# Open API

## Qu'est-ce qu'OpenAPI ?

- Open API définit les **spécifications** pour décrire une API HTTP
- Cette description se présente sous la forme d'un fichier Yaml ou Json
- Utile pour générer de la documentation, du code, des tests, ...
- `https://www.openapis.org/`


Figure – Exemple d'un fichier OpenAPI et la documentation générée

## Historique

- 2010 : Projet de spécification inclus dans le logiciel Swagger à l'origine
- 2015 : Dévient un projet autonome, open-source, et est confié à la Linux Foundation.
- 2016 : Se renomme Open API Spécification
- 2024 : Dernière version 3.1.1

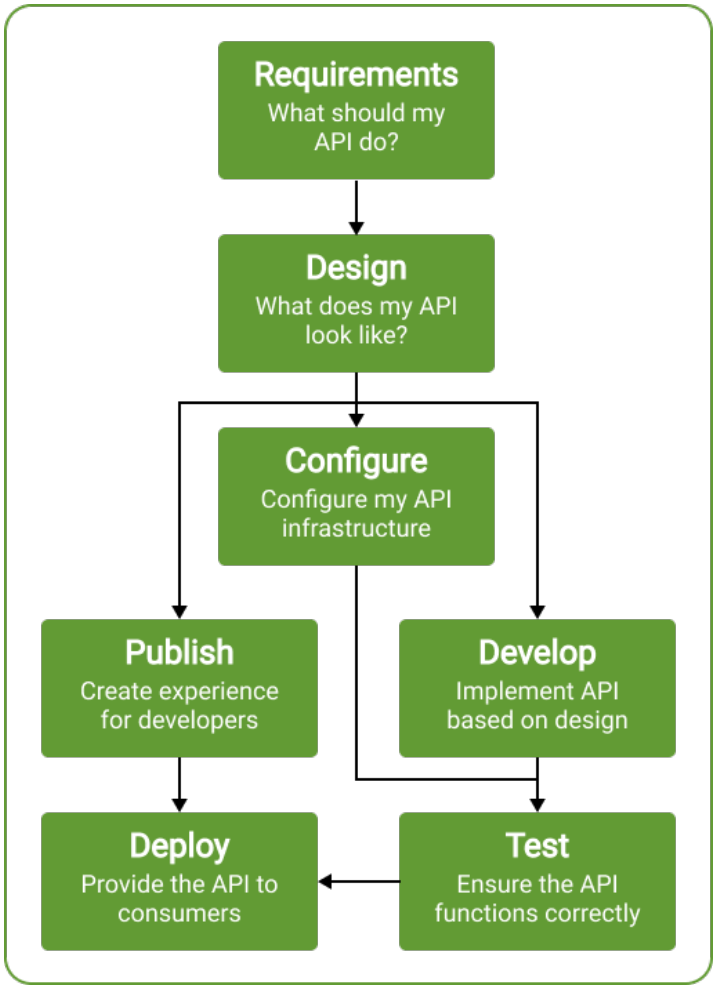


Figure – OpenApi utile tout au long du cycle de vie d'une API

# OpenAPI dans le cycle de vie du développement

- **Expression des besoins** : facilite la rédaction et la communication de ses idées pour l'application, dans un document portable
- **Conception** :
  - Point d'entrée pour le développement
  - *Code First* : le code est écrit d'abord, la documentation est générée ensuite
  - *Design First* : La spécification est écrite en premier, puis le code. On possède ainsi déjà le squelette du code (qui peut être généré automatiquement)
- **Déploiement** : les routes sont bien spécifiées. Base pour la sécurité
- **Tests** : Facilite l'écriture de tests.
- **Utilisation** : les clients de votre API n'ont pas besoin de connaître votre langage de programmation

# Intérêts

## Format standardisé

- Lisible à la fois par un être humain ou par un programme
  - Permet à un développeur d'utiliser plus facilement une API
  - Permet de réaliser des actions automatiques
- Non lié à un langage de programmation
- Description
  - ressource
  - paramètre
  - authentification
  - retour
  - ...

