

**Naam:** Samy Sah

**R-nummer:** R0798534

**Datum:** 20/09/2021

# OPDRACHT – POSTMAN TESTING

Opleidingsonderdeel: Software Testing

**Docent:** Serneels Frank

jaar: 2021-2022

# INHOUDSOPGAVE

<b>1</b>	<b>RESEARCH .....</b>	<b>3</b>
1.1	BENODIGDHEDEN .....	3
<b>2</b>	<b>LOGBOEK .....</b>	<b>4</b>
2.1	WEEK 1 – 20/09/2021: 4 UUR .....	4
2.2	WEEK 2 – 27/09/2021: 3 UUR .....	4
2.3	WEEK 3 – 04/10/2021: 3 UUR .....	5
2.3.1	<i>Eerste request met Postman: GET request .....</i>	<i>5</i>
2.3.2	<i>Collection en variable .....</i>	<i>6</i>
2.3.3	<i>Query parameters.....</i>	<i>7</i>
2.3.4	<i>Path variables.....</i>	<i>9</i>
2.3.5	<i>POST request en API authentication .....</i>	<i>10</i>
2.4	WEEK 3 – 05/10/2021: 3 UUR .....	15
2.4.1	<i>Testen: Random test data .....</i>	<i>15</i>
2.4.2	<i>PATCH request: bestelling bewerken .....</i>	<i>16</i>
2.4.3	<i>DELETE request: bestelling verwijderen.....</i>	<i>17</i>
2.4.4	<i>Writing API test.....</i>	<i>18</i>
<b>3</b>	<b>BRON VERMELDING .....</b>	<b>21</b>
3.1	VIDEO .....	21
3.2	GRATIS READ ONLY .....	21
3.3	UDEMY CURSUS.....	21

# 1 Research

Postman wordt gebruikt voor het testen van API's. Dit is een HTTP-client die HTTP-verzoeken test, waarbij gebruik wordt gemaakt van een grafische interface, waarmee we verschillende typen responsen ontvangen die nadien gevalideerd moeten worden.

GET: Het verkrijgen van informatie

POST: Toevoegen van informatie

PUT: Informatie vervangen

PATCH: Bepaalde informatie bijwerken

DELETE: informatie verwijderen

- Toegankelijkheid: om postman te gebruiken hoeft je alleen in te loggen.
- Samenwerking: Collecties en omgevingen kunnen worden geïmporteerd of geëxporteerd, waardoor het gemakkelijk is om bestanden te delen. Een directe link kan ook worden gebruikt om collecties te delen.
- Omgevingen creëren: Het hebben van meerdere omgevingen helpt bij minder herhaling van tests.
- Geautomatiseerd testen: Door het gebruik van de Collection Runner of Newman kunnen tests in meerdere iteraties worden uitgevoerd, waardoor tijd wordt bespaard voor repetitieve tests.
- Debugging: De Postman console helpt om te controleren welke gegevens zijn opgehaald, waardoor het gemakkelijk is om tests te debuggen.

## 1.1 benodigdheden

1. POSTMAN
2. API
3. Cursus
4. Video's

## 2 Logboek

### 2.1 Week 1 – 20/09/2021: 4 uur

- Les gevolgd: 4 uur: Tijdens week één kregen we vooral de inleiding om te zien wat er van ons verwacht werd.

### 2.2 Week 2 – 27/09/2021: 3 uur

- Research over Postman: 30 min
- Gesprek met leerkracht: 10 min
- begonnen aan cursus "API Testing and Development with Postman " en Udemy "Introduction to POSTMAN - A Beginners guide": 2 uur 20 min

★ Course Contents ★

★ Unit 1 - Introduction to Postman

- Lesson 1 - Welcome (0:00:00)
- Lesson 2 - What is Postman (0:01:12)
- Lesson 3 - How to install Postman (0:03:06)
- Lesson 4 - Your first request with Postman (0:04:45)
- Lesson 5 - HTTP (0:07:07)
- Lesson 6 - Postman collections and variables (0:11:10)
- Lesson 7 - Query parameters (0:15:55)
- Lesson 8 - Assignment (0:22:50)
- Lesson 9 - Path variables (0:25:21)
- Lesson 10 - POST request / API Authentication (0:30:07)
- Lesson 11 - JSON format (0:41:21)
- Lesson 12 - Assignment (0:45:11)
- Lesson 13 - Random test data (0:47:32)
- Lesson 14 - Is Postman the right tool for me? (0:50:59)
- Lesson 15 - Viewing existing orders (0:52:16)
- Lesson 16 - Assignment (0:53:59)
- Lesson 17 - PATCH request (0:55:56)
- Lesson 18 - DELETE request (0:59:03)

Contenu du cours

**Section 1 : Introduction**  
2 / 2 | 3 min

- ✓ 1. Introduction to Postman  
3 min
- ✓ 2. \*\*\*\*RATING THE COURSE\*\*\*\*  
1 min

**Section 2 : Installing Postman**  
1 / 1 | 3 min

- ✓ 3. Installing the Postman Standalone App(Windows & macOS)  
3 min

**Section 3 : Launching Student App with Docker**  
3 / 3 | 15 min

- ✓ 4. Installing Docker on Windows OS  
6 min
- ✓ 5. Installing Docker on MacOSx  
4 min
- ✓ 6. Launching Student app docker image  
5 min

