90%

De los developers usan APIs

19th Developer Economics Survey - Slashdata

PYCONES GRANADA 2022

La magia de OpenAPI Specification

Benjamin Granados *(he/him)*

Developer Evangelist twilio

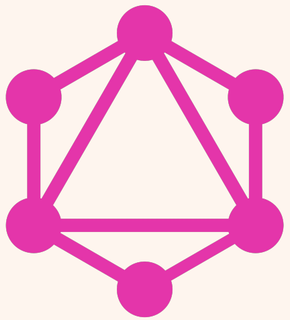
@benjagm

Las APIs hacen la vida más fácil a los developers

API Styles

REST

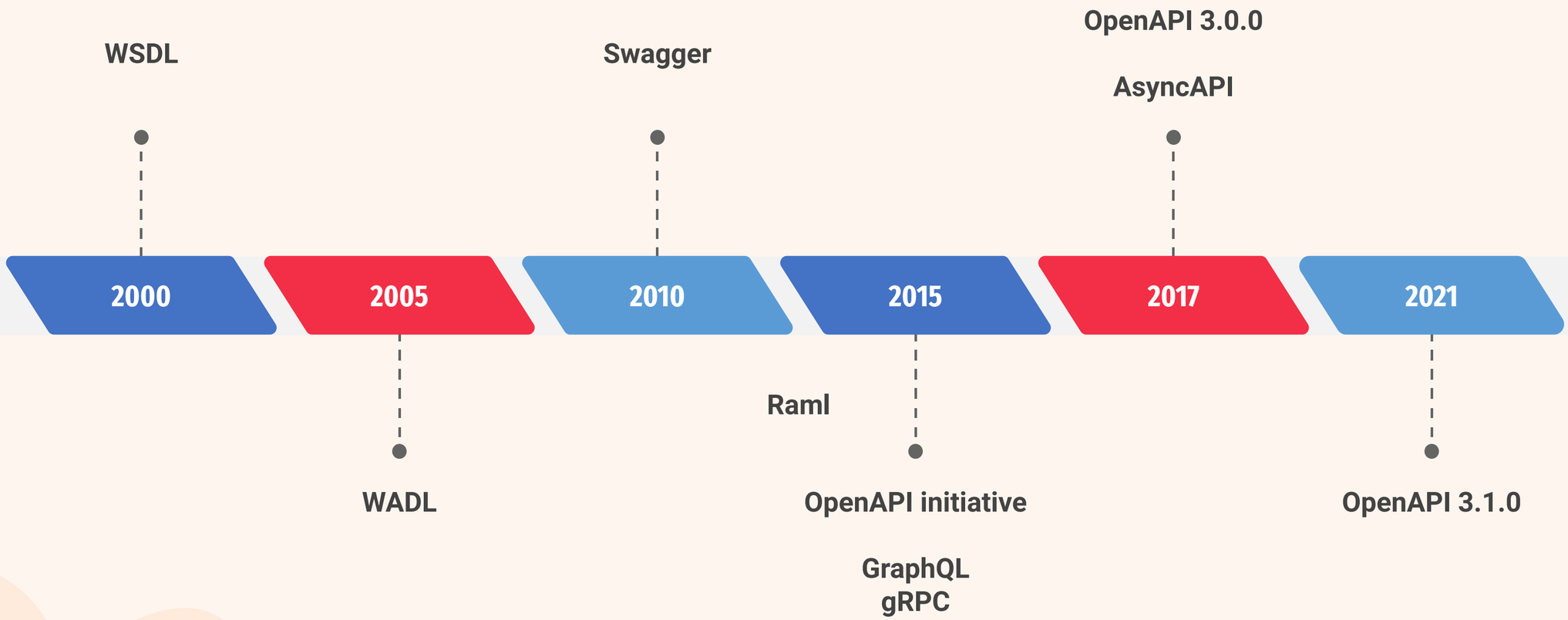
Async



graphQL

gRPC

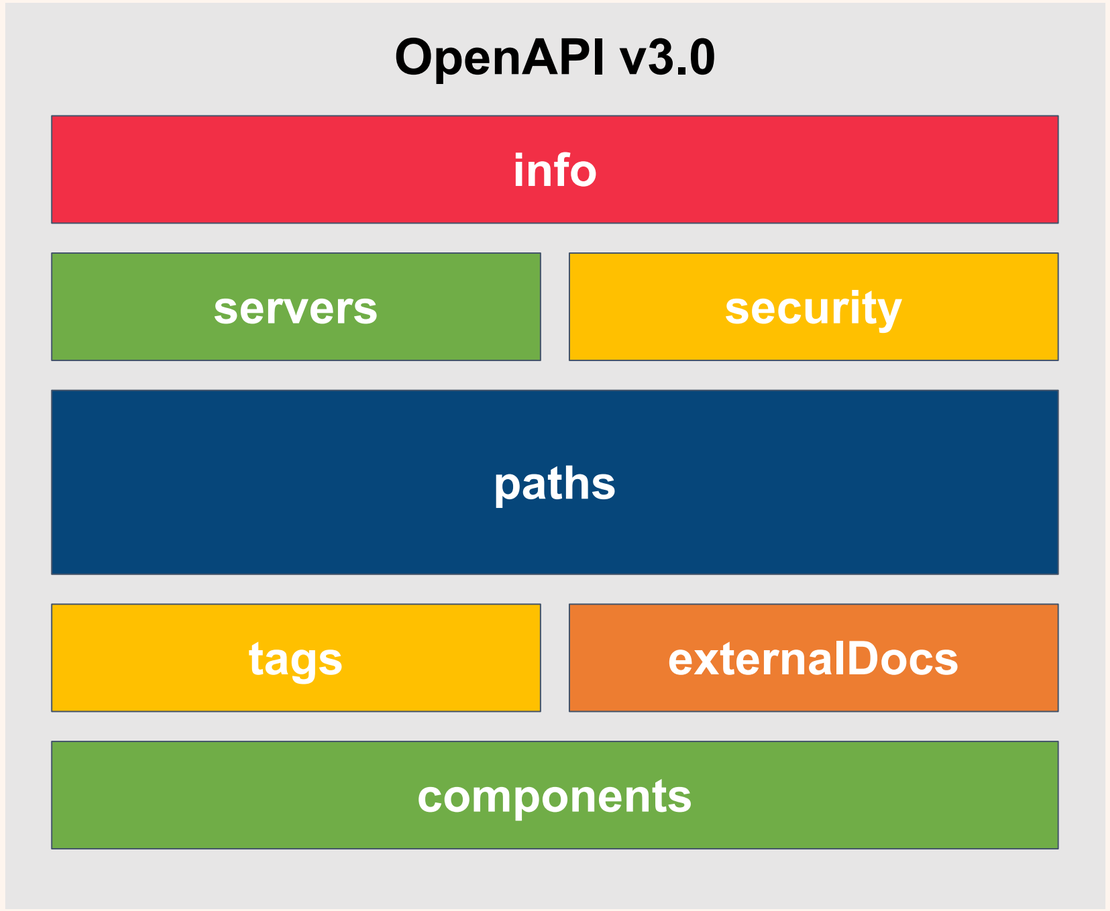
Un poco de historia



THE LINUX FOUNDATION

@benjagm

OpenAPI Specification



YAML

```yaml
openapi: 3.0.0
info:
  title: Sample API
  description: Multiline/single-line description in Common Mark or HTML.
  version: 0.1.9

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

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

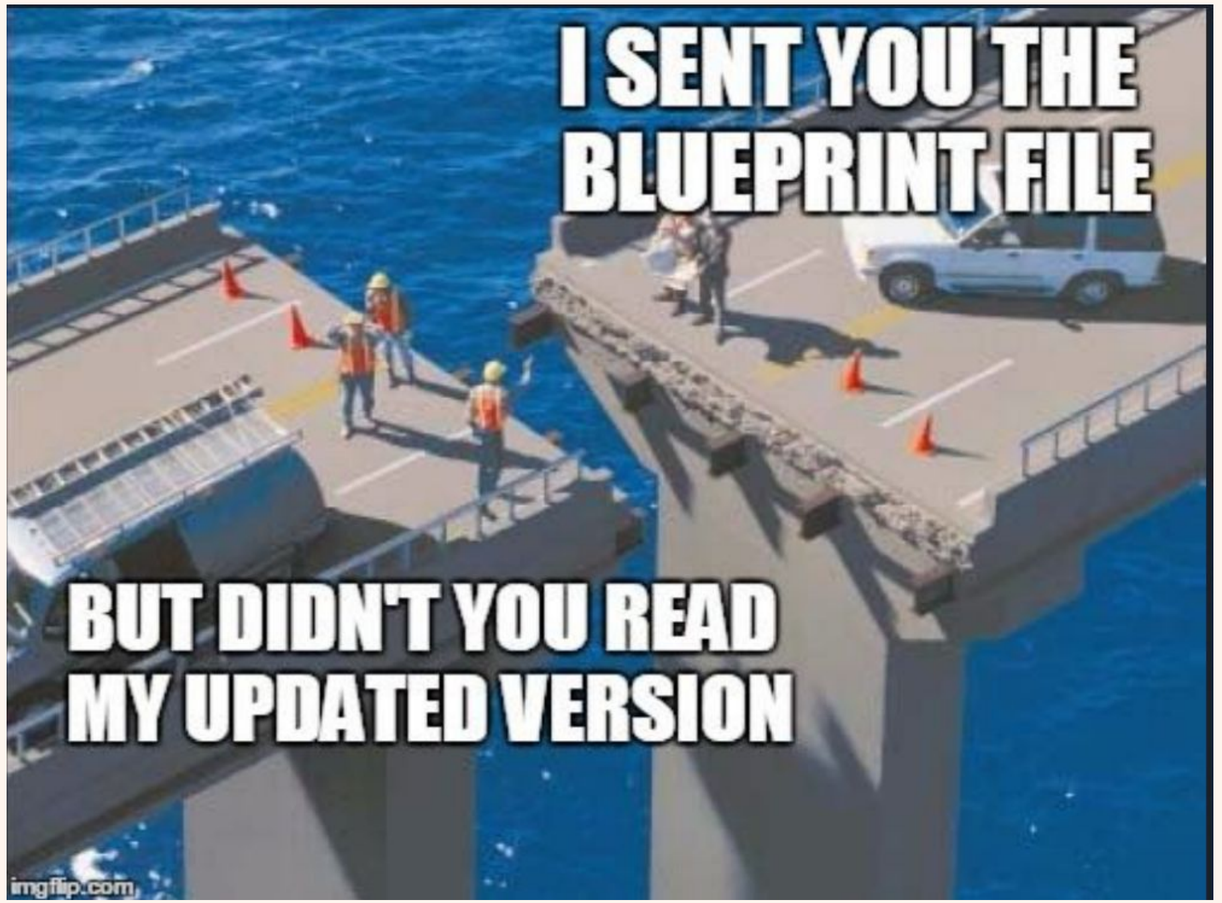
¿Por qué OpenAPI es tan importante?

