

# **CONTRACT FIRST WITH OPENAPI**

March 19, 2021

# ABOUT ME



Lead Software Engineer in Exadel



10+ years in Java-development

**ARTEM VEGERA**

# AGENDA

- ➊ Difference between Contract First and Code First
- ➋ How to do Contract First with OpenAPI Specification
- ➌ Pitfalls of Contract First approach
- ➍ OpenAPI tooling landscape

# THE INITIAL PROBLEM

## TYPICAL API DEVELOPMENT

- ➊ Feature request
- ➋ Ping-pong development between BE and UI devs
- ➌ QA can write API-tests after BE-implementation
- ➍ Feature in production

# THE INITIAL PROBLEM

TYPICAL API  
DEVELOPMENT

WHAT WE  
WANT?

- ➊ Be able to specify our exposed API
- ➋ Parallelize API-implementation and API-consumers work like BE, UI, Mobile, QA Automation, Integration Apps, etc.
- ➌ (optional) Have separate API design phase

# WITHOUT DOCUMENTATION

-  Just write what's on your mind
-  Easy to make changes and break things\*
-  Boost your personal productivity\*

## PROS

\* before sharing your API with other people

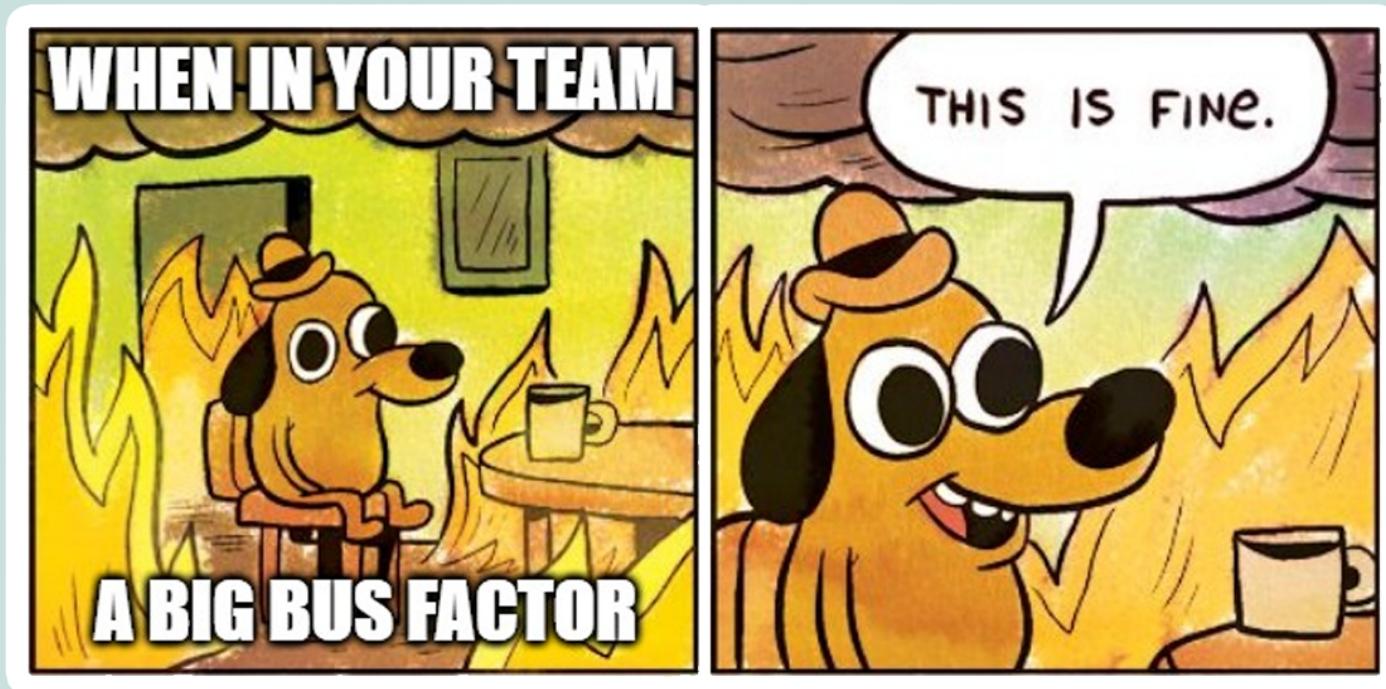
# WITHOUT DOCUMENTATION

PROS

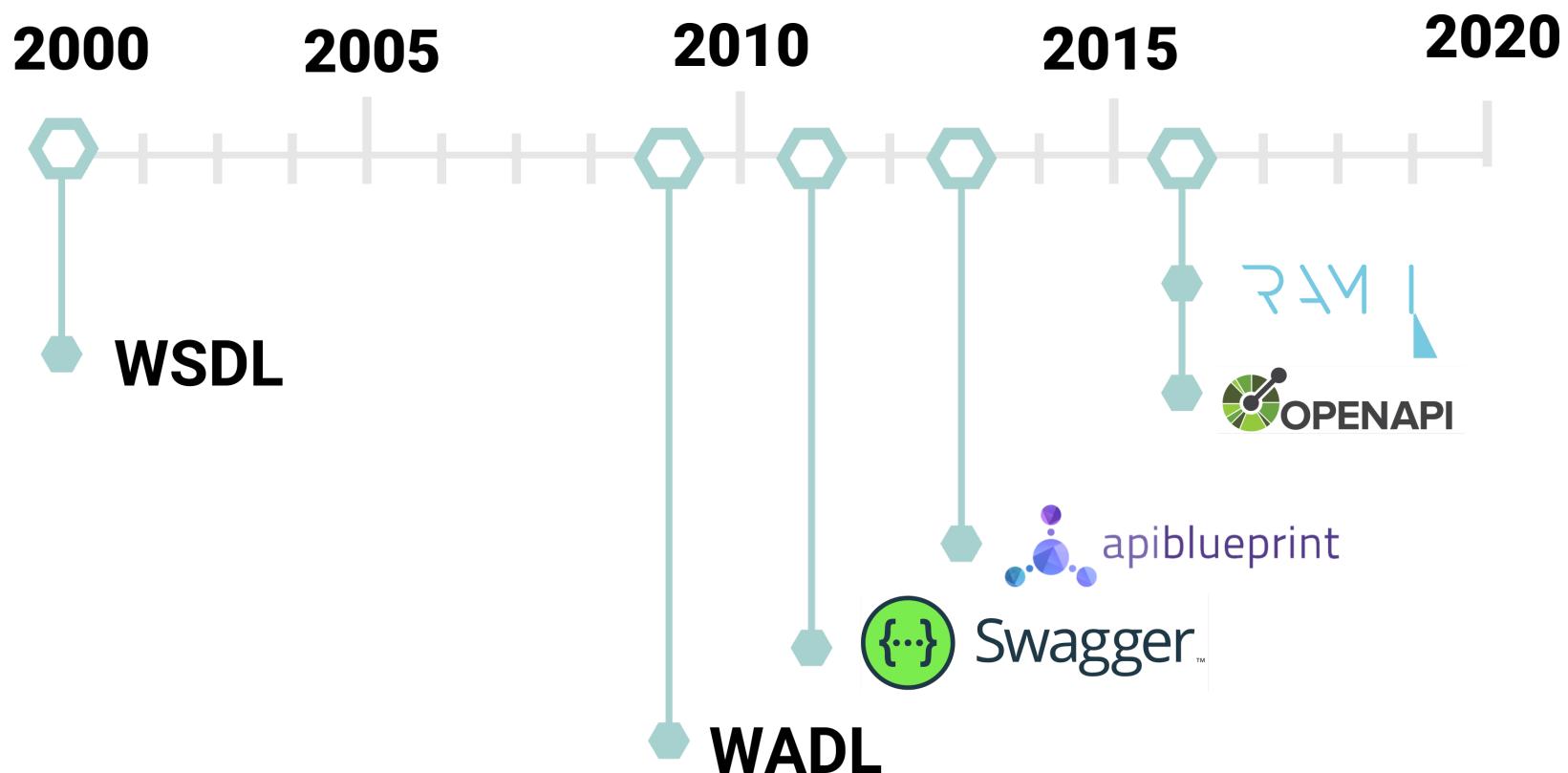
**CONS**

- Difficult collaboration
- Risk of bus factor
- No mechanism of notification about API changes

# WE SHOULD DO SOMETHING



# API DESCRIPTION LANGUAGES



# WHICH ONE IS BETTER?



# OPENAPI PITCH



# TELL ME MORE ABOUT OPENAPI!



# OPENAPI SPECIFICATION

## OVERVIEW



OPENAPI



# OPENAPI SPECIFICATION

## OVERVIEW

## HISTORY

	VERSION	RELEASE DATE
SWAGGER	1.0	Aug 10, 2011
	1.1	Aug 22, 2012
	2.0	Sep 08, 2014
OPENAPI	3.0.0	Jul 26, 2017
	3.0.1	Dec 06, 2017
	3.0.2	Oct 08, 2018
	3.0.3	Feb 02, 2020
	3.1	Feb 20, 2021

# OPENAPI SPECIFICATION

OVERVIEW

HISTORY

**STRUCTURE**

**INFO**

**SERVERS**

**SECURITY**

**PATHS**

**TAGS**

**EXTERNAL DOCS**

**COMPONENTS**

# OPENAPI SPECIFICATION

OVERVIEW

HISTORY

STRUCTURE

```
1  openapi: 3.0.2
2  info:
3    title: Swagger Petstore - OpenAPI 3.0
4    description: This is a sample of OAS3.
