Principes de Base des APIs REST

Définition d'une API REST

REST = Representational State Transfer Architecture pour systèmes distribués, basée sur : - **Ressources** identifiables (URI) - **Verbes HTTP** standardisés - **Stateless** : chaque requête contient toutes les informations - **Représentations** multiples (JSON, XML, etc)

Les Ressources

Toute donnée exposée est une ressource Exemples : - `/livres` - `/utilisateurs/42` - `/commandes/2023/10`

Bonnes pratiques : - Utiliser des **noms** (pas de verbes dans les URI) - **Pluriel** pour les collections `/books` □□ vs `/book` □ - Hiérarchie logique `/bibliotheques/5/livres`

Les Verbes HTTP

Verbe	Rôle	Idempotent
GET	Lire une ressource	□□
POST	Créer	□
PUT	Remplacer une ressource	□□
DELETE	Supprimer	□□
PATCH	Mise à jour partielle	□

Idempotent = Même résultat après exécutions multiples

Exemple de Requêtes

```
GET /api/v1/livres?categorie=sci-fi HTTP/1.1
Host: example.com
Accept: application/json

POST /api/v1/livres HTTP/1.1
Content-Type: application/json

{
  "titre": "Dune",
  "auteur": "Frank Herbert"
}
```

Codes Statut HTTP

Principaux codes : - `200 OK` : Succès - `201 Created` : Ressource créée - `204 No Content` : Pas de contenu à retourner - `400 Bad Request` : Requête malformée - `401 Unauthorized` : Authentification requise - `404 Not Found` : Ressource inexistante - `500 Internal Server Error` : Erreur serveur

Représentations

Même ressource → formats différents :

```
GET /livres/123 HTTP/1.1
Accept: application/json

GET /livres/123 HTTP/1.1
Accept: application/xml
```

Via header Content-Type et Accept

Bonnes Pratiques

Versionnage dans l'URI : /api/v1/...

Filtrage/Pagination : /livres?page=2&limit=10

Sécurité : HTTPS + JWT/OAuth2

Documentation (OpenAPI/Swagger)

HATEOAS pour la navigabilité

Anti-Patterns à Éviter

Verbes dans les URI : /getLivres □ → /livres □□

Codes statut génériques (200 pour les erreurs)

État côté serveur (REST doit être stateless)

Endpoints trop génériques : /api?action=deleteUser □

Outils pour Tester

Outil	Usage
cURL	CLI pour requêtes HTTP

Outil	Usage
Postman	Interface graphique
httpie	CLI moderne
Swagger UI	Documentation interactive

Pourquoi REST ?

Avantages :

- Standard largement adopté
- Scalable et flexible
- Découplage client/serveur
- Cache-friendly

Pourquoi REST ?

Cas d’usage :

- Microservices
- Applications web/mobiles
- Intégration de systèmes