**🚀 Guide Rapide Postman pour le Test d'API**

**Postman** est un outil essentiel pour tester, développer et documenter des API. Il vous permet d'envoyer des requêtes HTTP (GET, POST, PUT, DELETE, etc.) sans écrire une seule ligne de code côté client.

**1. Création et Envoi d'une Requête de Base**

**A. Les Éléments de la Requête**

Dans l'onglet principal d'une requête, vous configurez ces trois éléments clés :

1. **Méthode HTTP :** Sélectionnez la méthode appropriée pour l'action que vous souhaitez effectuer :
   * **GET** : Récupérer des données (lecture).
   * **POST** : Envoyer des données pour créer une nouvelle ressource.
   * **PUT** / **PATCH** : Modifier ou mettre à jour une ressource existante.
   * **DELETE** : Supprimer une ressource.
2. **URL de l'API :** Entrez l'adresse complète du point de terminaison (*endpoint*) de votre API (ex: http://localhost:3000/api/utilisateurs).
3. **Bouton Send :** Exécutez la requête.

**B. Configuration des Données**

Selon la méthode utilisée, vous devez configurer les données de la requête :

| Type de Donnée | Onglet Postman | Utilisation |
| --- | --- | --- |
| **Paramètres de requête** | **Params** | Pour les filtres ou les options passés directement dans l'URL (ex: ?limit=10&page=1). Postman les ajoute à l'URL. |
| **Données d'envoi** | **Body** | Pour les méthodes POST, PUT, PATCH. Sélectionnez raw puis JSON pour envoyer des données structurées (très courant pour les API REST). |
| **Authentification** | **Authorization** | Pour ajouter un token Bearer, une clé API, ou une authentification Basic. |
| **En-têtes** | **Headers** | Pour définir des informations de communication, comme le type de contenu (Content-Type: application/json) ou le token d'authentification. |

Exporter vers Sheets

**2. Analyse de la Réponse**

Après avoir cliqué sur Send, Postman affiche la réponse de l'API dans la partie inférieure.

| Élément de la Réponse | Description | Importance |
| --- | --- | --- |
| **Code de Statut** | Le code HTTP renvoyé par le serveur (ex: **200 OK**, **201 Created**, **404 Not Found**, **500 Internal Server Error**). | **Crucial** pour valider le succès ou l'échec de l'opération. |
| **Body (Corps)** | Les données renvoyées par l'API (généralement au format JSON). | Contient les données récupérées ou le message de confirmation/erreur. |
| **Time** | Le temps nécessaire à la réponse. | Indique la performance de l'API. |

Exporter vers Sheets

**3. Optimisation avec les Collections et les Environnements**

**A. Collections (Organiser vos tests)**

Une **Collection** est un dossier qui regroupe un ensemble de requêtes liées.

* **Intérêt :** Organiser les requêtes par projet ou par fonctionnalité.
* **Utilisation :** Vous pouvez exécuter toutes les requêtes d'une collection en séquence (via le **Collection Runner**), ce qui est utile pour les tests d'intégration.

**B. Environnements (Gérer les variables)**

Un **Environnement** est un ensemble de variables.

* **Intérêt :** Gérer les variables spécifiques à différents contextes (Développement, Staging, Production) sans modifier les requêtes.
* **Exemple :** Vous définissez une variable baseURL (http://localhost:3000 pour le dev et https://api.prod.com pour la production).
* **Utilisation :** Dans une requête, vous utilisez la variable comme ceci : {{baseURL}}/api/utilisateurs. En changeant l'environnement actif, vous changez automatiquement la cible de l'API.

**4. Tests Automatisés (Scripts)**

Postman vous permet d'ajouter du code JavaScript pour automatiser des actions.

**A. Scripts Pre-request**

Ce script s'exécute **avant** que la requête ne soit envoyée.

* **Usage typique :** Générer des données aléatoires ou calculer un jeton d'authentification requis pour la requête suivante.

**B. Scripts Tests**

Ce script s'exécute **après** avoir reçu la réponse du serveur. C'est ici que l'on fait les validations.

| Commande Clé | Description | Exemple |
| --- | --- | --- |
| **pm.test()** | Définit un test spécifique avec un nom lisible. | pm.test("Le statut doit être 200 OK", function () { ... }); |
| **response.code** | Accède au code de statut HTTP. | pm.response.to.have.status(200); |
| **response.json()** | Accède au corps de la réponse JSON. | pm.expect(responseData.utilisateur.nom).to.eql("Alice"); |

Exporter vers Sheets

**Exemple de test :**

JavaScript

// Vérifie le code de statut

pm.test("Le statut est bien 'Created' (201)", function () {

pm.response.to.have.status(201);

});

// Vérifie qu'un champ existe dans le corps de la réponse

pm.test("La réponse contient un ID utilisateur", function () {

const responseData = pm.response.json();

pm.expect(responseData).to.have.property('id');

});