



CONSUMINDO APIs REST COM OUTSYSTEMS

Eduardo Freitas

INTRODUÇÃO

Introdução

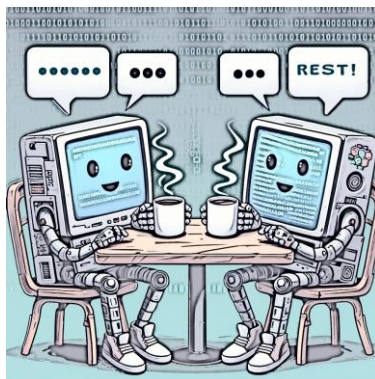
Bem-vindo ao guia sobre como utilizar APIs REST com o OutSystems

Neste eBook, vamos simplificar o entendimento e o uso de APIs REST, oferecendo exemplos práticos de código em contextos reais.

Dominar APIs REST é fundamental para qualquer desenvolvedor moderno. Elas permitem a comunicação eficiente entre sistemas e são essenciais para a integração de aplicativos e serviços.

Nosso objetivo é tornar o processo de compreensão e uso de APIs REST no OutSystems simples e acessível. Vamos abordar os fundamentos e explicar como consumir APIs, sempre fornecendo exemplos práticos ao longo do caminho.

Vamos começar!



01

APIs REST - CONCEITOS

O que são? Onde vivem? Do que se alimentam?
Neste capítulo você aprenderá sobre endpoints, métodos HTTP e respostas JSON, conceitos essenciais para entender como as APIs REST funcionam.

APIs REST - CONCEITOS

APIs REST (Representational State Transfer) são como pontes que conectam diferentes sistemas na internet. Elas usam uma linguagem comum chamada HTTP para permitir essa comunicação de forma eficiente. Em termos simples, elas são como

mensageiros que entregam suas solicitações de um lugar para outro na web.

Vamos imaginar um cenário simples para ilustrar como uma API funciona na prática. Imagine que você está usando um aplicativo de previsão do tempo em seu smartphone para verificar o clima em sua cidade. O fluxo de mensagens nesse caso é o seguinte:

- Quando você abre o aplicativo e solicita a previsão do tempo, ele envia uma solicitação ao servidor onde estão armazenados os dados meteorológicos;
- O servidor possui uma API de previsão do tempo que permite que outros aplicativos acessem seus dados meteorológicos.
- A API processa a solicitação e retorna os dados meteorológicos atualizados para o aplicativo.
- O aplicativo recebe os dados da API e os exibe na tela para que você possa ver a previsão do tempo atualizada.

APIs REST - CONCEITOS

PRINCIPAIS CONCEITOS

ENDPOINTS

São como endereços da internet que apontam para diferentes informações.

Cada endpoint representa um tipo específico de informação na API.

Por exemplo, no nosso exemplo de aplicação de previsão do tempo alguns endpoints típicos poderiam ser:

- `/weather/current`: fornece informações sobre as condições meteorológicas atuais, como temperatura, umidade, velocidade do vento, etc.
- `/weather/city/{city_name}`: retorna a previsão do tempo para uma cidade específica, fornecendo informações sobre temperatura, umidade, e condições climáticas esperadas para aquela localidade.
- `/weather/alerts`: retorna alertas meteorológicos ativos, como avisos de tempestades, tornados ou outras condições climáticas perigosas.

APIs REST - CONCEITOS

PRINCIPAIS CONCEITOS

MÉTODOS HTTP

São as ações que podemos realizar em um determinado endpoint.

Os métodos mais comuns são:

- **GET:** usado para recuperar informações de um endpoint. Por exemplo: podemos usar GET no endpoint `/weather/current` para obter informações sobre as condições meteorológicas atuais..
- **POST:** usado para criar novos recursos em um endpoint. Podemos usar POST no endpoint `/weather/alerts` para adicionar um novo alerta meteorológico ao sistema.
- **PUT:** usado para atualizar recursos existentes em um endpoint. Ao utilizar PUT no endpoint `/weather/location/{latitude}/{longitude}`, podemos atualizar as informações meteorológicas para uma determinada localização com base nas novas coordenadas geográficas fornecidas.
- **DELETE:** usado para excluir recursos de um endpoint. Por exemplo, podemos usar DELETE no endpoint `/weather/alerts/{alert_id}` para remover um alerta meteorológico específico do sistema.

APIs REST - CONCEITOS

PRINCIPAIS CONCEITOS

RESPOSTAS JSON

São as mensagens que recebemos de volta quando fazemos uma solicitação a um endpoint

Essas respostas são formatadas em JSON, que é um formato de texto simples e legível por humanos. Ele organiza os dados de uma forma que facilita a compreensão e manipulação pelos computadores.

Vamos ver como seria uma resposta em JSON contendo informações detalhadas sobre as condições meteorológicas.

```
{
  "localizacao": "São Paulo",
  "clima_atual": {
    "temperatura": 25,
    "umidade": 70,
    "velocidade_vento": 10,
    "condicoes": "Nublado"
  },
  "previsao_diaria": [
    {
      "data": "2024-05-10",
      "temperatura_maxima": 28,
      "temperatura_minima": 20,
      "condicoes": "Parcialmente Nublado"
    }
  ]
}
```


APIs REST - CONCEITOS

EXEMPLO DE API

Agora vamos trocar de cenário: em vez de previsão do tempo, mergulharemos no fascinante mundo de Star Wars!

Imagine que estamos desenvolvendo um aplicativo para fãs de Star Wars e queremos incorporar informações sobre os personagens da saga.

Para isso, vamos acessar a SWAPI, uma api pública que fornece informações sobre Star Wars:

- Acesse o site da SWAPI em <https://swapi.dev/>
- Digite a palavra “people” na caixa de texto, clique em Request e veja a mágica acontecer:

The screenshot shows the SWAPI website interface. At the top, there's a 'Try it now!' button. Below it, a search bar contains the text 'people' and a 'request' button. A hint text below the search bar says 'Need a hint? try people/1/ or planets/3/ or starships/9/'. Below the search bar, the word 'Result:' is displayed. A large code block shows the JSON response for the 'people' endpoint, specifically for Luke Skywalker. The JSON includes fields like 'count', 'next', 'previous', 'results', 'name', 'height', 'mass', 'hair_color', 'skin_color', 'eye_color', 'birth_year', 'gender', 'homeworld', and 'films'. At the bottom of the page, there are three columns of text: 'What is this?', 'How can I use it?', and 'What happened with old swapi.co?'. The 'What is this?' column explains that SWAPI is the world's first quantified and programmatically-accessible data source for all the data from the Star Wars canon. The 'How can I use it?' column states that all data is accessible through the HTTP web API and points to the documentation. The 'What happened with old swapi.co?' column explains that swapi.co is no longer supported and maintained, and that this is an 'unofficial' fork.

Try it now!

<https://swapi.dev/api/> people request

Need a hint? try [people/1/](#) or [planets/3/](#) or [starships/9/](#)

Result:

```
{
  "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/"
      ]
    }
  ]
}
```

What is this?

The Star Wars API, or "swapi" (Swah-pee) is the world's first quantified and programmatically-accessible data source for all the data from the Star Wars canon.

How can I use it?

All the data is accessible through our HTTP web API. Consult our [documentation](#) if you'd like to get started.

What happened with old swapi.co?

swapi.co is not supported and maintained anymore. But since so many projects and tutorials used it as their educational playground, this is an "unofficial" fork.

