

**ASP.NET Core Web API testen met Postman**

Deneyer Sander

3 Toegepaste Informatica

Contents

[Intro 3](#_Toc85721847)

[Setup 3](#_Toc85721848)

[Postman: The Complete Guide – REST API Testing 3](#_Toc85721849)

[Introduction and first steps in Postman 3](#_Toc85721850)

[Creating with API requests 4](#_Toc85721851)

[Requests 4](#_Toc85721852)

[Responses 6](#_Toc85721853)

[Saving requests 7](#_Toc85721854)

[Writing tests and scripts 8](#_Toc85721855)

[Een eerste simpele test 8](#_Toc85721856)

[De tabel Categories testen 9](#_Toc85721857)

[Writing tests and scripts using variables 12](#_Toc85721858)

[API workflow using Github & Twilio 12](#_Toc85721859)

[Advanced assertions 12](#_Toc85721860)

[Automatically running tests 12](#_Toc85721861)

[Running Newman with other CI servers/tools 12](#_Toc85721862)

[Workflows and Scenarios 12](#_Toc85721863)

[Data driven tests: Running a request multiple times with different data sets 13](#_Toc85721864)

[Team Collaboration 13](#_Toc85721865)

[Mock servers 13](#_Toc85721866)

[File uploads (testing, automatic uploads, uploading multiple files) 13](#_Toc85721867)

[Authentication / Authorization 13](#_Toc85721868)

[Advanced topics and user questions 13](#_Toc85721869)

[New features in Postman 13](#_Toc85721870)

[JavaScript fundamentals 13](#_Toc85721871)

[Postman Developer Challenge 2021 13](#_Toc85721872)

[Bonus section 13](#_Toc85721873)

[Timetable 14](#_Toc85721874)

# Intro

Als opdracht voor het vak software testing moeten we zelf onze test omgeving zoeken en wat we willen testen. Na even te denken heb ik ervoor gekozen om een REST-API te tesen via het programma postman. Ik heb hiervoor geopteerd door enkele zaken.

Ten eerste is het testen van API’s in postman nieuw voor mij. Alhoewel ik postman zelf al eens heb gebruikt was dit maar vlugtig en niet zo uitgebreid zoals ik het nu wil doen.

Ten tweede kan ik zo een eerder geschreven API testen dat we voor het vak ‘Advanced Programing Concepts’ moesten maken. Doot dit te doen kan ik mijn eigen werk testen, en dus zien hoe goed deze taak was. En kan ik eventueel ook nog fouten uit het programma halen zodat ik hieruit kan leren zowel voor dit vak als het andere.

Om dit allemaal te kunnen leren heb ik ervoor gekozen om een cursus aan te kopen op Udemy (Postman: The Complete Guide – REST API Testing). De weg dat ik zelf afleg tijdens deze cursus wordt hieronder beschreven.

# Setup

Om een API te kunnen testen heb ik natuurlijk een API nodig en een API test platform. Voor de API gebruik ik een al eerder door mezelf gemaakt programma. En het test platform zal Postman zijn.

Toen ik eerst postman wou installeren, installeerde het als een app. Geen kwaad zo je denken, maar als je dit dan opende dan ging het naar een web based platform dat ik nogal omslachtig vond. Maar omdat ik al eens eerder had gebruik gemaakt van postman wist ik dat er een een echte applicatie van bestond. En na altijd geredirect te worden naar de app vond ik het ineens. Hierna was het gewoon gemakkelijk downloaden en snel installeren.

# Postman: The Complete Guide – REST API Testing

## Introduction and first steps in Postman

In de cursus content hebben ze eerst eerst een overview gegeven van wat we kunnen verwachten van de cursus en wat we niet/wel kunnen verwachten van dit.

Hierna wordt er overlopen hoe je een account kunt maken en hoe je postman moet downloaden. Eens postman is opgestart wordt het programma getoond en wordt er al direct een simpele request gemaakt. Eens we het antwoord hebben gekregen van de request wordt hier van alles over gezegd.

Hierna werd het al een beetje complexer en heeft hij uitgelegd wat parameters zijn en hoe je ze kan gebruiken.

