



OPENAPI

Alessandro Bocci
name.surname@unipi.it

Advanced Software Engineering (Lab)
08/10/2024

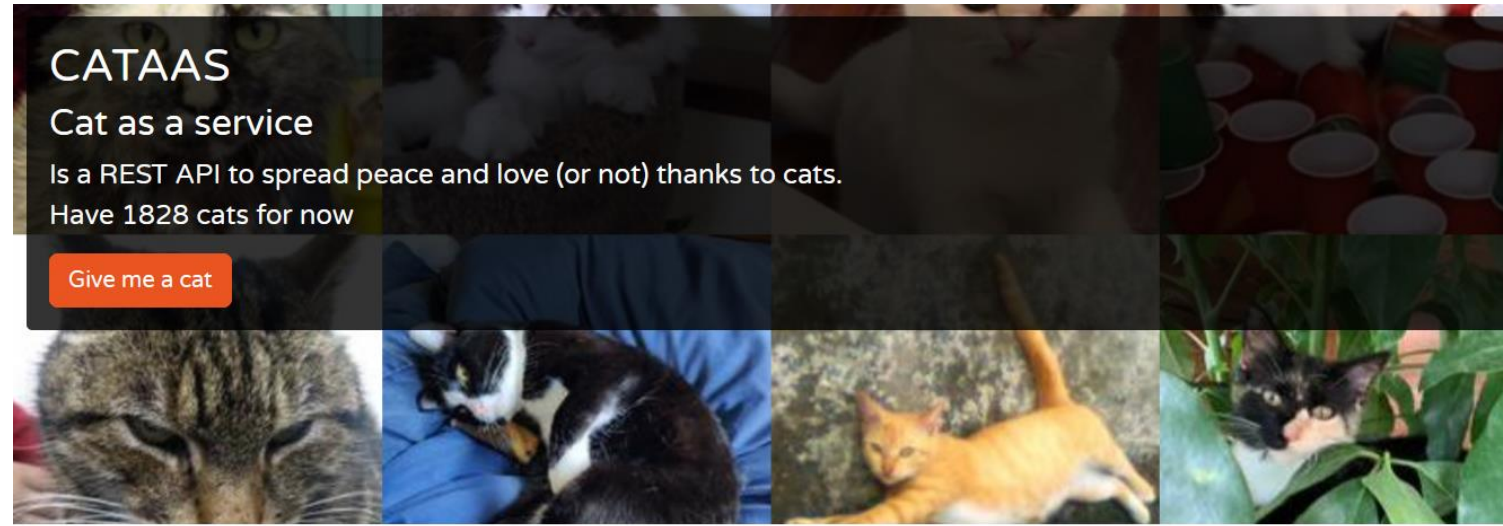
OpenAPI



Specification for a machine-readable interface oriented to RESTful API.

- Useful tools: Swagger Editor (online), Swagger API (app), Swagger Hub (needs registration)
- It uses either YAML or JSON.

Do you remember CATAAS?



Hey ! Do you like Cataas? Do you want to support the project ?

Buy me a beer 🍺

Hey cat lovers, new major version of cataas :