- **Confiable: Estabilidad + Base de usuarios enorme**
- **Mejor Developer Experience**
- **Facilita el gobierno y la colaboración**

Spec-Driven Dev 👍

VS

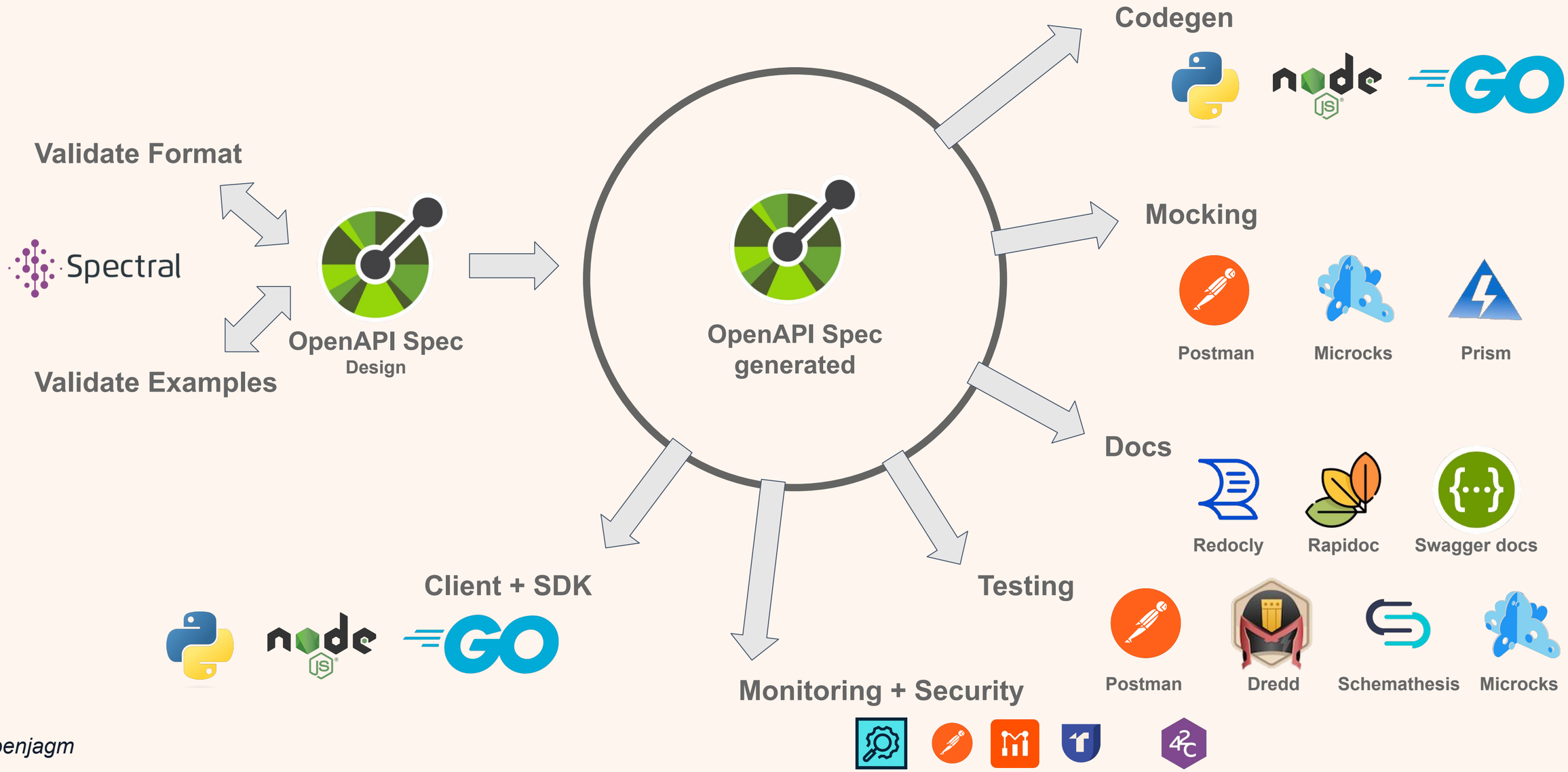
Implementation first 🤞



OpenAPI Specification alimenta el ciclo de vida de las APIs

https://www.youtube.com/watch?v=-X1X7HuFHSg

OpenAPI Specification alimenta el ciclo de vida de las APIs



API Ops

Diseñar, desarrollar, probar y desplegar APIs más rápido, más veces y mejor.

Encontrando las herramientas adecuadas

https://openapi.tools/

Tool Types

- **Auto Generators:** Tools that will take your code and turn it into an OpenAPI Specification document.
- **Converters:** Various tools to convert to and from OpenAPI and other API description formats.
- **Data Validators:** Check to see if API requests and responses are lining up with the API description.