02

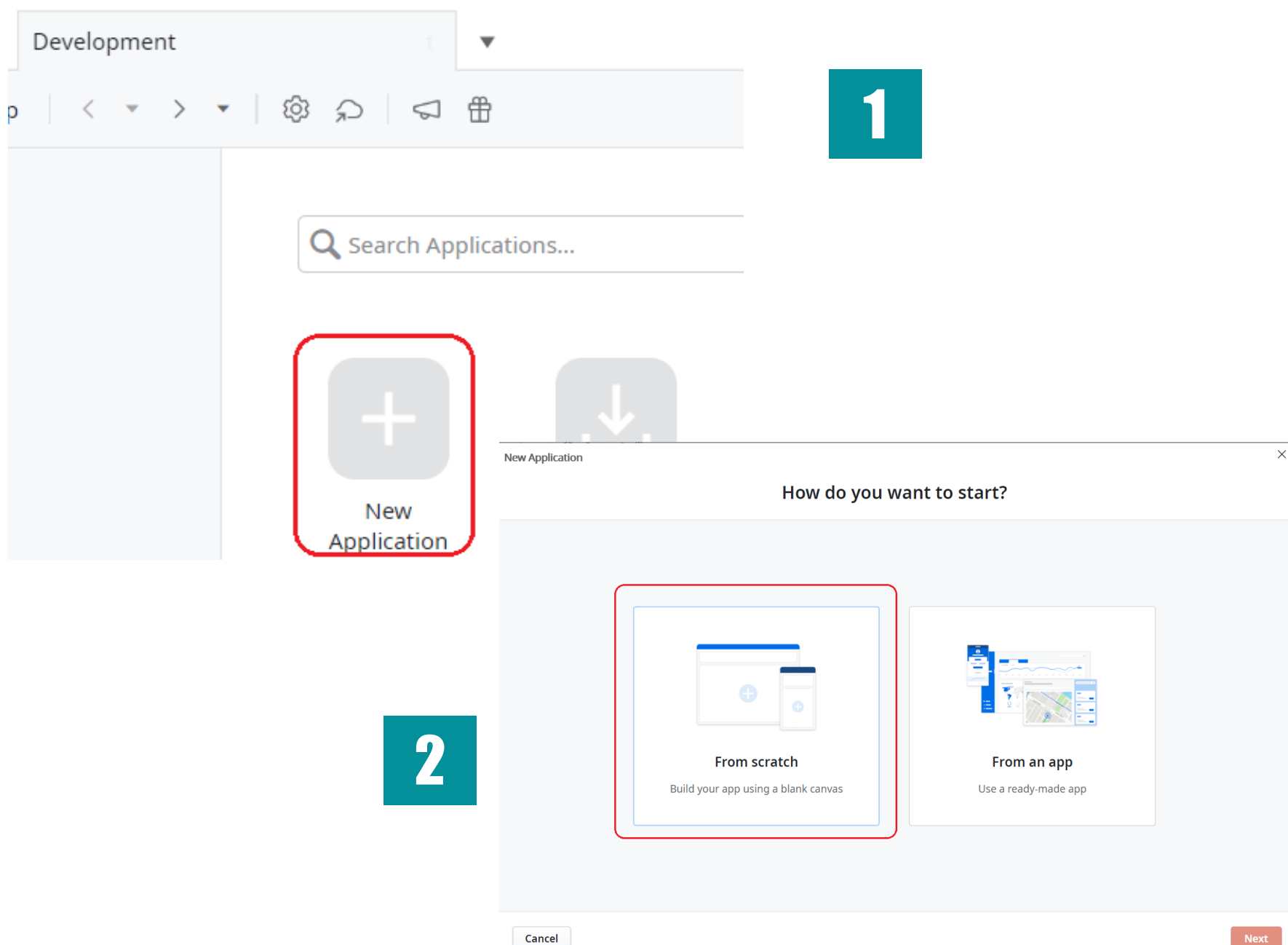
APIs REST COM OUTSYSTEMS

Neste capítulo vamos aprender a consumir APIs REST nas aplicações OutSystems. Você deve ter o Service Studio aberto e conectado a um ambiente OutSystems (por exemplo, personal environment)

APIs REST COM OUTSYSTEMS

Chegou a hora de aprender a consumir APIs REST em aplicações OutSystems! E vamos aprender com um exemplo prático: utilizaremos a SWAPI (Star Wars API) como fonte de dados de uma aplicação que lista todos os personagens da saga.

- Na janela principal do Service Studio, crie uma aplicação Reactive Web App. Vamos chamar a aplicação de *StarWarsExplorer*




APIs REST COM OUTSYSTEMS

3


New Application

×


What are you building?




Reactive Web App
Responsive interface that can be run on all browsers and devices




Tablet App
Optimized for tablets. Create native mobile apps or PWAs




Phone App
Optimized for phones. Create native mobile apps or PWAs



External Web Portal
Responsive interface that can be run on all browsers and devices



Traditional Web
Web and mobile responsive experience focused on server-side development.



Service

BackNext

New Application

×

Fill in your app's basic info



StarWarsExplorer


Description

4

Pick a color to bootstrap your app's interface and icon background

or use a custom icon

Upload icon



BackCreate App



APIs REST COM OUTSYSTEMS

5

[Back to applications](#)



StarWarsExplorer

- Edit
- Delete
- Download
- Convert To Service
- Open In Browser

Develop

Modules

Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

Module name

StarWarsExplorer

Choose module type

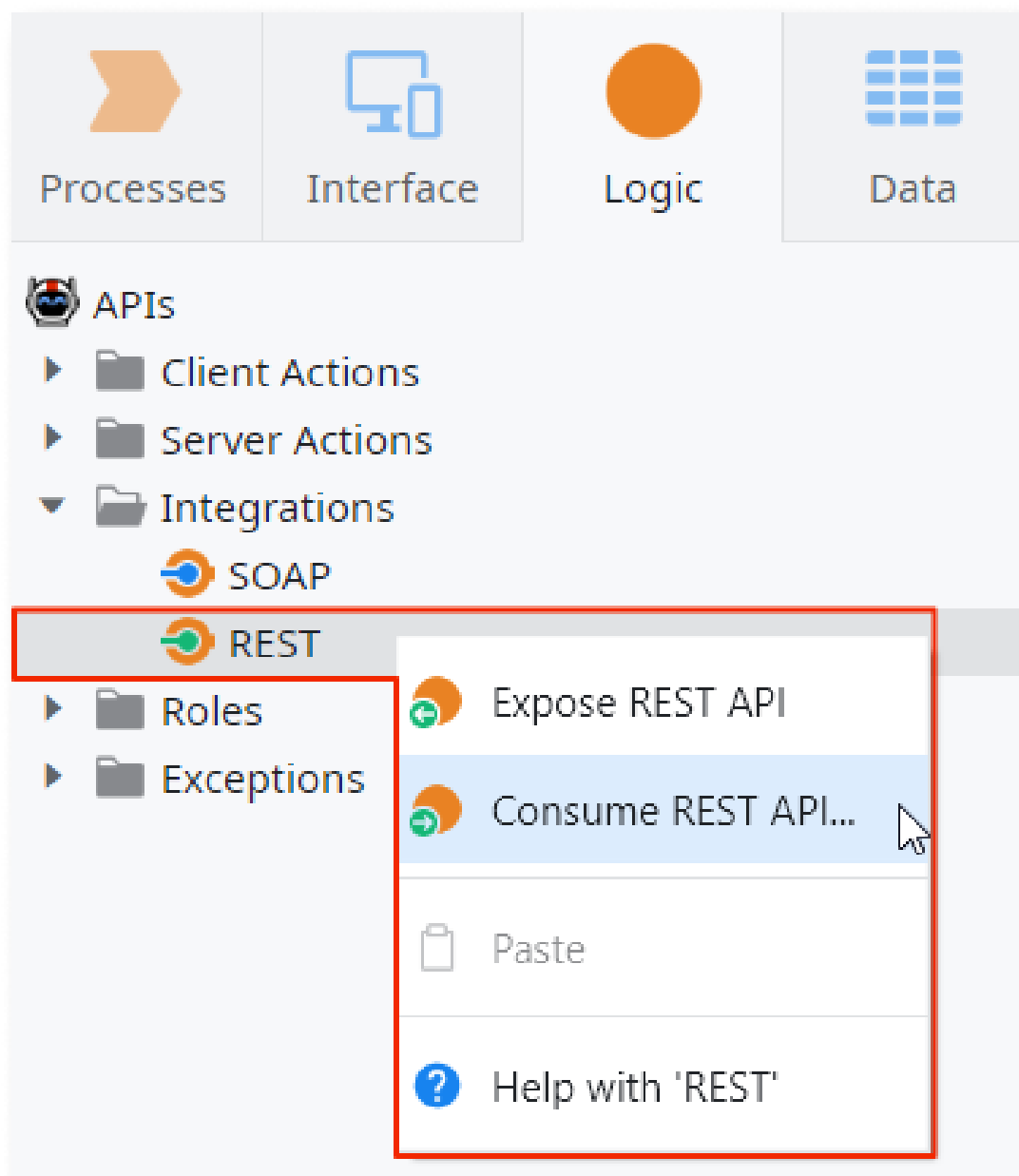
Reactive Web App

Create Module

Cancel

APIs REST COM OUTSYSTEMS

- Na aba Logic, abra a pasta Integrações.
- Clique com o botão direito no elemento REST e selecione Consume REST API...



