

Naam: Samy Sah

R-nummer: R0798534

Datum: 20/09/2021

OPDRACHT – POSTMAN TESTING

Opleidingsonderdeel: Software Testing

Docent: Serneels Frank

jaar: 2021-2022

INHOUDSOPGAVE

| | | |
|----------|--|-----------|
| 1 | RESEARCH | 3 |
| 1.1 | BENODIGDHEDEN | 3 |
| 2 | LOGBOEK | 4 |
| 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: 1 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> |
| 2.4.5 | <i>Code tests</i> | <i>20</i> |
| 2.5 | WEEK 4 – 11/10/2021: 2 UUR | 20 |
| 2.6 | WEEK 4 – 12/10/2021: 2 UUR 20 MIN | 20 |
| 2.6.1 | <i>Postman variables</i> | <i>20</i> |
| 2.6.2 | <i>Extracting data from the response</i> | <i>21</i> |
| 2.6.3 | <i>Collection runner</i> | <i>26</i> |
| 2.6.4 | <i>Request execution order</i> | <i>26</i> |
| 2.6.5 | <i>Postman monitors</i> | <i>27</i> |
| 2.6.6 | <i>Newman</i> | <i>28</i> |
| 2.6.7 | <i>HTML reports with Newman</i> | <i>31</i> |
| 2.7 | WEEK 5 – 18/10/2021: 2 UUR 25 MIN | 32 |
| 2.8 | WEEK 6 – 25/10/2021: 2 UUR | 32 |
| 2.9 | WEEK 7 – 07/11/2021: 2 UUR | 32 |
| 2.10 | WEEK 8 – 08/11/2021: 2 UUR | 32 |
| 2.11 | WEEK 8 – 09/11/2021: 30 MIN | 33 |
| 2.12 | WEEK 9 – 22/11/2021: 1 UUR 30 MIN | 36 |
| 3 | BRON VERMELDING | 42 |
| 3.1 | VIDEO | 42 |
| 3.2 | GRATIS READ ONLY | 42 |
| 3.3 | UDEMY CURSUS | 42 |
| 3.4 | POSTMAN DOCUMENTATIE | 42 |

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 Search

```

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 | | | |

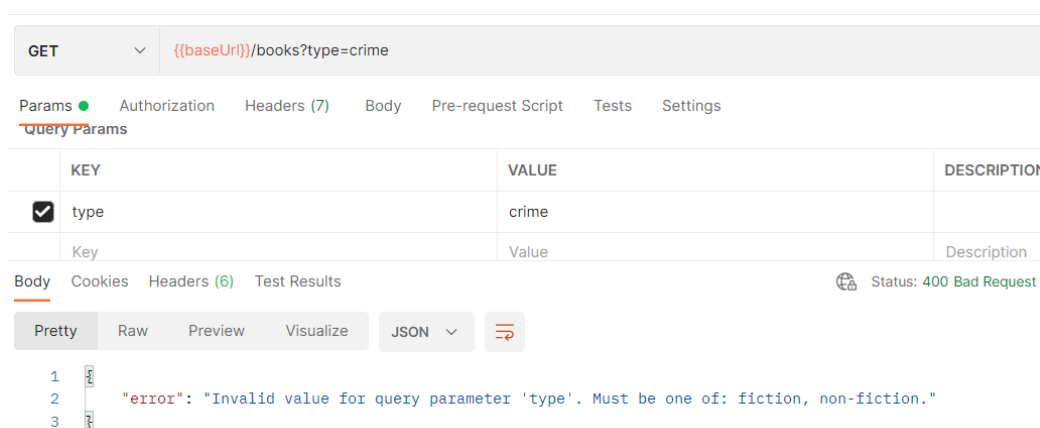
2.3.3 Query parameters

JSON is het meest populaire formaat dat API's gebruiken om gegevens te verzenden.

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 ⌵ 🔗

```

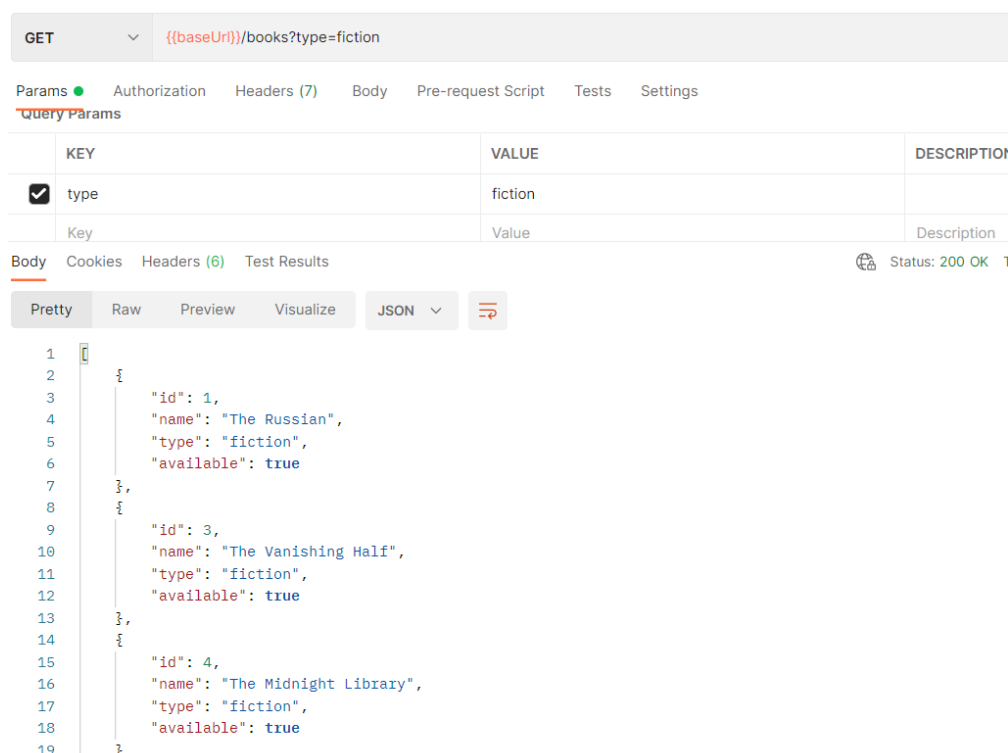
1  {
2    "error": "Invalid value for query parameter 'type'. Must be one of: fiction, non-fiction."
3  }

```

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 1

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 }

```

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: Key, Value: Value, Description: Description.
- Path Variables:** A table with 3 columns: KEY, VALUE, and DESCRIPTION. It contains one entry: bookId: 2, Description: 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: Description.
- Body:** A JSON object: `{ "error": "No book with id 200" }`
- Status:** 404 Not Found

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 vergeten 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.

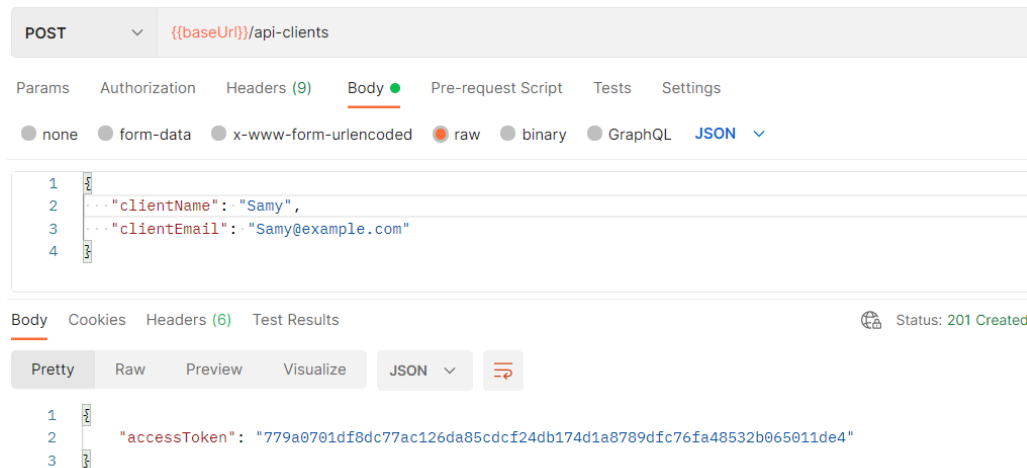
Link: `{{baseUrl}}/api-clients`

POST `/api-clients/`

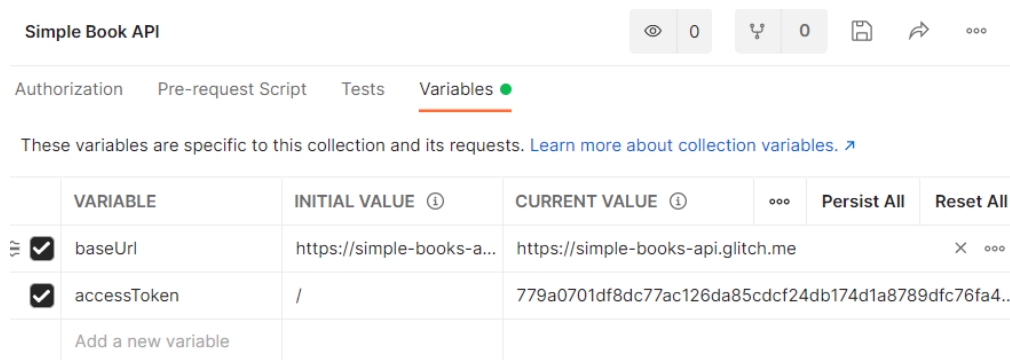
De request body moet in JSON formaat zijn en de volgende eigenschappen bevatten:

