Prof. Dr. Stefan Tai
Marco Peise

Your name: _____

Enterprise Computing (WS 2016/17)

Exercise 2 (3 Portfoliopunkte)

Info:

- The solution to this exercise must be handed in by Wednesday, Nov 16th 2016, 12AM.
- Any written solution must be in an accessible PDF. Any source code in a separate ZIP File. All Files are uploaded in the Information System for Instructors and Students in a single ZIP File (https://isis.tu-berlin.de/course/view.php?id=8586).
- Please write your name on the solution sheet.

Task 1 – REST Server (20%)

The task is to build and provide two Services which are described in Exercise 1 Task 1. Therefore, you may use a PaaS provider of your choice and build an RESTful Interface with an API Framework called StrongLoop in JavaScript (Node.js).

Service 1: should be a HTTP PUT Operation and be able to store a given String and a Date by the client in JSON Format into a Database

Service 2: should be a HTTP GET Operation (name: "timeToDate") should be able to search for a given String (i.e.: "chrismas_marco_peise) within the same Database, calculate the remaining time till that event and respond within a human friendly string (i.e.: "1 month 12 days 8 hours 42 minutes 40 seconds").

The Service should have a REST Interface with a valid Endpoint for at least one week (16.11 – 23.11.2016).

Your task is, to provide us with a valid Endpoint for each Service.

Prerequisites for Implementation with Node.js:

1. Download or clone the git repository at [1]
2. Install node.js [2]
3. Open the project in WebStorm / Eclipse or your favorite JavaScript Development Environment
4. "npm install" to install referenced packages
5. Run node.js application with node server/server.js
6. Deploy your code to your PaaS environment and start the Node.js server [3]

References:

[1] Project git repo: https://github.com/marcopeise/EC-Exercise2_1.git

[2] Node.js: https://nodejs.org/en/download/

[3] How to install Application to IBM Bluemix:

https://www.eu-gb.bluemix.net/docs/starters/install_cli.html

Task 2 – Amazon Simple Storage Service (S3) (40%)

- Clone the Eclipse project *https://gitlab.tubit.tu-berlin.de/peise/EC-AWSS3*
- Build the project with Maven

Prerequisites:
Set up your AWS credentials as follows:
~/.aws/credentials
[default]
aws_access_key_id=enteryourkeyhere
aws_secret_access_key=enteryoursecrethere

(If you don't know how to do this, please read this: http://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/credentials.html or post a question in the ISIS2 forum for Enterprise Computing)

a) Now fill the blanks with your code (30%)

```
// TODO create a bucket with name "ise-tu-berlin-exercise2-",
// followed by your student id (Matrikelnr)
log.info("Creating a bucket (if it does not exist, yet)");

// TODO Upload a text File object to your S3 bucket
// use the createSampleFile method to create the File object
log.info("Uploading an object");

// TODO Download the file from S3 and print it out using the
// displayTextInputStream method.
log.info("Downloading an object");
```

b) Which AWS S3 operation uses which HTTP method? (10%)

AWS operation	**HTTP method**
createBucket	
putObject	
getObject	
deleteObject	

Task 2 – Amazon Simple Queuing Service (SQS) (40%)

Set up the Eclipse project with the following application source code:

- Clone the project from *https://gitlab.tubit.tu-berlin.de/peise/sqs*
- Open the project in Eclipse.
- Build the project with Maven.

Rewrite the borrower/lender application by replacing JMS with AWS SQS. Most of the structure already exists but some pieces are missing. Fill out the blanks in the snippets below with your code.

Solution:

// *SqsBorrower.java*
// TODO check response queue for matching responses

// Print out the response

// delete the message from the queue

// *SqsLender.java*
// TODO Prepare receive loan request message request.

// TODO Check request queue for loan requests.

// TODO Delete loan request message from queue

// TODO Send out the response