****

Testplan

Integration and Communication

CODE TCIF-V2IAC1-15

HBO-ICT SIE Jaar 2 2016-2017

Auteurs

Jochem Kuus

Datum

22 – 03 – 2017

Versie 0.2

© Hogeschool Utrecht, Utrecht, 2015

# Inhoudsopgave

[Inhoudsopgave 2](#_Toc477097601)

[1. Versie 3](#_Toc477097602)

[2. Introductie 3](#_Toc477097603)

[3. Testomgeving 3](#_Toc477097604)

[4. Smoketest 3](#_Toc477097605)

[5. Testcases black box test (integratietest of functionele test) 4](#_Toc477097606)

[6. Unit Tests 4](#_Toc477097607)

# Versie

|  |  |  |
| --- | --- | --- |
| **Versie** | **Wie?** | **Opmerking** |
| 0.1 | Jochem Kuus | Initialisatie |
| 0.2 | Jochem Kuus | Aanpassing van het SOAP-testplan |

# Introductie

In dit project wordt er een service gebouwd die aan de hand van een postcode de bijbehorende latitude en longitude opzoekt. De huidige versie wordt beschouwd als versie 1.0, dit omdat de functionaliteit is toegevoegd en de resterende taken alleen maar test gerelateerd zijn.

Er is een JUnit testsuite klaargemaakt met een aantal mogelijke error inputs. Doormiddel van deze JUnit-tests wordt er dus ge-unittest, dit wordt gedaan om te controleren dat de Java module correct werkt. JUnit is een simpele manier om dit te testen aangezien zonder al te veel code uitvoerige tests geschreven kunnen worden.

Ook is er weer in SOAP-UI getest of de service werkt.

# Testomgeving

De enige service die getest gaat worden is de LocationService. Allereerst is er de vraag of de service correct werkt, om vervolgens te controleren of de opgelegd beperkingen niet overschreden kunnen worden. Er is een lengte beperking (maximale lengte is 6), een samenstellingsbeperking (4 cijfers en 2 letters in die volgorde) en er is een beperking in het resultaat (er moet wel een locatie terugkomen). Al deze beperkingen horen foutcodes terug te geven die dan bij SOAP-UI worden getoond, nu is het aannemelijk dat een regular expression de meeste syntax fouten eruit filtert dus dit drukt het aantal foutcodes.

# Smoketest

De volgende testcases worden uitgevoerd:

1. (Correct) Postcode = 3731XC
2. (Incorrect syntax /lengte) Postcode = 33731XC
3. (Incorrect syntax/lengte) Postcode = 37317X
4. (Incorrect bestaat niet) Postcode = 5555RR
5. (Incorrect syntax) Postcode = 3731@X

In mijn beleving zijn deze tests voldoende om alle functionaliteit van het systeem te controleren.

# Testcases black box test

Er wordt bij de requests gebuik gemaakt van de volgende link:

Localhost:8080/rest/location/lookup/{Request/Postcode}

SOAP-UI, dus de service, zal het volgende terug geven bij de volgende requests:

|  |  |
| --- | --- |
| Request | Response |
| 3731XC | Een lat- en longitude. |
| 33731XC | Verkeerde syntax |
| 37317X | Verkeerde syntax |
| 5555RR | Locatie bestaat niet. |
| 3731@X | Verkeerde syntax |

# Unit Tests

JUnit testcases:

|  |  |
| --- | --- |
| Sourcecode | Testcase |
| public class posTest {  // test in which the result will be correct, postcode found and lat/long are matching  @Test  public void classWorks(){  LocationServiceImpl lSI = new LocationServiceImpl();  String result = lSI.getLatLng("3731XC");  assertEquals(result, "{\"latitude\":52.1113981,\"longitude\":5.1796013}");  }  } | 1 |
| public class negTestInvalLeng {  // testing with a postcode of an invalid length  @Test  public void classCrashLen(){  LocationServiceImpl lSI = new LocationServiceImpl();  String result = lSI.getLatLng("33731XC"); // this postcode has 5 numbers and 2 letter instead of 4 numbers and 2 letters  assertEquals(result, "{\"Errorcode\":500,\"Errormessage\":\"Postcode (33731XC} does not follow the standard: 1234AB\"}");  } } | 2 |
| public class negTestManyNums {  // testing with a postcode that is incorrect  @Test  public void classCrashMoreNums(){  LocationServiceImpl lSI = new LocationServiceImpl();  String result = lSI.getLatLng("37317X"); // this postcode has 5 numbers and 1 letter instead of 4 numbers and 2 letters  assertEquals(result, "{\"Errorcode\":500,\"Errormessage\":\"Postcode (37317X} does not follow the standard: 1234AB\"}");  }  } | 3 |
| public class negTestInvalChar {  // testing with a postcode that is incorrect  @Test  public void classCrashInvalidChar(){  LocationServiceImpl lSI = new LocationServiceImpl();  String result = lSI.getLatLng("3731@X"); // this postcode has a @ which is an illegal character  assertEquals(result, "{\"Errorcode\":500,\"Errormessage\":\"Postcode (3731@X} does not follow the standard: 1234AB\"}");  }  } | 5 |
| public class negTestNotExist {  // testing with a postcode that doesn't exist  @Test  public void classCrashNoLoc(){  LocationServiceImpl lSI = new LocationServiceImpl();  String result = lSI.getLatLng("5555RR");  assertEquals(result, "{\"Errorcode\":404,\"Errormessage\":\"Location not found for postcode: 5555RR\"}");  }  } | 4 |

# Resultaat testcases

De resultaten van SOAP-UI tests en JUnit tests waren gelijk en als volgt;

De volgende testcases gaven een latitude en longitude terug:

1. (Correct) Postcode = 3731XC

De volgende tests gaven een syntax error terug:

1. (Incorrect syntax /lengte) Postcode = 33731XC
2. (Incorrect syntax/lengte) Postcode = 37317X
3. (Incorrect syntax) Postcode = 3731@X

De volgende testcases gaven een ‘Not found error’ terug:

1. (Incorrect bestaat niet) Postcode = 5555RR

## Conclusie testcases

Alle testcases zijn volgens verwachting geslaagd.