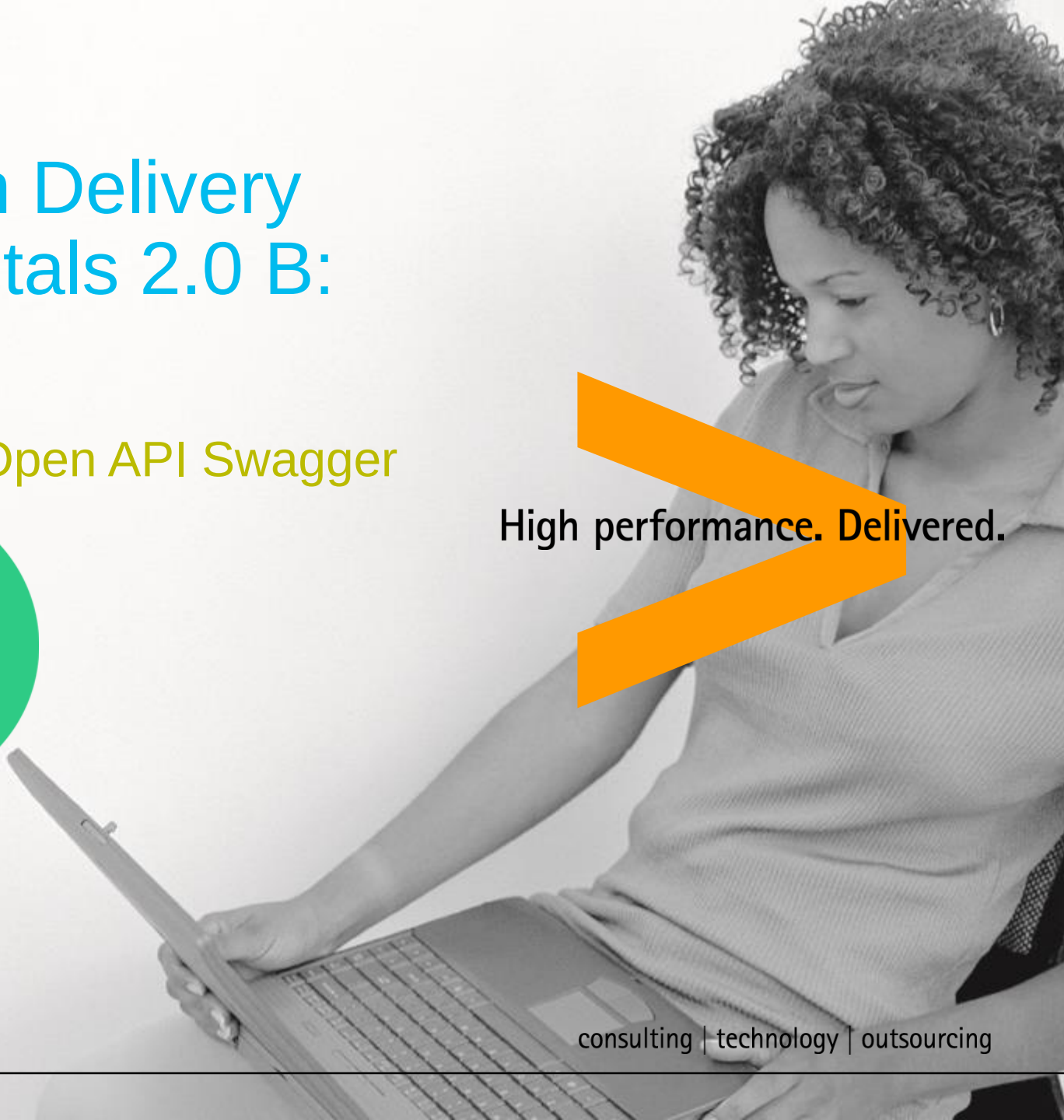
Application Delivery Fundamentals 2.0 B: Java

Introduction to Open API Swagger



High performance. Delivered.

consulting | technology | outsourcing

Course Goals / Objectives

- At the end of this module, participants will be able to understand:
  - Swagger
  - Getting set up to make API Requests
  - First OpenAPI definitions
  - Using SwaggerEditor to write OpenAPI definitions
  - Describing API responses
  - Creating resources
  - Adding Authentication and Authorization

Course Goals / Objectives

- Preparing and hosting API documentation
- Designing a web application
- Creating an API design using OpenAPI
- Building a change workflow around API Design First
- Implementing frontend code and reacting to changes
- Building a Backend with Swagger Codegen
- Integrating and releasing the web application
- The API Design First approach

Swagger

- Swagger is an alternative format to API Blueprint for describing your API that you can use in Apiary.
- Swagger is opensourced format for describing APIs.
- Swagger (Open API) is a language-agnostic specification for describing REST APIs.
- It allows both computers and humans to understand the capabilities of a REST API without direct access to the source code.
- Its main goals are to:
  - Minimize the amount of work needed to connect decoupled services.
  - Reduce the amount of time needed to accurately document a service.

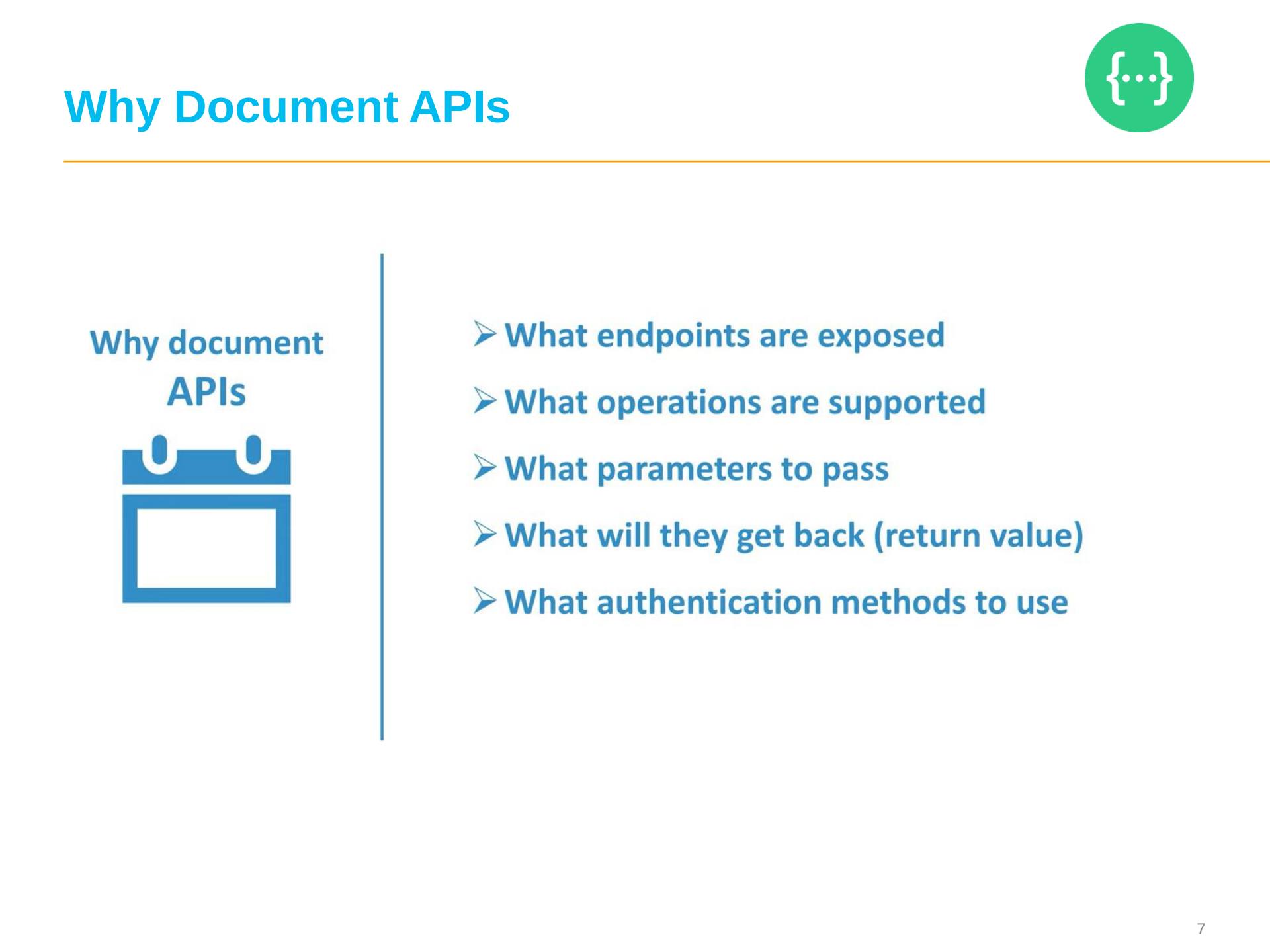
Swagger

- Swagger is the tooling ecosystem for developing APIs with the Open API Specification (OAS).
- Swagger consists of both open source as well as professional tools, catering to almost every need and use case.
- Swagger used to consist of the specification and a large ecosystem of tools to implement the specification.
- These tools include everything from front-end user interfaces, low-level code libraries and commercial API management solutions

Swagger

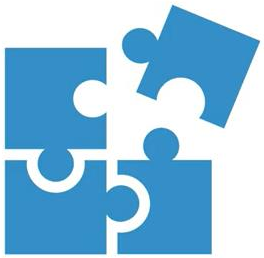
- In 2015, Smart Bear Software donated the Swagger specification to the Linux Foundation.
- It renamed the specification to the Open API Specification.
- Smart Bear also became the founding member of the Open API Initiative (OAI).
- It is a body to govern the development of the OAS in an open and transparent manner.

