Order Handling System

Code Test

None

Table of contents

1. Order Handling System

1.1 Start

Welcome to the Innovation labs code test.

1.1.1 Purpose

This test is not only designed to test your problem-solving skills, it is also intended to show some of the frameworks, problem areas and tech stack that are used within the labs.

You are encouraged to complete the tasks to the best of your ability as this would provide this basis for our tech conversation.

1.1.2 Questions?

- Guillaume Favreau@volvo.com - guillaume.favreau@volvo.com
- Robinson Mgbah - robinson.mgbah@volvo.com

1.2 Background

Welcome to the `ohs` application team.

`Ohs` is a `grocery order handling system` which manages clients grocery orders. The system provides 4 microservices

Order Service

This service manages all orders in the ecosystem. The `gRpc` definition file can be found in the `ohs-api-test/protobuf/order` folder

Product Service

This service keeps track of the products available to be ordered The `gRpc` definition file can be found in the `ohs-api-test/protobuf/product` folder

Supplier Service

This service keeps track of the suppliers available in the system The `gRpc` definition file can be found in the `ohs-api-test/protobuf/supplier` folder

User Service

This service keeps track of the users that interact in the system The `gRpc` definition file can be found in the `ohs-api-test/protobuf/user` folder

1.2.1 inter-service communication

The services interact with each other using `gRpc`.

1.2.2 configuration

We have already configured the services to work properly, but if you need to make changes, look at the `.env` files and bootstrap folder in the `ohs-api-test` folder.

1.3 Code Test

Our team has been working on integrating with one of the biggest supplier databases in the country. In this integration, we get `csv` files from an `external` database which contains all the information that can be used to populate our application and we can in-turn show the data to our end-users.

1.3.1 Your Task

Your task is to write an integration app that reads a csv file from a defined folder (there is an attached `csv` file in `ohs-api-test/bootstrap/integration/order-integration.csv` folder). The `csv file` contains all information needed to create an order. Using this information, do the following

Information contained in each row

```
id,first_name,last_name,email,supplier_pid,credit_card_number,credit_card_type,order_id,product_pid,shipping_address,country,date_created,quantity,full_name,order_
```

1. Use the user information from each row to create a user using the user service. If the user has been saved correctly, a response is returned from where you can get a `userPid` which is then used to populate the userPid row for an order and a processed row.
2. The products already exist in the database, so you only need to add the productPid as supplied in the `order-integration.csv` when creating a new order.
3. Use the order service to create a new order and save in the database. The created order endpoint returns an order object. Embedded in this response is the `orderPid` & `supplierPid`. Use these fields to populate the `orderPid` & `supplierPid` field for your processed order row.
4. For every row processed, write out the following values (`userPid` from step1, `orderPid` from step3 and `supplierPid` from step3) from the row to a new file called `processed-orders.json`

   `example -- processed-orders.json`

```
[{
    "userPid": "string", // generated user id from user service
    "orderPid": "string", //  order id from the row
    "supplierPid": "string" // supplier id from the row
}]
```

Allowed Techstack

- A spring boot application running minumum `java 17` and `spring version: 3.2`
- A batch/integration supported by spring (`spring batch or apache camel`)
- *Optional* `reactive spring/webflux`

1.3.2 What we've provided

Protobuf Definitions

All `APIs` have been defined in `.proto` files which can be found in the `protobuf` folder.

`supplier-service`

A working version of the `supplier-service`. Configuration for this service can be found `ohs-api-test/bootstrap/supplier`

`product-service`

A working version of the `product-service`. Configuration for this service can be found `ohs-api-test/bootstrap/product`

`order-service`

A working version of the `order-service`. Configuration for this service can be found `ohs-api-test/bootstrap/order`

`user-service`

A working version of the `user-service`. Configuration for this service can be found `ohs-api-test/user.env`.

`keycloak`

A working `keycloak service` which the user-service uses to store user data

`mongo database`

A working `mongo database` which `order`, `product` and `supplier` services use to store data

`mongo express`

A working `mongo express` which can be used visualize data stored in the database. This is currently configured to run on port `8081`. You'll find the credentials in the `ohs-api-test/mongo.env`.

`jaeger`

A working `jaeger container` which can be used trace requests across `order`, `product` and `supplier`. The ui for this container currently configured to run on port `16686`

`docker-compose files`

There are 2 `docker-compose` files which can be used to start the `ohs-service`.

MAC USERS WITH THE NEWER `M1 CHIP` SHOULD USE THE `DOCKER-COMPOSE.YML` FILE LOCATED IN THE `OHS-API-TEST/MAC-M1` FOLDER.

WINDOWS, LINUX AND ALL INTEL BASED SYSTEMS USERS SHOULD USE THE `DOCKER-COMPOSE.YML` FILE LOCATED IN THE `OHS-API-TEST/INTEL` FOLDER.

You do not have to configure anything, just navigate to the `folder which matches your system` and run `docker compose up`

1.3.3 What We'd Like You To Provide

- The working app called `integration-service` in the same folder(`ohs-api-test`)
- The app should contain a working dockerfile which we can use to build your application
- All files should be shared through any source control of your choosing(`github`, `bitbucket`, `gitlab` etc)

1.4 Running the other services

1.4.1 `docker-compose files`

There are 2 `docker-compose` files which can be used to start the `ohs-service`.

MAC USERS WITH THE NEWER `M1` `CHIP` SHOULD USE THE `DOCKER-COMPOSE.YML` FILE LOCATED IN THE `OHS-API-TEST/MAC-M1` FOLDER.

WINDOWS, LINUX AND ALL INTEL BASED SYSTEMS USERS SHOULD USE THE `DOCKER-COMPOSE.YML` FILE LOCATED IN THE `OHS-API-TEST/INTEL` FOLDER.

To start up all services,

- ensure you have docker running
- navigate to the right folder (`ohs-api-test/mac-m1` for mac m1/m2 or m3 chip users) and (`ohs-api-test/intel`) for all other `arch` .

1.4.2 configuration

We have already configured the services to work properly, but if you need to make changes, look at the `.env` files and bootstrap folder in the `ohs-api-test` folder.