**Toegepaste Informatica**

**Opdracht Software Testing**

*API Testing with POSTMAN*

**Nelly Nsengimana Ineza**

(r0699887)

Bachelor – Toegepaste Informatica

**Software Testing**

Frank Serneels

Academiejaar 2021-2022

Inhoud

# Onderzoek naar omgeving

Voor het kiezen van een testomgeving hadden we de keuze uit:

* …

Tijdens de eerste les kregen we uitleg over elke omgeving. Ik twijfelde tussen API tesing met Postman en testen met Selenium. De reden hiervoor is omdat ik met beide omgevingen al in aanraking ben geweest….

Ik heb dus om het maken van mijn keuze wat onderzocht op het internet.

Op basis van mijn voorkennis en vooral interesse koos ik voor API testing met Postman.

# Onderzoek naar tools

Voor het testen zal ik uiteraard Postman nodig hebben.

Voor het leren van (API) testen met Postman ben ik op zoek gegaan naar verschillende beschikbare tutorials/cursussen. Naast het document verkregen van de docent, heb ik een aantal andere links gevonden:

* www….

Deze links heb ik doorgenomen, alleen en samen met de docent, om te zien welke de beste een meest volledig zijn. Ik besloot dan dat ik volgende cursus ging gebruiken:

* www….

En volgende bronnen:

* www…

# Start theorie

Om te starten besloot ik om volgende Youtube video te volgen. Ik denk dat deze video zeker een goede start/opwarming was.

[THEORIE VIDEO]

Basic:

API = interface to a server that has some data of does some actions

Postman: tool for interacting with web-based API’s

Before using an API, understand how to interact with that API by looking at the API documentation.

Endpoint: offer different kind of responses (eg “status”)

Collection: list of multiple request (typically using the same API)

Er zijn verschillende statussen. De belangrijkste om te kennen zijn, status 200 (OK), 201 (Created), 400 (Bad request), 404 (Not found), 401 (Unauthorized)

Data wordt geformatteerd in JSON.

Variabelen aanmaken: je kan variabelen aanmaken waarin je de basis url bijhoudt. Vervolgens kan je dan deze variabele gebruiken en koppelen met de verschillende endpoints zonder de url steeds te moeten herschrijven. Deze variabele bewaar je in een bepaalde scope (global, collection,…) dus let op! Als je een variabele aanmaakt in een bepaalde collection, ga je deze niet kunnen gebruiken zolang query niet in uw collection is opgeslagen.

Les 7: query parameters = additional data that you can submit with your query. Deze kunnen optioneel of verplicht zijn afhangend van de API. Je kan meerdere paramters hebben bij een query. Hierbij heb je een key en een value. Hier in mijn voorbeeld is de key “type” van boeken en de value is “fiction”.

A screenshot of a computer screen

Description automatically generated with medium confidence



De bedoeling is dus dat in mijn respons enkel fictieve boeken terug krijg.

Graphical user interface, text, application

Description automatically generated



Je kan weten wat je kan gebruiken als query parameters enkel door naar de documentatie van de API te kijken. Anders had ik hier in dit voorbeeld niet kunnen weten dat “fiction” een type was, maar had ik misschien “thriller” gebruikt (maar volgens de API documentatie is dit geen type dus had ik een fout gekregen met status 400 Bad request).

Ik had ook bijvoorbeeld een parameter kunnen uitvinden (key=blabla, value=test). Het uitvinden van deze parameter had niets veranderd aan de respons. Postman zou dus in deze situatie de uitgevonden parameter negeren en gewoon de repons sturen zonder rekening te houden met deze parameter (met status code 200 OK).

In mijn voorbeeld zijn er maar 2 optionele parameters beschikbaar, nl. “type” en “limit”.

Opletten: het is case-sensitive! “Limit” gaat dus niet de verwachtte respons teruggeven, maar “limit” wel.

Les 9: Path variables: this endpoint allows to specify a variable which represents one of the books that you’re trying to get detail information from (eg. “key=bookId” en “value=1”), instead of having it in a list with less information.