5    termsOfService: http://swagger.io/terms/
6    license:
7      name: Apache 2.0
8      url: http://www.apache.org/licenses/LICENSE-2.0.html
9    version: 1.0.5
10   externalDocs: <2 keys>
11   servers: <1 item>
12   tags:
13     - name: pet
14       description: Everything about your Pets
15     - name: store
16       description: Everything about your Store
17   paths:
18     /pet:
19       put: <7 keys>
20       post: <7 keys>
21     /pet/findByStatus:
22       get: <7 keys>
23     /pet/findByTags:
24       get: <7 keys>
25     /pet/{petId}:
26       get: <7 keys>
27       post: <7 keys>
28       delete: <7 keys>
```

# OPENAPI SPECIFICATION

OVERVIEW

HISTORY

STRUCTURE

```
1   paths:
2     /pet/{petId}:
3       get:
4         tags:
5           - pet
6           summary: Find pet by Id
7           description: Returns a single pet
8           operationId: getPetById
9           parameters:
10          - name: petId
11            in: path
12            description: Id of pet to return
13            required: true
14            schema:
15              type: integer
16              format: int64
17           responses:
18             "200":
19               description: successful operation
20               content:
21                 application/json:
22                   schema:
23                     $ref: '#/components/schemas/Pet'
```

# OPENAPI SPECIFICATION

OVERVIEW

HISTORY

**STRUCTURE**

```
1  schemas:
2    Address:
3      type: object
4      properties:
5        street:
6          type: string
7          example: 437 Lytton
8        city:
9          type: string
10         example: Palo Alto
11        state:
12          type: string
13          example: CA
14        zip:
15          type: string
16          example: "94301"
17
18
19
20
21
22
23
```

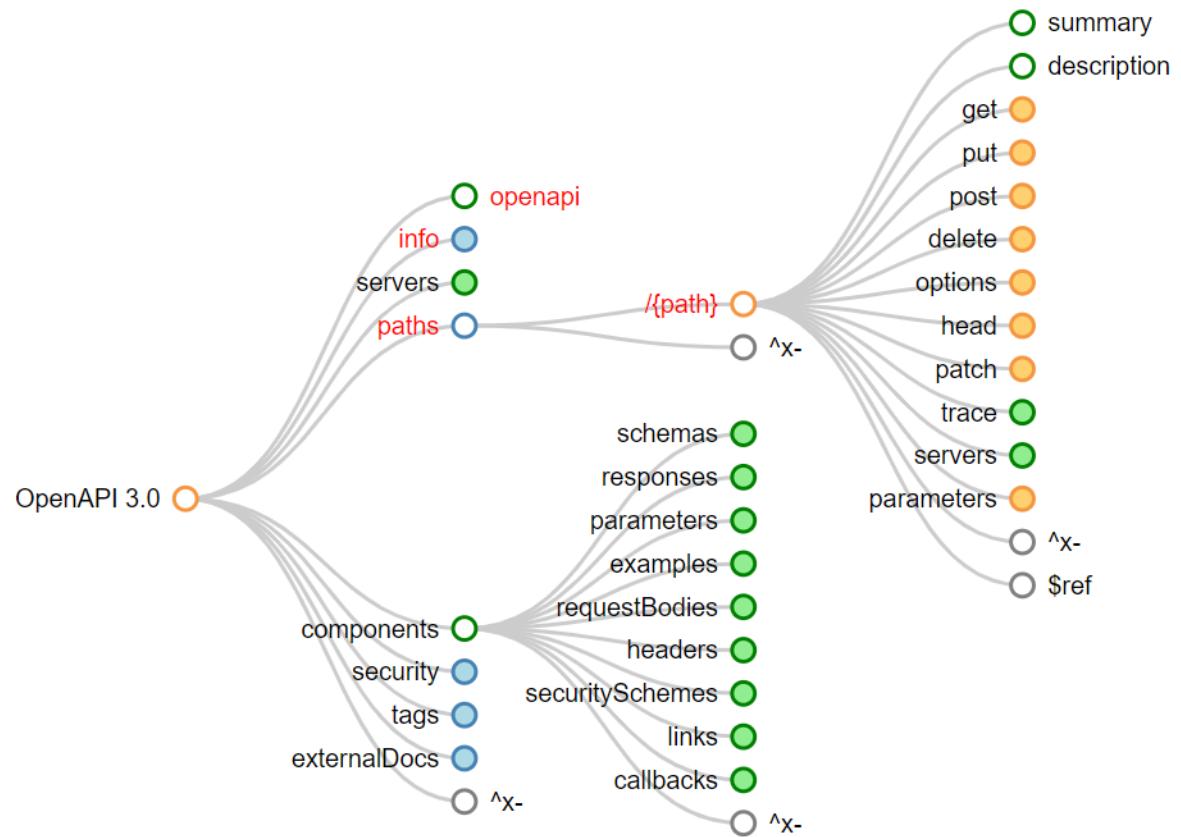
# OPENAPI SPECIFICATION

OVERVIEW

HISTORY

STRUCTURE

MIND MAP



[openapi-map.apihandyman.io](https://openapi-map.apihandyman.io)