Een request (met bepaalde parameters) kan ook altijd opgeslagen worden via postman. Maar dit kan niet zomaar we moeten eerst een collection maken om dit in op te slaan.

Een request kan meer dan alleen een GET zijn, hij heeft ook al getoond hoe je met postman een POST request kan maken, opstellen en weer oplsaan.

Hierna begon hij met iets belangrijks uit te leggen, hoe je via variabelen bepaalde data uit een response body kan gebruiken in een ander request zoals bv een post. Dit gebeurdt aan de hand van code.

Tenslotte heeft hij ook nog getoond hoe je gemakkelijk een simpele test kan opstellen via de snippets.

Via de Newman kan je bepaalde requests van één collection laten runnen na elkaar. Dit zorgt ervoor dat je verschillende requests/testen kunt combineren in een workflow.

Op het einde van de Intro heeft hij ook nog in het kort uitgelegd wat automated testing is.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Creating with API requests

### Requests

Een htpp request bestaat uit vier delen:

* Een request methode
* Een url waar het request moet naartoe gaan
* Headers
* Body

Graphical user interface, text, application, Teams

Description automatically generatedGraphical user interface, text

Description automatically generated

De request methode bij Postman zijn makkelijk aan te duiden, de voornaamste zijn GET, POST, PUT, DELETE. Over alle andere methodes wordt momenteel nog niks over gezegd omdat dit direct een ander niveau dan wat ze ons proberen te leren in deze cursus.

Een gemakkelijk feature van postman is dat het de vorige url adressen onthoud dat je al hebt gebruikt. Hierdoor kan je dus snel werken zonder steeds de juiste url op te hoeven zoeken.

Headers zijn ook een deel van het request. Er zijn al standaard headers dat postman aanbied maar je kan er ook nog altijd zelf meegeven.

Onder body kan je verschillende opties selecteren. 

Hierin kan je verschillende waardes meegeven aan uw request. En dit zijn dan eigenlijk post requests. Je kan bijvoorbeeld voor een Voornaam & Achternaam meegeven. Om gemakkelijk te zien wat we gaan meegeven gaan we werken met Hookbin. Hiermee gaan we kunnen zien wat we allemaal doorsturen.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email, website

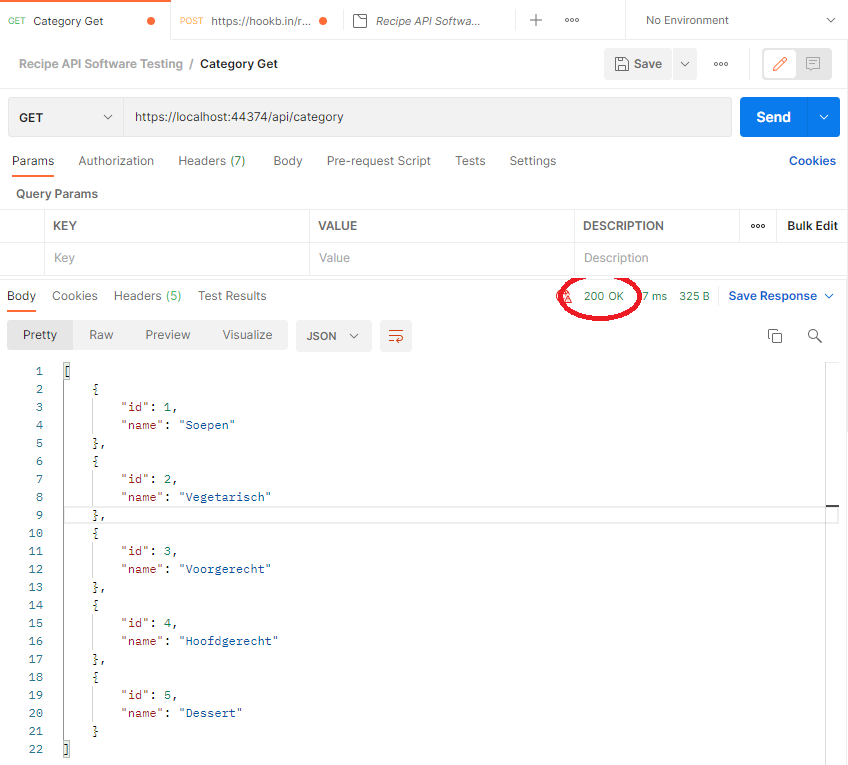
Description automatically generated

Je kan dus zien op de eerste screenshot dat ik via postman een PUT heb gedaan met een voornaam en achternaam in de Body. Op de tweede screenshot kan je daadwerkelijk zien dat de website Hookbin dit heeft ontvangen.

