****

Testplan Opdracht 1

Integration and Communication

CODE TCIF-V2IAC1-15

HBO-ICT SIE Jaar 2 2016-2017

Cursuseigenaar: jeroen.weber@hu.nl

Student Elias Karimi

Datum

22 – 03 – 2017

Versie 0.2

# Inhoudsopgave

[Inhoudsopgave 2](#_Toc473464760)

[1. Versie 3](#_Toc473464761)

[2. Introductie 3](#_Toc473464762)

[3. Testomgeving 3](#_Toc473464763)

[4. Smoketest 3](#_Toc473464764)

[5. Testcases black box test (integratietest of functionele test) 3](#_Toc473464765)

[6. Unit Tests 3](#_Toc473464766)

[7. Regressietests 4](#_Toc473464767)

# Versie

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Description | Version | Date |
| Elias Karimi | Up to Blackbox test | 0.1 | 15-03-2017 |
| Elias Karimi | Finalized | 0.2 | 22-03-2017 |

# Introductie

De ontworpen webservice is gemaakt om de bevolkingsgroei in landen bij te kunnen houden. Door middel van een SOAP webservice kunnen clients voor een land de inwonersaantallen oproepen. Als response krijgen zij dan een overzicht van het aantal mannen, vrouwen en de totale inwoners. Mijn eerste opzet was om dit voor steden of wijken te doen, maar hier vond ik geen publieke API voor. Ik ben daarom opzoek gegaan naar een API die dit toeliet voor landen en kwam terecht op <http://population.io>.

De service is op dit moment voor data per jaar, maar zou met een kleine aanpassing ook de percentuele bevolkingsgroei in kaart kunnen brengen over een range van jaren, of met een iets andere omzet per wereld deel. Om de webservice enigszins eenvoudig en testbaar te houden heb ik nu gekozen om het nog per land te doen.

De API van *Population.io* laat alleen data zien van de jaren 1950-2100 en alleen voor bestaande landen. Daardoor zal ik de data van de clients valideren op deze twee factoren. Een landenlijst is gelukkig ook te verkrijgen via de API. Als de client een foute request doet zal er een foutmelding worden gegeven.

We zullen gaan testen via smoketests en blackboxtests, dit zal ervoor zorgen dat we kunnen testen of de service te bereiken is, hij data returned en of dit zo werkt zoals het hoort.

# Testomgeving

De applicatie is getest op een Macbook met daarop OSX 11, Eclipse Neon 3, SOAPUI 5.3, een Tomcat 8.5 server en Java 1.8.

Voor het project kunt u terecht op <http://github.com/shyrinak/IAC_OPD1> .

De SOAP service heb ik getest met SOAP UI versie 5.3. Voor de instellingen van Tomcat 8.5 moet mogelijk de server.xml aangepast worden om het project in de root directory van de server te laten draaien, zodat de service te vinden is op <http://localhost:8080/soap/>

Er is gekozen voor een top-down configuratie, de skeleton classes zijn gegenereerd met *wsimport* vanuit het WSDL bestand en de XSD bestanden voor de response, request en errors. Hierna is de SEI gemaakt.

# Smoketests

Met SOAP UI voeren een simpele smoketest uit om te kijken of de service werkt en of hij werkt zoals het hoort via de schema’s.

We gaan kijken of de service bereikbaar is en dus geen “404: Not found error geeft”, of hij bereikenbaar is vanuit het schema en geen andere fouten aangeeft zonder iets verkeerds in te geven.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Input | Verwachte Output | Werkelijke Output | Succes |
| SMOKE\_01 | Service Reachable | Ja |  |  |
| SMOKE\_02 | Service Reachable trough WSDL | Ja |  |  |
| SMOKE\_03 | Shows no error on request | Ja |  |  |

# Testcases black box test (integratietest of functionele test)

We gaan hier kijken naar het WSDL bestand. Alleereerst de operatie die in het WSDL bestand gedefieneerd staat.

