

# WORKSHOP RESTFUL

CESI

## RESUME

L'objectif de ce workshop est de comprendre le fonctionnement et de manipuler une API publique, puis de mettre en pratique des mécanismes asynchrones.

**GALLET** *Jérémy*

Web Services / REST

## Sommaire :

<b>1 INTRODUCTION</b>	<b>2</b>
Objectif	2
Prérequis	2
<b>2 EXEMPLE D'UTILISATION D'UNE API</b>	<b>3</b>
Introduction	3
Mise en œuvre	3
Liste des utilisateurs	3
Informations pour un utilisateur	4
Création d'un utilisateur	4
Autres actions	4
<b>3 MISE EN PRATIQUE</b>	<b>5</b>
Objectif	5
Mise en place	5
Création de la page	5
Partie 1 : Affichage des utilisateurs	6
Partie 2: Création d'un utilisateur	8
Améliorations	9
<b>4 POUR ALLER PLUS LOIN</b>	<b>10</b>
Réalisation du prosit	10
API publiques	10

# 1 INTRODUCTION

---

## Objectif

L'objectif de ce workshop est de vous faire manipuler une API publique. Le but est en premier lieu de bien comprendre l'architecture d'une API RESTful et de manipuler les requêtes via un outil de construction de requêtes. La seconde partie de ce Workshop consistera à mettre en place des mécanismes asynchrones en utilisant JavaScript et jQuery.

## Prérequis

Vous aurez besoin dans la première partie de l'outil Postman que vous avez déjà dû installer en amont de ce workshop.

---

### Questions

---

1. *Quelles sont les verbes HTTP les plus populaires ? Quelles actions CRUD peut-on leur associer ?*
2. *Quelles sont les différences entre URL, URI et URN ?*
3. *Quelle est la structure d'une requête émise ? D'une requête reçue ?*
4. *Comment sont passés les paramètres ?*

## 2 EXEMPLE D'UTILISATION D'UNE API

### Introduction

Dans cette partie nous allons nous appuyer sur des données fictives hébergées sur le site de Reqres. Ce service met gratuitement à disposition une architecture API REST permettant de tester les requêtes.

### Mise en œuvre

Commencez par ouvrir l'application Postman (ou utilisez la version Web) précédemment installée.

#### Liste des utilisateurs

Dans Postman effectuez une requête GET en utilisant l'URL suivante : <https://reqres.in/api/users>

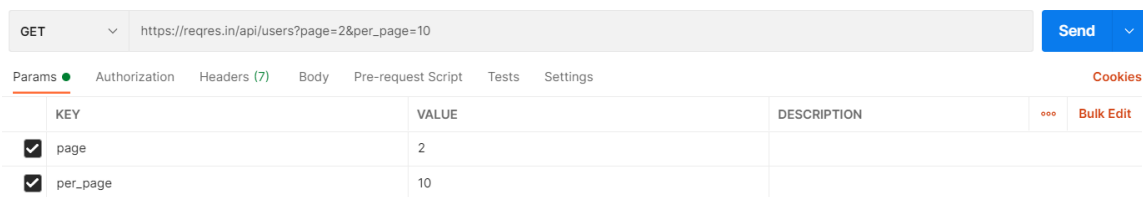


A screenshot of the Postman interface. The top bar shows a 'GET' method selected from a dropdown menu, followed by the URL 'https://reqres.in/api/users' in a text input field. To the right of the input field is a blue 'Send' button with a small downward arrow.

Observez la réponse de la requête :

1. *Quel est le statut ?*
2. *Quel est le format par défaut ?*
3. *Comment sont structurées les données ?*

En utilisant les paramètres (key/value), effectuez cette même requête en demandant une page particulière (par exemple la deuxième) et un avec 10 résultats par page.



A screenshot of the Postman interface showing a GET request with query parameters. The top bar shows the URL 'https://reqres.in/api/users?page=2&per\_page=10'. Below the top bar, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is selected, and it shows a table with two rows of parameters: 'page' with value '2' and 'per\_page' with value '10'. To the right of the table is a 'Cookies' section with a 'Bulk Edit' button.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> page	2	
<input checked="" type="checkbox"/> per_page	10	

*Comment sont passés les paramètres dans l'URL ?*

## Informations pour un utilisateur

Dans Postman effectuez une requête GET pour afficher les données d'un utilisateur précis. L'id à renseigner est celui présent dans l'un des résultats de la requête précédente.

GET	https://reqres.in/api/users/3	Send
-----	-------------------------------	------

*Comment est passé le paramètre relatif à l'utilisateur ?*

Utiliser à présent la même requête en mettant un id qui n'existe pas (par exemple 100).

*Que se passe-t-il dans la réponse ? Quel est le code d'erreur ?*

## Création d'un utilisateur

En utilisant une requête POST, créez un utilisateur.

POST	https://reqres.in/api/users?name=Steve&job=Director	Send
------	---	------

*Que se passe-t-il dans la réponse ? Quel est le code de retour ?*

## Autres actions

Vous pouvez essayer d'autres verbes HTTP et actions en testant les différentes requêtes offertes par Reqres.

**i** Vous pouvez effectuer des requêtes GET directement depuis votre navigateur, car les endpoints REST utilisent le même protocole HTTP que le web. Le résultat de la requête s'affichera en texte brut dans le navigateur (sauf si vous avez installé dans celui-ci une extension qui formate correctement le résultat).

Par contre les autres actions (POST/PUT/DELETE...) ne seront pas possibles, c'est tout l'intérêt d'un outil comme Postman !

## 3 MISE EN PRATIQUE

---

### Objectif

Cette partie va vous permettre de mettre en œuvre les mécanismes asynchrones en utilisant l'API de Reqres.

Le principe consiste à créer une page HTML qui permettra, au click sur un bouton, de remplir dynamiquement la page avec les informations récupérées sans recharger la page. Pour ce faire nous allons nous appuyer sur une technologie appelée **AJAX** (Asynchronous JavaScript and Xml) que nous allons utiliser avec du code JavaScript natif puis la librairie jQuery.

### Mise en place

#### Création de la page

Vous pouvez partir sur un nouveau projet HTML ou utiliser un existant.

Pour la seconde partie vous aurez besoin de **jQuery** pour manipuler les données.

Insérer le code suivant dans la partie body de votre page HTML :

```
<main>
  <article>
    <h2>Partie 1 - Affichage des utilisateurs (JavaScript natif)</h2>
    <div id="js_result"></div>
    <div><button id="btn_js">GET Data</button></div>
  </article>
  <article>
    <h2>Partie 2 - Création d'un utilisateur (jQuery)</h2>
    <div id="jquery_result"></div>
    <div><button id="btn_jquery">POST data</button></div>
  </article>
</main>
```

#### Fonctionnement global :

- Lors du click sur le bouton « GET Data », une requête **asynchrone** sera envoyée en utilisant du code JavaScript natif. Le résultat sera interprété en JavaScript puis le code HTML généré sera injecté dans le cadre ayant pour id `js_result`.
- Lors du click sur le bouton « POST Data », une requête asynchrone sera envoyée en utilisant jQuery. Le résultat sera interprété en JavaScript puis le code HTML généré sera injecté dans le cadre ayant pour id `jquery_result`.

# Partie 1 : Affichage des utilisateurs

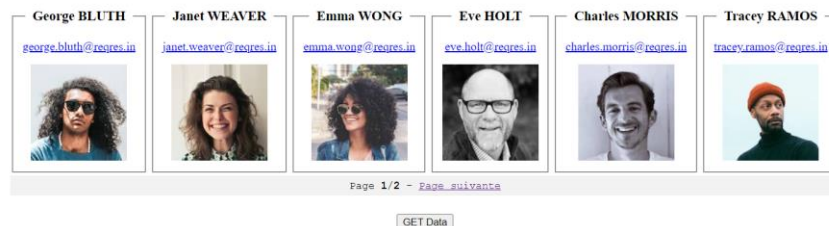
## Résultat final

Cette première partie va vous permettre de mettre en œuvre les mécanismes asynchrones en utilisant du code JavaScript natif.

### WORKSHOP API RESTFUL

---

#### Partie 1 - Affichage des utilisateurs (Javascript natif)



## Réalisation

Comme vous l'avez fait avec Postman au début du Workshop, vous allez ici récupérer la liste des utilisateurs via une requête GET.

En JavaScript natif, l'objet **XMLHttpRequest (XHR)** permet d'envoyer des requêtes HTTP de manière très simple. Il suffit de créer une instance de l'objet, d'ouvrir une URL, et d'envoyer la requête. Le statut HTTP du résultat, tout comme le contenu de la réponse, sont disponibles dans l'objet de la requête quand la transaction est terminée.

L'URL à utiliser est la suivante : `https://reqres.in/api/users`

Insérer le code JavaScript suivant directement dans une balise JavaScript de votre page HTML ou dans un fichier à part :


```
document.getElementById("btn_js").onclick = function() // Interception du click sur le bouton
{
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "https://reqres.in/api/users", true);
  xhr.onload = function()
  {
    var html = "" ;
    if( xhr.status == 200 )
    {
      ... code HTML
    }
    else{}
    document.getElementById("js_result").innerHTML = html;
  };
  xhr.send(); //Envoi de la requête au serveur (asynchrone par défaut)
};
```

Compléter ce code en JavaScript natif en effectuant les manipulations suivantes après réception de la requête :

- Faire une boucle sur chaque utilisateur en récupérant leur nom, adresse e-mail et avatar
- Récupérer le numéro de la page en cours et le nombre de pages
- Afficher le lien Page suivante et/ou Page précédente en fonction du résultat. Au click sur ce lien la requête sera renvoyée en demandant la bonne page.

