

Building Elegant API Contracts

From Zero to Hero



Neeraj Pandey

Generative Artist
Senior at Ashoka University

Computational Arts, Quantitative Finance,
Full Stack Web and Data Science

//02

ASHOKA UNIVERISTY

PyCascades
2022

Elements and Principles	05
Stateless vs Stateful architecture	06
Contract First vs Code First	07
Single Source of Truth	08
API Contracts	09
Hands On with Swagger, Postman, Fast API and Flask	20

Points for discussion

//03

Application Programming Interface (API)

ELEMENTS AND PRINCIPLES

PURPOSE

Identify the demands of the end developer, build your service with enough methods to handle the majority of needed requests, and make all parameters discoverable and well-defined in API documentation.

USABILITY

The API representation should be intuitive, the interactions should be as simple as feasible, and the operations should execute smoothly, allowing for faster and easier attainment of the desired objectives. As a result, the system as a whole must be consistent, adaptive, and discoverable.

CONSTRAINTS

Impose limits for successful API design if they assist you to develop your product. It's worth examining if one can make appropriate design concessions to optimize product delivery from your end, such as restricting API calls to better scalability or enabling asynchronous returns to improve performance.

Stateless vs Stateful

STATELESS API DESIGN

Stateless API integration in restful apis is a standard that enforces scalability through interoperability with contemporary architectures and promotes an agile approach. Furthermore, stateless allows for quicker upgrades and version deployments, as well as seamless integration with HTTP protocols.

Code First VS API First

//07

Code First

DELIVER SIMPLE FAST API'S

A code-first method entails creating an API from business needs and then converting that code into a machine-readable API description. It might also imply defining the API in code and then constructing an API description using comments or annotations, or even manually writing one from start.

API First

MOCK SERVERS, POSITIVE DEV EXP.

The API is based around a contract stated in an API description language under the API First methodology. It assures the API's consistency, reusability, and broad compatibility. It gives API Design and Development teams a solid source of truth.

Single Source of Truth



A single source of truth (SSOT) is the practice of structuring information models and associated data schema such that every data element is mastered (or edited) in only one place

~ wikipedia

API CONTRACTS

PyCascades
2022

OPEN API, RAML, API BLUEPRINT - APIARY

API BLUEPRINT

API Blueprint has a Markdown-like syntax, MSON compatibility, and complete support for all Apiary and open-source tooling, making it ideal for an API-first strategy.

RAML

RAML stands for RESTful API Modeling Language, a YAML-based API modeling language that aids in the creation of API contracts and provides a well-structured and easy-to-understand framework for defining APIs.

OPEN API (SWAGGER)

The OpenAPI Specification is a framework for defining REST API structure and terminology. OpenAPI documentation have a JSON and YAML format and are machine and human understandable, allowing anybody to quickly figure out how each API works.

API-Design First

Open API is an API-first strategy that enables enterprises to create APIs that service all apps, allowing for easy development and maintenance across all devices, platforms, and operating systems. It also allows developers to work in parallel, lowering costs and speeding up the development process.

Tooling

To speed up the development process, it includes documentation and the ability to prototype and work with mock servers, which may anticipate a variety of issues that may arise while using APIs and integrating systems.

Server Stubs

OpenAPI is the REST standard that has the most code generators that support it. It builds server stubs in our preferred language, which we must then connect to our backend services and databases.

Open API Specifications

Metadata

Incorporate basic information about your API within the specification, such as the version number, licensing comments, contact information, documentation links, and so on.

```
openapi: 3.0.0
servers:
  - description: Mock API for PyCascades 2022
    url: https://localhost:4000
info:
  description: This is a mock API
  termsOfService: http://example.com/terms
  version: "1.0.0"
  title: PyCascades Mock API
  contact:
    name: PyCascades Speaker
    email: you@your-company.com
    url: https://example.com/speakeremail
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
```

METADATA

Metadata Output

PyCascades Mock API

1.0.0OAS3

This is a mock API

[Terms of service](#)

[PyCascades Speaker - Website](#)

[Send email to PyCascades Speaker](#)

[Apache 2.0](#)

Servers

https://localhost:4000 - Mock API for PyCascades 2022 ▾

Filter by tag

No operations defined in spec!

