

# Structure YAML et schémas

## YAML en 30 secondes

YAML est un format de données lisible, basé sur l'**indentation** (des espaces, jamais de tabulations).

```
nom: Segfault Sunrise      # une clé et une valeur
prix: 9.5                  # un nombre
sans_alcool: false        # un booléen
tags:                      # une liste
  - classic
  - fruity
recette:                   # un objet imbriqué
  base: rhum
  dose_cl: 4
code: '404'                # des guillemets pour forcer une chaîne
```

Une indentation décalée d'un espace change complètement le sens du document. Si Swagger Editor affiche une erreur incompréhensible, vérifiez d'abord l'indentation.

## La structure d'un contrat

```
openapi: 3.1.0             # version d'OpenAPI

info:                      # présentation de l'API
  title: API-ritif
  version: 1.0.0
  description: Le bar de l'école de dev.

servers:                   # où se trouve l'API
  - url: http://api-ritif.portlatch.com/v1
```

```

paths:                                # les routes
  /cocktails: ...
  /cocktails/{cocktail_id}: ...

components:                           # les éléments réutilisables
  schemas: ...                         # formats de données
  parameters: ...                     # paramètres communs
  responses: ...                       # réponses communes
  securitySchemes: ...                # méthodes d'authentification

```

## Une opération

Sous chaque chemin, une clé par méthode HTTP. Chaque opération décrit ses paramètres, son body et ses réponses.

```

paths:
  /cocktails/{cocktail_id}/reviews:
    post:
      summary: Donner son avis sur un cocktail
      operationId: createReview
      parameters:
        - name: cocktail_id
          in: path                # path, query, header ou cookie
          required: true
          schema:
            type: integer
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ReviewCreate'
      responses:
        '201':
          description: Avis créé
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Review'

```

'422':

description: Données invalides

L'`operationId` donne un nom unique à l'opération. Les générateurs de code s'en servent pour nommer les fonctions : choisissez-le bien.

# Les schémas

Les schémas décrivent la forme des données, avec la syntaxe **JSON Schema**.

```
components:
  schemas:
    ReviewCreate:
      type: object
      required: [rating]           # champs obligatoires
      additionalProperties: false  # aucun autre champ accepté
      properties:
        rating:
          type: integer
          minimum: 1
          maximum: 5
          example: 4
        comment:
          type: string
          maxLength: 500
          example: Plante rarement, mais quel goût.

    OrderStatus:
      type: string
      enum: [pending, preparing, ready, served, cancelled]
```

| Mot-clé                                                                                                              | Rôle                                                                                                                                                   |
|----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>type</code>                                                                                                    | <code>string</code> , <code>integer</code> , <code>number</code> , <code>boolean</code> , <code>array</code> , <code>object</code> , <code>null</code> |
| <code>required</code>                                                                                                | Liste des champs obligatoires                                                                                                                          |
| <code>enum</code>                                                                                                    | Liste des valeurs autorisées                                                                                                                           |
| <code>minimum</code> , <code>maximum</code> , <code>minLength</code> , <code>maxLength</code> , <code>pattern</code> | Contraintes de validation                                                                                                                              |

| Mot-clé               | Rôle                                                                           |
|-----------------------|--------------------------------------------------------------------------------|
| <code>format</code>   | Précision : <code>date-time</code> , <code>email</code> , <code>uri</code> ... |
| <code>example</code>  | Un exemple, affiché dans la documentation                                      |
| <code>readOnly</code> | Champ renvoyé par l'API, mais jamais accepté en entrée                         |

## Ne pas se répéter : `$ref`

`$ref` fait référence à un élément défini ailleurs. Le schéma `Problem`, les paramètres de pagination ou la réponse `429` ne sont écrits qu'une fois dans `components`, puis réutilisés partout.

```
responses:
  '429':
    $ref: '#/components/responses/TooManyRequests'
```

## L'authentification

```
components:
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      in: header
      name: X-API-Key

paths:
  /me:
    get:
      security:
        - ApiKeyAuth: [] # cette route demande une clé

  /cocktails:
    get:
      security: [] # cette route est publique
```

Avec `securitySchemes`, le bouton **Authorize** de Swagger UI permet de saisir sa clé une fois pour toutes les requêtes de test.

Revision #3

Created 2 October 2026 20:00:50 by Thibaud FRICHET

Updated 12 October 2026 22:02:46 by Thibaud FRICHET