Spotitube Integratie Testen

Integratie testen voor de Spotitube applicatie.

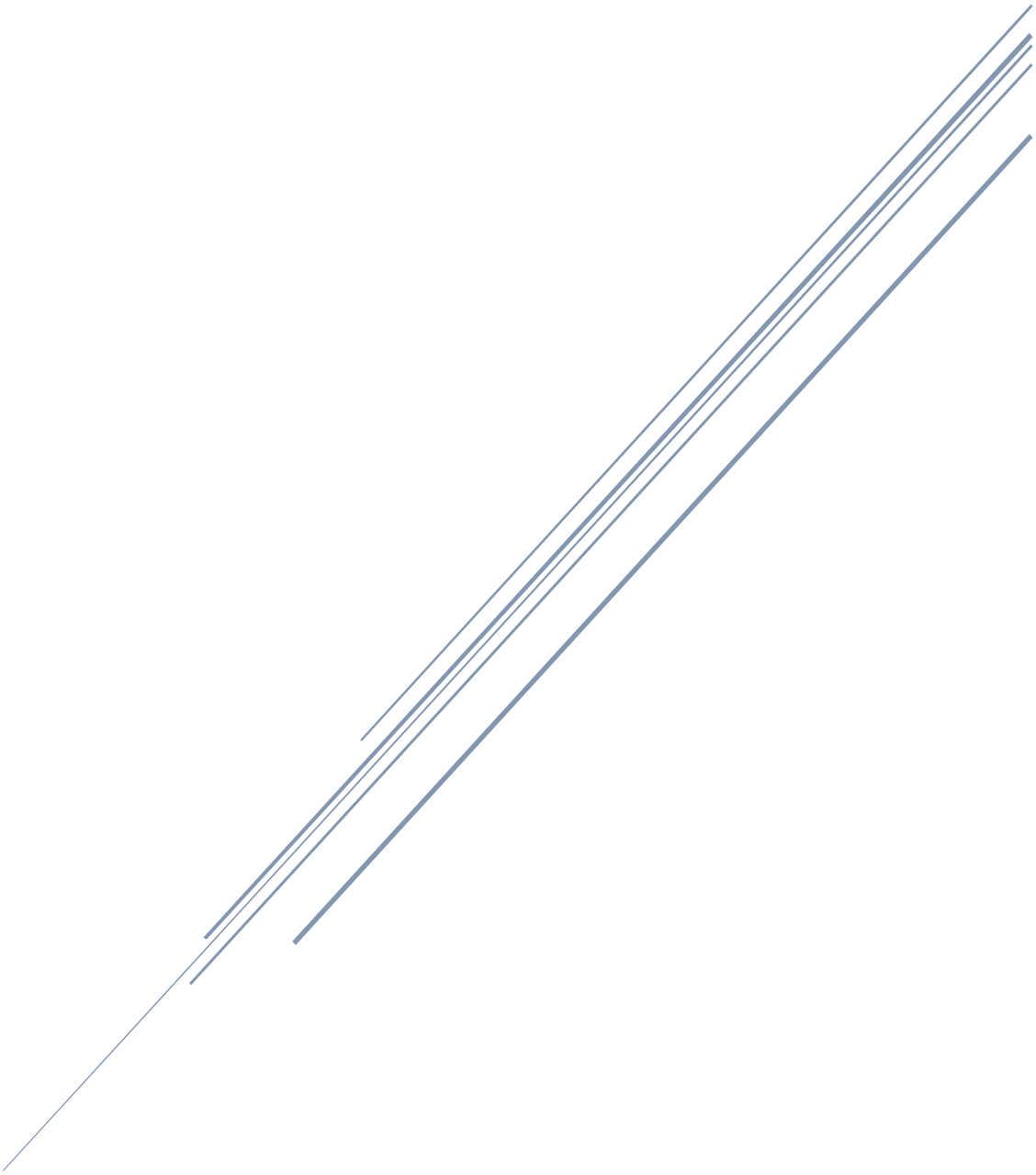
Maarten van der Lei (658215)

Klas: ITA-OOSE-A-f

Datum: 1-april-2022.

Course: DEA.

Opdrachtgever: Wiebe Rinsma.



HAN-OOSE
DEA-Spotitube

Inhoud

1. Inleiding

In opeenvolgende hoofdstukken ga je lezen **of er winst is, en zo ja,** 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, voor/nadelen bij gebruik in een soortgelijk java project, 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 2^{e} stap in het testen na unit tests zijn geschreven." (dzone, 2022).

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.
Om die reden testen we dus alleen de gemaakte API met een integratie test.

2. Hoofdvraag

“Wat is de winst van het gebruik van integratie testen voor de spotitube applicatie die gebaseerd zijn op Node.js(Jest & SuperTest)?”