METADATA OUTPUT

Path Objects

Each endpoint's relative paths and actions are stored in the Path Object.



```
paths:
  /:
    get:
      operationId: get-talks
      summary: Returns a list of talks.
      description: Returns a list of talks.
      tags:
        - Talks
```



Parameters

To describe a parameter, you specify its name, location (in), data type (defined by either schema or content) and other attributes, such as description or required.



```
parameters:  
  - name: listSize  
    in: query  
    description: Desired length  
    schema:  
    type: integer  
    format: int32  
    default: 1  
    minimum: 1  
    maximum: 100
```



\$ref object

Allows for a referenced definition of this path item. The referenced structure MUST be in the form of a Path Item Object.

```
# Parameters
parameters:
  - $ref: '#/components/parameters/listSize'

components:
  parameters:
    listSize:
      in: query
      description: Desired length
      name: listSize
      schema:
        type: integer
        format: int32
        default: 1
        minimum: 1
        maximum: 100
```

\$REF OBJECT BLOCK

responses object

Each operation must have at least one response defined, usually a successful response. A response is defined by its HTTP status code and the data returned in the response body and/or headers



```
responses:
  '200':
    description: Success. It works!
    content:
      application/json:
        schema:
          type: object
          properties:
            output:
              type: string
              example: This is a returned string!
```



path all-together

```
paths:
  /:
    get:
      operationId: get-talks
      summary: Returns a list of talks.
      description: Returns a list of all the talks.
      tags:
        - Talks
      parameters:
        - $ref: '#/components/parameters/listSize'
      responses:
        '200':
          description: Success. It works!
          content:
            application/json:
              schema:
                type: object
                properties:
                  output:
                    type: string
                    example: This is a returned string!

components:
  parameters:
    listSize:
      in: query
      description: Desired length
      name: page
      schema:
        type: integer
        format: int32
        default: 1
        minimum: 1
        maximum: 100
```

Path Block Output

GET

/ Returns a list of talks.

^

Returns a list of all the talks.

Parameters

Try it out

Name	Description
page	Desired length
<div><div>integer(\$int32)</div><div>(query)</div><div>minimum: 1</div><div>maximum: 100</div></div>	Default value : 1

1

Responses

Code	Description	Links
200	Success. It works!	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{  "output": "This is a returned string!"}
```

//19

Conference Voting

Mock API

/

/talk/{id}

/vote/{id}

Mock Voting App

Metadata

PyCascades
2022

```
openapi: 3.0.0
info:
  title: PyCascades Voting
  version: ""
  description: Vote for the talks you liked at PyCascades 2022
  contact:
    name: Neeraj Pandey
    email: neerajemail@email.com
    url: https://twitter.com/neerajp99
  license:
    name: XYZ License
    url: https://licence.xyz.com
servers:
  - url: http://localhost:8000
    description: Local dev server
tags:
  - name: details
    description: All details of the talks
  - name: detail_single
    description: Detail of a single talk
  - name: upvote
    description: Upvote a talk
```

Mock Voting App

Path

/

PyCascades
2022

```
    '/':
      get:
        tags:
          - details
        summary: fetches talks list
        description: Get all the talks
        parameters:
          - name: talkLimit
            in: query
            description: Number of talks fetched in a single request
            schema:
              type: integer
              minimum: 1
              maximum: 50
              example: 4
        responses:
          '200':
            description: It works!
            content:
              application/json:
                schema:
                  type: array
                  items:
                    properties:
                      talk_id:
                        type: string
                        example: pycascades2022-talk01
                      talk_name:
                        type: string
                        example: Talk Title
                      talk_speaker:
                        type: string
                        example: Neeraj Pandey
                      talk_pitch:
                        type: string
                        example: Talk Description goes here
          '404':
            description: No talk found on this url!
```

//21

Mock Voting App

Path

`/talk/{id}`

PyCascades 2022

//22

Mock Voting App

Path

/vote/{id}

PyCascades
2022