# OPENAPI SPECIFICATION

OVERVIEW

HISTORY

STRUCTURE

MIND MAP

LINKS

## LAST VERSION

-  [spec.openapis.org/oas/v3.1.0](https://spec.openapis.org/oas/v3.1.0)

## EXAMPLES

-  [petstore3.swagger.io](https://petstore3.swagger.io)
-  [github.com/github/rest-api-description](https://github.com/github/rest-api-description)
-  [github.com/azure/azure-rest-api-specs](https://github.com/azure/azure-rest-api-specs)

# OPENAPI INITIATIVE

## OVERVIEW



# OPENAPI INITIATIVE

## OVERVIEW MEMBERS



# OPENAPI INITIATIVE

OVERVIEW

MEMBERS

**OAS VS  
SWAGGER**



**STANDARD**



**TOOLING**

FROM SMARTBEAR COMPANY

# OPENAPI INITIATIVE

OVERVIEW

MEMBERS

OAS VS  
SWAGGER

**LINKS**

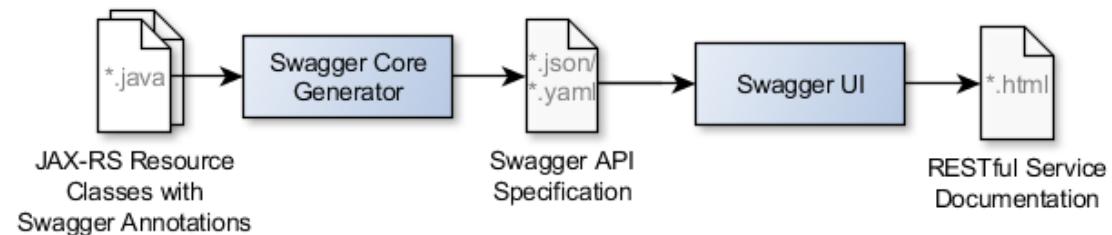
- openapis.org
- github.com/oai
- twitter.com/openapispec
- youtube.com/channel/UCaw7jsGhrsaoSIYPPUF5q2w

AAAAAAAAAAAAAAA



# OAS GENERATION FROM SOURCE CODE

## APPROACH



# OAS GENERATION FROM SOURCE CODE

APPROACH

EXAMPLE

The screenshot shows the Swagger Petstore API documentation. At the top, there's a header with the Swagger logo, the URL <https://petstore.swagger.io/v2/swagger.json>, and a green "Explore" button. Below the header, the title "Swagger Petstore 1.0.5" is displayed, along with the base URL <https://petstore.swagger.io/v2/>. A note says, "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." There's a dropdown menu for "Schemes" set to "HTTPS" and a "Authorize" button with a lock icon.

The main content area is titled "pet" and describes "Everything about your Pets". It lists various API endpoints:

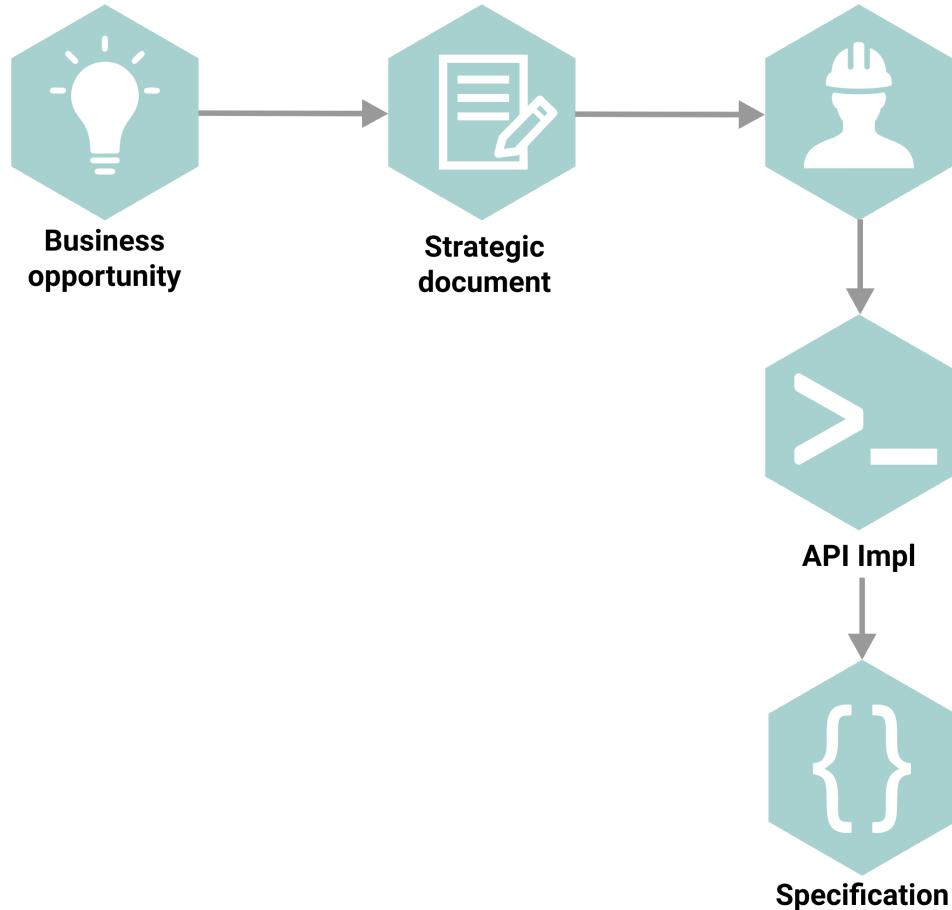
- POST /pet/{petId}/uploadImage** uploads an image (green background)
- POST /pet** Add a new pet to the store (green background)
- PUT /pet** Update an existing pet (orange background)
- GET /pet/findByStatus** Finds Pets by status (blue background)
- GET /pet/findByTags** Finds Pets by tags (light gray background)
- GET /pet/{petId}** Find pet by ID (blue background)
- POST /pet/{petId}** Updates a pet in the store with form data (green background)
- DELETE /pet/{petId}** Deletes a pet (red background)

Each endpoint row has a lock icon on the right side.