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

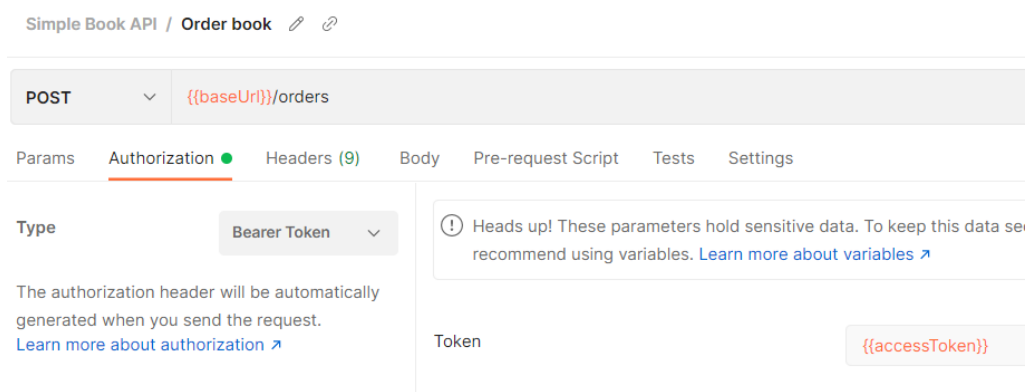
In de response body zullen we de access token terugvinden.



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 ① | Bearer 779a0701df8dc77ac126da85cdcf24db174d1a878... |
| <input checked="" type="checkbox"/> Cache-Control ① | no-cache |
| <input checked="" type="checkbox"/> Postman-Token ① | <calculated when request is sent> |
| <input checked="" type="checkbox"/> Content-Length ① | 0 |
| <input checked="" type="checkbox"/> Host ① | <calculated when request is sent> |
| <input checked="" type="checkbox"/> User-Agent ① | PostmanRuntime/7.28.4 |
| <input checked="" type="checkbox"/> Accept ① | */* |
| <input checked="" type="checkbox"/> Accept-Encoding ① | gzip, deflate, br |
| <input checked="" type="checkbox"/> Connection ① | 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`

GET

▼

{{baseUrl}}/books/2

Params

Authorization

Headers (7)

Body

Pre

Query Params

| | KEY |
|--|-----|
| | Key |

Body

Cookies

Headers (6)

Test Results

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  }
```

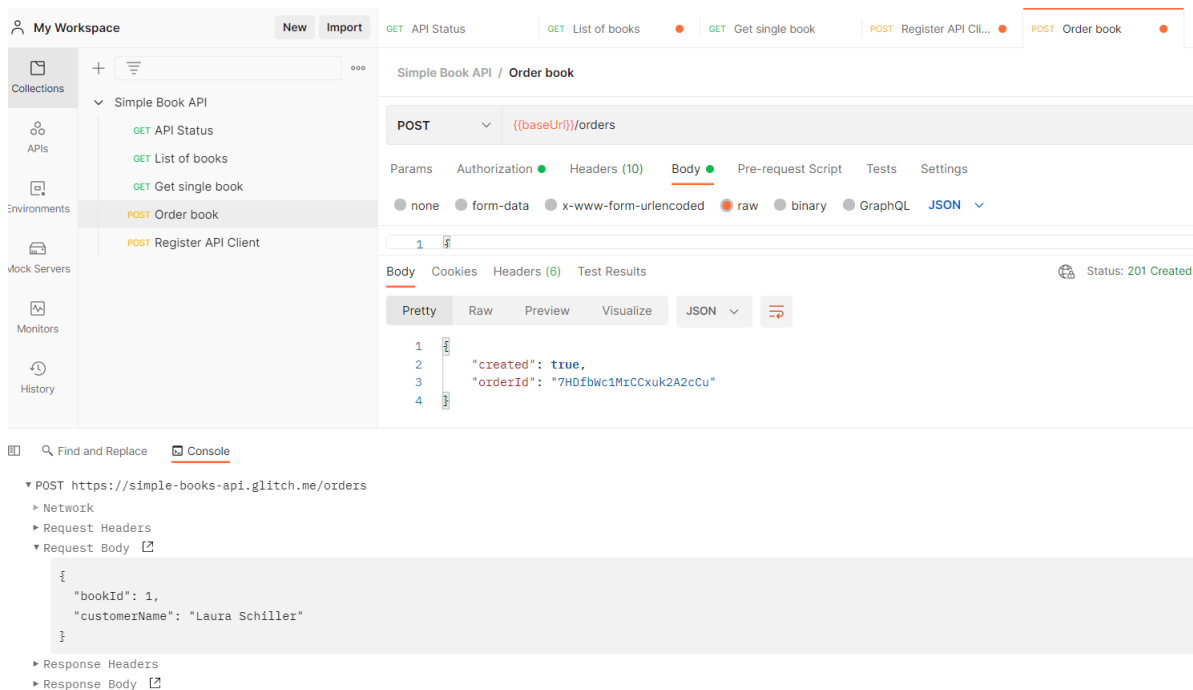
2.4 Week 3 – 05/10/2021: 1 uur

- 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**

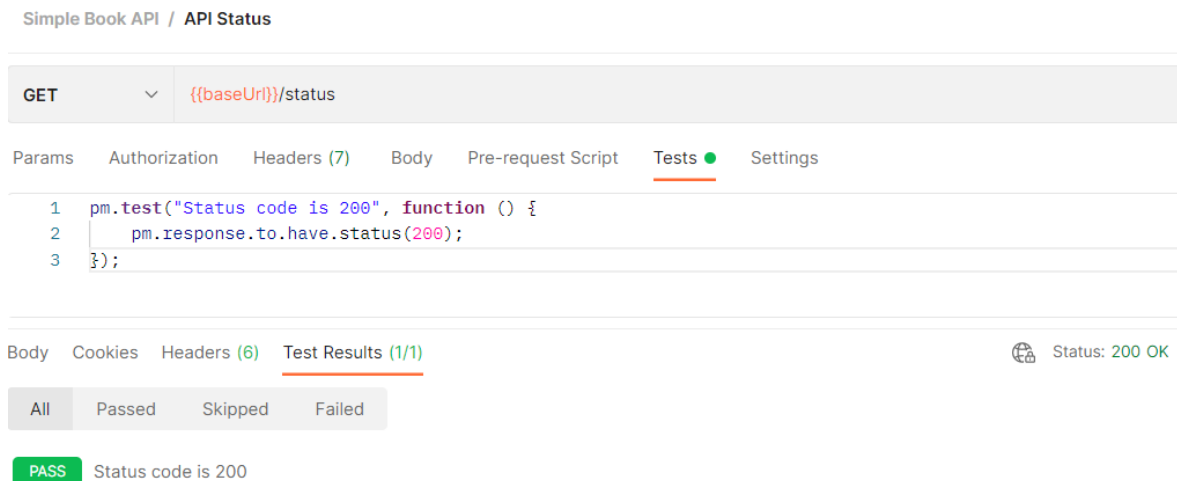
```

