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)

Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

Resumo

- Especificação OpenAPI
- Estrutura Básica

Especificação OpenAPI

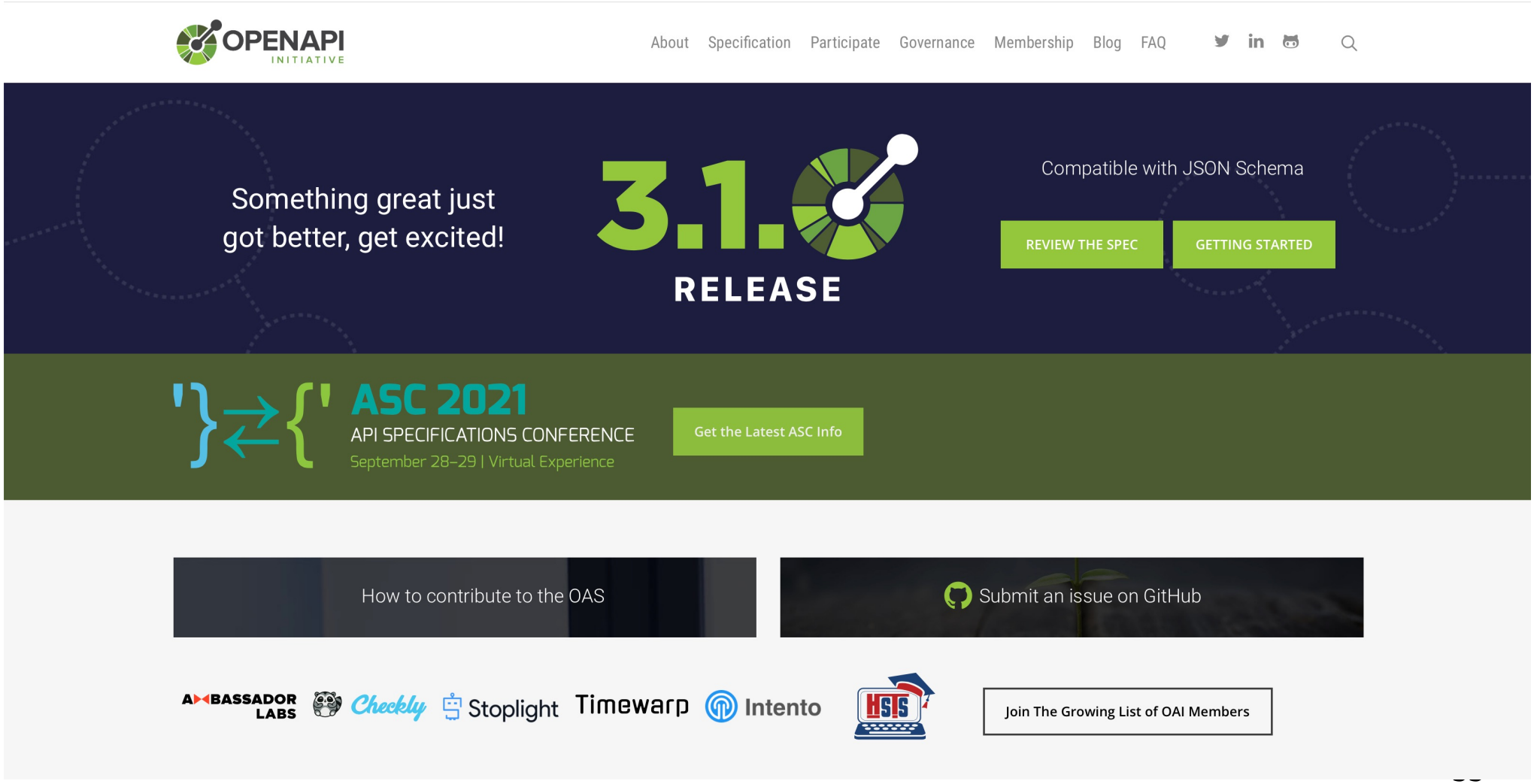
- A especificação OpenAPI (anteriormente Especificação Swagger Specification) é um formato de descrição de API para APIs REST.

