Logo, company name

Description automatically generated**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](#_Toc87173757)

[1.1 benodigdheden 3](#_Toc87173758)

[2 Logboek 4](#_Toc87173759)

[2.1 Week 1 – 20/09/2021: 4 uur 4](#_Toc87173760)

[2.2 Week 2 – 27/09/2021: 3 uur 4](#_Toc87173761)

[2.3 Week 3 – 04/10/2021: 3 uur 5](#_Toc87173762)

[2.3.1 Eerste request met Postman: GET request 5](#_Toc87173763)

[2.3.2 Collection en variable 6](#_Toc87173764)

[2.3.3 Query parameters 7](#_Toc87173765)

[2.3.4 Path variables 9](#_Toc87173766)

[2.3.5 POST request en API authentication 10](#_Toc87173767)

[2.4 Week 3 – 05/10/2021: 1 uur 15](#_Toc87173768)

[2.4.1 Testen: Random test data 15](#_Toc87173769)

[2.4.2 PATCH request: bestelling bewerken 16](#_Toc87173770)

[2.4.3 DELETE request: bestelling verwijderen 17](#_Toc87173771)

[2.4.4 Writing API test 18](#_Toc87173772)

[2.4.5 Code tests 20](#_Toc87173773)

[2.5 Week 4 – 11/10/2021: 2 uur 20](#_Toc87173774)

[2.6 Week 4 – 12/10/2021: 2 uur 20 min 20](#_Toc87173775)

[2.6.1 Postman variables 20](#_Toc87173776)

[2.6.2 Extracting data from the response 21](#_Toc87173777)

[2.6.3 Collection runner 26](#_Toc87173778)

[2.6.4 Request execution order 26](#_Toc87173779)

[2.6.5 Postman monitors 27](#_Toc87173780)

[2.6.6 Newman 28](#_Toc87173781)

[2.6.7 HTML reports with Newman 31](#_Toc87173782)

[2.7 Week 5 – 18/10/2021: 2 uur 25 min 32](#_Toc87173783)

[2.8 Week 6 – 25/10/2021: 2 uur 32](#_Toc87173784)

[2.9 Week 7 – 07/11/2021: 2 uur 32](#_Toc87173785)

[3 bron vermelding 33](#_Toc87173786)

[3.1 Video 33](#_Toc87173787)

[3.2 Gratis read only 33](#_Toc87173788)

[3.3 Udemy cursus 33](#_Toc87173789)

[3.4 Postman documentatie 33](#_Toc87173790)

# 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 hoef je allee 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.

## benodigdheden

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

# Logboek

## 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.

## 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

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## 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

Graphical user interface, text, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

### 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.

Graphical user interface, text, application, email

Description automatically generated

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

**Read:**

Graphical user interface, text, application, email

Description automatically generated

### Collection en variable

Hier heb ik van de API een collection gemaakt

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

### 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)

Graphical user interface, text, application, email

Description automatically generated

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)

Graphical user interface, text, application, email

Description automatically generated

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)

Graphical user interface, text, application, email

Description automatically generated

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)

Graphical user interface, text, application, email

Description automatically generated

### Path variables

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

**Key :** bookId

**Value :** 2

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, application

Description automatically generated

De path variables kunnen we namelijk ook zo schrijven

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

Graphical user interface, text, application, email

Description automatically generated

### 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.

Graphical user interface, text, application, email

Description automatically generated

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.

Graphical user interface, text, application, email

Description automatically generated

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

Text, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

Postman heeft automatisch een nieuwe header aangemaakt.