Why Document APIs



- What endpoints are exposed
- What operations are supported
- What parameters to pass
- What will they get back (return value)
- What authentication methods to use

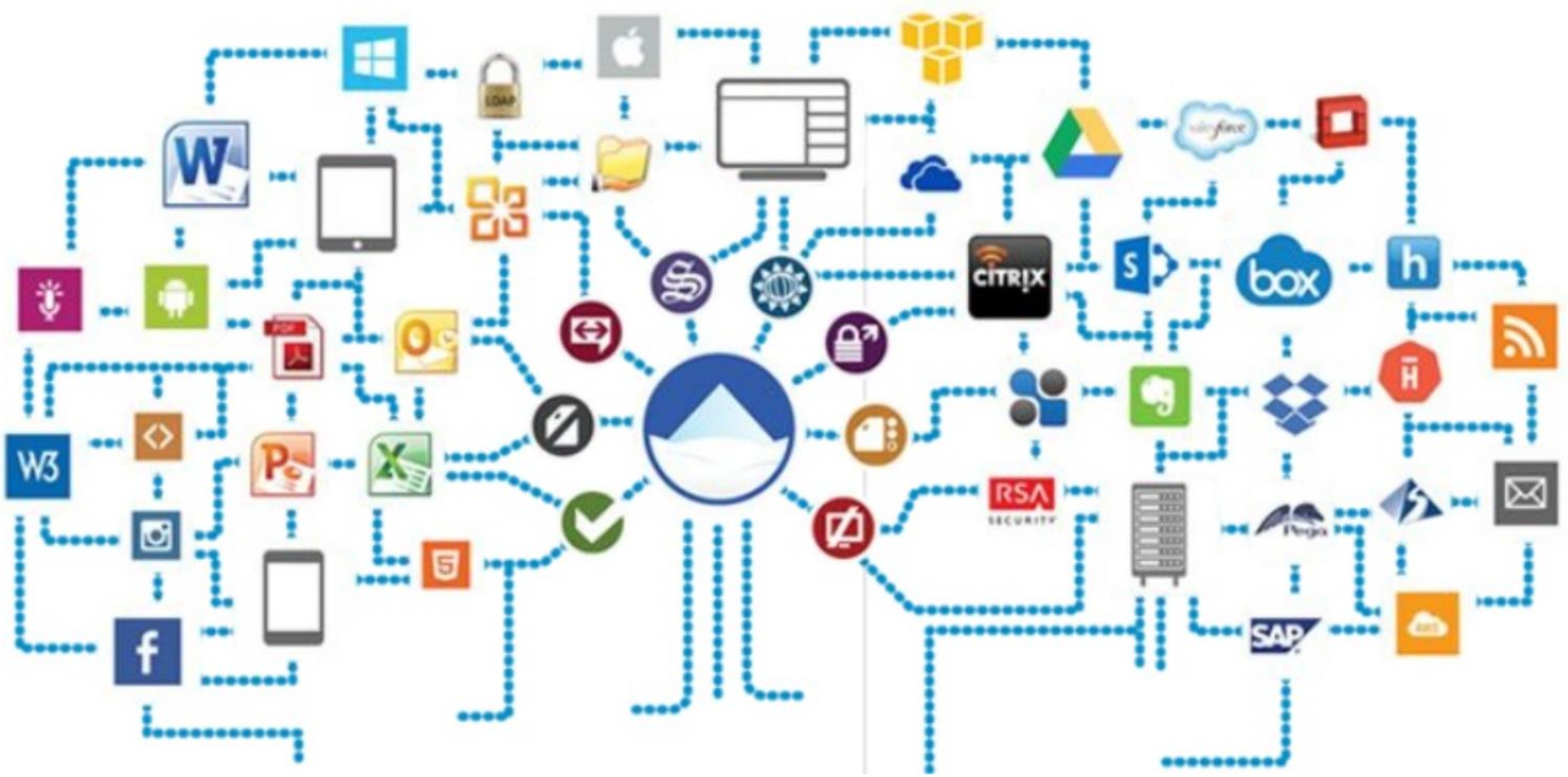
Specification



What is a specification ?

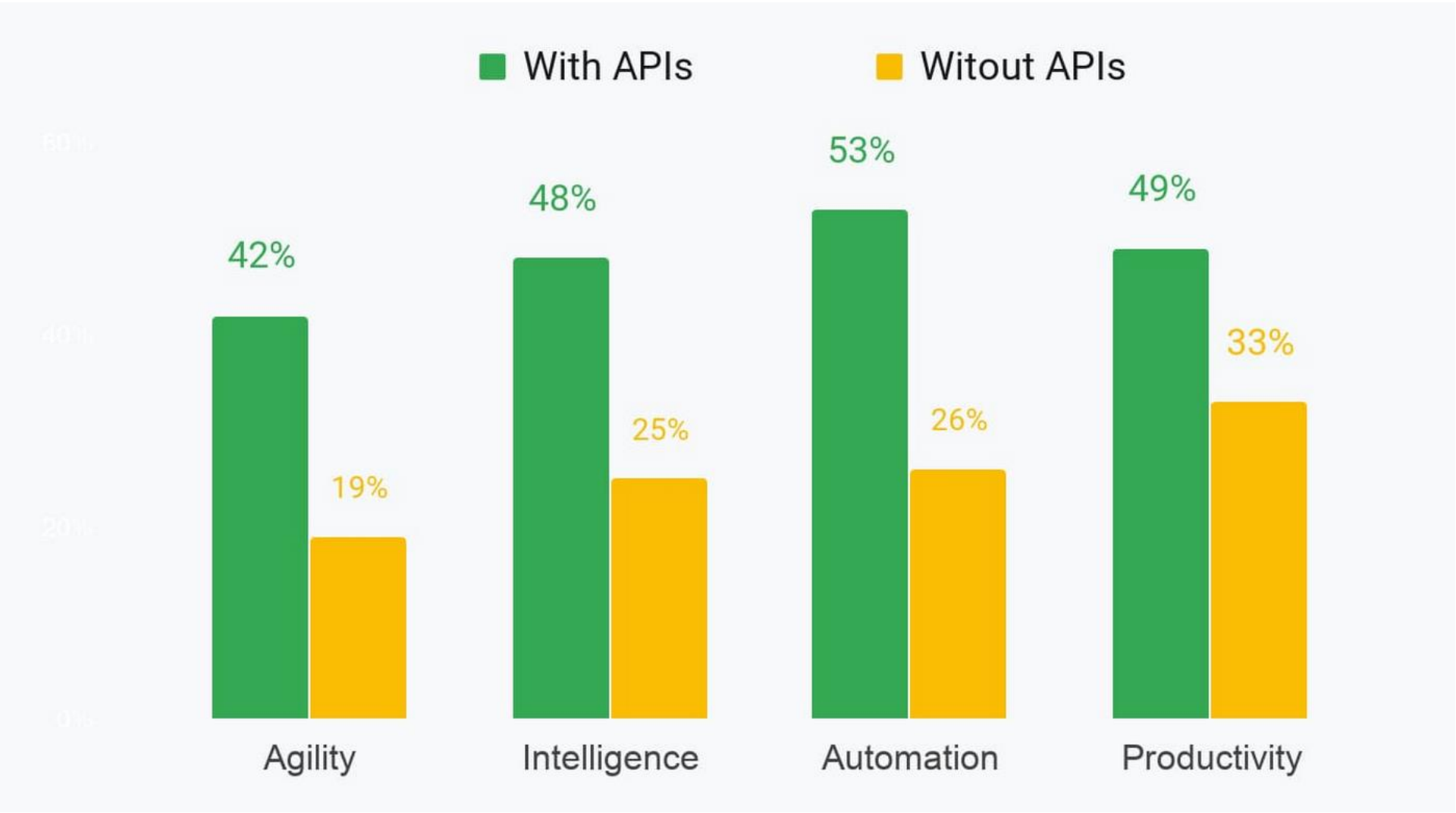
Set of rules to do something

API ecosystem: What's it all about?



API ecosystem: What's it all about?

API ecosystem: What's it all about?



Open API Specification

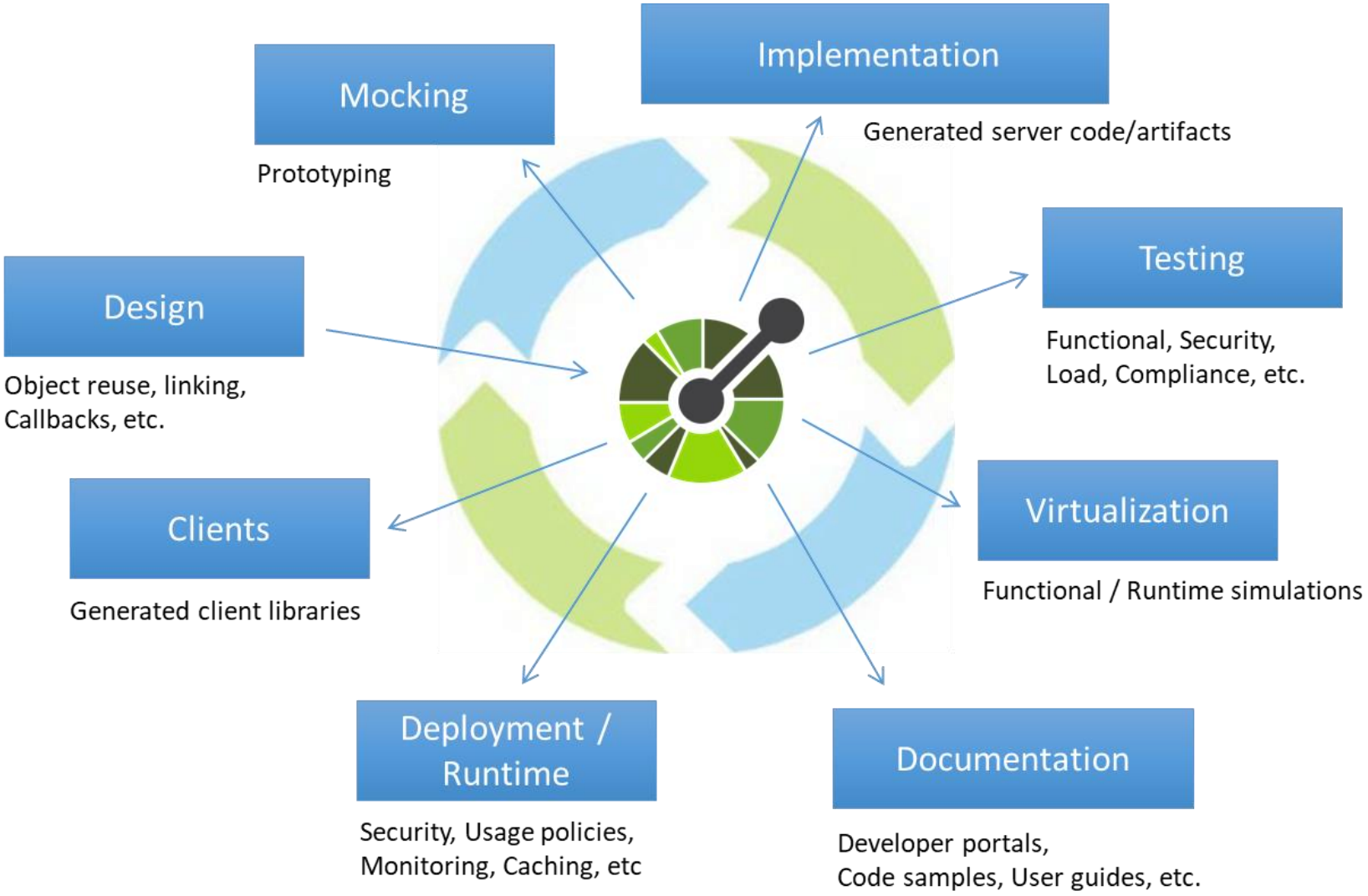
- Enter the Open API Specification (OAS)
- The OAS is to REST what WSDL was to SOAP.
- It provides a common framework of designers, developers, testers, and devops to build and maintain APIs.
- Think of the specification as a set of rules to build and implement a REST API.
- The OAS is language agnostic and is both human and machine readable.
- It allows both people and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic.