Bij path variables is het niet de key die wordt verzonden maar de value. Hier maakt het dus niet uit of je als key “bookId” kiest of “id” of iets anders. Wat verzonden zal worden is de “1”.

Les 10: post request

AUTHORIZATION

Acces token: soort van een tijdelijk paswoord dat je krijgt en kan gebruiken bij uw request waarbij authorization is. Deze token sla je best op bij current value (niet initial value omdat de kans groot is dat andere mensen dit zien wat je niet wilt).

Les 17: PATCH: om te updaten

Met patch kan je een body meegeven, hierin geef je de content mee die je wilt toevoegen

Les 18: DELETE

Les 19: test automation

Kortom, test automation is een techniek voor het testen van software om het werkelijke resultaat te testen en te vergelijken met het verwachte resultaat. Om dit te bereiken, schrijven testers testscripts en automatiseren ze die met behulp van een testautomatiseringstool (tests die steeds opnieuw kunnen worden uitgevoerd). U kunt testautomatisering gebruiken om repetitieve tests te automatiseren of tests die moeilijk handmatig uit te voeren zijn. Goede automatisering maakt testen sneller, systematischer, en vermindert menselijke fouten. Het doel van test automation is om de effectiviteit en efficiëntie van testen te verhogen.

Postman kan worden gebruikt om vele soorten tests te automatiseren, waaronder unit tests, functionele tests, integratietests, end-to-end tests, regressietests, mock tests, enz. De bedoeling is dus om Postman de testen te laten doen, en wij moeten er enkel tussenkomen wanneer er iets fout loopt.

API automatisering laat u de core logica valideren zonder afhankelijk te zijn van de UI. API testen zijn eenvoudig te automatiseren binnen dezelfde sprint. Met API automatisering kunt u al vroeg beginnen met het testen van uw applicatie - zelfs zonder een UI. Dit helpt om problemen vroeg in de ontwikkelingscyclus te identificeren en op te lossen die anders duur zouden zijn om op te lossen wanneer ze tijdens het testen van de UI zouden worden geïdentificeerd.

# Eerste testen

Eerste request met Postman

A screenshot of a computer

Description automatically generated with medium confidence

Eerst testen schrijven:

Bovenaan kan je een rubriek “tests” vinden. Hier is waar je testen kan schrijven om testen te automatiseren.

Graphical user interface, application

Description automatically generated

Postman zet ook “snippets” ter beschikking. Dit zijn voorafgemaakte typische testjes die je kan gebruiken. Een typische test is bijvoorbeeld nakijken of de status code 200 OK is. Hiervoor moet je deze test dus niet schrijven, maar ken je deze simpelweg van de snippets halen.

Text

Description automatically generatedGraphical user interface, text

Description automatically generated with medium confidence

Bij het verzenden van uw request gaat Postman dus nu ook deze test uitvoeren. Het resultaat kan je hier vinden:

Graphical user interface, application

Description automatically generated

Nu ga ik een test laten falen om te zien hoe dit resultaat eruit ziet:

Graphical user interface, application, website

Description automatically generated

We zien dus dat het verwachtte resultaat “status code is 200” is. Naast dit verwachtte resultaat krijgen we een *AssertionError*. Hierbij staat er dus wat er is gebeurt (waardoor de test is gefaald).

Nu gaan we naar de volgende stap: het is ook belangrijk dat je de response body kan nakijken. Meestal zal deze in JSON zijn. We gaan nu een test schrijven die nakijkt dat de status effectief “OK” is (in de body). We gaan hiervoor de response *parsen*. Waarom? Omdat de response body in JSON staat maar eigenlijk in ‘tekst’. Wij willen dit als een object in javascript (en dus niet tekst). We gaan dus een (javascript) variabele definiëren. “pm.repsponse” houdt de response bij. “json()” is een functie van Postman om de response van json naar javascript te parsen. Deze zal dus in de variabele “reponse” worden bijgehouden.

Graphical user interface, text

Description automatically generated

Om enkel de waarde terug te krijgen, dus hier enkel de “OK” i.p.v. “status: OK”, kan je “console.log(response.status)” schrijven i.p.v. “console.log(response)”.

