MTAT.03.229 – Enterprise System Integration

Regular Exam, 27 May 2019

Notes:

- The total duration of the exam is 3 hours and 30 minutes.
- Parts 1 and 2 of are closed-book and closed-laptop. These two parts should be submitted on paper. Once you submit parts 1 and 2, you will be allowed to open your laptop to start part 3.
- Part 3 is open-laptop. Web browsing is allowed. This part should be submitted using the "Submit" button on the course web site
- You are not allowed to communicate with anyone during the exam in any way (except with the lecturer).

Part 1. Foundations (10 points, 2 points per question)

1) In the Rentit system, the container of purchase orders (POs) has URL /api/sales/pos/. This URL is used to create POs and to retrieve POs. For example, this resource is accessed (with GET method) by customers who wish to retrieve all their POs, and by managers who wish to inspect all POs that have been placed in the past month and their status. Managers are keen to understand how the POs evolve over time, for example, which POs reach an "invoiced" status, and which ones are abandoned in one way or another along the way.

   The URL template of a given PO is /api/sales/pos/{po.id} where po.id is the identifier of the PO. This URL is used to retrieve and to make updates to a PO.

   During a meeting with the development team, the question is posed of how to cancel a PO? A junior developer proposes to do so by means of DELETE on /api/sales/pos/{po.id}. Would you agree with this proposal? If not, what would you tell to the developer and what alternative would you propose?

2) Another junior developer proposes that to accept a PO, one should use a PATCH operation on /api/sales/pos/{po.id.}/accept . Would you agree with this proposal? If not, what would you tell to the developer and what alternative would you propose?

3) Consider an application to manage room bookings. One of the resources in this application is the Booking. The application exposes a container resource /bookings/ and item resources with URL template /bookings/{booking.id}. A booking consist of a reference to a room, a start date-time and an end date-time and a status. You are discussing how to design the REST API for managing bookings. One of the requirements is to allow managers to view all bookings in the database that have status “pending”. A junior developer proposes to do so by doing a GET on /api/bookings/open/all. Would you agree with this proposal? If not, what would you tell to the developer and what alternative would you propose?

4) A portal for freelance translators allows customers to submit translation jobs. When a job is submitted, it is initially in a “submitted” state. Translators can place bids for “submitted” jobs. Jobs remain in the “submitted” state for 24 hours. After 24 hours, the job is assigned to the translator who placed the best bid (i.e. the bid with the lowest price, or if multiple bids have the lowest price, then the one with the earliest bidding time). The best bid is marked as “accepted” while the remaining bids are marked s “rejected”. Translators can view the set of bids they have placed (both the accepted and rejected ones). A junior developer proposes that the operation for creating a bid should be: POST /api/jobs/bids/. Would you agree with this proposal? If not, what would you tell to the developer and what alternative would you propose?

5) What is the relation between Docker containers and microservices?

Part 2. RESTful API of Pharmak (20 points)

Pharmak Ltd. is a distributor of pharmaceutical products. Its client base is primarily composed of pharmacy chains that regularly place bulk orders for a range of pharmaceutical products. At present, Pharmak provides a Web-based interface allowing customers to query its product catalogue, to submit, track, and modify purchase orders, and to view invoices. Following requests from its larger customers, Pharmak is seeking to offer a fully programmatic interface to expose its sales process, all the way from receipt of purchase order to invoicing.

The envisioned API shall allow the purchasing systems of Pharmak's customers to directly retrieve all products in Pharmak's catalogue or those products that match a given keyword (e.g. a product name or a name of an active ingredient). Such queries return a list of product catalogue entries, each consisting of a unique product code, a name, the active ingredients (e.g. dextromethorphan, levodopa or paracetamol), a description of the product, the price and availability. The availability of the product can be "immediate" or "on demand". An "immediate" availability means that the product can be shipped in 1 or 2 business days, while "on demand" means that the product is generally not maintained on stock at Pharmak, but instead it needs to be ordered from a manufacturer and may take several business days to be shipped.

The API will also allow customers to submit *Purchase Orders* (PO). A PO consists of one or more *line items*. Each line item corresponds to one product. A line item has a sequential number (1, 2, 3…), a product code, the required quantity of the product, the unit of measure (e.g. "per box"), the latest acceptable shipment date and the unit price. For example, one line item in an order may be for 300 boxes of Benadryl (code P0293) to be shipped by 31.01.2014 and payable at 1.53 euros per box. In addition to a list of line items, a PO contains the business identity code, the business name of the customer, and the identifier of the PO (this identifier is chosen by the customer).

When a customer submits a PO, it expects first a simple acknowledgment indicating that the PO has been received. Within one business day, Pharmak must reply to the customer with a PO Response. A PO Response contains a reference to the original PO to which it refers, and a list of *line item responses*. A line item response includes the sequential number of the PO line item to which it refers (e.g. 1, 2, 3…) and a code "ACK" for "accepted" and "DEC" for "declined". "ACK" means that that supplier agrees to provide the product in question. In this case, the line item response also indicates the expected shipment date. "DEC" means that the supplier cannot provide the product in question, in which case the response includes a note explaining the reason for rejection (e.g. because the price in the PO line is incorrect or because the product cannot be shipped by the requested "latest acceptable shipment date").