- 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

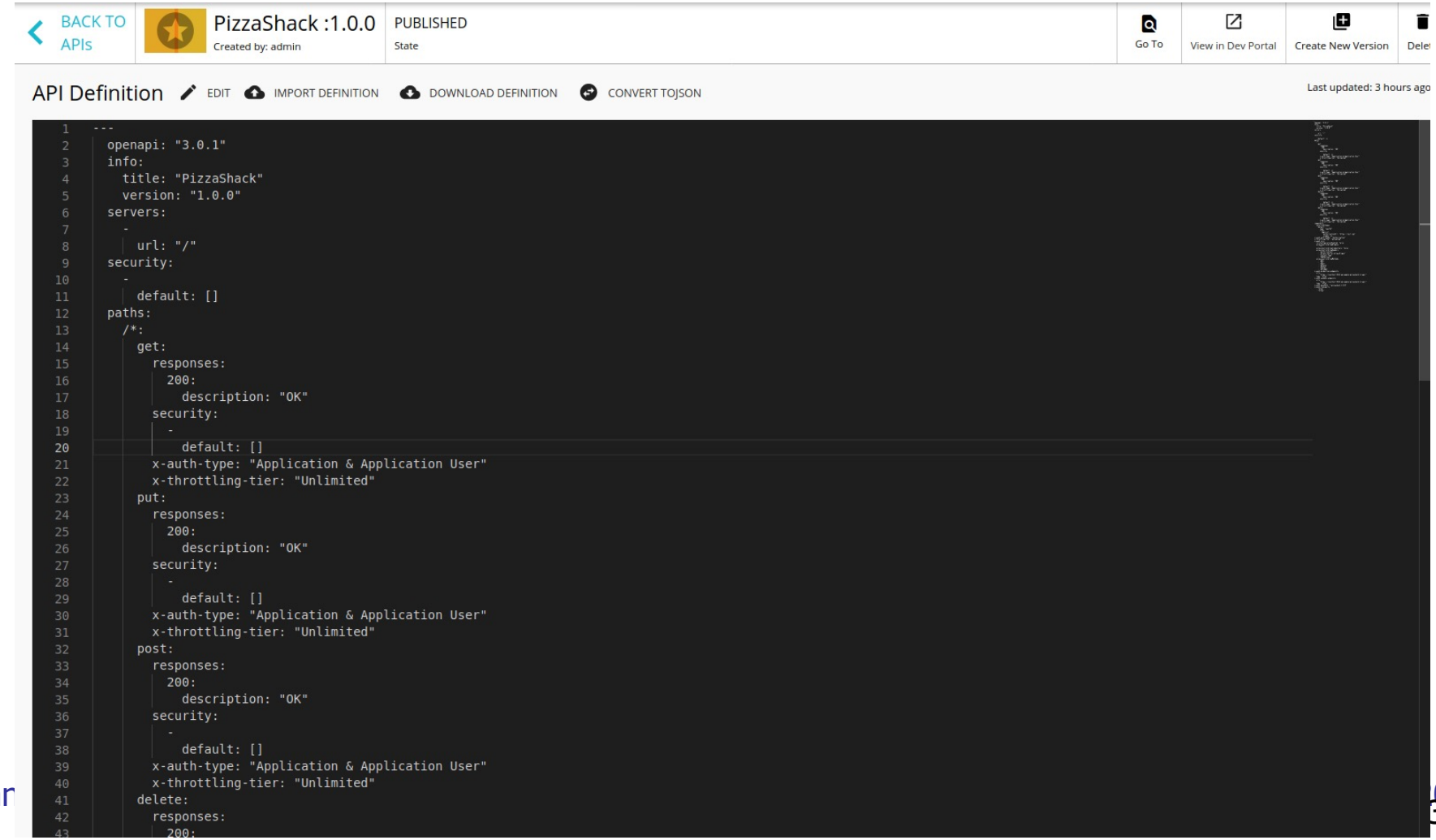
Especificação OpenAPI

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



Geração/Edição da Especificação OpenAPI no WSO2 API Manager

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



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

```yaml
swagger: "2.0"
info:
  description: "This is a sample server Petstore server.  You can find
    out more about     Swagger at [http://swagger.io](http://swagger.io)
    or on [irc.freenode.net, #swagger](http://swagger.io/irc/).      For
    this sample, you can use the api key `special-key` to test the
    authorization     filters."
  version: "1.0.0"
  title: "Swagger Petstore"
  termsOfService: "http://swagger.io/terms/"
  contact:
    email: "apiteam@swagger.io"
  license:
    name: "Apache 2.0"
    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
host: "petstore.swagger.io"
basePath: "/v2"
tags:
- name: "pet"
  description: "Everything about your Pets"
  externalDocs:
    description: "Find out more"
    url: "http://swagger.io"
- name: "store"
  description: "Access to Petstore orders"
- name: "user"
  description: "Operations about user"
  externalDocs:
    description: "Find out more about our store"
    url: "http://swagger.io"
schemes:
- "http"
paths:
  /pet:
    post:
      tags:
      - "pet"
      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, #swagger. 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

```yaml
openapi: 3.0.0
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9

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

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':    # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

Estrutura Básica
Metadados

- Versão da Especificação OpenAPI utilizada na especificação da API

```yaml
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

```yaml
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  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.

```
servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
  - url: http://staging-api.example.com
    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:

```yaml
paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML
      responses:
        '200':
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  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.

```
paths:
  /users/{userId}:
    get:
      summary: Returns a user by ID.
      parameters:
        - name: userId
          in: path
          required: true
          description: Parameter description in CommonMark or HTML.
          schema:
            type : integer
            format: int64
            minimum: 1
      responses:
        '200':
          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).

```yaml
paths:
  /users:
    post:
      summary: Creates a user.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                username:
                  type: string
      responses:
        '201':
          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**

```
      responses:
        '200':
          description: A user object.
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                    format: int64
                    example: 4
                  name:
                    type: string
                    example: Jessica Smith
        '400':
          description: The specified user ID is invalid (not a number).
        '404':
          description: A user with the specified ID was not found.
        default:
          description: Unexpected error
```

Estrutura Básica
Modelos dos dados da API

- Modelos dos dados
  - Exemplo modelo do document JSON

```json
{
  "id": 4,
  "name": "Arthur Dent"
}
```

⇔

```yaml
components:
  schemas:
    User:
      properties:
        id:
          type: integer
        name:
          type: string
      # Both properties are required
      required:
        - id
        - 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

```
components:
  securitySchemes:
    BasicAuth:
      type: http
      scheme: basic

security:
  - BasicAuth: []
```

FIM

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