APIs web

REST

REST (Representational State Transfer) ou RESTful permet de construire des api (Application Programming Interface ou Web Service), pour permettre la communication entre applications.
REST est fondé sur un ensemble de conventions et de bonnes pratiques à respecter et non sur une technologie à part entière. L'architecture REST utilise les spécifications du protocole HTTP, plutôt que de réinventer une surcouche (comme le font SOAP ou XML-RPC par exemple).

Principes REST

1. l'URI comme identifiant des ressources
2. les verbes HTTP comme identifiant des opérations
3. les réponses HTTP comme représentation des ressources
4. les liens comme relation entre ressources
5. un paramètre comme jeton d'authentification

1- URI

REST se base sur les URI pour identifier une ressource. Une API doit définir ses URI (et donc ses URL) de manière précise, en tenant compte des contraintes de lisibilité de REST, en prenant en compte la hiérarchie des ressources et la sémantique des URL :

Exemples de construction d'URL avec RESTful :

Liste des clients

```
NOK : https://myapi.com/client
OK : https://myapi.com/clients
```

Liste des clients avec filtre et tri

```
NOK : https://myapi.com/clients/filtre/caen/tri/asc
OK : https://myapi.com/clients?filtre=caen&tri=asc
```

Affichage d'un client

```
NOK : https://myapi.com/clients/display/87
OK : https://myapi.com/clients/87
```

Commandes d'un client

```
NOK : https://myapi.com/clients/commands/87
```

```
OK : https://myapi.com/clients/87/commands
```

Une Commande d'un client

```
NOK : https://myapi.com/clients/commands/87/1568
OK : https://myapi.com/clients/87/commands/1568
```

2 - Méthodes HTTP

Utilisation des verbes HTTP existants (méthodes) plutôt que d'inclure l'opération dans l'URI de la ressource.

Action	**Méthode (verbe)**
Créer (create)	POST
Afficher (read)	GET
Mettre à jour (update)	PUT
Supprimer (delete)	DELETE

Créer un client

```
NOK : GET https://myapi.com/clients/create
OK : POST https://myapi.com/clients
```

Afficher un client

```
NOK : GET https://myapi.com/clients/display/87
OK : GET https://myapi.com/clients/87
```

Mettre à jour un client

```
NOK : POST https://myapi.com/clients/update/87
OK : PUT https://myapi.com/clients/87
```

Supprimer un client

```
NOK : GET https://myapi.com/clients/delete/87
OK : DELETE https://myapi.com/clients/87
```

3 - Représentation de ressources

La réponse HTTP reçue est une représentation de ressource, et non la ressource elle même :

En fonction de la requête effectuée et de son en-tête **Accept**, plusieurs formmats de réponses sont envisageables :

- JSON
- XML

- HTML
- CSV
- etc…

Exemple

```
GET /clients
Host: myapi.com
Accept: application/xml
```

4 - Liens = relation entre ressources

Les liens d'une ressource indiquent les relations qu'elle peut avoir avac d'autres. Pour indiquer la nature de la relation, l'attribut **rel** doit être spécifié sur tous les liens. L'IANA donne une liste de relation parmi lesquelles :

- contents
- edit
- next
- last
- payment
- etc…

Liste complète sur le site de l'IANA

5 - Authentification par jeton

REST étant par principe **stateless** (pas de session HTTP par exemple), l'authentification se fait par jeton d'authentification.

Chaque requête est envoyée avec un jeton (token) passé en paramètre **GET** de la requête ou dans les headers. Ce jeton temporaire est obtenu en envoyant une première requête d'authentification puis en le combinant avec les requêtes.

JWT est l'un des standards d'authentification les plus utilisés.

Liens

- API Platform
- JamStack
- OpenAPI
- GraphQL
- Headless CMS Strapi