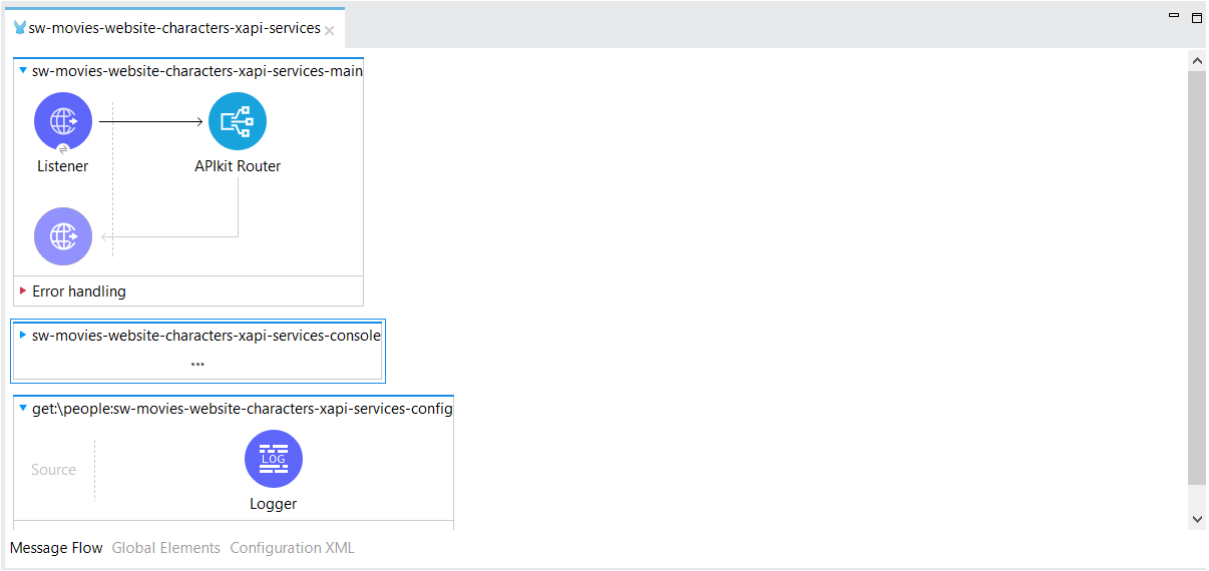
To make this exercise, the first step is the Raml elaboration of according to the indicated requirements.

```
#%RAML 1.0
title: sw-movies-website-characters-xapi-services
baseUri: https://swapi.dev/api/
version: 1.0
protocols:
  - HTTP
mediaType:
- application/json
description: This API will only retrieves all Star Wars movies characters

/people:
  get:
    queryParameters:
      gender:
        displayName: gender
        type: string
        description: Filter by Character gender
        example: male
        required: false

    responses:
      200:
        description: Retrieves all Star Wars characters
        body:
          application/json:
            properties:
              name:
                type: string
                example: Luke Skywalker
              height:
                type: number
                example: 172
              mass:
                type: number
                example: 77
              hair_color:
                type: string
                example: blond
              skin_color:
                type: string
                example: fair
              eye_color:
                type: string
                example: blue
              birth_year:
                type: string
                example: 19BBY
              gender:
                type: string
                example: male
      401:
        description: Bad request
      404:
        description: Not found
```

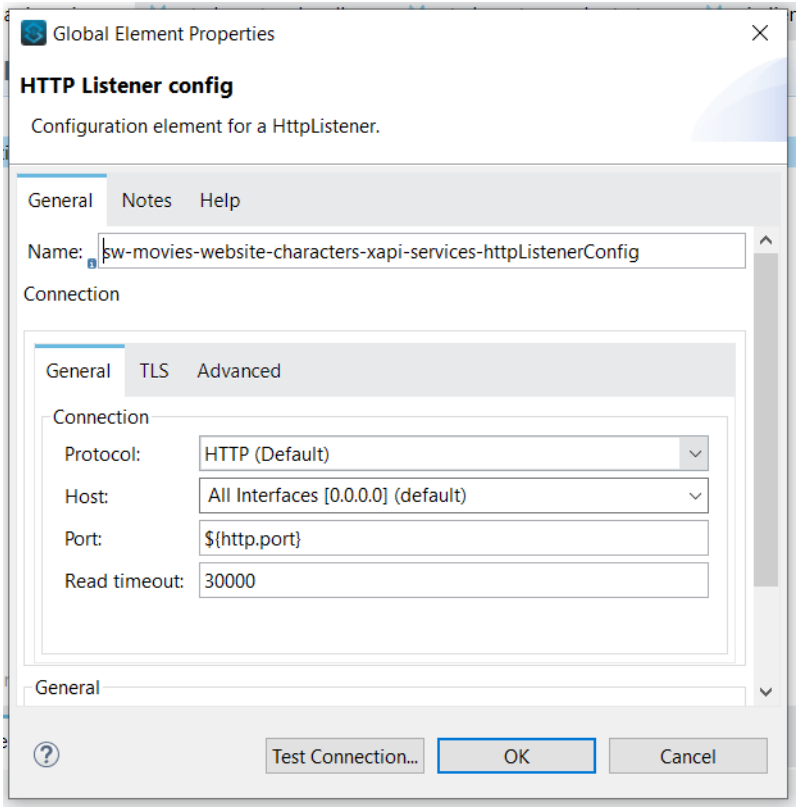
Once the Raml is done, we publish it on Exchage.

Then we open Anypoint Studio and create a new Mule project, where we are going to download our Raml from the Desing Center.

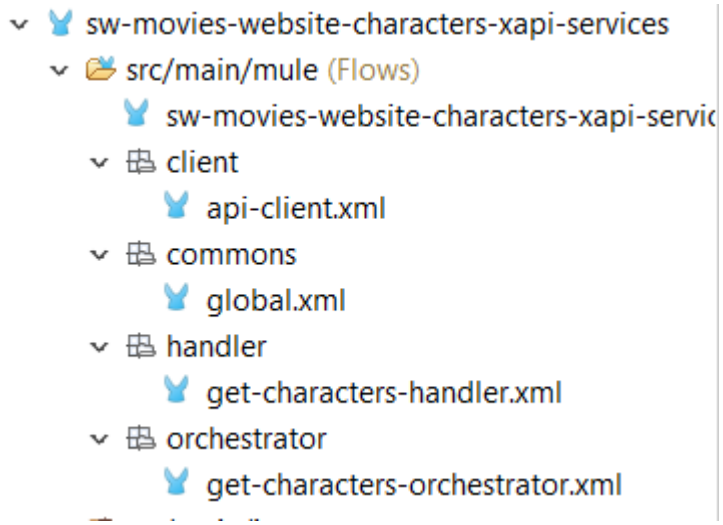
This will make automatically scaffold our application.



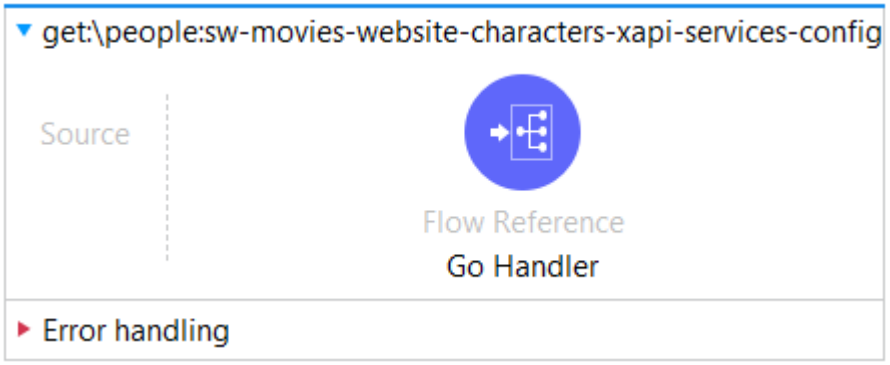
We configure our listener with our properties from our config file



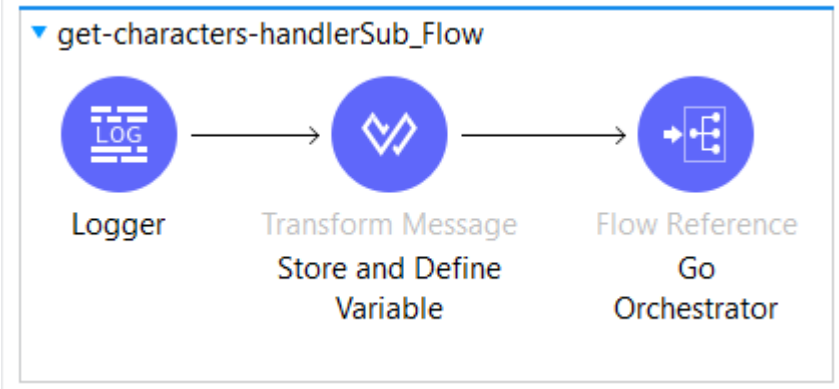
We make our flow structure.

