Eric Boot

SIE-V2A

22-06-2017

TesTPlan Praktijkopdracht 2

Versie 2.0

Inhoudsopgave

[1 Versie 3](#_Toc477028535)

[2 Introductie 3](#_Toc477028536)

[3 Tests 3](#_Toc477028537)

[3.1 Integratie test 3](#_Toc477028538)

[3.2 Unittest 3](#_Toc477028539)

[3.3 Smoketest 5](#_Toc477028540)

# 1 Versie

|  |  |  |
| --- | --- | --- |
| **Versie** | **Opmerking** | **Testplan-versie** |
| 1.0 | Definitieve testplan | 1.0 |
| 2.0 | Herkansing Testplan | 2.0 |
|  |  |  |

# 2 Introductie

Dit testplan is onderdeel van de 2e praktijkopdracht van de cursus Intigration and Communcation. Voor deze opdracht wordt er verwacht dat de student een REST Service realiseert in het thema ‘the Smart Cities’.

De test die zullen worden uitgevoerd zijn: Integratie test, Unittest en Smoketest. Deze tests zijn verplicht gesteld in de opdracht.

# 3 Gebruik service

De service die gemaakt is, zal op basis van input een voorspelling doen van de toekomstige windsnelheid op een bepaalde plek. Daarbij kan de windsnelheid worden opgevraagd in 2 formaten: m/s en km/h. De URI voor het bereiken van de methode voor de toekomstige windsnelheid in m/s is “/services/futurewindspeed/getms”. De URI voor km/h is “/services/futurewindspeed/getkmh”.

De input die hierbij wordt verwacht is een locatie, gedefinieerd volgens longitude en latitude, en een tijd in de toekomst, gedefinieerd volgens datum en tijd.

Omdat er sprake is van een REST Service, zal er gecommuniceerd worden via JSON. Een voorbeeld van een JSON dat zou kunnen worden gestuurd naar de gemaakte REST webservice is:

|  |
| --- |
| {  **"data"**: {  **"longitude"**: -12,  **"latitude"**: -55,  **"date"**: **"2017-04-08"**,  **"time"**: **"04:01:00"**  }  } |

In het bovenstaande object is te zien dat de gegevens in een object zitten met de key “data”. In dat object zitten 4 properties welke allemaal in het formaat key:value zijn. De keys van het data-object kunnen in elke volgorde voorkomen. Aan de values zitten enige voorwaarden:

* De longitude moet een getal zijn tussen de -180 en 180.
* De latitude moet een getal zijn tussen de -90 en 90.
* De date moet in het format “yyyy-mm-dd” (= jaar in 4 cijfers - maand in 2 cijfers - dag in 2 cijfers) zijn.
* De time moet in het format “HH:mm:ss” (= uur in 24-uurs notatie - minuten in 2 cijfers - seconden in 2 cijfers) zijn.

# 4 Tests

## 4.1 Integratie test

De integratie test wordt gerund op het moment dat de webservice wordt gestart. Dit gebeurt met Maven. De dependecies van Maven staan gespecificeerd in de pom.xml.

## 4.2 Unittest

Voor de unittest is er gebruik gemaakt van jUnit. Met dit framework worden er tests gerund die tonen dat de uitkomst correct is. Hieronder is de source code te vinden.

|  |
| --- |
| **import** org.json.JSONObject; **import** org.junit.Test; **import** services.FutureWindSpeedService;  **import static** org.junit.Assert.*assertEquals*; */\*\*  \* Created by Eric Boot on 25-03-17.  \*/* **public class** jUnit {  FutureWindSpeedService **futureWindSpeedService** = **new** FutureWindSpeedService();   @Test  **public void** testKmH(){  JSONObject data = **new** JSONObject();  JSONObject item = **new** JSONObject();  item.put(**"longitude"**, 52);  item.put(**"latitude"**, 12);  item.put(**"date"**, **"2017-01-01"**);  item.put(**"time"**, **"12:01:00"**);   data.put(**"data"**,item);   String message = data.toString();   *assertEquals*(**"De data is gelijk aan verwacht"**,  **"{\"unit\":\"km/h\",\"windspeed\":5.681951219512196}"**,  **futureWindSpeedService**.getKmH(message).getEntity());;   }   @Test  **public void** testMS(){  JSONObject data = **new** JSONObject();  JSONObject item = **new** JSONObject();  item.put(**"longitude"**, -12);  item.put(**"latitude"**, -55);  item.put(**"date"**, **"2017-04-08"**);  item.put(**"time"**, **"04:01:00"**);   data.put(**"data"**,item);   String message = data.toString();   *assertEquals*(**"De data is gelijk aan verwacht"**,  **"{\"unit\":\"m/s\",\"windspeed\":1.5054200542005418}"**,  **futureWindSpeedService**.getMS(message).getEntity());   } } |

De testcases die behoren bij de unittest zijn er 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase | Code naam | Versie 2.0 | Opmerking |
| 1 | testKmH |  |  |
| 2 | testMS |  |  |

## 4.3 Smoketest

Op de screenshots, aanwezig in de REST-folder is te zien dat de service werkt en de methoden met elkaar kunnen communiceren.