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

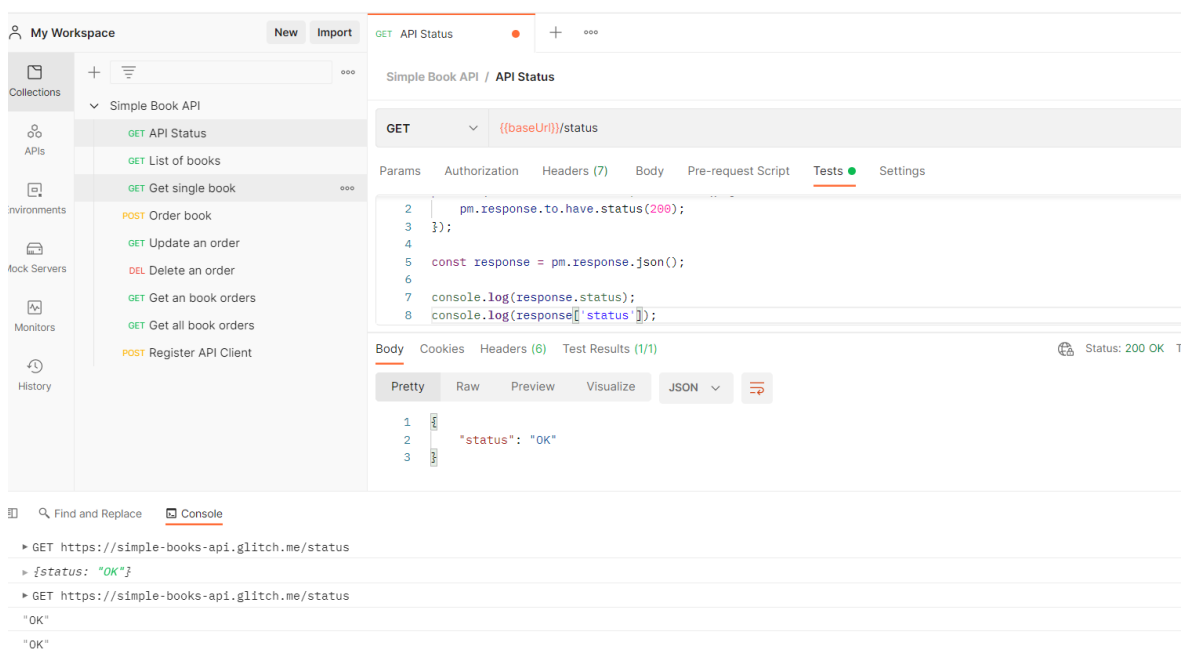
```

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).to.eql("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).to.eql("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");
});

```

2.5 Week 4 – 11/10/2021: 2 uur

- Gastcollege gevolgd: 2 uur

2.6 Week 4 – 12/10/2021: 2 uur 20 min

- Test automation with Postman: 2 uur
- Video gekeken in verband met writing API tests: 20 min

https://www.youtube.com/watch?v=Qlvbc6kIBOk&ab_channel=ValentinDespa

2.6.1 Postman variables

Ik heb eerst en vooral een globale variabele aangemaakt namelijk orderId

Globals

Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed [Learn more about globals](#)

| | VARIABLE | INITIAL VALUE ⓘ | CURRENT VALUE ⓘ |
|-------------------------------------|--------------------|-----------------|-----------------------|
| <input checked="" type="checkbox"/> | orderId | | FIWj4SNNtalzRoWQoKGVk |
| | Add a new variable | | |

Vervolgens heb ik deze variabeel meegegeven in de GET request als value en we kunnen zien dat dit dezelfde orderId is.

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 | DESCRIPTIC |
|-----|-------|-------------|
| Key | Value | Description |

Path Variables

| KEY | VALUE | DESCRIPTIC |
|---------|--------------------------|-------------|
| orderId | <code>{{orderId}}</code> | Description |

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

Pretty Raw Preview Visualize JSON

```

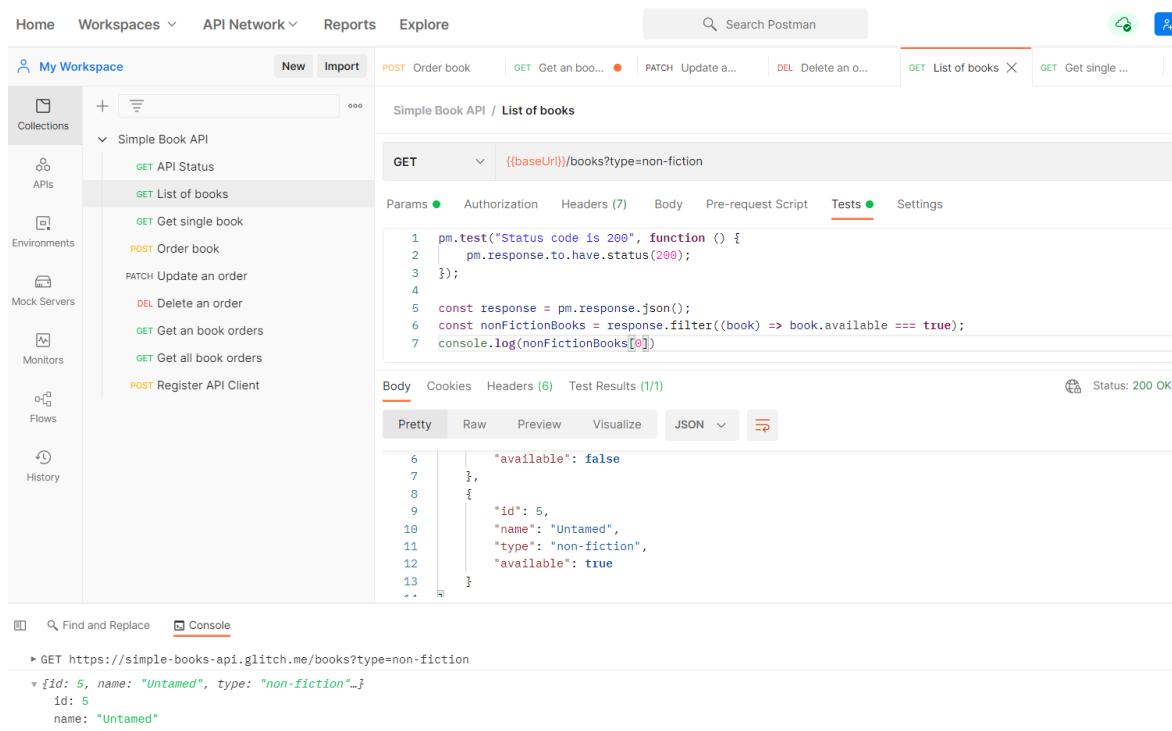
1  {
2    "id": "FIWj4SNNtalzRoWQoKGVk",
3    "bookId": 1,
4    "customerName": "Cody Wunsch",
5    "createdBy": "787a6eea757499e09fd06f0095692bd62e373ee7f1ecdd92a732e3c710fcb6f9",
6    "quantity": 1,
7    "timestamp": 1634030170703
8  }
```

2.6.2 Extracting data from the response

Eerst en vooral heb ik een constante aangemaakt met de naam `nonFictionBooks` met de response waar ik een methode filter heb gebruikt met een condition in een andere function. Hier zullen we een parameter `book` ontvangen en aan de hand van deze book kunnen we de condition definiëren. Namelijk of de book beschikbaar is. Aan de hand van de `console.log` zullen we de constante de eerste element ontvangen en deze element zal zowiezo true zijn

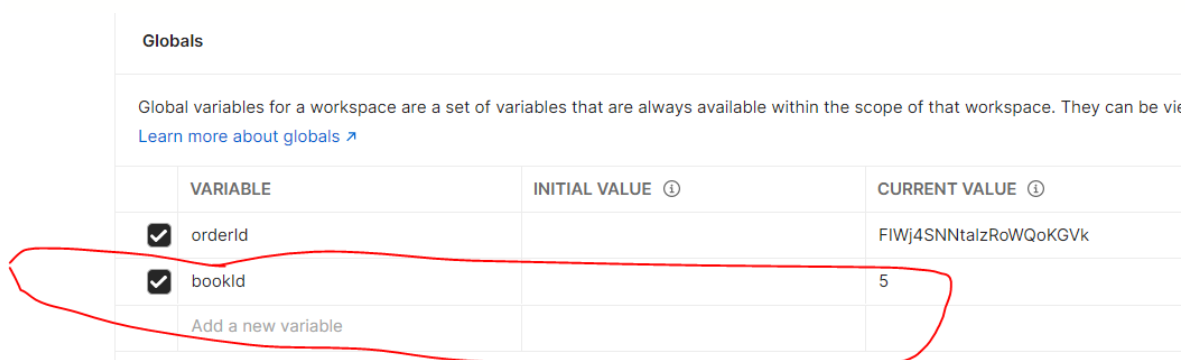
Code:

```
const response = pm.response.json();
const nonFictionBooks = response.filter((book) => book.available === true);
console.log(nonFictionBooks[0])
```



Aan de hand van de code snippets van Postman namelijk “Set a global variable”. Heb ik de global variabele gemaakt.

```
pm.globals.set("bookId", nonFictionBooks[0].id);
```



Vervolgens heb ik enkele test geschreven om de code te testen.

Aan de hand van pm.test() zullen we deze testen schrijven.

```
pm.expect(book.available).to.be.true;
pm.expect(book.available).to.eql(true);
```

dit zal hetzelfde resultaat weergeven maar in op een verschillende manier geschreven.

Simple Book API / List of books

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

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

```

7 console.log(nonFictionBooks[0])
8
9 const book = nonFictionBooks[0];
10 pm.test("Book found", () =>{
11   pm.expect(book).to.be.an('object');
12   pm.expect(book.available).to.be.true;
13   pm.expect(book.available).to.eql(true);
14 });
15
16 pm.globals.set("bookId", book.id);
17

```

Body Cookies Headers (6) Test Results (2/2)

Status: 200 OK T

All Passed Skipped Failed

PASS Status code is 200

PASS Book found

We kunnen de test ook laten falen.