```
    '/vote/{id}':
      post:
        tags:
          - upvote
        summary: upvotes for a talk
        description: Cast a vote for the given talk
        parameters:
          - name: id
            required: true
            in: path
            description: ID of the talk
            schema:
              type: string
        requestBody:
          required: true
          content:
            application/json:
              schema:
                type: object
                properties:
                  talk_id:
                    type: string
                    example: pycascades2022-talk01
        responses:
          '200':
            description: Your vote has been successfully cast.
          '400':
            description: Invalid input
          '422':
            description: You have already voted for this talk!
          '404':
            description: Talk not found!
```

//23

Mock Voting App

Swagger UI

PyCascades Voting

Vote for the talks you liked at PyCascades 2022

[Neeraj Pandey - Website](#)
[Send email to Neeraj Pandey](#)
[XYZ License](#)

Servers

[http://localhost:5000 - Local dev server](#) ▾

Filter by tag

details All details of the talks



GET / fetches talks list



detail_single Detail of a single talk



GET /talk/{id} fetch the current talk using talk id



upvote Upvote a talk



POST /vote/{id} upvotes for a talk



PyCascades
2022

//24

Mock Voting App

Executing

GET /

PyCascades
2022

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/?talkLimit=4' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/?talkLimit=4
```

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "talk_id": "pycascades2022-talk01", "talk_name": "Building Elegant API Contracts: From Zero to Hero", "talk_speaker": "Neeraj Pandey", "talk_pitch": "Learn how one can write efficient APIs with high-quality API specifications using Open API and RAML specs to create API contracts and achieve a better experience using the API with more reliable unit tests and increased response consistency." }, { "talk_id": "pycascades2022-talk02", "talk_name": "Computational Creativity: Could AI be the next DAD A movement?", "talk_speaker": "Neeraj Pandey", "talk_pitch": "Learn how the computational, psychological, and economic aspects of AI-generated art, music, and generative design along with automated computational tasks using Machine Learning and Artificial Intelligence." }, { "talk_id": "pycascades2022-talk03", "talk_name": "Genetic Algorithms: Better optimisation with Evolutionary Principles", }]</pre> <p>Response headers</p> <pre>content-length: 1907 content-type: application/json</pre>

//25

OpenAPI 3.0

SaveCancelYAML

General

openapi

info

tags

Servers

http://localhost:8000

Paths

/

/talk/{id}

/vote/{id}

Components

No components defined.

Security

No security schemes defined.

```
1 openapi: 3.0.0
2 info:
3   title: PyCascades Voting
4   version: ""
5   description: Vote for the talks you liked at PyCascades 2022
6   contact:
7     name: Neeraj Pandey
8     email: neerajemail@email.com
9     url: https://twitter.com/neerajp99
10  license:
11    name: XYZ License
12    url: https://licence.xyz.com
13 servers:
14   - url: http://localhost:8000
15     description: Local dev server
16 tags:
17   - name: details
18     description: All details of the talks
19   - name: detail_single
20     description: Detail of a single talk
21   - name: upvote
22     description: Upvote a talk
23 paths:
24   '/':
25     get:
26       tags:
27         - details
28       summary: fetches talks list
29       description: Get all the talks
30       parameters:
31         - name: talkLimit
32           in: query
33           description: Number of talks fetched in a single request
34           schema:
35             type: integer
36             minimum: 1
37             maximum: 50
38             example: 4
39       responses:
40         '200':
41           description: It works!
42           content:
43             application/json:
44               schema:
45                 type: array
46                 items:
47                   properties:
48                     talk_id:
49                       type: string
50                       example: pycascades2022-talk01
51                     talk_name:
52                       type: string
53                       example: 'Building Elegant API Contracts: From Zero To Hero'
54                     talk_speaker:
55                       type: string
56                       example: Neeraj Pandey
57                     talk_pitch:
58                       type: string
59                       example: Learn how one can write efficient APIs with high quality API specifications
```

Issues found in schema:

