



# Especificação OpenAPI / Swagger

Unidade Curricular de Aplicações Distribuídas  
Mestrado em Desenvolvimento de Software e Sistemas Interactivos

[Versão 1.0 (abril de 2021)]

Alexandre Fonte  
([adf@ipcb.pt](mailto:adf@ipcb.pt))



Instituto Politécnico de Castelo Branco  
Escola Superior de Tecnologia



# Resumo

---

- Especificação OpenAPI
- Estrutura Básica



# Especificação OpenAPI

- A especificação OpenAPI (anteriormente Especificação Swagger Specification) é um formato de descrição de API para APIs REST.
- Um ficheiro OpenAPI permite-lhe descrever toda a sua API, incluindo:
  - EndPoints (/utilizadores) e operações disponíveis em cada endpoint (GET /utilizadores, POST /utilizadores)
  - Parâmetros de Entrada e Saída para cada operação
  - Métodos de autenticação
  - Informações de contacto, licença, termos de utilização e outras informações.
- As especificações OpenAPI podem ser escritas em YAML ou JSON.
- A especificação completa do OpenAPI pode ser encontrada em GitHub ou no site openapis.org:
  - <https://github.com/OAI/OpenAPI-Specification/>
  - <https://spec.openapis.org/oas/v3.1.0>



# Especificação OpenAPI

- A última versão foi lançada em Fevereiro de 2021.
  - <https://www.openapis.org>

The screenshot shows the OpenAPI Initiative website. At the top, the OpenAPI logo is on the left, and a navigation menu with links like 'About', 'Specification', 'Participate', 'Governance', 'Membership', 'Blog', and 'FAQ' is on the right. Social media icons for Twitter, LinkedIn, and GitHub are also present. The main banner features the text 'Something great just got better, get excited!' on the left, a large '3.1.0 RELEASE' in the center with a circular progress indicator, and 'Compatible with JSON Schema' on the right. Below this, there are two green buttons: 'REVIEW THE SPEC' and 'GETTING STARTED'. A secondary banner for 'ASC 2021 API SPECIFICATIONS CONFERENCE' is below that, with dates 'September 28-29 | Virtual Experience' and a 'Get the Latest ASC Info' button. At the bottom, there are two dark bars: 'How to contribute to the OAS' and 'Submit an issue on GitHub'. The footer contains logos for 'BASSADOR LABS', 'Checkly', 'Stoplight', 'Timewarp', and 'Intento', along with a 'Join The Growing List of OAI Members' button.

**OPENAPI INITIATIVE**

About Specification Participate Governance Membership Blog FAQ

Something great just got better, get excited!

**3.1.0**

**RELEASE**

Compatible with JSON Schema

REVIEW THE SPEC GETTING STARTED

**ASC 2021**

API SPECIFICATIONS CONFERENCE

September 28-29 | Virtual Experience

Get the Latest ASC Info

How to contribute to the OAS

Submit an issue on GitHub

**BASSADOR LABS** Stoplight Timewarp Intento

Join The Growing List of OAI Members



# Geração/Edição da Especificação OpenAPI no WSO2 API Manager

- No menu lateral aceder à Definição da API
  - Download/Editar

[BACK TO APIs](#)

PizzaShack :1.0.0  
Created by: admin

PUBLISHED  
State

Go To

View in Dev Portal

Create New Version

Delete

API Definition EDIT IMPORT DEFINITION DOWNLOAD DEFINITION CONVERT TO JSON Last updated: 3 hours ago

```
1 ---
2 openapi: "3.0.1"
3 info:
4   title: "PizzaShack"
5   version: "1.0.0"
6 servers:
7   -
8     url: "/"
9 security:
10  -
11    default: []
12 paths:
13   /*:
14     get:
15       responses:
16         200:
17           description: "OK"
18       security:
19         -
20           default: []
21       x-auth-type: "Application & Application User"
22       x-throttling-tier: "Unlimited"
23     put:
24       responses:
25         200:
26           description: "OK"
27       security:
28         -
29           default: []
30       x-auth-type: "Application & Application User"
31       x-throttling-tier: "Unlimited"
32     post:
33       responses:
34         200:
35           description: "OK"
36       security:
37         -
38           default: []
39       x-auth-type: "Application & Application User"
40       x-throttling-tier: "Unlimited"
41     delete:
42       responses:
43         200:
```

Alexar

034



# Edição da Especificação OpenAPI usando o Swagger Editor

## ■ Swagger Editor

- Link: <https://swagger.io/tools/swagger-editor/>

Swagger Editor

File Edit Generate Server Generate Client Switch back to previous editor

```
1 swagger: "2.0"
2 info:
3   description: "This is a sample server Petstore server. You can find
4     out more about Swagger at [http://swagger.io](http://swagger.io)
5     or on [irc.freenode.net, #swagger](http://swagger.io/irc/). For
6     this sample, you can use the api key `special-key` to test the
7     authorization filters."
8   version: "1.0.0"
9   title: "Swagger Petstore"
10  termsOfService: "http://swagger.io/terms/"
11  contact:
12    email: "apiteam@swagger.io"
13  license:
14    name: "Apache 2.0"
15    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
16 host: "petstore.swagger.io"
17 basePath: "/v2"
18 tags:
19 - name: "pet"
20   description: "Everything about your Pets"
21   externalDocs:
22     description: "Find out more"
23     url: "http://swagger.io"
24 - name: "store"
25   description: "Access to Petstore orders"
26 - name: "user"
27   description: "Operations about user"
28   externalDocs:
29     description: "Find out more about our store"
30     url: "http://swagger.io"
31 schemes:
32 - "http"
33 paths:
34   /pet:
35     post:
36       tags:
37       - "pet"
38       summary: "Add a new pet to the store"
```

Swagger Petstore 1.0.0

[ Base URL: petstore.swagger.io/v2 ]

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net](http://irc.freenode.net), [#swagger](http://irc.freenode.net). For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)

[Contact the developer](#)

[Apache 2.0](#)

[Find out more about Swagger](#)

Schemes

HTTP

Authorize

pet

 Everything about your Pets Find out more: <http://swagger.io>

POST

 /pet Add a new pet to the store

PUT

 /pet Update an existing pet



# Estrutura Básica da Especificação OpenAPI

```
1. openapi: 3.0.0
2. info:
3.   title: Sample API
4.   description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
5.   version: 0.1.9
6.
7. servers:
8.   - url: http://api.example.com/v1
9.     description: Optional server description, e.g. Main (production) server
10.  - url: http://staging-api.example.com
11.    description: Optional server description, e.g. Internal staging server for testing
12.
13. paths:
14.   /users:
15.     get:
16.       summary: Returns a list of users.
17.       description: Optional extended description in CommonMark or HTML.
18.       responses:
19.         '200': # status code
20.           description: A JSON array of user names
21.           content:
22.             application/json:
23.               schema:
24.                 type: array
25.                 items:
26.                   type: string
```





# Estrutura Básica

## *Metadados*

- Versão da Especificação OpenAPI utilizada na especificação da API

```
1. openapi: 3.0.0
```

- Informações sobre a API
  - Nome (title)
  - Descrição – É opcional; pode múltiplas linhas ou uma única linha. Pode-se usar a linguagem CommonMark (ver: <https://commonmark.org/help/> )
  - Versão da API

```
1. info:  
2.   title: Sample API  
3.   description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.  
4.   version: 0.1.9
```





# Estrutura Básica

## *Servidores da API*

### ■ Servidores

- Especifica o(s) servidor(es) da API e o URL base.
- Podem ser definidos servidores da API em produção e de testes (sandbox).
- Todos os *API paths* são “relativos” ao URL base.
- Exemplo: a path /users
  - É equivalente a `http://api.example.com/v1/users` ou `http://staging-api.example.com/users`, dependendo do servidor que está a ser utilizado.

```
1. servers:
2.   - url: http://api.example.com/v1
3.     description: Optional server description, e.g. Main (production) server
4.   - url: http://staging-api.example.com
5.     description: Optional server description, e.g. Internal staging server for testing
```



# Estrutura Básica

## *Paths*

### ■ Paths

- Define os endpoints individuais (paths) da API;
- Os métodos HTTP (operações) suportadas por esses endpoints
- Exemplo: HTTP GET /users:

```
1. paths:
2.   /users:
3.     get:
4.       summary: Returns a list of users.
5.       description: Optional extended description in CommonMark or HTML
6.       responses:
7.         '200':
8.           description: A JSON array of user names
9.           content:
10.            application/json:
11.              schema:
12.                type: array
13.                items:
14.                  type: string
```



# Estrutura Básica

## *Parâmetros*

### ■ Parâmetros

- As Operações podem ter parâmetros passados como parâmetros na path (/users/{userId}), query strings (/users?role=admin), cabeçalhos HTTP (X-CustomHeader: Value), ou cookies (Cookie: debug=0).
- Podem ser definidos os tipos de dados, se são opcionais ou obrigatórios.

```
1. paths:
2.   /users/{userId}:
3.     get:
4.       summary: Returns a user by ID.
5.       parameters:
6.         - name: userId
7.           in: path
8.           required: true
9.           description: Parameter description in CommonMark or HTML.
10.          schema:
11.            type: integer
12.            format: int64
13.            minimum: 1
14.       responses:
15.         '200':
16.           description: OK
```



# Estrutura Básica

## *Body dos Pedidos (POST, PUT,...)*

- Body dos Pedidos

- Usa-se a palavra chave requestBody para descrever o conteúdo do body e o media type usado (ex. Application/json).

```
1. paths:
2.   /users:
3.     post:
4.       summary: Creates a user.
5.       requestBody:
6.         required: true
7.         content:
8.           application/json:
9.             schema:
10.              type: object
11.              properties:
12.                username:
13.                  type: string
14.       responses:
15.         '201':
16.           description: Created
```



# Estrutura Básica

## *Respostas da API*

### ■ Respostas aos Pedidos

- Para cada operação, podem ser definidos diferentes códigos de estado (200 OK ou 404 Not Found) e o esquema do body.
- O esquema do body especifica a estrutura/modelo do documento JSON, por exemplo. Estes podem ser referenciados para um apontador **\$ref**

```
4. responses:
5.   '200':
6.     description: A user object.
7.     content:
8.       application/json:
9.         schema:
10.          type: object
11.          properties:
12.            id:
13.              type: integer
14.              format: int64
15.              example: 4
16.            name:
17.              type: string
18.              example: Jessica Smith
19.   '400':
20.     description: The specified user ID is invalid (not a number).
21.   '404':
22.     description: A user with the specified ID was not found.
23.   default:
24.     description: Unexpected error
```



# Estrutura Básica

## *Modelos dos dados da API*

- Modelos dos dados
  - Exemplo modelo do document JSON

```
1. {  
2.   "id": 4,  
3.   "name": "Arthur Dent"  
4. }
```



```
1. components:  
2.   schemas:  
3.     User:  
4.       properties:  
5.         id:  
6.           type: integer  
7.         name:  
8.           type: string  
9.         # Both properties are required  
10.        required:  
11.          - id  
12.          - name
```



# Estrutura Básica

## *Autenticação*

- Tipos de Autenticação suportados pela API
  - Utilizam-se as palavras chave `securitySchemes` e `security` para indicar os métodos de autenticação.
    - Básica ou Bearer
    - OAuth
    - API-Key

```
1. components:
2.   securitySchemes:
3.     BasicAuth:
4.       type: http
5.       scheme: basic
6.
7. security:
8.   - BasicAuth: [ ]
```





---

FIM

Unidade Curricular de Aplicações Distribuídas  
Mestrado em Desenvolvimento de Software e  
Sistemas Interactivos  
Versão 1.0 (maio de 2020)