Het verschil tussen de form-data optie en de x-www-form-urlencoded is dat hetgeen wat je doorstuurd web encoded wordt. Dit betekent dat bijvoorbeeld spaties worden encoded. Bij Raw kan je kiezen dat je in Tekst schrijft, JavaScript, JSON, HTML of XML.

### Responses

De response is eigenlijk het gedeelte dat de api antwoord. Wanneer de code begint met een twee, dan weet je dat alles oké is gegaan. Wanneer de server code terugstuurt dat begint met een 4 dan weet je dat je iets fout hebt gedaan. En wanneer het begint met een 5 dan is er een probleem met de server.

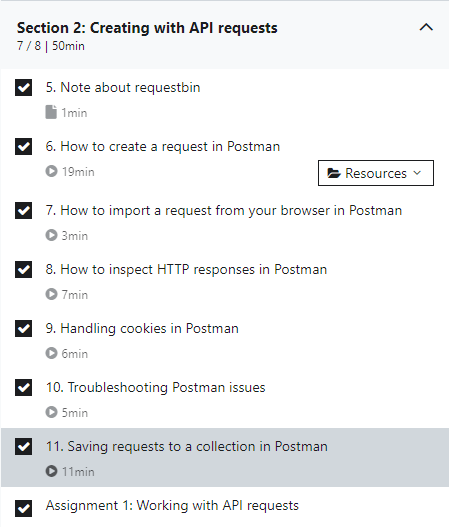


Zoals je nu kan zien hebben we een simpele GET request gemaakt waar we vragen achter /category. En in het programma staat dit gecodeert dat het een get all returnt. Je ziet dus 5 verschillende gerechten dat standaard waardes zijn. Wat je ook kan zien is dat de code 200 werd teruggegeven dus alles is in orde en het duurde 77ms om een antwoord te krijgen. Je kan het antwoord ook in verschillende formaten laten weergeven zoals JSON, XML, HTML en Tekst. En ten slotte kan je de response ook nog opslaan. Wat ook handig is dat je naast de response code kan zien hoe groot de response is.

### Saving requests

Text

Description automatically generatedJe kan verschillende requests opslaan door ze in een collectie te zetten. Een collectie bestaat normaal uit verschillende bijhorende requests. Zelf heb ik al een collectie gemaakt voor dit vak met al één request in dat ik hierboven heb getoond. Om requests nog beter te beheren kan je ook folders maken in een collectie maar dit is nog niet toepasselijk voor mij.



## Writing tests and scripts

### Een eerste simpele test

Wat is een test? Een test is eigenlijk een bewering waar je de response controleert tegen over een verwacht antwoord, bv een status code dat dan liefst 200 is.

Een zeer simpele test zou dit kunnen zijn:

Graphical user interface, text, application

Description automatically generated

Dit is een lijkt een zeer simpele test, maar toch kwam ik al snel op een probleem. Er kon geen request worden verzonden.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedNa een tijdje zoeken op het internet en zelf te reflecteren wat er mis was, kwam ik tot het besef dat ik zelf was vergeten mijn API te laten runnen. Dit moet elke keer gebeuren voor ik begin met testen anders kan er natuurlijk niets getest worden.

Nu kunnen we de test nog eens proberen te doen. En succesvol, bij de rechtse afbeelding kan je bij ‘Test Results’ zien dat de test is geslaagd en de status code echt 200 is.

Wanneer bijvoorbeeld de request url verkeerd geschereven is en dus niet bestaat zal je natuurlijk een ander resultaat moeten zien. De test zou moeten falen en zoals je kan zien gebeurt dit ook.