2.1. Deelvragen

Hoe snel zet je een integratie test op met node.js?

Wat doet een e2e test in tegenstelling tot een integratie test?

Wat is een integratie test?

Hoe snel is een integratie test met node.js?

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?

Hoelang duurt het om handmatig een endpoint te testen?

Wat is het verschil met java integratie tests?

Welke voordelen heeft het integratie testen met Node.js?

Welke nadelen heeft het integratie testen met Node.js?

3. 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(uitlezen in console) en integratie test.

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

Projectbasis, er zal gekeken worden wat de beste manier is om in een project/bij productie met integratie testen te werken, en wat de winst van Node.js hier dus is.

Code moet voldoen aan de volgende regels:

> Code hoeveelheid van 1 test mag niet meer dan 10 regels hebben(afgezien van globale test opzet zoals classes of describes).
>
> 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.

4. Integratie testen

In dit hoofdstuk vind je even kort wat informatie over wat integratie testen inhoud, wat het verschil is met e2e testen en waarom dat niet gekozen is voor dit onderzoek en integratie testen binnen een project basis om zo de workflow van het project te beïnvloeden.

4.1. Wat zijn integratie testen?

Een integratie test is een test die niet alleen een unit test, maar meerdere units, modules of componenten binnen een applicatie test.

Het doel van een integratie test is dan ook om de connectiviteit en samenhang tussen units te testen (techtarget, 2022)

4.2. Integratie testen versus e2e testen

Integratie testen zijn bedoeld om de uiteindelijke output van een systeem te testen.

Waar e2e bedoeld is om de end-to-end experience te testen van een eindgebruiker.

In de context van de spotitube API is daarom niet gekozen voor een onderzoek naar e2e tests van wege het feit dat de developers die de spotitube API maken niet de front-end gemaakt hebben.

4.3. Integratie testen binnen een project

Integratie testen binnen een project is belangrijk omdat je vooraf je endpoints kan bedenken, vast stelen, en vervolgens maken naar de specificaties, om ze daarna te testen met de voorafgaand gemaakte integratie testen, ook wel TDD genoemd.

Maar waarom is het dan juist binnen een project belangrijk?

Omdat je binnen een project meestal vanuit specificatie werkt, en de code schrijft om aan de specificatie te voldoen.

Met integratie testen kan je elke commit/ronde/sprint, de tests runnen om te kijken welke endpoints al aan de verwachtingen voldaan zijn.

Ook kan je met een pipeline een "*Continious Integration*" workflow opzetten, een van de manieren is github actions, daarmee kan je de integratie tests runnen bij een commit/push, maar voor dit onderzoek word daar niet op in gegaan.

4.4. Wat is Node.js?

Node.js is een javascript runtime environment gebouwd op chrome V8

Voor meer info: https://nodejs.org/en/about/ .

5. 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.

5.1. Snelheid end-points

/login wrong creds	postman	browser	integratietest
1	97	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
AVG	89,2	86,6	23,7

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
AVG	96,6	94,3	120,9

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
AVG	269,7	248,8	185,1

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
AVG	78	17

Snelheid is in milliseconden.

5.2. Leesbaarheid & code opbouw javascript

```
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.
7.  // actual test:
8.        await request.post(`/login`)
9.          .send({
10.           user: 'user',
11.           password: 'incorrect',
12.         })
13.         .expect(401)
14.         .then((res) => {
15.           expect(res.body.token).toBe(undefined);
16.           expect(res.body.user).toBe(undefined);
17.         });
18. // test end.
19.       const end = Date.now();
20.       const time = end - start;
21.       times.push(time);
22.     }
23.     console.table(times);
24.   });
```

Voorbeeld met onjuist wachtwoord/user combinatie + speedtest for-loop direct uit code.

5.3. Leesbaarheid + code opbouw Java

```
1.  @Test
2.  public void givenUserDoesNotExists_whenUserInfoIsRetrieved_then404IsReceived()
3.          throws ClientProtocolException, IOException {
4.   String name = RandomStringUtils.randomAlphabetic( 8 );
5.   HttpUriRequest request = new HttpGet( "https://api.github.com/users/" + name );
6.   HttpResponse httpResponse = HttpClientBuilder.create()
7.                                       .build()
8.                                       .execute( request );
9.   assertThat(
10.    httpResponse.getStatusLine().getStatusCode(),
11.    equalTo(HttpStatus.SC_NOT_FOUND)
12.  );
13. }
```

Voorbeeld van java code met maven HttpClient & Jackson2, herschreven, uit artikel van: (Paraschiv, 2022)

5.4. Voordelen integratie testen met nodejs

Voordelen van integratie testen met nodejs:

- Je test van buiten af, en kan dus de gebruikers ervaring beter nabootsen.
- Je schrijft makkelijk te lezen code met javascript in combinatie met async/await of Promises.

5.5. Nadelen integratie testen met nodejs

nadelen van integratie testen met nodejs:

- Omdat je van buiten de spotitube applicatie test, heb je geen invloed op de database connectie en ben je dus verbonden met de “productie” database.
- Door punt 1 kan je dus in productie geen testen runnen die impact hebben op de data die de eindgebruiker ziet.
- Door punt 1 en 2 sluit je daarmee veel POST, PUT, DELETE requests uit in de Spotitube app.

6. Discussie

6.1. Beantwoorden van deelvragen

Hoe snel zet je een integratie test op met node.js?

Node.js en NodePackageManager zijn vereist.
Een integratie test met node.js zet je op met npm.

- Door met npm de packages: Jest en Supertest te installeren.
- Wanneer deze zijn geinstalleerd, maak je een `__tests__` folder aan binnen het project, daar komen alle tests binnen, en jest zal deze testen.
- Vervolgens zet je een test commando in de package.json, dit commando zal gebruikt worden om alle test-suites(gelijk aan de test folder binnen java) in de __tests__ folder te runnen.
- Zet nu eerst aan het begin van je bestand de regels code die de supertest package importeert met de require import methode.
- Zet vervolgens de locatie van de API binnen de supertest('API_URL komt hier').
  Stel een constante gelijk aan deze functie, zodat je deze bij de tests kan gebruiken.

Dat gedaan hebbende, kan je beginnen met het schrijven van de 1e test.

- Dat doe je door een 'describe' te maken met een naam als string en een functie callback: de daadwerkelijke test.
- Binnen de functie callback van deze describe zet je een 'it' met dezelfde eigenschappen, in ons geval maak je de callback een asynchrone functie zodat async/await gebruikt kan worden.
- Roep nu een http methode aan op de zojuist aangemaakte constante van de **supertest('API_URL')**, en zet **await** hiervoor zodat je wacht totdat er een response terug is.
  vervolgens kan je allerlei soorten testen halen over de response.

Zie voorbeeld van experiment met supertest hier.

Om de deelvraag te beantwoorden: een gemiddeld ontwikkelaar kan deze stappen binnen **10minuten** uitvoeren(uitgaande van stabiele internet connectie en kennis van javascript).

Hoe snel zet je een handmatige test klaar om een endpoint van spotitube te testen?

Postman of soortgelijke tool is vereist.

- Wanneer je met postman een endpoint wil testen, hoef je alleen maar de API_URL in te voeren in de duidelijk zichtbare balk.
- Na dat ingevoerd te hebben, voer je eventuele query of path parameters in, en mogelijk geef je een body mee in de vorm van JSON.
- Daarna klik je op de verzend knop, en snel zie je een resultaat van de API.
- Dit resultaat is gemakkelijk te lezen, waarna je zelf kan checken of de waardes kloppen met wat je ingevoerd had.

Om antwoord te geven op de deelvraag: gemiddeld download en installeer je met een stabiele internet connectie, postman in ongeveer 2minuten, waarna nog eens 1minuut voor het typen van de juiste waardes, wat kan uitlopen tot 2minuten maximaal.
om op een totaal van **+-5minuten** uit te komen.

Hoe snel zijn integratie tests versus handmatig testen?

Volledige resultaten lijst

Hier onder samengevat de snelste methode, de responsetijd van postman versus een tests.

Deze tests bevatten geen menselijke input, en zijn daarom niet erg relevant, totdat we gaan kijken naar wat de tijd winst is over duur, op bijvoorbeeld projectbasis.

	Integratie	Handmatig
/login correct credentials		
/Login wrong credentials		
/Playlist correct token		
/Playlist wrong token		

Hier aan zien we dat de integratie test via node.js **75%** van de tests sneller is.

In de resultaten zien we ook het gemiddelde van de tests, waarbij het oploopt van **30 tot 60ms** per endpoint test.

Als je meerdere endpoints in 1 keer wil testen, scheelt dat gemiddeld **40ms** per keer, en bij 10 endpoints is dat al **bijna een halve seconde**.

Run je de integratie tests vaker op een dag, dan het verschil oplopen **tot een paar secondes per dag**, en vergeet niet dat de menselijke factor ook mee telt, als je bijvoorbeeld postman afsluit, ben je zo een paar minuten verder op een dag, terwijl de integratie tests, bij wijze van spreken, in je IDE leven.

In een project van 8 weken is die duur dus veel langer, dan ben je gemiddeld per week een paar minuten sneller, en in 8 weken kan dat oplopen tot een kwartier tot 30minuten.

Om de deelvraag te beantwoorden: het testen van endpoints met een integratie test versus handmatig, scheelt op dag basis **gemiddeld tot een paar minuten**, meegerekend de menselijke factor van het per ongeluk afsluiten van een applicatie.

Integratie testen in Java?

Integratie testen met Java zou in het geval van Spotitube in productie de beste oplossing zijn.

Waarom? Bekijk de onderstaande tabel met voor en nadelen in kleur aangegeven van het integratie testen met node.js op de Spotitube applicatie.

	Handmatig	*Node.js*	*Java*
Projectbasis/productie		Geen Test DB	Test DB
Development		geen impact productie DB.	Zelfde omgeving
Snelheid			-
Opzet			-

(- = niet getest)

Rood = niet geschikt vanwege in resultaten/deelvragen genoemde redenen

Oranje = mogelijk, maar zeker niet de beste in de categorie.

Geel = opzet mogelijk niet nodig/opzet afhankelijk van kennis.

Groen = goed voor de categorie.

Wanneer je niet met een productie omgeving werkt, en je database dus alleen voor ontwikkelaars zichtbaar is, dan is Node.js een goede optie.

Node.js maakt gebruik van een de chrome V8 runtime environment en is daarmee snel voor lokaal gebruik.

6.2. Beantwoorden hoofdvraag

De winst van het integratie testen van de Spotitube applicatie met Node.js is aanwezig op gebied van persoonlijke voorkeur van taal(java vs javascript vs handmatig), op gebied van snelheid is node.js sneller dan het handmatig testen van endpoints binnen de spotitube applicatie.

Op gebied van project/productie gericht testen, is Node.js geen goede optie voor de Spotitube applicatie door het feit dat je dan test data in je productie data krijgt wat in het algemeen een slechte gewoonte is binnen ontwikkelaars land.

Is Integratie testen met Node beter dan handmatig voor Spotitube? Ja.

Is Integratie testen met Node beter dan met Java voor Spotitube? Ja.

De Winst van het integratie testen van de Spotitube applicatie met Node.js:

- Je testen zijn **sneller dan** als je **handmatig** test.
- Je testen zijn **makkelijk** op te zetten.
- Je testen zijn met javascript(**persoonlijke voorkeur**) tegenover Java.
- Je test met Node.js van buitenaf de Spotitube applicatie waardoor je de **eindgebruiker beter kan nabootsen**.

7. Conclusie

Wanneer je werkt met een API die niet wijzigingen doorbrengt binnen een database, dan kan je prima de Node.js integratie tests gebruiken voor een Java applicatie.

Maar omdat we praten over de Spotitube API, en die maakt wel gebruik van database wijzigingen, wil je eigenlijk een test database maken om niet je productie data aan te passen met vervuilde test-data.

Dat kan je niet doen van buiten af de Spotitube applicatie.

Vanuit binnen in kan dit wel door een nieuwe database te maken, en daar in een @BeforeEach mee te verbinden(kan overigens ook bij supertest en express.js, maar dan moeten die wel lokaal samen draaien).

Daarmee komt het uiteindelijk voor de Spotitube applicatie neer op de context van de ontwikkelaar, **is Spotitube in productie?** Dan kies je **Java**, **is Spotitube in ontwikkeling en niet in productie?** Dan ligt de **keus bij de ontwikkelaar** voor welke taal hij/zij wil kiezen en of die graag de integratie tests in hetzelfde project heeft als de applicatie code.

Wat je ook kiest: Integratie testen schrijven is beter dan handmatig testen.

Verwijzingen

borg, b. (2022, maart 31). *is-integration-testing-really-necessary*. Opgehaald van onpathtesting: https://www.onpathtesting.com/blog/is-integration-testing-really-necessary

dzone. (2022, maart 30). *integration-testing-what.....* Opgehaald van dzon: https://dzone.com/articles/integration-testing-what-it-is-and-how-to-do-it-ri

geeksforgeeks. (2022, maart 30). *software-engineering-integration-testing*. Opgehaald van geeksforgeeks: https://www.geeksforgeeks.org/software-engineering-integration-testing/

Honig, R. (2022, maart 31). *6-best-practises-integration-testing*. Opgehaald van techbeacon.com: https://techbeacon.com/app-dev-testing/6-best-practices-integration-testing-continuous-integration

jest. (2022, maart 31). *docs/*. Opgehaald van jestjs.io: https://jestjs.io/docs/getting-started

npm. (2022, maart 31). *package/supertest*. Opgehaald van npmjs: https://www.npmjs.com/package/supertest

Paraschiv, E. (2022, maart 31). *integration-testing-a-rest-api*. Opgehaald van bealdung.com: https://www.baeldung.com/integration-testing-a-rest-api

techtarget. (2022, maart 31). *integration-testing*. Opgehaald van techtarget.com: https://www.techtarget.com/searchsoftwarequality/definition/integration-testing