**Section 4 : CRUD Operations with POSTMAN**  
4 / 4 | 17 min

- ✓ 7. GET Request(query parameters, path parameters)  
8 min
- ✓ 8. Create a new Student (POST method)  
4 min
- ✓ 9. Update student info(PUT request)  
3 min
- ✓ 10. Delete a student (DELETE method)  
2 min

**Section 5 : Postman Collections**  
4 / 4 | 34 min

- ✓ 11. Setup BestBuy API Playground  
7 min
- ✓ 12. Global,Environment & Collection Variables  
13 min
- ✓ 13. Creating collections in Postman  
8 min
- ✓ 14. Collection Runner in Postman  
6 min

## 2.3 Week 3 – 04/10/2021: 3 uur

- Verder gewerkt aan cursus "Introduction to POSTMAN - A Beginners guide" : 20 m
- Verder gewerkt aan cursus "API Testing and Development with Postman " : 1 uur
- Eerste request met Postman: 1 uur 40 min

**Section 6 : Authentication in Postman**  
1 / 1 | 4 min

☒ 15. Basic Authentication in Postman  
4 min

**Section 7 : Tests, Assertions in Postman**  
1 / 1 | 10 min

☒ 16. Writing Tests, Adding assertions!!  
10 min

**Section 8 : Next Steps in your Journey!!**  
1 / 1 | 1 min

☒ 17. Next Steps!!  
1 min

★ Unit 2 - Test automation with Postman

- Lesson 19 - Introduction to test automation (1:01:52)
- Lesson 20 - Your first API tests (1:02:52)
- Lesson 21 - Assignment (1:14:55)
- Lesson 22 - Postman variables (1:19:20)
- Lesson 23 - Extracting data from the response (1:24:13)
- Lesson 24 - Assignment (1:36:51)
- Lesson 25 - Assignment (1:38:08)
- Lesson 26 - Collection runner (1:42:52)
- Lesson 27 - Request execution order (1:49:00)
- Lesson 28 - Postman monitors (1:53:32)
- Lesson 29 - Newman (1:57:45)
- Lesson 30 - HTML reports with Newman (2:01:58)
- Lesson 30 - CI/CD overview (2:05:28)
- Lesson 31 - Conclusion (2:08:24)

### 2.3.1 Eerste request met Postman: GET request

#### API documentation:

<https://github.com/vdespa/introduction-to-postman-course/blob/main/simple-books-api.md>

API: <https://simple-books-api.glitch.me>

baseUrl: <https://simple-books-api.glitch.me>

Dit is mijn eerste request met de api namelijk een GET request met een status endpoint. De status geeft ok weer dit betekent dat de api werkt en dat we die kunnen gebruiken.

GET https://simple-boo... + ...

https://simple-books-api.glitch.me/status

GET https://simple-books-api.glitch.me/status

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

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON

```

1
2
3
{"status": "OK"}

```

**Link:** `{{baseUrl}}`/books (<https://simple-books-api.glitch.me/books>)

## Read:

Simple Book API / List of books

GET `{{baseUrl}}/books` Send

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (6) Test Results Status: 200 OK Time: 167 ms Size: 631 B Save Response

Pretty Raw Preview Visualize JSON Copy

```

1 [
2   {
3     "id": 1,
4     "name": "The Russian",
5     "type": "fiction",
6     "available": true
7   },
8   {
9     "id": 2,
10    "name": "Just as I Am",
11    "type": "non-fiction",
12    "available": false
13  },
14  {
15    "id": 3,
16    "name": "The Vanishing Half",
17    "type": "fiction",
18    "available": true
19  }
20 ]

```

## 2.3.2 Collection en variable

Hier heb ik van de API een collection gemaakt

My Workspace New Import

Collections + ...

Simple Book API

GET API Status

APIs

Vervolgens heb ik een variable aangemaakt met de naam baseUrl.

**Initial value:** zal gedeeld worden met andere mensen. Bv: als je jouw collection deelt met een vriend dan zal hij de mogelijkheid hebben om de initial value te zien.

**Current value:** Is wat er gebruikt wordt in postman en is privé

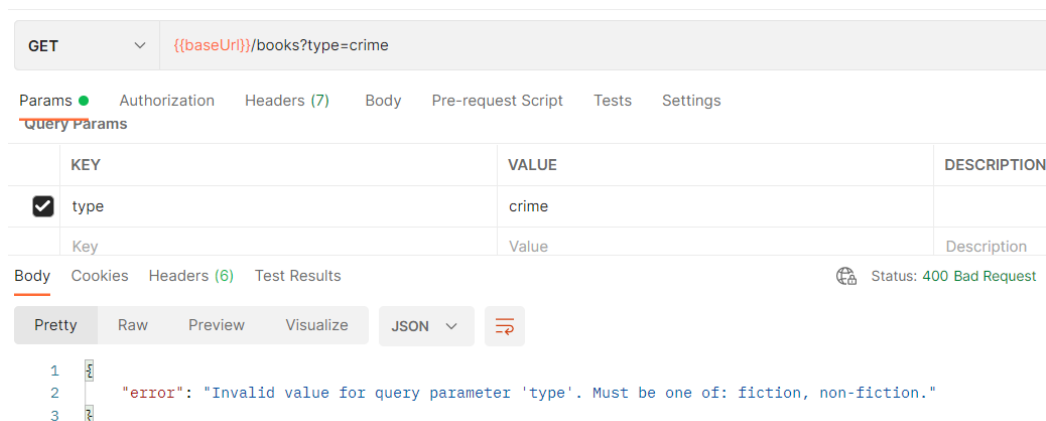
	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...
<input checked="" type="checkbox"/>	baseUrl	https://simple-books-a...	https://simple-books-api.glitch.me	
	Add a new variable			

### 2.3.3 Query parameters

Vervolgens heb ik gebruikt gemaakt van query's om te kunnen filteren om type maar zoals u kunt zien geeft postman een request 400 en postman geeft ons dan ook wat meer uitleg over de fout. De key dat ik heb meegegeven is type en value crime maar deze bestaat namelijk niet.

**Link:** `{{baseUrl}}/books?type=crime`

(<https://simple-books-api.glitch.me/books?type=crime>)



GET `{{baseUrl}}/books?type=crime`

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

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	type	crime	
	Key	Value	Description

Body Cookies Headers (6) Test Results Status: 400 Bad Request

Pretty Raw Preview Visualize JSON ⌵ ⌵

```

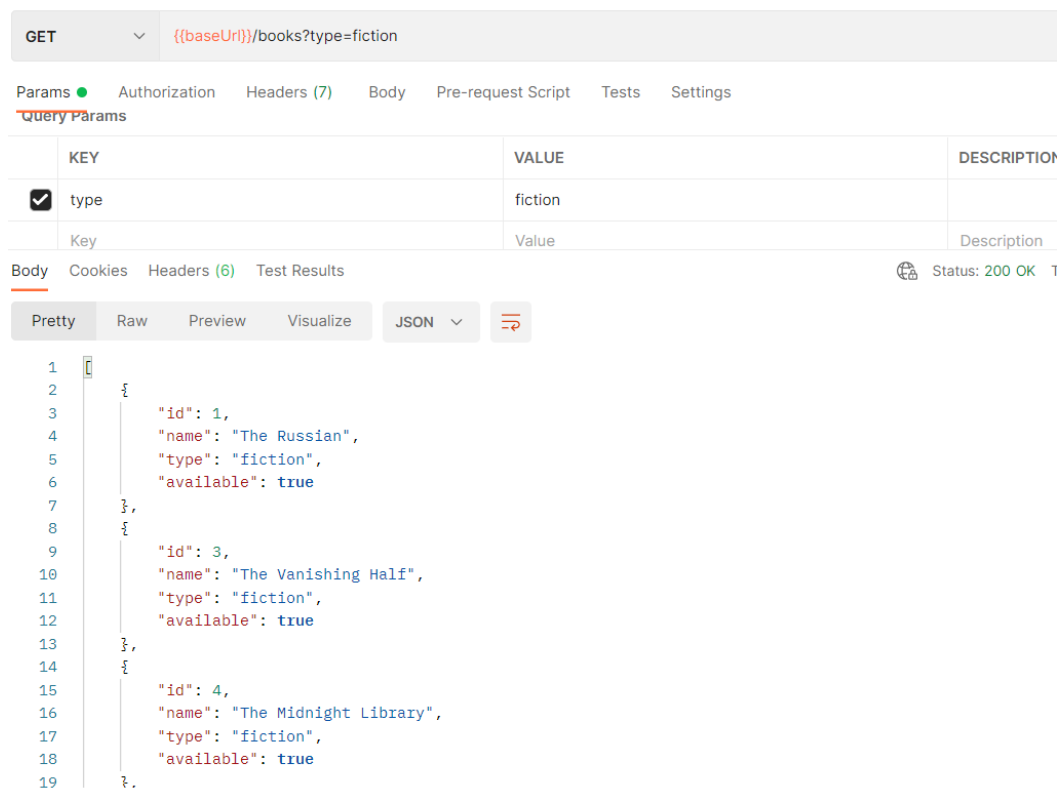
1
2
3
{"error": "Invalid value for query parameter 'type'. Must be one of: fiction, non-fiction."}

```

Maar als we de value veranderen naar fiction krijgen we weer een Json output.

**Link:** `{{baseUrl}}/books?type= fiction`

(<https://simple-books-api.glitch.me/books?type=fiction>)



GET `{{baseUrl}}/books?type=fiction`

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

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	type	fiction	
	Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON ⌵ ⌵

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
[
  {
    "id": 1,
    "name": "The Russian",
    "type": "fiction",
    "available": true
  },
  {
    "id": 3,
    "name": "The Vanishing Half",
    "type": "fiction",
    "available": true
  },
  {
    "id": 4,
    "name": "The Midnight Library",
    "type": "fiction",
    "available": true
  }
]

```

Ik het geval van deze API heb ik geprobeerd om een limit aantal mee te geven

De API geeft 2 outputs

**Link:** `{{baseUrl}}/books?type= fiction&limit=2`

(<https://simple-books-api.glitch.me/books?type= fiction&limit=2>)

GET `{{baseUrl}}/books?type=fiction&limit=2`

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

Key	Value	Description
<input checked="" type="checkbox"/> type	fiction	
<input checked="" type="checkbox"/> limit	2	

Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1  [
2    {
3      "id": 1,
4      "name": "The Russian",
5      "type": "fiction",
6      "available": true
7    },
8    {
9      "id": 3,
10     "name": "The Vanishing Half",
11     "type": "fiction",
12     "available": true
13   }
14 ]
  
```

maar in het geval dat je Limit met een hoofdletter schrijft wordt er niks gedaan.

**Link:** `{{baseUrl}}/books?type= fiction&Limit=2`

(<https://simple-books-api.glitch.me/books?type= fiction&Limit=2>)

GET `{{baseUrl}}/books?type=fiction&Limit=2`

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

Key	Value	Description
<input checked="" type="checkbox"/> type	fiction	
<input checked="" type="checkbox"/> Limit	2	

Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1  [
2    {
3      "id": 1,
4      "name": "The Russian",
5      "type": "fiction",
6      "available": true
7    },
8    {
9      "id": 3,
10     "name": "The Vanishing Half",
11     "type": "fiction",
12     "available": true
13   },
14   {
15     "id": 4,
16     "name": "The Midnight Library",
17     "type": "fiction",
18     "available": true
19   }
20 ]
  
```



### 2.3.4 Path variables

**Link:** `{{baseUrl}}/books/:bookId`

**Key :** bookId

**Value :** 2

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `{{baseUrl}}/books/:bookId`
- Params:** A table with 3 columns: KEY, VALUE, and DESCRIPTION. It contains one entry: Key: Value, Description.
- Path Variables:** A table with 3 columns: KEY, VALUE, and DESCRIPTION. It contains one entry: bookId: 2, Description.
- Body:** A JSON object: `{ "id": 2, "name": "Just as I Am", "author": "Cicely Tyson", "type": "non-fiction", "price": 20.33, "current-stock": 0, "available": false }`
- Status:** 200 OK

Als we opzoek gaan naar een boek met een id dat niet bestaat krijgen we deze bericht te zien namelijk een 404 not found

**Link:** `{{baseUrl}}/books/:bookId`

**Key :** bookId

**Value :** 200

The screenshot shows a REST client interface with the following details:

- Path Variables:** A table with 3 columns: KEY, VALUE, and DESCRIPTION. It contains one entry: bookId: 200, Description.
- Status:** 404 Not Found
- Body:** A JSON object: `{ "error": "No book with id 200" }`

De path variables kunnen we namelijk ook zo schrijven

**Link:** `{{baseUrl}}/books/2`


Simple Book API / Get single book


GET `{{baseUrl}}/books/2`

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results  Status: 200 OK

Pretty Raw Preview Visualize JSON 

```


1 {
2   "id": 2,
3   "name": "Just as I Am",
4   "author": "Cicely Tyson",
5   "type": "non-fiction",
6   "price": 20.33,
7   "current-stock": 0,
8   "available": false
9 }
```

### 2.3.5 POST request en API authentication


Hier proberen we een order te plaatsen maar we hebben een authentication in vergeleken met een GET request waar geen authentication nodig is.


POST `{{baseUrl}}/orders`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers  8 hidden

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results  Status: 401 Unauthorized

Pretty Raw Preview Visualize JSON 

```

1 {
2   "error": "Missing Authorization header."
3 }
```

Om een bestelling in te dienen of te bekijken, moeten we namelijk een API client registreren.

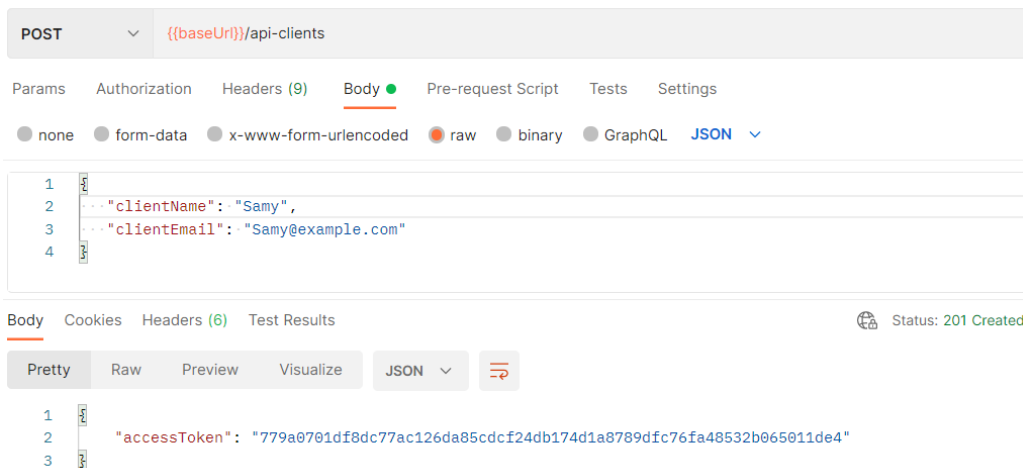
**Link:** `{{baseUrl}}/api-clients`

POST `/api-clients/`

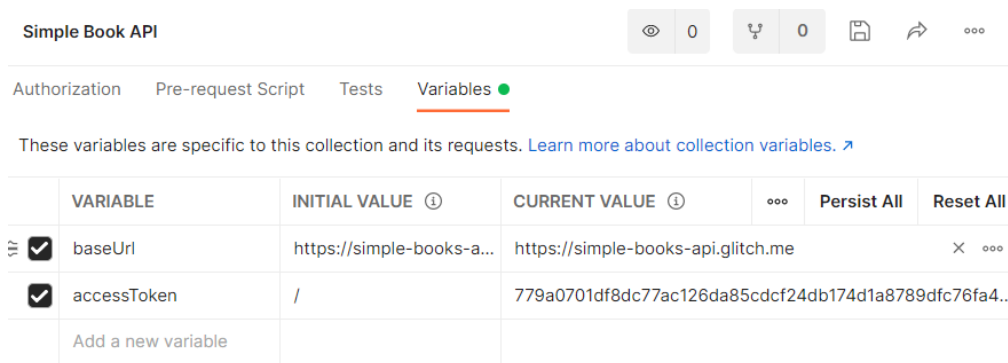
De request body moet in JSON formaat zijn en de volgende eigenschappen bevatten:

- `clientName` – String
- `clientEmail` – String

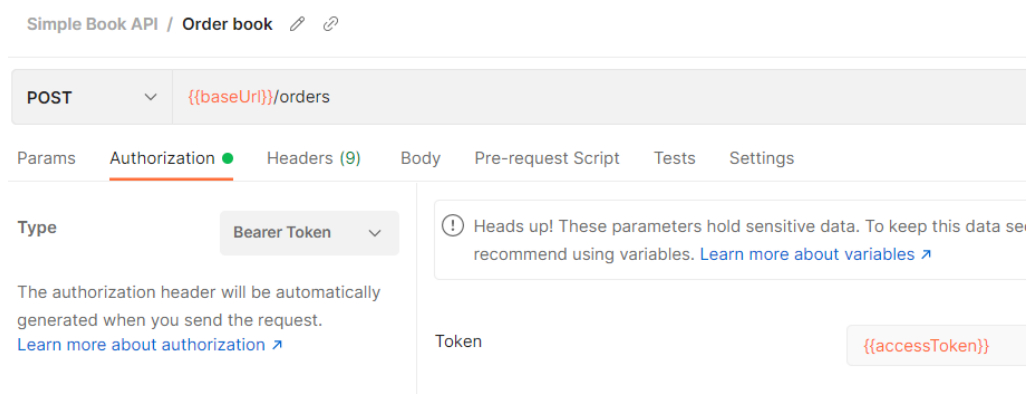
In de response body zullen we de access token terugvinden.



Ik zal dan de access token toevoegen aan mijn variable met een lege initial value



Vervolgens heb ik de access token namelijk de variable gebruikt in de Bearer token in de Authorization helper



Postman heeft automatisch een nieuwe header aangemaakt.

Simple Book API / Order book

POST ▼ `{{baseUrl}}/orders`

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

Headers 🔍 Hide auto-generated headers

KEY	VALUE
<input checked="" type="checkbox"/> Authorization ⓘ	Bearer 779a0701df8dc77ac126da85cdf24db174d1a878...
<input checked="" type="checkbox"/> Cache-Control ⓘ	no-cache
<input checked="" type="checkbox"/> Postman-Token ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> Content-Length ⓘ	0
<input checked="" type="checkbox"/> Host ⓘ	<calculated when request is sent>
<input checked="" type="checkbox"/> User-Agent ⓘ	PostmanRuntime/7.28.4
<input checked="" type="checkbox"/> Accept ⓘ	*/*
<input checked="" type="checkbox"/> Accept-Encoding ⓘ	gzip, deflate, br
<input checked="" type="checkbox"/> Connection ⓘ	keep-alive

**Link:** `{{baseUrl}}/orders`

Een bestelling doorgeven

POST/orders

Stelt u in staat een nieuwe bestelling in te dienen. Vereist authenticatie.

De request body moet in JSON formaat zijn en de volgende eigenschappen bevatten:

bookId - Integer - Verplicht

customerName - String - Verplicht

POST ▼ `{{baseUrl}}/orders`

Params Authorization ● Headers (10) **Body** ● Pre-request Script Tests Settings

☐ none 
 ☐ form-data 
 ☐ x-www-form-urlencoded 
 ☒ raw 
 ☐ binary 
 ☐ GraphQL 
 **JSON** ▼

```

1  {
2    "bookId": 1,
3    "customerName": "John"
4  }

```

Body Cookies Headers (6) Test Results 🌐 Status: 201 Created

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```

1  {
2    "created": true,
3    "orderId": "qJ9TRenu2HVC-msgQV2jV"
4  }

```

Als we een order maken van een boek dat niet meer in stock is zal deze bericht op onze scherm verschijnen.

