

Spécifications fonctionnelles du système API Rest

Préparé pour : Florian Gimbert <[fgimbert@pageup.fr](mailto:fgimbert@pageup.fr)>

Préparé par : Johnny Charcosset <[jcharcosset@pageup.fr](mailto:jcharcosset@pageup.fr)>, Sébastien Theurelle <[stheurelle@pageup.fr](mailto:stheurelle@pageup.fr)>, Mathieu Perroud <[mperroud@pageup.fr](mailto:mperroud@pageup.fr)>

Date : 28/08/2019

Numéro de version : 1.0.0

Sommaire

[**Mise à jour de ressources.**](#_z5x4o66vi5jh) **7**

Historique des modifications

| Date | Nom | Description | Version |
| --- | --- | --- | --- |
| 28/08/2019 | CHARCOSSET Johnny <jcharcosset@pageup.fr> | Initialisation | 1.0.0 |

Introduction

Description de cas d’utilisation

Les communications entre les différentes plateformes doivent échanger sous forme d’API et respecter la convention REST.

Lexique

| Mot | Définition |
| --- | --- |
| API | Application Programming Interface |
| REST | Representational state transfer |
| HTTP | Hypertext Transfer Protocol |

Objectif

L’objectif est d’harmoniser la structure de nos API et de suivre au mieux possible les différentes bonnes pratiques et conventions au sein de nos projets.

Besoins

Les différents acteurs utilise les API comme moyen de communication. Néanmoins, chacun développe selon sa propre vision des choses son API. Un document récapitulant comment mettre en place une API structurée et maintenable est nécessaire.

Ce document est complémentaire au spécification des formats de retour.

Méthodes HTTP

Il existe plusieurs méthodes HTTP afin d’indiquer l’action souhaitée à réaliser.

Il existe plusieurs types méthodes :

* Méthodes sûres : Qui ne modifient rien sur le serveur et qui n’ont pas d’effet de bord
* Méthodes idempotentes : Qui modifient des données sur le serveur mais qui ont le même effet à chaque appel

GET

La méthode GET n’effectue aucune modification sur le serveur. Elle permet de récupérer des données et n’a pas d’autre effet. Il s’agit du méthode sûre et non idempotente.

POST

La méthode POST n’est pas une méthode sûre mais uniquement idempotente. Elle effectue une action sur le serveur : Celle de créer une ressource.

PUT

La méthode PUT effectue une action sur le serveur : Elle sert à modifier totalement une ressource. Elle est donc non sûre et idempotente.

PATCH

La méthode PATCH effectue une action sur le serveur : Elle sert à modifier partiellement une ressource. Elle est donc non sûre et non idempotente car nous ne pouvons pas contrôler les effets de bord. Nous pouvons modifier un champs X, tandis qu’un autre processus modifie en même temps un champs Y.

DELETE

La méthode DELETE permet de supprimer une ressource sur le serveur. Elle modifie donc le serveur, elle n’est pas sûre mais uniquement idempotente.

Pour résumer :

| Méthode | Idempotente | Sûre |
| --- | --- | --- |
| GET | Non | Oui |
| POST | Oui | Non |
| PUT | Oui | Non |
| PATCH | Non | Non |
| DELETE | Oui | Non |

Cette notion de méthode sûre et de idempotent est important afin d’identifier facilement quelle type de requête vont affecter le serveur et quelle type de ressource peuvent être mise en cache.

# 

Codes retours

Les codes retours permettent d’identifier rapidement le résultat de la requête exécutée.

Tous les retours de type 2xx sont des retours de requêtes correctement exécutées.

Tous les retours de type 4xx sont des codes d’erreur côté client.

Tous les retours de type 3xx sont des codes de redirection.

Tous les retours de type 5xx sont des codes d’erreur côté serveur.

Récupération de ressources

Le code retour 200 OK est utilisé lorsque l’on récupère correctement une ou des ressources demandées.

Si la ou les ressources demandées sont inexistantes, un code 404 NOT FOUND est retourné.

Création de ressources

Lorsqu’une ou des ressources sont créées correctement sur le serveur, un code 201 CREATED est retourné avec la ou les ressources nouvellement créées.

Si la ressource ou les ressources ne peuvent être créées car la requête est mal formatée, le code 400 BAD REQUEST est retourné.

Si la ou les ressources ne peuvent être créées car la requête est correctement formatée, mais que les données de celle-ci ne permettent pas de créer la ou les ressources, un code 422 UNPROCESSABLE ENTITY est retourné.

Si la ou les ressources demandées sont inexistantes, un code 404 NOT FOUND est retourné.

## Mise à jour de ressources.

Lorsqu’une ou des ressources ont été correctement mises à jour, un code 200 OK est retourné avec la ou les ressources modifiées.

Si la ressource ou les ressources ne peuvent être mises à jour car la requête est mal formatée, le code 400 BAD REQUEST est retourné.

Si la ou les ressources ne peuvent être mises à jour car la requête est correctement formatée, mais que les données de celle-ci ne permettent pas de mettre à jour la ou les ressources, un code 422 UNPROCESSABLE ENTITY est retourné.

Si la ou les ressources demandées sont inexistantes, un code 404 NOT FOUND est retourné.

Suppression de ressources

Lorsqu’une ou des ressources sont correctement supprimées, un code 200 OK est retournée.

Si la ou les ressources demandées sont inexistantes, un code 404 NOT FOUND est retourné.

Cas particulier

Si l’auteur de la demande n’est pas authentifié, un code 401 UNAUTHORIZED est retourné.

Si l’auteur de la demande n’a pas les permissions, un code 403 FORBIDDEN est retourné.

Si la méthode utilisée n’est pas la bonne méthode, un code 405 METHOD NOT ALLOWED est retournée.

Récapitulation

| Méthodes | Rôles | Codes de succès | Codes d’erreurs |
| --- | --- | --- | --- |
| GET | Récupération | 200 | 404, 405, 401, 403 |
| POST | Création | 201 | 404, 400, 422, 405, 401, 403 |
| PUT | Mise à jour totale | 200 | 404, 400, 422, 405, 401, 403 |
| PATCH | Mise à jour partielle | 200 | 404, 400, 422, 405, 401, 403 |
| DELETE | Suppression | 200 | 404, 405, 401, 403 |

Urls

Les urls doivent suivre un modèle précis.

Elles doivent être en anglais.

Elles doivent être au pluriel.

Elles ne doivent pas répéter la méthode HTTP utilisée.

Elles doivent être préfixée par /api

Prenons la collection de User suivante :

| [  {  "id" : 1,  "name" : "Eric Clapton",  "roles" : ["guitarist"]  },  {  "id" : 2,  "name" : "Ginger Baker",  "role" : ["drummer"]  },  {  "id" : 1,  "name" : "Jack Bruce",  "role" : ["singer", "bassist"]  } ] |
| --- |

Récupération d’une ressource

GET /api/users/{id\_user}

Récupération de toutes les ressources

GET /api/users

Mise à jour d’une ressource

PUT /api/users/{id\_user}

PATCH /api/users/{id\_user}

Suppression d’une ressource

DELETE /api/users/{id\_user}

Récupération d’une ressource imbriquée

GET /api/users/{id\_user}/roles/{id\_role}

Récupération des toutes les ressources imbriquées

GET /api/users/{id\_user}/roles

Mise à jour d’une ressource imbriquée

PUT /api/users/{id\_user}/roles/{id\_role}

PATCH /api/users/{id\_user}/roles/{id\_role}

Suppression d’une ressource imbriquée

DELETE /api/users/{id\_user}/roles/{id\_role}

Réponses

Pour le format de réponse, voir la documentation des spécifications fonctionnelles concernant le format des réponses.

Versionnage

Il est parfois nécessaire de versionner l’API afin de faire cohabiter les différentes évolutions.

La version ne doit pas être spécifié dans l’url.

La version est uniquement spécifiée dans le header.

La clé du header doit être X-Accept-Version.

Ce paramètre est optionnel.