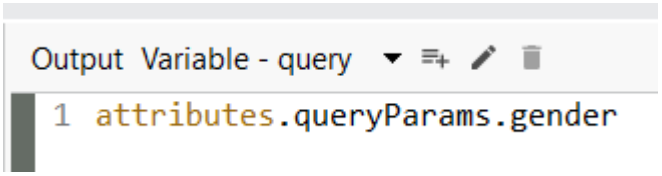


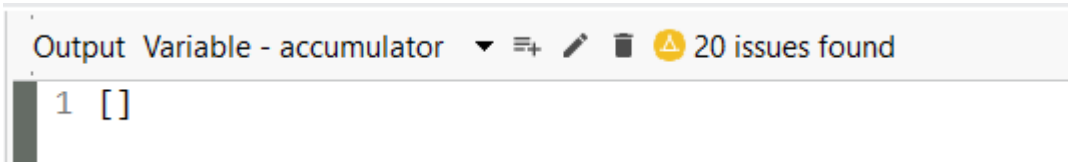
We make our flows and sub-flows
We start by directing our main flow to our handler package where we store everything that comes from the HTTP request.



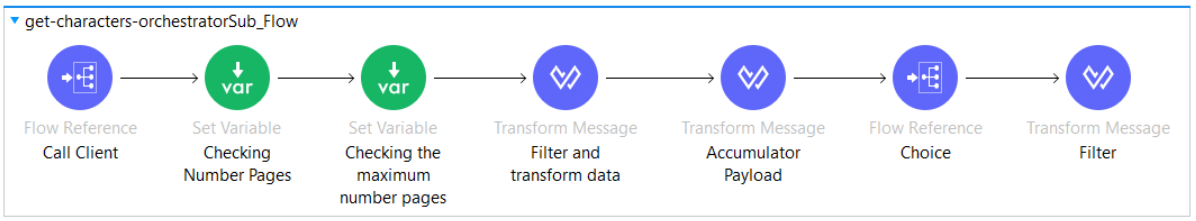
We store and define our variables with a Transform Message process

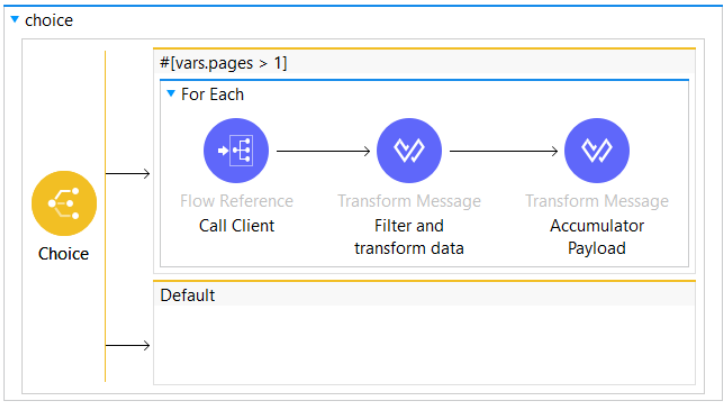




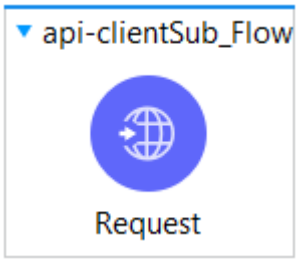


We direct our flow to our orchestrator, which orchestrates our flow and transforms the information to which we will respond.

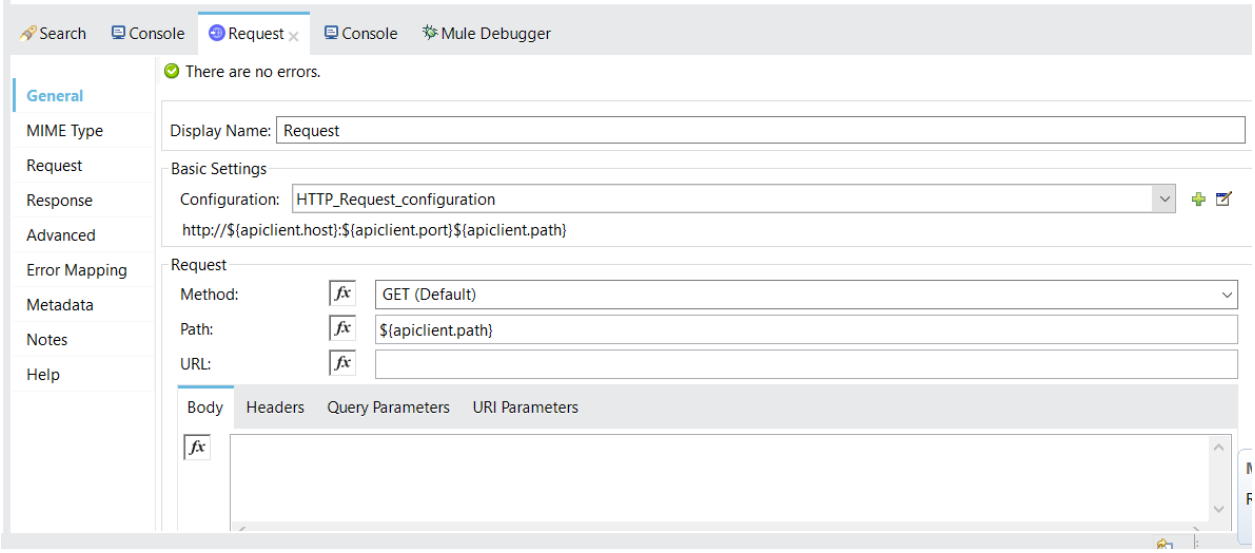


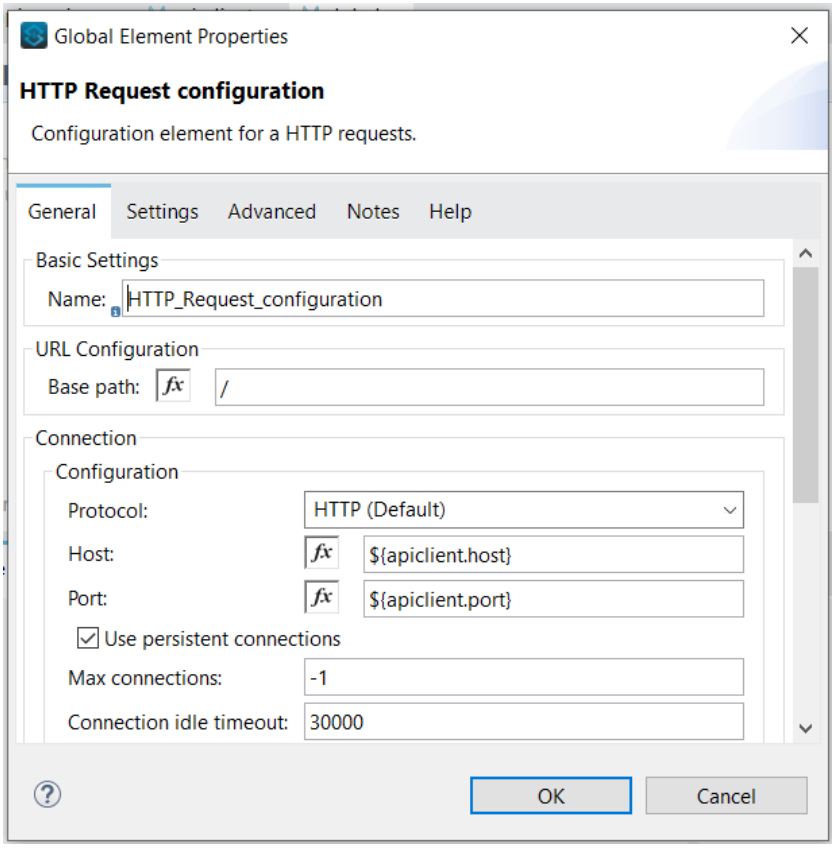


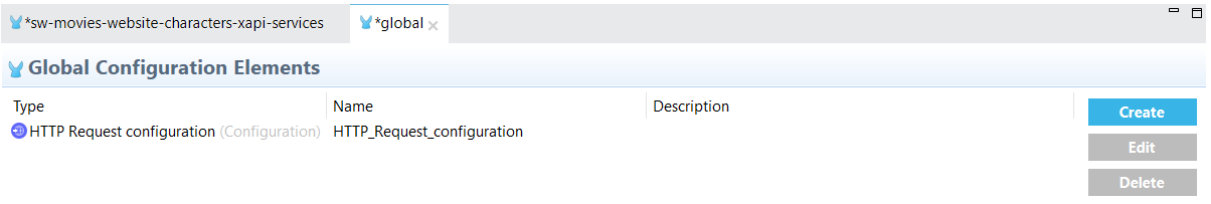
Before transforming the information we proceed to make the call to our client, in this case a web page through a Request connector