Open API Specification

Set of rules to describe RESTful APIs

- Available endpoints
- Available operations at each endpoint
- What parameters to pass and their data types
- What will the API return and the data type
- Authentication methods to use

Open API Specification



Swagger vs Open API

- The easiest way to understand the difference is:
  - Open API = Specification
  - Swagger = Tools for implementing the specification
- The development of the specification is fostered by the Open API Initiative, which involves more the 30 organizations from different areas of the tech world — including Microsoft, Google, IBM, and CapitalOne.
- Smart bear Software leads the development of the Swagger tools, is also a member of the Open API Initiative, helping lead the evolution of the specification.

Swagger vs Open API

Specification

Tools

Swagger vs Open API

- Swagger is the name associated with some of the most well-known, and widely used tools for implementing the Open API specification.
- The Swagger toolset includes a mix of open source, free, and commercial tools, which can be used at different stages of the API lifecycle.

Swagger vs Open API

- These tools include:
  - Swagger Editor: Swagger Editor lets you edit OpenAPI specifications in YAML inside your browser and to preview documentations in real time.
  - Swagger UI: Swagger UI is a collection of HTML, Javascript, and CSS assets that dynamically generate beautiful documentation from an OAS-compliant API.
  - Swagger Codegen: Allows generation of API client libraries (SDK generation), server stubs and documentation automatically given an OpenAPI Spec.
  - Swagger Parser: Standalone library for parsing OpenAPI definitions from Java

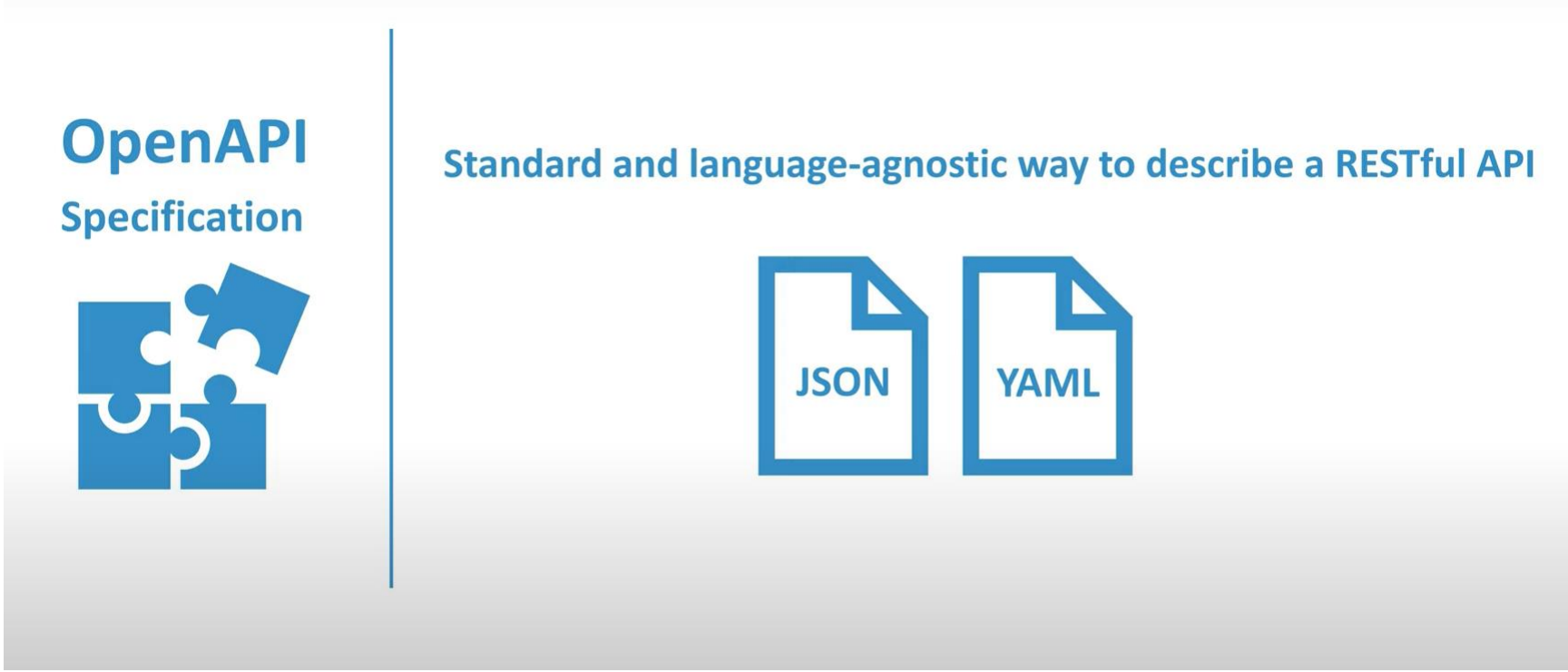
Swagger vs Open API

- Swagger Core: Java-related libraries for creating, consuming, and working with OpenAPI definitions
- Swagger Inspector (free): API testing tool that lets you validate your APIs & generate OpenAPI definitions from an existing API
- Swagger Hub (free and commercial): API design and documentation, built for teams working with OpenAPI.

Where do OpenAPI definitions fit in ?

- OpenAPI Specification (formerly Swagger Specification) is an API description format for REST APIs.
- An OpenAPI file allows you to describe your entire API, including:
  - Available endpoints (/users) and operations on each endpoint (GET /users, POST /users)
  - Operation parameters Input and output for each operation
  - Authentication methods
  - Contact information, license, terms of use and other information.

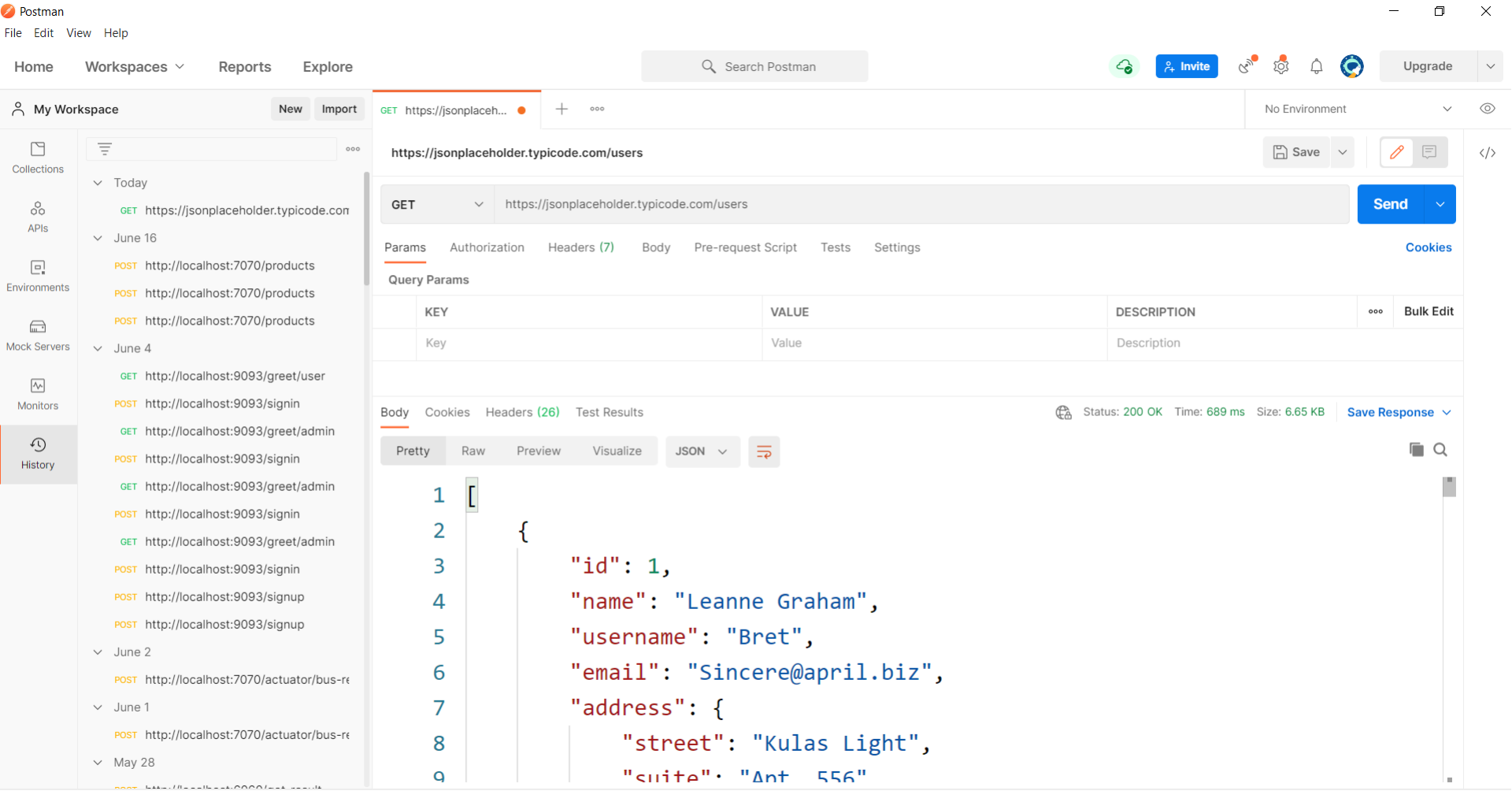
Open API Specification



Where do OpenAPI definitions fit in ?