**Astuce :** Pour faciliter le déploiement et ne pas dupliquer le code, vous pouvez créer une fonction `get_users` ayant pour paramètre le numéro de la page à demander.

Sur le fond le résultat devrait ressembler à la capture d'écran présente un peu plus haut. A moins que vous soyez très en avance, ne vous attardez pas sur la partie graphique.

 Vous pouvez utiliser la commande `console.log()` pour afficher des informations dans l'outil de développement intégré à votre navigateur.

Penser à bien **commenter chaque ligne de votre code** quand cela est nécessaire afin de bien comprendre son fonctionnement.

## Questions

---

1. *Quelle est la différence entre une requête synchrone et asynchrone ?*
2. *Quels sont les avantages et les inconvénients d'AJAX ?*



## Partie 2: Création d'un utilisateur

### Résultat final

Cette seconde partie va vous permettre d'envoyer des données via une méthode POST en utilisant les fonctions asynchrones de jQuery.

#### Partie 2 - Création d'un utilisateur (jQuery)

```
{
  "name": "Jérémy GALLET",
  "job": "Enseignant / Formateur",
  "place": "Saint-Hazaire",
  "id": "181",
  "createdAt": "2021-02-09T13:05:01.460Z"
}
```

POST data

### Réalisation

Comme pour la partie précédente, la même URL sera utilisée mais cette fois ci en utilisant une méthode POST.

En jQuery, la fonction **\$.ajax()** permet de créer un objet de type XHR. Les appels aux fonctions asynchrones sont donc simplifiés par rapport à un appel JavaScript natif mais le fonctionnement et le résultat derrière restent identiques.

Insérer le code JavaScript suivant au même emplacement que le précédent :

```
$("#btn_jquery" ).click(function()
{
  $.ajax({
    url: "https://reqres.in/api/users/",
    type: "POST",
    data: {name: "your_name", job: "what_you_want", place: "where_you_want"},
    success: function(response, textStatus, xhr)
    {
      var html = '';
      if( xhr.status == 201 )
      {
        ... code HTML
      }
      else
      {
      }
      $('#jquery_result').html(html);
    }
    error: function (xhr, ajaxOptions, thrownError)
    {
      $('#jquery_result').html('Error: ' + xhr.status);
      console.log(thrownError);
    }
  });
});
```

Compléter ce code en utilisant jQuery afin d'insérer dans le div `jquery_result` les résultats suivants :

- Affichage du résultat brut de la requête
- Affichage de chaque résultat (clé + valeur) en utilisant une boucle

**Note** : les données envoyées sont fictives, vous pouvez mettre ce que vous voulez.

**i** Pour envoyer des données de type POST ou GET, vous pouvez aussi utiliser les raccourcis `$.get()` et `$.post()` qui proposent une méthode simplifiée.

---

### Question

*Quels sont les 4 paramètres qui peuvent être utilisés par la méthode AJAX de jQuery ?*

## Améliorations

Lorsque tout fonctionne correctement autant en JavaScript natif qu'en jQuery, vous pouvez vous atteler à l'amélioration du code (par exemple pour la gestion des erreurs) et à la partie graphique.

## 4 POUR ALLER PLUS LOIN

---

### Réalisation du prosit

Vous êtes le roi de l'asynchrone ? Vous pouvez aussi passer à la réalisation du Prosit :

1. Récupérer le formulaire de facturation utilisé dans le Workshop précédent
2. En utilisant les données du site APICARTO, remplissez dynamiquement le nom de la ville en fonction du code postal saisie.

**Ressource de l'API :**

[https://apicarto.ign.fr/api/doc/codes-postaux#/codes-postaux/get\\_codes\\_postaux\\_communes\\_codePostal](https://apicarto.ign.fr/api/doc/codes-postaux#/codes-postaux/get_codes_postaux_communes_codePostal)

### API publiques

De nombreux services et réseaux sociaux que vous utilisez certainement tous les jours ont leurs propres API. Allez voir ce qu'ils proposent, la documentation est très bien faite et vous pourrez manipuler aisément les API REST au travers de leur interface de programmation. Vous aurez ainsi une bonne visibilité des formats et de l'aspect sécurité (token).

**Exemples d'API :**

- [Facebook](#)
- [Instagram](#)
- [Twitter](#)
- [GitHub](#)