

Nommer les ressources et les routes

Une URL est la première chose qu'un développeur voit de votre API. Si elle est claire, il devine la suite. Si elle ne l'est pas, il ouvre la documentation... ou abandonne.

Les règles

Des noms, pas des verbes

L'URL désigne une **ressource**. L'action, c'est la méthode HTTP qui la porte.

□ GET /getCocktails	□ GET /cocktails
□ POST /createOrder	□ POST /orders
□ POST /deleteOrder?id=42	□ DELETE /orders/42

Au pluriel

Une collection contient plusieurs éléments : `/cocktails` et `/cocktails/3`, plutôt que `/cocktail/3`. Surtout : **ne mélangez pas**.

En minuscules, avec des tirets

Les URL sont sensibles à la casse : `/Cocktails` et `/cocktails` sont deux adresses différentes. On écrit tout en minuscules, et on sépare les mots par des tirets (*kebab-case*).

□ /happyHours	□ /happy_hours	□ /happy-hours
---------------	----------------	----------------

Pour les champs JSON, la convention est différente : `snake_case` ou `camelCase`, au choix, mais le même partout.

Une hiérarchie courte

Les sous-ressources expriment une appartenance : `/cocktails/3/reviews` se lit « les avis du cocktail 3 ». Au-delà de deux niveaux, ça devient illisible.

□ /bars/1/customers/12/orders/42/items/3

□ /orders/42/items/3

Pas de détails techniques

Pas d'extension (`.php`, `.json`), pas de nom de technologie, pas de nom de table. Si vous réécrivez le back en Go, les URL ne doivent pas changer.

□ /api/v1/getCocktails.php

□ /tbl_cocktails

□ /v1/cocktails

Une seule langue

L'anglais est recommandé : c'est la langue de la plupart des API et des outils. Une API en français (comme la bibliothèque des chapitres 1 et 2) reste acceptable, à condition de ne jamais mélanger les deux : `/livres/3/reviews`, non merci.

Et les actions qui ne sont pas du CRUD ?

Annuler une commande, payer, rembourser... Ces actions ne sont pas « créer, lire, modifier, supprimer ». Plusieurs solutions, de la plus REST à la plus pragmatique :

- **Changer l'état** de la ressource : `PATCH /orders/42` avec `{"status": "cancelled"}`.
- **Utiliser le verbe qui correspond**, quand il existe : annuler une commande, c'est `DELETE /orders/42` (choix d'API-ritif).
- **Créer une sous-ressource** qui représente l'action : `POST /orders/42/payments` crée un paiement.
- En dernier recours, une **route d'action** : `POST /orders/42/cancel`. Ce n'est pas très REST, mais c'est clair et c'est très courant.

La règle d'or : un développeur qui découvre votre API doit pouvoir **deviner** une URL qu'il n'a jamais vue. Si `/cocktails/3/reviews` existe, il s'attend à trouver `/orders/42`, pas `/get0rderById?order=42`.

Revision #2

Created 2 October 2026 20:00:47 by Thibaud FRICHET

Updated 2 October 2026 21:06:37 by Thibaud FRICHET