- **Description Validators:** Check your API description to see if it is valid OpenAPI.
- **Documentation:** Render API Description as HTML (or maybe a PDF) so slightly less technical people can figure out how to work with the API.
- **DSL:** Domain Specific Language to write OpenAPI in your language of choice.
- **GUI Editors:** Visual editors help you design APIs without needing to memorize the entire OpenAPI specification.
- **Miscellaneous:** Anything else that does stuff with OpenAPI but hasn't quite got enough to warrant its own category.
- **Mock Servers:** Fake servers that take description document as input, then route incoming HTTP requests to example responses.
- **Parsers:** Loads and read OpenAPI descriptions, so you can work with them programmatically.
- **SDK Generators:** Generate code to give to consumers, to help them avoid interacting at a HTTP level.
- **Security:** By poking around your OpenAPI description, some tools can look out for attack vectors you might not have noticed.
- **Server Implementations:** Easily create and implement resources and routes for your APIs.
- **Testing:** Quickly execute API requests and validate responses on the fly through command line or GUI interfaces.
- **Text Editors:** Text editors give you visual feedback whilst you write OpenAPI, so you can see what docs might look like.
- **Learning:** Whether you're trying to get documentation for a third party API based on traffic, or are trying to switch to design-first at an organization with no OpenAPI at all, learning can help you move your API spec forward and keep it up to date

Linting con Spectral

