|  |
| --- |
|  |
| **Adrien Taneko**  **R0780417**  **Groep1-**2 |
| Tweede Bachelor - Toegepaste Informatica  Choose an item. |
|  |
|  |
| **Softaware testing**  Frank Serneels |
|  |
| Academiejaar 2021-2022 |

|  |  |
| --- | --- |
| Documentatie testomgeving | |
| REST API testing met Postman |

Inhoudstafel

[Inleiding 3](#_Toc89300646)

[Labo 1 - Inleiding tot API-verzoeken 4](#_Toc89300647)

[Labo 2 - Meer verzoeken en testen 5](#_Toc89300648)

[Labo 3 - Dynamische verzoeken met gebruik te maken van variabelen. 9](#_Toc89300649)

[Labo 4- Assertions 12](#_Toc89300650)

[Quiz 12](#_Toc89300651)

[Labo 5 – Introduction to Newman 14](#_Toc89300652)

[Newman installatie 14](#_Toc89300653)

[Uitvoeren van geëxporteerde collecties+environment met Newman 14](#_Toc89300654)

[Installatie + intro testen van jenkins omgeving 15](#_Toc89300655)

[Collectie uitvoeten met jenkins 16](#_Toc89300656)

[Rapporten generen met newman 18](#_Toc89300657)

[Collectie uitvoeren vanuit GIT repository 19](#_Toc89300658)

[Labo 6 – Automated testing with workflows and scenarios 20](#_Toc89300659)

[Sectie 10: Data Driven tests. 23](#_Toc89300660)

[Sectie 11: Team collaboration 23](#_Toc89300661)

[Sectie 12: Mock Servers 24](#_Toc89300662)

[Sectie 13: File Upload with Postman. 25](#_Toc89300663)

[Sectie 14: Authentication/authorization 25](#_Toc89300664)

[Authenticatie en autorisatie met basis access auth 25](#_Toc89300665)

[Autorisatie met Code Grant (OAuth 2.0) 26](#_Toc89300666)

[Form-based Authentication 27](#_Toc89300667)

[JSON Web tokens JWT-tokens 27](#_Toc89300668)

# Inleiding

Voor de vak software testing zijn we gevraagd om een ​​testomgeving naar keuze te onderzoeken en te documenteren.

Deze document bevat de documentatie van een REST-API testing met postman.

# Labo 1 - Inleiding tot API-verzoeken

*Open a new tab in Postman and create a GET request with this URL:*

*https://api.nytimes.com/svc/books/v3/lists.json*

*Submit the request.*

**What is the status code that you get back?**

{

"fault": {

"faultstring": "Failed to resolve API Key variable request.queryparam.api-key",

"detail": {

"errorcode": "steps.oauth.v2.FailedToResolveAPIKey"

}

}

}

*According to the error message you have received, the API is expecting your API key as a query parameter. The error that you see in the response body also indicates that is wrong.  
  
Go ahead and create a new query parameter called****api-key****and put the API key you have received in the Value field.*

*Now resubmit the request.*

**What status code do you get now?**

{

    "status": "ERROR",

    "copyright": "Copyright (c) 2021 The New York Times Company.  All Rights Reserved.",

    "errors": [

        "No Parameters specified\n",

        "Bad Request"

    ],

    "results": []

}

Try adding a GET query parameter called **list** with any value you want.

Resubmit the request.

**What status code do you see now?**

{

    "status": "OK",

    "copyright": "Copyright (c) 2021 The New York Times Company.  All Rights Reserved.",

    "num\_results": 15,

    "last\_modified": "2021-10-06T22:12:45-04:00",

    "results": [

}

# Labo 2 - Meer verzoeken en testen

*Using trello's online API documentation (https://developer.atlassian.com/cloud/trello/rest/)****build another request which creates a new list called TODO****inside your already created board.*

***Write tests to check that the request performed as expected.****As a minimum, please test the following:*

*- status is 200  
- the name of the list is TODO  
- should not be closed*

Eerst een bord aanmaken, en gebruik zijn id om de TODO-lijst erin te kunnen invoegen

(POST)

pm.test("Status code is 200", *function* () {

    pm.response.to.have.status(200);

});

*const* jsonData = pm.response.json();

pm.test("Check board name", *function* () {

    pm.expect(jsonData.name).to.eql(`Trello board`);

});

*let* id = jsonData.id;

pm.global.set("board\_id", id);

Dan TODO lijst aanmaken, (POST)

pm pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Check list name", function () {

var jsonData = pm.response.json();

pm.expect(jsonData.name).to.eql("TODO");

});

pm.test("Check list is not closed", function () {

var jsonData = pm.response.json();

pm.expect(jsonData.closed).to.eql(false);

});

let id = pm.response.json().id;

pm.globals.set("list\_TODO\_id", id);

*Additionally****create a new list****called****DONE****inside your already created board (in a similar way you have created the previous one).*

***Write tests to check that the request performed as expected.****As a minimum, please test the following:*

*- status is 200  
- the name of the list is DONE  
- should not be closed*

(POST)

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

pm.test("Check list name", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.name).to.eql("DONE");

});

pm.test("Check list is not closed", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.closed).to.eql(false);

});

pm.test("List is created in API\_test board", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.idBoard).to.eql

    (pm.globals.get("board\_id"));

});

let id = pm.response.json().id;

pm.globals.set("list\_DONE\_id", id);

*Inside the TODO list,****create a new card****(which is like a task) with the name****Learn Postman.***

*Write tests to check that the request performed as expected.*

*As a****minimum****, please test the following:*

*- status is 200  
- the name of the card is Learn Postman  
- should not be closed*

***Optional****:*

*- test that the card was created in the desired TODO list*

(POST)

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

pm.test("Check card name", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.name).to.eql("Learn Postman");

});

pm.test("Check card is not closed", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.closed).to.eql(false);

});

pm.test("card is created in TODO list", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.idList).to.eql

    (pm.globals.get("list\_TODO\_id"));

});

let id = pm.response.json().id;

pm.globals.set("card\_learnpostman\_id", id);

***Move the card****from the TODO list to the DONE list.*

*Write tests to check that the request performed as expected.*

*Please test the following:*

*- status is 200   
- the name of the card is still Learn Postman  
- test that the card was moved to the desired DONE list*

(PUT)

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

pm.test("Check card name", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.name).to.eql("Learn Postman");

});

pm.test("Check card is not closed", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.closed).to.eql(false);

});

pm.test("card is in DONE list", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.idList).to.eql

    (pm.globals.get("list\_DONE\_id"));

});

let id = pm.response.json().id;

pm.globals.set("card\_learnpostman\_id", id);

*It is time to clean things up. As you do not need the task, the lists and the board, make sure you****delete the board****, so that your Trello account does not get filled with useless boards.*

*Please test the following:*

*- status is 200*

**Hint:**You will be using the **DELETE** method in order to **delete the board**.

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

Alle testen werken !

Graphical user interface

Description automatically generated with medium confidence

# Labo 3 - Dynamische verzoeken met gebruik te maken van variabelen.

*When creating a new board, instead of My Board, try generating a unique board name by incrementing a number (using a pre-request script).*

*For example the first board you create will be:*

*My Board 1*

*The second board:*

*My Board 2*

*The third board:*

*My Board 3*

*and so on.*

***Hint #1****: Use an environment variable to save the last used number.*

***Hint #2****: Take into account that when retrieving a value for a variable that has not been defined, you will get the value null.*

Pre- request script (uitgevoerd vóór het verzoek)

//Get lastest value of last used number

var number = pm.environment.get("LastUsedNumber");

//increment last used number

pm.environment.set("LastUsedNumber",number+1);

*Use environment variables for any dynamic parts in your requests (boardId, listId) and use them in requests and tests. Replace all usages of global variables in your requests.*

*How do does your request and tests look like now?*

Aanmaken van een nieuwe omgeving met naam Trello waar ik alle variabelen ga bewaren in plaats van in de globale omgeving

A screenshot of a computer

Description automatically generated with medium confidence

*Inspect your environment variables after you reached the last request. Clear any variables that you do not need as soon as you do not use them anymore.*

*What were the lines of code needed to clear the variables?*

pm.environment.unset("board\_id");

pm.environment.unset("list\_TODO\_id");

pm.environment.unset("list\_DONE\_id");

pm.environment.unset("card\_learnpostman\_id");

pm.environment.unset("LastUsedNumber");

Alle testen werken weer !

Graphical user interface

Description automatically generated with medium confidence

# Labo 4- Assertions

## Quiz

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Labo 5 – Introduction to Newman

## Newman installatie

Text

Description automatically generated

## Uitvoeren van geëxporteerde collecties+environment met Newman

Text

Description automatically generated

Text

Description automatically generated

## Installatie + intro testen van jenkins omgeving

Graphical user interface, text, application, email

Description automatically generated

## Collectie uitvoeten met jenkins

Graphical user interface, text, application, email

Description automatically generated

Started by user Adrien Taneko

Running as SYSTEM

Building in workspace C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\httpbin\_rest\_api\_tests

[httpbin\_rest\_api\_tests] $ cmd /c call C:\WINDOWS\TEMP\jenkins847068180510724988.bat

C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\httpbin\_rest\_api\_tests>cd C:\Users\tanek\OneDrive - ODISEE\3TI\Sotware-testing\Collections\Trello

C:\Users\tanek\OneDrive - ODISEE\3TI\Sotware-testing\Collections\Trello>newman run "Trello API test.postman\_collection.json" --environment "Trello.postman\_environment.json"

newman

Trello API test

→ Create board

POST https://api.trello.com/1/boards/?key=437f504c6149afb8e1340def5a46a4c4&token=946ba3ffbc31a5bcb4d2668fc8fa1b5794a598d564ba941a30051842779dbea6&defaultLists=false&name=Trello board 1 [200 OK, 3.09kB, 495ms]

√ Status code is 200

√ Check board name

→ Create list TODO

POST https://api.trello.com/1/boards/616c72d91b55752592932ef1/lists?key=437f504c6149afb8e1340def5a46a4c4&token=946ba3ffbc31a5bcb4d2668fc8fa1b5794a598d564ba941a30051842779dbea6&name=TODO [200 OK, 2.24kB, 224ms]

√ Status code is 200

√ Check list name

√ Check list is not closed

√ List is created in API\_test board

→ Create list DONE

POST https://api.trello.com/1/boards/616c72d91b55752592932ef1/lists?key=437f504c6149afb8e1340def5a46a4c4&token=946ba3ffbc31a5bcb4d2668fc8fa1b5794a598d564ba941a30051842779dbea6&name=DONE [200 OK, 2.23kB, 195ms]

√ Status code is 200

√ Check list name

√ Check list is not closed

√ List is created in API\_test board

→ Create card Learn postman-TODO

POST https://api.trello.com/1/cards?key=437f504c6149afb8e1340def5a46a4c4&token=946ba3ffbc31a5bcb4d2668fc8fa1b5794a598d564ba941a30051842779dbea6&name=Learn Postman&idList=616c72da6ceea121f1d089f3 [200 OK, 3.24kB, 256ms]

√ Status code is 200

√ Check card name

√ Check card is not closed

√ card is created in TODO list

→ Move card to DONE

PUT https://api.trello.com/1/cards/616c72db5b98202d46ec65b2?key=437f504c6149afb8e1340def5a46a4c4&token=946ba3ffbc31a5bcb4d2668fc8fa1b5794a598d564ba941a30051842779dbea6&name=Learn Postman&idList=616c72da638b9d28f4fa5b60 [200 OK, 3.2kB, 249ms]

√ Status code is 200

√ Check card name

√ Check card is not closed

√ card is in DONE list

→ Delete Trello board

DELETE https://api.trello.com/1/boards/616c72d91b55752592932ef1?key=437f504c6149afb8e1340def5a46a4c4&token=946ba3ffbc31a5bcb4d2668fc8fa1b5794a598d564ba941a30051842779dbea6 [200 OK, 2.13kB, 472ms]

√ Status code is 200

┌─────────────────────────┬─────────────────────┬────────────────────┐

│ │ executed │ failed │

├─────────────────────────┼─────────────────────┼────────────────────┤

│ iterations │ 1 │ 0 │

├─────────────────────────┼─────────────────────┼────────────────────┤

│ requests │ 6 │ 0 │

├─────────────────────────┼─────────────────────┼────────────────────┤

│ test-scripts │ 6 │ 0 │

├─────────────────────────┼─────────────────────┼────────────────────┤

│ prerequest-scripts │ 1 │ 0 │

├─────────────────────────┼─────────────────────┼────────────────────┤

│ assertions │ 19 │ 0 │

├─────────────────────────┴─────────────────────┴────────────────────┤

│ total run duration: 2.4s │

├────────────────────────────────────────────────────────────────────┤

│ total data received: 3.42kB (approx) │

├────────────────────────────────────────────────────────────────────┤

│ average response time: 315ms [min: 195ms, max: 495ms, s.d.: 120ms] │

└────────────────────────────────────────────────────────────────────┘

Finished: SUCCESS

## Rapporten generen met newman

Graphical user interface, text, application, Word

Description automatically generated

Npm package voor genereren van overzichtelijk HTML rapporten

npm install -g newman-reporter-htmlextra

Graphical user interface, website

Description automatically generated

zie genereerde HTML rapporten in bestand 🡺 Collections\Trello\newman

## Collectie uitvoeren vanuit GIT repository

Graphical user interface, text, application, email

Description automatically generated

Uitvoeing is gelukt

Graphical user interface, text, application, email

Description automatically generated

# Labo 6 – Automated testing with workflows and scenarios

Text

Description automatically generated

Aantwoorden:

Try getting the first boardId from the list and print it to the Postman console.

// Parse the response body

var jsonData = pm.response.json();

// Get the first object from jsonData

var firstBoard = jsonData[0];

// Print the value

console.log(firstBoard.id);

Using workflows, try reusing the DELETE request you already have, without changing the order of the requests in your collection or duplicating any requests.

Get all boards verzoek

// Parse the response body

var jsonData = pm.response.json();

// Get the first object from jsonData array

var firstBoard = jsonData[0];

if (firstBoard) {

// Print the value

console.log(firstBoard.id);

// Set the variable boardId with the first board id from jsondata

pm.environment.set("boardId", firstBoard.id);

// Set the next request to the delete board

postman.setNextRequest(‘Delete board’);

}

If there are no boards left to be deleted, make sure you stop the workflow execution.

if (firstBoard) {

....

} else {

postman.setNextRequest(null);

}

Run the collection by using the collection runner. After the run is completed, there should no boards left in your account.

Text

Description automatically generated

Text

Description automatically generated with low confidence

# Sectie 10: Data Driven tests.

Deze hoofdstuk gaat over het gebruiken van externe scripts om een collectie of verzoek uit te voeren.

Graphical user interface, application

Description automatically generated

# Sectie 11: Team collaboration

Deze hoofdstuk gaat over de samenwerking met teamgenoten in postman er zijn meerdere opties om te kunnen samenwerken en de bedoeling is om degene te selecteren die het beste aan de gebruiker zijn behoeften voldoet. Ik heb geleerd hoe ik een verzameling kan downloaden, het is ook een optietijdens samenwerking. Ten tweede heb ik geleerd hoe ik een reeds geïntegreerde werkruimte kan gebruiken.

Graphical user interface, text, application

Description automatically generated

# Sectie 12: Mock Servers

Deze hoofdstuk gaat over het gebruiken van mock servers voor:

* API designing, prototyping and development
* API testing

Graphical user interface, text, application, email

Description automatically generated

# Sectie 13: File Upload with Postman.

Deze hoofdstuk gaat over het opladen van een/meerdere externe bestand(en) met postman. Er werd ook bekijken hoe wij testen kunnen schrijven om het oplaad functionaliteit te kunnen testen. Wij leren deze ook deze tests te automatiseren.

# Sectie 14: Authentication/authorization

Deze hoofdstuk gaat over de verschillende authenticatie methoden mogelijk binnen postman.

Basis access authentification 🡺 auth met usernaam en wachtwoord verstuurd als header

Graphical user interface, text, application

Description automatically generated

## Authenticatie en autorisatie met basis access auth

Kies basic auth als auth-type en vul usernaam en wachtwoord in

A screenshot of a computer screen

Description automatically generated

## Autorisatie met Code Grant (OAuth 2.0)

Kies OAuth 2.0 als auth-type en vul gegevens in.

A screenshot of a computer

Description automatically generated with medium confidence

## Form-based Authentication

Meestal voor HTML-formulieren waar gebruiker zijn usernaam en wachtwoord moet invoeren.

Recent gecodeerde formulieren hebben een extra beveiligingslaag, anti-forgery token genaamd, waarmee wordt geverifieerd dat de gebruiker die het formulier visualiseert dezelfde gebruiker is die het formulier verstuurt.

## JSON Web tokens JWT-tokens

JWT-tokens zijn een open, industriestandaard methode voor het veilig vertegenwoordigen van claims tussen twee partijen.

Deze cursus stelt de website JWT.IO voor om JWT-tokens te kunnen decoderen, verifiëren en genereren. We leren ook hoe u geautoriseerde JWT-verzoeken in Postman kunt versturen.