- The ability of APIs to describe their own structure is the root of all awesomeness in OpenAPI.
- Once written, an OpenAPI specification and Swagger tools can drive your API development further in various ways:
  - Design-first users: use Swagger Codegen to generate a server stub for your API. The only thing left is to implement the server logic – and your API is ready to go live!
  - Use Swagger Codegen to generate client libraries for your API in over 40 languages.
  - Use Swagger UI to generate interactive API documentation that lets your users try out the API calls directly in the browser.
  - Use the spec to connect API-related tools to your API. For example, import the spec to SoapUI to create automated tests for your API.

Getting set up to make API Requests



First Open API definitions

The Design-First and API-First Approach

- In the “Design-First” approach, write out API design in a human and machine-readable form that all stakeholders can understand, review, and discuss before starting to code.
- In API design, “Contract-First” is used interchangeably with Design-First, indicating that the API Designer defines the contract of the API before the implementation is coded.
- Applying the design-first / contact-first methodology to the world of API development results in “API-First” development.

The Design-First and API-First Approach

- The basic idea is that you create an API contract before doing a full code implementation.
- The human-readable API specification becomes the first deliverable, allowing fast feedback from various stakeholders.
- Once the team determines that the API specification is “good enough”, the skeleton of a full application can be generated.
- Many different programming languages are supported.

The Design-First and API-First Approach

- The basic idea is that you create an API contract before doing a full code implementation.
- The human-readable API specification becomes the first deliverable, allowing fast feedback from various stakeholders.
- Once the team determines that the API specification is “good enough”, the skeleton of a full application can be generated.
- Many different programming languages are supported.
- For example, a full Spring Boot Application including Maven scripts and JSON-to-Java mapping can be generated.

The Design-First and API-First Approach

- The developer codes the business logic implementation in the generated server stub.
- As the server-side developer is working on their implementation, consumers of the API can generate client-side stubs and start their implementation work.
- QA teams can use the API spec to get a head start on service testing.

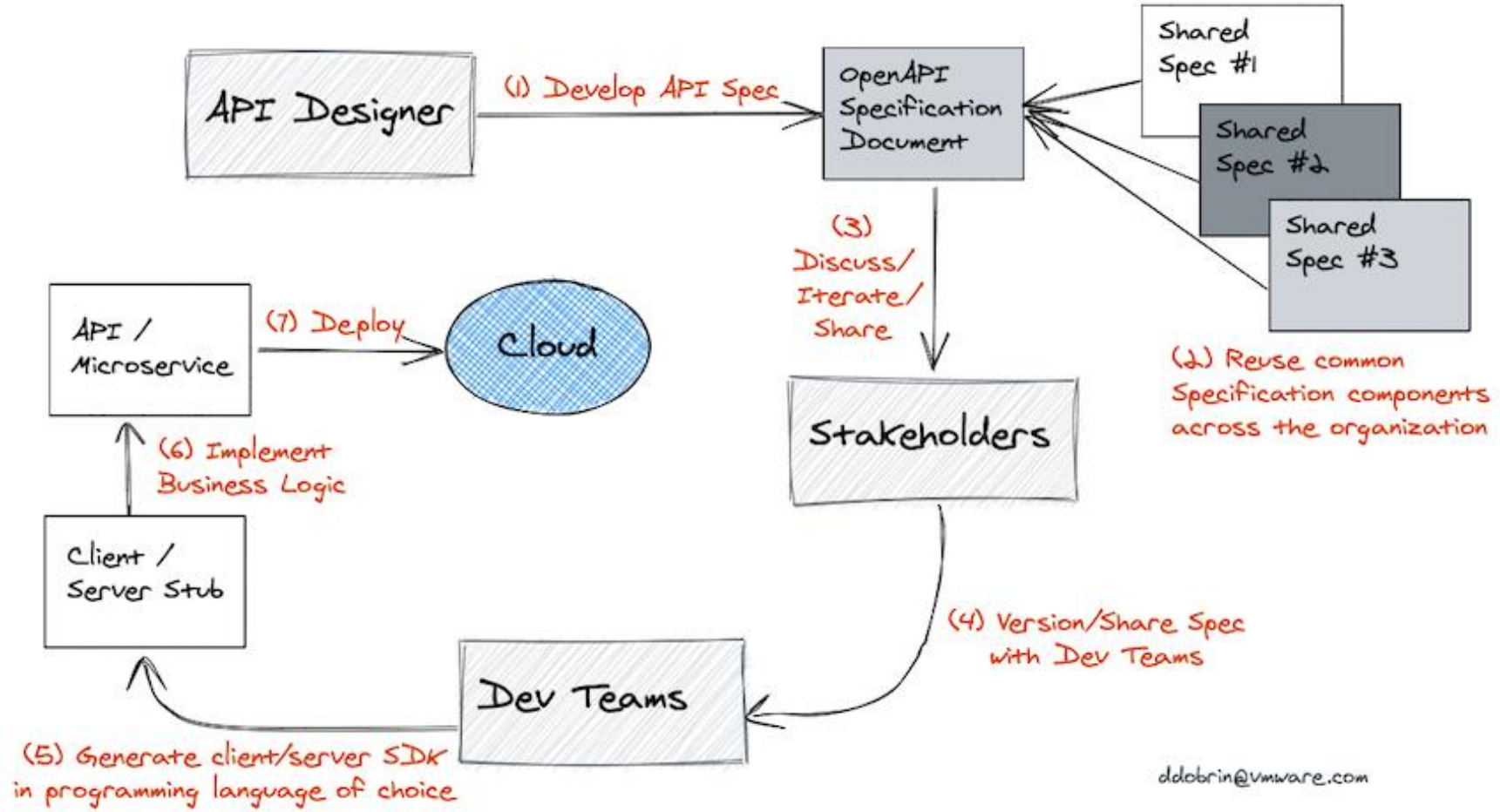
The Design-First and API-First Approach

- The API-First approach has several advantages:
  - Design-driven development - API designers use the API design to drive development efforts
  - Parallel work - Multiple stakeholders work in parallel (API designers, API consumers, technical writers, QA)
  - API Governance - API teams can use a *contract-as-code approach, versioning and publishing their API specifications
  - Assisting DevOps - DevOps team can use the API design to test the API before Production deployment
  - Agility in incorporating changes - API design becomes an ongoing and evolving process, well-supported by automated tooling

API-First Development Workflow

- There are 7 high-level steps in the API-First workflow:
  - Create the API specification
  - Build API specs at-scale within your organization
  - Collaborate with stakeholders
  - Version / Share API spec with other teams
  - Generate Client/Server scaffolding for Spring Boot
  - Implement business logic
  - Deploy the microservice

API-First Development Workflow



Step 1: Create the API Specification

- Use the OpenAPI 3.0 specification standard and the YAML format to describe the API.
- An OpenAPI document has three required sections or objects:
  - openapi - Semantic version number of the OpenAPI Specification version.
  - info - Metadata about the API.
  - paths - Available paths and operations for the API.

Step 1: Create the API Specification

- The openapi object states the version of the specification used for the document.
- The version is essential for users to understand how the document is structured, and for any tooling which may ingest the document for purposes of validation, or to create virtual services, among other reasons.
- The info object provides essential information about the API itself.
- The title and version are required fields, and we have the option to include additional information such as a description and contact and licensing information.

Step 1: Create the API Specification

OpenAPI and Info Objects

```
openapi: 3.0.1
info:
 version: "1.0.0"
 title: User Preference API
 description: This is an to support the creation, modification,
   retrieval and deletion of a User Preference.
```

Step 1: Create the API Specification

- Paths Object
  - The paths object is the heart of the API document.
  - This object details the paths available to interact with the application, what methods are available, and the details of what those interactions include.
  - This object includes request parameters and expected outcomes

Step 1: Create the API Specification

```yaml
paths:
 /preference:
   get:
     summary: Find preference by ID
     description: Returns user preference
     operationId: getPreferenceById
     parameters:
     - name: id
       in: query
       description: ID of preference to return
       required: true
       schema:
         type: string
     responses:
       '200':
         description: Successful request
         content:
           application/json:
             schema:
               $ref: '#/components/schemas/Preference'
       '400':
         description: Invalid ID supplied
       '404':
         description: User Preference not found
```

Media Types

- text/plain; charset=utf-8
- application/json
- application/vnd.github+json
- application/vnd.github.v3+json
- application/vnd.github.v3.raw+json
- application/vnd.github.v3.text+json
- application/vnd.github.v3.html+json
- application/vnd.github.v3.full+json
- application/vnd.github.v3.diff
- application/vnd.github.v3.patch

Data Types

type	format	Comments
integer	int32	signed 32 bits
integer	int64	signed 64 bits (a.k.a long)
number	float	
number	double	
string		
string	byte	base64 encoded characters
string	binary	any sequence of octets
boolean		
string	date	As defined by full-date - RFC3339
string	date-time	As defined by date-time - RFC3339
string	password	A hint to UIs to obscure input.

