Naast de kandidaat selectie op productaanbod en prijs, wil dit onderzoek ook de gebruiksvriendelijkheid in acht nemen. Hiervoor is er een simpele Java applicatie gebruikt. Alle Cloud platformen bieden starterhandleidingen aan voor het opzetten van een CI-pijpleiding voor Java op hun platform. Het doel van deze snelle test is om als eindresultaat een werkende Java JAR te hebben. Ook werd er bijgehouden hoelang het duurde om een pijpleiding te configureren om beter een idee te krijgen van hoe gebruiksvriendelijk het Cloud platform werkelijk is.

De Java Applicatie bestaat uit een hoofdklasse en een testklasse. De hoofdklasse, ‘MessageUtil’ bevat een variabele voor een bericht te bewaren en vier methodes. De eerste methode is een constructor voor het aanmaken van een ‘MessageUtil’ object met een meegegeven bericht. De tweede methode print dit bericht. In de derde methode wordt er een toevoegsel aan dit bericht gevoegd. Ook wordt het geheel geprint. De laatste methode is een uitvoerbare methode die een ‘MessageUtil’ object maakt en de twee print methodes uitvoert. De code voor ‘MessageUtil.java’ wordt afgebeeld in figuur (verwijzing).

public class MessageUtil {

private String message;

public MessageUtil(String message) {

this.message = message;

}

public String printMessage() {

System.out.println(message);

return message;

}

public String salutationMessage() {

message = "Hi!" + message;

System.out.println(message);

return message;

}

public static void main(String[] args) {

MessageUtil mu = new MessageUtil("Idioten zijn er overal...");

mu.printMessage();

mu.salutationMessage();

}

}

De testklasse ‘TestMessageUtil’ bevat twee testen voor de print methodes. Figuur (verwijzing) toont de code voor dit Javabestand.

import org.junit.Test;

import org.junit.Ignore;

import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

String message = "Robert";

MessageUtil messageUtil = new MessageUtil(message);

@Test

public void testPrintMessage() {

System.out.println("Inside testPrintMessage()");

assertEquals(message,messageUtil.printMessage());

}

@Test

public void testSalutationMessage() {

System.out.println("Inside testSalutationMessage()");

message = "Hi!" + "Robert";

assertEquals(message,messageUtil.salutationMessage());

}

}

Deze Javabestanden worden op alle Cloud platformen gecompileerd aan de hand van de Maven compiler. Om de compiler te vertellen wat er moet gebeuren tijdens de compilatie van de Javabestanden, moet er een Xml-bestand aangemaakt worden waarin deze opties gedefinieerd staan. Deze ‘pom.xml’ wordt afgebeeld door figuur (verwijzing). Hierin staat dat de meegeleverde Junit testen uitgevoerd moeten worden. Ook wordt er gedefinieerd dat de Java applicatie moet gecompileerd worden tot een JAR-bestand.

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>messageUtil</artifactId>

<version>1.0</version>

<packaging>jar</packaging>

<name>Message Utility Java Sample App</name>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<classpathPrefix>lib/</classpathPrefix>

<mainClass>MessageUtil</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

</project>

Deze bestanden zijn toegevoegd aan een folder die de structuur hanteert uit figuur (verwijzing). Deze hoofdmap is dan geïnitialiseerd als een Git repositorie. Ook is er een gitignore aangemaakt. Deze Java applicatie is het startpunt voor alle testen voor gebruiksvriendelijkheid op de Cloud platformen. Op deze manier is er geprobeerd om op basis van eventuele verschillen een geschikte kandidaat te selecteren voor een Proof Of Concept. In de volgende secties (verwijzing) wordt kort de werkwijze beschreven. Hier wordt de ‘hoe’ minder aangehaald omdat dit in de handleiding beschreven staat.

Demo\_java\_aws/

├── .gitignore

├── pom.xml

└── src

├── main

│ └── java

│ └── MessageUtil.java

└── test

└── java

└── TestMessageUtil.java