When Pharmak ships a line item of a PO (or several line items), it will communicate this shipment to customers programmatically by sending a *Shipment Notification*. A shipment notification contains a list of PO line item references, where a PO line item references consists of a PO identifier, a line item number and an optional comment (a string). Similarly, when the customer receives the shipment, it will communicate this receipt to Pharmak programmatically by sending a *Delivery Receipt*, which also consists of PO line item references.

A customer may request changes in their PO. To do so, they submit a *PO Change Request*. A PO Change Request includes a reference to the original PO and a list of *PO Change line items*. A PO Change line item contains the sequential number of the

PO line item to which the response refers (e.g. 1, 2, 3…), and the same data that a PO line item contains (product identifier, quantity, etc.). When a customer submits a PO Change Request, it expects first a simple acknowledgment indicating that the PO Change Request has been received. Within one business day, Pharmak must reply to the customer with a *PO Change Response*. A PO Change Response contains a reference to the original PO Change Request, and a list of line item responses (as defined previously). A change request for a PO line item cannot be accepted if the PO line item has already been shipped.

Similarly, customers may request an order to be cancelled by sending a *PO Cancellation Request*. The procedure for handling PO Cancellation Requests is the same as for PO Change Requests. First the customer expects an immediate acknowledgment and within one business day it expects a *PO Cancellation Response*, which consists of a reference to the original PO Change to which it refers, and a list of line item responses, where a line item response includes a line item sequential number and either ACK (cancellation accepted) or DEC (cancellation declined). A PO line item cannot be cancelled if the corresponding line item has already been shipped.

Customers may consult the status of a PO line item ("pending response", "responded", "shipped", "delivered", etc.) as well as the expected shipment date (if the shipment has not yet been made) or the actual shipment date (if the shipment has already been made).

Five business days after delivery of a PO line item, Pharmak shall send an *Invoice* for the PO line item in question to the customer. An invoice consists of a unique identifier, and a list of *invoice lines*. Each invoice line has a sequential number (1, 2, 3), a reference to a PO, a reference to the number of the line item in the PO (which is generally different from the sequential number of the invoice line itself) and the amount to be paid. Moreover, an invoice has a total and other data, which we will ignore in this exercise. An invoice may refer to line items belonging to multiple POs.

Tasks

1. Design a domain model of Pharmak's sales information system. [8 pts]
2. Specify a state machine capturing the lifecycle of a PO line item. Each transition triggered by an external operation should be labelled with the name of the operation (the "relation"), the corresponding HTTP verb and the URL of the resource. [4 pts]
3. Specify the REST API (set of operations) that Pharmak shall offer to its customers. The specification of a REST operation should include the (relative) URL of each resource, the HTTP verbs each resource accepts, the relation that each verb implements over a resource, and a brief description of the request body and the response if applicable (no need to provide a JSON example, just explain what attributes there should be in the input and output). [6 pts]
4. Write a concrete example (in JSON) of the HTTP body of a response to a catalogue query containing two product catalogue entries. [2pts]

Part 3. REST API Implementation and Testing (20 points)

In the practice session of 19 March, we implemented a version of Rentit's information system that allows us to search plants, create POs, and accept or reject POs. Your task is to extend this implementation with the ability for customers to request a change to the rental period of a PO. Note that this is different from requesting a rental extension, as you will see when reading the specification below.

A request to change the rental period of a PO is automatically accepted so long as the following conditions hold:

- The PO has already been accepted
- The start date of both the current rental period and of the new rental period are in the future (i.e. the rental has not started and it does not start today).
- There is an available plant inventory item of the same type (i.e. of the same plant inventory entry) for the new rental period. If the plant inventory item currently allocated to the PO is available during the new period, the corresponding reservation should be updated accordingly. Otherwise, if the currently allocated plant inventory item is not available for the new rental period, but another item is available, then this other item should be allocated to the PO instead of the old one.

This operation should return HTTP status code 200 if the change is possible, 404 if the PO does not exist, and 409 if the change of period is not possible. If 409 is returned, it means that no changes are made to the PO. If 200 is returned, it means that the change was recorded and then, the updated purchase order is sent back in the response. The operation should be use either the PATCH or the PUT verb.

You should keep the separation between the controller, the service and the repository, i.e. the business logic should be in the service and data access should be in the repository.

Task. [Tests: 7.5 points + Business logic: 12.5 points]. You are required to implement the above operation and to write one or more test(s) to ensure that the above requirement is met. Your test(s) should cover the cases where: (i) the same inventory item is available during the new period; (ii) the case where the same item is not available during the new period but a different item is available; and (iii) the case where no item is available during the new period and hence status code 409 is returned. To get full points for testing, you should provide the Cucumber feature(s) for your test(s).

You can start by cloning the last commit of the following git repository:
https://bitbucket.org/orlenyslp/esi-2019/

Your application does not need to have a frontend. It only needs to expose the required REST operations. The tests should be done on the API (i.e. you should call the API's operations).

You should submit a TXT or PDF file with the link to your Bitbutcket repository with the solution. Please leave your solution in the "master" branch of your repository and do not make any commits to "master" after the end of the exam. Please add also an explanation of the changes you made to the API, i.e. new REST operations you added with their URL template and HTTP verb.