# GENERATE DOCS USING ONE BUTTON?



# CODE FIRST

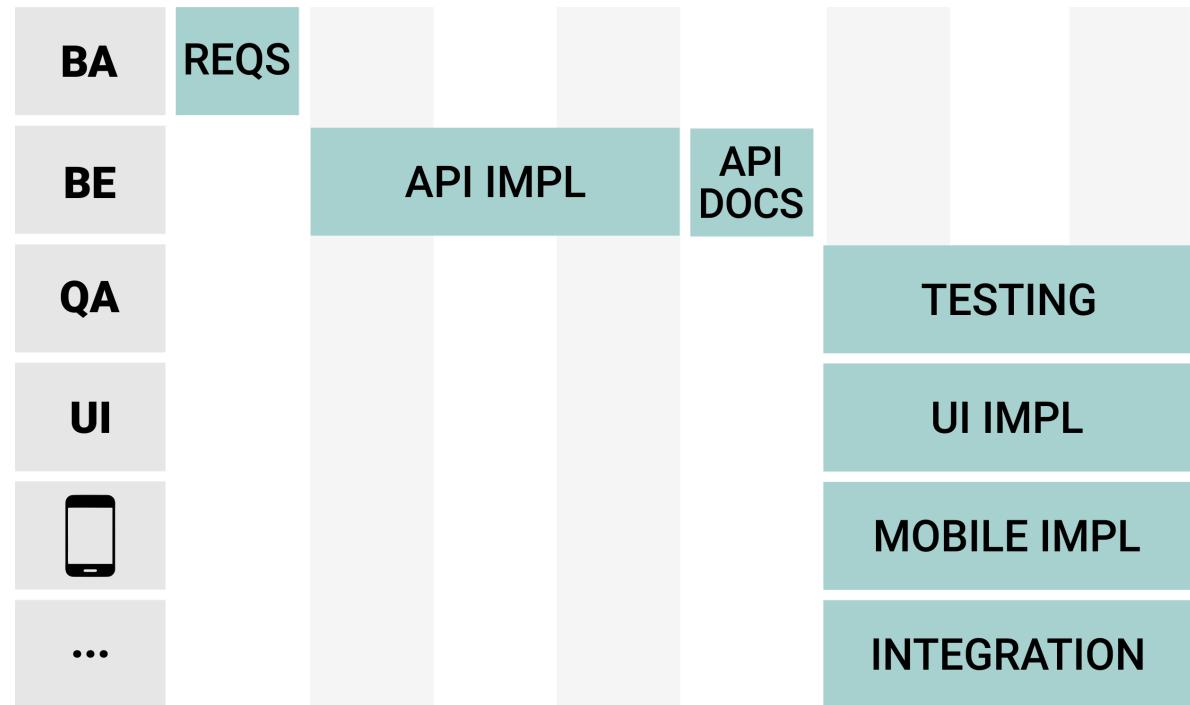


## OVERVIEW

# CODE FIRST

OVERVIEW

WORKFLOW



# CODE FIRST

OVERVIEW

WORKFLOW

**PROS**

-  Easy to implement and use
-  Documentation is a part of process out of the box
-  Documentation is up to date always
-  Code is single source of truth