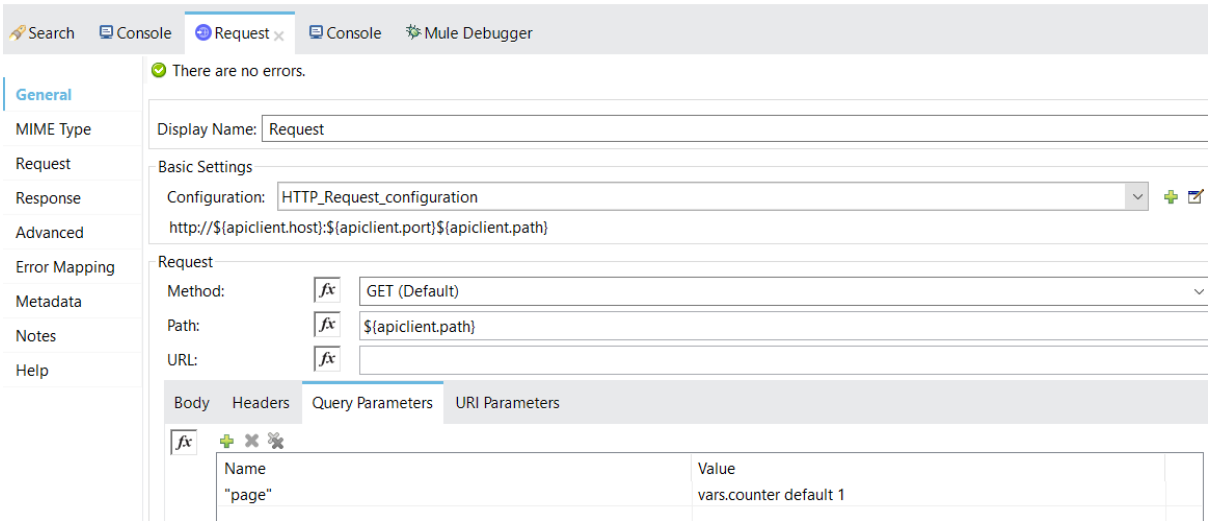
We configure our request and move our configuration to the commons package in our configuration file called global



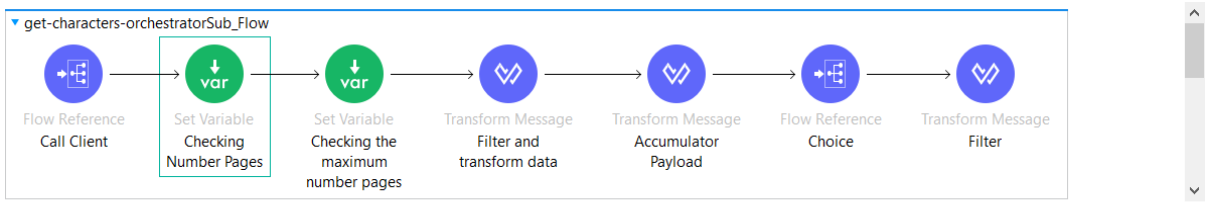




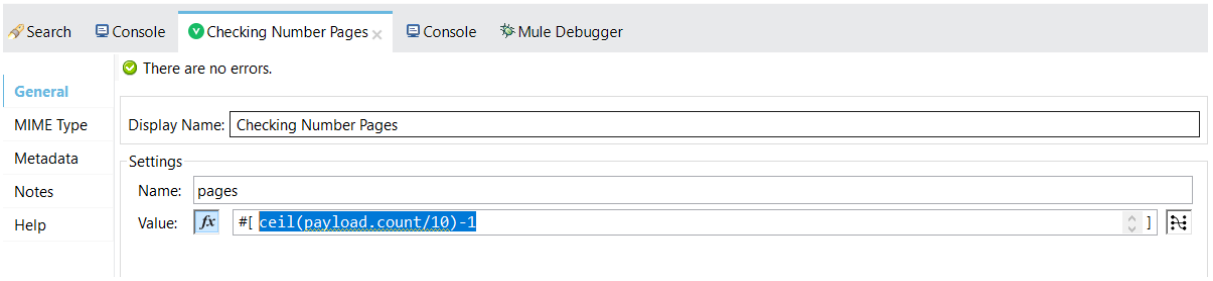
We put a Query Parameters to query all the pages that the response contains



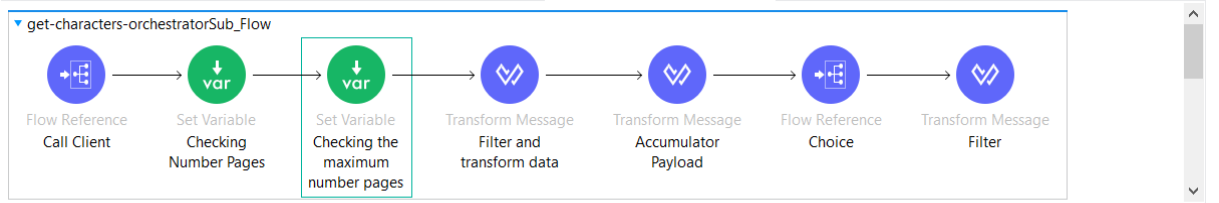
We search the number of Star Wars characters and divide them by 10



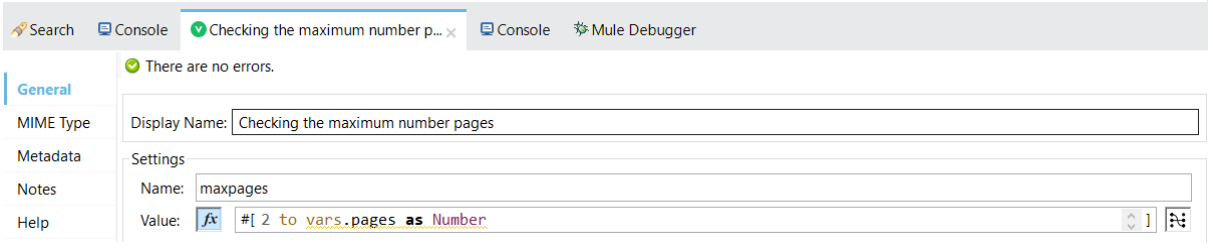
Message Flow Global Elements Configuration XML



