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](#_Toc84353656)

[1.1 benodigdheden 3](#_Toc84353657)

[2 Logboek 4](#_Toc84353658)

[2.1 Week 1 – 20/09/2021: 4 uur 4](#_Toc84353659)

[2.2 Week 2 – 27/09/2021: 3 uur 4](#_Toc84353660)

[2.3 Week 3 – 04/10/2021: 3 uur 5](#_Toc84353661)

[2.3.1 Eerste request met Postman: GET request 5](#_Toc84353662)

[2.3.2 Collection en variable 6](#_Toc84353663)

[2.3.3 Query parameters 7](#_Toc84353664)

[2.3.4 Path variables 9](#_Toc84353665)

[2.3.5 POST request en API authentication 10](#_Toc84353666)

[2.4 Week 3 – 05/10/2021: 3 uur 15](#_Toc84353667)

[2.4.1 Testen: Random test data 15](#_Toc84353668)

[2.4.2 PATCH request: bestelling bewerken 16](#_Toc84353669)

[2.4.3 DELETE request: bestelling verwijderen 17](#_Toc84353670)

[2.4.4 Writing API test 18](#_Toc84353671)

[3 bron vermelding 19](#_Toc84353672)

[3.1 Video 19](#_Toc84353673)

[3.2 Gratis read only 19](#_Toc84353674)

[3.3 Udemy cursus 19](#_Toc84353675)

# 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

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

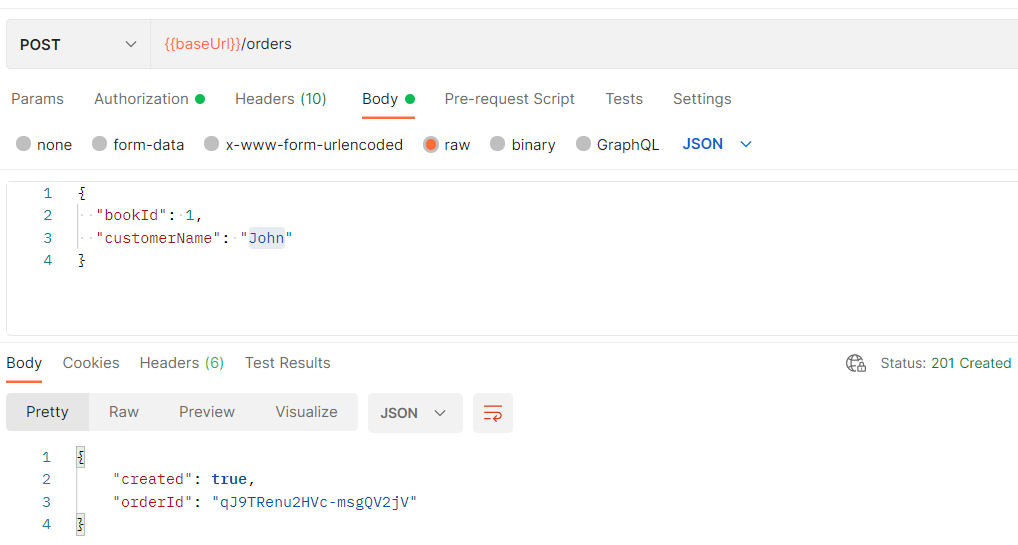
POST/orders

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



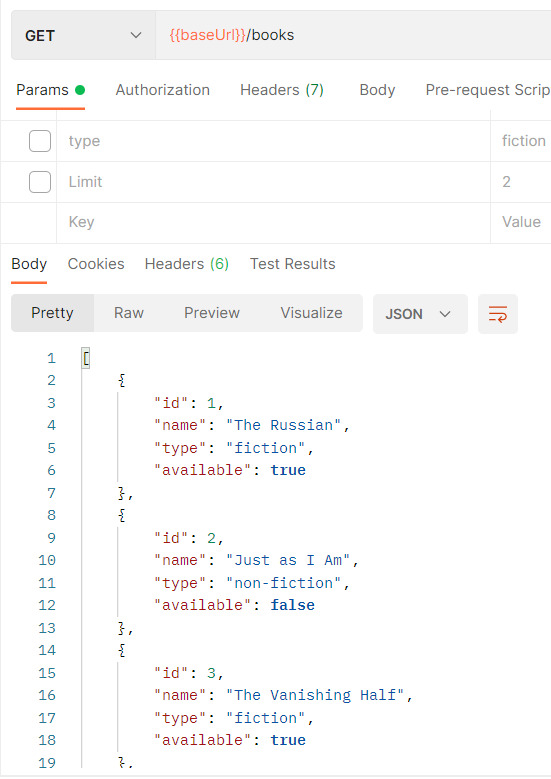
Als we een order maken van een boek dat niet meer in stock is zal deze bericht op onze scherm verschijnen.

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");

});

# bron vermelding

## Video

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

## 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>