INFO303 Course Project Phase 2

Due: Wednesday 20th May @ 5pm
Marks: 15%

Contents

1. The AJAX Client

The system must provide an AJAX client that allows a customer to create an account.

Refer to section 3.3.2 of the phase 1 document for the customer details that must be captured by this interface. The **`id`**, **`group`**, and **`uri`** fields are server-generated and can be ignored for the purposes of the AJAX client.

This client can be based on AngularJS (as demonstrated in lab 3 exercise 2).

The AJAX client should POST the customer account details to a URI that is provided by the Camel router. No other operations are required for this client.

2. The Camel Router

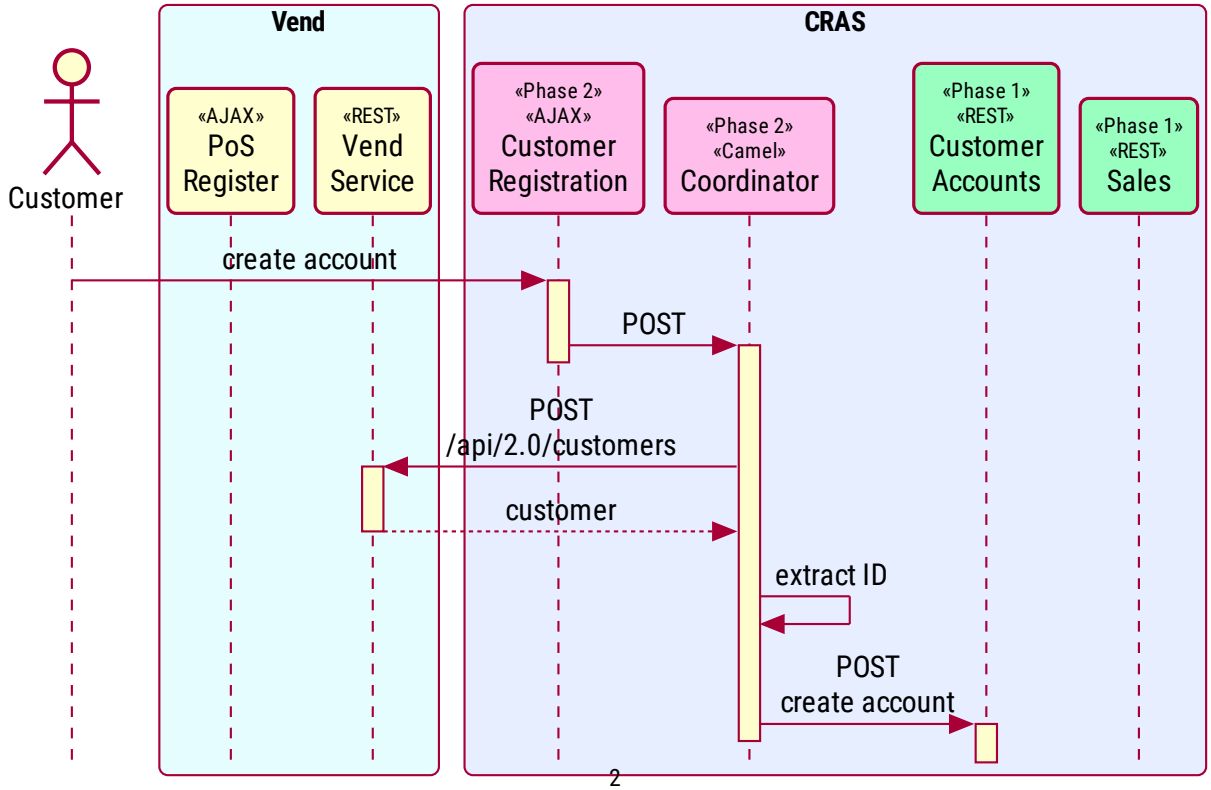
You should try to break the router into route builders that encapsulate operations. For example, it would make sense to have a route builder for the 'Customer Creates Account' operation, and another route builder for the 'Customer Makes a Sale' operation. You can add both builders to a single router.

2.1. Customer Creates Account

When a customer creates an account via the AJAX client, the router should create an equivalent customer account on Vend. The response from Vend will contain the Vend generated ID for the new customer. An equivalent customer account including the Vend ID should be created on the phase 1 accounts service.

The Camel router should provide a Jetty endpoint to receive the account details from the AJAX client. You will need to add code to the router to handle Cross Origin Resource Sharing (CORS) as shown in the reference document.

