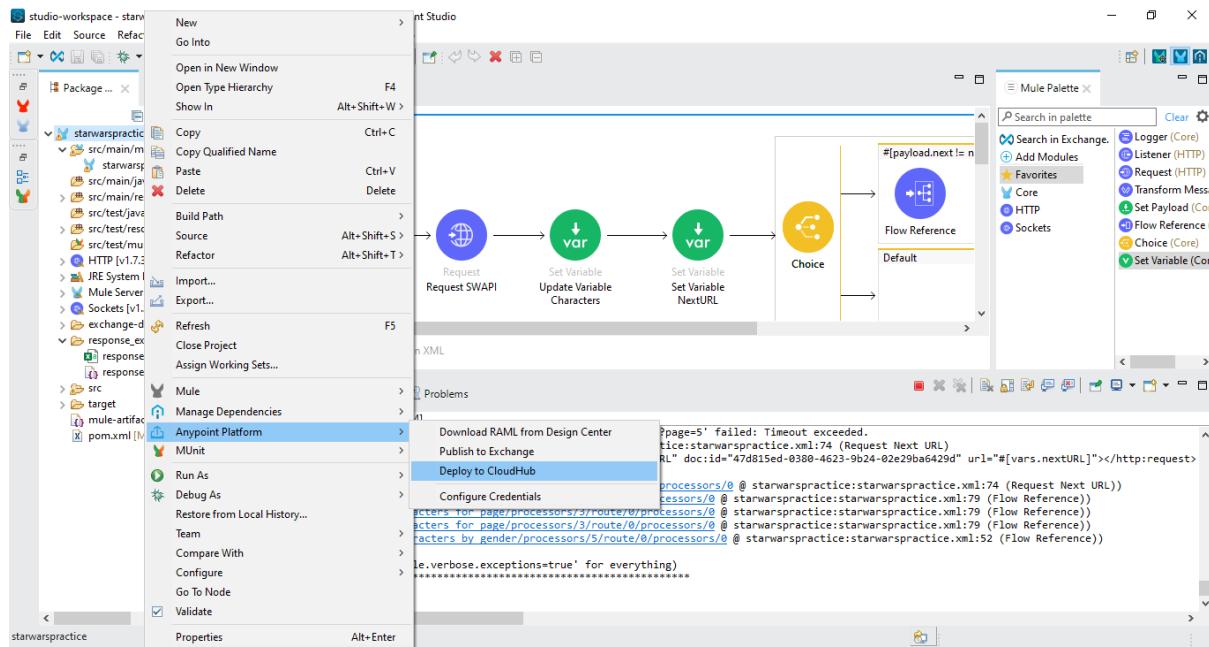
1 | name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 | C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
3 | R2-D2,96,32,n/a,white\, blue,red,33BBY,n/a
4 | R5-D4,97,32,n/a,white\, red,red,unknown,n/a

```

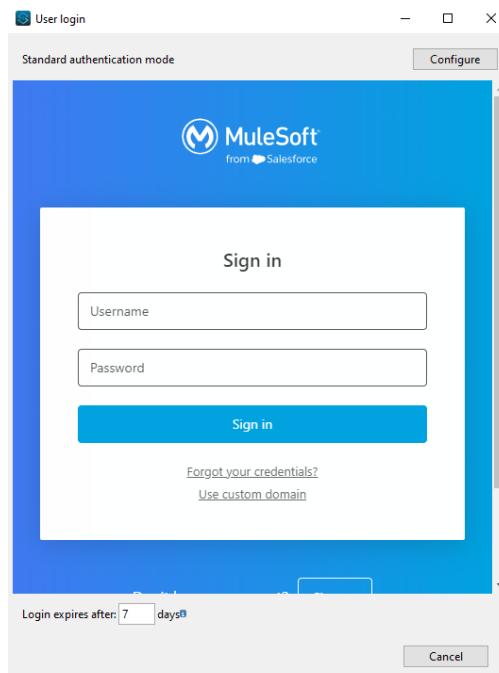
Deploy to CloudHub (Anypoint platform)

To deploy the project in CloudHub we follow the steps:

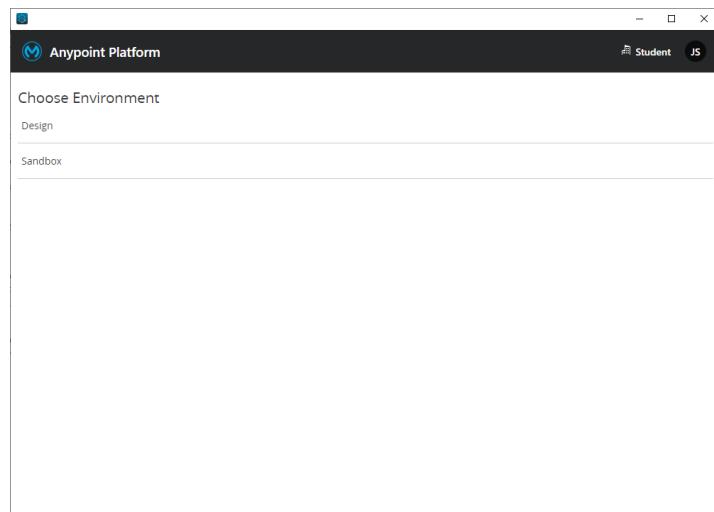
Right click on the project -> Anypoint Platform -> Deploy to CloudHub.



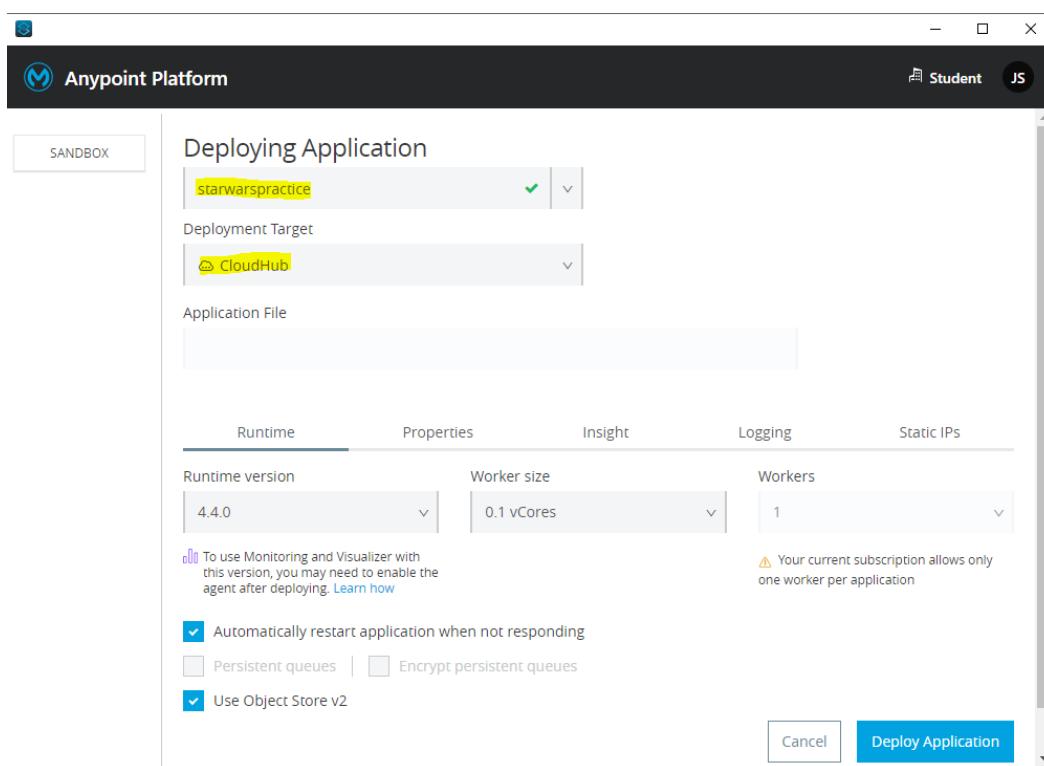
Log in with our Anypoint Platform account in the window that opens.



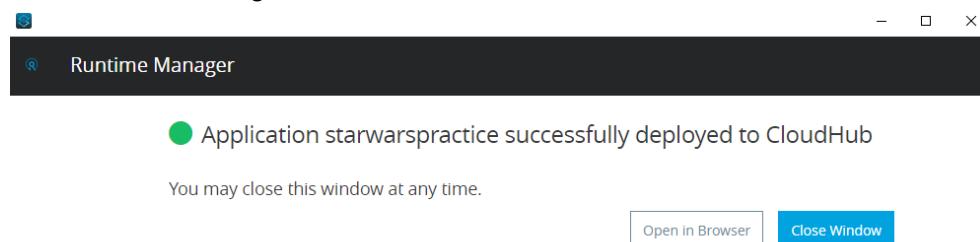
Choose “Sandbox” option



Assign the next properties



Wait to see the next message



In the "Runtime Manager" section of our Anypoint Platform session we can see our project deployed.

The screenshot shows the Runtime Manager interface with the following details:

- Sandbox** tab selected.
- Deploy application** button.
- Search Applications** input field.
- All Applications (2)** section header.
- Name**, **Target Name**, **Target Type**, **Status**, **Runtime Version**, and **Date Modified** columns.
- demo-invokerestapi**: Target Name CloudHub, Target Type CloudHub, Status Started, Runtime Version 4.4.0, Date Modified 2023-06-26 12:33:55.
- starwarspractice**: Target Name CloudHub, Target Type CloudHub, Status Started, Runtime Version 4.4.0, Date Modified 2023-06-27 00:59:31.

From here we can obtain our public endpoint:
<http://starwarspractice.us-e2.cloudhub.io/characters>

The screenshot shows the Runtime Manager dashboard for the **starwarspractice** application:

- Dashboard** tab selected.
- Domain**: starwarspractice.us-e2.cloudhub.io Last Updated 2023-06-27 12:59:31AM - 1 micro worker, using 4.4.0.
- Mule messages** section with time filters: Last hour, Last 24hs, Last week.
- CPU** usage chart from 01:01:27 to 23:31:27, showing 100% usage for Worker 3.138.137.22.

We can now make requests with the following URLs

ALL CHARACTERS:

The screenshot shows a Postman request for the **starwarspractice** API:

- Method**: GET
- URL**: <http://starwarspractice.us-e2.cloudhub.io/characters>
- Response** status: 200
- Time**: 40893 ms
- Size**: 4.21 KB
- Content** (text):