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

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

```

3  });
4
5  const response = pm.response.json();
6  const nonFictionBooks = response.filter((book) => book.available === 22);
7  console.log(nonFictionBooks[0])
8
9  const book = nonFictionBooks[0];
10
11  if(book){
12    pm.globals.set("bookId", book.id);
13  }
14
15  pm.test("Book found", () =>{
16    pm.expect(book).to.be.an('object');
17    pm.expect(book.available).to.be.true;
18    pm.expect(book.available).to.eql(true);
19  });
20
21

```

Body Cookies Headers (6) Test Results (1/2)

Status: 200 OK

All Passed Skipped Failed

PASS Status code is 200

FAIL Book found | AssertionError: expected undefined to be an object

Hier krijg ik te zien dat mijn laatste expectation niet is gelukt omdat we de query params van type hebben uitgezet.

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

Query Params

| | KEY | VALUE |
|--------------------------|------|-------------|
| <input type="checkbox"/> | type | non-fiction |

GET ▼ {{baseUrl}}/books

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

```

5 const response = pm.response.json();
6 const nonFictionBooks = response.filter((book) => book.available === true);
7 console.log(nonFictionBooks[0])
8
9 const book = nonFictionBooks[0];
10
11 if(book){
12   pm.globals.set("bookId", book.id);
13 }
14
15 pm.test("Book found", () =>{
16   pm.expect(book).to.be.an('object');
17   pm.expect(book.available).to.be.true;
18   pm.expect(book.available).to.eql(true);
19   pm.expect(book.type).to.eql("non-fiction");
20 });
21
22
```

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

All Passed Skipped Failed

PASS Status code is 200

FAIL Book found | AssertionError: expected 'fiction' to deeply equal 'non-fiction'

Maar als we deze weer activeren zullen alle testen lukken.

GET ▼ {{baseUrl}}/books?type=non-fiction

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

Query Params

| | KEY | VALUE | DESCRIPTIC |
|-------------------------------------|-------|-------------|-------------|
| <input checked="" type="checkbox"/> | type | non-fiction | |
| <input type="checkbox"/> | Limit | 2 | |
| | Key | Value | Description |

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

All Passed Skipped Failed

PASS Status code is 200

PASS Book found

Dit is een test waar we gaan kijken of de book in stock is of niet in dit geval in book met id 5 in stock.

Simple Book API / Get single book

GET ▼ `{{baseUrl}}/books/:id`

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   pm.test("Is in stock", function () {
8     |   pm.expect(response['current-stock']).to.be.above(0);
9     | });
10

```

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

All Passed Skipped Failed

PASS Status code is 200

PASS Is in stock

In het geval van book met Id 2 is deze niet in stock zoals u kunt zien aan de test.

Simple Book API / Get single book

GET ▼ `{{baseUrl}}/books/:id`

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

Query Params

| KEY | VALUE | DESCRIPTIC |
|-----|-------|-------------|
| Key | Value | Description |

Path Variables

| KEY | VALUE | DESCRIPTIC |
|-----|-------|-------------|
| id | 2 | Description |

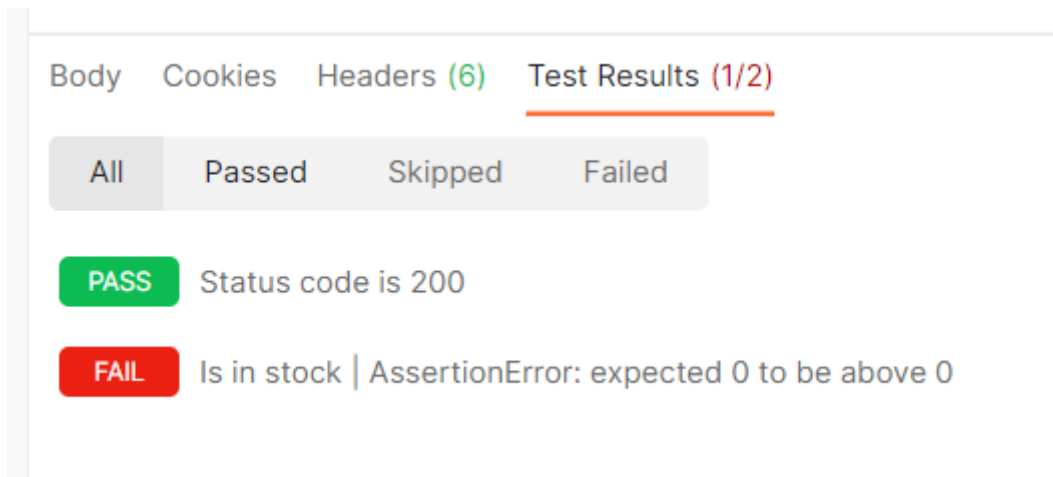
Body Cookies Headers (6) **Test Results (1/2)** 🌐 Status: 200 OK

Pretty Raw Preview Visualize JSON ▼ ≡

```

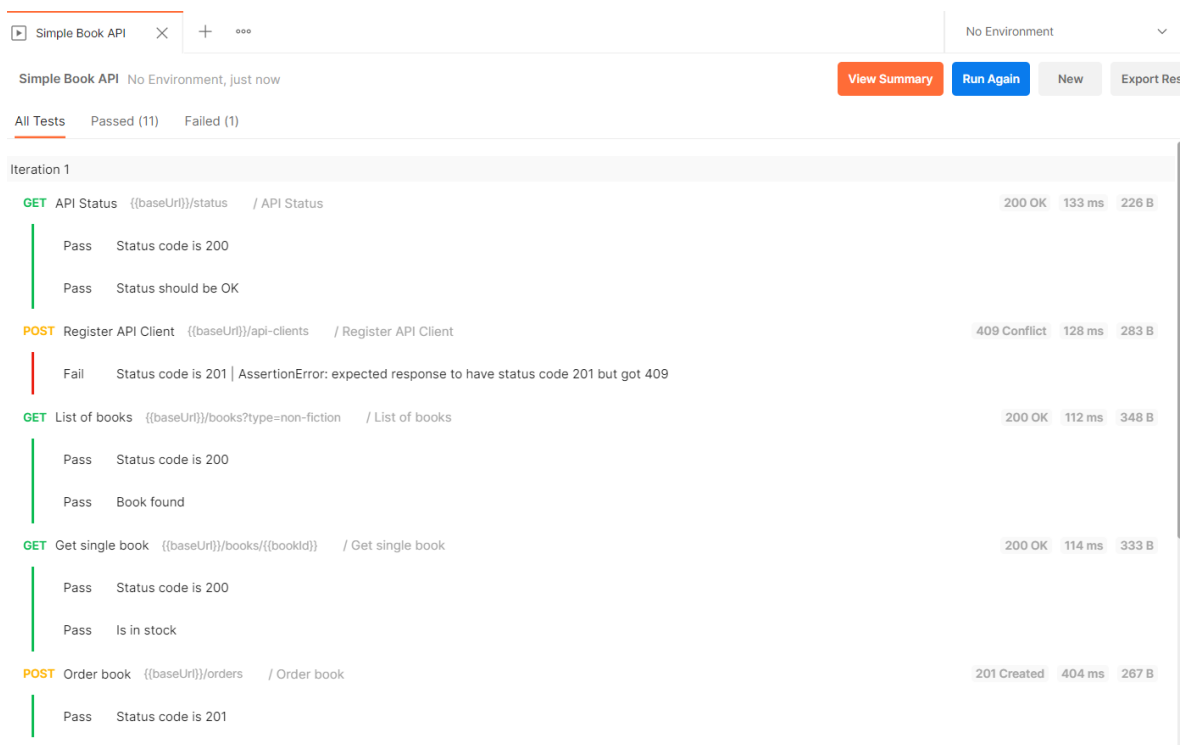
1 {
2   "id": 2,
3   "name": "Just as I Am",
4   "author": "Cicely Tyson",
5   "type": "non-fiction",
6   "price": 20.33,
7   "current-stock": 0,
8   "available": false
9 }

```



2.6.3 Collection runner


Met de Collection Runner is het mogelijk om sets van requests in een bepaalde volgorde uit te voeren. Verzamelingen kunnen tegen specifieke omgevingen worden uitgevoerd en gegevensbestanden kunnen aan een run worden toegevoegd. Met collection runs automatiseert u uw API testen. U kunt collection runs integreren in uw CI/CD pijplijn door gebruik te maken van Postman's CLI Newman.



2.6.4 Request execution order

Aan de hand van deze code kunnen we de test van register API skippen.