The following sequence diagram shows the process:



The Vend API for creating accounts is described at:

https://docs.vendhq.com/reference/2/spec/customers/createcustomer

The payload can be submitted to Vend in JSON format (**application/json**).

You should set the customer's group to the 'Regular Customers' group when you create the account on Vend. The **customer_group_id** for the regular customers group is:

```
0afa8de1-147c-11e8-edec-2b197906d816
```

2.2. Customer Makes a Sale

When a sale is made on Vend, an email containing a JSON encoded version of the sale details will be sent to your student email account (via a webhook service that we are hosting on a University server).

This payload should be picked up by the router and used to:

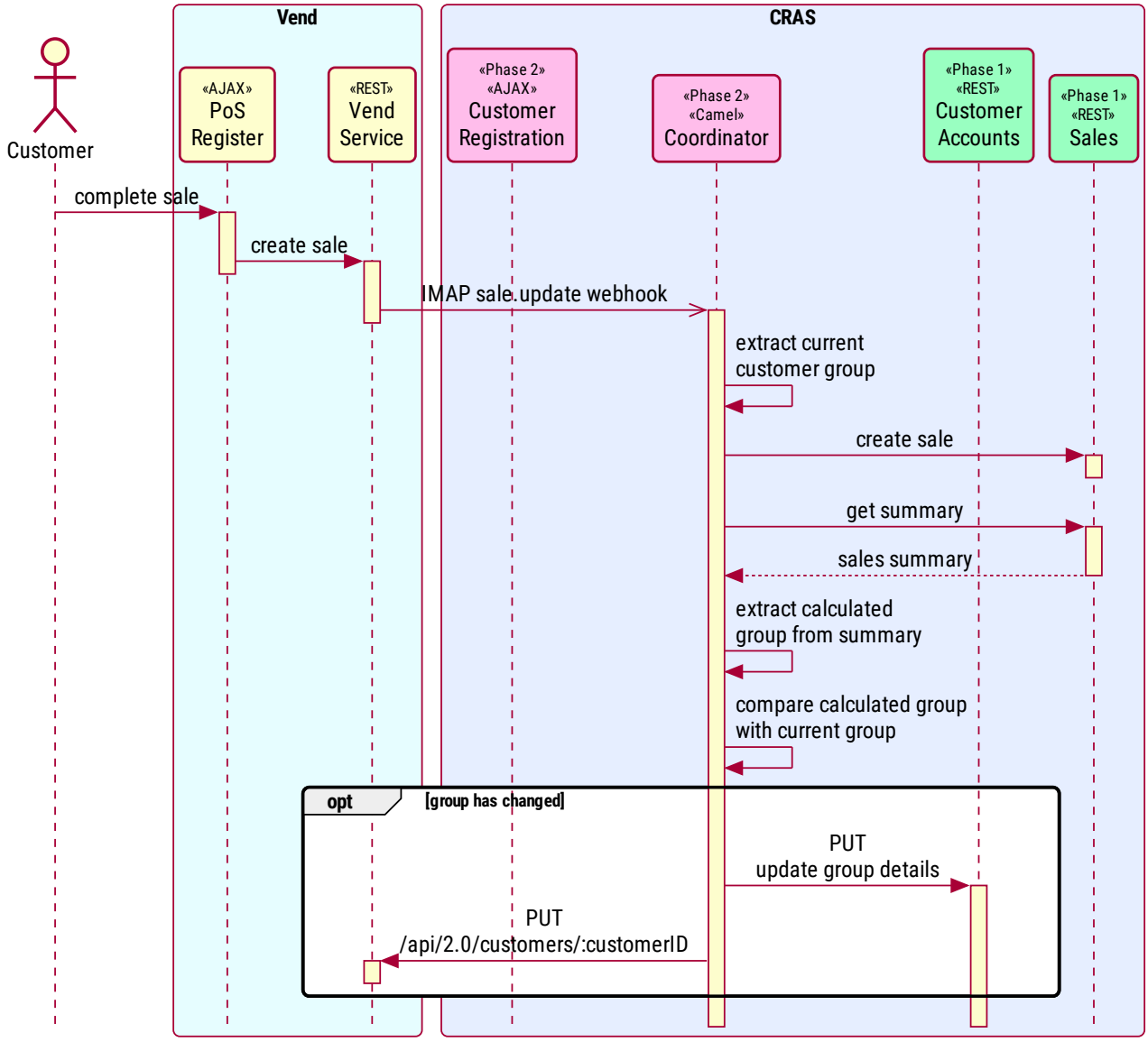
1. Extract the customer's current group.
2. Create a sale on the phase 1 sales service.