```

1 name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 Luke Skywalker,172,77,blond,fair,blue,19BBY,male
3 C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
4 R2-D2,96,32,n/a,white\, blue,red,33BBY,n/a
5 Darth Vader,202,136,none,white,yellow,41.9BBY,male
6 Leia Organa,150,49,brown,light,brown,19BBY,female
7 Owen Lars,178,120,brown\, grey,light,blue,52BBY,male
8 Beru Whitesun Lars,165,75,brown,light,blue,47BBY,female
9 R5-D4,97,32,n/a,white\, red,red,unknown,n/a
10 Biggs Darklighter,183,84,black,light,brown,24BBY,male
11 Obi-Wan Kenobi,182,77,auburn\, white,fair,blue-gray,37BBY,male
12 Anakin Skywalker,188,84,blond,fair,blue,41.9BBY,male
13 Willhuff Tarkin,180,unknown,auburn\, grey,fair,blue,64BBY,male
14 Chewbacca,228,112,brown,unknown,blue,200BBY,male
15 Han Solo,180,80,brown,fair,brown,29BBY,male
16 Greedo,173,74,n/a,green,black,44BBY,male

```

CHARACTERS BY GENDER (gender = female)

...gender?gender=female +

GET http://starwarspractice.us-e2.cloudhub.io/charactersbygender?gender=female

Response

200 Time: 35184 ms Size: 989 Bytes

```

1 | name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 | Leia Organa,150,49,brown,light,brown,19BBY,female
3 | Beru Whitesun Lars,165,75,brown,light,blue,47BBY,female
4 | Mon Mothma,150,unknown,auburn,fair,blue,48BBY,female
5 | Padmé Amidala,185,45,brown,light,brown,46BBY,female
6 | Shmi Skywalker,163,unknown,black,fair,brown,72BBY,female
7 | Ayla Secura,178,55,none,blue,hazel,48BBY,female
8 | Adi Gallia,184,50,none,dark,blue,unknown,female
9 | Cordé,157,unknown,brown,light,brown,unknown,female
10 | Luminara Unduli,170,56,2,black,yellow,blue,58BBY,female
11 | Barriss Offee,166,50,black,yellow,blue,40BBY,female
12 | Dormé,165,unknown,brown,light,brown,unknown,female
13 | Zam Wesell,168,55,blonde,fair,green,yellow,yellow,unknown,female
14 | Taun We,213,unknown,none,grey,black,unknown,female
15 | Jocasta Nu,167,unknown,white,fair,blue,unknown,female
16 |

```

CHARACTERS BY GENDER (gender = male)

...sbygender?gender=males +

GET http://starwarspractice.us-e2.cloudhub.io/charactersbygender?gender=males

Response

200 Time: 44190 ms Size: 3.07 KB

```

1 | name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 | Luke Skywalker,172,77,blond,fair,blue,19BBY,male
3 | Darth Vader,202,136,none,white,yellow,41.9BBY,male
4 | Owen Lars,178,120,brown,grey,light,blue,52BBY,male
5 | Biggs Darklighter,183,84,black,light,brown,24BBY,male
6 | Obi-Wan Kenobi,182,77,auburn,white,fair,blue-gray,57BBY,male
7 | Anakin Skywalker,188,84,blond,fair,blue,41.9BBY,male
8 | Wilhuff Tarkin,180,unknown,auburn,grey,fair,blue,64BBY,male
9 | Chewbacca,228,112,brown,unknown,blue,200BBY,male
10 | Han Solo,180,80,brown,fair,brown,29BBY,male
11 | Greedo,173,74,n/a,green,black,44BBY,male
12 | Wedge Antilles,170,77,brown,fair,hazel,21BBY,male
13 | Jek Tono Porkins,180,110,brown,fair,blue,unknown,male
14 | Yoda,66,17,white,green,brown,896BBY,male
15 | Palpatine,170,75,gray,pale,yellow,52BBY,male
16 | Boba Fett,188,78,2,black,fair,brown,31.5BBY,male

```

CHARACTERS BY GENDER (gender = n/a)

...ersbygender?gender=n/ax +

GET http://starwarspractice.us-e2.cloudhub.io/charactersbygender?gender=n/a

Response

200 Time: 46649 ms Size: 194 Bytes

```

1 | name,height,mass,hair_color,skin_color,eye_color,birth_year,gender
2 | C-3PO,167,75,n/a,gold,yellow,112BBY,n/a
3 | R2-D2,98,32,n/a,white,blue,red,33BBY,n/a
4 | RS-D4,97,32,n/a,white,red,red,unknown,n/a

```