OpenAPI Object

Fixed Fields

Field Name	Type	Description
openapi	`string`	**REQUIRED**. This string MUST be the semantic version number of the OpenAPI Specification version that the OpenAPI document uses. The `openapi` field SHOULD be used by tooling specifications and clients to interpret the OpenAPI document. This is *not* related to the API `info.version` string.
info	Info Object	**REQUIRED**. Provides metadata about the API. The metadata MAY be used by tooling as required.
servers	[Server Object]	An array of Server Objects, which provide connectivity information to a target server. If the `servers` property is not provided, or is an empty array, the default value would be a Server Object with a url value of `/`.
paths	Paths Object	**REQUIRED**. The available paths and operations for the API.
components	Components Object	An element to hold various schemas for the specification.

OpenAPI Object

security	[Security Requirement Object]	A declaration of which security mechanisms can be used across the API. The list of values includes alternative security requirement objects that can be used. Only one of the security requirement objects need to be satisfied to authorize a request. Individual operations can override this definition. To make security optional, an empty security requirement ({}) can be included in the array.
tags	[Tag Object]	A list of tags used by the specification with additional metadata. The order of the tags can be used to reflect on their order by the parsing tools. Not all tags that are used by the Operation Object must be declared. The tags that are not declared MAY be organized randomly or based on the tools' logic. Each tag name in the list MUST be unique.
externalDocs	External Documentation Object	Additional external documentation.

Info Object

Fixed Fields

Field Name	Type	Description
title	`string`	**REQUIRED**. The title of the API.
description	`string`	A short description of the API. CommonMark syntax MAY be used for rich text representation.
termsOfService	`string`	A URL to the Terms of Service for the API. MUST be in the format of a URL.
contact	Contact Object	The contact information for the exposed API.
license	License Object	The license information for the exposed API.
version	`string`	**REQUIRED**. The version of the OpenAPI document (which is distinct from the OpenAPI Specification version or the API implementation version).

Contact Object

Contact Object

Contact information for the exposed API.

Fixed Fields

Field Name	Type	Description
name	`string`	The identifying name of the contact person/organization.
url	`string`	The URL pointing to the contact information. MUST be in the format of a URL.
email	`string`	The email address of the contact person/organization. MUST be in the format of an email address.

License Object

License information for the exposed API.

Fixed Fields

Field Name	Type	Description
name	`string`	**REQUIRED**. The license name used for the API.
url	`string`	A URL to the license used for the API. MUST be in the format of a URL.

Create an API

- We will be designing an API for a record label.
- Let's assume that the record label has a database of artists with the following information:
  - Artist name
  - Artist genre
  - Number of albums published under the label
  - Artist username
- The API will let consumers obtain the list of artists stored in the database and add a new artist to the database.

Create an API

- An API defined using the OpenAPI Specification can be divided into 3 main sections –
- Meta information
- Path items (endpoints):
  - Parameters
  - Request bodies
  - Responses
- Reusable components:
  - Schemas (data models)
  - Parameters
  - Responses
  - Other components

Meta information

```yaml
openapi: 3.0.0
info:
  version: 1.0.0
  title: Simple Artist API
  description: A simple API to illustrate OpenAPI concepts

servers:
  - url: https://example.io/v1

# Basic authentication
components:
  securitySchemes:
    BasicAuth:
      type: http
      scheme: basic
security:
  - BasicAuth: []

paths: {}
```

Meta information

- Each API definition starts with the version of the OpenAPI Specification that this definition uses.
- In our example, it is openapi: 3.0.0.
- The info object contains the API title and version, which are required, and an optional description.
- The servers array specifies one or more server URLs for API calls.
- The API endpoint paths are appended to the server URL.
- Some APIs have a single server, others may have multiple servers, such as production and sandbox.
- In our example, the server URL is https://example.io/v1.

Path items

- The path items are the endpoints of your API under which you can specify HTTP verbs for manipulating the resources in the desired manner.
- These endpoints are relative to the server URL, which in our example is https://example.io/v1.
- We will define the /artists endpoint and the GET method for this endpoint.
- So, a client will use GET https://example.io/v1/artists to get a list of artists.

Path items

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: Simple API
  description: A simple API to illustrate OpenAPI concepts

servers:
  - url: https://example.io/v1

components:
  securitySchemes:
    BasicAuth:
      type: http
      scheme: basic
security:
  - BasicAuth: []

#  ----- Added lines  ----------------------------------------
paths:
  /artists:
    get:
     description: Returns a list of artists
#  ---- /Added lines  ----------------------------------------
```

Responses

- The GET method, under the artists endpoint, lets the consumer of the API obtain the details of a list of artists from the https://example.io/v1 database.
- Every response would need at least one HTTP status code to describe the kind of responses a consumer is likely to expect.
- The description gives details on what the responses of the API would be.
- In our sample code, we have specified 200, which is a successful client request, while 400 is an unsuccessful request.
- You can find more information about HTTP status codes here.
- A successful response will return the artist name, genre, username and albums recorded.
- An unsuccessful request is described under the 400 HTTP code, with a corresponding error message detailing why the response is invalid.

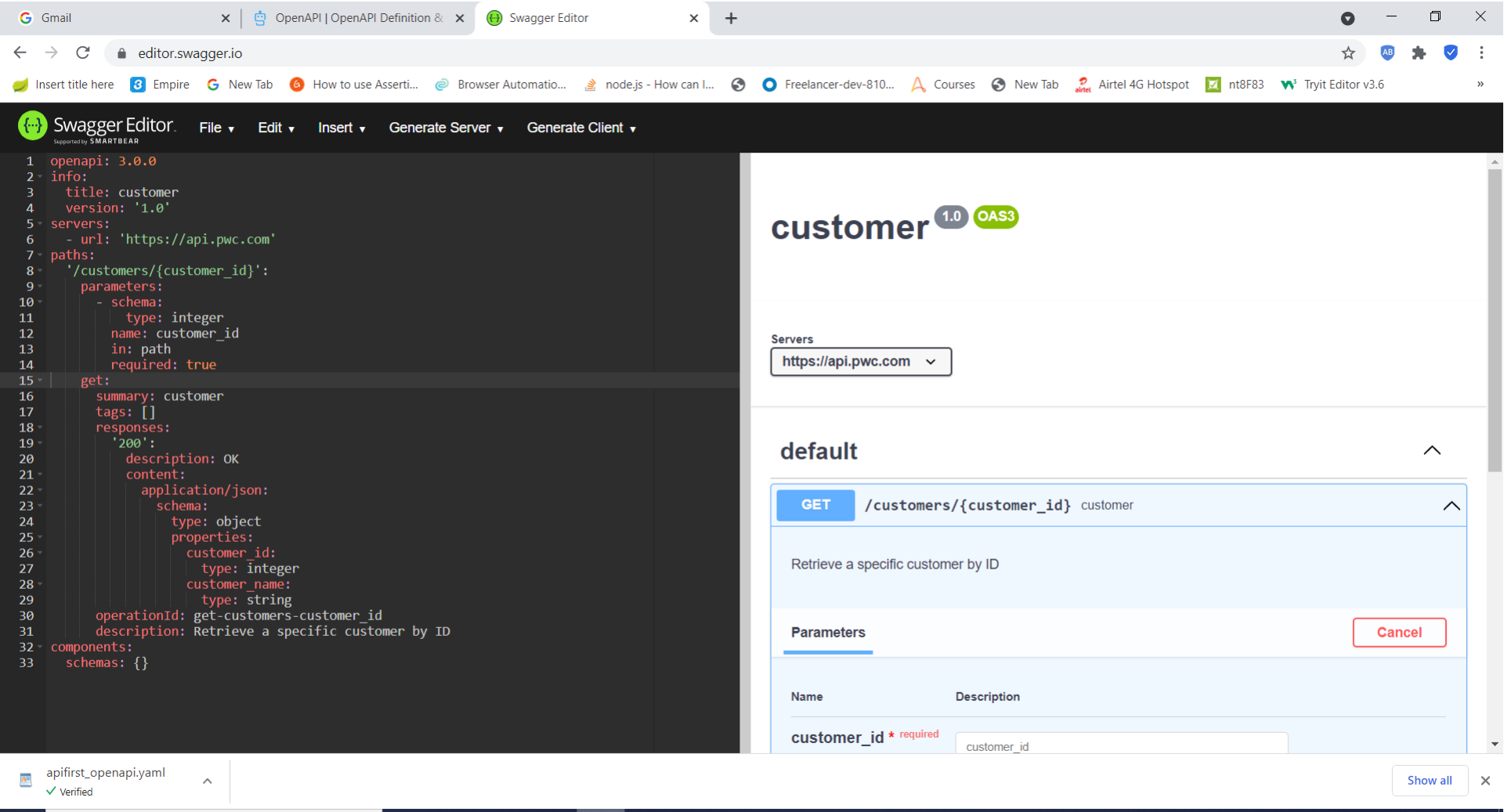
Responses

```yaml
openapi: 3.0.0
info:
  version: 1.0.0
  title: Simple API
  description: A simple API to illustrate OpenAPI concepts

servers:
  - url: https://example.io/v1

components:
  securitySchemes:
    BasicAuth:
      type: http
      scheme: basic
security:
  - BasicAuth: []

paths:
  /artists:
    get:
      description: Returns a list of artists
      #  ----- Added lines  ----------------------------------------
      responses:
        '200':
          description: Successfully returned a list of artists
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  required:
                    - username
```

Responses

```
                username
              properties:
                artist_name:
                  type: string
                artist_genre:
                  type: string
                albums_recorded:
                  type: integer
                username:
                  type: string