   You will first need to convert the JSON payload into Java objects that are compatible with your sales service.

   The sales service domain model from phase 1 can be used for this. The domain classes that were given to you in the common project contain Gson annotations that make the classes directly compatible with Vend's JSON sale payload.
3. Retrieve the customer's sales summary from the phase 1 sales service.
4. Extract the customer's calculated group from the summary.
5. Compare the calculated group with the current group.
6. If the group has changed then the customer's group should be updated on both the phase 1 customer accounts service and Vend. The Vend API for updating a customer is described at:

   https://docs.vendhq.com/reference/2/spec/customers/updatecustomerbyid

The following sequence diagram describes this process:



3. Interacting with Vend

3.1. Authentication

Vend uses OAuth 2 for authentication. We have created a token for you to use to authenticate your Vend API requests.

See the section in the reference document titled 'Vend API Authentication'. The example in that section includes the correct authentication token.

3.2. Customer Groups

The following customer groups exist on Vend:

Regular Customers – Customers in this group have normal pricing applied.

The Vend ID for this group is:

```
0afa8de1-147c-11e8-edec-2b197906d816
```

VIP Customers – Customers in this group have a 20% discount applied to their prices.

The Vend ID for this group is:

```
0afa8de1-147c-11e8-edec-201e0f00872c
```

You can hard-code these IDs in your system.

4. EIP Diagram

You need to document your router using the EIP notation. The intention is that your diagram(s) should show the processes that you router performs – you should be consulting your code as your draw the diagram. The diagram(s) should be included somewhere obvious in your submitted repository.

We will provide the SVG template that we use to draw EIP diagrams. We primarily use Inkscape to draw these diagrams (freely available for Windows, macOS, and Linux), but you can use any drawing tool that you are familiar with. Please ensure that the diagram(s) you produce have sufficient resolution to be comfortably viewed on a monitor.

Note that in the past there has been a weird interaction between NetBeans and Inkscape when you try to use copy and paste (they fight over the contents of the clipboard). This results in endless spawning of error dialogs appearing in Inksape. You can solve this by closing NetBeans when using Inkscape. You can look at your files on GitBucket if you need to refer to your code.

There is also an online editor that supports EIP diagrams:

https://draw.io

The EIP diagrams are found under the *Software* category.

It would be a good idea to at least sketch out a hand-drawn EIP diagram to plan your routers before writing any code. Lab 8 has some examples. You can also refer to the 'Enterprise Integration Pattern Key' section of the reference document.

5. Starting Project

A starting project has been provided for you (on Blackboard). This is a Gradle multi-project and includes all sub-projects and dependencies that you are likely going to need.

This project also includes our solutions to the phase 1 services. Given the current remote teaching situation, we highly reccomend that you use our solutions rather than your own.

Lab 8 describes how to work with Gradle multi-projects in section 3.

6. Submission and Requirements

- Implement the AJAX client as described in section 1.

- Implement the Camel router as described in section 2.
- Draw an EIP diagram that describes your router as described in section 4.

All files should be contained inside a single repository on the INFO303 GitBucket. You can create a new repository for phase 2 if you wish.

Submission will via the INFO303 GitBucket. More details on how to submit will be made available closer to the deadline.

6.1. Minimum Requirements for Phase 2

As mentioned in section 8.4 of the course outline we have some minimum standards that must be met before we will mark a submission. Submissions that do not meet this standard will receive a mark of 0.

The minimum requirements for phase 2 are as follows:

- A Camel router must be able to create a customer account in the customer accounts service.

  If you don't have a functional AJAX client then using a Camel message producer to supply the data to the router is sufficient for the minimum requirements.

- Your projects must all build without any errors.
- Your code should be reasonably presented. Code should be formatted and use appropriate identifier names. Industry standards for naming should be adhered to where appropriate.
- Your Git history should show incremental development over a reasonable period of time. Submissions that contain either:
  - very few commits, or
  - the majority of commits were made over a very short period of time

  will not be accepted.

  If you have trouble with your Git repository then you need to let us know ASAP so we can help you to fix it. As described in section 8.4 of the course outline, we operate a "no surprises" policy, meaning we need to know about Git related problems when they occur, and not at the time of submission/marking. Abandoning a repository and creating a new one when problems are encountered is not accepatable in either the industry or INFO303.