```
jakub@MacBook-Pro Developer % spectral lint hello-world.yaml

/Users/jakub/Developer/hello-world.yaml
  1:1  warning  asyncapi-servers               AsyncAPI object must have non-empty "servers" object.
  1:1  warning  asyncapi-tags                  AsyncAPI object must have non-empty "tags" array.
  2:6  warning  asyncapi-info-contact          Info object must have "contact" object.                        info
  2:6  warning  asyncapi-info-description      Info "description" must be present and non-empty string.       info
  2:6  warning  asyncapi-info-license          Info object must have "license" object.                        info
 4:12    error  valid-document-version         Version must match 1.x.x                                       info.version
 7:13  warning  asyncapi-operation-description Operation "description" must be present and non-empty string.  channels.hello.publish
 7:13    error  asyncapi-operation-operationId Operation must have "operationId".                             channels.hello.publish

✖ 8 problems (2 errors, 6 warnings, 0 infos, 0 hints)
```

https://github.com/stoplightio/spectral

Mocking con Prims

```
➜ Stoplight prism mock petstore.yml
[CLI] …  awaiting  Starting Prism…
[HTTP SERVER] i  info      Server listening at http://127.0.0.1:4010
[CLI] •  note      GET         http://127.0.0.1:4010/pets
[CLI] •  note      POST        http://127.0.0.1:4010/pets
[CLI] •  note      GET         http://127.0.0.1:4010/pets/{petId}
[CLI] ▶  start     Prism is listening on http://127.0.0.1:4010
[HTTP SERVER] get /pets i  info      Request received
    [NEGOTIATOR] i  info      Request contains an accept header: */*
    [VALIDATOR] ⚠  warning   Request did not pass the validation rules
    [NEGOTIATOR] •  note      Unable to find a 422 response definition
    [NEGOTIATOR] •  note      Unable to find a 400 response definition.
    [NEGOTIATOR] ✔  success   Created a 422 from a default response
    [NEGOTIATOR] •  note      Unable to find a content with an example defined for the response 422
    [NEGOTIATOR] ✔  success   The response 422 has a schema. I'll keep going with this one
    [NEGOTIATOR] ✔  success   Responding with the requested status code 422
```

https://github.com/stoplightio/prism

Presente de OpenAPI (Version 3.1)

- **Compatibilidad completa con JSON Schema**
- **Mejor especificación de Webhooks**
- **Reusabilidad**

Presente de OpenAPI (Version 3.1)