POST `{{baseUrl}}/orders`

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1 {
2   "bookId": 2,
3   "customerName": "John"
4 }

```

Body Cookies Headers (6) Test Results Status: 404 Not Found

Pretty Raw Preview Visualize **JSON**

```

1 {
2   "error": "This book is not in stock. Try again later."
3 }

```

Maar hoe weten we welke boek ik stock is en welke niet voor dit gaan we eerst naar de list of books kijken. Zoals we hier kunnen zien is boek met id 2 niet meer available

**Link:** `{{baseUrl}}/books`

GET `{{baseUrl}}/books`

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

<input type="checkbox"/>	type	fiction
<input type="checkbox"/>	Limit	2
	Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize **JSON**

```

1 {
2   {
3     "id": 1,
4     "name": "The Russian",
5     "type": "fiction",
6     "available": true
7   },
8   {
9     "id": 2,
10    "name": "Just as I Am",
11    "type": "non-fiction",
12    "available": false
13  },
14  {
15    "id": 3,
16    "name": "The Vanishing Half",
17    "type": "fiction",
18    "available": true
19  }
20 }

```

Dus gaan we verder kijken en we kunnen zien dat de current stock 0 is.

**Link:** `{{baseUrl}}/books/2`

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'GET' and the URL is `{{baseUrl}}/books/2`. Below this, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', and 'Preview'. The 'Params' tab is selected, showing 'Query Params' with a table containing one entry: 'KEY' with the value 'Key'. Below the tabs, there are tabs for 'Body', 'Cookies', 'Headers (6)', and 'Test Results'. The 'Body' tab is selected, showing a 'Pretty' view of the JSON response. The JSON response is as follows:

```
1 {
2   "id": 2,
3   "name": "Just as I Am",
4   "author": "Cicely Tyson",
5   "type": "non-fiction",
6   "price": 20.33,
7   "current-stock": 0,
8   "available": false
9 }
```

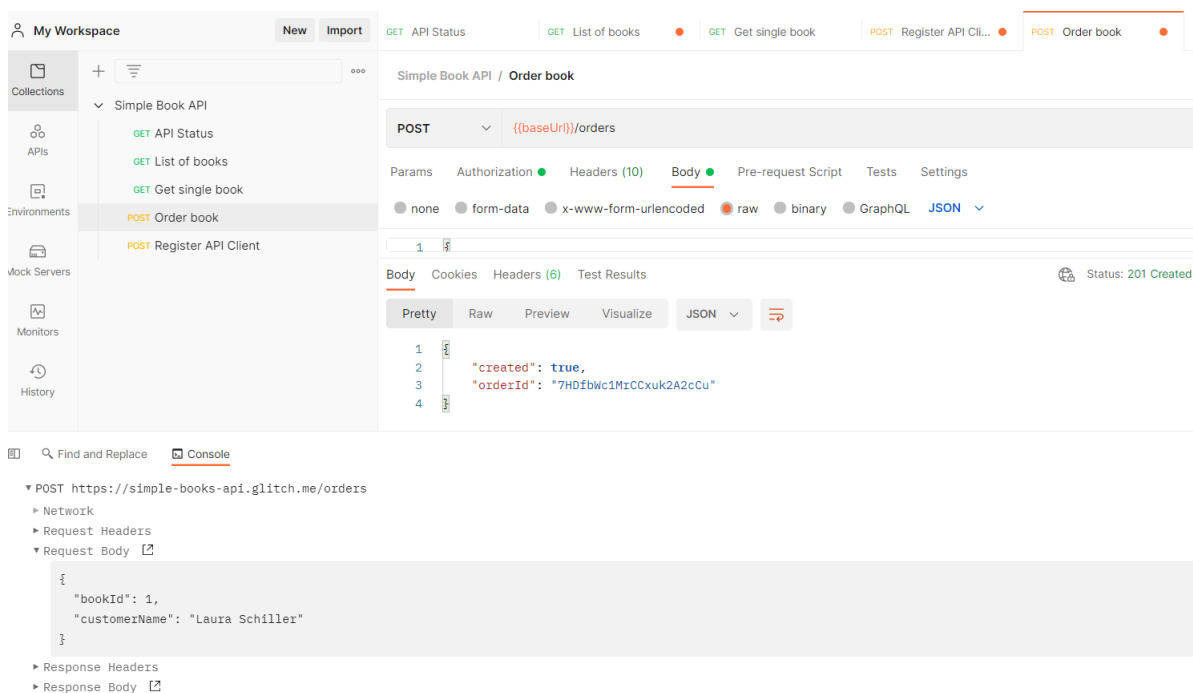
## 2.4 Week 3 – 05/10/2021: 1 uur

- Vervolg request met Postman: 1 uur

### 2.4.1 Testen: Random test data

Aan de hand van de random functie kunnen we een random naam meegeven zoals u hier kunt zien. De random request kunnen we terugvinden in de console

Link: `{{baseUrl}}/orders`



Om te kunnen zien hoeveel orders we hebben gemaakt zullen we een GET request sturen.

Link: `{{baseUrl}}/orders`

