

Pourquoi HATEOAS est rare

Fielding le répète depuis 2008 : sans hypermédia, ce n'est pas REST. Pourtant, la grande majorité des API publiques s'arrêtent au niveau 2 de Richardson. Pourquoi ?

Les raisons

Les clients codent quand même les URL en dur

La promesse « le serveur peut changer ses URL sans casser les clients » suppose que les clients suivent réellement les liens. En pratique, les développeurs lisent la documentation et écrivent `/v1/orders/42` directement dans leur code. Le bénéfice disparaît.

Un client ne comprend pas une relation inconnue

Un navigateur affiche un lien et laisse un humain décider. Un programme, lui, ne sait pas quoi faire d'un lien `refund` qu'il n'a jamais vu : quelqu'un doit avoir écrit le code qui le gère. Les liens ne remplacent donc pas la documentation.

Pas de standard unique

HAL, JSON:API, Siren, JSON-LD... Chaque format a ses outils, et aucun ne s'est imposé. Beaucoup d'équipes préfèrent ne rien choisir.

Des réponses plus lourdes

Des liens dans chaque élément d'une liste de 100 éléments, c'est du volume en plus, et du code serveur en plus pour les calculer selon l'état et les droits.

OpenAPI couvre une partie du besoin

La découverte de l'API se fait avec la documentation et les clients générés plutôt qu'à l'exécution.

Là où c'est utile

HATEOAS n'est pas tout ou rien. Certaines parties rendent service dans presque toutes les API :

- **La pagination** : `next`, `prev`, `first`, `last`. Presque toutes les API sérieuses le font.
- **Les workflows** : commandes, paiements, validations... les liens d'actions selon l'état évitent de dupliquer les règles métier dans chaque client.
- **Le header** `Location` après une création : c'est déjà un lien hypermédia.
- **Les agents IA** : un LLM qui découvre une API, lui, sait lire un lien qu'il n'a jamais vu et en déduire quoi faire. Un point d'entrée avec des liens explicites l'aide à naviguer sans documentation.

L'API REST de PayPal, par exemple, renvoie dans chaque paiement une liste de liens (`approve`, `capture`...) avec leur méthode HTTP. Le client suit le lien `approve` pour rediriger l'acheteur, sans construire l'URL.

En résumé

☐ Pour	☐ Contre
Le client suit des liens au lieu de construire des URL	Les clients codent souvent les URL en dur quand même
Les règles métier restent côté serveur	Un client ne sait pas gérer une relation inconnue
L'API se découvre depuis un point d'entrée	Pas de format standard unique
Utile pour la pagination, les workflows et les IA	Réponses plus lourdes, serveur plus complexe

La position raisonnable : des liens de pagination partout, des liens d'actions là où l'état compte, et une bonne documentation OpenAPI à côté.