```
postman.setNextRequest("List of books");
```

Simple Book API No Environment, just now 

View Summary

Run Again

New

Export Re

All Tests Passed (11) Failed (0)

Iteration 1

GET

API Status

{{baseUrl}}/status / API Status

200 OK403 ms226 B

Pass

Status code is 200

Pass

Status should be OK

GET

List of books

{{baseUrl}}/books?type=non-fiction / List of books

200 OK135 ms348 B

Pass

Status code is 200

Pass

Book found

GET

Get single book

{{baseUrl}}/books/{{bookId}} / Get single book

200 OK121 ms333 B

Pass

Status code is 200

Pass

Is in stock

POST

Order book

{{baseUrl}}/orders / Order book

201 Created468 ms267 B

Pass

Status code is 201

GET

Get an book orders

{{baseUrl}}/orders/{{orderId}} / Get an book orders

200 OK333 ms401 B

Pass

Status code is 200

Een tweede manier om die test niet de doen is door deze te verplaatsen aan het einde en in de tests van delete an order deze code te schrijven.

```
postman.setNextRequest (null) ;
```


2.6.5 Postman monitors

Deze testen falen omdat we de initial value niet hadden gespecificeerd.

Failures in the monitor - Check Books API

PM Postman Monitors <noreply@notifications.getpostman.com> 12:29 PM

To: Sammy Sah


POSTMAN

Houston, we have a problem

One or more requests in the Simple Book API collection have failed 5 tests across 1 region during the run.

| | |
|-------------|---------------------------------|
| Monitor | Check Books API |
| Collection | Simple Book API |
| Environment | None |
| Workspace | My Workspace |
| Failed at | 10:29 AM UTC 12 Oct 2021 |

[View Run Results](#)

Last 10 runs

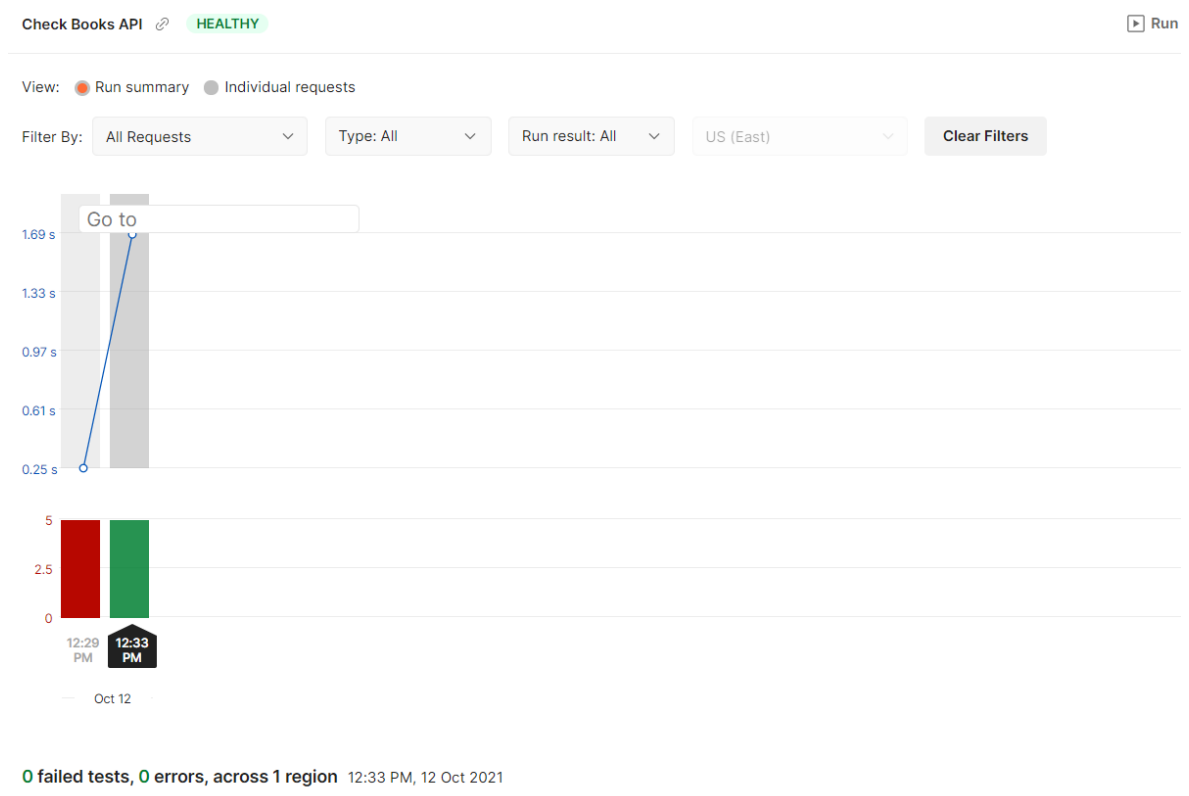
Simple Book API

Authorization Pre-request Script Tests **Variables**

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

| | VARIABLE | INITIAL VALUE ⓘ | CURRENT VALUE ⓘ | ... | Persist All | Reset |
|-------------------------------------|--------------------|---------------------------|--|-----|-------------|-------|
| <input checked="" type="checkbox"/> | baseUrl | https://simple-books-a... | https://simple-books-api.glitch.me | | | |
| <input checked="" type="checkbox"/> | accessToken | / | 779a0701df8dc77ac126da85cdcf24db174d1a8789dfc76f | | | |
| | Add a new variable | | | | | |

Na het geven van de initial value kunnen zien dat alle testen zijn gelukt.



2.6.6 Newman

Newman is een command-line Collection Runner voor Postman. Hiermee is het mogelijk om een Postman Collection direct vanaf de opdrachtregel te draaien en te testen.

Eerst en vooral heb ik newman geïnstalleerd aan de hand van deze commando:

```
$ npm install -g newman
```

```

Administrator: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\

C:\>npm i -g newman
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 128 packages, and audited 129 packages in 7s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\>newman --version
5.3.0

C:\>

```

In de command line zullen we eerst navigeren naar de folder waar we de json file hebben gesaved in dit geval is dat in deze folder: “C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing”.

Vervolgens zullen we aan de hand van deze commando newman laten starten: “newman run postman_collection.json”.

```

Administrator: C:\WINDOWS\system32\cmd.exe
10/12/2021 12:18 PM          5,722 Simple Book API.postman_test_run.json
          5 File(s)      4,647,136 bytes
          2 Dir(s)  26,162,860,032 bytes free

C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing>newman run postman_collection.json
newman

Simple Book API
→ API Status
GET https://simple-books-api.glitch.me/status [200 OK, 226B, 609ms]
✓ Status code is 200
[
  'OK',
  'OK'
]
✓ Status should be OK

Attempting to set next request to List of books

→ List of books
GET https://simple-books-api.glitch.me/books?type=non-fiction [200 OK, 348B, 207ms]
✓ Status code is 200
[
  { id: 5, name: 'Untamed', type: 'non-fiction', av
    aillable: true }
]
✓ Book found

→ Get single book

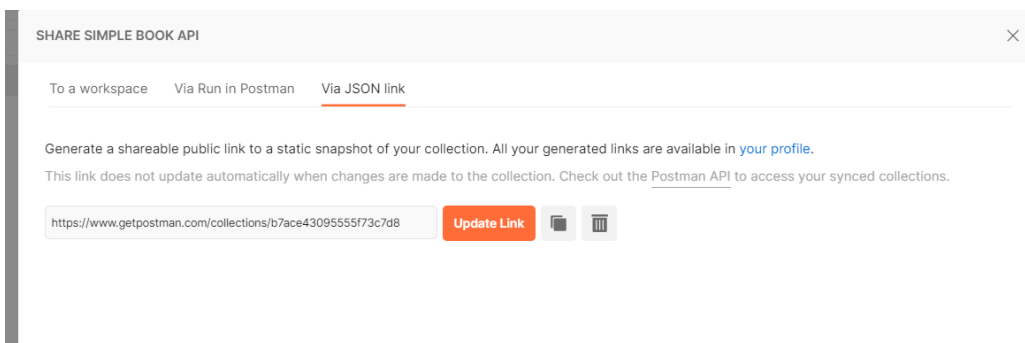
```

```
Administrator: C:\WINDOWS\system32\cmd.exe
✓ Status code is 204
→ Delete an order
DELETE https://simple-books-api.glitch.me/orders/BrjW4Hvbup_ef_c7iHH1Q [204 No Content, 111B, 665ms]
✓ Status code is 204
```

| | executed | failed |
|--|----------|--------|
| iterations | 1 | 0 |
| requests | 8 | 0 |
| test-scripts | 16 | 0 |
| prerequisite-scripts | 8 | 0 |
| assertions | 11 | 0 |
| total run duration: 4.1s | | |
| total data received: 2.41kB (approx) | | |
| average response time: 431ms [min: 166ms, max: 665ms, s.d.: 181ms] | | |

C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing>

Dit resultaten kunnen we ook verkrijgen aan de hand van de JSON link



We zullen de commando newman run uitvoeren met de link die we in Postman verkrijgen.

