

Swagger UI, Redoc et Scalar

Un contrat en YAML, c'est précis mais austère. Des outils le transforment en une documentation agréable à lire, sans rien écrire de plus.

Trois outils, un même contrat

Outil	Points forts
Swagger UI	Le classique. Chaque route est testable directement (Try it out). Intégré à FastAPI sur <code>/docs</code> .
Redoc	Une documentation en trois colonnes, très lisible, idéale pour une API publique. Lecture seule. Intégré à FastAPI sur <code>/redoc</code> .
Scalar	Le plus récent : moderne, avec un client de test intégré et des exemples de code générés dans de nombreux langages.

Publier la documentation

Chaque outil tient en une page HTML qui charge le contrat. Placez la page à côté du fichier `api-ritif.openapi.yaml`.

Swagger UI

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="https://unpkg.com/swagger-ui-dist@5/swagger-ui.css">
</head>
<body>
  <div id="swagger-ui"></div>
  <script src="https://unpkg.com/swagger-ui-dist@5/swagger-ui-bundle.js"></script>
  <script>
    SwaggerUIBundle({ url: "api-ritif.openapi.yaml", dom_id: "#swagger-ui" });
  </script>
```

```
</body>
</html>
```

Redoc

```
<!doctype html>
<html>
<head><meta charset="utf-8"></head>
<body>
  <redoc spec-url="api-ritif.openapi.yaml"></redoc>
  <script src="https://cdn.redoc.ly/redoc/latest/bundles/redoc.standalone.js"></script>
</body>
</html>
```

Scalar

```
<!doctype html>
<html>
<head><meta charset="utf-8"></head>
<body>
  <div id="app"></div>
  <script src="https://cdn.jsdelivr.net/npm/@scalar/api-reference"></script>
  <script>
    Scalar.createApiReference("#app", { url: "api-ritif.openapi.yaml" });
  </script>
</body>
</html>
```

Ouverte en double-cliquant (`file://`), la page ne pourra pas charger le contrat.
Servez le dossier avec un petit serveur web : `python3 -m http.server` , puis ouvrez
<http://localhost:8000>.

Ce qui fait une bonne documentation

Les outils mettent en forme. Le contenu, c'est vous qui l'écrivez dans le contrat :

- une `description` générale : à quoi sert l'API, comment s'authentifier, les limites de requêtes ;
- un `summary` court et clair pour chaque opération ;

- une `description` pour les règles métier (« annulable uniquement tant que la commande est `pending` ») ;
- des `example` réalistes pour chaque schéma ;
- **toutes** les réponses d'erreur possibles, pas seulement le cas qui marche ;
- des `tags` pour regrouper les routes par thème.

Le test ultime : un développeur qui n'a jamais vu votre API arrive-t-il à l'utiliser avec la doc seule, sans vous poser de question ?

Revision #3

Created 2 October 2026 20:00:51 by Thibaud FRICHET

Updated 13 October 2026 22:31:35 by Thibaud FRICHET