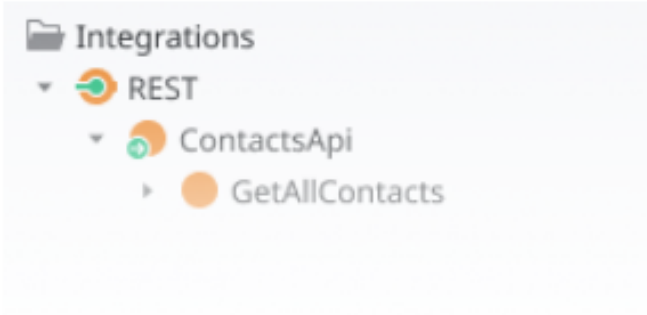
APIs REST COM OUTSYSTEMS

➤ Seleccione *Add single method*

Consume REST API

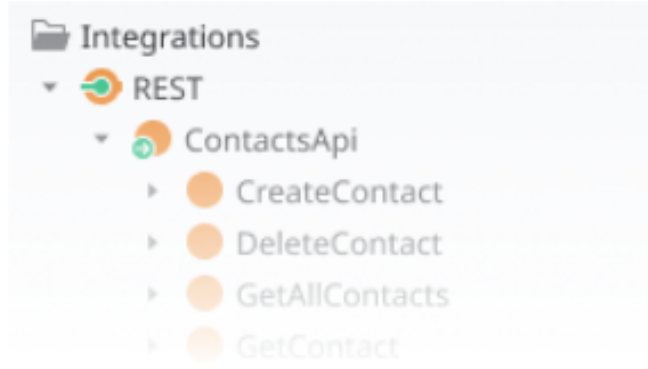


Choose how you want to consume REST API methods:



Add single method

Manually add a single method based on examples.



Add multiple methods

Add selected methods from a REST API exposed by OutSystems.

All REST APIs exposed by OutSystems have a Swagger specification to facilitate consumption.



Continue

Cancel

APIs REST COM OUTSYSTEMS

- Selecione o método GET
- Selecione a aba Test
- Digite a URL da swapi para listar os personagens da série:
<https://swapi.dev/api/people>

The screenshot shows the 'Consume Single API' window in OutSystems. The 'Method URL' section has 'GET' selected and the URL 'https://swapi.dev/api/people' entered. Below this, there are tabs for 'Body', 'Headers and Authentication', and 'Test'. The 'Test' tab is active, showing the 'API request test result' and 'API response test result'. The response is a JSON object with the following structure:

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN
Allow: GET, HEAD, OPTIONS
Strict-Transport-Security: max-age=15768000
Content-Type: application/json
Date: Fri, 10 May 2024 12:44:38 GMT
ETag: "b493126da505af6fec015ec116fec193"
Server: nginx/1.16.1
{
  "count": 82,
  "next": "https://swapi.dev/api/people/?page=2",
  "previous": null,
  "results": [
    {
      "name": "Luke Skywalker",
      "height": 172,
      "mass": 77,
      "hair": "blond",
      "skin": "fair",
      "eyes": "blue",
      "gender": "male",
      "birth_year": "1979-05-04",
      "homeworld": "https://swapi.dev/planets/1/"
    }
  ]
}
```

At the bottom of the 'Test' tab, there is a 'Copy to response body' button. The window also has a 'Finish' button and a 'Cancel' button at the bottom right.

APIs REST COM OUTSYSTEMS

- Cliquem em *Copy to response body*
- Clique em *Finish*

Consume Single API

Method URL

GET

https://swapi.dev/api/people

URL accepts parameters in braces, e.g. https://api.twitter.com/1.1/search/tweets.json?q={query}

Body

Headers and Authentication

Test

Paste a JSON or a text/plain example to generate the response output parameter and structures.
You can also use the Test tab to invoke and get the response from the REST API, and copy it here.

Response