    '400':
      description: Invalid request
      content:
        application/json:
          schema:
            type: object
            properties:
              message:
                type: string
  #  ---- /Added lines  ----------------------------------------
```

Open API Editor

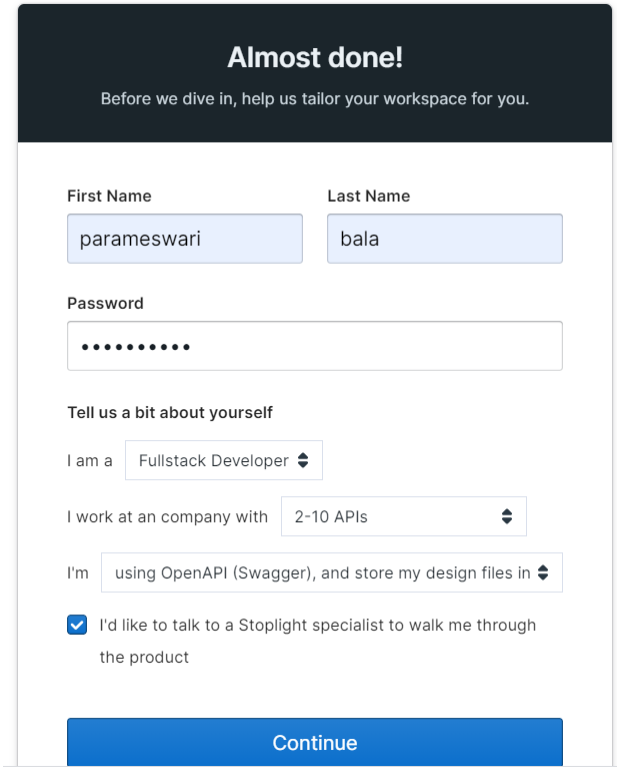


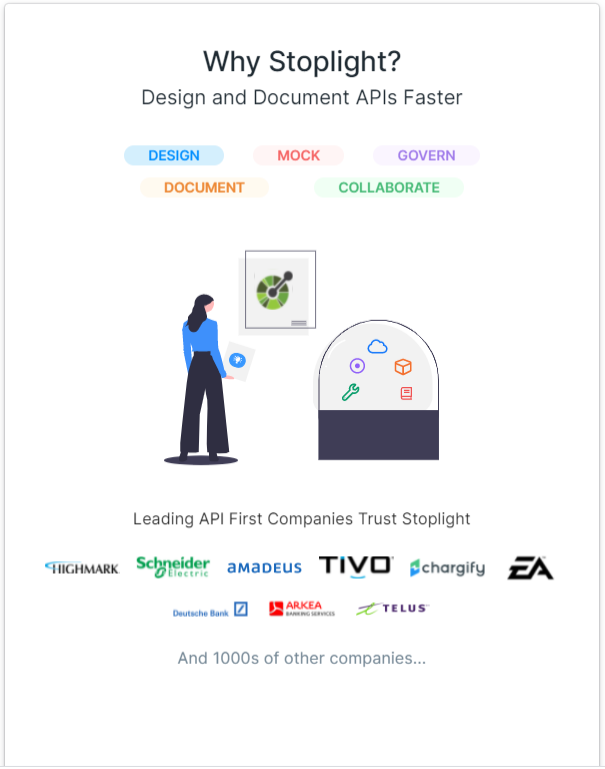
What OpenAPI Tools Are Available?

- Stoplight Studio - a free OpenAPI editor, to easily write API descriptions without memorizing syntax.
- Spectral an automated validation tool, helps maintain API consistency and style with customizable OpenAPI linting.
- Prism a mock server generator, which builds upon your OpenAPI definition to prototype or run integration tests.

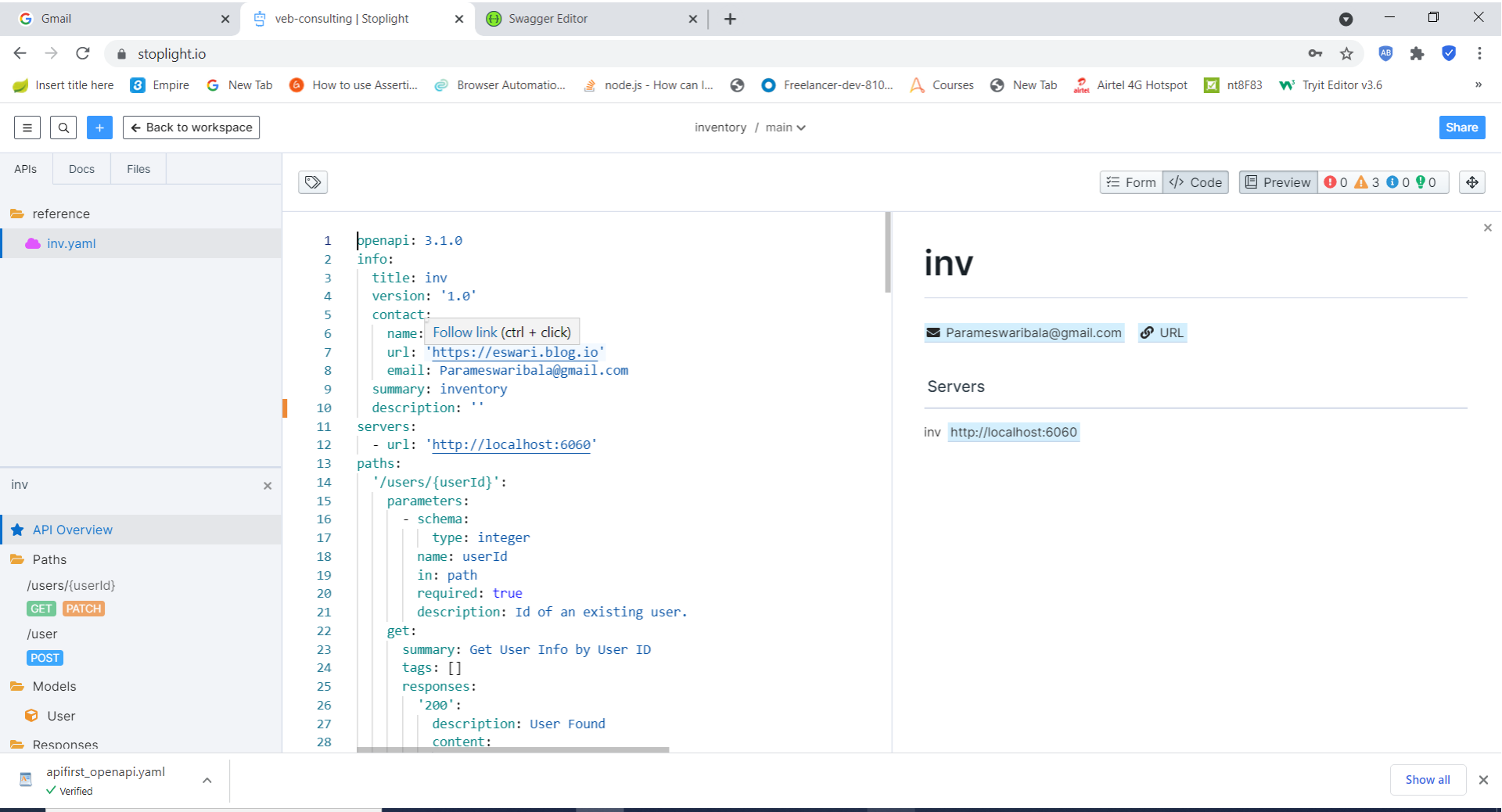
Stoplight