## Base pour des outils de :

- Validation, de vérification, de sécurité
- Génération de documentation
- Génération de code : construction automatique de client

## Exemples d'outils

- **Swagger** : outils de génération de code `https://editor-next.swagger.io/`
- Plugin VSCode pour la génération de documentation et la validation `https://marketplace.visualstudio.com/items?itemName=42Crunch.vscode-openapi`
- Liste d'outils `https://tools.openapis.org/`

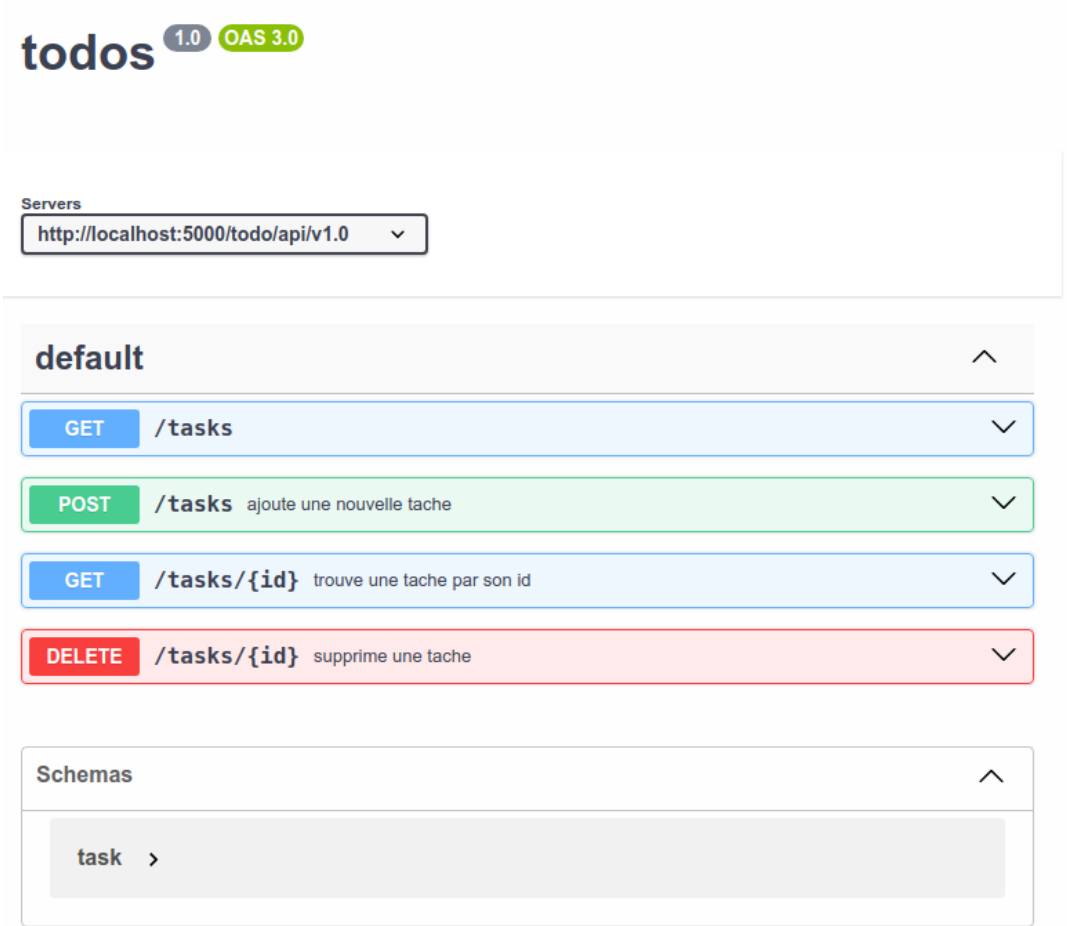


Figure – Génération d'une page html à partir des spécifications

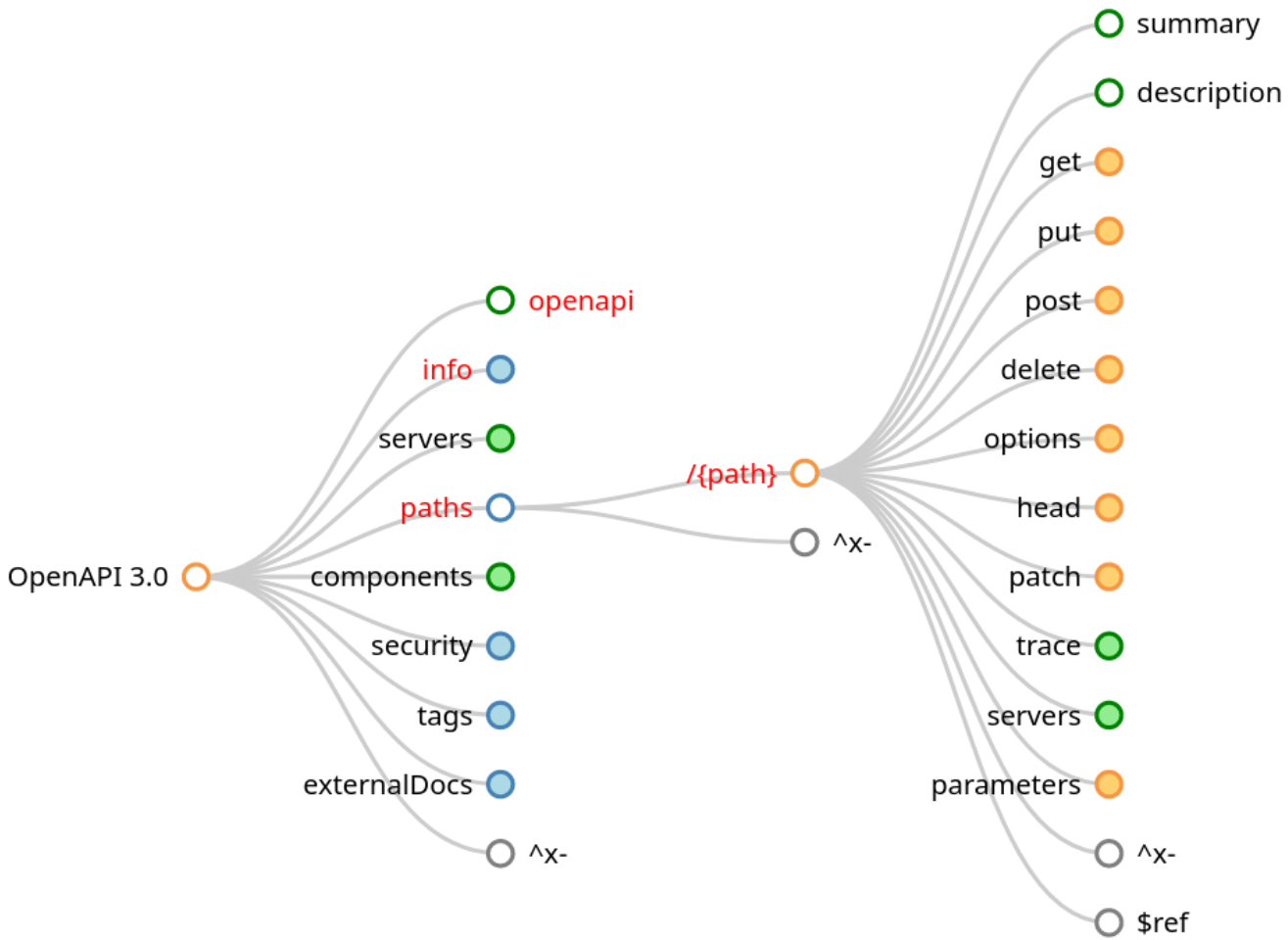


Figure – Spécification d'OpenAPI

https://openapi-map.apihandyman.io/

La spécification définit plusieurs étiquettes qui vont permettre entre autres de :

- définir les informations générales sur vos API : description, termes d'utilisation, licence, contact, etc.
- fournir la liste des services qui seront offerts, avec pour chacun, comment les appeler et la structure de la réponse qui est retournée
- définir le chemin pour consommer votre API
- ...

`https://spec.openapis.org/oas/latest.html`

```yaml
openapi: '3.0.2'
info:
  title: todos
  version: '1.0'
servers:
  - url: http://localhost:5000/todo/api/v1.0
```

L'en-tête décrit la version de l'API, et l'url de base

```yaml
paths:
  /tasks:
    get:
      responses:
        '200':
          description: retourne la liste des taches
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  required :
                    - title
                  properties:
                    title:
                      type: string