```
{
  "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/",
        "https://swapi.dev/api/films/3/",
        "https://swapi.dev/api/films/6/"
      ],
      "species": [

```

?

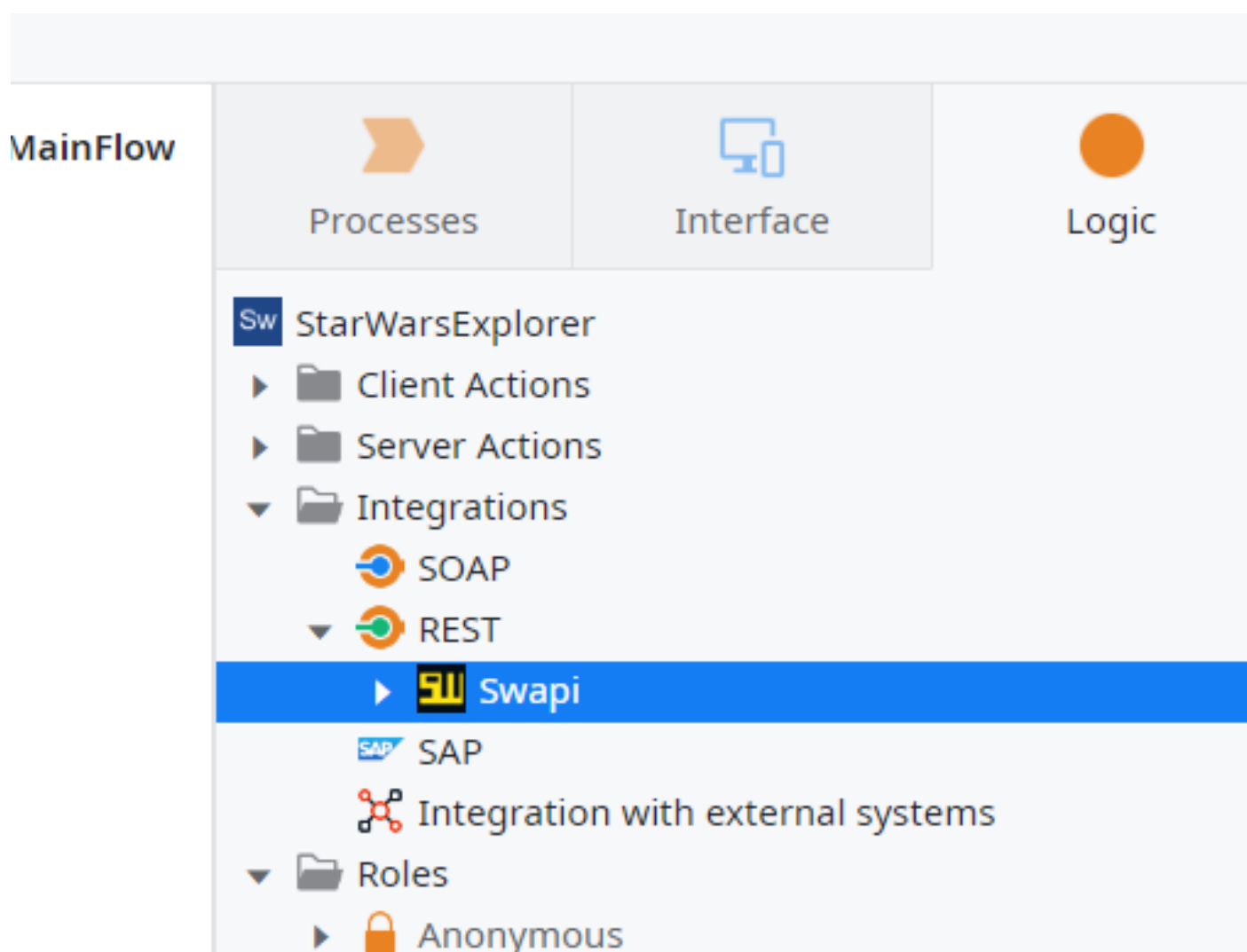
Finish

Cancel

APIs REST COM OUTSYSTEMS

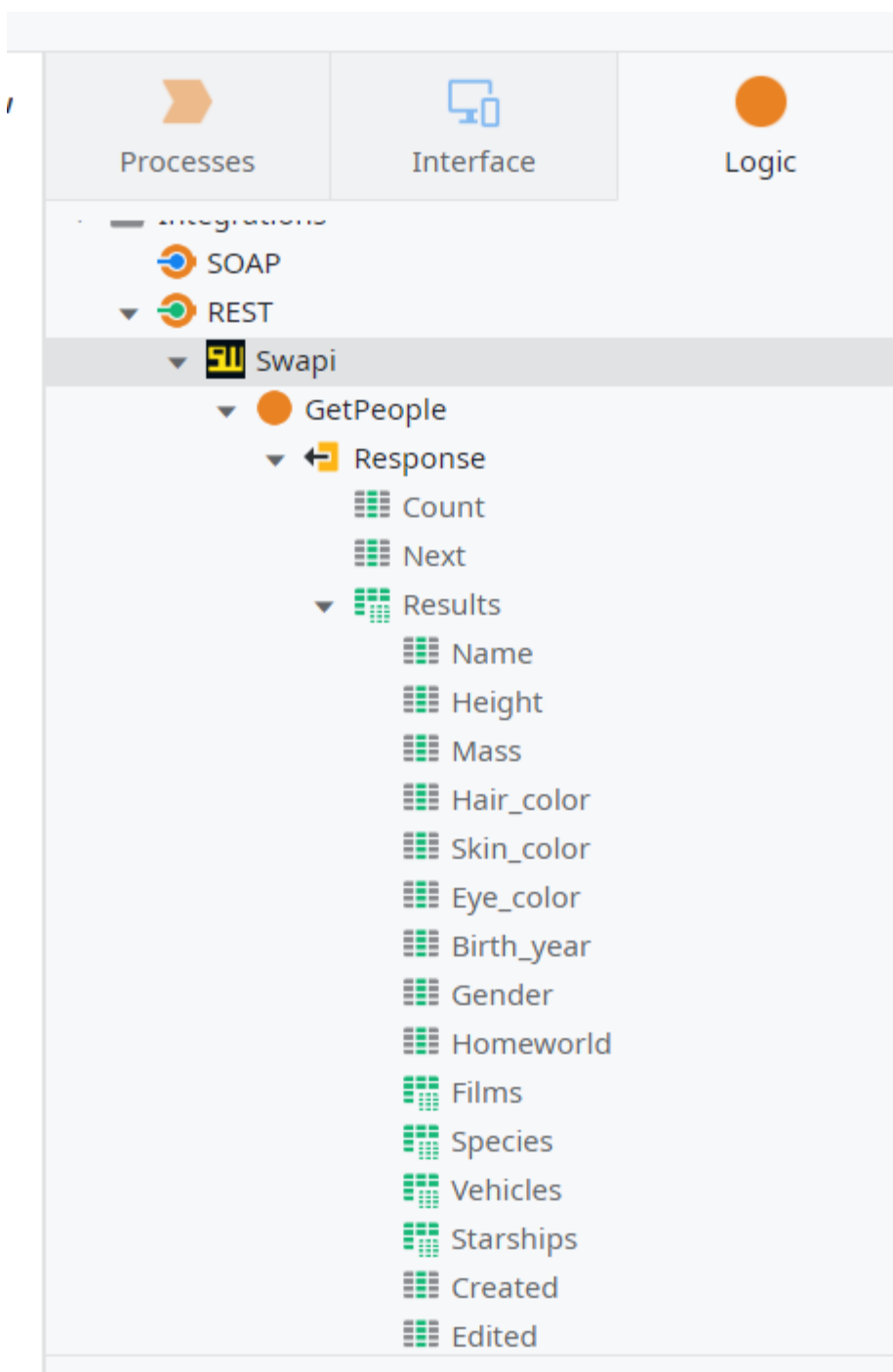
E agora assista ao show do Service Studio!
Olha o que ele faz! Olha o que ele faz!

➤ Cria um elemento REST API



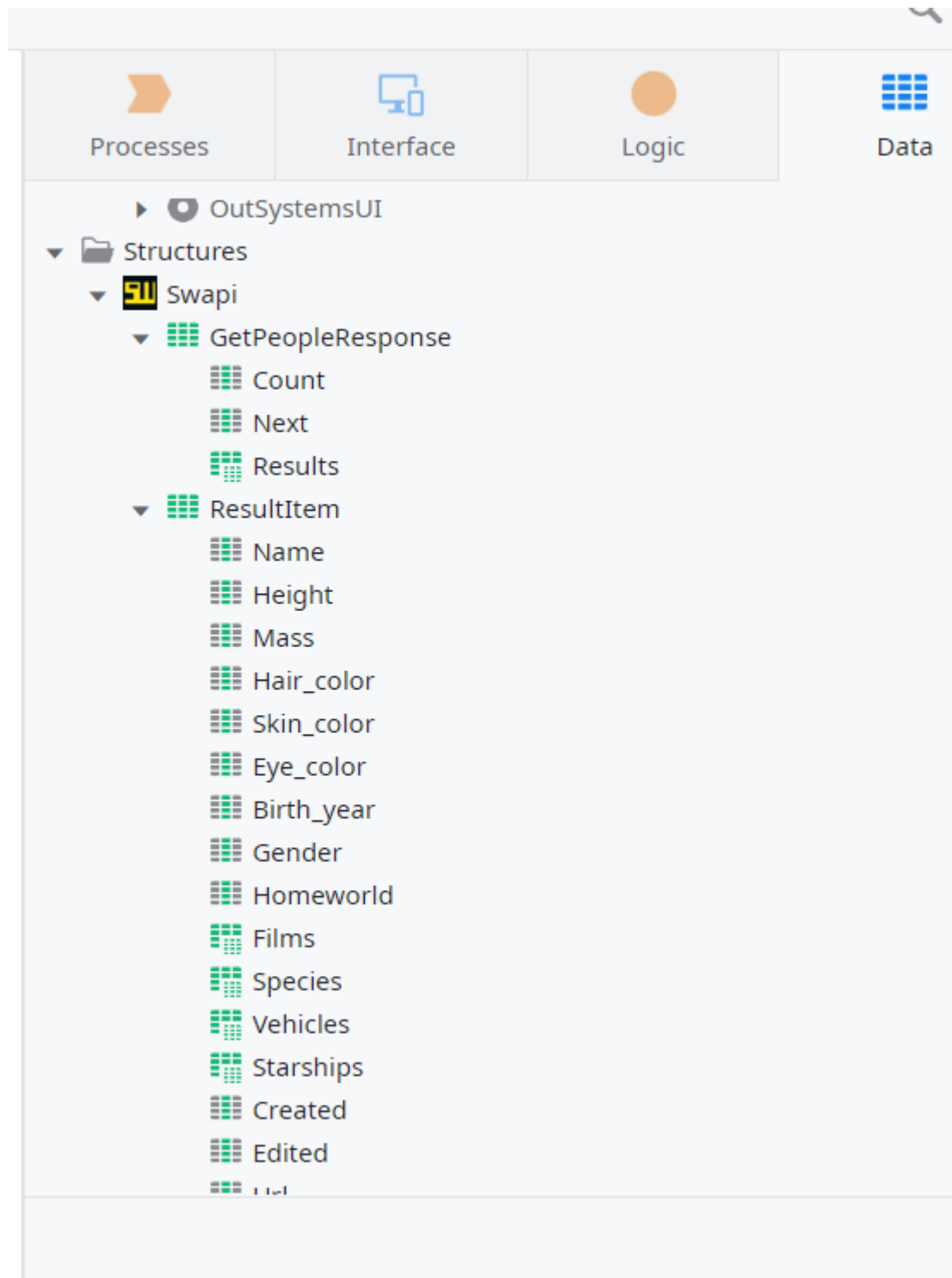
APIs REST COM OUTSYSTEMS