```
1 [
2   {
3     "id": "qJ9TRenu2HVC-msgQV2jV",
4     "bookId": 1,
5     "customerName": "John",
6     "createdBy": "787a6eea757499e09fd06f0095692bd62e373ee7f1ecdd92a732e3c710fcb6f9",
7     "quantity": 1,
8     "timestamp": 1633340262649
9   },
10  {
11    "id": "NDbD_GX33u6geyFU6pzUg",
12    "bookId": 1,
13    "customerName": "Gerald O'Reilly",
14    "createdBy": "787a6eea757499e09fd06f0095692bd62e373ee7f1ecdd92a732e3c710fcb6f9",
15    "quantity": 1,
16    "timestamp": 1633451976733
17  },
18  {
19    "id": "7HDFbWc1MrCCxuk2A2cCu",
20    "bookId": 1,
21    "customerName": "Laura Schiller",
22    "createdBy": "787a6eea757499e09fd06f0095692bd62e373ee7f1ecdd92a732e3c710fcb6f9",
23    "quantity": 1,
24    "timestamp": 1633452038109
25  }
26 ]
```

Van de orders die we hebben gemaakt heb ik 1 order uitgekozen en om deze order een GET request gedaan. Als volgt: orderId met de value de id.

**Link:** `{{baseUrl}}/orders/:orderId`

Simple Book API / Get an book orders

GET `{{baseUrl}}/orders/:orderId`

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
orderId	qJ9TRenu2HVC-msgQV2jV	Description

Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": "qJ9TRenu2HVC-msgQV2jV",
3   "bookId": 1,
4   "customerName": "John",
5   "createdBy": "787a6eea757499e09fd06f0095692bd62e373ee7f1ecdd92a732e3c710fcb6f9",
6   "quantity": 1,
7   "timestamp": 1633340262649
8 }
```

## 2.4.2 PATCH request: bestelling bewerken

De request body moet in JSON formaat zijn en maakt het mogelijk om de volgende eigenschappen bij te werken:

- customerName – String

We zullen eerst en vooral een PATCH doen om een randomlastname mee te geven

**Link:** `{{baseUrl}}/orders/:orderId`

PATCH `{{baseUrl}}/orders/:orderId`

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (3) Test Results Status: 204 No Content

Pretty Raw Preview Visualize Text

```

1 {
2   "customerName": "John-{{$randomLastName}}"
3 }
```



Vervolgens zullen we een GET request doen en we kunnen zien dat er een random last name mee gegeven werd.

**Link:** `{{baseUrl}}/orders/:orderId`

GET `{{baseUrl}}/orders/:orderId`

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```

1 {
2   "customerName": "John-{{$randomLastName}}"
3 }

```

Body Cookies Headers (6) Test Results Status: 200 OK

Pretty Raw Preview Visualize **JSON**

```

1 {
2   "id": "qJ9TRenu2Hvc-msgQV2jV",
3   "bookId": 1,
4   "customerName": "John Fay",
5   "createdBy": "787a6eea757499e09fd06f0095692bd62e373ee7f1ecdd92a732e3c710fcb6f9",
6   "quantity": 1,
7   "timestamp": 1633340262649
8 }

```

### 2.4.3 DELETE request: bestelling verwijderen

We zullen dus een order verwijderen aan de hand van een DELETE request

DELETE `{{baseUrl}}/orders/:orderId`

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
orderId	qJ9TRenu2Hvc-msgQV2jV	Description

Body Cookies Headers (3) Test Results Status: 204 No Content

Pretty Raw Preview Visualize **Text**

```

1

```

Om na te gaan of dit echt gelukt is gaan we een GET request doen van het verwijderde Id

GET `{{baseUrl}}/orders/:orderId`

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
orderId	qJ9TRenu2Hvc-msgQV2jV	Description

Body Cookies Headers (6) Test Results Status: 404 Not Found

Pretty Raw Preview Visualize **JSON**

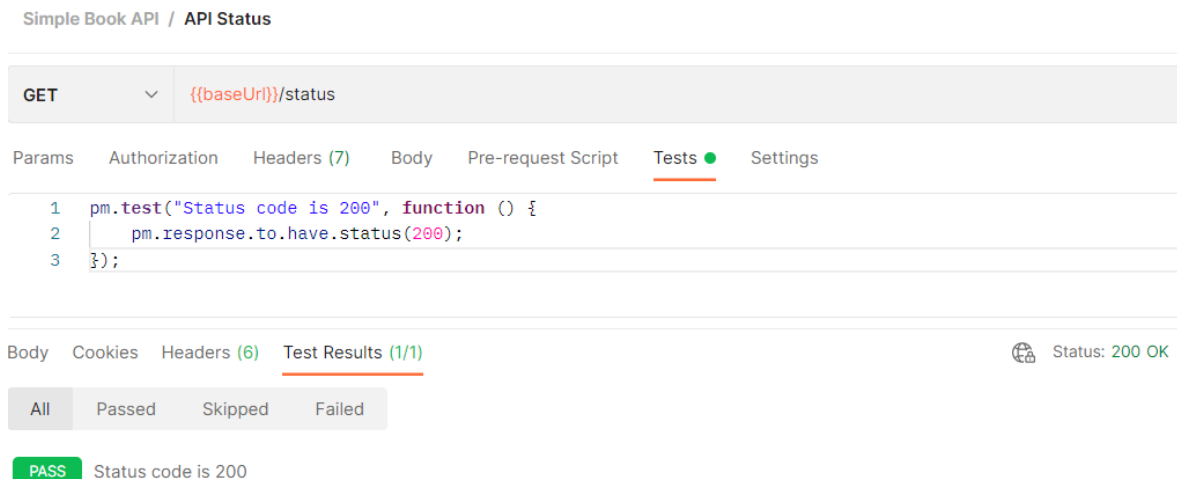
```