```
Administrator: C:\WINDOWS\system32\cmd.exe

C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing>
C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing>newman run https://www.getpostman.com/collections/b7ace4309555f73c7d8
newman

Simple Book API
→ API Status
GET https://simple-books-api.glitch.me/status [200 OK, 226B, 620ms]
✓ Status code is 200
[
  'OK'
  'OK'
]
✓ Status should be OK

Attempting to set next request to List of books
→ List of books
GET https://simple-books-api.glitch.me/books?type=non-fiction [200 OK, 348B, 208ms]
✓ Status code is 200
[
  { id: 5, name: 'Untamed', type: 'non-fiction', available: true }
]
✓ Book found
→ Get single book
```

2.6.7 HTML reports with Newman

Ik heb eerst deze commando uitgevoerd in de command line om gebruik te kunnen maken van html reports in newman.

“npm i -g newman-reporter-htmlextra”

Hierna heb ik deze commando uitgevoerd om een html reports te krijgen.

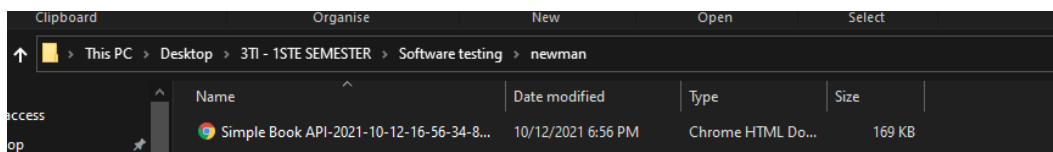
```

Select Administrator: C:\WINDOWS\system32\cmd.exe
C:\Users\sahsa>cd C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing
C:\Users\sahsa\Desktop\3TI - 1STE SEMESTER\Software testing>newman run https://www.getpostman.com/collections/b7ace4309555f73c7d8 --reporters cli,htmlextra
newman
Simple Book API
Using htmlextra version 1.22.1

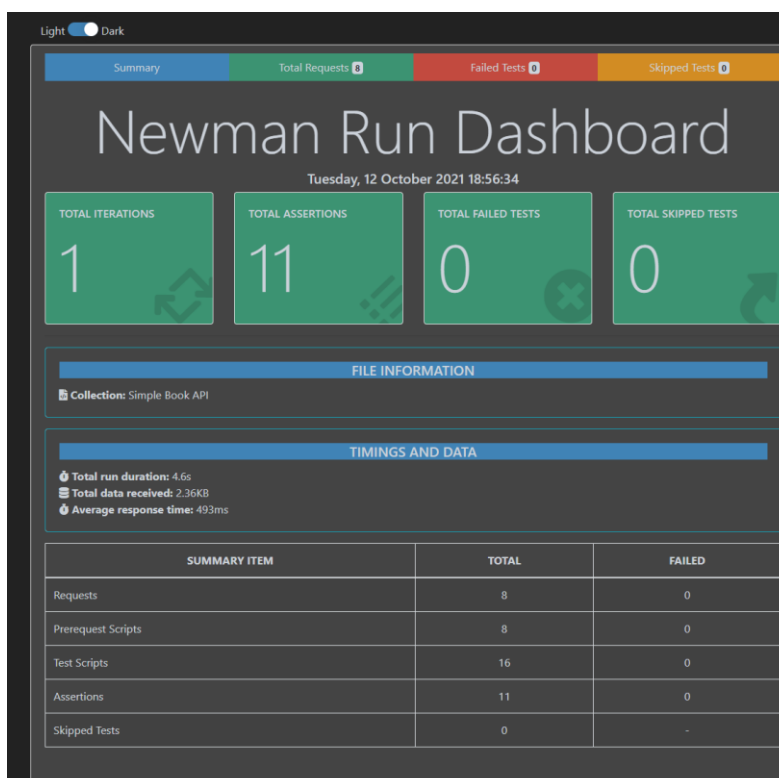
→ API Status
GET https://simple-books-api.glitch.me/status [200 OK, 226B, 955ms]
✓ Status code is 200
  [
    'OK',
    'OK'
  ]
✓ Status should be OK
Attempting to set next request to List of books

→ List of books
GET https://simple-books-api.glitch.me/books?type=non-fiction [200 OK, 348B, 183ms]
✓ Status code is 200
  [
    {
      'id': 1,
      'title': 'The Hobbit',
      'author': 'J.R.R. Tolkien',
      'year': 1937,
      'type': 'Fiction'
    },
    {
      'id': 2,
      'title': 'The Lord of the Rings: The Fellowship of the Ring',
      'author': 'J.R.R. Tolkien',
      'year': 1954,
      'type': 'Fiction'
    },
    {
      'id': 3,
      'title': 'The Lord of the Rings: The Two Towers',
      'author': 'J.R.R. Tolkien',
      'year': 1954,
      'type': 'Fiction'
    },
    {
      'id': 4,
      'title': 'The Lord of the Rings: The Return of the King',
      'author': 'J.R.R. Tolkien',
      'year': 1955,
      'type': 'Fiction'
    },
    {
      'id': 5,
      'title': 'The Silmarillion',
      'author': 'J.R.R. Tolkien',
      'year': 1977,
      'type': 'Fiction'
    },
    {
      'id': 6,
      'title': 'The Hobbit and The Lord of the Rings',
      'author': 'J.R.R. Tolkien',
      'year': 1937-1955,
      'type': 'Fiction'
    },
    {
      'id': 7,
      'title': 'The Hobbit and The Lord of the Rings: The Fellowship of the Ring',
      'author': 'J.R.R. Tolkien',
      'year': 1937-1954,
      'type': 'Fiction'
    },
    {
      'id': 8,
      'title': 'The Hobbit and The Lord of the Rings: The Two Towers',
      'author': 'J.R.R. Tolkien',
      'year': 1937-1954,
      'type': 'Fiction'
    }
  ]
  
```

Na het uitvoeren van deze commando heb ik een nieuwe folder met een file ontvangen in mijn folder.



In deze report kunnen we heel veel informatie terugvinden in verband met de Request orders.



2.7 Week 5 – 18/10/2021: 2 uur 25 min

- Write API test part twee en drie: 25 min
- Gastcollege gevolgd: 2 uur

2.8 Week 6 – 25/10/2021: 2 uur

- Gastcollege 2 gevolgd: 2 uur

2.9 Week 7 – 07/11/2021: 2 uur

- Writing API tests: 2 uur

2.10 Week 8 – 08/11/2021: 2 uur

- Gastcollege 3 gevolgd: 2 uur

GET request met testen

The screenshot shows the Postman interface with a GET request to `https://petstore3.swagger.io/api/v3/pet/10`. The **Tests** tab is selected, showing the following JavaScript test script:

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5
6 pm.test("Response time is less than 450ms", function () {
7   pm.expect(pm.response.responseTime).to.be.below(450);
8 });
9
10 var jsonData = pm.response.json();
11 pm.test("Correct ID", function () {
12   pm.expect(jsonData.id).to.eql(10);
13 });
14 pm.test("Correct Name", function () {
15   pm.expect(jsonData.name).to.eql("Dogs");
16 });

```

On the right, a list of snippets is visible:

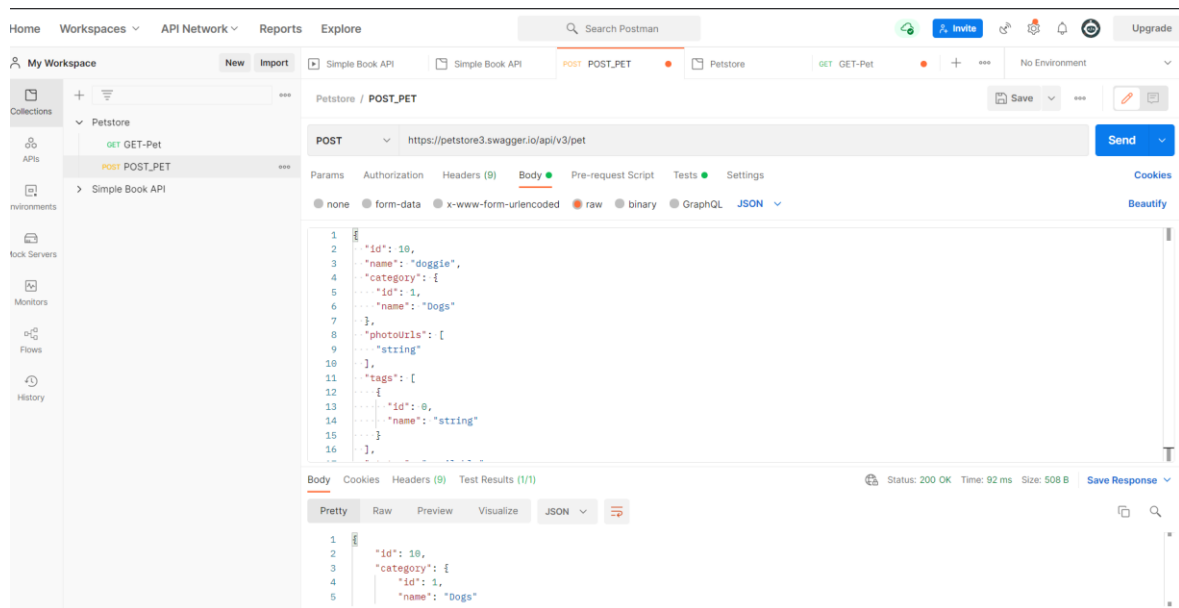
- Test scripts are written in JavaScript, and are run after the response is received. [Learn more about tests scripts](#)
- SNIPPETS
- Response body: Contains string
- Response body: JSON value check
- Response body: Is equal to a string
- Response headers: Content-Type header check
- Response time is less than 200ms
- Status code: Successful POST request
- Status code: Code name has string
- Response body: Convert XML body to a JSON Object
- Use Tiny Validator for JSON data