- Cria um método REST API com os parâmetros de entrada e saída correspondentes



APIs REST COM OUTSYSTEMS

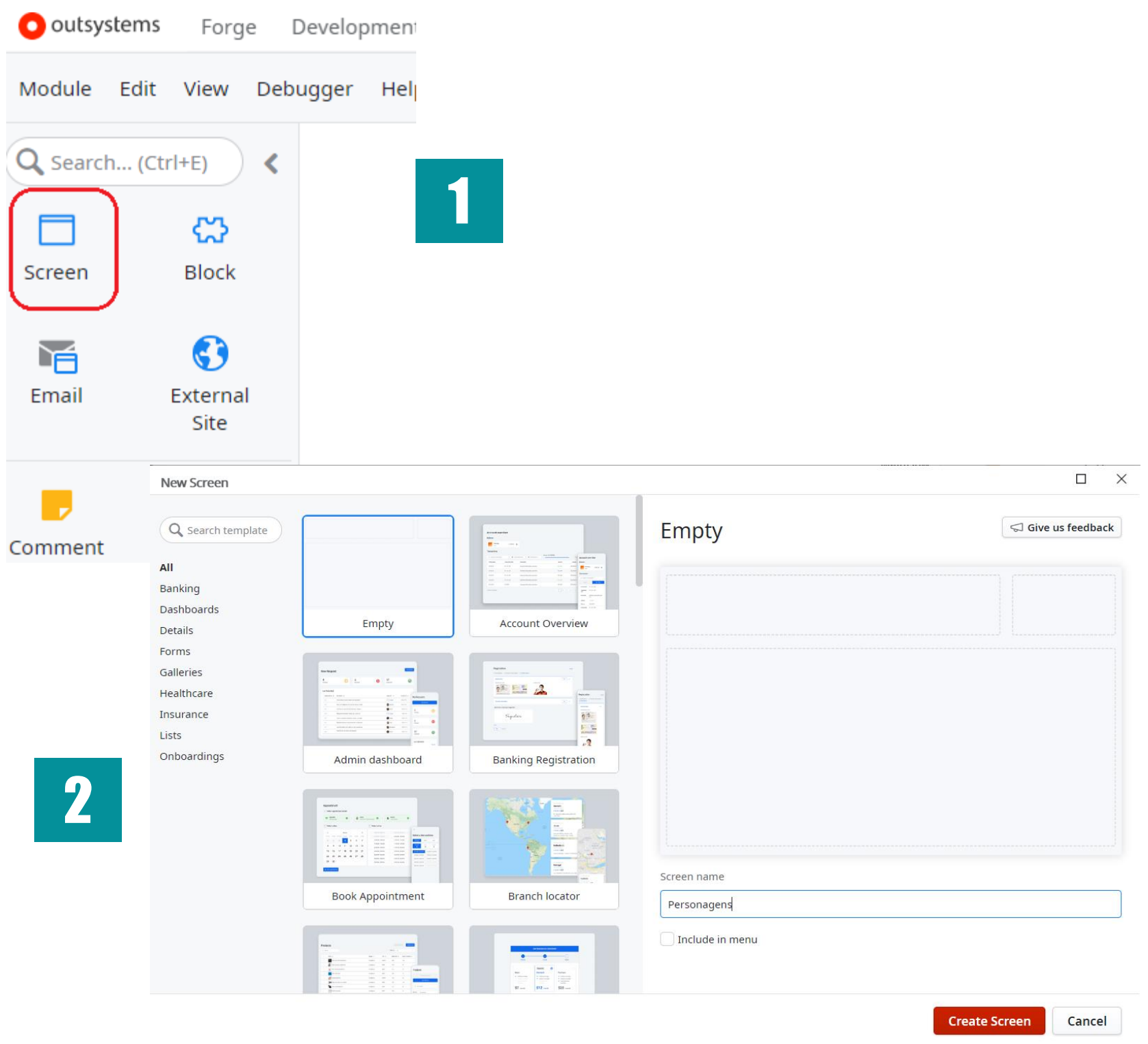
- Mapeia os tipos de dados REST em tipos de dados OutSystems



APIs REST COM OUTSYSTEMS

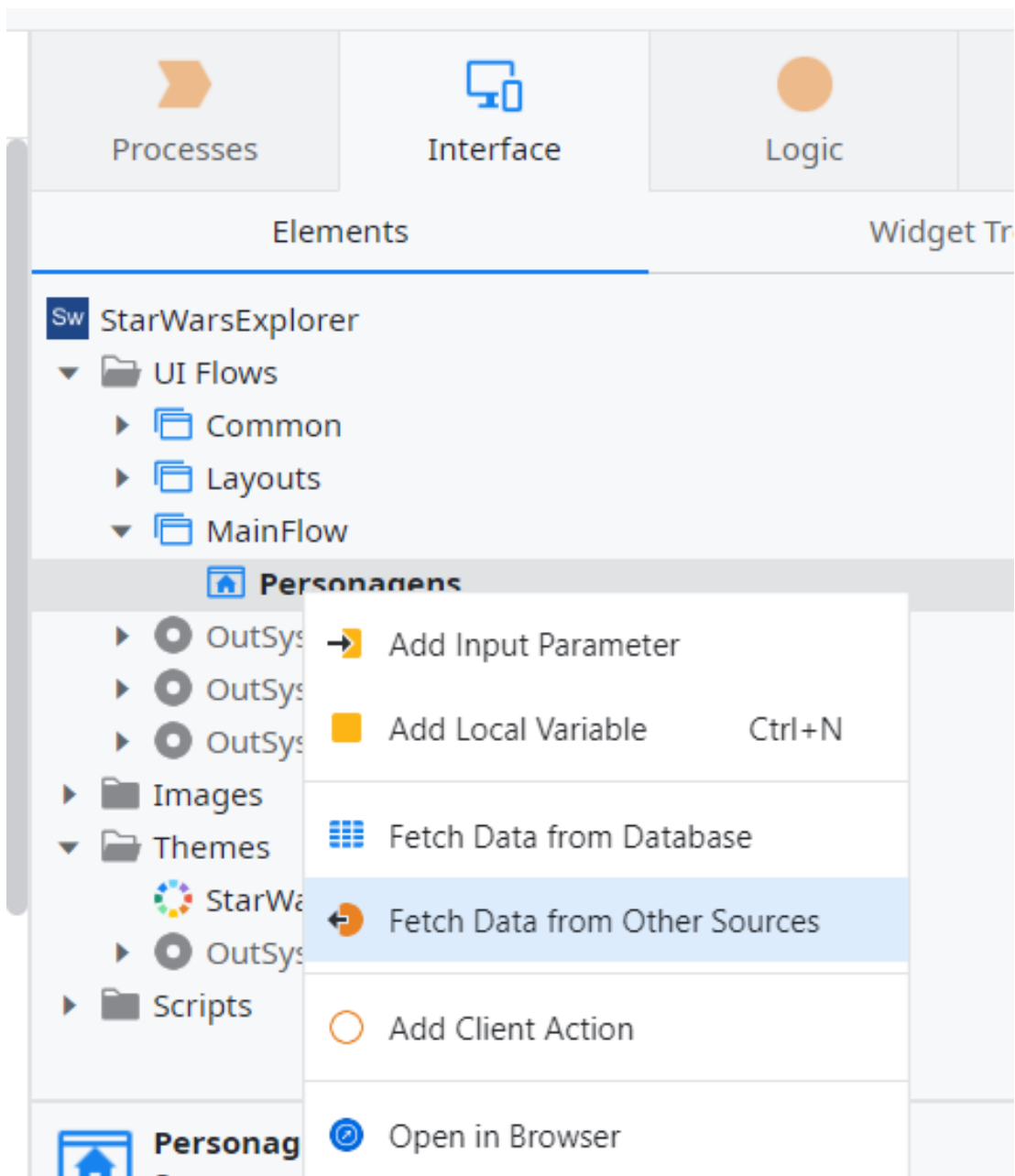
Próximo passo: usar a estrutura que o Service Studio montou para criar sua tela

➤ Crie uma tela *Personagens*



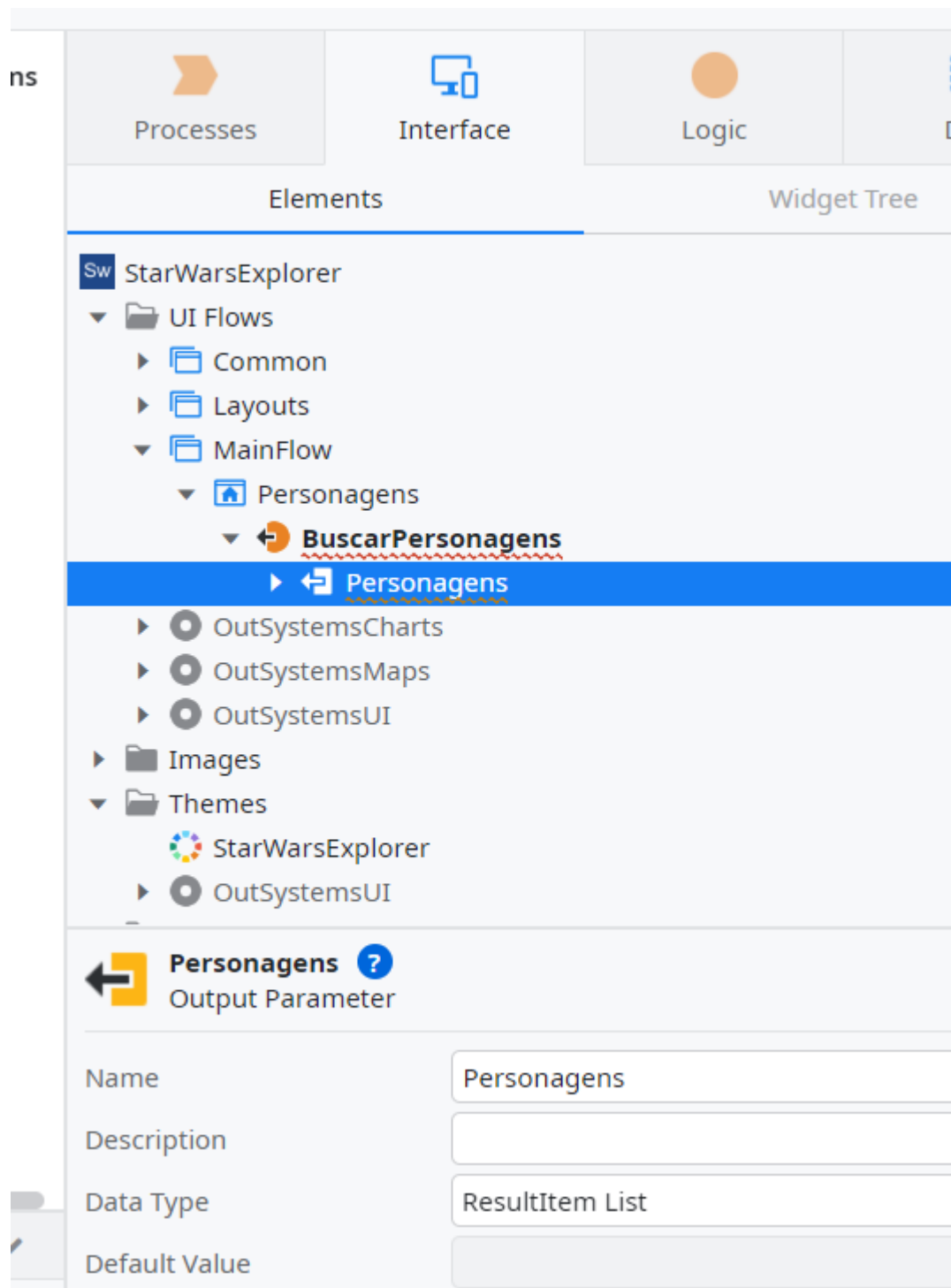
APIs REST COM OUTSYSTEMS

- Clique com o botão direito em Personagens
