**API REST**

Reste et un modèle d’API basée sur http. Une API et faite par des développeurs pour des développeurs.

Contrainte REST :

1. Un client http et un serveur.

Rôle du client : forger une requête http

Rôle du serveur Renvoyer une réponse http

1. Sans état (State Less)

Ne pas garder d’historique entre chaque requête

1. Cache http sur les réponses
2. Système en couche (Layout System)

Le client doit recevoir une réponse dans tous les cas.

1. Interface uniforme (Manipuler des ressources)

La réponse serveur doit contenir :

* Id de la ressource
* Représentation json
* Entête « Content-Type » dans le header

1. Code sur demande (Facultatifs)

Envoi un code dans la réponse qui est à exécuter par le client.

Model de maturité de Richardson :

* Niveau 0 (Swamp of POX)

Requêter l’api sur un seul point d’entrée « http://mon-api.com/api » en POST.

* Niveau 1 (Ressouses)

Requêter via plusieurs points d’entrée en fonction des ressources « http://mon-api.com/api/users/1 » en POST.

* Niveau 2 (http Verbs)

Adapter la méthode post en fonction du type de requête. ex : GET pour récupérer, POST pour ajouter.

* Niveau 3 (Hypermedia controls)

Imiter le fonctionnent d’une page web (des liens dans la page). Retourner une réponse avec des liens liés à la ressource (Les action possible avec cette ressource).

Les méthodes http (CRUD):

GET : Récupérer une ou des ressources

POST : Ajouter une ressource

PUT : Modifier une ressource

DELETE : Supprimer une ressource

Les types de réponsses http :

* **200 OK** Tout s'est bien passé ;
* **201 Created** La création de la ressource s'est bien passée (en général le contenu de la nouvelle ressource est aussi renvoyée dans la réponse, mais ce n'est pas obligatoire - on ajoute aussi un header Locationavec l'URL de la nouvelle ressource) ;
* **204 No content** Même principe que pour la 201, sauf que cette fois-ci, le contenu de la ressource nouvellement créée ou modifiée n'est pas renvoyée en réponse ;
* **304 Not modified** Le contenu n'a pas été modifié depuis la dernière fois qu'elle a été mise en cache ;
* **400 Bad request** La demande n'a pas pu être traitée correctement ;
* **401 Unauthorized** L'authentification a échoué ;
* **403 Forbidden** L'accès à cette ressource n'est pas autorisé ;
* **404 Not found** La ressource n'existe pas ;
* **405 Method not allowed** La méthode HTTP utilisée n'est pas traitable par l'API ;
* **406 Not acceptable** Le serveur n'est pas en mesure de répondre aux attentes des entêtes Accept . En clair, le client demande un format (XML par exemple) et l'API n'est pas prévue pour générer du XML ;
* **500 Server error** Le serveur a rencontré un problème.

Sérialisation / désérialisation :

Sérialisation : Une ressource passe d’un état textuel a un objet (Ex : json vers entité Symfony)

Désérialisation : Passe d’un état d’objet a un état textuel. (Ex : Objet doctrine vers json)

Gestion des erreurs :

Une réponse serveur autre que 200 est une erreur. Il ne faut pas afficher la Stack trace, mais retourner un message spécifique à l’erreur.

Pagination :

Documentation :

Une api doit être correctement documentée.