  <wsdl:binding name="WS-Binding" type="WS-Interface">

    <soap:binding style="document"

transport="http://schemas.xmlsoap.org/soap/http"/>

    <wsdl:operation name="PopulationCountryByYear">

    <soap:operation soapAction="" style="document"/>

        <wsdl:input>

        <soap:body use="literal"/>

        </wsdl:input>

       <wsdl:output>

        <soap:body use="literal"/>

       </wsdl:output>

       <wsdl:fault name="ApplicationError">

        <soap:fault name="ApplicationError" use="literal"/>

        </wsdl:fault>

      </wsdl:operation>

  </wsdl:binding>

We zien hier dat er drie type meldingen bestaan; de *fault* of error zodra er iets fout gaat, de input en de output melding. Voor elk van deze drie meldingen hebben we een message opgenomen in het WSDL.

<wsdl:message name="request">

    <wsdl:part name="parameters" element="req:request"/>

  </wsdl:message>

  <wsdl:message name="response">

      <wsdl:part name="response" element="rsp:response"/>

  </wsdl:message>

  <wsdl:message name="ApplicationError">

    <wsdl:part name="ApplicationError" element="flt:ApplicationError"/>

  </wsdl:message>

De request is wrapped en wordt in het request.xsd schema aangegeven als bestaande uit een *country* element, van het type string, en een *year* element type int, met een minimale waarde van 1950 en maximale waarde van 2100. Response en Application error zijn beide outputs.

Request zijn door het XSD in het forrmaat.

<soapenv:Envelope xmlns:soapenv=<http://schemas.xmlsoap.org/soap/envelope/>

xmlns:req="http://eliaskarimi/request">

<soapenv:Header/>

<soapenv:Body>

<req:request>

<country>Het land</country>

<year>2001</year>

</req:request>

</soapenv:Body>

</soapenv:Envelope>

Terwijl responses als volgt zijn bepaald:

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

<S:Body>

<ns2:response xmlns:ns2="http://eliaskarimi/response" xmlns:ns3="http://eliaskarimi/error" xmlns:ns4="http://eliaskarimi/request">

<result>

<female>8075895</female>

<male>7908470</male>

<total>15984365</total>

</result>

</ns2:response>

</S:Body>

</S:Envelope>

We gebruiken dus SOAP UI om de service te kunnen testen. Deze tool maakt gebruik van het originele WSDL bestand om zo request uit te kunnen voeren op de juist endpoints in het juiste formaat. Door vervolgens een request uit te voeren waarvan je weet welke response je terug moet krijgen kan je testen of deze dan werken.

We voeren ook een niet bestaand land en jaartal uit om te kijken of dit wordt gevangen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Input | Verwachte Output | Werkelijke Output | Succes |
| BLACK\_01 | <country>The Netherlands</country>  <year>2001</year> | <<female>8075895</female>  <male>7908470</male>  <total>15984365</total> |  |  |
| BLACK\_02 | <country>Brazil</country>  <year>1950</year> | <female>27172815</female>  <male>26775989</male>  <total>53948804</total> |  |  |
| BLACK\_03 | Foutief jaartal buiten range:  <country>Brazil</country>  <year>1550</year> | <faultcode>S:Server</faultcode>  <faultstring>Het jaartal valt niet binnen de minimale en maximale range</faultstring> |  |  |
| BLACK\_04 | Foutief Land:  <country>Het land van ooit</country>  <year>2001</year> | <faultcode>S:Server</faultcode>  <faultstring>Het opgegeven land bestaat niet</faultstring> |  |  |

# Regressietests

Alle deze testen zijn pas uitgevoerd nadat de code al op basis functionaliteit getest was. Daardoor zouden er nu geen wijzigingen meer plaats moeten vinden indien er geen fouten meer bijkomen. Ik ga dus gebruik maken van de hierboven Smoke en Blackbox testen en deze zouden allemaal de verwachte output moeten genereren.