UNIVERSITY OF TARTU

MTAT.03.229 – Enterprise System Integration

Regular Exam – 12 January 2015

Notes:

- The exam is open-book and open-laptop. Web browsing is allowed
- You are not allowed to communicate with anyone during the exam in any way (except with the lecturer).
- The solution to Part 1 (zip file containing the eclipse project) must be submitted using the "Submit" button in the course Web page. The solution to Part 2 can be submitted on paper or electronically. When electronically, you must include the file (e.g. PDF, DOC or DOCX) within the eclipse project used in Part 1.
- If you find that there is not enough information in the text below and you need to make additional assumptions, please write down your assumptions.

Part 1. BuildIT & RentIt: Change Implementation and Testing

In the current version of BuildIt's information system, a Plant Hire Request (PHR) is by default set into PENDING status when the Site Engineer creates it. Then, the Works Engineer is responsible for accepting or rejecting the PHR after analyzing the request. It has been recently decided that the PHR can be automatically accepted whenever its cost is less than or equal to 1500 EUR. In all the other cases, the processing of the PHR should proceed as usual (i.e., the Works Engineer has to review the PHR and then accept or reject it).

Task 1. [Test: 6 points + Business logic: 2 points]. You are requested to implement the changes at both the business logic and its corresponding integration test.

You can clone the git repository with the current implementation from https://bitbucket.org/lgbanuelos/esi2014-exam1. The code is distributed in the form of two eclipse projects, namely testingexample-jdk7 and testingexample-jd8. You should choose either one depending on the version of Java that you have installed in your computer.

As the code interacts with RentIt, you must use a mock of RentIt's API. To this end, you can use the file "apiary/rentit.md" provided in both eclipse projects (as for your practice sessions). Recall that you can use "api-mock" in your own computer or you can use apiary's cloud services (if you use apiary, you must update the URLs in the class "demo.services.RentalService" accordingly).

HINTS:

1. The changes on the business logic concern only one method "createPlantHireRequest" from class "PlantHireRequestRestController".
2. In this moment, the code contains only one test (method "createPHR" from class "PlantHireRequestRestControllerTests"). You must refactor this test into two methods:

a. one for the case where the PHR is automatically approved (i.e. when PHR cost <= 1500), and

b. the other one for the case where the PHR is not automatically approved (i.e. when PHR cost > 1500).

Part 2. Taxis of Strelsau

Traditionally, the taxi market in Strelsau (Ruritania) has been highly fragmented. As of 2014, there were around 20 taxi companies, each one operating an average of 100 taxis. Each company handles its own taxi ordering service inefficiently based on rudimentary systems used by operators at the call centres of the companies.

The situation is about to change with the launch of United Taxis of Strelsau (UTS), which will absorb about half of Strelsau's taxi companies. United taxis will manage a fleet of around 1000 taxis, with the ambition to grow up to 1500 taxis by 2017.

United Taxis will put in place a single taxi ordering system: The Strelsau Taxi Ordering System (STRS). STRS will be multi-channeled. Customers will still be able to order a taxi via phone (the traditional way), but additionally, they will be able to do so via a Web-based front-end and via mobile applications available for all major smartphones (Android, iPhone and Windows 8). The Web front-end will allow customers to enter the address where they require a taxi, their name and their mobile number. STRS will inform the customer of the time when the taxi will pick them up. This information will also be sent to the customer via SMS. Updates will also be sent to the customer in case the taxi is delayed.

When reserving via a smartphone application, the application will automatically fetch the GPS location of the customer (if available), so that that the customer does not need to enter the location explicitly – assuming they wish to be picked from their current location. Customers will instantly get an estimate of how much time the taxi will take to pick them up.

All taxis will be equipped with Android tablet PCs with GPS. An application called TaxiHome will be installed in each of these devices. Taxi drivers will use TaxiHome to notify STRS of their availability. At a given point in time, a taxi can be off-duty, available, busy and invisible (this status is taken when no status update has been received by STRS for more than 5 minutes). When taxis are in available or busy states, TaxiHome will periodically report the location of the taxi to STRS.

When STRS receives a new taxi order, it assigns it to the closest available taxi. If there are several taxis at equal or almost equal distance from the location, the taxi that has been available the longest is assigned (i.e. first-in-first-out). STRS will communicate the assignment to the corresponding taxi. STRS will be built using a location service called Awaze, which is able to calculate the distance and approximate travel time between any two locations in Strelsau, given as input the GPS coordinates or the address identifiers of the locations in question, and taking into account roadworks and traffic. Once an order has been assigned to a taxi, the taxi's terminal is notified that an order has been assigned to them. The taxi driver can retrieve the details of the order via TaxiHome, and accept the order or reject it. If rejected, an alternative taxi is assigned by STRS.

Taxi operators must pay a fee to UTS equivalent to 3% of the revenue generated by completed taxi orders made via STRS, where a "completed taxi order" means an order for a taxi where the customer was picked up taken to a destination and paid for the ride. To this end, taxi drivers need to indicate via TaxiHome when a customer is picked-up and dropped-off (in the latter case TaxiHome retrieves the cost of the ride from the taximeter). At any point in time, taxi drivers are able to see via the TaxiHome app their "account statement", which means the history of completed orders for the current month and the previous month, together with the total fee they have to pay for those orders.

Tasks

Task 2 [8 points]. Design a domain model for STRS.

Task 3 [8 points]. Based on the domain model, design a *resource model* using the RESTful service design method presented in the lectures.

Task 4 [8 points]. Specify a state machine capturing the lifecycle of a taxi order in STRS. Each transition triggered by an external operation should be labelled with the name of the operation, the corresponding HTTP verb and URL of the resource.

Task 5 [18 points] Specify a RESTful interface for STRS by means of an apiary blueprint. For each operation on a resource, you should specify an example of a request/response, using "links" where appropriate.