1 {
2   "error": "No order with id qJ9TRenu2Hvc-msgQV2jV."
3 }

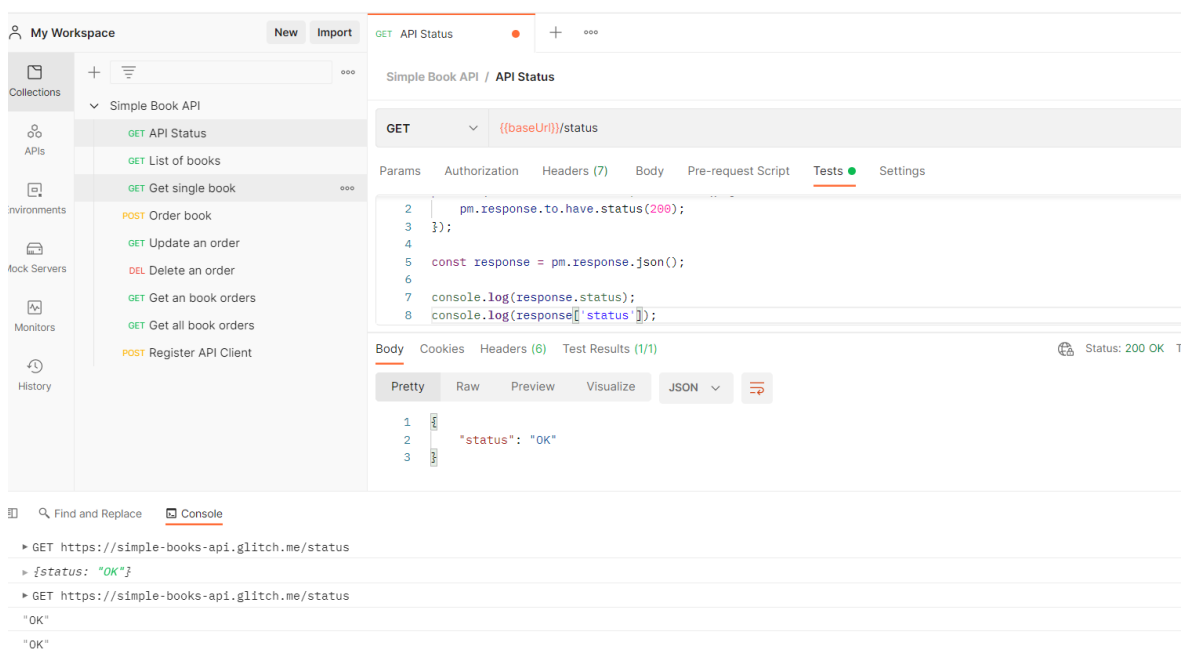
```

### 2.4.4 Writing API test

Hier ik heb gebruik gemaakt van de geschreven code dat postman ons aanbiedt namelijk status code: code is 200. Zoals u kunt zien lukt de test.



Aan de hand van de console kunnen we een javascript object terug krijgen.



We zullen hier een code schrijven waar we vragen dat de response gelijk moet zijn aan "OK" als waar is zal de test lukken anders niet.


GET ▼ `{{baseUrl}}/status`

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

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 const response = pm.response.json();
6
7 console.log(response.status);
8 console.log(response['status']);
9
10 pm.test("Status should be OK", () => {
11   pm.expect(response.status).toEqual("OK");
12 });

```

Body Cookies Headers (6) **Test Results (2/2)**  Status: 200 OK

All Passed Skipped Failed

**PASS** Status code is 200

**PASS** Status should be OK

Hier is de response niet gelijk aan “OK” dus is de test niet gelukt.

GET ▼ `{{baseUrl}}/status`

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

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 const response = pm.response.json();
6
7 console.log(response.status);
8 console.log(response['status']);
9
10 pm.test("Status should be OK", () => {
11   pm.expect(response.status).toEqual("Ossk");
12 });

```

Body Cookies Headers (6) **Test Results (1/2)**  Status: 200 OK

All Passed Skipped Failed

**PASS** Status code is 200

**FAIL** Status should be OK | AssertionError: expected 'OK' to deeply equal 'Ossk'

Order books status: 201

POST ▼ {{baseUrl}}/orders

Params Authorization ● Headers (10) Body ● Pre-request Script **Tests ●** Settings

```

1 pm.test("Status code is 201", function () {
2   pm.response.to.have.status(201);
3 });

```

Body Cookies Headers (6) **Test Results (1/1)** Status: 201 Created

All Passed Skipped Failed

**PASS** Status code is 201

## 2.4.5 Code tests

```

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

const response = pm.response.json();

console.log(response.status);
console.log(response['status']);

pm.test("Status should be OK", () => {
  pm.expect(response.status).toEqual("Ossk");
});

```

### **3 bron vermelding**

#### **3.1 Video**

[https://www.youtube.com/watch?v=VywxIQ2ZXw4&ab\\_channel=freeCodeCamp.org](https://www.youtube.com/watch?v=VywxIQ2ZXw4&ab_channel=freeCodeCamp.org)

#### **3.2 Gratis read only**

[https://drive.google.com/file/d/1JzwW9zwSBKii039\\_aOVC6USTLQzB\\_S/view](https://drive.google.com/file/d/1JzwW9zwSBKii039_aOVC6USTLQzB_S/view)

#### **3.3 Udemy cursus**

<https://www.udemy.com/course/postman-the-complete-guide/>

<https://www.udemy.com/course/introduction-to-postman-a-beginners-guide/learn/lecture/19582570#overview>