```yaml
openapi: 3.1.0
info:
  title: My Demo API
  version: 1.0.0
  summary: An API  with examples of features in 3.1
webhooks:
  $ref: '#/components/pathItems/newThingAlert'
components:
  pathItems:
    newThingAlert:
      summary: Notification that a new thing has been created
      post:
        requestBody:
          content:
            applicaton/json:
              schema:
                type: object
                properties:
                  thingName: null
                  type: string
```

Webhooks como campos de nivel superior

Path items reutilizables

Agenda OpenAPI para futuras versiones

- **Terminar la especificación de Overlay**
- **Soporte para RPC sobre HTTP**

Que retos enfrenta OpenAPI

- **Completar la migración de Swagger a OpenAPI**
- **Transición a un modelo de gobierno más democrático**

Claves

- **OpenAPI 3.1 es una buena versión**
- **Importancia de apoyarse en el tooling del ecosistema**
- **Completar la migración de Swagger a OpenAPI**

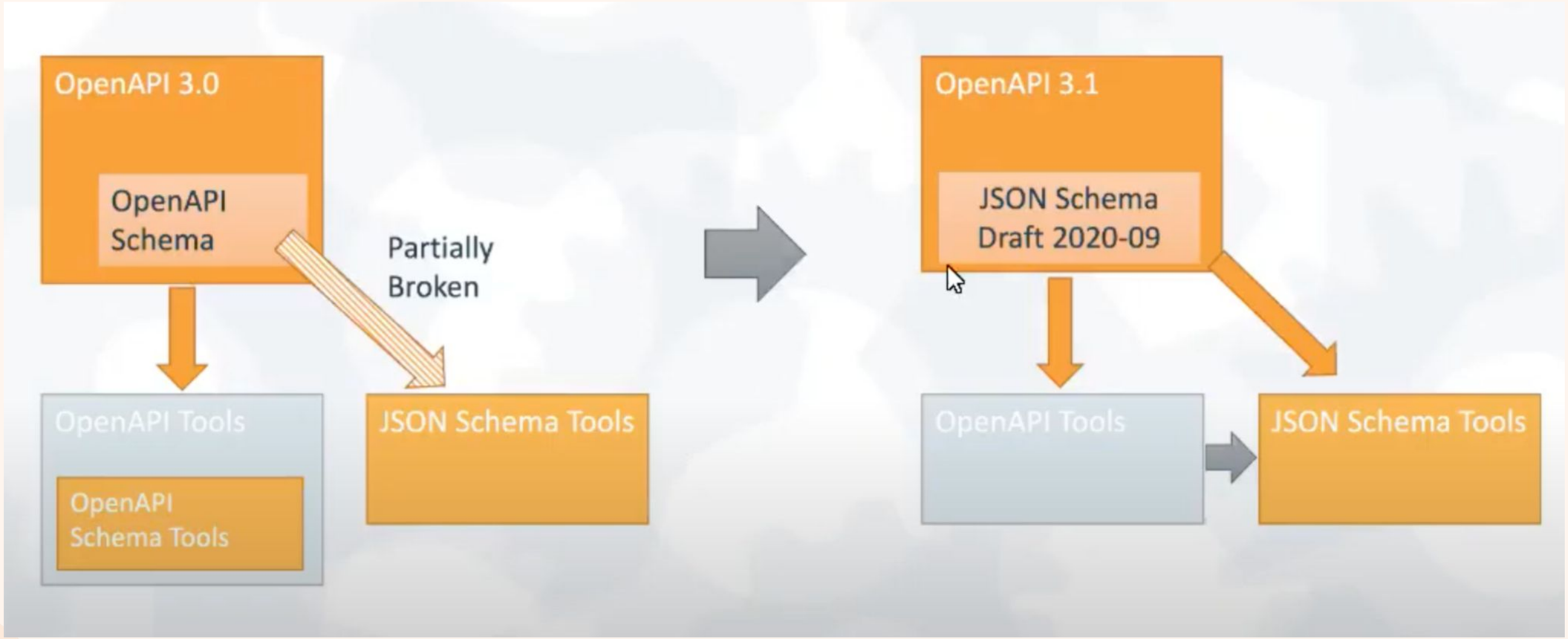
Recursos

https://github.com/benjagm/pycones22-openapi-pow

Introduction to OpenAPI Initiative here
What is OpenAPI Specification here
Benefits of OpenAPI Specification here
Standarized API Lifecycle here
Changelog OAS 3.1.0 here
Introduction to OAS 3.1.0 here
Swagger vs OpenAPI here
Introduction to Specification Driven development here
Documentation-driven development for Python web APIs v2 here
OpenAPI Specification as Mentos & Coke video here

Thank you

Presente de OpenAPI (Version 3.1)



OpenAPI Future

- Overlays: Separate document that augments another API description
- Reusable groups: $ref more than one component
- Alternative Schemas
- Optional and Multi-segment Paths
- Disambiguating based on query
- Digital Signatures and Encryption
- Discovery mechanism for security credentials (jwt, apikey, etc)

OpenAPI Specification alimenta el ciclo de vida de las APIs