Graphical user interface, text, application

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

Description automatically generated

Maar wat betekend deze test code nu? pm.test is een fuctie voor het schrijven van test specificaties. Tussen de haakjes staat een eerste parameter, dit is een string. Na de comma sraat een tweede parameter, dit is een call back funtie (gebeurt pas na het uitvoeren van de test) dat kijkt op de response status code in dit geval 200 is.

### De tabel Categories simpel testen

#### De data bekijken.

Om deze tests te kunnen doen gaan we eerst is kijken naar de data dat nu al in de database staat. Dit kan gemakkelijk met Visual Studio en de ingebouwde “SQL Server Object Explorer”.

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated

Zoals we kunnen zien is er nu enkel data in de tabel ‘Catagories’. We zullen nu eens testen of we alleen de informatie het eerste categorie kunnen weergeven. Hierin zou dus ‘Soepen’ moeten staan.

#### Eén categorie weergeven

Graphical user interface, text, application

Description automatically generated

Dit is de test, je ziet dat de request URL is aangepast en achter ‘category’ het getal ‘1’ staat. Dit staat dus voor het id van een categorie dat we willen zien.

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, application

Description automatically generated

Zoals je kan zien is deze test geslaagd en hebben we Soepen teruggekregen.

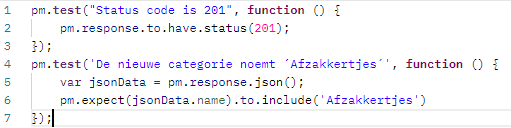
#### Een categorie toevoegen

Een categorie kan makkelijk toegevoegd worden door een beetje json code te schrijven in de body van de request. Het ziet er als volgt uit.

Graphical user interface, text, application, email

Description automatically generated

Hierin kan je zien dat we de catgorie ‘Afzakkertjes’ willen toevoegen. Nu moeten we nog juist de Tests toevoegen.



Eerst gaan we kijken of de POST is gelukt, dit kan via de status code ‘201’. Hierna gaan we kijken of er in de response bij het geelte ‘name’ ‘Afzakkertjes’ staat. Eens we de POST uitvoeren kunnen we de volgende body zien.

Graphical user interface, text, application

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

Description automatically generated

We kunnen ook al direct zien dat 2/2 tests zijn gelukt. En hier nog een diepere blik in kan ons bevestigen dat beide testen geslaagd zijn.

#### Een categorie verwijderen

Vooraleerst we een categorie gaan verwijderen gaan we eerst een nieuwe categorie toevoegen om te verwijderen met bovenstaande code.

A picture containing chart

Description automatically generated

Nu dit is gelukt kunnen we beginnen aan de DELETE. Deze soort request lijkt heel veel op het opvragen van één categorie. Het enige wat we eigenlijk moeten veranderen is de GET naar DELETE

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

Als de test geslaagd is zouden we de status code 204 moeten terugkrijgen.

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

Zoals je kan zien is de test gelukt maar hebben we geen body teruggekregen. Dit is wel normaal. Want wat zouden we moeten terug krijgen wanneer we iets verwijderen. Een status code geeft al meer dan genoeg weer, wanneer we weten of het met succes is verwijdert weten we al genoeg.

## Writing tests and scripts using variables

## API workflow using Github & Twilio

## Advanced assertions

## Automatically running tests

## Running Newman with other CI servers/tools

## Workflows and Scenarios

## Data driven tests: Running a request multiple times with different data sets

## Team Collaboration

## Mock servers

## File uploads (testing, automatic uploads, uploading multiple files)

## Authentication / Authorization

## Advanced topics and user questions

## New features in Postman

## JavaScript fundamentals

## Postman Developer Challenge 2021

## Bonus section

# Timetable

|  |  |  |
| --- | --- | --- |
| **Wat?** | **Tijdsduur** | **Datum** |
| Github in orde zetten | 30min | 4/10/2021 |
| Gepaste API zoeken + verslag | 30min | 5/10/2021 |
| Introduction and first steps in Postman | 55min | 5/10/2021 |
| Intro uitschrijven | 10min | 11/10/2021 |
| Creating with API requests (Requests) | 45min | 13/10/2021 |
| Creating with API requests (Vervolg) | 40min | 14/10/2021 |
| Een eerste test | 65min | 20/10/2021 |
| De tabel Categories simpel testen | 75min | 21/10/2021 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |