Onderzoeksopdracht

E2E voor spotitube

Maarten van der Lei (658215)

Datum: 1-4-22

HAN-OOSE

DEA-Spotitube

Inhoud

[1. Inleiding 2](#_Toc99563883)

[2. Hoofdvraag 3](#_Toc99563884)

[2.1. Deelvragen 3](#_Toc99563885)

[3. Onderzoeksmethode 4](#_Toc99563886)

[4. Resultaten 5](#_Toc99563887)

[4.1. Snelheid end-points 5](#_Toc99563888)

[4.2. Leesbaarheid & code opbouw 7](#_Toc99563889)

[22. Discussie 8](#_Toc99563890)

[23. Conclusie 9](#_Toc99563891)

[Verwijzingen 10](#_Toc99563892)

# Inleiding

Een Java-applicatie als API, een webserver die verbonden is met deze lokale API, maar na een verandering wil je testen of alles nog werkt via de website.

We kunnen inmiddels de individuele code methodes testen via unit tests, maar het gehele plaatje is nog niet getest.

Je *kan* alles zelf aanklikken, invullen en doornemen.

Maar je *kan* óók eenmalig ‘end-to-end’ tests maken voor de functionaliteiten die je wil testen, en die na elke verandering runnen.

Zo weet je zeker dat alle functionaliteit via de website, nog steeds werkt zoals je verwacht dat het werkt.

Maar hoe kan je dat het beste doen? En welke stappen moet je doorlopen?

Als maker van de spotitube applicatie, wil je dat je code net is, en werkt zoals verwacht.

Zo ver hebben we de methodes binnen de spotitube applicatie afzonderlijk getest met unit tests.

Doordat Unit tests een enkele methode testen, wordt het geheel niet getest.

Met integratie testen kan je het geheel testen: van request tot response, het aansluiten van alle units op elkaar, de integratie van units.

Integratie testen is vaak de 2e stap in het testen na unit tests zijn geschreven. (dzone, 22).

Je *kan* handmatig met een http-request tool zoals postman, de endpoints benaderen, en de responses checken.

Maar *beter* is om aan de hand van integratie testen, de end-points testen.

We hebben geen controle over de webserver, maar alleen over de zelfgemaakte API-server.

Om die reden worden er integratie testen gemaakt.

In opeenvolgende hoofdstukken ga je lezen wat de winst is van het integratie testen met node.js op de Spotitube applicatie.

Het doel van dit onderzoek is dan ook om aan te tonen of het winstgevend is, in tijd en gebruiksgemak, om integratie testen te maken voor de spotitube applicatie.

De winst kan uitgedrukt worden in snelheid, gemakkelijkheid van opzetten van een test en het lezen van de integratie testen.

In de Spotitube applicatie zijn al unit-tests gemaakt, maar een unit-test test niet het geheel, maar één unit. Met een Integratie-test test je de samenhang van components of services van elkaar, oftewel: de uiteindelijke resultaten die de webserver binnen zou moeten krijgen.  
*“Integratie testen is vaak de 2e stap in het testen na unit tests zijn geschreven.”* (dzone, 22).

Er is gekozen om geen onderzoek te doen naar e2e (end-to-end) tests, die keuze is gemaakt omdat de gebruiker wel via de website van Spotitube de API benaderd, maar de developers van de API niet de webserver gebouwd hebben.  
Dat zou hetzelfde effect hebben als het testen van google…  
Om die reden testen we dus alleen de gemaakte API met een integratie test.

# Hoofdvraag

*“Wat is de winst van het gebruik van een integratie test tool voor de spotitube applicatie die gebaseerd is op node.js(Jest & SuperTest)?”*

## Deelvragen

#### Hoe snel zet je een integratie test op?

#### Wat doet een e2e test?

#### Wat is een integratie test?

#### Wat is de winst van e2e t.o.v. integratie testen?

#### Wat is de winst in snelheid van een integratie test t.o.v. handmatig testen?

#### Waarom is er in dit geval van gekozen voor integratie testen?

#### Hoe snel is een integratie test?

#### Waarom zou je integratie over unit testen kiezen?

#### Wat is de leesbaarheid t.o.v. unit-tests?

#### Hoeveel covarage heb je met 1 integratie test?

#### Hoelang duurt het om handmatig een endpoint te testen?

# Onderzoeksmethode

Met behulp van de meest populaire integratie test tool, kunnen er verschillende metingen uitgevoerd worden op de spotitube applicatie.

Vervolgens kan er vergeleken worden met de handmatige http-requests, en hoelang dat duurde.

Ook wordt er gekeken naar de leesbaarheid van een integratie test binnen Node.js, samen met hoe snel je zelf een integratie test kan schrijven.

De **snelheid** zal getest worden door 10x dezelfde test te runnen, dat zal vergeleken worden met 10x dezelfde endpoint te benaderen.

Snelheid kan getest worden door middel van postman, website naar webserver en integratie test.

**Leesbaarheid** wordt bepaald door middel van naamgeving van ingebouwde functies, opbouw van benodigde code en hoeveelheid code nodig voor 1 test.

Omdat er geen tegenhanger is om mee te vergelijken qua code, wordt er alleen gekeken of de code volstaat aan bovenstaande leesbaarheid eisen.

Code hoeveelheid mag niet meer dan 100 regels hebben.

Functies ingebouwd in Supertest moeten leesbaar zijn in wat ze doen(geen functienamen zoals: gtstd(), maar wel zoals: getTimeSinceThisDate()).

De opbouw van code moet logisch zijn en voor een gemiddelde programmeur, van boven naar beneden goed te lezen zijn.

# Resultaten

Getest met clean browser: geen cookies, getest met target folder gereset.

In postman krijg je net zoals bij je browser in het network tabje een request te zien die word gedaan, daarbij komt de request tijd.

Browser is getest via de spotitube website.

Postman is getest met de laatste versie van postman en op localhost.

## Snelheid end-points

|  |  |  |  |
| --- | --- | --- | --- |
| /login wrong creds | postman | browser | integratietest |
| 1 | 917 | 90 | 21 |
| 2 | 89 | 91 | 47 |
| 3 | 82 | 90 | 20 |
| 4 | 89 | 81 | 20 |
| 5 | 108 | 83 | 21 |
| 6 | 77 | 86 | 24 |
| 7 | 86 | 88 | 23 |
| 8 | 94 | 87 | 19 |
| 9 | 82 | 83 | 22 |
| 10 | 88 | 87 | 20 |

*Snelheid is in milliseconden.*

|  |  |  |  |
| --- | --- | --- | --- |
| /login correct creds | postman | browser | integratietest |
| 1 | 102 | 87 | 119 |
| 2 | 94 | 93 | 120 |
| 3 | 87 | 84 | 118 |
| 4 | 107 | 89 | 128 |
| 5 | 98 | 98 | 125 |
| 6 | 90 | 105 | 112 |
| 7 | 88 | 95 | 121 |
| 8 | 91 | 102 | 122 |
| 9 | 104 | 99 | 117 |
| 10 | 105 | 91 | 127 |

*Snelheid is in milliseconden.*

|  |  |  |  |
| --- | --- | --- | --- |
| /playlist correct token | postman | browser | integratietest |
| 1 | 252 | 240 | 183 |
| 2 | 278 | 232 | 192 |
| 3 | 282 | 235 | 179 |
| 4 | 256 | 239 | 193 |
| 5 | 277 | 253 | 195 |
| 6 | 273 | 255 | 184 |
| 7 | 272 | 248 | 183 |
| 8 | 275 | 286 | 182 |
| 9 | 270 | 253 | 181 |
| 10 | 262 | 247 | 179 |

*Snelheid is in milliseconden.*

De volgende test is alleen getest met postman en integratie test, omdat de aanpassingen in de browser hier te lang zouden duren om volledig te testen.

|  |  |  |
| --- | --- | --- |
| /playlist wrong token | postman | integratietest |
| 1 | 72 | 16 |
| 2 | 69 | 17 |
| 3 | 77 | 20 |
| 4 | 72 | 17 |
| 5 | 80 | 16 |
| 6 | 84 | 17 |
| 7 | 85 | 17 |
| 8 | 75 | 19 |
| 9 | 79 | 16 |
| 10 | 87 | 15 |

*Snelheid is in milliseconden.*

## Leesbaarheid & code opbouw

1. const request = supertest(process.env.API\_URL || 'http://localhost:8010');
2. it('POST /login wrong creds', async () => {
3. const times = [];
4. for(let i = 0; i < 10; i++) {
5. const start = Date.now();
6. await request.post(`/login`)
7. .send({
8. user: 'user',
9. password: 'incorrect',
10. })
11. .expect(401)
12. .then((res) => {
13. expect(res.body.token).toBe(undefined);
14. expect(res.body.user).toBe(undefined);
15. });
16. const end = Date.now();
17. const time = end - start;
18. times.push(time);
19. }
20. console.table(times);
21. });

*Voorbeeld met onjuist wachtwoord/user combinatie direct uit code.*

# Discussie

// discussie voer:

E2e vs integratie

Unit vs integratie

Intern vs extern integratie testen?!(change db to connect to testDb)

# Conclusie

<https://medium.com/@adrian_lewis/top-5-most-rated-node-js-frameworks-for-end-to-end-web-testing-f8ebca4e5d44>

<https://www.cypress.io/>

<https://nightwatchjs.org/>

<https://www.slant.co/topics/8034/~javascript-end-to-end-testing-tools>

<https://www.testim.io/blog/end-to-end-testing-vs-integration-testing/>

----

<https://www.onpathtesting.com/blog/is-integration-testing-really-necessary>

<https://dzone.com/articles/integration-testing-what-it-is-and-how-to-do-it-ri>

<https://www.geeksforgeeks.org/software-engineering-integration-testing/>

# Verwijzingen

dzone. (22, 3 30). *integration-testing-what.....* Opgehaald van dzon: https://dzone.com/articles/integration-testing-what-it-is-and-how-to-do-it-ri

geeksforgeeks. (22, 3 30). *software-engineering-integration-testing*. Opgehaald van geeksforgeeks: https://www.geeksforgeeks.org/software-engineering-integration-testing/