                    description:
                      type: string
                    done:
                      type: boolean
                    uri:
```

```
/tasks/{id}:
  get:
    summary: trouve une tache par son id
    description : retourne une tache
    parameters:
      - name : id
        in: path
        description: id de la tache
        required: true
        schema:
          type: integer
          format: int64
    responses:
      '200':
        description : tache trouve
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/task"
      '400':
        description : Not found
```

- Nous avons une entrée pour chaque point d'entrée de l'API
- On précise le type de requête (get, put post, delete)
- On détaille les paramètres en entrée :
  - leur nom
  - leur rôle
  - leur type
  - s'ils sont obligatoires
- La réponse :
  - Le code retour http
  - le type de la réponse

```yaml
components:
  schemas:
    task:
      type: object
      properties:
        title:
          type: string
        description:
          type: string
        done:
          type: boolean
```

- Il est possible de définir des schémas des différents objets que l'on manipule.
- On peut ensuite y faire référence

```
schema:
        $ref: "#/components/schemas/task"
```

- Permet de regrouper les définitions, facilite l'écriture des spécifications

# Conclusion

- OpenAPI fournit les spécifications pour documenter vos API HTTP
- Utile dans plusieurs aspects (conception, documentation,tests ...)
- Écosystème d'outils autour de ses spécifications