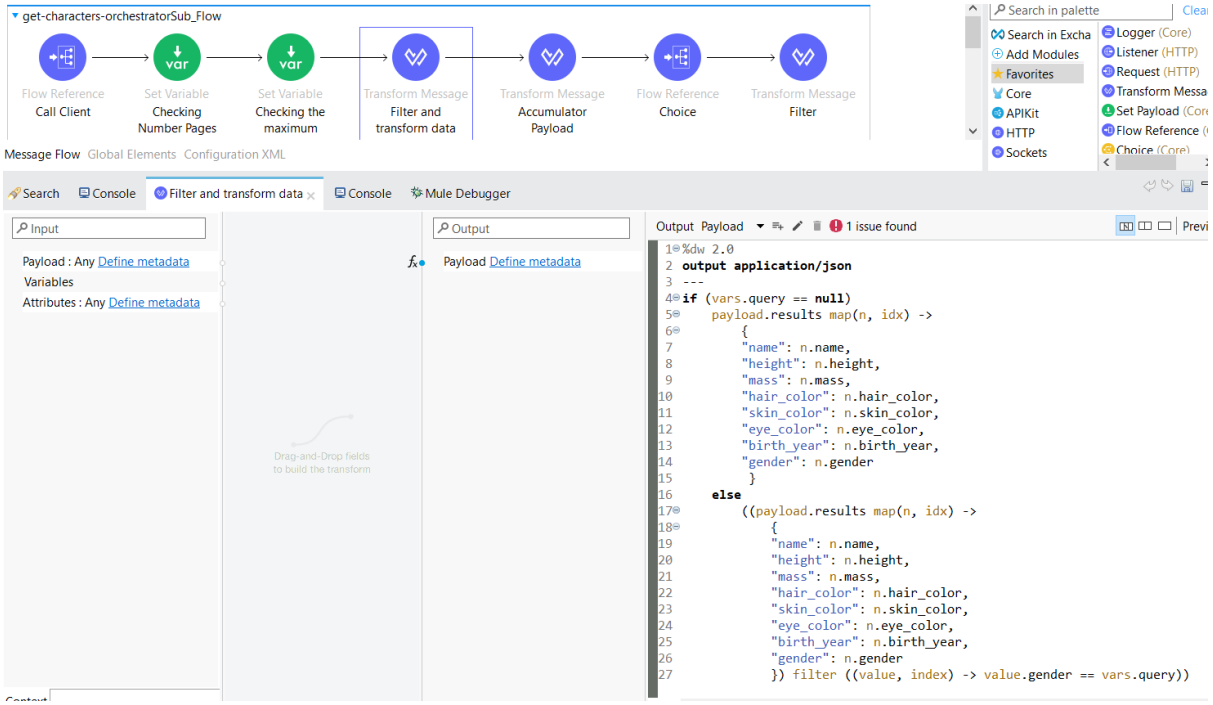
We make our array



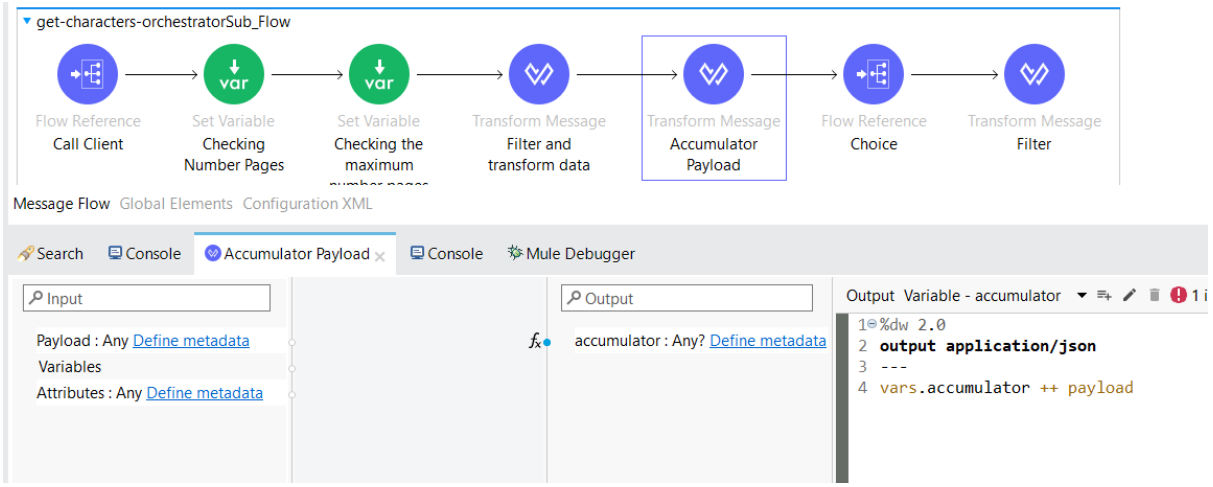
Message Flow Global Elements Configuration XML



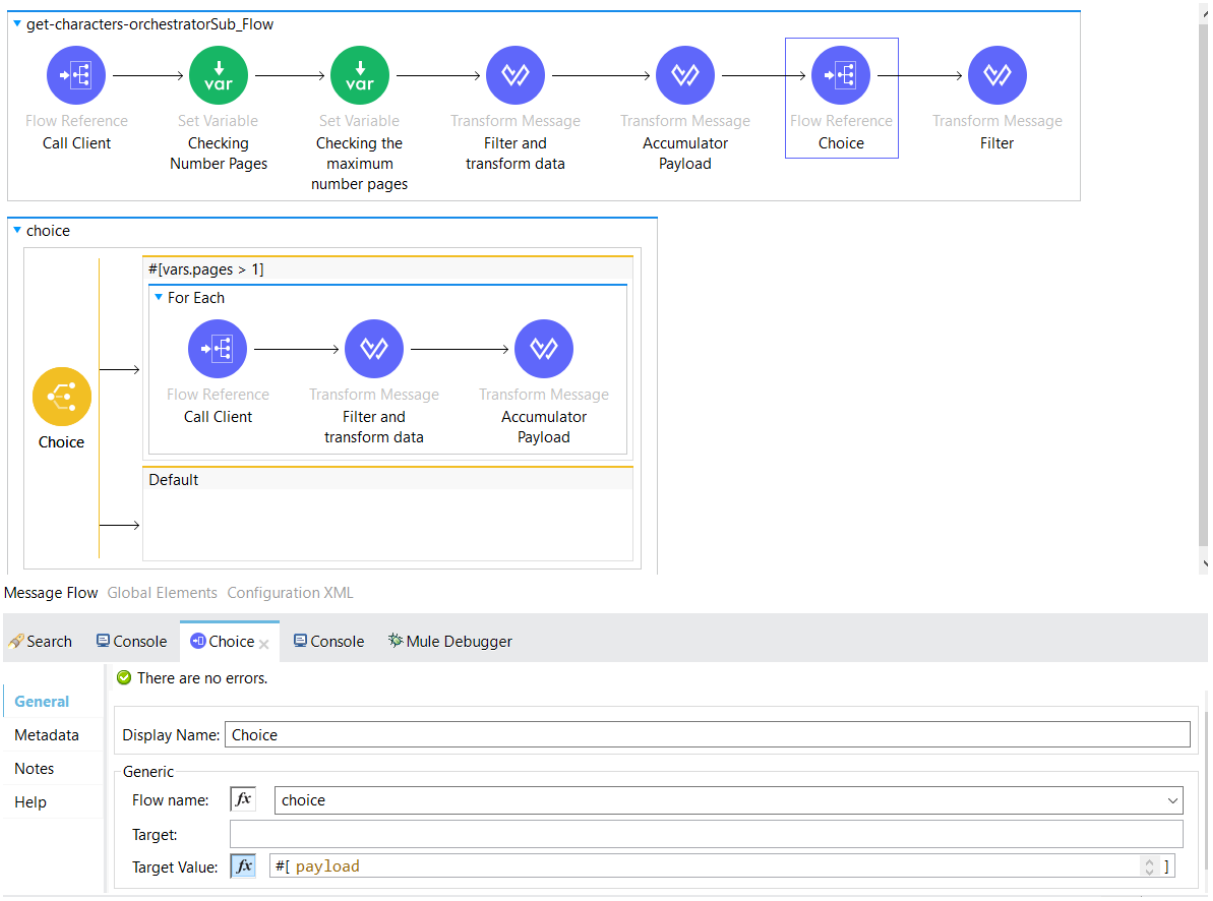
We configure our Transform Message process to transform and filter the information that we are going to accumulate in our array and display according to the request.



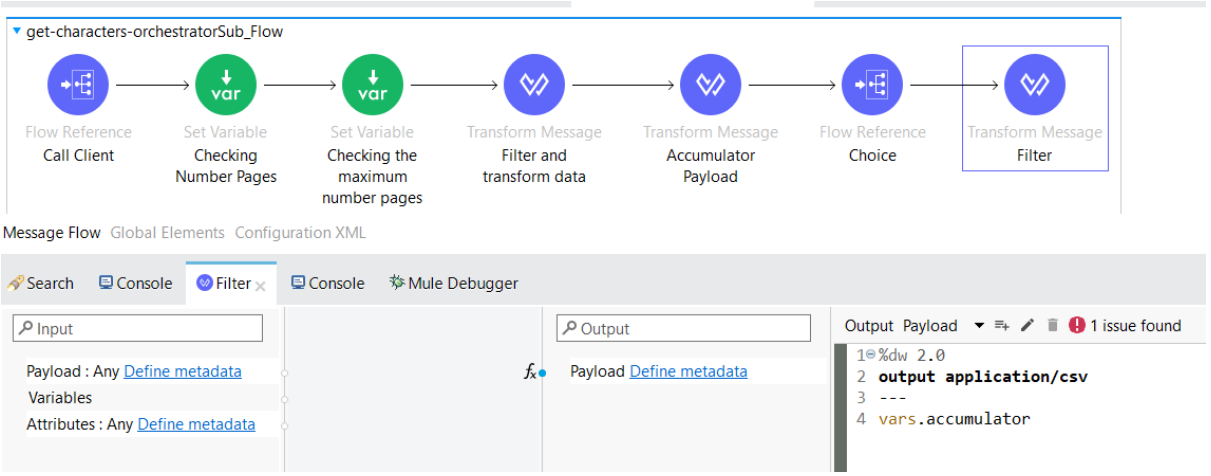
We accumulate the information



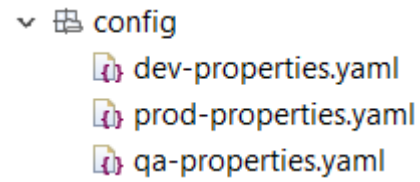
We make a condition if, if it is fulfilled, the cycle will be repeated until our array is formed

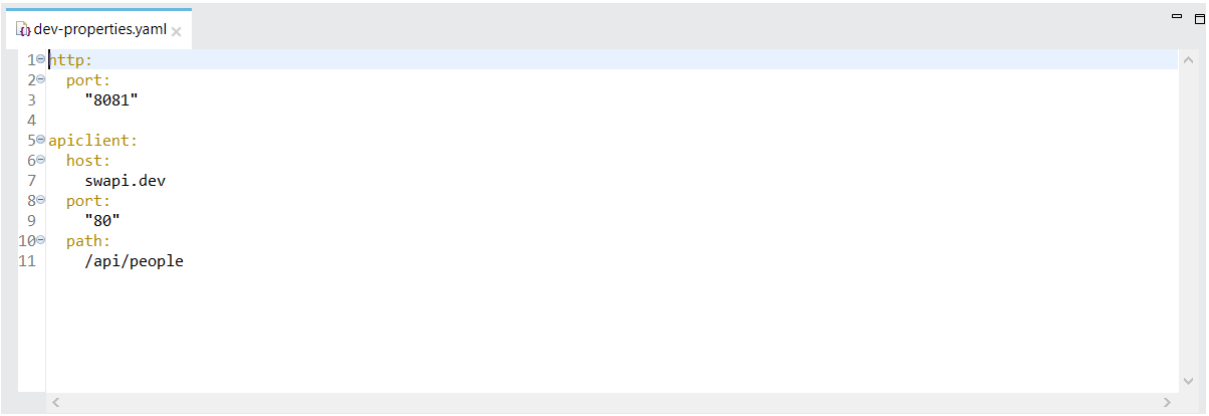


Finally we print our array in csv format

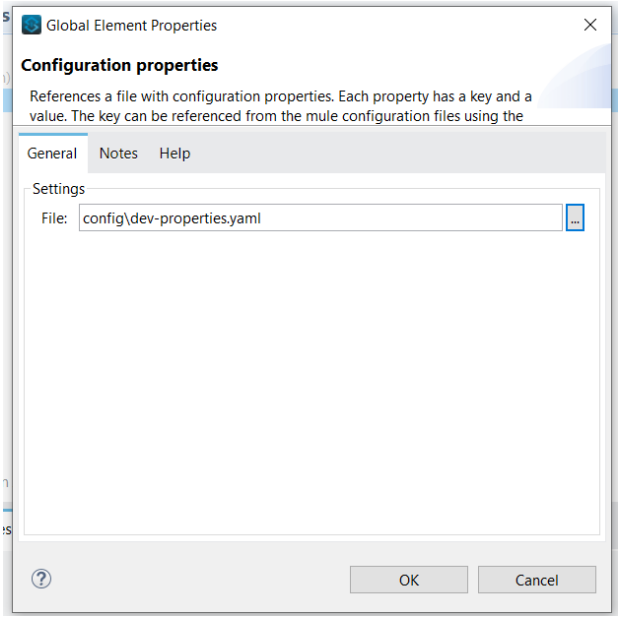


We proceed to create our properties file where we define our environment variables for each respective environment





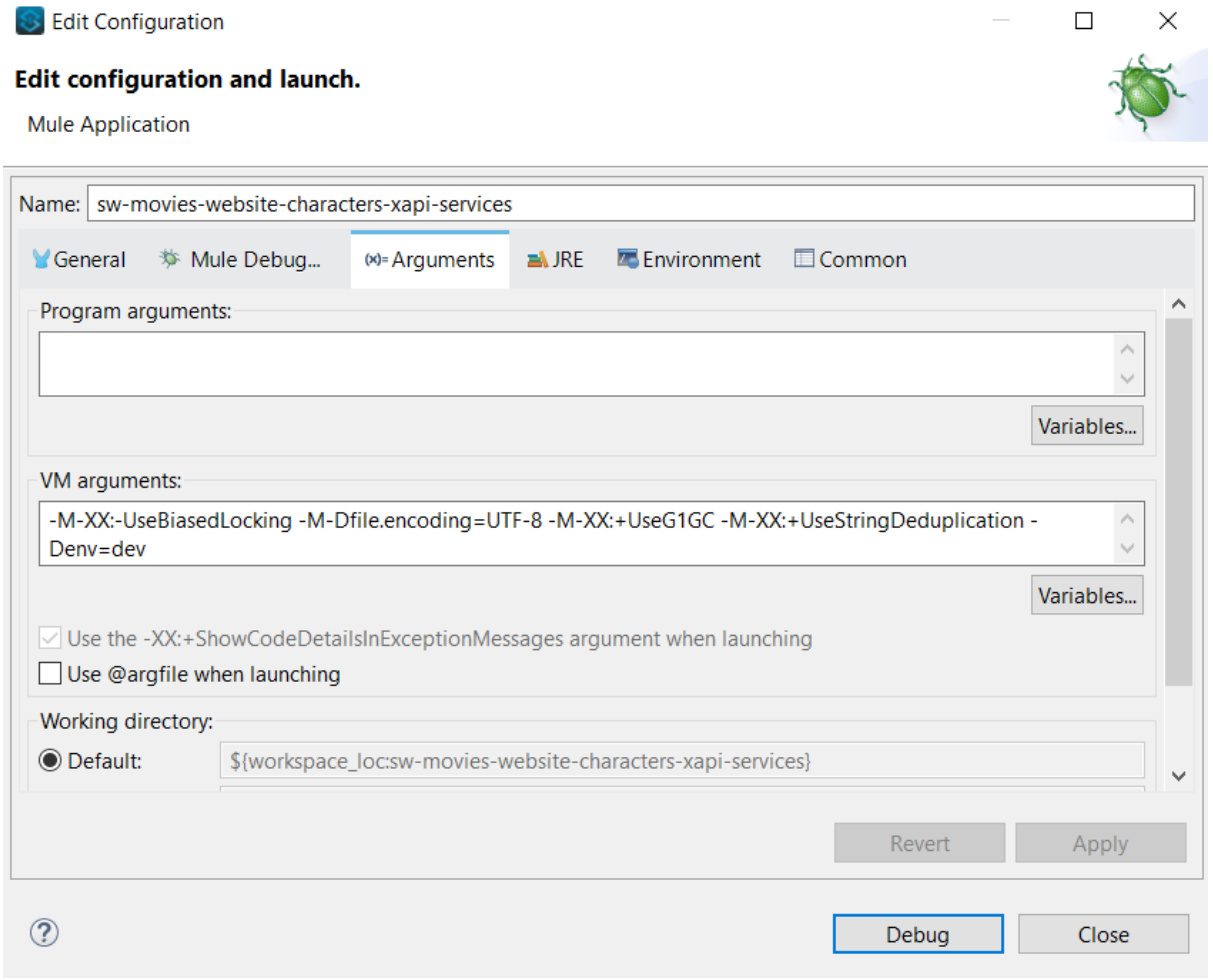
Later we proceed to create our configuration properties file and link it to the environment we are working on.



In order not to have to make changes to the file in its respective environment, we declare an environment variable to later specify the environment with which we are working.

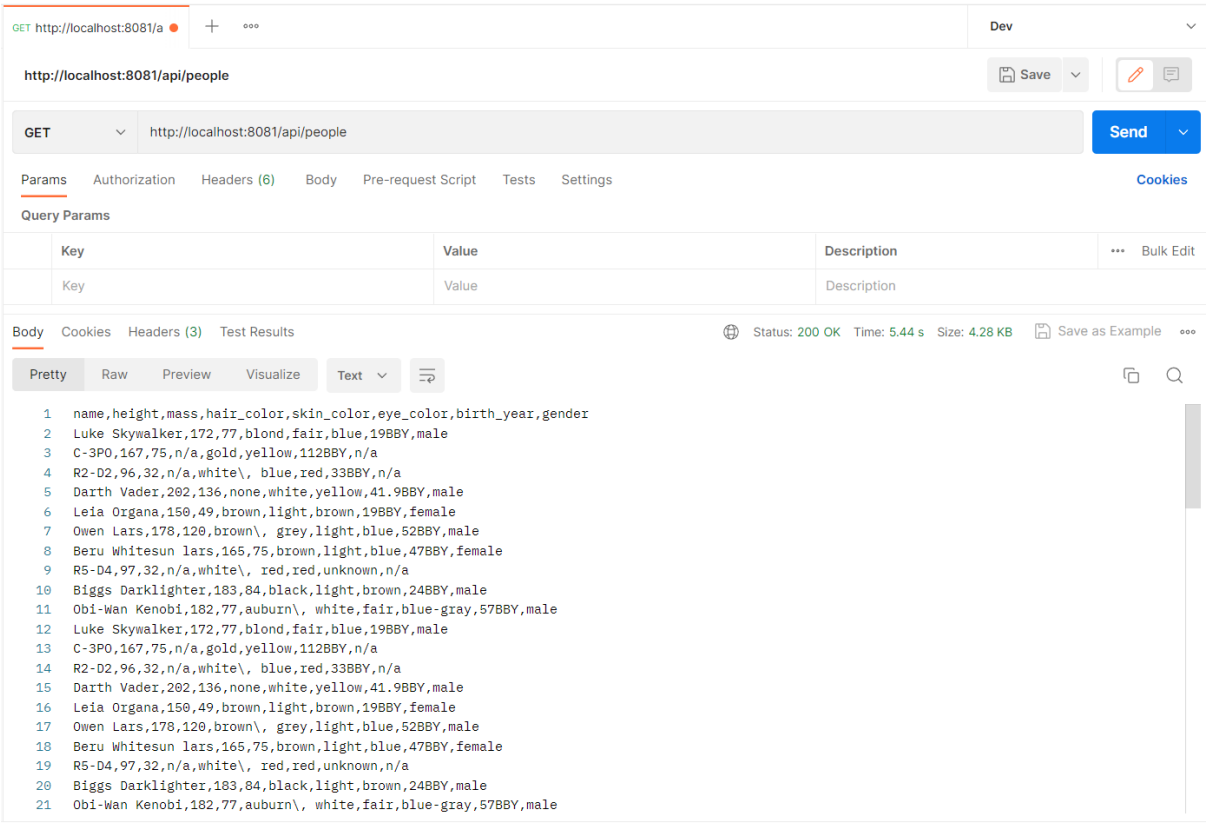
```
file="config\\${env}-properties.yaml" />
```

After all these steps we proceed to run our mule application:
In the arguments we declare our environment variable to work with the desired environment

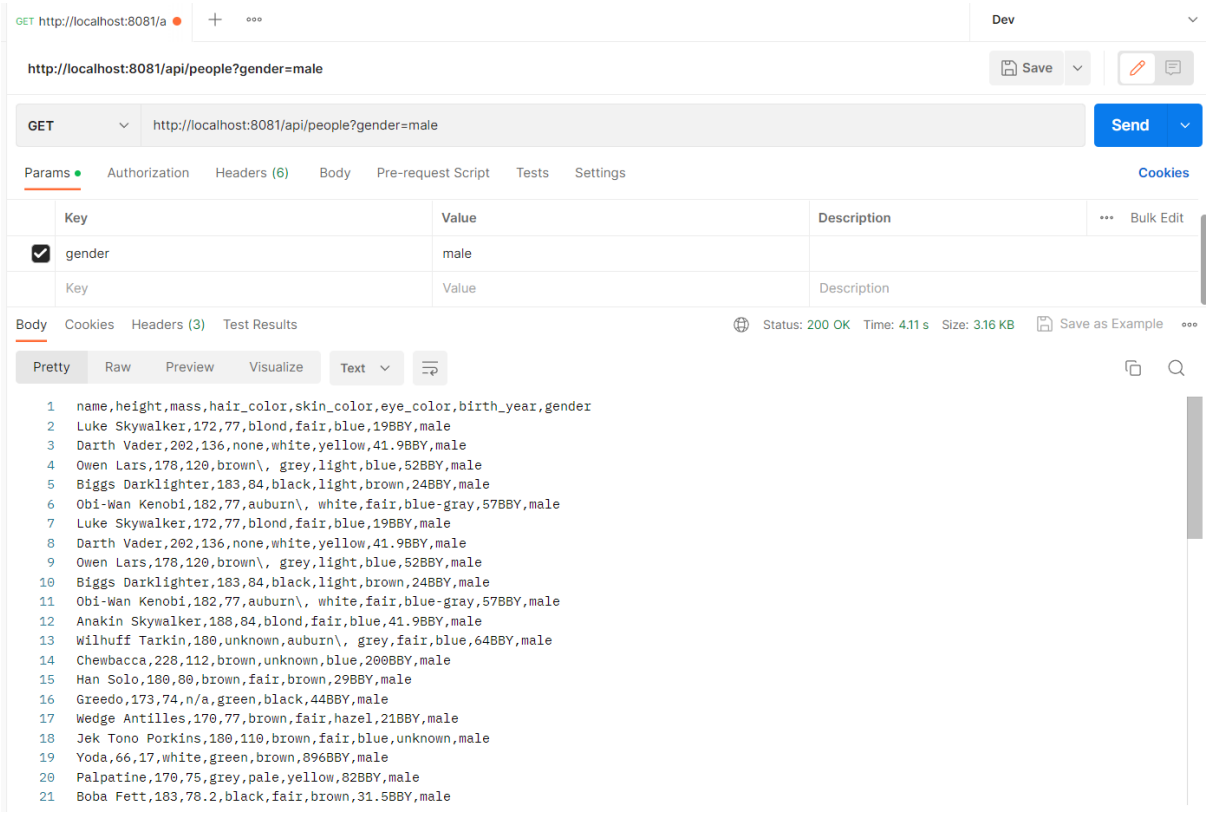


Once our application is deployed locally, we proceed to carry out tests with postman:

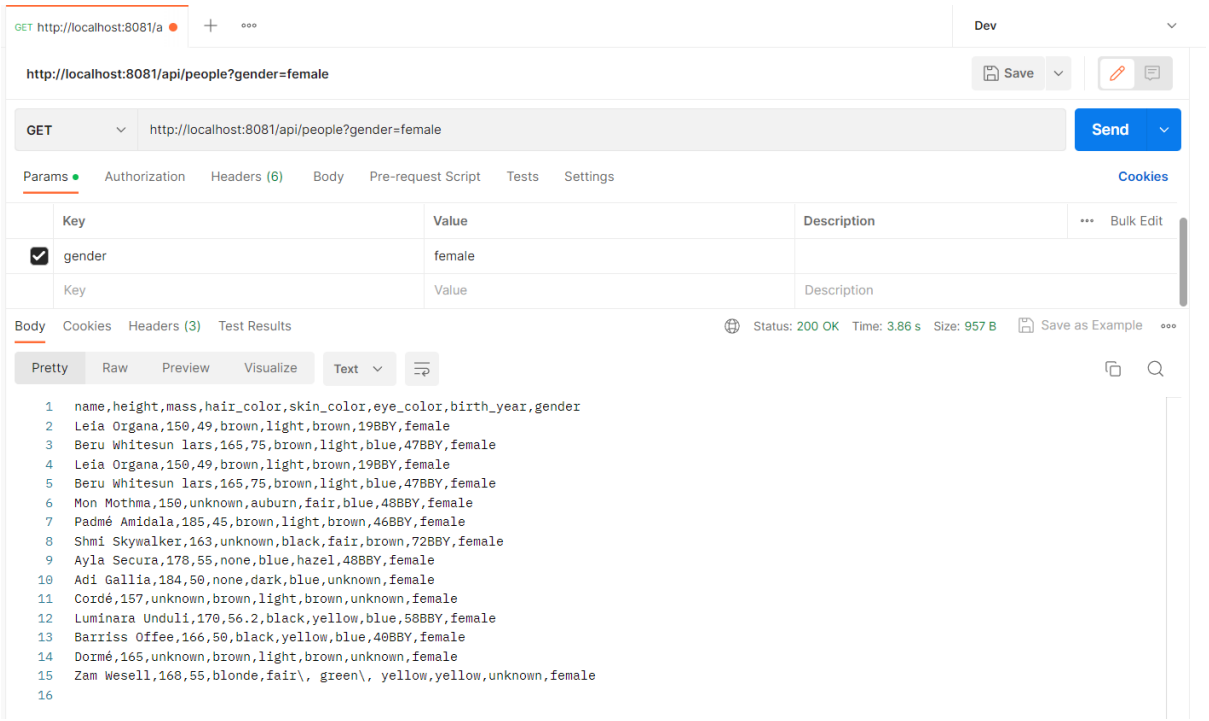
hhtp://localhost:8081/api/people



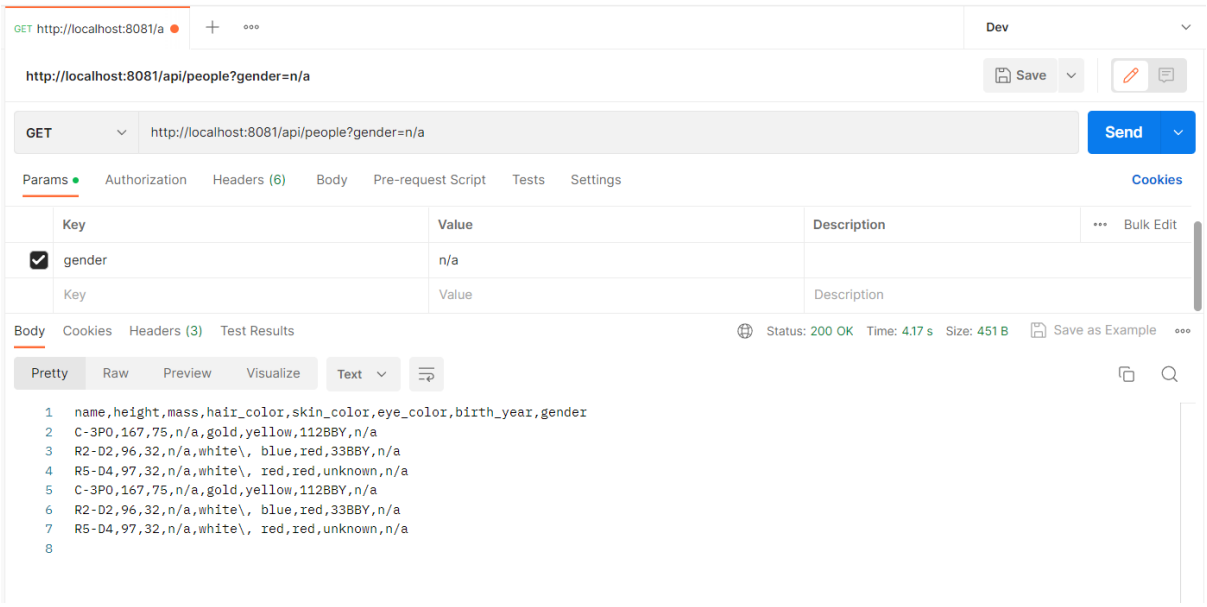
hhtp://localhost:8081/api/people?gender=male



hhttp://localhost:8081/api/people?gender=female



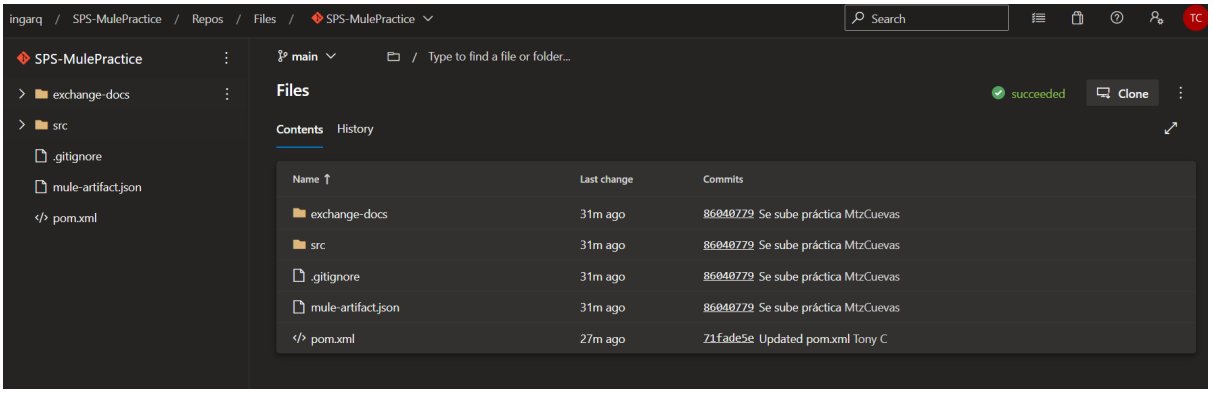
http://localhost:8081/api/people?gender=n/a



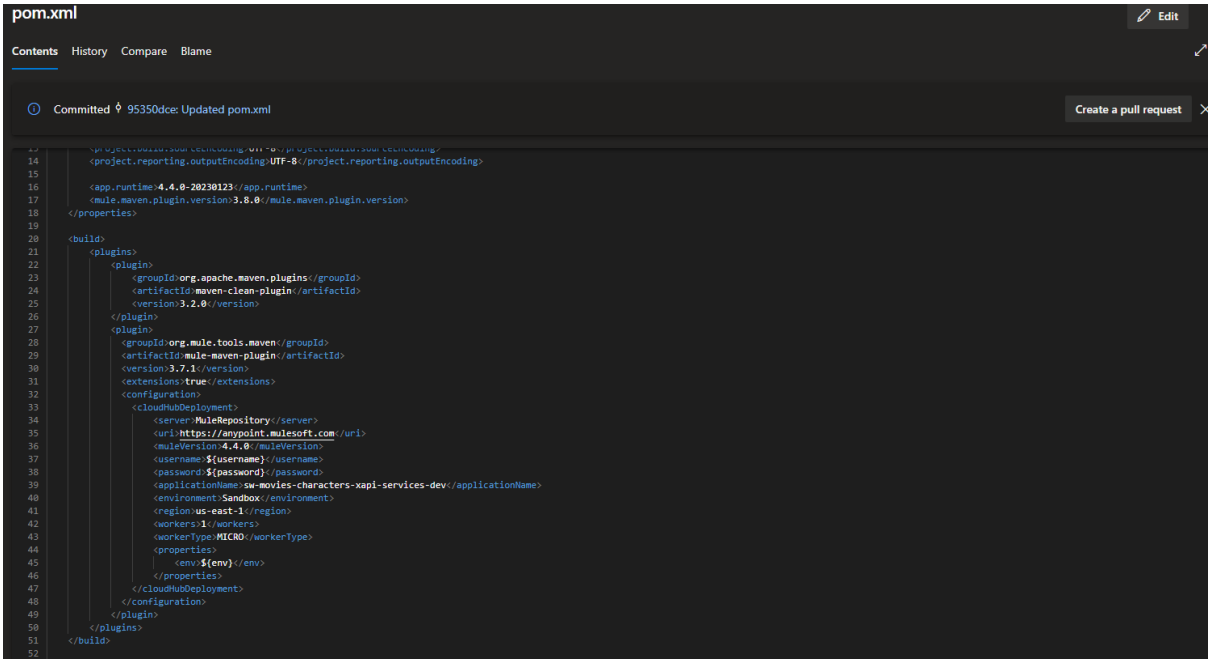
Once the tests have been maked out and verify that everything responds as required.
We proceed to deploy our mule application to the MuleSoft CloudHub cloud.

For this step i'll implementing CI/CD with Azure Pipelines:

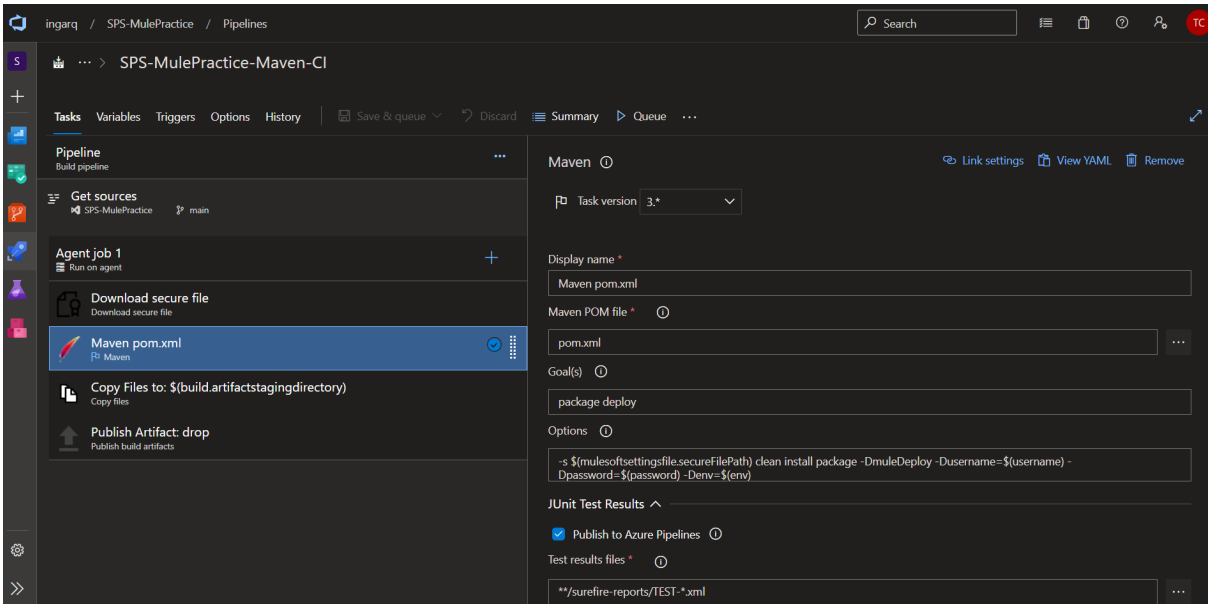
In this step, I import the GitHub repository into an Azure Repos repository, the changes are made in the pom.xml to be able to do the Deploy in CloudHub



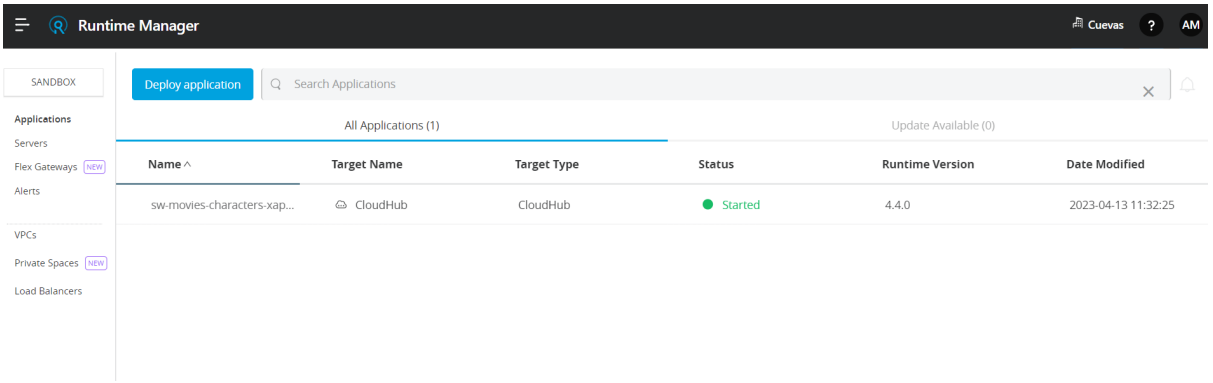
We indicate the server, which must be specified in our settings.xml file, the user, password, name of the deployed application, the environment where we will do our deploy, the region, the number of workers and the size with which we will will run our muleapp, in addition to the properties of our deploy.

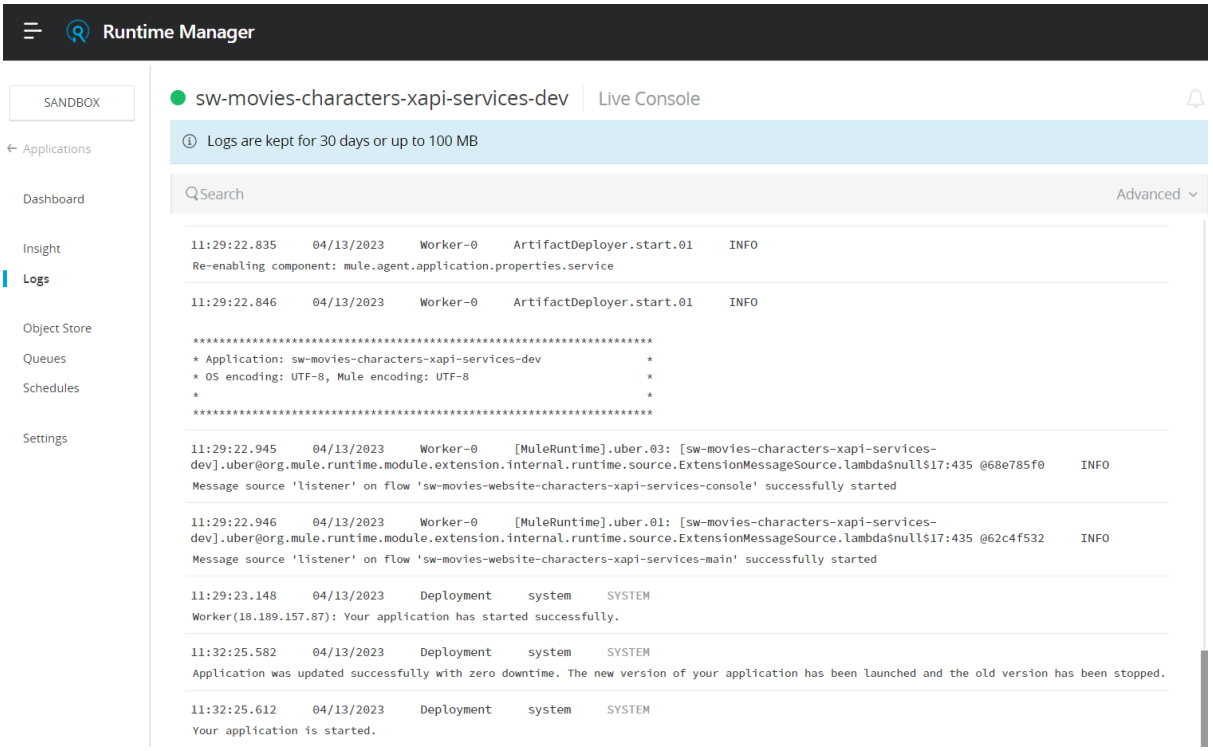


A pipeline is made, with a Maven template, it is configured and run

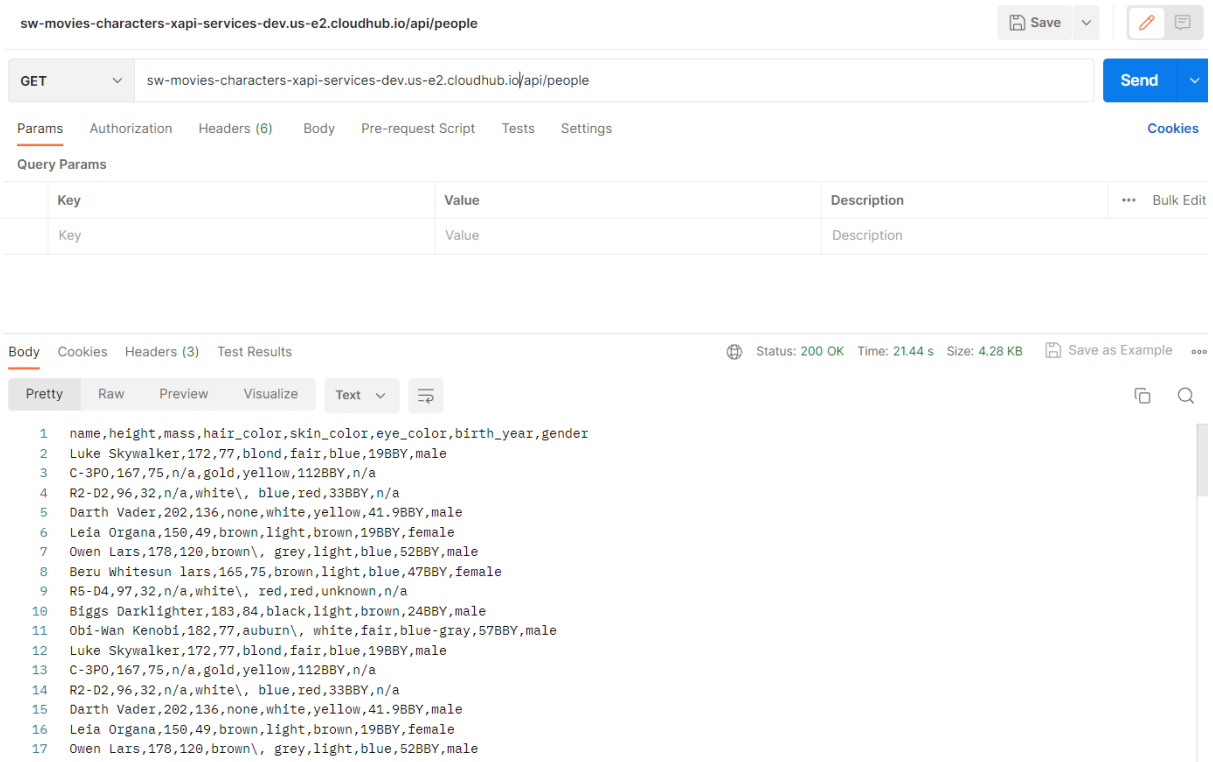


We deploy our cloud application on mulesoft CloudHub





Once our application is deployed locally, we proceed to carry out tests with postman:
sw-movies-characters-xapi-services-dev.us-e2.cloudhub.io/api/people



sw-movies-characters-xapi-services-dev.us-e2.cloudhub.io/api/people?gender=male