Graphical user interface, application, email

Description automatically generated

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

Een bestelling doorgeven

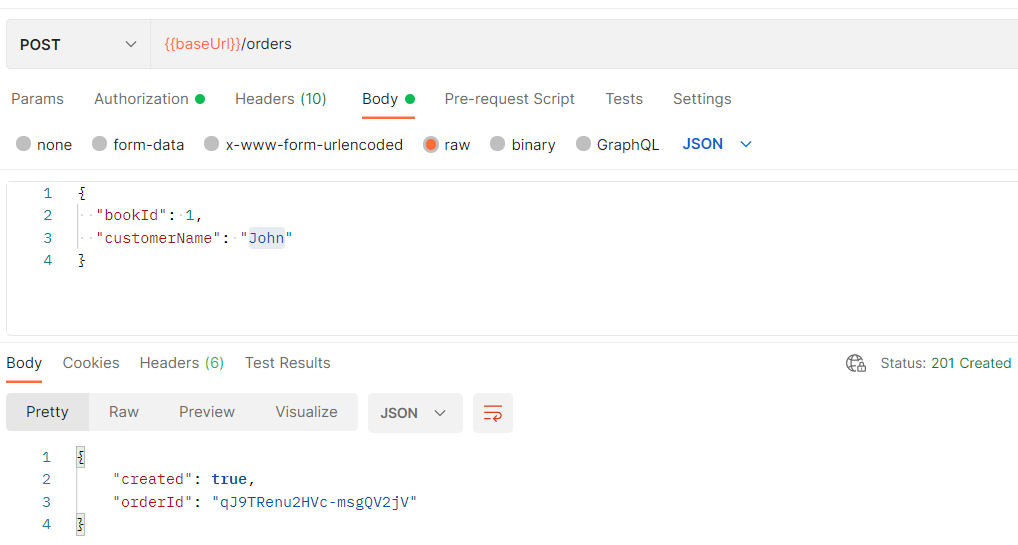
POST/orders

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



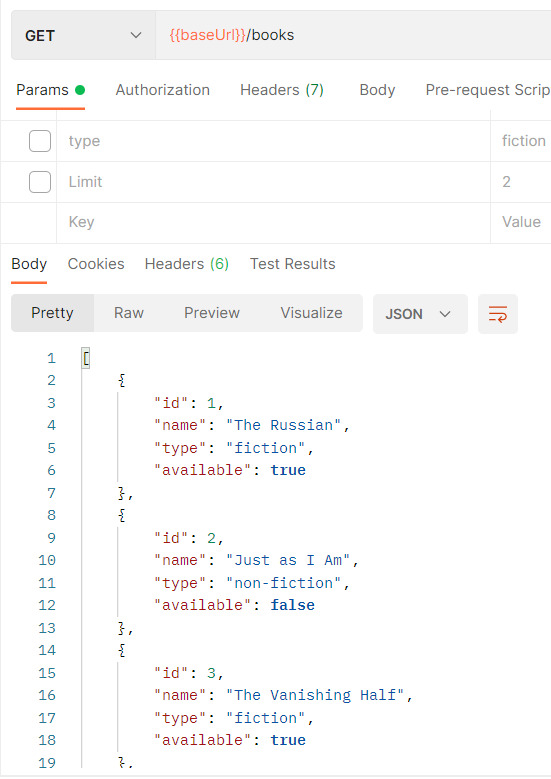
Als we een order maken van een boek dat niet meer in stock is zal deze bericht op onze scherm verschijnen.

Graphical user interface, text, application, email

Description automatically generated

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 avaible

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



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

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

Graphical user interface, text, application

Description automatically generated

## Week 3 – 05/10/2021: 1 uur

* Vervolg request met Postman: 1 uur

### 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

Graphical user interface, text, application, email

Description automatically generated

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

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

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

### 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

Graphical user interface, text, application, email

Description automatically generated

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

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

Graphical user interface, text, application, email

Description automatically generated

### DELETE request: bestelling verwijderen

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

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

### 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.

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

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.

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

Order books status: 201

Graphical user interface, text, application, email

Description automatically generated

### 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).to.eql("Ossk");

});

## Week 4 – 11/10/2021: 2 uur

* Gastcollege gevolgd: 2 uur

## 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>

### Postman variables

Ik heb eerst en vooral een globale variabel aangemaakt namelijk orderId

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

### 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])

Graphical user interface, text, application, email

Description automatically generated

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

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

Graphical user interface, text, application

Description automatically generated

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.

Graphical user interface, text, application, email

Description automatically generated

We kunnen de test ook laten falen.

Graphical user interface, text, application, email

Description automatically generated

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

A picture containing text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Maar als we deze weer activeren zullen alle testen lukken.

Graphical user interface, text, application, email

Description automatically generated

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.

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

### 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.

Graphical user interface, text, application, email

Description automatically generated

### Request execution order

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

postman.setNextRequest("List of books");

Graphical user interface, text, application, email

Description automatically generated

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**);

### Postman monitors

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

Graphical user interface

Description automatically generated with low confidence

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

### 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 geinstalleerd aan de hand van deze commando:

*$ npm install -g newman*

A screenshot of a computer

Description automatically generated with medium confidence

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”.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

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

Graphical user interface, application

Description automatically generated

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

Text

Description automatically generated

### HTML reports with Newman

Ik heb eerst deze commando uitvoerd 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.

Text

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

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

Graphical user interface

Description automatically generated

## Week 5 – 18/10/2021: 2 uur 25 min

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

## Week 6 – 25/10/2021: 2 uur

* Gastcollege 2 gevolgd: 2 uur

## Week 7 – 07/11/2021: 2 uur

* Writing API tests: 2 uur

# bron vermelding

## Video

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

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

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

https://www.youtube.com/watch?v=iPJCutFXzQE&ab\_channel=ValentinDespa

## Gratis read only

<https://drive.google.com/file/d/1JzwW9zwSBKiios039_aOVC6USTLQzB_S/view>

## 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>

## Postman documentatie

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