At the bottom, the **Test Results (3/4)** section shows:

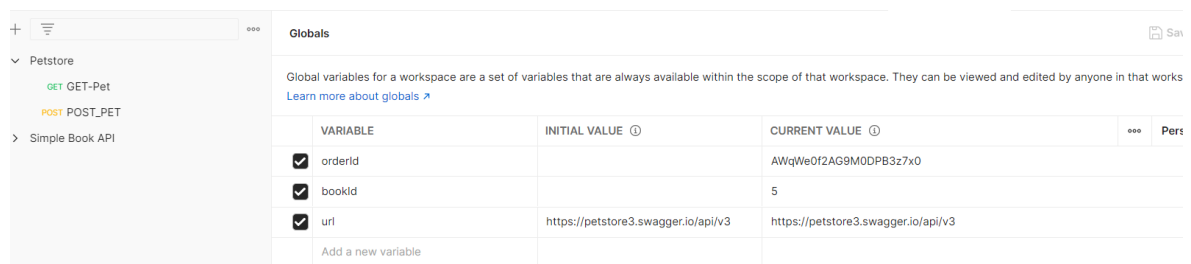
| All | Passed | Skipped | Failed |
|------|----------------------------------|---------|--------|
| PASS | Status code is 200 | | |
| PASS | Response time is less than 450ms | | |
| PASS | Correct ID | | |

The status bar at the bottom indicates: Status: 200 OK, Time: 97 ms, Size: 508 B, and a **Save Response** button.

POST request



Variable url aangemaakt



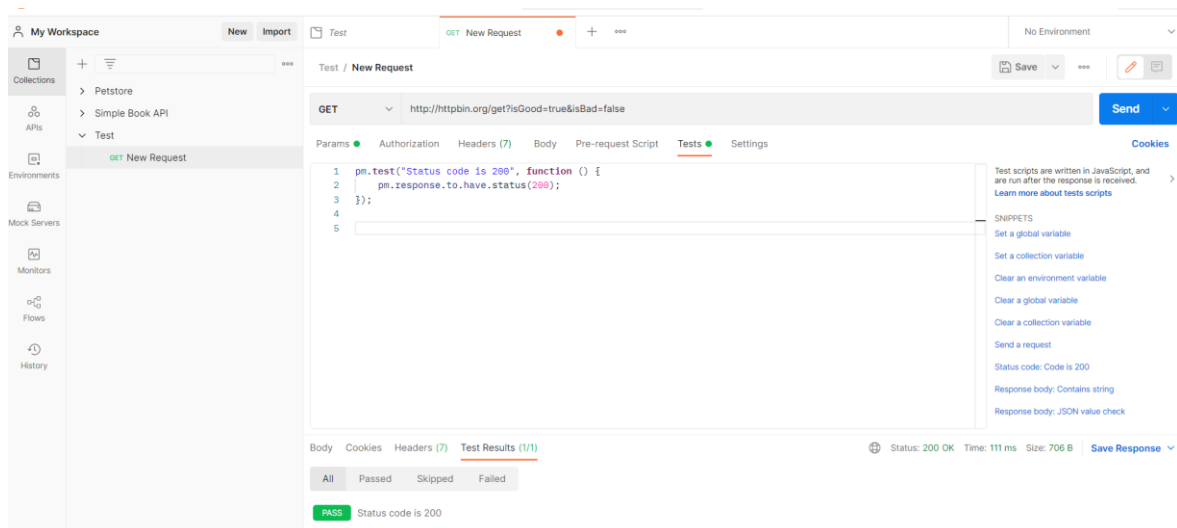
2.11 Week 8 – 09/11/2021: 30 min

- API tests schrijven

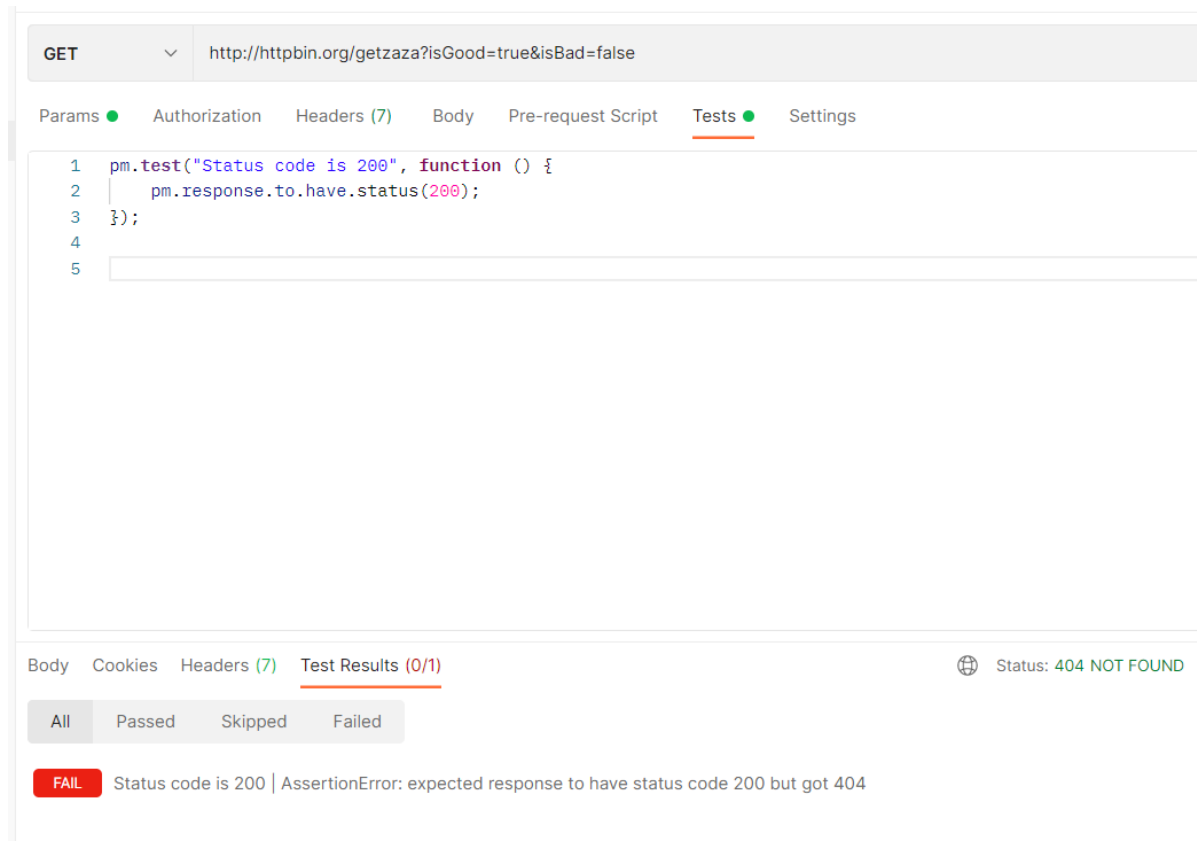
Eerst en vooral heb ik aan de had van de GET request een url meegegeven met twee Query Params namelijk:

| | KEY | VALUE |
|-------------------------------------|--------|-------|
| <input checked="" type="checkbox"/> | isGood | true |
| <input checked="" type="checkbox"/> | isBad | false |
| | Key | Value |

Vervolgens heb ik een simpele test geschreven om te zien of de status code 200 is.



Ik heb dan hierna ook de test doen falen doormiddel van de url te veranderen
Nu krijgen we te zien dat we een 404 not found krijgen.



Aan de hand van `console.log(jsonData.args.isGood)` kunnen we true laten verschijnen.