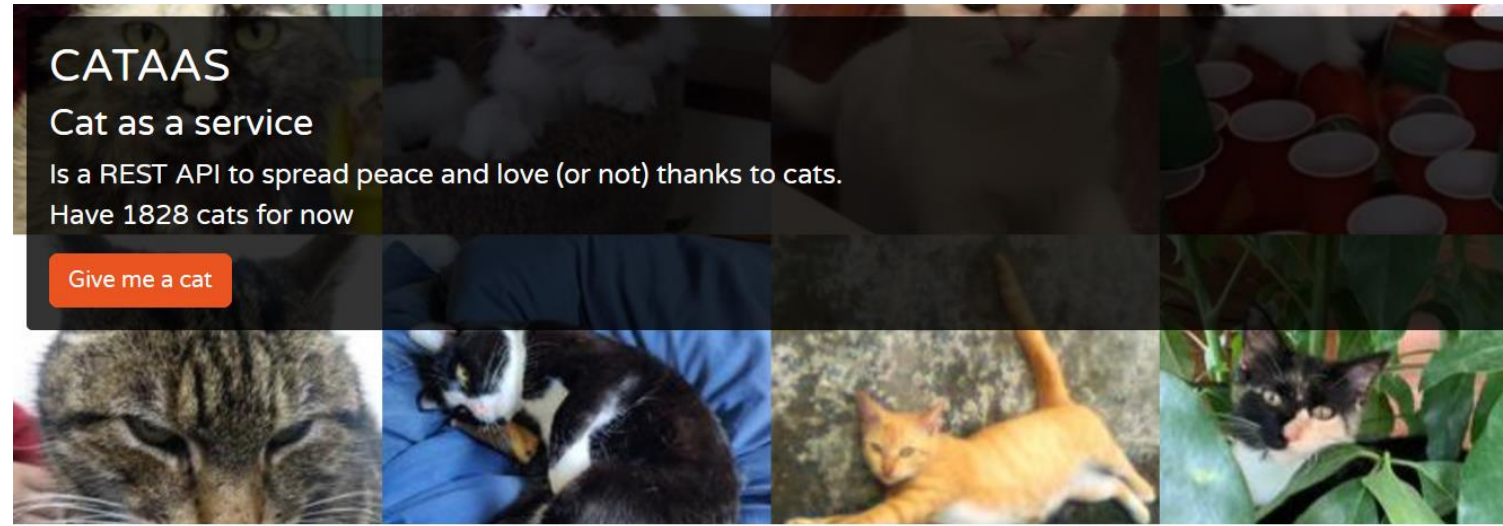
- Now JSON will returns if request contain `application/json` header, API doc updated [here](#)
- Fix tags search and fix tags combined with `" , "` separator (see documentation)

Basic

Url	Description	Example
<code>/cat</code>	Will return a random cat	Random cat
<code>/cat/:tag</code>	Will return a random cat with a <code>:tag</code> , You can combine multiple tags with <code>:tag</code> separator	Random orange cute cat
<code>/cat/gif</code>	Will return a random gif cat \o/	Random gif cat
<code>/cat/says/:text</code>	Will return a random cat saying <code>:text</code>	Random cat saying hello
<code>/cat/:tag/says/:text</code>	Will return a random cat with a <code>:tag</code> and saying <code>:text</code>	Random cute cat saying hello
<code>/cat/says/:text?fontSize=:size&fontColor=:color</code>	Will return a random cat saying <code>:text</code> with text's <code>:fontSize</code> and text's <code>:fontColor</code>	Custom random cat saying hello

<https://cataas.com/>

Do you remember CATAAS?



Hey ! Do you like Cataas? Do you want to support the project ?

Buy me a beer 🍺

Hey cat lovers, new major version of CATAAS

- Now JSON will returns if request contain `application/json` header, API doc updated [here](#)
- Fix tags search and fix tags combined with `" , "` separator (see documentation)

Basic

Url	Description	Example
<code>/cat</code>	Will return a random cat	Random cat
<code>/cat/:tag</code>	Will return a random cat with a <code>:tag</code> , You can combine multiple tags with <code>:tag</code> separator	Random orange cute cat
<code>/cat/gif</code>	Will return a random gif cat \o/	Random gif cat
<code>/cat/says/:text</code>	Will return a random cat saying <code>:text</code>	Random cat saying hello
<code>/cat/:tag/says/:text</code>	Will return a random cat with a <code>:tag</code> and saying <code>:text</code>	Random cute cat saying hello
<code>/cat/says/:text?fontSize=:size&fontColor=:color</code>	Will return a random cat saying <code>:text</code> with text's <code>:fontSize</code> and text's <code>:fontColor</code>	Custom random cat saying hello

<https://cataas.com/>

CATAAS REST API

Cat as a service (CATAAS) 1.0.0 OAS3

[/doc.json](#)

Cat as a service (CATAAS) is a REST API to spread peace and love (or not) thanks to cats.

Servers

<https://cataas.com> ▾

Authorize



Cats Cataas API



GET /cat ▾

GET /cat/{id} ▾

GET /cat/{tag} ▾

GET /cat/says/{text} ▾

GET /cat/{id}/says/{text} ▾

GET /cat/{tag}/says/{text} ▾

API Public API



GET /api/cats ▾

CATAAS REST API

Cat as a service (CATAAS) 1.0.0 OAS3

[/doc.json](#)

Cat as a service (CATAAS) is a REST API to spread peace and love (or not) thanks to cats.