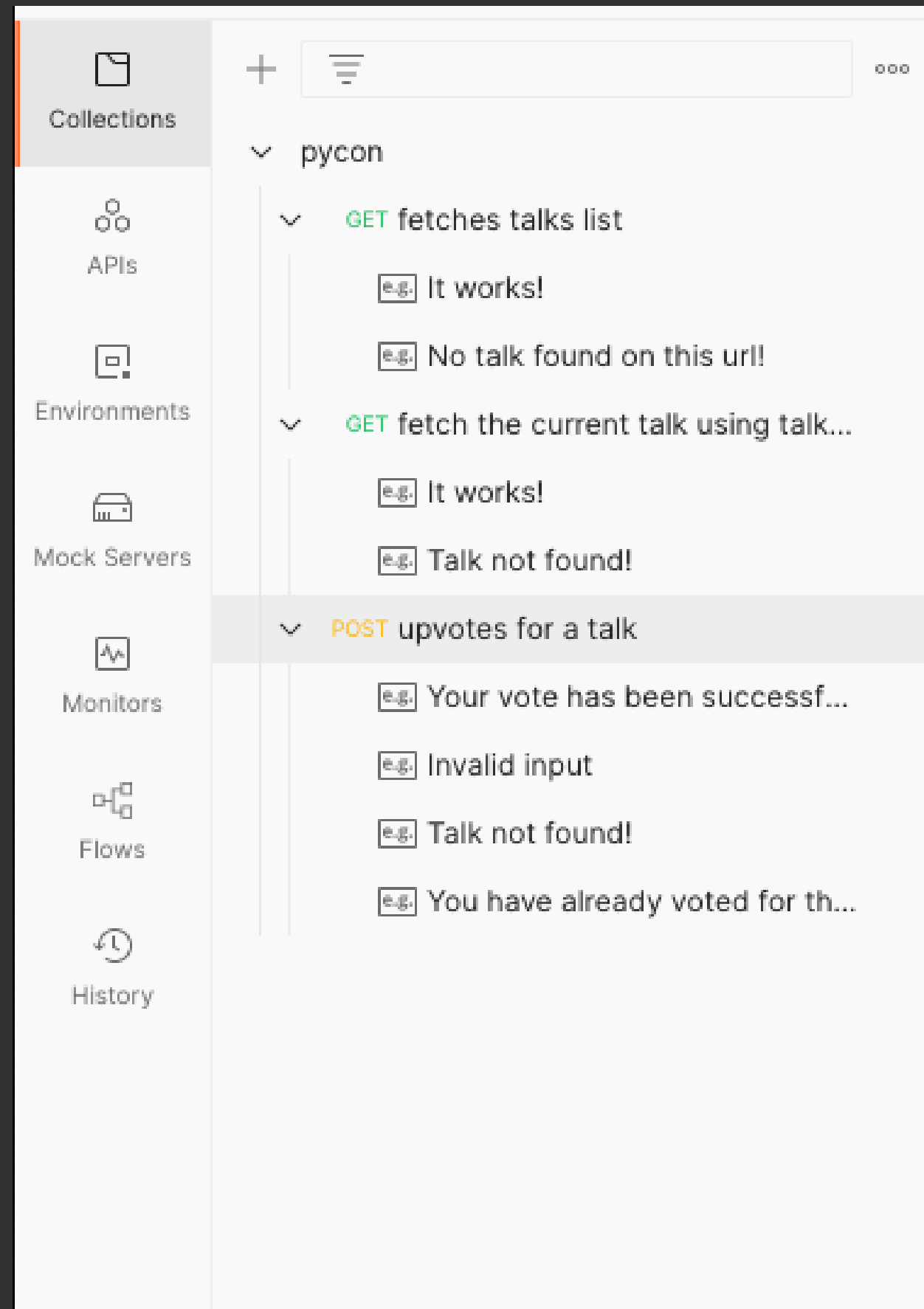
Errors (0)

Warnings (4)

BETA

Mock Voting App

Mock Sever
Postman



PyCascades
2022

Fast API

PyCascades
2022

```
db = json.load(open('db.json'))

@app.get("/", status_code=200)
def show_talks():
    return db

# Create a method that takes in talk id and returns the talk object from the db.json
@app.get("/talk/{id}")
def get_talk(id: str):
    try:
        for talk in db:
            if talk['talk_id'] == id:
                return JSONResponse (
                    status_code = 200,
                    content = {"talk": talk}
                )
        return JSONResponse(status_code=404, content={"Talk not found"})
    except:
        raise HTTPException(status_code=404, detail="Talk not found")
        return {"error": "Talk not found"}
```

Fast API comes with pre built Swagger UI based on the open standards for APIs: OpenAPI and JSON Schema.

