

# Tutorial Postman

## Table des matières

1.	Présentation Postman.....	3
2.	Postman .....	3
2.1.	Installation .....	3
2.2.	Présentation de l'interface .....	4
2.3.	Workspaces .....	4
2.4.	Collections .....	4
2.5.	Environnements .....	5
2.6.	Requêtes .....	5
3.	Application PyFlightsApp .....	5
4.	Préparation .....	6
4.1.	Création d'un workspace .....	6
4.2.	Création de collection .....	6
4.3.	Importer / Exporter une collection .....	7
4.4.	Création d'une requête API .....	8
4.5.	Liste des requêtes FlightsAPI utilisées .....	8
4.6.	Liste des requêtes Taiga.io utilisées.....	10
5.	Exercice 1 : Rechercher un vol.....	11
5.1.	Requête GetFlight : recherche par numéro.....	11
5.2.	Requête GetFlights : recherche par ville de départ, d'arrivée et par date .....	12
5.3.	Consulter le log d'exécution .....	15
6.	Exercice 2 : Les tests.....	16
6.1.	Valider le code retour de la requête .....	17
6.2.	Vérifier la présence d'un texte dans la réponse .....	19
6.3.	Vérifier le contenu d'une balise JSON .....	19
7.	Exercice 3 : Enchaîner plusieurs requêtes avec les variables globales.....	21
7.1.	Enregistrer la donnée FlightNumber dans une variable globale .....	21
7.2.	Utiliser la variable dans la requête BookFlight et enregistrer le numéro de réservation .....	22
7.3.	Afficher le contenu des variables globales .....	25
7.4.	Exécuter l'enchaînement complet des tests .....	26
8.	Exercice 4 : Variables d'environnement.....	27
8.1.	Création des variables d'environnement.....	28
8.2.	Rendre les requêtes indépendantes de l'adresse ses services Rest .....	29
8.3.	Utiliser les variables ville de départ, ville d'arrivée et date dans GetFlights.....	30
8.4.	Modifier BookFlight en utilisant la variable Date .....	31
8.5.	Exécuter la séquence et visualiser les variables .....	31
9.	Exercices 5 : Utiliser les données dans un fichier csv et JSON .....	33
9.1.	Utiliser les données dans un fichier csv .....	33
9.2.	Exécuter la séquence de test.....	35
9.3.	Utiliser les données dans un fichier json .....	38
10.	Exercice 6 : Créer une boucle .....	38
10.1.	Lire le nombre total de réservation .....	40
10.2.	Réaliser la boucle .....	40
2.	Compléments .....	42
3.	Cas pratique.....	43
3.1.	Gestion de projets Agile avec Taiga.io .....	43
3.2.	Créer un projet démo .....	45
3.3.	L'API rest .....	47
3.4.	Travail à faire .....	47

# 1. Présentation Postman

Lire le support de formation "Introduction aux tests Web Services.pdf"

## 2. Postman

- Logiciel gratuit et open source multi-plateforme. Des versions permettant le travail collaboratif (Team, Business, Entreprise) sont payantes.
- L'ensemble des fonctionnalités intégrées sur Postman, permettent une gestion de bout-en-bout du cycle de vie d'un API
- Permet de concevoir et simuler, déboguer, tester, documenter, surveiller et publier les API à partir de l'interface utilisateur
- Postman client web est disponible via le lien suivant : [go.postman.co/build](https://go.postman.co/build)

### 2.1. Installation

Télécharger Postman <https://www.postman.com/downloads/>

# The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

 **Download the App**

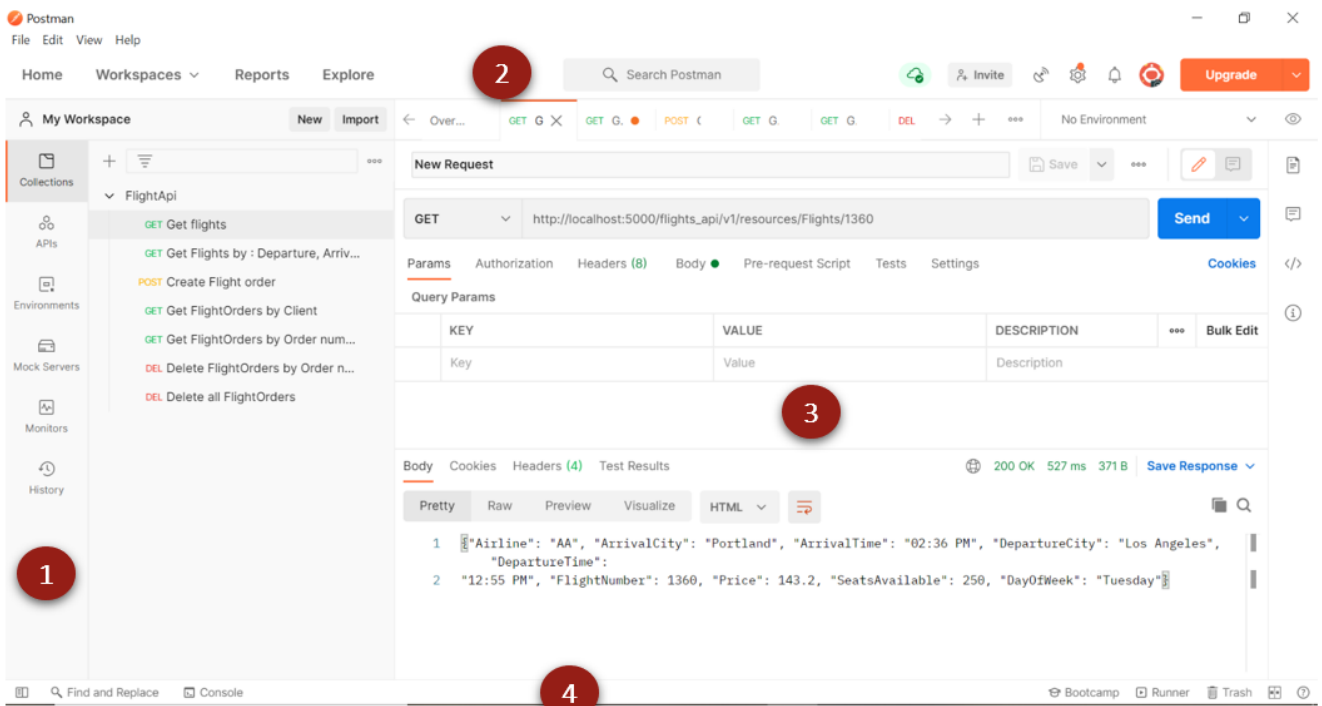
By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

**Version 8.1.0** | [Release Notes](#) | [Product Roadmap](#)

**Not your OS?** Download for Mac ([macOS](#)) or Linux ([x64](#))

La procédure d'installation : <https://learning.postman.com/docs/getting-started/installation-and-updates/>

## 2.2. Présentation de l'interface



1. La barre latérale gauche permet d'accéder aux collections, APIs, environnements, serveurs de simulation, moniteurs et historique des requêtes lancées.
2. L'en-tête permet de création des Workspaces, d'accéder aux rapports, d'explorer le réseau API public, de rechercher dans Postman, d'afficher l'état de synchronisation et les notifications, de déplacer et d'inviter des collaborateurs dans les Workspaces, de capturer des demandes et des cookies, d'accéder à vos paramètres, compte et plan Postman
3. La zone centrale est l'endroit où vous construisez et travaillez avec les demandes
4. La barre d'état en bas vous permet d'afficher / masquer la barre latérale, de rechercher et de remplacer et d'ouvrir la console sur la gauche. Sur la droite, vous pouvez lancer le Bootcamp, le collection runner, Trash, la vue à deux volets et accéder aux ressources d'aide.

## 2.3. Workspaces

Les espaces de travail personnels permettent aux utilisateurs de créer autant d'espaces de travail personnels qu'ils le souhaitent. Il est possible d'organiser les collections, l'environnement et d'autres éléments Postman dans des espaces de travail spécifiques à une rubrique.

Lorsque vous ouvrez Postman pour la première fois, vous serez dans votre espace de travail personnel par défaut.

## 2.4. Collections

Les collections sont le format d'API le plus utilisé. À l'aide des collections, vous pouvez regrouper les demandes, les paramètres, les descriptions, les tests et les scripts dans un seul dossier. Vous pouvez collaborer sur des API via des

collections partagées et créer des suites de tests, de la documentation, des serveurs simulés et des moniteurs sur des collections.

## 2.5. Environnements

Les variables d'environnement peuvent être utilisées dans Postman afin d'exécuter des API sur différents serveurs. Les variables peuvent être utilisées presque partout. Les variables sont accessibles à l'aide d'accolades doubles.

## 2.6. Requêtes

Vous pouvez envoyer des requêtes aux API dans Postman. Une requête API vous permet de récupérer des données à partir d'une source de données ou d'envoyer des données. Les API s'exécutent sur des serveurs Web et exposent des points de terminaison pour prendre en charge les opérations qu'utilisent les applications clientes pour fournir leurs fonctionnalités.

Chaque demande d'API utilise une méthode HTTP. Les méthodes les plus courantes sont GET, POST, PATCH, PUT et DELETE.

Dans Postman, vous pouvez effectuer des requêtes API et examiner les réponses sans utiliser de terminal ni écrire de code. Lorsque vous créez une demande et cliquez sur Envoyer, la réponse de l'API apparaît dans l'interface utilisateur de Postman.

# 3. Application PyFlightsApp

L'application Flights est une application de démonstration qui permet de réserver des vols.

Pour réserver un vol :

1. Recherche la disponibilité du vol, avec la ville de départ, la ville d'arrivée et la date du vol
2. Sélectionner un vol disponible
3. Réserver le vol avec le numéro du vol, la date du vol, le nom du client, la classe du siège et le nombre de ticket
4. Récupérer le numéro de réservation et le prix total du billet

L'application permet de rechercher un vol par son numéro ou par le nom du client.

On peut ensuite modifier les informations de la réservation ou l'annuler.

La documentation de l'api REST est disponible :

<https://app.swaggerhub.com/apis/imhah-hassan/PyFlightsApi>

## 4. Préparation

### 4.1. Création d'un workspace

Workspaces ^ Reports Explore

na

Search for a workspace + New Workspace

### Create New Workspace

Name

Formation Postman

Summary

Visibility

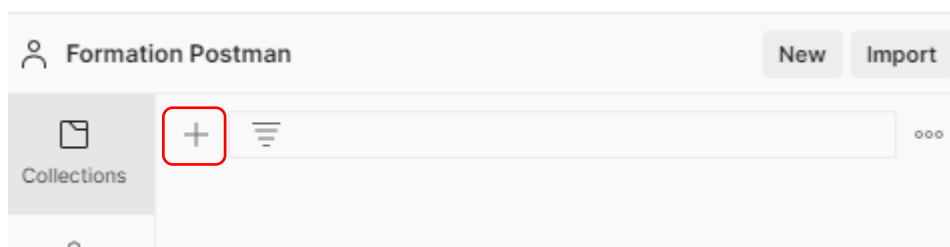
Personal v

Only you can access

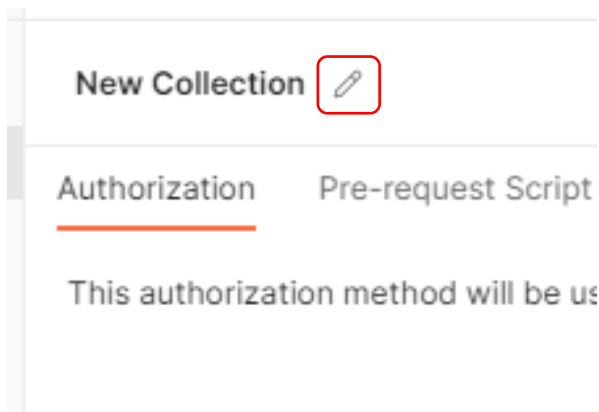
Cancel Create Workspace

### 4.2. Création de collection

Créer une collection **FlightsApi** dans Postman ou TaigaIssue pour les exercices avec Taiga.io



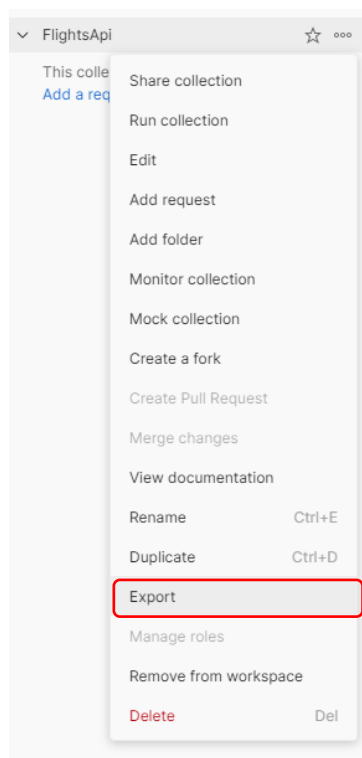
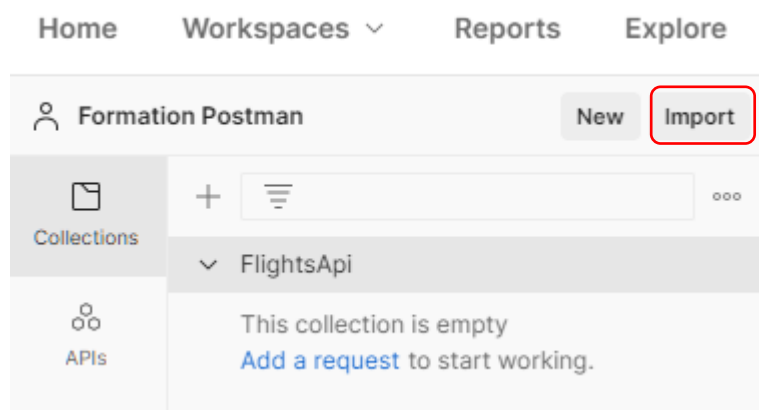
Cliquer sur le bouton « + » à côté de la barre latérale gauche. Cela créera une nouvelle collection sous le nom par défaut « New collection » qu'on renommera à « FlightsApi »



### 4.3. Importer / Exporter une collection

Pour exporter une collection, il suffit de faire un clic droit sur la collection qu'on souhaite exporter, et cliquer sur le bouton « Export », et par la suite choisir l'emplacement du fichier JSON.

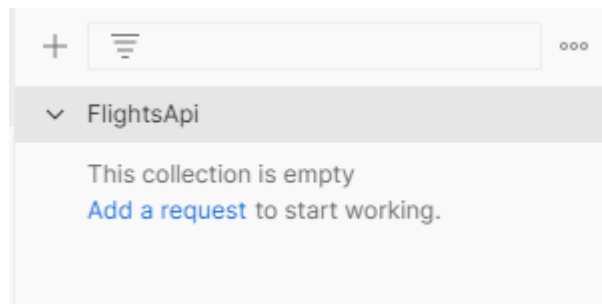
L'import d'une collection se fait en cliquant sur le bouton « Import », juste en dessous des collections.



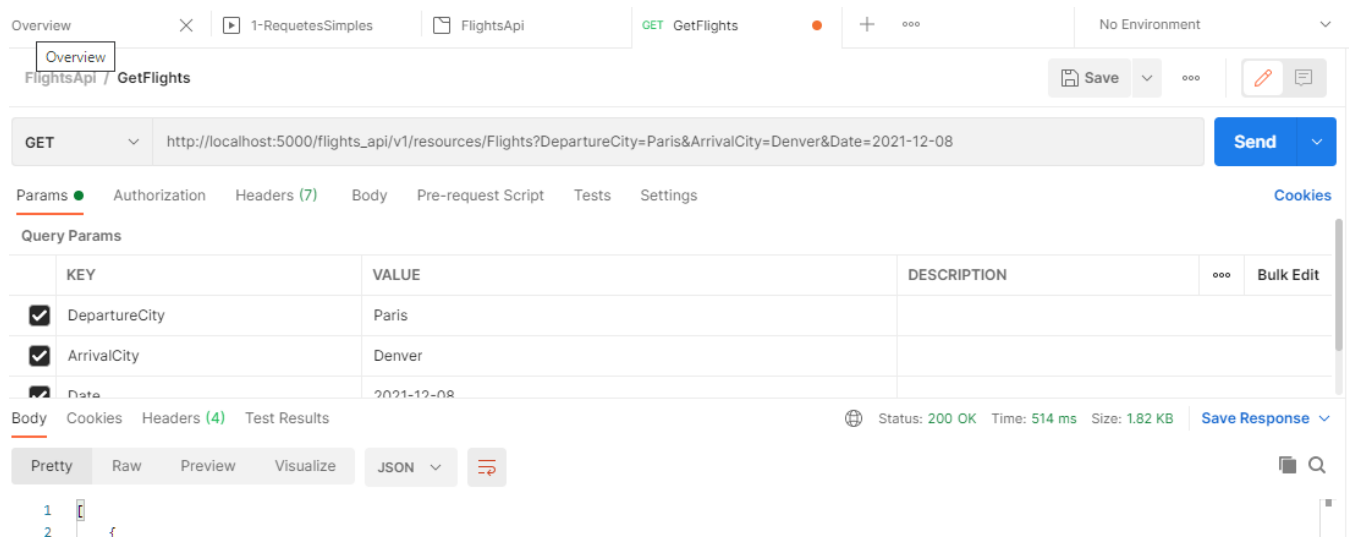
## 4.4. Création d'une requête API

Cliquer sur la collection, puis sur **Add a request**

Exemple Taiga : [https://api.taiga.io/api/v1/projects/by\\_slug?slug=himhah-demo\\_api](https://api.taiga.io/api/v1/projects/by_slug?slug=himhah-demo_api)



Choisir un nom pour la requête et de préférence significatif.



**N.B :** Postman ne permet pas la sauvegarde automatique des requêtes. Donc il est important de sauvegarder la requête via le bouton « Save » indiqué sur la figure ci-dessus.

## 4.5. Liste des requêtes FlightsAPI utilisées

URL : [http://localhost:5000/flights\\_api/v1/resources](http://localhost:5000/flights_api/v1/resources)

Rechercher un vol par son numéro :

Ressource	/Flights/{FlightNumber}
-----------	-------------------------



Méthode	GET
Paramètre	Aucun
Retour	[{"Airline": "SR", "ArrivalCity": "Denver", "ArrivalTime": "12:43 PM", "DepartureCity": "Paris", "DepartureTime": "11:02 AM", "FlightNumber": 16939, "Price": 168.2, "SeatsAvailable": 250, "DayOfWeek": "Wednesday"}, ...]

Recherche un vol par ville de départ, d'arrivée et par date

Ressource	/Flights
Méthode	GET
Paramètre	DepartureCity=Paris&ArrivalCity=Denver&Date=2021-12-08
Retour	{"Airline": "TWA", "ArrivalCity": "San Francisco", "ArrivalTime": "12:43 PM", "DepartureCity": "Seattle", "DepartureTime": "11:02 AM", "FlightNumber": 1060, "Price": 168.2, "SeatsAvailable": 250, "DayOfWeek": "Wednesday"}

Prendre une réservation

Ressource	/FlightOrders
Méthode	POST
Headers	Content-Type : application/json
body	{ "Class" : "Economy", "CustomerName" : "IMHAH", "DepartureDate" : "2022-12-08", "FlightNumber" : "1060", "NumberOfTickets" : "2" }
Retour	{ "OrderNumber": 107, "TotalPrice": 336.4 }

Rechercher une réservation par le nom du client

Ressource	/FlightOrders
Méthode	GET
Paramètre	CustomerName={{customer}}
Retour	[{"Class": "Economy", "CustomerName": "IMHAH", "DepartureDate": "2021-12-8 11:02", "FlightNumber": 1060, "NumberOfTickets": 2, "OrderNumber": 105}]

Rechercher une réservation par numéro

Ressource	/FlightOrders/{order}
Méthode	GET
Paramètre	Aucun
Retour	[{"Class": "Economy", "CustomerName": "IMHAH", "DepartureDate": "2021-12-8 11:02", "FlightNumber": 1060, "NumberOfTickets": 2, "OrderNumber": 105}]

Modifier une réservation par numéro

Ressource	/FlightOrders/{order}
Méthode	PATCH
Headers	Content-Type : application/json
body	{ "Class" : "First", "CustomerName" : "IMHAH", }

	<pre>"DepartureDate" : "20221-12-09", "FlightNumber" : "1060", "NumberOfTickets" : "5" }</pre>
Retour	<pre>{   "Class": "First",   "CustomerName": "IMHAH",   "DepartureDate": "2022-12-8 11:02",   "FlightNumber": 1060,   "NumberOfTickets": 5,   "OrderNumber": 89,   "TotalPrice": 336.4 }</pre>

Supprimer une réservation par numéro

Ressource	/FlightOrders/{order}
Méthode	DELETE
Paramètre	
Retour	<pre>{   "true": "Oder deleted 105 " }</pre>

Supprimer toutes les réservations

Ressource	/FlightOrders
Méthode	DELETE
Paramètre	
Retour	{"result": "ok"}

## 4.6. Liste des requêtes Taiga.io utilisées

A faire : sélectionner une liste de requetes dans

<https://docs.taiga.io/api.html#projects-list>

<https://docs.taiga.io/api.html#issues-list>

[https://tree.taiga.io/project/himhah-demo\\_api/issues](https://tree.taiga.io/project/himhah-demo_api/issues)

## 5. Exercice 1 : Rechercher un vol

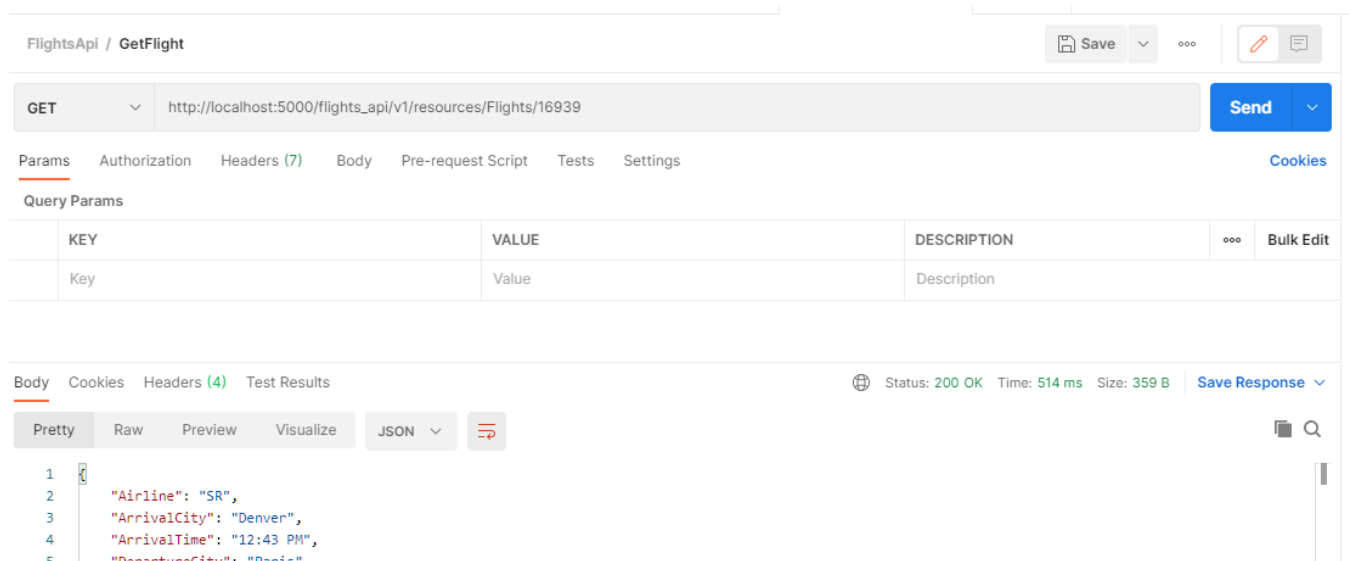
Objectif : Déclarer une requête Rest simple dans Postman

### 5.1. Requête GetFlight : recherche par numéro

On commencera par lancer une requête simple qui permettra de récupérer les informations d'un vol via son numéro.

Pour ce faire, il suffit de créer une requête comme vu ci-dessus, qu'on nommera « GetFlight ».

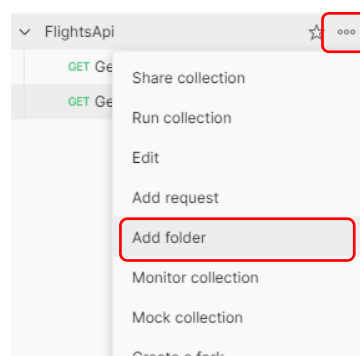
Ensuite dans l'URL de la requête, sélectionner la méthode GET ainsi que l'URL correspondant (Endpoint + Ressource + ID du vol ) ce qui donne : [http://localhost:5000/flights\\_api/v1/resources/Flights/16939](http://localhost:5000/flights_api/v1/resources/Flights/16939)



Ainsi, on a en retour une réponse 200 OK et les informations en format JSON du vol.

Créer un dossier Ex1-RequetesSimples

Clique sur les ... puis sur **add folder**



Puis renommer le dossier



Sauvegarder la requête dans ce dossier



Cliquer sur Save ou par **CTRL + S**

Puis déplacer la requête dans le dossier Ex1-RequetesSimples par drag & drop

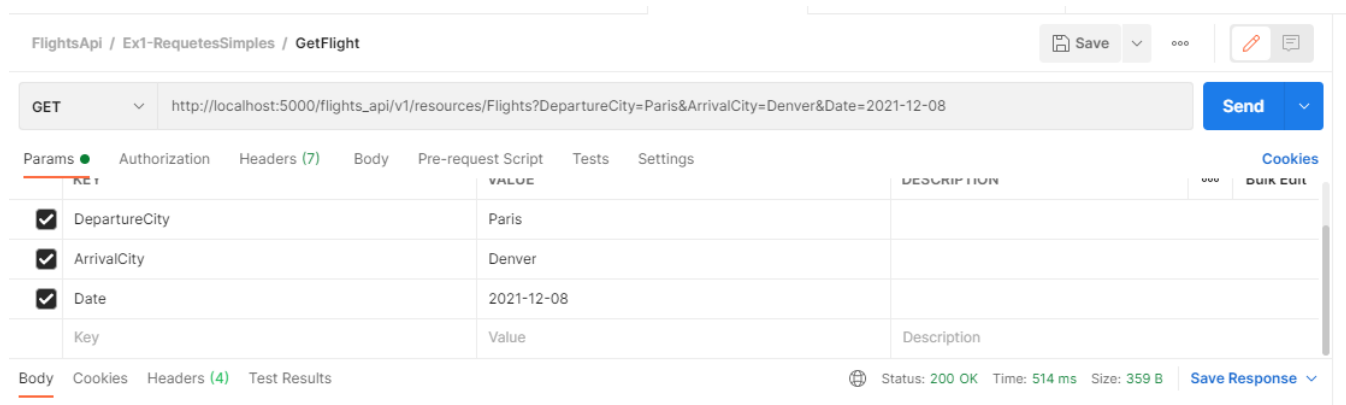
## 5.2. Requête GetFlights : recherche par ville de départ, d'arrivée et par date

Par la suite, on lancera une requête simple qui permettra de récupérer les informations de vols avec en paramètres la ville de départ, la ville d'arrivée et la date.

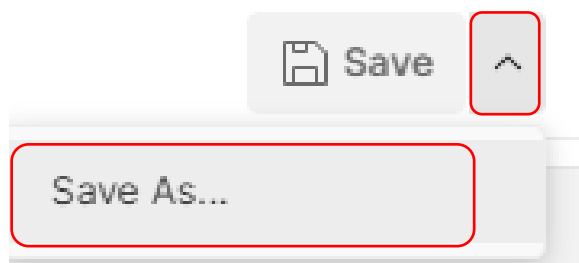
Pour ce faire on va utiliser la requête suivante :

[http://localhost:5000/flights\\_api/v1/resources/Flights?DepartureCity=Paris&ArrivalCity=Denver&Date=2021-12-08](http://localhost:5000/flights_api/v1/resources/Flights?DepartureCity=Paris&ArrivalCity=Denver&Date=2021-12-08)

Changer les paramètres de la 1ere requête et exécuter la requête modifiée



Déployer le menu Save et choisir **Save As**



Sélectionner le dossier **Ex1-RequetesSimples** et sauvegarder

Save to Ex1-RequetesSimples

SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).  
[Learn more about creating collections](#)

Request name

Request description (Optional)  

Make things easier for your teammates with a complete request description.

Descriptions support [Markdown](#)

Select a collection or folder to save to:  

Ex1-RequetesSimples [+ Create Folder](#)

GET GetFlight

Cancel

Save to Ex1-RequetesSimples

Poursuivre l'exercice en créant d'autres requêtes :

- Prendre une réservation

POST

{{baseURL}}/FlightOrders

Send

Params

Authorization

Headers (10)

Body

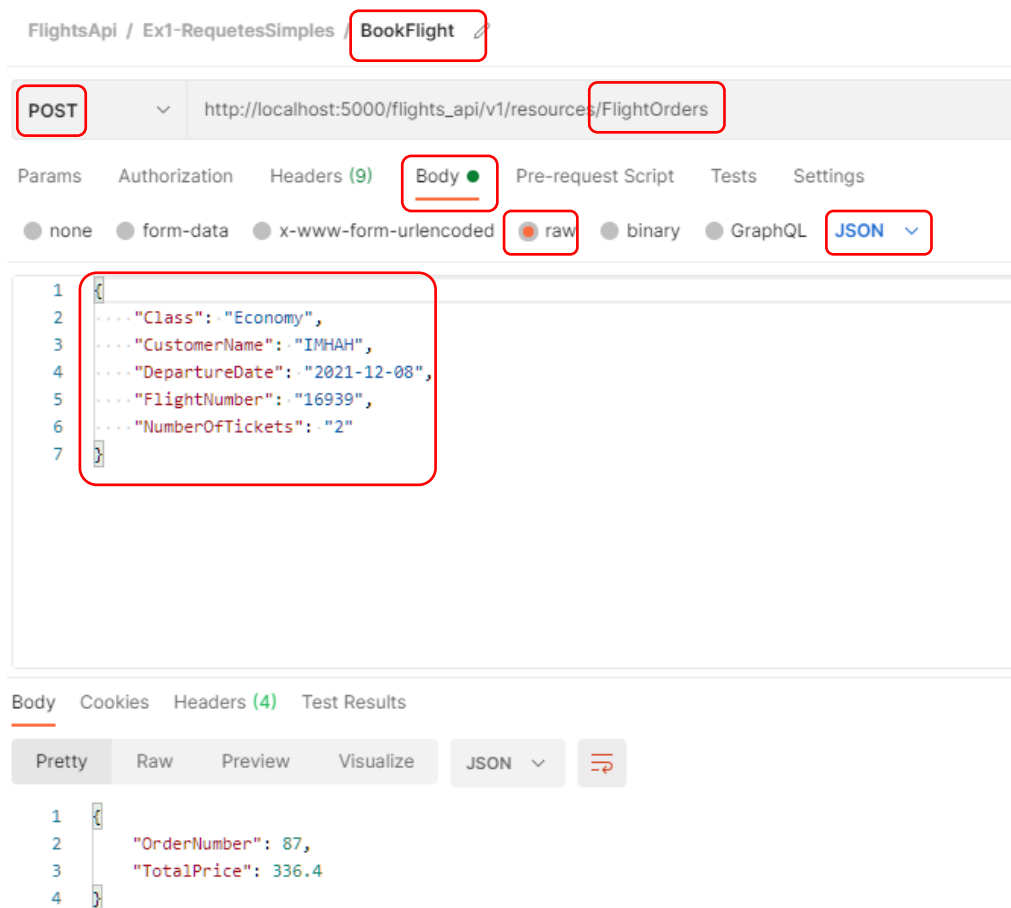
Pre-request Script

Tests

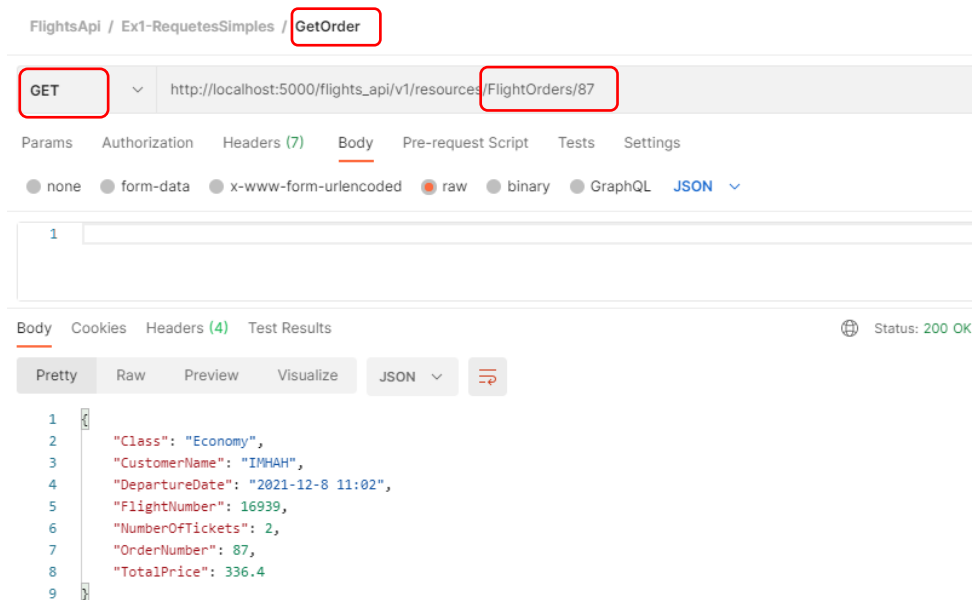
Settings

Cookies

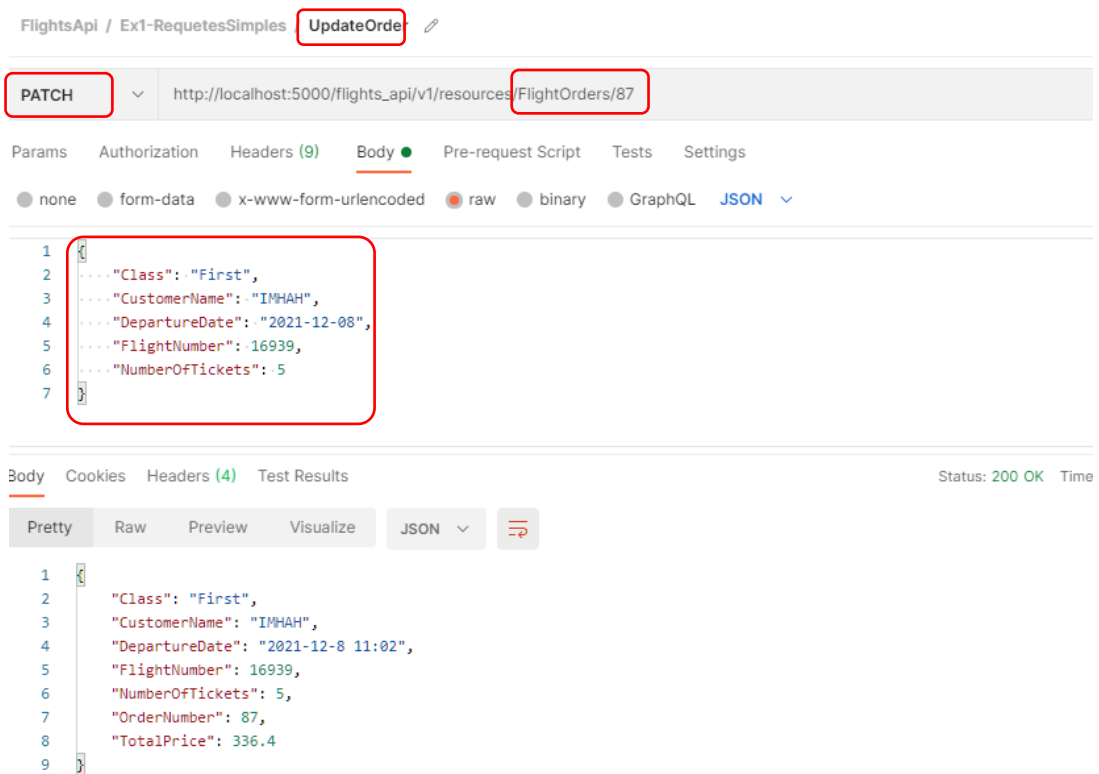
<input checked="" type="checkbox"/>	Postman-Token	<calculated when request is sent>
<input type="checkbox"/>	Content-Type	text/plain
<input checked="" type="checkbox"/>	Content-Length	<calculated when request is sent>
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.29.2
<input checked="" type="checkbox"/>	Accept	/*/*
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br
<input checked="" type="checkbox"/>	Connection	keep-alive
<input checked="" type="checkbox"/>	Content-Type	application/json



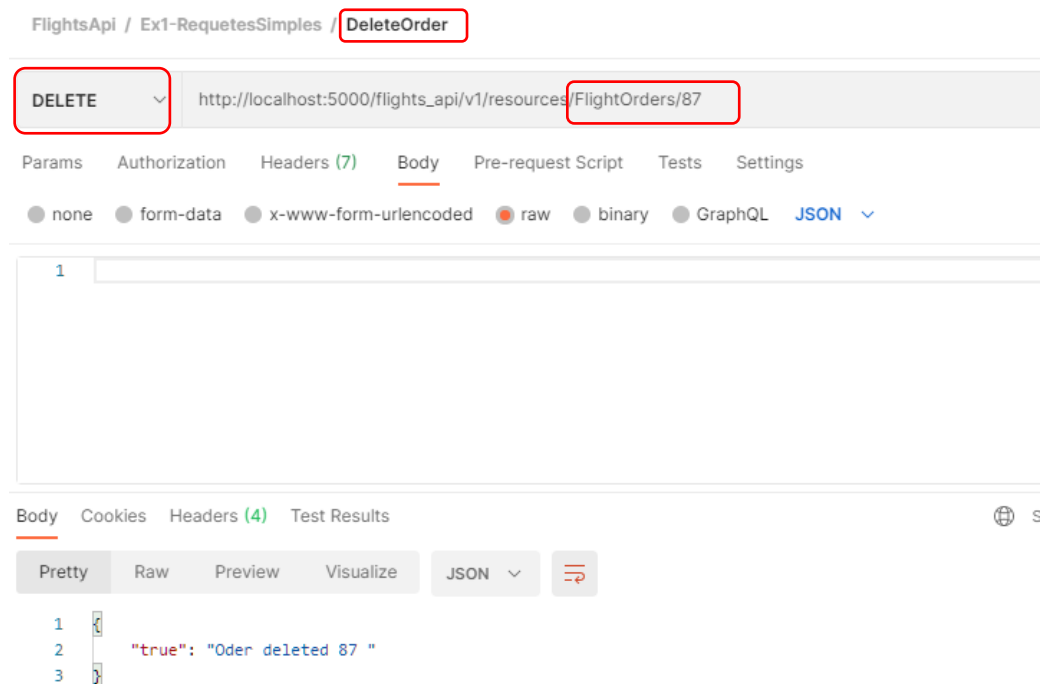
- Lire une réservation



- Modifier la réservation

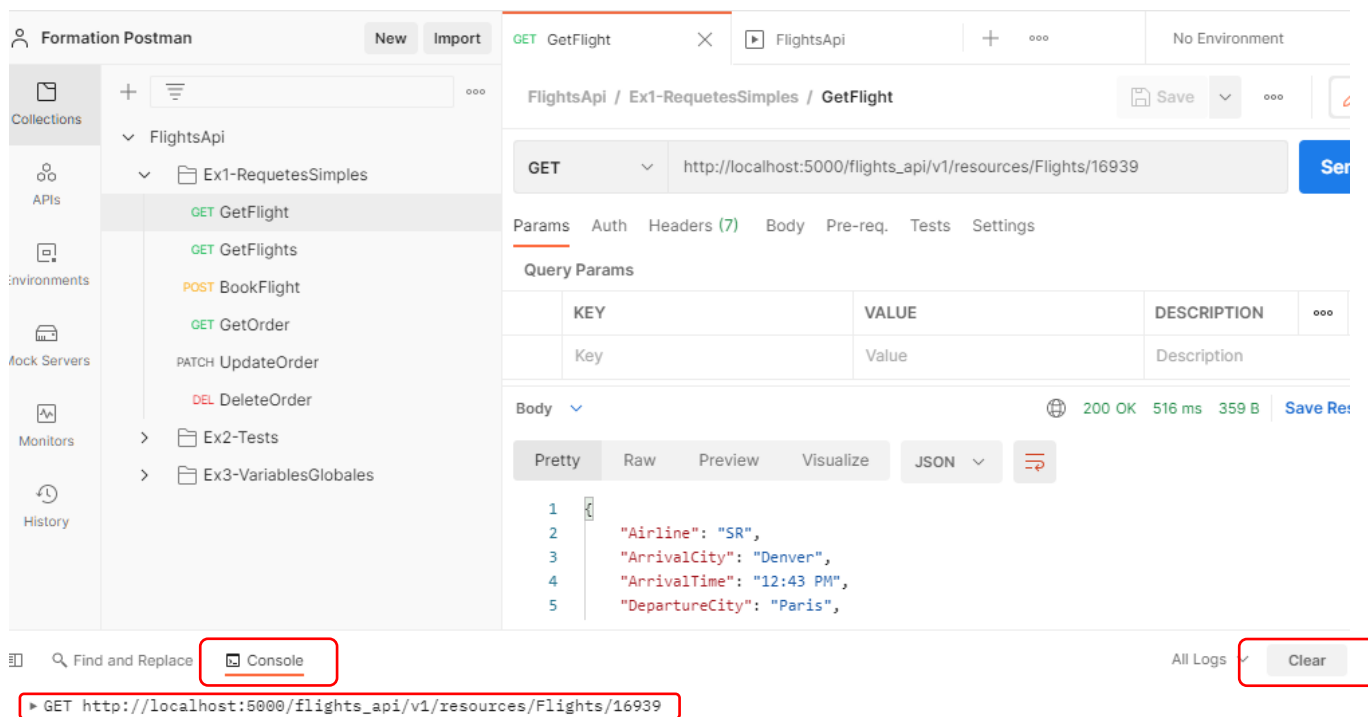


- Supprimer la réservation



### 5.3. Consulter le log d'exécution

Après chaque exécution, vous pouvez consulter le log d'exécution en cliquant sur Console



Pour effacer le contenu cliquer sur Clear.

Pour afficher le détail, cliquer sur la requête



## 6. Exercice 2 : Les tests

**Objectif : Ajouter des vérifications à un requêtes REST**

Introduction

Pour comprendre la syntaxe Postman, il est recommandé d'apprendre les éléments de base du langage JavaScript, par exemple avec le site Mozilla

[https://developer.mozilla.org/fr/docs/Learn/Getting\\_started\\_with\\_the\\_web/JavaScript\\_basics](https://developer.mozilla.org/fr/docs/Learn/Getting_started_with_the_web/JavaScript_basics)

La référence de javascript en Postman :

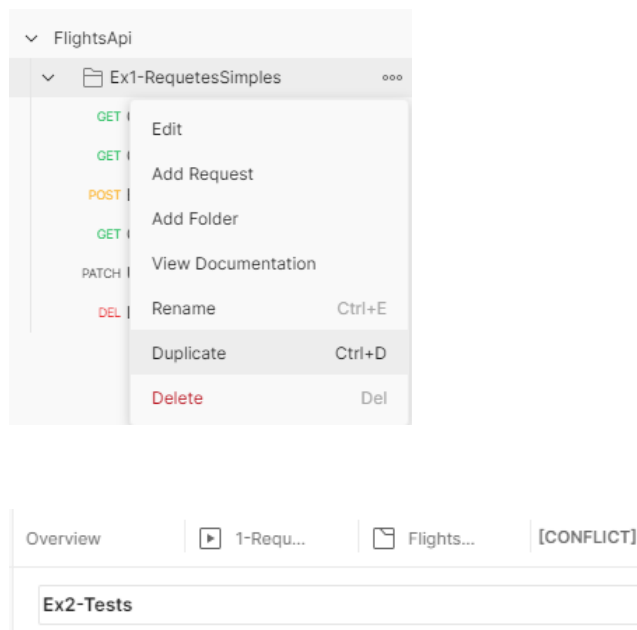
Tutorial Postman



<https://learning.postman.com/docs/writing-scripts/script-references/postman-sandbox-api-reference/>

Postman dispose de plusieurs raccourcis qui évitent de saisir le code javascript.

Avant de commencer dupliquer le dossier **Ex1-RequetesSimples** en **Ex2-Tests**



## 6.1. Valider le code retour de la requête

Sélectionner la requête GetFlight et cliquer sur Send, puis choisir l'onglet Tests

FlightsApi / Ex2-Tests / GetFlight

GET http://localhost:5000/flights\_api/v1/resources/Flights/16939 Send

Params Auth Headers (7) Body Pre-req. Tests **Settings** Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (4) Test Results 200 OK 527 ms 359 B Save Response

Pretty Raw Preview Visualize JSON 🔍

```

1  {
2    "Airline": "SR",
3    "ArrivalCity": "Denver",
4    "ArrivalTime": "12:43 PM",
5    "DepartureCity": "Paris",
6    "DepartureTime": "11:02 AM",
7    "FlightNumber": 16939,
8    "Price": 168.2,
9    "SeatsAvailable": 250,
10   "DayOfWeek": "Wednesday"
11 }

```

Nous utiliserons le bloque Snippets pour ajouter des vérifications =>

Pour ajouter la vérification du code retour, cliquer sur le Snippets  
**Status code : code is 200**

Le code suivant est ajouté automatiquement

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
```

Modifier l'intitulé du test par le texte « **Le code retour est 200** »

Exécuter la requête et afficher le résultat du test

**Cookies**

Test scripts are written in JavaScript, and are run after the response is received. [Learn more about tests scripts](#)

**SNIPPETS**

- [Get an environment variable](#)
- [Get a global variable](#)
- [Get a variable](#)
- [Get a collection variable](#)
- [Set an environment variable](#)
- [Set a global variable](#)
- [Set a collection variable](#)
- [Clear an environment variable](#)
- [Clear a global variable](#)
- [Clear a collection variable](#)
- [Send a request](#)
- [Status code: Code is 200](#)
- [Response body: Contains string](#)
- [Response body: JSON value check](#)
- [Response body: Is equal to a string](#)

FlightsApi / Ex2-Tests / GetFlight

GET http://localhost:5000/flights\_api/v1/resources/Flights/16939

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

```

1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4

```

Test scripts are written in JavaScript, and are run after the response is received. [Learn more about tests scripts](#)

SNIPPETS

- [Clear an environment variable](#)
- [Clear a global variable](#)
- [Clear a collection variable](#)
- [Send a request](#)

Body Cookies Headers (4) Test Results (1/1) 200 OK 524 ms 359 B Save Response

All Passed Skipped Failed

PASS Vérifier que le code retour est 200

## 6.2. Vérifier la présence d'un texte dans la réponse

Snippet [Response body: Contains string](#) vérifier la présence de la balise *FlightNumber*

Exécuter le test et lire le résultat.

FlightsApi / Ex2-Tests / GetFlight

GET http://localhost:5000/flights\_api/v1/resources/Flights/16939

Params Authorization Headers (7) Body Pre-request Script Tests Settings

```

1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("La balise FlightNumber est présente", function () {
6   pm.expect(pm.response.text()).to.include("FlightNumber");
7 });
8
9

```

Body Cookies Headers (4) Test Results (2/2) 200 OK

All Passed Skipped Failed


PASS Le code retour est 200


PASS La balise FlightNumber est présente

## 6.3. Vérifier le contenu d'une balise JSON

Snippet *"Response body: JSON value check"*, vérifier la valeur de la balise *FlightNumber*

Exécuter le test et lire le résultat.

FlightsApi / Ex2-Tests / GetFlight 

GET  http://localhost:5000/flights\_api/v1/resources/Flights/16939

Params Authorization Headers (7) Body Pre-request Script **Tests** Settings

```


1  pm.test("Le code retour est 200", function () {
2      pm.response.to.have.status(200);
3  });
4
5  pm.test("La balise FlightNumber est présente", function () {
6      pm.expect(pm.response.text()).to.include("FlightNumber");
7  });
8
9  pm.test("Le numéro de vol est 16939", function () {
10     var jsonData = pm.response.json();
11     pm.expect(jsonData.FlightNumber).to.eql(16939);
12 });
13

```

Test scripts are written in JavaScript and are run after the response is received. [Learn more about tests](#)

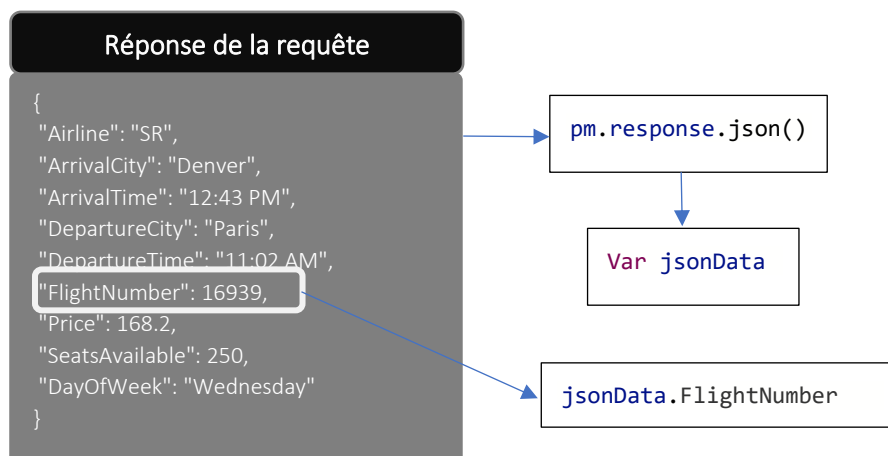
SNIPPETS

- Status code: Code is 200
- Response body: Contains
- Response body: JSON
- Response body: Is equal to
- Response headers: Contains
- Response time is less than

Body Cookies Headers (4) **Test Results (3/3)**  200 OK 517 ms 359 B

All Passed Skipped Failed

- PASS** Le code retour est 200
- PASS** La balise FlightNumber est présente
- PASS** Le numéro de vol est 16939



Ajouter des vérifications équivalentes aux requêtes : GetFlights, BookFlight, GetOrder, UpdateOrder et DeleteOrder

Pour les réponses contiennent plusieurs résultats voici comme on les vérifie :

```

pm.test("FlightNumber est présent dans json", function () {
  var jsonData = pm.response.json();
  for (let i = 0; i < jsonData.length; i++) {
    pm.expect(jsonData[i].ArrivalCity).to.eql("Denver");
  }
});

```

En plus, on peut les mettre en commentaire les tests dont on n'a pas besoin mais on ne veut pas les supprimer par : **CRL + / ou avec //** avant les bouts de code.

```
// pm.test("FlightNumber est présent dans json", function () {
```

pour vérifier l'absence d'un élément, on peut faire comme cela :

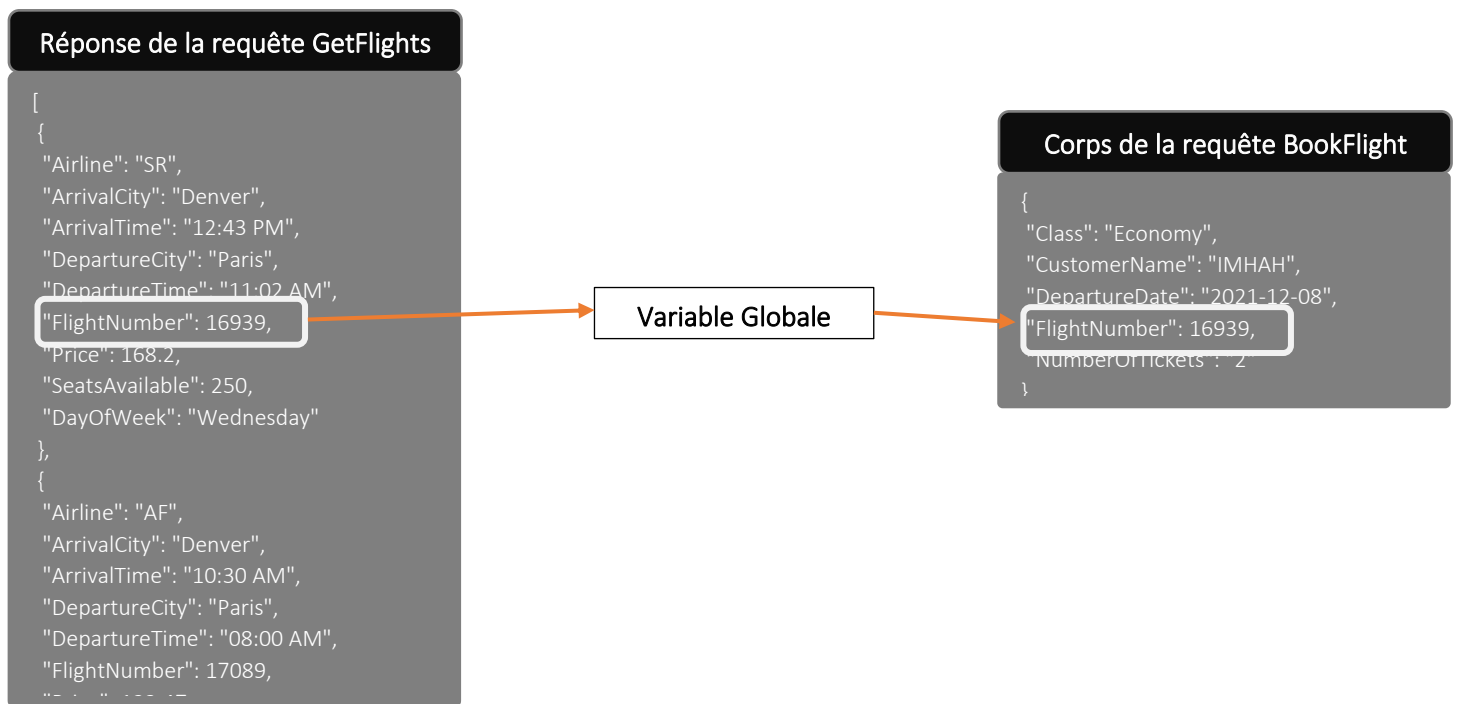
```
pm.expect(pm.response.text()).to.not.include("OrderNumber");
```

## 7. Exercice 3 : Enchaîner plusieurs requêtes avec les variables globales

Objectif : Simuler un scénario d'utilisation

Le scénario simple pour cet exercice est le suivant :

- Rechercher les vols disponibles avec une ville de départ, une ville d'arrivée et une date
- Sélectionner le 1<sup>er</sup> vol disponible
- Prendre une réservation sur ce vol



Avant de commencer dupliquer le dossier **Ex2-Tests** en **Ex3-VariablesGlobales**

### 7.1. Enregistrer la donnée FlightNumber dans une variable globale

Reprenez la requête GetFlights et ajouter les tests :

- Code retour 200
- Ville de départ du premier vol = Paris
- Le numéro de vol du 1<sup>er</sup> vol est au format nombre

Ajouter la snippet **Set a global variable** après la dernière vérification

FlightsApi / Ex3-VariablesGlobales / GetFlights

GET ▼ http://localhost:5000/flights\_api/v1/resources/Flights?DepartureCity=Paris&ArrivalCity=Denver&Date=2021-12-08

Params ● Authorization Headers (7) Body Pre-request Script Tests ● Settings

```
1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("La ville de départ du 1er vol est Paris", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData[0].DepartureCity).to.eql("Paris");
7 });
8
9 pm.test("Le 1er vol est au format nombre", function () {
10   var jsonData = pm.response.json();
11   pm.expect(jsonData[0].FlightNumber).to.be.a("number");
12
13   pm.globals.set("FlightNumber", jsonData[0].FlightNumber);
14 });
```

Body Cookies Headers (4) Test Results (3/3) 🌐 Status: 200 OK Time: 5

All Passed Skipped Failed

**PASS** Le code retour est 200

**PASS** La ville de départ du 1er vol est Paris

**PASS** Le 1er vol est au format nombre

## 7.2. Utiliser la variable dans la requête BookFlight et enregistrer le numéro de réservation

Il suffit de modifier dans le corps de la requête la valeur 16939 par {{FlightNumber}}

FlightNumber est le nom de la variable globale sauvegardée dans l'étape précédente

Exécuter la requête et vérifier que les tests sont OK (ajouter des tests si nécessaires)

Ajouter le snippet **Set a global variable** pour sauvegarder le numéro de réservation

FlightsApi / Ex3-VariablesGlobales / BookFlight

POST http://localhost:5000/flights\_api/v1/resources/FlightOrders

Params Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

```
1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("La balise TotalPrice est présente", function () {
6   pm.expect(pm.response.text()).to.include("TotalPrice");
7 });
8
9 pm.test("Le numéro de réservation est supérieur à 80", function () {
10   var jsonData = pm.response.json();
11   pm.expect(jsonData.OrderNumber).to.greaterThan(80);
12
13   pm.globals.set("OrderNumber", jsonData.OrderNumber);
14 });
15
```

Test scri  
are run a  
Learn mi


SNIPPET  
Get an e  
Get a glc  
Get a vai  
Get a coi  
Set an ei  
Set a glo  
Set a col

Body Cookies Headers (4) Test Results (3/3) Status: 200 OK Time: 532 ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "OrderNumber": 94,
3   "TotalPrice": 336.4
4 }
```



La variable globale OrderNumber sera ensuite utilisé dans les requêtes : GetOrder, UpdateOrder et DeleteOrder

GET  http://localhost:5000/flights\_api/v1/resources/FlightOrders/{{OrderNumber}}

Params Authorization Headers (7) Body Pre-request Script Tests ● Settings

```
1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Le numéro de vol est "+pm.variables.get("FlightNumber"), function () {
6   var jsonData = pm.response.json();
7   pm.expect(jsonData.FlightNumber).to.eql(pm.variables.get("FlightNumber"));
8 });
9 pm.test("la classe est Economy", function () {
10   var jsonData = pm.response.json();
11   pm.expect(jsonData.Class).to.eql("Economy");
12 });
```

Body Cookies Headers (4) Test Results (3/3)

Pretty Raw Preview Visualize JSON  

```
1 {
2   "Class": "Economy",
3   "CustomerName": "IMHAH",
4   "DepartureDate": "2021-12-8 11:02",
5   "FlightNumber": 16939,
6   "NumberOfTickets": 2,
7   "OrderNumber": 81,
8   "TotalPrice": 336.4
9 }
```



DELETE

http://localhost:5000/flights\_api/v1/resources/FlightOrders/{{OrderNumber}}

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests ●

Settings

```
1 pm.test("le code retour est 200", function () {
2   | pm.response.to.have.status(200);
3 });
4
5 pm.test("Le message contient true pour confirmer la suppression", function () {
6   | pm.expect(pm.response.text()).to.include("true");
7 });
```

Body

Cookies

Headers (4)

Test Results (2/2)

Pretty

Raw

Preview

Visualize

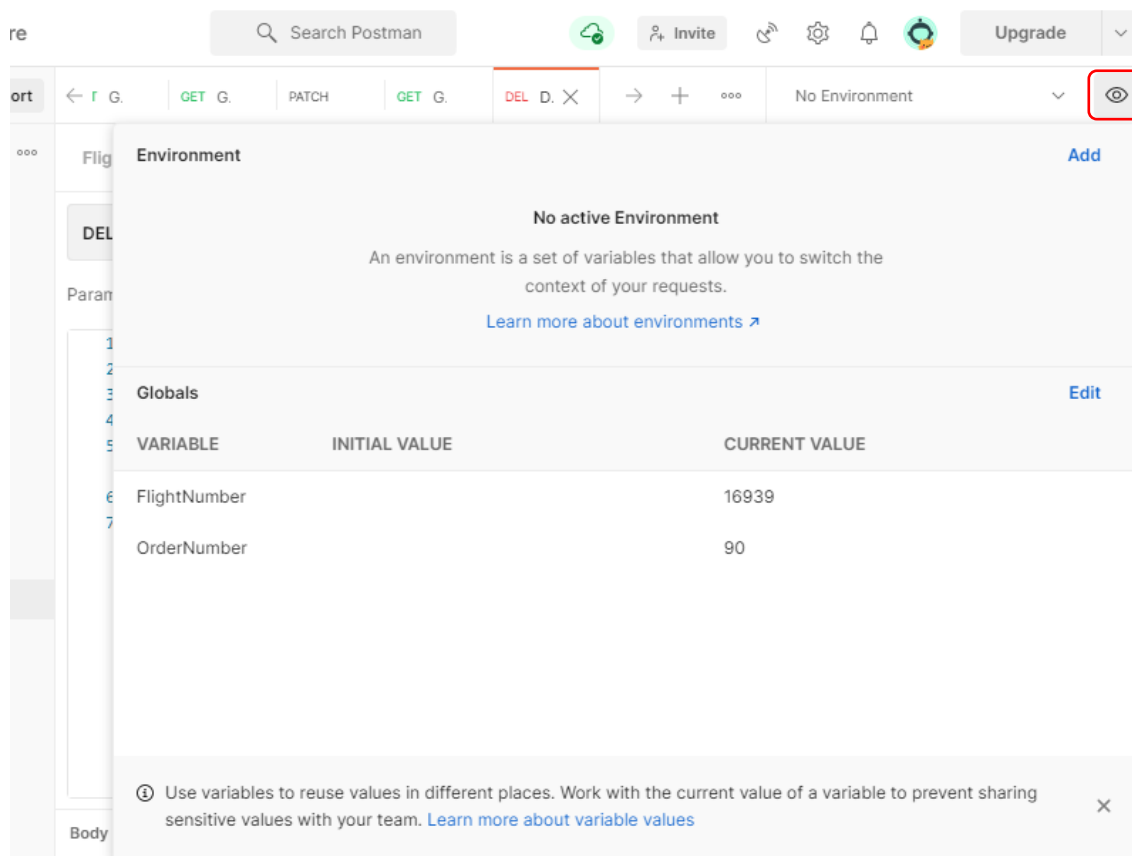
JSON

≡

```
1 {
2   "true": "Order deleted 87 "
3 }
```

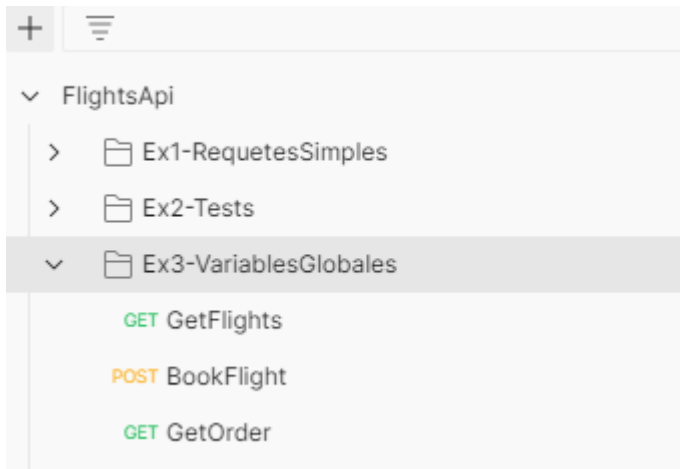
### 7.3. Afficher le contenu des variables globales

Pour afficher le contenu des variables, cliquer sur

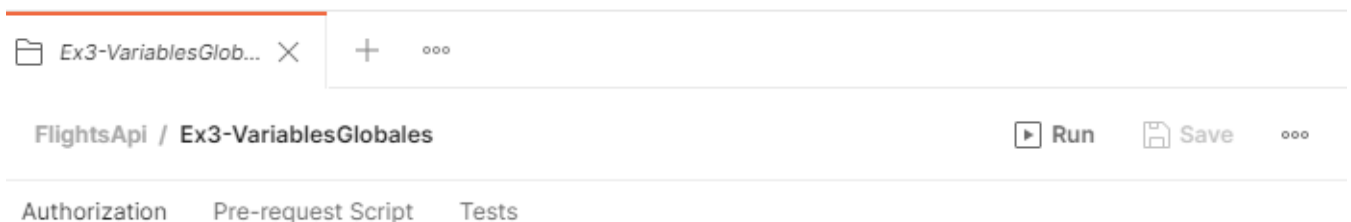


## 7.4. Exécuter l'enchaînement complet des tests

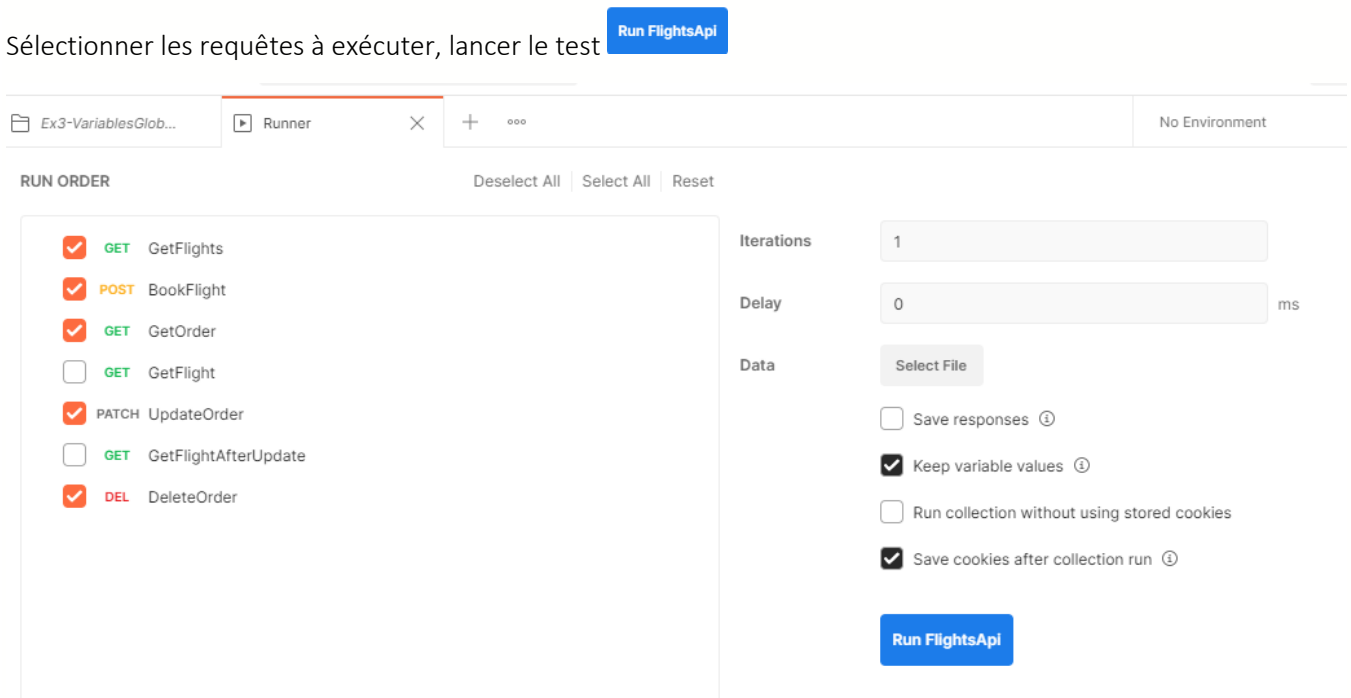
Sélectionner le dossier Ex3-VariablesGlobales



Cliquer sur le bouton Run



Sélectionner les requêtes à exécuter, lancer le test



Ex3-VariablesGlob... FlightsApi No Environment

FlightsApi No Environment, just now View Summary Run Again New Export Rest

All Tests Passed (13) Failed (0)

Iteration 1

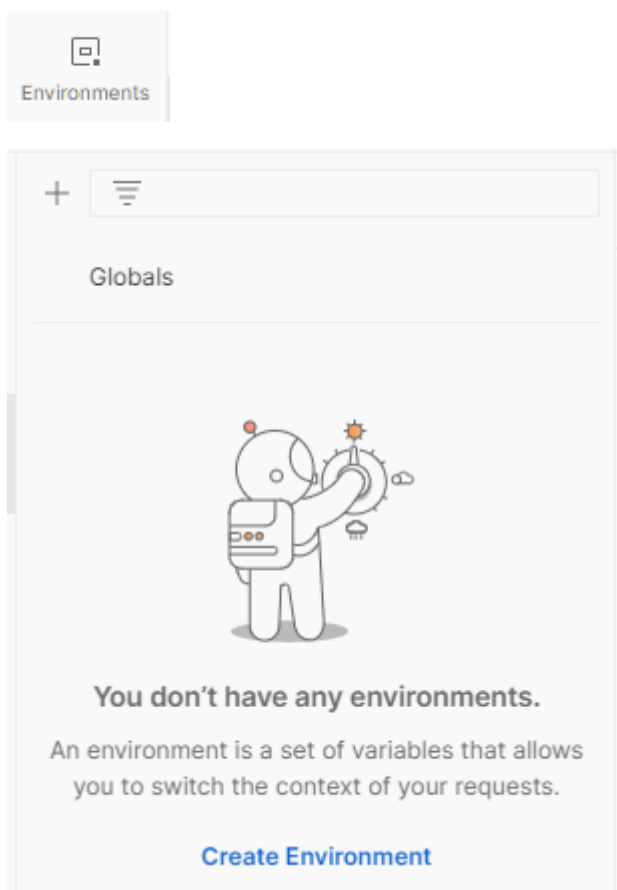
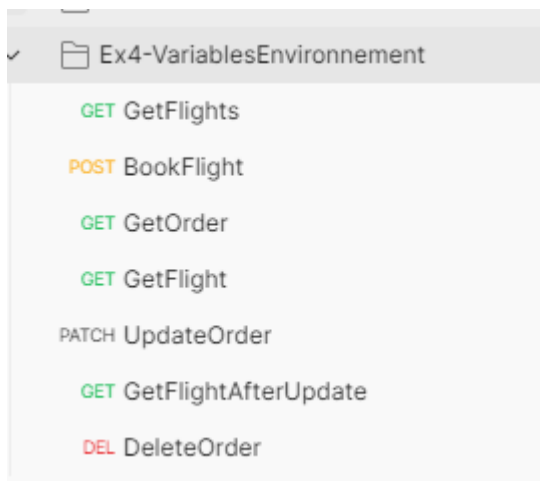
GET	GetFli...	http://localhost:5000/flights_api/v1/resources/Flights?DepartureCity=Paris&ArrivalCity...	/ Ex3-Var...	200 OK	527 ms	1.865 KB
Pass	Le code retour est 200					
Pass	La ville de départ du 1er vol est Paris					
Pass	Le 1er vol est au format nombre					
POST	BookFlight	http://localhost:5000/flights_api/v1/resources/FlightOrders	/ Ex3-VariablesGlobales / BookFlight	200 OK	537 ms	186 B
Pass	Le code retour est 200					
Pass	La balise TotalPrice est présente					
Pass	Le numéro de réservation est supérieur à 80					
GET	GetOrder	http://localhost:5000/flights_api/v1/resources/FlightOrders/{{OrderNumber}}	/ Ex3-VariablesGlobale...	200 OK	510 ms	313 B
Pass	Le code retour est 200					
Pass	Le numéro de vol est 16939					
Pass	la classe est Economy					
PATCH	UpdateOrder	http://localhost:5000/flights_api/v1/resources/FlightOrders/{{OrderNumber}}	/ Ex3-Variables...	200 OK	532 ms	311 B

## 8. Exercice 4 : Variables d'environnement

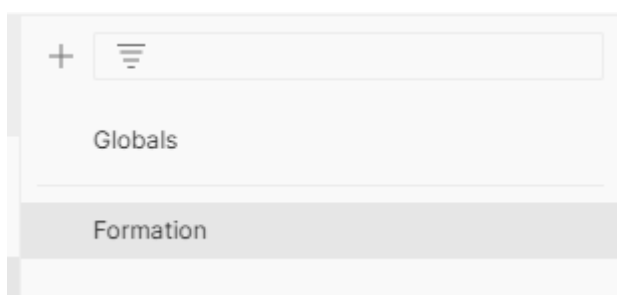
Objectif : Faciliter la maintenance des tests

Dupliquer le dossier *Ex3-VariablesGlobales* en *Ex4-VariablesEnvironnement*

## 8.1. Création des variables d'environnement



Créer un environnement Formation



Créer les variables :

Endpoint / baseURL	<a href="http://localhost:5000/flights_api/v1/resources">http://localhost:5000/flights_api/v1/resources</a>
Date	2021-12-15
Departure	Paris
Arrival	London

FlightAPI

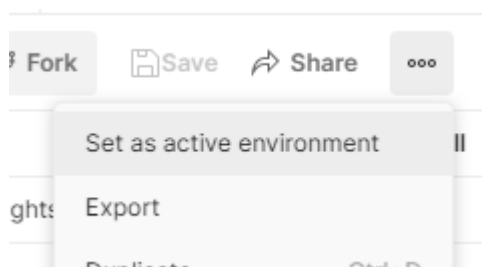
Fork 0 Save Share ...

	VARIABLE	TYPE ⓘ	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	Persist All	Reset
<input checked="" type="checkbox"/>	baseURL	default ▾	http://localhost:5000/flight...	http://localhost:5000/flights_api/v1/r...		
<input checked="" type="checkbox"/>	Date	default ▾	2021-12-15	2021-12-15		
<input checked="" type="checkbox"/>	Departure	default ▾	Paris	Paris		
<input checked="" type="checkbox"/>	Arrival	default ▾	London	London		

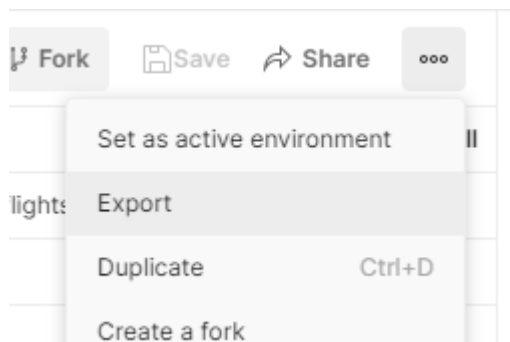
Save

**Attention : pensez à sauvegarder**

Rendre l'environnement actif



Vous pouvez exporter la déclaration des variables dans un fichier json



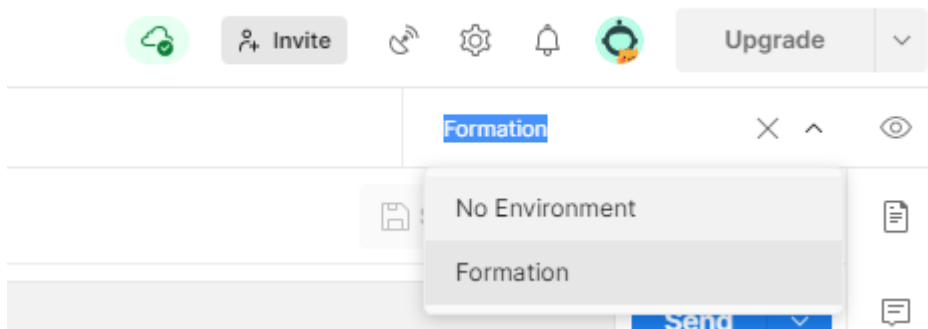
## 8.2. Rendre les requêtes indépendantes de l'adresse des services Rest

Si l'url qui permet d'accéder aux services REST change (par exemple tests sur un environnement d'intégration, puis sur un environnement de recette), il sera nécessaire de modifier toutes les requêtes pour changer l'url.

Pour éviter cela, nous allons utiliser la variable d'environnement endpoint dans toutes les requêtes.

Il suffit de remplacer le texte [http://localhost:5000/flights\\_api/v1/resources](http://localhost:5000/flights_api/v1/resources) par `{{endpoint}}` dans toutes les requêtes

Vérifier que l'environnement est bien sélectionné



Exécuter la requête et sauvegarder. Vérifier que les tests sont toujours OK

FlightsApi / Ex4-VariablesEnvironnement / BookFlight

Save

POST {{endpoint}}/FlightOrders

Send

Params Auth Headers (9) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies Headers (4) Test Results (3/3) 200 OK 536 ms 186 B Save Response

All Passed Skipped Failed

PASS Le code retour est 200

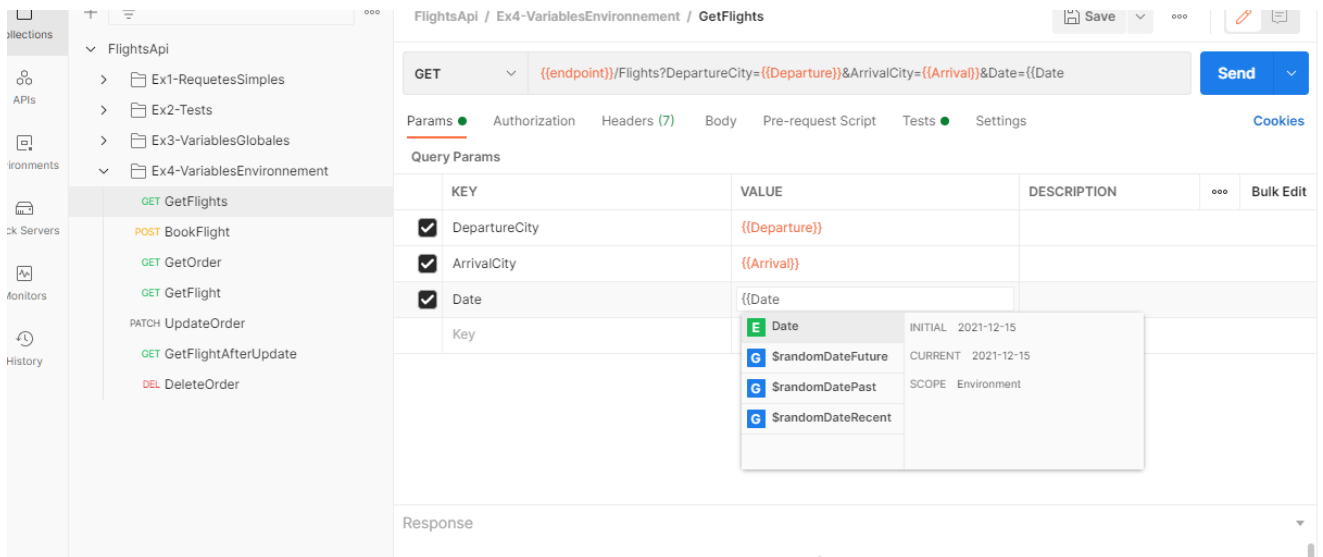
PASS La balise TotalPrice est présente

PASS Le numéro de réservation est supérieur à 80

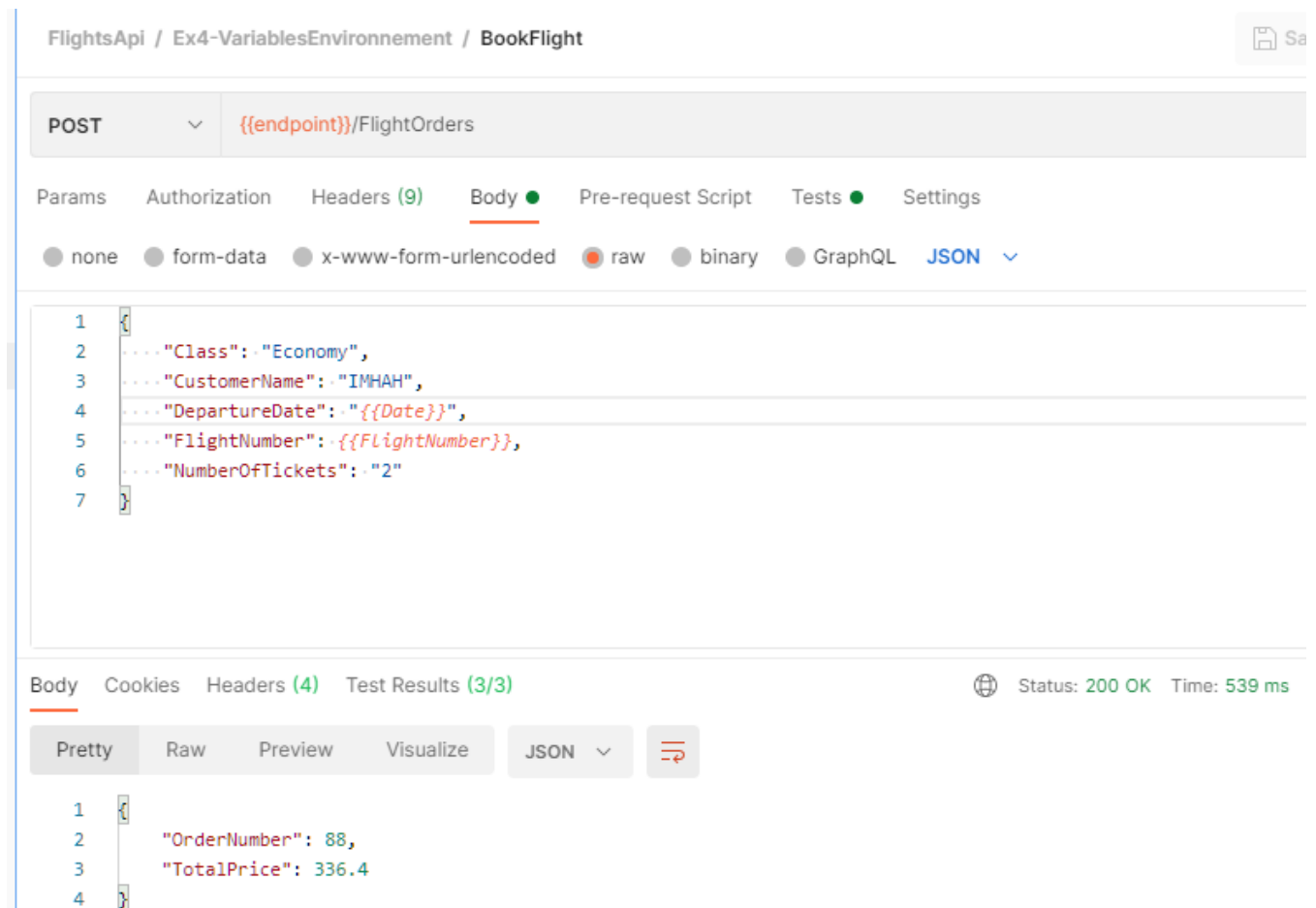
### 8.3. Utiliser les variables ville de départ, ville d'arrivée et date dans GetFlights

En commençant la saisie de la variable Postman propose le nom et affiche le contenu

Remplacer les valeurs par les paramètres {{Departure}} pour DepartureCity, {{Arrival}} pour ArrivalCity et {{Date}} pour Date

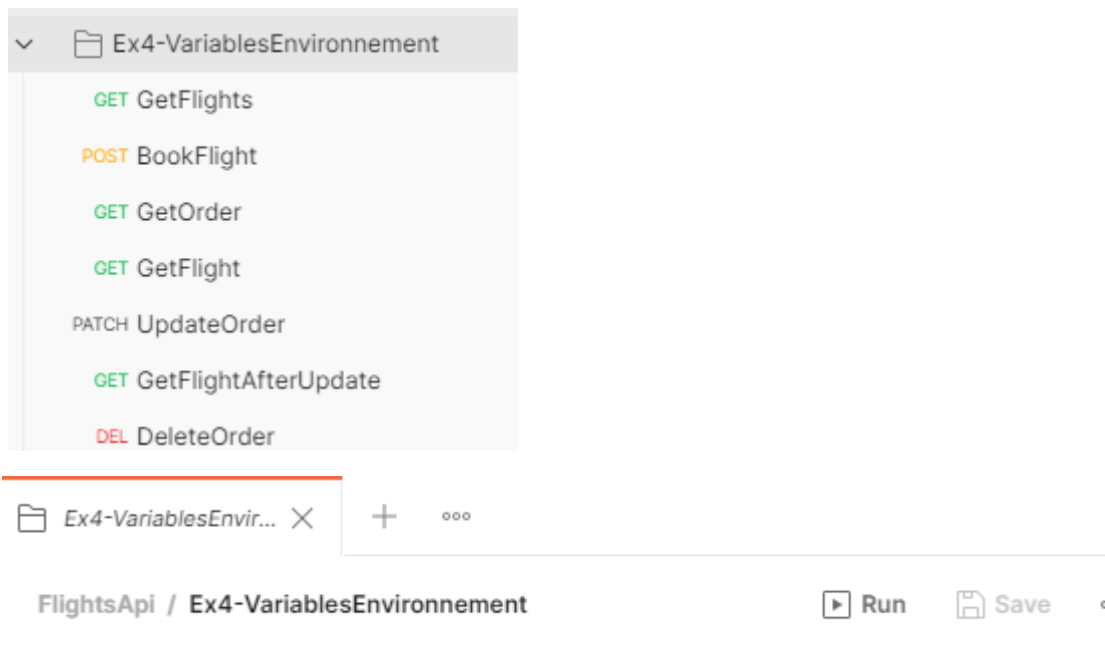


## 8.4. Modifier BookFlight en utilisant la variable Date



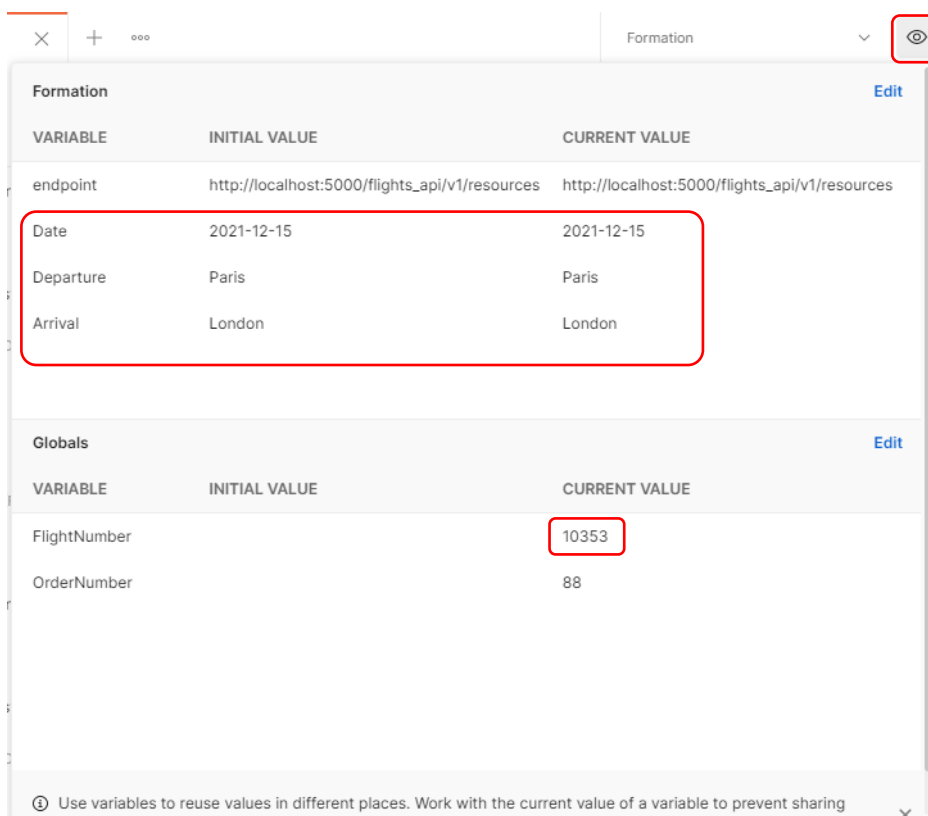
## 8.5. Exécuter la séquence et visualiser les variables

Sélectionner le dossier, et cliquer sur le bouton Run et ensuite sur RunFlightsApi



Run FlightsApi

Visualiser les variables



Pour aller plus loin :

- Calculer la date automatiquement entre +5 et +15 jours
- Sélectionner la ville de départ et la ville d'arrivée aléatoirement dans la liste des villes disponibles : San Francisco, Seattle, Denver, Frankfurt, London, Los Angeles, Paris, Portland, Sydney, Zurich



```

1
2 var max = 5;
3 var min = 15;
4 var randomNumber = Math.floor(Math.random() * (max - min + 1)) + min;
5 var date = pm.environment.get("date");
6
7 pm.environment.set("Date", date + randomNumber);
8
9 var listDepart = ["San Francisco", "Seattle", "Denver", "Frankfurt",
10                  "London", "Los Angeles", "Paris", "Portland", "Sydney", "Zurich"]
11 pm.environment.set("Departure", listDepart[Math.floor(Math.random()
12                  *listDepart.length)]);
13 pm.environment.set("Arrival", listDepart[Math.floor(Math.random()
14                  *listDepart.length)]);

```

- Vérifier que le prix est bien calculé (prix du vol \* nombre de tickets)
- Vérifier que le nombre de sièges disponibles est bien calculé après la réservation et après la mise à jour de la réservation.

## 9. Exercices 5 : Utiliser les données dans un fichier csv et JSON

### Objectif : Etendre la couverture des tests

Postman peut utiliser un fichier CSV ou un fichier Json comme donnée de test.

Le fichier est associé à la séquence de test au moment de l'exécution

Il est possible de choisir un nombre limité de lignes dans le fichier (Iteration)

Les colonnes du fichier seront utilisées comme nom de variables et accessible avec la syntaxe {{variable}}

Avant de commencer dupliquer le dossier *Ex4-VariablesEnvironnement* en *Ex5-FichierDonnees*

Iterations

Delay  ms

Data Select File

☐ Save responses ⓘ

☒ Keep variable values ⓘ

☐ Run collection without using stored cookies

☒ Save cookies after collection run ⓘ

Run FlightsApi

### 9.1. Utiliser les données dans un fichier csv

Créer un fichier csv *flights.csv* dans *C:\Formation\Postman\Data* avec le contenu suivant

```

Departure;Arrival;Date;FlightNumber;Price;SeatsAvailable
Seattle;Portland;11/05/2021;6542;157.4;250
London;San Francisco;20/05/2021;19104;166.8;250
Paris;Denver;13/05/2021;17622;156.6;250

```

Seattle;San Francisco;18/05/2021;5079;173.0;250  
 London;Zurich;13/05/2021;11784;127.8;250  
 Frankfurt;Paris;07/05/2021;13534;159.9;250  
 Paris;Los Angeles;19/05/2021;17090;124.47;250  
 Sydney;London;12/05/2021;12785;164.0;250  
 Seattle;Denver;17/05/2021;1890;243.2;250  
 San Francisco;Paris;09/05/2021;20219;112.2;250  
 San Francisco;Portland;09/05/2021;4135;149.4;250  
 San Francisco;Paris;15/05/2021;20217;112.2;250  
 Los Angeles;Portland;09/05/2021;9039;130.4;250  
 San Francisco;London;17/05/2021;13875;172.47;250  
 Frankfurt;London;06/05/2021;11724;123.0;250  
 Frankfurt;San Francisco;14/05/2021;20036;148.4;250  
 Paris;Sydney;16/05/2021;10573;173.47;250  
 Frankfurt;Zurich;09/05/2021;11245;159.2;250  
 Denver;Los Angeles;07/05/2021;2612;130.8;250  
 Sydney;Paris;11/05/2021;12685;189.2;250


Modifier la requête GetFlights, en supprimant la sauvegarde dans une variable globale de FlightNumber, cela forcera Postman à lire la donnée FlightNumber dans le fichier de données

```
GET {{endpoint}}/Flights?DepartureCity={{Departure}}&ArrivalCity={{Arrival}}&Date={{Date}}
```

Params Authorization Headers (7) Body Pre-request Script Tests Settings

```
1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("La ville de départ du 1er vol est Paris", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData[0].DepartureCity).to.eql("Paris");
7 });
8
9 pm.test("Le 1er vol est au format nombre", function () {
10  var jsonData = pm.response.json();
11  pm.expect(jsonData[0].FlightNumber).to.be.a("number");
12
13  pm.globals.set("FlightNumber", jsonData[0].FlightNumber);
14 });
```





Supprimer la variable déjà enregistrée, cliquer sur  et ensuite sur Edit

Formation <span>Edit</span>		
VARIABLE	INITIAL VALUE	CURRENT VALUE
endpoint	http://localhost:5000/flights_api/v1/resources	http://localhost:5000/flights_api/v1/resources
Date	2021-12-15	2021-12-15
Departure	Paris	Paris
Arrival	London	London

Globals <span>Edit</span>		
VARIABLE	INITIAL VALUE	CURRENT VALUE
OrderNumber		88
FlightNumber		10353

Survoler la ligne FlightNumber et cliquer sur , puis sauvegarder 

Globals <span>Save</span> <span>Export</span>					
Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. <a href="#">Learn more about globals</a>					
	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All Reset All
<input checked="" type="checkbox"/>	OrderNumber		88		
<input checked="" type="checkbox"/>	FlightNumber		10353		X ...
	Add a new variable				

Supprimer les variables d'environnement : Date, Departure et Arrival

Formation <span>Edit</span>		
VARIABLE	INITIAL VALUE	CURRENT VALUE
endpoint	http://localhost:5000/flights_api/v1/resources	http://localhost:5000/flights_api/v1/resources
Date	2021-12-15	2021-12-15
Departure	Paris	Paris
Arrival	London	London

A ce stade les requêtes ne fonctionneront plus sans le fichier csv

## 9.2. Exécuter la séquence de test

FlightsApi / Ex5-FichierDonnees

 Run [

Garder uniquement GetFlights et BookFlight

### RUN ORDER

- ☒ **GET** GetFlights
- ☒ **POST** BookFlight
- ☐ **GET** GetOrder
- ☐ **GET** GetFlight
- ☐ **PATCH** UpdateOrder
- ☐ **GET** GetFlightAfterUpdate
- ☐ **DEL** DeleteOrder

Charger le fichier csv en cliquant sur **Select File**, puis cliquer sur **Preview** pour le visualiser

**Run configuration**

Iterations

Delay  
 ms

Data  
**Select File** flights2.csv ×

Data File Type  
 **Preview**

Advanced settings

PREVIEW DATA				
Iteration	Departure	Arrival	Date	FlightNumber
1	"Seattle"	"Portland"	"2021-05-11"	6542
2	"London"	"San Francisco"	"2021-05-20"	19104
3	"Paris"	"Denver"	"2021-05-13"	17622
4	"Seattle"	"San Francisco"	"2021-05-18"	5079
5	"London"	"Zurich"	"2021-05-13"	11784
6	"Frankfurt"	"Paris"	"2021-05-07"	13534
7	"Paris"	"Los Angeles"	"2021-05-19"	17090
8	"Sydney"	"London"	"2021-05-12"	12785
9	"Seattle"	"Denver"	"2021-05-17"	1890
10	"San Francisco"	"Paris"	"2021-05-09"	20219

Note :

- si Postman n'arrive pas à détecter le fichier csv, changer **Custom Files (\*.json)** en **All Files (\*.\*)** pour faire afficher tous les fichiers.
- Si Postman n'arrive pas à virtualiser les données, essaie de remplacer les « ; » par « , » dans le fichier *flights.csv*

Mettre à jour le nombre d'itération à 3

Iterations

Delay

Data **Select File** Flights.csv ×

Data File Type  **Preview**

**Run FlightsApi**

Lancer le test

All Tests Passed (16) Failed (2)

Iteration 1

GET GetFlights {{endpoint}}/Flights?DepartureCity={{Departure}}&ArrivalCity={{Arrival}}&Date={{Date}} / Ex5-FichierDonnees / GetFlights 20

- Pass Le code retour est 200
- Fail La ville de départ du 1er vol est Paris | AssertionError: expected 'Seattle' to deeply equal 'Paris'
- Pass Le 1er vol est au format nombre

POST BookFlight {{endpoint}}/FlightOrders / Ex5-FichierDonnees / BookFlight

- Pass Le code retour est 200
- Pass La balise TotalPrice est présente
- Pass Le numéro de réservation est supérieur à 80

Iteration 2

Mettre à jour le test sur la ville de départ en remplaçant Paris par `pm.iterationData.get("Departure")`

GET GetFlights X FlightsApi + ...

FlightsApi / Ex5-FichierDonnees / GetFlights

GET {{endpoint}}/Flights?DepartureCity={{Departure}}&ArrivalCity={{Arrival}}&Date={{Date}}

Params Authorization Headers (7) Body Pre-request Script Tests Settings

```

1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("La ville de départ du 1er vol est " + pm.iterationData.get("Departure"), function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData[0].DepartureCity).to.equal(pm.iterationData.get("Departure"));
7 });
8
9 pm.test("Le 1er vol est au format nombre", function () {
10  var jsonData = pm.response.json();
11  pm.expect(jsonData[0].FlightNumber).to.be.a("number");
12 });

```

Run Again

Relancer le test avec

FlightsApi Formation, just now View Summary Run

All Tests Passed (18) Failed (0)

Iteration 1

GET GetFlights {{endpoint}}/Flights?DepartureCity={{Departure}}&ArrivalCity={{Arrival}}&Date={{Date}} / Ex5-FichierDonnees / GetFlights

- Pass Le code retour est 200
- Pass La ville de départ du 1er vol est Seattle
- Pass Le 1er vol est au format nombre

POST BookFlight {{endpoint}}/FlightOrders / Ex5-FichierDonnees / BookFlight

- Pass Le code retour est 200
- Pass La balise TotalPrice est présente
- Pass Le numéro de réservation est supérieur à 80

### 9.3. Utiliser les données dans un fichier json

Créer un fichier *json\_flights.json* dans le même répertoire que celui avant avec le contenu suivant

Le reste de l'exercice est identique au fichier csv

```
[{"Departure":"Seattle","Arrival":"Portland","Date":"2021-05-11","FlightNumber":6542,"Price":157.4,"SeatsAvailable":250},
{"Departure":"London","Arrival":"San Francisco","Date":"2021-05-20","FlightNumber":19104,"Price":166.8,"SeatsAvailable":250},
{"Departure":"Paris","Arrival":"Denver","Date":"2021-05-13","FlightNumber":17622,"Price":156.6,"SeatsAvailable":250},
{"Departure":"Seattle","Arrival":"San Francisco","Date":"2021-05-18","FlightNumber":5079,"Price":173,"SeatsAvailable":250},
{"Departure":"London","Arrival":"Zurich","Date":"2021-05-13","FlightNumber":11784,"Price":127.8,"SeatsAvailable":250},
{"Departure":"Frankfurt","Arrival":"Paris","Date":"2021-05-07","FlightNumber":13534,"Price":159.9,"SeatsAvailable":250},
{"Departure":"Paris","Arrival":"Los Angeles","Date":"2021-05-19","FlightNumber":17090,"Price":124.47,"SeatsAvailable":250},
{"Departure":"Sydney","Arrival":"London","Date":"2021-05-12","FlightNumber":12785,"Price":164,"SeatsAvailable":250},
{"Departure":"Seattle","Arrival":"Denver","Date":"2021-05-17","FlightNumber":1890,"Price":243.2,"SeatsAvailable":250},
{"Departure":"San Francisco","Arrival":"Paris","Date":"2021-05-09","FlightNumber":20219,"Price":112.2,"SeatsAvailable":250},
{"Departure":"San Francisco","Arrival":"Portland","Date":"2021-05-09","FlightNumber":4135,"Price":149.4,"SeatsAvailable":250},
{"Departure":"San Francisco","Arrival":"Paris","Date":"2021-05-15","FlightNumber":20217,"Price":112.2,"SeatsAvailable":250},
{"Departure":"Los Angeles","Arrival":"Portland","Date":"2021-05-09","FlightNumber":9039,"Price":130.4,"SeatsAvailable":250},
{"Departure":"San Francisco","Arrival":"London","Date":"2021-05-17","FlightNumber":13875,"Price":172.47,"SeatsAvailable":250},
{"Departure":"Frankfurt","Arrival":"London","Date":"2021-05-06","FlightNumber":11724,"Price":123,"SeatsAvailable":250},
{"Departure":"Frankfurt","Arrival":"San Francisco","Date":"2021-05-14","FlightNumber":20036,"Price":148.4,"SeatsAvailable":250},
{"Departure":"Paris","Arrival":"Sydney","Date":"2021-05-16","FlightNumber":10573,"Price":173.47,"SeatsAvailable":250},
{"Departure":"Frankfurt","Arrival":"Zurich","Date":"2021-05-09","FlightNumber":11245,"Price":159.2,"SeatsAvailable":250},
{"Departure":"Denver","Arrival":"Los Angeles","Date":"2021-05-07","FlightNumber":2612,"Price":130.8,"SeatsAvailable":250},
{"Departure":"Sydney","Arrival":"Paris","Date":"2021-05-11","FlightNumber":12685,"Price":189.2,"SeatsAvailable":250}]
```

## 10. Exercice 6 : Créer une boucle

**Objectif : Répéter la même séquence de tests**

Pour supprimer toutes les réservations d'un client, il faut lancer plusieurs fois la séquence : GetOrder, DeleteOrder.

Dupliquer le dossier *Ex5-FichierDonnees* en *Ex6-Boucle*, garder uniquement GetOrder et DeleteOrder

Renommer la requête GetOrder en GetOrders et changer la requête

```
GET {{endpoint}} / FlightOrders?CustomerName=IMHAH
```

Cette requête renvoie toutes les réservations du client IMHAH

GET `{{endpoint}}/FlightOrders?CustomerName=IMHAH`

Params ● Authorization Headers (7) Body Pre-request Script Tests ● Settings

Query Params

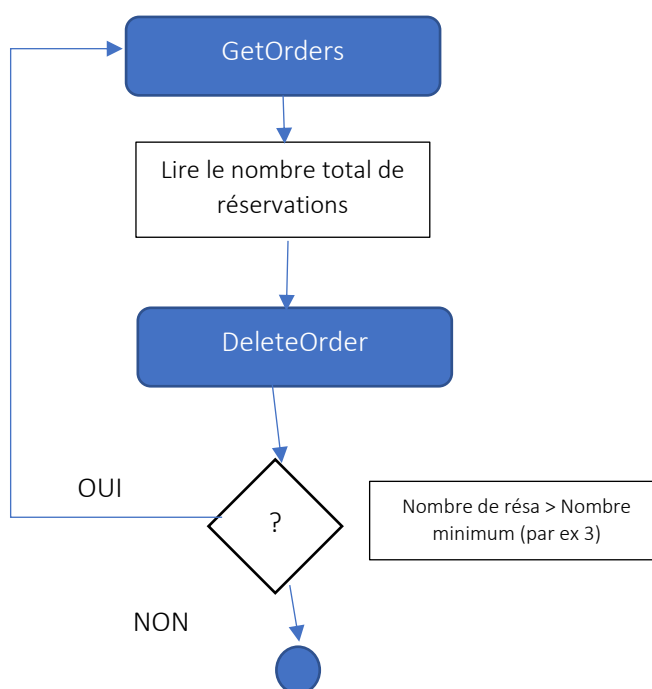
KEY	VALUE
<input checked="" type="checkbox"/> CustomerName	IMHAH
Key	Value

Body Cookies Headers (4) Test Results (2/3)

Pretty Raw Preview Visualize JSON ⌵ ≡

```
1 [
2   {
3     "Class": "Economy",
4     "CustomerName": "IMHAH",
5     "DepartureDate": "2021-5-11 15:35",
6     "FlightNumber": 6542,
7     "NumberOfTickets": 2,
8     "OrderNumber": 89,
9     "TotalPrice": 314.8
10  },
11  {
12    "Class": "Economy",
13    "CustomerName": "IMHAH",
14    "DepartureDate": "2021-5-20 08:00",
15    "FlightNumber": 19104,
16    "NumberOfTickets": 2,
17    "OrderNumber": 90,
18    "TotalPrice": 333.6
19  },
20  {
```

Le mécanisme Postman pour répondre à ce besoin est le suivant



## 10.1. Lire le nombre total de réservation

Modifier les tests de GetOrders en ajoutant un contrôle de la présence de la balise OrderNumber.