Servers

<https://cataas.com> ▾

Authorize



Cats Cataas API



GET /cat



GET /cat/{id}



GET /cat/{tag}



GET /cat/says/{text}



GET /cat/{id}/says/{text}



GET /cat/{tag}/says/{text}



API Public API



GET /api/cats



CATAAS REST API

```
openapi: "3.0.3"
info:
  version: "1.0.0"
  title: "Cat as a service (CATAAS)"
  description: "Cat as a service (CATAAS) is a REST API to spread peace and love (or not) thanks to cats."
servers:
  0:
    url: "https://cataas.com"
tags:
  0:
    name: "Cats"
    description: "Cataas API"
  1:
    name: "API"
    description: "Public API"
  2:
    name: "Security"
    description: "Security"
  3:
    name: "Admin"
    description: "Admin API"
components:
  securitySchemes:
    jwt:
      type: "http"
      scheme: "bearer"
      bearerFormat: "JWT"
  schemas:
    AdminCat:
      required:
        0: "file"
        1: "tags"
        2: "validated"
      type: "object"
      properties:
        _id:
          type: "string"
      validated:
```

OpenAPI - Specification

Mandatory parts:

- Version and info
- At least one of:
 - Paths -> endpoints
 - Webhooks -> callbacks
 - Components -> reusable parts of the openapi file

Optional parts:

- Servers
- Security
- Tags
- External Docs



<https://swagger.io/specification/>

OpenAPI – Version and info

```
openapi: 3.0.0 # The OpenAPI version you're using
info:
  title: Example API # The name of your API
  description: A simple API for example. # A brief description
  version: 1.0.0 # Version of the API
```

OpenAPI – paths

/add?a=1&b=1

```
paths:
  /add:
    get:
      summary: Perform addition
      parameters:
        - name: a
          in: query
          required: true
          description: First operand (a).
          schema:
            type: number
        - name: b
          in: query
          required: true
          description: Second operand (b).
          schema:
            type: number
      responses:
        '200':
          description: Addition result
          content:
            application/json:
              schema:
                type: object
                properties:
                  s:
                    type: number
        '400':
          description: Invalid input
        '404':
          description: Service down or invalid operation
```

OpenAPI – Swagger Editor

<https://editor.swagger.io/>

Example API 1.0.0 OAS 3.0

A simple API for example.

default

GET /add Perform addition

Try it out

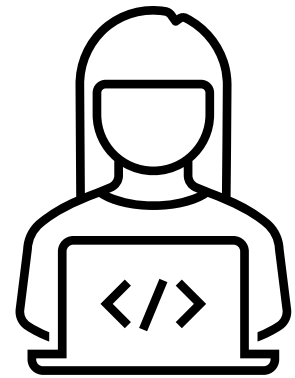
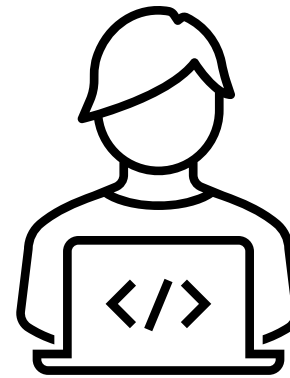
Name	Description
a <small>★ required</small> number <small>(query)</small>	First operand (a). <input type="text" value="a"/>
b <small>★ required</small> number <small>(query)</small>	Second operand (b). <input type="text" value="b"/>

Responses

Code	Description	Links
200	<div>Addition result</div> <div>Media type <div>application/json</div><div>Controls Accept header.</div><div>Example Value Schema</div><div><pre>{ "s": 0 }</pre></div></div> <td>No links</td>	No links
400	Invalid input	No links
404	Service down or invalid operation	No links

OpenAPI - Writing

Do I have to do it by hand?



OpenAPI - Writing

Not necessarily!

There are tools that from your code generate (part of) the OpenAPI spec and vice versa.

For example:

- [OAS tools](#) (npm based)
- [Flask-RESTX](#) (annotations of Flask code)

Now it's your turn

1. Use the service you coded in Lab 1 (or download it from the Moodle).
2. Write the OpenAPI specification for the REST service.
3. Use Swagger Editor to see its web representation.



Service paths

- /add
- /sub
- /mul
- /div
- /mod
- /random
- /concat
- /upper
- /lower
- /reduce
- /crash
- /last

Lab take away

- ☐ Learn about OpenAPI
- ☐ Write the OpenAPI specification of a REST API



Project take away

- ❑ The documentation of the REST API of the project must be expressed in OpenAPI format.