# CODE FIRST IN REAL LIFE



# CODE FIRST

OVERVIEW

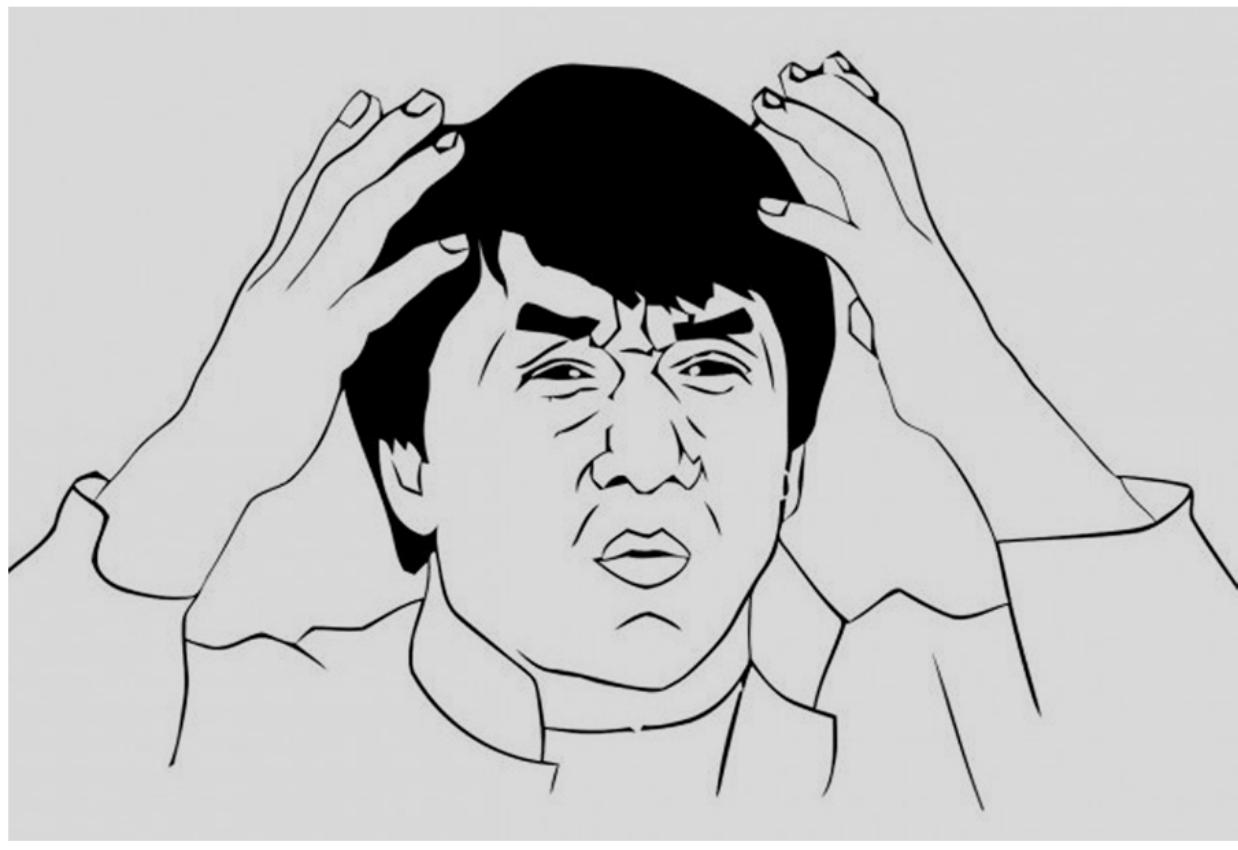
WORKFLOW

PROS

**CONS**

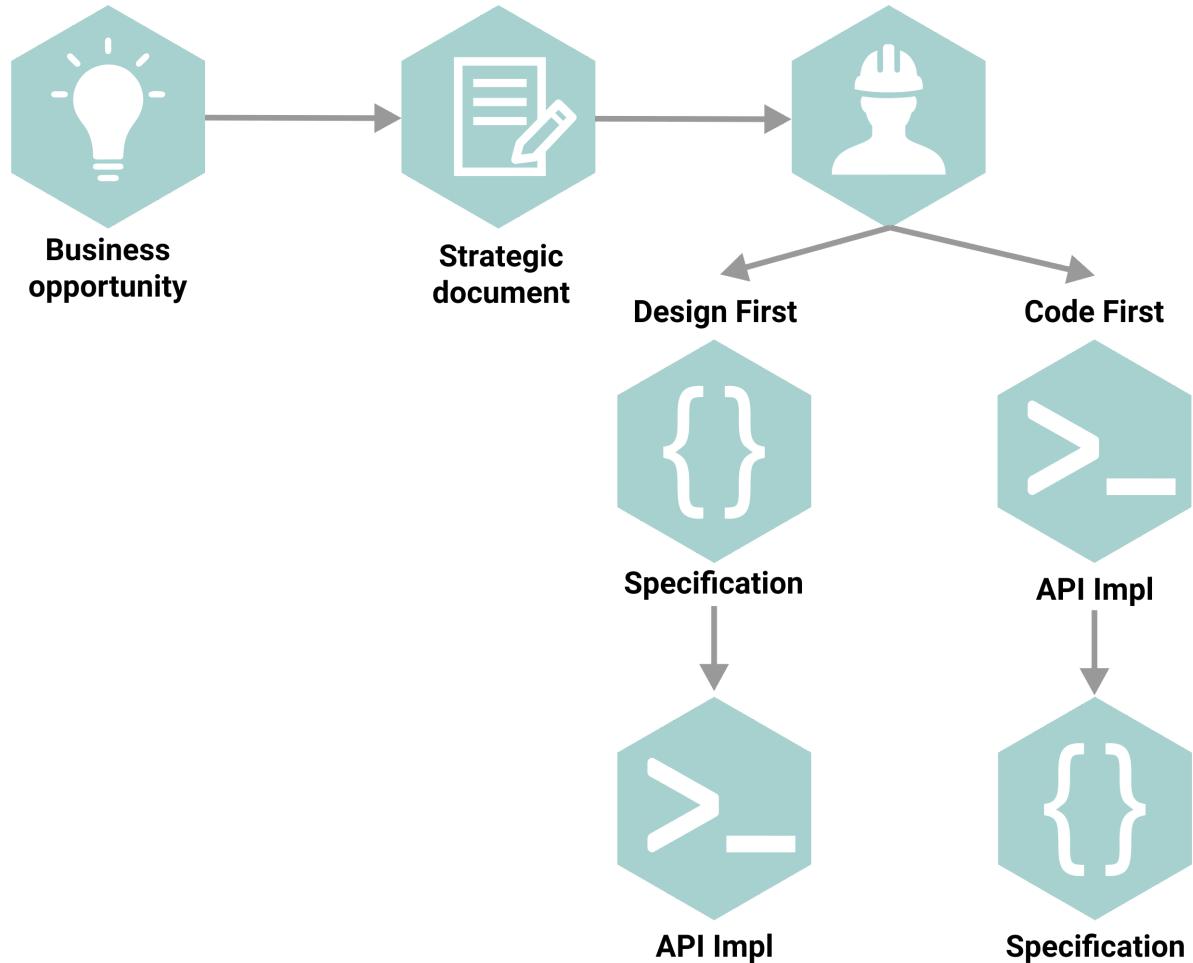
- API stakeholders should wait the implementation part
- Documentation is without business details by default
- Impossible to change documentation without developers

# WRITE OAS MANUALLY, REALLY?



# CONTRACT FIRST

## OVERVIEW



# CONTRACT FIRST

API      CONTRACT  
DESIGN  
SPECIFICATION      FIRST

OVERVIEW

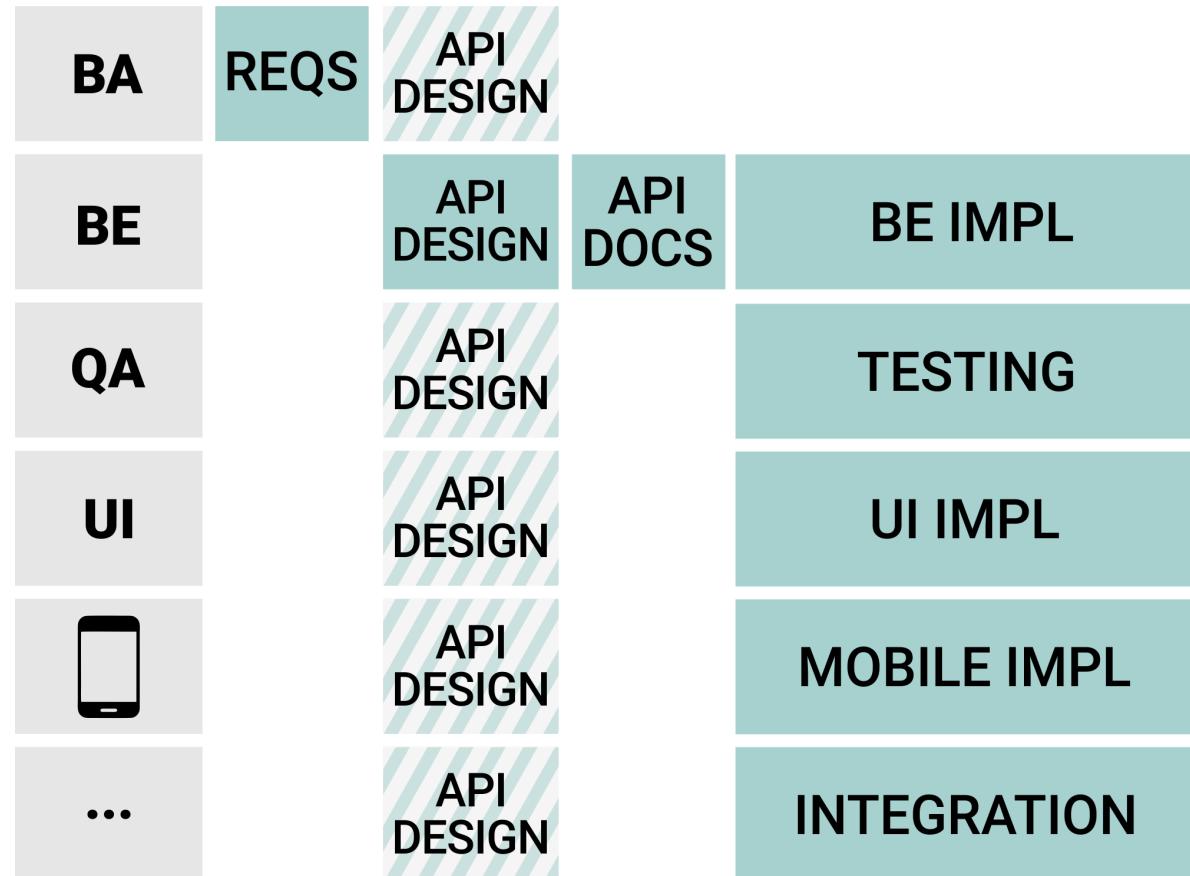
SYNOMYMS

# CONTRACT FIRST

OVERVIEW

SYNONYMS

**WORKFLOW**



# CONTRACT FIRST

OVERVIEW  
SYNONYMS  
WORKFLOW  
**PROS**

-  Contract is single source of truth
-  Parallel API-implementation and API-consumers work
-  Design phase with involvement of any stakeholder(s)
-  Not only developers can change documentation
-  (potentially) You can re-implement your application easily.
-  Complexity is immediately visible

# WHY SHOULD WE WAIT?



# CONTRACT FIRST

OVERVIEW

SYNONYMS

WORKFLOW

PROS

**CONS**

- Writing and review of specification is bottleneck of your process
- You should change your processes (not just development)
- You should be great in writing texts and soft words
- Specification review may take time to agreed on

# MAY BE TOOLING IS COOL?



# OPENAPI TOOLING

## OVERVIEW

- Converters
- Validators
- Documentation
- DSL
- Text & GUI Editors
- Mock Servers
- Parsers
- Generators
- Security
- Testing

See [openapi.tools](https://openapi.tools)

# OPENAPI TOOLING

## OVERVIEW

## TEXT EDITORS

- IntelliJ Idea
- Visual Studio
- Stoplight Desktop
- Swagger Editor

The screenshot shows the Swagger Editor application. On the left, there is a code editor window displaying an OpenAPI 3.0.0 specification for a Petstore API. The code includes definitions for the Petstore API, including paths for listing and creating pets, and responses for those operations. On the right, there is a generated API documentation page titled "Swagger Petstore". The page shows the "Servers" section with the URL "http://petstore.swagger.io/v1". Below that, the "pets" section is expanded, showing three operations: a blue "GET /pets" button labeled "List all pets", a green "POST /pets" button labeled "Create a pet", and another blue "GET /pets/{petId}" button labeled "Info for a specific pet". At the bottom, there is a "Models" section.

```
1 openapi: "3.0.0"
2 info:
3   version: 1.0.0
4   title: Swagger Petstore
5   license:
6     name: MIT
7 servers:
8   - url: http://petstore.swagger.io/v1
9 paths:
10  /pets:
11    get:
12      summary: List all pets
13      operationId: listPets
14      tags:
15        - pets
16      parameters:
17        - name: limit
18          in: query
19          description: How many items to return at one time (max 100)
20          required: false
21      schema:
22        type: integer
23        format: int32
24      responses:
25        '200':
26          description: A paged array of pets
27          headers:
28            x-next:
29              description: A link to the next page of responses
30          schema:
31            type: string
32            content:
```

# OPENAPI TOOLING

## OVERVIEW TEXT EDITORS GUI EDITORS

The screenshot shows the SwaggerHub interface for the "Swagger Petstore" API. On the left, there's a sidebar with navigation links for "Info", "Tags", "Search", and sections for "pet", "store", "user", and "Models". The main area displays the API definition in YAML format:

```
swagger: '2.0'
info:
  title: Swagger Petstore
  description: |-
    This is a sample Petstore server. You can find out more about Swagger at [http://swagger.io](http://swagger.io) or on irc.freenode.net in #swagger. For more information about these operations, refer to the Apache 2.0 license located at http://www.apache.org/licenses/LICENSE_2.0.html
  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
  # base path: /v2
  tags:
    - name: pet
      description: Everything about your Pets
    externalDocs:
      description: Find out more
      url: https://swagger.io
  store:
    - 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:
  #  - https:
  paths:
    /pet:
      get:
        summary: Finds Pets by status
        operationId: findPet
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
          '200':
            description: A pet object that needs to be added to the store
            schema:
              $ref: '#/definitions/Pet'
            examples:
              'pet': {
                'id': 1,
                'category': {
                  'id': 1,
                  'name': 'dog'
                },
                'name': 'doggie',
                'photoUrls': [
                  'http://example.com/doggie.jpg'
                ],
                'tags': [
                  {
                    'id': 1,
                    'name': 'cute'
                  }
                ],
                'status': 'available'
              }
      post:
        summary: Adds a new pet to the store
        operationId: addPet
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
          '200':
            description: successful operation
            schema:
              $ref: '#/definitions/Pet'
            examples:
              'pet': {
                'id': 1,
                'category': {
                  'id': 1,
                  'name': 'dog'
                },
                'name': 'doggie',
                'photoUrls': [
                  'http://example.com/doggie.jpg'
                ],
                'tags': [
                  {
                    'id': 1,
                    'name': 'cute'
                  }
                ],
                'status': 'available'
              }
      put:
        summary: Updates an existing pet
        operationId: updatePet
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
          '200':
            description: successful operation
            schema:
              $ref: '#/definitions/Pet'
            examples:
              'pet': {
                'id': 1,
                'category': {
                  'id': 1,
                  'name': 'dog'
                },
                'name': 'doggie',
                'photoUrls': [
                  'http://example.com/doggie.jpg'
                ],
                'tags': [
                  {
                    'id': 1,
                    'name': 'cute'
                  }
                ],
                'status': 'available'
              }
      delete:
        summary: Deletes a pet
        operationId: deletePet
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
          '200':
            description: successful operation
            schema:
              $ref: '#/definitions/Pet'
            examples:
              'pet': {
                'id': 1,
                'category': {
                  'id': 1,
                  'name': 'dog'
                },
                'name': 'doggie',
                'photoUrls': [
                  'http://example.com/doggie.jpg'
                ],
                'tags': [
                  {
                    'id': 1,
                    'name': 'cute'
                  }
                ],
                'status': 'available'
              }
    /store/order:
      get:
        summary: Finds purchases in the store
        operationId: findOrder
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/Order'
              examples:
                'order': {
                  'id': 1,
                  'petId': 1,
                  'quantity': 1,
                  'shipDate': '2014-01-01T00:00:00Z',
                  'status': 'placed',
                  'store': {
                    'name': 'store'
                  }
                }
      post:
        summary: Creates a new purchase
        operationId: createOrder
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/Order'
              examples:
                'order': {
                  'id': 1,
                  'petId': 1,
                  'quantity': 1,
                  'shipDate': '2014-01-01T00:00:00Z',
                  'status': 'placed',
                  'store': {
                    'name': 'store'
                  }
                }
    /user/login:
      get:
        summary: Logs user into the system
        operationId: loginUser
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/User'
              examples:
                'user': {
                  'id': 1,
                  'username': 'user'
                }
      post:
        summary: Creates an user account
        operationId: createUser
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/User'
              examples:
                'user': {
                  'id': 1,
                  'username': 'user'
                }
    /user/logout:
      get:
        summary: Logs out current logged in user session
        operationId: logoutUser
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/User'
              examples:
                'user': {
                  'id': 1,
                  'username': 'user'
                }
    /user/{username}:
      get:
        summary: Gets user by user name
        operationId: getUser
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/User'
              examples:
                'user': {
                  'id': 1,
                  'username': 'user'
                }
      put:
        summary: Updates an existing user in the store
        operationId: updateUser
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/User'
              examples:
                'user': {
                  'id': 1,
                  'username': 'user'
                }
      delete:
        summary: Deletes a user
        operationId: deleteUser
        consumes:
          - application/json
        produces:
          - application/xml
        responses:
            '200':
              description: successful operation
              schema:
                $ref: '#/definitions/User'
              examples:
                'user': {
                  'id': 1,
                  'username': 'user'
                }
  definitions:
    Pet:
      type: object
      required:
        - id
        - name
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
        photoUrls:
          type: array
          items:
            type: string
        tags:
          type: array
          items:
            type: string
        status:
          type: string
    Order:
      type: object
      required:
        - id
        - quantity
        - shipDate
        - status
        - store
      properties:
        id:
          type: integer
          format: int64
        quantity:
          type: integer
        shipDate:
          type: string
          format: date-time
        status:
          type: string
        store:
          type: object
          required:
            - name
          properties:
            name:
              type: string
    User:
      type: object
      required:
        - id
        - username
      properties:
        id:
          type: integer
          format: int64
        username:
          type: string
    Category:
      type: object
      required:
        - id
        - name
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
    Model:
      type: object
      required:
        - name
      properties:
        name:
          type: string
    Petstore:
      type: object
      required:
        - id
        - name
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
    
```

The right side of the interface shows the "Swagger Petstore" documentation, which includes the API title, version, and descriptions for each endpoint. It also features a "Schemes" dropdown set to "HTTPS" and a "Authorize" button.

# OPENAPI TOOLING

## OVERVIEW TEXT EDITORS GUI EDITORS

stoplight.io

The screenshot shows the Stoplight API tool interface. On the left, there's a sidebar with a tree view of the Petstore API endpoints. The main area displays the 'Finds Pets by tags' operation. It includes the URL `https://petstore.swagger.io/v2 /pet/findByTags`, method selection (GET), and a detailed description: "Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing." Below the description are sections for Security (using petstore\_auth with scopes write:pets and read:pets), Query Params (tags), and JSON Request Body/Form Data. The Responses section shows a successful 200 response and an error 400 response. The right side of the interface shows the 'Try It' button, authorization details (Implicit OAuth Flow with scopes read:pets and write:pets), and a detailed schema for the 200 response.

Finds Pets by tags

https://petstore.swagger.io/v2 /pet/findByTags Path Params

GET POST PUT PATCH DELETE HEAD OPTIONS TRACE

OPERATION ID: findPetsByTags DEPRECATED

DESCRIPTION: Multiple tags can be provided with comma separated strings. Use tag1, tag2, tag3 for testing.

+ Security + Header + Query Param

Security: petstore\_auth (write:pets, read:pets)

Query Params: tags (array required, type: string, description: Tags to filter by)

JSON Request Body Form Data

+ Response 200 400 200

successful operation

Headers +

Responses: 200 successful operation, 400

Schema: array[object] { id: integer <int64>, category: object (2) id: integer <int64> }

Docs Try It

Form Code Preview 23 Problems

# OPENAPI TOOLING

OVERVIEW

TEXT EDITORS

GUI EDITORS

VALIDATORS

- [github.com/openapitools/openapi-style-validator](https://github.com/openapitools/openapi-style-validator)
- [stoplight.io/open-source/spectral](https://stoplight.io/open-source/spectral)

```
● ● ●

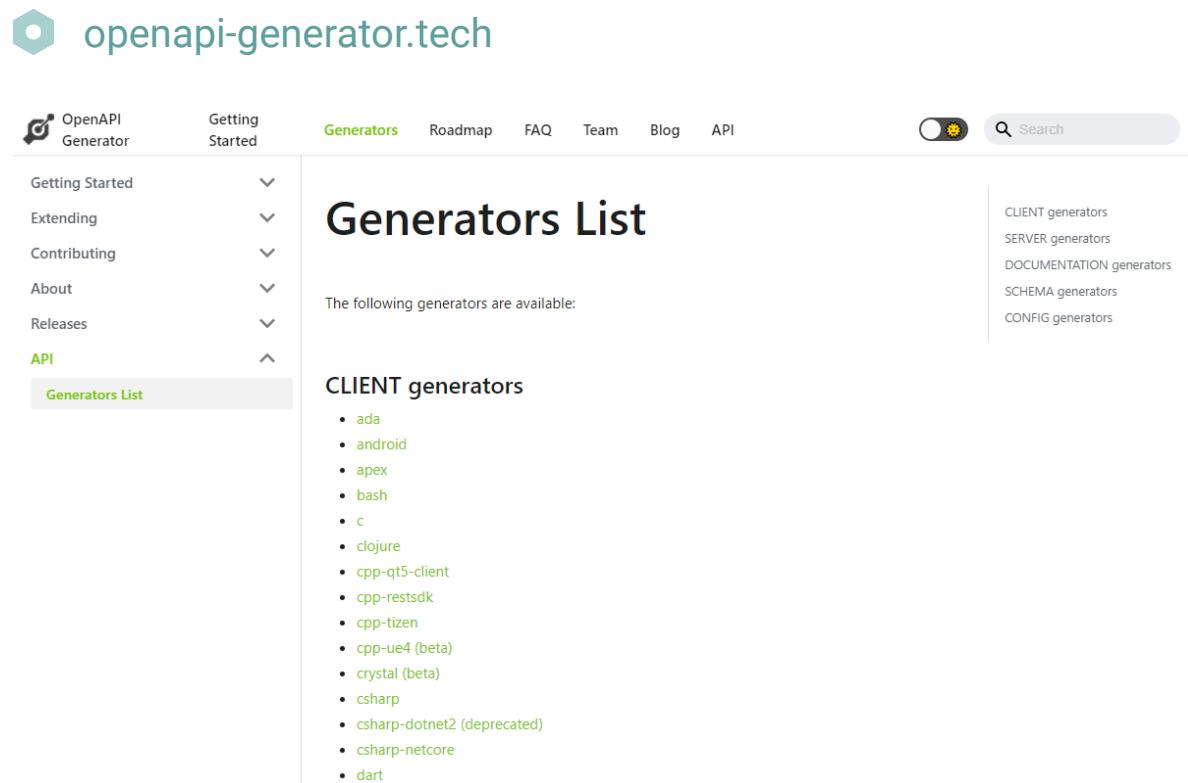
► spectral lint petstore.yaml
Adding OpenAPI 3.x functions
OpenAPI 3.x detected

petstore.yaml
  2:6  warning  info-contact          Info object should contain `contact` object.
  2:6  warning  info-description      OpenAPI object info `description` must be present and non-empty string.
  11:9  warning  operation-description Operation `description` must be present and non-empty string.
  42:10  warning  operation-description Operation `description` must be present and non-empty string.
  57:9  warning  operation-description Operation `description` must be present and non-empty string.

✖ 5 problems (0 errors, 5 warnings, 0 infos)
```

# OPENAPI TOOLING

OVERVIEW  
TEXT EDITORS  
GUI EDITORS  
VALIDATORS  
**GENERATORS**



The screenshot shows the homepage of [openapi-generator.tech](https://openapi-generator.tech). The header includes the site logo, navigation links for Getting Started, Generators, Roadmap, FAQ, Team, Blog, and API, and a search bar. A sidebar on the left provides links to documentation sections like Getting Started, Extending, Contributing, About, and Releases, along with an API link and a "Generators List" button. The main content area features a large heading "Generators List" and a sub-section titled "CLIENT generators" listing various generator names.

**Generators List**

The following generators are available:

**CLIENT generators**

- ada
- android
- apex
- bash
- c
- clojure
- cpp-qt5-client
- cpp-restsdk
- cpp-tizen
- cpp-ue4 (beta)
- crystal (beta)
- csharp
- csharp-dotnet2 (deprecated)
- csharp-netcore
- dart

CLIENT generators  
SERVER generators  
DOCUMENTATION generators  
SCHEMA generators  
CONFIG generators

# OPENAPI TOOLING

OVERVIEW  
TEXT EDITORS  
GUI EDITORS  
VALIDATORS  
GENERATORS  
**COVERAGE**

 [github.com/viclovsky/swagger-coverage](https://github.com/viclovsky/swagger-coverage)

Swagger Petstore 1.0.3 [Summary](#) [Operations details](#) [Tags details](#) [Conditions details](#) [Generation info](#)

### Operations details

All: 20 Full: 1 Partial: 6 Empty: 13 No call: 12 Missed: 1

Condition name	Details
✓ HTTP status 200	
✗ HTTP status 400	
✓ query «status» is not empty	
✗ query «status» contains all values from enum [available, pending, sold]	Missed values [sold]
✓ only declared status	
✗ query «status» contains values not only from enum	Checked values: [pending, available]

Full	GET	/pet/{petId}	2 calls	5/5 (100%) condition covered
Partial	GET	/pet/findByStatus	2 calls	3/6 (50%) condition covered
Partial	GET	/pet/findByTags	1 calls	3/4 (75%) condition covered
Partial	POST	/pet/{petId}/uploadImage	1 calls	1/5 (20%) condition covered
Partial	GET	/user/{username}	1 calls	3/5 (60%) condition covered
Partial	PUT	/user/{username}	1 calls	1/5 (20%) condition covered

# OPENAPI TOOLING

OVERVIEW

TEXT EDITORS

GUI EDITORS

VALIDATORS

GENERATORS

COVERAGE

MOCKING

- ➊ [github.com/danielgtaylor/apisprout](https://github.com/danielgtaylor/apisprout)
- ➋ [stoplight.io/open-source/prism](https://stoplight.io/open-source/prism)

```
[→ Stoplight prism mock --help
Start a mock server with the given spec file

USAGE
$ prism mock SPEC

ARGUMENTS
SPEC Path to a spec file. Can be both a file or a fetchable resource on the web

OPTIONS
-d, --dynamic      Dynamically generate examples.
-h, --host=host    [default: 127.0.0.1] Host that Prism will listen to.
-m, --multiprocess Fork the http server from the CLI
-p, --port=port     (required) [default: 4010] Port that Prism will run on.
```

# OPENAPI TOOLING

OVERVIEW

TEXT EDITORS

GUI EDITORS

VALIDATORS

GENERATORS

COVERAGE

MOCKING

VERSIONING

 [github.com/azure/openapi-diff](https://github.com/azure/openapi-diff)

```
vishrut@visshamac openapi-diff $ oad compare --help
Commands:
  compare <old-spec> <new-spec>  Compares old and new open api specification for
                                     breaking changes.

Options:
  --version          Show version number                                [boolean]
  -l, --logLevel    Set the logging level for console.
                     [choices: "off", "json", "error", "warn", "info", "verbose", "debug", "silly"]
                     [default: "warn"]
  -f, --logFilePath Set the log file path. It must be an absolute filepath. By
                     default the logs will stored in a timestamp based log file
                     at "/Users/vishrut/oad_output".
  -j, --inJson       A boolean flag indicating whether output format of the
                     messages is json.                               [boolean] [default: true]
  -h, --help         Show help                                         [boolean]
  -o, --oldTagName  The tag name for the old specification file. If include it
                     indicates that the old spec file is a readme file
  -n, --newTagName  The tag name for the new specification file. If include it
                     indicates that the new spec file is a readme file
```

# OPENAPI TOOLING

OVERVIEW

TEXT EDITORS

GUI EDITORS

VALIDATORS

GENERATORS

COVERAGE

MOCKING

VERSIONING

SECURITY

- 42crunch.com/api-security-audit
- zaproxy.org/docs/desktop/addons/openapi-support
- apisecurity.io/tools/audit

The screenshot shows the APISecurity.io Security Audit interface. At the top, it displays a score of "15 out of 100". Below this, there are two sections: "Security" and "Data validation". The "Security" section includes categories like Authentication (21 issues), Authorization (0 issues), and Transport (1 issue). The "Data validation" section includes categories like Parameters (0 issues), Response definition (88 issues), Response headers (0 issues), and Schema (115 issues). On the right side, a detailed error message is shown for a "Schema allows additional properties" issue. It includes a code snippet from the user's API definition, a severity of "High", and an impact score of "4". The error message states: "The schema you have defined allows additional properties, either intentionally or unintentionally. Unlike in the OpenAPI Specification (OAS) v2, in OAS v3 it is not enough to just state the type of the properties in the schema. By default, the property `additionalProperties` is `true`. Unless you specifically set `additionalProperties` to `false`, the schema continues to accept properties of any type. However, sometimes you intentionally want to allow additional properties, especially when using the combining operations `allOf`, `anyOf`, or `oneOf`".

# PITFALLS OF STRICT CONTRACT FIRST

- You should be ready for changes in your DEV/QA/BA/Workflow/etc. processes
- Text writing skill is really important
- Our design mistakes become more transparent
- Focus only on API design in OAS and don't expose impl details
- Background scheduled jobs are not supported
- OAS doesn't cover async messaging between apps
- GraphQL API is not supported out of the box
- Tooling are still in development

# CHOOSING THE RIGHT APPROACH

## CONTRACT FIRST WHEN

- ❖ Developer Experience Matters
- ❖ Delivering Mission-Critical APIs
- ❖ Ensuring Good Communication

## CODE FIRST WHEN

- ❖ Delivery Speedy Matters
- ❖ Developing Internal APIs

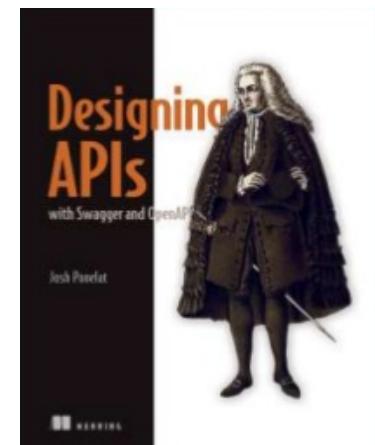
# CONCLUSION

- No documentation ⇒ Code First ⇒ Contract First
- Initial problem was solved and we can specify, design and parallelize API-implementation and API-consumers work
- It's still possible to use hybrid approach

# BONUS LINKS

## USEFUL LINKS

- ❶ [idratherbewriting.com/learnapidoc](http://idratherbewriting.com/learnapidoc)
- ❶ [apisyouwonthate.com/blog](http://apisyouwonthate.com/blog)
- ❶ [apihandyman.io/blog](http://apihandyman.io/blog)
- ❶ [apis.guru/awesome-openapi3](http://apis.guru/awesome-openapi3)
- ❶ API the Docs Youtube Channel



## WHAT ELSE?

- ❶ AsyncAPI - [asyncapi.com/docs/specifications/2.0.0](http://asyncapi.com/docs/specifications/2.0.0)
- ❶ SLA for OpenAPI - [github.com/isa-group/SLA4OAI-Specification](https://github.com/isa-group/SLA4OAI-Specification)