- https://veb-consulting.stoplight.io/welcome/account

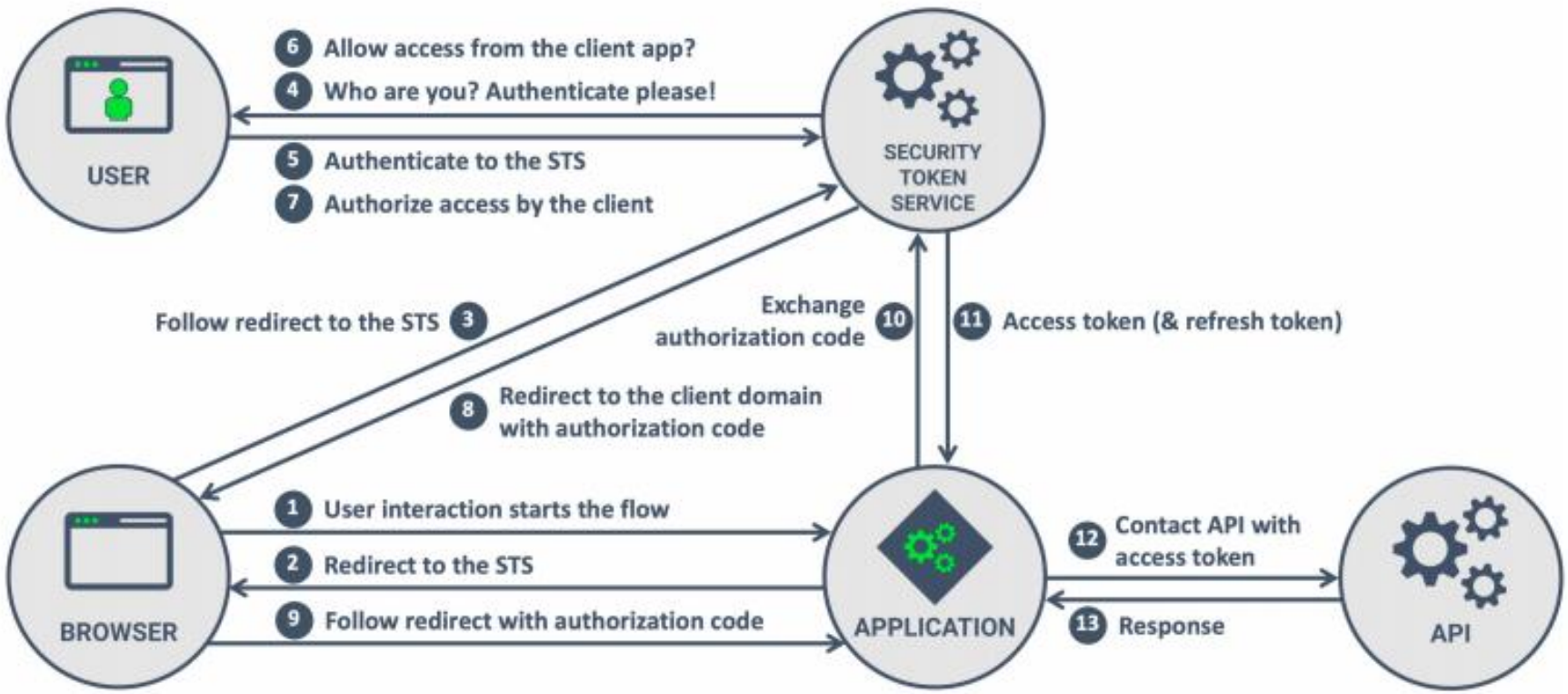




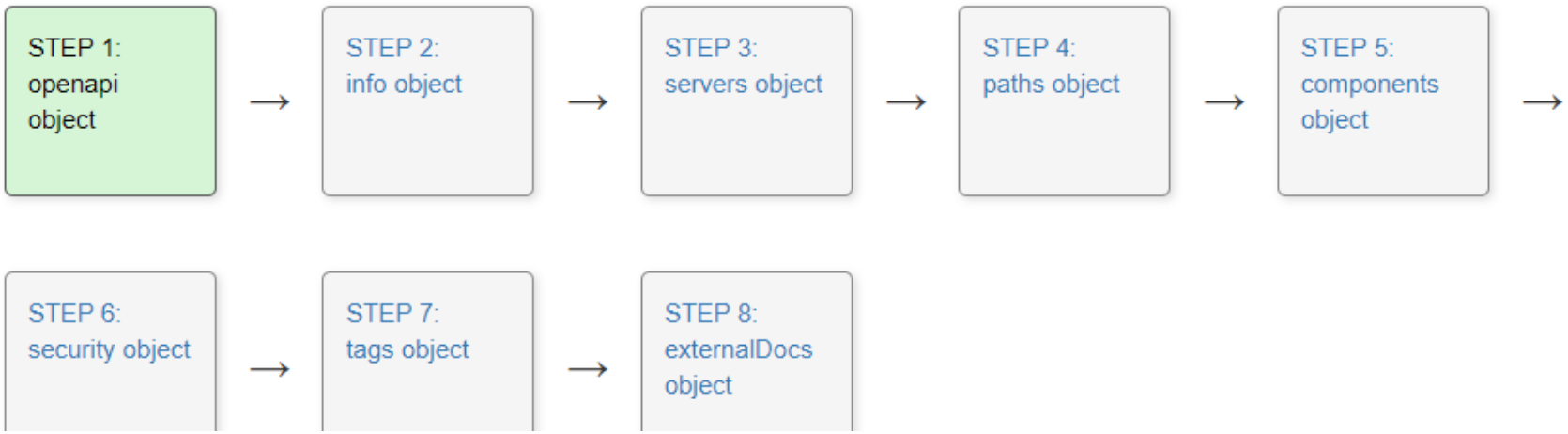
Stoplight


Gmail
veb-consulting | Stoplight
Swagger Editor
stoplight.io
Back to workspace
inventory / main
Share
APIs
Docs
Files
reference
inv.yaml
Form
Code
Preview
openapi: 3.1.0
info:
title: inv
version: '1.0'
contact:
name:
Follow link (ctrl + click)
url: 'https://eswari.blog.io'
email: Parameswaribala@gmail.com
summary: inventory
description: ''
servers:
- url: 'http://localhost:6060'
paths:
'/users/{userId}':
parameters:
- schema:
type: integer
name: userId
in: path
required: true
description: Id of an existing user.
get:
summary: Get User Info by User ID
tags: []
responses:
'200':
description: User Found
content:
inv
API Overview
Paths