Het resultaat in de console lijkt misschien op JSON, maar het is eigenlijk een javascript object.

Text

Description automatically generated

Nu ga ik zelf eens een test schrijven (en geen voorgemaakte test van de snippets). Omdat dit mijn eerste zelfgeschreven test is ga ik het heel simpel houden. Ik ga testen of “1” gelijk is aan “1”.

Graphical user interface, text, application

Description automatically generated

Een test start je altijd met “pm.test” in Postman. Hierbij zijn er 2 parameters. Als eerste parameter schrijf ik de naam van de test (kies je zelf) en als tweede parameter (na de komma) schrijf ik de function. Het resultaat hiervan is te verwachten, en zou dus “PASS” moeten zijn. Bij het verzenden zie ik dat deze test effectief geslaagd is.

Une image contenant texte

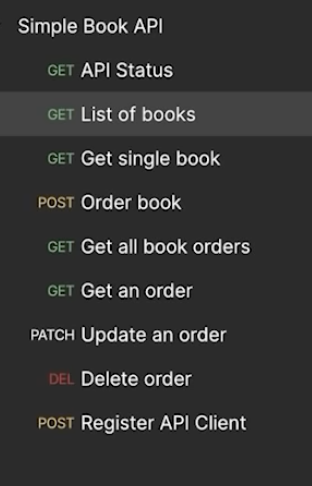
Description générée automatiquement

Ik ga nu “eql(1)” eens vervangen door “eql(2)” om uit te testen. 1 is niet gelijk aan 2, dus deze test zou moeten falen. Laten we eens zien:

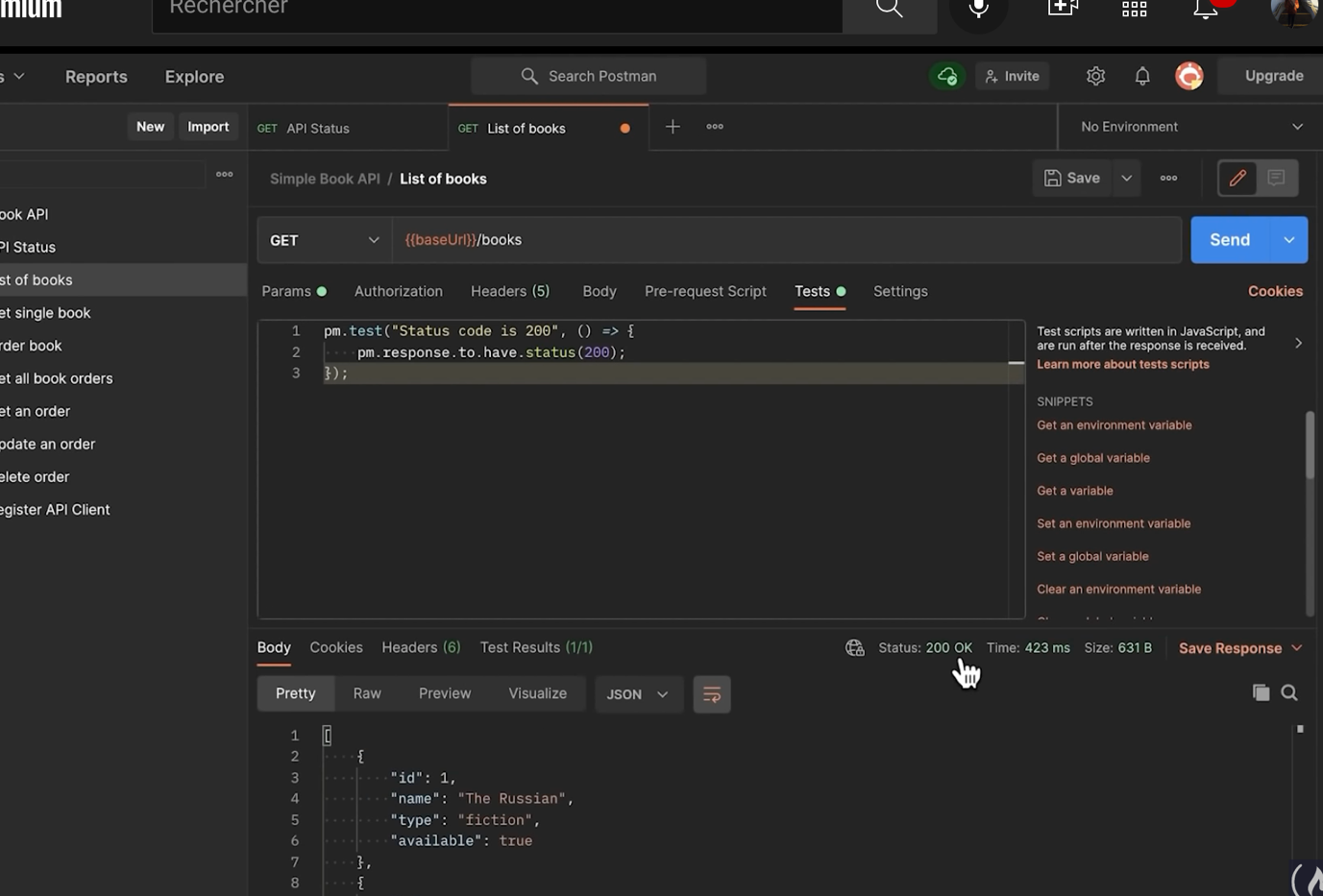
Une image contenant texte, capture d’écran, moniteur, écran