- Clique em Fetch Data from other Sources



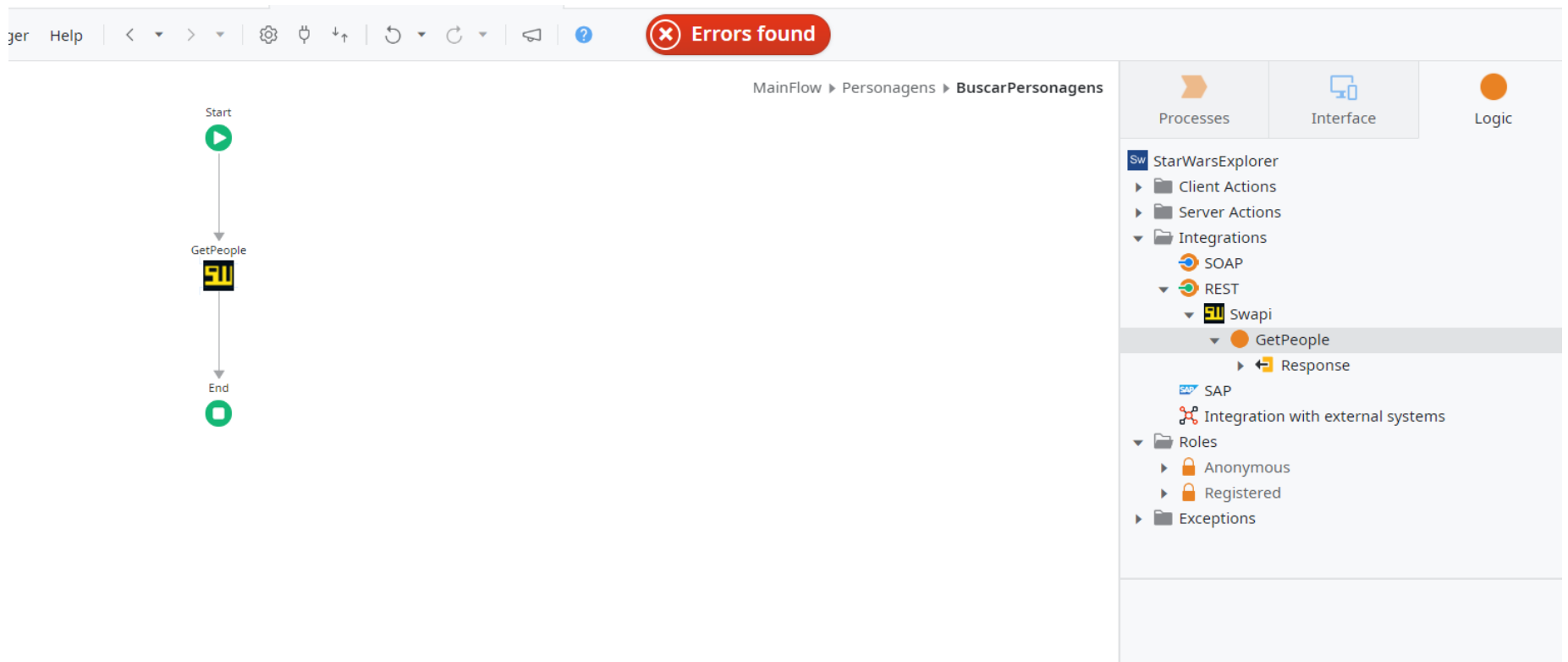
APIs REST COM OUTSYSTEMS

- Chame a action de *BuscarPersonagens*
- Configure o parâmetro de saída como uma lista de *ResultItem*



APIs REST COM OUTSYSTEMS

- Dentro da action BuscarPersonagens, chame o método da Swapi que obtém os personagens



APIs REST COM OUTSYSTEMS

- Atribua à variável Personagens o valor da saída da Swapi

1

Run Server Action

SQL

Switch

Assign

Excel To Record List

Start