ROUTE TO GET A TALK DETAILS BY TALK ID

//28

{baseUrl}/docs

localhost

FastAPI - Swagger UI

FastAPI

0.1.0

OAS3

/openapi.json

default

^

GET / Show Talks

^

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

"string"

GET /talk/{id} Get Talk

^

Parameters

Try it out

Name	Description
id required	
string (path)	id

PyCascades
2022

//29

FLASK APP

PyCascades
2022

```
from flask_cors import CORS
from typing import Optional
from apispec import APISpec
from apispec.ext.marshmallow import MarshmallowPlugin
from apispec_webframeworks.flask import FlaskPlugin
from flask import Flask, jsonify, render_template, send_from_directory, Response
from marshmallow import Schema, fields
from werkzeug.exceptions import HTTPException
import json

app = Flask(__name__)

# Import the json object from the db.json
import json
db = json.load(open('db.json'))
print(db[0])

spec = APISpec(
    title="Conference Voting",
    version="1.0.0",
    openapi_version="3.0.0",
    plugins=[FlaskPlugin(), MarshmallowPlugin()]
)
```

With Flask, we can use multiple external libraries to add the Swagger UI.

ROUTE TO GET A TALK DETAILS BY TALK ID

//30

```
@app.route("/talk/<id>", methods=['GET'])
def get_talk(id: str):
    """Get talk by talk id
    ---
    get:
      description: Get talk by talk id
      responses:
        200:
          description: Talk found
          content:
            application/json:
              schema:
                type: object
                properties:
                  talk_id:
                    type: string
                    example: pycascades2022-talk01
                  talk_name:
                    type: string
                    example: Talk Title
                  talk_speaker:
                    type: string
                    example: Neeraj Pandey
                  talk_pitch:
                    type: string
                    example: Talk Description
        '4XX':
          description: Talk not found!
          content:
            application/http-problems: {}

    """
    try:
        for talk in db:
            print('talk: ', talk)
            if talk['talk_id'] == id:
                return Response(
                    response=json.dumps(talk),
                    status=201,
                    mimetype='application/json'
                )
        return Response("Talk not found!", status=404, mimetype='application/json')
    except:
        return Response("Talk not found!", status=404, mimetype='application/json')
```

Route for the Swagger UI

```

@app.route('/api/openapi.json')
def create_openapi_json():
    return jsonify(spec.to_dict())

with app.test_request_context():
    spec.path(view=show_talks)
    spec.path(view=get_talk)
    spec.path(view=create_openapi_json)

if __name__ == "__main__":
    CORS(app)
    app.run(debug=True)

```


{baseUrl}/api/openapi.json

```
localhost
localhost:5000/api/openapi.json

{
  "info": {
    "title": "Conference Voting",
    "version": "1.0.0"
  },
  "openapi": "3.0.0",
  "paths": {
    "/": {
      "get": {
        "description": "Get details of all the beautiful talks submitted by the speakers.",
        "parameters": [
          {
            "description": "Number of talks fetched in a single request",
            "in": "query",
            "name": "talkLimit",
            "schema": {
              "example": 35,
              "maximum": 50,
              "minimum": 20,
              "type": "integer"
            }
          }
        ],
        "responses": {
          "200": {
            "content": {
              "application/json": {
                "schema": {
                  "items": {
                    "properties": {
                      "talk_id": {
                        "example": "pycascades2022-talk01",
                        "type": "string"
                      },
                      "talk_name": {
                        "example": "Building Elegant API Contracts: From Zero to Hero",
                        "type": "string"
                      },
                      "talk_pitch": {
                        "example": "Learn how one can write efficient APIs with high-quality API specifications using Open API and RAML s
more reliable unit tests and increased response consistency.",
                        "type": "string"
                      },
                      "talk_speaker": {
                        "example": "Neeraj Pandey",
                        "type": "string"
                      }
                    }
                  },
                  "type": "array"
                }
              }
            }
          },
          "404": {
            "description": "No talk found on this url!"
          }
        },
        "description": "It works!"
      }
    }
  },
  "summary": "fetches talks list",
}
```

//34

Thank You

TWITTER

[neerajp99](#)

SLIDES

<https://bit.ly/3rtM5Gt>