My Workspace

Test / GET

GET http://httpbin.org/get?isGood=true&isBad=false

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

```

1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 let jsonData = pm.response.json();
6 console.log(jsonData.args.isGood);
7

```

Body Cookies Headers (7) Test Results (1/2)

Pretty Raw Preview Visualize JSON

```

1 {
2   "args": {
3     "isBad": "false",

```

Find and Replace Console

Content-Length: "476"
 Connection: "keep-alive"
 Server: "gunicorn/19.9.0"
 Access-Control-Allow-Origin: "*"
 Access-Control-Allow-Credentials: "true"

Response Body

```

> {isBad: "false", isGood: "true"}

```

GET http://httpbin.org/get?isGood=true&isBad=false

"true"

In dit geval zal de test falen aangezien we hier een string verwachten.

GET http://httpbin.org/get?isGood=true&isBad=false

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 let jsonData = pm.response.json();
6 console.log(jsonData.args.isGood);
7
8 pm.test("isGood", function () {
9   pm.expect(jsonData.args.isGood).to.eql(true);
10 });

```

Body Cookies Headers (7) Test Results (1/2)

Status: 200 OK

All Passed Skipped Failed

PASS Status code is 200

FAIL isGood | AssertionError: expected 'true' to deeply equal true

Om dit op te lossen zullen we enkele aanhalingstekens gebruiken.

GET ▼ http://httpbin.org/get?isGood=true&isBad=false

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 let jsonData = pm.response.json();
6 console.log(jsonData.args.isGood);
7
8 pm.test("isGood", function () {
9   pm.expect(jsonData.args.isGood).to.eql('true');
10  });

```

Body Cookies Headers (7) **Test Results (2/2)** 🌐 Status: 200 OK

All Passed Skipped Failed

PASS Status code is 200

PASS isGood

2.12 Week 9 – 22/11/2021: 1 uur 30 min

- Vervolg API tests schrijven

Pre-request is een script dat altijd geëxecuteerd wordt voor de HTTP request verzonden is. Hiermee kunnen we Javascript code uitvoeren die de URL van het verzoek, de body, de headers enzovoort kan wijzigen.

Home Workspaces ▼ API Network ▼ Reports Explore 🔍 Search Postman 🔗 Invite 🔔 ⚙️ Upgrade

My Workspace New Import POST POST Copy X [CONFLICT] GET GET + ... No Environment 🔗

Collections + 🔍 ...

- Petstore
- Simple Book API
- Test
 - GET GET
 - POST POST Copy**

Environments + 🔍 ...

Mock Servers

Monitors

Flows

History

Test / POST Copy

POST ▼ http://httpbin.org/post Send

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

```

1 let date = Date.now();
2
3 pm.globals.set("currentDate", date);

```

Pre-request scripts are written in JavaScript, and are run before the request is sent.

[Learn more about pre-request scripts](#)

SNIPPETS

- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Send a request

Body Cookies Headers (7) Test Results (1/3) 🌐 Status: 200 OK Time: 252 ms Size: 931 B Save Response

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1 {
2   "args": {},
3   "data": "{\n  \"name\": \"john\", \n  \"permissions\": [2000,3000,4000]\n}",
4   "files": {},
5   "form": {},
6   "headers": {
7     "Accept": "*/*",
8     "Accept-Encoding": "gzip, deflate, br",
9     "Cache-Control": "no-cache",
10    "Content-Length": "63",

```

🔗 Bookmarks 🗑️ Runner 🗑️ Trash


```

14
15 pm.test("currentDate", function () {
16     var jsonData = pm.response.json();
17     pm.expect(jsonData.json.currentDate).to.eql(1637574463686);
18 });
19

```

Body Cookies Headers (7) **Test Results (1/4)**

All Passed Skipped Failed

FAIL isGood | AssertionError: expected undefined to deeply equal 'true'

FAIL isBad | AssertionError: expected undefined to deeply equal 'false'

FAIL currentDate | AssertionError: expected 1637574620226 to deeply equal 1637574463686

Om de test te doen lukken moet je gebruik maken van Get a global variable om de currentDate test te doen lukken. We gaan expect dat iets van de response body gelijk gaan zullen met eql globals variable.

POST http://httpbin.org/post

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

```

2 pm.response.to.have.status(200);
3 };
4
5 let jsonData = pm.response.json();
6 console.log(jsonData.args.isGood);
7
8 pm.test("isGood", function () {
9     pm.expect(jsonData.args.isGood).to.eql('true');
10 });
11 pm.test("isBad", function () {
12     pm.expect(jsonData.args.isBad).to.eql('false');
13 });
14
15 pm.test("currentDate", function () {
16     var jsonData = pm.response.json();
17     pm.expect(jsonData.json.currentDate).to.eql(pm.globals.get("currentDate"));
18 });
19

```

Body Cookies Headers (7) **Test Results (2/4)** Status: 200 OK Time

All Passed Skipped Failed

FAIL isGood | AssertionError: expected undefined to deeply equal 'true'

FAIL isBad | AssertionError: expected undefined to deeply equal 'false'

PASS currentDate

Test for permissions:

Test / POST Copy

POST http://httpbin.org/post

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

```

9 pm.expect(jsonData.args.isGood).to.eql('true');
10 };
11 pm.test("isBad", function () {
12   pm.expect(jsonData.args.isBad).to.eql('false');
13 });
14
15 pm.test("currentDate", function () {
16   var jsonData = pm.response.json();
17   pm.expect(jsonData.json.currentDate).to.eql(pm.globals.get("currentDate"));
18 });
19 console.log(jsonData.json.permissions[2])
20 pm.test("permissions", function () {
21   pm.expect(jsonData.json.permissions[2]).to.eql(4000);
22 });
23

```

Body Cookies Headers (7) Test Results (3/5) Status: 200 OK Tim

| All | Passed | Skipped | Failed |
|------|--------------------|--|--------|
| PASS | status code is 200 | | |
| FAIL | isGood | AssertionError: expected undefined to deeply equal 'true' | |
| FAIL | isBad | AssertionError: expected undefined to deeply equal 'false' | |
| PASS | currentDate | | |
| PASS | permissions | | |

Postman maakt ook gebruik van een framework namelijk Chai Assertion Library.

https://www.chaijs.com/api/bdd/#method_include

We zullen niet naar een specifieke plaats gaan zoeken van de object maar in het algemeen naar permission en we zullen include doen van 4000 en zoals u kunt zien lukt de test.

Test / POST Copy

POST http://httpbin.org/post

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

```

9 pm.expect(jsonData.args.isGood).to.eql('true');
10 };
11 pm.test("isBad", function () {
12   pm.expect(jsonData.args.isBad).to.eql('false');
13 });
14
15 pm.test("currentDate", function () {
16   var jsonData = pm.response.json();
17   pm.expect(jsonData.json.currentDate).to.eql(pm.globals.get("currentDate"));
18 });
19 console.log(jsonData.json.permissions[2])
20 pm.test("permissions", function () {
21   pm.expect(jsonData.json.permissions[2]).to.include(4000);
22 });
23

```

Body Cookies Headers (7) Test Results (3/5) Status: 200 OK T

| All | Passed | Skipped | Failed |
|------|--------------------|--|--------|
| PASS | status code is 200 | | |
| FAIL | isGood | AssertionError: expected undefined to deeply equal 'true' | |
| FAIL | isBad | AssertionError: expected undefined to deeply equal 'false' | |
| PASS | currentDate | | |
| PASS | permissions | | |

We kunnen gaan kijken of er een content-type beschikbaar is aan de hand van deze test.

The screenshot shows the Postman interface with a PUT test to `http://httpbin.org/put`. The test script is as follows:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Content-Type is present", function () {
6   pm.response.to.have.header("Content-Type");
7 });
```

The test results show two passed tests:

- PASS Status code is 200
- PASS Content-Type is present

Om na te gaan of content-type de juiste value heeft gaan we dit als volgt doen:

The screenshot shows the Postman interface with a PUT test to `http://httpbin.org/put`. The test script is as follows:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Content-Type is present", function () {
6   pm.response.to.have.header("Content-Type");
7   pm.expect(pm.response.headers.get("Content-Type")).to.eql('application/json');
8 });
```

The test results show two passed tests:

- PASS Status code is 200
- PASS Content-Type is present

Om na te gaan of deze test juist is moeten wij die is ook laten falen.

3 bron vermelding

3.1 Video

https://www.youtube.com/watch?v=VywxIQ2ZXw4&ab_channel=freeCodeCamp.org

https://www.youtube.com/watch?v=Qlvbc6klBOk&ab_channel=ValentinDespa

https://www.youtube.com/watch?v=AgBg0CMknfI&ab_channel=ValentinDespa

https://www.youtube.com/watch?v=iPJCutFXzQE&ab_channel=ValentinDespa

3.2 Gratis read only

https://drive.google.com/file/d/1JzwW9zwSBKii039_aOVC6USTLQzB_S/view

3.3 Udemmy 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>

3.4 Postman documentatie

<https://www.npmjs.com/package/newman>