Description générée automatiquement

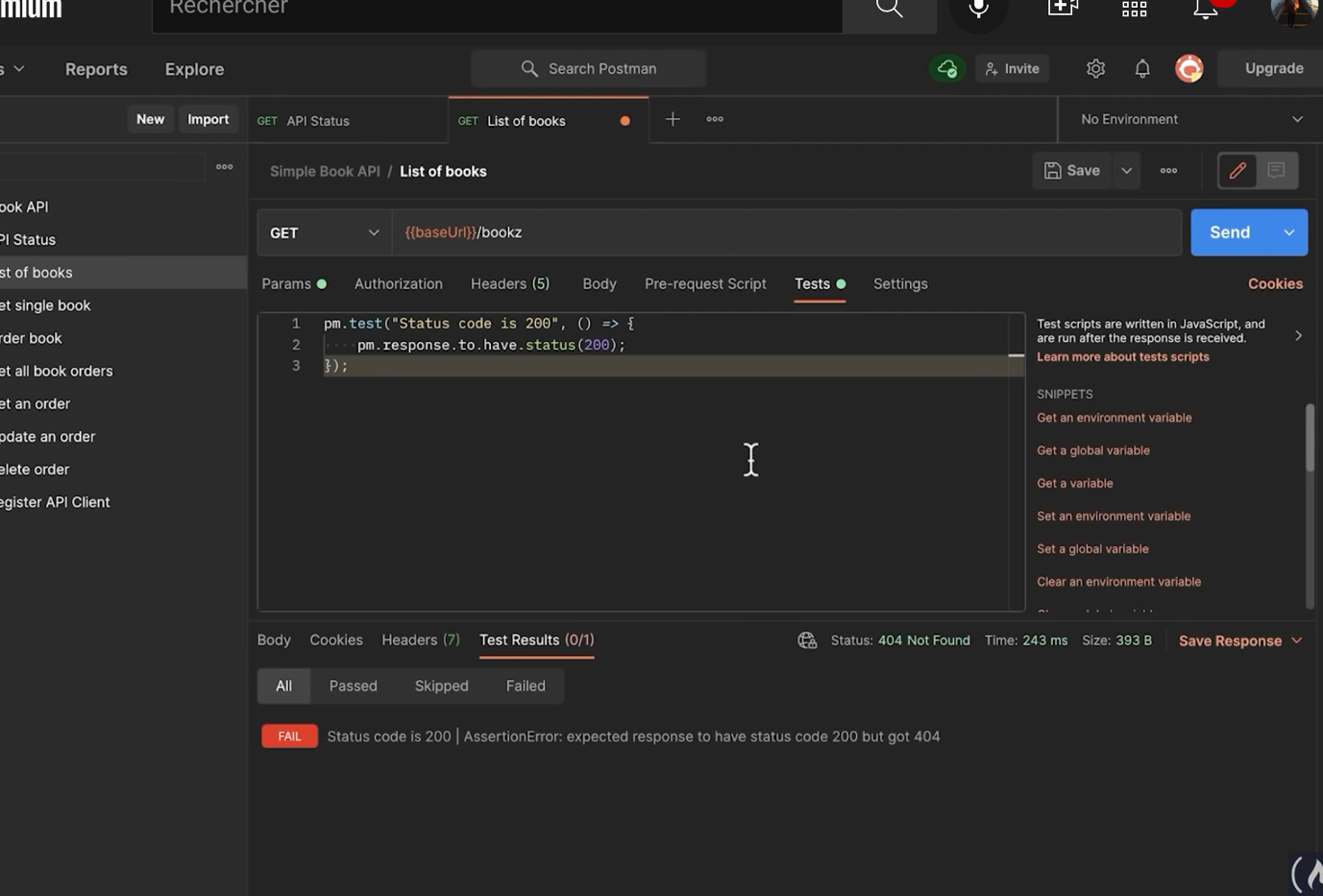
Oefening: als oefening ga ik een status code test schrijven voor alle requests in de collection. Deze testen moeten ook kunnen falen:



Om te starten vertrok ik van de code die ik al had vanuit de eerste test. Hierbij moest ik maar een kleine aanpassing maken. Ik begon eerst met de GET list of books request. Deze lukte zonder probleem.



Ik heb dan “books” vervangen door “bookz” om te zien of deze test zou falen. De test faalde effectief.



Ik deed vervolgens de test voor de GET single book request. Dit verliep op dezelfde manier als de vorige. Voor de POST order book request was het wat anders. Ik deed eerst hetzelfde als bij de eerste 2 testen. Als resultaat van de test kreeg ik echter een “fail”. Dit komt omdat mijn status code niet “200 OK” was, maar “201 CREATED”. Ik heb dan in mijn test “200” vervangen door “201”. Nu lukte de test wel.

Une image contenant texte, moniteur, capture d’écran, noir

Description générée automatiquement

Bij de PATCH request was het gelijkaardig als de POST. In de plaats van “200” of “201” was het hier dan “204”.

Wat sterk opvallend was, was dat ik vaak moest kopieëren en plakken bij het schrijven en uitvoeren van deze testen. De testen hadden allemaal dezelfde principe. Dit was uiteraard wat vervelend. Hier komt testing automation dan!

# Logboek

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software Testing: Logboek | | |  | |
|  | | | | |
| Datum | Beschrijving | Bron | | Commentaar |
| 20/09/2021 | Zoeken welke bronnen/courses ik zal gebruiken | Verschillende bronnen: links zie document Github | | Ik heb verschillende bronnen gevonden die zeer interessant lijken. Ik moet echter nog dieper inzien welke *volledig* zijn voor dit vak. |
| 27/09/2021 | Bronnen/courses uitkiezen | Verschillende bronnen: links zie document Github | | Na bespreking met de docent heb ik geselecteerd welke bronnen ik effectief zal gebruiken. |
| 27/09/2021 | Installatie Postman | https://www.postman.com/downloads/ | | Postman was al geïnstalleerd op mijn laptop. Ik heb deze nu ook geïnstalleerd op mijn desktop. |
| 29/09/2021 | Eerste bron bestuderen | https://www.youtube.com/watch?v=VywxIQ2ZXw4&t=1905s | | Ik heb de video deels bekeken (tot 01:01:40). Ik kon volgen en had geen moeilijkheden. |
| 08/10/2021 | Bovenstaande bron verder bestuderen/afwerken | https://www.youtube.com/watch?v=VywxIQ2ZXw4&t=1905s | | Ik heb de video afgekeken. Het 2e deel was wat “complexer”, maar het viel mee. |
| 08/10/2021 | Eerste tests in Postman | / | | Met behulp van de bovenstaande tutorial heb ik de eerste testen kunnen uitvoeren. Dit is gelukt. |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |