

TP : Les avis

Objectif

Le patron d'API-ritif veut recueillir l'avis de ses clients. Le contrat est déjà écrit (design-first !) : à vous de coder la ressource `/v1/cocktails/{cocktail_id}/reviews`, en **FastAPI**.

Votre API sera corrigée automatiquement par une **collection Bruno** fournie par le formateur : chaque test passe ou échoue.

Ce TP est individuel et noté. On part d'un projet vide : reprenez l'installation de la page « Premiers pas avec FastAPI ».

Consignes

1 - Démarrer le projet

- Créez un projet FastAPI vide (port `8000`).
- Ajoutez la fonction `problem()` et les gestionnaires d'erreurs de la page « Validation et codes de retour ».
- Copiez ces données en mémoire :

```
# Les cocktails (seul l'id compte pour ce TP)
cocktails = [
    {"id": 1, "name": "Hello World"},
    {"id": 2, "name": "Null Pointer Mojito"},
    {"id": 3, "name": "Segfault Sunrise"},
    {"id": 4, "name": "Pull Request Colada"},
    {"id": 5, "name": "Kernel Panic"},
]

# Les clés d'API valides et le pseudo associé
api_keys = {
    "cle-neo": "neo",
```

```
"cle-trinity": "trinity",  
"cle-morpheus": "morpheus",  
}
```

2 - Le contrat à respecter

Il se trouve dans `api-ritif.openapi.yaml`, au tag **Avis**. En résumé :

Route	Auth	Succès	Erreurs
GET <code>/v1/cocktails/{cocktail_id}/reviews</code>	Non	200	404, 422
POST <code>/v1/cocktails/{cocktail_id}/reviews</code>	X-API-Key	201 + Location	401, 404, 409, 422
GET <code>/v1/cocktails/{cocktail_id}/reviews/{review_id}</code>	Non	200	404

Le body attendu pour créer un avis :

```
ReviewCreate:  
  type: object  
  required: [rating]  
  additionalProperties: false  
  properties:  
    rating:  
      type: integer  
      minimum: 1  
      maximum: 5  
    comment:  
      type: string  
      maxLength: 500
```

Un avis renvoyé par l'API :

```
{  
  "id": 7,  
  "cocktail_id": 3,  
  "author": "neo",  
  "rating": 4,  
  "comment": "Plante rarement, mais quel goût.",  
}
```

```
"created_at": "2026-10-14T10:32:00Z"
}
```

3 - Les règles

- **Toutes** les erreurs sont au format `application/problem+json`.
- `401` si la clé est absente ou inconnue. L'`author` de l'avis est le pseudo associé à la clé : le client ne l'envoie jamais.
- `404` si le cocktail n'existe pas (y compris pour un `GET`).
- `422` si le body est invalide : note absente, hors de 1 à 5, pas un entier, commentaire trop long, champ inconnu... Le problème doit lister les champs en faute dans `errors`.
- `409` si ce client a déjà donné son avis sur ce cocktail.
- `201` en cas de succès, avec le header `Location: /v1/cocktails/{cocktail_id}/reviews/{id}` et l'avis créé dans le body.

4 - La pagination

La liste est paginée, triée du plus récent au plus ancien, et suit le format de la page « Filtres, tri et pagination » :

- paramètres `page` (à partir de 1, 1 par défaut) et `per_page` (de 1 à 50, 5 par défaut), avec un `422` hors de ces limites ;
- réponse `{"items": [...], "page", "per_page", "total", "_links"}` ;
- `_links` contient toujours `self`, `first` et `last`, et `prev` et `next` uniquement quand ils existent.

Une liste vide n'est pas une erreur : un cocktail sans avis renvoie `200` avec `"items": []`. Dans ce cas, `last` pointe vers la page 1.

5 - Tester

- Testez au fur et à mesure avec curl ou Bruno.
- Quand vous pensez avoir terminé, ouvrez la collection Bruno du formateur, réglez la variable `baseUrl` sur votre serveur (`http://localhost:8000/v1`) et lancez **toute la collection** (*Run*).

Les tests de la collection s'enchaînent et créent des avis. Redémarrez votre serveur avant chaque exécution complète pour repartir de données vides.

Bonus

- `DELETE /v1/cocktails/{cocktail_id}/reviews/{id}` : `204` pour l'auteur de l'avis, `403` pour les autres.
- Un paramètre `?sort=` sur la liste (`rating`, `-rating`, `created_at`, `-created_at`).
- Le header `Link` en plus de `_links`.
- Comparez votre `/openapi.json` au contrat fourni : quelles différences ?

Résultat attendu

- Le code de votre API (dépôt git ou archive, sans le dossier `.venv`).
- Une capture d'écran du résultat de la collection Bruno du formateur.

Revision #3

Created 2 October 2026 20:00:54 by Thibaud FRICHET

Updated 13 October 2026 22:31:37 by Thibaud FRICHET