Récupérer le nombre total d'éléments dans la réponse json (attribut *length*)

Récupérer le 1<sup>er</sup> numéro de réservation dans la variable globale OrderNumber

FlightsApi / Ex6-Boucle / GetOrders

```
GET {{endpoint}}/FlightOrders?CustomerName=IMHAH

Params Authorization Headers (7) Body Pre-request Script Tests Setting

1 pm.test("Le code retour est 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Le numéro de réservation est supérieur à 80", function () {
6   var jsonData = pm.response.json();
7   pm.expect(jsonData[0].OrderNumber).to.gte(80);
8   pm.globals.set("OrderNumber", jsonData[0].OrderNumber);
9 });
10
11 pm.test("La réponse contient la balise 'OrderNumber' ", function () {
12   pm.expect(pm.response.text()).to.include("\"OrderNumber\"");
13   // Get response in json format
14   var jsonData = pm.response.json();
15   // Get count of json response elements
16   var OrdersCount = jsonData.length;
17   // Save count in a global variable
18   pm.globals.set("OrdersCount", OrdersCount);
19   // Write to console
20   console.log("Orders count is : " + OrdersCount);
21
22 });
```

Vérifier le contenu de la console

```
Find and Replace Console
▶ GET http://localhost:5000/flights_api/v1/resources/FlightOrders?CustomerName=IMHAH
"Orders count is : 70"
```

Ajouter la partie *pre-request* test de *DeleteOrder* l'affichage de ce nombre total de réservation dans la console, utiliser le snippet *Get global variable*

```
Find and Replace Console
"Orders count is : 69"
▶ DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/158
```

## 10.2. Réaliser la boucle

La boucle est réalisée avec la commande `postman.setNextRequest ("GetOrders")`;

Il donne l'instruction d'enchaîner la requête GetOrders après la requête DeleteOrder.

Il faut donc ajouter une condition pour permettre cet enchaînement.

Il est nécessaire d'exécuter la séquence pour que la boucle se réalise



DELETE ▼ `{{endpoint}}/FlightOrders/{{OrderNumber}}`

Params Authorization Headers (7) Body Pre-request Script Tests ● Settings

```
1 var OrdersCount = pm.globals.get("OrdersCount");
2 console.log ("Orders count is : " + OrdersCount);
3
4 if ( OrdersCount > 2 ) {
5     postman.setNextRequest ("GetOrders");
6 }
7
```

Exécuter la séquence

Ex6-Boucle × + ...

FlightsApi / Ex6-Boucle ▶ Run 📄 S

Authorization Pre-request Script Tests

This authorization method will be used for every request in this folder. You can override this by specifying the request.

RUN ORDER

Deselect All | Select All | Reset

☒ GET GetOrders

☒ DEL DeleteOrders

Choose how to run your collection

☒ Run manually  
Run this collection in the Collection Runner.

☐ Schedule runs  
Periodically run collection at a specified time on the Postman Cloud.

☐ Automate runs via CLI  
Configure CLI command to run on your build pipeline.

Run configuration

Iterations

Delay  
 ms

Data

> Advanced settings

Exécuter la séquence Run FlightsApi, et observer la console

```

DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/151
GET http://localhost:5000/flights_api/v1/resources/FlightOrders?CustomerName=IMHAH
"Orders count is : 6"
DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/152
GET http://localhost:5000/flights_api/v1/resources/FlightOrders?CustomerName=IMHAH
"Orders count is : 5"
DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/153
GET http://localhost:5000/flights_api/v1/resources/FlightOrders?CustomerName=IMHAH
"Orders count is : 4"
DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/154
GET http://localhost:5000/flights_api/v1/resources/FlightOrders?CustomerName=IMHAH
"Orders count is : 3"
DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/155
GET http://localhost:5000/flights_api/v1/resources/FlightOrders?CustomerName=IMHAH
"Orders count is : 2"
DELETE http://localhost:5000/flights_api/v1/resources/FlightOrders/156

```

## 2. Compléments

### A, Lancer en ligne de commande

(<https://haquemousume.medium.com/how-to-run-postman-requests-tests-from-command-line-newman-f012afb06564> )

### B, Intégration avec Jenkins

**Jenkins** est un outil open source de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel liées au build, aux tests et au déploiement, et facilite l'intégration continue et la livraison continue.

### C, Virtualisation des APIs (mock server)

**Mock serveur** : une fonctionnalité de Postman qui nous permet de créer un serveur à partir de vos collections. Ce serveur se chargera de simuler l'appel aux requêtes en retournant les réponses statiques pré définies.

**Intérêt** : Utile pour permettre aux équipes front de commencer leurs développements **et les testeurs peuvent commencer à tester les APIs** sans attendre que les développements back soient terminés.

### D, Documentation swagger

**Swagger** est vite la technologie la plus appréciée pour la documentation REST API, le système est capable de représenter presque tous les services Web et informations ayant trait à l'interface

### E, Surveillance APIs

**Postman Monitors** nous fournit une visibilité sur la santé et la performance de nos APIs.


! : Document qui puisse vous aider pour finir ces complémentaires (onepoint - test.it - Complément de tutorial Postman.docx)

## 3. Cas pratique

### 3.1. Gestion de projets Agile avec Taiga.io

Taiga est un outil de gestion de projet pour les équipes agiles multifonctionnelles. Il dispose d'un ensemble riche de fonctionnalités, et il est très simple à utiliser grâce à son interface utilisateur intuitive.


Créer un compte sur le site taiga.io (<https://tree.taiga.io/register>)




# TAIGA


☐ When creating a new account, you agree to our [terms of service](#) and [privacy policy](#).

Or login with

 SIGN IN WITH GITLAB

 SIGN IN WITH GITHUB

Are you already registered? [Log in](#)





# TAIGA

## LOVE YOUR PROJECT

[Forgot it?](#)

Or login with

 SIGN IN WITH GITLAB

 SIGN IN WITH GITHUB

Not registered yet? [create your free account here](#)

### USER SETTINGS



CHANGE PHOTO

Use default image

Username

Email

Full name

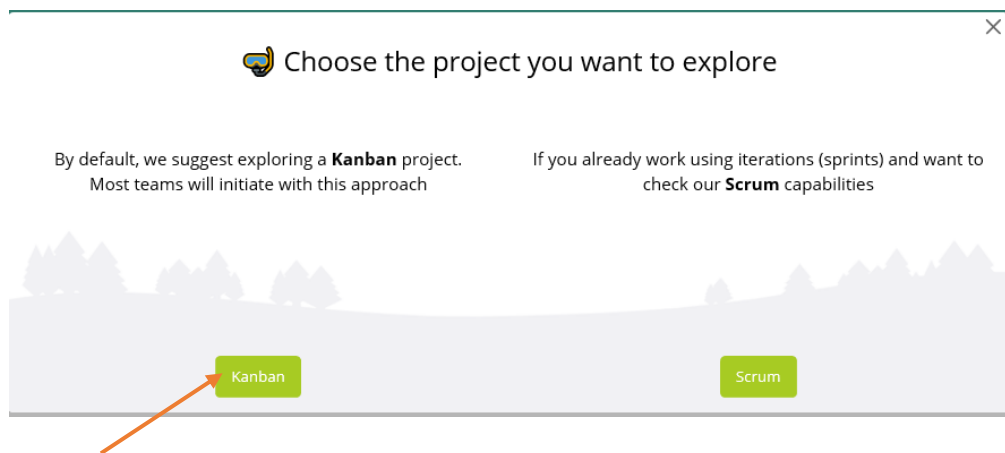
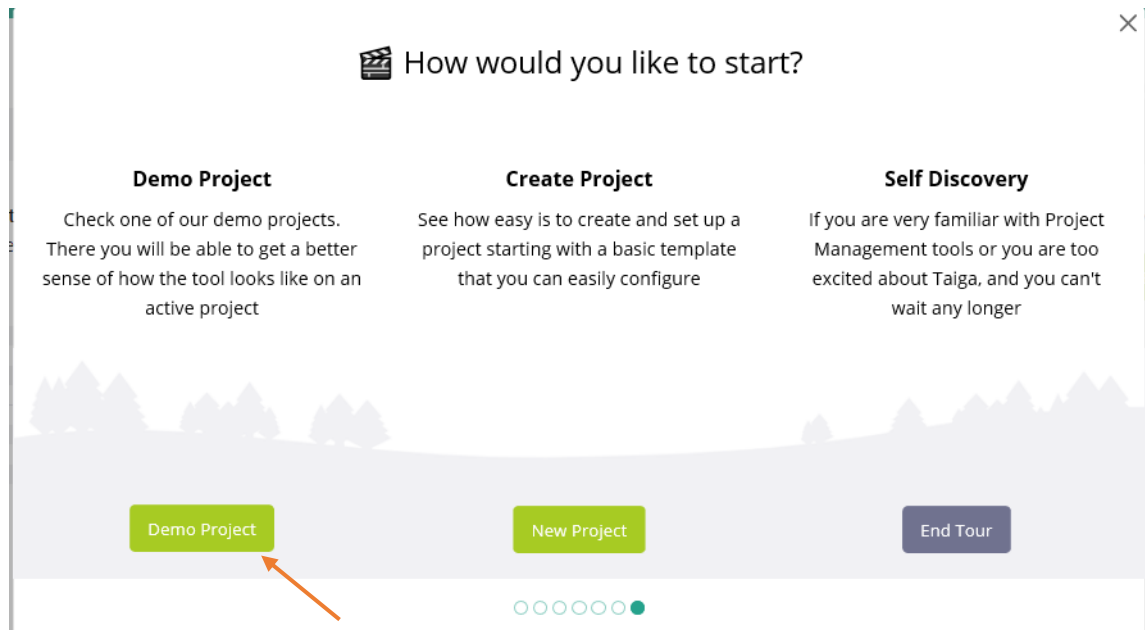
Language

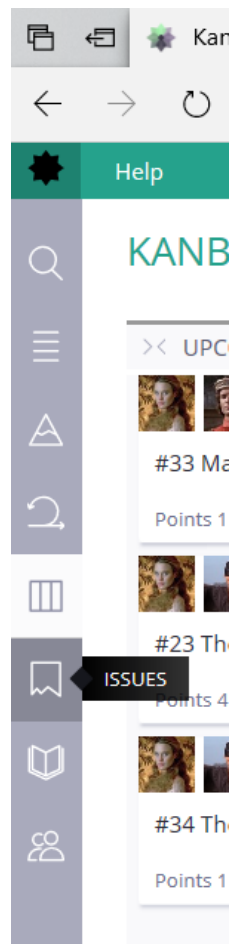
Theme

Bio (max. 210 chars)

[Download Taiga profile](#)

[Delete Taiga account](#)





Help Hassan IMHAH								
FILTERS		ISSUES THE PRINCESS BRIDE						
<input type="text" value="Subject or reference"/>		Type	Severity	Priority	Votes	Subject	Status	Modified
Advanced >								Assigned to
TYPE >					▲ 1	#53 Rodents of unusual size (R.O.U.S.)	In progress	09 Oct 2019
SEVERITY >					▲ 0	#52 Lightning sand	In progress	09 Oct 2019
PRIORITIES >					▲ 0	#51 Flame spurts	In progress	20 Jul 2020
STATUS >					▲ 0	#50 Life is pain	New	09 Oct 2019
TAGS >					▲ 0	#49 Not a lot of money in revenge	New	17 Aug 2020
ASSIGNED TO >					▲ 0	#48 He didn't fall!	In progress	09 Oct 2019
ROLE >					▲ 0	#47 Unemployed in Greenland	Closed	09 Oct 2019
CREATED BY >					▲ 0	#46 Friendless, brainless, helpless, ho...	Ready to test	09 Oct 2019
					▲ 0	#45 Princess Buttercup does not love ...	Needs info	09 Oct 2019
					▲ 0	#44 Not enough money	Closed	09 Oct 2019
					▲ 0	#43 Where's the sports in the book?	Ready to test	09 Oct 2019
					▲ 0	#42 The boy is sick in bed	In progress	09 Oct 2019
					▲ 0	#10 Dropping the sword when left-han...	Closed	09 Oct 2019
					▲ 3	#9 Excessive exposure to localine	Postponed	09 Oct 2019

## 3.2. Créer un projet démo

[https://tree.taiga.io/project/himhah-demo\\_api/issues](https://tree.taiga.io/project/himhah-demo_api/issues)



You don't have any project yet

CREATE PROJECT

## CREATE PROJECT

Which template fits your project better?



### SCRUM

Prioritize and solve your tasks in short time cycles.



### KANBAN

Keep a constant workflow on independent tasks



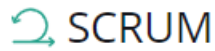
### DUPLICATE PROJECT

Start clean and keep your configuration



### IMPORT PROJECT

Import your project from multiple platforms into Taiga




Prioritize and solve your tasks in short time cycles.

#### New project details

Demo\_SoapUI

Demo pour la formation SoapUI

 PUBLIC PROJECT

 PRIVATE PROJECT

BACK

CREATE PROJECT

Accès direct aux anomalies [https://tree.taiga.io/project/hassanimhah-demo\\_soapui/issues](https://tree.taiga.io/project/hassanimhah-demo_soapui/issues)

### 3.3. L'API rest

Taiga dispose d'une API REST complète (celle utilisée par l'application Web)

<https://docs.taiga.io/api.html>

Nous allons principalement utiliser les APIs suivantes :

Connexion	<a href="https://taigaio.github.io/taiga-doc/dist/api.html#auth-normal-login">https://taigaio.github.io/taiga-doc/dist/api.html#auth-normal-login</a>
Projets	<a href="https://taigaio.github.io/taiga-doc/dist/api.html#projects-list">https://taigaio.github.io/taiga-doc/dist/api.html#projects-list</a>
Anomalies (Issues)	<a href="https://taigaio.github.io/taiga-doc/dist/api.html#issues-list">https://taigaio.github.io/taiga-doc/dist/api.html#issues-list</a>

### 3.4. Travail à faire

1. Se connecter à l'application
2. Lire les informations du projet à partir de son nom
3. Récupérer la liste des anomalies du projet
4. Créer une anomalie et vérifier sur le site web que l'anomalie est bien créée
5. Afficher les informations de l'anomalie créée
6. Modifier le statut de l'anomalie
7. Créer une liste d'anomalie à partir d'un fichier csv