GetPeople

Personagens

Personagens ?

Assign

Label

Assignments

Personagens

x.y = GetPeople.Response.Results

Variable

x.y = Value

2

10:18

10:18

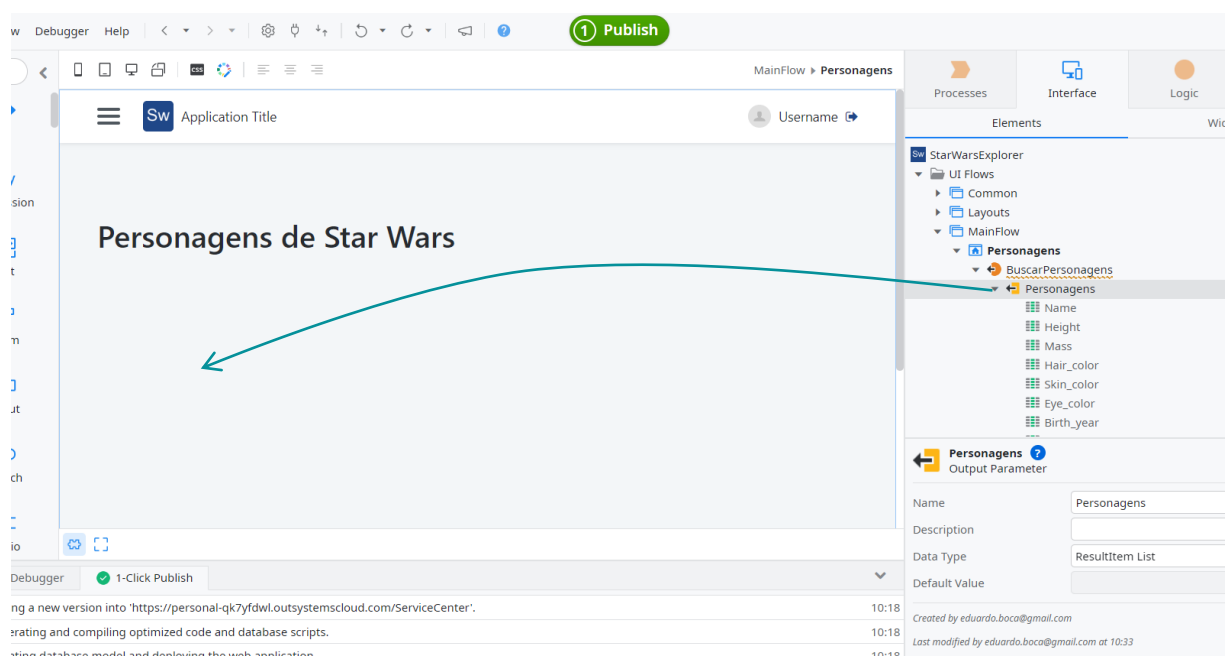
10:18

10:18

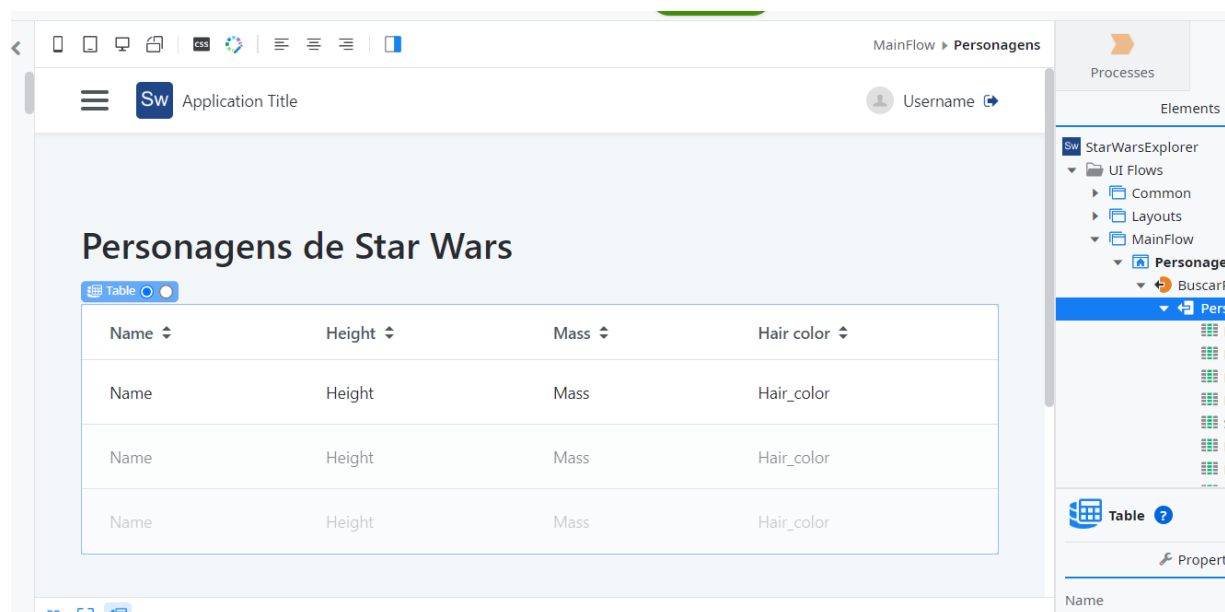
APIs REST COM OUTSYSTEMS

- Volte à tela de Personagens
- Dê o título *Personagens de Star Wars*
- Arraste a variável *Personagens* para a tela

1



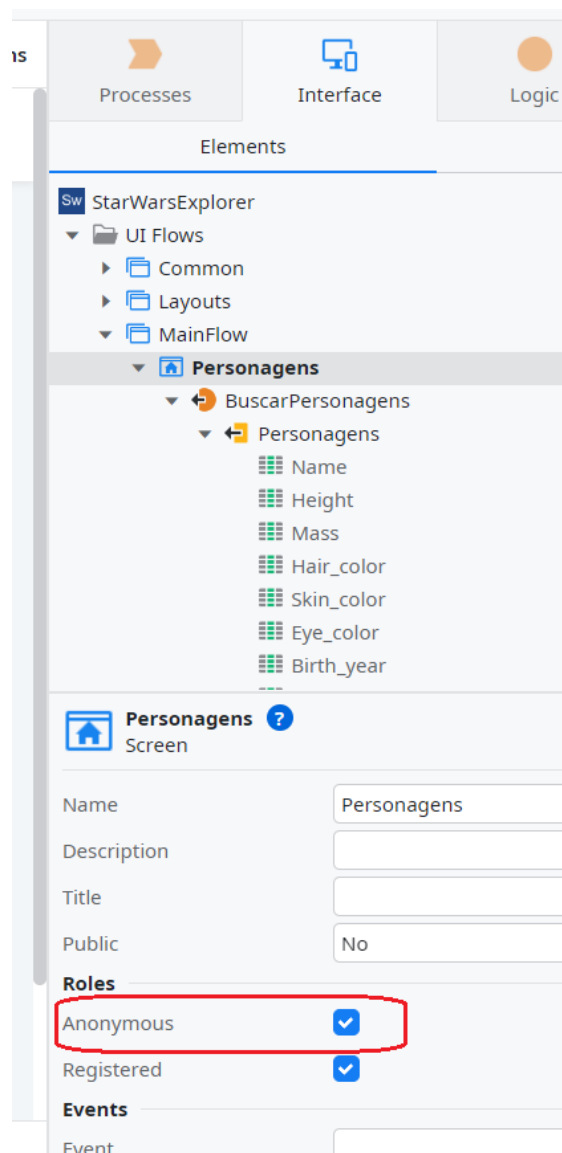
2



APIs REST COM OUTSYSTEMS

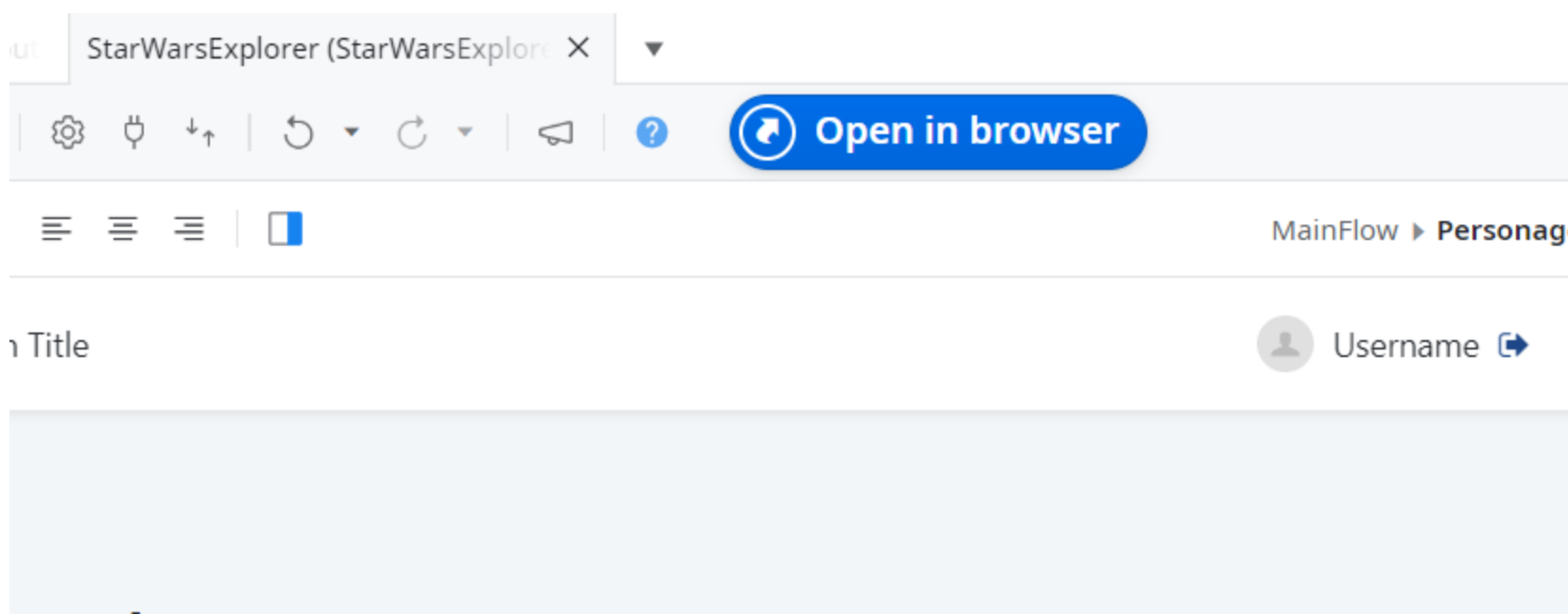
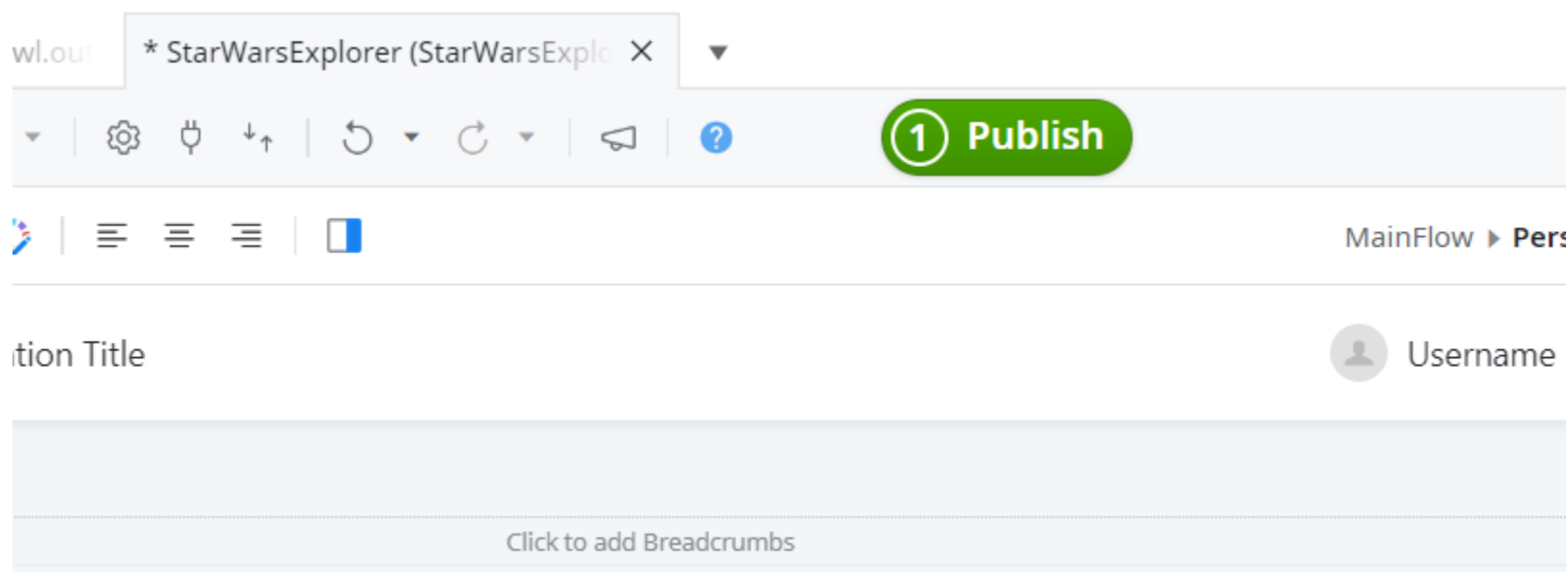
Quase lá...

- Configure a tela para Aceitar qualquer usuário



APIs REST COM OUTSYSTEMS

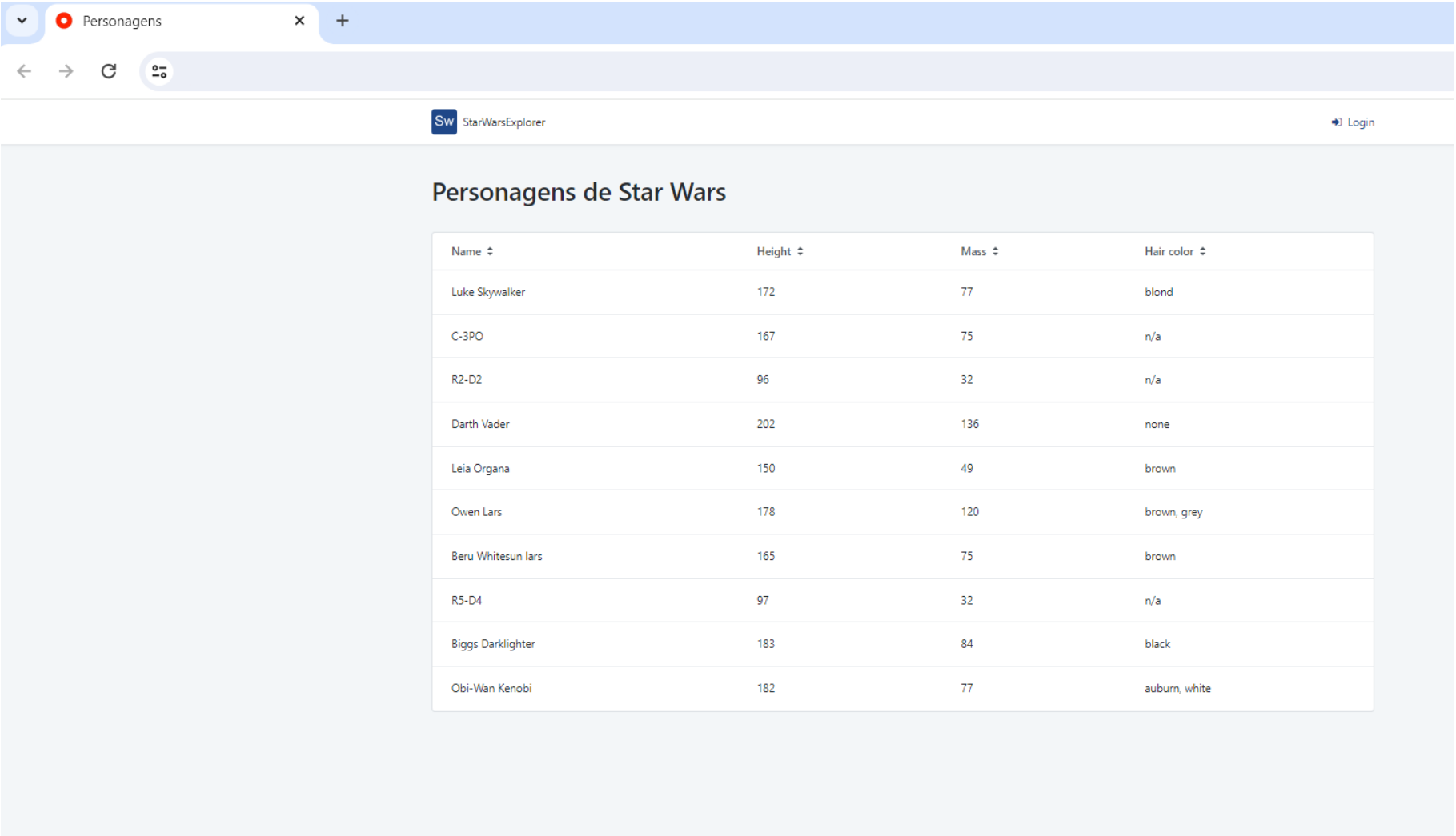
- Clique em Publish
- Clique em Open in Browser





APIs REST COM OUTSYSTEMS

E sua aplicação está pronta!



CONCLUSÃO



OBRIGADO!

Esse Ebook foi gerado por IA, e diagramado por humano.
O passo a passo se encontra no meu Github.



<https://github.com/eduardoboca/prompts-recipe-to-create-a-ebook>