/users/{userId}
GET
PATCH
/user
POST
Models
User
Responses
inv
Parameswaribala@gmail.com
URL
Servers
inv http://localhost:6060
apifirst_openapi.yaml
Verified
Show all
Type here to search
30°C
ENG
23:07
24/06/2021

Oauth Authentication



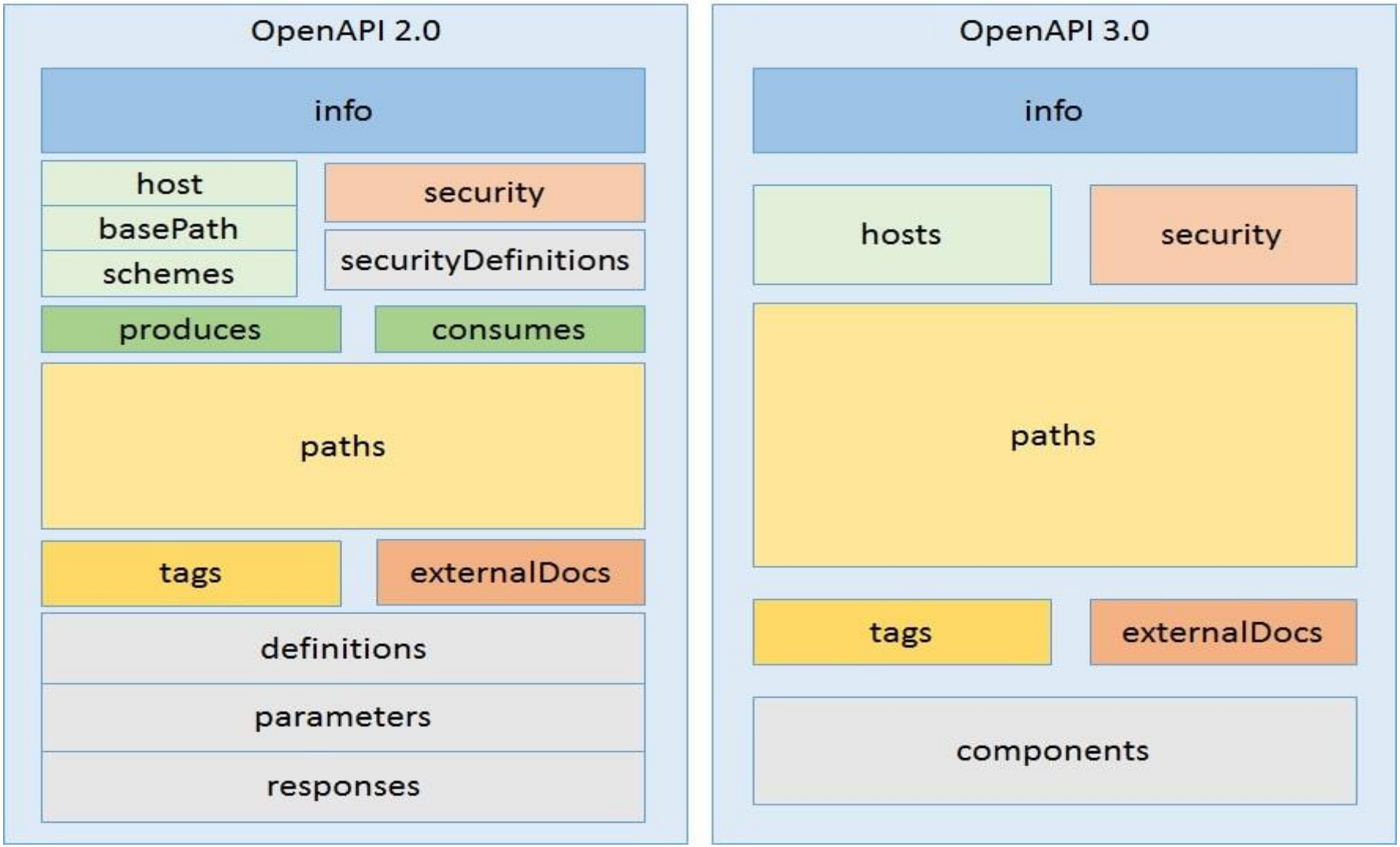
API-First Development Workflow



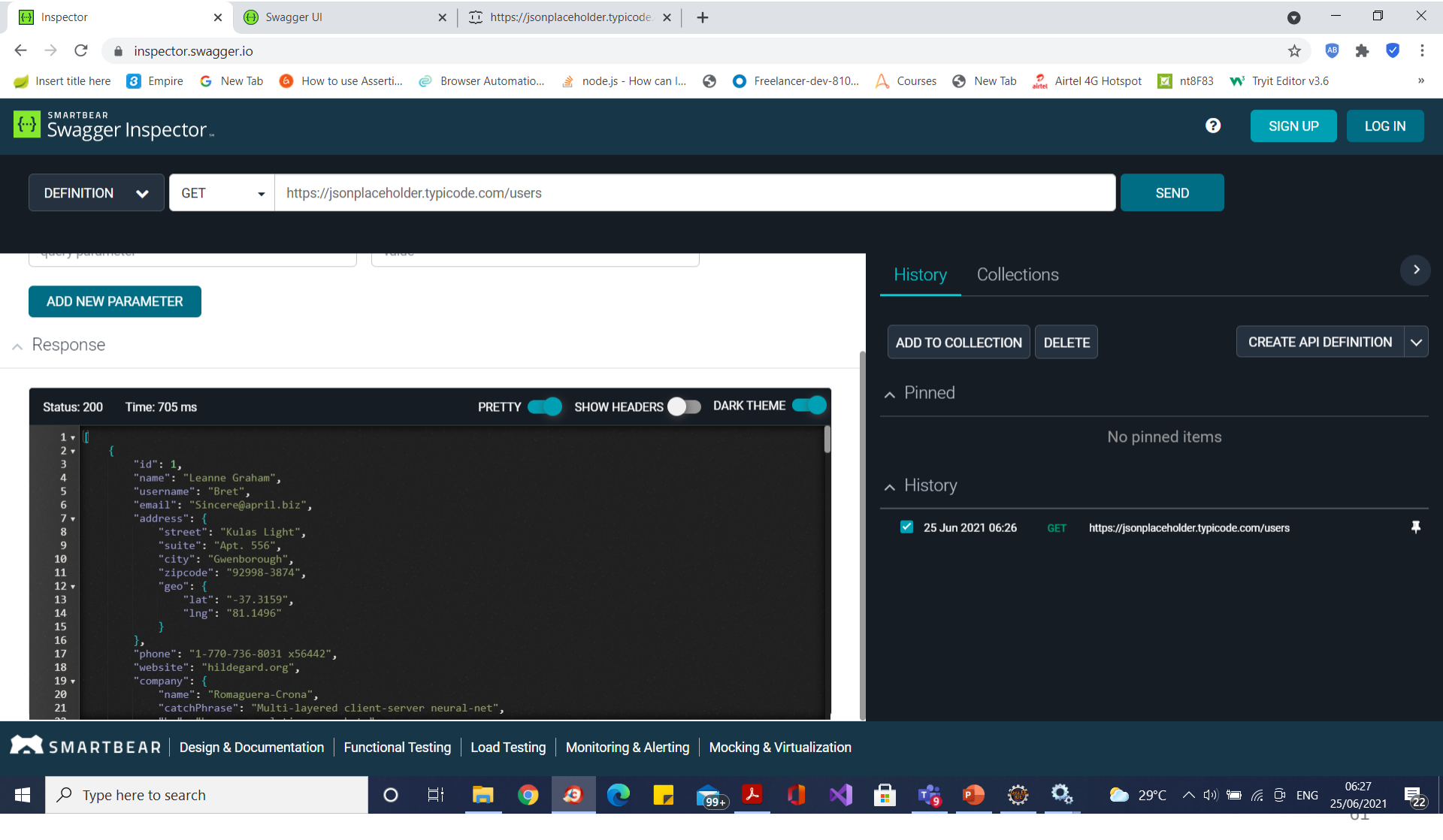
Which security scheme?

- API key
- HTTP
- OAuth 2.0
- Open ID Connect

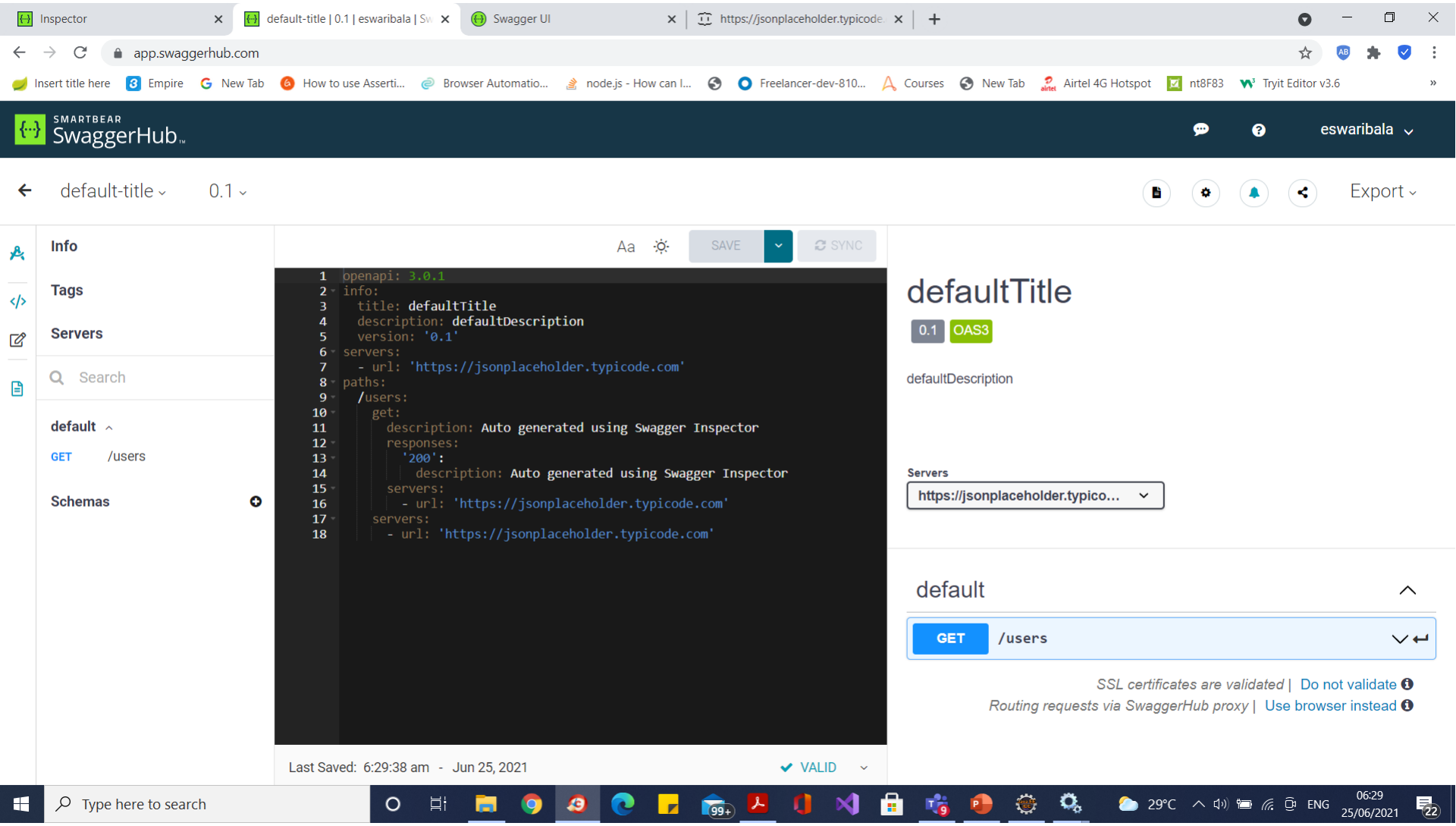
Open API 2.0 vs 3.0



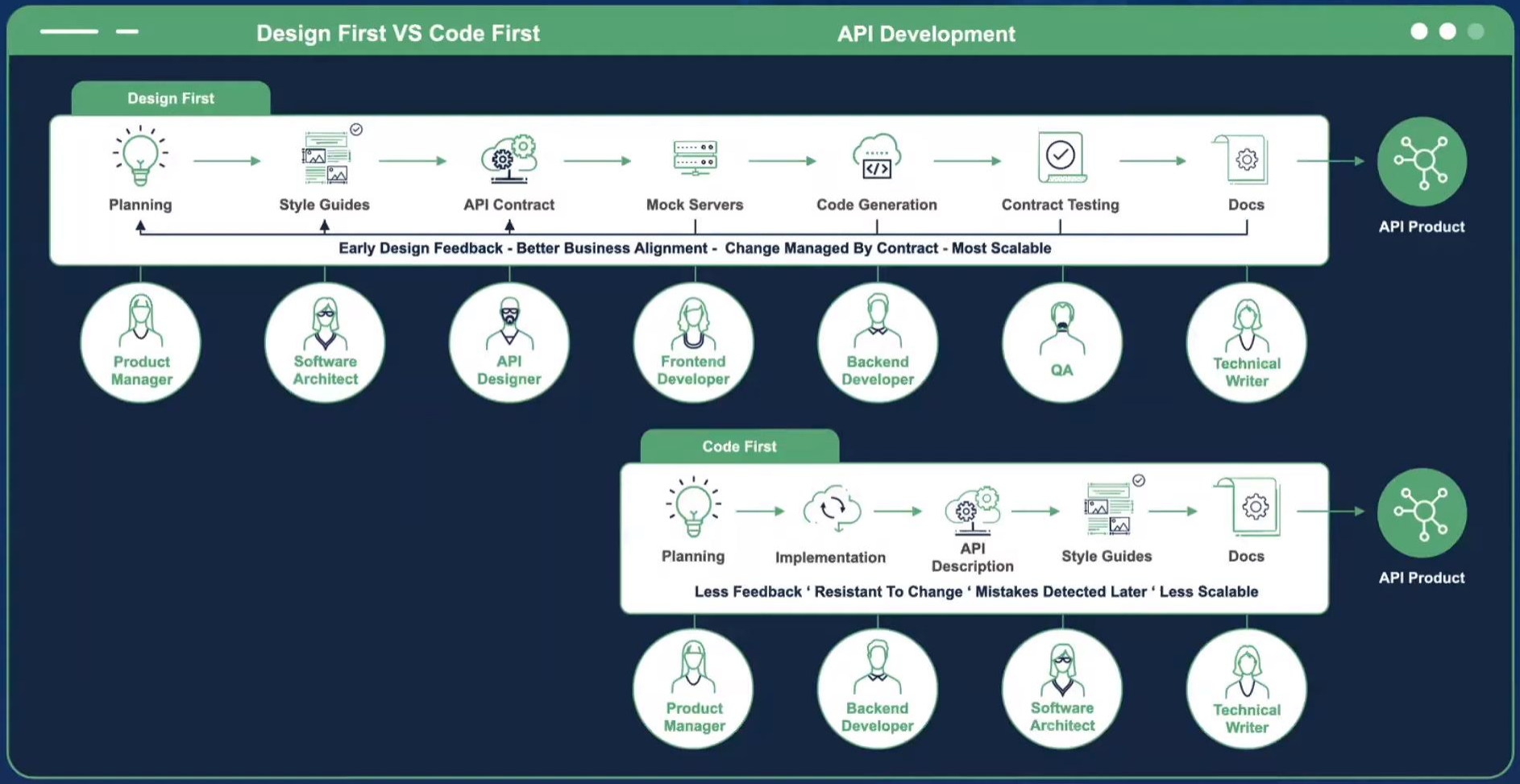
Swagger Inspector



Swagger Inspector



API Workflow



Questions

Module Summary

- In this module we discussed
  - Overview of Maven
  - Maven archetypes
  - Maven life cycle phases
  - The pom.xml file
  - Creation of Java projects using Maven
  - Creation of war files